ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ ΕΠΙΧΕΙΡΗΣΕΩΝ
SCHOOL OF BUSINESS

ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ & ΤΕΧΝΟΛΟΓΙΑΣ
DEPARTMENT OF MANAGEMENT SCIENCE & TECHNOLOGY

ΕΡΓΑΣΙΑ ΣΤΟ ΜΑΘΗΜΑ ΤΗΣ ΑΝΑΠΤΥΞΗΣ ΚΑΙ ΣΧΕΔΙΑΣΗΣ MOBILE ΕΦΑΡΜΟΓΩΝ

ΑΝΑΠΤΥΞΗ ΚΑΙ ΣΧΕΔΙΑΣΗ MOOD TRACKING ΕΦΑΡΜΟΓΗΣ ΓΙΑ ΛΕΙΤΟΥΡΓΙΚΟ ΣΥΣΤΗΜΑ ANDROID

ΑΡΓΥΡΩ ΤΣΑΜΠΙΚΑ ΣΚΟΥΜΙΟΥ (Α.Μ. 8210138)
ΔΗΜΗΤΡΑ ΠΑΡΑΒΑΤΟΥ (Α.Μ. 8210118)
ΜΑΡΙΑ ΚΟΜΙΣΣΑ ΚΑΛΑΦΑΤΗ (Α.Μ. 8190051)

Mootraki App

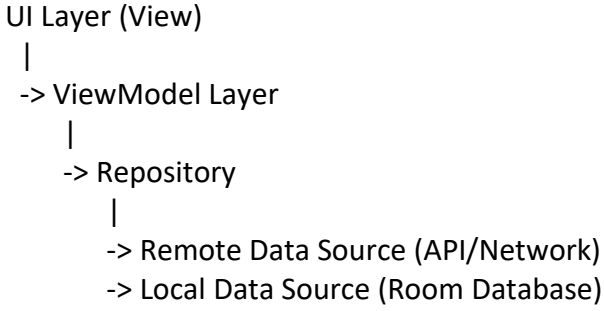
1. Η ΙΔΕΑ

Το Mootraki (mood tracker και τη λέξη μούτρο = πρόσωπο). Η εφαρμογή έχει σχεδιαστεί με στόχο να προσφέρει μια αίσθηση εσωτερικής αρμονίας στην καθημερινότητα του χρήστη. Μέσα από τις λειτουργίες της, φιλοδοξεί να καλλιεργήσει ένα περιβάλλον αυτογνωσίας, καθώς ο χρήστης μπορεί να καταγράφει και να παρακολουθεί την καθημερινή συναισθηματική του κατάσταση και να δέχεται ερεθίσματα που αποσκοπούν στην ευημερία και αυτοβελτίωση του. Απευθύνεται σε άτομα που θέλουν να εξερευνήσουν τις εσωτερικές τους σκέψεις και συναισθήματα με σκοπό την κατανόηση του εαυτού και των αναγκών τους.

2. ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΧΕΔΙΑΣΗ

Για την υλοποίηση της εφαρμογής χρησιμοποιήθηκε το Jetpack Compose και η αρχιτεκτονική του χωρίζεται σε δύο βασικά επίπεδα, τα UI Layer και Data Layer.
Το UI Layer περιλαμβάνει ό,τι σχετίζεται με την διεπαφή του χρήστη και υλοποιείται εξ ολοκλήρου με την jetpack compose. Σε αυτό το επίπεδο υπάρχουν και τα ViewModels για την διαχείριση της κατάστασης (state) και επικοινωνούν με το Data Layer μέσω StateFlow. Ακόμη, εκτός από το κώδικα για τα components που εμφανίζονται στην οθόνη του χρήστη, περιέχονται το navigation και η screen.
Το Data Layer περιλαμβάνει ό,τι σχετίζεται με την απόκτηση και τη διαχείριση δεδομένων είτε από εξωτερικές πηγές (API) για την λειτουργία Affirmations είτε από εσωτερικές πηγές. Ως τοπική βάση δεδομένων χρησιμοποιείται η Room. Αξιοποιείται το interface DAO (Data Access Objects) για την εκτέλεση Create, Read, Update, Delete λειτουργιών, η Flow για την αυτόματη ενημέρωση του UI μέσω του ViewModel για αλλαγές στη βάση και το Repository για την επικοινωνία του ViewModel με τη βάση δεδομένων και την παροχή των απαραίτητων δεδομένων. Ο βασικός πίνακας της εφαρμογής είναι ο entry που αποθηκεύει την καθημερινή καταχώρηση του χρήστη.

3. ΟΡΓΑΝΩΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ



4. ΤΕΚΜΗΡΙΩΣΗ ΣΧΕΔΙΑΣΤΙΚΩΝ ΕΠΙΛΟΓΩΝ

4.1 Καινοτομίες & Προκλήσεις

Καθ' όλη τη διάρκεια της ανάπτυξης της εφαρμογής, η ομάδα αντιμετώπισε προκλήσεις που απαιτούσαν καινοτόμες λύσεις για να βελτιώσουν την αποδοτικότητα, την αισθητική και τη διαχείριση των δεδομένων.

Α) Εκμάθηση γλώσσας προγραμματισμού Kotlin
Πρωταρχική πρόκληση, ήταν η απόρριψη της λογικής λειτουργίας της java την οποία γνωρίζουν τα μέλη της ομάδας και η εκμάθηση μιας νέας γλώσσας προγραμματισμού της Kotlin η οποία είχε ταυτόχρονα πολλές ομοιότητες αλλά και βασικές διαφορές με τη Java.

***Β)* Χρήση Jetpack Compose με Ελάχιστο Αριθμό Dependencies**
Η χρήση της jetpack compose προσφέρει σημαντικά πλεονεκτήματα, όπως βελτιωμένη αναγνωσιμότητα. Παράλληλα, υπήρξε πρόκληση να περιορίσουμε τον αριθμό των εξαρτήσεων (dependencies) στο απολύτως απαραίτητο, για να αποφύγουμε την περιττή πολυπλοκότητα στον κώδικα. Αυτή η προσέγγιση επέτρεψε την ευκολία συντήρησης και τη μείωση του χρόνου ανάπτυξης, ενώ διατηρήσαμε την εφαρμογή ελαφριά και γρήγορη. Η έμφαση στην απλότητα επιτεύχθηκε μέσω της χρήσης των πιο βασικών εργαλείων του Jetpack Compose, εξασφαλίζοντας ότι η εφαρμογή παραμένει ευέλικτη και λειτουργική χωρίς να εισάγονται υπερβολικές εξαρτήσεις.

***Γ)* Σχεδίαση Γραφημάτων με Canva**
Η σχεδίαση όλων των γραφημάτων και εικόνων της εφαρμογής πραγματοποιήθηκε εξ ολοκλήρου με το εργαλείο Canva, χάριν ευχρηστίας και την ταχύτητας. Ωστόσο, ήταν πρόκληση να υπολογιστεί/ βρεθεί ο σωστός τρόπος σχεδιασμού τους, καθώς χρειάστηκε η χρήση μαθηματικών συναρτήσεων.

***Δ)* Συγχρονισμός video, timer και κατηγοριών βίντεο**
Μια από τις προκλήσεις και ταυτόχρονα καινοτομίες που υλοποιήθηκαν στην εφαρμογή αφορά τον συγχρονισμό του video playback με τον timer και τις κατηγορίες των βίντεο. Επιτεύχθηκε μέσω χρήσης coroutines ,listener callbacks, State, if blocks και πολλές παραμέτρους, για την ταυτόχρονη παρακολούθηση και ενημέρωση της κατάστασης.

***Ε)* Οργάνωση των ViewModels μέσω της AppViewModelProvider**
Ένα από τα πιο σημαντικά σημεία καινοτομίας αλλά και πρόκλησης ήταν η λεπτομερής οργάνωση των ViewModels για τη σωστή διαχείριση της κατάστασης της εφαρμογής. Χρησιμοποιήσαμε τον AppViewModelProvider με τις βιβλιοθήκες lifecycle για να δημιουργήσουμε έναν κεντρικό κόμβο παραγωγής των ViewModels. Σκοπός είναι κάθε ViewModel να είναι υπεύθυνο για συγκεκριμένες λειτουργίες χωρίς να εισάγεται περιττό κόστος ή αλληλεπίδραση.

***Ζ)* Παρακολούθηση Δεδομένων σε Πραγματικό Χρόνο με Flow & Διαχείριση Βάσης**
Ένας από τους πιο σημαντικούς παράγοντες που βελτίωσε την εφαρμογή είναι η χρήση του Flow για την παρακολούθηση δεδομένων σε πραγματικό χρόνο από τη βάση δεδομένων Room. Ενσωματώσαμε το Flow για να διασφαλίσουμε ότι οποιαδήποτε αλλαγή στα

δεδομένα να ανανεώνει αυτόματα το UI. Ωστόσο, η πρόκληση σε αυτήν την υλοποίηση ήταν η σωστή διαχείριση της παρακολούθησης δεδομένων και της συνεχιζόμενης ροής τους μέσω των State. Η χρήση του Flow βοήθησε να διατηρηθεί η εφαρμογή ευέλικτη, αποδοτική και με διαρκή ανανέωση δεδομένων χωρίς να επιβαρύνεται η απόδοση του συστήματος.

H) **Gradle Dependencies SDK Versions για Compatibility και Reconfiguration**
Η διαχείριση των dependencies στο Gradle αποτέλεσε μία σημαντική πρόκληση, ειδικά για τη διατήρηση της συμβατότητας με διαφορετικές εκδόσεις του Android SDK. Οπότε επιλέχθηκαν οι κατάλληλες εκδόσεις βιβλιοθηκών για συμβατότητα με τη βάση της εφαρμογής και το target API level.

Θ) **Οργάνωση Βάσης και Αρχιτεκτονικής (DAO -> Repository -> ViewModel -> UI)**
Η υλοποίηση της ροής δεδομένων από τη βάση μέχρι την προβολή στον χρήστη αποτέλεσε μία σημαντική καινοτομία και πρόκληση: Η βάση δεδομένων (Room) παρέχει τα δεδομένα μέσω DAO. Τα Repositories αναλαμβάνουν την ενδιάμεση διαχείριση και τη σύνδεση με το ViewModel. Το ViewModel χειρίζεται τη λογική και την παρουσίαση, διασφαλίζοντας ότι το UI παραμένει καθαρό και απλό. Η αυστηρή τήρηση αυτού του σχεδιαστικού προτύπου βελτίωσε την αναγνωσιμότητα του κώδικα, τη συντηρησιμότητα και την επεκτασιμότητα.

I) **Χρήση Date Format αντί για java.time**
Λόγω περιορισμών και συμβατότητας με παλαιότερες εκδόσεις του Android, επιλέχθηκε η χρήση του Date Format αντί της java.time για τη διαχείριση ημερομηνιών. Κάτι που απαιτούσε προσαρμογή και χρήση APIs που ήταν διαθέσιμα σε περισσότερες εκδόσεις του Android, ειδική μέριμνα για την αποφυγή σφαλμάτων κατά τη μορφοποίηση και την επεξεργασία των ημερομηνιών, καθώς και για τη σωστή διαχείριση διαφορετικών χρονικών ζωνών.

4.2 Γενικές παρατηρήσεις

Έγινε η χρήση πολλών βιβλιοθηκών της Kotlin με σημαντικότερες τις androidx.room.Room, kotlinx.coroutines.flow.Flow, androidx.room.Dao, androidx.compose.foundation.layout.Box, androidx.lifecycle.ViewModelProvider, androidx.compose.ui, androidx.compose.material3.*, androidx.navigation. Με βάση τις απαιτήσεις, η εφαρμογή αποτελείται από 5 οθόνες , καθεμία περιλαμβάνει αλληλεπίδραση με το χρήστη με τη μορφή κάποιου γραφικού στοιχείου με drop down menu, κουμπιά, κείμενο, εικόνες και βίντεο με την κατάλληλη διάταξη αυτών και υπάρχουν δεδομένα διαχείρισης state στις οθόνες. Αντικειμενοστραφής προγραμματισμός χρησιμοποιείται για την δημιουργία data κλάσεων, ενδεικτικά της Entry. Χρησιμοποιείται τοπική βάση δεδομένων για τη διαχείριση των δεδομένων της εφαρμογής της οποίας ο κώδικας είναι στο φάκελο app/src/main/java/com/example/mootraki/data. Λαμβάνονται δεδομένα από το διαδίκτυο για τη λειτουργία Affirmations και για την εύρυθμη λειτουργία και απόδοση της εφαρμογής γίνεται συγχρονισμός (concurrency) με τη χρήση coroutines στις οθόνες με βασικό παράδειγμα την VideoPlayer. Τέλος, έχει υλοποιηθεί πλοήγηση της εφαρμογής από οθόνη σε οθόνη με κατάλληλη ένδειξη της οθόνης στην οποία βρίσκεται ο χρήστης.

5. ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ

Μεταβείτε στο αποθετήριο :
https://github.com/mariakalafati/MootrakiFinalVersion/tree/master

Από τα branches επιλέξτε το master και έπειτα στο κουμπί code πατήστε download.zip. Κάντε unzip το πακέτο και μεταβείτε στο android studio Project→Open και μεταβείτε στο path του υπολογιστή σας που είναι ο κατεβασμένος φάκελος. Τον επιλέγετε και πατήστε OK. Το αυτόματο set up του project ίσως χρειαστεί κάποια λεπτά. Μόλις κατεβούν τα configurations, η εφαρμογή είναι έτοιμη προς εκτέλεση.

5.1 Σημειώσεις Λειτουργίας

- Για την εκτέλεση του παραδείγματος της οθόνης Charts, πρέπει πρώτα να μεταβείτε στην οθόνη Calendar και να πατήσετε το κουμπί “Demo: Insert Dummy Data” για να εισαχθούν στη βάση τα ειδικά ενσωματωμένα δεδομένα για την εμφάνιση των γραφημάτων. Έπειτα στην οθόνη charts από το drop down menu των moods (emoji) επιλέξτε το 3^{ο} στη σειρά (" 😐 ") και στο drop down menu των emotions, το “Stressed”.
- Προτείνεται για την οθόνη Breathing, ο χρήστης πρώτα να πατάει pause (σταματώντας το βίντεο και το timer) σε περίπτωση που θέλει να επιλέξει από το drop down menu κάποιο άλλο breathing exercise.