ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Εφαρμογή Carpooling για Οργάνωση Κοινών Διαδρομών»

Του φοιτητή
Χρυσόστομου Αλέξανδρου Τράντου
Αρ. Μητρώου: 174927

Επιβλέπων
Ονοματεπώνυμο Ευκλείδης Κεραμόπουλος
Βαθμίδα Καθηγητής

Ημερομηνία 23/01/2026

«Εφαρμογή Carpooling για Οργάνωση Κοινών Διαδρομών»
Κωδικός Δ.Ε. 24331

Ονοματεπώνυμο φοιτητή
Χρυσόστομος Αλέξανδρος Τράντος

Ονοματεπώνυμο εισηγητή
Ευκλείδης Κεραμόπουλος

Ημερομηνία ανάληψης Δ.Ε.
09/12/2024
Ημερομηνία περάτωσης Δ.Ε.
23/01/2026

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

«Αφιερώνεται στην οικογένειά μου για τη διαρκή στήριξη και ενθάρρυνση καθ'όλη τη διάρκεια των σπουδών μου.»

Πρόλογος

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο των σπουδών μου στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε. Σίνδου. Η ιδέα προέκυψε από ένα υπαρκτό πρόβλημα που αντιμετωπίζουν καθημερινά οι φοιτητές του ιδρύματος: την ανάγκη για αξιόπιστη και οικονομική μετακίνηση. Μέσα από την ανάπτυξη μιας εφαρμογής carpooling, μου δόθηκε η δυνατότητα να εφαρμόσω τεχνολογίες mobile ανάπτυξης, να οργανώσω ένα πλήρες project, και να αξιοποιήσω στην πράξη γνώσεις που απέκτησα κατά τη διάρκεια των σπουδών μου. Η εργασία αυτή ενίσχυσε σημαντικά τις τεχνικές και οργανωτικές μου δεξιότητες, και αποτέλεσε ένα ουσιαστικό βήμα προετοιμασίας για το επόμενο επαγγελματικό μου στάδιο.

Περίληψη

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη μιας εφαρμογής carpooling, με χρήση του Flutter, που απευθύνεται αποκλειστικά στους φοιτητές του ΔΙ.ΠΑ.Ε. Σίνδου. Στόχος της εφαρμογής είναι να προσφέρει μια εύχρηστη και αποδοτική πλατφόρμα κοινής χρήσης οχημάτων, με έμφαση στην ασφάλεια, την απλότητα και την αξιοπιστία.

Η εργασία περιλαμβάνει την ανάλυση των λειτουργικών και μη λειτουργικών απαιτήσεων, τον σχεδιασμό της αρχιτεκτονικής, την υλοποίηση των βασικών λειτουργιών (όπως εγγραφή χρηστών, ανάρτηση και εύρεση διαδρομών, σύστημα ειδοποιήσεων και chat), καθώς και τη σύνδεση με το Firebase για αποθήκευση και διαχείριση δεδομένων.

Η εφαρμογή είναι λειτουργική και δοκιμασμένη σε Android και iOS συσκευές, υποστηρίζει πιστοποίηση μέσω email, δυναμική εμφάνιση διαδρομών στον χάρτη και βασική επικοινωνία μεταξύ οδηγών και επιβατών. Τα αποτελέσματα αξιολόγησης δείχνουν πως η εφαρμογή ανταποκρίνεται πλήρως στις ανάγκες των φοιτητών. Τέλος, παρατίθενται προτάσεις για μελλοντική επέκταση της εφαρμογής με πρόσθετες λειτουργίες.

«Carpooling Application for Organizing Shared Routes»

«Chrysostomos Alexandros Trantos»

Abstract

This thesis focuses on the development of a carpooling mobile application using Flutter, designed specifically for students of the International Hellenic University (IHU) in Sindos. The aim of the application is to provide a user-friendly and reliable platform for vehicle sharing, tailored to the needs and habits of the university's student population.

The work includes a detailed analysis of both functional and non-functional requirements, the design of the system architecture, implementation of core features (such as user authentication, route creation and discovery, real-time messaging, and notifications), as well as integration with Firebase for data storage and user management.

The final app is fully operational on Android and iOS devices, supporting secure email-based login, interactive map views for trip visualization, and communication between drivers and passengers. Evaluation results show that the app successfully addresses the core transportation issues faced by students. Future improvements are also proposed, including enhanced user ratings, ride scheduling, and automated matching algorithms.

Ευχαριστίες

Επιθυμώ να εκφράσω τις ειλικρινείς μου ευχαριστίες στον επιβλέποντα καθηγητή μου για την πολύτιμη καθοδήγηση, την υποστήριξη και τις εύστοχες παρατηρήσεις του καθ' όλη τη διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας. Η συμβολή του υπήρξε καθοριστική τόσο σε τεχνικό όσο και σε ερευνητικό επίπεδο.

Θα ήθελα επίσης να ευχαριστήσω την οικογένειά μου για την αδιάκοπη στήριξή της σε όλα τα στάδια των σπουδών μου, καθώς και τους φίλους και συμφοιτητές μου για την ενθάρρυνση, τις ιδέες και τη συνεισφορά τους κατά την ανάπτυξη της εφαρμογής.

Η συμβολή όλων υπήρξε ουσιώδης στην επιτυχή ολοκλήρωση αυτής της προσπάθειας.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Συντομογραφίες

ΤΠΕ	Τεχνολογίες Πληροφορικής και Επικοινωνιών
GPS	Global Positioning System
ΜΜΜ	Μέσα Μαζικής Μεταφοράς
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο της Ελλάδος
Δ.Ε.	Διπλωματική Εργασία
UI	User Interface
API	Application Programming Interface
cloud	Cloud Computing (όρος τεχνολογίας)
Ι.Χ.	Ιδιωτικής Χρήσης (όχημα)
IHU Apps	Πλατφόρμα αυθεντικοποίησης χρηστών του ΔΙΠΑΕ

Κεφάλαιο 1ο: Εισαγωγή και Θεωρητικό Πλαίσιο

1.1 Εισαγωγή

Η συνεχής εξέλιξη των Τεχνολογιών Πληροφορικής και Επικοινωνιών (ΤΠΕ) και η διάχυτη χρήση «έξυπνων» κινητών συσκευών έχουν μετασχηματίσει τον τρόπο με τον οποίο οι άνθρωποι επικοινωνούν, εργάζονται και μετακινούνται στην καθημερινότητά τους. Τα τελευταία χρόνια, έννοιες όπως η έξυπνη κινητικότητα (smart mobility) και οι ψηφιακά υποστηριζόμενες υπηρεσίες μεταφοράς έχουν ενσωματωθεί δυναμικά στα σύγχρονα αστικά οικοσυστήματα, προσφέροντας πιο αποδοτικές, οικονομικές και περιβαλλοντικά βιώσιμες λύσεις μετακίνησης [1]. Παράλληλα, η αυξανόμενη κυκλοφοριακή συμφόρηση, το υψηλό κόστος καυσίμων και η ανάγκη περιορισμού των εκπομπών διοξειδίου του άνθρακα καθιστούν επιτακτική την αναζήτηση εναλλακτικών μοντέλων αστικής μεταφοράς που βασίζονται στη συνεργατική χρήση πόρων.

Στο πλαίσιο αυτό, το carpooling (η κοινή χρήση ενός οχήματος από άτομα με παρόμοια ή ταυτόσημη διαδρομή) έχει αναδειχθεί ως μία από τις πλέον αποτελεσματικές προσεγγίσεις μείωσης του κόστους και του περιβαλλοντικού αποτυπώματος των μετακινήσεων. Η πρακτική αυτή συμβάλλει σημαντικά στη μείωση της κυκλοφοριακής συμφόρησης, του συνολικού αριθμού οχημάτων στους δρόμους και της ενεργειακής κατανάλωσης, ενώ ταυτόχρονα ενθαρρύνει τη συνεργατική συμπεριφορά και την κοινωνική αλληλεπίδραση [2]. Η αξιοποίηση σύγχρονων ψηφιακών τεχνολογιών, όπως υπηρεσίες γεωεντοπισμού (GPS), υποδομές cloud, real-time βάσεις δεδομένων και εφαρμογές κινητών, έχει ενισχύσει περαιτέρω τη διάδοση του carpooling, καθιστώντας δυνατή την αυτοματοποιημένη αντιστοίχιση οδηγών και επιβατών, την άμεση επικοινωνία και τον έξυπνο προγραμματισμό διαδρομών [3].

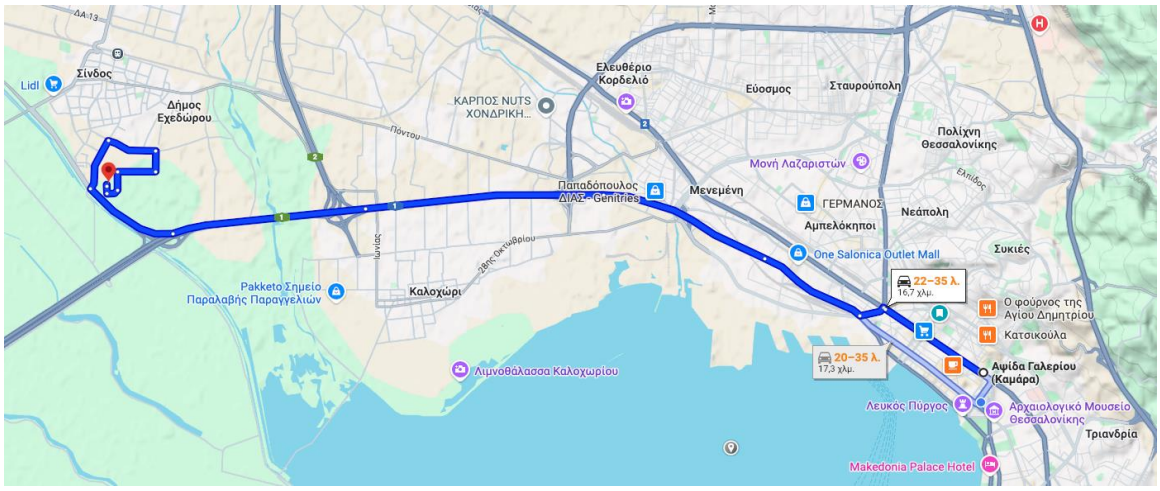
Η ένταξη τέτοιων λύσεων στο πανεπιστημιακό περιβάλλον παρουσιάζει ιδιαίτερο ενδιαφέρον, καθώς η πρόσβαση στις ακαδημαϊκές εγκαταστάσεις αποτελεί καθημερινή πρόκληση για μεγάλο μέρος του φοιτητικού πληθυσμού. Πολλοί φοιτητές διαμένουν σε απομακρυσμένες περιοχές ή εξαρτώνται από περιορισμένες και συχνά ασταθείς επιλογές Μέσων Μαζικής Μεταφοράς (ΜΜΜ). Το αποτέλεσμα είναι αυξημένο κόστος χρόνου και χρημάτων, μειωμένη συνέπεια στην παρακολούθηση μαθημάτων και επιβάρυνση της συνολικής φοιτητικής εμπειρίας [4].

Η κατάσταση αυτή είναι ιδιαίτερα εμφανής στο Διεθνές Πανεπιστήμιο της Ελλάδος (ΔΙΠΑΕ) στη Σίνδο, όπου η πανεπιστημιούπολη βρίσκεται εκτός του αστικού ιστού της Θεσσαλονίκης και η πρόσβαση εξαρτάται κυρίως από ιδιωτικά οχήματα ή περιορισμένα δρομολόγια των ΜΜΜ. Το κόστος μετακίνησης, ο χρόνος πρόσβασης και η έλλειψη επαρκών θέσεων στάθμευσης δημιουργούν μια έντονη ανάγκη για αξιόπιστες, οικονομικές και βιώσιμες λύσεις μετακίνησης.

Η αξιοποίηση σύγχρονων τεχνολογιών ανάπτυξης εφαρμογών, όπως το Flutter (πλαίσιο ανάπτυξης εφαρμογών πολλαπλών πλατφορμών της Google) και το Firebase (cloud πλατφόρμα για αποθήκευση, αυθεντικοποίηση και real-time επικοινωνία), προσφέρει τη δυνατότητα δημιουργίας στοχευμένων εργαλείων που εξυπηρετούν αποτελεσματικά τις ανάγκες της πανεπιστημιακής κοινότητας [5]. Η παρούσα Διπλωματική Εργασία (Δ.Ε.) διερευνά την ανάπτυξη μιας εξειδικευμένης εφαρμογής carpooling για φοιτητές του ΔΙΠΑΕ Σίνδου, με στόχο τη βελτίωση της καθημερινής μετακίνησης, την ενίσχυση της συνεργατικότητας και τη συμβολή στη βιώσιμη κινητικότητα μέσω σύγχρονων ψηφιακών τεχνολογιών.

1.2 Το πρόβλημα των μετακινήσεων φοιτητών στη Σίνδο

Η πρόσβαση στις πανεπιστημιακές εγκαταστάσεις αποτελεί καθοριστικό παράγοντα για την ομαλή ακαδημαϊκή πορεία και τη συνολική εμπειρία φοίτησης. Στην Ελλάδα, σημαντικό ποσοστό φοιτητών διαμένει σε περιοχές εκτός του άμεσου αστικού ιστού των Ανώτατων Εκπαιδευτικών Ιδρυμάτων, γεγονός που συνεπάγεται αυξημένες απαιτήσεις για καθημερινή μετακίνηση συχνά με υψηλό οικονομικό, χρονικό και ψυχολογικό κόστος [6]. Το πρόβλημα γίνεται εντονότερο όταν οι πανεπιστημιουπόλεις δεν διαθέτουν άμεση και συχνή σύνδεση με τα ΜΜΜ ή όταν αυτά λειτουργούν με αραιά και ασυνεπή δρομολόγια.



Σχήμα 1.1: Τυπικός χρόνος μετακίνησης προς το ΔΙΠΑΕ Σίνδου μέσω Google Maps με Ι.Χ.

Σχήμα 1.2: Τυπικός χρόνος μετακίνησης προς το ΔΙΠΑΕ Σίνδου μέσω Google Maps με ΜΜΜ.

Η πανεπιστημιούπολη του ΔΙΠΑΕ στη Σίνδο αποτελεί χαρακτηριστικό παράδειγμα. Απέχει περίπου 16-18 χιλιόμετρα από το κέντρο της Θεσσαλονίκης, με τον μέσο χρόνο μετακίνησης να κυμαίνεται από 35 έως 50 λεπτά, ανάλογα με τις κυκλοφοριακές συνθήκες και την ώρα αιχμής [7]. Για τους φοιτητές που μετακινούνται καθημερινά, ο σωρευτικός εβδομαδιαίος χρόνος μπορεί να φτάσει τις 6-8 ώρες, περιορίζοντας τον διαθέσιμο χρόνο για μελέτη, ξεκούραση και συμμετοχή σε ακαδημαϊκές ή εξωδιδακτικές δραστηριότητες.

Η βασική σύνδεση της Θεσσαλονίκης με το campus πραγματοποιείται μέσω των λεωφορειακών γραμμών του ΟΑΣΘ. Ωστόσο, καταγράφονται επαναλαμβανόμενα προβλήματα, όπως αραιά δρομολόγια, υψηλή πληρότητα, καθυστερήσεις, κυκλοφοριακή συμφόρηση και ανάγκη για μετεπιβίβαση [8]. Το μηνιαίο κόστος χρήσης των ΜΜΜ για έναν φοιτητή εκτιμάται μεταξύ 40 και 60 ευρώ, ενώ η χρήση Ιδιωτικής Χρήσης (Ι.Χ.) οχήματος μπορεί να υπερβεί τα 90 ευρώ μηνιαίως, λαμβάνοντας υπόψη καύσιμα, φθορές και λοιπά έξοδα συντήρησης [9]. Παράλληλα, η περιορισμένη διαθεσιμότητα χώρων στάθμευσης εντός της πανεπιστημιούπολης επιβαρύνει ακόμη περισσότερο την κατάσταση.

Αντίστοιχες προκλήσεις έχουν καταγραφεί και σε πανεπιστήμια του εξωτερικού, οδηγώντας πολλά ιδρύματα στην υιοθέτηση οργανωμένων προγραμμάτων μετακίνησης, όπως πανεπιστημιακά shuttle buses, εκπτώσεις στα δημόσια μέσα, υποδομές μικροκινητικότητας και, σημαντικότερα, επίσημες υπηρεσίες carpooling [10], [11]. Έρευνες δείχνουν ότι η συστηματική εφαρμογή carpooling μπορεί να μειώσει αισθητά τον αριθμό των Ι.Χ. που κινούνται προς και από την πανεπιστημιούπολη, να μετριάσει την κυκλοφοριακή φόρτιση και να ενισχύσει την κοινωνική συνοχή μεταξύ φοιτητών [12].

Στην ελληνική πραγματικότητα, ωστόσο, οι σχετικές πρωτοβουλίες παραμένουν κυρίως άτυπες. Πολλοί φοιτητές οργανώνουν κοινές διαδρομές μέσω ομάδων σε πλατφόρμες κοινωνικής δικτύωσης (π.χ. Facebook, Viber, Instagram), χωρίς τη χρήση επίσημης, δομημένης ή τεχνολογικά υποστηριζόμενης λύσης [13]. Η απουσία μηχανισμών αντιστοίχισης, αξιολόγησης και ελέγχου έχει ως αποτέλεσμα η διαδικασία να είναι συχνά αναξιόπιστη και δύσκολα διαχειρίσιμη.

Συνοψίζοντας, η μετακίνηση των φοιτητών προς το ΔΙΠΑΕ Σίνδου συνιστά ένα πολυδιάστατο πρόβλημα με οικονομικές, κοινωνικές, περιβαλλοντικές και λειτουργικές προεκτάσεις. Η ανάγκη για μια οργανωμένη, ασφαλή και τεχνολογικά υποστηριζόμενη λύση carpooling, ειδικά προσαρμοσμένη στις ανάγκες της φοιτητικής κοινότητας του ΔΙΠΑΕ, προκύπτει ως σαφής και επιτακτική, και αποτελεί το βασικό κίνητρο της παρούσας Δ.Ε..

1.3 Η έννοια του Carpooling και τα οφέλη του

Το carpooling, ή αλλιώς συνεπιβίβαση, αποτελεί μια μορφή κοινής μετακίνησης κατά την οποία δύο ή περισσότερα άτομα μοιράζονται την ίδια διαδρομή χρησιμοποιώντας ιδιωτικό όχημα. Η πρακτική αυτή αξιοποιεί συλλογικά τους διαθέσιμους μεταφορικούς πόρους, μειώνοντας το κόστος μετακίνησης, περιορίζοντας την κυκλοφοριακή συμφόρηση και ενισχύοντας τη βιώσιμη κινητικότητα στα αστικά περιβάλλοντα [13]. Στο πλαίσιο της σύγχρονης έξυπνης κινητικότητας (smart mobility), το carpooling έχει εξελιχθεί σε μια προσιτή και κοινωνικά υπεύθυνη επιλογή για καθημερινές μετακινήσεις, ιδίως όταν υποστηρίζεται από ψηφιακές εφαρμογές και τεχνολογίες αυτοματισμού.

Υπάρχουν διαφορετικοί τύποι carpooling, οι οποίοι διαφοροποιούνται βάσει συχνότητας, οργάνωσης και τεχνολογικής υποστήριξης. Η προγραμματισμένη συνεπιβίβαση (scheduled carpooling) αφορά συμφωνίες μεταξύ ατόμων που πραγματοποιούν τακτικά την ίδια διαδρομή, όπως εργαζόμενοι ή φοιτητές. Αντίθετα, η δυναμική συνεπιβίβαση (real-time carpooling) αξιοποιεί ψηφιακές πλατφόρμες με δυνατότητες GPS και αλγορίθμους αντιστοίχισης, ώστε να συνδέει οδηγούς και επιβάτες σε πραγματικό χρόνο με βάση τη θέση και το χρονικό πλαίσιο [3]. Επιπλέον, οργανωμένες πλατφόρμες carpooling ενσωματώνουν μηχανισμούς επαλήθευσης προφίλ, αξιολόγησης χρηστών και διαχείρισης αιτημάτων, γεγονός που ενισχύει τη διαφάνεια και την εμπιστοσύνη μεταξύ συμμετεχόντων [12].

Τα οφέλη του carpooling αποτυπώνονται σε τρεις βασικούς άξονες: οικονομικό, περιβαλλοντικό και κοινωνικό. Σε οικονομικό επίπεδο, η συνεπιβίβαση επιμερίζει τα έξοδα μετακίνησης-όπως καύσιμα, διόδια και συντήρηση-μειώνοντας σημαντικά το ατομικό κόστος [4]. Από περιβαλλοντικής πλευράς, συμβάλλει στον περιορισμό των εκπομπών CO_2, της κατανάλωσης ενέργειας και της ατμοσφαιρικής ρύπανσης, προωθώντας ένα πιο βιώσιμο μοντέλο αστικής κινητικότητας [2]. Σε κοινωνικό επίπεδο, ενισχύει την αλληλεπίδραση μεταξύ χρηστών, καλλιεργεί κουλτούρα συνεργασίας και δημιουργεί κοινότητες με αμοιβαία εμπιστοσύνη [13].

Ερευνητικά δεδομένα καταδεικνύουν ότι ακόμη και μικρή αύξηση της χρήσης carpooling μπορεί να επιφέρει σημαντικά οφέλη σε επίπεδο αστικής κυκλοφορίας. Για παράδειγμα, αύξηση της συμμετοχής κατά 10% δύναται να μειώσει την κυκλοφοριακή συμφόρηση έως και κατά 5% σε πυκνοκατοικημένες περιοχές [14]. Για τον λόγο αυτό, πολλές χώρες υιοθετούν πολιτικές ενθάρρυνσης -όπως λωρίδες υψηλής πληρότητας (HOV lanes), μειωμένα διόδια και υποδομές μικροκινητικότητας, προκειμένου να ενισχυθεί η συμμετοχή των πολιτών σε οργανωμένα σχήματα συνεπιβίβασης.

Παρά τα σημαντικά οφέλη του, το carpooling συνοδεύεται από προκλήσεις που αφορούν την ασφάλεια, την εμπιστοσύνη μεταξύ αγνώστων, τη διαχείριση ωραρίων και τη συνέπεια [12]. Η ενσωμάτωση τεχνολογικών λύσεων, όπως επαλήθευση λογαριασμών, συστήματα αξιολόγησης, ειδοποιήσεις, real-time ενημερώσεις και αυτοματοποιημένη διαχείριση διαδρομών, έχει αποδειχθεί ότι μειώνει τους κινδύνους και βελτιώνει την αποδοχή τέτοιων συστημάτων.

Στην Ελλάδα, η πρακτική του carpooling βρίσκεται σε διαδικασία σταδιακής ανάπτυξης, ενισχυμένη από οικονομικές πιέσεις, την ανάγκη μείωσης εξόδων μετακίνησης και την αυξημένη ευαισθησία σε ζητήματα περιβαλλοντικής βιωσιμότητας. Αν και δεν υπάρχουν ακόμη θεσμοθετημένες πλατφόρμες carpooling ειδικά για φοιτητικές κοινότητες, παρατηρούνται ανεπίσημες πρωτοβουλίες μέσω κοινωνικών δικτύων και εφαρμογών ανταλλαγής μηνυμάτων, γεγονός που υποδεικνύει υψηλή δυνητική αποδοχή της πρακτικής [13].

Συνολικά, το carpooling αποτελεί μία σύγχρονη, βιώσιμη και κοινωνικά επωφελή προσέγγιση μετακίνησης, πλήρως εναρμονισμένη με τις αρχές της έξυπνης κινητικότητας και της πράσινης ανάπτυξης. Η υιοθέτησή του σε πανεπιστημιακά περιβάλλοντα μπορεί να συμβάλλει ουσιαστικά στη βελτίωση της καθημερινής φοιτητικής μετακίνησης, μειώνοντας το κόστος, ενισχύοντας την προσβασιμότητα και καλλιεργώντας μια κοινότητα συνεργατικότητας και αλληλοϋποστήριξης.

1.4 Υφιστάμενες λύσεις και εφαρμογές Carpooling

Στην παρούσα ενότητα εξετάζονται οι υφιστάμενες λύσεις και εφαρμογές carpooling, με στόχο την αποτύπωση του τρέχοντος τοπίου της συνεργατικής κινητικότητας σε διεθνές και πανεπιστημιακό επίπεδο. Η ανάλυση των εννοιών και η παρουσίαση χαρακτηριστικών παραδειγμάτων συμβάλλουν στην κατανόηση των δομικών χαρακτηριστικών, των προκλήσεων και των δυνατοτήτων που συνοδεύουν την υιοθέτηση τέτοιων συστημάτων. Η χαρτογράφηση των υφιστάμενων πρακτικών αποτελεί θεμέλιο για τη σωστή τοποθέτηση της προτεινόμενης εφαρμογής carpooling στο οικοσύστημα της έξυπνης κινητικότητας, ειδικότερα στο φοιτητικό περιβάλλον του ΔΙΠΑΕ.

1.4.1 Διαχωρισμός εννοιών: Carpooling, Ridesharing και Ride-hailing

Οι όροι που σχετίζονται με τις σύγχρονες υπηρεσίες κινητικότητας χρησιμοποιούνται συχνά εναλλακτικά, γεγονός που μπορεί να οδηγήσει σε σύγχυση σχετικά με το επιχειρησιακό τους μοντέλο και τη λειτουργικότητά τους. Στη διεθνή βιβλιογραφία οι τρεις επικρατέστερες έννοιες είναι οι

carpooling, ridesharing και ride-hailing, οι οποίες παρουσιάζουν ομοιότητες, αλλά διαφοροποιούνται σημαντικά ως προς τον σκοπό, τον βαθμό οργάνωσης και το μοντέλο λειτουργίας [2], [12].

Το carpooling αναφέρεται στην κοινή χρήση ενός ιδιωτικού οχήματος από δύο ή περισσότερα άτομα που πραγματοποιούν παρόμοια διαδρομή, με στόχο τον επιμερισμό των εξόδων μετακίνησης. Η πρακτική αυτή δεν έχει εμπορικό χαρακτήρα, καθώς ο οδηγός δεν αποσκοπεί σε κέρδος και οποιαδήποτε οικονομική συνεισφορά των επιβατών περιορίζεται αποκλειστικά στην κάλυψη εξόδων καυσίμων ή διοδίων [2]. Το carpooling μπορεί να είναι προγραμματισμένο -με σταθερές καθημερινές διαδρομές- ή δυναμικό, μέσω ψηφιακών εφαρμογών που αντιστοιχίζουν οδηγούς και επιβάτες σε πραγματικό χρόνο, αξιοποιώντας υπηρεσίες γεωεντοπισμού [3].

Ο όρος ridesharing χρησιμοποιείται συχνά ως συνώνυμο του carpooling, όμως στη βιβλιογραφία περιγράφει ευρύτερες μορφές κοινής χρήσης μεταφορικών μέσων. Περιλαμβάνει όχι μόνο τη συνεπιβίβαση σε ιδιωτικά οχήματα, αλλά και την κοινή χρήση ποδηλάτων, ηλεκτρικών πατινιών (e-scooters) ή αυτοκινήτων μέσω συνδρομητικών υπηρεσιών [12]. Σε πολλές περιπτώσεις το ridesharing λειτουργεί ως "ομπρέλα" που περιγράφει τεχνολογικά υποστηριζόμενες υπηρεσίες μετακίνησης βασισμένες στην κοινή χρήση.

Αντίθετα, το ride-hailing αποτελεί εμπορική υπηρεσία μεταφοράς “κατ’ απαίτηση” (on-demand), όπου ο χρήστης καλεί οδηγό μέσω εφαρμογής και πληρώνει για τη μεταφορά του. Χαρακτηριστικά παραδείγματα είναι οι υπηρεσίες Uber, Bolt και FreeNow (πρώην Beat στην Ελλάδα) [15]. Στο μοντέλο αυτό, ο οδηγός λειτουργεί επαγγελματικά ή ημι-επαγγελματικά και η τιμολόγηση καθορίζεται από την πλατφόρμα. Το ride-hailing διαφοροποιείται ουσιαστικά από το carpooling, καθώς δεν βασίζεται στην κοινή χρήση διαδρομής, αλλά στην εμπορική παροχή υπηρεσίας μεταφοράς.

Συνοψίζοντας, το carpooling αποτελεί μη εμπορική και συνεργατική πρακτική, το ridesharing λειτουργεί ως ευρύτερος όρος που περιγράφει ποικίλες μορφές κοινής χρήσης μεταφορών και το ride-hailing αφορά επαγγελματική υπηρεσία επί πληρωμή. Η σαφής διάκριση των όρων αυτών είναι κρίσιμη για τη σωστή κατανόηση του πλαισίου εντός του οποίου εντάσσεται η προτεινόμενη εφαρμογή carpooling για το ΔΙΠΑΕ.

1.4.2 Διεθνείς πλατφόρμες Carpooling

Σε διεθνές επίπεδο, η υιοθέτηση του carpooling έχει ενισχυθεί σημαντικά την τελευταία δεκαετία, κυρίως χάρη στην ανάπτυξη ψηφιακών πλατφορμών που διευκολύνουν την εύρεση, οργάνωση και διαχείριση κοινών διαδρομών. Οι πλατφόρμες αυτές αξιοποιούν τεχνολογίες γεωεντοπισμού, αλγορίθμους αντιστοίχισης χρηστών και συστήματα αξιολόγησης, ενισχύοντας την εμπιστοσύνη μεταξύ αγνώστων και καθιστώντας τη συνεπιβίβαση αποτελεσματικότερη και πιο εύκολη στην πράξη [2], [12].

Η BlaBlaCar αποτελεί την πιο δημοφιλή και αναγνωρίσιμη πλατφόρμα carpooling παγκοσμίως. Ιδρύθηκε το 2006 και δραστηριοποιείται σε περισσότερες από 22 χώρες, έχοντας ξεπεράσει τα 100 εκατομμύρια εγγεγραμμένων χρηστών [16]. Η υπηρεσία επικεντρώνεται κυρίως σε μετακινήσεις μεσαίων και μεγάλων αποστάσεων μεταξύ πόλεων, προσφέροντας οικονομικότερη και πιο ευέλικτη εναλλακτική σε σχέση με τα παραδοσιακά ΜΜΜ. Η επιτυχία της βασίζεται σε συστήματα επαλήθευσης χρηστών, αξιολογήσεων, διαφάνειας κόστους και ασφαλών πληρωμών, τα οποία ενισχύουν την αίσθηση αξιοπιστίας και ασφάλειας.

Η Liftshare, μία από τις μεγαλύτερες πλατφόρμες carpooling στο Ηνωμένο Βασίλειο, επικεντρώνεται στη συνεργασία με οργανισμούς, επιχειρήσεις και εκπαιδευτικά ιδρύματα ώστε να μειώσουν το

περιβαλλοντικό τους αποτύπωμα μέσω προγραμμάτων συνεπιβίβασης [17]. Η πλατφόρμα παρέχει εργαλεία για την οργάνωση καθημερινών διαδρομών εργαζομένων και φοιτητών, καθώς και συστήματα παρακολούθησης της εξοικονόμησης κόστους και της μείωσης εκπομπών CO_2.

Η Karos, μια ιδιαιτέρως διαδεδομένη ευρωπαϊκή εφαρμογή με έδρα τη Γαλλία, εστιάζει στις καθημερινές μετακινήσεις προς τον χώρο εργασίας ή σπουδών. Χρησιμοποιεί προηγμένο αλγόριθμο "smart matching", ο οποίος υπολογίζει βέλτιστες αντιστοιχίσεις οδηγών και επιβατών βάσει τοποθεσίας, ωραρίου και προσωπικών συνηθειών μετακίνησης [18]. Η Karos συνεργάζεται με δημόσιους φορείς, δήμους και επιχειρήσεις, προσφέροντας κίνητρα και επιδοτήσεις για τη συστηματική χρήση carpooling.

Επιπλέον, πλατφόρμες όπως οι GoCarma (ΗΠΑ) [19] και sRide (Ινδία) [20] εστιάζουν σε τοπικές ανάγκες μετακίνησης, ενώ ενσωματώνουν προηγμένες δυνατότητες, όπως διαχείριση λωρίδων υψηλής πληρότητας (HOV lanes) και υποστήριξη εταιρικών προγραμμάτων συνεπιβίβασης. Οι λύσεις αυτές δείχνουν ότι το carpooling μπορεί να προσαρμοστεί σε διαφορετικά πολιτισμικά και γεωγραφικά περιβάλλοντα, εξυπηρετώντας τόσο αστικά όσο και προαστιακά δίκτυα μεταφοράς.

Παρά την επιτυχία των παραπάνω υπηρεσιών, η πλειονότητά τους επικεντρώνεται είτε σε μεγάλες αποστάσεις είτε σε εταιρικά ή δημόσια προγράμματα μεγάλης κλίμακας. Αυτό περιορίζει την εφαρμοσιμότητά τους σε περιβάλλοντα με τοπικές, εξειδικευμένες ανάγκες, όπως η καθημερινή μετακίνηση φοιτητών προς πανεπιστημιακές εγκαταστάσεις. Η πραγματικότητα αυτή υπογραμμίζει την ανάγκη για τοπικά στοχευμένες πλατφόρμες carpooling, σχεδιασμένες με βάση τις ανάγκες συγκεκριμένων κοινοτήτων χρηστών - όπως η φοιτητική κοινότητα του ΔΙΠΑΕ.

1.4.3 Ψηφιακές Εμπορικές Υπηρεσίες Μετακίνησης (Ride-hailing & Micromobility)

Οι υπηρεσίες ride-hailing και μικροκινητικότητας αποτελούν δύο σύγχρονες μορφές αστικής κινητικότητας που, παρότι δεν ταυτίζονται με το carpooling, επηρεάζουν σημαντικά το συνολικό οικοσύστημα μεταφορών και τις επιλογές των χρηστών. Οι λύσεις αυτές αξιοποιούν ψηφιακές πλατφόρμες, τεχνολογίες γεωεντοπισμού και συστήματα αυτοματισμού, προσφέροντας γρήγορες, ευέλικτες και εξατομικευμένες επιλογές μετακίνησης, λειτουργώντας ως συμπληρωματικοί μηχανισμοί στα ΜΜΜ και σε άλλες εναλλακτικές μορφές μεταφοράς [12].

Οι υπηρεσίες ride-hailing, όπως οι Uber, Bolt και FreeNow (πρώην Beat στην Ελλάδα), παρέχουν μετακίνηση κατ' απαίτηση μέσω εφαρμογής. Ο χρήστης καλεί όχημα, επιλέγει τύπο διαδρομής και πληρώνει με βάση την απόσταση, τη διάρκεια και τη ζήτηση, βάσει μοντέλων δυναμικής τιμολόγησης [15]. Το βασικό πλεονέκτημά τους είναι η άμεση διαθεσιμότητα και η ευελιξία, καθώς επιτρέπουν ταχεία εξυπηρέτηση χωρίς την ανάγκη ιδιόκτητου οχήματος. Σε αντίθεση με το carpooling, το μοντέλο λειτουργίας του ride-hailing βασίζεται σε επαγγελματίες οδηγούς, με την υπηρεσία να αποτελεί καθαρά εμπορική δραστηριότητα [12].

Η μικροκινητικότητα (micromobility) έχει επίσης γνωρίσει σημαντική ανάπτυξη μέσω κοινόχρηστων μέσων όπως ηλεκτρικά πατίνια, ποδήλατα και e-bikes. Εταιρείες όπως οι Lime, Tier και Bird προσφέρουν τέτοιες υπηρεσίες σε μεγάλες πόλεις, είτε μέσω σταθμών είτε μέσω dockless μοντέλων, όπου ο χρήστης εντοπίζει και ξεκλειδώνει το όχημα μέσω εφαρμογής. Οι υπηρεσίες μικροκινητικότητας εξυπηρετούν κυρίως διαδρομές μικρής απόστασης, λειτουργώντας συμπληρωματικά με τα ΜΜΜ και το ride-hailing και ενισχύοντας την πολυτροπική αστική κινητικότητα [21].

Παρά τα πλεονεκτήματά τους, οι δύο κατηγορίες υπηρεσιών έχουν δεχτεί κριτική για τις επιπτώσεις τους στη βιωσιμότητα των μετακινήσεων. Έρευνες δείχνουν ότι η αυξημένη χρήση ride-hailing μπορεί

να εντείνει την κυκλοφοριακή συμφόρηση και να αποθαρρύνει τη χρήση δημόσιων συγκοινωνιών, ιδίως σε ώρες αιχμής [12]. Από την άλλη πλευρά, η μικροκινητικότητα, παρότι θεωρείται περιβαλλοντικά πιο φιλική, συνοδεύεται από προβληματισμούς που αφορούν τη διάρκεια ζωής των οχημάτων, τη διαχείριση των μπαταριών και την ενεργειακά απαιτητική διαδικασία συλλογής και φόρτισης [21].

Συνολικά, οι υπηρεσίες ride-hailing και μικροκινητικότητας αποτελούν σημαντικά στοιχεία της σύγχρονης πολυτροπικής κινητικότητας, προσφέροντας ευελιξία και αυξημένη προσβασιμότητα. Ωστόσο, δεν υποκαθιστούν το carpooling, το οποίο παραμένει μοναδική λύση συνεργατικής μετακίνησης που συνδυάζει μείωση κόστους, περιορισμό κυκλοφοριακής συμφόρησης και περιβαλλοντική ωφέλεια. Αντίθετα, λειτουργούν συμπληρωματικά, καλύπτοντας διαφορετικές ανάγκες ανάλογα με το είδος, τη διάρκεια και τη συχνότητα των μετακινήσεων.

1.4.4 Εφαρμογές Carpooling σε Πανεπιστήμια και Εκπαιδευτικά Ιδρύματα

Η εφαρμογή του carpooling σε πανεπιστημιακά περιβάλλοντα έχει προσελκύσει αυξανόμενο ενδιαφέρον, καθώς τα ανώτατα εκπαιδευτικά ιδρύματα έρχονται αντιμέτωπα με σημαντικές προκλήσεις στις καθημερινές μετακινήσεις φοιτητών, προσωπικού και επισκεπτών. Παράγοντες όπως η περιορισμένη διαθεσιμότητα χώρων στάθμευσης, το υψηλό κόστος χρήσης Ι.Χ. και η περιβαλλοντική επιβάρυνση καθιστούν τη συνεπιβίβαση μια ιδιαίτερα ελκυστική και βιώσιμη επιλογή [21]. Σε αυτό το πλαίσιο, οργανωμένα προγράμματα carpooling αποτελούν μέρος ευρύτερων στρατηγικών βιώσιμης κινητικότητας (sustainable mobility strategies) που υιοθετούνται διεθνώς από πανεπιστήμια.

Πολλά διεθνή ιδρύματα έχουν υλοποιήσει εξειδικευμένες ψηφιακές πλατφόρμες carpooling ή συνεργάζονται με τεχνολογικές εταιρείες για την υποστήριξη της καθημερινής μετακίνησης της ακαδημαϊκής κοινότητας. Στις Ηνωμένες Πολιτείες, το MIT ανέπτυξε πιλοτικό πρόγραμμα real-time ridesharing, το οποίο αξιοποιεί τεχνολογίες αυτόματης αντιστοίχισης και κινητές εφαρμογές για τη διευκόλυνση των κοινών διαδρομών. Το πρόγραμμα κατέγραψε θετικά αποτελέσματα ως προς τη μείωση της χρήσης ιδιωτικών οχημάτων και την αύξηση της συμμετοχής φοιτητών σε σχήματα συνεπιβίβασης [3]. Παρόμοια πρωτοβουλία εφαρμόζεται στο Stanford University, στο πλαίσιο της στρατηγικής “Sustainable Stanford”, η οποία προσφέρει κίνητρα όπως προνομιακή στάθμευση, μειωμένες χρεώσεις και ειδικά προγράμματα για όσους συμμετέχουν σε ομάδες carpooling [11].

Στην Ευρώπη, ιδρύματα όπως το University of Cambridge και το ETH Zurich έχουν ενσωματώσει οργανωμένες υπηρεσίες carpooling στο πλαίσιο των στρατηγικών τους για βιώσιμη κινητικότητα. Οι υπηρεσίες αυτές επιτρέπουν τον προγραμματισμό καθημερινών διαδρομών μεταξύ φοιτητών και προσωπικού, προσφέρουν εργαλεία παρακολούθησης της εξοικονόμησης CO_2 και ενσωματώνουν μηχανισμούς επιβράβευσης για ενεργούς χρήστες [22], [21]. Σε ορισμένες περιπτώσεις, τα προγράμματα αυτά συνδυάζονται με ευρύτερες υποδομές κινητικότητας, όπως shuttle buses, ποδηλατοδρόμους και ζώνες προνομιακής στάθμευσης carpooling, ενισχύοντας περαιτέρω την αποτελεσματικότητά τους.

Πέρα από τα παραδείγματα των προηγμένων ιδρυμάτων, ιδιαίτερο ενδιαφέρον παρουσιάζουν εφαρμογές σε περιοχές με έντονη κυκλοφοριακή πίεση ή περιορισμένη πρόσβαση σε δημόσια μεταφορά. Έρευνες σε πανεπιστημιακά περιβάλλοντα υψηλής πυκνότητας πληθυσμού δείχνουν ότι η υιοθέτηση συστημάτων carpooling μπορεί να μειώσει τη ζήτηση για χώρους στάθμευσης εντός campus κατά 15–25%, να περιορίσει την κυκλοφοριακή συμφόρηση και να ενισχύσει τη μεταξύ των φοιτητών κοινωνική συνοχή [23]. Η επιτυχία τέτοιων προγραμμάτων εξαρτάται σε μεγάλο βαθμό από τον βαθμό ενσωμάτωσης της τεχνολογίας, την ευκολία χρήσης των πλατφορμών και την παροχή κατάλληλων κινήτρων.

Στην Ελλάδα, αντίστοιχες πρωτοβουλίες παραμένουν περιορισμένες και στην πλειονότητά τους άτυπες. Οι φοιτητές συνήθως οργανώνουν κοινές διαδρομές μέσω ομάδων σε κοινωνικά δίκτυα (π.χ. Facebook, Viber ή Instagram), χωρίς επίσημες ψηφιακές πλατφόρμες, χωρίς μηχανισμούς αξιολόγησης και χωρίς συστήματα ασφαλούς επικοινωνίας [13]. Η απουσία θεσμικού πλαισίου και εξειδικευμένων λύσεων μειώνει την αξιοπιστία και τη συνέπεια τέτοιων πρακτικών, ενώ παράλληλα υπογραμμίζει την ανάγκη για στοχευμένες πλατφόρμες carpooling που θα ανταποκρίνονται στις ανάγκες της ελληνικής φοιτητικής πραγματικότητας.

Συνολικά, τα διεθνή παραδείγματα καταδεικνύουν ότι η επιτυχής εφαρμογή συστημάτων carpooling σε πανεπιστήμια απαιτεί συνδυασμό τεχνολογικής υποδομής, αξιόπιστης αντιστοίχισης χρηστών, κατάλληλων κινήτρων, συστημάτων αξιολόγησης και ενεργού συμμετοχής της πανεπιστημιακής κοινότητας. Τα στοιχεία αυτά αποτελούν βασικά σημεία αναφοράς για τον σχεδιασμό της προτεινόμενης εφαρμογής carpooling στο ΔΙΠΑΕ Σίνδου, η οποία στοχεύει να καλύψει το υφιστάμενο κενό και να ενισχύσει μια πιο βιώσιμη και συνεργατική κουλτούρα μετακίνησης.

1.4.5 Συγκριτική Ανάλυση Υπηρεσιών και Συμπεράσματα

Η ανάπτυξη ψηφιακών υπηρεσιών μετακίνησης έχει δημιουργήσει ένα ιδιαίτερα σύνθετο και πολυδιάστατο οικοσύστημα, στο οποίο εντάσσονται το carpooling, το ride-hailing, το ridesharing και οι υπηρεσίες μικροκινητικότητας. Παρότι οι λύσεις αυτές διαφοροποιούνται ουσιαστικά στον τρόπο λειτουργίας, στο επιχειρησιακό τους μοντέλο και στο προφίλ των χρηστών, ανταγωνίζονται και συνυπάρχουν στο πλαίσιο της σύγχρονης αστικής κινητικότητας. Για την κατανόηση της θέσης και των δυνατοτήτων του carpooling, είναι απαραίτητη η συγκριτική αποτίμηση των υπηρεσιών ως προς κρίσιμες παραμέτρους, όπως το κόστος, η στοχευμένη χρήση, η τεχνολογική υποστήριξη, η περιβαλλοντική επίπτωση και η καταλληλότητα για καθημερινές διαδρομές [2], [10], [12], [24].

Σε αντίθεση με το ride-hailing -το οποίο αποτελεί εμπορική υπηρεσία με επαγγελματίες οδηγούς και υψηλότερο κόστος μετακίνησης- το carpooling στηρίζεται στη συνεργατική χρήση πόρων και στον επιμερισμό εξόδων μεταξύ ιδιωτών. Η πρακτική αυτή μειώνει το κόστος για τους χρήστες, περιορίζει την κυκλοφοριακή συμφόρηση και συμβάλλει στη μείωση των εκπομπών CO_2 [1], [14]. Σε σχέση με τις υπηρεσίες μικροκινητικότητας, οι οποίες εξυπηρετούν κυρίως μικρές αστικές αποστάσεις και λειτουργούν συμπληρωματικά με τα ΜΜΜ, το carpooling προσφέρει λύση για μεσαίες και μεγάλες αποστάσεις χωρίς ανάγκη εναλλαγής μεταφορικών μέσων [12], [25].

Ο Πίνακας 1.1 παρουσιάζει συγκριτική ανάλυση των παραπάνω υπηρεσιών ως προς βασικά κριτήρια αξιολόγησης - μεταξύ των οποίων η στοχευμένη χρήση, το κόστος για τον χρήστη, η περιβαλλοντική επίπτωση, η τεχνολογική πολυπλοκότητα, η ευελιξία, τα ζητήματα ασφάλειας και η καταλληλότητα για φοιτητικές κοινότητες. Η σύγκριση αναδεικνύει τα ιδιαίτερα πλεονεκτήματα και περιορισμούς της κάθε υπηρεσίας και διευκρινίζει τη θέση του carpooling στο μεταφορικό οικοσύστημα [24], [26].

Η ανάλυση δείχνει ότι, παρά την ποικιλία διαθέσιμων λύσεων, καμία από αυτές δεν είναι πλήρως προσαρμοσμένη στις ανάγκες φοιτητικών κοινοτήτων. Οι καθημερινές, επαναλαμβανόμενες διαδρομές, η ανάγκη ταυτοποίησης χρηστών, η ασφάλεια και η εμπιστοσύνη μεταξύ των συμμετεχόντων αποτελούν κρίσιμους παράγοντες που δεν καλύπτονται επαρκώς από τις υφιστάμενες γενικές λύσεις [23], [26]. Ειδικότερα, στο ελληνικό πανεπιστημιακό περιβάλλον δεν υπάρχουν επίσημες πλατφόρμες carpooling, με αποτέλεσμα η οργάνωση κοινών διαδρομών να γίνεται άτυπα μέσω κοινωνικών δικτύων, χωρίς μηχανισμούς ελέγχου και ασφάλειας [13].

Συνολικά, η συγκριτική αποτίμηση επιβεβαιώνει τη στρατηγική σημασία της ανάπτυξης μιας εξειδικευμένης εφαρμογής carpooling για το ΔΙΠΑΕ Σίνδου. Η προτεινόμενη λύση αξιοποιεί τα

πλεονεκτήματα της συνεπιβίβασης, ενσωματώνει σύγχρονες τεχνολογίες πραγματικού χρόνου και ανταποκρίνεται στις ιδιαίτερες απαιτήσεις της φοιτητικής κοινότητας. Η ανάγκη για στοχευμένο, ασφαλές και αξιόπιστο σύστημα carpooling καθιστά σαφή τη συμβολή της παρούσας Δ.Ε. και δημιουργεί το θεωρητικό υπόβαθρο για τα επόμενα κεφάλαια που ακολουθούν.

Πίνακας 1.1: Συγκριτική ανάλυση υπηρεσιών μετακίνησης

Κριτήριο Αξιολόγησης	**Carpooling**	**Ridesharing**	**Ride-hailing**	**Micromobility (e-scooters, bikes)**
Κόστος & Πρόσβαση				
Τύπος υπηρεσίας	Κοινή χρήση ιδιωτικού οχήματος	Κοινόχρηστες μετακινήσεις μέσω πλατφόρμας	Εμπορική υπηρεσία μεταφοράς	Κοινόχρηστα ατομικά οχήματα
Κόστος για τον χρήστη	Πολύ χαμηλό (επιμερισμός εξόδων)	Χαμηλό έως μέτριο	Υψηλό (κόστος ανά διαδρομή)	Μεσαίο (ανάλογα με χρήση)
Οφέλη οδηγού	Κάλυψη εξόδων μόνο	Περιορισμένο	Οικονομικό όφελος/αμοιβή	Δεν υπάρχει οδηγός
Περιβαλλοντική Επίδραση				
Περιβαλλοντικό όφελος / μείωση CO_2	Σημαντική μείωση (μείωση οχημάτων)	Αρνητική επιβάρυνση μειώνεται	Συχνά αυξάνει κίνηση σε πόλεις	Θετικό για μικρές αποστάσεις
Τεχνολογία & Προσβασιμότητα				
Ασφάλεια & Εμπιστοσύνη	Μεσαία (εξαρτάται από ταυτοποίηση)	Μεσαία έως υψηλή	Υψηλή (ρυθμιζόμενη υπηρεσία)	Μεσαία (κίνδυνοι χρήσης)
Ευελιξία & διαθεσιμότητα	Μεσαία (εξαρτάται από προσφορά/ζήτηση)	Καλή	Πολύ υψηλή	Υψηλή για μικρές αποστάσεις
Καταλληλότητα απόστασης	Μεσαίες και μεγάλες	Μεσαίες	Μικρές & μεσαίες	Πολύ μικρές
Χρήστες & Κοινότητα				
Κοινωνική διάδραση	Υψηλή	Μέτρια	Χαμηλή (σχέση πελάτη-οδηγού)	Ελάχιστη
Καταλληλότητα για φοιτητές	Πολύ υψηλή	Καλή	Μέτρια (κόστος)	Χαμηλή (κυρίως σε campus)
Απαιτήσεις τεχνολογίας	Χαμηλές έως μεσαίες	Μεσαίες	Υψηλή	Μεσαίες
Βιωσιμότητα ως κοινότητα λύσης	Υψηλή	Υψηλή	Χαμηλή	Μεσαία

1.5 Σκοπός και αντικειμενικοί στόχοι της εργασίας

Σκοπός της παρούσας Δ.Ε. είναι η σχεδίαση, ανάπτυξη και αξιολόγηση μιας mobile εφαρμογής carpooling, ειδικά προσαρμοσμένης στις ανάγκες των φοιτητών του ΔΙΠΑΕ Σίνδου. Η εφαρμογή αποσκοπεί στη βελτίωση της καθημερινής μετακίνησης, στη μείωση του οικονομικού κόστους, στη διευκόλυνση της σύνδεσης οδηγών και επιβατών και στη γενικότερη ενίσχυση της συνεργατικής κινητικότητας μέσω σύγχρονων τεχνολογιών όπως το Flutter και το Firebase. Επιπλέον, η εργασία επιδιώκει να αναδείξει τον ρόλο του carpooling ως λύση βιώσιμης μετακίνησης στο πανεπιστημιακό περιβάλλον.

Η εργασία διαθέτει ταυτόχρονα τεχνολογική και κοινωνική διάσταση. Σε τεχνολογικό επίπεδο, προτείνει και υλοποιεί μια ολοκληρωμένη λύση έξυπνης κινητικότητας, αξιοποιώντας υπηρεσίες cloud, real-time επικοινωνία και σύγχρονες αρχές σχεδιασμού mobile εφαρμογών [5], [24]. Σε κοινωνικό επίπεδο, η επίσημη εισαγωγή οργανωμένου συστήματος carpooling σε ένα ελληνικό πανεπιστήμιο μπορεί να ενισχύσει την προσβασιμότητα στις ακαδημαϊκές εγκαταστάσεις, να βελτιώσει την κοινωνική αλληλεπίδραση και να μειώσει την περιβαλλοντική επιβάρυνση της καθημερινής μετακίνησης [3], [11], [23]. Η προσέγγιση βρίσκεται σε πλήρη εναρμόνιση με τις στρατηγικές της Ευρωπαϊκής Ένωσης για έξυπνη και πράσινη κινητικότητα [1].

Οι βασικοί **αντικειμενικοί στόχοι** της εργασίας διαμορφώνονται ως εξής:

1. **Ανάλυση αναγκών μετακίνησης της φοιτητικής κοινότητας του ΔΙΠΑΕ Σίνδου:** Διερεύνηση υφιστάμενων συνηθειών μετακίνησης, προβλημάτων, ωρών αιχμής, διαθέσιμων μεταφορικών επιλογών και απαιτήσεων για μια αποτελεσματική λύση carpooling.
2. **Σχεδίαση αρχιτεκτονικής και λειτουργικών προδιαγραφών της εφαρμογής:** Καθορισμός ρόλων χρηστών (οδηγός, επιβάτης), ροών λειτουργίας, δομής δεδομένων, συστημάτων ειδοποιήσεων και αρχιτεκτονικής πραγματικού χρόνου.
3. **Ανάπτυξη mobile εφαρμογής με χρήση Flutter και Firebase:** Υλοποίηση λειτουργιών όπως δημιουργία/αναζήτηση διαδρομών, κράτηση θέσεων, διαχείριση προφίλ, ανταλλαγή δεδομένων σε πραγματικό χρόνο και ενσωμάτωση βασικών μηχανισμών αλληλεπίδρασης [5].
4. **Εφαρμογή μέτρων ασφάλειας και αξιοπιστίας:** Ενσωμάτωση μεθόδων ασφαλούς ταυτοποίησης, προστασίας δεδομένων, κανόνων πρόσβασης και βασικών μηχανισμών εμπιστοσύνης μεταξύ χρηστών, σύμφωνα με βέλτιστες πρακτικές mobile security [27].
5. **Βελτιστοποίηση εμπειρίας χρήσης (User Experience - UX):** Σχεδίαση φιλικού, απλού και λειτουργικού περιβάλλοντος διεπαφής, κατάλληλου για συστηματική χρήση από φοιτητές, με έμφαση στη χρηστικότητα και τη σαφήνεια.
6. **Αξιολόγηση λειτουργικότητας και αποτελεσματικότητας της λύσης:** Διενέργεια δοκιμών, αξιολόγηση επιδόσεων, συλλογή ανατροφοδότησης από χρήστες και καταγραφή πιθανών βελτιώσεων σε επόμενες εκδόσεις.
7. **Διερεύνηση κοινωνικού και περιβαλλοντικού αντίκτυπου:** Ανάλυση των επιπτώσεων της χρήσης carpooling στο κόστος, στον χρόνο μετακίνησης, στην περιβαλλοντική επιβάρυνση και στην κοινωνική συνοχή της φοιτητικής κοινότητας [13], [23].

Η επίτευξη των παραπάνω στόχων αναμένεται να προσφέρει μια ολοκληρωμένη εικόνα για τη σκοπιμότητα και την αποτελεσματικότητα ενός οργανωμένου συστήματος carpooling στο περιβάλλον ενός ελληνικού πανεπιστημίου. Το κεφάλαιο αυτό θέτει τα θεμέλια για τα επόμενα τμήματα της εργασίας, όπου παρουσιάζονται η δομή, η μεθοδολογία και η υλοποίηση της εφαρμογής.

1.6 Δομή της εργασίας

Η παρούσα Δ.Ε. αποτελείται από έξι κεφάλαια, καθένα από τα οποία εξετάζει μια διαφορετική πτυχή της σχεδίασης, ανάπτυξης και αξιολόγησης της προτεινόμενης εφαρμογής carpooling. Συγκεκριμένα:

- Στο Κεφάλαιο 1 παρουσιάζεται το θεωρητικό πλαίσιο της εργασίας, το πρόβλημα της φοιτητικής μετακίνησης στη Σίνδο, η έννοια και τα οφέλη του carpooling, καθώς και οι υφιστάμενες λύσεις κινητικότητας. Επιπλέον, διατυπώνονται ο σκοπός, οι στόχοι και το πεδίο εφαρμογής της εργασίας.
- Στο Κεφάλαιο 2 αναλύονται οι λειτουργικές και μη λειτουργικές απαιτήσεις της εφαρμογής, οι ρόλοι χρηστών, τα use cases, τα σενάρια χρήσης και τα διαγράμματα ροής που περιγράφουν τη λειτουργική δομή του συστήματος.
- Στο Κεφάλαιο 3 περιγράφεται η αρχιτεκτονική της εφαρμογής, οι τεχνολογικές επιλογές, η σχεδίαση της διεπαφής χρήστη, η δομή της βάσης δεδομένων και οι βασικές αρχές ασφάλειας που υιοθετούνται.
- Στο Κεφάλαιο 4 παρουσιάζεται η υλοποίηση του συστήματος, τα βασικά λειτουργικά υποσυστήματα, η ενσωμάτωση με το Firebase και εξωτερικά APIs, ενδεικτικά αποσπάσματα κώδικα, καθώς και οδηγίες εγκατάστασης και χρήσης.
- Στο Κεφάλαιο 5 παρουσιάζεται η τελική μορφή της εφαρμογής, συνοδευόμενη από screenshots, τα αποτελέσματα των δοκιμών, σχόλια χρηστών και αναλυτική αποτίμηση της λειτουργικότητας και της χρηστικότητας του συστήματος.
- Στο Κεφάλαιο 6 παρουσιάζονται τα τελικά συμπεράσματα της εργασίας, τα οφέλη της προτεινόμενης λύσης, οι περιορισμοί και οι προτάσεις για μελλοντική εξέλιξη της εφαρμογής.

Τέλος, ακολουθούν η Βιβλιογραφία και τα Παραρτήματα, στα οποία περιλαμβάνονται πρόσθετα στοιχεία, όπως τμήματα κώδικα, η λεπτομερής δομή της βάσης δεδομένων και ενδεικτικά στιγμιότυπα της εφαρμογής.

1.7 Επίλογος

Στο πρώτο κεφάλαιο παρουσιάστηκε το θεωρητικό και ερευνητικό υπόβαθρο της εργασίας, αναδεικνύοντας την ανάγκη ανάπτυξης μιας οργανωμένης λύσης carpooling στο περιβάλλον του ΔΙΠΑΕ Σίνδου. Μέσα από την ανάλυση του προβλήματος της φοιτητικής μετακίνησης, τον ορισμό και τα οφέλη του carpooling και την επισκόπηση υφιστάμενων υπηρεσιών, καταδείχθηκε το κενό που υπάρχει ως προς εξειδικευμένες λύσεις κινητικότητας για πανεπιστημιακές κοινότητες.

Παράλληλα, ορίστηκαν ο σκοπός και οι αντικειμενικοί στόχοι της εργασίας, οι οποίοι διαμορφώνουν το πλαίσιο πάνω στο οποίο βασίζεται η μεθοδολογική και τεχνική προσέγγιση που ακολουθεί. Το επόμενο κεφάλαιο επικεντρώνεται στην αναλυτική αποτύπωση των απαιτήσεων της εφαρμογής, θέτοντας τα θεμέλια για τον σχεδιασμό και την υλοποίηση του συστήματος.

Κεφάλαιο 2ο: Ανάλυση Απαιτήσεων

2.1 Ρόλοι χρηστών: οδηγός, επιβάτης

Στην προτεινόμενη εφαρμογή carpooling υιοθετείται ένας ενιαίος ρόλος χρήστη, χωρίς μόνιμη διάκριση μεταξύ οδηγού και επιβάτη. Η επιλογή αυτή ευθυγραμμίζεται με πρακτικές που συναντώνται σε πανεπιστημιακές πλατφόρμες ridesharing, όπου ο ίδιος χρήστης μπορεί, ανάλογα με τις ανάγκες του, να προσφέρει ή να ζητά μετακίνηση [23].

Κάθε εγγεγραμμένος χρήστης έχει τη δυνατότητα:

- να αναζητά διαθέσιμες διαδρομές και να συμμετέχει σε αυτές ως επιβάτης, ή
- να δημιουργεί δικές του διαδρομές ως οδηγός, υπό την προϋπόθεση ότι έχει δηλώσει τα στοιχεία του οχήματός του (π.χ. μάρκα/μοντέλο, αριθμό κυκλοφορίας).

Η εγγραφή και η είσοδος στην εφαρμογή πραγματοποιούνται αποκλειστικά μέσω της υπηρεσίας IHU Apps, η οποία παρέχει στο σύστημα ταυτοποιημένα ιδρυματικά στοιχεία (ονοματεπώνυμο, πανεπιστημιακό email κ.λπ.). Με αυτόν τον τρόπο διασφαλίζεται ότι όλοι οι χρήστες ανήκουν στη φοιτητική κοινότητα του ΔΙΠΑΕ, ενώ παράλληλα μειώνεται η ανάγκη χειροκίνητης εισαγωγής δεδομένων και η πολυπλοκότητα της διαδικασίας πιστοποίησης.

Σε λειτουργικό επίπεδο, ο ενιαίος αυτός ρόλος χρήστη μπορεί να αναλαμβάνει δύο διακριτές λειτουργικές συμπεριφορές:

- Ρόλος επιβάτη (Passenger): ο χρήστης αναζητά διαδρομές, υποβάλλει αιτήματα συμμετοχής και ενημερώνεται για την αποδοχή ή απόρριψή τους.
- Ρόλος οδηγού (Driver): ο χρήστης ορίζει νέα rides, διαχειρίζεται διαθέσιμες θέσεις, εξετάζει τα αιτήματα επιβατών και επικαιροποιεί την κατάσταση κάθε διαδρομής.

Με αυτό το μοντέλο, η εφαρμογή υποστηρίζει ένα καθολικό σχήμα χρήστη, το οποίο απλοποιεί την εμπειρία αλληλεπίδρασης, μειώνει την πολυπλοκότητα της υλοποίησης και ανταποκρίνεται στις ανάγκες μιας πανεπιστημιακής υπηρεσίας carpooling, όπου οι ρόλοι μπορούν να εναλλάσσονται δυναμικά.

2.2 Περιγραφή λειτουργικών απαιτήσεων

Στην προτεινόμενη εφαρμογή carpooling υιοθετείται ένας ενιαίος ρόλος χρήστη, χωρίς μόνιμη διάκριση μεταξύ οδηγού και επιβάτη. Η επιλογή αυτή ευθυγραμμίζεται με πρακτικές που συναντώνται σε πανεπιστημιακές πλατφόρμες ridesharing, όπου ο ίδιος χρήστης μπορεί, ανάλογα με τις ανάγκες του, να προσφέρει ή να ζητά μετακίνηση [23].

Κάθε εγγεγραμμένος χρήστης έχει τη δυνατότητα:

- να αναζητά διαθέσιμες διαδρομές και να συμμετέχει σε αυτές ως επιβάτης, ή
- να δημιουργεί δικές του διαδρομές ως οδηγός, υπό την προϋπόθεση ότι έχει δηλώσει τα στοιχεία του οχήματός του (π.χ. μάρκα/μοντέλο, αριθμό κυκλοφορίας).

Η εγγραφή και η είσοδος στην εφαρμογή πραγματοποιούνται αποκλειστικά μέσω της υπηρεσίας IHU Apps, η οποία παρέχει στο σύστημα ταυτοποιημένα ιδρυματικά στοιχεία (ονοματεπώνυμο, πανεπιστημιακό email κ.λπ.). Με αυτόν τον τρόπο διασφαλίζεται ότι όλοι οι χρήστες ανήκουν στη

φοιτητική κοινότητα του ΔΙΠΑΕ, ενώ παράλληλα μειώνεται η ανάγκη χειροκίνητης εισαγωγής δεδομένων και η πολυπλοκότητα της διαδικασίας πιστοποίησης.

Σε λειτουργικό επίπεδο, ο ενιαίος αυτός ρόλος χρήστη μπορεί να αναλαμβάνει δύο διακριτές λειτουργικές συμπεριφορές:

- Ρόλος επιβάτη (Passenger): ο χρήστης αναζητά διαδρομές, υποβάλλει αιτήματα συμμετοχής και ενημερώνεται για την αποδοχή ή απόρριψή τους.
- Ρόλος οδηγού (Driver): ο χρήστης ορίζει νέα rides, διαχειρίζεται διαθέσιμες θέσεις, εξετάζει τα αιτήματα επιβατών και επικαιροποιεί την κατάσταση κάθε διαδρομής.

Με αυτό το μοντέλο, η εφαρμογή υποστηρίζει ένα καθολικό σχήμα χρήστη, το οποίο απλοποιεί την εμπειρία αλληλεπίδρασης, μειώνει την πολυπλοκότητα της υλοποίησης και ανταποκρίνεται στις ανάγκες μιας πανεπιστημιακής υπηρεσίας carpooling, όπου οι ρόλοι μπορούν να εναλλάσσονται δυναμικά.

2.2.1 Διαχείριση χρήστη

Η διαχείριση χρήστη αποτελεί μία από τις κεντρικές λειτουργικές ενότητες της εφαρμογής, καθώς καθορίζει τον τρόπο πρόσβασης στο σύστημα, τη δημιουργία προφίλ και την οργάνωση των διαθέσιμων ρόλων. Η είσοδος στην εφαρμογή πραγματοποιείται αποκλειστικά μέσω της υπηρεσίας IHU Apps, η οποία λειτουργεί ως μηχανισμός ιδρυματικής αυθεντικοποίησης. Με τον τρόπο αυτό διασφαλίζεται ότι μόνο μέλη του ΔΙΠΑΕ μπορούν να χρησιμοποιήσουν την υπηρεσία, ενώ περιορίζεται η ανάγκη διαχείρισης κωδικών πρόσβασης και μειώνεται ο κίνδυνος μη εξουσιοδοτημένης πρόσβασης.

Κατά τη διαδικασία σύνδεσης, η εφαρμογή λαμβάνει αυτόματα βασικά στοιχεία του χρήστη (ονοματεπώνυμο, πανεπιστημιακό email), τα οποία χρησιμοποιούνται για την αρχική δημιουργία της εγγραφής στη βάση δεδομένων. Στη συνέχεια, ο χρήστης μεταφέρεται στην οθόνη δημιουργίας προφίλ, όπου καλείται να δηλώσει αν διαθέτει όχημα και, εφόσον επιθυμεί να λειτουργήσει και ως οδηγός, να εισαγάγει τα στοιχεία του (μοντέλο, αριθμός κυκλοφορίας κ.λπ.).

Η συμπεριφορά του συστήματος διαφοροποιείται ανάλογα με το αν ο χρήστης έχει καταχωρημένο όχημα:

- Χρήστης ως Επιβάτης (Passenger): Μπορεί να αναζητήσει διαθέσιμες διαδρομές, να υποβάλει αιτήματα συμμετοχής και να συμμετέχει σε rides που έχουν δημιουργηθεί από άλλους φοιτητές.
- Χρήστης ως Οδηγός (Driver): Μπορεί να δημιουργεί νέες διαδρομές, να καθορίζει διαθέσιμες θέσεις, να διαχειρίζεται τα αιτήματα επιβατών και να ενημερώνει την κατάσταση της διαδρομής (π.χ. ενεργή, ολοκληρωμένη ή ακυρωμένη).

Το σύστημα επιτρέπει επίσης την τροποποίηση στοιχείων προφίλ σε μεταγενέστερο στάδιο, δίνοντας τη δυνατότητα στον χρήστη να ενημερώσει τα προσωπικά του στοιχεία ή να προσθέσει όχημα εφόσον το αποκτήσει. Επιπλέον, παρέχεται πρόσβαση στο ιστορικό διαδρομών, τόσο αυτών στις οποίες συμμετείχε, όσο και εκείνων που δημιούργησε ως οδηγός.

Με αυτό τον τρόπο, η εφαρμογή υποστηρίζει έναν ενιαίο ρόλο χρήστη με δύο δυνατές λειτουργικές συμπεριφορές, εξασφαλίζοντας ευελιξία και απλότητα στη χρήση, χαρακτηριστικά κρίσιμα για ένα φοιτητικό περιβάλλον.

2.2.2 Διαχείριση διαδρομών (rides)

Η διαχείριση διαδρομών αποτελεί τον πυρήνα της λειτουργικότητας της εφαρμογής, καθώς υποστηρίζει τη δημιουργία, αναζήτηση και οργάνωση των rides μεταξύ οδηγών και επιβατών. Χρήστες που έχουν δηλώσει όχημα μπορούν να δημιουργήσουν νέα διαδρομή, καταχωρίζοντας βασικά στοιχεία όπως:

- το σημείο εκκίνησης (συντεταγμένες και διεύθυνση),
- την ημερομηνία και ώρα της μετακίνησης,
- τον μέγιστο αριθμό διαθέσιμων θέσεων,
- καθώς και αυτόματη σύνδεση της διαδρομής με το email του οδηγού.

Οι διαθέσιμες διαδρομές εμφανίζονται τόσο σε λίστα όσο και στον χάρτη, με δυνατότητα φιλτραρίσματος βάσει χρονικών κριτηρίων (π.χ. ημερομηνία, ώρα). Οι επιβάτες μπορούν να υποβάλλουν αίτημα συμμετοχής, το οποίο καταχωρείται στο αντίστοιχο πεδίο της διαδρομής στη βάση δεδομένων. Ο οδηγός έχει τη δυνατότητα να αποδεχθεί ή να απορρίψει τα αιτήματα. Σε περίπτωση αποδοχής, ο επιβάτης προστίθεται αυτόματα στη λίστα συμμετεχόντων της διαδρομής.

Η εφαρμογή επιτρέπει την επεξεργασία της ημερομηνίας και ώρας μιας διαδρομής. Ωστόσο, αλλαγή του σημείου εκκίνησης απαιτεί τη δημιουργία νέου ride, ώστε να διασφαλίζεται η συνοχή και η αξιοπιστία των δεδομένων. Επιπλέον, ο οδηγός μπορεί να ακυρώσει ή να επισημάνει ως ολοκληρωμένη μια διαδρομή, ενημερώνοντας αντίστοιχα το ιστορικό τόσο του ιδίου όσο και των επιβατών.

2.2.3 Διαχείριση επικοινωνίας (chat)

Η λειτουργία ανταλλαγής μηνυμάτων διευκολύνει τον συντονισμό μεταξύ οδηγού και επιβατών και αποτελεί κρίσιμο στοιχείο για την οργάνωση των rides. Το chat ενεργοποιείται αποκλειστικά για χρήστες που συμμετέχουν στην ίδια διαδρομή, εξασφαλίζοντας ότι η επικοινωνία περιορίζεται σε εμπλεκόμενα μέλη.

Όλα τα μηνύματα αποθηκεύονται στο Firestore, επιτρέποντας στους χρήστες να έχουν πρόσβαση τόσο στις ενεργές όσο και στις παλαιότερες συνομιλίες τους. Η δυνατότητα πρόσβασης σε ιστορικά στοιχεία διευκολύνει τη συνέχεια στην επικοινωνία και παρέχει πλήρη καταγραφή των συζητήσεων που σχετίζονται με κάθε διαδρομή.

2.2.4 Αναφορά προβλήματος

Η εφαρμογή παρέχει τη δυνατότητα υποβολής αναφοράς προβλήματος, επιτρέποντας την καταγραφή ζητημάτων που σχετίζονται με τη λειτουργία του συστήματος ή με τη συμπεριφορά χρηστών. Οι αναφορές αποθηκεύονται σε ειδικό τμήμα της βάσης δεδομένων και μπορούν να αξιοποιηθούν για τη διαχείριση περιστατικών, τον εντοπισμό προβλημάτων και τη μελλοντική βελτίωση της πλατφόρμας.

2.2.5 Λειτουργικότητα χάρτη

Η ενσωμάτωση του χάρτη χρησιμοποιείται για την οπτικοποίηση των διαδρομών που επιλέγει ο χρήστης από τη λίστα. Μετά την επιλογή μιας συγκεκριμένης διαδρομής (ride), η εφαρμογή εμφανίζει στον χάρτη το σημείο εκκίνησης και, όπου είναι διαθέσιμο, τα βασικά χωρικά χαρακτηριστικά της διαδρομής. Ο χάρτης δεν προσφέρει ταυτόχρονη απεικόνιση όλων των διαθέσιμων διαδρομών, αλλά εστιάζει κάθε φορά στην εκάστοτε επιλεγμένη.

Χάρη στον μηχανισμό caching της χαρτογραφικής πλατφόρμας, ο χάρτης μπορεί να παραμείνει διαθέσιμος ακόμη και χωρίς ενεργή σύνδεση στο διαδίκτυο. Η δυνατότητα αυτή ενισχύει τη

χρηστικότητα της εφαρμογής σε περιβάλλοντα με περιορισμένη συνδεσιμότητα, χαρακτηριστικό ιδιαίτερα χρήσιμο για φοιτητές που μετακινούνται σε περιοχές με ασταθές δίκτυο.

2.2.6 Περιορισμοί συμμετοχής σε διαδρομή

Για κάθε ride, το σύστημα τηρεί αυστηρά το μέγιστο όριο θέσεων που έχει ορίσει ο οδηγός κατά τη δημιουργία του. Οποιαδήποτε προσπάθεια υπέρβασης των διαθέσιμων θέσεων αποτρέπεται αυτόματα, τόσο σε επίπεδο αιτημάτων όσο και σε επίπεδο αποδοχής. Ο μηχανισμός αυτός διασφαλίζει την ορθότητα των δεδομένων, αποτρέπει προβλήματα ασυνέπειας και συμβάλλει στην αξιόπιστη λειτουργία της πλατφόρμας.

2.2.7 Ασφάλεια και πρόσβαση

Η εφαρμογή διασφαλίζει ότι κάθε χρήστης έχει πρόσβαση αποκλειστικά στα δικά του προσωπικά δεδομένα και στο προσωπικό του ιστορικό διαδρομών, σύμφωνα με θεμελιώδεις αρχές προστασίας δεδομένων. Όλες οι πληροφορίες (rides, αιτήματα, συνομιλίες, στοιχεία προφίλ) φορτώνονται από το Firestore σε πραγματικό χρόνο, απαιτώντας ενεργή σύνδεση στο διαδίκτυο.

Η προσέγγιση αυτή εγγυάται συνεχή συγχρονισμό μεταξύ συσκευών, ενώ παράλληλα αποτρέπει μη εξουσιοδοτημένη πρόσβαση και διασφαλίζει την ακεραιότητα των δεδομένων που μοιράζονται οι χρήστες κατά τη λειτουργία της πλατφόρμας.

2.3 Μη λειτουργικές απαιτήσεις (ασφάλεια, απόδοση, ευχρηστία)

Οι μη λειτουργικές απαιτήσεις καθορίζουν τις ποιοτικές προδιαγραφές που οφείλει να ικανοποιεί το σύστημα, ώστε να εξασφαλίζει ασφάλεια, σταθερότητα, αποδοτικότητα και ευχρηστία κατά την καθημερινή χρήση. Σε αντίθεση με τις λειτουργικές απαιτήσεις, οι οποίες περιγράφουν συγκεκριμένες ενέργειες και ροές της εφαρμογής, οι μη λειτουργικές απαιτήσεις αφορούν χαρακτηριστικά που επηρεάζουν τη συνολική εμπειρία χρήστη και την αξιοπιστία του συστήματος.

Στο πλαίσιο μιας εφαρμογής carpooling, τα στοιχεία αυτά αποκτούν ιδιαίτερη σημασία, καθώς η υπηρεσία βασίζεται στην ασφαλή διαχείριση δεδομένων, στη γρήγορη ανταλλαγή πληροφορίας μεταξύ χρηστών και στη φιλική διεπαφή χρήστη. Οι βασικοί άξονες που εξετάζονται είναι η ασφάλεια, η απόδοση και η ευχρηστία.

2.3.1 Ασφάλεια

Η ασφάλεια αποτελεί κρίσιμο παράγοντα για την αξιοπιστία της πλατφόρμας και αφορά τόσο την προστασία των προσωπικών δεδομένων όσο και τον έλεγχο πρόσβασης στο σύστημα.

Η εφαρμογή χρησιμοποιεί αποκλειστικά την υπηρεσία IHU Apps ως μηχανισμό ιδρυματικής αυθεντικοποίησης. Με αυτόν τον τρόπο διασφαλίζεται ότι κάθε χρήστης είναι μέλος της ακαδημαϊκής κοινότητας του ΔΙΠΑΕ, ενώ ταυτόχρονα αποφεύγεται η ανάγκη διαχείρισης password από την ίδια την εφαρμογή, πρακτική σύμφωνη με σύγχρονες προδιαγραφές ασφαλείας για mobile services [27].

Η αποθήκευση δεδομένων πραγματοποιείται στο Firebase Firestore, όπου εφαρμόζονται εξειδικευμένοι κανόνες πρόσβασης. Κάθε χρήστης έχει πρόσβαση αποκλειστικά στα δικά του στοιχεία, στις διαδρομές που δημιούργησε ή συμμετείχε και στα μηνύματα που σχετίζονται με ενεργές διαδρομές. Η αρχιτεκτονική αυτή μειώνει την πιθανότητα διαρροής δεδομένων και συμβαδίζει με βασικές αρχές ιδιωτικότητας σε εφαρμογές μετακίνησης.

Σημαντικό πλεονέκτημα αποτελεί και το γεγονός ότι η εφαρμογή δεν συλλέγει ούτε αποθηκεύει σε πραγματικό χρόνο την τοποθεσία του χρήστη. Οι συντεταγμένες που αποθηκεύονται αφορούν αποκλειστικά τα σημεία εκκίνησης των rides, όπως αυτά δηλώνονται από τον οδηγό. Αυτή η σχεδιαστική επιλογή περιορίζει την επεξεργασία ευαίσθητων δεδομένων και ενισχύει το επίπεδο προστασίας ιδιωτικότητας.

2.3.2 Απόδοση

Η απόδοση επηρεάζει άμεσα τη λειτουργικότητα της εφαρμογής και τη συνολική εμπειρία των χρηστών, ειδικά σε ένα περιβάλλον όπου οι ενημερώσεις δεδομένων (rides, αιτήματα, μηνύματα) πρέπει να συγχρονίζονται σε πραγματικό χρόνο.

Η επιλογή της τεχνολογίας Flutter εξασφαλίζει υψηλή ταχύτητα απόκρισης, ομαλή πλοήγηση και σταθερότητα σε συσκευές διαφορετικών προδιαγραφών. Ο rendering engine του framework επιτρέπει ρεαλιστική και αποδοτική απεικόνιση των γραφικών στοιχείων, ενώ η ύπαρξη ενός ενιαίου κώδικα βελτιστοποιεί τον χρόνο φόρτωσης και μειώνει τα περιθώρια σφαλμάτων [5].

Η χρήση του Firebase Firestore επιτρέπει άμεσο συγχρονισμό δεδομένων, διαδρομών, αιτημάτων συμμετοχής και μηνυμάτων, μεταξύ όλων των χρηστών. Το Firestore προσφέρει χαμηλό latency και κλιμακούμενη επίδοση, διασφαλίζοντας ότι κάθε αλλαγή εμφανίζεται άμεσα σε όλα τα συνδεδεμένα clients.

Τέλος, η ενσωμάτωση μηχανισμού caching χαρτογραφικών δεδομένων επιτρέπει την εμφάνιση του χάρτη ακόμη και σε συνθήκες περιορισμένης συνδεσιμότητας, βελτιώνοντας την αξιοπιστία της εφαρμογής σε πραγματικά περιβάλλοντα χρήσης.

2.3.3 Ευχρηστία

Η ευχρηστία (usability) αποτελεί καθοριστικό παράγοντα για την αποδοχή της εφαρμογής από τους φοιτητές, καθώς απευθύνεται σε χρήστες που αναζητούν άμεση, γρήγορη και απλή πλοήγηση. Η σχεδίαση της διεπαφής ακολουθεί βασικές ευρετικές αρχές του Nielsen Norman Group [28], όπως:

- σαφήνεια πληροφορίας,
- συνέπεια στη δομή και στην ορολογία,
- μείωση γνωστικού φορτίου,
- άμεση ανάδραση σε κάθε ενέργεια.

Η κύρια λειτουργικότητα, προβολή διαδρομών, δημιουργία ride, αναζήτηση διαθέσιμων επιλογών και ανταλλαγή μηνυμάτων, είναι άμεσα προσβάσιμη από τις βασικές οθόνες της εφαρμογής. Ο χάρτης προσφέρει οπτική αναπαράσταση των διαθέσιμων rides, ενώ η λίστα επιτρέπει γρήγορη περιήγηση και επιλογή.

Επιπλέον, η εφαρμογή χρησιμοποιεί ενημερωτικά μηνύματα (feedback alerts) για να καθοδηγεί τον χρήστη, είτε πρόκειται για επιτυχείς ενέργειες είτε για αποτροπή λανθασμένων ενεργειών (όπως υπέρβαση διαθέσιμων θέσεων). Αυτό ενισχύει την αποτελεσματικότητα της αλληλεπίδρασης και περιορίζει τα σφάλματα.

2.4 Σενάρια χρήσης (Use Cases)

Τα σενάρια χρήσης αποτελούν θεμελιώδες εργαλείο για την ανάλυση των λειτουργικών απαιτήσεων ενός συστήματος, καθώς περιγράφουν με δομημένο τρόπο τον τρόπο με τον οποίο ο τελικός χρήστης αλληλεπιδρά με τις βασικές λειτουργίες της εφαρμογής. Στο πλαίσιο της προτεινόμενης εφαρμογής

carpooling, τα σενάρια χρήσης τεκμηριώνουν διαδικασίες όπως η είσοδος στην εφαρμογή μέσω του IHU Apps, η δημιουργία και διαχείριση προφίλ, η αναζήτηση και δημιουργία διαδρομών, η συμμετοχή σε rides και η ανταλλαγή μηνυμάτων μεταξύ οδηγού και επιβατών.

Η μεθοδολογία των Use Cases επιτρέπει τη συστηματική καταγραφή ροών, προϋποθέσεων, εναλλακτικών ενεργειών και αποτελεσμάτων, παρέχοντας μια ολοκληρωμένη απεικόνιση της συμπεριφοράς του συστήματος. Η ενότητα περιλαμβάνει τους πρωτεύοντες δρώντες του συστήματος, το συνολικό διάγραμμα Use Cases και τις αναλυτικές περιγραφές των επιμέρους σεναρίων που προσδιορίζουν τη λειτουργική λογική της πλατφόρμας.

2.4.1 Εισαγωγή στα Σενάρια Χρήσης (Use Cases)

Τα σενάρια χρήσης αποτελούν μία από τις πιο διαδεδομένες τεχνικές καταγραφής λειτουργικών απαιτήσεων και βασίζονται στη μεθοδολογία που εισήγαγε ο Ivar Jacobson, σύμφωνα με την οποία ο σχεδιασμός λογισμικού πρέπει να καθοδηγείται από τους στόχους και τις ανάγκες του χρήστη [29]. Η προσέγγιση αυτή επιτρέπει την περιγραφή των λειτουργιών από την οπτική γωνία του τελικού χρήστη και ενισχύει την επικοινωνία ανάμεσα στην ομάδα ανάπτυξης και στους stakeholders.

Στο πλαίσιο της εφαρμογής carpooling, η χρήση Use Cases είναι εξαιρετικά χρήσιμη, καθώς το σύστημα βασίζεται σε αλληλεπιδράσεις μεταξύ διαφορετικών χρηστών που συνεργάζονται για την υλοποίηση μιας κοινής μετακίνησης. Μέσα από τα σενάρια χρήσης αποτυπώνονται λειτουργίες όπως:

- η είσοδος μέσω της υπηρεσίας IHU Apps,
- η δημιουργία και διαχείριση προφίλ,
- η αναζήτηση διαδρομών,
- η δημιουργία ride από οδηγούς,
- η υποβολή/αποδοχή αιτημάτων συμμετοχής και
- η ανταλλαγή μηνυμάτων στο πλαίσιο μιας διαδρομής.

Τα Use Cases περιλαμβάνουν τόσο βασικές όσο και εναλλακτικές ροές εκτέλεσης, καθώς και τις προϋποθέσεις και τα αποτελέσματα κάθε διαδικασίας. Η δομημένη αυτή προσέγγιση εξασφαλίζει σαφή κατανόηση της λειτουργικότητας και υποστηρίζει τον περαιτέρω σχεδιασμό της διεπαφής χρήστη και της αρχιτεκτονικής του συστήματος.

Παράλληλα, η ευθυγράμμιση των ροών με βασικές αρχές ευχρηστίας, όπως αυτές του Nielsen Norman Group [28], συμβάλλει στη διαμόρφωση σεναρίων που αντικατοπτρίζουν ρεαλιστική και φυσική αλληλεπίδραση με το σύστημα.

2.4.2 Πρωτεύοντες Δρώντες του Συστήματος

Ο εντοπισμός και ο καθορισμός των δρώντων (actors) αποτελεί βασικό στάδιο στην ανάλυση Use Cases, καθώς οι δρώντες εκπροσωπούν τις οντότητες που χρησιμοποιούν ή αλληλεπιδρούν με το σύστημα.

Στην παρούσα εφαρμογή carpooling, η οποία απευθύνεται αποκλειστικά στην ακαδημαϊκή κοινότητα του ΔΙΠΑΕ, υιοθετείται ένας ενιαίος πρωτεύων δρώντας: Χρήστης (User)

Ο χρήστης αποτελεί το κεντρικό σημείο αναφοράς όλων των λειτουργιών του συστήματος. Η είσοδος πραγματοποιείται μέσω αυθεντικοποίησης με την υπηρεσία IHU Apps, από την οποία ανακτώνται αυτόματα βασικά στοιχεία (ονοματεπώνυμο και ιδρυματικό email) για τη δημιουργία του προφίλ.

Ο χρήστης μπορεί να λάβει έναν από τους δύο λειτουργικούς ρόλους:

1. Επιβάτης (Passenger)
   Ο χρήστης ως επιβάτης μπορεί να:
   - αναζητεί διαθέσιμες διαδρομές,
   - υποβάλλει αιτήματα συμμετοχής σε rides,
   - ενημερώνεται για την αποδοχή ή απόρριψη του αιτήματός του,
   - επικοινωνεί με τον οδηγό και τους υπόλοιπους επιβάτες μέσω chat.

2. Οδηγός (Driver)
   Ο ρόλος του οδηγού ενεργοποιείται όταν ο χρήστης δηλώσει στοιχεία οχήματος (μοντέλο, πινακίδα). Ως οδηγός μπορεί να:
   - δημιουργεί νέες διαδρομές,
   - ορίζει το σημείο εκκίνησης, την ημερομηνία, ώρα και διαθέσιμες θέσεις,
   - διαχειρίζεται αιτήματα συμμετοχής,
   - επισημαίνει μια διαδρομή ως ολοκληρωμένη ή να την ακυρώνει,
   - επικοινωνεί με τους συμμετέχοντες επιβάτες.

Η υιοθέτηση ενός ενιαίου δρώντα με πολλαπλές λειτουργικές «εκφάνσεις» απλοποιεί τη σχεδίαση, καθιστά το σύστημα πιο ευέλικτο και αντικατοπτρίζει την πραγματικότητα της πανεπιστημιακής κοινότητας, όπου ο ίδιος χρήστης μπορεί να εναλλάσσει ρόλο ανάλογα με τις ανάγκες του.

Η κατανόηση του πρωτεύοντος δρώντα είναι απαραίτητη για την ερμηνεία του συνολικού διαγράμματος Use Cases και των αναλυτικών περιγραφών που ακολουθούν στις επόμενες ενότητες.

2.4.3 Διάγραμμα Σεναρίων Χρήσης (Use Case Diagram)

Το διάγραμμα σεναρίων χρήσης (Use Case Diagram) αποτελεί βασικό εργαλείο της UML για την απεικόνιση των αλληλεπιδράσεων μεταξύ των δρώντων και των λειτουργιών του συστήματος. Στο πλαίσιο της παρούσας εφαρμογής carpooling, το διάγραμμα χρησιμοποιείται για να παρουσιάσει συνοπτικά τις κύριες λειτουργίες που προσφέρονται στον τελικό χρήστη και να αποτυπώσει τη συνολική λειτουργική έκταση της πλατφόρμας.

Καθώς η εφαρμογή υιοθετεί έναν ενιαίο τύπο δρώντα (User), ο οποίος μπορεί να ενσαρκώνει είτε τον ρόλο του επιβάτη είτε του οδηγού ανάλογα με τα στοιχεία του προφίλ του, το διάγραμμα επικεντρώνεται στην απεικόνιση των βασικών ενεργειών που είναι διαθέσιμες σε κάθε χρήστη. Οι λειτουργίες αυτές καλύπτουν ολόκληρο τον κύκλο αλληλεπίδρασης με το σύστημα: από την είσοδο και τη δημιουργία προφίλ, μέχρι την αναζήτηση ή δημιουργία διαδρομών, την αποστολή αιτημάτων συμμετοχής, καθώς και τη διαχείρισή τους.

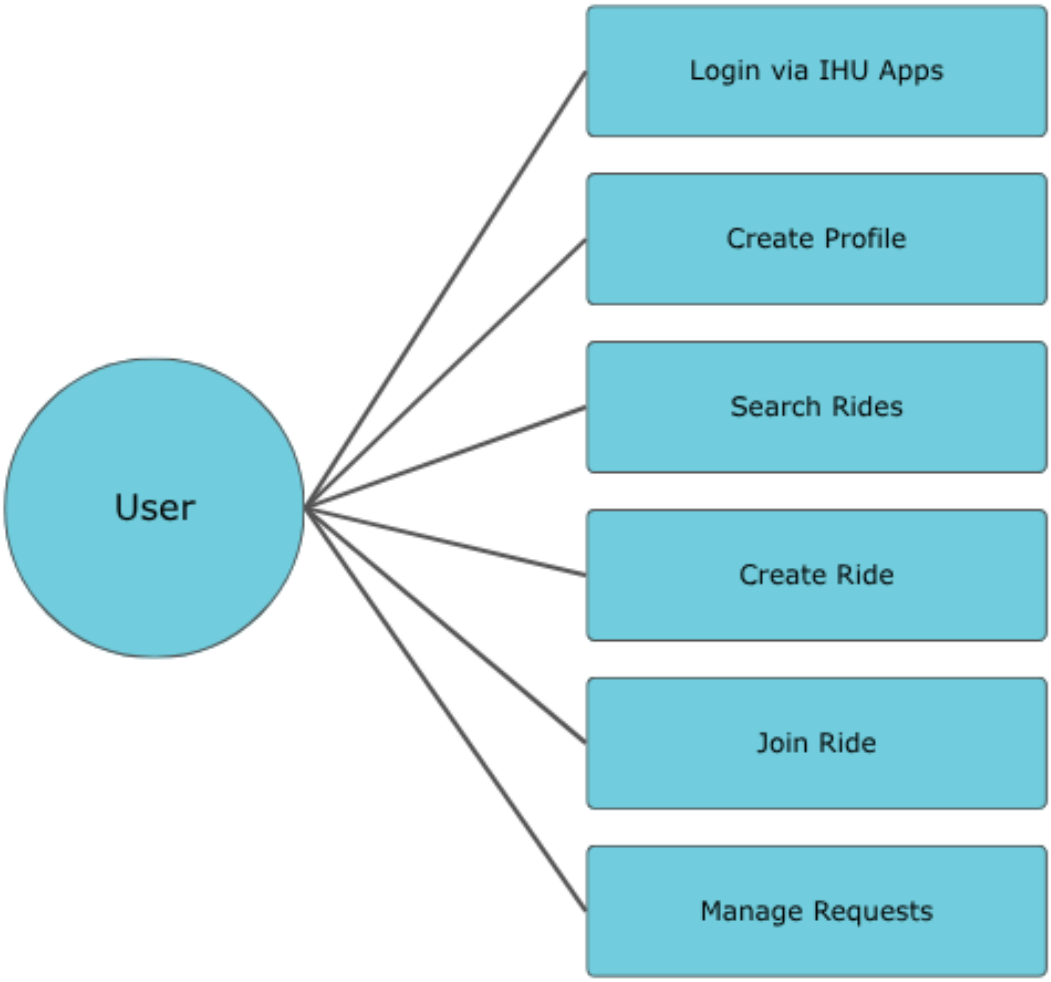
Οι βασικές λειτουργίες που αποτυπώνονται στο διάγραμμα είναι οι εξής:

- **Σύνδεση μέσω IHU Apps (Login via IHU Apps)**
  Ασφαλής είσοδος στο σύστημα με χρήση ιδρυματικών διαπιστευτηρίων.
- **Δημιουργία Προφίλ (Create Profile)**
  Καταχώρηση των βασικών προσωπικών στοιχείων και προαιρετικά των πληροφοριών οχήματος.
- **Αναζήτηση Διαδρομών (Search Rides)**
  Προβολή διαθέσιμων διαδρομών με δυνατότητα φιλτραρίσματος βάσει ημερομηνίας και ώρας.
- **Δημιουργία Διαδρομής (Create Ride)**

Καθορισμός σημείου εκκίνησης, ημερομηνίας, ώρας και αριθμού διαθέσιμων θέσεων.

- **Συμμετοχή σε Διαδρομή (Join Ride)**
  Υποβολή αιτήματος συμμετοχής προς τον οδηγό.
- **Διαχείριση Αιτημάτων (Manage Requests)**
  Αποδοχή ή απόρριψη αιτημάτων συμμετοχής από τον οδηγό.

Η γραφική απεικόνιση στο Σχήμα 2.1 συνοψίζει τις παραπάνω λειτουργίες, προσφέροντας μια καθαρή και ολοκληρωμένη εικόνα της λειτουργικότητας της εφαρμογής. Το διάγραμμα λειτουργεί ως σημείο αναφοράς για τις αναλυτικές περιγραφές των σεναρίων χρήσης που ακολουθούν.



Σχήμα 2.1: Διάγραμμα σεναρίων χρήσης της εφαρμογής carpooling.

2.4.4 Περιγραφή Σεναρίων Χρήσης (Use Case Descriptions)

Η περιγραφή των σεναρίων χρήσης (Use Case Descriptions) αποσκοπεί στην αναλυτική αποτύπωση της συμπεριφοράς του συστήματος κατά την αλληλεπίδρασή του με τον τελικό χρήστη. Κάθε Use Case περιγράφει τις ενέργειες που εκτελεί ο χρήστης, τις αντίστοιχες αντιδράσεις του συστήματος, καθώς και τις προϋποθέσεις, τις εναλλακτικές ροές και τις καταστάσεις ολοκλήρωσης που συνδέονται με κάθε λειτουργία. Η αναλυτική αυτή τεκμηρίωση αποτελεί θεμέλιο για την ορθή κατανόηση της

λειτουργικότητας της εφαρμογής, τον σχεδιασμό της διεπαφής χρήστη, αλλά και την αξιολόγηση της ευχρηστίας και της επεκτασιμότητας του συστήματος.

Στο πλαίσιο της παρούσας Δ.Ε., έχουν εντοπιστεί επτά βασικά σενάρια χρήσης, τα οποία προκύπτουν από τον συνολικό λειτουργικό σχεδιασμό της εφαρμογής carpooling. Τα σενάρια αυτά καλύπτουν τον κύκλο ζωής της αλληλεπίδρασης με το σύστημα, από τη διαδικασία ταυτοποίησης και τη δημιουργία προφίλ, έως τη διαχείριση διαδρομών, την αποστολή και αξιολόγηση αιτημάτων συμμετοχής και τη λειτουργία ολοκλήρωσης των rides.

Οι υποενότητες που ακολουθούν παρουσιάζουν αναλυτικά τα επτά Use Cases (UC1-UC7), με τυποποιημένη δομή που περιλαμβάνει προϋποθέσεις, βασική ροή, εναλλακτικές ροές και τελικές καταστάσεις, υποστηρίζοντας μια σαφή, συστηματική και συνεπή τεκμηρίωση της λειτουργικότητας της εφαρμογής.

2.4.4.1 UC1 - Είσοδος στο σύστημα μέσω της υπηρεσίας IHU Apps

Πρωτεύων δρώντας: Χρήστης (User)

Στόχος: Η ασφαλής και αυτόματη είσοδος του χρήστη στο σύστημα, με τη χρήση των ιδρυματικών διαπιστευτηρίων που παρέχονται από την υπηρεσία IHU Apps.

Προϋποθέσεις (Preconditions):

- Ο χρήστης διαθέτει ενεργό λογαριασμό στο IHU Apps.
- Το κινητό τηλέφωνο είναι συνδεδεμένο στο διαδίκτυο.
- Η εφαρμογή έχει πρόσβαση στις απαιτούμενες υπηρεσίες ταυτοποίησης του IHU.

Βασική Ροή (Basic Flow):

1. Ο χρήστης ανοίγει την εφαρμογή carpooling στο κινητό του.
2. Η εφαρμογή εμφανίζει την επιλογή “Είσοδος μέσω IHU Apps”.
3. Ο χρήστης επιλέγει την είσοδο και η εφαρμογή ανακατευθύνει το αίτημα στο IHU authentication endpoint.
4. Το IHU Apps επαληθεύει ότι ο χρήστης έχει έγκυρα ιδρυματικά credentials.
5. Μετά την επιτυχή ταυτοποίηση, το IHU Apps επιστρέφει στην εφαρμογή:
    - το πανεπιστημιακό email,
    - το ονοματεπώνυμο,
    - τον μοναδικό κωδικό χρήστη (ID).
6. Η εφαρμογή δημιουργεί (ή ενημερώνει) το αντίστοιχο προφίλ στο Firestore αν είναι η πρώτη είσοδος.
7. Ο χρήστης μεταφέρεται αυτόματα στην οθόνη “Create Profile” αν είναι νέος χρήστης, ή στην αρχική οθόνη (Home) εφόσον έχει ολοκληρωμένο προφίλ.

Εναλλακτικές Ροές (Alternative Flows):

A1 - Αποτυχία Ταυτοποίησης από το IHU Apps

1. Το IHU Apps δεν επιβεβαιώνει την ταυτότητα του χρήστη (π.χ. λανθασμένο ή απενεργοποιημένο ιδρυματικό προφίλ).
2. Η εφαρμογή ενημερώνει τον χρήστη ότι η είσοδος δεν ήταν επιτυχής.
3. Ο χρήστης παραμένει στην οθόνη εισόδου.

A2 - Αδυναμία Σύνδεσης στο Διαδίκτυο

1. Ο χρήστης επιλέγει είσοδο, αλλά το κινητό δεν έχει ενεργό δίκτυο.
2. Εμφανίζεται ενημερωτικό μήνυμα για την ανάγκη σύνδεσης.
3. Η διαδικασία εισόδου διακόπτεται μέχρι ο χρήστης να αποκτήσει σύνδεση.

Τελικές Καταστάσεις (Postconditions):

- Η εφαρμογή διαθέτει τα ταυτοποιημένα στοιχεία του χρήστη από το IHU.
- Δημιουργείται ή ενημερώνεται η εγγραφή του στο Firestore.
- Ο χρήστης αποκτά πρόσβαση στις λειτουργίες της εφαρμογής.

2.4.4.2 UC2 - Δημιουργία Προφίλ Χρήστη (Create Profile)

Πρωτεύων δρώντας: Χρήστης (User)

Στόχος: Η αρχική δημιουργία και καταχώρηση του προφίλ του χρήστη στην εφαρμογή, συμπεριλαμβανομένης της προαιρετικής εισαγωγής στοιχείων οχήματος, ώστε να ενεργοποιηθούν πλήρως οι λειτουργίες αναζήτησης και δημιουργίας διαδρομών.

Προϋποθέσεις (Preconditions):

- Ο χρήστης έχει ολοκληρώσει με επιτυχία την είσοδο μέσω της υπηρεσίας IHU Apps (UC1).
- Τα βασικά στοιχεία του χρήστη (ονοματεπώνυμο, email) έχουν ήδη ληφθεί από το IHU Apps.
- Η εφαρμογή είναι συνδεδεμένη στο διαδίκτυο ώστε να μπορεί να ενημερώσει το Firestore.

Βασική Ροή (Basic Flow):

1. Μετά την πρώτη επιτυχή είσοδο, ο χρήστης μεταφέρεται αυτόματα στην οθόνη “Create Profile”.
2. Η εφαρμογή εμφανίζει προ-συμπληρωμένο το ονοματεπώνυμο και το ιδρυματικό email, όπως έχουν ληφθεί από το IHU Apps.
3. Ο χρήστης καλείται να δηλώσει εάν διαθέτει αυτοκίνητο μέσω ενός διακόπτη επιλογής (toggle).
4. Εάν ο χρήστης επιλέξει ότι έχει αυτοκίνητο, εμφανίζονται τα αντίστοιχα πεδία εισαγωγής:
    - Μοντέλο οχήματος
    - Αριθμός πινακίδας
5. Ο χρήστης συμπληρώνει τα στοιχεία (ή τα αφήνει κενά εάν δεν διαθέτει όχημα).
6. Ο χρήστης επιλέγει “Submit”.
7. Η εφαρμογή δημιουργεί στο Firestore το αρχικό προφίλ χρήστη το οποίο περιλαμβάνει:
    - πλήρες ονοματεπώνυμο,
    - ιδρυματικό email,
    - μοναδικό userID,
    - boolean τιμή για το αν διαθέτει όχημα,
    - στοιχεία οχήματος (εφόσον δηλώθηκαν).
8. Μετά τη δημιουργία του προφίλ, ο χρήστης μεταφέρεται στην αρχική οθόνη (Home) της εφαρμογής.

Εναλλακτικές Ροές (Alternative Flows):

A1 - Ο χρήστης δεν διαθέτει όχημα

1. Ο χρήστης αφήνει απενεργοποιημένη την επιλογή “I have a car”.
2. Το σύστημα δημιουργεί προφίλ χωρίς στοιχεία οχήματος.
3. Ο χρήστης έχει πρόσβαση μόνο ως επιβάτης (αναζήτηση και αίτημα συμμετοχής σε rides).

Α2 - Ελλιπή ή μη έγκυρα στοιχεία οχήματος

1. Ο χρήστης επιλέγει ότι διαθέτει αυτοκίνητο, αλλά αφήνει κενά ή λανθασμένα στοιχεία.
2. Η εφαρμογή εμφανίζει σχετικό μήνυμα σφάλματος (π.χ. “Παρακαλώ εισάγετε έγκυρη πινακίδα”).
3. Ο χρήστης διορθώνει τα στοιχεία και επαναλαμβάνει την υποβολή.

Α3 - Αδυναμία επικοινωνίας με το Firestore

1. Κατά την υποβολή του προφίλ, η σύνδεση με το Firestore αποτυγχάνει (π.χ. πρόβλημα δικτύου).
2. Η εφαρμογή ενημερώνει τον χρήστη ότι το προφίλ δεν μπόρεσε να αποθηκευτεί.
3. Ο χρήστης μπορεί να επαναλάβει την υποβολή ή να περιμένει μέχρι την αποκατάσταση της σύνδεσης.

Τελικές Καταστάσεις (Postconditions):

- Το προφίλ χρήστη έχει δημιουργηθεί και αποθηκευτεί επιτυχώς στο Firestore.
- Ο χρήστης μπορεί πλέον να χρησιμοποιεί:
    - την αναζήτηση rides και
    - εφόσον έχει δηλώσει όχημα, τη δημιουργία δικών του διαδρομών.
- Η εφαρμογή αναγνωρίζει πλέον τον χρήστη ως ολοκληρωμένο μέλος του συστήματος carpooling.

2.4.4.3 UC3 - Αναζήτηση Διαθέσιμων Διαδρομών (Search Rides)

Πρωτεύων δρώντας: Χρήστης (User)

Στόχος: Η αναζήτηση και προβολή των διαθέσιμων διαδρομών που έχουν δημιουργηθεί από άλλους χρήστες, με δυνατότητα φιλτραρίσματος βάσει ημερομηνίας και ώρας, ώστε ο χρήστης να εντοπίσει μια διαδρομή που τον εξυπηρετεί μέσω της λίστας διαθέσιμων rides.

Προϋποθέσεις (Preconditions):

- Ο χρήστης έχει ολοκληρωμένο προφίλ (UC2).
- Υπάρχουν διαθέσιμες διαδρομές καταχωρημένες στο Firestore από άλλους χρήστες.
- Η εφαρμογή είναι συνδεδεμένη στο διαδίκτυο ώστε να ανακτά τα δεδομένα σε πραγματικό χρόνο.

Βασική Ροή (Basic Flow):

1. Ο χρήστης μεταβαίνει στην αρχική οθόνη (Home), όπου εμφανίζεται η λίστα διαθέσιμων rides.
2. Η εφαρμογή λαμβάνει από το Firestore όλες τις ενεργές διαδρομές (rides που δεν έχουν ολοκληρωθεί ή ακυρωθεί) και τις προβάλλει αποκλειστικά σε μορφή λίστας (Available Rides).
3. Ο χρήστης έχει τη δυνατότητα να φιλτράρει τις διαδρομές βάσει:
    - ημερομηνίας,
    - ώρας αναχώρησης.
4. Ο χρήστης επιλέγει μία διαδρομή από τη λίστα για να δει περισσότερες πληροφορίες.
5. Με την επιλογή ενός ride, ο χρήστης μεταφέρεται στην οθόνη λεπτομερειών (Ride Details).
6. Η οθόνη προβολής εμφανίζει τις σχετικές πληροφορίες της συγκεκριμένης διαδρομής, όπως:
    - το όνομα του οδηγού,
    - τη διεύθυνση και τις συντεταγμένες αφετηρίας,
    - την ημερομηνία και ώρα αναχώρησης,
    - το μέγιστο αριθμό θέσεων και όσες παραμένουν κενές,

- την κατάσταση της διαδρομής (active, completed)
- και έναν χάρτη που προβάλλει μόνο το σημείο εκκίνησης της συγκεκριμένης επιλεγμένης διαδρομής.

7. Ο χρήστης μπορεί πλέον να επιλέξει "Send Request" για να ζητήσει συμμετοχή στο ride (βλ. UC4).

Εναλλακτικές Ροές (Alternative Flows):

A1 - Δεν υπάρχουν διαθέσιμες διαδρομές

1. Η εφαρμογή δεν εντοπίζει rides στο Firestore που ταιριάζουν στα κριτήρια.
2. Εμφανίζεται σχετικό μήνυμα ("Δεν βρέθηκαν διαθέσιμες διαδρομές").
3. Ο χρήστης μπορεί να αφαιρέσει τα φίλτρα ή να αναζητήσει ξανά αργότερα.

A2 - Σφάλμα σύνδεσης με το Firestore

1. Η εφαρμογή δεν μπορεί να ανακτήσει τα δεδομένα (π.χ. πρόβλημα στο δίκτυο).
2. Εμφανίζεται μήνυμα ενημέρωσης για αδυναμία φόρτωσης.
3. Ο χρήστης μπορεί να επιχειρήσει νέα φόρτωση (refresh) όταν επανέλθει η σύνδεση.

A3 - Φίλτρα που δεν επιστρέφουν αποτελέσματα

1. Ο χρήστης εισάγει ημερομηνία/ώρα που δεν αντιστοιχεί σε καμία διαδρομή.
2. Η λίστα αδειάζει και εμφανίζεται ενημερωτικό μήνυμα.
3. Ο χρήστης μπορεί να τροποποιήσει τα φίλτρα.

Τελικές Καταστάσεις (Postconditions):

- Οι διαθέσιμες διαδρομές έχουν ανακτηθεί επιτυχώς από το Firestore και προβληθεί στον χρήστη.
- Ο χρήστης έχει ενημερωθεί για όλες τις λεπτομέρειες των rides που τον ενδιαφέρουν.
- Ο χρήστης είναι σε θέση να στείλει αίτημα συμμετοχής στο ride (UC4).

2.4.4.4 UC4 - Αίτημα Συμμετοχής σε Διαδρομή (Send Join Request)

Πρωτεύων δρώντας: Χρήστης (User) - Επιβάτης (Passenger)

Στόχος: Η αποστολή αιτήματος συμμετοχής σε μια διαθέσιμη διαδρομή, ώστε ο χρήστης να εκφράσει το ενδιαφέρον του να ταξιδέψει ως επιβάτης και να αναμένει την έγκριση ή απόρριψη από τον οδηγό.

Προϋποθέσεις (Preconditions):

- Ο χρήστης έχει ολοκληρωμένο προφίλ (UC2).
- Ο χρήστης έχει εντοπίσει μια διαθέσιμη διαδρομή μέσω του UC3 (Search Rides).
- Η διαδρομή έχει τουλάχιστον μία κενή θέση.
- Η διαδρομή δεν έχει ολοκληρωθεί, ακυρωθεί ή φτάσει στο μέγιστο αριθμό επιβατών.
- Ο χρήστης δεν έχει ήδη στείλει αίτημα στη συγκεκριμένη διαδρομή.

Βασική Ροή (Basic Flow):

1. Ο χρήστης ανοίγει το "Ride Details" μιας διαθέσιμης διαδρομής.
2. Η εφαρμογή εμφανίζει το κουμπί "Send Request" αν υπάρχουν διαθέσιμες θέσεις.
3. Ο χρήστης επιλέγει "Send Request".
4. Η εφαρμογή ενημερώνει το Firestore, προσθέτοντας το email του χρήστη στη λίστα:
    - requests: [list of pending requests].

5. Ο οδηγός της διαδρομής λαμβάνει άμεση ενημέρωση μέσα από την εφαρμογή ότι υπάρχει νέο αίτημα συμμετοχής.
6. Η κατάσταση του αιτήματος για τον επιβάτη αλλάζει σε “Pending”.
7. Ο χρήστης επιστρέφει στην οθόνη των λεπτομερειών της διαδρομής, όπου εμφανίζεται ότι το αίτημα βρίσκεται σε εκκρεμότητα.

Εναλλακτικές Ροές (Alternative Flows):

A1 - Η διαδρομή δεν έχει διαθέσιμες θέσεις

1. Ο χρήστης ανοίγει το ride, αλλά οι θέσεις έχουν ήδη εξαντληθεί.
2. Το κουμπί “Send Request” απενεργοποιείται ή δεν εμφανίζεται.
3. Η εφαρμογή ενημερώνει με μήνυμα (“Δεν υπάρχουν διαθέσιμες θέσεις”).

A2 - Ο χρήστης έχει ήδη στείλει αίτημα

1. Κατά τον έλεγχο στο Firestore, η εφαρμογή αναγνωρίζει ότι το email του χρήστη υπάρχει ήδη στη λίστα “requests”.
2. Το κουμπί εμφανίζεται ως “Request Sent” και δεν επιτρέπεται νέα αίτηση.

A3 - Απώλεια σύνδεσης κατά την αποστολή του αιτήματος

1. Κατά την υποβολή, η εφαρμογή χάνει τη σύνδεση με το Firestore.
2. Εμφανίζεται μήνυμα αποτυχίας (“Το αίτημα δεν στάλθηκε λόγω προβλήματος σύνδεσης”).
3. Ο χρήστης μπορεί να δοκιμάσει ξανά όταν επανέλθει η σύνδεση.

A4 - Η διαδρομή ακυρώνεται πριν ολοκληρωθεί το αίτημα

1. Ο οδηγός ακυρώνει το ride την ίδια χρονική στιγμή.
2. Η εφαρμογή ενημερώνει τον χρήστη ότι η διαδρομή δεν είναι πλέον διαθέσιμη.
3. Το αίτημα δεν καταχωρείται.

Τελικές Καταστάσεις (Postconditions):

- Το email του χρήστη έχει καταχωρηθεί με επιτυχία στη λίστα requests του συγκεκριμένου ride στο Firestore.
- Η εφαρμογή εμφανίζει την κατάσταση “Pending” για το αίτημα του χρήστη.
- Ο οδηγός της διαδρομής έχει λάβει ενημέρωση για το νέο αίτημα.

2.4.4.5 UC5 - Έγκριση ή Απόρριψη Αιτήματος Συμμετοχής (Approve/Reject Request)

Πρωτεύων δρώντας: Χρήστης (User) - Οδηγός (Driver)

Στόχος: Η διαχείριση των αιτημάτων συμμετοχής σε μια διαδρομή που έχει δημιουργήσει ο οδηγός, επιτρέποντας την έγκριση ή απόρριψή τους και ενημερώνοντας αντίστοιχα τους επιβάτες.

Προϋποθέσεις (Preconditions):

- Ο χρήστης είναι ο δημιουργός της συγκεκριμένης διαδρομής (driver).
- Υπάρχουν εκκρεμή αιτήματα συμμετοχής στη λίστα requests του ride.
- Η εφαρμογή είναι συνδεδεμένη στο διαδίκτυο για επικοινωνία με το Firestore.
- Ο οδηγός έχει πρόσβαση στη σελίδα προφίλ του ή στη διαχείριση των rides του.

Βασική Ροή (Basic Flow):

1. Ο οδηγός μεταβαίνει στο προφίλ του, όπου εμφανίζεται λίστα με τα rides που έχει δημιουργήσει.
2. Επιλέγει τη διαδρομή που θέλει να διαχειριστεί.
3. Η εφαρμογή εμφανίζει τα εκκρεμή αιτήματα συμμετοχής, περιλαμβάνοντας:
    - όνομα επιβάτη (όπως έχει δηλωθεί),
    - email,
    - χρονική στιγμή υποβολής του αιτήματος.
4. Ο οδηγός επιλέγει ένα από τα αιτήματα.
5. Η εφαρμογή εμφανίζει δύο επιλογές:
    - “Approve Request”
    - “Reject Request”
6. Αν ο οδηγός επιλέξει Approve:
    - Το email του επιβάτη μεταφέρεται από τη λίστα requests στη λίστα passengers.
    - Ο αριθμός διαθέσιμων θέσεων μειώνεται κατά 1.
7. Αν ο οδηγός επιλέξει Reject:
    - Το email του επιβάτη αφαιρείται από τη λίστα requests χωρίς να καταχωρηθεί στους passengers.
8. Ο επιβάτης λαμβάνει ενημέρωση μέσα στην εφαρμογή για το αποτέλεσμα (έγκριση ή απόρριψη).
9. Η λίστα αιτημάτων ενημερώνεται σε πραγματικό χρόνο.

Εναλλακτικές Ροές (Alternative Flows):

A1 - Δεν υπάρχουν εκκρεμή αιτήματα

1. Ο οδηγός ανοίγει τη διαχείριση της διαδρομής, αλλά η λίστα “requests” είναι κενή.
2. Εμφανίζεται μήνυμα (“Δεν υπάρχουν εκκρεμή αιτήματα”).
3. Ο οδηγός επιστρέφει στη σελίδα της διαδρομής.

A2 - Το ride έχει γεμίσει πριν ο οδηγός εγκρίνει το αίτημα

(π.χ. εγκρίθηκε από άλλο οδηγό ταυτόχρονα ή έγινε συντονισμένο update στο Firestore)

1. Ο οδηγός πατά “Approve Request”, αλλά οι θέσεις έχουν ήδη μηδενιστεί.
2. Η εφαρμογή ενημερώνει τον οδηγό: “Δεν υπάρχουν διαθέσιμες θέσεις”.
3. Το αίτημα παραμένει σε κατάσταση pending έως ότου απορριφθεί.
4. Ο οδηγός επιλέγει “Reject”.

A3 - Το ride έχει ακυρωθεί

1. Ο οδηγός ακύρωσε προηγουμένως τη διαδρομή ή αυτή ορίστηκε ως completed.
2. Η εφαρμογή μπλοκάρει τη διαχείριση αιτημάτων και ενημερώνει σχετικά.
3. Τα αιτήματα δεν μπορούν να εγκριθούν.

A4 - Αποτυχία σύνδεσης με Firestore

1. Ο οδηγός επιλέγει approve/reject, αλλά το σύστημα δεν μπορεί να ενημερώσει τη βάση.
2. Εμφανίζεται μήνυμα σφάλματος (“Αποτυχία ενημέρωσης. Παρακαλώ δοκιμάστε ξανά”).
3. Ο οδηγός μπορεί να επαναλάβει την ενέργεια όταν η σύνδεση αποκατασταθεί.

Τελικές Καταστάσεις (Postconditions):

- Το Firestore έχει ενημερωθεί με το αποτέλεσμα του αιτήματος.

- Ο αριθμός διαθέσιμων θέσεων έχει ανανεωθεί (αν έγινε approval).
- Ο επιβάτης έχει ενημερωθεί για το αποτέλεσμα.
- Η εφαρμογή έχει ενημερώσει τη λίστα "requests" και "passengers" σε πραγματικό χρόνο.

2.4.4.6 UC6 - Δημιουργία Νέας Διαδρομής (Create Ride)

Πρωτεύων δρώντας: Χρήστης (User) - Οδηγός (Driver)

Στόχος: Η δημιουργία μιας νέας διαδρομής από χρήστη που διαθέτει δηλωμένο όχημα, με καθορισμό σημείου εκκίνησης, ημερομηνίας, ώρας και μέγιστου αριθμού επιβατών.

Προϋποθέσεις (Preconditions):

- Ο χρήστης έχει ολοκληρωμένο προφίλ στο οποίο έχει δηλώσει στοιχεία οχήματος (UC2).
- Ο χρήστης είναι συνδεδεμένος στην εφαρμογή.
- Το σύστημα έχει πρόσβαση στο Firestore για να καταχωρήσει τη νέα διαδρομή.
- Ο χρήστης βρίσκεται στην αρχική οθόνη (Home) της εφαρμογής.

Βασική Ροή (Basic Flow):

1. Ο χρήστης επιλέγει την επιλογή “Create Ride” από την κεντρική οθόνη ή το μενού της εφαρμογής.
2. Εμφανίζεται φόρμα δημιουργίας διαδρομής με τα ακόλουθα πεδία:
    - Σημείο εκκίνησης (επιλογή από χάρτη)
    - Ημερομηνία αναχώρησης
    - Ώρα αναχώρησης
    - Μέγιστος αριθμός επιβατών (max seats)
3. Ο χρήστης επιλέγει το σημείο εκκίνησης πατώντας πάνω στον χάρτη.
4. Η εφαρμογή αποθηκεύει τις συντεταγμένες (startLocationCoords) και ανακτά την αντίστοιχη διεύθυνση (startLocationAddress).
5. Ο χρήστης εισάγει ημερομηνία και ώρα αναχώρησης μέσω date/time picker.
6. Ο χρήστης δηλώνει τον μέγιστο αριθμό διαθέσιμων θέσεων (π.χ. 1-4).
7. Ο χρήστης πατά “Submit”.
8. Η εφαρμογή δημιουργεί στο Firestore μια νέα εγγραφή ride που περιλαμβάνει:
    - rideID (αυτόματα δημιουργημένο)
    - startLocationCoords
    - startLocationAddress
    - rideDate (ημερομηνία και ώρα)
    - riderEmail (email του οδηγού)
    - maxSeats
    - passengers: κενή λίστα
    - requests: κενή λίστα
    - completed: false
9. Ο χρήστης επιστρέφει στην αρχική οθόνη, όπου η διαδρομή εμφανίζεται πλέον ως ενεργή.

Εναλλακτικές Ροές (Alternative Flows):

A1 - Ο χρήστης δεν έχει δηλώσει όχημα

1. Ο χρήστης πατά “Create Ride”.
2. Η εφαρμογή εντοπίζει ότι το προφίλ του δεν περιλαμβάνει στοιχεία οχήματος.

3. Εμφανίζεται σχετικό μήνυμα (“Για να δημιουργήσετε ride πρέπει να δηλώσετε όχημα”).
4. Ο χρήστης μεταφέρεται στην οθόνη προφίλ για να συμπληρώσει τα στοιχεία.

Α2 - Ελλιπή ή μη έγκυρα στοιχεία

1. Ο χρήστης αφήνει κάποιο πεδίο κενό (π.χ. δεν έχει επιλέξει σημείο εκκίνησης).
2. Η εφαρμογή εμφανίζει ειδοποίηση σφάλματος.
3. Η δημιουργία της διαδρομής δεν ολοκληρώνεται μέχρι τη διόρθωση.

Α3 - Πρόβλημα σύνδεσης με Firestore

1. Ο χρήστης πατά “Submit”, αλλά η εφαρμογή αδυνατεί να αποθηκεύσει τα δεδομένα.
2. Εμφανίζεται μήνυμα (“Αποτυχία δημιουργίας διαδρομής. Ελέγξτε τη σύνδεσή σας.”).
3. Ο χρήστης μπορεί να δοκιμάσει ξανά.

Α4 - Ο χρήστης ακυρώνει τη διαδικασία

1. Ο χρήστης αποφασίζει να μην ολοκληρώσει την καταχώρηση και πατά “Cancel” ή επιστρέφει πίσω.
2. Κανένα δεδομένο δεν αποθηκεύεται στο Firestore.

Τελικές Καταστάσεις (Postconditions):

- Η νέα διαδρομή έχει καταχωρηθεί επιτυχώς στο Firestore.
- Η διαδρομή εμφανίζεται στη λίστα ενεργών rides του οδηγού.
- Οι επιβάτες μπορούν πλέον να εντοπίσουν και να κάνουν αίτημα συμμετοχής στο ride (UC3 - UC4).

2.4.4.7 UC7 - Ολοκλήρωση Διαδρομής (Complete Ride)

Πρωτεύων δρώντας: Χρήστης (User) - Οδηγός (Driver)

Στόχος: Η σήμανση μιας διαδρομής ως ολοκληρωμένης μετά την πραγματοποίησή της, ώστε να ενημερωθεί το σύστημα και να μεταφερθεί το ride στο ιστορικό του χρήστη χωρίς να είναι πλέον διαθέσιμο σε νέους επιβάτες.

Προϋποθέσεις (Preconditions):

- Ο χρήστης είναι ο δημιουργός της συγκεκριμένης διαδρομής.
- Η διαδρομή έχει φτάσει στη χρονική στιγμή της προγραμματισμένης αναχώρησης ή έχει ήδη ολοκληρωθεί.
- Η διαδρομή δεν έχει ήδη σημειωθεί ως completed ή ακυρωθεί.
- Η εφαρμογή είναι συνδεδεμένη στο διαδίκτυο ώστε να ενημερώσει το Firestore.

Βασική Ροή (Basic Flow):

1. Ο οδηγός μεταβαίνει στο προφίλ του, όπου εμφανίζεται λίστα με τις διαδρομές που έχει δημιουργήσει.
2. Επιλέγει τη συγκεκριμένη διαδρομή που θέλει να ολοκληρώσει.
3. Η εφαρμογή εμφανίζει τις τρέχουσες λεπτομέρειες της διαδρομής καθώς και την επιλογή “Mark as Completed”.
4. Ο οδηγός επιλέγει “Mark as Completed”.
5. Η εφαρμογή ενημερώνει το Firestore αλλάζοντας την ιδιότητα:
    - completed: true

6. Το ride αφαιρείται από τη λίστα ενεργών διαδρομών και μεταφέρεται στο προσωπικό ιστορικό του οδηγού (Ride History).
7. Οι επιβάτες που έχουν εγκριθεί για τη διαδρομή βλέπουν ότι η διαδρομή έχει ολοκληρωθεί στο δικό τους ιστορικό.
8. Η εφαρμογή επιστρέφει τον οδηγό στη σελίδα των διαδρομών του, όπου εμφανίζεται ενημερωμένη η κατάσταση.
9. Πέντε (5) λεπτά μετά την ολοκλήρωση της διαδρομής, η εφαρμογή αποστέλλει αυτόματα ειδοποίηση (notification) σε όλους τους επιβάτες, ζητώντας αξιολόγηση της εμπειρίας και του οδηγού.

Εναλλακτικές Ροές (Alternative Flows):

A1 - Η διαδρομή έχει ήδη ολοκληρωθεί

1. Ο οδηγός επιχειρεί να ολοκληρώσει μια διαδρομή που έχει ήδη χαρακτηριστεί completed.
2. Η εφαρμογή εμφανίζει μήνυμα (“Η διαδρομή έχει ήδη ολοκληρωθεί.”).
3. Η ενέργεια τερματίζεται.

A2 - Αδυναμία σύνδεσης με το Firestore

1. Κατά την αποστολή της ενημέρωσης, η εφαρμογή δεν μπορεί να επικοινωνήσει με τη βάση δεδομένων.
2. Εμφανίζεται μήνυμα σφάλματος (“Αποτυχία ενημέρωσης. Παρακαλώ δοκιμάστε ξανά.”).
3. Ο οδηγός μπορεί να επαναλάβει την ενέργεια μόλις αποκατασταθεί η σύνδεση.

A3 - Ο χρήστης δεν είναι ο οδηγός της διαδρομής

1. Ο χρήστης προσπαθεί να ολοκληρώσει μια διαδρομή για την οποία δεν είναι οδηγός.
2. Η εφαρμογή αποκρύπτει την επιλογή “Mark as Completed” ή εμφανίζει μήνυμα μη εξουσιοδότησης.
3. Η ενέργεια δεν ολοκληρώνεται.

A4 - Ο οδηγός επιθυμεί να ακυρώσει τη διαδρομή αντί να την ολοκληρώσει

1. Ο οδηγός επιλέγει “Cancel Ride” αντί για “Completed”.
2. Η εφαρμογή ενημερώνει το Firestore (completed = false, αλλά το status αλλάζει σε cancelled).
3. Όλοι οι επιβάτες βλέπουν την ακύρωση στο ιστορικό τους.

Τελικές Καταστάσεις (Postconditions):

- Η διαδρομή έχει σημειωθεί ως ολοκληρωμένη.
- Δεν εμφανίζεται πλέον στις διαθέσιμες διαδρομές ούτε δέχεται νέα αιτήματα.
- Η διαδρομή εμφανίζεται στο ιστορικό του οδηγού και των επιβατών.
- Το σύστημα διαθέτει ενημερωμένα δεδομένα για μελλοντική ανάλυση ή αναφορά.
- Έπειτα από 5 λεπτά, οι επιβάτες λαμβάνουν ειδοποίηση για την αξιολόγηση της διαδρομής και του οδηγού.
- Η μέση βαθμολογία του οδηγού ενημερώνεται στο Firestore και εμφανίζεται πλέον δίπλα σε κάθε διαθέσιμη διαδρομή (Available Ride) στην αρχική οθόνη.

2.4.5 Διαγράμματα Ροής (Flow Diagrams)

Η μοντελοποίηση της ροής των διαδικασιών αποτελεί κρίσιμο στάδιο στην ανάλυση και σχεδίαση λογισμικού, καθώς επιτρέπει την οπτική αποτύπωση των διαδοχικών ενεργειών του χρήστη και των

αντίστοιχων αντιδράσεων του συστήματος. Τα διαγράμματα ροής (Flow Diagrams) χρησιμοποιούνται για να περιγράψουν με σαφήνεια την ακολουθία των βημάτων, τα σημεία λήψης αποφάσεων, τις συνθήκες που μπορούν να μεταβάλουν τη ροή, καθώς και τις τελικές καταστάσεις κάθε διεργασίας. Με αυτόν τον τρόπο αποτελούν πολύτιμο εργαλείο τεκμηρίωσης και διευκολύνουν την επικοινωνία μεταξύ αναλυτών, προγραμματιστών και ομάδων ανάπτυξης.

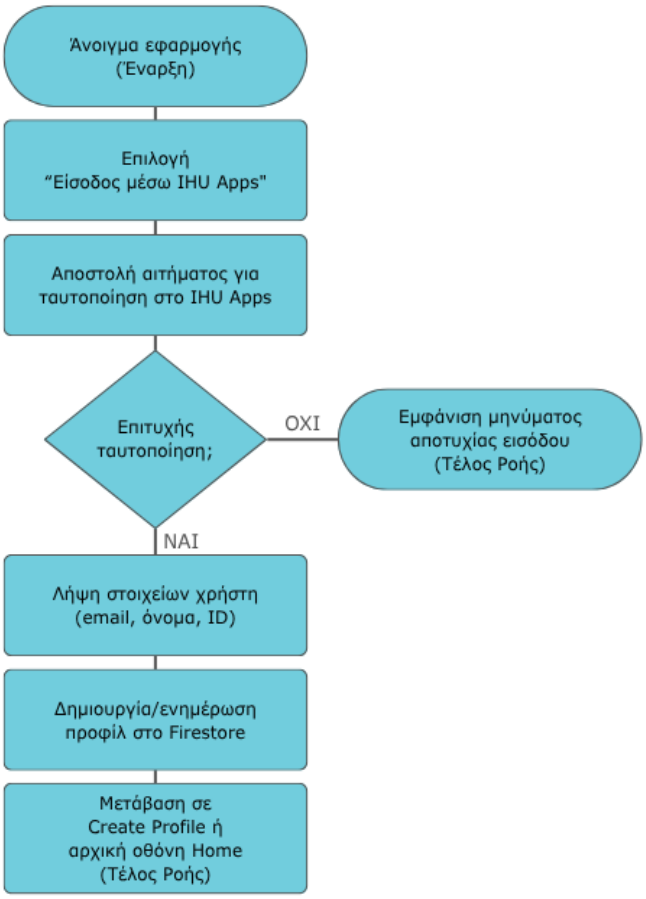
Στο πλαίσιο της παρούσας Δ.Ε., τα διαγράμματα ροής που ακολουθούν αποτυπώνουν τις σημαντικότερες και πιο σύνθετες λειτουργίες της εφαρμογής, προσφέροντας μια καθαρή εικόνα της εσωτερικής λειτουργίας του συστήματος. Συγκεκριμένα, παρουσιάζονται διαγράμματα που αφορούν:

- την είσοδο του χρήστη μέσω της υπηρεσίας IHU Apps,
- την αναζήτηση διαθέσιμων διαδρομών,
- τη δημιουργία νέου ride,
- την υποβολή αιτήματος συμμετοχής και
- τη διαχείριση αιτημάτων από τον οδηγό.

Οι διαδικασίες αυτές συγκροτούν τον λειτουργικό πυρήνα της εφαρμογής και περιλαμβάνουν τα περισσότερα σημεία αλληλεπίδρασης με το Firestore. Η οπτική αναπαράστασή τους συμπληρώνει τις περιγραφές των Use Cases της προηγούμενης ενότητας, διευκολύνοντας την κατανόηση της συνολικής λειτουργικότητας και βελτιώνοντας τη σαφήνεια της τεκμηρίωσης.

2.4.5.1 FD1 - Διάγραμμα Ροής για την Είσοδο Χρήστη μέσω IHU Apps (UC1)

Το παρακάτω διάγραμμα ροής απεικονίζει τη διαδικασία εισόδου του χρήστη στην εφαρμογή μέσω της υπηρεσίας IHU Apps. Η ροή περιλαμβάνει όλα τα βασικά βήματα αυθεντικοποίησης, καθώς και τις πιθανές εναλλακτικές εκβάσεις, όπως αποτυχία σύνδεσης ή απόρριψη της ταυτοποίησης. Το διάγραμμα αποτυπώνει τον τρόπο με τον οποίο ο χρήστης αλληλεπιδρά με την υπηρεσία authentication και πώς τα στοιχεία του μεταφέρονται στο σύστημα της εφαρμογής για τη δημιουργία ή ανάκτηση του προφίλ του (Σχήμα 2.2).



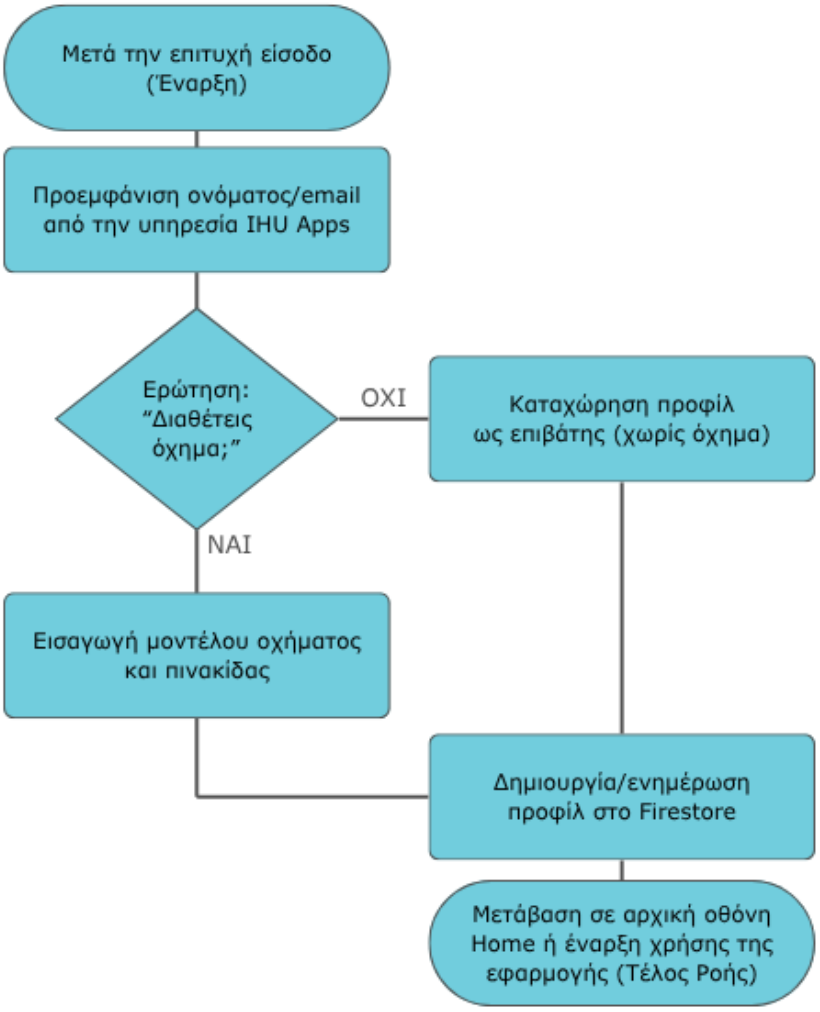
Σχήμα 2.2: FD1 - Διάγραμμα Ροής για την Είσοδο Χρήστη μέσω IHU Apps (UC1).

Το συγκεκριμένο διάγραμμα αποτελεί το πρώτο από τα βασικά μοντέλα ροής της εφαρμογής. Στις υποενότητες που ακολουθούν παρουσιάζονται αντίστοιχα διαγράμματα για τις υπόλοιπες κρίσιμες λειτουργίες του συστήματος, όπως η αναζήτηση διαδρομών, η δημιουργία ride και η διαχείριση αιτημάτων συμμετοχής.

2.4.5.2 FD2 - Διάγραμμα Ροής για τη Δημιουργία Προφίλ Χρήστη (UC2)

Η διαδικασία δημιουργίας προφίλ αποτελεί ένα από τα πρώτα και πιο κρίσιμα βήματα αλληλεπίδρασης του χρήστη με το σύστημα. Αμέσως μετά την επιτυχή είσοδο μέσω της υπηρεσίας IHU Apps, ο χρήστης καλείται να επιβεβαιώσει και να συμπληρώσει τα στοιχεία εκείνα που είναι απαραίτητα για τη λειτουργία της εφαρμογής. Το σύστημα λαμβάνει αυτόματα τα βασικά ιδρυματικά δεδομένα (όνομα και email), ενώ ο χρήστης καλείται να δηλώσει επιπλέον αν διαθέτει όχημα και, εφόσον ισχύει αυτό, να εισαγάγει το μοντέλο και τον αριθμό πινακίδας του.

Η συγκεκριμένη διαδικασία είναι κρίσιμη, καθώς από την ύπαρξη οχήματος εξαρτάται αν ο χρήστης θα έχει δικαίωμα δημιουργίας διαδρομών (rides) ή μόνο συμμετοχής σε αυτές. Το διάγραμμα ροής που ακολουθεί παρουσιάζει τα διαδοχικά βήματα, τις πιθανές διακλαδώσεις και τις τελικές καταστάσεις αυτής της διαδικασίας, επιτρέποντας μια πλήρη και εύληπτη κατανόηση του τρόπου με τον οποίο ολοκληρώνεται η δημιουργία προφίλ στο Firestore (Σχήμα 2.3).

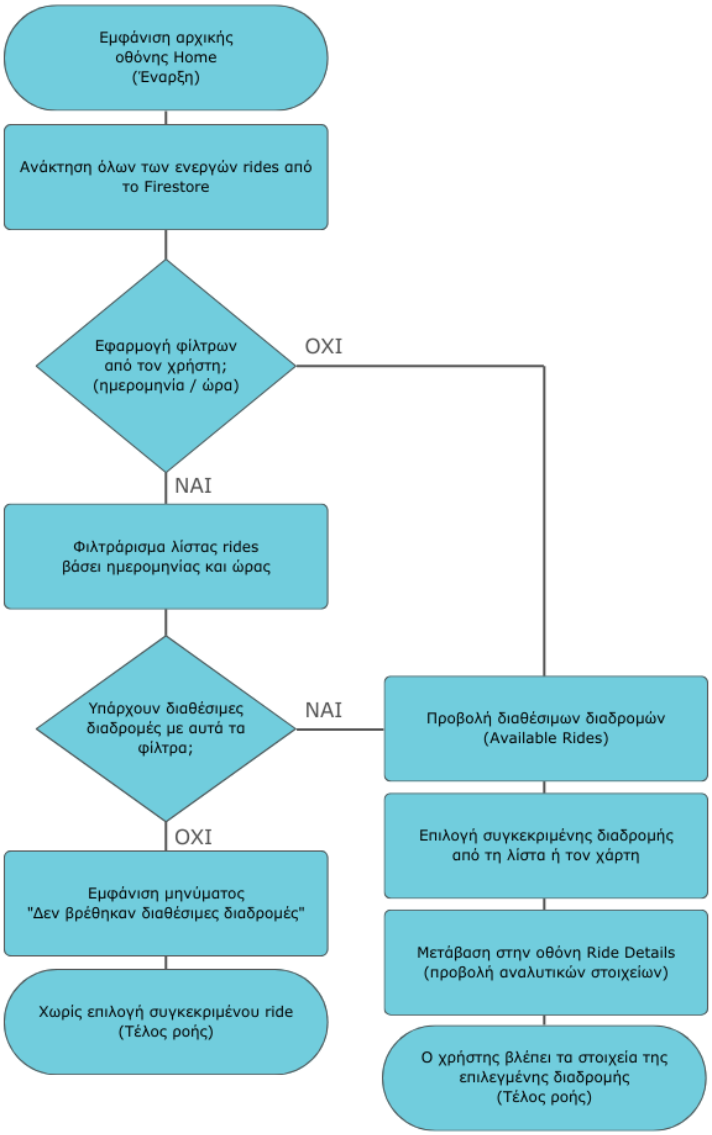


Σχήμα 2.3: FD2 - Διάγραμμα Ροής για τη Δημιουργία Προφίλ Χρήστη (UC2).

2.4.5.3 FD3 - Διάγραμμα Ροής για την Αναζήτηση Διαδρομών (UC3)

Η αναζήτηση διαθέσιμων διαδρομών αποτελεί μία από τις κεντρικές λειτουργίες της εφαρμογής, καθώς επιτρέπει στον χρήστη να εντοπίζει rides που έχουν δημιουργηθεί από άλλους οδηγούς και να αξιολογεί αν τον εξυπηρετούν χρονικά και χωρικά. Κατά την είσοδο στην αρχική οθόνη, η εφαρμογή ανακτά από το Firestore όλες τις ενεργές διαδρομές και τις παρουσιάζει στον χρήστη. Ο χάρτης δεν χρησιμοποιείται σε αυτό το στάδιο, εμφανίζεται μόνο κατά την προβολή μιας συγκεκριμένης διαδρομής. Προαιρετικά, ο χρήστης μπορεί να εφαρμόσει φίλτρα βάσει ημερομηνίας και ώρας, ώστε να περιορίσει τα αποτελέσματα σε διαδρομές που τον ενδιαφέρουν.

Το διάγραμμα ροής που ακολουθεί παρουσιάζει τα βασικά βήματα της διαδικασίας αναζήτησης, από την αρχική φόρτωση των διαδρομών μέχρι την εμφάνιση των διαθέσιμων αποτελεσμάτων και την επιλογή μιας συγκεκριμένης διαδρομής για περαιτέρω προβολή, όπου εμφανίζεται και ο χάρτης που απεικονίζει μόνο τη διαδρομή που έχει επιλέξει ο χρήστης. Παράλληλα, αποτυπώνονται οι περιπτώσεις όπου δεν βρίσκονται διαθέσιμες διαδρομές για τα κριτήρια που έχει ορίσει ο χρήστης, οπότε η εφαρμογή εμφανίζει το κατάλληλο ενημερωτικό μήνυμα (Σχήμα 2.4).

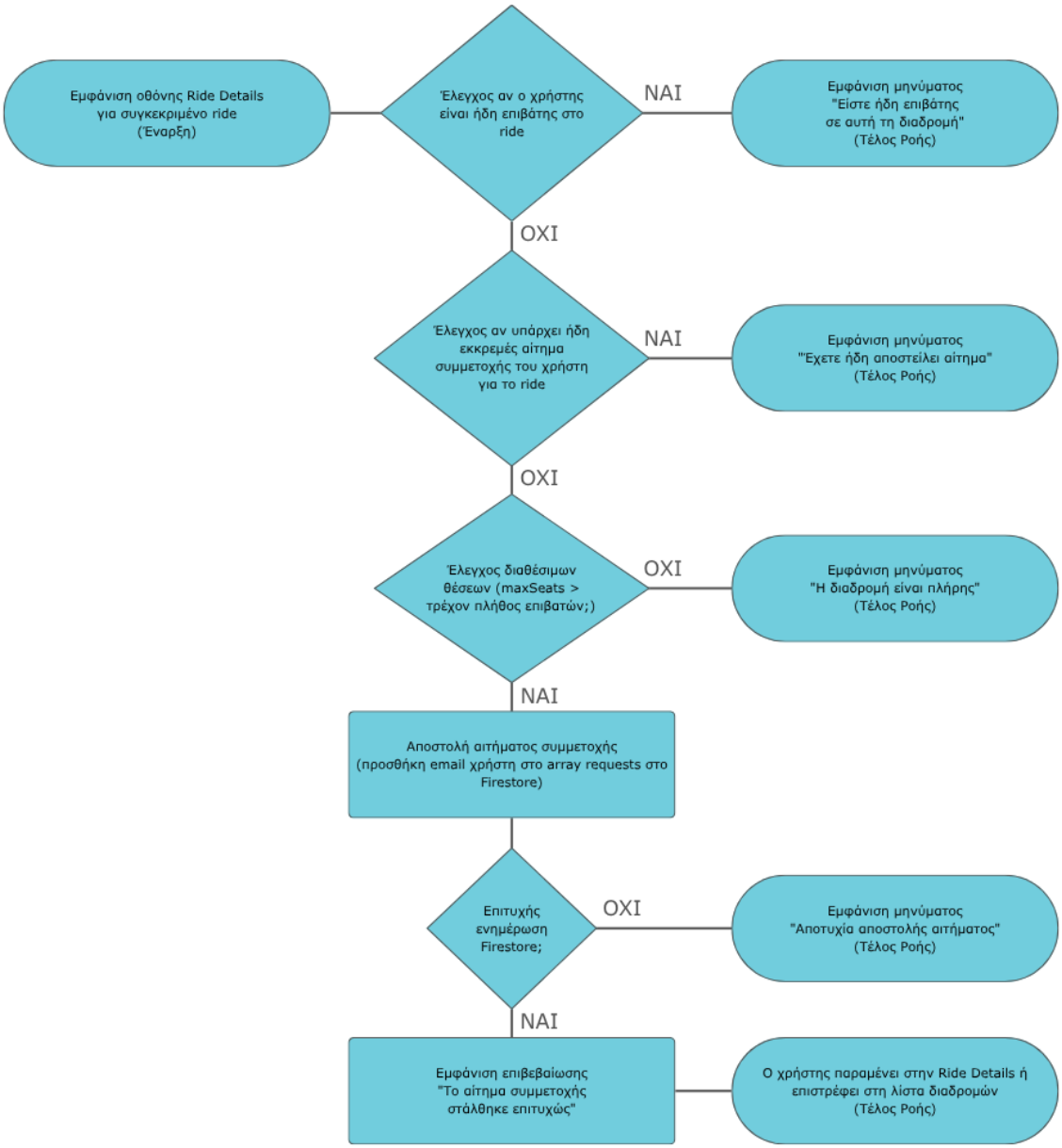


Σχήμα 2.4: FD3 - Διάγραμμα Ροής για την Αναζήτηση Διαδρομών (UC3).

2.4.5.4 FD4 - Διάγραμμα Ροής για την Αποστολή Αιτήματος Συμμετοχής (UC4)

Η λειτουργία αποστολής αιτήματος συμμετοχής σε διαδρομή αποτελεί το βασικό σημείο στο οποίο ο απλός χρήστης αλληλεπιδρά με έναν οδηγό, ζητώντας να καταλάβει μία από τις διαθέσιμες θέσεις ενός ride. Αφού προηγουμένως έχει αναζητήσει και επιλέξει μία συγκεκριμένη διαδρομή από τη λίστα ή τον χάρτη, ο χρήστης μεταφέρεται στην οθόνη αναλυτικών στοιχείων (Ride Details), όπου έχει τη δυνατότητα να υποβάλει αίτημα συμμετοχής.

Το παρακάτω διάγραμμα ροής (Σχήμα 2.5) αποτυπώνει τα διαδοχικά βήματα αυτής της διαδικασίας: τον έλεγχο διαθεσιμότητας θέσεων, την επιβεβαίωση ότι ο χρήστης δεν είναι ήδη επιβάτης ή δεν έχει εκκρεμές αίτημα για την ίδια διαδρομή, την αποστολή του αιτήματος στο Firestore και την ενημέρωση του χρήστη για την επιτυχή ή μη υποβολή. Παράλληλα, παρουσιάζονται οι εναλλακτικές εκβάσεις σε περίπτωση πλήρους διαδρομής ή σφάλματος κατά την επικοινωνία με τη βάση δεδομένων.

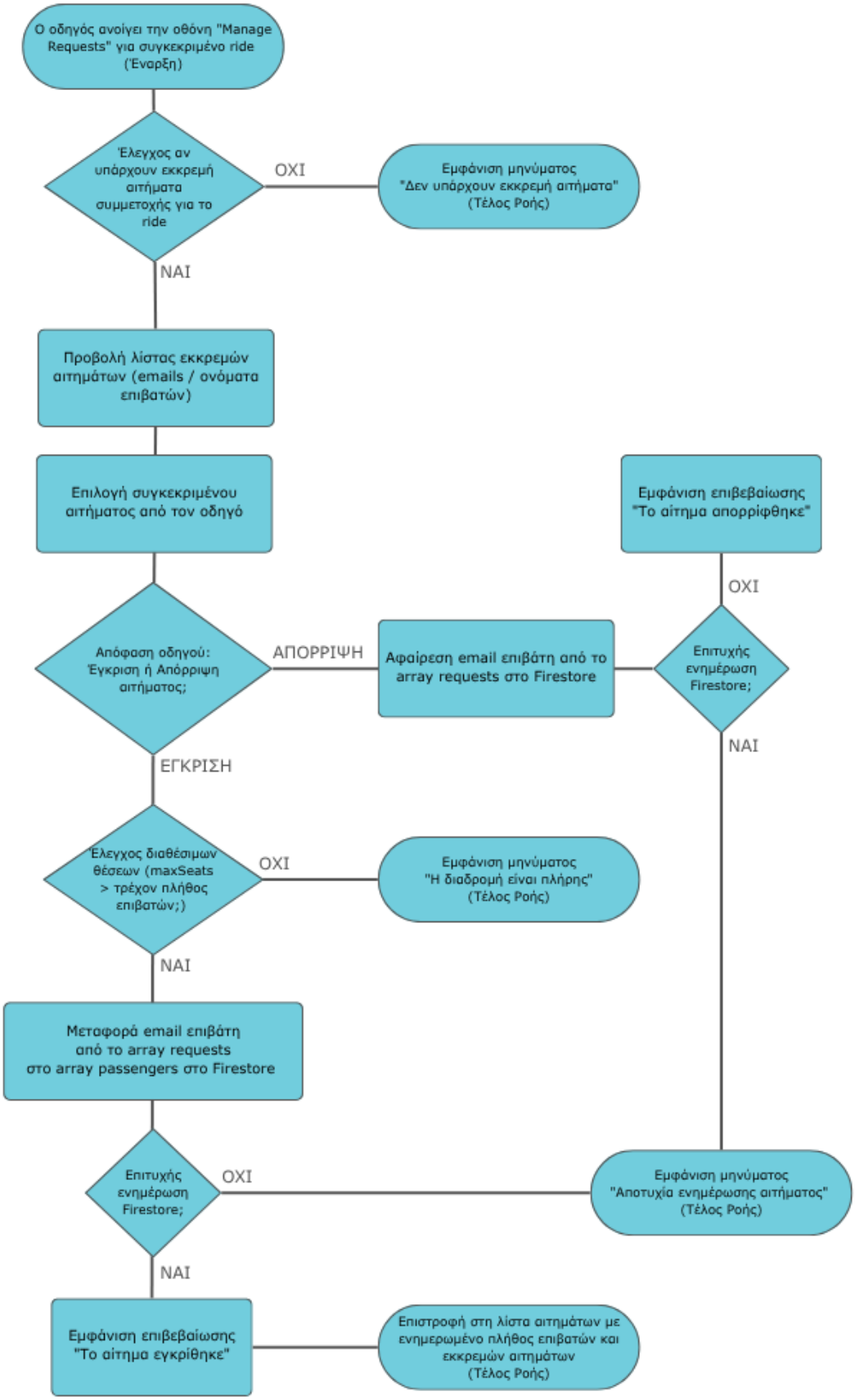


Σχήμα 2.5: FD4 - Διάγραμμα Ροής για την Αποστολή Αιτήματος Συμμετοχής (UC4).

2.4.5.5 FD5 - Διάγραμμα Ροής για τη Διαχείριση Αιτημάτων Συμμετοχής (UC5)

Η διαχείριση αιτημάτων συμμετοχής αποτελεί κεντρική λειτουργία για τον ρόλο του οδηγού, καθώς μέσω αυτής αποφασίζει ποιοι επιβάτες θα συμμετάσχουν τελικά στη διαδρομή. Από την οθόνη “Manage Requests” ο οδηγός βλέπει τα εκκρεμή αιτήματα για κάθε ride, επιλέγει ένα συγκεκριμένο αίτημα και προχωρά είτε σε έγκριση είτε σε απόρριψη.

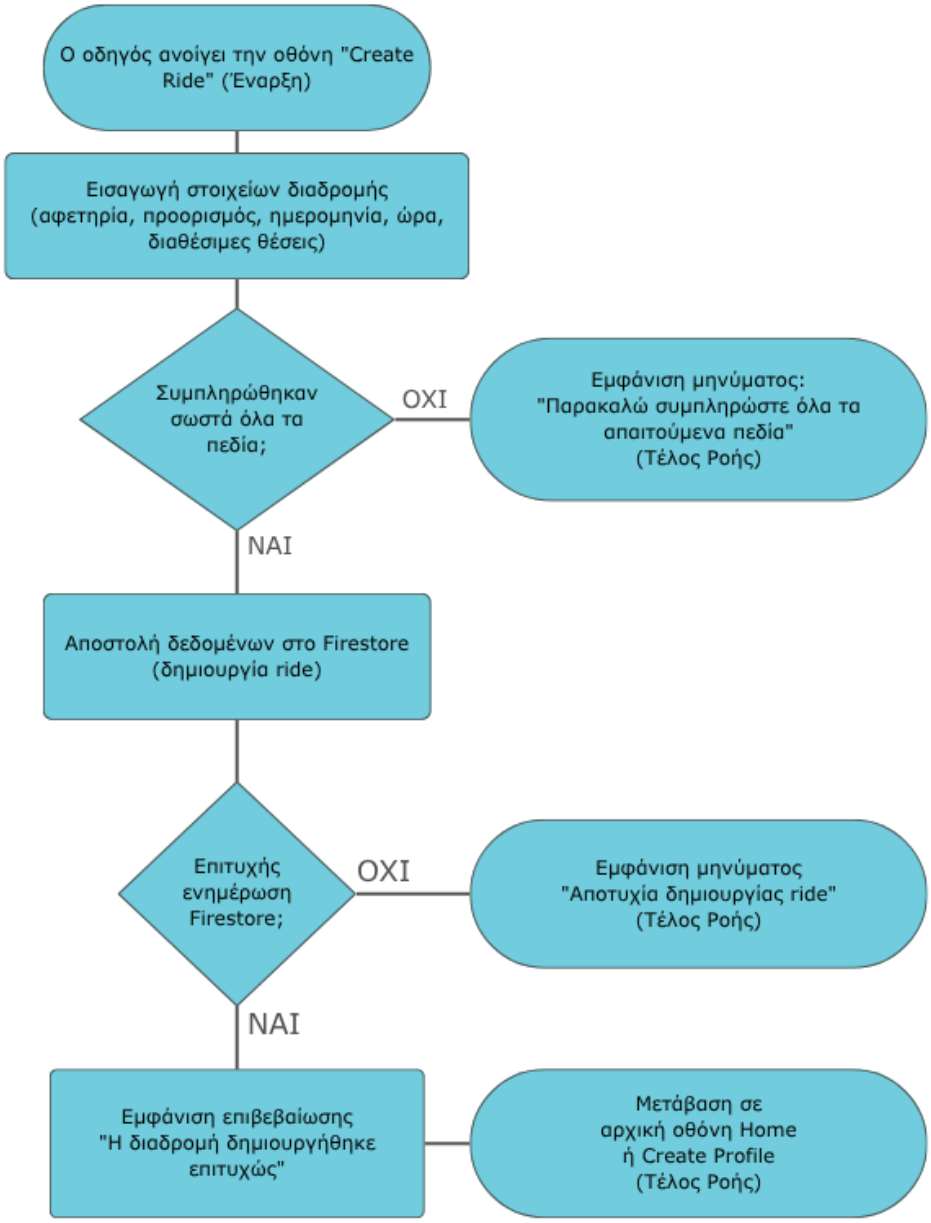
Η ροή που ακολουθεί περιλαμβάνει τον έλεγχο ύπαρξης εκκρεμών αιτημάτων, την επιλογή συγκεκριμένου αιτήματος από τον οδηγό, τη διάκριση μεταξύ έγκρισης και απόρριψης, τον έλεγχο διαθεσιμότητας θέσεων στην περίπτωση έγκρισης, καθώς και την ενημέρωση της βάσης Firestore (μετακίνηση του email από τη λίστα requests στη λίστα passengers ή απλή αφαίρεσή του στην περίπτωση απόρριψης). Τέλος, παρουσιάζονται οι πιθανές περιπτώσεις σφάλματος κατά την ενημέρωση των δεδομένων και η αντίστοιχη ενημέρωση του οδηγού (Σχήμα 2.6).



Σχήμα 2.6: FD5 - Διάγραμμα Ροής για τη Διαχείριση Αιτημάτων Συμμετοχής (UC5).

2.4.5.6 FD6 - Διάγραμμα Ροής για τη Δημιουργία Νέου Ride (UC6)

Η διαδικασία δημιουργίας μιας νέας διαδρομής (Create Ride) αποτελεί μία από τις βασικότερες λειτουργίες της εφαρμογής carpooling, καθώς παρέχει στον οδηγό τη δυνατότητα να ορίσει ένα νέο διαθέσιμο ride, καθορίζοντας όλα τα απαραίτητα χαρακτηριστικά της διαδρομής. Η ροή περιλαμβάνει τη συμπλήρωση των κρίσιμων πληροφοριών, όπως το σημείο αφετηρίας, ο προορισμός, η ημερομηνία και ώρα αναχώρησης και ο μέγιστος αριθμός διαθέσιμων θέσεων για επιβάτες. Στη συνέχεια, πραγματοποιείται έλεγχος εγκυρότητας των δεδομένων, ώστε να διασφαλιστεί ότι τα πεδία έχουν συμπληρωθεί σωστά, πριν από την αποστολή των πληροφοριών στο Firestore. Το διάγραμμα ροής που ακολουθεί απεικονίζει τα βασικά στάδια της διαδικασίας, καθώς και τις πιθανές εκβάσεις, όπως η αποτυχία καταχώρησης ή η επιτυχής δημιουργία του ride, παρέχοντας πλήρη εικόνα της λειτουργικής συμπεριφοράς του συστήματος κατά τη δημιουργία νέας διαδρομής (Σχήμα 2.7).



Σχήμα 2.7: FD6 - Διάγραμμα Ροής για τη Δημιουργία Νέου Ride (UC6).

Τα παραπάνω διαγράμματα ροής παρουσιάζουν με δομημένο και αναλυτικό τρόπο τις κύριες λειτουργίες της εφαρμογής carpooling, αποτυπώνοντας τόσο τη βασική ροή όσο και τις πιθανές εναλλακτικές εκβάσεις κάθε διαδικασίας. Μέσα από την οπτική αναπαράσταση των βημάτων αναδεικνύονται με σαφήνεια τα σημεία αλληλεπίδρασης του χρήστη με το σύστημα, οι ενσωματωμένοι έλεγχοι εγκυρότητας, οι επικοινωνίες με το Firestore σε πραγματικό χρόνο και οι μηχανισμοί διαχείρισης σφαλμάτων. Συνολικά, τα διαγράμματα ροής συμπληρώνουν τα Σενάρια Χρήσης της προηγούμενης ενότητας και παρέχουν μια ολοκληρωμένη, συνεκτική και άμεσα κατανοητή απεικόνιση των λειτουργικών μηχανισμών της εφαρμογής.

2.5 Περιορισμοί και προϋποθέσεις λειτουργίας της εφαρμογής

Η ομαλή λειτουργία της εφαρμογής carpooling καθορίζεται από ένα σύνολο τεχνικών, λειτουργικών και οργανωτικών προϋποθέσεων, οι οποίες σχετίζονται τόσο με την αρχιτεκτονική του συστήματος όσο και με τις δυνατότητες που παρέχει το οικοσύστημα του Ιδρύματος. Οι περιορισμοί αυτοί δεν λειτουργούν ως εμπόδιο, αλλά οριοθετούν το πλαίσιο εντός του οποίου η εφαρμογή μπορεί να λειτουργήσει με συνέπεια, ασφάλεια και αξιοπιστία, εξασφαλίζοντας παράλληλα μια ομοιογενή εμπειρία χρήσης.

2.5.1 Προϋποθέσεις ταυτοποίησης και χρήσης

Η πρόσβαση στην εφαρμογή πραγματοποιείται αποκλειστικά μέσω της υπηρεσίας IHU Apps, γεγονός που εισάγει συγκεκριμένες προϋποθέσεις:

- Ο χρήστης πρέπει να διαθέτει ενεργό ιδρυματικό λογαριασμό, πρόσβαση από εξωτερικούς χρήστες δεν υποστηρίζεται.
- Η ταυτοποίηση θεωρείται πλήρως έγκυρη, καθώς βασίζεται στο κεντρικό authentication σύστημα του Ιδρύματος [29].
- Τα βασικά στοιχεία χρήστη παρέχονται αυτόματα από την υπηρεσία IHU Apps, μειώνοντας την ανάγκη χειροκίνητης εισαγωγής δεδομένων.
- Υπάρχει ενιαίος ρόλος χρήστη (User), ο οποίος μπορεί να λειτουργήσει τόσο ως οδηγός όσο και ως επιβάτης· όσοι χρήστες επιθυμούν να δημιουργήσουν δική τους διαδρομή οφείλουν πρώτα να εισάγουν τα στοιχεία του οχήματός τους.

2.5.2 Τεχνικοί περιορισμοί και απαιτήσεις λειτουργίας

Η εφαρμογή βασίζεται στην υποδομή του Firebase Firestore, γεγονός το οποίο εισάγει ορισμένες τεχνικές απαιτήσεις:

- Απαιτείται σύνδεση στο διαδίκτυο για την ανάκτηση και ενημέρωση των δεδομένων (rides, requests, passengers κ.λπ.).
- Υποστηρίζεται offline caching μόνο για τον χάρτη, μέσω του Google Maps SDK.
- Δεν αποθηκεύονται δεδομένα τοποθεσίας χρηστών, καθώς η εφαρμογή δεν απαιτεί real-time GPS.
- Στο Firestore αποθηκεύονται αποκλειστικά τα απαραίτητα λειτουργικά δεδομένα κάθε ride: startLocationCoords, startLocationAddress, rideDate, riderEmail, maxSeats, passengers[], requests[], completed.

Λόγω του εκπαιδευτικού χαρακτήρα της εφαρμογής, εφαρμόστηκαν απλοποιημένοι κανόνες Firestore security rules, καθώς η ανάπτυξη δεν προορίζεται για παραγωγική λειτουργία σε πραγματικό περιβάλλον.

2.5.3 Λειτουργικοί περιορισμοί

Ο λειτουργικός σχεδιασμός της εφαρμογής ενσωματώνει συγκεκριμένες κανόνες:

- Ο οδηγός μπορεί να επεξεργαστεί μόνο ημερομηνία και ώρα μιας διαδρομής ή να την ακυρώσει. Για αλλαγές στο σημείο έναρξης απαιτείται δημιουργία νέου ride.
- Η αναζήτηση διαδρομών από επιβάτες υποστηρίζει φίλτρα βάσει ημερομηνίας και ώρας.
- Η συμμετοχή επιβατών γίνεται με σύστημα αιτημάτων (request → approval), ώστε ο οδηγός να έχει πλήρη έλεγχο των συμμετεχόντων στο ride.
- Ο οδηγός καθορίζει το μέγιστο πλήθος επιβατών (maxSeats), το οποίο αποτελεί τον μοναδικό περιορισμό σχετικά με τη χωρητικότητα της διαδρομής.
- Κάθε χρήστης έχει πρόσβαση μόνο στη δική του σελίδα προφίλ.
- Η εφαρμογή δεν υποστηρίζει push notifications ή email alerts, η ενημέρωση (έγκριση/απόρριψη αιτήματος) γίνεται αποκλειστικά μέσω της εφαρμογής.

2.5.4 Απόρρητο και προστασία δεδομένων

Η εφαρμογή συμμορφώνεται προς τις βασικές αρχές του GDPR και τις οδηγίες ασφαλούς σχεδιασμού mobile εφαρμογών [27], [30]:

- Δεν αποθηκεύονται δεδομένα τοποθεσίας ή ευαίσθητα προσωπικά στοιχεία χρηστών.
- Τα μοναδικά προσωπικά δεδομένα που διατηρούνται είναι όσα παρέχονται από το IHU Apps (email, ονοματεπώνυμο) και τα στοιχεία του οχήματος όπου απαιτούνται.
- Δεν γίνεται χρήση τοπικής αποθήκευσης προσωπικών δεδομένων (local storage), πέρα από την αυτόματη προσωρινή αποθήκευση (cache) χαρτογραφικών δεδομένων που πραγματοποιείται από το Google Maps SDK.
- Λόγω του μη παραγωγικού χαρακτήρα της εφαρμογής, δεν απαιτείται πλήρης εφαρμογή πολιτικών ασφαλείας σε επίπεδο υποδομής.

Με τον τρόπο αυτό, η εφαρμογή διασφαλίζει την προστασία των προσωπικών δεδομένων και την ελάχιστη συλλογή πληροφοριών, σύμφωνα με τις σύγχρονες πρακτικές ασφαλείας.

2.6 Επίλογος

Το κεφάλαιο αυτό παρουσίασε τις βασικές αρχές που διέπουν τη λειτουργία της εφαρμογής carpooling, προσδιορίζοντας τόσο τις τεχνικές προϋποθέσεις όσο και τους πρακτικούς περιορισμούς του συστήματος. Αναλύθηκαν οι μηχανισμοί ταυτοποίησης μέσω IHU Apps, οι απαιτήσεις της αρχιτεκτονικής Firestore, οι λειτουργικοί κανόνες των διαδρομών και οι πρακτικές προστασίας δεδομένων. Συνολικά, οι περιορισμοί και προϋποθέσεις που παρουσιάστηκαν διαμορφώνουν το πλαίσιο εντός του οποίου η εφαρμογή λειτουργεί με ασφάλεια, συνέπεια και ευχρηστία, αποτελώντας τη βάση για τον αναλυτικό τεχνικό σχεδιασμό που παρουσιάζεται στο επόμενο κεφάλαιο.

Κεφάλαιο 3ο: Σχεδίαση και Αρχιτεκτονική του Συστήματος

Η σχεδίαση και αρχιτεκτονική ενός πληροφοριακού συστήματος αποτελούν κρίσιμα στάδια μεταξύ της ανάλυσης απαιτήσεων και της υλοποίησης. Μετά τον καθορισμό των λειτουργικών και μη λειτουργικών απαιτήσεων στο Κεφάλαιο 2, το παρόν κεφάλαιο επικεντρώνεται στη δομική και τεχνική περιγραφή της εφαρμογής carpooling, αναλύοντας τις αρχές σχεδίασης, τις τεχνολογίες που χρησιμοποιήθηκαν, καθώς και τη συνολική αρχιτεκτονική του συστήματος.

Συγκεκριμένα, παρουσιάζεται η αρχιτεκτονική client-server που υιοθετήθηκε, η λειτουργική διάρθρωση των επιμέρους υποσυστημάτων και η επιλογή των τεχνολογιών ανάπτυξης, όπως το Flutter για το frontend και η πλατφόρμα Firebase για την αποθήκευση δεδομένων και την ταυτοποίηση χρηστών. Παράλληλα, περιγράφεται η δομή των αρχείων και φακέλων του έργου, η σχεδίαση της διεπαφής χρήστη βάσει αρχών UI/UX και το μοντέλο δεδομένων της βάσης Firestore, στο οποίο βασίζεται η επιχειρησιακή λογική της εφαρμογής.

Τέλος, εξετάζονται ζητήματα ασφάλειας και προστασίας προσωπικών δεδομένων, σύμφωνα με τις απαιτήσεις του GDPR και τις ιδιαιτερότητες της χρήσης ιδρυματικής αυθεντικοποίησης (IHU Apps). Το κεφάλαιο αυτό δημιουργεί το αναγκαίο υπόβαθρο για την κατανόηση της υλοποίησης που παρουσιάζεται στο Κεφάλαιο 4.

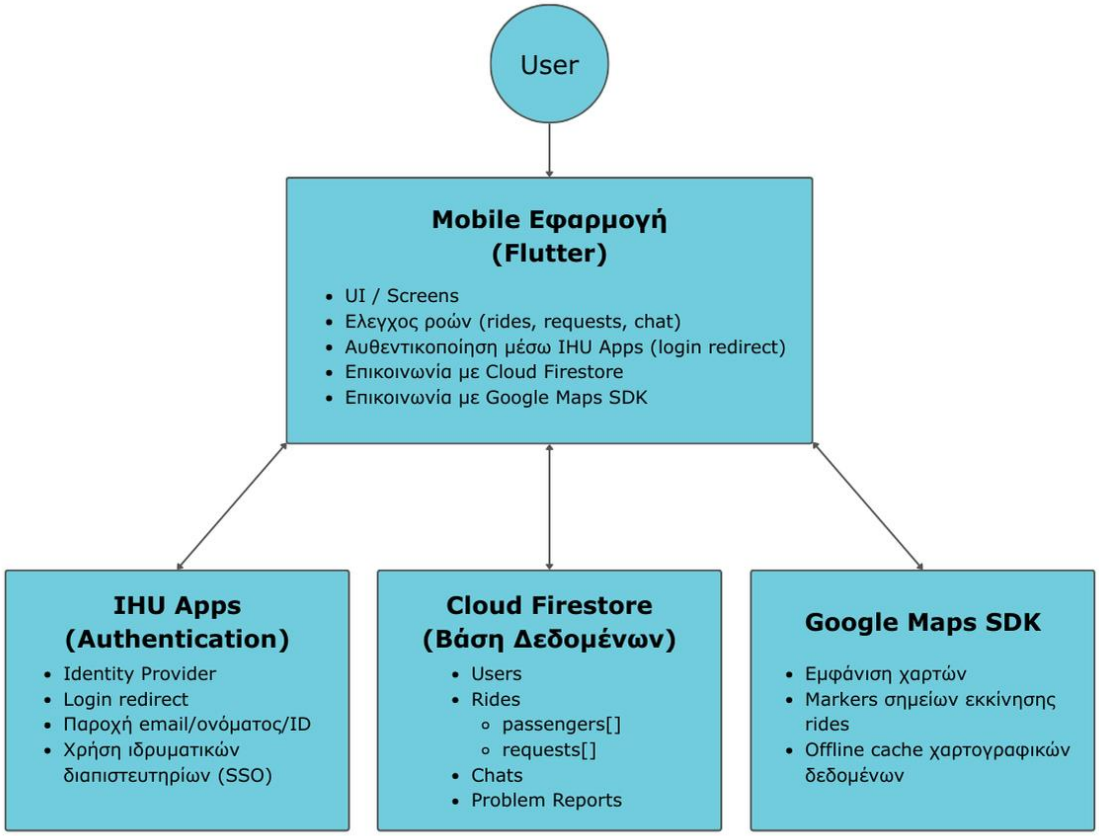
3.1 Αρχιτεκτονική και λειτουργική δομή της εφαρμογής

Η εφαρμογή carpooling που αναπτύχθηκε στο πλαίσιο της παρούσας Δ.Ε.βασίζεται σε μια σύγχρονη, πλήρως mobile-προσανατολισμένη αρχιτεκτονική, η οποία αξιοποιεί τις δυνατότητες του οικοσυστήματος Flutter και των υπηρεσιών Firebase. Η αρχιτεκτονική του συστήματος οργανώνεται γύρω από την έννοια της απλότητας, της επεκτασιμότητας και της ασφαλούς διαχείρισης δεδομένων, ενώ υποστηρίζει όλες τις λειτουργίες που περιγράφονται στο Κεφάλαιο 2, όπως η είσοδος μέσω IHU Apps, η δημιουργία και αναζήτηση διαδρομών, η αποστολή αιτημάτων συμμετοχής, η επικοινωνία χρηστών μέσω chat και η επεξεργασία rides από τον οδηγό.

Σε υψηλό επίπεδο, η εφαρμογή ακολουθεί τη λογική ενός client-server μοντέλου, όπου:

- Η εφαρμογή Flutter λειτουργεί ως client, αναλαμβάνοντας την παρουσίαση, την αλληλεπίδραση με τον χρήστη και την επικοινωνία με το backend.
- Οι υπηρεσίες της πλατφόρμας Firebase λειτουργούν ως backend, παρέχοντας μηχανισμούς αυθεντικοποίησης, αποθήκευσης δεδομένων και λήψης ενημερώσεων σε πραγματικό χρόνο.
- Το IHU Apps λειτουργεί ως εξωτερικός πάροχος ταυτοποίησης, μέσω OAuth-based redirect, επιτρέποντας την ασφαλή είσοδο αποκλειστικά χρηστών που διαθέτουν ιδρυματικό λογαριασμό.

Η συνολική αρχιτεκτονική μπορεί να αποτυπωθεί ως ένα σύνολο συνεργαζόμενων υπηρεσιών που συνθέτουν ένα ολοκληρωμένο πληροφοριακό σύστημα, όπου το client μέρος διαχειρίζεται την επιχειρησιακή λογική της διεπαφής, ενώ το backend αναλαμβάνει τη μόνιμη αποθήκευση και τον συγχρονισμό των δεδομένων.



Σχήμα 3.1: Υψηλού επιπέδου αρχιτεκτονικό διάγραμμα της εφαρμογής carpooling.

3.1.1 Επίπεδα του συστήματος

Η αρχιτεκτονική χωρίζεται ενδεικτικά σε τρία λειτουργικά επίπεδα:

3.1.1.1 Επίπεδο Παρουσίασης (Client-Frontend: Flutter)

Αποτελεί το ορατό μέρος της εφαρμογής και περιλαμβάνει όλες τις οθόνες, την πλοήγηση (navigation), την αλληλεπίδραση με στοιχεία UI/UX και τη διαχείριση της τοπικής κατάστασης (state). Το frontend:

- Προβάλλει χάρτες μέσω του Google Maps SDK.
- Ανταποκρίνεται σε ενέργειες χρήστη (αναζήτηση, δημιουργία ride, αποστολή αιτήματος κ.λπ.).
- Επικοινωνεί με το Firestore μέσω API calls.
- Εμφανίζει δεδομένα σε πραγματικό χρόνο χάρη στους μηχανισμούς real-time updates του Firestore.

3.1.1.2 Επίπεδο Υπηρεσιών και Αυθεντικοποίησης (Authentication Layer)

Η είσοδος πραγματοποιείται με redirect προς την υπηρεσία IHU Apps, η οποία παρέχει:

- επαλήθευση ιδρυματικών credentials
- επιστροφή βασικών στοιχείων χρήστη (email, όνομα, μοναδικό userID)
- ασφαλή διάκριση ταυτότητας χωρίς αποθήκευση κωδικών στην εφαρμογή

Η εφαρμογή χρησιμοποιεί το επιστρεφόμενο email ως μοναδικό αναγνωριστικό στον Firestore, αποφεύγοντας την ανάγκη διαχείρισης τοπικών credentials.

3.1.1.3 Επίπεδο Δεδομένων (Backend: Firestore)

Το backend της εφαρμογής βασίζεται στο Cloud Firestore, όπου αποθηκεύονται όλα τα απαραίτητα λειτουργικά δεδομένα, όπως:

- προφίλ χρηστών
- στοιχεία οχήματος
- rides (startLocationCoords, startLocationAddress, rideDate, riderEmail, maxSeats, passengers, requests, completed)
- αναφορές προβλημάτων
- μηνύματα chat

Οι ενημερώσεις των δεδομένων συγχρονίζονται άμεσα με όλους τους χρήστες μέσω real-time listeners, χωρίς την ανάγκη παραδοσιακού server.

3.1.2 Λειτουργική Ροή και Αλληλεπίδραση Υποσυστημάτων

Η λειτουργική δομή της εφαρμογής ενοποιεί όλα τα παραπάνω επίπεδα σε μια συνεκτική ροή, η οποία ακολουθεί τα σενάρια χρήσης που αναλύθηκαν στο Κεφάλαιο 2. Ενδεικτικά:

1. Ο χρήστης εισέρχεται μέσω redirect στο IHU Apps → λαμβάνει ταυτότητα → δημιουργείται ή ενημερώνεται το Firestore profile.
2. Ο χρήστης βλέπει την αρχική οθόνη και ανακτά rides από το Firestore.
3. Ο επιβάτης μπορεί να στείλει αίτημα συμμετοχής → το αίτημα εμφανίζεται στον οδηγό.
4. Ο οδηγός εγκρίνει/απορρίπτει → το Firestore ενημερώνεται → ο επιβάτης ενημερώνεται από το UI.
5. Για αποστολές μηνυμάτων chat, το Firestore λειτουργεί ως real-time message store.

Η απουσία παραδοσιακού server μειώνει την πολυπλοκότητα και ενισχύει την αξιοπιστία, δεδομένου ότι σχεδόν όλη η επιχειρησιακή λογική εκτελείται στον client ή στο Firestore ως atomic operations.

3.1.3 Πλεονεκτήματα της επιλεγμένης αρχιτεκτονικής

Η αρχιτεκτονική αυτή προσφέρει:

- Ευκολία συντήρησης και επεκτασιμότητας, λόγω διαχωρισμού client-backend.
- Ασφάλεια, καθώς η αυθεντικοποίηση βασίζεται σε ιδρυματικό πάροχο και δεν αποθηκεύονται ευαίσθητα credentials.
- Απλότητα στην υλοποίηση, λόγω απουσίας custom server-side infrastructure.
- Real-time ενημερώσεις, που καλύπτουν πλήρως ανάγκες όπως requests, approvals και chat.
- Συμβατότητα με mobile πλατφόρμες, χάρη στη χρήση Flutter.

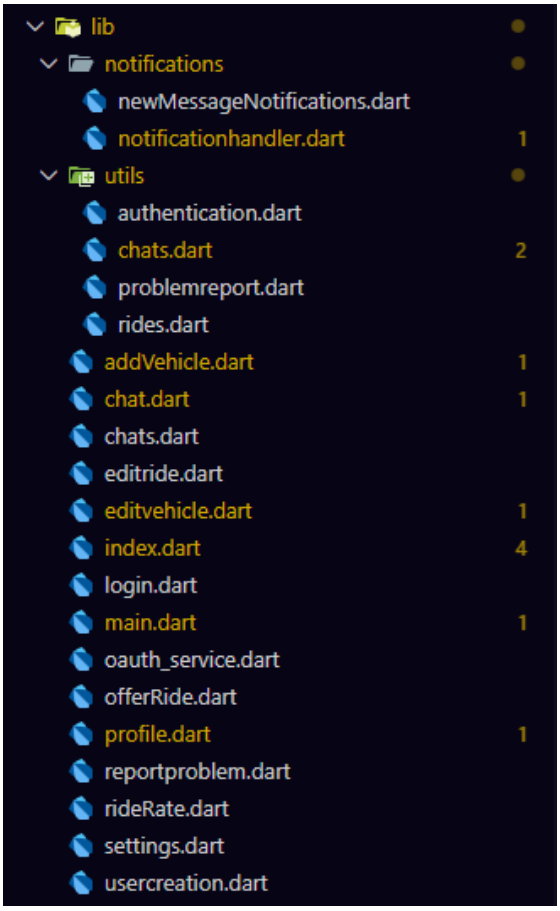
3.2 Δομή αρχείων και φακέλων του έργου

Η δομή των αρχείων της εφαρμογής carpooling έχει σχεδιαστεί με στόχο την καθαρή οργάνωση του κώδικα, την ευκολία συντήρησης και την απομόνωση των επιμέρους λειτουργικών ενοτήτων. Το έργο ακολουθεί τις βέλτιστες πρακτικές αρχιτεκτονικής για Flutter εφαρμογές, διαχωρίζοντας την επιχειρησιακή λογική, τα γραφικά περιβάλλοντα (UI), τις υπηρεσίες επικοινωνίας με το Firebase και τα δεδομένα.

Η οργάνωση των φακέλων εξασφαλίζει επεκτασιμότητα, ευκολία εντοπισμού των βασικών modules (authentication, rides, chat, profile) και καθιστά δυνατή τη μελλοντική προσθήκη νέων λειτουργιών χωρίς να απαιτούνται εκτεταμένες αλλαγές στο ήδη υπάρχον codebase.

3.2.1 Αναλυτική δομή φακέλων

Η συνολική δομή του έργου αποτυπώνεται συνοπτικά παρακάτω:



Σχήμα 3.2: Δομή του φακέλου lib/ της εφαρμογής.

3.2.2 Περιγραφή βασικών αρχείων και φακέλων του lib/

Παρακάτω παρουσιάζεται η λειτουργική ανάλυση των σημαντικότερων αρχείων και φακέλων του καταλόγου lib/:

1. main.dart
   Αρχείο εισόδου της εφαρμογής.
   - Αρχικοποιεί το Firebase.
   - Ορίζει το MaterialApp και το βασικό routing μεταξύ των οθονών.
   - Ελέγχει αν ο χρήστης είναι συνδεδεμένος, ώστε να μεταφερθεί στην κατάλληλη αρχική οθόνη.

2. login.dart
   Οθόνη εισόδου μέσω της υπηρεσίας IHU Apps.
   - Εμφανίζει το κουμπί “Log in with IHU Apps”.
   - Καλεί τις ροές αυθεντικοποίησης του oauth_service.dart

3. usercreation.dart
   Οθόνη αρχικής δημιουργίας προφίλ.
   - Προσυμπληρώνει το όνομα και το email που έρχονται από το IHU Apps.
   - Επιτρέπει στον χρήστη να δηλώσει αν διαθέτει όχημα και, εφόσον ναι, να εισάγει μοντέλο και πινακίδα.

4. addVehicle.dart / editvehicle.dart
   Φόρμες προσθήκης και επεξεργασίας οχήματος.
   - Καταχώρηση ή ενημέρωση στοιχείων οχήματος στο Firestore.

5. offerRide.dart / editride.dart
   Αρχεία που υλοποιούν τη δημιουργία και την επεξεργασία διαδρομών.
   - Ορισμός σημείου εκκίνησης, ημερομηνίας/ώρας και διαθέσιμων θέσεων.
   - Δυνατότητα ενημέρωσης ημερομηνίας και ώρας ή ακύρωσης μιας διαδρομής.

6. rideRate.dart
   Διαχείριση αξιολόγησης διαδρομών από τους χρήστες.
   - Εμφανίζει φόρμα αξιολόγησης (κριτική).
   - Αποθηκεύει τα δεδομένα στο Firestore.

7. index.dart
   Κεντρική οθόνη της εφαρμογής (Home).
   - Εμφανίζει τον χάρτη με τα σημεία εκκίνησης των rides.
   - Προβάλλει τη λίστα “Available Rides” και συνδέει τον χρήστη με την οθόνη λεπτομερειών.

8. chat.dart / chats.dart
   - chats.dart: λίστα συνομιλιών στις οποίες συμμετέχει ο χρήστης.
   - chat.dart: οθόνη ανταλλαγής μηνυμάτων σε πραγματικό χρόνο για ένα συγκεκριμένο ride

9. profile.dart
   Οθόνη προφίλ χρήστη.
   - Εμφανίζει στατιστικά (rides taken, rides completed).
   - Προβάλλει και επιτρέπει την επεξεργασία στοιχείων οχήματος.
   - Παρουσιάζει τις διαδρομές που έχει δημιουργήσει ο χρήστης.

10. reportproblem.dart / settings.dart
    - reportproblem.dart: φόρμα αναφοράς προβλήματος που αποθηκεύει τα reports στο Firestore.
    - settings.dart: βασικές ρυθμίσεις λογαριασμού, όπως διαγραφή account.

11. notifications/
    Φάκελος με κώδικα που σχετίζεται με τις ειδοποιήσεις της εφαρμογής.
    - π.χ. newMessageNotifications.dart, notificationhandler.dart για δημιουργία και χειρισμό ειδοποιήσεων νέων μηνυμάτων.

12. oauth_service.dart
    Υλοποιεί τη ροή αυθεντικοποίησης μέσω IHU Apps.

- Διαχειρίζεται τα OAuth tokens και την ασφαλή επιστροφή του χρήστη στην εφαρμογή.

13. utils/
    Φάκελος με βοηθητικά αρχεία (utility / service-like κλάσεις):
    - authentication.dart → έλεγχος/διαχείριση του session του χρήστη μέσω Firebase Auth.
    - rides.dart → επαναχρησιμοποιήσιμες συναρτήσεις για CRUD πράξεις πάνω στις διαδρομές.
    - chats.dart → βοηθητικές συναρτήσεις για ανάγνωση/εγγραφή μηνυμάτων.
    - problemreport.dart → βοηθητικές συναρτήσεις για τις αναφορές προβλημάτων.

3.2.3 Πλεονεκτήματα της δομής

Η παραπάνω δομή προσφέρει τα ακόλουθα πλεονεκτήματα:

- Καθαρός διαχωρισμός λογικής και UI, διευκολύνοντας τη συντήρηση και την κατανόηση του κώδικα.
- Ευκολία επεκτασιμότητας, καθώς νέα modules μπορούν να προστεθούν χωρίς ανάγκη αναδόμησης των υπαρχόντων.
- Διαφάνεια στην οργάνωση του έργου, με ξεκάθαρους φακέλους ανά λειτουργική περιοχή (UI, services, models, utilities).
- Αποδοτική συνεργασία σε περιβάλλον ανάπτυξης, επειδή ο κώδικας ακολουθεί προβλέψιμη και συνεκτική αρχιτεκτονική.
- Μείωση πιθανότητας σφαλμάτων μέσω χρήσης τυποποιημένων models και centralized services για επικοινωνία με Firebase.

3.3 Επιλογή Τεχνολογιών και Εργαλείων Ανάπτυξης

Η επιλογή των τεχνολογιών και των εργαλείων ανάπτυξης αποτελεί κρίσιμο παράγοντα για την επιτυχία μιας mobile εφαρμογής, καθώς επηρεάζει άμεσα την απόδοση, την επεκτασιμότητα, την ευκολία συντήρησης και το συνολικό κόστος υλοποίησης. Στην παρούσα εφαρμογή carpooling, η οποία απευθύνεται στην ακαδημαϊκή κοινότητα του ΔΙΠΑΕ και βασίζεται σε πραγματικού χρόνου ενημερώσεις και ασφαλή ταυτοποίηση χρηστών, επιλέχθηκαν τεχνολογίες που προσφέρουν υψηλή αξιοπιστία, άμεση ενσωμάτωση με mobile περιβάλλοντα και άριστη υποστήριξη από την κοινότητα ανάπτυξης.

Στις υποενότητες που ακολουθούν παρουσιάζονται αναλυτικά τα κύρια εργαλεία και πλαίσια που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής, καθώς και ο ρόλος που διαδραματίζει το καθένα στην τεχνική υποδομή του έργου.

3.3.1 Flutter

Το Flutter αποτελεί το κύριο framework που χρησιμοποιήθηκε για την ανάπτυξη της mobile εφαρμογής carpooling. Πρόκειται για ένα σύγχρονο, open-source UI toolkit που αναπτύχθηκε από τη Google και επιτρέπει τη δημιουργία εφαρμογών υψηλής απόδοσης για Android, iOS, web και desktop μέσω ενός ενιαίου κώδικα. Η επιλογή του Flutter βασίστηκε σε τρεις βασικούς άξονες: την ταχύτητα ανάπτυξης, την υψηλή απόδοση στο τελικό περιβάλλον και την ευκολία συντήρησης του έργου.

Ένα από τα σημαντικότερα πλεονεκτήματα του Flutter είναι η λειτουργία Hot Reload, η οποία επιτρέπει στον προγραμματιστή να βλέπει άμεσα τις αλλαγές στην εφαρμογή χωρίς να απαιτείται πλήρης ανακατασκευή. Το χαρακτηριστικό αυτό επιταχύνει σημαντικά τον κύκλο ανάπτυξης, ιδιαίτερα σε έργα που περιλαμβάνουν πειραματισμό στη διεπαφή χρήστη ή συχνές τροποποιήσεις στη λογική της εφαρμογής.

Επιπλέον, το Flutter χρησιμοποιεί τη μηχανή γραφικών Skia, προσφέροντας εγγενή απόδοση και ομαλή απεικόνιση UI στοιχείων ανεξάρτητα από τη συσκευή. Αυτό εξασφαλίζει σταθερή εμπειρία χρήσης σε όλα τα Android devices που στοχεύει η εφαρμογή. Παράλληλα, η αρχιτεκτονική του Flutter βασίζεται σε widgets, τα οποία ενισχύουν τη modular ανάπτυξη και τη δυνατότητα επαναχρησιμοποίησης κώδικα.

Για την παρούσα εφαρμογή, το Flutter αποτέλεσε την ιδανική επιλογή, καθώς παρέχει άμεση ενσωμάτωση με Firebase SDKs, Google Maps SDK και υπηρεσίες τρίτων. Η δυνατότητα σύνδεσης με real-time βάσεις δεδομένων, καθώς και η ευελιξία του στη διαχείριση state και navigation, υποστήριξαν πλήρως τις απαιτήσεις της εφαρμογής όσον αφορά την αλληλεπίδραση με τον χάρτη, την εμφάνιση δυναμικών λιστών rides και την επικοινωνία με το backend σε πραγματικό χρόνο.

3.3.2 Dart

Η Dart αποτελεί τη γλώσσα προγραμματισμού στην οποία βασίζεται ολόκληρη η εφαρμογή carpooling, καθώς είναι η κύρια γλώσσα που υποστηρίζει το Flutter Framework. Η Dart σχεδιάστηκε από τη Google με στόχο την ανάπτυξη αποδοτικών, ασφαλών και επεκτάσιμων εφαρμογών, τόσο για mobile όσο και για web και desktop περιβάλλοντα.

Πρόκειται για μία strongly typed, object-oriented γλώσσα που υποστηρίζει μοντέρνες προγραμματιστικές πρακτικές, όπως classes, asynchronous programming και robust error handling. Η Dart διαθέτει ενσωματωμένη υποστήριξη για async/await, γεγονός που την καθιστά ιδιαίτερα κατάλληλη για εφαρμογές που αλληλεπιδρούν με απομακρυσμένες υπηρεσίες, όπως το Firebase Firestore και το Firebase Authentication, όπου οι περισσότερες λειτουργίες εκτελούνται ασύγχρονα.

Ένα επιπλέον πλεονέκτημα της Dart είναι ο AOT (Ahead-of-Time) και JIT (Just-in-Time) compiler της. Σε mobile περιβάλλοντα χρησιμοποιείται AOT compilation, το οποίο μεταφράζει τον κώδικα σε εγγενή μορφή και εξασφαλίζει υψηλή απόδοση στην εκτέλεση. Κατά τη διάρκεια της ανάπτυξης, η Dart χρησιμοποιεί JIT compilation, το οποίο επιτρέπει το Hot Reload του Flutter, επιταχύνοντας τον συντονισμό μεταξύ κώδικα και UI.

Για την παρούσα εφαρμογή, η Dart προσφέρει όλα τα απαραίτητα χαρακτηριστικά ώστε να υλοποιηθούν:

- Η επικοινωνία με Firebase Firestore με ασφαλή και ασύγχρονο τρόπο.
- Η μοντελοποίηση δεδομένων μέσω Dart classes (π.χ. RideModel, UserModel).
- Η διαχείριση της επιχειρησιακής λογικής (business logic) της εφαρμογής.
- Η δημιουργία responsive UI components που βασίζονται σε widget trees.

Η σταθερότητα, η απόδοση και η άριστη συνεργασία της Dart με το Flutter αποτέλεσαν καθοριστικούς παράγοντες για την επιλογή της ως βασικής γλώσσας ανάπτυξης της εφαρμογής.

3.3.3 Firebase

Η πλατφόρμα Firebase αξιοποιείται ως βασικό backend της εφαρμογής, παρέχοντας υπηρεσίες αποθήκευσης δεδομένων και real-time ενημερώσεων χωρίς ανάγκη ανάπτυξης ξεχωριστού server. Η

επιλογή της έγινε λόγω της άμεσης ενσωμάτωσης με το Flutter SDK, της υψηλής αξιοπιστίας, της εύκολης κλιμάκωσης και της απουσίας κόστους για μικρές εφαρμογές εκπαιδευτικού χαρακτήρα.

Στο πλαίσιο της εφαρμογής carpooling, χρησιμοποιούνται δύο βασικά υποσυστήματα του Firebase: το Cloud Firestore και το Firebase SDK για την αλληλεπίδραση με την πλατφόρμα.

3.3.3.1 Cloud Firestore

Το Cloud Firestore αποτελεί τη βασική βάση δεδομένων της εφαρμογής. Όλα τα λειτουργικά δεδομένα αποθηκεύονται σε collections και documents, τα οποία διαβάζονται και ενημερώνονται σε πραγματικό χρόνο. Η εφαρμογή χρησιμοποιεί τις εξής βασικές συλλογές:

- Users
  Περιέχει τα προφίλ των χρηστών που δημιουργούνται μετά την είσοδο μέσω IHU Apps.
  Αποθηκεύονται:
  - email
  - πλήρες όνομα
  - στοιχεία οχήματος (μοντέλο, πινακίδα)
  - στατιστικά (π.χ. rides taken, rides completed)

- Rides
  Περιλαμβάνει κάθε διαθέσιμη διαδρομή.
  Πεδία:
  - startLocationCoords
  - startLocationAddress
  - rideDate (ημερομηνία/ώρα)
  - riderEmail
  - maxSeats
  - passengers[] (array email χρηστών)
  - requests[] (αίτηματα συμμετοχής)
  - completed (boolean)

- Chats
  Collection που περιέχει συνομιλίες ανά διαδρομή ή ανά ζεύγος χρηστών.
  Η εφαρμογή χρησιμοποιεί real-time ενημερώσεις για την άμεση λήψη νέων μηνυμάτων.

- ProblemReports
  Καταγράφει τα προβλήματα που αναφέρουν οι χρήστες μέσω της οθόνης Report Problem.

Λόγοι επιλογής Firestore:

- Real-time ενημερώσεις χωρίς χειροκίνητο polling
- Document-based μοντέλο, ιδανικό για mobile εφαρμογές
- Πλήρης ενσωμάτωση με Flutter
- Εύκολος χειρισμός arrays (π.χ. passengers[], requests[])
- Αυτόματη κλιμάκωση και υψηλή διαθεσιμότητα

Η Firestore χρησιμοποιείται αναγνωστικά και εγγραφικά από κεντρικές service κλάσεις (π.χ. rides.dart, authentication.dart, problemreport.dart), εξασφαλίζοντας καθαρή επιχειρησιακή λογική.

3.3.3.2 Firebase SDK

Το Firebase SDK αποτελεί τη βιβλιοθήκη μέσω της οποίας η εφαρμογή Flutter επικοινωνεί με τις υπηρεσίες του Firebase. Παρέχει έτοιμες εντολές και APIs που επιτρέπουν την ασφαλή και αποδοτική πρόσβαση στα δεδομένα της εφαρμογής, χωρίς την ανάγκη ανάπτυξης ξεχωριστού backend server.

Οι βασικές λειτουργίες που υποστηρίζονται μέσω του Firebase SDK είναι οι εξής:

- Διαχείριση κατάστασης χρήστη
    - ανάκτηση στοιχείων χρήστη που προέρχονται από το IHU Apps
    - έλεγχος πρόσβασης σε δεδομένα μέσω των Firestore Security Rules, με βάση το αναγνωρισμένο προφίλ χρήστη.

- Firestore API
    - εκτέλεση πράξεων CRUD για τις συλλογές users, rides, requests, messages και reports,
    - υποστήριξη real-time ενημερώσεων μέσω listeners (snapshots),
    - atomic ενημερώσεις πεδίων και πινάκων (π.χ. FieldValue.arrayUnion()).

- Offline υποστήριξη και συγχρονισμός
  Παρότι η εφαρμογή απαιτεί ενεργή σύνδεση για τη βασική της λειτουργία, το Firestore SDK παρέχει προσωρινό τοπικό caching δεδομένων και αυτόματο συγχρονισμό με τη βάση όταν αποκατασταθεί η σύνδεση.

Το Firebase SDK υποστηρίζεται επίσημα από την Google και προσφέρει:
- ασύγχρονες λειτουργίες μέσω του μηχανισμού async/await,
- μηχανισμούς χειρισμού σφαλμάτων,
- plugins ειδικά σχεδιασμένα για mobile περιβάλλοντα.

Χάρη στη χρήση του Firebase SDK, η εφαρμογή υλοποιεί όλη τη λειτουργικότητα backend μέσω ασφαλών και βελτιστοποιημένων μηχανισμών επικοινωνίας, διατηρώντας απλή και καθαρή την αρχιτεκτονική της.

3.3.4 Google Maps SDK

Η εφαρμογή αξιοποιεί το Google Maps SDK for Flutter για την απεικόνιση χαρτών και τον εντοπισμό των σημείων εκκίνησης των διαθέσιμων διαδρομών. Το SDK προσφέρει ολοκληρωμένες λειτουργίες χαρτογράφησης, οι οποίες ενσωματώνονται άμεσα στο UI χωρίς την ανάγκη ανάπτυξης custom map engine. Λειτουργίες που αξιοποιεί η εφαρμογή:

- Εμφάνιση διαδραστικού χάρτη
  Ο χάρτης αποδίδεται μέσα από το widget GoogleMap, προσφέροντας:
    - μεγέθυνση/σμίκρυνση (zoom gestures)
    - μετακίνηση (pan gestures)
    - τη δυνατότητα εμφάνισης πολλών markers.

Η κεντρική οθόνη Home βασίζεται σε αυτό το widget, επιτρέποντας στον χρήστη να δει εύκολα τα διαθέσιμα rides στο campus και στην περιοχή της Σίνδου.

- Markers σημείων εκκίνησης rides
  Κάθε διαθέσιμη διαδρομή (ride) εμφανίζεται ως marker, ο οποίος δημιουργείται από τις συντεταγμένες startLocationCoords που λαμβάνονται από το Firestore.
  Οι markers υποστηρίζουν actions:

- εμφάνιση σύντομων πληροφοριών (info window)
- σύνδεση με την οθόνη λεπτομερειών της διαδρομής

- Offline cache χαρτογραφικών δεδομένων
  Το Google Maps SDK εκτελεί:
  - αυτόματη προσωρινή αποθήκευση (cache) χαρτογραφικών tiles
  - χρήση τους σε συνθήκες περιορισμένης συνδεσιμότητας

Η λειτουργία αυτή βελτιώνει την εμπειρία χρήσης, ειδικά σε εξωτερικούς χώρους με αδύναμο ή ασταθές δίκτυο.

- Εύκολη ενσωμάτωση με Firestore
  Οι markers ενημερώνονται δυναμικά:
  - όταν δημιουργείται νέο ride
  - όταν κάποιο ride ακυρώνεται
  - όταν αλλάζει η ώρα/ημερομηνία μιας διαδρομής

Χάρη στη real-time φύση του Firestore, ο χάρτης ανανεώνεται χωρίς χειροκίνητο refresh.

Λόγοι επιλογής Google Maps SDK:

- Υψηλή ακρίβεια χαρτογράφησης και συνεχής ενημέρωση χαρακτηριστικών των δρόμων.
- Πλήρης ενσωμάτωση στο Flutter μέσω του επίσημου plugin.
- Ευκολία διαχείρισης markers, gestures, events και custom UI components.
- Υποστήριξη caching, χρήσιμο σε εκπαιδευτικό περιβάλλον με πιθανές διακυμάνσεις συνδεσιμότητας.
- Βέλτιστη εμπειρία χρήστη, δεδομένου ότι οι φοιτητές είναι εξοικειωμένοι με το Google Maps.

3.3.5 IHU Apps / SSO

Η εφαρμογή δεν υλοποιεί δικό της μηχανισμό αυθεντικοποίησης, αλλά βασίζεται αποκλειστικά στο σύστημα ιδρυματικής ταυτοποίησης του ΔΙΠΑΕ (IHU Apps). Το IHU Apps λειτουργεί ως Identity Provider (IdP), επιτρέποντας την ασφαλή είσοδο των φοιτητών με τα ιδρυματικά τους διαπιστευτήρια και την παροχή βασικών στοιχείων ταυτότητας στην εφαρμογή.

3.3.5.1 Ρόλος του IHU Apps στην αρχιτεκτονική

Η ταυτοποίηση αποτελεί θεμελιώδες σημείο στην αρχιτεκτονική του συστήματος, καθώς:

- Η εφαρμογή δεν διαχειρίζεται κωδικούς, ούτε αποθηκεύει στοιχεία πρόσβασης.
- Ο χρήστης εισέρχεται μέσω redirect στην πλατφόρμα IHU Apps, όπου ολοκληρώνει τη διαδικασία login.
- Μετά την επιτυχημένη αυθεντικοποίηση, το IHU Apps επιστρέφει στην εφαρμογή:
  - email
  - όνομα χρήστη
  - μοναδικό userID
    χωρίς να εκθέτει τους πραγματικούς κωδικούς.

Αυτό το μοντέλο συμβαδίζει με τις απαιτήσεις ασφαλείας των πανεπιστημιακών συστημάτων και μειώνει σημαντικά τις ευθύνες προστασίας δεδομένων που θα είχε διαφορετικά μια mobile εφαρμογή.

3.3.5.2 Μηχανισμός SSO (Single Sign-On)

Η ταυτοποίηση πραγματοποιείται μέσω ενός απλουστευμένου μηχανισμού SSO, στον οποίο:

1. Ο χρήστης επιλέγει "Log in with IHU Apps".
2. Η εφαρμογή μεταφέρει τον χρήστη μέσω redirect στο IHU login portal.
3. Μετά την επιβεβαίωση της ταυτότητας, το IHU Apps:
    - επιστρέφει τον χρήστη στην εφαρμογή,
    - παρέχει τα απαραίτητα στοιχεία (email, όνομα, userID),
    - εκδίδει ένα OAuth token.
4. Το token αποστέλλεται στο Firebase Authentication, το οποίο:
    - το επικυρώνει,
    - δημιουργεί ή ενημερώνει το session του χρήστη στην εφαρμογή.

Έτσι επιτυγχάνεται πλήρης ταυτοποίηση με συνδυασμό:

- IHU SSO → απόδειξη πραγματικής ταυτότητας φοιτητή
- Firebase Auth → ασφαλής διαχείριση session στην εφαρμογή

3.3.5.3 Πλεονεκτήματα χρήσης ιδρυματικής ταυτοποίησης

Η αξιοποίηση του IHU Apps προσφέρει πολλαπλά οφέλη:

- Ενισχυμένη ασφάλεια
  Δεν ανταλλάσσονται κωδικοί χρήστη, αποτρέποντας διαρροή ευαίσθητων δεδομένων.

- Ευκολία χρήσης
  Οι φοιτητές χρησιμοποιούν τα ίδια credentials που ήδη διαθέτουν για όλες τις υπηρεσίες του ιδρύματος.

- Εξασφάλιση εγκυρότητας συμμετεχόντων
  Μόνο επαληθευμένοι χρήστες (ενεργοί φοιτητές/μέλη του Ιδρύματος) μπορούν να εισέλθουν και να συμμετάσχουν στις διαδρομές.

- Μείωση φόρτου ανάπτυξης
  Η εφαρμογή δεν χρειάζεται:
    - μηχανισμό διαχείρισης κωδικών,
    - password reset,
    - σύστημα επαλήθευσης email.

- Απόλυτη συμβατότητα με το οικοσύστημα Firebase
  Το OAuth token που παρέχει το IHU Apps μετατρέπεται σε Firebase Auth credential, χωρίς επιπλέον ρυθμίσεις.

3.3.6 Visual Studio Code (IDE / Editor)

Το Visual Studio Code (VS Code) χρησιμοποιήθηκε ως ελαφρύ και ευέλικτο περιβάλλον ανάπτυξης για την υλοποίηση και δοκιμή της εφαρμογής. Αποτελεί έναν από τους πιο δημοφιλείς editors για Flutter, χάρη στην υψηλή απόδοση, στη δυνατότητα παραμετροποίησης και στη μεγάλη ποικιλία διαθέσιμων επεκτάσεων. Το VS Code προσφέρει ένα περιβάλλον γρήγορης ανάπτυξης, ιδανικό για επαναληπτικό προγραμματισμό και άμεση ανατροφοδότηση μέσω του Flutter Hot Reload.

Η επιλογή του VS Code βασίστηκε σε συγκεκριμένα πλεονεκτήματα:

- Ελαφρύ IDE με υψηλή απόδοση, κατάλληλο για συστήματα χαμηλότερων πόρων σε σχέση με το Android Studio.
- Ενσωματωμένη υποστήριξη Flutter και Dart μέσω επίσημων plugins, που προσφέρουν syntax highlighting, debugging, code completion και εργαλεία ανάλυσης κώδικα.
- Ευκολία στη διαχείριση πακέτων μέσω του πακέτου Flutter και του αρχείου pubspec.yaml.
- Ενεργό debugging με breakpoints, real-time logs και δυνατότητα εκτέλεσης της εφαρμογής σε εξωτερικό emulator ή φυσική συσκευή.
- Ευρεία γκάμα επεκτάσεων (GitLens, Firebase Explorer κ.λπ.) που απλοποιούν τη συνεργασία, τον έλεγχο εκδόσεων και την παρακολούθηση δεδομένων στο Firestore.
- Ταχύτητα στην ανάπτυξη UI χάρη στην άμεση εφαρμογή Hot Reload / Hot Restart και στη δυνατότητα split-view για ταυτόχρονη προβολή πολλαπλών αρχείων.

Το VS Code συνέβαλε στη γρήγορη και ευέλικτη ανάπτυξη της εφαρμογής, προσφέροντας ένα περιβάλλον ιδιαίτερα φιλικό προς τον προγραμματιστή και πλήρως συμβατό με το οικοσύστημα Flutter.

3.3.7 Android Studio (IDE)

Το Android Studio χρησιμοποιήθηκε ως το κύριο Αναπτυξιακό Περιβάλλον (IDE) για την υλοποίηση της εφαρμογής, προσφέροντας ένα ολοκληρωμένο σύνολο εργαλείων που διευκολύνουν την ανάπτυξη, δοκιμή και αποσφαλμάτωση (debugging) εφαρμογών Flutter [31]. Παρότι το Flutter είναι cross-platform και μπορεί να χρησιμοποιηθεί σε διάφορα IDEs, το Android Studio παρέχει την πιο ολοκληρωμένη εμπειρία χάρη στην πλήρη ενσωμάτωσή του με το Flutter SDK, τα plugins της Google και τα Android developer tools.

Η επιλογή του Android Studio βασίστηκε σε συγκεκριμένα πλεονεκτήματα:

- Ενσωματωμένη υποστήριξη Flutter και Dart μέσω των αντίστοιχων plugins, τα οποία προσφέρουν auto-completion, debugging εργαλεία, ανάλυση κώδικα και άμεση προεπισκόπηση UI.
- Ισχυρό σύστημα debugging, με δυνατότητα τοποθέτησης breakpoints, παρακολούθησης μεταβλητών και real-time profiling.
- Android Emulator υψηλής απόδοσης, που επιτρέπει την εκτέλεση και δοκιμή της εφαρμογής σε πολλαπλές συσκευές χωρίς φυσικό κινητό.
- Ενσωματωμένα εργαλεία build & Gradle, τα οποία διευκολύνουν τη δημιουργία release APKs / AABs.
- Flutter Inspector, χρήσιμος για έλεγχο layout, ιδιοτήτων widgets και rendering tree.
- Hot Reload & Hot Restart, που επιτρέπουν ταχεία ανάπτυξη και άμεση οπτική ανατροφοδότηση κατά τη διάρκεια της υλοποίησης.

Επιπλέον, η χρήση του Android Studio επέτρεψε την ομαλή διαχείριση του έργου σε συνδυασμό με το GitHub, μέσω του ενσωματωμένου Version Control System, διευκολύνοντας commit, branching και παρακολούθηση αλλαγών.

Συνολικά, το Android Studio αποτέλεσε το κύριο εργαλείο ανάπτυξης της εφαρμογής, εξασφαλίζοντας σταθερότητα, παραγωγικότητα και εύκολη ενοποίηση με όλες τις τεχνολογίες του project.

3.3.8 GitHub

Το GitHub χρησιμοποιήθηκε ως το βασικό σύστημα διαχείρισης εκδόσεων (Version Control System – VCS) και συνεργασίας κατά την ανάπτυξη της εφαρμογής. Προσέφερε ένα σταθερό και ασφαλές περιβάλλον για την αποθήκευση του κώδικα, την οργάνωση των αλλαγών και τον συγχρονισμό της εργασίας μεταξύ διαφορετικών σταδίων ανάπτυξης (development, debugging, testing).

Η χρήση του GitHub παρείχε τα ακόλουθα πλεονεκτήματα:

- Ασφαλής αποθήκευση του κώδικα στο cloud, με δυνατότητα πρόσβασης από οποιαδήποτε συσκευή.
- Ακριβές ιστορικό αλλαγών (commit history), το οποίο επιτρέπει την αναλυτική καταγραφή της εξέλιξης του έργου και τη δυνατότητα επαναφοράς σε προηγούμενες εκδόσεις (rollback).
- Διαχείριση παρακλαδιών (branching), επιτρέποντας την ανεξάρτητη ανάπτυξη νέων λειτουργιών χωρίς να επηρεάζεται η κύρια έκδοση της εφαρμογής.
- Pull Requests (PRs) για έλεγχο και συγχώνευση αλλαγών, προσφέροντας μεγαλύτερη διαφάνεια και έλεγχο στην ενσωμάτωση νέου κώδικα.
- Συνδυασμός με το Android Studio, το οποίο διαθέτει ενσωματωμένα εργαλεία Git, επιτρέποντας εύκολο commit, push, pull και συγχρονισμό χωρίς ανάγκη εξωτερικών εργαλείων.
- Αυτόματος έλεγχος συγκρούσεων (merge conflicts), διασφαλίζοντας τη σταθερότητα του κώδικα σε περιβάλλον με συνεχείς αλλαγές.
- Δυνατότητα χρήσης Issues και Project Boards, εφόσον χρειάζεται, για την οργάνωση εργασιών, την παρακολούθηση bugs και την καταγραφή νέων απαιτήσεων.

Συνολικά, το GitHub συνέβαλε στην αποδοτική διαχείριση του κώδικα και στην ομαλή οργάνωση της ανάπτυξης της εφαρμογής, εξασφαλίζοντας σταθερότητα, διαφάνεια και ευκολία διαχείρισης του κώδικα.

3.4 Σχεδίαση διεπαφής χρήστη (UI/UX)

Η σχεδίαση της διεπαφής χρήστη (UI) και η συνολική εμπειρία χρήσης (UX) αποτελούν κρίσιμους παράγοντες για τη λειτουργικότητα και την αποδοχή της εφαρμογής από τους φοιτητές. Στο πλαίσιο της ανάπτυξης του συστήματος carpooling του ΔΙΠΑΕ, η UI/UX σχεδίαση εστιάζει στη δημιουργία μιας απλής, καθαρής και διαισθητικής διεπαφής, η οποία μειώνει τον γνωστικό φόρτο και υποστηρίζει την άμεση κατανόηση των βασικών λειτουργιών της εφαρμογής.

Κατά τη διαδικασία σχεδιασμού ακολουθήθηκαν καθιερωμένες αρχές χρηστικότητας και διεθνείς πρακτικές του πεδίου, όπως οι δέκα Heuristics for User Interface Design των Nielsen-Norman Group [28], καθώς και βασικές κατευθυντήριες οδηγίες UI που προτείνονται από το Material Design της Google [32]. Στόχος είναι η διασφάλιση συνέπειας, η άμεση ανατροφοδότηση, η ευκολία πλοήγησης και η ελαχιστοποίηση πιθανών λαθών κατά την αλληλεπίδραση του χρήστη με το σύστημα.

Η παρούσα ενότητα παρουσιάζει τις σχεδιαστικές αρχές, τις χρωματικές επιλογές, τη διάταξη της πλοήγησης και τις βασικές οθόνες του συστήματος, χωρίς να εισέλθει σε λεπτομέρειες υλοποίησης ή screenshots, τα οποία αναλύονται εκτενώς στο Κεφάλαιο 4.

3.4.1 Σχεδιαστικές αρχές και στόχοι

Η σχεδίαση της εφαρμογής βασίστηκε σε ένα σύνολο αρχών που στοχεύουν στη διασφάλιση μιας ομοιόμορφης, απλής και φιλικής εμπειρίας για τον τελικό χρήστη. Οι κύριες σχεδιαστικές αρχές που υιοθετήθηκαν είναι οι εξής:

- Απλότητα και καθαρότητα διεπαφής
  Η διεπαφή διατηρεί χαμηλή οπτική πολυπλοκότητα, με λιτό layout, σαφείς ετικέτες και περιορισμένη χρήση χρωμάτων. Η αρχή αυτή μειώνει το γνωστικό φορτίο και διευκολύνει τη γρήγορη κατανόηση των διαθέσιμων λειτουργιών.
- Συνέπεια και προβλεψιμότητα
  Όλα τα κουμπιά, κάρτες και τυπογραφικά στοιχεία ακολουθούν κοινά πρότυπα και διατηρούν συνεπή συμπεριφορά σε όλη την εφαρμογή, σύμφωνα με τις οδηγίες Material Design [32]. Με αυτόν τον τρόπο, ο χρήστης δεν χρειάζεται να μάθει νέες αλληλεπιδράσεις ανά οθόνη.
- Άμεση ανατροφοδότηση (Feedback)
  Η εφαρμογή παρέχει άμεση οπτική ή κειμενική ανατροφοδότηση σε κάθε ενέργεια του χρήστη: υποβολή αιτήματα, επεξεργασία rides, ενημερώσεις προφίλ κ.λπ. Η αρχή αυτή συνάδει με τα usability heuristics της NNG σχετικά με την ορατότητα της κατάστασης του συστήματος [28].
- Μείωση πιθανότητας λαθών
  Φόρμες, ημερομηνίες και εισαγωγή κειμένου έχουν σχεδιαστεί ώστε να περιορίζουν σφάλματα (validators, περιορισμοί εισόδου, επιβεβαιώσεις πριν από κρίσιμες ενέργειες). Αυτό ενισχύει την αξιοπιστία του συστήματος και μειώνει την ανάγκη επανυποβολών.
- Προσβασιμότητα
  Δίνεται έμφαση στην επαρκή αντίθεση χρωμάτων, στο κατάλληλο μέγεθος γραμματοσειράς και στην ευκρίνεια των στοιχείων πλοήγησης. Παρά το ότι δεν εφαρμόζεται πλήρης WCAG συμμόρφωση, η εφαρμογή ακολουθεί βασικές αρχές προσβασιμότητας για κινητές συσκευές.

3.4.2 Θέματα χρωματικής παλέτας και typography

Η χρωματική παλέτα και η επιλογή γραμματοσειρών αποτελούν βασικά στοιχεία της σχεδίασης της διεπαφής χρήστη, καθώς επηρεάζουν άμεσα την ευκρίνεια, τη συνοχή και την αισθητική ταυτότητα της εφαρμογής. Στην παρούσα εφαρμογή, η επιλογή των χρωμάτων έγινε με στόχο τη δημιουργία μιας καθαρής, λειτουργικής και ευχάριστης οπτικής εμπειρίας, κατάλληλης για καθημερινή χρήση από φοιτητές.

3.4.2.1 Χρωματική παλέτα

Η εφαρμογή υιοθετεί σκοτεινή χρωματική παλέτα (dark theme) ως βασικό οπτικό μοτίβο. Το μεγαλύτερο μέρος των οθονών (Home, Profile, Settings, Chat, φόρμες επεξεργασίας) χρησιμοποιεί πολύ σκούρο γκρι/μαύρο φόντο, πάνω στο οποίο το κείμενο αποδίδεται με λευκό ή ανοιχτό γκρι. Με αυτόν τον τρόπο επιτυγχάνεται υψηλή αντίθεση και άνετη ανάγνωση, ειδικά σε συνθήκες χαμηλού φωτισμού, ενώ παράλληλα μειώνεται η οπτική κόπωση κατά την χρήση του χάρτη.

Η οθόνη εισαγωγής (splash / login) ακολουθεί μια δίχρωμη προσέγγιση: το πάνω τμήμα έχει ανοιχτό μωβ-γκρι φόντο, ώστε να αναδεικνύεται το λογότυπο του ΔΙΠΑΕ και το κεντρικό μήνυμα (“Car Pooling. Share the experience”), ενώ το κάτω τμήμα παραμένει σκούρο, φιλοξενώντας το βασικό call-to-action κουμπί “Log in with IHU Apps” με λευκό φόντο και σκούρο κείμενο.

Για την ανάδειξη κατάστασης και δράσης χρησιμοποιούνται σταθερά χρώματα έμφασης:

- Μπλε για primary actions και στοιχεία πλοήγησης (κουμπιά Edit Vehicle, Edit Ride, κουμπί αποστολής μηνύματος, bubbles του chat).
- Πράσινο για θετικές ή ολοκληρωμένες ενέργειες (Save Changes, Complete Ride).
- Κόκκινο για κρίσιμες ή καταστροφικές ενέργειες (Delete Vehicle, Cancel Ride, Delete Account) καθώς και για ενδείξεις κατάστασης όπως Pending.

3.4.2.2 Typography

Η τυπογραφία βασίζεται σε μοντέρνα sans-serif γραμματοσειρά (τυπική του Material Design), με σαφή ιεραρχία:

- Μεγάλοι, έντονοι τίτλοι στο app bar (π.χ. Profile, Settings, Chats, Report a Problem).
- Μεσαίας έντασης labels για πεδία φόρμας και πληροφορίες (π.χ. Vehicle Model, License Plate, Start Location).
- Απλό σώμα κειμένου για δευτερεύουσες πληροφορίες (ημερομηνίες, περιγραφές, τελευταία μηνύματα).
- Κουμπιά με σύντομα λεκτικά σε Title Case (Offer A Ride, Submit, Save Changes), ώστε να είναι άμεσα αναγνωρίσιμα ως ενέργειες.

Η συνέπεια στη χρήση χρωμάτων και typographic ιεραρχίας βοηθά τον χρήστη να αναγνωρίζει γρήγορα πού εμφανίζεται πληροφορία, πού υπάρχει διαθέσιμη ενέργεια και ποια στοιχεία απαιτούν προσοχή (κόκκινα/πράσινα κουμπιά), ενισχύοντας την συνολική ευχρηστία του UI.

3.4.3 Διάταξη και πλοήγηση (Navigation Structure)

Η εφαρμογή ακολουθεί μια απλή και γραμμική αρχιτεκτονική πλοήγησης, εστιάζοντας στη σαφήνεια και την εύκολη πρόσβαση στις βασικές λειτουργίες. Δεν χρησιμοποιείται bottom navigation bar ή πολύπλοκο tab layout, αντίθετα, η πλοήγηση βασίζεται σε μια κεντρική αρχική οθόνη (Home) και σε συνεχή μετάβαση μέσω κουμπιών και app bar back buttons, ώστε το UI να παραμένει καθαρό και προσανατολισμένο στις κύριες ενέργειες του χρήστη.

3.4.3.1 Κεντρική πλοήγηση μέσω Home Screen

Μετά τη διαδικασία εισόδου μέσω IHU Apps, ο χρήστης μεταφέρεται στην Home Screen, η οποία λειτουργεί ως κόμβος της εφαρμογής:

- Προβάλλεται ο χάρτης με τα σημεία εκκίνησης διαδρομών.
- Διαθέτει κουμπί Offer A Ride για δημιουργία νέας διαδρομής.
- Περιλαμβάνει side-button (hamburger-style) για πρόσβαση σε Profile, Settings, Report a Problem και Chats.
- Εμφανίζει τη λίστα “Available Rides”, διευκολύνοντας την άμεση μετάβαση στη λεπτομερή προβολή μιας διαδρομής.

Η Home Screen επομένως συνδυάζει πλοήγηση και περιεχόμενο στην ίδια σελίδα, μειώνοντας το depth-level της εφαρμογής.

3.4.3.2 Πλοήγηση μέσω AppBar Back Button

Οι περισσότερες δευτερεύουσες οθόνες (Profile, Edit Vehicle, Edit Ride, Settings, Report a Problem, Chats, Chat Room) χρησιμοποιούν την κλασική πίσω πλοήγηση μέσω του βέλους της επάνω μπάρας.

Αυτό επιτρέπει μια μονοκατευθυντική, ιεραρχική ροή που είναι εύκολη για τον χρήστη να προβλέψει:

- Home → Profile → Edit Vehicle
- Home → Available Rides → Ride Details → Chat
- Home → Chats → Chat Room
- Home → Settings → Delete Account

Αυτό το μοντέλο μειώνει τη γνωστική φόρτιση, αφού ο χρήστης έχει πάντα ξεκάθαρη διαδρομή επιστροφής.

3.4.3.3 Κουμπιά δράσης για κάθε ενότητα

Η εφαρμογή βασίζεται σε ξεκάθαρα action buttons για μετάβαση σε λειτουργίες όπως:

- Edit Vehicle
- Edit Ride
- Cancel Ride / Complete Ride
- Submit (σε φόρμες)
- Send Message (στην οθόνη chat)

Αυτό κάνει τη ροή της εφαρμογής task-oriented, όπου η πλοήγηση συνοδεύεται από ξεκάθαρες ενέργειες.

3.4.3.4 Πλοήγηση στο Chat

Το chat έχει δύο επίπεδα:

1. Λίστα συνομιλιών (Chats)
2. Προβολή συγκεκριμένης συνομιλίας (Chat Room)

Το μοντέλο είναι παρόμοιο με messaging apps, ώστε ο χρήστης να αναγνωρίζει άμεσα την λειτουργία.

3.4.3.5 Απουσία περιττών navigation layers

Η εφαρμογή δεν περιλαμβάνει:

- bottom navigation,
- tabs,
- πολύπλοκα nested navigators.

Αυτό είναι συνειδητή σχεδιαστική επιλογή που ταιριάζει σε εφαρμογές με ξεκάθαρα καθορισμένη λειτουργικότητα όπως το carpooling.

3.4.4 Ανάλυση βασικών οθονών της εφαρμογής

Η εφαρμογή αποτελείται από ένα σύνολο διακριτών οθονών, καθεμία από τις οποίες εξυπηρετεί συγκεκριμένες λειτουργικές ανάγκες του χρήστη. Στην ενότητα αυτή παρουσιάζονται οι βασικές οθόνες με έμφαση στη δομή τους, το περιεχόμενο και τη λογική αλληλεπίδρασης που υποστηρίζουν.

3.4.4.1 Οθόνη Εισόδου (Login Screen)

Η αρχική οθόνη αποτελεί το πρώτο σημείο επαφής του χρήστη με την εφαρμογή.

Χαρακτηριστικά:

- Εμφάνιση λογοτύπου και ονομασίας του ΔΙΠΑΕ.
- Σύντομο tagline “Car Pooling. Share the experience”.
- Κεντρικό κουμπί “Log in with IHU Apps”, το οποίο ενεργοποιεί τη διαδικασία SSO μέσω των ιδρυματικών υπηρεσιών.
- Footer με το copyright της εφαρμογής.

Ο σχεδιασμός είναι μινιμαλιστικός, χωρίς περιττές πληροφορίες, υποστηρίζοντας γρήγορη είσοδο.

3.4.4.2 Κεντρική Οθόνη (Home Screen / Map + Available Rides)

Η κεντρική οθόνη λειτουργεί ως κόμβος της εφαρμογής.

- Μεγάλος διαδραστικός χάρτης Google Maps.
- Markers που αντιπροσωπεύουν σημεία εκκίνησης διαδρομών.
- Κουμπί “Offer A Ride” στην πάνω δεξιά γωνία για δημιουργία νέας διαδρομής.
- Πλευρικό κουμπί τύπου hamburger για πρόσβαση σε επιπλέον λειτουργίες.
- Κάτω τμήμα με τη λίστα “Available Rides”, που παρουσιάζει διαθέσιμες διαδρομές άλλων χρηστών.

Η οθόνη σχεδιάζεται ως “dashboard” του οδηγού και επιβάτη.

3.4.4.3 Οθόνη Προφίλ Χρήστη (Profile Screen)

Παρέχει συνολική εικόνα της δραστηριότητας του χρήστη.

Περιλαμβάνει:

- Προφίλ εικόνα (placeholder).
- Όνομα χρήστη από το IHU Apps.
- Στατιστικά: Rides Taken, Rides Completed.
- Στοιχεία οχήματος: μοντέλο και πινακίδα.
- Κουμπί Edit Vehicle για επεξεργασία στοιχείων οχήματος.
- Λίστα Your Rides, όπου εμφανίζονται οι διαδρομές που έχει δημιουργήσει ο χρήστης.
- Κουμπί Edit Ride σε κάθε διαδρομή για τροποποίηση.

Η οθόνη βοηθά τον χρήστη να παρακολουθεί τη δραστηριότητά του και να διαχειρίζεται τα στοιχεία του.

3.4.4.4 Οθόνη Επεξεργασίας Οχήματος (Edit Vehicle Screen)

Απλή φόρμα με δύο πεδία:

- Vehicle Model
- Vehicle Plate

Κάτω μέρος:

- Κουμπί Delete Vehicle (κόκκινο).
- Κουμπί Save Changes (πράσινο).

Η απλότητα της οθόνης εξασφαλίζει γρήγορη και ακριβή επεξεργασία.

3.4.4.5 Οθόνη Επεξεργασίας Διαδρομής (Edit Ride Screen)

Εμφανίζει συνοπτικά τα στοιχεία της διαδρομής:

- Starting Location
- Driver
- Date & Time (με δυνατότητα επεξεργασίας)

Διαθέτει δύο κύρια κουμπιά ενεργειών:

- Cancel Ride (κόκκινο)
- Complete Ride (πράσινο)

Επιτρέπει εύκολη διαχείριση της κατάστασης της διαδρομής.

3.4.4.6 Οθόνη Ρυθμίσεων (Settings Screen)

Αποτελείται από ένα μόνο στοιχείο:

- Delete Account

Η σχεδιαστική επιλογή της απλότητας αντανακλά τον περιορισμένο αριθμό διαθέσιμων ρυθμίσεων, προσφέροντας καθαρή και άμεση εμπειρία.

3.4.4.7 Οθόνη Αναφοράς Προβλήματος (Report a Problem)

Περιλαμβάνει:

- Μεγάλο πεδίο κειμένου για την περιγραφή του προβλήματος.
- Κουμπί Submit σε λευκό φόντο.

Κατάλληλη για την αναφορά προβλημάτων από τους χρήστες με ελάχιστες απαιτούμενες ενέργειες.

3.4.4.8 Λίστα Συνομιλιών (Chats Screen)

Παρουσιάζει τις τελευταίες ενεργές συνομιλίες του χρήστη:

- Όνομα συνομιλητή
- Προεπισκόπηση τελευταίου μηνύματος
- Χρονική σήμανση τελευταίας δραστηριότητας

Κάθε στοιχείο της λίστας οδηγεί σε πραγματικό chat room.

3.4.4.9 Οθόνη Συνομιλίας (Chat Screen)

Πλήρης οθόνη ανταλλαγής μηνυμάτων:

- Φυσαλίδες μηνυμάτων με χρωματική διαφοροποίηση.
- Συνεχόμενη προβολή μηνυμάτων (scrollable).
- Πεδίο εισαγωγής κειμένου στο κάτω μέρος.
- Κουμπί αποστολής (send icon).

Η υλοποίηση ακολουθεί κλασική διάταξη messaging apps για μέγιστη οικειότητα στον χρήστη.

3.4.5 UX βελτιστοποιήσεις & Επιλογές Προσβασιμότητας (Accessibility Choices)

Η εφαρμογή έχει σχεδιαστεί με στόχο τη μέγιστη απλότητα και λειτουργικότητα, ακολουθώντας καθιερωμένες πρακτικές UX και βασικές οδηγίες προσβασιμότητας. Στο πλαίσιο αυτό, υιοθετήθηκαν

μια σειρά από επιλογές που βελτιώνουν τη συνολική εμπειρία χρήσης, μειώνουν τα λάθη, ενισχύουν τη σαφήνεια και καθιστούν το σύστημα πιο φιλικό σε διαφορετικούς τύπους χρηστών.

1. Ελαχιστοποίηση γνωστικού φόρτου

   Η πλοήγηση είναι γραμμική και ξεκάθαρη: κάθε οθόνη έχει έναν μόνο κεντρικό σκοπό (login → home → ride details → requests → chat). Τα περιττά στοιχεία έχουν αφαιρεθεί ώστε ο χρήστης να επικεντρώνεται μόνο στη βασική λειτουργία της στιγμής.

2. Σαφείς και άμεσες επιλογές δράσης

   Κουμπιά όπως Create Ride, Send Request, Approve, Reject χρησιμοποιούν καθαρή, περιγραφική ορολογία που δεν επιτρέπει παρερμηνείες.

   Επιπλέον, τοποθετούνται σε εμφανή, σταθερά σημεία της οθόνης.

3. Χρωματικός κώδικας ανά λειτουργία
   - Κύρια κουμπιά: μπλε χρώμα → άμεση ενέργεια
   - Ενέργειες έγκρισης: πράσινη απόχρωση
   - Ενέργειες απόρριψης/ακύρωσης: κόκκινη απόχρωση

   Ο χρωματικός διαχωρισμός αυξάνει τη χρηστικότητα χωρίς να απαιτεί επιπλέον κείμενο.

4. Ομοιομορφία στοιχείων UI

   Όλες οι οθόνες ακολουθούν το ίδιο pattern:

   - ίδια typography
   - σταθερό padding
   - ίδιοι τίτλοι header
   - ίδιο χρωματικό σύστημα

   Αυτή η συνέπεια μειώνει το learning curve και ενισχύει την προβλεψιμότητα των ενεργειών.

5. Καθαρή παρουσίαση δεδομένων

   Λίστες rides, λίστες επιβατών και λίστες αιτημάτων είναι οργανωμένες σε card-based layout, με:

   - τίτλο
   - υπότιτλο
   - κατάσταση
   - εικονίδιο/χρώμα

   Η διάταξη αυτή διευκολύνει γρήγορη σάρωση (scannability).

6. Feedback μετά από κάθε ενέργεια

   Η εφαρμογή ενημερώνει άμεσα τον χρήστη σε όλες τις κρίσιμες ενέργειες:

   - “Request Sent”
   - “Ride Updated”
   - “Request Approved/Rejected”
   - error messages για συνδεσιμότητα
   - alerts για απουσία θέσεων

   Το άμεσο feedback αποτελεί βασική UX αρχή (σύμφωνα με Nielsen Norman Group [28]).

7. Επιλογές Προσβασιμότητας (Accessibility)

   Παρότι η εφαρμογή δεν αποτελεί εμπορικό προϊόν, τηρήθηκαν βασικές πρακτικές προσβασιμότητας:

   - υψηλή αντίθεση κειμένου
   - μεγάλοι clickable targets στα κουμπιά
   - σταθερή διάταξη layout χωρίς "jumping elements"
   - ευανάγνωστα χρώματα (ειδικά στα κουμπιά approve/reject)
   - απουσία περιττών animations που θα μπορούσαν να δυσκολέψουν χρήστες με μαθησιακές δυσκολίες
8. Mobile-first σχεδιασμός

   Η εφαρμογή έχει δημιουργηθεί αποκλειστικά για μικρές οθόνες:

   - κάθε οθόνη εμφανίζει μόνο τον απολύτως απαραίτητο αριθμό στοιχείων
   - όλες οι δράσεις γίνονται με ένα χέρι
   - κουμπιά μεγάλης επιφάνειας για ευκολία αφής
9. Αποφυγή λαθών (Error Prevention)

   Σημαντικά UX μέτρα περιλαμβάνουν:

   - απενεργοποίηση του Send Request όταν δεν υπάρχουν διαθέσιμες θέσεις
   - μηνύματα για άκυρη πινακίδα/ελλιπές προφίλ
   - μη δυνατότητα αλλαγής start location (ώστε να αποφευχθεί ασυνέπεια δεδομένων)

3.4.6 Συνολική αξιολόγηση UI/UX επιλογών

Η συνολική αξιολόγηση του UI/UX της εφαρμογής δείχνει ότι η σχεδίαση ακολουθεί τις βασικές αρχές ευχρηστίας, προβλεψιμότητας και απλότητας, εξασφαλίζοντας μια ομαλή εμπειρία χρήσης για ένα κοινό που δεν έχει απαραίτητα τεχνικές γνώσεις. Η δομή των οθονών, η πλοήγηση και η παρουσίαση της πληροφορίας έχουν σχεδιαστεί με γνώμονα τη μείωση της γνωστικής επιβάρυνσης και την αποφυγή σφαλμάτων, δύο παράγοντες ιδιαίτερα κρίσιμοι σε εφαρμογές καθημερινής χρήσης.

Η χρήση λιτής και καθαρής διεπαφής, χωρίς περιττά γραφικά και με ξεκάθαρες επιλογές δράσης, επιτρέπει στον χρήστη να κατανοεί άμεσα τον σκοπό κάθε οθόνης. Οι λειτουργίες οργανώνονται με τρόπο που υποστηρίζει την ιεράρχηση της πληροφορίας: ο χρήστης βλέπει πρώτα τις διαθέσιμες διαδρομές, στη συνέχεια έχει πρόσβαση στις λεπτομέρειες, και τέλος αποκτά τη δυνατότητα να εκτελέσει ενέργειες όπως αποστολή αιτήματος, έγκριση ή συνομιλία. Η ροή αυτή συνάδει πλήρως με τις ευρετικές αρχές ευχρηστίας του Nielsen Norman Group [28], οι οποίες προτείνουν σαφήνεια, συνέπεια και άμεση ανατροφοδότηση.

Αντίστοιχα, η εφαρμογή επιτυγχάνει ιδιαίτερα καλή ισορροπία ανάμεσα στη λειτουργικότητα και την απλότητα. Παρότι υλοποιεί διαδικασίες που εμπλέκουν πολλαπλούς χρήστες (οδηγούς και επιβάτες), αιτήματα, ενημερώσεις, συζητήσεις και διαχείριση δεδομένων σε πραγματικό χρόνο, η διεπαφή καταφέρνει να παραμένει οικεία και εύχρηστη. Ο χρήστης δεν χρειάζεται να παρακολουθήσει περίπλοκες διαδικασίες, καθώς κάθε ενέργεια συνοδεύεται από κατάλληλη οπτική και λεκτική ανατροφοδότηση, μειώνοντας την πιθανότητα λαθών και την ανάγκη για τεχνική υποστήριξη.

Επιπλέον, η υιοθέτηση ενιαίας χρωματικής παλέτας, σταθερής τυπογραφίας και κοινών UI patterns συμβάλλει σημαντικά στη συνοχή της εφαρμογής. Η συνέπεια αυτή ενισχύει την προβλεψιμότητα της εμπειρίας, μία από τις πιο σημαντικές παραμέτρους σε εφαρμογές που χρησιμοποιούνται συχνά και σε

πραγματικές συνθήκες μετακίνησης. Η διάταξη των κουμπιών, η τοποθέτηση των τίτλων και ο τρόπος οργάνωσης της πληροφορίας παραμένουν σταθερά σε όλες τις οθόνες, κάτι που διευκολύνει την ταχεία εξοικείωση του χρήστη.

Συνολικά, οι επιλογές UI/UX που υιοθετήθηκαν υποστηρίζουν αποτελεσματικά τους στόχους της εφαρμογής: απλότητα, λειτουργικότητα, σαφήνεια και αξιοπιστία. Η διεπαφή παραμένει εύχρηστη και κατανοητή, ακόμη και χωρίς προηγούμενη εμπειρία σε εφαρμογές carpooling, γεγονός που την καθιστά κατάλληλη για την ακαδημαϊκή κοινότητα στην οποία απευθύνεται. Μέσω της σωστής ισορροπίας ανάμεσα σε σχεδιαστική λιτότητα και λειτουργικό βάθος, η εφαρμογή επιτυγχάνει μια συνολικά θετική και χρηστικά αποδοτική εμπειρία χρήσης.

3.5 Σχεδίαση βάσης δεδομένων (Firebase / Firestore schema)

Η σχεδίαση της βάσης δεδομένων αποτελεί κρίσιμο στοιχείο για τη λειτουργική αρτιότητα της εφαρμογής carpooling, καθώς καθορίζει τον τρόπο με τον οποίο αποθηκεύονται, οργανώνονται και ανακτώνται τα δεδομένα των χρηστών, των διαδρομών και των επιμέρους ενεργειών του συστήματος. Για τις ανάγκες της εφαρμογής επιλέχθηκε η χρήση του Firebase Cloud Firestore, μιας ευέλικτης NoSQL βάσης δεδομένων που υποστηρίζει αποθήκευση σε πραγματικό χρόνο, εύκολη κλιμάκωση και άμεση ενσωμάτωση με το οικοσύστημα Firebase και Flutter.

Στην παρούσα ενότητα παρουσιάζονται η συλλογιστική πίσω από την επιλογή του Firestore και η γενική αρχιτεκτονική αποθήκευσης δεδομένων, πριν ακολουθήσει η αναλυτική τεκμηρίωση των επιμέρους collections και της δομής τους.

3.5.1 Επιλογή Firebase Firestore

Το Firebase Cloud Firestore επελέγη ως κύρια βάση δεδομένων της εφαρμογής λόγω των χαρακτηριστικών του που ευθυγραμμίζονται με τις απαιτήσεις ενός δυναμικού mobile συστήματος carpooling. Ως NoSQL document-based database, το Firestore παρέχει ευελιξία στη δομή των δεδομένων, επιτρέποντας στο σύστημα να εξελίσσεται χωρίς ανάγκη για αυστηρά schemas ή πολύπλοκες σχέσεις μεταξύ πινάκων.

Βασικά πλεονεκτήματα που δικαιολογούν την επιλογή του Firestore είναι:

- Αμφίδρομη επικοινωνία σε πραγματικό χρόνο (real-time listeners)
  Οι αλλαγές στα δεδομένα εμφανίζονται άμεσα στην εφαρμογή χωρίς επιπλέον αιτήματα, στοιχείο ιδανικό για λειτουργίες όπως requests συμμετοχής, ενημέρωση διαθέσιμων θέσεων, ή completion rides.
- Native υποστήριξη στο Flutter μέσω του Firebase SDK
  Η ενσωμάτωση είναι απλή, σταθερή και πλήρως τεκμηριωμένη, μειώνοντας σημαντικά την πολυπλοκότητα υλοποίησης.
- Ασφαλής πρόσβαση μέσω Firebase Authentication
  Η ταυτοποίηση των χρηστών μέσω IHU Apps επιτρέπει την ασφαλή αντιστοίχιση των εγγράφων με μοναδικούς χρήστες, χωρίς διαχείριση κωδικών ή ευαίσθητων δεδομένων από την εφαρμογή.
- Υψηλή κλιμακωσιμότητα και χαμηλό κόστος
  Κατάλληλο για εφαρμογές που ενδέχεται να αυξήσουν τον όγκο των δεδομένων, χωρίς ανάγκη μελλοντικής μετεγκατάστασης.

- Ευκολία στη δομή τύπου “modules/collections”
  Η ανεξαρτησία των collections (users, rides, requests) επιτρέπει καθαρή αρχιτεκτονική και απλοποιεί την επέκταση της λειτουργικότητας.

Η φύση του Firestore προσαρμόζεται απόλυτα σε ένα περιβάλλον εφαρμογής όπως το carpooling, όπου οι λειτουργίες είναι έντονα event-driven και απαιτούν συνεχή ενημέρωση της διεπαφής του χρήστη.

3.5.2 Γενική αρχιτεκτονική αποθήκευσης δεδομένων

Η αρχιτεκτονική αποθήκευσης της εφαρμογής βασίζεται στην οργάνωση των δεδομένων σε βασικές collections, καθεμία από τις οποίες περιλαμβάνει έγγραφα (documents) με συγκεκριμένες πληροφορίες. Η δομή είναι ιεραρχικά επίπεδη, επιτρέποντας γρήγορη πρόσβαση και ευελιξία, χωρίς περίπλοκες συσχετίσεις.

Στο υψηλό επίπεδο, η Firestore βάση δεδομένων της εφαρμογής οργανώνεται γύρω από τις παρακάτω κεντρικές συλλογές:

- users
  Περιέχει τα προφίλ των χρηστών που εισέρχονται μέσω IHU Apps.

  Κάθε document αποθηκεύει στοιχεία ταυτοποίησης, πληροφορίες οχήματος (εφόσον έχουν δηλωθεί) και βοηθητικές ρυθμίσεις.

- rides
  Περιλαμβάνει όλες τις διαδρομές που δημιουργούνται από οδηγούς.

  Σε κάθε ride αποθηκεύονται στοιχεία όπως συντεταγμένες αφετηρίας, ημερομηνία/ώρα, διαθέσιμες θέσεις, requests και επιβεβαιωμένοι επιβάτες.

- messages
  Περιέχει συνομιλίες μεταξύ οδηγών και επιβατών, οργανωμένες ανά διαδρομή ή ανά ζεύγος χρηστών, και υποστηρίζει real-time ανταλλαγή μηνυμάτων μέσω Firestore.

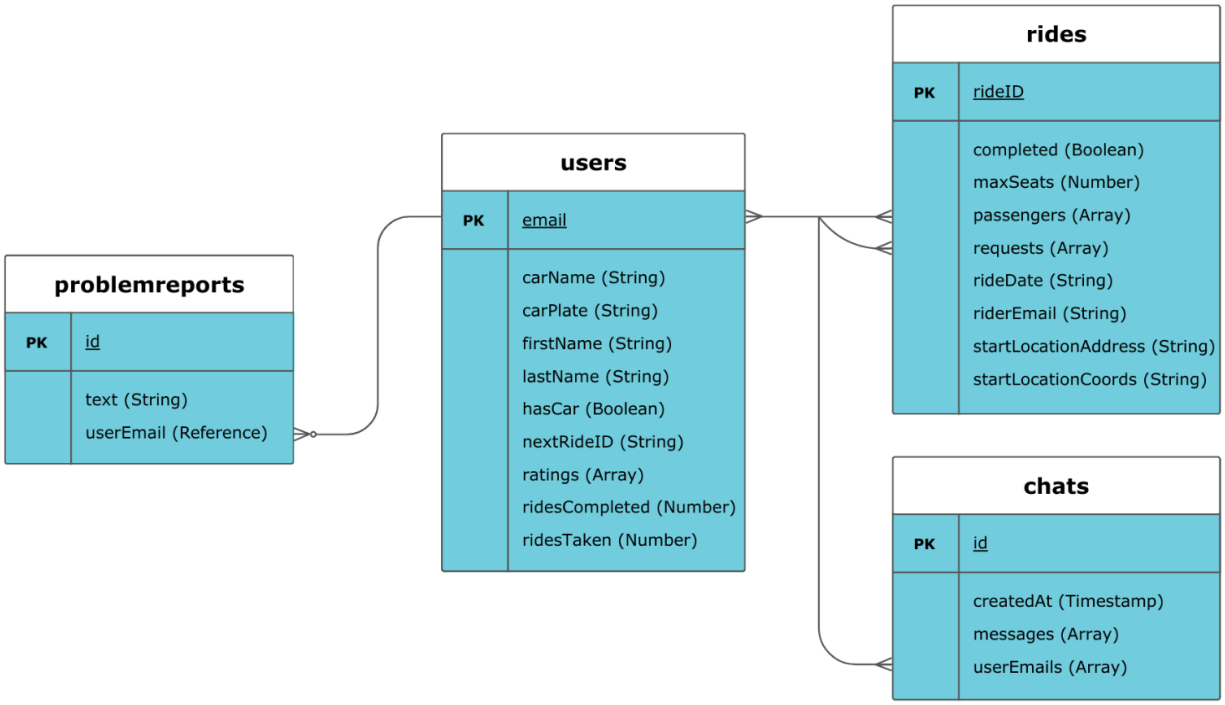
- reports
  Επιτρέπει την υποβολή αναφορών και παραπόνων από τους χρήστες μέσω της αντίστοιχης οθόνης της εφαρμογής, καταγράφοντας πληροφορίες που σχετίζονται με προβλήματα χρήσης ή συμπεριφοράς.

Κάθε collection αποτελείται από document IDs που δημιουργούνται αυτόματα από το Firestore, ενώ τα πεδία των documents είναι τυπικά σε μορφή key-value. Η επιλογή ανεξάρτητων collections αντί για nested δομές εξασφαλίζει:

- ευκολία αναζήτησης (queries),
- χαμηλή πολυπλοκότητα,
- ευκολία επεκτασιμότητας,
- ανεξαρτησία μεταξύ των λειτουργικών υπομονάδων της εφαρμογής.

Η πλήρης ανάλυση του schema, συμπεριλαμβανομένης της δομής κάθε collection, των πεδίων, των τύπων δεδομένων και των παραδειγμάτων εγγράφων, παρουσιάζεται στις επόμενες υποενότητες, με βάση το πραγματικό schema της υλοποίησης.

3.5.3 Αναλυτική Δομή Firestore (Firestore Schema)



Σχήμα 3.3: Αναλυτική Δομή Firestore.

3.6 Ζητήματα Ασφαλείας και Προστασίας Προσωπικών Δεδομένων (GDPR)

Η ασφάλεια των δεδομένων και η συμμόρφωση με τον Γενικό Κανονισμό Προστασίας Δεδομένων (GDPR) αποτελούν κρίσιμες παραμέτρους για την ανάπτυξη της εφαρμογής carpooling, δεδομένου ότι διαχειρίζεται προσωπικά στοιχεία χρηστών, όπως στοιχεία ταυτότητας, σημεία εκκίνησης/προορισμού και δεδομένα επικοινωνίας. Η αρχιτεκτονική της εφαρμογής έχει σχεδιαστεί με τρόπο που να περιορίζει την πρόσβαση σε ευαίσθητες πληροφορίες και να διασφαλίζει ότι κάθε λειτουργία συμμορφώνεται με τις σχετικές απαιτήσεις.

3.6.1 Ασφαλής πιστοποίηση χρηστών

Η είσοδος στο σύστημα πραγματοποιείται μέσω του ιδρυματικού συστήματος IHU Apps (Single Sign-On), το οποίο εξασφαλίζει ότι η εφαρμογή δεν αποθηκεύει κωδικούς πρόσβασης ούτε διαχειρίζεται διαδικασίες ταυτοποίησης. Η αυθεντικοποίηση ολοκληρώνεται εξ ολοκλήρου στο περιβάλλον του Ιδρύματος, και η εφαρμογή λαμβάνει μόνο τα απαραίτητα στοιχεία ταυτότητας (όπως πανεπιστημιακό email, ονοματεπώνυμο και μοναδικό αναγνωριστικό χρήστη), χωρίς πρόσβαση σε ευαίσθητα διαπιστευτήρια.

Με αυτόν τον τρόπο εφαρμόζονται οι αρχές του GDPR σχετικά με την ελαχιστοποίηση δεδομένων, ενώ η ευθύνη για την ασφάλεια των λογαριασμών παραμένει στο επίσημο σύστημα του Ιδρύματος.

3.6.2 Κανόνες ασφαλείας Firestore

Η αποθήκευση και ανάκτηση δεδομένων προστατεύεται μέσω Firestore Security Rules, που περιορίζουν την πρόσβαση μόνο σε πιστοποιημένους χρήστες και μόνο στα δεδομένα που τους αφορούν. Οι κανόνες επιτρέπουν ανάγνωση/εγγραφή αποκλειστικά στα αντίστοιχα documents, μειώνοντας τον κίνδυνο διαρροής δεδομένων μεταξύ χρηστών.

3.6.3 Ελαχιστοποίηση και περιορισμός δεδομένων

Η εφαρμογή συλλέγει μόνο τα απαραίτητα δεδομένα για τη λειτουργία της (όνομα, στοιχεία οχήματος, διαδρομές, μηνύματα), σύμφωνα με την αρχή της ελαχιστοποίησης δεδομένων του GDPR. Δεν αποθηκεύονται δεδομένα τοποθεσίας σε πραγματικό χρόνο ούτε στοιχεία υψηλής ευαισθησίας.

3.6.4 Δικαίωμα διαγραφής (Right to Erasure)

Παρέχεται δυνατότητα οριστικής διαγραφής λογαριασμού μέσα από το μενού ρυθμίσεων. Η διαγραφή ενεργοποιεί διαδικασία πλήρους απομάκρυνσης του χρήστη και των συνδεδεμένων δεδομένων από το Firestore, καλύπτοντας τις απαιτήσεις του άρθρου 17 του GDPR.

3.6.5 Επικοινωνία & ανταλλαγή μηνυμάτων

Το σύστημα chat χρησιμοποιεί Firestore documents, χωρίς εξωτερικές υπηρεσίες μεταφοράς δεδομένων. Τα μηνύματα παραμένουν ορατά μόνο στους συμμετέχοντες της διαδρομής, με αυστηρούς κανόνες πρόσβασης.

3.6.6 Συνολική αξιολόγηση ασφάλειας

Η συνδυασμένη χρήση:

- Ιδρυματικής πιστοποίησης (IHU SSO)
- Firebase Authentication
- Firestore Security Rules
- Ελαχιστοποίησης και καθαρού διαχωρισμού συλλογών

εξασφαλίζει ότι η εφαρμογή καλύπτει τα βασικά σημεία του GDPR και παρέχει υψηλό επίπεδο προστασίας σε κάθε στάδιο της χρήσης.

3.7 Επίλογος

Στο Κεφάλαιο 3 παρουσιάστηκε η αρχιτεκτονική της εφαρμογής carpooling, η οποία βασίζεται σε έναν καθαρό διαχωρισμό ανάμεσα στο frontend (Flutter), την αυθεντικοποίηση μέσω IHU Apps και Firebase Authentication, και το επίπεδο δεδομένων που υλοποιείται με Firebase Firestore. Αναλύθηκαν επίσης η δομή του έργου και οι τεχνολογικές επιλογές που υποστηρίζουν την επεκτασιμότητα, την απλότητα συντήρησης και την ομαλή ανάπτυξη.

Η ενότητα UI/UX περιέγραψε τις βασικές αρχές σχεδίασης, τη χρωματική παλέτα, την πλοήγηση και τα κύρια στοιχεία διεπαφής, που στόχο έχουν μια ευχάριστη και λειτουργική εμπειρία χρήστη. Παρουσιάστηκαν επίσης οι βασικές αρχές αποθήκευσης δεδομένων στο Firestore και τα συνοπτικά ζητήματα ασφάλειας, με έμφαση στη χρήση της ιδρυματικής ταυτοποίησης και στη μη αποθήκευση ευαίσθητων προσωπικών δεδομένων.

Συνολικά, το Κεφάλαιο 3 θέτει το θεωρητικό και τεχνικό υπόβαθρο πάνω στο οποίο βασίζεται η υλοποίηση που ακολουθεί στο Κεφάλαιο 4.

Κεφάλαιο 4ο: Υλοποίηση της Εφαρμογής

Στο παρόν κεφάλαιο παρουσιάζεται η υλοποίηση της εφαρμογής carpooling, με έμφαση στη λειτουργική οργάνωση των επιμέρους μονάδων και στη διασύνδεσή τους με την υποδομή backend. Αρχικά αναλύονται τα βασικά modules της εφαρμογής, όπως η αυθεντικοποίηση χρηστών, η διαχείριση διαδρομών και οχημάτων, η ανταλλαγή μηνυμάτων και η υποβολή αναφορών. Στη συνέχεια παρουσιάζονται οι μηχανισμοί επικοινωνίας με το Firebase και τα APIs, οι συλλογές δεδομένων και οι real-time λειτουργίες, οι οποίες επιτρέπουν την άμεση ενημέρωση των χρηστών. Τέλος, παρατίθενται ενδεικτικά αποσπάσματα κώδικα που αναδεικνύουν κρίσιμα σημεία της υλοποίησης και τεκμηριώνουν τον τρόπο με τον οποίο οι επιλεγμένες τεχνολογίες υποστηρίζουν τον σχεδιασμό της εφαρμογής.

4.1 Περιγραφή βασικών modules

Η υλοποίηση της εφαρμογής οργανώθηκε σε επιμέρους λειτουργικές ενότητες (modules), καθεμία από τις οποίες καλύπτει διαφορετική πτυχή της εμπειρίας χρήσης. Ο διαχωρισμός αυτός επιτρέπει τη σαφή κατανομή ευθυνών, την ευκολότερη συντήρηση του κώδικα και την ανεξάρτητη εξέλιξη των επιμέρους υποσυστημάτων. Τα modules περιλαμβάνουν την αυθεντικοποίηση των χρηστών, τη διαχείριση διαδρομών και οχημάτων, την ανταλλαγή μηνυμάτων μεταξύ συμμετεχόντων και την υποβολή αναφορών/διαχείριση ρυθμίσεων. Στις υποενότητες που ακολουθούν παρουσιάζεται η λειτουργικότητα κάθε module, η ροή αλληλεπίδρασης με τον χρήστη και ο τρόπος με τον οποίο τα δεδομένα αποθηκεύονται και ενημερώνονται μέσω του Firestore.

4.1.1 Διαδικασία πιστοποίησης χρηστών (Authentication)

Η διαδικασία αυθεντικοποίησης υλοποιείται κατά την πρώτη εκκίνηση της εφαρμογής και βασίζεται στην υπηρεσία εισόδου IHU Apps. Η εφαρμογή χρησιμοποιεί την OAuth 2.0 ροή εξουσιοδότησης μέσω της βιβλιοθήκης flutter_appauth, η οποία ανακατευθύνει τον χρήστη στον επίσημο μηχανισμό ταυτοποίησης του ΔΙΠΑΕ. Μετά την επιτυχή εισαγωγή των διαπιστευτηρίων του χρήστη, η υπηρεσία επιστρέφει access token και βασικά στοιχεία προφίλ, τα οποία χρησιμοποιούνται για τη συνέχιση της λειτουργίας.

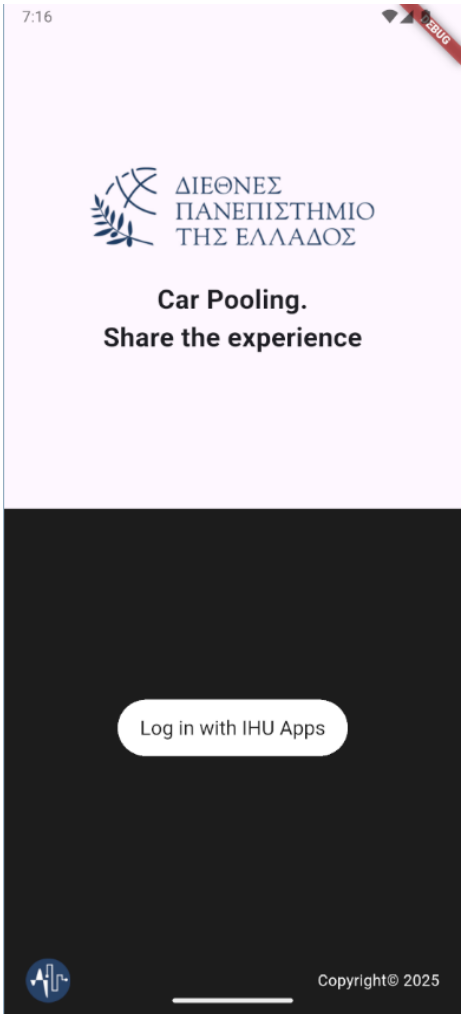
Στο backend, η εφαρμογή διασταυρώνει το email του χρήστη με την εγγραφή του στο Firestore. Εάν δεν υπάρχει ήδη αντίστοιχο έγγραφο στη συλλογή /users, δημιουργείται νέα καταχώρηση με τα ακόλουθα πεδία:

- email: το ιδρυματικό email του χρήστη
- fullName: το όνομα που επέλεξε ο χρήστης κατά την πρώτη είσοδο
- hasCar: δηλωμένη κατοχή οχήματος (true/false)
- vehicleModel, vehiclePlate: στοιχεία του οχήματος (αν υπάρχουν)
- ridesCompleted, ridesTaken: μετρητές δραστηριότητας

Η εφαρμογή αποθηκεύει το UID τοπικά (π.χ. μέσω SharedPreferences) ώστε να αποφεύγεται η επαναλαμβανόμενη αυθεντικοποίηση σε μελλοντικές εκκινήσεις. Σε περίπτωση επανασύνδεσης, τα δεδομένα χρήστη ανακτώνται απευθείας από το Firestore και ενημερώνουν την κατάσταση της εφαρμογής, χωρίς να απαιτείται νέο login.

Η παραπάνω διαδικασία διασφαλίζει ότι κάθε χρήστης είναι μοναδικά ταυτοποιημένος, αποτρέποντας την ανώνυμη χρήση και επιτρέποντας τον έλεγχο πρόσβασης σε λειτουργίες όπως η δημιουργία διαδρομών ή η αποστολή αιτημάτων συμμετοχής. Παράλληλα, η χρήση ενός ενιαίου λογαριασμού ανά

προφίλ διευκολύνει την παρακολούθηση της ιστορικότητας των διαδρομών και της συμπεριφοράς των συμμετεχόντων.



Σχήμα 4.1: Οθόνη εισόδου και αυθεντικοποίησης.

Σε επίπεδο διεπαφής, η οθόνη αυθεντικοποίησης εμφανίζει το κουμπί “Log in with IHU Apps”, το οποίο ενεργοποιεί την OAuth ροή. Με την επιβεβαίωση, ο χρήστης μεταφέρεται αυτόματα στο περιβάλλον της εφαρμογής, ενώ σε περίπτωση πρώτης εισόδου οδηγείται στην αρχική φόρμα διαμόρφωσης προφίλ.

4.1.2 Διαχείριση διαδρομών και οχημάτων

Η λειτουργία διαδρομών αποτελεί τον πυρήνα της εφαρμογής. Κάθε χρήστης έχει τη δυνατότητα να αναζητά υπάρχουσες διαδρομές και να συμμετέχει σε αυτές, ενώ όσοι διαθέτουν όχημα μπορούν να δημιουργούν και να διαχειρίζονται δικές τους rides.

Η εφαρμογή δεν διαχωρίζει ρόλους οδηγού-επιβάτη, ο ρόλος προκύπτει από τη δημιουργία ή όχι διαδρομής.

Όλα τα rides αποθηκεύονται στη συλλογή /rides του Firebase Firestore ως ανεξάρτητα έγγραφα.

Κάθε document περιλαμβάνει τα ακόλουθα πεδία:

- rideID: μοναδικό αναγνωριστικό της διαδρομής,
- startLocationCoords: συντεταγμένες σημείου εκκίνησης στον χάρτη,

- startLocationAddress: διεύθυνση σημείου εκκίνησης,
- rideDate: ημερομηνία και ώρα αναχώρησης,
- riderEmail: email του οδηγού,
- maxSeats: μέγιστος αριθμός διαθέσιμων θέσεων,
- passengers[]: πίνακας με emails επιβαινόντων,
- requests[]: πίνακας με emails χρηστών που έχουν αιτηθεί συμμετοχή,
- completed: λογική μεταβλητή που δηλώνει αν η διαδρομή έχει ολοκληρωθεί.

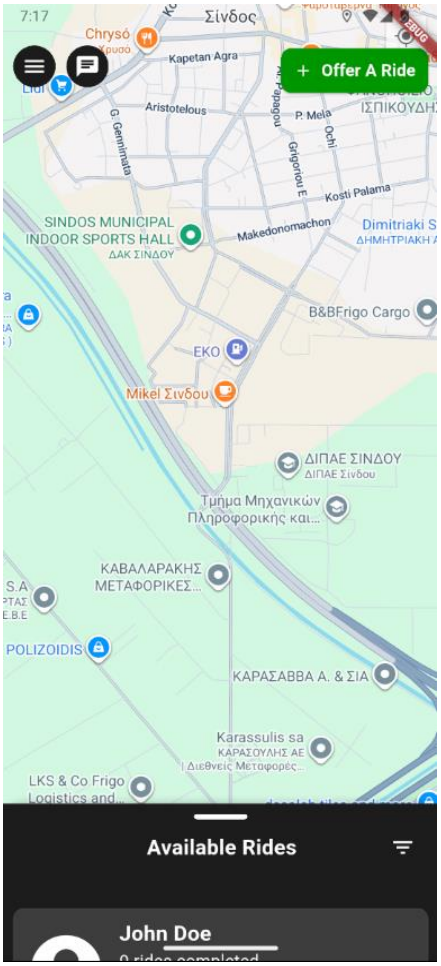
4.1.2.1 Δημιουργία διαδρομής από οδηγό (Offer a Ride)

Η αρχική οθόνη της εφαρμογής βασίζεται στον χάρτη και παρέχει το κουμπί “Offer a Ride”, μέσω του οποίου ο χρήστης δημιουργεί νέα διαδρομή.

Ο οδηγός επιλέγει το σημείο εκκίνησης πάνω στον χάρτη. Η εφαρμογή λαμβάνει:

- τις γεωγραφικές συντεταγμένες (latitude/longitude),
- τη μορφοποιημένη διεύθυνση.

Τα δεδομένα εισάγονται στο document της νέας διαδρομής στο Firestore.



Σχήμα 4.2: Οθόνη χάρτη με το κουμπί δημιουργίας διαδρομής.

Στη συνέχεια ο οδηγός καθορίζει:

- ημερομηνία/ώρα εκκίνησης,
- αριθμό θέσεων (maxSeats).

Με την επιβεβαίωση, η εφαρμογή δημιουργεί νέο document στο /rides, το οποίο εμφανίζεται άμεσα σε όλους τους χρήστες.

4.1.2.2 Αναζήτηση και εμφάνιση διαθέσιμων διαδρομών

Οι επιβάτες μπορούν να δουν διαθέσιμες διαδρομές μέσα από την ενότητα Available Rides, η οποία εμφανίζεται κάτω από τον χάρτη.

Η λίστα περιλαμβάνει rides που:

- έχουν ημερομηνία μελλοντική,
- διαθέτουν ελεύθερες θέσεις.

Η ταξινόμηση γίνεται βάσει του πεδίου rideDate, ενώ διαδρομές που έχουν ολοκληρωθεί (completed = true) δεν εμφανίζονται.

Κάθε ride στη λίστα παρουσιάζει:

- το όνομα του οδηγού,
- τον τόπο εκκίνησης,
- τον χρόνο αναχώρησης,
- την τρέχουσα κατάσταση συμμετοχών.

4.1.2.3 Αίτημα συμμετοχής

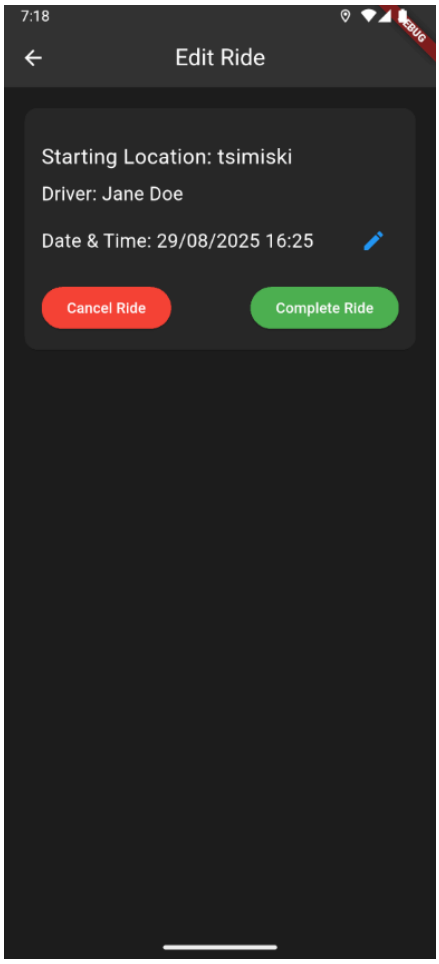
Η συμμετοχή σε μια διαδρομή δεν είναι άμεση. Ο χρήστης υποβάλλει αίτημα (request) και προστίθεται στο πεδίο requests[] του αντίστοιχου document. Ο οδηγός βλέπει τα requests και διαχειρίζεται τις εγκρίσεις από την οθόνη προφίλ/διαδρομών του.

4.1.2.4 Επεξεργασία και ολοκλήρωση διαδρομών

Ο οδηγός μπορεί να επεξεργαστεί τη διαδρομή του μέσα από την οθόνη Edit Ride, όπου τροποποιεί:

- ημερομηνία,
- ώρα αναχώρησης.

Η αλλαγή του σημείου εκκίνησης δεν υποστηρίζεται. Αν απαιτείται διαφορετική τοποθεσία, ο οδηγός πρέπει να δημιουργήσει νέο ride για να μην επηρεαστούν οι επιβάτες.



Σχήμα 4.3: Οθόνη επεξεργασίας διαδρομής και ολοκλήρωσης ride.

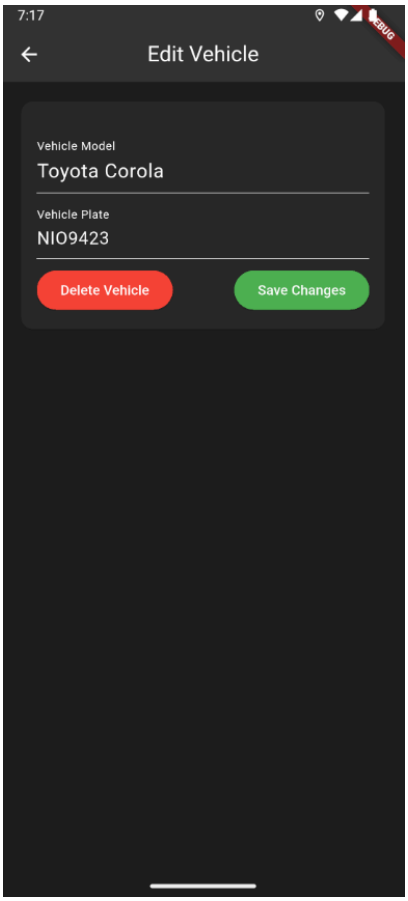
Επιπλέον, ο οδηγός μπορεί:

- να ακυρώσει τη διαδρομή (Cancel Ride),
- να τη μαρκάρει ως ολοκληρωμένη (Complete Ride).

Με την ολοκλήρωση, το πεδίο completed = true και η διαδρομή αφαιρείται από τα διαθέσιμα rides.

4.1.2.5 Διαχείριση οχήματος

Τα στοιχεία οχήματος αποθηκεύονται στο document του χρήστη στη συλλογή /users. Κατά την πρώτη είσοδο, ο χρήστης εισάγει το μοντέλο και την πινακίδα του αυτοκινήτου του, αν διαθέτει. Αργότερα μπορεί να τροποποιήσει ή να διαγράψει το όχημα μέσω της οθόνης Edit Vehicle. Οι αλλαγές ενημερώνουν το Firestore σε πραγματικό χρόνο.



Σχήμα 4.4: Οθόνη επεξεργασίας στοιχείων οχήματος.

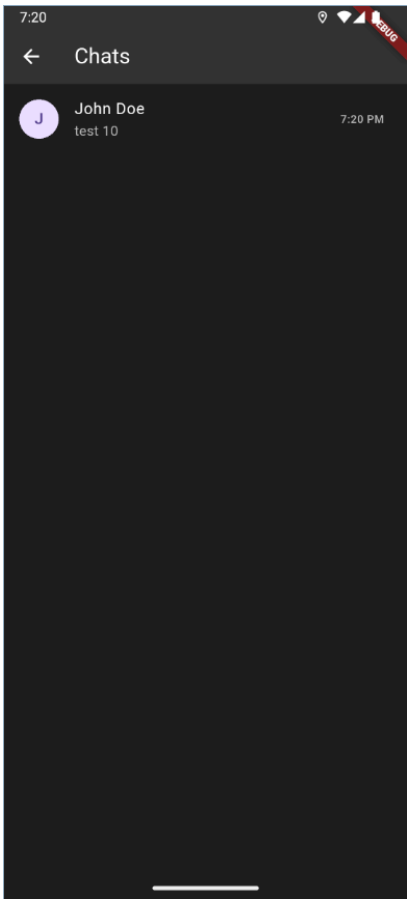
Εάν ο χρήστης δεν διαθέτει όχημα, η επιλογή δημιουργίας νέας διαδρομής παραμένει κρυφή, περιορίζοντας τις ενέργειες του σε συμμετοχή σε rides άλλων χρηστών.

4.1.3 Επικοινωνία χρηστών (chat)

Η εφαρμογή υποστηρίζει σύστημα επικοινωνίας σε πραγματικό χρόνο μεταξύ των χρηστών που συμμετέχουν σε κοινές διαδρομές. Η λειτουργία της συνομιλίας παρέχει έναν ασφαλή και ελεγχόμενο τρόπο ανταλλαγής πληροφοριών μεταξύ οδηγού και επιβατών, επιτρέποντας συνεννόηση πριν ή μετά τη μετακίνηση σχετικά με ώρα συνάντησης, σημείο επιβίβασης ή πιθανές αλλαγές στο πρόγραμμα. Η επικοινωνία αυτή είναι αυστηρά one-to-one και δεν υποστηρίζονται ομαδικές συνομιλίες.

Η υλοποίηση του υποσυστήματος βασίζεται στο Firebase Firestore, το οποίο παρέχει real-time ενημερώσεις σε χαμηλό latency. Για κάθε ζεύγος χρηστών δημιουργείται ένα μοναδικό thread συνομιλίας, στο οποίο καταχωρούνται τα μηνύματα ως εγγραφές με τις αντίστοιχες μεταδεδομένες πληροφορίες (περιεχόμενο, αποστολέας, timestamp). Η αποθήκευση timestamp διευκολύνει την ταξινόμηση των μηνυμάτων κατά χρονολογική σειρά, επιτρέποντας την ανάκτηση και ομαλή εμφάνιση ιστορικού κατά την είσοδο του χρήστη στην οθόνη συνομιλίας.

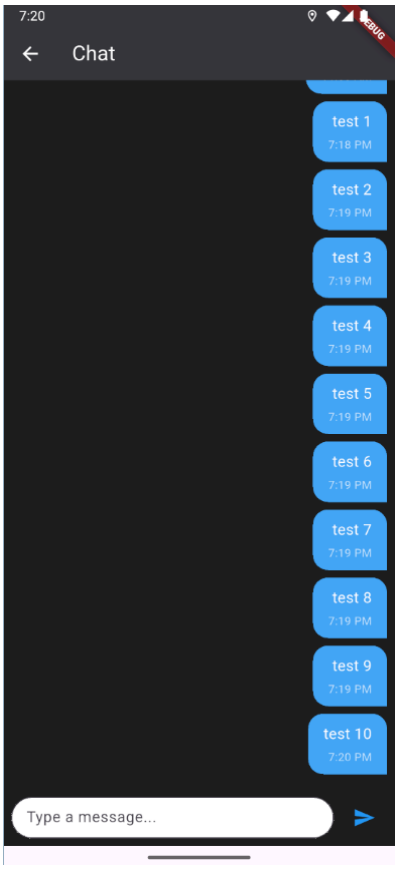
Για την προβολή των συνομιλιών, η εφαρμογή διαθέτει οθόνη επισκόπησης, όπου κάθε συνομιλία εμφανίζεται ως ξεχωριστό entry μαζί με το όνομα του έτερου συμμετέχοντα και προεπισκόπηση του τελευταίου μηνύματος. Η παρουσίαση περιορίζεται αποκλειστικά σε threads στα οποία ο χρήστης συμμετέχει, αποτρέποντας την πρόσβαση σε συνομιλίες τρίτων και διασφαλίζοντας τη διακριτικότητα των αλληλεπιδράσεων.



Σχήμα 4.5: Οθόνη λίστας συνομιλιών.

Η οθόνη ενεργής συνομιλίας υλοποιείται μέσω διεπαφής τύπου bubble messaging. Τα εισερχόμενα και εξερχόμενα μηνύματα διαφοροποιούνται χρωματικά για ευκολότερη αναγνώριση, ενώ διαθέτουν ευθυγράμμιση ανάλογα με τον αποστολέα. Η εισαγωγή νέου μηνύματος γίνεται μέσω πεδίου κειμένου στο κάτω μέρος της οθόνης, με το μήνυμα να δημοσιεύεται στο Firestore και να αποστέλλεται σε πραγματικό χρόνο στον συνομιλούντα χρήστη.

Η δημιουργία συνομιλίας γίνεται δυναμικά σύμφωνα με την αρχή lazy initialization. Δεν παράγονται προληπτικά threads κατά την αποδοχή αιτήματος συμμετοχής σε διαδρομή, το thread δημιουργείται μόνο κατά την αποστολή του πρώτου μηνύματος από έναν εκ των δύο χρηστών. Η πρακτική αυτή μειώνει τον αριθμό αχρησιμοποίητων εγγραφών στη βάση και αντανακλά ρεαλιστικά τις πραγματικές αλληλεπιδράσεις μετακίνησης.



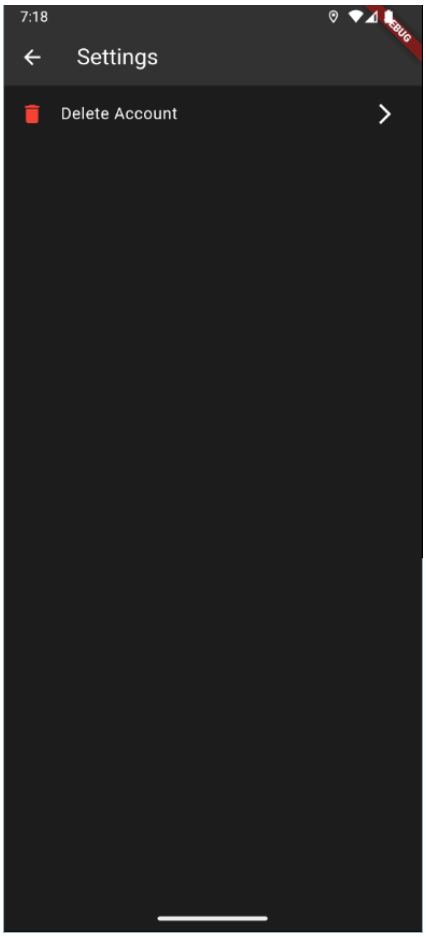
Σχήμα 4.6: Οθόνη ενεργής συνομιλίας.

Στον σχεδιασμό του chat εφαρμόζεται επίσης βασική ιδιωτικότητα. Οι χρήστες δεν μοιράζονται αριθμό τηλεφώνου ή άλλα προσωπικά στοιχεία εντός της εφαρμογής, όλα τα μηνύματα ανταλλάσσονται μέσω του συστήματος Firebase, χωρίς πρόσβαση τρίτων. Η προσέγγιση αυτή είναι συμβατή με τις αρχές προστασίας δεδομένων που ακολουθεί η εφαρμογή, καθώς περιορίζει την έκθεση προσωπικών πληροφοριών και αποτρέπει ανεπιθύμητες επικοινωνίες εκτός του περιβάλλοντος της εφαρμογής.

4.1.4 Αναφορές προβλημάτων & ρυθμίσεις

Η εφαρμογή ενσωματώνει βασικές λειτουργίες υποστήριξης χρήστη, οι οποίες συγκεντρώνονται στο τμήμα “Settings”. Σκοπός της ενότητας αυτής είναι η παροχή εργαλείων διαχείρισης λογαριασμού και επικοινωνίας με την ομάδα ανάπτυξης, χωρίς να επηρεάζεται η λειτουργικότητα των κύριων υποσυστημάτων της εφαρμογής. Το μενού ρυθμίσεων διατηρείται σκόπιμα απλό, ώστε ο χρήστης να μην αποσπάται από τη διαδικασία μετακίνησης και διαχείρισης διαδρομών.

Η επιλογή “Delete Account” επιτρέπει στον χρήστη να διαγράψει τον λογαριασμό του από την εφαρμογή και να απομακρύνει τα στοιχεία που έχουν καταχωρηθεί στη βάση δεδομένων. Η διαγραφή αφορά αποκλειστικά τις πληροφορίες που έχουν δημιουργηθεί από τον φοιτητή μέσω της εφαρμογής (όπως στοιχεία οχήματος, ιστορικό διαδρομών και εγγραφές συνομιλιών) και όχι τα στοιχεία ταυτοποίησης μέσω IHU Apps, καθώς αυτά ανήκουν στο σύστημα πιστοποίησης του ιδρύματος. Η λειτουργία αυτή παρέχει ένα επιπλέον επίπεδο ελέγχου των προσωπικών δεδομένων και συμμορφώνεται με τις αρχές προστασίας ιδιωτικότητας που έχουν παρουσιαστεί στα προηγούμενα κεφάλαια.

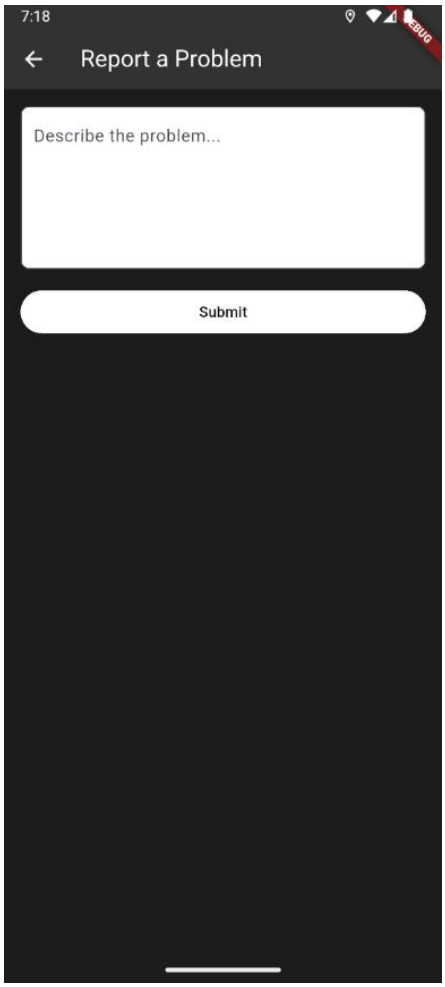


Σχήμα 4.7: Οθόνη ρυθμίσεων με την επιλογή “Delete Account”.

Η εφαρμογή διαθέτει επίσης λειτουργία “Report a Problem”, η οποία επιτρέπει την υποβολή αναφορών προβλημάτων ή προτάσεων βελτίωσης. Η οθόνη αναφοράς περιλαμβάνει πεδίο κειμένου στο οποίο ο χρήστης περιγράφει το ζήτημα που αντιμετώπισε ή την παρατήρησή του. Οι αναφορές αποστέλλονται αποθηκευμένες στο backend, παρέχοντας στους υπεύθυνους ανάπτυξης μια οργανωμένη βάση feedback για μελλοντικές διορθώσεις ή επεκτάσεις. Η λειτουργία αυτή κρίνεται ιδιαίτερα χρήσιμη κατά τη διάρκεια της πειραματικής και ακαδημαϊκής φάσης της εφαρμογής, όπου ο ρυθμός εμφάνισης νέων σεναρίων χρήσης και edge cases είναι αυξημένος.

Σχεδιαστικά, οι δύο λειτουργίες τοποθετούνται σε ξεχωριστές οθόνες ώστε να αποφευχθεί η υπερφόρτωση του χρήστη με πληροφορίες και να μειωθεί η πιθανότητα λανθασμένης ενεργοποίησης κρίσιμων ενεργειών, όπως η διαγραφή λογαριασμού. Η διαδρομή αλληλεπίδρασης ακολουθεί την αρχή «μια ενέργεια ανά οθόνη», πρακτική που μειώνει τα γνωστικά λάθη και βελτιώνει τη χρηστικότητα σε εφαρμογές που χρησιμοποιούνται κατά την καθημερινή μετακίνηση.

Αξίζει να σημειωθεί ότι και οι δύο λειτουργίες περιορίζονται αποκλειστικά στο περιβάλλον της εφαρμογής και δεν διασυνδέονται με υπηρεσίες τρίτων. Δεν υφίσταται αυτοματοποιημένο email, push notification ή εξωτερικό ticketing σύστημα. Η επιλογή αυτή συμβαδίζει με τη φιλοσοφία του πρωτοτύπου, το οποίο εστιάζει στη λειτουργικότητα της διαμοιραζόμενης μετακίνησης και όχι σε πλήρες οικοσύστημα υποστήριξης χρηστών.

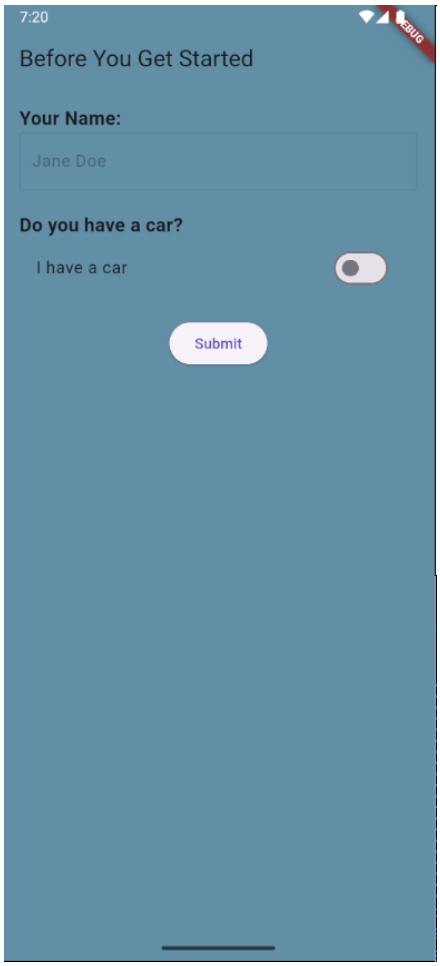


Σχήμα 4.8: Οθόνη υποβολής αναφοράς προβλήματος με πεδίο εισαγωγής και επιλογή “Submit”.

4.1.5 Αρχική διαμόρφωση χρήστη (Onboarding)

Η διαδικασία αρχικής διαμόρφωσης χρήστη (onboarding) αποτελεί το πρώτο σημείο αλληλεπίδρασης του φοιτητή με την εφαρμογή μετά τη σύνδεσή του μέσω του συστήματος IHU Apps. Καθώς τα στοιχεία ταυτοποίησης (ονοματεπώνυμο και ηλεκτρονική διεύθυνση) μεταφέρονται αυτόματα από τον πάροχο αυθεντικοποίησης, η εφαρμογή αποφεύγει τη δημιουργία πρόσθετων λογαριασμών και δεν απαιτεί κωδικό πρόσβασης. Με αυτόν τον τρόπο, αφενός μειώνεται η τριβή κατά την είσοδο, αφετέρου διασφαλίζεται ότι όλοι οι χρήστες είναι ήδη επαληθευμένοι φοιτητές του ιδρύματος.

Η οθόνη onboarding έχει σχεδιαστεί ώστε να συγκεντρώνει τις απολύτως απαραίτητες πληροφορίες που επηρεάζουν τη λειτουργικότητα της εφαρμογής. Ο χρήστης καλείται να δηλώσει εάν διαθέτει αυτοκίνητο και, επομένως, αν μπορεί να αναρτά νέες διαδρομές. Η επιλογή αυτή καθορίζει δυναμικά τις επιτρεπόμενες ενέργειες στην εφαρμογή: όσοι δεν διαθέτουν όχημα μπορούν μόνο να αιτηθούν συμμετοχή σε διαδρομές, ενώ όσοι διαθέτουν όχημα αποκτούν τη δυνατότητα δημιουργίας νέων rides. Η σχεδιαστική προσέγγιση είναι εσκεμμένα ελάχιστη, ώστε να επιτρέψει την γρήγορη μετάβαση στον κεντρικό σκοπό της εφαρμογής, την αναζήτηση ή προσφορά μετακινήσεων.



Σχήμα 4.9: Οθόνη αρχικής διαμόρφωσης χρήστη και επιλογή δήλωσης οχήματος.

Η προσέγγιση αυτή ακολουθεί την αρχή της ελάχιστης απαίτησης δεδομένων (data minimization). Η εφαρμογή δεν συλλέγει επιπλέον προσωπικές πληροφορίες πέραν όσων είναι αναγκαίες για τη βασική λειτουργία της υπηρεσίας: το όνομα, το email και, μόνο αν απαιτείται, τα στοιχεία του οχήματος. Δεν υπάρχουν πολυσέλιδα ερωτηματολόγια, πρόσθετες ρυθμίσεις ή επιπλέον στάδια. Η διαδικασία onboarding ολοκληρώνεται σε ένα βήμα, ώστε ο χρήστης να αποκτήσει άμεσα πρόσβαση στις βασικές λειτουργίες της εφαρμογής.

Η επιλογή αυτής της σχεδιαστικής στρατηγικής έχει διττό αποτέλεσμα: μειώνει δραστικά την πιθανότητα εγκατάλειψης της διαδικασίας στην πρώτη χρήση και ταυτόχρονα προσφέρει μια καθαρή διαχωριστική γραμμή μεταξύ χρηστών που μπορούν να δημιουργούν διαδρομές και χρηστών που συμμετέχουν ως επιβάτες. Η απλότητα του onboarding αποτελεί συνειδητή επιλογή, αντανακλώντας τον πανεπιστημιακό χαρακτήρα της εφαρμογής και την ανάγκη γρήγορης πρόσβασης σε λειτουργικότητα χωρίς περιττά εμπόδια.

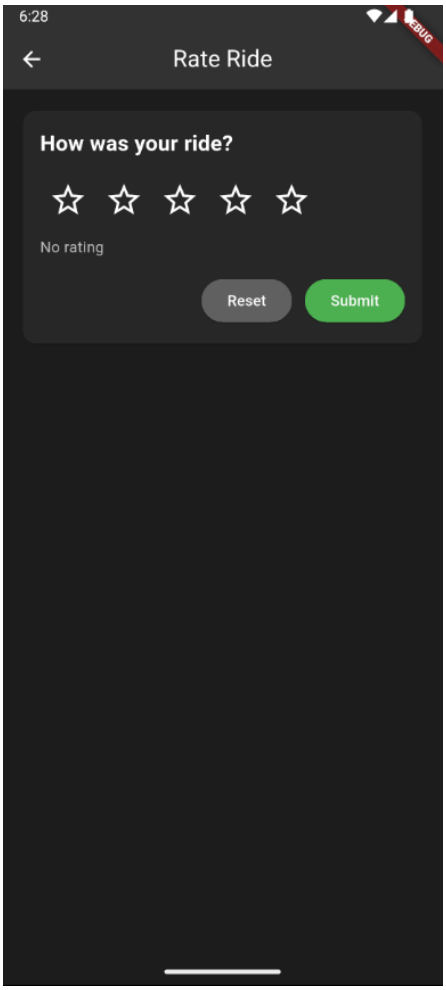
4.1.6 Αξιολόγηση διαδρομής (Rate Ride)

Η εφαρμογή παρέχει τη δυνατότητα στους χρήστες να αξιολογούν την εμπειρία τους μετά από κάθε διαδρομή μέσω της λειτουργίας Rate Ride. Αυτή η λειτουργία επιτρέπει στους φοιτητές να δώσουν βαθμολογία από 1 έως 5 αστέρια για την διαδρομή που συμμετείχαν, προσφέροντας ανατροφοδότηση στον οδηγό και στους υπόλοιπους χρήστες.

Η οθόνη αξιολόγησης εμφανίζεται μετά την ολοκλήρωση της διαδρομής και περιλαμβάνει τα ακόλουθα στοιχεία:

- Επιλογή αστέρων (1–5): Ο χρήστης μπορεί να επιλέξει τον αριθμό αστέρων που αντιπροσωπεύει την εμπειρία του.
- Κουμπί “Reset”: Επιτρέπει την επαναφορά της βαθμολογίας πριν την υποβολή.
- Κουμπί “Submit”: Υποβάλλει την αξιολόγηση και αποθηκεύει τη βαθμολογία στη βάση δεδομένων.

Η αξιολόγηση διαδρομών συμβάλλει στη διαφάνεια και στη βελτίωση της ποιότητας των υπηρεσιών carpooling μεταξύ των φοιτητών, καθώς επιτρέπει στους οδηγούς να λαμβάνουν χρήσιμα σχόλια και στους νέους χρήστες να έχουν εικόνα για την εμπειρία των προηγούμενων συμμετοχών.



Σχήμα 4.10: Οθόνη αξιολόγησης διαδρομής με επιλογή 1-5 αστεριών.

4.2 Διασύνδεση με Firebase και APIs

Η εφαρμογή αξιοποιεί το οικοσύστημα υπηρεσιών Firebase ως κεντρική υποδομή δεδομένων και αυθεντικοποίησης. Η επιλογή της πλατφόρμας βασίστηκε στην πλήρη ενσωμάτωσή της με κινητές εφαρμογές Flutter, στη δυνατότητα διαχείρισης πραγματικού χρόνου και στην απουσία ανάγκης ανάπτυξης ανεξάρτητου backend server. Η εφαρμογή αλληλεπιδρά απευθείας με το Firebase, χωρίς middleware ή REST endpoints, γεγονός που μειώνει την πολυπλοκότητα του συστήματος και επιταχύνει την ανάπτυξη.

4.2.1 Firebase Authentication (IHU Apps Sign-In)

Η διαδικασία σύνδεσης βασίζεται στην υπηρεσία Firebase Authentication, η οποία λειτουργεί ως gateway μεταξύ της εφαρμογής και του συστήματος IHU Apps. Ο χρήστης δεν δημιουργεί λογαριασμό με email/κωδικό, αλλά γίνεται απλή ανάθεση των credentials που παρέχονται από το ίδρυμα. Η μέθοδος αυτή εξασφαλίζει ότι όλοι οι χρήστες είναι πραγματικοί φοιτητές και έχουν ήδη ταυτοποιηθεί από το κεντρικό σύστημα, απαλλάσσοντας την εφαρμογή από διαδικασίες verification όπως email confirmation ή επανέλεγχο ταυτότητας.

Μετά τη σύνδεση, η εφαρμογή λαμβάνει το email και το πλήρες ονοματεπώνυμο και τα αποθηκεύει στο Firestore ώστε να υπάρχει ενοποιημένο χρήστη-προφίλ στο εσωτερικό της εφαρμογής. Δεν αποθηκεύονται κωδικοί ή tokens στο τοπικό περιβάλλον. Η αυθεντικοποίηση λειτουργεί ως single-source-of-truth, επιτρέποντας στον χρήστη απρόσκοπτη πρόσβαση χωρίς πρόσθετα credentials σε μεταγενέστερες χρήσεις.

4.2.2 Firestore Database (Real-time Cloud Storage)

Για την αποθήκευση των δεδομένων χρησιμοποιείται η υπηρεσία Firestore, η οποία προσφέρει cloud βάση εγγράφων (document-oriented). Η δομή της βάσης επιλέχθηκε ώστε να αντανακλά τη λειτουργικότητα της εφαρμογής. Τα βασικά μοντέλα δεδομένων περιλαμβάνουν:

- Users: στοιχεία προφίλ χρήστη, όνομα, email, hasCar, μοντέλο και πινακίδα οχήματος.
- Rides: διαθέσιμες διαδρομές, γεωγραφική θέση έναρξης, ημερομηνία/ώρα, χωρητικότητα, αιτήματα συμμετοχής, επιβάτες και κατάσταση ολοκλήρωσης.
- Chats: threads επικοινωνίας μεταξύ χρηστών, με μηνύματα σε χρονολογική σειρά.

Η υιοθέτηση document-based αρχιτεκτονικής μειώνει την ανάγκη για περίπλοκες σχέσεις και foreign keys, καθώς τα δεδομένα μπορούν να συγκεντρώνονται ανα χρήστη, ανα ride ή ανα chat thread. Επιπλέον, κάθε αλλαγή σε συλλογή Firestore παράγει real-time updates στην εφαρμογή μέσω listeners, κάτι που επιτρέπει άμεση ενημέρωση της διεπαφής, για παράδειγμα:

- εμφάνιση νέων rides,
- αποδοχή αιτημάτων επιβατών,
- λήψη νέων μηνυμάτων chat.

Η φύση real-time των listeners καθιστά περιττή την περιοδική ενημέρωση μέσω polling και εξαλείφει καθυστερήσεις στη διάδραση μεταξύ οδηγών και επιβατών.

4.2.3 Google Maps API

Για την απόδοση του χάρτη και την επιλογή σημείου εκκίνησης των διαδρομών χρησιμοποιείται το Google Maps API. Η διεπαφή επιτρέπει στον χρήστη να καθορίσει την τοποθεσία έναρξης είτε μέσω χειροκίνητης εισαγωγής, είτε πατώντας σε σημείο της χαρτογραφικής προβολής. Η API προσφέρει αυτόματο rendering των σημείων, offline caching χαρτογραφικών πλακιδίων και δυνατότητες εμφάνισης markers.

Στην τρέχουσα υλοποίηση δεν συλλέγεται η ζωντανή γεωγραφική θέση του χρήστη, τόσο για λόγους απλούστευσης όσο και για συμμόρφωση με τις απαιτήσεις ιδιωτικότητας. Η εφαρμογή καταγράφει μόνο τη διεύθυνση και τις συντεταγμένες που επιλέγει ο οδηγός για την εκκίνηση του ride, οι οποίες αποθηκεύονται στη βάση δεδομένων Firestore ως startLocationCoords και startLocationAddress.

4.2.4 Απουσία εξωτερικών APIs

Παρά το γεγονός ότι η εφαρμογή βασίζεται σε cloud υπηρεσίες, δεν χρησιμοποιεί τρίτες APIs πέραν της πλατφόρμας Firebase και της χαρτογραφικής υποδομής Google Maps. Δεν υπάρχουν REST endpoints, middleware services ή server-side micro-services. Η επιλογή αυτή είναι συνειδητή, καθώς η εφαρμογή υλοποιείται στο πλαίσιο ακαδημαϊκού project και δεν στοχεύει σε πλήρη παραγωγική διάθεση. Με τον τρόπο αυτό μειώνεται η πολυπλοκότητα της αρχιτεκτονικής και ελαχιστοποιούνται τα πιθανά σημεία αστοχίας.

4.2.5 Ασφάλεια και περιορισμοί

Η απουσία συστήματος παραγωγικής διάθεσης συνεπάγεται ότι δεν έχουν εφαρμοστεί εκτεταμένοι μηχανισμοί ασφαλείας, όπως rule-based access policies, περιορισμοί εγγραφής σε συλλογές ή user-scoped permissions. Τα δεδομένα είναι προσβάσιμα μέσω της εφαρμογής, αλλά όχι δημόσια, ενώ η σύνδεση μέσω IHU Apps λειτουργεί ως βασικός φραγμός μη εξουσιοδοτημένων χρηστών. Σε περιβάλλον παραγωγής απαιτείται αυστηρή πολιτική Firestore Security Rules, καθώς και μηχανισμοί logging, auditing και ιεραρχικής πρόσβασης.

4.3 Επίλογος

Στο Κεφάλαιο 4 παρουσιάστηκαν οι βασικές λειτουργικές και τεχνικές πτυχές της εφαρμογής, με έμφαση στην υλοποίηση των κύριων ενοτήτων, τη δομή των διεπαφών χρήστη και τη διασύνδεση με το backend. Οι επιμέρους οθόνες υλοποιούν διακριτές λειτουργίες, όπως η δημιουργία και αναζήτηση διαδρομών, η διαχείριση προφίλ και οχήματος, η ανταλλαγή μηνυμάτων μεταξύ χρηστών και η υποβολή αναφορών προβλημάτων. Η σχεδιαστική προσέγγιση βασίστηκε στην αρχή της λειτουργικής διάκρισης, με στόχο τη μείωση της γνωστικής επιβάρυνσης και τη βελτίωση της εμπειρίας χρήστη. Ενδεικτικά αποσπάσματα κώδικα παρατίθενται στα Παραρτήματα της εργασίας, όπου στο Παράρτημα Α παρουσιάζεται ο κώδικας δημιουργίας χρήστη, στο Παράρτημα C η υλοποίηση επεξεργασίας οχήματος και στο Παράρτημα Ε το σύστημα συνομιλιών.

Σε τεχνικό επίπεδο, η εφαρμογή αξιοποιεί το οικοσύστημα Firebase ως υποδομή αποθήκευσης και διαχείρισης δεδομένων, σε συνδυασμό με ιδρυματική αυθεντικοποίηση μέσω της υπηρεσίας IHU Apps. Η ταυτοποίηση πραγματοποιείται αποκλειστικά μέσω του μηχανισμού Single Sign-On, εξαλείφοντας την ανάγκη διαχείρισης κωδικών εντός της εφαρμογής. Παράλληλα, η χρήση της Firestore ως document-based βάσης δεδομένων επιτρέπει την άμεση και ευέλικτη διαχείριση δεδομένων που αφορούν χρήστες, διαδρομές και συνομιλίες, ενώ οι real-time ενημερώσεις εξασφαλίζουν δυναμική ανανέωση της διεπαφής. Η ενσωμάτωση του Google Maps SDK υποστηρίζει την οπτικοποίηση γεωγραφικών δεδομένων και τη διαχείριση των σημείων εκκίνησης των διαδρομών.

Η αρχική διαμόρφωση του χρήστη έχει σχεδιαστεί με γνώμονα την απλότητα και την ταχεία πρόσβαση στη λειτουργικότητα της εφαρμογής, απαιτώντας μόνο τις απολύτως απαραίτητες πληροφορίες για τον καθορισμό της ιδιότητας του χρήστη ως οδηγού ή επιβάτη. Με τον τρόπο αυτό επιτυγχάνεται ομαλή έναρξη χρήσης χωρίς περιττά βήματα.

Συνολικά, το κεφάλαιο ανέδειξε μια υλοποίηση που συνδυάζει απλότητα, λειτουργική σαφήνεια και αξιοποίηση σύγχρονων cloud τεχνολογιών. Η απουσία πολύπλοκων ρόλων και server-side υποδομών αντανακλά τον ακαδημαϊκό χαρακτήρα του έργου, διατηρώντας παράλληλα δυνατότητες μελλοντικής επέκτασης, όπως προηγμένα συστήματα ειδοποιήσεων, μηχανισμούς σύζευξης οδηγών-επιβατών και συστήματα αξιολόγησης χρηστών.

Κεφάλαιο 5ο: Αποτελέσματα και Αξιολόγηση

5.1 Περιγραφή της τελικής εφαρμογής

Η τελική υλοποίηση της εφαρμογής carpooling παρουσιάζει όλες τις βασικές λειτουργίες που απαιτούνται για την οργάνωση και τον συντονισμό κοινών διαδρομών μεταξύ φοιτητών του ΔΙΠΑΕ. Η εφαρμογή αναπτύχθηκε σε περιβάλλον Flutter και αξιοποιεί Firebase Cloud Firestore ως backend, επιτυγχάνοντας αποθήκευση σε πραγματικό χρόνο και άμεση ενημέρωση των δεδομένων.

Κατά την είσοδο στο σύστημα, ο χρήστης ταυτοποιείται μέσω του ιδρυματικού Single Sign-On (IHU Apps SSO), χωρίς δημιουργία ή διαχείριση κωδικών. Η διαδικασία αυτή επιτρέπει την ασφαλή αναγνώριση των χρηστών και τη δημιουργία βασικού προφίλ (π.χ. όνομα και email), το οποίο συμπληρώνεται στη συνέχεια με προαιρετικά στοιχεία οχήματος. Όλοι οι χρήστες έχουν δυνατότητα συμμετοχής σε δρομολόγια, ενώ όσοι διαθέτουν αυτοκίνητο μπορούν παράλληλα να αναρτούν δικά τους rides.

Η εφαρμογή επιτρέπει τη δημιουργία, επεξεργασία και ακύρωση διαδρομών από τον οδηγό, καθώς και την υποβολή αιτημάτων συμμετοχής από επιβάτες. Τα αιτήματα γίνονται σε πραγματικό χρόνο και ο οδηγός έχει τη δυνατότητα αποδοχής ή απόρριψης κάθε αίτησης. Η πλατφόρμα επιπλέον υποστηρίζει τη διαχείριση ολοκληρωμένων διαδρομών (completion), παρέχοντας στους χρήστες ιστορικό των ενεργειών τους.

Ως επιπρόσθετη λειτουργικότητα, αποθηκεύονται συνομιλίες μεταξύ οδηγού και επιβατών σε επίπεδο ride, επιτρέποντας τη συνεννόηση σε περίπτωση αλλαγών ή διευκρινίσεων. Οι συνομιλίες ενημερώνονται instantaneously λόγω του push-based μηχανισμού της Firestore. Τα δεδομένα δεν συγχρονίζονται offline, γεγονός το οποίο απλοποιεί την αρχιτεκτονική και αποτρέπει τοπική αποθήκευση ευαίσθητων πληροφοριών.

Με βάση τον συνολικό σχεδιασμό, η εφαρμογή παρέχει μια πλήρως λειτουργική εμπειρία carpooling σε demo/beta μορφή, αξιοποιώντας τα πλεονεκτήματα της cloud υποδομής και διατηρώντας εστίαση στη χρηστικότητα και την απλότητα.

5.2 Αξιολόγηση λειτουργικότητας

Η αξιολόγηση της εφαρμογής πραγματοποιήθηκε μέσω δοκιμών σε πραγματικό Android device και σε emulator ανάπτυξης. Οι δοκιμές επικεντρώθηκαν τόσο στη ροή χρήσης όσο και στην αξιοπιστία των κύριων λειτουργιών (authentication, rides, requests, chat). Η συνολική εμπειρία κρίνεται σταθερή, χωρίς σφάλματα που να εμποδίζουν τη χρήση.

Η διαδικασία ταυτοποίησης μέσω IHU Apps SSO λειτουργεί απρόσκοπτα, εξαλείφοντας το ενδεχόμενο λανθασμένης διαχείρισης κωδικών και επιτρέποντας ασφαλή είσοδο. Η δημιουργία προφίλ χρήστη μετά το login ολοκληρώνεται χωρίς ασυμβατότητες, ενώ τα στοιχεία του οδηγού επηρεάζουν ορθά τις διαθέσιμες δυνατότητες (ανάρτηση διαδρομών).

Η λειτουργία δημιουργίας και επεξεργασίας rides αξιολογήθηκε θετικά. Οι αλλαγές σε ημερομηνία και χρόνο εφαρμόζονται άμεσα, ενώ η λογική ακύρωσης λειτουργεί προβλέψιμα και χωρίς απώλεια δεδομένων. Η εφαρμογή εμποδίζει τη μεταβολή της τοποθεσίας εκκίνησης για υφιστάμενες διαδρομές, μειώνοντας τον κίνδυνο σύγχυσης μεταξύ επιβατών.

Το σύστημα αιτημάτων συμμετοχής είναι από τα πιο κρίσιμα στοιχεία και ανταποκρίθηκε χωρίς αστοχίες: οι χρήστες μπορούν να υποβάλουν request στο ride, και τα δεδομένα ενημερώνονται σε πραγματικό χρόνο. Η αποδοχή/απόρριψη ενημερώνει άμεσα και τις δύο πλευρές χωρίς ανάγκη refresh.

Η λειτουργία συνομιλίας (chat) επί ενός ride υλοποιείται με document-based storage και λειτουργεί αξιόπιστα. Τα μηνύματα εμφανίζονται με σωστή χρονολογική ταξινόμηση και δεν παρατηρήθηκαν διπλές ή χαμένες εγγραφές κατά τη δοκιμή. Η χρήση Firestore listeners υποστηρίζει instant feed των νέων μηνυμάτων.

Σε θέματα απόδοσης, η εφαρμογή ανταποκρίνεται ικανοποιητικά τόσο σε emulator όσο και σε φυσική συσκευή, με μικρούς χρόνους φόρτωσης και ομαλή πλοήγηση. Οι λειτουργίες δεν απαιτούν ισχυρό hardware, κάτι που υποστηρίζει τη χρήση από μεγάλο εύρος φοιτητικών συσκευών.

Συνολικά, η λειτουργικότητα της εφαρμογής σε επίπεδο MVP κρίνεται ολοκληρωμένη και συμβατή με τους αρχικούς στόχους σχεδίασης. Οι βασικοί μηχανισμοί συνεργασίας οδηγού-επιβάτη λειτουργούν ορθά, εκμεταλλευόμενοι τα πλεονεκτήματα της υποδομής Firebase.

5.3 Δοκιμές χρήσης (User testing / manual test cases)

Για την αξιολόγηση της εφαρμογής πραγματοποιήθηκαν χειροκίνητες δοκιμές σε demo/beta περιβάλλον, τόσο σε πραγματική Android συσκευή όσο και μέσω emulator.

Οι δοκιμές στόχευσαν στην επιβεβαίωση βασικών σεναρίων χρήσης που σχετίζονται με την είσοδο στο σύστημα, τη δημιουργία διαδρομών, την αναζήτηση διαθέσιμων rides και την επικοινωνία μεταξύ χρηστών.

5.3.1 Test Case 1 - Είσοδος μέσω IHU Apps

Στόχος: Επιβεβαίωση ορθής αρχικοποίησης λογαριασμού.

Ενέργεια: Ο χρήστης συνδέεται μέσω IHU Apps και οδηγείται στην αρχική ρύθμιση.

Αναμενόμενο αποτέλεσμα: Ανάκτηση ονοματεπωνύμου και email από το identity provider και αποθήκευση στο Firestore.

Παρατηρήσεις: Η διαδικασία ολοκληρώθηκε χωρίς σφάλματα σε emulator και συσκευή.

5.3.2 Test Case 2 - Δημιουργία νέας διαδρομής (Offer Ride)

Στόχος: Επιβεβαίωση σωστής καταχώρησης route και εμφάνισης στη λίστα.

Ενέργεια: Ο χρήστης ορίζει σημείο εκκίνησης, ημερομηνία/ώρα και διαθέσιμες θέσεις.

Αναμενόμενο αποτέλεσμα: Η διαδρομή αποθηκεύεται και εμφανίζεται άμεσα στο “Available Rides”.

Παρατηρήσεις: Οι αλλαγές συγχρονίζονται σε πραγματικό χρόνο χωρίς διπλές εγγραφές.

5.3.3 Test Case 3 - Αίτημα συμμετοχής (Request to Join)

Στόχος: Έλεγχος της αλληλεπίδρασης μεταξύ οδηγού και επιβάτη.

Ενέργεια: Επιβάτης στέλνει request, οδηγός εγκρίνει.

Αναμενόμενο αποτέλεσμα: Το email μεταφέρεται από requests[] σε passengers[].

Παρατηρήσεις: Η ροή λειτούργησε ομαλά· δεν παρατηρήθηκαν conflicts δεδομένων.

5.3.4 Test Case 4 - Μηνύματα (Chat per Ride)

Στόχος: Επικοινωνία οδηγού–επιβάτη.

Ενέργεια: Ανταλλαγή σύντομων μηνυμάτων.

Αναμενόμενο αποτέλεσμα: Τα μηνύματα εμφανίζονται σε χρονολογική σειρά χωρίς καθυστέρηση.

Παρατηρήσεις: Δεν διαπιστώθηκαν απώλειες· UI remained responsive.

5.3.5 Test Case 5 - Ρυθμίσεις / Αναφορά προβλήματος

Στόχος: Έλεγχος βοηθητικών λειτουργιών.

Ενέργεια: Υποβολή αναφοράς και διαγραφή λογαριασμού.

Αναμενόμενο αποτέλεσμα: Δημιουργία εγγράφου αναφοράς και ασφαλής έξοδος χρήστη.

Παρατηρήσεις: Και οι δύο λειτουργίες ολοκληρώθηκαν επιτυχώς.

Οι δοκιμές επιβεβαίωσαν τη λειτουργία των βασικών ροών χρήσης της εφαρμογής. Δεν εντοπίστηκαν κρίσιμα σφάλματα που να εμποδίζουν τη χρήση της. Μικρές βελτιώσεις αφορούν θέματα UX και ενημέρωσης χρήστη σε edge cases (π.χ. αποτυχία σύνδεσης).

5.4 Συζήτηση αποτελεσμάτων και περιορισμοί

Τα αποτελέσματα των δοκιμών επιβεβαιώνουν ότι η εφαρμογή επιτυγχάνει τον βασικό της στόχο: την υποστήριξη ενός απλού και λειτουργικού συστήματος carpooling για μέλη της ακαδημαϊκής κοινότητας του ΔΙΠΑΕ. Οι κύριες ροές χρήσης (είσοδος, δημιουργία προφίλ, αναζήτηση διαδρομών, υποβολή αιτήματος συμμετοχής και διαχείριση rides) ολοκληρώθηκαν επιτυχώς χωρίς σημαντικά σφάλματα. Η χρήση του Firebase Firestore επέτρεψε την άμεση ενημέρωση των δεδομένων (real-time) και διευκόλυνε την ασύγχρονη επικοινωνία μεταξύ οδηγών και επιβατών.

Παρά τα θετικά αποτελέσματα, η υλοποίηση παρουσιάζει ορισμένους περιορισμούς που προκύπτουν από επιλογές σχεδιασμού και το πεδίο εφαρμογής της λύσης. Πρώτον, το σύστημα βασίζεται αποκλειστικά σε έναν τύπο χρήστη (User), ο οποίος μεταβάλλει ρόλο ανάλογα με την ύπαρξη δηλωμένου οχήματος. Η προσέγγιση αυτή απλοποιεί την αρχιτεκτονική αλλά μειώνει τη δυνατότητα μελλοντικής εισαγωγής επιμέρους ρόλων, όπως διαχειριστές πλατφόρμας ή moderators.

Δεύτερον, η εφαρμογή δεν αποθηκεύει την τρέχουσα τοποθεσία των χρηστών ούτε παρέχει διαδρομές βάσει προορισμού. Η επιλογή αυτή προστατεύει την ιδιωτικότητα και μειώνει την πολυπλοκότητα, ωστόσο περιορίζει την εμπειρία χρήσης, καθώς δεν υποστηρίζεται πλήρης ταύτιση διαδρομών ή οδική βελτιστοποίηση.

Τρίτον, η εμπειρία χρήσης (UX) εξαρτάται από τη σταθερότητα της σύνδεσης με το Firebase. Σε συνθήκες χαμηλής συνδεσιμότητας, η ενημέρωση των αιτημάτων και μηνυμάτων ενδέχεται να καθυστερήσει, ενώ η offline λειτουργία περιορίζεται μόνο στην προσωρινή αποθήκευση χαρτογραφικών δεδομένων μέσω Google Maps SDK.

Τέλος, η εφαρμογή δεν έχει ακόμη αξιολογηθεί σε πραγματικό περιβάλλον χρήσης από πολλαπλούς φοιτητές ή προσωπικό του πανεπιστημίου. Οι δοκιμές έγιναν σε demo/beta περιβάλλον και δεν συμπεριέλαβαν metrics αποδοχής χρηστών, ικανοποίησης ή φόρτου συστήματος. Ως εκ τούτου, η ανατροφοδότηση είναι περιορισμένη και δεν επαληθεύεται η εμπειρία χρήσης υπό κλιμακούμενες συνθήκες.

Συνολικά, η εφαρμογή λειτουργεί σύμφωνα με τον αρχικό σχεδιασμό, όμως οι παραπάνω παράγοντες υποδεικνύουν πεδία μελλοντικής βελτίωσης τόσο σε επίπεδο UX όσο και σε επίπεδο λειτουργικής επέκτασης.

5.5 Επίλογος

Στο Κεφάλαιο 5 παρουσιάστηκαν τα αποτελέσματα της υλοποίησης της εφαρμογής carpooling και η αξιολόγηση της λειτουργικότητας της μέσα από δοκιμές χρήσης και χρονολογημένα σενάρια. Περιγράφηκαν τα βασικά χαρακτηριστικά της τελικής έκδοσης, καθώς και ο τρόπος με τον οποίο ανταποκρίνεται στις ανάγκες της στοχευμένης ομάδας χρηστών.

Αξιολογήθηκε η γενική λειτουργικότητα της εφαρμογής βάσει των πραγματικών ροών χρήσης, ανάμεσα στις οποίες περιλαμβάνονται η αναζήτηση διαθέσιμων μετακινήσεων, η δημιουργία διαδρομών από οδηγούς, η επεξεργασία προσωπικών δεδομένων προφίλ, οι λειτουργίες συνομιλίας (chat), καθώς και ο μηχανισμός αιτήσεων συμμετοχής. Παρουσιάστηκαν επίσης οι περιπτώσεις χειροκίνητων ελέγχων που επιβεβαίωσαν την ορθή λειτουργία των βασικών μηχανισμών, την ομαλή εμπειρία αλληλεπίδρασης και την απουσία κρίσιμων σφαλμάτων.

Στη συνέχεια συζητήθηκαν τα αποτελέσματα και οι περιορισμοί της τρέχουσας έκδοσης, όπως η απουσία αυτοματοποιημένων ειδοποιήσεων, ο περιορισμός της ταυτοποίησης στο IHU Apps SSO και η ανάγκη επέκτασης ορισμένων λειτουργιών σε πραγματικές συνθήκες χρήσης. Οι συγκεκριμένοι περιορισμοί δεν επηρεάζουν τον βασικό στόχο της υλοποίησης, αλλά αποτελούν πεδία βελτίωσης για μελλοντικές εκδόσεις.

Συνολικά, το Κεφάλαιο 5 ανέδειξε την ομαλή λειτουργία της εφαρμογής στην πράξη, τη σταθερότητα της αρχιτεκτονικής και την καταλληλότητά της για τις ανάγκες ενός πανεπιστημιακού περιβάλλοντος. Τα ευρήματα αυτά αποτελούν τη βάση για τα γενικότερα συμπεράσματα και τις προτάσεις εξέλιξης που παρουσιάζονται στο Κεφάλαιο 6.

Κεφάλαιο 6ο: Συμπεράσματα και Μελλοντική Βελτίωση

6.1 Συνολική αξιολόγηση της εργασίας

Η παρούσα Δ.Ε. έχει ως στόχο την ανάπτυξη μιας ολοκληρωμένης εφαρμογής carpooling προσανατολισμένης στις ανάγκες της πανεπιστημιακής κοινότητας. Η λύση αξιοποίησε ένα σύγχρονο τεχνολογικό οικοσύστημα, με Flutter για το frontend, Firebase Firestore για την αποθήκευση δεδομένων και Firebase Authentication μέσω του IHU Apps SSO για την ταυτοποίηση των χρηστών.

Η εφαρμογή που υλοποιήθηκε επιτυγχάνει τον βασικό στόχο: επιτρέπει σε οδηγούς να δημοσιεύουν διαθέσιμες διαδρομές και σε επιβάτες να αιτούνται συμμετοχή σε αυτές. Το σύστημα ενισχύει τον καθημερινό διαμοιρασμό μετακινήσεων προς και από το ΔΙΠΑΕ, παρέχοντας ένα πιο βιώσιμο, οικονομικό και κοινωνικά επωφελές μοντέλο μετακίνησης.

Από πλευράς λειτουργικότητας, το αποτέλεσμα κρίνεται σταθερό και πλήρως λειτουργικό, καλύπτοντας τα κύρια σενάρια χρήσης:

- δημιουργία διαδρομής από οδηγούς,
- αναζήτηση διαδρομών από επιβάτες,
- διαχείριση αιτημάτων συμμετοχής,
- ανταλλαγή μηνυμάτων μεταξύ χρηστών,
- βασικές βοηθητικές λειτουργίες (αναφορά προβλήματος, επεξεργασία προφίλ).

Παράλληλα, η αρχιτεκτονική Firestore επέτρεψε λιτό αλλά αποτελεσματικό μοντέλο δεδομένων, χωρίς περιττή πολυπλοκότητα ή nested collections, ευνοώντας επεκτασιμότητα και ευκολία συντήρησης.

6.2 Συμπεράσματα

Η ανάπτυξη της εφαρμογής ανέδειξε ορισμένα σημαντικά συμπεράσματα:

1. Η ολοκλήρωση Flutter - Firebase προσφέρει ιδανική ισορροπία μεταξύ ευκολίας υλοποίησης και λειτουργικής σταθερότητας.
   Η απουσία backend server, σε συνδυασμό με real-time listeners και event-driven ενημερώσεις, μείωσε σημαντικά την πολυπλοκότητα.

2. Η χρήση ιδρυματικής ταυτοποίησης (IHU Apps) εξασφαλίζει εμπιστοσύνη μεταξύ χρηστών.
   Η εφαρμογή δεν επιτρέπει αυθαίρετες εγγραφές ή ανώνυμη χρήση, μειώνοντας τον κίνδυνο καταχρήσεων και ψευδών προφίλ.

3. Ο σχεδιασμός UI/UX με κεντρικό άξονα τη χαμηλή γνωστική επιβάρυνση (cognitive load) βελτίωσε την εμπειρία χρήστη.
   Οι βασικές λειτουργίες απαιτούν ελάχιστες ενέργειες και δεν αποπροσανατολίζουν τον χρήστη από τον κύριο στόχο.

4. Η υλοποίηση chat per ride απλοποιεί την επικοινωνία οδηγού–επιβατών, αποφεύγοντας αποσπάσεις ή εκτροπές.

5. Το Firestore schema αποδείχθηκε επεκτάσιμο.
   Παρότι ξεκίνησε με περιορισμένα πεδία, επιτρέπει εύκολη εισαγωγή νέων χαρακτηριστικών όπως ratings, αποθήκευση ιστορικού ή monitoring analytics.

6.3 Προτάσεις για μελλοντική βελτίωση

Παρότι η εφαρμογή υλοποιήθηκε ολοκληρωμένα, υπάρχουν κατευθύνσεις για περαιτέρω εξέλιξη:

- Δημιουργία συστήματος αξιολόγησης οδηγών και επιβατών.
  Features όπως ratings, reviews ή ιστορική συμπεριφορά ενισχύουν την εμπιστοσύνη και βελτιώνουν την ποιότητα των rides.
- Υλοποίηση αυτόματης αντιστοίχισης οδηγών–επιβατών βάσει απόστασης/ώρας.
  Αλγόριθμοι best-match μπορούν να μειώσουν τα ανεπιτυχή αιτήματα και να αυξήσουν τη χρηστικότητα.
- Σύστημα ειδοποιήσεων (push notifications).
  Για επιβεβαιώσεις συμμετοχής, ενημέρωση αλλαγών, υπενθυμίσεις διαδρομών κ.λπ.
- Υποστήριξη ολοκλήρωσης διαδρομών σε πραγματικό χρόνο.
  Π.χ. καταγραφή θέσης οχήματος κατά τη διάρκεια της διαδρομής, μόνο με ρητή συγκατάθεση χρήστη.
- Ενσωμάτωση μηχανισμού ασφαλούς διαγραφής λογαριασμού και export δεδομένων χρήστη.
  Επιπλέον ευθυγράμμιση με τις απαιτήσεις GDPR.
- Δημιουργία web dashboard για administrators.
  Παρακολούθηση αναφορών, moderation chats, εποπτεία rides και trends χρήσης.
- Βελτίωση chat σε επίπεδο εμπειρίας.
  Υποστήριξη multimedia, timestamps, read receipts.

6.4 Επίλογος

Η Δ.Ε. ανέπτυξε μια λειτουργική εφαρμογή carpooling προσαρμοσμένη στις ανάγκες της πανεπιστημιακής κοινότητας, βασισμένη σε Flutter, Firebase Firestore και ιδρυματική ταυτοποίηση μέσω IHU Apps. Η υλοποίηση απέδειξε ότι μια ελαφριά αρχιτεκτονική, χωρίς περίπλοκες δομές, μπορεί να εξυπηρετήσει αποτελεσματικά βασικές ανάγκες συνεργατικής μετακίνησης.

Παρά τους περιορισμούς, όπως η απουσία εκτεταμένης δοκιμής με πραγματικούς χρήστες και η ανάγκη για περαιτέρω αυτοματοποίηση, το σύστημα θέτει σταθερή βάση για μελλοντική εξέλιξη, επιβεβαιώνοντας τη χρησιμότητα του ως εργαλείο βελτίωσης της φοιτητικής μετακίνησης και ενίσχυσης της βιώσιμης κινητικότητας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] European Commission, “Smart Mobility and Transport,” 2023. [Online]. Available: https://transport.ec.europa.eu

[2] S. Shaheen and A. Cohen, “Shared ride services in North America: definitions, impacts, and the future of carpooling,” Transport Reviews, vol. 38, no. 6, pp. 710-724, 2018.

[3] A. Amey, “Real-Time Ridesharing: Exploring the Opportunities and Challenges of Designing a Technology-Based Rideshare Trial for the MIT Community,” M.S. thesis, Massachusetts Institute of Technology, 2011.

[4] D. Efthymiou and C. Antoniou, “Understanding the effects of carpooling on students’ travel behavior,” Procedia - Social and Behavioral Sciences, vol. 48, pp. 1883-1892, 2012.

[5] Google Developers, “Flutter Documentation,” 2024. [Online]. Available: https://docs.flutter.dev

[6] Eurostat, “Student mobility and commuting patterns in Europe,” 2022.

[7] Google Maps, “Thessaloniki to Sindos Campus - Typical Travel Time,” 2024.

[8] ΟΑΣΘ, “Δρομολόγια και συχνότητες γραμμών,” 2023. [Online]. Available: https://oasth.gr

[9] ELSTAT, “Μέσο κόστος μετακίνησης νοικοκυριών,” Έρευνα Οικογενειακών Προϋπολογισμών, 2023.

[10] C. Mulley and J. Nelson, “Flexible transport services: A new mobility paradigm?”, Research in Transportation Economics, vol. 59, pp. 1-7, 2016.

[11] University of California, “Campus Mobility Programs and Sustainable Transport,” 2021.

[12] S. Shaheen, A. Cohen, and I. Zohdy, “Shared Mobility: Current Practices and Guiding Principles,” U.S. Department of Transportation, 2016.

[13] Green Mobilities Greece, “Carpooling Trends in Greek Urban Areas,” 2022.

[14] M. Burris and K. Sung, “An analysis of the impacts of carpooling on traffic congestion,” Journal of Transportation Systems, vol. 12, pp. 55-63, 2019.

[15] FreeNow, “About Us - Ride Hailing in Europe,” 2024. [Online]. Available: https://www.free-now.com

[16] BlaBlaCar, “About Us,” 2024. [Online]. Available: https://www.blablacar.com/about

[17] Liftshare, “Corporate Car Sharing Solutions,” 2024. [Online]. Available: https://liftshare.com

[18] Karos, “Smart Carpooling for Daily Commutes,” 2024. [Online]. Available: https://www.karos.fr

[19] GoCarma, “Smarter Carpooling in Dallas-Fort Worth,” 2023. [Online]. Available: https://www.gocarma.com

[20] sRide, “Corporate Carpooling Platform in India,” 2023. [Online]. Available: https://www.sride.co

[21] ETH Zurich, “Sustainable Mobility Strategy and Carpooling Platform,” ETH Zürich Sustainability Reports, 2023.

[22] University of Cambridge, “Travel Plan and Car Sharing Scheme,” Transport Department, 2022.

[23] M. Lee, T. Jacobs, and R. van der Meer, “Reducing Campus Car Use Through Ridesharing: A Quantitative Study,” Journal of Sustainable Mobility, vol. 15, no. 2, pp. 112-127, 2021.

[24] R. Agatz, A. Erera, M. Savelsbergh, and X. Wang, “Dynamic ride-sharing: A simulation study in metro Atlanta,” Transportation Research Part B: Methodological, vol. 45, no. 9, pp. 1450-1464, 2011.

[25] NACTO, “Shared Micromobility in the U.S.: 2020 Snapshot,” National Association of City Transportation Officials, 2021. [Online]. Available: https://nacto.org/shared-micromobility-2020/

[26] R. Conway and B. Nelson, “A Comparative Framework for Evaluating Urban Mobility Services,” Urban Transport Review, vol. 17, no. 1, pp. 45-59, 2022.

[27] M. C. Kim and L. Johnson, “Mobile Application Security Guidelines for User-Centric Platforms,” Journal of Mobile Systems, vol. 18, no. 3, pp. 201-215, 2022.

[28] Nielsen Norman Group, “Usability Heuristics for User Interface Design,” 2024. [Online]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/

[29] I. Jacobson, “Object-Oriented Software Engineering: A Use Case Driven Approach,” Addison-Wesley, 1992.

[30] European Union, “General Data Protection Regulation (GDPR), Regulation (EU) 2016/679,” Official Journal of the European Union, 2016.

[31] Google Developers, “Android Studio Documentation,” 2024. [Online]. Available: https://developer.android.com/studio

[32] Google Material Design, “Material Design Guidelines,” 2024. [Online]. Available: https://m3.material.io/

ΠΑΡΑΡΤΗΜΑ Α : Δημιουργία Χρήστη (User Creation)

```dart
/*
 * Creates a user
 * Also saves whether the user has a car or not
 */
import 'package:dipaecarpooling/utils/authentication.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:shared_preferences/shared_preferences.dart';

/// Formats license plates into AAA-1234 while typing.
/// - uppercases letters
/// - removes invalid chars
/// - inserts a hyphen after 3 letters
/// - limits to 3 letters and 4 digits
class LicensePlateFormatter extends TextInputFormatter {
  final int letters = 3;
  final int digits = 4;

  @override
  TextEditingValue formatEditUpdate(
      TextEditingValue oldValue, TextEditingValue newValue) {
    // Uppercase and remove all non-alphanumeric
    String sanitized = newValue.text.toUpperCase().replaceAll(RegExp(r'[^A-ZΑ-Ω0-9]'), '');

    // Ensure first part contains only letters
    String part1 = '';
    String part2 = '';

    if (sanitized.isNotEmpty) {
      // Take up to `letters` letters for the prefix
      StringBuffer buf = StringBuffer();
      int letterCount = 0;
      for (int i = 0; i < sanitized.length && letterCount < letters; i++) {
        String ch = sanitized[i];
        if (RegExp(r'[A-ZΑ-Ω]').hasMatch(ch)) {
          buf.write(ch);
          letterCount++;
        }
      }
      part1 = buf.toString();
      // The rest should be digits (take up to `digits`)
      StringBuffer buf2 = StringBuffer();
      int digitCount = 0;
      // We iterate original sanitized string after consuming letters to allow user to have mixed input order,
      // but only digits will be collected for second part.
      for (int i = 0; i < sanitized.length && digitCount < digits; i++) {
        String ch = sanitized[i];
        if (RegExp(r'[0-9]').hasMatch(ch)) {
          buf2.write(ch);
          digitCount++;
        }
      }
      part2 = buf2.toString();
    }
```

```
    String formatted = part1;
    if (part2.isNotEmpty) formatted = '$formatted-$part2';

    // Limit overall length
    if (formatted.length > letters + 1 + digits) {
      formatted = formatted.substring(0, letters + 1 + digits);
    }

    // Try to maintain a reasonable cursor position: put at end
    return TextEditingValue(
      text: formatted,
      selection: TextSelection.collapsed(offset: formatted.length),
    ); // TextEditingValue
  }
}

class BeforeYouGetStartedScreen extends StatefulWidget {
  @override
  _BeforeYouGetStartedScreenState createState() =>
      _BeforeYouGetStartedScreenState();
}

class _BeforeYouGetStartedScreenState extends State<BeforeYouGetStartedScreen> {
  bool hasCar = false;
  final TextEditingController licensePlateController = TextEditingController();
  final TextEditingController carBrandController = TextEditingController();
  String userName = '';

  @override
  void initState() {
    super.initState();
    _loadUserName();
  }

  Future<void> _loadUserName() async {
    final prefs = await SharedPreferences.getInstance();
    setState(() {
      userName =
          '${prefs.getString('firstName') ?? ''} ${prefs.getString('lastName') ?? ''}';
    });
  }

  @override
  void dispose() {
    licensePlateController.dispose();
    carBrandController.dispose();
    super.dispose();
  }

```

```dart
void _submit() async {
  // If user has a car, ensure both fields are filled
  if (hasCar) {
    String plate = licensePlateController.text.trim().toUpperCase();
    licensePlateController.text = plate; // normalize displayed text

    if (plate.isEmpty || carBrandController.text.trim().isEmpty) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
      content: Text('Please enter your license plate and car brand'),
      ), // SnackBar
    );
    return;
    }

    // Validate license plate format: AAA-1234 (3 letters including Greek, hyphen, 4 digits)
    final plateRegex = RegExp(r'^[A-ZΑ-Ω]{3}-\d{4}$');
    if (!plateRegex.hasMatch(plate)) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
      content: Text('License plate must be in format AAA-1234 (Greek letters allowed)'),
      ), // SnackBar
    );
    return;
    }
  }

  // Handle submission logic here
  print('User Name: $userName');
  print('Has Car: $hasCar');
  if (hasCar) {
    print('License Plate: ${licensePlateController.text}');
    print('Car Brand: ${carBrandController.text}');
  }
  final prefs = await SharedPreferences.getInstance();
  String email = prefs.getString('email') ?? '';
  String firstName = prefs.getString('firstName') ?? '';
  String lastName = prefs.getString('lastName') ?? '';
  saveNewUser(
    email: email,
    firstName: firstName,
    lastName: lastName,
    ridesTaken: 0,
    ratings: [],
    ridesCompleted: 0,
    nextRideID: "-1",
    hasCar: hasCar,
    carName: carBrandController.text,
    carPlate: licensePlateController.text,
  );

```

```
    prefs.setInt('ridesTaken', 0);
    prefs.setInt('ridesCompleted', 0);
    prefs.setString('nextRideID', "-1");
    prefs.setBool('hasCar', hasCar);
    Navigator.pushReplacementNamed(context, '/maps');
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Before You Get Started'),
        backgroundColor: const Color.fromRGBO(99, 143, 166, 1),
        automaticallyImplyLeading: false,
      ), // AppBar
      backgroundColor: const Color.fromRGBO(99, 143, 166, 1),
      body: Padding(
        padding: const EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              'Your Name:',
              style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ), // Text
            TextField(
              controller: TextEditingController(text: userName),
              decoration: InputDecoration(border: OutlineInputBorder()),
              enabled: false,
            ), // TextField
            SizedBox(height: 20),
            Text(
              'Do you have a car?',
              style: TextStyle(fontSize: 18, fontWeight: FontWeight.bold),
            ), // Text
            SwitchListTile(
              title: Text('I have a car'),
              value: hasCar,
              onChanged: (value) {
                setState(() {
                  hasCar = value;
                });
              },
            ), // SwitchListTile
            if (hasCar) ...[
              TextField(
                controller: licensePlateController,
                keyboardType: TextInputType.text,
                textCapitalization: TextCapitalization.characters,
                inputFormatters: [
                  LicensePlateFormatter(),
                ],
```

```
                decoration: InputDecoration(
                  labelText: 'License Plate (format: AAA-1234)',
                  border: OutlineInputBorder(),
                ), // InputDecoration
              ), // TextField
              SizedBox(height: 10),
              TextField(
                controller: carBrandController,
                decoration: InputDecoration(
                  labelText: 'Car Brand (e.g., Toyota Corolla)',
                  border: OutlineInputBorder(),
                ), // InputDecoration
              ), // TextField
            ],
            SizedBox(height: 20),
            Center(
              child: ElevatedButton(onPressed: _submit, child: Text('Submit')),
            ), // Center
          ],
        ), // Column
      ), // Padding
    ); // Scaffold
  }
}

```

ΠΑΡΑΡΤΗΜΑ Β : Προσθήκη Οχήματος (Add Vehicle)

```dart
import 'package:dipaecarpooling/utils/authentication.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';
import './utils/rides.dart';

class AddVehicle extends StatefulWidget {
  final void Function() hideBottomSheet;
  final String carName;
  final String licensePlate;

  AddVehicle({
    required this.hideBottomSheet,
    required this.carName,
    required this.licensePlate,
  });

  @override
  _AddVehicleState createState() => _AddVehicleState();
}

class _AddVehicleState extends State<AddVehicle> {
  late String _carName = "";
  late String _licensePlate;

  final TextEditingController _licensePlateController = TextEditingController();
  final TextEditingController _carNameController = TextEditingController();

  @override
  void initState() {
    super.initState();
    _carName = widget.carName;
    _licensePlate = widget.licensePlate;
    _licensePlateController.text = _licensePlate;
    _carNameController.text = _carName;
  }

  @override
  void dispose() {
    _licensePlateController.dispose();
    _carNameController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Add a Vehicle'),
      ), // AppBar
      body: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16.0),
          child: Column(
```

```
54                children: [
55                  Focus(
56                    child: TextField(
57                      decoration: InputDecoration(
58                        labelText: 'Car Model Name',
59                        border: OutlineInputBorder(),
60                      ), // InputDecoration
61                      controller: _carNameController,
62                      onChanged: (value) {
63                        setState(() {
64                          _carName = value;
65                        });
66                      },
67                    ), // TextField
68                  ), // Focus
69                  const SizedBox(height: 16),
70                  Focus(
71                    child: TextField(
72                      decoration: InputDecoration(
73                        labelText: 'Car License Plate',
74                        border: OutlineInputBorder(),
75                      ), // InputDecoration
76                      controller: _licensePlateController,
77                      onChanged: (value) {
78                        setState(() {
79                          _licensePlate = value;
80                        });
81                      },
82                    ), // TextField
83                  ), // Focus
84                  const SizedBox(height: 24),
85                  ElevatedButton(
86                    onPressed: () async {
87                      if (_carName.isEmpty ||
88                          _licensePlate.isEmpty) {
89                        ScaffoldMessenger.of(context).showSnackBar(
90                          SnackBar(content: Text('Please fill in all fields')),
91                        );
92                        return;
93                      } else {
94                        widget.hideBottomSheet();
95                        SharedPreferences prefs =
96                            await SharedPreferences.getInstance();
97                        String? riderEmail = prefs.getString('email');
98                        if (riderEmail == null) {
99                          ScaffoldMessenger.of(context).showSnackBar(
100                           SnackBar(content: Text('User not logged in')),
101                         );
102                         return;
103                       }
```

```
104                         addCarToUser(
105                           email: riderEmail,
106                           carName: _carName,
107                           carPlate: _licensePlate,
108                         );
109                       }
110                     },
111                     child: Text('Submit'),
112                   ), // ElevatedButton
113                 ],
114               ), // Column
115             ), // Padding
116           ), // SingleChildScrollView
117         ); // Scaffold
118       }
119     }
```

ΠΑΡΑΡΤΗΜΑ C : Επεξεργασία Οχήματος (Edit Vehicle)

```dart
import 'package:dipaecarpooling/utils/authentication.dart';
import 'package:dipaecarpooling/utils/rides.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:shared_preferences/shared_preferences.dart';

class EditVehicleScreen extends StatefulWidget {
  const EditVehicleScreen({super.key});

  @override
  _EditVehicleScreenState createState() => _EditVehicleScreenState();
}

class _EditVehicleScreenState extends State<EditVehicleScreen> {
  String userEmail = 'Loading';
  late TextEditingController modelController;
  late TextEditingController plateController;
  bool isPlateValid = false;

  @override
  void initState() {
    super.initState();
    modelController = TextEditingController();
    plateController = TextEditingController();
    _loadUserData();
  }

  Future<void> _loadUserData() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    final String fetchedUserEmail = prefs.getString('email') ?? '';
    final String fetchedVehicleModel = prefs.getString('carName') ?? '';
    final String fetchedVehiclePlate = prefs.getString('carPlate') ?? '';

    setState(() {
      userEmail = fetchedUserEmail;
      // keep empty strings instead of "Vehicle not found" so validation works as expected
      modelController.text = fetchedVehicleModel;
      plateController.text = fetchedVehiclePlate.toUpperCase();
      isPlateValid = _isPlateValid(plateController.text.toUpperCase());
    });
  }

  @override
  void dispose() {
    modelController.dispose();
    plateController.dispose();
    super.dispose();
  }

  // Auto-format input to "AAA-XXXX" style while typing
  void _onPlateChanged(String value) {
    String v = value.toUpperCase();
    // keep only letters and digits
    v = v.replaceAll(RegExp('[^A-Z0-9]'), '');
```

```
    // limit to 7 chars (3 letters + 4 digits)
    if (v.length > 7) v = v.substring(0, 7);

    // insert hyphen after 3 chars if there are more than 3 chars
    String formatted;
    if (v.length <= 3) {
      formatted = v;
    } else {
      formatted = v.substring(0, 3) + '-' + v.substring(3);
    }

    // limit full length to 8 including hyphen
    if (formatted.length > 8) formatted = formatted.substring(0, 8);

    if (formatted != plateController.text) {
      final selectionIndex = formatted.length;
      plateController.value = TextEditingValue(
        text: formatted,
        selection: TextSelection.collapsed(offset: selectionIndex),
      ); // TextEditingValue
    }

    // update validity state
    setState(() {
      isPlateValid = _isPlateValid(formatted);
    });
  }

  bool _isPlateValid(String plate) {
    final pattern = RegExp(r'^[A-Z]{3}-\d{4}$');
    return pattern.hasMatch(plate);
  }

  void _showInvalidPlateMessage() {
    ScaffoldMessenger.of(context).showSnackBar(
      const SnackBar(content: Text('License plate must be in format AAA-1234')),
    );
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text(
          'Edit Vehicle',
          style: TextStyle(color: Colors.white),
        ), // Text
        iconTheme: const IconThemeData(color: Colors.white),
        centerTitle: true,
        backgroundColor: const Color.fromRGBO(50, 50, 50, 1),
      ), // AppBar
```

```
107         body: Container(
108           color: const Color.fromRGBO(30, 30, 30, 1),
109           padding: const EdgeInsets.all(16.0),
110           child: Column(
111             crossAxisAlignment: CrossAxisAlignment.start,
112             children: [
113               Card(
114                 color: const Color.fromRGBO(40, 40, 40, 1),
115                 child: Padding(
116                   padding: const EdgeInsets.all(16.0),
117                   child: Column(
118                     crossAxisAlignment: CrossAxisAlignment.start,
119                     children: [
120                       const SizedBox(height: 16),
121                       TextField(
122                         controller: modelController,
123                         style: const TextStyle(color: Colors.white, fontSize: 20),
124                         decoration: const InputDecoration(
125                           labelText: 'Vehicle Model',
126                           labelStyle: TextStyle(color: Colors.white),
127                           enabledBorder: UnderlineInputBorder(
128                             borderSide: BorderSide(color: Colors.white),
129                           ), // UnderlineInputBorder
130                           focusedBorder: UnderlineInputBorder(
131                             borderSide: BorderSide(color: Colors.green),
132                           ), // UnderlineInputBorder
133                         ), // InputDecoration
134                       ), // TextField
135                       const SizedBox(height: 8),
136                       TextField(
137                         controller: plateController,
138                         onChanged: _onPlateChanged,
139                         inputFormatters: [
140                           FilteringTextInputFormatter.allow(
141                             RegExp(r'[A-ZΑ-Ωα-ωa-z0-9-]'),
142                           ), // FilteringTextInputFormatter.allow
143                           LengthLimitingTextInputFormatter(8),
144                         ],
145                         style: const TextStyle(color: Colors.white, fontSize: 18),
146                         decoration: const InputDecoration(
147                           hintText: 'AAA-1234',
148                           labelText: 'Vehicle Plate',
149                           labelStyle: TextStyle(color: Colors.white),
150                           enabledBorder: UnderlineInputBorder(
151                             borderSide: BorderSide(color: Colors.white),
152                           ), // UnderlineInputBorder
153                           focusedBorder: UnderlineInputBorder(
154                             borderSide: BorderSide(color: Colors.green),
155                           ), // UnderlineInputBorder
156                         ), // InputDecoration
157                       ), // TextField
```

```
158                 if (plateController.text.isNotEmpty && !isPlateValid) ...[
159                   const SizedBox(height: 6),
160                   const Text(
161                     'License plate must be in format AAA-1234',
162                     style: TextStyle(color: Colors.red, fontSize: 13),
163                   ), // Text
164                 ],
165                 const SizedBox(height: 8),
166                 Row(
167                   mainAxisAlignment: MainAxisAlignment.spaceBetween,
168                   children: [
169                     ElevatedButton(
170                       style: ElevatedButton.styleFrom(
171                         backgroundColor: Colors.red,
172                       ),
173                       onPressed: () {
174                         showDialog(
175                           context: context,
176                           builder: (BuildContext context) {
177                             return AlertDialog(
178                               title: const Text('Are you sure?'),
179                               content: const Text(
180                                 'Do you really want to delete this vehicle?',
181                               ), // Text
182                               actions: [
183                                 TextButton(
184                                   onPressed: () {
185                                     Navigator.of(
186                                       context,
187                                     ).pop(); // Close the dialog
188                                   },
189                                   child: const Text('No'),
190                                 ), // TextButton
191                                 TextButton(
192                                   onPressed: () {
193                                     addCarToUser(
194                                       email: userEmail,
195                                       carName: "",
196                                       carPlate: "",
197                                     );
198                                     modelController.text = '';
199                                     plateController.text = '';
200                                     setState(() {
201                                       isPlateValid = false;
202                                     });
203                                     Navigator.of(
204                                       context,
205                                     ).pop(); // Close dialog
206                                     if (context.mounted) {
207                                       ScaffoldMessenger.of(
208                                         context,
209                                       ).showSnackBar(
```

```
                                    const SnackBar(
                                      content: Text(
                                        'Vehicle deleted successfully',
                                      ), // Text
                                    ), // SnackBar
                                  );
                                }
                              },
                              child: const Text('Yes'),
                            ), // TextButton
                          ],
                        ); // AlertDialog
                      },
                    );
                  },
                  child: const Text(
                    'Delete Vehicle',
                    style: TextStyle(color: Colors.white),
                  ), // Text
                ), // ElevatedButton
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.green,
                  ),
                  onPressed:
                      isPlateValid
                          ? () {
                            final plateText =
                                plateController.text.toUpperCase();
                            if (!_isPlateValid(plateText)) {
                              _showInvalidPlateMessage();
                              return;
                            }
                            addCarToUser(
                              email: userEmail,
                              carName: modelController.text,
                              carPlate: plateText,
                            );
                            Navigator.of(context).pop();
                            if (context.mounted) {
                              ScaffoldMessenger.of(
                                context,
                              ).showSnackBar(
                                const SnackBar(
                                  content: Text(
                                    'Vehicle updated successfully',
                                  ), // Text
                                ), // SnackBar
                              );
                            }
                          }
```

```
                                    : null,
                                child: const Text(
                                  'Save Changes',
                                  style: TextStyle(color: Colors.white),
                                ), // Text
                              ), // ElevatedButton
                            ],
                          ), // Row
                        ],
                      ), // Column
                    ), // Padding
                  ), // Card
                ],
              ), // Column
            ), // Container
          ); // Scaffold
        }
      }
```

ΠΑΡΑΡΤΗΜΑ D : Επεξεργασία Διαδρομής (Edit Ride)

```dart
import 'package:dipaecarpooling/utils/authentication.dart';
import 'package:dipaecarpooling/utils/rides.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';

class EditRideScreen extends StatefulWidget {
  const EditRideScreen({super.key});

  @override
  _EditRideScreenState createState() => _EditRideScreenState();
}

class _EditRideScreenState extends State<EditRideScreen> {
  String rideID = '';
  String driverName = 'Loading';
  String startingLocation = 'Loading';
  String datetime = 'Loading';
  List<dynamic> passengers = [];
  List<dynamic> requests = [];

  @override
  void initState() {
    super.initState();
    _loadUserData();
  }

  Future<void> _loadUserData() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    final String fetchedRideID = prefs.getString('rideID') ?? '';
    Map<String, dynamic>? rideDetails = await getRideDetailsByID(fetchedRideID);
    Map<String, dynamic>? driverDetails = await getUserDetailsByEmail(
      rideDetails?['riderEmail'] ?? '',
    );
    if (rideDetails != null) {
      setState(() {
        rideID = fetchedRideID;
        driverName =
            driverDetails != null
                ? '${driverDetails['firstName']} ${driverDetails['lastName']}'
                : 'Driver not found';
        startingLocation =
            rideDetails['startLocationAddress'] ?? 'Starting location not set';
        datetime = rideDetails['rideDate'] ?? 'Date & Time not set';
        passengers = rideDetails['passengers'] ?? [];
        requests = rideDetails['requests'] ?? [];
      });
    } else {
      setState(() {
        rideID = fetchedRideID;
        driverName = 'Driver not found';
      });
    }
  }

```

```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
      title: const Text('Edit Ride', style: TextStyle(color: Colors.white)),
      iconTheme: const IconThemeData(color: Colors.white),
      centerTitle: true,
      backgroundColor: const Color.fromRGBO(50, 50, 50, 1),
    ), // AppBar
    body: Container(
      color: const Color.fromRGBO(30, 30, 30, 1),
      padding: const EdgeInsets.all(16.0),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Card(
            color: const Color.fromRGBO(40, 40, 40, 1),
            child: Padding(
              padding: const EdgeInsets.all(16.0),
              child: Column(
                crossAxisAlignment: CrossAxisAlignment.start,
                children: [
                  const SizedBox(height: 16),
                  Text(
                    'Starting Location: $startingLocation',
                    style: TextStyle(color: Colors.white, fontSize: 20),
                  ), // Text
                  const SizedBox(height: 8),
                  Text(
                    'Driver: $driverName',
                    style: TextStyle(color: Colors.white, fontSize: 18),
                  ), // Text
                  const SizedBox(height: 8),
                  Row(
                    children: [
                      Expanded(
                        child: Text(
                          'Date & Time: $datetime',
                          style: TextStyle(color: Colors.white, fontSize: 18),
                        ), // Text
                      ), // Expanded
                      IconButton(
                        icon: Icon(Icons.edit, color: Colors.blue),
                        tooltip: 'Edit Date & Time',
                        onPressed: () async {
                          DateTime? pickedDate = await showDatePicker(
                            context: context,
                            initialDate:
                                DateTime.tryParse(datetime) ?? DateTime.now(),
                            firstDate: DateTime(2000),
                            lastDate: DateTime(2100),
                          );
```

```
                    if (pickedDate != null) {
                      TimeOfDay? pickedTime = await showTimePicker(
                        context: context,
                        initialTime: TimeOfDay.fromDateTime(
                          DateTime.tryParse(datetime) ?? DateTime.now(),
                        ), // TimeOfDay.fromDateTime
                      );
                      if (pickedTime != null) {
                        final newDateTime = DateTime(
                          pickedDate.year,
                          pickedDate.month,
                          pickedDate.day,
                          pickedTime.hour,
                          pickedTime.minute,
                        ); // DateTime
                        // Format as DD/MM/YYYY HH:mm
                        final formattedDateTime =
                            '${newDateTime.day.toString().padLeft(2, '0')}/'
                            '${newDateTime.month.toString().padLeft(2, '0')}/'
                            '${newDateTime.year} '
                            '${newDateTime.hour.toString().padLeft(2, '0')}:'
                            '${newDateTime.minute.toString().padLeft(2, '0')}';
                        // Save the new datetime to backend here if needed
                        updateRideDate(rideID, formattedDateTime);
                        setState(() {
                          datetime = formattedDateTime;
                        });
                      }
                    }
                  },
                ), // IconButton
              ],
            ), // Row
            const SizedBox(height: 16),
            if (requests.isNotEmpty) ...[
              Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Text(
                    'Requests:',
                    style: const TextStyle(
                      fontSize: 20,
                      fontWeight: FontWeight.bold,
                    ), // TextStyle
                  ), // Text
                ],
              ), // Row
```

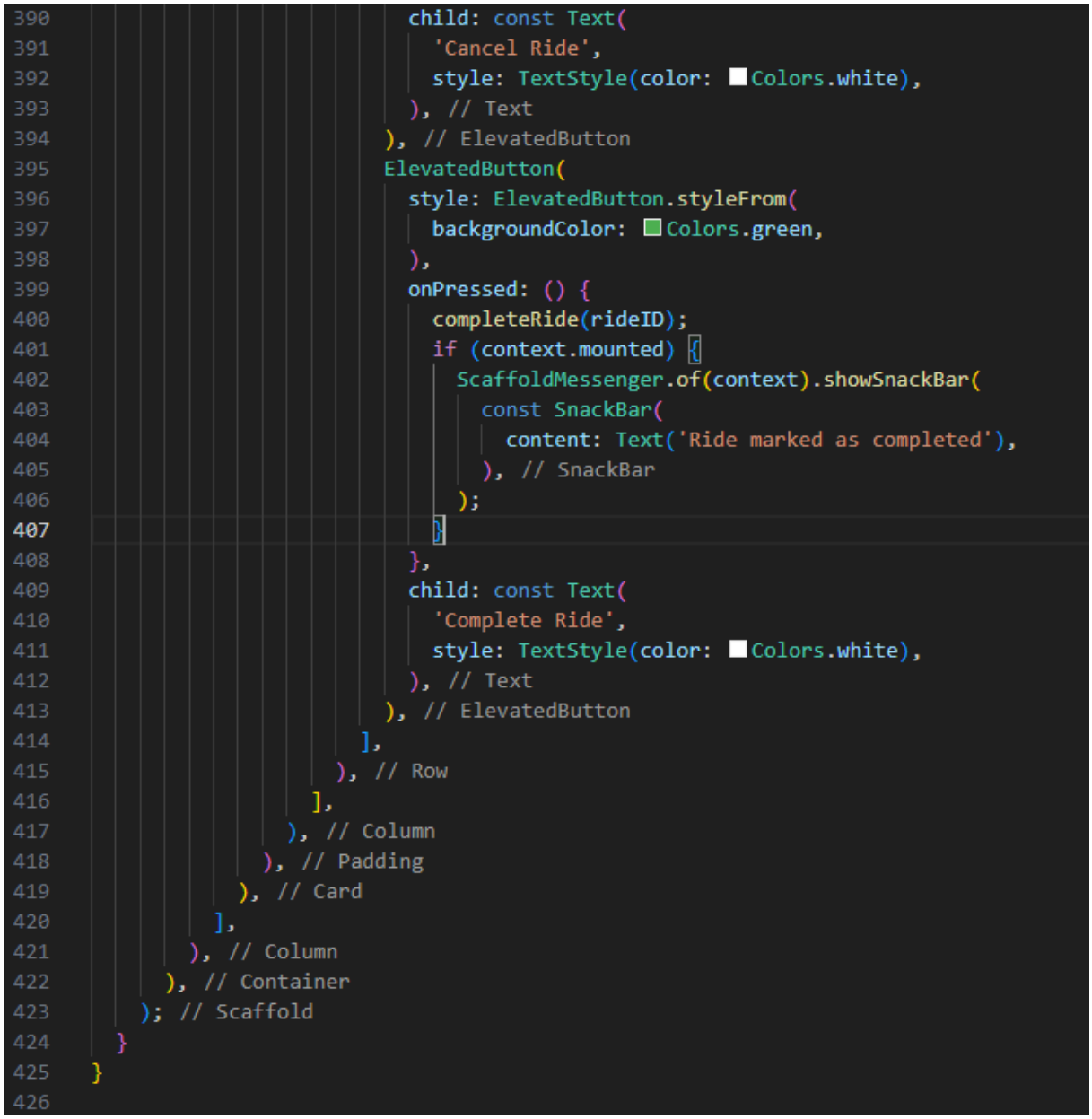
```
for (var request in requests) ...[
  FutureBuilder<Map<String, dynamic>>(
    future: getUserDetailsByEmail(
      request,
    ).then((value) => value ?? {}),
    builder: (context, snapshot) {
      if (snapshot.connectionState ==
          ConnectionState.waiting) {
        return const CircularProgressIndicator();
      } else if (snapshot.hasError) {
        return Text(
          'Error: ${snapshot.error}',
          style: const TextStyle(
            fontSize: 18,
            color: Colors.red,
          ), // TextStyle
        ); // Text
      } else if (snapshot.hasData) {
        final userDetails = snapshot.data!;
        final passengerName =
            "${userDetails['firstName'] ?? 'Unknown'} ${userDetails['lastName'] ?? ''}"
                .trim();
        return Column(
          children: [
            Text(
              passengerName,
              style: const TextStyle(
                fontSize: 18,
                color: Colors.white,
              ), // TextStyle
            ), // Text
            Row(
              mainAxisAlignment:
                  MainAxisAlignment.spaceBetween,
              children: [
                ElevatedButton(
                  style: ElevatedButton.styleFrom(
                    backgroundColor: Colors.red,
                  ),
                  onPressed: () {
                    rejectRequest(rideID, request);
                    setState(() {
                      requests.remove(request);
                    });
```

```
                        if (context.mounted) {
                          ScaffoldMessenger.of(
                            context,
                          ).showSnackBar(
                            const SnackBar(
                              content: Text(
                                'Request rejected',
                              ), // Text
                            ), // SnackBar
                          );
                        }
                      },
                      child: const Text(
                        'Reject Request',
                        style: TextStyle(color: Colors.white),
                      ), // Text
                    ), // ElevatedButton
                    ElevatedButton(
                      style: ElevatedButton.styleFrom(
                        backgroundColor: Colors.green,
                      ),
                      onPressed: () {
                        acceptAsPassenger(rideID, request);
                        setState(() {
                          passengers.add(request);
                          requests.remove(request);
                          if (context.mounted) {
                            ScaffoldMessenger.of(
                              context,
                            ).showSnackBar(
                              SnackBar(
                                content: Text(
                                  '$passengerName has been accepted as passenger',
                                ), // Text
                              ), // SnackBar
                            );
                          }
                        });
                      },
                      child: const Text(
                        'Accept Request',
                        style: TextStyle(color: Colors.white),
                      ), // Text
                    ), // ElevatedButton
                  ],
                ), // Row
              ],
            ); // Column
          } else {
```

```
                    return const Text(
                      'No data available',
                      style: TextStyle(fontSize: 18),
                    ); // Text
                  }
                },
              ), // FutureBuilder
            ],
          ],
          if (passengers.isNotEmpty) ...[
            Row(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                Text(
                  'Passengers:',
                  style: const TextStyle(
                    fontSize: 20,
                    fontWeight: FontWeight.bold,
                    color: Colors.white,
                  ), // TextStyle
                ), // Text
              ],
            ), // Row
            for (var passenger in passengers) ...[
              FutureBuilder<Map<String, dynamic>>(
                future: getUserDetailsByEmail(
                  passenger,
                ).then((value) => value ?? {}),
                builder: (context, snapshot) {
                  if (snapshot.connectionState ==
                      ConnectionState.waiting) {
                    return const CircularProgressIndicator();
                  } else if (snapshot.hasError) {
                    return Text(
                      'Error: ${snapshot.error}',
                      style: const TextStyle(
                        fontSize: 18,
                        color: Colors.red,
                      ), // TextStyle
                    ); // Text
                  } else if (snapshot.hasData) {
                    final userDetails = snapshot.data!;
                    final passengerName =
                        "${userDetails['firstName'] ?? 'Unknown'} ${userDetails['lastName'] ?? ''}"
                            .trim();
                    return Row(
                      mainAxisAlignment:
                          MainAxisAlignment.spaceBetween,
                      children: [
```

```
                        children: [
                          Text(
                            passengerName,
                            style: const TextStyle(
                              fontSize: 18,
                              color: Colors.white,
                            ), // TextStyle
                          ), // Text
                          ElevatedButton(
                            style: ElevatedButton.styleFrom(
                              backgroundColor: Colors.red,
                            ),
                            onPressed: () {
                              removePassenger(rideID, passenger);
                              setState(() {
                                passengers.remove(passenger);
                              });
                              if (context.mounted) {
                                ScaffoldMessenger.of(
                                  context,
                                ).showSnackBar(
                                  SnackBar(
                                    content: Text(
                                      '$passengerName has been removed from passengers',
                                    ), // Text
                                  ), // SnackBar
                                );
                              }
                            },
                            child: const Text(
                              'Delete Passenger',
                              style: TextStyle(color: Colors.white),
                            ), // Text
                          ), // ElevatedButton
                        ],
                      ); // Row
                    } else {
                      return const Text(
                        'No data available',
                        style: TextStyle(fontSize: 18),
                      ); // Text
                    }
                  },
                ), // FutureBuilder
              ],
            ],
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                ElevatedButton(
```

```
345                     style: ElevatedButton.styleFrom(
346                       backgroundColor: Colors.red,
347                     ),
348                     onPressed: () {
349                       showDialog(
350                         context: context,
351                         builder: (BuildContext context) {
352                           return AlertDialog(
353                             title: const Text('Are you sure?'),
354                             content: const Text(
355                               'Do you really want to cancel this ride?',
356                             ), // Text
357                             actions: [
358                               TextButton(
359                                 onPressed: () {
360                                   Navigator.of(
361                                     context,
362                                   ).pop(); // Close the dialog
363                                 },
364                                 child: const Text('No'),
365                               ), // TextButton
366                               TextButton(
367                                 onPressed: () {
368                                   deleteRide(rideID);
369                                   Navigator.of(context).pop();
370                                   Navigator.of(context).pop();
371                                   if (context.mounted) {
372                                     ScaffoldMessenger.of(
373                                       context,
374                                     ).showSnackBar(
375                                       const SnackBar(
376                                         content: Text(
377                                           'Ride cancelled successfully',
378                                         ), // Text
379                                       ), // SnackBar
380                                     );
381                                   }
382                                 },
383                                 child: const Text('Yes'),
384                               ), // TextButton
385                             ],
386                           ); // AlertDialog
387                         },
388                       );
389                     },
390                     child: const Text(
391                       'Cancel Ride',
392                       style: TextStyle(color: Colors.white),
393                     ), // Text
394                   ), // ElevatedButton
```



```
                        child: const Text(
                          'Cancel Ride',
                          style: TextStyle(color: Colors.white),
                        ), // Text
                      ), // ElevatedButton
                      ElevatedButton(
                        style: ElevatedButton.styleFrom(
                          backgroundColor: Colors.green,
                        ),
                        onPressed: () {
                          completeRide(rideID);
                          if (context.mounted) {
                            ScaffoldMessenger.of(context).showSnackBar(
                              const SnackBar(
                                content: Text('Ride marked as completed'),
                              ), // SnackBar
                            );
                          }
                        },
                        child: const Text(
                          'Complete Ride',
                          style: TextStyle(color: Colors.white),
                        ), // Text
                      ), // ElevatedButton
                    ],
                  ), // Row
                ],
              ), // Column
            ), // Padding
          ), // Card
        ],
      ), // Column
    ), // Container
  ); // Scaffold
 }
}

```

ΠΑΡΑΡΤΗΜΑ Ε : Σύστημα Συνομιλιών (Chat)

```dart
import 'package:dipaecarpooling/utils/authentication.dart';
import 'package:dipaecarpooling/utils/chats.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';

class Chat {
  final String name;
  final String lastMessage;
  final String time;
  final String email;

  Chat({required this.name, required this.lastMessage, required this.time, required this.email});
}

// Function to retrieve chat data
Future<List<Chat>> getChats() async {
  var chats = <Chat>[];
  SharedPreferences prefs = await SharedPreferences.getInstance();
  String currentUserEmail = prefs.getString('email') ?? '';
  var allUserDetails = await getAllUsers();
  //print("All user details fetched: $allUserDetails");
  var chats_ = await getAllChats(currentUserEmail);
  //print("Chats fetched: $chats_");
  for (var chat in chats_) {
    String email = chat['userEmails'].firstWhere(
          (email) => email != currentUserEmail,
        );
    //print('trying to get user details for email: $email');
    var userDetails = await getUserDetailsByEmail(email);
    if (userDetails == null) {
      print('No user details found for email: $email');
    }
    String name = userDetails?['firstName'] + ' ' + (userDetails?['lastName'] ?? '') ?? 'Unknown';
    chats.add(
      Chat(
        name: name,
        lastMessage:
            chat['messages'][chat['messages'].length - 1]['message'] ?? '',
        time: chat['messages'][chat['messages'].length - 1]['timestamp'],
        email: email,
      ),
    );
  }

  return chats;
}

```

```dart
class ChatsScreen extends StatelessWidget {
  const ChatsScreen({Key? key}) : super(key: key);

String _formatTimestamp(BuildContext context, String? iso) {
  if (iso == null) return '';
  final dt = DateTime.tryParse(iso);
  if (dt == null) return '';
  final time = TimeOfDay.fromDateTime(dt.toLocal());
  return '${time.format(context)}';
}
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
      title: const Text('Chats', style: TextStyle(color: Colors.white)),
      backgroundColor: const Color.fromARGB(255, 50, 50, 50),
      iconTheme: const IconThemeData(color: Colors.white), // Makes back button white
      ), // AppBar
      backgroundColor: const Color.fromARGB(255, 30, 30, 30),
      body: FutureBuilder<List<Chat>>(
      future: getChats(),
        builder: (context, snapshot) {
          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
          } else if (snapshot.hasError) {
            return Center(child: Text('Error: ${snapshot.error}'));
          } else if (!snapshot.hasData || snapshot.data!.isEmpty) {
            return const Center(child: Text('No chats found.'));
          }
          final chats = snapshot.data!;
          return ListView.builder(
            itemCount: chats.length,
            itemBuilder: (context, index) {
              final chat = chats[index];
              return ListTile(
                leading: CircleAvatar(child: Text(chat.name[0])),
                title: Text(chat.name, style: TextStyle(color: Colors.white)),
                subtitle: Text(chat.lastMessage, style: TextStyle(color: Colors.white70)),
                trailing: Text(_formatTimestamp(context, chat.time), style: TextStyle(color: Colors.white70)),
                onTap: () async {
                  SharedPreferences prefs = await SharedPreferences.getInstance();
                  prefs.setString('chatEmail', chat.email);
                  Navigator.pushNamed(context, '/chat');
                },
              ); // ListTile
            },
          ); // ListView.builder
        },
      ), // FutureBuilder
    ); // Scaffold
  }
}
```

ΠΑΡΑΡΤΗΜΑ F : Σύστημα Αναφορών Προβλημάτων (Report Problem)

```dart
import 'package:dipaecarpooling/utils/problemreport.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';

class ReportProblemPage extends StatefulWidget {
  @override
  _ReportProblemPageState createState() => _ReportProblemPageState();
}

class _ReportProblemPageState extends State<ReportProblemPage> {
  final TextEditingController _controller = TextEditingController();

  void _submitReport() async {
    String problemText = _controller.text;
    if (problemText.isNotEmpty) {
      final prefs = await SharedPreferences.getInstance();
      String userEmail = prefs.getString('email') ?? '';
      createReport(text: problemText, userEmail: userEmail);
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(content: Text('Problem reported successfully!')),
      );
      _controller.clear();
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Report a Problem', style: TextStyle(color: Colors.white)),
        iconTheme: IconThemeData(color: Colors.white),
        backgroundColor: Color.fromRGBO(50, 50, 50, 1),
      ), // AppBar
      body: Container(
        color: Color.fromRGBO(30, 30, 30, 1),
        padding: EdgeInsets.all(16.0),
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.stretch,
          children: [
            TextField(
              controller: _controller,
              maxLines: 5,
              decoration: InputDecoration(
                hintText: 'Describe the problem...',
                filled: true,
                fillColor: Colors.white,
                border: OutlineInputBorder(
                  borderRadius: BorderRadius.circular(8.0),
                ), // OutlineInputBorder
              ), // InputDecoration
            ), // TextField
            SizedBox(height: 16),
```

```
53                  ElevatedButton(
54                    onPressed: _submitReport,
55                    child: Text('Submit'),
56                    style: ElevatedButton.styleFrom(
57                      backgroundColor: Colors.white,
58                      foregroundColor: Colors.black,
59                    ),
60                  ), // ElevatedButton
61                ],
62              ), // Column
63            ), // Container
64          ); // Scaffold
65        }
66      }
```

ΠΑΡΑΡΤΗΜΑ G : Αξιολόγηση Διαδρομής και Οδηγού (Ride Rate)

```dart
import 'package:dipaecarpooling/utils/authentication.dart';
import 'package:flutter/material.dart';
import 'package:shared_preferences/shared_preferences.dart';

class RateRideScreen extends StatefulWidget {
  const RateRideScreen({super.key});

  @override
  _RateRideScreenState createState() => _RateRideScreenState();
}

class _RateRideScreenState extends State<RateRideScreen> {
  int _rating = 0;
  String userEmail = '';
  String riderEmail = '';

  @override
  void initState() {
    super.initState();
    _loadUserEmail();
  }

  Future<void> _loadUserEmail() async {
    final prefs = await SharedPreferences.getInstance();
    setState(() {
      userEmail = prefs.getString('email') ?? '';
      riderEmail = prefs.getString('rateemail') ?? '';
    });
  }

  Widget _buildStar(int index) {
    final isSelected = index <= _rating;
    return IconButton(
      onPressed: () {
        setState(() {
          _rating = index;
        });
      },
      icon: Icon(
        isSelected ? Icons.star : Icons.star_border,
        color: isSelected ? Colors.amber : Colors.white,
        size: 36,
      ), // Icon
      tooltip: '$index star${index > 1 ? 's' : ''}',
    ); // IconButton
  }

  Future<void> _submitRating() async {
    if (_rating == 0) {
      ScaffoldMessenger.of(context).showSnackBar(
        const SnackBar(
          content: Text('Please select a rating before submitting.'),
        ), // SnackBar
      );
```

```
55          return;
56        }
57        await updateUserRatings(riderEmail, _rating);
58        ScaffoldMessenger.of(context).showSnackBar(
59          const SnackBar(content: Text('Rating submitted. Thank you!')),
60        );
61        setState(() {
62          _rating = 0;
63        });
64        Navigator.of(context).pop();
65      }
66
67      @override
68      Widget build(BuildContext context) {
69        return Scaffold(
70          appBar: AppBar(
71            title: const Text('Rate Ride', style: TextStyle(color: Colors.white)),
72            iconTheme: const IconThemeData(color: Colors.white),
73            centerTitle: true,
74            backgroundColor: const Color.fromRGBO(50, 50, 50, 1),
75          ), // AppBar
76          body: Container(
77            color: const Color.fromRGBO(30, 30, 30, 1),
78            padding: const EdgeInsets.all(16.0),
79            child: Column(
80              crossAxisAlignment: CrossAxisAlignment.start,
81              children: [
82                Card(
83                  color: const Color.fromRGBO(40, 40, 40, 1),
84                  child: Padding(
85                    padding: const EdgeInsets.all(16.0),
86                    child: Column(
87                      crossAxisAlignment: CrossAxisAlignment.start,
88                      children: [
89                        const Text(
90                          'How was your ride?',
91                          style: TextStyle(
92                            color: Colors.white,
93                            fontSize: 20,
94                            fontWeight: FontWeight.bold,
95                          ), // TextStyle
96                        ), // Text
97                        const SizedBox(height: 12),
98                        Row(
99                          mainAxisAlignment: MainAxisAlignment.start,
100                         children: List.generate(5, (i) => _buildStar(i + 1)),
101                       ), // Row
102                       const SizedBox(height: 8),
```

```
                    Text(
                      _rating == 0
                          ? 'No rating'
                          : 'You selected $_rating star${_rating > 1 ? 's' : ''}',
                      style: const TextStyle(color: Colors.white70),
                    ), // Text
                    const SizedBox(height: 16),
                    Row(
                      mainAxisAlignment: MainAxisAlignment.end,
                      children: [
                        ElevatedButton(
                          style: ElevatedButton.styleFrom(
                            backgroundColor: Colors.grey[700],
                          ),
                          onPressed: () {
                            setState(() {
                              _rating = 0;
                            });
                          },
                          child: const Text(
                            'Reset',
                            style: TextStyle(color: Colors.white),
                          ), // Text
                        ), // ElevatedButton
                        const SizedBox(width: 12),
                        ElevatedButton(
                          style: ElevatedButton.styleFrom(
                            backgroundColor: Colors.green,
                          ),
                          onPressed: _submitRating,
                          child: const Text(
                            'Submit',
                            style: TextStyle(color: Colors.white),
                          ), // Text
                        ), // ElevatedButton
                      ],
                    ), // Row
                  ],
                ), // Column
              ), // Padding
            ), // Card
          ],
        ), // Column
      ), // Container
    ); // Scaffold
  }
}
```

ΠΑΡΑΡΤΗΜΑ Η : Ρυθμίσεις Χρήστη (Settings)

```dart
import 'package:flutter/material.dart';

class SettingsPage extends StatelessWidget {
  const SettingsPage({super.key});

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text('Settings', style: TextStyle(color: Colors.white)),
        backgroundColor: const Color.fromRGBO(50, 50, 50, 1),
        iconTheme: const IconThemeData(color: Colors.white),
      ), // AppBar
      backgroundColor: const Color.fromRGBO(30, 30, 30, 1),
      body: ListView(
        children: const [
          ListTile(
            leading: Icon(Icons.delete, color: Colors.red),
            title: Text('Delete Account', style: TextStyle(color: Color.fromRGBO(255, 255, 255, 1))),
            trailing: Icon(Icons.arrow_forward_ios, color: Colors.white),
          ), // ListTile
        ],
      ), // ListView
    ); // Scaffold
  }
}

```