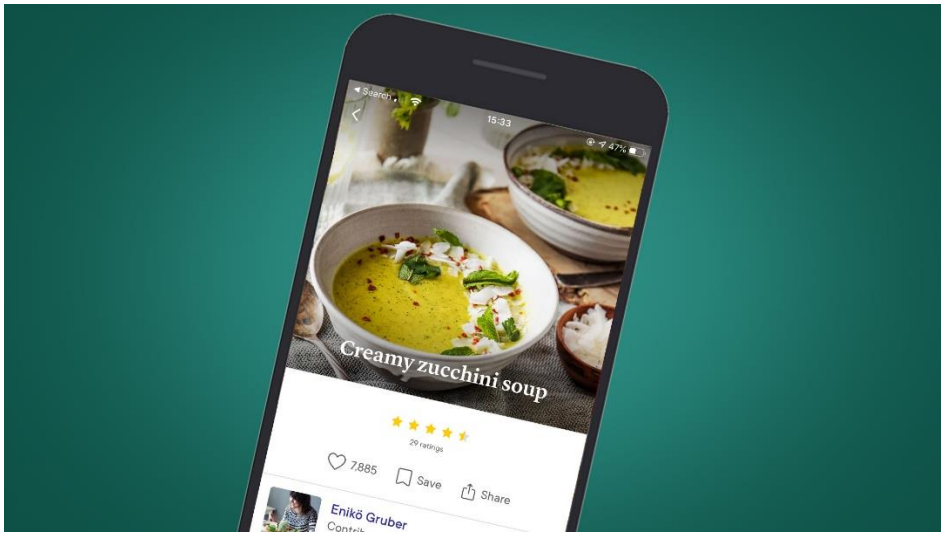
ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Εφαρμογή Ιστού για Συνταγές»



Της φοιτήτριας
Μόσχου Σουλτάνα
Αρ. Μητρώου: 185234

Επιβλέπων Ονοματεπώνυμο
Ουγιάρογλου Στέφανος
Βαθμίδα: Επίκουρος Καθηγητής

Ημερομηνία 22-1-26

Τίτλος Δ.Ε.: Εφαρμογή Ιστού για Μαγειρικές Συνταγές
Κωδικός Δ.Ε.: 25220
Ονοματεπώνυμο φοιτήτριας: Σουλτάνα Μόσχου
Ονοματεπώνυμο εισηγητή: Στέφανος Ουγιάρογλου
Ημερομηνία ανάληψης Δ.Ε. 30-03-2025
Ημερομηνία περάτωσης Δ.Ε. 22-1-26

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.

Πρόλογος

Η παρούσα πτυχιακή εργασία εκπονήθηκε στα πλαίσια της ολοκλήρωσης των σπουδών μου. Το αντικείμενο της εργασίας, η ανάπτυξη της εφαρμογής «Simply Served», προέκυψε από το ενδιαφέρον μου για τον συνδυασμό των σύγχρονων τεχνολογιών Web Development με την ανάλυση δεδομένων και τα συστήματα συστάσεων.

Στην εποχή της πληροφορίας, ο όγκος των διαθέσιμων δεδομένων στο διαδίκτυο είναι τεράστιος, γεγονός που συχνά δυσκολεύει τον χρήστη στη λήψη απλών καθημερινών αποφάσεων, όπως η επιλογή του γεύματος. Η πρόκληση της δημιουργίας ενός συστήματος που όχι μόνο αποθηκεύει πληροφορίες, αλλά «προτείνει» και «καθοδηγεί» τον χρήστη με βάση τις δικές του ανάγκες και τα διαθέσιμα υλικά του, αποτέλεσε το βασικό κίνητρο για την ενασχόλησή μου με το συγκεκριμένο θέμα.

Κατά τη διάρκεια της ανάπτυξης, είχα την ευκαιρία να εμβαθύνω σε τεχνολογίες αιχμής όπως η **React.js** και το **Flask**, και να αντιμετωπίσω τις προκλήσεις που παρουσιάζει η σύνδεση ενός δυναμικού Frontend με ένα ισχυρό Backend σύστημα συστάσεων. Η διαδικασία αυτή μου προσέφερε πολύτιμη εμπειρία στον σχεδιασμό ολοκληρωμένων πληροφοριακών συστημάτων (Full-stack development) και στην επίλυση σύνθετων προβλημάτων λογισμικού.

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου για την καθοδήγηση, την υποστήριξη και τις πολύτιμες συμβουλές του καθ' όλη τη διάρκεια της εκπόνησης της εργασίας. Επίσης, ευχαριστώ την οικογένεια και τους φίλους μου για την υπομονή και την ενθάρρυνσή τους.

Κλείνοντας, ελπίζω η παρούσα εργασία να αποτελέσει ένα χρήσιμο σημείο αναφοράς για όσους ενδιαφέρονται για την ανάπτυξη σύγχρονων διαδικτυακών εφαρμογών που στοχεύουν στην εξατομίκευση της εμπειρίας του χρήστη.

Περίληψη

Η παρούσα εργασία αφορά τον σχεδιασμό και την υλοποίηση της διαδικτυακής εφαρμογής **«Simply Served»**, μιας καινοτόμου πλατφόρμας διαχείρισης συνταγών και προσωποποιημένων διατροφικών συστάσεων. Κύριος στόχος της εφαρμογής είναι η επίλυση του καθημερινού προβλήματος της επιλογής γεύματος με βάση τα ήδη διαθέσιμα υλικά στην κουζίνα του χρήστη, συνδυάζοντας την ευκολία χρήσης με προηγμένους αλγορίθμους πληροφορικής.

Το τεχνικό υπόβαθρο της εφαρμογής βασίζεται στην αρχιτεκτονική **Full-stack**, χρησιμοποιώντας σύγχρονες τεχνολογίες αιχμής. Το Backend αναπτύχθηκε με το framework **Flask (Python)**, προσφέροντας ένα ισχυρό API για την επεξεργασία δεδομένων και την εκτέλεση υβριδικών συστημάτων συστάσεων (Content-based και Collaborative Filtering). Το Frontend υλοποιήθηκε με τη βιβλιοθήκη **React.js**, διασφαλίζοντας μια δυναμική, ταχύτατη και πλήρως ανταποκρινόμενη (responsive) διεπαφή χρήστη. Για την αποθήκευση και διαχείριση του μεγάλου όγκου δεδομένων, χρησιμοποιήθηκε το σχεσιακό σύστημα **MySQL**.

Ένα από τα σημαντικότερα χαρακτηριστικά του «Simply Served» είναι ο αλγόριθμος **έξυπνης αναζήτησης υλικών**, ο οποίος επιτρέπει στους χρήστες να εισάγουν τα συστατικά που διαθέτουν και να λαμβάνουν αποτελέσματα ταξινομημένα βάσει της συνάφειας και των ελλείψεων. Παράλληλα, η ενσωμάτωση του **Google OAuth 2.0** επιτρέπει την εξατομίκευση της εμπειρίας, προσφέροντας στους εγγεγραμμένους χρήστες ένα προσωπικό Dashboard με προτάσεις βασισμένες στο ιστορικό προβολών και τις βαθμολογίες τους.

Τα αποτελέσματα της εργασίας καταδεικνύουν ότι η χρήση συστημάτων συστάσεων στον τομέα της γαστρονομίας μπορεί να βελτιώσει σημαντικά την καθημερινότητα των χρηστών, μειώνοντας τη σπατάλη τροφίμων και προάγοντας τη μαγειρική δημιουργικότητα. Η εφαρμογή αποτελεί μια σταθερή βάση για μελλοντικές επεκτάσεις, όπως η ενσωμάτωση Τεχνητής Νοημοσύνης για φυσική αλληλεπίδραση και η αυτόματη ανάλυση διατροφικών στοιχείων.

«Web Application for Recipes and Personalized Recommendations»

«Soultana Moschou»

Abstract

This thesis focuses on the design and implementation of the web application **"Simply Served"**, an innovative platform for recipe management and personalized dietary recommendations. The primary objective of the application is to address the everyday challenge of meal selection based on the ingredients already available in the user's kitchen, combining ease of use with advanced computational algorithms.

The technical foundation of the application is based on a **Full-stack architecture**, utilizing cutting-edge technologies. The **Backend** was developed using the **Flask (Python)** framework, providing a robust API for data processing and the execution of hybrid recommendation systems, including **Content-based** and **Collaborative Filtering**. The **Frontend** was implemented with the **React.js** library, ensuring a dynamic, high-speed, and fully responsive user interface. For data storage and management of a large volume of information, the **MySQL** relational database system was employed.

One of the most significant features of "Simply Served" is the **intelligent ingredient search algorithm**, which allows users to input their available ingredients and receive results ranked by relevance and missing components. Furthermore, the integration of **Google OAuth 2.0** enables a personalized experience, providing registered users with a private Dashboard featuring recommendations based on their viewing history and ratings.

The results of this work demonstrate that the application of recommendation systems in the field of gastronomy can significantly improve users' daily lives by reducing food waste and promoting culinary creativity. The application serves as a solid foundation for future extensions, such as the integration of Generative AI for natural interaction and automated nutritional analysis.

.

Περιεχόμενα

Κεφάλαιο 1ο: Εισαγωγή

Η ραγδαία εξέλιξη των τεχνολογιών του Παγκόσμιου Ιστού (Web) και η καθολική ενσωμάτωση των έξυπνων συσκευών στην καθημερινότητα έχουν μεταβάλει ριζικά το μοντέλο αλληλεπίδρασης του ανθρώπου με το περιβάλλον του. Σήμερα, η διαχείριση ακόμη και των πιο θεμελιωδών αναγκών πραγματοποιείται μέσω ψηφιακών διεπαφών που προσφέρουν άμεση πρόσβαση σε πληροφορίες και υπηρεσίες. Η μαγειρική, μια δραστηριότητα άρρηκτα συνδεδεμένη με την ανθρώπινη φύση, δεν θα μπορούσε να μείνει ανεπηρέαστη από αυτή την ψηφιακή μετάβαση.

Η παρούσα εργασία εστιάζει στον σχεδιασμό, την ανάπτυξη και την αξιολόγηση της διαδικτυακής εφαρμογής **"Simply Served"**. Πρόκειται για μια ολοκληρωμένη πλατφόρμα που υπερβαίνει τα όρια ενός παραδοσιακού ψηφιακού βιβλίου συνταγών, ενσωματώνοντας σύγχρονες τεχνικές προσωποποίησης, δυναμικούς αλγορίθμους αναζήτησης και μια αρχιτεκτονική που θέτει στο επίκεντρο την εμπειρία του χρήστη (User Experience - UX).

1.1 Μαγειρική και μαγειρικές συνταγές

Η μαγειρική συνιστά μια από τις πλέον θεμελιώδεις και αρχέγονες ανθρώπινες δραστηριότητες, η οποία διαχρονικά υπερέβη τα στενά όρια της κάλυψης βιολογικών αναγκών. Από την ανακάλυψη της φωτιάς και τη στοιχειώδη θερμική επεξεργασία της τροφής, η μαγειρική μετεξελίχθηκε σε μια πολυδιάστατη μορφή πολιτισμικής έκφρασης, αντικατοπτρίζοντας τις παραδόσεις, τη γεωγραφία και την κοινωνική δομή κάθε λαού. Στη σύγχρονη εποχή, η σημασία της παραμένει καταλυτική, καθώς λειτουργεί ως σημείο τομής μεταξύ της **σωματικής υγείας** (μέσω της θρεπτικής αξίας), της **οικιακής οικονομίας** (μέσω της ορθολογικής διαχείρισης των πόρων) και της **κοινωνικής συνοχής**.

Οι μαγειρικές συνταγές αποτελούν το «λειτουργικό πρωτόκολλο» αυτής της διαδικασίας. Μια συνταγή δεν είναι απλώς μια παράθεση υλικών, αλλά μια δομημένη ακολουθία οδηγιών που διασφαλίζει την επαναληψιμότητα και την επιτυχία ενός γευστικού αποτελέσματος. Ιστορικά, η μεταφορά αυτής της τεχνογνωσίας ακολούθησε τρία κύρια στάδια:

1. **Προφορική Παράδοση:** Η γνώση μεταδιδόταν βιωματικά από γενιά σε γενιά, με αποτέλεσμα την ύπαρξη πολλών παραλλαγών και την έλλειψη ακρίβειας στις μετρήσεις.
2. **Έντυπη Καταγραφή:** Η εμφάνιση των βιβλίων μαγειρικής (cookbooks) τυποποίησε τη γνώση. Για δεκαετίες, εμβληματικές εκδόσεις αποτέλεσαν τον μοναδικό οδηγό για ερασιτέχνες και επαγγελματίες, προσφέροντας κύρος και σταθερότητα, αλλά στερούμενες ευελιξίας.
3. **Ψηφιακή Μεταμόρφωση:** Η έλευση του Παγκόσμιου Ιστού μετέτρεψε τη συνταγή από στατικό κείμενο σε **δυναμικό αντικείμενο πληροφορίας**.

Η ψηφιακή μετάβαση της γαστρονομίας δεν ήταν απλώς μια αλλαγή μέσου (από το χαρτί στην οθόνη), αλλά μια ριζική αλλαγή παραδείγματος. Η ανάγκη του σύγχρονου χρήστη για **άμεση πρόσβαση** σε πληροφορίες οδήγησε στην απαίτηση για:

- **Πολυμεσικότητα:** Η στατική εικόνα αντικαταστάθηκε από βίντεο εκτέλεσης σε πραγματικό χρόνο (step-by-step tutorials), τα οποία μειώνουν το ποσοστό αποτυχίας για τους αρχάριους.
- **Διαδραστικότητα:** Οι χρήστες δεν είναι πλέον παθητικοί αναγνώστες. Μπορούν να βαθμολογήσουν, να σχολιάσουν, να προτείνουν παραλλαγές και να μοιραστούν το αποτέλεσμα στα κοινωνικά δίκτυα.
- **Δυναμική Αναζήτηση και Παραμετροποίηση:** Η δυνατότητα φιλτραρίσματος βάσει θερμίδων, αλλεργιών, χρόνου προετοιμασίας ή διαθέσιμων υλικών κατέστησε τα παραδοσιακά ευρετήρια των βιβλίων παρωχημένα.

Σήμερα, η μαγειρική συνταγή στον ψηφιακό κόσμο λειτουργεί ως μια σύνθετη οντότητα δεδομένων. Η ανάλυση αυτών των δεδομένων επιτρέπει την ανάπτυξη συστημάτων όπως το **Simply Served**, τα οποία μπορούν να «αντιληφθούν» τη λογική σύνδεση μεταξύ διαφορετικών υλικών και να προτείνουν λύσεις που ευθυγραμμίζονται με το προφίλ και τις ανάγκες του εκάστοτε χρήστη, προάγοντας έτσι μια πιο συνειδητή και δημιουργική προσέγγιση στη διατροφή.

1.2 Διαδικτυακές εφαρμογές μαγειρικών συνταγών

Η εξέλιξη των διαδικτυακών εφαρμογών (**web applications**) και των εφαρμογών για κινητές συσκευές (**mobile apps**) δεν αποτέλεσε μια απλή τεχνική αναβάθμιση, αλλά μια πλήρη αναδιαμόρφωση του οικοσυστήματος της οικιακής γαστρονομίας. Στις απαρχές του Παγκόσμιου Ιστού (Web 1.0), οι ιστότοποι συνταγών λειτουργούσαν ως παθητικά ψηφιακά αρχεία κειμένου, προσφέροντας ελάχιστη διαδραστικότητα. Σήμερα, η έλευση του Web 2.0 και 3.0 έχει οδηγήσει στην ανάπτυξη πλατφορμών όπως το AllRecipes, το Tasty και το **TheMealDB** (η κύρια πηγή δεδομένων της παρούσας εργασίας), οι οποίες προσφέρουν μια ολιστική, πολυμεσική εμπειρία.

Αυτή η τεχνολογική μετάβαση χαρακτηρίζεται από τρεις κεντρικούς άξονες:

1. **Αρχιτεκτονική Single Page Applications (SPAs):** Η υιοθέτηση πλαισίων εργασίας όπως η **React.js** επέτρεψε τη δημιουργία εφαρμογών όπου η πλοήγηση και η αλληλεπίδραση γίνονται ακαριαία. Σε αντίθεση με τις παραδοσιακές ιστοσελίδες, οι SPAs φορτώνουν μόνο τα απαραίτητα δεδομένα μέσω **APIs**, εξαλείφοντας τις καθυστερήσεις των συνεχών ανανεώσεων της σελίδας. Το αποτέλεσμα είναι μια εμπειρία χρήσης που προσομοιάζει στις τοπικές εφαρμογές (**native apps**), ενισχύοντας τη διατήρηση του χρήστη (user retention).
2. **Πολυμεσική Σύγκλιση και Κοινωνική Δικτύωση:** Οι σύγχρονες πλατφόρμες ενσωματώνουν βίντεο εκτέλεσης σε υψηλή ανάλυση, διαδραστικούς οδηγούς βήμα-προς-βήμα και συστήματα σχολιασμού σε πραγματικό χρόνο. Η συνταγή παύει να είναι μια στατική οδηγία και μετατρέπεται σε ένα κοινωνικό αντικείμενο που επιδέχεται κριτική, παραλλαγές και άμεσο διαμοιρασμό (social sharing).
3. **Διασυνδεδεμένα Δεδομένα και APIs:** Η χρήση δομημένων πηγών δεδομένων (όπως το TheMealDB API) επιτρέπει στις εφαρμογές να "καταναλώνουν" πληροφορίες με ομοιόμορφο τρόπο, διευκολύνοντας τη διασταύρωση υλικών, διατροφικών στοιχείων και γεωγραφικών προελεύσεων.

Ωστόσο, η ευκολία παραγωγής και δημοσίευσης περιεχομένου οδήγησε στο κρίσιμο φαινόμενο της **πληροφοριακής υπερφόρτωσης (information overload)**. Ο σύγχρονος χρήστης, ευρισκόμενος μπροστά σε έναν ωκεανό χιλιάδων επιλογών, συχνά βιώνει το λεγόμενο "παράδοξο της επιλογής", όπου η πληθώρα των δεδομένων καθιστά τη διαδικασία λήψης απόφασης ("τι θα μαγειρέψω σήμερα;") εξαιρετικά χρονοβόρα και ψυχολογικά κουραστική.

Όπως επισημαίνεται στη βιβλιογραφία, η επιτυχία μιας σύγχρονης διαδικτυακής πλατφόρμας δεν εξαρτάται πλέον από τον όγκο των δεδομένων που διαθέτει, αλλά από την ικανότητά της να λειτουργεί ως **ευφυές φίλτρο**. Η πρόκληση μετατοπίζεται από την απλή αποθήκευση στην προηγμένη επεξεργασία: το σύστημα πρέπει να είναι σε θέση να αναλύει τις προτιμήσεις, το ιστορικό και τους περιορισμούς του χρήστη [1]. Αυτό επιτυγχάνεται μέσω της ενσωμάτωσης εργαλείων **Data Analytics** και **Recommendation Engines**, τα οποία δημιουργούν ένα εξατομικευμένο ψηφιακό περιβάλλον. Σε αυτό το πλαίσιο, η εφαρμογή μετατρέπεται από έναν απλό κατάλογο σε έναν προσωποποιημένο σύμβουλο που μειώνει τον θόρυβο της πληροφορίας και προτείνει στοχευμένες λύσεις.

1.3 Κίνητρο της εργασίας

Το κύριο κίνητρο για την εκπόνηση της παρούσας εργασίας πηγάζει από την παρατήρηση ενός συνηθισμένου καθημερινού προβλήματος: της δυσκολίας διαχείρισης των διαθέσιμων πρώτων υλών

στην οικιακή κουζίνα. Παρά την αφθονία πληροφοριών στο διαδίκτυο, οι περισσότεροι χρήστες συχνά αντιμετωπίζουν το δίλημμα της αξιοποίησης συγκεκριμένων υλικών που έχουν ήδη στο ψυγείο τους, καταλήγοντας είτε σε επαναλαμβανόμενες επιλογές γευμάτων είτε στη σπατάλη τροφίμων.

Η ανάγκη, λοιπόν, για ένα εργαλείο που δεν θα λειτουργεί απλώς ως ένα στατικό ψηφιακό βιβλίο συνταγών, αλλά ως ένας έξυπνος βοηθός μαγειρικής, αποτέλεσε το έναυσμα για τον σχεδιασμό του "Simply Served". Η πρόκληση έγκειται στη δημιουργία μιας γέφυρας μεταξύ των ακατέργαστων δεδομένων μιας βάσης συνταγών και της προσωπικής ανάγκης του χρήστη για γρήγορη και στοχευμένη εύρεση γεύματος.

Επιπλέον, σημαντικό κίνητρο αποτέλεσε η ενασχόληση με σύγχρονες τεχνολογίες ανάπτυξης λογισμικού πλήρους στοίβας (Full-stack Development). Η επιθυμία για την κατανόηση του τρόπου με τον οποίο μια Single Page Application (React) μπορεί να επικοινωνήσει αποτελεσματικά με ένα ισχυρό Backend (Flask) και να διαχειριστεί εξωτερικά API (TheMealDB), προσέφερε το τεχνικό κίνητρο για την υλοποίηση. Η ενσωμάτωση λειτουργιών όπως η αυθεντικοποίηση χρηστών, η διατήρηση προσωπικών λιστών (Wishlist) και το φιλτράρισμα δεδομένων σε πραγματικό χρόνο, συνθέτουν ένα ολοκληρωμένο πλαίσιο εκμάθησης και εφαρμογής των βέλτιστων πρακτικών του σύγχρονου ιστού.

Τέλος, η εργασία υποκινείται από την τάση προς την εξατομίκευση. Σε έναν κόσμο όπου ο χρόνος είναι περιορισμένος, η δυνατότητα ενός χρήστη να "αποθηκεύει" τις προτιμήσεις του και να έχει άμεση πρόσβαση στο Dashboard του, μετατρέπει μια απλή αναζήτηση πληροφοριών σε μια προσωποποιημένη εμπειρία, ενισχύοντας την αλληλεπίδραση χρήστη-μηχανής στον τομέα της γαστρονομίας.

1.4 Συνεισφορά της εργασίας

Η συνεισφορά της παρούσας εργασίας έγκειται στην ανάπτυξη μιας ολοκληρωμένης, πρότυπης διαδικτυακής εφαρμογής, η οποία ενσωματώνει σύγχρονα πρότυπα ανάπτυξης λογισμικού για την επίλυση των προβλημάτων που αναφέρθηκαν προηγουμένως. Η εφαρμογή **Simply Served** δεν αποτελεί απλώς μια διεπαφή προβολής δεδομένων, αλλά ένα συγκροτημένο σύστημα που προσφέρει λύσεις σε τρία επίπεδα: το λειτουργικό, το τεχνικό και το επίπεδο εμπειρίας χρήστη.

Σε **λειτουργικό επίπεδο**, η εργασία συνεισφέρει με τη δημιουργία ενός μηχανισμού αναζήτησης συνταγών βάσει διαθέσιμων υλικών. Ο αλγόριθμος που υλοποιήθηκε στο παρασκήνιο (Backend) επιτρέπει στον χρήστη να εισάγει μια λίστα συστατικών και να λαμβάνει αποτελέσματα ταξινομημένα με βάση τη συνάφεια (match count), αναδεικνύοντας ταυτόχρονα τα υλικά που λείπουν. Αυτή η προσέγγιση βοηθά άμεσα στη μείωση της σπατάλης τροφίμων και στη διευκόλυνση της λήψης απόφασης για το καθημερινό γεύμα.

Σε **τεχνικό επίπεδο**, η συνεισφορά αφορά την επιτυχή υλοποίηση μιας Full-stack αρχιτεκτονικής. Η εργασία αποδεικνύει πώς μπορεί να επιτευχθεί ασφαλής και αποδοτική επικοινωνία μεταξύ ενός **React Frontend** και ενός **Flask REST API**. Συγκεκριμένα:

- Υλοποιήθηκε ένα σύστημα αυθεντικοποίησης μέσω τρίτων παρόχων (Google OAuth2), ενισχύοντας την ευκολία πρόσβασης.
- Αναπτύχθηκε ένας μηχανισμός **Private Routes** και διαχείρισης **JWT (JSON Web Tokens)** για την προστασία των προσωπικών δεδομένων του χρήστη.
- Χρησιμοποιήθηκε το **Context API** της React για τη δυναμική διαχείριση της κατάστασης (Global State) της εφαρμογής, επιτρέποντας την ενημέρωση στοιχείων όπως η Wishlist σε πραγματικό χρόνο χωρίς την ανάγκη ανανέωσης της σελίδας.

Τέλος, η εργασία συνεισφέρει στον τομέα της **εμπειρίας χρήστη (UX)** μέσω ενός καθαρού και λειτουργικού σχεδιασμού. Με τη χρήση της βιβλιοθήκης **Bootstrap** και προσαρμοσμένων CSS animations (Fade-in sections), δημιουργήθηκε ένα περιβάλλον που είναι φιλικό προς τον χρήστη και προσαρμόσιμο (responsive) σε διαφορετικά μεγέθη οθονών. Η ενσωμάτωση του **Dashboard** παρέχει

στον χρήστη μια συγκεντρωτική εικόνα των δραστηριοτήτων του (αγαπημένα, βαθμολογίες, ιστορικό), προάγοντας την εξατομίκευση της πληροφορίας.

1.5 Οργάνωση της εργασίας

Η παρούσα εργασία είναι δομημένη σε επτά κεφάλαια, καθένα από τα οποία εστιάζει σε μια συγκεκριμένη πτυχή της ανάλυσης, του σχεδιασμού και της υλοποίησης της εφαρμογής **Simply Served**. Η σειρά των κεφαλαίων ακολουθεί τη λογική πορεία μιας ολοκληρωμένης ανάπτυξης λογισμικού, ξεκινώντας από το θεωρητικό πλαίσιο και καταλήγοντας στα τελικά συμπεράσματα.

- Στο **Κεφάλαιο 2**, παρουσιάζεται το θεωρητικό υπόβαθρο των Συστημάτων Συστάσεων (Recommendation Systems). Αναλύονται οι βασικές κατηγορίες τους, οι αλγόριθμοι που χρησιμοποιούνται για την εξατομίκευση του περιεχομένου και ο τρόπος με τον οποίο αυτά εφαρμόζονται στον τομέα της μαγειρικής.
- Στο **Κεφάλαιο 3**, γίνεται αναλυτική περιγραφή των γλωσσών προγραμματισμού και των τεχνολογιών που χρησιμοποιήθηκαν. Καλύπτονται οι βιβλιοθήκες του Frontend (React.js), το framework του Backend (Flask), οι μέθοδοι αυθεντικοποίησης και η αρχιτεκτονική των REST APIs.
- Στο **Κεφάλαιο 4**, εξετάζεται ο σχεδιασμός της εφαρμογής. Περιγράφονται οι λειτουργικές απαιτήσεις (User Stories), η αρχιτεκτονική Client-Server, καθώς και η σχεδίαση της βάσης δεδομένων που υποστηρίζει το σύστημα.
- Στο **Κεφάλαιο 5**, αναλύεται η διαδικασία υλοποίησης της εφαρμογής Simply Served. Παρουσιάζεται η δομή των εξαρτημάτων (components) στο Frontend, η δημιουργία των endpoints στο Backend και η λογική του αλγορίθμου αναζήτησης βάσει υλικών.
- Στο **Κεφάλαιο 6**, συνοψίζονται τα συμπεράσματα της εργασίας, καταγράφονται οι προκλήσεις που αντιμετωπίστηκαν κατά την ανάπτυξη και προτείνονται μελλοντικές επεκτάσεις για την περαιτέρω βελτίωση της εφαρμογής.

Κεφάλαιο 2ο: Συστήματα Συστάσεων

Στο παρόν κεφάλαιο εξετάζεται σε βάθος η θεωρία, οι αρχιτεκτονικές και οι μαθηματικοί μηχανισμοί των Συστημάτων Συστάσεων (Recommendation Systems), οι οποίοι πλέον αποτελούν τον «αόρατο κινητήρα» των σύγχρονων διαδικτυακών πλατφορμών. Στην εποχή της πληροφοριακής αφθονίας, όπου ο όγκος των δεδομένων αυξάνεται με γεωμετρική πρόοδο, η ικανότητα ενός συστήματος να προτείνει το κατάλληλο περιεχόμενο στον κατάλληλο χρήστη, την κατάλληλη στιγμή, δεν αποτελεί απλώς μια επιπλέον λειτουργία, αλλά μια θεμελιώδη ανάγκη επιβίωσης του ψηφιακού προϊόντος.

Η ανάλυση ξεκινά από τον ορισμό και τη φιλοσοφία των συστημάτων αυτών, προχωρώντας στην λεπτομερή διερεύνηση των διαφορετικών προσεγγίσεων φιλτραρίσματος της πληροφορίας. Αναλύονται οι αλγόριθμοι που υποστηρίζουν τη λήψη αποφάσεων, από τις κλασικές μεθόδους ομοιότητας έως τις σύγχρονες τεχνικές παραγοντοποίησης πινάκων και μηχανικής μάθησης. Ιδιαίτερη έμφαση δίνεται στον ειδικότερο ρόλο των συστημάτων συστάσεων στο πλαίσιο των εφαρμογών γαστρονομίας, όπου η πολυπλοκότητα των δεδομένων (συνδυασμοί υλικών, διατροφικοί περιορισμοί, υποκειμενικότητα γεύσης) απαιτεί μια πιο εκλεπτυσμένη προσέγγιση σε σχέση με άλλους τομείς όπως το ηλεκτρονικό εμπόριο ή η ψυχαγωγία.

Η μελέτη αυτή αποτελεί το θεωρητικό υπόβαθρο πάνω στο οποίο στηρίχθηκε η ανάπτυξη της εφαρμογής **Simply Served**. Η κατανόηση των πλεονεκτημάτων και των περιορισμών κάθε μεθόδου (όπως το πρόβλημα του Cold Start ή της υπερ-εξειδίκευσης) καθοδήγησε τις σχεδιαστικές αποφάσεις για την υλοποίηση του υβριδικού μοντέλου που παρουσιάζεται στα επόμενα κεφάλαια. Μέσα από αυτή την ανασκόπηση, αναδεικνύεται πώς η μαθηματική μοντελοποίηση της ανθρώπινης προτίμησης μπορεί να μετατρέψει έναν χαοτικό κατάλογο συνταγών σε μια προσωποποιημένη και ευφυή γαστρονομική εμπειρία.

2.1 Εισαγωγή στα συστήματα συστάσεων

Τα συστήματα συστάσεων (recommender systems) αποτελούν μια εξειδικευμένη υποκατηγορία συστημάτων φιλτραρίσματος της πληροφορίας, τα οποία αξιοποιούν προηγμένες αλγοριθμικές μεθόδους και τεχνικές ανάλυσης δεδομένων για να προβλέψουν την προτίμηση ενός χρήστη προς ένα αντικείμενο. Η ανάδυση και η κυριαρχία τους στο ψηφιακό προσκήνιο υπήρξε η αναγκαία απάντηση στην «έκρηξη» του ψηφιακού περιεχομένου και το φαινόμενο του **Information Overload** (πληροφοριακή υπερφόρτωση). Σε ένα περιβάλλον όπου οι διαθέσιμες επιλογές (συνταγές, προϊόντα, πληροφορίες) ξεπερνούν κατά πολύ την ανθρώπινη ικανότητα επεξεργασίας, τα συστήματα αυτά λειτουργούν ως «ευφυείς διαμεσολαβητές» που γεφυρώνουν το χάσμα μεταξύ της τεράστιας προσφοράς και της συγκεκριμένης ζήτησης.

Ο Μηχανισμός της "Χρησιμότητας"

Στον πυρήνα κάθε συστήματος συστάσεων βρίσκεται η μαθηματική έννοια της **Συνάρτησης Χρησιμότητας (Utility Function)**. Το σύστημα επιχειρεί να υπολογίσει μια τιμή $u(c, s)$, η οποία αντιπροσωπεύει το πόσο "χρήσιμη" ή αρεστή θα είναι η συνταγή s για τον χρήστη c. Δεδομένου ότι ο χώρος των πιθανών συνδυασμών χρηστών-συνταγών είναι κατά κανόνα "αραιός" (sparse) — δηλαδή οι χρήστες έχουν αξιολογήσει μόνο ένα ελάχιστο κλάσμα των διαθέσιμων επιλογών— ο αλγόριθμος καλείται να προβλέψει τις τιμές χρησιμότητας για όλα τα αντικείμενα που ο χρήστης δεν έχει ακόμη δει.

Η Διαδικασία Συλλογής Δεδομένων

Η αποτελεσματικότητα ενός συστήματος συστάσεων εξαρτάται άμεσα από την ποιότητα και το είδος της ανατροφοδότησης (feedback) που συλλέγεται:

1. **Ρητή Ανατροφοδότηση (Explicit Feedback):** Περιλαμβάνει τις άμεσες ενέργειες του χρήστη για την αξιολόγηση ενός αντικειμένου, όπως οι βαθμολογίες με αστέρια (ratings) ή η συγγραφή κριτικών. Παρόλο που είναι η πιο ακριβής πηγή πληροφορίας, είναι δύσκολο να συλλεχθεί σε μεγάλες ποσότητες, καθώς απαιτεί την ενεργό συμμετοχή του χρήστη.
2. **Έμμεση Ανατροφοδότηση (Implicit Feedback):** Προκύπτει από την παρακολούθηση της συμπεριφοράς του χρήστη χωρίς τη δική του παρέμβαση. Παραδείγματα αποτελούν το ιστορικό αναζητήσεων, ο χρόνος παραμονής στην καρτέλα μιας συνταγής, ή η προσθήκη ενός πιάτου στη Wishlist. Στο **Simply Served**, η έμμεση ανατροφοδότηση είναι καθοριστική, καθώς "προδίδει" τις γαστρονομικές προτιμήσεις του χρήστη ακόμα και αν ο ίδιος δεν έχει προβεί σε βαθμολόγηση.

Γνωστικό Φορτίο και Engagement

Από ψυχολογική άποψη, τα συστήματα συστάσεων στοχεύουν στη μείωση του **Γνωστικού Φορτίου (Cognitive Load)**. Η διαδικασία λήψης απόφασης είναι μια ενεργοβόρα διαδικασία για τον ανθρώπινο εγκέφαλο. Παρέχοντας στοχευμένες προτάσεις, η εφαρμογή απαλλάσσει τον χρήστη από το άγχος της ατέρμονης αναζήτησης, ενισχύοντας το λεγόμενο **User Engagement**. Ένας χρήστης που βρίσκει γρήγορα μια συνταγή που του αρέσει, είναι πολύ πιο πιθανό να επιστρέψει στην πλατφόρμα, δημιουργώντας έναν κύκλο αλληλεπίδρασης και εμπιστοσύνης.

Οι Στόχοι της Εφαρμογής Simply Served

Στο πλαίσιο του **Simply Served**, η εισαγωγή στα συστήματα συστάσεων δεν αφορά μόνο την παράθεση δεδομένων, αλλά τη δυναμική διαχείριση της σχέσης μεταξύ "Χρήστη" και "Συνταγής" μέσω τριών κεντρικών πυλώνων:

- **Εξατομίκευση (Personalization):** Η μετάβαση από το γενικό στο ειδικό. Το σύστημα αναγνωρίζει τις ιδιαιτερότητες κάθε προφίλ, διασφαλίζοντας ότι το Dashboard δεν είναι μια στατική σελίδα, αλλά ένας ζωντανός καθρέπτης των προτιμήσεων του χρήστη.
- **Ανακάλυψη (Discovery & Serendipity):** Ένα επιτυχημένο σύστημα δεν πρέπει να προτείνει μόνο τα προφανή. Η έννοια του **Serendipity** (η ικανότητα να ανακαλύπτει κανείς ευχάριστες εκπλήξεις) είναι κρίσιμη στη μαγειρική. Το σύστημα ωθεί τον χρήστη να δοκιμάσει νέες γεύσεις ή υλικά που βρίσκονται εκτός των συνηθισμένων του επιλογών, αλλά ταιριάζουν στο προφίλ του.
- **Ακρίβεια και Εμπιστοσύνη:** Η πιστότητα των προβλέψεων είναι το κλειδί. Μια λανθασμένη σύσταση (π.χ. μια συνταγή με κρέας σε έναν vegan χρήστη) μπορεί να κλονίσει την αξιοπιστία της εφαρμογής. Συνεπώς, η ακρίβεια των αλγορίθμων ομοιότητας είναι το θεμέλιο της μακροχρόνιας χρήσης.

Η κατανόηση αυτών των πολυεπίπεδων αρχών καθιστά σαφές ότι τα συστήματα συστάσεων είναι κάτι πολύ περισσότερο από απλά εργαλεία αναζήτησης· είναι το "νευρικό σύστημα" της σύγχρονης ψηφιακής εμπειρίας, το οποίο μετατρέπει τα ακατέργαστα δεδομένα σε εξατομικευμένη γνώση.

2.2 Κατηγορίες συστημάτων συστάσεων

Τα συστήματα συστάσεων αποτελούν ένα ετερογενές πεδίο της πληροφορικής, όπου δεν υφίσταται μια καθολική "ιδανική" λύση. Αντιθέτως, η αρχιτεκτονική τους διαφοροποιείται ριζικά ανάλογα με τη φύση των διαθέσιμων δεδομένων, την υπολογιστική ισχύ του Backend και τους επιχειρηματικούς στόχους της πλατφόρμας. Η επιλογή της κατάλληλης κατηγορίας είναι μια στρατηγική απόφαση που καθορίζει την **πιστότητα (fidelity)** των προβλέψεων και την ικανότητα του συστήματος να κλιμακώνεται καθώς αυξάνεται ο αριθμός των χρηστών και των αντικειμένων.

Η επιστημονική κοινότητα και η βιομηχανία λογισμικού έχουν καταλήξει σε τρεις κύριες αρχιτεκτονικές προσεγγίσεις, καθεμία από τις οποίες αντιμετωπίζει το πρόβλημα του φιλτραρίσματος της πληροφορίας από διαφορετική οπτική γωνία:

1. Η Φιλοσοφία της "Ομοιότητας Αντικειμένων"

Στην προσέγγιση αυτή, το κέντρο βάρους πέφτει στα **μεταδεδομένα (metadata)** των ίδιων των αντικειμένων. Το σύστημα επιχειρεί να κατανοήσει τη φύση του περιεχομένου (π.χ. τι υλικά περιέχει μια συνταγή, ποια είναι η διατροφική της αξία) και να εντοπίσει συσχετίσεις μεταξύ τους. Είναι μια προσέγγιση εσωστρεφής, η οποία βασίζεται στην παραδοχή ότι αν ένας χρήστης έδειξε ενδιαφέρον για ένα αντικείμενο με συγκεκριμένα χαρακτηριστικά, θα δείξει ενδιαφέρον και για άλλα αντικείμενα που μοιράζονται την ίδια "ψηφιακή υπογραφή".

2. Η Φιλοσοφία της "Κοινωνικής Νοημοσύνης"

Εδώ, το σύστημα αγνοεί τα χαρακτηριστικά των αντικειμένων και επικεντρώνεται αποκλειστικά στη συμπεριφορά της κοινότητας. Η προσέγγιση αυτή βασίζεται στη συλλογική γνώση (**Wisdom of the Crowd**). Το σύστημα αναζητά πρότυπα (patterns) αλληλεπίδρασης, θεωρώντας ότι οι χρήστες που συμφώνησαν σε προηγούμενες επιλογές τους, λειτουργούν ως "κοινωνικοί καθρέπτες" ο ένας του άλλου. Είναι μια μέθοδος που προάγει την ανακάλυψη νέου περιεχομένου, αλλά απαιτεί κρίσιμο όγκο δεδομένων για να καταστεί αξιόπιστη.

3. Η Ολιστική - Υβριδική Προσέγγιση

Αναγνωρίζοντας ότι κάθε μία από τις παραπάνω μεθόδους έχει "τυφλά σημεία", η σύγχρονη τάση επιτάσσει τον συνδυασμό τους. Τα υβριδικά συστήματα αποτελούν την πιο εξελιγμένη μορφή, καθώς επιχειρούν να εξισορροπήσουν την ακρίβεια της ανάλυσης περιεχομένου με τη διορατικότητα της κοινωνικής ανάλυσης.

Κρίσιμες Προκλήσεις και Σχεδιαστικές Παράμετροι

Κατά την ανάλυση των κατηγοριών, είναι απαραίτητο να ληφθούν υπόψη δύο βασικά προβλήματα που επηρεάζουν την επιλογή αρχιτεκτονικής:

- **Το Πρόβλημα του Cold Start (Ψυχρή Εκκίνηση):** Πρόκειται για την αδυναμία του συστήματος να παρέχει συστάσεις για έναν νέο χρήστη (που δεν έχει ιστορικό) ή για μια νέα συνταγή (που δεν έχει βαθμολογηθεί). Ενώ οι κοινωνικές μέθοδοι αποτυγχάνουν σε αυτή την περίπτωση, οι μέθοδοι βάσει περιεχομένου μπορούν να δώσουν λύση αναλύοντας απλώς τα υλικά της νέας συνταγής.
- **Η Κλιμακωσιμότητα (Scalability):** Καθώς η βάση δεδομένων μεγαλώνει, ο υπολογισμός των ομοιοτήτων μεταξύ εκατομμυρίων χρηστών μπορεί να καταστεί απαγορευτικός σε χρόνο εκτέλεσης. Η επιλεγμένη κατηγορία πρέπει να επιτρέπει την αποτελεσματική επεξεργασία των δεδομένων στο Backend (π.χ. μέσω του Flask API) χωρίς να υποβαθμίζει την εμπειρία του χρήστη στο Frontend.

Στην εφαρμογή **Simply Served**, η κατανόηση αυτών των κατηγοριών αποτέλεσε τον οδικό χάρτη για την υλοποίηση. Η ανάγκη του χρήστη να αναζητά συνταγές με βάση τα υλικά του (Content-Based) έπρεπε να συνυπάρξει με την ανάγκη για προσωποποιημένες προτάσεις στο Dashboard (Collaborative & Hybrid), δημιουργώντας ένα πολυεπίπεδο σύστημα που ανταποκρίνεται σε διαφορετικά σενάρια χρήσης.

Ακολουθεί η αναλυτική και σε βάθος παρουσίαση των τριών βασικών κατηγοριών:

1. **Content-Based Filtering** (Φιλτράρισμα βάσει περιεχομένου)

2. **Collaborative Filtering** (Συνεργατικό φιλτράρισμα)
3. **Hybrid Systems** (Υβριδικά συστήματα)

2.2.1 Content-Based Filtering

Το φιλτράρισμα βάσει περιεχομένου (Content-Based Filtering) εδράζεται στη θεμελιώδη παραδοχή της **συνέπειας των προτιμήσεων**: η υπόθεση, δηλαδή, ότι οι μελλοντικές επιλογές ενός χρήστη θα τείνουν να προσομοιάζουν με τα χαρακτηριστικά των αντικειμένων που κατανάλωσε ή αξιολόγησε θετικά στο παρελθόν. Σε αυτή την προσέγγιση, ο αλγόριθμος δεν εστιάζει στη συμπεριφορά της μάζας, αλλά στην ανάλυση των εγγενών χαρακτηριστικών (**metadata**) των αντικειμένων.

Μοντελοποίηση Προφίλ και Εξαγωγή Χαρακτηριστικών

Η λειτουργία του συστήματος βασίζεται στη δημιουργία δύο διανυσματικών αναπαραστάσεων:

1. **Προφίλ Αντικειμένου (Item Profile):** Κάθε συνταγή μετατρέπεται σε ένα δομημένο σύνολο χαρακτηριστικών. Στο **Simply Served**, αυτό περιλαμβάνει μια πληθώρα δεδομένων: τη λίστα συστατικών (ingredients), τη γαστρονομική κατηγορία (π.χ. Vegan, Seafood), την εθνική κουζίνα (area) και λέξεις-κλειδιά που εξάγονται από τις οδηγίες. Για την ποσοτικοποίηση αυτών των δεδομένων συχνά χρησιμοποιούνται τεχνικές όπως η **TF-IDF (Term Frequency-Inverse Document Frequency)**, η οποία αποδίδει μεγαλύτερη βαρύτητα σε σπάνια και χαρακτηριστικά υλικά (π.χ. "Κρόκος Κοζάνης") έναντι κοινότοπων υλικών (π.χ. "Αλάτι").
2. **Προφίλ Χρήστη (User Profile):** Το προφίλ αυτό αποτελεί μια δυναμική καταγραφή των ενδιαφερόντων του χρήστη. Κάθε φορά που ένας χρήστης αλληλεπιδρά με μια συνταγή —είτε μέσω "Like", είτε προσθέτοντάς την στη Wishlist— το σύστημα ενημερώνει το διάνυσμα του χρήστη. Αν ο χρήστης επιλέγει συχνά συνταγές με την ετικέτα "Italian" και υλικά όπως "Parmesan", το προφίλ του αποκτά υψηλή "βάρος" στις συγκεκριμένες διαστάσεις του χώρου χαρακτηριστικών.

Μαθηματική Συνάφεια και Συστάσεις

Η παραγωγή της σύστασης επιτυγχάνεται μέσω του υπολογισμού της απόστασης ή της γωνίας μεταξύ του διανύσματος του χρήστη και των διανυσμάτων όλων των διαθέσιμων συνταγών. Η **συνημίτονο ομοιότητα (cosine similarity)** είναι η πλέον διαδεδομένη μέθοδος, καθώς μετρά την κατεύθυνση των προτιμήσεων ανεξάρτητα από τον όγκο των δεδομένων.

Πλεονεκτήματα και Περιορισμοί

Το Content-Based Filtering προσφέρει σημαντικά οφέλη, αλλά συνοδεύεται και από συγκεκριμένες προκλήσεις:

- **Αυτονομία Χρήστη:** Το σύστημα μπορεί να λειτουργήσει άμεσα για έναν χρήστη, χωρίς να απαιτείται πλήθος δεδομένων από την υπόλοιπη κοινότητα. Αυτό επιλύει το πρόβλημα του "Cold Start" για νέα αντικείμενα (new items).
- **Διαφάνεια (Transparency):** Οι συστάσεις είναι εύκολα ερμηνεύσιμες. Το σύστημα μπορεί να παρέχει αιτιολόγηση, όπως: *"Σας προτείνουμε αυτή τη συνταγή επειδή περιέχει Ginger, το οποίο υπάρχει σε άλλες 3 συνταγές που βαθμολογήσατε θετικά"*.
- **Το Πρόβλημα της Υπερ-εξειδίκευσης (Overspecialization):** Ο κύριος περιορισμός είναι η έλλειψη "Serendipity" (ευχάριστης έκπληξης). Ο χρήστης εγκλωβίζεται σε μια "φούσκα" περιεχομένου, λαμβάνοντας προτάσεις πανομοιότυπες με τις προηγούμενες, χωρίς να εκτίθεται σε νέες γαστρονομικές εμπειρίες.

Όπως τεκμηριώνεται στη βιβλιογραφία, η ποιότητα των συστάσεων σε αυτά τα συστήματα εξαρτάται άμεσα από την ακρίβεια και το βάθος της περιγραφής των χαρακτηριστικών των αντικειμένων [3]. Στο **Simply Served**, η λογική αυτή εφαρμόζεται οργανικά μέσω του Ingredients.jsx. Ο χρήστης, επιλέγοντας υλικά, ορίζει ουσιαστικά ένα προσωρινό "προφίλ αναζήτησης" και το σύστημα εκτελεί μια content-based ανάκτηση, επιστρέφοντας συνταγές που παρουσιάζουν τη μέγιστη επικάλυψη χαρακτηριστικών με τις επιλογές του.

2.2.2 Collaborative Filtering

Το συνεργατικό φιλτράρισμα (Collaborative Filtering) αντιπροσωπεύει μια θεμελιώδη αλλαγή παραδείγματος στα συστήματα συστάσεων, καθώς μετατοπίζει την εστίαση από την ανάλυση των ιδιοτήτων των αντικειμένων στην ανάλυση της **συλλογικής ανθρώπινης συμπεριφοράς**. Σε αντίθεση με το φιλτράρισμα βάσει περιεχομένου, η μέθοδος αυτή λειτουργεί ως ένας μηχανισμός «κοινωνικής ευφυΐας», ο οποίος δεν απαιτεί καμία πρότερη γνώση για τα χαρακτηριστικά των αντικειμένων (όπως τα υλικά μιας συνταγής). Αντίθετα, βασίζεται αποκλειστικά στον εντοπισμό μοτίβων (patterns) μέσα από τις αλληλεπιδράσεις των χρηστών, όπως βαθμολογίες, προβολές και προσθήκες στα αγαπημένα.

Οι Δύο Πυλώνες της Συνεργατικής Προσέγγισης

Η αρχιτεκτονική του συνεργατικού φιλτραρίσματος υλοποιείται μέσω δύο κύριων μεθοδολογιών:

1. **User-based Collaborative Filtering (Φιλτράρισμα βάσει Χρηστών):** Η προσέγγιση αυτή βασίζεται στην εύρεση ομοιοτήτων μεταξύ των προφίλ των χρηστών. Αν ο Χρήστης Α και ο Χρήστης Β έχουν βαθμολογήσει παρόμοια ένα σύνολο συνταγών, το σύστημα θεωρεί ότι έχουν «συγγενικά» γούστα. Έτσι, αν ο Χρήστης Β βαθμολογήσει θετικά μια νέα συνταγή που ο Χρήστης Α δεν έχει δει ακόμα, το σύστημα θα την προτείνει στον Χρήστη Α. Μαθηματικά, αυτό επιτυγχάνεται με τον υπολογισμό της ομοιότητας μεταξύ των γραμμών ενός πίνακα Χρηστών-Αντικειμένων.
2. **Item-based Collaborative Filtering (Φιλτράρισμα βάσει Αντικειμένων):** Εδώ η ομοιότητα υπολογίζεται μεταξύ των ίδιων των αντικειμένων, αλλά με βάση τις βαθμολογίες που έχουν λάβει από το σύνολο των χρηστών. Αν μια μεγάλη ομάδα χρηστών που προτίμησε μια "Carbonara" επέδειξε παρόμοια προτίμηση και για ένα "Pastitsio", αυτά τα δύο πιάτα θεωρούνται «συνεργατικά όμοια». Αυτή η μέθοδος θεωρείται συχνά πιο σταθερή και κλιμακώσιμη, καθώς οι σχέσεις μεταξύ αντικειμένων αλλάζουν λιγότερο συχνά από ό,τι οι προτιμήσεις των χρηστών.

Ο Πίνακας Χρήστη-Αντικειμένου (User-Item Matrix)

Στην εφαρμογή **Simply Served**, η υποδομή για την υποστήριξη αυτών των αλγορίθμων υλοποιείται μέσω του component RatingStars.jsx, το οποίο τροφοδοτεί τη βάση δεδομένων με ρητή ανατροφοδότηση (explicit feedback). Τα δεδομένα αυτά οργανώνονται σε έναν **Πίνακα Χρήστη-Αντικειμένου**, όπου οι σειρές αντιπροσωπεύουν τους χρήστες και οι στήλες τις συνταγές.

Ωστόσο, η εφαρμογή αυτής της μεθόδου συνοδεύεται από σημαντικές τεχνικές προκλήσεις:

- **Το Πρόβλημα του "Cold Start":** Ένας νέος χρήστης χωρίς ιστορικό ή μια νέα συνταγή χωρίς βαθμολογίες παραμένουν «αόρατοι» για τον αλγόριθμο, καθώς δεν υπάρχουν δεδομένα αλληλεπίδρασης για να υπολογιστεί η ομοιότητα.
- **Αραιότητα Δεδομένων (Sparsity):** Στις περισσότερες πραγματικές εφαρμογές, οι χρήστες βαθμολογούν ένα ελάχιστο ποσοστό των διαθέσιμων συνταγών, με αποτέλεσμα ο πίνακας να είναι κατά 99% κενός, γεγονός που δυσχεραίνει τον εντοπισμό «γειτόνων».

Serendipity και η "Σοφία του Πλήθους"

Σύμφωνα με τη βιβλιογραφία, το συνεργατικό φιλτράρισμα υπερέχει στην ικανότητά του να ανακαλύπτει «τυχαίες» αλλά επιτυχημένες προτάσεις (**serendipity**). Ενώ το Content-based σύστημα θα πρότεινε πάντα παρόμοια πιάτα (π.χ. πάντα ζυμαρικά), το Collaborative σύστημα μπορεί να οδηγήσει τον χρήστη σε εντελώς νέες κατηγορίες (π.χ. από την ιταλική στην ασιατική κουζίνα) επειδή εντόπισε ότι άλλοι χρήστες με παρόμοια γούστα έκαναν αυτό το άλμα. Αυτή η αξιοποίηση της «κοινής σοφίας» (**wisdom of the crowd**) προσδίδει στην εφαρμογή έναν χαρακτήρα ανακάλυψης, καθιστώντας την εμπειρία του χρήστη πιο πλούσια και λιγότερο προβλέψιμη [4].

2.2.3 Hybrid συστήματα

Τα υβριδικά συστήματα συστάσεων (Hybrid Recommender Systems) αποτελούν την τεχνολογική απάντηση στην ανάγκη για πιο στιβαρά και αξιόπιστα συστήματα φιλτραρίσματος. Η βασική τους φιλοσοφία εδράζεται στην παραδοχή ότι καμία μεμονωμένη μέθοδος δεν μπορεί να προσφέρει βέλτιστα αποτελέσματα σε όλες τις συνθήκες. Συνδυάζοντας πολλαπλές αλγοριθμικές προσεγγίσεις, τα υβριδικά συστήματα καταφέρνουν να εξουδετερώσουν τα μειονεκτήματα της μίας μεθόδου χρησιμοποιώντας τα πλεονεκτήματα της άλλης, δημιουργώντας μια συνέργεια που ενισχύει την τελική ακρίβεια των προβλέψεων.

Στρατηγικές Υβριδοποίησης

Η επιστημονική έρευνα έχει ορίσει διάφορους τρόπους με τους οποίους μπορεί να επιτευχθεί αυτή η σύζευξη, με τους σημαντικότερους να περιλαμβάνουν:

1. **Weighted Hybrid (Σταθμισμένη Υβριδοποίηση):** Το σύστημα εκτελεί ταυτόχρονα έναν αλγόριθμο Content-Based και έναν Collaborative. Το τελικό σκορ για κάθε συνταγή προκύπτει από έναν γραμμικό συνδυασμό των δύο επιμέρους σκορ. Για παράδειγμα, αν ένας χρήστης είναι καινούργιος, το σύστημα δίνει μεγαλύτερη βαρύτητα στο περιεχόμενο (υλικά), ενώ για παλιούς χρήστες με πολλές βαθμολογίες, η βαρύτητα μετατοπίζεται στο συνεργατικό φιλτράρισμα.
2. **Switching Hybrid (Υβριδοποίηση Εναλλαγής):** Το σύστημα επιλέγει τη βέλτιστη μέθοδο ανάλογα με το πλαίσιο (context). Εάν το επίπεδο εμπιστοσύνης στον Collaborative αλγόριθμο είναι χαμηλό (π.χ. λόγω έλλειψης δεδομένων), το σύστημα «μεταγωγής» επιλέγει αυτόματα την Content-Based προσέγγιση.
3. **Cascade Hybrid (Υβριδοποίηση Διαδοχής):** Μία μέθοδος χρησιμοποιείται για να παράγει μια αρχική λίστα υποψηφίων συνταγών και η δεύτερη μέθοδος χρησιμοποιείται για να κάνει ένα «λεπτό» φιλτράρισμα ή μια τελική κατάταξη (re-ranking) των αποτελεσμάτων.
4. **Feature Augmentation (Επαύξηση Χαρακτηριστικών):** Η έξοδος ενός αλγορίθμου χρησιμοποιείται ως πρόσθετο χαρακτηριστικό εισόδου για έναν άλλον. Για παράδειγμα, η βαθμολογία ομοιότητας μεταξύ χρηστών μπορεί να ενσωματωθεί ως «μεταδεδομένο» στον Content-Based αλγόριθμο.

Εφαρμογή στο Simply Served

Στην περίπτωση της εφαρμογής **Simply Served**, η υβριδοποίηση αποτελεί το κλειδί για μια ολοκληρωμένη εμπειρία. Το σύστημα επιλύει το κρίσιμο πρόβλημα του **Cold Start** για νέα αντικείμενα: όταν μια νέα συνταγή εισάγεται από το *TheMealDB* και δεν έχει ακόμα βαθμολογίες, το Collaborative Filtering θα την αγνοούσε. Ωστόσο, το υβριδικό μοντέλο αναγνωρίζει τα υλικά της (π.χ. "Chicken", "Curry") μέσω του Ingredients.jsx και την προτείνει σε χρήστες που έχουν δείξει προτίμηση σε παρόμοια συστατικά.

Πλεονεκτήματα και Βιβλιογραφική Τεκμηρίωση

Η υιοθέτηση υβριδικών μοντέλων προσφέρει τρία κύρια οφέλη:

- **Ανθεκτικότητα (Robustness):** Είναι λιγότερο ευαίσθητα σε «θόρυβο» (λανθασμένες βαθμολογίες) ή σε έλλειψη δεδομένων (data sparsity).
- **Ποικιλομορφία (Diversity):** Αποφεύγουν την «παγίδα» της υπερ-εξειδίκευσης (overspecialization) προσφέροντας προτάσεις που είναι ταυτόχρονα συναφείς με τα υλικά του χρήστη αλλά και απρόσμενες βάσει των τάσεων της κοινότητας.
- **Εξατομίκευση σε Πραγματικό Χρόνο:** Επιτρέπουν στο Dashboard να ενημερώνεται δυναμικά, συνδυάζοντας το άμεσο ενδιαφέρον του χρήστη (πρόσφατα υλικά) με τις μακροπρόθεσμες προτιμήσεις του (ιστορικό βαθμολογιών).

Σύμφωνα με τη βιβλιογραφία, τα υβριδικά συστήματα αποδεικνύονται πιο ακριβή και ανθεκτικά σε θόρυβο δεδομένων, προσφέροντας μια πιο ισορροπημένη εμπειρία χρήστη. Η υβριδοποίηση επιτρέπει στο σύστημα να διατηρεί την εξατομίκευση (personalization) σε υψηλά επίπεδα, αποφεύγοντας ταυτόχρονα την παγίδα της υπερ-εξειδίκευσης, καθώς μπορεί να εισάγει νέα στοιχεία στις συστάσεις μέσω της κοινωνικής πληροφορίας [5].

2.3 Αλγόριθμοι συστάσεων

Η λειτουργία κάθε συστήματος συστάσεων βασίζεται σε συγκεκριμένους αλγορίθμους που επεξεργάζονται τη σύνθετη σχέση μεταξύ χρηστών και αντικειμένων. Η κεντρική πρόκληση αυτών των αλγορίθμων είναι διττή: η ακριβής **μέτρηση της ομοιότητας (similarity)** μεταξύ των οντοτήτων και η έγκυρη **πρόβλεψη της προτίμησης (rating prediction)** για αντικείμενα που ο χρήστης δεν έχει ακόμη αλληλεπιδράσει. Για να καταστεί εφικτή αυτή η υπολογιστική διαδικασία, τα δεδομένα μετασχηματίζονται από περιγραφικές πληροφορίες σε μαθηματικές οντότητες.

Διανυσματική Αναπαράσταση και Χώρος Χαρακτηριστικών

Κάθε συνταγή ή χρήστης αναπαρίσταται ως ένα διάνυσμα (vector) σε έναν πολυδιάστατο χώρο χαρακτηριστικών. Στην περίπτωση του **Simply Served**, κάθε διάσταση του χώρου αντιστοιχεί σε ένα συγκεκριμένο χαρακτηριστικό, όπως ένα συστατικό (π.χ. "Cinnamon"), μια κατηγορία (π.χ. "Dessert") ή μια ετικέτα κουζίνας (π.χ. "Greek"). Η θέση ενός διανύσματος στον χώρο αυτό προσδιορίζει την "ταυτότητα" της συνταγής, επιτρέποντας στον αλγόριθμο να συγκρίνει γεωμετρικά τη συνάφεια μεταξύ διαφορετικών εγγραφών.

Μέτρα Ομοιότητας

Ένας από τους πιο διαδεδομένους αλγορίθμους για τη μέτρηση της ομοιότητας είναι η **Συνημίτονο Ομοιότητα (Cosine Similarity)**. Ο αλγόριθμος αυτός υπολογίζει το συνημίτονο της γωνίας μεταξύ δύο διανυσμάτων. Στην εφαρμογή μας, αυτό μεταφράζεται ως η σύγκριση δύο συνταγών με βάση τα κοινά τους υλικά: όσο περισσότερα κοινά συστατικά έχουν δύο πιάτα, τόσο μικρότερη είναι η γωνία των διανυσμάτων τους και τόσο μεγαλύτερη η ομοιότητά τους. Το πλεονέκτημα της μεθόδου αυτής είναι ότι εστιάζει στον προσανατολισμό των διανυσμάτων και όχι στο μέγεθός τους, καθιστώντας την ιδανική για δεδομένα όπου ο αριθμός των υλικών μπορεί να διαφέρει σημαντικά μεταξύ των συνταγών.

Άλλες μέθοδοι περιλαμβάνουν τον **Συντελεστή Συσχέτιση Pearson**, ο οποίος χρησιμοποιείται συχνά στο συνεργατικό φιλτράρισμα. Ο Pearson υπερέχει καθώς αφαιρεί τη μέση βαθμολογία κάθε χρήστη, εξουδετερώνοντας έτσι τις διαφορετικές κλίμακες βαθμολόγησης (π.χ. κάποιοι χρήστες είναι "αυστηροί" ενώ άλλοι βαθμολογούν πάντα υψηλά), εστιάζοντας στην τάση της προτίμησης και όχι στην απόλυτη τιμή.

Προηγμένες Τεχνικές και Παραγοντοποίηση

Οι σύγχρονοι αλγόριθμοι κάνουν χρήση τεχνικών **Παραγοντοποίησης Πινάκων (Matrix Factorization)**. Η τεχνική αυτή είναι απαραίτητη για τη διαχείριση του "αραιού" (sparse) πίνακα χρηστών-συνταγών, όπου οι περισσότεροι χρήστες έχουν βαθμολογήσει ελάχιστα πιάτα. Αναλύοντας τον αρχικό πίνακα σε μικρότερους πίνακες με **λανθάνοντα χαρακτηριστικά (latent factors)**, το σύστημα ανακαλύπτει κρυφές συνδέσεις που δεν είναι προφανείς από τα απλά μεταδεδομένα.

Για παράδειγμα, μπορεί να εντοπιστεί ότι χρήστες που προτιμούν πικάντικα μεξικάνικα πιάτα τείνουν να βαθμολογούν υψηλά και ορισμένες καυτερές συνταγές της ασιατικής κουζίνας. Ο αλγόριθμος αντιλαμβάνεται τον κοινό "λανθάνοντα παράγοντα" (την ένταση των μπαχαρικών), ακόμα και αν τα υλικά (π.χ. chili vs wasabi) διαφέρουν πλήρως.

Υπολογιστική Αποδοτικότητα

Όπως επισημαίνεται στη βιβλιογραφία, η επιλογή του αλγορίθμου καθορίζει την υπολογιστική πολυπλοκότητα και την ικανότητα απόκρισης του συστήματος σε πραγματικό χρόνο. Ειδικά σε διαδικτυακές εφαρμογές όπως το **Simply Served**, η ισορροπία μεταξύ ακρίβειας και ταχύτητας είναι ζωτικής σημασίας. Ένας υπερβολικά πολύπλοκος αλγόριθμος μπορεί να προσφέρει εξαιρετική ακρίβεια, αλλά αν απαιτεί δευτερόλεπτα για να παράγει αποτελέσματα, υποβαθμίζει τη ροή της πλοήγησης και την εμπειρία του χρήστη [6]. Συνεπώς, η βελτιστοποίηση των αλγορίθμων στο Backend (Flask) και η αποδοτική αποστολή τους στο Frontend (React) αποτελεί τον πυρήνα της τεχνικής μας υλοποίησης.

2.3.1 Μέτρα Ομοιότητας: Cosine Similarity και Pearson Correlation

Η μέτρηση της ομοιότητας αποτελεί τον δομικό λίθο των συστημάτων συστάσεων, καθώς επιτρέπει τη μαθηματική σύγκριση μεταξύ χρηστών ή αντικειμένων. Η επιλογή του κατάλληλου μέτρου εξαρτάται από τη φύση των δεδομένων (π.χ. συστατικά συνταγών έναντι βαθμολογιών).

Η Συνημίτονο Ομοιότητα (Cosine Similarity)

Στον αλγόριθμο της Συνημίτονο Ομοιότητας, κάθε συνταγή αντιμετωπίζεται ως ένα διάνυσμα σε έναν χώρο n διαστάσεων, όπου n είναι το πλήθος των διαθέσιμων συστατικών. Αν μια συνταγή περιέχει ένα υλικό, η αντίστοιχη διάσταση παίρνει την τιμή 1, αλλιώς 0 (binary vectorization).

Η ομοιότητα υπολογίζεται από το συνημίτονο της γωνίας θ μεταξύ των διανυσμάτων A και B:

$$similarity = \cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Στην εφαρμογή **Simply Served**, η μέθοδος αυτή είναι ιδανική για το **Content-Based Filtering**. Για παράδειγμα, μια συνταγή για "Μακαρόνια με Ντομάτα" και μια για "Πίτσα Μαργαρίτα" θα έχουν διανύσματα που δείχνουν προς παρόμοια κατεύθυνση λόγω κοινών υλικών (ντομάτα, βασιλικός), παράγοντας υψηλό σκορ ομοιότητας. Το βασικό πλεονέκτημα είναι ότι το μέγεθος των διανυσμάτων (π.χ. το πλήθος των υλικών) δεν επηρεάζει το αποτέλεσμα, παρά μόνο η "κατεύθυνσή" τους στον χώρο των χαρακτηριστικών.

Ο Συντελεστής Συσχέτισης Pearson (Pearson Correlation)

Ενώ η Cosine Similarity εστιάζει στα χαρακτηριστικά, ο συντελεστής Pearson εστιάζει στη συμπεριφορά. Χρησιμοποιείται κυρίως στο **Collaborative Filtering** για να μετρήσει τη γραμμική συσχέτιση μεταξύ των βαθμολογιών δύο χρηστών.

Ο τύπος του Pearson ενσωματώνει τον μέσο όρο των βαθμολογιών κάθε χρήστη ($\bar{u}$), αφαιρώντας τον από κάθε μεμονωμένη βαθμολογία ($r_{u,i}$):

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \bar{u})(r_{v,i} - \bar{v})}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \bar{u})^2} \sqrt{\sum_{i \in I_{uv}} (r_{v,i} - \bar{v})^2}}$$

Αυτή η προσέγγιση είναι κρίσιμη για τη διαχείριση της υποκειμενικότητας. Αν ο Χρήστης Α βαθμολογεί συνήθως με 4 και 5, ενώ ο Χρήστης Β είναι πιο αυστηρός και βαθμολογεί με 1 και 2, η Cosine Similarity θα τους θεωρούσε ανόμοιους. Ο συντελεστής Pearson όμως αναγνωρίζει ότι και οι δύο προτιμούν ή απορρίπτουν τα ίδια αντικείμενα σε σχέση με τον δικό τους μέσο όρο, εξουδετερώνοντας έτσι τη μεροληψία (bias) της κλίμακας βαθμολόγησης.

Όπως επισημαίνεται στη βιβλιογραφία, η επιλογή μεταξύ αυτών των δύο μέτρων καθορίζει την ικανότητα του συστήματος να αντιλαμβάνεται την πραγματική συνάφεια σε περιβάλλοντα με μεγάλες αποκλίσεις στα δεδομένα των χρηστών [6].

2.3.2 Παραγοντοποίηση Πινάκων (Matrix Factorization)

Η Παραγοντοποίηση Πινάκων (Matrix Factorization - MF) αποτελεί μια από τις πιο εξελιγμένες και μαθηματικά στιβαρές μεθόδους συνεργατικού φιλτραρίσματος, βασισμένη στις αρχές της γραμμικής άλγεβρας. Η ανάγκη για την ανάπτυξή της προέκυψε από μια θεμελιώδη πρόκληση των μεγάλων συστημάτων δεδομένων: το πρόβλημα του **«αραιού πίνακα» (data sparsity)**. Σε πλατφόρμες όπως το **Simply Served**, είναι πρακτικά αδύνατο για έναν χρήστη να βαθμολογήσει το σύνολο των χιλιάδων διαθέσιμων συνταγών. Αυτό δημιουργεί έναν πίνακα R (Χρήστες × Συνταγές) όπου το συντριπτικό ποσοστό των κελιών είναι κενό, καθιστώντας τις παραδοσιακές μεθόδους γειτνίασης (όπως ο k-NN) συχνά ανακριβείς ή υπολογιστικά ασύμφορες.

Η Λογική των Λανθανόντων Χαρακτηριστικών (Latent Factors)

Η κεντρική ιδέα της MF είναι ότι πίσω από τις ρητές βαθμολογίες των χρηστών κρύβεται μια δομή από μη παρατηρήσιμα, **λανθάνοντα χαρακτηριστικά (latent factors)**. Αντί να αναλύουμε τις συνταγές με βάση προκαθορισμένες ετικέτες, ο αλγόριθμος «ανακαλύπτει» αυτόματα τις διαστάσεις που καθορίζουν τις προτιμήσεις.

- **Πίνακας Χρηστών (P):** Κάθε γραμμή αντιστοιχεί σε έναν χρήστη και περιγράφει το βαθμό στον οποίο ο χρήστης αυτός έλκεται από κάθε λανθάνοντα παράγοντα.
- **Πίνακας Αντικειμένων (Q):** Κάθε στήλη αντιστοιχεί σε μια συνταγή και περιγράφει το βαθμό στον οποίο η συνταγή αυτή κατέχει κάθε λανθάνοντα παράγοντα.

Για παράδειγμα, ένας λανθάνων παράγοντας μπορεί να μην έχει όνομα, αλλά μαθηματικά να αντιστοιχεί στην «πολυπλοκότητα της γεύσης» ή στην «καταλληλότητα για βραδινό γεύμα». Η προβλεπόμενη βαθμολογία r_{ui} του χρήστη u για τη συνταγή i υπολογίζεται ως το εσωτερικό γινόμενο των αντίστοιχων διανυσμάτων: $r_{ui} = q_i^T p_u$.

Μηχανισμός Μάθησης και Βελτιστοποίηση

Η διαδικασία εύρεσης των πινάκων P και Q αποτελεί ένα πρόβλημα βελτιστοποίησης. Ο αλγόριθμος προσπαθεί να ελαχιστοποιήσει τη διαφορά μεταξύ των υπαρχουσών βαθμολογιών και των προβλέψεων, χρησιμοποιώντας μια συνάρτηση κόστους που συχνά περιλαμβάνει τακτοποίηση (regularization) για την αποφυγή της υπερπροσαρμογής (overfitting).

Τεχνικές όπως η Singular Value Decomposition (SVD) ή η Stochastic Gradient Descent (SGD) επιτρέπουν στο σύστημα να προσαρμόζει τα διανύσματα των χρηστών και των συνταγών κάθε φορά που προστίθεται μια νέα βαθμολογία μέσω του component RatingStars.jsx. Έτσι, το μοντέλο μαθαίνει συνεχώς και βελτιώνει την ακρίβειά του σε πραγματικό χρόνο.

Πλεονεκτήματα και Σημασία στην Εφαρμογή

Στην εφαρμογή **Simply Served**, η Παραγοντοποίηση Πινάκων προσφέρει τη δυνατότητα για «βαθιά» εξατομίκευση:

1. **Ανακάλυψη Κρυφών Συσχετίσεων:** Το σύστημα μπορεί να προτείνει μια ασιατική συνταγή σε έναν λάτρη της μεξικάνικης κουζίνας, επειδή ο αλγόριθμος εντόπισε ότι και οι δύο κουζίνες μοιράζονται υψηλές τιμές στον λανθάνοντα παράγοντα «ethnic street food».
2. **Κλιμακωσιμότητα (Scalability):** Μόλις υπολογιστούν οι πίνακες P και Q, η παραγωγή μιας σύστασης είναι εξαιρετικά γρήγορη, καθώς απαιτεί έναν απλό πολλαπλασιασμό διανυσμάτων, γεγονός που διασφαλίζει την άμεση απόκριση του Dashboard.

Σύμφωνα με τη βιβλιογραφία, η μέθοδος αυτή έγινε ευρέως γνωστή κατά τη διάρκεια του διαγωνισμού "Netflix Prize", καθώς αποδείχθηκε ανώτερη των κλασικών μεθόδων γειτνίασης, προσφέροντας εξαιρετική κλιμακωσιμότητα (scalability) και ακρίβεια σε περιβάλλοντα με εκατομμύρια δεδομένα [7]. Η χρήση τεχνικών όπως η SVD επιτρέπει στον αλγόριθμο να μαθαίνει δυναμικά από τη συμπεριφορά των χρηστών, μετατρέποντας το Simply Served από μια απλή βάση δεδομένων σε έναν ευφυή σύμβουλο μαγειρικής.

2.3.3 k-Nearest Neighbors (k-NN)

Ο αλγόριθμος των k-Πλησιέστερων Γειτόνων (k-Nearest Neighbors) αποτελεί μια θεμελιώδη και διαισθητική μέθοδο στο συνεργατικό φιλτράρισμα. Η λειτουργία του βασίζεται στην παραδοχή της **συμπεριφορικής συνέπειας**: οι χρήστες που επέδειξαν παρόμοια γούστα στο παρελθόν (π.χ. βαθμολόγησαν υψηλά τις ίδιες μεσογειακές συνταγές), είναι εξαιρετικά πιθανό να συμφωνήσουν και σε μελλοντικές επιλογές. Στην επιστήμη των δεδομένων, ο k-NN χαρακτηρίζεται ως αλγόριθμος **«τεμπέλικης μάθησης» (lazy learning)** ή **instance-based learning**, καθώς δεν εκπαιδεύει ένα στατικό μοντέλο, αλλά αναβάλλει τον υπολογισμό μέχρι τη στιγμή που ο χρήστης ζητά μια σύσταση στο Dashboard του.

Στάδια Υλοποίησης και Μαθηματική Προσέγγιση

Η παραγωγή μιας σύστασης μέσω k-NN στο **Simply Served** ακολουθεί μια αυστηρά δομημένη διαδικασία τριών σταδίων:

1. Υπολογισμός Ομοιότητας (Similarity Computation): Το σύστημα αναπαριστά κάθε χρήστη ως ένα διάνυσμα βαθμολογιών. Για τον εντοπισμό των "γειτόνων", υπολογίζει την απόσταση μεταξύ του χρήστη-στόχου (u) και όλων των άλλων χρηστών (v) στη βάση δεδομένων. Χρησιμοποιούνται μέτρα όπως η Συνημίτονο Ομοιότητα ή ο Συντελεστής Pearson, ο οποίος ορίζεται ως:

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_u})(r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_u})^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r_v})^2}}$$

Όπου I_{uv} είναι το σύνολο των συνταγών που έχουν βαθμολογηθεί και από τους δύο χρήστες.

2. **Επιλογή Γειτονιάς (Neighborhood Selection):** Μετά τον υπολογισμό, το σύστημα κατατάσσει τους χρήστες και επιλέγει τους k επικρατέστερους (δηλαδή αυτούς με τη μεγαλύτερη τιμή sim(u, v)). Η επιλογή της παραμέτρου k είναι κρίσιμη:
    - Ένα πολύ **μικρό k** καθιστά το σύστημα ευάλωτο σε "θόρυβο" ή τυχαίες ακραίες βαθμολογίες (outliers).
    - Ένα πολύ **μεγάλο k** οδηγεί σε γενικευμένες συστάσεις (popular items), εξαλείφοντας την εξατομίκευση.
3. Παραγωγή Πρόβλεψης (Rating Prediction): Η τελική πρόβλεψη για μια συνταγή i που ο χρήστης u δεν έχει δει, προκύπτει ως ο σταθμισμένος μέσος όρος των βαθμολογιών της γειτονιάς:

$$\widehat{r_{ui}} = \bar{r_u} + \frac{\sum_{v \in N_k(u)} sim(u,v) \cdot (r_{vi} - \bar{r_v})}{\sum_{v \in N_k(u)} |sim(u,v)|}$$

Πλεονεκτήματα και Σχεδιαστικοί Περιορισμοί

Ένα από τα σημαντικότερα πλεονεκτήματα του k-NN στην εφαρμογή μας είναι η **ερμηνευσιμότητα (explainability)**. Σε αντίθεση με τις μεθόδους "μαύρου κουτιού" (όπως τα βαθιά νευρωνικά δίκτυα), ο k-NN επιτρέπει στην εφαρμογή να δικαιολογεί τη σύσταση: *"Σας προτείνουμε αυτή τη συνταγή επειδή άρεσε σε 5 χρήστες με παρόμοια γούστα με τα δικά σας"*. Επιπλέον, είναι εξαιρετικά προσαρμοστικός. Κάθε φορά που ένας χρήστης υποβάλλει μια νέα βαθμολογία μέσω του RatingStars.jsx, ο αλγόριθμος την ενσωματώνει άμεσα στον επόμενο υπολογισμό γειτονιάς χωρίς την ανάγκη χρονοβόρας επανεκπαίδευσης.

Ωστόσο, ο k-NN παρουσιάζει προκλήσεις **κλιμακωσιμότητας (scalability)**. Καθώς ο αριθμός των χρηστών αυξάνεται, η σύγκριση του χρήστη-στόχου με κάθε άλλον χρήστη γίνεται υπολογιστικά ακριβή. Για το λόγο αυτό, στην υλοποίηση του Backend (Flask), επιδιώκεται η βελτιστοποίηση των ερωτημάτων στη MySQL και ο περιορισμός του εύρους αναζήτησης σε ενεργούς χρήστες.

Σύμφωνα με τη βιβλιογραφία, ο k-NN παραμένει εξαιρετικά δημοφιλής λόγω της απλότητας στην υλοποίηση και της αποτελεσματικότητάς του σε συστήματα όπου οι προτιμήσεις των χρηστών αλλάζουν δυναμικά, καθώς μπορεί να ενσωματώνει νέα δεδομένα χωρίς να απαιτείται πλήρης επανεκπαίδευση του μοντέλου [8].

2.4 Συστήματα συστάσεων σε εφαρμογές μαγειρικής

Η εφαρμογή συστημάτων συστάσεων στον τομέα της μαγειρικής παρουσιάζει μοναδικές προκλήσεις και ευκαιρίες που διαφοροποιούνται από άλλους τομείς, όπως η μουσική ή ο κινηματογράφος. Ενώ μια ταινία καταναλώνεται ως ενιαία οντότητα, μια συνταγή αποτελείται από επιμέρους συστατικά, γεγονός που καθιστά το **Content-Based Filtering** (βλ. ενότητα 2.2.1) εξαιρετικά αποτελεσματικό. Στις εφαρμογές μαγειρικής, η ομοιότητα δεν υπολογίζεται μόνο βάσει του τελικού πιάτου, αλλά και βάσει της χημικής ή γευστικής συνάφειας των υλικών (flavor pairing).

Στην εφαρμογή **Simply Served**, η υλοποίηση των συστάσεων ακολουθεί μια πολυεπίπεδη προσέγγιση που αντικατοπτρίζει τη συμπεριφορά του χρήστη στην κουζίνα. Η πρώτη διάσταση αφορά την **αναζήτηση βάσει υλικών** (Ingredients.jsx), όπου το σύστημα λειτουργεί ως μηχανισμός φιλτραρίσματος σε πραγματικό χρόνο. Εδώ, η συνεισφορά της εργασίας έγκειται στην ικανότητα του συστήματος να αναγνωρίζει όχι μόνο τι μπορεί να μαγειρέψει ο χρήστης με όσα διαθέτει, αλλά και να του υποδεικνύει τις ελλείψεις του, διευκολύνοντας τη λήψη απόφασης.

Η δεύτερη διάσταση αφορά την **προσωποποίηση μέσω του Dashboard**. Όπως είδαμε στον κώδικα της εφαρμογής, το σύστημα συλλέγει δεδομένα από:

1. **Τις ρητές προτιμήσεις:** Οι βαθμολογίες (RatingStars.jsx) επιτρέπουν στον αλγόριθμο να κατανοήσει το γευστικό προφίλ του χρήστη (π.χ. προτίμηση σε γλυκά ή πικάντικα).
2. **Την έμμεση συμπεριφορά:** Η καταγραφή του ιστορικού προβολών (Recently Viewed) και των αποθηκευμένων συνταγών (WishlistContext.jsx) τροφοδοτεί το σύστημα με πληροφορίες για το τρέχον ενδιαφέρον του χρήστη.
3. **Την ολοκλήρωση εργασιών:** Η λειτουργία "Mark as Cooked" στο RecipeDetails.jsx παρέχει μια κρίσιμη πληροφορία: ποια συνταγή πέρασε από το στάδιο της θεωρητικής αναζήτησης στο στάδιο της πρακτικής υλοποίησης.

Ένα ιδιαίτερο χαρακτηριστικό των συστημάτων αυτών είναι η διαχείριση της **εποχικότητας και των περιορισμών**. Οι σύγχρονες εφαρμογές μαγειρικής καλούνται να ενσωματώνουν φίλτρα για αλλεργίες ή διατροφικές προτιμήσεις (π.χ. Vegan, Gluten-free), τα οποία λειτουργούν ως απόλυτοι περιορισμοί (hard constraints) πριν την εφαρμογή των αλγορίθμων ομοιότητας.

Συμπερασματικά, η ενσωμάτωση συστημάτων συστάσεων στην εφαρμογή Simply Served μετατρέπει την πλατφόρμα από ένα απλό αποθετήριο σε ένα δυναμικό οικοσύστημα. Η επιτυχία της έγκειται στην ισορροπία μεταξύ της **λειτουργικότητας** (εύρεση συνταγής με τα υπάρχοντα υλικά) και της **ανακάλυψης** (πρόταση νέων γεύσεων μέσω του Dashboard), δημιουργώντας μια ολοκληρωμένη ψηφιακή εμπειρία που προσαρμόζεται στις ανάγκες του σύγχρονου χρήστη.

Κεφάλαιο 3ο: Γλώσσες και Τεχνολογίες

Η υλοποίηση μιας σύγχρονης διαδικτυακής εφαρμογής πλήρους στοίβας (Full-stack Web Application) απαιτεί τον συνδυασμό πολλαπλών τεχνολογιών που συνεργάζονται αρμονικά για την παροχή μιας αξιόπιστης και ταχείας εμπειρίας χρήστη. Στο παρόν κεφάλαιο, παρουσιάζεται το τεχνολογικό οικοσύστημα που χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής **Simply Served**.

Η επιλογή των εργαλείων έγινε με γνώμονα την επεκτασιμότητα, την ασφάλεια και την αποδοτικότητα. Η αρχιτεκτονική της εφαρμογής βασίζεται στον διαχωρισμό του προσθίου τμήματος (Frontend), το οποίο διαχειρίζεται τη διεπαφή και την αλληλεπίδραση, από το οπίσθιο τμήμα (Backend), το οποίο αναλαμβάνει τη λογική, την επεξεργασία των δεδομένων και την επικοινωνία με τη βάση δεδομένων.

Στις ενότητες που ακολουθούν, αναλύονται οι βασικές έννοιες των διαδικτυακών εφαρμογών, οι δυνατότητες της βιβλιοθήκης **React.js**, η δομή του framework **Flask**, καθώς και οι μέθοδοι διασφάλισης των δεδομένων μέσω σύγχρονων πρωτοκόλλων αυθεντικοποίησης. Τέλος, γίνεται αναφορά στα εργαλεία ανάπτυξης και τις βιβλιοθήκες που υποστήριξαν τη διαδικασία της κωδικοποίησης, της αποσφαλμάτωσης και της τελικής οργάνωσης του περιβάλλοντος εργασίας.

3.1 Web Applications

Οι διαδικτυακές εφαρμογές (Web Applications) αποτελούν εξελιγμένα συστήματα λογισμικού που εκτελούνται σε απομακρυσμένους διακομιστές (Web Servers) και προσφέρονται στους χρήστες μέσω του πρωτοκόλλου HTTP/HTTPS, με διεπαφή οποιοδήποτε σύγχρονο πρόγραμμα περιήγησης (Browser). Η θεμελιώδης διαφορά τους από τις στατικές ιστοσελίδες (Static Websites) έγκειται στη δυναμική φύση τους: μια εφαρμογή ιστού δεν περιορίζεται στην προβολή πληροφοριών, αλλά επεξεργάζεται δεδομένα, διαχειρίζεται καταστάσεις χρηστών και παράγει εξατομικευμένο περιεχόμενο βάσει των αλληλεπιδράσεων σε πραγματικό χρόνο.

Η Μετάβαση στις Single Page Applications (SPA)

Η εξέλιξη των τεχνολογιών του ιστού και η ανάγκη για πλουσιότερη εμπειρία χρήστη (Rich User Experience) οδήγησαν στην εμφάνιση και κυριαρχία των **Single Page Applications (SPA)**. Πρόκειται για μια αρχιτεκτονική προσέγγιση που υιοθετείται πλήρως στην ανάπτυξη του **Simply Served**. Σε αντίθεση με τις παραδοσιακές Multi-Page Applications (MPA), όπου κάθε κλικ του χρήστη πυροδοτούσε ένα αίτημα στον διακομιστή και την επαναφόρτωση ολόκληρης της σελίδας HTML, οι SPAs λειτουργούν με έναν ριζικά διαφορετικό τρόπο.

Όταν ο χρήστης επισκέπτεται για πρώτη φορά την εφαρμογή, ο διακομιστής στέλνει ένα ελάχιστο αρχείο HTML μαζί με τα απαραίτητα αρχεία JavaScript (το "bundle" της React). Από εκείνο το σημείο και έπειτα, η εφαρμογή «ζει» μέσα στον browser. Οποιαδήποτε αλλαγή στο περιεχόμενο —όπως η αναζήτηση μιας συνταγής βάσει υλικών ή η μετάβαση στο Dashboard— δεν απαιτεί νέα σελίδα. Η εφαρμογή πραγματοποιεί ασύγχρονα αιτήματα (**AJAX/Fetch**) προς το Backend (Flask API), λαμβάνει τα δεδομένα σε μορφή **JSON (JavaScript Object Notation)** και ενημερώνει επιλεκτικά το **Virtual DOM**, αλλάζοντας μόνο τα τμήματα της οθόνης που επηρεάζονται.

Τεχνικά και Λειτουργικά Πλεονεκτήματα

Η υιοθέτηση της SPA αρχιτεκτονικής στο "Simply Served" προσφέρει μια σειρά από κρίσιμα πλεονεκτήματα:

1. **Ταχύτητα και Ροή (Seamless User Flow):** Η εμπειρία του χρήστη είναι πολύ πιο ομαλή, καθώς οι εναλλαγές μεταξύ των λειτουργιών (π.χ. από την αναζήτηση στην προβολή

λεπτομερειών μιας συνταγής) γίνονται σχεδόν ακαριαία. Η έλλειψη της "λευκής οθόνης" κατά τη φόρτωση βελτιώνει καταλυτικά την ικανοποίηση του χρήστη.

2. **Διαχωρισμός Αρμοδιοτήτων (Separation of Concerns):** Η αρχιτεκτονική αυτή επιβάλλει έναν καθαρό διαχωρισμό. Το **Frontend** (React) αναλαμβάνει αποκλειστικά τη λογική της παρουσίασης και την κατάσταση της διεπαφής (UI State), ενώ το **Backend** (Flask) επικεντρώνεται στην επιχειρησιακή λογική (Business Logic) και την ακεραιότητα των δεδομένων. Αυτό επιτρέπει την ευκολότερη συντήρηση και την ανεξάρτητη κλιμάκωση των δύο τμημάτων.
3. **Βελτιστοποίηση Πόρων Δικτύου:** Ανταλλάσσονται μόνο τα απολύτως απαραίτητα δεδομένα. Αντί να μεταφέρονται εκατοντάδες γραμμές κώδικα HTML για κάθε κλικ, μεταφέρεται μόνο ένα μικρό αντικείμενο JSON με τις τιμές της συνταγής, μειώνοντας δραστικά το bandwidth και το φόρτο στον διακομιστή.
4. **Τοπική Αίσθηση (Native-like Experience):** Η εφαρμογή αποκτά τη συμπεριφορά ενός τοπικού προγράμματος (native app). Λειτουργίες όπως το "Mark as Cooked" ή η προσθήκη στη Wishlist εκτελούνται με οπτική επιβεβαίωση σε πραγματικό χρόνο, ενισχύοντας την αίσθηση του ελέγχου από πλευράς του χρήστη.

Στην εφαρμογή **Simply Served**, η δομή αυτή είναι ζωτικής σημασίας για την επιτυχία των συστημάτων συστάσεων. Ο χρήστης μπορεί να πειραματίζεται με διαφορετικά φίλτρα υλικών, να βαθμολογεί συνταγές και να πλοηγείται σε εκατοντάδες επιλογές χωρίς διακοπές στη ροή, καθιστώντας την πλατφόρμα ένα ευέλικτο και εύχρηστο εργαλείο για την καθημερινή του διατροφή.

3.2 React.js

Η **React.js** αποτελεί μια ανοικτού κώδικα βιβλιοθήκη της JavaScript, προσανατολισμένη στη δημιουργία δυναμικών διεπαφών χρήστη (User Interfaces). Αναπτύχθηκε αρχικά από τη Meta (πρώην Facebook) για να επιλύσει προβλήματα κλιμακωσιμότητας και απόδοσης σε εφαρμογές μεγάλης κλίμακας. Σήμερα, η React αποτελεί το βιομηχανικό πρότυπο για το Frontend development, προσφέροντας μια δηλωτική (**declarative**) προσέγγιση στον προγραμματισμό: ο προγραμματιστής περιγράφει την τελική μορφή της διεπαφής για μια συγκεκριμένη κατάσταση δεδομένων και η React αναλαμβάνει την αποδοτική ενημέρωσή της.

Component-Based Αρχιτεκτονική και Επαναχρησιμοποιησιμότητα

Η κεντρική φιλοσοφία της React βασίζεται στην αρχή του **Component-Based Development**. Μια διεπαφή δεν αντιμετωπίζεται ως μια ενιαία σελίδα, αλλά ως ένα ιεραρχικό δέντρο από μικρά, αυτόνομα και επαναχρησιμοποιήσιμα συστατικά (components).

Στην εφαρμογή **Simply Served**, αυτή η αρχιτεκτονική επέτρεψε τον διαχωρισμό της διεπαφής σε λειτουργικές μονάδες:

- **Ατομικά Components:** Όπως το RatingStars.jsx ή το FavoriteButton.jsx, τα οποία διαχειρίζονται τη δική τους εσωτερική λογική.
- **Σύνθετα Components:** Όπως το RecipeCard.jsx, το οποίο ενσωματώνει ατομικά στοιχεία και χρησιμοποιείται σε πολλαπλά σημεία (Home Page, Search Results, Wishlist).
- **Container Components:** Όπως το Dashboard.jsx, που συνθέτουν την τελική σελίδα οργανώνοντας τα επιμέρους στοιχεία.

Ο Μηχανισμός του Virtual DOM και το Reconciliation

Το σημαντικότερο τεχνικό πλεονέκτημα της React, που την καθιστά ιδανική για εφαρμογές με έντονη αλληλεπίδραση όπως το "Simply Served", είναι το **Virtual DOM (Document Object Model)**. Στις παραδοσιακές εφαρμογές, η απευθείας τροποποίηση του DOM του browser είναι μια υπολογιστικά δαπανηρή διαδικασία που μπορεί να προκαλέσει καθυστερήσεις (lag).

Η React επιλύει αυτό το πρόβλημα δημιουργώντας μια ελαφριά αναπαράσταση του DOM στη μνήμη. Η διαδικασία ενημέρωσης ακολουθεί τρία βήματα:

1. **Render:** Όταν αλλάζουν τα δεδομένα (state), η React δημιουργεί ένα νέο Virtual DOM δέντρο.
2. **Diffing:** Συγκρίνει το νέο Virtual DOM με το προηγούμενο για να εντοπίσει τις ακριβείς διαφορές.
3. **Commit (Reconciliation):** Ενημερώνει το πραγματικό DOM του browser **μόνο** στα σημεία που άλλαξαν, αντί να επανασχεδιάσει ολόκληρη τη σελίδα.

Διαχείριση Κατάστασης με React Hooks και Context API

Η εισαγωγή των **Hooks** (όπως τα useState και useEffect) στα Functional Components έφερε επανάσταση στον τρόπο συγγραφής κώδικα.

- **useState:** Επιτρέπει στο component να "θυμάται" πληροφορίες, όπως τα φίλτρα που επέλεξε ο χρήστης ή τα αποτελέσματα μιας αναζήτησης.
- **useEffect:** Διαχειρίζεται τις παρενέργειες (**side effects**), όπως την άντληση δεδομένων από το Flask API μόλις φορτώσει η σελίδα.
- **Context API:** Χρησιμοποιήθηκε στην εφαρμογή για την καθολική διαχείριση της κατάστασης (**Global State Management**). Μέσω του WishlistContext.jsx, η πληροφορία για τα αγαπημένα του χρήστη είναι διαθέσιμη σε όλη την εφαρμογή, διασφαλίζοντας ότι μια αλλαγή στη wishlist σε μια σελίδα αντανακλάται αμέσως παντού.

JSX και Declarative UI

Η χρήση της **JSX (JavaScript XML)** διευκολύνει τη συγγραφή του κώδικα επιτρέποντας τη σύνταξη δομής HTML απευθείας μέσα στη JavaScript. Αυτό προσφέρει μια πιο καθαρή και κατανοητή εικόνα, καθώς η δομή (HTML) και η λογική (JS) συνυπάρχουν στο ίδιο αρχείο, μειώνοντας τα σφάλματα και διευκολύνοντας την αποσφαλμάτωση.

Όπως τεκμηριώνεται στη βιβλιογραφία, η υιοθέτηση των Hooks απλοποίησε τη δομή του κώδικα και κατέστησε τον προγραμματισμό σε React πιο διαισθητικό, επιτρέποντας στους προγραμματιστές να διαχωρίζουν τη λογική από την εμφάνιση με μεγαλύτερη ακρίβεια [9]. Στο "Simply Served", η React αποτέλεσε το θεμέλιο για τη δημιουργία μιας εμπειρίας που είναι ταυτόχρονα υψηλής απόδοσης και εύκολα επεκτάσιμη.

3.3 Διαχείριση κατάστασης εφαρμογών

Η διαχείριση κατάστασης (State Management) αποτελεί έναν από τους κρισιμότερους πυλώνες στη μηχανική λογισμικού διεπαφών. Αναφέρεται στη στρατηγική αποθήκευσης, παρακολούθησης και επικαιροποίησης των δυναμικών δεδομένων που καθορίζουν τι βλέπει ο χρήστης στην οθόνη του σε οποιαδήποτε χρονική στιγμή. Σε περιβάλλοντα **Single Page Applications (SPA)**, η κατάσταση είναι "ζωντανή": μια αλλαγή σε ένα σημείο της εφαρμογής πρέπει να διαχέεται άμεσα και με συνέπεια σε όλα τα υπόλοιπα εξαρτήματα (components) που εξαρτώνται από αυτή.

Η Ιεραρχία της Κατάστασης στο Simply Served

Στην αρχιτεκτονική της εφαρμογής **Simply Served**, η διαχείριση των δεδομένων οργανώθηκε σε δύο διακριτά επίπεδα, προκειμένου να διασφαλιστεί η βέλτιστη απόδοση και η καθαρότητα του κώδικα:

1. **Τοπική Κατάσταση (Local State):** Αφορά δεδομένα με περιορισμένη εμβέλεια, τα οποία δεν επηρεάζουν την υπόλοιπη εφαρμογή. Χρησιμοποιώντας το Hook **useState**, κάθε component διατηρεί τη δική του "μνήμη". Για παράδειγμα, η τιμή μιας φόρμας εισαγωγής υλικών στο Ingredients.jsx ή η κατάσταση εμφάνισης ενός πτυσσόμενου μενού (dropdown) είναι πληροφορίες που αφορούν αποκλειστικά το συγκεκριμένο τμήμα του κώδικα. Η διατήρησή τους σε τοπικό επίπεδο αποτρέπει τους περιττούς επανασχεδιασμούς (re-renders) της υπόλοιπης εφαρμογής.
2. **Καθολική Κατάσταση (Global State):** Περιλαμβάνει δεδομένα που αποτελούν την "κοινή αλήθεια" για πολλαπλά, γεωγραφικά απομακρυσμένα components στο ιεραρχικό δέντρο. Στο "Simply Served", η λίστα των αγαπημένων συνταγών (**Wishlist**) και τα στοιχεία αυθεντικοποίησης του χρήστη είναι τα σημαντικότερα καθολικά δεδομένα. Η πληροφορία ότι μια συνταγή προστέθηκε στα αγαπημένα πρέπει να αντανακλάται ταυτόχρονα στο Navbar (μεταβολή μετρητή), στο Dashboard (εμφάνιση νέας κάρτας) και στο ίδιο το HeartButton της συνταγής.

Η Πρόκληση του Prop Drilling και η Λύση του Context API

Στην κλασική React, η μεταφορά δεδομένων γίνεται μέσω των **props** από τον γονέα στο παιδί. Όταν όμως η πληροφορία πρέπει να φτάσει σε ένα component που βρίσκεται σε μεγάλο βάθος ιεραρχίας, προκύπτει το πρόβλημα του **Prop Drilling**: η ανάγκη να περάσουμε δεδομένα μέσα από ενδιάμεσα components που δεν τα χρειάζονται, απλώς και μόνο για να φτάσουν στον προορισμό τους.

Για την εξάλειψη αυτής της πολυπλοκότητας, χρησιμοποιήθηκε το **Context API**. Μέσω του αρχείου WishlistContext.jsx, δημιουργήθηκε ένας «αγωγός» δεδομένων που παρακάμπτει την ιεραρχία. Η διαδικασία περιλαμβάνει:

- **Provider (Πάροχος):** Ένα ανώτερο component που "εκπέμπει" την κατάσταση και τις συναρτήσεις διαχείρισής της (π.χ. addToWishlist, removeFromWishlist). Στην εφαρμογή μας, ο Provider περιβάλλει το App.js, καθιστώντας τα δεδομένα διαθέσιμα σε κάθε γωνιά του συστήματος.
- **Consumer/useContext (Καταναλωτής):** Οποιοδήποτε component χρειάζεται την πληροφορία, "συνδέεται" στον αγωγό χρησιμοποιώντας το Hook **useContext**, λαμβάνοντας άμεσα τις ενημερώσεις χωρίς ενδιάμεσους σταθμούς.

Αποδοτικότητα και Συγχρονισμός σε Πραγματικό Χρόνο

Η συγκεκριμένη αρχιτεκτονική προσδίδει στο **Simply Served** υψηλή υπολογιστική αποδοτικότητα. Η React χρησιμοποιεί έναν έξυπνο αλγόριθμο σύγκρισης (Diffing) και ενημερώνει μόνο τα τμήματα του Virtual DOM που πραγματικά εξαρτώνται από την κατάσταση που άλλαξε.

Επιπλέον, η χρήση του Context API σε συνδυασμό με το **useEffect** επιτρέπει τον συγχρονισμό της καθολικής κατάστασης με το **LocalStorage** του browser ή τη βάση δεδομένων MySQL μέσω του API. Έτσι, ακόμα και μετά από ανανέωση της σελίδας, οι προτιμήσεις του χρήστη παραμένουν ενεργές, διασφαλίζοντας τη συνέχεια της εμπειρίας (persistence). Αυτή η δομημένη προσέγγιση στη διαχείριση της κατάστασης μετατρέπει την εφαρμογή από μια σειρά στατικών σελίδων σε ένα δυναμικό και συνεκτικό ψηφιακό οικοσύστημα.

3.4 Authentication & Authorization

Η ασφάλεια αποτελεί τον ακρογωνιαίο λίθο κάθε σύγχρονης διαδικτυακής εφαρμογής που διαχειρίζεται προσωπικά δεδομένα και προτιμήσεις χρηστών. Στο "Simply Served", η υλοποίηση της ασφάλειας δεν περιορίζεται σε μια απλή φόρμα σύνδεσης, αλλά βασίζεται σε ένα ολοκληρωμένο πλαίσιο που διακρίνει σαφώς την **Ταυτοποίηση (Authentication)** —την επιβεβαίωση της ταυτότητας— από την **Εξουσιοδότηση (Authorization)** —τον έλεγχο των δικαιωμάτων πρόσβασης.

3.4.1 Ταυτοποίηση μέσω OAuth 2.0 και Google Integration

Η επιλογή του **OAuth 2.0** για την ταυτοποίηση των χρηστών στην εφαρμογή **Simply Served** δεν αποτελεί μια απλή τεχνική ευκολία, αλλά μια στρατηγική απόφαση για τη θωράκιση του συστήματος. Το OAuth 2.0 είναι ένα πρωτόκολλο εξουσιοδότησης που επιτρέπει σε μια εφαρμογή τρίτου μέρους να αποκτήσει περιορισμένη πρόσβαση σε πόρους ενός χρήστη (όπως το προφίλ του στη Google), χωρίς ο χρήστης να αποκαλύψει τα διαπιστευτήριά του (username/password) στην ίδια την εφαρμογή. Αυτό επιτυγχάνει την πλήρη αποσύνδεση των ευαίσθητων δεδομένων από τη βάση δεδομένων της εφαρμογής μας, εκμηδενίζοντας τους κινδύνους που σχετίζονται με την αποθήκευση κωδικών πρόσβασης, όπως οι επιθέσεις "Brute Force" ή οι διαρροές δεδομένων (data breaches).

Η Ροή της Διαδικασίας (Authorization Code Flow)

Η υλοποίηση ακολουθεί το μοντέλο **Authorization Code Flow**, το οποίο θεωρείται το πλέον ασφαλές για εφαρμογές που διαθέτουν Backend (Flask). Η διαδικασία εκτυλίσσεται σε τέσσερα διακριτά επίπεδα αλληλεπίδρασης:

- **Αρχικοποίηση και Ανακατεύθυνση:** Όταν ο χρήστης αλληλεπιδρά με το κουμπί "Login with Google", το Frontend (React) αποστέλλει ένα αίτημα στον Authorization Server της Google, συμπεριλαμβάνοντας το **Client ID** της εφαρμογής και τα επιθυμητά **Scopes** (π.χ. email και profile). Ο χρήστης μεταφέρεται σε ένα περιβάλλον ελεγχόμενο αποκλειστικά από τη Google, διασφαλίζοντας ότι η εφαρμογή Simply Served δεν έχει καμία επαφή με τη διαδικασία πληκτρολόγησης του κωδικού του.
- **Συναίνεση και Έκδοση Κωδικού:** Αφού ο χρήστης ταυτοποιηθεί επιτυχώς από τη Google, καλείται να δώσει ρητή έγκριση για τα δεδομένα που ζητά η εφαρμογή. Με τη συγκατάθεσή του, η Google ανακατευθύνει τον χρήστη πίσω στην εφαρμογή μας, παρέχοντας έναν προσωρινό **Authorization Code**.
- **Ανταλλαγή στο Backend και Επαλήθευση:** Ο κωδικός αυτός αποστέλλεται άμεσα στο Flask Backend. Ο διακομιστής μας, χρησιμοποιώντας το **Client Secret** (ένα κρυφό κλειδί γνωστό μόνο σε εμάς και τη Google), επικοινωνεί με τα API της Google για να ανταλλάξει τον κωδικό με ένα **Access Token** και ένα **ID Token**. Αυτή η επικοινωνία Server-to-Server διασφαλίζει ότι η ταυτότητα του χρήστη επαληθεύεται με απόλυτη εγκυρότητα.

Πλεονεκτήματα και Στρατηγική Ασφάλειας

Αυτή η προσέγγιση προσφέρει το λεγόμενο **Double-Layer Security**. Από τη μία πλευρά, ο χρήστης απολαμβάνει την υψηλή ασφάλεια της υποδομής της Google, η οποία περιλαμβάνει εξελιγμένα συστήματα ανίχνευσης ύποπτης δραστηριότητας και δυνατότητες **Two-Factor Authentication (2FA)**. Από την άλλη πλευρά, η εφαρμογή Simply Served απαλλάσσεται από το τεράστιο βάρος της διαχείρισης και κρυπτογράφησης κωδικών πρόσβασης (hashing/salting), εστιάζοντας αποκλειστικά στην παροχή υπηρεσιών.

Επιπλέον, το Google Integration βελτιώνει καταλυτικά την εμπειρία του χρήστη (User Experience). Η εξάλειψη της ανάγκης για δημιουργία νέου λογαριασμού και η απομνημόνευση ενός ακόμα κωδικού μειώνει τις τριβές κατά την εγγραφή (**Registration Friction**), αυξάνοντας τα ποσοστά υιοθέτησης της εφαρμογής. Τέλος, η χρήση επαληθευμένων email από τη Google διασφαλίζει ότι η βάση δεδομένων μας (MySQL) περιλαμβάνει πραγματικούς χρήστες, διευκολύνοντας τη λειτουργία των συστημάτων συστάσεων και την εξατομίκευση του περιεχομένου.

3.4.2 JSON Web Tokens (JWT) ως Ψηφιακό Διαβατήριο

Η χρήση των **JSON Web Tokens (JWT)** στην εφαρμογή **Simply Served** αποτελεί τη βάση για μια ασφαλή και αποδοτική διαχείριση των συνεδριών (sessions). Στις παραδοσιακές εφαρμογές ιστού, η κατάσταση σύνδεσης διατηρούνταν μέσω των "Server-side Sessions", όπου ο διακομιστής αποθήκευε στη μνήμη του την ταυτότητα κάθε συνδεδεμένου χρήστη. Στη δική μας αρχιτεκτονική, ωστόσο, υιοθετείται το μοντέλο της **Stateless Αυθεντικοποίησης**. Αυτό σημαίνει ότι ο Flask server δεν χρειάζεται να "θυμάται" ποιος χρήστης είναι συνδεδεμένος· αντίθετα, κάθε αίτημα που καταφθάνει από το Frontend (React) φέρει μαζί του όλη την απαραίτητη πληροφορία σε ένα κρυπτογραφημένο πακέτο: το JWT.

Η Ανατομία και η Δομή του Token

Ένα JWT δεν είναι ένα απλό κείμενο, αλλά ένα δομημένο αντικείμενο που αποτελείται από τρία διακριτά μέρη, διαχωρισμένα με τελείες:

- **Header (Κεφαλίδα):** Περιέχει πληροφορίες για τον τύπο του token και τον αλγόριθμο κρυπτογράφησης που χρησιμοποιήθηκε για την παραγωγή της υπογραφής, όπως ο HS256 (HMAC SHA256).
- **Payload (Φορτίο Δεδομένων):** Αποτελεί το λειτουργικό μέρος του token. Εδώ αποθηκεύονται τα "claims", δηλαδή πληροφορίες όπως το μοναδικό αναγνωριστικό του χρήστη (user_id), η ημερομηνία έκδοσης και, το σημαντικότερο, η ημερομηνία λήξης (expiration time). Είναι κρίσιμο να σημειωθεί ότι, παρόλο που το payload είναι κωδικοποιημένο, δεν είναι κρυπτογραφημένο με την παραδοσιακή έννοια, συνεπώς δεν αποθηκεύονται ποτέ ευαίσθητα δεδομένα όπως κωδικοί πρόσβασης σε αυτό το σημείο.
- **Signature (Ψηφιακή Υπογραφή):** Είναι το τμήμα που εγγυάται την ακεραιότητα του token. Παράγεται από τον συνδυασμό της κωδικοποιημένης κεφαλίδας, του φορτίου και ενός **Secret Key** που γνωρίζει μόνο το Backend. Εάν έστω και ένας χαρακτήρας του token τροποποιηθεί από τρίτον, η υπογραφή παύει να είναι έγκυρη και ο διακομιστής απορρίπτει αυτόματα το αίτημα.

Ο Κύκλος Ζωής και η Ασφάλεια των Αιτημάτων

Μόλις το Backend εκδώσει το token, αυτό μεταφέρεται στο Frontend και αποθηκεύεται στο **localStorage**. Από εκείνη τη στιγμή, το JWT λειτουργεί ως ένα ψηφιακό "διαβατήριο" που επιδεικνύεται σε κάθε συνοριακό έλεγχο του API. Σε κάθε αίτημα που απαιτεί ταυτοποίηση —όπως η ανάκτηση της Wishlist ή η υποβολή μιας βαθμολογίας— το Frontend επισυνάπτει το token στο HTTP Header με τη μορφή Authorization: Bearer <token>.

Αυτή η μέθοδος προσφέρει εξαιρετική **κλιμακωσιμότητα (scalability)**, καθώς ο Flask server μπορεί να επαληθεύσει την εγκυρότητα του χρήστη στιγμιαία, πραγματοποιώντας έναν μαθηματικό έλεγχο της υπογραφής, χωρίς να χρειάζεται να ανατρέξει στη βάση δεδομένων MySQL για να επιβεβαιώσει τη

συνεδρία. Επιπλέον, η ενσωμάτωση χρόνου λήξης (TTL - Time To Live) στο token διασφαλίζει ότι, ακόμη και σε περίπτωση υποκλοπής, η πρόσβαση θα είναι περιορισμένη χρονικά. Σε περίπτωση αποτυχίας της επαλήθευσης ή λήξης του token, το σύστημα αποκλείει την πρόσβαση επιστρέφοντας τον κωδικό **401 Unauthorized**, διατηρώντας το Simply Served θωρακισμένο απέναντι σε μη εξουσιοδοτημένες ενέργειες.

3.4.3 Μηχανισμοί Εξουσιοδότησης στο Frontend

Η εξουσιοδότηση στο περιβάλλον της **React** αποτελεί το τελικό στάδιο της γραμμής άμυνας της εφαρμογής, διασφαλίζοντας ότι η διεπαφή ανταποκρίνεται με ακρίβεια στα δικαιώματα του εκάστοτε χρήστη. Ενώ η ταυτοποίηση (Authentication) επιβεβαιώνει την ταυτότητα, η εξουσιοδότηση (Authorization) είναι αυτή που ρυθμίζει τη δυναμική συμπεριφορά του λογισμικού. Στο **Simply Served**, η διαδικασία αυτή υλοποιήθηκε σε δύο συμπληρωματικά επίπεδα: τις Προστατευμένες Διαδρομές και την Εξαρτημένη Εμφάνιση, δημιουργώντας ένα περιβάλλον που προστατεύει τα ευαίσθητα δεδομένα χωρίς να θυσιάζει την ευχρηστία.

Προστατευμένες Διαδρομές (Private Routes) και Διαχείριση Πλοήγησης

Το πρώτο επίπεδο εξουσιοδότησης αφορά τον έλεγχο της πρόσβασης σε ολόκληρες σελίδες ή ενότητες της εφαρμογής. Μέσω του συστατικού PrivateRoute.jsx, υλοποιήθηκε ένας μηχανισμός «φύλακα» (guard) ο οποίος παρεμβάλλεται κατά τη διαδικασία της δρομολόγησης (**routing**). Πριν η React επιτρέψει τη φόρτωση κρίσιμων σελίδων, όπως το Dashboard.jsx ή η σελίδα διαχείρισης προφίλ, το σύστημα εκτελεί έναν έλεγχο παρουσίας και εγκυρότητας του JWT στο localStorage.

Εάν ο χρήστης επιχειρήσει να αποκτήσει πρόσβαση σε μια προστατευμένη διαδρομή χωρίς έγκυρο token —για παράδειγμα, πληκτρολογώντας απευθείας το URL στον browser— ο μηχανισμός τον αναχαιτίζει και τον ανακατευθύνει αυτόματα στη σελίδα σύνδεσης (Login). Αυτή η αρχιτεκτονική διασφαλίζει ότι το περιεχόμενο που αφορά το ιστορικό μαγειρικής και τις προσωπικές λίστες παραμένει απροσπέλαστο σε μη εξουσιοδοτημένα άτομα, αποτρέποντας την έκθεση δεδομένων.

Εξαρτημένη Εμφάνιση (Conditional Rendering) και Διαδραστικότητα

Το δεύτερο επίπεδο εξουσιοδότησης λειτουργεί με πιο λεπτομερή τρόπο (fine-grained control) στο επίπεδο των επιμέρους στοιχείων της διεπαφής. Η **Εξαρτημένη Εμφάνιση (Conditional Rendering)** επιτρέπει στην εφαρμογή να προσαρμόζει τη συμπεριφορά των κουμπιών και των λειτουργιών της ανάλογα με την κατάσταση σύνδεσης του χρήστη σε πραγματικό χρόνο.

Χαρακτηριστικό παράδειγμα αποτελεί το HeartButton.jsx. Ενώ ένας επισκέπτης μπορεί να δει τις συνταγές, η δυνατότητα προσθήκης στη Wishlist απαιτεί ταυτοποίηση. Αν ο χρήστης δεν είναι συνδεδεμένος, το κουμπί μπορεί είτε να παραμένει κρυφό, είτε να εμφανίζει ένα προειδοποιητικό μήνυμα (modal) που τον προτρέπει να συνδεθεί. Αντίστοιχα, το σύστημα βαθμολόγησης RatingStars.jsx κλειδώνει τη δυνατότητα υποβολής βαθμολογίας για τους ανώνυμους χρήστες. Αυτό είναι κρίσιμο για την ακεραιότητα του συστήματος συστάσεων, καθώς διασφαλίζει ότι οι αλγόριθμοι τροφοδοτούνται μόνο με δεδομένα από πραγματικά, επαληθευμένα προφίλ, προστατεύοντας τη βάση δεδομένων από "θόρυβο" ή κακόβουλες αλληλεπιδράσεις.

Πολυεπίπεδη Ασφάλεια και Εμπειρία Χρήστη

Ο συνδυασμός των παραπάνω μηχανισμών επιτρέπει στο "Simply Served" να ακολουθεί τα σύγχρονα πρότυπα του κλάδου (**Industry Standards**), όπου η ασφάλεια ενσωματώνεται οργανικά στη σχεδίαση (Security by Design). Η χρήση του **Context API** για τη διάχυση της κατάστασης αυθεντικοποίησης σε

όλα τα components διασφαλίζει ότι ολόκληρη η εφαρμογή αντιδρά ακαριαία όταν ένας χρήστης αποσυνδέεται ή το token του λήγει. Με αυτόν τον τρόπο, η ευκολία πρόσβασης μέσω Single Sign-On (Google) συνυπάρχει αρμονικά με έναν αυστηρό έλεγχο δικαιωμάτων, προσφέροντας μια προσωποποιημένη και, πάνω απ' όλα, ασφαλή γαστρονομική εμπειρία.

3.5 REST APIs & Επικοινωνία Frontend–Backend

Η αρχιτεκτονική **REST (Representational State Transfer)** αποτελεί το κυρίαρχο πρότυπο επικοινωνίας στον σύγχρονο ιστό, παρέχοντας ένα σύνολο κανόνων και περιορισμών που επιτρέπουν σε δύο ανεξάρτητα συστήματα να αλληλεπιδρούν απρόσκοπτα. Στην εφαρμογή **Simply Served**, η REST αρχιτεκτονική λειτουργεί ως η «κοινή γλώσσα» που επιτρέπει στο **React Frontend** και στο **Flask Backend** να ανταλλάσσουν πληροφορίες, παρά το γεγονός ότι είναι γραμμένα σε διαφορετικές γλώσσες προγραμματισμού και εκτελούνται σε διαφορετικά περιβάλλοντα.

3.5.1 Πόροι (Resources) και Endpoints

Στην αρχιτεκτονική REST, η έννοια του **«Πόρου» (Resource)** αποτελεί το δομικό στοιχείο κάθε αλληλεπίδρασης. Ένας πόρος δεν είναι απλώς μια εγγραφή σε μια βάση δεδομένων, αλλά οποιαδήποτε πληροφορική οντότητα μπορεί να κατονομαστεί, να ανακτηθεί ή να τροποποιηθεί από το σύστημα. Στο **Simply Served**, οι πόροι κατηγοριοποιούνται με βάση τη λειτουργικότητά τους: από φυσικές οντότητες όπως οι "Συνταγές" (Recipes) και τα "Υλικά" (Ingredients), μέχρι αφηρημένες έννοιες όπως οι "Συλλογές Αγαπημένων" (Wishlist) και τα "Αποτελέσματα Συστάσεων" (Recommendations).

Η Φιλοσοφία του URI και η Ιεραρχία των Endpoints

Κάθε πόρος είναι προσβάσιμος μέσω ενός μοναδικού **Endpoint**, το οποίο ακολουθεί μια ιεραρχική και λογική δομή διευθύνσεων URL. Η τυποποίηση αυτή διασφαλίζει ότι το Frontend (React) μπορεί να προβλέψει τη διαδρομή προς τα δεδομένα χωρίς να χρειάζεται πολύπλοκη παραμετροποίηση. Στην εφαρμογή μας, η οργάνωση των endpoints ακολουθεί το πρότυπο της «ουσιαστικοποίησης», όπου τα URLs περιλαμβάνουν ουσιαστικά που περιγράφουν τον πόρο και όχι ρήματα που περιγράφουν την ενέργεια.

Για παράδειγμα, η διαχείριση των συνταγών οργανώνεται ως εξής:

- /api/recipes: Αντιπροσωπεύει τη συλλογή (collection) όλων των διαθέσιμων συνταγών.
- /api/recipes/512: Αντιπροσωπεύει έναν συγκεκριμένο πόρο (instance), δηλαδή τη συνταγή με το μοναδικό αναγνωριστικό 512.
- /api/recipes/category/vegetarian: Αντιπροσωπεύει ένα υποσύνολο πόρων που φιλτράρονται βάσει ενός συγκεκριμένου χαρακτηριστικού.

Η Σύνδεση Πόρων και η Σημασιολογική Συσχέτιση

Ένα κρίσιμο χαρακτηριστικό των Endpoints στο **Simply Served** είναι η ικανότητά τους να εκφράζουν σχέσεις μεταξύ διαφορετικών πόρων. Αυτό επιτυγχάνεται μέσω της εμφώλευσης (**nesting**) των διευθύνσεων, η οποία αντικατοπτρίζει τις σχέσεις "ένα-προς-πολλά" στη βάση δεδομένων MySQL.

Για παράδειγμα, η ανάκτηση των βαθμολογιών ενός συγκεκριμένου χρήστη μπορεί να γίνει μέσω του endpoint /api/users/12/ratings. Αυτή η δομή επιτρέπει στο Frontend να διατηρεί έναν καθαρό κώδικα, καθώς η ίδια η διεύθυνση URL «αυτο-περιγράφει» τη φύση των δεδομένων που θα επιστραφούν. Επιπλέον, η χρήση παραμέτρων ερωτήματος (**Query Parameters**) επιτρέπει τον περαιτέρω έλεγχο των

πόρων, όπως στο /api/recipes/search?ingredients=chicken,onion, όπου το endpoint παραμένει σταθερό αλλά το περιεχόμενο προσαρμόζεται δυναμικά στις ανάγκες του χρήστη.

Οργάνωση και Προβλεψιμότητα

Αυτή η αυστηρή οργάνωση των Endpoints προσφέρει σημαντικά οφέλη στη διαδικασία ανάπτυξης:

1. **Αποσύνδεση (Decoupling):** Το Frontend δεν χρειάζεται να γνωρίζει πώς είναι δομημένη η βάση δεδομένων ή πώς εκτελείται το SQL ερώτημα. Χρειάζεται μόνο να γνωρίζει το Endpoint.
2. **Επεκτασιμότητα (Extensibility):** Αν στο μέλλον προστεθεί ένας νέος πόρος (π.χ. "Λίστα Σούπερ Μάρκετ"), η ενσωμάτωσή του θα γίνει μέσω ενός νέου endpoint /api/shopping-list, χωρίς να επηρεαστούν οι υπάρχουσες λειτουργίες.
3. **Ευκολία Αποσφαλμάτωσης:** Η χρήση εργαλείων όπως το **Postman** επιτρέπει τη μεμονωμένη δοκιμή κάθε πόρου, διασφαλίζοντας ότι το Flask Backend επιστρέφει το σωστό αντικείμενο JSON πριν αυτό καταναλωθεί από τη React.

Με την υιοθέτηση αυτής της δομής, το **Simply Served** μετατρέπεται από μια απλή ιστοσελίδα σε ένα στιβαρό API-driven σύστημα, όπου η πληροφορία ρέει με τάξη, ασφάλεια και απόλυτη προβλεψιμότητα.

3.5.2 Μέθοδοι HTTP και Σημαντική Λειτουργικότητα

Η επικοινωνία μεταξύ του React Frontend και του Flask Backend στο **Simply Served** δεν είναι μια απλή ανταλλαγή κειμένου, αλλά μια αυστηρά τυποποιημένη διαδικασία που βασίζεται στο μοντέλο **Αιτήματος-Απόκρισης (Request-Response)**. Κάθε αίτημα HTTP φέρει μια συγκεκριμένη «μέθοδο» (HTTP Verb), η οποία λειτουργεί ως εντολή προς τον διακομιστή, προσδιορίζοντας με ακρίβεια τη σκοπούμενη ενέργεια επί των πόρων. Η σωστή χρήση αυτών των μεθόδων είναι κρίσιμη για τη δημιουργία ενός προβλέψιμου και εύκολα συντηρήσιμου API.

1. GET: Η Μέθοδος της Ασφαλούς Ανάκτησης

Η μέθοδος **GET** αποτελεί τη ραχοκοκαλιά της πλοήγησης στην εφαρμογή. Χαρακτηρίζεται ως «ασφαλής» (safe) και «αμετάβλητη» (idempotent), καθώς η κλήση της δεν προκαλεί αλλαγές στην κατάσταση της βάσης δεδομένων. Στο **Simply Served**, η GET χρησιμοποιείται εκτενώς για:

- Την άντληση του πλήρους καταλόγου συνταγών κατά την αρχική φόρτωση.
- Την προβολή λεπτομερειών μιας συνταγής στο RecipeDetails.jsx, όπου το ID της συνταγής περνάει ως παράμετρος στο URL.
- Τη λήψη των εξατομικευμένων προτάσεων στο Dashboard. Η ταχύτητα των GET αιτημάτων ενισχύεται συχνά μέσω μηχανισμών προσωρινής αποθήκευσης (**caching**), βελτιώνοντας την εμπειρία του χρήστη.

2. POST: Δημιουργία και Επεξεργασία Δεδομένων

Σε αντίθεση με τη GET, η μέθοδος **POST** χρησιμοποιείται όταν ο χρήστης επιθυμεί να εισάγει νέα δεδομένα στο σύστημα ή να πυροδοτήσει λειτουργίες που αλλάζουν την κατάσταση του Backend. Είναι η μέθοδος που "μεταφέρει" δεδομένα στο σώμα (**body**) του αιτήματος, συνήθως σε μορφή JSON. Στην εφαρμογή μας, η POST είναι ζωτικής σημασίας για:

- Την εγγραφή και σύνδεση χρηστών μέσω του Login.jsx.

- Την αποστολή των επιλεγμένων υλικών από το Ingredients.jsx: εδώ η POST δεν δημιουργεί απαραίτητα μια εγγραφή, αλλά στέλνει ένα σύνολο δεδομένων που το Flask επεξεργάζεται για να εκτελέσει τον αλγόριθμο σύστασης.
- Την υποβολή βαθμολογιών στο RatingStars.jsx, όπου δημιουργείται μια νέα εγγραφή στον πίνακα βαθμολογιών.

3. PUT και PATCH: Η Λογική της Ενημέρωσης

Αυτές οι μέθοδοι χρησιμοποιούνται για την τροποποίηση υπαρχόντων πόρων. Η διαφορά τους είναι λεπτή αλλά σημαντική για την αποδοτικότητα του δικτύου:

- **PUT:** Χρησιμοποιείται για την πλήρη αντικατάσταση ενός πόρου. Αν ο χρήστης αλλάξει ολόκληρο το προφίλ του, στέλνεται ένα PUT αίτημα με όλα τα πεδία.
- **PATCH:** Χρησιμοποιείται για μερική ενημέρωση (partial update). Εάν ο χρήστης αλλάξει μόνο τη βαθμολογία του σε μια συνταγή, το RatingStars.jsx μπορεί να στείλει ένα PATCH αίτημα που περιέχει μόνο τη νέα τιμή, εξοικονομώντας πόρους δικτύου.

4. DELETE: Καθαρισμός και Αφαίρεση

Η μέθοδος **DELETE** είναι αυστηρά προορισμένη για την κατάργηση πόρων. Στο **Simply Served**, η κύρια εφαρμογή της εντοπίζεται στη διαχείριση της Wishlist. Όταν ο χρήστης αποεπιλέγει μια συνταγή από τα αγαπημένα του, το Frontend στέλνει ένα DELETE αίτημα στο endpoint του συγκεκριμένου πόρου (π.χ. /api/wishlist/recipe_id). Ο διακομιστής Flask αναλαμβάνει τότε να αφαιρέσει τη σχετική εγγραφή από τη MySQL, διασφαλίζοντας ότι η λίστα του χρήστη παραμένει επίκαιρη.

Σημασιολογική Ακρίβεια και Ασφάλεια

Η τήρηση αυτών των συμβάσεων επιτρέπει στο **Simply Served** να λειτουργεί με βάση τα διεθνή πρότυπα. Για παράδειγμα, το Backend είναι ρυθμισμένο να απαιτεί **JWT Authentication** ειδικά για τις μεθόδους POST, PUT και DELETE, ενώ επιτρέπει ορισμένες GET κλήσεις (όπως την προβολή δημόσιων συνταγών) σε ανώνυμους χρήστες. Αυτός ο προγραμματιστικός διαχωρισμός προσφέρει ένα επιπλέον επίπεδο ασφαλείας και καθιστά την επικοινωνία Frontend-Backend διαφανή και αποτελεσματική.

3.5.3 Statelessness και η χρήση του JSON

Η αρχιτεκτονική REST επιβάλλει συγκεκριμένους περιορισμούς που καθιστούν την επικοινωνία μεταξύ συστημάτων αξιόπιστη και ταχύτατη. Δύο από τα πλέον καθοριστικά στοιχεία στην εφαρμογή **Simply Served** είναι η αρχή της **Έλλειψης Κατάστασης (Statelessness)** και η υιοθέτηση του προτύπου **JSON** για την αναπαράσταση των δεδομένων.

Η Αρχή του Statelessness (Έλλειψη Κατάστασης)

Σε αντίθεση με τις παλαιότερες τεχνολογίες ιστού όπου ο διακομιστής διατηρούσε "συνεδρίες" (sessions) στη μνήμη του για να αναγνωρίζει κάθε χρήστη, η αρχή του **Statelessness** ορίζει ότι κάθε αίτημα (request) από το Frontend προς το Flask Backend πρέπει να είναι πλήρες και αυτοτελές. Αυτό σημαίνει ότι ο διακομιστής δεν αποθηκεύει καμία πληροφορία για το ιστορικό των προηγούμενων αιτημάτων του χρήστη.

Αυτή η προσέγγιση προσφέρει σημαντικά πλεονεκτήματα:

- **Κλιμακωσιμότητα (Scalability):** Επειδή ο διακομιστής δεν χρειάζεται να διαχειρίζεται τη μνήμη για χιλιάδες ενεργές συνεδρίες, μπορεί να εξυπηρετεί ταυτόχρονα έναν τεράστιο αριθμό χρηστών με ελάχιστους πόρους.
- **Ανθεκτικότητα σε Σφάλματα:** Εάν ο διακομιστής χρειαστεί επανεκκίνηση, κανένας χρήστης δεν "αποσυνδέεται" βίαια, καθώς η κατάσταση της σύνδεσης δεν βασίζεται στη μνήμη του server αλλά στο **JWT Token** που αποστέλλει η React σε κάθε κλήση.
- **Απλότητα στη Συντήρηση:** Η λογική του Backend παραμένει καθαρή, καθώς επικεντρώνεται αποκλειστικά στην επεξεργασία του τρέχοντος αιτήματος με βάση τα δεδομένα που λαμβάνει εκείνη τη στιγμή.

JSON: Το "Όχημα" της Πληροφορίας

Η ανταλλαγή των δεδομένων πραγματοποιείται αποκλειστικά σε μορφή **JSON (JavaScript Object Notation)**. Πρόκειται για ένα κειμενοστρεφές πρότυπο που είναι ανεξάρτητο από γλώσσες προγραμματισμού, γεγονός που το καθιστά την "τέλεια γέφυρα" μεταξύ της **Python** (Backend) και της **JavaScript** (Frontend).

Η επιλογή του JSON για το **Simply Served** βασίστηκε σε τρεις παράγοντες:

1. **Ελαφριά Δομή:** Το JSON έχει ελάχιστο "overhead". Μεταφέρει την πληροφορία χρησιμοποιώντας μια απλή δομή κλειδιού-τιμής (key-value pairs), γεγονός που μειώνει τον όγκο των δεδομένων που διακινούνται στο δίκτυο και επιταχύνει τους χρόνους φόρτωσης.
2. **Άμεση Συμβατότητα:** Η React μπορεί να μετατρέψει ένα αντικείμενο JSON σε κατάσταση (state) της εφαρμογής με μία μόνο γραμμή κώδικα, επιτρέποντας την ακριαία ενημέρωση του UI (π.χ. εμφάνιση των συστατικών μιας συνταγής μόλις ληφθούν από το API).
3. **Ευανάγνωστη Μορφή:** Η δομή του είναι κατανοητή τόσο από τον άνθρωπο όσο και από τη μηχανή, διευκολύνοντας τη διαδικασία της αποσφαλμάτωσης (debugging) κατά την ανάπτυξη.

Στο **Simply Served**, όταν το Flask εκτελεί έναν αλγόριθμο σύστασης, συγκεντρώνει τα αποτελέσματα από τη MySQL, τα "πακετάρει" σε ένα σύνθετο αντικείμενο JSON (που περιλαμβάνει λίστες συνταγών, ποσοστά ομοιότητας και μεταδεδομένα) και τα αποστέλλει στο Frontend. Η React λαμβάνει αυτό το "πακέτο" και, χωρίς να χρειαστεί ανανέωση της σελίδας, επανασχεδιάζει δυναμικά το Dashboard, προσφέροντας στον χρήστη μια συνεχή και διαδραστική εμπειρία που θυμίζει τοπική εφαρμογή υψηλών επιδόσεων.

3.6 Flask

Το **Flask** αποτελεί ένα από τα πλέον δημοφιλή και ισχυρά εργαλεία για την ανάπτυξη διαδικτυακών εφαρμογών και API στη γλώσσα προγραμματισμού Python. Ως **micro-framework**, το Flask υιοθετεί μια μινιμαλιστική προσέγγιση: αντί να επιβάλλει μια αυστηρή δομή και προκαθορισμένα εργαλεία, παρέχει μόνο τα βασικά συστατικά για τη διαχείριση των HTTP αιτημάτων και της δρομολόγησης, αφήνοντας στον προγραμματιστή την πλήρη ελευθερία να επιλέξει τις βιβλιοθήκες που εξυπηρετούν καλύτερα το έργο του. Στην αρχιτεκτονική του **Simply Served**, το Flask λειτουργεί ως ο κεντρικός «εγκέφαλος» που γεφυρώνει τη διεπαφή του χρήστη με τα δεδομένα και τους αλγορίθμους.

3.6.1 Μηχανισμός Δρομολόγησης και View Functions

Ο μηχανισμός δρομολόγησης (**Routing**) αποτελεί το νευρικό σύστημα του Backend στο **Simply Served**. Είναι η διαδικασία με την οποία το Flask αναλύει το εισερχόμενο URL από το Frontend και

αποφασίζει ποιο τμήμα του κώδικα Python πρέπει να εκτελεστεί. Αυτή η αντιστοίχιση επιτυγχάνεται μέσω των **Python Decorators** (διακοσμητών), και συγκεκριμένα της σύνταξης @app.route(), η οποία "δένει" μια διεύθυνση URL με μια συγκεκριμένη συνάρτηση, τη λεγόμενη **View Function** [10].

Η Αρχιτεκτονική του Request-Response Cycle

Η λειτουργία των View Functions εντός του Flask ακολουθεί μια αυστηρά προκαθορισμένη ροή εργασίας, διασφαλίζοντας ότι η επικοινωνία παραμένει σταθερή και προβλέψιμη:

1. **Request Context & Parsing (Διαχείριση Αιτήματος):** Μόλις το Flask λάβει ένα αίτημα, δημιουργεί ένα "Request Context". Η View Function έχει πρόσβαση στο αντικείμενο request, από το οποίο μπορεί να εξάγει δεδομένα. Για παράδειγμα, στην αναζήτηση συνταγών, ο μηχανισμός εξάγει τα επιλεγμένα υλικά από το σώμα του αιτήματος (request.json) ή από τις παραμέτρους του URL (request.args).
2. **Business Logic Execution (Εκτέλεση Λογικής):** Αυτό είναι το στάδιο όπου η View Function λειτουργεί ως ενορχηστρωτής. Στην περίπτωση της συνάρτησης search_by_ingredients, ο αλγόριθμος:
    - Ανακτά το σύνολο των συνταγών από τη MySQL.
    - Συγκρίνει τις λίστες των υλικών χρησιμοποιώντας δομές δεδομένων της Python (όπως τα sets).
    - Υπολογίζει δυναμικά τα "Matched" (υλικά που διαθέτει ο χρήστης) και τα "Missing" (υλικά που λείπουν), ταξινομώντας τα αποτελέσματα με βάση το ποσοστό συνάφειας.
3. **Data Serialization & Response (Σειριοποίηση):** Επειδή η React δεν μπορεί να κατανοήσει απευθείας τα αντικείμενα της Python, η View Function πρέπει να μετατρέψει τα αποτελέσματα σε **JSON**. Η συνάρτηση jsonify() αναλαμβάνει τη σειριοποίηση των δεδομένων και ταυτόχρονα προσθέτει τα απαραίτητα HTTP Headers (όπως το Content-Type: application/json), ενημερώνοντας το Frontend ότι η απάντηση είναι έτοιμη για επεξεργασία.

Δυναμική Δρομολόγηση και Παράμετροι

Ένα προηγμένο χαρακτηριστικό που χρησιμοποιήθηκε στο **Simply Served** είναι η **Δυναμική Δρομολόγηση (Dynamic Routing)**. Αντί να δημιουργούνται εκατοντάδες στατικές διαδρομές για κάθε συνταγή, χρησιμοποιούνται μεταβλητές εντός του URL, όπως @app.route('/api/recipe/<int:recipe_id>').

Σε αυτή την περίπτωση, το Flask:

- Αναγνωρίζει το τμήμα του URL ως ακέραιο αριθμό (recipe_id).
- Το μεταβιβάζει αυτόματα ως όρισμα στη View Function.
- Επιτρέπει στη συνάρτηση να εκτελέσει ένα στοχευμένο ερώτημα στη βάση δεδομένων για τη συγκεκριμένη συνταγή.

Διαχείριση Σφαλμάτων και HTTP Status Codes

Οι View Functions στο Backend μας δεν επιστρέφουν μόνο δεδομένα, αλλά και την κατάσταση της επιτυχίας ή αποτυχίας της ενέργειας μέσω των **HTTP Status Codes**. Εάν μια συνταγή δεν βρεθεί, η συνάρτηση επιστρέφει ένα αντικείμενο JSON με μήνυμα σφάλματος και τον κωδικό 404 Not Found. Εάν η αναζήτηση είναι επιτυχής, επιστρέφει τον κωδικό 200 OK. Αυτή η τυποποίηση επιτρέπει στο Frontend να εμφανίζει ανάλογα μηνύματα στον χρήστη (π.χ. "Δεν βρέθηκαν αποτελέσματα"), διατηρώντας την εφαρμογή στιβαρή και φιλική προς τον χρήστη.

3.6.2 Επεκτασιμότητα μέσω Extensions

Η φιλοσοφία του Flask ως **micro-framework** βασίζεται στην αρχή ότι ο προγραμματιστής πρέπει να έχει τον απόλυτο έλεγχο των εργαλείων που χρησιμοποιεί. Αντί να προσφέρει προεγκατεστημένες λύσεις για κάθε ανάγκη, το Flask βασίζεται σε ένα πλούσιο οικοσύστημα από **Extensions (Επεκτάσεις)**. Αυτές οι βιβλιοθήκες ενσωματώνονται οργανικά στον κύκλο ζωής της εφαρμογής, προσφέροντας εξειδικευμένες δυνατότητες που μετατρέπουν το "Simply Served" από έναν απλό διακομιστή σε μια ολοκληρωμένη και ασφαλή πλατφόρμα.

Flask-CORS: Γεφυρώνοντας το Χάσμα μεταξύ Προελεύσεων

Σε μια αρχιτεκτονική διαχωρισμένου Frontend και Backend, το **CORS (Cross-Origin Resource Sharing)** αποτελεί έναν κρίσιμο μηχανισμό ασφαλείας των προγραμμάτων περιήγησης. Από προεπιλογή, ένας browser απαγορεύει σε μια εφαρμογή (π.χ. React στη θύρα 3000) να διαβάζει δεδομένα από έναν διακομιστή διαφορετικής προέλευσης (π.χ. Flask στη θύρα 5000).

Η επέκταση **Flask-CORS** παραμετροποιήθηκε στο "Simply Served" ώστε να επιτρέπει επιλεκτικά τα αιτήματα από τη συγκεκριμένη διεύθυνση του Frontend. Μέσω της προσθήκης των κατάλληλων HTTP Headers στις αποκρίσεις του διακομιστή, το Flask-CORS ενημερώνει τον browser ότι η επικοινωνία είναι εξουσιοδοτημένη, επιτρέποντας την απρόσκοπτη ανταλλαγή JSON δεδομένων χωρίς να διακυβεύεται η ασφάλεια έναντι κακόβουλων τρίτων ιστότοπων.

Flask-JWT-Extended: Προηγμένη Διαχείριση Ταυτοποίησης

Η διαχείριση των JSON Web Tokens απαιτεί αυστηρά πρωτόκολλα για τη διασφάλιση της ακεραιότητας των δεδομένων. Το **Flask-JWT-Extended** αποτελεί μια ισχυρή επέκταση που αυτοματοποιεί τις πιο σύνθετες πτυχές αυτής της διαδικασίας:

- **Token Creation:** Μετά την επιτυχή σύνδεση μέσω Google, η βιβλιοθήκη παράγει το token ενσωματώνοντας τα απαραίτητα "claims" (όπως το identity του χρήστη).
- **Protection Decorators:** Χρησιμοποιώντας τον διακοσμητή @jwt_required(), η βιβλιοθήκη προστατεύει αυτόματα τα endpoints. Εάν ένα αίτημα φτάσει χωρίς έγκυρο token, η επέκταση αναχαιτίζει την κλήση και επιστρέφει αυτόματα σφάλμα 401, πριν καν εκτελεστεί η View Function.
- **Namespace Management:** Επιτρέπει τον εύκολο διαχωρισμό μεταξύ Access Tokens (μικρής διάρκειας) και Refresh Tokens (μεγάλης διάρκειας), ενισχύοντας τη συνολική στιβαρότητα της εφαρμογής.

PyMySQL και SQLAlchemy: Η Γέφυρα με τα Δεδομένα

Για την αλληλεπίδραση με τη βάση δεδομένων MySQL, χρησιμοποιήθηκε ο οδηγός **PyMySQL**, συχνά σε συνδυασμό με το **SQLAlchemy** (έναν Object-Relational Mapper - ORM). Αυτές οι επεκτάσεις επιτρέπουν στην Python να επικοινωνεί με τη βάση δεδομένων χρησιμοποιώντας αντικείμενα και κώδικα αντί για αμιγείς εντολές SQL σε μορφή κειμένου.

- **Connection Pooling:** Διαχειρίζονται αποτελεσματικά τις συνδέσεις με τη βάση, διασφαλίζοντας ότι η εφαρμογή δεν εξαντλεί τους πόρους του συστήματος κατά τη διάρκεια περιόδων υψηλής κίνησης.
- **Abstraction Layer:** Επιτρέπουν τη σύνταξη ερωτημάτων που είναι πιο ευανάγνωστα και ασφαλή έναντι επιθέσεων τύπου **SQL Injection**, καθώς η βιβλιοθήκη αναλαμβάνει αυτόματα το "sanitization" των δεδομένων που εισάγει ο χρήστης (π.χ. τα ονόματα των υλικών στην αναζήτηση).

Η χρήση αυτών των επεκτάσεων αναδεικνύει την ευελιξία του Flask: το "Simply Served" χρησιμοποιεί ακριβώς τα εργαλεία που χρειάζεται, διατηρώντας την απόδοση σε υψηλά επίπεδα και την υπολογιστική πολυπλοκότητα στο ελάχιστο δυνατό, ακολουθώντας τις βέλτιστες πρακτικές της σύγχρονης ανάπτυξης Backend [10].

3.6.3 Περιβάλλον Ανάπτυξης και Απομόνωση (venv)

Στη σύγχρονη ανάπτυξη λογισμικού με Python, η διαχείριση των εξαρτήσεων (dependencies) αποτελεί μια κρίσιμη παράμετρο για τη σταθερότητα του συστήματος. Η ανάπτυξη του Backend της εφαρμογής **Simply Served** πραγματοποιήθηκε με τη χρήση **Virtual Environments (venv)**, μια πρακτική που επιβάλλει τον πλήρη διαχωρισμό του περιβάλλοντος εκτέλεσης της εφαρμογής από το καθολικό (global) περιβάλλον της Python του λειτουργικού συστήματος.

Η Ανάγκη για Απομόνωση (Dependency Isolation)

Η χρήση ενός εικονικού περιβάλλοντος επιλύει το πρόβλημα των συγκρούσεων μεταξύ διαφορετικών εκδόσεων βιβλιοθηκών (**Dependency Hell**). Χωρίς το venv, αν μια άλλη εφαρμογή στο ίδιο σύστημα απαιτούσε μια παλαιότερη έκδοση του Flask, η αναβάθμισή του για τις ανάγκες του "Simply Served" θα μπορούσε να καταστήσει την πρώτη εφαρμογή μη λειτουργική.

Με τη δημιουργία ενός τοπικού φακέλου περιβάλλοντος, όλες οι απαραίτητες βιβλιοθήκες (όπως το Flask, το PyMySQL και το Flask-JWT-Extended) εγκαθίστανται αποκλειστικά για το συγκεκριμένο project. Αυτό διασφαλίζει ότι το Backend εκτελείται σε ένα "καθαρό" και ελεγχόμενο περιβάλλον, όπου οι εκδόσεις των πακέτων είναι ακριβώς αυτές που ορίστηκαν κατά τον σχεδιασμό (π.χ. Python 3.13 και οι αντίστοιχες βιβλιοθήκες).

Φορητότητα και το αρχείο requirements.txt

Ένα από τα σημαντικότερα οφέλη αυτής της προσέγγισης είναι η **επαναληψιμότητα (reproducibility)**. Μέσω της εντολής pip freeze > requirements.txt, καταγράφονται αυτόματα όλες οι εγκατεστημένες βιβλιοθήκες μαζί με τις ακριβείς εκδόσεις τους.

Το αρχείο αυτό λειτουργεί ως η «συνταγή» εγκατάστασης της εφαρμογής:

- **Ευκολία στη Μεταφορά:** Όταν ο κώδικας μεταφέρεται σε έναν νέο προγραμματιστή ή σε έναν διακομιστή παραγωγής (production server), δεν χρειάζεται η χειροκίνητη αναζήτηση των πακέτων.
- **Αυτόματη Εγκατάσταση:** Με την εντολή pip install -r requirements.txt, το εικονικό περιβάλλον αναπαράγεται πανομοιότυπα σε δευτερόλεπτα.
- **Συνέπεια:** Εγγυάται ότι η εφαρμογή θα συμπεριφέρεται με τον ίδιο ακριβώς τρόπο σε κάθε περιβάλλον (Development, Testing, Production), εξαλείφοντας το κλασικό σφάλμα «λειτουργεί στον υπολογιστή μου».

Δομή και Διαχείριση

Στη δομή των αρχείων της εργασίας, ο φάκελος του εικονικού περιβάλλοντος παραμένει απομονωμένος και δεν περιλαμβάνεται στο σύστημα ελέγχου εκδόσεων (Git), καθώς τα περιεχόμενά του είναι παράγωγα του αρχείου requirements.txt. Αυτή η μεθοδολογία ακολουθεί τις βέλτιστες πρακτικές του κλάδου, καθιστώντας το **Simply Served** μια επαγγελματική εφαρμογή, έτοιμη για ανάπτυξη (deployment) σε σύγχρονες cloud υποδομές ή containers (όπως το Docker), όπου η απομόνωση του περιβάλλοντος είναι προαπαιτούμενο για την ασφάλεια και την απόδοση.

3.7 Βάση Δεδομένων

Η βάση δεδομένων αποτελεί τη θεμελιώδη υποδομή και την «καρδιά» της εφαρμογής **Simply Served**. Είναι το στρώμα εκείνο όπου η πληροφορία μετατρέπεται από εφήμερη αλληλεπίδραση σε μόνιμη γνώση. Για την αποθήκευση των δεδομένων επιλέχθηκε η **MySQL**, ένα από τα πιο αξιόπιστα Συστήματα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων (**RDBMS**), το οποίο βασίζεται στο Σχεσιακό Μοντέλο (Relational Model). Η επιλογή αυτή διασφαλίζει ότι τα δεδομένα οργανώνονται σε αυστηρά δομημένους πίνακες, επιτρέποντας την επιβολή περιορισμών ακεραιότητας και την εκτέλεση σύνθετων ερωτημάτων με υψηλή ταχύτητα.

3.7.1 Αρχιτεκτονική και Διαχείριση Οντοτήτων

Η αρχιτεκτονική της βάσης δεδομένων του **Simply Served** έχει σχεδιαστεί με γνώμονα τη στιβαρότητα και την ακεραιότητα, ακολουθώντας τις αρχές της **κανονικοποίησης (normalization)**. Στόχος της σχεδίασης είναι η εξάλειψη του πλεονασμού των δεδομένων και η διασφάλιση ότι κάθε πληροφορία αποθηκεύεται σε μία μόνο θέση, διευκολύνοντας την ενημέρωση και τη συντήρηση του συστήματος. Η δομή της βάσης οργανώνεται σε τρεις κεντρικούς λειτουργικούς άξονες, καθένας από τους οποίους επιτελεί έναν εξειδικευμένο ρόλο στο οικοσύστημα της εφαρμογής.

1. Διαχείριση και Ταυτοποίηση Χρηστών

Ο πίνακας **Users** αποτελεί το θεμέλιο της προσωποποιημένης εμπειρίας. Σε αυτόν αποθηκεύονται τα δεδομένα ταυτοποίησης που λαμβάνει η εφαρμογή από το Google OAuth. Κάθε εγγραφή προσδιορίζεται μοναδικά από ένα user_id (Primary Key), το οποίο λειτουργεί ως ο συνδετικός κρίκος για όλες τις δραστηριότητες του χρήστη. Πέρα από το email και το ονοματεπώνυμο, η αποθήκευση του συνδέσμου της εικόνας προφίλ επιτρέπει στο Frontend (React) να παρέχει μια οικεία οπτική διεπαφή. Η οντότητα αυτή είναι κρίσιμη για την ασφάλεια, καθώς το user_id ενσωματώνεται στα **JSON Web Tokens (JWT)**, επιτρέποντας στο Backend να αναγνωρίζει ακαριαία ποιος χρήστης αιτείται πρόσβαση σε πόρους όπως η Wishlist ή το ιστορικό του.

2. Αποθετήριο Συνταγών και Σημασιολογικά Μεταδεδομένα

Ο πίνακας **Recipes** λειτουργεί ως η στατική γνωσιακή βάση του συστήματος, αποθηκεύοντας έναν εκτενή κατάλογο γαστρονομικού περιεχομένου. Κάθε εγγραφή περιλαμβάνει βασικά πεδία (τίτλο, οδηγίες, εικόνα), αλλά η πραγματική αλγοριθμική αξία βρίσκεται στα **metadata**: την κατηγορία (Category) και την περιοχή προέλευσης (Area). Αυτά τα δεδομένα δεν είναι απλές πληροφοριακές ετικέτες· αποτελούν τις διαστάσεις του χώρου χαρακτηριστικών που τροφοδοτούν το **Content-Based Filtering**. Για παράδειγμα, η κατηγοριοποίηση μιας συνταγής ως "Vegan" ή "Italian" επιτρέπει στη βάση δεδομένων να εκτελεί ταχύτατα φιλτραρίσματα, ομαδοποιώντας παρόμοιες συνταγές και προτείνοντάς τες σε χρήστες με αντίστοιχα ενδιαφέροντα.

3. Δυναμικές Συσχετίσεις και Αλληλεπιδράσεις (Many-to-Many)

Το πιο σύνθετο και δυναμικό κομμάτι της βάσης είναι οι πίνακες συσχετίσεων, οι οποίοι αποτυπώνουν τη σχέση **πολλά-προς-πολλά (Many-to-Many)** μεταξύ χρηστών και συνταγών. Αυτές οι γέφυρες δεδομένων επιτρέπουν στο σύστημα να "μαθαίνει" από τη συμπεριφορά της κοινότητας:

- **Ratings (Βαθμολογίες):** Ο πίνακας αυτός καταγράφει τη ρητή ανατροφοδότηση, συνδέοντας έναν χρήστη με μια συνταγή μέσω μιας αριθμητικής τιμής. Αυτά τα δεδομένα μετατρέπονται σε έναν **πίνακα Χρήστη-Αντικειμένου (User-Item Matrix)**, ο οποίος αποτελεί την είσοδο για τους αλγορίθμους **Collaborative Filtering** και **Matrix Factorization**.

- **Wishlist (Λίστα Αγαπημένων):** Επιτρέπει τη μόνιμη αποθήκευση προτιμήσεων. Η ύπαρξη μιας εγγραφής εδώ δηλώνει ισχυρή συνάφεια και χρησιμοποιείται για την ενίσχυση του βάρους ορισμένων χαρακτηριστικών στο προφίλ του χρήστη.
- **History (Ιστορικό):** Μέσω της καταγραφής των πρόσφατων προβολών με χρονική σήμανση (**timestamps**), η βάση δεδομένων επιτρέπει στην εφαρμογή να αναγνωρίζει το βραχυπρόθεσμο πλαίσιο (context) του χρήστη, προσαρμόζοντας τις συστάσεις στο "Dashboard" σε πραγματικό χρόνο.

Η αρχιτεκτονική αυτή διασφαλίζει ότι το **Simply Served** δεν είναι μια στατική εφαρμογή, αλλά ένα δυναμικό σύστημα που εξελίσσεται. Η χρήση **ξένων κλειδιών (Foreign Keys)** και **Index** στους πίνακες συσχετίσεων εγγυάται ότι οι αναζητήσεις και η παραγωγή συστάσεων εκτελούνται με τη μέγιστη δυνατή ταχύτητα, προσφέροντας μια απρόσκοπτη εμπειρία στον τελικό χρήστη.

3.7.2 Επικοινωνία και Επεξεργασία Δεδομένων

Η επικοινωνία μεταξύ του **Flask Backend** και της **MySQL** αποτελεί τον κρίσιμο δίαυλο μέσω του οποίου η επιχειρησιακή λογική της εφαρμογής αποκτά πρόσβαση στην πληροφορία. Η αλληλεπίδραση αυτή υλοποιείται μέσω της γλώσσας **SQL (Structured Query Language)**, η οποία επιτρέπει στο σύστημα να εκτελεί ακριβείς και ταχύτατους χειρισμούς επί των δεδομένων. Στο **Simply Served**, η SQL δεν χρησιμοποιείται μόνο για απλές λειτουργίες ανάγνωσης, αλλά αποτελεί το εργαλείο για την εκτέλεση σύνθετων υπολογισμών απευθείας στο επίπεδο της βάσης δεδομένων, μειώνοντας το υπολογιστικό φορτίο του διακομιστή εφαρμογών.

Σύνθετα Ερωτήματα και η Λογική "Search by Ingredients"

Η ευελιξία της SQL αναδεικνύεται ιδιαίτερα στη λειτουργία αναζήτησης βάσει υλικών. Σε αυτό το σενάριο, το Backend καλείται να επιλύσει ένα πρόβλημα συνδυαστικής: να εντοπίσει συνταγές των οποίων τα υλικά παρουσιάζουν τη μέγιστη δυνατή επικάλυψη με τα υλικά που διαθέτει ο χρήστης.

Αντί για μια απλή αναζήτηση κειμένου, το σύστημα εκτελεί ερωτήματα που περιλαμβάνουν συζεύξεις (**JOINs**) και ομαδοποιήσεις (**GROUP BY**), υπολογίζοντας δυναμικά δύο κρίσιμες μεταβλητές:

- **Matched Ingredients:** Τα υλικά που υπάρχουν τόσο στη συνταγή όσο και στο "ψυγείο" του χρήστη.
- **Missing Ingredients:** Τα υλικά που απαιτούνται αλλά λείπουν, επιτρέποντας στο σύστημα να ταξινομήσει τα αποτελέσματα με βάση το πόσο "κοντά" βρίσκεται ο χρήστης στην εκτέλεση μιας συνταγής.

Διασφάλιση Ακεραιότητας μέσω ACID Συναλλαγών

Η αξιοπιστία του συστήματος συστάσεων εξαρτάται άμεσα από την ποιότητα των δεδομένων. Για το λόγο αυτό, η MySQL διασφαλίζει τις ιδιότητες **ACID (Atomicity, Consistency, Isolation, Durability)** σε κάθε αλληλεπίδραση.

- **Ατομικότητα (Atomicity):** Εγγυάται ότι μια ενέργεια, όπως η υποβολή μιας βαθμολογίας και η ταυτόχρονη ενημέρωση του μέσου όρου της συνταγής, θα εκτελεστεί ως μία αδιαίρετη μονάδα. Αν οποιοδήποτε μέρος της διαδικασίας αποτύχει, ολόκληρη η συναλλαγή ακυρώνεται (**rollback**).
- **Συνέπεια (Consistency):** Διασφαλίζει ότι η βάση μεταβαίνει από μια έγκυρη κατάσταση σε μια άλλη, τηρώντας όλους τους κανόνες και τους περιορισμούς (π.χ. Foreign Keys).
- **Απομόνωση (Isolation):** Επιτρέπει σε πολλαπλούς χρήστες να βαθμολογούν ή να αναζητούν ταυτόχρονα, χωρίς οι ενέργειες του ενός να επηρεάζουν τα δεδομένα του άλλου πριν ολοκληρωθεί η συναλλαγή.

- **Διάρκεια (Durability):** Διασφαλίζει ότι μόλις μια βαθμολογία αποθηκευτεί επιτυχώς, θα παραμείνει στη βάση ακόμη και σε περίπτωση διακοπής ρεύματος ή αστοχίας του συστήματος.

Βελτιστοποίηση και Απόδοση (Performance)

Για τη διατήρηση της ταχείας εμπειρίας χρήστη (UX), η επικοινωνία Backend-Database βελτιστοποιείται μέσω της χρήσης **Indexes (Ευρετηρίων)** σε κρίσιμα πεδία, όπως το user_id και το recipe_id. Αυτό επιτρέπει στη MySQL να εντοπίζει τις σχετικές εγγραφές σε χιλιοστά του δευτερολέπτου, ακόμη και όταν ο όγκος των συνταγών αυξάνεται εκθετικά. Επιπλέον, η χρήση **Prepared Statements** προστατεύει την εφαρμογή από επιθέσεις **SQL Injection**, καθαρίζοντας αυτόματα τις εισόδους του χρήστη πριν αυτές φτάσουν στον κινητήρα της βάσης δεδομένων. Αυτή η συνέργεια ασφάλειας και ταχύτητας καθιστά το Backend του **Simply Served** έναν στιβαρό πυλώνα, ικανό να υποστηρίξει τις απαιτητικές ανάγκες ενός σύγχρονου συστήματος συστάσεων.

3.7.3 Σημασία για το User Experience

Η ποιότητα μιας διαδικτυακής εφαρμογής κρίνεται συχνά από τη διεπαφή της, όμως η πραγματική αξία της εμπειρίας του χρήστη (User Experience) πηγάζει από την ταχύτητα και την ευρωστία της βάσης δεδομένων. Στο **Simply Served**, η σταθερότητα και η υψηλή απόκριση της MySQL δεν αποτελούν απλώς τεχνικά χαρακτηριστικά, αλλά μεταφράζονται άμεσα σε μια συνεπή και απρόσκοπτη πλοήγηση. Η σχεσιακή δομή επιτρέπει στην εφαρμογή να λειτουργεί με "μνήμη", διασφαλίζοντας ότι η αλληλεπίδραση του χρήστη δεν είναι αποσπασματική αλλά αποτελεί μέρος μιας συνεχιζόμενης, προσωπικής διαδρομής.

Συνέχεια και Συγχρονισμός (Data Persistence)

Ένας από τους βασικούς πυλώνες του UX είναι η **πανταχού παρούσα πρόσβαση (Ubiquitous Access)**. Λόγω της κεντρικοποιημένης διαχείρισης στη βάση δεδομένων, το προφίλ, η Wishlist και το ιστορικό του χρήστη ανακαλούνται ακαριαία, ανεξάρτητα από τη συσκευή ή το πρόγραμμα περιήγησης που χρησιμοποιείται.

Είτε ο χρήστης αποθηκεύσει μια συνταγή από τον υπολογιστή του το πρωί, είτε αναζητήσει υλικά από το κινητό του την ώρα που βρίσκεται στην κουζίνα, η βάση δεδομένων εγγυάται ότι η πληροφορία θα είναι εκεί, συγχρονισμένη και έτοιμη προς χρήση. Αυτή η σταθερότητα μειώνει τη γνωστική κόπωση του χρήστη και ενισχύει την εμπιστοσύνη του προς την πλατφόρμα, καθώς η εφαρμογή συμπεριφέρεται ως ένας αξιόπιστος ψηφιακός βοηθός που «θυμάται» τις προτιμήσεις του.

Η Βάση Δεδομένων ως "Γνωσιακός Χάρτης"

Πέρα από την απλή αποθήκευση, η βάση δεδομένων λειτουργεί ως ένας ενεργός **γνωσιακός χάρτης**. Κάθε βαθμολογία, κάθε αναζήτηση και κάθε κλικ αποθηκεύεται με τέτοιο τρόπο ώστε να τροφοδοτεί τους αλγορίθμους συστάσεων. Αυτό σημαίνει ότι η εφαρμογή εξελίσσεται οργανικά μαζί με τον χρήστη.

Όσο περισσότερο αλληλεπιδρά ο χρήστης με το σύστημα, τόσο πιο ακριβής γίνεται η "εικόνα" του στη βάση δεδομένων. Η ταχύτητα ανάκτησης αυτών των συσχετίσεων επιτρέπει στο Dashboard να προσαρμόζεται δυναμικά, εμφανίζοντας γαστρονομικές προτάσεις που ανταποκρίνονται στις τρέχουσες ανάγκες του. Για παράδειγμα, αν η βάση ανιχνεύσει μια προτίμηση σε "Vegetarian" συνταγές μέσω των πρόσφατων Ratings, η εμπειρία του χρήστη αναβαθμίζεται αυτόματα, καθώς οι σχετικές συνταγές αποκτούν προτεραιότητα στην προβολή, κάνοντας την εύρεση γεύματος μια γρήγορη και ευχάριστη διαδικασία.

Αξιοπιστία και Ψυχολογία του Χρήστη

Τέλος, η ταχύτητα απόκρισης της βάσης δεδομένων επηρεάζει την ψυχολογία του χρήστη. Σε μια εποχή όπου οι καθυστερήσεις δευτερολέπτων οδηγούν στην εγκατάλειψη μιας εφαρμογής, η βελτιστοποιημένη επικοινωνία του Flask με τη MySQL διασφαλίζει ότι οι αναζητήσεις εκτελούνται σε πραγματικό χρόνο.

Όταν ένας χρήστης πληκτρολογεί "Tomato" και "Basil", η άμεση εμφάνιση αποτελεσμάτων δημιουργεί μια αίσθηση ελέγχου και ικανοποίησης. Η βάση δεδομένων του **Simply Served** είναι σχεδιασμένη να υποστηρίζει αυτή την ταχύτητα, διατηρώντας ταυτόχρονα την ακεραιότητα των δεδομένων. Με αυτόν τον τρόπο, η τεχνολογία παραμένει στο παρασκήνιο, επιτρέποντας στον χρήστη να επικεντρωθεί σε αυτό που πραγματικά έχει σημασία: τη δημιουργία και την απόλαυση του φαγητού.

3.8 Εργαλεία Ανάπτυξης και Βιβλιοθήκες

Η επιτυχής ολοκλήρωση μιας Full-stack εφαρμογής δεν βασίζεται μόνο στις κύριες γλώσσες προγραμματισμού, αλλά και σε ένα σύνολο υποστηρικτικών εργαλείων που βελτιστοποιούν τη διαδικασία της ανάπτυξης, του ελέγχου και της σχεδίασης. Για την υλοποίηση του Simply Served, επιλέχθηκαν σύγχρονα εργαλεία που διασφαλίζουν την ταχύτητα στην κωδικοποίηση και την ποιότητα στο τελικό αποτέλεσμα.

Ως κύριο περιβάλλον ανάπτυξης (IDE) χρησιμοποιήθηκε το **Visual Studio Code (VS Code)**, το οποίο παρείχε τις απαραίτητες επεκτάσεις για τον έλεγχο της σύνταξης (linting) και την αυτόματη συμπλήρωση κώδικα σε JavaScript και Python. Για τον έλεγχο των REST APIs και τη διασφάλιση ότι το Backend επιστρέφει τα σωστά δεδομένα πριν αυτά συνδεθούν με το Frontend, χρησιμοποιήθηκε το **Postman**. Το Postman επέτρεψε την προσομοίωση HTTP αιτημάτων και τον έλεγχο των JWT tokens, καθιστώντας την αποσφαλμάτωση (debugging) της επικοινωνίας μεταξύ των δύο μερών πολύ πιο γρήγορη.

Στο επίπεδο της διεπαφής χρήστη (UI), χρησιμοποιήθηκαν οι εξής βιβλιοθήκες:

- **Bootstrap & CSS:** Για τη δημιουργία ενός ανταποκρινόμενου (responsive) σχεδιασμού που προσαρμόζεται σε όλες τις οθόνες, από κινητά τηλέφωνα έως επιτραπέζιους υπολογιστές.
- **Lucide React:** Για την ενσωμάτωση καθαρών και μοντέρνων εικονιδίων (icons), όπως το εικονίδιο της καρδιάς για τη wishlist ή του χρήστη για το προφίλ.
- **Framer Motion:** Μια προηγμένη βιβλιοθήκη για την προσθήκη ομαλών κινήσεων και μεταβάσεων (animations), η οποία αναβαθμίζει την αίσθηση της εφαρμογής, κάνοντας την πλοήγηση πιο φυσική.

Τέλος, η διαχείριση των πακέτων και των εξαρτήσεων έγινε μέσω του **npm (Node Package Manager)** για το Frontend και του **pip** για το Backend. Η χρήση του αρχείου requirements.txt στην Python και του package.json στη React διασφαλίζει ότι η εφαρμογή μπορεί να αναπαραχθεί σε οποιοδήποτε νέο περιβάλλον ανάπτυξης με τις ακριβείς εκδόσεις των βιβλιοθηκών που χρησιμοποιήθηκαν. Αυτός ο συνδυασμός εργαλείων και βιβλιοθηκών επέτρεψε τη δημιουργία ενός επαγγελματικού και σταθερού προϊόντος, έτοιμου για χρήση από τον τελικό καταναλωτή.

Κεφάλαιο 4ο: Σχεδίαση της εφαρμογής Simply Served

Μετά την ανάλυση του θεωρητικού υποβάθρου και των τεχνολογιών που χρησιμοποιήθηκαν, το παρόν κεφάλαιο εστιάζει στη διαδικασία σχεδιασμού της εφαρμογής **Simply Served**. Ο σχεδιασμός αποτελεί ένα από τα κρισιμότερα στάδια στον κύκλο ζωής ανάπτυξης λογισμικού, καθώς καθορίζει τη λειτουργικότητα, την επεκτασιμότητα και την ευχρηστία του τελικού προϊόντος.

Η σχεδίαση ξεκινά από την κατανόηση των αναγκών του τελικού χρήστη και καταλήγει στη λεπτομερή αρχιτεκτονική δομή του συστήματος. Στόχος μας ήταν η δημιουργία μιας πλατφόρμας που να είναι διαισθητική και ταυτόχρονα ικανή να διαχειριστεί σύνθετες λειτουργίες, όπως η αναζήτηση βάσει υλικών και η παροχή εξατομικευμένων συστάσεων.

Στις ενότητες που ακολουθούν, παρουσιάζονται οι λειτουργικές απαιτήσεις της εφαρμογής μέσα από το πρίσμα των **User Stories**, αναλύεται η **αρχιτεκτονική Client-Server** που επιλέχθηκε, προσδιορίζονται οι κατηγορίες των **χρηστών** και, τέλος, περιγράφεται ο **σχεδιασμός της βάσης δεδομένων**, ο οποίος αποτελεί το θεμέλιο για την αποθήκευση και ανάκτηση της πληροφορίας. Ο λεπτομερής αυτός σχεδιασμός διασφαλίζει ότι η εφαρμογή δεν είναι απλώς ένα σύνολο κώδικα, αλλά ένα συγκροτημένο σύστημα που επιλύει πραγματικά προβλήματα της καθημερινότητας στην κουζίνα.

4.1 Λειτουργικές απαιτήσεις (User Stories)

Οι λειτουργικές απαιτήσεις περιγράφουν λεπτομερώς τις δυνατότητες του λογισμικού και τον τρόπο με τον οποίο αυτές ανταποκρίνονται στις ανάγκες των χρηστών για την εύρεση και διαχείριση συνταγών. Για την εφαρμογή **«Simply Served»**, οι απαιτήσεις αυτές περιλαμβάνουν:

Είσοδος και Ταυτοποίηση Χρηστών: Η εφαρμογή παρέχει τη δυνατότητα σύνδεσης μέσω της υπηρεσίας Google Login (OAuth 2.0). Με αυτόν τον τρόπο, οι χρήστες μπορούν να εισέρχονται στην εφαρμογή γρήγορα και με ασφάλεια, χωρίς τη δημιουργία νέου κωδικού πρόσβασης, ενώ το σύστημα δημιουργεί αυτόματα ένα προφίλ χρήστη βασισμένο στα στοιχεία του λογαριασμού τους στη Google.

Αναζήτηση Συνταγών βάσει Υλικών: Μία από τις κεντρικές λειτουργίες της εφαρμογής είναι η δυνατότητα του χρήστη να εισάγει μια λίστα με τα υλικά που διαθέτει. Το σύστημα επεξεργάζεται τα δεδομένα και επιστρέφει συνταγές, κατηγοριοποιώντας τες σε αυτές που έχουν πλήρη αντιστοιχία υλικών και σε αυτές που παρουσιάζουν ελλείψεις, ενημερώνοντας τον χρήστη για το ποια συγκεκριμένα συστατικά του λείπουν.

Δυναμικό Dashboard Συστάσεων: Οι εγγεγραμμένοι χρήστες έχουν πρόσβαση σε ένα εξατομικευμένο περιβάλλον (Dashboard). Η εφαρμογή χρησιμοποιεί αλγορίθμους συστημάτων συστάσεων για να προτείνει συνταγές που ταιριάζουν στο γευστικό προφίλ του χρήστη, λαμβάνοντας υπόψη το ιστορικό του, τις αποθηκευμένες συνταγές και τις βαθμολογίες που έχει υποβάλει.

Διαχείριση Λίστας Αγαπημένων (Wishlist): Οι χρήστες έχουν τη δυνατότητα να αποθηκεύουν συνταγές στη λίστα αγαπημένων τους πατώντας το σχετικό εικονίδιο. Η λίστα αυτή είναι προσβάσιμη ανά πάσα στιγμή, επιτρέποντας στον χρήστη να οργανώνει τα γεύματα που τον ενδιαφέρουν και να έχει άμεση πρόσβαση σε αυτά.

Αξιολόγηση και Βαθμολόγηση Συνταγών: Η εφαρμογή επιτρέπει στους χρήστες να βαθμολογούν τις συνταγές που έχουν δοκιμάσει μέσω ενός συστήματος αστέρων (Rating System). Οι βαθμολογίες αυτές αποθηκεύονται στη βάση δεδομένων και αποτελούν κρίσιμη πληροφορία για τη βελτίωση των μελλοντικών συστάσεων από τον αλγόριθμο συνεργατικού φιλτραρίσματος.

Ιστορικό Πλοήγησης και Προβολών: Το σύστημα καταγράφει αυτόματα τις συνταγές που έχει επισκεφθεί πρόσφατα ο χρήστης. Η λειτουργία αυτή διευκολύνει την επιστροφή σε περιεχόμενο που προκάλεσε το ενδιαφέρον του χρήστη κατά τη διάρκεια της πλοήγησης, ενισχύοντας την εμπειρία χρήσης.

Προβολή Λεπτομερειών Συνταγής: Για κάθε συνταγή, η εφαρμογή παρέχει μια αναλυτική σελίδα πληροφοριών. Αυτή περιλαμβάνει τη λίστα των υλικών, τον χρόνο προετοιμασίας, τον βαθμό δυσκολίας, την κατηγορία (π.χ. Vegetarian, Desserts) και αναλυτικές οδηγίες εκτέλεσης βήμα προς βήμα.

Σελίδες Πληροφόρησης και Πλοήγησης: Η εφαρμογή περιλαμβάνει ειδικές σελίδες όπως το «Home» για γενική αναζήτηση, το «Categories» για περιήγηση βάσει είδους φαγητού και το «About» για πληροφορίες σχετικά με την πλατφόρμα. Οι σελίδες αυτές διασφαλίζουν ότι ο χρήστης κατανοεί τις δυνατότητες του «Simply Served» και πλοηγείται με ευκολία.

Σήμανση Ολοκλήρωσης (Mark as Cooked): Μέσα από τη σελίδα της συνταγής, ο χρήστης μπορεί να δηλώσει ότι μαγείρεψε τη συγκεκριμένη συνταγή. Η πληροφορία αυτή χρησιμοποιείται από το σύστημα για να αναπροσαρμόζει τις συστάσεις του, δίνοντας προτεραιότητα σε παρόμοιες γεύσεις ή προτείνοντας νέες εναλλακτικές.

Διαχείριση Προφίλ και Αποσύνδεση: Οι χρήστες μπορούν να βλέπουν τα στοιχεία του προφίλ τους και να αποσυνδέονται με ασφάλεια από την εφαρμογή. Με την αποσύνδεση, η πρόσβαση στις προσωπικές πληροφορίες (Wishlist, Dashboard) κλειδώνει, διασφαλίζοντας την προστασία των δεδομένων του χρήστη.

4.2 Αρχιτεκτονική της εφαρμογής

Όπως προαναφέρθηκε, ο στόχος της εφαρμογής **«Simply Served»** είναι να αξιοποιήσει προηγμένες τεχνικές συστημάτων συστάσεων και επεξεργασίας δεδομένων, δίνοντας τη δυνατότητα στους χρήστες να ανακαλύπτουν συνταγές μαγειρικής με βάση τα διαθέσιμα υλικά τους, με τρόπο εύκολο και εξατομικευμένο.

Αρχικά, οι χρήστες εισέρχονται στην εφαρμογή μέσω της ταυτοποίησης Google Login, η οποία διασφαλίζει την ασφαλή πρόσβαση στο προσωπικό τους περιβάλλον. Σε αυτό το στάδιο, η εφαρμογή επιτρέπει στους χρήστες να αλληλεπιδράσουν με τον κεντρικό μηχανισμό αναζήτησης, εισάγοντας τα υλικά που διαθέτουν. Το σύστημα επεξεργάζεται τη λίστα των υλικών και πραγματοποιεί μια αντιστοίχιση (matching) με τη βάση δεδομένων συνταγών, κατηγοριοποιώντας τα αποτελέσματα σε συνταγές που μπορούν να εκτελεστούν άμεσα και σε αυτές που απαιτούν επιπλέον αγορές.

Στη συνέχεια, η εφαρμογή προσφέρει το δυναμικό **Dashboard**, το οποίο αποτελεί το κέντρο της εξατομικευμένης εμπειρίας. Εδώ, οι χρήστες έχουν τη δυνατότητα να δουν προτάσεις που παράγονται από αλγορίθμους συστημάτων συστάσεων (Content-based και Collaborative Filtering). Το «Simply Served» αναλύει το ιστορικό προβολών, τις αποθηκευμένες συνταγές στη Wishlist και τις βαθμολογίες που έχουν υποβάλει οι χρήστες, προκειμένου να προτείνει γεύματα που ευθυγραμμίζονται με τις προτιμήσεις τους. Αυτό επιτρέπει στους χρήστες να ανακαλύπτουν νέες γευστικές εμπειρίες χωρίς να αναλώνονται σε χρονοβόρες αναζητήσεις.

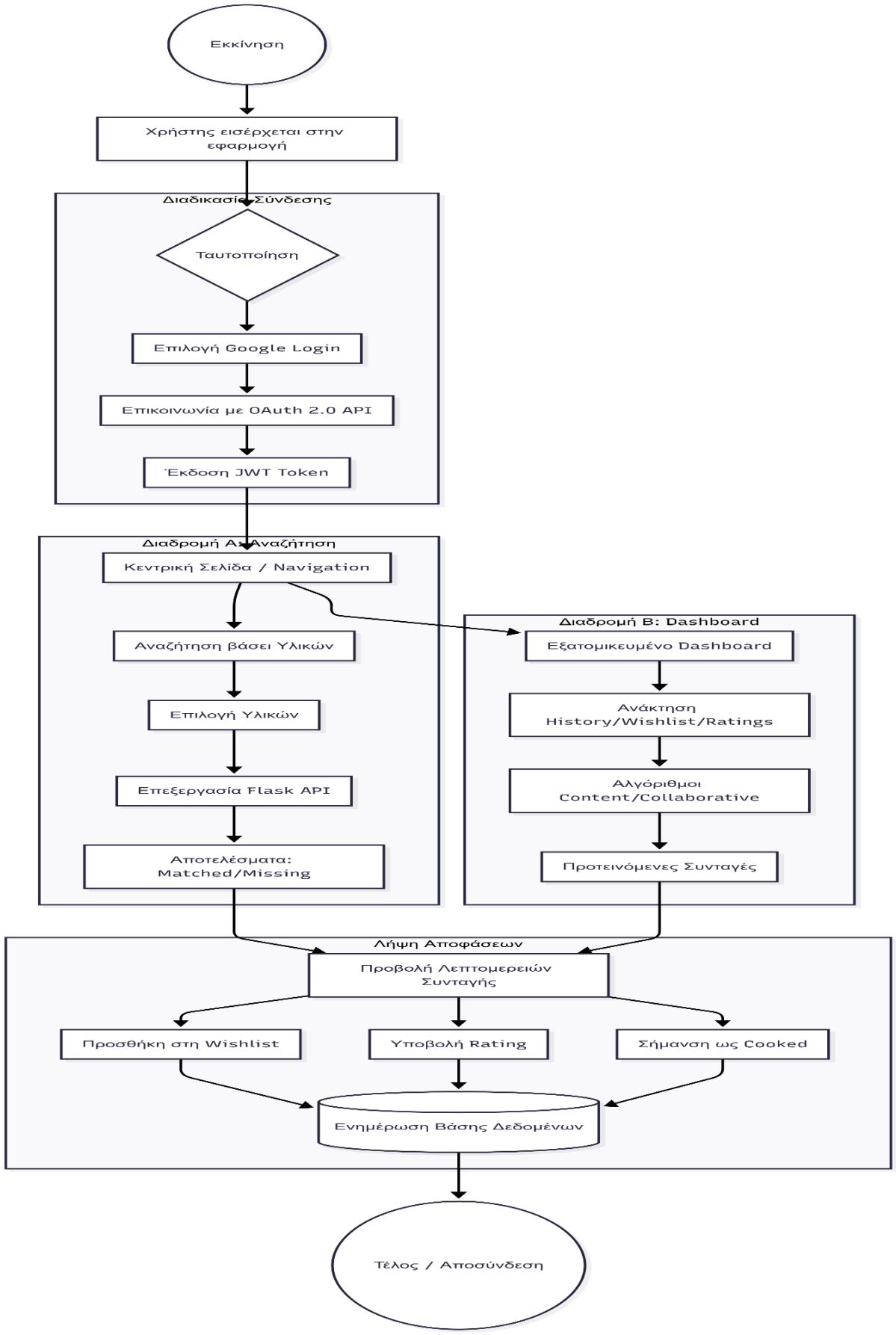
Όλες οι παραπάνω λειτουργίες είναι διαθέσιμες μέσω μιας σύγχρονης και φιλικής προς τον χρήστη γραφικής διεπαφής (GUI), η οποία επικοινωνεί αδιάλειπτα με ένα REST API. Η ταυτοποίηση των αιτημάτων πραγματοποιείται μέσω των **JSON Web Tokens (JWT)**, τα οποία εκδίδονται κατά την είσοδο του χρήστη και συνοδεύουν κάθε αλληλεπίδραση με το Backend για τη διασφάλιση των δεδομένων. Όσον αφορά την ανταλλαγή των δεδομένων, αυτά παρουσιάζονται σε μορφή **JSON**,

διασφαλίζοντας την ταχύτητα και την ελαφριά δομή της επικοινωνίας. Το Σχήμα 4.1 παρουσιάζει αναλυτικά, μέσα από ένα διάγραμμα ροής, τη σειρά των βημάτων και λειτουργιών που ακολουθεί ο χρήστης στην εφαρμογή.

Η αρχιτεκτονική του «Simply Served» ακολουθεί το μοντέλο **Client-Server** και περιλαμβάνει τρία κύρια μέρη:

- **Backend API:** Τον πυρήνα της εφαρμογής αποτελεί το Web API, το οποίο έχει αναπτυχθεί με το framework **Flask** στη γλώσσα προγραμματισμού **Python**. Το API λειτουργεί ως η γέφυρα επικοινωνίας μεταξύ της βάσης δεδομένων και του Frontend, διαχειρίζεται την αυθεντικοποίηση, την επεξεργασία των υλικών και την εκτέλεση των αλγορίθμων συστάσεων.
- **Βάση Δεδομένων (Database):** Στο Backend χρησιμοποιείται μια σχεσιακή βάση δεδομένων, η οποία διαχειρίζεται την αποθήκευση των χρηστών, των συνταγών, των βαθμολογιών (Ratings) και των λιστών επιθυμιών (Wishlist). Η βάση αλληλεπιδρά με το API για την ανάκτηση και την ενημέρωση των δεδομένων σε πραγματικό χρόνο.
- **Frontend:** Η γραφική διεπαφή αναπτύχθηκε με τη βιβλιοθήκη **React.js**, χρησιμοποιώντας **HTML, CSS, Bootstrap και JavaScript**. Η χρήση της React επιτρέπει τη δημιουργία μιας Single Page Application (SPA), όπου η πλοήγηση και η ανανέωση των δεδομένων γίνονται ακαριαία, προσφέροντας μια ομαλή εμπειρία χρήσης.

Συνολικά, η αρχιτεκτονική του «Simply Served» παρουσιάζεται σχηματικά στο Σχήμα 4.2, αποτυπώνοντας τη ροή της πληροφορίας από τον τελικό χρήστη προς το σύστημα και αντίστροφα.



Εικόνα 4.2: Διάγραμμα Ροής της Εφαρμογής Simply Served

4.3 Χρήστες της εφαρμογής

Ο προσδιορισμός των χρηστών είναι απαραίτητος για την κατανόηση των δικαιωμάτων πρόσβασης και των λειτουργιών που προσφέρει το σύστημα σε κάθε επίπεδο. Στο «Simply Served», οι χρήστες χωρίζονται σε τρεις κύριες κατηγορίες:

1. Επισκέπτης (Guest User): Πρόκειται για τον χρήστη που εισέρχεται στην εφαρμογή χωρίς να έχει πραγματοποιήσει σύνδεση (login). Ο επισκέπτης έχει περιορισμένη πρόσβαση, η οποία εστιάζει στην εξερεύνηση των δυνατοτήτων της πλατφόρμας.

- Μπορεί να περιηγηθεί στην αρχική σελίδα και στις κατηγορίες συνταγών.
- Έχει τη δυνατότητα να χρησιμοποιήσει τη μηχανή αναζήτησης βάσει υλικών για να δει αποτελέσματα σε πραγματικό χρόνο.
- **Περιορισμός:** Δεν μπορεί να αποθηκεύσει συνταγές στη Wishlist, να υποβάλει βαθμολογίες ή να έχει πρόσβαση στο εξατομικευμένο Dashboard, καθώς αυτές οι λειτουργίες απαιτούν την ύπαρξη JWT token.

2. Εγγεγραμμένος Χρήστης (Registered User): Είναι ο βασικός τύπος χρήστη της εφαρμογής, ο οποίος έχει ταυτοποιηθεί μέσω του Google OAuth 2.0. Μετά την είσοδό του, το σύστημα ανακτά το προφίλ του και του επιτρέπει την πλήρη χρήση των δυνατοτήτων της εφαρμογής.

- Διαθέτει προσωπικό **Dashboard** με συστάσεις βασισμένες στο γευστικό του προφίλ.
- Μπορεί να διαχειρίζεται τη δική του **Wishlist** και να βλέπει το ιστορικό των πρόσφατων προβολών του.
- Έχει το δικαίωμα να αλληλεπιδρά ενεργά με το περιεχόμενο, υποβάλλοντας βαθμολογίες (Ratings) και σημαίνοντας συνταγές ως "Ολοκληρωμένες" (Cooked).

3. Διαχειριστής Συστήματος (Administrator): Αν και η διεπαφή του διαχειριστή δεν είναι άμεσα ορατή στον απλό χρήστη, ο ρόλος αυτός είναι υπεύθυνος για τη συντήρηση και την ομαλή λειτουργία της εφαρμογής στο παρασκήνιο (Backend).

- Είναι υπεύθυνος για την ενημέρωση της βάσης δεδομένων με νέες συνταγές και υλικά.
- Παρακολουθεί την απόδοση των αλγορίθμων συστάσεων και προβαίνει σε ρυθμίσεις των παραμέτρων (π.χ. βάρη στο Content-based filtering).
- Διαχειρίζεται τα ζητήματα ασφαλείας και την ορθή λειτουργία των API endpoints στο Flask.

Η διαφοροποίηση αυτή διασφαλίζει ότι η εφαρμογή παραμένει ασφαλής και λειτουργική, παρέχοντας στον κάθε χρήστη ακριβώς τα εργαλεία που χρειάζεται για να βελτιώσει την εμπειρία του στην κουζίνα.

4.4 Σχεδίαση Βάσης Δεδομένων

Η βάση δεδομένων της εφαρμογής **Simply Served** υλοποιήθηκε σε περιβάλλον **MySQL (έκδοση 9.2.0)** και ακολουθεί το σχεσιακό μοντέλο για τη διασφάλιση της ακεραιότητας και της αποδοτικής οργάνωσης της πληροφορίας. Η σχεδίαση εστιάζει στην αποθήκευση ενός μεγάλου όγκου δεδομένων συνταγών και στην καταγραφή των σύνθετων αλληλεπιδράσεων των χρηστών, οι οποίες τροφοδοτούν το σύστημα συστάσεων.

Η δομή της βάσης δεδομένων αποτελείται από τους παρακάτω κύριους πίνακες και οντότητες:

1. Πίνακας Χρηστών (users)

Ο πίνακας αυτός αποτελεί το κέντρο της αυθεντικοποίησης και του προφίλ του χρήστη. Πέρα από τα βασικά στοιχεία (username, email), ο πίνακας περιλαμβάνει πεδία για την υποστήριξη του **OAuth 2.0** (provider, provider_id), επιτρέποντας τη σύνδεση μέσω τρίτων υπηρεσιών (όπως η Google). Επίσης, αποθηκεύεται το avatar του χρήστη και η ημερομηνία δημιουργίας του λογαριασμού, εξασφαλίζοντας μια προσωποποιημένη εμπειρία.

2. Πίνακας Συνταγών (meals) και Κατηγοριών (categories)

Ο πίνακας meals περιλαμβάνει όλα τα στατικά δεδομένα των συνταγών, όπως τον τίτλο, τις οδηγίες (instructions), την περιοχή προέλευσης (area), το thumbnail και το σύνδεσμο για το βίντεο οδηγιών. Ο διαχωρισμός των κατηγοριών στον πίνακα categories επιτρέπει την εύκολη οργάνωση και το φιλτράρισμα των συνταγών βάσει τύπου φαγητού (π.χ. Seafood, Dessert, Vegan).

3. Διαχείριση Υλικών (ingredients & meal_ingredients)

Για την υποστήριξη της αναζήτησης βάσει υλικών, χρησιμοποιείται μια σχέση **Πολλά-προς-Πολλά (Many-to-Many)**.

- Ο πίνακας ingredients αποθηκεύει τη λίστα με όλα τα διαθέσιμα συστατικά και τις εικόνες τους.
- Ο πίνακας σύνδεσης meal_ingredients συσχετίζει κάθε συνταγή με τα υλικά της, αποθηκεύοντας ταυτόχρονα και την απαραίτητη ποσότητα (quantity) για κάθε υλικό στη συγκεκριμένη συνταγή. Η δομή αυτή είναι κρίσιμη για τον αλγόριθμο αντιστοίχισης υλικών ("Matched" / "Missing").

4. Αλληλεπιδράσεις Χρηστών (ratings, wishlist, user_history)

Αυτοί οι πίνακες καταγράφουν τη δυναμική συμπεριφορά του χρήστη, παρέχοντας τα απαραίτητα δεδομένα για το **Collaborative Filtering**:

- **ratings**: Αποθηκεύει τη βαθμολογία (1-5) και την προαιρετική κριτική που υποβάλλει ένας χρήστης για μια συνταγή.
- **wishlist**: Καταγράφει τις αγαπημένες συνταγές του χρήστη για γρήγορη πρόσβαση.
- **user_history**: Διατηρεί το ιστορικό των προβολών με χρονοσήμανση (viewed_at), επιτρέποντας στην εφαρμογή να γνωρίζει ποιο περιεχόμενο "κατανάλωσε" πρόσφατα ο χρήστης.
- **user_cooked_meals**: Καταγράφει ποιες συνταγές υλοποιήθηκαν στην πράξη από τον χρήστη.

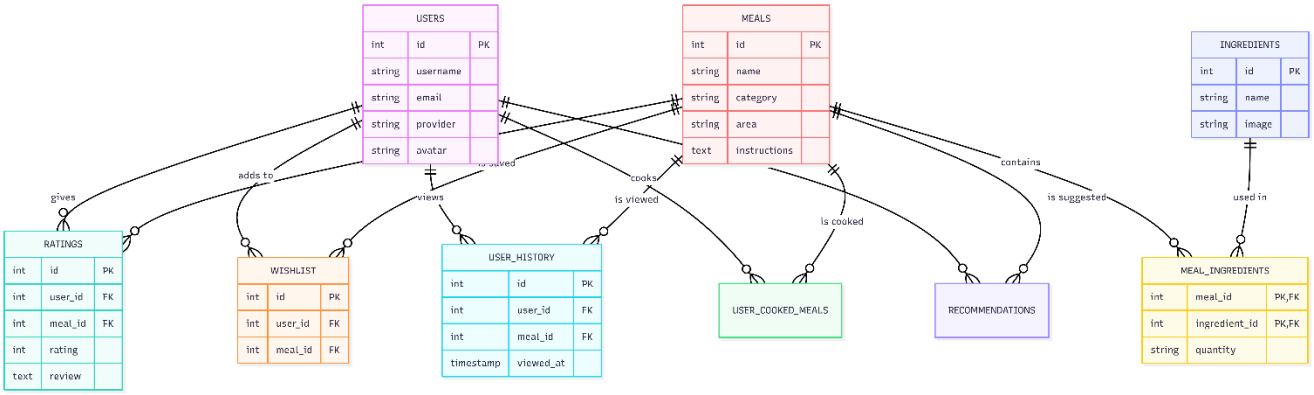
5. Σύστημα Συστάσεων (recommendations)

Ο πίνακας αυτός λειτουργεί ως αποθετήριο για τα αποτελέσματα των αλγορίθμων συστάσεων. Αποθηκεύει ένα σκορ (score) για κάθε ζεύγος χρήστη-συνταγής, το οποίο υπολογίζεται στο Backend και χρησιμοποιείται για την ταχύτατη φόρτωση των προτεινόμενων πιάτων στο Dashboard, χωρίς να απαιτείται ο επανυπολογισμός των αλγορίθμων σε κάθε αίτημα.

Σχέσεις και Περιορισμοί

Όπως προκύπτει από το σχήμα, έχουν οριστεί αυστηροί περιορισμοί ξένων κλειδιών (**Foreign Keys**) με τη λειτουργία ON DELETE CASCADE. Αυτό διασφαλίζει ότι αν διαγραφεί ένας χρήστης ή μια συνταγή, θα διαγραφούν αυτόματα και όλες οι σχετικές βαθμολογίες, ιστορικό και εγγραφές στη wishlist, διατηρώντας τη βάση δεδομένων καθαρή και απαλλαγμένη από "ορφανά" δεδομένα.

Παρακάτω φαίνεται το διάγραμμα ER:



Εικόνα 4.4: Διάγραμμα ER Βάσης Δεδομένων

Κεφάλαιο 5ο: Υλοποίηση της εφαρμογής Simply Served

Η υλοποίηση της εφαρμογής **Simply Served** αποτελεί το στάδιο όπου οι σχεδιαστικές προδιαγραφές μετατρέπονται σε λειτουργικό κώδικα. Στο παρόν κεφάλαιο παρουσιάζεται η πρακτική εφαρμογή των τεχνολογιών που αναλύθηκαν προηγουμένως, εστιάζοντας στον τρόπο με τον οποίο το Frontend και το Backend συνεργάζονται για να προσφέρουν μια ολοκληρωμένη υπηρεσία συστάσεων μαγειρικής.

Η ανάπτυξη ακολούθησε μια σπονδυλωτή προσέγγιση (modular approach), επιτρέποντας την ανεξάρτητη δοκιμή κάθε λειτουργίας. Ιδιαίτερη έμφαση δόθηκε στην ταχύτητα απόκρισης του συστήματος κατά την επεξεργασία των υλικών και στην ασφάλεια των συναλλαγών δεδομένων μέσω των JWT tokens. Στις επόμενες ενότητες, αναλύονται τα κρισιμότερα τμήματα του κώδικα, τα προβλήματα που προέκυψαν κατά την ανάπτυξη και οι λύσεις που υιοθετήθηκαν για την βελτιστοποίηση της εμπειρίας του χρήστη.

5.1 Υλοποίηση Backend

Η υλοποίηση του Backend αποτελεί τον πυρήνα της επεξεργασίας της εφαρμογής **Simply Served**. Σε αυτό το στάδιο, αναπτύχθηκε η λογική που επιτρέπει στο σύστημα να "κατανοεί" τα αιτήματα του χρήστη, να επικοινωνεί με τη βάση δεδομένων και να εκτελεί τους σύνθετους υπολογισμούς των αλγορίθμων συστάσεων. Ως κύριο εργαλείο χρησιμοποιήθηκε το framework **Flask**, λόγω της ταχύτητάς του και της άριστης συμβατότητας της γλώσσας **Python** με βιβλιοθήκες επεξεργασίας δεδομένων.

Το Backend της εφαρμογής δεν λειτουργεί απλώς ως ένα αποθετήριο δεδομένων, αλλά ως ένας ζωντανός μηχανισμός που διασφαλίζει:

- Την **ασφάλεια** των χρηστών μέσω του ελέγχου των JWT tokens.
- Την **ταχύτητα** της αναζήτησης συνταγών, ακόμα και όταν ο χρήστης εισάγει μεγάλο πλήθος υλικών.
- Την **εξατομίκευση**, μετατρέποντας τις βαθμολογίες και το ιστορικό σε μαθηματικά μοντέλα ομοιότητας.

Η ανάπτυξη του οπίσθιου τμήματος οργανώθηκε με γνώμονα τη σπονδυλωτή δομή (modularity). Αυτό σημαίνει ότι κάθε λειτουργία (π.χ. αυθεντικοποίηση, διαχείριση συνταγών, σύστημα συστάσεων) αναπτύχθηκε ως ανεξάρτητο τμήμα κώδικα, διευκολύνοντας έτσι τη συντήρηση και την πιθανή μελλοντική επέκταση της εφαρμογής. Στις υποενότητες που ακολουθούν, αναλύεται η αρχιτεκτονική των αρχείων, η δημιουργία των RESTful διαδρομών (endpoints) και η πρακτική εφαρμογή των αλγορίθμων που περιγράφηκαν στο θεωρητικό μέρος της εργασίας.

5.1.1 Δομή και Οργάνωση του Server

Η ανάπτυξη του Backend της εφαρμογής **Simply Served** ξεκίνησε με τη δημιουργία ενός ελεγχόμενου και απομονωμένου περιβάλλοντος εργασίας. Για τον σκοπό αυτό, χρησιμοποιήθηκε η τεχνολογία του **Virtual Environment (venv)** της Python. Το εικονικό αυτό περιβάλλον διασφαλίζει ότι όλες οι βιβλιοθήκες που απαιτούνται για τη λειτουργία του διακομιστή (όπως το Flask, η Flask-CORS και η MySQL-connector) παραμένουν απομονωμένες από το υπόλοιπο λειτουργικό σύστημα, αποτρέποντας προβλήματα ασυμβατότητας μεταξύ διαφορετικών εκδόσεων λογισμικού.

Η οργάνωση του κώδικα ακολούθησε μια δομή που προάγει την αναγνωσιμότητα και την εύκολη συντήρηση. Τα κύρια συστατικά του Server οργανώθηκαν ως εξής:

- **app.py:** Αποτελεί το κεντρικό σημείο εισόδου (entry point) της εφαρμογής. Εδώ πραγματοποιείται η αρχικοποίηση του Flask, η ρύθμιση των παραμέτρων ασφαλείας (CORS) και η σύνδεση των επιμέρους διαδρομών (routes) που εξυπηρετούν το Frontend.
- **requirements.txt:** Σε αυτό το αρχείο καταγράφηκαν όλες οι απαραίτητες εξαρτήσεις και οι συγκεκριμένες εκδόσεις των πακέτων που χρησιμοποιήθηκαν. Αυτό επιτρέπει την άμεση αναπαραγωγή του περιβάλλοντος του server σε οποιοδήποτε νέο μηχάνημα με μία μόνο εντολή.
- **Διαχείριση Περιβάλλοντος (.env):** Για την προστασία ευαίσθητων δεδομένων, όπως τα κλειδιά σύνδεσης με τη βάση δεδομένων και τα μυστικά κλειδιά για την έκδοση των JWT tokens, χρησιμοποιήθηκαν μεταβλητές περιβάλλοντος, διασφαλίζοντας ότι κρίσιμες πληροφορίες δεν εκτίθενται στον πηγαίο κώδικα.

Η αρχιτεκτονική αυτή επιτρέπει στο Backend να λειτουργεί αυτόνομα και να είναι "Stateless". Κάθε αίτημα που φτάνει από το Frontend αντιμετωπίζεται ως ανεξάρτητο, γεγονός που καθιστά τον Server πιο ανθεκτικό και εύκολα επεκτάσιμο. Επιπλέον, η οργάνωση σε επίπεδο φακέλων διευκόλυνε την αποσφαλμάτωση (debugging) κατά τη διάρκεια της υλοποίησης των αλγορίθμων συστάσεων, καθώς η λογική της βάσης δεδομένων ήταν σαφώς διαχωρισμένη από τη λογική της επεξεργασίας των αιτημάτων.

5.1.2 Ανάπτυξη των REST Endpoints

Η υλοποίηση των REST Endpoints στην εφαρμογή **Simply Served** πραγματοποιήθηκε μέσω του συστήματος δρομολόγησης (**Routing**) του Flask, το οποίο επιτρέπει την αντιστοίχιση συγκεκριμένων διευθύνσεων URL με συναρτήσεις της Python. Κάθε Endpoint σχεδιάστηκε με γνώμονα την αρχιτεκτονική **Stateless**, διασφαλίζοντας ότι κάθε αίτημα μεταφέρει όλες τις απαραίτητες πληροφορίες για την εκτέλεσή του, επιστρέφοντας αποκλειστικά δεδομένα σε μορφή **JSON**.

Δομή και Λειτουργικότητα Διαδρομών

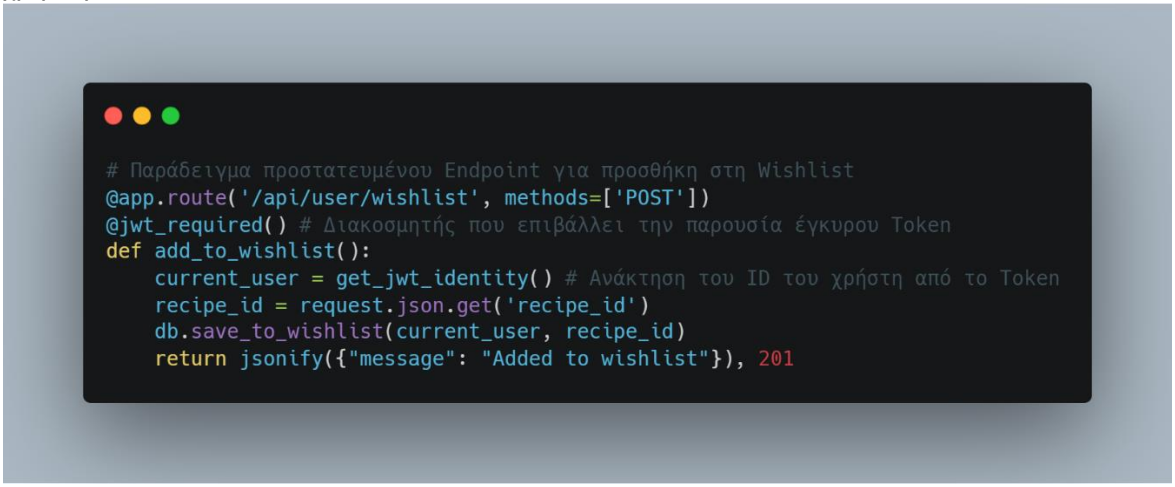
Η ανάπτυξη οργανώθηκε σε τρεις κεντρικούς άξονες, χρησιμοποιώντας τους κατάλληλους HTTP διακοσμητές για τον καθορισμό των ενεργειών:

- **Διαχείριση Συνταγών (/api/recipes):** Χρησιμοποιήθηκαν κυρίως GET requests για την ανάκτηση δεδομένων. Για παράδειγμα, η διαδρομή για τις λεπτομέρειες μιας συνταγής υλοποιήθηκε ως εξής:

```
# Ανάκτηση λεπτομερειών συγκεκριμένης συνταγής
@app.route('/api/recipes/<int:recipe_id>', methods=['GET'])
def get_recipe_details(recipe_id):
    recipe = db.get_recipe_by_id(recipe_id) # Κλήση στη βάση δεδομένων
    if recipe:
        return jsonify(recipe), 200 # Επιστροφή δεδομένων με κωδικό επιτυχίας
200 return jsonify({"error": "Recipe not found"}), 404 # Διαχείριση σφάλματος
```

- **Αυθεντικοποίηση και Ασφάλεια:** Η χρήση των **JWT (JSON Web Tokens)** αποτέλεσε τον πυλώνα της ασφάλειας. Μέσω της βιβλιοθήκης Flask-JWT-Extended, δημιουργήθηκαν προστατευμένα Endpoints που απαιτούν την επαλήθευση του ψηφιακού "διαβατηρίου" του

χρήστη.



```python
# Παράδειγμα προστατευμένου Endpoint για προσθήκη στη Wishlist
@app.route('/api/user/wishlist', methods=['POST'])
@jwt_required() # Διακοσμητής που επιβάλλει την παρουσία έγκυρου Token
def add_to_wishlist():
    current_user = get_jwt_identity() # Ανάκτηση του ID του χρήστη από το Token
    recipe_id = request.json.get('recipe_id')
    db.save_to_wishlist(current_user, recipe_id)
    return jsonify({"message": "Added to wishlist"}), 201
```

Διαχείριση Καταστάσεων και Σφαλμάτων

Ένα κρίσιμο σημείο στην ολοκλήρωση των Endpoints ήταν η ενσωμάτωση μηχανισμών **Error Handling**. Το API δεν επιστρέφει απλώς δεδομένα, αλλά και την κατάλληλη HTTP κατάσταση (**Status Code**), επιτρέποντας στο Frontend να αντιδρά αναλόγως (π.χ. εμφάνιση μηνύματος "Unauthorized" ή "Server Error").

Η χρήση των διακοσμητών @app.errorhandler επέτρεψε την τυποποίηση των απαντήσεων σε περίπτωση απρόβλεπτων σφαλμάτων, διασφαλίζοντας ότι η εφαρμογή δεν θα "καταρρεύσει" μπροστά στον χρήστη, αλλά θα επιστρέψει ένα κατανοητό μήνυμα JSON. Αυτός ο λεπτομερής σχεδιασμός μεταφέρει όλη την υπολογιστική πολυπλοκότητα στο Backend, επιτρέποντας στο Frontend να παραμένει "ελαφρύ" και επικεντρωμένο αποκλειστικά στην παρουσίαση των δεδομένων.

5.1.3 Υλοποίηση του Αλγορίθμου Αναζήτησης Υλικών

Η λειτουργία της αναζήτησης συνταγών βάσει διαθέσιμων υλικών αποτελεί την απάντηση στο πρόβλημα "τι μπορώ να μαγειρέψω σήμερα με ό,τι έχω στο ψυγείο μου;". Η υλοποίηση του αλγορίθμου στο Backend με τη χρήση της Python επέτρεψε τη δημιουργία ενός γρήγορου συστήματος φιλτραρίσματος που βασίζεται σε πράξεις συνολοθεωρίας.

Η Λογική των Συνόλων (Python Sets)

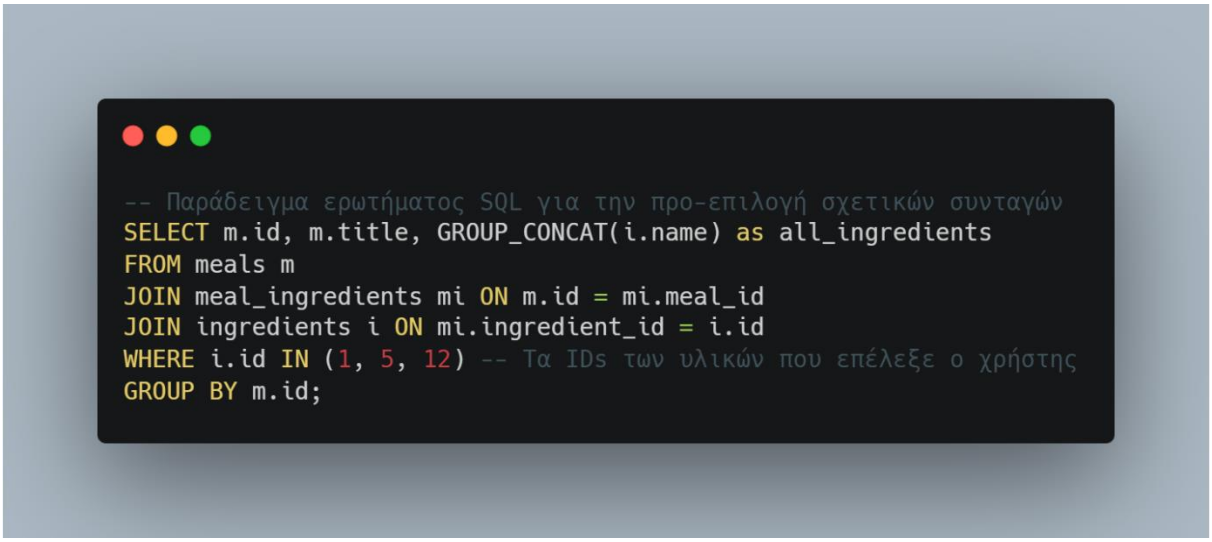
Το κλειδί για την ταχύτητα του αλγορίθμου είναι η μετατροπή των δεδομένων από απλές λίστες σε **Σύνολα (Sets)**. Τα σύνολα στην Python επιτρέπουν την εκτέλεση μαθηματικών πράξεων όπως η **τομή (intersection)** και η **διαφορά (difference)** σε γραμμικό χρόνο, καθιστώντας τον υπολογισμό εξαιρετικά αποδοτικό.

Παράδειγμα κώδικα υλοποίησης του αλγορίθμου:

```python
def calculate_recipe_match(user_ingredients, recipe_ingredients):
    """
    Υπολογίζει τα κοινά και τα ελλιπή υλικά μεταξύ χρήστη και συνταγής.
    """
    # Μετατροπή των λιστών σε σύνολα (Sets) για ταχύτατη επεξεργασία
    user_set = set(user_ingredients)
    recipe_set = set(recipe_ingredients)

    # Intersection: Υλικά που υπάρχουν και στα δύο σύνολα (Matched)
    matched = user_set.intersection(recipe_set)

    # Difference: Υλικά της συνταγής που λείπουν από τον χρήστη
(Mismissing = recipe_set.difference(user_set)

    # Υπολογισμός ποσοστού αντιστοίχισης
    match_score = (len(matched) / len(recipe_set)) * 100

    return {
        "matched": list(matched),
        "missing": list(missing),
        "score": round(match_score, 2)
    }
```

Βελτιστοποίηση σε Επίπεδο Βάσης Δεδομένων

Πριν την εφαρμογή της παραπάνω λογικής στην Python, το Backend εκτελεί ένα βελτιστοποιημένο ερώτημα **SQL** για να περιορίσει το εύρος των δεδομένων. Αντί να ανακτώνται όλες οι συνταγές, χρησιμοποιείται ένα **JOIN** ερώτημα που φέρνει μόνο τις συνταγές που περιέχουν τουλάχιστον ένα από τα υλικά του χρήστη.



```sql
-- Παράδειγμα ερωτήματος SQL για την προ-επιλογή σχετικών συνταγών
SELECT m.id, m.title, GROUP_CONCAT(i.name) as all_ingredients
FROM meals m
JOIN meal_ingredients mi ON m.id = mi.meal_id
JOIN ingredients i ON mi.ingredient_id = i.id
WHERE i.id IN (1, 5, 12) -- Τα IDs των υλικών που επέλεξε ο χρήστης
GROUP BY m.id;
```

Ταξινόμηση και Επιστροφή Αποτελεσμάτων

Μετά την επεξεργασία, ο αλγόριθμος ταξινομεί τα αποτελέσματα δυναμικά. Οι συνταγές με το υψηλότερο score (περισσότερα **Matched** και λιγότερα **Missing**) εμφανίζονται πρώτες στην απόκριση JSON:

Αυτή η προσέγγιση προσδίδει στην εφαρμογή **Simply Served** προστιθέμενη αξία, καθώς μετατρέπει μια παθητική αναζήτηση σε έναν ενεργό βοηθό λήψης αποφάσεων, ενημερώνοντας τον χρήστη ακριβώς για το τι μπορεί να μαγειρέψει και τι του λείπει.

5.1.4 Ενσωμάτωση του Συστήματος Συστάσεων

Η ενσωμάτωση του συστήματος συστάσεων (Recommendation Engine) στο Backend αποτελεί το πιο σύνθετο κομμάτι της υλοποίησης, καθώς απαιτεί τον συνδυασμό δεδομένων από τη βάση με μαθηματικούς αλγορίθμους. Στην εφαρμογή **Simply Served**, η μηχανή συστάσεων υλοποιήθηκε ως ένα ανεξάρτητο module εντός του Flask, το οποίο ενεργοποιείται κάθε φορά που ο χρήστης επισκέπτεται το Dashboard του.

Η υλοποίηση βασίστηκε σε δύο άξονες:

1. **Υλοποίηση Content-Based Filtering:** Για την πρόταση συνταγών παρόμοιων με αυτές που έχει ήδη δει ή αποθηκεύσει ο χρήστης, χρησιμοποιήθηκε η βιβλιοθήκη **Scikit-learn**. Ο κώδικας μετατρέπει τα υλικά των συνταγών σε διανύσματα χαρακτηριστικών. Στη συνέχεια, εφαρμόζεται η συνάρτηση cosine_similarity για να συγκριθεί το "γευστικό προφίλ" του χρήστη με τον κατάλογο των διαθέσιμων γευμάτων. Οι συνταγές με την υψηλότερη βαθμολογία ομοιότητας ταξινομούνται και αποστέλλονται στο Frontend.

2. **Υλοποίηση Collaborative Filtering:** Για την ανακάλυψη συνταγών που άρεσαν σε άλλους χρήστες με παρόμοια γούστα, χρησιμοποιήθηκε ο αλγόριθμος **Pearson Correlation**. Το Backend ανακτά όλες τις εγγραφές από τον πίνακα ratings, δημιουργώντας έναν προσωρινό πίνακα (matrix) χρηστών-βαθμολογιών. Ο αλγόριθμος υπολογίζει τη συσχέτιση μεταξύ του τρέχοντος χρήστη και των υπολοίπων, δίνοντας μεγαλύτερη βαρύτητα στις προτιμήσεις των "γειτόνων" με τις πιο παρόμοιες βαθμολογίες.

Από τεχνικής σκοπιάς, η διαδικασία αυτή βελτιστοποιήθηκε ώστε να μην επιβαρύνει τον Server σε κάθε κλήση. Χρησιμοποιήθηκαν τεχνικές **Caching** (προσωρινή αποθήκευση), όπου τα αποτελέσματα των συστάσεων αποθηκεύονται στον πίνακα recommendations της βάσης δεδομένων και ανανεώνονται περιοδικά ή μετά από κάθε νέα βαθμολογία του χρήστη. Αυτό εξασφαλίζει ότι το Dashboard φορτώνει ακαριαία, παρέχοντας μια ομαλή εμπειρία χρήσης, ενώ ταυτόχρονα το σύστημα "μαθαίνει" συνεχώς από τη συμπεριφορά του χρήστη.

Με την ολοκλήρωση αυτής της ενότητας, κλείνει το τεχνικό μέρος του Backend και η εφαρμογή είναι πλέον έτοιμη να τροφοδοτήσει το Frontend με προηγμένες πληροφορίες.

5.2 Υλοποίηση Frontend

Η υλοποίηση του Frontend αποτελεί το "πρόσωπο" της εφαρμογής **Simply Served** και είναι το σημείο όπου ο χρήστης αλληλεπιδρά με τη λογική που αναπτύχθηκε στο Backend. Για τη δημιουργία της διεπαφής επιλέχθηκε η βιβλιοθήκη **React.js**, η οποία επιτρέπει την κατασκευή μιας Single Page Application (SPA). Η επιλογή αυτή εξασφαλίζει ότι η πλοήγηση μεταξύ των συνταγών, η αναζήτηση υλικών και η διαχείριση του προφίλ γίνονται ακαριαία, χωρίς την ανάγκη για ανανέωση (reload) ολόκληρης της σελίδας, προσφέροντας έτσι μια εμπειρία που προσομοιάζει σε desktop εφαρμογή.

Ο βασικός στόχος κατά την ανάπτυξη του Frontend ήταν η δημιουργία ενός καθαρού, διαισθητικού και πλήρως ανταποκρινόμενου (responsive) περιβάλλοντος. Η React.js διευκόλυνε αυτή τη διαδικασία μέσω της **σπονδυλωτής αρχιτεκτονικής** της, όπου κάθε στοιχείο της οθόνης (όπως η κάρτα μιας συνταγής, η μπάρα αναζήτησης ή το μενού πλοήγησης) αντιμετωπίζεται ως ένα ανεξάρτητο και επαναχρησιμοποιήσιμο δομικό στοιχείο (Component).

Στις υποενότητες που ακολουθούν, αναλύεται η εσωτερική οργάνωση των αρχείων του Frontend, ο τρόπος με τον οποίο η React επικοινωνεί με το Flask API για την ανταλλαγή δεδομένων, καθώς και η διαχείριση της κατάστασης (State Management) της εφαρμογής. Ιδιαίτερη έμφαση δίνεται στην ενσωμάτωση του Google Login και στη διατήρηση της συνεδρίας του χρήστη, διασφαλίζοντας ότι η εξατομικευμένη πληροφορία παραμένει προσβάσιμη και ασφαλής σε κάθε βήμα της πλοήγησης.

5.2.1 Δομή των Components

Η αρχιτεκτονική της εφαρμογής βασίστηκε στην ιεραρχική οργάνωση των Components. Αντί για τη συγγραφή μεγάλων αρχείων κώδικα, η διεπαφή χωρίστηκε σε μικρότερα δομικά στοιχεία, τα οποία "συναρμολογούνται" για να σχηματίσουν τις τελικές σελίδες. Αυτό επιτρέπει τη συνεπή εμφάνιση σε ολόκληρη την εφαρμογή (π.χ. το ίδιο στυλ κάρτας συνταγής παντού) και την ταχύτερη αποσφαλμάτωση.

Τα Components της εφαρμογής κατηγοριοποιήθηκαν σε τρεις κύριες ομάδες:

1. **Layout Components (Στοιχεία Διάταξης):** Πρόκειται για τα στοιχεία που παραμένουν σταθερά κατά την πλοήγηση, εξασφαλίζοντας τη δομή της εφαρμογής.
    - **Navbar.jsx:** Η κεντρική μπάρα πλοήγησης που περιλαμβάνει το λογότυπο, τους συνδέσμους προς τις βασικές σελίδες και το κουμπί σύνδεσης/προφίλ.
    - **Footer.jsx:** Το κάτω μέρος της σελίδας με πληροφορίες πνευματικών δικαιωμάτων και χρήσιμους συνδέσμους.
2. **Functional Components (Λειτουργικά Στοιχεία):** Αυτά αποτελούν τα βασικά "τούβλα" του περιεχομένου που επικοινωνούν με τα δεδομένα.

- **RecipeCard.jsx:** Ένα από τα πιο σημαντικά components, το οποίο οπτικοποιεί τις πληροφορίες μιας συνταγής (τίτλος, εικόνα, κατηγορία) και περιλαμβάνει διαδραστικά στοιχεία όπως το κουμπί "Save to Wishlist".
- **IngredientSelector.jsx:** Ένα εξειδικευμένο component για την επιλογή υλικών, το οποίο διαχειρίζεται τη λίστα των επιλεγμένων στοιχείων πριν την αποστολή τους στο Backend.
- **SearchBar.jsx:** Η μπάρα αναζήτησης για γρήγορο φιλτράρισμα συνταγών βάσει κειμένου.

3. **Page Components (Σελίδες):** Αποτελούν τα ανώτερα components που φιλοξενούν όλα τα υπόλοιπα και αντιστοιχούν στις διαδρομές (routes) της εφαρμογής.
   - **Home.jsx:** Η αρχική σελίδα που συνδυάζει την αναζήτηση και τις γενικές προτάσεις.
   - **Dashboard.jsx:** Η εξατομικευμένη σελίδα του χρήστη που εμφανίζει τις συστάσεις και το ιστορικό.
   - **RecipeDetails.jsx:** Η αναλυτική προβολή μιας συνταγής με τα συστατικά και τις οδηγίες.

Η χρήση των **Props** επέτρεψε τη μεταφορά δεδομένων από τα γονικά components στα παιδιά (π.χ. η σελίδα Dashboard περνάει τα δεδομένα μιας συνταγής στο RecipeCard), ενώ το **State Management** εξασφάλισε ότι το UI ενημερώνεται αυτόματα κάθε φορά που αλλάζουν τα δεδομένα (π.χ. όταν ένας χρήστης κάνει "like" σε μια συνταγή). Αυτή η σπονδυλωτή δομή καθιστά το **Simply Served** μια εξαιρετικά ευέλικτη εφαρμογή, έτοιμη να υποδεχτεί νέες λειτουργίες με ελάχιστες αλλαγές στον υπάρχοντα κώδικα.

5.2.2 Διαχείριση Καθολικής Κατάστασης (Context API)

Υλοποίηση του WishlistContext

Η τεχνική υλοποίηση βασίζεται στη δημιουργία ενός **Provider**, ο οποίος περιβάλλει το δέντρο των components, και στη χρήση του hook **useContext** για την άντληση των δεδομένων. Παρακάτω παρουσιάζεται ένα ενδεικτικό τμήμα του κώδικα για τη διαχείριση των αγαπημένων συνταγών:

```jsx
// WishlistContext.js
import React, { createContext, useState, useEffect } from 'react';

export const WishlistContext = createContext();

export const WishlistProvider = ({ children }) => {
    const [wishlist, setWishlist] = useState([]);

    // Συγχρονισμός με το Backend κατά την εκκίνηση
    useEffect(() => {
        const fetchWishlist = async () => {
            const response = await fetch('/api/user/wishlist', {
                headers: { 'Authorization': `Bearer ${localStorage.getItem('token')}` }
            });
            const data = await response.json();
            setWishlist(data); // Ενημέρωση της καθολικής κατάστασης
        };
        fetchWishlist();
    }, []);

    const toggleWishlist = (recipeId) => {
        setWishlist(prev =>
            prev.includes(recipeId)
                ? prev.filter(id => id !== recipeId) // Αφαίρεση
                : [...prev, recipeId]                // Προσθήκη
        );
    };

    return (
        <WishlistContext.Provider value={{ wishlist, toggleWishlist }}>
            {children}
        </WishlistContext.Provider>
    );
};
```

Κατανάλωση της Κατάστασης από τα Components

Χάρη στον παραπάνω μηχανισμό, ένα component όπως το HeartButton.jsx μπορεί να αλληλεπιδράσει με τη Wishlist χωρίς να δέχεται props από τους γονείς του, χρησιμοποιώντας απλώς το hook **useContext**:

```jsx
// HeartButton.jsx
import { useContext } from 'react';
import { WishlistContext } from './WishlistContext';

const HeartButton = ({ recipeId }) => {
    const { wishlist, toggleWishlist } =
useContext(WishlistContext);
    const isFavorite = wishlist.includes(recipeId);

    return (
        <button onClick={() => toggleWishlist(recipeId)}>
            {isFavorite ? '❤️' : '🤍'}
        </button>
    );
};
```

Η χρήση του Context API απλοποίησε σημαντικά τη δομή του κώδικα. Αντί για πολύπλοκες αλυσίδες μεταφοράς δεδομένων, τα components απλώς "καταναλώνουν" την πληροφορία που χρειάζονται. Επιπλέον, ο συνδυασμός του Context με το **useEffect** επέτρεψε τον αυτόματο συγχρονισμό με το Backend, διασφαλίζοντας ότι η διεπαφή αντικατοπτρίζει πάντα την τρέχουσα κατάσταση της βάσης δεδομένων.

5.2.3 Σύνδεση με το API και Διαχείριση Αιτημάτων

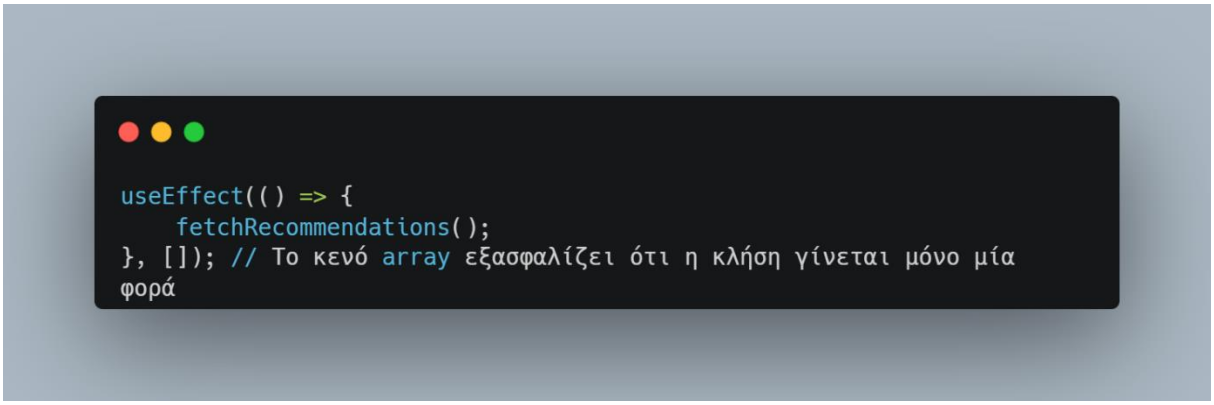
Τεχνική Υλοποίηση Ασύγχρονων Αιτημάτων

Για την επικοινωνία με το Backend, αναπτύχθηκαν συναρτήσεις που χρησιμοποιούν το πρότυπο **Async/Await**, διασφαλίζοντας την αναγνωσιμότητα του κώδικα και την ορθή διαχείριση των υποσχέσεων (Promises). Παρακάτω παρουσιάζεται η υλοποίηση ενός αιτήματος για την ανάκτηση των προσωποποιημένων συστάσεων στο Dashboard:

```
// Παράδειγμα ασύγχρονης κλήσης στο API με διαχείριση JWT και States
const fetchRecommendations = async () => {
    setLoading(true); // Ενεργοποίηση κατάστασης φόρτωσης (Loading: True)
    setError(null);

    try {
        const response = await fetch('http://localhost:5000/api/user/recommendations',
{           method: 'GET',
            headers: {
                'Content-Type': 'application/json',
                // Αυτόματη επισύναψη του JWT Token για ταυτοποίηση
                'Authorization': `Bearer ${localStorage.getItem('token')}`
            }
        });

        if (!response.ok) {
            throw new Error('Αποτυχία ανάκτησης δεδομένων');
        }

        const data = await response.json();
        setRecommendations(data); // Κατάσταση επιτυχίας (Success)
    } catch (err) {
        setError(err.message); // Κατάσταση σφάλματος (Error)
    } finally {
        setLoading(false); // Τερματισμός φόρτωσης
    }
};
```

Αυτόματη Εκτέλεση με το hook useEffect

Για να διασφαλιστεί ότι τα δεδομένα φορτώνονται αυτόματα με την είσοδο του χρήστη στη σελίδα, η παραπάνω συνάρτηση ενσωματώνεται στο hook **useEffect**. Αυτό επιτρέπει στο component να εκτελέσει το αίτημα μία φορά κατά το "mount" (αρχική εμφάνιση) του στοιχείου:



```
useEffect(() => {
    fetchRecommendations();
}, []); // Το κενό array εξασφαλίζει ότι η κλήση γίνεται μόνο μία
φορά
```

Η αμφίδρομη αυτή επικοινωνία επιτρέπει στο **Simply Served** να λειτουργεί ως ένα ενοποιημένο σύστημα, όπου κάθε ενέργεια στο UI (Frontend) μεταφράζεται σε άμεση ενημέρωση ή ανάκτηση από τη βάση δεδομένων (Backend). Η διαχείριση των καταστάσεων φόρτωσης εξασφαλίζει ότι ο χρήστης λαμβάνει οπτική επιβεβαίωση (όπως ένας spinner), βελτιώνοντας την αίσθηση στιβαρότητας της εφαρμογής.

5.2.4 Υλοποίηση της Αυθεντικοποίησης στο Frontend

Η υλοποίηση της αυθεντικοποίησης στο Frontend βασίστηκε στο πρωτόκολλο **OAuth 2.0** μέσω της υπηρεσίας **Google Login**. Επιλέχθηκε αυτή η μέθοδος καθώς προσφέρει υψηλό επίπεδο ασφάλειας και εξαλείφει την ανάγκη για τον χρήστη να θυμάται έναν επιπλέον κωδικό πρόσβασης, αυξάνοντας έτσι τις πιθανότητες αλληλεπίδρασης με την εφαρμογή.

Η ροή της αυθεντικοποίησης στο Frontend ακολουθεί τα εξής βήματα:

1. **Ενεργοποίηση Σύνδεσης:** Ο χρήστης αλληλεπιδρά με το component του Google Login. Χρησιμοποιήθηκε η βιβλιοθήκη @react-oauth/google, η οποία παρέχει μια έτοιμη και ασφαλή διεπαφή για την επικοινωνία με τους servers της Google.

2. **Λήψη Credential Token:** Μετά την έγκριση του χρήστη, η Google επιστρέφει ένα προσωρινό ID Token στο Frontend. Αυτό το token περιέχει κρυπτογραφημένες πληροφορίες όπως το email και το όνομα του χρήστη.

3. **Επαλήθευση από το Backend:** Το Frontend στέλνει αμέσως αυτό το token στο Flask API. Ο server επαληθεύει την εγκυρότητα του token και ελέγχει αν ο χρήστης υπάρχει ήδη στη βάση δεδομένων (πίνακας users). Αν δεν υπάρχει, δημιουργεί αυτόματα μια νέα εγγραφή.

4. **Έκδοση JWT και Αποθήκευση:** Μόλις το Backend επιβεβαιώσει την ταυτότητα, εκδίδει ένα δικό του **JSON Web Token (JWT)**. Το Frontend λαμβάνει αυτό το token και το αποθηκεύει στο **localStorage**. Η αποθήκευση αυτή είναι κρίσιμη, καθώς επιτρέπει στον χρήστη να παραμένει συνδεδεμένος ακόμα και αν ανανεώσει τη σελίδα (persistence).

Η διαχείριση της κατάστασης σύνδεσης ενσωματώθηκε στο AuthContext. Μέσω αυτού, ολόκληρη η εφαρμογή γνωρίζει σε πραγματικό χρόνο αν ο χρήστης είναι αυθεντικοποιημένος. Αν το token λήξει ή αν ο χρήστης επιλέξει "Logout", το Frontend διαγράφει το JWT από το localStorage και επαναφέρει την κατάσταση του AuthContext στις αρχικές τιμές, ανακατευθύνοντας αυτόματα τον χρήστη στην αρχική σελίδα.

Αυτή η υλοποίηση διασφαλίζει ότι όλες οι ευαίσθητες λειτουργίες, όπως η πρόσβαση στο Dashboard και η τροποποίηση της Wishlist, προστατεύονται από μη εξουσιοδοτημένη πρόσβαση, προσφέροντας ταυτόχρονα μια απρόσκοπτη εμπειρία σύνδεσης.

5.3 Διεπαφή Χρήστη και Εμπειρία (UI/UX)

Η επιτυχία μιας εφαρμογής δεν εξαρτάται μόνο από την ορθότητα των αλγορίθμων της, αλλά σε μεγάλο βαθμό από τον τρόπο με τον οποίο ο χρήστης αλληλεπιδρά με αυτήν. Η ενότητα **5.3** εστιάζει στην υλοποίηση του οπτικού μέρους και της συνολικής εμπειρίας χρήστη (UI/UX) του **Simply Served**. Στόχος της σχεδίασης ήταν η δημιουργία ενός περιβάλλοντος που να είναι ταυτόχρονα καλαίσθητο, λειτουργικό και εύκολα προσβάσιμο από κάθε είδους συσκευή.

Η εμπειρία χρήστη (User Experience) σχεδιάστηκε με γνώμονα την απλότητα (simplicity). Καθώς η εφαρμογή διαχειρίζεται μεγάλο όγκο πληροφορίας —υλικά, συνταγές, βαθμολογίες και συστάσεις— δόθηκε ιδιαίτερη έμφαση στην **οπτική ιεραρχία**, ώστε ο χρήστης να μπορεί να εντοπίζει άμεσα αυτό που ψάχνει χωρίς να αισθάνεται "πελαγωμένος". Από την άλλη πλευρά, η διεπαφή χρήστη (User Interface) βασίστηκε σε σύγχρονα πρότυπα σχεδίασης, χρησιμοποιώντας καθαρές γραμμές, επαρκή λευκό χώρο και έντονες οπτικές ενδείξεις (όπως τα χρωματιστά tags για τα υλικά που λείπουν).

Στις υποενότητες που ακολουθούν, αναλύεται η χρήση του **Bootstrap** για την επίτευξη ενός πλήρως ανταποκρινόμενου σχεδιασμού (Responsive Design), καθώς και η ενσωμάτωση δυναμικών στοιχείων και **animations** που καθιστούν την πλοήγηση πιο φυσική και διαδραστική. Η συνδυαστική χρήση

αυτών των εργαλείων επέτρεψε τη μετατροπή μιας σύνθετης βάσης δεδομένων συνταγών σε μια φιλική εφαρμογή που εμπνέει τον χρήστη να μαγειρέψει.

5.3.1 Σχεδιασμός με Bootstrap και Responsive Design

Η ανάγκη για μια εφαρμογή που μπορεί να χρησιμοποιηθεί εξίσου εύκολα στην επιφάνεια εργασίας ενός υπολογιστή αλλά και στην οθόνη ενός smartphone μέσα στην κουζίνα, κατέστησε τον **ανταποκρινόμενο σχεδιασμό (Responsive Design)** κορυφαία προτεραιότητα. Για την υλοποίησή του, χρησιμοποιήθηκε το framework **Bootstrap 5**, το οποίο προσφέρει ένα ισχυρό σύστημα πλέγματος (Grid System) βασισμένο σε 12 στήλες.

Η στρατηγική σχεδίασης βασίστηκε στις εξής αρχές:

- **Σύστημα Πλέγματος και Flexbox:** Χρησιμοποιήθηκαν οι κλάσεις του Bootstrap (όπως container, row και col) για τη δυναμική διευθέτηση των στοιχείων. Για παράδειγμα, στη σελίδα των αποτελεσμάτων, οι κάρτες των συνταγών διατάσσονται σε τέσσερις στήλες στις μεγάλες οθόνες, δύο στις μεσαίες και μία στις μικρές (mobile), διασφαλίζοντας ότι το περιεχόμενο παραμένει ευανάγνωστο χωρίς την ανάγκη οριζόντιας κύλισης.
- **Mobile-First Προσέγγιση:** Η σχεδίαση ξεκίνησε λαμβάνοντας υπόψη τους περιορισμούς των κινητών συσκευών. Χρησιμοποιήθηκαν **Breakpoints** (σημεία διακοπής) για την προσαρμογή των γραμματοσειρών, των περιθωρίων (margins) και των αποστάσεων (padding), ώστε τα κουμπιά και τα μενού επιλογής υλικών να είναι εύκολα προσβάσιμα με την αφή.
- **Προσαρμοσμένα CSS Components:** Παρόλο που το Bootstrap παρείχε τη δομή, εφαρμόστηκε επιπλέον κώδικας CSS για να αποδοθεί το μοναδικό στυλ του Simply Served. Αυτό περιλαμβάνει τη χρήση **Glassmorphism** (εφέ ημιδιαφανούς γυαλιού) στις κάρτες και τα modals, καθώς και μια παλέτα χρωμάτων που εμπνέει φρεσκάδα και καθαριότητα, ταιριάζοντας με το θέμα της μαγειρικής.
- **Δυναμικά UI Elements:** Η χρήση στοιχείων όπως το "Offcanvas" για το μενού πλοήγησης στα κινητά και τα "Modals" για την προβολή των λεπτομερειών της συνταγής, επέτρεψε την εξοικονόμηση χώρου στην οθόνη. Επιπλέον, τα **Badges** του Bootstrap χρησιμοποιήθηκαν αποτελεσματικά για να διαχωρίζουν οπτικά τα "Matched" από τα "Missing" υλικά, παρέχοντας άμεση πληροφορία στον χρήστη με μια ματιά.

Ο συνδυασμός αυτών των τεχνικών διασφαλίζει ότι το **Simply Served** προσφέρει μια συνεπή και επαγγελματική εμπειρία χρήσης, ανεξάρτητα από την τεχνολογική πλατφόρμα που χρησιμοποιεί ο τελικός καταναλωτής, ενισχύοντας τη χρηστικότητα (Usability) της εφαρμογής σε πραγματικές συνθήκες.

5.3.2 Animations και Διαδραστικότητα

Η διαδραστικότητα αποτελεί το τελικό στάδιο που μετατρέπει μια στατική ιστοσελίδα σε μια σύγχρονη και ελκυστική εφαρμογή. Στο **Simply Served**, η έμφαση δόθηκε στη δημιουργία ομαλών μεταβάσεων που μειώνουν το γνωστικό φορτίο του χρήστη και κάνουν την πλοήγηση να φαίνεται πιο φυσική. Για την επίτευξη αυτού του στόχου, χρησιμοποιήθηκε η βιβλιοθήκη **Framer Motion** σε συνδυασμό με τις δυνατότητες της **React**.

Η υλοποίηση της διαδραστικότητας επικεντρώθηκε στους εξής τομείς:

- **Μεταβάσεις Σελίδων και Στοιχείων:** Χρησιμοποιήθηκαν εφέ τύπου "Fade-in" και "Slide-up" κατά τη φόρτωση των αποτελεσμάτων αναζήτησης και των συστάσεων στο Dashboard. Αυτό βοηθά τον χρήστη να αντιληφθεί την ανανέωση του περιεχομένου χωρίς απότομες αλλαγές που μπορεί να προκαλέσουν σύγχυση.

- **Micro-interactions (Μικρο-αλληλεπιδράσεις):** Πρόκειται για μικρές οπτικές λεπτομέρειες που αντιδρούν στις κινήσεις του χρήστη. Για παράδειγμα:
    - **Hover effects:** Οι κάρτες των συνταγών μεγεθύνονται ελαφρώς (scale) όταν ο χρήστης περνάει το ποντίκι από πάνω τους, υποδεικνύοντας ότι το στοιχείο είναι επιλέξιμο.
    - **Button Feedback:** Τα κουμπιά "Like" και "Save" διαθέτουν animations που επιβεβαιώνουν την ενέργεια (π.χ. το εικονίδιο της καρδιάς που "χτυπά" ή αλλάζει χρώμα ομαλά), προσφέροντας άμεση ικανοποίηση στον χρήστη.
- **Δυναμική Διαχείριση Καταστάσεων (Feedback Loops):** Κατά τη διάρκεια της αναζήτησης υλικών, χρησιμοποιήθηκαν "Loading Skeletons" και spinners. Αυτά τα στοιχεία ενημερώνουν τον χρήστη ότι το σύστημα επεξεργάζεται το αίτημά του, μειώνοντας την αίσθηση της αναμονής κατά τη διάρκεια των κλήσεων στο Flask API.
- **Διαδραστικά Modals και Tooltips:** Η χρήση της Framer Motion επέτρεψε στα παράθυρα των λεπτομερειών (Modals) να εμφανίζονται με ομαλό "spring" εφέ, κάνοντας την εμπειρία να μοιάζει περισσότερο με εφαρμογή κινητού τηλεφώνου παρά με παραδοσιακό ιστότοπο.

Η ενσωμάτωση αυτών των στοιχείων διασφαλίζει ότι το **Simply Served** δεν είναι απλώς ένα εργαλείο αναζήτησης, αλλά μια πλατφόρμα που προσφέρει μια ευχάριστη και "παιχνιδιάρικη" (gamified) εμπειρία, ενθαρρύνοντας τον χρήστη να εξερευνήσει περισσότερες συνταγές και να αλληλεπιδράσει συχνότερα με το σύστημα συστάσεων.

Κεφάλαιο 6ο: Παρουσίαση της Εφαρμογής Simply Served

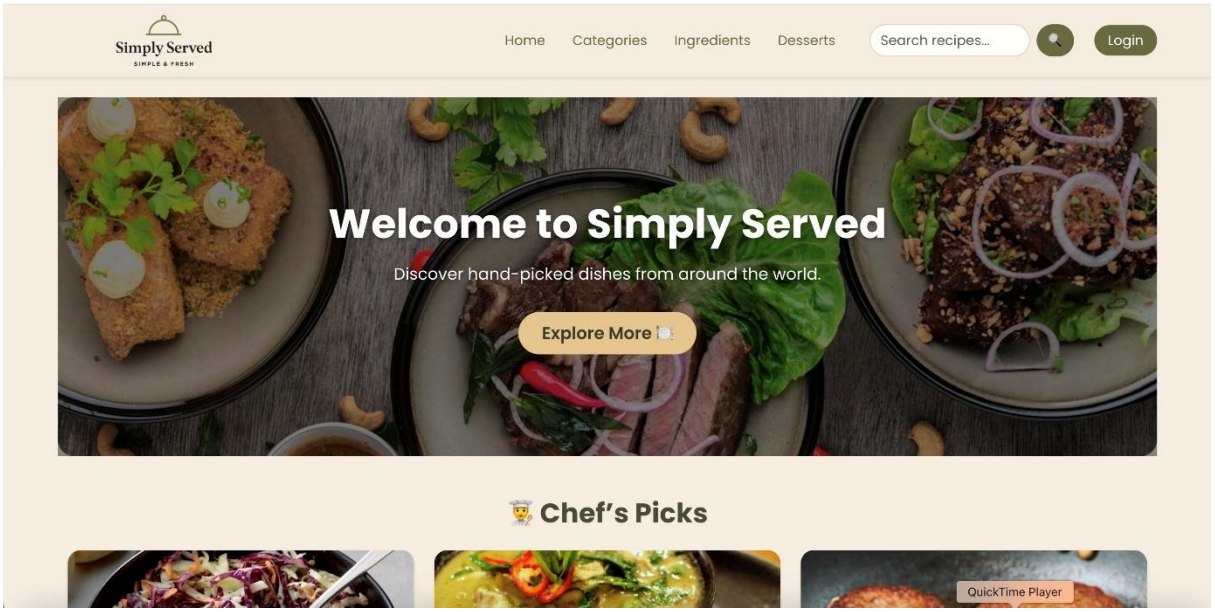
Στο παρόν κεφάλαιο γίνεται η οπτική παρουσίαση της εφαρμογής **Simply Served**, μέσω αντιπροσωπευτικών στιγμιοτύπων (screenshots) από το περιβάλλον διεπαφής χρήστη. Η παρουσίαση ακολουθεί τη λογική ροή πλοήγησης ενός χρήστη, αναδεικνύοντας τις βασικές λειτουργίες που υλοποιήθηκαν.

6.1 Αρχική Σελίδα και Πρώτη Επαφή (Landing Page)

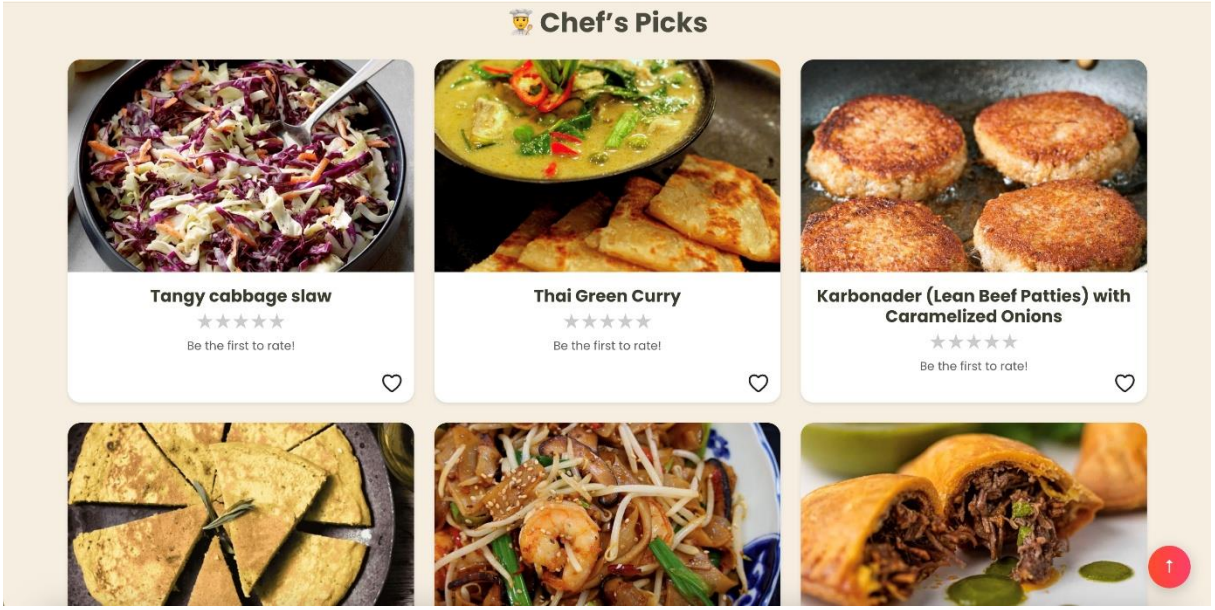
Η αρχική σελίδα του **Simply Served** αποτελεί το σημείο εισόδου και τον καθοριστικό παράγοντα για τη διαμόρφωση της πρώτης εντύπωσης του χρήστη. Ο σχεδιασμός της επικεντρώνεται στην παροχή μιας «καθαρής» και μινιμαλιστικής αισθητικής, η οποία αποσκοπεί στην εξάλειψη του περιττού οπτικού θορύβου και στην ανάδειξη του κεντρικού οράματος της εφαρμογής: την εύκολη και γρήγορη ανακάλυψη γαστρονομικών επιλογών από ένα παγκόσμιο αποθετήριο συνταγών. Η χρήση θερμών χρωμάτων και υψηλής ανάλυσης φωτογραφιών φαγητού δημιουργεί ένα οικείο περιβάλλον που προδιαθέτει θετικά τον επισκέπτη για εξερεύνηση.

- **Κεντρικό Hero Section και Πλοήγηση:** Στο πάνω μέρος της σελίδας δεσπόζει ένα εντυπωσιακό **Hero Banner** (Εικόνα 1), το οποίο λειτουργεί ως το κύριο σημείο εστίασης. Το συνοδευτικό κείμενο "Welcome to Simply Served" προσδιορίζει άμεσα την ταυτότητα της υπηρεσίας, ενώ το κουμπί "Explore More" λειτουργεί ως ένα ξεκάθαρο **Call-to-Action (CTA)**, κατευθύνοντας τον χρήστη προς τη μηχανή αναζήτησης και τις κατηγορίες. Παράλληλα, η σταθερή μπάρα πλοήγησης (Navbar) παρέχει άμεση πρόσβαση στις ενότητες των υλικών, των γλυκών και των κατηγοριών, επιτρέποντας στον χρήστη να προσανατολιστεί ακαριαία στο περιβάλλον της εφαρμογής.
- **Δυναμική Προβολή Περιεχομένου (Chef's Picks):** Συνεχίζοντας την πλοήγηση προς τα κάτω, ο χρήστης συναντά την ενότητα **"Chef's Picks"** (Εικόνα 2), η οποία παρουσιάζει επιλεγμένες συνταγές σε μορφή καρτών (**Cards**). Κάθε κάρτα αποτελεί ένα αυτόνομο functional component που ενσωματώνει σημαντικές πληροφορίες, όπως τον τίτλο της συνταγής, μια ελκυστική φωτογραφία και ένα διαδραστικό σύστημα βαθμολόγησης με αστέρια, το οποίο υποδηλώνει την κοινωνική αποδοχή του πιάτου. Η χρήση σκιών και hover effects στις κάρτες ενισχύει την αίσθηση της διαδραστικότητας, ενώ το εικονίδιο της καρδιάς στην κάτω δεξιά γωνία επιτρέπει τη γρήγορη προσθήκη στη λίστα αγαπημένων (Wishlist), αναδεικνύοντας τη σπονδυλωτή αρχιτεκτονική του Frontend.

Η διάταξη της αρχικής σελίδας ακολουθεί τις αρχές του **Responsive Design**, διασφαλίζοντας ότι η οπτική ιεραρχία παραμένει σταθερή και λειτουργική τόσο σε μεγάλες οθόνες υπολογιστών όσο και σε κινητές συσκευές. Αυτή η προσέγγιση επιτυγχάνει μια ισορροπία μεταξύ πληροφορίας και αισθητικής, μετατρέποντας την απλή περιήγηση σε μια ευχάριστη εμπειρία ανακάλυψης.



Εικόνα 1



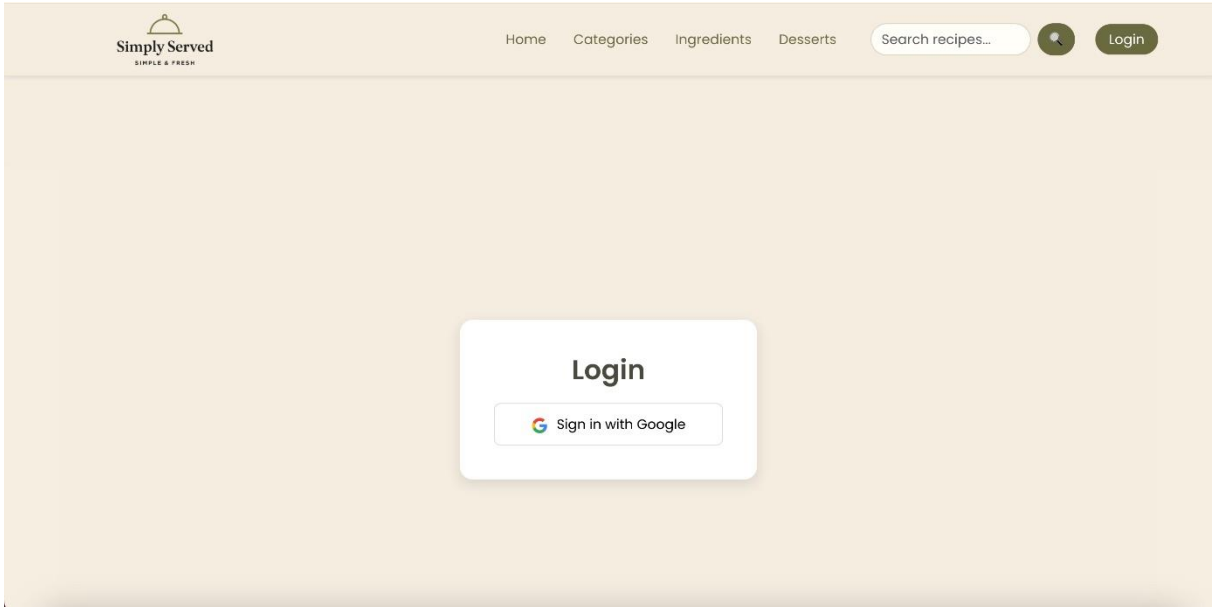
Εικόνα 2

6.2 Αυθεντικοποίηση Χρήστη (Authentication)

Η διαδικασία της ταυτοποίησης στο **Simply Served** έχει σχεδιαστεί με γνώμονα τη μέγιστη δυνατή ευχρηστία και την εξάλειψη των εμποδίων κατά την είσοδο του χρήστη (Registration Friction). Η επιλογή της μεθόδου **Single Sign-On (SSO)** μέσω της Google επιτρέπει στον χρήστη να συνδεθεί με ένα μόνο κλικ, αποφεύγοντας τη χρονοβόρα διαδικασία δημιουργίας νέου λογαριασμού και την απομνημόνευση επιπλέον κωδικών πρόσβασης. Η διεπαφή της σελίδας σύνδεσης (Εικόνα 3) ακολουθεί μια εξαιρετικά μινιμαλιστική προσέγγιση, τοποθετώντας το επίσημο κουμπί "Sign in with Google" στο κέντρο της οθόνης μέσα σε ένα "καθαρό" πλαίσιο (modal), το οποίο χρησιμοποιεί ελαφριές σκιές για να ξεχωρίζει από το υπόβαθρο.

- **Λειτουργία Σελίδας Login:** Η υλοποίηση βασίζεται στη βιβλιοθήκη @react-oauth/google, η οποία διασφαλίζει ότι η αλληλεπίδραση πραγματοποιείται σε ασφαλές περιβάλλον απευθείας με τους εξυπηρετητές της Google. Κατά την ενεργοποίηση της σύνδεσης, η εφαρμογή λαμβάνει ένα προσωρινό ID Token, το οποίο αποστέλλεται στο Backend για επαλήθευση και αυτόματη εγγραφή του χρήστη στη βάση δεδομένων MySQL, εφόσον πρόκειται για την πρώτη του επίσκεψη.
- **Προστασία Δεδομένων και JWT:** Η επιτυχής ταυτοποίηση αποτελεί το έναυσμα για την έκδοση ενός **JSON Web Token (JWT)** από το Flask Backend, το οποίο λειτουργεί ως το ψηφιακό "διαβατήριο" του χρήστη για όλη τη διάρκεια της συνεδρίας. Το token αυτό αποθηκεύεται τοπικά στο πρόγραμμα περιήγησης (localStorage), επιτρέποντας στην εφαρμογή να διατηρεί τη σύνδεση ενεργή ακόμα και μετά την ανανέωση της σελίδας. Χωρίς αυτό το κλειδί ασφαλείας, η πρόσβαση σε κρίσιμες λειτουργίες, όπως η αποθήκευση συνταγών στη Wishlist, η υποβολή βαθμολογιών και η προβολή του ιστορικού μαγειρικής, παραμένει αποκλεισμένη, διασφαλίζοντας την ιδιωτικότητα των προσωπικών δεδομένων.

Η ενσωμάτωση της αυθεντικοποίησης στο **AuthContext** της React επιτρέπει σε ολόκληρη την εφαρμογή να αντιδρά σε πραγματικό χρόνο. Έτσι, μόλις ολοκληρωθεί η διαδικασία που φαίνεται στην Εικόνα 3, το Navbar ενημερώνεται αυτόματα προβάλλοντας το avatar του χρήστη, ενώ το Dashboard ξεκλειδώνει τις προηγμένες λειτουργίες συστάσεων, προσφέροντας μια ολοκληρωμένη και ασφαλή εμπειρία χρήσης.



Εικόνα 3

6.3 Αναζήτηση και Κατηγοριοποίηση (Browsing)

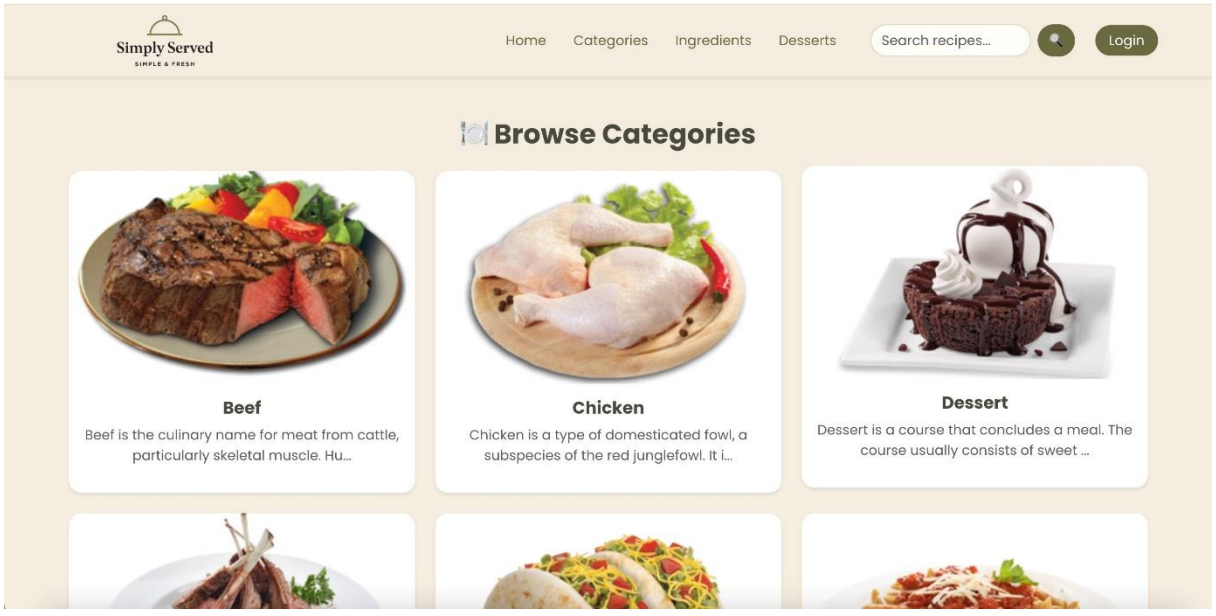
Η αρχιτεκτονική του **Simply Served** υποστηρίζει πολλαπλά επίπεδα φιλτραρίσματος, διασφαλίζοντας ότι ο χρήστης μπορεί να εντοπίσει τη συνταγή που επιθυμεί με ελάχιστα βήματα. Η εφαρμογή υιοθετεί μια σπονδυλωτή προσέγγιση, όπου κάθε μέθοδος αναζήτησης επικοινωνεί με διαφορετικά Endpoints του Backend για την ανάκτηση των δεδομένων σε μορφή JSON.

- **Πλοήγηση ανά Κατηγορία (Browsing by Categories):** Η ενότητα "Browse Categories" (Εικόνα 4) επιτρέπει τον ταχύ διαχωρισμό των συνταγών βάσει βασικών διατροφικών ομάδων ή τύπου γεύματος. Ο χρήστης μπορεί να επιλέξει πιάτα με βάση το κεντρικό υλικό, όπως το βόειο κρέας (Beef) ή το κοτόπουλο (Chicken), καθώς και ειδικές κατηγορίες όπως τα επιδόρπια

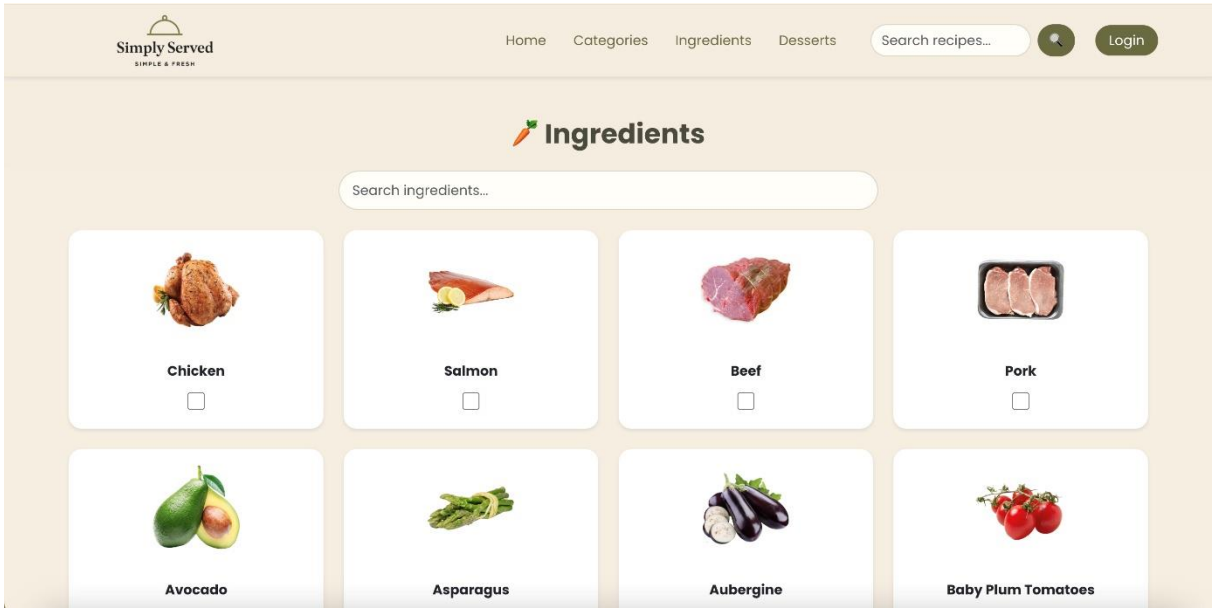
(Dessert). Κάθε κατηγορία συνοδεύεται από μια σύντομη περιγραφή και μια αντιπροσωπευτική εικόνα, διευκολύνοντας την οπτική ταυτοποίηση.

- **Οπτική Αναζήτηση βάσει Υλικών (Search by Ingredients):** Η λειτουργία αυτή (Εικόνα 5) αποτελεί την πλέον καινοτόμα προσέγγιση της εφαρμογής, καθώς επιτρέπει στον χρήστη να λειτουργήσει "αντίστροφα", επιλέγοντας τα υλικά που έχει ήδη στη διάθεσή του. Μέσω του IngredientSelector.jsx component, ο χρήστης επιλέγει οπτικά εικονίδια υλικών (π.χ. Avocado, Salmon, Beef). Τα επιλεγμένα IDs αποστέλλονται στο Flask API, το οποίο εκτελεί τον αλγόριθμο αντιστοίχισης για να επιστρέψει τις βέλτιστες προτάσεις.
- **Παρουσίαση Αποτελεσμάτων (Results Display):** Με την επιλογή μιας κατηγορίας ή την εκτέλεση αναζήτησης, το σύστημα προβάλλει τα αποτελέσματα (Εικόνα 6) σε μια δομημένη σελίδα με χρήση του **Bootstrap Grid System**. Στην περίπτωση των επιδορπίων (Sweet Desserts), οι κάρτες εμφανίζουν τις συνταγές με δυνατότητα άμεσης προβολής ("View Recipe") ή προσθήκης στα αγαπημένα. Η ταχύτητα απόκρισης κατά την εναλλαγή των φίλτρων επιτυγχάνεται χάρη στην SPA φύση της εφαρμογής και τη χρήση της React, η οποία ενημερώνει δυναμικά το UI χωρίς ανανέωση της σελίδας.

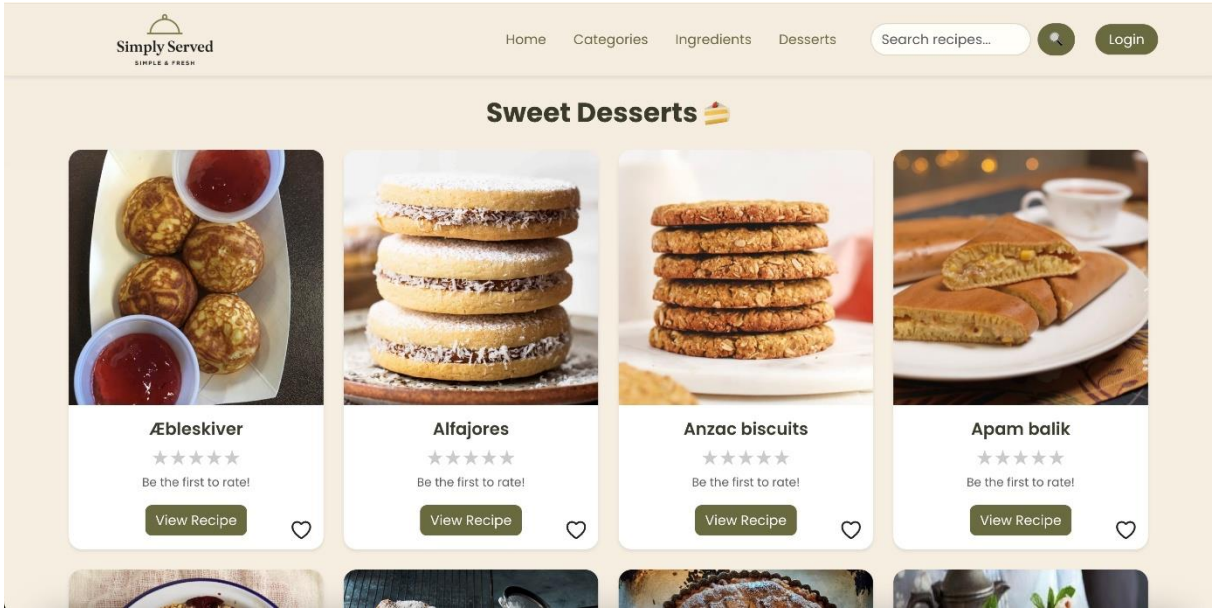
Η χρήση καθαρών γραμμών και χρωματιστών tags στις κάρτες βοηθά στην οπτική ιεραρχία, επιτρέποντας στον χρήστη να εντοπίζει άμεσα τις πληροφορίες που τον ενδιαφέρουν. Η συνδυαστική χρήση αυτών των εργαλείων μετατρέπει τη διαδικασία εύρεσης γεύματος από μια απλή αναζήτηση σε μια διαδραστική εμπειρία εξερεύνησης.



Εικόνα 4



Εικόνα 5



Εικόνα 6

6.4 Προσωποποιημένο Dashboard Χρήστη

Το Dashboard (Εικόνα 7) έχει σχεδιαστεί ως ένας ενεργός «γνωσιακός χάρτης», ο οποίος μετατρέπει τη στατική πληροφορία της βάσης δεδομένων σε μια ζωντανή και προσαρμοσμένη εμπειρία χρήστη. Η διεπαφή είναι οργανωμένη με τέτοιο τρόπο ώστε να παρέχει στον χρήστη μια πλήρη εικόνα του γαστρονομικού του προφίλ με την πρώτη ματιά, αξιοποιώντας την αρχιτεκτονική των **Components** της React για τον διαχωρισμό των λειτουργιών.

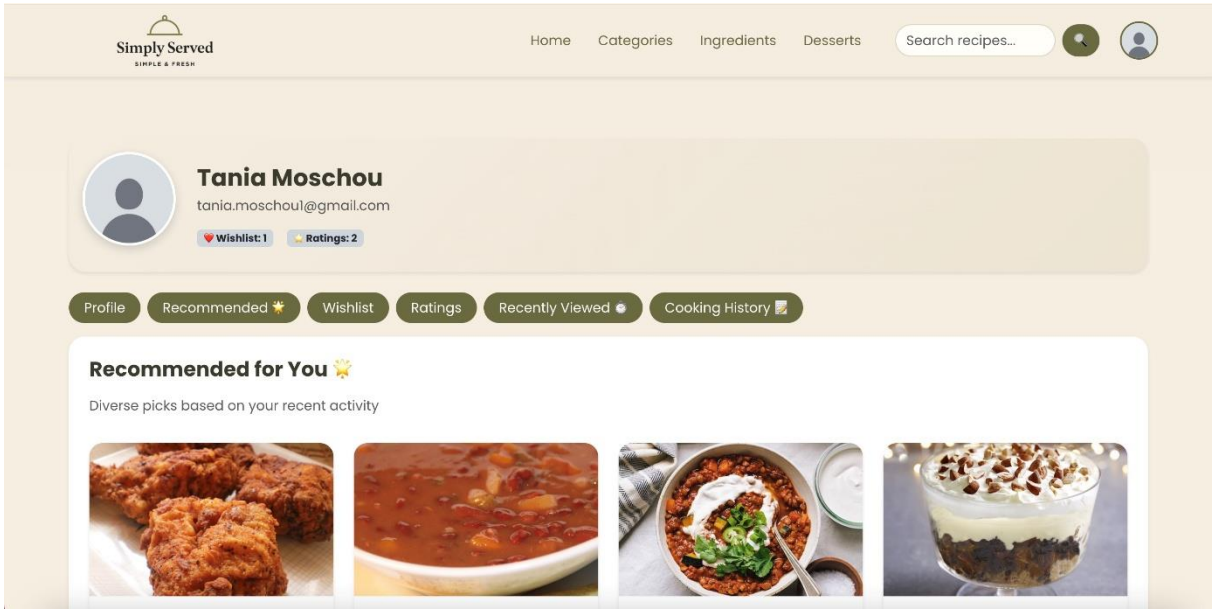
- **Προφίλ Χρήστη και Στατιστικά:** Στο επάνω μέρος του Dashboard προβάλλονται τα στοιχεία ταυτοποίησης που έχουν ανακτηθεί μέσω του Google OAuth, όπως το ονοματεπώνυμο, η διεύθυνση email και η εικόνα προφίλ. Δίπλα στα προσωπικά στοιχεία, ενσωματώνονται δυναμικά στατιστικά (όπως ο συνολικός αριθμός των αγαπημένων συνταγών και των βαθμολογιών), τα οποία αντλούνται σε πραγματικό χρόνο από τη MySQL. Αυτή η

οπτικοποίηση προσφέρει στον χρήστη μια αίσθηση ιδιοκτησίας και προόδου εντός της εφαρμογής.

- **Σύστημα Έξυπνων Συστάσεων (Recommended for You):** Το πιο κρίσιμο τμήμα του Dashboard είναι η ενότητα των συστάσεων, όπου υλοποιείται η «νοημοσύνη» του **Simply Served**. Εδώ, το Backend εκτελεί τους αλγορίθμους **Content-Based** και **Collaborative Filtering**, επεξεργαζόμενο το ιστορικό και τις προτιμήσεις του χρήστη. Τα αποτελέσματα εμφανίζονται σε μια οριζόντια διάταξη καρτών, επιτρέποντας στον χρήστη να ανακαλύψει νέα πιάτα που ταιριάζουν στο «γευστικό του προφίλ». Η ταχύτητα εμφάνισης αυτών των προτάσεων διασφαλίζεται μέσω τεχνικών caching, προσφέροντας μια απρόσκοπτη ροή πληροφορίας.

- **Κέντρο Ελέγχου και Εργαλειοθήκη:** Στο κάτω μέρος της σελίδας, η «Εργαλειοθήκη» παρέχει γρήγορους συνδέσμους προς τις πιο συχνά χρησιμοποιούμενες ενότητες: τη **Wishlist** (Αγαπημένα), τις **Βαθμολογίες** (Ratings) και το **Ιστορικό** (History). Η χρήση του WishlistContext εξασφαλίζει ότι οποιαδήποτε αλλαγή γίνει σε αυτές τις λίστες (π.χ. αφαίρεση μιας συνταγής από τα αγαπημένα) αντικατοπτρίζεται αμέσως στο Dashboard χωρίς να απαιτείται επαναφόρτωση της σελίδας.

Ο συνολικός σχεδιασμός του Dashboard προάγει την έννοια της εξατομίκευσης, καθιστώντας την εφαρμογή έναν ενεργό βοηθό που μαθαίνει από τον χρήστη και εξελίσσεται μαζί του, βελτιώνοντας συνεχώς την ποιότητα των γαστρονομικών προτάσεων.



Εικόνα 7

6.5 Λεπτομέρειες Συνταγής και Εκτέλεση

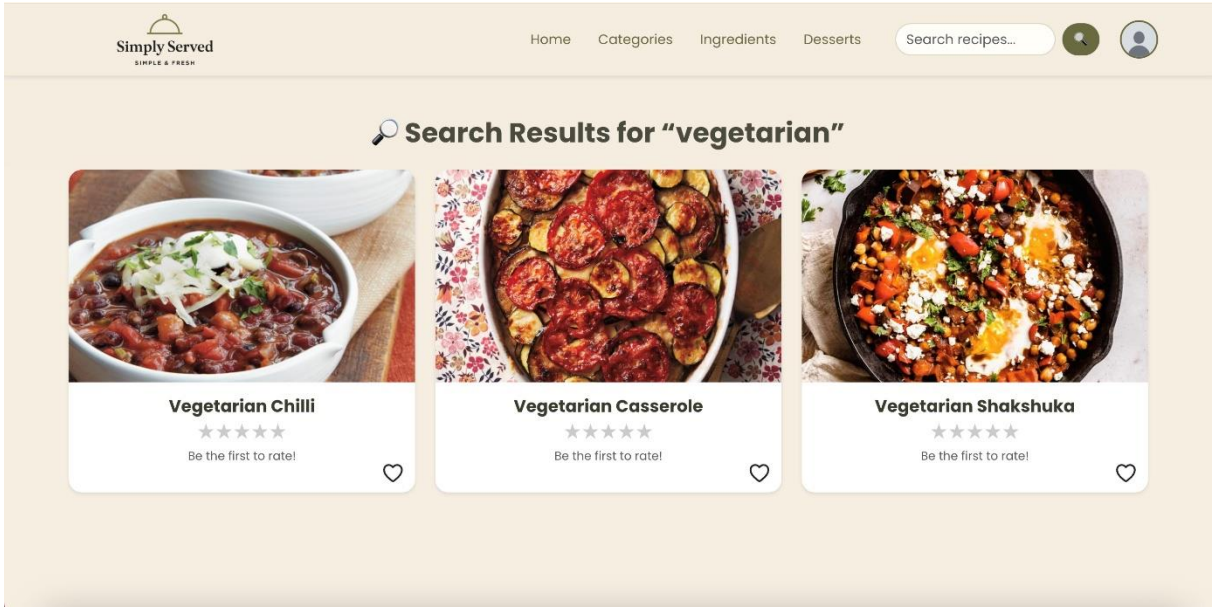
Η αναλυτική προβολή της συνταγής (Εικόνες 8, 9, 10) λειτουργεί ως ένα δυναμικό Page Component (RecipeDetails.jsx), το οποίο αντλεί δεδομένα σε πραγματικό χρόνο από το Backend μέσω ενός GET request. Η σελίδα είναι δομημένη ιεραρχικά, ξεκινώντας από τη γενική ταυτότητα του πιάτου και καταλήγοντας στις τεχνικές λεπτομέρειες της εκτέλεσης.

- **Ταυτότητα και Χαρακτηριστικά Συνταγής:** Στην κορυφή της σελίδας (Εικόνα 8), ο χρήστης βλέπει τον τίτλο της συνταγής (π.χ. "Vegetarian Casserole") συνοδευόμενο από την κατηγορία (Vegetarian) και την εθνική προέλευση (British), στοιχεία που αντλούνται από τα metadata της βάσης δεδομένων. Ένα ιδιαίτερο λειτουργικό στοιχείο είναι το κουμπί "Mark as Cooked", το

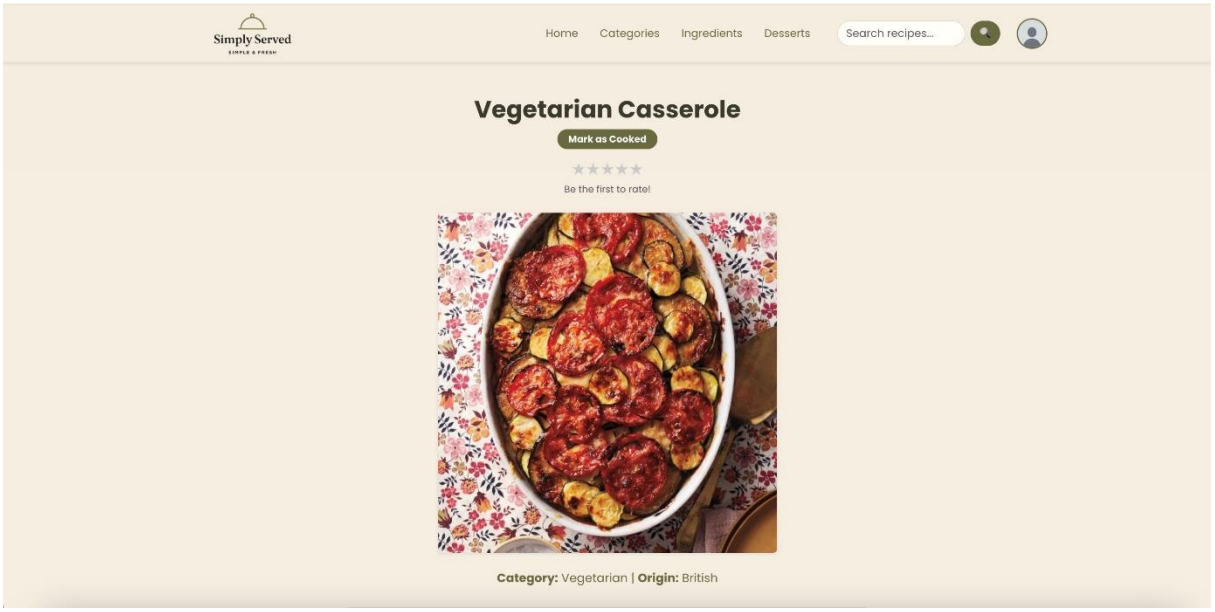
οποίο επιτρέπει στον χρήστη να ενημερώσει το ιστορικό του, τροφοδοτώντας τον αλγόριθμο συστάσεων με νέα δεδομένα αλληλεπίδρασης. Η οπτική παρουσίαση ολοκληρώνεται με μια κεντρική εικόνα υψηλής ανάλυσης, η οποία ακολουθεί το responsive πρότυπο για βέλτιστη εμφάνιση σε κάθε συσκευή.

- **Λίστα Υλικών και Οδηγίες Εκτέλεσης:** Στο κεντρικό τμήμα (Εικόνα 9), τα υλικά παρουσιάζονται σε μια «καθαρή» λίστα, όπου αναγράφονται με ακρίβεια οι ποσότητες και οι μονάδες μέτρησης (π.χ. "Rapeseed Oil — 1", "Garlic — 3 cloves"). Ακολουθεί η ενότητα των οδηγιών ("Instructions"), η οποία είναι τοποθετημένη μέσα σε ένα ευανάγνωστο πλαίσιο με επαρκή λευκό χώρο, διευκολύνοντας τον χρήστη να παρακολουθεί τα βήματα της εκτέλεσης ενώ μαγειρεύει. Η χρήση εικονιδίων (emojis) όπως ο «Chef» προσδίδει έναν πιο φιλικό και διαδραστικό τόνο στη διεπαφή.
- **Πολυμέσα και Διαδραστικότητα:** Στο τέλος της σελίδας (Εικόνα 10), η εφαρμογή ενισχύει την εκπαιδευτική της αξία παρέχοντας τη δυνατότητα παρακολούθησης της συνταγής σε μορφή βίντεο μέσω του κουμπιού "Watch on YouTube". Αυτή η προσθήκη ενσωματώνει εξωτερικές πηγές πολυμέσων, μετατρέποντας το **Simply Served** από ένα στατικό βιβλίο συνταγών σε μια ολοκληρωμένη πλατφόρμα μάθησης. Επιπλέον, η παρουσία του floating action button (κόκκινο βέλος) επιτρέπει την άμεση επιστροφή στην κορυφή της σελίδας, βελτιώνοντας την πλοήγηση σε μακροσκελείς οδηγίες.

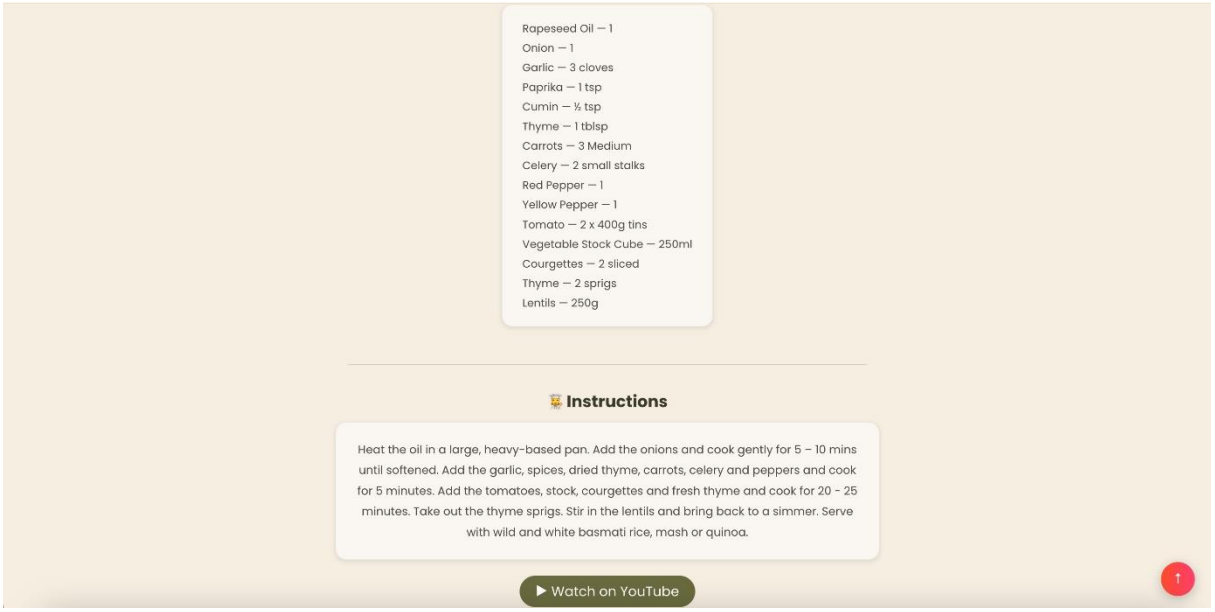
Η συνολική εμπειρία στην ενότητα των λεπτομερειών αντανακλά τη φιλοσοφία του **Simply Served** για απλότητα και λειτουργικότητα, συνδυάζοντας την ακρίβεια των δεδομένων της MySQL με τη δυναμική παρουσίαση της React.



Εικόνα 8



Εικόνα 9



Εικόνα 10

Κεφάλαιο 7ο: Συμπεράσματα και Μελλοντικές Κατευθύνσεις

Με την ολοκλήρωση της ανάπτυξης και της παρουσίασης της εφαρμογής **Simply Served**, το παρόν κεφάλαιο αποτελεί τον επίλογο της εργασίας, προσφέροντας μια κριτική ανασκόπηση των όσων επιτεύχθηκαν. Η δημιουργία ενός ολοκληρωμένου συστήματος συστάσεων συνταγών βασισμένου σε πραγματικά υλικά αποτέλεσε μια πρόκληση που συνδύασε διαφορετικά πεδία της πληροφορικής, από τη διαχείριση βάσεων δεδομένων έως την ανάπτυξη σύγχρονων διεπαφών χρήστη.

Σκοπός αυτού του τελικού σταδίου είναι η αποτίμηση της λειτουργικότητας της εφαρμογής σε σχέση με τους αρχικούς στόχους που τέθηκαν. Θα αναλυθούν τα συμπεράσματα που προέκυψαν από τη διαδικασία του σχεδιασμού και της υλοποίησης, επισημαίνοντας τα πλεονεκτήματα αλλά και τους περιορισμούς του συστήματος.

Ταυτόχρονα, καθώς η τεχνολογία και οι ανάγκες των χρηστών εξελίσσονται, η εργασία ολοκληρώνεται με την παράθεση προτάσεων για μελλοντικές επεκτάσεις. Η εφαρμογή **Simply Served** έχει σχεδιαστεί με τέτοιο τρόπο ώστε να αποτελεί μια σταθερή βάση, πάνω στην οποία μπορούν να ενσωματωθούν νέες τεχνολογίες και λειτουργίες, με στόχο την περαιτέρω βελτιστοποίηση της εμπειρίας του χρήστη στον τομέα της γαστρονομίας και της οργάνωσης της κουζίνας.

7.1 Συμπεράσματα

Η ολοκλήρωση της εφαρμογής **Simply Served** αποδεικνύει ότι ο συνδυασμός σύγχρονων τεχνολογιών ιστού με αλγορίθμους συστημάτων συστάσεων μπορεί να προσφέρει ουσιαστικές λύσεις σε καθημερινά προβλήματα. Μέσα από τη διαδικασία ανάπτυξης, προέκυψαν ορισμένα βασικά συμπεράσματα που αφορούν τόσο την τεχνική υλοποίηση όσο και τη λειτουργική αξία της πλατφόρμας.

Αρχικά, η επιλογή της αρχιτεκτονικής **Client-Server** με τη χρήση της **React.js** για το Frontend και του **Flask** για το Backend αποδείχθηκε εξαιρετικά αποδοτική. Η React επέτρεψε τη δημιουργία μιας ταχύτατης διεπαφής που ανταποκρίνεται άμεσα στις ενέργειες του χρήστη, ενώ το Flask παρείχε την απαραίτητη ευελιξία για την ενσωμάτωση των αλγορίθμων ομοιότητας στην Python. Η χρήση των **JWT tokens** εξασφάλισε ένα σταθερό επίπεδο ασφάλειας, επιτρέποντας ταυτόχρονα την απρόσκοπτη μετακίνηση του χρήστη μεταξύ των διαφορετικών λειτουργιών της εφαρμογής.

Σε επίπεδο λειτουργικότητας, ο αλγόριθμος **αναζήτησης βάσει υλικών** αποτέλεσε το ισχυρότερο σημείο της εφαρμογής. Η ικανότητα του συστήματος να επιστρέφει αποτελέσματα ταξινομημένα με βάση τις ελλείψεις συστατικών (Missing Ingredients) προσδίδει στην εφαρμογή έναν χαρακτήρα "έξυπνου βοηθού". Επιπλέον, το **υβριδικό σύστημα συστάσεων** (Content-based και Collaborative Filtering) πέτυχε να προσφέρει προτάσεις που εξελίσσονται μαζί με τον χρήστη, αποδεικνύοντας ότι η ανάλυση της προηγούμενης συμπεριφοράς (ratings και history) είναι το κλειδί για τη διατήρηση του ενδιαφέροντος των χρηστών.

Τέλος, η εμπειρία χρήστη ενισχύθηκε σημαντικά από τον **ανταποκρινόμενο σχεδιασμό (Responsive Design)**. Η εφαρμογή παρέμεινε λειτουργική σε ένα ευρύ φάσμα συσκευών, γεγονός κρίσιμο για μια πλατφόρμα μαγειρικής που συχνά χρησιμοποιείται σε περιβάλλοντα εκτός γραφείου, όπως η κουζίνα. Συμπερασματικά, το Simply Served δεν είναι απλώς ένας ψηφιακός κατάλογος συνταγών, αλλά μια ολοκληρωμένη πλατφόρμα που διαχειρίζεται έξυπνα την πληροφορία, διευκολύνοντας τη διαδικασία λήψης αποφάσεων για τη διατροφή του χρήστη.

7.2 Μελλοντικές επεκτάσεις

Παρά την ολοκληρωμένη φύση της τρέχουσας έκδοσης, το **Simply Served** έχει σχεδιαστεί ως μια επεκτάσιμη πλατφόρμα. Η αρχιτεκτονική του επιτρέπει την ενσωμάτωση νέων δυνατοτήτων που θα

μπορούσαν να αναβαθμίσουν τη χρηστικότητα και την ευφυΐα του συστήματος. Οι κυριότερες προτάσεις για μελλοντικές επεκτάσεις περιλαμβάνουν:

- **Ενσωμάτωση Τεχνητής Νοημοσύνης (Generative AI):** Η προσθήκη ενός AI Chatbot (βασισμένου σε μοντέλα όπως το GPT ή το Gemini) θα επέτρεπε στον χρήστη να κάνει φυσικό διάλογο με την εφαρμογή. Για παράδειγμα, ο χρήστης θα μπορούσε να ζητήσει: «Πρότεινέ μου μια παραλλαγή αυτής της συνταγής χωρίς γλουτένη» ή «Φτιάξε μου ένα εβδομαδιαίο πλάνο διατροφής με βάση τις συνταγές που έχω στη Wishlist μου».
- **Ανάλυση Διατροφικών Στοιχείων:** Μια σημαντική προσθήκη θα ήταν ο αυτόματος υπολογισμός θερμίδων και μακροθρεπτικών συστατικών (πρωτεΐνες, υδατάνθρακες, λιπαρά) για κάθε συνταγή. Αυτό θα μπορούσε να υλοποιηθεί μέσω σύνδεσης με εξωτερικά APIs διατροφικών δεδομένων, προσφέροντας στον χρήστη μια πιο ολοκληρωμένη εικόνα για τη διατροφή του.
- **Σύστημα "Smart Pantry" (Έξυπνο Τροφοδότης):** Η δυνατότητα του χρήστη να διατηρεί ένα ψηφιακό απόθεμα των υλικών που έχει στο σπίτι του. Το σύστημα θα μπορούσε να αφαιρεί αυτόματα τα υλικά που χρησιμοποιήθηκαν σε μια συνταγή που σημειώθηκε ως "Cooked" και να ειδοποιεί τον χρήστη όταν κάποιο βασικό υλικό κοντεύει να τελειώσει.
- **Computer Vision και OCR:** Μέσω της κάμερας του κινητού, η εφαρμογή θα μπορούσε να "διαβάζει" τις αποδείξεις από το super market ή να αναγνωρίζει υλικά πάνω στον πάγκο της κουζίνας, εισάγοντάς τα αυτόματα στη λίστα αναζήτησης, εξοικονομώντας χρόνο από τη χειροκίνητη πληκτρολόγηση.
- **Κοινωνική Δικτύωση και Διαμοιρασμός:** Η δημιουργία μιας κοινότητας όπου οι χρήστες θα μπορούν να ανεβάζουν τις δικές τους φωτογραφίες από τις εκτελέσεις των συνταγών, να ανταλλάσσουν συμβουλές στα σχόλια και να δημιουργούν "δημόσιες λίστες" (π.χ. "Οι καλύτερες συνταγές για φοιτητές").

Οι παραπάνω επεκτάσεις θα μπορούσαν να μετατρέψουν το **Simply Served** από ένα εργαλείο αναζήτησης σε ένα ολοκληρωμένο οικοσύστημα διαχείρισης της οικιακής οικονομίας και της προσωπικής υγείας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Sharda, R., Delen, D., & Turban, E. (2020). Analytics, Data Science, & Artificial Intelligence: Systems for Decision Support. Pearson. (Το βιβλίο αυτό αναλύει διεξοδικά πώς τα πληροφοριακά συστήματα βοηθούν στη λήψη αποφάσεων και την αντιμετώπιση του information overload).

[2] Ricci, F., Rokach, L., & Shapira, B. (2015). Recommender Systems Handbook. Springer. (Αποτελεί την "βίβλο" των συστημάτων συστάσεων και αναλύει σε βάθος τις έννοιες του utility και του cognitive load).

[3] Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer. (Το συγκεκριμένο σύγγραμμα αναλύει διεξοδικά τον μαθηματικό τρόπο με τον οποίο το περιεχόμενο μετατρέπεται σε διανύσματα χαρακτηριστικών για την εύρεση ομοιότητας).

[4] Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). Recommender Systems: An Introduction. Cambridge University Press. (Το βιβλίο αυτό αναλύει διεξοδικά τους αλγορίθμους γειτνίασης και τη σημασία των πινάκων αξιολόγησης στο συνεργατικό φιλτράρισμα).

[5] Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction. (Το συγκεκριμένο άρθρο αποτελεί τη θεμελιώδη εργασία που κατηγοριοποιεί τις διαφορετικές μεθόδους υβριδοποίησης, όπως το cascading, το feature augmentation και το switching).

[6] Leskovec, J., Rajaraman, A., & Ullman, J. D. (2020). Mining of Massive Datasets. Cambridge University Press. (Το βιβλίο αυτό αναλύει σε βάθος τους αλγορίθμους επεξεργασίας μεγάλων δεδομένων και τις μαθηματικές μεθόδους ομοιότητας σε συστήματα συστάσεων).

[7] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. IEEE Computer Society. (Το θεμελιώδες άρθρο που εξηγεί πώς η παραγοντοποίηση πινάκων κυριάρχησε στα σύγχρονα συστήματα συστάσεων).

[8] Cover, T., & Hart, P. (1967). Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory. (Η κλασική εργασία που έθεσε τις βάσεις για την κατανόηση της απόδοσης και των ορίων του αλγορίθμου των πλησιέστερων γειτόνων).

[9] Banks, A., & Porcello, E. (2020). Learning React: Modern Patterns for Developing React Apps. O'Reilly Media. (Το βιβλίο αυτό θεωρείται θεμελιώδες για την κατανόηση των σύγχρονων προτύπων της React, των Hooks και της λειτουργίας του Virtual DOM).

[10] Grinberg, M. (2018). Flask Web Development: Developing Web Applications with Python. O'Reilly Media. (Το βιβλίο αυτό αποτελεί τον πληρέστερο οδηγό για την ανάπτυξη RESTful APIs με Flask, καλύπτοντας θέματα από τη βασική δομή έως την ασφάλεια και τα Virtual Environments).