ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Σύστημα ελέγχου παρουσίας φοιτητών/τριών με τη χρήση Raspberry Pi»

Του φοιτητή
Χάιτα Κωνσταντίνου
Αρ. Μητρώου: 2019182

Επιβλέπουσα
Παπαδοπούλου Μαρία
Επίκουρη Καθηγήτρια

24 Μαΐου 2024

Σύστημα ελέγχου παρουσίας φοιτητών/τριών με τη χρήση Raspberry Pi
Κωδικός Δ.Ε. 24149
Ονοματεπώνυμο φοιτητή Κωνσταντίνος Χάιτας
Ονοματεπώνυμο εισηγητή Μαρία Παπαδοπούλου
Ημερομηνία ανάληψης Δ.Ε. 14/03/2024
Ημερομηνία περάτωσης Δ.Ε. 24/05/2024

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

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

«Αφιερώνω αυτή τη διατριβή στην ακατάπαυστη αναζήτηση του αγνώστου, στις σιωπηλές ώρες προβληματισμού όπου ο νους ταξιδεύει ανάμεσα στα αστέρια της γνώσης.»

Πρόλογος

Η επιλογή μου να επικεντρωθώ στην ανάπτυξη ενός συστήματος ελέγχου παρουσίας φοιτητών πηγάζει από το ενδιαφέρον μου στην αξιοποίηση της τεχνολογίας για την αντιμετώπιση εκπαιδευτικών προκλήσεων. Η παρατήρηση των αναποτελεσματικοτήτων και των παρωχημένων μεθόδων που επικρατούν στην παρακολούθηση της παρουσίας με έκανε να αναζητήσω μια λύση που όχι μόνο θα απλοποιούσε τη διαδικασία αλλά και θα ενσωματωνόταν ομαλά στο εκπαιδευτικό πλαίσιο.

Αυτό το έργο, μου επέτρεψε να συνδυάσω το πάθος μου για την τεχνολογία με μια πρακτική εφαρμογή που έχει απτά οφέλη τόσο για τους εκπαιδευτικούς όσο και για τους φοιτητές. Με την αυτοματοποίηση της παρακολούθησης της παρουσίας, εξοικονομούμε πολύτιμο χρόνο από τη διδακτική διαδικασία αλλά και ενισχύουμε την ακρίβεια και την αξιοπιστία των δεδομένων. Αυτή η πρωτοβουλία ξεπερνά την απλή ευκολία και αποτελεί ένα βήμα προς την εκσυγχρονισμό των εκπαιδευτικών πρακτικών.

Η διαδικασία ανάπτυξης αυτού του συστήματος με προκάλεσε και διεύρυνε τις τεχνικές μου δεξιότητες, παρέχοντας μια πλούσια εμπειρία μάθησης που εκτείνεται πέρα από τη θεωρητική γνώση που αποκτήθηκε στην αίθουσα διδασκαλίας. Παρείχε επίσης μια μοναδική ευκαιρία να συμβάλω θετικά στην εκπαιδευτική κοινότητα, συμφωνώντας με την πεποίθησή μου στη δύναμη της τεχνολογίας να διευκολύνει και να βελτιώσει την καθημερινή μας ζωή.

Περίληψη

Η παρούσα διπλωματική εργασία διερευνά τη δημιουργία και την εφαρμογή ενός συστήματος ελέγχου παρουσίας φοιτητών με τη χρήση του Raspberry Pi, καταδεικνύοντας τις δυνατότητες της τεχνολογίας να μεταμορφώσει την εκπαιδευτική διοίκηση και να ενισχύσει τη μαθησιακή εμπειρία. Το έργο επικεντρώνεται στην ανάπτυξη ενός συνεκτικού συστήματος που ενσωματώνει μια βάση δεδομένων Firebase στο Firestore, τα Google Apps Scripts ως μέσω αυτοματοποίησης στα Google Sheets, ένα Raspberry Pi και μια εφαρμογή Android. Η αρχιτεκτονική του συστήματος έχει σχεδιαστεί για να εξασφαλίζει συγχρονισμό και αποθήκευση δεδομένων σε πραγματικό χρόνο, παρέχοντας μια αξιόπιστη και αποτελεσματική μέθοδο για τη διαχείριση της παρουσίας των φοιτητών. Τα βασικά επιτεύγματα περιλαμβάνουν την ενσωμάτωση διαφόρων τεχνολογιών σε μια επιχειρησιακή μονάδα που απλοποιεί και επιταχύνει τη διαχείριση των παρουσιών, για την οποία προηγουμένως χρησιμοποιούνταν χειροκίνητες διαδικασίες. Η χρήση του Raspberry Pi ως φυσική διεπαφή για σάρωση κώδικα QR στις εισόδους των τάξεων αποτελεί παράδειγμα της πρακτικής εφαρμογής της τεχνολογίας σε εκπαιδευτικά περιβάλλοντα. Επιπλέον, η προσαρμοσμένη εφαρμογή Android επιτρέπει στους χρήστες να δημιουργούν κωδικούς QR για την παρακολούθηση μαθημάτων, αυτοματοποιώντας περαιτέρω την εισαγωγή δεδομένων και ελαχιστοποιώντας τα σφάλματα. Καθ' όλη τη διάρκεια των φάσεων ανάπτυξης, το έργο αντιμετώπισε πολλαπλές προκλήσεις, συμπεριλαμβανομένης της αξιοπιστίας του συστήματος, της ασφάλειας των δεδομένων και του σχεδιασμού της διεπαφής χρήστη, οδηγώντας σε σημαντικά μαθησιακά αποτελέσματα σχετικά με την επεκτασιμότητα του συστήματος και την αλληλεπίδραση των χρηστών. Προτείνονται μελλοντικές δράσεις για τη βελτίωση του συστήματος και της εμπειρίας του χρήστη, με στόχο την επέκταση των δυνατοτήτων και της προσαρμοστικότητας του συστήματος σε διαφορετικά εκπαιδευτικά περιβάλλοντα. Το έργο αυτό καταδεικνύει την αποτελεσματικότητα της ενσωμάτωσης ψηφιακών εργαλείων στις εκπαιδευτικές διαδικασίες, σηματοδοτώντας μια σημαντική πρόοδο στη διαχείριση των ακαδημαϊκών πόρων και τη συμμετοχή των φοιτητών.

«Student Attendance System Using Raspberry Pi»

«Konstantinos Chaitas»

Abstract

This thesis explores the creation and implementation of a student attendance system using Raspberry Pi, demonstrating the potential of technology to transform educational administration and enhance the learning experience. The project centers around developing a cohesive system that integrates Firestore database, Google Apps Scripts, Raspberry Pi hardware, and an Android application to automate the attendance recording process. The system's architecture is designed to ensure real-time data synchronization and storage, providing a reliable and efficient method for managing student attendance. Key achievements include the successful integration of various technologies into a seamless operational unit that simplifies and speeds up attendance management, previously burdened by manual processes. The use of Raspberry Pi as a physical interface for QR code scanning at classroom entries exemplifies the practical application of affordable technology in educational settings. Additionally, the custom Android application allows users to generate QR codes for class attendance, further automating data entry and minimizing errors. Throughout the development and deployment phases, the project addressed multiple challenges, including system reliability, data security, and user interface design, leading to significant learning outcomes regarding system scalability and user interaction. Future enhancements are suggested to improve system robustness and user experience, aiming to expand the system’s capabilities and adaptability to different educational environments. This work illustrates the effectiveness of integrating digital tools into educational processes, marking a substantial advancement in academic resource management and student engagement.

Ευχαριστίες

Σε αυτήν την ενότητα, θα ήθελα να εκφράσω τις ειλικρινείς ευχαριστίες μου σε όλους όσοι συνέβαλαν στην ολοκλήρωση της διπλωματικής μου εργασίας.

Πρώτα και κύρια, θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια μου, Παπαδοπούλου Μαρία, για την καθοδήγηση, την υποστήριξη και την πολύτιμη γνώση που μου παρείχε καθ' όλη τη διάρκεια της εργασίας. Η συνεχής ενθάρρυνση και οι συμβουλές της ήταν καθοριστικές για την πρόοδό μου.

Επίσης, ευχαριστώ θερμά την οικογένειά μου για την αμέριστη ηθική στήριξη που μου παρείχαν. Η υπομονή και η κατανόησή τους καθ' όλη τη διάρκεια αυτής της απαιτητικής διαδικασίας ήταν ανεκτίμητες.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους και συναδέλφους μου για την υποστήριξη και τις ενθαρρυντικές κουβέντες τους. Οι συζητήσεις και οι ιδέες τους βοήθησαν σημαντικά στην ανάπτυξη και τη βελτίωση της εργασίας μου.

Σας ευχαριστώ όλους από καρδιάς.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

G.A.S.	Google Apps Scripts
LMS	Learning Management Systems
SIS	Student Information Systems
BaaS	Backend as a Service
ΔΕΠ	Διδακτικό και Ερευνητικό Προσωπικό
GPIO	General-Purpose Input/Output
OEM	Original Equipment Manufacturer

Κεφάλαιο 1ο: Εισαγωγή

1.1 Υπόβαθρο και Κίνητρο

Η διαδικασία παρακολούθησης της φοίτησης αποτελεί θεμελιώδη πτυχή της εκπαιδευτικής διοίκησης, χρησιμεύοντας όχι μόνο ως καταγραφή της συμμετοχής των φοιτητών αλλά και ως βασικός δείκτης ενασχόλησης και ακαδημαϊκής προόδου. Παραδοσιακά, αυτή η διαδικασία βασιζόταν σε μεγάλο βαθμό σε μη αυτόματες μεθόδους, όπως ονομαστικές κλήσεις και φύλλα εισόδου σε χαρτί. Ενώ αυτές οι πρακτικές είναι βαθιά ριζωμένες στο εκπαιδευτικό σύστημα, έρχονται με σημαντικούς περιορισμούς που μπορεί να θέτουν σε κίνδυνο την αποδοτικότητα, την ακρίβεια και τη συνολική αποτελεσματικότητα.

1.1.1 Η Πρόκληση της Παρακολούθησης Παρουσίας

Η πιο συμβατική μέθοδος παρακολούθησης της παρουσίας περιλαμβάνει εκπαιδευτές που φωνάζουν ονόματα από έναν κατάλογο, με τους φοιτητές να ανταποκρίνονται για να επισημάνουν την παρουσία τους. Εναλλακτικά, τα έντυπα φύλλα εισόδου διανέμονται στην αρχή του μαθήματος, απαιτώντας από τους φοιτητές να σημειώσουν χειροκίνητα τη συμμετοχή τους. Και οι δύο μέθοδοι είναι χρονοβόρες, διακόπτοντας πολύτιμο διδακτικό χρόνο εκτρέποντας την εστίαση από τη διδασκαλία στα διοικητικά καθήκοντα. Αυτές οι παραδοσιακές μέθοδοι είναι ένας γρήγορος και εύκολος τρόπος να καταγραφεί μια παρουσία φοιτητή σε μια τάξη με μικρό αριθμό φοιτητών αποτελεσματικά. Αντίθετα όμως σε μεγαλύτερο αριθμό φοιτητών οι μη αυτόματες ονομαστικές κλήσεις μπορούν να καταναλώσουν σημαντικό μέρος του χρόνου της τάξης, ενώ η διαχείριση και η οργάνωση έντυπων φύλλων εισόδου επιβάλλουν σημαντικό διοικητικό φόρτο. Η χειροκίνητη εισαγωγή δεδομένων παρουσίας σε ηλεκτρονικά συστήματα τήρησης αρχείων επιδεινώνει περαιτέρω αυτές τις ανεπάρκειες, αυξάνοντας τον φόρτο εργασίας του εκπαιδευτικού προσωπικού. Η φύση αυτών των μεθόδων παρακολούθησης παρουσίας εισάγει μεγάλη πιθανότητα σφαλμάτων. Λανθασμένες προφορές ή ακροάσεις κατά τη διάρκεια της ονομαστικής κλήσης, δυσανάγνωστος γραφικός χαρακτήρας στα φύλλα εισόδου και φύλλα που λείπουν μπορεί να οδηγήσουν σε ανακρίβειες ή ακόμη και απώλεια στα αρχεία παρουσίας. Τέτοια σφάλματα μπορούν να θέσουν σε κίνδυνο την αξιοπιστία των δεδομένων συμμετοχής, καθώς και να έχουν άμεσες επιπτώσεις στους φοιτητές, επηρεάζοντας την συμμετοχή τους, τους βαθμούς τους και την επιλογή τους για ορισμένα ακαδημαϊκά οφέλη. Ένα ακόμα μειονέκτημα αυτής της προσέγγισης είναι η δυσκολία στην παρακολούθηση των τάσεων συμμετοχής καθώς οι χειροκίνητες μέθοδοι καθιστούν δύσκολη και χρονοβόρα την ανάλυση των μοτίβων ή των τάσεων συμμετοχής με αποτέλεσμα πολλές φορές να παραλείπονται. Επιπλέον όταν πραγματοποιούνται αυτές οι διαδικασίες, η συγκέντρωση και η ανάλυση των δεδομένων συμμετοχής απαιτούν πρόσθετα βήματα εισαγωγής και χειρισμού δεδομένων, που συχνά οδηγούν σε καθυστερημένα ή προκαθορισμένα συμπεράσματα σχετικά με την παρουσία των φοιτητών και την συμμετοχή τους στην τάξη. Αυτή η έλλειψη άμεσων, αξιοποιήσιμων δεδομένων ίσως εμποδίζει την ικανότητα των εκπαιδευτικών ιδρυμάτων να αντιμετωπίζουν προληπτικά ζητήματα που σχετίζονται με τη συμμετοχή, όπως οι χρόνιες απουσίες και η αποδέσμευση.

Οι περιορισμοί των παραδοσιακών μεθόδων παρακολούθησης παρουσίας υπογραμμίζουν την ανάγκη για την αναζήτηση μιας πιο ακριβής και αποτελεσματικής λύσης. Τα πρακτικά προβλήματα που σχετίζονται με τις μη αυτόματες, ονομαστικές κλήσεις και τις έντυπες εγγραφές υπογραμμίζουν τον επείγοντα χαρακτήρα της υιοθέτησης καινοτόμων προσεγγίσεων στη διαχείριση των δεδομένων. Μια τέτοια λύση θα πρέπει όχι μόνο να ελαφρύνει τον διοικητικό φόρτο για τους εκπαιδευτικούς, αλλά και

να ενισχύσει την αξιοπιστία των αρχείων παρακολούθησης, υποστηρίζοντας έτσι τους ευρύτερους εκπαιδευτικούς στόχους της συμμετοχής και της ακαδημαϊκής επιτυχίας.

1.1.2 Η Ανάγκη για μια Καινοτόμο Λύση

Στο πλαίσιο των σύγχρονων εκπαιδευτικών περιβαλλόντων, η λήψη αποφάσεων βάσει δεδομένων και η αποτελεσματικότητα αυτών είναι όλο και πιο σημαντική [17]. Αυτή η μετατόπιση υπογραμμίζει την ανάγκη για καινοτόμες λύσεις που υπερβαίνουν τους περιορισμούς των παραδοσιακών μεθόδων. Το εξελισσόμενο τοπίο της εκπαίδευσης, που χαρακτηρίζεται από μεγαλύτερους αριθμούς φοιτητών, ποικίλους τρόπους μάθησης (συμπεριλαμβανομένων διαδικτυακών και υβριδικών μοντέλων) και την αυξανόμενη έμφαση στην εξατομικευμένη εκπαίδευση, απαιτεί ένα σύστημα παρακολούθησης παρουσίας που δεν είναι μόνο πιο αποτελεσματικό και ακριβές, αλλά κλιμακούμενο και προσαρμόσιμο σε διάφορα εκπαιδευτικά περιβάλλοντα.

Οι χειρωνακτικές μέθοδοι καταγραφής της παρουσίας είναι χρονοβόρες, μειώνοντας τον ωφέλιμο χρόνο που έχει κάθε συνεδρία για αποτελεσματική εκπαίδευση και μάθηση. Μια πρωτοποριακή λύση θα αυτοματοποιήσει τη διαδικασία, μειώνοντας σημαντικά το χρόνο που δαπανάται σε διοικητικά καθήκοντα από τους εκπαιδευτικούς, επιτρέποντάς τους να αφιερώσουν περισσότερους πόρους σε εκπαιδευτικές δραστηριότητες. Με την βελτιστοποίηση της καταγραφής των παρουσιών, τα πανεπιστήμια έχουν την δυνατότητα να ενισχύσουν την επιχειρησιακή τους αποτελεσματικότητα, διασφαλίζοντας ότι οι διοικητικές διαδικασίες δεν αντλούν πολύτιμο χρόνο από την παράδοση του μαθήματος. Τα ακριβή δεδομένα συμμετοχής είναι κρίσιμα για πολλές βασικές διαδικασίες, συμπεριλαμβανομένης της βαθμολόγησης και της αναφοράς συμμόρφωσης. Η ευαισθησία των μεθόδων χειροκίνητης καταγραφής σε ανθρώπινο λάθος μπορεί να οδηγήσει σε ανακρίβειες που επηρεάζουν τα ακαδημαϊκά αρχεία των φοιτητών και τη λογοδοσία των ιδρυμάτων. Μια καινοτόμος λύση πρέπει να προσφέρει ένα αξιόπιστο μέσο καταγραφής της παρακολούθησης, ελαχιστοποιώντας τα σφάλματα και διασφαλίζοντας την ακεραιότητα των εκπαιδευτικών αρχείων [18]. Αυτό το επίπεδο ακρίβειας είναι απαραίτητο για τη διατήρηση της εμπιστοσύνης στο εκπαιδευτικό σύστημα και την υποστήριξη δίκαιων και διαφανών ακαδημαϊκών πολιτικών. Καθώς τα εκπαιδευτικά ιδρύματα μεγαλώνουν και εξελίσσονται, η ικανότητα κλιμάκωσης των διαδικασιών παρακολούθησης παρουσίας καθίσταται ζωτικής σημασίας. Οι παρόν μέθοδοι δεν είναι εύκολα κλιμακούμενοι, ειδικά ενόψει του αυξανόμενου μεγέθους των τάξεων και της επέκτασης των διαδικτυακών και υβριδικών περιβαλλόντων μάθησης. Μια καινοτόμος λύση καταχώρισης παρουσιών πρέπει να είναι ικανή να προσαρμόζεται σε διαφορετικά μεγέθη και μορφές τάξεων, παρέχοντας συνεπή και αποτελεσματική λειτουργικότητα σε ένα ευρύ φάσμα εκπαιδευτικών πλαισίων. Πέρα από τη λειτουργική ανάγκη αυτής της διαδικασίας, υπάρχει μια στρατηγική ευκαιρία να αξιοποιηθούν τα δεδομένα συμμετοχής για πληροφορίες σχετικά με την αφοσίωση των μαθητών και τις ακαδημαϊκές επιδόσεις [19]. Η διευκόλυνση της ανάλυσης των προτύπων συμμετοχής, εντοπίζοντας τάσεις και συσχετισμούς που μπορούν να ενημερώσουν τις υπηρεσίες υποστήριξης αποτελεί ένα σημαντικό μέρος στη βελτίωση αυτής της διαδικασίας. Επιτρέποντας την ανάλυση δεδομένων σε πραγματικό χρόνο, τα εκπαιδευτικά ιδρύματα μπορούν να αντιμετωπίσουν προληπτικά τις προκλήσεις που σχετίζονται με τη συμμετοχή, ενισχύοντας την επιτυχία και τη διατήρηση των φοιτητών στα μαθήματα.

Η ενσωμάτωση της διαδικασίας καταγραφής της παρουσίας με άλλες εκπαιδευτικές τεχνολογίες παρουσιάζει μια άλλη διάσταση καινοτομίας. Μια λύση που διασυνδέεται απρόσκοπτα με συστήματα διαχείρισης μάθησης (**LMS**), πληροφοριακά συστήματα φοιτητών (**SIS**) και άλλες ψηφιακές πλατφόρμες μπορεί να δημιουργήσει ένα ενοποιημένο και ολοκληρωμένο εκπαιδευτικό οικοσύστημα [20]. Αυτή η ενσωμάτωση διευκολύνει μια ολιστική προσέγγιση στην εκπαιδευτική διαχείριση, όπου

τα δεδομένα συμμετοχής των φοιτητών συμπληρώνουν τα ακαδημαϊκά αρχεία, τις μετρήσεις συμμετοχής, άλλους δείκτες των φοιτητών καθώς και των μαθησιακών αποτελεσμάτων.

Συμπερασματικά, η ανάγκη για βελτίωση της διαδικασίας καταχώρισης παρακολούθησης των φοιτητών οδηγείται από τις απαιτήσεις των σύγχρονων εκπαιδευτικών περιβαλλόντων για μεγαλύτερη αποτελεσματικότητα, ακρίβεια, επεκτασιμότητα και λήψη αποφάσεων βάσει δεδομένων. Μια τέτοια λύση όχι μόνο αντιμετωπίζει τις άμεσες πρακτικές προκλήσεις που συνδέονται με τις παραδοσιακές μεθόδους παρακολούθησης παρουσίας, αλλά ευθυγραμμίζεται επίσης με τους ευρύτερους στόχους της ενίσχυσης της εκπαιδευτικής παράδοσης, της συμμετοχής των φοιτητών και της θεσμικής αποτελεσματικότητας.

1.2 Οι Στόχοι του Έργου

Η ανάπτυξη ενός συστήματος παρακολούθησης φοιτητών με τη χρήση του Raspberry Pi υποδεικνύει την ανάγκη υπέρβασης των περιορισμών των παραδοσιακών μεθόδων και της ευθυγράμμισης με τις απαιτήσεις του σύγχρονου εκπαιδευτικού τοπίου. Το έργο έχει σχεδιαστεί με συγκεκριμένους, πρακτικούς στόχους, με πρόθεση να φέρει επανάσταση στον τρόπο με τον οποίο τα εκπαιδευτικά ιδρύματα διαχειρίζονται και χρησιμοποιούν τα δεδομένα παρακολούθησης. Τα βασικά αποτελέσματα που επιδιώκει να επιτύχει αυτό το έργο περιλαμβάνουν:

Αυτοματοποιημένη παρακολούθηση παρουσίας

- **Στόχος:**
  Η εφαρμογή ενός πλήρως αυτοματοποιημένου συστήματος που βελτιώνει τη διαδικασία καταγραφής παρουσίας, εξαλείφοντας την ανάγκη για χειροκίνητες ονομαστικές κλήσεις ή έντυπες εγγραφές.

- ✓ **Αποτέλεσμα:**
  Αυτός ο αυτοματισμός θα μειώσει σημαντικά τη χειρωνακτική παρέμβαση, ελαχιστοποιώντας το ανθρώπινο λάθος και εξασφαλίζοντας υψηλότερο βαθμό ακρίβειας στα αρχεία παρουσίας. Οι εκπαιδευτικοί θα είναι σε θέση να ανακατανείμουν χρόνο και πόρους από διοικητικά καθήκοντα σε δραστηριότητες διδασκαλίας και συμμετοχής των μαθητών.

Επεξεργασία δεδομένων σε πραγματικό χρόνο

- **Στόχος:**
  Η ανάπτυξη μιας λύσης που επεξεργάζεται τα δεδομένα παρουσίας σε πραγματικό χρόνο, επιτρέποντας την άμεση καταγραφή και ενημέρωση των απουσιολογίων καθώς οι φοιτητές εισέρχονται από την τάξη.

- ✓ **Αποτέλεσμα:**
  Η δυνατότητα επεξεργασίας δεδομένων σε πραγματικό χρόνο θα διασφαλίσει ότι τα αρχεία παρουσίας είναι πάντα ενημερωμένα και μπορούν να προσπελαστούν άμεσα από εκπαιδευτικούς και διαχειριστές. Αυτή η αμεσότητα θα διευκολύνει μια πιο δυναμική αλληλεπίδραση με τα δεδομένα συμμετοχής, επιτρέποντας την άμεση ανταπόκριση σε μοτίβα απουσιών ή καθυστερημένων αφίξεων.

Φιλική προς το χρήστη διεπαφή

- **Στόχος:**

Να δημιουργηθεί μια κατανοητή και προσβάσιμη διεπαφή χρήστη τόσο για τους φοιτητές όσο και για τους καθηγητές, διευκολύνοντας την αλληλεπίδραση με το σύστημα για τον έλεγχο των αρχείων παρουσίας, τη δημιουργία αναφορών και την ανάλυση τάσεων.

- ✓ **Αποτέλεσμα:**
  Μια φιλική προς το χρήστη διεπαφή θα μειώσει το εμπόδιο εισόδου για τους χρήστες του συστήματος, διασφαλίζοντας ότι η τεχνολογία χρησιμεύει για να ενισχύσει και όχι να περιπλέξει την εκπαιδευτική εμπειρία. Η ευκολία χρήσης στοχεύει επίσης σε υψηλότερα ποσοστά υιοθέτησης και μια πιο θετική στάση απέναντι στο σύστημα τόσο μεταξύ των φοιτητών όσο και μεταξύ των διδασκόντων.

Επεκτασιμότητα και ευελιξία

- ▪ **Στόχος:**
  Να διασφαλιστεί ότι το σύστημα είναι κλιμακούμενο, ικανό να φιλοξενήσει την ανάπτυξη των εκπαιδευτικών ιδρυμάτων και ευέλικτο, ικανό να προσαρμοστεί σε διάφορα εκπαιδευτικά περιβάλλοντα και τρόπους μάθησης (π.χ. αυτοπροσώπως, διαδικτυακά, υβριδικά).

- ✓ **Αποτέλεσμα:**
  Η επεκτασιμότητα και η ευελιξία θα εγγυηθούν ότι το σύστημα παραμένει αποτελεσματικό και αποδοτικό ανεξάρτητα από το μέγεθος της τάξης ή τις αλλαγές στην μορφής του. Αυτή η προσαρμοστικότητα θα επιτρέψει στα ιδρύματα να διατηρούν συνεπείς πρακτικές καταγραφής συμμετοχής, ακόμη και καθώς εξελίσσονται και αναπτύσσονται.

Η επίτευξη αυτών των στόχων στοχεύει να αντιμετωπίσει τις άμεσες ανεπάρκειες και ανακρίβειες που σχετίζονται με τις παραδοσιακές μεθόδους, ξεκλειδώνοντας νέες ευκαιρίες για την αξιοποίηση των δεδομένων συμμετοχής για την υποστήριξη των εκπαιδευτικών στόχων. Η επιτυχής υλοποίηση αυτού του έργου θα σηματοδοτήσει ένα σημαντικό βήμα μπροστά, στη χρήση της τεχνολογίας για την ενίσχυση της εκπαιδευτικής διοίκησης και της συμμετοχής των φοιτητών.

1.3 Εύρος και Περιορισμοί

Η ενότητα αυτή ορίζει το εύρος του έργου και εξετάζει τους περιορισμούς που αναμένεται να επηρεάσουν την εφαρμογή και τη λειτουργία του. Μέσα από την ανάλυση αυτών των δύο κρίσιμων πτυχών, καθορίζεται το πλαίσιο δράσης και αναδεικνύονται τα όρια και οι δυνατότητες του προτεινόμενου συστήματος. Το έργο αναλαμβάνει την ανάπτυξη ενός συστήματος παρουσιολόγιου φοιτητών, ενσωματώνοντας καινοτόμα εργαλεία και τεχνολογίες με στόχο την υλοποίηση ενός αποτελεσματικού και ευέλικτου μοντέλου.

1.3.1 Το Εύρος του Έργου

Παρακάτω θα αναλύσουμε τα διάφορα στοιχεία του έργου, καθένα από τα οποία διαδραματίζει κρίσιμο ρόλο στη συνολική λειτουργικότητά του.

- ➢ **Βάση δεδομένων Firestore:**
  Λειτουργεί ως κεντρική αποθήκη για όλα τα δεδομένα που σχετίζονται με τη συμμετοχή, τα προφίλ μαθητών, τα προγράμματα των τάξεων και άλλα στοιχεία που είναι απαραίτητα για να αναπτυχθεί το σύστημα. Η χρήση του Firestore, μιας πλατφόρμας **BaaS** διασφαλίζει ότι η υλοποίηση μπορεί να χειριστεί ενημερώσεις δεδομένων σε πραγματικό χρόνο και συγχρονισμό με τα διάφορα στοιχεία της, διευκολύνοντας την άμεση πρόσβαση σε αναλυτικά στοιχεία.

- **Google Apps Scripts – Google Sheets:**
  Χρησιμοποιούνται για τη δημιουργία δυναμικών υπολογιστικών φύλλων (**Google Sheets**) που αλληλεπιδρούν με τη βάση δεδομένων Firestore για την αλληλεπίδραση του καθηγητή με τα δεδομένα. Αυτοματοποιούν πολλές από τις διοικητικές εργασίες που σχετίζονται με την καταγραφή και αποθήκευση παρουσίας, όπως ο υπολογισμός των ποσοστών συμμετοχής, ο μέσος όρος του βαθμού των φοιτητών, ενώ επιπλέον ειδοποιεί τους εκπαιδευτικούς για μοτίβα απουσιών.

- **Raspberry Pi για σάρωση κώδικα QR:**
  Χρησιμεύει ως διεπαφή υλικού για τη φυσική καταγραφή της συμμετοχής των φοιτητών. Οι φοιτητές παρουσιάζουν κωδικούς QR, -που δημιουργούνται μέσω της εφαρμογής για κινητά Android του συστήματος, είτε τους παρέχονται από τους καθηγητές- τους οποίους το Raspberry Pi σαρώνει στην είσοδο της τάξης για να δηλώσει τη συμμετοχή τους.

  Με την σειρά τους οι καθηγητές έχουν την δυνατότητα να δημιουργούν το πρόγραμμα του μαθήματος δυναμικά σαρώνοντας τους δικούς τους κωδικούς QR αντίστοιχα. Αυτή η μέθοδος απλοποιεί τη διαδικασία, μειώνοντας την ανάγκη χρονοβόρων διαδικασιών εξατομίκευσης από τους καθηγητές.

- **Εφαρμογή Android για αλληλεπίδραση χρήστη:**
  Παρέχει μια διεπαφή για κινητά μέσω της οποίας οι φοιτητές μπορούν να δημιουργήσουν μοναδικούς κωδικούς QR για να πραγματοποιήσουν μια παρουσία. Για το ακαδημαϊκό ίδρυμα, η εφαρμογή προσφέρει λειτουργίες για τη διαχείριση των προγραμμάτων μαθημάτων, ενισχύοντας την προσβασιμότητα και την ευκολία διαχείρισης του συστήματος.

Στο σύνολο τους, αυτά τα στοιχεία προσπαθούν να δημιουργήσουν ένα ολοκληρωμένο σύστημα που αντιμετωπίζει τις βασικές προκλήσεις των υπαρχόντων μεθόδων, προσφέροντας βελτιώσεις στην ακρίβεια, την αποτελεσματικότητα και τη χρηστικότητα των δεδομένων παρουσίας.

1.3.2 Οι Περιορισμοί της Υλοποίησης

Παρά την ολοκληρωμένη προσέγγιση και την καινοτόμο χρήση της τεχνολογίας, το έργο αντιμετωπίζει ορισμένους περιορισμούς με την συγκεκριμένη προσέγγιση. Ενώ η επιλογή τεχνολογιών όπως το Firestore και το Raspberry Pi ενισχύει τις δυνατότητες του συστήματος, εισάγει επίσης εξαρτήσεις από τους υπάρχοντες περιορισμούς αυτών των πλατφορμών, όπως απαιτήσεις συνδεσιμότητας στο διαδίκτυο για συγχρονισμό δεδομένων σε πραγματικό χρόνο, εξαρτήσεις από βιβλιοθήκες και υπηρεσίες από τρίτους, καθώς και την αξιοπιστία του υλικού. Παρόλο που το σύστημα έχει σχεδιαστεί με γνώμονα την επεκτασιμότητα, πρακτικές δοκιμές υπό ποικίλες συνθήκες, όπως εξαιρετικά μεγάλα μεγέθη μαθητών ή εξαιρετικά μεγάλος όγκος ταυτόχρονων συναλλαγών δεδομένων, ενδέχεται να αποκαλύψουν απρόβλεπτα όρια επεκτασιμότητας. Το πεδίο εφαρμογής του έργου περιορίζεται φυσικά και από τους διαθέσιμους πόρους, συμπεριλαμβανομένων των αναπτυξιακών εργαλείων, του οικονομικού προϋπολογισμού και της ανθρώπινης συμβολής του πανεπιστήμιου στην υιοθέτηση και την περεταίρω ανάπτυξή του. Ομοίως, το χρονοδιάγραμμα του έργου περιορίζει το βάθος των δοκιμών που είναι απαραίτητες για τη βελτίωση της ανάπτυξης του συστήματος. Η διαδικασία ανάπτυξης του έργου περιλαμβάνει υποθέσεις που μπορεί να επηρεάσουν την εφαρμοσιμότητα και την απόδοσή του. Αυτές περιλαμβάνουν υποθέσεις σχετικά με την εξοικείωση του χρήστη με την τεχνολογία, τη σταθερότητα και την αξιοπιστία της υποδομής του διαδικτύου στο εκπαιδευτικό ίδρυμα καθώς και τη διαθεσιμότητα των εξαρτημάτων του υλικού.

Η αντιμετώπιση αυτών των περιορισμών και παραδοχών απαιτεί συνεχή αξιολόγηση και αναπροσαρμογή του συστήματος. Οι μελλοντικές επεκτάσεις του έργου μπορούν να επωφεληθούν από

την ανατροφοδότηση που θα συγκεντρωθεί κατά την αρχική ανάπτυξή του, επιτρέποντας βελτιώσεις που ενισχύουν περαιτέρω την αποτελεσματικότητα και τη χρηστικότητά του σε ένα ευρύτερο φάσμα εκπαιδευτικών πλαισίων.

1.4 Η Δομή της Διατριβής

Αυτή η διατριβή είναι δομημένη έτσι ώστε να παρέχει μια ολοκληρωμένη επισκόπηση της ανάπτυξης και εφαρμογής ενός συστήματος παρουσιολόγιου φοιτητών χρησιμοποιώντας ένα Raspberry Pi, περιγράφοντας λεπτομερώς τη συμβολή κάθε στοιχείου σε μια ενοποιημένη λύση. Η δομή έχει σχεδιαστεί για να καθοδηγήσει τον αναγνώστη από την αρχική ιδέα του έργου στην πρακτική του εφαρμογή, αναδεικνύοντας τις προκλήσεις και τα επιτεύγματα στην πορεία της υλοποίησής του.

Κεφάλαιο 2:

Επισκόπηση συστήματος και προκαταρκτικές έννοιες: Αυτό το κεφάλαιο παρουσιάζει τη συνολική αρχιτεκτονική του συστήματος ελέγχου παρουσίας φοιτητών και τις βασικές τεχνολογίες που αναπτύσσονται. Θέτει τις βάσεις για την κατανόηση του τρόπου με τον οποίο στοιχεία, όπως η βάση δεδομένων Firestore, τα Google Apps Scripts, το Raspberry Pi και η εφαρμογή Android λειτουργούν συντονισμένα για να αυτοματοποιήσουν και να βελτιώσουν την διαχείριση των δεδομένων.

Κεφάλαιο 3:

Βάση δεδομένων Firestore: Περιγράφει λεπτομερώς το σχεδιασμό, τη δομή και την υλοποίηση της βάσης δεδομένων Firestore στην καρδιά του συστήματος. Εξηγεί πώς το Firestore υποστηρίζει συγχρονισμό και αποθήκευση δεδομένων σε πραγματικό χρόνο, χρησιμεύοντας ως κεντρικό αποθετήριο για όλα τα δεδομένα που σχετίζονται με το έργο.

Κεφάλαιο 4:

Google Apps Scripts για Ακαδημαϊκή Διαχείριση: Εξηγεί τον ρόλο των Google Apps Scripts στην αυτοματοποίηση διαχειριστικών εργασιών που σχετίζονται με τη διαχείριση των παρουσιών. Αυτό το κεφάλαιο καλύπτει τον τρόπο αλληλεπίδρασης των διδασκόντων με τα Google Sheets και των G.A.S. με το Firestore για τη διαχείριση των δεδομένων και την απλοποίηση εργασιών που προηγουμένως ήταν μη αυτόματες και χρονοβόρες.

Κεφάλαιο 5:

Raspberry Pi για παρακολούθηση παρουσίας: Καλύπτει τις πρακτικές πτυχές της χρήσης του Raspberry Pi για σάρωση κώδικα QR, συμπεριλαμβανομένης της εγκατάστασης υλικού και της διαδικασίας ανάπτυξης λογισμικού. Αυτό το κεφάλαιο δείχνει πώς το Raspberry Pi χρησιμεύει ως η φυσική διεπαφή για τη λήψη δεδομένων και αποφάσεων σε πραγματικό χρόνο.

Κεφάλαιο 6:

Ανάπτυξη εφαρμογής Android: Περιγράφει την ανάπτυξη της εφαρμογής Android **EduEntry**, εστιάζοντας στο σχεδιασμό και τις λειτουργίες της με σκοπό να διευκολύνουν την αλληλεπίδραση του χρήστη με το σύστημα. Περιγράφει λεπτομερώς τον τρόπο με τον οποίο η εφαρμογή επιτρέπει στους χρήστες να συνδεθούν με το ακαδημαϊκό τους προφίλ και να δημιουργήσουν κωδικούς QR για την παρακολούθηση και διαχείριση των μαθημάτων.

Κεφάλαιο 7:

Συζήτηση και Συμπεράσματα: Συζητά την ενσωμάτωση και την αξιολόγηση της προσέγγισης που επιλέχθηκε για τον έλεγχο παρουσίας φοιτητών, αναφέροντας λεπτομερώς την ανάπτυξη και την απόδοση που μπορεί να έχει το σύστημα σε ένα εκπαιδευτικό περιβάλλον. Υπογραμμίζει την λειτουργία του συστήματος, τις προκλήσεις που αντιμετωπίστηκαν και τις μελλοντικές ενέργειες για τη βελτίωση της λειτουργικότητας και της εμπειρίας των χρηστών.

Αυτή η διατριβή αντιπροσωπεύει μια λεπτομερή διερεύνηση μιας πρακτικής υλοποίησης με σκοπό την βελτίωση κάποιων διαδικασιών στην εκπαίδευση. Μέσω της ανάπτυξης και της εφαρμογής ενός ολοκληρωμένου συστήματος παρακολούθησης της φοίτησης των φοιτητών, η εργασία αυτή προσπαθεί να αναδείξει τις δυνατότητες της τεχνολογίας με σκοπό να καλύψει τις συνεχώς αυξανόμενες εκπαιδευτικές ανάγκες και να ενισχύσει τη μαθησιακή εμπειρία. Οι αναγνώστες καλούνται να ακολουθήσουν την αφήγηση της καινοτομίας, τις προκλήσεις και τα μαθησιακά αποτελέσματα που χαρακτηρίζουν αυτό το έργο.

Κεφάλαιο 2ο: Επισκόπηση Συστήματος και Προκαταρκτικές Έννοιες

2.1 Επισκόπηση της Αρχιτεκτονικής του Συστήματος

Στο εξελισσόμενο τοπίο της εκπαίδευσης, όπου η αποτελεσματικότητα, η ακρίβεια και η λήψη αποφάσεων βάσει δεδομένων έχουν καταστεί υψίστης σημασίας, οι παραδοσιακές μέθοδοι παρακολούθησης της φοίτησης των φοιτητών έχουν μεγάλο περιθώριο βελτίωσης [17]. Η ανάγκη για ένα σύστημα που όχι μόνο απλοποιεί αλλά και ενισχύει τη διαδικασία συλλογής δεδομένων δεν ήταν ποτέ πιο εμφανής. Αυτή η ανάγκη οδήγησε στην ανάπτυξη ενός ολοκληρωμένου συστήματος παρακολούθησης και διαχείρισης παρουσίας, σχεδιασμένο για την αντιμετώπιση των προβλημάτων των συμβατικών προσεγγίσεων με μια σύγχρονη, τεχνολογικά καθοδηγούμενη λύση.

2.1.1 Εισαγωγή

Ο πρωταρχικός στόχος αυτού του έργου είναι να δημιουργήσει ένα σύστημα που αυτοματοποιεί ολόκληρη τη διαδικασία παρακολούθησης της συμμετοχής των φοιτητών στα μαθήματα, από τη στιγμή που ένας φοιτητής εισέρχεται στην τάξη μέχρι το σημείο όπου γίνεται η ανάλυση των δεδομένων συμμετοχής για να βγει κάποιο πόρισμα. Αυτό το σύστημα ενσωματώνει τεχνολογίες αιχμής, συμπεριλαμβανομένης μιας βάσης δεδομένων Firestore για ασφαλή και επεκτάσιμη αποθήκευση δεδομένων, τα Google Apps Scripts για αυτοματοποιημένες διοικητικές εργασίες στα Google Sheets, συσκευές Raspberry Pi για σάρωση κωδικών QR σε πραγματικό χρόνο και μια εφαρμογή Android για την αλληλεπίδραση του χρήστη. Μαζί, αυτά τα στοιχεία σχηματίζουν μια αρχιτεκτονική που επιτρέπει την επεξεργασία δεδομένων σε πραγματικό χρόνο, ελαχιστοποιεί τη χειροκίνητη παρέμβαση και μειώνει σημαντικά την πιθανότητα σφαλμάτων.

Αξιοποιώντας τις δυνατότητες αυτών των τεχνολογιών, το προτεινόμενο σύστημα προσπαθεί να προσφέρει μια καινοτόμο λύση στις πρακτικές προκλήσεις στον έλεγχο παρουσίας. Στόχος του είναι να παρέχει στα εκπαιδευτικά ιδρύματα ένα εργαλείο που όχι μόνο απλοποιεί τις διοικητικές διαδικασίες, αλλά προσφέρει επίσης πληροφορίες σχετικά με τα πρότυπα παρακολούθησης, ενδεχομένως ενημερώνοντας παρεμβάσεις που ενισχύουν τη συμμετοχή των μαθητών και την ακαδημαϊκή επιτυχία. Μέσω αυτής της ολοκληρωμένης προσέγγισης, το έργο προσπαθεί να θέσει ένα νέο πρότυπο για τον τρόπο διαχείρισης της φοίτησης σε σύγχρονα εκπαιδευτικά περιβάλλοντα, με την εφαρμογή της τεχνολογίας για τη βελτίωση των εκπαιδευτικών διαδικασιών.

2.1.2 Διάγραμμα της Αρχιτεκτονικής του Συστήματος

Ενσωματωμένο παρακάτω είναι το διάγραμμα αρχιτεκτονικής του συστήματος που αναδεικνύει το περίπλοκο πλαίσιο του συστήματος παρακολούθησης και διαχείρισης παρουσίας. Αυτό το οπτικό βοήθημα έχει σχεδιαστεί για να προσφέρει μια σαφή, με μια ματιά κατανόηση του τρόπου με τον οποίο τα διάφορα στοιχεία του συστήματος διασυνδέονται και αλληλεπιδρούν μεταξύ τους.

Στην καρδιά της αρχιτεκτονικής του συστήματος πάνω αριστερά, βρίσκεται η βάση δεδομένων Firestore, ο κεντρικός κόμβος που κρατά το σύστημα ενωμένο. Απεικονίζεται ως το θεμελιώδες στρώμα, μια απόδειξη του ρόλου της ως την κύρια αποθήκη δεδομένων.

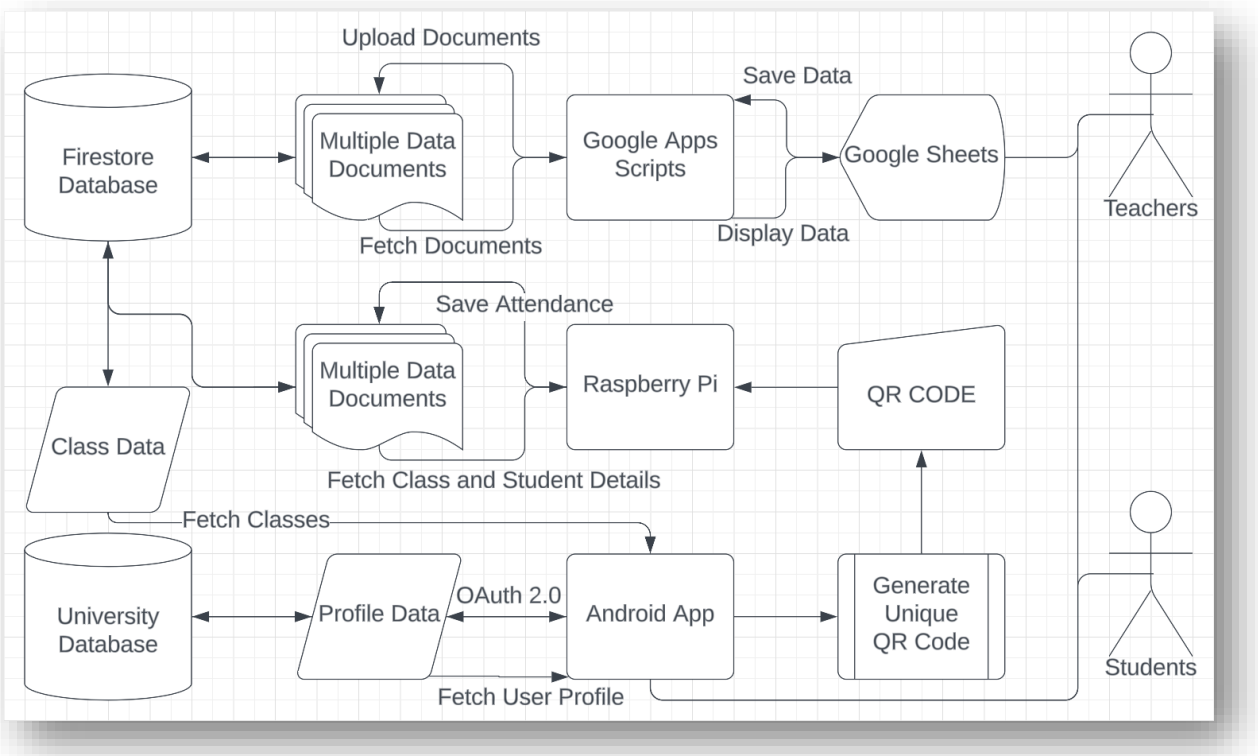
Δίπλα σε αυτή, απεικονίζονται τα Google Apps Scripts, παρουσιάζοντας την αλληλεπίδρασή τους με τη βάση δεδομένων Firestore και τα Google Sheets. Αυτή η αμφίδρομη ροή δεδομένων υποδηλώνει τη διπλή ικανότητα των Apps Scripts να λαμβάνουν και να αναμεταδίδουν δεδομένα ανάμεσα στα δύο στοιχεία, απαραίτητη για την ομαλή διαχείριση των διοικητικών καθηκόντων.

Ακριβώς από κάτω, βλέπουμε το Raspberry Pi να αλληλεπιδρά με τον κώδικα QR. Αντιπροσωπεύει τον σκοπό της συσκευής στη φυσική σάρωση κωδικών QR, η οποία είναι κρίσιμη για την καταγραφή της παρουσίας σε πραγματικό χρόνο. Η κατευθυντική γραμμή που το συνδέει με τη βάση δεδομένων Firestore υπογραμμίζει τη διαδρομή των δεδομένων για την λήψη αποφάσεων, στην αποθήκευση των παρουσιών.

Αντικατοπτρίζοντας το Raspberry Pi στην άλλη πλευρά είναι η εφαρμογή Android, που συνδέεται με τη σειρά της στην βάση δεδομένων Firestore και την βάση δεδομένων του ιδρύματος επιτρέποντας στους χρήστες να έχουν πρόσβαση και να αλληλεπιδρούν με τα δεδομένα συμμετοχής τους. Τα βέλη της επιπλέον απεικονίζουν τον ρόλο της εφαρμογής στη δημιουργία κωδικών QR.

Δεν πρέπει επίσης να παραλειφθούν τα εικονίδια χρήστη που συνδέονται με την εφαρμογή Android και τα Google Sheets, συμβολίζοντας τους φοιτητές και τους καθηγητές που αλληλεπιδρούν με το σύστημα. Αυτό το σημείο επαφής ενισχύει τον χαρακτήρα του συστήματος με επίκεντρο τον χρήστη, όπου η ευκολία χρήσης και η προσβασιμότητα είναι υψίστης σημασίας.

Μαζί, αυτά τα στοιχεία αποτελούν τον βασικό σκελετό του συστήματός, αναδεικνύοντας την συνεργασία μεταξύ λογισμικού και υλικού. Αυτό το διάγραμμα είναι μια αναπαράσταση των τεχνολογιών που χρησιμοποιήθηκαν για την ανάπτυξη του έργου με στόχο την καινοτομία στις εκπαιδευτικές μεθόδους.



Σχήμα 2.1: Διάγραμμα Αρχιτεκτονικής Συστήματος

2.1.3 Περιγραφή των Διασυνδέσεων Μεταξύ των Στοιχείων

Το σύστημα ελέγχου παρουσίας φοιτητών είναι ένα εναρμονισμένο μείγμα διαφόρων στοιχείων, καθένα από τα οποία διαδραματίζει έναν μοναδικό ρόλο ενώ αλληλεπιδρά ομαλά με τα υπόλοιπα. Αυτή η συνέργεια εξασφαλίζει τη συνολική αποτελεσματικότητα, ακρίβεια και φιλικότητα του συστήματος προς τον χρήστη. Ακολουθεί μια λεπτομερής εξήγηση του τρόπου με τον οποίο κάθε στοιχείο διασυνδέεται και αλληλεπιδρά μέσα στο σύστημα:

Βάση δεδομένων Firestore: "Ο κεντρικός κόμβος του συστήματος"

Στην καρδιά του συστήματος βρίσκεται η βάση δεδομένων Firestore, που χρησιμεύει ως κεντρικός κόμβος για όλη τη διαχείριση των δεδομένων. Αποθηκεύει ολοκληρωμένα αρχεία, συμπεριλαμβανομένων των προφίλ των φοιτητών, των προγραμμάτων τάξεων και των λεπτομερών αρχείων παρουσιών. Αυτή η μη σχεσιακή βάση δεδομένων (**NoSQL)** που βασίζεται στο **cloud** επιτρέπει ενημερώσεις σε πραγματικό χρόνο και συγχρονισμό σε όλο το σύστημα, διασφαλίζοντας ότι όλα τα στοιχεία έχουν άμεση πρόσβαση στα πιο πρόσφατα δεδομένα. Η κλιμακούμενη και ευέλικτη αρχιτεκτονική της υποστηρίζει τη δυναμική φύση των εκπαιδευτικών περιβαλλόντων, κάνοντας εύκολες τις αλλαγές και την περεταίρω ανάπτυξη.

Google Apps Scripts για τα Google Sheets: "Αυτοματοποίηση διαχειριστικών εργασιών"

Τα Google Apps Scripts γεφυρώνουν το χάσμα μεταξύ των ακατέργαστων δεδομένων που αποθηκεύονται και των αξιοποιήσιμων πληροφοριών και πορισμάτων. Διασυνδέονται απευθείας με τη βάση δεδομένων Firestore, εξάγοντας σχετικά δεδομένα για περαιτέρω επεξεργασία. Αυτοματοποιούν μια ποικιλία διοικητικών εργασιών **backend**, όπως την καταχώρηση των βαθμών των φοιτητών, τον υπολογισμό των συνολικών ποσοστών συμμετοχής, ακόμη και την επισήμανση μοτίβων συμμετοχής που μπορεί να απαιτούν προσοχή από τους καθηγητές. Διευκολύνουν επίσης τη δυναμική δημιουργία και ενημέρωση των Google Sheets, όπου τα δεδομένα παρουσιάζονται σε προσβάσιμη μορφή για εκπαιδευτικούς και διαχειριστές. Αυτός ο αυτοματισμός μειώνει σημαντικά τον χειροκίνητο χειρισμό των δεδομένων, ενισχύοντας την αποτελεσματικότητα του συστήματος ενώ παράλληλα μειώνει την δυσκολία υιοθέτησης παρέχοντας στους εκπαιδευτικούς ένα εργαλείο που είναι γνώριμο προσθέτοντας το εκτεταμένες λειτουργείες.

Raspberry Pi: "Η φυσική διεπαφή για τη συλλογή δεδομένων"

Οι συσκευές Raspberry Pi αναπτύσσονται ως φυσική διεπαφή για τη συλλογή δεδομένων παρουσίας σε πραγματικό χρόνο. Εξοπλισμένα με μια φωτογραφική μηχανή επιτρέπουν την δυνατότητα σάρωσης των κωδικών QR και χρησιμεύουν ως σημείο εισόδου για τα δεδομένα παρουσίας στο σύστημα. Όταν ένας φοιτητής παρουσιάζει έναν κωδικό QR κατά την είσοδό του στην τάξη, το Raspberry Pi τον σαρώνει, τον αποκωδικοποιεί και τον αποκρυπτογραφεί, εξάγοντας πληροφορίες για την ταυτότητα του. Αυτά τα δεδομένα στη συνέχεια επεξεργάζονται και μεταδίδονται αμέσως στη βάση δεδομένων Firestore, ενημερώνοντας το αρχείο παρακολούθησης του φοιτητή σε πραγματικό χρόνο. Αυτή η ρύθμιση όχι μόνο ελαχιστοποιεί την καθυστέρηση αλλά και εξαλείφει τα σφάλματα που προκύπτουν από την μη αυτόματη εισαγωγή παρουσίας, εξασφαλίζοντας υψηλή ακρίβεια.

Εφαρμογή Android: "Το **frontend** για την αλληλεπίδραση χρήστη"

Η εφαρμογή Android αποτελεί την κύρια διεπαφή για τους τελικούς χρήστες, δηλαδή τους φοιτητές αλλά και τους καθηγητές. Για τους φοιτητές, η εφαρμογή προσφέρει λειτουργίες όπως αυθεντικοποίηση με τους κωδικούς του υδρώματος και δημιουργία ενός κώδικα QR για την παρακολούθηση των μαθημάτων, ενώ τα μέλη του διδακτικού προσωπικού χρησιμοποιούν την εφαρμογή για να

διαχειρίζονται και να εξατομικεύουν τα μαθήματα τους. Η άμεση αλληλεπίδραση με τη βάση δεδομένων Firestore και με τη βάσης δεδομένων του ιδρύματος μέσω της εφαρμογής διασφαλίζει ότι όλοι οι χρήστες έχουν άμεση πρόσβαση, διευκολύνοντας την αποτελεσματική επικοινωνία και διαχείριση εντός του εκπαιδευτικού οικοσυστήματος.

2.2 Προκαταρκτικές Έννοιες

Το σύστημα αξιοποιεί μια σειρά τεχνολογιών και πλατφορμών, καθεμία από τις οποίες επιλέγεται για τις μοναδικές δυνατότητές της, για την αντιμετώπιση συγκεκριμένων πτυχών και απαιτήσεων του. Κεντρικό στοιχείο της λειτουργίας του συστήματος είναι η βάση δεδομένων Firestore, γύρω από την οποία διαμορφώνονται οι λειτουργίες των άλλων στοιχείων.

2.2.1 Βάση Δεδομένων Firestore

Η βάση δεδομένων Firestore, ένα προϊόν της πλατφόρμας Firebase της Google, χρησιμεύει ως η θεμελιώδης ραχοκοκαλιά του συστήματος ελέγχου παρουσίας. Ο πρωταρχικός ρόλος της είναι να αποθηκεύει και να διαχειρίζεται όλα τα δεδομένα του συστήματος. Η φύση του Firestore που βασίζεται στο cloud επιτρέπει την εύκολη πρόσβαση και τον χειρισμό των δεδομένων σε όλο το σύστημα, διασφαλίζοντας ότι κάθε στοιχείο, από το backend έως τις εφαρμογές του τελικού χρήστη, λειτουργεί με τις πιο πρόσφατες πληροφορίες.

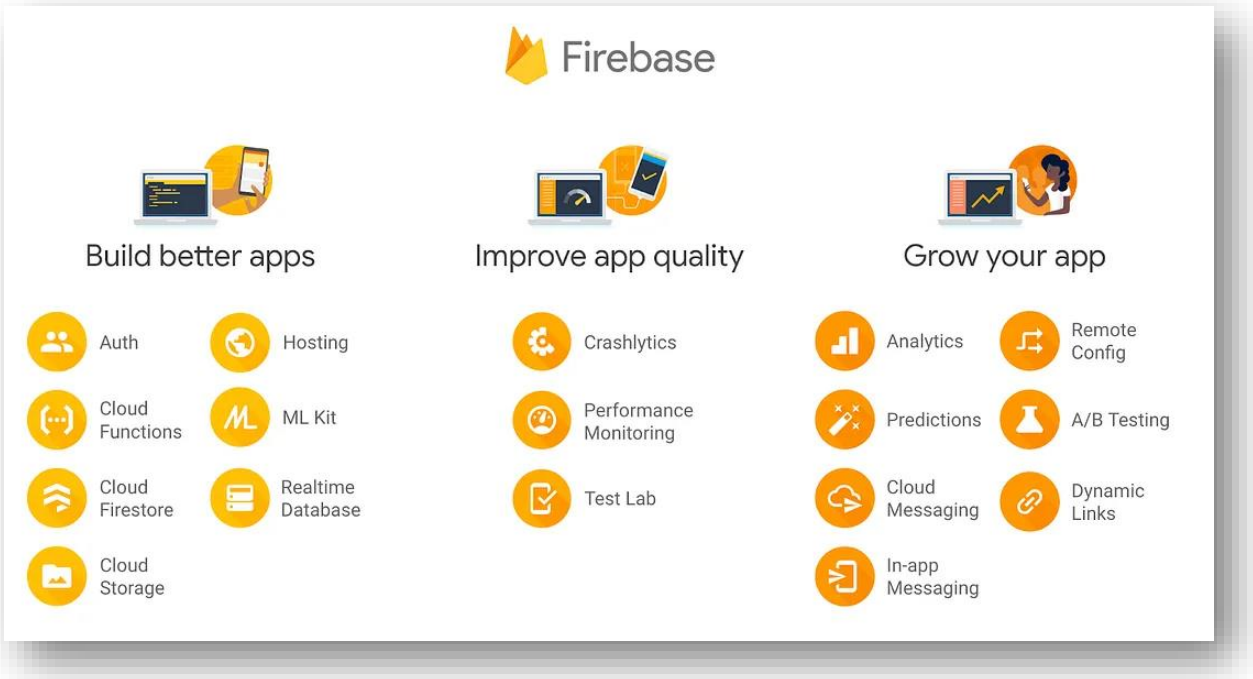
Δυνατότητες του Firestore:

- **Συγχρονισμός δεδομένων σε πραγματικό χρόνο:**
  Ένα από τα πιο συναρπαστικά χαρακτηριστικά του Firestore είναι η ικανότητά του να συγχρονίζει δεδομένα σε πραγματικό χρόνο. Αυτό σημαίνει ότι οποιαδήποτε αλλαγή στη βάση δεδομένων αντικατοπτρίζεται αμέσως στις συνδεδεμένες εφαρμογές, είτε πρόκειται για τις συσκευές Raspberry Pi στην είσοδο των τάξεων, είτε για την εφαρμογή Android στα χέρια των μαθητών, είτε για τα Google Sheets για την προβολή των δεδομένων στους καθηγητές. Αυτή η δυνατότητα διασφαλίζει ότι τα δεδομένα είναι πάντα ενημερωμένα, διευκολύνοντας τις άμεσες απαντήσεις σε ερωτήσεις και ενέργειες που πραγματοποιούνται στο σύστημα.

- **Επεκτασιμότητα:**
  Το Firestore έχει σχεδιαστεί για κλιμάκωση, μπορεί να χειριστεί μια πληθώρα σεναρίων, από μικρά σύνολα δεδομένων έως εφαρμογές με εκατομμύρια χρήστες. Αυτή η επεκτασιμότητα είναι ζωτικής σημασίας για το σύστημα καταγραφής συμμετοχής, καθώς μπορεί να φιλοξενεί διάφορα μεγέθη εκπαιδευτικών ιδρυμάτων. Η ικανότητα κλιμάκωσης σημαίνει ότι το σύστημα μπορεί να αναπτυχθεί ανάλογα με τις ανάγκες του ιδρύματος, προσθέτοντας περισσότερες τάξεις και φοιτητές.

- **Ευέλικτη μοντελοποίηση δεδομένων:**
  Το Firestore επιτρέπει την ευέλικτη μοντελοποίηση δεδομένων, η οποία είναι ιδιαίτερα επωφελής για τη διαχείριση των ποικίλων και δυναμικών δεδομένων που εμπλέκονται σε ένα ακαδημαϊκό περιβάλλον. Το σύστημα μπορεί εύκολα να προσαρμοστεί σε αλλαγές στα προγράμματα τάξεων, στις εγγραφές μαθητών και να προσθέσει άλλα χαρακτηριστικά χωρίς σημαντική αναδιαμόρφωση.

- **Ασφάλεια και αξιοπιστία:**
  Με ισχυρά ενσωματωμένα χαρακτηριστικά ασφαλείας και τακτικά αντίγραφα ασφαλείας, το Firestore διασφαλίζει ότι τα ευαίσθητα εκπαιδευτικά δεδομένα αποθηκεύονται με ασφάλεια και προστατεύονται από τη μη εξουσιοδοτημένη πρόσβαση και την απώλεια δεδομένων. Αυτή η ασφάλεια είναι υψίστης σημασίας για τη διατήρηση της εμπιστευτικότητας και της ακεραιότητας των αρχείων παρακολούθησης των φοιτητών και των σχετικών τους πληροφοριών.



Σχήμα 2.2: Οι Δυνατότητες του Firebase [2]

Αξιοποιώντας αυτές τις δυνατότητες, το Firestore αποθηκεύει τα δεδομένα, διαδραματίζοντας επίσης κεντρικό ρόλο στην αρχιτεκτονική του συστήματος. Διευκολύνει ένα ενοποιημένο λειτουργικό πλαίσιο όπου διασφαλίζεται η διαθεσιμότητα και η ακεραιότητα των δεδομένων σε πραγματικό χρόνο, επιτρέποντας την αποτελεσματική και ακριβή καταγραφή της παρουσίας. Η κλιμακούμενη και ασφαλής υποδομή του υποστηρίζει τη μακροπρόθεσμη βιωσιμότητα και προσαρμοστικότητα του συστήματος στις εξελισσόμενες εκπαιδευτικές ανάγκες.

Συνοπτικά, η βάση δεδομένων Firestore δεν είναι απλώς μια λύση αποθήκευσης, αλλά μια ισχυρή πλατφόρμα που επιτρέπει την επεκτάσιμη και ασφαλή διαχείριση δεδομένων σε πραγματικό χρόνο. Η ενσωμάτωσή του στο σύστημα ελέγχου παρουσίας βοηθάει στην επίτευξη των στόχων του έργου για τη βελτίωση της αποτελεσματικότητας και της ακρίβειας στην εκπαιδευτική διοίκηση.

2.2.2 Google Apps Scripts

Τα Google Apps Scripts είναι μια ευέλικτη πλατφόρμα δέσμης ενεργειών που και αυτή με την σειρά της βασίζεται στο cloud και επιτρέπει την αυτοματοποίηση εργασιών στο Google Workspace, συμπεριλαμβανομένων των Google Docs, των Google Sheets, των Google Forms και του Google Drive. Χρησιμεύει ως ένα ισχυρό εργαλείο για τη δημιουργία προσαρμοσμένων λειτουργιών, την αυτοματοποίηση ροών εργασίας και την ενσωμάτωση διαφόρων υπηρεσιών στο οικοσύστημα της Google. Στο πλαίσιο του συστήματος ελέγχου παρουσίας φοιτητών, στα Google Apps Scripts (G.A.S.) έχει ανατεθεί η επεξεργασία και η παρουσίαση των ακατέργαστων δεδομένων που είναι αποθηκευμένα στο Firestore μετατρέποντας τα σε αξιοποιήσιμες πληροφορίες προσβάσιμες μέσω των Google Sheets.

Η βασική τους χρήση είναι να αλληλεπιδράσουν δυναμικά με το Firestore για να λάβουν ενημερωμένα αρχεία παρουσίας, να επεξεργαστούν αυτά τα δεδομένα (π.χ. υπολογισμός ποσοστών συμμετοχής, εντοπισμός μοτίβων απουσιών, υπολογισμός μέσω ορού βαθμού) και στη συνέχεια να δημιουργήσουν δυναμικά φύλλα και να τα συμπληρώσουν με τις επεξεργασμένες πληροφορίες. Αυτός ο αυτοματισμός έχει σκοπό να βελτιώσει σημαντικά τις διαδικασίες αναφοράς και ανάλυσης των δεδομένων, προσπαθώντας να διευκολύνει τους εκπαιδευτικούς διαχειριστές να έχουν πρόσβαση, να ερμηνεύουν και να ενεργούν βάσει επεξεργασμένων πληροφοριών.

Παρακάτω θα αναφέρουμε κάποιες από τις πρακτικές εφαρμογές που μπορούν να υλοποιηθούν χρησιμοποιώντας τα Google Apps Scripts για τα εκπαιδευτικά διοικητικά καθήκοντα:

- **Αυτοματοποιημένες αναφορές:**
  Τα G.A.S. μπορούν να ρυθμιστούν ώστε να δημιουργούν αυτόματα εβδομαδιαίες ή μηνιαίες αναφορές παρουσίας, συνδυάζοντας δεδομένα όπως η συνολική συμμετοχή, τα περιστατικά καθυστέρησης και τα μοτίβα απουσιών. Αυτές οι αναφορές μπορούν να κοινοποιηθούν στους καθηγητές και τους διαχειριστές, παρέχοντάς τους πληροφορίες σχετικά με την συμμετοχή των μαθητών και τη εξέλιξη του μαθήματος μέσα στο ακαδημαϊκό εξάμηνο.

- **Ειδοποιήσεις σε πραγματικό χρόνο:**
  Έχουν την δυνατότητα να διαμορφωθούν έτσι ώστε να στέλνουν ειδοποιήσεις σε πραγματικό χρόνο σε εκπαιδευτικούς ή διοικητικό προσωπικό με βάση συγκεκριμένα εναύσματα. Για παράδειγμα, εάν η συμμετοχή ενός φοιτητή πέσει κάτω από ένα ορισμένο όριο, ένα σενάριο μπορεί να στείλει αυτόματα μια ειδοποίηση μέσω ηλεκτρονικού ταχυδρομείου στο ενδιαφερόμενο μέλος ΔΕΠ, διευκολύνοντας την έγκαιρη παρέμβαση του σε αυτό το συμβάν.

- **Ανάλυση και οπτικοποίηση δεδομένων:**
  Πέρα από τις βασικές αναφορές, τα G.A.S. μπορούν επίσης να χρησιμοποιηθούν για την εκτέλεση πιο σύνθετης ανάλυσης δεδομένων και τη απεικόνιση τους απευθείας στα Google Sheets. Αυτό περιλαμβάνει την ανάλυση τάσεων κατά τη διάρκεια του εξαμήνου, σύγκριση των ποσοστών συμμετοχής σε διαφορετικές τάξεις ή τμήματα και προσδιορισμό συσχετισμών μεταξύ φοίτησης και ακαδημαϊκών επιδόσεων.

- **Δημιουργία προσαρμοσμένου πίνακα ελέγχου:**
  Για μια πιο διαδραστική προσέγγιση, τα G.A.S. μπορούν να βοηθήσουν στη δημιουργία προσαρμοσμένων πινάκων ελέγχου στα Υπολογιστικά φύλλα. Αυτά τα εργαλεία μπορούν να παρέχουν στους εκπαιδευτικούς και τους διαχειριστές μια φιλική προς το χρήστη διεπαφή για την παρακολούθηση των δεδομένων παρουσίας, το φιλτράρισμα πληροφοριών βάσει διαφόρων κριτηρίων (π.χ. εύρος ημερομηνιών, τάξεις, φοιτητές) ακόμη και την αυτοματοποίηση κοινών εργασιών με προσαρμοσμένες επιλογές μενού και κουμπιά.

Η ενσωμάτωση των Google Apps Scripts στο σύστημα ελέγχου παρουσίας αποτελεί παράδειγμα της πρακτικής εφαρμογής των τεχνολογιών αυτοματισμού και επεξεργασίας δεδομένων στην εκπαιδευτική διοίκηση. Αυτοματοποιώντας επαναλαμβανόμενες εργασίες, προσπαθώντας να βελτιώσουν την προσβασιμότητα στα δεδομένων και διευκολύνοντας την εις βάθος ανάλυση τους, τα G.A.S. προσπαθούν να συμβάλλουν σημαντικά στην αποδοτικότητα και την αποτελεσματικότητα του συστήματος, επιτρέποντας στους εκπαιδευτικούς να επικεντρωθούν ακόμα περισσότερο στη συμμετοχή των φοιτητών και λιγότερο στα επαναλαμβανόμενα διοικητικά τους καθήκοντα.

2.2.3 Raspberry Pi

Το Raspberry Pi είναι μια συμπαγής, προσιτή και εξαιρετικά ευέλικτη υπολογιστική συσκευή που έχει συγκεντρώσει ευρεία αναγνώριση και υιοθέτηση σε διάφορους τομείς, από εκπαιδευτικά έργα έως βιομηχανικές εφαρμογές. Το μικρό του μέγεθος, η χαμηλή κατανάλωση ενέργειας και οι ισχυρές δυνατότητες επεξεργασίας, το καθιστούν ιδανική επιλογή για ένα ευρύ φάσμα υπολογιστικών εργασιών, συμπεριλαμβανομένης της χρήσης του σε καινοτόμα έργα που γεφυρώνουν τον ψηφιακό και τον φυσικό κόσμο.

Ένα από τα βασικά χαρακτηριστικά του Raspberry Pi είναι η ευελιξία του. Εξοπλισμένο με ακίδες εισόδου/εξόδου γενικής χρήσης (GPIO), θύρες USB και επιλογές ασύρματης συνδεσιμότητας, μπορεί να αλληλεπιδράσει με μια ποικιλία περιφερειακών συσκευών και αισθητήρων, επιτρέποντάς του να προσαρμοστεί σε αμέτρητες εφαρμογές. Η σχέση κόστους-αποτελεσματικότητας προσθέτει περαιτέρω αξία στην ελκυστικότητά του, παρέχοντας μια ισχυρή υπολογιστική πλατφόρμα σε τιμή που είναι προσβάσιμη για εκπαιδευτικά ιδρύματα και φοιτητές σε σχέση με άλλες επιλογές.

Παρακάτω θα αναλύσουμε την εφαρμογή του Raspberry Pi στο σύστημα ελέγχου παρουσιών:

- **Σάρωση QR Code για καταγραφή της παρουσίας:**
  Στο πλαίσιο του συστήματος ελέγχου παρουσίας φοιτητών, το Raspberry Pi αναπτύσσεται ως η κύρια διεπαφή για τη συλλογή φυσικών δεδομένων, συγκεκριμένα τη σάρωση κωδικών QR που παρουσιάζονται από τους φοιτητές καθώς εισέρχονται στην τάξη. Κάθε κωδικός QR περιέχει κωδικοποιημένες πληροφορίες σχετικά με τον φοιτητή και τη συνεδρία της τάξης, τις οποίες το Raspberry Pi έχει την δυνατότητα να αποκωδικοποιεί και να αποκρυπτογραφεί κατά τη σάρωση.

- **Ενημερώσεις σε πραγματικό χρόνο στη βάση δεδομένων Firestore:**
  Μετά την επιτυχή σάρωση και αποκωδικοποίηση ενός κωδικού QR, το Raspberry Pi κοινοποιεί αμέσως αυτά τα δεδομένα παρουσίας στη βάση δεδομένων Firestore. Αυτός ο άμεσος μηχανισμός ενημέρωσης σε πραγματικό χρόνο διασφαλίζει ότι τα αρχεία παρουσίας αντικατοπτρίζονται άμεσα στο σύστημα, εξαλείφοντας τις καθυστερήσεις και μειώνοντας την πιθανότητα σφαλμάτων ετεροχρονισμένης εισαγωγής δεδομένων.

- **Ρύθμιση υλικού και ενσωμάτωση λογισμικού:**
  Η ρύθμιση του Raspberry Pi για αυτήν την εφαρμογή περιλαμβάνει την ενσωμάτωση ενός σαρωτή κωδικών QR, είτε μέσω συνδεδεμένου σαρωτή USB είτε μέσω μιας μονάδας φωτογραφικής μηχανής ικανής να συλλεγεί εικόνες σε διαφορά περιβάλλοντα φωτισμού σε υψηλή ευκρίνεια. Το προσαρμοσμένο λογισμικό που εκτελείται στο Raspberry Pi χειρίζεται τη διαδικασία σάρωσης, αποκωδικοποιεί και αποκρυπτογραφεί τους κωδικούς QR και διαχειρίζεται την επικοινωνία με τη βάση δεδομένων Firestore μέσω σύνδεσης στο διαδίκτυο. Αυτό το λογισμικό αναπτύχθηκε χρησιμοποιώντας Python, αξιοποιώντας υπάρχουσες βιβλιοθήκες για επεξεργασία κώδικα QR και ενσωμάτωση του Firebase, αποδεικνύοντας έτσι τη συμβατότητα του Raspberry Pi με ευρέως χρησιμοποιούμενες γλώσσες και εργαλεία προγραμματισμού.

- **Πλεονεκτήματα για εκπαιδευτικά περιβάλλοντα:**
  Το Raspberry Pi μπορεί να προσφέρει πολλά πλεονεκτήματα σε ένα εκπαιδευτικό περιβάλλον. Το μικρό του μέγεθος επιτρέπει την εύκολη ενσωμάτωσή του σε διαφορά σημεία μιας τάξης. Είναι μια εξαιρετική πλατφόρμα για τη διδασκαλία, καθώς υποστηρίζει πολλές γλώσσες προγραμματισμού, ενθαρρύνοντας τους φοιτητές να εφαρμόσουν τις δεξιότητές τους σε πρακτικά σενάρια πραγματικού κόσμου, καθιστώντας τη μάθηση πιο διαδραστική και πολυδιάστατη.

Συνοπτικά, ο ρόλος του Raspberry Pi στο σύστημα ελέγχου παρουσίας αποτελεί παράδειγμα της χρησιμότητάς του ως γέφυρα μεταξύ φυσικής συλλογής δεδομένων και ψηφιακής επεξεργασίας δεδομένων. Η εφαρμογή του για τη σάρωση κωδικών QR και η ενημέρωση δεδομένων παρουσίας σε πραγματικό χρόνο δείχνει πώς μπορούν να αξιοποιηθούν οικονομικά, αποδοτικές και ευέλικτες υπολογιστικές συσκευές για την επίλυση πρακτικών προβλημάτων στην εκπαίδευση.

2.2.4 Εφαρμογή Android

Η εφαρμογή Android χρησιμεύει ως μια φιλική προς το χρήστη διεπαφή τόσο για φοιτητές όσο και για καθηγητές στην υλοποίηση του συστήματος μας. Σχεδιασμένη με έμφαση στην ευκολία χρήσης και την προσβασιμότητα, είναι ένα κρίσιμο στοιχείο που επιτρέπει στους χρήστες να αλληλεπιδρούν απευθείας με το σύστημα. Η ανάπτυξή του καθοδηγείται από τις αρχές του Clean Architecture και της λειτουργικότητας, διασφαλίζοντας την ευκολία στην επέκταση της και ότι οι χρήστες μπορούν να πλοηγηθούν στην εφαρμογή χωρίς κόπο για να έχουν πρόσβαση στις δυνατότητες που χρειάζονται.

Σχεδιασμός και λειτουργίες:

- **Έλεγχος ταυτότητας σύνδεσης:**
  Ο ασφαλής έλεγχος ταυτότητας σύνδεσης είναι ένα θεμελιώδες χαρακτηριστικό της εφαρμογής, διασφαλίζοντας ότι η πρόσβαση σε ευαίσθητες πληροφορίες περιορίζεται μόνο σε εξουσιοδοτημένους χρήστες. Οι φοιτητές και οι καθηγητές μπορούν να συνδεθούν χρησιμοποιώντας τα διαπιστευτήρια του ιδρύματός, με το σύστημα να υποστηρίζει ασφαλείς μηχανισμούς ελέγχου ταυτότητας για την προστασία των ταυτοτήτων και των δεδομένων των χρηστών. Αυτή η διαδικασία ελέγχου ταυτότητας είναι η πύλη για εξατομικευμένες εμπειρίες εντός της εφαρμογής, παρέχοντας στους χρήστες πρόσβαση σε πληροφορίες σχετικές με τους ρόλους και τις ανάγκες τους.
- **Δημιουργία κώδικα QR:**
  Μία από τις βασικές λειτουργίες της εφαρμογής για τους φοιτητές είναι η δημιουργία εξατομικευμένων κωδικών QR που αντιπροσωπεύουν τη συμμετοχή τους στα μαθήματα. Αυτοί οι κωδικοί QR, δημιουργούνται δυναμικά εντός της εφαρμογής, κωδικοποιούν πληροφορίες ειδικά για τον φοιτητή και τη συνεδρία τάξης που παρακολουθεί και θέλει να δηλώσει παρουσία. Οι φοιτητές παρουσιάζουν αυτούς τους κωδικούς QR για σάρωση από τις συσκευές Raspberry Pi στις εισόδους των τάξεων, διευκολύνοντας μια ομαλή και αυτοματοποιημένη διαδικασία κατάθεσης παρουσίας.
- **Φιλική προς το χρήστη διεπαφή:**
  Η διεπαφή της εφαρμογής έχει σχεδιαστεί για να είναι εύκολη στην πλοήγηση της, ελαχιστοποιώντας την καμπύλη εκμάθησης για νέους χρήστες και βελτιώνοντας τη συνολική εμπειρία χρήστη. Η προσοχή στις λεπτομέρειες σχεδίασης, όπως η λογική οργάνωση των πληροφοριών και η σαφής επισήμανση τους, διασφαλίζει ότι οι χρήστες μπορούν να βρουν και να χρησιμοποιήσουν αποτελεσματικά τις δυνατότητες της εφαρμογής, ανεξάρτητα από την τεχνολογική τους επάρκεια.

Η εφαρμογή Android βελτιώνει σημαντικά τη χρηστικότητα και την προσβασιμότητα του συστήματος ελέγχου παρουσίας για τους τελικούς χρήστες. Παρέχοντας μια άμεση, κινητή διεπαφή στο σύστημα. Επιπλέον, η εφαρμογή μπορεί να αποκτήσει επιπλέον χαρακτηριστικά από το σύστημα όπως διασύνδεση με τη βάση δεδομένων Firestore για ανάκτηση δεδομένων παρουσίας και ενημερώσεις. Αυτή η δυνατότητα, σε συνδυασμό με τον σχεδιασμό της εφαρμογής με επίκεντρο τον χρήστη, την καθιστά ένα εργαλείο που μπορεί να εκσυγχρονίσει και να βελτιώσει την παρακολούθηση παρουσίας στο εκπαιδευτικό περιβάλλον.

Συμπερασματικά, η εφαρμογή Android όχι μόνο παρέχει βασικές λειτουργίες που απαιτούνται για την αποτελεσματική λειτουργία του συστήματος, αλλά αποτελεί επίσης παράδειγμα για το πώς μπορεί να αξιοποιηθεί η τεχνολογία για την ενίσχυση της εκπαιδευτικής διοίκησης και την υποστήριξη της ακαδημαϊκής κοινότητας.

2.3 Συμπεράσματα

Σε αυτό το κεφάλαιο, παρουσιάσαμε τις βασικές τεχνολογίες και πλατφόρμες που υποστηρίζουν το σύστημα ελέγχου παρουσίας φοιτητών, το καθένα επιλεγμένο για τα μοναδικά πλεονεκτήματα και τις δυνατότητές του. Η βάση δεδομένων Firestore χρησιμεύει ως κεντρικό αποθετήριο για όλα τα δεδομένα, παρέχοντας ενημερώσεις σε πραγματικό χρόνο και κλιμακούμενες λύσεις αποθήκευσης. Τα Google Apps Scripts αυτοματοποιούν πολύπλοκες διοικητικές εργασίες, γεφυρώνοντας το χάσμα μεταξύ ανεπεξέργαστων δεδομένων και αξιοποιήσιμων πληροφοριών. Οι συσκευές Raspberry Pi λειτουργούν ως φυσική διεπαφή για τη λήψη δεδομένων σε πραγματικό χρόνο μέσω της σάρωσης κωδικών QR, ενημερώνοντας τα αρχεία παρουσίας στη βάση δεδομένων Firestore. Η εφαρμογή Android, με τον λυτό σχεδιασμό της και τις ολοκληρωμένες λειτουργίες της, προσφέρει στους τελικούς χρήστες εύκολη πρόσβαση στο σύστημα, ενισχύοντας την εμπειρία τόσο για τους φοιτητές όσο και για το διδακτικό προσωπικό.

Η ενσωμάτωση αυτών των στοιχείων αποτελεί μια συνεκτική και καινοτόμο λύση που προσπαθεί να αντιμετωπίζει κάποια προβλήματα των παραδοσιακών μεθόδων καταγραφής παρουσίας. Αξιοποιώντας όλα τα παραπάνω μέρη του συστήματος, το τελικό προϊόν μπορεί να παρέχει επίσης μια πλατφόρμα ανάλυσης, αναφοράς και διαχείρισης που είναι προσβάσιμη, αποτελεσματική και αξιόπιστη.

Αυτό το κεφάλαιο έχει θέσει τις βάσεις για την κατανόηση της τεχνολογικής βάσης του έργου. Τα επόμενα κεφάλαια θα εμβαθύνουν στο σχεδιασμό, την ανάπτυξη και την ενσωμάτωση κάθε στοιχείου, δείχνοντας πώς συμβάλλουν συλλογικά σε μια ολοκληρωμένη λύση που προσπαθεί να ανταποκριθεί στις εξελισσόμενες ανάγκες των σύγχρονων εκπαιδευτικών ιδρυμάτων. Μέσα από αυτή τη διερεύνηση, οι αναγνώστες θα αποκτήσουν μια εικόνα για τη διαδικασία ανάπτυξης του έργου, τις προκλήσεις που αντιμετωπίστηκαν και τις καινοτόμες λύσεις που χρησιμοποιήθηκαν για την αντιμετώπισή τους, υπογραμμίζοντας τη συμβολή του έργου στην προώθηση της τεχνολογίας στην εκπαίδευση.

Κεφάλαιο 3ο: Βάση Δεδομένων Firestore

3.1 Εισαγωγή

Η ενσωμάτωση του Firestore, μιας πλατφόρμας βάσης δεδομένων NoSQL της Google που δημιουργήθηκε το 2017, στο σύστημα παρακολούθησης παρουσίας αποτελεί έναν κρίσιμο καταλύτη για τη μετασχηματιστική αλλαγή στη διαχείριση και οργάνωση των κρίσιμων δεδομένων μέσα στο σύστημα μας. Η βάση δεδομένων Firestore προσφέρει μια αυτοματοποιημένη διαχείριση προσπελάσεων και δυνατότητες συγχρονισμού δεδομένων σε πραγματικό χρόνο, κάνοντας την ιδανική για εφαρμογές υψηλών απαιτήσεων όπως ο έλεγχος παρουσίας.

Μέσα από τη χρήση του Firestore, τα δεδομένα παρουσίας αποθηκεύονται, επεξεργάζονται και ανακτώνται με απόλυτη ακρίβεια και ταχύτητα, επιτρέποντας την άμεση απόκριση σε αλλαγές και ενημερώσεις. Αυτή η λειτουργικότητα προσπαθεί να βελτιώσει σημαντικά την εμπειρία τόσο των διδασκόντων όσο και των φοιτητών, παρέχοντας ένα δυναμικό και εύχρηστο περιβάλλον για τη διαχείριση της τάξης. Η εισαγωγή του Firestore έχει επίσης επιτρέψει την υιοθέτηση πιο προηγμένων αναλυτικών εργαλείων και μεθοδολογιών, ενισχύοντας τη δυνατότητα παρακολούθησης και ανάλυσης της παρουσίας και της συμμετοχής των μαθητών με πιο εξελιγμένους τρόπους. Η καταγραφή της παρουσίας δεν είναι απλώς μια διαδικασία συλλογής δεδομένων αλλά μια διαδικασία στρατηγικής ανάλυσης που μπορεί να βοηθήσει τα ιδρύματα να κατανοήσουν καλύτερα τις συμπεριφορές και τις ανάγκες των φοιτητών τους.

Στις επόμενες υποενότητες, θα εξετάσουμε αναλυτικά τις βασικές λειτουργίες του Firestore, τον κρίσιμο ρόλο του στο σύστημα και τις συγκεκριμένες προσαρμογές που έχουν γίνει για να καλύψουν τις απαιτήσεις του συστήματος, με ειδική αναφορά στη σχεδίαση και υλοποίηση της βάσης δεδομένων.

3.2 Σχεδίαση Βάσης Δεδομένων

Η σχεδίαση της βάσης δεδομένων Firestore για το σύστημα παρακολούθησης παρουσίας είναι μία κρίσιμη διαδικασία που επηρεάζει την αποδοτικότητα, την ευελιξία, και την επεκτασιμότητα του συστήματος. Η προσεκτική και στοχευμένη σχεδίαση διασφαλίζει ότι η βάση δεδομένων μπορεί να χειρίζεται όχι μόνο τον όγκο των δεδομένων αλλά και τις πολύπλοκες απαιτήσεις ανάλυσης και επεξεργασίας δεδομένων, ενώ ταυτόχρονα διασφαλίζει την ακεραιότητα και την ασφάλεια των πληροφοριών.

3.2.1 Επισκόπηση Σχήματος

Το Firestore χρησιμοποιεί ένα μη σχηματικό (**schema-less**) NoSQL σχεδιασμό, προσφέροντας μεγάλη ευελιξία στον τρόπο δόμησης των δεδομένων. Αντί για παραδοσιακούς πίνακες, τα δεδομένα οργανώνονται σε συλλογές (**collections**) που περιέχουν δικαιολογητικά ή εγγραφές (**documents**), καθεμία από τις οποίες μπορεί να έχει μια δυναμική δομή με ζεύγη κλειδιού-τιμής. Κάθε document μπορεί να περιέχει σύνθετα δεδομένα, όπως υποδοχείς (**nested objects**), πίνακες, ακόμη και να περιέχει άλλες υποσυλλογές (**subcollections**), δημιουργώντας ένα ιεραρχικό δομικό μοντέλο.

Ο σχεδιασμός της βάσης δεδομένων Firestore για το συγκεκριμένο έργο είναι προσεκτικά μελετημένος για να διαχειρίζεται αποτελεσματικά τα δεδομένα που σχετίζονται με τις τάξεις, τους φοιτητές, και τη συμμετοχή τους. Η βάση δεδομένων οργανώνεται σε συλλογές και υποσυλλογές που απεικονίζουν τη δομή και τη δυναμική της εκπαιδευτικής διαδικασίας.

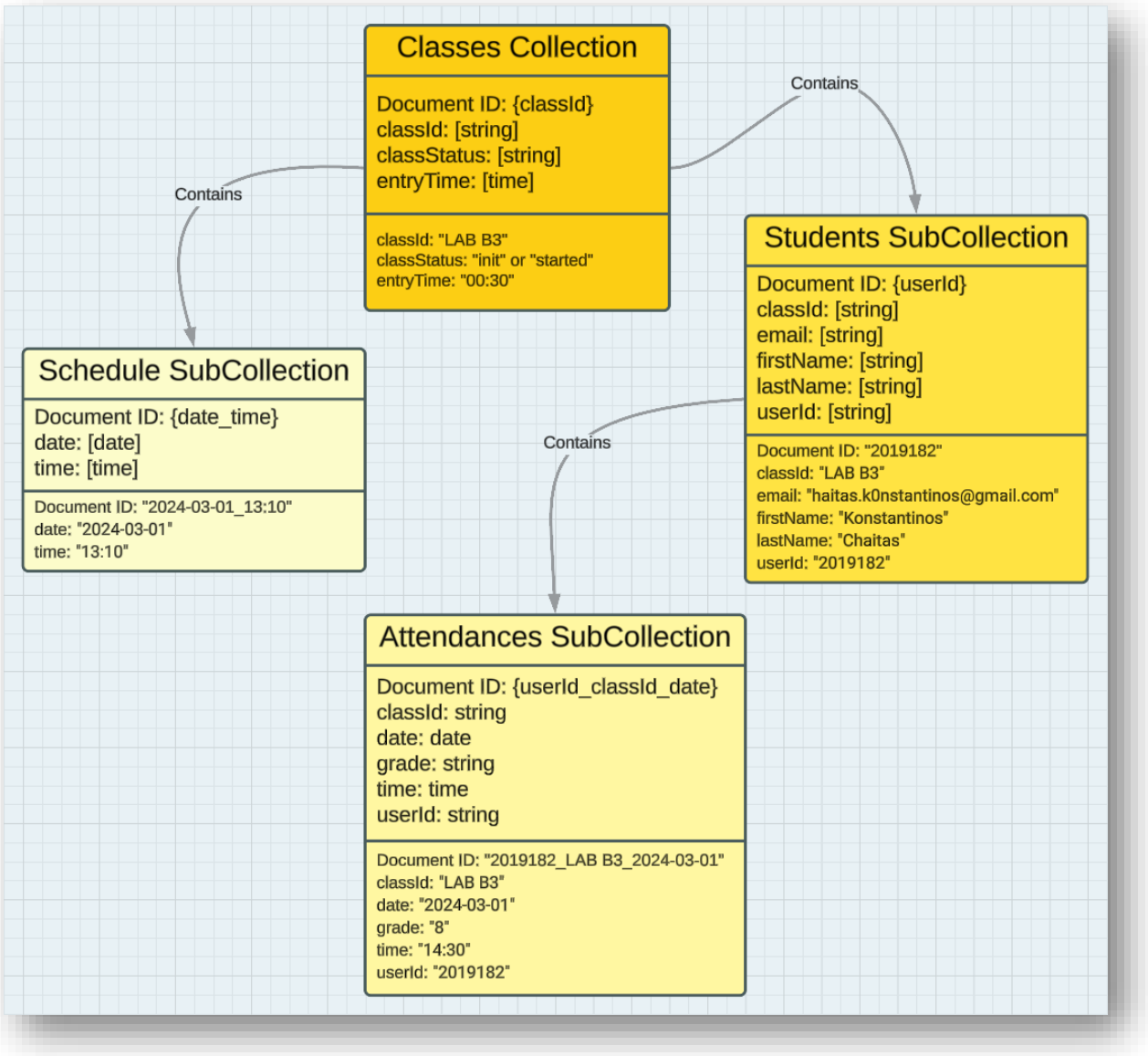
Παρακάτω υπάρχει το αρχικό πλάνο σχεδίασης της βάσης πάνω στο οποίο βασίστηκε η σχεδίαση της:

- **Συλλογή '<u>Μαθήματα</u>':**
  - **Σκοπός:** Αποτελεί τον κύριο κόμβο της βάσης δεδομένων, οργανώνοντας τα δεδομένα συγκεκριμένα για κάθε τάξη.
  - **Πεδία Εγγράφου:**
    1. **Αναγνωριστικό μαθήματος**
    2. **Κατάσταση μαθήματος**
    3. **Μέγιστος χρόνος υποβολής παρουσίας**

- **Υποσυλλογή 'Πρόγραμμα Μαθημάτων':**
  - **Σκοπός:** Κρίσιμο στοιχείο για τη διαχείριση του χρονικού περιθωρίου κατάθεσης παρουσίας, περιέχει εγγραφές για κάθε προγραμματισμένη συνάντηση της τάξης.
  - **Πεδία Εγγράφου:**
    1. **Ημερομηνία**
    2. **Ώρα**

- **Υποσυλλογή '<u>Φοιτητές</u>':**
  - **Σκοπός:** Καταγραφεί κάθε φοιτητή που είναι εγγεγραμμένος στο μάθημα, λειτουργώντας ως αποθετήριο για πληροφορίες σχετικά με τους φοιτητές και την αλληλεπίδρασή τους με την τάξη.
  - **Πεδία Εγγράφου** (Πεδία που καταγράφουν τις προσωπικές και εγγραφικές πληροφορίες του φοιτητή) **:**
    1. **Αναγνωριστικό μαθήματος που είναι εγγεγραμμένος ο φοιτητής**
    2. **Email επικοινωνίας**
    3. **Όνομα**
    4. **Επώνυμο**
    5. **Αναγνωριστικό φοιτητή**

- **Υποσυλλογή '<u>Παρουσίες</u>':**
  - **Σκοπός:** Καταγράφει επιμελώς την παρουσία, παρέχοντας μια λεπτομερή εικόνα της συμμετοχής του φοιτητή.
  - **Πεδία Εγγράφου** (Πεδία που συνδυαστικά παρέχουν ένα πλήρες στιγμιότυπο της παρουσίας και της απόδοσης του φοιτητή σε μια δεδομένη ημερομηνία) **:**
    1. **Αναγνωριστικό μαθήματος**
    2. **Ημερομηνία**
    3. **Ώρα**
    4. **Βαθμός**
    5. **Αναγνωριστικό φοιτητή**



Σχήμα 3.1: Σχήμα Βάσης Δεδομένων Firestore

Η καλά οργανωμένη δομή της βάσης δεδομένων Firestore επιτρέπει την αποτελεσματική και αξιόπιστη διαχείριση των δεδομένων, διευκολύνοντας την ομαλή λειτουργία του συστήματος παρακολούθησης παρουσίας και ενισχύοντας τη συνολική εμπειρία των χρηστών.

3.2.2 Πρακτικές Σκέψεις για την Επιλογή της Δομής

Η επεκτασιμότητα είναι ένα βασικό χαρακτηριστικό που καθορίζει τον τρόπο σχεδίασης του σχήματος της βάσης δεδομένων στο σύστημά μας. Καθώς τα εκπαιδευτικά ιδρύματα αναπτύσσονται, αυξάνονται και οι ανάγκες για διαχείριση μεγαλύτερων όγκων δεδομένων, από τον αριθμό των μαθητών μέχρι τις τάξεις που προσφέρονται. Το Firestore υποστηρίζει αυτή την επέκταση με μια δυναμική αρχιτεκτονική που προσαρμόζεται αυτόματα για να διαχειρίζεται αυξημένα φορτία χωρίς την ανάγκη

για αλλαγές στο σχέδιο ή περίπλοκες διαδικασίες ενημέρωσης. Βέβαια αυτό προϋποθέτει τον σωστό σχεδιασμό της βάσης εξαρχής.

Η ακρίβεια και η συνέπεια των δεδομένων είναι κρίσιμη, ιδίως όταν πρόκειται για την καταγραφή παρουσιών και ακαδημαϊκών επιδόσεων. Το Firestore εφαρμόζει αυστηρούς κανόνες επικύρωσης και χρησιμοποιεί μηχανισμούς συναλλαγών για ενημερώσεις που εμπλέκουν πολλαπλά δεδομένα, εξασφαλίζοντας την ακεραιότητα και την ποιότητα της πληροφορίας. Αυτό καθιστά το Firestore ιδιαίτερα κατάλληλο για εκπαιδευτικά δεδομένα, όπου η δομή και ο όγκος των δεδομένων μπορεί να μεταβάλλονται συχνά. Η δομή του Firestore είναι σχεδιασμένη να αξιοποιεί την ευελιξία των NoSQL βάσεων δεδομένων, οι οποίες είναι εξαιρετικές στη διαχείριση δεδομένων που δεν είναι απολύτως δομημένα ή που αλλάζουν συχνά [21]. Οι NoSQL βάσεις δεδομένων, όπως το Firestore, επιτρέπουν την αποθήκευση δεδομένων με ευέλικτο σχήμα, γεγονός που μειώνει την ανάγκη για αναδιάρθρωση των δεδομένων όταν προστίθενται νέες πληροφορίες. Αυτό είναι ιδιαίτερα χρήσιμο σε ένα εκπαιδευτικό περιβάλλον όπου οι ανάγκες μπορεί να αλλάζουν γρήγορα. Επιπλέον, οι NoSQL βάσεις δεδομένων είναι κατασκευασμένες για να διαχειρίζονται μεγάλα δεδομένα από την αρχή, προσφέροντας υψηλή απόδοση και ευκολία στην κλιμάκωση. Η χρήση μιας προσέγγισης «scale-out» καθιστά τη διαδικασία της επέκτασης οικονομικά αποδοτική και τεχνικά απλή, σε αντίθεση με την παραδοσιακή προσέγγιση «scale-up» που απαιτεί πιο σύνθετες και δαπανηρές αναβαθμίσεις [22].

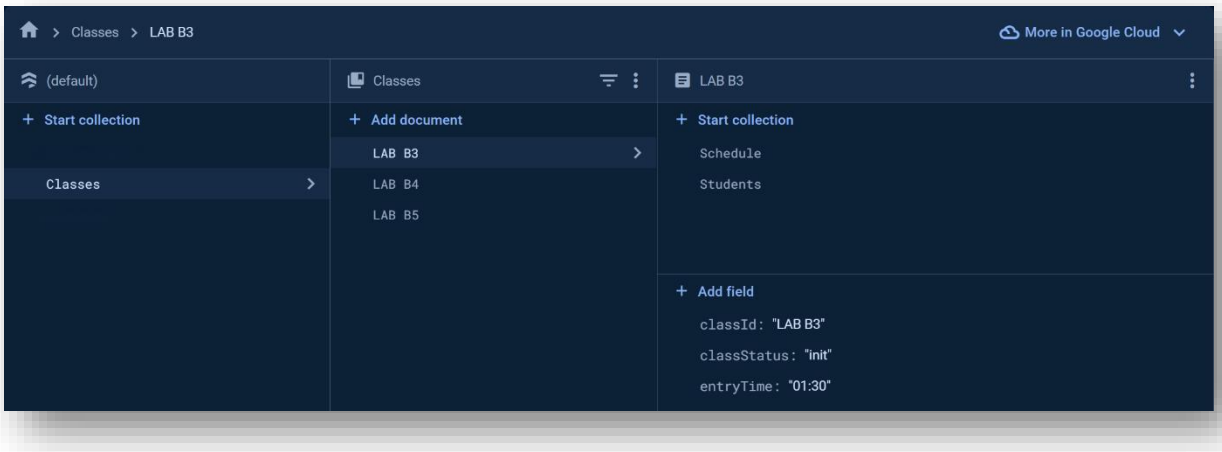
Συνολικά, η σχεδίαση της βάσης δεδομένων Firestore για το σύστημα ελέγχου παρουσίας προσπαθεί να αντιμετωπίσει τις προκλήσεις με μια στοχευμένη προσέγγιση, η οποία εστιάζει στην επεκτασιμότητα, την ακεραιότητα των δεδομένων, τον συγχρονισμό σε πραγματικό χρόνο και την ασφάλεια, εξασφαλίζοντας την αποδοτικότητα και τη βιωσιμότητα του συστήματος.

3.3 Λεπτομέρειες της Υλοποίησης

Ο σχεδιασμός και η αρχιτεκτονική της βάσης δεδομένων Firestore έχουν μεγάλη επιρροή στη λειτουργία του συστήματος, υποστηρίζοντας τις λειτουργίες που σχετίζονται με τη διαχείριση της τάξης, το χειρισμό πληροφοριών των φοιτητών και την παρακολούθηση των παρουσιών.

3.3.1 Διαχείριση Τάξεων

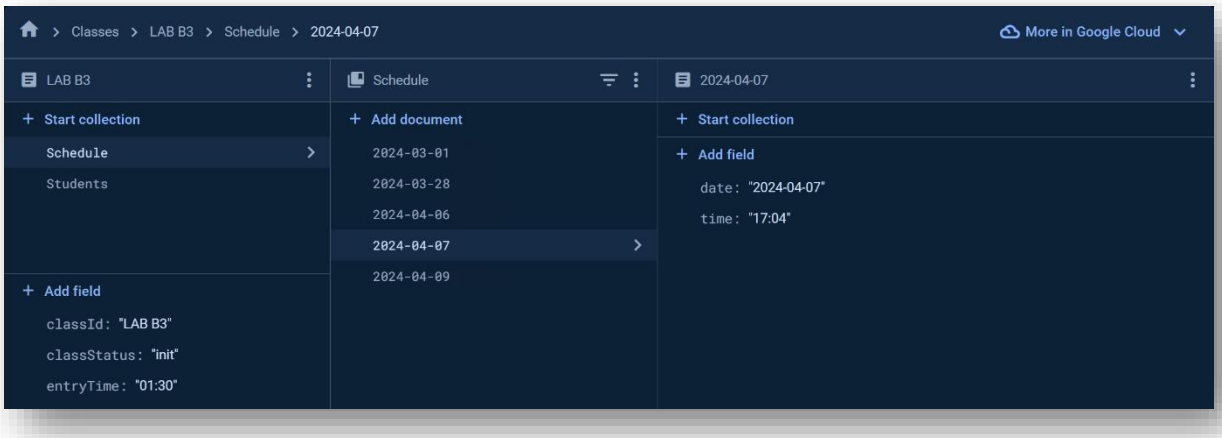
Η αποθήκευση και η οργάνωση των πληροφοριών της κάθε τάξης στο Firestore επικεντρώνονται γύρω από τη συλλογή τάξεων (**Classes**). Κάθε έγγραφο αυτής της συλλογής αναγνωρίζεται μοναδικά από ένα {classId} και περιέχει πεδία που αποθηκεύουν ζωτικές λεπτομέρειες σχετικά με την τάξη, όπως {**classStatus**} και {**entryTime**}. Πιο αναλυτικά το πεδίο{**classStatus**} μπορεί να πάρει την τιμή String(**init**) για την αρχικοποίησή του μαθήματος, την τιμή String(**started**) όταν το μάθημα έχει ξεκινήσει και οι φοιτητές που θα το παρακολουθήσουν έχουν συμπληρωθεί - *περισσότερες λεπτομερείς για αυτή την λειτουργεία στο Κεφάλαιο 5 Raspberry Pi για Παρακολούθηση Παρουσίας και στο Κεφάλαιο 6 Ανάπτυξη Εφαρμογής Android*- και την τιμή String(**finished**) όταν το μάθημα έχει ολοκληρωθεί. Επιπρόσθετα το πεδίο {**entryTime**} μπορεί να λάβει τιμή της εξής μορφής String(**hh:mm**) όπου hh = ώρα και mm τα λεπτά. Με αυτό το πεδίο καθορίζουμε τον επιτρεπόμενο χρόνο που έχει κάθε φοιτητής να δηλώσει την παρουσία του μετα την έναρξη της συνεδρίας. Αυτά τα έγγραφα χρησιμεύουν ως το θεμελιώδες στρώμα πάνω στο οποίο δομούνται τα λειτουργικά δεδομένα της τάξης.



Σχήμα 3.2: Classes Subcollection

➢ **Χρονοδιάγραμμα:**

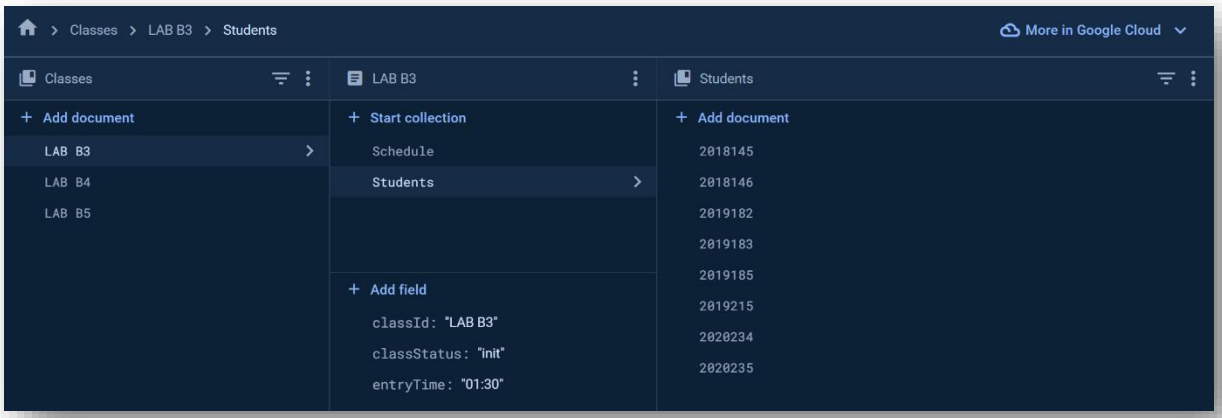
Ένθετη μέσα σε κάθε έγγραφο τάξης (**Class**) είναι μια δευτερεύουσα συλλογή χρονοδιαγράμματος (**Schedule**), όπου κάθε έγγραφο αντιπροσωπεύει μια συγκεκριμένη παρουσία της σύσκεψης τάξης. Περιέχει πεδία για την ημερομηνία {**date**} με τιμή String(**YYYY-MM-DD**) όπου YYYY = η χρονιά, MM = ο μήνας, DD = η ημέρα και την ώρα {**time**} με τιμή String(**hh:mm**) όπου hh = η ώρα, mm = τα λεπτά. Αυτό επιτρέπει μια λεπτομερή προσέγγιση στη διαχείριση του προγράμματος, φιλοξενώντας τακτικές και εφάπαξ συνεδρίες τάξης.



Σχήμα 3.3: Schedule Subcollection

➢ **Συμμετέχοντες φοιτητές:**

Η υποσυλλογή Φοιτητές (**Students**) μέσα σε κάθε έγγραφο τάξης εμπεριέχει όλους τους εγγεγραμμένους φοιτητές. Αυτό περιλαμβάνει όχι μόνο τα προσωπικά τους στοιχεία, όπως email, όνομα και επώνυμο, αλλά και συνδέσμους πίσω στην τάξη μέσω του {**classId**}, δημιουργώντας μια αμφίδρομη σχέση μεταξύ των μαθητών και των τάξεων τους. Αυτή η δομή διευκολύνει την εύκολη ανάκτηση όλων των μαθητών που σχετίζονται με μια τάξη και αντίστροφα.



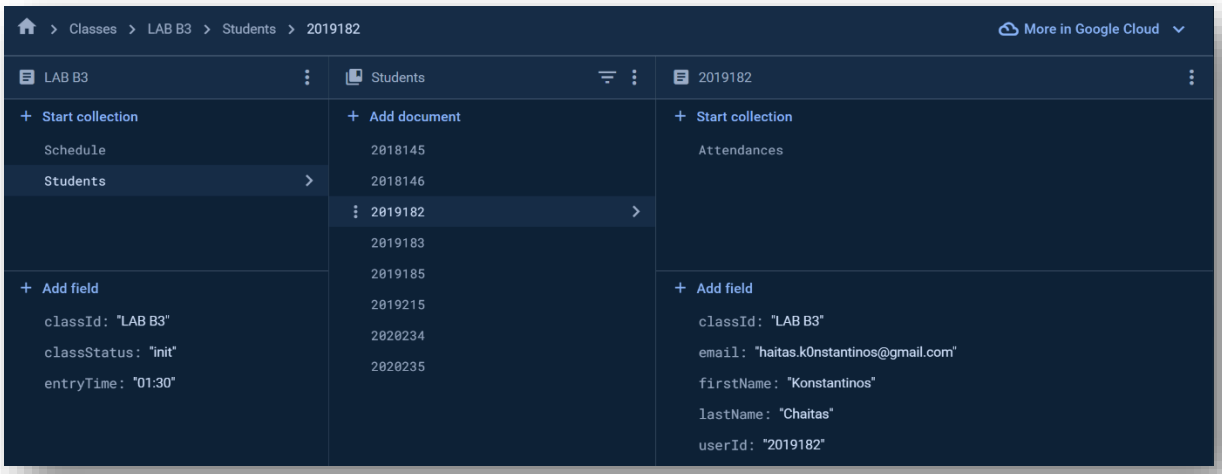
Σχήμα 3.4: Students Subcollection

3.3.2 Χειρισμός Πληροφορίων των Φοιτητών

Η αποτελεσματική διαχείριση των πληροφοριών των φοιτητών είναι ζωτικής σημασίας για τη ακεραιότητα οποιουδήποτε εκπαιδευτικού συστήματος. Στο σύστημα, το Firestore διαδραματίζει κεντρικό ρόλο στο χειρισμό των προφίλ των φοιτητών, διασφαλίζοντας ότι τα προσωπικά και ακαδημαϊκά δεδομένα αποθηκεύονται με ασφάλεια. Αυτή η ενότητα διερευνά τη δομή των προφίλ φοιτητών στο Firestore.

- **Προφίλ Φοιτητή:**

Η υποσυλλογή Φοιτητές (**Students**) μέσα σε κάθε έγγραφο τάξης στη συλλογή τάξεις (**Classes**) είναι ζωτικής σημασίας για την αποθήκευση λεπτομερών πληροφοριών σχετικά με τους φοιτητές. Κάθε έγγραφο σε αυτήν την υποσυλλογή αντιπροσωπεύει ένα μοναδικό προφίλ φοιτητή, το οποίο προσδιορίζεται από ένα {**userId**}. Η δομή έχει σχεδιαστεί σχολαστικά για να ενσωματώνει τόσο προσωπικές όσο και ακαδημαϊκές πληροφορίες απαραίτητες για τη λειτουργικότητα του συστήματος. Το {**classId**} συνδέει τον φοιτητή με τα εγγεγραμμένα μαθήματα του, επιτρέποντας μια σχεσιακή προβολή μεταξύ των μαθητών και των ακαδημαϊκών τους δεσμεύσεων. Τα βασικά προσωπικά στοιχεία κάθε φοιτητή όπως το όνομα {**firstName**}, επώνυμο {**lastName**} και email {**email**} διευκολύνουν την επικοινωνία και την εξατομίκευση της διεπαφής του συστήματος.



Σχήμα 3.5: Student Document

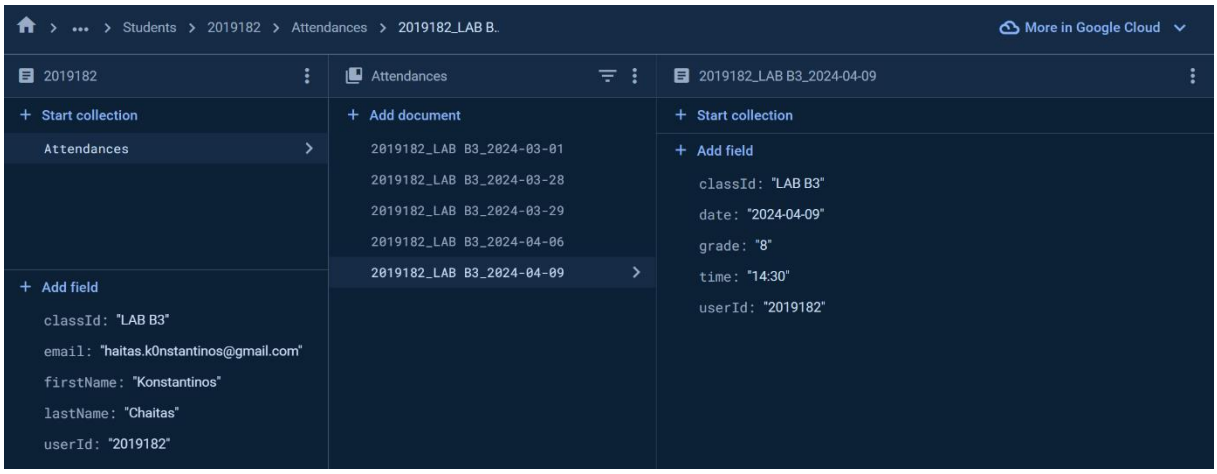
3.3.3 Καταγραφή της Παρουσίας

Η καταγραφή των παρουσιών αποτελεί τον βασικό στόχο του συστήματος για αυτόν τον λόγο είναι και η βασική λειτουργία του, παρέχοντας πολύτιμα δεδομένα για την ακαδημαϊκή παρακολούθηση και την ανάλυση της επίδοσης των μαθητών. Η αρχιτεκτονική της βάσης δεδομένων Firestore έχει σχεδιαστεί περίτεχνα για να διευκολύνει την αποτελεσματική και ακριβή καταγραφή της παρουσίας.

Η δομή των αρχείων παρακολούθησης στη βάση δεδομένων Firestore είναι τόσο λεπτομερής όσο και οργανωμένη, διασφαλίζοντας ότι κάθε εγγραφή συνδέεται με ακρίβεια με συγκεκριμένες τάξεις και φοιτητές. Αυτή η σύνδεση επιτυγχάνεται μέσω της υποσυλλογής Παρουσίες (**Attendances**) που είναι ένθετη μέσα στην υποσυλλογή Φοιτητές (**Students**) σε κάθε έγγραφο τάξης. Κάθε αρχείο παρουσίας αποθηκεύεται σε ένα έγγραφο εντός της υποσυλλογής Παρουσίες (**Attendances**), το οποίο προσδιορίζεται από ένα σύνθετο κλειδί ({**StudentId**}_{**classId**}_{**date**}), το οποίο αντιπροσωπεύει μοναδικά μια παρουσία για έναν φοιτητή σε μια συγκεκριμένη συνεδρία και ημερομηνία.

- **Πεδία Εγγράφου:**

Το {**classId**} προσδιορίζει την τάξη στην οποία αναφέρεται το αρχείο παρουσίας. Η ημερομηνία {**date**} και η ώρα {**time**} καταγράφει την ακριβή ημερομηνία και ώρα παρακολούθησης του φοιτητή, διευκολύνοντας την ακριβή τήρηση αρχείων. Το {**userId**}συνδέει το αρχείο παρακολούθησης με τον συγκεκριμένο φοιτητή. Τέλος ο βαθμός {**grade**} παίρνει τιμές String(float(**N.n**)) όπου N = μονάδες, n = δεκάδες και αντιπροσωπεύει την αξιολόγηση του φοιτητή σε κάθε παρουσία. Αυτή η δομημένη προσέγγιση επιτρέπει την απλή ανάκτηση και ανάλυση των δεδομένων συμμετοχής, υποστηρίζοντας διάφορες ακαδημαϊκές και διοικητικές λειτουργίες εντός του συστήματος.



Σχήμα 3.6: Attendance Document

3.4 Προκλήσεις που Αντιμετωπίστηκαν

Ο σχεδιασμός και η υλοποίηση της βάσης δεδομένων Firestore για το σύστημα ελέγχου παρουσίας δεν ήταν χωρίς προκλήσεις. Αυτά τα εμπόδια απαιτούσαν προσεκτικές λύσεις και στρατηγικές αποφάσεις για τη διασφάλιση της αποτελεσματικότητας και της αξιοπιστίας του συστήματος. Αυτή η ενότητα περιγράφει μερικές από τις κύριες προκλήσεις που αντιμετωπίστηκαν και τις λύσεις που επινοήθηκαν για την αντιμετώπισή τους, ρίχνοντας φως στην κρίσιμη διαδικασία λήψης αποφάσεων καθ' όλη τη διάρκεια του αναπτυξιακού πλάνου.

Η ιεραρχική φύση του Firestore απαιτούσε μια σύνθετη δομή δεδομένων για την αποτελεσματική διαχείριση των τάξεων, των μαθητών και των αρχείων παρουσίας. Η διασφάλιση της αποτελεσματικής ανάκτησης και ενημέρωσης των δεδομένων, διατηρώντας παράλληλα μια κλιμακούμενη δομή, αποτέλεσε σημαντική πρόκληση. Η λύση περιλάμβανε την υιοθέτηση μιας ένθετης συλλογής που αναπαριστά τις πραγματικές σχέσεις μεταξύ τάξεων, μαθητών και αρχείων παρουσίας. Για τη βελτιστοποίηση της ανάκτησης δεδομένων μπορεί επιπλέον να εφαρμοστεί βελτιστοποίηση ερωτημάτων για να ελαχιστοποιηθούν οι λειτουργίες ανάγνωσης και εγγραφής.

Η διασφάλιση της ασφάλειας των ευαίσθητων πληροφοριών των μαθητών και η διατήρηση της ακεραιότητας των δεδομένων παρακολούθησης ήταν υψίστης σημασίας. Το σύστημα έπρεπε να αποτρέψει τη μη εξουσιοδοτημένη πρόσβαση και παραποίηση των δεδομένων, υποστηρίζοντας παράλληλα μια μεγάλη και ποικίλη βάση χρηστών. Χρησιμοποιήσαμε τους ισχυρούς κανόνες ασφαλείας του Firestore για να επιβάλουμε έλεγχο ταυτότητας και εξουσιοδότησης σε επίπεδο βάσης δεδομένων. Αυτοί οι κανόνες δημιουργήθηκαν σχολαστικά για να επιτρέπουν την πρόσβαση μόνο σε χρήστες με τα κατάλληλα δικαιώματα, με βάση τους ρόλους τους και τα δεδομένα που είναι εξουσιοδοτημένοι να προβάλλουν ή να τροποποιούν. Επιπλέον, μπορούν να χρησιμοποιηθούν ενημερώσεις συναλλαγών και επικύρωση πεδίου για τη διατήρηση της ακεραιότητας των δεδομένων, διασφαλίζοντας ότι όλες οι λειτουργίες της βάσης δεδομένων είναι μοναδικές και συνεπείς.

Ο σχεδιασμός ενός συστήματος που παραμένει αποδοτικό και επεκτάσιμο καθώς αυξάνεται ο αριθμός των χρηστών, των τάξεων και των αρχείων παρακολούθησης ήταν μια σημαντική πρόκληση. Η δομή της βάσης δεδομένων έπρεπε να υποστηρίξει την ταχεία ανάπτυξη χωρίς να υποβαθμίσει την εμπειρία του χρήστη. Η δομή της βάσης δεδομένων σχεδιάστηκε με γνώμονα τη δυνατότητα κλιμάκωσης, χρησιμοποιώντας βέλτιστες πρακτικές για τη δομή εγγράφων και συλλογής για τη διασφάλιση

αποτελεσματικής υποβολής ερωτημάτων και αλλαγών. Με την χρήση της δυνατότητας του Firestore να κλιμακώνεται αυτόματα, παράλληλα με την εφαρμογή τεχνικών ασύγχρονης φόρτωσης και σελιδοποίησης στην εφαρμογή έχουμε ως αποτέλεσμα την βελτιστοποίηση της διαχείρισης μεγάλων συνόλων δεδομένων. Αυτή η προσέγγιση εξασφαλίζει ότι το σύστημα θα μπορούσε να φιλοξενήσει την ανάπτυξη χωρίς να θυσιάσει την απόδοση του.

Οι προκλήσεις που αντιμετωπίστηκαν κατά το σχεδιασμό και την υλοποίηση της βάσης δεδομένων Firestore για το σύστημα παρακολούθησης παρουσίας αντιμετωπίστηκαν με καινοτόμες λύσεις που αξιοποίησαν τα δυνατά σημεία του Firestore. Αυτές οι λύσεις όχι μόνο αντιμετώπισαν τα άμεσα ζητήματα, αλλά έθεσαν επίσης τα θεμέλια για ένα κλιμακούμενο, ασφαλές και αποτελεσματικό σύστημα ικανό να υποστηρίξει τις δυναμικές ανάγκες των εκπαιδευτικών ιδρυμάτων. Μέσω προσεκτικού σχεδιασμού, στρατηγικής λήψης αποφάσεων και δέσμευσης για την αξιοποίηση της τεχνολογίας αιχμής, καταφέραμε να ξεπεράσουμε αυτά τα εμπόδια, με αποτέλεσμα ένα ισχυρό και αξιόπιστο σύστημα παρακολούθησης παρουσίας.

3.5 Σύνοψη

Σε όλο αυτό το κεφάλαιο, έχουμε εμβαθύνει στις λεπτομέρειες του τρόπου με τον οποίο η βάση δεδομένων Firestore υποστηρίζει το σύστημα καταγραφής παρουσιών, επιδεικνύοντας τον κρίσιμο ρόλο του στη διαχείριση πληροφοριών τάξης, προφίλ φοιτητών και αρχείων παρουσίας. Η βάση δεδομένων Firestore, με τον συγχρονισμό δεδομένων σε πραγματικό χρόνο, την επεκτασιμότητα και τις ευέλικτες δυνατότητές της, χρησιμεύει ως η ραχοκοκαλιά του συστήματος, διασφαλίζοντας ότι όλα τα στοιχεία λειτουργούν συνεκτικά και αποτελεσματικά.

Ο σχεδιασμός και η υλοποίηση της βάσης δεδομένων Firestore προσεγγίστηκαν με σχολαστική προσοχή για την αντιμετώπιση των ειδικών αναγκών της τάξης, των φοιτητών και της διαχείρισης παρουσιών. Η ιεραρχική δομή της βάσης δεδομένων, με τις συλλογές, τα έγγραφα και τις υποσυλλογές της, αντικατοπτρίζει την λογική των εκπαιδευτικών ιδρυμάτων, επιτρέποντας την απλή ανάκτηση και ενημέρωση δεδομένων. Αυτή η δομή υποστηρίζει τη δυναμική διαχείριση των προγραμμάτων των τάξεων, τη λεπτομερή τήρηση αρχείων με πληροφορίες των φοιτητών και την παρακολούθηση σε πραγματικό χρόνο της παρουσίας των φοιτητών, διατηρώντας παράλληλα υψηλό βαθμό ακεραιότητας και ασφάλειας δεδομένων.

Αντιμετωπίσαμε επίσης τις προκλήσεις που συναντήσαμε κατά τη διαδικασία ανάπτυξης, συμπεριλαμβανομένης της πολυπλοκότητας του σχεδιασμού ενός κλιμακούμενου μοντέλου δεδομένων, της διασφάλισης της συνέπειας δεδομένων σε πραγματικό χρόνο μεταξύ χρηστών και συσκευών, της προστασίας ευαίσθητων πληροφοριών και της βελτιστοποίησης της απόδοσης του συστήματος. Οι λύσεις σε αυτές τις προκλήσεις αποδεικνύουν τις προσεκτικές προσπάθειες λήψης αποφάσεων και επίλυσης προβλημάτων που συνέβαλαν στην επιτυχία του συστήματος και στην πρόληψη μελλοντικών προβλημάτων.

Συμπερασματικά, αυτό το κεφάλαιο παρέχει μια ολοκληρωμένη επισκόπηση του ρόλου της βάσης δεδομένων Firestore, υπογραμμίζοντας την καθοριστική συμβολή της στους στόχους του έργου. Το Firestore είναι κάτι περισσότερο από ένα τεχνολογικό στοιχείο. Διευκολύνει την εκπαιδευτική καινοτομία, επιτρέποντας στα ιδρύματα να διαχειρίζονται και να αναλύουν τα δεδομένα παρακολούθησης με ακρίβεια και ευκολία.

Κεφάλαιο 4ο: Google Apps Scripts για Ακαδημαϊκή Διαχείριση

4.1 Εισαγωγή

Στον κόσμο της ακαδημαϊκής διοίκησης, η ενσωμάτωση της τεχνολογίας και του αυτοματισμού έχει γίνει απαραίτητη για την αποτελεσματικότητα. Τα Google Apps Scripts, μια δυναμική και ευέλικτη πλατφόρμα scripting, μπορούν να προσφέρουν εξέλιξη σε αυτόν τον τομέα. Διευκολύνουν τις περίπλοκες διαδικασίες διαχείρισης τεράστιων συστοιχιών δεδομένων - από την παρακολούθηση της παρουσίας των φοιτητών σχολαστικά έως την επεξεργασία των προσωπικών τους στοιχείων και τον υπολογισμό των βαθμών τους. Ως μεσάζων μεταξύ των Google Sheets και της ισχυρής αποθήκευσης και ανάλυσης δεδομένων του Firestore, τα Google Apps Scripts προσπαθούν να αλλάξουν τον τρόπο με τον οποίο τα εκπαιδευτικά ιδρύματα χειρίζονται τα διοικητικά τους καθήκοντα, σηματοδοτώντας την εξέλιξή τους από ένα απλό εργαλείο δέσμης ενεργειών σε μια ουσιαστική πτυχή της εκπαιδευτικής τεχνολογίας.

Εντοπίζοντας τις ρίζες τους πίσω στο 2009, τα Google Apps Scripts σχεδιάστηκαν αρχικά ως μέσο βελτίωσης της παραγωγικότητας στα Google Sheets. Από τότε έχουν ξεπεράσει τις ρίζες τους, επεκτείνοντας σε ένα ολοκληρωμένο περιβάλλον ανάπτυξης που υιοθετεί την JavaScript, μια γλώσσα που φημίζεται για την πανταχού παρουσία και τη δύναμή της. Η πλατφόρμα έχει ενισχυθεί εκτεταμένα με μια πληθώρα υπηρεσιών και APIs, εδραιώνοντας έτσι τη θέση της ως ζωτικό εργαλείο όχι μόνο για την απλοποίηση των λειτουργιών του Google Workspace αλλά και την αναβάθμιση τους.

4.1.1 Βασικά Χαρακτηριστικά και πλεονεκτήματα σε ένα εκπαιδευτικό περιβάλλον

- **Ισχυρή αυτοματοποίηση και προσαρμογή:**
  Στον πυρήνα τους, τα Google Apps Scripts υπερέχουν στην αυτοματοποίηση περίπλοκων και χρονοβόρων εργασιών που είναι επιρρεπείς σε ανθρώπινο λάθος, όπως η διαχείριση δεδομένων και η αναφορά. Μέσω προσαρμοσμένων σεναρίων, επιτρέπεται η εκτέλεση σύνθετων ροών εργασίας, ενισχύοντας έτσι την ακρίβεια και απελευθερώνοντας πολύτιμο χρόνο για το διοικητικό προσωπικό.

- **Ενσωμάτωση σε όλες τις υπηρεσίες Google:**
  Τα G.A.S. έχουν την δυνατότητα να δημιουργήσουν μια γέφυρα μεταξύ διαφόρων υπηρεσιών της Google. Λειτουργούν ως καταλυτής στη δημιουργία ολοκληρωμένων ροών εργασίας που ενισχύουν τη συνεργασία και την παραγωγικότητα, συνδέοντας εργαλεία όπως τα Google Sheets, τα Google Docs, το Google Calendar και το Gmail για να σχηματίσουν ένα συνεκτικό οικοσύστημα.

- **Ευρεία προσβασιμότητα και ευκολία χρήσης:**
  Τα G.A.S. έχουν σχεδιαστεί με την προσβασιμότητα στον πυρήνα τους, εξυπηρετώντας ένα ευρύ φάσμα χρηστών. Η αξιοποίηση της JavaScript επιτρέπει στην πλατφόρμα να είναι τόσο ελκυστική σε έναν αρχάριο που κάνει τα πρώτα του βήματα στον προγραμματισμό όσο και σε έναν έμπειρο προγραμματιστή που επιδιώκει να εφαρμόσει προηγμένες λύσεις.

- **Προηγμένη επεκτασιμότητα για ολοκληρωμένες εφαρμογές:**
  Η επεκτασιμότητα των Google Apps Scripts, ιδιαίτερα η ενσωμάτωσή τους με το Google Cloud Platform και η δυνατότητα διασύνδεσης με εξωτερικά APIs, τα ωθεί πέρα από το αρχικό πεδίο εφαρμογής τους. Λειτουργούν ως γέφυρα με άλλες εφαρμογές, προωθώντας τη δημιουργία προσαρμοσμένων εφαρμογών που εκτείνονται σε πολλαπλές πλατφόρμες και υπηρεσίες.

Η ανάπτυξη των G.A.S. στον ακαδημαϊκό τομέα αναδεικνύει την ικανότητά τους στην απλοποίηση και αυτοματοποίηση των διοικητικών λειτουργιών. Παρέχει ένα μέσο ψηφιοποίησης και συστηματοποίησης διαδικασιών ρουτίνας, δίνοντας τη δυνατότητα στα εκπαιδευτικά ιδρύματα να ξεφύγουν από τις παραδοσιακές μεθόδους και να στραφούν σε πιο προηγμένες προσεγγίσεις που βασίζονται πάνω σε δεδομένα. Η επιρροή τους επεκτείνεται στην ενίσχυση των διαδικασιών λήψης αποφάσεων, παρέχοντας στους εκπαιδευτικούς και τους διαχειριστές άμεση πρόσβαση σε δεδομένα που μπορούν να επηρεάζουν θετικά τις αποφάσεις τους. Αυτό το κεφάλαιο προσφέρει μια λεπτομερή διερεύνηση αυτών των πτυχών, υπογραμμίζοντας τον ζωτικό ρόλο που διαδραματίζουν τα Google Apps Scripts στη συνεχή εξέλιξη της ακαδημαϊκής διοίκησης.

4.2 Λειτουργικότητα και Σχεδίαση

Με την πρώτη ματιά στη διεπαφή των Google Sheets, είναι αμέσως σαφές ότι το υπολογιστικό φύλλο χρησιμεύει ως ένα ολοκληρωμένο εργαλείο για την διαχείριση των παρουσιών από τους εκπαιδευτικούς. Αυτός είναι ο πίνακας ελέγχου του χρήστη για τη διαχείριση δεδομένων που σχετίζονται με την τάξη, παρουσιάζοντας αποτελεσματικά τη διπλή λειτουργικότητα τόσο της καταγραφής της συμμετοχής όσο και της ανάλυσης των βαθμών.

Η διάταξη είναι προσεκτικά δομημένη για να παρέχει γρήγορη πρόσβαση σε ζωτικές πληροφορίες. Οι αρχικές στήλες παρέχουν μια αναφορά που μοιάζει με βάση δεδομένων για τις πληροφορίες των φοιτητών, απαραίτητη για οποιοδήποτε εκπαιδευτικό σύστημα παρακολούθησης παρουσίας. Μετά από αυτές τις πληροφορίες, το υπολογιστικό φύλλο μεταβαίνει σε ένα λεπτομερές αρχείο καταγραφής παρουσίας, με στήλες επικεφαλίδας ημερομηνιών που περιέχουν αρχεία παρακολούθησης για κάθε συνεδρία της τάξης.

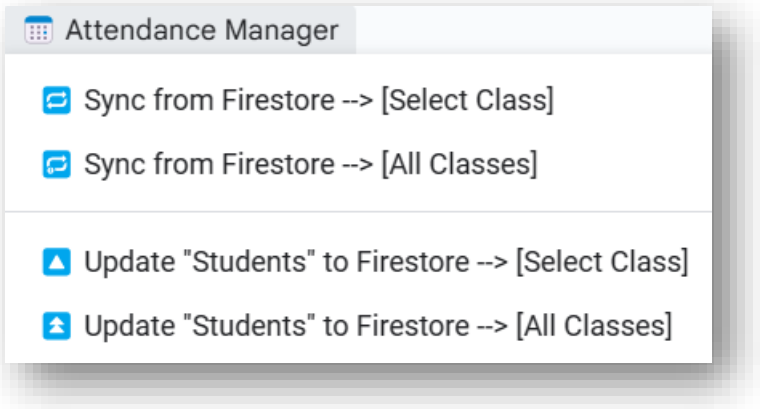
Η χρωματική κωδικοποίηση διαδραματίζει στρατηγικό ρόλο στη διεπαφή του χρήστη, καθοδηγώντας το μάτι του στα στιγμιότυπα παρακολούθησης και βαθμού. Οι αποχρώσεις του πράσινου και του κόκκινου προσφέρουν οπτική ανατροφοδότηση σχετικά με τη συμμετοχή των μαθητών, ενώ η στήλη σύνοψης παρακολούθησης μετατρέπει αυτά τα δεδομένα σε μια άμεσα κατανοητή μορφή. Ομοίως, η στήλη βαθμολόγησης απλοποιεί τα δεδομένα αξιολόγησης σε κατανοητές, αξιοποιήσιμες πληροφορίες για την πρόοδο των φοιτητών.

Αυτή η προβολή είναι το αποτέλεσμα μιας σχολαστικής δέσμης ενεργειών στο backend, η οποία διαχειρίζεται την εισαγωγή δεδομένων και τη δημιουργία αναφορών, διασφαλίζοντας ότι ο χρήστης - είτε πρόκειται για καθηγητή ή διαχειριστή - παρουσιάζεται με ενημερωμένες πληροφορίες που είναι ζωτικής σημασίας για την ακαδημαϊκή επιτυχία και τις στρατηγικές παρεμβάσεις. Ο σχεδιασμός με επίκεντρο τον χρήστη διευκολύνει την ευκολία χρήσης και μειώνει το γνωστικό φορτίο, επιτρέποντας στους χρήστες να επικεντρωθούν στην ερμηνεία των δεδομένων και όχι στην οργάνωσή τους.

4.2.1 Δημιουργία και Διαχείριση Φύλλων

Η διαδικασία αυτοματοποίησης για τη διαχείριση ακαδημαϊκών υπολογιστικών φύλλων μέσω των Google Apps Scripts ξεκινά τη στιγμή που ένας χρήστης ανοίγει το Google Sheets, καθώς η λειτουργία onOpen στον κώδικα μας ενεργοποιεί την προσθήκη ενός προσαρμοσμένου μενού, με τίτλο “ Attendance Manager”. Αυτό το μενού ενισχύει τη χρηστικότητα του υπολογιστικού φύλλου παρέχοντας εύκολη πρόσβαση σε κοινές εργασίες, αλλά και προσανατολίζει τον χρήστη προς τις

διαθέσιμες αυτοματοποιημένες λειτουργίες, όπως ο συγχρονισμός από το Firestore και η αποθήκευση των δεδομένων σε αυτό.



Σχήμα 4.1: Custom UI Elements για την διαχείριση διεργασιών στα Google Sheets

Η επιλογή **Sync from Firestore → [Select Class]** ξεχωρίζει ως το βασικό στοιχείο της αλληλεπίδρασης του χρήστη. Όταν καλείται, δημιουργεί μια ερώτηση περιβάλλοντος εργασίας χρήστη που ζητά από το χρήστη να καθορίσει την τάξη που επιθυμεί για συγχρονισμό. Η διαδικασία είναι έμφυτη, αντικατοπτρίζοντας την απλότητα των καθημερινών ροών εργασίας, μειώνοντας έτσι την δυσκολία προσαρμογής για χρήστες με διαφορετικά επίπεδα τεχνικής εξειδίκευσης. Αντίστοιχα η διαδικασία **Sync from Firestore → [All Classes]** εκτελεί των συγχρονισμό όλων των διαθέσιμων τάξεων που βρίσκονται στην βάση δεδομένων. Ο διαχωρισμός αυτών τον δυο επιλογών έχει γίνει με σκοπό την ταχύτητα του συστήματος.

Η διαδικασία ξεκινάει με την λήψη του αναγνωριστικού τάξης από τον χρήστη (εκτός αν επιθυμούμε τον συγχρονισμό όλων των τάξεων), το σενάριο αποθηκεύει το καθορισμένο αναγνωριστικό και εάν το αντίστοιχο φύλλο δεν υπάρχει, το σενάριο δημιουργεί ευέλικτα ένα νέο, διασφαλίζοντας ότι οι χρήστες δεν παρεμποδίζονται από την ανάγκη για χειροκίνητη ρύθμιση. Για κάθε τάξη που έχει επιλεχθεί για συγχρονισμό, το σενάριο διαγράφει τα τυχόν ξεπερασμένα δεδομένα. Αυτό το βήμα είναι απαραίτητο για τη διατήρηση της ακεραιότητας και της συνάφειας των δεδομένων, ιδιαίτερα σημαντικό σε ένα περιβάλλον τάξης όπου οι πληροφορίες αλλάζουν γρήγορα. Μετά την αρχική ρύθμιση, η δέσμη ενεργειών συμπληρώνει αυτόματα το φύλλο με σταθερές κεφαλίδες όπως "UserID", "LastName", "FirstName" και "Email", ακολουθούμενες από τις ημερομηνίες των συνεδριών της τάξης, το σύνολο των παρουσιών "Attendances" και τέλος, τον μέσο όρο του βαθμού του φοιτητή στην τάξη "Grade".

Σχήμα 4.2: Κεφαλίδες Στοιχείων Τάξης

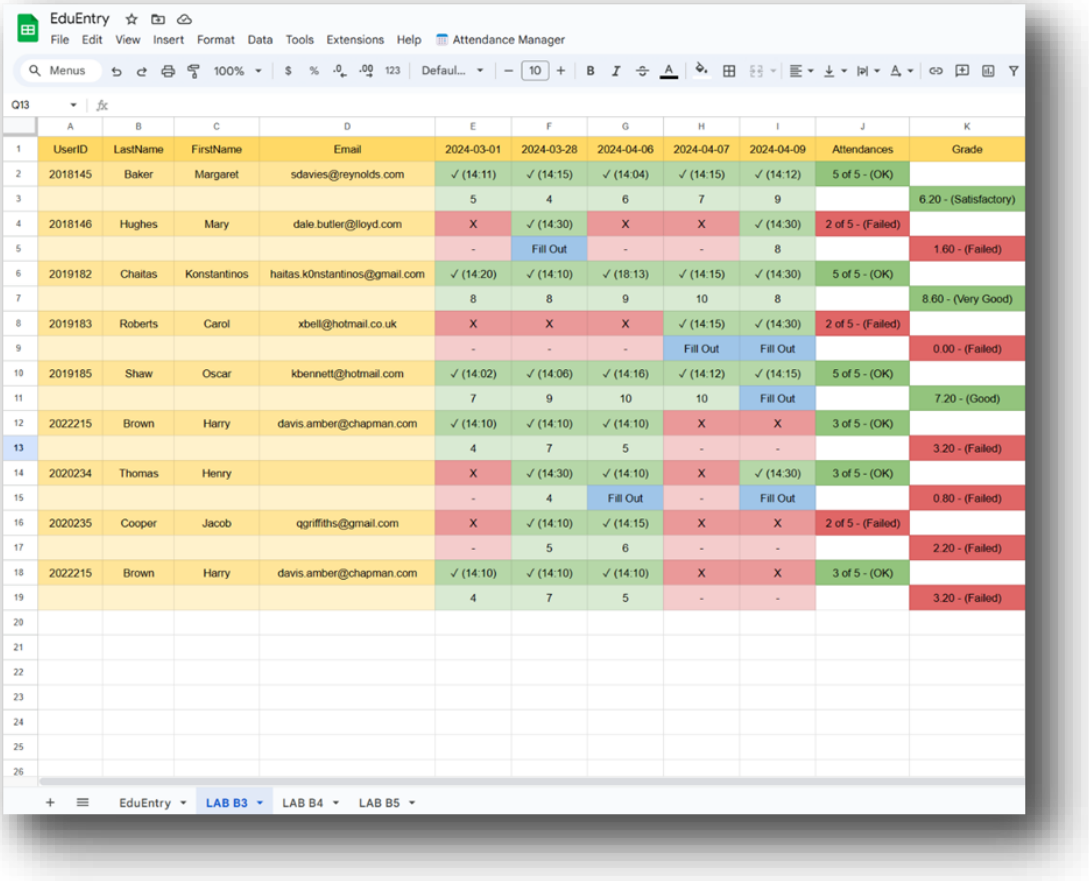
Αυτές οι κεφαλίδες δημιουργούν ένα οργανωμένο πλαίσιο διασφαλίζοντας επίσης και ότι όλοι οι ενδιαφερόμενοι βρίσκονται στην ίδια σελίδα όσον αφορά την ερμηνεία και την εισαγωγή δεδομένων για την μετέπειτα σωστή αποθήκευση αυτών στην βάση δεδομένων του Firebase.

Έπειτα η δέσμη ενεργειών ανακτά τα δεδομένα του φοιτητή από το Firestore. Με βάση το αναγνωριστικό τάξης ανακτάται η υποσυλλογή των φοιτητών που ανήκουν στο στιγμιότυπό της. Για κάθε φοιτητή, αντλούμε μια σειρά δεδομένων που αποτελείται από τα στοιχεία ταυτοποίησής του, τις συμμετοχές του και τους βαθμούς του σε κάθε συνεδρία. Τα στιγμιότυπα της υποσυλλογής παρακολούθησης του φοιτητή ανακτώνται και σημειώνονται για κάθε ημερομηνία μαθήματος, υποδεικνύοντας παρουσία ή απουσία, ενώ οι βαθμοί είτε ανακτώνται αν έχουν συμπληρωθεί, είτε εμφανίζεται ένα μήνυμα υπενθύμισης για συμπλήρωση.

UserID	LastName	FirstName	Email	2024-03-01	2024-03-28	2024-04-06	2024-04-07	2024-04-09	Attendances	Grade
2018145	Baker	Margaret	sdavies@reynolds.com	✓ (14:11)	✓ (14:15)	✓ (14:04)	✓ (14:15)	✓ (14:12)		
				5	4	6	7	9		
2018146	Hughes	Mary	dale.butler@lloyd.com	X	✓ (14:30)	X	X	✓ (14:30)		
				-	Fill Out	-	-	8		
2019182	Chaitas	Konstantinos	haitas.k0nstantinos@gmail.com	✓ (14:20)	✓ (14:10)	✓ (18:13)	✓ (14:15)	✓ (14:30)		
				8	8	9	10	8		
2019183	Roberts	Carol	xbell@hotmail.co.uk	X	X	X	✓ (14:15)	✓ (14:30)		
				-	-	-	Fill Out	Fill Out		
2019185	Shaw	Oscar	kbennett@hotmail.com	✓ (14:02)	✓ (14:06)	✓ (14:16)	✓ (14:12)	✓ (14:15)		
				7	9	10	10	Fill Out		
2022215	Brown	Harry	davis.amber@chapman.com	✓ (14:10)	✓ (14:10)	✓ (14:10)	X	X		
				4	7	5	-	-		
2020234	Thomas	Henry		X	✓ (14:30)	✓ (14:10)	X	✓ (14:30)		
				-	4	Fill Out	-	Fill Out		
2020235	Cooper	Jacob	qgriffiths@gmail.com	X	✓ (14:10)	✓ (14:15)	X	X		
				-	5	6	-	-		
2022215	Brown	Harry	davis.amber@chapman.com	✓ (14:10)	✓ (14:10)	✓ (14:10)	X	X		
				4	7	5	-	-		

Σχήμα 4.3: Στοιχεία Φοιτητή, Παρουσίες και Βαθμοί

Αφού επισυναφθούν όλα τα δεδομένα των μαθητών, το σενάριο υπολογίζει συνοπτικές πληροφορίες για τη παρακολούθηση και τους βαθμούς. Εφαρμόζεται επίσης μορφοποίηση στο φύλλο για καλύτερη ανάγνωση και κατανόηση της πληροφορίας, όπως ρύθμιση πλάτους στηλών και εφαρμογή στοίχισης κειμένου. Στη συνέχεια, εφαρμόζεται χρωματική κωδικοποίηση στα δεδομένα παρακολούθησης και βαθμολογίας για οπτική ευκολία, για παράδειγμα χρησιμοποιώντας πράσινο για να δηλώσει την παρουσία, κόκκινο για απουσία, όπως και άλλα χρώματα για βαθμούς ή ενέργειες που απαιτούνται.



	A	B	C	D	E	F	G	H	I	J	K
1	UserID	LastName	FirstName	Email	2024-03-01	2024-03-28	2024-04-06	2024-04-07	2024-04-09	Attendances	Grade
2	2018145	Baker	Margaret	sdavies@reynolds.com	✓ (14:11)	✓ (14:15)	✓ (14:04)	✓ (14:15)	✓ (14:12)	5 of 5 - (OK)	
3					5	4	6	7	9		6.20 - (Satisfactory)
4	2018146	Hughes	Mary	dale.butler@lloyd.com	X	✓ (14:30)	X	X	✓ (14:30)	2 of 5 - (Failed)	
5					-	Fill Out	-	-	8		1.60 - (Failed)
6	2019182	Chaitas	Konstantinos	haitas.k0nstantinos@gmail.com	✓ (14:20)	✓ (14:10)	✓ (18:13)	✓ (14:15)	✓ (14:30)	5 of 5 - (OK)	
7					8	8	9	10	8		8.60 - (Very Good)
8	2019183	Roberts	Carol	xbell@hotmail.co.uk	X	X	X	✓ (14:15)	✓ (14:30)	2 of 5 - (Failed)	
9					-	-	-	Fill Out	Fill Out		0.00 - (Failed)
10	2019185	Shaw	Oscar	kbennett@hotmail.com	✓ (14:02)	✓ (14:06)	✓ (14:16)	✓ (14:12)	✓ (14:15)	5 of 5 - (OK)	
11					7	9	10	10	Fill Out		7.20 - (Good)
12	2022215	Brown	Harry	davis.amber@chapman.com	✓ (14:10)	✓ (14:10)	✓ (14:10)	X	X	3 of 5 - (OK)	
13					4	7	5	-	-		3.20 - (Failed)
14	2020234	Thomas	Henry		X	✓ (14:30)	✓ (14:10)	X	✓ (14:30)	3 of 5 - (OK)	
15					-	4	Fill Out	-	Fill Out		0.80 - (Failed)
16	2020235	Cooper	Jacob	qgriffiths@gmail.com	X	✓ (14:10)	✓ (14:15)	X	X	2 of 5 - (Failed)	
17					-	5	6	-	-		2.20 - (Failed)
18	2022215	Brown	Harry	davis.amber@chapman.com	✓ (14:10)	✓ (14:10)	✓ (14:10)	X	X	3 of 5 - (OK)	
19					4	7	5	-	-		3.20 - (Failed)
20											
21											
22											
23											
24											
25											
26											

Σχήμα 4.4: Όριο Απουσιών – Υπολογισμός Μέσου Όρου Βαθμού – Τελική Μορφοποίηση και Χρωματισμός

4.2.2 Παρακολούθηση Παρουσίας και Αναφορά

Τα Google Apps Scripts διευκολύνουν το σύστημα παρακολούθησης της συμμετοχής των φοιτητών σε πραγματικό χρόνο, ενσωματώνοντας δεδομένα από το Firestore στα Google Sheets. Παρακάτω θα αναλύσουμε κάποιες ενέργειες που αφορούν την αναπαράσταση στιγμιότυπων των παρουσιών. Η ημερομηνία κάθε συνεδρίας εμφανίζεται ως κεφαλίδα στήλης στο Φύλλο Google. Κάτω από αυτές τις κεφαλίδες, η συμμετοχή καταγράφεται με ένα οπτικό σύστημα. Ένα πράσινο τικ (✓) αντιπροσωπεύει την παρουσία ενός φοιτητή στη συνεδρία, συνοδευόμενη από την ώρα άφιξης. Αυτή η άμεση ανατροφοδότηση επιτρέπει στους εκπαιδευτικούς να δουν όχι μόνο ότι ένας φοιτητής παρακολούθησε το μάθημα αλλά και πότε έφτασε στην αίθουσα. Ενώ το "X" σηματοδοτεί μια απουσία, υποδεικνύοντας ότι ο φοιτητής δεν παρακολούθησε τη συνεδρία. Αυτό το απλό, δυαδικό σύστημα διευκολύνει τη σάρωση και την κατανόηση των μοτίβων συμμετοχής με μια ματιά.

Από τα δεδομένα συμμετοχής δημιουργείται μια στήλη "Attendances" που συγκεντρώνει τον συνολικό αριθμό των παρουσιών, δείχνοντας το ποσοστό συμμετοχής του φοιτητή σε σχέση με τις συνολικές συνεδρίες της τάξης ("$Attendances of $TotalSessions - OK : Failed "). Το "OK" εμφανίζεται εάν οι απουσίες είναι δύο ή λιγότερες. Αυτή η κατάσταση υποδηλώνει ότι η φοίτηση του φοιτητή είναι εντός των αποδεκτών ορίων και δεν επισημαίνεται άμεση ακαδημαϊκή ανησυχία. Ενώ το Απέτυχε "Failed" σημειώνεται όταν οι απουσίες ενός φοιτητή υπερβαίνουν τις δύο συνεδρίες. Αυτή η ετικέτα ενεργοποιεί

μια ειδοποίηση στον εκπαιδευτικό ότι το ποσοστό συμμετοχής του φοιτητή επηρεάζει αρνητικά την ακαδημαϊκή του απόδοση και χρειάζεται παρέμβαση.

Αυτό το σύστημα διασφαλίζει ότι η παρακολούθηση παρουσίας δεν είναι μόνο αυτοματοποιημένη αλλά και ενσωματωμένη με διορατικούς δείκτες ακαδημαϊκής απόδοσης. Αξιοποιώντας τη μορφοποίηση υπό όρους και τις συνοπτικές περιλήψεις, η διεπαφή των Υπολογιστικών φύλλων της Google χρησιμεύει ως αποτελεσματικό εργαλείο για τη διαχείριση και τον έλεγχο της συμμετοχής των μαθητών. Αυτή η λεπτομερής παρακολούθηση παρουσίας υποστηρίζει το ακαδημαϊκό και διοικητικό προσωπικό στη διατήρηση υψηλών εκπαιδευτικών προτύπων και στην προληπτική διαχείριση των φοιτητών.

4.2.3 Υπολογισμός και Ανάλυση Βαθμών

Στο σύστημά μας, ο υπολογισμός και η ανάλυση των βαθμών των φοιτητών αυτοματοποιείται χρησιμοποιώντας τα Google Apps Scripts. Αυτή η λειτουργία ξεκινά με την εξαγωγή των δεδομένων παρακολούθησης για κάθε φοιτητή, που αντιστοιχούν σε μεμονωμένες ημερομηνίες μαθημάτων. Ο βαθμός σε κάθε νέα παρουσία φοιτητή αρχικοποιείται με ένα μήνυμα για τον καθηγητή, που δίνει έμφαση για να τον συμπληρώσει, αν ο βαθμός δεν έχει συμπληρωθεί πριν τον υπολογισμό του μέσου όρου προσμετράται ως μηδέν. Επιπλέον η μη παρουσία ενός φοιτητή σε μια ημερομηνία που έχει πραγματοποιηθεί μια συνεδρία του μαθήματος οδηγεί πάλι σε βαθμό μηδέν. Στη συνέχεια το σενάριο προσθέτει τον συνολικό βαθμό του φοιτητή και υπολογίζει τον μέσο όρο, προσαρμόζοντας στις παραπάνω εξαιρέσεις και τον ταξινομεί σε ονομαστικούς όρους όπως «Επαρκής», «Καλός» και «Εξαιρετικός» με βάση κάποια προκαθορισμένα όρια. Για παράδειγμα, μια μέση βαθμολογία ανάμεσα στο 5 και στο 6 μπορεί να ταξινομηθεί ως «Επαρκής» ή «Εξαιρετική» αν ο φοιτητής έχει μέσο όρο 10. Αυτή η ανάλυση της βαθμολόγησης αυτοματοποιεί τη διαδικασία αξιολόγησης, αλλά παρέχει επίσης μια άμεση οπτική ανατροφοδότηση και πληροφορίες για την απόδοση. Το αποτέλεσμα που προκύπτει περιλαμβάνει μια λεπτομερή ανάλυση της φοίτησης και των βαθμών, επιτρέποντας στους εκπαιδευτικούς να κατανοήσουν γρήγορα τις ακαδημαϊκές επιδόσεις και να εντοπίσουν μαθητές που μπορεί να χρειάζονται πρόσθετη υποστήριξη ή επιβράβευση. Το σύστημα αυτό εξασφαλίζει μια αποτελεσματική προσέγγιση στην ακαδημαϊκή διαχείριση, καθιστώντας τον υπολογισμό της βαθμολογίας διαφανή και προσβάσιμο στο εκπαιδευτικό προσωπικό.

4.3 Καταγραφή Προσθηκών στο Firestore

Οι επιλογές **Update “Students” to Firestore → [Select Class]** και **Update “Students” to Firestore → [All Classes]** διαδραματίζουν κρίσιμο ρόλο στη διαχείριση παρουσίας για τον συγχρονισμό των ενημερωμένων δεδομένων φοιτητών από τα Google Sheets στο Firestore. Αυτές οι επιλογές επιτρέπουν στοχευμένες ή ολοκληρωμένες ενημερώσεις, είτε για κάποια συγκεκριμένη τάξη είτε για όλες ταυτόχρονα, ενισχύοντας την ευελιξία της διαχείρισης των δεδομένων.

Η διαδικασία ξεκινά με τον χρήστη να επιλέγει μια συγκεκριμένη τάξη ή όλες τις τάξεις, ακριβώς όπως και στις διαδικασίες συγχρονισμού που αναλύσαμε παραπάνω. Αυτή η ενεργεία ενεργοποιεί μια σειρά ενεργειών για τη λήψη των σχετικών δεδομένων από το φύλλο της τάξης που επιλέχθηκε και στη συνέχεια, ενημερώνει τα αντίστοιχα έγγραφα στο Firestore με τις πιο πρόσφατες πληροφορίες. Αυτό περιλαμβάνει τη συμμετοχή των φοιτητών, τους βαθμούς και άλλες σχετικές ακαδημαϊκές πληροφορίες. Εάν η τάξη δεν υπάρχει ήδη στο Firestore, το σενάριο δημιουργεί μια νέα καταχώρηση, δίνοντας έναν εύκολο τρόπο στους εκπαιδευτικούς να ξεκινήσουν ένα νέο μάθημα χωρίς την παρέμβαση κάποιου διαχειριστή στην ίδια την βάση δεδομένων, διασφαλίζοντας τη συνέχεια και την ακεραιότητα των

δεδομένων. Για ευρύτερες ενημερώσεις που αφορούν όλες τις τάξεις, το σενάριο επαναλαμβάνεται σε κάθε φύλλο που είναι διαθέσιμο στο Google Sheets. Επεξεργάζεται και ενημερώνει το αρχείο κάθε φοιτητή στο Firestore, έχοντας την δυνατότητα να κάνει αλλαγές όπως να προσθέσει έναν διαφορετικό τρόπο επικοινωνίας (email) αλλά μπορεί ακόμα και να δημιουργήσει ένα στιγμιότυπο νέο φοιτητή. Το τελευταίο χαρακτηριστικό δίνει την δυνατότητα στο σύστημά μας, να δουλεύει για όλους τους φοιτητές ανεξαρτήτως αν έχουν ένα τηλέφωνο Android ή σε πιο γενικό πλαίσιο ένα τηλέφωνο μαζί τους στην τάξη. Όλες αυτές οι ενέργειες διασφαλίζουν ότι όλες οι εγγραφές είναι ενημερωμένες και αντικατοπτρίζουν τον πραγματικό κόσμο και τις πιο πρόσφατες αλλαγές που έγιναν στα Google Sheets.

Αυτός ο αυτοματοποιημένος συγχρονισμός ελαχιστοποιεί τα σφάλματα μη αυτόματης εισαγωγής δεδομένων και μειώνει σημαντικά τον φόρτο εργασίας για το ακαδημαϊκό προσωπικό, καθιστώντας την όλη διαδικασία πιο αποτελεσματική. Αυτή η ισχυρή ενσωμάτωση μεταξύ των Google Sheets και του Firestore έχει στόχο να διευκολύνει τη διαχείριση δεδομένων και την προσβασιμότητα σε πραγματικό χρόνο, ζωτικής σημασίας για δυναμικά εκπαιδευτικά περιβάλλοντα.

4.4 Προκλήσεις – Λύσεις και Μελλοντικές Εξελίξεις

Τα G.A.S. προσφέρουν ισχυρές δυνατότητες αυτοματισμού εντός του Google Workspace, αλλά αυτές δεν είναι χωρίς περιορισμούς, ειδικά όσον αφορά την ταχύτητα επεξεργασίας και τον χρόνο εκτέλεσης. Ακολουθούν ορισμένες από τις προκλήσεις και προσεγγίσεις για την αντιμετώπιση αυτών των ζητημάτων.

Τα G.A.S. μπορεί μερικές φορές να εκτελούνται αργά, ειδικά όταν τα σενάρια εκτελούν λειτουργίες ανά κελί σε μεγάλα υπολογιστικά φύλλα. Αυτή η λεπτομέρεια, μπορεί να επιβραδύνει σημαντικά τους χρόνους εκτέλεσης καθώς κάθε λειτουργία κελιού επεξεργάζεται διαδοχικά. Για να βελτιωθεί η απόδοση, συνιστάται η ελαχιστοποίηση των αλληλεπιδράσεων με μεμονωμένα κελιά. Αυτό συμβαίνει στην υλοποίησή μας, αποθηκεύοντας μεγάλα εύρη δεδομένων στη μνήμη ταυτόχρονα, επεξεργάζοντάς τα μέσα στο σενάριο και γράφοντάς τα μαζικά. Αυτό μειώνει τον αριθμό των λειτουργιών ανάγνωσης/εγγραφής που εκτελούνται στο φύλλο, επιταχύνοντας σημαντικά το σενάριο. Όμως υπάρχουν σημεία όπως ο χρωματισμός υπό κανόνες στα οποία αυτή η αντιμετώπιση δεν μπορεί να λειτουργήσει και απαιτείται η εύρεση κάποιας άλλης τεχνικής για την αντιμετώπιση του προβλήματος σε αυτά τα σημεία .

Τα G.A.S. έχουν χρονικό όριο εκτέλεσης (επί του παρόντος έως 6 λεπτά για λογαριασμούς καταναλωτών και περισσότερο για λογαριασμούς Workspace), το οποίο μπορεί να αποτελέσει σημαντικό εμπόδιο για εκτεταμένες εργασίες επεξεργασίας δεδομένων σε μεγάλα τμήματα φοιτητών. Για να επιλυθεί αυτό το πρόβλημα, οι δέσμες ενεργειών έχουν σχεδιαστεί ώστε να μπορούν να εκτελούνται τμηματικά. Χρησιμοποιώντας triggers βάσει χρόνου ή βάσει συμβάντων, η δέσμη ενεργειών θα μπορούσε να επεξεργαστεί δεδομένα σταδιακά. Κάθε trigger μπορεί να συνεχίσει από εκεί που σταμάτησε το τελευταίο, κατανέμοντας έτσι το φορτίο σε πολλές εκτελέσεις αντί η δέσμη να προσπαθεί να επεξεργαστεί τα πάντα με μία κίνηση.

Καθώς αυξάνεται η κλίμακα των δεδομένων και η πολυπλοκότητα των λειτουργιών, οι δέσμες ενεργειών ενδέχεται να γίνουν πιο δύσκολες στη διαχείριση και πιο επιρρεπείς σε σφάλματα ή χρονικά όρια. Η αναδιαμόρφωση δεσμών ενεργειών για τη βελτιστοποίηση της αποτελεσματικότητας, η χρήση χειρισμού σφαλμάτων για τη διαχείριση και την επανάληψη αποτυχημένων λειτουργιών και ο διαχωρισμός μεγάλων εργασιών σε μικρότερες, πιο διαχειρίσιμες λειτουργίες μπορούν να βοηθήσουν στη διατήρηση της επεκτασιμότητας. Επιπλέον, η χρήση του BigQuery της Google [15] ή η ενσωμάτωση με πιο ισχυρές λύσεις βάσεων δεδομένων μπορεί να μειώσει τον φόρτο των υπολογισμών

από τα Google Apps Scripts [16]. Αντιμετωπίζοντας αυτές τις προκλήσεις με στρατηγικές τεχνικές διαχείρισης σεναρίων, τα ιδρύματα μπορούν να αξιοποιήσουν καλύτερα τα G.A.S. για να καλύψουν τις ανάγκες τους χωρίς να περιορίζονται υπερβολικά από τους εγγενείς περιορισμούς της πλατφόρμας. Η συνεχής μάθηση και η προσαρμογή στις εξελισσόμενες δυνατότητες είναι επίσης ζωτικής σημασίας για τη μεγιστοποίηση της αποτελεσματικότητας αυτών των εργαλείων στην ακαδημαϊκή διοίκηση.

4.5 Σύνοψη

Σε αυτό το κεφάλαιο, διερευνήσαμε τον σημαντικό ρόλο που διαδραματίζουν τα Google Apps Scripts στον τομέα της ακαδημαϊκής διοίκησης. Ξεκινώντας με μια ευρεία επισκόπηση, συζητήσαμε πώς τα G.A.S. εξέλιξαν τα υπολογιστικά φύλλα σε μια ισχυρή πλατφόρμα για σύνθετη διαχείριση δεδομένων και αυτοματοποίηση ροής εργασιών χρησιμοποιώντας την JavaScript. Η εξέλιξη αυτή αντικατοπτρίζει τη στροφή των ακαδημαϊκών ιδρυμάτων προς αποδοτικότερες επιχειρησιακές πρακτικές που βασίζονται περισσότερο στα δεδομένα.

Επισημάνθηκαν βασικές λειτουργίες των G.A.S. , όπως η αυτοματοποίηση εργασιών εισαγωγής δεδομένων, η ενσωμάτωση διαφόρων υπηρεσιών της Google και η διευκόλυνση ενημερώσεων και αναφορών σε πραγματικό χρόνο. Αυτά τα χαρακτηριστικά βελτιώνουν την παραγωγικότητα του διοικητικού προσωπικού ενισχύοντας την ακρίβεια και την προσβασιμότητα των ακαδημαϊκών δεδομένων, υποστηρίζοντας έτσι τις διαδικασίες λήψης αποφάσεων. Αντιμετωπίστηκαν οι προκλήσεις που σχετίζονται με τις επιδόσεις, συμπεριλαμβανομένης της ταχύτητας επεξεργασίας και των χρονικών ορίων εκτέλεσης, μαζί με πιθανές λύσεις για τον μετριασμό αυτών των ζητημάτων. Τεχνικές όπως η επεξεργασία δέσμης, η εκτέλεση τμηματικά βάσει triggers και η ελαχιστοποίηση των λειτουργιών ανά κελί συμβάλλουν στη βελτιστοποίηση της απόδοσης των ενεργειών και στην αποτελεσματική διαχείριση μεγάλων συνόλων δεδομένων. Επιπλέον, συζητήσαμε τις μελλοντικές εξελίξεις και τις δυνατότητες συνεχής βελτίωσης των Google Apps Scripts. Καθώς τα ακαδημαϊκά ιδρύματα βασίζονται όλο και περισσότερο στην τεχνολογία, η δυνατότητα ενσωμάτωσης των G.A.S. με πιο ισχυρές βάσεις δεδομένων και εργαλεία ανάλυσης, όπως το BigQuery της Google, υποδηλώνει μια πολλά υποσχόμενη κατεύθυνση για την επέκταση της χρησιμότητας και του πεδίου εφαρμογής τους σε εκπαιδευτικά περιβάλλοντα.

Αυτό το κεφάλαιο υπογραμμίζει τον κεντρικό ρόλο που διαδραματίζουν τα Google Apps Scripts στον μετασχηματισμό της ακαδημαϊκής διοίκησης, αυτοματοποιώντας και απλοποιώντας τις λειτουργίες, συμβάλλοντας τελικά στην ικανότητα του εκπαιδευτικού ιδρύματος να προσαρμόζεται και να ευδοκιμεί σε ένα συνεχώς εξελισσόμενο τεχνολογικό τοπίο.

Κεφάλαιο 5ο: Raspberry Pi για Έλεγχο Παρουσίας

5.1 Εισαγωγή

Το Raspberry Pi είναι μια σειρά μικρών single-board υπολογιστών που αναπτύχθηκε από το Ίδρυμα Raspberry Pi στο Ηνωμένο Βασίλειο. Αρχικά σχεδιάστηκε για να προωθήσει τη διδασκαλία της βασικής επιστήμης των υπολογιστών στα σχολεία και στις αναπτυσσόμενες χώρες, έχει όμως αποκτήσει τεράστια δημοτικότητα σε ένα ευρύ φάσμα επαγγελματικών και ερασιτεχνικών έργων λόγω του χαμηλού κόστους, της ευελιξίας και του ανοιχτού σχεδιασμού. Κάθε μοντέλο Raspberry Pi έχει περίπου το μέγεθος μιας πιστωτικής κάρτας και εμπεριέχει την επεξεργαστική ισχύ, τη μνήμη και τη συνδεσιμότητα που θα βρίσκαμε συνήθως σε έναν επιτραπέζιο ή φορητό υπολογιστή. Αυτό περιλαμβάνει μια κεντρική μονάδα επεξεργασίας που βασίζεται στην αρχιτεκτονική ARM, διάφορα μεγέθη μνήμης RAM (που κυμαίνονται από 256MB έως και 8GB στα πιο πρόσφατα μοντέλα), πολλαπλές θύρες USB για περιφερειακά, Bluetooth, Ethernet και Wi-Fi για σύνδεση στο διαδίκτυο. Το μικρό του μέγεθος και τα πλούσια χαρακτηριστικά του, καθιστούν το Raspberry Pi ιδανικό υποψήφιο για έργα όπου ο χώρος και το κόστος είναι περιορισμένα, αλλά όπου εξακολουθεί να απαιτείται ισχυρή υπολογιστική ισχύ.

Σε αυτό το κεφάλαιο, διερευνούμε τη χρήση του Raspberry Pi ως βασικού στοιχείου για την ανάπτυξη ενός αποτελεσματικού συστήματος ελέγχου παρουσίας για τα ακαδημαϊκά ιδρύματα. Αυτό το κεφάλαιο στοχεύει να παρέχει μια ολοκληρωμένη επισκόπηση της ενσωμάτωσης του Raspberry Pi στην καταγραφή και τη διαχείριση της παρουσίας των φοιτητών μέσω σάρωσης κωδικού QR. Η ευελιξία και η ικανότητά του να υποστηρίζει διάφορα περιβάλλοντα προγραμματισμού και περιφερειακές συσκευές το καθιστούν ιδανική επιλογή για την εφαρμογή μας. Θα ξεκινήσουμε συζητώντας την επιλογή του μοντέλου Raspberry Pi που ταιριάζει καλύτερα στις απαιτήσεις του προβλήματός μας. Τα κριτήρια για αυτήν την επιλογή περιλαμβάνουν την επεξεργαστική ισχύ, τη χωρητικότητα μνήμης και τις επιλογές συνδεσιμότητας, οι οποίες είναι κρίσιμες για το χειρισμό των εργασιών που εμπλέκονται στην επεξεργασία εικόνας και τη διαχείριση δεδομένων σε πραγματικό χρόνο. Μετά την επιλογή του υλικού, θα εμβαθύνουμε στα λεπτομερή βήματα διαμόρφωσης που απαιτούνται για την προετοιμασία του Raspberry Pi για ανάπτυξη. Αυτό περιλαμβάνει τη ρύθμιση του λειτουργικού συστήματος, τη διαμόρφωση των ρυθμίσεων δικτύου και τη διασφάλιση ασφαλούς πρόσβασης στη συσκευή, θέτοντας μια σταθερή βάση για τα στοιχεία λογισμικού που θα χρησιμοποιήσουμε. Στις επόμενες ενότητες θα ασχοληθούμε με τη ρύθμιση του λογισμικού, ξεκινώντας με την εγκατάσταση βιβλιοθηκών και τη διαμόρφωση του λογισμικού σάρωσης κώδικα QR. Αυτό το λογισμικό είναι αναπόσπαστο μέρος του συστήματος, επιτρέποντας στο Raspberry Pi να διαβάζει και να μεταφράζει κωδικούς QR που παρουσιάζονται από τους φοιτητές κατά την είσοδό τους στην τάξη. Επιπλέον, θα εμβαθύνουμε στο πώς το Raspberry Pi επεξεργάζεται και καταγράφει τα δεδομένα παρουσίας στην βάση δεδομένων του Firebase. Τέλος, το κεφάλαιο θα καλύψει τις προκλήσεις που αντιμετωπίστηκαν κατά την εφαρμογή του συστήματος παρακολούθησης του Raspberry Pi. Αυτές οι προκλήσεις κυμαίνονται από περιορισμούς υλικού έως σφάλματα λογισμικού και ζητήματα διεπαφής χρήστη. Θα παρουσιαστούν λύσεις που αναπτύχθηκαν για την αντιμετώπιση αυτών των προκλήσεων, παρέχοντας πολύτιμες πληροφορίες σχετικά με τις πρακτικές πτυχές της ανάπτυξης ενός συστήματος που βασίζεται στο Raspberry Pi σε εκπαιδευτικό περιβάλλον.

Συνοπτικά, αυτό το κεφάλαιο όχι μόνο περιγράφει λεπτομερώς την τεχνική εφαρμογή του Raspberry Pi για τον έλεγχο παρουσίας, αλλά αναδεικνύει επίσης τις πρακτικές προκλήσεις και τα μαθησιακά αποτελέσματα που σχετίζονται με την ανάπτυξη μιας λύσης που βασίζεται στην τεχνολογία σε ένα εκπαιδευτικό περιβάλλον. Μέσω αυτής της συζήτησης, στοχεύουμε να δείξουμε τη σκοπιμότητα και τα οφέλη της χρήσης της τεχνολογίας Raspberry Pi για την ενίσχυση των διοικητικών διαδικασιών εντός των ακαδημαϊκών ιδρυμάτων.

5.2 Εγκατάσταση και Διαμόρφωση Υλικού

Αυτή η ενότητα υπογραμμίζει τη σημασία της σωστής ρύθμισης του υλικού για την κάλυψη των συγκεκριμένων αναγκών του έργου, διασφαλίζοντας ότι η συσκευή λειτουργεί αποτελεσματικά και αποδοτικά σε ένα εκπαιδευτικό περιβάλλον. Ξεκινά με την επιλογή του κατάλληλου μοντέλου Raspberry Pi που εξισορροπεί την επεξεργαστική ισχύ, τη χωρητικότητα μνήμης και τις επιλογές συνδεσιμότητας για να χειριστεί τις απαιτήσεις επεξεργασίας δεδομένων σε πραγματικό χρόνο. Στη συνέχεια, η ενότητα μεταβαίνει στη λεπτομερή περιγραφή των βασικών βημάτων διαμόρφωσης που είναι απαραίτητα για τη λειτουργεία του Raspberry Pi. Περιγράφοντας αυτές τις διαδικασίες, αυτή η ενότητα θέτει τις βάσεις για το Raspberry Pi να λειτουργήσει ως αξιόπιστο στοιχείο στην ευρύτερη αρχιτεκτονική του συστήματος. Η εγκατάσταση και η διαμόρφωση του υλικού δεν αφορούν μόνο τη ρύθμιση του ίδιου του Raspberry Pi, αλλά περιλαμβάνουν επίσης την ενσωμάτωση πρόσθετων μονάδων που επεκτείνουν τη λειτουργικότητα του συστήματος. Μια επιλογή πρόσθετου υλικού επιλέχθηκε για τη συμβατότητα και τη χρησιμότητά τους στην ενίσχυση του συστήματός μας.

Μια τέτοια προσθήκη είναι το Gravity Digital Buzzer από την DFROBOT, μια μονάδα ικανή να παράγει μια σειρά τόνων με βάση τη συχνότητα του σήματος εισόδου. Η υλοποίησή του στο πλαίσιο του έργου εξυπηρετεί την παροχή ηχητικής ανατροφοδότησης κατά τη διάρκεια της διαδικασίας σάρωσης του κωδικού QR.

Σχήμα 5.1: Gravity: Digital Buzzer for Raspberry Pi [4]

Ένα άλλο βασικό στοιχείο είναι το LED Traffic Light Display Module 5V από έναν κατασκευαστή OEM. Η ενσωμάτωση του χρησιμεύει ως ένας απλός οπτικός δείκτης της κατάστασης του συστήματος - πράσινο για λειτουργία, κίτρινο για επεξεργασία και κόκκινο για σφάλματα ή ζητήματα. Ο φιλικός προς ένα breadboard σχεδιασμός του διευκολύνει την ενσωμάτωση με το Raspberry Pi.

Σχήμα 5.2: LED Traffic Light Display Module 5V [6]

Τέλος, το Raspberry Pi Camera Module V3 αντιπροσωπεύει μια σημαντική πρόοδο λήψης εικόνων σε αυτόν τον τομέα της τεχνολογίας, με τον αισθητήρα της Sony IMX708. Ο οποίος προσφέρει βελτιωμένη ευαισθησία σε χαμηλό φωτισμό, υποστήριξη HDR και αυτόματη εστίαση. Είτε μέσω φωτογραφιών είτε μέσω βίντεο, η έξοδος αυτής της μονάδας είναι ζωτικής σημασίας για την ακριβή σάρωση κωδικών QR και για άλλες προσθήκες του συστήματος μελλοντικά.

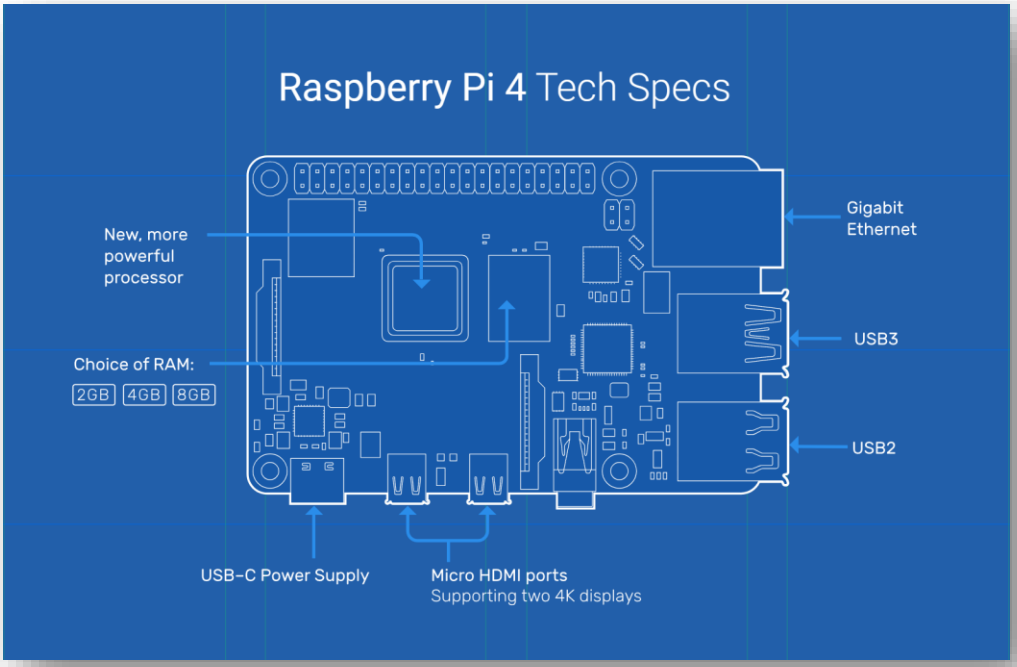
Σχήμα 5.3: Raspberry Pi Camera Module V3 - Standard 11.9MP 75° [7]

Κάθε ένα από αυτά τα στοιχεία επιλέχθηκε όχι μόνο για τις ατομικές τους δυνατότητες, αλλά και για το πώς συνεργάζονται πάνω στο Raspberry Pi. Η συνέργεια μεταξύ του Raspberry Pi και αυτών των εξαρτημάτων δημιουργεί ένα πιο ισχυρό σύστημα ελέγχου παρουσίας, υπογραμμίζοντας τη σημασία ενός καλά μελετημένου οικοσυστήματος υλικού που λειτουργεί με αρμονία.

5.2.1 Επιλογή Μοντέλου Raspberry Pi

Στα αρχικά στάδια ανάπτυξης του συστήματος, αποφάσισα να πειραματιστώ με το τελευταίο μοντέλο της σειράς το Raspberry Pi 5 8GB που εντάχθηκε στην αγορά τον Οκτώβριο του 2023, αναμένοντας ότι τα ανανεωμένα του χαρακτηριστικά θα ενισχύσουν την απόδοση του συστήματος. Ωστόσο, κατά τη διάρκεια της υλοποίησης, έγινε προφανές ότι το νέο λειτουργικό σύστημα που χρησιμοποιεί το Raspberry Pi 5 και ορισμένες βιβλιοθήκες απαραίτητες για το έργο δεν ήταν πλήρως συμβατές ή αρκετά σταθερές για τις συγκεκριμένες ανάγκες μας. Αυτό οδήγησε σε απροσδόκητες προκλήσεις που επηρέασαν τη διαδικασία ανάπτυξης. Μετά την αξιολόγηση αυτών των επιπλοκών, ελήφθη η απόφαση να επιστρέψω στο Raspberry Pi 4 Model B. Αρχικά, η έκδοση των 8GB αυτού του μοντέλου δοκιμάστηκε για να διασφαλιστεί ότι υπήρχε άφθονη μνήμη για τις απαιτήσεις της εφαρμογής. Ωστόσο, μετά από περαιτέρω δοκιμές και βελτιστοποίηση του κώδικα, έγινε προφανές ότι

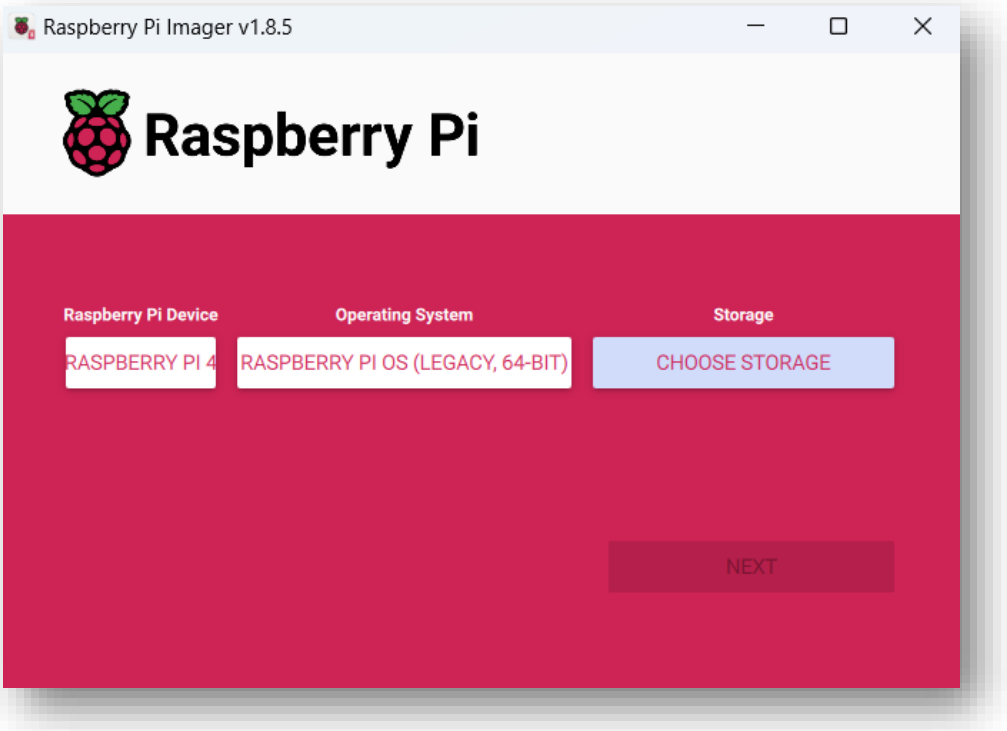
το σύστημα χρησιμοποίησε λιγότερους πόρους από τους αναμενόμενους. Κατά συνέπεια, επιλέχθηκε το Raspberry Pi 4B με 4GB μνήμης RAM καθώς παρείχε τη βέλτιστη ισορροπία απόδοσης, χρήσης και πόρων. Αυτό το μοντέλο αντεπεξήλθε ικανοποιητικά έχοντας την απαραίτητη επεξεργαστική ισχύ για τον χειρισμό δεδομένων σε πραγματικό χρόνο και την αναγνώριση κώδικα QR χωρίς τους πλεονάζοντες πόρους και το κόστος που σχετίζεται με το μοντέλο των υψηλότερων προδιαγραφών, αποδεικνύοντας τελικά ότι είναι η πιο αποτελεσματική και οικονομικά αποδοτική λύση για το σύστημα ελέγχου παρουσίας.



Σχήμα 5.4: Raspberry Pi 4 Model B Tech Specifications [3]

5.2.2 Βήματα Διαμόρφωσης

Παρακάτω θα αναλύσουμε τη διαδικασία προετοιμασίας της κάρτας SD για την εγκατάσταση του λειτουργικού συστήματος του Raspberry Pi και την ρύθμισης της κάμερας. Ξεκινώντας με το Raspberry Pi Imager, το επιλεγμένο OS image αντιγράφεται σε μια κάρτα SD μαζί με το αρχικό configuration και χρησιμοποιείται ως μονάδα εκκίνησης για το Raspberry Pi. Αυτό το βήμα περιλαμβάνει την επιλογή της κατάλληλης έκδοσης του λειτουργικού συστήματος Raspberry Pi (Raspberry Pi OS Full Legacy 64-bit “Bullseye”) και τον ορισμό της κάρτας SD για τη διαδικασία της εγγραφής.



Σχήμα 5.5: Raspberry Pi Imager – Setting up Operating System and Configuration

Μόλις εγκατασταθεί το λειτουργικό σύστημα, το Raspberry Pi εκκινείται και πραγματοποιείται μια αρχική ενημέρωση και αναβάθμιση χρησιμοποιώντας το εργαλείο διαχείρισης πακέτων apt για να διασφαλιστεί ότι όλα τα πακέτα του συστήματος είναι ενημερωμένα.

sudo apt update

sudo apt full-upgrade

Η διαμόρφωση συνεχίζεται με τη ρύθμιση της κάμερας, όπου απαιτούνται τροποποιήσεις στο αρχείο διαμόρφωσης εκκίνησης για να μπορέσει το Raspberry Pi να επικοινωνήσει με τη μονάδα της κάμερας.

sudo nano /boot/firmware/config.txt

Κατά την πρόσβαση στο αρχείο «config.txt» μέσω ενός προγράμματος επεξεργασίας κειμένου γραμμής εντολών, όπως το **nano**, προστίθενται συγκεκριμένες γραμμές για τη φόρτωση των προγραμμάτων οδήγησης της κάμερας.

dtoverlay=vc4-fkms-v3d

dtoverlay=imx219

Οι γραμμές «dtoverlay» είναι ζωτικής σημασίας εδώ, καθώς καθοδηγούν το σύστημα να χρησιμοποιεί συγκεκριμένα overlays που επιτρέπουν στο Raspberry Pi να διασυνδέεται σωστά με το υλικό της κάμερας.

sudo reboot

Μετά την αποθήκευση αυτών των αλλαγών και την έξοδο από το πρόγραμμα επεξεργασίας κειμένου, εκδίδεται μια εντολή επανεκκίνησης για να εφαρμόσει το σύστημα τη νέα διαμόρφωση. Μετά την

επανεκκίνηση, το Raspberry Pi είναι πλέον εξοπλισμένο με το λειτουργικό σύστημα και την απαραίτητη διαμόρφωση κάμερας για να προχωρήσει στη ρύθμιση του συστήματος ελέγχου παρουσίας, με τη μονάδα κάμερας έτοιμη για εργασίες σάρωσης κώδικα QR.

5.3 Διαμόρφωση Λογισμικού

Για να δημιουργήσουμε τη βάση του λογισμικού του συστήματός μας, χρησιμοποιούμε μια σειρά βιβλιοθηκών της Python που αλληλεπιδρούν με το υλικό του Raspberry Pi και με άλλες εξωτερικές υπηρεσίες. Η βιβλιοθήκη «**RPi.GPIO**» παρέχει τα απαραίτητα μέσα για τον έλεγχο των GPIO Pins, απαραίτητη για το χειρισμό των LED και του Buzzer που χρησιμοποιούμε.

Η διαδικασία σάρωσης του κώδικα QR διευκολύνεται από τη βιβλιοθήκη «pyzbar» που λειτουργεί σε συνδυασμό με το «**Pillow**», ένα fork της **PIL** (Python Imaging Library), για την επεξεργασία εικόνων που λαμβάνονται από την κάμερα. Η λειτουργία αποκωδικοποίησης από το «**pyzbar**» εξάγει δεδομένα από τις εικόνες, επιτρέποντας στο σύστημα να ερμηνεύσει τους κωδικούς QR.

Η διαχείριση της επικοινωνίας με το Firebase για αποθήκευση και ανάκτηση δεδομένων γίνεται μέσω custom modules, έχουμε δημιουργήσει το "**firebase_auth**" για τον έλεγχο ταυτότητας και το "**firebase_ops**" για τις διάφορες λειτουργίες της βάσης δεδομένων. Το "**firebase_auth**" χρησιμοποιεί τη βιβλιοθήκη «**jwt**» για τη δημιουργία JSON Web Tokens για πιστοποιημένες συνεδρίες με το Firebase, ενώ το "**firebase_ops**" χειρίζεται τις περίπλοκες λεπτομέρειες των λειτουργιών **CRUD** (create, read, update and delete) στη βάση δεδομένων. Για τη διευκόλυνση αυτών των λειτουργιών, η βιβλιοθήκη «**urllib.requests**» χρησιμοποιείται για τον χειρισμό των αιτημάτων **HTTP** (Hypertext Transfer Protocol), μια αναγκαιότητα για την αλληλεπίδραση με το Firebase REST API. Η ικανότητα του λογισμικού να προωθεί και να αντλεί δεδομένα από το cloud εξαρτάται από αυτά τα ασφαλή και αξιόπιστα αιτήματα.

Για να λειτουργήσει σωστά το ο κώδικάς μας, αυτές οι βιβλιοθήκες πρέπει να εγκατασταθούν στο Raspberry Pi. Αν και βιβλιοθήκες όπως το «**RPi.GPIO**» είναι συχνά προεγκατεστημένες, άλλες βιβλιοθήκες μπορούν να ληφθούν μέσω του διαχειριστή πακέτων της Python, pip ή pip3.

Ακολουθούν οι εντολές για την εγκατάσταση των απαιτούμενων βιβλιοθηκών:

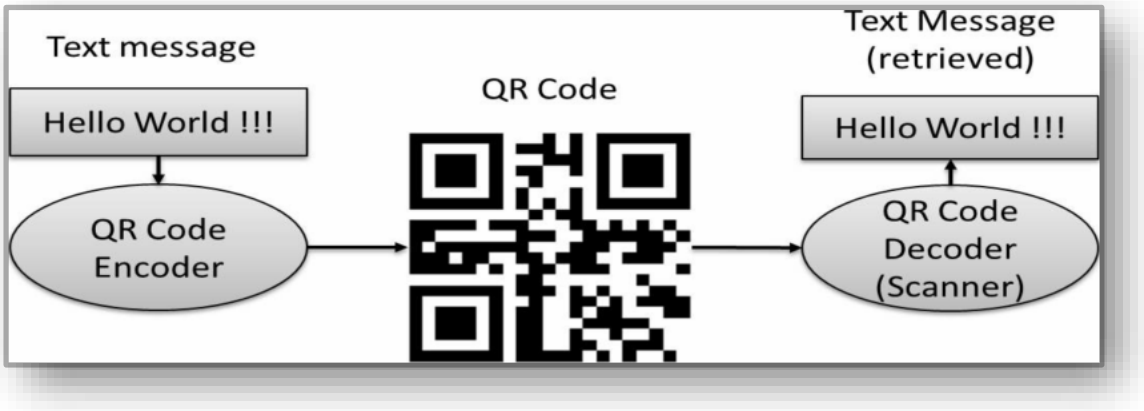
pip install pyzbar pillow PyJWT requests

5.3.1 Σάρωση, Αποκωδικοποίηση και Αποκρυπτογράφηση QR Κώδικα

Η διαδικασία ξεκινά με το πράσινο LED να ενεργοποιείται για να δείξει στον χρήστη ότι η συσκευή είναι έτοιμη για χρήση και στη συνέχεια γίνεται λήψη μιας εικόνας χρησιμοποιώντας την κάμερα του Raspberry Pi. Αυτή η εργασία εκτελείται από τη λειτουργία «**libcamera-still**» η οποία ενεργοποιείται για τη λήψη φωτογραφίας χωρίς προεπισκόπηση, μια μέθοδος που διατηρεί τους πόρους του συστήματος και είναι κατάλληλη για μια headless εγκατάσταση όπου δεν είναι συνδεδεμένη μια οθόνη. Η εικόνα αποθηκεύεται σε μια καθορισμένη διαδρομή στην RAM "**/dev/shm/**", καθώς πρόκειται για μια προσωρινή αποθήκευση. Η χρήση της RAM γίνεται για την υπεροχή της σε ταχύτητα σε σχέση με την κάρτα SD και με αυτόν τον τρόπο αποφεύγονται και οι περιττοί κύκλοι εγγραφής που θα μπορούσαν να μειώσουν τη διάρκεια ζωής της κάρτας SD.

Μόλις αποθηκευτεί η εικόνα στη μνήμη RAM, χρησιμοποιείται η βιβλιοθήκη «**pyzbar**» για να σαρώσει την εικόνα για κωδικούς QR και η βιβλιοθήκη «**Pillow**» για να χειριστεί το αρχείο της εικόνας. Η διαδικασία αποκωδικοποίησης μεταφράζει τα μοτίβα του κώδικα QR σε μια σειρά δεδομένων, η οποία περιέχει πολύτιμες πληροφορίες για το σύστημα ελέγχου παρουσίας. Το πλεονέκτημα της χρήσης της

μνήμης RAM για προσωρινή αποθήκευση είναι ιδιαίτερα εμφανές εδώ. Η δυνατότητα γρήγορης ανάγνωσης-εγγραφής επιταχύνει τη διαδικασία σάρωσης και η πτητική φύση της μνήμης RAM έχει ως αποτέλεσμα τα δεδομένα να μην αποθηκεύονται, κάτι που βοηθάει με την προστασία της ιδιωτικότητας και τη διαχείριση του χώρου.



Σχήμα 5.6: Working (overview) of QR code [11]

Αφού γίνει η λήψη και η αποκωδικοποίηση των QR κωδικών, τα δεδομένα που εξάγονται είναι κρυπτογραφημένα για να εξασφαλιστεί η ασφάλεια και η προστασία των δεδομένων. Η ενσωμάτωση της αποκρυπτογράφησης στη διαδικασία εξασφαλίζει ότι οι πληροφορίες παραμένουν ασφαλείς κατά τη μετάδοση και την επεξεργασία, παρέχοντας μια στρώση προστασίας από ανεπιθύμητη πρόσβαση ή τροποποίηση. Η διαδικασία αποκρυπτογράφησης είναι κρίσιμη για τη μετατροπή των κρυπτογραφημένων δεδομένων πίσω σε μια χρήσιμη και κατανοητή μορφή. Χρησιμοποιείται ο αλγόριθμος **AES** (Advanced Encryption Standard) με τη ρύθμιση **CBC** (Cipher Block Chaining) για την αποκρυπτογράφηση, με τη χρήση των βιβλιοθηκών «**Crypto.Cipher**» και «**base64**».Ο AES είναι ένας συμμετρικός αλγόριθμος κρυπτογράφησης, που χρησιμοποιείται ευρέως για την ασφαλή κρυπτογράφηση δεδομένων. Προτάθηκε αρχικά από τους Vincent Rijmen και Joan Daemen και έγινε διεθνές πρότυπο από την **NIST** (National Institute of Standards and Technology) των ΗΠΑ το 2001 [1]. Ο AES λειτουργεί με κλειδιά κρυπτογράφησης που μπορεί να έχουν μήκος 128, 192, ή 256 bit, προσφέροντας έναν υψηλό βαθμό ασφάλειας [13]. Το CBC είναι ένας τρόπος λειτουργίας για τους συμμετρικούς αλγορίθμους κρυπτογράφησης, όπως και ο AES. Στον CBC, κάθε μπλοκ κειμένου που πρέπει να κρυπτογραφηθεί συνδυάζεται με το προηγούμενο κρυπτογραφημένο μπλοκ πριν εφαρμοστεί ο αλγόριθμος κρυπτογράφησης. Το πρώτο μπλοκ δεδομένων συνδυάζεται με ένα αρχικό διάνυσμα **IV** (Initialization Vector), το οποίο πρέπει να είναι τυχαίο και μοναδικό για κάθε εκτέλεση της κρυπτογράφησης, για να εξασφαλιστεί ότι το κρυπτογραφημένο αποτέλεσμα δεν είναι προβλέψιμο [13]. Αυτή η μέθοδος εξασφαλίζει ότι ακόμα και αν τα αντίγραφα του ίδιου κειμένου κρυπτογραφηθούν ξανά, τα κρυπτογραφημένα αποτελέσματα θα είναι διαφορετικά, προσθέτοντας ένα επιπλέον στρώμα ασφάλειας. Στο πλαίσιο της αποκρυπτογράφησης δεδομένων στο σύστημα μας, ο συνδυασμός AES και CBC παρέχει μια ισχυρή λύση που διασφαλίζει την ασφάλεια των δεδομένων κατά τη μετάδοση και την αποθήκευση, προστατεύοντάς τα από ανεπιθύμητη πρόσβαση ή τροποποίηση.

Τα αποκωδικοποιημένα και αποκρυπτογραφημένα δεδομένα που προκύπτουν έχουν την μορφή **JSON** (JavaScript Object Notation). Το JSON είναι μια δομή ανταλλαγής δεδομένων που χαρακτηρίζεται για την ευανάγνωστη μορφή του, καθιστώντας το ιδανικό για ανταλλαγή δεδομένων μεταξύ εφαρμογών

ιστού και διακομιστών. Η απλότητα του και η δομή του το επιτρέπουν να είναι ανεξάρτητο από την πλατφόρμα που χρησιμοποιείται, διασφαλίζοντας της ανταλλαγές δεδομένων σε διαφορετικά συστήματα. Η συμβατότητα του JSON με τις τεχνολογίες ιστού και η ικανότητά του να αναπαριστά πολύπλοκες δομές δεδομένων όπως αντικείμενα και πίνακες ενισχύουν τη χρησιμότητά του σε περιβάλλοντα ιστού, όπου η απόδοση και το bandwidth είναι ζωτικής σημασίας [12]. Ιδιαίτερα στις βάσεις δεδομένων NoSQL όπως το Firestore που προσανατολίζονται σε Documents, η χωρίς σχήμα φύση του JSON επιτρέπει την ευέλικτη και δυναμική μοντελοποίηση των δεδομένων, φιλοξενώντας ποικίλες δομές δεδομένων χωρίς την ανάγκη κάποιου σταθερού σχήματος. Αυτή η προσαρμοστικότητα είναι ζωτικής σημασίας για εφαρμογές που ασχολούνται με διαφορετικούς τύπους δεδομένων ή ταχέως εξελισσόμενα μοντέλα δεδομένων.

Το JSON που έχουμε ανακτήσει μετα από όλη αυτή την διαδικασία, χρησιμοποιείται για τα επόμενα βήματα του συστήματος, όπου το περιεχόμενο του κώδικα QR θα υποβληθεί σε διαφορετική επεξεργασία ανάλογα με το αν αφορά φοιτητή ή εκπαιδευτικό. Αυτές οι λειτουργίες, αποτελούν τον πυρήνα του τμήματος της επεξεργασίας των δεδομένων και περιλαμβάνουν διάφορους ελέγχους και αλληλεπιδράσεις με τη βάση δεδομένων για τη σωστή καταγραφή της παρουσίας.

5.3.2 Επεξεργασία Δεδομένων Φοιτητή

Σε αυτή την ενότητα θα αναλύσουμε πως το λογισμικό χειρίζεται τα αποκωδικοποιημένα δεδομένα QR που προέρχονται από τους φοιτητές. Όταν το κλειδί 'type' στα δεδομένα αντιστοιχίζεται με το 'student', ενεργοποιείται μια σειρά από λειτουργίες που αποσκοπούν στην επαλήθευση και στην καταγραφή της παρουσίας του φοιτητή σε αυτό το σημείο το κίτρινο LED ενεργοποιείται για να δείξει ότι η διαδικασία επεξεργασίας των δεδομένων έχει ξεκινήσει.

Αρχικά εξάγουμε τις πληροφορίες του φοιτητή από τα δεδομένα QR, όπως το αναγνωριστικό χρήστη, το αναγνωριστικό τάξης και τα προσωπικά στοιχεία, συμπεριλαμβανομένου του ονόματος και του επωνύμου, καθώς και το email. Αυτές οι πληροφορίες είναι ζωτικής σημασίας για την αναγνώριση κάθε φοιτητή μοναδικά και τη διασφάλιση της σωστής καταγραφής των αρχείων παρουσίας. Στη συνέχεια, το πρόγραμμα επικοινωνεί με το Firestore, για να ανακτήσει το τρέχον πρόγραμμα του μαθήματος και τους επιτρεπόμενους χρόνους εισόδου στην τάξη. Αυτά τα βήματα είναι κρίσιμα για τον καθορισμό του χρονικού πλαισίου καταγραφής της παρουσίας - επιβεβαιώνοντας ότι ο φοιτητής προσπαθεί να κάνει «check in» κατά τη διάρκεια του κατάλληλου χρονικού παραθύρου. Έπειτα, το σύστημα λαμβάνει την τρέχουσα ημερομηνία και ώρα από το εσωτερικό ρολόι του Raspberry Pi για να δημιουργήσει μια χρονική σήμανση για την εκδήλωση της συμμετοχής. Αυτό συνδυάζεται με τα αναγνωριστικά του χρήστη και της τάξης για να σχηματίσουν ένα μοναδικό αναγνωριστικό, το οποίο θα χρησιμοποιηθεί για την κατάθεση του αρχείου παρουσίας στη βάση δεδομένων. Με αυτά τα δεδομένα, το πρόγραμμα κατασκευάζει δύο 'dictionaries', ένα που αντιπροσωπεύει το προφίλ του φοιτητή και ένα άλλο για το αρχείο παρουσίας. Το πρόγραμμα προχωρά στην εκτέλεση δύο ελέγχων, αρχικά ελέγχει την κατάσταση του μαθήματος στο οποίο ο φοιτητής θέλει να πραγματοποιήσει παρουσία. Εάν η τάξη βρίσκεται σε κατάσταση αρχικοποίησης το πρόγραμμα προχωρά στη διαδικασία δημιουργίας του προφίλ του φοιτητή στη βάση δεδομένων και στην εγγραφή του στο συγκεκριμένο μάθημα. Μετά από αυτήν την διαδικασία αρχικοποίησης, ελέγχεται εάν ο φοιτητής είναι ήδη εγγεγραμμένος στην τάξη και εάν η σάρωση πραγματοποιήθηκε εντός του προκαθορισμένου περιθωρίου καταβολής παρουσίας του μαθήματος. Εάν πληρούνται αυτές οι προϋποθέσεις, δημιουργείται ένα αρχείο παρουσίας του φοιτητή στο Firestore.

Εάν κάποιος από αυτούς τους ελέγχους αποτύχει (για παράδειγμα, εάν ο φοιτητής σαρώσει εκτός της προγραμματισμένης ώρας ή δεν είναι εγγεγραμμένος στο μάθημα) ενεργοποιείται μια ρουτίνα

σφάλματος. Αυτό περιλαμβάνει την ενεργοποίηση της κόκκινης λυχνίας LED για να σηματοδοτήσει ένα σφάλμα, συνοδευόμενο από έναν ήχο σφάλματος για να ειδοποιήσει τον χρήστη. Αυτή η ολοκληρωμένη επεξεργασία διασφαλίζει ότι τα δεδομένα παρακολούθησης όχι μόνο καταγράφονται με ακρίβεια αλλά και πλαισιώνονται σωστά σύμφωνα με το πρόγραμμα της τάξης, διατηρώντας την ακεραιότητα του συστήματος παρακολούθησης.

5.3.3 Δημιουργία και Διαχείριση Προγράμματος Τάξης

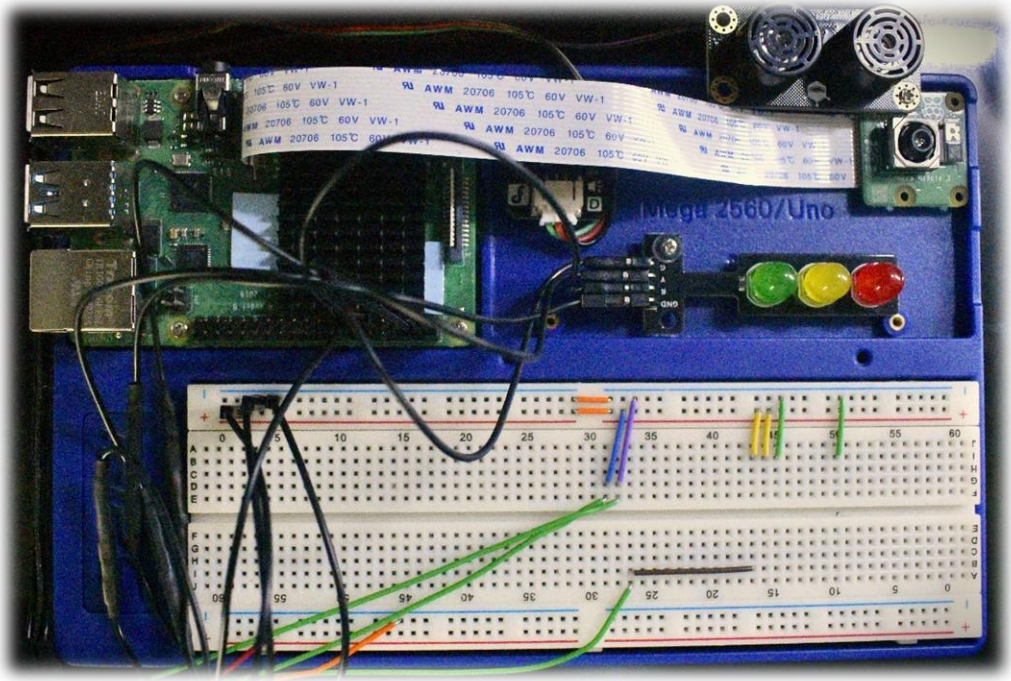
Επιπλέον, το λογισμικό έχει σχεδιαστεί για να χειρίζεται τυχόν ενημερώσεις που μπορεί να κάνουν οι εκπαιδευτικοί στο πρόγραμμα της τάξης, είτε προσθέτοντας νέες συνεδρίες είτε προσαρμόζοντας την κατάσταση του. Αυτές οι αλλαγές γίνονται άμεσα στο σύστημα και συγχρονίζονται όλες οι σχετικές εγγραφές στη βάση δεδομένων. Η διαδικασία περιλαμβάνει τον χειρισμό των δεδομένων σε πραγματικό χρόνο, όπου οι ενημερώσεις του προγράμματος του μαθήματος προωθούνται στο Firestore και στη συνέχεια ανακτώνται από το σύστημα για να διασφαλιστεί ότι όλες οι ενδιαφερόμενες λειτουργίες έχουν πρόσβαση στις πιο πρόσφατες πληροφορίες.

Οι ενδείξεις των LED και οι ηχητικές ειδοποιήσεις παίζουν ρόλο και εδώ, σηματοδοτώντας την επιτυχή ενσωμάτωση των δεδομένων με το πρόγραμμα της τάξης ή προειδοποιώντας για πιθανά ζητήματα που χρειάζονται επίλυση. Για παράδειγμα, εάν δεν υπάρχει αναντιστοιχία μεταξύ των σαρωμένων δεδομένων και του χρονοδιαγράμματος ή εάν μια ενημέρωση αποτύχει να επεξεργαστεί σωστά, το σύστημα θα ειδοποιήσει τους χρήστες μέσω οπτικών και ακουστικών ενδείξεων.

Αυτή η ενότητα δείχνει πώς το σύστημα όχι μόνο παρακολουθεί την ατομική παρακολούθηση, αλλά διαχειρίζεται και ενημερώνει δυναμικά τα προγράμματα των τάξεων, ένα ζωτικό χαρακτηριστικό που ενισχύει τη λειτουργικότητα των διοικητικών διαδικασιών του εκπαιδευτικού ιδρύματος.

5.4 Διαδικασίες Ενσωμάτωσης του Τελικού Συστήματος στην Τάξη

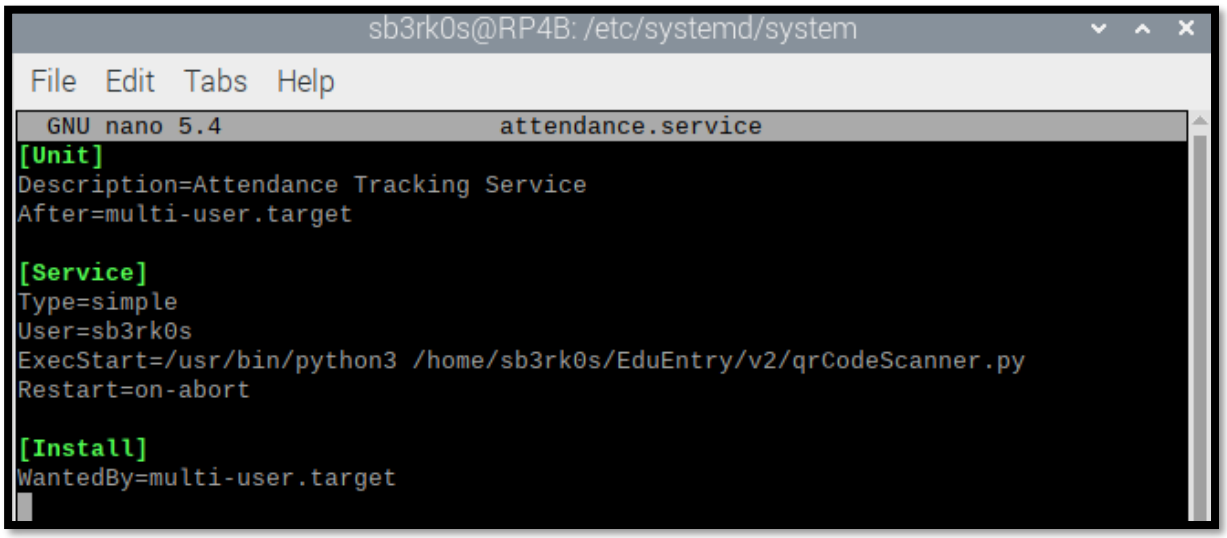
Στην τελική φάση της ανάπτυξης του συστήματος ελέγχου παρουσίας Raspberry Pi στο ακαδημαϊκό περιβάλλον, θα αναλύσουμε κάποια τελευταία βήματα που διασφαλίζουν την ομαλή λειτουργία και την χρηστικότητα του. Το development kit του συστήματος παρουσιάζεται στο παρακάτω σχήμα, τονίζοντας τον ευέλικτο, επεκτάσιμο και αποτελεσματικό σχεδιασμό της υλοποίησης του συστήματος προσαρμοσμένο για τη χρήση στην τάξη.



Σχήμα 5.7: Τελική Υλοποίηση Συστήματος

Η ενσωμάτωση του λογισμικού σάρωσης QR κώδικα και παρακολούθησης παρουσίας στη ρουτίνα εκκίνησης του Raspberry Pi είναι ζωτικής σημασίας για την αυτονομία και την πρακτικότητα. Αυτή η ενσωμάτωση επιτρέπει στο σύστημα να λειτουργεί ανεξάρτητα κάθε φορά που ενεργοποιείται η συσκευή, χωρίς χειροκίνητη παρέμβαση από εκπαιδευτικούς ή διοικητικό προσωπικό.

Αρχικά δημιουργούμε ένα αρχείο υπηρεσίας συστήματος (**systemd service file**), το οποίο επιτρέπει τη διαχείριση του λογισμικού μας σαν μια τυπική υπηρεσία του λειτουργικού συστήματος. Αυτό το αρχείο θα αποθηκευτεί στο "**/etc/systemd/system/**", με όνομα "**attendance.service**" :



Σχήμα 5.8: Creating Attendance System Service

Μόλις δημιουργηθεί το αρχείο υπηρεσίας, πρέπει να γίνει επανεκκίνηση της διαμόρφωσης του διαχειριστή συστήματος, να ενεργοποιηθεί η εκκίνηση της υπηρεσίας κατά την εκκίνηση της συσκευής και, στη συνέχεια να ξεκινήσει η υπηρεσία χρησιμοποιώντας τις ακόλουθες εντολές:

sudo systemctl daemon-reload

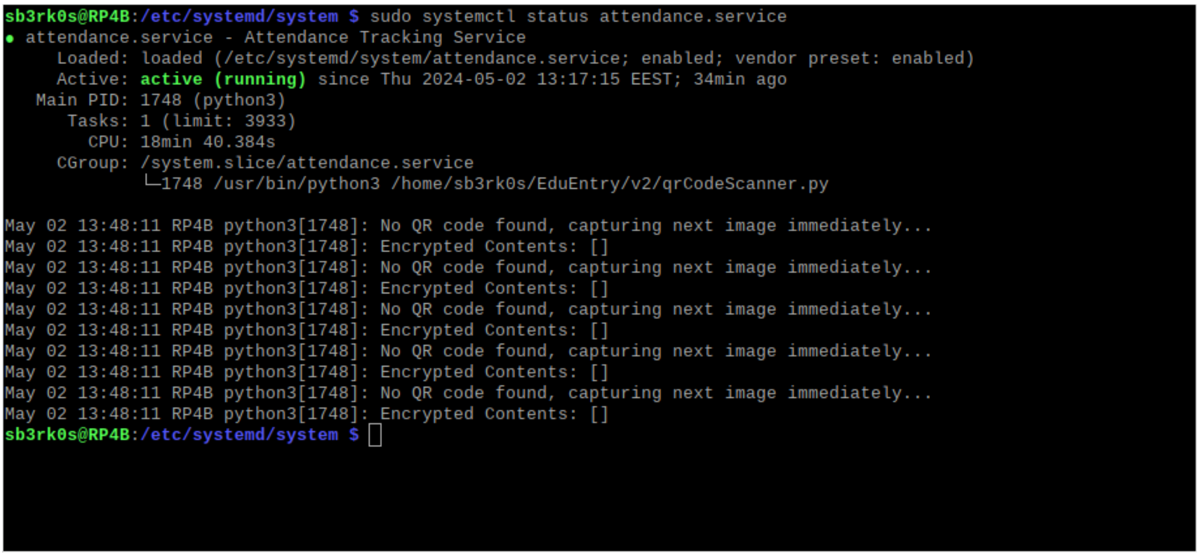
sudo systemctl enable attendance.service

sudo systemctl start attendance.service

Για να ελέγξουμε ότι η υπηρεσία εκτελείται σωστά, χρησιμοποιήσουμε την εντολή:

sudo systemctl status attendance.service

Αυτή η εντολή παρέχει πληροφορίες σχετικά με την κατάσταση της υπηρεσίας και θα επιβεβαιώσει εάν είναι ενεργή και εκτελείται.



Σχήμα 5.9: Attendance System Service Status

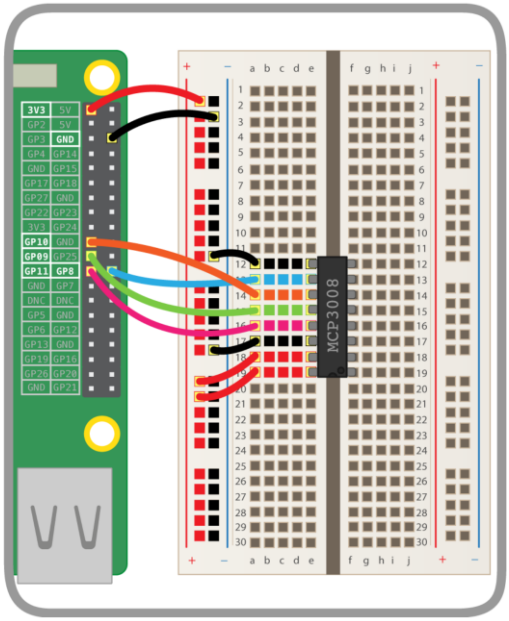
Δεδομένου ότι η συσκευή θα λειτουργεί headless σε μια τάξη, η καταγραφή των σφαλμάτων είναι σημαντική. Το λογισμικό μας έχει προσαρμοστεί για να καταγράφει τα σημαντικά συμβάντα και τα σφάλματα σε ένα αρχείο, το οποίο μπορεί να βοηθήσει στην αντιμετώπιση των προβλημάτων που ενδέχεται να προκύψουν και στην συντήρηση του συστήματος χωρίς να χρειάζεται άμεση πρόσβαση στη συσκευή.

Ακολουθώντας αυτά τα βήματα, το σύστημα παρακολούθησης Raspberry Pi θα λειτουργεί αξιόπιστα κάθε φορά που εκκινεί η συσκευή, καθιστώντας το μια πρακτική και αποτελεσματική λύση για την παρακολούθηση της παρουσίας των μαθητών στις τάξεις. Αυτή η προσέγγιση όχι μόνο μεγιστοποιεί τη χρησιμότητα του συστήματος, αλλά ενισχύει επίσης την ενσωμάτωσή του στις καθημερινές ακαδημαϊκές λειτουργίες, εξασφαλίζοντας ελάχιστη διακοπή και μέγιστη αποτελεσματικότητα.

5.5 Μελλοντική Εξέλιξη του Συστήματος

Για την ενίσχυση της απόδοσης του συστήματος, ιδιαίτερα στον τομέα της κατανάλωσης ενέργειας, μπορεί να ενσωματωθεί ο αναλογικός αισθητήρας υπερήχων - URM09 από την DFROBOT,

ο οποίος διακρίνεται στο σχήμα 5.7 της συσκευής μας. Ο συγκεκριμένος αισθητήρας καθώς είναι αναλογικός δεν μπορεί να συνδεθεί απευθείας με τα GPIO Pins του Raspberry Pi τα οποία χρησιμοποιούνται μόνο για ψηφιακά σήματα. Ο μεσάζοντας που έχει σκοπό να κάνει αυτή την διασύνδεση είναι ένας **Analog to Digital Converter** (ADC). Για την μελλοντική εξέλιξη του συστήματος μας επιλέχθηκε ο MCP3008 για τον οποίο έχει γίνει και η απαραίτητη προεργασία στο Breadboard.



Σχήμα 5.10: Analogue Inputs for Raspberry Pi [8]

Ο αισθητήρας υπερήχων αξιοποιεί τα ηχητικά κύματα για τη μέτρηση της απόστασης και μπορεί να χρησιμοποιηθεί μέσα στο σύστημα για την ανίχνευση της παρουσίας των χρηστών. Με αυτόν τον τρόπο, το σύστημα ενεργοποιείται και ξεκινά τη διαδικασία καταγραφής παρουσίας μόνο όταν ένας χρήστης είναι κοντά, αντί να φωτογραφίζει συνεχώς. Αυτή η στοχευμένη ενεργοποίηση έχει σκοπό την εξοικονόμηση της ενέργειας αλλά και την μείωση της περιττής φθοράς των εξαρτημάτων του συστήματος, συμβάλλοντας σε μια πιο βιώσιμη και οικονομικά αποδοτική λειτουργία. Αυτή η προσέγγιση αποτελεί παράδειγμα της έξυπνης χρήσης της τεχνολογίας αισθητήρων για την επίτευξη ενεργειακής απόδοσης.

Σχήμα 5.11: Gravity: URM09 Ultrasonic Distance Sensor (2～500cm, Analog) [5]

Στη μελλοντική ανάπτυξη του συστήματος, οι πρακτικές βελτιώσεις μπορούν να επικεντρωθούν στη βελτίωση της αλληλεπίδρασης των χρηστών και της αυτοματοποίησης του συστήματος. Μια απλή λύση θα μπορούσε να είναι η ενσωμάτωση της τεχνολογίας αναγνώρισης προσώπου για την περαιτέρω αυτοματοποίηση της διαδικασίας παρακολούθησης. Αυτό θα επέτρεπε στο σύστημα να επιβεβαιώνει ταυτότητες χωρίς να απαιτούνται σαρώσεις κωδικών QR, επιταχύνοντας έτσι τη διαδικασία εισόδου και μειώνοντας τις ουρές. Μια άλλη πρακτική αναβάθμιση θα μπορούσε να περιλαμβάνει τη βελτίωση της διεπαφής χρήστη στην ίδια τη συσκευή. Μια μικρή οθόνη θα μπορούσε να προστεθεί για να παρέχει ανατροφοδότηση σε πραγματικό χρόνο στους χρήστες σχετικά με την κατάσταση παρουσίας τους ή τυχόν σφάλματα στη διαδικασία. Αυτή η άμεση ανατροφοδότηση θα μπορούσε να βελτιώσει την εμπειρία του χρήστη και να μειώσει την ανάγκη για χειροκίνητους ελέγχους στο αρχείο σφαλμάτων της υπηρεσίας. Αυτές οι βελτιώσεις στοχεύουν να καταστήσουν το σύστημα πιο φιλικό προς το χρήστη και ευθυγραμμισμένο με τις ανάγκες ενός δυναμικού εκπαιδευτικού περιβάλλοντος.

5.6 Σύνοψη

Το κεφάλαιο 5 παρέχει μια λεπτομερή εξερεύνηση των δυνατοτήτων του Raspberry Pi ως ένα θεμελιώδες εργαλείο για την ανάπτυξη ενός αποτελεσματικού συστήματος ελέγχου παρουσίας σε εκπαιδευτικά ιδρύματα. Αυτό το κεφάλαιο προσφέρει μια ολοκληρωμένη επισκόπηση της ενσωμάτωσης του Raspberry Pi στη διαχείριση της παρουσίας των φοιτητών μέσω της σάρωσης ενός κωδικού QR. Η ευελιξία του Raspberry Pi και η ικανότητά του να υποστηρίζει διάφορα περιβάλλοντα προγραμματισμού και περιφερειακές συσκευές το καθιστούν ιδανική επιλογή για αυτήν την εφαρμογή. Ξεκινήσαμε συζητώντας την επιλογή του μοντέλου Raspberry Pi που ανταποκρίνεται καλύτερα στις απαιτήσεις του έργου μας, συμπεριλαμβανομένης της επεξεργαστικής ισχύος, της χωρητικότητας μνήμης και των επιλογών συνδεσιμότητας - κρίσιμης σημασίας για το χειρισμό εργασιών επεξεργασίας δεδομένων και χειρισμού εικόνας σε πραγματικό χρόνο. Αφού επιλέξουμε το υλικό, εξετάσαμε τα λεπτομερή βήματα ρύθμισης που απαιτούνται για την προετοιμασία του Raspberry Pi. Αυτά περιλαμβάνουν τη διαμόρφωση του λειτουργικού συστήματος, των ρυθμίσεων δικτύου και την εξασφάλιση πρόσβασης στη συσκευή, θέτοντας μια σταθερή βάση για τα στοιχεία λογισμικού που θα χρησιμοποιήσουμε. Οι επόμενες υποενότητες αφορούν τη ρύθμιση του λογισμικού, ξεκινώντας με την εγκατάσταση βιβλιοθηκών και τη διαμόρφωση του κώδικα σάρωσης των QR. Αυτό το λογισμικό είναι αναπόσπαστο μέρος του συστήματος, επιτρέποντας στο Raspberry Pi να διαβάζει και να ερμηνεύει τους κωδικούς QR που παρουσιάζονται από τους φοιτητές κατά την είσοδό τους στην τάξη. Επιπλέον, διερευνούμε πώς το Raspberry Pi επεξεργάζεται και καταγράφει τα δεδομένα παρουσίας στη βάση δεδομένων Firebase. Τέλος, το κεφάλαιο καλύπτει τις προκλήσεις που αντιμετωπίστηκαν κατά την ανάπτυξη του συστήματος Raspberry Pi. Αυτές οι προκλήσεις κυμαίνονται από περιορισμούς υλικού έως σφάλματα λογισμικού και ζητήματα διεπαφής χρήστη. Παρουσιάζονται λύσεις που αναπτύχθηκαν για την αντιμετώπιση αυτών των προκλήσεων, προσφέροντας πολύτιμες πληροφορίες σχετικά με τις πρακτικές πτυχές της ανάπτυξης ενός συστήματος που βασίζεται στο Raspberry Pi σε εκπαιδευτικό περιβάλλον. Συνοπτικά, αυτό το κεφάλαιο όχι μόνο περιγράφει λεπτομερώς την τεχνική εφαρμογή του Raspberry Pi για την παρακολούθηση παρουσίας, αλλά υπογραμμίζει επίσης τις πρακτικές προκλήσεις και τα μαθησιακά αποτελέσματα που σχετίζονται με την ανάπτυξη μιας λύσης βασισμένης στην τεχνολογία σε ένα εκπαιδευτικό περιβάλλον. Μέσω αυτής της συζήτησης, στοχεύουμε να δείξουμε τη χρησιμότητα και τα οφέλη της χρήσης της τεχνολογίας Raspberry Pi για την ενίσχυση των διοικητικών διαδικασιών εντός των ακαδημαϊκών ιδρυμάτων.

Κεφάλαιο 6ο: Ανάπτυξη Εφαρμογής Android

6.1 Εισαγωγή

Η εφαρμογή Edu Entry έχει σχεδιαστεί για να ομαλοποιήσει τη διαδικασία κατάθεσης παρουσίας των φοιτητών στα εκπαιδευτικά ιδρύματα. Επιπλέον δίνει την δυνατότητα στους εκπαιδευτικούς να προσθέσουν συνεδρίες στο πρόγραμμα των μαθημάτων. Χρησιμοποιώντας κωδικούς QR που σαρώνονται στους σταθμούς Raspberry Pi στις αίθουσες διδασκαλίας, η εφαρμογή προσπαθεί να αποτελέσει μια αποτελεσματική και αυτοματοποιημένη μέθοδο για την καταγραφή της παρουσίας κάθε φοιτητή τη στιγμή που εισέρχεται στην τάξη. Ο κύριος σκοπός της δημιουργίας αυτής της εφαρμογής είναι η απλοποίηση της διαδικασίας καταγραφής των παρουσιών αλλά και η ελαχιστοποίηση των σφαλμάτων που σχετίζονται με τη χειροκίνητη εισαγωγή, διασφαλίζοντας έτσι την ακρίβεια και την αξιοπιστία στη διαχείριση των δεδομένων παρουσίας.

Μια εφαρμογή Android μπορεί να είναι καθοριστικής σημασίας για την ενίσχυση των εκπαιδευτικών διαδικασιών λόγω της προσαρμοστικότητάς της και των δυνατοτήτων ενσωμάτωσής της με μια ποικιλία υλικού και διαδικτυακών υπηρεσιών. Χρησιμοποιώντας μια εφαρμογή όπως το Edu Entry, οι φοιτητές μπορούν να επισημάνουν γρήγορα τη συμμετοχή τους κατά την είσοδό τους σε μια τάξη με μια απλή σάρωση ενός κωδικού QR. Αυτή η μέθοδος όχι μόνο επιταχύνει τη διαδικασία, αλλά εισάγει επίσης ένα επίπεδο αλληλεπίδρασης που είναι ελκυστικό για φοιτητές με γνώσεις τεχνολογίας. Επιπλέον, μια εφαρμογή Android μπορεί να λειτουργεί σε διαφορετικές συνθήκες δικτύου, διασφαλίζοντας την λειτουργικότητά της σε διαφορετικές περιοχές της πανεπιστημιούπολης χωρίς να χρειάζεται συνεχή συνδεσιμότητα. Ο σχεδιασμός της εφαρμογής επικεντρώνεται στην ελαχιστοποίηση του χρόνου και της προσπάθειας που απαιτείται για την χρήση της, επιτρέποντας έτσι στους φοιτητές και το διδακτικό προσωπικό να επικεντρωθούν περισσότερο σε εκπαιδευτικές δραστηριότητες. Μέσω του λιτού σχεδιασμού και της εύκολης χρήσης του, το Edu Entry αποτελεί ένα παράδειγμα για τον τρόπο με τον οποίο οι εφαρμογές Android μπορούν να συμβάλουν σημαντικά στην απλοποίηση της εκπαιδευτικής διοίκησης.

6.2 Σχεδίαση και Λειτουργικότητα Εφαρμογής

Η εφαρμογή Edu Entry έχει σχεδιαστεί με ένα περιβάλλον χρήστη που δίνει έμφαση στη χρηστικότητα και την προσβασιμότητα. Η διάταξη της διεπαφής είναι δομημένη για να παρέχει μια κατανοητή εμπειρία πλοήγησης, καθοδηγώντας τους χρήστες από τη σύνδεσή τους στον τελικό σκοπό της εφαρμογής χωρίς περιττή πολυπλοκότητα. Η χρήση οικείων στοιχείων σχεδίασης Android, διασφαλίζει ότι οι χρήστες όλων των τεχνολογικών ικανοτήτων μπορούν να χρησιμοποιούν αποτελεσματικά την εφαρμογή.

Η ασφάλεια και ο έλεγχος της ταυτότητας του χρήστη είναι ζωτικής σημασίας, ειδικά σε εκπαιδευτικά περιβάλλοντα όπου εμπλέκονται προσωπικά και ακαδημαϊκά δεδομένα. Το Edu Entry χρησιμοποιεί το OAuth 2.0 για τη διαδικασία ελέγχου ταυτότητας, το οποίο είναι ένα ισχυρό πρωτόκολλο που επιτρέπει την ασφαλή εξουσιοδότηση από εφαρμογές ιστού ή στην περίπτωσή μας τα συστήματα του πανεπιστήμιου. Η εφαρμογή ενσωματώνει αυτό το πρωτόκολλο για να παρέχει μια ασφαλή διαδικασία σύνδεσης όπου οι χρήστες πραγματοποιούν έλεγχο ταυτότητας μέσω των θεσμικών διαπιστευτηρίων του ιδρύματος. Η εφαρμογή ενισχύει την αλληλεπίδραση των χρηστών χρησιμοποιώντας στοιχεία σχεδίασης που ανταποκρίνονται σε αλλαγές. Αφού συνδεθούν, οι χρήστες μπορούν να επιλέξουν την

τάξη που τους ενδιαφέρει να πραγματοποιήσουν παρουσία από μια δυναμικά συμπληρωμένη λίστα που λαμβάνεται από το Firebase, επιδεικνύοντας την αλληλεπίδραση των δεδομένων σε πραγματικό χρόνο. Η διαδικασία δημιουργίας ενός κωδικού QR για την παρακολούθηση μιας συνεδρίας στην τάξη είναι άμεση μόλις επιλεγεί μια τάξη και χρησιμοποιεί τόσο τις δυνατότητες της συσκευής όσο και του διακομιστή για να εξασφαλίσει γρήγορη και αποτελεσματική επεξεργασία δεδομένων. Αυτός ο κωδικός QR μπορεί στη συνέχεια να σαρωθεί σε έναν σταθμό Raspberry Pi, ο οποίος έχει ρυθμιστεί σε κάθε τάξη, για να ελέγχει και να καταγράφει τη συμμετοχή. Η εφαρμογή παρέχει επίσης σχόλια και προτροπές για να καθοδηγήσει τους χρήστες σε κάθε βήμα, καθιστώντας τη διαδικασία κατανοητή για νέους χρήστες.

Συνοπτικά, η εφαρμογή Edu Entry συνδυάζει τον σχεδιασμό με την λειτουργικότητα για να δημιουργήσει ένα αποτελεσματικό εργαλείο για τη διαχείριση της φοίτησης των φοιτητών σε εκπαιδευτικά ιδρύματα. Εστιάζοντας στην ευκολία χρήσης, την ασφάλεια και τη διαδραστικότητα, η εφαρμογή διασφαλίζει ότι όλοι οι χρήστες, ανεξάρτητα από τις τεχνικές δεξιότητές τους, μπορούν να διαχειριστούν την ακαδημαϊκή τους παρουσία αποτελεσματικά και με ασφάλεια.

6.2.1 Σχεδίαση Διεπαφής Χρήστη

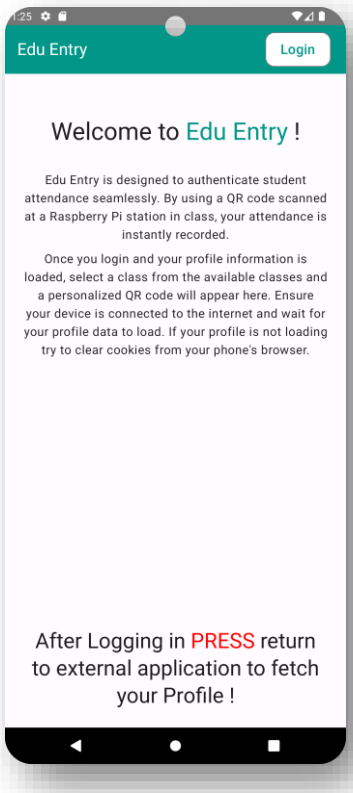
Το Edu Entry χρησιμοποιεί το Jetpack Compose, μια σύγχρονη εργαλειοθήκη της Google για τη δημιουργία native διεπαφών χρήστη σε εφαρμογές Android. Το Jetpack Compose είναι μια δηλωτική δομή διεπαφής χρήστη που απλοποιεί τη διαδικασία δημιουργίας ισχυρών και διαδραστικών UI στοιχείων, μειώνοντας σημαντικά τον κώδικα που σχετίζεται από παλαιότερες διατάξεις που βασίζονται στην προβολή στοιχείων [14]. Βελτιώνει την απόδοση του συστήματος διασφαλίζοντας ότι επανασχεδιάζονται μόνο τα στοιχεία που πρέπει να ενημερωθούν, αντί να ανανεώνεται ολόκληρη η ιεραρχία των στοιχείων που προβάλλονται, όπως συνηθιζόταν στην παραδοσιακή ανάπτυξη Android εφαρμογών. Το Material 3, η τελευταία υλοποίηση σχεδιασμού υλικού της Google, ενσωματώνεται στην εφαρμογή μέσω του Jetpack Compose, φέρνοντας προηγμένες δυνατότητες σχεδιασμού, όπως δυναμικό χρωματικό θέμα, βελτιωμένη αρχιτεκτονική στοιχείων και βελτιωμένες λειτουργίες προσβασιμότητας [9]. Οι αρχές σχεδιασμού του Material 3 καθοδηγούν τον αισθητικό και λειτουργικό σχεδιασμό της εφαρμογής, διασφαλίζοντας μια συνεκτική εμπειρία χρήστη που είναι οπτικά ελκυστική και εύκολη στην πλοήγηση. Το Jetpack Compose και το Material 3 μαζί παρέχουν μια βελτιωμένη προσέγγιση για τη δημιουργία εφαρμογών Android. Αξιοποιώντας το Compose, η εφαρμογή επωφελείται από μια λιγότερο περίπλοκη διαχείριση καταστάσεων και μια εκ φύσεως modular αρχιτεκτονική. Αυτή η διαμόρφωση επιτρέπει επαναχρησιμοποιήσιμα στοιχεία περιβάλλοντος χρήστη, διευκολύνοντας τη συντήρηση και την κλιμάκωση της εφαρμογής. Η έμφαση του Material 3 στα responsive animations και transitions διασφαλίζει ότι η οπτική ανατροφοδότηση είναι άμεση και ομαλή, ενισχύοντας την αλληλεπίδραση του χρήστη με την εφαρμογή.

Η επιλογή του χρώματος εντός της εφαρμογής ευθυγραμμίζεται με την οπτική ταυτότητα των άλλων διαδικτυακών εφαρμογών του πανεπιστημίου, δημιουργώντας μια συνεπή εμπειρία επωνυμίας σε όλες τις πλατφόρμες. Το κύριο χρώμα που χρησιμοποιείται, μια χαρακτηριστική απόχρωση του teal, όχι μόνο ενισχύει την ταυτότητα του πανεπιστημίου, αλλά βοηθά επίσης στην οπτική συνέχεια η οποία είναι σημαντική για τη χρηστικότητα και την αλληλεπίδραση των χρηστών. Αυτή η συνέπεια στο σχεδιασμό και το συνδυασμό χρωμάτων σε όλη την ψηφιακή παρουσία του πανεπιστημίου προσπαθεί να ενισχύσει την εμπιστοσύνη των χρηστών στην πλατφόρμα.

6.2.2 Σημείο Εισόδου στην Εφαρμογή

Η αρχική οθόνη της εφαρμογής παίζει κρίσιμο ρόλο στην εμπειρία χρήστη, λειτουργώντας ως το σημείο εισόδου στις λειτουργίες της εφαρμογής. Αυτή η οθόνη έχει σχεδιαστεί για να είναι ενημερωτική και φιλόξενη, παρέχοντας στους χρήστες μια σαφή κατανόηση του σκοπού της εφαρμογής και του τρόπου πλοήγησης σε αυτήν.

Ο σχεδιασμός της αρχικής οθόνης είναι απλός και φιλικός προς το χρήστη. Διαθέτει μια καθαρή διάταξη με ένα εμφανές κουμπί "Login" στο επάνω μέρος, καθοδηγώντας τους νέους χρήστες να συνδεθούν για να έχουν πρόσβαση στις λειτουργείες της εφαρμογής. Το φόντο είναι μινιμαλιστικό, εστιάζοντας την προσοχή των χρηστών στο κείμενο και την λειτουργία σύνδεσης, γεγονός που ενισχύει τη χρηστικότητα. Το κείμενο που εμφανίζεται στην αρχική οθόνη εξηγεί συνοπτικά τον σκοπό της εφαρμογής δηλαδή να πιστοποιήσει τη συμμετοχή των φοιτητών στην τάξη μέσω κωδικών QR που σαρώνονται στους σταθμούς Raspberry Pi. Αυτή η άμεση γνωστοποίηση βοηθά στον καθορισμό των προσδοκιών των χρηστών από την αρχή. Κάτω από το εισαγωγικό κείμενο, παρέχονται λεπτομερείς οδηγίες σχετικά με τον τρόπο λειτουργίας της εφαρμογής μόλις συνδεθεί ο χρήστης. Αυτό περιλαμβάνει τη φόρτωση των πληροφοριών προφίλ, την επιλογή μιας τάξης και τη δημιουργία ενός εξατομικευμένου κωδικού QR. Επίσης παρέχονται κοινά βήματα για την αντιμετώπιση προβλημάτων, όπως η διασφάλιση ότι η συσκευή είναι συνδεδεμένη στο διαδίκτυο και η εκκαθάριση των cookies από τον browser της συσκευής, γεγονός που επιλύει προληπτικά πιθανά προβλήματα, βελτιώνοντας τη συνολική εμπειρία.



Σχήμα 6.1: Android Application Entry Point

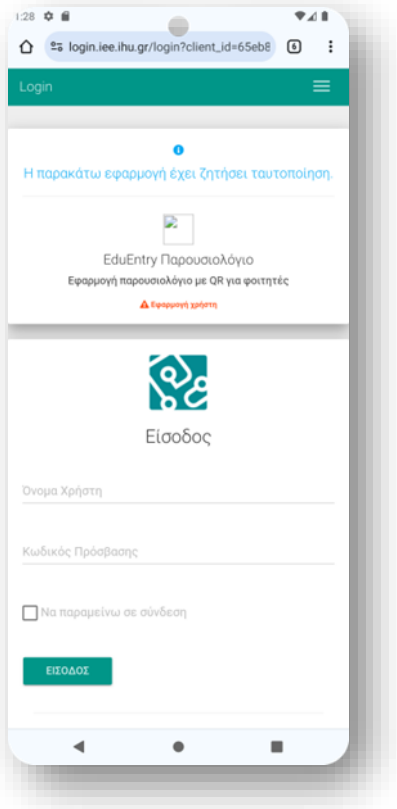
Η σαφήνεια και η απλότητα του σχεδιασμού του περιεχομένου διασφαλίζει ότι οι χρήστες αισθάνονται σιγουριά καθώς αρχίζουν να περιηγούνται στην εφαρμογή. Ο φιλόξενος και ενημερωτικός χαρακτήρας

αυτής της οθόνης βοηθά στη μείωση της σύγχυσης και της πιθανής απογοήτευσης των χρηστών, βοηθώντας στην ομαλότερη λειτουργία.

6.3 Σύνδεση Χρήστη και Πιστοποίηση Ταυτότητας

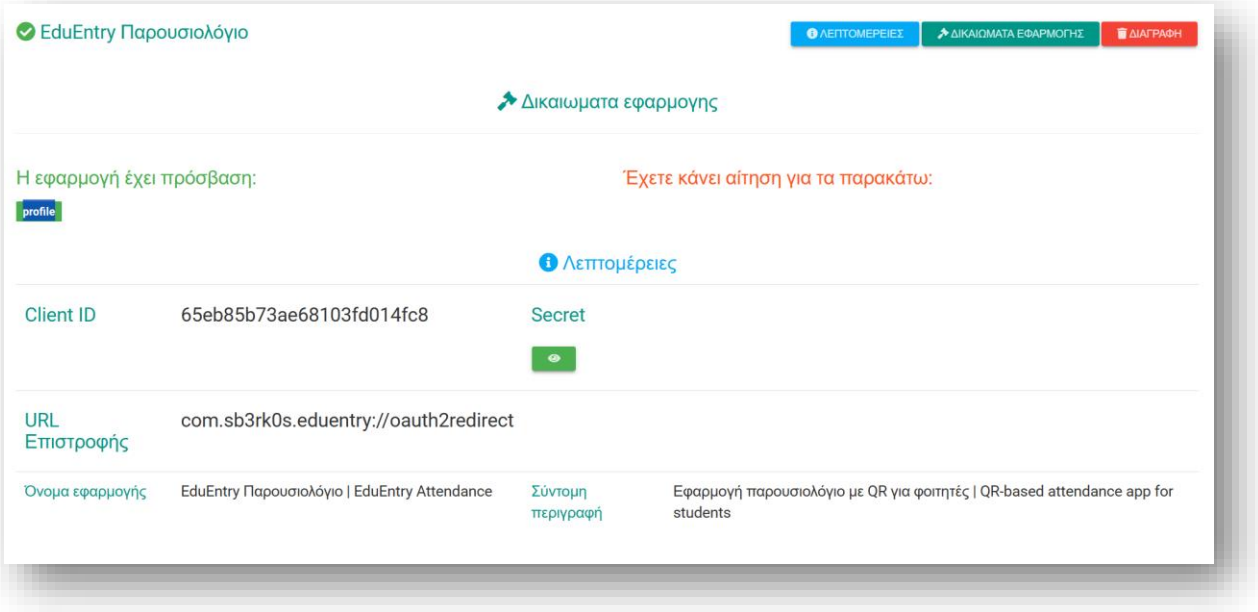
Η διαδικασία σύνδεσης χρήστη και ελέγχου ταυτότητας αποτελεί θεμελιώδη πτυχή της εφαρμογής Edu Entry, διασφαλίζοντας ότι μόνο εξουσιοδοτημένοι χρήστες του ιδρύματος μπορούν να έχουν πρόσβαση και να χρησιμοποιούν την εφαρμογή. Αυτή η ενότητα περιγράφει λεπτομερώς την προσέγγιση που ακολουθείται για τον ασφαλή έλεγχο ταυτότητας χρήστη.

Η εφαρμογή Edu Entry χρησιμοποιεί το σύστημα ταυτοποίησης που παρέχεται από το πανεπιστήμιο. Αμέσως μετά το πάτημα του κουμπιού “Login” γίνεται μια ανακατεύθυνση στον default browser της συσκευής στην υπηρεσία “login.iee.ihu.gr/login” και ακολουθεί μια ροή σύνδεσης βασισμένη στο πρότυπο OAuth 2.0 protocol.



Σχήμα 6.2: Redirect to login.iee.ihu.gr and Connect with University’s credentials

Αρχικά, πρέπει να δημιουργηθεί μια εφαρμογή στο οικοσύστημα του πανεπιστημίου. Αυτό διευκολύνεται μέσω μιας εσωτερικής πύλης όπου οι ενδιαφερόμενοι μπορούν να καταχωρούν νέες εφαρμογές. Η πύλη επιτρέπει να καθορίσουν το πεδίο υλοποίησης της εφαρμογής, όπως σε ποια δεδομένα μπορεί να έχει πρόσβαση και ποιες λειτουργίες θα έχει.



Σχήμα 6.3: Creating an App (My Apps at login.iee.ihu.gr/apps)

6.3.1 OAuth 2.0

Στην ανάπτυξη της εφαρμογής Edu Entry στο Android, το OAuth 2.0 αποτελεί αναπόσπαστο μέρος της διασφάλισης του ελέγχου ταυτότητας χρήστη και της διατήρησης της ακεραιότητας των δεδομένων. Αυτό το πρωτόκολλο διασφαλίζει ότι μόνο πιστοποιημένοι χρήστες από το ίδρυμα μπορούν να έχουν πρόσβαση και να αλληλεπιδρούν με τις λειτουργίες της εφαρμογής. Με την ενσωμάτωση στα υπάρχοντα πλαίσια ελέγχου ταυτότητας του πανεπιστημίου, το Edu Entry παρέχει μια ασφαλή και οικεία εμπειρία σύνδεσης για όλους τους χρήστες. Αυτή η μέθοδος απλοποιεί τη διαδικασία εισόδου του χρήστη και τηρεί αυστηρά πρότυπα ασφαλείας για την προστασία ευαίσθητων εκπαιδευτικών δεδομένων.

Η διαδικασία OAuth ξεκινά με τον χρήστη να ανακατευθύνεται σε μια διεύθυνση URL εξουσιοδότησης, όπου πρέπει να συνδεθεί και είτε να επιτρέψει είτε να απορρίψει το αίτημα πρόσβασης της εφαρμογής. Εάν ο χρήστης συναινέσει, ανακατευθύνεται πίσω στην εφαρμογή στο σημείο "**com.sb3rk0s.eduentry://oauth2redirect**" με έναν κωδικό εξουσιοδότησης (**authorization code**). Στη συνέχεια, αυτός ο κωδικός ανταλλάσσεται για ένα διακριτικό πρόσβασης (**access token**) στο endpoint "**login.iee.ihu.gr/token**". Το access token εξουσιοδοτεί την εφαρμογή να έχει πρόσβαση σε συγκεκριμένους πόρους που αφορούν τον χρήστη, όπως δεδομένα προφίλ με περιορισμένη διάρκεια ζωής για τη επίτευξη της ασφάλειας κατά τη διάρκεια της ενεργού περιόδου του.

6.3.2 REST API

Παράλληλα με το OAuth 2.0, το Edu Entry χρησιμοποιεί το Retrofit για την εκτέλεση αιτημάτων HTTP και τη διευκόλυνση των κλήσεων API (**REST API Calls**), διευκολύνοντας την επικοινωνία μεταξύ της εφαρμογής και του διακομιστή του πανεπιστημίου. Μόλις ληφθεί το authorization code από τη διαδικασία OAuth 2.0, χρησιμοποιείται για την πραγματοποίηση API Calls στον διακομιστή RESTful αρχιτεκτονικής του ιδρύματος "**api.it.teithe.gr**". Οι διασυνδέσεις

(**Interfaces**) "**AuthService**" και "**UserProfileService**" εντός της εφαρμογής διαχειρίζονται αυτές τις αλληλεπιδράσεις. Για παράδειγμα, το AuthService στέλνει ένα **POST request** για την ανταλλαγή του authorization code με ένα access token και το **UserProfileService** ανακτά το προφίλ του χρήστη πραγματοποιώντας ένα API Call χρησιμοποιώντας το access token.

Την όλη διαδικασία διαχειρίζεται το **AuthViewModel**, όπου ξεκινούν κλήσεις δικτύου για τον έλεγχο ταυτότητας του χρήστη και τη λήψη των δεδομένων του προφίλ του. Οι επιτυχείς απαντήσεις από αυτές τις κλήσεις έχουν ως αποτέλεσμα την αποθήκευση του προφίλ χρήστη στο **SharedPreferences**, διασφαλίζοντας ότι η κατάσταση σύνδεσης και τα δεδομένα του χρήστη παραμένουν διαθέσιμα σε όλες τις περιόδους λειτουργίας της εφαρμογής. Τα σφάλματα κατά τη διάρκεια αυτών των διαδικασιών καταγράφονται και η κατάσταση του περιβάλλοντος χρήστη ενημερώνεται ανάλογα, ώστε να δείχνει εάν ο χρήστης είναι συνδεδεμένος ή εάν έχει παρουσιαστεί κάποιο σφάλμα.

6.3.3 Διαχείριση Προφίλ Χρήστη

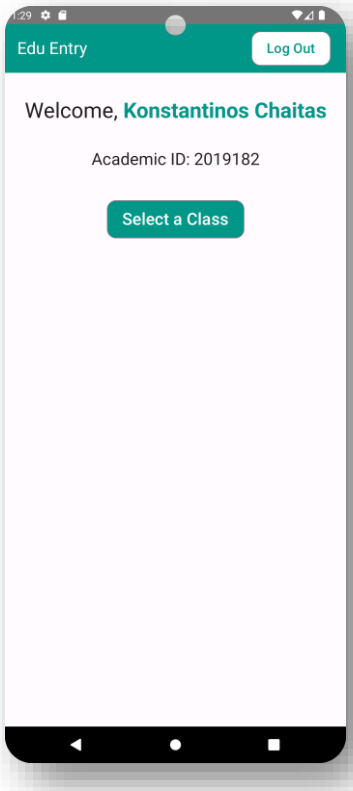
Για την αποθήκευση των δεδομένων χρήστη όπως το όνομα και τα ακαδημαϊκά αναγνωριστικά χρησιμοποιείται το SharedPreferences αρχείο του Android, λόγω της απλότητας και της αποτελεσματικότητάς του στον χειρισμό μιας μικρής ποσότητας δεδομένων. Αυτή η μέθοδος επιτρέπει την γρήγορη προσπέλαση και την διατήρηση των δεδομένων σε όλες τις περιόδους λειτουργίας καθώς εξασφαλίζει τη διατήρηση τους χωρίς να χρειάζεται να τα ανακτήσετε επανειλημμένα από έναν απομακρυσμένο διακομιστή, επιταχύνοντας έτσι την ανταπόκριση της εφαρμογής και παρέχοντας επιπλέον μια ασφαλή λύση αποθήκευσης που είναι προσβάσιμη μόνο από την εφαρμογή.

Παρακάτω βρίσκεται μια εικόνα που απεικονίζει την μορφή JSON αλλά και ποια δεδομένα ανακτώνται από το API Call.


```
{
  "am": "123456",
  "regyear": "2012",
  "regsem": "2",
  "sem": "2",
  "givenName;lang-el": "ΓΕΩΡΓΙΟΣ",
  "sn;lang-el": "ΠΑΠΑΔΟΠΟΥΛΟΣ",
  "fathersname;lang-el": "ΙΩΑΝΝΗ",
  "eduPersonAffiliation": "student",
  "eduPersonPrimaryAffiliation": "it",
  "title": "Undergraduate Student",
  "title;lang-el": "Προπτυχιακός Φοιτητής",
  "cn;lang-el": "ΓΕΩΡΓΙΟΣ ΠΑΠΑΔΟΠΟΥΛΟΣ",
  "cn": "GEORGIOS PAPADOPOULOS",
  "sn": "PAPADOPOULOS",
  "givenName": "GEORGIOS",
  "fathersname": "IOANNH",
  "secondarymail": "-",
  "telephoneNumber": "0",
  "labeledURI": "-",
  "id": "1234",
  "mail": "mail@mail.com",
  "pwdChangedTime": "20180808152441Z",
  "socialMedia": {
    "socialMediaExtra": []
  },
  "profilePhoto": ""
}
```


Σχήμα 6.4: JSON Object Containing User Profile Data [10]

Μετά τη διαδικασία σύνδεσης, όπως φαίνεται στο UI, εμφανίζεται το όνομα του χρήστη, η ακαδημαϊκή του ταυτότητα και ένα κουμπί για την επιλογή της τάξης στην οποία θέλει να πραγματοποιήσει παρουσία. Το κουμπί “Log In” αλλάζει σε “Log Out” το οποίο διαγράφει τα δεδομένα προφίλ του χρήστη και επαναφέρει την εφαρμογή στην αρχική της κατάσταση.

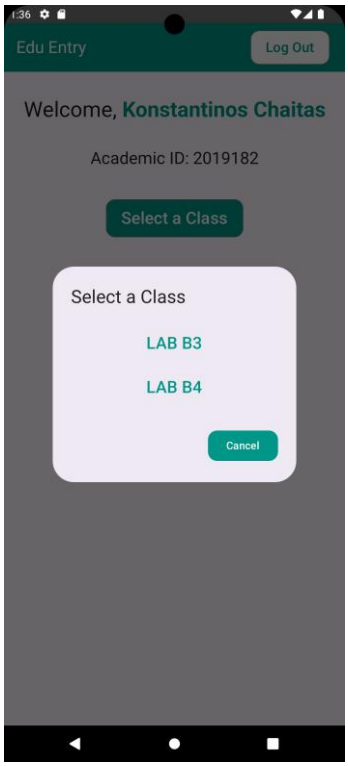


Σχήμα 6.5: Android Application (Select a Class – Welcome User) Page

6.4 Ανάκτηση Πληροφορίων Τάξης από το Firebase

Η διαδικασία λήψης πληροφοριών τάξης από το Firebase, όπως χρησιμοποιείται στην εφαρμογή Edu Entry στο Android, ακολουθεί πιστά την ίδια μεθοδολογία που περιγράφηκε νωρίτερα στην ενότητα σχετικά με το Raspberry Pi. Αυτή η μέθοδος διασφαλίζει ότι τα δεδομένα τάξης ανακτώνται σε πραγματικό χρόνο, επιτρέποντας στην εφαρμογή να ενημερώνεται δυναμικά και να παρέχει τρέχουσες πληροφορίες.

Όσον αφορά τη διεπαφή χρήστη, η εικόνα που παρέχεται δείχνει ένα παράθυρο διαλόγου που εμφανίζεται όταν ο χρήστης πατήσει το κουμπί "Select a Class". Αυτός ο διάλογος έχει σχεδιαστεί για να είναι φιλικός προς το χρήστη, εμφανίζοντας μια λίστα διαθέσιμων τάξεων (σε αυτήν την περίπτωση, "LAB B3" και "LAB B4") από τις οποίες ο χρήστης μπορεί να κάνει μια επιλογή. Το παράθυρο διαλόγου περιλαμβάνει επίσης ένα κουμπί "Cancel", επιτρέποντας στους χρήστες να κλείσουν το παράθυρο διαλόγου χωρίς να κάνουν επιλογή, εάν το επιθυμούν. Κάθε επιλογή ενεργοποιεί περαιτέρω διαδικασίες, όπως η δημιουργία και η εμφάνιση του κωδικού QR που σχετίζεται με τη συγκεκριμένη τάξη, τον οποίο στη συνέχεια χρησιμοποιούν οι φοιτητές για να δηλώσουν τη συμμετοχή τους μέσω σταθμών σάρωσης.



Σχήμα 6.6: Android Application Select a Class Dialog

6.5 Προετοιμασία και Επεξεργασία Δεδομένων Κώδικα QR

Στην εφαρμογή Edu Entry, η διαδικασία δημιουργίας κώδικα QR ξεκινά μόλις ο χρήστης επιλέξει μια τάξη. Ο κωδικός QR ενσωματώνει πολλές κρίσιμες πληροφορίες για να διασφαλίσει την ακριβή και ασφαλή επαλήθευση του χρήστη κατά τον έλεγχο και την καταγραφή της παρουσίας. Οι ενσωματωμένες πληροφορίες περιλαμβάνουν τη θεσμική σχέση (τύπος) του χρήστη, το όνομα και το επώνυμό του, ένα αναγνωριστικό χρήστη που προέρχεται από τον αριθμό μητρώου, φιλτραρισμένο ώστε να περιλαμβάνει μόνο ψηφία, τη διεύθυνση ηλεκτρονικού ταχυδρομείου του χρήστη και το συγκεκριμένο αναγνωριστικό τάξης που έχει επιλέξει ο χρήστης στο προηγούμενο βήμα. Αυτό το σύνολο δεδομένων είναι δομημένο σε μορφή JSON, προετοιμάζοντάς το για το επόμενο βήμα κρυπτογράφησης.

```
"type"      :  "student"              ,
"firstName" :  "Konstantinos"         ,
"lastName"  :  "Chaitas"              ,
"userId"    :  "2019182"              ,
"email"     :  "email@example.com"    ,
"classId"   :  "LAB B3"
```

Πίνακας 6.1: Student JSON Object

6.5.1 Κρυπτογράφηση και Εμφάνιση Κώδικα QR

Η διαδικασία κρυπτογράφησης στην εφαρμογή Edu Entry είναι ζωτικής σημασίας για την προστασία των ευαίσθητων πληροφοριών που περιέχονται σε κάθε κωδικό QR. Αυτή η διαδικασία χρησιμοποιεί κρυπτογράφηση AES, ένα ισχυρό πρότυπο που διασφαλίζει την εμπιστευτικότητα και την ακεραιότητα των δεδομένων. Ένα προκαθορισμένο κλειδί και ένα διάνυσμα αρχικοποίησης IV (**Initialization Vector**) χρησιμοποιούνται για την κρυπτογράφηση. Αυτά τα στοιχεία είναι ζωτικής σημασίας για τον αλγόριθμο AES, καθορίζοντας την ασφάλεια της διαδικασίας κρυπτογράφησης. Η συνάρτηση "**AESUtils.encrypt**" λαμβάνει τα σειριακά δεδομένα JSON, το κλειδί και το IV και στην συνέχεια επιστρέφει μια συμβολοσειρά με κωδικοποίηση Base64 που αντιπροσωπεύει τα κρυπτογραφημένα δεδομένα.

Η κρυπτογράφηση AES, σε συνδυασμό με τη λειτουργία **CBC** και το **PKCS5** padding, διασφαλίζει ότι τα δεδομένα είναι ασφαλή έναντι απειλών όπως η ανάλυση μοτίβων και οι επιθέσεις brute-force, οι οποίες αποτελούν κοινές ανησυχίες στα κρυπτογραφικά συστήματα. Η έξοδος αυτής της διαδικασίας κρυπτογράφησης είναι μια συμβολοσειρά που κωδικοποιείται με ασφάλεια και δεν είναι αναγνώσιμη χωρίς το κατάλληλο κλειδί αποκρυπτογράφησης και το IV, διατηρώντας έτσι το απόρρητο και την ασφάλεια των δεδομένων του χρήστη μέχρι να αποκωδικοποιηθούν από το εξουσιοδοτημένο σύστημα σάρωσης στο σημείο εισόδου της τάξης.

Τέλος, αφού έχουν προετοιμαστεί τα δεδομένα, μετατρέπονται σε μια εικόνα κώδικα QR. Έχοντας την κρυπτογραφημένη συμβολοσειρά δημιουργείται μια εικόνα bitmap του κωδικού QR, η οποία στη συνέχεια εμφανίζεται στη συσκευή του χρήστη, έτοιμη για σάρωση σε σημεία εισόδου εξοπλισμένα με τους σαρωτές Raspberry Pi.

Σχήμα 6.7: Android Application QR Code Screen

6.6 Σύνοψη

Το κεφάλαιο 6 αναλύει την ανάπτυξη της εφαρμογής Edu Entry για το Android, η οποία είναι σχεδιασμένη για να εξοπλίσει τα εκπαιδευτικά ιδρύματα με ένα αυτοματοποιημένο και αποδοτικό μέσο για την καταγραφή παρουσίας των φοιτητών. Η εφαρμογή, εκμεταλλευόμενη την τεχνολογία QR και τις δυνατότητες διαφόρων APIs, προσφέρει μια γρήγορη και ασφαλή μέθοδο για την ανάδειξη της ταυτότητας των χρηστών. Είναι κατασκευασμένη με στόχο την απλούστευση της διαδικασίας κατάθεσης παρουσίας, μειώνοντας τα λάθη που συνδέονται με την χειροκίνητη καταχώριση και ενισχύοντας την ακρίβεια και την αξιοπιστία των δεδομένων παρουσίας.

Μέσα από την ενσωμάτωση στα υπάρχοντα συστήματα ταυτοποίησης του πανεπιστημίου, παρέχει μια ομαλή και ασφαλή εμπειρία σύνδεσης για τους χρήστες, ενώ διασφαλίζει την ιδιωτικότητα και την ασφάλεια των εκπαιδευτικών δεδομένων. Το σύστημα ταυτοποίησης βασίζεται στο πρωτόκολλο OAuth 2.0, παρέχοντας έναν ασφαλή μηχανισμό για την πρόσβαση και την αλληλεπίδραση με το σύστημα του πανεπιστημίου. Ο σχεδιασμός της διεπαφής χρήστη και η αρχιτεκτονική της εφαρμογής προωθούν την αποδοτικότητα και την ευκολία χρήσης, καθιστώντας την εφαρμογή προσβάσιμη και ευχάριστη για όλους τους χρήστες. Τέλος, η δημιουργία και διαχείριση των κωδικών QR αποτελεί τον κεντρικό πυλώνα της εφαρμογής, με την κρυπτογράφηση δεδομένων να παίζει κρίσιμο ρόλο στην προστασία των πληροφοριών των χρηστών. Η δυνατότητα γρήγορης ανάκτησης πληροφοριών τάξης και η ενσωμάτωση με το Firebase ενισχύουν την απόδοση και τη δυνατότητα κλιμάκωσης της εφαρμογής.

Μέσω αυτού του κεφαλαίου, προσπαθούμε να αναδείξουμε πώς η εφαρμογή Edu Entry ανταποκρίνεται σε ένα σύνθετο εκπαιδευτικό πρόβλημα με τεχνολογική επιδεξιότητα, παρέχοντας ένα συνδυασμό από ασφάλεια, αξιοπιστία και ευκολία στη χρήση.

Κεφάλαιο 7ο: Συζήτηση και Συμπεράσματα

7.1 Περίληψη Πρακτικών Επιτευγμάτων

Αυτή η ενότητα παρέχει μια ολοκληρωμένη ανασκόπηση των κύριων στόχων που επιτεύχθηκαν μέσω της εφαρμογής του συστήματος ελέγχου παρουσίας φοιτητών χρησιμοποιώντας την τεχνολογία Raspberry Pi. Η ενσωμάτωση με το υπάρχον σύστημα του πανεπιστημίου για τον έλεγχο της ταυτότητας των χρηστών του συστήματος σηματοδότησε ένα σημαντικό ορόσημο, διευκολύνοντας την πρόσβασή τους, επιτυγχάνοντας ασφαλείς, συνεπείς αλληλεπιδράσεις με όλες τις πτυχές του συστήματος. Με την ενίσχυση των Google Sheets, τα οποία ήδη χρησιμοποιούσαν οι καθηγητές, το έργο μετέτρεψε αυτά τα εργαλεία σε έναν δυναμικό κόμβο αυτοματισμού. Ένα άλλο βασικό επίτευγμα ήταν η δημιουργία μιας κεντρικής βάσης δεδομένων για όλα τα διασυνδεδεμένα συστήματα, εξασφαλίζοντας τον συγχρονισμό δεδομένων σε πραγματικό χρόνο. Αυτή η συγκεντρωτική προσέγγιση βελτίωσε την αξιοπιστία και την ακρίβεια όλων των αρχείων του συστήματος ενώ επιπλέον διευκόλυνε την υλοποίηση και την επεκτασιμότητα του συστήματος.

Η ανάπτυξη ενός συστήματος που βασίζεται στο Raspberry Pi για τη διαχείριση της παρουσίας των φοιτητών αντιπροσωπεύει μια καθοριστική πρόοδο. Αυτό το σύστημα αυτοματοποιεί την καταγραφή της παρουσίας. Οι σταθμοί Raspberry Pi, τοποθετημένοι στις αίθουσες διδασκαλίας, επιτρέπουν την άμεση συλλογή και επεξεργασία δεδομένων, επιδεικνύοντας την αποτελεσματικότητα του συστήματος.

7.2 Προκλήσεις στην Πρακτική Υλοποίηση

Στην ενότητα αυτή θα εξετάσουμε τις τεχνικές και λειτουργικές προκλήσεις που αντιμετωπίστηκαν κατά την ανάπτυξη και την υλοποίηση του συστήματος. Ένα από τα κύρια τεχνικά εμπόδια αφορούσε το νέο λογισμικό που αναπτύχθηκε για το Raspberry Pi 5, ιδιαίτερα τις βιβλιοθήκες που χρησιμοποιούνται για την κάμερα. Αυτές οι προκλήσεις ανάγκασαν τη μετάβαση στη χρήση του Raspberry Pi 4 με μια παλαιότερη έκδοση του λειτουργικού συστήματος και μια τροποποιημένη προσέγγιση για τη λήψη εικόνων. Αρχικά το σύστημα προοριζόταν να επεξεργάζεται τη ζωντανή ροή του βίντεο από την κάμερα για να αποκωδικοποιεί τους κωδικούς QR. Λόγω προβλημάτων συμβατότητας με τις διαθέσιμες βιβλιοθήκες διαφοροποιήθηκε ο τρόπος εκτέλεσης αυτής της λειτουργείας.

Μια άλλη σημαντική πρόκληση προέκυψε στη χρήση των Google Apps Scripts που προσθέτουν αυτοματισμό στα Google Sheets. Οι λειτουργίες των Google Apps Scripts απαιτούσαν σημαντικό χρόνο επεξεργασίας, επηρεάζοντας αρνητικά την αποτελεσματικότητα και φτάνοντας τα όρια του χρόνου εκτέλεσης που υπάρχουν για τις δέσμες ενεργειών. Για να αντιμετωπιστεί αυτό, το σύστημα αναπροσαρμόστηκε για να εκτελεί υπολογισμούς και εγγραφές παρτίδας όπου ήταν δυνατόν, μειώνοντας σημαντικά τον χρόνο που απαιτείται για την επεξεργασία δεδομένων και βελτιώνοντας τη συνολική απόδοση του συστήματος χωρίς όμως να έχει εξαλειφθεί το πρόβλημα τελείως.

Η προστασία και η ασφάλεια των δεδομένων ήταν υψίστης σημασίας, δεδομένης της ευαίσθητης φύσης των εκπαιδευτικών δεδομένων. Το σύστημα σχεδιάστηκε με ισχυρά μέτρα ασφαλείας για τη διασφάλιση της ακεραιότητας των δεδομένων και της ιδιωτικότητας των χρηστών. Αυτό περιλαμβάνει ασφαλείς μηχανισμούς ελέγχου ταυτότητας για την πρόσβαση των χρηστών και κρυπτογραφημένη

αποθήκευση και μετάδοση δεδομένων, διασφαλίζοντας τη συμμόρφωση με τις βέλτιστες πρακτικές για την ασφάλεια των δεδομένων.

Συνολικά, ενώ το έργο αντιμετώπισε αρκετές προκλήσεις στην υλοποίησή του, έγιναν στρατηγικές προσαρμογές και βελτιστοποιήσεις για να ξεπεραστούν αυτά τα εμπόδια, οδηγώντας σε ένα λειτουργικό και αποτελεσματικό σύστημα.

7.3 Χρηστικότητα Συστήματος

Ο κύριος σκοπός αυτής της διπλωματικής εργασίας είναι η πρόταση ενός συστήματος διαχείρισης παρουσιών για φοιτητές και η υλοποίηση του. Η υλοποίηση περιλαμβάνει την ενσωμάτωση οικείων περιβαλλόντων χρήσης και τεχνολογιών όπως τα Google Sheets και το Android, οι οποίες στοχεύουν στην ελαχιστοποίηση της καμπύλης εκμάθησης και στην ενίσχυση της προσβασιμότητας του συστήματος.

Βασικά χαρακτηριστικά που βελτίωσαν την εμπειρία του χρήστη περιλαμβάνουν την αυτοματοποίηση της καταγραφής παρουσίας και της διαχείρισης της τάξης μέσω των Google Sheets. Αυτή η ενσωμάτωση επέτρεψε τις ενημερώσεις και τροποποιήσεις στους συμμετέχοντες φοιτητές καθώς και τη βαθμολόγηση της τάξης, αξιοποιώντας το φιλικό προς το χρήστη περιβάλλον της πλατφόρμας. Η εφαρμογή στο Android, σχεδιασμένη με γνώμονα την απλότητα, παρέχει μια απλή διεπαφή για τη διαχείριση των κωδικών QR που χρησιμοποιούν οι φοιτητές για καταγράψουν την παρουσία τους, καθιστώντας την προσβάσιμη ακόμη και για χρήστες με περιορισμένη τεχνική εξειδίκευση.

7.4 Μελλοντικές Κατευθύνσεις

Αυτή η ενότητα περιγράφει αρκετές σημαντικές τεχνολογικές βελτιώσεις και επεκτάσεις που σχεδιάζονται για το σύστημα, με στόχο τη βελτίωση τόσο της απόδοσης όσο και της εμπειρίας των χρηστών.

Μια κρίσιμη αναβάθμιση είναι η ενσωμάτωση μιας οθόνης στις συσκευές Raspberry Pi. Αυτή η οθόνη θα παρέχει σχόλια και μηνύματα σε πραγματικό χρόνο στους χρήστες, όπως "Καθυστερήσατε κατά 10 λεπτά στην είσοδο σας στην τάξη" ή "Δεν είστε εγγεγραμμένοι σε αυτό το μάθημα". Αυτή η διαδραστική ανατροφοδότηση είναι ζωτικής σημασίας για την άμεση πληροφόρηση των χρηστών και ενισχύει τη διαδικασία αντιμετώπισης προβλημάτων, καθιστώντας το σύστημα πιο φιλικό προς το χρήστη και ανταποκρινόμενο στις καθημερινές προκλήσεις.

Η βάση δεδομένων Firebase θα επεκταθεί για να συμπεριλάβει μια νέα συλλογή για συνδέσεις χρηστών μέσω της εφαρμογής Android. Η εξέλιξη αυτή στοχεύει στην απλοποίηση της διαδικασίας πρόσβασης στο σύστημα και στη μείωση της εξάρτησης από τους πανεπιστημιακούς διακομιστές για τον έλεγχο της ταυτότητας των χρηστών. Επιτρέποντας μια πιο απλοποιημένη και αξιόπιστη διαδικασία σύνδεσης, το σύστημα θα ενισχύσει τη χρηστικότητα και την ασφάλειά του. Περαιτέρω βελτιώσεις θα περιλαμβάνουν τη δημιουργία νέων συνδέσεων μεταξύ συλλογών και τη βελτιστοποίηση των ερωτημάτων (queries). Αυτές οι βελτιώσεις θα διευκολύνουν την ανάπτυξη άλλων στοιχείων του συστήματος και θα βελτιώσουν τη συνολική απόδοση και ταχύτητα του.

Η εφαρμογή Android θα εισαγάγει ανεξάρτητες μεθόδους σύνδεσης, επιτρέποντας μεγαλύτερη ευελιξία και ασφάλεια στον έλεγχο ταυτότητας του χρήστη. Αυτή η αλλαγή είναι ιδιαίτερα σημαντική σε περιβάλλοντα όπου η αξιοπιστία του πανεπιστημιακού διακομιστή μπορεί να είναι ένα ζήτημα. Νέες διεπαφής χρήστη θα προστεθούν για να εμφανίσουν τα αρχεία παρακολούθησης και τους βαθμούς των φοιτητών για κάθε τάξη. Αυτή η δυνατότητα επεκτείνει τη λειτουργικότητα από τους εκπαιδευτικούς

στους μαθητές, παρέχοντάς τους άμεση πρόσβαση στα εκπαιδευτικά τους δεδομένα. Πρόσθετες λειτουργίες για τους εκπαιδευτικούς θα περιλαμβάνουν τη δυνατότητα διαχείρισης τάξεων και εισαγωγής βαθμών απευθείας από τις κινητές τους συσκευές. Αυτό θα ενισχύσει σημαντικά την ευελιξία και την αποτελεσματικότητα της ακαδημαϊκής διαχείρισης. Για την περαιτέρω υποστήριξη των φοιτητών χωρίς πρόσβαση μέσω κινητού, θα προστεθεί λειτουργικότητα για την αυτόματη δημιουργία κωδικών QR και την αποστολή τους στα email για την είσοδο στην τάξη. Αυτό εξασφαλίζει ότι όλοι οι φοιτητές, ανεξάρτητα από την πρόσβασή τους στην τεχνολογία αυτή τη στιγμή, μπορούν να συμμετάσχουν στο σύστημα ελέγχου παρουσίας.

Θα αναπτυχθεί μια έκδοση iOS της εφαρμογής για να διασφαλιστεί η προσβασιμότητα σε διαφορετικές πλατφόρμες κινητής τηλεφωνίας. Αυτή η επέκταση είναι ζωτικής σημασίας για την εξυπηρέτηση χρηστών που προτιμούν τις συσκευές iOS, ενισχύοντας έτσι το ποσοστό υιοθέτησης και την εμβέλεια της εφαρμογής.

Τα Google Apps Scripts θα αναβαθμιστούν για τη διαχείριση ξεχωριστών φύλλων για διαφορετικούς εκπαιδευτικούς. Επί του παρόντος, το σύστημα λειτουργεί σε ένα ενοποιημένο μοντέλο όπου όλοι οι εκπαιδευτικοί έχουν πρόσβαση σε ένα Google Sheet, το οποίο μπορεί να δημιουργήσει κινδύνους ασφαλείας και προκλήσεις διαχείρισης δεδομένων. Παρέχοντας ατομική πρόσβαση στα αντίστοιχα φύλλα, το σύστημα θα βελτιωθεί σημαντικά την ασφάλεια και την εξατομικευμένη διαχείριση των δεδομένων.

Αυτές οι μελλοντικές κατευθύνσεις στοχεύουν να αντιμετωπίσουν τους τρέχοντες περιορισμούς και να ενισχύσουν τη λειτουργικότητα του συστήματος, καθιστώντας το πιο ισχυρό, φιλικό προς το χρήστη και προσαρμόσιμο στις εξελισσόμενες ανάγκες των εκπαιδευτικών ιδρυμάτων.

7.5 Σύνοψη

Το συγκεκριμένο κεφάλαιο ολοκληρώνεται συνοψίζοντας τα βασικά σημεία συζήτησης, προβάλλοντας κατά κύριο λόγο τον θεωρητικό βαθμό απόδοσης του εφαρμοζόμενου συστήματος διαχείρισης παρουσίας φοιτητών χρησιμοποιώντας τη τεχνολογία Raspberry Pi ως πιθανή υποψήφια κινητήρια δύναμη για την αποτελεσματικότητα και την καινοτομία. Εστιάζοντας στα ενδεχόμενα οφέλη τα οποία θα μπορούσαν να προκύψουν μέσω των προτεινόμενων μελλοντικών κατευθύνσεων.

Το σύστημα εμφανίζεται να επιδεικνύει θεωρητικά μια σειρά από χρηστικά αποτελέσματα τα οποία παρουσιάζουν ενδιαφέρον. Θα μπορούσε να αναφερθεί ότι με την επίτευξη σε ικανοποιητικό βαθμό της ενσωμάτωσής του με τα ήδη υπάρχοντα πανεπιστημιακά συστήματα ελέγχου ταυτότητας χρηστών, αλλά και με την ενίσχυση κατά μεγάλου ποσοστού της χρήσης των Google Sheets, επιτεύχθηκε ο βασικός στόχος αναφορικά με τη διαχείριση της τάξης, όπως επίσης και με τη δημιουργία μιας κεντρικής βάσης δεδομένων η οποία καθιστά δυνατό τον συγχρονισμό των δεδομένων αυτών.

Εύλογα θα μπορούσε να σημειωθεί ότι οι προαναφερθείσες προσπάθειες θα μπορούσαν πιθανότατα να συμβάλλουν στην εξοικονόμηση χρόνου κατά τη διδακτική διαδικασία μέσω της αυτοματοποιημένης παρακολούθησης της παρουσίας των φοιτητών στα εκπαιδευτικά ιδρύματα στα οποία φοιτούν. Επίσης, θα μπορούσαν να παρέχουν τη βάση για τη συνεχή βελτίωση αναφορικά με την ακρίβεια και την αξιοπιστία των δεδομένων, με κύριο στόχο την εξασφάλιση της αποτελεσματικότητας στο μέγιστο δυνατό βαθμό και τον εκσυγχρονισμό των εκπαιδευτικών πρακτικών.

Οι προκλήσεις οι οποίες τέθηκαν στο επίκεντρο προς επίλυση κατά την υλοποίηση του συστήματος, όπως ήταν οι περιορισμοί υλικού με το Raspberry Pi 5 αλλά και τα ζητήματα ενσωμάτωσης λογισμικού, θα μπορούσε να θεωρηθεί ότι αντιμετωπίστηκαν σε κάποιο βαθμό μέσω στρατηγικών προσαρμογών

οι οποίες επιλέχθηκαν ώστε να οδηγήσουν στην εξασφάλιση τη λειτουργικότητας και την αποτελεσματικότητας του συστήματος. Επιπλέον, η χρηστικότητα του συστήματος θα μπορούσε εν μέρει να αξιολογηθεί μέσω των πολλαπλών προσωπικών δοκιμών και της παράλληλης ανατροφοδότησης από την επιβλέπουσα καθηγήτρια, δίνοντας έμφαση στα στοιχεία της λειτουργικότητας και του φιλικού σχεδιασμού προς τον χρήστη, την εμβάθυνση στις επιπτώσεις και στα οφέλη της ενσωμάτωσης του εν λόγω συστήματος με τα αντίστοιχα υπάρχοντα πανεπιστημιακά συστήματα.

Οραματίζοντας αισιόδοξα το μέλλον, ευχή θα ήταν το εν λόγω σύστημα να μπορούσε ίσως να κριθεί σε ένα βαθμό ικανό και να δεχθεί τεχνολογικές βελτιώσεις, συμπεριλαμβανομένης της ανάπτυξης μιας εφαρμογής iOS αλλά και επεμβάσεων προς τη βελτίωση της εφαρμογής Android, προκειμένου να επιτευχθεί η δυνατότητα διαχείρισης μεμονωμένων δεδομένων των χρηστών. Οι τεχνολογικές αυτές βελτιώσεις, απώτερο στόχο θα είχαν να καταστήσουν το σύστημα προσαρμόσιμο στις εκάστοτε εξελισσόμενες ανάγκες των εκπαιδευτικών ιδρυμάτων, να θέσουν προς αξιοποίηση τη δυνατότητα λήψης και συλλογής δεδομένων σύμφωνα με τα αρχεία πραγματικού χρόνου ώστε να βελτιστοποιηθούν οι υπάρχουσες διδακτικές πρακτικές όπου αυτό απαιτείται.

Σύμφωνα με τα όσα έχουν προαναφερθεί, το εν λόγω σύστημα συνολικά, θεωρητικά και όχι πρακτικά αποδεδειγμένα, φαίνεται να στοχεύει σε έναν βαθμό στην ευρεία βελτίωση της διοικητικής απόδοσης, παρέχοντας άμεσα και ταυτόχρονα ακριβή, έγκυρα εκπαιδευτικά δεδομένα, τα οποία μπορούν να συμβάλλουν καθοριστικά αν και όπου κρίνεται ότι απαιτείται εκπαιδευτικός επανασχεδιασμός και αξιολόγηση του προγράμματος σπουδών. Ως εκ τούτου, η συνέχιση του συγκεκριμένου έργου, ενσωματώνοντας περαιτέρω βελτιώσεις και καινοτομίες, θα μπορούσε να ενισχύσει τόσο την εμπειρία του χρήστη όσο και τις λειτουργικές δυνατότητες του συστήματος, παρέχοντας ολοκληρωμένες λύσεις στο ταχέως εξελισσόμενο εκπαιδευτικό τοπίο για την κάλυψη των εμφανιζόμενων αναγκών στη σύγχρονη εκπαίδευση.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Dworkin, M. (2023), Advanced Encryption Standard (AES), Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards and Technology, Gaithersburg, MD, [online], https://doi.org/10.6028/NIST.FIPS.197-upd1, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=936594 (Accessed May 1, 2024)

[2] Medium “What is Firebase? The complete story, abridged” [Online]. Available: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

[3] RasberryPi “raspberry pi 4 model b specifications” [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/

[4] DFRobot “Gravity: Digital Buzzer for Arduino / ESP32 / micro:bit / Raspberry Pi” [Online]. Available: https://www.dfrobot.com/product-84.html`

[5] DFRobot “Gravity: URM09 Ultrasonic Distance Sensor (2～500cm, Analog)” [Online]. Available: https://www.dfrobot.com/product-1862.html

[6] Grobotronics “LED Traffic Light Display Module 5V” [Online]. Available: https://grobotronics.com/led-traffic-light-display-module-5v.html?sl=en#group_13656662bfeeb4d74d-1

[7] Grobotronics “Raspberry Pi Camera Module V3 - Standard 11.9MP 75°” [Online]. Available: https://grobotronics.com/raspberry-pi-camera-module-v3-standard-11.9mp-75.html?sl=en

[8] Raspberry Pi Projects “Physical Computing with Python” [Online]. Available: https://projects.raspberrypi.org/en/projects/physical-computing/13

[9] Android Developers Core Areas UI Guides “Material Design 3 in Compose” [Online]. Available: https://developer.android.com/develop/ui/compose/designsystems/material3

[10] Λήψη ενός Access Token “Authorization code (Server Side εφαρμογές)” [Online]. Available: https://login.iee.ihu.gr/apps/doc

[11] S. Tiwari, "An Introduction to QR Code Technology," 2016 International Conference on Information Technology (ICIT), Bhubaneswar, India, 2016, pp. 39-44, doi: 10.1109/ICIT.2016.021. keywords: {Error correction codes; Encoding; Decoding; Timing; Particle separators; Uniform resource locators; Arrays; QR code; Quick Response code; QR code structure; QR Code Encoding; QR Code Decoding}

[12] A. A. Frozza, R. d. S. Mello and F. d. S. d. Costa, "An Approach for Schema Extraction of JSON and Extended JSON Document Collections," 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, USA, 2018, pp. 356-363, doi: 10.1109/IRI.2018.00060. keywords: {Databases; Arrays; Data mining; Task analysis; Data models; Social network services; NoSQL; JSON; Extended JSON; Schema Extraction; JSON Schema; Document-Oriented Database},

[13] M. Vaidehi and B. J. Rabi, "Design and analysis of AES-CBC mode for high security applications," Second International Conference on Current Trends In Engineering and Technology - ICCTET 2014, Coimbatore, India, 2014, pp. 499-502, doi: 10.1109/ICCTET.2014.6966347. keywords: {Encryption; Satellites; Ciphers; Hardware; Circuit faults; Conferences; AES; CBC mode; Initialization Vectors; Electronic Code Block},

[14] M. Kusuma, A. H. Rifani and B. Sugiantoro, "Comparison analysis of Jetpack Compose and Flutter in Android-based application development using Technical Domain," 2023 Eighth International Conference on Informatics and Computing (ICIC), Manado, Indonesia, 2023, pp. 1-5, doi: 10.1109/ICIC60109.2023.10381987. keywords: {Computer languages; Calculators; Social networking (online); Operating systems; Memory management; Multimedia Web sites; XML; android; framework; domain; Jetpack Compose; Flutter; performance; testability},

[15] E. Abdelhamid et al., "How Global Retailer ADEO Migrated to Google BigQuery with Database Virtualization," 2023 IEEE International Conference on Big Data (BigData), Sorrento, Italy, 2023, pp. 1895-1898, doi: 10.1109/BigData59044.2023.10386640. keywords: {Databases; Data warehouses; Big Data; Internet; Virtualization; database virtualization; data warehousing; database migrations; Teradata; Google BigQuery},

[16] K. M. Varma and G. B. Se, "Efficient Scalable Migrations in the Cloud," 2022 IEEE/ACIS 7th International Conference on Big Data, Cloud Computing, and Data Science (BCD), Danang, Vietnam, 2022, pp. 3-6, doi: 10.1109/BCD54882.2022.9900725. keywords: {Cloud computing; Databases; Scalability; Pipelines; Companies; Maintenance engineering; Software; Cloud migration; Refactoring; Re-platforming; Teradata; BigQuery; Microsoft Azure; Google Cloud Platform.},

[17] P. R. R. Fernando, "Improving the quality of education system using Data Science Technologies: Survey," 2020 5th International Conference on Innovative Technologies in Intelligent Systems and Industrial Applications (CITISIA), Sydney, Australia, 2020, pp. 1-6, doi: 10.1109/CITISIA50690.2020.9371793. keywords: {Education; Warehousing; Decision making; Data science; Data warehouses; Data mining; Unemployment; Data Science; Data warehousing in education; data warehouse potentiality in education sector; big data in education; data mining in education; business intelligence},

[18] Bowen, E., Price, T., Lloyd, Steve., & Thomas, S. (2005). Improving the quantity and quality of attendance data to enhance student retention. Journal of Further and Higher Education, 29(4), 375–385. https://doi.org/10.1080/03098770500353714

[19] P. H. S. Panahy, F. Sidi, L. S. Affendey and M. A. Jabar, "The impact of data quality dimensions on business process improvement," 2014 4th World Congress on Information and Communication Technologies (WICT 2014), Melaka, Malaysia, 2014, pp. 70-73, doi: 10.1109/WICT.2014.7077304. keywords: {Information systems; Organizations; Accuracy; Quality assessment; Correlation; Data mining; Data Quality; Data quality Dimension; Business Process Improvement; Information System; Data Quality Problem},

[20] O. Kovalenko, Y. Palamarchuk and R. Yatskovska, "Assessing the level of maturity of the automated management system of a higher education institution," 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), LVIV, Ukraine, 2021, pp. 167-172, doi: 10.1109/CSIT52700.2021.9648663. keywords: {Learning management systems; Automation; Electronic learning; Education; Ecosystems; Organizations; Mobile applications; training management system and support of management processes; educational processes; automation; maturity of automation system; learning process management; management of a higher educational institution; automated processes; maturity of learning management systems and management of educational institutions},

[21] W. Y. Mok, "A Logical Database Design Methodology for MongoDB NoSQL Databases," 2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM),

Singapore, Singapore, 2021, pp. 1451-1455, doi: 10.1109/IEEM50564.2021.9673004. keywords: {Databases; Design methodology; NoSQL databases; Engineering management; Conferences; Unified modeling language; Semantics; Logical Database Design Methodology; NoSQL Databases; MongoDB; MongoDB's collections; Scheme Tress; Entity-Relationship Models},

[22] C. Praschl, S. Pritz, O. Krauss and M. Harrer, "A Comparison Of Relational, NoSQL and NewSQL Database Management Systems For The Persistence Of Time Series Data," 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Maldives, Maldives, 2022, pp. 1-6, doi: 10.1109/ICECCME55909.2022.9988333. keywords: {Mechatronics; NoSQL databases; Time series analysis; Random access memory; Sensor phenomena and characterization; Data models; Real-time systems; Database; Time Series; Big Data; SQL},

ΠΑΡΑΡΤΗΜΑ Α : Google Apps Scripts Code

Auth.js

```
/**
 * Generates an authentication token for accessing Firestore.
 * This token is used in the authorization header for HTTP requests to
Firestore's REST API.
 *
 * @returns {string} An authentication token.
 */

function getAuthToken() {

  const url = 'https://accounts.google.com/o/oauth2/token'; // Endpoint for
obtaining OAuth2 tokens

  // Assemble the POST body with the appropriate grant type and the JWT
  const payload = {
    grant_type: 'urn:ietf:params:oauth:grant-type:jwt-bearer',
    assertion: createJwt(clientEmail, privateKey, projectId), // Use the
createJwt function to generate the assertion (JWT)
  };

  // Set the options for the HTTP request
  const options = {
    method: 'post', // Use POST method
    contentType: 'application/x-www-form-urlencoded', // Set content type for
form urlencoded data
    payload: payload, // Attach the payload to the request body
  };

  // Send the request and parse the response
  const response = UrlFetchApp.fetch(url, options); // Make the request to the
OAuth2 token endpoint
  const result = JSON.parse(response.getContentText()); // Parse the JSON
response

  // Return the access token from the response
  return result.access_token; // The token is used in subsequent requests to
Firestore

}
```

```
/**
 * Creates a JSON Web Token (JWT) for OAuth2 authentication with Firestore.
 * This JWT is part of the OAuth 2.0 assertion flow where it asserts the
identity of the application to Google's OAuth 2.0 server.
 *
 * @param {string} clientEmail - The email address of the client service
account from GCP.
 * @param {string} privateKey - The private key associated with the service
account, used to sign the JWT.
 * @param {string} projectId - The ID of the GCP project that contains the
Firestore database.
 * @returns {string} A JWT for authentication.
 */

function createJwt(clientEmail, privateKey, projectId) {

  // Define JWT header specifying the algorithm for signing the JWT (RS256)
  const header = {
    alg: 'RS256', // RSA SHA-256 algorithm
    typ: 'JWT', // Type of token
  };

  // Define the payload of the JWT, including the standard claims
  const payload = {

    iat: Math.floor(Date.now() / 1000), // Issued at time (in seconds)
    exp: Math.floor(Date.now() / 1000) + 3600, // Expiration time (1 hour
after iat)
    iss: clientEmail, // Issuer (service account email)
    aud: 'https://accounts.google.com/o/oauth2/token', // Audience (OAuth2
token endpoint)
    scope: 'https://www.googleapis.com/auth/datastore', // Scopes indicating
which services the access token is for
  };

  // Encode the JWT Header and JWT Claims Set to URL-safe Base64
  const encodedHeader = Utilities.base64EncodeWebSafe(JSON.stringify(header));
  const encodedPayload =
Utilities.base64EncodeWebSafe(JSON.stringify(payload));

  // Join the encoded header and payload with a period as per the JWT standard
  const toSign = `${encodedHeader}.${encodedPayload}`;

  // Sign the UTF-8 representation of the input using SHA256withRSA
  const signature = Utilities.computeRsaSha256Signature(toSign, privateKey);

  // Encode the signature to URL-safe Base64
  const encodedSignature = Utilities.base64EncodeWebSafe(signature);
```

```
  // Concatenate the encoded header, payload, and signature with periods
  return `${encodedHeader}.${encodedPayload}.${encodedSignature}`;

}
```

Calculations.js

```
function calculateAttendanceAndGradesSummary(sheet, classDates, totalStudents)
{
    // Starting from the second student data row, since the first row is the
header
    for (let i = 2; i <= totalStudents * 2; i += 2) {
      // Fetch attendance and grade data for the student
      const attendanceData = sheet.getRange(i, 5, 1,
classDates.length).getValues()[0];
      const gradeData = sheet.getRange(i + 1, 5, 1,
classDates.length).getValues()[0];

      // Process attendance
      const totalAttendances = attendanceData.filter(value =>
String(value).includes('✓')).length;
      const attendanceStatus = (classDates.length - totalAttendances >= 2) ?
'Failed' : 'OK';
      const attendanceSummary = `${totalAttendances} of ${classDates.length} -
(${attendanceStatus})`;

      // Process grades
      const validGrades = gradeData.map(value => parseFloat(value) || 0); //
Assign 0 for missed classes
      const sumGrades = validGrades.reduce((sum, value) => sum + value, 0);
      const meanGrade = (sumGrades / classDates.length).toFixed(2); // Now
dividing by the total number of classes

      // Determine nominal value
      let nominalGrade = "Failed"; // Default value
      if (parseFloat(meanGrade) >= 5) {
        // Adding more nominal values between 5 and 10 with a step of 1
        const roundedGrade = Math.floor(parseFloat(meanGrade));
        switch(roundedGrade) {
```

```
          case 5: nominalGrade = "Sufficient"; break;
          case 6: nominalGrade = "Satisfactory"; break;
          case 7: nominalGrade = "Good"; break;
          case 8: nominalGrade = "Very Good"; break;
          case 9: nominalGrade = "Excellent"; break;
          case 10: nominalGrade = "Outstanding"; break;
          default: nominalGrade = "Passed"; break; // For non-integer values
between 5 and 6
        }
      }

      const gradeSummary = `${meanGrade} - (${nominalGrade})`; // Combining
the mean grade and the nominal value

      // Set values in the sheet
      const attendanceSummaryColumn = classDates.length + 5;
      const gradeSummaryColumn = classDates.length + 6;

      // Assuming the Total Attendances and Grade are the next two columns
after all the dates
      sheet.getRange(i, attendanceSummaryColumn).setValue(attendanceSummary);
      sheet.getRange(i+1, gradeSummaryColumn).setValue(gradeSummary);
    }
  }
```

Credentials.js

```
// Firestore API information
const firestoreApiUrl = '';
const privateKey = '-----BEGIN PRIVATE KEY----- -----END PRIVATE KEY-----\n';
const clientEmail = '';
const projectId = '';


// Google Sheet information
const spreadsheetId = '';
```

FirebaseToSheets.js

```
/**
 * Prompts the user to select a class to sync and stores the selection.
 * Triggers the synchronization process for the selected class.
 */

function selectClassToSync() {
  const ui = SpreadsheetApp.getUi();
  // Prompt the user to enter the name of the class to sync
  const response = ui.prompt('Enter the name of the class to sync:');

  // Check if the user clicked "OK" and provided a response text
  if (response.getSelectedButton() == ui.Button.OK &&
response.getResponseText()) {
    // Store the classId (sheetName) in Script Properties
    PropertiesService.getScriptProperties().setProperty('classToSync',
response.getResponseText().trim());
    // Trigger synchronization for the selected class
    syncSelectedClass();
  }
}

/**
 * Synchronizes data from Firestore for the stored classId from the script
properties.
 */

function syncSelectedClass() {
  // Retrieve the classId (sheetName) from Script Properties
  const classToSync =
PropertiesService.getScriptProperties().getProperty('classToSync');

  // If a classId was found, proceed with synchronization
  if (classToSync) {
    syncFirestoreToSheets(classToSync);
    // Clear the stored classId from the script properties after syncing
    PropertiesService.getScriptProperties().deleteProperty('classToSync');
  } else {
    // Alert the user if no class name was found in the properties
    SpreadsheetApp.getUi().alert('Class name not found.');
  }
}

/**
 * Triggers the synchronization process for all classes.
 * This function can be called directly without specifying a classId.
 */

function syncAllClasses() {
```

```
  syncFirestoreToSheets();
}

/**
 * Synchronizes data from Firestore into the spreadsheet for a specific class
or all classes.
 * @param {string|null} classId - The ID of the class to sync. If null, syncs
all classes.
 */

function syncFirestoreToSheets(classId = null) {
  const authToken = getAuthToken();
  // Open the spreadsheet by ID
  const ss = SpreadsheetApp.openById(spreadsheetId);

  // Determine if syncing a single class or all classes based on the provided
classId
  const classesToSync = classId ? [{ fields: { classId: { stringValue: classId
} } }] : getCollection('Classes', authToken);

  // Iterate over each class to sync
  classesToSync.forEach(classDocument => {
    const classData = classDocument.fields;
    const classId = classData.classId.stringValue;

    // Get the sheet by name or create a new one if it doesn't exist, then
clear it
    let sheet = ss.getSheetByName(classId) || ss.insertSheet(classId);
    sheet.clear();

    // Fetch dates for the class and set them as headers in the sheet
    const classDates = fetchClassDates(classId, authToken);
    const headerRow = ["UserID", "LastName", "FirstName", "Email",
...classDates, "Attendances", "Grade"];
    sheet.appendRow(headerRow);

    // Fetch students in the class from Firestore
    const studentCollectionPath = `Classes/${classId}/Students`;
    const students = getCollection(studentCollectionPath, authToken);

    // Check if any students were found
    if (students && students.length > 0) {

      // Process each student's data
      students.forEach(studentDocument => {

        // Extract student data from Firestore document
        const studentData = studentDocument.fields;
```

```
        // Initialize student data, with a fallback to an empty string if a
field is not present
        const userId = studentData.userId ? studentData.userId.stringValue :
'';
        const lastName = studentData.lastName ?
studentData.lastName.stringValue : '';
        const firstName = studentData.firstName ?
studentData.firstName.stringValue : '';
        const email = studentData.email ? studentData.email.stringValue : '';

        // Fetch attendance records for the student
        const attendanceRecords = fetchStudentAttendance(classId, userId,
authToken);

        // Prepare to record attendance and grades in the spreadsheet
        const attendanceMarks = [];
        const attendanceGrades = [];

        // Process each date for the class
        classDates.forEach((date, index) => {

          // Check for an attendance record on the given date
          const attendance = attendanceRecords.find(record => record.date ===
date);
          if (attendance) {

            // Record attendance with the time of attendance
            attendanceMarks.push(`✓ (${attendance.time})`);

            // Add the grade if it exists, or prompt to fill out if not
            attendanceGrades.push(attendance.grade || "Fill Out");
          } else {

            // Mark as absent and no grade for this date
            attendanceMarks.push('X');
            attendanceGrades.push("-");

          }
        });

        // Compile the student's row data and append to the sheet
        const studentRow = [userId, lastName, firstName, email,
...attendanceMarks];
        sheet.appendRow(studentRow);

        // Append a separate row for grades beneath the attendance marks
        const gradeRow = Array(4).fill(""); // Align with grade columns
```

```
        sheet.appendRow(gradeRow.concat(attendanceGrades));
      });

      // Summarize attendance and grades after all students are processed
      calculateAttendanceAndGradesSummary(sheet, classDates, students.length);

      // Apply formatting to the sheet for readability
      formatSheet(sheet, classDates.length + 6);

      // Apply color coding based on attendance and grades
      setColorForAttendanceAndGrades(sheet, classDates);

    } else {

      // Log to the console if no students are found for the class
      console.log(`No students found for class ${classId}`);

    }
  });
}
```

FirestoreAPI.js

```
/**
 * This function is responsible for fetching data from a specified Firestore
collection.
 * @param {string} collectionPath - The path to the Firestore collection to
fetch data from.
 * @param {string} authToken - The Bearer token used for authenticating the
API request.
 * @returns {Array} An array of document data from the specified Firestore
collection.
 */

function getCollection(collectionPath, authToken) {

  // Set up the authorization headers using the authToken
  const headers = {
    Authorization: 'Bearer ' + authToken,
    'Content-Type': 'application/json',
  };

  // Configure the HTTP GET request options
  const options = {
    headers: headers,
    method: 'GET',
    muteHttpExceptions: true,
  };
```

```
  // Perform the network request to Firestore and parse the response as JSON
  const response = UrlFetchApp.fetch(firestoreApiUrl + collectionPath,
options);
  const result = JSON.parse(response.getContentText());

  // Return the documents within the collection as an array
  return result.documents;

}

/**
 * Retrieves the dates associated with a class schedule from Firestore.
 * @param {string} classId - The identifier of the class to get dates for.
 * @param {string} authToken - The authentication token for accessing
Firestore.
 * @returns {Array<string>} An array of dates (in string format) representing
the class schedule.
 */

function fetchClassDates(classId, authToken) {

  // Construct the path to the schedule collection for the class
  const schedulePath = `Classes/${classId}/Schedule`;

  // Fetch the schedule documents using the getCollection helper function
  const scheduleDocs = getCollection(schedulePath, authToken);

  // Handle the case where no documents were returned
  if (!scheduleDocs) {
    console.error('Failed to fetch schedule documents or none exist for the
class:', classId);
    return []; // Return an empty array as a fallback
  }

  // Map over the schedule documents to extract the date string, filtering out
any malformed documents
  const classDates = scheduleDocs.map(doc => {

    // Check that the document has a fields object with a date property
    if (doc.fields && doc.fields.date && doc.fields.date.stringValue) {
      return doc.fields.date.stringValue; // Extract the date string
    } else {
      console.error('Document is missing a valid date field:', doc);
      return null; // Use null to indicate an invalid or missing date field
    }
  }).filter(date => date); // Remove any null entries from the array
```

```
  // Sort the dates in ascending order
  classDates.sort((a, b) => new Date(a).getTime() - new Date(b).getTime());

  // Return the sorted array of date strings
  return classDates;

}

/**
 * Retrieves attendance records for a specific student within a class from
Firestore.
 * @param {string} classId - The ID of the class to fetch attendance for.
 * @param {string} userId - The ID of the student whose attendance is to be
fetched.
 * @param {string} authToken - The authentication token for accessing
Firestore.
 * @returns {Array<Object>} An array of objects, each representing an
attendance record with date, time, and grade.
 */

function fetchStudentAttendance(classId, userId, authToken) {

  // Define the path to the student's attendance documents
  const attendancePath = `Classes/${classId}/Students/${userId}/Attendances`;

  // Fetch the attendance documents for the student
  const attendanceDocs = getCollection(attendancePath, authToken);

  // Handle the case where no attendance documents are returned
  if (!attendanceDocs || !Array.isArray(attendanceDocs)) {
    console.error(`Failed to fetch attendance for student ${userId} in class
${classId}`);
    return []; // Return an empty array as a fallback
  }

  // Map over the attendance documents to create a structured representation
of the attendance data
  return attendanceDocs.map(doc => {

    // Extract date, time, and grade from the document, using empty strings as
fallbacks
    const attendanceDate = doc.fields.date.stringValue;
    const attendanceTime = doc.fields.time.stringValue;
    const grade = doc.fields.grade ? doc.fields.grade.stringValue : '';

    // Return an object representing the attendance record
    return { date: attendanceDate, time: attendanceTime, grade: grade };
  });
```

```
}

/**
 * Updates an existing Firestore document or creates a new one if it doesn't
exist.
 * This function uses a PATCH request, which is suitable for update operations
and also creates a document if not present.
 *
 * @param {string} path - The Firestore path where the document should be
created or updated.
 * @param {object} data - The data to be written to the Firestore document.
 * @param {string} authToken - The authentication token to authorize the API
request.
 */

function updateOrCreateFirestoreDocument(path, data, authToken) {

  // Validate the inputs to make sure they're not empty or just whitespace.
  if (!path || path.trim() === '' || !data) {
    Logger.log("Invalid path or data provided to
updateOrCreateFirestoreDocument function.");
    return; // Exit the function if inputs are not valid.
  }

  // Ensure the Firestore path is correctly formatted without a trailing
slash.
  const trimmedPath = path.endsWith('/') ? path.slice(0, -1) : path;

  // Construct the complete URL for the Firestore API endpoint.
  const url = `https://firestore.googleapis.com/v1/projects/edu-entry-iee-
ihu/databases/(default)/documents/${trimmedPath}`;

  // Set up the options for the PATCH request.
  const options = {
    method: 'PATCH', // Use PATCH method to update or create a document.
    headers: {
      'Authorization': `Bearer ${authToken}`, // Use the authToken to
authenticate.
      'Content-Type': 'application/json', // Set the content type as JSON.
    },
    payload: JSON.stringify(data), // Convert the data object to a JSON string
for the request body.
    muteHttpExceptions: true // Prevent throwing exceptions for HTTP errors to
handle them manually.
  };

  // Try to make the HTTP request to Firestore.
```

```
  try {
    const response = UrlFetchApp.fetch(url, options);
    // Log the response for debugging purposes.
    Logger.log(`Response: ${response.getContentText()}`);
  } catch (e) {
    // Catch and log any errors that occur during the fetch operation.
    Logger.log(`Exception occurred: ${e.toString()}`);
  }

}

/**
 * Deletes a Firestore document at the specified path.
 *
 * @param {string} path - The Firestore path where the document should be
deleted.
 * @param {string} authToken - The authentication token to authorize the API
request.
 */

function deleteFirestoreDocument(path, authToken) {

  // Validate the input to make sure the path is not empty or just whitespace.
  if (!path || path.trim() === '') {
    Logger.log("Invalid path provided to deleteFirestoreDocument function.");
    return; // Exit the function if the path is invalid.
  }

  // Ensure the Firestore path is correctly formatted without a trailing
slash.
  const trimmedPath = path.endsWith('/') ? path.slice(0, -1) : path;

  // Construct the complete URL for the Firestore API endpoint.
  const url = `https://firestore.googleapis.com/v1/projects/edu-entry-iee-
ihu/databases/(default)/documents/${trimmedPath}`;

  // Set up the options for the DELETE request.
  const options = {
    method: 'DELETE', // Use DELETE method to remove a document.
    headers: {
      'Authorization': `Bearer ${authToken}`, // Use the authToken to
authenticate.
    },
    muteHttpExceptions: true // Prevent throwing exceptions for HTTP errors to
handle them manually.
  };

  // Try to make the HTTP request to Firestore to delete the document.
```

```
  try {
    const response = UrlFetchApp.fetch(url, options);
    // Log the response to confirm the document was deleted.
    Logger.log(`Deleted document at path: ${path}, response:
${response.getContentText()}`);
  } catch (e) {
    // Catch and log any errors that occur during the deletion operation.
    Logger.log(`Exception occurred when trying to delete the document at path:
${path}, error: ${e.toString()}`);
  }

}
```

Layout.js

```
// Formats the entire sheet with specific styles and adjustments to rows and
columns.
function formatSheet(sheet, totalColumns) {

    const maxRows = sheet.getMaxRows();  // Retrieve the highest number of
rows in the sheet.

    // Center align text horizontally and vertically for all cells.
    sheet.getRange(1, 1, maxRows,
totalColumns).setHorizontalAlignment('center').setVerticalAlignment('middle');
;

    // Adjust the width of each column based on its contents and add extra
space for clarity.
    for (let col = 1; col <= totalColumns; col++) {

      sheet.autoResizeColumn(col); // Automatically adjust the column width to
fit the text.
      const currentWidth = sheet.getColumnWidth(col); // Get the current width
of the column.
      const extraSpace = 20; // Specify the extra space to add to each column.
      sheet.setColumnWidth(col, currentWidth + extraSpace); // Apply the new
width to the column.

    }

    // Set a uniform height for all rows in the sheet.
    const preferredRowHeight = 30; // Define a preferred row height.
    sheet.setRowHeights(1, maxRows, preferredRowHeight); // Apply the
preferred height to all rows.
```

```
  }

  // Sets colors for cells based on attendance and grade values.
  function setColorForAttendanceAndGrades(sheet, classDates) {

    const numRows = sheet.getLastRow(); // Get the number of rows with data in
the sheet.
    const dateStartColumnIndex = 5; // Define the column where date data
starts.
    const datesRangeLength = classDates.length; // Get the count of date
columns.
    const attendancesColumnIndex = dateStartColumnIndex + datesRangeLength; //
Identify the 'Attendances' column index.
    const gradeColumnIndex = attendancesColumnIndex + 1; // Identify the
'Grade' column index.

    // Highlight the header row across all columns with a specific color.
    sheet.getRange(1, 1, 1, gradeColumnIndex).setBackground('#FFD966'); // Use
a yellow color for the header row.

    // Iterate over the rows and columns to set background colors based on
cell values.
    for (let row = 2; row <= numRows; row++) {

      // Determine if the row is even or odd
      const isEvenRow = row % 2 === 0; // Check if the current row is even.

      if (isEvenRow) {

        for (let col = 1; col < dateStartColumnIndex; col++) {

          const cell = sheet.getRange(row, col);
          cell.setBackground('#FFE599');

        }

        // Loop over the range of date columns to set the background colors
        for (let col = dateStartColumnIndex; col < attendancesColumnIndex;
col++) {

          const cell = sheet.getRange(row, col);
          const cellValue = cell.getDisplayValue();
          cell.setBackground(getColorBasedOnAttendance(cellValue));

        }

        // Set the background color for the 'Attendances' cell
```

```
        const attendancesCell = sheet.getRange(row, attendancesColumnIndex);
        const attendancesValue = attendancesCell.getDisplayValue();

attendancesCell.setBackground(getColorBasedOnAttendanceSummary(attendancesValu
e));

      } else {

        for (let col = 1; col < dateStartColumnIndex; col++) {

          const cell = sheet.getRange(row, col);
          cell.setBackground('#FFF2CC');

        }

        for (let col = dateStartColumnIndex; col < attendancesColumnIndex;
col++) {

          // Loop over the range of date columns to set the background colors
on grades
          const cell = sheet.getRange(row, col);
          const cellValue = cell.getDisplayValue();
          cell.setBackground(getColorBasedOnGrade(cellValue));
        }

          // Set the background color for the 'Grade' column
          const gradeCell = sheet.getRange(row, gradeColumnIndex);
          const gradeValue = gradeCell.getDisplayValue();
          gradeCell.setBackground(getColorBasedOnMeanGrade(gradeValue));

      }
    }
  }

  // The getColorBasedOnAttendance function returns a color string based on
attendance marks ('X' for absent, '✓' for present).
  function getColorBasedOnAttendance(value) {
    if (value.includes('X')) {
      return '#EA9999'; // red for 'X'
    } else if (value.includes('✓')) {
      return '#B6D7A8'; // green for '✓'
    }
    return null; // no background color
  }

  // The getColorBasedOnAttendanceSummary function provides a color based on
the summary of attendance (e.g., 'Failed', 'OK').
  function getColorBasedOnAttendanceSummary(value) {
```

```
    if (value.includes('Failed')) {
      return '#E06666'; // red for 'Failed'
    } else if (value.includes('OK')) {
      return '#93C47D'; // green for 'OK'
    }
    return null; // no background color
  }

  // The getColorBasedOnGrade function assigns a color for cells that need
attention (e.g., 'Fill Out') or are empty ('-').
  function getColorBasedOnGrade(value) {
    if (value === ('Fill Out')) {
      return '#9FC5E8';
    } else if (value === ('-')) {
      return '#F4CCCC';
    }
    return '#D9EAD3'; // green for 'Grade'
  }

  // The getColorBasedOnMeanGrade function returns a color based on the
numeric value of the grade (e.g., red for grades below 5).
  function getColorBasedOnMeanGrade(value) {
    const grade = parseFloat(value);
    if (!isNaN(grade)) {
      return grade < 5 ? '#E06666' : '#93C47D'; // red for grade < 5, green
for grade >= 5
    }
    return null; // no background color
  }
```

MenuSetup.js

```
/**
 * This function runs automatically when the Google Sheets document is opened.
 */

function onOpen() {

    // Get the user interface instance for the active spreadsheet.
    const ui = SpreadsheetApp.getUi();

    // Create a custom menu with the title '🧮 Attendance Manager'.
    const menu = ui.createMenu('🧮 Attendance Manager')
```

```
        .addItem('🔁 Sync from Firestore  -->  [Select Class]',
'selectClassToSync')
        .addItem('🔁 Sync from Firestore  -->  [All Classes]',
'syncAllClasses')
        .addSeparator()
        .addItem('🔼 Update "Students" to Firestore  -->  [Select Class]',
'selectClassToUpdate')
        .addItem('⏫ Update "Students" to Firestore  -->  [All Classes]',
'updateAllClassesToFirestore')
        .addToUi();
  }
```

SheetToFirebase.js

```
/**
 * Prompts the user to select a class to update in Firestore and stores the
selection.
 * Triggers the update process for the selected class.
 */
function selectClassToUpdate() {
    const ui = SpreadsheetApp.getUi();
    const response = ui.prompt('Enter the name of the class to update in Fire-
store:');

    if (response.getSelectedButton() == ui.Button.OK && re-
sponse.getResponseText()) {
      // Store the classId (sheetName) in Script Properties
      PropertiesService.getScriptProperties().setProperty('classToUpdate', re-
sponse.getResponseText().trim());
      // Call the update function
      updateClassInFirestore();
    }
  }

  /**
   * Updates data in Firestore for a specific class.
   */
  function updateClassInFirestore() {
    // Retrieve the classId (sheetName) from Script Properties
    const classToUpdate = PropertiesService.getScriptProperties().getProp-
erty('classToUpdate');
```

```
    if (classToUpdate) {
      updateFirestoreBasedOnSheet(classToUpdate); // Function that updates
Firestore with sheet data
      // Optionally clear the property after use
      PropertiesService.getScriptProperties().deleteProperty('classToUpdate');
    } else {
      SpreadsheetApp.getUi().alert('Class name not found.');
    }
  }

  /**
   * Triggers the update process for all classes in the Google Sheets to
Firestore.
   */
  function updateAllClassesToFirestore() {
    const ss = SpreadsheetApp.getActiveSpreadsheet();
    ss.getSheets().forEach(sheet => {
      // Call the update function for each sheet
      updateFirestoreBasedOnSheet(sheet.getName()); // Pass each sheet name to
the function
    });
  }

  function updateFirestoreBasedOnSheet(classId) {
    const authToken = getAuthToken(); // Function to get auth token for
Firestore
    const ss = SpreadsheetApp.getActiveSpreadsheet();
    const sheet = ss.getSheetByName(classId); // Use classId as sheet name
    if (!sheet) {
      Logger.log(`Sheet with name "${classId}" not found.`);
      return;
    }
    const values = sheet.getDataRange().getValues();

    const totalColumns = sheet.getLastColumn();
    const classDates = totalColumns - 6;

    // Retrieve and format the dates from the header row (assuming dates start
at the 5th column)
    const headerRange = sheet.getRange(1, 5, 1, classDates);
    const dateHeaders = headerRange.getValues()[0];

    // Format each date header
    const formattedDates = dateHeaders.map(dateHeader =>
formatDateForDocumentId(dateHeader));

    SpreadsheetApp.getUi().alert(`${classDates}`);
```

```
    // Loop through each student row in the sheet, skipping the header row
    for (let i = 1; i < values.length; i++) {
      const row = values[i];
      const userId = row[0].toString(); // Assuming the UserID is in the first
column
      const lastName = row[1]; // Assuming LastName is in the second column
      const firstName = row[2]; // Assuming FirstName is in the third column
      const email = row[3]; // Assuming Email is in the fourth column

      // Create or update the student document in Firestore
      const studentDocument = {
        fields: {
          classId: { stringValue: classId },
          firstName: { stringValue: firstName },
          lastName: { stringValue: lastName },
          email: { stringValue: email },
          userId: { stringValue: userId.toString() }
        }
      };

      updateOrCreateFirestoreDocument(`Classes/${classId}/Students/${userId}`,
studentDocument, authToken);

      // Loop through each date column, which starts right after email (4th
column)
      for (let j = 0; j < classDates; j++) {
        const dateColumnIndex = 4 + j; // Calculate the index of the date
column
        const attendanceCell = String(row[dateColumnIndex]);
        const gradeCell = values[i + 1] ? values[i + 1][dateColumnIndex] :
null; // Check for the next row existence and fetch the grade

        // Format the date in 'yyyy-MM-dd' format
        const attendanceDate = formattedDates[j];
        // Debug log for date alignment
        Logger.log(`Attendance Date for index ${j}: ${attendanceDate}`);

        const attendanceId = `${userId}_${classId}_${attendanceDate}`;

        if (attendanceCell && attendanceCell.includes('✓')) {
          const time = attendanceCell.match(/\(([^)]+)\)/); // Extract time
from parentheses
          const grade = gradeCell !== null ? gradeCell : 'Fill Out'; // Use
'No grade' or similar text if the next row doesn't exist or the grade cell is
empty

          const attendanceDocument = {
            fields: {
```

```
              classId: { stringValue: classId },
              userId: { stringValue: userId },
              date: { stringValue: attendanceDate },
              time: { stringValue: time ? time[1] : '' },
              grade: { stringValue: grade.toString() }
            }
          }

updateOrCreateFirestoreDocument(`Classes/${classId}/Students/${userId}/Attenda
nces/${attendanceId}`, attendanceDocument, authToken);
        } else if (attendanceCell.includes('X')) {

deleteFirestoreDocument(`Classes/${classId}/Students/${userId}/Attendances/${a
ttendanceId}`, authToken);
        }
      }
    }
  }

  function formatDateForDocumentId(date) {
    // Ensure the date is in the correct format, even if the input is already
a Date object
    const dateObj = (date instanceof Date) ? date : new Date(date);
    return Utilities.formatDate(dateObj, Session.getScriptTimeZone(), 'yyyy-
MM-dd');
  }
```

ΠΑΡΑΡΤΗΜΑ Β : Raspberry Pi Code

credentials.py

```
# credentials.py

# Service account email
CLIENT_EMAIL = ""

# Private key
PRIVATE_KEY = """-----BEGIN PRIVATE KEY----- -----END PRIVATE KEY-----\n"""

# Project ID
PROJECT_ID = ""

# Firebase database URL
FIREBASE_DATABASE_URL = ""
```

firebase_auth.py

```
# firebase_auth.py
import json
import time
import requests
import jwt
from credentials import CLIENT_EMAIL, PRIVATE_KEY, PROJECT_ID

def create_jwt(client_email, private_key, project_id):
    """
    Create a JWT (JSON Web Token) for authenticating with Google's OAuth 2.0
server.
    """
    issued_at_time = int(time.time())
    expiry_time = issued_at_time + 3600  # Token valid for 1 hour

    # JWT Header
    headers = {
        "alg": "RS256",
        "typ": "JWT",
    }
```

firebase_ops.py

```
# firebase_ops.py
import json
import datetime
import re
from urllib import request
from urllib.parse import quote_plus
from firebase_auth import get_access_token
from credentials import PROJECT_ID, FIREBASE_DATABASE_URL

def sanitize_am(am_value):
    """Remove all characters except for numbers and letters."""
    return re.sub(r'[^a-zA-Z0-9]', '', am_value)

def create_schedule_to_firestore(document_data, class_id, schedule_id,
access_token):

    # URL-encode the components that will be inserted into the URL to ensure
they are valid
    class_id_encoded = quote_plus(class_id)
    schedule_id_encoded = quote_plus(schedule_id)

    # Construct the Firestore document path
    document_path =
f'Classes/{class_id_encoded}/Schedule/{schedule_id_encoded}'

    # Construct the Firestore REST API endpoint URL
    endpoint = f'{FIREBASE_DATABASE_URL}/{document_path}'

    # Headers including the Authorization token
    headers = {
        'Authorization': f'Bearer {access_token}',
        'Content-Type': 'application/json'
    }

    # Firestore expects the data in the "fields" key
    encoded_data = json.dumps({"fields": document_data}).encode()
    req = request.Request(endpoint, data=encoded_data, headers=headers,
method='PATCH')

    try:
        response = request.urlopen(req)
        print("Successfully uploaded:", response.read().decode())
    except request.HTTPError as e:  # More specific exception
        print(f"HTTP error occurred while uploading to Firestore: {e.code} -
{e.reason}")
    except Exception as e:
        print(f"An error occurred while uploading to Firestore: {e}")
```

```
def change_class_status_to_Firestore(document_data, class_id, access_token):

    # URL-encode the components that will be inserted into the URL to ensure
they are valid
    class_id_encoded = quote_plus(class_id)

    # Construct the Firestore document path
    document_path = f'Classes/{class_id_encoded}'

    # Construct the Firestore REST API endpoint URL
    fields_to_update = "classStatus"
    endpoint =
f'{FIREBASE_DATABASE_URL}/{document_path}?updateMask.fieldPaths={fields_to_upd
ate}'

    # Headers including the Authorization token
    headers = {
        'Authorization': f'Bearer {access_token}',
        'Content-Type': 'application/json'
    }

    # Firestore expects the data in the "fields" key
    encoded_data = json.dumps({"fields": document_data}).encode()
    req = request.Request(endpoint, data=encoded_data, headers=headers,
method='PATCH')

    try:
        response = request.urlopen(req)
        print("Successfully uploaded:", response.read().decode())
    except request.HTTPError as e:  # More specific exception
        print(f"HTTP error occurred while uploading to Firestore: {e.code} -
{e.reason}")
    except Exception as e:
        print(f"An error occurred while uploading to Firestore: {e}")

def create_attendance_to_firestore(document_data, class_id, user_id,
attendance_id, access_token):

    # URL-encode the components that will be inserted into the URL to ensure
they are valid
    class_id_encoded = quote_plus(class_id)
    user_id_encoded = quote_plus(user_id)
    attendance_id_encoded = quote_plus(attendance_id)

    # Construct the Firestore document path
```

```
    document_path =
f'Classes/{class_id_encoded}/Students/{user_id_encoded}/Attendances/{attendanc
e_id_encoded}'

    # Construct the Firestore REST API endpoint URL
    endpoint = f'{FIREBASE_DATABASE_URL}/{document_path}'

    # Headers including the Authorization token
    headers = {
        'Authorization': f'Bearer {access_token}',
        'Content-Type': 'application/json'
    }

    # Firestore expects the data in the "fields" key
    encoded_data = json.dumps({"fields": document_data}).encode()
    req = request.Request(endpoint, data=encoded_data, headers=headers,
method='PATCH')

    try:
        response = request.urlopen(req)
        print("Successfully uploaded:", response.read().decode())
    except request.HTTPError as e:  # More specific exception
        print(f"HTTP error occurred while uploading to Firestore: {e.code} -
{e.reason}")
    except Exception as e:
        print(f"An error occurred while uploading to Firestore: {e}")

def create_student_on_firestore(document_data, class_id, user_id,
access_token):

    # URL-encode the components that will be inserted into the URL to ensure
they are valid
    class_id_encoded = quote_plus(class_id)
    user_id_encoded = quote_plus(user_id)

    # Construct the Firestore document path
    document_path = f'Classes/{class_id_encoded}/Students/{user_id_encoded}'

    # Construct the Firestore REST API endpoint URL
    endpoint = f'{FIREBASE_DATABASE_URL}/{document_path}'

    # Headers including the Authorization token
    headers = {
        'Authorization': f'Bearer {access_token}',
        'Content-Type': 'application/json'
    }

    # Firestore expects the data in the "fields" key
```

```
    encoded_data = json.dumps({"fields": document_data}).encode()
    req = request.Request(endpoint, data=encoded_data, headers=headers,
method='PATCH')

    try:
        response = request.urlopen(req)
        print("Successfully uploaded:", response.read().decode())
    except request.HTTPError as e:  # More specific exception
        print(f"HTTP error occurred while uploading to Firestore: {e.code} -
{e.reason}")
    except Exception as e:
        print(f"An error occurred while uploading to Firestore: {e}")

def fetch_latest_schedule_from_firestore(class_id, access_token):
    """Fetch the latest schedule for the specified class."""
    class_id_encoded = quote_plus(class_id)
    schedule_path = f'Classes/{class_id_encoded}/Schedule'
    endpoint = f'{FIREBASE_DATABASE_URL}/{schedule_path}'
    headers = {'Authorization': f'Bearer {access_token}'}

    req = request.Request(endpoint, headers=headers, method='GET')
    try:
        response = request.urlopen(req)
        schedules_json = json.loads(response.read().decode())
        schedules = [
            {
                'date': doc['fields']['date']['stringValue'],
                'time': doc['fields']['time']['stringValue']
            }
            for doc in schedules_json.get('documents', [])
        ]

        # Sort schedules by date and time
        sorted_schedules = sorted(
            schedules,
            key=lambda d: (d['date'], d['time']),
            reverse=True  # Latest first
        )
        return sorted_schedules
    except Exception as e:
        print(f"Error fetching schedule: {e}")
        return []

def fetch_class_entry_time_from_firestore(class_id, access_token):
    """Fetch the entry time for the specified class."""
    class_id_encoded = quote_plus(class_id)
    class_path = f'Classes/{class_id_encoded}'
    endpoint = f'{FIREBASE_DATABASE_URL}/{class_path}'
```

```
    headers = {'Authorization': f'Bearer {access_token}'}

    req = request.Request(endpoint, headers=headers, method='GET')
    try:
        response = request.urlopen(req)
        class_info = json.loads(response.read().decode())
        entry_time = class_info.get('fields', {}).get('entryTime',
{}).get('stringValue', '00:30')  # Default to 30 minutes
        print("Class info fetched:", class_info)
        return entry_time
    except Exception as e:
        print(f"Error fetching class entry time: {e}")
        return '00:30'  # Default to 30 minutes in case of an error

def fetch_class_status_from_firestore(class_id, access_token):
    """Fetch the entry time for the specified class."""
    class_id_encoded = quote_plus(class_id)
    class_path = f'Classes/{class_id_encoded}'
    endpoint = f'{FIREBASE_DATABASE_URL}/{class_path}'
    headers = {'Authorization': f'Bearer {access_token}'}

    req = request.Request(endpoint, headers=headers, method='GET')
    try:
        # Perform the request and decode the response
        response = request.urlopen(req)
        class_info = json.loads(response.read().decode())

        # Attempt to extract the classStatus value
        class_status = class_info.get('fields', {}).get('classStatus',
{}).get('stringValue', '')

        # Check if the classStatus is 'init' and return True if so
        if class_status == 'init':
            return True
        else:
            return False
    except Exception as e:
        print(f"Error fetching class status: {e}")
        return False

def is_within_schedule_time(current_date, current_time, schedules,
class_entry_time):
    if not schedules:
        print("No schedules available.")
        return None

    # Convert current date and time strings into datetime objects
```

```
    current_date_obj = datetime.datetime.strptime(current_date, '%Y-%m-
%d').date()
    current_time_obj = datetime.datetime.strptime(current_time,
'%H:%M').time()

    # Parse the entry time from string to a timedelta object
    hours, minutes = map(int, class_entry_time.split(":"))
    entry_time_delta = datetime.timedelta(hours=hours, minutes=minutes)

    # Loop through all the schedules
    for schedule in schedules:
        # Parse the schedule date and start time into datetime objects
        schedule_date_obj = datetime.datetime.strptime(schedule['date'], '%Y-
%m-%d').date()
        schedule_time_start_obj = datetime.datetime.strptime(schedule['time'],
'%H:%M').time()

        # Calculate the end time by adding the entry time delta to the start
time
        schedule_time_end_obj = (datetime.datetime.combine(datetime.date.min,
schedule_time_start_obj) + entry_time_delta).time()

        print(f"Checking schedule: {schedule['date']} {schedule['time']}
against current time {current_date} {current_time}")
        print(f"Schedule time window: {schedule_time_start_obj} -
{schedule_time_end_obj}")

        # Check if current date and time are within the schedule's time window
        if current_date_obj == schedule_date_obj and schedule_time_start_obj
<= current_time_obj <= schedule_time_end_obj:
            print("Within scheduled time.")
            return schedule  # Return the matching schedule document

    print("Not within scheduled time.")
    return None  # No schedule matched or not within time window

def student_exists_on_class(user_id, class_id, access_token):
    """Check if a student exists in the specified class."""
    # URL-encode the class_id to ensure it's valid in the URL
    class_id_encoded = quote_plus(class_id)
    user_id_encoded = quote_plus(user_id)
    # Construct the path to query all students in the class
    students_path = f'Classes/{class_id_encoded}/Students'
    # Construct the Firestore REST API endpoint URL
    endpoint = f'{FIREBASE_DATABASE_URL}/{students_path}'
    # Headers including the Authorization token
    headers = {'Authorization': f'Bearer {access_token}'}
```

```
    # Make the request to Firestore to get all students in the class
    req = request.Request(endpoint, headers=headers, method='GET')
    try:
        response = request.urlopen(req)
        students_data = json.loads(response.read().decode())

        # Check if the user_id matches any document ID in the class
        for document in students_data.get('documents', []):
            # Extract the document ID and compare it with the user_id
            document_id = document['name'].split('/')[-1]
            if document_id == user_id_encoded:
                return True  # Student exists in the class

        return False  # No matching student found in the class
    except Exception as e:
        print(f"Error checking if student exists in class: {e}")
        return False  # Default to False in case of an error
```

qrCodeScanner.py

```
# qrCodeScanner.py

from firebase_auth import get_access_token
from firebase_ops import *
import os
import time
from pyzbar.pyzbar import decode
from PIL import Image
import datetime
import json
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)

red = 17
yellow = 27
green = 22
buzzer = 23
```

```
GPIO.setup(red, GPIO.OUT)
GPIO.setup(yellow, GPIO.OUT)
GPIO.setup(green, GPIO.OUT)

GPIO.output(red, GPIO.LOW)
GPIO.output(yellow, GPIO.LOW)
GPIO.output(green, GPIO.LOW)

# Function to capture an image
def capture_image(image_path='image.jpg'):
    os.system('libcamera-still --nopreview --timeout 100 -o ' + image_path + '
> /dev/null 2>&1')

# Function to decode QR codes from an image
def decode_qr(image_path):
    data = decode(Image.open(image_path))
    return [item.data.decode("utf-8") for item in data]

def setup_buzzer(pin = buzzer):
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(pin, GPIO.OUT)
    return GPIO.PWM(pin, 1)  # Initialize PWM on the pin at 1 Hz

def confirmation_sound():
    buzzer = setup_buzzer(23)
    try:
        # First tone (Lower frequency)
        frequency1 = 523  # Frequency in Hz (C5 note)
        buzzer.start(30)  # 50% duty cycle
        buzzer.ChangeFrequency(frequency1)  # Set frequency
        time.sleep(0.1)  # Beep duration

        # Short pause between tones
        buzzer.stop()
        time.sleep(0.05)

        # Second tone (Higher frequency)
        frequency2 = 659  # Frequency in Hz (E5 note)
        buzzer.start(50)
        buzzer.ChangeFrequency(frequency2)
        time.sleep(0.1)  # Beep duration
    finally:
        buzzer.stop()  # Stop the buzzer
        GPIO.cleanup(23)  # Clean up GPIO

def error_sound():
    buzzer = setup_buzzer()
```

```
    frequency = 440  # Frequency in Hz (A4 note)
    buzzer.start(20)
    buzzer.ChangeFrequency(frequency)  # Set frequency
    time.sleep(0.1)  # Beep duration
    time.sleep(0.05)  # Pause between beeps
    buzzer.ChangeFrequency(frequency)  # Same tone for error
    time.sleep(0.1)  # Repeat beep
    buzzer.stop()
    GPIO.cleanup(23)

# class_id = 'LAB B3'

try:
    while True:

        GPIO.output(green, GPIO.HIGH)

        # Capture an image
        image_path = '/dev/shm/image.jpg'
        capture_image(image_path)

        # Decode QR codes
        qr_contents = decode_qr(image_path)

        # Print the decoded QR contents
        print(qr_contents)

        # Check if any QR content was detected
        if qr_contents:

            confirmation_sound()
            GPIO.output(green, GPIO.LOW)
            GPIO.output(yellow, GPIO.HIGH)

            for content in qr_contents:

                try:
                    qr_data = json.loads(content)

                    # Obtain an access token
                    access_token = get_access_token()

                    # Process Student QR code data
                    if qr_data.get('type') == 'student':

                        user_id = sanitize_am(qr_data['userId'])
                        class_id = qr_data['classId']
                        first_name = qr_data['firstName']
```

```
                        last_name = qr_data['lastName']
                        email = qr_data['email']

                        # Fetch the latest schedule for the class
                        schedules =
fetch_latest_schedule_from_firestore(class_id, access_token)

                        # Fetch the entry time for the class
                        class_entry_time =
fetch_class_entry_time_from_firestore(class_id, access_token)

                        # Get the current date from the Raspberry Pi
                        current_attendance_date =
datetime.datetime.now().strftime('%Y-%m-%d')

                        # Get the current time from the Raspberry Pi
                        current_attendance_time =
datetime.datetime.now().strftime('%H:%M')

                        # Construct attendanceId
                        attendance_id =
f"{user_id}_{class_id}_{current_attendance_date}"

                        # Construct student document data
                        student_document_data = {
                                "userId": {"stringValue": user_id},
                                "firstName": {"stringValue": first_name},
                                "lastName": {"stringValue": last_name},
                                "email": {"stringValue": email}
                            }
                        # Construct attendance document data
                        attendance_document_data = {
                                "classId": {"stringValue": class_id},
                                "userId": {"stringValue": user_id},
                                "date": {"stringValue":
current_attendance_date},
                                "time": {"stringValue":
current_attendance_time},
                                "grade": {"stringValue": "Fill Out"}
                            }

                        if (fetch_class_status_from_firestore(class_id,
access_token)):

                            # Upload to Firestore
                            print(f"Creating Student to Firestore:
{student_document_data}")
```

```
                            create_student_on_firestore(student_document_data,
class_id, user_id, access_token)
                            time.sleep(2)

                        if(student_exists_on_class(user_id, class_id,
access_token) and is_within_schedule_time(current_attendance_date,
current_attendance_time, schedules, class_entry_time)):

                            # Upload to Firestore
                            print(f"Uploading Attendance to Firestore:
{attendance_document_data}")

create_attendance_to_firestore(attendance_document_data, class_id, user_id,
attendance_id, access_token)
                            time.sleep(2)

                        else:
                            GPIO.output(yellow, GPIO.LOW)
                            GPIO.output(red, GPIO.HIGH)
                            error_sound()

                    # Process teacher QR code data
                    elif qr_data.get('type') == 'teacher':

                        class_id = qr_data['classId']
                        class_status = qr_data['classStatus']

                        if(class_status != 'init'):

                            status_document_data = {
                                "classStatus": {"stringValue": 'started'}
                            }


change_class_status_to_Firestore(status_document_data, class_id, access_token)

                        # Get the current date from the Raspberry Pi
                        current_schedule_date =
datetime.datetime.now().strftime('%Y-%m-%d')

                        # Get the current time from the Raspberry Pi
                        current_schedule_time =
datetime.datetime.now().strftime('%H:%M')

                        # Construct attendanceId
                        schedule_id = f"{current_schedule_date}"

                        # Construct attendance document data
```

```
                        schedule_document_data = {
                                "date": {"stringValue":
current_schedule_date},
                                "time": {"stringValue": current_schedule_time}
                            }

                        # Upload to Firestore
                        print(f"Uploading Schedule to Firestore:
{schedule_document_data}")
                        create_schedule_to_firestore(schedule_document_data,
class_id, schedule_id, access_token)
                        time.sleep(3)

                    else:
                        print("Unknown QR code type.")
                        GPIO.output(yellow, GPIO.LOW)
                        GPIO.output(red, GPIO.HIGH)
                        error_sound()

                except json.JSONDecodeError as e:
                    print(f"Error decoding QR code: {e}")

            time.sleep(3)  # Sleep to throttle the upload frequency
        else:
            print("No QR code found, capturing next image immediately...")

        GPIO.output(yellow, GPIO.LOW)
        GPIO.output(red, GPIO.LOW)

        os.remove(image_path)  # Remove the image file after scanning

except KeyboardInterrupt:
    print("Script interrupted by user.")

finally:
    GPIO.cleanup()  # Ensure GPIO resources are freed properly
    print("GPIO cleanup complete.")
```

ΠΑΡΑΡΤΗΜΑ C Android App Code

UserProfile.kt

```
package com.sb3rk0s.eduentry.data.models

data class UserProfile(
    val am: String,
    val regyear: String,
    val regsem: String,
    val sem: String,
    val givenName: String,
    val sn: String,
    val fathersname: String,
    val eduPersonAffiliation: String,
    val eduPersonPrimaryAffiliation: String,
    val title: String,
    val cn: String,
    val secondarymail: String,
    val telephoneNumber: String,
    val labeledURI: String,
    val id: String,
    val mail: String,
    val pwdChangedTime: String,
    val socialMedia: SocialMedia,
    val profilePhoto: String
)

data class SocialMedia(
    val socialMediaExtra: List<Any>
)
```

AuthService

```
package com.sb3rk0s.eduentry.data.network

import retrofit2.Response
import retrofit2.http.Field
import retrofit2.http.FormUrlEncoded
import retrofit2.http.POST

interface AuthService {
    @FormUrlEncoded
    @POST("token")
    suspend fun authenticateUser(
        @Field("client_id") clientId: String,
        @Field("client_secret") clientSecret: String,
        @Field("grant_type") grantType: String,
        @Field("code") code: String
    ): Response<String>
}
```

UserProfileService

```
package com.sb3rk0s.eduentry.data.network

import com.sb3rk0s.eduentry.data.models.UserProfile
import retrofit2.Response
import retrofit2.http.GET
import retrofit2.http.Header

interface UserProfileService {
    @GET("profile")
    suspend fun getUserProfile(
        @Header("x-access-token") accessToken: String,
        @Header("Content-Type") contentType: String,
    ): Response<UserProfile>
}
```

ClassSelectionDialog.kt

```
package com.sb3rk0s.eduentry.presentation.ui.composables

import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.AlertDialog
import androidx.compose.material3.Button
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.Text
import androidx.compose.material3.TextButton
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp

@Composable
fun ClassSelectionDialog(
    classIds: List<String>,
    onDismissRequest: () -> Unit,
    onClassSelected: (String) -> Unit
) {
    AlertDialog(
        onDismissRequest = onDismissRequest,
        title = { Text("Select a Class") },
        text = {
            Column {
                classIds.forEach { classId ->
                    TextButton(
                        onClick = { onClassSelected(classId) },
                        modifier = Modifier.fillMaxWidth()
                    ) {
                        Text(text = classId,
                            modifier = Modifier.padding(8.dp),
                            fontSize = 22.sp,
                            color = Color(0xFF009688),
                            fontWeight = FontWeight.Medium)
                    }
                }
            }
        },
```

```
        confirmButton = {
            Button(
                onClick = onDismissRequest,
                shape = RoundedCornerShape(12.dp),
                colors = ButtonDefaults.buttonColors(
                    containerColor = Color(0xFF009688),
                    contentColor = Color.White
                ),
                content = { Text("Cancel") }
            )
        }
    )
}
```

QrCodeScreen.kt

```
package com.sb3rk0s.eduentry.presentation.ui.composables

import androidx.compose.foundation.Image
import androidx.compose.foundation.layout.Arrangement
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.aspectRatio
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.ButtonDefaults
import androidx.compose.material3.CircularProgressIndicator
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.OutlinedButton
import androidx.compose.material3.Scaffold
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.graphics.asImageBitmap
import androidx.compose.ui.platform.LocalConfiguration
import androidx.compose.ui.platform.LocalContext
```

```
import androidx.compose.ui.platform.LocalDensity
import androidx.compose.ui.text.SpanStyle
import androidx.compose.ui.text.buildAnnotatedString
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.text.withStyle
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.sb3rk0s.eduentry.presentation.viewmodel.AuthViewModel
import com.sb3rk0s.eduentry.utility.AESUtils
import com.sb3rk0s.eduentry.utility.AESUtils.hexStringToByteArray
import com.sb3rk0s.eduentry.utility.filterDigits
import com.sb3rk0s.eduentry.utility.generateQRCode
import org.json.JSONObject


@Composable
fun QrCodeScreen(authViewModel: AuthViewModel) {
    val userProfile by authViewModel.userProfileData
    val isLoading by authViewModel.isLoading
    val isUserLoggedIn by authViewModel.isUserLoggedIn
    val density = LocalDensity.current
    val screenWidthDp = LocalConfiguration.current.screenWidthDp.dp
    val screenWidthPx = with(density) { screenWidthDp.roundToPx() }
    val showDialog = remember { mutableStateOf(false) }
    val selectedClassId = remember { mutableStateOf<String?>(null) }
    val key =
"f78fc226618e78880e436b9b40fcce2a8b9a3154aeea66d897898959132ef39f".hexStrin
gToByteArray()
    val iv = "dd59dee819aadb704afeb8e03ff7c9bb".hexStringToByteArray()

    Scaffold(
        topBar = {
            EduEntryTopBar(
                title = "Edu Entry",
                context = LocalContext.current,
                isUserLoggedIn = isUserLoggedIn,
                onLogoutClick = {
                    authViewModel.clearUserProfile()
                }
            )
        }
    ) { paddingValues ->
        Box(
            modifier = Modifier
                .fillMaxSize()
                .padding(paddingValues),
            contentAlignment = Alignment.Center
        ) {
            if (isLoading) {
                CircularProgressIndicator(color = Color(0xFF009688))
            } else {
                userProfile?.let { profile ->
                    Column(
                        modifier = Modifier
                            .fillMaxWidth(),
                        verticalArrangement = Arrangement.Center,
                        horizontalAlignment = Alignment.CenterHorizontally
                    ) {
                        fun String.capitalizeFirstLetter(): String {
                            return this.lowercase().replaceFirstChar {
```

```
it.uppercase() }
                        }

                        val formattedFirstName =
profile.givenName.capitalizeFirstLetter()
                        val formattedLastName =
profile.sn.capitalizeFirstLetter()

                        Text(
                            text = buildAnnotatedString {
                                append("Welcome, ")
                                withStyle(
                                    style = SpanStyle(
                                        fontWeight = FontWeight.Bold,
                                        color = Color(0xFF009688)
                                    )
                                ) {
                                    append("$formattedFirstName
$formattedLastName")
                                }
                            },
                            style =
MaterialTheme.typography.headlineMedium,
                            modifier = Modifier
                                .padding(
                                    bottom = 32.dp,
                                    top = 32.dp
                                ),
                            textAlign = TextAlign.Center
                        )
                        Text(
                            "Academic ID: ${profile.am.filterDigits()}",
                            style = MaterialTheme.typography.titleLarge,
                            modifier = Modifier.padding(bottom = 32.dp),
                            textAlign = TextAlign.Center
                        )

                        OutlinedButton(
                            onClick = { showDialog.value = true },
                            shape = RoundedCornerShape(12.dp),
                            colors = ButtonDefaults.buttonColors(
                                containerColor = Color(0xFF009688),
                                contentColor = Color.White
                            ),
                            contentPadding = PaddingValues(start = 20.dp,
top = 12.dp, end = 20.dp, bottom = 12.dp),
                            modifier = Modifier.padding(8.dp)
                        ) {
                            Text(
                                text = selectedClassId.value ?: "Select a
Class",
                                fontSize = 22.sp,
                                fontWeight = FontWeight.Medium
                            )
                        }

                        if (showDialog.value) {
                            ClassSelectionDialog(
                                classIds = authViewModel.classIds.value,
                                onDismissRequest = { showDialog.value =
false },
```

```
                                onClassSelected = { classId ->
                                    selectedClassId.value = classId
                                    showDialog.value = false
                                }
                            )
                        }

                        Spacer(Modifier.weight(1f))

                        selectedClassId.value?.let {
                            val jsonContent = JSONObject().apply {
                                put("type", profile.eduPersonAffiliation)
                                put("firstName", formattedFirstName)
                                put("lastName", formattedLastName)
                                put("userId", profile.am.filterDigits())
                                put("email", profile.mail)
                                put("classId", selectedClassId.value)
                            }.toString()

                            val encryptedJsonContent =
AESUtils.encrypt(key, iv, jsonContent)
                            val qrCodeBitmap =
generateQRCode(encryptedJsonContent, screenWidthPx)
                            Image(
                                bitmap = qrCodeBitmap.asImageBitmap(),
                                contentDescription = "User QR Code",
                                modifier = Modifier
                                    .fillMaxWidth()
                                    .aspectRatio(1f)
                            )
                        }
                        Spacer(Modifier.weight(1f))
                    }
                } ?: run {
                    Column(
                        modifier = Modifier
                            .fillMaxWidth()
                            .padding(24.dp),
                        verticalArrangement = Arrangement.Center,
                        horizontalAlignment = Alignment.CenterHorizontally
                    ) {
                        Text(
                            text = buildAnnotatedString {
                                append("Welcome to ")
                                withStyle(style = SpanStyle(color =
Color(0xFF009688))) {
                                    append("Edu Entry")
                                }
                                append(" !")
                            },
                            style = MaterialTheme.typography.headlineLarge,
                            textAlign = TextAlign.Center,
                            modifier = Modifier
                                .padding(
                                    bottom = 32.dp,
                                    top = 32.dp
                                )
                        )
                        Text(
                            text = "Edu Entry is designed to authenticate
student attendance seamlessly. By using a QR code scanned at a Raspberry Pi
```

```
station in class, your attendance is instantly recorded.",
                                style = MaterialTheme.typography.bodyLarge,
                                textAlign = TextAlign.Center,
                                modifier = Modifier.padding(bottom = 8.dp)
                            )
                            Text(
                                text = "Once you login and your profile
information is loaded, select a class from the available classes and a
personalized QR code will appear here. Ensure your device is connected to
the internet and wait for your profile data to load. If your profile is not
loading try to clear cookies from your phone's browser.",
                                style = MaterialTheme.typography.bodyLarge,
                                textAlign = TextAlign.Center
                            )
                            Spacer(Modifier.weight(1f))
                            Text(
                                text = buildAnnotatedString {
                                    append("After Logging in ")
                                    withStyle(style = SpanStyle(color =
Color.Red)) {
                                        append("PRESS")
                                    }
                                    append(" return to external application to
fetch your Profile !")
                                },
                                style =
MaterialTheme.typography.headlineMedium,
                                textAlign = TextAlign.Center,
                            )
                        }
                    }
                }
            }
        }
    }
}
```

TopBar.kt

```
package com.sb3rk0s.eduentry.presentation.ui.composables

import android.content.Context
import android.content.Intent
import android.net.Uri
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.PaddingValues
import androidx.compose.foundation.layout.fillMaxWidth

import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.*
import androidx.compose.runtime.Composable
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
```

```
@OptIn(ExperimentalMaterial3Api::class)
@Composable
fun EduEntryTopBar(
    title: String,
    context: Context,
    isUserLoggedIn: Boolean,
    onLogoutClick: () -> Unit
) {
    TopAppBar(
        title = { Text(text = title, color = Color.White) },
        actions = {
            if (isUserLoggedIn) {
                LogoutButton(onLogoutClick)
            } else {
                OAuthButton {
                    val oAuthURL =
"https://login.iee.ihu.gr/authorization/?client_id=65eb85b73ae68103fd014fc8
&response_type=code&scope=profile&redirect_uri=com.sb3rk0s.eduentry://oauth
2redirect"
                    context.startActivity(Intent(Intent.ACTION_VIEW,
Uri.parse(oAuthURL)))
                }
            }
        },
        colors = TopAppBarDefaults.topAppBarColors(containerColor =
Color(0xFF009688))
    )
}

@Composable
fun LogoutButton(onClick: () -> Unit) {
    OutlinedButton(
        onClick = onClick,
        shape = RoundedCornerShape(12.dp),
        colors = ButtonDefaults.buttonColors(
            containerColor = Color.White,
            contentColor = Color(0xFF009688)
        ),
        contentPadding = PaddingValues(start = 20.dp, top = 12.dp, end =
20.dp, bottom = 12.dp),
        modifier = Modifier.padding(end = 16.dp)
    ) {
        Text("Log Out", fontSize = 18.sp, fontWeight = FontWeight.Medium)
    }
}

@Composable
fun OAuthButton(onClick: () -> Unit) {
    OutlinedButton(
        onClick = onClick,
        shape = RoundedCornerShape(12.dp),
        colors = ButtonDefaults.buttonColors(
            containerColor = Color.White,
            contentColor = Color(0xFF009688)
        ),
        contentPadding = PaddingValues(start = 20.dp, top = 12.dp, end =
20.dp, bottom = 12.dp),
        modifier = Modifier.padding(end = 16.dp)
    ) {
        Text("Login",
            fontSize = 18.sp,
```

```
            fontWeight = FontWeight.Medium)
    }
}
```

MainActivity

```
package com.sb3rk0s.eduentry.presentation.ui

import android.content.Intent
import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.activity.viewModels
import androidx.compose.material3.MaterialTheme
import com.sb3rk0s.eduentry.R
import com.sb3rk0s.eduentry.presentation.ui.composables.QrCodeScreen
import com.sb3rk0s.eduentry.presentation.viewmodel.AuthViewModel
import com.sb3rk0s.eduentry.presentation.viewmodel.AuthViewModelFactory

class MainActivity : ComponentActivity() {
    private val authViewModel: AuthViewModel by viewModels{
        AuthViewModelFactory(getSharedPreferences("YourPreferenceName",
MODE_PRIVATE))
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            MaterialTheme {
                QrCodeScreen(authViewModel)
            }
        }
        handleIntent(intent)
    }

    override fun onNewIntent(intent: Intent?) {
        super.onNewIntent(intent)
        handleIntent(intent)
    }

    private fun handleIntent(intent: Intent?) {
        if (intent?.action == Intent.ACTION_VIEW) {
            val uri = intent.data
            if (uri != null &&
uri.toString().startsWith("com.sb3rk0s.eduentry")) {
                val code = uri.getQueryParameter("code")
                if (code != null) {
                    Log.d("MainActivity", "Authorization code: $code")
                    val clientId = getString(R.string.clientId)
                    val clientSecret = getString(R.string.clientSecret)
                    authViewModel.authenticateUser(code, clientId,
clientSecret)
                }
            }
        }
    }
}
```

AuthViewModel

```
package com.sb3rk0s.eduentry.presentation.viewmodel

import android.content.SharedPreferences
import android.util.Log
import androidx.compose.runtime.mutableStateOf
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.google.firebase.firestore.FirebaseFirestore
import com.google.gson.Gson
import com.sb3rk0s.eduentry.data.models.UserProfile
import com.sb3rk0s.eduentry.data.network.AuthService
import com.sb3rk0s.eduentry.data.network.UserProfileService
import kotlinx.coroutines.launch
import org.json.JSONObject
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import retrofit2.converter.scalars.ScalarsConverterFactory

class AuthViewModel(private val sharedPreferences: SharedPreferences) :
ViewModel() {
    private val authService: AuthService
    private val userProfileService: UserProfileService
    private val gson = Gson()
    private val firestore = FirebaseFirestore.getInstance()

    var classIds = mutableStateOf<List<String>>(listOf())
    var userProfileData = mutableStateOf<UserProfile?>(null)
        private set
    var isLoading = mutableStateOf(false)
        private set
    val isUserLoggedIn = mutableStateOf(false)

    init {
        val authRetrofit = Retrofit.Builder()
            .addConverterFactory(ScalarsConverterFactory.create())
            .addConverterFactory(GsonConverterFactory.create())
            .baseUrl("https://login.iee.ihu.gr/")
            .build()

        authService = authRetrofit.create(AuthService::class.java)

        val userRetrofit = Retrofit.Builder()
            .addConverterFactory(ScalarsConverterFactory.create())
            .addConverterFactory(GsonConverterFactory.create())
            .baseUrl("https://api.it.teithe.gr/")
            .build()

        userProfileService =
userRetrofit.create(UserProfileService::class.java)

        loadUserProfileFromSharedPreferences()
        fetchClasses()
    }


    fun authenticateUser(authorizationCode: String, clientId: String,
clientSecret: String) {
        isLoading.value = true
        viewModelScope.launch {
            try {
                val response = authService.authenticateUser(clientId,
```

```
clientSecret, "authorization_code", authorizationCode)
                if (response.isSuccessful) {
                    val jsonResponse = JSONObject(response.body()!!)
                    val accessToken =
jsonResponse.getString("access_token")
                    getUserProfile(accessToken)
                    Log.d("AuthViewModel", "Authorization code:
$accessToken")
                } else {
                    Log.e("AuthViewModel", "Error during authentication:
HTTP ${response.code()}")
                    isLoading.value = false
                }
            } catch (e: Exception) {
                Log.e("AuthViewModel", "Error during authentication", e)
                isLoading.value = false
            }
        }
    }

    private fun getUserProfile(accessToken: String) {
        viewModelScope.launch {
            try {
                val response =
userProfileService.getUserProfile(accessToken, "application/json")
                if (response.isSuccessful && response.body() != null) {
                    val userProfile: UserProfile = response.body()!!
                    userProfileData.value = userProfile
                    saveUserProfile(userProfile)
                    Log.d("AuthViewModel", "User profile fetched
successfully.")
                } else {
                    Log.e("AuthViewModel", "Error fetching user profile:
HTTP ${response.code()}")
                }
            } catch (e: Exception) {
                Log.e("AuthViewModel", "Error fetching user profile", e)
            } finally {
                isLoading.value = false
            }
        }
    }

    private fun saveUserProfile(userProfile: UserProfile) {
        viewModelScope.launch {
            val userProfileJson = gson.toJson(userProfile)
            sharedPreferences.edit().putString("userProfile",
userProfileJson).apply()
            Log.d("AuthViewModel", "User profile saved successfully.")
            updateLoginState()
        }
    }

    private fun getUserProfileFromSharedPreferences(): UserProfile? {
        val userProfileJson = sharedPreferences.getString("userProfile",
null)
        return if (userProfileJson != null) {
            gson.fromJson(userProfileJson, UserProfile::class.java).also {
                userProfileData.value = it
                Log.d("AuthViewModel", "User profile retrieved
successfully.")
```

```
            }
        } else {
            Log.e("AuthViewModel", "No user profile found in shared
preferences.")
            null
        }
    }

    private fun updateLoginState() {
        isUserLoggedIn.value = getUserProfileFromSharedPreferences() !=
null
    }

    fun clearUserProfile() {
        viewModelScope.launch {
            sharedPreferences.edit().remove("userProfile").apply()
            userProfileData.value = null
            updateLoginState()
            Log.d("AuthViewModel", "User profile cleared successfully.")
        }
    }

    private fun loadUserProfileFromSharedPreferences() {
        userProfileData.value = getUserProfileFromSharedPreferences()
        updateLoginState()
    }

    private fun fetchClasses() {
        firestore.collection("Classes").get()
            .addOnSuccessListener { result ->
                val ids = result.documents.map { it.id }
                classIds.value = ids
            }
            .addOnFailureListener { exception ->
                Log.e("AuthViewModel", "Error fetching class IDs",
exception)
            }
    }
}
```

AuthViewModelFactory

```
package com.sb3rk0s.eduentry.presentation.viewmodel

import android.content.SharedPreferences
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class AuthViewModelFactory(private val sharedPreferences:
SharedPreferences) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(AuthViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
            return AuthViewModel(sharedPreferences) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

AESUtils

```
package com.sb3rk0s.eduentry.utility

import javax.crypto.Cipher
import javax.crypto.spec.SecretKeySpec
import javax.crypto.spec.IvParameterSpec
import android.util.Base64

object AESUtils {
    private const val ALGORITHM = "AES"
    private const val TRANSFORMATION = "AES/CBC/PKCS5Padding"

    fun String.hexStringToByteArray(): ByteArray {
        val len = this.length
        val data = ByteArray(len / 2)
        for (i in 0 until len step 2) {
            data[i / 2] = ((Character.digit(this[i], 16) shl 4) +
Character.digit(this[i + 1], 16)).toByte()
        }
        return data
    }

    fun encrypt(key: ByteArray, ivBytes: ByteArray, dataToEncrypt: String):
String {
        val secretKeySpec = SecretKeySpec(key, ALGORITHM)
        val ivSpec = IvParameterSpec(ivBytes)
        val cipher = Cipher.getInstance(TRANSFORMATION)
        cipher.init(Cipher.ENCRYPT_MODE, secretKeySpec, ivSpec)
        val encrypted =
cipher.doFinal(dataToEncrypt.toByteArray(Charsets.UTF_8))
        return Base64.encodeToString(encrypted, Base64.DEFAULT)
    }

    fun decrypt(key: ByteArray, ivBytes: ByteArray, encryptedData: String):
String {
        val secretKeySpec = SecretKeySpec(key, ALGORITHM)
        val ivSpec = IvParameterSpec(ivBytes)
        val cipher = Cipher.getInstance(TRANSFORMATION)
        cipher.init(Cipher.DECRYPT_MODE, secretKeySpec, ivSpec)
        val decodedData = Base64.decode(encryptedData, Base64.DEFAULT)
        val decrypted = cipher.doFinal(decodedData)
        return String(decrypted, Charsets.UTF_8)
    }
}
```

generateQRCode.kt

```
package com.sb3rk0s.eduentry.utility

import android.graphics.Bitmap
import com.google.zxing.BarcodeFormat
import com.google.zxing.qrcode.QRCodeWriter

fun generateQRCode(content: String, size: Int): Bitmap {
    val writer = QRCodeWriter()
    val bitMatrix = writer.encode(content, BarcodeFormat.QR_CODE, size,
size)
    val bitmap = Bitmap.createBitmap(size, size, Bitmap.Config.RGB_565)
    for (x in 0 until size) {
        for (y in 0 until size) {
            bitmap.setPixel(x, y, if (bitMatrix[x, y])
android.graphics.Color.BLACK else android.graphics.Color.WHITE)
        }
    }
    return bitmap
}

fun String.filterDigits(): String = this.filter { it.isDigit() }
```