Faculty of Engineering and Technology

Department of Computer Science

COMP 4300 - Graduation Project

Section - A

Title of Project:

CarPal: Ridesharing App

Group Members with IDS:

BELAL HMEIDAT, 1202295

LOTFI HAJI, 1202064

BASEL ABU HAMED, 1202397

Supervisor:

NAEL QARAEEN

Section - B

Title of Project:

CarPal: Ridesharing App

Project No.:

9

Supervisor:

NAEL QARAEEN

Key Areas:

Transportation, Carpooling, Ride-sharing, Mobile App, Flutter, Firebase

Abstract

The project aims to develop a carpool mobile app that facilitates sharing rides between people going to the same destination. There are two parties involved, drivers and passengers, and the app will facilitate their ride transactions so they split the travel costs. The app will benefit both drivers and passengers, as drivers can cover gas fees for their trip or even make a small profit while the passengers get to their destination faster, in one commute, and when public transportation is not available. The project's goal is to reduce travel expenses, improve access to rides in areas with limited public transportation, and further grow hitchhiking community in Palestine. The application will be supporting both Android and iOS as we are going to be mainly using cross-platform technologies such as Flutter coupled with Firebase to manage the backed and the database.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Overview

When it comes to commuting from one place to another in Palestine, options can be narrowed down into two options, public and private transportation services and privately owned automobiles; people relying on the latter might carry with them colleagues, family, or friends or may be traveling alone. Here we will give an overview of the downsides involving these two norms of commute, briefly introduce the idea of our app, and how it fits among these two.

Public transportation is very dependent on in Palestine. Thousands of people use them every day. However, it suffers from a lot of problems that vary in severity from one area to another. The service is only available up to a certain time in the day. In addition to the long time the roads take, passengers waste a lot of time waiting for cars to come by or fill up with passengers. During rush hours when there are a lot of people and not so many buses available, people have to compete to get into a car. Moreover, and while not a big deal, it can be a hustle to find where these buses are stationed in each city/ town, even costing another transportation to get to them. On the other hand, there are a lot of people who use their cars every day to get to their jobs. They can pick up a friend or relative along their way but that's not always the case as many will go empty. This method can provide individuals with flexibility and freedom, and save them the wait time. However, it can be very pricey given the road traffic situation here and the gas prices. Here we come to introduce our idea of organized yet very accessible carpooling, where anyone with a car can sign up for an account to offer to share their ride with people going to the same destination as them and split the cost of the trip together.

The implementation of this project revolves around a mobile app that allows people to book trips with other people driving to the same destination. The two parties interacting with the app are the passengers and the drivers. The app will facilitate scheduling a trip, booking it, paying for it, and rating it at the end of each trip. The app will feature an easy setup process for both the passengers and drivers to get started using it quickly. The app will let all users create an account while users who want to become drivers will need to undergo a verification process. After signing up, drivers can schedule trips whilst passengers can request trips for passing by drivers to accept and upon getting accepted, proceed to pay. The app will have intuitive features

involving suggesting matching trips, displaying live driver location on the map, tracking trip progress, user profiles, and the ability to provide ratings and feedback from both sides. The app will let users pay using easy digital payment methods available via local payment services e.g., PalPay.

We believe this project will help people with travel expenses, especially those who have to travel to work/ study frequently and would like to make up for travel expenses without sacrificing much of the freedom and independence of owning a private car. Similarly, people who rely on public transportation to commute regularly can avoid the waiting times and inconveniences of public transportation services while still paying the same affordable fee. This app will also greatly help people find a ride at times when public transportation services are lacking.

1.2 Aims and Objectives

Aims

Developing a Mobile application that enables users to be able to have multiple options when traveling from one place to another and to give them what we hope is a better experience during their trips.

Objectives

To further develop carpooling and hitchhiking community in Palestine.

To develop an application that is easily accessible to a wide range of users.

To achieve a safe carpooling community that is supported by people.

To improve the traveling experience and choices for users both passengers and drivers.

1.3 Technologies

We are trying to use relevant technologies that are easy to implement, common, up-to-date, and compatible with our previous experiences.

Firebase

”Firebase is a set of backend cloud computing services and application development platforms provided by Google. It hosts databases, services, authentication and integration for a variety of

applications.” -Firebase Wiki [1]

Why Firebase? Our app needs to have a real-time connection and the ability to synchronize data throughout different devices. Firebase enables this in addition to a place to store our users and trips’ data and authenticate accounts using Firebase Firestore, Firestorage, and Firebase authentication services. in addition to having the Google Analytics support just by activating Firebase on the application, which will help with tracking customers events and get feedback on the application which will help us with scaling our project in the future, and by that we would be having a huge benefit especially when Firebase is developed and supported by Google. Here are some benefits of using Firebase:

Real Time Updates	Firebase uses data synchronization rather than typical HTTP requests. It allows all connected devices to receive immediate updates every time data changes. This helps with the real-time location and messaging through our app without having to dive into abstract networking.
Offline Data Availability	Firebase Realtime Database SDK persists the data to disk keeping it available offline. It syncs with the client device once connectivity is reestablished.
Ease of Access	The Firebase Realtime Database can be accessed directly from a mobile device or web browser. There’s no need for an application server. Moreover, it has security and data validation available through the Firebase Security Rules.
Google Integration	Google Analytics for Firebase will allow for tracking users’ trips through real-time and custom reporting. According to Google, ”Firebase provides unlimited free reporting on up to 500 distinct events. Just like the regular Google Analytics, Google Analytics for Firebase automatically tracks certain key events and user parameters straight out of the box, and allows you to define custom events that are important to your application.” -Firebase
Scalability	Firebase can scale well for real-time updates, which is crucial for a carpooling app with a potentially large user base.

Table 1.1: Relevant Firebase Features [2]

Figure 1.1: Firebase Logo [3]

Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop cross-platform applications for Android, iOS, Linux, Mac OS, Windows, Google Fuchsia, and the web from a single code base. [4]

Figure 1.2: Flutter Logo [5]

Why Flutter? With Flutter, we can save a lot of time and work by writing the code once knowing it will work on both of our target platforms iOS and Android. The main reason we are using Flutter over other cross-development frameworks such as React Native is due to our previous experience developing in Flutter. Flutter while it is still not as fast as native development, still outperforms React Native. Also since we are using Google Maps API for routes, and Google's Firebase as a Database, we expect Flutter will work more easily and reliably with them.

Java Spring Boot (no longer needed)

Java Spring Framework is a popular, open-source framework used for creating standalone, production-grade applications that run on Java Virtual Machine. [6]

Figure 1.3: Spring Logo [7]

Why Java Spring Boot Java Spring Boot is a reliable and robust choice for backend development that is widely used, in addition our team members have current and past experience working with java and Spring Boot, with addition to Spring Boot capabilities, we saw it as the perfect choice for our backend development and future scalability.

Transition to Firebase After doing further research and the following the initial development phase, we discovered that integrating Flutter with Firebase as a backend offers a more feature-rich experience. Firebase provides a comprehensive suit of tools and services, which are designed to work smoothly with Flutter. This integration allowed us to take better advantage of firebase's real-time database, authentication, Firebase storage, and other services without extensive configurations and with easier code implementation.

More over our decision to switch to Firebase was mostly driven by these factors:

1. Ease of Integration: Firebase's integration with Flutter is straightforward and well-supported, reducing complexity and development time for the backend, as both were developed are supported by google.

2. Feature-Rich Environement: Firebase offers built-in features like real-time data synchronization, user authentication, and more which are essential for our application and can be used in a more efficient way when directly implemented in flutter.

3. Scalability and Maintenance: Firebase's infrastructure is managed by Google, which

ensures high reliability and scalability, which allowed us to focus on building the application itself, rather than managing backend servers.

By choosing Firebase over Spring Boot, we allowed ourselves to take advantage of a modern backend-as-a-service platform (Firebase) that aligns well with the needs of our Flutter application.

Google Maps

Google Maps is the leading maps provider in the world and is used by a billion people a month [8] so people will feel at home using it in our app. We also find Google Maps easy to integrate with Flutter being also developed by Google and ut provided us with the features needed for our app such as custom markers, live location tracking, and trip planning.

Figure 1.4: Google Maps Logo [9]

This is not an inclusive list. For all the libraries and plugins we used in developing the app, see appendix A.

2 Literature Review and Similar Projects

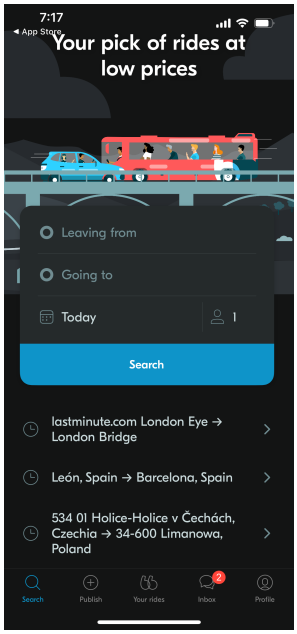
In this chapter, we give a brief overview of other similar projects that we took inspiration from and how will they compare to our project. We also list some research papers we found helpful to our project, how they are related, and how they shaped the way we are going to take certain approaches in our project.

2.1 Similar Ideas

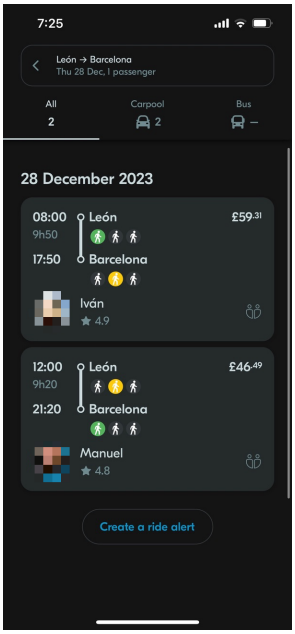
Here is how our app will compare to other similar projects. We have selected these projects each for a different reason. BlaBlaCar is the most similar app as it is mainly a carpooling app for all the people to share a trip and split the cost. Gett is a carpooling app that is widely known as it's active in our target area while inDrive is a very feature-rich carpooling app.

BlaBlaCar

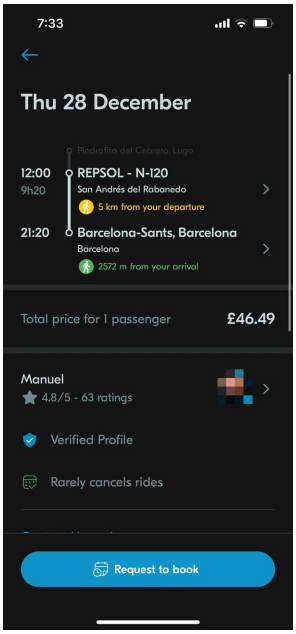
French carpooling company that lets everyone make trips and share the expenses with those who are going to the same destination. BlaBlaCar is available in 21 countries and has an easy setup and booking process. The app has a clean and easy-to-navigate interface that lets passengers and drivers book and schedule trips easily. It offers a unique aspect built solely around organized social ride-sharing to reduce trip costs. It doesn't contract drivers working for it which allows everyone to become a verified driver just by uploading some papers. The app makes it clear that its purpose is ride-sharing to reduce costs; it prohibits the use of its service as a means of bringing income to the driver. The company also offers its bus services from within the same app. Since their app is operational in European countries where hitchhiking and expense-sharing trips are widely accepted, the app has few restrictions and requirements for its users. [10] BlaBlaCar app is the closest to our idea and is its primary source of inspiration; therefore, our app shares many features and implementations with it such as checking the profile of other users to get to know who you will be riding with and setting specific rules for the trip. Additionally, our app will offer new features like on-the-way pickup, live map location services, Arabic language support, and a localized framework that fits into the roads situation in the West Bank.



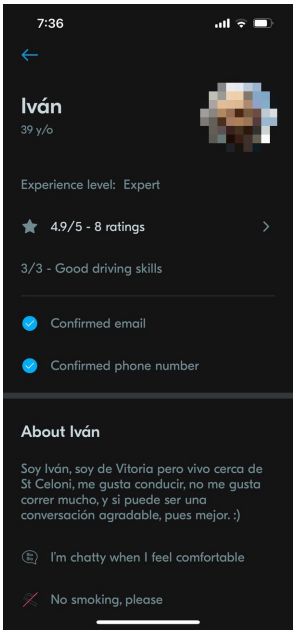
(a) Search Page



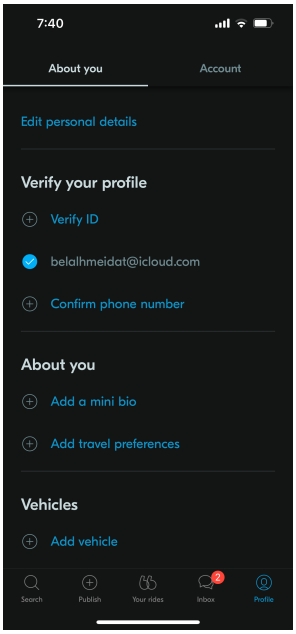
(b) Trip Search Results Screen



(c) Trip Detail Screen



(d) User Profile Screen

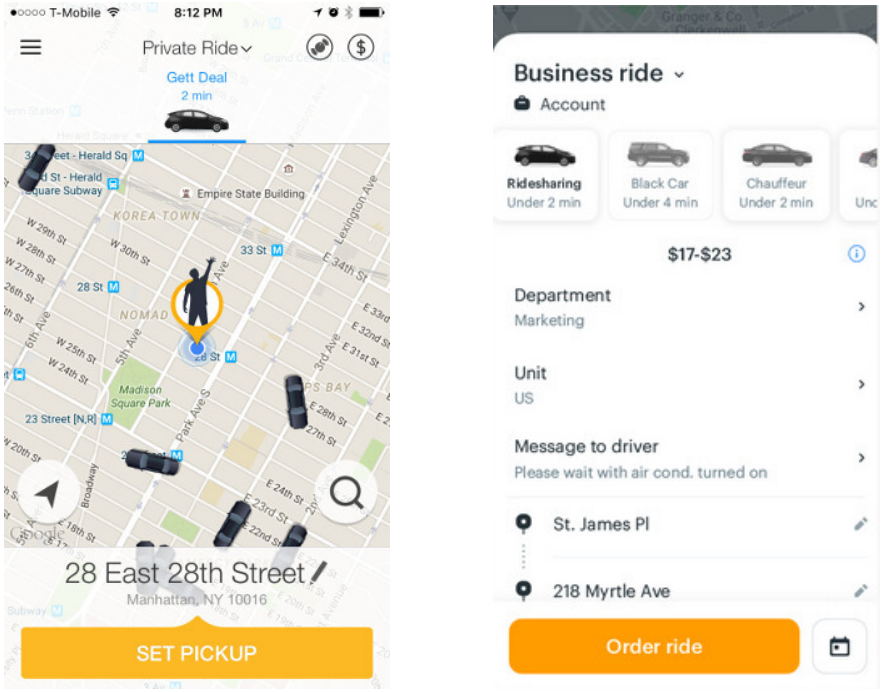


(e) Settings Screen

Figure 2.1: BlaBlaCar App Pages Overview [11]

Gett

Also called Gett Taxi, which is a B2B Ground Transportation Management (GTM) platform, which was designed to help people save a lot of time and effort as it offers a faster, easier, and cheaper way for people to find transportation from one place to another. [12, 13] Gett is available and operating in Palestine particularly inside pre-1948 borders. The app offers many quality features such as being able to see drivers on the map and utilizing it to choose exact pickup and drop-off locations. Moreover, the app allows users to choose the type of ride and car size they want: shared ride, chauffeur, SUVs, and more. While being fairly popular around, the company's business model is primarily providing cab services rather than ride-sharing, hence the name Gett Taxi. It can offer a better experience for passengers going on long trips more than public transportation services do but it doesn't have other noteworthy advantages over them.



(a) Ride Location on Map (b) Trip Detail Screen

Figure 2.2: Pictures from withing Gett Taxi App [14]

inDrive

International ride-hailing service and one of the most popular carpooling apps in the world. It sits as the second most downloaded ride-sharing and taxi app. [15] In addition to being a taxi alternative, inDrive is used for many services such as courier delivery, freight transport, and professional services such as home repairs and tutoring. inDrive appeals to its user base by allowing them to negotiate a price with different drivers and thus helping them find the price they're looking for. The app itself allows users to select a service and select locations on the map. It also shows nearby drivers on the map. It also gives users access to message or start a call with the driver from within the app.[16] Despite being very popular, inDrive is not available in Palestine. The business operates differently from BlaBlaCar and our app as it falls in line with the workflow of other famous carpooling apps such as Uber which is primarily a carpooling app that drivers rely on for bringing in income rather than merely sharing a ride. It also offers more general use cases such as professional maintenance services.

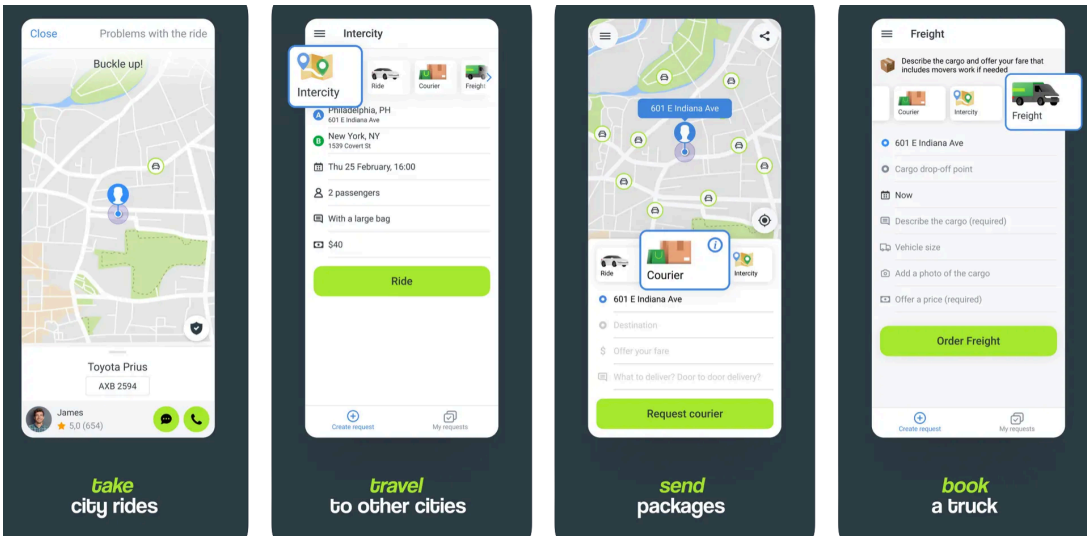


Figure 2.3: Screens showcasing inDrive app [16]

Here is how they compare to our app:

Table 2.1: Our app compared to similar other known apps.

Feature	inDrive	BlaBlaCar	Gett	Our App
Anyone can make a trip after verification (no contract)	✓	✓	✓	✓
Driver can schedule trips days prior	✓	✓		✓
Passengers can browse as a guest		✓		✓
Passengers can filter through drivers	✓			✓
Users can set rules visible on their profile		✓		✓
Cancellations allowed	✓	✓	✓	✓
Pickup Locations				
Pickup on the way				✓ (from stations)
List and map of trips to destination[1]				✓
Alternative routes				
Provide other trip suggestions to complete the trip*				✓
Provide public transportation station information to be able to complete the trip *				✓
Map integration				
Trip route on the map		✓	✓	✓ (limited)
Live location of driver[2]	✓		✓	✓
Comunication and Trip Sharing				
User profile with rules, cars, trip counting, and ratings	✓	✓		✓

Our app compared to similar other known apps (continued)

Feature	inDrive	BlaBlaCar	Gett	Our App
Users can see driver's rating and other passengers' rating in trip[3]Rating and feedback for both passengers and drivers		✓		✓

* Feature is not of high priority. We will try to implement this feature if time allows but don't promise to.

[1] The passengers can see a list of drivers heading to their destination sorted by certain criteria as well as a map showing the location of drivers in the list whose trips are ongoing.

[2] Shows live location of the driver when the trip is started. This only works if driver enabled location sharing and location time is up to date (not older than a certain duration).

[3] Users can see the name and rating of a passenger who is on the same trip when browsing trips. They can visit their profiles after booking the trip.

2.2 Relevant Research

Carpooling, Who, Why, and How

In his article about who and why people do carpooling, ROGER F. TEAL [17] outlines some of the factors that make carpooling appealing to people or not. He mentions numerous statistics and research done combined with comprehensive analysis on the topic. While the article is based on the U.S. environment, there are general conclusions that can be drawn from the paper that serve as useful indicators to guide us in how we approach our app. One of which is his conclusion that socio-economic factors have little effect on one's propensity to carpool. This conclusion has been echoed multiple times by other researches he mentioned in his paper and ones that we came across in our further readings as well. That being said, he later states that people of lower income are more willing to carpool. The paper also suggests that women are more likely to carpool than men. He also brings up the agreement among researchers he mentions in his paper that carpoolers travel much bigger distances to work than people who commute alone. This might be the case for Palestine as well since examples of people carpooling with others going to their work in further cities are fairly common. This information is helpful since our app relies on stations of pickup. The range these stations cover should be larger if people are more inclined to carpool to far places. This will streamline the station selection to fewer stations that

people can use reliably instead of creating many other substations that people would more likely use other traditional transport methods to get to. He also finds that the percentage of unrelated people who carpool together is larger than those who are related. This is to be expected as the chance of family members working in the same place or at the same time is unlikely. However, in Palestine, families are fairly larger than American families; so the difference between these two percentages might be smaller. There is still a sizeable percentage of people who carpool with unrelated people and could benefit from our app to find them. Moreover, Dr. Teal finds that the number of unrelated people carpooling is higher than that of internal carpooling, where people carpool with their relatives, by 2.25 to 2.63 people respectively. He finds that internal carpooling is incentivized by vehicle shortage whereas external ones are motivated primarily by cost burden.[17] Since our app is mainly focused on external - non-related - carpooling, we will have to focus on ways to make the app cost-cutting, maybe by limiting the price people set or suggest a recommended price tag that is proportional to the travel cost.

What Motivates Carpooling

In their paper about what encourages people to carpool [18], Jun Guan Neoh, Maxwell Chipulu, and Alasdair Marshall dive into the factors that affect people's willingness to carpool. They reviewed 908 papers from previous studies and filtered them down to 19 eligible articles to perform a meta-analysis on them. They summarize the literature on the carpooling field as "rich in studies but doubts remain about the generalisability of the results, making it difficult for policy makers to translate the findings into practice." Their research splits the factors affecting carpooling into internal individual and judgemental factors and external environmental and situational factors. Like Teal (1984) [17] and many other publications on the topic, this research agrees that socio-demographics don't strongly influence carpooling. One key point drawn from their research regarding socio-demographic factors and one which agrees with Teal (1984) [17] is that women carpool more than women do. We don't know if this translates well to Palestine but we aim that our app will be usable for both men and women easily by taking some measures that are friendly with the gender rules in Palestine such as giving female users the ability to filter female drivers only. In terms of situational factors, the research also confirms the previously discussed findings that carpooling favors longer distances. It also claims that the inconvenience of waiting for other carpooling members can be a major factor in pushing away people from carpooling. We see that this issue might be exacerbated in our app since we are allowing drivers

to pick up people on the way. We need to bring countermeasures that will make pickup easier and faster. One such measure is limiting the number of people a person can pick up. In addition to the ability of drivers and passengers to communicate and decide on a pickup location, we also discussed utilizing the map so passengers can send their exact pickup location from the start when requesting the trip to the driver who can, in return, agree or send a new waiting location for the passenger. As for judgemental factors, the research notes that psychological factors play a major role in making carpooling decisions. More so than socio-demographic factors. People feel more comfortable carpooling with their relatives and co-workers than when they are carpooling with strangers. Commuter privacy is also a big factor as carpoolers feel a sense of giving up some privacy and personal space when carpooling. However, since public transportation is very popular in Palestine, we feel carpooling will be just as acceptable in this regard as it offers similar, if not better, privacy levels. The study also points out that a sense of control can encourage people to carpool, we believe we can achieve this in our app by offering the option to find trips that agree with the user's set rules. We expect people's willingness to carpool can be negatively affected by their inclination to socialize. We are trying to mitigate the effect of this by designating a rule for chatting and socializing. We can also benefit from the fact that our app is oriented around picking up people along the way, which in turn, encourages shy people to participate knowing that other people will be on the trip with them providing them similar environment to familiar public transportation. Pollution is also a factor that encourages people to carpool according to this research but while reducing carbon footprints is one of our goals, we don't believe it is a major incentive for carpoolers in Palestine. Nonetheless, we see that traffic plays a similar role in encouraging people but we don't know to what extent since the traffic and road situation in Palestine can be abnormal most of the time. Moreover, the research results point out the important and the not-so-significant factors that affect carpooling. It finds that transportation costs, the number of employees in the carpooler's workplace, and finding potential partners have the biggest effect. Some other less important factors include the availability of carpooling lanes, fixed working schedules, and urban areas. Factors such as age, marital status, parking costs, parking availability, and being a university student, have negligible effects on carpooling. [18] Overall, we believe this research gives very good insight for us to decide on how we implement certain aspects of our app and will try to make good use of it in this project and future works to come.

Carpooling Challenges

In their research paper on carpooling and its economic implications for drivers [19], Hai-Jun Huang, Hai Yang, and Michael G.H. Bell address the challenges associated with carpooling, including inconvenience for those with flexible hours, increased travel time, and concerns about privacy and independence. The study primarily focuses on the economic effects of carpooling, employing two main models: deterministic and stochastic.

Where deterministic mode, explores passengers' behavior and thoughts about carpooling, and driving alone modes under various scenarios, including no-tool equilibrium and social conditions. On the other hand, the stochastic model takes the path of logit-based modeling, considering different factors such as fuel cost, assembly cost, value of time, and traffic congestion.

The research adopts three main approaches to studying carpooling which are: estimating ride-sharing potential, predicting ride-sharing demand, and considering demand and supply effects. Several numerical examples are shown in the paper that demonstrate the impact of carpooling on car trips for users, which indicates a decrease in the overall cost for both passengers and drivers. To conclude, this research provides valuable insights into the economic aspects of carpooling, showing users' behavior and preferences, and offering a good understanding of the challenges and benefits that this mode of transportation can bring with it. [19]

3 Approach

3.1 Methodology

To ensure an effective development process, we have adopted the Agile development methodology [20] as the software development process throughout our journey.

Where we went through our project using iterative cycles, which allowed for flexible development process.

Following the selection of the software development process, our intial step was conducting a survey in which we gathered data from what we believe could be potential users. This helped us understand user expectations and desires for the application. Additionally we referenced the coparative analysis of existing carpooling applications which helped us identify areas of improvement and features that would make the application stand out.

After understanding the user needs and market landscape, we dived into the system analysis phase. In this phase, we defined the system requirements and use cases, which helped us understand the system's functionality and how users will interact with it. in which then we moved to technologies selection, where we chose the technologies that best fit our project requirements and our team's expertise. As in this phase we decided to migrate from using Java Spring Boot to Firebase as our backend, with Flutter being kept as our front end technology.

As we progressed, we made use of the existing system requirements both functional and non functional, in addition to referencing the diagrams, such as use case diagram. which helped us to align our development progress with a visually mapped system architecture and functionality.

During the development phase we began by establishing the database and connecting it with the flutter application, followed by implementing user interfaces with their respective backend functionalities in parallel.while ensuring a robust and scalable architecture that can be easily maintained and extended in the future.

Throughout the semester, we followed the Agile methodology, where each phase of the development was planned, implemented and tested in iterations, which allowed us to adapt to changes and feedback quickly, ensuring a robust and reliable development process.

This methodology has proven to be effective in ensuring a smooth and efficient development process, allowing us to deliver a high-quality product that meets user expectations and requirements within the specified time frame.

3.2 Resources

West Bank Cities Roads Dataset

A dataset of all the roads commonly used to commute from one Palestinian town or city to another in the West Bank. This will be used to define fixed stations that allow for passengers to be picked up along the way from towns and cities that drivers normally pass by to arrive at their destination. There is no already made database that we know of as of the writing of this draft, 08/02/2024. We are to ask the responsible parties for one or start collecting it ourselves. Due to the narrow time window of the project, the scope of operational routes, or the routes the app will work on, is reduced to a smaller area to use as proof of concept and a test case for development; we chose the route of Qalqilya - Ramallah for this, see the figure 3.1 showcasing possible routes and possible stations in between. We aim to scale the app's operational scope from this to the whole of the West Bank wherever possible.

Figure 3.1: Test Case Route[1]

[1]Orange denotes a branch.

Public Opinion Questionnaire

A questionnaire to get public opinion on the idea. We assessed people's acceptance of the idea, and whether they will partake in it, we took a look into their concerns and asked them for inputs and suggestions, which we took into consideration throughout development.

3.3 Considerations

User Privacy Consideration

We will make sure the app respects user privacy. The app will be asking for user permission to access all needed device sensors and data such as camera access, location, calendar, and contacts if needed. It will not partake in collecting user data except for their necessary information such as phone number, real name, and payment method information as well as driver information that are necessary to make our app function optimally. We will maintain the secrecy of this data and store it in a secure encrypted database. An in-app chat to allow passengers an easy channel with the driver is discussed. In case it is implemented we do not promise utilizing an end to end encrypted messaging. Lastly, users will be allowed the option to delete their accounts from our database at any time.

Effect on Public Transportation Sector Consideration

Our app is not meant to replace the public transportation sector and is not intended to be used as a main source of income for drivers. We have taken into consideration measures to combat that such as suspending accounts that do partake in that.

Legal Consideration

In case of a reported felony or abuse committed by the driver or the passenger, we are willing to provide the driver's information to the legal authorities upon request. Access to users' (passengers and drivers) information can be acquired in case required by law. We are **not** to be held accountable for any of the drivers and passengers' actions.

3.4 Work Schedule

The implementation work plan clearly outlines the different stages needed to be completed to finish the project. Figure 3.2 shows the work plan. The work plan starts with setting up the Firebase database with the collected sample routes, then moves to work on the trip scheduling logic and UI, then the trip booking logic and UI, then integrating the map and location services, then adding user profiles, user rules, trip filters, feedback, and rating features, then a polishing and fine-tuning phase. Going by the times we designated for each stage, we expect to finish all stages in 15 weeks from 01/03/2024 to 15/06/2024.

Figure 3.2: Implementation Work Plan

4 System Analysis

4.1 Functional Requirements

Below are the details of all the user functional requirements (UR) and system requirements (SR) accompanying them.

UR1. Users shall be able to log in and log out after creating an account.

SR1.1 The system shall allow users to sign up for the same type of account through the same page and app.

SR1.2 The system shall enable both drivers and passengers to log in, with the ability to stay logged in till logout is requested or session is expired.

UR2. User shall have a profile where they can edit personal information, add a set of rules for their trips, and edit driver details if applicable for them.

SR2.1 The system should allow the user to add/edit their personal information.

SR2.2 The system should allow users to choose from a variety of rules to add to their trip preferences, such as smoking, music, etc.

UR3. Users shall search for trips by selecting the departure location, destination, and date for a trip as a member/guest.

SR3.1 The system shall look for drivers going to the selected destination on the same date and passing by the user's departure location.

SR3.2 The system shall display relevant trip details, including driver information, route, and rules, and expected arrival time.

SR3.3 The system shall sort the drivers list depending on the time and route taken whether they respect the user's set rules or not.

SR3.4 The system shall allow users to browse trips without signing in (as guests).

SR3.5 The system shall display a list of trips based on the user's given requirements.

SR3.6 The system shall provide locations of nearby drivers making the requested trip on a map mirroring the information provided in the list.

UR4. Users should be able to become drivers after undergoing a verification process.

SR4.1 System shall allow users with cars willing to collect passengers along their way to undergo a verification process to become verified drivers.

SR4.2 The system shall notify the Users if they were accepted or not.

UR5. Admins shall be notified about new applying Drivers.

SR5.1 The system shall send a notification to the responsible department about newly applied drivers

SR5.2 The responsible department shall evaluate the user's request and approve or deny their request.

UR6. Users shall be able to file complaints and reports which shall be received by the responsible department to resolve issues or provide support.

SR6.1 The system shall notify Admins about drivers' being reported more than 3 times.

SR6.2 The admins can review the complaints and take action if they are valid complaints.

UR7. Verified drivers shall be allowed to schedule trips.

SR7.1 The system shall allow drivers to schedule trips by selecting the departure and destination locations and time of the trip.

SR7.2 The system shall ask the driver to specify available seats and luggage rooms.

SR7.3 The system should allow the driver to select the route throughout the trip.

UR8. Users shall be able to report other users (drivers/ passengers) for any complaints.

SR8.1 The system shall allow passengers to report drivers

SR8.2 The system shall allow drivers to report passengers.

SR8.3 The system shall allow passengers to report each other if they were on the same trip.

UR9. Users shall be able to give feedback and ratings to different users.

SR9.1 The system shall allow passengers to provide feedback and ratings for their driver.

SR9.2 The system shall allow drivers to provide feedback and rating for their passengers.

SR9.3 The system shall allow passengers to provide feedback and ratings to other passengers on the same trip.

UR10. Passengers shall be able to book trips.

SR10.1 The system shall allow only signed-in members to book trips.

SR10.2 The system shall display relevant trip details, including driver information, route, and set rules.

SR10.3 The system shall notify the driver of this booking.

SR10.4 The system should wait for the driver to accept the request before confirming the booking.

SR10.5 The system shall request the passenger to pay for the trip.

UR11. Passengers shall be able to pay for the trip.

SR11.1 The system shall allow passengers to pay through different online payment methods.

SR11.2The system shall complete the payment securely.

SR11.3 The system shall confirm the booking for the driver/passenger.

SR11.4 The system shall complete the payment for the driver when trip end confirmation is done.

UR12. Users shall be able to cancel trips (Both Drivers and passengers).

SR12.1 The system shall allow users to cancel the trip.

SR12.2 The system shall start the refund process for this trip.

SR12.3 The system shall notify the driver/passengers when cancellation is completed.

SR12.4 The system shall calculate the refund based on the time of the cancellation.

SR12.5 The system should allow for a full refund if the driver is too late from his planned arrival time.

UR13. The passenger shall be notified sometime when the driver starts the trip.

UR14. Users (driver and passenger) should confirm that the trip is finished and will have the option to provide feedback when the trip is complete.

SR14.1 The system should receive confirmation from both the driver and passenger at the end of the trip.

SR14.2 The system will prompt the driver to rate the passenger.

SR14.3 The system shall prompt the passenger to rate the trip, the driver, and other passengers.

UR15. The user shall be able to change their account's settings e.g. change email, password, and privacy settings.

4.2 Non-Functional Requirements

1. The app must submit to the average acceptable response time for mobile apps which is 2-3 seconds.
2. The app must be reliable, this means that it must work correctly at 97%, and have a 3% downtime. This means 12 hours of downtime per year.
3. The app must submit to the industry standards security practices to protect collected and analyzed data.
4. The app must be designed with well-documented code to allow future maintenance and enhancements.
5. The app must be compatible with widely used Android and iOS versions, namely Android 10, iOS 13, and later.
6. The app must feature an intuitive UI that is easy to use and navigate.
7. The app must be scalable such as it can support the ability to scale the program in the future to keep up with rising demand.
8. The app must be recoverable, as the database will be backed up with scheduled plans.

4.3 Use Cases

Actors Description

- Admin: This actor is responsible for verifying drivers, and reviewing reports/complaints from app users.
- Driver: This actor is responsible for scheduling trips, accepting passengers to his trip, and operating the trip.
- Passenger: This actor can search for trips, and request to join trips that are available on the application.
- Guest User: This actor can download the application, but is limited to only searching for trips and seeing them (can not request to join trips).

External Actors Description

- Notification Manager: This service handles notifying actors about events such as informing admins of a newly filed report/ complaint from users, notifying admins about new requests from users wishing to become drivers, informing drivers of new ride requests from passengers, and reminding drivers and passengers of their upcoming trips.
- Map Services: This service is fired up whenever a new trip is added with its final destination located on the map, passenger checks to see a driver's live location on the map, or when passengers choose a point on the map to be picked up from and the driver approves or sends back a different point for pick up.
- Payment Services: This service is responsible for facilitating payments between passengers and drivers. It is used whenever a payment transaction is made.

Use Cases Description

This is a description of the use cases shown in 4.1.

1. Login/ Registration

   Actors: Driver, passenger

   The page users are welcomed the first time they launch the app. It also offers an option to browse trips in guest mode.

2. Login/Registration Verification

   Actors: Driver, passenger

   When a user signs up for a new account, they are required to verify their phone number and email. When they sign in, a verification code is sent to their confirmed phone number

3. Create an Account

   Actors: Unregistered users

   An unregistered user can create an account by entering the needed information: email, phone number, and password.

4. Login

   Actors: Driver, passenger

   A user can log in by entering his email and password to the login page.

5. Logout

   Actors: Driver, passenger

   A registered user can log in by clicking the logout button on the main screen.

6. Become Verified

   Actors: Driver, Passenger

   Users can choose to upload a picture of their ID card to earn a verified account badge. This serves as a measure to increase other user's trust in them.

7. Show/Edit Profile

   Actors: Driver, passenger

   A user can see his profile details, and edit them from adding preferences to his trips to updating their name and profile picture. From here they can also apply to become drivers.

8. Apply to Become Driver

   Actors: passenger

   A passenger can apply to become a registered driver by uploading their car registration, and their driving license, which then is revised by the admins.

9. Notify driver about approval/decline for his application

   Actor: notification manager

   The notification manager should send a notification to the user to notify them about their driver's application being approved or declined by the admin.

10. Upload Driver Documents

    Actors: Passenger

    A driver needs to submit certain documents to prove eligible to start making trips. These documents are their driver's license and their car license. They are also required to take a photo of their face to undergo a verification process after they are sent to the admins.

11. Notify Admins about new Drivers

    Actors: Notification Manager

    The notification manager is responsible for notifying admins about new driver applications.

12. Unlock Driver Features

    Actors: admin

    When a user's submitted paper to become a driver is approved by the admin, the driver will unlock the ability to schedule trips for other users to book.

13. Notify Users of Approved or Decline Requests

    Actors: Admin, Notification Manager

    When a user's request to become a driver is examined by admin and approved or declined, the user will get a notification back telling him about the decision and the reasons behind refusal in case of one.

14. Add Payment method

    Actors: Passenger, Driver

    The User can choose his preferred payment method that is presented in the application for future payments.

15. Report/ Submit Complaint

    Actors: Passenger, Driver

    The user can report/submit a complaint about another user if he has recently been on a trip with them.

16. Notify About Upcoming Trips

    Actors: Notification Manager, Passenger, Driver

    The notification manager is responsible for reminding a passenger/driver about their upcoming trips before a reasonable amount of time.

17. Show Scheduled Trips

    Actors: Driver, passenger

    Drivers and passengers can see a page with their upcoming trips sorted by the time they are scheduled to start. They can also see their previous trips history.

18. Manage Account Settings

    Actors: Driver, passenger

    Driver and passengers can access their account settings through the app to manage settings like changing their email, password.

19. Search Trips

    Actors: passenger, Driver, Guest

    Drivers, passengers, and application guests can search for trips by entering the departure, destination, and time of the trip he is looking for.

20. View available trips

    Actors: Driver, passenger, guest

    Drivers, passengers, and application guests can view available trips after searching for them.

21. Apply Trip Filter/ Sorting if Any

    Actors: Driver, passenger, guest

    Driver, passengers, and guests can filter their trip search results based on different criteria, such as driver rating, rules they would like the trip to abide by, or the verified status of the drivers.

22. Show Chosen Trip Details

    Actors: Driver, Passenger, Guest

    Users searching for a trip can see more details about available trips, like driver information and rating, driver's rules, names of other passengers on the trip and their ratings, and access to the driver profile from there.

23. Schedule Trips

    Actors: Driver, Map Service

    The driver can schedule a trip by adding departure, destination, and time of the trip, preferred routes, and additional trip rules.

24. Set trip route

    Actors: Driver, Map Service

    The driver can select the trip route using predefined stations allowing passengers to be picked up along the way. If the driver chooses not to select the routes, their trip will show up only to passengers coming from the same starting destination and going to the same final destination.

25. Suggest New Stations

    Actors: Driver, Map Service

    The driver can suggest a new station to be added to the set of stations throughout a certain route.

26. Validate Trip

    Actors: Driver

    The system checks if the scheduled trip by the driver doesn't conflict with other trips they already set up.

27. Request to Join Trip

    Actors: Driver, passenger

    Driver and passenger can send a request to the trip driver to book a seat on the trip.

28. Validate Request

    Actors: passenger

    A user's request to join a trip is validated so that it does not conflict with other scheduled Trips.

29. Review Join Trip Request

    Actors: Driver

    The driver can see requests from passengers wanting to ride with them and accept or decline depending on the ratings of the passenger, and other factors such as the verified account status of the passenger and their location.

30. Notify Passenger of Driver Response to Trip Request

    Actors: Driver, Notification Manager

    When a driver accepts or declines a passenger's request to be picked up then the system notifies the passenger.

31. Send Payment to Driver for Trip

    Actors: payment service, Admin

    At the end of trip confirmation, The system processes the payment to the driver's account.

32. Receive Payment

    Actors: payment Service, Admin

    When a payment is made by a passenger the payment service transfers the money into a median pocket owned by the application admins which holds the payment until trip completion.

33. Confirm Trip Completion

    Actors: Driver, passenger

    At the end of the trip:

    The driver should ask and ensure that the passenger confirms that his trip has ended.

    The driver should confirm that the trip has ended when he reaches his destination.

34. Cancel a Reservation

    Actors: passenger

    passengers are allowed to cancel a trip reservation within a limited amount of time estimated xx hours, with a full refund, and a partial refund accounted for the remaining time of the trip after that.

35. View Accepted Trips

    Actors: Passenger

    Passengers can view the trips that they have been accepted to and decide which one they are going to attend.

36. Perform Payment

    Actors: Passenger, Payment Service

    Passenger can pay for the trip they have chosen to attend and their seat is booked on successful payment. Additionally, all previous requests sent to other drivers will be canceled.

37. Validate Payment

    Actors: Payment Service

The payment service is responsible for validating the payment and transferring the money from the passenger to the admin's account.

38. Cancel a Reservation

    Actors: Passenger

    The passenger is allowed to cancel an already booked trip or a trip request. If the trip is paid for, a refund is issued back to the passenger's payment method. The refunded amount will be deferred depending on how late the passenger requests to cancel.

39. Ask for Rating/ Feedback

    Actors: Passenger, Driver

    On the trip's end, both passengers and drivers are able to rate each other and provide feedback if they wish.

40. View trip history

    Actors: Passenger, Driver

    Passengers and drivers can see their trip history from their profile section, and if a user is registered as a driver he can choose to display trips he made or trips he has been part of.

Use Case Diagram

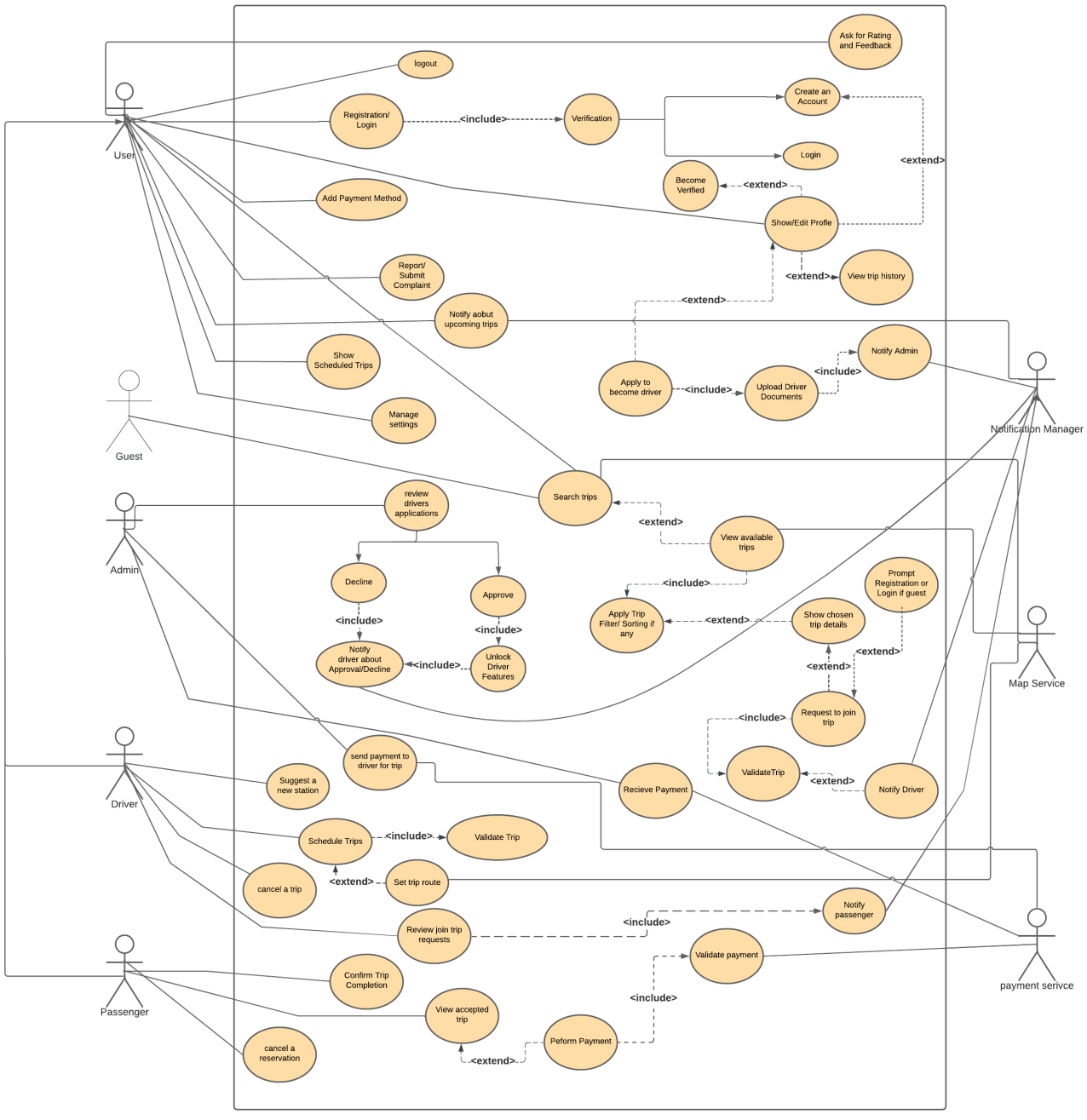


Figure 4.1: Use Case Diagram

4.4 Sequence, UML, State, and Deployment Diagrams

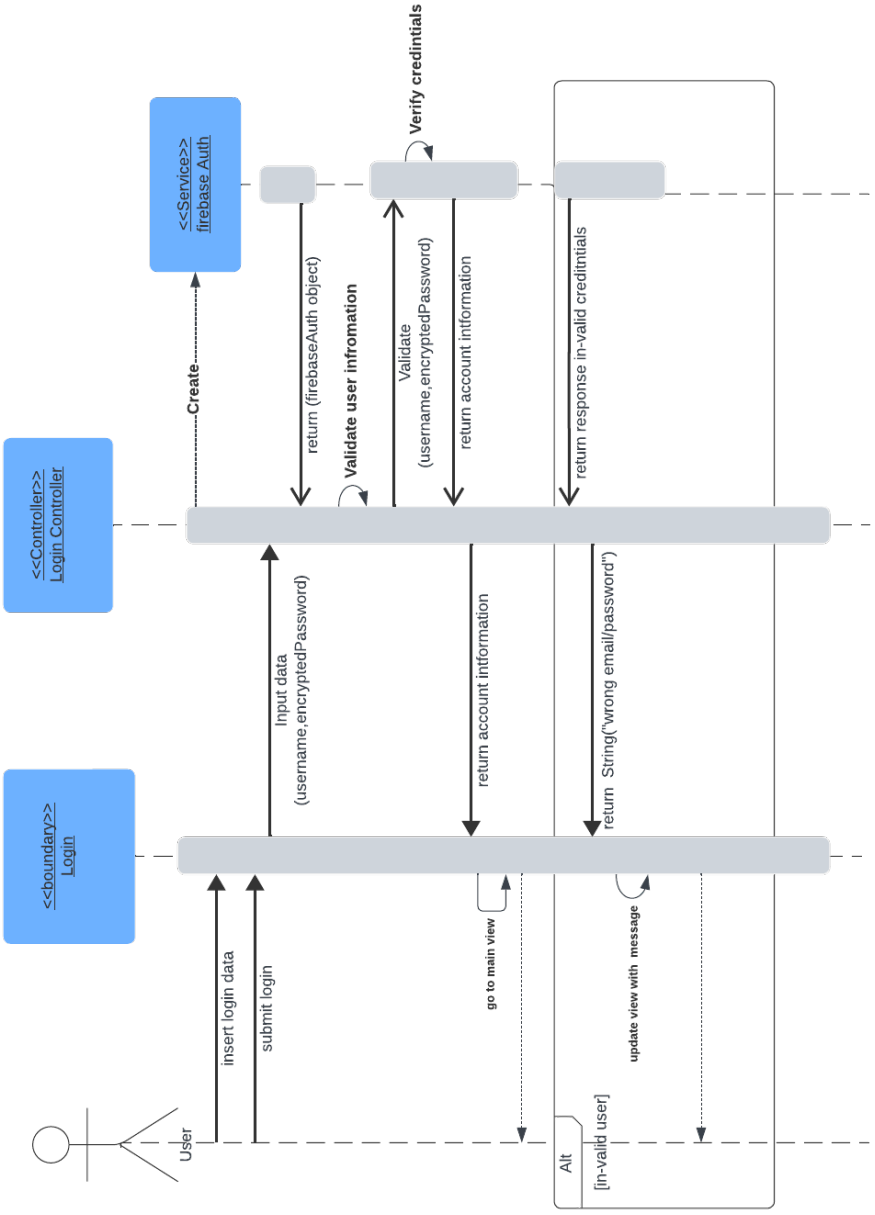


Figure 4.2: Login Workflow

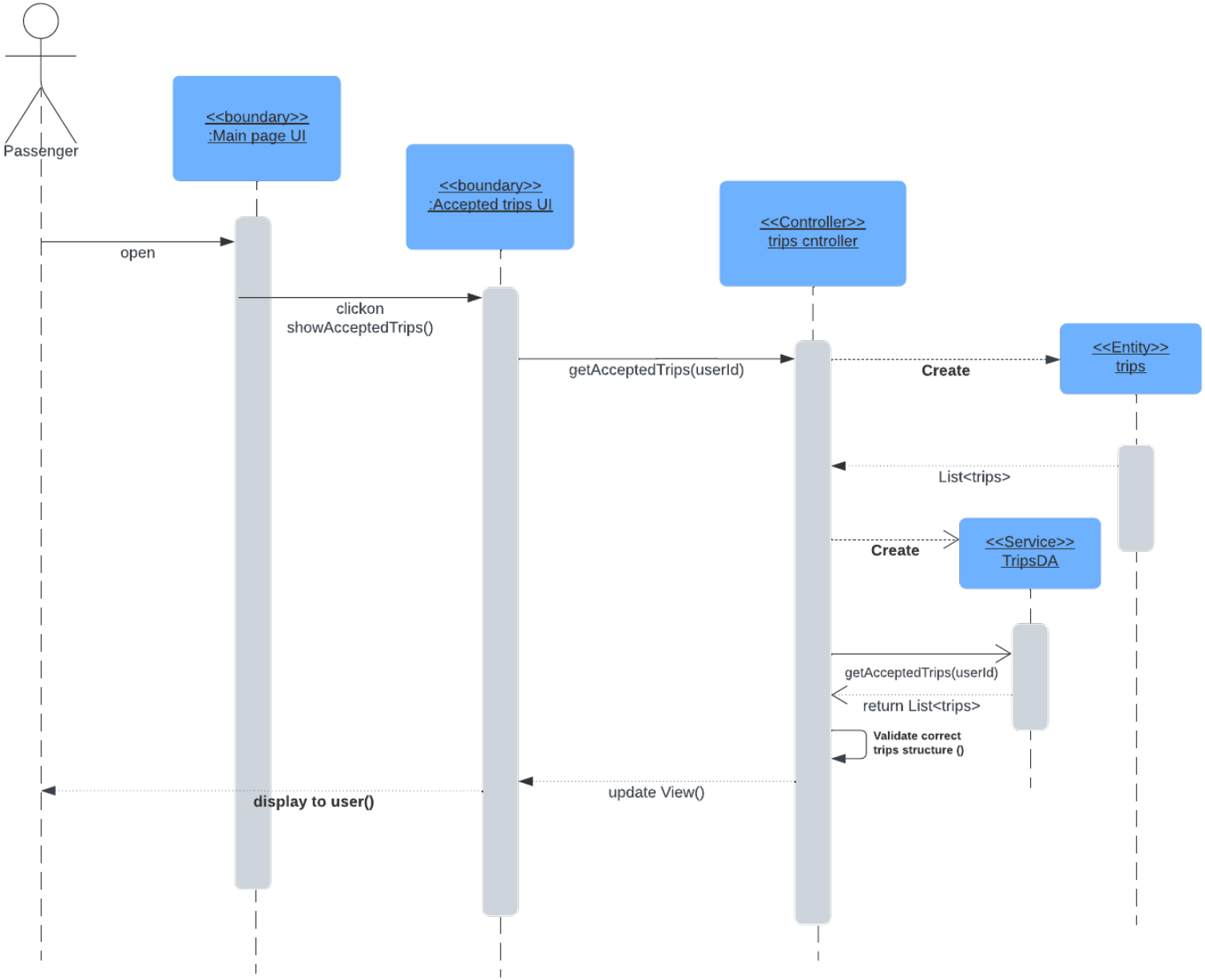


Figure 4.3: Showing Passenger's Accepted Trips Action

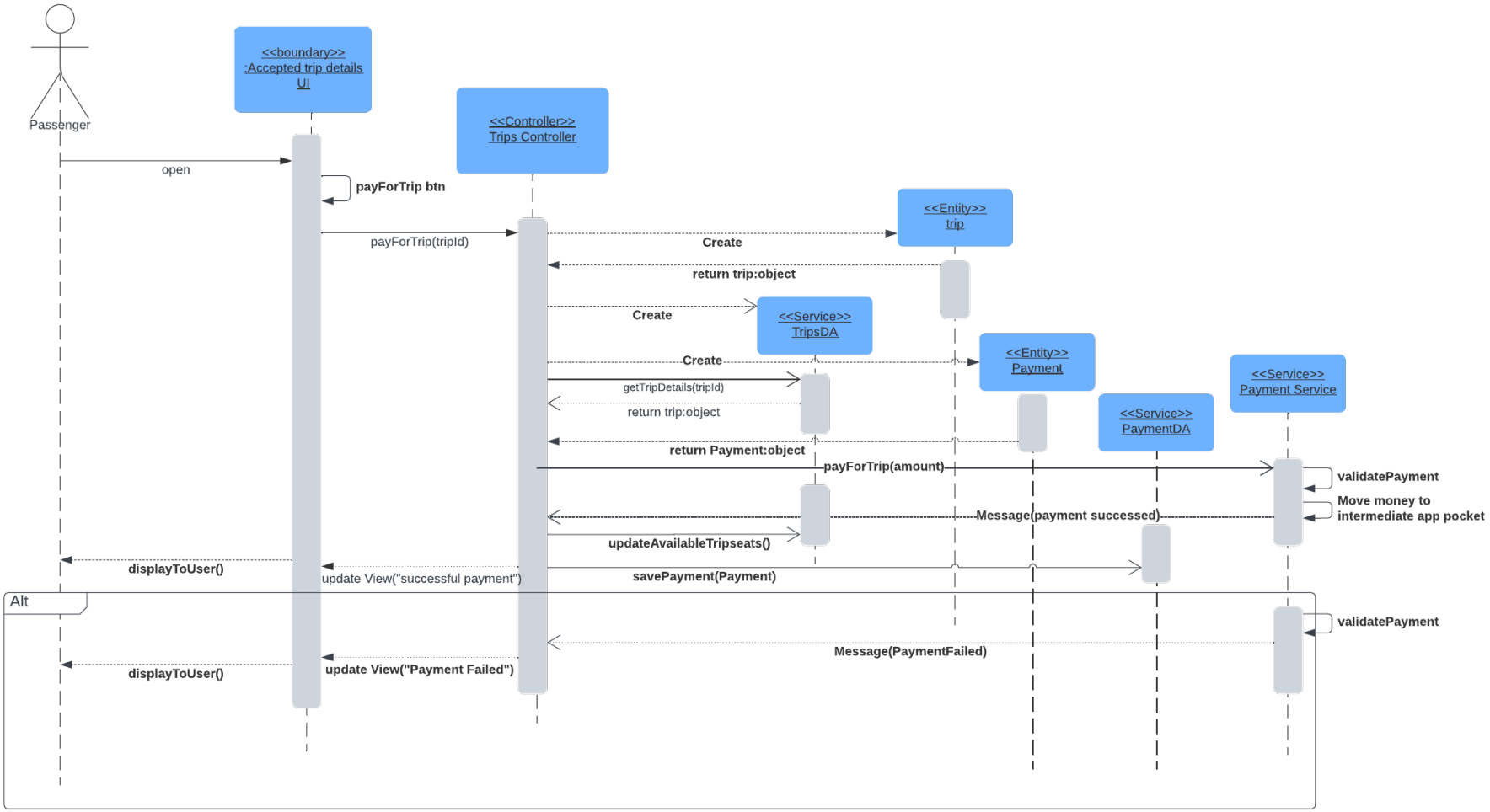


Figure 4.4: Passenger Payment Workflow

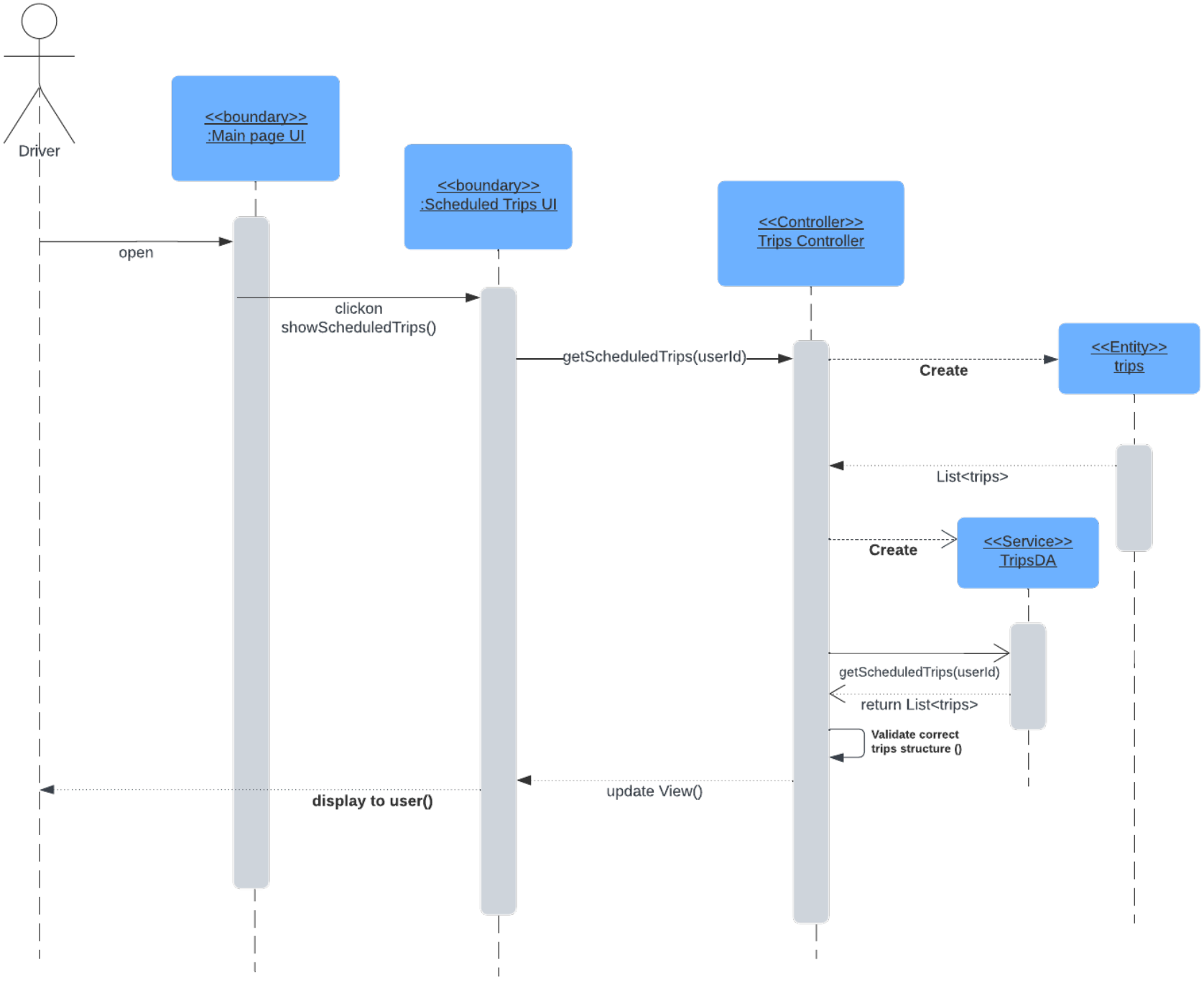


Figure 4.5: Showing Driver's Scheduled Trips Action

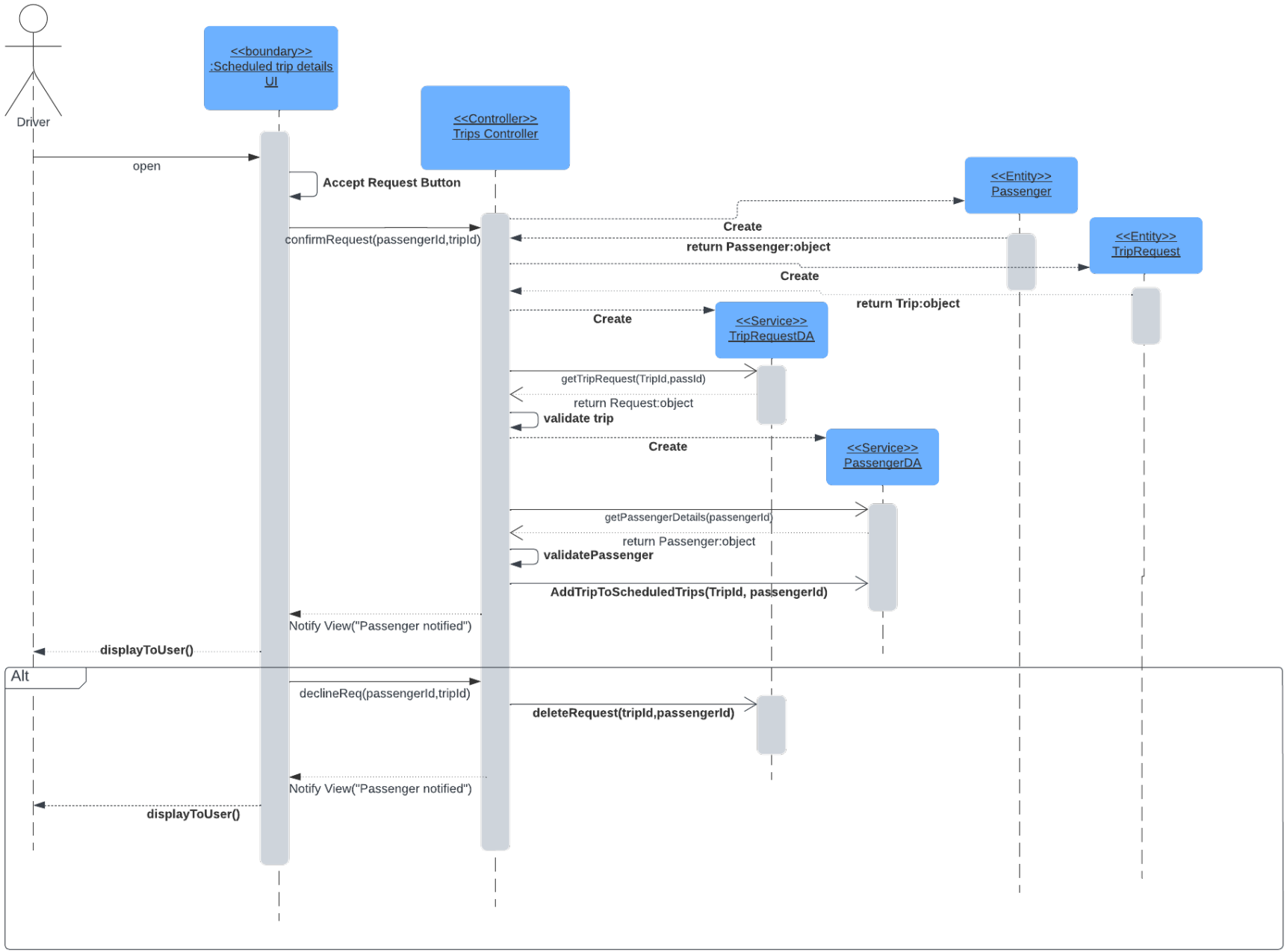


Figure 4.6: Passenger Request Workflow

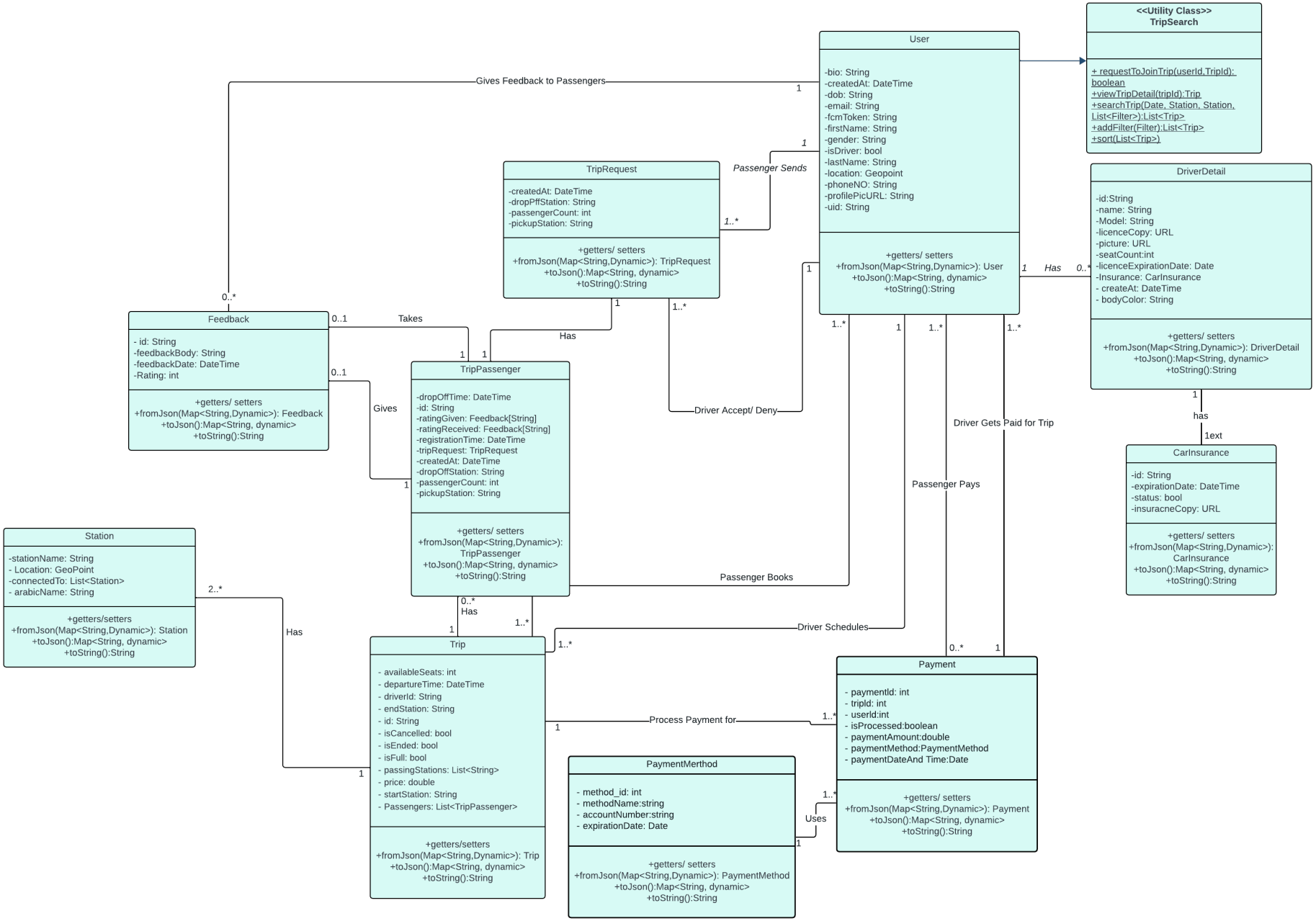


Figure 4.7: UML Class Diagram

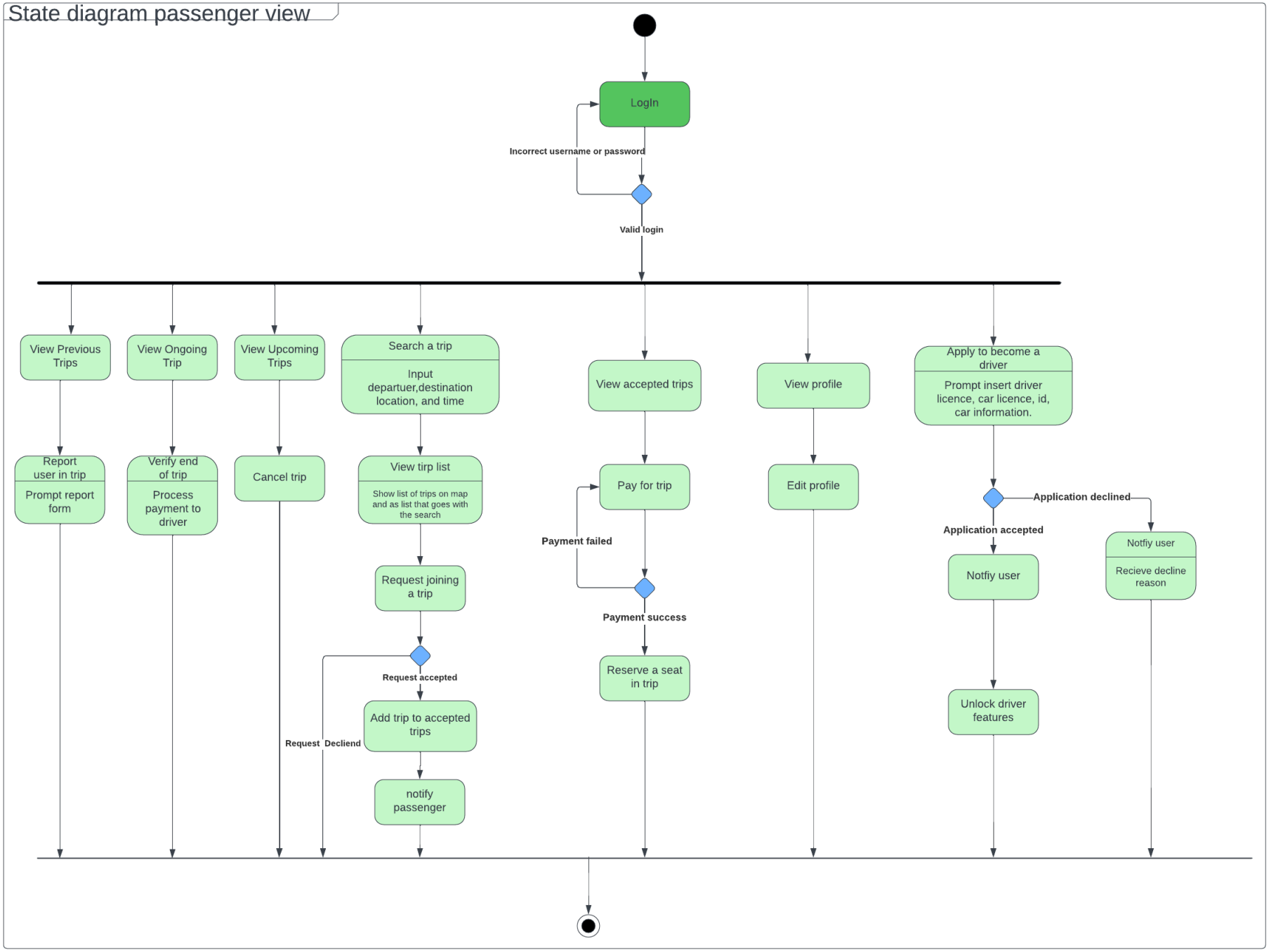


Figure 4.8: State Diagram for Passenger

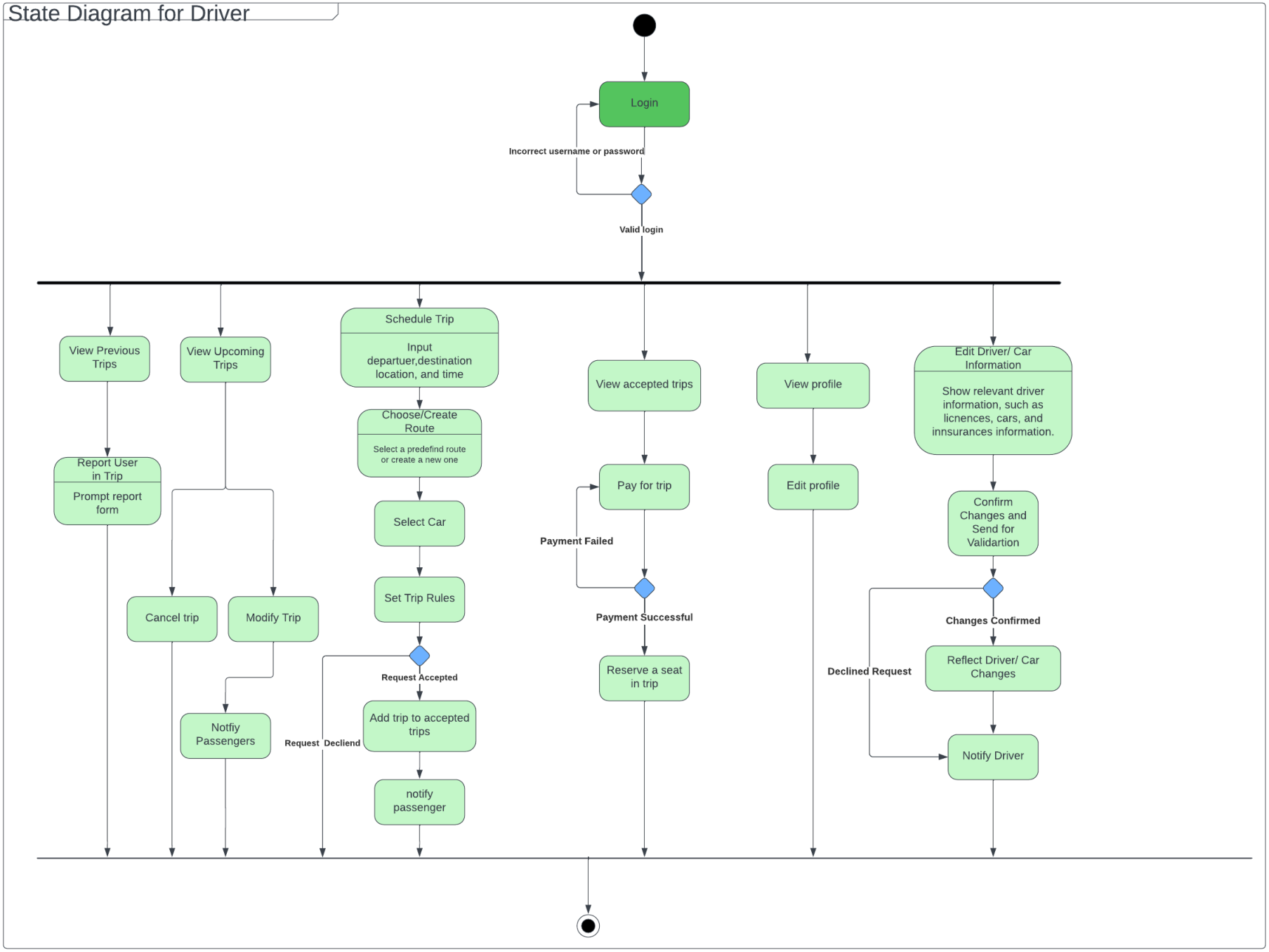


Figure 4.9: State Diagram for Driver

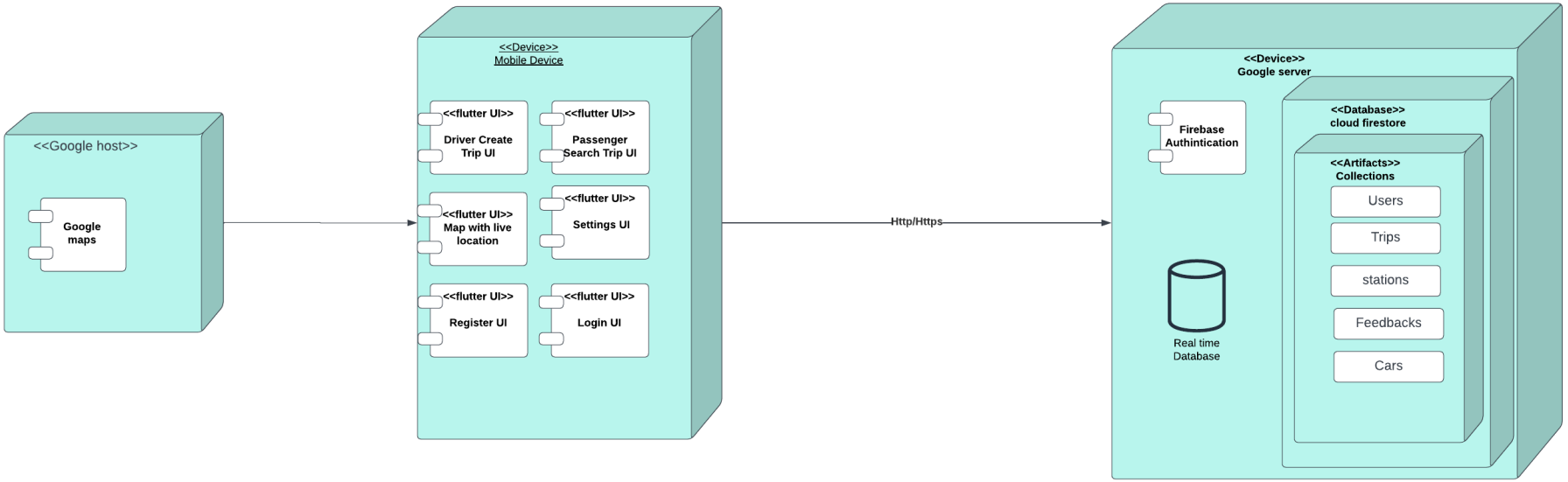


Figure 4.10: Deployment Diagram

5 Implementation

5.1 Introduction

This is a continuation of the first part of the project, which was dedicated to the analysis and design of the project, while this part is dedicated to the implementation and testing of the project. The project is a ridesharing app that allows people signed up as drivers to pick up passengers along the way with them to a mutual destination. The mobile application is built with flutter, and firebase as the backend and database servers.

Since the workflow of the project is incremental, we started by working on the core features that our app depends on first. The core features of our app such as, backend building, searching for trips, booking trips logic, and account creation and authentication. Then we worked on providing map and location services to make it easier for drivers to create trips (pick stations) and for both drivers and passengers to easily find each other. In addition to that, we worked on user profiles, trip filters, and trip reviews.

Throughout these stages, we have worked on the UI and UX for each feature simultaneously. The final stage was to test the app. This section outlines some of the work we have done across stages 1 - 8 of implementation backlog found in Figure 3.2. It includes screenshot from the app running on both Android and iOS presenting the user interface and the features implemented.

5.2 Application Architecture

The application architecture consists of presentation, models, controllers, services, repositories, and others like assets and utility classes, constants, enums, and extensions. Presentations are flutter stateless and stateful widgets which can be whole screens or building blocks used in the various screens of the application. Models represent the entity classes in the application like: stations, users, trips and others. Services and repositories following the Layered Architecture. Controllers are there to handle data being sent to the screens from one or more services, and update the state of the widget to match new data. Most screens in our application have a controller. We use Riverpod State Management solution to manage the state of the widgets, see appendix B. The assets consist of the in-memory images used in the application like the logo and default images. Utils and utility classes consists of different utility classes and functionalities that are shared across the application, for example logic to calculate distances between different points, shared providers like current user provider. Constants hold values that are used across the app such as the location sensor precision level, passenger limit, and others. Extensions can be for built-in classes or custom classes to add more functionalities such a different fromJson method for a model. See Figure 5.1

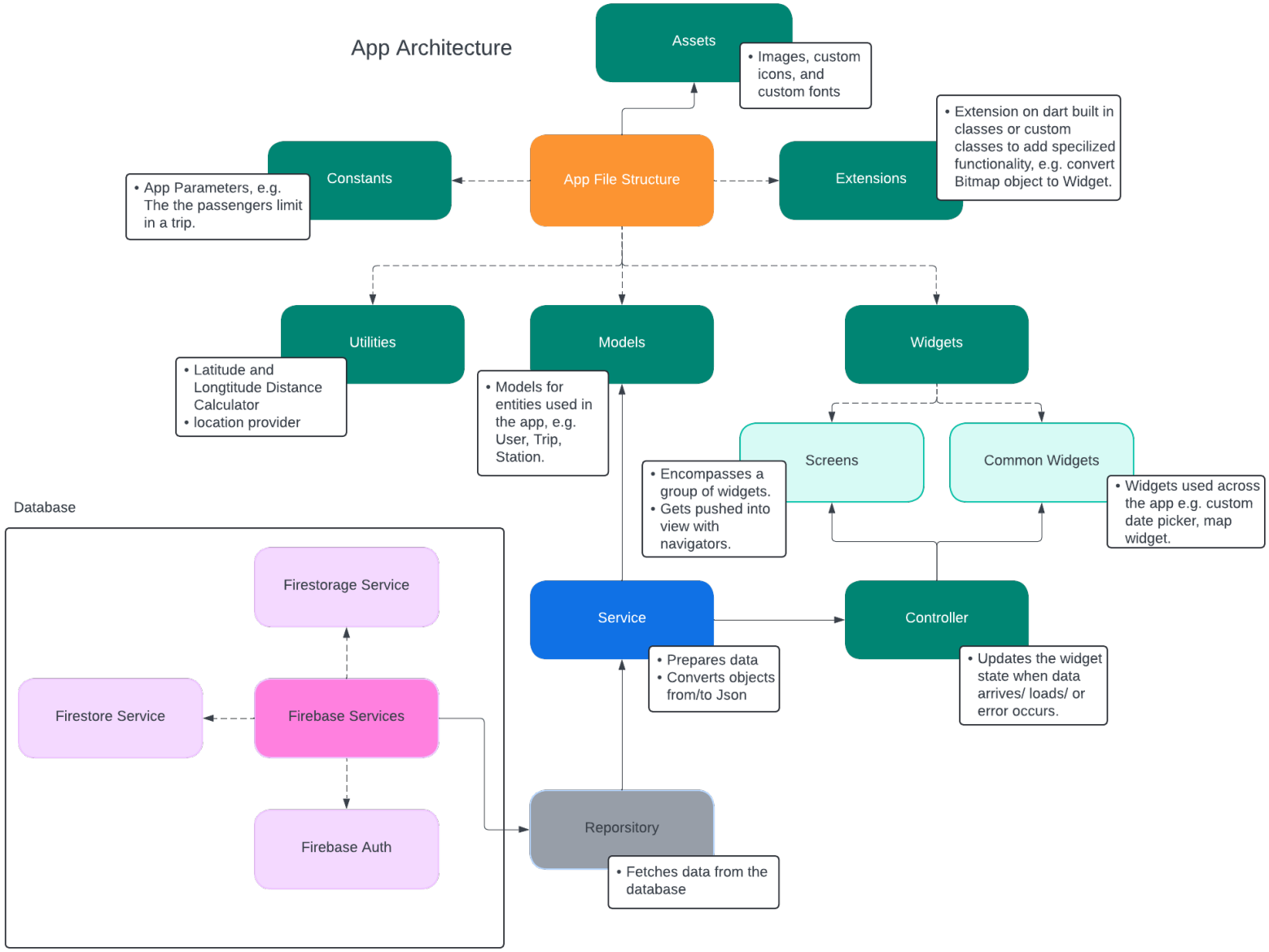
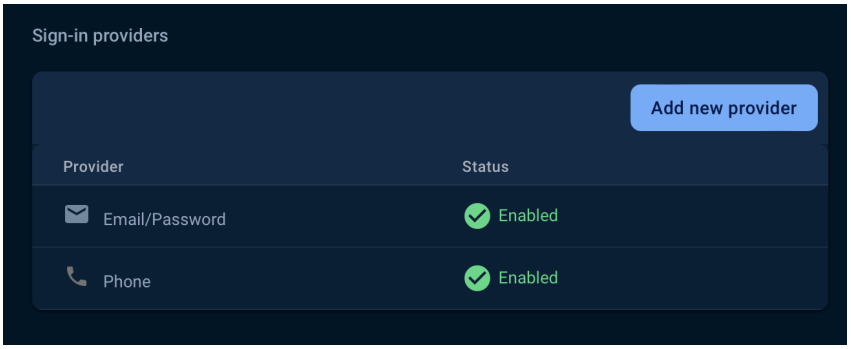


Figure 5.1: Application Architecture

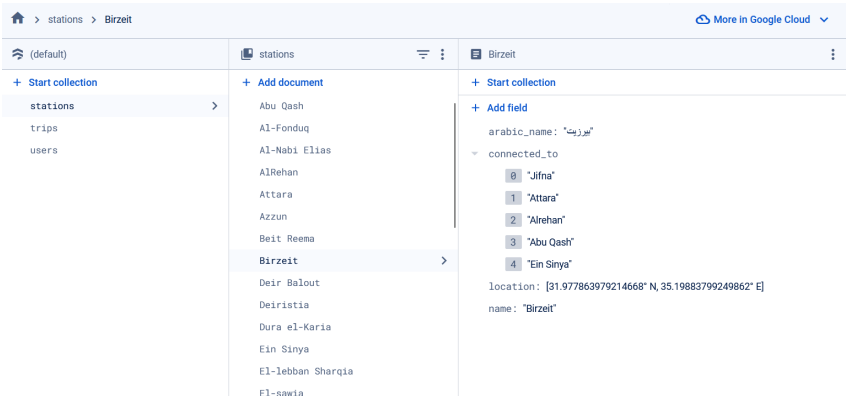
5.3 Firebase Setup

We have created a Firebase project and connected it to the app. We have set up the Firebase Authentication, Firestorage, and Firestore services. Figure 5.2 shows the Firebase console with the Authentication methods enabled and some of the documents stored in the Firestore Database. Firebase Authentication is one of the first services running once the app is started. When the app starts, it goes to a router page called AuthGate. There it checks if there is a saved session and lets the user in if that's the case or reroutes them to the sign in and sign up screen. Appendix C shows the process of starting the app and authenticating using Firebase Authentication.

(a) Firebase Dashboard



(b) Firebase Authentication

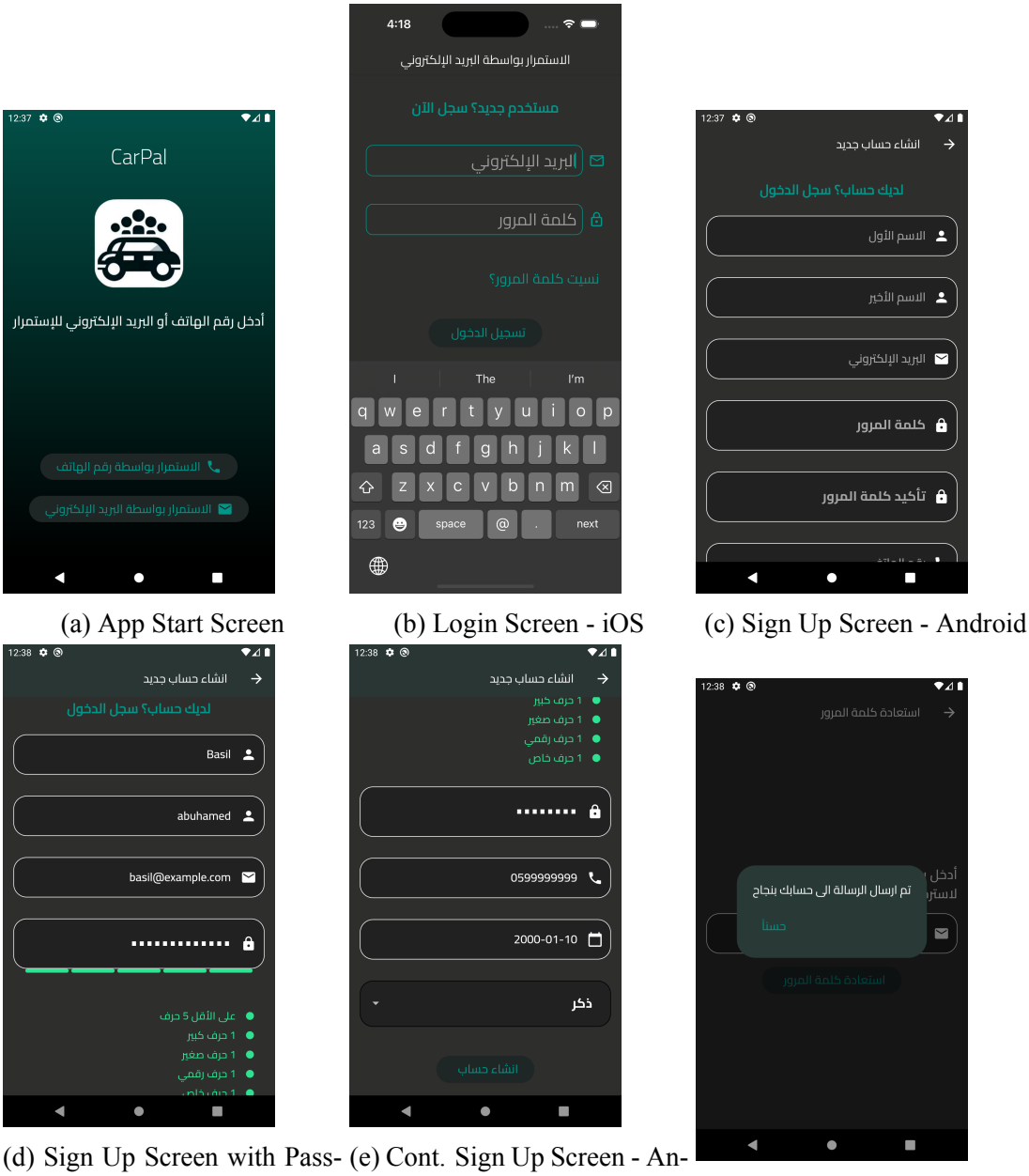


(c) Firebase Firestore Database

Figure 5.2: Firebase Console

5.4 User Sign up and Login

We have implemented the user sign up and login functionalities using Firebase Authentication. The user can currently sign up using their email. They will need to verify their email to be able to login. Email verification as well as password reset are implemented. Figure 5.3 shows the start screen the user sign up screen and login screens. Guest accounts are not implemented yet.



(a) App Start Screen (b) Login Screen - iOS (c) Sign Up Screen - Android

(d) Sign Up Screen with Password Validation - Android (e) Cont. Sign Up Screen - Android (f) Password Reset - Android

Figure 5.3: User Sign Up and Login Screens

5.5 Trips List Screen

The app has a screen for the user to see their scheduled trips and any ongoing trips if they are on trips that are already started. The passenger sees all the trips they requested or registered, whereas the driver sees the list of trips they have created. There is also a button to access a list of past trips that they already finished or canceled. This allows for users to go back and rate they haven't already rated or just review them. This also allows for users to report trips they might have forgotten to report.

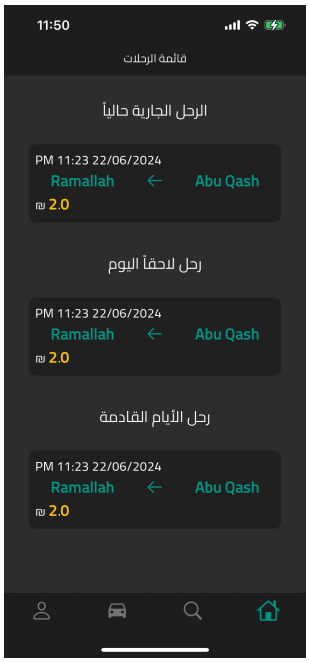
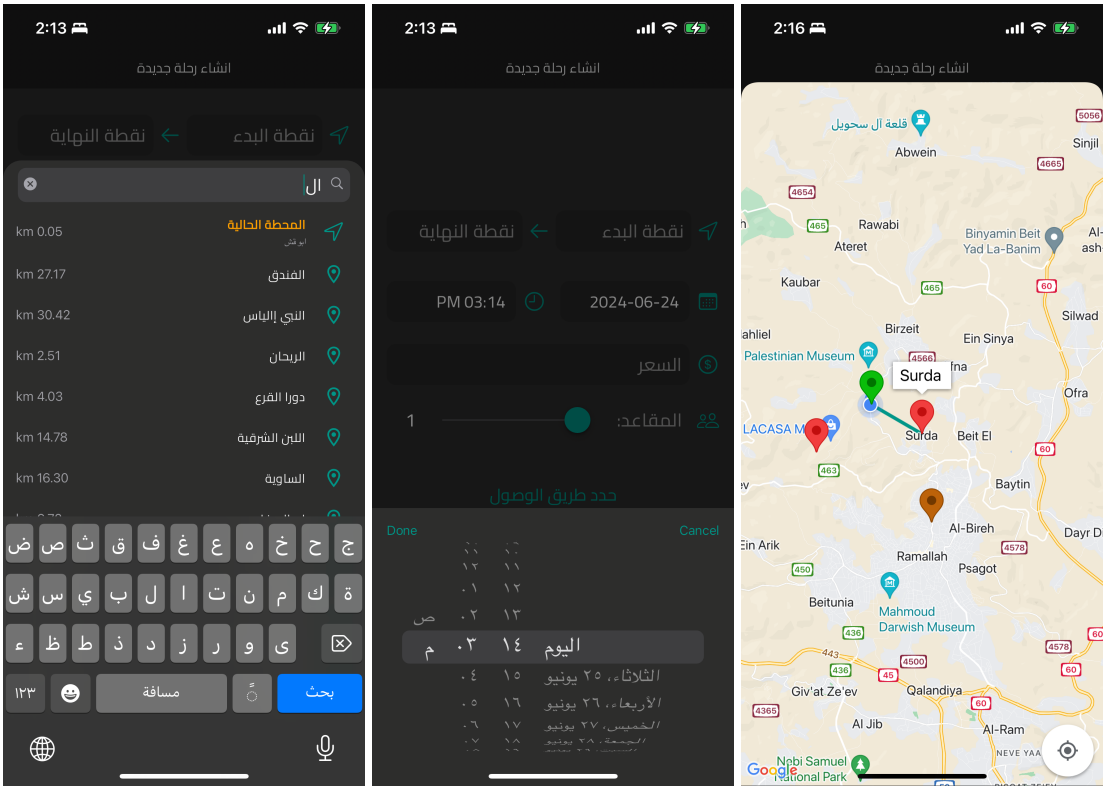


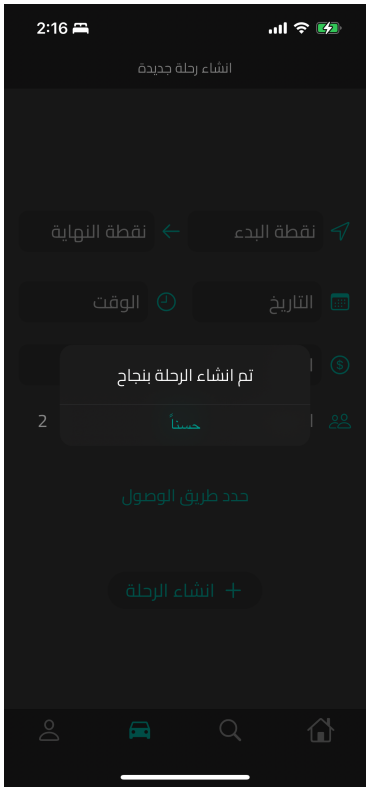
Figure 5.4: Trip List Screen - iOS

5.6 Driver Trip Creation

Drivers can create trips by specifying the departure and destination locations, time of the trip, available seats, and the price of the trip. The driver also has the option to decide on the stations he will pass through. They can do so via an easy to interact with map screen. Further options such as specifying trip rules are not implemented yet. Figure 5.5 shows the trip creation screens.



(a) Station Selection Sheet - iOS (b) Trip Date and Time Picker - iOS (c) Trip Route Station Selection - iOS

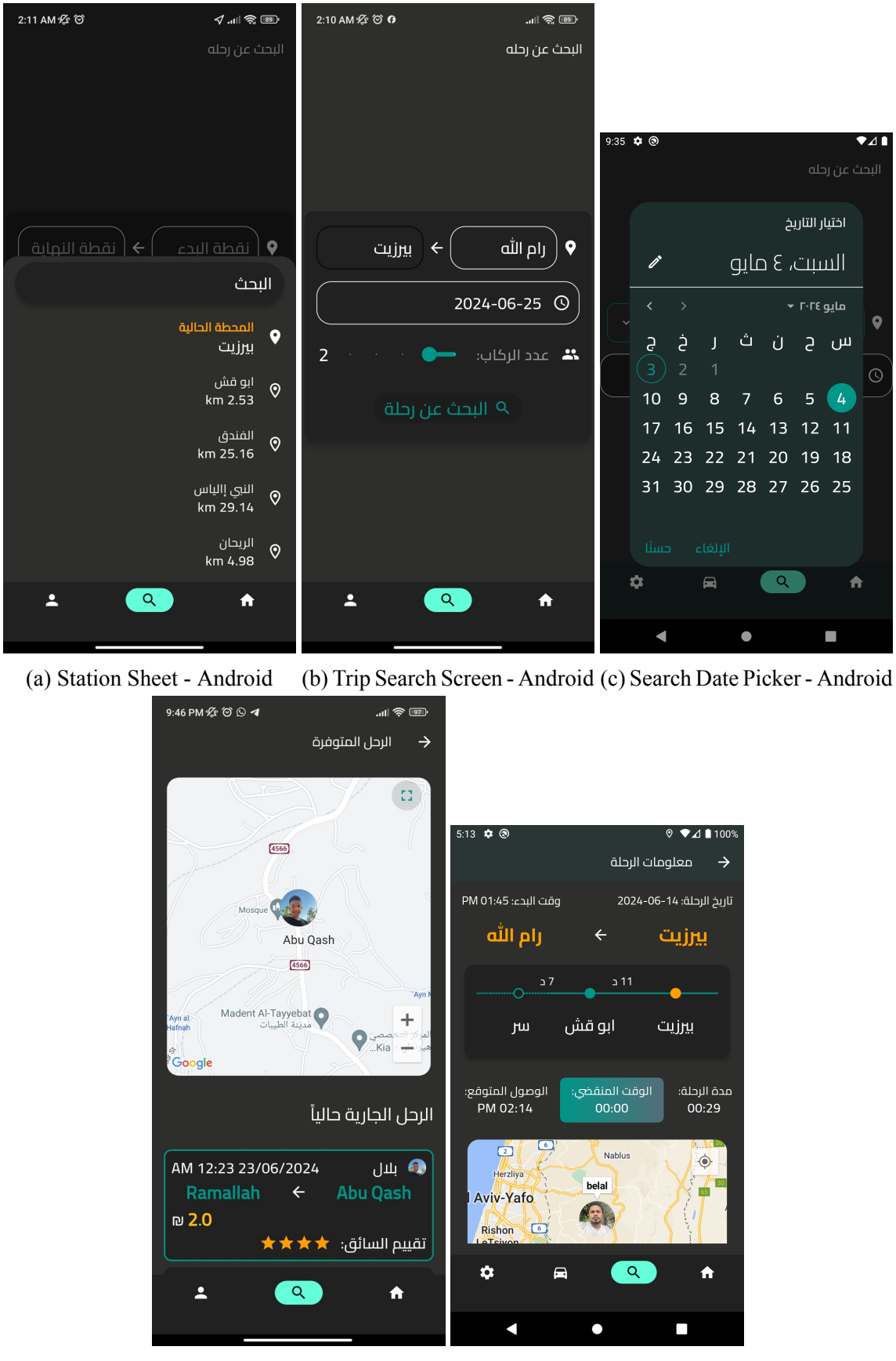


(d) Trip Creation Confirmation - iOS

Figure 5.5: Driver Trip Creation Screens - iOS

5.7 Passenger Trip Search

The passengers can search for a trip using the *Trip Search Screen*. Similar to the *Trip Creation Screen* in UI and parameters, the passenger can search for the trip by entering the departure and destination locations, date of the trip, and the number of passengers that need to be accommodated. When a search query is submitted, the app will transition to a new screen where the passenger can see the trips available and if there are already started trip, there will be a map of the last location the drivers of available trips were seen in. See Figure 5.6. The passenger can pull down to refresh the page and update the trip list and the drivers' locations. The trip list includes information such as the driver's rating, trip, status, and price. The trip list should also be sorted by different parameters (chosen by user) such time, price, distance, filters, and ratings. Each item in the trip list is clickable. Clicking an already started trip will pan the map to the clicked driver's location while clicking it again will open the *Trip Detail Screen* 5.6e.



(a) Station Sheet - Android (b) Trip Search Screen - Android (c) Search Date Picker - Android

(d) Trip Search Result - Android (e) Trip Details Screen- Android

Figure 5.6: Passenger Trip Search Screens

5.8 Trip Detail Screen

The *Trip Detail Screen* has information about the trip. In addition to where and from the trip started, and trip date and time, it will have information such as the estimated finish time as well as the passing stations selected by the driver on creation with the estimated time between each station and the next. The stations will be displayed in a scrollable timeline widget that can update to reflect the driver's current station have they started the trip and were sharing their location; in this case, the trip will also have a map of the driver's live location. Moreover, there is a timer that can count up the time elapsed since the start of the trip to give passengers better sense of the driver's progress. The driver name, profile picture, rating, and car information will be visible in the screen and the passenger can visit the driver's profile from there. The passenger can request to book a seat, cancel a request or booking, drop off when they are dropped off and report the trip to us for any complaints. The driver can start the trip[1], end it, cancel it if it's not started, hide the trip so that it doesn't get seen by any more passengers. He will also see a map showing the passengers' location. Moreover, there will be a list of the passengers registered on the trip showing their names, and rating. When clicked, it will redirect to the passenger's profile. Both the driver and passengers can rate one another at the end. See Figure 5.6e. For more on the life cycle of the *Trip Detail Screen* see appendix D.

5.9 User Live Location

The app provides features to see the live location of the driver and the passenger. We have configured the app so that it asks for device location permissions on both Android and iOS. We are using Google Maps API to show the live location of the driver and the passenger on an instance of an intractable Google Maps Widget built into the app. We have established a stream of realtime locations of users stored in FireStore. 5.7 shows the live location feature in action, streaming the location data of users stored in Firestore. We have used this fea-

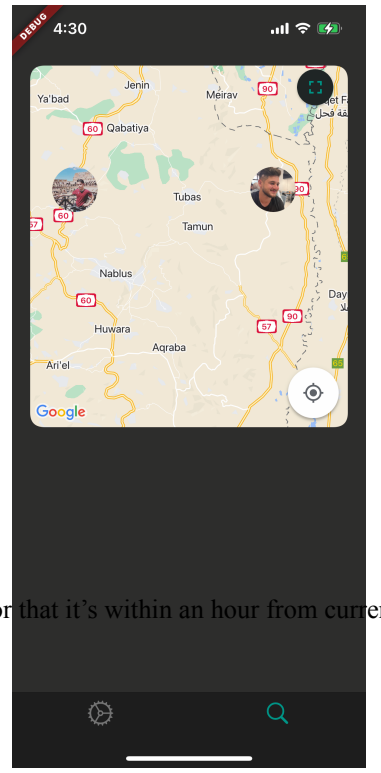


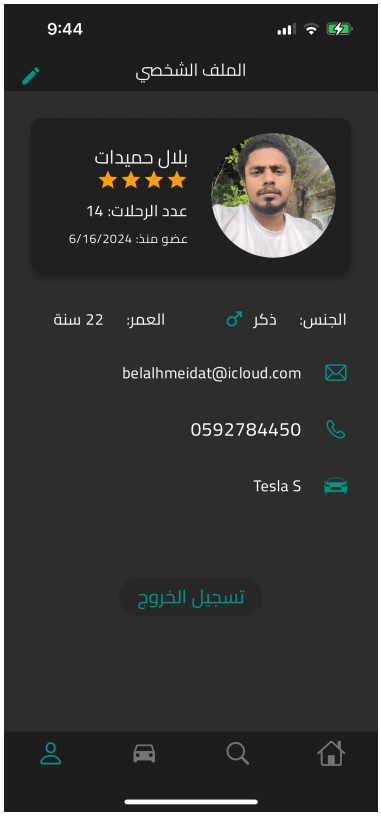
Figure 5.7: User Live Location Demo - iOS

[1]To be able to start a trip, the trip scheduled time needs to be due or that it's within an hour from current time

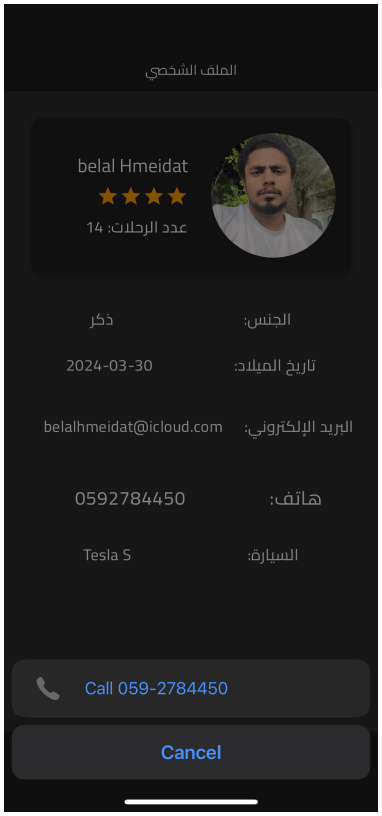
ture for the passenger to track the driver's location and for the driver to see the passenger's location when a trip is ongoing.

5.10 User Profile

The user profile screen shows a user detail, Figure 5.8. The app profile page shows this screen with the option to edit personal details and profile picture. A user can apply to become driver this screen as well. Moreover, phone and email are urls that can be clicked to call or send an email to the profile owner in case the user's are ride sharing.

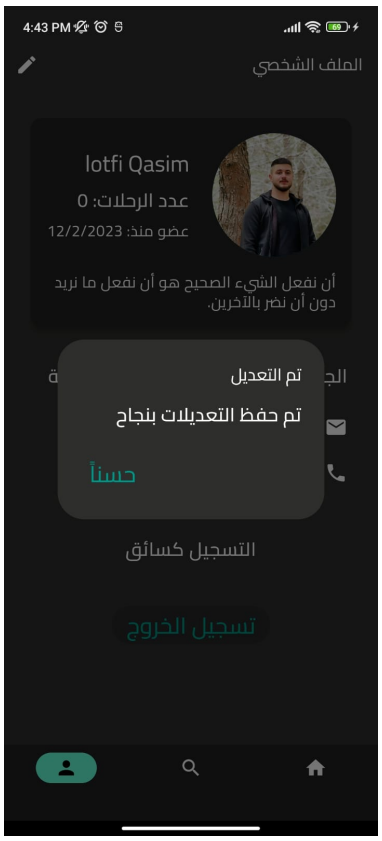


(a) Profile Screen - iOS



(b) Profile Screen Phone No. URL - iOS

(c) Profile Screen Edit - Android

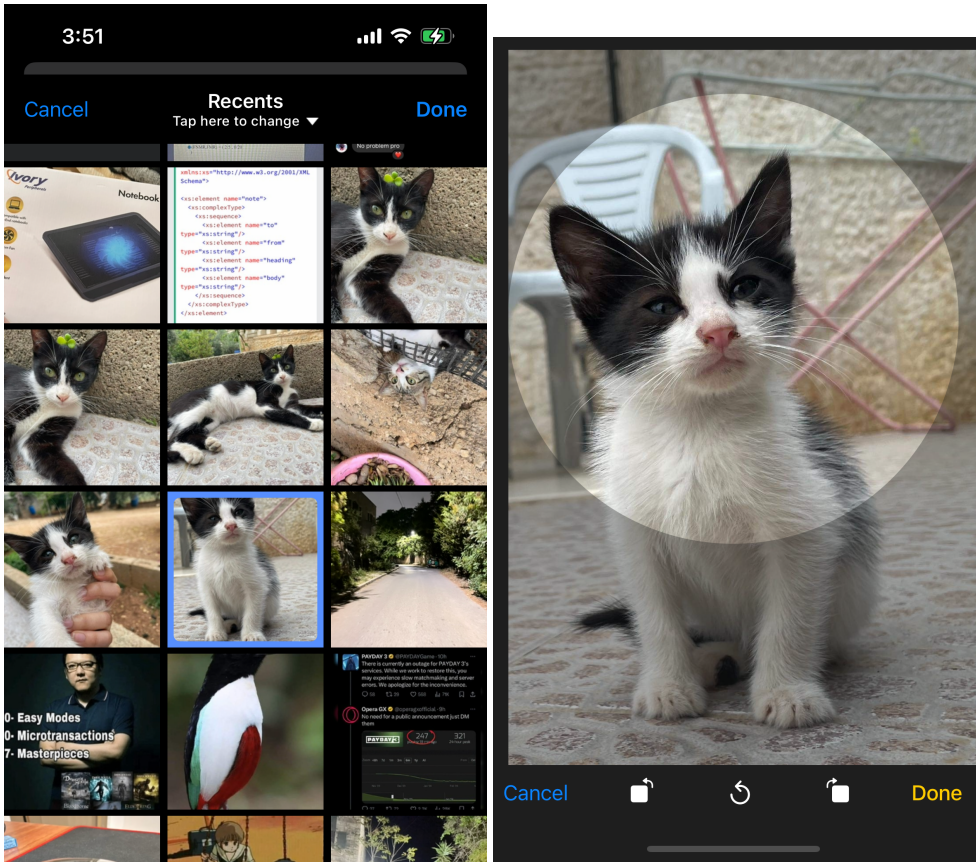


(d) Profile Screen Edit Confirm - Android

Figure 5.8: Passenger Trip Search Screens

5.11 Profile Picture Picker

We have added a functionality for users to add profile pictures using their camera or photo gallery using the Flutter library HL Image Picker [21]. After getting relevant permissions, it allows the user to take/pick photos, crop them, and then set them as their profile picture.



(a) Profile Picture Picker - iOS (b) Cont. Profile Picture Picker - iOS

Figure 5.9: Profile Picture Picker Screens

6 Evaluation and Testing

During app development with each feature implementation we conducted a series of tests to ensure that the functionality is working as expected and that the app is user friendly. As for each feature, we ensured that the implemented interfaces are simple consistent, conventional and responsive, in addition to achieving their purpose while integrating with the backend of the application. And after completing the application conducted system testing of the application which included:

6.1 Functional Testing

In this stage we tested the functionality of the mobile application, including user interactions, and transactions that may be performed on the application. Where for time matter we put some scenarios that the user may perform and conducted a test on it as followed:

Downloading the application Test

The test was conducted by connecting actual phones instead of emulators and downloading the application directly to them, in which the application was successful downloaded on both android and Iphone mobiles, without any issues.

Login and Registration test

After downloading the application and running it the main page of the application should appear in which there are several scenarios that can happen:

1. User is already a member: user can choose to login via email and password or via phoneNo, in which on a successful login attempt he will see the home page.
2. New user in which he will click signup button and enter his credintials and information to create an account, on successful signup user is required to verify his email address by entering a link sent to his email, before processing to the home page.
3. User is already a member but he forgot his password: in which he will enter his email or phone number, and a message to reset his password will be sent to him.

4. User is not connected to the internet: in which the user will be prompted to connect to the internet.

Test Case	Inputs	Expected Result	Actual Result	Pass/ Fail	Comments
Registration	User credentials: email: lutfi@gmail.com, password: 00456eX@23, etc..	User is registered and awaits email verification	user is registered and awaits email verification	Pass	Test successful, email verified after the process for further testing
Registration with missing/ incorrect fields	User credentials with weak password and wrong email format	missing/ incorrect fields are marked red and user is asked to correct them	Incorrect/ missing fields are marked red with a prompt message	pass	
Registration with an already used email	User credentials with email in use	User email already in use message	User email already in use message	Pass	other correct user credentials are kept as they are he is only alarmed about email
Login	User email and password	User is logged in and redirected to the home page	User is logged in and redirected to the home page	Pass	
Login with wrong email or password	User email and password	User is prompted to enter correct email or password	User is prompted to enter correct email or password	Pass	

Table 6.1: Login and Registration Test Cases

Test Case	Inputs	Expected Result	Actual Result	Pass/ Fail	Comments
Login with forgot password	Email address	Email is sent to reset the password if account exists	Email is sent to reset the password if account exist	Pass	password is reset via the link
Login/ registration without internet connection	User credentials	User is prompted to connect to the internet	User is prompted to connect to the internet	Pass	

Login and Registration Test Cases (continued)

Search A Trip test

After successful login, user can search for a trip by entering the search trip screen, entering required filed: such as starting station, end station, date and time of the trip and number of passengers. then clicking the search button in which different results can appear:

1. finding matching trips, in which a split screen is shown with the map and driver last seen locations on it, and a card list in which trip information are listed.
2. No matching trip is found/ all matching trips are full, in which the user is prompted and told that no matching trips are found.
3. User missed to enter a required field, in which the user is prompted to enter the missing field.

Test Case	Inputs	Expected Result	Actual Result	Pass/ Fail	Comments
Search for a trip	User inputs (eg: start station: Ramallah, end station: Birzeit, date: 2024-6-24, passengers: 2)	User is shown a list of matching trips how are from the same date up to the next 48 hours	List of matching trips which have available seats are shown	Pass	Trips and trip drivers are shown both on map and as a list in a split screen
Search for a trip with no matching trips	User inputs (eg: start station: Ramallah, end station: Qalqilya, date: 2024-6-24, passengers: 2)	User is prompted that no matching trips are found	User is prompted that no matching trips are found	Pass	full trips are considered a non-match
Search for a trip with missing fields	User inputs (eg: start station: Ramallah, end station: Birzeit, date: 2024-6-24)	User is prompted to enter the missing field	User is prompted to enter the missing field	Pass	
Search for a trip with same start and end station	User inputs (eg: start station: Ramallah, end station: Ramallah, date: 2024-6-24, passengers: 2)	User is prompted that start and end station can’t be the same	User is prompted that start and end station can’t be the same	Pass	
Search for a trip with no internet connection	User inputs (eg: start station: Ramallah, end station: Birzeit, date: 2024-6-24, passengers: 2)	User is prompted to connect to the internet	User is prompted to connect to the internet	Pass	User session is locally stored, including stations and are updated when the internet connection is established

Table 6.2: Search A Trip Test Cases

Book a Trip test

After finding a matching trip, user can book a trip by clicking on the trip card, in which the trip details are shown, and the user can click on the book button. On each stage the user has chosen a certain trip, and is shown the trip details screen, in which the user can see the driver information, and trip rules, estimated time for the trip, the full path of the trip (trip stations), price, and trip state(started, pending) On successful stage the driver is notified about the booking request, and the passenger is notified about the driver response when it happens.

Create a Trip for Driver test

After successful login, if user is registered as driver he can create a trip by entering the create trip screen, entering required filed: such as starting station, end station, datetime of the trip, number of passengers, and price of the trip. then clicking the create button in which different results can appear:

1. Trip is created successfully, in which the user is redirected to the trip details page.
2. User missed to enter a required field, in which the user is prompted to enter the missing field.
3. User returned to the previous page, in which the home page is shown again.

Test Case	Inputs	Expected Result	Actual Result	Pass/ Fail	Comments
Create a trip	User inputs (eg: start station: Ramallah, end station: Birzeit, start-time: 2024-6-24:12:00PM, passengers: 2, price: 10)	User is redirected to the trip details page	User is redirected to the trip details page	Pass	option for the driver is to start the trip if its 30 minutes away from the start time
Create a trip with missing fields	User inputs (eg: start station: Ramallah, end station: Birzeit, start-time: 2024-6-24:12:00PM, passengers: 2)	User is prompted to enter the missing field	User is prompted to enter the missing field and field is marked red	Pass	
Create a trip with same start and end station	User inputs (eg: start station: Ramallah, end station: Ramallah, start-time: 2024-6-24:12:00PM, passengers: 2, price: 10)	User is prompted that start and end station can't be the same	User is prompted that start and end station can't be the same	Pass	
Create a trip with no internet connection	User inputs (eg: start station: Ramallah, end station: Birzeit, start-time: 2024-6-24:12:00PM, passengers: 2, price: 10)	User is prompted to connect to the internet	User is prompted to connect to the internet	Pass	User session is locally stored, including stations and are updated when the internet connection is established

Table 6.3: Create A Trip Test Cases

Accepting a Trip Request test

After a passenger books a trip, the driver is notified about the booking request, and the driver can accept or decline the request. The driver can accept the request by clicking on the notification, in which the driver is redirected to the trip requests page, and then accepting/declining the request, and the passenger is notified about the driver response.

Test Case	Inputs	Expected Result	Actual Result	Pass/ Fail	Comments
Accept a trip request	Driver clicks on a trip request notification	Driver is redirected to the trip requests page	Driver is redirected to the trip requests page	Pass	nottification is sent to the passenger with updates info
Decline a trip request	Driver clicks on a trip request notification	Driver is redirected to the trip requests page	Driver is redirected to the trip requests page	Pass	notification is sent to the passenger with updates info

Table 6.4: Accepting A Trip Request Test Cases

Live Location test

As for the live location feature, the user can see the live location of the driver and the passenger, in which the user can see the live location of the driver and the passenger on an instance of an intractable Google Maps Widget built into the app. For this feature we tested the working of the live location feature by running the application in a test screen environment, in which the live location of two users where shown on the map, and in which they were updated in real time and synchronized with the location in the database. As for this feature its not specific to a certain user, but it is a general feature that can be used by all users. and we saw that an evaluation test as the one we conducted is enough to ensure the working of the feature

Performance Testing

For this stage we wanted to test the performance of the application, in which we tested the application on different devices, where we found that the application worked as expected on both android and iOS devices, and that the application was responsive and fast, and that the application was able to handle the different functionalities and features that it has. For the load testing we used a firebase database which is a database hosted by google, and is able to handle a large number of requests, and operations and is able to scale with the application. in addition we validated the response time of the application, and asked different users to use the application and give us feedback on the application, in which we validated that the application response time was acceptable. As for this seciton we assured that the application is able to perform well and it can handle the different requests and operations that it was given.

Usablity Testing

As for the usability testing, we have conducted a survey where we asked a total of 15 different random people aged 18 - 26 from both genders and from different fields to evaluate the app. The test was to assess how easy people can navigate through the app, how long does it take them to get hang of it, how they find the app experience, design, idea, and features. We asked them to rate each aspect from 1 to 5 where 1 is lowest and 5 is highest. We also asked them if they would use the app or not, and to give us feedback if any. Figures 6.1 to 6.5 show the survey results.

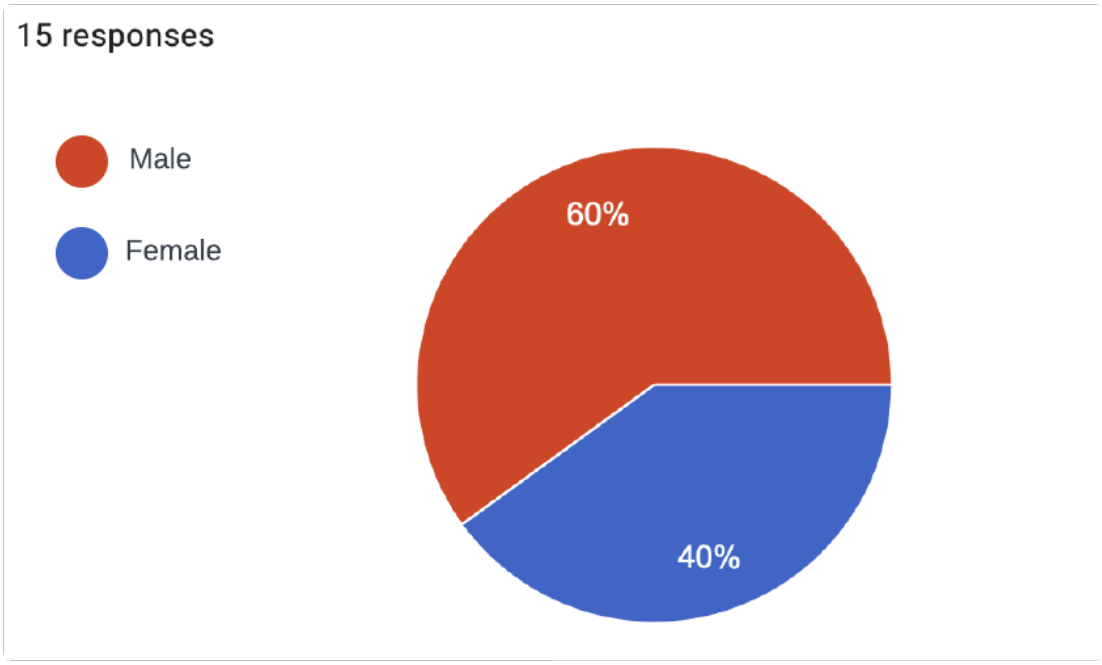


Figure 6.1: Gender distribution of surveyed people

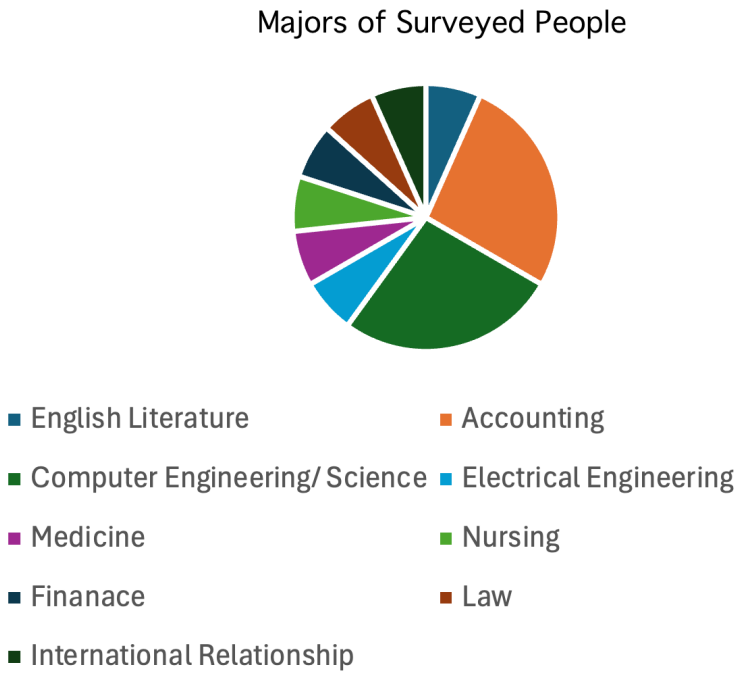


Figure 6.2: Majors distribution of surveyed people

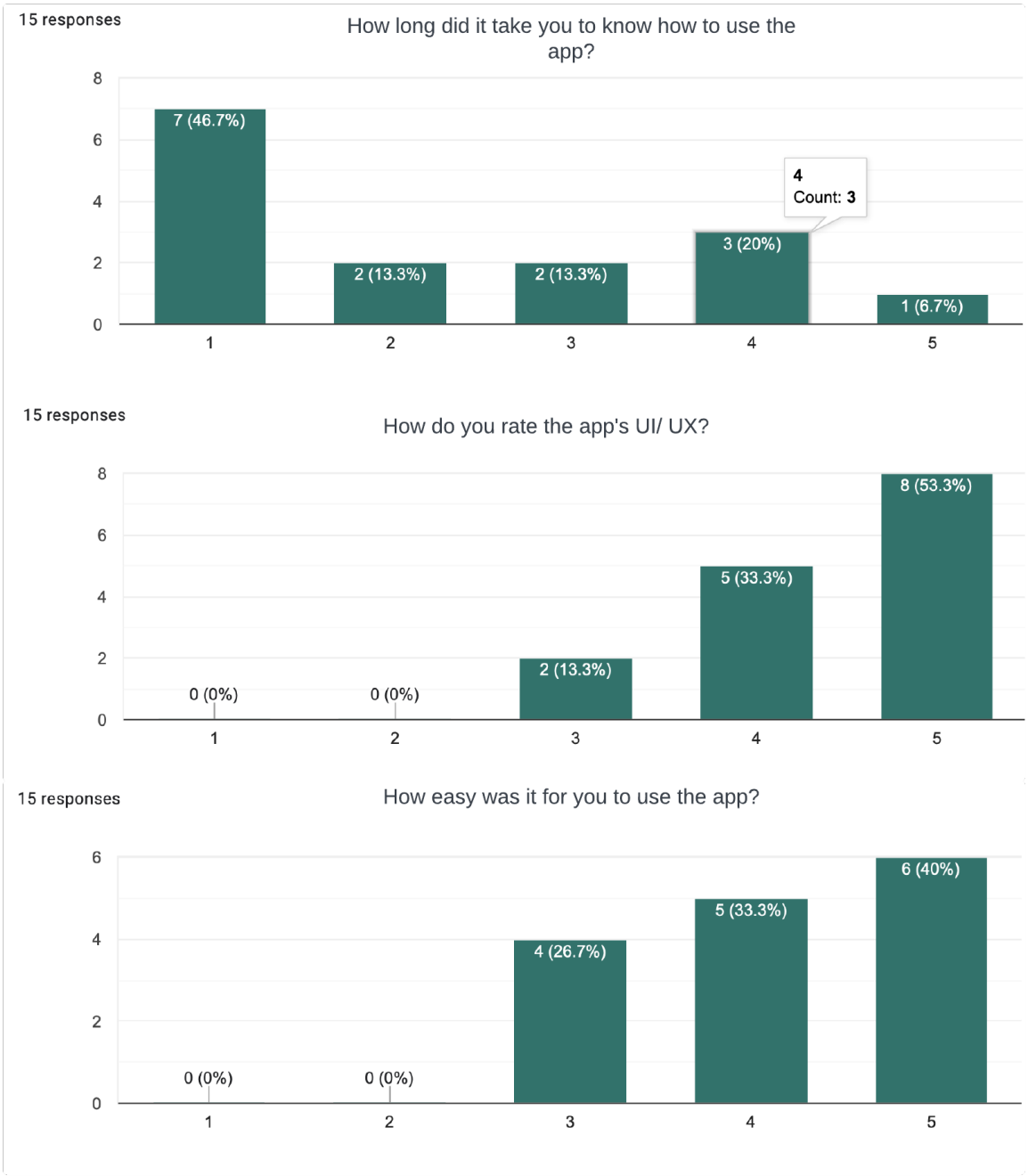


Figure 6.3: Survey Answers - Part I

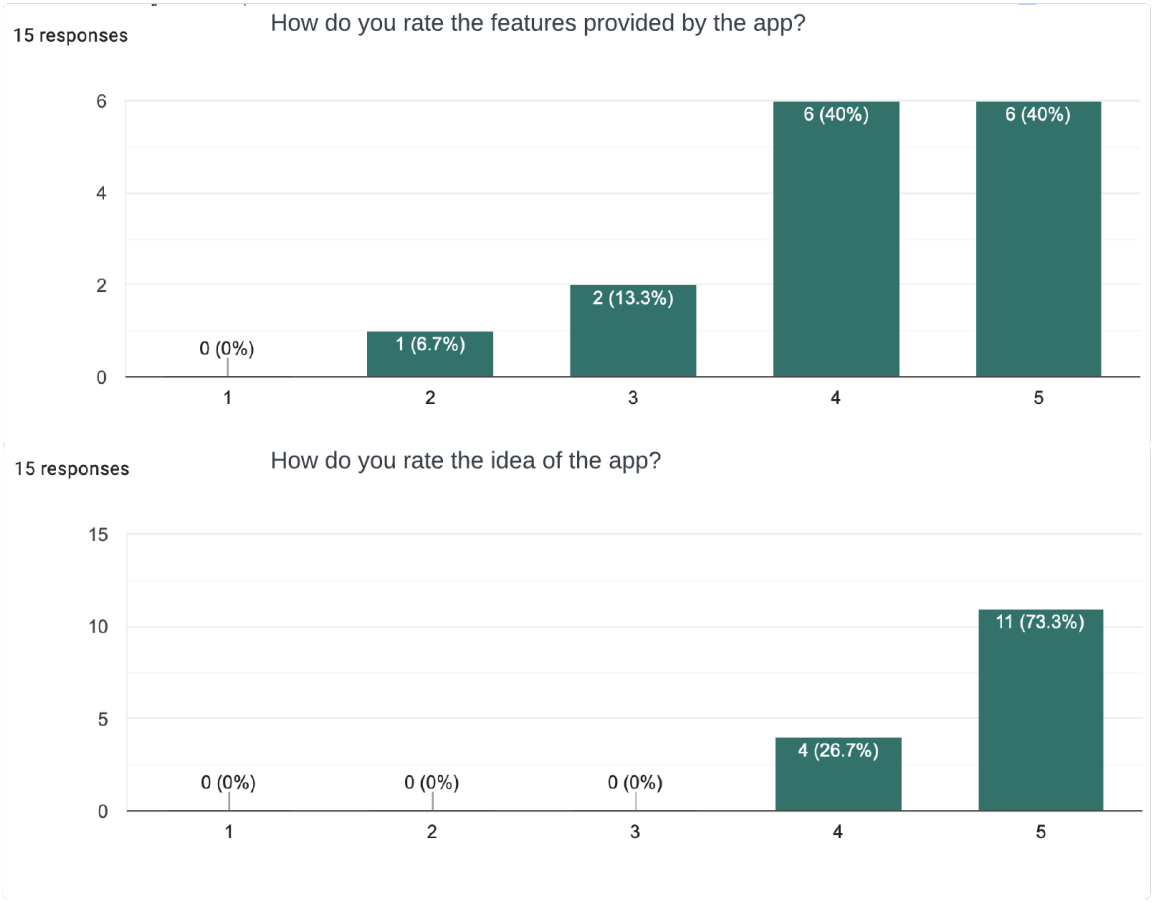


Figure 6.4: Survey Answers - Part II

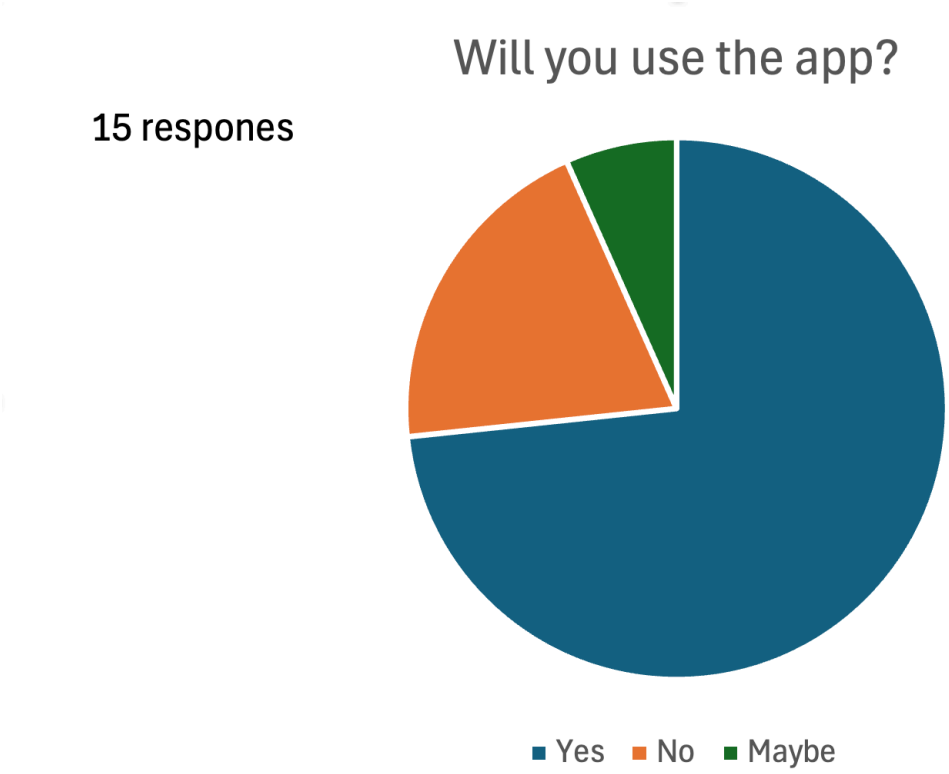


Figure 6.5: Whether surveyed people would use the app or not

Overall the results were positive, with most of the people surveyed finding the app easy to use not taking long to learn, and good UI/UX design. They also gave positive rating for the features they saw and found the idea of the app useful. The vast majority of the surveyed people said they would use the app. We got multiple feedbacks regarding adding more cities, a feedback to ensure safety, and a feedback to improve the UI.

7 Conclusion

This project has successfully developed, the Carpal Ridesharing App, tailored to the unique road conditions and transportation needs in Palestine, where thousands of people rely on various forms of transportation for their daily commutes. Recognizing the challenges with conventional transportation methods—such as issues with time, availability, and comfort—our app provides a viable alternative that addresses these problems effectively.

The CarPal Ridesharing App allows anyone with a driver's license and a car to offer rides to others, facilitating pickups along their routes at any time. This innovative solution helps reduce the costs of commuting, minimizes the number of trips required, and significantly cuts down on waiting time. It is designed to be both affordable and easily accessible through a user-friendly mobile app.

With our app, users can create accounts and become verified drivers, enabling them to schedule trips, or they can search for rides as passengers. Passengers have the convenience of booking through the app. At the conclusion of each trip, both passengers and drivers can provide feedback, fostering a community of trust and reliability.

We believe that Carpal will significantly improve the daily commute for many people and hope it will soon become an integral part of Palestine's transportation ecosystem.

8 Future Work

Adding More Stations

The app is currently limited to the station sample covering stations on the route from Ramallah to Qalqilya. We are planning to expand the dataset to include more stations in Palestine. We already have a feature for the drivers to suggest stations. This feature should help speed up the process of adding more stations during the beta testing phase and the app's initial launch.

Organized Customer Support System

Our app is community based; all people can use and start making trips without our involvement. However, we want to our users to feel safe using our app. We also don't want them to avoid our app for bad experience they might have with other users. Therefore, we are planning to build a better customer support system to deal with user complaints and to make it easier for customers to reach us. This will include designated support line, email, and a ticketing system. This feature is vital for the growth of our app and for the introduction of features such as digital payment, in-app chat, and other features down the line.

In-App Payment for Trip

Digital payment through the app is one of the features that we planned to implement. It gives users more choice as it allows passengers to pay for their trips through the app with different local and familiar payment methods such as PalPay, JawwalPay, and traditional credit and debit card payments. We couldn't implement this feature in time due to our low understanding of the financial requirements to implement such feature, unavailability of commonly used services in our area, and the limited time we have to resolve any constraints that might arise after implementing this feature. We plan to add digital payment in the near future by the time the app is ready for public use.

Roads Status Telegram Channel Integration

The whole idea of the app is to create a ridesharing solution that works in Palestine. We are committed to this cause and that's why we look for ways to make the app more relevant locally. One of these ideas is adding integration with Telegram channel "أحوال الطرق" that provides in

time updates on road conditions, checkpoints, and other important information that can help drivers plan and finish their trips more effectively. We are planning to add Telegram message forwarding from this channel to a server we could use to send notifications and status updates to the app whenever the app user is undergoing a trip to affected area.

Predefined and Saved Trip Routes for Drivers

We are planning on adding a feature that allows drivers to select from predefined routes that are commonly taken when making a trip between two stations. Moreover, they can save trip routes to use them again for trips they make frequently. This will make the process of creating a trip even faster without the need to enter the same information again.

More Language Support

Adding support for more languages such as English, is a feature we feel that we must add as it will make the app accessible to foreign visitors and tourists who like to hitchhike and share rides with locals.

In-App Chat

Implementing a chat feature that allows passengers and drivers to communicate with one another before and during the trip in order to make it easier to coordinate between them is a feature that we thought of from the get go even though we never promised to implement it due to the time constraints and other factors such as the relatively low importance of this feature and its functionality in driving scenarios and the app already provides users with drivers' contact info and makes it easy to initiate a call from the app. However, we feel it would be a great addition to the app and we plan to implement down the road.

More Login Options

We want to give choice to the people and help make our app more accessible. That's why we are planning to introduce more login options from other providers such as Google, Facebook, and Apple. This will make the enrolment process faster for our users. Most importantly, we are planning to add mobile number sign up and authentication method.

Sharable Trip Link

We are planning on implementing a feature that allows users to share their trip with other users in case they want to invite their friends to join their trip or if the driver wants to post their planned trip online. It will be in the form of a link that when opened on iOS or Android it will direct users to the app and open the trip detail screen. This feature will need some form a web implementation of the app Trip Detail Screen in case users open the link on a desktop or a device that doesn't have the app installed.

Carplay and Android Auto Support

We are also looking to find our app a new home outside iOS and Android devices. Since the app will be used a lot while driving, we are planning on porting the app to car platforms such as Carplay and Android Auto. This way drivers can keep track of the trip, get notifications, and locate passengers right from their car screen. This will make the app more accessible and safer to use while driving. Whether we go through with this feature or not will depend on how the mobile app is received by the public and the size of our user base.

Bibliography

[1] Wikimedia Foundation Wikipedia. *Firebase*. URL: https://en.wikipedia.org/wiki/Firebase (visited on 10/16/2023).

[2] Google. *Firebase Realtime Database*. URL: https://firebase.google.com/docs/database (visited on 01/06/2024).

[3] Firebase. *Firebase Logo*. https://commons.wikimedia.org/wiki/File:Firebase_Logo.png. Oct. 2018.

[4] Wikimedia Foundation Wikipedia. *Flutter (Software)*. URL: https://en.wikipedia.org/wiki/Flutter_(software) (visited on 11/14/2023).

[5] Google. *Google Flutter Logo*. https://commons.wikimedia.org/wiki/File:Google-flutter-logo.png. Sept. 2018.

[6] Wikimedia Foundation Wikipedia. *Spring Framework*. URL: https://en.wikipedia.org/wiki/Spring_Framework (visited on 10/13/2023).

[7] Pivotal Software. *Spring Framework Logo*. https://commons.wikimedia.org/wiki/File:Spring_Framewor
Dec. 2018.

[8] Wikimedia Foundation Wikipedia. *Google Maps*. URL: https://en.wikipedia.org/wiki/Google%5C_Maps (visited on 06/19/2024).

[9] Google. *Google Maps Logo*. Feb. 2020. URL: https://commons.wikimedia.org/wiki/File:Google%5C_Maps%5C_Logo_2020.svg.

[10] Wikimedia Foundation Wikipedia. *Blablacar*. URL: https://en.wikipedia.org/wiki/BlaBlaCar (visited on 09/27/2023).

[11] Blablacar App for iOS.

[12] Wikimedia Foundation Wikipedia. *Gett*. URL: https://en.wikipedia.org/wiki/Gett (visited on 10/18/2023).

[13] Bloomberg Bloomberg.Com. *Gett Inc - Company Profile and News*. URL: https://www.bloomberg.com/profile/company/0852787D:IT (visited on 11/15/2023).

[14] Max Eddy PCMAG. *Gett (for iPhone) Review*. URL: https://www.pcmag.com/reviews/gett-for-iphone (visited on 01/10/2024).

[15] Wikimedia Foundation Wikipedia. *InDrive*. URL: https://en.wikipedia.org/wiki/InDrive (visited on 11/06/2023).

[16] inDrive App Store. *InDrive: Ride & Drive With Us*. URL: https://apps.apple.com/ma/app/indrive-ride-drive-with-us/id780125801 (visited on 01/10/2024).

[17] Roger F. Teal. "Carpooling: Who, How, and Why". In: *Pergamon Journals Ltd.* (1984).

[18] Jun Guan Neoh, Maxwell Chipulu, and Alasdair Marshall. "What encourages people to carpool? An evaluation of factors with meta-analysis". In: *Transportation* 44.2 (Mar. 2017), pp. 423–447. URL: https://eprints.soton.ac.uk/381789/.

[19] Huang Hai-Jun, Yang Hai, and G.H. Bell Michael. "The models and economics of carpools". In: *The Annals of Regional Science* (1997).

[20] Kate Brush and Valerie Silverthorne. *What Is Agile Software Development (Agile Methodologies)?* URL: https://www.techtarget.com/searchsoftwarequality/definition/agile-software-development (visited on 12/06/2023).

[21] howljs. *hl image picker*. URL: https://github.com/howljs/hl_image_picker/tree/main (visited on 05/04/2024).

Appendix A

Used Libraries and Packages

The following list includes Flutter libraries and packages used in creating the App. All these packages support both Android and iOS platforms and are downloaded from *pub.dev* the home repository of Dart and Flutter plugins. Additionally, all the packages are up to date and compatible with the latest stable version of Flutter 3.22.2 and Dart 3.4.3 as of the time of writing this report, 26th of June 2024.

1. **firebase_core:** Firebase bindings for Flutter.
2. **firebase_auth:** Plugin to enable Firebase Authentication for Flutter. Creates user session and enables sing-in, sign-up, and sign-out, as well as password reset.
3. **cloud_firestore** Plugin that enables Flutter to read and write to Firebase Firestore database.
4. **firebase_storage:** A plugin to use the Firebase Cloud Storage API in Flutter.
5. **firebase_messaging** A Firebase Cloud Messaging (FCM) plugin to deliver messages and enable push notifications in Flutter.
6. **google_maps_flutter:** A plugin made by Google to integrate Google Maps in Flutter. It communicates with Google Maps API.
7. **riverpod, riverpod_annotation, flutter_riverpod:** State management library for Flutter.
8. **riverpod_generator** Code generation for Riverpod.
9. **freezed,freezed_annotation, json_annotation, json_serializable**: Code generation for immutable classes which allows for creating model classes with autogenerated serialization and deserialization methods.
10. **build_runner** Flutter package to generate code. Used to generate Freezed, Json serialization, and Riverpod code.

11. **flutter_platform_widgets:** A Flutter wrapper library to make creating widgets designed for Android and iOS easier.

12. **hl_image_picker** A plugin to pick images from the gallery app or take a photo using the camera in in iOS and Android.

13. **email_validator:** An advanced email validation library.

14. **flutter_pw_validator** A package used to set rules for password creation and validate them.

15. **geolocator** A Flutter geolocation plugin that provides easy access to 'platform-specific' location services.

16. **permission_handler** A plugin to request permissions such location and camera and check their status in Flutter for iOS and Android.

17. **shared_preferences** A cross platform Flutter plugin for reading and writing simple key-value pairs and persist them in device storage.

18. **cached_network_image** A library that fetches photos from the internet and caches them in device storage.

19. **flutter_local_notifications** A Flutter plugin for displaying local iOS, Android, Linux, and macOS notifications and styling them.

20. **timelines** A library to create timeline widgets. It is used in places like the *Trip Detail Screen* to show the trip timeline.

21. **workmanager** A Flutter plugin for running background tasks on both Android and iOS.

22. **google_fonts** Flutter package to use Google Fonts in Flutter. Provides the Arabic font *Cairo* used in the app.

23. **flutter_decorated_text** A package to enable partial text styling in Flutter.

24. **dropdown_button2** A Flutter package to create simple dropdown menus.

25. **connectivity_plus** A plugin to check if the device is connected to the internet through WiFi, mobile data, or ethernet.

26. **cupertino_icons:** Set of iOS icons to use in Flutter.
27. **flutter_launcher_icons** Flutter package to create and customize the app icon for both Android and iOS.
28. **tuple** A library that enables tuples in Dart.
29. **url_launcher** Plugin to direct urls like phone and email to the device's default url launchers, such as the phone app or email client app.
30. **flutter_native_splash** Creates a splash screen that appears when launching the app on iOS and Android.

Appendix B

State Management Overview

Most of the screens in the app rely on a controller to mange their state. The controller communicates with the service fetches the data for the screen widget. The screen widget watching the controller using Riverpod Provider wrapper updates whenever the controller data changes to reflect new state.

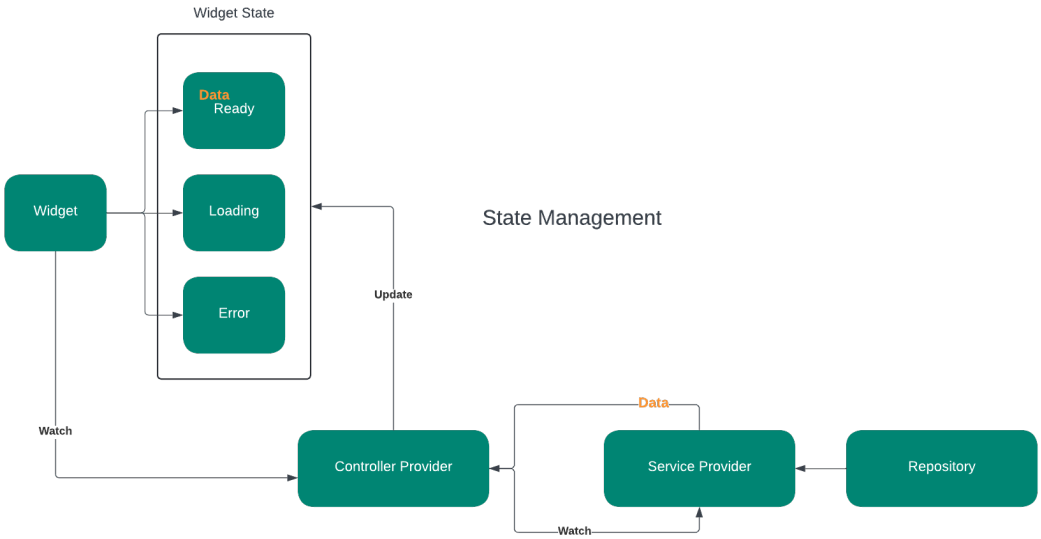


Figure B.1: Application State Management Overview

Appendix C

The Flow of Starting the App

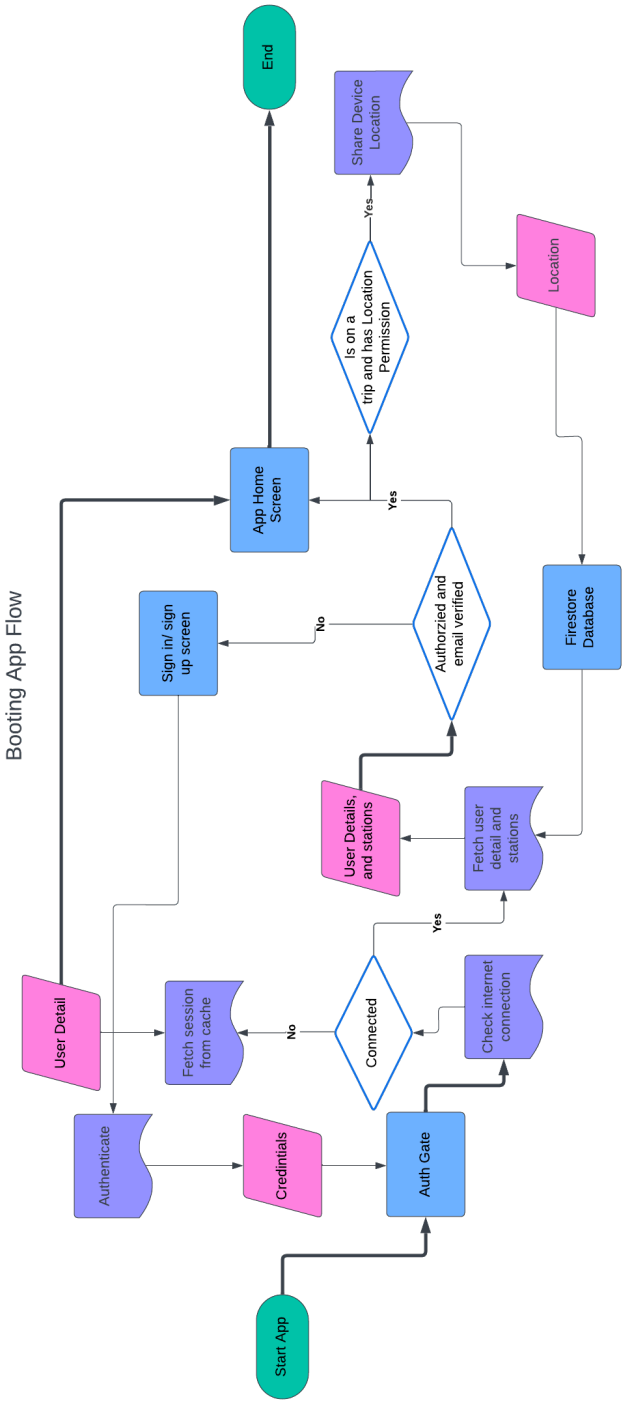


Figure C.1: App Start Flow

Appendix D

Trip Detail Screen Flow

The trip detail screen can be static that doesn't update unless the user does, or dynamic that updates in real-time to reflect trip status and thus available actions, driver location, and the station reached. The status of the trip clicked determines which detail screen is going to be routed in view; a trip that is cancelled or ended or is far from starting will have a static detail screen, while a trip that is ongoing or locked because it's full or about to start will be watching for any updates in the trip status and in the case of an ongoing trip, the driver's location.

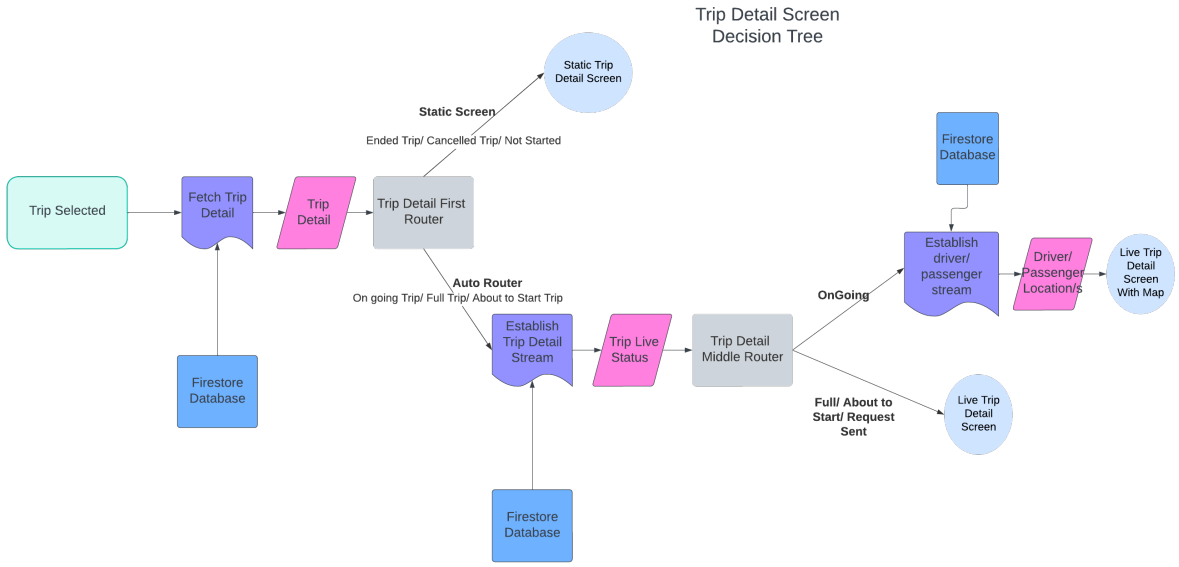


Figure D.1: Trip Detail Screen Router