ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και Υλοποίηση Συστήματος Διαχείρισης ενός Χώρου Εστίασης (Restaurant Management System)

Του φοιτητή

Γιώργος Καραχρήστος

Αρ. Μητρώου: 185192

Επιβλέπων

Παναγιώτης Τζέκης

Καθηγητής

Ημερομηνία 18/01/2026

Τίτλος Δ.Ε. Σχεδιασμός και Υλοποίηση Συστήματος Διαχείρισης ενός Χώρου Εστίασης (Restaurant Management System)

Κωδικός Δ.Ε. 24154

Ονοματεπώνυμο φοιτητή/τών Γιώργος Καραχρήστος

Ονοματεπώνυμο εισηγητή Παναγιώτης Τζέκης

Ημερομηνία ανάληψης Δ.Ε. 19-03-2024

Ημερομηνία περάτωσης Δ.Ε.18-01-2026

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Η έγκριση της διπλωματικής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

«Στους γονείς μου»

Πρόλογος

Η παρούσα πτυχιακή εργασία συντάχθηκε και αναπτύχθηκε από τον φοιτητή Καραχρήστο Γεώργιο υπό την επίβλεψη του καθηγητή κ. Παναγιώτη Τζέκη. Η επιλογή του θέματος πραγματοποιήθηκε έπειτα από την εμπειρία που αποκτήθηκε κατά τη διάρκεια της πρακτικής άσκησης σε εταιρεία η οποία ανέπτυσσε το δικό της σύστημα διαχείρισης εστίασης (Restaurant Management System). Η ενασχόληση αυτή ανέδειξε τη σημασία των πληροφοριακών συστημάτων στη βελτίωση της λειτουργίας, της οργάνωσης και της αποδοτικότητας ενός χώρου εστίασης, γεγονός που αποτέλεσε το κίνητρο για την περαιτέρω εμβάθυνση στο αντικείμενο.

Σκοπός της εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός ολοκληρωμένου συστήματος διαχείρισης εστίασης, το οποίο να καλύπτει τις ανάγκες τόσο του προσωπικού όσο και της διοίκησης. Για τον λόγο αυτό αναπτύχθηκαν δύο εφαρμογές: μία για desktop υπολογιστές και μία για κινητές συσκευές Android.

Η εκπόνηση της παρούσας εργασίας συνέβαλε στην ανάπτυξη γνώσεων και δεξιοτήτων που σχετίζονται με τον σχεδιασμό, την ανάλυση και την ανάπτυξη πληροφοριακών συστημάτων, καθώς και στην καλύτερη κατανόηση των πραγματικών αναγκών ενός επιχειρησιακού περιβάλλοντος στον χώρο της εστίασης.

Περίληψη

Η παρούσα πτυχιακή εργασία έχει ως αντικείμενο τον σχεδιασμό και την υλοποίηση ενός ολοκληρωμένου Συστήματος Διαχείρισης Χώρου Εστίασης, με στόχο την αποτελεσματική υποστήριξη των καθημερινών λειτουργιών μιας σύγχρονης επιχείρησης εστίασης. Το σύστημα αναπτύχθηκε με γνώμονα την αυτοματοποίηση διαδικασιών όπως η διαχείριση παραγγελιών, προσωπικού, αποθέματος και ταμείου, καθώς και τη βελτίωση της επικοινωνίας μεταξύ των επιμέρους λειτουργικών ρόλων.

Η αρχιτεκτονική του συστήματος βασίζεται στο μοντέλο client–server και περιλαμβάνει τρία κύρια υποσυστήματα: μια desktop εφαρμογή για τη συνολική διαχείριση της επιχείρησης, μια Android εφαρμογή για φορητή και άμεση εξυπηρέτηση παραγγελιών και ταμείου, καθώς και ένα backend σύστημα υλοποιημένο σε PHP, το οποίο παρέχει υπηρεσίες μέσω RESTful API. Η αποθήκευση και διαχείριση των δεδομένων πραγματοποιείται σε σχεσιακή βάση δεδομένων MySQL, ενώ για την ανάλυση και οπτικοποίηση επιχειρησιακών δεδομένων αξιοποιήθηκαν εργαλεία της γλώσσας Python.

Κατά την υλοποίηση εφαρμόστηκαν σύγχρονα αρχιτεκτονικά πρότυπα, όπως M.V.C. και M.V.VM., εξασφαλίζοντας καθαρό διαχωρισμό της επιχειρησιακής λογικής από το γραφικό περιβάλλον και διευκολύνοντας τη συντήρηση και μελλοντική επέκταση του συστήματος. Ιδιαίτερη έμφαση δόθηκε στην ασφαλή και αξιόπιστη επικοινωνία μεταξύ των εφαρμογών και του διακομιστή, καθώς και στη διαχείριση σφαλμάτων και δικαιωμάτων πρόσβασης.

Τα αποτελέσματα της πτυχιακής δείχνουν ότι το σύστημα λειτουργεί ομαλά και ανταποκρίνεται στις λειτουργικές ανάγκες ενός χώρου εστίασης, προσφέροντας βελτιωμένη οργάνωση, ταχύτερη εξυπηρέτηση και καλύτερη αξιοποίηση των διαθέσιμων δεδομένων για τη λήψη διοικητικών αποφάσεων.

«Design and Implementation of a Restaurant Management System»

«George Karachristos»

Abstract

This thesis focuses on the design and implementation of an integrated Restaurant Management System, aiming to effectively support the daily operations of a modern food service establishment. The system was developed with the goal of automating key processes such as order management, staff management, inventory control, and cashier operations, while also improving coordination among the various functional roles within the business.

The system architecture follows the client–server model and consists of three main subsystems: a desktop application used for overall business management, an Android application that provides mobile and real-time access to order and cashier functions, and a backend system implemented in PHP that exposes services through a RESTful API. Data storage and management are handled using a MySQL relational database, while Python-based tools were utilized for data analysis and visualization of business-related statistics.

During the implementation phase, modern architectural patterns such as M.V.C. and M.V.VM. were applied, ensuring a clear separation between business logic and user interface components, and facilitating maintainability and future extensibility of the system. Special emphasis was placed on secure and reliable communication between the client applications and the server, as well as on access control mechanisms and error handling.

The results of this thesis demonstrate that the implemented system operates reliably and effectively meets the functional requirements of a restaurant environment. It contributes to improved organization, faster service, and better utilization of operational data, supporting informed decision-making and overall business efficiency.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στην οικογένειά μου για τη συνεχή στήριξη και κατανόηση που μου προσέφερε καθ' όλη τη διάρκεια των σπουδών μου. Ιδιαίτερες ευχαριστίες οφείλω στον επιβλέποντα καθηγητή μου, Παναγιώτης Τζέκης, για την πολύτιμη καθοδήγηση, τις συμβουλές και τη στήριξή του στην εκπόνηση της παρούσας πτυχιακής εργασίας. Τέλος, θα ήθελα να ευχαριστήσω τους φίλους και τους συναδέλφους μου για τη βοήθεια και την ενθάρρυνση που μου παρείχαν μέχρι την ολοκλήρωσή της.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙ.ΠΑ.Ε.	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία
R.M.S.	Restaurant Management System
Σ.Δ.Χ.Ε.	Σύστημα Διαχείρισης Χώρου Εστίασης
P.O.S.	Point Of Sale
Σ.Σ.Π.	Σύστημα Σημείου Πώλησης
A.P.I.	Application Programming Interface
U.M.L.	Unified Modeling Language
G.U.I.	Graphical User Interface
E.R. Diagram	Entity-Relationship Diagram
JS.O.N.	JavaScript Object Notation
M.V.C.	Model-View-Controller
M.V.VM	Model-View-ViewModel
HT.T.P.	HyperText Transfer Protocol
Q.R. Code	Quick Response code
RE.S.T.	Representational State Transfer

Εισαγωγή

Η παρούσα πτυχιακή εργασία εντάσσεται στο πλαίσιο της ανάπτυξης πληροφοριακών συστημάτων που στοχεύουν στη βελτίωση της διαχείρισης ενός χώρου εστίασης. Στη σύγχρονη εποχή, η τεχνολογία παίζει καθοριστικό ρόλο στην οργάνωση και στην αποτελεσματική λειτουργία των επιχειρήσεων. Οι χώροι εστίασης, έχουν ανάγκη από ολοκληρωμένα συστήματα που να υποστηρίζουν την παρακολούθηση παραγγελιών, τη διαχείριση προσωπικού, αποθεμάτων και ταμείου, καθώς και την εξαγωγή στατιστικών στοιχείων που βοηθούν στη λήψη αποφάσεων.

Σκοπός της παρούσας εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός ολοκληρωμένου Συστήματος Διαχείρισης Χώρου Εστίασης (Σ.Δ.Χ.Ε.), το οποίο θα προσφέρει λειτουργίες διαχείρισης προσωπικού, αποθέματος και παραγγελιών, αλλά και δυνατότητα εμφάνισης στατιστικών διαγραμμάτων. Το σύστημα αποτελείται από δύο εφαρμογές: μία desktop εφαρμογή, η οποία απευθύνεται στη διοίκηση και στο προσωπικό του καταστήματος, και μία Android εφαρμογή, η οποία διευκολύνει τη γρήγορη και αποδοτική διαχείριση των παραγγελιών.

Τα παραδοτέα της εργασίας περιλαμβάνουν τον σχεδιασμό και την υλοποίηση των εφαρμογών, τη δημιουργία και διαχείριση της βάσης δεδομένων, τη διασύνδεση μεταξύ των συστημάτων μέσω PHP, καθώς και την παρουσίαση στατιστικών διαγραμμάτων με χρήση Python.

Η εργασία αποτελείται από πέντε κεφάλαια. Στο πρώτο κεφάλαιο παρουσιάζεται το θεωρητικό πλαίσιο, οι βασικές έννοιες που σχετίζονται με τα συστήματα διαχείρισης εστίασης και οι υφιστάμενες λύσεις. Στο δεύτερο κεφάλαιο αναλύονται οι απαιτήσεις του συστήματος, οι ρόλοι των χρηστών καθώς και τα διαγράμματα περιπτώσεων χρήσης. Στο τρίτο κεφάλαιο περιγράφεται η διαδικασία σχεδίασης του συστήματος, περιλαμβάνοντας τα διαγράμματα ροής, την σχεδίαση της βάσης δεδομένων και την αρχιτεκτονική των εφαρμογών. Στο τέταρτο κεφάλαιο παρουσιάζεται η υλοποίηση των εφαρμογών, όπου παρουσιάζονται αναλυτικά οι τεχνολογίες που χρησιμοποιήθηκαν, η λειτουργία των εφαρμογών και η επικοινωνία τους μέσω του διακομιστή. Στο πέμπτο και τελευταίο κεφάλαιο παρουσιάζονται τα συμπεράσματα της εργασίας καθώς και προτάσεις για μελλοντικές επεκτάσεις του έργου.

Κεφάλαιο 1ο: Θεωρητικό πλαίσιο

1.1. Εισαγωγή

Το παρόν κεφάλαιο παρουσιάζει το θεωρητικό πλαίσιο που αφορά τα Συστήματα Διαχείρισης Χώρου Εστίασης (Restaurant Management Systems). Αρχικά δίνεται ο ορισμός τους και περιγράφονται οι βασικές λειτουργίες τους, ενώ στη συνέχεια αναφέρονται τα πλεονεκτήματα που προσφέρουν στις επιχειρήσεις εστίασης. Επιπλέον, γίνεται αναφορά σε ορισμένες από τις υπάρχουσες λύσεις που χρησιμοποιούνται ευρέως στην αγορά, με σκοπό την καλύτερη κατανόηση του πλαισίου στο οποίο εντάσσεται η παρούσα εργασία. Η ανάλυση αυτή αποτελεί τη βάση για το επόμενο κεφάλαιο, που αφορά την ανάλυση και σχεδίαση του συστήματος.

1.2. Ορισμός και βασικές λειτουργίες του Συστήματος Διαχείρισης Χώρου Εστίασης

Στη σύγχρονη εποχή της ψηφιακής τεχνολογίας, τα πληροφοριακά συστήματα διαχείρισης αποτελούν βασικό εργαλείο για την οργάνωση και τον έλεγχο των επιχειρησιακών διαδικασιών. Ειδικά στον χώρο της εστίασης, τα Συστήματα Διαχείρισης Χώρου Εστίασης (Restaurant Management Systems) έχουν αναδειχθεί ως απαραίτητα εργαλεία, τα οποία επιτρέπουν τον αποτελεσματικό συντονισμό λειτουργιών όπως η διαχείριση παραγγελιών, η τιμολόγηση, η διαχείριση αποθεμάτων και η ανάλυση δεδομένων.

Ένα Σ.Δ.Χ.Ε. (R.M.S.) είναι ένα ολοκληρωμένο λογισμικό που έχει σχεδιαστεί για να αυτοματοποιεί, να οργανώνει και να απλοποιεί τις διάφορες λειτουργίες ενός χώρου εστίασης. Λειτουργεί ως το κεντρικό επιχειρησιακό σύστημα μιας επιχείρησης εστίασης, ενοποιώντας σε μία ενιαία πλατφόρμα διαδικασίες όπως η διαχείριση παραγγελιών, αποθεμάτων, προσωπικού, πληρωμών και αναφορών.

Ο βασικός σκοπός ενός Σ.Δ.Χ.Ε. (R.M.S.) είναι να βελτιώσει την αποδοτικότητα, να μειώσει τα ανθρώπινα λάθη, να ενισχύσει την επικοινωνία μεταξύ των τμημάτων (π.χ. σέρβις και κουζίνα) και να αναβαθμίσει τη συνολική εμπειρία του πελάτη. Σε αντίθεση με ένα απλό Σύστημα Σημείου Πώλησης (Point Of Sale), ένα Σ.Δ.Χ.Ε. (R.M.S.) καλύπτει τόσο τις λειτουργίες του χώρου εξυπηρέτησης (front-of-house) όσο και τις λειτουργίες του παρασκηνίου (back-of-house), προσφέροντας πλήρη έλεγχο και καλύτερη εικόνα στις καθημερινές δραστηριότητες του χώρου εστίασης [1-6].

Ενδεικτικά, οι βασικές λειτουργίες ενός Σ.Δ.Χ.Ε. (R.M.S.) περιλαμβάνουν:

- Διαχείριση Παραγγελιών (Point Of Sale),
- Διαχείριση Αποθήκης / Υλικών (Inventory Management),
- Διαχείριση Προσωπικού (Staff Management),
- Διαχείριση Σχέσεων με Πελάτες και Προγράμματα Πιστότητας (Customer Relationship Management & Loyalty Programs),
- Αναφορές και Στατιστική Ανάλυση (Reporting & Analytics).

1.2.1. Διαχείριση Παραγγελιών (Point Of Sale)

Η διαχείριση των παραγγελιών μέσω του Συστήματος Σημείου Πώλησης (Point of Sale) αποτελεί τον πυρήνα ενός Σ.Δ.Χ.Ε. (R.M.S.), καθώς επιτρέπει την ομαλή λήψη, καταχώρηση και επεξεργασία των παραγγελιών σε πραγματικό χρόνο. Οι σερβιτόροι μπορούν να καταχωρούν τις παραγγελίες ψηφιακά και αυτές να αποστέλλονται αυτόματα στην κουζίνα μέσω του Συστήματος Προβολής Κουζίνας (Kitchen Display Systems), εξαλείφοντας λάθη και καθυστερήσεις. Παράλληλα, το Σ.Σ.Π. (P.O.S.)

χειρίζεται τη διαδικασία πληρωμών — είτε με κάρτες, μετρητά ή ανέπαφες συναλλαγές — ενώ τα δεδομένα πωλήσεων συγχρονίζονται αυτόματα με το υπόλοιπο σύστημα για ακριβή οικονομική παρακολούθηση. Έτσι, διασφαλίζεται ταχύτητα, ακρίβεια και καλύτερη εξυπηρέτηση πελατών [1],[5].

1.2.2. Διαχείριση Αποθήκης / Υλικών (Inventory Management)

Εξίσου σημαντική λειτουργία ενός Σ.Δ.Χ.Ε. (R.M.S.) είναι η διαχείριση της αποθήκης, η οποία βοηθά τον υπεύθυνο να γνωρίζει ανά πάσα στιγμή τα διαθέσιμα υλικά και προϊόντα. Μέσω της παρακολούθησης αποθεμάτων σε πραγματικό χρόνο, το σύστημα ενημερώνει αυτόματα τις ποσότητες που καταναλώνονται ή παραλαμβάνονται και μπορεί να εμφανίζει ειδοποιήσεις όταν κάποιο προϊόν πλησιάζει το ελάχιστο όριο. Επιπλέον, δίνει τη δυνατότητα για αυτόματη δημιουργία παραγγελιών προς τους προμηθευτές, τον έλεγχο των παραδόσεων αλλά και την πρόβλεψη μελλοντικών αναγκών. Με αυτόν τον τρόπο, μειώνεται η σπατάλη, ελέγχονται τα κόστη και διασφαλίζεται η ομαλή λειτουργία της κουζίνας χωρίς ελλείψεις [1-3],[5].

1.2.3. Διαχείριση Προσωπικού (Staff Management)

Η διαχείριση του προσωπικού μέσω ενός Σ.Δ.Χ.Ε. (R.M.S.) διευκολύνει τον έλεγχο και στην οργάνωση του ανθρώπινου δυναμικού μιας επιχείρησης εστίασης. Το σύστημα καταγράφει τα ωράρια, την παρουσία, τις επιδόσεις και τις βάρδιες των εργαζομένων, δίνοντας στους διαχειριστές μια ολοκληρωμένη εικόνα για τον τρόπο που λειτουργεί το προσωπικό. Επιπλέον, υποστηρίζει την αυτόματη δημιουργία προγραμμάτων εργασίας με βάση τη διαθεσιμότητα των εργαζομένων και τις περιόδους αιχμής, βοηθώντας έτσι στη μείωση του λειτουργικού κόστους και αποτρέποντας φαινόμενα υπερφόρτωσης. Μέσω εργαλείων μισθοδοσίας και αξιολόγησης, ενισχύεται η διαφάνεια και η αποδοτικότητα μέσα στην ομάδα [2],[5].

1.2.4. Διαχείριση Σχέσεων με Πελάτες και Προγράμματα Πιστότητας (Customer Relationship Management & Loyalty Programs)

Παράλληλα με τις εσωτερικές λειτουργίες, ένα Σ.Δ.Χ.Ε. (R.M.S.) συμβάλλει σημαντικά και στη διαχείριση των σχέσεων με τους πελάτες. Η διαχείριση σχέσεων με τους πελάτες εστιάζει στη συλλογή και ανάλυση δεδομένων πελατών, όπως προτιμήσεις, το ιστορικό κρατήσεων και παραγγελιών. Αυτά τα στοιχεία επιτρέπουν την παροχή προσωποποιημένων υπηρεσιών και τη δημιουργία προγραμμάτων πιστότητας ή επιβράβευσης — όπως εκπτώσεις, πόντοι και δώρα για ειδικές περιστάσεις. Επιπλέον, ένα Σ.Δ.Χ.Ε. (R.M.S.) μπορεί να ενσωματώνει εργαλεία marketing, τα οποία βοηθούν στη στόχευση πελατών με εξατομικευμένες καμπάνιες και προσφορές. Με αυτόν τον τρόπο, ενισχύεται η ικανοποίηση, η αφοσίωση και η επαναλαμβανόμενη πελατεία, συμβάλλοντας ουσιαστικά στη μακροχρόνια επιτυχία του χώρου εστίασης [1],[2].

1.2.5. Αναφορές και Στατιστική Ανάλυση (Reporting & Analytics)

Τέλος, η λειτουργία των αναφορών και της στατιστικής ανάλυσης παρέχει στους υπεύθυνους πολύτιμα δεδομένα για τη λήψη στρατηγικών αποφάσεων. Μέσω των εργαλείων αναφορας, ένα Σ.Δ.Χ.Ε. (R.M.S.) μπορεί να δημιουργεί αναλυτικές αναφορές σχετικά με για πωλήσεις, την απόδοση του προσωπικού, τη διαχείριση του αποθέματος, τα κέρδη αλλά και τη συμπεριφορά των πελατών. Τα δεδομένα αυτά επιτρέπουν την ανάλυση τάσεων, την πρόβλεψη μελλοντικών αναγκών και τη βελτιστοποίηση των επιχειρησιακών διαδικασιών. Η σωστή αξιοποίηση των αναλύσεων συμβάλλει

στη μείωση του κόστους, στην αύξηση της αποδοτικότητας και στη συνεχή βελτίωση της εμπειρίας του πελάτη [1-3].

Συνοψίζοντας, με την ολοκληρωμένη του δομή, ένα Σ.Δ.Χ.Ε. (R.M.S.) προσφέρει μια ενοποιημένη πλατφόρμα που βελτιώνει την αποτελεσματικότητα όλων των πτυχών μιας επιχείρησης εστίασης — από τη λήψη παραγγελιών μέχρι τη στρατηγική ανάπτυξη και τη διαχείριση πελατειακών σχέσεων.

1.3. Πλεονεκτηματα

Στην παρούσα υποενότητα παρουσιάζονται τα βασικότερα πλεονεκτήματα που προκύπτουν από την εφαρμογή ενός τέτοιου συστήματος, καθώς και ο τρόπος με τον οποίο συμβάλλει στη βελτίωση της λειτουργικής αποδοτικότητας, της εμπειρίας των πελατών και της ανταγωνιστικότητας της επιχείρησης στον κλάδο της εστίασης.

1.3.1. Εξοικονόμηση χρόνου και προσπάθειας

Η εφαρμογή ενός Σ.Δ.Χ.Ε. (R.M.S.) συμβάλλει ουσιαστικά στην εξοικονόμηση χρόνου και ανθρώπινων πόρων μέσω της αυτοματοποίησης των καθημερινών λειτουργιών. Διαδικασίες όπως η διαχείριση παραγγελιών, πληρωμών και αποθεμάτων εκτελούνται με μεγαλύτερη ταχύτητα και ακρίβεια, μειώνοντας τα περιθώρια ανθρώπινου λάθους. Ως αποτέλεσμα, το διοικητικό προσωπικό έχει τη δυνατότητα να εστιάσει σε στρατηγικές δραστηριότητες που αφορούν τη βελτίωση της ποιότητας υπηρεσιών και την ενίσχυση της ικανοποίησης των πελατών [4].

1.3.2. Αύξηση επιχειρησιακών δυνατοτήτων

Η ενσωμάτωση ενός Σ.Δ.Χ.Ε. (R.M.S.) ενισχύει σημαντικά τις λειτουργικές δυνατότητες μιας επιχείρησης εστίασης, καθώς επιτρέπει την ταχύτερη εκτέλεση παραγγελιών και τη μείωση του χρόνου αναμονής των πελατών. Μέσω της διασύνδεσης με το Σύστημα Σημείου Πώλησης (Point of Sale) και τις οθόνες κουζίνας, επιτυγχάνεται αποτελεσματικότερος συντονισμός μεταξύ του προσωπικού εξυπηρέτησης και του προσωπικού παραγωγής. Επιπλέον, το σύστημα υποστηρίζει τη διαχείριση ανθρώπινου δυναμικού και μισθοδοσίας, συμβάλλοντας στη μείωση λειτουργικών σφαλμάτων και στην αύξηση της αποδοτικότητας [4].

1.3.3. Ενοποίηση και διασύνδεση με άλλα συστήματα

Τα σύγχρονα, cloud-based Σ.Δ.Χ.Ε. (R.M.S.) μπορούν να συνδεθούν με άλλες επιχειρησιακές εφαρμογές, όπως λογιστικά συστήματα, προγράμματα διαχείρισης αποθεμάτων και συστήματα διαχείρισης πελατειακών σχέσεων. Η ενοποίηση αυτή εξασφαλίζει την ολική παρακολούθηση των λειτουργιών και τη δημιουργία ενός ενιαίου πλαισίου διαχείρισης δεδομένων, το οποίο συμβάλλει στη λήψη τεκμηριωμένων αποφάσεων και στην αύξηση της λειτουργικής συνέπειας εντός της επιχείρησης [4].

1.3.4. Αναφορές και παρακολούθηση δεδομένων

Ένα από τα σημαντικότερα πλεονεκτήματα των Σ.Δ.Χ.Ε. (R.M.S.) είναι η δυνατότητά τους να παράγουν αναλυτικές και αξιόπιστες αναφορές για τις πωλήσεις, τα έξοδα και τη συνολική οικονομική απόδοση της επιχείρησης. Μέσω εργαλείων ανάλυσης δεδομένων, οι υπεύθυνοι μπορούν να εντοπίζουν τάσεις κατανάλωσης, να αξιολογούν την αποδοτικότητα προϊόντων και να προσαρμόζουν το μενού ή τη στρατηγική λειτουργίας βάσει πραγματικών δεδομένων. Με αυτόν τον τρόπο,

ενισχύεται η ικανότητα λήψης στρατηγικών αποφάσεων βασισμένων σε αντικειμενικά στοιχεία [4], [7].

1.3.5. Διευκόλυνση στη διαφήμιση και στις στοχευμένες καμπάνιες

Η δυνατότητα που προσφέρει ένα Σ.Δ.Χ.Ε. (R.M.S.) να συλλέγει και να αναλύει δεδομένα πελατών δίνει στις επιχειρήσεις τη δυνατότητα να σχεδιάζουν πιο στοχευμένες διαφημιστικές ενέργειες.. Μέσω της ανάλυσης των προτιμήσεων και της καταναλωτικής συμπεριφοράς των πελατών, μπορούν να διαμορφωθούν εξατομικευμένες προσφορές και προγράμματα πιστότητας που ενισχύουν τη δέσμευση και ικανοποίηση των πελατών. Παράλληλα, η αξιοποίηση των δεδομένων συμβάλλει στη βελτίωση των επιδόσεων μάρκετινγκ και στην αύξηση της ανταγωνιστικότητας της επιχείρησης [4].

1.3.6. Διαχείριση πολλαπλών καταστημάτων

Τα Σ.Δ.Χ.Ε. (R.M.S.) παρέχουν τη δυνατότητα κεντρικής διαχείρισης πολλαπλών σημείων πώλησης μέσω ενός ενιαίου πληροφοριακού περιβάλλοντος. Οι υπεύθυνοι μπορούν να πραγματοποιούν άμεσα αλλαγές στο μενού, στις τιμές ή στις προσφορές για όλα τα καταστήματα, διασφαλίζοντας ομοιομορφία στην παρεχόμενη υπηρεσία και συνεκτικότητα στη στρατηγική λειτουργίας της επιχείρησης. Παράλληλα, το σύστημα διευκολύνει τη συγκριτική αξιολόγηση της απόδοσης κάθε υποκαταστήματος με στόχο τη λήψη ταχύτερων και αποτελεσματικότερων αποφάσεων [4].

1.3.7. Βελτίωση εμπειρίας πελατών

Η χρήση ενός Σ.Δ.Χ.Ε. (R.M.S.) συμβάλλει καθοριστικά στη βελτίωση της εμπειρίας των πελατών μέσω της επιτάχυνσης των διαδικασιών παραγγελίας και πληρωμής, καθώς και της μείωσης των λαθών. Η αξιοποίηση τεχνολογιών, όπως οι οθόνες πελατών και τα περίπτερα αυτοεξυπηρέτησης (self-service kiosks), προσφέρει μεγαλύτερη ευκολία και ακρίβεια στις συναλλαγές. Παράλληλα, η ενσωμάτωση προγραμμάτων επιβράβευσης πελατών ενισχύουν τη σχέση εμπιστοσύνης και συμβάλλουν στη διαμόρφωση θετικής εμπειρίας εξυπηρέτησης [4], [7].

1.3.8. Συμμόρφωση με κανονισμούς και φορολογικές απαιτήσεις

Ένα σύγχρονο Σ.Δ.Χ.Ε. (R.M.S.) μπορεί να υποστηρίζει την αυτόματη έκδοση ηλεκτρονικών τιμολογίων και να βοηθά στην τήρηση των νομικών και φορολογικών απαιτήσεων, διασφαλίζοντας ότι η επιχείρηση παραμένει σύμφωνη με το ισχύον θεσμικό πλαίσιο. Παρέχει επίσης τη δυνατότητα δημιουργίας λεπτομερών οικονομικών αναφορών, οι οποίες μπορούν να υποβληθούν στις αρμόδιες αρχές, μειώνοντας τον κίνδυνο παραβάσεων και προστίμων [4].

1.3.9. Μείωση κόστους λειτουργίας

Η αποτελεσματική διαχείριση αποθεμάτων, προσωπικού και λειτουργικών διαδικασιών μέσω ενός Σ.Δ.Χ.Ε. (R.M.S.) οδηγεί σε σημαντική μείωση του λειτουργικού κόστους. Ο ακριβής έλεγχος των προμηθειών και η αποτροπή σπατάλης πρώτων υλών επιτρέπουν την καλύτερη αξιοποίηση των πόρων και την αύξηση της κερδοφορίας. Επιπλέον, η δυνατότητα παρακολούθησης των εξόδων συμβάλλει στη βελτίωση του προϋπολογισμού και στον εντοπισμό ευκαιριών αύξησης των εσόδων [2], [7].

1.3.10. Ενίσχυση ασφάλειας δεδομένων

Η ασφάλεια των δεδομένων αποτελεί κρίσιμη παράμετρο για τις επιχειρήσεις εστίασης, ιδιαίτερα στη σύγχρονη ψηφιακή εποχή. Τα Σ.Δ.Χ.Ε. (R.M.S.), ειδικά εκείνα που βασίζονται σε υποδομές cloud, προσφέρουν προηγμένα πρωτόκολλα ασφαλείας για την προστασία των επιχειρησιακών και πελατειακών δεδομένων από επιθέσεις ή απώλειες. Η ασφαλής αποθήκευση και η ελεγχόμενη πρόσβαση στα δεδομένα διασφαλίζει την ακεραιότητα των πληροφοριών και ενισχύει την εμπιστοσύνη προς την επιχείρηση [7].

1.4. Υπαρχουσες λυσεις

Υπάρχουν πολλά και διαφορετικά συστήματα διαχείρισης χώρου εστίασης διαθέσιμα στην αγορά κάποια από τα πιο διάσημα είναι τα παρακάτω:

1.4.1. Restroworks

Σχήμα 1.1: Logo Restroworks

Το Restroworks αποτελεί ένα ολοκληρωμένο Σύστημα Διαχείρισης Χώρου Εστίασης, που συνδυάζει λειτουργικότητα, αυτοματοποίηση και ανάλυση δεδομένων με στόχο τη βελτιστοποίηση της απόδοσης. Μέσω της ενοποίησης διαδικασιών τόσο στο επίπεδο εξυπηρέτησης πελατών όσο και στη διαχείριση της κουζίνας και των αποθεμάτων, προσφέρει στους υπεύθυνους πλήρη έλεγχο στις λειτουργίες του χώρου εστίασης. Παρακάτω παρουσιάζονται συνοπτικά οι βασικές λειτουργίες του συστήματος, κατηγοριοποιημένες με βάση το τμήμα στο οποίο εφαρμόζονται [8].

Διαχείριση του χώρου εξυπηρέτησης (Front-of-House Management)

Η κατηγορία αυτή περιλαμβάνει λειτουργίες που στοχεύουν στην ενίσχυση της εμπειρίας του πελάτη μέσω καινοτόμων εργαλείων, όπως:

- Σύστημα Σημείου Πώλησης (Restaurant POS) για τη διαχείριση συναλλαγών και παραγγελιών.
- Παραγγελιοληψία μέσω Tablet (Tablet Ordering) για ταχύτερη και ακριβέστερη εξυπηρέτηση.
- Διάταξη Τραπεζιών (Table Layout) που επιτρέπει την οπτική παρακολούθηση της πληρότητας του χώρου..
- Οθόνη Εμφάνισης Παραγγελιών (Customer Order Display) για διαφάνεια και αλληλεπίδραση με τον πελάτη.
- Σύστημα Διαχείρισης Αναμονής (Queue Management) που οργανώνει κρατήσεις και ουρές σε πραγματικό χρόνο.

Διαχείριση παρασκηνίου (Back-of-House Management)

Η κατηγορία αυτή επικεντρώνεται στις εσωτερικές λειτουργίες της επιχείρησης, όπως:

- Σύστημα Διαχείρισης Αποθεμάτων (Inventory Management) για τον έλεγχο και την παρακολούθηση των πρώτων υλών.
- Σύστημα Διαχείρισης Κουζίνας (Base Kitchen Management) για συντονισμό προμηθειών και διανομών.
- Σύστημα Διαχείρισης Συνταγών (Recipe Management) για ακρίβεια στη χρήση συστατικών.
- Σύστημα Εφοδιαστικής Αλυσίδας (Supply Chain Management) και Διαχείριση Διανομών (Restaurant Delivery Management), που βελτιστοποιούν τη ροή υλικών και παραδόσεων, μειώνοντας κόστος και καθυστερήσεις.

Λειτουργίες κουζίνας (Kitchen Suite)

Το Restroworks εκσυγχρονίζει τη λειτουργία της κουζίνας μέσω εργαλείων όπως:

- Σύστημα Προβολής Κουζίνας (Kitchen Display System), το οποίο αντικαθιστά τα έντυπα δελτία με ψηφιακή ροή εργασιών.
- Σύστημα Παραγωγής Κουζίνας (Kitchen Production System) που προβλέπει τη ζήτηση και οργανώνει την παραγωγή.
- Σύστημα Πρόβλεψης Εστιατορίου (Restaurant Forecasting), το οποίο αυτοματοποιεί τον προγραμματισμό αποθεμάτων.

Οι λειτουργίες αυτές ενισχύουν την αποδοτικότητα και μειώνουν τα σφάλματα στην παραγωγή.

Αναφορές, αναλύσεις και έλεγχος (Reporting and Analytics)

Το Restroworks παρέχει προηγμένα εργαλεία παρακολούθησης και ανάλυσης δεδομένων, όπως:

- Επιχειρησιακές Αναφορές (Enterprise Reports) και Εφαρμογή Cockpit (Cockpit Application) για πρόσβαση σε δεδομένα σε πραγματικό χρόνο.
- Σύστημα Αντιμετώπισης Κλοπών (Anti-Theft Application) για την μείωση λειτουργικών απωλειών.
- Λογισμικό Αναλύσεων Εστιατορίου (Restaurant Analytics Software) για τη λήψη τεκμηριωμένων αποφάσεων και τη συνεχή βελτίωση της απόδοσης.

1.4.2. TouchBistro

Σχήμα 1.2: Logo TouchBistro

Το TouchBistro αποτελεί ένα ολοκληρωμένο Σύστημα Διαχείρισης Χώρου Εστίασης που έχει σχεδιαστεί ειδικά για τον κλάδο της εστίασης, συνδυάζοντας λειτουργικότητα, ευχρηστία και αυτοματοποίηση. Μέσω της ενσωμάτωσης εργαλείων για τη διαχείριση παραγγελιών, αποθεμάτων, προσωπικού και πελατειακών σχέσεων, συμβάλλει στη βελτίωση της αποδοτικότητας και της

εμπειρίας πελάτη. Παρακάτω παρουσιάζονται συνοπτικά οι βασικές λειτουργίες του συστήματος κατηγοριοποιημένες με βάση το τμήμα στο οποίο εφαρμόζονται [9].

Διαχείριση του χώρου εξυπηρέτησης (Front-of-House Solutions)

Το TouchBistro προσφέρει ένα ολοκληρωμένο σύνολο εργαλείων που ενισχύουν την αποδοτικότητα και την ποιότητα εξυπηρέτησης των πελατών, όπως:

- Σύστημα Σημείου Πώλησης (Point of Sale): επιτρέπει την άμεση λήψη παραγγελιών, τον διαχωρισμό λογαριασμών και την ενημέρωση του μενού σε πραγματικό χρόνο.
- Διαχείριση Τραπεζιών (Floor Plan & Table Management) και Παραγγελιοληψία στο Τραπέζι (Tableside Ordering): βελτιστοποιούν την εξυπηρέτηση, μειώνοντας καθυστερήσεις και σφάλματα.
- Οθόνη Εμφάνισης Παραγγελιών (Customer Facing Display) και Σύστημα Πληρωμών (Payments): ενισχύουν τη διαφάνεια και την ασφάλεια των συναλλαγών, συμβάλλοντας σε μια πιο ολοκληρωμένη εμπειρία πελάτη.

Διαχείριση παρασκηνίου (Back-of-House Solutions)

Οι λειτουργίες του TouchBistro για τη διαχείριση κουζίνας και αποθεμάτων στοχεύουν στην αποδοτική ροή εργασιών και τη μείωση κόστους.

- Διαχείριση Αποθεμάτων (Inventory Management): παρακολουθεί και ελέγχει τη διαθεσιμότητα υλικών.
- Διαχείριση Ανθρώπινου Δυναμικού (Labor Management): διευκολύνει τον προγραμματισμό βαρδιών και τη βελτιστοποίηση του κόστους εργασίας.
- Σύστημα Προβολής Κουζίνας (Kitchen Display System): ενισχύει την επικοινωνία μεταξύ του προσωπικού.
- Διαχείριση Κερδοφορίας (Profit Management): αυτοματοποιεί διαδικασίες τιμολόγησης και λογιστικού ελέγχου, βελτιώνοντας τη συνολική αποδοτικότητα.

Διαχείριση μενού και αναφορών (Menu, Reporting & Analytics)

Το TouchBistro επιτρέπει ευέλικτη διαχείριση περιεχομένου και ανάλυση δεδομένων σε πραγματικό χρόνο.

- Διαχείριση Μενού (Menu Management): δίνει τη δυνατότητα δημιουργίας και τροποποίησης μενού εξ αποστάσεως, προσαρμοσμένων στις ανάγκες της επιχείρησης.
- Αναφορές και Αναλύσεις (Reporting & Analytics): παρέχουν πληθώρα δεικτών και αναφορών για πωλήσεις, απόδοση προσωπικού και καταναλωτικές τάσεις.
- Ενσωματώσεις (POS Integrations): επιτρέπουν τη σύνδεση του TouchBistro με άλλες επιχειρησιακές εφαρμογές, μετατρέποντάς το σε ένα ολοκληρωμένο κέντρο επιχειρησιακής πληροφόρησης.

Διαχείριση Σχέσεων με Πελάτες (Guest Engagement Solutions)

Το TouchBistro στοχεύει στην ενίσχυση της εμπιστοσύνης και της επαναλαμβανόμενης επισκεψιμότητας των πελατών.

- Σύστημα Online Παραγγελιών (Online Ordering) και Κρατήσεις (Reservations): επιτρέπουν στους πελάτες να παραγγέλνουν ή να κάνουν κράτηση εύκολα, χωρίς μεσάζοντες.
- Προγράμματα Πιστότητας (Loyalty) και Σύστημα Μάρκετινγκ (Marketing): συμβάλλουν στη διατήρηση πελατών μέσω εξατομικευμένων προσφορών και αυτοματοποιημένων καμπανιών.
- Δωροκάρτες (Gift Cards): ενισχύουν τη ρευστότητα και λειτουργούν ως εργαλείο προώθησης και προσέλκυσης νέου κοινού.

1.4.3. Restaurant365

Σχήμα 1.3: Logo Restaurant365

Το Restaurant365 (R365) αποτελεί μια ολοκληρωμένη πλατφόρμα διαχείρισης χώρου εστίασης, που συνδυάζει λογιστική, επιχειρησιακή και διοικητική λειτουργικότητα σε ένα ενιαίο περιβάλλον. Εστιάζει στην αυτοματοποίηση διαδικασιών, στη βελτιστοποίηση του κόστους και στη λήψη αποφάσεων βάσει δεδομένων, επιτρέποντας στις επιχειρήσεις εστίασης να αυξήσουν την αποδοτικότητα και την κερδοφορία τους. Παρακάτω παρουσιάζονται συνοπτικά οι βασικές λειτουργίες του συστήματος κατηγοριοποιημένες με βάση το τμήμα στο οποίο εφαρμόζονται [10].

Οικονομική διαχείριση και λογιστική (Financial Management & Accounting)

Το Restaurant365 προσφέρει ένα ολοκληρωμένο σύστημα οικονομικής διαχείρισης που αυτοματοποιεί κρίσιμες λογιστικές διαδικασίες και ενισχύει τον έλεγχο κόστους.

- Αυτοματοποίηση Πληρωτέων Λογαριασμών (AP Automation): επιτρέπει την ασφαλή επεξεργασία τιμολογίων και πληρωμών, μειώνοντας χρόνο και σφάλματα.
- Οικονομική Αναφορά (Financial Reporting) και Τραπεζική Ενσωμάτωση (Banking Integration): προσφέρουν πλήρη ορατότητα στη ρευστότητα και στις οικονομικές επιδόσεις.
- Διαχείριση Παγίων (Fixed Asset Management) και Προϋπολογισμός & Πρόβλεψη (Budgeting & Forecasting): υποστηρίζουν τον στρατηγικό σχεδιασμό και τη βιώσιμη ανάπτυξη μέσω ανάλυσης πραγματικών δεδομένων.

Διαχείριση αποθεμάτων, προμηθειών και παραγωγής (Inventory, Purchasing & Kitchen Operations)

Οι λειτουργίες του Restaurant365 για τη διαχείριση αποθεμάτων και κουζίνας επιτρέπουν την ακριβή παρακολούθηση πόρων και τη μείωση σπατάλης.

- Διαχείριση Αποθεμάτων (Inventory Management) και Διαχείριση Συνταγών (Recipe Management): προσφέρουν έλεγχο κόστους και ακρίβεια στη χρήση πρώτων υλών.
- Οργάνωση Προετοιμασίας Κουζίνας (Kitchen Prep Software): εξασφαλίζει αποδοτική παραγωγή βάσει ζήτησης.
- Εργαλεία Αγοράς και Παραλαβών (Purchasing & Receiving Tools) και Κεντρική Παραγωγή (Commissary Management): βελτιστοποιούν τη ροή υλικών και την επικοινωνία μεταξύ πολλαπλών τοποθεσιών, ενισχύοντας τη συνοχή και την αποδοτικότητα της εφοδιαστικής αλυσίδας.

Επιχειρηματική ευφυΐα και αναφορές (Business Intelligence & Reporting)

Το Restaurant365 ενσωματώνει προηγμένα εργαλεία ανάλυσης δεδομένων που διευκολύνουν τη λήψη τεκμηριωμένων αποφάσεων βάσει πραγματικών δεικτών.

- Πλατφόρμα Επιχειρηματικής Νοημοσύνης (Business Intelligence & Reporting): επιτρέπει τη σύγκριση πωλήσεων, κόστους και επιδόσεων ανά τοποθεσία.
- Διαχείριση Απόδοσης και Κερδοφορίας (Profit Management) και Πρόβλεψη Πωλήσεων (Sales Forecasting): παρέχουν δυναμικούς πίνακες ελέγχου και προγνωστικά εργαλεία για εντοπισμό τάσεων, βελτιστοποίηση κόστους εργασίας και διατήρηση υψηλών περιθωρίων κέρδους.

Διαχείριση ανθρώπινου δυναμικού και μισθοδοσίας (Workforce, HR & Payroll Management)

Το Restaurant365 διαθέτει ένα ολοκληρωμένο σύστημα διαχείρισης ανθρώπινου δυναμικού, το οποίο καλύπτει ολόκληρο τον κύκλο ζωής του προσωπικού.

- Προγραμματισμός και Εκπαίδευση Προσωπικού (Scheduling & Training), Πρόσληψη και Ενσωμάτωση (Recruiting & Onboarding) και Διαχείριση Ανθρώπινου Δυναμικού (HR Tools): βελτιώνουν την παραγωγικότητα και μειώνουν την κινητικότητα του προσωπικού.
- Μισθοδοσία και Διαχείριση Φιλοδωρημάτων (Payroll & Tip Automation): διασφαλίζουν ακρίβεια στις πληρωμές και συμμόρφωση με τη φορολογική νομοθεσία.
- Σύστημα Διαχείρισης Εργασιών και Επικοινωνίας (Task Management & Logbook Chat): ενισχύει τη συνεργασία και την οργανωτική συνέπεια σε πραγματικό χρόνο.

1.5. Σύνοψη κεφαλαίου

Στο παρόν κεφάλαιο παρουσιάστηκε το θεωρητικό πλαίσιο που αφορά τα Συστήματα Διαχείρισης Χώρου Εστίασης (Restaurant Management Systems), τα οποία αποτελούν αναπόσπαστο εργαλείο για τη σύγχρονη λειτουργία των επιχειρήσεων εστίασης. Αρχικά, δόθηκε ο ορισμός του Σ.Δ.Χ.Ε. και αναλύθηκαν οι βασικές του λειτουργίες, όπως η διαχείριση παραγγελιών, αποθεμάτων, προσωπικού και πελατειακών σχέσεων, καθώς και οι δυνατότητες αναφορών και στατιστικής ανάλυσης που υποστηρίζουν τη λήψη στρατηγικών αποφάσεων. Στη συνέχεια, παρουσιάστηκαν αναλυτικά τα πλεονεκτήματα που προκύπτουν από την εφαρμογή ενός τέτοιου συστήματος, όπως η εξοικονόμηση χρόνου και πόρων, η βελτίωση της αποδοτικότητας και της εμπειρίας πελατών, η ενοποίηση λειτουργιών, η μείωση κόστους, η ενίσχυση της ασφάλειας δεδομένων και η συμμόρφωση με κανονισμούς και φορολογικές απαιτήσεις. Επιπλέον, αναδείχθηκε ο ρόλος των Σ.Δ.Χ.Ε. στη στρατηγική ανάπτυξη των επιχειρήσεων μέσω της αξιοποίησης δεδομένων και της αυτοματοποίησης διαδικασιών. Τέλος, εξετάστηκαν ενδεικτικά ορισμένες υπάρχουσες λύσεις της αγοράς, όπως τα Restroworks, TouchBistro και Restaurant365 (R365), οι οποίες ενσωματώνουν προηγμένες

τεχνολογίες για την ολική διαχείριση των λειτουργιών ενός χώρου εστίασης. Η ανάλυση αυτών των συστημάτων ανέδειξε την ποικιλία και την εξέλιξη των σύγχρονων Σ.Δ.Χ.Ε., που προσφέρουν ευελιξία, επεκτασιμότητα και υψηλό επίπεδο αυτοματοποίησης.

Η θεωρητική αυτή θεμελίωση αποτελεί τη βάση για το επόμενο κεφάλαιο, στο οποίο θα ακολουθήσει η ανάλυση και ο σχεδιασμός του προτεινόμενου συστήματος, με στόχο την πρακτική εφαρμογή των αρχών που παρουσιάστηκαν.

Κεφάλαιο 2ο: Ανάλυση Συστήματος

2.1. Εισαγωγή

Στο παρόν κεφάλαιο παρουσιάζεται η ανάλυση του συστήματος, η οποία αποτελεί το θεμέλιο για τη σχεδίαση και την υλοποίηση της εφαρμογής. Η ανάλυση συστήματος στοχεύει στην κατανόηση των αναγκών των χρηστών, στον καθορισμό των λειτουργικών και μη λειτουργικών απαιτήσεων, καθώς και στην αποτύπωση των ρόλων και των σεναρίων χρήσης. Το Σύστημα Διαχείρισης Εστίασης που αναπτύχθηκε στο πλαίσιο της παρούσας εργασίας αναπτύχθηκε με στόχο να καλύψει τις ανάγκες ενός σύγχρονου χώρου εστίασης, επιτρέποντας την αποτελεσματική διαχείριση παραγγελιών, αποθέματος, προσωπικού και οικονομικών δεδομένων, μέσα από ένα φιλικό και εύχρηστο περιβάλλον. Η ανάλυση αυτή αποτελεί τη βάση για το επόμενο κεφάλαιο, που αφορά την σχεδίαση του συστήματος.

2.2. Απαιτήσεις Συστήματος

Η ανάλυση των απαιτήσεων αποτελεί ένα από τα πιο κρίσιμα στάδια της ανάπτυξης λογισμικού, καθώς καθορίζει τις λειτουργίες που πρέπει να υποστηρίζει το σύστημα, καθώς και τα χαρακτηριστικά που θα εξασφαλίζουν την ομαλή και αποδοτική λειτουργία του. Στην παρούσα υποενότητα παρουσιάζονται οι απαιτήσεις του Συστήματος Διαχείρισης Χώρου Εστίασης (Restaurant Management System), οι οποίες προέκυψαν ύστερα από τη μελέτη των αναγκών ενός σύγχρονου χώρου εστίασης και των διαδικασιών που πραγματοποιούνται σε αυτόν.

Οι απαιτήσεις χωρίζονται σε δύο βασικές κατηγορίες:

1. Λειτουργικές απαιτήσεις, που περιγράφουν τις κύριες λειτουργίες που πρέπει να παρέχει το σύστημα.
2. Μη λειτουργικές απαιτήσεις, που αφορούν χαρακτηριστικά όπως η απόδοση, η ασφάλεια, η ευχρηστία και η επεκτασιμότητα του λογισμικού.

2.2.1. Λειτουργικές απαιτήσεις

Οι λειτουργικές απαιτήσεις καθορίζουν τις βασικές δυνατότητες του συστήματος και περιγράφουν τις ενέργειες που μπορούν να εκτελούν οι χρήστες του. Οι σημαντικότερες από αυτές είναι οι εξής:

1. Διαχείριση Προσωπικού: Το σύστημα πρέπει να επιτρέπει την καταχώρηση, επεξεργασία και διαγραφή στοιχείων εργαζομένων.
2. Διαχείριση Ρόλων: Το σύστημα πρέπει να υποστηρίζει τη δημιουργία, τροποποίηση και διαγραφή ρόλων. Κάθε ρόλος συνδέεται εσωτερικά με συγκεκριμένες λειτουργίες πρόσβασης του συστήματος (π.χ. ο ρόλος Σεφ έχει πρόσβαση στη λειτουργία Κουζίνα).
3. Διαχείριση Αντικειμένων (Πρώτες Ύλες & Προϊόντα): Το σύστημα πρέπει να επιτρέπει την προσθήκη, επεξεργασία και διαγραφή των πρώτων υλών και των προϊόντων της επιχείρησης.
4. Διαχείριση Κατηγοριών: Το σύστημα πρέπει να επιτρέπει την προσθήκη, επεξεργασία και διαγραφή κατηγοριών στις οποίες ομαδοποιούνται οι πρώτες ύλες και τα προϊόντα.
5. Διαχείριση Αποθέματος: Το σύστημα πρέπει να επιτρέπει την παρακολούθηση του αποθέματος, καθώς και την προσθήκη, επεξεργασία και διαγραφή εγγραφών που αφορούν την ποσότητα των διαθέσιμων αντικειμένων.

6. Διαχείριση Συναλλαγών Αποθέματος: Το σύστημα πρέπει να επιτρέπει την καταχώρηση και διαγραφή συναλλαγών που επηρεάζουν την ποσότητα του αποθέματος (π.χ. παραλαβές, καταναλώσεις).
7. Διαχείριση Πληροφοριών Επιχείρησης: Το σύστημα πρέπει να επιτρέπει την καταχώρηση και ενημέρωση βασικών πληροφοριών της επιχείρησης (π.χ. στοιχεία επικοινωνίας).
8. Προβολή Αναφορών και Στατιστικών: Το σύστημα πρέπει να δημιουργεί και να προβάλλει στατιστικα γραφήματα σχετικά με πωλήσεις, απόδοση προϊόντων και αποθέματα, ώστε να υποστηρίζει τη διαδικασία λήψης αποφάσεων.
9. Διαχείριση Παραγγελιών: Το σύστημα πρέπει να επιτρέπει τη δημιουργία, επεξεργασία, ακύρωση και αποστολή παραγγελιών προς την κουζίνα σε πραγματικό χρόνο.
10. Διαχείριση Ταμείου: Το σύστημα πρέπει να υποστηρίζει την καταγραφή πληρωμών, την παρακολούθηση συναλλαγών και την έκδοση φορολογικών αποδείξεων.
11. Διαχείριση Κουζίνας: Το σύστημα πρέπει να επιτρέπει στον υπεύθυνο κουζίνας να ενημερώνει την κατάσταση ετοιμασίας των προϊόντων ανά παραγγελία.

2.2.2. Μη Λειτουργικές Απαιτήσεις

Οι μη λειτουργικές απαιτήσεις αφορούν τις ιδιότητες και τους περιορισμούς που πρέπει να πληροί το σύστημα, ώστε να διασφαλίζεται η ποιότητα, η ασφάλεια και η ευχρηστία του. Οι σημαντικότερες από αυτές είναι οι εξής:

1. Ασφάλεια: Η πρόσβαση στα δεδομένα πρέπει να γίνεται μόνο από εξουσιοδοτημένους χρήστες μέσω προσωπικών διαπιστευτηρίων (όνομα χρήστη και κωδικός).
2. Ευχρηστία: Οι διεπαφές χρήστη πρέπει να είναι απλές, κατανοητές και εύκολες στη χρήση, ώστε να μειώνεται ο χρόνος εκπαίδευσης του προσωπικού και να διευκολύνεται η καθημερινή λειτουργία του συστήματος.
3. Διαλειτουργικότητα: Οι δύο εφαρμογές (desktop και Android) πρέπει να συνεργάζονται ομαλά, ανταλλάσσοντας δεδομένα μέσω ενιαίας βάσης δεδομένων και A.P.I., εξασφαλίζοντας συνοχή και ενημέρωση σε πραγματικό χρόνο.
4. Συντηρησιμότητα: Ο κώδικας του συστήματος πρέπει να είναι καλά δομημένος ώστε να επιτρέπει εύκολη ενημέρωση, διόρθωση σφαλμάτων και ενσωμάτωση νέων λειτουργιών.
5. Επεκτασιμότητα: Η αρχιτεκτονική του λογισμικού πρέπει να επιτρέπει την εύκολη προσθήκη νέων λειτουργιών, ρόλων ή τύπων δεδομένων χωρίς να απαιτούνται σημαντικές αλλαγές στη βασική δομή του συστήματος.

2.3. Περιγραφή Χρηστών και Ρόλων

Η επιτυχής λειτουργία ενός πληροφοριακού συστήματος εξαρτάται σε μεγάλο βαθμό από την ορθή κατανόηση των χρηστών του και των ρόλων που αυτοί εκτελούν. Στην παρούσα υποενότητα περιγράφονται οι κύριοι τύποι χρηστών του Συστήματος Διαχείρισης Χώρου Εστίασης (Restaurant Management System) και οι αντίστοιχες αρμοδιότητές τους. Ο καθορισμός των ρόλων συμβάλλει στην οργάνωση των δικαιωμάτων πρόσβασης, στην ασφάλεια των δεδομένων και στην αποτελεσματική εκτέλεση των καθημερινών λειτουργιών του χώρου εστίασης. Το σύστημα υποστηρίζει τέσσερις βασικούς ρόλους χρηστών: Διευθυντής, Ταμίας, Σερβιτόρος και Μάγειρας. Κάθε

ρόλος έχει διαφορετικές ευθύνες και επίπεδο πρόσβασης, ανάλογα με τις ανάγκες της επιχείρησης, ποιο αναλυτικά.

Ο Διευθυντής αποτελεί τον χρήστη με τις περισσότερες λειτουργίες στο σύστημα και είναι υπεύθυνος για τη συνολική διαχείριση του χώρου εστίασης. Μέσω της desktop εφαρμογής έχει τη δυνατότητα να διαχειρίζεται όλα τα κρίσιμα στοιχεία της επιχείρησης, εξασφαλίζοντας τη σωστή λειτουργία και την ακρίβεια των δεδομένων. Πιο συγκεκριμένα, ο διευθυντής μπορεί να καταχωρεί, να επεξεργάζεται και να διαγράφει στοιχεία εργαζομένων να ρυθμίζει τους ρόλους κάθε χρήστη ανάλογα με τις αρμοδιότητές του. Επιπλέον, έχει τη δυνατότητα να οργανώνει το σύνολο των προϊόντων και των κατηγοριών του μενού, τροποποιώντας ή αφαιρώντας εγγραφές όποτε αυτό απαιτείται. Ο διευθυντής είναι επίσης υπεύθυνος για τη διαχείριση του αποθέματος, συμπεριλαμβανομένων των συναλλαγών που αφορούν την προσθήκη ή αφαίρεση ποσοτήτων προϊόντων. Παράλληλα, μπορεί να ενημερώνει ή να τροποποιεί βασικές πληροφορίες της επιχείρησης, όπως στοιχεία επικοινωνίας, πληθος τραπεζιων και άλλα γενικά δεδομένα. Τέλος ο διευθυντής έχει τη δυνατότητα να παρακολουθεί στατιστικά στοιχεία και διαγράμματα πωλήσεων που παράγονται αυτόματα απο το συστημα. Με αυτόν τον τρόπο μπορεί να εξάγει πολύτιμα συμπεράσματα σχετικά με την πορεία της επιχείρησης, τις επιδόσεις του προσωπικού και τη διαχείριση των αποθεμάτων.

Ο Ταμίας είναι υπεύθυνος για τη διαχείριση των οικονομικών συναλλαγών και τη διεκπεραίωση των παραγγελιών που ολοκληρώνονται στο χώρο εστίασης. Μέσω της desktop εφαρμογής έχει τη δυνατότητα να δημιουργεί νέες παραγγελίες, να τις επεξεργάζεται ή να τις ακυρώνει όταν αυτό κρίνεται απαραίτητο, εξασφαλίζοντας την ορθή καταγραφή των πωλήσεων. Παράλληλα, εκτελεί πληρωμές, καταχωρώντας το αντίστοιχο ποσό στο ταμείο και ενημερώνοντας το σύστημα για την ολοκλήρωση της συναλλαγής. Επιπλέον, ο ταμίας εκδίδει φορολογικές αποδείξεις που αντιστοιχούν σε κάθε παραγγελία, τα οποία μπορούν να αποθηκευτούν ή να εκτυπωθούν. Έχει επίσης τη δυνατότητα να προβάλλει λίστα όλων των ενεργών παραγγελιών, ώστε να παρακολουθεί την πορεία εξυπηρέτησης των πελατών, καθώς και λίστα με τις πληρωμένες παραγγελίες. Ο ρόλος του ταμία είναι κρίσιμος για την οικονομική οργάνωση της επιχείρησης, καθώς εξασφαλίζει την ακρίβεια των οικονομικών δεδομένων και τη σωστή ενημέρωση του συστήματος σε πραγματικό χρόνο.

Ο Σερβιτόρος χρησιμοποιεί κυρίως την Android εφαρμογή του συστήματος, μέσω της οποίας πραγματοποιεί τη διαχείριση των παραγγελιών που προέρχονται από τα τραπέζια του χώρου εστίασης. Έχει τη δυνατότητα να δημιουργεί νέες παραγγελίες, να τις επεξεργάζεται καθώς και να τις ακυρώνει σε περίπτωση λάθους ή αλλαγής από τον πελάτη. Με τον τρόπο αυτό εξασφαλίζεται η άμεση και ακριβής καταγραφή των παραγγελιών στο σύστημα. Επιπλέον, ο σερβιτόρος μπορεί να προβάλλει τη λίστα όλων των ενεργών παραγγελιών, ώστε να παρακολουθεί την κατάσταση κάθε τραπεζιού. Όταν ολοκληρώνεται η εξυπηρέτηση ενός πελάτη, ο σερβιτόρος έχει επίσης τη δυνατότητα να εκτελεί την πληρωμή μέσω της εφαρμογής, ενημερώνοντας το ταμείο και ολοκληρώνοντας τη διαδικασία συναλλαγής. Ο ρόλος του σερβιτόρου είναι ουσιαστικός για την ομαλή λειτουργία του χώρου εστίασης, καθώς αποτελεί τον συνδετικό κρίκο μεταξύ των πελατών, της κουζίνας και του ταμείου, διασφαλίζοντας την ταχύτητα και την ακρίβεια στην εξυπηρέτηση.

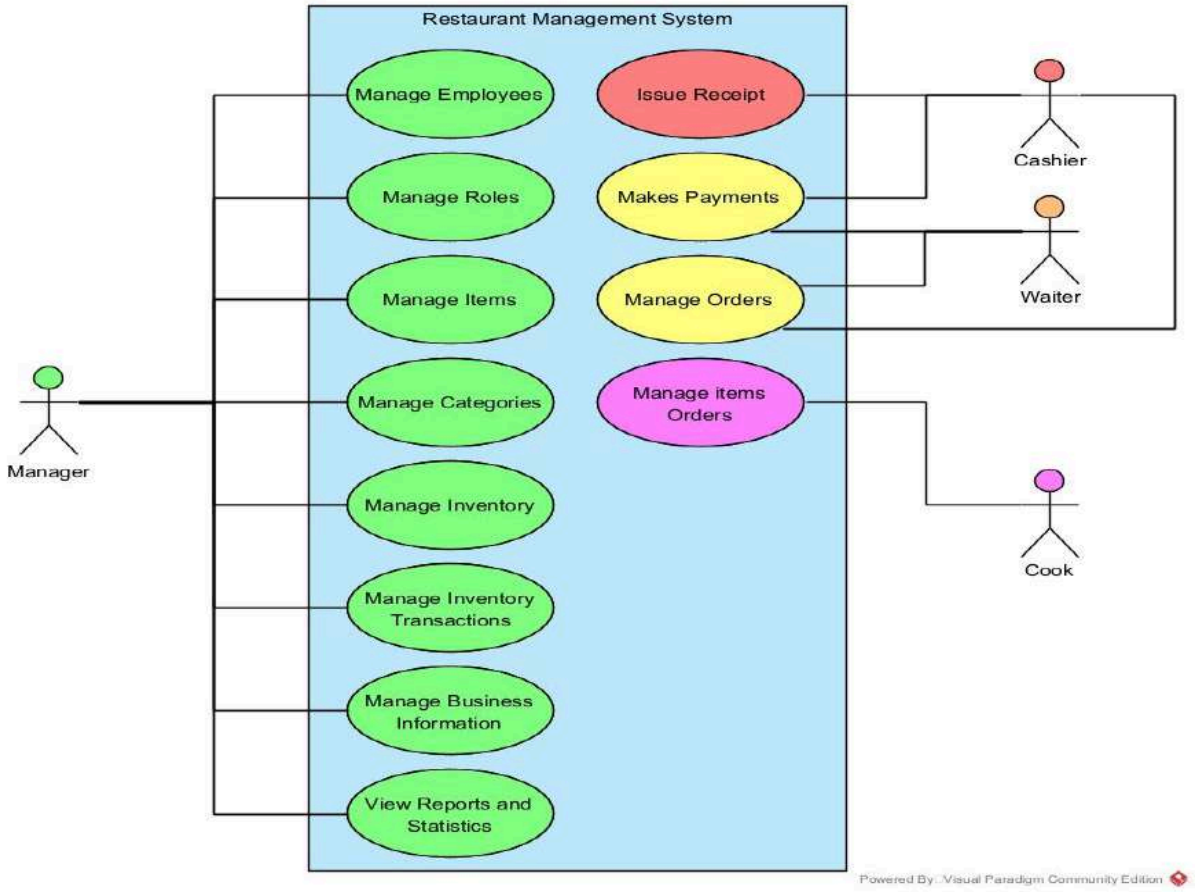
Ο Μάγειρας είναι υπεύθυνος για την παρακολούθηση και εκτέλεση των παραγγελιών που αποστέλλονται από το προσωπικό εξυπηρέτησης. Μέσω της desktop εφαρμογής έχει τη δυνατότητα να προβάλλει τη λίστα όλων των ενεργών παραγγελιών, οι οποίες εμφανίζονται οργανωμένες ανά χρονική σειρά, διευκολύνοντας τη σωστή προτεραιοποίηση της προετοιμασίας των πιάτων. Κατά τη διάρκεια της παρασκευής, ο μάγειρας μπορεί να ενημερώνει την κατάσταση κάθε πιάτου. Με αυτόν τον τρόπο, το προσωπικό εξυπηρέτησης ενημερώνεται σε πραγματικό χρόνο για την πρόοδο των

παραγγελιών, εξασφαλίζοντας την ομαλή συνεργασία μεταξύ κουζίνας και σερβιτόρων. Ο ρόλος του μάγειρα συμβάλλει καθοριστικά στην αποδοτική ροή εργασιών του χώρου εστίασης,, μειώνοντας τον χρόνο αναμονής των πελατών και ενισχύοντας τον συντονισμό ανάμεσα στα τμήματα του καταστήματος.

2.4. Διάγραμμα Περιπτώσεων Χρήσης (Use Case Diagram)

Στην παρούσα υποενότητα παρουσιάζεται το Διάγραμμα Περιπτώσεων Χρήσης (Use Case Diagrams) του συστήματος. Το διάγραμμα αυτό αποτελεί μέρος της Ενοποιημένης Γλώσσας Σχεδίασης Προτύπων (Unified Modeling Language), η οποία χρησιμοποιείται για τον καθορισμό, την απεικόνιση, την κατασκευή και την τεκμηρίωση των στοιχείων ενός λογισμικού συστήματος. Μέσω της U.M.L. είναι δυνατή η γραφική αναπαράσταση της δομής και της λειτουργίας ενός συστήματος, συμβάλλοντας έτσι στην καλύτερη κατανόηση και ανάλυσή του [11]. Ένα από τα βασικότερα διαγράμματα της U.M.L. είναι το Διάγραμμα Περιπτώσεων Χρήσης, το οποίο παρέχει μια γραφική επισκόπηση των στόχων που οι χρήστες (actors) επιθυμούν να επιτύχουν μέσω της αλληλεπίδρασής τους με το σύστημα. Το διάγραμμα αυτό αποτυπώνει τις κύριες λειτουργίες που προσφέρει το σύστημα και τις σχέσεις που υπάρχουν μεταξύ αυτών και των διαφορετικών τύπων χρηστών [12].

Για τη δημιουργία του διαγράμματος χρησιμοποιήθηκε το λογισμικό Visual Paradigm Community Edition, το οποίο παρέχει πλήρη υποστήριξη για τη μοντελοποίηση U.M.L. και διευκολύνει την απεικόνιση των σχέσεων μεταξύ χρηστών και λειτουργιών του συστήματος.

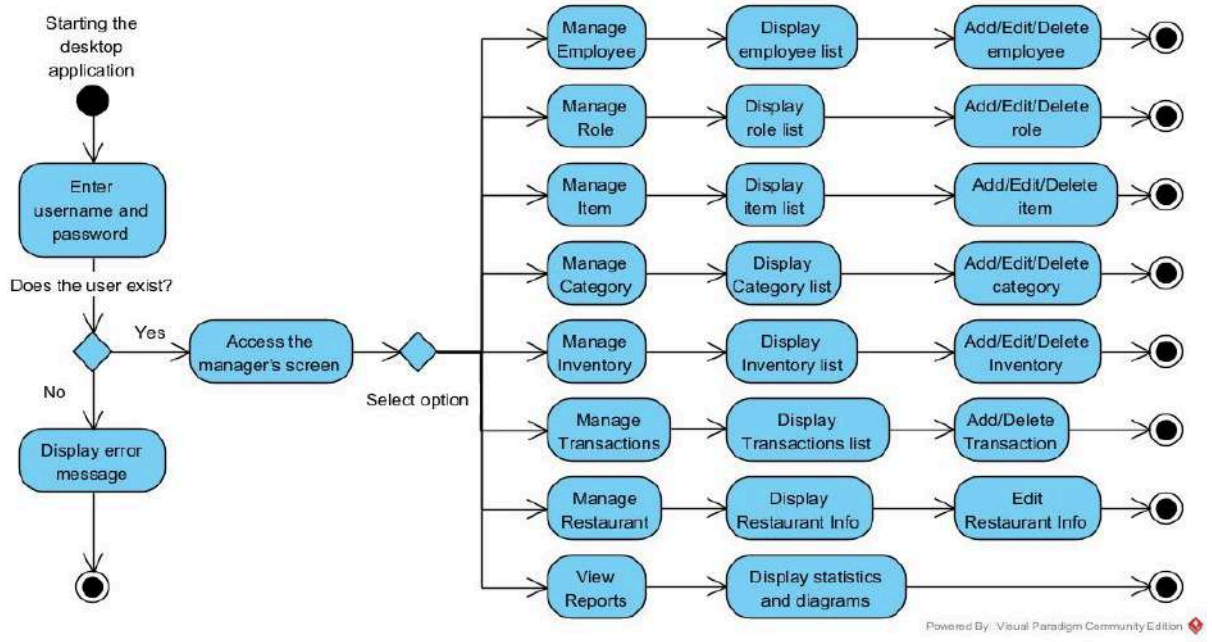


Σχήμα 2.1: Συνολικό Διάγραμμα Περιπτώσεων Χρήσης

Παραπάνω παρουσιάζεται το Συνολικό Διάγραμμα Περιπτώσεων Χρήσης, το οποίο απεικονίζει όλους τους ρόλους του συστήματος καθώς και τις λειτουργίες που είναι διαθέσιμες σε καθέναν από αυτούς. Οι βασικοί ρόλοι του συστήματος είναι ο Διευθυντής, ο Ταμίας, ο Σερβιτόρος και ο Μάγειρας. Η αποτύπωση αυτή αποτέλεσε τη βάση για την περιγραφή του διαγράμματος δραστηριότητας, που ακολουθούν στην επόμενη υποενότητα.

2.5. Διαγράμματα δραστηριότητας (Activity diagram)

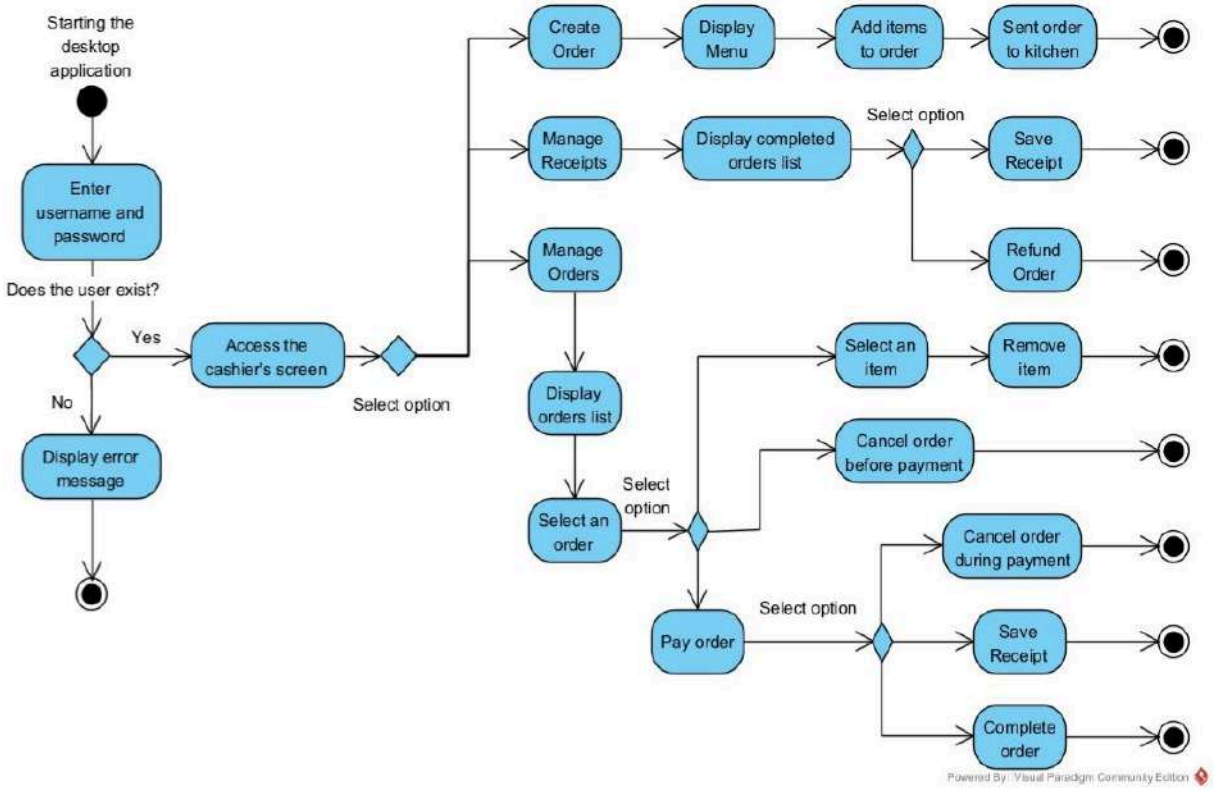
Στην παρούσα υποενότητα παρουσιάζονται τα Διαγράμματα Δραστηριότητας (Activity Diagrams) του κάθε ρόλου του συστήματος. Ο συγκεκριμένος τύπος διαγράμματος αποτελεί μέρος της Ενοποιημένης Γλώσσας Σχεδίασης Προτύπων (Unified Modeling Language) και χρησιμοποιείται για την αναπαράσταση της ροής των ενεργειών μέσα σε ένα σύστημα. Μέσω των διαγραμμάτων δραστηριότητας αποτυπώνονται οι διαδικασίες που εκτελούνται είτε από τους χρήστες είτε από το ίδιο το σύστημα, καθώς και οι πιθανές διαδρομές που μπορεί να ακολουθηθούν ανάλογα με τις συνθήκες. Τα διαγράμματα αυτά συμβάλλουν στην κατανόηση της λογικής ροής των λειτουργιών και διευκολύνουν τη μετάβαση από το στάδιο της ανάλυσης στο στάδιο της σχεδίασης και της υλοποίησης. Για τη δημιουργία των διαγραμμάτων χρησιμοποιήθηκε το λογισμικό Visual Paradigm Community Edition.



Σχήμα 2.2: Διάγραμμα Δραστηριότητας Διευθυντή

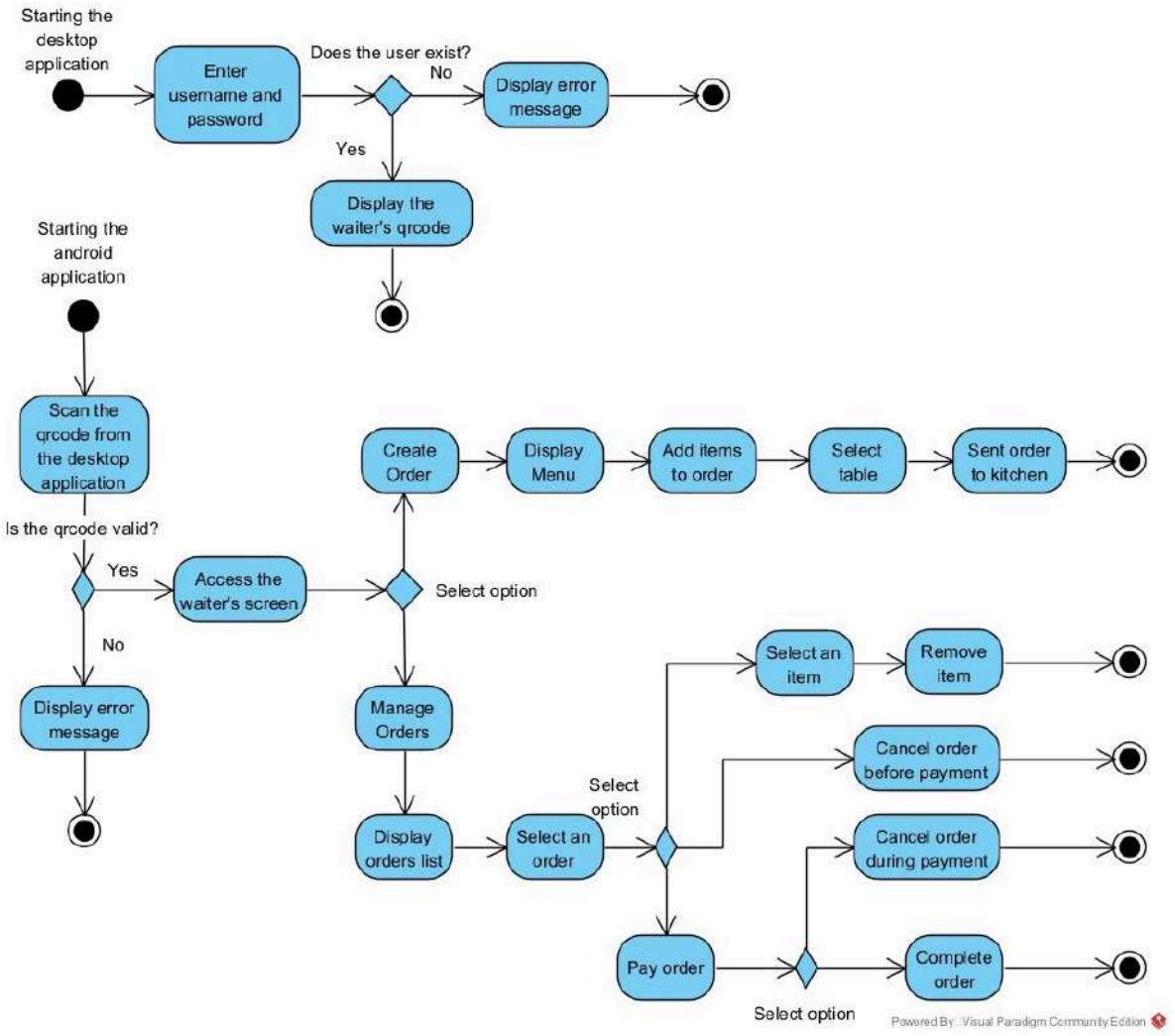
Παραπάνω παρουσιάζεται το διάγραμμα δραστηριότητας του διευθυντή, το οποίο απεικονίζει τη ροή των κύριων λειτουργιών που εκτελεί ο διευθυντής κατά τη χρήση της εφαρμογής. Αρχικά, ο χρήστης αφου ταυτοποιηθεί από το σύστημα και επιβεβαιωθεί ότι διαθέτει λογαριασμό και ότι κατέχει τον ρόλο του διευθυντή του εμφανίζεται η κατάλληλη οθόνη. Μέσα από αυτή την οθόνη, ο διευθυντής μπορεί να διαχειριστεί το προσωπικό, τους ρόλους, τα αντικείμενα (πρώτες ύλες και προϊόντα), τις κατηγορίες, το απόθεμα, τις συναλλαγές του αποθέματος καθώς και τις πληροφορίες του χώρου εστίασης. Για κάθε ενότητα, παρουσιάζεται η προβολή των αντίστοιχων δεδομένων, μαζί με τις

διαθέσιμες ενέργειες επεξεργασίας, όπως προσθήκη, τροποποίηση ή διαγραφή. Τέλος, μέσω της ίδιας οθόνης ο διευθυντής μπορεί να παρακολουθεί στατιστικά στοιχεία και διαγράμματα πωλήσεων.



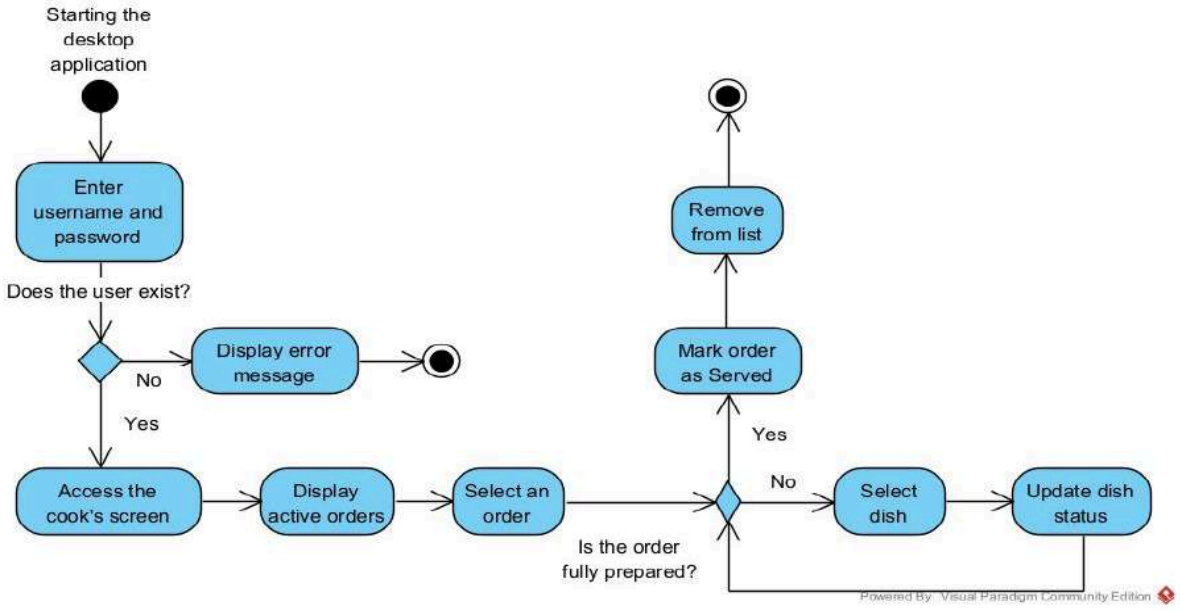
Σχήμα 2.3: Διάγραμμα Δραστηριότητας Ταμία

Παραπάνω παρουσιάζεται το διάγραμμα δραστηριότητας του ταμία, το οποίο απεικονίζει τη ροή των κύριων λειτουργιών που εκτελεί ο χρήστης κατά τη χρήση της εφαρμογής. Αρχικά, ο χρήστης αφου ταυτοποιηθεί από το σύστημα και επιβεβαιωθεί ότι διαθέτει λογαριασμό και ότι κατέχει τον ρόλο του ταμία του εμφανιζεται η κατάλληλη οθόνη. Μέσα από αυτή την οθόνη, ο ταμίας μπορεί να δημιουργεί νέες παραγγελίες μέσω του μενού προϊόντων, να προσθέσει αντικείμενα και να αποστείλει την παραγγελία στην κουζίνα. Παράλληλα, έχει τη δυνατότητα να διαχειρίζεται ολοκληρωμένες παραγγελίες, να εκδίδει ή να αποθηκεύει αποδείξεις, καθώς και να πραγματοποιεί επιστροφές (refunds). Επιπλέον, ο ταμίας μπορεί να διαχειρίζεται τις ενεργές παραγγελίες, να επιλέγει συγκεκριμένη παραγγελία, να αφαιρεί αντικείμενα, να ακυρώνει παραγγελίες πριν ή κατά τη διάρκεια της πληρωμής, ή να προχωρά στην ολοκλήρωση της πληρωμής και την έκδοση της αντίστοιχης απόδειξης.



Σχήμα 2.4: Διάγραμμα Δραστηριότητας Σερβιτόρου

Παραπάνω παρουσιάζεται το διάγραμμα δραστηριότητας του σερβιτόρου, το οποίο απεικονίζει τη ροή των κύριων λειτουργιών που εκτελεί ο χρήστης κατά τη χρήση της εφαρμογής. Αρχικά, ο χρήστης συνδέεται στην desktop εφαρμογή και, εφόσον ταυτοποιηθεί από το σύστημα, λαμβάνει έναν μοναδικό κωδικό QR. Στη συνέχεια, μέσω της Android εφαρμογής, ο σερβιτόρος σαρώνει τον κωδικό και, εφόσον αυτός είναι έγκυρος, αποκτά πρόσβαση στην κεντρική οθόνη της εφαρμογής. Από εκεί μπορεί να δημιουργήσει νέα παραγγελία, να επιλέξει προϊόντα από το μενού, να προσθέσει αντικείμενα και να ορίσει το τραπέζι πριν αποστείλει την παραγγελία στην κουζίνα. Παράλληλα, έχει τη δυνατότητα να διαχειρίζεται τις ενεργές παραγγελίες, να επιλέγει μία συγκεκριμένη παραγγελία, να αφαιρεί αντικείμενα, να ακυρώνει την παραγγελία πριν ή κατά τη διάρκεια της πληρωμής, καθώς και να ολοκληρώσει την πληρωμή και την καταχώρηση της παραγγελίας.



Σχήμα 2.5: Διάγραμμα Δραστηριότητας Μάγειρα

Παραπάνω παρουσιάζεται το διάγραμμα δραστηριότητας του μάγειρα, το οποίο απεικονίζει τη ροή των κύριων λειτουργιών που εκτελεί ο χρήστης κατά τη χρήση της εφαρμογής. Αρχικά, ο χρήστης αφου ταυτοποιηθεί από το σύστημα και επιβεβαιωθεί ότι διαθέτει λογαριασμό και ότι κατέχει τον ρόλο του μάγειρα του εμφανίζεται η λίστα των ενεργών παραγγελιών. Στην συνέχεια για καθε μια παραγγελία ο μάγειρας ελέγχει αν έχει ολοκληρωθεί η προετοιμασία της. Στην περίπτωση που η παραγγελία δεν είναι πλήρως έτοιμη, ο μάγειρας επιλέγει το αντίστοιχο πιάτο και ενημερώνει την κατάστασή του. Όταν όλα τα πιάτα της παραγγελίας έχουν ολοκληρωθεί, αυτή σημειώνεται ως “Served” και αφαιρείται από τη λίστα των ενεργών παραγγελιών.

2.6. Σύνοψη κεφαλαίου

Στο παρόν κεφάλαιο πραγματοποιήθηκε η αναλυτική περιγραφή του Συστήματος Διαχείρισης Χώρου Εστίασης, με στόχο την αποτύπωση των απαιτήσεων και της λειτουργικής του δομής. Αρχικά παρουσιάστηκαν οι λειτουργικές και μη λειτουργικές απαιτήσεις, οι οποίες καθόρισαν τις βασικές δυνατότητες του συστήματος και τα ποιοτικά χαρακτηριστικά που πρέπει αυτό να πληροί. Στη συνέχεια, περιγράφηκαν οι χρήστες και οι ρόλοι τους, αναλύοντας τις αρμοδιότητες του Διευθυντή, του Ταμία, του Σερβιτόρου και του Μάγειρα.

Ακολούθως, μέσω του Διαγράμματος Περιπτώσεων Χρήσης αποτυπώθηκαν γραφικά οι βασικές λειτουργίες και η αλληλεπίδραση των χρηστών με το σύστημα, ενώ τα Διαγράμματα Δραστηριότητας ανέδειξαν τη ροή των ενεργειών που εκτελούνται από κάθε ρόλο. Συνολικά, το κεφάλαιο αυτό παρείχε μία ολοκληρωμένη εικόνα της λειτουργικής ανάλυσης του συστήματος, αποτελώντας τη βάση για το επόμενο στάδιο, που αφορά τη σχεδίαση και υλοποίηση της εφαρμογής.

Κεφάλαιο 3ο: Σχεδίαση Συστήματος

3.1. Εισαγωγή

Το κεφάλαιο αυτό ασχολείται με τη σχεδίαση του συστήματος, παρουσιάζοντας τη δομή και τον τρόπο λειτουργίας του. Αρχικά περιγράφεται η αρχιτεκτονική του συστήματος και τα τρία βασικά επίπεδα λειτουργίας: το επίπεδο διεπαφής χρήστη, το επίπεδο λογικής εφαρμογής και το επίπεδο δεδομένων. Στη συνέχεια παρουσιάζονται τα διαγράμματα κλάσεων, τα οποία δείχνουν τη στατική δομή του λογισμικού και τις σχέσεις μεταξύ των κλάσεων. Τέλος, αναλύεται η βάση δεδομένων και οι πίνακές της, καθώς και οι σχέσεις τους μέσω του Διαγράμματος Οντοτήτων–Συσχετίσεων (Entity-Relationship Diagram), παρέχοντας μια ολοκληρωμένη εικόνα για τη λειτουργία και την οργάνωση του συστήματος.

3.2. Αρχιτεκτονική Συστήματος

Η αρχιτεκτονική λογισμικού περιγράφει τη συνολική δομή ενός συστήματος, δηλαδή τον τρόπο με τον οποίο αυτό οργανώνεται σε επιμέρους αλληλεπιδρώντα μέρη, τις βασικές οδούς επικοινωνίας μεταξύ τους, καθώς και τις κύριες ιδιότητές τους και τις ιδιότητες του συστήματος ως σύνολο. Αποτελεί μια αφηρημένη, υψηλού επιπέδου αναπαράσταση που λειτουργεί ως γέφυρα ανάμεσα στις απαιτήσεις και την υλοποίηση, επιτρέποντας τον κριτικό έλεγχο, την ανάλυση και την αξιολόγηση της σχεδιαστικής λύσης. Με άλλα λόγια, η αρχιτεκτονική λειτουργεί ως «σχέδιο» (blueprint) για την κατασκευή του συστήματος, καθοδηγεί τις αποφάσεις σχεδίασης και αναδεικνύει τις σημαντικές πτυχές, ενώ ταυτόχρονα αποκρύπτει λεπτομέρειες της υλοποίησης [13].

Ένα από τα πιο διαδεδομένα πρότυπα οργάνωσης κατανεμημένων συστημάτων είναι η αρχιτεκτονική client–server. Στο μοντέλο αυτό, πολλοί υπολογιστές-πελάτες (clients) υποβάλλουν αιτήματα σε έναν κεντρικό διακομιστή (server), ο οποίος επεξεργάζεται τα αιτήματα και επιστρέφει τις αντίστοιχες απαντήσεις ή υπηρεσίες. Οι clients αποτελούν το περιβάλλον αλληλεπίδρασης με τον χρήστη, ενώ ο server αναλαμβάνει την επεξεργασία, τον συντονισμό και την πρόσβαση στα δεδομένα. Ένα βασικό χαρακτηριστικό της client–server αρχιτεκτονικής είναι ότι οι λεπτομέρειες του υποκείμενου συστήματος (υλικό, λογισμικό, βάση δεδομένων) αποκρύπτονται από τον client, ο οποίος βλέπει μια τυποποιημένη και «διαφανή» υπηρεσία. Το μοντέλο αυτό είναι ιδιαίτερα αποτελεσματικό όταν client και server εκτελούν διαφορετικές, εξειδικευμένες λειτουργίες [14].

Στο πλαίσιο της παρούσας εργασίας, το προτεινόμενο σύστημα υιοθετεί μια αρχιτεκτονική τύπου three-tier client–server. Η three-tier αρχιτεκτονική αποτελεί ένα καθιερωμένο μοντέλο αρχιτεκτονικής λογισμικού, στο οποίο η εφαρμογή οργανώνεται σε τρία διακριτά επίπεδα (tiers), τόσο σε λογικό όσο και σε φυσικό επίπεδο. Ο διαχωρισμός αυτός επιτρέπει καλύτερη οργάνωση, ανάπτυξη, συντήρηση και κλιμάκωση της εφαρμογής, καθώς κάθε επίπεδο μπορεί να αναπτυχθεί, να τροποποιηθεί ή να επεκταθεί ανεξάρτητα από τα υπόλοιπα, υπό την προϋπόθεση ότι διατηρούνται οι μεταξύ τους διεπαφές.

Τα τρία επίπεδα μιας τυπικής three-tier αρχιτεκτονικής είναι το presentation tier, το application tier και το data tier. Το presentation tier αποτελεί τη διεπαφή χρήστη και το σημείο αλληλεπίδρασης με την εφαρμογή. Κύριος ρόλος του είναι να εμφανίζει πληροφορίες στον χρήστη και να συλλέγει δεδομένα από αυτόν, χωρίς να εμπλέκεται στην επεξεργασία ή την αποθήκευση των δεδομένων. Το application tier, ή επίπεδο λογικής, θεωρείται η «καρδιά» της εφαρμογής, καθώς εκεί υλοποιείται η επιχειρησιακή

λογική (business logic) δηλαδή στο επίπεδο αυτό γίνεται η επεξεργασία των δεδομένων, εφαρμόζονται κανόνες, γίνονται έλεγχοι εγκυρότητας και λαμβάνονται αποφάσεις σχετικά με τις λειτουργίες του συστήματος. Τέλος, το data tier είναι υπεύθυνο για την αποθήκευση και τη διαχείριση των δεδομένων της εφαρμογής και συνήθως υλοποιείται μέσω ενός συστήματος διαχείρισης βάσεων δεδομένων [15].

Η αρχιτεκτονική του υπό ανάπτυξη Restaurant Management System εντάσσεται σε αυτό το μοντέλο ως εξής:

3.2.1. Presentation tier

Στο επίπεδο αυτό ανήκουν η desktop εφαρμογή και η Android εφαρμογή. Και οι δύο εφαρμογές παρέχουν γραφικό περιβάλλον διεπαφής (G.U.I.) προς τον χρήστη, μέσω του οποίου μπορεί να εκτελέσει διάφορες λειτουργίες, όπως η καταχώρηση παραγγελιών, η διαχείριση του ταμείου, η ενημέρωση του αποθέματος και η προβολή πληροφοριών σχετικά με τον χώρο εστίασης. Το tier αυτό είναι υπεύθυνο αποκλειστικά για την παρουσίαση και την εισαγωγή δεδομένων, χωρίς να έχει άμεση πρόσβαση στη βάση δεδομένων.

3.2.2. Application tier

Στο επίπεδο ανήκει ο server-side κώδικας σε PHP, ο οποίος υλοποιεί την επιχειρησιακή λογική του συστήματος και λειτουργεί ως ενδιάμεσος μεταξύ των εφαρμογών-πελατών και της βάσης δεδομένων. Το application tier δέχεται αιτήματα από τη desktop και την Android εφαρμογή (π.χ. δημιουργία νέας παραγγελίας, ενημέρωση αποθέματος, έλεγχος στοιχείων χρήστη), τα επεξεργάζεται σύμφωνα με τους κανόνες της εφαρμογής και στη συνέχεια επικοινωνεί με το επίπεδο δεδομένων για ανάγνωση ή τροποποίηση πληροφοριών. Επιπλέον, στο επίπεδο αυτό εντάσσεται και η παραγωγή στατιστικών, καθώς ο server είναι υπεύθυνος για τη συγκέντρωση, την επεξεργασία και την προετοιμασία των αναγκαίων δεδομένων.

3.2.3. Data tier

Στο επίπεδο δεδομένων βρίσκεται η βάση δεδομένων, η οποία αποθηκεύει όλες τις πληροφορίες που σχετίζονται με το προσωπικό, τα προϊόντα, τις παραγγελίες, το απόθεμα και τις οικονομικές συναλλαγές του χώρου εστίασης. Η πρόσβαση στη βάση δεδομένων γίνεται αποκλειστικά μέσω του application tier, γεγονός που ενισχύει την ασφάλεια και τη συνέπεια των δεδομένων. Επιπλέον, τα Python scripts έχουν πρόσβαση στο επίπεδο αυτό για την εξαγωγή των απαραίτητων στοιχείων και τη δημιουργία στατιστικών διαγραμμάτων, χωρίς να επηρεάζουν τη βασική λειτουργία της εφαρμογής.

Με τον τρόπο αυτό, η υλοποίηση βασίζεται σε μια ξεκάθαρη three-tier client–server αρχιτεκτονική, όπου η διάκριση των ευθυνών ανάμεσα στα επίπεδα συμβάλλει στην ευκολότερη συντήρηση, στην επεκτασιμότητα του συστήματος και στην πιθανή μελλοντική ενσωμάτωση επιπλέον διεπαφών (π.χ. web εφαρμογή) χωρίς την ανάγκη ριζικών αλλαγών στη λογική ή στη δομή των δεδομένων.

3.3. Διάγραμμα κλάσεων (Class Diagram)

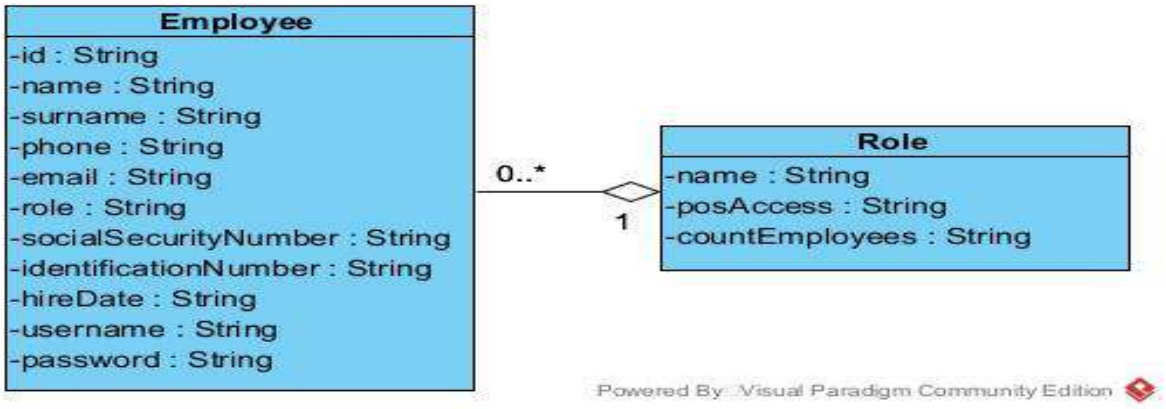
Στην παρούσα υποενότητα παρουσιάζονται τα Διαγράμματα Κλάσεων (Class Diagrams) του συστήματος. Ο συγκεκριμένος τύπος διαγράμματος αποτελεί μέρος της Ενοποιημένης Γλώσσας Σχεδίασης Προτύπων (Unified Modeling Language – U.M.L.) και είναι διάγραμμα στατικής δομής

που περιγράφει τη δομή ενός συστήματος, εμφανίζοντας τις κλάσεις, τα χαρακτηριστικά (attributes), τις μεθόδους (operations) τους και τις σχέσεις μεταξύ τους. Το διάγραμμα κλάσεων αποτυπώνει τη στατική δομή των κατηγοριοποιητων (classifiers) ενός συστήματος και παρέχει τη βάση για άλλα διαγράμματα δομής σύμφωνα με την U.M.L. Αποτελεί χρήσιμο εργαλείο τόσο για προγραμματιστές και μέλη ομάδων ανάπτυξης, όσο και για αναλυτές επιχειρησιακών διαδικασιών, οι οποίοι μπορούν να μοντελοποιούν το σύστημα από επιχειρησιακή σκοπιά.

Ένα διάγραμμα κλάσεων αποτελείται από τα εξής στοιχεία:

- Συλλογή κλάσεων, που περιγράφουν ομάδες αντικειμένων με παρόμοιους ρόλους στο σύστημα. Κάθε κλάση περιλαμβάνει:
    - Δομικά χαρακτηριστικά (attributes), τα οποία περιγράφουν τι «γνωρίζουν» τα αντικείμενα της κλάσης και αναπαριστούν την κατάσταση του αντικειμένου.
    - Χαρακτηριστικά συμπεριφοράς (operations), τα οποία καθορίζουν τι «μπορούν να κάνουν» τα αντικείμενα της κλάσης και πώς αλληλεπιδρούν μεταξύ τους.
- Σχέσεις μεταξύ κλάσεων, που δείχνουν πώς οι κλάσεις συνδέονται και συνεργάζονται (συσχέτιση, συσχέτιση πολλαπλότητας, εξάρτηση, συσσωμάτωση/aggregation, σύνθεση/composition, κληρονομικότητα/generalization).

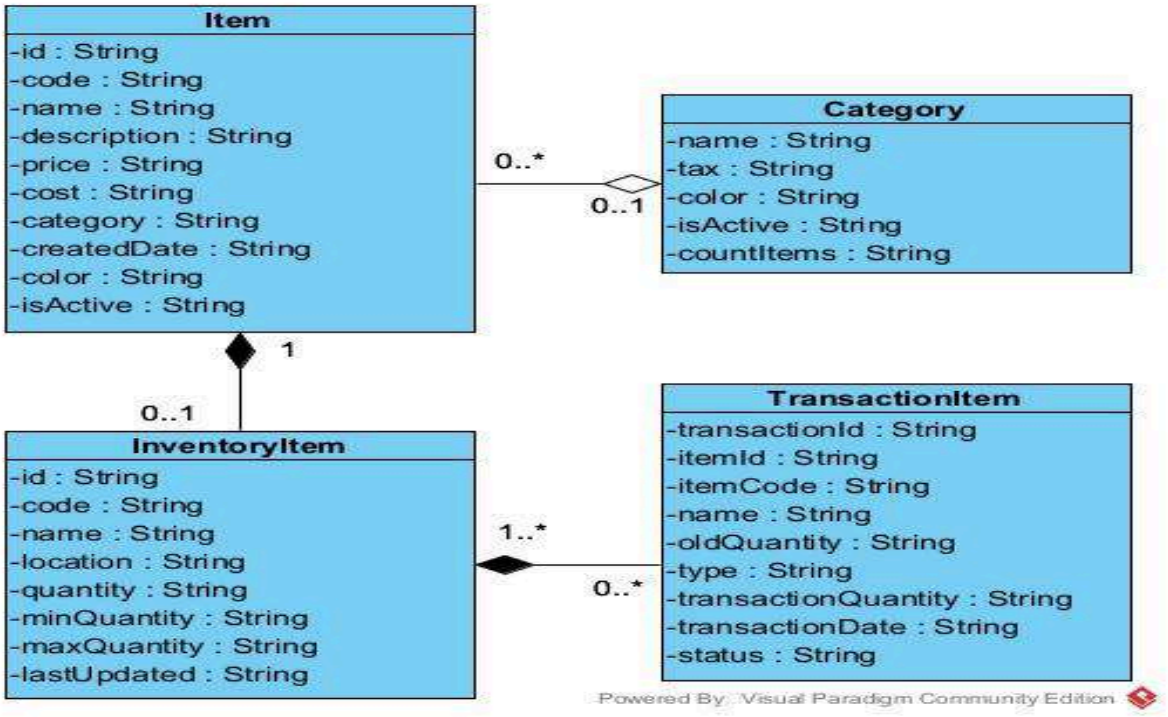
Συνοπτικά, το διάγραμμα κλάσεων επιτρέπει την κατανόηση και τεκμηρίωση της δομής ενός συστήματος και λειτουργεί ως βάση για την ανάπτυξη και την υλοποίηση του λογισμικού [16]. Για τη δημιουργία των παρακατων διαγραμμάτων χρησιμοποιήθηκε το λογισμικό Visual Paradigm Community Edition.



Σχήμα 3.1: Διάγραμμα κλάσεων Υπαλλήλων

Το παραπάνω διάγραμμα κλάσεων παρουσιάζει τη δομή και τις συσχετίσεις των οντοτήτων που αφορούν τη διαχείριση του προσωπικού και τα δικαιώματα πρόσβασης. Το διάγραμμα περιλαμβάνει δύο βασικές κλάσεις: την κλάση Employee (Υπάλληλος), η οποία αναπαριστά τους εργαζομένους που χρησιμοποιούν την εφαρμογή, και την κλάση Role (Ρόλος), η οποία ορίζει τους διαθέσιμους ρόλους και τα αντίστοιχα επίπεδα πρόσβασης. Οι δύο κλάσεις συνδέονται με μια σχέση συσσωμάτωσης (aggregation), η οποία περιγράφει μια σχέση «Whole–Part»: ο ρόλος λειτουργεί ως οντότητα που ομαδοποιεί πολλούς υπαλλήλους. Η επιλογή της συσσωμάτωσης υποδηλώνει ότι η ύπαρξη ενός υπαλλήλου δεν εξαρτάται αποκλειστικά από τον ρόλο δηλαδή, ένας υπάλληλος μπορεί να αλλάξει ρόλο, ή ένας ρόλος μπορεί να διαγραφεί χωρίς να διαγραφούν αυτόματα οι συνδεδεμένοι υπάλληλοι. Η πολλαπλότητα (multiplicity) της σχέσης ορίζει ότι ένας ρόλος μπορεί να αντιστοιχεί σε μηδέν ή περισσότερους υπαλλήλους (0..*), ενώ κάθε υπάλληλος συνδέεται υποχρεωτικά με ακριβώς ένα ρόλο

(1). Η επιλογή αυτής της μοντελοποίησης αντιπροσωπεύει μια ρεαλιστική δομή για συστήματα διαχείρισης χώρων εστίασης, όπου οι ρόλοι είναι προκαθορισμένοι και κάθε υπάλληλος εντάσσεται σε έναν συγκεκριμένο τύπο λειτουργίας (π.χ. Ταμίας, Μάγειρας, Σερβιτόρος, Διευθυντής).



Σχήμα 3.2: Διάγραμμα κλάσεων Προϊόντων

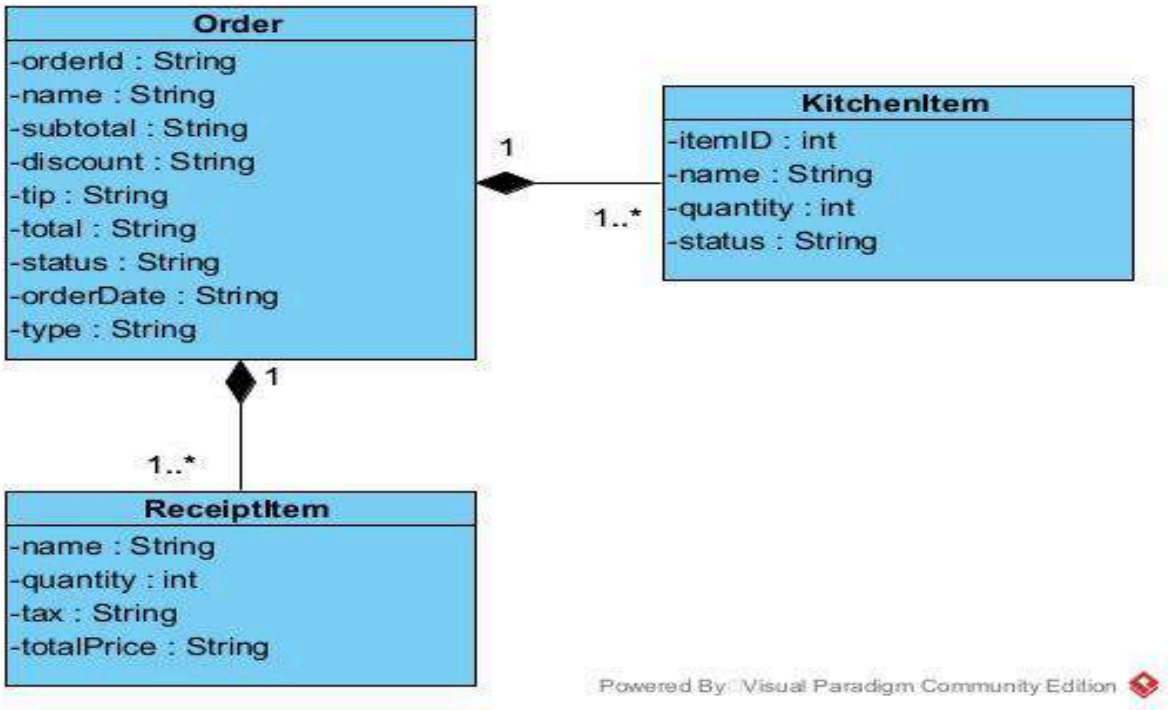
Το παραπάνω διάγραμμα κλάσεων παρουσιάζει την αρχιτεκτονική του υποσυστήματος που είναι υπεύθυνο για τη διαχείριση των προϊόντων, την παρακολούθηση του αποθέματος και την καταγραφή του ιστορικού συναλλαγών. Το μοντέλο αποτελείται από τέσσερις βασικές κλάσεις: την κλάση Item (Προϊόν), η οποία αποτελεί την κεντρική οντότητα αναφοράς για τα προϊόντα της επιχείρησης, την κλάση Category (Κατηγορία), η οποία χρησιμοποιείται για την ομαδοποίηση των προϊόντων, την κλάση InventoryItem (Εγγραφή Αποθέματος), η οποία αποθηκεύει πληροφορίες σχετικά με την κατάσταση αποθήκευσης των προϊόντων, και την κλάση TransactionItem (Συναλλαγή Αποθέματος), η οποία λειτουργεί ως αρχείο καταγραφής για τις κινήσεις των αποθεμάτων.

Οι σχέσεις μεταξύ των κλάσεων αναδεικνύουν την επιχειρησιακή λογική του υποσυστήματος. Μεταξύ της κλάσης Category και της κλάσης Item υπάρχει σχέση συσσωμάτωσης (aggregation), η οποία δηλώνει ότι τα προϊόντα μπορούν να ομαδοποιούνται σε κατηγορίες, χωρίς όμως η ύπαρξη ενός προϊόντος να εξαρτάται αποκλειστικά από την κατηγορία του. Η πολλαπλότητα της συσχέτισης ορίζει ότι μία κατηγορία μπορεί να περιέχει από μηδέν έως πολλά προϊόντα (0..*), ενώ κάθε προϊόν μπορεί να ανήκει το πολύ σε μία κατηγορία (1).

Μεταξύ των κλάσεων Item και InventoryItem παρατηρείται σχέση σύνθεσης (composition), η οποία εκφράζει μια ισχυρή εξάρτηση και ιδιοκτησία. Η εγγραφή αποθέματος δεν έχει νόημα ύπαρξης χωρίς το αντίστοιχο προϊόν συνεπώς, αν διαγραφεί το προϊόν, διαγράφονται αυτόματα και οι σχετικές εγγραφές αποθέματος. Η πολλαπλότητα αποτυπώνει ότι ένα προϊόν μπορεί να διαθέτει καμία ή μία εγγραφή αποθέματος (0..1), ενώ κάθε εγγραφή αποθέματος αντιστοιχεί σε ένα συγκεκριμένο προϊόν (1).

Αντίστοιχα, σύνθεση υπάρχει και μεταξύ των κλάσεων InventoryItem και TransactionItem. Κάθε συναλλαγή αποθέματος εξαρτάται από την ύπαρξη της αντίστοιχης εγγραφής αποθέματος, γεγονός που σημαίνει ότι αν διαγραφεί η εγγραφή αποθέματος, διαγράφονται και οι συναλλαγές που την αφορούν. Η πολλαπλότητα υποδεικνύει ότι μία εγγραφή αποθέματος μπορεί να συσχετίζεται με καμία ή περισσότερες συναλλαγές (0..*), ενώ κάθε εγγραφή συναλλαγής πρέπει να αντιστοιχεί υποχρεωτικά σε μία εγγραφή αποθέματος (1).

Συνολικά, η παραπάνω μοντελοποίηση αποτυπώνει μια ρεαλιστική και συνεκτική επιχειρησιακή λογική: οι κατηγορίες οργανώνουν τα προϊόντα χωρίς να δημιουργούν δεσμευτικό δεσμό ύπαρξης, ενώ οι εγγραφές αποθέματος και οι σχετικές συναλλαγές συνδέονται στενά με τα προϊόντα, εξασφαλίζοντας την ακεραιότητα και τη συνέπεια των δεδομένων που αφορούν τη διαχείριση του αποθέματος.



Σχήμα 3.3: Διάγραμμα κλάσεων Παραγγελιών

Το παρόν διάγραμμα κλάσεων παρουσιάζει την αρχιτεκτονική του υποσυστήματος που είναι υπεύθυνο για τη διαχείριση, την εκτέλεση και την οικονομική τακτοποίηση των παραγγελιών. Το μοντέλο περιστρέφεται γύρω από την κεντρική κλάση Order (Παραγγελία), η οποία διασυνδέεται με δύο εξειδικευμένες κλάσεις: την KitchenItem (Προϊόντα Κουζίνας) και την ReceiptItem (Προϊόντα Απόδειξης). Η κλάση Order αποτελεί την κεντρική οντότητα του διαγράμματος και συγκεντρώνει όλες τις πληροφορίες που αφορούν μια παραγγελία στο σύνολό της. Η κλάση KitchenItem αναπαριστά τα επιμέρους προϊόντα της παραγγελίας που έχουν προωθηθεί στην κουζίνα για παρασκευή, ενώ η κλάση ReceiptItem αποτυπώνει τα προϊόντα της παραγγελίας όπως αυτά εμφανίζονται στην απόδειξη πληρωμής.

Οι σχέσεις στο διάγραμμα χαρακτηρίζονται από υψηλό βαθμό εξάρτησης, διασφαλίζοντας την ακεραιότητα των δεδομένων. Μεταξύ της Order και της KitchenItem υφίσταται σχέση σύνθεσης (composition), που σημαίνει πως τα είδη κουζίνας αποτελούν αναπόσπαστα μέρη της παραγγελίας· δεν νοείται ύπαρξη KitchenItem χωρίς την αντίστοιχη Order. Σε περίπτωση διαγραφής της

παραγγελίας διαγράφονται αυτόματα και τα συνδεδεμένα KitchenItems. Η πολλαπλότητα (1 → 1..*) υποδηλώνει ότι κάθε παραγγελία πρέπει να περιλαμβάνει τουλάχιστον ένα προϊόν προς παρασκευή, ενώ κάθε KitchenItem ανήκει αποκλειστικά σε μία παραγγελία.

Αντίστοιχη είναι η σχέση σύνθεσης μεταξύ της Order και της ReceiptItem: τα προϊόντα της απόδειξης υφίστανται υπαρξιακή εξάρτηση από την παραγγελία, δηλαδή ένα ReceiptItem έχει νόημα μόνο εντός του πλαισίου της συγκεκριμένης συναλλαγής. Η πολλαπλότητα διασφαλίζει ότι κάθε παραγγελία συνοδεύεται από τουλάχιστον ένα προϊόν χρέωσης και ότι κάθε ReceiptItem συνδέεται αποκλειστικά με μία παραγγελία.

Συνολικά, η παραπάνω μοντελοποίηση αποτυπώνει με σαφήνεια την επιχειρησιακή λογική του υποσυστήματος: οι λεπτομέρειες της παραγγελίας, της προετοιμασίας και της χρέωσης οργανώνονται με τρόπο που διασφαλίζει την ακεραιότητα των δεδομένων και διευκολύνει την επεξεργασία, την ακύρωση ή την οικονομική τακτοποίηση των παραγγελιών.

3.4. Βάση δεδομένων – Σχεδίαση και Διάγραμμα E.R.

Η βάση δεδομένων αποτελεί ένα από τα πιο κρίσιμα υποσυστήματα της εφαρμογής, καθώς είναι υπεύθυνη για την αποθήκευση, την οργάνωση και την αξιόπιστη διαχείριση των πληροφοριών που απαιτούνται για τη λειτουργία του συστήματος. Η σχεδίασή της πραγματοποιήθηκε με βάση τις λειτουργικές και μη λειτουργικές απαιτήσεις, με στόχο την εξασφάλιση συνοχής, ακεραιότητας και αποδοτικότητας κατά την ανάκτηση και ενημέρωση των δεδομένων. Η τελική δομή της βάσης δεδομένων αποτελείται από έντεκα πίνακες, οι οποίοι παρουσιάζονται και περιγράφονται αναλυτικά παρακάτω.

Ο πίνακας EMPLOYEES αποθηκεύει τα βασικά στοιχεία των εργαζομένων της επιχείρησης και χρησιμοποιείται τόσο για τη διαχείριση του προσωπικού όσο και για τη διαδικασία ελέγχου πρόσβασης στο σύστημα. Περιλαμβάνει το μοναδικό αναγνωριστικό EMPLOYEE_ID τύπου int, το οποίο αποτελεί το πρωτεύον κλειδί του πίνακα και χρησιμοποιείται για την διαχείριση της εγγραφής στη βάση δεδομένων. Τα προσωπικά στοιχεία των εργαζομένων αποθηκεύονται στα πεδία NAME (Όνομα) και SURNAME (Επώνυμο), ενώ τα στοιχεία επικοινωνίας καταγράφονται στα πεδία PHONE (Τηλέφωνο) και EMAIL (Ηλεκτρονικη Διευθυνση). Ο πίνακας περιλαμβάνει επίσης διοικητικά στοιχεία, όπως το SOCIAL_SECURITY_NUMBER (Αριθμός Κοινωνικής Ασφάλισης) και το IDENTIFICATION_NUMBER (Αριθμός Ταυτότητας), τα οποία είναι μοναδικά για κάθε εργαζόμενο, διασφαλίζοντας ότι δεν μπορούν να υπάρξουν διπλοεγγραφές. Για την αυθεντικοποίηση των χρηστών, αποθηκεύονται τα πεδία USERNAME και PASSWORD. Αξίζει να σημειωθεί ότι ο κωδικός πρόσβασης δεν αποθηκεύεται σε απλή μορφή κειμένου, αλλά ως κρυπτογραφημένο hash μέσω της μεθόδου SHA-2, ενισχύοντας την ασφάλεια του συστήματος. Το πεδίο ROLE αποτελεί ξένο κλειδί το οποίο συνδέεται με τον πίνακα ROLE, καθορίζοντας τα δικαιώματα και τις αρμοδιότητες του εργαζομένου μέσα στο σύστημα. Τέλος, το πεδίο HIRE_DATE τύπου timestamp αποθηκεύει την ημερομηνία πρόσληψης και συμπληρώνεται αυτόματα κατά την εισαγωγή της εγγραφής.

Ο πίνακας ROLE χρησιμοποιείται για την αντιστοίχιση των ρόλων που εμφανίζονται στους χρήστες της εφαρμογής με τους εσωτερικούς ρόλους που χρησιμοποιεί το σύστημα για τον έλεγχο πρόσβασης. Με αυτόν τον τρόπο επιτυγχάνεται η ανεξαρτησία ανάμεσα στους λειτουργικούς ρόλους που κατανοεί ο χρήστης (όπως Διευθυντής, Ταμίας ή Σερβιτόρος) και στους τεχνικούς ρόλους που χρησιμοποιεί το σύστημα για τη διαχείριση δικαιωμάτων. Το πεδίο NAME (Όνομα) αποτελεί το πρωτεύον κλειδί του πίνακα και αντιπροσωπεύει το όνομα του ρόλου όπως εμφανίζεται στον χρήστη. Το πεδίο POS_ACCESS λειτουργεί ως ξένο κλειδί και συνδέεται με τον πίνακα POS_ROLES, στον οποίο

βρίσκονται οι συστημικοί ρόλοι και τα αντίστοιχα επίπεδα πρόσβασης που παρέχουν. Η δομή αυτή επιτρέπει, για παράδειγμα, ο ρόλος Διευθυντής όπως εμφανίζεται στην εφαρμογή να αντιστοιχίζεται σε έναν εσωτερικό ρόλο —π.χ. Cook ή Manager— που καθορίζει τα πραγματικά δικαιώματα που έχει ο χρήστης στο σύστημα. Η αντιστοίχιση αυτή παρέχει ευελιξία στη διαχείριση των δικαιωμάτων και επιτρέπει την τροποποίηση ή επέκταση των ρόλων χωρίς αλλαγές στο υπόλοιπο σύστημα.

Ο πίνακας POS_ROLES αποτελεί τον βασικό μηχανισμό καθορισμού των εσωτερικών ρόλων πρόσβασης του συστήματος. Σε αντίθεση με τους ρόλους που εμφανίζονται στον χρήστη, οι οποίοι μπορεί να προσαρμόζονται ανάλογα με τις ανάγκες της επιχείρησης, οι ρόλοι στον πίνακα POS_ROLES αντιπροσωπεύουν σταθερές και τεχνικά ορισμένες κατηγορίες δικαιωμάτων. Το πεδίο NAME (Όνομα) αποτελεί το πρωτεύον κλειδί και περιέχει το όνομα κάθε ρόλου του συστήματος. Κάθε τιμή σε αυτό το πεδίο συνδέεται με τα αντίστοιχα επίπεδα πρόσβασης που υπαρχουν. Ο πίνακας αυτός συνδέεται με τον πίνακα ROLE, στον οποίο αντιστοιχίζονται οι φιλικοί προς τον χρήστη ρόλοι της εφαρμογής. Έτσι, οι POS_ROLES λειτουργούν ως ο χαμηλού επιπέδου μηχανισμός ελέγχου πρόσβασης, ενώ οι ρόλοι στον πίνακα ROLE λειτουργούν ως η «προβολή» που βλέπει ο χρήστης. Η διάκριση αυτή επιτρέπει την προσαρμοστικότητα της εφαρμογής και την ασφαλή διαχείριση δικαιωμάτων.

Ο πίνακας RESTAURANT αποθηκεύει τις βασικές πληροφορίες που αφορούν το ίδιο το κατάστημα εστίασης και χρησιμοποιείται για την παραμετροποίηση της εφαρμογής. Περιλαμβάνει δεδομένα που είναι απαραίτητα για την ταυτοποίηση της επιχείρησης, καθώς και στοιχεία που σχετίζονται με τη λειτουργία της. Το πεδίο NAME (Όνομα) καταγράφει την επωνυμία του καταστήματος, ενώ το πεδίο ADDRESS (Διεύθυνση) αποθηκεύει τη διεύθυνση στην οποία βρίσκεται ο χώρος εστίασης. Το πεδίο PHONE (Τηλέφωνο) χρησιμοποιείται για την καταγραφή του τηλεφωνικού αριθμού επικοινωνίας της επιχείρησης, ο οποίος μπορεί να αξιοποιηθεί σε λειτουργίες όπως εξυπηρέτηση πελατών ή διαχείριση κρατήσεων. Το πεδίο TABLES (Τραπεζια), τύπου int, αποθηκεύει τον αριθμό των διαθέσιμων τραπεζιών στον χώρο. Η πληροφορία αυτή είναι κρίσιμη γιατι την χρησιμοποιεί ο σερβιτορος για να αναθεση παραγγελιων.

Ο πίνακας WAITER_TOKENS χρησιμοποιείται για τη διαχείριση των διακριτικών πρόσβασης (tokens) που εκχωρούνται στους σερβιτόρους ή γενικότερα στους εργαζομένους που συνδέονται στο σύστημα μέσω της Android εφαρμογής. Το πεδίο EMPLOYEE_ID λειτουργεί ως πρωτεύον κλειδί και ταυτόχρονα ως ξένο κλειδί που συνδέεται με τον πίνακα EMPLOYEES, επιτρέποντας την αντιστοίχιση κάθε token με έναν συγκεκριμένο υπάλληλο. Το πεδίο TOKEN αποθηκεύει το κρυπτογραφημένο διακριτικό το οποίο χρησιμοποιείται για την αυθεντικοποίηση του χρήστη κατά την επικοινωνία της εφαρμογής με τον server. Το πεδίο CREATED_AT καταγράφει την ακριβή χρονική στιγμή κατά την οποία δημιουργήθηκε το token. Το EXPIRES_AT καθορίζει το χρονικό σημείο λήξης του token, διασφαλίζοντας ότι η πρόσβαση του χρήστη έχει χρονικό περιορισμό για λόγους ασφάλειας. Η ύπαρξη καθορισμένης ημερομηνίας λήξης μειώνει τον κίνδυνο κατάχρησης ή μη εξουσιοδοτημένης πρόσβασης στην περίπτωση που ένα token παραβιαστεί ή μείνει ενεργό για μεγάλο χρονικό διάστημα. Συνολικά, ο πίνακας WAITER_TOKENS αποτελεί κρίσιμο στοιχείο για την υλοποίηση ενός ασφαλούς μηχανισμού session management, μειώνοντας τον κίνδυνο μη εξουσιοδοτημένης πρόσβασης και διασφαλίζοντας την ορθή λειτουργία της διαδικασίας login στην Android εφαρμογή.

Ο πίνακας ITEMS αποτελεί έναν από τους βασικότερους πίνακες της βάσης δεδομένων, καθώς αποθηκεύει όλα τα διαθέσιμα προϊόντα που προσφέρει το κατάστημα εστίασης. Τα προϊόντα αυτά μπορεί να περιλαμβάνουν φαγητά, ποτά, αναψυκτικά ή οποιοδήποτε άλλο είδος πωλείται μέσω του

συστήματος. Ο πίνακας χρησιμοποιείται τόσο για τη διαδικασία παραγγελιοληψίας όσο και για τη διαχείριση αποθέματος και τιμολογιακής πολιτικής. Το πεδίο ITEM_ID τύπου int είναι το πρωτεύον κλειδί και λειτουργεί ως μοναδικό αναγνωριστικό κάθε προϊόντος, διευκολύνοντας την εσωτερική διαχείρισή του. Το ITEM_CODE αποτελεί έναν μοναδικό κωδικό προϊόντος, το πεδίο NAME (Όνομα) αποθηκεύει το όνομα του προϊόντος και είναι επίσης μοναδικό, ώστε να αποφεύγονται διπλοεγγραφές. Το πεδίο DESCRIPTION (Περιγραφή) χρησιμοποιείται για την περιγραφή του προϊόντος και μπορεί να περιλαμβάνει πρόσθετες πληροφορίες, όπως συστατικά ή χαρακτηριστικά. Τα πεδία PRICE (τιμή πώλησης) και COST (τιμή κόστους) τύπου decimal αποθηκεύουν την τελική τιμή πώλησης και το κόστος παραγωγής αντίστοιχα, επιτρέποντας τον υπολογισμό του κέρδους και την υποστήριξη λειτουργιών ανάλυσης και στατιστικών. Το πεδίο CATEGORY (Κατηγορια) αποτελεί ξένο κλειδί και συνδέεται με τον πίνακα CATEGORY, κατατάσσοντας κάθε προϊόν στη σωστή κατηγορία (π.χ. Ποτά, Κυρίως Πιάτα, Ορεκτικά). Το CREATED_AT τύπου timestamp καταγράφει την ημερομηνία και ώρα δημιουργίας της εγγραφής, παρέχοντας ιστορική πληροφόρηση. Το πεδίο COLOR (Χρώμα) αποθηκεύει χρωματικό κωδικό σε μορφή HEX, ο οποίος χρησιμοποιείται για την οπτική αναπαράσταση του προϊόντος στην εφαρμογή. Τέλος, το πεδίο IS_ACTIVE δηλώνει αν το προϊόν είναι ενεργό ή όχι, επιτρέποντας την απενεργοποίηση παλαιών ή μη διαθέσιμων ειδών χωρίς τη διαγραφή τους. Ο πίνακας ITEMS αποτελεί κεντρικό σημείο αναφοράς για το σύστημα, καθώς συνδέεται τόσο με λειτουργίες παραγγελιών όσο και με τη διαχείριση της επιχειρησιακής πληροφορίας.

Ο πίνακας CATEGORY αποθηκεύει τις κατηγορίες προϊόντων του καταστήματος και χρησιμοποιείται για την ομαδοποίηση, τιμολόγηση και οπτική αναπαράσταση των ειδών στο σύστημα. Το πεδίο NAME λειτουργεί ως πρωτεύον κλειδί και αποτελεί το μοναδικό αναγνωριστικό κάθε κατηγορίας. Το πεδίο TAX καταγράφει τον συντελεστή φόρου που εφαρμόζεται στα προϊόντα της κατηγορίας, υποστηρίζοντας τον σωστό υπολογισμό των τιμών και των φορολογικών εγγραφών. Το πεδίο COLOR αποθηκεύει έναν χρωματικό κωδικό (μορφή HEX) που χρησιμοποιείται στην παρουσίαση του μενού και στις διεπαφές POS για γρήγορη οπτική διάκριση των κατηγοριών. Το πεδίο IS_ACTIVE δηλώνει αν η κατηγορία είναι ενεργή και κατά συνέπεια αν μπορεί να εμφανιστεί στο μενού της εφαρμογής. Ο πίνακας CATEGORY συνδέεται με τον πίνακα ITEMS μέσω ξένου κλειδιού, επιτρέποντας την αξιόπιστη κατηγοριοποίηση των προϊόντων και διευκολύνοντας λειτουργίες αναζήτησης, φιλτραρίσματος και αναφορών.

Ο πίνακας INVENTORY διαχειρίζεται τα αποθέματα των προϊόντων του συστήματος, προσφέροντας μια ολοκληρωμένη εικόνα της διαθεσιμότητας κάθε είδους. Το πεδίο ITEM_ID λειτουργεί ως πρωτεύον κλειδί και ταυτόχρονα ως ξένο κλειδί προς τον πίνακα ITEMS, εξασφαλίζοντας ότι κάθε εγγραφή αποθέματος αντιστοιχεί σε ένα υπαρκτό προϊόν. Το πεδίο LOCATION καθορίζει τη φυσική τοποθεσία αποθήκευσης (π.χ. κεντρική αποθήκη, κουζίνα, μπαρ), επιτρέποντας την παρακολούθηση αποθεμάτων ανά χώρο. Τα πεδία QUANTITY, MIN_QUANTITY και MAX_QUANTITY τυπου int αποτυπώνουν την τρέχουσα ποσότητα, το ελάχιστο και το μέγιστο επιτρεπτό όριο για κάθε είδος, ώστε το σύστημα να μπορεί να ανιχνεύει ελλείψεις ή υπερβολικές ποσότητες. Το πεδίο LAST_UPDATED ενημερώνεται αυτόματα σε κάθε αλλαγή, παρέχοντας ιστορικό για τον χρόνο της τελευταίας τροποποίησης. Μέσω της σχέσης με τον πίνακα ITEMS και των ορίων ποσότητας, ο πίνακας INVENTORY προσφέρει αξιόπιστη και δυναμική παρακολούθηση των αποθεμάτων, υποστηρίζοντας κρίσιμες λειτουργίες όπως αναπλήρωση, έλεγχο αποθήκης και διαχείριση πόρων.

Ο πίνακας TRANSACTIONS καταγράφει όλες τις μεταβολές που πραγματοποιούνται στα αποθέματα, λειτουργώντας ως ιστορικό κινήσεων για κάθε είδος. Το πεδίο TRANSACTION_ID αποτελεί το πρωτεύον κλειδί και δημιουργείται αυτόματα, εξασφαλίζοντας μοναδικότητα σε κάθε συναλλαγή. Το

ITEM_ID αποτελεί ξένο κλειδί προς τον πίνακα INVENTORY και καθορίζει σε ποιο προϊόν αφορά η κίνηση. Το πεδίο OLD_QUANTITY αποθηκεύει την ποσότητα που υπήρχε πριν την αλλαγή, ενώ το TRANSACTION_QUANTITY καταγράφει την ποσότητα που προστέθηκε ή αφαιρέθηκε, επιτρέποντας την ακριβή παρακολούθηση της διακύμανσης των αποθεμάτων. Το πεδίο TYPE προσδιορίζει το είδος της συναλλαγής, όπως εισαγωγή ή αφαίρεση, ενώ το STATUS χαρακτηρίζει την κατάσταση της, για παράδειγμα αν είναι ενεργή ή ακυρωμένη. Το TRANSACTION_DATE τύπου datetime αποθηκεύει την ημερομηνία και ώρα κατά την οποία πραγματοποιήθηκε η κίνηση και αρχικοποιείται αυτόματα. Μέσω αυτής της δομής, ο πίνακας TRANSACTIONS προσφέρει μια πλήρη εικόνα των αλλαγών στο απόθεμα, ενισχύοντας τον έλεγχο και την αξιοπιστία της διαχείρισης αποθήκης.

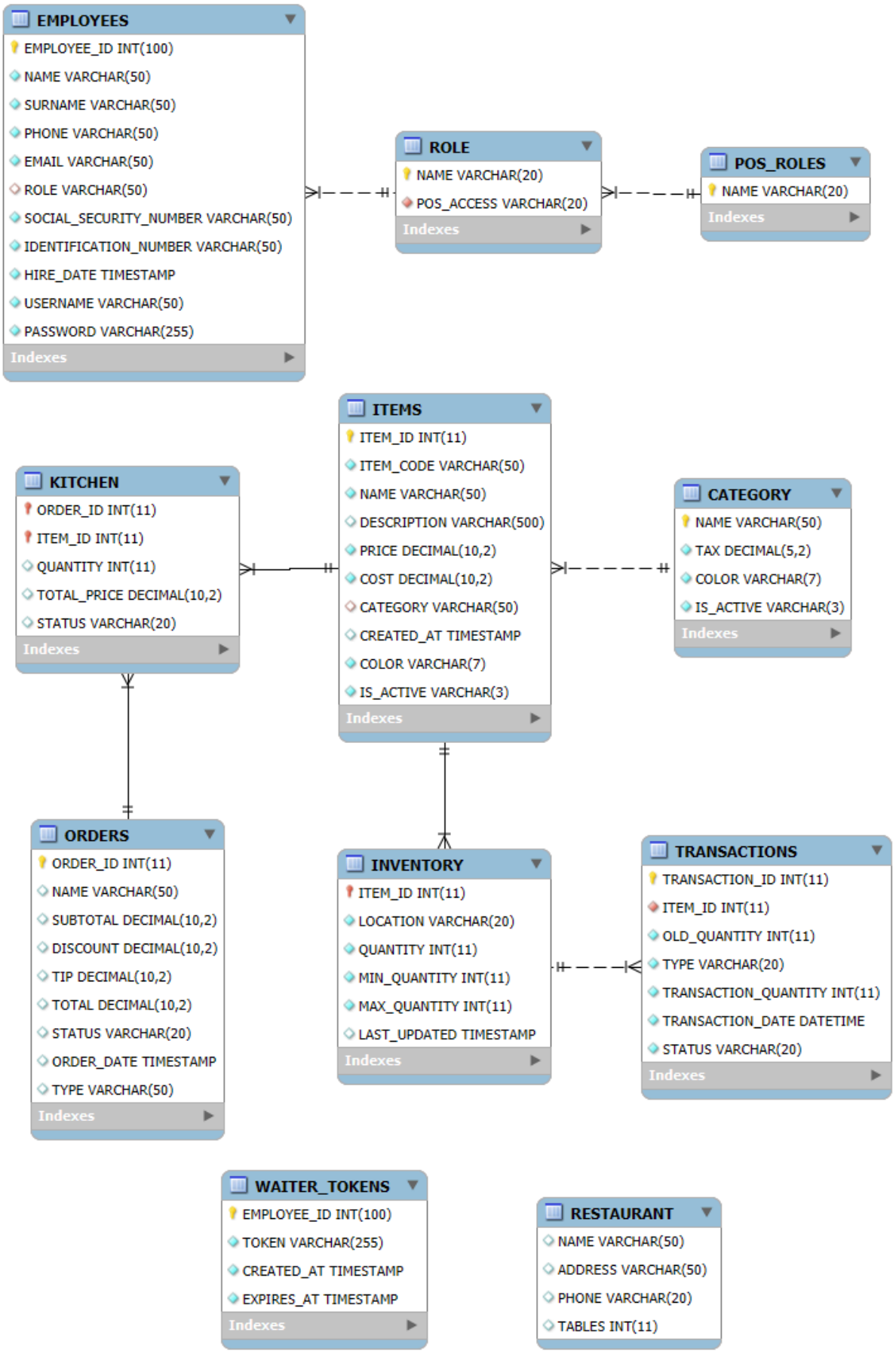
Ο πίνακας KITCHEN λειτουργεί ως ενδιάμεσος πίνακας μεταξύ παραγγελιών και ειδών, αποθηκεύοντας τα προϊόντα που πρέπει να προετοιμαστούν από την κουζίνα στο πλαίσιο κάθε παραγγελίας. Το πρωτεύον κλειδί αποτελείται από το ζεύγος ORDER_ID και ITEM_ID, διασφαλίζοντας ότι κάθε είδος εμφανίζεται μία φορά σε κάθε παραγγελία. Το ORDER_ID αποτελεί ξένο κλειδί προς τον πίνακα ORDERS και προσδιορίζει σε ποια παραγγελία ανήκει το καταχωρημένο είδος, ενώ το ITEM_ID συνδέεται με τον πίνακα ITEMS και καθορίζει το συγκεκριμένο προϊόν που πρέπει να ετοιμαστεί. Το πεδίο QUANTITY αποθηκεύει την ποσότητα του είδους που ζητήθηκε, ενώ το TOTAL_PRICE καταγράφει το συνολικό κόστος του συγκεκριμένου προϊόντος μέσα στην παραγγελία, όπως αυτό υπολογίστηκε τη στιγμή της καταχώρησης. Επιπλέον, το πεδίο STATUS παρακολουθεί την τρέχουσα κατάσταση εκτέλεσης του είδους στην κουζίνα, όπως «υπό προετοιμασία», «ακυρώθηκε» ή «ολοκληρώθηκε». Μέσω αυτής της δομής, ο πίνακας KITCHEN επιτρέπει την ομαλή ροή εργασίας στην κουζίνα, προσφέροντας συνεχή παρακολούθηση των ειδών που πρέπει να εκτελεστούν για κάθε παραγγελία.

Ο πίνακας ORDERS αποτελεί τον κεντρικό πίνακα διαχείρισης των παραγγελιών του συστήματος και αποθηκεύει όλες τις βασικές πληροφορίες που σχετίζονται με κάθε συναλλαγή. Το πεδίο ORDER_ID είναι το πρωτεύον κλειδί και δημιουργείται αυτόματα, εξασφαλίζοντας τον μοναδικό προσδιορισμό κάθε παραγγελίας. Το πεδίο NAME μπορεί να χρησιμοποιηθεί για την καταγραφή μιας ονομασίας που συνδέεται με την παραγγελία, όπως το όνομα πελάτη ή η περιγραφή του τραπεζιού. Το πεδίο SUBTOTAL αποθηκεύει το αρχικό ποσό της παραγγελίας πριν από τυχόν εκπτώσεις, ενώ το DISCOUNT καταγράφει την αξία της εφαρμοζόμενης έκπτωσης, εφόσον υπάρχει. Το TIP αναφέρεται στο φιλοδώρημα που δόθηκε, ενώ στο πεδίο TOTAL αποθηκεύεται το τελικό πληρωτέο ποσό, όπως προκύπτει μετά τον υπολογισμό όλων των χρηματικών παραμέτρων. Το STATUS υποδεικνύει την τρέχουσα κατάσταση της παραγγελίας, όπως «ολοκληρωμένη» ή «ακυρωμένη», επιτρέποντας στο σύστημα να παρακολουθεί τον κύκλο ζωής κάθε εγγραφής. Το πεδίο ORDER_DATE καταγράφει την ημερομηνία και ώρα δημιουργίας της παραγγελίας, με αυτόματη συμπλήρωση κατά την καταχώρησή της. Τέλος, το πεδίο TYPE ορίζει τον τύπο της παραγγελίας, όπως dine-in ή take away, υποστηρίζοντας διαφορετικές ροές λειτουργίας ανάλογα με τη φύση της εξυπηρέτησης.

Το Διάγραμμα Οντοτήτων–Συσχετίσεων (E.R. Diagram) που ακολουθεί απεικονίζει τη λογική δομή της βάσης δεδομένων και παρουσιάζει τον τρόπο με τον οποίο οι πίνακες συνδέονται μεταξύ τους μέσω πρωτευόντων και ξένων κλειδιών. Οι σημαντικότερες συσχετίσεις που προκύπτουν από το μοντέλο είναι οι εξής:

- Η σχέση EMPLOYEES – ROLE είναι πολλά-προς-ένα (N:1), καθώς κάθε εργαζόμενος αντιστοιχίζεται σε έναν συγκεκριμένο ρόλο, ενώ ένας ρόλος μπορεί να σχετίζεται με πολλούς υπαλλήλους. Η συσχέτιση υλοποιείται μέσω του πεδίου ROLE που λειτουργεί ως ξένο κλειδί.
- Η οντότητες ROLE και POS_ROLES συνδέονται μέσω μιας σχέσης πολλά-προς-ένα (N:1), καθώς κάθε ρόλος που εμφανίζονται στους χρήστες αντιστοιχεί σε έναν προκαθορισμένο "εσωτερικό" ρόλο του συστήματος.
- Μεταξύ των ITEMS – CATEGORY υπάρχει συσχέτιση πολλά-προς-ένα (N:1), επειδή κάθε προϊόν ανήκει σε μία κατηγορία, ενώ μια κατηγορία μπορεί να περιλαμβάνει πολλά προϊόντα.
- Η σχέση ITEMS – INVENTORY είναι ένα-προς-ένα (1:1), αφού για κάθε προϊόν υπάρχει μία και μόνο μία εγγραφή αποθέματος. Το πεδίο ITEM_ID λειτουργεί ταυτόχρονα ως πρωτεύον και ξένο κλειδί που συνδέεται με τον πίνακα ITEMS.
- Μεταξύ INVENTORY – TRANSACTIONS υπαρχη η σχέση πολλά-προς-ένα (N:1), καθώς για κάθε προϊόν αποθέματος μπορεί να υπάρχουν πολλές κινήσεις (transactions), ενώ κάθε κίνηση αφορά αποκλειστικά ένα προϊόν.
- Η σχέση ORDERS – KITCHEN είναι ένα-προς-πολλά (1:N), αφού μία παραγγελία μπορεί να περιλαμβάνει πολλά προϊόντα, και κάθε προϊόν μπορει να υπαρχει μια φορα σε καθε παραγγελια.
- Αντίστοιχα, η σχέση ITEMS – KITCHEN είναι ένα-προς-πολλά (1:N), καθώς ένα προϊόν μπορεί να εμφανιστεί σε πολλές παραγγελίες, ενω το ιδιο προιον απο διαφορες παραγγελιες αντηστηχει σε ενα μονο προιον.
- Τέλος, η σχέση WAITER_TOKENS – EMPLOYEES είναι ένα-προς-ένα (1:1), γιατί σε κάθε υπάλληλο αντιστοιχεί ένα ενεργό token πρόσβασης, το οποίο χρησιμοποιείται για την αυθεντικοποίησή του στην Android εφαρμογή.

Το παρακάτω Διάγραμμα Οντοτήτων–Συσχετίσεων (ERD) παρουσιάζει τη δομή της βάσης δεδομένων δηλαδη τους πίνακες με τα βασικά τους πεδία και τα πρωτεύοντα/ξένα κλειδιά, καθώς και τις κύριες σχέσεις μεταξύ τους. Το διάγραμμα αυτο διευκολύνει την κατανόηση του τρόπου διασύνδεσης των δεδομένων και της ροής πληροφοριών στο σύστημα.



Σχήμα 3.4: Διάγραμμα Οντοτήτων–Συσχετίσεων

3.5. Σύνοψη κεφαλαίου

Στο κεφάλαιο αυτό παρουσιάστηκε η σχεδίαση του συστήματος, ξεκινώντας από την αρχιτεκτονική του και τα τρία επίπεδα λειτουργίας (presentation, application, data), τα οποία καθορίζουν τον τρόπο αλληλεπίδρασης των χρηστών με την εφαρμογή και τη διαχείριση των δεδομένων. Στη συνέχεια αναλύθηκαν τα διαγράμματα κλάσεων, τα οποία αποτυπώνουν τη στατική δομή του λογισμικού και τις σχέσεις μεταξύ των κλάσεων που υλοποιούν τη λειτουργικότητα της εφαρμογής. Τέλος, παρουσιάστηκε η βάση δεδομένων με τους πίνακές της και το Διάγραμμα Οντοτήτων–Συσχετίσεων (E.R. Diagram), επισημαίνοντας τις βασικές σχέσεις και περιορισμούς που διασφαλίζουν την ακεραιότητα, τη συνοχή και την αποτελεσματική διαχείριση των πληροφοριών του συστήματος.

Κεφάλαιο 4ο: Υλοποίηση Συστήματος

4.1. Εισαγωγή

Το παρόν κεφάλαιο παρουσιάζει αναλυτικά τη διαδικασία υλοποίησης του Συστήματος Διαχείρισης Χώρου Εστίασης, όπως αυτό σχεδιάστηκε στα προηγούμενα κεφάλαια. Η υλοποίηση αποτελεί το πρακτικό στάδιο της εργασίας, στο οποίο οι απαιτήσεις, τα διαγράμματα και η αρχιτεκτονική του συστήματος μετατρέπονται σε λειτουργικές εφαρμογές και υπηρεσίες. Στη συνέχεια παρουσιάζονται οι τεχνολογίες που χρησιμοποιήθηκαν, καθώς και η περιγραφή και η λειτουργικότητα των εφαρμογών, ενώ αναλύεται ο τρόπος λειτουργίας των επιμέρους υποσυστημάτων και της μεταξύ τους επικοινωνίας.

4.2. Χρησιμοποιούμενες τεχνολογίες και εργαλεία

Η υλοποίηση του Συστήματος Διαχείρισης Χώρου Εστίασης απαιτούσε τη χρήση διαφορετικών τεχνολογιών, ώστε να καλυφθούν οι ανάγκες μιας desktop εφαρμογής, μιας Android εφαρμογής, του backend συστήματος και της δημιουργίας στατιστικών διαγραμμάτων. Η επιλογή των τεχνολογιών πραγματοποιήθηκε με γνώμονα την αξιοπιστία, την απόδοση, την ευκολία στην ανάπτυξη και τη συμβατότητα μεταξύ των επιμέρους υποσυστημάτων. Ιδιαίτερη έμφαση δόθηκε επίσης στην αποτελεσματική επικοινωνία μεταξύ των εφαρμογών μέσω διαδικτυακών υπηρεσιών, καθώς και στην υποστήριξη γραφικών διεπαφών φιλικών προς τον χρήστη.

4.2.1. Τεχνολογίες desktop εφαρμογής

Η desktop εφαρμογή του συστήματος υλοποιήθηκε με τη γλώσσα προγραμματισμού Java, μια πολυπλατφορμική και αντικειμενοστρεφή γλώσσα. Η Java επιτρέπει τη δημιουργία εφαρμογών που εκτελούνται σε διαφορετικά λειτουργικά συστήματα χωρίς την ανάγκη τροποποίησης του κώδικα, αξιοποιώντας το μοντέλο “write once, run anywhere”. Αυτό επιτυγχάνεται μέσω της μεταγλώττισης του πηγαίου κώδικα σε αρχεία bytecode (.class), τα οποία εκτελούνται σε οποιοδήποτε σύστημα διαθέτει εγκατεστημένη την Java Virtual Machine [17].

Η αντικειμενοστρεφής φύση της Java διευκολύνει την οργάνωση του κώδικα σε αντικείμενα και κλάσεις, ενισχύοντας την αναγνωσιμότητα, τη συντηρησιμότητα και την επεκτασιμότητα της εφαρμογής. Η δομή αυτή είναι ιδιαίτερα χρήσιμη σε έργα όπως το παρόν, όπου η εφαρμογή περιλαμβάνει πολλαπλά υποσυστήματα (διαχείριση παραγγελιών, προσωπικού, αποθέματος και ταμείου) που πρέπει να λειτουργούν συνεργατικά και απομονωμένα όταν απαιτείται.

Η επιλογή της Java στη συγκεκριμένη υλοποίηση έγινε λόγω της σταθερότητας της γλώσσας, της ευκολίας διαχείρισης αντικειμενοστρεφών σχεδίων και της ευρείας υποστήριξης εργαλείων ανάπτυξης. Επιπλέον, η συμβατότητά της με βιβλιοθήκες τρίτων, όπως Swing, FlatLaf, JS.O.N. και ZXing διευκολύνει την ανάπτυξη εφαρμογών.

Για την ανάπτυξη του γραφικού περιβάλλοντος χρήστη (Graphical User Interface) χρησιμοποιήθηκε η βιβλιοθήκη Java Swing, η οποία αποτελεί μέρος των Java Foundation Classes. Η Swing παρέχει ένα πλήρες σύνολο παραμετροποιήσιμων γραφικών στοιχείων, όπως παράθυρα, πίνακες, κουμπιά και μενού, τα οποία μπορούν να προσαρμοστούν κατά τον χρόνο εκτέλεσης. Το γεγονός ότι τα components της Swing είναι επεκτάσιμα επιτρέπει την προσαρμογή τους στις ιδιαίτερες απαιτήσεις της εφαρμογής, καθιστώντας τη βιβλιοθήκη ιδανική για σύνθετες desktop εφαρμογές [18].

Για τη βελτίωση της εμφάνισης και της εμπειρίας χρήστη, στην εφαρμογή χρησιμοποιήθηκε η βιβλιοθήκη FlatLaf, μια μοντέρνα open-source υλοποίηση Look and Feel για Swing. Το FlatLaf

εφαρμόζει ένα σύγχρονο, «flat» ύφος σχεδιασμού, χωρίς σκιές ή περίπλοκες διαβαθμίσεις, προσφέροντας καθαρά και απλά στοιχεία διεπαφής που βελτιώνουν την οπτική συνοχή και την ευχρηστία της εφαρμογής [19].

Η επιλογή της Swing και του FlatLaf επιτρέπει την ανάπτυξη μιας φιλικής, λειτουργικής και επεκτάσιμης διεπαφής που συνδέεται άμεσα με το αρχιτεκτονικό μοντέλο της εφαρμογής. Τα γραφικά στοιχεία συνεργάζονται με τα modules της Java για τη διαχείριση παραγγελιών, του προσωπικού, του αποθέματος και του ταμείου, διασφαλίζοντας ταυτόχρονα εύκολη συντήρηση και δυνατότητα μελλοντικών επεκτάσεων.

Η αρχιτεκτονική που χρησιμοποιήθηκε για την ανάπτυξη της desktop εφαρμογής βασίζεται στο σχεδιαστικό πρότυπο Model-View-Controller (M.V.C.), το οποίο στοχεύει στον διαχωρισμό των δεδομένων και της επιχειρησιακής λογικής από την παρουσίασή τους στον χρήστη. Η αρχιτεκτονική M.V.C. διαχωρίζει την εφαρμογή σε τρία βασικά συνεργαζόμενα στοιχεία: Model, View και Controller.

- Το Model αναπαριστά τα δεδομένα της εφαρμογής και την επιχειρησιακή λογική τους, χωρίς να γνωρίζει την ύπαρξη του View ή του Controller. Ενημερώνει τα ενδιαφερόμενα μέρη για αλλαγές μέσω μηχανισμών ειδοποίησης γεγονότων, διασφαλίζοντας ότι οι αλλαγές στα δεδομένα αντικατοπτρίζονται σωστά σε όλα τα τμήματα της εφαρμογής.
- Το View είναι υπεύθυνο για την παρουσίαση των δεδομένων στον χρήστη και ενημερώνεται αυτόματα από το Model για τυχόν αλλαγές.
- Το Controller λειτουργεί ως ενδιάμεσος μεταξύ του χρήστη και της εφαρμογής, μεταφράζοντας τις ενέργειες του χρήστη σε κατάλληλες τροποποιήσεις στο Model. Μέσω της συνεργασίας αυτών των τριών στοιχείων, η εφαρμογή επιτυγχάνει καθαρή διαχωρισμένη λογική, ευκολία συντήρησης και δυνατότητα επέκτασης.

Ένα σημαντικό πλεονέκτημα του M.V.C. είναι η δυνατότητα ύπαρξης πολλαπλών Views που χρησιμοποιούν το ίδιο Model, επιτρέποντας διαφορετικές παρουσιάσεις των ίδιων δεδομένων χωρίς αλληλεξαρτήσεις [20]. Στην υλοποίηση της συγκεκριμένης εφαρμογής, η αρχιτεκτονική M.V.C. διευκολύνει τη διαχείριση των διαφορετικών λειτουργιών, όπως παραγγελίες, προσωπικό, απόθεμα και ταμείο, παρέχοντας παράλληλα ένα σταθερό και επεκτάσιμο πλαίσιο για το γραφικό περιβάλλον της εφαρμογής.

Για την ανταλλαγή δεδομένων μεταξύ της desktop εφαρμογής και του backend χρησιμοποιήθηκε η βιβλιοθήκη JS.O.N.-Java (org.json), η οποία υποστηρίζει τη διαχείριση δεδομένων σε μορφή JS.O.N. (JavaScript Object Notation). Η βιβλιοθήκη παρέχει μηχανισμούς για ανάλυση (parsing) JS.O.N. κειμένου σε αντικείμενα Java, καθώς και για τη δημιουργία JS.O.N. δομών από δεδομένα της εφαρμογής. Η JS.O.N.-Java περιλαμβάνει βασικές κλάσεις όπως τα JS.O.N.Object και JS.O.N.Array, οι οποίες διευκολύνουν την αναπαράσταση σύνθετων δομών δεδομένων με απλό και κατανοητό τρόπο για τον προγραμματιστή. Επιπλέον, η βιβλιοθήκη υποστηρίζει μετατροπές μεταξύ JSON και άλλων μορφών, όπως X.M.L. ή HT.T.P. headers, επιτρέποντας την διαλειτουργικότητα μεταξύ διαφορετικών υποσυστημάτων της εφαρμογής [21].

Η επιλογή της JS.O.N.-Java έγινε λόγω της απλότητας, της αξιοπιστίας και της ευκολίας ενσωμάτωσής της σε Java έργα, χωρίς εξωτερικές εξαρτήσεις. Στο πλαίσιο της συγκεκριμένης εφαρμογής, η βιβλιοθήκη χρησιμοποιείται για την επικοινωνία με το PHP A.P.I., επιτρέποντας ασφαλή και αποδοτική ανταλλαγή δεδομένων μεταξύ των υποσυστημάτων και την αποθήκευση των πληροφοριών στη βάση δεδομένων.

Τέλος, για την υποστήριξη δημιουργίας του κωδικου ταχείας απόκρισης (Quick Response Code-QR Code) στην desktop εφαρμογή χρησιμοποιήθηκε η βιβλιοθήκη ZXing (Zebra Crossing). Η ZXing αποτελεί μια ανοιχτού κώδικα λύση για τη γλώσσα Java. Η βιβλιοθήκη παρέχει μηχανισμούς για την ανάγνωση (decoding) και δημιουργία (encoding) μονοδιάστατων και δισδιάστατων γραμμωτών κωδικών (barcodes), υποστηρίζοντας δημοφιλείς μορφές [22]. Στο πλαίσιο της εφαρμογής, η βιβλιοθήκη χρησιμοποιείται για την παραγωγή Q.R. Codes, διευκολύνοντας την διαδικασία ταυτοποίησης του σερβιτορου.

4.2.2. Τεχνολογίες Android εφαρμογής

Η Android εφαρμογή υλοποιήθηκε επίσης στη γλώσσα προγραμματισμού Java, η οποία έχει παρουσιαστεί αναλυτικά στην προηγούμενη ενότητα (4.2.1 Τεχνολογίες desktop εφαρμογής). Στο παρόν υποκεφάλαιο δίνεται έμφαση στις τεχνολογίες και τα αρχιτεκτονικά χαρακτηριστικά που αφορούν το περιβάλλον Android και τη λειτουργία της εφαρμογής σε κινητές συσκευές.

Για τον σχεδιασμό του γραφικού περιβάλλοντος χρήστη χρησιμοποιήθηκε η γλώσσα Extensible Markup Language (X.M.L.), η οποία στο Android αξιοποιείται για τον σχεδιασμό των δομημένων διατάξεων των οπτικών στοιχείων της εφαρμογής. Η X.M.L. είναι μια γλώσσα σήμανσης που χρησιμοποιείται για την περιγραφή της δομής και του περιεχομένου των δεδομένων σε ένα έγγραφο και αποτελεί απλοποιημένη μορφή της Standard Generalized Markup Language (S.G.M.L.). Έχει καθιερωθεί ως αποδεκτό πρότυπο για τη διάθεση και την ανταλλαγή δεδομένων στο διαδίκτυο, ενώ ο χαρακτηρισμός της ως extensible οφείλεται στη δυνατότητα ορισμού προσαρμοσμένων ετικετών, ανάλογα με τις ανάγκες της εκάστοτε εφαρμογής [23]. Στο πλαίσιο του Android, μέσω των αρχείων X.M.L. καθορίζεται η διάταξη των στοιχείων του γραφικού περιβάλλοντος, όπως κουμπιά, λίστες και πεδία εισαγωγής, καθώς και οι ιδιότητές τους, χωρίς να απαιτείται η άμεση υλοποίησή τους στον κώδικα Java. Με τον τρόπο αυτό επιτυγχάνεται ο σαφής διαχωρισμός της παρουσίασης από τη λογική της εφαρμογής, γεγονός που διευκολύνει τη συντήρηση και τη μελλοντική επέκτασή της.

Η Android εφαρμογή βασίζεται στο αρχιτεκτονικό πρότυπο Model–View–ViewModel (M.V.VM.), το οποίο διαχωρίζει την επιχειρησιακή λογική και τα δεδομένα της εφαρμογής από το γραφικό περιβάλλον χρήστη. Ο διαχωρισμός αυτός διευκολύνει τη συντήρηση, τον έλεγχο και την επέκταση της εφαρμογής, ενώ παράλληλα ενισχύει την επαναχρησιμοποίηση κώδικα και τη συνεργασία μεταξύ προγραμματιστών και σχεδιαστών διεπαφής [24].

Στο πρότυπο M.V.VM., το Model περιλαμβάνει τα δεδομένα και την επιχειρησιακή λογική της εφαρμογής, καθώς και μηχανισμούς πρόσβασης σε πηγές δεδομένων, όπως διαδικτυακές υπηρεσίες. Το View αναλαμβάνει αποκλειστικά την παρουσίαση των δεδομένων και τη συλλογή της αλληλεπίδρασης του χρήστη, χωρίς επιχειρησιακή λογική, ενώ ενημερώνεται αυτόματα μέσω μηχανισμών data binding προς το ViewModel. Το ViewModel λειτουργεί ως ενδιάμεσο στρώμα, εκθέτοντας ιδιότητες και εντολές προς το View και μετατρέποντας τις ενέργειες του χρήστη σε λειτουργίες προς το Model [24].

Η αλληλεπίδραση των τριών συστατικών ακολουθεί μονόδρομη ροή εξαρτήσεων: το View γνωρίζει το ViewModel και το ViewModel γνωρίζει το Model, χωρίς άμεση σύνδεση μεταξύ View και Model. Με αυτόν τον τρόπο, το ViewModel απομονώνει το γραφικό περιβάλλον από την επιχειρησιακή λογική, επιτρέποντας την αλλαγή ή τον ανασχεδιασμό της διεπαφής χωρίς τροποποίηση του λογικού πυρήνα [24]. Το M.V.VM. προάγει την καθαρή αρχιτεκτονική και αποτελεί κατάλληλη επιλογή για εφαρμογές με σύνθετο γραφικό περιβάλλον, όπως η παρούσα Android εφαρμογή.

Για την υλοποίηση επικοινωνίας με το backend μέσω του πρωτοκόλλου HT.T.P. χρησιμοποιήθηκε η βιβλιοθήκη Retrofit. Η βιβλιοθήκη αυτή παρέχει έναν δηλωτικό τρόπο κατανάλωσης RESTful υπηρεσιών, επιτρέποντας στον προγραμματιστή να ορίζει αιτήματα προς τον διακομιστή μέσω διεπαφών (interfaces) και annotations, όπως GET και POST. Με τον τρόπο αυτό επιτυγχάνεται η type-safe διαχείριση των δεδομένων, καθώς τα αποτελέσματα των αιτημάτων αντιστοιχίζονται σε συγκεκριμένους τύπους αντικειμένων. Επιπλέον, το Retrofit βασίζεται εσωτερικά στη βιβλιοθήκη OkHttp και υποστηρίζει τη μετατροπή δεδομένων JSON μέσω converters, όπως το Gson [25].

Η βιβλιοθήκη OkHttp αποτελεί ένα χαμηλού επιπέδου εργαλείο για την υλοποίηση HT.T.P. επικοινωνίας σε εφαρμογές Java και Android. Παρέχει μηχανισμούς για την αποστολή και λήψη δικτυακών αιτημάτων, υποστηρίζοντας λειτουργίες όπως η διαχείριση συνδέσεων, η επαναχρησιμοποίηση συνδέσεων (connection pooling) και η ασύγχρονη εκτέλεση αιτημάτων, συμβάλλοντας στη βελτίωση της απόδοσης της εφαρμογής. Επιπλέον, μέσω της υποστήριξης interceptors, το OkHttp επιτρέπει την τροποποίηση και παρακολούθηση αιτημάτων και αποκρίσεων, καθιστώντας το ιδιαίτερα ευέλικτο για λειτουργίες όπως η αυθεντικοποίηση χρηστών και η καταγραφή δικτυακής δραστηριότητας [26].

Για τη μετατροπή δεδομένων μεταξύ JS.O.N. και αντικειμενοστραφών δομών της Java χρησιμοποιήθηκε η βιβλιοθήκη Gson. Η Gson παρέχει δυνατότητες σειριοποίησης (serialization) και αποσειριοποίησης (deserialization), επιτρέποντας την εύκολη χαρτογράφηση JSON δεδομένων σε αντικείμενα Java και αντίστροφα. Η βιβλιοθήκη Gson χρησιμοποιείται ευρέως σε εφαρμογές Android για την επεξεργασία δεδομένων που προέρχονται από διαδικτυακές υπηρεσίες, προσφέροντας απλό και ευέλικτο A.P.I. και υποστήριξη σύνθετων δομών δεδομένων. Συχνά ενσωματώνεται σε συνδυασμό με βιβλιοθήκες δικτύωσης, όπως το Retrofit, λειτουργώντας ως μηχανισμός μετατροπής των δεδομένων που λαμβάνονται από HT.T.P. αποκρίσεις και διευκολύνοντας την ασφαλή και αποτελεσματική διαχείριση τους μέσα στην εφαρμογή [27].

Τέλος, η λειτουργία σάρωσης κωδικών ταχείας απόκρισης (Q.R. Code) στην Android εφαρμογή υλοποιείται μέσω του Google ML Kit, ενός συνόλου βιβλιοθηκών μηχανικής όρασης (computer vision) για κινητές συσκευές. Η λειτουργία αυτή επιτρέπει την αναγνώριση και αποκωδικοποίηση διαφόρων τύπων barcode και Q.R. codes απευθείας στη συσκευή, χωρίς να απαιτείται αποστολή δεδομένων σε εξωτερικούς διακομιστές. Το ML Kit παρέχεται και υποστηρίζεται επίσημα από την Google και ενσωματώνεται στο οικοσύστημα Android μέσω των Android Developers APIs. Η χρήση του προσφέρει αξιόπιστη και αποδοτική αναγνώριση barcodes, αξιοποιώντας μοντέλα μηχανικής μάθησης βελτιστοποιημένα για φορητές συσκευές, καθιστώντας το ιδανικό για εφαρμογές που απαιτούν άμεση και ασφαλή επεξεργασία οπτικών δεδομένων [28].

4.2.3. Τεχνολογίες Backend – PHP και API

Για την υλοποίηση του backend χρησιμοποιήθηκε η PHP (Hypertext Preprocessor), μια ευρέως διαδεδομένη γλώσσα προγραμματισμού ανοιχτού κώδικα, η οποία χρησιμοποιείται κυρίως για την ανάπτυξη διαδικτυακών εφαρμογών και backend συστημάτων. Πρόκειται για server-side γλώσσα, γεγονός που σημαίνει ότι ο κώδικας εκτελείται στον διακομιστή και το αποτέλεσμα της επεξεργασίας αποστέλλεται στον client σε μορφή δεδομένων, όπως HT.M.L. ή JS.O.N., χωρίς να γίνεται ορατός ο πηγαίος κώδικας [29].

Η PHP επιλέχθηκε για την παρούσα εργασία λόγω της απλότητας, της ευρείας υποστήριξης και της άμεσης διασύνδεσής της με βάσεις δεδομένων. Υποστηρίζει τόσο διαδικαστικό όσο και αντικειμενοστραφή προγραμματισμό, γεγονός που επιτρέπει την οργάνωση του κώδικα σε

επαναχρησιμοποιήσιμα και ευανάγνωστα τμήματα. Επιπλέον, συνεργάζεται απρόσκοπτα με σύγχρονους web servers και υποστηρίζει μορφές δεδομένων όπως JS.O.N. και X.M.L., οι οποίες είναι απαραίτητες για την επικοινωνία με εφαρμογές πελάτη [29].

Στην παρούσα υλοποίηση, η PHP χρησιμοποιήθηκε για την ανάπτυξη του backend του συστήματος διαχείρισης χώρου εστίασης και την υλοποίηση των επιχειρησιακών κανόνων. Μέσω της PHP πραγματοποιείται η διαχείριση των δεδομένων της βάσης, ο έλεγχος πρόσβασης των χρηστών και η εξυπηρέτηση αιτημάτων που προέρχονται τόσο από τη desktop όσο και από την Android εφαρμογή.

Η επικοινωνία μεταξύ του backend και των εφαρμογών βασίζεται στο Representational State Transfer Application Programming Interface (RE.S.T. A.P.I.). Το RE.S.T. A.P.I. αποτελεί μια αρχιτεκτονική προσέγγιση για τον σχεδιασμό και την υλοποίηση δικτυακών εφαρμογών και διαδικτυακών υπηρεσιών, η οποία έχει καθιερωθεί ως πρότυπο στη σύγχρονη ανάπτυξη λογισμικού. Παρέχει ένα σύνολο κανόνων και περιορισμών που, όταν ακολουθούνται, οδηγούν στη δημιουργία υπηρεσιών που είναι απλές, επεκτάσιμες και εύκολα ενοποιήσιμες με άλλα συστήματα [30].

Η λειτουργία ενός RE.S.T. A.P.I. βασίζεται στην έννοια των πόρων (resources), οι οποίοι αναπαριστούν οντότητες πληροφορίας, όπως χρήστες, παραγγελίες ή προϊόντα. Κάθε πόρος ταυτοποιείται μοναδικά μέσω ενός Uniform Resource Identifier (U.R.I.), ενώ η αλληλεπίδραση του client με τον server πραγματοποιείται μέσω της αναπαράστασης (representation) του πόρου, συνήθως σε μορφή JSON. Η επικοινωνία πραγματοποιείται μέσω του πρωτοκόλλου HTTP, αξιοποιώντας τυποποιημένες μεθόδους όπως GET, POST, PUT, PATCH και DELETE. Κάθε αίτημα είναι stateless, δηλαδή περιέχει όλες τις απαραίτητες πληροφορίες για την επεξεργασία του χωρίς να απαιτεί την διατήρηση κατάστασης από τον διακομιστή [30].

Η αρχιτεκτονική RE.S.T. στηρίζεται σε βασικές αρχές, όπως ο διαχωρισμός client–server, η ομοιόμορφη διεπαφή (uniform interface), η δυνατότητα προσωρινής αποθήκευσης (cacheability) και η πολυεπίπεδη αρχιτεκτονική. Χάρη σε αυτά τα χαρακτηριστικά, τα RE.S.T. A.P.I.s προσφέρουν απλότητα, υψηλή επεκτασιμότητα και ανεξαρτησία από τη γλώσσα ή την πλατφόρμα [30]. Στο πλαίσιο της παρούσας εργασίας, το RE.S.T. A.P.I. λειτουργεί ως ο μηχανισμός επικοινωνίας μεταξύ του backend συστήματος και των εφαρμογών desktop και Android, επιτρέποντας την ασφαλή και αποδοτική ανταλλαγή δεδομένων. Αυτή η προσέγγιση επιτρέπει την ανεξαρτησία των εφαρμογών-πελατών από την εσωτερική υλοποίηση του συστήματος, προσφέροντας ασφάλεια, ευελιξία και δυνατότητα μελλοντικής επέκτασης.

4.2.4. Τεχνολογίες βάσης δεδομένων

Για την υλοποίηση της βάσης δεδομένων χρησιμοποιήθηκε η MySQL, ένα ανοικτού κώδικα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (Relational Database Management System), το οποίο χρησιμοποιείται για την αποθήκευση, οργάνωση και διαχείριση δεδομένων. Τα δεδομένα αποθηκεύονται σε πίνακες, οι οποίοι αποτελούνται από γραμμές και στήλες και οργανώνονται σε σχήματα (schemas), τα οποία καθορίζουν τη δομή των δεδομένων καθώς και τις σχέσεις μεταξύ των πινάκων. Η διαχείριση και η ανάκτηση των δεδομένων πραγματοποιούνται μέσω της γλώσσας Structured Query Language (S.Q.L.), ενώ υποστηρίζονται επίσης σύγχρονοι τύποι δεδομένων, όπως JS.O.N., επιτρέποντας την ευέλικτη αποθήκευση σύνθετων δομών πληροφορίας [31].

Η MySQL επιλέχθηκε για την αξιοπιστία, την ταχύτητα και την επεκτασιμότητα που διακρίνεται καθιστώντας την κατάλληλη για την παρούσα εργασία. Παρέχει υποστήριξη για συναλλαγές και μηχανισμούς ακεραιότητας δεδομένων, γεγονός που είναι απαραίτητο για εφαρμογές όπως ένα

σύστημα διαχείρισης χώρου εστίασης. Παράλληλα, παρέχει λειτουργίες ασφάλειας, υψηλής διαθεσιμότητας και ευελιξίας, καλύπτοντας ανάγκες τόσο παραδοσιακών S.Q.L. όσο και υβριδικών S.Q.L./NoS.Q.L. εφαρμογών [31].

Στο πλαίσιο της συγκεκριμένης υλοποίησης, η MySQL χρησιμοποιήθηκε για την αποθήκευση όλων των βασικών δεδομένων του συστήματος, όπως τα στοιχεία των εργαζομένων, οι ρόλοι πρόσβασης, τα προϊόντα, οι κατηγορίες, τα αποθέματα, οι παραγγελίες και οι κινήσεις αποθήκης. Η βάση δεδομένων λειτουργεί ως κεντρικό σημείο αναφοράς για το σύστημα, με το backend να αναλαμβάνει την πρόσβαση και τη διαχείριση των δεδομένων μέσω του RE.S.T. A.P.I..

Η χρήση της MySQL επιτρέπει την αξιόπιστη και αποδοτική ανάκτηση πληροφοριών από τις εφαρμογές desktop και Android, διασφαλίζοντας τη σωστή λειτουργία του συστήματος σε πραγματικό χρόνο. Παράλληλα, η δομημένη σχεδίαση της βάσης διευκολύνει τη συντήρηση και τη μελλοντική επέκταση του συστήματος, όπως την προσθήκη νέων λειτουργιών ή επιπλέον υποσυστημάτων.

4.2.5. Τεχνολογίες δημιουργίας διαγραμμάτων (Python)

Η Python είναι μια γενικού σκοπού, υψηλού επιπέδου γλώσσα προγραμματισμού, η οποία διακρίνεται για την απλή και κατανοητή σύνταξή της, καθώς και για τον μεγάλο αριθμό ενσωματωμένων βιβλιοθηκών και εργαλείων που διαθέτει. Η φιλοσοφία της βασίζεται στην ευκολία κατανόησης του κώδικα, γεγονός που την καθιστά κατάλληλη για την ανάπτυξη εφαρμογών ανάλυσης δεδομένων και παραγωγής στατιστικών αποτελεσμάτων. Πρόκειται για μια ερμηνευόμενη (interpreted) και αντικειμενοστραφή (object-oriented) γλώσσα με δυναμική σημασιολογία (dynamic semantics), η οποία επιτρέπει την εκτέλεση του κώδικα χωρίς μεταγλώττιση [32].

Η Python υποστηρίζει πολλαπλες αρχές του προγραμματισμου, όπως η αντικειμενοστρεφια, ο διαδικασιακος και λειτουργικος προγραμματισμος, και μπορεί να εκτελεστεί σε διαφορετικά λειτουργικά συστήματα (Windows, Linux και macOS) χωρίς τροποποίηση του κώδικα. Διαθέτει εκτεταμένη τυποποιημένη βιβλιοθήκη για πλήθος λειτουργιών, ενώ εκατοντάδες τρίτες βιβλιοθήκες επεκτείνουν τις δυνατότητές της σε κλαδους της επιστημης, οπως μηχανική μάθηση, ανάπτυξη ιστού και ανάλυση δεδομένων [32].

Στο πλαίσιο της παρούσας εργασίας, η Python χρησιμοποιήθηκε αποκλειστικά για την τη δημιουργία στατιστικών διαγραμμάτων που αφορούν τη λειτουργία του συστήματος διαχείρισης χώρου εστίασης. Τα δεδομένα προέρχονται από τη λειτουργία των παραγγελιών και του ταμείου και αξιοποιούνται για την εξαγωγή χρήσιμων συμπερασμάτων, όπως πωλήσεις ανά χρονική περίοδο, δημοφιλή προϊόντα και συνολική απόδοση του καταστήματος. Για την υλοποίηση των διαγραμμάτων χρησιμοποιήθηκαν οι ακόλουθες βιβλιοθήκες

Pandas

Η βιβλιοθήκη pandas χρησιμοποιήθηκε ως βασικό εργαλείο για τη διαχείριση και επεξεργασία των δεδομένων. Παρέχει δομές δεδομένων υψηλού επιπέδου, με κυριότερη το DataFrame, το οποίο επιτρέπει την αποθήκευση και διαχείριση δεδομένων σε μορφή πινάκων, παρόμοια με βάσεις δεδομένων ή φύλλα Excel [33]. Στην παρούσα υλοποίηση, η pandas αξιοποιήθηκε για τη συγκέντρωση και αποθήκευση δεδομένων, διευκολύνοντας έτσι την περαιτέρω επεξεργασία και απεικόνισή τους σε διαγράμματα τύπου διασποράς. Η χρήση της συνέβαλε στη σαφή δομή των δεδομένων και στη βελτίωση της αναγνωσιμότητας του κώδικα.

NumPy

Η NumPy αποτελεί τη βασική βιβλιοθήκη αριθμητικών υπολογισμών της Python η οποια προσφέρει αποδοτικές δομές για τον χειρισμό πινάκων δεδομένων, καθώς και την εκτέλεση αριθμητικών και στατιστικών πράξεων υψηλής απόδοσης [34]. Στο πλαίσιο της συγκεκριμένης υλοποίησης, η NumPy χρησιμοποιήθηκε άμεσα για τον υπολογισμό στατιστικών μεγεθών, όπως οι διάμεσοι (medians) των δεδομένων, οι οποίοι αξιοποιήθηκαν για τη δημιουργία τεταρτημορίων (quadrants) σε διαγράμματα τύπου Magic Quadrant.

Matplotlib

Η Matplotlib είναι βιβλιοθήκη οπτικοποίησης δεδομένων της Python και χρησιμοποιήθηκε για τη δημιουργία των στατιστικών διαγραμμάτων της παρούσας εργασίας. Υποστηρίζει πληθώρα τύπων γραφημάτων, όπως ραβδογράμματα, διαγράμματα γραμμής και διαγράμματα διασποράς, επιτρέποντας την αποτελεσματική και κατανοητή παρουσίαση των δεδομένων [35].

Στην παρούσα εργασία, η Matplotlib αξιοποιήθηκε για την παραγωγή ραβδογραμμάτων που απεικονίζουν την κατάσταση αποθέματος (Inventory Diagram), διαγραμμάτων γραμμής που παρουσιάζουν συνοπτικά στοιχεία πωλήσεων (Refunds, Discounts, Net Sales, Gross Profit), καθώς και για διαγράμματα διασποράς που αναδεικνύουν τη σχέση μεταξύ της συχνότητας πωλήσεων και του μικτού κέρδους των προϊόντων. Τα παραγόμενα διαγράμματα αποθηκεύονται σε μορφή εικόνας και χρησιμοποιούνται για την υποστήριξη της αξιολόγησης της λειτουργίας του συστήματος και την οπτική παρουσίαση των αποτελεσμάτων.

AdjustText

Η βιβλιοθήκη adjustText χρησιμοποιήθηκε συμπληρωματικά για τη βελτίωση της αναγνωσιμότητας των διαγραμμάτων διασποράς. Συγκεκριμένα, αξιοποιήθηκε για τη δυναμική προσαρμογή της θέσης των ετικετών δεδομένων, με στόχο την αποφυγή επικάλυψης κειμένου και σημείων στο γράφημα. Η χρήση της βιβλιοθήκης αυτής συνέβαλε στη δημιουργία πιο καθαρών και ευανάγνωστων διαγραμμάτων, ιδιαίτερα σε περιπτώσεις όπου απεικονίζονται πολλαπλά σημεία με συνοδευτικές ετικέτες, βελτιώνοντας έτσι τη συνολική ποιότητα της οπτικοποίησης [36].

4.2.6. Εργαλεία ανάπτυξης και υποστήριξης

Για την υλοποίηση του Συστήματος Διαχείρισης Χώρου Εστίασης χρησιμοποιήθηκαν διάφορα εργαλεία ανάπτυξης και υποστήριξης, τα οποία συνέβαλαν στην αποδοτική σχεδίαση, υλοποίηση, δοκιμή και τεκμηρίωση του συστήματος. Η επιλογή τους έγινε με βάση τη σταθερότητα, τη λειτουργικότητα και τη συμβατότητά τους με τις τεχνολογίες που χρησιμοποιήθηκαν σε κάθε υποσύστημα.

Για την ανάπτυξη της desktop εφαρμογής σε Java χρησιμοποιήθηκε το IntelliJ IDEA, το οποίο λειτούργησε ως το βασικό ολοκληρωμένο περιβάλλον ανάπτυξης. Το εργαλείο αυτο παρέχει δυνατότητες οργάνωσης του έργου, αυτόματης συμπλήρωσης κώδικα και εντοπισμού σφαλμάτων, διευκολύνοντας σημαντικά την ανάπτυξη και τη συντήρηση της εφαρμογής [37].

Για την ανάπτυξη της Android εφαρμογής χρησιμοποιήθηκε το Android Studio, το επίσημο περιβάλλον ανάπτυξης για εφαρμογές Android. Το εργαλείο αυτο προσέφερε ενσωματωμένη

υποστήριξη για το Android SDK, δυνατότητες δοκιμών μέσω emulator και εργαλεία debugging, εξασφαλίζοντας τη σωστή λειτουργία της εφαρμογής [38].

Για την ανάπτυξη του backend σε PHP, χρησιμοποιήθηκε το Visual Studio Code.Το συγκεκριμένο εργαλείο επιλέχθηκε λόγω της ευελιξίας του, της υποστήριξης πολλαπλών γλωσσών προγραμματισμού και της δυνατότητας επέκτασής του μέσω πρόσθετων, τα οποία διευκόλυναν την ανάπτυξη και τον έλεγχο του κώδικα [39].

Για τη διαχείριση και μεταφορά αρχείων στον απομακρυσμένο διακομιστή χρησιμοποιήθηκε το FileZilla, μέσω του οποίου πραγματοποιήθηκε η μεταφόρτωση των αρχείων του backend και η ενημέρωση του περιβάλλοντος λειτουργίας του συστήματος [40].

Για τη διαχείριση της βάσης δεδομένων MySQL χρησιμοποιήθηκε το MySQL Workbench.Το εργαλείο αυτο επέτρεψε τη δημιουργία και επεξεργασία των πινάκων, την εκτέλεση ερωτημάτων S.Q.L. και τον έλεγχο της ακεραιότητας των δεδομένων, συμβάλλοντας στη σωστή οργάνωση και λειτουργία της βάσης δεδομένων του συστήματος [41].

Τέλος, για τη μοντελοποίηση και τεκμηρίωση του συστήματος χρησιμοποιήθηκε το Visual Paradigm, μέσω του οποίου δημιουργήθηκαν τα διαγράμματα U.M.L. και Use Case που παρουσιάζονται στα αντίστοιχα κεφάλαια της εργασίας. Το εργαλείο αυτο συνέβαλε στην καλύτερη αποτύπωση της αρχιτεκτονικής και της συνολικής λειτουργικότητας του συστήματος [42].

Συνολικά, τα εργαλεία που χρησιμοποιήθηκαν υποστήριξαν αποτελεσματικά όλα τα στάδια ανάπτυξης, από τον αρχικό σχεδιασμό έως την τελική υλοποίηση και τεκμηρίωση, συμβάλλοντας καθοριστικά στην επιτυχή ολοκλήρωση της παρούσας πτυχιακής εργασίας.

4.3. Desktop εφαρμογή – Περιγραφή και λειτουργικότητα

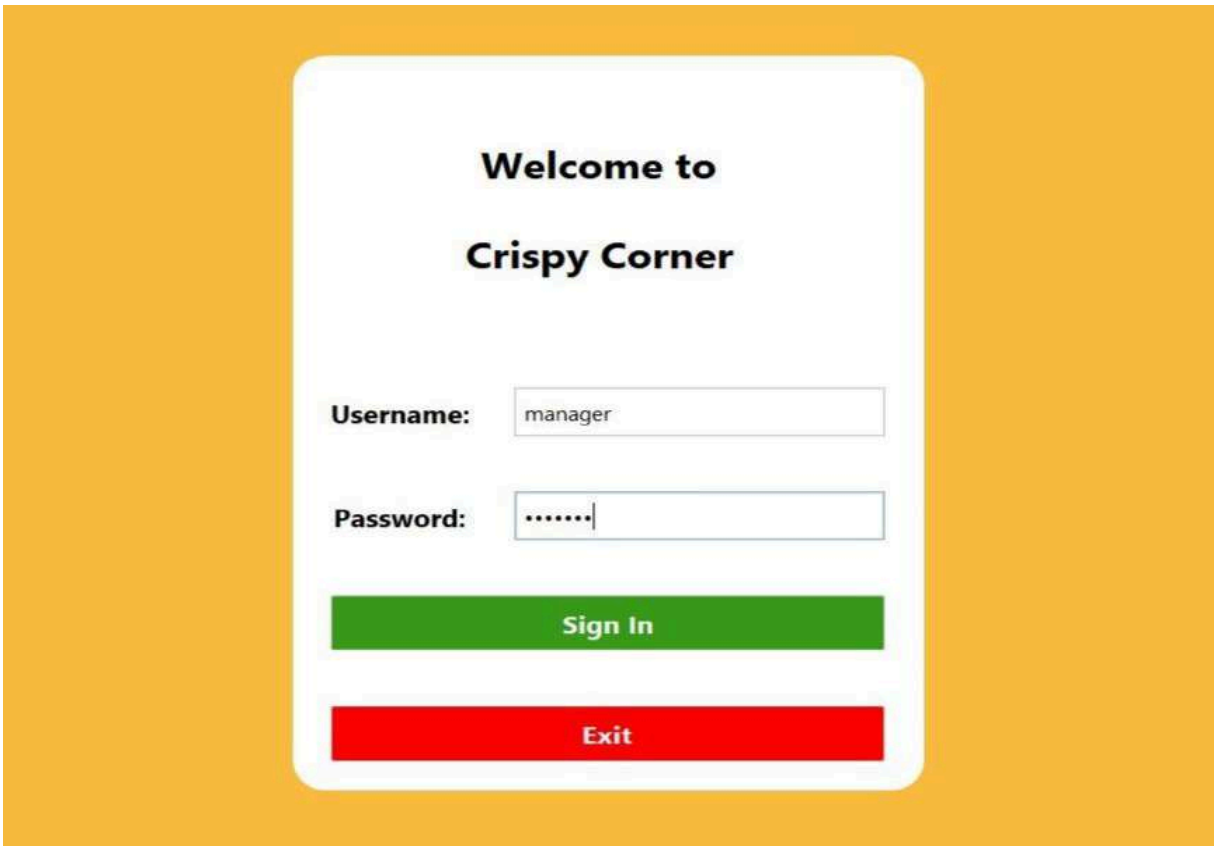
Η desktop εφαρμογή αποτελεί το βασικό εργαλείο του Συστήματος Διαχείρισης Χώρου Εστίασης και απευθύνεται σε ολο το προσωπικό της επιχείρησης. Γιατι μέσω της εφαρμογής παρέχεται ολοκληρωμένος έλεγχος των καθημερινών λειτουργιών, όπως η διαχείριση προσωπικού, παραγγελιών, αποθέματος και ταμείου, καθώς και η συλλογή δεδομένων για την εξαγωγή στατιστικών στοιχείων.

Η εφαρμογή έχει σχεδιαστεί με στόχο την ευχρηστία, την αξιοπιστία και την ταχύτητα στην εκτέλεση των λειτουργιών, ώστε να υποστηρίζει αποτελεσματικά το προσωπικό σε περιβάλλοντα αυξημένων απαιτήσεων, όπως οι χώροι εστίασης. Επιπλέον, λειτουργεί σε άμεση συνεργασία με το backend και τη βάση δεδομένων του συστήματος, εξασφαλίζοντας τη συνέπεια και την ακεραιότητα των δεδομένων που διαχειρίζεται.

Στις επόμενες υποενότητες παρουσιάζονται συνοπτικα οι βασικές λειτουργίες της desktop εφαρμογής, η δομή της διεπαφής χρήστη και ο τρόπος με τον οποίο υποστηρίζονται οι διαφορετικοί ρόλοι χρηστών.

4.3.1. Είσοδος και έξοδος στην εφαρμογή

Με την εκκίνηση της desktop εφαρμογής εμφανίζεται η φόρμα εισόδου, η οποία αποτελεί το αρχικό σημείο αλληλεπίδρασης του χρήστη με το σύστημα. Η φόρμα περιλαμβάνει μήνυμα καλωσορίσματος και το όνομα της επιχείρησης (στην παρούσα περίπτωση Crispy Corner), καθώς και δύο πεδία εισαγωγής για το όνομα χρήστη και τον κωδικό πρόσβασης. Επιπλέον, παρέχονται δύο κουμπιά, ένα για την είσοδο στο σύστημα και ένα για την έξοδο/τερματισμό της εφαρμογής.



Σχήμα 4.1: Φορμα εισοδου

Για να μπορέσει ένας χρήστης να αποκτήσει πρόσβαση στην εφαρμογή, είναι απαραίτητο να διαθέτει έγκυρα ένα συνδυασμό από όνομα χρήστη και κωδικό πρόσβασης, τα οποία έχουν προηγουμένως καταχωριστεί στο σύστημα. Κατά την επιτυχή εισαγωγή των στοιχείων, πραγματοποιείται έλεγχος αυθεντικοποίησης και ο χρήστης μεταφέρετε στην αντίστοιχη οθόνη, η οποία καθορίζεται από τον ρόλο που του έχει ανατεθεί (π.χ. ταμίας, διευθυντής). Οι επιμέρους οθόνες και λειτουργίες παρουσιάζονται αναλυτικά στις επόμενες ενότητες.

Σε περίπτωση που ο χρήστης εισάγει λανθασμένα στοιχεία σύνδεσης, η εφαρμογή εμφανίζει το παρακάτω μήνυμα σφάλματος, ενημερώνοντάς τον ότι τα δεδομένα εισόδου δεν είναι έγκυρα.

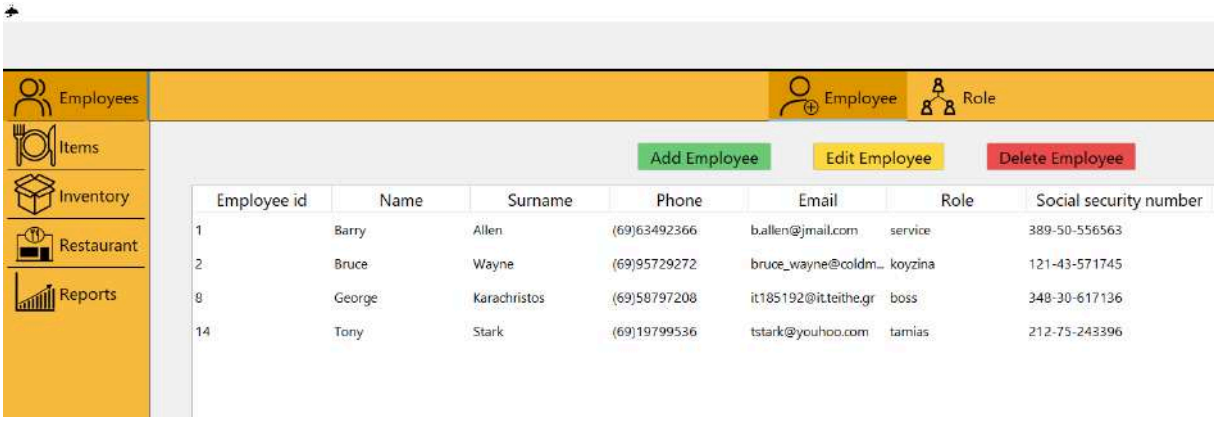
Σχήμα 4.2: Είσοδος χρήστη με λανθασμένα στοιχεία

Η αποσύνδεση από την εφαρμογή μπορεί να πραγματοποιηθεί οποιαδήποτε στιγμή. Ο χρήστης επιλέγει το εικονίδιο χρήστη (ανθρωπάκι) που βρίσκεται στο επάνω δεξί μέρος της διεπαφής και στη συνέχεια την επιλογή “Sign out”. Με την ενέργεια αυτή τερματίζεται η ενεργή συνεδρία και ο χρήστης επιστρέφει στη φόρμα εισόδου.

Σχήμα 4.3: Αποσύνδεση από την εφαρμογή

4.3.2. Διευθυντής

Η οθόνη του διευθυντή παρέχει πρόσβαση σε όλες τις βασικές λειτουργίες διαχείρισης της εφαρμογής και είναι οργανωμένη σε πέντε κύριες καρτέλες, οι οποίες εμφανίζονται στο αριστερό τμήμα της διεπαφής. Οι καρτέλες αυτές αντιστοιχούν στις βασικές ενότητες που μπορεί να διαχειριστεί ο διευθυντής και είναι οι εξής: Employees, Items, Inventory, Restaurant και Reports.



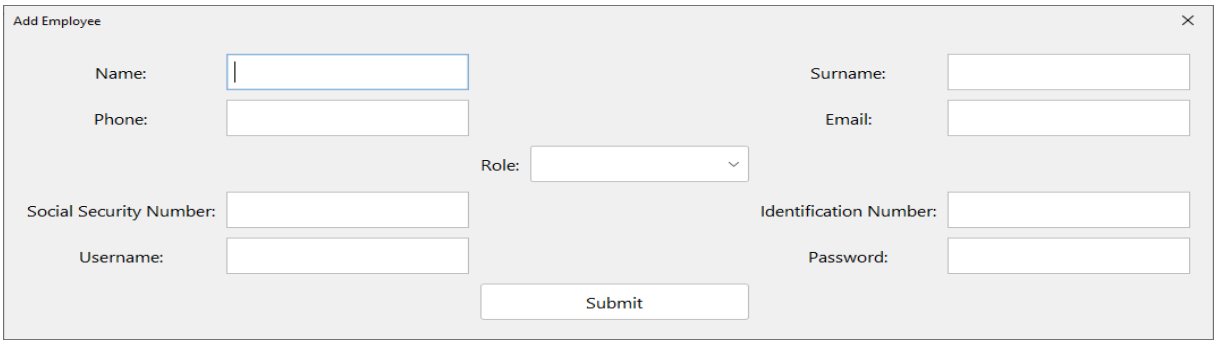
Σχήμα 4.4: Οθόνη του διευθυντή

Με εξαίρεση την καρτέλα Restaurant, όλες οι υπόλοιπες περιλαμβάνουν δύο υποκαρτέλες, οι οποίες αντιστοιχούν σε επιμέρους υποενότητες διαχείρισης. Πιο συγκεκριμένα:

- η καρτέλα Employees περιλαμβάνει τις υποκαρτέλες Employee και Role,
- η καρτέλα Items περιλαμβάνει τις υποκαρτέλες Item και Category,
- η καρτέλα Inventory περιλαμβάνει τις υποκαρτέλες Inventory και Transactions,
- η καρτέλα Reports περιλαμβάνει τις υποκαρτέλες Sales και Items.

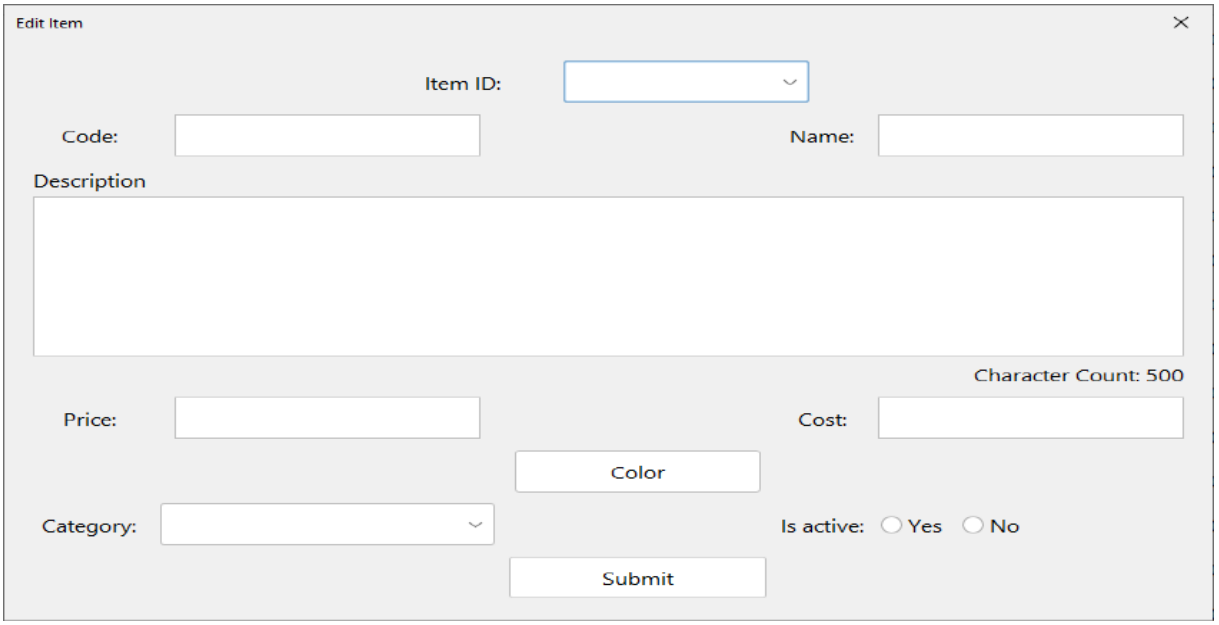
Η πλειονότητα των υποκαρτελών ακολουθεί κοινή δομή διεπαφής, η οποία αποτελείται από έναν πίνακα δεδομένων και τρία βασικά κουμπιά λειτουργιών: Εισαγωγή, Επεξεργασία και Διαγραφή. Κάθε κουμπί αντιστοιχεί σε συγκεκριμένη ενέργεια διαχείρισης των δεδομένων της αντίστοιχης υποενότητας.

Η λειτουργία Εισαγωγής ανοίγει νέο παράθυρο, το οποίο περιέχει την κατάλληλη φόρμα για την καταχώρηση νέων δεδομένων, καθώς και κουμπί επιβεβαίωσης για την αποθήκευσή τους στο σύστημα. Για παράδειγμα, στην υποενότητα Employee, το κουμπί “Add Employee” ανοίγει παράθυρο με φόρμα εισαγωγής στοιχείων υπαλλήλου.



Σχήμα 4.5: Παράθυρο Add Employee

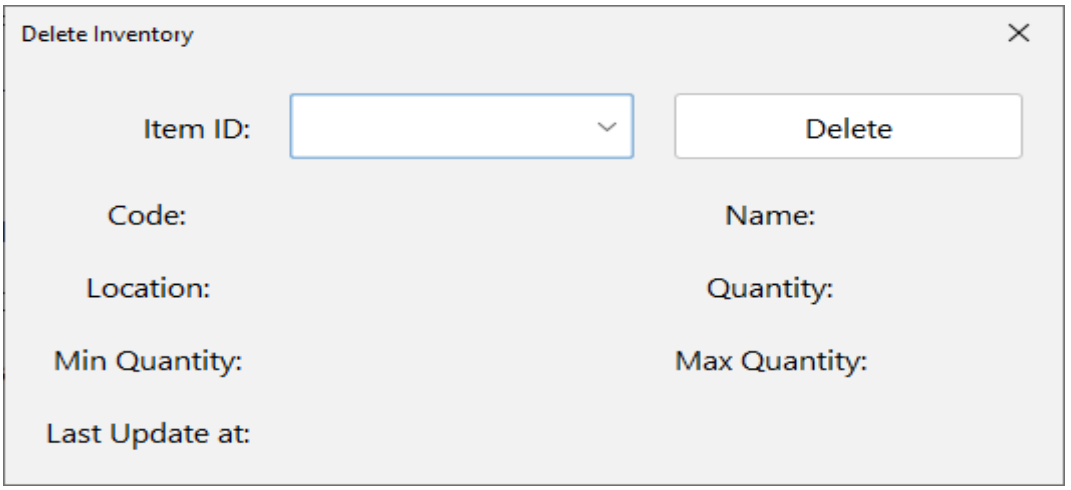
Η λειτουργία Επεξεργασίας ανοίγει παράθυρο που περιλαμβάνει τουλάχιστον μία αναδιπλούμενη λίστα (drop-down list), από την οποία ο διευθυντής επιλέγει μία από τις ήδη υπάρχουσες εγγραφές. Μετά την επιλογή, η φόρμα συμπληρώνεται αυτόματα με τα αντίστοιχα δεδομένα, τα οποία μπορούν να τροποποιηθούν και να αποθηκευτούν μέσω ειδικού κουμπιού. Για παράδειγμα, στην υποενότητα Item, το κουμπί “Edit Item” επιτρέπει την επεξεργασία των στοιχείων ενός προϊόντος.



Σχήμα 4.6: Παράθυρο Edit Item

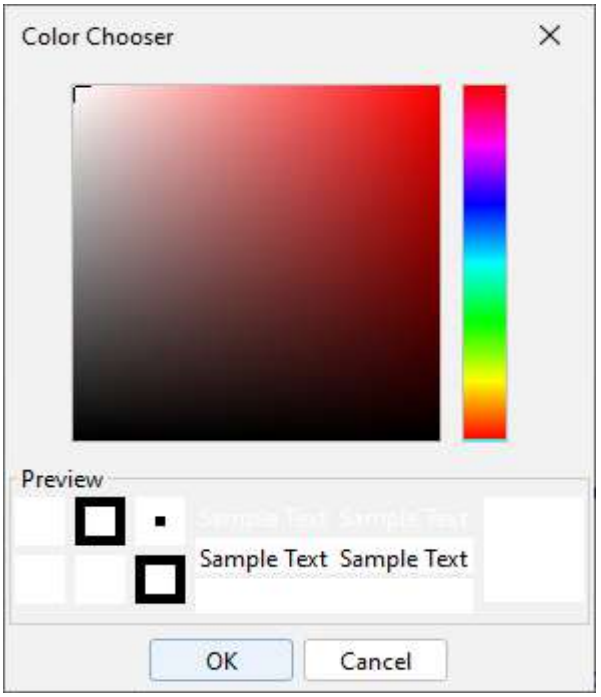
Αξίζει να σημειωθεί ότι η υποενότητα Transactions δεν περιλαμβάνει δυνατότητα επεξεργασίας, καθώς σε αυτή καταγράφονται όλες οι μεταβολές που πραγματοποιούνται στα αποθέματα, λειτουργώντας ως ιστορικό κινήσεων και όχι ως πίνακας τροποποιήσιμων δεδομένων.

Η λειτουργία Διαγραφής ανοίγει παράθυρο στο οποίο ο διευθυντής επιλέγει την εγγραφή προς διαγραφή μέσω αναδιπλούμενης λίστας (drop-down list). Μετά την επιλογή, τα στοιχεία εμφανίζονται αυτόματα και παρέχεται κουμπί για την οριστική διαγραφή της εγγραφής. Για παράδειγμα, στην υποενότητα Inventory, το κουμπί “Delete Inventory” ανοίγει φόρμα διαγραφής αποθέματος προϊόντος.



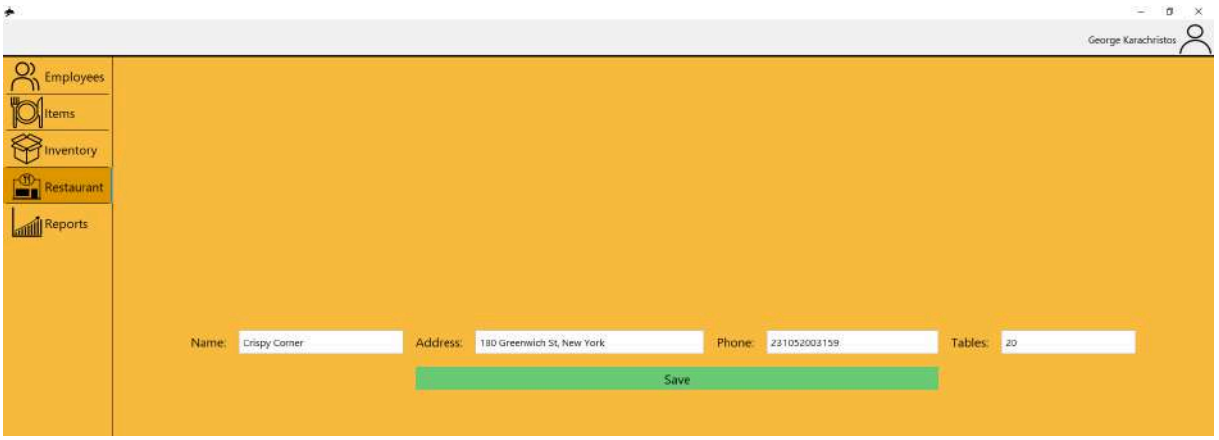
Σχήμα 4.7: Παράθυρο Delete Inventory

Ορισμένες φόρμες περιλαμβάνουν επιπλέον κουμπί “Color”, το οποίο ανοίγει ειδικό παράθυρο επιλογής χρώματος (color chooser), επιτρέποντας την αλλαγη του χρωματος του φοντου στο menu του ταμια και σερβιτορου.



Σχήμα 4.8: Παράθυρο Color Chooser

Η καρτέλα Restaurant διαφέρει από τις υπόλοιπες, καθώς περιλαμβάνει μία ενιαία φόρμα με τέσσερα πεδία εισαγωγής για το όνομα της επιχείρησης, τη διεύθυνση, το τηλέφωνο επικοινωνίας και το πλήθος των τραπεζιών, καθώς και κουμπί για την αποθήκευση των στοιχείων αυτών.



Σχήμα 4.9: Καρτέλα Restaurant

Η καρτέλα Reports παρέχει γραφική απεικόνιση των στατιστικών στοιχείων του συστήματος διαχείρισης χώρου εστίασης και περιέχει συνολικά έξι διαγράμματα: τέσσερα στην υποκαρτέλα Sales και δύο στην υποκαρτέλα Items.

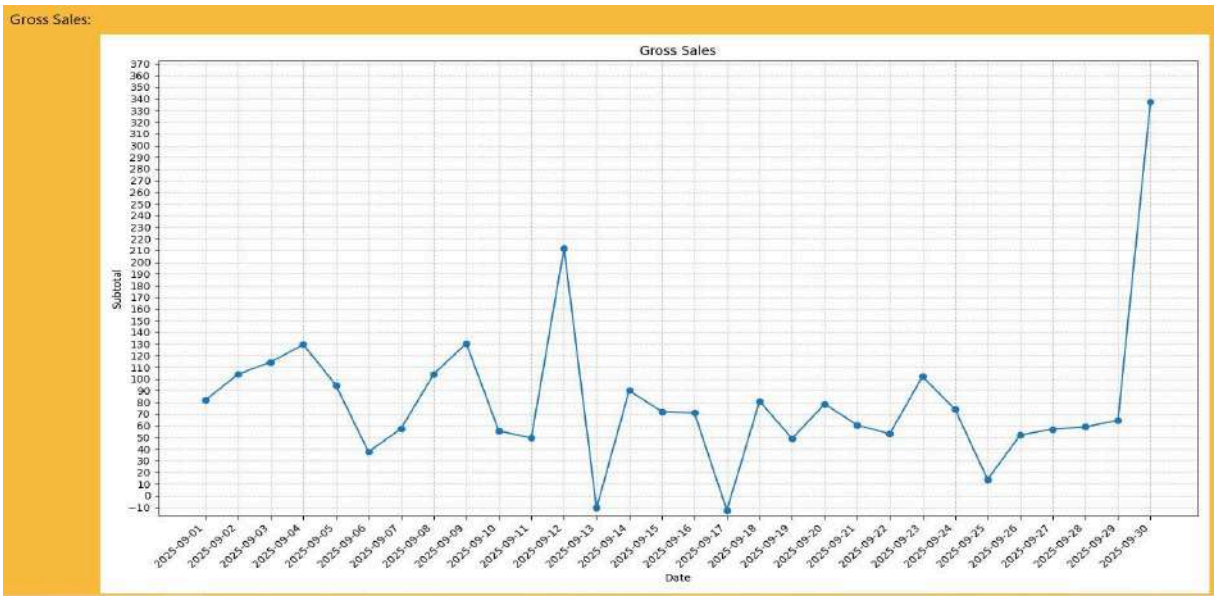
Η υποκαρτέλα Sales περιλαμβάνει πέντε ετικέτες με συνολικά ποσά για τα εξής:

- Gross Sales: Συνολικές μικτές πωλήσεις προϊόντων πριν την αφαίρεση εκπτώσεων και άμεσων εξόδων παραγωγής.
- Refunds: Συνολικό ποσό επιστροφών χρημάτων από ολοκληρωμένες πωλήσεις προϊόντων.
- Discounts: Συνολικές εκπτώσεις που έχουν δοθεί στους πελάτες.
- Net Sales: Καθαρές πωλήσεις, δηλαδή τα έσοδα μετά την αφαίρεση επιστροφών και εκπτώσεων.
- Gross Profit: Μικτό κέρδος, δηλαδή τα έσοδα μετά την αφαίρεση των άμεσων εξόδων παραγωγής προϊόντων.

Σχήμα 4.10: Ετικέτες Sales

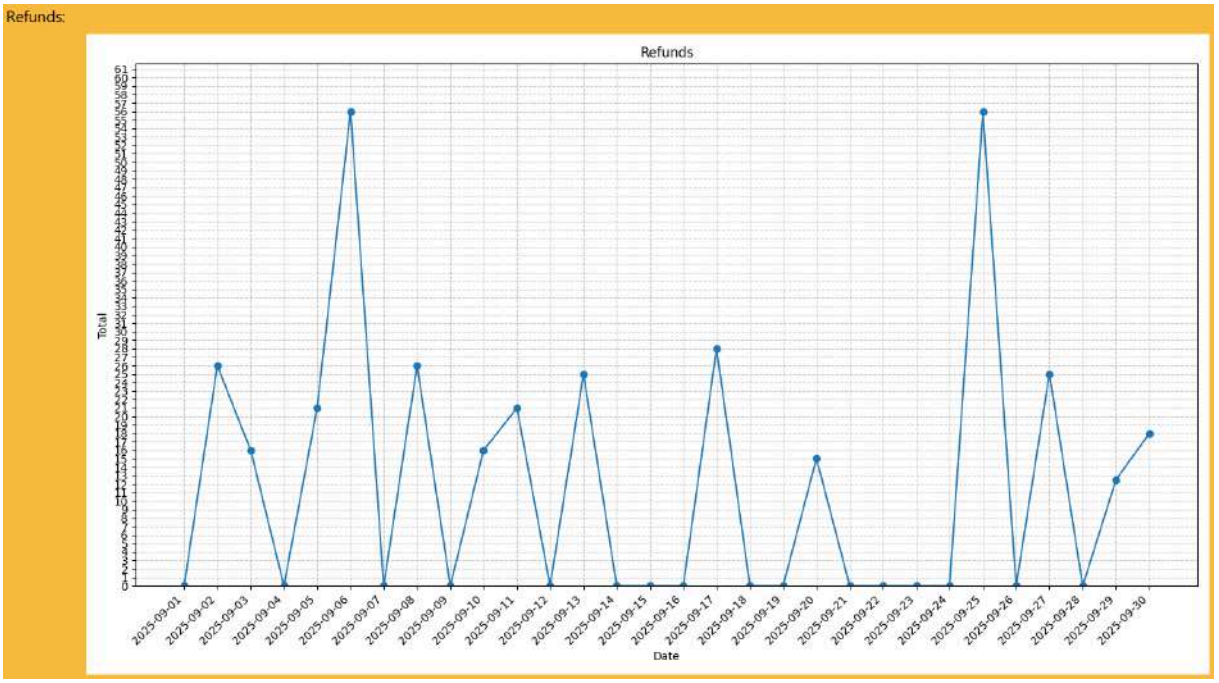
Τα διαγράμματα της υποκαρτέλας Sales περιγράφονται ως εξής:

Διάγραμμα Gross Sales: Απεικονίζει τα συνολικά έσοδα από πωλήσεις προϊόντων ανά ημέρα, πριν την αφαίρεση εκπτώσεων και άμεσων εξόδων παραγωγής.



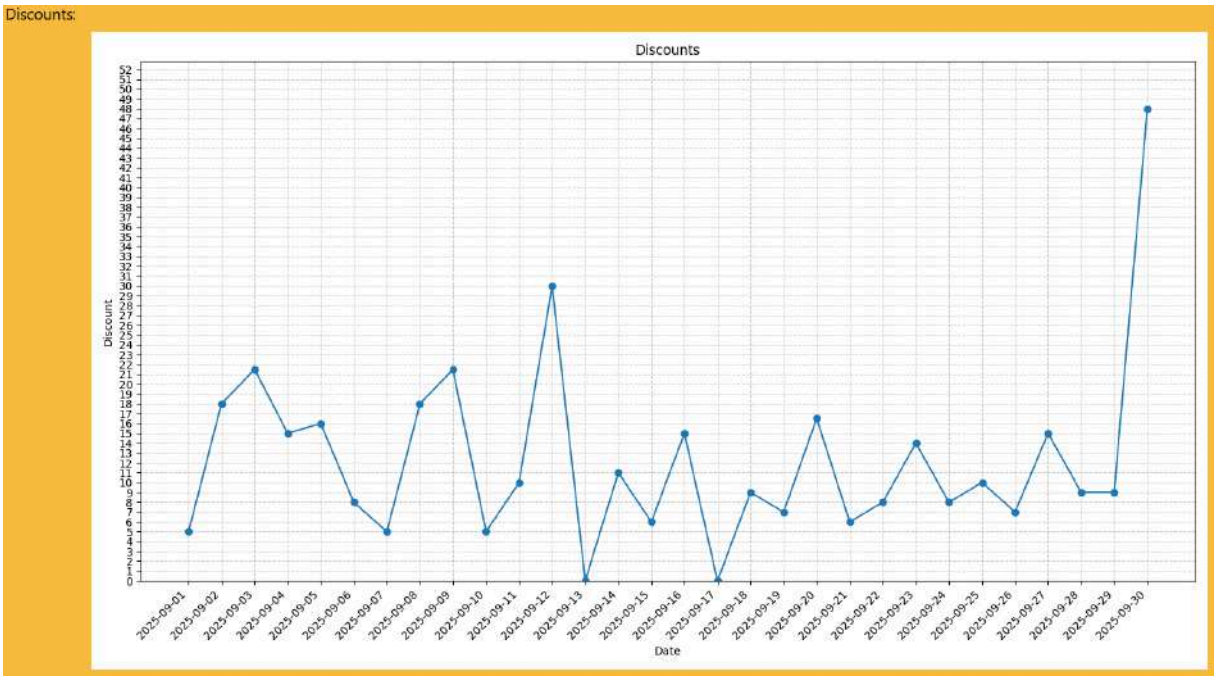
Σχήμα 4.11: Διάγραμμα Gross Sales

Διάγραμμα Refunds: Απεικονίζει τις συνολικές επιστροφές χρημάτων στους πελάτες ανά ημέρα.



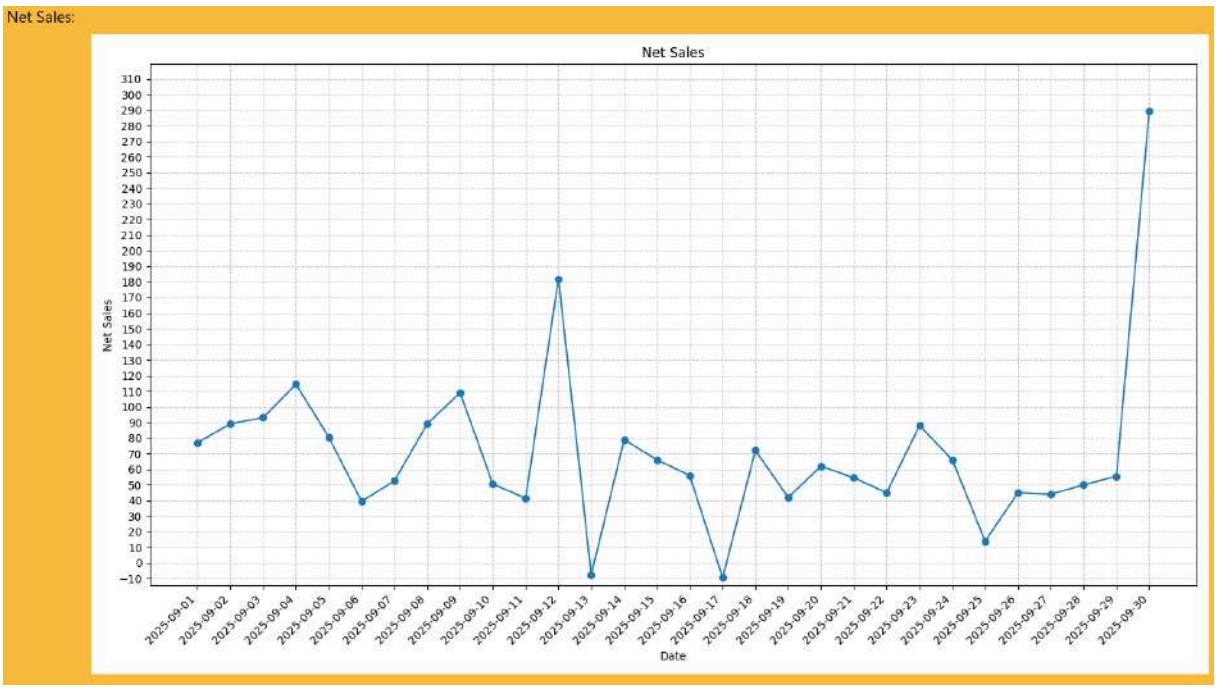
Σχήμα 4.12: Διάγραμμα Refunds

Διάγραμμα Discounts: Απεικονίζει τις συνολικές εκπτώσεις που εφαρμόστηκαν στους πελάτες ανά ημέρα.



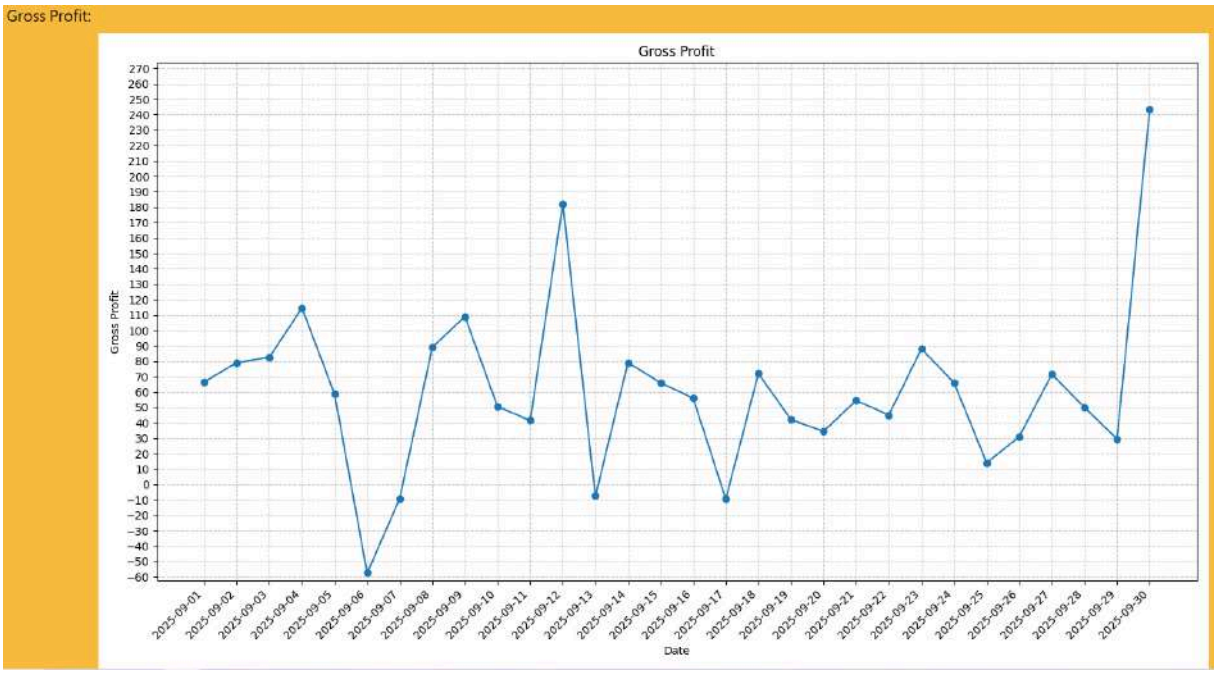
Σχήμα 4.13: Διάγραμμα Discounts

Διάγραμμα Net Sales: Παρουσιάζει τα καθαρά έσοδα της επιχείρησης μετά την αφαίρεση επιστροφών και εκπτώσεων, παρέχοντας μια πιο ρεαλιστική εικόνα του εισοδήματος.



Σχήμα 4.14: Διάγραμμα Net Sales

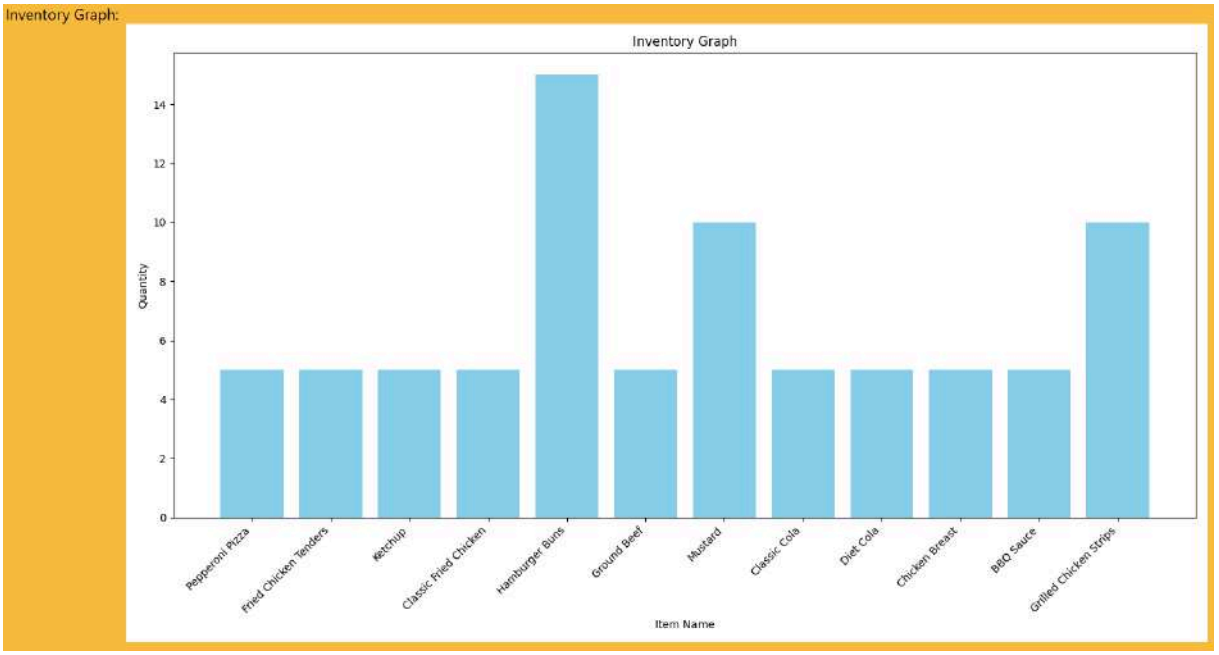
Διάγραμμα Gross Profit: Δείχνει τα συνολικά κέρδη από τις πωλήσεις προϊόντων μετά την αφαίρεση των άμεσων εξόδων παραγωγής, παρέχοντας ένδειξη της αποτελεσματικότητας των βασικών λειτουργιών πριν τα γενικά έξοδα.



Σχήμα 4.15: Διάγραμμα Gross Profit

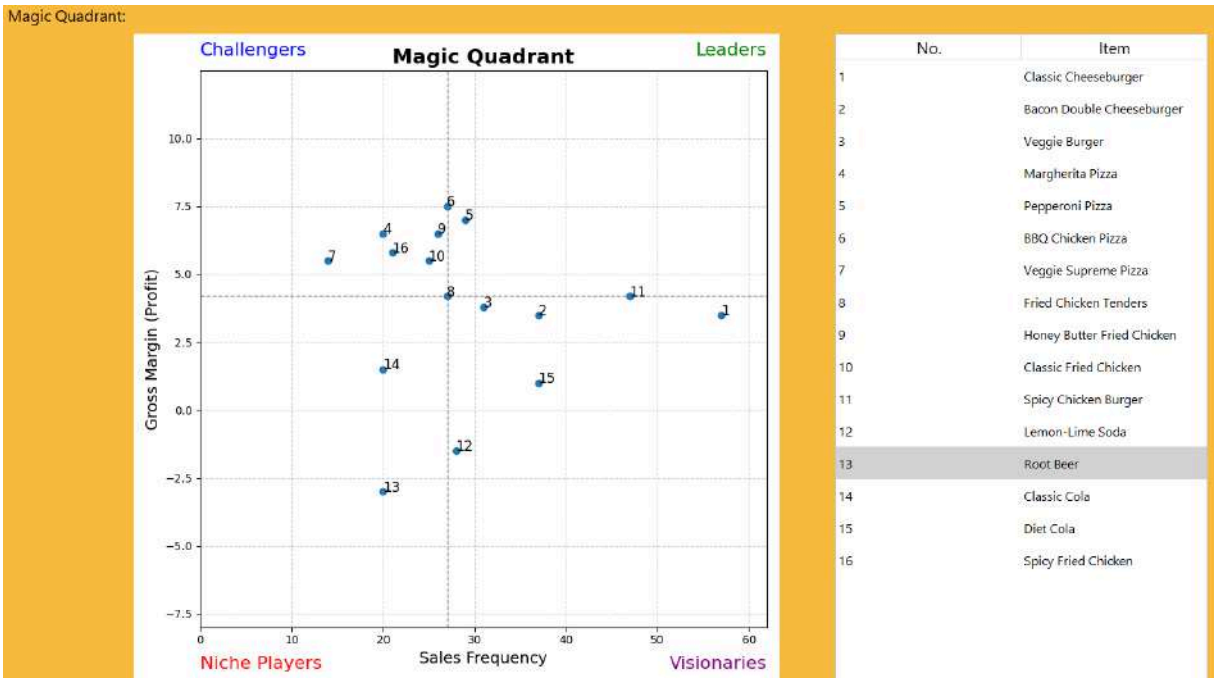
Η υποκαρτέλα Items περιλαμβάνει δύο διαγράμματα:

Inventory Diagram: Παρουσιάζει γραφικά το συνολικό απόθεμα προϊόντων που διαχειρίζεται το σύστημα, επιτρέποντας τον έλεγχο των διαθεσίμων.



Σχήμα 4.16: Διάγραμμα Inventory Diagram

Magic Quadrant: Παρέχει ανάλυση προϊόντων με βάση την απόδοση και τη ζήτηση, απεικονίζοντας τις θέσεις των προϊόντων σε μια τετραγωνική διάταξη για στρατηγική λήψη αποφάσεων.



Σχήμα 4.17: Διάγραμμα Magic Quadrant

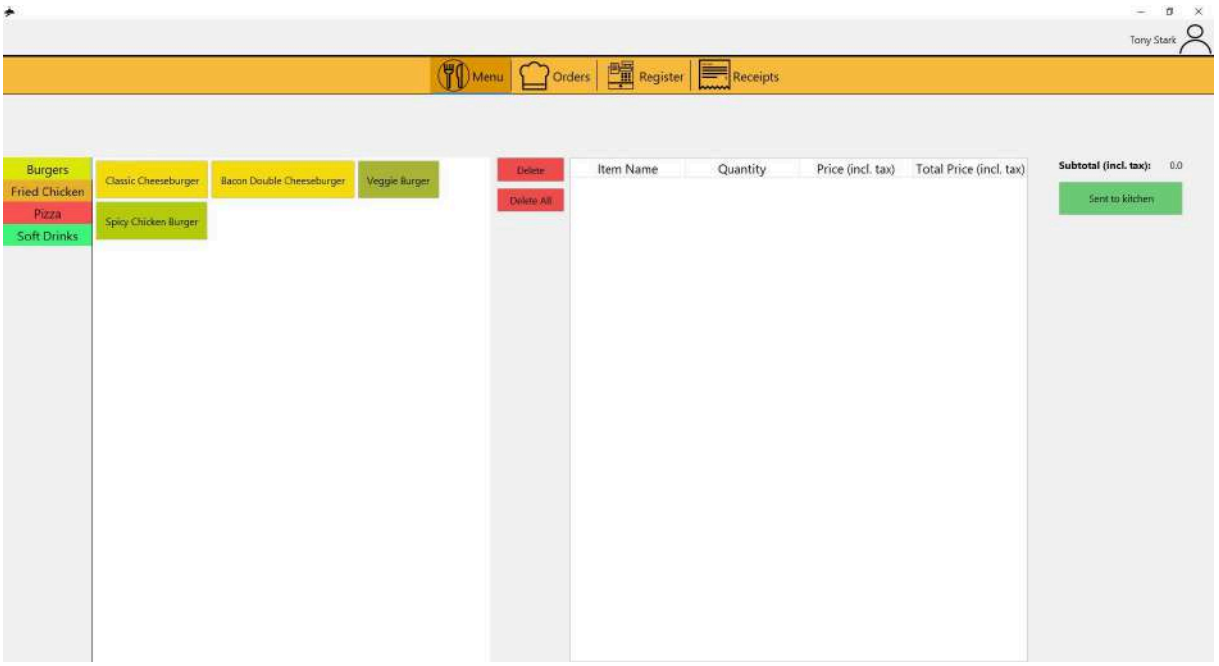
4.3.3. Ταμίας

Η οθόνη του ταμία παρέχει πρόσβαση σε όλες τις βασικές λειτουργίες διαχείρισης της παραγγελίας και είναι οργανωμένη σε τέσσερις κύριες καρτέλες. Οι καρτέλες αυτές αντιστοιχούν στα βασικά στάδια που ακολουθεί μια παραγγελία και είναι οι εξής: Menu, Orders, Register και Receipts.

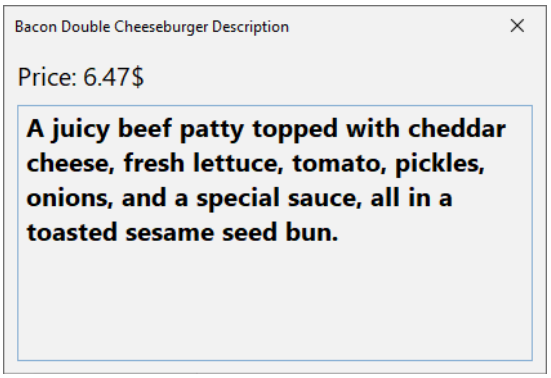
Η καρτέλα Menu περιέχει:

- Μια διεπαφή με καρτέλες, όπου κάθε καρτέλα αντιστοιχεί σε μια κατηγορία προϊόντων και περιλαμβάνει κουμπιά, με κάθε κουμπί να αντιπροσωπεύει ένα προϊόν της επιλεγμένης κατηγορίας.
- Έναν πίνακα που αντιπροσωπεύει την τρέχουσα παραγγελία.
- Τρία κουμπιά, συγκεκριμένα για τη διαγραφή ενός προϊόντος από την παραγγελία (Delete), για το άδειασμα της παραγγελίας (Delete All) και για την αποστολή της παραγγελίας στην κουζίνα (Send to kitchen).
- Μια ετικέτα που αντιστοιχεί στο συνολικο ποσο της παραγγελιας

Κάθε κουμπί προϊόντος, όταν επιλεγεί, εισάγει το αντίστοιχο προϊόν στον πίνακα της παραγγελίας, ενώ σε περίπτωση παρατεταμένης επιλογής ανοίγει παράθυρο με πληροφορίες για το προϊόν.



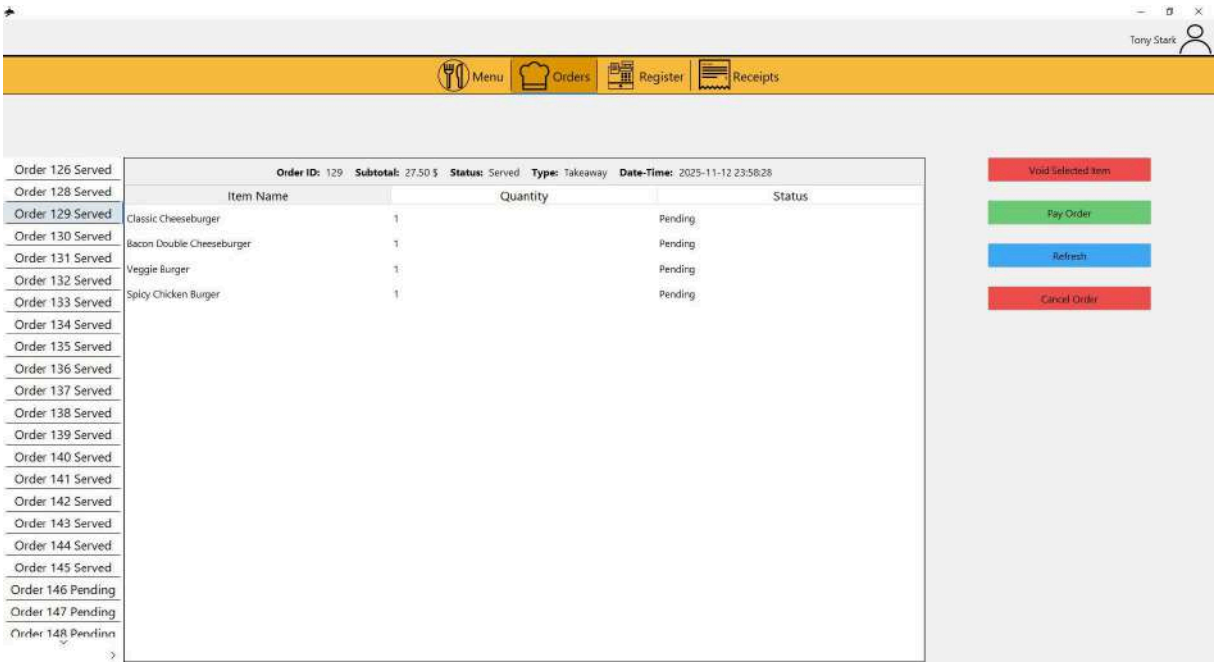
Σχήμα 4.18: Καρτέλα Menu



Σχήμα 4.19: Παράθυρο με πληροφορίες για ένα προϊόν

Η καρτέλα Orders περιέχει:

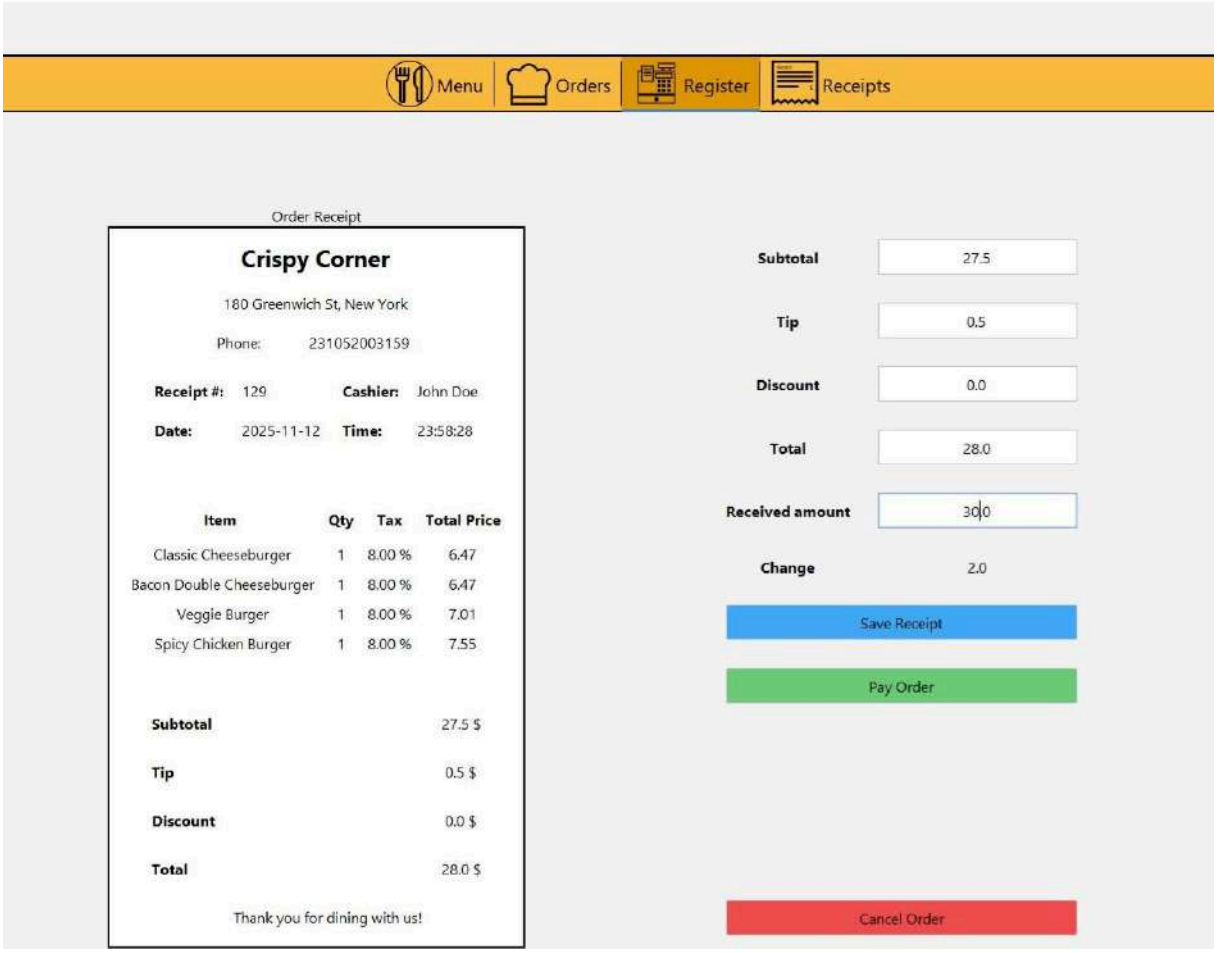
- Μια διεπαφή με καρτέλες, όπου κάθε καρτέλα αντιστοιχεί σε μία παραγγελία. Στο όνομα της καρτέλας εμφανίζεται ο αριθμός της παραγγελίας και η κατάστασή της (Pending/Served), ενώ στο κύριο περιεχόμενο παρουσιάζονται αναλυτικές πληροφορίες. Πιο συγκεκριμένα, εμφανίζονται το μερικό σύνολο (Subtotal), ο τύπος της παραγγελίας (Takeaway/Dine-in), η ημερομηνία και η ώρα, καθώς και επαναλαμβάνονται ο αριθμός και η κατάσταση της παραγγελίας. Επιπλέον, περιλαμβάνεται πίνακας με πληροφορίες για τα προϊόντα της παραγγελίας, όπως το όνομα, η ποσότητα και η κατάσταση κάθε προϊόντος (Pending/Served/Complete/Void).
- Τέσσερα κουμπιά: ένα για την ακύρωση της παραγγελίας (Cancel Order), ένα για την ακύρωση του επιλεγμένου προϊόντος (Void selected item), ένα για την ανανέωση της καρτέλας (Refresh) και ένα για την προώθηση της παραγγελίας στην καρτέλα Register για πληρωμή (Pay Order).



Σχήμα 4.20: Καρτέλα Orders

Η καρτέλα Register περιέχει:

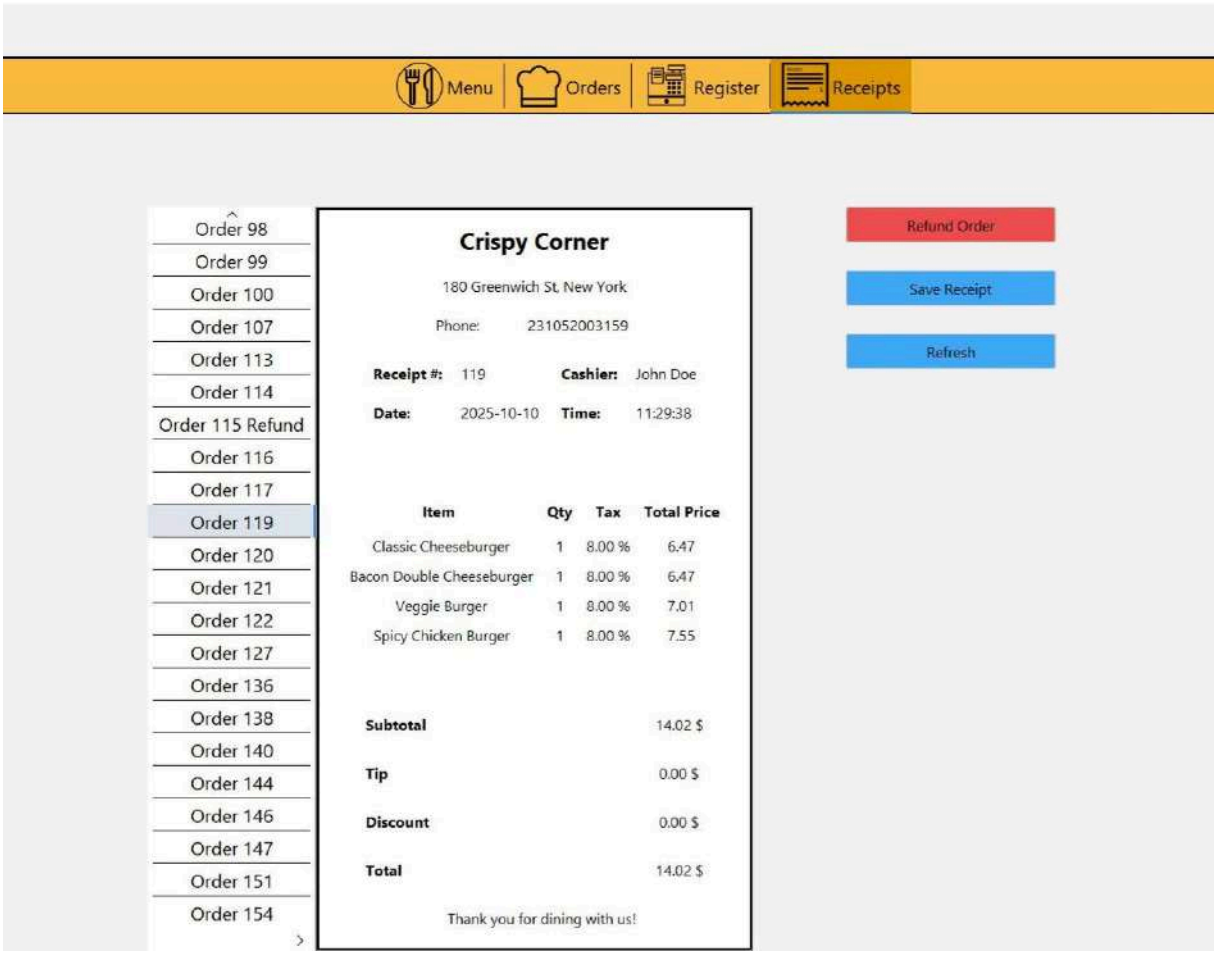
- Προεπισκόπηση της απόδειξης, η οποία περιλαμβάνει πληροφορίες της επιχείρησης (όνομα, διεύθυνση, τηλέφωνο επικοινωνίας), στοιχεία της παραγγελίας (αριθμός παραγγελίας, όνομα ταμία, ημερομηνία και ώρα), καθώς και πληροφορίες για τα προϊόντα (όνομα προϊόντος, ποσότητα, φόρος προστιθέμενης αξίας και συνολικό ποσό)
- Πέντε πεδία εισαγωγής τιμών για το μερικό σύνολο (Subtotal), το φιλοδώρημα (Tip), την έκπτωση (Discount), το συνολικό ποσό (Total) και το ποσό που έδωσε ο πελάτης, για τον υπολογισμό των ρέστων. Αξίζει να σημειωθεί ότι οι τιμες του μερικό σύνολο, του φιλοδωρηματος, της έκπτωσης και το συνολικό ποσό εμφανιζονται στην προεπισκόπηση της απόδειξης
- Τρία κουμπιά: ένα για την ολοκλήρωση της πληρωμής (Pay Order), ένα για την αποθήκευση της απόδειξης (Save Receipt) και ένα για την ακύρωση της παραγγελίας (Cancel Order).



Σχήμα 4.21: Καρτέλα Register

Η καρτέλα Receipts περιέχει:

- Μια διεπαφή με καρτέλες, όπου κάθε καρτέλα αντιστοιχεί σε μία ολοκληρωμένη παραγγελία. Στο όνομα της καρτέλας εμφανίζεται ο αριθμός της παραγγελίας και η κατάστασή της (Refund/No Refund), ενώ στο περιεχόμενο παρουσιάζονται οι ίδιες πληροφορίες με την προεπισκόπηση της απόδειξης της καρτέλας Register.
- Τρία κουμπιά: ένα για την αποθήκευση της απόδειξης (Save Receipt), ένα για την ανανέωση της καρτέλας (Refresh) και ένα για τη δήλωση επιστροφής χρημάτων (Refund Order).

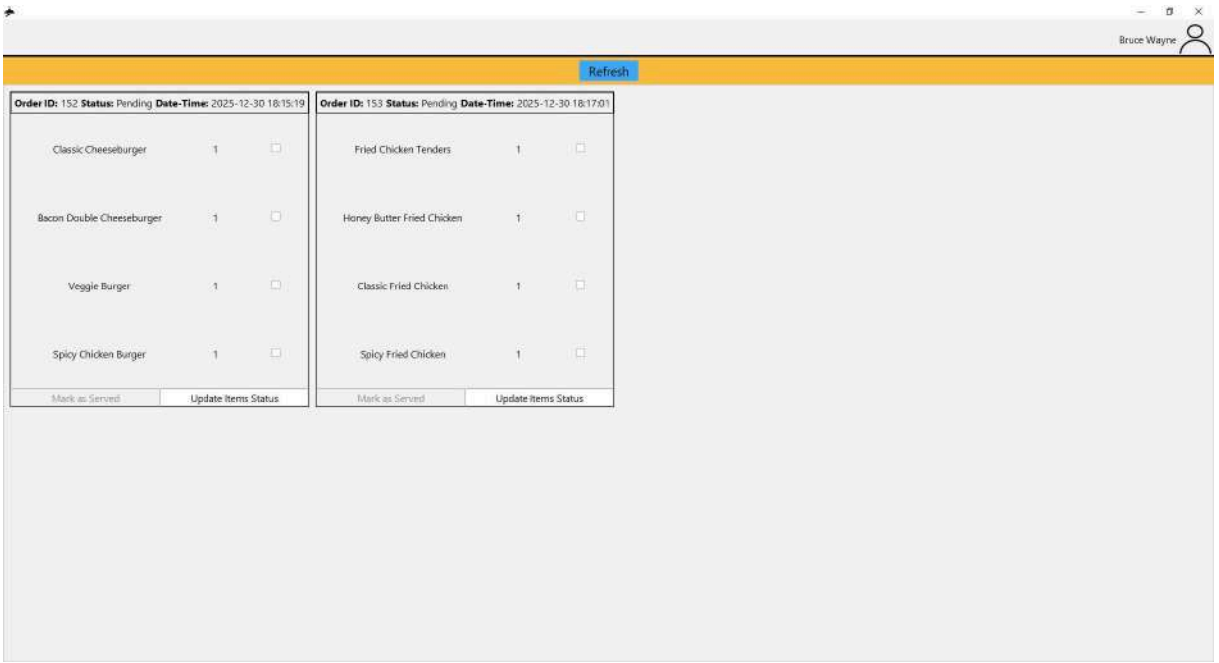


Σχήμα 4.22: Καρτέλα Receipts

4.3.4. Μάγειρας

Η οθόνη του μάγειρα περιέχει μια λίστα από ορθογώνια πλαίσια, τα οποία αντιστοιχούν στις ενεργές παραγγελίες. Κάθε παραγγελία χωρίζεται σε κεφαλίδα και κύριο μέρος. Η κεφαλίδα περιλαμβάνει τον αριθμό της παραγγελίας, την κατάστασή της (Pending/Served), καθώς και την ημερομηνία και την ώρα καταχώρησης. Στο κύριο μέρος της παραγγελίας εμφανίζεται το όνομα κάθε προϊόντος μαζί με την αντίστοιχη ποσότητά του, ενώ δίπλα από κάθε προϊόν υπάρχει ένα πλαίσιο ελέγχου (checkbox), το οποίο χρησιμοποιείται για την ενημέρωση της κατάστασης του προϊόντος.

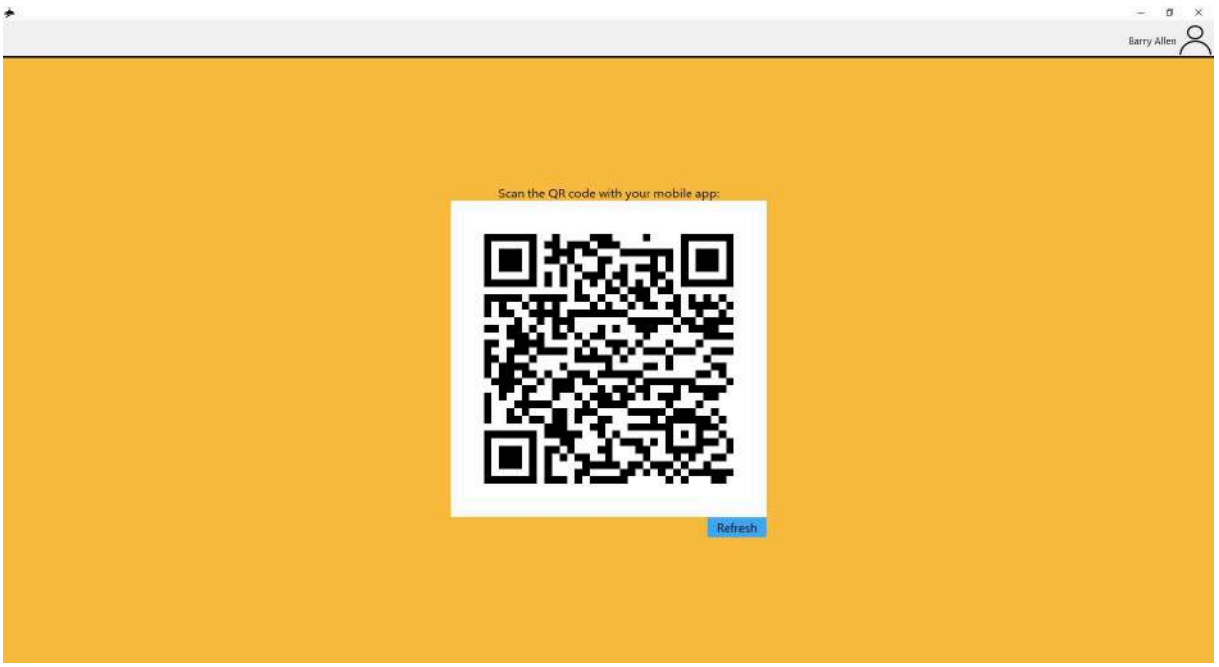
Κάθε ορθογώνιο πλαίσιο παραγγελίας περιέχει δύο κουμπιά. Το πρώτο κουμπί ενημερώνει την κατάσταση των προϊόντων για τα οποία έχει επιλεγεί το πλαίσιο ελέγχου. Το δεύτερο κουμπί ενημερώνει την κατάσταση ολόκληρης της παραγγελίας και μπορεί να ενεργοποιηθεί μόνο στην περίπτωση που έχουν προηγουμένως επιλεγεί όλα τα πλαίσια ελέγχου των προϊόντων της συγκεκριμένης παραγγελίας. Επιπλέον, στο κέντρο της οθόνης υπάρχει ένα κουμπί για την ανανέωση της λίστας των ενεργών παραγγελιών.



Σχήμα 4.23: Οθόνη μάγειρα

4.3.5. Σερβιτόρος

Η οθόνη του σερβιτόρου περιέχει έναν κωδικό ταχείας απόκρισης (Quick Response Code), μέσω του οποίου, όταν σαρωθεί από τον σερβιτόρο, παρέχεται πρόσβαση στην Android εφαρμογή. Επιπλέον, περιλαμβάνει ένα κουμπί για την ανανέωση του κωδικού Q.R., σε περίπτωση που έχει λήξει ο χρονικός περιορισμός ισχύος του.



Σχήμα 4.24: Οθόνη σερβιτόρου

4.4. Android εφαρμογή – Περιγραφή και λειτουργικότητα

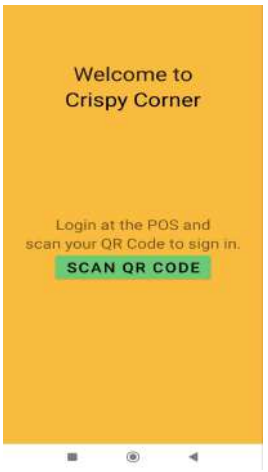
Η Android εφαρμογή αποτελεί συμπληρωματικό μέρος του Συστήματος Διαχείρισης Χώρου Εστίασης και έχει σχεδιαστεί με στόχο την υποστήριξη λειτουργιών που απαιτούν άμεση και φορητή πρόσβαση. Σε αντίθεση με τη desktop εφαρμογή, η οποία καλύπτει το σύνολο των διαχειριστικών διαδικασιών, η Android εφαρμογή επικεντρώνεται κυρίως στη διαχείριση παραγγελιών και στη διαχείριση ταμείου, διευκολύνοντας την καθημερινή λειτουργία του καταστήματος.

Η εφαρμογή απευθύνεται στον σερβιτόρο, ο οποιος χρειάζεται γρήγορη και απλή καταχώρηση δεδομένων κατά τη διάρκεια της εργασίας του. Μέσω ενός φιλικού και εύχρηστου γραφικού περιβάλλοντος, η Android εφαρμογή επιτρέπει την άμεση αλληλεπίδραση με το σύστημα, συμβάλλοντας στη μείωση λαθών, στη βελτίωση της ταχύτητας εξυπηρέτησης και στη συνεχή ενημέρωση των δεδομένων του καταστήματος.

Στην παρούσα ενότητα παρουσιάζονται η δομή, οι βασικές λειτουργίες και η συνολική λειτουργικότητα της Android εφαρμογής, καθώς και ο τρόπος με τον οποίο ενσωματώνεται στο συνολικό πληροφοριακό σύστημα της επιχείρησης.

4.4.1. Είσοδος στην εφαρμογή

Με την εκκίνηση της Android εφαρμογής εμφανίζεται η οθόνη εισόδου, η οποία αποτελεί το αρχικό σημείο αλληλεπίδρασης του χρήστη με το σύστημα. Η οθόνη περιλαμβάνει μήνυμα καλωσορίσματος, ακολουθούμενο από το όνομα της επιχείρησης (στην παρούσα περίπτωση Crispy Corner), καθώς και σύντομες οδηγίες σχετικά με τη διαδικασία σύνδεσης στην εφαρμογή. Επιπλέον, παρέχεται ένα κουμπί με την ένδειξη “Scan QR Code”, το οποίο εκκινεί τη διαδικασία εισόδου. Για να μπορέσει ένας χρήστης να συνδεθεί στην Android εφαρμογή, θα πρέπει αρχικά να διαθέτει τον ρόλο του σερβιτόρου και να έχει συνδεθεί προηγουμένως στη desktop εφαρμογή. Αφού η σύνδεση στη desktop εφαρμογή ολοκληρωθεί με επιτυχία, ο σερβιτόρος επιλέγει το κουμπί της οθόνης εισόδου, το οποίο ενεργοποιεί την κάμερα της κινητής συσκευής και επιτρέπει την φωτογράφιση του κωδικού Q.R. που εμφανίζεται στη desktop εφαρμογή. Η εφαρμογή σαρώνει τον κωδικό Q.R. και πραγματοποιεί τη διαδικασία αυθεντικοποίησης του χρήστη. Σε περίπτωση επιτυχούς αυθεντικοποίησης, ο χρήστης μεταφέρεται στην οθόνη Menu, η οποία παρουσιάζεται στην επόμενη υποενότητα. Αν η αυθεντικοποίηση δεν ολοκληρωθεί επιτυχώς, ο χρήστης επιστρέφει στην οθόνη εισόδου.



Σχήμα 4.25: Οθόνη εισόδου

4.4.2. Οθόνη Menu

Η οθόνη Menu αποτελεί την κύρια οθόνη καταχώρησης παραγγελιών στην Android εφαρμογή. Περιλαμβάνει μια αναδιπλούμενη λίστα, στην οποία εμφανίζονται τα διαθέσιμα τραπέζια του καταστήματος, επιτρέποντας την επιλογή του τραπεζιού στο οποίο αντιστοιχεί η παραγγελία. Παράλληλα, περιλαμβάνει μια διεπαφή με καρτέλες, όπου κάθε καρτέλα αντιστοιχεί σε μία κατηγορία προϊόντων. Κάθε καρτέλα περιέχει μια λίστα με τα προϊόντα της συγκεκριμένης κατηγορίας. Κάθε αντικείμενο της λίστας εμφανίζει το όνομα του προϊόντος, την τιμή του, δύο κουμπιά για την αύξηση και τη μείωση της ποσότητας, καθώς και μία ετικέτα που εμφανίζει το συνολικό κόστος του συγκεκριμένου προϊόντος. Σε περίπτωση επιλογής ενός αντικειμένου της λίστας, το αντικείμενο μεγεθύνεται και εμφανίζεται η περιγραφή του προϊόντος. Στο κάτω μέρος της οθόνης εμφανίζεται μια ετικέτα με το συνολικό ποσό όλων των προϊόντων της παραγγελίας, καθώς και ένα κουμπί για την αποστολή της παραγγελίας στην κουζίνα με την ένδειξη “Send to kitchen”.

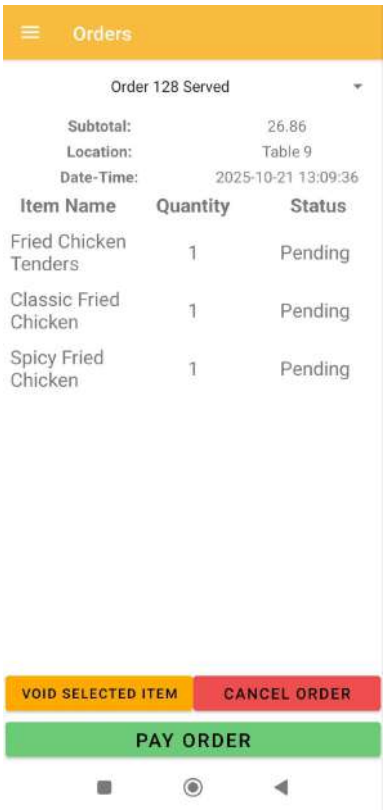
Σχήμα 4.26: Οθόνη Menu

4.4.3. Οθόνη Orders

Η οθόνη Orders περιέχει μια αναδιπλούμενη λίστα, στην οποία εμφανίζονται μόνο οι παραγγελίες που αφορούν τραπέζια. Για κάθε παραγγελία εμφανίζεται ο αριθμός της και η κατάστασή της (Pending/Served). Με την επιλογή μιας παραγγελίας, προβάλλονται επιπλέον πληροφορίες, όπως το μερικό σύνολο, το τραπέζι στο οποίο αντιστοιχεί, η ημερομηνία και η ώρα καταχώρησης, καθώς και μια λίστα με τα προϊόντα της παραγγελίας. Για κάθε προϊόν εμφανίζεται το όνομα, η ποσότητα και η κατάστασή του (Pending/Served/Complete/Void).

Στην οθόνη παρέχονται τρία κουμπιά: ένα για την ακύρωση της παραγγελίας (Cancel Order), ένα για την ακύρωση του επιλεγμένου προϊόντος (Void selected item) και ένα για την προώθηση της παραγγελίας στην οθόνη Register για την ολοκλήρωση της πληρωμής (Pay Order).

Η ανανέωση της καρτέλας πραγματοποιείται με κίνηση του δαχτύλου προς τα κάτω (pull-to-refresh).



Σχήμα 4.27: Οθόνη Orders

4.4.4. Οθόνη Register

Η οθόνη Register αφορά τη διαδικασία ολοκλήρωσης της πληρωμής της παραγγελίας. Περιλαμβάνει μια ετικέτα στην οποία εμφανίζεται ο αριθμός της παραγγελίας, καθώς και μια λίστα με πληροφορίες για τα προϊόντα της παραγγελίας, όπως το όνομα του προϊόντος, η ποσότητα, ο φόρος προστιθέμενης αξίας και το συνολικό ποσό ανά προϊόν.

Επιπλέον, παρέχονται πέντε πεδία εισαγωγής τιμών για το μερικό σύνολο (Subtotal), το φιλοδώρημα (Tip), την έκπτωση (Discount), το συνολικό ποσό (Total) και το ποσό που έδωσε ο πελάτης, το οποίο χρησιμοποιείται για τον υπολογισμό των ρέστων.

Στο κάτω μέρος της οθόνης υπάρχουν δύο κουμπιά: ένα για την ολοκλήρωση της πληρωμής (Pay Order) και ένα για την ακύρωση της παραγγελίας (Cancel Order).

Σχήμα 4.28: Οθόνη Register

4.5. Επικοινωνία με τον διακομιστή (PHP κώδικας – API)

Η επικοινωνία μεταξύ των εφαρμογών του συστήματος (desktop και Android) και του διακομιστή αποτελεί βασικό στοιχείο για τη σωστή λειτουργία του Συστήματος Διαχείρισης Χώρου Εστίασης. Για τον σκοπό αυτό υλοποιήθηκε ένα backend σύστημα βασισμένο στη γλώσσα PHP, το οποίο παρέχει ένα σύνολο υπηρεσιών μέσω RESTful A.P.I., επιτρέποντας την ανταλλαγή δεδομένων με ασφαλή και αποδοτικό τρόπο. Η αρχιτεκτονική που ακολουθήθηκε βασίζεται στο μοντέλο client–server, όπου οι εφαρμογές λειτουργούν ως πελάτες (clients) και ο διακομιστής αναλαμβάνει την επεξεργασία των αιτημάτων, την αλληλεπίδραση με τη βάση δεδομένων και την επιστροφή των κατάλληλων αποτελεσμάτων.

Η ροή επικοινωνίας ξεκινά όταν ο χρήστης εκτελεί μία ενέργεια σε κάποια από τις εφαρμογές, όπως η καταχώριση μιας παραγγελίας ή η ανάκτηση στοιχείων προϊόντων. Η εφαρμογή δημιουργεί ένα HTTP αίτημα προς το αντίστοιχο A.P.I. endpoint του διακομιστή, αποστέλλοντας τα απαραίτητα δεδομένα. Ο διακομιστής λαμβάνει το αίτημα, το επεξεργάζεται μέσω PHP σεναρίων, εκτελεί τις απαιτούμενες λειτουργίες στη βάση δεδομένων και επιστρέφει την αντίστοιχη απάντηση στον πελάτη. Η απάντηση αυτή περιλαμβάνει είτε τα ζητούμενα δεδομένα είτε πληροφορίες σχετικά με την κατάσταση εκτέλεσης της ενέργειας, επιτρέποντας στην εφαρμογή να ενημερώσει κατάλληλα τον χρήστη.

Η υλοποίηση του RESTful API βασίζεται σε μια κεντρική αρχιτεκτονική δρομολόγησης αιτημάτων (routing), η οποία υλοποιείται μέσω ενός κεντρικού αρχείου. Το αρχείο αυτό λειτουργεί ως «χάρτης» (front controller), αναλαμβάνοντας την αρχική παραλαβή κάθε HT.T.P. αιτήματος και τον καθορισμό της κατάλληλης μεθόδου που θα το επεξεργαστεί. Κατά την παραλαβή ενός αιτήματος, το κεντρικό αρχείο αναλύει το endpoint και τη χρησιμοποιούμενη HT.T.P. μέθοδο και, στη συνέχεια, το δρομολογεί το αίτημα στον αντίστοιχο διαχειριστή (handler) ανάλογα με τη λειτουργική κατηγορία στην οποία ανήκει. Για παράδειγμα, ένα αίτημα προς το endpoint /kitchen/getOrdersList προωθείται στον διαχειριστή που αφορά τις λειτουργίες της κουζίνας, όπου υλοποιείται και η αντίστοιχη μέθοδος επεξεργασίας. Στην περίπτωση των λειτουργιών που αφορούν τους ρόλους του manager και του cashier, η δομή του A.P.I. είναι πιο αναλυτική και οργανώνεται σε επιμέρους φακέλους, οι οποίοι αντιστοιχούν σε συγκεκριμένες λειτουργικές ενότητες. Κάθε φάκελος περιλαμβάνει ξεχωριστά αρχεία

για τις αντίστοιχες λειτουργίες. Ενδεικτικά, ένα αίτημα προς το endpoint /cashier/menu/getCategoryList δρομολογείται αρχικά στον διαχειριστή του cashier και στη συνέχεια στον υποφάκελο menu, όπου υλοποιείται η μέθοδος getCategoryList. Η συγκεκριμένη προσέγγιση επιτρέπει την κεντρική διαχείριση της δρομολόγησης των αιτημάτων, τη σαφή οργάνωση του κώδικα και τον διαχωρισμό της επιχειρησιακής λογικής ανά λειτουργική κατηγορία. Παράλληλα, διευκολύνει τη συντήρηση και την επέκταση του συστήματος, καθώς η προσθήκη νέων endpoints μπορεί να πραγματοποιηθεί χωρίς τροποποιήσεις στη βασική δομή του A.P.I..

Κάθε endpoint χρησιμοποιεί συγκεκριμένη HT.T.P. μέθοδο οι οποίες οργανώνονται ως εξής:

- Η μέθοδος GET χρησιμοποιείται για την ανάκτηση δεδομένων από τη βάση δεδομένων σε μορφή JS.O.N.. Σε περιπτώσεις όπου απαιτούνται συγκεκριμένα δεδομένα, τοτε χρησιμοποιούνται παράμετροι, όπως για παράδειγμα στο endpoint /cashier/menu/getItemsList, όπου η παράμετρος category καθορίζει την κατηγορία των προϊόντων που επιστρέφονται.
- Η μέθοδος POST χρησιμοποιείται για την εισαγωγή νέων δεδομένων στη βάση δεδομένων, καθώς και για τη διαδικασία αυθεντικοποίησης των χρηστών κατά την είσοδό τους στις εφαρμογές desktop και Android. Τα δεδομένα αποστέλλονται σε μορφή JS.O.N. στο σώμα (body) του HT.T.P. αιτήματος, όπως συμβαίνει στο endpoint /cashier/register/completeOrder.
- Η μέθοδος PUT χρησιμοποιείται για την επεξεργασία και ενημέρωση υπαρχόντων δεδομένων στη βάση δεδομένων, ακολουθώντας αντίστοιχη τεχνική αποστολής δεδομένων με τη μέθοδο POST.
- Η μέθοδος DELETE χρησιμοποιείται για τη διαγραφή δεδομένων από τη βάση δεδομένων, με παρόμοιο τρόπο αποστολής δεδομένων μέσω JS.O.N..

Η ανταλλαγή δεδομένων μεταξύ client και server πραγματοποιείται αποκλειστικά σε μορφή JS.O.N., λόγω της απλότητας του και της ευρείας υποστήριξής του τόσο σε desktop όσο και σε Android περιβάλλοντα. Τα αιτήματα περιλαμβάνουν οργανωμένα αντικείμενα JS.O.N. με τα δεδομένα εισόδου, ενώ οι απαντήσεις περιέχουν πληροφορίες κατάστασης και, όπου απαιτείται, δεδομένα αποτελεσμάτων. Οι εφαρμογές αναλύουν (parse) τα δεδομένα που λαμβάνουν και τα μετατρέπουν σε εσωτερικές δομές, επιτρέποντας την άμεση εμφάνιση ή περαιτέρω επεξεργασία τους.

Ιδιαίτερη έμφαση δόθηκε στη διαχείριση σφαλμάτων και στην αξιόπιστη ενημέρωση των εφαρμογών σε περιπτώσεις αποτυχίας. Το A.P.I. επιστρέφει κατάλληλους HT.T.P. κωδικούς κατάστασης, όπως επιτυχής εκτέλεση, μη έγκυρα δεδομένα ή αποτυχία σύνδεσης, επιτρέποντας στις εφαρμογές να χειρίζονται σωστά τις εξαιρέσεις και να ενημερώνουν τον χρήστη με κατανοητά μηνύματα.

Τέλος, η ασφάλεια της επικοινωνίας αποτέλεσε κρίσιμο παράγοντα κατά τον σχεδιασμό του συστήματος. Για τον λόγο αυτό εφαρμόστηκαν μηχανισμοί ελέγχου πρόσβασης, ώστε μόνο εξουσιοδοτημένοι χρήστες να μπορούν να εκτελούν συγκεκριμένες ενέργειες. Επιπλέον, πραγματοποιείται έλεγχος και επικύρωση των δεδομένων εισόδου πριν από την εκτέλεση οποιασδήποτε λειτουργίας στη βάση δεδομένων, μειώνοντας τον κίνδυνο λανθασμένων εισαγωγών.

Συνολικά, η υλοποίηση της επικοινωνίας μέσω PHP και RESTful A.P.I. εξασφαλίζει την ομαλή συνεργασία των επιμέρους εφαρμογών του συστήματος, προσφέροντας ευελιξία, επεκτασιμότητα και αξιοπιστία. Ο διαχωρισμός της λογικής του συστήματος από τις εφαρμογές πελάτη επιτρέπει τη μελλοντική αναβάθμιση και προσαρμογή του συστήματος στις ανάγκες ενός σύγχρονου χώρου εστίασης.

4.6. Δημιουργία στατιστικών διαγραμμάτων (Python script)

Στο πλαίσιο του Συστήματος Διαχείρισης Χώρου Εστίασης κρίθηκε απαραίτητη η αξιοποίηση μηχανισμών ανάλυσης δεδομένων, με στόχο την εξαγωγή στατιστικών πληροφοριών που υποστηρίζουν τη διοικητική λήψη αποφάσεων. Για τον σκοπό αυτό υλοποιήθηκε συνδυασμός επεξεργασίας δεδομένων από τη desktop εφαρμογή και δημιουργίας στατιστικών διαγραμμάτων μέσω ανεξάρτητων scripts σε γλώσσα Python.

Κατά τη σύνδεση χρήστη με ρόλο manager στη desktop εφαρμογή, η εφαρμογή αναλαμβάνει την ανάκτηση των απαραίτητων δεδομένων μέσω του RESTful A.P.I. από τη βάση δεδομένων του συστήματος. Τα δεδομένα αφορούν οικονομικά στοιχεία (πωλήσεις, επιστροφές, εκπτώσεις, καθαρά έσοδα και μικτό κέρδος), καθώς και στοιχεία αποθέματος και απόδοσης προϊόντων. Στη συνέχεια, η desktop εφαρμογή επεξεργάζεται τα δεδομένα αυτά, τα οργανώνει σε κατάλληλες δομές και τα μετατρέπει σε μορφή κατάλληλη για περαιτέρω ανάλυση. Αφού ολοκληρωθεί η συλλογή και η επεξεργασία των δεδομένων, ενεργοποιούνται τα Python scripts, τα οποία αναλαμβάνουν αποκλειστικά τη δημιουργία των στατιστικών διαγραμμάτων. Τα scripts χρησιμοποιούν τα επεξεργασμένα δεδομένα που παρέχονται από την εφαρμογή και παράγουν αυτόματα τα αντίστοιχα γραφήματα, τα οποία αποθηκεύονται σε μορφή εικόνας.

Οι παραγόμενες εικόνες εμφανίζονται στις καρτέλες Sales και Items, οι οποίες βρίσκονται κάτω από την καρτέλα Reports της desktop εφαρμογής, παρέχοντας στον διαχειριστή μια συγκεντρωτική και εύληπτη απεικόνιση της οικονομικής και λειτουργικής κατάστασης του καταστήματος. Η αρχική δημιουργία των διαγραμμάτων πραγματοποιείται αυτόματα κατά τη σύνδεση του manager στην εφαρμογή. Επιπλέον, παρέχεται η δυνατότητα επανεκτέλεσης των Python scripts μέσω ειδικού κουμπιού Refresh που διατίθεται στο περιβάλλον της εφαρμογής. Με τη χρήση της λειτουργίας αυτής, ο χρήστης μπορεί να ανανεώσει τα στατιστικά διαγράμματα οποιαδήποτε στιγμή, εξασφαλίζοντας ότι οι απεικονίσεις βασίζονται στα πιο πρόσφατα διαθέσιμα δεδομένα. Η δυνατότητα αυτή είναι ιδιαίτερα χρήσιμη σε περιβάλλοντα με συχνές μεταβολές, όπως η καταχώρηση νέων παραγγελιών ή η ενημέρωση αποθεμάτων κατά τη διάρκεια της λειτουργίας του καταστήματος.

Η συγκεκριμένη προσέγγιση διαχωρίζει με σαφήνεια την επεξεργασία δεδομένων από την οπτικοποίησή τους, ενισχύοντας τη συντηρησιμότητα και την επεκτασιμότητα του συστήματος. Παράλληλα, επιτρέπει την άμεση πρόσβαση σε επικαιροποιημένες πληροφορίες, προσφέροντας στον διαχειριστή αξιόπιστα εργαλεία ανάλυσης και υποστήριξης της στρατηγικής διαχείρισης του χώρου εστίασης.

4.7. Λογική Υλοποίησης και Ροές Λειτουργίας

Στην παρούσα υποενότητα παρουσιάζεται η λογική υλοποίησης του συστήματος και οι βασικές ροές λειτουργίας του. Μέσα από τα διαγράμματα που ακολουθούν απεικονίζεται η αρχιτεκτονική των επιμέρους υποσυστημάτων και η ροή δεδομένων μεταξύ τους, καθώς και η συνολική λειτουργία του προγράμματος. Επιπλέον, παρουσιάζεται ένα ενδεικτικό σενάριο χρήσης, με στόχο τη σαφή κατανόηση του τρόπου με τον οποίο υλοποιούνται και εκτελούνται οι βασικές λειτουργίες της εφαρμογής. Για τη δημιουργία των διαγραμμάτων χρησιμοποιήθηκε το εργαλείο μοντελοποίησης Visual Paradigm σε ηλεκτρονική μορφή.

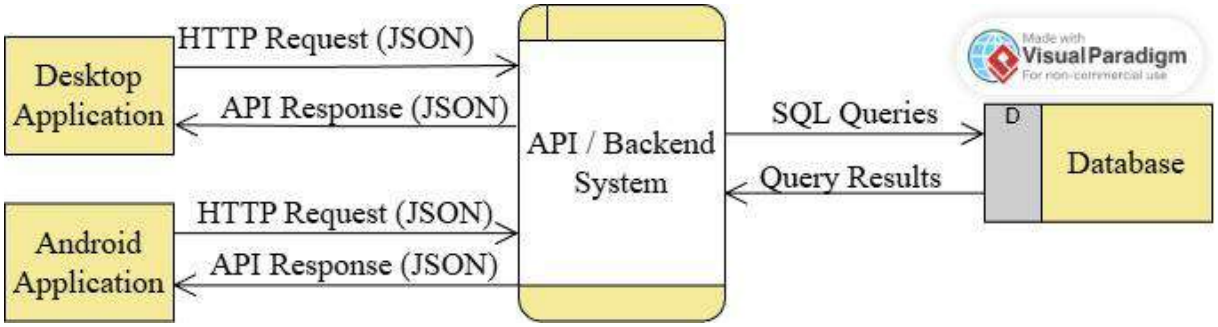
Το παρακάτω Διάγραμμα Ροής Δεδομένων (Data Flow Diagram) απεικονίζει τη συνολική αρχιτεκτονική επικοινωνίας μεταξύ των επιμέρους υποσυστημάτων του Συστήματος Διαχείρισης

Χώρου Εστίασης. Οι εφαρμογές desktop και Android λειτουργούν ως εξωτερικές οντότητες (external entities), οι οποίες αλληλεπιδρούν με το backend σύστημα μέσω διαδικτυακών υπηρεσιών.

Η επικοινωνία πραγματοποιείται μέσω HTTP αιτημάτων, με τη μορφή δεδομένων JSON, τα οποία αποστέλλονται από τις εφαρμογές προς το API του backend. Το backend σύστημα αποτελεί το κεντρικό process του διαγράμματος και είναι υπεύθυνο για την επεξεργασία των αιτημάτων, την εφαρμογή της επιχειρησιακής λογικής και την επικοινωνία με τη βάση δεδομένων. Για την πρόσβαση στα αποθηκευμένα δεδομένα, το backend εκτελεί SQL ερωτήματα προς τη βάση δεδομένων, η οποία αναπαρίσταται ως data store στο διάγραμμα.

Τα αποτελέσματα των ερωτημάτων επιστρέφονται από τη βάση δεδομένων στο backend, το οποίο στη συνέχεια δημιουργεί τις κατάλληλες απαντήσεις σε μορφή JSON και τις αποστέλλει πίσω στις εφαρμογές desktop και Android. Με τον τρόπο αυτό διασφαλίζεται ο διαχωρισμός των ρόλων μεταξύ των εφαρμογών-πελατών και του διακομιστή, καθώς και η ασφαλής και οργανωμένη διαχείριση της ροής των δεδομένων.

Το συγκεκριμένο διάγραμμα αποτυπώνει τη βασική ροή δεδομένων του συστήματος σε υψηλό επίπεδο (Level 0 DFD) και βοηθά στην κατανόηση της λειτουργίας και της συνεργασίας των υποσυστημάτων, ανεξάρτητα από τις λεπτομέρειες υλοποίησης του κώδικα.

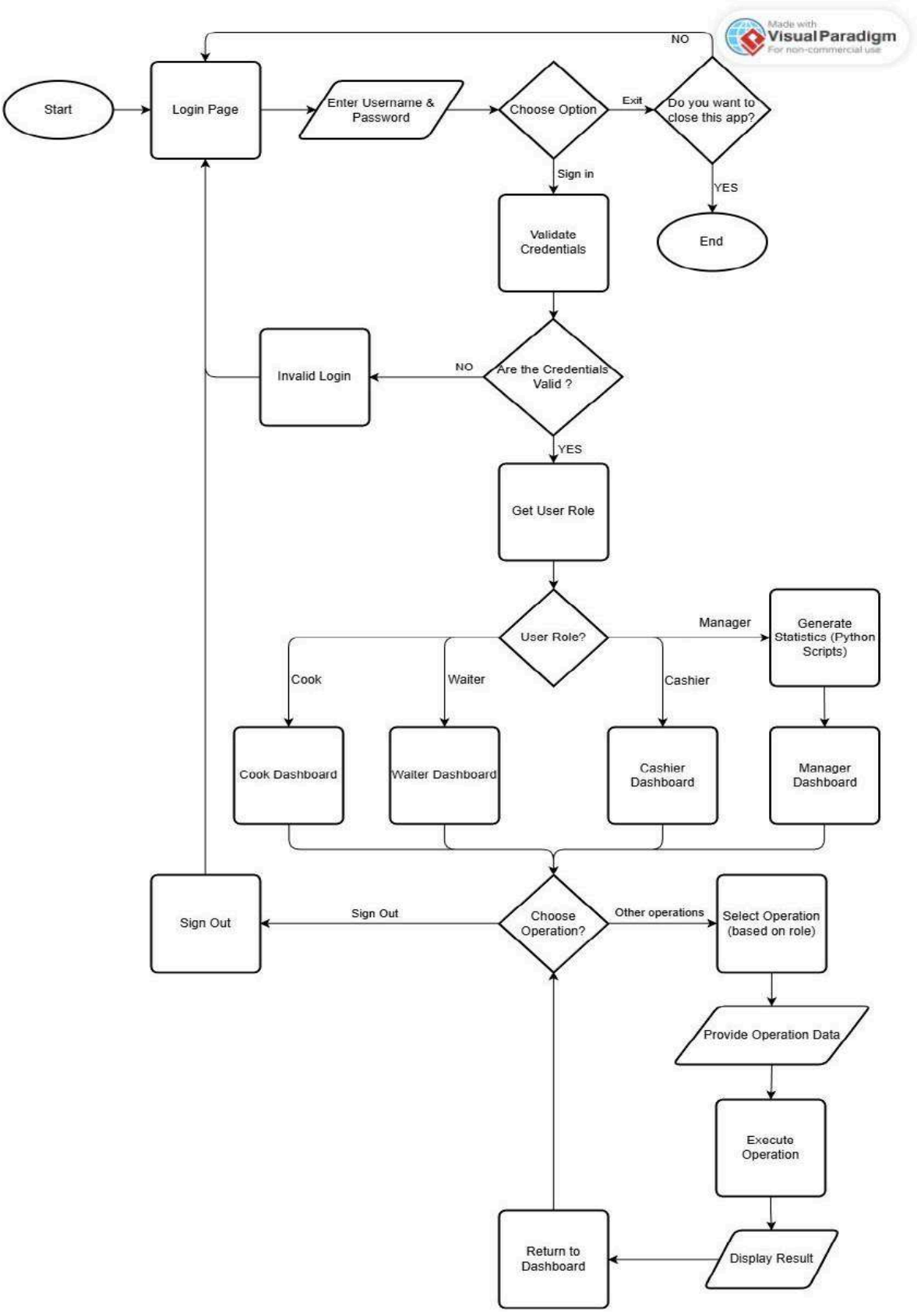


Σχήμα 4.29: Διάγραμμα ροής δεδομένων συστήματος

Το γενικό διάγραμμα βασικής λογικής του συστήματος αποτυπώνει τη συνολική ροή λειτουργίας του Συστήματος Διαχείρισης Χώρου Εστίασης, από την είσοδο του χρήστη έως την εκτέλεση των διαθέσιμων λειτουργιών και την έξοδο από το σύστημα. Η διαδικασία ξεκινά με την αυθεντικοποίηση του χρήστη μέσω της εισαγωγής ονόματος χρήστη και κωδικού πρόσβασης, τα οποία ελέγχονται από το backend σύστημα. Σε περίπτωση επιτυχούς ελέγχου, ανακτάται ο ρόλος του χρήστη και παρέχεται πρόσβαση στο αντίστοιχο περιβάλλον εργασίας (dashboard).

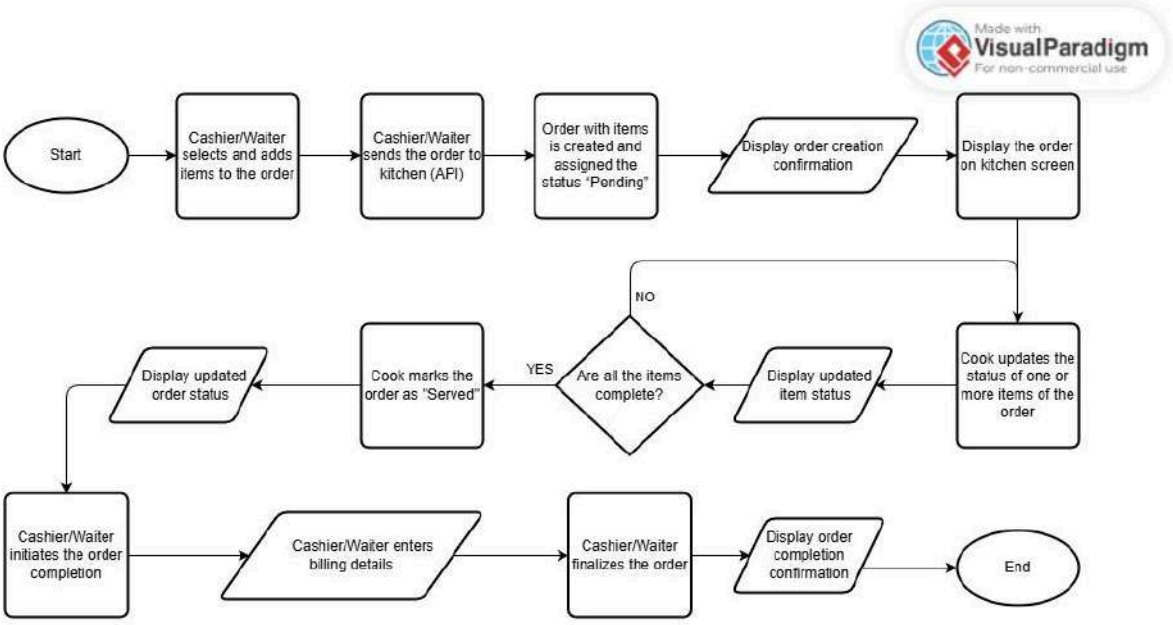
Ανάλογα με τον ρόλο του χρήστη (cook, waiter, cashier ή manager), το σύστημα ενεργοποιεί διαφορετικές λειτουργίες και οθόνες, διασφαλίζοντας τον έλεγχο πρόσβασης και την σωστή κατανομή αρμοδιοτήτων. Ο χρήστης μπορεί να επιλέξει και να εκτελέσει επιμέρους ενέργειες που σχετίζονται με τον ρόλο του, όπως διαχείριση παραγγελιών, ταμειακές πράξεις ή προβολή στατιστικών στοιχείων. Μετά την ολοκλήρωση κάθε ενέργειας, το σύστημα επιστρέφει στο κεντρικό περιβάλλον του ρόλου, επιτρέποντας τη συνέχιση της εργασίας ή την αποσύνδεση από το σύστημα.

Το διάγραμμα ροής παρουσιάζει τη γενική λειτουργία του συστήματος σε υψηλό επίπεδο αφαίρεσης, αναδεικνύοντας τα βασικά στάδια και τις αποφάσεις που λαμβάνονται κατά τη χρήση του. Παράλληλα, τα διαγράμματα δραστηριότητας που παρουσιάστηκαν σε προηγούμενες ενότητες αναλύουν λεπτομερώς τις επιμέρους ενέργειες και διαδικασίες ανά ρόλο χρήστη, προσφέροντας πληρέστερη κατανόηση της εσωτερικής λειτουργίας και της επιχειρησιακής λογικής του συστήματος.



Σχήμα 4.30: Γενικό διάγραμμα βασικής λογικής

Το διάγραμμα που ακολουθεί απεικονίζει τον κύκλο ζωής μιας παραγγελίας στο Σύστημα Διαχείρισης Χώρου Εστίασης μέσω ενός διαγράμματος ροής (Flowchart). Η διαδικασία ξεκινά με την καταχώριση των προϊόντων από τον cashier ή τον waiter μέσω της εφαρμογής και την αποστολή της παραγγελίας στο backend σύστημα μέσω του API. Κατά τη δημιουργία της, η παραγγελία αποθηκεύεται στη βάση δεδομένων και λαμβάνει αρχική κατάσταση «Pending», ενώ τα επιμέρους προϊόντα εμφανίζονται στο σύστημα της κουζίνας προς εκτέλεση. Καθώς τα προϊόντα προετοιμάζονται, η κουζίνα ενημερώνει σταδιακά την κατάσταση των ειδών, μέχρι την ολοκλήρωσή τους. Όταν όλα τα είδη της παραγγελίας ολοκληρωθούν, η παραγγελία χαρακτηρίζεται ως «Served» και ο cashier ή ο waiter προχωρά στη διαδικασία τιμολόγησης και ολοκλήρωσης. Με την καταχώριση των στοιχείων πληρωμής και την τελική επιβεβαίωση, η παραγγελία ολοκληρώνεται και το σύστημα ενημερώνει τον χρήστη για την επιτυχή ολοκλήρωση της διαδικασίας.



Σχήμα 4.31: Διάγραμμα Κύκλου Ζωής Παραγγελίας

Η παρουσίαση της λειτουργικής ροής και της αλληλεπίδρασης των υποσυστημάτων επιτρέπει την κατανόηση της εσωτερικής λειτουργίας του συστήματος, ανεξάρτητα από τη διεπαφή χρήστη ή την υλοποίηση του κώδικα. Μέσω των παραπάνω διαγραμμάτων αποτυπώνεται με σαφήνεια η λογική, η δομή και ο τρόπος εκτέλεσης των βασικών λειτουργιών της εφαρμογής.

4.8. Σύνοψη κεφαλαίου

Στο παρόν κεφάλαιο παρουσιάστηκε αναλυτικά η υλοποίηση του Συστήματος Διαχείρισης Χώρου Εστίασης, καλύπτοντας τόσο τις τεχνολογικές επιλογές όσο και τη λειτουργικότητα των επιμέρους υποσυστημάτων. Αρχικά, περιγράφηκαν οι βασικές τεχνολογίες και τα εργαλεία που χρησιμοποιήθηκαν για την ανάπτυξη της desktop και της Android εφαρμογής, του backend συστήματος, της βάσης δεδομένων, καθώς και των μηχανισμών ανάλυσης δεδομένων και οπτικοποίησης. Στη συνέχεια, αναλύθηκε η desktop εφαρμογή, η οποία αποτελεί το κεντρικό εργαλείο διαχείρισης της επιχείρησης, με έμφαση στις βασικές λειτουργίες που υποστηρίζει και στον ρόλο της στη συλλογή και επεξεργασία δεδομένων. Παράλληλα, παρουσιάστηκε η Android εφαρμογή, η οποία λειτουργεί συμπληρωματικά, παρέχοντας φορητή και άμεση πρόσβαση σε κρίσιμες λειτουργίες,

κυρίως στη διαχείριση παραγγελιών και ταμείου. Ακολούθως, εξετάστηκε η επικοινωνία των εφαρμογών με τον διακομιστή μέσω RESTful A.P.I. σε περιβάλλον PHP, αναδεικνύοντας την αρχιτεκτονική client–server, τη δομή των endpoints, τη χρήση των HT.T.P. μεθόδων και τη σημασία της ασφαλούς και αξιόπιστης ανταλλαγής δεδομένων. Τέλος, παρουσιάστηκε η διαδικασία δημιουργίας στατιστικών διαγραμμάτων μέσω Python scripts, η οποία επιτρέπει την ανάλυση των επιχειρησιακών δεδομένων και την οπτική απεικόνιση της απόδοσης του καταστήματος, υποστηρίζοντας τη λήψη διοικητικών αποφάσεων. Συνολικά, το κεφάλαιο αυτό ανέδειξε τον τρόπο με τον οποίο τα επιμέρους υποσυστήματα συνεργάζονται αρμονικά, διαμορφώνοντας ένα ολοκληρωμένο, επεκτάσιμο και αξιόπιστο πληροφοριακό σύστημα, ικανό να καλύψει τις λειτουργικές ανάγκες ενός σύγχρονου χώρου εστίασης.

Κεφάλαιο 5ο: Συμπεράσματα και Μελλοντικές Επεκτάσεις

5.1. Εισαγωγή

Στο πλαίσιο της παρούσας πτυχιακής εργασίας υλοποιήθηκε ένα ολοκληρωμένο Σύστημα Διαχείρισης Χώρου Εστίασης (Restaurant Management System), με στόχο την υποστήριξη και αυτοματοποίηση βασικών λειτουργιών που απαιτούνται στην καθημερινή λειτουργία ενός σύγχρονου καταστήματος εστίασης. Το σύστημα σχεδιάστηκε ώστε να καλύπτει ανάγκες όπως η διαχείριση προσωπικού, παραγγελιών, ταμείου και αποθέματος, καθώς και η δημιουργία στατιστικών διαγραμμάτων.

Για την επίτευξη του παραπάνω στόχου αναπτύχθηκαν δύο ξεχωριστές εφαρμογές: μία για desktop υπολογιστές και μία για κινητές συσκευές Android. Η desktop εφαρμογή απευθύνεται κυρίως στη διοίκηση και στο προσωπικό του καταστήματος, παρέχοντας δυνατότητες διαχείρισης και εποπτείας, ενώ η Android εφαρμογή επικεντρώνεται στη γρήγορη και άμεση διαχείριση παραγγελιών και ταμείου, υποστηρίζοντας την κινητικότητα του προσωπικού.

Η επικοινωνία μεταξύ των εφαρμογών και της βάσης δεδομένων υλοποιήθηκε μέσω ενός backend συστήματος βασισμένου στη γλώσσα PHP, το οποίο λειτουργεί ως ενδιάμεσο επίπεδο ανταλλαγής δεδομένων. Παράλληλα, για την ανάλυση και οπτικοποίηση των δεδομένων πωλήσεων αξιοποιήθηκε η γλώσσα Python, μέσω της δημιουργίας στατιστικών διαγραμμάτων που προσφέρουν χρήσιμη πληροφόρηση για τη λήψη αποφάσεων.

Συνολικά, το σύστημα που αναπτύχθηκε συμβάλλει στη βελτίωση της οργάνωσης και της αποδοτικότητας ενός χώρου εστίασης, προσφέροντας ένα ενιαίο πληροφοριακό σύστημα που συνδυάζει λειτουργικότητα, ευχρηστία και δυνατότητες επεκτασιμότητας.

5.2. Συμπεράσματα και αξιολόγηση αποτελεσμάτων

Με την ολοκλήρωση της παρούσας πτυχιακής εργασίας διαπιστώνεται ότι οι αρχικοί στόχοι που είχαν τεθεί επιτεύχθηκαν σε ικανοποιητικό βαθμό. Το αναπτυγμένο Σύστημα Διαχείρισης Χώρου Εστίασης καλύπτει τις βασικές λειτουργικές ανάγκες ενός καταστήματος εστίασης και προσφέρει ένα συνεκτικό και λειτουργικό περιβάλλον διαχείρισης τόσο μέσω της desktop όσο και της Android εφαρμογής.

Η υλοποίηση των επιμέρους υποσυστημάτων, όπως η διαχείριση προσωπικού, παραγγελιών, ταμείου και αποθέματος, συνέβαλε στη δημιουργία ενός ολοκληρωμένου πληροφοριακού συστήματος που μειώνει την πολυπλοκότητα των καθημερινών διαδικασιών και περιορίζει την πιθανότητα ανθρώπινων λαθών. Παράλληλα, η ύπαρξη διαφορετικών ρόλων χρηστών ενισχύει την ασφάλεια και την ορθολογική κατανομή αρμοδιοτήτων εντός του συστήματος.

Ιδιαίτερη σημασία παρουσιάζει η αξιοποίηση των στατιστικών διαγραμμάτων, τα οποία προκύπτουν από τη λειτουργία του συστήματος που δημιουργούνται με τη χρήση της γλώσσας Python. Τα διαγράμματα αυτά παρέχουν πολύτιμη πληροφόρηση σχετικά με τις πωλήσεις και τη συνολική απόδοση του καταστήματος, διευκολύνοντας τη λήψη αποφάσεων σε διοικητικό επίπεδο.

Σε τεχνικό επίπεδο, η ανάπτυξη του backend συστήματος με τη χρήση της γλώσσας PHP επέτρεψε την ομαλή επικοινωνία μεταξύ των εφαρμογών και της βάσης δεδομένων, εξασφαλίζοντας την ακεραιότητα και τη συνέπεια των δεδομένων. Η αρχιτεκτονική που ακολουθήθηκε προσφέρει τη δυνατότητα μελλοντικών επεκτάσεων, ενώ η διαχωρισμένη ανάπτυξη των εφαρμογών διευκολύνει τη συντήρηση και τη βελτίωση του συστήματος.

Τέλος, η εκπόνηση της παρούσας εργασίας συνέβαλε ουσιαστικά στην απόκτηση εμπειρίας σε όλα τα στάδια ανάπτυξης ενός πληροφοριακού συστήματος, από την ανάλυση απαιτήσεων και τον σχεδιασμό

έως την υλοποίηση και τον έλεγχο λειτουργίας. Η εμπειρία αυτή ανέδειξε τη σημασία του σωστού σχεδιασμού, της τεκμηρίωσης και της επιλογής κατάλληλων τεχνολογιών για την επιτυχή ολοκλήρωση ενός σύνθετου λογισμικού.

5.3. Μελλοντικές επεκτάσεις και προτάσεις βελτίωσης

Παρόλο που το αναπτυγμένο Σύστημα Διαχείρισης Χώρου Εστίασης καλύπτει τις βασικές λειτουργικές ανάγκες ενός καταστήματος, υπάρχουν αρκετές δυνατότητες περαιτέρω βελτίωσης και επέκτασης, οι οποίες θα μπορούσαν να ενισχύσουν τη λειτουργικότητα και τη χρηστικότητά του σε πραγματικές συνθήκες χρήσης.

Σε επίπεδο λειτουργικών επεκτάσεων, θα μπορούσε να προστεθεί υποστήριξη online παραγγελιών για υπηρεσίες delivery, επιτρέποντας στους πελάτες να πραγματοποιούν παραγγελίες μέσω διαδικτύου. Παράλληλα, θα μπορούσε να ενσωματωθεί σύστημα διαχείρισης πελατών, (Customer Relationship Management) το οποίο θα καταγράφει το ιστορικό παραγγελιών και θα υποστηρίζει προγράμματα επιβράβευσης ή προσφορών. Επιπλέον, το σύστημα θα μπορούσε να επεκταθεί με λειτουργίες που αφορούν την καλύτερη παρακολούθηση της λειτουργίας του καταστήματος, όπως η καταγραφή βαρδιών προσωπικού, η παρακολούθηση της απόδοσης ανά υπάλληλο και η αυτόματη ειδοποίηση για χαμηλά επίπεδα αποθέματος. Τέτοιες λειτουργίες θα συνέβαλαν στη βελτίωση της οργάνωσης και στον αποτελεσματικότερο προγραμματισμό των καθημερινών εργασιών.

Στον τομέα της ανάλυσης δεδομένων, θα μπορούσε να προστεθεί μεγαλύτερη ποικιλία στατιστικών διαγραμμάτων και αναφορών, όπως ανάλυση πωλήσεων ανά κατηγορία προϊόντων και επισκέψεις του καταστήματος ανά χρονική περίοδο. Επιπλέον, θα μπορούσαν να βελτιωθούν τα ήδη υπάρχοντα διαγράμματα, καθιστώντας τα πιο διαδραστικά, για παράδειγμα μέσω της δυνατότητας επιλογής συγκεκριμένου χρονικού διαστήματος. Η παροχή πιο αναλυτικών και δυναμικών στατιστικών στοιχείων θα διευκόλυνε τη διοίκηση του καταστήματος στη λήψη τεκμηριωμένων αποφάσεων και στον εντοπισμό τάσεων στη λειτουργία του.

Συνολικά, οι παραπάνω επεκτάσεις καταδεικνύουν ότι το σύστημα που αναπτύχθηκε μπορεί να αποτελέσει μια σταθερή βάση για περαιτέρω εξέλιξη και προσαρμογή στις αυξανόμενες ανάγκες ενός σύγχρονου χώρου εστίασης.

5.4. Σύνοψη κεφαλαίου

Στο παρόν κεφάλαιο παρουσιάστηκαν τα συνολικά συμπεράσματα που προέκυψαν από την υλοποίηση του Συστήματος Διαχείρισης Χώρου Εστίασης, καθώς και η αξιολόγηση των αποτελεσμάτων του. Από την ανάλυση που πραγματοποιήθηκε, προκύπτει ότι το σύστημα ανταποκρίνεται στους αρχικούς στόχους της πτυχιακής εργασίας και παρέχει μια ολοκληρωμένη λύση για την υποστήριξη των βασικών λειτουργιών ενός χώρου εστίασης. Παράλληλα, προτάθηκαν μελλοντικές επεκτάσεις και βελτιώσεις, οι οποίες μπορούν να ενισχύσουν περαιτέρω τη λειτουργικότητα και την αποδοτικότητα του συστήματος, καθιστώντας το μια αξιόπιστη βάση για περαιτέρω ανάπτυξη και αξιοποίηση σε πραγματικές συνθήκες.

Βιβλιογραφία

Internet Site

[1] K. Kuligowski, “Small Business Guide to a Restaurant Management System,” *business.com*, Nov. 26, 2024. [Online]. Available: https://www.business.com/articles/restaurant-management-system-guide/. [Accessed: Oct. 27, 2025].

[2] 7shifts Staff, “Restaurant Management Systems: Everything You Need to Know,” *7shifts,* Feb. 4, 2025. [Online]. Available: https://www.7shifts.com/blog/restaurant-management-system/. [Accessed: Nov. 3, 2025].

[3] Restaurant365, “Guide to Restaurant Management Systems,” *Restaurant365.* [Online]. Available: https://www.restaurant365.com/blog/restaurant-management-system-guide/. [Accessed: Oct. 27, 2025].

[4] “8 Reasons to Use a Restaurant Management System in 2025,” *foodics,* Apr. 13, 2025. [Online]. Available: https://www.foodics.com/8-reasons-to-use-a-restaurant-management-system/. [Accessed: Oct. 31, 2025].

[5] “Restaurant Management System Explained: Benefits And Key Features,” *app2food.* [Online]. Available: https://www.app2food.com/blog/restaurant-management-system-explained-benefits-and-key-features/. [Accessed: Oct. 25, 2025].

[6] “What is a Restaurant Management System and What Are Its Features?,” *eptera.* [Online]. Available: https://eptera.com/blog/restaurant-management-system-features/index.html. [Accessed: Oct. 24, 2025].

[7] Fatma Humayun, “Top 10 Advantages of Restaurant Management System,” *bdtask*, Jan. 21, 2025. [Online]. Available: https://www.bdtask.com/blog/advantages-of-restaurant-management-system. [Accessed: Nov. 4, 2025].

[8] “Restroworks: Restaurant Management Platform,” *Restroworks*. [Online]. Available: https://www.restroworks.com/. [Accessed: Nov. 7, 2025].

[9] “TouchBistro: Restaurant Management POS System,” *TouchBistro.* [Online]. Available: https://www.touchbistro.com/. [Accessed: Nov. 8, 2025].

[10] “Restaurant Management Software | Restaurant365,” *restaurant365*. [Online]. Available: https://www.restaurant365.com/?nab=0. [Accessed: Nov. 9, 2025].

[11] Object Management Group (OMG), “UML History FAQ,” *omg.org*. [Online]. Available: https://www.omg.org/uml/uml-history-faq.htm. [Accessed: Nov. 15, 2025].

[12] Visual Paradigm, “How to Write Effective Use Cases?,” *visual paradigm,* Jan. 27, 2016. [Online]. Available: https://www.visual-paradigm.com/tutorials/writingeffectiveusecase.jsp. [Accessed: Nov. 15, 2025].

[13] David Garlan, “Software architecture: a travelogue,” *ACM Digital Library*, May 31, 2014. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/2593882.2593886. [Accessed: Nov. 20, 2025].

[14] Britannica Editors, “client-server architecture,” *Encyclopedia Britannica*, Oct. 31 2025. [Online]. Available: https://www.britannica.com/technology/client-server-architecture. [Accessed: Nov. 20, 2025].

[15] “What is three-tier architecture?,” ibm, [Online]. Available: https://www.ibm.com/think/topics/three-tier-architecture. [Accessed: Nov. 20, 2025].

[16] Visual Paradigm, “What is Class Diagram?” *visual paradigm*. [Online]. Available: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/. [Accessed: Nov. 24, 2025].

[17] Microsoft, “What is Java?,” *Microsoft Azure*. [Online]. Available: https://azure.microsoft.com/en-us/resources/cloud-computing-dictionary/what-is-java-programming-language. [Accessed: Dec 16, 2025].

[19] FormDev Software, “FlatLaf - Flat Look and Feel,” *Formdev*. [Online]. Available: https://www.formdev.com/flatlaf/. [Accessed: Dec 16, 2025].

[20] Robert Eckstein, “Java SE Application Design With MVC,” *Oracle*, Mar. 2007. [Online]. Available: https://www.oracle.com/technical-resources/articles/javase/mvc.html. [Accessed: Dec 16, 2025].

[21] json, “Introducing JSON,” *json*. [Online]. Available: https://www.json.org/json-en.html. [Accessed: Dec 16, 2025].

[22] ZXing, “ZXing (“Zebra Crossing”) barcode scanning library for Java, Android,” *GitHub repository*. [Online]. Available: https://github.com/zxing/zxing. [Accessed: Dec 16, 2025].

[23] Oracle, “XML Overview,” *Oracle*. [Online]. Available: https://docs.oracle.com/cd/E13222_01/wls/docs100/xml/intro.html. [Accessed: Dec 20, 2025].

[24] Microsoft, “Model-View-ViewModel (MVVM),” *Microsoft*. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm. [Accessed: Dec 20, 2025].

[25] Square, “Retrofit,” *Square*. [Online]. Available: https://square.github.io/retrofit/. [Accessed: Dec 20, 2025].

[26] Square, “OkHttp,” *Square*. [Online]. Available: https://square.github.io/okhttp/. [Accessed: Dec 20, 2025].

[27] Google, “gson,” *GitHub repository*. [Online]. Available: https://github.com/google/gson. [Accessed: Dec 20, 2025].

[28] Google, “Barcode scanning,” *Google for Developers*. [Online]. Available: https://developers.google.com/ml-kit/vision/barcode-scanning. [Accessed: Dec 20, 2025].

[29] php, “What is PHP and what can it do?,” *php*. [Online]. Available: https://www.php.net/manual/en/introduction.php. [Accessed: Dec 23, 2025].

[30] google, “What is REST API?,” *Google*. [Online]. Available: https://cloud.google.com/discover/what-is-rest-api. [Accessed: Dec 23, 2025].

[31] Jeffrey Erickson, “MySQL: Understanding What It Is and How It’s Used,” Oracle, Aug. 29, 2024. [Online]. Available: https://www.oracle.com/africa/mysql/what-is-mysql/. [Accessed: Dec 23, 2025].

[32] André Munro, “Python,” *Britannica*, Nov. 27, 2025. [Online]. Available: https://www.britannica.com/technology/Python-computer-language. [Accessed: Dec 27, 2025].

[33] pandas, “Package overview,” *pandas*. [Online]. Available: https://pandas.pydata.org/pandas-docs/stable/getting_started/overview.html. [Accessed: Dec 27, 2025].

[34] numpy, “What is NumPy?,” *numpy*. [Online]. Available: https://numpy.org/doc/stable/user/whatisnumpy.html#whatisnumpy. [Accessed: Dec 27, 2025].

[35] pydata, “Matplotlib,” *pydata*. [Online]. Available: https://pydata.org/project/matplotlib/. [Accessed: Dec 27, 2025].

[36] adjusttext, “Welcome to the documentation for adjustText!,” *adjusttext*. [Online]. Available: https://adjusttext.readthedocs.io/en/latest/. [Accessed: Dec 27, 2025].

Βιβλία

[18] M. Loy, R. Eckstein, D. Wood, J. Elliott and B. Cole, *Java Swing.* Sebastopol, CA, USA: O’Reilly & Associates, Inc., 2003.

Λογισμικό

[37] JetBrains, IntelliJ IDEA. [Online]. Available: https://www.jetbrains.com/idea/. [Accessed: Dec 30, 2025].

[38] Google, Android Studio. [Online]. Available: https://developer.android.com/studio. [Accessed: Dec 30, 2025].

[39] Microsoft, Visual Studio Code. [Online]. Available: https://code.visualstudio.com/. [Accessed: Dec 30, 2025].

[40] T. Kosse, FileZilla. [Online]. Available: https://filezilla-project.org/. [Accessed: Dec 30, 2025].

[41] Oracle Corporation, MySQL Workbench. [Online]. Available: https://www.mysql.com/products/workbench/. [Accessed: Dec 30, 2025].

[42] Visual Paradigm International, Visual Paradigm. [Online]. Available: https://www.visual-paradigm.com/. [Accessed: Dec 30, 2025].

Παράρτημα Α: Κώδικας Desktop Εφαρμογής

Α.1 Δομή έργου

```
C:.
├───controller
│   ├───category
│   ├───employee
│   ├───inventory
│   ├───item
│   ├───kitchen
│   ├───login
│   ├───manager
│   ├───menu
│   ├───order
│   ├───pos
│   ├───receipts
│   ├───register
│   ├───reports
│   ├───restaurant
│   ├───role
│   ├───transaction
│   └───waiter
├───entities
├───META-INF
├───model
│   ├───category
│   ├───employee
│   ├───inventory
│   ├───item
│   ├───kitchen
│   ├───login
│   ├───manager
│   ├───menu
│   ├───order
│   ├───receipts
│   ├───register
│   ├───reports
│   ├───restaurant
│   ├───role
│   ├───transaction
│   └───waiter
├───python
├───resources
├───util
└───view
    ├───cashier
    ├───category
    ├───employee
    ├───inventory
    ├───item
    ├───kitchen
    ├───login
    ├───manager
    ├───menu
    ├───order
    ├───pos
    ├───receipts
    ├───register
    ├───reports
    ├───restaurant
    ├───role
    ├───transaction
    └───waiter
```

A.2 Entry Point

PointOfSale.java

```java
import controller.pos.POSController;

import javax.swing.*;

//TIP To <b>Run</b> code, press <shortcut actionId="Run"/> or
// click the <icon src="AllIcons.Actions.Execute"/> icon in the gutter.
public class PointOfSale {  Giorgos-Karachristos
    public static void main(String[] args) { SwingUtilities.invokeLater(POSController::new); }
}
```

A.3 Controller

A.3.1 Category Controller

CategoryController.java

```java
package controller.category;

import entities.Category;
import model.category.CategoryDAO;
import model.category.CategoryModel;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public class CategoryController {  18 usages  Giorgos-Karachristos

    private final CategoryDAO categoryDAO;  7 usages

    public CategoryController() { this.categoryDAO = new CategoryModel(); }

    public void getCategoryTable(DefaultTableModel tableModel) { categoryDAO.getCategoryTable(tableModel); }

    public List<Category> getCategoryList(DefaultComboBoxModel<String> boxModel) {  2 usages  Giorgos-Karachristos
        return categoryDAO.getCategoryList(boxModel);
    }

    public void getCategoryName(DefaultComboBoxModel<String> boxModel) { categoryDAO.getCategoryName(boxModel); }

    public String insertCategory(String name, double tax, String color, String isActive) {  1 usage  Giorgos-Karachris
        return categoryDAO.insertCategory(name, tax, color, isActive);
    }

    public String updateCategory(String name, double tax, String color, String isActive) {  1 usage  Giorgos-Karachris
        return categoryDAO.updateCategory(name, tax, color, isActive);
    }

    public String deleteCategory(String name) { return categoryDAO.deleteCategory(name); }
}
```

A.3.2 Employee Controller

EmployeeController.java

```
EmployeeController.java
package controller.employee;

import entities.Employee;
import model.employee.EmployeeDAO;
import model.employee.EmployeeModel;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public class EmployeeController {  12 usages  Giorgos-Karachristos *
    private final EmployeeDAO employeeDAO;  6 usages

    public EmployeeController() { this.employeeDAO = new EmployeeModel(); }

    public void getEmployeeTable(DefaultTableModel tableModel) { employeeDAO.getEmployeeTable(tableModel); }

    public List<Employee> getEmployeeList(DefaultComboBoxModel<String> boxModel) {  2 usages  Giorgos-Karachristos
        return employeeDAO.getEmployeeList(boxModel);
    }

    public String insertEmployee(String name, String surname, String phone, String email,  1 usage  Giorgos-Karach
                                 String role, String ssn, String idn, String username, String password) {
        return employeeDAO.insertEmployee(name, surname, phone, email, role, ssn, idn, username, password);
    }

    public String updateEmployee(int id, String name, String surname, String phone, String email,  1 usage  Gior
                                 String role, String ssn, String idn, String username, String password) {
        return employeeDAO.updateEmployee(id, name, surname, phone, email, role, ssn, idn, username, password);
    }

    public String deleteEmployee(int id) { return employeeDAO.deleteEmployee(id); }
}
```

A.3.3 Inventory Controller

InventoryController.java

```
InventoryController.java
package controller.inventory;

import entities.InventoryItem;
import entities.Item;
import model.inventory.InventoryDAO;
import model.inventory.InventoryModel;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public class InventoryController {  15 usages  Giorgos-Karachristos
    private final InventoryDAO inventoryDAO;  7 usages

    public InventoryController() {  5 usages  Giorgos-Karachristos
        this.inventoryDAO = new InventoryModel();
    }

    public void getInventoryTable(DefaultTableModel tableModel) {  4 usages  Giorgos-Karachristos
        inventoryDAO.getInventoryTable(tableModel);
    }

    public List<InventoryItem> getInventoryList(DefaultComboBoxModel<String> boxModel) {  3 usages  Giorgos-Kar
        return inventoryDAO.getInventoryList(boxModel);
    }

    public List<Item> getInventoryComboBox(DefaultComboBoxModel<String> boxModel) {  1 usage  Giorgos-Karachristos
        return inventoryDAO.getInventoryComboBox(boxModel);
    }

    public String insertInventory(int id, String location, int quantity, int minQuantity, int maxQuantity) {
        return inventoryDAO.insertInventory(id, location, quantity, minQuantity, maxQuantity);
    }

    public String updateInventory(int id, String location, int quantity, int minQuantity, int maxQuantity) {
        return inventoryDAO.updateInventory(id, location, quantity, minQuantity, maxQuantity);
    }

    public String deleteInventory(int id) {  1 usage  Giorgos-Karachristos
        return inventoryDAO.deleteInventory(id);
    }
}
```

A.3.4 Item Controller

ItemController.java

```java
package controller.item;

import entities.Item;
import model.item.ItemDAO;
import model.item.ItemModel;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public class ItemController {  12 usages  Giorgos-Karachristos
    private final ItemDAO itemDAO;  6 usages

    public ItemController() { this.itemDAO = new ItemModel(); }

    public void getItemTable(DefaultTableModel tableModel) { itemDAO.getItemTable(tableModel); }

    public List<Item> getItemList(DefaultComboBoxModel<String> boxModel) { return itemDAO.getItemList(boxModel); }

    public String insertItem(String code, String name, String description, double price, double cost, String category, String color, String isActive) {  1 usage
        return itemDAO.insertItem(code, name, description, price, cost, category, color, isActive);
    }

    public String updateItem(int id, String code, String name, String description, double price, double cost, String category, String color, String isActive) {
        return itemDAO.updateItem(id, code, name, description, price, cost, category, color, isActive);
    }

    public String deleteItem(int id) { return itemDAO.deleteItem(id); }
}
```

A.3.5 Kitchen Controller

KitchenController.java

```java
package controller.kitchen;

import entities.KitchenItem;
import entities.Order;
import model.kitchen.KitchenDAO;
import model.kitchen.KitchenModel;

import java.util.List;

public class KitchenController {  6 usages  Giorgos-Karachristos

    private final KitchenDAO model;  5 usages

    public KitchenController() { this.model = new KitchenModel(); }

    public List<Order> getOrdersList() { return model.getOrdersList(); }

    public List<KitchenItem> getCookList(String orderID) { return model.getCookList(orderID); }

    public String markAsServed(String orderID) { return model.markAsServed(orderID); }

    public String updateStatus(String orderID, String item) { return model.updateStatus(orderID, item); }
}
```

A.3.6 Login Controller

LoginController.java

```java
LoginController.java
package controller.login;

import entities.Employee;
import model.login.LoginModel;
import controller.pos.POSController;

import javax.swing.*;

public class LoginController {  3 usages  Giorgos-Karachristos
    private final LoginModel model;  2 usages
    private final POSController posController;  2 usages

    public LoginController(POSController posController) {  1 usage  Giorgos-Karachristos
        this.posController = posController;
        this.model = new LoginModel();
    }

    public void handleLogin(String username, String password) {  1 usage  Giorgos-Karachristos
        String result = model.signIn(username, password);

        if ("Invalid username or password".equals(result) || "An error occurred".equals(result)) {
            JOptionPane.showMessageDialog( parentComponent: null, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else if (result != null) {
            String[] words = result.split( regex: " ");
            if (words.length >= 3) {
                Employee user = new Employee(words[0], words[1], words[2]);
                posController.onLoginSuccess(user);
            }
        }
    }
}
```

A.3.7 Manager Controllers

CustomTabController.java (1/4)

CustomTabController.java

```java
package controller.manager;

import controller.category.CategoryController;
import controller.employee.EmployeeController;
import controller.inventory.InventoryController;
import controller.item.ItemController;
import controller.role.RoleController;
import controller.transaction.TransactionController;
import view.category.DeleteCategoryView;
import view.category.InsetCategoryView;
import view.category.UpdateCategoryView;
import view.employee.DeleteEmployeeView;
import view.employee.InsertEmployeeView;
import view.employee.UpdateEmployeeView;
import view.inventory.DeleteInventoryView;
import view.inventory.InsertInventoryView;
import view.inventory.UpdateInventoryView;
import view.item.DeleteItemView;
import view.item.InsertItemView;
import view.item.UpdateItemView;
import view.role.DeleteRoleView;
import view.role.InsertRoleView;
import view.role.UpdateRoleView;
import view.transaction.DeleteTransactionView;
import view.transaction.InsertTransactionView;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.Arrays;

public class CustomTabController {  3 usages  Giorgos-Karachristos
    private final ManagerController managerController;  7 usages
    private final EmployeeController employeeController;  2 usag
    private final RoleController roleController;  2 usages
    private final ItemController itemController;  2 usages
    private final CategoryController categoryController;  2 usag
    private final InventoryController inventoryController;  2 u
    private final TransactionController transactionController;
    private final String tabName;  6 usages

    public CustomTabController(String tabName) {  1 usage  Giorg
        managerController = new ManagerController();
        employeeController = new EmployeeController();
        roleController = new RoleController();
        itemController = new ItemController();
        categoryController = new CategoryController();
        inventoryController = new InventoryController();
        transactionController = new TransactionController();
        this.tabName = tabName;
    }
```

CustomTabController.java (2/4)

CustomTabController.java

```
public String[] getColumnNames() {  1 usage   Giorgos-Karachristos
    switch (tabName) {
        case "Employee":
            String[] strings = managerController.getAttributeNames( table: "SELECT_EMPLOYEE");
            strings = Arrays.copyOf(strings,  newLength: strings.length - 1);
            return strings;
        case "Role":
            return managerController.getAttributeNames( table: "SELECT_ROLE_COUNT");
        case "Item":
            return managerController.getAttributeNames( table: "SELECT_ITEMS");
        case "Category":
            return managerController.getAttributeNames( table: "SELECT_CATEGORY_COUNT");
        case "Inventory":
            return managerController.getAttributeNames( table: "SELECT_INVENTORY");
        case "Transactions":
            return managerController.getAttributeNames( table: "SELECT_TRANSACTION");
        default:
            return new String[]{};
    }
}

public void loadTableData(DefaultTableModel tableModel) {  2 usages   Giorgos-Karachristos
    switch (tabName) {
        case "Employee":
            employeeController.getEmployeeTable(tableModel);
            break;
        case "Role":
            roleController.getRoleTable(tableModel);
            break;
        case "Item":
            itemController.getItemTable(tableModel);
            break;
        case "Category":
            categoryController.getCategoryTable(tableModel);
            break;
        case "Inventory":
            inventoryController.getInventoryTable(tableModel);
            break;
        case "Transactions":
            transactionController.getTransactionTable(tableModel);
            break;
    }
}
```

CustomTabController.java (3/4)

CustomTabController.java

```
public void handleAddAction(DefaultTableModel tableModel) {  1 us
    JDialog addDialog;
    switch (tabName) {
        case "Employee":
            addDialog = new InsertEmployeeView(tableModel);
            break;
        case "Role":
            addDialog = new InsertRoleView(tableModel);
            break;
        case "Item":
            addDialog = new InsertItemView(tableModel);
            break;
        case "Category":
            addDialog = new InsetCategoryView(tableModel);
            break;
        case "Inventory":
            addDialog = new InsertInventoryView(tableModel);
            break;
        case "Transactions":
            addDialog = new InsertTransactionView(tableModel);
            break;
        default:
            addDialog = new JDialog();
            break;
    }
    addDialog.setModal(true);
    addDialog.setVisible(true);
}

public void handleUpdateAction(DefaultTableModel tableModel) {
    JDialog updateDialog;
    switch (tabName) {
        case "Employee":
            updateDialog = new UpdateEmployeeView(tableModel);
            break;
        case "Role":
            updateDialog = new UpdateRoleView(tableModel);
            break;
        case "Item":
            updateDialog = new UpdateItemView(tableModel);
            break;
        case "Category":
            updateDialog = new UpdateCategoryView(tableModel);
            break;
        case "Inventory":
            updateDialog = new UpdateInventoryView(tableModel);
            break;
        default:
            updateDialog = new JDialog();
            break;
    }
    updateDialog.setModal(true);
    updateDialog.setVisible(true);
}
```

CustomTabController.java (4/4)

```
CustomTabController.java
    public void handleDeleteAction(DefaultTableModel tableModel) {
        JDialog deleteDialog;
        switch (tabName) {
            case "Employee":
                deleteDialog = new DeleteEmployeeView(tableModel);
                break;
            case "Role":
                deleteDialog = new DeleteRoleView(tableModel);
                break;
            case "Item":
                deleteDialog = new DeleteItemView(tableModel);
                break;
            case "Category":
                deleteDialog = new DeleteCategoryView(tableModel);
                break;
            case "Inventory":
                deleteDialog = new DeleteInventoryView(tableModel);
                break;
            case "Transactions":
                deleteDialog = new DeleteTransactionView(tableModel);
                break;
            default:
                deleteDialog = new JDialog();
                break;
        }
        deleteDialog.setModal(true);
        deleteDialog.setVisible(true);
    }
}
```

ManagerController.java

```
ManagerController.java
package controller.manager;

import model.manager.ManagerDAO;
import model.manager.ManagerModel;

public class ManagerController {  2 usages  Giorgos-Karachristos
    private final ManagerDAO managerDAO;  2 usages

    public ManagerController() { this.managerDAO = new ManagerModel(); }

    public String[] getAttributeNames(String table) {  6 usages  Giorgos-Ka
        return managerDAO.getAttributeNames(table);
    }
}
```

A.3.8 Menu Controller

MenuController.java

```java
package controller.menu;

import entities.Category;
import entities.Item;
import model.menu.MenuDAO;
import model.menu.MenuModel;

import java.util.List;


public class MenuController {  3 usages  Giorgos-Karachristos
    private final MenuDAO model;  5 usages

    public MenuController() {  1 usage  Giorgos-Karachristos
        this.model = new MenuModel();
    }

    public List<Category> getCategoryList() {  1 usage  Giorgos-Karachristos
        return model.getCategoryList();
    }

    public List<Item> getItemsList(String category) {  1 usage  Giorgos-Karachristos
        return model.getItemsList(category);
    }

    public String insertOrder(String username, String subtotal, String type) {
        return model.insertOrder(username, subtotal, type);
    }

    public String insertKitchen(int id, double quantity, double total) {  1 usag
        return model.insertKitchen(id, quantity, total);
    }
}
```

A.3.9 Order Controller

OrderController.java

```java
package controller.order;

import entities.Order;
import model.order.OrderDAO;
import model.order.OrderModel;

import javax.swing.table.DefaultTableModel;
import java.util.List;

public class OrderController {
    private final OrderDAO model;

    public OrderController() { this.model = new OrderModel(); }

    public List<Order> getActiveOrdersList() {
        return model.getActiveOrdersList();
    }

    public void getKitchenList(DefaultTableModel table, String orderID) { model.getKitchenList(table, orderID); }

    public String voidKitchen(int orderID, String itemName) { return model.voidKitchen(orderID, itemName); }

    public String cancelOrder(int orderID) { return model.cancelOrder(orderID); }
}
```

A.3.10 POS Controller

POSController.java

```java
package controller.pos;

import entities.Employee;
import view.pos.POSView;

public class POSController {
    private final POSView posView;

    public POSController() {
        posView = new POSView( controller: this);
        posView.showLoginView();
    }

    // Called by LoginController when login succeeds
    public void onLoginSuccess(Employee user) {
        posView.showMainView(user);
    }

    public void onSignOut() {
        posView.showLoginView();
    }
}
```

A.3.11 Receipts Controller

ReceiptController.java

```
POSController.java
package controller.pos;

import entities.Employee;
import view.pos.POSView;

public class POSController {  12 usages  Giorgos-Karachris

    private final POSView posView;  4 usages

    public POSController() {  1 usage  Giorgos-Karachristc
        posView = new POSView( controller: this);
        posView.showLoginView();
    }

    // Called by LoginController when login succeeds
    public void onLoginSuccess(Employee user) {  1 usa
        posView.showMainView(user);
    }

    public void onSignOut() {  1 usage  Giorgos-Karachris
        posView.showLoginView();
    }
}
```

A.3.12 Register Controller

RegisterController.java

```
RegisterController.java
package controller.register;

import entities.ReceiptItem;
import model.register.RegisterDAO;
import model.register.RegisterModel;

import java.util.List;

public class RegisterController {  6 usages  Giorgos-Karachristos
    private final RegisterDAO registerDAO;  5 usages

    public RegisterController() { this.registerDAO = new RegisterModel(); }

    public String[] getOrder(int order_ID) { return registerDAO.getOrder(order_ID); }

    public List<ReceiptItem> getItemsList(String orderID) { return registerDAO.getItemsList(orderID); }

    public String completeOrder(int orderID, double subtotal, double discount, double tip, double total) {
        return registerDAO.completeOrder(orderID, subtotal, discount, tip, total);
    }

    public String cancelOrder(int orderID) { return registerDAO.cancelOrder(orderID); }
}
```

A.3.13 Reports Controller

ReportsController.java

```java
package controller.reports;

import model.reports.ReportsDAO;
import model.reports.ReportsModel;

import javax.swing.table.DefaultTableModel;

public class ReportsController {  6 usages  Giorgos-Karachristos
    private final ReportsDAO reportsDAO;  10 usages

    public ReportsController() {  2 usages  Giorgos-Karachristos
        this.reportsDAO = new ReportsModel();
    }

    public String[] getSalesLabels() {  1 usage  Giorgos-Karachristos
        return reportsDAO.getSalesLabels();
    }

    public String[] getRefundsLabel() { return reportsDAO.getRefundsLabel(); }

    public String[] getGrossSalesGraph() {  1 usage  Giorgos-Karachristos
        return reportsDAO.getGrossSalesGraph();
    }

    public String[] getRefundsGraph() { return reportsDAO.getRefundsGraph(); }

    public String[] getDiscountsGraph() { return reportsDAO.getDiscountsGraph(); }

    public String[] getNetSalesGraph() { return reportsDAO.getNetSalesGraph(); }

    public String[] getGrossProfitGraph() {  1 usage  Giorgos-Karachristos
        return reportsDAO.getGrossProfitGraph();
    }

    public String[] getInventoryGraph() { return reportsDAO.getInventoryGraph(); }

    public String[] getItemsGraph(DefaultTableModel tableModel) { return reportsDAO.getItemsGraph(tableModel); }
}
```

A.3.14 Restaurant Controller

RestaurantController.java

```java
package controller.restaurant;

import model.restaurant.RestaurantDAO;
import model.restaurant.RestaurantModel;

public class RestaurantController {  12 usages  Giorgos-Karachristos
    private final RestaurantDAO restaurantDAO;  4 usages

    public RestaurantController() { this.restaurantDAO = new RestaurantModel(); }

    public String[] getRestaurant() { return restaurantDAO.getRestaurant(); }

    public String updateRestaurant(String name, String address, String phone, int tables) {
        return restaurantDAO.updateRestaurant(name, address, phone, tables);
    }

    public String getRestaurantName() { return restaurantDAO.getRestaurantName(); }
}
```

A.3.15 Role Controller

RoleController.java

```java
package controller.role;

import entities.Role;
import model.role.RoleDAO;
import model.role.RoleModel;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public class RoleController {  18 usages  Giorgos-Karachristos

    private final RoleDAO roleDAO;  8 usages

    public RoleController() {  6 usages  Giorgos-Karachristos
        this.roleDAO = new RoleModel();
    }

    public void getRoleTable(DefaultTableModel tableModel) {  4 usages  Gio
        roleDAO.getRoleTable(tableModel);
    }

    public List<Role> getRoleList(DefaultComboBoxModel<String> boxModel) {
        return roleDAO.getRoleList(boxModel);
    }

    public void getRoleName(DefaultComboBoxModel<String> boxModel) {  2 usag
        roleDAO.getRoleName(boxModel);
    }

    public void getPOSAccess(DefaultComboBoxModel<String> boxModel) {  2 usa
        roleDAO.getPOSAccess(boxModel);
    }

    public String insertRole(String roleName, String access) {  1 usage  Gi
        return roleDAO.insertRole(roleName,access);
    }

    public String updateRole(String roleName, String access) {  1 usage  Gi
        return roleDAO.updateRole(roleName,access);
    }

    public String deleteRole(String roleName) {  1 usage  Giorgos-Karachristos
        return roleDAO.deleteRole(roleName);
    }
}
```

A.3.16 Transaction Controller

TransactionController.java

TransactionController.java

```java
package controller.transaction;

import entities.TransactionItem;
import model.transaction.TransactionDAO;
import model.transaction.TransactionModel;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public class TransactionController {  9 usages  Giorgos-Karachristos
    private final TransactionDAO transactionDAO;  5 usages

    public TransactionController() { this.transactionDAO = new TransactionModel(); }

    public void getTransactionTable(DefaultTableModel tableModel) { transactionDAO.getTransactionTable(tableModel); }

    public List<TransactionItem> getTransactionID(DefaultComboBoxModel<String> boxModel) {  1 usage  Giorgos-Karachristos
        return transactionDAO.getTransactionID(boxModel);
    }

    public String insertTransaction(int p_item_id, int p_old_quantity, String p_type, int p_transaction_quantity, int p_new_quantity) {
        return transactionDAO.insertTransaction(p_item_id, p_old_quantity, p_type, p_transaction_quantity, p_new_quantity);
    }

    public String deleteInventory(int id) { return transactionDAO.deleteInventory(id); }
}
```

A.3.17 Waiter Controller

WaiterController.java

```java
package controller.waiter;

import com.google.zxing.BarcodeFormat;
import com.google.zxing.EncodeHintType;
import com.google.zxing.WriterException;
import com.google.zxing.common.BitMatrix;
import com.google.zxing.qrcode.QRCodeWriter;
import model.waiter.WaiterDAO;
import model.waiter.WaiterModel;

import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class WaiterController {  3 usages  Giorgos-Karachristos
    private final WaiterDAO waiterDAO;  2 usages

    public WaiterController() {  1 usage  Giorgos-Karachristos
        this.waiterDAO = new WaiterModel();
    }

    public String fetchWaiterToken(String name, String surname) {  1 usage  Giorgos-Karachristos
        return waiterDAO.getWaiterToken(name, surname);
    }

    public BufferedImage generateQRCodeImage(String text, int width, int height) throws WriterException, IOException {

        Map<EncodeHintType, Object> hints = new HashMap<>();
        hints.put(EncodeHintType.CHARACTER_SET, "UTF-8");

        QRCodeWriter qrCodeWriter = new QRCodeWriter();
        BitMatrix bitMatrix = qrCodeWriter.encode(text, BarcodeFormat.QR_CODE, width, height, hints);

        BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
        for (int x = 0; x < width; x++) {
            for (int y = 0; y < height; y++) {
                image.setRGB(x, y, bitMatrix.get(x, y) ? Color.BLACK.getRGB() : Color.WHITE.getRGB());
            }
        }
        return image;
    }
}
```

A.4 Entities

Category.java

```java
package entities;

public class Category {  Giorgos-Karachristos
    private String name;  3 usages
    private String tax;  3 usages
    private String color;  3 usages
    private String isActive;  2 usages
    private String countItems;  2 usages

    public Category(String name, String color, String tax) {  Giorgos-Karachristos
        this.name = name;
        this.color = color;
        this.tax = tax;
    }

    public Category(String name, String tax, String color, String isActive, String countItems) {
        this.name = name;
        this.tax = tax;
        this.color = color;
        this.isActive = isActive;
        this.countItems = countItems;
    }

    public String getName() { return name; }

    public String getTax() { return tax; }

    public String getColor() { return color; }

    public String getIsActive() { return isActive; }

    public String getCountItems() { return countItems; }
}
```

Employee.java

```java
package entities;

public class Employee {  Giorgos-Karachristos *
    private String employeeID;  2 usages
    private String name;  3 usages
    private String surname;  3 usages
    private String phone;  2 usages
    private String email;  2 usages
    private String role;  3 usages
    private String ssn;  2 usages
    private String id;  2 usages
    private String hireDate;  2 usages
    private String username;  2 usages
    private String password;  1 usage

    public Employee(String employeeID, String name, String surname, String phone, String email,
                    String role, String ssn, String id, String hireDate, String username) {
        this.employeeID = employeeID;
        this.name = name;
        this.surname = surname;
        this.phone = phone;
        this.email = email;
        this.role = role;
        this.ssn = ssn;
        this.id = id;
        this.hireDate = hireDate;
        this.username = username;
    }

    public Employee(String name, String surname, String role) {  Giorgos-Karachristos
        this.name = name;
        this.surname = surname;
        this.role = role;
    }

    public String getEmployeeID() {  2 usages  Giorgos-Karachristos
        return employeeID;
    }

    public String getName() {  Giorgos-Karachristos
        return name;
    }

    public String getSurname() {  Giorgos-Karachristos
        return surname;
    }

    public String getPhone() {  Giorgos-Karachristos
        return phone;
    }

    public String getEmail() {  2 usages  Giorgos-Karachristos
        return email;
    }

    public String getRole() {  Giorgos-Karachristos
        return role;
    }

    public String getSsn() {  2 usages  Giorgos-Karachristos
        return ssn;
    }

    public String getId() {  Giorgos-Karachristos
        return id;
    }

    public String getHireDate() {  1 usage  Giorgos-Karachristos
        return hireDate;
    }

    public String getUsername() {  Giorgos-Karachristos
        return username;
    }

    public String getPassword() {  Giorgos-Karachristos
        return password;
    }
}
```

InventoryItem.java

```java
package entities;

public class InventoryItem {  17 usages  Giorgos-Karachristos *
    private String itemId;  2 usages
    private String itemCode;  2 usages
    private String name;  2 usages
    private String location;  2 usages
    private String quantity;  2 usages
    private String minQuantity;  2 usages
    private String maxQuantity;  2 usages
    private String lastUpdated;  2 usages

    public InventoryItem(String itemId, String itemCode, String name, String location, String quantity,
                         String minQuantity, String maxQuantity, String lastUpdated) {
        this.itemId = itemId;
        this.itemCode = itemCode;
        this.name = name;
        this.location = location;
        this.quantity = quantity;
        this.minQuantity = minQuantity;
        this.maxQuantity = maxQuantity;
        this.lastUpdated = lastUpdated;
    }

    public String getItemId() { return itemId; }

    public String getItemCode() { return itemCode; }

    public String getName() { return name; }

    public String getLocation() { return location; }

    public String getQuantity() { return quantity; }

    public String getMinQuantity() { return minQuantity; }

    public String getMaxQuantity() { return maxQuantity; }

    public String getLastUpdated() { return lastUpdated; }
}
```

Item.java

```java
package entities;

public class Item {  Giorgos-Karachristos
    private String itemId;  4 usages
    private String itemCode;  3 usages
    private String name;  4 usages
    private String description;  3 usages
    private String price;  3 usages
    private String cost;  2 usages
    private String category;  2 usages
    private String createdAt;  2 usages
    private String color;  3 usages
    private String isActive;  2 usages

    public Item(String name, String color, String price, String itemId, String description) {
        this.name = name;
        this.color = color;
        this.price = price;
        this.itemId = itemId;
        this.description = description;
    }

    public Item(String itemId, String itemCode, String name, String description, String price,
                String cost, String category, String createdAt, String color, String isActive) {
        this.itemId = itemId;
        this.itemCode = itemCode;
        this.name = name;
        this.description = description;
        this.price = price;
        this.cost = cost;
        this.category = category;
        this.createdAt = createdAt;
        this.color = color;
        this.isActive = isActive;
    }

    public Item(String itemId, String itemCode, String name) {  Giorgos-Karachristos
        this.itemId = itemId;
        this.itemCode = itemCode;
        this.name = name;
    }

    public String getItemId() {  Giorgos-Karachristos
        return itemId;
    }

    public String getItemCode() { return itemCode; }

    public String getName() { return name; }

    public String getDescription() { return description; }

    public String getPrice() { return price; }

    public String getCost() { return cost; }

    public String getCategory() { return category; }

    public String getCreatedAt() { return createdAt; }

    public String getColor() { return color; }

    public String getIsActive() { return isActive; }
}
```

KitchenItem.java

KitchenItem.java

```java
package entities;

public class KitchenItem {  11 usages  Giorgos-Karachristos
    private int itemID;  2 usages
    private int quantity;  2 usages
    private String status;  2 usages
    private String name;  2 usages

    public KitchenItem(String name, int quantity, int itemID, String status) {
        this.name = name;
        this.quantity = quantity;
        this.itemID = itemID;
        this.status = status;
    }

    public int getItemID() {  1 usage  Giorgos-Karachristos
        return itemID;
    }

    public int getQuantity() {  Giorgos-Karachristos
        return quantity;
    }

    public String getStatus() {  Giorgos-Karachristos
        return status;
    }

    public String getName() {  Giorgos-Karachristos
        return name;
    }
}
```

Order.java

```java
Order.java
package entities;

public class Order {  Giorgos-Karachristos *
    private String orderID;  4 usages
    private String name;  2 usages
    private String subtotal;  3 usages
    private String discount;  2 usages
    private String tip;  2 usages
    private String total;  2 usages
    private String status;  4 usages
    private String orderDate;  4 usages
    private String type;  2 usages

    public Order(String orderID, String status, String orderDate) {  Giorgos-Karachristos
        this.orderID = orderID;
        this.status = status;
        this.orderDate = orderDate;
    }

    public Order(String orderID, String subtotal, String status, String type, String orderDate) {
        this.orderID = orderID;
        this.subtotal = subtotal;
        this.status = status;
        this.type = type;
        this.orderDate = orderDate;
    }

    public Order(String orderID, String name, String subtotal, String discount, String tip,  G
                 String total, String orderDate, String status) {
        this.orderID = orderID;
        this.name = name;
        this.subtotal = subtotal;
        this.discount = discount;
        this.tip = tip;
        this.total = total;
        this.orderDate = orderDate;
        this.status = status;
    }

    public String getOrderID() { return orderID; }

    public String getName() { return name; }

    public String getSubtotal() { return subtotal; }

    public String getDiscount() { return discount; }

    public String getTip() { return tip; }

    public String getTotal() { return total; }

    public String getStatus() { return status; }

    public String getOrderDate() { return orderDate; }

    public String getType() { return type; }
}
```

ReceiptItem.java

```java
package entities;

public class ReceiptItem {  22 usages  Giorgos-Karachristos
    private String name;  2 usages
    private int quantity;  2 usages
    private String tax;  2 usages
    private String totalPrice;  2 usages

    public ReceiptItem(String name, int quantity, String tax, String totalPrice) {
        this.name = name;
        this.quantity = quantity;
        this.tax = tax;
        this.totalPrice = totalPrice;
    }

    public String getName() { return name; }

    public int getQuantity() { return quantity; }

    public String getTax() { return tax; }

    public String getTotalPrice() { return totalPrice; }
}
```

Role.java

```java
package entities;

public class Role {  Giorgos-Karachristos
    private String name;  2 usages
    private String posAccess;  2 usages
    private String countEmployees;  2 usages

    public Role(String name, String posAccess, String countEmployees) {
        this.name = name;
        this.posAccess = posAccess;
        this.countEmployees = countEmployees;
    }

    public String getPosAccess() {  2 usages  Giorgos-Karachristos
        return posAccess;
    }

    public String getCountEmployees() {  2 usages  Giorgos-Karachristos
        return countEmployees;
    }

    public String getName() {  Giorgos-Karachristos
        return name;
    }
}
```

TransactionItem.java

```java
package entities;

public class TransactionItem {  11 usages  Giorgos-Karachristos *
    private String transactionId;  2 usages
    private String itemId;  2 usages
    private String itemCode;  2 usages
    private String name;  2 usages
    private String oldQuantity;  2 usages
    private String type;  2 usages
    private String transactionQuantity;  2 usages
    private String transactionDate;  2 usages
    private String status;  2 usages

    public TransactionItem(String transactionId, String itemId, String itemCode,  1 usage  Giorgos-Kar
                           String name, String oldQuantity, String type, String transactionQuantity,
                           String transactionDate, String status) {
        this.transactionId = transactionId;
        this.itemId = itemId;
        this.itemCode = itemCode;
        this.name = name;
        this.oldQuantity = oldQuantity;
        this.type = type;
        this.transactionQuantity = transactionQuantity;
        this.transactionDate = transactionDate;
        this.status = status;
    }

    public String getTransactionId() { return transactionId; }

    public String getItemId() { return itemId; }

    public String getItemCode() { return itemCode; }

    public String getName() { return name; }

    public String getOldQuantity() { return oldQuantity; }

    public String getType() { return type; }

    public String getTransactionQuantity() { return transactionQuantity; }

    public String getTransactionDate() { return transactionDate; }

    public String getStatus() { return status; }
}
```

A.5 Model

A.5.1 Category

CategoryDAO.java

```
CategoryDAO.java
package model.category;

import entities.Category;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public interface CategoryDAO {  3 usages  1 implementation  Giorgos-Karachristos
    void getCategoryTable(DefaultTableModel tableModel);  1 usage  1 implementation  Gio

    List<Category> getCategoryList(DefaultComboBoxModel<String> boxModel);  1 usage

    void getCategoryName(DefaultComboBoxModel<String> boxModel);  1 usage  1 implementat

    String insertCategory(String name, double tax, String color, String isActive);

    String updateCategory(String name, double tax, String color, String isActive);

    String deleteCategory(String name);  1 usage  1 implementation  Giorgos-Karachristos
}
```

CategoryModel.java (1/3)

```
CategoryModel.java
package model.category;

import entities.Category;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class CategoryModel implements CategoryDAO {  3 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(CategoryModel.class.getName());
    private static final String ENDPOINT = "manager/category";  4 usages
```

CategoryModel.java (2/3)

```java
CategoryModel.java
@Override 1 usage  Giorgos-Karachristos
public void getCategoryTable(DefaultTableModel tableModel) {
    tableModel.setRowCount(0);
    JSONArray categories = fetchCategoryData();
    if (categories != null) {
        for (int i = 0; i < categories.length(); i++) {
            try {
                JSONObject category = categories.getJSONObject(i);
                Object[] rowData = new Object[5];
                rowData[0] = category.getString( key: "NAME");
                rowData[1] = category.getString( key: "TAX");
                rowData[2] = category.getString( key: "COLOR");
                rowData[3] = category.getString( key: "IS_ACTIVE");
                rowData[4] = category.getString( key: "COUNT_ITEMS");
                tableModel.addRow(rowData);
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing category data for table", e);
            }
        }
    }
}

//DeleteCategory
//UpdateCategory
@Override 1 usage  Giorgos-Karachristos *
public List<Category> getCategoryList(DefaultComboBoxModel<String> boxModel) {
    boxModel.removeAllElements();
    List<Category> categoryArrayList = new ArrayList<>();
    boxModel.addElement( anObject: "");
    JSONArray categories = fetchCategoryData();
    if (categories != null) {
        for (int i = 0; i < categories.length(); i++) {
            try {
                JSONObject category = categories.getJSONObject(i);
                boxModel.addElement(category.getString( key: "NAME"));
                categoryArrayList.add(new Category(category.getString( key: "NAME"), category.getString( key: "TAX"),
                        category.getString( key: "COLOR"), category.getString( key: "IS_ACTIVE"),
                        category.getString( key: "COUNT_ITEMS")));
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing category data for list", e);
            }
        }
    }
    return categoryArrayList;
}

//InsertItem
//UpdateItemView
@Override 1 usage  Giorgos-Karachristos
public void getCategoryName(DefaultComboBoxModel<String> boxModel) {
    boxModel.removeAllElements();
    boxModel.addElement( anObject: "");
    JSONArray categories = fetchCategoryData();
    if (categories != null) {
        for (int i = 0; i < categories.length(); i++) {
            try {
                JSONObject category = categories.getJSONObject(i);
                boxModel.addElement(category.getString( key: "NAME"));
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing category data for ComboBox", e);
            }
        }
    }
}
```

CategoryModel.java (3/3)

```
CategoryModel.java
    private JSONArray fetchCategoryData() {  3 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchRoleData", e);
        }
        return null;
    }


    @Override  1 usage  Giorgos-Karachristos
    public String insertCategory(String name, double tax, String color, String isActive) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("NAME", name);
        jsonInput.put("TAX", tax);
        jsonInput.put("COLOR", color);
        jsonInput.put("IS_ACTIVE", isActive);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "POST", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String updateCategory(String name, double tax, String color, String isActive) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("NAME", name);
        jsonInput.put("TAX", tax);
        jsonInput.put("COLOR", color);
        jsonInput.put("IS_ACTIVE", isActive);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "PUT", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String deleteCategory(String name) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("NAME", name);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "DELETE", jsonInput);
    }
}
```

A.5.2 Employee

EmployeeDAO.java

```
EmployeeDAO.java
package model.employee;

import entities.Employee;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public interface EmployeeDAO {  3 usages  1 implementation  Giorgos-Karachristos *
     void getEmployeeTable(DefaultTableModel tableModel);  1 usage  1 implementation  Giorgos-Karachristos

     List<Employee> getEmployeeList(DefaultComboBoxModel<String> boxModel);  1 usage  1 implementation

     String insertEmployee(String name, String surname, String phone, String email,  1 usage  1 implen
                           String role, String ssn, String idn, String username, String password);

     String updateEmployee(int id, String name, String surname, String phone, String email,  1 usag
                           String role, String ssn, String idn, String username, String password);

     String deleteEmployee(int id);  1 usage  1 implementation  Giorgos-Karachristos
}
```

EmployeeModel.java (1/3)

```
EmployeeModel.java
package model.employee;

import entities.Employee;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class EmployeeModel implements EmployeeDAO {  3 usages  Giorgos-Karachristos

    private static final Logger LOGGER = Logger.getLogger(EmployeeModel.class.getName());
    private static final String ENDPOINT = "manager/employee";  4 usages
```

EmployeeModel.java (2/3)

```java
EmployeeModel.java
@Override  1 usage  Giorgos-Karachristos
public void getEmployeeTable(DefaultTableModel tableModel) {
    tableModel.setRowCount(0);
    JSONArray employees = fetchEmployeeData();
    if (employees != null) {
        for (int i = 0; i < employees.length(); i++) {
            try {
                JSONObject employee = employees.getJSONObject(i);
                Object[] rowData = new Object[11];
                rowData[0] = employee.getString( key: "EMPLOYEE_ID");
                rowData[1] = employee.getString( key: "NAME");
                rowData[2] = employee.getString( key: "SURNAME");
                rowData[3] = employee.getString( key: "PHONE");
                rowData[4] = employee.getString( key: "EMAIL");
                rowData[5] = employee.isNull( key: "ROLE") ? "" : employee.getString( key: "ROLE");
                rowData[6] = employee.getString( key: "SOCIAL_SECURITY_NUMBER");
                rowData[7] = employee.getString( key: "IDENTIFICATION_NUMBER");
                rowData[8] = employee.getString( key: "HIRE_DATE").substring(0, 10);
                rowData[9] = employee.getString( key: "USERNAME");
                tableModel.addRow(rowData);
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing employee data for table", e);
            }
        }
    }
}

//DeleteEmployee
//UpdateEmployee
@Override  1 usage  Giorgos-Karachristos *
public List<Employee> getEmployeeList(DefaultComboBoxModel<String> boxModel) {
    boxModel.removeAllElements();
    List<Employee> employeeList = new ArrayList<>();
    boxModel.addElement( anObject: "");
    JSONArray employees = fetchEmployeeData();
    if (employees != null) {
        for (int i = 0; i < employees.length(); i++) {
            try {
                JSONObject employee = employees.getJSONObject(i);
                boxModel.addElement(employee.getString( key: "EMPLOYEE_ID"));
                employeeList.add(new Employee(employee.getString( key: "EMPLOYEE_ID"),
                        employee.getString( key: "NAME"), employee.getString( key: "SURNAME"),
                        employee.getString( key: "PHONE"), employee.getString( key: "EMAIL"),
                        employee.isNull( key: "ROLE") ? "" : employee.getString( key: "ROLE"),
                        employee.getString( key: "SOCIAL_SECURITY_NUMBER"),
                        employee.getString( key: "IDENTIFICATION_NUMBER"),
                        employee.getString( key: "HIRE_DATE").substring(0, 10),
                        employee.getString( key: "USERNAME")));
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing employee data for list", e);
            }
        }
    }
    return employeeList;
}
```

EmployeeModel.java (3/3)

```java
EmployeeModel.java
    private JSONArray fetchEmployeeData() {  2 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchEmployeeData", e);
        }
        return null;
    }

    @Override  1 usage  Giorgos-Karachristos *
    public String insertEmployee(String name, String surname, String phone, String email,
                                 String role, String ssn, String idn, String username, String password) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("NAME", name);
        jsonInput.put("SURNAME", surname);
        jsonInput.put("PHONE", phone);
        jsonInput.put("EMAIL", email);
        jsonInput.put("ROLE", role);
        jsonInput.put("SOCIAL_SECURITY_NUMBER", ssn);
        jsonInput.put("IDENTIFICATION_NUMBER", idn);
        jsonInput.put("USERNAME", username);
        jsonInput.put("PASSWORD", password);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "POST", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos *
    public String updateEmployee(int id, String name, String surname, String phone, String email,
                                 String role, String ssn, String idn, String username, String password) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("EMPLOYEE_ID", id);
        jsonInput.put("NAME", name);
        jsonInput.put("SURNAME", surname);
        jsonInput.put("PHONE", phone);
        jsonInput.put("EMAIL", email);
        jsonInput.put("ROLE", role);
        jsonInput.put("SOCIAL_SECURITY_NUMBER", ssn);
        jsonInput.put("IDENTIFICATION_NUMBER", idn);
        jsonInput.put("USERNAME", username);
        jsonInput.put("PASSWORD", password);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "PUT", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String deleteEmployee(int id) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("EMPLOYEE_ID", id);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "DELETE", jsonInput);
    }
}
```

A.5.3 Inventory

InventoryDAO.java

```
InventoryDAO.java
package model.inventory;

import entities.InventoryItem;
import entities.Item;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public interface InventoryDAO {  3 usages  1 implementation  Giorgos-Karachristos
    void getInventoryTable(DefaultTableModel tableModel);  1 usage  1 implementation  Giorgos-Karachristos

    List<InventoryItem> getInventoryList(DefaultComboBoxModel<String> boxModel);  1 usage  1 implementation

    List<Item> getInventoryComboBox(DefaultComboBoxModel<String> boxModel);  1 usage  1 implementation  G

    String insertInventory(int id, String location, int quantity, int minQuantity, int maxQuantity);

    String updateInventory(int id, String location, int quantity, int minQuantity, int maxQuantity);

    String deleteInventory(int id);  1 usage  1 implementation  Giorgos-Karachristos
}
```

InventoryModel.java (1/3)

```
InventoryModel.java
package model.inventory;

import entities.InventoryItem;
import entities.Item;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class InventoryModel implements InventoryDAO {  3 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(InventoryModel.class.getName());
    private static final String ENDPOINT = "manager/inventory";  4 usages
```

InventoryModel.java (2/3)

```
InventoryModel.java
@Override  1 usage  Giorgos-Karachristos
public void getInventoryTable(DefaultTableModel tableModel) {
    tableModel.setRowCount(0);
    JSONArray inventories = fetchInventoryData( endpoint: "");
    if (inventories != null) {
        for (int i = 0; i < inventories.length(); i++) {
            try {
                JSONObject inventory = inventories.getJSONObject(i);
                Object[] rowData = new Object[8];
                rowData[0] = inventory.getString( key: "ITEM_ID");
                rowData[1] = inventory.getString( key: "ITEM_CODE");
                rowData[2] = inventory.getString( key: "NAME");
                rowData[3] = inventory.getString( key: "LOCATION");
                rowData[4] = inventory.getString( key: "QUANTITY");
                rowData[5] = inventory.getString( key: "MIN_QUANTITY");
                rowData[6] = inventory.getString( key: "MAX_QUANTITY");
                rowData[7] = inventory.getString( key: "LAST_UPDATED");
                tableModel.addRow(rowData);
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing inventory data for table", e);
            }
        }
    }
}

//UpdateInventory
//DeleteInventory
//InsertTransaction
@Override  1 usage  Giorgos-Karachristos *
public List<InventoryItem> getInventoryList(DefaultComboBoxModel<String> boxModel) {
    boxModel.removeAllElements();
    List<InventoryItem> rowData = new ArrayList<>();
    boxModel.addElement( anObject: "");
    JSONArray inventories = fetchInventoryData( endpoint: "");
    if (inventories != null) {
        for (int i = 0; i < inventories.length(); i++) {
            try {
                JSONObject inventory = inventories.getJSONObject(i);
                boxModel.addElement(inventory.getString( key: "ITEM_ID"));
                rowData.add(new InventoryItem(inventory.getString( key: "ITEM_ID"),
                        inventory.getString( key: "ITEM_CODE"), inventory.getString( key: "NAME"),
                        inventory.getString( key: "LOCATION"), inventory.getString( key: "QUANTITY"),
                        inventory.getString( key: "MIN_QUANTITY"), inventory.getString( key: "MAX_QUANTITY"),
                        inventory.getString( key: "LAST_UPDATED").substring(0, 10)));
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing inventory data for list", e);
            }
        }
    }
    return rowData;
}

//InsertInventory
@Override  1 usage  Giorgos-Karachristos *
public List<Item> getInventoryComboBox(DefaultComboBoxModel<String> boxModel) {
    boxModel.removeAllElements();
    List<Item> rowData = new ArrayList<>();
    boxModel.addElement( anObject: "");
    JSONArray inventories = fetchInventoryData( endpoint: "/getInventoryComboBox");
    if (inventories != null) {
        for (int i = 0; i < inventories.length(); i++) {
            try {
                JSONObject inventory = inventories.getJSONObject(i);
                boxModel.addElement(inventory.getString( key: "ITEM_ID"));
                rowData.add(new Item(inventory.getString( key: "ITEM_ID"),
                        inventory.getString( key: "ITEM_CODE"), inventory.getString( key: "NAME")));
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing inventory data for ComboBox", e);
            }
        }
    }
    return rowData;
}
```

InventoryModel.java (3/3)

```java
InventoryModel.java
    private JSONArray fetchInventoryData(String endpoint) {  3 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + endpoint,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchInventoryData", e);
        }
        return null;
    }

    @Override  1 usage  Giorgos-Karachristos
    public String insertInventory(int id, String location, int quantity, int minQuantity, int maxQuantity) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ITEM_ID", id);
        jsonInput.put("LOCATION", location);
        jsonInput.put("QUANTITY", quantity);
        jsonInput.put("MIN_QUANTITY", minQuantity);
        jsonInput.put("MAX_QUANTITY", maxQuantity);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "POST", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String updateInventory(int id, String location, int quantity, int minQuantity, int maxQuantity) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ITEM_ID", id);
        jsonInput.put("LOCATION", location);
        jsonInput.put("QUANTITY", quantity);
        jsonInput.put("MIN_QUANTITY", minQuantity);
        jsonInput.put("MAX_QUANTITY", maxQuantity);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "PUT", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String deleteInventory(int id) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ITEM_ID", id);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "DELETE", jsonInput);
    }
}
```

A.5.4 Item

ItemDAO.java

```
ItemDAO.java
package model.item;

import entities.Item;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public interface ItemDAO {  3 usages  1 implementation  Giorgos-Karachristos *
    void getItemTable(DefaultTableModel tableModel);  1 usage  1 implementation  Giorgos-Karachristos

    List<Item> getItemList(DefaultComboBoxModel<String> boxModel);  1 usage  1 implementation  Giorgos-Karachris

    String insertItem(String code, String name, String description, double price, double cost,  1 usage
                      String category, String color, String isActive);

    String updateItem(int id, String code, String name, String description, double price, double cost,
                      String category, String color, String isActive);

    String deleteItem(int id);  1 usage  1 implementation  Giorgos-Karachristos
}
```

ItemModel.java (1/3)

```
ItemModel.java
package model.item;

import entities.Item;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ItemModel implements ItemDAO {  3 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(ItemModel.class.getName());
    private static final String ENDPOINT = "manager/item";  4 usages
```

ItemModel.java (2/3)

```java
@Override  1 usage  Giorgos-Karachristos
public void getItemTable(DefaultTableModel tableModel) {
    tableModel.setRowCount(0);
    JSONArray items = fetchItemData();
    if (items != null) {
        for (int i = 0; i < items.length(); i++) {
            try {
                JSONObject item = items.getJSONObject(i);
                Object[] rowData = new Object[10];
                rowData[0] = item.getString( key: "ITEM_ID");
                rowData[1] = item.getString( key: "ITEM_CODE");
                rowData[2] = item.getString( key: "NAME");
                rowData[3] = item.isNull( key: "DESCRIPTION") ? "" : item.getString( key: "DESCRIPTION");
                rowData[4] = item.getString( key: "PRICE");
                rowData[5] = item.getString( key: "COST");
                rowData[6] = item.isNull( key: "CATEGORY") ? "" : item.getString( key: "CATEGORY");
                rowData[7] = item.getString( key: "CREATED_AT").substring(0, 10);
                rowData[8] = item.getString( key: "COLOR");
                rowData[9] = item.getString( key: "IS_ACTIVE");
                tableModel.addRow(rowData);
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing item data for table", e);
            }
        }
    }
}

//DeleteItem
//UpdateItemView
@Override  1 usage  Giorgos-Karachristos *
public List<Item> getItemList(DefaultComboBoxModel<String> boxModel) {
    boxModel.removeAllElements();
    List<Item> itemArrayList = new ArrayList<>();
    boxModel.addElement( anObject: "");
    JSONArray items = fetchItemData();
    if (items != null) {
        for (int i = 0; i < items.length(); i++) {
            try {
                JSONObject item = items.getJSONObject(i);
                boxModel.addElement(item.getString( key: "ITEM_ID"));
                itemArrayList.add(new Item(item.getString( key: "ITEM_ID"), item.getString( key: "ITEM_CODE"),
                        item.getString( key: "NAME"),
                        item.isNull( key: "DESCRIPTION") ? "" : item.getString( key: "DESCRIPTION"),
                        item.getString( key: "PRICE"), item.getString( key: "COST"),
                        item.isNull( key: "CATEGORY") ? "" : item.getString( key: "CATEGORY"),
                        item.getString( key: "CREATED_AT").substring(0, 10),
                        item.getString( key: "COLOR"), item.getString( key: "IS_ACTIVE")));
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing item data for list", e);
            }
        }
    }
    return itemArrayList;
}
```

ItemModel.java (3/3)

```java
ItemModel.java
    private JSONArray fetchItemData() {  2 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchRoleData", e);
        }
        return null;
    }

    @Override  1 usage  Giorgos-Karachristos *
    public String insertItem(String code, String name, String description, double price, double cost,
                             String category, String color, String isActive) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ITEM_CODE", code);
        jsonInput.put("NAME", name);
        jsonInput.put("DESCRIPTION", description);
        jsonInput.put("PRICE", price);
        jsonInput.put("COST", cost);
        jsonInput.put("CATEGORY", category);
        jsonInput.put("COLOR", color);
        jsonInput.put("IS_ACTIVE", isActive);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "POST", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos *
    public String updateItem(int id, String code, String name, String description, double price,
                             double cost, String category, String color, String isActive) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ITEM_ID", id);
        jsonInput.put("ITEM_CODE", code);
        jsonInput.put("NAME", name);
        jsonInput.put("DESCRIPTION", description);
        jsonInput.put("PRICE", price);
        jsonInput.put("COST", cost);
        jsonInput.put("CATEGORY", category);
        jsonInput.put("COLOR", color);
        jsonInput.put("IS_ACTIVE", isActive);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "PUT", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String deleteItem(int id) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ITEM_ID", id);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "DELETE", jsonInput);
    }
}
```

A.5.5 Kitchen

KitchenDAO.java

```java
package model.kitchen;

import entities.KitchenItem;
import entities.Order;

import java.util.List;

public interface KitchenDAO {  3 usages  1 implementation
    List<Order> getOrdersList();  1 usage  1 implementation

    List<KitchenItem> getCookList(String orderID);  1

    String markAsServed(String orderID);  1 usage  1 impl

    String updateStatus(String orderID, String item);
}
```

KitchenModel.java (1/2)

```java
package model.kitchen;

import entities.KitchenItem;
import entities.Order;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class KitchenModel implements KitchenDAO {  3 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(KitchenModel.class.getName());  3 usages
    private static final String ENDPOINT = "kitchen/";  3 usages

    @Override  1 usage  Giorgos-Karachristos *
    public List<Order> getOrdersList() {
        List<Order> ordersList = new ArrayList<>();
        JSONArray orders = fetchKitchenData( endpoint: "getOrdersList");
        if (orders != null) {
            for (int i = 0; i < orders.length(); i++) {
                try {
                    JSONObject orderJSON = orders.getJSONObject(i);
                    ordersList.add(new Order(orderJSON.getString( key: "ORDER_ID"),
                            orderJSON.getString( key: "STATUS"), orderJSON.getString( key: "ORDER_DATE")));
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing order data", e);
                }
            }
        }
        return ordersList;
    }
```

KitchenModel.java (2/2)

```java
    @Override  1 usage  Giorgos-Karachristos *
    public List<KitchenItem> getCookList(String orderID) {
        List<KitchenItem> cookList = new ArrayList<>();
        JSONArray items = fetchKitchenData( endpoint: "getCookList?orderID=" + orderID);
        if (items != null) {
            for (int i = 0; i < items.length(); i++) {
                try {
                    JSONObject item = items.getJSONObject(i);
                    cookList.add(new KitchenItem(item.getString( key: "NAME"), item.getInt( key: "QUANTITY"),
                            item.getInt( key: "ITEM_ID"), item.getString( key: "STATUS")));
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing item data", e);
                }
            }
        }
        return cookList;
    }

    private JSONArray fetchKitchenData(String endpoint) {  2 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + endpoint,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchKitchenData", e);
        }
        return null;
    }

    @Override  1 usage  Giorgos-Karachristos
    public String markAsServed(String orderID) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("orderID", orderID);
        return HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "markAsServed",  method: "POST", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String updateStatus(String orderID, String item) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("orderID", orderID);
        jsonInput.put("itemID", item);
        return HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "updateStatus",  method: "POST", jsonInput);
    }
}
```

A.5.6 Login

LoginDAO.java

```java
package model.login;

public interface LoginDAO {  2 usages  1 implementation
    String signIn(String username, String password);
}
```

LoginModel.java

```java
package model.login;

import org.json.JSONObject;
import util.HttpRequestUtil;

public class LoginModel implements LoginDAO{  3 usages  Giorgos-Karachristos

    public String signIn(String username, String password) {  1 usage  Giorgos-Karachristos
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("username", username);
        jsonInput.put("password", password);

        return HttpRequestUtil.sendHttpRequest( endpoint: "login",  method: "POST", jsonInput);
    }
}
```

A.5.7 Manager

ManagerDAO.java

```java
package model.login;

import org.json.JSONObject;
import util.HttpRequestUtil;

public class LoginModel implements LoginDAO{  3 usages  Giorgos-Karachristos

    public String signIn(String username, String password) {  1 usage  Giorgos-Karachristos
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("username", username);
        jsonInput.put("password", password);

        return HttpRequestUtil.sendHttpRequest( endpoint: "login",  method: "POST", jsonInput);
    }
}
```

ManagerModel.java

```java
package model.manager;

import org.json.JSONObject;
import util.HttpRequestUtil;

public class ManagerModel implements ManagerDAO{  2 usages  Giorgos-Karachristos

    public String[] getAttributeNames(String table) {  1 usage  Giorgos-Karachristos
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("table", table);

        String[] columnNames = HttpRequestUtil.sendHttpRequest( endpoint: "manager/atributes",  method: "POST", jsonInput).split( regex: " ");
        for (int i = 0; i < columnNames.length; i++) {
            String s = columnNames[i].replace( target: "_",  replacement: " ");
            String res = s.charAt(0) + s.substring( beginIndex: 1).toLowerCase();
            columnNames[i] = res;
        }
        return columnNames;
    }
}
```

A.5.8 Menu

MenuDAO.java

```
package model.menu;

import entities.Category;
import entities.Item;

import java.util.List;

public interface MenuDAO {  3 usages  1 implementation  Giorgos-Karachristos
    List<Category> getCategoryList();  1 usage  1 implementation  Giorgos-Kara

    List<Item> getItemsList(String category);  1 usage  1 implementation  Gi

    String insertOrder(String username, String subtotal, String type);

    String insertKitchen(int id, double quantity, double total);  1 usag
}
```

MenuModel.java (1/2)

```
package model.menu;

import entities.Category;
import entities.Item;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class MenuModel implements MenuDAO {  3 usages  Giorgos-Karachristos *
    private static final Logger LOGGER = Logger.getLogger(MenuModel.class.getName());  3 usages
    private static final String ENDPOINT = "cashier/menu/";  3 usages

    @Override  1 usage  Giorgos-Karachristos *
    public List<Category> getCategoryList() {
        List<Category> categoryArrayList = new ArrayList<>();
        JSONArray categories = fetchMenuData( endpoint: "getCategoryList");
        if (categories != null) {
            for (int i = 0; i < categories.length(); i++) {
                try {
                    JSONObject category = categories.getJSONObject(i);
                    categoryArrayList.add(new Category(category.getString( key: "NAME"),
                            category.getString( key: "COLOR"), category.getString( key: "TAX")));
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing category data for list", e);
                }
            }
        }
        return categoryArrayList;
    }
```

MenuModel.java (2/2)

```java
    @Override  1 usage  Giorgos-Karachristos *
    public List<Item> getItemsList(String category) {
        List<Item> itemArrayList = new ArrayList<>();
        String encodedCategory = URLEncoder.encode(category, StandardCharsets.UTF_8);
        JSONArray items = fetchMenuData( endpoint: "getItemsList?category=" + encodedCategory);
        if (items != null) {
            for (int i = 0; i < items.length(); i++) {
                try {
                    JSONObject item = items.getJSONObject(i);
                    itemArrayList.add(new Item(item.getString( key: "NAME"), item.getString( key: "COLOR"),
                            item.getString( key: "PRICE"), String.valueOf(item.getInt( key: "ITEM_ID")),
                            item.isNull( key: "DESCRIPTION") ? "" : item.getString( key: "DESCRIPTION")));
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing item data for list", e);
                }
            }
        }
        return itemArrayList;
    }

    private JSONArray fetchMenuData(String endpoint) {  2 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + endpoint,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchMenuData", e);
        }
        return null;
    }

    @Override  1 usage  Giorgos-Karachristos
    public String insertOrder(String username, String subtotal, String type) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("NAME", username);
        jsonInput.put("SUBTOTAL", subtotal);
        jsonInput.put("TYPE", type);

        return HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "insertOrder",  method: "POST", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String insertKitchen(int id, double quantity, double total) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ITEM_ID", id);
        jsonInput.put("QUANTITY", quantity);
        jsonInput.put("TOTAL_PRICE", total);

        return HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "insertKitchen",  method: "POST", jsonInput);
    }
}
```

A.5.9 Order

OrderDAO.java

```
OrderDAO.java
package model.order;

import entities.Order;

import javax.swing.table.DefaultTableModel;
import java.util.List;

public interface OrderDAO {  3 usages  1 implementation  Giorgos-Karachristo
    List<Order> getActiveOrdersList();  1 usage  1 implementation  Giorg

    void getKitchenList(DefaultTableModel table, String orderID);

    String voidKitchen(int orderID, String itemName);  1 usage  1 imp

    String cancelOrder(int orderID);  1 usage  1 implementation  Giorgos
}
```

OrderModel.java (1/2)

```
OrderModel.java
package model.order;

import entities.Order;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class OrderModel implements OrderDAO {  3 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(OrderModel.class.getName());  3 usag
    private static final String ENDPOINT = "cashier/order/";  3 usages

    @Override  1 usage  Giorgos-Karachristos *
    public List<Order> getActiveOrdersList() {
        List<Order> orderArrayList = new ArrayList<>();
        JSONArray orders = fetchOrderData( endpoint: "getActiveOrdersList");
        if (orders != null) {
            for (int i = 0; i < orders.length(); i++) {
                try {
                    JSONObject order = orders.getJSONObject(i);
                    orderArrayList.add(new Order(order.getString( key: "ORDER_ID"),
                            order.getString( key: "SUBTOTAL"), order.getString( key: "STATUS"),
                            order.isNull( key: "TYPE") ? "" : order.getString( key: "TYPE"),
                            order.getString( key: "ORDER_DATE")));
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing order data for list", e);
                }
            }
        }
        return orderArrayList;
    }
```

OrderModel.java (2/2)

```java
OrderModel.java
    @Override  1 usage  Giorgos-Karachristos
    public void getKitchenList(DefaultTableModel table, String orderID) {
        Object[] rowData = new Object[3];
        JSONArray items = fetchOrderData( endpoint: "getKitchenList?orderID=" + orderID);
        if (items != null) {
            for (int i = 0; i < items.length(); i++) {
                try {
                    JSONObject item = items.getJSONObject(i);
                    rowData[0] = item.getString( key: "NAME");
                    rowData[1] = item.getInt( key: "QUANTITY");
                    rowData[2] = item.getString( key: "STATUS");
                    table.addRow(rowData);
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing item data for table", e);
                }
            }
        }
    }

    private JSONArray fetchOrderData(String endpoint) {  2 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + endpoint,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchOrderData", e);
        }
        return null;
    }

    @Override  1 usage  Giorgos-Karachristos
    public String voidKitchen(int orderID, String itemName) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ORDER_ID", orderID);
        jsonInput.put("NAME", itemName);

        return HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "voidKitchen",  method: "POST", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String cancelOrder(int orderID) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ORDER_ID", orderID);

        return HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "cancelOrder",  method: "POST", jsonInput);
    }
}
```

A.5.10 Receipts

ReceiptDAO.java

```java
package model.receipts;

import entities.ReceiptItem;
import entities.Order;

import java.util.List;

public interface ReceiptDAO {  3 usages  1 implementation  Giorgc
    List<Order> getFinalizedOrdersList();  1 usage  1 implemer

    List<ReceiptItem> getActiveItemsList(String orderID);

    String refundOrder(int orderID);  1 usage  1 implementation
}
```

ReceiptModel.java (1/2)

```java
package model.receipts;

import entities.ReceiptItem;
import entities.Order;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ReceiptModel implements ReceiptDAO {  3 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(ReceiptModel.class.getName());  3 usa
    private static final String ENDPOINT = "cashier/receipt/";  2 usages

    @Override  1 usage  Giorgos-Karachristos *
    public List<Order> getFinalizedOrdersList() {
        List<Order> orderArrayList = new ArrayList<>();
        JSONArray orders = fetchReceiptData( endpoint: "getFinalizedOrdersList");
        if (orders != null) {
            for (int i = 0; i < orders.length(); i++) {
                try {
                    JSONObject order = orders.getJSONObject(i);
                    orderArrayList.add(new Order(order.getString( key: "ORDER_ID"),
                            order.getString( key: "NAME"), order.getString( key: "SUBTOTAL"),
                            order.getString( key: "DISCOUNT"), order.getString( key: "TIP"),
                            order.getString( key: "TOTAL"), order.getString( key: "ORDER_DATE"),
                            order.getString( key: "STATUS")));
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing order data for list", e);
                }
            }
        }
        return orderArrayList;
    }
```

ReceiptModel.java (2/2)

```java
ReceiptModel.java
    @Override 1 usage  Giorgos-Karachristos *
    public List<ReceiptItem> getActiveItemsList(String orderID) {
        List<ReceiptItem> receiptItemList = new ArrayList<>();
        JSONArray items = fetchReceiptData( endpoint: "getActiveItemsList?orderID=" + orderID);
        if (items != null) {
            for (int i = 0; i < items.length(); i++) {
                try {
                    JSONObject item = items.getJSONObject(i);
                    receiptItemList.add(new ReceiptItem(item.getString( key: "NAME"),
                            item.getInt( key: "QUANTITY"), item.getString( key: "TAX"),
                            item.getString( key: "TOTAL_PRICE")));
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing item data for list", e);
                }
            }
        }
        return receiptItemList;
    }

    private JSONArray fetchReceiptData(String endpoint) {  2 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + endpoint,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchReceiptData", e);
        }
        return null;
    }

    @Override 1 usage  Giorgos-Karachristos
    public String refundOrder(int orderID) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ORDER_ID", orderID);

        return HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "refundOrder",  method: "POST", jsonInput);
    }
}
```

A.5.11 Register

RegisterDAO.java

```java
RegisterDAO.java
package model.register;

import entities.ReceiptItem;

import java.util.List;

public interface RegisterDAO {  3 usages  1 implementation  Giorgos-Karachristos
    String[] getOrder(int order_ID);  1 usage  1 implementation  Giorgos-Karachristos

    List<ReceiptItem> getItemsList(String orderID);  1 usage  1 implementation  Giorgos-Karachristos

    String completeOrder(int orderID, double subtotal, double discount, double tip, double total);

    String cancelOrder(int orderID);  1 usage  1 implementation  Giorgos-Karachristos
}
```

RegisterModel.java (1/2)

```java
package model.register;

import entities.ReceiptItem;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class RegisterModel implements RegisterDAO {  3 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(RegisterModel.class.getName());  3 usages
    private static final String ENDPOINT = "cashier/register/";  3 usages

    @Override  1 usage  Giorgos-Karachristos
    public String[] getOrder(int order_ID) {
        String[] details = {"", "", ""};
        JSONArray jSONArray = fetchRegisterData( endpoint: "getOrder?orderID=" + order_ID);
        if (jSONArray != null) {
            for (int i = 0; i < jSONArray.length(); i++) {
                try {
                    JSONObject jSONObject = jSONArray.getJSONObject(i);
                    details[0] = jSONObject.getString( key: "NAME");
                    details[1] = jSONObject.getString( key: "ORDER_DATE").substring(0, 11);
                    details[2] = jSONObject.getString( key: "ORDER_DATE").substring( beginIndex: 11);
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing employee data for table", e);
                }
            }
        }
        return details;
    }

    @Override  1 usage  Giorgos-Karachristos *
    public List<ReceiptItem> getItemsList(String orderID) {
        List<ReceiptItem> receiptItemList = new ArrayList<>();
        JSONArray items = fetchRegisterData( endpoint: "getItemsList?orderID=" + orderID);
        if (items != null) {
            for (int i = 0; i < items.length(); i++) {
                try {
                    JSONObject item = items.getJSONObject(i);
                    receiptItemList.add(new ReceiptItem(item.getString( key: "NAME"), item.getInt( key: "QUANTITY"),
                            item.getString( key: "TAX"), item.getString( key: "TOTAL_PRICE")));
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing item data for list", e);
                }
            }
        }
        return receiptItemList;
    }
```

RegisterModel.java (2/2)

```
RegisterModel.java
    private JSONArray fetchRegisterData(String endpoint) {  2 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + endpoint,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchRegisterData", e);
        }
        return null;
    }

    @Override  1 usage  Giorgos-Karachristos
    public String completeOrder(int orderID, double subtotal, double discount, double tip, double total) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ORDER_ID", orderID);
        jsonInput.put("SUBTOTAL", subtotal);
        jsonInput.put("DISCOUNT", discount);
        jsonInput.put("TIP", tip);
        jsonInput.put("TOTAL", total);

        return HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "completeOrder",  method: "POST", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String cancelOrder(int orderID) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("ORDER_ID", orderID);

        return HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "cancelOrder",  method: "POST", jsonInput);
    }
}
```

A.5.12 Reports

ReportsDAO.java

```
ReportsDAO.java
package model.reports;

import javax.swing.table.DefaultTableModel;

public interface ReportsDAO {  3 usages  1 implementation  Giorg
    String[] getSalesLabels();  1 usage  1 implementation  Giorg

    String[] getRefundsLabel();  1 usage  1 implementation  Gio

    String[] getGrossSalesGraph();  1 usage  1 implementation

    String[] getRefundsGraph();  1 usage  1 implementation  Gio

    String[] getDiscountsGraph();  1 usage  1 implementation

    String[] getNetSalesGraph();  1 usage  1 implementation  G

    String[] getGrossProfitGraph();  1 usage  1 implementation

    String[] getInventoryGraph();  1 usage  1 implementation

    String[] getItemsGraph(DefaultTableModel tableModel);

}
```

ReportsModel.java (1/3)

```java
ReportsModel.java
package model.reports;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ReportsModel implements ReportsDAO {  3 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(ReportsModel.class.getName());  3 usages
    private static final String ENDPOINT = "manager/report/";  3 usages

    public String[] getSalesLabels() {  1 usage  Giorgos-Karachristos *
        String[] labels = {"0", "0", "0", "0"};
        JSONArray jSONArray = fetchLabelData( endpoint: "getSalesLabels");
        if (jSONArray != null && !jSONArray.isEmpty()) {
            JSONObject jSONObject = jSONArray.getJSONObject( index: 0);
            labels = new String[]{jSONObject.optString( key: "total_subtotal",  defaultValue: "0"),
                    jSONObject.optString( key: "total_discount",  defaultValue: "0"),
                    jSONObject.optString( key: "net_subtotal",  defaultValue: "0"),
                    jSONObject.optString( key: "total_net_profit",  defaultValue: "0")};
        }
        return labels;
    }

    public String[] getRefundsLabel() {  1 usage  Giorgos-Karachristos
        String[] label = {"0"};
        JSONArray jSONArray = fetchLabelData( endpoint: "getRefundsLabel");
        if (jSONArray != null && !jSONArray.isEmpty()) {
            label[0] = jSONArray.getJSONObject( index: 0).optString( key: "total_refunds",  defaultValue: "0");
        }
        return label;
    }

    private JSONArray fetchLabelData(String endpoint) {  2 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + endpoint,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchLabelData for " + endpoint, e);
        }
        return null;
    }

    public String[] getGrossSalesGraph() {  1 usage  Giorgos-Karachristos
        return fetchGraphData( endpoint: "getGrossSalesGraph",  key: "DATE",  key1: "total_completed_subtotal");
    }

    public String[] getRefundsGraph() {  1 usage  Giorgos-Karachristos
        return fetchGraphData( endpoint: "getRefundsGraph",  key: "DATE",  key1: "total_refunt_total");
    }

    public String[] getDiscountsGraph() {  1 usage  Giorgos-Karachristos
        return fetchGraphData( endpoint: "getDiscountsGraph",  key: "DATE",  key1: "total_completed_discount");
    }
```

ReportsModel.java (2/3)

```java
public String[] getNetSalesGraph() {  1 usage  Giorgos-Karachristos
    return fetchGraphData( endpoint: "getNetSalesGraph",  key: "DATE",  key1: "total_net_revenue");
}

public String[] getGrossProfitGraph() {  1 usage  Giorgos-Karachristos
    return fetchGraphData( endpoint: "getGrossProfitGraph",  key: "DATE",  key1: "total_gross_profit");
}

public String[] getInventoryGraph() {  1 usage  Giorgos-Karachristos
    return fetchGraphData( endpoint: "getInventoryGraph",  key: "NAME",  key1: "QUANTITY");
}

public String[] getItemsGraph(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos
    tableModel.setRowCount(0);
    ArrayList<String> graphData = new ArrayList<>();
    Object[] rowData = new Object[2];
    try {
        String response = HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "getItemsGraph",  method: "GET",  jsonInput: null);
        if (response != null) {
            JSONArray jSONArray = new JSONArray(response);
            for (int i = 0; i < jSONArray.length(); i++) {
                JSONObject jSONObject = jSONArray.getJSONObject(i);
                rowData[0] = i + 1;
                rowData[1] = jSONObject.getString( key: "NAME");
                graphData.add(jSONObject.optString( key: "NAME"));
                graphData.add(jSONObject.optString( key: "order_count",  defaultValue: "0"));
                graphData.add(jSONObject.optString( key: "profit_margin",  defaultValue: "0"));
                tableModel.addRow(rowData);
            }
        }
    } catch (JSONException e) {
        LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in getItemsGraph", e);
    }
    return get3ValuesAsString(graphData);
}

private String[] fetchGraphData(String endpoint, String key, String key1) {  6 usages  Giorgos-Karachristos
    ArrayList<String> graphData = new ArrayList<>();
    try {
        String response = HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + endpoint,  method: "GET",  jsonInput: null);
        if (response != null) {
            JSONArray jSONArray = new JSONArray(response);
            for (int i = 0; i < jSONArray.length(); i++) {
                JSONObject jSONObject = jSONArray.getJSONObject(i);
                graphData.add(jSONObject.getString(key));
                graphData.add(jSONObject.optString(key1,  defaultValue: "0"));
            }
        }
    } catch (JSONException e) {
        LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchGraphData for " + endpoint, e);
    }
    return get2ValuesAsString(graphData);
}
```

ReportsModel.java (3/3)

```java
    private String[] get2ValuesAsString(ArrayList<String> list) {
        String[] s = new String[2];

        StringBuilder yString = new StringBuilder();
        StringBuilder xString = new StringBuilder();

        for (int i = 0; i < list.size(); i += 2) {
            if (!xString.isEmpty()) {
                xString.append(",");
            }
            xString.append(list.get(i));

            if (!yString.isEmpty()) {
                yString.append(",");
            }
            yString.append(list.get(i + 1));
        }
        s[0] = xString.toString();
        s[1] = yString.toString();
        if (s[1].isEmpty()) {
            s[1] = "0";
        }
        return s;
    }

    private String[] get3ValuesAsString(ArrayList<String> list) {
        String[] s = new String[3];

        StringBuilder nameString = new StringBuilder();
        StringBuilder xString = new StringBuilder();
        StringBuilder yString = new StringBuilder();

        for (int i = 0; i < list.size(); i += 3) {
            if (!nameString.isEmpty()) {
                nameString.append(",");
            }
            nameString.append(list.get(i));

            if (!xString.isEmpty()) {
                xString.append(",");
            }
            xString.append(list.get(i + 1));

            if (!yString.isEmpty()) {
                yString.append(",");
            }
            yString.append(list.get(i + 2));
        }
        s[0] = nameString.toString();
        s[1] = xString.toString();
        s[2] = yString.toString();
        return s;
    }
}
```

A.5.13 Restaurant

RestaurantDAO.java

```java
package model.restaurant;

public interface RestaurantDAO {  3 usages  1 implementation  Giorgos-Karachristos
    String[] getRestaurant();  1 usage  1 implementation  Giorgos-Karachristos

    String updateRestaurant(String name, String address, String phone, int tables);

    String getRestaurantName();  1 usage  1 implementation  Giorgos-Karachristos
}
```

RestaurantModel.java (1/2)

```java
package model.restaurant;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import java.util.logging.Level;
import java.util.logging.Logger;

public class RestaurantModel implements RestaurantDAO {  3 usages  Giorgos-Karachristos

    private static final Logger LOGGER = Logger.getLogger(RestaurantModel.class.getName());  3 usages
    private static final String ENDPOINT = "manager/restaurant";  2 usages

    @Override  1 usage  Giorgos-Karachristos
    public String[] getRestaurant() {
        String[] restaurant = {"", "", "", ""};
        JSONArray restaurantInfo = fetchRestaurantData();
        if (restaurantInfo != null) {
            for (int i = 0; i < restaurantInfo.length(); i++) {
                try {
                    JSONObject restaurantObject = restaurantInfo.getJSONObject(i);
                    restaurant[0] = restaurantObject.getString( key: "NAME");
                    restaurant[1] = restaurantObject.getString( key: "ADDRESS");
                    restaurant[2] = restaurantObject.getString( key: "PHONE");
                    restaurant[3] = restaurantObject.getString( key: "TABLES");
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing restaurantObject data for table", e);
                }
            }
        }
        return restaurant;
    }
```

RestaurantModel.java (2/2)

```
RestaurantModel.java
    @Override  1 usage  Giorgos-Karachristos
    public String updateRestaurant(String name, String address, String phone,int tables) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("NAME", name);
        jsonInput.put("ADDRESS", address);
        jsonInput.put("PHONE", phone);
        jsonInput.put("TABLES", tables);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "PUT", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String getRestaurantName() {
        String result = null;
        JSONArray restaurantInfo = fetchRestaurantData();
        if (restaurantInfo != null) {
            for (int i = 0; i < restaurantInfo.length(); i++) {
                try {
                    JSONObject restaurantObject = restaurantInfo.getJSONObject(i);
                    result = restaurantObject.getString( key: "NAME");
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing restaurantObject data for list", e);
                }
            }
        }
        return result;
    }

    private JSONArray fetchRestaurantData() {  2 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchRestaurantData", e);
        }
        return null;
    }
}
```

A.5.14 Role

RoleDAO.java

```
RoleDAO.java
package model.role;

import entities.Role;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public interface RoleDAO {  3 usages  1 implementation  Giorgos-Karachristos
    void getRoleTable(DefaultTableModel tableModel);  1 usage  1 implen

    List<Role> getRoleList(DefaultComboBoxModel<String> boxModel);

    void getRoleName(DefaultComboBoxModel<String> boxModel);  1 usag

    void getPOSAccess(DefaultComboBoxModel<String> boxModel);  1 usa

    String insertRole(String roleName, String access);  1 usage  1 imp

    String updateRole(String roleName, String access);  1 usage  1 imp

    String deleteRole(String roleName);  1 usage  1 implementation  Giorg
}
```

RoleModel.java (1/3)

```
RoleModel.java
package model.role;

import entities.Role;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class RoleModel implements RoleDAO {  3 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(RoleModel.class.getName());
    private static final String ENDPOINT = "manager/role";  4 usages
```

RoleModel.java (2/3)

```java
@Override  1 usage  Giorgos-Karachristos
public void getRoleTable(DefaultTableModel tableModel) {
    tableModel.setRowCount(0);
    JSONArray roles = fetchRoleData();
    if (roles != null) {
        for (int i = 0; i < roles.length(); i++) {
            try {
                JSONObject role = roles.getJSONObject(i);
                Object[] rowData = new Object[3];
                rowData[0] = role.getString( key: "NAME");
                rowData[1] = role.getString( key: "POS_ACCESS");
                rowData[2] = role.getString( key: "COUNT_EMPLOYEES");
                tableModel.addRow(rowData);
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing role data for table", e);
            }
        }
    }
}

//DeleteRole
//UpdateRole
@Override  1 usage  Giorgos-Karachristos *
public List<Role> getRoleList(DefaultComboBoxModel<String> boxModel) {
    boxModel.removeAllElements();
    List<Role> roleArrayList = new ArrayList<>();
    boxModel.addElement( anObject: "");
    JSONArray roles = fetchRoleData();
    if (roles != null) {
        for (int i = 0; i < roles.length(); i++) {
            try {
                JSONObject role = roles.getJSONObject(i);
                boxModel.addElement(role.getString( key: "NAME"));
                roleArrayList.add(new Role(role.getString( key: "NAME"), role.getString( key: "POS_ACCESS"),
                        role.getString( key: "COUNT_EMPLOYEES")));
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing role data for list", e);
            }
        }
    }
    return roleArrayList;
}

private JSONArray fetchRoleData() {  3 usages  Giorgos-Karachristos
    try {
        String response = HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "GET",  jsonInput: null);
        if (response != null) {
            return new JSONArray(response);
        }
    } catch (JSONException e) {
        LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchRoleData", e);
    }
    return null;
}
```

RoleModel.java (3/3)

```java
//InsertEmployee
//UpdateEmployee
@Override  1 usage  Giorgos-Karachristos
public void getRoleName(DefaultComboBoxModel<String> boxModel) {
    boxModel.removeAllElements();
    boxModel.addElement( anObject: "");
    JSONArray roles = fetchRoleData();
    if (roles != null) {
        for (int i = 0; i < roles.length(); i++) {
            try {
                JSONObject role = roles.getJSONObject(i);
                boxModel.addElement(role.getString( key: "NAME"));
            } catch (JSONException e) {
                LOGGER.log(Level.WARNING,  msg: "Error parsing role data for ComboBox", e);
            }
        }
    }
}

@Override  1 usage  Giorgos-Karachristos
public void getPOSAccess(DefaultComboBoxModel<String> boxModel) {
    boxModel.addElement( anObject: "");
    String endpoint = "manager/pos_access";
    try {
        String response = HttpRequestUtil.sendHttpRequest(endpoint,  method: "GET",  jsonInput: null);
        if (response != null) {
            JSONArray roles = new JSONArray(response);
            for (int i = 0; i < roles.length(); i++) {
                try {
                    JSONObject role = roles.getJSONObject(i);
                    boxModel.addElement(role.getString( key: "NAME"));
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing role data for table", e);
                }
            }
        }
    } catch (JSONException e) {
        LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in getPOSAccess", e);
    }
}

@Override  1 usage  Giorgos-Karachristos
public String insertRole(String roleName, String access) {
    JSONObject jsonInput = new JSONObject();
    jsonInput.put("NAME", roleName);
    jsonInput.put("POS_ACCESS", access);

    return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "POST", jsonInput);
}

@Override  1 usage  Giorgos-Karachristos
public String updateRole(String roleName, String access) {
    JSONObject jsonInput = new JSONObject();
    jsonInput.put("NAME", roleName);
    jsonInput.put("POS_ACCESS", access);

    return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "PUT", jsonInput);
}

@Override  1 usage  Giorgos-Karachristos
public String deleteRole(String roleName) {
    JSONObject jsonInput = new JSONObject();
    jsonInput.put("NAME", roleName);

    return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "DELETE", jsonInput);
}
}
```

A.5.15 Transaction

TransactionDAO.java

```
TransactionDAO.java
package model.transaction;

import entities.TransactionItem;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.List;

public interface TransactionDAO {  3 usages  1 implementation  Giorgos-Karachristos
    void getTransactionTable(DefaultTableModel tableModel);  1 usage  1 implementation  Giorgos-Karachristos

    List<TransactionItem> getTransactionID(DefaultComboBoxModel<String> boxModel);  1 usage  1 implementation  Giorgos-Karachristos

    String insertTransaction(int p_item_id, int p_old_quantity, String p_type, int p_transaction_quantity, int p_new_quantity);

    String deleteInventory(int id);  1 usage  1 implementation  Giorgos-Karachristos
}
```

TransactionModel.java (1/2)

```
TransactionModel.java
package model.transaction;

import entities.TransactionItem;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import util.HttpRequestUtil;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class TransactionModel implements TransactionDAO {  3 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(TransactionModel.class.getName());  3 usa
    private static final String ENDPOINT = "manager/transaction";  3 usages

    @Override  1 usage  Giorgos-Karachristos
    public void getTransactionTable(DefaultTableModel tableModel) {
        tableModel.setRowCount(0);
        JSONArray transactions = fetchTransactionData( variable: "Table");
        if (transactions != null) {
            for (int i = 0; i < transactions.length(); i++) {
                try {
                    JSONObject transaction = transactions.getJSONObject(i);
                    Object[] rowData = new Object[9];
                    rowData[0] = transaction.getString( key: "TRANSACTION_ID");
                    rowData[1] = transaction.getString( key: "ITEM_ID");
                    rowData[2] = transaction.getString( key: "ITEM_CODE");
                    rowData[3] = transaction.getString( key: "NAME");
                    rowData[4] = transaction.getString( key: "OLD_QUANTITY");
                    rowData[5] = transaction.getString( key: "TYPE");
                    rowData[6] = transaction.getString( key: "TRANSACTION_QUANTITY");
                    rowData[7] = transaction.getString( key: "TRANSACTION_DATE");
                    rowData[8] = transaction.getString( key: "STATUS");
                    tableModel.addRow(rowData);
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing transaction data for table", e);
                }
            }
        }
    }
```

TransactionModel.java (2/2)

```java
TransactionModel.java
    @Override  1 usage  Giorgos-Karachristos *
    public List<TransactionItem> getTransactionID(DefaultComboBoxModel<String> boxModel) {
        boxModel.removeAllElements();
        List<TransactionItem> rowData = new ArrayList<>();
        boxModel.addElement( anObject: "");
        JSONArray transactions = fetchTransactionData( variable: "ID");
        if (transactions != null) {
            for (int i = 0; i < transactions.length(); i++) {
                try {
                    JSONObject transaction = transactions.getJSONObject(i);
                    boxModel.addElement(transaction.getString( key: "TRANSACTION_ID"));
                    rowData.add(new TransactionItem(transaction.getString( key: "TRANSACTION_ID"),
                            transaction.getString( key: "ITEM_ID"), transaction.getString( key: "ITEM_CODE"),
                            transaction.getString( key: "NAME"), transaction.getString( key: "OLD_QUANTITY"),
                            transaction.getString( key: "TYPE"), transaction.getString( key: "TRANSACTION_QUANTITY"),
                            transaction.getString( key: "TRANSACTION_DATE"), transaction.getString( key: "STATUS")));
                } catch (JSONException e) {
                    LOGGER.log(Level.WARNING,  msg: "Error parsing transaction data for list", e);
                }
            }
        }
        return rowData;
    }

    private JSONArray fetchTransactionData(String variable) {  2 usages  Giorgos-Karachristos
        try {
            String response = HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "?variable=" + variable,  method: "GET",  jsonInput: null);
            if (response != null) {
                return new JSONArray(response);
            }
        } catch (JSONException e) {
            LOGGER.log(Level.WARNING,  msg: "Error parsing JSON in fetchTransactionData", e);
        }
        return null;
    }

    @Override  1 usage  Giorgos-Karachristos
    public String insertTransaction(int p_item_id, int p_old_quantity, String p_type, int p_transaction_quantity, int p_new_quantity) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("TRANSACTION_ID", p_item_id);
        jsonInput.put("OLD_QUANTITY", p_old_quantity);
        jsonInput.put("TYPE", p_type);
        jsonInput.put("TRANSACTION_QUANTITY", p_transaction_quantity);
        jsonInput.put("NEW_QUANTITY", p_new_quantity);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "POST", jsonInput);
    }

    @Override  1 usage  Giorgos-Karachristos
    public String deleteInventory(int id) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("TRANSACTION_ID", id);

        return HttpRequestUtil.sendHttpRequest(ENDPOINT,  method: "DELETE", jsonInput);
    }
}
```

A.5.16 Waiter

WaiterDAO.java

```java
WaiterDAO.java
package model.waiter;

public interface WaiterDAO {  3 usages  1 implementation  Gio
    String getWaiterToken(String name, String surname);
}
```

WaiterModel.java

```java
package model.waiter;

import org.json.JSONObject;
import util.HttpRequestUtil;

public class WaiterModel implements WaiterDAO {  2 usages  Giorgos-Karachristos
    private static final String ENDPOINT = "waiter/";  1 usage

    @Override  1 usage  Giorgos-Karachristos
    public String getWaiterToken(String name, String surname) {
        JSONObject jsonInput = new JSONObject();
        jsonInput.put("NAME", name);
        jsonInput.put("SURNAME", surname);
        return HttpRequestUtil.sendHttpRequest( endpoint: ENDPOINT + "insertWaiterToken",  method: "POST", jsonInput);
    }

}
```

A.6 Python

generate_bar_graph.py

```python
import matplotlib.pyplot as plt
import sys
import os

def create_bar_chart(x_labels, y_values, x_title, y_title, title, image_name):  1 usage  Giorgos-Karachristc
    try:
        # Create the Graphs directory if it doesn't exist
        if not os.path.exists('Graphs'):
            os.makedirs('Graphs')

        x = x_labels.split(',')
        # Convert strings of numbers to lists of floats
        y = list(map(float, y_values.split(',')))

        # Create a larger figure
        plt.figure(figsize=(14, 8))

        # Create the bar chart
        plt.bar(x, y, color='skyblue')
        plt.title(title)
        plt.xlabel(x_title)
        plt.ylabel(y_title)

        # Rotate the x-axis labels for better readability
        plt.xticks(rotation=45, ha='right', fontsize=10)

        # Adjust layout to avoid clipping the labels
        plt.tight_layout()

        # Save the bar chart as an image in the Graphs folder
        image_path = os.path.join('Graphs', f'{image_name}.png')
        plt.savefig(image_path)

        # Close the plot to free up memory
        plt.close()

    except Exception as e:
        print(f"Error creating the graph: {e}")

if __name__ == '__main__':
    # Expecting x_values, y_values, x_title, y_title, title, and image_name as command-line arguments
    if len(sys.argv) == 7:
        x_labels = sys.argv[1]
        y_values = sys.argv[2]
        x_title = sys.argv[3]
        y_title = sys.argv[4]
        title = sys.argv[5]
        image_name = sys.argv[6]

        create_bar_chart(x_labels, y_values, x_title, y_title, title, image_name)
    else:
        print("Error: Expected 6 arguments (x_labels, y_values, x_title, y_title, title, image_name)")
```

generate_line_graph.py

```python
generate_line_graph.py
import math

import matplotlib.pyplot as plt
import sys
import os

from matplotlib.ticker import MultipleLocator


def create_line_chart(x_labels, y_values, x_title, y_title, title, image_name):  1 usage  Giorgos-Karachris
    try:
        # Create the Graphs directory if it doesn't exist
        if not os.path.exists('Graphs'):
            os.makedirs('Graphs')

        x = x_labels.split(',')
        # Convert strings of numbers to lists of integers
        y = list(map(float, y_values.split(',')))

        # Create a larger figure
        plt.figure(figsize=(14, 8))

        # Create the line chart
        plt.plot( *args: x, y, marker='o')
        plt.title(title)
        plt.xlabel(x_title)
        plt.ylabel(y_title)

        # Rotate the x-axis labels for better readability
        plt.xticks(rotation=45, ha='right', fontsize=10)

        # Adjust y-axis limits
        y_min = min(y)
        y_max = max(y)

        if y_min == 0:
            min_padding = 0
        else:
            min_padding = 5

        y_range = y_max - y_min

        if y_max != y_min:
            max_padding = y_range * 0.1
        else:
            max_padding = 1

        plt.ylim( *args: y_min - min_padding, y_max + max_padding)

        # Round to a nice power of ten
        step = 10 ** math.floor(math.log10(y_range / 10))

        # Get the current Axes object and set Y-axis ticks to appear every 'step' units
        plt.gca().yaxis.set_major_locator(MultipleLocator(step))

        #add a grid
        plt.grid( visible: True, linestyle='--', alpha=0.6)
        # Adjust layout to avoid clipping the labels
        plt.tight_layout()

        # Save the line chart as an image in the Graphs folder
        image_path = os.path.join('Graphs', f'{image_name}.png')
        plt.savefig(image_path)

        # Close the plot to free up memory
        plt.close()

    except Exception as e:
        print(f"Error creating the graph: {e}")

if __name__ == '__main__':
    # Expecting x_values, y_values, x_title, y_title, title, and image_name as command-line arguments
    if len(sys.argv) == 7:
        x_labels = sys.argv[1]
        y_values = sys.argv[2]
        x_title = sys.argv[3]
        y_title = sys.argv[4]
        title = sys.argv[5]
        image_name = sys.argv[6]

        create_line_chart(x_labels, y_values, x_title, y_title, title, image_name)
    else:
        print("Error: Expected 6 arguments (x_labels, y_values, x_title, y_title, title, image_name)")
```

generate_magic_quadrant.py

```python
generate_magic_quadrant.py
import matplotlib.pyplot as plt
import pandas as pd
import sys
from adjustText import adjust_text
import numpy as np
import os

def create_scatter_plot(names, x_values, y_values):  1 usage  & Giorgos-Karachristos
    try:
        # Create the Graphs directory if it doesn't exist
        if not os.path.exists('Graphs'):
            os.makedirs('Graphs')

        # Splitting the names string into a list
        n = names.split(',')

        # Convert strings of numbers to lists of integers/floats
        x = list(map(int, x_values.split(',')))
        y = list(map(float, y_values.split(',')))

        # Creating a dictionary to hold the data
        data = {
            'Item': n,
            'Sales Frequency': x,
            'Gross Margin (Profit)': y
        }

        # Creating a DataFrame from the data dictionary
        df = pd.DataFrame(data)

        # Create the plot
        fig, ax = plt.subplots(figsize=(8, 8))

        # Scatter plot
        ax.scatter(df['Sales Frequency'], df['Gross Margin (Profit)'])

        # Add numbers for each point
        texts = []
        for i, row in df.iterrows():
            text = ax.text(row['Sales Frequency'], row['Gross Margin (Profit)'], str(i+1), fontsize=12)
            texts.append(text)

        # Adjust the text labels to avoid overlaps
        adjust_text(texts, arrowprops=dict(arrowstyle='->', color='gray'))

        # Allow negatives
        ax.set_ylim(min(0, min(y) - 5), max(y) + 5)
        ax.set_xlim(min(0, min(x) - 5), max(x) + 5)

        # Median lines
        median_x = np.median(x)
        median_y = np.median(y)

        # Add horizontal and vertical lines to create quadrants
        plt.axhline(y=median_y, color='gray', linestyle='--', linewidth=1) # Horizontal line (Profit threshold)
        plt.axvline(x=median_x, color='gray', linestyle='--', linewidth=1) # Vertical line (Sales Frequency threshold)

        # Add quadrant labels
        ax.text(1, 1.05, 'Leaders', transform=ax.transAxes, fontsize=16, color='green', ha='right', va='top')
        ax.text(0, 1.05, 'Challengers', transform=ax.transAxes, fontsize=16, color='blue', ha='left', va='top')
        ax.text(0, -0.08, 'Niche Players', transform=ax.transAxes, fontsize=16, color='red', ha='left', va='bottom')
        ax.text(1, -0.08, 'Visionaries', transform=ax.transAxes, fontsize=16, color='purple', ha='right', va='bottom')

        plt.xlabel( xlabel: 'Sales Frequency', fontsize=14)
        plt.ylabel( ylabel: 'Gross Margin (Profit)', fontsize=14)
        plt.title( label: 'Magic Quadrant', fontsize=18, fontweight='bold')

        plt.grid( visible: True, linestyle='--', alpha=0.5)
        plt.tight_layout()

        # Save the Magic Quadrant as an image in the Graphs folder
        plt.savefig(os.path.join('Graphs', 'magicQuadrant.png'))

        # Close the plot to free up memory
        plt.close()

    except Exception as e:
        print(f"Error creating the graph: {e}")

if __name__ == '__main__':
    if len(sys.argv) == 4:
        names = sys.argv[1]
        x_values = sys.argv[2]
        y_values = sys.argv[3]
        create_scatter_plot(names, x_values, y_values)
    else:
        print("Error: Expected 3 arguments (names, x_values, y_values)")
```

A.7 Util

CallPython.java (1/2)

CallPython.java

```
package util;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.image.BufferedImage;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.logging.Level;
import java.util.logging.Logger;

public class CallPython {  7 usages  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(CallPython.class.getName());  2 usages

    public void callPythonLineGraph(String xData, String yData, String xTitle, String yTitle,  5 usages  new *
                                    String title, JLabel lineGraphLabel) {
        callPythonGraph( script: "C:\\Users\\USER\\IdeaProjects\\PointOfSale\\src\\python\\generate_line_graph.py",
                xData, yData, xTitle, yTitle, title, lineGraphLabel);
    }

    public void callPythonBarGraph(String xData, String yData, String xTitle, String yTitle,  1 usage  new *
                                   String title, JLabel barGraphLabel) {
        callPythonGraph( script: "C:\\Users\\USER\\IdeaProjects\\PointOfSale\\src\\python\\generate_bar_graph.py",
                xData, yData, xTitle, yTitle, title, barGraphLabel);
    }

    public void callPythonGraph(String names, String xData, String yData, JLabel lineGraphLabel) {  1 usage  Giorgos-Karachristos *
        try {
            // Pass dynamic data as arguments to Python script
            ProcessBuilder pb = new ProcessBuilder( ...command: "C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python312\\python.exe",
                    "C:\\Users\\USER\\IdeaProjects\\PointOfSale\\src\\python\\generate_magic_quadrant.py", names, xData, yData);

            pb.redirectErrorStream(true);  // Redirect error stream to standard output
            Process process = pb.start();

            // Capture the output from the process (Python script)
            BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);  // Print Python script output
            }

            int exitCode = process.waitFor();
            System.out.println("Python script exited with code: " + exitCode);

            // Load and refresh the graph image
            refreshGraphImage(lineGraphLabel);

        } catch (IOException | InterruptedException e) {
            LOGGER.log(Level.WARNING,  msg: "InterruptedException", e);
        }
    }
```

CallPython.java (2/2)

```
CallPython.java
    private void callPythonGraph(String script, String x, String y, String x_title, String y_title,   2 usages  Giorgos-Karachristos *
                                 String title, JLabel jLabel) {
        try {
            // Pass dynamic data as arguments to Python script
            ProcessBuilder pb = new ProcessBuilder( ...command: "C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python312\\python.exe",
                    script, x, y, x_title, y_title, title, jLabel.getName());

            pb.redirectErrorStream(true);  // Redirect error stream to standard output
            Process process = pb.start();

            // Capture the output from the process (Python script)
            BufferedReader reader = new BufferedReader(new InputStreamReader(process.getInputStream()));
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);  // Print Python script output
            }

            int exitCode = process.waitFor();
            System.out.println("Python script exited with code: " + exitCode);

            // Load and refresh the graph image
            refreshGraphImage(jLabel);

        } catch (IOException | InterruptedException e) {
            LOGGER.log(Level.WARNING,  msg: "InterruptedException", e);
        }
    }

    private void refreshGraphImage(JLabel graphLabel) throws IOException {  2 usages  Giorgos-Karachristos
        String name = graphLabel.getName();
        BufferedImage bufferedImage = ImageIO.read(new File( pathname: "Graphs/" + name + ".png"));
        ImageIcon graphIcon = new ImageIcon(bufferedImage);  // Create a new ImageIcon

        graphLabel.setIcon(graphIcon);
        graphLabel.revalidate();  // Revalidate the label to update the UI
    }
}
```

CustomColorChooser.java (1/2)

```
CustomColorChooser.java
package util;

import javax.swing.*;
import javax.swing.colorchooser.AbstractColorChooserPanel;
import java.awt.*;
import java.awt.event.ActionEvent;

public class CustomColorChooser extends JDialog {  12 usages  Giorgos-Karachristos

    public CustomColorChooser(JButton colorButton) {  4 usages  Giorgos-Karachristc

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setResizable(false);
        this.setTitle("Color Chooser");
        this.setLayout(new GridBagLayout());

        GridBagConstraints gbc;

        JColorChooser colorChooser = new JColorChooser();
        colorChooser.setColor(colorButton.getBackground());

        AbstractColorChooserPanel[] panels = colorChooser.getChooserPanels();
        for (AbstractColorChooserPanel panel : panels) {
            if (!"HSV".equals(panel.getDisplayName())) {
                colorChooser.removeChooserPanel(panel);
            }
        }
```

CustomColorChooser.java (2/2)

CustomColorChooser.java

```
        removeComponents(colorChooser, className: "javax.swing.JSpinner");
        removeComponents(colorChooser, className: "javax.swing.JRadioButton");
        removeComponents(colorChooser, className: "javax.swing.JLabel");
        removeComponents(colorChooser, className: "javax.swing.JSlider");

        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        this.add(colorChooser, gbc);

        JPanel jPanel1 = new JPanel();
        JButton okButton = new JButton();
        okButton.setText("OK");
        okButton.addActionListener((ActionEvent evt) -> {
            Color newColor = colorChooser.getColor();
            colorButton.setBackground(newColor);
            this.dispose();
        });
        jPanel1.add(okButton);

        JButton cancelButton = new JButton();
        cancelButton.setText("Cancel");
        cancelButton.addActionListener((ActionEvent evt) -> this.dispose());
        jPanel1.add(cancelButton);

        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 1;
        this.add(jPanel1, gbc);

        this.pack();
        this.setLocationRelativeTo(null);
    }

    // Method to recursively remove components with the specified class name
    public static void removeComponents(Container container, String className) {
        Component[] components = container.getComponents();
        for (Component component : components) {
            if (component.getClass().getName().equals(className)) {
                container.remove(component);
            } else if (component instanceof Container) {
                removeComponents((Container) component, className);
            }
        }
        // Revalidate and repaint the container to reflect the changes
        container.revalidate();
        container.repaint();
    }
}
```

CustomDocumentFilter.java

```java
package util;

import javax.swing.*;
import javax.swing.text.AbstractDocument;
import javax.swing.text.AttributeSet;
import javax.swing.text.BadLocationException;
import javax.swing.text.DocumentFilter;
import java.awt.*;

public class CustomDocumentFilter {

    public void setDocumentFilter(JTextArea textArea, int charLimit) {
        ((AbstractDocument) textArea.getDocument()).setDocumentFilter(new DocumentFilter() {
            @Override
            public void insertString(DocumentFilter.FilterBypass fb, int offset,
                                     String string, AttributeSet attr) throws BadLocationException {
                if ((fb.getDocument().getLength() + string.length()) <= charLimit) {
                    super.insertString(fb, offset, string, attr);
                } else {
                    Toolkit.getDefaultToolkit().beep();
                }
            }

            @Override
            public void replace(DocumentFilter.FilterBypass fb, int offset, int length,
                                String text, AttributeSet attrs) throws BadLocationException {
                if ((fb.getDocument().getLength() + text.length() - length) <= charLimit) {
                    super.replace(fb, offset, length, text, attrs);
                } else {
                    Toolkit.getDefaultToolkit().beep();
                }
            }
        });
    }
}
```

CustomImageIcon.java

```java
package util;

import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.net.URL;
import java.util.logging.Level;
import java.util.logging.Logger;

public class CustomImageIcon {

    private static final Logger LOGGER = Logger.getLogger(CustomImageIcon.class.getName());

    public static ImageIcon getScaledImageIcon(String imagePath, int width, int height) {
        URL location = CustomImageIcon.class.getClassLoader().getResource(imagePath);
        if (location != null) {
            ImageIcon icon = new ImageIcon(location);
            Image scaledImage = icon.getImage().getScaledInstance(width, height, Image.SCALE_SMOOTH);
            return new ImageIcon(scaledImage);
        } else {
            LOGGER.log(Level.WARNING, msg: "Image not found: {0}", imagePath);
            BufferedImage blankImage = new BufferedImage(width, height, BufferedImage.TYPE_INT_ARGB);
            return new ImageIcon(blankImage);
        }
    }
}
```

CustomTableCellRenderer.java

```
CustomTableCellRenderer.java
package util;

import javax.swing.*;
import javax.swing.table.DefaultTableCellRenderer;
import java.awt.*;

public class CustomTableCellRenderer extends DefaultTableCellRenderer {  3 usages  Giorgos-Karachristos *

    @Override  Giorgos-Karachristos *
    public Component getTableCellRendererComponent(JTable table, Object value,
                                                   boolean isSelected, boolean hasFocus, int row, int column) {
        Component cell = super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);
        cell.setBackground(Color.decode((String) value));
        setText("");
        return cell;
    }
}
```

ExportPanelToImage.java

```
ExportPanelToImage.java
package util;

import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.*;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;

public class ExportPanelToImage {  4 usages  Giorgos-Karachristos
    public static void exportPanelToImage(JPanel panel, String fileName) throws IOException {  2 usages  Giorgos-Kara
        // Get the current working directory
        String currentDir = System.getProperty("user.dir");
        // Set the directory path relative to the current working directory
        String directoryPath = currentDir + "/Receipts";

        File directory = new File(directoryPath);
        if (!directory.exists()) {
            directory.mkdirs();
        }

        // Create a BufferedImage with the size of the JPanel
        BufferedImage image = new BufferedImage(panel.getWidth(), panel.getHeight(), BufferedImage.TYPE_INT_RGB);

        // Paint the JPanel onto the BufferedImage
        Graphics2D g2d = image.createGraphics();
        panel.paint(g2d);
        g2d.dispose();

        // Save the BufferedImage as an image file (e.g., PNG)
        File outputFile = new File( pathname: directoryPath + "/" + fileName);
        ImageIO.write(image,  formatName: "png", outputFile);
    }
}
```

HttpRequestUtil.java

```java
package util;

import org.json.JSONObject;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.nio.charset.StandardCharsets;
import java.util.logging.Level;
import java.util.logging.Logger;

public class HttpRequestUtil {  Giorgos-Karachristos *

    private static final Logger LOGGER = Logger.getLogger(HttpRequestUtil.class.getName());  2 usages
    private static final String API_URL = "https://users.it.teithe.gr/~it185192/pos/pos.php/";  1 usage

    public static String sendHttpRequest(String endpoint, String method, JSONObject jsonInput) {  Giorgos-Karachristos
        StringBuilder response = new StringBuilder();
        try {
            HttpURLConnection connection = (HttpURLConnection) new URL( spec: API_URL + endpoint).openConnection();
            connection.setRequestMethod(method);

            // Set the request content type to JSON
            connection.setRequestProperty("Content-Type", "application/json; utf-8");

            if (jsonInput != null) {
                // Enable output streams
                connection.setDoOutput(true);
                // Convert the JSON object to bytes and send it in the request
                try (OutputStream os = connection.getOutputStream()) {
                    os.write(jsonInput.toString().getBytes(StandardCharsets.UTF_8));
                }
            }

            int responseCode = connection.getResponseCode();
            if (responseCode == HttpURLConnection.HTTP_OK) {
                try (BufferedReader bufferedReader = new BufferedReader(
                        new InputStreamReader(connection.getInputStream(), StandardCharsets.UTF_8))) {
                    String responseLine;
                    while ((responseLine = bufferedReader.readLine()) != null) {
                        response.append(responseLine.trim());
                    }
                }
            } else {
                LOGGER.log(Level.SEVERE,  msg: "{0} request failed. Response Code: {1}",
                        new Object[]{method, responseCode});
                return null;
            }

        } catch (IOException e) {
            LOGGER.log(Level.SEVERE,  msg: "Network error in " + method + " request", e);
        }
        String result = response.toString();
        if (result.startsWith("\"") && result.endsWith("\"")) {
            result = result.substring(1, result.length() - 1);
        }
        return result;
    }
}
```

A.8 View

CustomFrame.java

```
CustomFrame.java
package view;

import javax.swing.*;
import java.awt.*;

public class CustomFrame extends JFrame {  4 usages  Giorgos-Karachristos

    public CustomFrame() {  1 usage  Giorgos-Karachristos
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setExtendedState(JFrame.MAXIMIZED_BOTH);

        this.setResizable(true);
        this.setSize( width: 800,  height: 600);
        this.setLocationRelativeTo(null);

        this.getContentPane().setBackground(new Color( r: 250,  g: 188,  b: 63));

        ImageIcon icon = new ImageIcon(getClass().getClassLoader().getResource( name: "resources/AppIcon.png"));
        this.setIconImage(icon.getImage());
    }
}
```

HeaderView.java

```
HeaderView.java
package view;

import controller.pos.POSController;
import util.CustomImageIcon;

import javax.swing.*;
import java.awt.*;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class HeaderView extends JPanel {  3 usages  Giorgos-Karachristos

    public HeaderView(POSController controller, String name, String surname) {  1 usage  Giorgos-Karachristos
        int screenHeight = Toolkit.getDefaultToolkit().getScreenSize().height;
        int panel1Height = (int) (screenHeight * 0.055);
        this.setPreferredSize(new Dimension(getSize().width, panel1Height));
        this.setLayout(new FlowLayout(FlowLayout.RIGHT));
        this.setBorder(BorderFactory.createMatteBorder( top: 0,  left: 0,  bottom: 3,  right: 0, Color.BLACK));

        JLabel usernameLabel = new JLabel( text: name + " " + surname);
        usernameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        this.add(usernameLabel);

        JMenuItem signOutMenuItem = new JMenuItem( text: "Sign Out");
        signOutMenuItem.addActionListener( ActionEvent e -> controller.onSignOut());

        JPopupMenu popupMenu = new JPopupMenu();
        popupMenu.add(signOutMenuItem);

        ImageIcon imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/User.png",  width: 50,  height: 50);
        JLabel pictureLabel = new JLabel(imageIcon);
        pictureLabel.addMouseListener((MouseAdapter) mouseClicked(e) → {
                popupMenu.show(pictureLabel, e.getX(), e.getY());
        });
        this.add(pictureLabel);
    }
}
```

A.8.1 Cashier

CashierView.java

CashierView.java

```
package view.cashier;

import util.CustomImageIcon;
import view.menu.MenuView;
import view.order.OrdersView;
import view.receipts.ReceiptsView;
import view.register.RegisterView;

import javax.swing.*;
import javax.swing.event.ChangeEvent;
import java.awt.*;

public class CashierView extends JTabbedPane {  2 usages  Giorgos-Karachristos

    public CashierView(String name, String surname) {  1 usage  Giorgos-Karachristos
        this.setTabPlacement(JTabbedPane.TOP);
        this.putClientProperty("JTabbedPane.tabAlignment", "leading");
        this.putClientProperty("JTabbedPane.tabAreaAlignment", "center");
        this.putClientProperty("JTabbedPane.showTabSeparators", true);
        this.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        this.setBackground(new Color( r: 250,  g: 188,  b: 63));

        UIManager.put("TabbedPane.selectedBackground", null);
        UIManager.put("TabbedPane.focusColor", null);

        MenuView menuView = new MenuView(name, surname,  jp: this);
        ImageIcon imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/Menu.png",  width: 50,  height: 50);
        this.addTab( title: "Menu", imageIcon, menuView);

        RegisterView registerView = new RegisterView( jtp: this);
        OrdersView ordersView = new OrdersView( jtp: this, registerView);
        imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/Orders.png",  width: 50,  height: 50);
        this.addTab( title: "Orders", imageIcon, ordersView);

        imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/Register.png",  width: 50,  height: 50);
        this.addTab( title: "Register", imageIcon, registerView);

        ReceiptsView receiptsView = new ReceiptsView();
        imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/Receipts.png",  width: 50,  height: 50);
        this.addTab( title: "Receipts", imageIcon, receiptsView);

        this.addChangeListener((ChangeEvent e) -> {
            this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
            if (this.getSelectedIndex() == 1) {
                ordersView.refreshButtonActionPerformed( force: false);
            }
            if (this.getSelectedIndex() != 2) {
                registerView.setOrderID(-1);
            }
            if (this.getSelectedIndex() == 3) {
                receiptsView.refreshButtonActionPerformed( force: false);
            }
            this.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
        });
    }
}
```

A.8.2 Category

DeleteCategoryView.java (1/4)

DeleteCategoryView.java

```java
package view.category;

import controller.category.CategoryController;
import entities.Category;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class DeleteCategoryView extends JDialog {  2 usages  ♙ Giorgos-Karachristos

    private final JComboBox<String>nameComboBox;  9 usages
    private final JLabel sqlTaxLabel;  5 usages
    private final JLabel sqlColorLabel;  11 usages
    private final JLabel sqlIsActiveLabel;  5 usages
    private final JTextArea jTextArea;  12 usages
    private final DefaultComboBoxModel<String> categoryBoxModel;  2 usages
    private final CategoryController categoryController;  4 usages

    private List<Category> categoryList;  2 usages

    public DeleteCategoryView(DefaultTableModel tableModel) {  1 usage  ♙ Giorgos-Karachristos
        categoryController = new CategoryController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setResizable(false);
        this.setTitle("Delete Category");
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0, 10, 0};
        this.setLayout(layout);

        JLabel nameLabel = new JLabel();
        nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        nameLabel.setText("Name:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 20,  left: 20,  bottom: 5,  right: 5);
        this.add(nameLabel, gbc);

        nameComboBox = new JComboBox<>();
        nameComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        categoryBoxModel = (DefaultComboBoxModel<String>) nameComboBox.getModel();
        populateComboBox();
        nameComboBox.addActionListener( ActionEvent e -> nameComboBoxActionPerformed());
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.ipadx = 100;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
        getContentPane().add(nameComboBox, gbc);
```

DeleteCategoryView.java (2/4)

```
JButton deleteButton = new JButton();
deleteButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
deleteButton.setText("Delete");
deleteButton.addActionListener((ActionEvent evt) -> deleteButtonActionPerformed(tableModel));
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 0;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 20);
this.add(deleteButton, gbc);

JLabel taxLabel = new JLabel();
taxLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
taxLabel.setText("Tax:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.gridwidth = 3;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(taxLabel, gbc);

sqlTaxLabel = new JLabel();
sqlTaxLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlTaxLabel, gbc);

JLabel colorLabel = new JLabel();
colorLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
colorLabel.setText("Color:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.gridwidth = 3;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(colorLabel, gbc);

sqlColorLabel = new JLabel();
sqlColorLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
sqlColorLabel.setText("1234567");
sqlColorLabel.setForeground(this.getBackground());
sqlColorLabel.setBackground(this.getBackground());
sqlColorLabel.setOpaque(true);
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlColorLabel, gbc);

JLabel isActiveLabel = new JLabel();
isActiveLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
isActiveLabel.setText("Is active:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 6;
gbc.gridwidth = 3;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(isActiveLabel, gbc);
```

DeleteCategoryView.java (3/4)

```
DeleteCategoryView.java
        sqlIsActiveLabel = new JLabel();
        sqlIsActiveLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        gbc = new GridBagConstraints();
        gbc.gridx = 4;
        gbc.gridy = 6;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
        this.add(sqlIsActiveLabel, gbc);

        jTextArea = new JTextArea();
        jTextArea.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        jTextArea.setWrapStyleWord(true);
        jTextArea.setLineWrap(true);
        jTextArea.setOpaque(false);
        jTextArea.setEditable(false);
        jTextArea.setFocusable(false);
        jTextArea.setBorder(null);
        gbc = new java.awt.GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 8;
        gbc.gridwidth = 5;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets( top: 5,  left: 20,  bottom: 20,  right: 20);
        this.add(jTextArea, gbc);

        addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });

        this.pack();
        this.setLocationRelativeTo(null);
    }

    private void formWindowClosing() {  2 usages   Giorgos-Karachristos
        nameComboBox.setSelectedIndex(0);
        sqlTaxLabel.setText("");
        sqlColorLabel.setForeground(this.getBackground());
        sqlColorLabel.setBackground(this.getBackground());
        sqlIsActiveLabel.setText("");
        jTextArea.setText("");
    }

    private void nameComboBoxActionPerformed() {  1 usage   Giorgos-Karachristos
        for (Category category : categoryList) {
            if (category.getName().equals(nameComboBox.getSelectedItem())) {
                sqlTaxLabel.setText(category.getTax());
                sqlColorLabel.setForeground(Color.decode(category.getColor()));
                sqlColorLabel.setBackground(Color.decode(category.getColor()));
                sqlIsActiveLabel.setText(category.getIsActive());
                if (Integer.parseInt(category.getCountItems()) > 0) {
                    jTextArea.setText("Deleting this category will affect " + category.getCountItems() + " items."
                            + "\nTheir categories will be set to empty.");
                } else {
                    jTextArea.setText("");
                }
                this.pack();
                break;
            }
        }
    }
```

DeleteCategoryView.java (4/4)

```
    private void deleteButtonActionPerformed(DefaultTableModel tableModel) { 1 usage  Giorgos-Karachristos
        if (nameComboBox.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please select a category to delete",  title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        String result = categoryController.deleteCategory((String) nameComboBox.getSelectedItem());
        if (result.contains("Delete failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        categoryController.getCategoryTable(tableModel);
        populateComboBox();
        formWindowClosing();
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { categoryList = categoryController.getCategoryList(categoryBoxModel); }
}
```

InsertCategoryView.java (1/3)

```
package view.category;

import controller.category.CategoryController;
import util.CustomColorChooser;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class InsetCategoryView extends JDialog {  2 usages  Giorgos-Karachristos

    private final JTextField nameTextField;  5 usages
    private final JTextField taxTextField;  6 usages
    private final JButton colorButton;  8 usages
    private final ButtonGroup buttonGroup1;  5 usages
    private final JRadioButton noRadioButton;  7 usages
    private final JRadioButton yesRadioButton;  7 usages
    private final CategoryController categoryController;  3 usages

    public InsetCategoryView(DefaultTableModel tableModel) {  1 usage  Giorgo
        categoryController = new CategoryController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Add Category");
        this.setResizable(false);
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0};
        this.setLayout(layout);

        JLabel nameLabel = new JLabel();
        nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        nameLabel.setText("Name:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 20,  left: 20,  bottom: 5,  right: 5);
        this.add(nameLabel, gbc);
```

InsertCategoryView.java (2/3)

```
nameTextField = new JTextField();
nameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 0;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
this.add(nameTextField, gbc);

JLabel taxLabel = new JLabel();
taxLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
taxLabel.setText("Tax:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 0;
gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
this.add(taxLabel, gbc);

taxTextField = new JTextField();
taxTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 0;
gbc.gridwidth = 3;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 20);
this.add(taxTextField, gbc);

colorButton = new JButton();
colorButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
colorButton.setText("Color");
colorButton.addActionListener( ActionEvent evt1 -> colorButtonActionPerformed());
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 2;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(colorButton, gbc);

JLabel isActiveLabel = new JLabel();
isActiveLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
isActiveLabel.setText("Is active:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(isActiveLabel, gbc);

buttonGroup1 = new ButtonGroup();
yesRadioButton = new JRadioButton();
buttonGroup1.add(yesRadioButton);
yesRadioButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
yesRadioButton.setText("Yes");
yesRadioButton.setActionCommand("Yes");
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(yesRadioButton, gbc);
```

InsertCategoryView.java (3/3)

```
InsertCategory\View.java
        noRadioButton = new JRadioButton();
        buttonGroup1.add(noRadioButton);
        noRadioButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        noRadioButton.setText("No");
        noRadioButton.setActionCommand("No");
        gbc = new GridBagConstraints();
        gbc.gridx = 8;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
        this.add(noRadioButton, gbc);

        JButton submitButton = new JButton();
        submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        submitButton.setText("Submit");
        submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(tableModel));
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.gridwidth = 9;
        gbc.ipadx = 100;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 20,  right: 5);
        this.add(submitButton, gbc);

        addWindowListener((WindowAdapter) windowClosing(evt) → { formWindowClosing(); });

        this.pack();
        this.setLocationRelativeTo(null);
    }

    private void colorButtonActionPerformed() {  1 usage   Giorgos-Karachristos
        CustomColorChooser chooser = new CustomColorChooser(colorButton);
        chooser.setModal(true);
        chooser.setVisible(true);
    }

    private void formWindowClosing() {  2 usages   Giorgos-Karachristos
        nameTextField.setText("");
        taxTextField.setText("");
        colorButton.setBackground(Color.WHITE);
        buttonGroup1.clearSelection();
    }

    private void submitButtonActionPerformed(DefaultTableModel tableModel) {  1 usage   Giorgos-Karachristos *
        String name = nameTextField.getText();
        String taxText = taxTextField.getText().isBlank() ? "0" : taxTextField.getText();

        if (name.isBlank() || !noRadioButton.isSelected() && !yesRadioButton.isSelected()) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "The 'Name' and 'Is active' fields are mandatory.",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        double tax;
        try {
            tax = Double.parseDouble(taxText);
            if (tax < 0) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "Tax cannot be negative",
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                return;
            }
        } catch (NumberFormatException exception) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Tax must be number",
                    title: "Validation Error", JOptionPane.WARNING_MESSAGE);
            return;
        }

        Color color = colorButton.getBackground();
        String hexColor = String.format("#%02x%02x%02x", color.getRed(), color.getGreen(), color.getBlue());

        String result = categoryController.insertCategory(name, tax, hexColor,
                buttonGroup1.getSelection().getActionCommand());
        if (result.contains("Insertion failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        categoryController.getCategoryTable(tableModel);
        formWindowClosing();
    }
}
```

UpdateCategoryView.java (1/4)

```java
package view.category;

import controller.category.CategoryController;
import entities.Category;
import util.CustomColorChooser;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class UpdateCategoryView extends JDialog {

    private final JComboBox<String>nameComboBox;
    private final JTextField taxTextField;
    private final JButton colorButton;
    private final ButtonGroup buttonGroup1;
    private final JRadioButton noRadioButton;
    private final JRadioButton yesRadioButton;
    private final DefaultComboBoxModel<String> categoryBoxModel;
    private final CategoryController categoryController;

    private List<Category> categoryList;

    public UpdateCategoryView(DefaultTableModel tableModel) {
        categoryController = new CategoryController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Edit Category");
        this.setResizable(false);
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0};
        this.setLayout(layout);

        JLabel nameLabel = new JLabel();
        nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        nameLabel.setText("Name:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 20,  left: 20,  bottom: 5,  right: 5);
        this.add(nameLabel, gbc);

        nameComboBox = new JComboBox<>();
        nameComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        categoryBoxModel = (DefaultComboBoxModel<String>) nameComboBox.getModel();
        populateComboBox();
        nameComboBox.addActionListener( ActionEvent e -> nameComboBoxActionPerformed());
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.ipadx = 140;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
        getContentPane().add(nameComboBox, gbc);
```

UpdateCategoryView.java (2/4)

```
JLabel taxLabel = new JLabel();
taxLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
taxLabel.setText("Tax:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 0;
gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
this.add(taxLabel, gbc);

taxTextField = new JTextField();
taxTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 0;
gbc.gridwidth = 3;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 20);
this.add(taxTextField, gbc);

colorButton = new JButton();
colorButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
colorButton.setText("Color");
colorButton.addActionListener( ActionEvent evt1 -> colorButtonActionPerformed());
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 2;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(colorButton, gbc);

JLabel isActiveLabel = new JLabel();
isActiveLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
isActiveLabel.setText("Is active:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(isActiveLabel, gbc);

buttonGroup1 = new ButtonGroup();
yesRadioButton = new JRadioButton();
buttonGroup1.add(yesRadioButton);
yesRadioButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
yesRadioButton.setText("Yes");
yesRadioButton.setActionCommand("Yes");
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(yesRadioButton, gbc);

noRadioButton = new JRadioButton();
buttonGroup1.add(noRadioButton);
noRadioButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
noRadioButton.setText("No");
noRadioButton.setActionCommand("No");
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 2;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(noRadioButton, gbc);
```

UpdateCategoryView.java (3/4)

```
Update Category View.java
    JButton submitButton = new JButton();
    submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    submitButton.setText("Submit");
    submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(tableModel));
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 4;
    gbc.gridwidth = 9;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 20,  right: 5);
    this.add(submitButton, gbc);

    addWindowListener((WindowAdapter) windowClosing(evt) → { formWindowClosing(); });

    this.pack();
    this.setLocationRelativeTo(null);
}

private void colorButtonActionPerformed() {  1 usage  Giorgos-Karachristos
    CustomColorChooser chooser = new CustomColorChooser(colorButton);
    chooser.setModal(true);
    chooser.setVisible(true);
}

private void formWindowClosing() {  2 usages  Giorgos-Karachristos
    nameComboBox.setSelectedIndex(0);
    taxTextField.setText("");
    colorButton.setBackground(Color.WHITE);
    buttonGroup1.clearSelection();
}

private void nameComboBoxActionPerformed() {  1 usage  Giorgos-Karachristos
    for (Category category : categoryList) {
        if (category.getName().equals(nameComboBox.getSelectedItem())) {
            taxTextField.setText(category.getTax());
            colorButton.setBackground(Color.decode(category.getColor()));
            if (category.getIsActive().equals("Yes")) {
                yesRadioButton.setSelected(true);
            } else {
                noRadioButton.setSelected(true);
            }
            break;
        }
    }
}
```

UpdateCategoryView.java (4/4)

```
Update Category \ View.java
    private void submitButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
        if (nameComboBox.getSelectedIndex() == 0 || !noRadioButton.isSelected() && !yesRadioButton.isSelected()) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please select a category to edit",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        double tax;
        String taxText = taxTextField.getText().isBlank() ? "0" : taxTextField.getText();
        try {
            tax = Double.parseDouble(taxText);
            if (tax < 0) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: "Tax cannot be negative",
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                return;
            }
        } catch (NumberFormatException exception) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Tax must be number",
                    title: "Validation Error", JOptionPane.WARNING_MESSAGE);
            return;
        }

        Color color = colorButton.getBackground();
        String hexColor = String.format("#%02x%02x%02x", color.getRed(), color.getGreen(), color.getBlue());

        String result = categoryController.updateCategory((String) nameComboBox.getSelectedItem(),
                tax, hexColor, buttonGroup1.getSelection().getActionCommand());
        if (result.contains("Update failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        categoryController.getCategoryTable(tableModel);
        populateComboBox();
        formWindowClosing();
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { categoryList = categoryController.getCategoryList(categoryBoxModel); }
}
```

A.8.3 Employee

DeleteEmployeeView.java (1/6)

DeleteEmployeeView.java

```java
package view.employee;

import controller.employee.EmployeeController;
import entities.Employee;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class DeleteEmployeeView extends JDialog {  2 usages  ♙ Giorgos-Karachristos *

    private final JComboBox<String> employeeComboBox;  9 usages
    private final JLabel sqlNameLabel;  5 usages
    private final JLabel sqlSurnameLabel;  5 usages
    private final JLabel sqlPhoneLabel;  5 usages
    private final JLabel sqlEmailLabel;  5 usages
    private final JLabel sqlRoleLabel;  5 usages
    private final JLabel sqlHireDateLabel;  5 usages
    private final JLabel sqlSsnLabel;  5 usages
    private final JLabel sqlIdnLabel;  5 usages
    private final JLabel sqlUsernameLabel;  5 usages
    private final JLabel sqlPasswordLabel;  5 usages
    private final DefaultComboBoxModel<String> roleBoxModel;  2 usages
    private final EmployeeController employeeController;  4 usages

    private List<Employee> employeeList;  2 usages

    public DeleteEmployeeView(DefaultTableModel tableModel) {  1 usage  ♙ Giorgos-Karachristos
        employeeController = new EmployeeController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setResizable(false);
        this.setTitle("Delete Employee");
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0, 10, 0};
        this.setLayout(layout);

        JLabel employeeLabel = new JLabel();
        employeeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        employeeLabel.setText("Employee ID:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 20,  left: 20,  bottom: 5,  right: 5);
        this.add(employeeLabel, gbc);

        employeeComboBox = new JComboBox<>();
        employeeComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        roleBoxModel = (DefaultComboBoxModel<String>) employeeComboBox.getModel();
        populateComboBox();
        employeeComboBox.addActionListener( ActionEvent e -> employeeComboBoxActionPerformed());
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.gridwidth = 3;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 100;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
        this.add(employeeComboBox, gbc);
```

DeleteEmployeeView.java (2/6)

DeleteEmployeeView.java

```
JButton deleteButton = new JButton();
deleteButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
deleteButton.setText("Delete");
deleteButton.addActionListener((ActionEvent evt) -> deleteButtonActionPerformed(tableModel));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 0;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 20);
this.add(deleteButton, gbc);

JLabel nameLabel = new JLabel();
nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
nameLabel.setText("Name:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(nameLabel, gbc);

sqlNameLabel = new JLabel();
sqlNameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlNameLabel, gbc);

JLabel surnameLabel = new JLabel();
surnameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
surnameLabel.setText("Surname:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 2;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(surnameLabel, gbc);

sqlSurnameLabel = new JLabel();
sqlSurnameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlSurnameLabel, gbc);

JLabel phoneLabel = new JLabel();
phoneLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
phoneLabel.setText("Phone:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(phoneLabel, gbc);
```

DeleteEmployeeView.java (3/6)

```
DeleteEmployeeView.java
sqlPhoneLabel = new JLabel();
sqlPhoneLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlPhoneLabel, gbc);

JLabel emailLabel = new JLabel();
emailLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
emailLabel.setText("Email:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(emailLabel, gbc);

sqlEmailLabel = new JLabel();
sqlEmailLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlEmailLabel, gbc);

JLabel roleLabel = new JLabel();
roleLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
roleLabel.setText("Role:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 6;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(roleLabel, gbc);

sqlRoleLabel = new JLabel();
sqlRoleLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 6;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlRoleLabel, gbc);

JLabel hireDateLabel = new JLabel();
hireDateLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
hireDateLabel.setText("Hire Date:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 6;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(hireDateLabel, gbc);

sqlHireDateLabel = new JLabel();
sqlHireDateLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 6;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlHireDateLabel, gbc);
```

DeleteEmployeeView.java (4/6)

```
DeleteEmployeeView.java
JLabel ssnLabel = new JLabel();
ssnLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
ssnLabel.setText("Social Security Number:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 8;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(ssnLabel, gbc);

sqlSsnLabel = new JLabel();
sqlSsnLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 8;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlSsnLabel, gbc);

JLabel idnLabel = new JLabel();
idnLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
idnLabel.setText("Identification Number:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 8;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(idnLabel, gbc);

sqlIdnLabel = new JLabel();
sqlIdnLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 8;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlIdnLabel, gbc);

JLabel usernameLabel = new JLabel();
usernameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
usernameLabel.setText("Username:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 10;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 20,  right: 5);
this.add(usernameLabel, gbc);

sqlUsernameLabel = new JLabel();
sqlUsernameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 20,  right: 5);
this.add(sqlUsernameLabel, gbc);
```

DeleteEmployeeView.java (5/6)

```
DeleteEmployeeView.java
    JLabel passwordLabel = new JLabel();
    passwordLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN, size: 16));
    passwordLabel.setText("Password:");
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 10;
    gbc.anchor = GridBagConstraints.LINE_START;
    gbc.insets = new Insets( top: 5, left: 5, bottom: 20, right: 5);
    this.add(passwordLabel, gbc);

    sqlPasswordLabel = new JLabel();
    sqlPasswordLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN, size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 6;
    gbc.gridy = 10;
    gbc.insets = new Insets( top: 5, left: 5, bottom: 20, right: 20);
    this.add(sqlPasswordLabel, gbc);

    addWindowListener(new WindowAdapter() {  Giorgos-Karachristos
        @Override  Giorgos-Karachristos
        public void windowClosing(WindowEvent evt) {
            formWindowClosing();
        }
    });

    this.pack();
    this.setLocationRelativeTo(null);
}

private void formWindowClosing() {  2 usages  Giorgos-Karachristos
    employeeComboBox.setSelectedIndex(0);
    sqlNameLabel.setText("");
    sqlSurnameLabel.setText("");
    sqlPhoneLabel.setText("");
    sqlEmailLabel.setText("");
    sqlRoleLabel.setText("");
    sqlHireDateLabel.setText("");
    sqlSsnLabel.setText("");
    sqlIdnLabel.setText("");
    sqlUsernameLabel.setText("");
    sqlPasswordLabel.setText("");
}

private void employeeComboBoxActionPerformed() {  1 usage  Giorgos-Karachristos
    for (Employee employee : employeeList) {
        if (employee.getEmployeeID().equals(employeeComboBox.getSelectedItem())) {
            sqlNameLabel.setText(employee.getName());
            sqlSurnameLabel.setText(employee.getSurname());
            sqlPhoneLabel.setText(employee.getPhone());
            sqlEmailLabel.setText(employee.getEmail());
            sqlRoleLabel.setText(employee.getRole());
            sqlSsnLabel.setText(employee.getSsn());
            sqlIdnLabel.setText(employee.getId());
            sqlHireDateLabel.setText(employee.getHireDate());
            sqlUsernameLabel.setText(employee.getUsername());
            sqlPasswordLabel.setText(employee.getPassword());
            //repaint
            this.pack();
            break;
        }
    }
}
```

DeleteEmployeeView.java (6/6)

```
DeleteEmployeeView.java
    private void deleteButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
        if (employeeComboBox.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please select an employee to delete",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        String result = employeeController.deleteEmployee(Integer.parseInt((String) employeeComboBox.getSelectedItem()));
        if (result.contains("Delete failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        employeeController.getEmployeeTable(tableModel);
        populateComboBox();
        formWindowClosing();
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() {  3 usages  Giorgos-Karachristos
        employeeList = employeeController.getEmployeeList(roleBoxModel);
    }
}
```

InsertEmployeeView.java (1/5)

```
InsertEmployee\View.java
package view.employee;

import controller.employee.EmployeeController;
import controller.role.RoleController;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class InsertEmployeeView extends JDialog {  2 usages  Giorgos-Karachristos

    private final JTextField nameTextField;  5 usages
    private final JTextField surnameTextField;  5 usages
    private final JTextField phoneTextField;  5 usages
    private final JTextField emailTextField;  5 usages
    private final JTextField ssnTextField;  5 usages
    private final JComboBox<String> roleComboBox;  7 usages
    private final JTextField idnTextField;  5 usages
    private final JTextField usernameTextField;  5 usages
    private final JTextField passwordTextField;  5 usages
    private final DefaultComboBoxModel<String> roleBoxModel;  2 usages
    private final EmployeeController employeeController;  3 usages
    private final RoleController roleController;  2 usages

    public InsertEmployeeView(DefaultTableModel tableModel) {  1 usage  Giorgos-K
        employeeController = new EmployeeController();
        roleController = new RoleController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Add Employee");
        this.setResizable(false);
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0, 10, 0};
        this.setLayout(layout);
```

InsertEmployeeView.java (2/5)

```
JLabel nameLabel = new JLabel();
nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
nameLabel.setText("Name:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 0;
gbc.insets = new Insets( top: 20,  left: 20,  bottom: 0,  right: 0);
this.add(nameLabel, gbc);

nameTextField = new JTextField();
nameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 0;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 20,  left: 0,  bottom: 0,  right: 0);
this.add(nameTextField, gbc);

JLabel surnameLabel = new JLabel();
surnameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
surnameLabel.setText("Surname:");
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 0;
gbc.insets = new Insets( top: 20,  left: 0,  bottom: 0,  right: 0);
this.add(surnameLabel, gbc);

surnameTextField = new JTextField();
surnameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 10;
gbc.gridy = 0;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 20,  left: 0,  bottom: 0,  right: 20);
this.add(surnameTextField, gbc);

JLabel phoneLabel = new JLabel();
phoneLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
phoneLabel.setText("Phone:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.insets = new Insets( top: 0,  left: 20,  bottom: 0,  right: 0);
this.add(phoneLabel, gbc);

phoneTextField = new JTextField();
phoneTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 2;
gbc.ipadx = 150;
gbc.ipady = 10;
this.add(phoneTextField, gbc);

JLabel emailLabel = new JLabel();
emailLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
emailLabel.setText("Email:");
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 2;
this.add(emailLabel, gbc);
```

InsertEmployeeView.java (3/5)

```
emailTextField = new JTextField();
emailTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 10;
gbc.gridy = 2;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 0,  left: 0,  bottom: 0,  right: 20);
this.add(emailTextField, gbc);

JLabel roleLabel = new JLabel();
roleLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
roleLabel.setText("Role:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 4;
this.add(roleLabel, gbc);

roleComboBox = new JComboBox<>();
roleComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
roleBoxModel = (DefaultComboBoxModel<String>) roleComboBox.getModel();
populateComboBox();
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 4;
gbc.ipadx = 100;
gbc.ipady = 10;
this.add(roleComboBox, gbc);

JLabel ssnLabel = new JLabel();
ssnLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
ssnLabel.setText("Social Security Number:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 6;
gbc.insets = new Insets( top: 0,  left: 20,  bottom: 0,  right: 0);
this.add(ssnLabel, gbc);

ssnTextField = new JTextField();
ssnTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 6;
gbc.ipadx = 150;
gbc.ipady = 10;
this.add(ssnTextField, gbc);

JLabel idnLabel = new JLabel();
idnLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
idnLabel.setText("Identification Number:");
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 6;
this.add(idnLabel, gbc);

idnTextField = new JTextField();
idnTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 10;
gbc.gridy = 6;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 0,  left: 0,  bottom: 0,  right: 20);
this.add(idnTextField, gbc);
```

InsertEmployeeView.java (4/5)

```
InsertEmployeeView.java
    JLabel usernameLabel = new JLabel();
    usernameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    usernameLabel.setText("Username:");
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 8;
    gbc.insets = new Insets( top: 0,  left: 20,  bottom: 0,  right: 0);
    this.add(usernameLabel, gbc);

    usernameTextField = new JTextField();
    usernameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 2;
    gbc.gridy = 8;
    gbc.ipadx = 150;
    gbc.ipady = 10;
    this.add(usernameTextField, gbc);

    JLabel passwordLabel = new JLabel();
    passwordLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    passwordLabel.setText("Password:");
    gbc = new GridBagConstraints();
    gbc.gridx = 8;
    gbc.gridy = 8;
    this.add(passwordLabel, gbc);

    passwordTextField = new JTextField();
    passwordTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 10;
    gbc.gridy = 8;
    gbc.ipadx = 150;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 0,  left: 0,  bottom: 0,  right: 20);
    this.add(passwordTextField, gbc);

    JButton submitButton = new JButton();
    submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    submitButton.setText("Submit");
    submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(tableModel));
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 10;
    gbc.gridwidth = 3;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 0,  left: 0,  bottom: 20,  right: 0);
    this.add(submitButton, gbc);

    addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });
    this.pack();
    this.setLocationRelativeTo(null);
}
```

InsertEmployeeView.java (5/5)

```
InsertEmployeeView.java
    private void formWindowClosing() {  2 usages  Giorgos-Karachristos
        nameTextField.setText("");
        surnameTextField.setText("");
        phoneTextField.setText("");
        emailTextField.setText("");
        roleComboBox.setSelectedIndex(0);
        ssnTextField.setText("");
        idnTextField.setText("");
        usernameTextField.setText("");
        passwordTextField.setText("");
    }

    private void submitButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
        String name = nameTextField.getText().trim();
        String surname = surnameTextField.getText().trim();
        String phone = phoneTextField.getText().trim();
        String email = emailTextField.getText().trim();
        String ssn = ssnTextField.getText().trim();
        String idn = idnTextField.getText().trim();
        String username = usernameTextField.getText().trim();
        String password = passwordTextField.getText().trim();

        if (name.isBlank() || surname.isBlank() || phone.isBlank() || email.isBlank() || roleComboBox.getSelectedIndex() == 0
                || ssn.isBlank() || idn.isBlank() || username.isBlank() || password.isBlank()) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "All the fields are required",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        String result = employeeController.insertEmployee(name, surname, phone, email, (String) roleComboBox.getSelectedItem(),
                ssn, idn, username, password);
        if (result.contains("Insertion failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        employeeController.getEmployeeTable(tableModel);
        formWindowClosing();
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { roleController.getRoleName(roleBoxModel); }
}
```

UpdateEmployeeView.java (1/5)

UpdateEmployee\view.java

```java
package view.employee;

import controller.employee.EmployeeController;
import controller.role.RoleController;
import entities.Employee;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class UpdateEmployeeView extends JDialog {  2 usages  Giorgos-Karachristos

    private final JTextField nameTextField;  6 usages
    private final JTextField surnameTextField;  6 usages
    private final JTextField phoneTextField;  6 usages
    private final JTextField emailTextField;  6 usages
    private final JTextField ssnTextField;  6 usages
    private final JComboBox<String> roleComboBox;  10 usages
    private final JComboBox<String> employeeComboBox;  9 usages
    private final JTextField idnTextField;  6 usages
    private final JTextField usernameTextField;  6 usages
    private final JTextField passwordTextField;  6 usages
    private final DefaultComboBoxModel<String> employeeBoxModel;  2 usages
    private final DefaultComboBoxModel<String> roleBoxModel;  2 usages
    private final EmployeeController employeeController;  4 usages
    private final RoleController roleController;  2 usages

    private List<Employee> employeeList;  2 usages

    public UpdateEmployeeView(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos
        employeeController = new EmployeeController();
        roleController = new RoleController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Edit Employee");
        this.setResizable(false);
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0, 10, 0, 10, 0};
        this.setLayout(layout);

        JLabel employeeLabel = new JLabel();
        employeeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        employeeLabel.setText("Employee ID:");
        gbc = new GridBagConstraints();
        gbc.gridx = 4;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 20,  left: 0,  bottom: 0,  right: 0);
        this.add(employeeLabel, gbc);

        employeeComboBox = new JComboBox<>();
        employeeComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        employeeBoxModel = (DefaultComboBoxModel<String>) employeeComboBox.getModel();
        gbc = new GridBagConstraints();
        gbc.gridx = 6;
        gbc.gridy = 0;
        gbc.ipadx = 100;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 20,  left: 0,  bottom: 0,  right: 0);
        this.add(employeeComboBox, gbc);
```

UpdateEmployeeView.java (2/5)

```
UpdateEmployeeView.java
JLabel nameLabel = new JLabel();
nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
nameLabel.setText("Name:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.insets = new Insets( top: 0,  left: 20,  bottom: 0,  right: 0);
this.add(nameLabel, gbc);

nameTextField = new JTextField();
nameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 2;
gbc.ipadx = 150;
gbc.ipady = 10;
this.add(nameTextField, gbc);

JLabel surnameLabel = new JLabel();
surnameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
surnameLabel.setText("Surname:");
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 2;
this.add(surnameLabel, gbc);

surnameTextField = new JTextField();
surnameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 10;
gbc.gridy = 2;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 0,  left: 0,  bottom: 0,  right: 20);
this.add(surnameTextField, gbc);

JLabel phoneLabel = new JLabel();
phoneLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
phoneLabel.setText("Phone:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.insets = new Insets( top: 0,  left: 20,  bottom: 0,  right: 0);
this.add(phoneLabel, gbc);

phoneTextField = new JTextField();
phoneTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 4;
gbc.ipadx = 150;
gbc.ipady = 10;
this.add(phoneTextField, gbc);

JLabel emailLabel = new JLabel();
emailLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
emailLabel.setText("Email:");
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 4;
this.add(emailLabel, gbc);
```

UpdateEmployeeView.java (3/5)

```
UpdateEmployeeView.java
emailTextField = new JTextField();
emailTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 10;
gbc.gridy = 4;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 0,  left: 0,  bottom: 0,  right: 20);
this.add(emailTextField, gbc);

JLabel roleLabel = new JLabel();
roleLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
roleLabel.setText("Role:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 6;
this.add(roleLabel, gbc);

roleComboBox = new JComboBox<>();
roleComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
roleBoxModel = (DefaultComboBoxModel<String>) roleComboBox.getModel();
populateComboBoxes();
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 6;
gbc.ipadx = 100;
gbc.ipady = 10;
this.add(roleComboBox, gbc);

employeeComboBox.addActionListener( ActionEvent e -> employeeComboBoxActionPerformed());

JLabel ssnLabel = new JLabel();
ssnLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
ssnLabel.setText("Social Security Number:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 8;
gbc.insets = new Insets( top: 0,  left: 20,  bottom: 0,  right: 0);
this.add(ssnLabel, gbc);

ssnTextField = new JTextField();
ssnTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 8;
gbc.ipadx = 150;
gbc.ipady = 10;
this.add(ssnTextField, gbc);

JLabel idnLabel = new JLabel();
idnLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
idnLabel.setText("Identification Number:");
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 8;
this.add(idnLabel, gbc);

idnTextField = new JTextField();
idnTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 10;
gbc.gridy = 8;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 0,  left: 0,  bottom: 0,  right: 20);
this.add(idnTextField, gbc);
```

UpdateEmployeeView.java (4/5)

```
UpdateEmployeeView.java
    JLabel usernameLabel = new JLabel();
    usernameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    usernameLabel.setText("Username:");
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 10;
    gbc.insets = new Insets( top: 0,  left: 20,  bottom: 0,  right: 0);
    this.add(usernameLabel, gbc);

    usernameTextField = new JTextField();
    usernameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 2;
    gbc.gridy = 10;
    gbc.ipadx = 150;
    gbc.ipady = 10;
    this.add(usernameTextField, gbc);

    JLabel passwordLabel = new JLabel();
    passwordLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    passwordLabel.setText("Password:");
    gbc = new GridBagConstraints();
    gbc.gridx = 8;
    gbc.gridy = 10;
    this.add(passwordLabel, gbc);

    passwordTextField = new JTextField();
    passwordTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 10;
    gbc.gridy = 10;
    gbc.ipadx = 150;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 0,  left: 0,  bottom: 0,  right: 20);
    this.add(passwordTextField, gbc);

    JButton submitButton = new JButton();
    submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    submitButton.setText("Submit");
    submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(tableModel));
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 12;
    gbc.gridwidth = 3;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 0,  left: 0,  bottom: 20,  right: 0);
    this.add(submitButton, gbc);

    addWindowListener((WindowAdapter) windowClosing(evt) → { formWindowClosing(); });

    this.pack();
    this.setLocationRelativeTo(null);
}

private void formWindowClosing() {  2 usages   Giorgos-Karachristos
    employeeComboBox.setSelectedIndex(0);
    nameTextField.setText("");
    surnameTextField.setText("");
    phoneTextField.setText("");
    emailTextField.setText("");
    roleComboBox.setSelectedIndex(0);
    ssnTextField.setText("");
    idnTextField.setText("");
    usernameTextField.setText("");
    passwordTextField.setText("");
}
```

UpdateEmployeeView.java (5/5)

```
UpdateEmployeeView.java
    private void employeeComboBoxActionPerformed() {  1 usage  Giorgos-Karachristos
        for (Employee employee : employeeList) {
            if (employee.getEmployeeID().equals(employeeComboBox.getSelectedItem())) {
                nameTextField.setText(employee.getName());
                surnameTextField.setText(employee.getSurname());
                phoneTextField.setText(employee.getPhone());
                emailTextField.setText(employee.getEmail());
                String role = employee.getRole();
                for (int x = 0; x <= roleComboBox.getItemCount(); x++) {
                    if (role.equals(roleComboBox.getItemAt(x))) {
                        roleComboBox.setSelectedIndex(x);
                        break;
                    }
                }
                ssnTextField.setText(employee.getSsn());
                idnTextField.setText(employee.getId());
                //8 date
                usernameTextField.setText(employee.getUsername());
                passwordTextField.setText(employee.getPassword());
                break;
            }
        }
    }

    private void submitButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
        String name = nameTextField.getText().trim();
        String surname = surnameTextField.getText().trim();
        String phone = phoneTextField.getText().trim();
        String email = emailTextField.getText().trim();
        String ssn = ssnTextField.getText().trim();
        String idn = idnTextField.getText().trim();
        String username = usernameTextField.getText().trim();
        String password = passwordTextField.getText().trim();

        if (employeeComboBox.getSelectedIndex() == 0 || name.isBlank() || surname.isBlank() || phone.isBlank()
                || email.isBlank() || roleComboBox.getSelectedIndex() == 0 || ssn.isBlank() || idn.isBlank()
                || username.isBlank() || password.isBlank()) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "All the fields are required",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        String result = employeeController.updateEmployee(Integer.parseInt((String) employeeComboBox.getSelectedItem()),
                name, surname, phone, email, (String) roleComboBox.getSelectedItem(), ssn, idn, username, password);
        if (result.contains("Update failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        employeeController.getEmployeeTable(tableModel);
        populateComboBoxes();
        formWindowClosing();
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBoxes();
        }
        super.setVisible(visible);
    }

    private void populateComboBoxes() {  3 usages  Giorgos-Karachristos
        employeeList = employeeController.getEmployeeList(employeeBoxModel);
        roleController.getRoleName(roleBoxModel);
    }
}
```

A.8.4 Inventory

DeleteInventoryView.java (1/4)

DeleteInventory\view.java

```
package view.inventory;

import controller.inventory.InventoryController;
import entities.InventoryItem;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class DeleteInventoryView extends JDialog {  2 usages  Giorgos-Karachristos

    private final JComboBox<String>itemComboBox;  9 usages
    private final JLabel sqlCodeLabel;  5 usages
    private final JLabel sqlNameLabel;  5 usages
    private final JLabel sqlLocationLabel;  5 usages
    private final JLabel sqlQuantityLabel;  5 usages
    private final JLabel sqlMinQuantityLabel;  5 usages
    private final JLabel sqlMaxQuantityLabel;  5 usages
    private final JLabel sqlUpdateLabel;  5 usages
    private final DefaultComboBoxModel<String> itemBoxModel;  2 usages
    private final InventoryController inventoryController;  4 usages

    private List<InventoryItem> rowData;  2 usages

    public DeleteInventoryView(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos
        inventoryController = new InventoryController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setResizable(false);
        this.setTitle("Delete Inventory");
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0};
        this.setLayout(layout);

        JLabel itemLabel = new JLabel();
        itemLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        itemLabel.setText("Item ID:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.LINE_END;
        gbc.insets = new Insets( top: 20,  left: 20,  bottom: 5,  right: 5);
        this.add(itemLabel, gbc);

        itemComboBox = new JComboBox<>();
        itemComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        itemBoxModel = (DefaultComboBoxModel<String>) itemComboBox.getModel();
        populateComboBox();
        itemComboBox.addActionListener( ActionEvent evt1 -> itemComboBoxActionPerformed());
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.ipadx = 100;
        gbc.ipady = 10;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
        this.add(itemComboBox, gbc);
```

DeleteInventoryView.java (2/4)

```
DeleteInventoryView.java
JButton deleteButton = new JButton();
deleteButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
deleteButton.setText("Delete");
deleteButton.addActionListener((ActionEvent evt) -> deleteButtonActionPerformed(tableModel));
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 0;
gbc.gridwidth = 3;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 20);
this.add(deleteButton, gbc);

JLabel codeLabel = new JLabel();
codeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
codeLabel.setText("Code:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(codeLabel, gbc);

sqlCodeLabel = new JLabel();
sqlCodeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlCodeLabel, gbc);

JLabel nameLabel = new JLabel();
nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
nameLabel.setText("Name:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(nameLabel, gbc);

sqlNameLabel = new JLabel();
sqlNameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlNameLabel, gbc);

JLabel locationLabel = new JLabel();
locationLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
locationLabel.setText("Location:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(locationLabel, gbc);
```

DeleteInventoryView.java (3/4)

```
DeleteInventoryView.java
sqlLocationLabel = new JLabel();
sqlLocationLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlLocationLabel, gbc);

JLabel quantityLabel = new JLabel();
quantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
quantityLabel.setText("Quantity:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(quantityLabel, gbc);

sqlQuantityLabel = new JLabel();
sqlQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlQuantityLabel, gbc);

JLabel minQuantityLabel = new JLabel();
minQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
minQuantityLabel.setText("Min Quantity:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 6;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(minQuantityLabel, gbc);

sqlMinQuantityLabel = new JLabel();
sqlMinQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 6;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlMinQuantityLabel, gbc);

JLabel maxQuantityLabel = new JLabel();
maxQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
maxQuantityLabel.setText("Max Quantity:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 6;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(maxQuantityLabel, gbc);

sqlMaxQuantityLabel = new JLabel();
sqlMaxQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 6;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlMaxQuantityLabel, gbc);

JLabel updateLabel = new JLabel();
updateLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
updateLabel.setText("Last Update at:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 8;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 20,  right: 5);
this.add(updateLabel, gbc);
```

DeleteInventoryView.java (4/4)

```
DeleteInventoryView.java
        sqlUpdateLabel = new JLabel();
        sqlUpdateLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 8;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 20,  right: 5);
        this.add(sqlUpdateLabel, gbc);

        addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });

        this.pack();
        this.setLocationRelativeTo(null);
    }

    private void formWindowClosing() {  2 usages   Giorgos-Karachristos
        itemComboBox.setSelectedIndex(0);
        sqlCodeLabel.setText("");
        sqlNameLabel.setText("");
        sqlLocationLabel.setText("");
        sqlQuantityLabel.setText("");
        sqlMinQuantityLabel.setText("");
        sqlMaxQuantityLabel.setText("");
        sqlUpdateLabel.setText("");
    }

    private void itemComboBoxActionPerformed() {  1 usage   Giorgos-Karachristos
        for (InventoryItem  inventoryItem : rowData) {
            if (inventoryItem.getItemId().equals(itemComboBox.getSelectedItem())) {
                sqlCodeLabel.setText(inventoryItem.getItemCode());
                sqlNameLabel.setText(inventoryItem.getName());
                sqlLocationLabel.setText(inventoryItem.getLocation());
                sqlQuantityLabel.setText(inventoryItem.getQuantity());
                sqlMinQuantityLabel.setText(inventoryItem.getMinQuantity());
                sqlMaxQuantityLabel.setText(inventoryItem.getMaxQuantity());
                sqlUpdateLabel.setText(inventoryItem.getLastUpdated());
                this.pack();
                break;
            }
        }
    }

    private void deleteButtonActionPerformed(DefaultTableModel tableModel) {  1 usage   Giorgos-Karachristos *
        if (itemComboBox.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please select an item to delete",  title: "Info",
                    JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        String result = inventoryController.deleteInventory(Integer.parseInt((String) itemComboBox.getSelectedItem()));
        if (result.contains("Delete failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        inventoryController.getInventoryTable(tableModel);
        populateComboBox();
        formWindowClosing();
    }

    @Override   Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { rowData = inventoryController.getInventoryList(itemBoxModel); }
}
```

InsertInventoryView.java (1/5)

```java
package view.inventory;

import controller.inventory.InventoryController;
import entities.Item;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class InsertInventoryView extends JDialog {  2 usages  & Giorgos-Karachristos *

    private final JComboBox<String>itemComboBox;  9 usages
    private final JLabel sqlCodeLabel;  5 usages
    private final JLabel sqlNameLabel;  5 usages
    private final JTextField locationTextField;  5 usages
    private final JTextField quantityTextField;  5 usages
    private final JTextField minQuantityTextField;  5 usages
    private final JTextField maxQuantityTextField;  5 usages
    private final DefaultComboBoxModel<String> itemBoxModel;  2 usages
    private final InventoryController inventoryController;  4 usages

    private List<Item> itemList;  2 usages

    public InsertInventoryView(DefaultTableModel tableModel) {  1 usage  & Giorgos-Karachristo
        inventoryController = new InventoryController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Add Inventory");
        this.setResizable(false);
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0};
        this.setLayout(layout);

        JLabel itemLabel = new JLabel();
        itemLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        itemLabel.setText("Item ID:");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.LINE_END;
        gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
        this.add(itemLabel, gbc);

        itemComboBox = new JComboBox<>();
        itemComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        itemBoxModel = (DefaultComboBoxModel<String>) itemComboBox.getModel();
        populateComboBox();
        itemComboBox.addActionListener( ActionEvent e -> itemComboBoxActionPerformed());
        gbc = new GridBagConstraints();
        gbc.gridx = 4;
        gbc.gridy = 0;
        gbc.ipadx = 100;
        gbc.ipady = 10;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
        this.add(itemComboBox, gbc);

        JLabel codeLabel = new JLabel();
        codeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        codeLabel.setText("Code:");
        gbc = new GridBagConstraints();
        gbc.gridx = 4;
        gbc.gridy = 2;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
        this.add(codeLabel, gbc);
```

InsertInventoryView.java (2/5)

```
sqlCodeLabel = new JLabel();
sqlCodeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlCodeLabel, gbc);

JLabel nameLabel = new JLabel();
nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
nameLabel.setText("Name:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(nameLabel, gbc);

sqlNameLabel = new JLabel();
sqlNameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlNameLabel, gbc);

JLabel locationLabel = new JLabel();
locationLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
locationLabel.setText("Location:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(locationLabel, gbc);

locationTextField = new JTextField();
locationTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 4;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(locationTextField, gbc);

JLabel quantityLabel = new JLabel();
quantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
quantityLabel.setText("Quantity:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(quantityLabel, gbc);

quantityTextField = new JTextField();
quantityTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 4;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(quantityTextField, gbc);
```

InsertInventoryView.java (3/5)

```
InsertInventoryView.java
    JLabel minQuantityLabel = new JLabel();
    minQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    minQuantityLabel.setText("Min Quantity:");
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 6;
    gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
    this.add(minQuantityLabel, gbc);

    minQuantityTextField = new JTextField();
    minQuantityTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 2;
    gbc.gridy = 6;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    this.add(minQuantityTextField, gbc);

    JLabel maxQuantityLabel = new JLabel();
    maxQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    maxQuantityLabel.setText("Max Quantity:");
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 6;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    this.add(maxQuantityLabel, gbc);

    maxQuantityTextField = new JTextField();
    maxQuantityTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 6;
    gbc.gridy = 6;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
    this.add(maxQuantityTextField, gbc);

    JButton submitButton = new JButton();
    submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    submitButton.setText("Submit");
    submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(tableModel));
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 8;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 20,  right: 5);
    this.add(submitButton, gbc);

    this.addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });
    this.pack();
    this.setLocationRelativeTo(null);
}
```

InsertInventoryView.java (4/5)

```java
private void formWindowClosing() {  2 usages  Giorgos-Karachristos
    itemComboBox.setSelectedIndex(0);
    sqlCodeLabel.setText("");
    sqlNameLabel.setText("");
    locationTextField.setText("");
    quantityTextField.setText("");
    minQuantityTextField.setText("");
    maxQuantityTextField.setText("");
}

private void itemComboBoxActionPerformed() {  1 usage  Giorgos-Karachristos
    for (Item item : itemList) {
        if (item.getItemId().equals(itemComboBox.getSelectedItem())) {
            sqlCodeLabel.setText(item.getItemCode());
            sqlNameLabel.setText(item.getName());
            this.pack();
            break;
        }
    }
}

private void submitButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
    String location = locationTextField.getText();
    String quantityText = quantityTextField.getText();
    String minQuantityText = minQuantityTextField.getText();
    String maxQuantityText = maxQuantityTextField.getText();

    if (itemComboBox.getSelectedIndex() == 0 || location.isBlank() || quantityText.isBlank()
            || minQuantityText.isBlank() || maxQuantityText.isBlank()) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "All the fields are required",
                title: "Info", JOptionPane.INFORMATION_MESSAGE);
        return;
    }

    if (!isValid(quantityText,  type: "Quantity") || !isValid(minQuantityText,  type: "Min Quantity")
            || !isValid(maxQuantityText,  type: "Max Quantity")) {
        return;
    }
    int quantity = Integer.parseInt(quantityText);
    int minQuantity = Integer.parseInt(minQuantityText);
    int maxQuantity = Integer.parseInt(maxQuantityText);

    if (minQuantity > maxQuantity) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Min quantity cannot be greater than max quantity",
                title: "Validation Error", JOptionPane.WARNING_MESSAGE);
        return;
    }

    if (quantity > maxQuantity) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Quantity can not pass the maximum quantity",
                title: "Validation Error", JOptionPane.WARNING_MESSAGE);
        return;
    }

    String result = inventoryController.insertInventory(Integer.parseInt((String) itemComboBox.getSelectedItem()),
            location, quantity, minQuantity, maxQuantity);
    if (result.contains("Insertion failed: ")) {
        JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
    } else {
        JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
    }

    inventoryController.getInventoryTable(tableModel);
    populateComboBox();
    formWindowClosing();
}
```

InsertInventoryView.java (5/5)

```
InsertInventoryView.java
    private boolean isValid(String price, String type) {  3 usages  Giorgos-Karachristos *
        try {
            int number = Integer.parseInt(price);
            if (number < 0) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: type + " cannot be negative",
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                return false;
            }
        } catch (NumberFormatException exception) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: type + " must be number",
                    title: "Validation Error", JOptionPane.WARNING_MESSAGE);
            return false;
        }
        return true;
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { itemList = inventoryController.getInventoryComboBox(itemBoxModel); }
}
```

UpdateInventoryView.java (1/5)

```
UpdateInventory\View.java
package view.inventory;

import controller.inventory.InventoryController;
import entities.InventoryItem;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class UpdateInventoryView extends JDialog {  2 usages  Giorgos-Karach

    private final JComboBox<String>itemComboBox;  9 usages
    private final JLabel sqlCodeLabel;  5 usages
    private final JLabel sqlNameLabel;  5 usages
    private final JTextField locationTextField;  6 usages
    private final JTextField quantityTextField;  6 usages
    private final JTextField minQuantityTextField;  6 usages
    private final JTextField maxQuantityTextField;  6 usages
    private final DefaultComboBoxModel<String> itemBoxModel;  2 usages
    private final InventoryController inventoryController;  4 usages

    private List<InventoryItem> inventoryItemList;  2 usages

    public UpdateInventoryView(DefaultTableModel tableModel) {  1 usage
        inventoryController = new InventoryController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Edit Inventory");
        this.setResizable(false);
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0};
        this.setLayout(layout);
```

UpdateInventoryView.java (2/5)

```
JLabel itemLabel = new JLabel();
itemLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
itemLabel.setText("Item ID:");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 0;
gbc.anchor = GridBagConstraints.LINE_END;
gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
this.add(itemLabel, gbc);

itemComboBox = new JComboBox<>();
itemComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
itemBoxModel = (DefaultComboBoxModel<String>) itemComboBox.getModel();
populateComboBox();
itemComboBox.addActionListener( ActionEvent e -> itemComboBoxActionPerformed());
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 0;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
this.add(itemComboBox, gbc);

JLabel codeLabel = new JLabel();
codeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
codeLabel.setText("Code:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(codeLabel, gbc);

sqlCodeLabel = new JLabel();
sqlCodeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlCodeLabel, gbc);

JLabel nameLabel = new JLabel();
nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
nameLabel.setText("Name:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(nameLabel, gbc);

sqlNameLabel = new JLabel();
sqlNameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlNameLabel, gbc);

JLabel locationLabel = new JLabel();
locationLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
locationLabel.setText("Location:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(locationLabel, gbc);
```

UpdateInventoryView.java (3/5)

```
UpdateInventoryView.java
locationTextField = new JTextField();
locationTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 4;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(locationTextField, gbc);

JLabel quantityLabel = new JLabel();
quantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
quantityLabel.setText("Quantity:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(quantityLabel, gbc);

quantityTextField = new JTextField();
quantityTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 4;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(quantityTextField, gbc);

JLabel minQuantityLabel = new JLabel();
minQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
minQuantityLabel.setText("Min Quantity:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 6;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(minQuantityLabel, gbc);

minQuantityTextField = new JTextField();
minQuantityTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 6;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(minQuantityTextField, gbc);

JLabel maxQuantityLabel = new JLabel();
maxQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
maxQuantityLabel.setText("Max Quantity:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 6;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(maxQuantityLabel, gbc);

maxQuantityTextField = new JTextField();
maxQuantityTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 6;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(maxQuantityTextField, gbc);
```

UpdateInventoryView.java (4/5)

```
UpdateInventoryView.java
    JButton submitButton = new JButton();
    submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN, size: 16));
    submitButton.setText("Submit");
    submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(tableModel));
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 8;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 5, left: 5, bottom: 20, right: 5);
    this.add(submitButton, gbc);

    this.addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });
    this.pack();
    this.setLocationRelativeTo(null);
}

private void formWindowClosing() {  2 usages  Giorgos-Karachristos
    itemComboBox.setSelectedIndex(0);
    sqlCodeLabel.setText("");
    sqlNameLabel.setText("");
    locationTextField.setText("");
    quantityTextField.setText("");
    minQuantityTextField.setText("");
    maxQuantityTextField.setText("");
}

private void itemComboBoxActionPerformed() {  1 usage  Giorgos-Karachristos
    for (InventoryItem  inventoryItem : inventoryItemList) {
        if (inventoryItem.getItemId().equals(itemComboBox.getSelectedItem())) {
            sqlCodeLabel.setText(inventoryItem.getItemCode());
            sqlNameLabel.setText(inventoryItem.getName());
            locationTextField.setText(inventoryItem.getLocation());
            quantityTextField.setText(inventoryItem.getQuantity());
            minQuantityTextField.setText(inventoryItem.getMinQuantity());
            maxQuantityTextField.setText(inventoryItem.getMaxQuantity());
            this.pack();
            break;
        }
    }
}

private void submitButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
    String location = locationTextField.getText();
    String quantityText = quantityTextField.getText();
    String minQuantityText = minQuantityTextField.getText();
    String maxQuantityText = maxQuantityTextField.getText();

    if (itemComboBox.getSelectedIndex() == 0 || location.isBlank() || quantityText.isBlank()
            || minQuantityText.isBlank() || maxQuantityText.isBlank()) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "All the fields are required",
                title: "Info", JOptionPane.INFORMATION_MESSAGE);
        return;
    }

    if (!isValid(quantityText,  type: "Quantity") || !isValid(minQuantityText,  type: "Min Quantity")
            || !isValid(maxQuantityText,  type: "Max Quantity")) {
        return;
    }
    int quantity = Integer.parseInt(quantityText);
    int minQuantity = Integer.parseInt(minQuantityText);
    int maxQuantity = Integer.parseInt(maxQuantityText);

    if (minQuantity > maxQuantity) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Min quantity cannot be greater than max quantity",
                title: "Validation Error", JOptionPane.WARNING_MESSAGE);
        return;
    }

    if (quantity > maxQuantity) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Quantity can not pass the maximum quantity",
                title: "Validation Error", JOptionPane.WARNING_MESSAGE);
        return;
    }

    String result = inventoryController.updateInventory(Integer.parseInt((String) itemComboBox.getSelectedItem()),
            location, quantity, minQuantity, maxQuantity);
    if (result.contains("Update failed: ")) {
        JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
    } else {
        JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
    }

    inventoryController.getInventoryTable(tableModel);
    populateComboBox();
    formWindowClosing();
}
```

UpdateInventoryView.java (5/5)

```
UpdateInventoryView.java
    private boolean isValid(String price, String type) {  3 usages  Giorgos-Karachristos *
        try {
            int number = Integer.parseInt(price);
            if (number < 0) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: type + " cannot be negative",
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                return false;
            }
        } catch (NumberFormatException exception) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: type + " must be number",
                    title: "Validation Error", JOptionPane.WARNING_MESSAGE);
            return false;
        }
        return true;
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { inventoryItemList = inventoryController.getInventoryList(itemBoxModel); }
}
```

A.8.5 Item

DeleteItemView.java (1/6)

```
DeleteItem\View.java
package view.item;

import controller.item.ItemController;
import entities.Item;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class DeleteItemView extends JDialog {  2 usages  Giorg

    private final JComboBox<String> itemComboBox;  9 usages
    private final JLabel sqlCodeLabel;  5 usages
    private final JLabel sqlNameLabel;  5 usages
    private final JTextArea descriptionTextArea;  9 usages
    private final JLabel sqlPriceLabel;  5 usages
    private final JLabel sqlCostLabel;  5 usages
    private final JLabel sqlColorLabel;  11 usages
    private final JLabel sqlDateLabel;  5 usages
    private final JLabel sqlCategoryLabel;  5 usages
    private final JLabel sqlIsActiveLabel;  5 usages
    private final DefaultComboBoxModel<String> itemBoxModel;
    private final ItemController itemController;  4 usages

    private List<Item> itemList;  2 usages
```

DeleteItemView.java (2/6)

```
DeleteItemView.java
public DeleteItemView(DefaultTableModel tableModel) {  1 usage   Giorgos-Karachristos
    itemController = new ItemController();
    GridBagConstraints gbc;

    this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    this.setTitle("Delete Item");
    this.setResizable(false);
    GridBagLayout layout = new GridBagLayout();
    layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0};
    layout.rowHeights = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0, 10, 0, 10, 0};
    this.setLayout(layout);

    JLabel itemLabel = new JLabel();
    itemLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    itemLabel.setText("Item ID:");
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.anchor = GridBagConstraints.LINE_START;
    gbc.insets = new Insets( top: 20,  left: 20,  bottom: 5,  right: 5);
    this.add(itemLabel, gbc);

    itemComboBox = new JComboBox<>();
    itemComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    itemBoxModel = (DefaultComboBoxModel<String>) itemComboBox.getModel();
    populateComboBox();
    itemComboBox.addActionListener( ActionEvent e -> itemComboBoxActionPerformed());
    gbc = new GridBagConstraints();
    gbc.gridx = 2;
    gbc.gridy = 0;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
    this.add(itemComboBox, gbc);

    JButton deleteButton = new JButton();
    deleteButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    deleteButton.setText("Delete");
    deleteButton.addActionListener((ActionEvent evt) -> deleteButtonActionPerformed(tableModel));
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 0;
    gbc.gridwidth = 3;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 20);
    this.add(deleteButton, gbc);

    JLabel codeLabel = new JLabel();
    codeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    codeLabel.setText("Code:");
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 2;
    gbc.anchor = GridBagConstraints.LINE_START;
    gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
    this.add(codeLabel, gbc);

    sqlCodeLabel = new JLabel();
    sqlCodeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 2;
    gbc.gridy = 2;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    this.add(sqlCodeLabel, gbc);

    JLabel nameLabel = new JLabel();
    nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    nameLabel.setText("Name:");
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 2;
    gbc.anchor = GridBagConstraints.LINE_START;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    this.add(nameLabel, gbc);
```

DeleteItemView.java (3/6)

```
sqlNameLabel = new JLabel();
sqlNameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 2;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlNameLabel, gbc);

JLabel descriptionLabel = new JLabel();
descriptionLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
descriptionLabel.setText("Description");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.PAGE_END;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(descriptionLabel, gbc);

JScrollPane jScrollPane1 = new JScrollPane();
descriptionTextArea = new JTextArea();
descriptionTextArea.setEditable(false);
descriptionTextArea.setColumns(20);
descriptionTextArea.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
descriptionTextArea.setLineWrap(true);
descriptionTextArea.setRows(4);
jScrollPane1.setViewportView(descriptionTextArea);

gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 6;
gbc.gridwidth = 7;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 100;
gbc.ipady = 50;
gbc.insets = new Insets( top: 0,  left: 20,  bottom: 0,  right: 20);
this.add(jScrollPane1, gbc);

JLabel priceLabel = new JLabel();
priceLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
priceLabel.setText("Price:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 8;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(priceLabel, gbc);

sqlPriceLabel = new JLabel();
sqlPriceLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 8;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlPriceLabel, gbc);

JLabel costLabel = new JLabel();
costLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
costLabel.setText("Cost:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 8;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(costLabel, gbc);
```

DeleteItemView.java (4/6)

```
sqlCostLabel = new JLabel();
sqlCostLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 8;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlCostLabel, gbc);

JLabel colorLabel = new JLabel();
colorLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
colorLabel.setText("Color:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 10;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(colorLabel, gbc);

sqlColorLabel = new JLabel();
sqlColorLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
sqlColorLabel.setText("1234567");
sqlColorLabel.setForeground(this.getBackground());
sqlColorLabel.setBackground(this.getBackground());
sqlColorLabel.setOpaque(true);
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(sqlColorLabel, gbc);

JLabel dateLabel = new JLabel();
dateLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
dateLabel.setText("Create at:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 10;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(dateLabel, gbc);

sqlDateLabel = new JLabel();
sqlDateLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(sqlDateLabel, gbc);

JLabel categoryLabel = new JLabel();
categoryLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
categoryLabel.setText("Category:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 12;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 20,  right: 5);
this.add(categoryLabel, gbc);

sqlCategoryLabel = new JLabel();
sqlCategoryLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 12;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 20,  right: 5);
this.add(sqlCategoryLabel, gbc);
```

DeleteItemView.java (5/6)

```
DeleteItemView.java
    JLabel isActiveLabel = new JLabel();
    isActiveLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    isActiveLabel.setText("Is active:");
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 12;
    gbc.anchor = GridBagConstraints.LINE_START;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 20,  right: 5);
    this.add(isActiveLabel, gbc);

    sqlIsActiveLabel = new JLabel();
    sqlIsActiveLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 6;
    gbc.gridy = 12;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 20,  right: 20);
    this.add(sqlIsActiveLabel, gbc);

    this.addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });

    this.pack();
    this.setLocationRelativeTo(null);
}

private void formWindowClosing() {  2 usages   Giorgos-Karachristos
    itemComboBox.setSelectedIndex(0);
    sqlCodeLabel.setText("");
    sqlNameLabel.setText("");
    descriptionTextArea.setText("");
    sqlPriceLabel.setText("");
    sqlCostLabel.setText("");
    sqlColorLabel.setForeground(this.getBackground());
    sqlColorLabel.setBackground(this.getBackground());
    sqlDateLabel.setText("");
    sqlCategoryLabel.setText("");
    sqlIsActiveLabel.setText("");
}

private void itemComboBoxActionPerformed() {  1 usage   Giorgos-Karachristos
    for (Item item : itemList) {
        if (item.getItemId().equals(itemComboBox.getSelectedItem())) {
            sqlCodeLabel.setText(item.getItemCode());
            sqlNameLabel.setText(item.getName());
            descriptionTextArea.setText(item.getDescription());
            sqlPriceLabel.setText(item.getPrice());
            sqlCostLabel.setText(item.getCost());
            sqlCategoryLabel.setText(item.getCategory());
            sqlDateLabel.setText(item.getCreatedAt());
            sqlColorLabel.setForeground(Color.decode(item.getColor()));
            sqlColorLabel.setBackground(Color.decode(item.getColor()));
            sqlIsActiveLabel.setText(item.getIsActive());
            //repaint
            this.pack();
            break;
        }
    }
}
```

DeleteItemView.java (6/6)

```
DeleteItem\View.java
    private void deleteButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
        if (itemComboBox.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please select an item to delete",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        String result = itemController.deleteItem(Integer.parseInt((String) itemComboBox.getSelectedItem()));
        if (result.contains("Delete failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        itemController.getItemTable(tableModel);
        populateComboBox();
        formWindowClosing();
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { itemList = itemController.getItemList(itemBoxModel); }
}
```

InsertItemView.java (1/4)

```
InsertItem\View.java
package view.item;

import controller.category.CategoryController;
import controller.item.ItemController;
import util.CustomColorChooser;
import util.CustomDocumentFilter;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.*;

public class InsertItemView  extends JDialog {  2 usages  Giorgos-K

    private static final int CHAR_LIMIT = 500;  4 usages

    private final JTextField codeTextField;  5 usages
    private final JTextField nameTextField;  5 usages
    private final JTextArea descriptionTextArea;  11 usages
    private final JLabel countLabel;  6 usages
    private final JTextField priceTextField;  6 usages
    private final JTextField costTextField;  5 usages
    private final JButton colorButton;  8 usages
    private final JComboBox<String>categoryComboBox;  7 usages
    private final ButtonGroup buttonGroup1;  5 usages
    private final JRadioButton noRadioButton;  7 usages
    private final JRadioButton yesRadioButton;  7 usages
    private final DefaultComboBoxModel<String> categoryBoxModel;
    private final ItemController itemController;  3 usages
    private final CategoryController categoryController;  2 usages
```

InsertItemView.java (2/4)

```
public InsertItemView(DefaultTableModel tableModel) {  1 usage   Giorgos-Karachristos
    itemController = new ItemController();
    categoryController = new CategoryController();
    GridBagConstraints gbc;

    this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    this.setTitle("Add Item");
    this.setResizable(false);
    this.setLayout(new GridBagLayout());

    JLabel codeLabel = new JLabel();
    codeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    codeLabel.setText("Code:");
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.insets = new Insets( top: 20,  left: 20,  bottom: 5,  right: 5);
    this.add(codeLabel, gbc);

    codeTextField = new JTextField();
    codeTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 0;
    gbc.ipadx = 150;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
    this.add(codeTextField, gbc);

    JLabel nameLabel = new JLabel();
    nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    nameLabel.setText("Name:");
    gbc = new GridBagConstraints();
    gbc.gridx = 3;
    gbc.gridy = 0;
    gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
    this.add(nameLabel, gbc);

    nameTextField = new JTextField();
    nameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 0;
    gbc.gridwidth = 2;
    gbc.ipadx = 150;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 20);
    this.add(nameTextField, gbc);

    JLabel descriptionLabel = new JLabel();
    descriptionLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    descriptionLabel.setText("Description");
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 1;
    gbc.anchor = GridBagConstraints.PAGE_END;
    gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
    this.add(descriptionLabel, gbc);

    descriptionTextArea = new JTextArea();
    descriptionTextArea.setColumns(20);
    descriptionTextArea.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    descriptionTextArea.setLineWrap(true);
    descriptionTextArea.setRows(4);
    descriptionTextArea.addKeyListener((KeyAdapter) keyTyped(evt) -> {
            descriptionTextAreaKeyTyped();
    });
    CustomDocumentFilter df = new CustomDocumentFilter();
    df.setDocumentFilter(descriptionTextArea, CHAR_LIMIT);
```

InsertItemView.java (3/4)

```
JScrollPane jScrollPane1 = new JScrollPane();
jScrollPane1.setViewportView(descriptionTextArea);
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.gridwidth = 6;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 100;
gbc.ipady = 50;
gbc.insets = new Insets( top: 0,  left: 20,  bottom: 0,  right: 20);
this.add(jScrollPane1, gbc);

countLabel = new JLabel();
countLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
countLabel.setText("Character Count: " + CHAR_LIMIT);
gbc = new GridBagConstraints();
gbc.gridx = 5;
gbc.gridy = 3;
gbc.fill = GridBagConstraints.VERTICAL;
gbc.anchor = GridBagConstraints.LINE_END;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(countLabel, gbc);

JLabel priceLabel = new JLabel();
priceLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
priceLabel.setText("Price:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(priceLabel, gbc);

priceTextField = new JTextField();
priceTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 4;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(priceTextField, gbc);

JLabel costLabel = new JLabel();
costLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
costLabel.setText("Cost:");
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 4;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(costLabel, gbc);

costTextField = new JTextField();
costTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 4;
gbc.gridwidth = 2;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 10,  bottom: 5,  right: 20);
this.add(costTextField, gbc);

colorButton = new JButton();
colorButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
colorButton.setText("Color");
colorButton.addActionListener( ActionEvent evt1 -> colorButtonActionPerformed());
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 5;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(colorButton, gbc);
```

InsertItemView.java (4/4)

```
InsertItem\View.java
        JLabel categoryLabel = new JLabel();
        categoryLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        categoryLabel.setText("Category:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 6;
        gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
        this.add(categoryLabel, gbc);

        categoryComboBox = new JComboBox<>();
        categoryComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        categoryBoxModel = (DefaultComboBoxModel<String>) categoryComboBox.getModel();
        populateComboBox();
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 6;
        gbc.ipadx = 100;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
        this.add(categoryComboBox, gbc);

        JLabel isActiveLabel = new JLabel();
        isActiveLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        isActiveLabel.setText("Is active:");
        gbc = new GridBagConstraints();
        gbc.gridx = 3;
        gbc.gridy = 6;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
        this.add(isActiveLabel, gbc);

        buttonGroup1 = new ButtonGroup();
        yesRadioButton = new JRadioButton();
        buttonGroup1.add(yesRadioButton);
        yesRadioButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        yesRadioButton.setText("Yes");
        yesRadioButton.setActionCommand("Yes");
        gbc = new GridBagConstraints();
        gbc.gridx = 4;
        gbc.gridy = 6;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
        this.add(yesRadioButton, gbc);

        noRadioButton = new JRadioButton();
        buttonGroup1.add(noRadioButton);
        noRadioButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        noRadioButton.setText("No");
        noRadioButton.setActionCommand("No");
        gbc = new GridBagConstraints();
        gbc.gridx = 5;
        gbc.gridy = 6;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
        this.add(noRadioButton, gbc);

        JButton submitButton = new JButton();
        submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        submitButton.setText("Submit");
        submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(tableModel));
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 7;
        gbc.ipadx = 100;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 5,  left: 5,  bottom: 20,  right: 5);
        this.add(submitButton, gbc);

        this.addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });

        this.pack();
        this.setLocationRelativeTo(null);
    }
```

InsertItemView.java (5/6)

```java
private void descriptionTextAreaKeyTyped() {  1 usage  Giorgos-Karachristos
    int charCount = descriptionTextArea.getDocument().getLength();
    countLabel.setText("Character Count: " + (CHAR_LIMIT - charCount));
}

private void colorButtonActionPerformed() {  1 usage  Giorgos-Karachristos
    CustomColorChooser chooser = new CustomColorChooser(colorButton);
    chooser.setModal(true);
    chooser.setVisible(true);
}

private void formWindowClosing() {  2 usages  Giorgos-Karachristos
    codeTextField.setText("");
    nameTextField.setText("");
    descriptionTextArea.setText("");
    countLabel.setText("Character Count: " + CHAR_LIMIT);
    priceTextField.setText("");
    costTextField.setText("");
    colorButton.setBackground(Color.WHITE);
    categoryComboBox.setSelectedIndex(0);
    buttonGroup1.clearSelection();
}

private void submitButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
    String code = codeTextField.getText().trim();
    String name = nameTextField.getText().trim();
    String costText = costTextField.getText().trim();
    String priceText = priceTextField.getText().isBlank() ? "0" : priceTextField.getText().trim();

    if (code.isBlank() || name.isBlank() || costText.isBlank() || categoryComboBox.getSelectedIndex() == 0
            || !noRadioButton.isSelected() && !yesRadioButton.isSelected()) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "All the fields are required",
                title: "Info", JOptionPane.INFORMATION_MESSAGE);
        return;
    }

    if (!isValid(costText,  type: "Cost") || !isValid(priceText,  type: "Price")) {
        return;
    }
    double cost = Double.parseDouble(costText);
    double price = Double.parseDouble(priceText);

    Color color = colorButton.getBackground();
    String hexColor = String.format("#%02x%02x%02x", color.getRed(), color.getGreen(), color.getBlue());

    String result = itemController.insertItem(code, name, descriptionTextArea.getText().trim(),
            price, cost, (String) categoryComboBox.getSelectedItem(),
            hexColor, buttonGroup1.getSelection().getActionCommand());
    if (result.contains("Insertion failed: ")) {
        JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
    } else {
        JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
    }

    itemController.getItemTable(tableModel);
    formWindowClosing();
}
```

InsertItemView.java (6/6)

```
InsertItem\view.java
    private boolean isValid(String price, String type) {  2 usages   Giorgos-Karachristos *
        try {
            double number = Double.parseDouble(price);
            if (number < 0) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: type + " cannot be negative",
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                return false;
            }
        } catch (NumberFormatException exception) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: type + " must be number",
                    title: "Validation Error", JOptionPane.WARNING_MESSAGE);
            return false;
        }
        return true;
    }

    @Override   Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { categoryController.getCategoryName(categoryBoxModel); }
}
```

UpdateItemView.java (1/7)

```
UpdateItem\view.java
package view.item;

import controller.category.CategoryController;
import controller.item.ItemController;
import entities.Item;
import util.CustomColorChooser;
import util.CustomDocumentFilter;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.*;
import java.util.List;

public class UpdateItemView extends JDialog {  2 usages   Giorgos-Ka

    private static final int CHAR_LIMIT = 500;  5 usages

    private final JTextField codeTextField;  6 usages
    private final JTextField nameTextField;  6 usages
    private final JTextArea descriptionTextArea;  13 usages
    private final JLabel countLabel;  7 usages
    private final JTextField priceTextField;  7 usages
    private final JTextField costTextField;  6 usages
    private final JButton colorButton;  9 usages
    private final JComboBox<String> categoryComboBox;  10 usages
    private final JComboBox<String> itemComboBox;  9 usages
    private final ButtonGroup buttonGroup1;  5 usages
    private final JRadioButton noRadioButton;  8 usages
    private final JRadioButton yesRadioButton;  8 usages
    private final DefaultComboBoxModel<String> itemBoxModel;  2 u
    private final DefaultComboBoxModel<String> categoryBoxModel;
    private final ItemController itemController;  4 usages
    private final CategoryController categoryController;  2 usages

    private List<Item> itemList;  2 usages
```

UpdateItemView.java (2/7)

```
UpdateItem\View.java
public UpdateItemView(DefaultTableModel tableModel) {  1 usage   Giorgos-Karachri
    itemController = new ItemController();
    categoryController = new CategoryController();
    GridBagConstraints gbc;

    this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
    this.setTitle("Edit Item");
    this.setResizable(false);
    this.setLayout(new GridBagLayout());

    JLabel itemLabel = new JLabel();
    itemLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    itemLabel.setText("Item ID:");
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 0;
    gbc.gridwidth = 2;
    gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
    this.add(itemLabel, gbc);

    itemComboBox = new JComboBox<>();
    itemComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    itemBoxModel = (DefaultComboBoxModel<String>) itemComboBox.getModel();
    gbc = new GridBagConstraints();
    gbc.gridx = 2;
    gbc.gridy = 0;
    gbc.gridwidth = 2;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 20,  left: 5,  bottom: 5,  right: 5);
    this.add(itemComboBox, gbc);

    JLabel codeLabel = new JLabel();
    codeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    codeLabel.setText("Code:");
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 1;
    gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
    this.add(codeLabel, gbc);

    codeTextField = new JTextField();
    codeTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 1;
    gbc.ipadx = 150;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    this.add(codeTextField, gbc);

    JLabel nameLabel = new JLabel();
    nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    nameLabel.setText("Name:");
    gbc = new GridBagConstraints();
    gbc.gridx = 3;
    gbc.gridy = 1;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    this.add(nameLabel, gbc);

    nameTextField = new JTextField();
    nameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 1;
    gbc.gridwidth = 2;
    gbc.ipadx = 150;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
    this.add(nameTextField, gbc);
```

UpdateItemView.java (3/7)

```
UpdateItem\View.java
JLabel descriptionLabel = new JLabel();
descriptionLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
descriptionLabel.setText("Description");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.anchor = GridBagConstraints.PAGE_END;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(descriptionLabel, gbc);

descriptionTextArea = new JTextArea();
descriptionTextArea.setColumns(20);
descriptionTextArea.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
descriptionTextArea.setLineWrap(true);
descriptionTextArea.setRows(4);
descriptionTextArea.addKeyListener((KeyAdapter) keyTyped(evt) -> {
        descriptionTextAreaKeyTyped();
});
CustomDocumentFilter df = new CustomDocumentFilter();
df.setDocumentFilter(descriptionTextArea, CHAR_LIMIT);

JScrollPane jScrollPane1 = new JScrollPane();
jScrollPane1.setViewportView(descriptionTextArea);
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 3;
gbc.gridwidth = 6;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 100;
gbc.ipady = 50;
gbc.insets = new Insets( top: 0,  left: 20,  bottom: 0,  right: 20);
this.add(jScrollPane1, gbc);

countLabel = new JLabel();
countLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
countLabel.setText("Character Count: " + CHAR_LIMIT);
gbc = new GridBagConstraints();
gbc.gridx = 5;
gbc.gridy = 4;
gbc.fill = GridBagConstraints.VERTICAL;
gbc.anchor = GridBagConstraints.LINE_END;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(countLabel, gbc);

JLabel priceLabel = new JLabel();
priceLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
priceLabel.setText("Price:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 5;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(priceLabel, gbc);

priceTextField = new JTextField();
priceTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 5;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(priceTextField, gbc);
```

UpdateItemView.java (4/7)

```
JLabel costLabel = new JLabel();
costLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
costLabel.setText("Cost:");
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 5;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(costLabel, gbc);

costTextField = new JTextField();
costTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 5;
gbc.gridwidth = 2;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 20);
this.add(costTextField, gbc);

colorButton = new JButton();
colorButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
colorButton.setText("Color");
colorButton.addActionListener( ActionEvent evt1 -> colorButtonActionPerformed());
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 6;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(colorButton, gbc);

JLabel categoryLabel = new JLabel();
categoryLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
categoryLabel.setText("Category:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 7;
gbc.insets = new Insets( top: 5,  left: 20,  bottom: 5,  right: 5);
this.add(categoryLabel, gbc);

categoryComboBox = new JComboBox<>();
categoryComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
categoryBoxModel = (DefaultComboBoxModel<String>) categoryComboBox.getModel();
populateComboBoxs();
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 7;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(categoryComboBox, gbc);

itemComboBox.addActionListener( ActionEvent e -> itemComboBoxActionPerformed());

JLabel isActiveLabel = new JLabel();
isActiveLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
isActiveLabel.setText("Is active:");
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 7;
gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
this.add(isActiveLabel, gbc);
```

UpdateItemView.java (5/7)

```
    buttonGroup1 = new ButtonGroup();
    yesRadioButton = new JRadioButton();
    buttonGroup1.add(yesRadioButton);
    yesRadioButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    yesRadioButton.setText("Yes");
    yesRadioButton.setActionCommand("Yes");
    gbc = new GridBagConstraints();
    gbc.gridx = 4;
    gbc.gridy = 7;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    this.add(yesRadioButton, gbc);

    noRadioButton = new JRadioButton();
    buttonGroup1.add(noRadioButton);
    noRadioButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    noRadioButton.setText("No");
    noRadioButton.setActionCommand("No");
    gbc = new GridBagConstraints();
    gbc.gridx = 5;
    gbc.gridy = 7;
    gbc.anchor = GridBagConstraints.LINE_START;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    this.add(noRadioButton, gbc);

    JButton submitButton = new JButton();
    submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    submitButton.setText("Submit");
    submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(tableModel));
    gbc = new GridBagConstraints();
    gbc.gridx = 2;
    gbc.gridy = 8;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 20,  right: 5);
    this.add(submitButton, gbc);

    this.addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });

    this.pack();
    this.setLocationRelativeTo(null);
}

private void descriptionTextAreaKeyTyped() {  1 usage   Giorgos-Karachristos
    int charCount = descriptionTextArea.getDocument().getLength();
    countLabel.setText("Character Count: " + (CHAR_LIMIT - charCount));
}

private void colorButtonActionPerformed() {  1 usage   Giorgos-Karachristos
    CustomColorChooser chooser = new CustomColorChooser(colorButton);
    chooser.setModal(true);
    chooser.setVisible(true);
}
```

UpdateItemView.java (6/7)

```
UpdateItemView.java
private void formWindowClosing() {  2 usages  Giorgos-Karachristos
    itemComboBox.setSelectedIndex(0);
    codeTextField.setText("");
    nameTextField.setText("");
    descriptionTextArea.setText("");
    countLabel.setText("Character Count: " + CHAR_LIMIT);
    priceTextField.setText("");
    costTextField.setText("");
    colorButton.setBackground(Color.WHITE);
    categoryComboBox.setSelectedIndex(0);
    buttonGroup1.clearSelection();
}

private void itemComboBoxActionPerformed() {  1 usage  Giorgos-Karachristos
    for (Item item : itemList) {
        if (item.getItemId().equals(itemComboBox.getSelectedItem())) {
            codeTextField.setText(item.getItemCode());
            nameTextField.setText(item.getName());
            descriptionTextArea.setText(item.getDescription() == null ? "" : item.getDescription());
            priceTextField.setText(item.getPrice());
            costTextField.setText(item.getCost());
            String category = item.getCategory();
            for (int x = 0; x <= categoryComboBox.getItemCount(); x++) {
                if (category.equals(categoryComboBox.getItemAt(x))) {
                    categoryComboBox.setSelectedIndex(x);
                    break;
                }
            }
            //7 date
            colorButton.setBackground(Color.decode(item.getColor()));
            if ("Yes".equals(item.getIsActive())) {
                yesRadioButton.setSelected(true);
            } else {
                noRadioButton.setSelected(true);
            }
            int charCount = descriptionTextArea.getDocument().getLength();
            countLabel.setText("Character Count: " + (CHAR_LIMIT - charCount));
            this.pack();
            break;
        }
    }
}

private void submitButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
    String code = codeTextField.getText().trim();
    String name = nameTextField.getText().trim();
    String costText = costTextField.getText().trim();
    String priceText = priceTextField.getText().isBlank() ? "0" : priceTextField.getText().trim();

    if (itemComboBox.getSelectedIndex() == 0 || code.isBlank() || name.isBlank() || costText.isBlank()
            || categoryComboBox.getSelectedIndex() == 0 || !noRadioButton.isSelected() && !yesRadioButton.isSelected()) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "All the fields are required",
                title: "Info", JOptionPane.INFORMATION_MESSAGE);
        return;
    }

    if (!isValid(costText,  type: "Cost") || !isValid(priceText,  type: "Price")) {
        return;
    }
    double cost = Double.parseDouble(costText);
    double price = Double.parseDouble(priceText);

    Color color = colorButton.getBackground();
    String hexColor = String.format("#%02x%02x%02x", color.getRed(), color.getGreen(), color.getBlue());

    String result = itemController.updateItem(Integer.parseInt((String) itemComboBox.getSelectedItem()),
            code, name, descriptionTextArea.getText().trim(), price, cost, (String) categoryComboBox.getSelectedItem(),
            hexColor, buttonGroup1.getSelection().getActionCommand());
    if (result.contains("Update failed: ")) {
        JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
    } else {
        JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
    }

    itemController.getItemTable(tableModel);
    populateComboBoxs();
    formWindowClosing();
}
```

UpdateItemView.java (7/7)

```
UpdateItemView.java
    private boolean isValid(String price, String type) {  2 usages  Giorgos-Karachristos *
        try {
            double number = Double.parseDouble(price);
            if (number < 0) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: type + " cannot be negative",
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                return false;
            }
        } catch (NumberFormatException exception) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: type + " must be number",
                    title: "Validation Error", JOptionPane.WARNING_MESSAGE);
            return false;
        }
        return true;
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBoxs();
        }
        super.setVisible(visible);
    }

    private void populateComboBoxs() {  3 usages  Giorgos-Karachristos
        itemList = itemController.getItemList(itemBoxModel);
        categoryController.getCategoryName(categoryBoxModel);
    }
}
```

A.8.6 Kitchen

KitchenPanel.java (1/3)

```
KitchenPanel.java
package view.kitchen;

import controller.kitchen.KitchenController;
import entities.KitchenItem;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class KitchenPanel extends JPanel {  2 usages  Giorgos-Karachristos

    private final KitchenController kitchenController;  4 usages
    private final Map<String, ArrayList<JCheckBox>> jCheckBoxMap = new HashMap<>();  2 usages
    private final JButton servedButton;  7 usages

    public KitchenPanel(String orderID, String orderStatus, String orderDateTime) {  1 usage
        kitchenController = new KitchenController();
        GridBagConstraints gbc;

        this.setLayout(new BorderLayout());
        this.setBorder(BorderFactory.createLineBorder(new Color( r: 0,  g: 0,  b: 0)));

        JPanel orderTopPanel = new JPanel();
        orderTopPanel.setBorder(BorderFactory.createLineBorder(new Color( r: 0,  g: 0,  b: 0)));
        orderTopPanel.setLayout(new FlowLayout(FlowLayout.LEFT));

        JLabel orderIDLabel = new JLabel();
        orderIDLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        orderIDLabel.setText("Order ID:");
        orderTopPanel.add(orderIDLabel);
```

KitchenPanel.java (2/3)

```
KitchenPanel.java
JLabel sqlOrderIDLabel = new JLabel();
sqlOrderIDLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
sqlOrderIDLabel.setText(orderID);
orderTopPanel.add(sqlOrderIDLabel);

JLabel statusLabel = new JLabel();
statusLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
statusLabel.setText("Status:");
orderTopPanel.add(statusLabel);

JLabel sqlStatusLabel = new JLabel();
sqlStatusLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
sqlStatusLabel.setText(orderStatus);
orderTopPanel.add(sqlStatusLabel);

JLabel dateTimeLabel = new JLabel();
dateTimeLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
dateTimeLabel.setText("Date-Time:");
orderTopPanel.add(dateTimeLabel);

JLabel sqlDateTimeLabel = new JLabel();
sqlDateTimeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
sqlDateTimeLabel.setText(orderDateTime);
orderTopPanel.add(sqlDateTimeLabel);

this.add(orderTopPanel, BorderLayout.PAGE_START);

List<KitchenItem> kitchenItemList = kitchenController.getCookList(orderID);
JPanel orderCenterPanel = new JPanel();
orderCenterPanel.setLayout(new GridBagLayout());

ArrayList<JCheckBox> boxes = new ArrayList<>();
int gridy = 0;
for (KitchenItem kitchenItem : kitchenItemList) {
    JLabel nameLabel = new JLabel();
    nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    nameLabel.setText(kitchenItem.getName());
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = gridy;
    gbc.weightx = 1.0;
    gbc.weighty = 1.0;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    orderCenterPanel.add(nameLabel, gbc);

    JLabel quantityLabel = new JLabel();
    quantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    quantityLabel.setText(String.valueOf(kitchenItem.getQuantity()));
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = gridy;
    gbc.weightx = 1.0;
    gbc.weighty = 1.0;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    orderCenterPanel.add(quantityLabel, gbc);

    JCheckBox jCheckBox1 = new JCheckBox();
    jCheckBox1.setName(String.valueOf(kitchenItem.getItemID()));
    if ("Completed".equals(kitchenItem.getStatus())) {
        jCheckBox1.setSelected(true);
    }
    gbc = new GridBagConstraints();
    gbc.gridx = 2;
    gbc.gridy = gridy;
    gbc.weightx = 1.0;
    gbc.weighty = 1.0;
    gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
    orderCenterPanel.add(jCheckBox1, gbc);
    if (!"Completed".equals(kitchenItem.getStatus())) {
        boxes.add(jCheckBox1);
    }
    gridy++;
}
jCheckBoxMap.put(orderID, boxes);
```

KitchenPanel.java (3/3)

KitchenPanel.java

```
        JScrollPane jScrollPane = new JScrollPane(orderCenterPanel);
        this.add(jScrollPane, BorderLayout.CENTER);

        JPanel bottomPanel = new JPanel();
        bottomPanel.setLayout(new GridLayout());

        servedButton = new JButton();
        servedButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        servedButton.setText("Mark as Served");
        servedButton.addActionListener((ActionEvent evt) -> servedButtonActionPerformed(orderID));
        servedButton.setEnabled(boxes.isEmpty());
        bottomPanel.add(servedButton);

        JButton updateStatusButton = new JButton();
        updateStatusButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        updateStatusButton.addActionListener((ActionEvent evt) ->
                updateStatusButtonActionPerformed(orderID, jCheckBoxMap.get(orderID)));
        updateStatusButton.setText("Update Items Status");
        bottomPanel.add(updateStatusButton);

        this.add(bottomPanel, BorderLayout.PAGE_END);
    }

    @Override  Giorgos-Karachristos
    public Dimension getPreferredSize() {
        Dimension preferredSize = super.getPreferredSize();
        preferredSize.height = 500;
        return preferredSize;
    }

    private void servedButtonActionPerformed(String orderID) {  1 usage  Giorgos-Karachristos
        String result = kitchenController.markAsServed(orderID);
        if (result.contains("Delete failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }
    }

    private void updateStatusButtonActionPerformed(String orderID, ArrayList<JCheckBox> jCheckBoxArrayList) {  1 usage  
        int succed = 0;
        int failed = 0;
        String result;
        for (int i = jCheckBoxArrayList.size() - 1; i >= 0; i--) {
            if (jCheckBoxArrayList.get(i).isSelected()) {
                result = kitchenController.updateStatus(orderID, jCheckBoxArrayList.get(i).getName());
                if (result.contains("Insertion failed: ")) {
                    JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
                    failed++;
                } else {
                    succed++;
                }
                jCheckBoxArrayList.remove(i);
            }
            if (jCheckBoxArrayList.isEmpty()) {
                servedButton.setEnabled(true);
            }
        }
        if (succed == 0 && failed == 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please select an item to update.",
                    title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this,  message: succed + " items updated successfully.\n"
                    + failed + " items failed to update.",  title: "Transaction Results", JOptionPane.INFORMATION_MESSAGE);
        }
    }
}
```

KitchenView.java

```java
package view.kitchen;

import controller.kitchen.KitchenController;
import entities.Order;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.util.List;

public class KitchenView extends JPanel {  2 usages  Giorgos-Karachristos

    private final KitchenController kitchenController;  2 usages
    private JScrollPane jScrollPane;  6 usages
    private List<Order> ordersData;  5 usages

    public KitchenView() {  1 usage  Giorgos-Karachristos
        kitchenController = new KitchenController();
        this.setLayout(new BorderLayout());

        JPanel kitchenTopPanel = new JPanel();
        kitchenTopPanel.setBackground(new Color( r: 250,  g: 188,  b: 63));

        JButton refreshButton = new JButton();
        refreshButton.setBackground(new Color( r: 63,  g: 169,  b: 245));
        refreshButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        refreshButton.setText("Refresh");
        refreshButton.addActionListener((ActionEvent evt) -> refreshButtonActionPerformed());
        kitchenTopPanel.add(refreshButton);

        this.add(kitchenTopPanel, BorderLayout.PAGE_START);

        updateTabbedPane();
    }

    private void updateTabbedPane() {  2 usages  Giorgos-Karachristos
        int spacing = 10;

        if (jScrollPane != null) {
            this.remove(jScrollPane);
        }

        JPanel kitchenCenterPanel = new JPanel();
        kitchenCenterPanel.setLayout(new FlowLayout(FlowLayout.LEFT, spacing, spacing));

        KitchenPanel kp = null;
        ordersData = kitchenController.getOrdersList();
        for (Order order : ordersData) {
            String orderID = order.getOrderID();
            String orderStatus = order.getStatus();
            String orderDateTime = order.getOrderDate();

            kp = new KitchenPanel(orderID, orderStatus, orderDateTime);
            kitchenCenterPanel.add(kp);
        }
        if (kp != null) {
            int screenWidth = Toolkit.getDefaultToolkit().getScreenSize().width;

            int innerPanelWidth = kp.getPreferredSize().width;

            int panelsPerRow = screenWidth / (innerPanelWidth + spacing);

            int numRows = (int) Math.ceil((double) ordersData.size() / panelsPerRow);

            int outerPanelWidth = screenWidth - 100;

            int innerPanelHeight = kp.getPreferredSize().height;
            int outerPanelHeight = (innerPanelHeight + (int) Math.ceil(spacing * 1.5)) * numRows;

            kitchenCenterPanel.setPreferredSize(new Dimension(outerPanelWidth, outerPanelHeight));
        }

        jScrollPane = new JScrollPane(kitchenCenterPanel);

        this.add(jScrollPane, BorderLayout.CENTER);
        this.revalidate();
        this.repaint();
    }

    private void refreshButtonActionPerformed() {  1 usage  Giorgos-Karachristos
        ordersData.removeAll(ordersData);
        updateTabbedPane();
        jScrollPane.revalidate();
        jScrollPane.repaint();
    }
}
```

A.8.7 Login

LoginView.java (1/2)

```
LoginView.java
package view.login;

import controller.login.LoginController;
import controller.pos.POSController;
import controller.restaurant.RestaurantController;

import javax.swing.*;
import java.awt.*;
import java.awt.geom.RoundRectangle2D;

public class LoginView extends JPanel {  3 usages  Giorgos-Karachristos

    private final JTextField usernameField;  5 usages
    private final JPasswordField passwordField;  5 usages
    private final LoginController loginController;  2 usages

    public LoginView(POSController posController) {  1 usage  Giorgos-Karachristos
        this.loginController = new LoginController(posController);
        this.setOpaque(false);
        this.setBackground(Color.WHITE);
        this.setPreferredSize(new Dimension( width: 450,  height: 600));
        this.setLayout(new GridBagLayout());

        GridBagConstraints gbc;

        JLabel welcomeLabel = new JLabel();
        welcomeLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 30));
        welcomeLabel.setText("Welcome to");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 2;
        gbc.ipadx = 12;
        gbc.ipady = 12;
        gbc.anchor = java.awt.GridBagConstraints.PAGE_END;
        gbc.weighty = 0.2;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(welcomeLabel, gbc);

        JLabel nameLabel = new JLabel();
        nameLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 30));
        RestaurantController restaurantController = new RestaurantController();
        nameLabel.setText(restaurantController.getRestaurantName());
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 2;
        gbc.ipadx = 12;
        gbc.ipady = 12;
        gbc.anchor = java.awt.GridBagConstraints.PAGE_START;
        gbc.weighty = 0.2;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(nameLabel, gbc);

        JLabel usernameLabel = new JLabel( text: "Username:");
        usernameLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 20));
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.ipadx = 12;
        gbc.ipady = 12;
        gbc.weighty = 0.1;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(usernameLabel, gbc);
```

LoginView.java (2/2)

```
        usernameField = new JTextField();
        usernameField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 2;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 200;
        gbc.ipady = 12;
        gbc.weighty = 0.1;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(usernameField, gbc);

        JLabel passwordLabel = new JLabel( text: "Password:");
        passwordLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 20));
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 3;
        gbc.ipadx = 12;
        gbc.ipady = 12;
        gbc.weighty = 0.1;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(passwordLabel, gbc);

        passwordField = new JPasswordField();
        passwordField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 3;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 200;
        gbc.ipady = 12;
        gbc.weighty = 0.1;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(passwordField, gbc);

        JButton signInButton = new JButton( text: "Sign In");
        signInButton.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 20));
        signInButton.setForeground(Color.WHITE);
        signInButton.setBackground(new Color( r: 57,  g: 153,  b: 24));
        signInButton.addActionListener( ActionEvent e -> {
            setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
            loginController.handleLogin(usernameField.getText(), String.valueOf(passwordField.getPassword()));
            setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
            usernameField.setText("");
            passwordField.setText("");
        });
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.gridwidth = 2;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 12;
        gbc.ipady = 12;
        gbc.weighty = 0.1;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(signInButton, gbc);

        JButton exitButton = new JButton( text: "Exit");
        exitButton.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 20));
        exitButton.setForeground(Color.WHITE);
        exitButton.setBackground(Color.RED);
        exitButton.addActionListener( ActionEvent e -> {
            int a = JOptionPane.showConfirmDialog( parentComponent: this,  message: "Do you want to close this app?",
                    title: "Exit", JOptionPane.YES_NO_OPTION);
            if (a == 0) System.exit( status: 0);
        });
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 5;
        gbc.gridwidth = 2;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 12;
        gbc.ipady = 12;
        gbc.weighty = 0.1;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(exitButton, gbc);
    }

    @Override  Giorgos-Karachristos *
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2d = (Graphics2D) g.create();
        int cornerRadius = 50;
        RoundRectangle2D roundedRectangle = new RoundRectangle2D.Float( x: 0,  y: 0, getWidth(), getHeight(),
                cornerRadius, cornerRadius);
        g2d.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIALIAS_ON);
        g2d.setColor(getBackground());
        g2d.fill(roundedRectangle);
        g2d.dispose();
    }
}
```

A.8.8 Manager

CustomTabPanel.java

```
CustomTabPanel.java
package view.manager;

import controller.manager.CustomTabController;
import util.CustomTableCellRenderer;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;

public class CustomTabPanel extends JPanel {  2 usages  Giorgos-Karachristos
    private final DefaultTableModel tableModel;  7 usages
    private final CustomTabController controller;  7 usages

    public CustomTabPanel(String tabName,JTabbedPane jtp) {  1 usage  Giorgos-Karachristos
        this.controller = new CustomTabController(tabName);
        this.setLayout(new BorderLayout());

        // Top buttons panel
        JPanel northPanel = new JPanel(); //Panel with 3 buttons in the center
        northPanel.setPreferredSize(new Dimension( width: 75,  height: 75));
        northPanel.setLayout(new FlowLayout(FlowLayout.CENTER,  hgap: 50,  vgap: 25));

        JButton addButton = new JButton();
        addButton.setBackground(new Color( r: 107,  g: 203,  b: 119));
        addButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        addButton.setText("Add " + tabName);
        northPanel.add(addButton);

        JButton updateButton = new JButton();
        if (!tabName.equals("Transactions")) {
            updateButton.setBackground(new Color( r: 255,  g: 217,  b: 61));
            updateButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
            updateButton.setText("Edit " + tabName);
            northPanel.add(updateButton);
        }

        JButton deleteButton = new JButton();
        deleteButton.setBackground(new Color( r: 238,  g: 78,  b: 78));
        deleteButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        deleteButton.setText("Delete " + tabName);
        northPanel.add(deleteButton);

        this.add(northPanel, BorderLayout.NORTH);

        //Empty panel
        JPanel westPanel = new JPanel();
        westPanel.setPreferredSize(new Dimension( width: 50,  height: 50));
        this.add(westPanel, BorderLayout.WEST);

        // Table
        tableModel = new DefaultTableModel(new Object[][]{}, controller.getColumnNames()) {  Giorgos-Karachristos
            @Override  Giorgos-Karachristos
            public boolean isCellEditable(int row, int column) { return false; }
        };
        JTable jTable = new JTable(tableModel);
        jTable.getTableHeader().setReorderingAllowed(false);
        jTable.getTableHeader().setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        jTable.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        jTable.setRowHeight(40);

        JScrollPane scrollPane = new JScrollPane(jTable);
        this.add(scrollPane, BorderLayout.CENTER);

        //Empty panel
        JPanel eastPanel = new JPanel();
        eastPanel.setPreferredSize(new Dimension( width: 50,  height: 50));
        this.add(eastPanel, BorderLayout.EAST);

        // Load initial data
        controller.loadTableData(tableModel);

        // Event listeners
        addButton.addActionListener( ActionEvent e -> controller.handleAddAction(tableModel));
        updateButton.addActionListener( ActionEvent e -> controller.handleUpdateAction(tableModel));
        deleteButton.addActionListener( ActionEvent e -> controller.handleDeleteAction(tableModel));

        switch (tabName) {
            case "Item":
                jTable.getColumnModel().getColumn( columnIndex: 8).setCellRenderer(new CustomTableCellRenderer());
                break;
            case "Category":
                jTable.getColumnModel().getColumn( columnIndex: 2).setCellRenderer(new CustomTableCellRenderer());
                break;
        }
        jtp.addChangeListener( ChangeEvent e -> refreshTable());
    }

    public void refreshTable() { controller.loadTableData(tableModel); }
}
```

ManagerPane.java

```java
package view.manager;

import util.CustomImageIcon;
import view.reports.ReportsView;
import view.restaurant.RestaurantView;

import javax.swing.*;
import java.awt.*;

public class ManagerPane extends JTabbedPane {  2 usages  Giorgos-Karachristos

    public ManagerPane() {  1 usage  Giorgos-Karachristos
        this.setTabPlacement(JTabbedPane.LEFT);
        this.putClientProperty("JTabbedPane.tabAlignment", "leading");
        this.putClientProperty("JTabbedPane.showTabSeparators", true);
        this.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        this.setBackground(new Color( r: 250,  g: 188,  b: 63));

        ImageIcon imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/Employees.png",  width: 50,  height: 50);
        this.addTab( title: "Employees", imageIcon, new SubTabbedPane("Employee", "Role"));

        imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/Items.png",  width: 50,  height: 50);
        this.addTab( title: "Items", imageIcon, new SubTabbedPane("Item", "Category"));

        imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/Inventory.png",  width: 50,  height: 50);
        this.addTab( title: "Inventory", imageIcon, new SubTabbedPane("Inventory", "Transactions"));

        imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/Restaurant.png",  width: 50,  height: 50);
        this.addTab( title: "Restaurant", imageIcon, new RestaurantView());

        imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/Reports.png",  width: 50,  height: 50);
        this.addTab( title: "Reports", imageIcon, new ReportsView());
    }
}
```

SubTabbedPane.java

```java
package view.manager;

import util.CustomImageIcon;

import javax.swing.*;
import java.awt.*;

public class SubTabbedPane extends JTabbedPane {  3 usages  Giorgos-Karachristos

    public SubTabbedPane(String tab1, String tab2) {  3 usages  Giorgos-Karachristos
        this.setBackground(new Color( r: 250,  g: 188,  b: 63));
        this.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        this.putClientProperty("JTabbedPane.tabAreaAlignment", "center");

        addCustomTab(tab1);
        addCustomTab(tab2);
    }

    private void addCustomTab(String tabName) {  2 usages  Giorgos-Karachristos
        ImageIcon imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/" + tabName + ".png",  width: 50,  height: 50);

        CustomTabPanel customPanel = new CustomTabPanel(tabName,  jtp: this);
        this.addTab(tabName, imageIcon, customPanel);
    }
}
```

A.8.9 Menu

DescriptionDialog.java

```java
DescriptionDialog.java
package view.menu;

import javax.swing.*;
import java.awt.*;

public class DescriptionDialog extends JDialog {  2 usages  & Giorgos-Karachristos

    public DescriptionDialog(String itemName, String des, String price) {  1 usage  & G
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle(itemName + " Description");
        this.setLayout(new GridBagLayout());

        JLabel jLabel = new JLabel();
        jLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        jLabel.setText("Price: " + price + "$");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(jLabel, gbc);

        JTextArea descriptionTextArea = new JTextArea();
        descriptionTextArea.setEditable(false);
        descriptionTextArea.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 20));
        descriptionTextArea.setLineWrap(true);
        descriptionTextArea.setWrapStyleWord(true);
        descriptionTextArea.setText(des);

        JScrollPane jScrollPane1 = new JScrollPane(descriptionTextArea);
        jScrollPane1.setPreferredSize(new Dimension( width: 400,  height: 200));

        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 2;
        gbc.insets = new Insets( top: 0,  left: 10,  bottom: 10,  right: 10);
        this.add(jScrollPane1, gbc);

        this.pack();
        this.setLocationRelativeTo(null);
    }
}
```

MenuView.java (1/6)

```java
package view.menu;

import controller.menu.MenuController;
import entities.Category;
import entities.Item;

import javax.swing.*;
import javax.swing.event.TableModelEvent;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MenuView extends JPanel {  3 usages  Giorgos-Karachristos
    private static final int LONG_CLICK_DELAY = 500;  1 usage

    private final DefaultTableModel tableModel;  24 usages
    private final JTable jTable;  8 usages
    private final JLabel numberSubtotalLabel;  7 usages
    private final Map<String, Double> taxMap = new HashMap<>();  3 usages
    private final Map<String, Integer> idMap = new HashMap<>();  2 usages

    private final MenuController menuController;  5 usages
    private Timer longClickTimer;  5 usages
    private boolean longClickTriggered = false;  3 usages

    public MenuView(String name, String surname, JTabbedPane jp) {  1 usage  Gio
        menuController = new MenuController();
        GridBagConstraints gbc;
        this.setLayout(new GridBagLayout());

        JTabbedPane jTabbedPane = new JTabbedPane();
        jTabbedPane.setTabPlacement(JTabbedPane.LEFT);
        jTabbedPane.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        jTabbedPane.setPreferredSize(new Dimension( width: 500,  height: 100));

        List<Category> categoryList = menuController.getCategoryList();
        int tab = 0;
        for (Category category : categoryList) {
            JPanel categoryPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
            categoryPanel.setBackground(Color.WHITE);

            String categoryName = category.getName();
            String categoryColor = category.getColor();
            Double categoryTax = Double.valueOf(category.getTax());

            List<Item> itemData = menuController.getItemsList(categoryName);
            for (Item item : itemData) {
                String itemName = item.getName();
                String itemColor = item.getColor();
                double itemPrice = Double.parseDouble(item.getPrice());
                int itemID = Integer.parseInt(item.getItemId());
                String itemDescription = item.getDescription();
```

MenuView.java (2/6)

```
        taxMap.put(itemName, categoryTax);
        idMap.put(itemName, itemID);

        double priceTax = itemPrice + itemPrice * taxMap.get(itemName) / 100;
        BigDecimal bd = new BigDecimal(priceTax);
        bd = bd.setScale( newScale: 2, RoundingMode.HALF_UP);
        double roundedValue = bd.doubleValue();

        JButton itemButton = new JButton(itemName) {  Giorgos-Karachristos
            @Override  Giorgos-Karachristos
            public Dimension getPreferredSize() {
                Dimension preferredSize = super.getPreferredSize();
                preferredSize.height = 60;
                return preferredSize;
            }
        };
        itemButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        itemButton.setBackground(Color.decode(itemColor));
        itemButton.addMouseListener(new MouseAdapter() {  Giorgos-Karachristos *
            @Override  Giorgos-Karachristos *
            public void mousePressed(MouseEvent e) {
                longClickTriggered = false;
                if (itemDescription != null) {
                    longClickTimer = new Timer(LONG_CLICK_DELAY, (ActionEvent evt) -> {
                        longClickTriggered = true;

                        DescriptionDialog DescriptionDialog = new
                                DescriptionDialog(itemName, itemDescription, String.valueOf(roundedValue));
                        DescriptionDialog.setModal(true);
                        DescriptionDialog.setVisible(true);
                    });
                    longClickTimer.setRepeats(false); // Ensure it only fires once
                    longClickTimer.start();
                }
            }

            @Override  Giorgos-Karachristos
            public void mouseReleased(MouseEvent e) {
                if (longClickTimer != null) {
                    longClickTimer.stop();
                }
                if (!longClickTriggered) {
                    itemButtonActionPerformed(itemName, itemPrice);
                }
            }
        });
        categoryPanel.add(itemButton);
    }

    jTabbedPane.addTab(categoryName, categoryPanel);
    jTabbedPane.setBackgroundAt(tab++, Color.decode(categoryColor));
}

gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridheight = 2;
gbc.fill = GridBagConstraints.BOTH;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.insets = new Insets( top: 100,  left: 0,  bottom: 0,  right: 0);
this.add(jTabbedPane, gbc);

JButton deleteButton = new JButton();
deleteButton.setBackground(new Color( r: 238,  g: 78,  b: 78));
deleteButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
deleteButton.setText("Delete");
deleteButton.addActionListener( ActionEvent evt1 -> deleteButtonActionPerformed());
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 0;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 10;
gbc.ipady = 10;
gbc.anchor = GridBagConstraints.PAGE_START;
gbc.insets = new Insets( top: 100,  left: 10,  bottom: 0,  right: 10);
this.add(deleteButton, gbc);
```

MenuView.java (3/6)

```
JButton deleteAllButton = new JButton();
deleteAllButton.setBackground(new Color( r: 238,  g: 78,  b: 78));
deleteAllButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
deleteAllButton.setText("Delete All");
deleteAllButton.addActionListener((ActionEvent evt) -> deleteAllButtonActionPerformed());
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 1;
gbc.ipadx = 10;
gbc.ipady = 10;
gbc.anchor = GridBagConstraints.PAGE_START;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 0,  right: 10);
this.add(deleteAllButton, gbc);

jTable = new JTable();
jTable.getTableHeader().setReorderingAllowed(false);
jTable.getTableHeader().setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
jTable.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
jTable.setRowHeight(40);

tableModel = new DefaultTableModel(new Object[][]{}, new String[]{"Item Name", "Quantity", "Price (incl. tax)",
        "Total Price (incl. tax)"}) {
    final boolean[] canEdit = new boolean[]{false, true, false, false};  1 usage

    @Override  Giorgos-Karachristos
    public boolean isCellEditable(int rowIndex, int columnIndex) { return canEdit[columnIndex]; }
};
tableModel.addTableModelListener(this::tableChanged);

jTable.setModel(tableModel);

JScrollPane jScrollPane = new JScrollPane();
jScrollPane.setViewportView(jTable);

gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 0;
gbc.gridheight = 2;
gbc.fill = GridBagConstraints.BOTH;
gbc.weightx = 1.0;
gbc.weighty = 1.0;
gbc.insets = new Insets( top: 100,  left: 0,  bottom: 0,  right: 0);
this.add(jScrollPane, gbc);

JLabel subtotalAmountLabel = new JLabel();
subtotalAmountLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
subtotalAmountLabel.setText("Subtotal (incl. tax):");
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 0;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 100,  left: 50,  bottom: 15,  right: 15);
this.add(subtotalAmountLabel, gbc);

numberSubtotalLabel = new JLabel();
numberSubtotalLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
numberSubtotalLabel.setText("0.0");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 0;
gbc.insets = new Insets( top: 100,  left: 15,  bottom: 15,  right: 50);
this.add(numberSubtotalLabel, gbc);

JButton kitchenButton = new JButton();
kitchenButton.addActionListener((ActionEvent evt) -> kitchenButtonActionPerformed(name, surname, jp));
kitchenButton.setBackground(new Color( r: 107,  g: 203,  b: 119));
kitchenButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
kitchenButton.setText("Sent to kitchen");
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 1;
gbc.gridwidth = 2;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipady = 25;
gbc.anchor = GridBagConstraints.PAGE_START;
gbc.insets = new Insets( top: 0,  left: 50,  bottom: 0,  right: 50);
this.add(kitchenButton, gbc);

}
```

MenuView.java (4/6)

```java
private static final int ITEM_NAME_COL = 0;  3 usages
private static final int QUANTITY_COL = 1;  8 usages
private static final int PRICE_COL = 2;  1 usage
private static final int TOTAL_PRICE_COL = 3;  4 usages

private void itemButtonActionPerformed(String itemName, Double itemPrice) {  1 usage  Giorgos-Karachristos
    int row;
    int rowCount = tableModel.getRowCount();
    // Check if the item already exists in the table and increase its quantity by 1 if found
    for (row = 0; row < rowCount; row++) {
        if (tableModel.getValueAt(row, ITEM_NAME_COL).equals(itemName)) {
            int result = 0;
            Object quantity = tableModel.getValueAt(row, QUANTITY_COL);
            if (quantity.getClass() == String.class) {
                result = Integer.parseInt((String) quantity);
            } else if (quantity.getClass() == Integer.class) {
                result = (int) quantity;
            }
            tableModel.setValueAt((result + 1), row, QUANTITY_COL);
            break;
        }
    }
    // If the item doesn't exist in the table, add it as a new row
    if (row == rowCount) {
        double priceTax = itemPrice + itemPrice * taxMap.get(itemName) / 100;

        BigDecimal bd = new BigDecimal(priceTax);
        bd = bd.setScale( newScale: 2, RoundingMode.HALF_UP);
        double roundedValue = bd.doubleValue();

        tableModel.addRow(new Object[]{itemName, null, roundedValue, roundedValue});
        tableModel.setValueAt( aValue: 1,  row: tableModel.getRowCount() - 1, QUANTITY_COL);
    }
}

private void deleteButtonActionPerformed() {  1 usage  Giorgos-Karachristos *
    int row = jTable.getSelectedRow();
    if (row == -1) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Please select an item to delete",
                title: "Info", JOptionPane.INFORMATION_MESSAGE);
        return;
    }
    tableModel.removeRow(row);
    updateSubtotal();
}

private void deleteAllButtonActionPerformed() {  2 usages  Giorgos-Karachristos
    tableModel.setRowCount(0);
    numberSubtotalLabel.setText("0.0");
}
```

MenuView.java (5/6)

```
MenuView.java
private void tableChanged(TableModelEvent e) {  1 usage  Giorgos-Karachristos *
    int row = e.getFirstRow();
    int column = e.getColumn();

    if (column == QUANTITY_COL) {
        double price = Double.parseDouble(tableModel.getValueAt(row, PRICE_COL).toString());
        Object itemName = tableModel.getValueAt(row, ITEM_NAME_COL);
        Object quantityValue = tableModel.getValueAt(row, QUANTITY_COL);
        if (quantityValue.getClass() == String.class) {
            if (quantityValue.toString().isBlank()) {
                JOptionPane.showMessageDialog( parentComponent: this,
                        message: "Quantity must be not empty for the item " + itemName,
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                tableModel.setValueAt( aValue: 0, row, QUANTITY_COL);
                return;
            }
            int quantity;
            try {
                quantity = Integer.parseInt(quantityValue.toString());
                if (quantity <= 0) {
                    JOptionPane.showMessageDialog( parentComponent: this,
                            message: "Quantity must be more than zero for the item " + itemName,
                            title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                    return;
                }
            } catch (NumberFormatException exception) {
                JOptionPane.showMessageDialog( parentComponent: this,
                        message: "Quantity must be number for the item " + itemName,
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                tableModel.setValueAt( aValue: 0, row, QUANTITY_COL);
                return;
            }
            price *= quantity;
        } else if (quantityValue.getClass() == Integer.class) {
            price *= (int) quantityValue;
        }

        BigDecimal bd = new BigDecimal(price);
        bd = bd.setScale( newScale: 2, RoundingMode.HALF_UP);
        double roundedValue = bd.doubleValue();

        tableModel.setValueAt(roundedValue, row, TOTAL_PRICE_COL);
    }
    if (column == TOTAL_PRICE_COL) {
        updateSubtotal();
    }
}

private void updateSubtotal() {  2 usages  Giorgos-Karachristos
    double subtotal = 0.0;
    for (int i = 0; i < tableModel.getRowCount(); i++) {
        subtotal += (Double) tableModel.getValueAt(i, TOTAL_PRICE_COL);
    }
    BigDecimal bd = new BigDecimal(subtotal);
    bd = bd.setScale( newScale: 2, RoundingMode.HALF_UP);
    double roundedValue = bd.doubleValue();
    numberSubtotalLabel.setText(Double.toString(roundedValue));
}
```

MenuView.java (6/6)

```
    private void kitchenButtonActionPerformed(String name, String surname, JTabbedPane jtp) {  1 usage   Giorgos-Karachristc
        int rowCount = tableModel.getRowCount();
        if (rowCount == 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please add items into the order",
                    title: "Error", JOptionPane.WARNING_MESSAGE);
            return;
        }
        String result = menuController.insertOrder( username: name + " " + surname, numberSubtotalLabel.getText(),
                type: "Takeaway");
        if (result.contains("Insertion failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        int failed = 0;
        for (int row = 0; row < rowCount; row++) {
            String itemName = tableModel.getValueAt(row, ITEM_NAME_COL).toString();
            int quantity = Integer.parseInt(tableModel.getValueAt(row, QUANTITY_COL).toString());
            double total = Double.parseDouble(tableModel.getValueAt(row, TOTAL_PRICE_COL).toString());

            result = menuController.insertKitchen(idMap.get(itemName), quantity, total);
            if (result.contains("Insertion failed: ")) {
                JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
                failed++;
            }
        }
        JOptionPane.showMessageDialog( parentComponent: this,  message: rowCount - failed + " items added successfully.\n"
                + failed + " items failed.",  title: "Order Results", JOptionPane.INFORMATION_MESSAGE);
        jtp.setSelectedIndex(1);
        deleteAllButtonActionPerformed();
    }
}
```

A.8.10 Order

OrderPanel.java (1/3)

```
package view.order;

import controller.order.OrderController;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;

public class OrderPanel extends JPanel {  2 usages   Giorgos-Karachristos

    public OrderPanel(String orderName, String orderSubtotal, String orderStatus,String orderType, String orderDateTime) {
        GridBagConstraints gbc;

        this.setBackground(Color.WHITE);
        this.setBorder(BorderFactory.createLineBorder(new Color( r: 0,  g: 0,  b: 0),  thickness: 1));
        this.setLayout(new BorderLayout());

        JPanel topPanel = new JPanel();
        topPanel.setLayout(new GridBagLayout());

        JLabel orderIDLabel = new JLabel();
        orderIDLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        orderIDLabel.setText("Order ID:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 5);
        topPanel.add(orderIDLabel, gbc);

        JLabel sqlOrderIDLabel = new JLabel();
        sqlOrderIDLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        sqlOrderIDLabel.setText(orderName);
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 10,  left: 5,  bottom: 10,  right: 10);
        topPanel.add(sqlOrderIDLabel, gbc);
```

OrderPanel.java (2/3)

```
OrderPanel.java
JLabel subtotalLabel = new JLabel();
subtotalLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
subtotalLabel.setText("Subtotal:");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 0;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 5);
topPanel.add(subtotalLabel, gbc);

JLabel sqlSubtotalLabel = new JLabel();
sqlSubtotalLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
sqlSubtotalLabel.setText(orderSubtotal);
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 0;
gbc.insets = new Insets( top: 10,  left: 5,  bottom: 10,  right: 10);
topPanel.add(sqlSubtotalLabel, gbc);

JLabel statusLabel = new JLabel();
statusLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
statusLabel.setText("Status:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 0;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 5);
topPanel.add(statusLabel, gbc);

JLabel sqlStatusLabel = new JLabel();
sqlStatusLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
sqlStatusLabel.setText(orderStatus);
gbc = new GridBagConstraints();
gbc.gridx = 5;
gbc.gridy = 0;
gbc.insets = new Insets( top: 10,  left: 5,  bottom: 10,  right: 10);
topPanel.add(sqlStatusLabel, gbc);

JLabel typeLabel = new JLabel();
typeLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
typeLabel.setText("Type:");
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 0;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 5);
topPanel.add(typeLabel, gbc);

JLabel sqlTypeLabel = new JLabel();
sqlTypeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
sqlTypeLabel.setText(orderType);
gbc = new GridBagConstraints();
gbc.gridx = 7;
gbc.gridy = 0;
gbc.insets = new Insets( top: 10,  left: 5,  bottom: 10,  right: 10);
topPanel.add(sqlTypeLabel, gbc);

JLabel dateTimeLabel = new JLabel();
dateTimeLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
dateTimeLabel.setText("Date-Time:");
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 0;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 5);
topPanel.add(dateTimeLabel, gbc);

JLabel sqlDateTimeLabel = new JLabel();
sqlDateTimeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
sqlDateTimeLabel.setText(orderDateTime);
gbc = new GridBagConstraints();
gbc.gridx = 9;
gbc.gridy = 0;
gbc.insets = new Insets( top: 10,  left: 5,  bottom: 10,  right: 10);
topPanel.add(sqlDateTimeLabel, gbc);
```

OrderPanel.java (3/3)

```
OrderPanel.java
        this.add(topPanel, BorderLayout.PAGE_START);

        JTable jTable = new JTable();
        jTable.getTableHeader().setReorderingAllowed(false);
        jTable.getTableHeader().setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        jTable.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        jTable.setRowHeight(40);

        DefaultTableModel tableModel = new DefaultTableModel(  Giorgos-Karachristos
                new Object[][]{},
                new String[]{"Item Name", "Quantity", "Status"}
        ) {

            @Override  Giorgos-Karachristos
            public boolean isCellEditable(int rowIndex, int columnIndex) { return false; }
        };

        jTable.setModel(tableModel);

        OrderController orderController = new OrderController();
        orderController.getKitchenList(tableModel, orderName);

        JScrollPane jScrollPane = new JScrollPane();
        jScrollPane.setViewportView(jTable);

        this.add(jScrollPane, BorderLayout.CENTER);
    }
}
```

OrdersView.java (1/4)

```
OrdersView.java
package view.order;

import controller.order.OrderController;
import entities.Order;
import view.register.RegisterView;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.util.List;

public class OrdersView extends JPanel {  3 usages  Giorgos-Karachristos

    private final JTabbedPane jTabbedPane;  19 usages
    private final OrderController orderController;  5 usages
    private final List<Order> orderList;  3 usages

    public OrdersView(JTabbedPane jtp, RegisterView registerView) {  1 usage
        orderController = new OrderController();
        GridBagConstraints gbc;
        this.setLayout(new GridBagLayout());

        jTabbedPane = new JTabbedPane();
        jTabbedPane.setTabPlacement(JTabbedPane.LEFT);
        jTabbedPane.putClientProperty("JTabbedPane.showTabSeparators", true);
        jTabbedPane.setTabLayoutPolicy(JTabbedPane.SCROLL_TAB_LAYOUT);
        jTabbedPane.setBackground(Color.white);
        jTabbedPane.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        jTabbedPane.setPreferredSize(new Dimension( width: 500,  height: 100));

        jTabbedPane.removeAll();
        orderList = orderController.getActiveOrdersList();
        updateTabbedPane(orderList);
```

OrdersView.java (2/4)

```
Orders\View.java
    gbc = new GridBagConstraints();
    gbc.gridx = 0;
    gbc.gridy = 0;
    gbc.gridheight = 5;
    gbc.fill = GridBagConstraints.BOTH;
    gbc.weightx = 1.0;
    gbc.weighty = 1.0;
    gbc.insets = new Insets( top: 100,  left: 0,  bottom: 0,  right: 100);
    this.add(jTabbedPane, gbc);

    JButton voidButton = new JButton();
    voidButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    voidButton.setBackground(new Color( r: 238,  g: 78,  b: 78));
    voidButton.setText("Void Selected Item");
    voidButton.addActionListener((ActionEvent evt) -> voidButtonActionPerformed());
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 0;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 100,  left: 0,  bottom: 15,  right: 100);
    this.add(voidButton, gbc);

    JButton payButton = new JButton();
    payButton.setBackground(new Color( r: 107,  g: 203,  b: 119));
    payButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    payButton.setText("Pay Order");
    payButton.addActionListener((ActionEvent evt) -> payButtonActionPerformed(jtp, registerView));
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 2;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 15,  left: 0,  bottom: 15,  right: 100);
    this.add(payButton, gbc);

    JButton refreshButton = new JButton();
    refreshButton.setBackground(new Color( r: 63,  g: 169,  b: 245));
    refreshButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    refreshButton.setText("Refresh");
    refreshButton.addActionListener((ActionEvent evt) -> refreshButtonActionPerformed( force: true));
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 3;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.insets = new Insets( top: 15,  left: 0,  bottom: 15,  right: 100);
    this.add(refreshButton, gbc);

    JButton cancelButton = new JButton();
    cancelButton.setBackground(new Color( r: 238,  g: 78,  b: 78));
    cancelButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    cancelButton.setText("Cancel Order");
    cancelButton.addActionListener((ActionEvent evt) -> cancelButtonActionPerformed());
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 4;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.anchor = GridBagConstraints.PAGE_START;
    gbc.insets = new Insets( top: 15,  left: 0,  bottom: 0,  right: 100);
    this.add(cancelButton, gbc);
}
```

OrdersView.java (3/4)

```java
Orders\View.java
private void updateTabbedPane(List<Order> orderList) {  2 usages  Giorgos-Karachristos
    OrderPanel op;
    for (Order orders : orderList) {
        String orderID = orders.getOrderID();
        String orderSubtotal = orders.getSubtotal() + " $";
        String orderStatus = orders.getStatus();
        String typeStatus = orders.getType();
        String orderDateTime = orders.getOrderDate();

        op = new OrderPanel(orderID, orderSubtotal, orderStatus, typeStatus, orderDateTime);
        jTabbedPane.addTab( title: "Order " + orderID + " " + orderStatus, op);
    }
}

private void voidButtonActionPerformed() {  1 usage  Giorgos-Karachristos *
    int selectedTabIndex = jTabbedPane.getSelectedIndex();
    if (selectedTabIndex == -1) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "No order selected.",
                title: "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    String[] words = jTabbedPane.getTitleAt(selectedTabIndex).split( regex: " ");
    int orderID = Integer.parseInt(words[1]);

    JPanel selectedPanel = (JPanel) jTabbedPane.getComponentAt(selectedTabIndex);
    JScrollPane scrollPane = (JScrollPane) selectedPanel.getComponent( n: 1);
    JTable jTable1 = (JTable) scrollPane.getViewport().getView();

    int selectedRowIndex = jTable1.getSelectedRow();

    if (selectedRowIndex == -1) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "No item selected in the order.",
                title: "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    String itemName = jTable1.getValueAt(selectedRowIndex,  column: 0).toString();

    String result = orderController.voidKitchen(orderID, itemName);
    if (result.contains("Update failed: ")) {
        JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
    } else {
        JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
    }

    refreshButtonActionPerformed( force: true);
    jTabbedPane.setSelectedIndex(selectedTabIndex);
}

private void payButtonActionPerformed(JTabbedPane jtp, RegisterView r) {  1 usage  Giorgos-Karachristos *
    int selectedTabIndex = jTabbedPane.getSelectedIndex();

    if (selectedTabIndex == -1) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "No order selected.",
                title: "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    String[] words = jTabbedPane.getTitleAt(selectedTabIndex).split( regex: " ");
    int orderID = Integer.parseInt(words[1]);

    jtp.setSelectedIndex(2);
    r.setOrderID(orderID);
}

public void refreshButtonActionPerformed(boolean force) {  4 usages  Giorgos-Karachristos
    int currentSize = orderList.size();
    List<Order> newOrderList = orderController.getActiveOrdersList();

    if (newOrderList.size() != currentSize || force) {
        jTabbedPane.removeAll();
        updateTabbedPane(newOrderList);
    }
}
```

OrdersView.java (4/4)

```
Orders\View.java
    private void cancelButtonActionPerformed() {  1 usage   Giorgos-Karachristos *
        int selectedTabIndex = jTabbedPane.getSelectedIndex();

        if (selectedTabIndex == -1) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "No order selected.",
                    title: "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        String[] words = jTabbedPane.getTitleAt(selectedTabIndex).split( regex: " ");
        int orderID = Integer.parseInt(words[1]);

        String result = orderController.cancelOrder(orderID);
        if (result.contains("Update failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }
        refreshButtonActionPerformed( force: true);
    }
}
```

A.8.11 POS

POSView.java (1/2)

```
POS\View.java
package view.pos;

import com.formdev.flatlaf.FlatLightLaf;
import controller.pos.POSController;
import entities.Employee;
import view.cashier.CashierView;
import view.CustomFrame;
import view.HeaderView;
import view.kitchen.KitchenView;
import view.login.LoginView;
import view.manager.ManagerPane;
import view.waiter.WaiterView;

import javax.swing.*;
import java.awt.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class POSView {  4 usages   Giorgos-Karachristos

    private static final Logger LOGGER = Logger.getLogger(POSView.class.getName());
    private final CustomFrame frame;  14 usages
    private final POSController controller;  3 usages

    public POSView(POSController controller) {  1 usage   Giorgos-Karachristos
        this.controller = controller;
        frame = new CustomFrame();

        // Apply FlatLaf look and feel
        try {
            UIManager.setLookAndFeel(new FlatLightLaf());
        } catch (UnsupportedLookAndFeelException e) {
            LOGGER.log(Level.WARNING,  msg: "UnsupportedLookAndFeelException", e);
        }

        frame.setVisible(true);
    }
```

POSView.java (2/2)

```
POS\view.java
    public void showLoginView() {  2 usages  Giorgos-Karachristos
        frame.getContentPane().removeAll();
        LoginView loginView = new LoginView(controller);
        frame.setLayout(new GridBagLayout());
        GridBagConstraints gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        frame.add(loginView, gbc);
        refresh();
    }

    public void showMainView(Employee employee) {  1 usage  Giorgos-Karachristos
        frame.getContentPane().removeAll();
        frame.setLayout(new BorderLayout());

        HeaderView headerView = new HeaderView(controller, employee.getName(), employee.getSurname());
        frame.add(headerView, BorderLayout.NORTH);

        UIManager.put("TabbedPane.contentAreaColor", Color.BLACK);
        UIManager.put("TabbedPane.selectedBackground", new Color( r: 223,  g: 151,  b: 0));
        UIManager.put("TabbedPane.focusColor", new Color( r: 223,  g: 151,  b: 0));

        switch (employee.getRole()) {
            case "Manager":
                frame.add(new ManagerPane(), BorderLayout.CENTER);
                break;
            case "Cashier":
                frame.add(new CashierView(employee.getName(), employee.getSurname()), BorderLayout.CENTER);
                break;
            case "Kitchen":
                frame.add(new KitchenView(), BorderLayout.CENTER);
                break;
            case "Waiter":
                frame.add(new WaiterView(employee.getName(), employee.getSurname()), BorderLayout.CENTER);
                break;
        }

        refresh();
    }

    private void refresh() {  2 usages  Giorgos-Karachristos
        frame.revalidate();
        frame.repaint();
    }
}
```

A.8.12 Receipts

ReceiptPanel.java (1/5)

```
ReceiptPanel.java
package view.receipts;

import controller.receipts.ReceiptController;
import controller.restaurant.RestaurantController;
import entities.ReceiptItem;

import javax.swing.*;
import java.awt.*;
import java.util.List;

public class ReceiptPanel extends JPanel {  2 usages  & Giorgos-Karachristos *

    public ReceiptPanel(String orderID, String orderName, String orderSubtotal, String orderDiscount,
                        String orderTip, String orderTotal, String date, String time) {
        ReceiptController receiptController = new ReceiptController();
        GridBagConstraints gbc;
        this.setLayout(new GridBagLayout());

        this.setBorder(BorderFactory.createLineBorder(new Color( r: 0,  g: 0,  b: 0),  thickness: 2));

        JPanel infoPanel = new JPanel();
        infoPanel.setBackground(new Color( r: 255,  g: 255,  b: 255));
        infoPanel.setLayout(new GridBagLayout());

        RestaurantController restaurantController = new RestaurantController();
        String[] res = restaurantController.getRestaurant();

        JLabel restaurantNameLabel = new JLabel();
        restaurantNameLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 25));
        restaurantNameLabel.setText(res[0]);
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 4;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        infoPanel.add(restaurantNameLabel, gbc);

        JLabel addressLabel = new JLabel();
        addressLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        addressLabel.setText(res[1]);
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 4;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        infoPanel.add(addressLabel, gbc);

        JLabel phoneLabel = new JLabel();
        phoneLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        phoneLabel.setText("Phone:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.gridwidth = 2;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
        infoPanel.add(phoneLabel, gbc);

        JLabel phoneValueLabel = new JLabel();
        phoneValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        phoneValueLabel.setText(res[2]);
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 2;
        gbc.gridwidth = 3;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
        infoPanel.add(phoneValueLabel, gbc);
```

ReceiptPanel.java (2/5)

ReceiptPanel.java

```
JLabel receiptLabel = new JLabel();
receiptLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
receiptLabel.setText("Receipt #:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 3;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
infoPanel.add(receiptLabel, gbc);

JLabel receiptIDLabel = new JLabel();
receiptIDLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
receiptIDLabel.setText(orderID);
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 3;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
infoPanel.add(receiptIDLabel, gbc);

JLabel cashierLabel = new JLabel();
cashierLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
cashierLabel.setText("Cashier:");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 3;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
infoPanel.add(cashierLabel, gbc);

JLabel cashierValueLabel = new JLabel();
cashierValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
cashierValueLabel.setText(orderName);
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 3;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
infoPanel.add(cashierValueLabel, gbc);

JLabel dateLabel = new JLabel();
dateLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
dateLabel.setText("Date:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
infoPanel.add(dateLabel, gbc);

JLabel dateValueLabel = new JLabel();
dateValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
dateValueLabel.setText(date);
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
infoPanel.add(dateValueLabel, gbc);
```

ReceiptPanel.java (3/5)

```
ReceiptPanel.java
JLabel timeLabel = new JLabel();
timeLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
timeLabel.setText("Time:");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
infoPanel.add(timeLabel, gbc);

JLabel timeValueLabel = new JLabel();
timeValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
timeValueLabel.setText(time);
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
infoPanel.add(timeValueLabel, gbc);

gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 0;
gbc.weightx = 1.0;
gbc.weighty = 0.1;
gbc.fill = GridBagConstraints.BOTH;
this.add(infoPanel, gbc);
//--------------------------------------------------------------------------------

JPanel itemsPanel = new JPanel();
itemsPanel.setLayout(new GridBagLayout());
itemsPanel.setBackground(new Color( r: 255,  g: 255,  b: 255));

JLabel itemLabel = new JLabel();
itemLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
itemLabel.setText("Item");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 0;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
itemsPanel.add(itemLabel, gbc);

JLabel qtyLabel = new JLabel();
qtyLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
qtyLabel.setText("Qty");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 0;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
itemsPanel.add(qtyLabel, gbc);

JLabel taxLabel = new JLabel();
taxLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
taxLabel.setText("Tax");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 0;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
itemsPanel.add(taxLabel, gbc);

JLabel totalPriceLabel = new JLabel();
totalPriceLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
totalPriceLabel.setText("Total Price");
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 0;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
itemsPanel.add(totalPriceLabel, gbc);
```

ReceiptPanel.java (4/5)

```
ReceiptPanel.java
JLabel timeLabel = new JLabel();
timeLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
timeLabel.setText("Time:");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
infoPanel.add(timeLabel, gbc);

JLabel timeValueLabel = new JLabel();
timeValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
timeValueLabel.setText(time);
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
infoPanel.add(timeValueLabel, gbc);

gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 0;
gbc.weightx = 1.0;
gbc.weighty = 0.1;
gbc.fill = GridBagConstraints.BOTH;
this.add(infoPanel, gbc);
//--------------------------------------------------------------------------------

JPanel itemsPanel = new JPanel();
itemsPanel.setLayout(new GridBagLayout());
itemsPanel.setBackground(new Color( r: 255,  g: 255,  b: 255));

JLabel itemLabel = new JLabel();
itemLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
itemLabel.setText("Item");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 0;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
itemsPanel.add(itemLabel, gbc);

JLabel qtyLabel = new JLabel();
qtyLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
qtyLabel.setText("Qty");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 0;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
itemsPanel.add(qtyLabel, gbc);

JLabel taxLabel = new JLabel();
taxLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
taxLabel.setText("Tax");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 0;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
itemsPanel.add(taxLabel, gbc);

JLabel totalPriceLabel = new JLabel();
totalPriceLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
totalPriceLabel.setText("Total Price");
gbc = new GridBagConstraints();
gbc.gridx = 3;
gbc.gridy = 0;
gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
itemsPanel.add(totalPriceLabel, gbc);
```

ReceiptPanel.java (5/5)

```
ReceiptPanel.java
        JLabel tipLabel = new JLabel();
        tipLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        tipLabel.setText("Tip");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        summaryPanel.add(tipLabel, gbc);

        JLabel tipValueLabel = new JLabel();
        tipValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        tipValueLabel.setText(orderTip + " $");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 1;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        summaryPanel.add(tipValueLabel, gbc);

        JLabel discountLabel = new JLabel();
        discountLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        discountLabel.setText("Discount");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        summaryPanel.add(discountLabel, gbc);

        JLabel discountValueLabel = new JLabel();
        discountValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        discountValueLabel.setText(orderDiscount + " $");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 2;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        summaryPanel.add(discountValueLabel, gbc);

        JLabel totalLabel = new JLabel();
        totalLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        totalLabel.setText("Total");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 3;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        summaryPanel.add(totalLabel, gbc);

        JLabel totalValueLabel = new JLabel();
        totalValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        totalValueLabel.setText(orderTotal + " $");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 3;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        summaryPanel.add(totalValueLabel, gbc);

        JLabel thanksLabel = new JLabel();
        thanksLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        thanksLabel.setText("Thank you for dining with us!");
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 4;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        summaryPanel.add(thanksLabel, gbc);

        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.weighty = 0.1;
        gbc.fill = GridBagConstraints.BOTH;
        this.add(summaryPanel, gbc);
    }
}
```

ReceiptsView.java (1/3)

```java
package view.receipts;

import controller.receipts.ReceiptController;
import entities.Order;
import util.ExportPanelToImage;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ReceiptsView extends JPanel {  4 usages  & Giorgos-Karachristos

    private static final Logger LOGGER = Logger.getLogger(ReceiptsView.class.getName());  1
    private final JTabbedPane jTabbedPane;  19 usages
    private final ReceiptController receiptController;  4 usages
    private final List<Order> orderList;  3 usages

    public ReceiptsView() {  1 usage  & Giorgos-Karachristos
        receiptController = new ReceiptController();
        GridBagConstraints gbc;
        this.setLayout(new GridBagLayout());

        jTabbedPane = new JTabbedPane();
        jTabbedPane.setTabPlacement(JTabbedPane.LEFT);
        jTabbedPane.putClientProperty("JTabbedPane.showTabSeparators", true);
        jTabbedPane.setTabLayoutPolicy(JTabbedPane.SCROLL_TAB_LAYOUT);
        jTabbedPane.setBackground(Color.white);
        jTabbedPane.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        jTabbedPane.setPreferredSize(new Dimension( width: 650,  height: 100));

        jTabbedPane.removeAll();
        orderList = receiptController.getFinalizedOrdersList();
        updateTabbedPane(orderList);

        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridheight = 5;
        gbc.fill = GridBagConstraints.BOTH;
        gbc.weighty = 1.0;
        gbc.insets = new Insets( top: 100,  left: 0,  bottom: 0,  right: 100);
        this.add(jTabbedPane, gbc);

        JButton refundButton = new JButton();
        refundButton.setBackground(new Color( r: 238,  g: 78,  b: 78));
        refundButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        refundButton.setText("Refund Order");
        refundButton.addActionListener((ActionEvent evt) -> refundButtonActionPerformed());
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 0;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 100;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 100,  left: 0,  bottom: 15,  right: 100);
        this.add(refundButton, gbc);
```

ReceiptsView.java (2/3)

```
    JButton saveReceiptButton = new JButton();
    saveReceiptButton.setBackground(new Color( r: 63,  g: 169,  b: 245));
    saveReceiptButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    saveReceiptButton.setText("Save Receipt");
    saveReceiptButton.addActionListener((ActionEvent evt) -> saveReceiptButtonActionPerformed());
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 1;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.anchor = GridBagConstraints.PAGE_START;
    gbc.insets = new Insets( top: 15,  left: 0,  bottom: 15,  right: 100);
    this.add(saveReceiptButton, gbc);

    JButton refreshButton = new JButton();
    refreshButton.setBackground(new Color( r: 63,  g: 169,  b: 245));
    refreshButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    refreshButton.setText("Refresh");
    refreshButton.addActionListener((ActionEvent evt) -> refreshButtonActionPerformed( force: true));
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 2;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.anchor = GridBagConstraints.PAGE_START;
    gbc.insets = new Insets( top: 15,  left: 0,  bottom: 15,  right: 100);
    this.add(refreshButton, gbc);
}

private void updateTabbedPane(List<Order> orderList) {  2 usages  & Giorgos-Karachristos
    ReceiptPanel rp;
    for (Order orders : orderList) {
        String orderID = orders.getOrderID();
        String orderName = orders.getName();
        String orderSubtotal = orders.getSubtotal();
        String orderDiscount = orders.getDiscount();
        String orderTip = orders.getTip();
        String orderTotal = orders.getTotal();
        String date = orders.getOrderDate().substring(0, 11);
        String time = orders.getOrderDate().substring( beginIndex: 11);
        String status = orders.getStatus();

        rp = new ReceiptPanel(orderID, orderName, orderSubtotal, orderDiscount, orderTip, orderTotal, date, time);

        JScrollPane jScrollPane = new JScrollPane();
        jScrollPane.setViewportView(rp);

        if ("Refunt".equals(status)) {
            JPanel jPanel = new JPanel();
            jPanel.setLayout(new OverlayLayout(jPanel));

            JLabel refundLabel = new JLabel();
            refundLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 70));
            refundLabel.setForeground(new java.awt.Color( r: 204,  g: 0,  b: 51));
            refundLabel.setText("Refund");
            refundLabel.setAlignmentX(0.5F);
            refundLabel.setOpaque(false);
            jPanel.add(refundLabel);

            jPanel.add(jScrollPane);

            jTabbedPane.addTab( title: "Order " + orderID + " Refund", jPanel);
        } else {
            jTabbedPane.addTab( title: "Order " + orderID, jScrollPane);
        }
    }
}
```

ReceiptsView.java (3/3)

```
ReceiptsView.java
    private void refundButtonActionPerformed() {  1 usage  Giorgos-Karachristos *
        int selectedTabIndex = jTabbedPane.getSelectedIndex();

        if (selectedTabIndex == -1) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "No order selected.",
                    title: "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }

        String[] words = jTabbedPane.getTitleAt(selectedTabIndex).split( regex: " ");
        int orderID = Integer.parseInt(words[1]);

        String result = receiptController.refundOrder(orderID);
        if (result.contains("Update failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }
        refreshButtonActionPerformed( force: true);
        jTabbedPane.setSelectedIndex(selectedTabIndex);
    }

    private void saveReceiptButtonActionPerformed() {  1 usage  Giorgos-Karachristos *
        int selectedTabIndex = jTabbedPane.getSelectedIndex();
        if (selectedTabIndex == -1) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "No order selected.",
                    title: "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        String[] words = jTabbedPane.getTitleAt(selectedTabIndex).split( regex: " ");

        JPanel selectedPanel;
        try{
            JScrollPane scrollPane = (JScrollPane) jTabbedPane.getComponentAt(selectedTabIndex);
            selectedPanel = (JPanel) scrollPane.getViewport().getView();
        } catch (Exception e){
            selectedPanel = (JPanel) jTabbedPane.getComponentAt(selectedTabIndex);
        }


        try {
            ExportPanelToImage.exportPanelToImage(selectedPanel,  fileName: "receipt" + words[1] + ".png");
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Image file exported successfully!",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
        } catch (Exception ex) {
            LOGGER.log(Level.WARNING,  msg: "ExportPanelToImage Exception", ex);
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Failed to export image file.",
                    title: "Error", JOptionPane.WARNING_MESSAGE);
        }
    }

    public void refreshButtonActionPerformed(boolean force) {  3 usages  Giorgos-Karachristos
        int currentSize = orderList.size();
        List<Order> newOrderList = receiptController.getFinalizedOrdersList();

        if (newOrderList.size() != currentSize || force) {
            jTabbedPane.removeAll();
            updateTabbedPane(newOrderList);
        }
    }
}
```

A.8.13 Register

ItemsPanel.java (1/2)

ItemsPanel.java

```
package view.register;

import controller.register.RegisterController;
import entities.ReceiptItem;

import javax.swing.*;
import java.awt.*;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.List;

public class ItemsPanel extends JPanel {  2 usages  & Giorgos-Karachristos

    public ItemsPanel(String receiptID, RegisterView registerView, SummaryPanel summaryPanel) {
        RegisterController registerController = new RegisterController();
        GridBagConstraints gbc;
        this.setLayout(new GridBagLayout());
        this.setBackground(new Color( r: 255,  g: 255,  b: 255));

        JLabel itemLabel = new JLabel();
        itemLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        itemLabel.setText("Item");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
        this.add(itemLabel, gbc);

        JLabel qtyLabel = new JLabel();
        qtyLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        qtyLabel.setText("Qty");
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
        this.add(qtyLabel, gbc);

        JLabel taxLabel = new JLabel();
        taxLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        taxLabel.setText("Tax");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
        this.add(taxLabel, gbc);

        JLabel totalPriceLabel = new JLabel();
        totalPriceLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        totalPriceLabel.setText("Total Price");
        gbc = new GridBagConstraints();
        gbc.gridx = 3;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
        this.add(totalPriceLabel, gbc);
```

ItemsPanel.java (2/2)

ItemsPanel.java

```
        int gridy = 1;
        if (!"Receipt".equals(receiptID) && !"".equals(receiptID)) {
            List<ReceiptItem> receiptItemList = registerController.getItemsList(receiptID);
            double totalPrice = 0;
            for (ReceiptItem receiptItem : receiptItemList) {
                JLabel itemValueLabel = new JLabel();
                itemValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
                itemValueLabel.setText(receiptItem.getName());
                gbc = new GridBagConstraints();
                gbc.gridx = 0;
                gbc.gridy = gridy;
                gbc.insets = new Insets( top: 5,  left: 10,  bottom: 5,  right: 5);
                this.add(itemValueLabel, gbc);

                JLabel qtyValueLabel = new JLabel();
                qtyValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
                qtyValueLabel.setText(String.valueOf(receiptItem.getQuantity()));
                gbc = new GridBagConstraints();
                gbc.gridx = 1;
                gbc.gridy = gridy;
                gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
                this.add(qtyValueLabel, gbc);

                JLabel taxValueLabel = new JLabel();
                taxValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
                taxValueLabel.setText(receiptItem.getTax() + " %");
                gbc = new GridBagConstraints();
                gbc.gridx = 2;
                gbc.gridy = gridy;
                gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
                this.add(taxValueLabel, gbc);

                JLabel totalPriceValueLabel = new JLabel();
                totalPriceValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
                totalPriceValueLabel.setText(receiptItem.getTotalPrice());
                gbc = new GridBagConstraints();
                gbc.gridx = 3;
                gbc.gridy = gridy;
                gbc.insets = new Insets( top: 5,  left: 5,  bottom: 5,  right: 5);
                this.add(totalPriceValueLabel, gbc);

                totalPrice = totalPrice + Double.parseDouble(receiptItem.getTotalPrice());
                gridy++;
                this.repaint();
            }
            BigDecimal bd = new BigDecimal(totalPrice);
            bd = bd.setScale( newScale: 2, RoundingMode.HALF_UP);
            double roundedValue = bd.doubleValue();

            registerView.setSubtotalTextField(String.valueOf(roundedValue));
            summaryPanel.updateSubtotalValueLabel(String.valueOf(roundedValue));
        }
    }
}
```

Receipt.java (1/3)

```java
Receipt.java
package view.register;

import controller.restaurant.RestaurantController;

import javax.swing.*;
import java.awt.*;

public class Receipt extends JPanel {  2 usages  Giorgos-Karachristos

    private final JLabel receiptIDLabel;  4 usages
    private final JLabel cashierValueLabel;  4 usages
    private final JLabel dateValueLabel;  4 usages
    private final JLabel timeValueLabel;  4 usages

    private ItemsPanel itemsPanel;  3 usages
    private SummaryPanel summaryPanel;  5 usages

    public Receipt(RegisterView registerView) {  1 usage  Giorgos-Karachristos

        GridBagConstraints gridBagConstraints;
        this.setLayout(new GridBagLayout());

        this.setBorder(BorderFactory.createLineBorder(new Color( r: 0,  g: 0,  b: 0),  thickness: 2));

        JPanel infoPanel = new JPanel();
        infoPanel.setBackground(new Color( r: 255,  g: 255,  b: 255));
        infoPanel.setLayout(new GridBagLayout());

        RestaurantController restaurantController = new RestaurantController();
        String[] res = restaurantController.getRestaurant();

        JLabel restaurantNameLabel = new JLabel();
        restaurantNameLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 25));
        restaurantNameLabel.setText(res[0]);
        gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.gridwidth = 4;
        gridBagConstraints.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        infoPanel.add(restaurantNameLabel, gridBagConstraints);

        JLabel addressLabel = new JLabel();
        addressLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        addressLabel.setText(res[1]);
        gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.gridwidth = 4;
        gridBagConstraints.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        infoPanel.add(addressLabel, gridBagConstraints);

        JLabel phoneLabel = new JLabel();
        phoneLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        phoneLabel.setText("Phone:");
        gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.gridwidth = 2;
        gridBagConstraints.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
        infoPanel.add(phoneLabel, gridBagConstraints);

        JLabel phoneValueLabel = new JLabel();
        phoneValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        phoneValueLabel.setText(res[2]);
        gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 1;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.gridwidth = 3;
        gridBagConstraints.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
        infoPanel.add(phoneValueLabel, gridBagConstraints);
```

Receipt.java (2/3)

```
Receipt.java
JLabel receiptLabel = new JLabel();
receiptLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
receiptLabel.setText("Receipt #:");
gridBagConstraints = new GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = GridBagConstraints.LINE_START;
gridBagConstraints.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
infoPanel.add(receiptLabel, gridBagConstraints);

receiptIDLabel = new JLabel();
receiptIDLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gridBagConstraints = new GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = GridBagConstraints.LINE_START;
gridBagConstraints.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
infoPanel.add(receiptIDLabel, gridBagConstraints);

JLabel cashierLabel = new JLabel();
cashierLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
cashierLabel.setText("Cashier:");
gridBagConstraints = new GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = GridBagConstraints.LINE_START;
gridBagConstraints.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
infoPanel.add(cashierLabel, gridBagConstraints);

cashierValueLabel = new JLabel();
cashierValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gridBagConstraints = new GridBagConstraints();
gridBagConstraints.gridx = 3;
gridBagConstraints.gridy = 3;
gridBagConstraints.anchor = GridBagConstraints.LINE_START;
gridBagConstraints.insets = new Insets( top: 15,  left: 10,  bottom: 10,  right: 10);
infoPanel.add(cashierValueLabel, gridBagConstraints);

JLabel dateLabel = new JLabel();
dateLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
dateLabel.setText("Date:");
gridBagConstraints = new GridBagConstraints();
gridBagConstraints.gridx = 0;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = GridBagConstraints.LINE_START;
gridBagConstraints.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
infoPanel.add(dateLabel, gridBagConstraints);

dateValueLabel = new JLabel();
dateValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gridBagConstraints = new GridBagConstraints();
gridBagConstraints.gridx = 1;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = GridBagConstraints.LINE_START;
gridBagConstraints.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
infoPanel.add(dateValueLabel, gridBagConstraints);

JLabel timeLabel = new JLabel();
timeLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
timeLabel.setText("Time:");
gridBagConstraints = new GridBagConstraints();
gridBagConstraints.gridx = 2;
gridBagConstraints.gridy = 4;
gridBagConstraints.anchor = GridBagConstraints.LINE_START;
gridBagConstraints.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
infoPanel.add(timeLabel, gridBagConstraints);
```

Receipt.java (3/3)

```
Receipt.java
        timeValueLabel = new JLabel();
        timeValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 3;
        gridBagConstraints.gridy = 4;
        gridBagConstraints.anchor = GridBagConstraints.LINE_START;
        gridBagConstraints.insets = new Insets( top: 10,  left: 10,  bottom: 15,  right: 10);
        infoPanel.add(timeValueLabel, gridBagConstraints);

        gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 0;
        gridBagConstraints.weightx = 1.0;
        gridBagConstraints.weighty = 0.1;
        gridBagConstraints.fill = GridBagConstraints.BOTH;
        this.add(infoPanel, gridBagConstraints);

        addItemsSummaryPanels( receiptID: "Receipt", registerView);
    }

    void updateInfoValues(String receiptID, String cashierID, String date, String time, RegisterView registerView) {
        receiptIDLabel.setText(receiptID);
        cashierValueLabel.setText(cashierID);
        dateValueLabel.setText(date);
        timeValueLabel.setText(time);
        addItemsSummaryPanels(receiptID, registerView);
    }

    private void addItemsSummaryPanels(String receiptID, RegisterView registerView) {  2 usages   Giorgos-Karachristos
        if (!"Receipt".equals(receiptID)) {
            this.remove(itemsPanel);
            this.remove(summaryPanel);
        }
        GridBagConstraints gridBagConstraints;

        summaryPanel = new SummaryPanel();
        itemsPanel = new ItemsPanel(receiptID, registerView, summaryPanel);
        gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 1;
        gridBagConstraints.weighty = 0.8;
        gridBagConstraints.fill = GridBagConstraints.BOTH;
        this.add(itemsPanel, gridBagConstraints);

        gridBagConstraints = new GridBagConstraints();
        gridBagConstraints.gridx = 0;
        gridBagConstraints.gridy = 2;
        gridBagConstraints.weighty = 0.1;
        gridBagConstraints.fill = GridBagConstraints.BOTH;
        this.add(summaryPanel, gridBagConstraints);

        this.repaint();
    }

    void updateSummaryValues(String subtotal, String tip, String discount, String total) {  1 usage   Giorgos-Karachristos
        summaryPanel.updateSummaryValues(subtotal, tip, discount, total);
    }
}
```

RegisterView.java (1/6)

```java
package view.register;

import controller.register.RegisterController;
import util.ExportPanelToImage;

import javax.swing.*;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.logging.Level;
import java.util.logging.Logger;

public class RegisterView extends JPanel {  11 usages  & Giorgos-Karachristos

    private static final Logger LOGGER = Logger.getLogger(RegisterView.class.getName());
    private int orderID = -1;  6 usages
    private final Receipt receipt;  6 usages
    private final JTextField subtotalTextField;  12 usages
    private final JTextField tipTextField;  11 usages
    private final JTextField discountTextField;  11 usages
    private final JTextField totalTextField;  11 usages
    private final JTextField receivedTextField;  9 usages
    private final JLabel changeValueLabel;  6 usages
    private final RegisterController registerController;  4 usages

    public RegisterView(JTabbedPane jtp) {  1 usage  & Giorgos-Karachristos
        registerController = new RegisterController();

        GridBagConstraints gbc;

        this.setLayout(new GridBagLayout());

        JLabel orderReceiptLabel = new JLabel();
        orderReceiptLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        orderReceiptLabel.setText("Order Receipt");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 100,  left: 0,  bottom: 0,  right: 200);
        this.add(orderReceiptLabel, gbc);

        receipt = new Receipt( registerView: this);
        JScrollPane jScrollPane = new JScrollPane();
        jScrollPane.setPreferredSize(new Dimension( width: 450,  height: 250));
        jScrollPane.setViewportView(receipt);

        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridheight = 9;
        gbc.fill = GridBagConstraints.BOTH;
        gbc.weighty = 1.0;
        gbc.insets = new Insets( top: 0,  left: 0,  bottom: 0,  right: 200);
        this.add(jScrollPane, gbc);

        JLabel subtotalLabel = new JLabel();
        subtotalLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        subtotalLabel.setText("Subtotal");
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 1;
        gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 15);
        this.add(subtotalLabel, gbc);

        subtotalTextField = new JTextField();
        subtotalTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        subtotalTextField.setHorizontalAlignment(JTextField.CENTER);
        subtotalTextField.setText("0.0");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 1;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 100);
        this.add(subtotalTextField, gbc);
```

RegisterView.java (2/6)

```
JLabel tipLabel = new JLabel();
tipLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
tipLabel.setText("Tip");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 2;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 15);
this.add(tipLabel, gbc);

tipTextField = new JTextField();
tipTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
tipTextField.setHorizontalAlignment(JTextField.CENTER);
tipTextField.setText("0.0");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 2;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 100);
this.add(tipTextField, gbc);

JLabel discountLabel = new JLabel();
discountLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
discountLabel.setText("Discount");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 3;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 15);
this.add(discountLabel, gbc);

discountTextField = new JTextField();
discountTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
discountTextField.setHorizontalAlignment(JTextField.CENTER);
discountTextField.setText("0.0");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 3;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 100);
this.add(discountTextField, gbc);

JLabel totalLabel = new JLabel();
totalLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
totalLabel.setText("Total");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 4;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 15);
this.add(totalLabel, gbc);

totalTextField = new JTextField();
totalTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
totalTextField.setHorizontalAlignment(JTextField.CENTER);
totalTextField.setText("0.0");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 4;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 100);
this.add(totalTextField, gbc);

subtotalTextField.getDocument().addDocumentListener(totalCalculatorListener());
tipTextField.getDocument().addDocumentListener(totalCalculatorListener());
discountTextField.getDocument().addDocumentListener(totalCalculatorListener());
```

RegisterView.java (3/6)

```
JLabel receivedAmountLabel = new JLabel();
receivedAmountLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
receivedAmountLabel.setText("Received amount");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 5;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 15);
this.add(receivedAmountLabel, gbc);

receivedTextField = new JTextField();
receivedTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
receivedTextField.setHorizontalAlignment(JTextField.CENTER);
receivedTextField.setText("0.0");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 5;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 150;
gbc.ipady = 10;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 100);
this.add(receivedTextField, gbc);

JLabel changeLabel = new JLabel();
changeLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
changeLabel.setText("Change");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 6;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 15);
this.add(changeLabel, gbc);

changeValueLabel = new JLabel();
changeValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
changeValueLabel.setText("0.0");
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 6;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 100);
this.add(changeValueLabel, gbc);

receivedTextField.getDocument().addDocumentListener(changeCalculatorListener());

JButton saveReceiptButton = new JButton();
saveReceiptButton.setBackground(new Color( r: 63,  g: 169,  b: 245));
saveReceiptButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
saveReceiptButton.setText("Save Receipt");
saveReceiptButton.addActionListener((ActionEvent evt) -> saveReceiptButtonActionPerformed());
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 7;
gbc.gridwidth = 2;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 100);
this.add(saveReceiptButton, gbc);

JButton payOrderButton = new JButton();
payOrderButton.setBackground(new Color( r: 107,  g: 203,  b: 119));
payOrderButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
payOrderButton.setText("Pay Order");
payOrderButton.addActionListener((ActionEvent evt) -> payOrderButtonActionPerformed(orderID, jtp));
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 8;
gbc.gridwidth = 2;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 100);
this.add(payOrderButton, gbc);
```

RegisterView.java (4/6)

```
RegisterView.java
    JButton cancelOrderButton = new JButton();
    cancelOrderButton.setBackground(new Color( r: 238,  g: 78,  b: 78));
    cancelOrderButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    cancelOrderButton.setText("Cancel Order");
    cancelOrderButton.addActionListener((ActionEvent evt) -> cancelOrderButtonActionPerformed(jtp));
    gbc = new GridBagConstraints();
    gbc.gridx = 1;
    gbc.gridy = 9;
    gbc.gridwidth = 2;
    gbc.fill = GridBagConstraints.HORIZONTAL;
    gbc.ipadx = 100;
    gbc.ipady = 10;
    gbc.anchor = GridBagConstraints.PAGE_END;
    gbc.insets = new Insets( top: 15,  left: 15,  bottom: 15,  right: 100);
    this.add(cancelOrderButton, gbc);

}

public void setOrderID(int order_ID) {  Giorgos-Karachristos
    orderID = order_ID;
    if (order_ID == -1) {
        receipt.updateInfoValues( receiptID: "",  cashierID: "",  date: "",  time: "",  registerView: this);
        setTextFieldText();
    } else {
        String[] details = registerController.getOrder(order_ID);
        receipt.updateInfoValues(String.valueOf(order_ID), details[0], details[1], details[2],  registerView: this);
    }
}

private void setTextFieldText() {  1 usage  Giorgos-Karachristos
    subtotalTextField.setText("0.0");
    tipTextField.setText("0.0");
    discountTextField.setText("0.0");
    totalTextField.setText("0.0");
    receivedTextField.setText("0.0");
    changeValueLabel.setText("0.0");
}

private DocumentListener totalCalculatorListener() {  3 usages  Giorgos-Karachristos
    return new DocumentListener() {  Giorgos-Karachristos
        @Override  Giorgos-Karachristos
        public void changedUpdate(DocumentEvent e) {
            calculateTotal();
        }

        @Override  Giorgos-Karachristos

        public void removeUpdate(DocumentEvent e) {
            calculateTotal();
        }

        @Override  Giorgos-Karachristos
        public void insertUpdate(DocumentEvent e) {
            calculateTotal();
        }
```

RegisterView.java (5/6)

```
RegisterView.java
        public void calculateChange() {  3 usages  Giorgos-Karachristos
            double total = 0;
            double received = 0;
            String totalText = totalTextField.getText().isBlank() ? "0" : totalTextField.getText();
            String receivedText = receivedTextField.getText().isBlank() ? "0" : receivedTextField.getText();
            if (isValid(totalText,  type: "Total") && isValid(receivedText,  type: "Received")) {
                total = Double.parseDouble(totalText);
                received = Double.parseDouble(receivedText);
            }
            double change = received - total;

            BigDecimal bd = new BigDecimal(change);
            bd = bd.setScale( newScale: 2, RoundingMode.HALF_UP);
            double roundedValue = bd.doubleValue();

            changeValueLabel.setText(String.valueOf(roundedValue));
        }
    };
}

private void saveReceiptButtonActionPerformed() {  1 usage  Giorgos-Karachristos *
    if (orderID == -1) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "No order selected.",
                title: "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }
    try {
        ExportPanelToImage.exportPanelToImage(receipt,  fileName: "receipt" + orderID + ".png");
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Image file exported successfully!",
                title: "Info", JOptionPane.INFORMATION_MESSAGE);
    } catch (Exception ex) {
        LOGGER.log(Level.WARNING,  msg: "ExportPanelToImage Exception", ex);
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Failed to export image file.",
                title: "Error", JOptionPane.WARNING_MESSAGE);
    }
}

private void payOrderButtonActionPerformed(int orderID, JTabbedPane jtp) {  1 usage  Giorgos-Karachristos *
    if (orderID == -1) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "No order selected.",
                title: "Error", JOptionPane.ERROR_MESSAGE);
        return;
    }

    String subtotal = subtotalTextField.getText();
    String discount = discountTextField.getText();
    String tip = tipTextField.getText();
    String total = totalTextField.getText();
    if (isValid(subtotal,  type: "Subtotal") && isValid(discount,  type: "Discount")
            && isValid(tip,  type: "Tip") && isValid(total,  type: "Total")) {
        String result = registerController.completeOrder(orderID, Double.parseDouble(subtotal),
                Double.parseDouble(discount), Double.parseDouble(tip), Double.parseDouble(total));
        if (result.contains("Update failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }
        jtp.setSelectedIndex(0);
    }
}
```

RegisterView.java (6/6)

```
RegisterView.java
    private boolean isValid(String price, String type) {  9 usages  Giorgos-Karachristos *
        if (price.isBlank()) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: type + " must be not empty",
                    title: "Validation Error", JOptionPane.WARNING_MESSAGE);
            return false;
        }
        double doublePrice;
        try {
            doublePrice = Double.parseDouble(price);
            if (doublePrice < 0) {
                JOptionPane.showMessageDialog( parentComponent: this,  message: type + " must be more than zero",
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                return false;
            }
        } catch (NumberFormatException exception) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: type + " must be number",
                    title: "Validation Error", JOptionPane.WARNING_MESSAGE);
            return false;
        }
        return true;
    }

    public void setSubtotalTextField(String text) { subtotalTextField.setText(text); }

    private void cancelOrderButtonActionPerformed(JTabbedPane jtp) {  1 usage  Giorgos-Karachristos *
        if (orderID == -1) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "No order selected.",
                    title: "Error", JOptionPane.ERROR_MESSAGE);
            return;
        }
        String result = registerController.cancelOrder(orderID);
        if (result.contains("Update failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }
        jtp.setSelectedIndex(0);
    }
}
```

SummaryPanel.java (1/3)

```
SummaryPanel.java
package view.register;

import javax.swing.*;
import java.awt.*;

public class SummaryPanel extends JPanel {  3 usages  Giorgos-Karachristos

    private final JLabel subtotalValueLabel;  6 usages
    private final JLabel tipValueLabel;  5 usages
    private final JLabel discountValueLabel;  5 usages
    private final JLabel totalValueLabel;  5 usages

    public SummaryPanel() {  1 usage  Giorgos-Karachristos
        GridBagConstraints gbc;
        this.setLayout(new GridBagLayout());
        this.setBackground(new Color( r: 255,  g: 255,  b: 255));

        JLabel subtotalLabel = new JLabel();
        subtotalLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        subtotalLabel.setText("Subtotal");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        this.add(subtotalLabel, gbc);
```

SummaryPanel.java (2/3)

```
Summary Panel.java
        subtotalValueLabel = new JLabel();
        subtotalValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        subtotalValueLabel.setText("0.0 $");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        this.add(subtotalValueLabel, gbc);

        JLabel tipLabel = new JLabel();
        tipLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        tipLabel.setText("Tip");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        this.add(tipLabel, gbc);

        tipValueLabel = new JLabel();
        tipValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        tipValueLabel.setText("0.0 $");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 1;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        this.add(tipValueLabel, gbc);

        JLabel discountLabel = new JLabel();
        discountLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        discountLabel.setText("Discount");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        this.add(discountLabel, gbc);

        discountValueLabel = new JLabel();
        discountValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        discountValueLabel.setText("0.0 $");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 2;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        this.add(discountValueLabel, gbc);

        JLabel totalLabel = new JLabel();
        totalLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 16));
        totalLabel.setText("Total");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 3;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        this.add(totalLabel, gbc);

        totalValueLabel = new JLabel();
        totalValueLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        totalValueLabel.setText("0.0 $");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 3;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        this.add(totalValueLabel, gbc);

        JLabel thanksLabel = new JLabel();
        thanksLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        thanksLabel.setText("Thank you for dining with us!");
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 4;
        gbc.insets = new Insets( top: 15,  left: 10,  bottom: 15,  right: 10);
        this.add(thanksLabel, gbc);
    }
```

SummaryPanel.java (3/3)

```
SummaryPanel.java
    void updateSummaryValues(String subtotal, String tip, String discount, String total) {
        subtotalValueLabel.setText(subtotal + " $");
        tipValueLabel.setText(tip + " $");
        discountValueLabel.setText(discount + " $");
        totalValueLabel.setText(total + " $");
        this.repaint();
    }

    void updateSubtotalValueLabel(String subtotal) {  1 usage  Giorgos-Karachristos
        subtotalValueLabel.setText(subtotal + " $");
        this.repaint();
    }
}
```

A.8.14 Reports

ItemsReport.java (1/2)

```
ItemsReport.java
package view.reports;

import controller.reports.ReportsController;
import util.CallPython;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;

public class ItemsReport extends JPanel {  2 usages  Giorgos-Karachristos

    private final JLabel inventoryGraphLabel;  4 usages
    private final JLabel magicQuadrantGraphLabel;  4 usages

    public ItemsReport() {  1 usage  Giorgos-Karachristos
        this.setLayout(new BorderLayout());

        JPanel topPanel = new JPanel();
        topPanel.setLayout(new FlowLayout(FlowLayout.CENTER,  hgap: 10,  vgap: 10));
        topPanel.setBackground(new Color( r: 250,  g: 188,  b: 63));

        JButton refreshButton = new JButton();
        refreshButton.setBackground(new Color( r: 63,  g: 169,  b: 245));
        refreshButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        refreshButton.setText("Refresh");
        topPanel.add(refreshButton);

        this.add(topPanel, BorderLayout.PAGE_START);

        JPanel jPanel = new JPanel();
        jPanel.setBackground(new Color( r: 250,  g: 188,  b: 63));
        jPanel.setLayout(new GridBagLayout());
```

ItemsReport.java (2/2)

```
ItemsReport.java
        GridBagConstraints gbc;

        JLabel inventoryLabel = new JLabel();
        inventoryLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        inventoryLabel.setText("Inventory Graph:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        jPanel.add(inventoryLabel, new GridBagConstraints());

        inventoryGraphLabel = new JLabel();
        inventoryGraphLabel.setName("inventoryGraphLabel");
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 1;
        gbc.gridwidth = 2;
        jPanel.add(inventoryGraphLabel, gbc);

        JLabel magicQuadrantLabel = new JLabel();
        magicQuadrantLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        magicQuadrantLabel.setText("Magic Quadrant:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        jPanel.add(magicQuadrantLabel, gbc);

        magicQuadrantGraphLabel = new JLabel();
        magicQuadrantGraphLabel.setName("magicQuadrant");
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 3;
        jPanel.add(magicQuadrantGraphLabel, gbc);

        JTable jTable = new JTable();
        jTable.getTableHeader().setReorderingAllowed(false);
        jTable.getTableHeader().setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        jTable.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        jTable.setRowHeight(40);
        jTable.setModel(new DefaultTableModel(new Object[][]{}, new String[]{"No.", "Item"}) {  Giorgos-
            @Override  Giorgos-Karachristos
            public boolean isCellEditable(int rowIndex, int columnIndex) {
                return false;
            }
        });
        JScrollPane jScrollPane1 = new JScrollPane();
        jScrollPane1.setViewportView(jTable);
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 3;
        gbc.fill = GridBagConstraints.VERTICAL;
        jPanel.add(jScrollPane1, gbc);

        JScrollPane jScrollPane2 = new JScrollPane();
        jScrollPane2.setViewportView(jPanel);
        jScrollPane2.setBorder(BorderFactory.createEmptyBorder());
        jScrollPane2.getVerticalScrollBar().setUnitIncrement(20);
        this.add(jScrollPane2, BorderLayout.CENTER);

        DefaultTableModel tableModel = (DefaultTableModel) jTable.getModel();
        refreshButton.addActionListener((ActionEvent evt) -> refreshButtonActionPerformed(tableModel));
        loadGraphs(tableModel);
    }

    private void refreshButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos
        this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
        loadGraphs(tableModel);
        this.revalidate();
        this.repaint();
        this.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
    }

    private void loadGraphs(DefaultTableModel tableModel) {  2 usages  Giorgos-Karachristos *
        ReportsController reportsController = new ReportsController();
        CallPython cp = new CallPython();

        String[] inventoryData = reportsController.getInventoryGraph();
        cp.callPythonBarGraph(inventoryData[0], inventoryData[1],  xTitle: "Item Name",  yTitle: "Quantity",
                title: "Inventory Graph", inventoryGraphLabel);

        String[] itemsData = reportsController.getItemsGraph(tableModel);
        cp.callPythonGraph(itemsData[0], itemsData[1], itemsData[2], magicQuadrantGraphLabel);
    }
}
```

ReportsView.java

```java
package view.reports;

import util.CustomImageIcon;
import view.CustomFrame;

import javax.swing.*;
import java.awt.*;

public class ReportsView extends JTabbedPane {  2 usages  Giorgos-Karachristos

    public ReportsView() {  1 usage  Giorgos-Karachristos
        this.setBackground(new Color( r: 250,  g: 188,  b: 63));
        this.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        this.putClientProperty("JTabbedPane.tabAreaAlignment", "center");

        SalesReport sales = new SalesReport();
        ImageIcon imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/SalesReport.png",  width: 50,  height: 50);
        this.addTab( title: "Sales", imageIcon, sales);

        ItemsReport items = new ItemsReport();
        imageIcon = CustomImageIcon.getScaledImageIcon( imagePath: "resources/ItemsReport.png",  width: 50,  height: 50);
        this.addTab( title: "Items", imageIcon, items);
    }
}
```

SalesReport.java (1/4)

```java
package view.reports;

import controller.reports.ReportsController;
import util.CallPython;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

public class SalesReport extends JPanel {  2 usages  Giorgos-Karachristos

    private final ReportsController reportsController;  8 usages
    private final JLabel grossSalesGraphLabel;  4 usages
    private final JLabel refundsGraphLabel;  4 usages
    private final JLabel discountsGraphLabel;  4 usages
    private final JLabel netSalesGraphLabel;  4 usages
    private final JLabel grossProfitGraphLabel;  4 usages

    public SalesReport() {  1 usage  Giorgos-Karachristos
        reportsController = new ReportsController();
        this.setLayout(new BorderLayout());

        JPanel topPanel = new JPanel();
        topPanel.setLayout(new FlowLayout(FlowLayout.CENTER,  hgap: 10,  vgap: 10));
        topPanel.setBackground(new Color( r: 250,  g: 188,  b: 63));

        JButton refreshButton = new JButton();
        refreshButton.setBackground(new Color( r: 63,  g: 169,  b: 245));
        refreshButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        refreshButton.setText("Refresh");
        refreshButton.addActionListener((ActionEvent evt) -> refreshButtonActionPerformed());
        topPanel.add(refreshButton);

        this.add(topPanel, BorderLayout.PAGE_START);

        String[] string = reportsController.getSalesLabels();
```

SalesReport.java (2/4)

SalesReport.java

```
JPanel jPanel = new JPanel();
jPanel.setBackground(new Color( r: 250,  g: 188,  b: 63));
jPanel.setLayout(new GridBagLayout());

JPanel jPanel1 = new JPanel();
jPanel1.setBackground(new Color( r: 250,  g: 188,  b: 63));
jPanel1.setLayout(new FlowLayout(FlowLayout.CENTER,  hgap: 10,  vgap: 10));

JLabel grossSalesLabel = new JLabel();
grossSalesLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
grossSalesLabel.setText("Gross Sales:");
jPanel1.add(grossSalesLabel);

JLabel sqlGrossSalesLabel = new JLabel();
sqlGrossSalesLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 20));
sqlGrossSalesLabel.setText(string[0]);
jPanel1.add(sqlGrossSalesLabel);

JLabel refundsLabel = new JLabel();
refundsLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
refundsLabel.setText("Refunds:");
jPanel1.add(refundsLabel);

String[] str = reportsController.getRefundsLabel();

JLabel sqlRefundsLabel = new JLabel();
sqlRefundsLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 20));
sqlRefundsLabel.setText(str[0]);
jPanel1.add(sqlRefundsLabel);

JLabel discountsLabel = new JLabel();
discountsLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
discountsLabel.setText("Discounts:");
jPanel1.add(discountsLabel);

JLabel sqlDiscountsLabel = new JLabel();
sqlDiscountsLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 20));
sqlDiscountsLabel.setText(string[1]);
jPanel1.add(sqlDiscountsLabel);

JLabel netSalesLabel = new JLabel();
netSalesLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
netSalesLabel.setText("Net Sales:");
jPanel1.add(netSalesLabel);

JLabel sqlNetSalesLabel = new JLabel();
sqlNetSalesLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 20));
sqlNetSalesLabel.setText(string[2]);
jPanel1.add(sqlNetSalesLabel);

JLabel grossProfitLabel = new JLabel();
grossProfitLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
grossProfitLabel.setText("Gross Profit:");
jPanel1.add(grossProfitLabel);

JLabel sqlGrossProfitLabel = new JLabel();
sqlGrossProfitLabel.setFont(new Font( name: "Segoe UI", Font.BOLD,  size: 20));
sqlGrossProfitLabel.setText(string[3]);
jPanel1.add(sqlGrossProfitLabel);

GridBagConstraints gbc;

gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 0;
gbc.gridwidth = 2;
jPanel.add(jPanel1, gbc);
```

SalesReport.java (3/4)

```
SalesReport.java
JLabel grossSalesLabel1 = new JLabel();
grossSalesLabel1.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
grossSalesLabel1.setText("Gross Sales:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 1;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
jPanel.add(grossSalesLabel1, gbc);

grossSalesGraphLabel = new JLabel();
grossSalesGraphLabel.setName("grossSalesGraph");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 2;
jPanel.add(grossSalesGraphLabel, gbc);

JLabel refundsLabel1 = new JLabel();
refundsLabel1.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
refundsLabel1.setText("Refunds:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 3;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
jPanel.add(refundsLabel1, gbc);

refundsGraphLabel = new JLabel();
refundsGraphLabel.setName("refundsGraph");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 4;
jPanel.add(refundsGraphLabel, gbc);

JLabel discountsLabel1 = new JLabel();
discountsLabel1.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
discountsLabel1.setText("Discounts:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 5;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
jPanel.add(discountsLabel1, gbc);

discountsGraphLabel = new JLabel();
discountsGraphLabel.setName("discountsGraph");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 6;
jPanel.add(discountsGraphLabel, gbc);

JLabel netSalesLabel1 = new JLabel();
netSalesLabel1.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
netSalesLabel1.setText("Net Sales:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 7;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
jPanel.add(netSalesLabel1, gbc);

netSalesGraphLabel = new JLabel();
netSalesGraphLabel.setName("netSalesGraph");
gbc = new GridBagConstraints();
gbc.gridx = 1;
gbc.gridy = 8;
jPanel.add(netSalesGraphLabel, gbc);
```

SalesReport.java (4/4)

SalesReport.java

```
        JLabel grossProfitLabel1 = new JLabel();
        grossProfitLabel1.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        grossProfitLabel1.setText("Gross Profit:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 9;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        jPanel.add(grossProfitLabel1, gbc);

        grossProfitGraphLabel = new JLabel();
        grossProfitGraphLabel.setName("grossProfitGraph");
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 10;
        jPanel.add(grossProfitGraphLabel, gbc);

        JScrollPane jScrollPane = new JScrollPane();
        jScrollPane.setBorder(BorderFactory.createEmptyBorder());
        jScrollPane.getVerticalScrollBar().setUnitIncrement(20);
        jScrollPane.setViewportView(jPanel);

        this.add(jScrollPane, BorderLayout.CENTER);

        loadGraphs();
    }

    private void refreshButtonActionPerformed() {  1 usage   Giorgos-Karachristos
        this.setCursor(Cursor.getPredefinedCursor(Cursor.WAIT_CURSOR));
        loadGraphs();
        this.revalidate();
        this.repaint();
        this.setCursor(Cursor.getPredefinedCursor(Cursor.DEFAULT_CURSOR));
    }

    private void loadGraphs() {  2 usages   Giorgos-Karachristos
        CallPython cp = new CallPython();

        String[] data = reportsController.getGrossSalesGraph();
        cp.callPythonLineGraph(data[0], data[1],  xTitle: "Date",  yTitle: "Subtotal",  title: "Gross Sales", grossSalesGraphLabel);

        data = reportsController.getRefundsGraph();
        cp.callPythonLineGraph(data[0], data[1],  xTitle: "Date",  yTitle: "Total",  title: "Refunds", refundsGraphLabel);

        data = reportsController.getDiscountsGraph();
        cp.callPythonLineGraph(data[0], data[1],  xTitle: "Date",  yTitle: "Discount",  title: "Discounts", discountsGraphLabel);

        data = reportsController.getNetSalesGraph();
        cp.callPythonLineGraph(data[0], data[1],  xTitle: "Date",  yTitle: "Net Sales",  title: "Net Sales", netSalesGraphLabel);

        data = reportsController.getGrossProfitGraph();
        cp.callPythonLineGraph(data[0], data[1],  xTitle: "Date",  yTitle: "Gross Profit",  title: "Gross Profit", grossProfitGraphLabel);
    }
}
```

A.8.15 Restaurant

RestaurantView.java (1/2)

```
RestaurantView.java
package view.restaurant;

import controller.restaurant.RestaurantController;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;

public class RestaurantView extends JPanel {  2 usages  Giorgos-Karachristos

    private final JTextField nameTextField;  6 usages
    private final JTextField addressTextField;  6 usages
    private final JTextField phoneTextField;  6 usages
    private final JTextField tablesTextField;  6 usages
    private final RestaurantController restaurantController;  3 usages

    public RestaurantView() {  1 usage  Giorgos-Karachristos
        restaurantController = new RestaurantController();
        GridBagConstraints gbc;
        this.setLayout(new GridBagLayout());
        this.setBackground(new Color( r: 250,  g: 188,  b: 63));

        JLabel nameLabel = new JLabel();
        nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        nameLabel.setText("Name:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(nameLabel, gbc);

        String[] res = restaurantController.getRestaurant();

        nameTextField = new JTextField();
        nameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        nameTextField.setText(res[0]);
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 0;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(nameTextField, gbc);

        JLabel addressLabel = new JLabel();
        addressLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        addressLabel.setText("Address:");
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(addressLabel, gbc);

        addressTextField = new JTextField();
        addressTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        addressTextField.setText(res[1]);
        gbc = new GridBagConstraints();
        gbc.gridx = 3;
        gbc.gridy = 0;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(addressTextField, gbc);

        JLabel phoneLabel = new JLabel();
        phoneLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        phoneLabel.setText("Phone:");
        gbc = new GridBagConstraints();
        gbc.gridx = 4;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(phoneLabel, gbc);
```

RestaurantView.java (2/2)

```
RestaurantView.java
        phoneTextField = new JTextField();
        phoneTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN, size: 16));
        phoneTextField.setText(res[2]);
        gbc = new GridBagConstraints();
        gbc.gridx = 5;
        gbc.gridy = 0;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 10, left: 10, bottom: 10, right: 10);
        this.add(phoneTextField, gbc);

        JLabel tableLabel = new JLabel();
        tableLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN, size: 20));
        tableLabel.setText("Tables:");
        gbc = new GridBagConstraints();
        gbc.gridx = 6;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 10, left: 10, bottom: 10, right: 10);
        this.add(tableLabel, gbc);

        tablesTextField = new JTextField();
        tablesTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN, size: 16));
        tablesTextField.setText(res[3]);
        gbc = new GridBagConstraints();
        gbc.gridx = 7;
        gbc.gridy = 0;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 10, left: 10, bottom: 10, right: 10);
        this.add(tablesTextField, gbc);

        JButton saveButton = new JButton();
        saveButton.setBorder(BorderFactory.createEmptyBorder());
        saveButton.setBackground(new Color( r: 107, g: 203, b: 119));
        saveButton.setFont(new Font( name: "Segoe UI", Font.PLAIN, size: 20));
        saveButton.setText("Save");
        saveButton.addActionListener((ActionEvent evt) -> saveButtonActionPerformed());
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 1;
        gbc.gridwidth = 4;
        gbc.fill = java.awt.GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 10, left: 10, bottom: 10, right: 10);
        this.add(saveButton, gbc);
    }

    private void saveButtonActionPerformed() {  1 usage   Giorgos-Karachristos *
        if (!nameTextField.getText().isBlank()
                && !addressTextField.getText().isBlank()
                && !phoneTextField.getText().isBlank()
                && !tablesTextField.getText().isBlank()) {

            String result = restaurantController.updateRestaurant(
                    nameTextField.getText(),
                    addressTextField.getText(),
                    phoneTextField.getText(),
                    Integer.parseInt(tablesTextField.getText()));
            if (result.contains("Update failed: ")) {
                JOptionPane.showMessageDialog( parentComponent: this, result, title: "Error", JOptionPane.ERROR_MESSAGE);
            } else {
                JOptionPane.showMessageDialog( parentComponent: this, result, title: "Info", JOptionPane.INFORMATION_MESSAGE);
            }
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, message: "All the fields are required",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }
    }
}
```

A.8.16 Role

DeleteRoleView.java (1/3)

DeleteRoleView.java

```java
package view.role;

import controller.role.RoleController;
import entities.Role;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class DeleteRoleView extends JDialog {  2 usages  & Giorgos-Karachristos

    private final JComboBox<String> nameComboBox;  9 usages
    private final JLabel sqlAccessLabel;  5 usages
    private final JTextArea jTextArea;  12 usages
    private final DefaultComboBoxModel<String> nameBoxModel;  2 usages
    private final RoleController roleController;  4 usages

    private List<Role> roleList;  2 usages

    public DeleteRoleView(DefaultTableModel tableModel) {  1 usage  & Giorgos-Karachristos
        roleController = new RoleController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Delete Role");
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0, 10, 0};
        this.setLayout(layout);

        JLabel nameLabel = new JLabel();
        nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        nameLabel.setText("Name:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 20,  left: 20,  bottom: 10,  right: 10);
        this.add(nameLabel, gbc);

        nameComboBox = new JComboBox<>();
        nameComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        nameBoxModel = (DefaultComboBoxModel<String>) nameComboBox.getModel();
        populateComboBox();
        nameComboBox.addActionListener( ActionEvent evt1 -> nameComboBoxActionPerformed());
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 20,  left: 10,  bottom: 10,  right: 20);
        this.add(nameComboBox, gbc);

        JLabel accessLabel = new JLabel();
        accessLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        accessLabel.setText("Access:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.insets = new Insets( top: 10,  left: 20,  bottom: 10,  right: 10);
        this.add(accessLabel, gbc);
```

DeleteRoleView.java (2/3)

```
DeleteRoleView.java
        sqlAccessLabel = new JLabel();
        sqlAccessLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
        this.add(sqlAccessLabel, gbc);

        jTextArea = new JTextArea();
        jTextArea.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        jTextArea.setWrapStyleWord(true);
        jTextArea.setLineWrap(true);
        jTextArea.setOpaque(false);
        jTextArea.setEditable(false);
        jTextArea.setFocusable(false);
        jTextArea.setBorder(null);
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.gridwidth = 3;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.insets = new Insets( top: 10,  left: 20,  bottom: 10,  right: 20);
        this.add(jTextArea, gbc);

        JButton deleteButton = new JButton();
        deleteButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        deleteButton.setText("Delete");
        deleteButton.addActionListener((ActionEvent evt) -> deleteButtonActionPerformed(tableModel));
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 6;
        gbc.gridwidth = 3;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 10,  left: 20,  bottom: 20,  right: 20);
        this.add(deleteButton, gbc);

        addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });

        this.pack();
        this.setLocationRelativeTo(null);
    }

    private void formWindowClosing() {  2 usages   Giorgos-Karachristos
        nameComboBox.setSelectedIndex(0);
        sqlAccessLabel.setText("");
        jTextArea.setText("");
    }
```

DeleteRoleView.java (3/3)

```
DeleteRoleView.java
    private void nameComboBoxActionPerformed() {  1 usage   Giorgos-Karachristos *
        for (Role role: roleList) {
            if (role.getName().equals(nameComboBox.getSelectedItem())) {
                sqlAccessLabel.setText(role.getPosAccess());
                if (Integer.parseInt(role.getCountEmployees()) > 0) {
                    jTextArea.setText("Deleting this role will affect " + role.getCountEmployees() +
                            " employees. Their roles will be set to empty.");
                } else {
                    jTextArea.setText("");
                }
                this.pack();
                break;
            }
        }
    }

    private void deleteButtonActionPerformed(DefaultTableModel tableModel) {  1 usage   Giorgos-Karachristos *
        if (nameComboBox.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Please select a role to delete",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        String result = roleController.deleteRole((String) nameComboBox.getSelectedItem());
        if (result.contains("Delete failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        roleController.getRoleTable(tableModel);
        populateComboBox();
        formWindowClosing();
    }

    @Override   Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { roleList = roleController.getRoleList(nameBoxModel); }
}
```

InsertRoleView.java (1/2)

```java
package view.role;

import controller.role.RoleController;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class InsertRoleView extends JDialog {  2 usages  & Giorgos-Karachristos

    private final JTextField nameTextField;  5 usages
    private final JComboBox<String> accessComboBox;  7 usages
    private final RoleController roleController;  4 usages

    public InsertRoleView(DefaultTableModel tableModel) {  1 usage  & Giorgos-Karachristos
        roleController = new RoleController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Add Role");
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0};
        this.setLayout(layout);

        JLabel nameLabel = new JLabel();
        nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        nameLabel.setText("Name:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 20,  left: 20,  bottom: 10,  right: 10);
        this.add(nameLabel, gbc);

        nameTextField = new JTextField();
        nameTextField.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 20,  left: 10,  bottom: 10,  right: 20);
        this.add(nameTextField, gbc);

        JLabel accessLabel = new JLabel();
        accessLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        accessLabel.setText("Access:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.insets = new Insets( top: 10,  left: 20,  bottom: 10,  right: 10);
        this.add(accessLabel, gbc);

        accessComboBox = new JComboBox<>();
        accessComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        DefaultComboBoxModel<String> boxModel = (DefaultComboBoxModel<String>) accessComboBox.getModel();
        roleController.getPOSAccess(boxModel);
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 2;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 20);
        this.add(accessComboBox, gbc);
```

InsertRoleView.java (2/2)

```
InsertRoleView.java
        JButton submitButton = new JButton();
        submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        submitButton.setText("Submit");
        submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(tableModel));
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.gridwidth = 3;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 10,  left: 20,  bottom: 20,  right: 20);
        this.add(submitButton, gbc);

        addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });

        this.pack();
        this.setLocationRelativeTo(null);
    }

    private void formWindowClosing() {  2 usages  Giorgos-Karachristos
        nameTextField.setText("");
        accessComboBox.setSelectedIndex(0);
    }

    private void submitButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
        String roleName = nameTextField.getText();
        if (roleName.isBlank() || accessComboBox.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "All the fields are required",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        String result = roleController.insertRole(roleName, (String) accessComboBox.getSelectedItem());
        if (result.contains("Insertion failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        roleController.getRoleTable(tableModel);
        formWindowClosing();
    }
}
```

UpdateRoleView.java (1/2)

```java
package view.role;

import controller.role.RoleController;
import entities.Role;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class UpdateRoleView extends JDialog {  2 usages  Giorgos-Karachristos

    private final JComboBox<String> nameComboBox;  9 usages
    private final JComboBox<String> accessComboBox;  10 usages
    private final DefaultComboBoxModel<String> nameBoxModel;  2 usages
    private final RoleController roleController;  5 usages

    private List<Role> roleList;  2 usages

    public UpdateRoleView(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos
        roleController = new RoleController();
        GridBagConstraints gbc;

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Edit Role");
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0};
        this.setLayout(layout);

        JLabel nameLabel = new JLabel();
        nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        nameLabel.setText("Name:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.insets = new Insets( top: 20,  left: 20,  bottom: 10,  right: 10);
        this.add(nameLabel, gbc);

        nameComboBox = new JComboBox<>();
        nameComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        nameBoxModel = (DefaultComboBoxModel<String>) nameComboBox.getModel();
        populateComboBox();
        nameComboBox.addActionListener( ActionEvent e -> nameComboBoxActionPerformed());
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 20,  left: 10,  bottom: 10,  right: 20);
        this.add(nameComboBox, gbc);

        JLabel accessLabel = new JLabel();
        accessLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        accessLabel.setText("Access:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 2;
        gbc.insets = new Insets( top: 10,  left: 20,  bottom: 10,  right: 10);
        this.add(accessLabel, gbc);

        accessComboBox = new JComboBox<>();
        accessComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        DefaultComboBoxModel<String> accessBoxModel = (DefaultComboBoxModel<String>) accessComboBox.getModel();
        roleController.getPOSAccess(accessBoxModel);
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 2;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 20);
        this.add(accessComboBox, gbc);
```

UpdateRoleView.java (2/2)

```
UpdateRoleView.java
        JButton submitButton = new JButton();
        submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        submitButton.setText("Submit");
        submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(tableModel));
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 4;
        gbc.gridwidth = 3;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 150;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 10,  left: 20,  bottom: 20,  right: 20);
        this.add(submitButton, gbc);

        addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });

        this.pack();
        this.setLocationRelativeTo(null);
    }

    private void formWindowClosing() {  2 usages  Giorgos-Karachristos
        nameComboBox.setSelectedIndex(0);
        accessComboBox.setSelectedIndex(0);
    }

    private void nameComboBoxActionPerformed() {  1 usage  Giorgos-Karachristos
        for (Role role: roleList) {
            if (role.getName().equals(nameComboBox.getSelectedItem())) {
                String roleName = role.getPosAccess();
                for (int x = 0; x <= accessComboBox.getItemCount(); x++) {
                    if (roleName.equals(accessComboBox.getItemAt(x))) {
                        accessComboBox.setSelectedIndex(x);
                        break;
                    }
                }
                break;
            }
        }
    }

    private void submitButtonActionPerformed(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos *
        if (nameComboBox.getSelectedIndex() == 0 || accessComboBox.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "All the fields are required",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }

        String result = roleController.updateRole((String) nameComboBox.getSelectedItem(),
                (String) accessComboBox.getSelectedItem());
        if (result.contains("Update failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        roleController.getRoleTable(tableModel);
        populateComboBox();
        formWindowClosing();
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { roleList = roleController.getRoleList(nameBoxModel); }
}
```

A.8.17 Transaction

DeleteTransactionView.java (1/5)

DeleteTransaction\view.java

```
package view.transaction;

import controller.transaction.TransactionController;
import entities.TransactionItem;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class DeleteTransactionView extends JDialog {  2 usages  Giorgos-Karachristos *

    private final JComboBox<String>transactionIDComboBox;  9 usages
    private final JLabel sqlItemIDLabel;  5 usages
    private final JLabel sqlCodeLabel;  5 usages
    private final JLabel sqlNameLabel;  5 usages
    private final JLabel sqlOldQuantityLabel;  5 usages
    private final JLabel sqlTypeLabel;  5 usages
    private final JLabel sqlTransactionQuantityLabel;  5 usages
    private final JLabel sqlDateLabel;  5 usages
    private final JLabel sqlStatusLabel;  5 usages
    private final DefaultComboBoxModel<String> transactionBoxModel;  2 usages
    private final TransactionController transactionController;  4 usages

    private List<TransactionItem> rowData;  2 usages

    public DeleteTransactionView(DefaultTableModel tableModel) {  1 usage  Giorgos-Karachristos
        transactionController = new TransactionController();

        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Delete Transaction");
        this.setResizable(false);

        GridBagConstraints gbc;
        GridBagLayout layout = new GridBagLayout();
        layout.columnWidths = new int[]{0, 10, 0, 10, 0, 10, 0, 10, 0, 10, 0};
        layout.rowHeights = new int[]{0, 10, 0, 10, 0, 10, 0};
        getContentPane().setLayout(layout);

        JLabel transactionIDLabel = new JLabel();
        transactionIDLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        transactionIDLabel.setText("Transaction ID:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.anchor = GridBagConstraints.LINE_START;
        gbc.insets = new Insets( top: 20,  left: 20,  bottom: 10,  right: 10);
        getContentPane().add(transactionIDLabel, gbc);

        transactionIDComboBox = new JComboBox<>();
        transactionIDComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
        transactionBoxModel = (DefaultComboBoxModel<String>) transactionIDComboBox.getModel();
        populateComboBox();
        transactionIDComboBox.addActionListener( ActionEvent evt1 -> comboBoxActionPerformed());
        gbc = new GridBagConstraints();
        gbc.gridx = 2;
        gbc.gridy = 0;
        gbc.gridwidth = 3;
        gbc.fill = GridBagConstraints.HORIZONTAL;
        gbc.ipadx = 100;
        gbc.ipady = 10;
        gbc.insets = new Insets( top: 20,  left: 10,  bottom: 10,  right: 10);
        getContentPane().add(transactionIDComboBox, gbc);
```

DeleteTransactionView.java (2/5)

```
DeleteTransaction\dao.java
JButton deleteButton = new JButton();
deleteButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
deleteButton.setText("Delete");
deleteButton.addActionListener((ActionEvent evt) -> deleteButtonActionPerformed(tableModel));
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 0;
gbc.gridwidth = 3;
gbc.fill = GridBagConstraints.HORIZONTAL;
gbc.ipadx = 100;
gbc.ipady = 10;
gbc.insets = new Insets( top: 20,  left: 10,  bottom: 10,  right: 20);
getContentPane().add(deleteButton, gbc);

JLabel itemIDLabel = new JLabel();
itemIDLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
itemIDLabel.setText("Item ID:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 2;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 20,  bottom: 10,  right: 10);
getContentPane().add(itemIDLabel, gbc);

sqlItemIDLabel = new JLabel();
sqlItemIDLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 2;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
getContentPane().add(sqlItemIDLabel, gbc);

JLabel codeLabel = new JLabel();
codeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
codeLabel.setText("Item Code:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 2;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
getContentPane().add(codeLabel, gbc);

sqlCodeLabel = new JLabel();
sqlCodeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 2;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
getContentPane().add(sqlCodeLabel, gbc);

JLabel nameLabel = new JLabel();
nameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
nameLabel.setText("Name:");
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 2;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
getContentPane().add(nameLabel, gbc);

sqlNameLabel = new JLabel();
sqlNameLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 10;
gbc.gridy = 2;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 20);
getContentPane().add(sqlNameLabel, gbc);

JLabel oldQuantityLabel = new JLabel();
oldQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
oldQuantityLabel.setText("Old Quantity:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 20,  bottom: 10,  right: 10);
getContentPane().add(oldQuantityLabel, gbc);
```

DeleteTransactionView.java (3/5)

```
DeleteTransaction\view.java
sqlOldQuantityLabel = new JLabel();
sqlOldQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 4;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
getContentPane().add(sqlOldQuantityLabel, gbc);

JLabel typeLabel = new JLabel();
typeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
typeLabel.setText("Type:");
gbc = new GridBagConstraints();
gbc.gridx = 4;
gbc.gridy = 4;
gbc.anchor = GridBagConstraints.LINE_START;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
getContentPane().add(typeLabel, gbc);

sqlTypeLabel = new JLabel();
sqlTypeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 6;
gbc.gridy = 4;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
getContentPane().add(sqlTypeLabel, gbc);

JLabel transactionQuantityLabel = new JLabel();
transactionQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
transactionQuantityLabel.setText("Transaction Quantity:");
gbc = new GridBagConstraints();
gbc.gridx = 8;
gbc.gridy = 4;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 10);
getContentPane().add(transactionQuantityLabel, gbc);

sqlTransactionQuantityLabel = new JLabel();
sqlTransactionQuantityLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 10;
gbc.gridy = 4;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 10,  right: 20);
getContentPane().add(sqlTransactionQuantityLabel, gbc);

JLabel dateLabel = new JLabel();
dateLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
dateLabel.setText("Transaction Date:");
gbc = new GridBagConstraints();
gbc.gridx = 0;
gbc.gridy = 6;
gbc.insets = new Insets( top: 10,  left: 20,  bottom: 20,  right: 10);
getContentPane().add(dateLabel, gbc);

sqlDateLabel = new JLabel();
sqlDateLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
gbc = new GridBagConstraints();
gbc.gridx = 2;
gbc.gridy = 6;
gbc.insets = new Insets( top: 10,  left: 10,  bottom: 20,  right: 10);
getContentPane().add(sqlDateLabel, gbc);
```

DeleteTransactionView.java (4/5)

```
DeleteTransaction\view.java
    JLabel statusLabel = new JLabel();
    statusLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    statusLabel.setText("Status:");
    gbc = new GridBagConstraints();
    gbc.gridx = 8;
    gbc.gridy = 6;
    gbc.anchor = GridBagConstraints.LINE_START;
    gbc.insets = new Insets( top: 10,  left: 10,  bottom: 20,  right: 10);
    getContentPane().add(statusLabel, gbc);

    sqlStatusLabel = new JLabel();
    sqlStatusLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    gbc = new GridBagConstraints();
    gbc.gridx = 10;
    gbc.gridy = 6;
    gbc.insets = new Insets( top: 10,  left: 10,  bottom: 20,  right: 20);
    getContentPane().add(sqlStatusLabel, gbc);

    this.addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });
    pack();
    setLocationRelativeTo(null);
}

private void formWindowClosing() {  2 usages  Giorgos-Karachristos
    transactionIDComboBox.setSelectedIndex(0);
    sqlItemIDLabel.setText("");
    sqlCodeLabel.setText("");
    sqlNameLabel.setText("");
    sqlOldQuantityLabel.setText("");
    sqlTypeLabel.setText("");
    sqlTransactionQuantityLabel.setText("");
    sqlDateLabel.setText("");
    sqlStatusLabel.setText("");
}

private void comboBoxActionPerformed() {  1 usage  Giorgos-Karachristos
    for (TransactionItem transactionItem : rowData) {
        if (transactionItem.getTransactionId().equals(transactionIDComboBox.getSelectedItem())) {
            sqlItemIDLabel.setText(transactionItem.getItemId());
            sqlCodeLabel.setText(transactionItem.getItemCode());
            sqlNameLabel.setText(transactionItem.getName());
            sqlOldQuantityLabel.setText(transactionItem.getOldQuantity());
            sqlTypeLabel.setText(transactionItem.getType());
            sqlTransactionQuantityLabel.setText(transactionItem.getTransactionQuantity());
            sqlDateLabel.setText(transactionItem.getTransactionDate());
            sqlStatusLabel.setText(transactionItem.getStatus());
            this.pack();
            break;
        }
    }
}
```

DeleteTransactionView.java (5/5)

```
DeleteTransaction\view.java
    private void deleteButtonActionPerformed(DefaultTableModel tableModel) { 1 usage  Giorgos-Karachristos *
        if (transactionIDComboBox.getSelectedIndex() == 0) {
            JOptionPane.showMessageDialog( parentComponent: this, message: "Please select a transaction to delete",
                    title: "Info", JOptionPane.INFORMATION_MESSAGE);
            return;
        }
        String result = transactionController.deleteInventory(Integer.parseInt(
                (String) transactionIDComboBox.getSelectedItem()));
        if (result.contains("Delete failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result, title: "Error", JOptionPane.ERROR_MESSAGE);
        } else {
            JOptionPane.showMessageDialog( parentComponent: this, result, title: "Info", JOptionPane.INFORMATION_MESSAGE);
        }

        transactionController.getTransactionTable(tableModel);
        populateComboBox();
        formWindowClosing();
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { rowData = transactionController.getTransactionID(transactionBoxModel); }
}
```

InsertTransactionView.java (1/5)

```
InsertTransaction\view.java
package view.transaction;

import controller.inventory.InventoryController;
import controller.transaction.TransactionController;
import entities.InventoryItem;

import javax.swing.*;
import javax.swing.event.TableModelEvent;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.util.List;

public class InsertTransactionView extends JDialog {  2 usages  Giorgos-Karach

    private final JComboBox<String> typeComboBox;  9 usages
    private final JTable jTable;  14 usages
    private final DefaultTableModel tableModel;  24 usages
    private final JComboBox<String> itemComboBox;  6 usages
    private final DefaultComboBoxModel<String> itemBoxModel;  2 usages
    private final InventoryController inventoryController;  2 usages
    private final TransactionController transactionController;  3 usages

    private List<InventoryItem> inventoryItemList;  2 usages

    public InsertTransactionView(DefaultTableModel backgroundTableModel) {
        inventoryController = new InventoryController();
        transactionController = new TransactionController();
        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);
        this.setTitle("Add Transaction");
        this.setPreferredSize(new Dimension( width: 950,  height: 435));
        this.setResizable(false);
```

InsertTransactionView.java (2/5)

```
InsertTransactionView.java
    JPanel topPanel = new JPanel();

    JLabel typeLabel = new JLabel();
    typeLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    typeLabel.setText("Type:");
    topPanel.add(typeLabel);

    typeComboBox = new JComboBox<>();
    typeComboBox.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    typeComboBox.setModel(new DefaultComboBoxModel<>(new String[]{"Add", "Delete"}));
    topPanel.add(typeComboBox);

    JButton addItemButton = new JButton();
    addItemButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    addItemButton.setText("Add Item");
    addItemButton.addActionListener( ActionEvent evt1 -> addItemButtonActionPerformed());
    topPanel.add(addItemButton);

    JButton deleteItemButton = new JButton();
    deleteItemButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    deleteItemButton.setText("Delete Item");
    deleteItemButton.addActionListener( ActionEvent e -> deleteItemButtonActionPerformed());
    topPanel.add(deleteItemButton);

    JButton submitButton = new JButton();
    submitButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    submitButton.setText("Submit");
    submitButton.addActionListener((ActionEvent evt) -> submitButtonActionPerformed(backgroundTableModel));
    topPanel.add(submitButton);

    this.add(topPanel, BorderLayout.PAGE_START);

    jTable = new JTable();
    jTable.getTableHeader().setReorderingAllowed(false);
    jTable.getTableHeader().setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
    jTable.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 16));
    jTable.setRowHeight(40);
    jTable.setGridColor(new Color( r: 0,  g: 0,  b: 0));
    jTable.setShowGrid(true);

    tableModel = new DefaultTableModel(new Object[][]{}, new String[]{"Item ID", "Code", "Name",  Giorgos-Ka
            "In Stock", "Quantity", "New Quantity", "Max Quantity"}) {
        final boolean[] canEdit = new boolean[]{true, false, false, false, true, false, false};  1 usage

        @Override  Giorgos-Karachristos
        public boolean isCellEditable(int rowIndex, int columnIndex) { return canEdit[columnIndex]; }
    };
    jTable.setModel(tableModel);

    itemComboBox = new JComboBox<>();
    itemBoxModel = (DefaultComboBoxModel<String>) itemComboBox.getModel();
    populateComboBox();

    itemComboBox.addActionListener( ActionEvent evt -> comboBoxActionPerformed());
    jTable.getColumnModel().getColumn( columnIndex: 0).setCellEditor(new DefaultCellEditor(itemComboBox));

    tableModel.addTableModelListener(this::tableChanged);

    JScrollPane jScrollPane = new JScrollPane();
    jScrollPane.setViewportView(jTable);
    this.add(jScrollPane, BorderLayout.CENTER);

    this.addWindowListener((WindowAdapter) windowClosing(evt) -> { formWindowClosing(); });
    this.pack();
    this.setLocationRelativeTo(null);
}

private void addItemButtonActionPerformed() {  1 usage  Giorgos-Karachristos
    tableModel.addRow(new Object[]{});
    int lastRow = jTable.getRowCount() - 1;
    jTable.setRowSelectionInterval(lastRow, lastRow);
    typeComboBox.setEnabled(false);
}
```

InsertTransactionView.java (3/5)

```
InsertTransactionView.java
private void deleteItemButtonActionPerformed() {  1 usage   Giorgos-Karachristos *
    int row = jTable.getSelectedRow();
    if (row != -1) {
        tableModel.removeRow(row);
    } else {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Please select a row to delete",
                title: "Info", JOptionPane.INFORMATION_MESSAGE);
    }
    if (tableModel.getRowCount() == 0) {
        typeComboBox.setEnabled(true);
    }
}

private static final int ITEM_ID_COL = 0;  3 usages
private static final int CODE_COL = 1;  1 usage
private static final int NAME_COL = 2;  1 usage
private static final int IN_STOCK_COL = 3;  3 usages
private static final int QUANTITY_COL = 4;  5 usages
private static final int NEW_QUANTITY_COL = 5;  3 usages
private static final int MAX_QUANTITY_COL = 6;  2 usages

private void submitButtonActionPerformed(DefaultTableModel backgroundTableModel) {  1 usage   Giorgos-Karachristos *
    int rowCount = tableModel.getRowCount();
    boolean allValid = true;

    if (rowCount == 0) {
        JOptionPane.showMessageDialog( parentComponent: this,  message: "Please add items into the transaction",
                title: "Info", JOptionPane.INFORMATION_MESSAGE);
        return;
    }

    for (int row = 0; row < rowCount; row++) {
        String quantityText = (String) tableModel.getValueAt(row, QUANTITY_COL);
        if (tableModel.getValueAt(row, ITEM_ID_COL) == null || quantityText == null
                || tableModel.getValueAt(row, NEW_QUANTITY_COL) == null) {
            allValid = false;
            JOptionPane.showMessageDialog( parentComponent: this,  message: "All fields are required for row "
                    + (row + 1),  title: "Validation Error", JOptionPane.WARNING_MESSAGE);
            return;
        }
        allValid = isValid(quantityText, row);
    }

    if (allValid) {
        processAllTransactions(rowCount);
        transactionController.getTransactionTable(backgroundTableModel);
        populateComboBox();
        formWindowClosing();
    }
}

private void processAllTransactions(int rowCount) {  1 usage   Giorgos-Karachristos *
    int failed = 0;
    String result;

    for (int row = 0; row < rowCount; row++) {
        result = transactionController.insertTransaction(Integer.parseInt(
                tableModel.getValueAt(row, ITEM_ID_COL).toString()), Integer.parseInt(
                        tableModel.getValueAt(row, IN_STOCK_COL).toString()), String.valueOf(
                                typeComboBox.getSelectedItem()), Integer.parseInt(
                                        tableModel.getValueAt(row, QUANTITY_COL).toString()), Integer.parseInt(
                                                tableModel.getValueAt(row, NEW_QUANTITY_COL).toString()));

        if (result.contains("Insertion failed: ")) {
            JOptionPane.showMessageDialog( parentComponent: this, result,  title: "Error", JOptionPane.ERROR_MESSAGE);
            failed++;
        }
    }
    JOptionPane.showMessageDialog( parentComponent: this,  message: rowCount - failed
            + " transactions inserted successfully.\n" + failed + " transactions failed.",
            title: "Transaction Results", JOptionPane.INFORMATION_MESSAGE);
}
```

InsertTransactionView.java (4/5)

```
InsertTransactionView.java
private void comboBoxActionPerformed() {  1 usage   Giorgos-Karachristos
    int row = jTable.getSelectedRow();
    for (InventoryItem inventoryItem : inventoryItemList) {
        if (inventoryItem.getItemId().equals(itemComboBox.getSelectedItem())) {
            tableModel.setValueAt(inventoryItem.getItemCode(), row, CODE_COL);
            tableModel.setValueAt(inventoryItem.getName(), row, NAME_COL);
            tableModel.setValueAt(Integer.valueOf(inventoryItem.getQuantity()), row, IN_STOCK_COL);
            tableModel.setValueAt(Integer.valueOf(inventoryItem.getMaxQuantity()), row, MAX_QUANTITY_COL);
            break;
        }
    }
}

private void tableChanged(TableModelEvent e) {  1 usage   Giorgos-Karachristos *
    int row = e.getFirstRow();
    int column = e.getColumn();

    if (column == ITEM_ID_COL) {
        tableModel.setValueAt( aValue: null, row, QUANTITY_COL);
    }

    if (itemComboBox.getSelectedIndex() != 0 && column == QUANTITY_COL) {//User Input column changed
        Object stock = tableModel.getValueAt(row, IN_STOCK_COL);
        if (stock != null) {
            String quantityText = (String) tableModel.getValueAt(row, QUANTITY_COL);
            if (!isValid(quantityText, row)) {
                return;
            }
            int quantity = Integer.parseInt(quantityText);
            int inStock = (int) stock;
            int newQuantity;
            if ("Add".equals(typeComboBox.getSelectedItem())) {
                newQuantity = inStock + quantity;
            } else {
                newQuantity = inStock - quantity;
            }

            if (newQuantity < 0) {
                JOptionPane.showMessageDialog( parentComponent: this,
                        message: "New quantity cannot be negative for row " + (row + 1),
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                return;
            }

            int maxQuantity = Integer.parseInt(tableModel.getValueAt(row, MAX_QUANTITY_COL).toString());
            if (newQuantity > maxQuantity) {
                JOptionPane.showMessageDialog( parentComponent: this,
                        message: "New quantity cannot pass the maximum quantity for row " + (row + 1),
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                return;
            }
            tableModel.setValueAt(newQuantity, row, NEW_QUANTITY_COL);
        }
    }
}
```

InsertTransactionView.java (5/5)

```
InsertTransaction\view.java
    private boolean isValid(String price, int row) {  2 usages  Giorgos-Karachristos *
        try {
            int number = Integer.parseInt(price);
            if (number <= 0) {
                JOptionPane.showMessageDialog( parentComponent: this,
                        message: "Quantity must be greater than zero for row " + (row + 1),
                        title: "Validation Error", JOptionPane.WARNING_MESSAGE);
                return false;
            }
        } catch (NumberFormatException exception) {
            JOptionPane.showMessageDialog( parentComponent: this,
                    message: "Quantity must be number for row " + (row + 1),
                    title: "Validation Error", JOptionPane.WARNING_MESSAGE);
            return false;
        }
        return true;
    }

    private void formWindowClosing() {  2 usages  Giorgos-Karachristos
        tableModel.setRowCount(0);
        typeComboBox.setEnabled(true);
    }

    @Override  Giorgos-Karachristos
    public void setVisible(boolean visible) {
        if (visible) {
            populateComboBox();
        }
        super.setVisible(visible);
    }

    private void populateComboBox() { inventoryItemList = inventoryController.getInventoryList(itemBoxModel); }
}
```

A.8.18 Waiter

WaiterView.java (1/2)

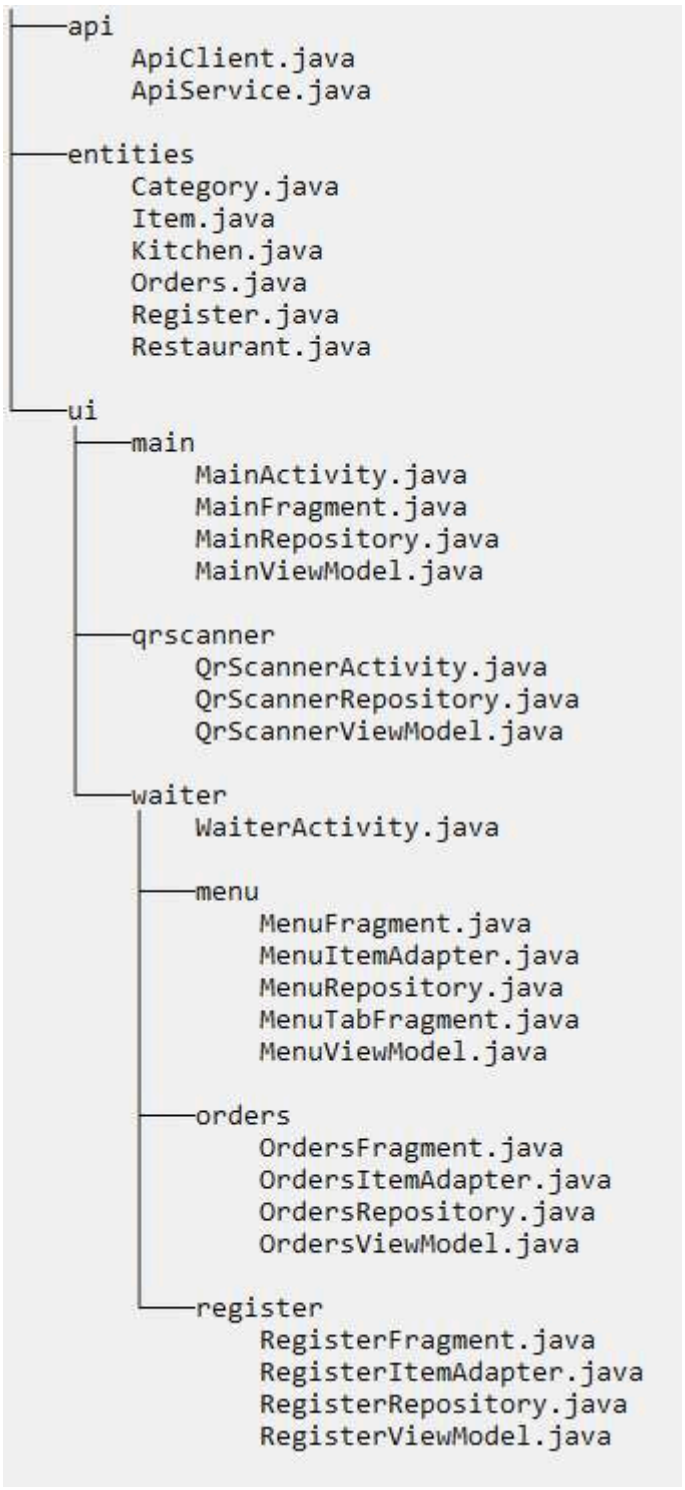
```
Waiter\view.java
package view.waiter;

import com.google.zxing.WriterException;
import controller.waiter.WaiterController;

import javax.swing.*;
import java.awt.*;
import java.io.IOException;

public class WaiterView extends JPanel {  2 usages  Giorgc

    private final WaiterController controller;  3 usages
    private final JLabel qrCodeLabel;  4 usages
    private final String name;  2 usages
    private final String surname;  2 usages

    public WaiterView(String name, String surname) {  1 u
        this.name = name;
        this.surname = surname;
        this.controller = new WaiterController();

        GridBagConstraints gbc;
        setLayout(new GridBagLayout());
        setBackground(new Color( r: 250,  g: 188,  b: 63));
```

WaiterView.java (2/2)

```
        JLabel textLabel = new JLabel();
        textLabel.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        textLabel.setText("Scan the QR code with your mobile app:");
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 0;
        gbc.gridwidth = 2;
        gbc.anchor = GridBagConstraints.CENTER;
        this.add(textLabel, gbc);

        qrCodeLabel = new JLabel();
        gbc = new GridBagConstraints();
        gbc.gridx = 0;
        gbc.gridy = 1;
        gbc.gridwidth = 2;
        this.add(qrCodeLabel, gbc);

        JButton refreshButton = new JButton();
        refreshButton.setBackground(new Color( r: 63,  g: 169,  b: 245));
        refreshButton.setFont(new Font( name: "Segoe UI", Font.PLAIN,  size: 20));
        refreshButton.setText("Refresh");
        gbc = new GridBagConstraints();
        gbc.gridx = 1;
        gbc.gridy = 2;
        gbc.anchor = GridBagConstraints.LAST_LINE_END;
        this.add(refreshButton, gbc);

        refreshButton.addActionListener( ActionEvent evt -> onRefreshClicked());

        createQRCode();
    }

    private void onRefreshClicked() {  1 usage   Giorgos-Karachristos
        createQRCode();
    }

    private void createQRCode() {  2 usages   Giorgos-Karachristos *
        try {
            String qrText = controller.fetchWaiterToken(name, surname);
            qrCodeLabel.setIcon(new ImageIcon(controller.generateQRCodeImage(qrText,  width: 500,  height: 500)));
            qrCodeLabel.revalidate();
        } catch (WriterException | IOException e) {
            JOptionPane.showMessageDialog( parentComponent: this,  message: "Error generating QR code: " + e.getMessage(),
                    title: "Error", JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

Παράρτημα Β: Κώδικας Android Εφαρμογής

Β.1 Δομή πηγαίου κώδικα Android (app/src/main/java)



B.2 AndroidManifest.xml

AndroidManifest.xml

AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA" />

    <uses-feature android:name="android.hardware.camera.autofocus" />
    <uses-feature android:name="android.hardware.camera" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher_icon"
        android:label="Waiter POS"
        android:roundIcon="@mipmap/ic_launcher_icon"
        android:supportsRtl="true"
        android:theme="@style/Theme.WaiterPOS">
        <activity
            android:name=".ui.waiter.WaiterActivity"
            android:exported="false"
            android:label="WaiterActivity"
            android:theme="@style/Theme.WaiterPOS.NoActionBar" />
        <activity
            android:name=".ui.qrscanner.QrScannerActivity"
            android:exported="false" />
        <activity
            android:name=".ui.main.MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

B.3 API

ApiClient.java

ApiClient.java

```
package com.waiterpos.api;

import android.util.Log;

import java.util.concurrent.TimeUnit;

import okhttp3.OkHttpClient;
import okhttp3.Request;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

17 usages
public class ApiClient {
    1 usage
    private static final String BASE_URL = "https://users.it.teithe.gr/~it185192/pos/pos.php/";
    3 usages
    private static Retrofit retrofit = null;

    12 usages
    public static ApiService getApiService() {
        if (retrofit == null) {
            OkHttpClient okHttpClient = new OkHttpClient.Builder().addInterceptor( Chain chain -> {
                Request request = chain.request();
                Log.i( tag: "ApiClient",  msg: "Request URL: " + request.url());
                return chain.proceed(request);
            }).connectTimeout( timeout: 15, TimeUnit.SECONDS).readTimeout( timeout: 15, TimeUnit.SECONDS).
                    writeTimeout( timeout: 15, TimeUnit.SECONDS).build();

            retrofit = new Retrofit.Builder().baseUrl(BASE_URL).client(okHttpClient).
                    addConverterFactory(GsonConverterFactory.create()).build();
        }
        return retrofit.create(ApiService.class);
    }
}
```

ApiService.java

```
ApiService.java
package com.waiterpos.api;

import com.waiterpos.entities.Category;
import com.waiterpos.entities.Item;
import com.waiterpos.entities.Kitchen;
import com.waiterpos.entities.Orders;
import com.waiterpos.entities.Register;
import com.waiterpos.entities.Restaurant;

import java.util.List;

import okhttp3.RequestBody;
import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.GET;
import retrofit2.http.POST;
import retrofit2.http.Query;

2 usages
public interface ApiService {
    //MainRepository
    1 usage
    @GET("manager/restaurant")
    Call<List<Restaurant>> getRestaurantInfo();

    //QrScannerRepository
    1 usage
    @POST("waiter/isValid")
    Call<String> isValid(@Body RequestBody requestBody);

    //MenuRepository
    1 usage
    @GET("cashier/menu/getCategoryList")
    Call<List<Category>> getCategoryList();

    1 usage
    @GET("cashier/menu/getItemsList")
    Call<List<Item>> getItemsListFromCategory(@Query("category") String category);

    1 usage
    @POST("cashier/menu/insertOrder")
    Call<String> insertOrderToKitchen(@Body RequestBody requestBody);

    1 usage
    @POST("cashier/menu/insertKitchen")
    Call<String> insertItemToOrder(@Body RequestBody requestBody);

    //OrdersRepository
    1 usage
    @GET("cashier/order/getActiveOrdersList")
    Call<List<Orders>> getActiveOrdersList();

    1 usage
    @GET("cashier/order/getKitchenList")
    Call<List<Kitchen>> getKitchenList(@Query("orderID") String category);

    1 usage
    @POST("cashier/order/voidKitchen")
    Call<String> voidKitchen(@Body RequestBody requestBody);

    1 usage
    @POST("cashier/order/cancelOrder")
    Call<String> cancelOrder(@Body RequestBody requestBody);

    //RegisterRepository
    1 usage
    @GET("cashier/register/getItemsList")
    Call<List<Register>> getRegisterList(@Query("orderID") String category);

    1 usage
    @POST("cashier/register/completeOrder")
    Call<String> completeOrder(@Body RequestBody requestBody);
}
```

B.4 Entities

Category.java

```
Category.java
package com.waiterpos.entities;

import com.google.gson.annotations.SerializedName;

14 usages
public class Category {
    1 usage
    @SerializedName("NAME")
    private String name;
    1 usage
    @SerializedName("COLOR")
    private String color;
    1 usage
    @SerializedName("TAX")
    private String tax;

    public String getName() { return name; }

    public String getColor() { return color; }

    1 usage
    public String getTax() { return tax; }
}
```

Item.java

```
Item.java
package com.waiterpos.entities;

import com.google.gson.annotations.SerializedName;

21 usages
public class Item {
    1 usage
    @SerializedName("NAME")
    private String name;
    1 usage
    @SerializedName("COLOR")
    private String color;
    1 usage
    @SerializedName("PRICE")
    private String price;
    1 usage
    @SerializedName("ITEM_ID")
    private String itemID;
    1 usage
    @SerializedName("DESCRIPTION")
    private String description;

    public String getName() { return name; }

    public String getColor() { return color; }

    2 usages
    public String getPrice() { return price; }

    10 usages
    public String getItemID() { return itemID; }

    1 usage
    public String getDescription() { return description; }
}
```

Kitchen.java

```java
Kitchen.java
package com.waiterpos.entities;

import com.google.gson.annotations.SerializedName;

22 usages
public class Kitchen {
    1 usage
    @SerializedName("NAME")
    private String name;
    1 usage
    @SerializedName("QUANTITY")
    private String quantity;
    1 usage
    @SerializedName("STATUS")
    private String status;

    public String getName() { return name; }

    public String getQuantity() { return quantity; }

    1 usage
    public String getStatus() { return status; }
}
```

Orders.java

```java
Orders.java
package com.waiterpos.entities;

import com.google.gson.annotations.SerializedName;

20 usages
public class Orders {
    1 usage
    @SerializedName("ORDER_ID")
    private String orderID;
    1 usage
    @SerializedName("SUBTOTAL")
    private String subtotal;
    1 usage
    @SerializedName("STATUS")
    private String status;
    1 usage
    @SerializedName("TYPE")
    private String type;
    1 usage
    @SerializedName("ORDER_DATE")
    private String orderDate;

    7 usages
    public String getOrderID() { return orderID; }

    1 usage
    public String getSubtotal() { return subtotal; }

    1 usage
    public String getStatus() { return status; }

    2 usages
    public String getType() { return type; }

    1 usage
    public String getOrderDate() { return orderDate; }
}
```

Register.java

```java
Register.java
package com.waiterpos.entities;

import com.google.gson.annotations.SerializedName;

18 usages
public class Register {
    1 usage
    @SerializedName("NAME")
    private String name;
    1 usage
    @SerializedName("QUANTITY")
    private int quantity;
    1 usage
    @SerializedName("TAX")
    private String tax;

    1 usage
    @SerializedName("TOTAL_PRICE")
    private String totalPrice;

    public String getName() { return name; }

    public int getQuantity() { return quantity; }

    1 usage
    public String getTax() { return tax; }

    2 usages
    public String getTotalPrice() { return totalPrice; }
}
```

Restaurant.java

```java
Restaurant.java
package com.waiterpos.entities;

import com.google.gson.annotations.SerializedName;

11 usages
public class Restaurant {
    1 usage
    @SerializedName("NAME")
    private String name;

    1 usage
    @SerializedName("TABLES")
    private int table;

    public String getName() { return name; }

    1 usage
    public int getTable() { return table; }
}
```

B.5 UI

B.5.1 Main

MainActivity.java

```java
MainActivity.java
package com.waiterpos.ui.main;

import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.graphics.Insets;
import androidx.core.view.ViewCompat;
import androidx.core.view.WindowInsetsCompat;

import com.waiterpos.R;

public class MainActivity extends AppCompatActivity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction().replace(R.id.main,
                    MainFragment.newInstance()).commitNow();
        }

        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main),
                ( View v, WindowInsetsCompat insets) -> {
            Insets systemBars = insets.getInsets(WindowInsetsCompat.Type.systemBars());
            v.setPadding(systemBars.left, systemBars.top, systemBars.right, systemBars.bottom);
            return insets;
        });
    }
}
```

MainFragment.java

```java
package com.waiterpos.ui.main;

import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;

import com.waiterpos.databinding.FragmentMainBinding;
import com.waiterpos.ui.qrscanner.QrScannerActivity;

public class MainFragment extends Fragment {

    5 usages
    private FragmentMainBinding binding;
    2 usages
    private int tablesNumber = 0;

    1 usage
    public static MainFragment newInstance() { return new MainFragment(); }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        binding = FragmentMainBinding.inflate(inflater, container, attachToParent: false);
        return binding.getRoot();
    }

    14 usages
    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        MainViewModel mainViewModel = new ViewModelProvider(requireActivity())
                .get(MainViewModel.class);

        mainViewModel.getRestaurantName().observe(getViewLifecycleOwner(),
                String name -> binding.restaurantNameTextView.setText(name));

        mainViewModel.getTablesNumber().observe(getViewLifecycleOwner(),
                Integer tables -> tablesNumber = tables);

        mainViewModel.getErrorLiveData().observe(getViewLifecycleOwner(),
                String error -> {
            if (error != null && !error.isEmpty()) {
                Toast.makeText(getContext(), text: "Failed to load restaurant info.",
                        Toast.LENGTH_SHORT).show();
                Log.e( tag: "MainFragment", msg: "Error: " + error);
            }
        });

        mainViewModel.fetchRestaurantInfo();

        binding.scanQrButton.setOnClickListener( View v -> openQRScanner());
    }

    1 usage
    private void openQRScanner() {
        Intent intent = new Intent(requireContext(), QrScannerActivity.class);
        intent.putExtra( name: "TABLES_COUNT", tablesNumber);
        startActivity(intent);
    }

    15 usages
    @Override
    public void onDestroyView() {
        super.onDestroyView();
        binding = null;
    }
}
```

MainRepository.java

```java
MainRepository.java
package com.waiterpos.ui.main;

import androidx.annotation.NonNull;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;

import com.waiterpos.api.ApiClient;
import com.waiterpos.entities.Restaurant;

import java.util.List;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

2 usages
public class MainRepository {
    2 usages
    private final MutableLiveData<Restaurant> restaurantLiveData = new MutableLiveData<>();
    3 usages
    private final MutableLiveData<String> errorLiveData = new MutableLiveData<>();

    1 usage
    public LiveData<Restaurant> getRestaurantInfo() {
        Call<List<Restaurant>> call = ApiClient.getApiService().getRestaurantInfo();
        call.enqueue(new Callback<>() {
            @Override
            public void onResponse(@NonNull Call<List<Restaurant>> call,
                                   @NonNull Response<List<Restaurant>> response) {
                if (response.isSuccessful() && response.body() != null
                        && !response.body().isEmpty()) {
                    restaurantLiveData.postValue(response.body().get(0));
                } else {
                    errorLiveData.postValue("No restaurant info found");
                }
            }

            @Override
            public void onFailure(@NonNull Call<List<Restaurant>> call, @NonNull Throwable t) {
                errorLiveData.postValue(t.getMessage());
            }
        });
        return restaurantLiveData;
    }

    public LiveData<String> getErrorLiveData() { return errorLiveData; }
}
```

MainViewModel.java

```java
MainViewModel.java
package com.waiterpos.ui.main;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.Transformations;
import androidx.lifecycle.ViewModel;

import com.waiterpos.entities.Restaurant;

2 usages
public class MainViewModel extends ViewModel {
    2 usages
    private final MutableLiveData<Boolean> triggerFetch = new MutableLiveData<>( value: false);
    2 usages
    private final LiveData<String> restaurantName;
    2 usages
    private final LiveData<Integer> tablesNumber;
    2 usages
    private final LiveData<String> errorLiveData;
    2 usages
    private final MainRepository mainRepository = new MainRepository();

    no usages
    public MainViewModel() {
        // When triggerFetch changes, request new data
        LiveData<Restaurant> restaurantLiveData = Transformations.switchMap(triggerFetch,
                Boolean trigger ->
                trigger ? mainRepository.getRestaurantInfo() : new MutableLiveData<>( value: null));

        // Expose mapped values
        restaurantName = Transformations.map(restaurantLiveData,
                Restaurant restaurant -> restaurant != null
                ? restaurant.getName() : "Unknown");

        tablesNumber = Transformations.map(restaurantLiveData,
                Restaurant tables -> tables != null
                ? tables.getTable() : 0);

        errorLiveData = mainRepository.getErrorLiveData();
    }

    1 usage
    public LiveData<String> getRestaurantName() { return restaurantName; }

    1 usage
    public LiveData<Integer> getTablesNumber() { return tablesNumber; }

    public LiveData<String> getErrorLiveData() { return errorLiveData; }

    1 usage
    public void fetchRestaurantInfo() { triggerFetch.setValue(true); }
}
```

B.5.2 QR Scanner

QrScannerActivity.java (1/2)

QrScannerActivity.java

```java
package com.waiterpos.ui.qrscanner;

import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.lifecycle.ViewModelProvider;

import com.waiterpos.ui.main.MainActivity;
import com.waiterpos.ui.waiter.WaiterActivity;
import com.google.mlkit.vision.barcode.BarcodeScanning;
import com.google.mlkit.vision.barcode.common.Barcode;
import com.google.mlkit.vision.common.InputImage;

public class QrScannerActivity extends AppCompatActivity {

    5 usages
    private QrScannerViewModel viewModel;
    2 usages
    private static final int CAMERA_PERMISSION_REQUEST = 100;
    2 usages
    private static final int REQUEST_IMAGE_CAPTURE = 1;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        viewModel = new ViewModelProvider( owner: this).get(QrScannerViewModel.class);

        viewModel.getQrValidationResult().observe( owner: this,  String result -> {
            if (!"No ok".equals(result)) {
                navigateToWaiterActivity(result);
            } else {
                Toast.makeText( context: this,  text: "Invalid QR Code", Toast.LENGTH_SHORT).show();
                goBackToMainActivity();
            }
        });

        viewModel.getErrorLiveData().observe( owner: this,  String error -> {
            if (error != null && !error.isEmpty()) {
                Toast.makeText( context: this,  text: "Network error occurred", Toast.LENGTH_SHORT).show();
                Log.e( tag: "QrScannerActivity",  msg: "Error: " + error);
                viewModel.clearError();
                goBackToMainActivity();
            }
        });

        if (ContextCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA)
                != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.CAMERA},
                    CAMERA_PERMISSION_REQUEST);
        } else {
            openCamera();
        }
    }
```

QrScannerActivity.java (2/2)

QrScannerActivity.java

```
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
                                           @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (requestCode == CAMERA_PERMISSION_REQUEST) {
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                openCamera();
            } else {
                Toast.makeText( context: this,  text: "Camera permission denied", Toast.LENGTH_SHORT).show();
            }
        }
    }

    2 usages
    private void openCamera() {
        Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
            startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
        } else {
            Toast.makeText( context: this,  text: "No camera app found", Toast.LENGTH_SHORT).show();
            goBackToMainActivity();
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
            if (data != null && data.getExtras() != null) {
                Bitmap imageBitmap = (Bitmap) data.getExtras().get("data");
                if (imageBitmap != null) {
                    scanQRCode(imageBitmap);
                } else {
                    Toast.makeText( context: this,  text: "Failed to capture image",
                            Toast.LENGTH_SHORT).show();
                    goBackToMainActivity();
                }
            }
        }
    }

    1 usage
    private void scanQRCode(Bitmap imageBitmap) {
        InputImage image = InputImage.fromBitmap(imageBitmap,  rotationDegrees: 0);
        BarcodeScanning.getClient().process(image).addOnSuccessListener( List<Barcode> barcodes -> {
            if (barcodes.isEmpty()) {
                Toast.makeText( context: this,  text: "No QR code found", Toast.LENGTH_SHORT).show();
                goBackToMainActivity();
                return;
            }
            for (Barcode barcode : barcodes) {
                String qrValue = barcode.getRawValue();
                if (qrValue != null) {
                    viewModel.validateQrCode(qrValue);
                    return;
                }
            }
        }).addOnFailureListener( Exception e -> {
            Toast.makeText( context: QrScannerActivity.this,  text: "Failed to scan QR code.",
                    Toast.LENGTH_SHORT).show();
            goBackToMainActivity();
        });
    }

    1 usage
    private void navigateToWaiterActivity(String headerText) {
        Log.i( tag: "QrScannerActivity",  msg: "Tables Number:"
                + getIntent().getIntExtra( name: "TABLES_COUNT",  defaultValue: 0) + " Name: " + headerText);
        Intent intent = new Intent( packageContext: QrScannerActivity.this, WaiterActivity.class);
        intent.putExtra( name: "HEADER_TEXT", headerText);
        intent.putExtra( name: "TABLES_COUNT", getIntent().getIntExtra( name: "TABLES_COUNT",  defaultValue: 0));
        startActivity(intent);
        finish();
    }

    6 usages
    private void goBackToMainActivity() {
        Intent intent = new Intent( packageContext: QrScannerActivity.this, MainActivity.class);
        startActivity(intent);
        finish();
    }
}
```

QrScannerRepository.java

```java
QrScannerRepository.java
package com.waiterpos.ui.qrscanner;

import android.util.Log;

import androidx.annotation.NonNull;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;

import com.waiterpos.api.ApiClient;

import org.json.JSONException;
import org.json.JSONObject;

import okhttp3.MediaType;
import okhttp3.RequestBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

2 usages
public class QrScannerRepository {
    6 usages
    private final MutableLiveData<String> errorLiveData = new MutableLiveData<>();
    2 usages
    private final MutableLiveData<String> qrScannerLiveData = new MutableLiveData<>();

    1 usage
    public LiveData<String> validateQrCode(String qrCodeContent) {
        JSONObject jsonPayload = new JSONObject();
        try {
            jsonPayload.put( name: "TOKEN", qrCodeContent);
        } catch (JSONException e) {
            Log.e( tag: "QrScannerRepository",  msg: "Invalid QR payload" + e.getMessage());
            errorLiveData.postValue("Invalid QR payload");
            return errorLiveData;
        }

        RequestBody requestBody = RequestBody.create(jsonPayload.toString(),
                MediaType.parse( $this$parse: "application/json"));
        Call<String> call = ApiClient.getApiService().isValid(requestBody);
        call.enqueue(new Callback<>() {
            @Override
            public void onResponse(@NonNull Call<String> call, @NonNull Response<String> response) {
                if (response.isSuccessful() && response.body() != null && !response.body().isEmpty()) {
                    qrScannerLiveData.postValue(response.body());
                } else {
                    errorLiveData.postValue("No info found");
                }
            }

            @Override
            public void onFailure(@NonNull Call<String> call, @NonNull Throwable t) {
                errorLiveData.postValue(t.getMessage());
            }
        });

        return qrScannerLiveData;
    }

    public LiveData<String> getErrorLiveData() { return errorLiveData; }

    public void clearError() { errorLiveData.setValue(null); }
}
```

QrScannerViewModel.java

```java
QrScannerViewModel.java
package com.waiterpos.ui.qrscanner;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.Transformations;
import androidx.lifecycle.ViewModel;

2 usages
public class QrScannerViewModel extends ViewModel {
    3 usages
    private final QrScannerRepository qrScannerRepository = new QrScannerRepository();

    2 usages
    private final MutableLiveData<String> qrCodeTrigger = new MutableLiveData<>();

    1 usage
    public LiveData<String> getQrValidationResult() {
        return Transformations.switchMap(qrCodeTrigger, qrScannerRepository::validateQrCode);
    }

    1 usage
    public void validateQrCode(String qrCodeContent) { qrCodeTrigger.setValue(qrCodeContent); }

    public LiveData<String> getErrorLiveData() { return qrScannerRepository.getErrorLiveData(); }

    public void clearError() { qrScannerRepository.clearError(); }
}
```

B.5.3 Waiter

WaiterActivity.java (1/2)

```java
WaiterActivity.java
package com.waiterpos.ui.waiter;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.navigation.ui.AppBarConfiguration;
import androidx.navigation.ui.NavigationUI;

import com.waiterpos.R;
import com.waiterpos.databinding.ActivityWaiterBinding;
import com.google.android.material.navigation.NavigationView;

public class WaiterActivity extends AppCompatActivity {

    3 usages
    private AppBarConfiguration mAppBarConfiguration;
    7 usages
    private ActivityWaiterBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

WaiterActivity.java (2/2)

```
WaiterActivity.java
        binding = ActivityWaiterBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        setSupportActionBar(binding.appBarWaiter.toolbar);
        DrawerLayout drawer = binding.drawerLayout;
        NavigationView navigationView = binding.navView;
        // Passing each menu ID as a set of Ids because each
        // menu should be considered as top level destinations.
        mAppBarConfiguration = new AppBarConfiguration.Builder(R.id.nav_menu,
                R.id.nav_orders, R.id.nav_register).setOpenableLayout(drawer).build();
        NavController navController = Navigation.findNavController( activity: this,
                R.id.nav_host_fragment_content_waiter);
        NavigationUI.setupActionBarWithNavController( activity: this, navController,
                mAppBarConfiguration);
        NavigationUI.setupWithNavController(navigationView, navController);

        Intent intent = getIntent();
        if (intent != null) {
            if (intent.hasExtra( name: "HEADER_TEXT")) {
                setNavigationHeaderText(intent.getStringExtra( name: "HEADER_TEXT"));
            }
            if (intent.hasExtra( name: "ORDER_ID")) {
                Bundle bundle = new Bundle();
                bundle.putString("ORDER_ID", intent.getStringExtra( name: "ORDER_ID"));
                navController.navigate(R.id.nav_register, bundle);
            } else if (intent.hasExtra( name: "TABLES_COUNT")) {
                Bundle bundle = new Bundle();
                bundle.putInt("TABLES_KEY", intent.getIntExtra( name: "TABLES_COUNT",  defaultValue: 0));
                navController.navigate(R.id.nav_menu, bundle);
            }
        }
    }

    5 usages
    @Override
    public boolean onSupportNavigateUp() {
        NavController navController = Navigation.findNavController( activity: this,
                R.id.nav_host_fragment_content_waiter);
        return NavigationUI.navigateUp(navController, mAppBarConfiguration)
                || super.onSupportNavigateUp();
    }

    1 usage
    private void setNavigationHeaderText(String text) {
        TextView headerTextView = headerTextView();
        headerTextView.setText(text);
    }

    1 usage
    public String getNavigationHeaderText() {
        TextView headerTextView = headerTextView();
        return headerTextView.getText().toString();
    }

    2 usages
    private TextView headerTextView() {
        NavigationView navigationView = binding.navView;
        View headerView = navigationView.getHeaderView( index: 0);
        return headerView.findViewById(R.id.nav_header_title);
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        binding = null;
    }
}
```

B.5.3.1 Menu

MenuFragment.java (1/4)

MenuFragment.java

```java
package com.waiterpos.ui.waiter.menu;

import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ProgressBar;
import android.widget.Spinner;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.viewpager2.adapter.FragmentStateAdapter;

import com.waiterpos.databinding.FragmentMenuBinding;
import com.waiterpos.entities.Category;
import com.waiterpos.ui.waiter.WaiterActivity;
import com.google.android.material.tabs.TabLayoutMediator;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public class MenuFragment extends Fragment {
    14 usages
    private FragmentMenuBinding menuBinding;
    14 usages
    private MenuViewModel menuViewModel;
    5 usages
    private Double roundSubtotal = 0.0;

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        menuViewModel = new ViewModelProvider(requireActivity()).get(MenuViewModel.class);
        if (getArguments() != null) {
            int tables = getArguments().getInt( key: "TABLES_KEY",  defaultValue: 0);
            menuViewModel.setTablesMLD(tables);
        }
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        menuBinding = FragmentMenuBinding.inflate(inflater, container,  attachToParent: false);
        ProgressBar progressBar = menuBinding.progressBar2;

        // Observe categories
        menuViewModel.getCategories().observe(getViewLifecycleOwner(),  List<Category> categories -> {
            if (categories != null) {
                progressBar.setVisibility(View.GONE);
                setupViewPager(categories);
            } else {
                Toast.makeText(getContext(),  text: "Failed to load categories",
                        Toast.LENGTH_LONG).show();
            }
        });

        menuViewModel.tablesLiveData.observe(getViewLifecycleOwner(), this::setupSpinner);
```

MenuFragment.java (2/4)

```
MenuFragment.java
    // Observe subtotal updates
    menuViewModel.subtotalLiveData.observe(getViewLifecycleOwner(), Double subtotal -> {
        roundSubtotal = Math.round(subtotal * 100.0) / 100.0;
        menuBinding.numberSubtotalTextView.setText(roundSubtotal.toString());
    });

    // Error observer
    menuViewModel.getErrorLiveData().observe(getViewLifecycleOwner(), List<String> error -> {
        if (error != null && !error.isEmpty()) {
            Toast.makeText(getContext(), text: "Error Occurred", Toast.LENGTH_LONG).show();
            for (String er : error) {
                Log.e( tag: "MainFragment", msg: "Error: " + er);
            }
            menuViewModel.clearError();
        }
    });

    return menuBinding.getRoot();
}

2 usages
private void setupViewPager(List<Category> categories) {
    FragmentStateAdapter adapter = new FragmentStateAdapter(this) {
        1 usage
        @NonNull
        @Override
        public Fragment createFragment(int position) {
            return MenuTabFragment.newInstance(categories.get(position).getName(),
                    categories.get(position).getTax());
        }

        @Override
        public int getItemCount() { return categories.size(); }
    };

    menuBinding.categoryViewPager.setAdapter(adapter);

    new TabLayoutMediator(menuBinding.categoryTabLayout, menuBinding.categoryViewPager,
            ( Tab tab, int position) -> {
        try {
            int color = Color.parseColor(categories.get(position).getColor());
            tab.view.setBackgroundColor(color);
        } catch (IllegalArgumentException e) {
            Log.e( tag: "MenuFragment", msg: "Invalid color: "
                    + categories.get(position).getColor());
        }
        tab.setText(categories.get(position).getName());
    }).attach();

    menuBinding.kitchenButton.setOnClickListener( View v -> kitchenButtonActionPerformed());
}

1 usage
private void kitchenButtonActionPerformed() {
    WaiterActivity waiterActivity = (WaiterActivity) getActivity();
    if (waiterActivity == null) return;

    String name = waiterActivity.getNavigationHeaderText();
```

MenuFragment.java (3/4)

```
MenuFragment.java
    if (roundSubtotal == 0) {
        Toast.makeText(getContext(), text: "Please add items to the order",
                Toast.LENGTH_LONG).show();
        return;
    }

    Spinner tablesSpinner = menuBinding.tableSpinner;
    if (tablesSpinner.getSelectedItemPosition() == 0) {
        Toast.makeText(getContext(), text: "Please select a table", Toast.LENGTH_LONG).show();
        return;
    }
    String type = tablesSpinner.getSelectedItem().toString();

    Log.i( tag: "kitchenButton", msg: "name: " + name + " subtotal: "
            + roundSubtotal + " type: " + type);

    // Observe order insert response
    menuViewModel.insertOrderToKitchen(name, roundSubtotal, type)
            .observe(getViewLifecycleOwner(), String response -> {
        if (response != null && response.contains("Insertion failed:")) {
            Toast.makeText(getContext(), response, Toast.LENGTH_LONG).show();
        } else if (response != null) {
            Toast.makeText(getContext(), text: "Order inserted successfully",
                    Toast.LENGTH_LONG).show();
            insertKitchenItems();
        }
    });
}

4 usages
private int success = 0;

1 usage
private void insertKitchenItems() {
    Map<String, Double> totals = menuViewModel.getItemTotals();
    Map<String, Integer> quantities = menuViewModel.getItemQuantities();

    success = 0;

    for (String itemId : quantities.keySet()) {
        int qty = quantities.get(itemId);
        double totalPrice = totals.get(itemId);

        Log.i( tag: "insertKitchenItems", msg: "idItem: " + itemId + " quantity: "
                + qty + " totalPrice: " + totalPrice);
        menuViewModel.insertItemToOrder(Integer.parseInt(itemId), qty, totalPrice)
                .observe(getViewLifecycleOwner(), String response -> {
            if (response != null) {
                Log.i( tag: "insertKitchenItems", msg: "response: " + response);
                success++;
            }
            if (success == quantities.size()) {
                Toast.makeText(getContext(), text: "Successfully inserted: " + success
                        + "/" + quantities.size(), Toast.LENGTH_SHORT).show();
                menuViewModel.resetSubtotal();
                menuBinding.numberSubtotalTextView.setText("0.0");
                menuBinding.tableSpinner.setSelection(0);
                setupViewPager(menuViewModel.getCategories().getValue());
            }
        });
    }
}
```

MenuFragment.java (4/4)

```
MenuFragment.java
    private void setupSpinner(int tables) {
        Spinner tablesSpinner = menuBinding.tableSpinner;
        List<String> items = new ArrayList<>();
        items.add("Select a table");
        for (int i = 1; i <= tables; i++) {
            items.add("Table " + i);
        }

        ArrayAdapter<String> adapter = new ArrayAdapter<>(getContext(),
                android.R.layout.simple_spinner_item, items);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        tablesSpinner.setAdapter(adapter);
        tablesSpinner.setSelection(0);
    }

    15 usages
    @Override
    public void onDestroyView() {
        super.onDestroyView();
        menuBinding = null;
    }

    @Override
    public void onResume() {
        super.onResume();
        menuViewModel.resetSubtotal();
        menuBinding.tableSpinner.setSelection(0);
    }
}
```

MenuItemAdapter.java (1/3)

```
MenuItemAdapter.java
package com.waiterpos.ui.waiter.menu;

import android.graphics.Color;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.waiterpos.databinding.MenuItemAdapterBinding;
import com.waiterpos.entities.Item;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

4 usages
public class MenuItemAdapter extends RecyclerView.Adapter<MenuItemAdapter.ItemViewHolder> {
```

MenuItemAdapter.java (2/3)

```
MenuItemAdapter.java
private final List<Item> items;
8 usages
private final Map<String, Integer> quantityMap = new HashMap<>();
2 usages
private final String tax;
5 usages
private final OnQuantityChangeListener listener;

3 usages
private int expandedPosition = -1;

2 usages
public interface OnQuantityChangeListener {
    2 usages
    void onQuantityChanged(String itemId, int quantity, double totalPrice);
}

1 usage
public MenuItemAdapter(List<Item> items, String tax, OnQuantityChangeListener listener) {
    this.items = items;
    this.tax = tax;
    this.listener = listener;

    for (Item item : items) {
        quantityMap.put(item.getItemID(),  v: 0);
    }
}

1 usage
public void updateData(List<Item> newItems) {
    items.clear();
    items.addAll(newItems);
    quantityMap.clear();
    for (Item item : newItems) {
        quantityMap.put(item.getItemID(),  v: 0);
    }
    notifyDataSetChanged();
}

@NonNull
@Override
public ItemViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    MenuItemAdapterBinding binding = MenuItemAdapterBinding.inflate(
            LayoutInflater.from(parent.getContext()), parent,  attachToParent: false);
    return new MenuItemAdapter.ItemViewHolder(binding);
}

@Override
public void onBindViewHolder(@NonNull ItemViewHolder holder, int position) {
    Item item = items.get(position);
    holder.bind(item);

    if (item.getColor() != null) {
        try {
            holder.itemView.setBackgroundColor(Color.parseColor(item.getColor()));
        } catch (IllegalArgumentException e) {
            Log.e( tag: "MenuItemAdapter",  msg: "Invalid color: " + item.getColor());
        }
    }

    final boolean isExpanded = position == expandedPosition;
    holder.descriptionText.setVisibility(isExpanded ? View.VISIBLE : View.GONE);

    // Handle click event to toggle expansion
    holder.itemView.setOnClickListener( View v -> {
        expandedPosition = isExpanded ? -1 : position;
        if (holder.descriptionText.getText().length() == 0) {
            expandedPosition = -1;
        }
        notifyDataSetChanged();
    });
}

@Override
public int getItemCount() { return items.size(); }
```

MenuItemAdapter.java (2/3)

```java
MenuItemAdapter.java
    class ItemViewHolder extends RecyclerView.ViewHolder {
        5 usages
        private final MenuItemAdapterBinding binding;
        4 usages
        private final TextView quantityText, priceText, totalPriceText, descriptionText;

        1 usage
        public ItemViewHolder(MenuItemAdapterBinding binding) {
            super(binding.getRoot());
            this.binding = binding;

            quantityText = binding.quantity;
            priceText = binding.price;
            totalPriceText = binding.totalPrice;
            descriptionText = binding.textDescription;
        }

        1 usage
        public void bind(Item item) {
            binding.itemId.setText(item.getItemID());
            binding.itemName.setText(item.getName());
            descriptionText.setText(item.getDescription());

            // Parse price string to double safely
            double price = 0;
            try {
                price = Double.parseDouble(item.getPrice());
            } catch (NumberFormatException e) {
                Log.e( tag: "MenuItemAdapter",  msg: "Invalid price: " + item.getPrice());
            }
            double taxNumber = Double.parseDouble(tax);
            double priceInclTax = price + price * taxNumber / 100;
            double roundPriceInclTax = Math.round(priceInclTax * 100.0) / 100.0;
            priceText.setText(String.format("%.2f", roundPriceInclTax));

            // Get quantity from map
            int quantity = quantityMap.getOrDefault(item.getItemID(),  defaultValue: 0);
            quantityText.setText(String.valueOf(quantity));
            totalPriceText.setText(String.format("%.2f", roundPriceInclTax * quantity));

            binding.minusButton.setOnClickListener( View v -> {
                int qty = quantityMap.getOrDefault(item.getItemID(),  defaultValue: 0);
                if (qty > 0) {
                    qty--;
                    quantityMap.put(item.getItemID(), qty);
                    quantityText.setText(String.valueOf(qty));
                    totalPriceText.setText(String.format("%.2f", roundPriceInclTax * qty));
                    if (listener != null)
                        listener.onQuantityChanged(item.getItemID(), qty,  totalPrice: roundPriceInclTax * qty);
                }
            });

            binding.plusButton.setOnClickListener( View v -> {
                int qty = quantityMap.getOrDefault(item.getItemID(),  defaultValue: 0);
                qty++;
                quantityMap.put(item.getItemID(), qty);
                quantityText.setText(String.valueOf(qty));
                totalPriceText.setText(String.format("%.2f", roundPriceInclTax * qty));
                if (listener != null)
                    listener.onQuantityChanged(item.getItemID(), qty,  totalPrice: roundPriceInclTax * qty);
            });
        }
    }
}
```

MenuRepository.java (1/3)

```java
MenuRepository.java
package com.waiterpos.ui.waiter.menu;

import android.util.Log;

import androidx.annotation.NonNull;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;

import com.waiterpos.api.ApiClient;
import com.waiterpos.entities.Category;
import com.waiterpos.entities.Item;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

import okhttp3.MediaType;
import okhttp3.RequestBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

2 usages
public class MenuRepository {
    4 usages
    private final MutableLiveData<List<String>> errorLiveData = new MutableLiveData<>();

    1 usage
    public LiveData<List<Category>> getCategoryList() {
        MutableLiveData<List<Category>> categoriesLiveData = new MutableLiveData<>();
        Call<List<Category>> call = ApiClient.getApiService().getCategoryList();
        call.enqueue(new Callback<>() {
            @Override
            public void onResponse(@NonNull Call<List<Category>> call,
                                   @NonNull Response<List<Category>> response) {
                if (response.isSuccessful() && response.body() != null
                        && !response.body().isEmpty()) {
                    categoriesLiveData.postValue(response.body());
                } else {
                    postError("Failed to load categories");
                }
            }

            @Override
            public void onFailure(@NonNull Call<List<Category>> call, @NonNull Throwable t) {
                postError(t.getMessage());
            }
        });

        return categoriesLiveData;
    }

    1 usage
    public LiveData<List<Item>> getItemsList(String category) {
        MutableLiveData<List<Item>> itemsLiveData = new MutableLiveData<>();
        Call<List<Item>> call = ApiClient.getApiService().getItemsListFromCategory(category);
        call.enqueue(new Callback<>() {
            @Override
            public void onResponse(@NonNull Call<List<Item>> call,
                                   @NonNull Response<List<Item>> response) {
                if (response.isSuccessful() && response.body() != null
                        && !response.body().isEmpty()) {
                    itemsLiveData.postValue(response.body());
                } else {
                    postError("No items found for category: " + category);
                }
            }

            @Override
            public void onFailure(@NonNull Call<List<Item>> call, @NonNull Throwable t) {
                postError(t.getMessage());
            }
        });

        return itemsLiveData;
    }
```

MenuRepository.java (2/3)

```
MenuRepository.java
public LiveData<String> insertOrderToKitchen(String name, Double subtotal, String type) {
    MutableLiveData<String> insertOrderLiveData = new MutableLiveData<>();
    JSONObject jsonPayload = new JSONObject();
    try {
        jsonPayload.put( name: "NAME", name);
        jsonPayload.put( name: "SUBTOTAL", subtotal);
        jsonPayload.put( name: "TYPE", type);
    } catch (JSONException e) {
        Log.e( tag: "MenuRepository insertOrderToKitchen",
                msg: "Invalid order payload" + e.getMessage());
        postError("Invalid order payload");
        return insertOrderLiveData;
    }

    RequestBody requestBody = RequestBody.create(jsonPayload.toString(),
            MediaType.parse( $this$parse: "application/json"));
    Call<String> call = ApiClient.getApiService().insertOrderToKitchen(requestBody);
    call.enqueue(new Callback<>() {
        @Override
        public void onResponse(@NonNull Call<String> call, @NonNull Response<String> response) {
            if (response.isSuccessful() && response.body() != null && !response.body().isEmpty()) {
                insertOrderLiveData.postValue(response.body());
            } else {
                postError("Failed to insert order");
            }
        }

        @Override
        public void onFailure(@NonNull Call<String> call, @NonNull Throwable t) {
            postError(t.getMessage());
        }
    });

    return insertOrderLiveData;
}

public LiveData<String> insertItemToOrder(int item, int quantity, Double totalPrice) {
    MutableLiveData<String> insertItemLiveData = new MutableLiveData<>();
    JSONObject jsonPayload = new JSONObject();
    try {
        jsonPayload.put( name: "ITEM_ID", item);
        jsonPayload.put( name: "QUANTITY", quantity);
        jsonPayload.put( name: "TOTAL_PRICE", totalPrice);
    } catch (JSONException e) {
        Log.e( tag: "MenuRepository insertItemToOrder",
                msg: "Invalid order item payload" + e.getMessage());
        postError("Invalid order item payload");
        return insertItemLiveData;
    }

    RequestBody requestBody = RequestBody.create(jsonPayload.toString(),
            MediaType.parse( $this$parse: "application/json"));
    Call<String> call = ApiClient.getApiService().insertItemToOrder(requestBody);
    call.enqueue(new Callback<>() {
        @Override
        public void onResponse(@NonNull Call<String> call, @NonNull Response<String> response) {
            if (response.isSuccessful() && response.body() != null && !response.body().isEmpty()) {
                insertItemLiveData.postValue(response.body());
            } else {
                postError("Failed to insert item");
            }
        }

        @Override
        public void onFailure(@NonNull Call<String> call, @NonNull Throwable t) {
            postError(t.getMessage());
        }
    });

    return insertItemLiveData;
}
```

MenuRepository.java (3/3)

```
MenuRepository.java
    private void postError(String message) {
        List<String> currentErrors = errorLiveData.getValue();
        if (currentErrors == null) {
            currentErrors = new ArrayList<>();
        }
        currentErrors.add(message);
        errorLiveData.postValue(new ArrayList<>(currentErrors));
    }

    public LiveData<List<String>> getErrorLiveData() { return errorLiveData; }

    public void clearError() { errorLiveData.setValue(null); }
}
```

MenuTabFragment.java (1/2)

```
MenuTabFragment.java
package com.waiterpos.ui.waiter.menu;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ProgressBar;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.waiterpos.databinding.FragmentMenuTabBinding;

import java.util.ArrayList;

public class MenuTabFragment extends Fragment {
    2 usages
    private String categoryName;
    2 usages
    private String tax;
    5 usages
    private MenuViewModel viewModel;

    1 usage
    public static MenuTabFragment newInstance(String categoryName, String tax) {
        Log.i( tag: "MenuTabFragment",  msg: "categoryName: " + categoryName + " tax: " + tax);
        MenuTabFragment fragment = new MenuTabFragment();
        Bundle args = new Bundle();
        args.putString("CATEGORY_NAME", categoryName);
        args.putString("CATEGORY_TAX", tax);
        fragment.setArguments(args);
        return fragment;
    }
```

MenuTabFragment.java (2/2)

```java
MenuTabFragment.java
    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            categoryName = getArguments().getString( key: "CATEGORY_NAME");
            tax = getArguments().getString( key: "CATEGORY_TAX");
        }

        viewModel = new ViewModelProvider(requireActivity()).get(MenuViewModel.class);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        FragmentMenuTabBinding binding = FragmentMenuTabBinding.inflate(inflater, container,  attachToParent: false);
        RecyclerView recyclerView = binding.recyclerView;
        recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));

        ProgressBar progressBar = binding.progressBar;

        MenuItemAdapter adapter = new MenuItemAdapter(new ArrayList<>(), tax, ( String itemId,
                                                                                int quantity,
                                                                                double totalPrice) -> {
            viewModel.updateItemTotal(itemId, quantity, totalPrice);
        });
        recyclerView.setAdapter(adapter);

        viewModel.getItemsForCategory(categoryName).observe(getViewLifecycleOwner(),
                List<Item> items -> {
            progressBar.setVisibility(View.GONE);
            if (items != null) {
                recyclerView.setVisibility(View.VISIBLE);
                adapter.updateData(items);
            } else {
                recyclerView.setVisibility(View.GONE);
                Toast.makeText(getContext(),  text: "Failed to load items", Toast.LENGTH_SHORT).show();
            }
        });

        viewModel.getErrorLiveData().observe(getViewLifecycleOwner(),  List<String> error -> {
            if (error != null && !error.isEmpty()) {
                Toast.makeText(getContext(),  text: "Error Occurred", Toast.LENGTH_LONG).show();
                for (String er : error) {
                    Log.e( tag: "MainFragment",  msg: "Error: " + er);
                }
                viewModel.clearError();
            }
        });

        return binding.getRoot();
    }
}
```

MenuViewModel.java (1/2)

```
MenuViewModel.java
package com.waiterpos.ui.waiter.menu;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.ViewModel;

import com.waiterpos.entities.Category;
import com.waiterpos.entities.Item;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

4 usages
public class MenuViewModel extends ViewModel {
    6 usages
    private final MenuRepository menuRepository = new MenuRepository();

    3 usages
    private LiveData<List<Category>> categoriesLiveData;

    2 usages
    public LiveData<List<Category>> getCategories() {
        if (categoriesLiveData == null) {
            categoriesLiveData = menuRepository.getCategoryList();
        }
        return categoriesLiveData;
    }

    3 usages
    private final Map<String, LiveData<List<Item>>> itemsCache = new HashMap<>();

    1 usage
    public LiveData<List<Item>> getItemsForCategory(String category) {
        if (!itemsCache.containsKey(category)) {
            LiveData<List<Item>> liveData = menuRepository.getItemsList(category);
            itemsCache.put(category, liveData);
        }
        return itemsCache.get(category);
    }

    1 usage
    public LiveData<String> insertOrderToKitchen(String name, double subtotal, String type) {
        return menuRepository.insertOrderToKitchen(name, subtotal, type);
    }

    1 usage
    public LiveData<String> insertItemToOrder(int itemId, int quantity, double totalPrice) {
        return menuRepository.insertItemToOrder(itemId, quantity, totalPrice);
    }

    //Error Handling
    public LiveData<List<String>> getErrorLiveData() { return menuRepository.getErrorLiveData(); }

    public void clearError() { menuRepository.clearError(); }

    //Subtotal Logic
    3 usages
    private final MutableLiveData<Double> subtotalMLD = new MutableLiveData<>( value: 0.0);
    1 usage
    public LiveData<Double> subtotalLiveData = subtotalMLD;

    5 usages
    private final Map<String, Double> itemTotals = new HashMap<>();
    4 usages
    private final Map<String, Integer> itemQuantities = new HashMap<>();
```

MenuViewModel.java (2/2)

MenuViewModel.java

```
    public void updateItemTotal(String itemId, int quantity, double totalPrice) {
        if (quantity > 0) {
            itemQuantities.put(itemId, quantity);
            itemTotals.put(itemId, totalPrice);
        } else {
            itemQuantities.remove(itemId);
            itemTotals.remove(itemId);
        }
        double newSubtotal = 0;
        for (double price : itemTotals.values()) {
            newSubtotal += price;
        }
        subtotalMLD.postValue(newSubtotal);
    }

    2 usages
    public void resetSubtotal() {
        itemTotals.clear();
        itemQuantities.clear();
        subtotalMLD.postValue(0.0);
    }

    1 usage
    public Map<String, Double> getItemTotals() { return new HashMap<>(itemTotals); }

    1 usage
    public Map<String, Integer> getItemQuantities() { return new HashMap<>(itemQuantities); }

    2 usages
    private final MutableLiveData<Integer> tablesMLD = new MutableLiveData<>();
    1 usage
    public LiveData<Integer> tablesLiveData = tablesMLD;

    1 usage
    public void setTablesMLD(int order) { tablesMLD.setValue(order); }
}
```

B.5.3.2 Orders

OrdersFragment.java (1/4)

OrdersFragment.java

```java
package com.waiterpos.ui.waiter.orders;

import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.swiperefreshlayout.widget.SwipeRefreshLayout;

import com.waiterpos.R;
import com.waiterpos.databinding.FragmentOrdersBinding;
import com.waiterpos.entities.Kitchen;
import com.waiterpos.entities.Orders;
import com.google.android.material.navigation.NavigationView;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Objects;

public class OrdersFragment extends Fragment {
    14 usages
    private FragmentOrdersBinding binding;
    6 usages
    private Spinner orderSpinner;
    5 usages
    private OrdersItemAdapter itemAdapter;
    19 usages
    private OrdersViewModel ordersViewModel;
    3 usages
    private int lastSelectedPosition = 0;
    4 usages
    private boolean keepSpinnerPosition = false;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ordersViewModel = new ViewModelProvider(requireActivity()).get(OrdersViewModel.class);
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        binding = FragmentOrdersBinding.inflate(inflater, container,  attachToParent: false);

        orderSpinner = binding.orderSpinner;

        setupRecyclerView();
        setupObservers();

        refreshOrders();

        SwipeRefreshLayout swipeRefreshLayout = binding.swipeRefreshLayout;

        swipeRefreshLayout.setOnRefreshListener(() -> {
            keepSpinnerPosition = true;
            refreshOrders();

            swipeRefreshLayout.setRefreshing(false);
        });
        binding.voidButton.setOnClickListener( View v -> voidOrder());
        binding.cancelButton.setOnClickListener( View v -> cancelOrder());
        binding.payButton.setOnClickListener( View v -> payOrder());

        return binding.getRoot();
    }
```

OrdersFragment.java (2/4)

```
OrdersFragment.java
private void setupRecyclerView() {
    RecyclerView orderRecyclerView = binding.orderRecyclerView;
    orderRecyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
    itemAdapter = new OrdersItemAdapter(new ArrayList<>(),
            Kitchen item -> ordersViewModel.setSelectedItemLD(item));
    orderRecyclerView.setAdapter(itemAdapter);
}

1 usage
private void setupObservers() {
    ordersViewModel.getActiveOrders().observe(getViewLifecycleOwner(), List<Orders> ordersList -> {
        if (ordersList == null) {
            Toast.makeText(getContext(), text: "Failed to load orders", Toast.LENGTH_SHORT).show();
            return;
        }
        if (ordersList.isEmpty()) {
            orderSpinner.setAdapter(null);

            binding.valueSubtotalTextView.setText("");
            binding.valueTypeTextView.setText("");
            binding.valueDateTimeTextView.setText("");

            itemAdapter.updateItems(Collections.emptyList());
            return;
        }

        ordersList.removeIf( Orders order -> "Takeaway".equals(order.getType()));

        List<String> orderNames = new ArrayList<>();
        for (Orders order : ordersList) {
            orderNames.add("Order " + order.getOrderID() + " " + order.getStatus());
        }

        ArrayAdapter<String> adapter = new ArrayAdapter<>(requireContext(),
                android.R.layout.simple_spinner_item, orderNames);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        orderSpinner.setAdapter(adapter);

        if (keepSpinnerPosition && lastSelectedPosition < ordersList.size()) {
            orderSpinner.setSelection(lastSelectedPosition);
        } else {
            orderSpinner.setSelection(0);
        }
        keepSpinnerPosition = false;

        orderSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            no usages
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                lastSelectedPosition = position;
                Orders selectedOrder = ordersList.get(position);
                ordersViewModel.setSelectedItemLD(null);
                ordersViewModel.setSelectedOrderLD(selectedOrder);
                int orderID = Integer.parseInt(selectedOrder.getOrderID());
                fetchKitchenData(orderID);
            }

            no usages
            @Override
            public void onNothingSelected(AdapterView<?> parent) {
                itemAdapter.updateItems(new ArrayList<>());
            }
        });
    });
```

OrdersFragment.java (3/4)

OrdersFragment.java

```
    // Observe selected order
    ordersViewModel.selectedOrderLD.observe(getViewLifecycleOwner(), Orders order -> {
        if (order != null) {
            binding.valueSubtotalTextView.setText(order.getSubtotal());
            binding.valueTypeTextView.setText(order.getType());
            binding.valueDateTimeTextView.setText(order.getOrderDate());
        }
    });

    // Repository errors
    ordersViewModel.getErrorLiveData().observe(getViewLifecycleOwner(), List<String> error -> {
        if (error != null && !error.isEmpty()) {
            Toast.makeText(getContext(), text: "Error Occurred", Toast.LENGTH_LONG).show();
            for (String er : error) {
                Log.e( tag: "OrdersFragment", msg: "Error: " + er);
            }
            ordersViewModel.clearError();
        }
    });
}

1 usage
private void fetchKitchenData(int orderId) {
    ordersViewModel.getKitchenItems(orderId).observe(getViewLifecycleOwner(), List<Kitchen> kitchens -> {
        if (kitchens != null) {
            itemAdapter.updateItems(kitchens);
        } else {
            Toast.makeText(getContext(), text: "Failed to load kitchen data", Toast.LENGTH_SHORT).show();
        }
    });
}

1 usage
private void voidOrder() {
    Orders selectedOrder = ordersViewModel.selectedOrderLD.getValue();
    Kitchen selectedItem = ordersViewModel.selectedItemLD.getValue();

    if (selectedOrder == null || selectedItem == null) {
        Toast.makeText(getContext(), text: "Select order and item", Toast.LENGTH_SHORT).show();
        return;
    }
    int orderID = Integer.parseInt(selectedOrder.getOrderID());
    Log.i( tag: "voidOrder", msg: "voidOrder: " + selectedOrder.getOrderID()
            + " selectedItem: " + selectedItem.getName());

    ordersViewModel.voidKitchen(orderID, selectedItem.getName()).observe(getViewLifecycleOwner(),
            String response -> {
        Toast.makeText(getContext(), Objects.requireNonNullElse(response,
                defaultObj: "Failed to void item"), Toast.LENGTH_SHORT).show();
        ordersViewModel.setSelectedItemLD(null);
        keepSpinnerPosition = true;
        refreshOrders();
    });
}
```

OrdersFragment.java (4/4)

OrdersFragment.java

```
    private void cancelOrder() {
        Orders selectedOrder = ordersViewModel.selectedOrderLD.getValue();
        if (selectedOrder == null) {
            Toast.makeText(getContext(), text: "Select order first", Toast.LENGTH_SHORT).show();
            return;
        }
        int orderID = Integer.parseInt(selectedOrder.getOrderID());
        Log.i( tag: "cancelOrder",  msg: "cancelOrder: " + selectedOrder.getOrderID());

        ordersViewModel.cancelOrder(orderID).observe(getViewLifecycleOwner(),  String response -> {
            Toast.makeText(getContext(), Objects.requireNonNullElse(response,
                    defaultObj: "Failed to cancel order"), Toast.LENGTH_SHORT).show();
            ordersViewModel.setSelectedOrderLD(null);
            ordersViewModel.setSelectedItemLD(null);
            refreshOrders();
        });
    }

    4 usages
    private void refreshOrders() { ordersViewModel.refreshOrders(); }

    1 usage
    private void payOrder() {
        Orders selectedOrder = ordersViewModel.selectedOrderLD.getValue();
        if (selectedOrder == null) {
            Toast.makeText(getContext(), text: "No order selected", Toast.LENGTH_SHORT).show();
            return;
        }

        Bundle args = new Bundle();
        args.putString("ORDER_ID", selectedOrder.getOrderID());
        NavController navController = Navigation.findNavController(requireView());
        navController.navigate(R.id.nav_register, args);

        NavigationView navigationView = requireActivity().findViewById(R.id.nav_view);
        Menu menu = navigationView.getMenu();

        menu.findItem(R.id.nav_menu).setEnabled(false);
        menu.findItem(R.id.nav_orders).setEnabled(false);
    }
}
```

OrdersItemAdapter.java

```
OrdersItemAdapter.java
package com.waiterpos.ui.waiter.orders;

import android.graphics.Color;
import android.view.LayoutInflater;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.waiterpos.databinding.OrdersItemAdapterBinding;
import com.waiterpos.entities.Kitchen;

import java.util.List;

4 usages
public class OrdersItemAdapter extends RecyclerView.Adapter<OrdersItemAdapter.OrderViewHolder> {
    5 usages
    private final List<Kitchen> items;
    2 usages
    private final OnItemClickListener listener;
    5 usages
    private int selectedItemPosition = RecyclerView.NO_POSITION;

    1 usage
    public OrdersItemAdapter(List<Kitchen> items, OnItemClickListener listener) {
        this.items = items;
        this.listener = listener;
    }

    @NonNull
    @Override
    public OrderViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        OrdersItemAdapterBinding binding = OrdersItemAdapterBinding.inflate(
                LayoutInflater.from(parent.getContext()), parent,  attachToParent: false);
        return new OrdersItemAdapter.OrderViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(@NonNull OrderViewHolder holder, int position) {
        holder.bind(items.get(position), position);
    }

    @Override
    public int getItemCount() { return items.size(); }

    3 usages
    public void updateItems(List<Kitchen> newItems) {
        items.clear();
        items.addAll(newItems);
        selectedItemPosition = RecyclerView.NO_POSITION;
        notifyDataSetChanged();
    }

    2 usages
    public interface OnItemClickListener {
        1 usage
        void onItemClick(Kitchen item);
    }

    4 usages
    class OrderViewHolder extends RecyclerView.ViewHolder {
        4 usages
        private final OrdersItemAdapterBinding binding;

        1 usage
        public OrderViewHolder(OrdersItemAdapterBinding binding) {
            super(binding.getRoot());
            this.binding = binding;
        }

        1 usage
        public void bind(Kitchen item, int position) {
            binding.valueItemNameTextView.setText(item.getName());
            binding.valueQuantityTextView.setText(item.getQuantity());
            binding.valueStatusTextView.setText(item.getStatus());

            itemView.setBackgroundColor(position ==
                    selectedItemPosition ? Color.LTGRAY : Color.TRANSPARENT);

            itemView.setOnClickListener( View v -> {
                int previousPosition = selectedItemPosition;
                selectedItemPosition = getAdapterPosition();
                notifyItemChanged(previousPosition);
                notifyItemChanged(selectedItemPosition);
                listener.onItemClick(item);
            });
        }
    }
}
```

OrdersRepository.java (1/2)

OrdersRepository.java

```java
package com.waiterpos.ui.waiter.orders;

import android.util.Log;

import androidx.annotation.NonNull;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;

import com.waiterpos.api.ApiClient;
import com.waiterpos.entities.Kitchen;
import com.waiterpos.entities.Orders;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

import okhttp3.MediaType;
import okhttp3.RequestBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

2 usages
public class OrdersRepository {
    4 usages
    private final MutableLiveData<List<String>> errorLiveData = new MutableLiveData<>();

    1 usage
    public LiveData<List<Orders>> getActiveOrdersList() {
        MutableLiveData<List<Orders>> activeOrdersLiveData = new MutableLiveData<>();
        Call<List<Orders>> call = ApiClient.getApiService().getActiveOrdersList();
        call.enqueue(new Callback<>() {
            @Override
            public void onResponse(@NonNull Call<List<Orders>> call,
                                   @NonNull Response<List<Orders>> response) {
                if (response.isSuccessful() && response.body() != null) {
                    activeOrdersLiveData.postValue(response.body());
                }
            }

            @Override
            public void onFailure(@NonNull Call<List<Orders>> call, @NonNull Throwable t) {
                postError(t.getMessage());
            }
        });
        return activeOrdersLiveData;
    }

    1 usage
    public LiveData<List<Kitchen>> getKitchenList(String orderId) {
        MutableLiveData<List<Kitchen>> kitchenItemsLiveData = new MutableLiveData<>();
        Call<List<Kitchen>> call = ApiClient.getApiService().getKitchenList(orderId);
        call.enqueue(new Callback<>() {
            @Override
            public void onResponse(@NonNull Call<List<Kitchen>> call,
                                   @NonNull Response<List<Kitchen>> response) {
                if (response.isSuccessful() && response.body() != null) {
                    kitchenItemsLiveData.postValue(response.body());
                } else {
                    postError("No items found for the order id: " + orderId);
                }
            }

            @Override
            public void onFailure(@NonNull Call<List<Kitchen>> call, @NonNull Throwable t) {
                postError(t.getMessage());
            }
        });
        return kitchenItemsLiveData;
    }
```

OrdersRepository.java (2/2)

```
OrdersRepository.java
    public LiveData<String> voidKitchen(int orderID, String itemName) {
        MutableLiveData<String> voidItemLiveData = new MutableLiveData<>();
        JSONObject jsonPayload = new JSONObject();
        try {
            jsonPayload.put( name: "ORDER_ID", orderID);
            jsonPayload.put( name: "NAME", itemName);
        } catch (JSONException e) {
            Log.e( tag: "OrdersRepository voidKitchen",  msg: "Invalid order payload" + e.getMessage());
            postError("Invalid kitchen payload");
            return voidItemLiveData;
        }
        RequestBody requestBody = RequestBody.create(jsonPayload.toString(),
                MediaType.parse( $this$parse: "application/json"));

        Call<String> call = ApiClient.getApiService().voidKitchen(requestBody);
        call.enqueue(new Callback<>() {
            @Override
            public void onResponse(@NonNull Call<String> call, @NonNull Response<String> response) {
                if (response.isSuccessful() && response.body() != null && !response.body().isEmpty()) {
                    voidItemLiveData.postValue(response.body());
                } else {
                    postError("Failed to void item");
                }
            }

            @Override
            public void onFailure(@NonNull Call<String> call, @NonNull Throwable t) {
                postError(t.getMessage());
            }
        });

        return voidItemLiveData;
    }

    1 usage
    public LiveData<String> cancelOrder(int orderID) {
        MutableLiveData<String> cancelOrderLiveData = new MutableLiveData<>();
        JSONObject jsonPayload = new JSONObject();
        try {
            jsonPayload.put( name: "ORDER_ID", orderID);
        } catch (JSONException e) {
            Log.e( tag: "OrdersRepository voidKitchen",  msg: "Invalid order payload" + e.getMessage());
            postError("Invalid order payload");
            return cancelOrderLiveData;
        }
        RequestBody requestBody = RequestBody.create(jsonPayload.toString(),
                MediaType.parse( $this$parse: "application/json"));

        Call<String> call = ApiClient.getApiService().cancelOrder(requestBody);
        call.enqueue(new Callback<>() {
            @Override
            public void onResponse(@NonNull Call<String> call, @NonNull Response<String> response) {
                if (response.isSuccessful() && response.body() != null && !response.body().isEmpty()) {
                    cancelOrderLiveData.postValue(response.body());
                } else {
                    postError("Failed to cancel the order");
                }
            }

            @Override
            public void onFailure(@NonNull Call<String> call, @NonNull Throwable t) {
                postError(t.getMessage());
            }
        });

        return cancelOrderLiveData;
    }

    9 usages
    private void postError(String message) {
        List<String> currentErrors = errorLiveData.getValue();
        if (currentErrors == null) currentErrors = new ArrayList<>();
        currentErrors.add(message);
        errorLiveData.postValue(new ArrayList<>(currentErrors));
    }

    public LiveData<List<String>> getErrorLiveData() { return errorLiveData; }

    public void clearError() { errorLiveData.setValue(null); }
}
```

OrdersViewModel.java

```java
package com.waiterpos.ui.waiter.orders;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.Transformations;
import androidx.lifecycle.ViewModel;

import com.waiterpos.entities.Kitchen;
import com.waiterpos.entities.Orders;

import java.util.List;

5 usages
public class OrdersViewModel extends ViewModel {
    6 usages
    private final OrdersRepository ordersRepository = new OrdersRepository();

    2 usages
    private final MutableLiveData<Boolean> refreshTrigger = new MutableLiveData<>( value: false);

    1 usage
    public LiveData<List<Orders>> getActiveOrders() {
        return Transformations.switchMap(refreshTrigger, Boolean trigger -> {
            if (Boolean.TRUE.equals(trigger)) {
                return ordersRepository.getActiveOrdersList();
            } else {
                return new MutableLiveData<>( value: null);
            }
        });
    }

    1 usage
    public void refreshOrders() { refreshTrigger.setValue(true); }

    1 usage
    public LiveData<List<Kitchen>> getKitchenItems(int orderId) {
        return ordersRepository.getKitchenList(String.valueOf(orderId));
    }

    1 usage
    public LiveData<String> voidKitchen(int itemId, String itemName) {
        return ordersRepository.voidKitchen(itemId, itemName);
    }

    2 usages
    public LiveData<String> cancelOrder(int itemId) { return ordersRepository.cancelOrder(itemId); }

    //Error Handling
    public LiveData<List<String>> getErrorLiveData() { return ordersRepository.getErrorLiveData(); }

    public void clearError() { ordersRepository.clearError(); }

    2 usages
    private final MutableLiveData<Orders> selectedOrderMLD = new MutableLiveData<>();
    2 usages
    private final MutableLiveData<Kitchen> selectedItemMLD = new MutableLiveData<>();
    4 usages
    public LiveData<Orders> selectedOrderLD = selectedOrderMLD;
    1 usage
    public LiveData<Kitchen> selectedItemLD = selectedItemMLD;

    2 usages
    public void setSelectedOrderLD(Orders order) { selectedOrderMLD.setValue(order); }

    4 usages
    public void setSelectedItemLD(Kitchen item) { selectedItemMLD.setValue(item); }
}
```

B.5.3.3 Register

RegisterFragment.java (1/4)

RegisterFragment.java

```java
package com.waiterpos.ui.waiter.register;

import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;
import androidx.lifecycle.ViewModelProvider;
import androidx.navigation.NavController;
import androidx.navigation.Navigation;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.waiterpos.R;
import com.waiterpos.databinding.FragmentRegisterBinding;
import com.waiterpos.entities.Register;
import com.waiterpos.ui.waiter.orders.OrdersViewModel;
import com.google.android.material.navigation.NavigationView;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.List;
import java.util.Objects;

public class RegisterFragment extends Fragment {
    12 usages
    private FragmentRegisterBinding binding;
    7 usages
    private EditText subtotalEditText, tipEditText, discountEditText, totalEditText, receivedEditText;
    3 usages
    private RecyclerView orderRecyclerView;
    9 usages
    private String orderID;
    7 usages
    private RegisterViewModel registerViewModel;
    2 usages
    private OrdersViewModel ordersViewModel;

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        binding = FragmentRegisterBinding.inflate(inflater, container,  attachToParent: false);

        orderRecyclerView = binding.recyclerView2;
        subtotalEditText = binding.subtotalEditTextNumberDecimal;
        tipEditText = binding.tipEditTextNumberDecimal;
        discountEditText = binding.discountEditTextNumberDecimal4;
        totalEditText = binding.totalEditTextNumberDecimal;
        receivedEditText = binding.receivedAmountEditTextNumberDecimal;

        registerViewModel = new ViewModelProvider( owner: this).get(RegisterViewModel.class);
        ordersViewModel = new ViewModelProvider(requireActivity()).get(OrdersViewModel.class);

        Bundle args = getArguments();
        if (args != null) {
            orderID = args.getString( key: "ORDER_ID");
            binding.orderIDTextView.setText("Order ID: " + orderID);
        }
```

RegisterFragment.java (2/4)

RegisterFragment.java

```
    binding.payOrderButton.setOnClickListener( View v -> payOrder());
    binding.cancelOrderButton.setOnClickListener( View v -> cancelOrder());

    registerViewModel.getRegisterList().observe(getViewLifecycleOwner(), List<Register> list -> {
        if (list != null) {
            setupRecyclerView(list);
        } else {
            Log.e( tag: "RegisterFragment", msg: "Failed to load register list");
        }
    });

    registerViewModel.getCompleteOrderResponse().observe(getViewLifecycleOwner(),
            String response -> {
        if (response != null) {
            Toast.makeText(getContext(), response, Toast.LENGTH_LONG).show();
            Log.i( tag: "RegisterFragment", msg: "payOrder response: " + response);
        } else {
            Toast.makeText(getContext(), text: "Error completing order", Toast.LENGTH_LONG).show();
            Log.e( tag: "RegisterFragment", msg: "payOrder response is null");
        }
        NavController navController = Navigation.findNavController(requireView());
        navController.navigate(R.id.nav_menu);
    });

    registerViewModel.getErrorLiveData().observe(getViewLifecycleOwner(), List<String> error -> {
        if (error != null && !error.isEmpty()) {
            Toast.makeText(getContext(), text: "Error Occurred", Toast.LENGTH_LONG).show();
            for (String er : error) {
                Log.e( tag: "MainFragment", msg: "Error: " + er);
            }
            registerViewModel.clearError();
        }
    });

    if (orderID != null) {
        registerViewModel.fetchRegisterList(orderID);
    }

    subtotalEditText.addTextChangedListener(totalCalculatorListener());
    tipEditText.addTextChangedListener(totalCalculatorListener());
    discountEditText.addTextChangedListener(totalCalculatorListener());
    receivedEditText.addTextChangedListener(changeCalculatorListener());

    return binding.getRoot();
}

1 usage
private void setupRecyclerView(List<Register> registerList) {
    RegisterItemAdapter itemAdapter = new RegisterItemAdapter(registerList);
    orderRecyclerView.setLayoutManager(new LinearLayoutManager(getContext()));
    orderRecyclerView.setAdapter(itemAdapter);

    // Calculate subtotal
    double subtotal = itemAdapter.getTotalPrice();
    BigDecimal bd = new BigDecimal(subtotal).setScale( newScale: 2, RoundingMode.HALF_UP);
    subtotalEditText.setText(String.valueOf(bd.doubleValue()));
}
```

RegisterFragment.java (3/4)

```
RegisterFragment.java
private void payOrder() {
    if (orderID == null) {
        Toast.makeText(getContext(),  text: "No order selected", Toast.LENGTH_LONG).show();
        return;
    }
    String subtotal = ((!subtotalEditText.getText().toString().isEmpty())
            ? subtotalEditText.getText().toString() : "0.0");
    String discount = ((!discountEditText.getText().toString().isEmpty())
            ? discountEditText.getText().toString() : "0.0");
    String tip = ((!tipEditText.getText().toString().isEmpty())
            ? tipEditText.getText().toString() : "0.0");
    String total = ((!totalEditText.getText().toString().isEmpty())
            ? totalEditText.getText().toString() : "0.0");

    if ("0.0".equals(subtotal) || "0.0".equals(total)) {
        Toast.makeText(getContext(),  text: "Subtotal/Total must be greater than zero",
                Toast.LENGTH_LONG).show();
        return;
    }

    registerViewModel.completeOrder(orderID, subtotal, discount, tip, total);
}

1 usage
private void cancelOrder() {
    if (orderID == null) {
        Toast.makeText(getContext(),  text: "No order selected", Toast.LENGTH_SHORT).show();
        return;
    }
    int orderIdInt = Integer.parseInt(orderID);
    Log.i( tag: "cancelOrder",  msg: "cancelOrder: " + orderID);
    ordersViewModel.cancelOrder(orderIdInt).observe(getViewLifecycleOwner(),  String response -> {
        Toast.makeText(getContext(), Objects.requireNonNullElse(response,
                defaultObj: "Failed to cancel order"), Toast.LENGTH_SHORT).show();
        NavController navController = Navigation.findNavController(requireView());
        navController.navigate(R.id.nav_menu);
    });
}

3 usages
private TextWatcher totalCalculatorListener() {
    return new TextWatcher() {
        no usages
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {
            calculateTotal();
        }

        no usages
        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
            calculateTotal();
        }

        no usages
        @Override
        public void afterTextChanged(Editable s) { calculateTotal(); }

        3 usages
        public void calculateTotal() {
            double subtotal = 0;
            double tip = 0;
            double discount = 0;
            String subtotalText = subtotalEditText.getText().toString().isBlank() ? "0"
                    : subtotalEditText.getText().toString();
            String discountText = discountEditText.getText().toString().isBlank() ? "0"
                    : discountEditText.getText().toString();
            String tipText = tipEditText.getText().toString().isBlank() ? "0"
                    : tipEditText.getText().toString();
            if (isValid(subtotalText,  type: "Subtotal") && isValid(discountText,
                    type: "Discount") && isValid(tipText,  type: "Tip")) {
                subtotal = Double.parseDouble(subtotalText);
                tip = Double.parseDouble(tipText);
                discount = Double.parseDouble(discountText);
            }
            double total = subtotal + tip - discount;

            BigDecimal bd = new BigDecimal(total);
            bd = bd.setScale( newScale: 2, RoundingMode.HALF_UP);
            double roundedValue = bd.doubleValue();

            totalEditText.setText(String.valueOf(roundedValue));
        }
    };
}
```

RegisterFragment.java (4/4)

```
RegisterFragment.java
    private TextWatcher changeCalculatorListener() {
        return new TextWatcher() {
            no usages
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {
                calculateChange();
            }

            no usages
            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                calculateChange();
            }

            no usages
            @Override
            public void afterTextChanged(Editable s) { calculateChange(); }

            3 usages
            public void calculateChange() {
                double total = 0;
                double received = 0;
                String totalText = totalEditText.getText().toString().isBlank() ? "0"
                        : totalEditText.getText().toString();
                String receivedText = receivedEditText.getText().toString().isBlank() ? "0"
                        : receivedEditText.getText().toString();
                if (isValid(totalText, type: "Total") && isValid(receivedText, type: "Received")) {
                    total = Double.parseDouble(totalText);
                    received = Double.parseDouble(receivedText);
                }
                double change = received - total;

                BigDecimal bd = new BigDecimal(change);
                bd = bd.setScale( newScale: 2, RoundingMode.HALF_UP);
                double roundedValue = bd.doubleValue();

                binding.changeValueTextView.setText(String.valueOf(roundedValue));
            }
        };
    }

    5 usages
    private boolean isValid(String price, String type) {
        if (price.isBlank()) {
            Toast.makeText(getContext(), text: type + " must be not empty", Toast.LENGTH_LONG).show();
            return false;
        }
        double doublePrice;
        try {
            doublePrice = Double.parseDouble(price);
            if (doublePrice < 0) {
                Toast.makeText(getContext(), text: type + " must be more than zero",
                        Toast.LENGTH_LONG).show();
                return false;
            }
        } catch (NumberFormatException exception) {
            Toast.makeText(getContext(), text: type + " must be number", Toast.LENGTH_LONG).show();
            return false;
        }
        return true;
    }

    15 usages
    @Override
    public void onDestroyView() {
        super.onDestroyView();

        NavigationView navigationView = requireActivity().findViewById(R.id.nav_view);
        Menu menu = navigationView.getMenu();
        menu.findItem(R.id.nav_menu).setEnabled(true);
        menu.findItem(R.id.nav_orders).setEnabled(true);
    }
}
```

RegisterItemAdapter.java

```java
RegisterItemAdapter.java
package com.waiterpos.ui.waiter.register;

import android.view.LayoutInflater;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.waiterpos.databinding.RegisterItemAdapterBinding;
import com.waiterpos.entities.Register;

import java.util.List;

3 usages
public class RegisterItemAdapter extends RecyclerView.Adapter<RegisterItemAdapter.ItemViewHolder> {

    4 usages
    private final List<Register> items;

    1 usage
    public RegisterItemAdapter(List<Register> items) { this.items = items; }

    @NonNull
    @Override
    public ItemViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        RegisterItemAdapterBinding binding = RegisterItemAdapterBinding.inflate(
                LayoutInflater.from(parent.getContext()), parent,  attachToParent: false);
        return new ItemViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(@NonNull ItemViewHolder holder, int position) {
        holder.bind(items.get(position));
    }

    @Override
    public int getItemCount() { return items.size(); }

    1 usage
    public double getTotalPrice() {
        double total = 0;
        for (Register item : items) {
            double totalPrice = Double.parseDouble(item.getTotalPrice());
            total += totalPrice;
        }
        return total;
    }

    4 usages
    class ItemViewHolder extends RecyclerView.ViewHolder {
        5 usages
        private final RegisterItemAdapterBinding binding;

        1 usage
        public ItemViewHolder(RegisterItemAdapterBinding binding) {
            super(binding.getRoot());
            this.binding = binding;
        }

        1 usage
        public void bind(Register item) {
            binding.itemValueTextView.setText(item.getName());
            binding.qtyValueTextView.setText(String.valueOf(item.getQuantity()));
            binding.taxValueTextView.setText(item.getTax() + " %");
            binding.priceValueTextView.setText(String.valueOf(item.getTotalPrice()));
        }
    }
}
```

RegisterRepository.java (1/2)

RegisterRepository.java

```java
package com.waiterpos.ui.waiter.register;

import android.util.Log;

import androidx.annotation.NonNull;
import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;

import com.waiterpos.api.ApiClient;
import com.waiterpos.entities.Register;

import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

import okhttp3.MediaType;
import okhttp3.RequestBody;
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

2 usages
public class RegisterRepository {

    4 usages
    private final MutableLiveData<List<String>> errorLiveData = new MutableLiveData<>();

    1 usage
    public LiveData<List<Register>> getRegisterList(String orderId) {
        MutableLiveData<List<Register>> data = new MutableLiveData<>();

        Call<List<Register>> call = ApiClient.getApiService().getRegisterList(orderId);
        call.enqueue(new Callback<>() {
            @Override
            public void onResponse(@NonNull Call<List<Register>> call,
                                   @NonNull Response<List<Register>> response) {
                if (response.isSuccessful() && response.body() != null
                        && !response.body().isEmpty()) {
                    data.postValue(response.body());
                } else {
                    postError("No items found for the order id: " + orderId);
                }
            }

            @Override
            public void onFailure(@NonNull Call<List<Register>> call, @NonNull Throwable t) {
                postError(t.getMessage());
            }
        });

        return data;
    }

    1 usage
    public LiveData<String> completeOrder(int orderId, double subtotal, double discount,
                                          double tip, double total) {
        MutableLiveData<String> data = new MutableLiveData<>();

        JSONObject jsonPayload = new JSONObject();
        try {
            jsonPayload.put( name: "ORDER_ID", orderId);
            jsonPayload.put( name: "SUBTOTAL", subtotal);
            jsonPayload.put( name: "DISCOUNT", discount);
            jsonPayload.put( name: "TIP", tip);
            jsonPayload.put( name: "TOTAL", total);
        } catch (JSONException e) {
            Log.e( tag: "RegisterRepository completeOrder",
                    msg: "Invalid order payload" + e.getMessage());
            postError("Invalid order payload");
            return data;
        }
```

RegisterRepository.java (2/2)

```
RegisterRepository.java
        RequestBody requestBody = RequestBody.create(jsonPayload.toString(),
                MediaType.parse( $this$parse: "application/json"));

        Call<String> call = ApiClient.getApiService().completeOrder(requestBody);
        call.enqueue(new Callback<>() {
            @Override
            public void onResponse(@NonNull Call<String> call, @NonNull Response<String> response) {
                if (response.isSuccessful() && response.body() != null
                        && !response.body().isEmpty()) {
                    data.postValue(response.body());
                } else {
                    postError("Failed to complete the order");
                }
            }

            @Override
            public void onFailure(@NonNull Call<String> call, @NonNull Throwable t) {
                postError(t.getMessage());
            }
        });

        return data;
    }

    5 usages
    private void postError(String message) {
        List<String> currentErrors = errorLiveData.getValue();
        if (currentErrors == null) currentErrors = new ArrayList<>();
        currentErrors.add(message);
        errorLiveData.postValue(new ArrayList<>(currentErrors));
    }

    public LiveData<List<String>> getErrorLiveData() { return errorLiveData; }

    public void clearError() { errorLiveData.setValue(null); }
}
```

RegisterViewModel.java

RegisterViewModel.java

```java
package com.waiterpos.ui.waiter.register;

import androidx.lifecycle.LiveData;
import androidx.lifecycle.MutableLiveData;
import androidx.lifecycle.Transformations;
import androidx.lifecycle.ViewModel;

import com.waiterpos.entities.Register;

import java.util.List;

2 usages
public class RegisterViewModel extends ViewModel {

    4 usages
    private final RegisterRepository registerRepository = new RegisterRepository();

    2 usages
    private final MutableLiveData<String> registerListTrigger = new MutableLiveData<>();

    1 usage
    public LiveData<List<Register>> getRegisterList() {
        return Transformations.switchMap(registerListTrigger, String orderId -> {
            if (orderId != null && !orderId.isEmpty()) {
                return registerRepository.getRegisterList(orderId);
            } else {
                return new MutableLiveData<>( value: null);
            }
        });
    }

    1 usage
    public void fetchRegisterList(String orderId) { registerListTrigger.setValue(orderId); }

    2 usages
    private final MutableLiveData<String[]> completeOrderTrigger = new MutableLiveData<>();

    1 usage
    public void completeOrder(String orderId, String subtotal, String discount, String tip, String total) {
        completeOrderTrigger.setValue(new String[]{orderId, subtotal, discount, tip, total});
    }

    1 usage
    public LiveData<String> getCompleteOrderResponse() {
        return Transformations.switchMap(completeOrderTrigger, String[] params -> {
            if (params != null && params.length == 5) {
                int orderId = Integer.parseInt(params[0]);
                double subtotal = Double.parseDouble(params[1]);
                double discount = Double.parseDouble(params[2]);
                double tip = Double.parseDouble(params[3]);
                double total = Double.parseDouble(params[4]);
                return registerRepository.completeOrder(orderId, subtotal, discount, tip, total);
            }
            return new MutableLiveData<>( value: null);
        });
    }

    public LiveData<List<String>> getErrorLiveData() {
        return registerRepository.getErrorLiveData();
    }

    public void clearError() { registerRepository.clearError(); }
}
```