ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εφαρμογή Παροχής Ιατρικής Φροντίδας

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

της

ΜΠΑΤΙΛΑ ΔΗΜΗΤΡΑΣ

(ΑΕΜ: 4127)

Επιβλέπων : **Νικόλαος Δημόκας**

Επίκουρος Καθηγητής

Καστοριά Οκτώβριος - 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Εφαρμογή Παροχής Ιατρικής Φροντίδας

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

της

ΜΠΑΤΙΛΑ ΔΗΜΗΤΡΑΣ

(ΑΕΜ: 4127)

Επιβλέπων : **Νικόλαος Δημόκας**
Επίκουρος Καθηγητής

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 03/10/2024

Δημόκας Νικόλαος	Βαρδάκας Ιωάννης	Βέργαδος Δημήτρης
Επίκουρος Καθηγητής	Αναπληρωτής Καθήγητης	Αναπληρωτής Καθήγητης

Καστοριά Οκτώβριος - 2024

Ευχαριστίες

Θα ήθελα να εκφράσω τις ειλικρινείς μου ευχαριστίες στον επιβλέποντα καθηγητή μου, κ. Νικόλαο Δημόκα, για την αμέριστη υποστήριξη και την πολύτιμη καθοδήγησή του κατά τη διάρκεια αυτής της πτυχιακής εργασίας. Η εμπειρία του και οι συμβουλές του υπήρξαν καθοριστικές για την επιτυχή ολοκλήρωση του έργου μου. Ήταν πάντα διαθέσιμος να με βοηθήσει, να συζητήσουμε και να ανταλλάξουμε απόψεις, και έδειξε μεγάλη κατανόηση στις δυσκολίες που αντιμετώπισα κατά τη διάρκεια της εργασίας. Είμαι πραγματικά ευγνώμων που είχα την ευκαιρία να εργαστώ υπό την καθοδήγησή του κατά τη διάρκεια αυτής της εμπειρίας.

Επιπλέον, θα ήθελα να ευχαριστήσω την οικογένειά μου για την αδιάκοπη στήριξη και την ενθάρρυνση που μου προσέφεραν καθ' όλη τη διάρκεια των σπουδών μου. Η στήριξή τους υπήρξε ανεκτίμητη, δίνοντάς μου τη δύναμη να συνεχίσω ακόμα και στις πιο δύσκολες στιγμές.

Ευχαριστώ θερμά τους φίλους και τους συναδέλφους μου, οι οποίοι με ενθάρρυναν και με βοήθησαν με τις πολύτιμες παρατηρήσεις τους καθ' όλη τη διάρκεια αυτής της προσπάθειας. Οι συζητήσεις μαζί τους και οι ιδέες που αντάλλαξα με αυτούς συνέβαλαν σημαντικά στην εξέλιξη της εργασίας μου.

Ιδιαίτερες ευχαριστίες οφείλω σε όλους τους καθηγητές του Πανεπιστημίου Δυτικής Μακεδονίας, των οποίων η διδασκαλία και καθοδήγηση συνέβαλαν στη διαμόρφωση των γνώσεων και των δεξιοτήτων που μου επέτρεψαν να ολοκληρώσω αυτή την πτυχιακή εργασία.

Τέλος, ευχαριστώ όλους εκείνους που με οποιονδήποτε τρόπο συνέβαλαν στην πραγματοποίηση αυτής της εργασίας. Η στήριξή σας υπήρξε πολύτιμη και βοήθησε στην επίτευξη των στόχων μου.

Περίληψη

Η συγκεκριμένη πτυχιακή εργασία στοχεύει στην ανάπτυξη εφαρμογής για κινητά τηλέφωνα που απευθύνεται στη διαχείριση ιατρικής περίθαλψης. Η εφαρμογή επιτρέπει στους χρήστες την καταγραφή και διαχείριση των φαρμάκων τους, επαφών έκτακτης ανάγκης καθώς και υπενθυμίσεων για τη λήψη των φαρμάκων τους.

Βασικά στοιχεία αυτής της εργασίας είναι η χρήση της πλατφόρμας Firebase για την υλοποίηση της ασφάλειας μέσω του Firebase Authentication και την αποθήκευση δεδομένων της Firebase Realtime Database. Η ανάπτυξη της εφαρμογής πραγματοποιήθηκε σε γλώσσα προγραμματισμού Kotlin, η οποία αποτελεί την προτεινόμενη επιλογή για την ανάπτυξη εφαρμογών σε περιβάλλον Android. Ιδιαίτερη έμφαση δόθηκε στην δημιουργία μιας φιλικής προς τον χρήστη εφαρμογής, με εστίαση στην βελτίωση της εμπειρίας χρήστη (UX), ώστε η εφαρμογή να είναι ευχάριστη και εύχρηστη.

Στην παρούσα πτυχιακή εργασία αναλύονται τα στάδια για τη δημιουργία της εφαρμογής, όπως η μελέτη λειτουργικών απαιτήσεων, η αρχιτεκτονική του συστήματος, η υλοποίηση της εφαρμογής και η παρουσίαση των συμπερασμάτων, καθώς και οι μελλοντικές προτάσεις που υπάρχουν για την εφαρμογή.

Λέξεις Κλειδιά: Ιατρική Φροντίδα, Firebase, Realtime Database, Kotlin, Jetpack Compose, Android, Κινητή Εφαρμογή

Abstract

This thesis aims to develop a mobile phone application aimed at medical care management. The application allows users to record and manage their medications, emergency contacts as well as reminders to take their medications.

Key elements of this work are the use of the Firebase platform to implement security through Firebase Authentication and the storage of Firebase Realtime Database data. The development of the application was done in Kotlin programming language, which is the recommended choice for developing applications in Android environment. Special emphasis was placed on creating a user-friendly application, with a focus on improving the user experience (UX) to make the application pleasant and easy to use.

In this thesis, the stages for the creation of the application are discussed, such as the functional requirements study, system architecture, implementation of the application and the presentation of the conclusions, as well as the future recommendations for the application.

Key Words: Medical Care, Firebase, Realtime Database, Kotlin, Jetpack Compose, Android, Mobile App

Πίνακας Περιεχομένων

Εισαγωγή 1
1. Βιβλιογραφική Ανασκόπηση 2
1.1 Εισαγωγή 2
1.2 Εφαρμογή Medisafe 2
1.3 Εφαρμογή Pillbox 3
1.4 Εφαρμογή MyTherapy 3
1.5 Συγγραφή Κώδικα Εφαρμογής 4
2. Ανασκόπηση Τεχνολογιών 6
2.1 Γλώσσες Προγραμματισμού 6
2.1.1 Java 6
2.1.2 Kotlin 6
2.2 Λειτουργικό Σύστημα Android 7
2.2.1 Θετικά και Αρνητικά του Android 7
2.3 Εργαλεία Ανάπτυξης Εφαρμογών σε Android 8
2.3.1 Android Studio 9
2.3.2 IntelliJ IDEA 9
2.3.3 Eclipse 9
2.4 Εργαλεία Ανάπτυξης Εφαρμογών σε Android 9
2.4.1 Κριτήρια Επιλογής Κατάλληλης Κατηγορίας Εφαρμογής 10
2.5 Υπηρεσίες Ιστού και Cloud Storage 11
3. Καθορισμός Απαιτήσεων 15
3.1 Χρήστες της εφαρμογής 15
3.2 Προσέγγιση Καθορισμού Απαιτήσεων 15
3.3 Λειτουργικές Απαιτήσεις 15
3.4 Μη Λειτουργικές Απαιτήσεις 16
4. Σχεδίαση Διεπαφής Χρήστη (User Interface) 17
4.1 Εισαγωγή 17
4.2 Επισκόπηση Εργαλείων Σχεδιασμού Διεπαφής 17
4.2.1 Figma 17
4.2.2 Adobe XD 18
4.2.3 Sketch 18
4.2.4 InVision Studio 18

4.2.5 Android Studio (XML) 19
4.2.6 Επιλογή προγράμματος σχεδιασμού 19
4.3 Χρωματική παλέτα της εφαρμογής 19
4.4 Material Design και Jetpack Compose 20
4.4.1 Material Design 21
4.4.2 Jetpack Compose 21
5. Υλοποίηση Εφαρμογής 23
5.1 Εισαγωγή 23
5.2 Δομή και Σχεδίαση του Συστήματος 23
5.2.1 Αρχές Σχεδιασμού 23
5.2.2 Αρχιτεκτονικές Προσεγγίσεις 24
5.2.3 Προηγμένες Τεχνικές και Βέλτιστες Πρακτικές Σχεδίασης 24
5.3 Διαχείριση Κώδικα και Έκδοσης 27
5.4 Διαχείριση Ταυτοποίησης Χρηστών 27
5.5 Αποθήκευση Δεδομένων του Χρήστη 29
6. Η εφαρμογή Vitalis 33
6.1 Εισαγωγή 33
6.2 Οθόνη Εκκίνησης και Οθόνη Καλωσορίσματος 33
6.3 Οθόνη Σύνδεσης και Εγγραφής 34
6.4 My Medicine (Αρχική Οθόνη) 38
6.5 Λεπτομέρειες Φαρμάκων και Οθόνη Ημερολογίου 42
6.6 Emergency Contacts (Οθόνη Έκτακτων Επαφών) 45
6.7 Account Settings (Οθόνη Ρυθμίσεων Λογαριασμού και Επιλογές) 49
6.7.1 Account Details (Ο Λογαριασμός Μου) 50
6.7.2 Change Language (Αλλαγή Γλώσσας) 51
6.7.3 SOS Contact (SOS Επαφή) 53
6.7.4 Logout (Αποσύνδεση) 55
7. Συμπεράσματα και Μελλοντικές Επεκτάσεις 57
7.1 Συμπεράσματα 57
7.2 Μελλοντικές επεκτάσεις 57
Βιβλιογραφία 59

Λίστα Εικόνων

Εικόνα 1: Παλέτα για Light και Dark Mode 20

Εικόνα 2: Παράδειγμα κώδικα για την δημιουργία λογαριασμού και το login 28

Εικόνα 3: Παράδειγμα κώδικα για αποθήκευση και ανάκτηση δεδομένων μέσω Firebase Realtime Database..... 30

Εικόνα 4: Δομή δεδομένων στη Firebase Realtime Database 32

Εικόνα 5: Οθόνη Εκκίνησης (αριστερά) – Οθόνη Καλωσορίσματος (δεξιά) 34

Εικόνα 6: Οθόνες Εγγραφής (αριστερά) και Σύνδεσης (δεξιά) χωρίς συμπληρωμένα στοιχεία 35

Εικόνα 7: Οθόνες Εγγραφής (αριστερά) και Σύνδεσης (δεξιά) με συμπληρωμένα στοιχεία 36

Εικόνα 8: Οθόνες Εγγραφής (αριστερά) και Σύνδεσης (δεξιά) με λάθη 37

Εικόνα 9: Διάλογοι με μηνύματα λάθους στην Σύνδεση (δεξιά) και Εγγραφή (αριστερά) 38

Εικόνα 10: Οθόνη καταχώρησης ονόματος φαρμάκου (αριστερά) – Οθόνη επιλογής είδους φαρμάκου (δεξιά)..... 39

Εικόνα 11: Οθόνη καταχώρησης αίτιου λήψης του φαρμάκου (αριστερά) – Οθόνη επιτυχούς καταχώρησης φαρμάκου (δεξιά)..... 40

Εικόνα 12: Οθόνη My Medicine χωρίς φάρμακα καταχωρημένα (αριστερά) - με φάρμακα καταχωρημένα (δεξιά)..... 41

Εικόνα 13: Τύποι φαρμάκων και τα εικονίδιά τους 42

Εικόνα 14: Οθόνη λεπτομερειών φαρμάκου (αριστερά), επιλογή διαγραφής (μέση), επιλογή προσθήκης υπενθύμισης (δεξιά) 43

Εικόνα 15: Οθόνη υπενθύμισης: Επανάληψη Υπενθύμισης (αριστερά) – Ημερολόγιο (μέση) – Ρολόι (δεξιά) 44

Εικόνα 16: Οθόνες χωρίς (αριστερά) και με υπενθύμιση (δεξιά)..... 45

Εικόνα 19: Οθόνη για καταχώρηση επαφής χωρίς λάθη (αριστερά) - με λάθη (δεξιά) .. 46

Εικόνα 20: Οθόνη με επιλεγμένο: Γιατρό (αριστερά) – Φαρμακείο (μέση) – Φροντιστή (δεξιά)..... 47

Εικόνα 22: Οθόνη Emergency Contacts χωρίς (αριστερά) και με καταχωρημένες επαφές (δεξιά)..... 48

Εικόνα 23: Οθόνη λεπτομερειών επαφής (αριστερά) και επιλογές (δεξιά) 49

Εικόνα 24: Οθόνη Account Settings χωρίς (αριστερά) και με καταχωρημένα στοιχεία (δεξιά) στο My Account και SOS Contact 50

Εικόνα 25: Οθόνη Account Details : χωρίς συμπληρωμένα στοιχεία (αριστερά) με λάθος (μέση) και με συμπληρωμένα (δεξιά) 51

Εικόνα 27: Οθόνη αλλαγής γλώσσας Αγγλικά (αριστερά) – Ελληνικά (δεξιά) 52

Εικόνα 28: Παραδείγματα οθονών στα ελληνικά 53

Εικόνα 29: Οθόνη καταχώρησης SOS επαφής χωρίς καταχώρηση (αριστερά) – με λάθος καταχώρηση (δεξιά) 54

Εικόνα 31: Οθόνη SOS επαφής με κουμπί για διαγραφή (αριστερά) - επιτυχής οθόνη διαγραφής (δεξιά) 55

Εικόνα 32: Διάλογος αποσύνδεσης 56

Εισαγωγή

Η ταχεία εξέλιξη της τεχνολογίας και η διάδοση των κινητών συσκευών έχουν δημιουργήσει νέες ευκαιρίες για την ανάπτυξη εφαρμογών που μπορούν να βελτιώσουν την καθημερινότητα των χρηστών, ειδικά στον τομέα της υγείας. Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη μιας κινητής εφαρμογής που παρέχει ολοκληρωμένες υπηρεσίες ιατρικής φροντίδας, διευκολύνοντας τη διαχείριση φαρμάκων και την επικοινωνία σε καταστάσεις έκτακτης ανάγκης.

Η σημαντικότητα του έργου αυτού έγκειται στο γεγονός ότι οι εφαρμογές στον τομέα της υγείας είναι ιδιαίτερα κρίσιμες, καθώς οι χρήστες αναζητούν εργαλεία που ενισχύουν την καθημερινή παρακολούθηση της υγείας τους. Αυτή η εφαρμογή έχει τη δυνατότητα να προσφέρει ουσιαστικές βελτιώσεις στην ποιότητα ζωής, παρέχοντας αξιόπιστες και εύχρηστες λύσεις για την παρακολούθηση της φαρμακευτικής αγωγής και την εξασφάλιση άμεσης επικοινωνίας σε περιπτώσεις έκτακτης ανάγκης.

Με στόχο την παροχή μιας εύχρηστης και λειτουργικής εμπειρίας χρήστη, αξιοποιήθηκαν σύγχρονες τεχνολογίες όπως το Firebase [1] για τον συγχρονισμό και την αποθήκευση δεδομένων σε πραγματικό χρόνο, καθώς και η Kotlin [2] και το Jetpack Compose [3]για τη δημιουργία ενός σύγχρονου και φιλικού προς τον χρήστη περιβάλλοντος. Σκοπός της εφαρμογής είναι να βελτιώσει την ποιότητα ζωής των χρηστών, παρέχοντας εύκολη πρόσβαση και διαχείριση πληροφοριών υγείας.

Κατά την υλοποίηση της εφαρμογής, ακολουθήθηκαν οι αρχές του Clean Code [4] και της μεθοδολογίας SOLID, οι οποίες στοχεύουν στη δημιουργία κώδικα που είναι ευανάγνωστος, εύκολα συντηρήσιμος και επεκτάσιμος. Αυτές οι αρχές διασφαλίζουν ότι η εφαρμογή παραμένει ευέλικτη στις αλλαγές και εύκολη στη συντήρηση, προσφέροντας παράλληλα μια σταθερή και αξιόπιστη εμπειρία χρήστη.

Η εργασία χωρίζεται σε επτά κεφάλαια, τα οποία καλύπτουν την ανάλυση των χρησιμοποιούμενων τεχνολογιών, τον σχεδιασμό και την υλοποίηση της εφαρμογής, καθώς και τα συμπεράσματα που προέκυψαν από την ανάπτυξη και τη δοκιμή της.

1. Βιβλιογραφική Ανασκόπηση

1.1 Εισαγωγή

Σε αυτό το κεφάλαιο, γίνεται ανάλυση και σύγκριση εφαρμογών που βρέθηκαν στο Play Store και έχουν παρόμοια λειτουργία με την εφαρμογή που αναπτύσσουμε. Οι εφαρμογές αυτές εστιάζουν στη διαχείριση φαρμάκων και την υποστήριξη των χρηστών στην παρακολούθηση της φαρμακευτικής τους αγωγής. Οι δοκιμές των εφαρμογών πραγματοποιήθηκαν με τη χρήση του smartphone Xiaomi Mi 9 Lite και Android 10, με έμφαση στην αξιολόγηση της λειτουργικότητας, της ευχρηστίας και της αξιοπιστίας των καταχωρήσεων και των ειδοποιήσεων, καθώς και στη συνολική εμπειρία της χρήσης.

1.2 Εφαρμογή Medisafe

Το Medisafe είναι μια από τις πιο διαδεδομένες εφαρμογές στον τομέα της διαχείρισης φαρμάκων, σχεδιασμένη για να βοηθάει τους χρήστες να παρακολουθούν και να διαχειρίζονται την φαρμακευτική τους αγωγή. Η κύρια λειτουργικότητα της εφαρμογής περιλαμβάνει:

- Υπενθυμίσεις Φαρμάκων
- Παρακολούθηση Φαρμακευτικής Αγωγής
- Υπενθυμίσεις Ραντεβού με Γιατρούς
- Προσθήκη προσωπικών σημειώσεων
- Παρακολούθηση υγείας με συμπλήρωση ημερήσιων δεδομένων

Το Medisafe έχει αναπτυχθεί ως native εφαρμογή για Android κυρίως σε γλώσσα Java, η οποία χρησιμοποιείται για την υλοποίηση του μεγαλύτερου μέρους της λειτουργικότητας της εφαρμογής ενώ μικρά τμήματά της έχουν υλοποιηθεί με Shell και Python, τα οποία πιθανώς χρησιμοποιούνται για scripts αυτοματοποίησης ή διαχείρισης της υποδομής.

Τα πλεονεκτήματα της εφαρμογής περιλαμβάνουν τη φιλική προς τον χρήστη διεπαφή, τη δυνατότητα διαχείρισης πολλαπλών προφίλ, τις αξιόπιστες ειδοποιήσεις για τη λήψη φαρμάκων και την υπενθύμιση για ραντεβού, καθώς και την ευελιξία και προσαρμογή στις ανάγκες του χρήστη.

Ωστόσο, υπάρχουν και ορισμένα μειονεκτήματα. Πρώτον, οι επιλογές παραμετροποίησης των ειδοποιήσεων μπορεί να φανούν περιορισμένες για ορισμένους χρήστες. Επιπλέον, η χρήση της εφαρμογής απαιτεί σύνδεση στο διαδίκτυο για τον συγχρονισμό των δεδομένων, γεγονός που μπορεί να προκαλέσει προβλήματα σε περιπτώσεις αποσύνδεσης. Χωρίς σύνδεση στο διαδίκτυο, η εφαρμογή υπολειτουργεί, καθώς πολλές από τις πιο προηγμένες λειτουργίες της, εξαρτώνται από τη συνεχή πρόσβαση στο διαδίκτυο. Η διαχείριση πιο πολύπλοκων φαρμακευτικών αγωγών με συγκεκριμένα χρονοδιαγράμματα είναι επίσης περιορισμένη. Τέλος, οι ειδοποιήσεις μπορεί να μην ληφθούν εάν η συσκευή δεν είναι συνδεδεμένη στο

διαδίκτυο ή αν έχουν ενεργοποιηθεί συγκεκριμένες ρυθμίσεις ή και μετά από επανεκκίνηση της συσκευής.

1.3 Εφαρμογή Pillbox

Το Pillbox είναι μια εφαρμογή διαχείρισης φαρμάκων που βοηθά τους χρήστες να οργανώσουν και να παρακολουθούν την φαρμακευτική τους αγωγή. Οι βασικές λειτουργίες της εφαρμογής περιλαμβάνουν:

- Υπενθυμίσεις φαρμάκων
- Προγραμματισμό δόσεων
- Ιστορικό λήψης φαρμάκων

Η εφαρμογή προσφέρει αξιόπιστες ειδοποιήσεις για τη λήψη των φαρμάκων και ευελιξία στον προγραμματισμό των δόσεων. Η δυνατότητα καταγραφής ιστορικού λήψης φαρμάκων επιτρέπουν στους χρήστες να παρακολουθούν την συνέπεια και την αποτελεσματικότητα της φαρμακευτικής τους αγωγής. Επιπλέον η εφαρμογή μπορεί να λειτουργεί χωρίς σύνδεση στο διαδίκτυο, διασφαλίζοντας της πρόσβαση στις βασικές της λειτουργίες ακόμα και όταν δεν υπάρχει διαθέσιμο δίκτυο.

Παρά τα οφέλη της, η εφαρμογή εμφανίζει ορισμένα σημαντικά μειονεκτήματα. Καταρχάς, η διεπαφή της δεν είναι ιδιαίτερα φιλική προς τον χρήστη, κάτι που μπορεί να δυσκολέψει την πλοήγηση και την χρήση, ειδικά για άτομα με μικρή τεχνολογική εξοικείωση. Οι επιλογές παραμετροποίησης των ειδοποιήσεων μπορεί επίσης να θεωρηθούν περιορισμένες για χρήστες με εξειδικευμένες ανάγκες. Οι ειδοποιήσεις μπορεί επίσης να μην ληφθούν αν έχουν ενεργοποιηθεί συγκεκριμένες ρυθμίσεις ή μετά από επανεκκίνηση της συσκευής.

1.4 Εφαρμογή MyTherapy

Το MyTherapy είναι μια εφαρμογή διαχείρισης φαρμάκων που προσφέρει μια ολοκληρωμένη λύση για την οργάνωση και παρακολούθηση της φαρμακευτικής αγωγής και της συνολικής υγείας των χρηστών. Οι βασικές λειτουργίες της εφαρμογής περιλαμβάνουν:

- Υπενθυμίσεις φαρμάκων
- Λειτουργία ομάδας για φροντιστές
- Παρακολούθηση υγείας και αναφορές

Η εφαρμογή επιτρέπει στους χρήστες να δημιουργούν υπενθυμίσεις για τη λήψη των φαρμάκων, είτε με σάρωση των barcodes των συνταγών είτε με χειροκίνητη εισαγωγή των δεδομένων. Η δυνατότητα παρακολούθησης της υγείας επιτρέπει την καταγραφή μετρήσεων, όπως βάρος, αρτηριακή πίεση και γλυκόζη αίματος, προσφέροντας μια συνολική εικόνα της ευεξίας του χρήστη. Επιπλέον, η λειτουργία ομάδας επιτρέπει στους χρήστες να προσθέσουν έμπιστους φροντιστές, οι οποίοι ειδοποιούνται σε περίπτωση μη συμμόρφωσης με το πρόγραμμα φαρμακευτικής

αγωγής. Όλες αυτές οι πληροφορίες συγκεντρώνονται σε καθημερινές και μηνιαίες αναφορές, παρέχοντας μια σαφή εικόνα της υγείας του χρήστη. Ένα από τα κύρια πλεονεκτήματα της είναι η υποστήριξη σχεδόν 30 γλωσσών, καθιστώντας την προσβάσιμη σε χρήστες από όλο τον κόσμο.

Παρά τα πλεονεκτήματά της, η εφαρμογή παρουσιάζει ορισμένα μειονεκτήματα. Για παράδειγμα, δεν παρέχει πολλές επιλογές υπενθύμισης για την ανανέωση φαρμάκων και δεν διαθέτει προειδοποιήσεις για αλληλεπιδράσεις φαρμάκων. Επιπλέον, η έλλειψη δυνατοτήτων για προσθήκη σημειώσεων σχετικά με τα φάρμακα μπορεί να είναι περιοριστική για ορισμένους χρήστες.

1.5 Συγγραφή Κώδικα Εφαρμογής

Η συγγραφή του κώδικα για την εφαρμογή μας ακολουθεί μια δομημένη και συστηματική προσέγγιση, με σκοπό τη διασφάλιση της ποιότητας, της συντηρησιμότητας και της επεκτασιμότητας του λογισμικού. Κατά τη διάρκεια της ανάπτυξης, δόθηκε έμφαση στη χρήση καλών πρακτικών προγραμματισμού, όπως είναι η σαφής ονοματολογία των μεταβλητών και των μεθόδων, η κατάλληλη δομή των κλάσεων και των αρχείων, καθώς και η τεκμηρίωση του κώδικα με σχόλια, όπου ήταν απαραίτητο.

Οι μεθοδολογίες που ακολουθήθηκαν για την ανάπτυξη του κώδικα βασίστηκαν στο βιβλίο του Robert C. Martin Clean Code: A Handbook of Agile Software Craftsmanship [4]. Το βιβλίο αυτό αποτελεί έναν θεμελιώδη οδηγό για προγραμματιστές και προωθεί τις βέλτιστες πρακτικές για τη συγγραφή καθαρού και συντηρήσιμου κώδικα. Ο Martin, επίσης γνωστός ως "Uncle Bob", προτείνει αρχές όπως η απλότητα, η αναγνωσιμότητα και η σωστή δόμηση, οι οποίες συνεισφέρουν στη δημιουργία κώδικα υψηλής ποιότητας.

Συγκεκριμένα, εφαρμόστηκαν οι ακόλουθες αρχές:

1. **Ονομασία(Naming):** Η σωστή ονοματολογία είναι κρίσιμη για την κατανόηση του κώδικα από άλλους προγραμματιστές. Ονόματα που είναι περιγραφικά και αποκαλύπτουν την πρόθεση χρήσης τους βελτιώνουν την αναγνωσιμότητα του κώδικα.
2. **Αρχές SOLID(SOLID Principles):** Οι αρχές SOLID εφαρμόστηκαν με στόχο τη δημιουργία ευέλικτου και εύκολα συντηρήσιμου κώδικα. Αυτές περιλαμβάνουν τις αρχές Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation και Dependency Inversion.
3. **Διαχείριση Σφαλμάτων(Error Handling):** Υιοθετήθηκαν πρακτικές για την κατάλληλη διαχείριση εξαιρέσεων, εξασφαλίζοντας ότι η εφαρμογή μπορεί να αντιμετωπίσει απρόβλεπτες καταστάσεις χωρίς να διακινδυνεύει η σταθερότητά της.

4. **Σχόλια (Comments):** Η τεκμηρίωση του κώδικα με σχόλια περιορίστηκε στα απολύτως απαραίτητα σημεία, διασφαλίζοντας ότι ο ίδιος ο κώδικας παραμένει αυτοεξηγούμενος και κατανοητός.

Η εφαρμογή αυτών των αρχών επέτρεψε την ανάπτυξη ενός κώδικα που δεν είναι μόνο λειτουργικός αλλά και καλά οργανωμένος, διευκολύνοντας την κατανόηση, τη συντήρηση και την επέκταση του λογισμικού στο μέλλον.

2. Ανασκόπηση Τεχνολογιών

2.1 Γλώσσες Προγραμματισμού

Η ανάπτυξη εφαρμογών για κινητές συσκευές απαιτεί τη χρήση κατάλληλων γλωσσών προγραμματισμού, οι οποίες μπορούν να υποστηρίξουν τις λειτουργίες και τις απαιτήσεις του σύγχρονου λογισμικού. Σε αυτό το κεφάλαιο, θα εξεταστούν δύο βασικές γλώσσες προγραμματισμού για την ανάπτυξη εφαρμογών σε Android, η Java και η Kotlin.

2.1.1 Java

Η Java [5] είναι μια από τις πιο διαδεδομένες γλώσσες προγραμματισμού στον κόσμο, ιδιαίτερα στον χώρο της ανάπτυξης εφαρμογών Android. Αναπτύχθηκε από την Sun Microsystems το 1995 και γρήγορα καθιερώθηκε ως η κύρια γλώσσα για ανάπτυξη εφαρμογών που τρέχουν σε πολλαπλές πλατφόρμες, χάρη στην αρχιτεκτονική της βασισμένη στη μηχανή εικονικής πραγματικότητας (Virtual Machine). Στην περίπτωση του Android, η Java υποστηρίζεται από την Dalvik Virtual Machine (DVM) και, μεταγενέστερα, από την Android Runtime (ART), που επιτρέπει τη μετατροπή του κώδικα σε Dalvik Executable format (.dex), το οποίο είναι κατάλληλο για εκτέλεση σε συσκευές Android [6].

Η δημοτικότητα της Java για την ανάπτυξη Android εφαρμογών οφείλεται στην ευκολία χρήσης της, τη διαλειτουργικότητά της και την εκτεταμένη κοινότητα προγραμματιστών που έχει δημιουργηθεί γύρω από αυτήν. Επιτρέπει την ανάπτυξη εφαρμογών με συνεπή απόδοση, αν και οι κριτικές για την ταχύτητά της σε σχέση με άλλες γλώσσες προγραμματισμού παραμένουν. Επιπλέον, παρά το γεγονός ότι χρησιμοποιεί πιο εκτενή κώδικα, η καθιέρωση της ως η κύρια γλώσσα για το Android συνεχίζει να την καθιστά απαραίτητη για πολλούς προγραμματιστές [7].

2.1.2 Kotlin

Η Kotlin [2] είναι μια νεότερη γλώσσα προγραμματισμού που αναπτύχθηκε από την JetBrains και παρουσιάστηκε το 2016. Από το 2018, υιοθετήθηκε επίσημα από την Google ως η κύρια γλώσσα προγραμματισμού για την ανάπτυξη εφαρμογών Android. Η Kotlin προσφέρει σημαντικά πλεονεκτήματα σε σχέση με την Java, όπως η δυνατότητα γραφής πιο συμπυκνωμένου και λιγότερου κώδικα, κάτι που βελτιώνει την παραγωγικότητα των προγραμματιστών και μειώνει τα πιθανά σφάλματα [7].

Η Kotlin είναι πλήρως συμβατή με την Java, επιτρέποντας τη συγγραφή και εκτέλεση κώδικα και στις δύο γλώσσες χωρίς προβλήματα συμβατότητας. Επιπλέον, η συγκεκριμένη γλώσσα παρέχει βελτιώσεις σε περιοχές όπου η Java παρουσιάζει αδυναμίες, όπως η διαχείριση μηδενικών τιμών (nullability) και οι σύνθετοι κατασκευαστές (complex constructors), προσφέροντας ένα πιο σύγχρονο και αποτελεσματικό περιβάλλον ανάπτυξης [6].

2.2 Λειτουργικό Σύστημα Android

Το λειτουργικό σύστημα Android είναι το πιο διαδεδομένο λογισμικό για κινητές συσκευές, επιτρέποντας την ανάπτυξη και χρήση εφαρμογών που καλύπτουν ένα ευρύ φάσμα αναγκών. Ως ανοιχτό λογισμικό, βασίζεται στον πυρήνα του Linux και επιτρέπει στους προγραμματιστές να προσαρμόζουν τον κώδικα σύμφωνα με τις ανάγκες τους. Η ανάπτυξή του ξεκίνησε το 2003 από την Android Inc. στην Καλιφόρνια, με την καθοδήγηση των Andy Rubin, Rich Miner, και Nick Sears. Το 2005, η Google απέκτησε την εταιρεία, προχωρώντας στη συνεχή εξέλιξη και ενσωμάτωση του Android σε κινητές συσκευές [8, 9].

Το λογισμικό παρουσιάστηκε επίσημα το 2007 και έκτοτε υποστηρίζεται από την Open Handset Alliance, μια κοινοπραξία τεχνολογικών εταιρειών που ενισχύουν την ανάπτυξη και την προσαρμογή του Android. Οι κύριες γλώσσες προγραμματισμού για την ανάπτυξη εφαρμογών είναι η Java και η Kotlin, οι οποίες επιτρέπουν στους προγραμματιστές να δημιουργούν εφαρμογές με μεγάλη ευελιξία και διαλειτουργικότητα [8].

Το Android εξελίσσεται διαρκώς με συνεχείς αναβαθμίσεις που ενισχύουν τόσο την ασφάλεια όσο και την εμφάνιση του, ανταποκρινόμενο στις διαρκώς μεταβαλλόμενες ανάγκες των χρηστών και την ανάπτυξη της τεχνολογίας.

2.2.1 Θετικά και Αρνητικά του Android

Θετικά

1. **Ευρεία Υποστήριξη και Κοινότητα (Wide Support and Community):** Το Android είναι ένα λειτουργικό σύστημα ανοιχτού κώδικα που υποστηρίζεται από μια μεγάλη και ενεργή κοινότητα προγραμματιστών και χρηστών. Αυτή η ευρεία υποστήριξη επιτρέπει τη γρήγορη ανάπτυξη νέων χαρακτηριστικών και τη συνεχή βελτίωση του λειτουργικού συστήματος, καθιστώντας το ευέλικτο και προσαρμόσιμο στις ανάγκες διαφόρων χρηστών και συσκευών [10].
2. **Προσαρμοστικότητα (Customization):** Το Android προσφέρει μεγάλη ευελιξία στην προσαρμογή, επιτρέποντας στους κατασκευαστές και στους χρήστες να αλλάζουν τον τρόπο λειτουργίας και την εμφάνιση της συσκευής τους. Αυτή η δυνατότητα προσαρμογής είναι ένας σημαντικός παράγοντας για την ευρεία χρήση του σε διαφορετικές συσκευές από smartphones μέχρι και tablets [10, 11].
3. **Ποικιλία Συσκευών και Προσιτές Τιμές (Variety of Devices and Affordable Prices):** Το Android διατίθεται σε μια τεράστια γκάμα συσκευών, από οικονομικές μέχρι υψηλών προδιαγραφών, καθιστώντας το προσιτό σε χρήστες με διαφορετικά δημογραφικά και οικονομικά υπόβαθρα. Αυτή η

ποικιλία δίνει στους καταναλωτές πολλές επιλογές και μεγαλύτερη ευελιξία στην επιλογή συσκευής [12].

Αρνητικά

1. **Κατακερματισμός (Fragmentation):** Ένα από τα κύρια προβλήματα του Android είναι ο κατακερματισμός. Καθώς πολλές διαφορετικές εκδόσεις του λειτουργικού συστήματος είναι σε χρήση ταυτόχρονα, δημιουργούνται προβλήματα συμβατότητας με τις εφαρμογές και την ασφάλεια. Οι κατασκευαστές συσκευών έχουν τον έλεγχο των ενημερώσεων, κάτι που οδηγεί σε αργές ή μη συχνές ενημερώσεις.
2. **Ασφάλεια (Security):** Η ανοιχτή φύση του Android το καθιστά πιο ευάλωτο σε κακόβουλο λογισμικό και επιθέσεις σε σχέση με πιο κλειστά συστήματα όπως το iOS. Η έλλειψη έγκαιρων ενημερώσεων για πολλές συσκευές επιδεινώνει αυτό το πρόβλημα, καθιστώντας τις συσκευές Android πιο ευάλωτες σε επιθέσεις.
3. **Προεγκατεστιμένο Λογισμικό (Bloatware):** Οι κατασκευαστές συχνά φορτώνουν τις συσκευές Android με προεγκατεστιμένες εφαρμογές, οι οποίες καταναλώνουν πόρους και επηρεάζουν αρνητικά την απόδοση των συσκευών. Αυτό μπορεί να προκαλέσει ενοχλήσεις στους χρήστες, καθώς αυτές οι εφαρμογές καταλαμβάνουν χώρο και μπορούν να μειώσουν την ταχύτητα του συστήματος.
4. **Ποιότητα Εφαρμογών (App Quality):** Λόγω της ευκολίας ανάπτυξης εφαρμογών στο Android και της λιγότερο αυστηρής πολιτικής αναθεώρησης στο Google Play Store, η ποιότητα των διαθέσιμων εφαρμογών μπορεί να διαφέρει σημαντικά. Υπάρχουν πολλές εφαρμογές χαμηλής ποιότητας, ακόμα και κακόβουλες, που μπορεί να δημιουργήσουν προβλήματα στους χρήστες.
5. **Διάρκεια Ζωής Μπαταρίας (Battery Life):** Η διάρκεια ζωής της μπαταρίας σε συσκευές Android συχνά υποφέρει λόγω της υπερβολικής βελτιστοποίησης από τους κατασκευαστές, κάτι που μπορεί να οδηγήσει σε απρόσμενες διακοπές λειτουργίας εφαρμογών στο παρασκήνιο ή σε καθυστερημένες ειδοποιήσεις. Αυτό το πρόβλημα είναι ιδιαίτερα εμφανές σε ορισμένες συσκευές που εφαρμόζουν επιθετικές τεχνικές εξοικονόμησης μπαταρίας [13].

2.3 Εργαλεία Ανάπτυξης Εφαρμογών σε Android

Για την δημιουργία εφαρμογών σε Android, υπάρχουν διάφορα εργαλεία που οι προγραμματιστές μπορούν να αξιοποιήσουν, καθένα με τα δικά του πλεονεκτήματα και μειονεκτήματα. Αυτά τα εργαλεία συμβάλλουν στη βελτιστοποίηση της διαδικασίας ανάπτυξης, επιτρέποντας την εύκολη ενσωμάτωση, τον εντοπισμό σφαλμάτων και τη δοκιμή εφαρμογών.

2.3.1 Android Studio

Το **Android Studio** [14] αποτελεί το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) της Google για εφαρμογές Android. Βασισμένο στην πλατφόρμα IntelliJ IDEA [15], το Android Studio προσφέρει ένα εκτενές σύνολο εργαλείων, συμπεριλαμβανομένων προηγμένων δυνατοτήτων επεξεργασίας κώδικα, εντοπισμού σφαλμάτων και ανάλυσης απόδοσης. Το ενσωματωμένο Layout Editor [16] επιτρέπει τη σχεδίαση γραφικών διεπαφών χρήστη μέσω μιας εύχρηστης drag-and-drop διεπαφής, ενώ η χρήση του Gradle [17] ως εργαλείου κατασκευής παρέχει ευελιξία στη διαχείριση εξαρτήσεων και στη δημιουργία διαφορετικών εκδόσεων των εφαρμογών. Ωστόσο, μπορεί να είναι απαιτητικό σε πόρους συστήματος, γεγονός που μπορεί να επιβραδύνει τη διαδικασία ανάπτυξης σε συστήματα με χαμηλότερες προδιαγραφές**.**

2.3.2 IntelliJ IDEA

Το **IntelliJ IDEA** [15], ανεπτυγμένο από την JetBrains, αποτελεί μια ισχυρή επιλογή για την ανάπτυξη εφαρμογών Android, ιδιαίτερα για προγραμματιστές που εργάζονται με τη γλώσσα Java. Το περιβάλλον προσφέρει προηγμένες δυνατότητες όπως ανάλυση κώδικα σε πραγματικό χρόνο, ανακατασκευή κώδικα (refactoring), και έξυπνη αυτόματη συμπλήρωση. Αυτές οι δυνατότητες βοηθούν στην παραγωγή καθαρότερου και πιο αποδοτικού κώδικα, αν και η πολυπλοκότητα του μπορεί να αποτελέσει πρόκληση για λιγότερο έμπειρους προγραμματιστές.

2.3.3 Eclipse

Το Eclipse [18] ήταν το κύριο εργαλείο για την ανάπτυξη εφαρμογών Android πριν την κυκλοφορία του Android Studio. Αν και έχει αντικατασταθεί ως το κύριο εργαλείο από την Google, το Eclipse εξακολουθεί να χρησιμοποιείται από προγραμματιστές λόγω της ευελιξίας του και της δυνατότητας προσαρμογής μέσω plugins. Παρέχει βασικές δυνατότητες ανάπτυξης, όπως η αυτόματη συμπλήρωση κώδικα και ο εντοπισμός σφαλμάτων, αλλά δεν διαθέτει τις εξειδικευμένες λειτουργίες που προσφέρει το Android Studio, καθιστώντας το λιγότερο αποδοτικό.

2.4 Εργαλεία Ανάπτυξης Εφαρμογών σε Android

Οι εφαρμογές για κινητές συσκευές κατηγοριοποιούνται σε τρεις βασικούς τύπους, τις εγγενείς εφαρμογές (Native Apps), εφαρμογές ιστού (Web Apps) και υβριδικές εφαρμογές (Hybrid Apps). Κάθε τύπος έχει τα δικά του πλεονεκτήματα και μειονεκτήματα, και η επιλογή του κατάλληλου τύπου εξαρτάται από παράγοντες όπως οι τεχνικές απαιτήσεις, ο προϋπολογισμός, και οι στόχοι της εφαρμογής.

Εγγενείς Εφαρμογές (Native Apps): Οι εγγενείς εφαρμογές αναπτύσσονται αποκλειστικά για μία συγκεκριμένη πλατφόρμα, όπως το Android ή το iOS. Χρησιμοποιούν τις γλώσσες προγραμματισμού και τα εργαλεία που είναι βελτιστοποιημένα για την πλατφόρμα αυτή, όπως η Java ή η Kotlin για το Android και

το Swift [19] για το iOS. Λόγω της εξειδικευμένης ανάπτυξης, οι εγγενείς εφαρμογές έχουν υψηλή απόδοση και παρέχουν μια ομαλή και ευχάριστη εμπειρία χρήστη. Επιπλέον, οι εγγενείς εφαρμογές έχουν πλήρη πρόσβαση στις λειτουργίες του υλικού της συσκευής, όπως η κάμερα ή το GPS. Ωστόσο, η ανάπτυξη αυτών των εφαρμογών απαιτεί περισσότερο χρόνο και μεγαλύτερο προϋπολογισμό, ειδικά αν η εφαρμογή πρέπει να αναπτυχθεί για πολλαπλές πλατφόρμες.

Εφαρμογές Ιστού (Web Apps): Οι εφαρμογές ιστού είναι ουσιαστικά ιστοσελίδες που έχουν σχεδιαστεί για να προσφέρουν μια εμπειρία παρόμοια με αυτήν των εφαρμογών κινητών. Αυτές οι εφαρμογές είναι κατασκευασμένες χρησιμοποιώντας τεχνολογίες ανάπτυξης ιστοσελίδων, όπως HTML, CSS και JavaScript, και είναι προσβάσιμες μέσω ενός προγράμματος περιήγησης ιστού σε οποιαδήποτε κινητή συσκευή. Το πλεονέκτημα των εφαρμογών ιστού είναι ότι είναι ανεξάρτητες από την πλατφόρμα και μπορούν να λειτουργούν σε οποιαδήποτε συσκευή με σύνδεση στο διαδίκτυο. Ωστόσο, μπορεί να μην προσφέρουν την ίδια απόδοση και αίσθηση όπως οι εγγενείς εφαρμογές, και έχουν περιορισμένη πρόσβαση σε λειτουργίες της συσκευής.

Υβριδικές Εφαρμογές (Hybrid Apps): Οι υβριδικές εφαρμογές συνδυάζουν τα χαρακτηριστικά των εγγενών εφαρμογών και των εφαρμογών ιστού. Είναι κατασκευασμένες χρησιμοποιώντας τεχνολογίες ιστού όπως HTML(Hypertext Markup Language**)**, CSS(Cascading Style Sheets) και JavaScript, αλλά ενσωματώνονται σε ένα κέλυφος εγγενούς εφαρμογής, το οποίο τους επιτρέπει να εγκαθίστανται και να εκτελούνται σαν εγγενείς εφαρμογές. Οι υβριδικές εφαρμογές μπορούν να χρησιμοποιούν τις λειτουργίες της συσκευής, ενώ ταυτόχρονα είναι ευκολότερο να αναπτυχθούν για πολλαπλές πλατφόρμες, αφού χρησιμοποιούν κοινή βάση κώδικα. Ωστόσο, η απόδοσή τους μπορεί να μην είναι εξίσου υψηλή με αυτή των εγγενών εφαρμογών, και η εμπειρία χρήστη μπορεί να είναι λιγότερο ομαλή.

2.4.1 Κριτήρια Επιλογής Κατάλληλης Κατηγορίας Εφαρμογής

Η επιλογή του κατάλληλου τύπου εφαρμογής για κινητές συσκευές πρέπει να βασίζεται σε διάφορους παράγοντες, οι οποίοι θα διασφαλίσουν ότι η εφαρμογή ανταποκρίνεται στις ανάγκες του κοινού-στόχου και επιτυγχάνει τους στόχους της ανάπτυξης.

1. **Εμπειρία Χρήστη (User Experience):** Η επιθυμητή εμπειρία χρήστη είναι ένας βασικός παράγοντας. Οι εγγενείς εφαρμογές προσφέρουν την καλύτερη εμπειρία λόγω της άμεσης πρόσβασης στις δυνατότητες της συσκευής και της βελτιστοποιημένης απόδοσης. Αν η εμπειρία χρήστη είναι κρίσιμη, η ανάπτυξη μιας εγγενούς εφαρμογής μπορεί να είναι η καλύτερη επιλογή.
2. **Δυνατότητες (Features):** Οι απαιτήσεις της εφαρμογής σε δυνατότητες και λειτουργικότητα είναι επίσης κρίσιμες. Αν η εφαρμογή απαιτεί εκτεταμένη

χρήση των λειτουργιών της συσκευής, όπως η κάμερα ή το GPS, τότε οι εγγενείς ή υβριδικές εφαρμογές είναι οι κατάλληλες επιλογές.

3. **Προϋπολογισμός (Budget):** Ο διαθέσιμος προϋπολογισμός επηρεάζει άμεσα την επιλογή. Οι εγγενείς εφαρμογές είναι πιο ακριβές στην ανάπτυξη, ειδικά αν απαιτούνται πολλαπλές εκδόσεις για διαφορετικές πλατφόρμες. Οι υβριδικές εφαρμογές μπορεί να προσφέρουν μια πιο οικονομική λύση με αποδεκτή απόδοση και εμπειρία χρήστη.
4. **Κοινό-στόχος (Target Audience):** Η ανάλυση του κοινού-στόχου είναι ζωτικής σημασίας. Αν οι χρήστες χρησιμοποιούν κυρίως μία πλατφόρμα, τότε μια εγγενής εφαρμογή μπορεί να είναι προτιμότερη. Αντίθετα, αν το κοινό είναι διασκορπισμένο σε πολλαπλές πλατφόρμες, οι υβριδικές ή εφαρμογές ιστού προσφέρουν μεγαλύτερη ευελιξία και προσβασιμότητα.

Αυτοί οι παράγοντες είναι καθοριστικοί για την επιλογή του κατάλληλου τύπου εφαρμογής, καθώς διασφαλίζουν ότι η εφαρμογή όχι μόνο ανταποκρίνεται στις τεχνικές απαιτήσεις αλλά και προσφέρει την καλύτερη δυνατή εμπειρία χρήστη εντός των προϋπολογιστικών περιορισμών.

2.5 Υπηρεσίες Ιστού και Cloud Storage

Η ανάπτυξη εφαρμογών για κινητές συσκευές συχνά απαιτεί τη χρήση υπηρεσιών ιστού (web services) και αποθήκευσης στο cloud (cloud storage) για την αποθήκευση, διαχείριση και επεξεργασία δεδομένων. Αυτές οι τεχνολογίες επιτρέπουν στους προγραμματιστές να δημιουργούν εφαρμογές που μπορούν να προσφέρουν δυναμικό περιεχόμενο και να λειτουργούν σε πραγματικό χρόνο.

Firebase: Το Firebase [1] είναι μια πλατφόρμα ανάπτυξης εφαρμογών που παρέχεται από την Google και προσφέρει ένα πλήθος υπηρεσιών που καλύπτουν τις ανάγκες τόσο των μικρών όσο και των μεγάλων εφαρμογών. Οι κύριες υπηρεσίες που παρέχει το Firebase περιλαμβάνουν:

- **Αυθεντικοποίηση(Authentication):** Επιτρέπει την εύκολη ενσωμάτωση συστημάτων ταυτοποίησης χρηστών, όπως σύνδεση μέσω Google, Facebook, email και κωδικού πρόσβασης, χωρίς την ανάγκη για πολύπλοκη ανάπτυξη στο backend.
- **Realtime Database:** Παρέχει μια βάση δεδομένων NoSQL που αποθηκεύει και συγχρονίζει δεδομένα σε πραγματικό χρόνο. Αυτό είναι ιδιαίτερα χρήσιμο για εφαρμογές που απαιτούν άμεση ανταπόκριση, όπως chat εφαρμογές και συνεργατικά εργαλεία.
- **Firestore:** Μια σύγχρονη, cloud-based βάση δεδομένων που υποστηρίζει queries, αυτοματοποιημένες ενημερώσεις, και μπορεί να χρησιμοποιηθεί τόσο για μικρές όσο και για πολύπλοκες εφαρμογές.

- **Cloud Storage:** Προσφέρει αποθήκευση αρχείων (όπως εικόνες, βίντεο και ήχο) με υψηλό επίπεδο ασφάλειας και δυνατότητες γρήγορης ανάκτησης. Το Cloud Storage της Firebase είναι ιδανικό για εφαρμογές που απαιτούν αποθήκευση μεγάλων ποσοτήτων δεδομένων.
- **Cloud Functions:** Επιτρέπει την εκτέλεση backend κώδικα ως απόκριση σε γεγονότα που προκύπτουν στη βάση δεδομένων, στο σύστημα ταυτοποίησης ή σε άλλα συστήματα του Firebase, προσφέροντας τη δυνατότητα να εκτελούνται λειτουργίες που απαιτούν πόρους server χωρίς την ανάγκη διαχείρισης server.
- **Analytics και Crashlytics**: Το Firebase παρέχει επίσης εργαλεία για την ανάλυση της συμπεριφοράς των χρηστών, την παρακολούθηση της απόδοσης της εφαρμογής, και τη διαχείριση σφαλμάτων, καθιστώντας το μια ολοκληρωμένη λύση για την ανάπτυξη και τη συντήρηση εφαρμογών.

Αρνητικά

Παρά τα πλεονεκτήματα που προσφέρει το Firebase, υπάρχουν και ορισμένα αρνητικά σημεία που πρέπει να ληφθούν υπόψη. Το Firebase μπορεί να οδηγήσει σε εξάρτηση από τον πάροχο (vendor lock-in), καθιστώντας δύσκολη τη μετεγκατάσταση σε άλλες πλατφόρμες στο μέλλον. Επίσης, το κόστος μπορεί να αυξηθεί γρήγορα καθώς η εφαρμογή μεγαλώνει και απαιτεί περισσότερους πόρους. Επιπλέον, οι δυνατότητες του Firebase Firestore είναι περιορισμένες σε σχέση με μια παραδοσιακή SQL βάση δεδομένων, ενώ η πλήρης αξιοποίηση του απαιτεί σημαντικό χρόνο εκμάθησης, ιδιαίτερα για νέους προγραμματιστές.

Amazon Web Services (AWS): Το AWS [20] είναι μία από τις μεγαλύτερες και πιο διαδεδομένες πλατφόρμες cloud, προσφέροντας ένα ευρύ φάσμα υπηρεσιών για την ανάπτυξη και διαχείριση εφαρμογών. Η ευελιξία και η κλιμακωσιμότητα των υπηρεσιών της AWS την καθιστούν ιδανική επιλογή για εφαρμογές κάθε μεγέθους, από μικρά startups μέχρι μεγάλες επιχειρήσεις.

- **Amazon S3 (Simple Storage Service):** Το S3 είναι μια υπηρεσία αποθήκευσης αντικειμένων που επιτρέπει την αποθήκευση και ανάκτηση δεδομένων σε οποιαδήποτε ποσότητα, από οπουδήποτε. Προσφέρει υψηλή διαθεσιμότητα και ανθεκτικότητα, καθιστώντας το κατάλληλο για εφαρμογές που απαιτούν αποθήκευση μεγάλων αρχείων, όπως πολυμέσα ή αρχεία backup.
- **Amazon RDS (Relational Database Service):** Το RDS παρέχει μια κλιμακωτή, διαχειριζόμενη υπηρεσία βάσεων δεδομένων που υποστηρίζει πολλαπλές μηχανές βάσεων δεδομένων όπως MySQL, PostgreSQL, και SQL Server. Αυτό επιτρέπει στους προγραμματιστές να αναπτύσσουν και να διαχειρίζονται εφαρμογές με πολύπλοκες δομές δεδομένων χωρίς να ανησυχούν για την υποδομή.

- **Amazon DynamoDB:** Πρόκειται για μια υπηρεσία βάσεων δεδομένων NoSQL που προσφέρει γρήγορη και ευέλικτη αποθήκευση δεδομένων για εφαρμογές που απαιτούν υψηλή ταχύτητα και απόδοση. Είναι ιδανικό για εφαρμογές που απαιτούν χαμηλή καθυστέρηση και μπορούν να κλιμακωθούν σε παγκόσμια κλίμακα.
- **Amazon Lambda:** Μια υπηρεσία υπολογιστικού νέφους χωρίς διακομιστές, που επιτρέπει στους προγραμματιστές να εκτελούν backend κώδικα σε απόκριση σε γεγονότα χωρίς να διαχειρίζονται τους ίδιους τους servers. Αυτή η προσέγγιση μπορεί να μειώσει τα κόστη και την πολυπλοκότητα της υποδομής.

Αρνητικά

Η χρήση των υπηρεσιών της AWS μπορεί να είναι σύνθετη και απαιτεί σημαντική καμπύλη εκμάθησης, ειδικά για νεότερους προγραμματιστές. Επιπλέον, τα κόστη μπορούν να αυξηθούν γρήγορα εάν η υποδομή δεν είναι καλά διαχειριζόμενη ή εάν οι απαιτήσεις κλιμακωθούν απροσδόκητα.

Google Cloud Platform (GCP): Το GCP [21] είναι η πλατφόρμα cloud της Google που παρέχει ένα ολοκληρωμένο σύνολο υπηρεσιών για την ανάπτυξη, αποθήκευση, και ανάλυση δεδομένων. Το GCP προσφέρει βαθιά ενσωμάτωση με τα προϊόντα της Google, γεγονός που το καθιστά μια ελκυστική επιλογή για εφαρμογές που βασίζονται σε big data και τεχνητή νοημοσύνη.

- **Google Cloud Storage:** Μια υπηρεσία αποθήκευσης αντικειμένων παρόμοια με το Amazon S3, που επιτρέπει την αποθήκευση και ανάκτηση δεδομένων σε κλίμακα. Είναι ιδιαίτερα κατάλληλη για αποθήκευση μεγάλων αρχείων και στατικών περιεχομένων, όπως εικόνες, βίντεο και αρχεία backup.
- **Google BigQuery:** Μια ισχυρή υπηρεσία ανάλυσης δεδομένων που επιτρέπει στους χρήστες να εκτελούν σύνθετα queries σε μεγάλα σύνολα δεδομένων σε λίγα δευτερόλεπτα. Είναι ιδανικό για εφαρμογές που απαιτούν ανάλυση μεγάλου όγκου δεδομένων σε πραγματικό χρόνο.
- **Google Cloud Functions:** Παρέχει δυνατότητες υπολογισμού χωρίς διακομιστές, επιτρέποντας την εκτέλεση backend κώδικα σε απόκριση σε γεγονότα χωρίς να απαιτείται διαχείριση της υποδομής.

Αρνητικά

Η πλήρης αξιοποίηση των υπηρεσιών της GCP μπορεί να απαιτεί βαθιά τεχνική γνώση και εξειδίκευση, κάτι που μπορεί να είναι πρόκληση για μικρότερες ομάδες ανάπτυξης. Επιπλέον, το κόστος μπορεί να αυξηθεί γρήγορα ανάλογα με τη χρήση και τις απαιτήσεις της εφαρμογής.

Microsoft Azure: Το Microsoft Azure [22] είναι η πλατφόρμα cloud της Microsoft, η οποία προσφέρει ένα ευρύ φάσμα υπηρεσιών, από αποθήκευση και βάσεις δεδομένων μέχρι εργαλεία ανάλυσης και τεχνητής νοημοσύνης. Το Azure είναι ιδιαίτερα κατάλληλο για επιχειρήσεις που ήδη χρησιμοποιούν τα προϊόντα της Microsoft, λόγω της άμεσης ενσωμάτωσης και διαλειτουργικότητας.

- **Azure Blob Storage:** Μια υπηρεσία αποθήκευσης αντικειμένων που επιτρέπει την αποθήκευση μεγάλων ποσοτήτων δεδομένων όπως αρχεία εικόνων, βίντεο και backup. Προσφέρει κλιμακωτή αποθήκευση και υψηλή διαθεσιμότητα, καθιστώντας την κατάλληλη για επιχειρήσεις που απαιτούν ασφαλή και αξιόπιστη αποθήκευση δεδομένων.
- **Azure SQL Database:** Μια διαχειριζόμενη υπηρεσία βάσεων δεδομένων που προσφέρει υποστήριξη για SQL Server και επιτρέπει τη διαχείριση και ανάλυση δεδομένων με υψηλή απόδοση και ασφάλεια. Είναι ιδανικό για εφαρμογές που απαιτούν σταθερές και κλιμακωτές βάσεις δεδομένων.
- **Azure Functions:** Μια υπηρεσία υπολογισμού χωρίς διακομιστές, που επιτρέπει την εκτέλεση κώδικα σε απόκριση σε γεγονότα χωρίς τη διαχείριση της υποδομής.

Αρνητικά

Παρότι το Azure προσφέρει ευρεία γκάμα υπηρεσιών, η καμπύλη εκμάθησης μπορεί να είναι απότομη για τους προγραμματιστές που δεν είναι εξοικειωμένοι με το οικοσύστημα της Microsoft. Επίσης, το κόστος μπορεί να είναι υψηλό αν δεν υπάρχει προσεκτικός σχεδιασμός και παρακολούθηση της χρήσης των υπηρεσιών.

3. Καθορισμός Απαιτήσεων

3.1 Χρήστες της εφαρμογής

Η εφαρμογή απευθύνεται σε διάφορες κατηγορίες χρηστών, με κύριο στόχο την υποστήριξη της διαχείρισης φαρμάκων και επαφών έκτακτης ανάγκης. Οι χρήστες μπορεί να είναι ηλικιωμένοι, ασθενείς με χρόνιες παθήσεις ή άτομα που απλώς θέλουν να έχουν οργανωμένες και εύκολα προσβάσιμες πληροφορίες για τα φάρμακά τους και τις επαφές τους. Επιπλέον, οι φροντιστές μπορούν να χρησιμοποιούν την εφαρμογή για να παρακολουθούν την φαρμακευτική αγωγή των ατόμων που φροντίζουν αλλά και να επικοινωνήσουν με κάποιο κοντινό πρόσωπο ή γιατρό σε περίπτωση ανάγκης.

3.2 Προσέγγιση Καθορισμού Απαιτήσεων

Η διαδικασία καθορισμού των απαιτήσεων για την εφαρμογή ακολούθησε μια συστηματική προσέγγιση, με στόχο να εξασφαλιστεί η κάλυψη όλων των κρίσιμων λειτουργιών και η παροχή μιας ευχάριστης και αποδοτικής εμπειρίας.

Η διαδικασία βασίστηκε κυρίως στην επισκόπηση των ήδη υπαρχουσών εφαρμογών στον τομέα διαχείρισης φαρμάκων και επαφών έκτακτης ανάγκης. Μέσα από αυτή την ανάλυση, εντοπίστηκαν τα κοινά προβλήματα και οι περιορισμοί που αντιμετωπίζουν οι χρήστες στις υπάρχουσες λύσεις. Τα ευρήματα της επισκόπησης αποτέλεσαν τη βάση για τον καθορισμό των λειτουργικών απαιτήσεων της νέας εφαρμογής και τη συνολική βελτίωση της εμπειρίας του χρήστη.

3.3 Λειτουργικές Απαιτήσεις

- Ένας χρήστης θα μπορεί να εγγραφεί και να συνδεθεί χρησιμοποιώντας το email τους και έναν κωδικό που ο ίδιος θα δημιουργήσει.
- Ένας χρήστης θα μπορεί να προσθέσει και να αφαιρέσει καταχωρήσεις με φάρμακα και επαφές.
- Ένας χρήστης θα μπορεί να δει όλες τις λεπτομέρειες για κάθε μια από τις καταχωρήσεις που έχει κάνει στις λίστες με τα καταγεγραμμένα φάρμακα και τις επαφές.
- Κάθε καταχώρηση που δημιουργείται από τον χρήστη θα αποθηκεύεται στη Realtime Database αλλά και τοπικά.
- Ένας χρήστης θα έχει την δυνατότητα να προσθέσει και να αφαιρέσει καταχωρήσεις από τις λίστες με τα καταγεγραμμένα φάρμακα και τις επαφές.

- Ένας χρήστης θα μπορεί να προσθέσει υπενθύμιση των φαρμάκων που έχει καταχωρήσει.
- Ένας χρήστης θα μπορεί να καταχωρήσει πληροφορίες σχετικά με τον εαυτό του (όνομα, επώνυμο, τηλέφωνο).

3.4 Μη Λειτουργικές Απαιτήσεις

- Η εφαρμογή πρέπει να είναι αποδοτική καθώς και να ανταποκρίνεται γρήγορα στις ενέργειες των χρηστών, διασφαλίζοντας ευχάριστη εμπειρία χρήσης.
- Πρέπει να είναι φιλική προς τον χρήστη, με ευανάγνωστη και εύκολη διεπαφή.
- Η εφαρμογή θα πρέπει να μπορεί να εγκαθίσταται και να εκτελείται σε συσκευές που τρέχουν Android 8 (API 26) ή κάποια νεότερη έκδοση.
- Η εφαρμογή πρέπει να παρέχει υποστήριξη για καταγραφή διάφορων σφαλμάτων.
- Για την γενική λειτουργία της εφαρμογής απαιτείται σύνδεση στο διαδίκτυο.
- Η εφαρμογή πρέπει να υποστηρίζει ελληνικά και αγγλικά για τη γλώσσα παρουσίασης της διεπαφής.
- Η εφαρμογή πρέπει να υλοποιηθεί βάση των αρχών του Clean Code όπως ορίζονται στο Clean Code: A Handbook of Agile Software Craftsmanship.
- Η εφαρμογή πρέπει να υλοποιηθεί στη γλώσσα προγραμματισμού Kotlin με την χρήση Jetpack Compose για τη διεπαφή (User Interface).

4. Σχεδίαση Διεπαφής Χρήστη (User Interface)

4.1 Εισαγωγή

Προτού ξεκινήσει η διαδικασία της ανάπτυξης της εφαρμογής, είναι σημαντικό να διαμορφωθεί η δομή και η λειτουργικότητα της γραφικής διεπαφής. Κάθε οθόνη πρέπει να εξεταστεί λεπτομερώς, λαμβάνοντας υπόψη διάφορες καταστάσεις και σενάρια που μπορεί να προκύψουν κατά την χρήση της. Σημεία που μπορεί να προσεχθούν είναι πιθανές ανωμαλίες, όπως δυσλειτουργίες δικτύου ή λάθη που μπορεί να προκύψουν από λανθασμένες ενέργειες του χρήστη. Κατά τη διαμόρφωση της γραφικής διεπαφής, είναι σημαντικό να ληφθούν υπόψη οι ανάγκες του χρήστη και να διασφαλιστεί η απλότητα και η ευκολία χρήσης.

4.2 Επισκόπηση Εργαλείων Σχεδιασμού Διεπαφής

Για τη δημιουργία της διεπαφής χρήστη (UI) της εφαρμογής μας, εξετάστηκαν διάφορα εργαλεία σχεδιασμού που προσφέρουν διαφορετικές προσεγγίσεις και λειτουργίες. Η επιλογή του κατάλληλου εργαλείου ήταν κρίσιμη για την επίτευξη ενός αποδοτικού και ευέλικτου σχεδιασμού.

Η ανάλυση αυτών των εργαλείων επικεντρώθηκε στην ικανότητά τους να διευκολύνουν τον σχεδιασμό της διεπαφής για κινητές εφαρμογές, με ιδιαίτερη έμφαση στην επαναχρησιμοποίηση στοιχείων σε διαφορετικές οθόνες. Η στρατηγική που ακολουθήθηκε, βασίστηκε στη λογική της σχεδίασης από τα μικρότερα στοιχεία προς τα μεγαλύτερα και τελικώς ολόκληρες τις οθόνες.

Αρχικά, θα σχεδιαστούν τα βασικά στοιχεία (Components) όπως κουμπιά, κείμενα, πεδία εισαγωγής δεδομένων, αντικείμενα για λίστες, η πάνω και κάτω μπάρα, καθώς και οι χρωματικές παλέτες και οι γραμματοσειρές. Αυτά τα βασικά Components θα αποτελέσουν τη βάση για τον σχεδιασμό των υπόλοιπων οθονών της εφαρμογής. Με την ολοκλήρωση της προετοιμασίας αυτών των στοιχείων, η διαδικασία σχεδιασμού των οθονών θα ολοκληρωθεί με τη χρήση αυτών των δομικών στοιχείων. Με γνώμονα αυτό το ζητούμενο, πραγματοποιήθηκε λεπτομερής εξέταση των πιο δημοφιλών εργαλείων σχεδίασης UI/UX για να διαπιστωθεί ποιο από αυτά πληροί καλύτερα τις ανάγκες της πτυχιακής εργασίας μας.

4.2.1 Figma

Το Figma είναι ένα web-based εργαλείο σχεδιασμού που προσφέρει εξαιρετικές δυνατότητες συνεργασίας σε πραγματικό χρόνο, επιτρέποντας σε πολλές ομάδες να εργάζονται ταυτόχρονα πάνω σε ένα project. Ένα από τα κύρια πλεονεκτήματά του

είναι η δυνατότητα πρόσβασης από οποιαδήποτε συσκευή με σύνδεση στο διαδίκτυο, διευκολύνοντας την ευέλικτη εργασία και την εύκολη διανομή αρχείων. Παράλληλα, το Figma υποστηρίζει την επαναχρησιμοποίηση στοιχείων μέσω της ισχυρής διαχείρισης components, κάνοντας τον σχεδιασμό διεπαφών πιο αποδοτικό. Παρόλο που η φιλική του διεπαφή το καθιστά εύκολο στη χρήση ακόμα και για αρχάριους, το εργαλείο παρουσιάζει ορισμένα μειονεκτήματα, όπως η εξάρτηση από τη σύνδεση στο διαδίκτυο και οι περιορισμένες δυνατότητες διαχείρισης αρχείων εκτός σύνδεσης. Επιπλέον, ενώ είναι εύκολο στη χρήση για βασικές λειτουργίες, οι πιο προηγμένες δυνατότητες του μπορεί να απαιτούν περισσότερο χρόνο εκμάθησης, ενώ η αποθήκευση στο cloud μπορεί να προκαλεί ανησυχίες σχετικά με την ασφάλεια και την ιδιωτικότητα των δεδομένων.

4.2.2 Adobe XD

Το Adobe XD είναι ένα ισχυρό εργαλείο σχεδιασμού και δημιουργίας πρωτοτύπων που ενσωματώνεται άψογα με άλλα προϊόντα της Adobe, όπως το Photoshop και το Illustrator, διευκολύνοντας τη ροή εργασίας για σχεδιαστές που χρησιμοποιούν ήδη αυτά τα εργαλεία. Παράλληλα, διαθέτει μια φιλική προς τον χρήστη διεπαφή, καθιστώντας το εύκολο στη χρήση τόσο για αρχάριους όσο και για πιο έμπειρους σχεδιαστές. Το Adobe XD είναι ιδιαίτερα διαδεδομένο στην αγορά, καθώς πολλοί σχεδιαστές το προτιμούν για την ευκολία ενσωμάτωσής του με το ευρύτερο οικοσύστημα της Adobe. Παρέχει ισχυρά εργαλεία για τη δημιουργία πρωτοτύπων, επιτρέποντας στους χρήστες να δοκιμάσουν και να βελτιώσουν τη λειτουργικότητα των διεπαφών τους. Ωστόσο, το εργαλείο έχει περιορισμένη υποστήριξη για επεκτάσεις από τρίτους, γεγονός που μπορεί να περιορίσει την προσαρμοστικότητα και τις δυνατότητές του. Επίσης ενώ η ενσωμάτωσή του με το οικοσύστημα Adobe είναι χρήσιμη, μπορεί να δημιουργήσει εξάρτηση, περιορίζοντας τις επιλογές για χρήστες που δεν χρησιμοποιούν άλλα εργαλεία της εταιρίας.

4.2.3 Sketch

Το Sketch είναι ένα δημοφιλές εργαλείο σχεδιασμού που χρησιμοποιείται ευρέως για τη δημιουργία διεπαφών UI/UX, με μια φιλική προς το χρήστη διεπαφή, ακόμα και για αρχάριους σχεδιαστές. Υποστηρίζει τη δημιουργία components και διαθέτει μια εκτεταμένη βιβλιοθήκη από plugins που επεκτείνουν τις δυνατότητες των χρηστών σύμφωνα με τις ανάγκες τους. Ένα σημαντικό πλεονέκτημα του είναι η ισχυρή κοινότητα χρηστών που παρέχει συνεχώς νέους πόρους και βοηθά στην επίλυση προβλημάτων, κάτι που ενισχύει την εμπειρία χρήσης. Ωστόσο, κύριο μειονέκτημα του Sketch είναι ότι είναι διαθέσιμο μόνο για χρήστες των macOS, κάτι που περιορίζει τη χρήση του από άλλες πλατφόρμες. Επιπλέον, είναι επιρρεπές σε σφάλματα, επιβαρύνει τους πόρους των υπολογιστών και συχνά απαιτεί plugins για βασικές λειτουργίες, κάτι που μπορεί να κατακερματίσει την εμπειρία χρήσης.

4.2.4 InVision Studio

Το InVision Studio είναι ένα ισχυρό εργαλείο σχεδιασμού, ιδιαίτερα γνωστό για τις δυνατότητές του στη δημιουργία διαδραστικών πρωτοτύπων και κινούμενων γραφικών. Ενισχύει τη συνεργασία μεταξύ των μελών μιας ομάδας, επιτρέποντας την εύκολη κοινή χρήση και ανατροφοδότηση σε πραγματικό χρόνο. Παρ ’όλα αυτά, μπορεί να παρουσιάσει περιορισμένη ευελιξία στη διαχείριση σύνθετων σχεδίων και απαιτεί μέτριο έως υψηλό επίπεδο γνώσης για την πλήρη αξιοποίηση των δυνατοτήτων του. Παρά την ισχύ του, η έλλειψη προσαρμοστικότητας μπορεί να αποτελέσει πρόκληση για πιο προχωρημένους χρήστες.

4.2.5 Android Studio (XML)

Το Android Studio είναι το επίσημο περιβάλλον ανάπτυξης για εφαρμογές Android και προσφέρει εκτεταμένες δυνατότητες για τη δημιουργία διεπαφών χρήστη μέσω XML. Είναι ιδιαίτερα χρήσιμο για προγραμματιστές που θέλουν να συνδυάσουν τον σχεδιασμό με την άμεση υλοποίηση της εφαρμογής. Η εξοικείωση με το Android Studio είναι σημαντική για όσους εργάζονται σε Android εφαρμογές, καθώς είναι το επίσημο εργαλείο ανάπτυξης που παρέχεται από την Google. Παρόλο που παρέχει ευελιξία και ισχυρά εργαλεία, απαιτεί ένα μέτριο έως υψηλό επίπεδο γνώσης και εξειδίκευσης. Επίσης, η διαδικασία σχεδιασμού μέσω XML μπορεί να είναι πιο χρονοβόρα και λιγότερο φιλική για μη προγραμματιστές σε σύγκριση με άλλα εργαλεία σχεδιασμού.

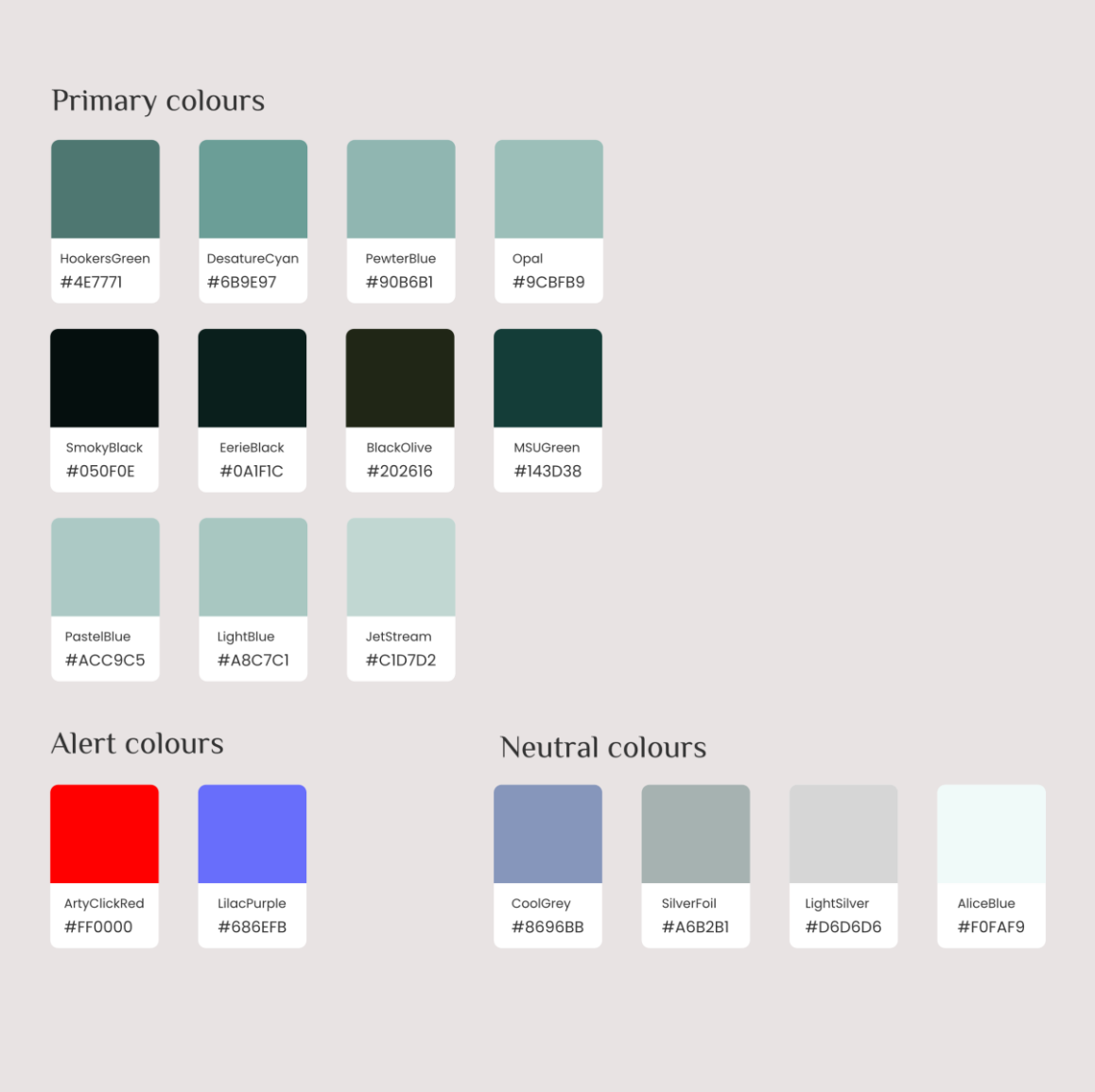
4.2.6 Επιλογή προγράμματος σχεδιασμού

Μετά από ενδελεχή ανάλυση των διαθέσιμων εργαλείων σχεδιασμού, επιλέξαμε το Figma ως την καλύτερη λύση για τον σχεδιασμό της διεπαφής χρήστη της εφαρμογής. Η πλατφόρμα αυτή διακρίθηκε για την ευκολία χρήσης της, επιτρέποντας ακόμη και σε λιγότερο έμπειρους σχεδιαστές να τη χρησιμοποιούν με άνεση, υποβοηθούμενοι από το πλούσιο υλικό που είναι διαθέσιμο στο διαδίκτυο, όπως plugins και tutorials. Επιπλέον, η δυνατότητα πρόσβασης από οποιαδήποτε συσκευή με σύνδεση στο διαδίκτυο το καθιστά εξαιρετικά ευέλικτο και λειτουργικό.

Ένα άλλο σημαντικό πλεονέκτημα είναι ότι η δωρεάν έκδοση του εργαλείου καλύπτει πλήρως τις ανάγκες μου, επιτρέποντας την επαναχρησιμοποίηση στοιχείων και υποστηρίζοντας έναν αποδοτικό και αποτελεσματικό σχεδιασμό. Σε σύγκριση με τα άλλα εργαλεία που εξετάστηκαν, το Figma προσέφερε την καλύτερη ισορροπία μεταξύ δυνατοτήτων και ευχρηστίας, καθιστώντας το την ιδανική επιλογή για το project αυτό.

4.3 Χρωματική παλέτα της εφαρμογής

Η χρωματική παλέτα αποτελεί βασικό στοιχείο της οπτικής ταυτότητας της εφαρμογής και επηρεάζει την εμπειρία χρήσης. Στην εφαρμογή μας, έχουμε υιοθετήσει την ίδια παλέτα χρωμάτων τόσο για το Light Theme όσο και για το Dark Theme, διασφαλίζοντας έτσι τη συνέπεια στην οπτική ταυτότητα ανεξάρτητα από την επιλεγμένη εμφάνιση.



Η παλέτα αυτή περιλαμβάνει χρώματα που είναι σχεδιασμένα για να λειτουργούν εξίσου καλά σε φωτεινό και σκοτεινό περιβάλλον, προσφέροντας καθαρότητα και αναγνωσιμότητα σε όλες τις συνθήκες. Αυτή η συνεπής χρωματική προσέγγιση ενισχύει την αισθητική συνοχή και την οπτική άνεση του χρήστη, ανεξαρτήτως του θέματος που επιλέγεται.

Εικόνα 1: Παλέτα για Light και Dark Mode

Τα χρώματα που επιλέξαμε θα χρησιμοποιηθούν για τη γραφική απεικόνιση των οθονών της εφαρμογής και θα καθοδηγήσουν την ανάπτυξη του κώδικα. Ωστόσο, υπάρχουν ορισμένα components στην εφαρμογή που δεν δέχονται παραμετροποιήσεις στα χρώματα, και σε αυτά τα σημεία χρησιμοποιούνται τα αυτόματα χρώματα που υπολογίζονται από την τελευταία έκδοση του Android, βάσει του φόντου της ταπετσαρίας του χρήστη.

4.4 Material Design και Jetpack Compose

Το Material Design [23] και το Jetpack Compose [3] είναι δύο θεμελιώδεις τεχνολογίες στην ανάπτυξη Android εφαρμογών. Το Material Design αποτελεί ένα καθιερωμένο σύστημα σχεδίασης από την Google, το οποίο παρέχει ένα σύνολο αρχών και κατευθυντήριων γραμμών για τη δημιουργία συνεπών, λειτουργικών και οπτικά ευχάριστων διεπαφών χρήστη (UI). Το Jetpack Compose, από την άλλη, είναι το νεότερο toolkit της Google που φέρνει έναν δηλωτικό τρόπο ανάπτυξης UI, επιτρέποντας την ευκολότερη και πιο αποδοτική δημιουργία διεπαφών, μειώνοντας ταυτόχρονα την πολυπλοκότητα του κώδικα.

4.4.1 Material Design

Το Material Design είναι ένα σύστημα σχεδίασης που στοχεύει στη δημιουργία συνεκτικών και προσιτών οπτικών εμπειριών. Βασίζεται σε θεμελιώδεις αρχές όπως η μετατροπή (motion), το υλικό (material), και η αλληλεπίδραση (material), οι οποίες διασφαλίζουν ότι οι εφαρμογές είναι όχι μόνο αισθητικά ευχάριστες, αλλά και λειτουργικά αποδοτικές.

Βασικά στοιχεία του περιλαμβάνουν:

- **Τυπογραφία**: Η επιλογή γραμματοσειρών και τοποθέτησης κειμένου παίζει σημαντικό ρόλο στην αναγνωσιμότητα και την εμπειρία του χρήστη.
- **Χρωματικές Παλέτες**: Το Material Design προτείνει τη χρήση συνδυασμών χρωμάτων που ενισχύουν τη συνεκτικότητα και την αισθητική των εφαρμογών.
- **Μεταφορές και Animations**: Ομαλές μεταβάσεις και animations βοηθούν στη δημιουργία πιο φυσικών και κατανοητών αλληλεπιδράσεων με τον χρήστη.

Η έκδοση Material 3 (γνωστή και ως Material You) προσφέρει βελτιωμένη προσαρμοστικότητα, επιτρέποντας στους χρήστες να προσαρμόζουν την εμφάνιση της εφαρμογής σύμφωνα με τις προσωπικές τους προτιμήσεις.

4.4.2 Jetpack Compose

Το Jetpack Compose φέρνει μια επαναστατική προσέγγιση στην ανάπτυξη UI. Αντί για την παραδοσιακή χρήση XML για τον ορισμό των διεπαφών χρήστη, το Compose επιτρέπει τη δημιουργία διεπαφής χρησιμοποιώντας κώδικα, καθιστώντας τη διαδικασία πιο απλή και πιο ευέλικτη.

Τα βασικά του πλεονεκτήματα περιλαμβάνουν:

- **Δηλωτικός Προγραμματισμός (Declarative Programming)**: Οι προγραμματιστές ορίζουν πώς πρέπει να φαίνεται και να συμπεριφέρεται το UI μέσω του κώδικα, καθιστώντας την ανάπτυξη πιο εκφραστική και κατανοητή.

- **Ενσωμάτωση με Material Design**: Το Jetpack Compose υποστηρίζει πλήρως το Material Design, επιτρέποντας την εύκολη δημιουργία σύγχρονων και συνεπών διεπαφών χρήστη, οι οποίες ανταποκρίνονται στις τελευταίες τάσεις σχεδιασμού της Google.
- **Επεκτασιμότητα και Ευκολία Συντήρησης**: Το Compose διευκολύνει την αναβάθμιση και τη συντήρηση του κώδικα, καθώς όλα τα UI components είναι δηλωτικά και εύκολα επαναχρησιμοποιήσιμα.

Ο συνδυασμός του Material Design με το Jetpack Compose προσφέρει στους προγραμματιστές τα εργαλεία για να δημιουργήσουν σύγχρονες, προσαρμοστικές και συνεπείς εφαρμογές που ακολουθούν τις τελευταίες τάσεις στο σχεδιασμό διεπαφών χρήστη. Η χρήση αυτών των τεχνολογιών όχι μόνο βελτιώνει την αποδοτικότητα της ανάπτυξης, αλλά εξασφαλίζει και τη δημιουργία εμπειριών χρήστη που είναι οπτικά ελκυστικές και πλήρως λειτουργικές σε όλες τις συσκευές και τα περιβάλλοντα Android.

5. Υλοποίηση Εφαρμογής

5.1 Εισαγωγή

Για την ανάπτυξη της εφαρμογής, είναι αναγκαίο να επιλέξουμε πρώτα την πλατφόρμα όπου θα γίνει η ανάπτυξη του κώδικα, διότι από αυτή εξαρτάται συνήθως και η επιλογή της γλώσσας προγραμματισμού. Υπάρχουν πολλές επιλογές πλατφόρμας για τη δημιουργία κινητών εφαρμογών, με ορισμένες από αυτές να υποστηρίζουν την ανάπτυξη εφαρμογών που λειτουργούν τόσο σε Android όσο και σε iOS, μέσω μιας κοινής βάσης κώδικα. Ωστόσο, σε αυτή την εργασία, η εστίαση θα δοθεί αποκλειστικά στην υλοποίηση μιας εφαρμογής για την πλατφόρμα Android, παρακάμπτοντας εργαλεία που προσφέρουν δια λειτουργικότητα με άλλες πλατφόρμες.

Μια σύντομη αναζήτηση στα επίσημα έγγραφα της Google αποκαλύπτει ότι το Android Studio είναι το προτεινόμενο εργαλείο [14], με την Kotlin [2] να θεωρείται η πλέον προτιμώμενη γλώσσα προγραμματισμού.

5.2 Δομή και Σχεδίαση του Συστήματος

Η δομή και η σχεδίαση του συστήματος παίζουν έναν κρίσιμο ρόλο στην ανάπτυξη εφαρμογών για Android. Ο τρόπος με τον οποίο οργανώνεται ο κώδικας και οι δομές του μπορεί να επηρεάσει σημαντικά τη συντηρησιμότητα, την επεκτασιμότητα και τη συνολική αποδοτικότητα της εφαρμογής. Η σωστή αρχιτεκτονική εξασφαλίζει ότι ο κώδικας παραμένει καθαρός, ευανάγνωστος και εύκολα διαχειρίσιμος.

5.2.1 Αρχές Σχεδιασμού

Για την ανάπτυξη της εφαρμογής, θα ακολουθηθούν οι αρχές SOLID, όπως αυτές προτάθηκαν από τον R. C. Martin. Οι αρχές αυτές είναι θεμελιώδεις για τη διασφάλιση της ποιότητας του κώδικα και περιλαμβάνουν:

1. **Single Responsibility Principle (SRP)**: Κάθε κλάση (class) αναλαμβάνει μια μοναδική ευθύνη, επιτρέποντας έτσι την ευκολότερη διαχείριση και την πιο αποτελεσματική τροποποίηση του κώδικα.
2. **Open – Closed Principle (OCP):** Οι κλάσεις (classes) σχεδιάζονται έτσι ώστε να επιτρέπουν την επέκταση με νέες λειτουργίες χωρίς να απαιτείται η τροποποίηση του υπάρχοντος κώδικα, διατηρώντας την σταθερότητα του συστήματος.
3. **Liskov Substitution Principle (LSP)**: Οι υποκλάσεις (subclasses) πρέπει να μπορούν να αντικαθιστούν τις υπερκλάσεις (superclasses) τους χωρίς να

προκαλούν ασυμβατότητες ή σφάλματα, διασφαλίζοντας έτσι τη συνοχή του συστήματος.

4. **Interface Segregation (ISP)**: Οι διεπαφές (interfaces) πρέπει να είναι διακριτά και να περιλαμβάνουν μόνο τις μεθόδους που είναι σχετικές με την κλάση που τα υλοποιεί, αποφεύγοντας περιττές εξαρτήσεις (dependencies).
5. **Dependency Inversion Principle (DIP)**: Οι υψηλού επιπέδου κλάσεις (high – level classes) δεν πρέπει να εξαρτώνται από χαμηλού επιπέδου κλάσεις (low – level classes), αλλά και οι δύο να βασίζονται σε αφαιρετικά στρώματα, όπως interfaces, για να ενισχύεται η επαναχρησιμοποίηση και η ευελιξία του κώδικα [24].

5.2.2 Αρχιτεκτονικές Προσεγγίσεις

Για την οργάνωση και τη διαχείριση του κώδικα, υπάρχουν διάφορες αρχιτεκτονικές προσεγγίσεις που μπορούν να χρησιμοποιηθούν. Δύο από τις πιο δημοφιλείς προσεγγίσεις είναι η Model – View – ViewModel (MVVM) και η Model – View – Controller (MVC).

1. **Model – View – ViewModel (MVVM):**
   - **Model:** Διαχειρίζεται τα δεδομένα και την επιχειρησιακή λογική.
   - **View:** Παρουσιάζει τα δεδομένα και αλληλοεπιδρά με τον χρήστη.
   - **ViewModel:** Ενεργεί ως ενδιάμεσος μεταξύ του Model και του View, εξασφαλίζοντας την επικοινωνία και την αλληλεπίδρασή τους.
2. **Model-View-Controller (MVC):**
   - **Model:** Ασχολείται με τη διαχείριση των δεδομένων και της επιχειρησιακής λογικής της εφαρμογής.
   - **View:** Εστιάζει στην παρουσίαση των δεδομένων και στη διαχείριση των γραφικών στοιχείων του χρήστη.
   - **Controller:** Ενώνει το Model και το View, διαχειρίζοντας τις εισερχόμενες ενέργειες του χρήστη και τα αιτήματά του.

5.2.3 Προηγμένες Τεχνικές και Βέλτιστες Πρακτικές Σχεδίασης

Στο παρόν έργο, θα εφαρμοστεί η αρχιτεκτονική MVVM, σε συνδυασμό με τις αρχές του Clean Code, την τεχνική Dependency Injection (DI) και την προσέγγιση Clean Architecture. Ο συνδυασμός αυτών των τεχνικών εξασφαλίζει ότι ο κώδικας της εφαρμογής θα είναι καλά δομημένος, ευανάγνωστος, συντηρήσιμος και επεκτάσιμος.

Clean Code

Το Clean Code αποτελεί μια φιλοσοφία προγραμματισμού που επικεντρώνεται στην παραγωγή κώδικα ο οποίος είναι ευανάγνωστος, κατανοητός και εύκολα συντηρήσιμος. Οι αρχές του Clean Code, όπως αναπτύχθηκαν, προωθούν την ευκολία

κατανόησης του κώδικα, όχι μόνο από τον αρχικό δημιουργό του, αλλά και από άλλους προγραμματιστές που ενδέχεται να εργαστούν σε αυτό το έργο μελλοντικά.

Ορισμένες από τις βασικές αρχές του Clean Code περιλαμβάνουν:

- **Κατανοητή Ονομασία**: Η ονομασία των μεταβλητών, των συναρτήσεων και των κλάσεων θα πρέπει να είναι περιγραφική και να αντικατοπτρίζει τη λειτουργικότητά τους. Αυτό επιτρέπει στον αναγνώστη να κατανοήσει τον κώδικα χωρίς να χρειάζεται να εξετάσει τις λεπτομέρειες της υλοποίησης.
- **Μικρές και Συνεκτικές Συναρτήσεις**: Κάθε συνάρτηση πρέπει να εκτελεί μία και μόνο λειτουργία και να είναι όσο το δυνατόν πιο σύντομη. Οι μικρές συναρτήσεις είναι ευκολότερες στην κατανόηση, την αποσφαλμάτωση και τη συντήρηση.
- **Μείωση Πολυπλοκότητας:** Η πολυπλοκότητα του κώδικα πρέπει να περιορίζεται στο ελάχιστο. Η χρήση καλά σχεδιασμένων δομών δεδομένων και αλγορίθμων συμβάλλει στην ευκολότερη κατανόηση και στη βελτίωση της απόδοσης του συστήματος.
- **Σχόλια και Τεκμηρίωση:** Η χρήση σχολίων πρέπει να γίνεται με φειδώ και να περιορίζεται σε σημεία όπου πραγματικά χρειάζεται να εξηγηθεί κάτι πολύπλοκο ή μη προφανές. Η τεκμηρίωση του κώδικα είναι κρίσιμη για την κατανόηση από άλλους προγραμματιστές και για τη διατήρηση της ποιότητας κατά τη διάρκεια της ανάπτυξης.
- **Δοκιμές:** Οι συχνές δοκιμές του κώδικα (Unit Testing) διασφαλίζουν τη σωστή λειτουργία του και μειώνουν τις πιθανότητες εμφάνισης σφαλμάτων κατά την εξέλιξη του έργου. Η τήρηση αυτών των αρχών συμβάλλει στη δημιουργία ενός κώδικα που είναι εύκολα επεκτάσιμος, σταθερός καθώς και αξιόπιστος.

Dependency Injection (DI)

Το Dependency Injection είναι μια προγραμματιστική τεχνική που μειώνει τις σκληρές συνδέσεις (hardcoded connections) μεταξύ των κλάσεων, επιτρέποντας την εισαγωγή των εξαρτήσεων από εξωτερικές πηγές, αντί να δημιουργούνται εντός της κλάσης. Αυτή η προσέγγιση προωθεί την επαναχρησιμοποίηση του κώδικα και ενισχύει τη συντηρησιμότητα, διευκολύνοντας την αντικατάσταση ή την ενημέρωση των εξαρτήσεων χωρίς να απαιτείται αλλαγή στον κύριο κώδικα της εφαρμογής.

Υπάρχουν τρεις κύριοι τύποι DI [25]:

1. **Constructor Injection**: Οι εξαρτήσεις παρέχονται μέσω του κατασκευαστή(constructor) της κλάσης κατά τη δημιουργία του αντικειμένου.
2. **Setter Injection**: Οι εξαρτήσεις εισάγονται μέσω setter μεθόδων, επιτρέποντας την αλλαγή τους μετά τη δημιουργία του αντικειμένου.

3. **Method Injection**: Οι εξαρτήσεις παρέχονται κατά τη διάρκεια της εκτέλεσης μιας συγκεκριμένης μεθόδου, επιτρέποντας πιο ευέλικτη διαχείριση της εξάρτησης.

Το DI συνδυάζεται συχνά με τη χρήση IoC (Inversion of Control) Containers, όπως η Hilt [26] και η Koin [27] για την Kotlin, τα οποία διαχειρίζονται και παρέχουν τις απαιτούμενες εξαρτήσεις με έναν αυτοματοποιημένο τρόπο, αυξάνοντας την απόδοση και μειώνοντας τα λάθη.

Clean Architecture

Η Clean Architecture [28], αποτελεί μια αρχιτεκτονική προσέγγιση που επιτρέπει τη δημιουργία εφαρμογών με χαμηλή ζεύξη (coupling) και ανεξαρτησία από τεχνικές λεπτομέρειες, όπως οι βάσεις δεδομένων, τα frameworks, και άλλες εξωτερικές εξαρτήσεις. Αυτό επιτρέπει στην εφαρμογή να είναι πιο εύκολα συντηρήσιμη και ευέλικτη στις αλλαγές, κάτι που είναι ζωτικής σημασίας για τη μακροπρόθεσμη επιτυχία του λογισμικού.

Η βασική ιδέα πίσω από την Clean Architecture είναι ο σαφής διαχωρισμός των ευθυνών σε διαφορετικά επίπεδα της εφαρμογής. Αυτή η προσέγγιση διασφαλίζει ότι τα εξωτερικά επίπεδα εξαρτώνται από τα εσωτερικά, αλλά όχι το αντίστροφο, διατηρώντας έτσι την ανεξαρτησία της επιχειρησιακής λογικής από τα εξωτερικά συστήματα και τεχνολογίες.

Στην εφαρμογή μας, χρησιμοποιούνται τρία κύρια επίπεδα:

1. **Domain Layer:** Το Domain Layer περιγράφει τι κάνει η εφαρμογή. Περιλαμβάνει τα μοντέλα (models) που αντιπροσωπεύουν τα αντικείμενα της εφαρμογής, τα interfaces των αποθετηρίων (repositories) που καθορίζουν τις λειτουργίες για τη διαχείριση των δεδομένων, και τις περιπτώσεις χρήσης (use cases) που περιγράφουν την επιχειρησιακή λογική της εφαρμογής.
2. **Data Layer:** Το Data Layer ασχολείται με την υλοποίηση των αποθετηρίων και τη διαχείριση των εξωτερικών πηγών δεδομένων, όπως οι βάσεις δεδομένων και τα APIs. Εδώ, οι εξωτερικές πηγές δεδομένων και οι οντότητες (entities) που αντιπροσωπεύουν τα δεδομένα της εφαρμογής στον εξωτερικό κόσμο υλοποιούνται και χαρτογραφούνται στα αντίστοιχα domain models.
3. **Presentation Layer**: Το Presentation Layer είναι υπεύθυνο για την αλληλεπίδραση της εφαρμογής με τον χρήστη ή άλλες εξωτερικές διεπαφές. Αυτό το επίπεδο περιλαμβάνει τις οθόνες της εφαρμογής και τα ViewModels που διαχειρίζονται την κατάσταση των οθονών, καθώς και τον τρόπο με τον οποίο η εφαρμογή παρουσιάζει και χειρίζεται τα δεδομένα που παρέχονται από το Domain Layer.

Στο δικό μας έργο, η εφαρμογή της Clean Architecture σε συνδυασμό με το MVVM, το Clean Code και το Dependency Injection, εξασφαλίζει ότι η εφαρμογή θα είναι επεκτάσιμη, ευέλικτη και εύκολα συντηρήσιμη. Η καθαρή διαχωριστική γραμμή μεταξύ των διαφόρων επιπέδων της εφαρμογής θα μας επιτρέψει να κάνουμε αλλαγές με μεγαλύτερη ευκολία, διατηρώντας την σταθερότητα και την αξιοπιστία του συστήματος. Ο συνδυασμός αυτών των προηγμένων τεχνικών σχεδιασμού καθιστά την εφαρμογή όχι μόνο τεχνολογικά προηγμένη, αλλά και πρακτικά υλοποιήσιμη, έτοιμη να αντιμετωπίσει μελλοντικές προκλήσεις και αναβαθμίσεις.

5.3 Διαχείριση Κώδικα και Έκδοσης

Η διαχείριση του κώδικα (code management) και των εκδόσεων (version control) είναι ένα από τα πιο κρίσιμα στοιχεία στην ανάπτυξη του λογισμικού. Η χρήση ενός συστήματος ελέγχου, όπως το Git, παρέχει μια σειρά από πλεονεκτήματα που διευκολύνουν τη διαχείριση και την παρακολούθηση των αλλαγών στον κώδικα. Μέσω ενός συστήματος version control, μπορούμε να καταγράψουμε όλες τις τροποποιήσεις, επιτρέποντας την αναστροφή σε προηγούμενες εκδόσεις σε περίπτωση προβλημάτων, καθώς και την παρακολούθηση της εξέλιξης του κώδικα με σαφήνεια και διαφάνεια.

Στο συγκεκριμένο project, χρησιμοποιήθηκε το GitHub [29] ως πλατφόρμα για τη φιλοξενία και τη διαχείριση του κώδικα. Το GitHub προσφέρει όχι μόνο αποθήκευση και διαχείριση του κώδικα , αλλά και εργαλεία συνεργασίας που διευκολύνουν τη διαχείριση κλάδων (branches), όπου διαφορετικά χαρακτηριστικά (features) μπορούν να αναπτυχθούν παράλληλα σε ξεχωριστά branches και να ενσωματωθούν (merge) στο κύριο branch όταν ολοκληρωθούν. Αυτός ο τρόπος εργασίας μειώνει τις πιθανότητες συγκρούσεων στον κώδικα και διευκολύνει την ανάπτυξη, ακόμα και όταν η ομάδα αποτελείται από έναν μόνο προγραμματιστή.

Παρόλο που το έργο αυτό θα μπορούσε να υλοποιηθεί χωρίς σύστημα ελέγχου εκδόσεων, δεδομένου ότι υπάρχει μόνο ένας προγραμματιστής, τα πλεονεκτήματα της καταγραφής ιστορικού και της δυνατότητας επιστροφής σε προηγούμενες εκδόσεις καθιστούν το version control απαραίτητο εργαλείο. Αυτή η μεθοδολογία διασφαλίζει την ποιότητα και τη σταθερότητα της εφαρμογής, προσφέροντας μια οργανωμένη και ασφαλή διαδικασία ανάπτυξης.

5.4 Διαχείριση Ταυτοποίησης Χρηστών

Η διαχείριση της ταυτοποίησης των χρηστών αποτελεί ένα από τα πιο βασικά στοιχεία σε μια εφαρμογή, ιδιαίτερα όσον αφορά την προστασία των προσωπικών δεδομένων και τη διασφάλιση της πρόσβασης μόνο σε εξουσιοδοτημένους χρήστες. Στο συγκεκριμένο project, η εγγραφή και η σύνδεση των χρηστών πραγματοποιούνται μέσω του Firebase Authentication [30], χρησιμοποιώντας email και password.

Με τη χρήση του Firebase Authentication, οι χρήστες μπορούν εύκολα να δημιουργήσουν έναν λογαριασμό ή να συνδεθούν σε έναν ήδη υπάρχοντα λογαριασμό. Η διαδικασία είναι η εξής: ο χρήστης εισάγει το email και τον κωδικό πρόσβασής του, τα οποία αποστέλλονται στο Firebase για επαλήθευση. Αν τα στοιχεία είναι έγκυρα, ο χρήστης είτε εγγράφεται είτε συνδέεται στον λογαριασμό του και αποκτά πρόσβαση σε τμήματα της εφαρμογής που απαιτούν ταυτοποίηση.

Παρακάτω παρατίθεται ένα παράδειγμα κώδικα που δείχνει πώς υλοποιούνται αυτές οι λειτουργίες:

```
class UserAccountUseCase(private val activity: WeakReference<Activity>) :
    FirebaseAccountRepository {
    private val _authState =
        MutableStateFlow<GenericFlowState<FirebaseUser>>(FlowLoading())
    override val authState = _authState.asStateFlow()
    private val _accountUseCaseEvents: MutableStateFlow<UserAccountUseCaseEvents?> =
        MutableStateFlow( value: null)
    override val accountUseCaseEvents = _accountUseCaseEvents.asStateFlow()
    private var continuationFlow: ContinuationFlow? = null

    private lateinit var oneTapClient: SignInClient

    init {
        resetToInitialState()
        activity.get()?.let {
            oneTapClient = Identity.getSignInClient(it)
        }
    }

    override fun createWithEmailAndPassword(username: String, password: String) {
        if (username.isBlank() || password.isBlank()) return
        _authState.setLoading()
        Firebase.auth.createUserWithEmailAndPassword(username, password)
            .addOnCompleteListener { task →
                handleFirebaseResponse(task)
            }
    }

    override fun signInWithEmailAndPassword(username: String, password: String) {
        if (username.isBlank() || password.isBlank()) return
        _authState.setLoading()
        val credential = EmailAuthProvider
            .getCredential(username, password)

        if (_accountUseCaseEvents.value is UserAccountUseCaseEvents.ReAuthenticate) {
            reAuthenticate(credential)
        } else {
            Firebase.auth.signInWithCredential(credential)
                .addOnCompleteListener { task →
                    handleFirebaseResponse(task)
                }
        }
    }
```

Εικόνα 2: Παράδειγμα κώδικα για την δημιουργία λογαριασμού και το login

- Η μέθοδος **createWithEmailAndPassword** είναι υπεύθυνη για τη δημιουργία νέου λογαριασμού χρήστη. Όταν ο χρήστης εισάγει το email και τον κωδικό πρόσβασής του στη φόρμα εγγραφής, τα δεδομένα αυτά αποστέλλονται στη μέθοδο. Το Firebase τότε δημιουργεί έναν νέο λογαριασμό και αποθηκεύει τα

στοιχεία του χρήστη. Εάν η δημιουργία του λογαριασμού είναι επιτυχής, ο χρήστης μπορεί να συνδεθεί στον λογαριασμό του.

Στον κώδικα, πριν ξεκινήσει η διαδικασία, ελέγχεται αν τα πεδία του username και του password είναι κενά, και αν ναι, η διαδικασία σταματάει. Αφού γίνει η κλήση στο Firebase για δημιουργία χρήστη, ο κώδικας παρακολουθεί την ολοκλήρωση της ενέργειας με τον listener addOnCompleteListener, ο οποίος χειρίζεται την απάντηση (response) του Firebase, δηλαδή αν η ενέργεια ήταν επιτυχής ή αν προέκυψε κάποιο σφάλμα.

- Η μέθοδος **signInWithEmailAndPassword** διαχειρίζεται τη σύνδεση ενός χρήστη που έχει ήδη λογαριασμό. Ο χρήστης εισάγει το email και τον κωδικό πρόσβασής του, τα οποία στη συνέχεια ελέγχονται μέσω της μεθόδου. Αν τα στοιχεία είναι σωστά, η μέθοδος επιστρέφει ένα token ταυτοποίησης, το οποίο χρησιμοποιείται από την εφαρμογή για να επιτρέψει την πρόσβαση του χρήστη στις προστατευμένες περιοχές.

  Στον κώδικα, όπως και στην εγγραφή, πρώτα ελέγχεται αν τα πεδία του username και του password είναι κενά. Ακολούθως, δημιουργείται ένα αντικείμενο credential που περιέχει τα στοιχεία ταυτοποίησης του χρήστη. Το credential αυτό στη συνέχεια αποστέλλεται στο Firebase για επαλήθευση. Εάν τα στοιχεία είναι έγκυρα, ο χρήστης μπορεί να συνδεθεί στην εφαρμογή. Όπως και με την εγγραφή, η απάντηση του Firebase παρακολουθείται και διαχειρίζεται μέσω του listener addOnCompleteListener.

Η υλοποίηση αυτή προσφέρει μια αξιόπιστη λύση για τη διαχείριση ταυτοποίησης χρηστών, καθιστώντας την εφαρμογή πιο ασφαλή και προσβάσιμη για τους χρήστες της, χωρίς να επιβαρύνει τον προγραμματιστή με την υλοποίηση περίπλοκων μηχανισμών ελέγχου ταυτότητας.

5.5 Αποθήκευση Δεδομένων του Χρήστη

Η αποθήκευση δεδομένων των χρηστών αποτελεί ένα θεμελιώδες στοιχείο για τη λειτουργικότητα της εφαρμογής που απαιτεί διατήρηση και διαχείριση πληροφοριών. Αποφασίστηκε να χρησιμοποιηθεί το Firebase Realtime Database [31] που είναι μια cloud – based NoSQL βάση για την αποθήκευση, επεξεργασία και ανάκτηση δεδομένων χρηστών σε πραγματικό χρόνο.

Κάθε χρήστης έχει έναν μοναδικό χώρο αποθήκευσης, στον οποίο μπορούν να αποθηκευτούν διάφορες πληροφορίες, όπως τα προσωπικά του στοιχεία, τα φάρμακα που λαμβάνει, οι υπενθυμίσεις, και άλλα συναφή δεδομένα.

Παρακάτω παρουσιάζεται ένα παράδειγμα κώδικα για το πώς αποθηκεύουμε και ανακτούμε δεδομένα ενός χρήστη:

```
class FirebaseRepository {

    private val database = Firebase.database
    private val userId: String?
        get() = FirebaseAuth.getInstance().currentUser?.uid


    //medication
    suspend fun saveMedication(medication: Medication) {
        userId?.let {
            val myRef = database.getReference( path: "users").child(it).child( pathString: "medications")
            val key = myRef.push().key // Generate a unique key
            key?.let { id ->
                val medicationWithId = medication.copy(id = id)
                myRef.child(id).setValue(medicationWithId).await() // Save with the generated id
            }
        }
    }

    fun generateMedicationId(): String {
        return database.getReference( path: "dummy").push().key ?: UUID.randomUUID().toString()
    }

    suspend fun getMedicationById(medicationId: String): Medication? {
        return userId?.let {
            val myRef =
                database.getReference( path: "users").child(it).child( pathString: "medications").child(medicationId)
            myRef.get().await().getValue(Medication::class.java)
        }
    }

    fun getMedicationsFlow(): Flow<List<Medication>> = callbackFlow {
        val myRef = database.getReference( path: "users").child(userId!!).child( pathString: "medications")
        val listener = object : ValueEventListener {
            override fun onDataChange(snapshot: DataSnapshot) {
                val medications = mutableListOf<Medication>()
                snapshot.children.forEach { dataSnapshot ->
                    dataSnapshot.getValue(Medication::class.java)?.let { medication ->
                        medications.add(medication)
                    }
                }
                trySend(medications)
            }

            override fun onCancelled(error: DatabaseError) {
                close(error.toException())
            }
        }
        myRef.addValueEventListener(listener)
        awaitClose { myRef.removeEventListener(listener) }
    }
```

Εικόνα 3: Παράδειγμα κώδικα για αποθήκευση και ανάκτηση δεδομένων μέσω Firebase Realtime Database

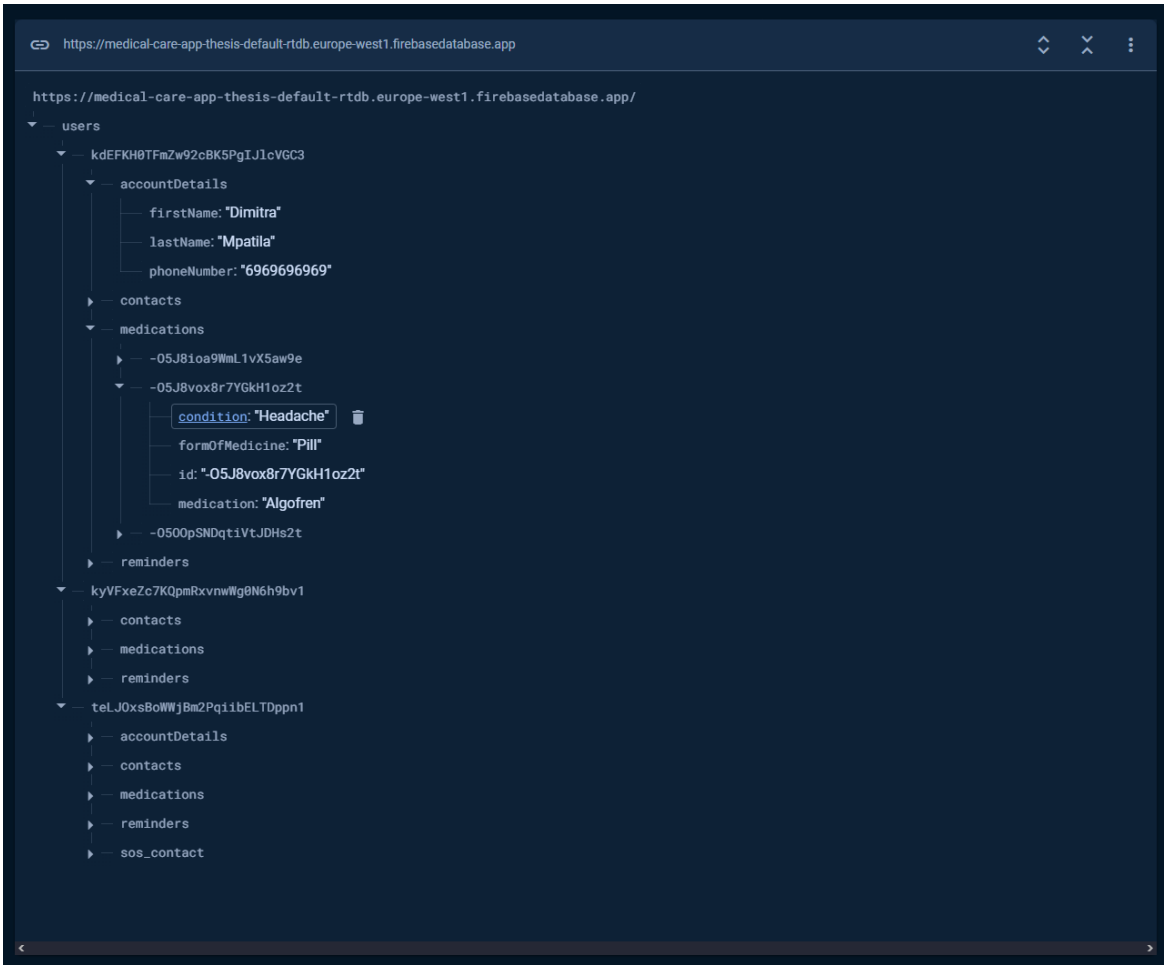
Παρακάτω παρουσιάζεται ένα παράδειγμα κώδικα για το πώς αποθηκεύουμε και ανακτούμε δεδομένα ενός χρήστη:

- **Αποθήκευση Δεδομένων**: Η μέθοδος **saveMedication** επιτρέπει την αποθήκευση πληροφοριών σχετικά με τη φαρμακευτική αγωγή ενός χρήστη. Η μέθοδος αυτή δημιουργεί μια μοναδική καταχώριση για κάθε φάρμακο που αποθηκεύεται στο δέντρο της Realtime, κάτω από το συγκεκριμένο χρήστη. Συγκεκριμένα, η μέθοδος παράγει ένα μοναδικό κλειδί (key) χρησιμοποιώντας τη λειτουργία **push(),** το οποίο στη συνέχεια χρησιμοποιείται για να αποθηκευτεί το φάρμακο στη βάση δεδομένων μαζί με αυτό το κλειδί. Η αποθήκευση ολοκληρώνεται ασύγχρονα,

εξασφαλίζοντας ότι η εφαρμογή δεν θα μπλοκάρει κατά τη διάρκεια της διαδικασίας.

- **Ανάκτηση Δεδομένων**: Η μέθοδος **getMedicationById** επιτρέπει την ανάκτηση μιας συγκεκριμένης καταχώρισης φαρμάκου από τη βάση δεδομένων, χρησιμοποιώντας το μοναδικό ID που δημιουργήθηκε κατά την αποθήκευσή του. Η μέθοδος συνδέεται με το αντίστοιχο μονοπάτι (path) στη βάση δεδομένων και χρησιμοποιεί την εντολή get() για να ανακτήσει τα δεδομένα. Η ανάκτηση γίνεται επίσης ασύγχρονα, διασφαλίζοντας ότι η λειτουργία ολοκληρώνεται χωρίς να επηρεάζεται η εμπειρία του χρήστη.
- **Παρακολούθηση Δεδομένων σε Πραγματικό Χρόνο**: Η μέθοδος **getMedicationsFlow** δημιουργεί έναν συνεχή ροή δεδομένων (Flow) που παρακολουθεί τις αλλαγές στις καταχωρίσεις φαρμάκων σε πραγματικό χρόνο. Χρησιμοποιώντας έναν ValueEventListener, η μέθοδος παρακολουθεί όλες τις αλλαγές που πραγματοποιούνται στα φάρμακα ενός χρήστη. Κάθε φορά που προστίθεται ή αλλάζει μια καταχώριση, ο listener ενεργοποιείται και ενημερώνει το Flow, επιτρέποντας στην εφαρμογή να παραμένει συγχρονισμένη με τη βάση δεδομένων. Αυτό διασφαλίζει ότι οι πληροφορίες της εφαρμογής είναι πάντα ενημερωμένες και συγχρονισμένες με τα δεδομένα του χρήστη στη βάση δεδομένων.

Στην παρακάτω εικόνα παρουσιάζεται η δομή της Realtime, όπου φαίνεται πώς οργανώνονται τα δεδομένα για κάθε χρήστη. Κάθε χρήστης έχει ένα μοναδικό αναγνωριστικό (UID) κάτω από το οποίο αποθηκεύονται διάφορες πληροφορίες όπως τα προσωπικά στοιχεία, τα φάρμακα, οι υπενθυμίσεις κ.λπ.



Εικόνα 4: Δομή δεδομένων στη Firebase Realtime Database

Αυτή η προσέγγιση επιτρέπει τη διαχείριση δεδομένων χρηστών με αποτελεσματικό και κλιμακούμενο τρόπο, διασφαλίζοντας ότι η εφαρμογή μπορεί να χειριστεί μεγάλο όγκο δεδομένων και πολλούς χρήστες, διατηρώντας παράλληλα υψηλή απόδοση και διαθεσιμότητα.

6. Η εφαρμογή Vitalis

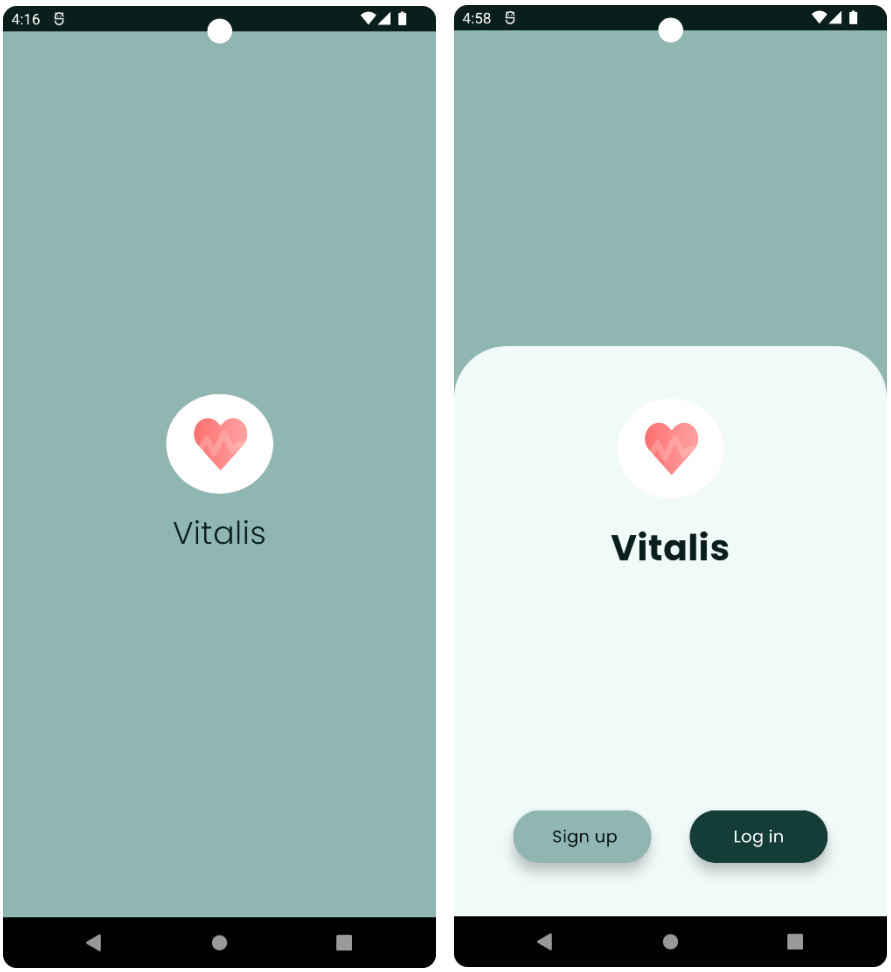
6.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστεί η λειτουργικότητα της εφαρμογής Vitalis και θα αναλυθούν οι δυνατότητες που παρέχονται στον χρήστη. Η παρουσίαση αυτή θα συνοδεύεται από στιγμιότυπα οθόνης, προκειμένου να διευκολυνθεί η κατανόηση των λειτουργιών της εφαρμογής. Τα στιγμιότυπα αυτά λειτουργούν επίσης και ως ένας χρήσιμος οδηγός χρήσης (user manual) για τους μελλοντικούς χρήστες.

6.2 Οθόνη Εκκίνησης και Οθόνη Καλωσορίσματος

Η πρώτη οθόνη που εμφανίζεται κατά την εκκίνηση της εφαρμογής είναι η οθόνη εκκίνησης (splash screen). Αυτή η οθόνη προβάλλει το λογότυπο της εφαρμογής, συνοδευόμενο από ένα ευχάριστο χρώμα φόντου, το οποίο προσδίδει μια πρώτη γεύση της αισθητικής της εφαρμογής. Η διάρκεια εμφάνισης της οθόνης είναι σύντομη, προετοιμάζοντας τον χρήστη για τη μετάβαση στην κύρια λειτουργικότητα της εφαρμογής.

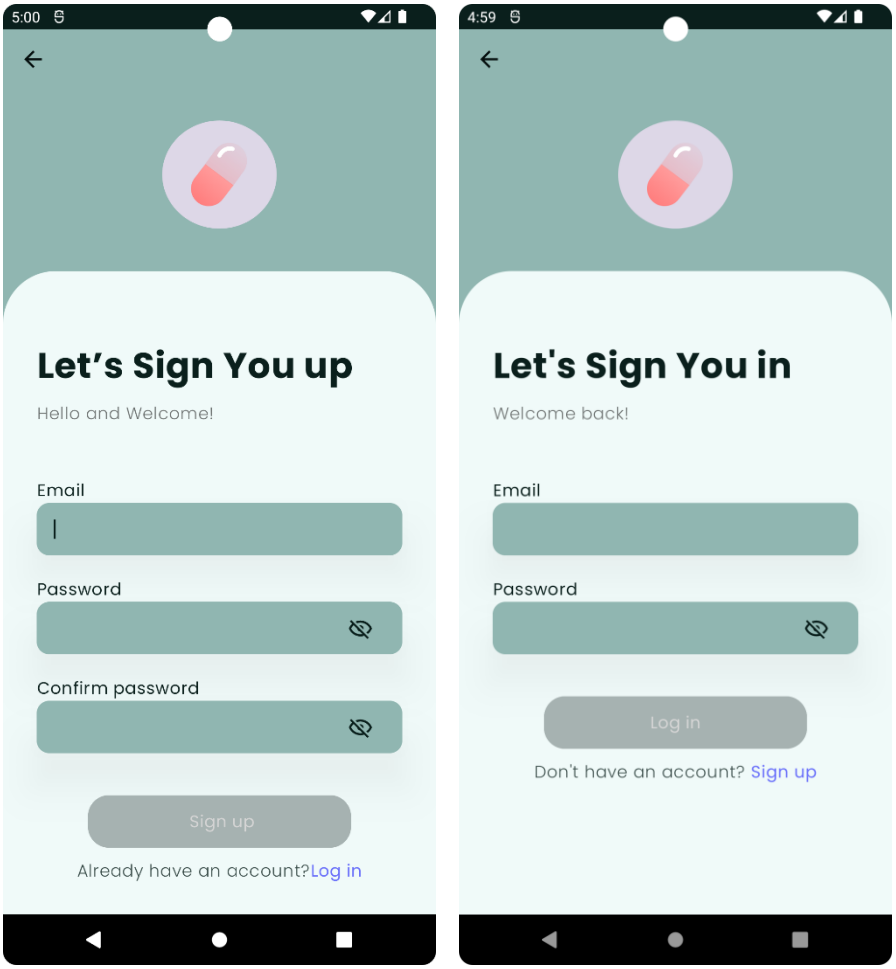
Ακολουθεί η οθόνη καλωσορίσματος, η οποία προσφέρει δύο βασικές επιλογές: "Sign Up" για τους νέους χρήστες που επιθυμούν να δημιουργήσουν λογαριασμό και "Log In" για τους υπάρχοντες χρήστες. Η οπτική συνέπεια μεταξύ της οθόνης εκκίνησης και της οθόνης καλωσορίσματος εξασφαλίζει μια αρμονική και φιλική προς τον χρήστη εμπειρία.



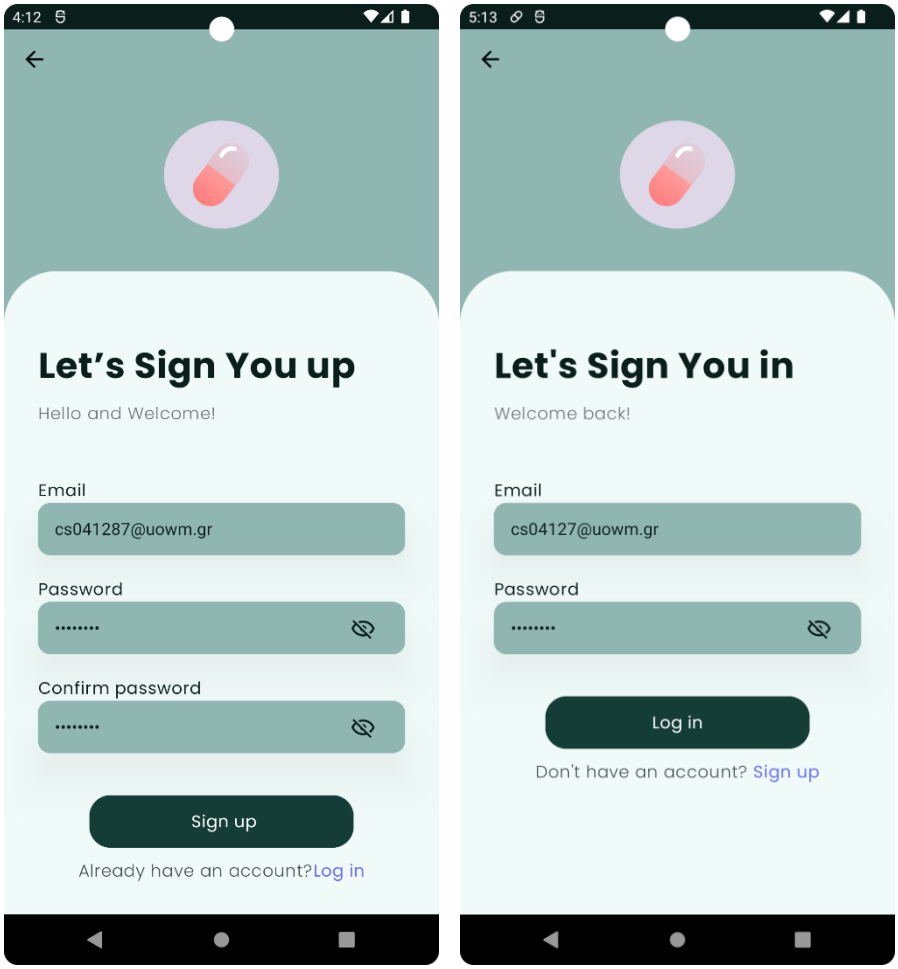
Εικόνα 5: Οθόνη Εκκίνησης (αριστερά) – Οθόνη Καλωσορίσματος (δεξιά)

6.3 Οθόνη Σύνδεσης και Εγγραφής

Στην οθόνη εγγραφής, ο χρήστης καλείται να συμπληρώσει τα πεδία του email, του κωδικού πρόσβασης, και της επιβεβαίωσης κωδικού. Αφού συμπληρωθούν όλα τα πεδία σωστά, το κουμπί "Sign up " ενεργοποιείται, επιτρέποντας στον χρήστη να ολοκληρώσει την εγγραφή του. Στην οθόνη σύνδεσης, ο χρήστης εισάγει το email και τον κωδικό πρόσβασης του. Εάν τα στοιχεία εισαχθούν σωστά, το κουμπί "Login" ενεργοποιείται, δίνοντας τη δυνατότητα στον χρήστη να εισέλθει στον λογαριασμό του.

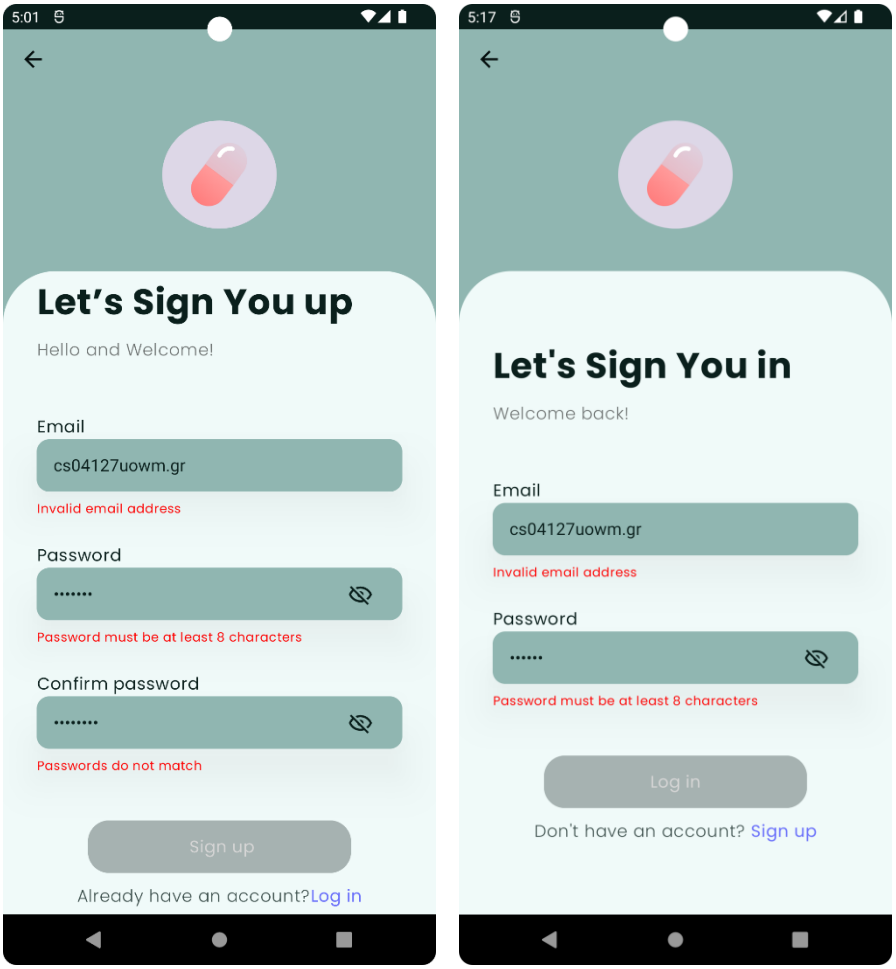


Εικόνα 6: Οθόνες Εγγραφής (αριστερά) και Σύνδεσης (δεξιά) χωρίς συμπληρωμένα στοιχεία



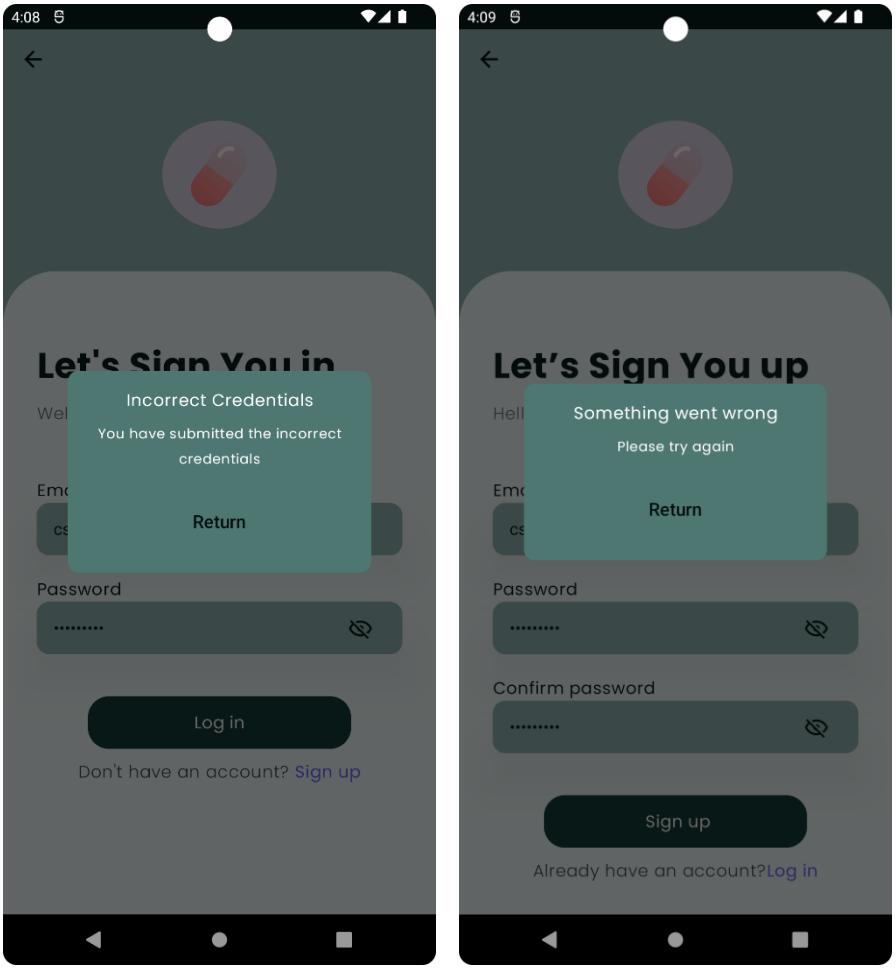
Εικόνα 7: Οθόνες Εγγραφής (αριστερά) και Σύνδεσης (δεξιά) με συμπληρωμένα στοιχεία

Όταν τα πεδία δεν συμπληρωθούν σωστά, εμφανίζονται μηνύματα λάθους κάτω από τα αντίστοιχα πεδία, τόσο στην οθόνη εγγραφής όσο και στην οθόνη σύνδεσης. Για παράδειγμα, εάν το email δεν ακολουθεί το σωστό πρότυπο (regex) ή εάν ο κωδικός πρόσβασης έχει λιγότερους από 8 χαρακτήρες, τα πεδία επισημαίνονται με κόκκινο και εμφανίζονται οι σχετικές ειδοποιήσεις σφάλματος, καθοδηγώντας τον χρήστη να διορθώσει τα λάθη του.



Εικόνα 8: Οθόνες Εγγραφής (αριστερά) και Σύνδεσης (δεξιά) με λάθη

Μετά την επιτυχή εγγραφή, ο χρήστης μεταφέρεται στην οθόνη επιβεβαίωσης της εγγραφής, όπου ενημερώνεται ότι η εγγραφή του ολοκληρώθηκε με επιτυχία. Επιπλέον, αν προσπαθήσει να εγγραφεί με ένα email που είναι ήδη καταχωρημένο, ή αν κατά τη σύνδεση εισάγει λανθασμένα στοιχεία, όπως λάθος κωδικό ή email, εμφανίζονται αντίστοιχα μηνύματα λάθους, τα οποία ενημερώνουν τον χρήστη για την αποτυχία της διαδικασίας και του προτείνουν να διορθώσει τα στοιχεία του.



Εικόνα 9: Διάλογοι με μηνύματα λάθους στην Σύνδεση (δεξιά) και Εγγραφή (αριστερά)

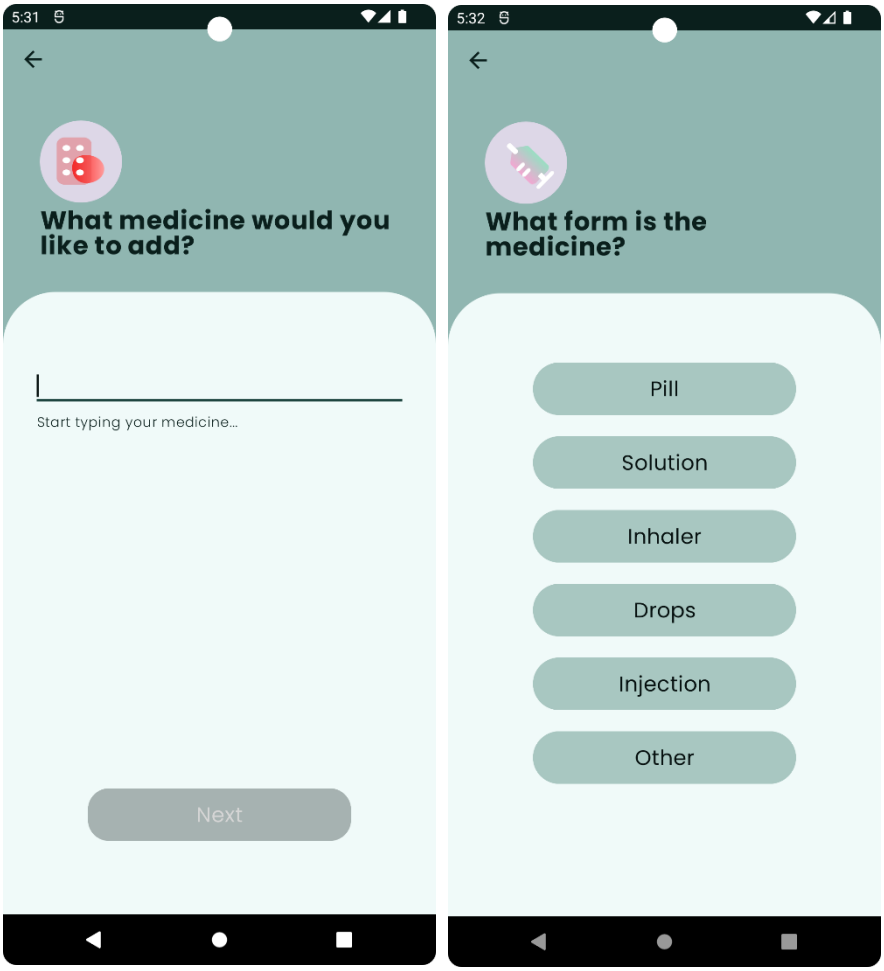
6.4 My Medicine (Αρχική Οθόνη)

Αφού ο χρήστης έχει ολοκληρώσει επιτυχώς τη διαδικασία εγγραφής ή σύνδεσης, μεταφέρεται στην αρχική οθόνη της εφαρμογής, η οποία ονομάζεται "My Medicine". Στην αρχική της μορφή, αυτή η οθόνη είναι κενή, καθώς δεν έχουν καταχωρηθεί ακόμη φάρμακα. Στο κέντρο της οθόνης εμφανίζεται η επιλογή "Add Medicine", η οποία επιτρέπει στον χρήστη να προσθέσει νέα φάρμακα στη λίστα του.

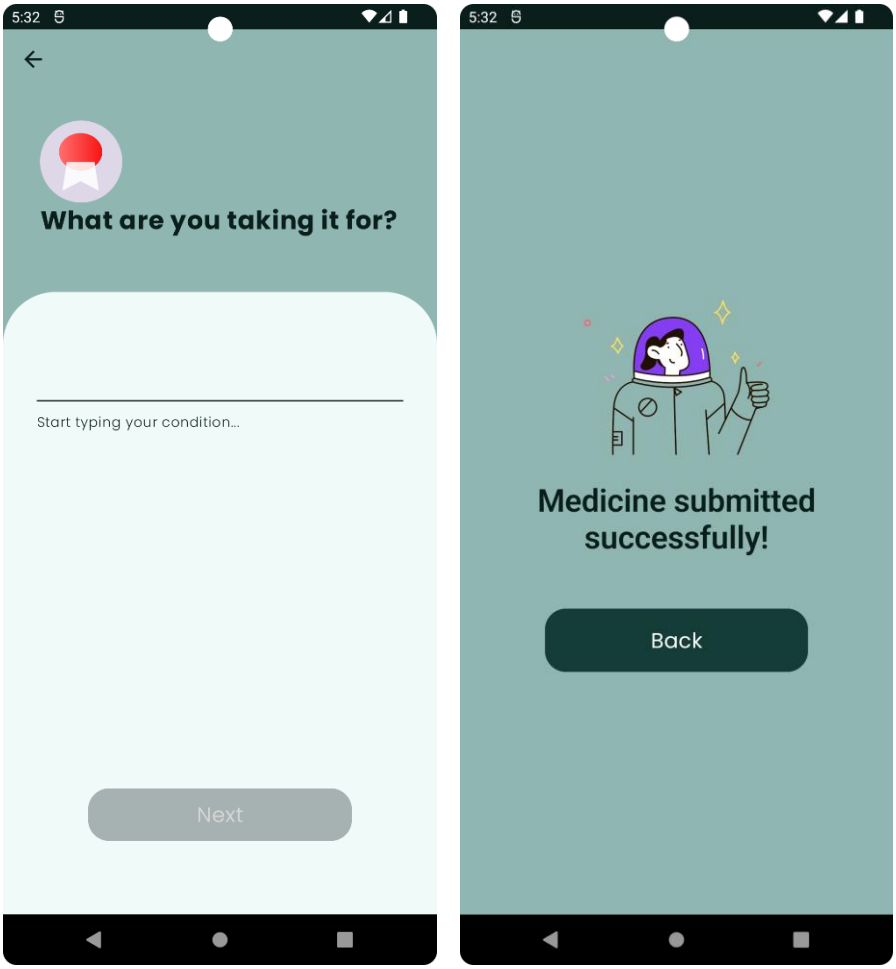
Μόλις ο χρήστης επιλέξει να προσθέσει ένα φάρμακο, μεταφέρεται σε μια σειρά από οθόνες όπου καλείται να εισάγει τις απαραίτητες πληροφορίες:

1. **Όνομα Φαρμάκου:** Ο χρήστης εισάγει το όνομα του φαρμάκου που λαμβάνει.
2. **Μορφή Φαρμάκου:** Ο χρήστης επιλέγει τη μορφή του φαρμάκου (π.χ. χάπι, σταγόνες, εισπνεόμενο).

3. Σκοπός Χρήσης: Ο χρήστης εισάγει τον λόγο για τον οποίο λαμβάνει συγκεκριμένο φάρμακο.

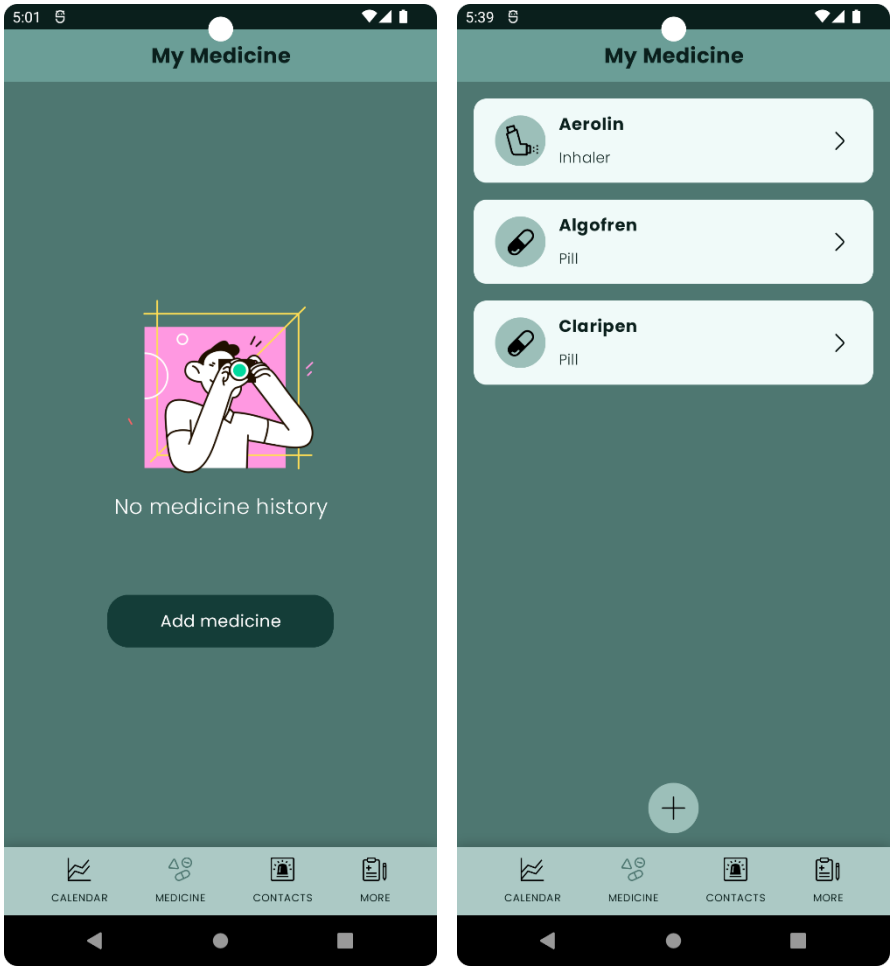


Εικόνα 10: Οθόνη καταχώρησης ονόματος φαρμάκου (αριστερά) – Οθόνη επιλογής είδους φαρμάκου (δεξιά)



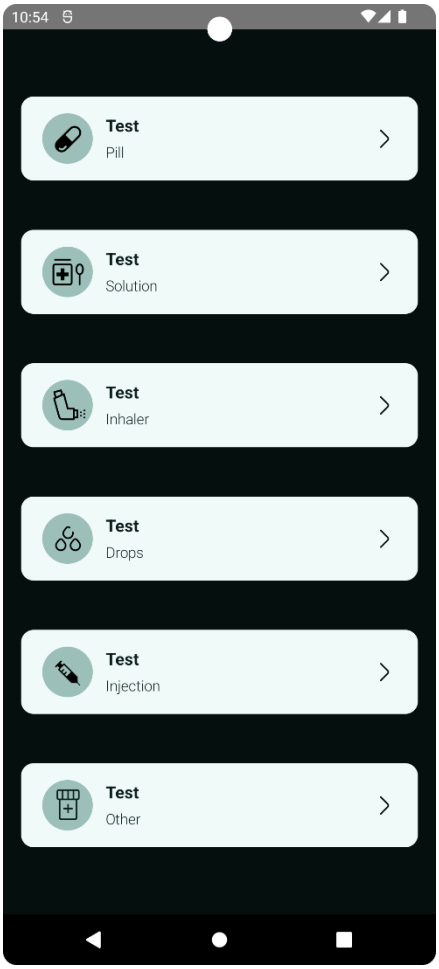
Εικόνα 11: Οθόνη καταχώρησης αίτιου λήψης του φαρμάκου (αριστερά) – Οθόνη επιτυχούς καταχώρησης φαρμάκου (δεξιά)

Μετά την εισαγωγή των πληροφοριών αυτών, εμφανίζεται μια οθόνη επιβεβαίωσης που ενημερώνει τον χρήστη ότι η εγγραφή του φαρμάκου ολοκληρώθηκε επιτυχώς. Πατώντας "Back", ο χρήστης επιστρέφει στην αρχική οθόνη "My Medicine", η οποία πλέον εμφανίζει το καταχωρημένο φάρμακο στη λίστα του. Κάτω από τη λίστα φαρμάκων, υπάρχει ένα κουμπί για την προσθήκη επιπλέον φαρμάκων.



Εικόνα 12: Οθόνη My Medicine χωρίς φάρμακα καταχωρημένα (αριστερά) - με φάρμακα καταχωρημένα (δεξιά)

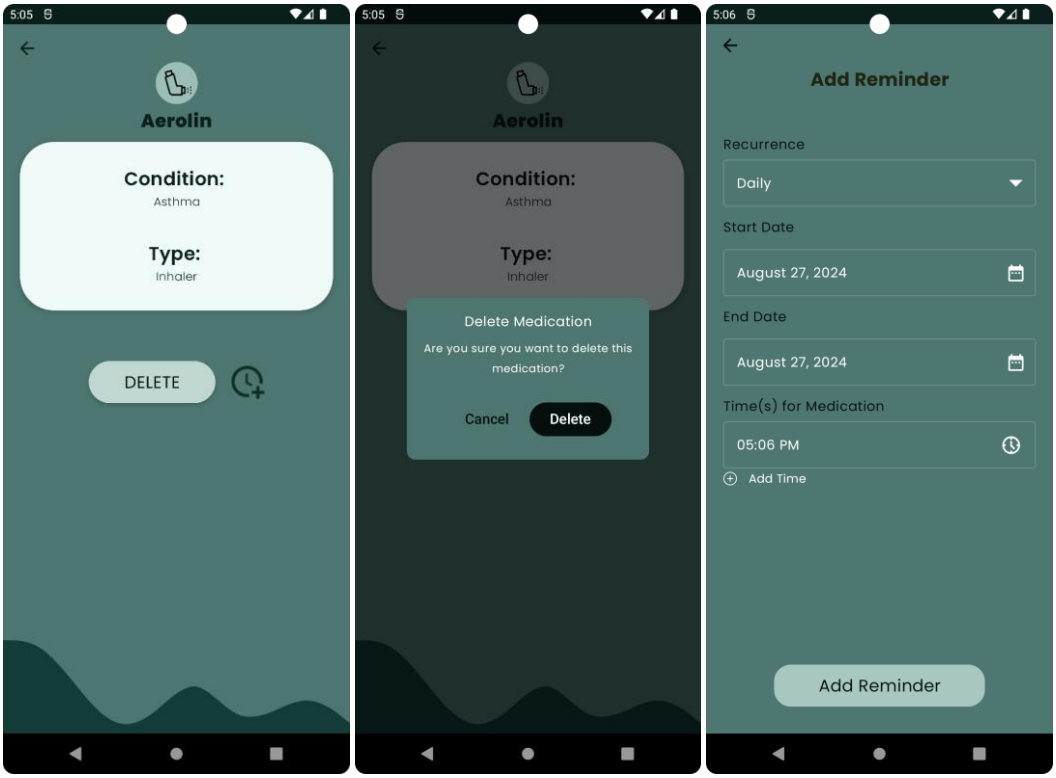
Κάθε φάρμακο συνοδεύεται από ένα διαφορετικό εικονίδιο, το οποίο αντικατοπτρίζει τον τύπο του φαρμάκου. Για παράδειγμα, τα χάπια εμφανίζονται με ένα εικονίδιο κάψουλας κτλ. Αυτή η διαφοροποίηση των εικονιδίων βοηθάει τον χρήστη να αναγνωρίσει άμεσα τον τύπο του φαρμάκου που έχει καταχωρήσει, προσφέροντας έτσι μια πιο οπτικά κατανοητή και λειτουργική εμπειρία χρήσης.



Εικόνα 13: Τύποι φαρμάκων και τα εικονίδιά τους

6.5 Λεπτομέρειες Φαρμάκων και Οθόνη Ημερολογίου

Μετά την εισαγωγή φαρμάκων στην οθόνη "My Medicine", ο χρήστης μπορεί να πατήσει σε οποιοδήποτε φάρμακο για να δει τις λεπτομέρειες του. Στην οθόνη αυτή εμφανίζονται οι πληροφορίες που έχουν ήδη καταχωρηθεί για το φάρμακο, όπως η πάθηση για την οποία χρησιμοποιείται και ο τύπος του φαρμάκου. Ο χρήστης έχει δύο βασικές επιλογές: να διαγράψει το φάρμακο ή να προσθέσει μια υπενθύμιση.



Εικόνα 14: Οθόνη λεπτομερειών φαρμάκου (αριστερά), επιλογή διαγραφής (μέση), επιλογή προσθήκης υπενθύμισης (δεξιά)

Στην οθόνη προσθήκης υπενθύμισης, ο χρήστης έχει τις εξής επιλογές:

Επιλογή Συχνότητας Ειδοποιήσεων και Ημερολόγιο:

- Ημερήσια (Daily): Η επιλογή αυτή εξασφαλίζει ότι ο χρήστης θα λαμβάνει ειδοποιήσεις καθημερινά, από την ημερομηνία έναρξης (Start Date) μέχρι την ημερομηνία λήξης (End Date). Οι ειδοποιήσεις θα υπενθυμίζουν στον χρήστη να λάβει το φάρμακο σε συγκεκριμένη ώρα κάθε ημέρας.

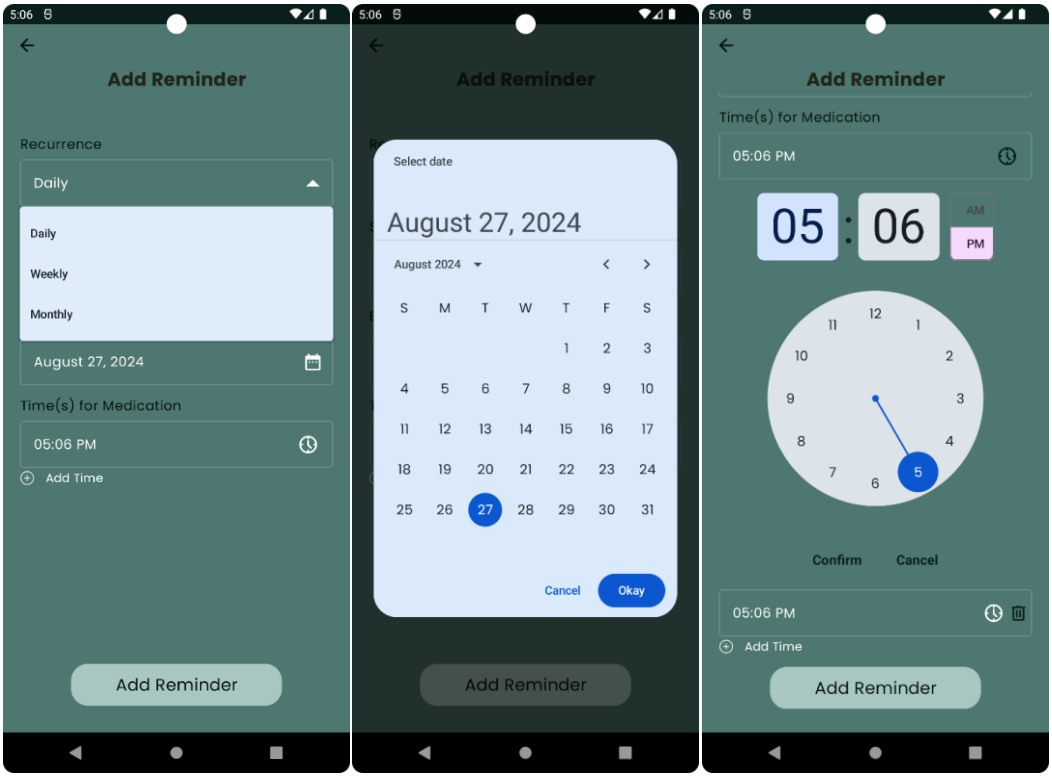
- Εβδομαδιαία (Weekly): Σε αυτή την επιλογή, ο χρήστης θα λαμβάνει ειδοποιήσεις κάθε εβδομάδα, την ίδια ημέρα της εβδομάδας όπως η ημερομηνία έναρξης. Αυτό σημαίνει ότι αν η ημερομηνία έναρξης είναι μια Δευτέρα, ο χρήστης θα λαμβάνει ειδοποίηση κάθε Δευτέρα μέχρι την ημερομηνία λήξης.

- Μηνιαία (Monthly): Εδώ, οι ειδοποιήσεις θα αποστέλλονται μία φορά το μήνα, την ίδια ημέρα του μήνα με την ημερομηνία έναρξης. Για παράδειγμα, αν η ημερομηνία έναρξης είναι 1 Σεπτεμβρίου, η ειδοποίηση θα έρχεται κάθε 1η του μήνα μέχρι την ημερομηνία λήξης.

Χρονόμετρα για Υπενθυμίσεις:

Οι υπενθυμίσεις μπορούν να περιλαμβάνουν πολλαπλά χρονικά σημεία (timers) για την ίδια ημέρα. Ο χρήστης μπορεί να ορίσει μέχρι και τέσσερα διαφορετικά χρονικά σημεία μέσα στην ημέρα για να λάβει ειδοποίηση. Για παράδειγμα, αν ένας χρήστης χρειάζεται να πάρει το φάρμακο του στις 08:00, στις 12:00, στις 16:00 και στις 20:00, μπορεί να ορίσει αυτά τα τέσσερα timers στην ίδια ημέρα, και η εφαρμογή θα του υπενθυμίσει να πάρει το φάρμακό του σε κάθε ένα από αυτά τα χρονικά σημεία.

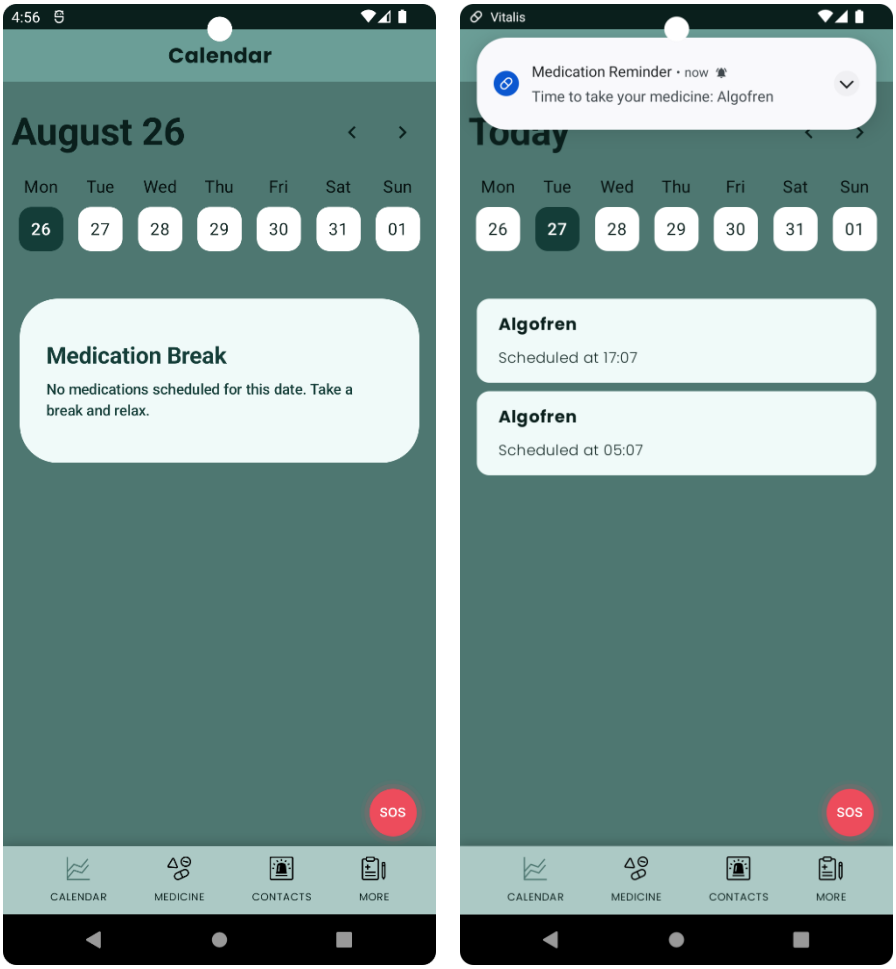
Η εφαρμογή προσαρμόζεται στις ανάγκες του χρήστη παρέχοντας τον έλεγχο για τη διαχείριση του χρονισμού των υπενθυμίσεων, εξασφαλίζοντας έτσι ότι δεν θα χάσει καμία δόση του φαρμάκου του.



Εικόνα 15: Οθόνη υπενθύμισης: Επανάληψη Υπενθύμισης (αριστερά) – Ημερολόγιο (μέση) – Ρολόι (δεξιά)

Αφού οριστούν οι Timers, ο χρήστης θα λαμβάνει ειδοποιήσεις στις συγκεκριμένες ώρες κάθε ημέρα, εβδομάδα ή μήνα, ανάλογα με την επιλεγμένη συχνότητα. Αυτό εξασφαλίζει ότι ο χρήστης θα θυμάται να πάρει το φάρμακό του ακριβώς την κατάλληλη στιγμή.

Αυτές οι λειτουργίες δίνουν στον χρήστη τον πλήρη έλεγχο για την οργάνωση της φαρμακευτικής του αγωγής, με τρόπο που είναι προσαρμοσμένος στις ατομικές του ανάγκες και στον τρόπο ζωής του.



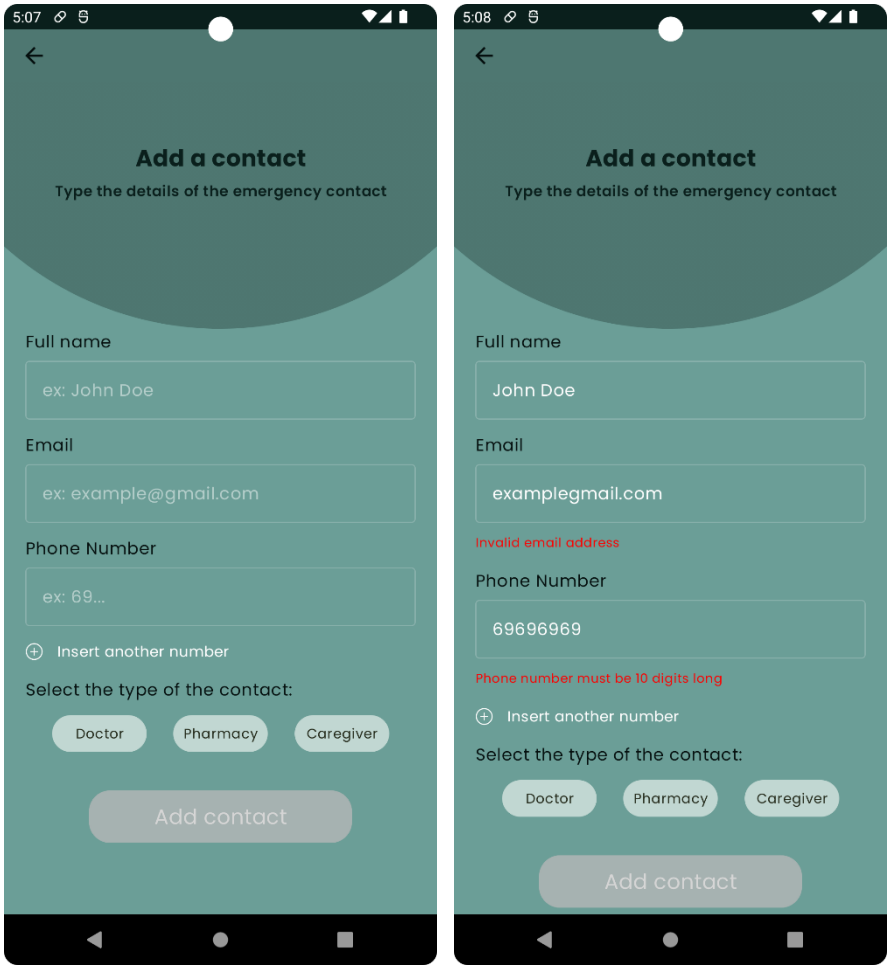
Εικόνα 16: Οθόνες χωρίς (αριστερά) και με υπενθύμιση (δεξιά)

6.6 Emergency Contacts (Οθόνη Έκτακτων Επαφών)

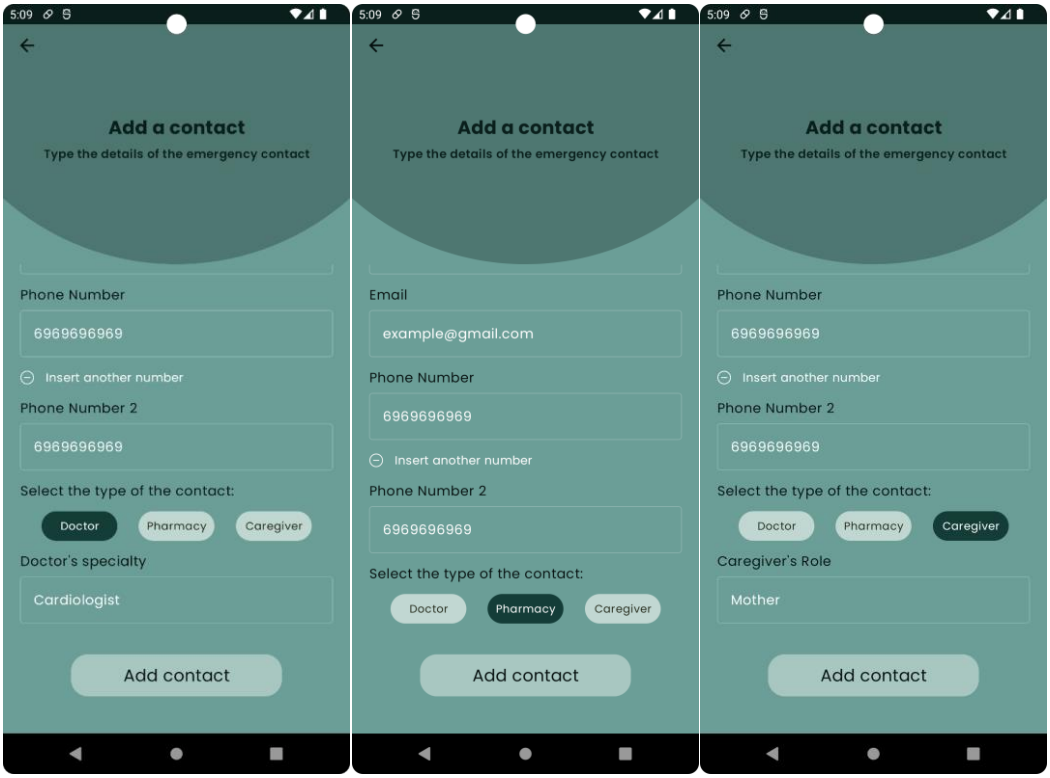
Με την πλοήγηση στο "Contacts" μέσω της κάτω μπάρας της εφαρμογής, ο χρήστης μεταφέρεται στην οθόνη των "Emergency Contacts", όπου παρουσιάζονται οι επαφές έκτακτης ανάγκης. Αρχικά, η οθόνη αυτή είναι κενή, δίνοντας τη δυνατότητα στο χρήστη να προσθέσει μια νέα επαφή μέσω του κουμπιού "Add Record". Με την επιλογή αυτή, ο χρήστης κατευθύνεται στην οθόνη "Add a Contact", όπου μπορεί να καταχωρήσει το όνομα, το email, έναν ή δύο αριθμούς τηλεφώνου, και να καθορίσει αν η επαφή είναι γιατρός (doctor), φαρμακείο (pharmacy) ή φροντιστής (caregiver).

Σε περίπτωση που τα πεδία δεν συμπληρωθούν σωστά, εμφανίζονται ανάλογα μηνύματα σφάλματος, καθοδηγώντας τον χρήστη για τις απαραίτητες διορθώσεις. Για παράδειγμα, το πεδίο του email απαιτεί έγκυρη διεύθυνση, ενώ ο αριθμός τηλεφώνου πρέπει να περιέχει 10 ψηφία.

Αν επιλεγεί η επαφή ως γιατρός, δίνεται η δυνατότητα προσθήκης της ειδικότητας (π.χ. καρδιολόγος), ενώ για τους φροντιστές υπάρχει η δυνατότητα καταχώρησης της σχέσης (π.χ. μητέρα). Μετά την ολοκλήρωση της καταχώρησης και την επιτυχή υποβολή της, ο χρήστης βλέπει μια επιβεβαίωση της προσθήκης.

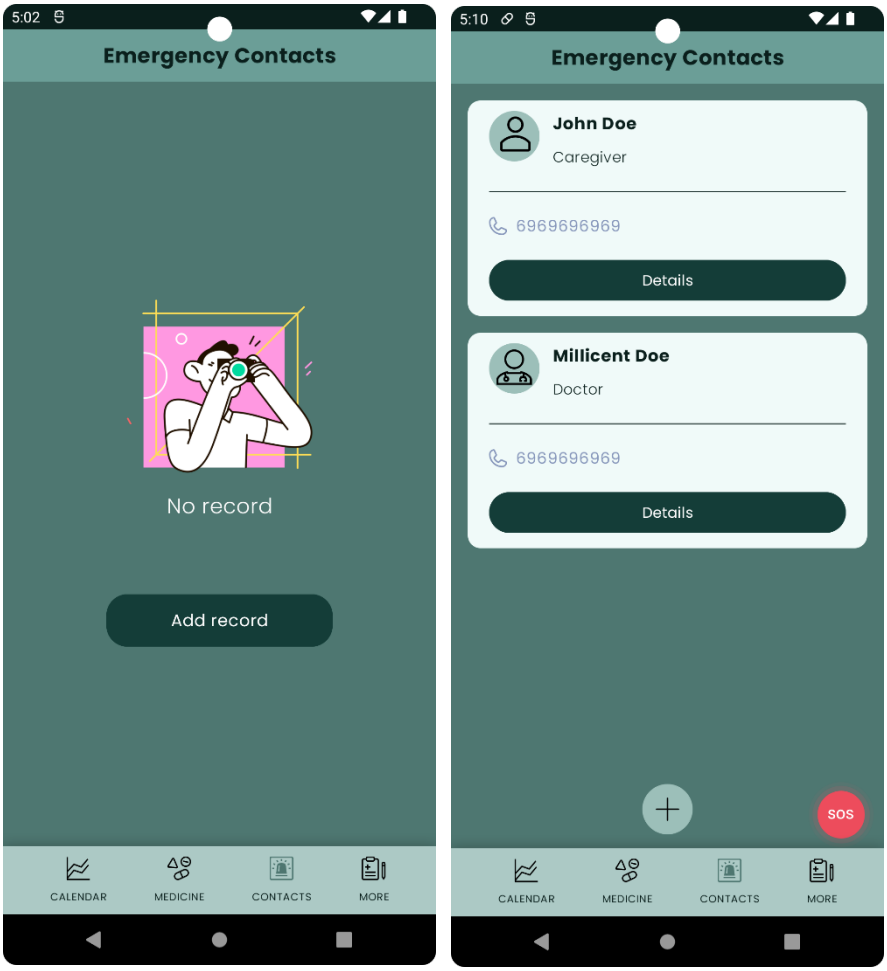


Εικόνα 17: Οθόνη για καταχώρηση επαφής χωρίς λάθη (αριστερά) - με λάθη (δεξιά)

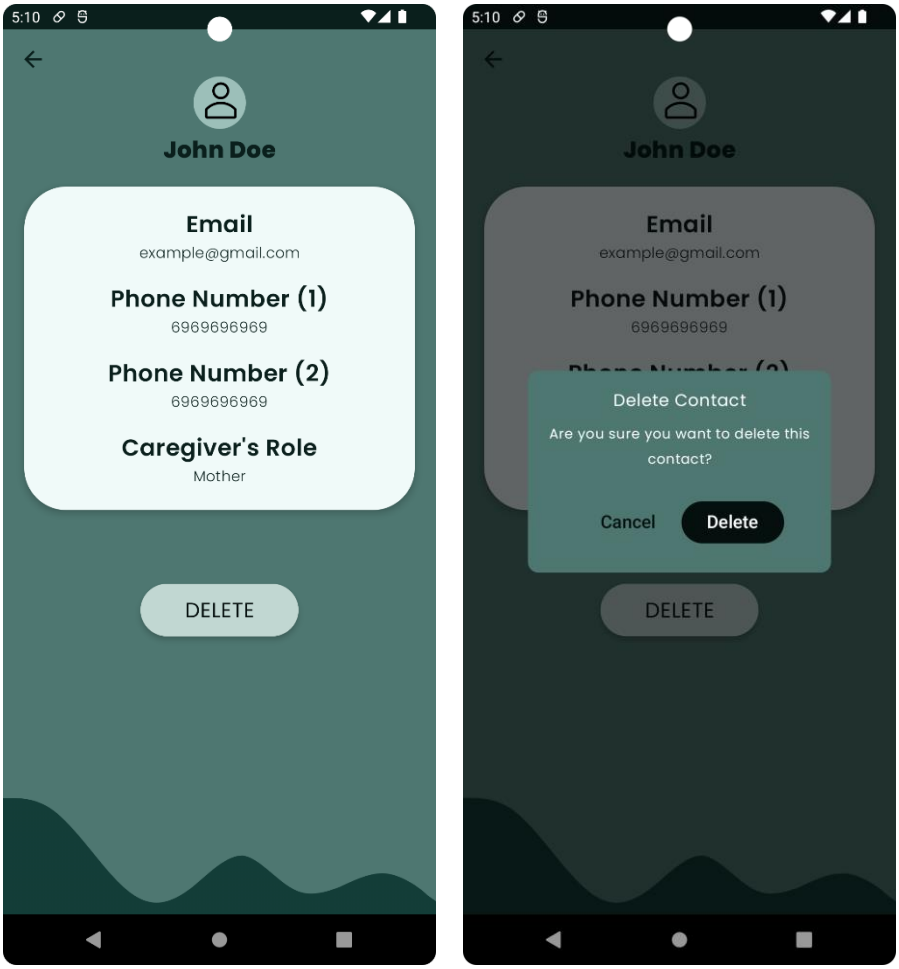


Εικόνα 18: Οθόνη με επιλεγμένο: Γιατρό (αριστερά) – Φαρμακείο (μέση) – Φροντιστή (δεξιά)

Στην οθόνη των "Emergency Contacts", οι καταχωρημένες επαφές εμφανίζονται σε ξεχωριστές κάρτες. Ο χρήστης μπορεί να πατήσει τον αριθμό τηλεφώνου για να καλέσει κατευθείαν, ή να επιλέξει "Details" για να δει περισσότερες πληροφορίες για την επαφή, να καλέσει έναν από τους αριθμούς τηλεφώνου ή να στείλει email. Επιπλέον, από την οθόνη λεπτομερειών υπάρχει η δυνατότητα διαγραφής της επαφής.



Εικόνα 19: Οθόνη Emergency Contacts χωρίς (αριστερά) και με καταχωρημένες επαφές (δεξιά)



Εικόνα 20: Οθόνη λεπτομερειών επαφής (αριστερά) και επιλογές (δεξιά)

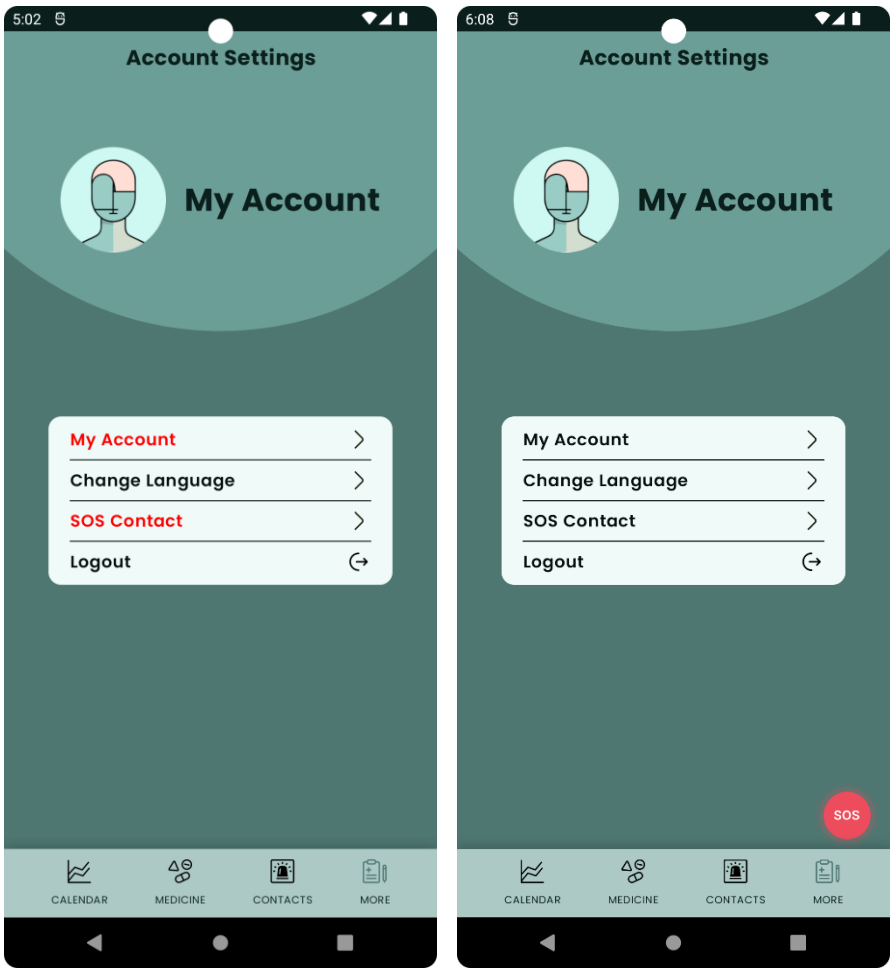
6.7 Account Settings (Οθόνη Ρυθμίσεων Λογαριασμού και Επιλογές)

Η οθόνη "Account Settings" προσφέρει στο χρήστη μια σειρά από βασικές επιλογές διαχείρισης του λογαριασμού και των ρυθμίσεων της εφαρμογής. Όταν ο χρήστης πλοηγηθεί στην επιλογή "More" από το κάτω μενού (bottom bar), θα μεταφερθεί σε αυτή την οθόνη, όπου θα έχει πρόσβαση στις εξής επιλογές:

- **Ο Λογαριασμός μου (My Account):** Επιτρέπει την προβολή και επεξεργασία των προσωπικών στοιχείων του λογαριασμού.
- **Αλλαγή Γλώσσας (Change Language):** Δίνει τη δυνατότητα αλλαγής της γλώσσας της εφαρμογής.
- **SOS Επαφή (SOS Contact):** Επιτρέπει τη διαχείριση των επαφών έκτακτης ανάγκης που μπορεί να ειδοποιηθούν σε περίπτωση ανάγκης.
- **Αποσύνδεση (Logout):** Επιτρέπει την ασφαλή αποσύνδεση από την εφαρμογή.

Αν οι πληροφορίες του λογαριασμού ή της SOS επαφής δεν έχουν συμπληρωθεί, οι αντίστοιχες επιλογές θα εμφανίζονται με κόκκινο χρώμα, υποδεικνύοντας ότι απαιτείται ενέργεια από το χρήστη. Μόλις οι ρυθμίσεις αυτές ολοκληρωθούν, το χρώμα

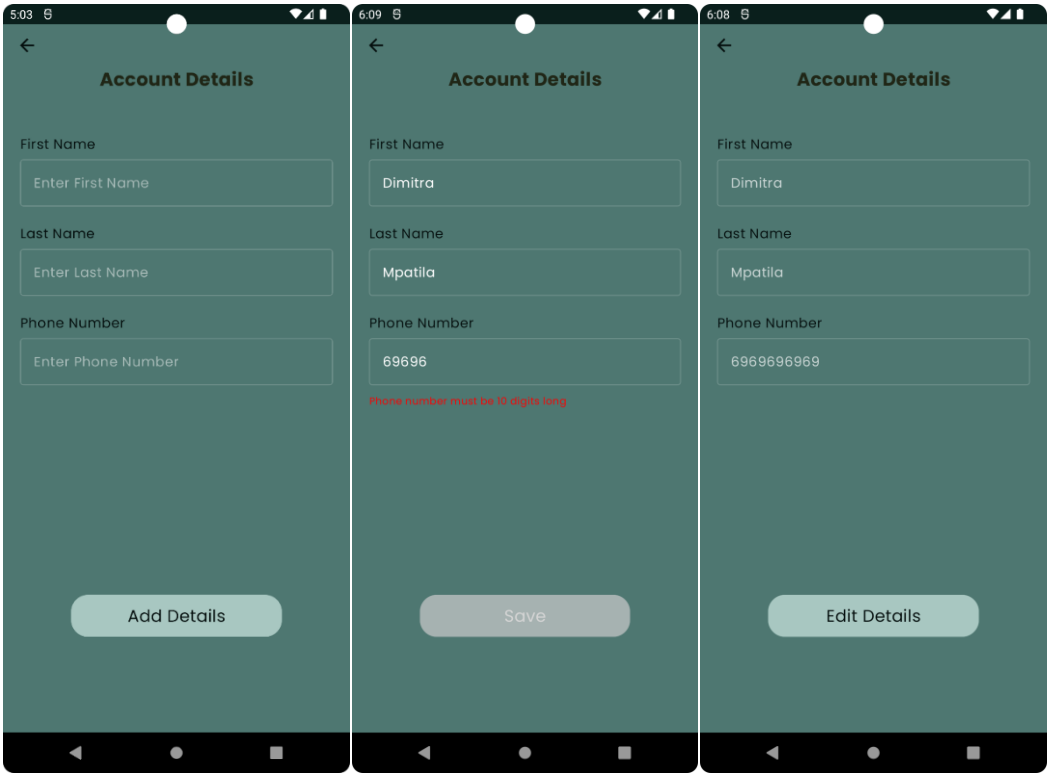
αλλάζει και εναρμονίζεται με τις υπόλοιπες επιλογές, προσφέροντας μια καθαρή ένδειξη ότι η ρύθμιση έχει ολοκληρωθεί.



Εικόνα 21: Οθόνη Account Settings χωρίς (αριστερά) και με καταχωρημένα στοιχεία (δεξιά) στο My Account και SOS Contact

6.7.1 Account Details (Ο Λογαριασμός Μου)

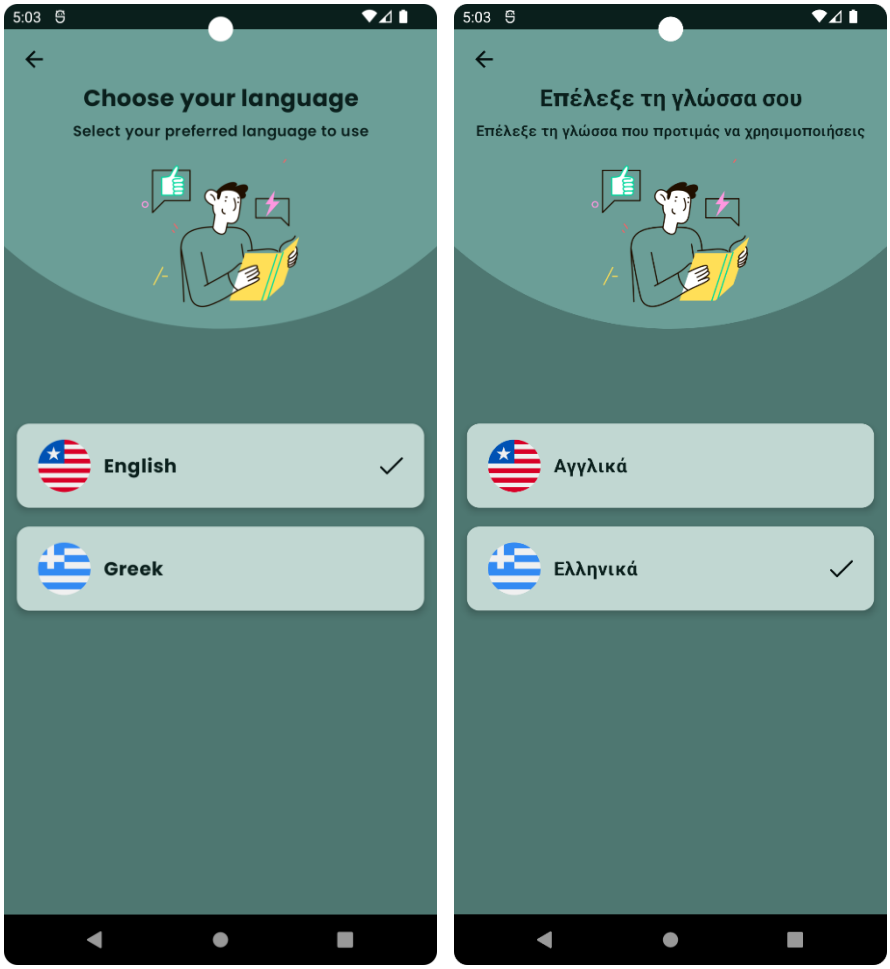
Πατώντας την επιλογή "My Account" στην οθόνη "Account Settings", μεταφερόμαστε στην οθόνη "Account Details". Εδώ μπορούμε να δούμε και να επεξεργαστούμε τα προσωπικά μας στοιχεία, όπως το όνομα, το επώνυμο και τον αριθμό τηλεφώνου. Αν κάποιο από αυτά τα πεδία είναι λανθασμένα συμπληρωμένο ή αν λείπουν απαραίτητα δεδομένα, εμφανίζονται μηνύματα σφάλματος για τη διόρθωσή τους. Μετά τη διόρθωση, μπορούμε να αποθηκεύσουμε τις αλλαγές μας.



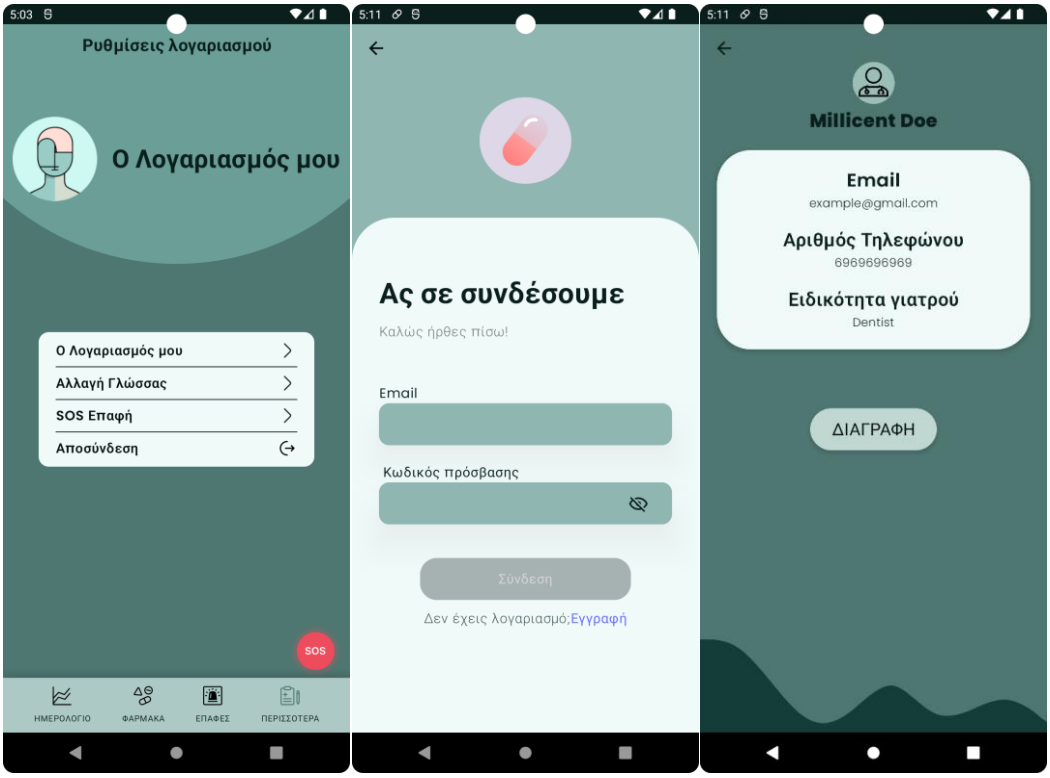
Εικόνα 22: Οθόνη Account Details : χωρίς συμπληρωμένα στοιχεία (αριστερά) με λάθος (μέση) και με συμπληρωμένα (δεξιά)

6.7.2 Change Language (Αλλαγή Γλώσσας)

Η λειτουργία Change Language επιτρέπει στους χρήστες να επιλέξουν την προτιμώμενη γλώσσα για την εφαρμογή. Αυτή η δυνατότητα υποστηρίζει δύο γλώσσες, Αγγλικά και Ελληνικά. Με την αλλαγή γλώσσας, όλα τα κείμενα και τα μηνύματα της εφαρμογής προσαρμόζονται στη νέα γλώσσα, παρέχοντας μια άμεση και ολοκληρωμένη εμπειρία χρήσης στη γλώσσα επιλογής του χρήστη. Παρακάτω παραθέτουμε στιγμιότυπα της εφαρμογής τόσο στα Αγγλικά όσο και στα Ελληνικά, για να αναδείξουμε πώς μεταφράζονται οι διάφορες λειτουργίες και οι επιλογές της εφαρμογής στις δύο γλώσσες.



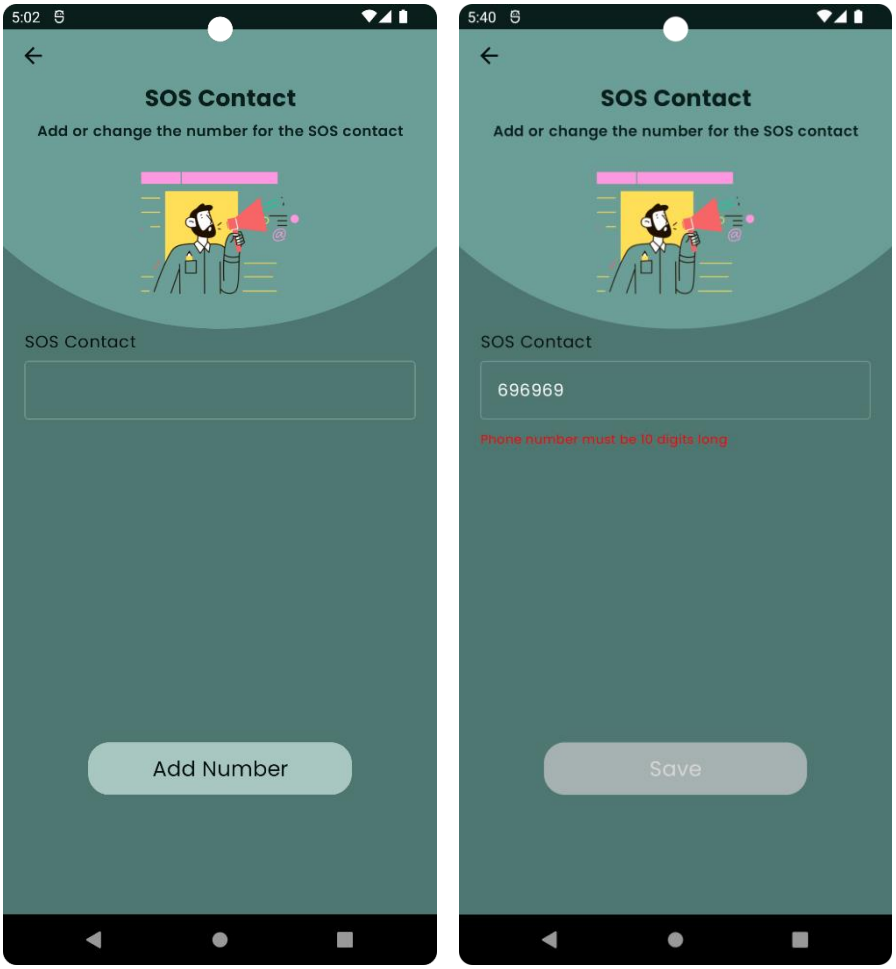
Εικόνα 23: Οθόνη αλλαγής γλώσσας Αγγλικά (αριστερά) – Ελληνικά (δεξιά)



Εικόνα 24: Παραδείγματα οθονών στα ελληνικά

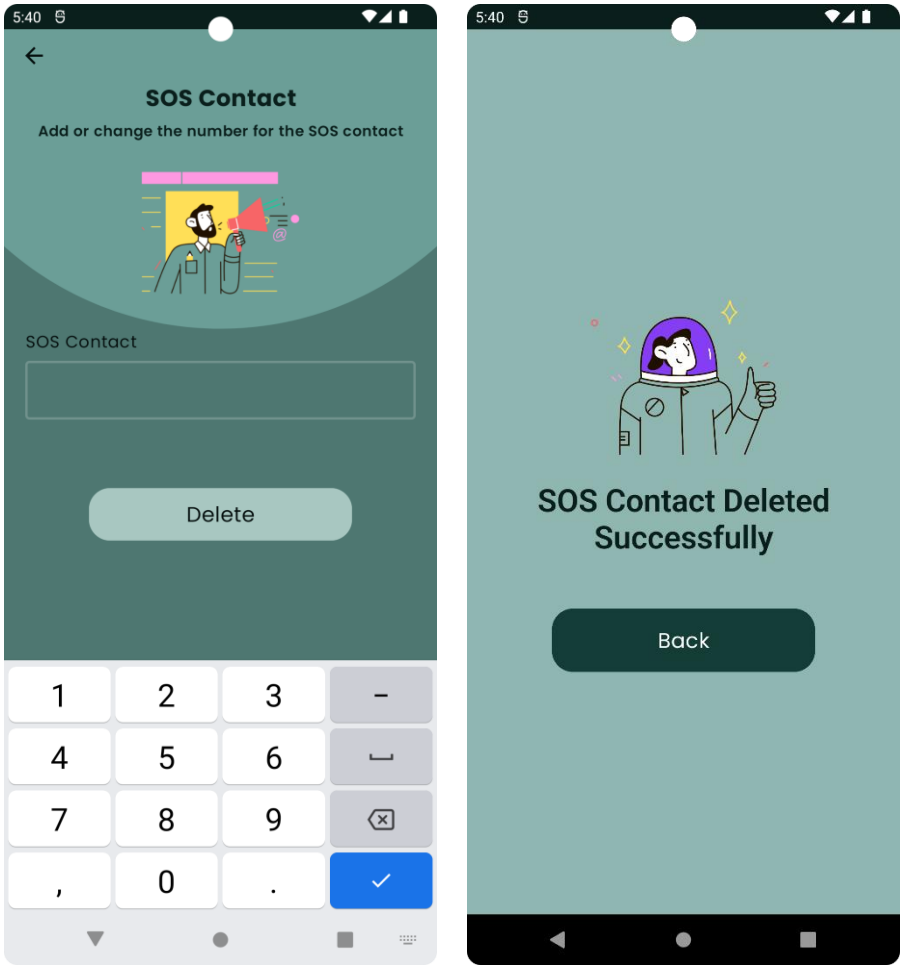
6.7.3 SOS Contact (SOS Επαφή)

Στη συνέχεια, υπάρχει η επιλογή καταχώρησης ενός αριθμού έκτακτης ανάγκης στην ενότητα SOS Contact (SOS Επαφή). Ο αριθμός που καταχωρείται εμφανίζεται σε όλες τις οθόνες της εφαρμογής μέσω ενός κουμπιού SOS, επιτρέποντας στον χρήστη να καλέσει απευθείας την επαφή έκτακτης ανάγκης από οποιοδήποτε σημείο της εφαρμογής. Εάν καταχωρηθεί ένας μη έγκυρος αριθμός (π.χ. λιγότερα από 10 ψηφία), το σύστημα εμφανίζει μήνυμα λάθους και δεν επιτρέπει την αποθήκευση.



Εικόνα 25: Οθόνη καταχώρησης SOS επαφής χωρίς καταχώρηση (αριστερά) – με λάθος καταχώρηση (δεξιά)

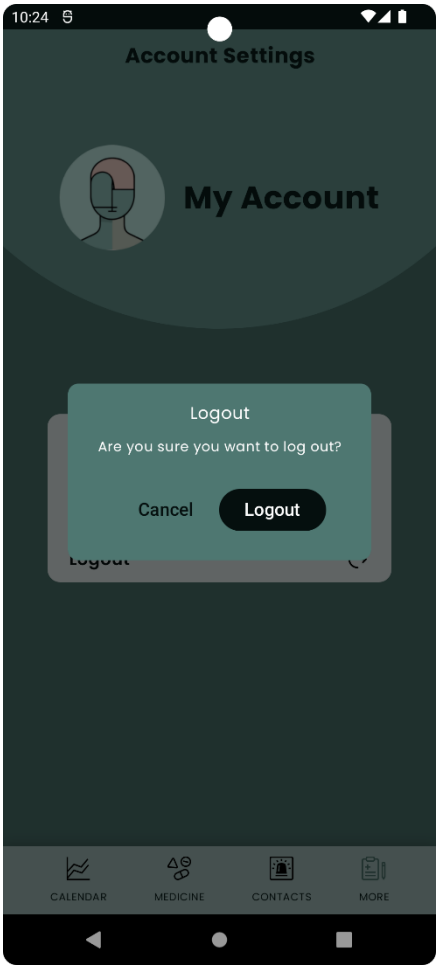
Στην περίπτωση που ο χρήστης θέλει να αφαιρέσει τον καταχωρημένο αριθμό έκτακτης ανάγκης, υπάρχει η επιλογή Delete (Διαγραφή), και μετά τη διαγραφή, εμφανίζεται μήνυμα επιτυχίας.



Εικόνα 26: Οθόνη SOS επαφής με κουμπί για διαγραφή (αριστερά) - επιτυχής οθόνη διαγραφής (δεξιά)

6.7.4 Logout (Αποσύνδεση)

Με την επιλογή Logout, ο χρήστης μπορεί να αποσυνδεθεί από την εφαρμογή, κλείνοντας την τρέχουσα συνεδρία του.



Εικόνα 27: Διάλογος αποσύνδεσης

7. Συμπεράσματα και Μελλοντικές Επεκτάσεις

7.1 Συμπεράσματα

Η διαδικασία ανάπτυξης της κινητής εφαρμογής παροχής ιατρικής φροντίδας ανέδειξε την αναγκαιότητα και τη σημασία της χρήσης σύγχρονων τεχνολογιών στην ανάπτυξη λογισμικού. Η ενσωμάτωση της πλατφόρμας Firebase διασφάλισε την αξιόπιστη διαχείριση και συγχρονισμό των δεδομένων σε πραγματικό χρόνο, ενώ η χρήση της γλώσσας προγραμματισμού Kotlin και του Jetpack Compose συνέβαλαν στη δημιουργία ενός ευχάριστου και αποδοτικού περιβάλλοντος χρήστη.

Κατά την υλοποίηση της εφαρμογής, έγινε εμφανής η σημασία του σωστού σχεδιασμού και της επιλογής κατάλληλης αρχιτεκτονικής για την επίτευξη μιας ευέλικτης και κλιμακούμενης λύσης. Η εφαρμογή ανταποκρίνεται στις ανάγκες των χρηστών για καθημερινή διαχείριση της υγείας, προσφέροντας εργαλεία που διευκολύνουν την παρακολούθηση της φαρμακευτικής αγωγής και την εξασφάλιση της άμεσης επικοινωνίας με τις επαφές έκτακτης ανάγκης.

Παρά τις προσπάθειες για την επίτευξη τελειότητας, η ανάπτυξη λογισμικού συνοδεύεται αναπόφευκτα από την εμφάνιση σφαλμάτων. Η ανίχνευση και η επίλυσή τους είναι ζωτικής σημασίας για τη διασφάλιση της σταθερότητας και της αξιοπιστίας της εφαρμογής. Η συστηματική παρακολούθηση και η προσεκτική αντιμετώπιση των σφαλμάτων επιτρέπουν την άμεση ανταπόκριση στις ανάγκες των χρηστών και την αποφυγή επιπλοκών στο μέλλον.

Τέλος, η εργασία αναδεικνύει τη σημασία της συνεχούς βελτίωσης και επέκτασης της εφαρμογής. Η ενσωμάτωση νέων λειτουργιών και η αξιοποίηση προηγμένων τεχνολογιών, θα μπορούσαν να ενισχύσουν περαιτέρω τις δυνατότητες πρόβλεψης και πρόληψης προβλημάτων υγείας, καθιστώντας την εφαρμογή ακόμα πιο πολύτιμη και αποτελεσματική για τους χρήστες της.

7.2 Μελλοντικές επεκτάσεις

Παρά τη σημαντική πρόοδο που επιτεύχθηκε, υπάρχουν περιθώρια για περαιτέρω βελτιώσεις και επεκτάσεις της εφαρμογής:

- **Προσθήκη Χάρτη και Εντοπισμός Φαρμακείων:** Μια μελλοντική επέκταση της εφαρμογής θα μπορούσε να περιλαμβάνει την ενσωμάτωση ενός διαδραστικού χάρτη, όπου ο χρήστης θα έχει τη δυνατότητα να εντοπίζει φαρμακεία στην περιοχή του. Επιπλέον, η εφαρμογή θα μπορούσε να παρέχει πληροφορίες σχετικά με τα φαρμακεία που είναι ανοιχτά και εφημερεύουν, διευκολύνοντας τον χρήστη να βρει το πλησιέστερο διαθέσιμο φαρμακείο οποιαδήποτε στιγμή.
- **Ενσωμάτωση Σαρωτή και Ανέβασμα Εξετάσεων**: Η προσθήκη λειτουργίας σαρωτή για την καταγραφή εξετάσεων και η δυνατότητα αποθήκευσης αυτών

των αρχείων στην εφαρμογή ως μέρος του ιατρικού ιστορικού του χρήστη θα επιτρέψει την εύκολη και οργανωμένη πρόσβαση σε σημαντικά ιατρικά δεδομένα. Αυτή η επέκταση θα διευκολύνει τους χρήστες να διατηρούν ένα πλήρες ιστορικό εξετάσεων, που θα είναι διαθέσιμο ανά πάσα στιγμή.

- **Επέκταση Υποστήριξης Συσκευών**: Η υποστήριξη για έξυπνες συσκευές, όπως έξυπνα ρολόγια και άλλες φορητά (wearables), θα μπορούσε να ενισχύσει τη λειτουργικότητα της εφαρμογής, επιτρέποντας την παρακολούθηση της υγείας σε πραγματικό χρόνο και την αυτόματη καταγραφή δεδομένων υγείας.
- **Διασύνδεση με Ιατρικά Συστήματα**: Η διασύνδεση της εφαρμογής με ηλεκτρονικούς φακέλους υγείας και άλλες πλατφόρμες υγείας θα μπορούσε να προσφέρει μια πιο ολοκληρωμένη εικόνα της υγείας του χρήστη, διευκολύνοντας τη συνεργασία με γιατρούς και άλλους επαγγελματίες υγείας.
- **Ενίσχυση της Ασφάλειας και της Προστασίας Προσωπικών Δεδομένων**: Με την αυξανόμενη ανησυχία για την προστασία των προσωπικών δεδομένων, η ενσωμάτωση πιο ισχυρών μηχανισμών ασφαλείας και η συμμόρφωση με τις τελευταίες νομοθεσίες προστασίας δεδομένων θα είναι απαραίτητες για την εμπιστοσύνη των χρηστών.

Οι προτάσεις αυτές στοχεύουν στην περαιτέρω βελτίωση της εφαρμογής, παρέχοντας μια ολοκληρωμένη και εξαιρετικά λειτουργική πλατφόρμα διαχείρισης υγείας για τους χρήστες.

Βιβλιογραφία

[1] «Firebase,» [Ηλεκτρονικό]. Available: https://firebase.google.com. [Πρόσβαση 13 September 2024].

[2] «Kotlin,» [Ηλεκτρονικό]. Available: https://kotlinlang.org. [Πρόσβαση 13 September 2024].

[3] «Jetpack Compose,» [Ηλεκτρονικό]. Available: https://developer.android.com/compose. [Πρόσβαση 13 September 2024].

[4] R. C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship., Upper Saddle River, NJ, USA: Prentice Hall, 2008.

[5] «Java,» [Ηλεκτρονικό]. Available: https://www.java.com/en/. [Πρόσβαση 13 September 2024].

[6] L. A. S. &. E. K. N. Domingo, «A Comparative Review of Mobile Application Development Frameworks: Kotlin Vs Java.,» *IRE Journals,* τόμ. 6(3), pp. 112-119, 2022.

[7] T. T. K. R. A. &. R. D. W. Putranto, «A Comparative Study of Java and Kotlin for Android Mobile Application Development,» σε *International Seminar on Research of Information Technology and Intelligent Systems (ISRITI) (pp. 59-64)*, 2020.

[8] M. Setter, «CodeSubmit,» 14 June 2022. [Ηλεκτρονικό]. Available: https://codesubmit.io/blog/history-of-android-operating-system/. [Πρόσβαση 13 September 2024].

[9] C. Haase, Androids: The Team That Built the Android Operating System, Mountain View, California: Gretchen Achilles, 2021.

[10] RowdyTech, «Android vs. iOS: A Comparison of the Pros and Cons,» 24 February 2023. [Ηλεκτρονικό]. Available: https://rowdytech.com/android/android-vs-ios/. [Πρόσβαση 13 September 24].

[11] K. Jakimoski, «Analysis of the Advantages and Disadvantages of Android and iOS Systems and Converting Applications from Android to iOS Platform and Vice Versa,» *American Journal of Software Engineering and Applications,* τόμ. 6(5), pp. 116-120, 2017.

[12] A. Moroni, «Android vs. iOS: Exploring The Pros and Cons of Two Leading Mobile Operating Systems,» Hardcore Droid, 10 August 2023. [Ηλεκτρονικό]. Available: https://www.hardcoredroid.com/android-vs-ios-exploring-the-pros-and-cons-of-two-leading-mobile-operating-systems/. [Πρόσβαση 13 September 2024].

«Don't kill my app!,» [Ηλεκτρονικό]. Available: https://dontkillmyapp.com. [Πρόσβαση

13] 13 September 2024].

«Android Studio,» [Ηλεκτρονικό]. Available: https://developer.android.com/studio.
14] [Πρόσβαση 13 September 2024].

«IntelliJ IDEA,» [Ηλεκτρονικό]. Available: https://www.jetbrains.com/idea/. [Πρόσβαση
15] 13 September 2024].

«Develop a UI with Views,» Android Studio, [Ηλεκτρονικό]. Available:
16] https://developer.android.com/studio/write/layout-editor. [Πρόσβαση 13 September 2024].

«Gradle Build Tool,» [Ηλεκτρονικό]. Available: https://gradle.org. [Πρόσβαση 13
17] September 2024].

«Eclipse,» [Ηλεκτρονικό]. Available: https://www.eclipse.org. [Πρόσβαση 13 September
18] 2024].

«Swift,» [Ηλεκτρονικό]. Available: https://www.swift.org. [Πρόσβαση 13 September
19] 2024].

«AWS,» [Ηλεκτρονικό]. Available: https://aws.amazon.com. [Πρόσβαση 13 September
20] 2024].

«Google Cloud Platform,» [Ηλεκτρονικό]. Available: https://cloud.google.com.
21] [Πρόσβαση 13 September 2024].

«Microsoft Azure,» [Ηλεκτρονικό]. Available: https://azure.microsoft.com/en-us.
22] [Πρόσβαση 13 September 2024].

«Material Design,» [Ηλεκτρονικό]. Available: https://m3.material.io. [Πρόσβαση 13
23] September 2024].

«SOLID Design Principles Explained: Dependency Inversion Principle with Code
24] Examples,» Stackify, [Ηλεκτρονικό]. Available: https://stackify.com/dependency-inversion-principle/. [Πρόσβαση 13 September 2024].

N. Hodges, «Dependency Injection: Everything You Need to Know,» [Ηλεκτρονικό].
25] Available: https://builtin.com/articles/dependency-injection. [Πρόσβαση 13 September 2024].

«Hilt,» [Ηλεκτρονικό]. Available: https://dagger.dev/hilt/. [Πρόσβαση 13 September
26] 2024].

«Koin,» [Ηλεκτρονικό]. Available: https://insert-koin.io. [Πρόσβαση 13 September
27] 2024].

«Clean Architecture: Kotlin and Compose,» [Ηλεκτρονικό]. Available:
28] https://paulallies.medium.com/clean-architecture-in-the-flavour-of-jetpack-compose-dd4b0016f815. [Πρόσβαση 13 September 2024].

«GitHub,» [Ηλεκτρονικό]. Available: https://github.com. [Πρόσβαση 13 September
29] 2024].

«Firebase Authentication,» [Ηλεκτρονικό]. Available:
30] https://firebase.google.com/docs/auth. [Πρόσβαση 13 September 2024].

«Firebase Realtime Database,» [Ηλεκτρονικό]. Available:
31] https://firebase.google.com/docs/database. [Πρόσβαση 13 September 2024].