# CARPOOLING CONNECT


KHADEEJAH RIAZ
**01-134221-108**
ALI BASSAM
**01-134212-210**

Supervisor: Muhammad Siddique

Co-Supervisor: Muhammad Ahtesham Noor

**Bachelor of Science in Computer Science**

Department of Computer Science
Bahria University, Islamabad

December 2025

# Certificate

We accept the work contained in the report titled CARPOOLING CONNECT, written by KHADEEJAH RIAZ AND ALI BASSAM as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by

Supervisor: Sr Siddique (Senior Lecturer)

_____

Internal Examiner: Name of Internal Examiner (title)

_____

External Examiner: Name of External Examiner (title)

_____

Project Coordinator: Mr. Asim Ali Lecturer

_____

Head of Department: Dr. Khuram Ehsan Professor

_____

December, 2025

# Abstract

Carpooling Connect is a secure, efficient, and convenient ride-hailing framework that will resolve the problems of transportation in the urban areas like congestion, elevated transportation expenses, and safety issues. The architecture is designed as a passenger and driver mobile model based on Flutter, an administration panel based on React, a scalable, real-time data storage operation on a cloud-based platform based on Supabase. It has significant characteristics such as real-time location tracking, route planning, and distance estimation with the help of Mapbox and Geolocator, instant notification with Firebase Cloud Messaging, and safe authentication and confirmation systems such as CNIC, driving license, and vehicle number plate verification. The application can also be used to communicate in real-time between the riders and drivers via chat. All these technologies can be used together to ensure that carpooling experience is reliable, safe and efficient, with the operations being well preserved and the level of user confidence and the overall usability of the carpooling experience being greater. Carpooling Connect shows that the combination of innovative cloud-based services and real-time applications can be relevant to offering a convenient and scalable way of making urban mobility safer and smarter.The system will promote the efficiency of ride sharing because it will optimize the route, and minimize redundant use of vehicles, which will help to consume less fuel and to produce environmental impact that is less harmful. The platform is scalable and, therefore, it can serve more users without affecting its performance. Its modular design allows it to be improved in the future with additional functionality like sophisticated analytics, dynamic pricing, and smart ride suggestions. All in all, Carpooling Connect is a convenient and viable solution to the contemporary transportation problems of the city.

# Acknowledgments

We are deeply grateful to Allah Almighty, the Most Merciful and Beneficent, for His countless blessings throughout our academic journey. We sincerely thank our supervisor, Sr Siddique, for his invaluable guidance, unwavering support, and constructive feedback, which played a crucial role in the successful completion of this project. Our heartfelt appreciation goes to our parents, whose continuous encouragement and prayers made this achievement possible, and to our families and friends for being a constant source of motivation. We also acknowledge the Department of Computer Science, Bahria University, for providing a supportive academic environment. This project stands as a reflection of the collective support we have received.

KHADEEJAH RIAZ, ALI BASSAM
Bahria University
E-8, Islamabad, Pakistan

December 2025

# Contents

# List of Figures

# List of Tables

# Acronyms and Abbreviations

| | |
|---|---|
| API | Application Programming Interface |
| GUI | Graphical User Interface |
| GPS | Global Positioning System |
| CNIC | Computerized National Identity Card |
| DB | Database |
| SQL | Structured Query Language |
| IDE | Integrated Development Environment |
| REST | Representational State Transfer |
| iOS | iPhone Operating System |
| OS | Operating System |
| Flutter | UI Toolkit for building cross-platform applications |
| React | JavaScript Library for building user interfaces |
| Supabase | Backend-as-a-Service platform |
| DART | Programming language used by Flutter |
| FCM | Firebase Cloud Messaging |

# Chapter 1

# Introduction

## 1.1 Project Overview

In recent years, urban transportation has been experiencing a fast change due to the growth in population, congestion, and sustainability challenges in cities. Human beings are in a continuous search of low cost, safe, and convenient commuting modes that can accommodate their day-to-day activities. Conventional ride-hailing and public transportation are also helpful, however, they fail to satisfy the needs of daily commuters.

Carpooling is a viable alternative that encourages cost savings, reduces environmental impact, and promotes community-based travel. Nevertheless, the current platforms lack effective verification procedures, matchmaking, and safe payment systems. This project presents a new solution named Carpooling Connect, which aims to improve security, usability, and scalability for urban commuters providing convenience, trust, and a better user experience.

## 1.2 Problem Description

Notwithstanding the benefits of car-pooling, some obstacles prevent its popularization:

1. **Safety Concerns:**The safety concern is the weak verification system that does not encourage the user to share rides with strangers.
2. **Absence of Real-Time Tracking:** he existing apps do not always have live monitoring forming ambiguity and ineffectiveness.
3. **Lack of Convenient Payment:** It is a fact that many platforms do not offer secure and automated payment systems.

4. **Limited User Experience:** Lack of user features like gender-based preferences, feedback and ride variety makes it less convenient.

## 1.3 Project Objectives

The grand purpose of Carpooling Connect) is to develop a secure, effective, and intelligent application in sharing commuting. The system must allow the users to connect easily, control rides, and keep them safe by verifying and tracking them in real time. The other objective is to incorporate seamless payment system and the local financial services to provide safe transaction.

Another aspect the project is concerned with is the establishment of trust by rating/review systems, ride choices that are gender specific, and various mechanisms. The innovative technologies and user-friendly design provide efficiency of transportation, cost savings, and make urban commuting more friendly to the environment, which can be achieved through the Carpooling Connect solution.

The main objectives include:

1. **User Management:** Have secure user registration, login and profile management with Supabase and authentication systems.
2. **Ride Posting and Searching:** Facilitate the process by allowing passengers and carpoolers to post, search and reserve rides.
3. **Real-Time Location Tracking:** Mapbox and Geolocator integrate to track the position with the correct GPS tracking, routing, and distances.
4. **Secure Verification:** Perform CNIC, driving license and vehicle number plate verification to boost user comfort and security.
5. **Communication System:** It should have the ability to chat in real-time so that there is a smooth rider-driver communication prior to and during the trips.
6. **Notifications:** Firebase Cloud Messaging is to be used to provide instant notifications about ride confirmations, cancellations, and alerts.
7. **Admin Management:** Build a react application that records user, ride, report, verification and system analytics.
8. **System Testing and Validation:** Test the application and verify that it can be used in various applications and devices and is reliable and effective.

## 1.4 Project Scope

Carpooling Connect will be a safe, effective and convenient carpooling service to urban commuters. The project covers:

- Supabase and authentication user registration and profile management.
- Checking of CNIC, driving license, and vehicle number plate.
- Posting, searching, and reserving rides.
- Mapbox and Geolocator real-time ride tracking.
- Real-time chat to facilitate smooth rider-driver communication.
- Real-time ride confirmations and cancellations and ride alerts through Firebase Cloud Messaging.
- React-based admin panel to track users, rides, reports, edits, andufferings, analytics of the system.

The scope is limited to intra-city travel. Commercial ride-hailing and inter-city services are excluded but may be added in future updates. Key platform goals include eco-friendliness, security, and scalability to support smart city integration.

## 1.5 Solution Application Areas

This type of system is especially effective with the urban commuters who need to travel to their work place, school and other common places. It can be applied in areas that have a high traffic where the use of public transport is not reliable. It can also be implemented in the local communities and organizations to lower the cost of traveling and environmental impact. Even in the areas that have security issues, it can be used because of its fraud detection and secure payment systems.

## 1.6 Risk Involved

### 1.6.1 Technical Challenges

It can be difficult to integrate and be compatible with third-party APIs, and to be able to maintain high user load performance. Large datasets are necessary to train machine learning models, and they might be unavailable. Performance may also be impacted by network instability and infrastructure constraints in certain areas. It will have to be tested, streamlined, and updated every time.

### 1.6.2 User Adoption

It might also take time to get the commuters to use the new system since most of them would use the ones they are used to. Effective communication, user-friendly design and incentives will be very crucial in drawing users.

### 1.6.3 Legal Compliance

The system has to be in line with the transportation and financial regulations. The privacy laws should be used to manage user information like CNIC and Driving License.

### 1.6.4 Security

Priority is on security of transactions and user identities. Techniques that will reduce fraud and misuse will be multi-layer authentication and strong encryption.

## 1.7 Summary

The Carpooling Connect project addresses limitations in existing ride-sharing platforms and introduces a safer, user-friendly, and community-based solution. This chapter outlined the project background, objectives, scope, and associated risks. The system increases user trust through verification, gender-based options, and real-time ride tracking. By integrating technologies like Map box route optimization API, Supabase's scalable data storage, and Smart Fare Calculation, the project aims to redefine urban commuting. Subsequent chapters will cover the literature review, system requirements, and detailed design specifications.

# Chapter 2

# Literature Review

## 2.1 Introduction

The transport in cities is transforming rapidly because urban centers are experiencing a surge in congestion as well as environmental concerns [1]. Carpooling has been shown to be a good alternative to the daily commuters with some benefits such as cheaper costs, reduced traffic, and reduced carbon emissions[2]. Although there are a number of ride-sharing platforms in the market, the majority of the current solutions do not offer solutions to some of the fundamental problems, such as safety, gender-specific preferences, real-time tracking, and hassle-free payments[3]. The chapter constitutes an extensive literature review of the carpooling websites, the technologies, and the research studies related to the same. It demonstrates the shortcomings of the current options and the way Carpooling Connect can provide a place that is safe, convenient, and efficient to the city commuters.

### 2.1.1 Structure of the Chapter

This chapter is organized as follows:

1. **Section 2.2: Related Applications** Investigates existing carpooling sites, their features, limitations, and emerging trends.
2. **Section 2.3: Related Technologies** Discusses key technologies such as APIs, payment systems, and mapping.
3. **Section 2.4: Related Research Studies** Reviews academic research on ride-sharing, trust, and user behavior.
4. **Section 2.5: Gaps Identified** Summarizes limitations in existing solutions and outlines the motivation for Carpooling Connect.
5. **Section 2.6: Summary of Findings** Concludes the chapter and transitions to the next chapter on system design.

## 2.2 Related Applications

Carpooling ride-hailing applications aim to provide low-cost, convenient, and greener transportation through pairing customers with drivers who share common routes. This part analyses major international and regional platforms: UberPool, Poolyfi, Liftee, and DriveLu.

### 2.2.1 UberPool

UberPool is a ride-sharing service that allows commuters traveling in the same direction to share fares. It provides real-time tracking and in-app payment features, ensuring convenience. However, it lacks comprehensive safety verification and does not provide gender-specific ride options, which may discourage certain user groups [4] [5].

### 2.2.2 Poolyfi

Poolyfi is the carpooling rideshare solution which is the best and the ultimate as it links you to riders and drivers in your area to experience a hassle free carpooling. It assists you in a split travel costs sharing rides with other people so you will spend less on the fuel, tolls and transportation costs.By offering a ride matching feature in real-time, and the facility to search rides or post rides easily, it makes shared traveling affordable, easy, and eco-friendly.Poolyfi promises you a reliable ride management and verified profiles in order to have a trusted experience to start sharing the ride today and have a better ride to travel [6].

### 2.2.3 Liftee

Liftee is a Pakistani car sharing service, which links drivers and customers to travel by car on a daily basis or to long distances, saving on fuel expenses and minimizing harm to the environment. It is trustworthy as it tries to work with verified users and provides a flexible matching infrastructure so that individuals could find or offer rides based on their schedule. Through ride share, Liftee will enhance a sustainable mode of transportation and make commuting more affordable, efficient, and social [7].

### 2.2.4 DriveLu

DriveLu, a Pakistani ride-sharing application, matches users with verified drivers to facilitate affordable, sustainable commuting. It supports ride reservations, real-time trip monitoring, and secure in-app payments. DriveLu also incorporates safety verification and gender-segregated rides to improve accessibility.Its pay as you go scheme and customized scheduling makes it even more convenient to the user[8].

### 2.2.5 Comparative Analysis

Figure 2.1: Comparative Analysis of Existing Carpooling Applications

| Ref | Application | Ride Matching | Security Verification | Gender Options | Real-time Tracking | Payment Integration | Scope |
|---|---|---|---|---|---|---|---|
| [4][5] | Uber Pool | ✓ | Limited | × | ✓ | ✓ | Urban |
| [6] | Poolyfi | ✓ | Verified | × | ✓ | × | Urban/Intercity |
| [7] | Liftee | ✓ | Verified | ✓ | × | × | Urban |
| [8] | Drive Lu | ✓ | Limited | Limited | ✓ | ✓ | Urban/Campus |

As shown in Figure 2.1, this comparison indicates that most existing carpooling platforms lack comprehensive safety verification mechanisms, gender-based matching options, and efficient route optimization features, even though they generally provide basic functionalities such as real-time tracking and payment integration.

## 2.3 Related Technologies

Carpooling platforms rely on a number of enabling technologies:

- **Mapbox & Geolocator:** The applications are applied to track the location in real-time, plan routes, estimate distance, and precise pickup-dropoff navigation [9].
- **Authentication & Verification Systems:** CNIC, vehicle number plate, and driving license check helps bolster user confidence and security.
- **Database:** Supabase is a cloud platform that allows managing user profiles, ride information, chats, complaints, and real-time operations[10].
- **Flutter (Mobile App Development):** It is the main platform of creating a high performance, cross-platform rider and driver application[11].
- **Firebase Cloud Messaging (FCM):** This helps to make immediate push notifications, such as ride updates, booking confirmations, and chat notifications[12].
- **React (Admin Panel):** This will be used to support the web-based administrative platform to track the users, rides, reports, verifications, and system analytics[13].

All these technologies offer effective, smooth and safe user experience in the process of ride-sharing.

## 2.4 Related Research Studies

### 2.4.1 Trust and Safety in Ride-Sharing

Part of the trust and safety in ride-sharing involves a collaboration between the government and personal transportation companies to guarantee the safety of the driving and ridden segments. [14].

### 2.4.2 Ride-Sharing and Urban Mobility

Research indicates that ride sharing can alleviate traffic jam, carbon emissions, and costs of traveling and facilitate sustainable city mobility [15].

### 2.4.3 Technology Integration in Ride-Sharing

The combination of GPS positioning, auto payment, and AI-enhanced route optimization makes driving efficient and convenient. AI-based predictive ride-matching reduces wait time and streamlines trip routes in the most efficient way possible [16].

### 2.4.4 Behavioral Studies

Social trust, personal comfort, and culture determine the adoption of carpooling by the user. The provision of gender-sensitive choices and feedback functions can boost the interaction, particularly in traditional cultures [17].

## 2.5 Gaps Identified

The literature review and analysis of comparative data indicate that current carpooling and ride-sharing systems have a number of critical gaps which limit the level of safety, efficiency, and acceptance of these systems.

**Safety Verification:** The majority of existing carpooling platforms are based on simplified user registration procedures such as email or phone number verification, which are not sufficient to properly verify user identity. The absence of reliable verification mechanisms increases the likelihood of fraudulent accounts and unsafe interactions, thereby lowering the overall level of trust between drivers and passengers [18].

**Gender-Specific Preferences:** Most current systems either do not provide gender-based ride matching or consider it a non-compulsory option. This limitation discourages female users from using such platforms, particularly in regions where cultural norms and safety concerns strongly influence travel decisions [19].

**Payment Integration:** There exists a significant gap in the implementation of automated and secure payment systems. Many platforms rely on manual payments or fixed pricing models, which can result in fare disputes, reduced transparency, and a lack of user trust in the financial reliability of the platform [20].

**Route Optimization:** Existing systems generally lack real-time route optimization, especially in scenarios where passengers have different pickup and drop-off locations. Without intelligent routing mechanisms, travel efficiency decreases, fuel consumption increases, and overall user experience is negatively affected [21].

**Community Trust:** Weak or ineffective rating and feedback systems fail to accurately reflect user behavior. This reduces accountability and limits the ability of users to assess

the credibility of drivers and passengers, ultimately undermining long-term community trust [22].

**Implications for Carpooling Connect:** Carpooling Connect directly addresses these gaps by incorporating strong identity verification through CNIC, police clearance, and driving license validation to enhance user safety. The system introduces enforced gender-based ride matching to ensure comfort and inclusivity for all users. A distance-based and ride-based intelligent fare calculation mechanism is implemented to provide fair and transparent payments. Real-time route optimization using the Mapbox Route Optimization API improves efficiency for multi-passenger rides. Additionally, a comprehensive rating and review system is included to promote accountability and build trust within the user community.

## 2.6 Summary of Findings

The literature review reveals that the apps like UberPool, Liftee, Poolyfi and DriveLu suit the recent urban transportation but they do not address the problem of safety, personalization, and routing efficiency fully. It is confirmed that adoption requirements are trust, verification, and technology integration. The lessons thus derived out of applications and research justify the development of Carpooling Connect with an aim of providing a safe, fast, and convenient carpooling system that bridges the gaps of other solutions.

# Chapter 3

# Requirement Specification

## 3.1 Existing System

The available carpool and ride-sharing services in the market including UberPool, Poolyfi, Liftee and DriveLu are usually aimed at matching passengers with drivers to minimize the cost of travelling and traffic jams. Despite the handy functions of these applications such as in-app payments and live tracking of the ride, they have a number of weaknesses. The level of safety verification is also not very high because most of the platforms provide little to no document authentication, which decreases the trust of the passengers. Likewise, the problem of gender-sensitive options is commonly overlooked, i.e., the preference of men and women in regards to the ride is not considered, and many passengers cannot feel comfortable. There is also a lack of route optimization, and little emphasis is given on reducing the detours and efficiency. Moreover, most of these platforms are not integrated with local payment systems limiting their accessibility to ordinary commuters. Although such systems are convenient, they end up failing to address the fundamental requirements of safety, trust and reliability to urban travelers.

## 3.2 Proposed Solution

Carpooling connect is a mobile carpooling application that will be secure, fast and easy to use focusing on trust, safety and reliability of the operations. CNIC, driving license and vehicle number plate check is mandatory to verify the identities of the users as a means of credibility and safety. The mobile app has been developed with Flutter, which offers a cross-platform experience to passengers and carpoolers and the React-based Administrator panel enables administrators track users, rides, reports, verifications, and system analytics. The backend is Supabase-based, which allows operating the user profiles, ride data, driver checks, chat records, complaints, and real-time data functionality. The services of Mapbox and Geolocator are combined with precise real-time location tracking and route planning and distance calculation, and the algorithm developed by Dijkstra is used to perfect the

matching of the ride and the effectiveness of routes. The system will have real-time chat to enable smooth communication between the riders and drivers. Firebase Cloud Messaging (FCM) is a service that provides an instant notification when booking a ride, confirming and cancellations as well as updates. Payments are done safely using Smart fare calculation API where cashless transactions are allowed. Also, there are rating and review systems to ensure that communal accountability is established and there are ride posting regulations to avoid the system being abused, hence controlled and fair use of the system. These features and technologies allow building a reliable, secure, and efficient carpooling platform that is able to support safe and sustainable urban movement.

## 3.3 Target Audience

The main consumer of Carpooling Connect is the daily commuters like students, professionals and workers who require frequent transport both in the cities and suburbs. It also caters to the needs of individual car owners who want to share rides and reduce their costs of travelling and passengers who want affordable and reliable services. Gender-based preferences provide females and families comfort and safety, and the local users appreciate the combination of Smart fare calculation with local security requirements [23][24].

## 3.4 Functional Requirements

### 3.4.1 Mobile Application

**Registration and Authentication of users**

1. CNIC registered and approved driving license.
2. Hashing of email / phone passwords and profile management.

**Ride Posting and Searching**

1. Drivers can post advertisement rides depending on the route, time, seats, and gender.
2. The passengers search the rides with the assistance of filters and the system offers automatic matches.

**Real-Time Ride Tracking**

1. The GPS integration will give real-time cab sharing of the positions of the drivers with passengers.
2. The system presents the ETA and it keeps on updating the route every minute depending on the changes in the real time location.

**Secure Payment Processing**

1. The fare is based on distance, pickup location and petrol price.
2. The system offers immediate payment reception and online transparency.

**Route Optimization**

1. The detours are minimized by the Map box route optimization API.
2. There are several passengers that are suggested to have effective pickup and drop-off points.

**Rating and Review System**

1. The drivers and the passengers can make the ratings and reviews.
2. Feedback of such scores directly updates the user reputation scores to be accountable.

**Notifications**

1. Live notifications help the user to be informed of requests, approvals, and payments.
2. Notifications include also ride status changes and notifications, before-ride notifications, and alerts.

**Ride Posting Limits**

1. The number of rides that a user can post is limited on a daily basis.
2. The threshold limits do not permit last minute ride posting and there is time limit before the ride starts.

## 3.5 Non-Functional Requirements

**Performance Engineering**

1. **Time Response**: The response is good with all leading functions being streamlined to make sure that the user can respond with the system without hiccups or hesitations.
2. **Scalability**: The system can support the demand and can be easily expanded to cover more users, more rides and transactions without impacting the performance and stability.

**Quality Standards**

1. **Reliability**: The system will be put in such a way that it does not go off easily, but even during peak hours, there will be always minimal downtime.

2. **Security**: The user information and transactions are highly encrypted and secured to meet the industry standards, which provides security and privacy of data at all times.

**Design Constraints**

1. **Resolution**: The application is based on the mobile-first platform, targeting the Android platform and accommodating iOS. It is compatible with local systems through Smart fare calculation API to provide secure payment and responsive layouts to provide a user-friendly experience across devices.

**Usability and Accessibility**

1. **User-friendly Interface**: The interface of the Carpooling connect is straightforward, understandable, and user-friendly even to new and seasoned users.

2. **Accessibility**: Conformity to accessibility requirements will provide inclusivity and suitability of the users with diverse needs, thereby enhancing overall usability.

**Regulatory Compliance**

1. **Data Protection**: The system is in accordance with local transport laws, information privacy laws, and monetary necessities of mobile payments. To achieve personal and financial data security on every level, sensitive information is coded and kept in a safe location.Regular reviews and encryption updates are performed to maintain the highest level of security and Observance.

**Maintenance**

1. **System Maintenance**: The platform will be easily maintained because of its modular structure and well documented, it will require less effort and less cost to improve in future.

## 3.6 Use Cases

A UML use case diagram visually represents the interactions between users (drivers, passengers) and system functionalities. It helps stakeholders understand the scope and flow of the system. Figures 3.1 to 3.7 illustrate the detailed use case diagrams for all system actors, highlighting their respective functionalities and interactions within the system.

### 3.6.1 Use Case: Real-Time Tracking

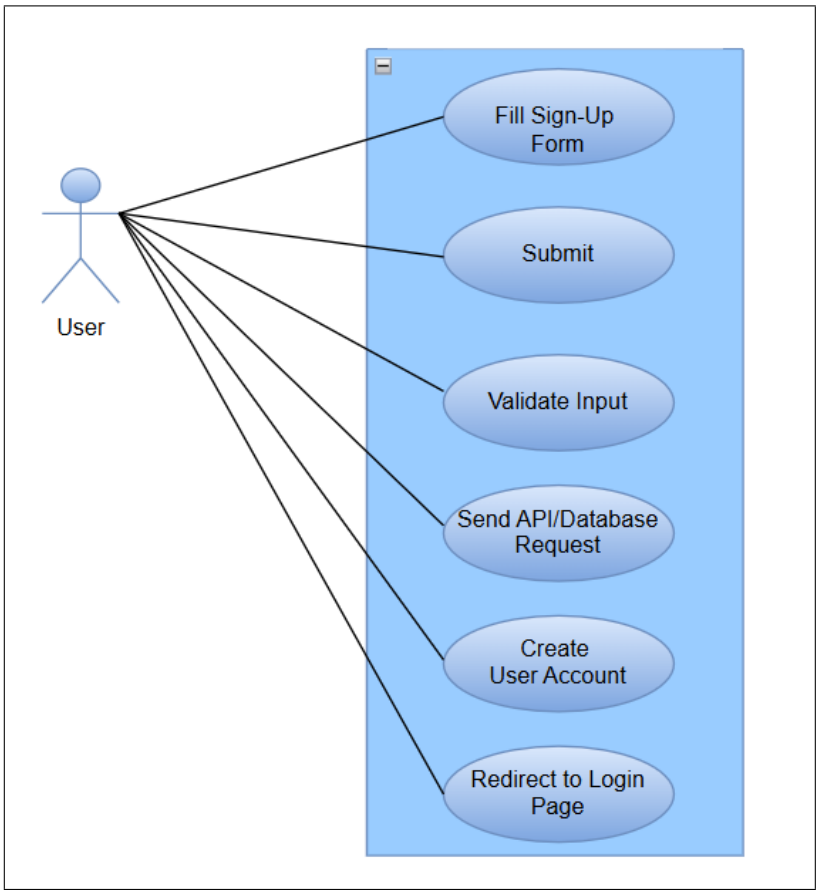


Figure 3.1: Real-Time Tracking Use Case

Table 3.1: Real-Time Tracking Use Case

| | |
|---|---|
| **Use Case Name** | Real-Time Tracking |
| **Use Case ID** | CC-01 |
| **Priority** | High |
| **Primary Actor** | Driver, Passenger |
| **Description** | Driver and passengers can view live ride location, ETA, and route changes. |
| **Basic Flow** | System fetches driver GPS location, updates passenger dashboard with live location, passengers see ETA and route changes. |
| **Alternate Flow** | Location not available → shows last updated location; GPS/Network issues show error message. |
| **Pre-condition** | Ride must be in progress. |
| **Course Event** | **Actor:** start ride, open ride details. **System:** start tracking location, show live map, update ETA/route. |
| **Post-condition** | Passengers and driver have live ride updates. |

### 3.6.2 Use Case: Posting a Ride

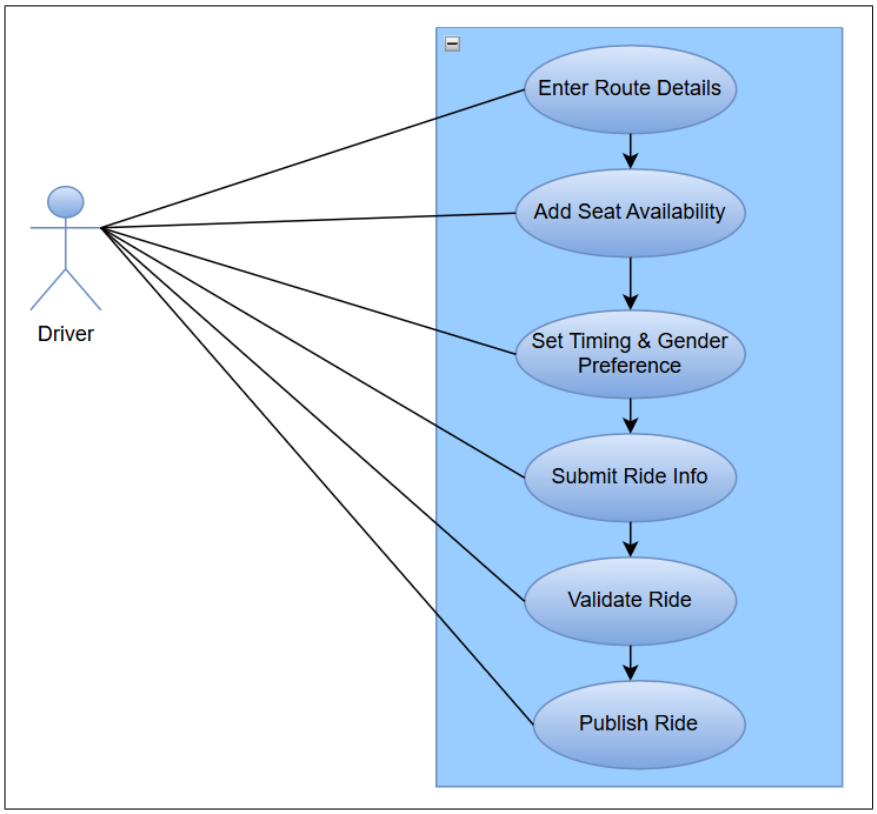


Figure 3.2: Posting a Ride Use Case

Table 3.2: Posting a Ride Use Case

| | |
|---|---|
| **Use Case Name** | Posting a Ride |
| **Use Case ID** | CC-02 |
| **Priority** | High |
| **Primary Actor** | Driver |
| **Description** | Driver posts a ride by entering route details, timing, available seats, and gender preferences. |
| **Basic Flow** | Driver logs in, enters ride details, submits form, system validates and stores data, ride becomes available to passengers. |
| **Alternate Flow** | Missing/invalid ride details → ride not posted; database errors may occur. |
| **Pre-condition** | Driver registered and logged in. |
| **Course Event** | **Actor:** navigate to Post Ride page, enter details, submit form. **System:** show form, validate input, store ride details, confirm posting. |
| **Post-condition** | Ride details stored and available for passengers. |

### 3.6.3 Use Case: Searching for a Ride

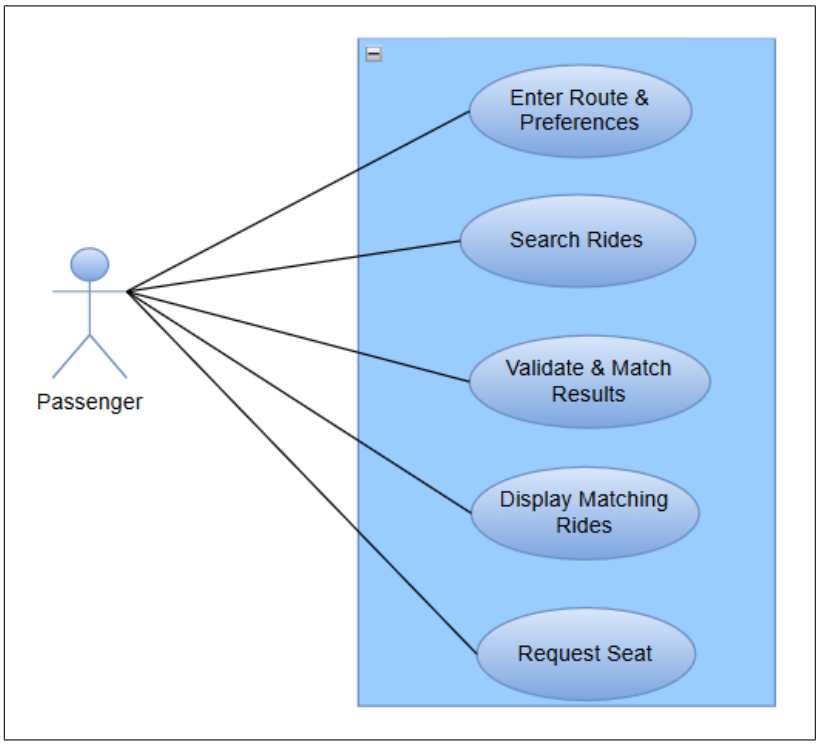


Figure 3.3: Searching for a Ride Use Case

Table 3.3: Searching for a Ride Use Case

| | |
|---|---|
| **Use Case Name** | Searching for a Ride |
| **Use Case ID** | CC-03 |
| **Priority** | High |
| **Primary Actor** | Passenger |
| **Description** | Passenger searches for rides by specifying route and preferences; matching rides are displayed. |
| **Basic Flow** | Log in, enter route and preferences, system searches database, displays matching rides, passenger requests seat. |
| **Alternate Flow** | No rides found → display "No Results" message. |
| **Pre-condition** | Passenger registered and logged in. |
| **Course Event** | **Actor:** navigate to search page, enter route/preferences, submit query. **System:** display search fields, validate input, fetch matching rides, display results. |
| **Post-condition** | Matching ride options displayed. |

### 3.6.4 Use Case: Ride Acceptance

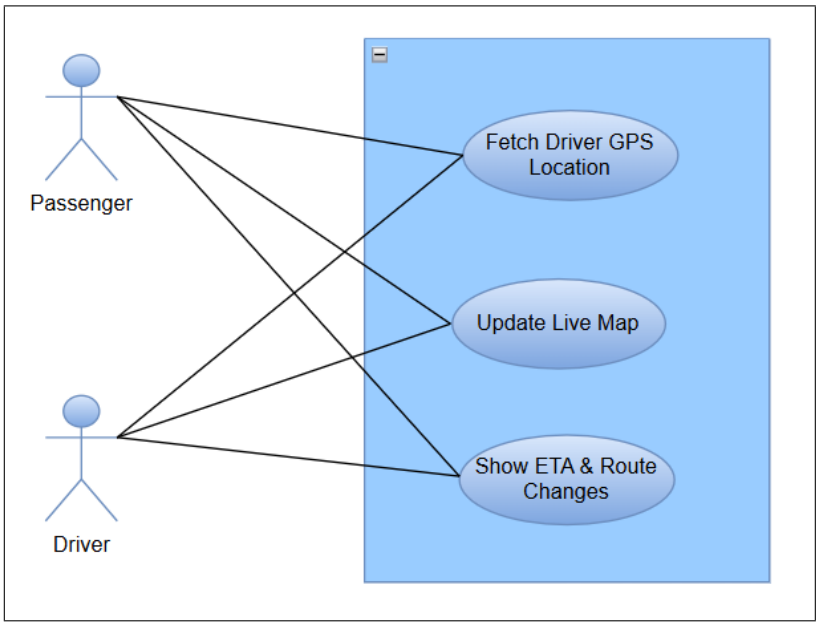


Figure 3.4: Ride Acceptance Use Case

Table 3.4: Ride Acceptance Use Case

| | |
|---|---|
| **Use Case Name** | Ride Acceptance |
| **Use Case ID** | CC-04 |
| **Priority** | High |
| **Primary Actor** | Driver |
| **Description** | Driver accepts or rejects passenger ride requests. |
| **Basic Flow** | Passenger sends request, driver reviews request, accepts request, system confirms booking. |
| **Alternate Flow** | Driver rejects request → passenger notified. |
| **Pre-condition** | Ride posted and passenger requested seat. |
| **Course Event** | **Actor:** review request, accept/reject. **System:** notify passenger, update booking status. |
| **Post-condition** | Ride booking confirmed or rejected. |

### 3.6.5 Use Case: Payment Processing

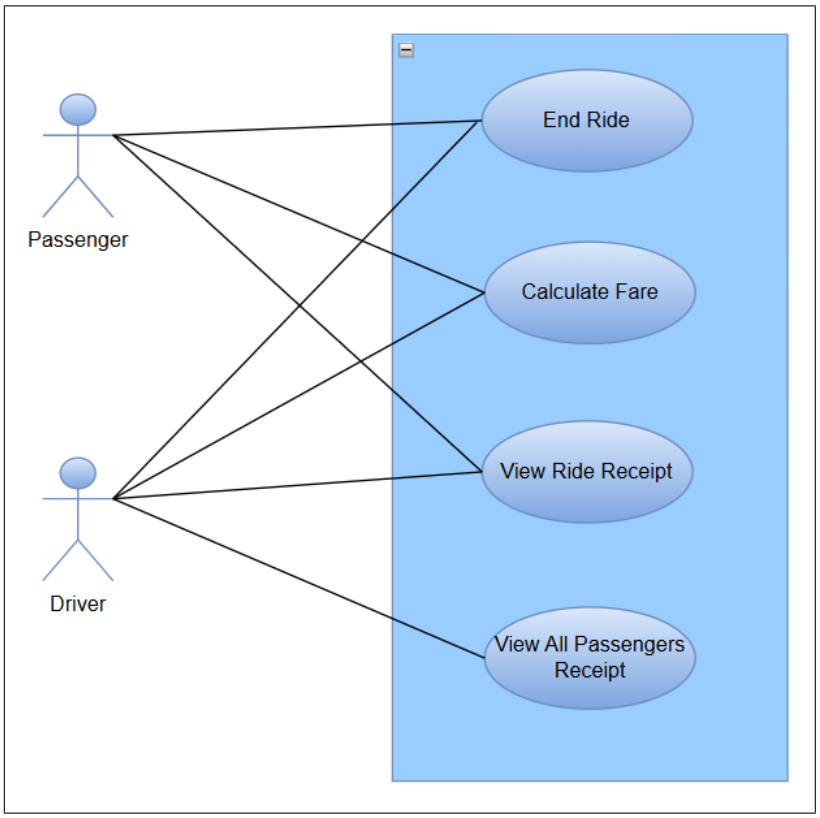


Figure 3.5: Payment Processing Use Case

Table 3.5: Payment Processing Use Case

| | |
|---|---|
| **Use Case Name** | Payment Processing |
| **Use Case ID** | CC-05 |
| **Priority** | High |
| **Primary Actor** | Driver, Passenger |
| **Description** | Fare is calculated and receipts are shown. |
| **Basic Flow** | Calculate fare, generate receipt. |
| **Alternate Flow** | Payment fails → retry or alternate method. |
| **Pre-condition** | Ride completed. |
| **Course Event** | **Actor:** ride ends, select payment option. **System:** calculate fare, confirm receipt. |
| **Post-condition** | Payment processed and stored. |

### 3.6.6 Use Case: Rating & Reviews

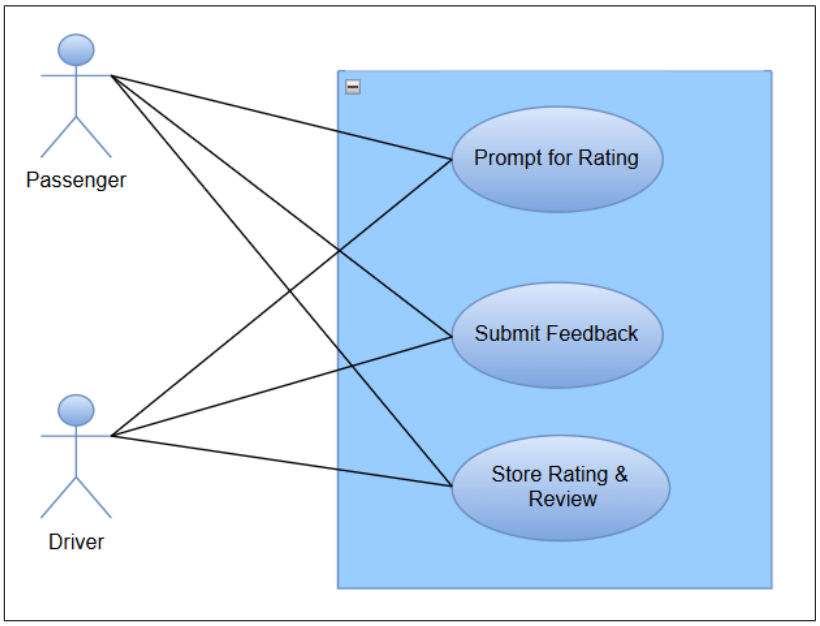


Figure 3.6: Rating and Reviews Use Case

Table 3.6: Rating and Reviews Use Case

| | |
|---|---|
| **Use Case Name** | Rating and Reviews |
| **Use Case ID** | CC-06 |
| **Priority** | Medium |
| **Primary Actor** | Driver, Passenger |
| **Description** | Riders and drivers submit ratings and feedback after ride completion. |
| **Basic Flow** | Ride ends, system prompts feedback, user submits rating/review, system saves feedback and updates profile. |
| **Alternate Flow** | User skips feedback → record "No Review Given". |
| **Pre-condition** | Ride completed. |
| **Course Event** | **Actor:** submit rating/review. **System:** save feedback, update profile. |
| **Post-condition** | Feedback stored, reputation updated. |

### 3.6.7 Use Case: Notifications

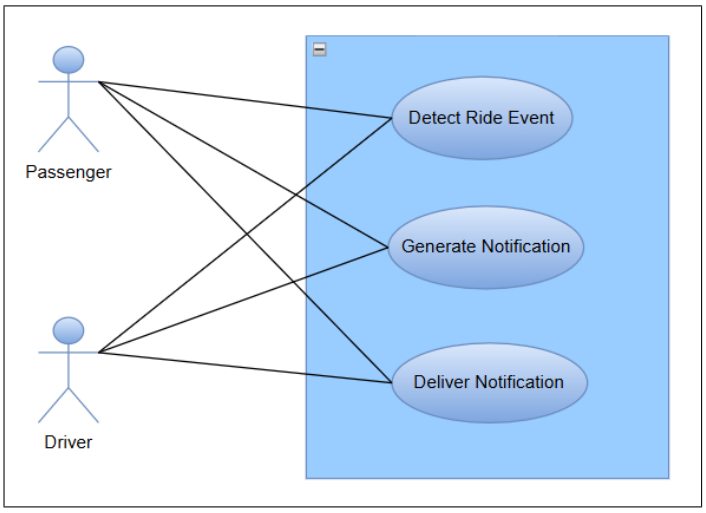


Figure 3.7: Notifications Use Case

Table 3.7: Notifications Use Case

| | |
|---|---|
| **Use Case Name** | Notifications |
| **Use Case ID** | CC-07 |
| **Priority** | Medium |
| **Primary Actor** | Driver, Passenger |
| **Description** | System sends alerts for ride requests, confirmations, cancellations, and updates in real-time. |
| **Basic Flow** | Generate notification on event, user receives it, views details. |
| **Alternate Flow** | Network delay → notification delivered later. |
| **Pre-condition** | User logged in with notifications enabled. |
| **Course Event** | **Actor:** receive notification. **System:** generate and send alert. |
| **Post-condition** | User stays updated with ride status. |

## 3.7 Project Feasibility

The Carpooling Connect system is considered highly feasible because it is built on reliable and scalable technologies. Features such as real-time tracking, secure digital payments, and route optimization ensure efficiency while reducing manual effort. Since the system makes use of cost-effective, open-source tools and is designed for easy adoption by local commuters, it offers a practical and long-term solution for everyday travel needs.

# Chapter 4

# Design

The Carpooling Connect design is aimed at converting the needs into a systematic, realistic and scalable solution. In this chapter, the architecture, design methodology and system models that are used to make the platform secure, reliable and user friendly are described. The design is also based on performance, scalability, and adherence to the local standards. Both high and low-level design models are provided so as to give a clear picture on the aspects of components of the system, how they interact with each other and the technical basis on which they are to be developed.

## 4.1 System Architecture

The Carpooling Connect system is constructed based on a three level architecture comprising of the mobile application, backend systems and the database.Every tier is responsible, which guarantees modularity, scalability and simple connection with third-party APIs.Figure 4.1 illustrates the overall System Architecture Design of the Carpooling Connect system.

- **Mobile Application:** The mobile app is written in Flutter; it is the main interface of drivers and passengers. It manages the registration, ride posting, search, tracking, payment, and notifications having a simple and easy-to-use interface.
- **Database:** Supabase Firestore is the database following its real-time syncretism, scalability, and the simplicity with which it can be integrated into mobile apps. It saves user information, ride, payments, ratings, and notifications and secures the data.

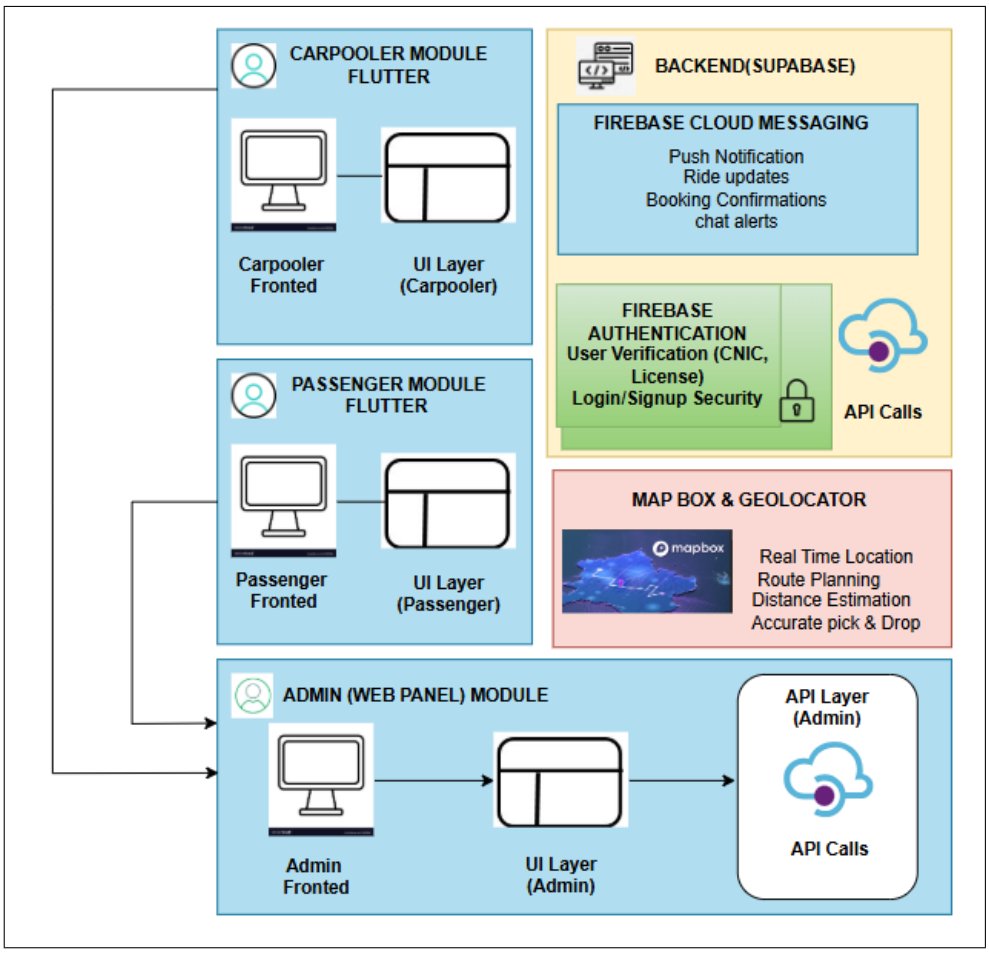


Figure 4.1: System Architecture Design

## 4.2 Design Constraints

During the design, various factors were taken into consideration in order to make the system work.

- **Balancing Functionality and Performance:** The application is geared to provide a fast response time and handle various functionalities such as tracking, ride matching and payments without delays.
- **Security and Data Privacy:** High-end encryption and verification can be done to maintain confidentiality of user data and adherence to local privacy laws.
- **Scalability and Future Growth:** The system will be built with a cloud based platform, which will easily expand with the growth of users and ride requests as time goes by.

## 4.3 Design Methodology

Carpooling connect is designed and developed on the basis of Agile iterative methodology. Agile is flexible, rapid in iterations and ensures constant interaction with the stakeholders, thus the system is not fixed to hard and fast requirements.The adopted Agile Development Methodology is illustrated in Figure 4.2.

- **Rapid Prototyping:** The first prototypes are developed within a short period to prove ideas and collect user feedback before actual implementation.
- **Incremental Delivery:** The features are provided in small and manageable modules and guaranteed to be tested and deployed in time.
- **Flexible Design:** The modular architecture enables the addition or alteration of new features to the system without impacting the whole system.
- **Close Collaboration:** Developers, designers, and stakeholders collaborate throughout the process to make sure that it is in line with the needs of the users.

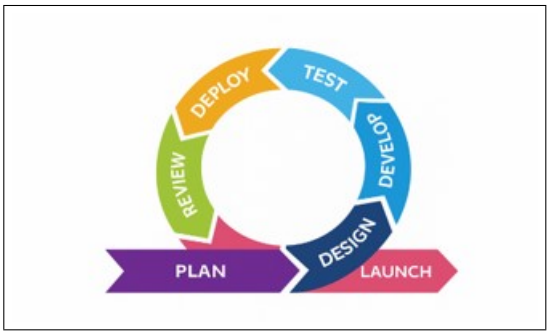


Figure 4.2: Agile Development Methodology

## 4.4 High Level Design

High-level design refers to the act of defining the overall structure of the system and its larger objects, which is used as a guide in the development of the system in greater detail. Key components include:

- **Functional View:**Displays the interaction of users with the main operations such as registration, posting of rides, tracking, payment, and notifications.
- **Data Model:** Entities Relationships are defined between User, Ride, Booking, Payment, Notification, and Review.

- **API Design:** API will allow secure communication between the mobile application and the backend and database and integration with third-party services such as Map box, Geo locator and FCM.

Figure 4.3 shows the Passenger Activity Diagram, while Figure 4.4 illustrates the Carpooler Activity Diagram, representing high-level user workflows and system interactions.

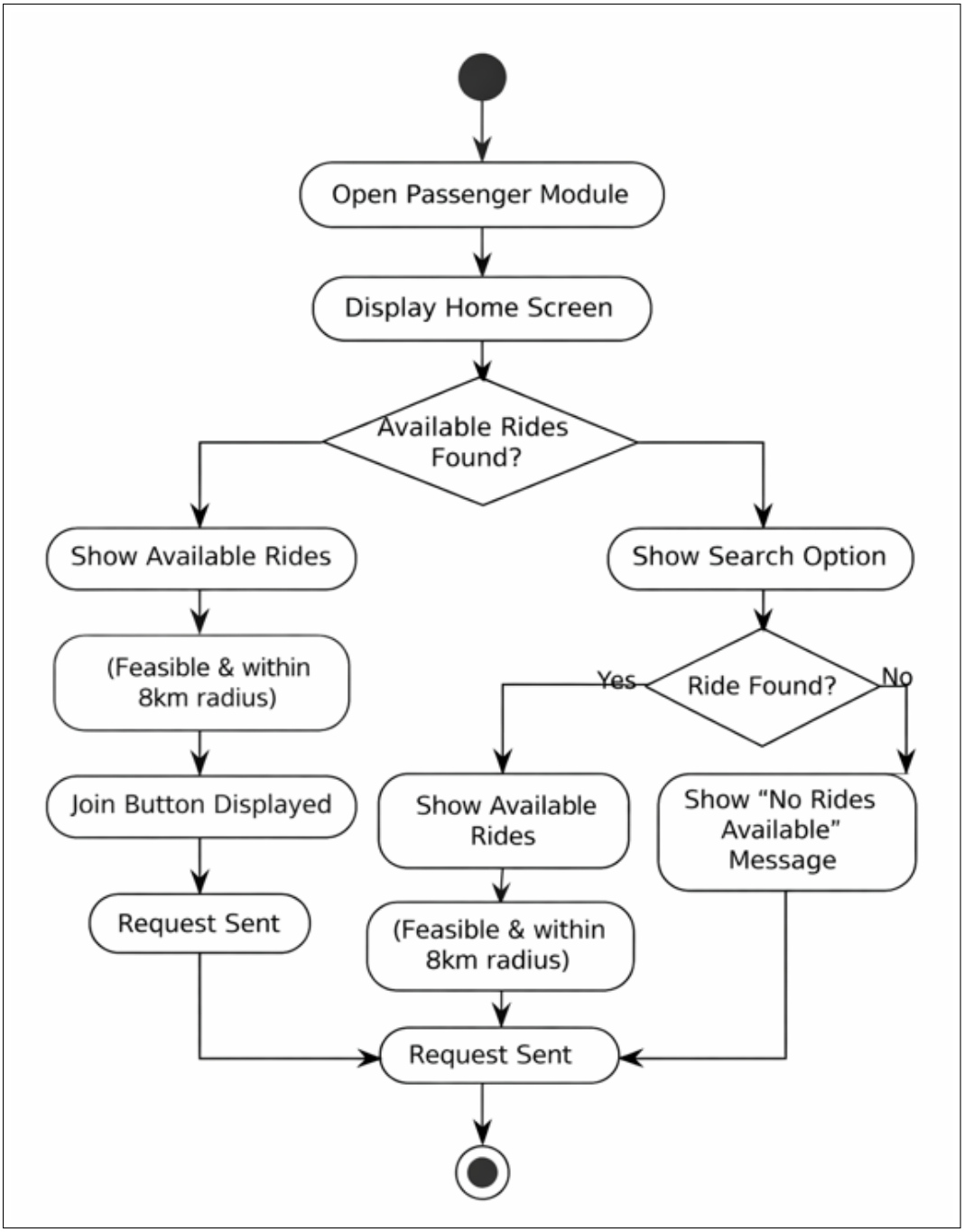


Figure 4.3: Passenger Activity Diagram

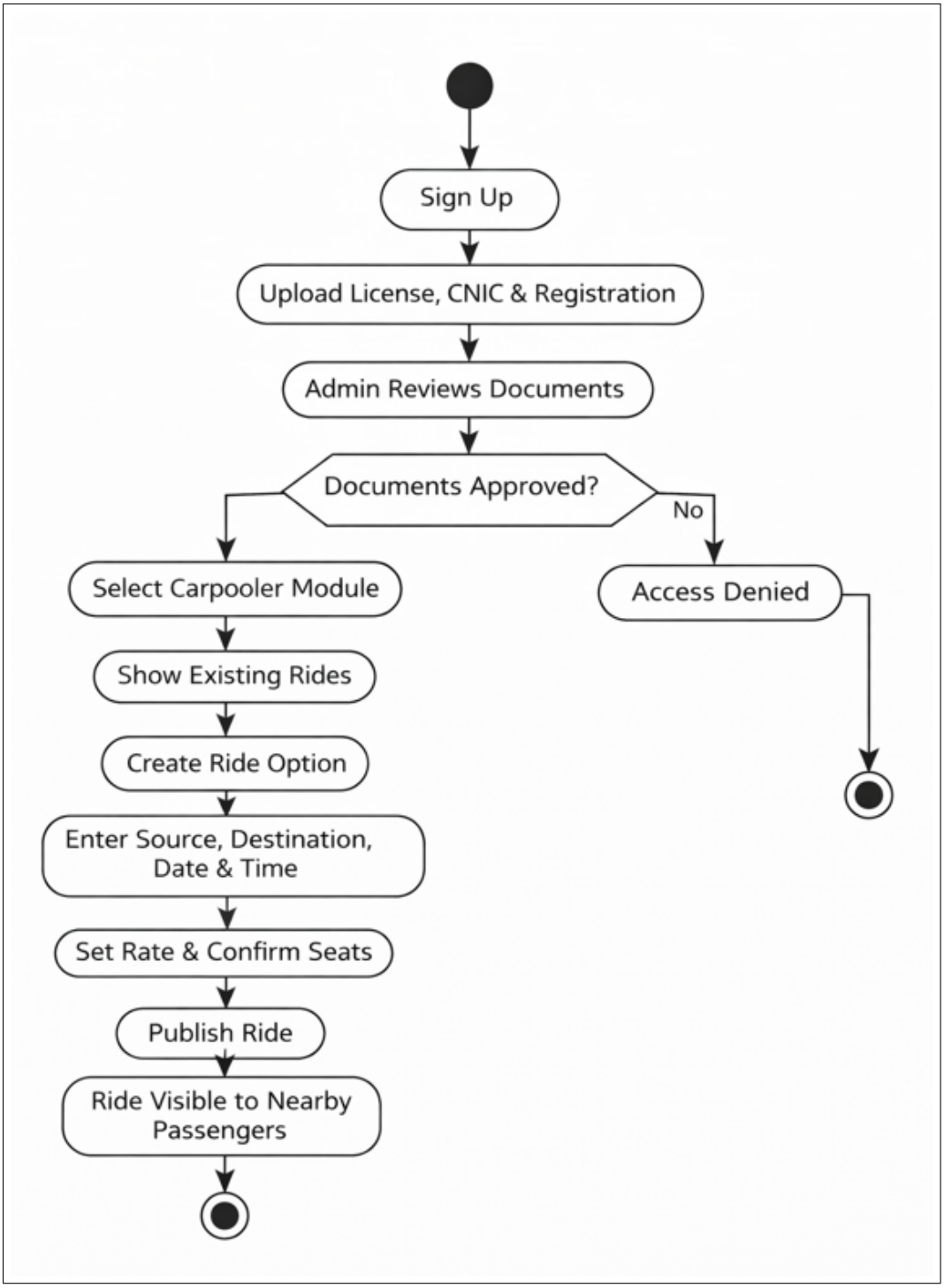


Figure 4.4: Carpooler Activity Diagram

## 4.5 Low Level Design

The low-level design defines specific structures of each component, that is, divides them into smaller units that can be modules and functions. The modules are well defined, which

makes the code reusable, maintainable and easier to test. Such modularization helps to ensure real-time updates, erroneous processing, and easy user interaction. Sequential diagrams are also employed to show how certain processes such as posting rides, search, tracking, payments and notifications move within the system.Figures 4.5 to 4.10 illustrate the sequence diagrams for these key system processes.Additionally, the low-level design defines the fine logic, data model, and interfaces of every module that assists developers to realize the system in a consistent and efficient manner. It also separates high-level architecture and real code, minimizing errors in implementation, and allowing the design of components which can be developed, debugged, and integrated separately.

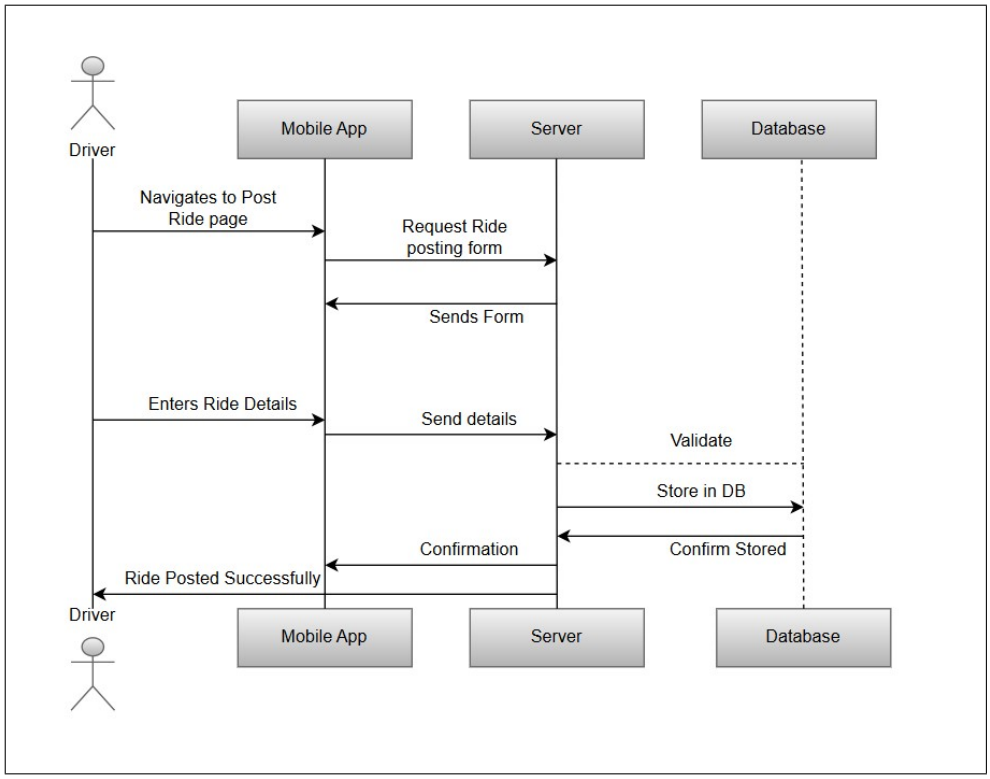


Figure 4.5: Posting a Ride

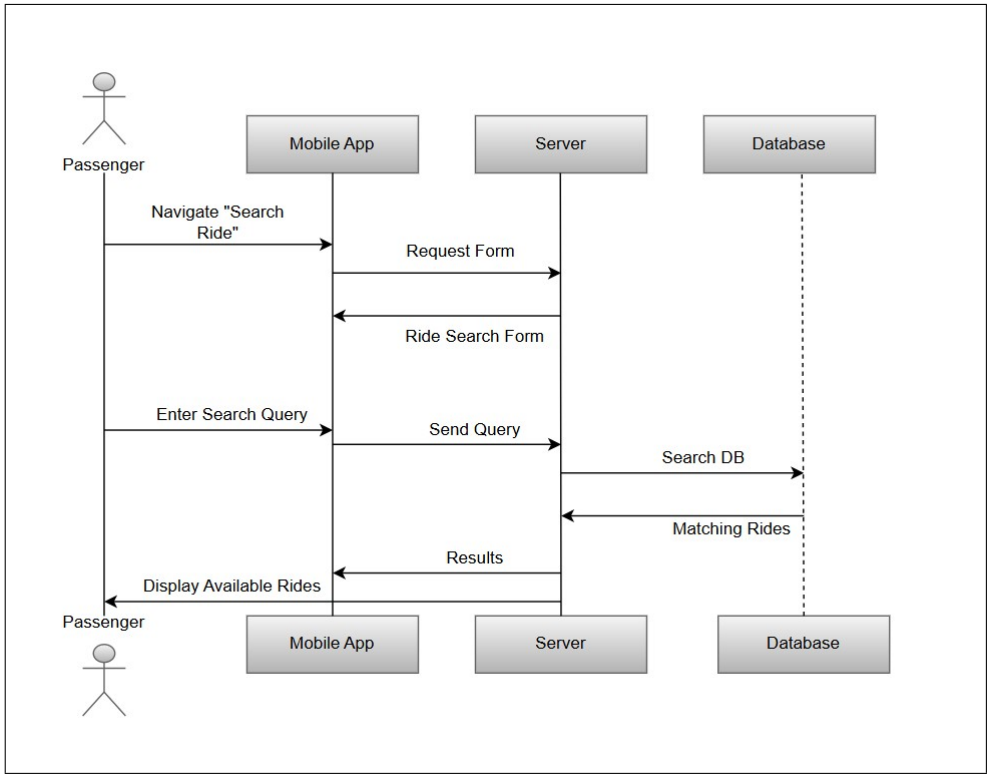


Figure 4.6: Searching a Ride

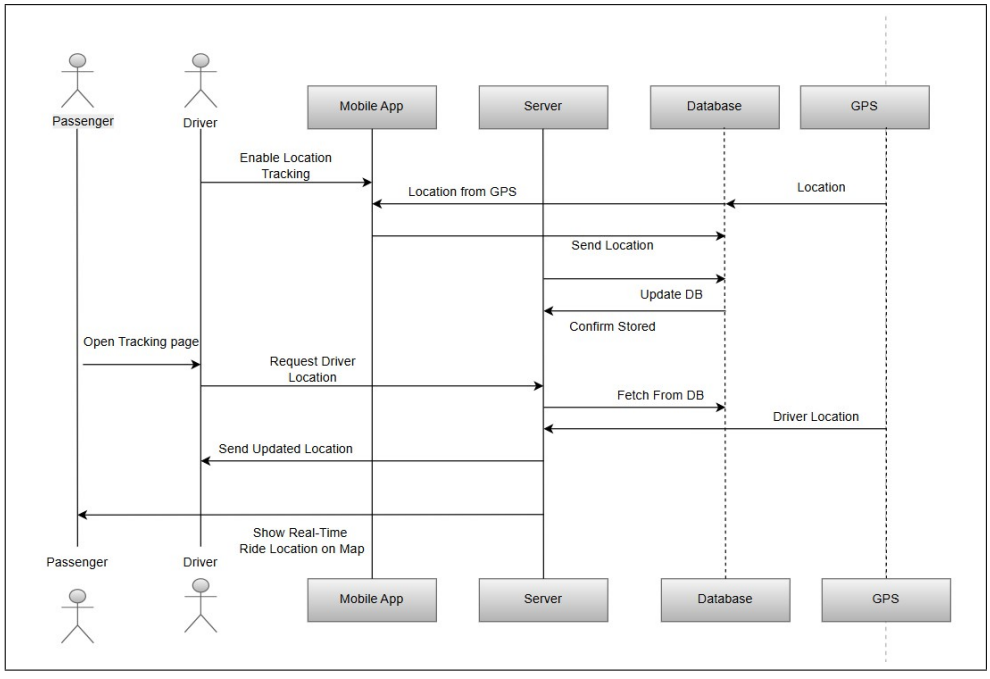


Figure 4.7: Real-Time Tracking

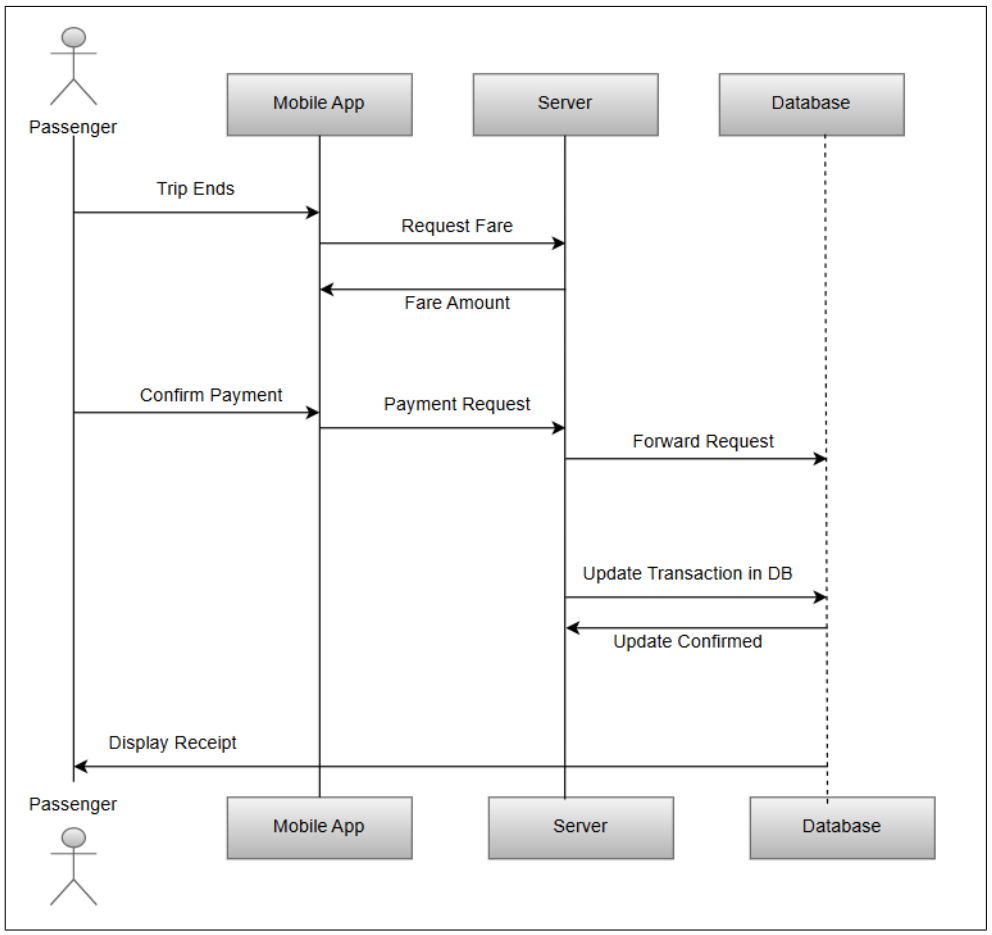


Figure 4.8: Smart Fare Calculation

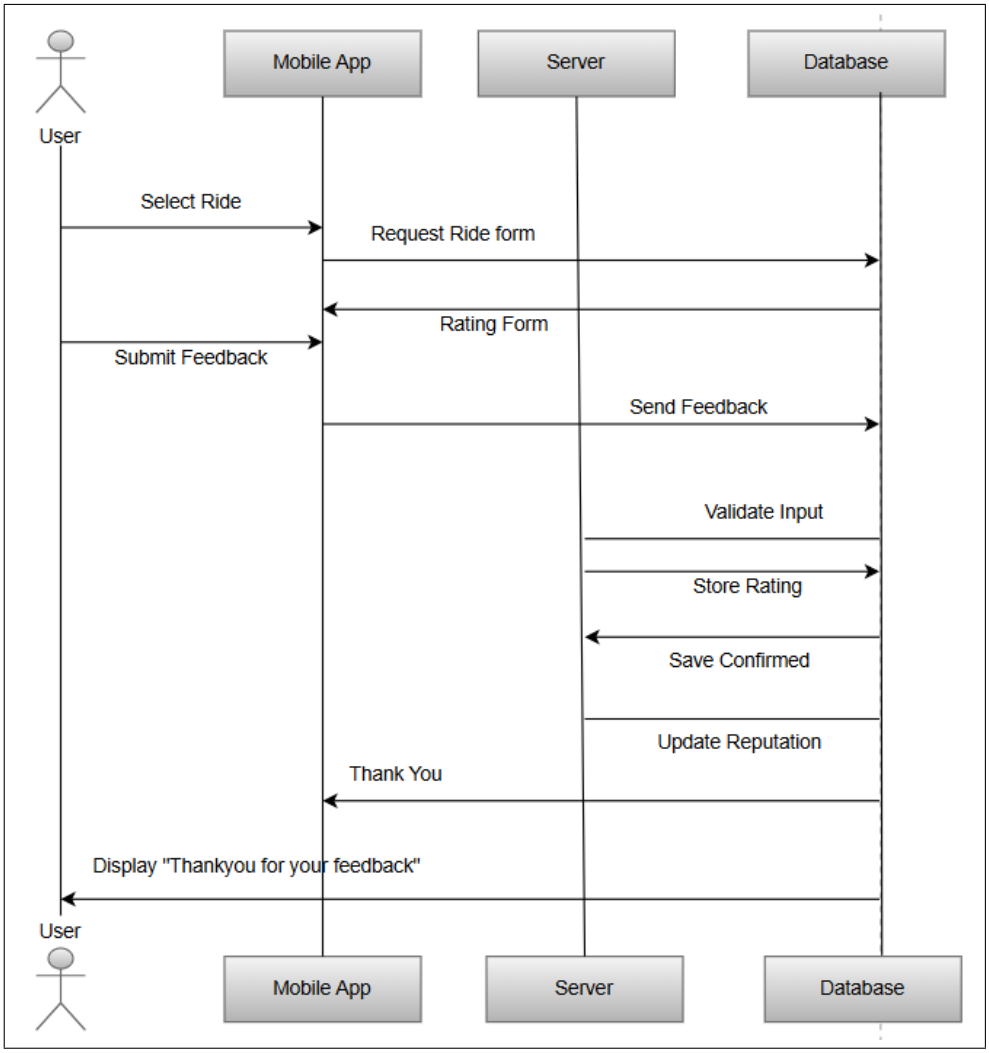


Figure 4.9: Rating and Review

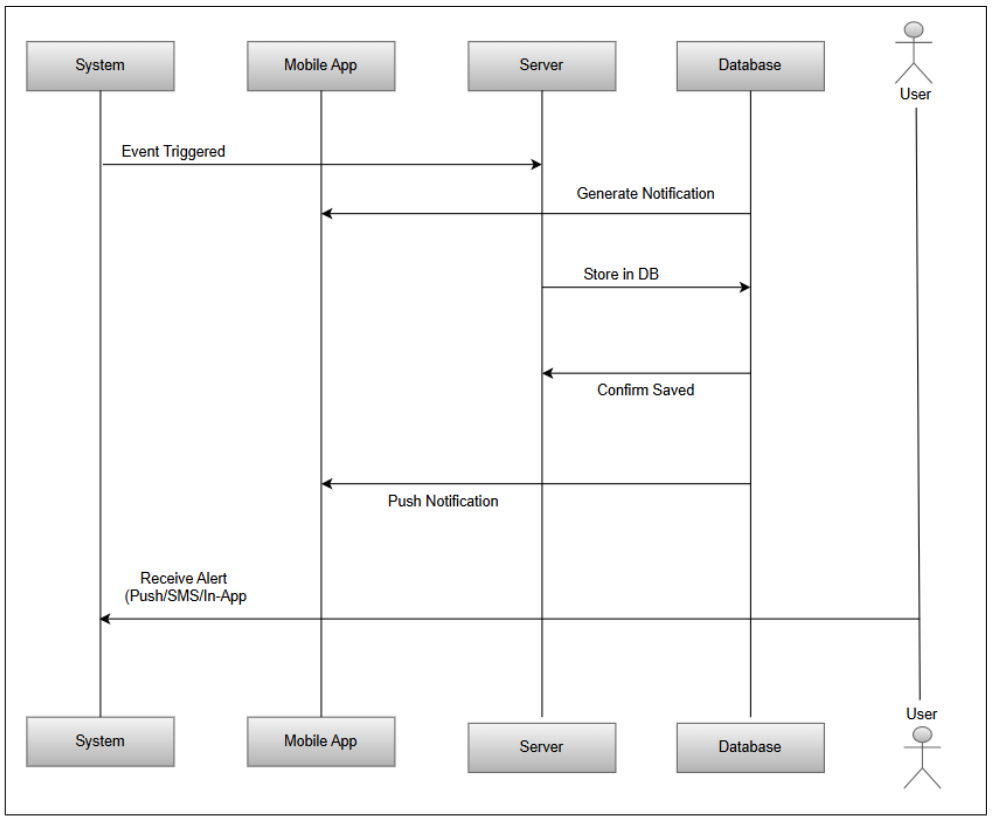


Figure 4.10: Notifications

## 4.6 Database Design

Carpooling Connect system is configured with a robust and well-structured database to store all the important data pertaining to its main features, users, rides, payments, ratings and reviews, notifications. Supabase is selected as a backend because it is scaled, flexible, and its relational data processing is robust, and using the PostgreSQL. It is completely aligned with the mobile application, which guarantees great reliability, safe storage, and efficient functioning even when the number of users and requests related to the ride rises. It has been designed in a way that facilitates easy management of real time rides, driver and rider authentication, payment and system wide notifications to facilitate a hassle free and safe user experience.

## 4.7 GUI Design

The graphical user interface (GUI) of Carpooling Connect is designed to be simple, modern, and intuitive. The app uses clean layouts, smooth navigation, and minimal input steps to improve usability. Color coding, icons, and notifications are used to enhance readability and

quick decision-making. Special focus is given to accessibility so that all users, including those with special needs, can benefit from the platform.

### 4.7.1 Mobile App Interface

The mobile interface will be fashioned with:

**User-Friendly Navigation:** The app is well structured and clean and can be easily understood by any user. There are only simple menus and clearly labeled buttons that help users to navigate through features easily. Frequent action shortcuts, which include posting a ride or checking the bookings, save time and make navigation fast and easy.

**Search Functionality:** The search tool will enable the passengers to search rides by applying various parameters, including time, place, seat availability, and gender. This makes sure that one gets results that correspond to personal requirements and make things more comfortable. The exhibited rides are clearly arranged and users can easily compare and make the necessary decisions.

**Virtual Assistant Integration:** The virtual assistant, which operates on an API, is going to be integrated in a way that this interaction process is not too pronounced so that users can interact with it when using the app. It enables the passengers and the drivers to find the information concerning Carpooling Connector, get the quick and precise results, and get the immediate assistance without browsing the various menus. In addition to making ride reservations, the assistant has the ability to assist users in other questions, including the analysis of travel patterns or valuable information on safety and other payment methods, to ensure that experience is smooth and dependable.

Figures 4.11 to 4.15 illustrate the GUI screens of the Carpooling Connect mobile and web application.

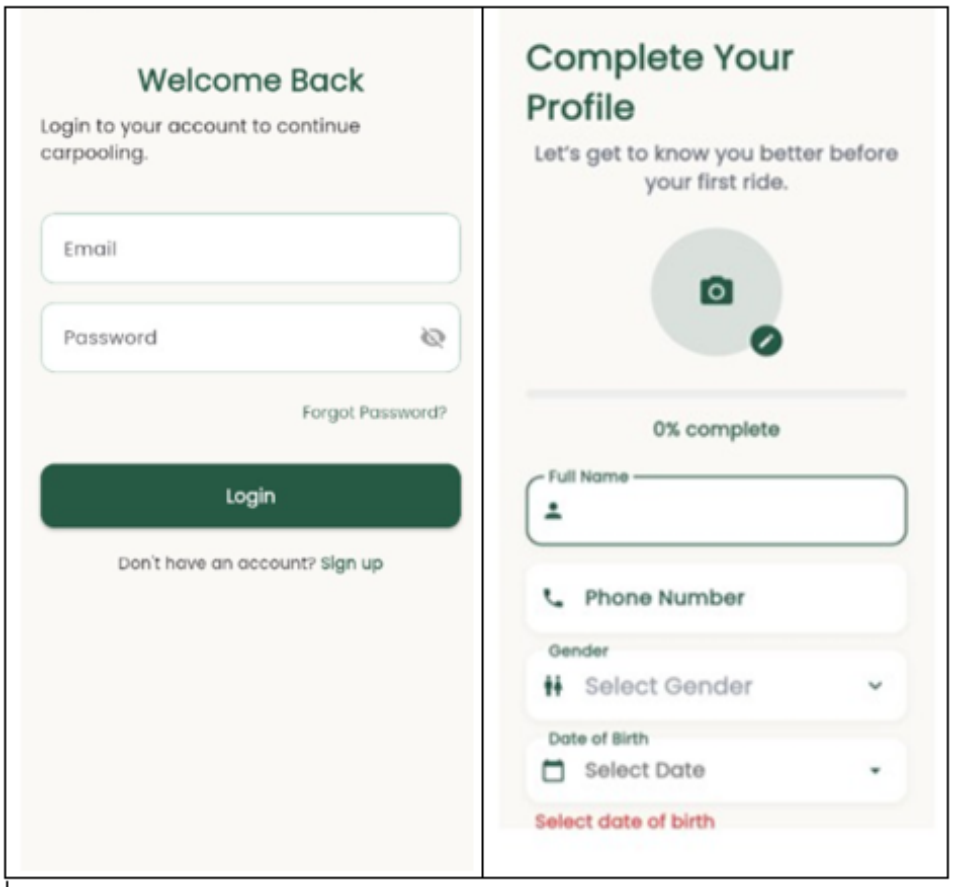


Figure 4.11: Screens for Carpooling Connect Mobile Application

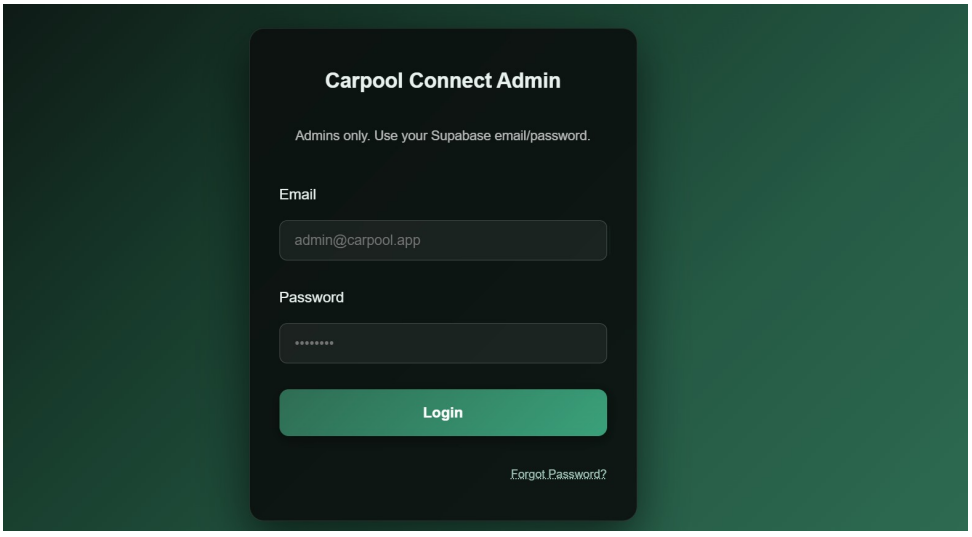


Figure 4.12: Screens for Carpooling Connect Web Admin Panel

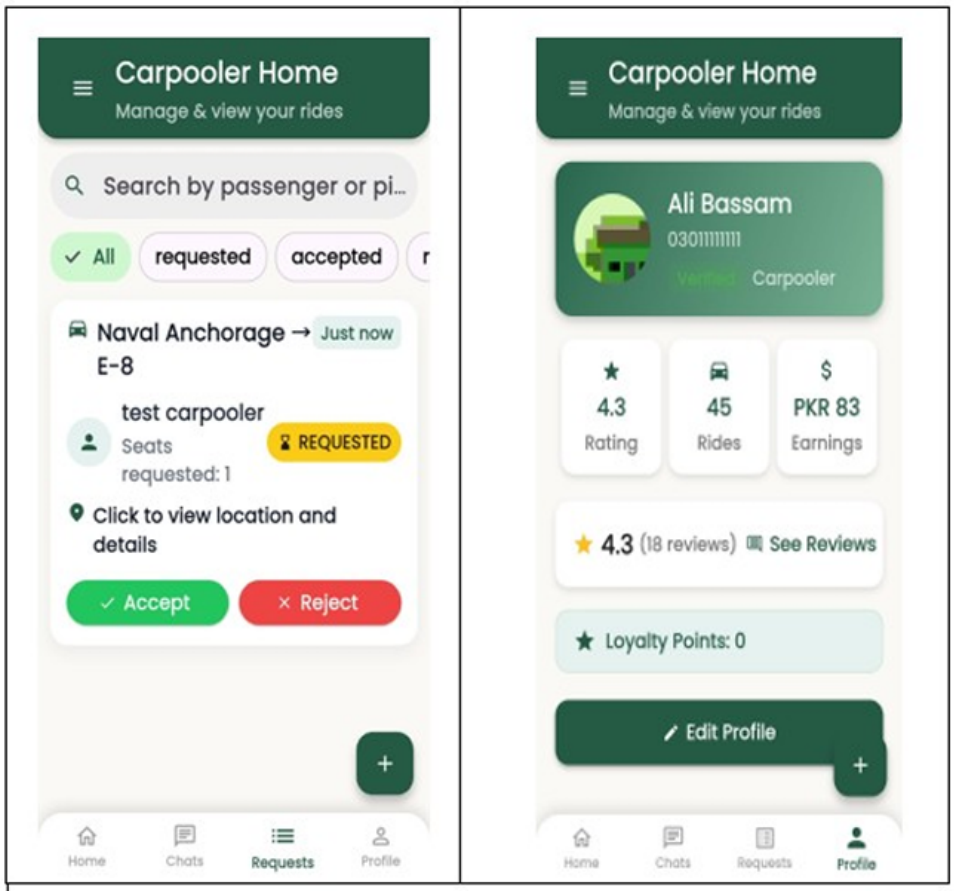


Figure 4.13: Screens for Carpooling Connect Mobile Application

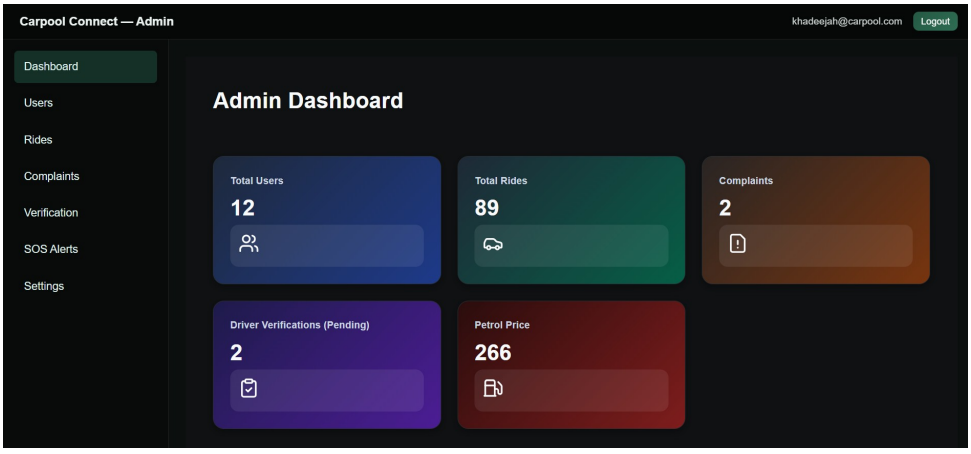


Figure 4.14: Screens for Carpooling Connect Web Admin Panel

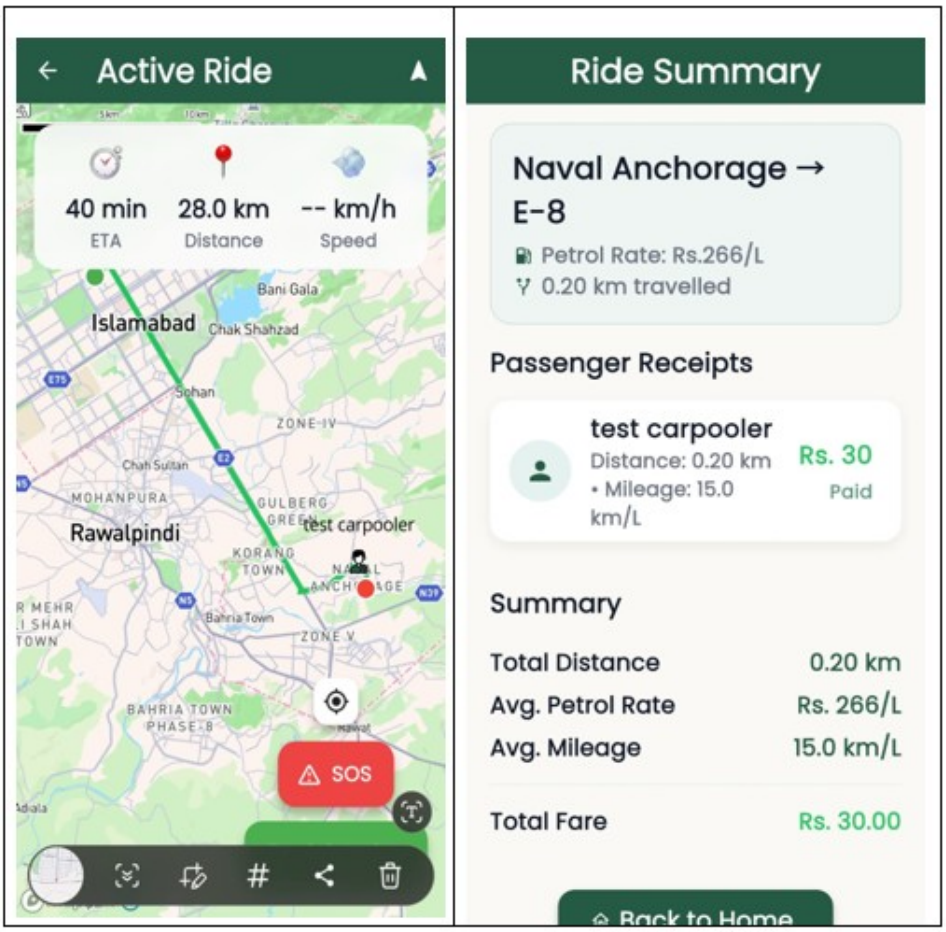


Figure 4.15: Screens for Carpooling Connect Mobile Application

## 4.8 External Interfaces

Carpooling Connect interacts with several external systems:

- **Maps and Navigation:**MapBox and Google Maps APIs to guide and track routes.
- **Geo-location Services:** GeoLocator APIs to fetch real-time locations.
- **Notification Services:** Firebase Cloud Messaging (FCM) for push notifications.
- **Authentication Services:** Supabase Auth for secure login and identity management.

## 4.9 Summary

The chapter gave a description of system design of Carrpooling connect i.e. architecture, design constraints and methodology. The three-level architecture makes it modular and scalable, whereas the mobile application offers an easy-to-use platform to communicate.

The constraints of the design were well-balanced with the performance, security, and scalability. Agile methodology is useful in prototyping fast, delivering in bits, and working together with stakeholders. The high-level design (functional view, data model, and API design) and low-level design (modular components and sequential flows) were described to be clear in terms of implementation. The GUI and database architecture also ensure usability, access, and reliability, which are the basis of a scalable and secure carpooling solution.

# Chapter 5

# System Implementation

The method of turning an idea into reality is called implementation. The system implementation is the process of using programming and deployment to make a technical specification or methodology a program, software component, or other computer system.

## 5.1 System Architecture

Implementation stage is the one at which the design of Carpooling connect is converted into a work system. The various components of the application frontend, backend and database and security layers are all integrated at this stage to deliver a complete and working platform. Figure 5.1 shows the UML Diagram of the Carpooling Connect system. This is aimed at making the system operate efficiently, maintain all the features that it is intended to offer and provide the drivers and the passengers with a sense of reliability. This chapter describes the system architecture configuration, tools, key internal components and how these components interact to provide the desired functionalities.

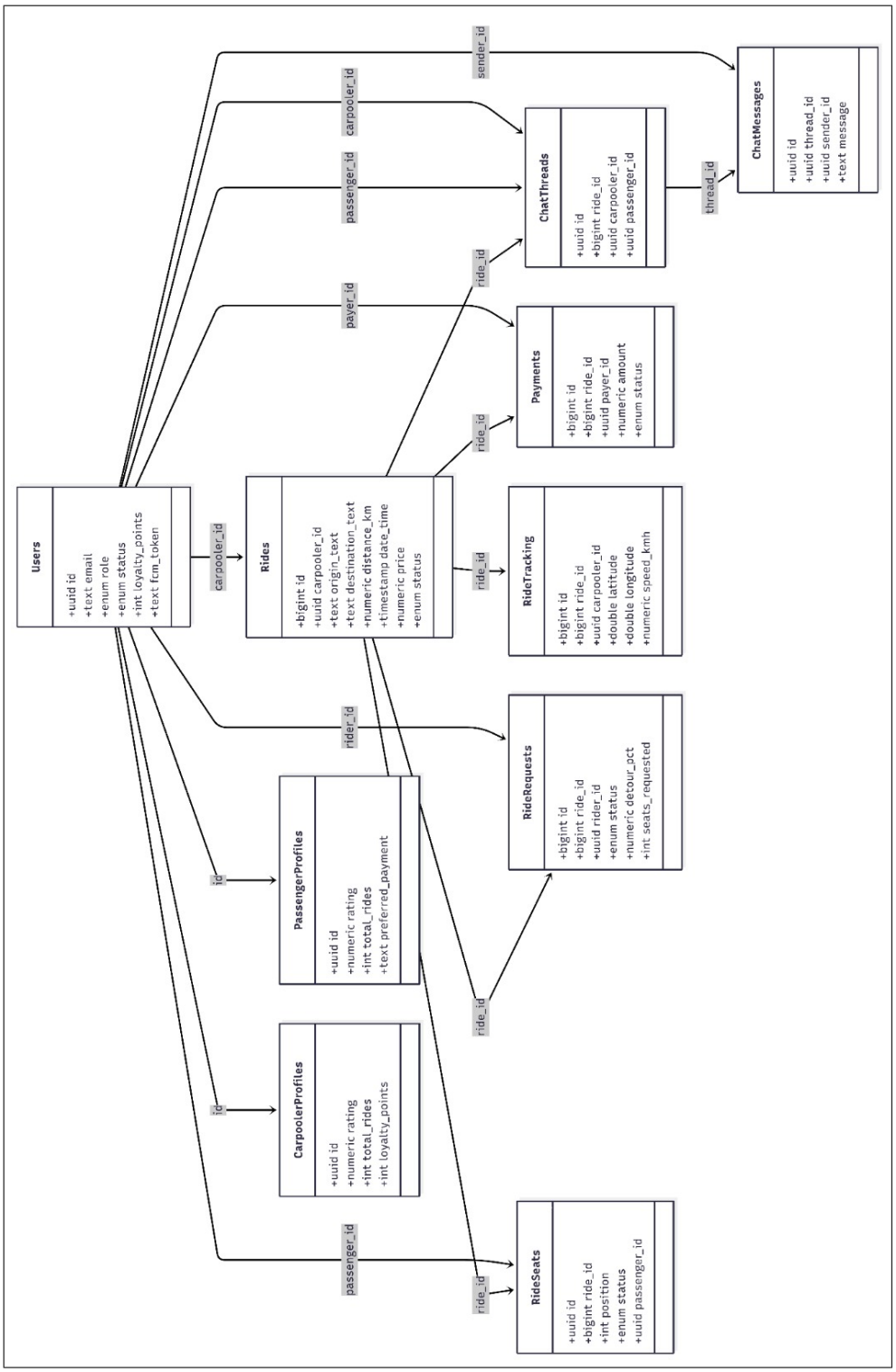


Figure 5.1: UML Diagram of Carpooling Connect System

Carpooling connect is a client server based application. The client system consists of the mobile application (Flutter) and the administration panel (React), and the backend, which runs on Supabase, serves to handle data, authentication, business logic. The database layer is stored in the PostgreSQL database to store user, ride and payment information. The system with authentication and encryption of information and transactions is used to ensure safe communication.

## 5.2 Tools and Technology Used

**I. Mobile Application Development:**
The mobile app is built in Flutter where it supports cross-platforms and features responsive and high-quality user interface. The administrative panel is developed on the basis of React, where the web-based dashboard is offered to observe and regulate the system.

**II. Backend:**
Supabase is used as the server platform, which provides authentication, triggers, functions, and effective security features to enable smooth and secure operations.

**III. Database:**
The data, which is managed through Supabase, is stored in PostgreSQL that can store structured data including user profiles, ride details, payments, reviews and notifications making it reliable, scalable and data management is secure.

**IV. Integrated Development Environment (IDE):**
Development was done with Visual Studio Code and Android Studio and testing environments with emulators were provided both to mobile and web applications.

**V. Version Control:**
GitHub enables collaborative work, versioning of code and tracking of changes, makes certain of seamless team work and management of projects.

**VI. Cloud Hosting:**
Supabase offers cloud computing infrastructure, which offers scalability of backend services and high availability to the application.

**VII. Route Optimization:**
Map box root optimization API is to be applied in route optimization to provide an efficient matching of rides and decrease time and traveling distances.

**VIII. Additional Tools:**
Mapbox and Geolocator are both combined to track the location in real-time, plan a route, and use a precise navigation to pick up/drop-off locations. FCM facilitates real-time notification of updates on rides, booking confirmation, and notifications.

## 5.3 System Internal Components

### 5.3.1 Front-End Components

All of the user-facing options, such as posting rides, searching, tracking, payments, and notifications are offered through the mobile application. React admin panel will enable the administrators to control the users, track rides, and maintain the compliance with the rules. The two interfaces are also device friendly and responsive.

### 5.3.2 Back-End Components

Supabase takes care of the backend, including authentication, authorization and logic of the system. It also makes use of triggers and functions to handle tasks such as ride validation, booking update, and sending notifications.

### 5.3.3 Database Layer

The database is PostgreSQL where all the data is stored, including user data, rides, bookings, payments, ratings, and notifications.

## 5.4 Application Access Security

Security is a major concern in Carpooling connect to make sure the user is confident of the security of its operations. The system ensures that all user information is securely logged in using encrypted ways and communication in order to avoid unauthorized access. Access control is enforced such that only the legitimate users can view or modify certain information. Payment details and other personal identities are sensitive information that is well preserved with high level of protection of data.

## 5.5 Database Security

The postgreSQL database layer is housed on Supabase. All user information, rides, reservations, payments, reviews, and messages are secured. Sensitive information is safeguarded through authentication, authorization and encryption. Authorized users have access to some form of information and stringent regulations of information storage add to the security.

## 5.6 Processing Logic / Algorithms

### 5.6.1 User Profiles and Authentication

Users sign up using CNIC, driving license and phone/email authentication. Supabase is used to perform authentication and secure sessions.

### 5.6.2 Ride Posting and Searching

The drivers are able to make ride posts that includes information such as origin, destination, time, and available seats as well as gender choice. Riders find passengers in the search with the help of filters, and the route optimization API is used to guarantee matching routes.

### 5.6.3 Real-Time Tracking

GPS integration also allows drivers to share location. Supabase manages updates in real time and gives passengers the opportunity to learn the route of the driver and their approximate arrival times.

### 5.6.4 Smart Fare Calculation

The payment is done using smart fare calculation.The transactions are kept in the PostgreSQL and passengers are issued with digital receipts.

### 5.6.5 Rating and Reviews

The users have the chance to leave ratings and feedback after every trip. It is a system that provides updates to the reputation scores and establishes accountability between the driver and the passengers.

### 5.6.6 Notifications

Riders get live notifications on confirmation of ride, payment, cancellation and updates. The messages are notified through push notifications and in-app alerts.

## 5.7 Communication Between Components

The mobile application and the administration panel are connected to the backend through Supabase APIs. Any request and response is sent in an encrypted and secure fashion. Data flow is handled by the backend and storage and retrieval is done by PostgreSQL. This is appropriate communication flow that will provide rapid response and instant updates to all users.

## 5.8 Technology Stack Summary

Carpooling Connect is a blend of innovative technologies to provide a strong and scalable solution. Flutter and React have responsive and smooth interfaces, and Supabase is an authentication and backend logic manager. PostgreSQL provides the organization of data storage, smart fare calculation will allow smooth transactions, Map box provides Map and route optimization services, Geolocator is used for location services and FCM is used for real time notifications and one of the products of these technologies is a system that is growth-oriented, secure, and easy to use.

# Chapter 6

# System Testing and Evaluation

System testing and evaluation would be used to determine that the Carpooling Connect platform is working properly in various conditions and that it satisfies the needs of the user. This is done to ensure the functionality, usability, and performance of the mobile application and the admin panel. This chapter is aimed at describing the method of testing that is applied to establish the reliability and correctness of the system.

The system testing is performed on the whole system that is already integrated to validate the system with the specifications. The analysis covers both qualitative and quantitative elements like performance, usability, stability and reliability. It is also a process that reveals the weaknesses of the system and areas that need to be improved. System testing was done using the following testing techniques.

## 6.1 Importance of System Testing

System testing will be very important in making sure that every aspect of the Carpooling Connect functions as desired. It helps to find defects, test specifications compliance, and open the platform to the real world. Without testing, small issues could be developed into large system failures, and this could impact trust and security. Therefore, testing would be required to offer a valid and convenient system.

## 6.2 Graphical User Interface (GUI) Testing

The GUI testing is done to verify the regularity, readability and functionality of visual elements of the system. Carpooling connect In the case of Carpooling Connect, it verified navigation menus, ride posting forms, and live maps. The tests were made such that buttons, text fields, and icons acted and were accessible on various devices, which offered easy and user-friendly experience.

## 6.3 Usability Testing

The usability testing was done so as to test the convenience of using the app. Users were monitored in the process of posting rides, searching trips and payments. Both the new and old users gave reviews which showed that the interface was easy to use and navigate and the filters and navigation tools made it easier to access. This made the app inclusive and friendly to the user.

## 6.4 Software Performance Testing

Performance testing was done to ensure that Carpooling Connect could handle a load. The system was also tested on its speed in responding to search of rides, confirming booking, and payment. Findings demonstrated that the app was rapid and reliable in frequent use and this proved the reliability of the app to daily commuters.

## 6.5 Compatibility Testing

A compatibility test was done to ensure the app is compatible with various devices, screen sizes, and operating systems. The android platform was used to test the mobile application, and the administration area was tested using several browsers. Findings were that the platform was responsive and stable in a number of environments.

## 6.6 Exception Handling

Exception handling tests allowed the system to successfully handle errors and not crash. There were proper error messages on invalid logins and clear messages with failed payments. This directed the users in the right way in case of mistakes and ensured system stability and augmented the credibility and reliability.

## 6.7 Test Cases

Here, the test cases which were conducted in the case of Carpooling Connect are presented. Every table contains information about test ID, description, platform where the test can be conducted, preconditions, test steps, expected result, actual result, and status.

Table 6.1: Login Test Case – Successful

| | |
|---|---|
| **Test Case ID** | Login_TC_01 |
| **Description** | Verify that users can log in successfully |
| **Applicable for** | Mobile Application |
| **Pre conditions** | User account exists |
| **Test Steps** | 1. Open the app and navigate to the login page.<br>2. Enter valid credentials.<br>3. Click on the login button. |
| **Expected Result** | User should be logged in successfully |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.2: Login Test Case – Failed

| | |
|---|---|
| **Test Case ID** | Login_TC_02 |
| **Description** | Verify that login fails with incorrect credentials |
| **Applicable for** | Mobile Application |
| **Pre conditions** | User account exists |
| **Test Steps** | 1. Open the app and navigate to the login page.<br>2. Enter invalid email or password.<br>3. Click on the login button. |
| **Expected Result** | System should display “Login Failed” message |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.3: Passenger Analytics Test Case – Successful

| | |
|---|---|
| **Test Case ID** | PassengerAnalytics_TC_01 |
| **Description** | Verify that users can view and analyze passenger ride records in the application. |
| **Applicable for** | Passenger Mobile Application |
| **Pre conditions** | Passenger ride records must exist in the system. |
| **Test Steps** | 1. Log in to the Passenger Mobile Application.<br>2. Navigate to the Passenger Analytics section.<br>3. Select and view the available analytics dashboard. |
| **Expected Result** | The system should display passenger ride records along with trip history, distance, and cost details. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.4: Passenger Analytics Test Case – Failed

| | |
|---|---|
| **Test Case ID** | PassengerAnalytics_TC_02 |
| **Description** | Verify system behavior when passenger analytics data is unavailable or fails to load. |
| **Applicable for** | Passenger Mobile Application |
| **Pre conditions** | Passenger ride records must exist in the system. |
| **Test Steps** | 1. Log in to the Passenger Mobile Application.<br>2. Navigate to the Passenger Analytics section.<br>3. Attempt to view the analytics dashboard. |
| **Expected Result** | The system should display passenger ride records with correct details. |
| **Actual Result** | The system fails to retrieve or display passenger analytics. Trip details or history are missing or not shown correctly. |
| **Status** | Fail |

Table 6.5: Carpooler Analytics Test Case – Successful

| | |
|---|---|
| **Test Case ID** | CarpoolerAnalytics_TC_01 |
| **Description** | Verify that users can view and analyze carpooler ride records in the application. |
| **Applicable for** | Carpooler Mobile Application |
| **Pre conditions** | Carpooler ride records must exist in the system. |
| **Test Steps** | 1. Log in to the Carpooler Mobile Application.<br>2. Navigate to the Carpooler Analytics section.<br>3. Select and view the available analytics dashboard. |
| **Expected Result** | The system should display carpooler ride records along with trip counts, earnings, and distance covered. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.6: Carpooler Analytics Test Case – Failed

| | |
|---|---|
| **Test Case ID** | CarpoolerAnalytics_TC_02 |
| **Description** | Verify system behavior when carpooler analytics data is unavailable or fails to load. |
| **Applicable for** | Carpooler Mobile Application |
| **Pre conditions** | Carpooler ride records must exist in the system. |
| **Test Steps** | 1. Log in to the Carpooler Mobile Application.<br>2. Navigate to the Carpooler Analytics section.<br>3. Attempt to view the analytics dashboard. |
| **Expected Result** | The system should display carpooler ride records with correct details. |
| **Actual Result** | The system fails to retrieve or display carpooler analytics. Ride counts or earnings are missing or not shown correctly. |
| **Status** | Fail |

Table 6.7: Admin Panel Analytics Test Case – Successful

| | |
|---|---|
| **Test Case ID** | AdminAnalytics_TC_01 |
| **Description** | Verify that the admin can successfully view and analyze system-wide analytics including passengers and carpoolers. |
| **Applicable for** | Admin Panel |
| **Pre conditions** | Passenger and carpooler activity data must exist in the database. |
| **Test Steps** | 1. Log in to the Admin Panel.<br>2. Navigate to the Analytics Dashboard section.<br>3. Select and view system-wide analytics. |
| **Expected Result** | The system should display accurate analytics, including total rides, payments, ratings, and usage statistics. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.8: Admin Panel Analytics Test Case – Failed

| | |
|---|---|
| **Test Case ID** | AdminAnalytics_TC_02 |
| **Description** | Verify system behavior when admin analytics data is unavailable or fails to load. |
| **Applicable for** | Admin Panel |
| **Pre conditions** | No analytics data or system failure in data retrieval. |
| **Test Steps** | 1. Log in to the Admin Panel.<br>2. Navigate to the Analytics Dashboard section.<br>3. Attempt to view analytics data. |
| **Expected Result** | The system should notify the admin with a ‘No Data Available” or relevant error message. |
| **Actual Result** | The system fails to retrieve or display analytics. Dashboard shows missing or incorrect data. |
| **Status** | Fail |

Table 6.9: Ride Posting Test Case – Successful

| | |
|---|---|
| **Test Case ID** | RidePost_TC_01 |
| **Description** | Verify that a user can successfully post a ride. |
| **Applicable for** | Mobile Application |
| **Pre conditions** | User account exists and user is logged in. |
| **Test Steps** | 1. Navigate to the Ride Posting section.<br>2. Enter valid ride details (pickup, drop-off, time, seats).<br>3. Submit the ride post. |
| **Expected Result** | Ride should be posted successfully and visible in the ride listings. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.10: Ride Posting Test Case – Failed

| | |
|---|---|
| **Test Case ID** | RidePost_TC_02 |
| **Description** | Verify system behavior when invalid or incomplete ride details are entered. |
| **Applicable for** | Mobile Application |
| **Pre conditions** | User account exists and user is logged in. |
| **Test Steps** | 1. Navigate to the Ride Posting section.<br>2. Enter invalid/incomplete ride details (e.g., missing pickup location).<br>3. Submit the ride post. |
| **Expected Result** | The system should reject the request and display an error message. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.11: Ride Search Test Case – Successful

| | |
|---|---|
| **Test Case ID** | RideSearch_TC_01 |
| **Description** | Verify that users can search for available rides successfully. |
| **Applicable for** | Mobile Application |
| **Pre conditions** | Active rides exist in the system. |
| **Test Steps** | 1. Navigate to the Ride Search section.<br>2. Enter valid search criteria (pickup, drop-off, time).<br>3. View the list of available rides. |
| **Expected Result** | Matching rides should be displayed correctly. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.12: Ride Search Test Case – Failed

| | |
|---|---|
| **Test Case ID** | RideSearch_TC_02 |
| **Description** | Verify system behavior when no rides match the search criteria. |
| **Applicable for** | Mobile Application |
| **Pre conditions** | No rides exist for the given criteria. |
| **Test Steps** | 1. Navigate to the Ride Search section.<br>2. Enter invalid or unmatched search criteria.<br>3. View search results. |
| **Expected Result** | System should display ‘No Rides Available” message. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.13: Payment Test Case – Successful

| | |
|---|---|
| **Test Case ID** | Payment_TC_01 |
| **Description** | Show the ride receipts to their respective passengers. |
| **Applicable for** | Mobile Application |
| **Pre conditions** | User is logged in and ride is completed. |
| **Test Steps** | 1. Book a ride.<br>2. Proceed to payment.<br>3. View Receipt.<br>4. Confirm transaction. |
| **Expected Result** | Payment should be processed successfully, and receipt displayed. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.14: Payment Test Case – Failed

| | |
|---|---|
| **Test Case ID** | Payment_TC_02 |
| **Description** | Verify system behavior when invalid payment details are entered. |
| **Applicable for** | Mobile Application |
| **Pre conditions** | User is logged in. |
| **Test Steps** | 1. Book a ride.<br>2. Proceed to payment.<br>3. Invalid route.<br>4. Confirm transaction. |
| **Expected Result** | System should reject payment and display error message. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.15: Notification Test Case – Successful

| | |
|---|---|
| **Test Case ID** | Notification_TC_01 |
| **Description** | Verify that users receive notifications for ride updates. |
| **Applicable for** | Mobile Application |
| **Pre conditions** | User account exists and rides are active. |
| **Test Steps** | 1. Book a ride.<br>2. Wait for ride confirmation or updates.<br>3. Check notification panel. |
| **Expected Result** | User should receive a notification about ride status. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.16: Notification Test Case – Failed

| | |
|---|---|
| **Test Case ID** | Notification_TC_02 |
| **Description** | Verify system behavior when notification service fails. |
| **Applicable for** | Mobile Application |
| **Pre conditions** | User account exists, but notification service is unavailable. |
| **Test Steps** | 1. Book a ride.<br>2. Wait for ride confirmation or updates.<br>3. Check notification panel. |
| **Expected Result** | User should see fallback messages in the app (e.g., ride status in dashboard). |
| **Actual Result** | As expected |
| **Status** | Pass |

## 6.8 Summary

This chapter presented the testing of Carpooling Connect, which was based on functional accuracy, usability, performance, compatibility, and exception handling. A number of test cases such as login, analytics and the administrator panel cases ensured that the system acted as intended when under various conditions. The findings indicated that the platform is reliable, stable and user friendly, thus it can be deployed in actual commuting conditions.

# Chapter 7

# Conclusion

## 7.1 Conclusion

Carpooling Connect system has been able to fulfill its mission, which is to offer a secure, efficient, and user-friendly platform that matches passengers with carpoolers. The system is supported by the use of modern technologies, including Flutter for the mobile application, React for the admin panel, and Supabase as a backend with PostgreSQL, to provide smooth system operation with real-time updates and scalability.

Analytics integration assists passengers and carpoolers in tracking their activities and enhances the overall experience. Ride posting, ride searching, booking, payments, and notifications were successfully implemented and tested.Authentication and encryption mechanisms were put in place and this enhanced the security level in the system and gave confidence to the users.

It further got tested in terms of usability where it was confirmed that the interface is easy and user-friendly with first-time users finding it easy to navigate through the system. The system demonstrated it can manage many concurrent requests during performance testing, though a considerable delay was observed.. The compatibility testing also proved that the system is compatible with different devices and platforms. Exception handling meant that the errors were handled graciously and the relevant feedback was given to the users.

Altogether, Carpooling Connect is an effective, scalable and safe service to alleviate the problem of transportation in a city, to facilitate the interests of saving, convenience, and friendliness to the environment. This project illustrates the way in which technology may be implemented to facilitate smarter mobility and community based transport solutionss.

## 7.2 Future Work

Although the features that are provided by the Carpooling Connect system have the most important features and consistent architecture, there are several points that can be

improved in the future. To better the system, enhanced ride-matching algorithm with artificial intelligence can be incorporated to achieve efficiency in the carpooler matches and minimise the waiting time. Real-time location tracking with GPS will also help with improving transparency and safety in rides.

The rating and review option can be enhanced with the possibility of more detailed feedback that will strengthen the user trust even more. Multilingual support can also be added to the future releases to enable the platform to be accessible to other users. To contain the payment flexibility can be added by incorporating digital wallets and other safe gateways.

Push notification could be designed to be more lively to incorporate ride reminders, fare updates and emergency alerts. It is possible to improve the Supabase and React-based administration panel by adding sophisticated analytics to assist in tracking user behavior, abnormal activities, and policy violations. Online support can be included to make the user view and reserve rides even where there is low connectivity. In addition, the gamification can be provided with reward points or discounts, which will make more people adopt carpooling.

In general, the further development of the Carpooling Connect is expected to be aimed at building the user confidence, enhancing the operation effectiveness, and ensuring the sustainability of the urban mobility with the help of the constant innovation.

# Appendix A

# User Manual

The User Manual will offer instructions on how a user can use the Carpooling Connect mobile app and administration panel. The manual covers the requirements of the systems, registering of accounts, main features of the system to the passengers and carpoolers, and administration. It is to aid the users in navigating the system and learning its features.

## A.1 System Requirements

Carpooling Connect application requires the following to operate:

- **Device:** Android phone with Android 9.0 or higher.
- **Network:** Wi-Fi or mobile data connection with an active internet connection.
- **Application:** Installed Carpooling Connect mobile application.

## A.2 Getting Started

### A.2.1 Create an Account

One has to create an account in order to use the application. During registration, one will be asked to provide simple details like the name, email address, phone number, and password. The system confirms the input of the user to ascertain accuracy prior to creation of an account.

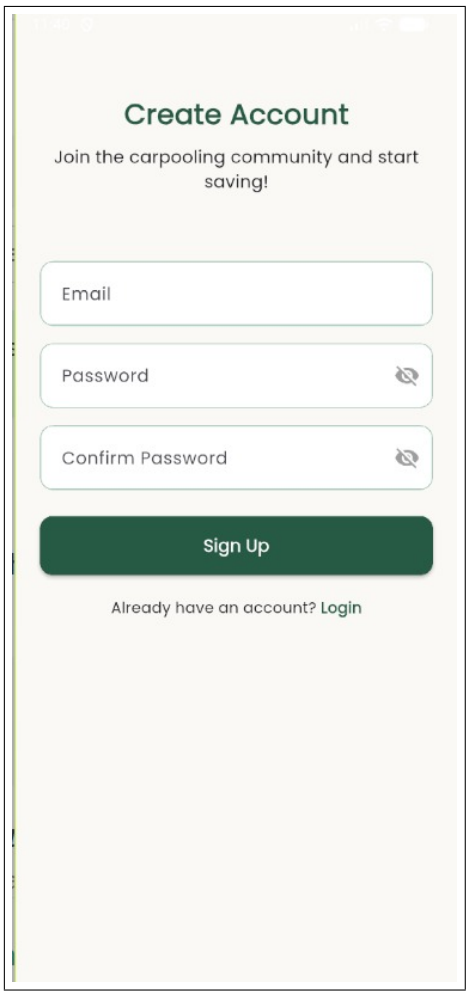


Figure A.1: Create Account Interface

### A.2.2 Log In

Registered users are able to log in with their email and password. The login interface further offers secure access to the system and provides the user with an option of retrieving his or her password in case of forgetfulness.

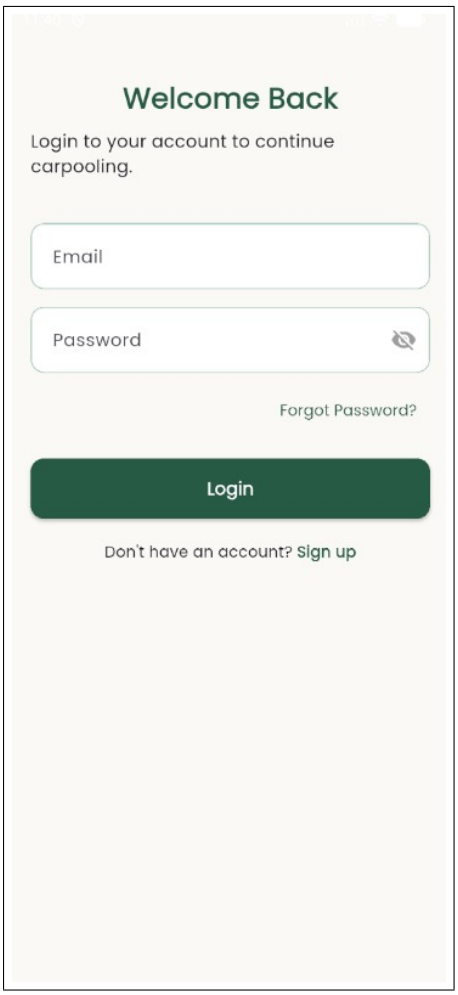


Figure A.2: Login Interface

## A.3 Using the Application

### A.3.1 Home Dashboard

Upon successful log in, the users are redirected to the home dashboard. The dashboard is easily accessible with major features being ride posting, ride searching, analytics, and profile management.

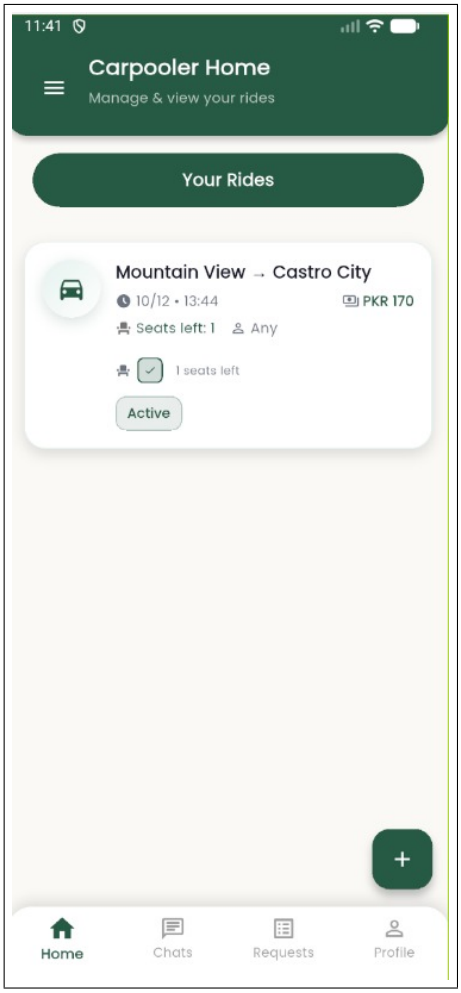


Figure A.3: Home Dashboard

### A.3.2 Posting a Ride

Carpoolers may create a ride by typing the pick-up and drop-off points, date, and time and the number of seats. The ride is made available to passengers who seek rides once it is submitted.

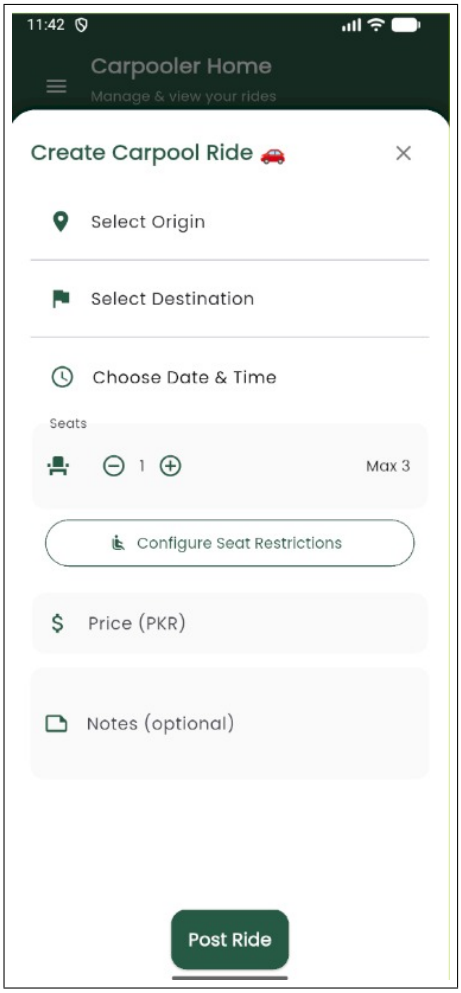


Figure A.4: Posting a Ride

### A.3.3 Searching and Booking a Ride

The passengers get to search on the available rides by entering their preferred pickup and drop-off points and time. Rides are also matched and passengers are able to book a ride using the application.

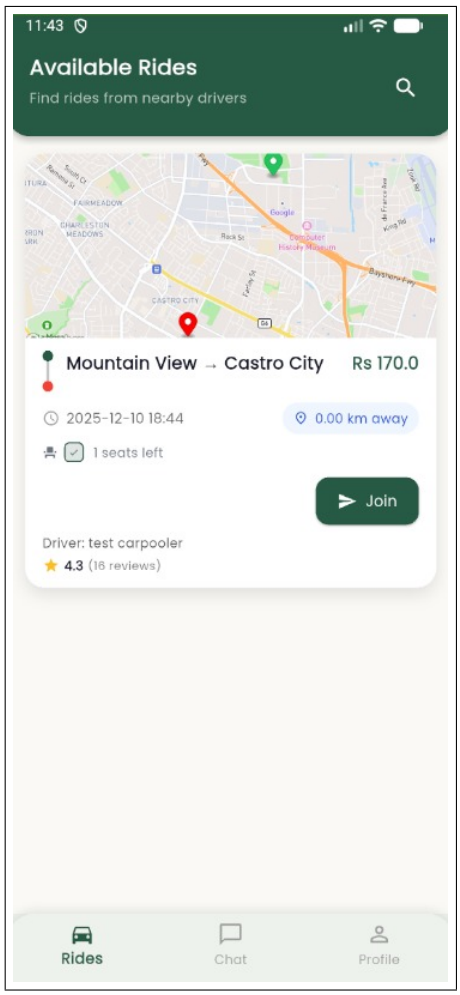


Figure A.5: Searching and Booking a Ride

### A.3.4 Real-Time Ride Tracking

After booking a ride, users are able to monitor the status of the ride in real time. The system also displays location updates and ride progress to improve the transparency and safety.

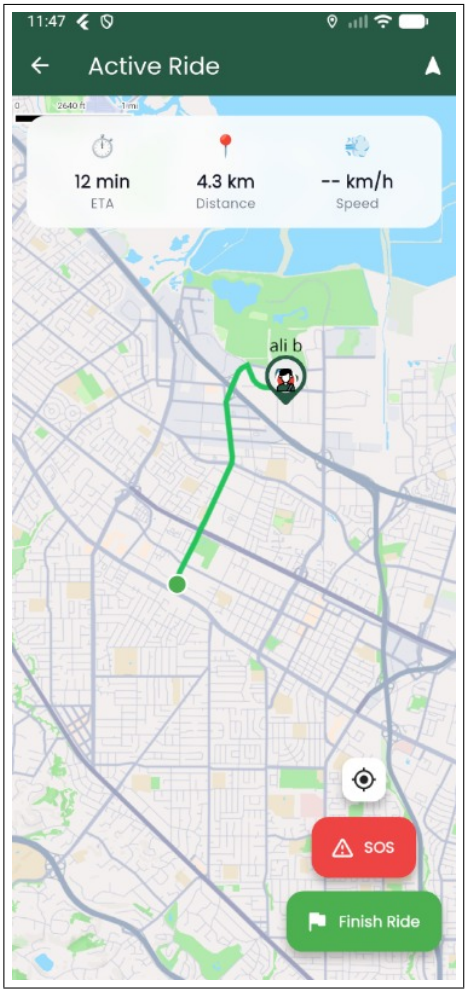


Figure A.6: Real-Time Ride Tracking

### A.3.5 Payments and Receipts

Upon leaving the ride, users will be able to see payment details and receipts. The system will guarantee secure processing of transactions and will have a record of the transactions done.

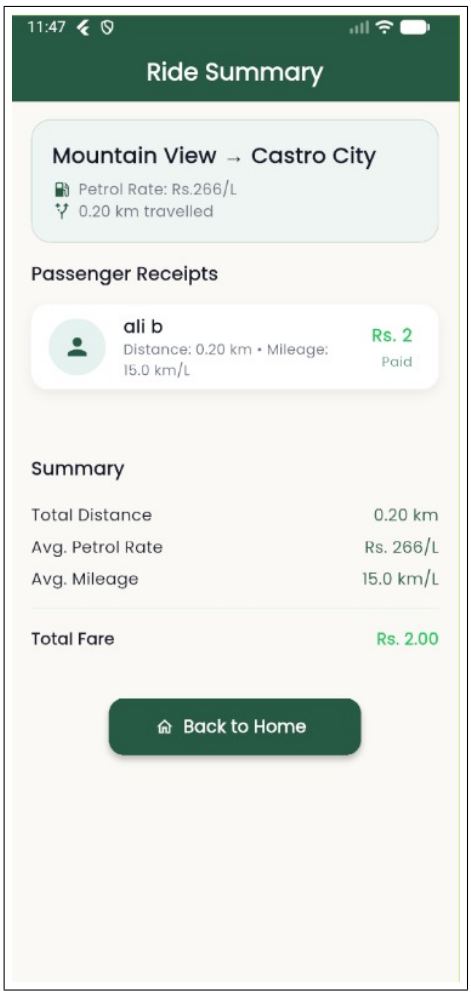


Figure A.7: Payments and Receipts

### A.3.6 Notifications

The application sends users notifications concerning ride confirmations, updates, cancellations, and essential system notifications.

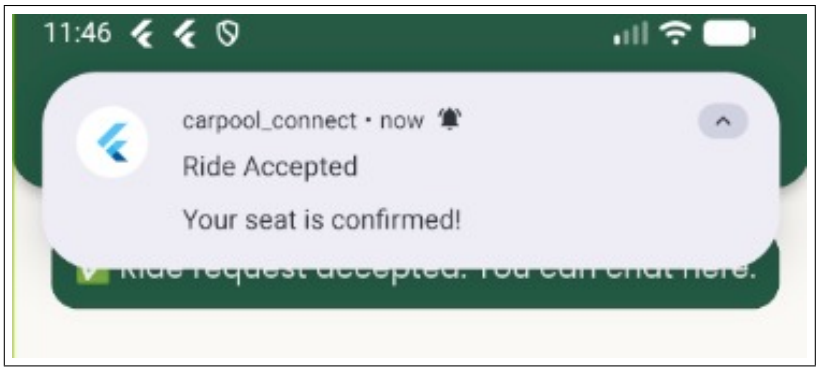


Figure A.8: Notifications

## A.4 Analytics

Carpoolers and passengers will be able to see the analytics about their ride history. These include details like total number of rides provided or taken, distances covered and summaries of payment.

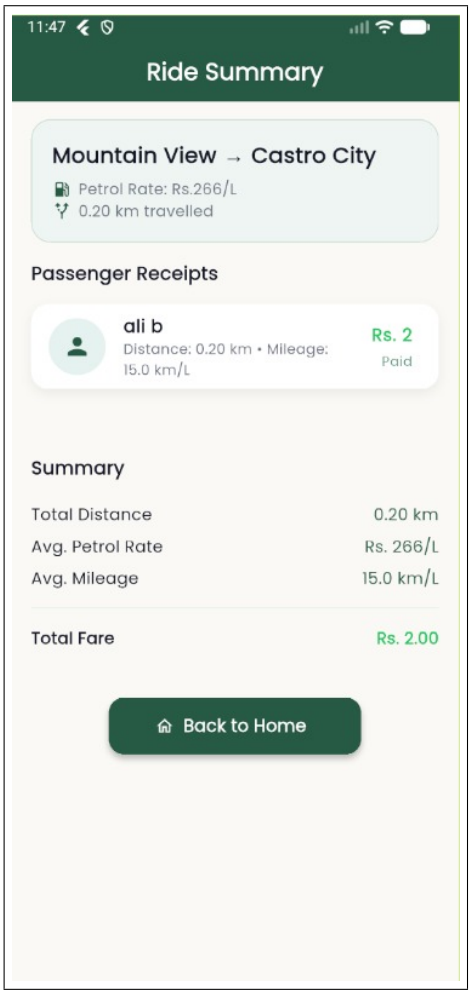


Figure A.9: Analytics Dashboard

## A.5 Admin Panel

The administration panel gives authorized administrators access to the activity of the system. Administrators are in a position to look at analytics, handle users, check ride records, and secure a smooth running of the platform.

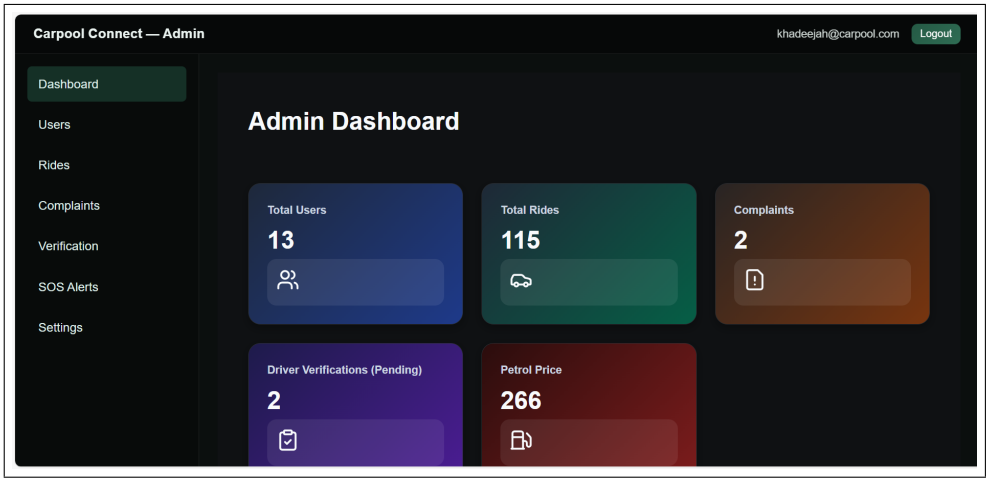


Figure A.10: Admin Panel Interface

## A.6 Tips and Troubleshooting

- Make sure the internet is reliable to track and give notifications in real time.
- Check user names and passwords in case of difficulties.
- Reread the application in case of insufficient loading of ride data.

## A.7 Privacy and Security

The data of the users is safely stored and processed in the system. The authentication procedures and access control are deployed to safeguard the user information and to guarantee the privacy of the data.

# References


[1] Estimating emissions reductions with carpooling and vehicle dispatching in ridesourcing mobility. *Nature Communications*, 2025. Accessed: Nov. 23, 2025. Cited on p. 5.

[2] Carpooling and the reduction of urban traffic. ResearchGate, 2025. Accessed: Nov. 23, 2025. Cited on p. 5.

[3] Safety of female ride-hailing passengers: Perception and prevention. ResearchGate, 2025. Accessed: Nov. 23, 2025. Cited on p. 5.

[4] Uberpool api – making carpooling possible for everyone. Uber Blog, 2025. Accessed: Nov. 22, 2025. Cited on pp. 6 and 7.

[5] J. Lo and S. Morseman. The perfect uberpool: A case study on trade-offs. Uber Engineering Blog, 2025. Accessed: Nov. 22, 2025. Cited on pp. 6 and 7.

[6] Poolyfi – carpool & ridesharing. Android Mobile Application, Google Play Store, 2025. Accessed: Nov. 22, 2025. Cited on pp. 6 and 7.

[7] Liftee – affordable carpooling & ride sharing in pakistan. Liftee Platform, 2025. Accessed: Nov. 22, 2025. Cited on pp. 6 and 7.

[8] Drivelu – let's pool the car! DriveLu Platform, 2025. Accessed: Nov. 22, 2025. Cited on pp. 6 and 7.

[9] Mapbox geocoding 101: Location intelligence essentials. Mapbox Blog, 2024. Accessed: Nov. 24, 2025. Cited on p. 7.

[10] Supabase for beginners: Resources. Supabase Documentation, 2024. Accessed: Nov. 24, 2025. Cited on p. 7.

[11] Learn flutter. Flutter Official Documentation, 2024. Accessed: Nov. 24, 2025. Cited on p. 7.

[12] Firebase cloud messaging documentation. Google Firebase Documentation, 2024. Accessed: Nov. 24, 2025. Cited on p. 7.

[13] Let's react: Articles and learning resources. LetsReact.org, 2024. Accessed: Nov. 24, 2025. Cited on p. 7.

[14] J. H. Fahim, M. Rashed, and M. I. Kakon. User perceptions of security in ride-sharing: An extended ase framework. *Preprint*, July 2025. Accessed: Nov. 22, 2025. Cited on p. 7.

[15] T. Çetin. *The Rise of Ride Sharing in Urban Transport: Threat or Opportunity*. InTech, Rijeka, 2017. Accessed: Nov. 22, 2025. Cited on p. 8.

[16] M. Stiglic, N. A. H. Agatz, M. W. P. Savelsbergh, and M. Gradisar. Enhancing urban mobility: Integrating ride-sharing and public transit. *Computers & Operations Research*, 90:12–21, 2018. Accessed: Nov. 22, 2025. Cited on p. 8.

[17] A. Ayaz, A. Waheed, H. Saleem, and M. M. Abid. Travelers' attitude towards carpooling in islamabad. *PLoS ONE*, 16(11):e0259312, 2021. Accessed: Nov. 22, 2025. Cited on p. 8.

[18] User safety in carpooling apps: How we ensure a secure ride every time. Appicial Blog, 2025. Accessed: Nov. 24, 2025. Cited on p. 8.

[19] Safety and trust factors in carpooling applications. *ScienceDirect*, 2025. Accessed: Nov. 24, 2025. Cited on p. 8.

[20] Handling payment and commission structure in a carpooling app. NectarBits Blog, 2025. Accessed: Nov. 24, 2025. Cited on p. 8.

[21] Artificial intelligence and decision support systems for ride-sharing platforms. *ScienceDirect*, 2023. Accessed: Nov. 24, 2025. Cited on p. 8.

[22] How carpool ratings work. BlaBlaCar Support, 2025. Accessed: Nov. 24, 2025. Cited on p. 9.

[23] Marketing strategies for promoting a carpooling app. Appicial Blog, 2024. Accessed: Dec. 2025. Cited on p. 11.

[24] Carpooling market size, share, growth and industry analysis. Business Research Insights, 2024. Accessed: Dec. 2025. Cited on p. 11.