# Wheel Share App

Izzah Malik
**01-135211-039**

Umair Khan
**01-135212-096**

Supervisor: Syed Junaid Hussain

**Bachelor of Science in Information Technology**

Department of Computer Science
Bahria University, Islamabad

December 2025

# Certificate

We acknowledge the work presented in the report titled **Wheel Share App**, authored by Ms. Izzah Malik and Mr. Umair Khan, as meeting the required standards for the partial fulfillment of the Bachelor of Science in Information Technology degree.

**Approved by:**

Supervisor: Syed Junaid Hussain

---

Internal Examiner: Name of the Internal Examiner (Title)

---

External Examiner: Name of the External Examiner (Title)

---

Project Coordinator: Dr. Kabeer Ahmed Bhatti

---

Head of Department: Dr. Muhammad Khurram Ehsan

---

**December 15th, 2025**

# Abstract

Wheel Share is a Flutter-based ride-sharing program that links drivers and riders going to comparable locations. It is powered by Supabase (PostgreSQL). Through shared trips, it encourages economical, convenient, and environmentally responsible transportation. The two main roles the application offers are Driver and User. While drivers may post rides, manage earnings, and accept offers, users can search for rides, issue offers, and monitor spending. Important components include an admin panel for monitoring rides, users, and alerts, ratings and reviews for trust, and an emergency alert system for safety. Wheel Share's cross-platform functionality, secure authentication, and real-time updates make it a dependable and scalable solution for smart urban mobility, reducing traffic and promoting sustainable transportation.

# Acknowledgement

We are incredibly appreciative of Allah Almighty, the Most Merciful and Beneficent, for all His bounties during our academic careers. We are deeply grateful to our supervisor, Syed Junaid Hussain, for his important advice, steadfast support, and helpful criticism, all of which were essential to this project's successful conclusion. We would like to express our sincere gratitude to our parents, whose unwavering support and prayers made this accomplishment possible, as well as to our family and friends, who have always been a source of inspiration. We also thank Bahria University's Department of Computer Science for providing a helpful learning environment. The support we have received from everyone is reflected in this initiative.

Izzah Malik, Umair Khan
Islamabad, Pakistan

*"Learning is the spark that ignites change in the world."*

– Malala Yousafzai

# Contents

**Certificate** i

**Abstract** ii

**Acknowledgement** iii

**Acronyms and Abbreviations** xi

**1 Introduction** 1

1.1 Background . . . . . 1
1.2 Objective . . . . . 2
1.3 Problem Description . . . . . 2
1.4 Project Objectives . . . . . 3
1.5 Project Scope . . . . . 3
1.6 Solution Application Areas . . . . . 4
1.7 Risk Involved . . . . . 5
1.7.1 Technical Challenges . . . . . 5
1.7.2 User Adoption . . . . . 5
1.7.3 Legal Compliance . . . . . 5

**2 Literature Review** 6

2.1 Ride-Sharing Systems . . . . . 6
2.1.1 User Factors and Barriers in Ride-Sharing Adoption . . . . . 6
2.1.2 Passenger Safety . . . . . 7
2.1.3 Trust Mechanisms in Ride-Sharing . . . . . 7
2.1.4 Technological Advancements in Ride-Sharing Systems . . . . . 7
2.1.5 Automated Ride Monitoring Safety Insights . . . . . 8
2.1.6 Experian Hunter . . . . . 8
2.1.7 Intelligent Ride Pattern Monitoring . . . . . 8
2.1.8 FICO Falcon Fraud Manager . . . . . 9
2.1.9 ClearSale . . . . . 9
2.1.10 Comparison Table . . . . . 9

**3 Requirement Specifications** **11**
3.1 Existing System . . . 11
3.2 Proposed Solution . . . 11
3.3 Functional Requirements . . . 12
3.4 Non-Functional Requirements . . . 13
3.4.1 Performance: . . . 13
3.4.2 Audit and Compliance: . . . 13
3.5 Use Cases . . . 13
3.5.1 Use Case Diagram of WheelShare System . . . 14
3.6 Use Case Specifications . . . 16
3.6.1 Use Case: User Login . . . 16
3.6.2 Use Case: Sign Up . . . 16
3.6.3 Use Case: Post Ride . . . 17
3.6.4 Use Case: Search Ride / Send Offer / Accept Offer . . . 17
3.6.5 Use Case: Ratings and Reviews . . . 18
3.6.6 Use Case: Spending Summary . . . 18
3.6.7 Use Case: Emergency Feature . . . 19
3.6.8 Use Case: Admin Monitoring . . . 19
3.7 Project Feasibility . . . 19

**4 Design** **21**
4.1 System Architecture . . . 21
4.2 Design Constraints . . . 22
4.3 Design Methodology . . . 23
4.4 High-Level Design . . . 24
4.4.1 User Login Authentication and Authorization Flow . . . 26
4.4.2 Ride Posting and Real-Time Notification Flow . . . 27
4.4.3 Offer Management and Ride Confirmation Flow . . . 28
4.4.4 Emergency Alert Handling and Notification Flow . . . 29
4.5 Low-Level Design . . . 29
4.5.1 Post Ride Submission and Validation Workflow . . . 30
4.5.2 Ride Offer Acceptance and Rejection Workflow . . . 31
4.6 GUI Design . . . 31
4.7 Mobile App Interface . . . 32
4.7.1 Initial App Launch and Location Permission Screen . . . 32
4.7.2 User Registration (Sign Up) Interface . . . 33
4.7.3 Admin Dashboard and System Monitoring Interface . . . 34
4.7.4 User Authentication (Login) Interface . . . 35
4.7.5 User Profile and Account Information Interface . . . 36

4.7.6 Ride Posting Interface . . . 37
4.7.7 Location Selection for Ride Posting . . . 38
4.7.8 Ride Details and Availability Interface . . . 39
4.7.9 Driver Ride Management Interface . . . 40
4.7.10 Passenger Ride Offers Dashboard . . . 41
4.7.11 History Dashboard . . . 42
4.8 External Interfaces . . . 43

**5 System Implementation 44**
5.1 System Architecture . . . 44
5.2 Tools and Technologies . . . 45
5.3 Development Environment . . . 45
5.4 Processing Logic / Algorithms . . . 45
5.4.1 Login and Role Routing . . . 46
5.4.2 Ride Posting and Offer Management . . . 46
5.4.3 Earnings and Spending Summary . . . 46
5.4.4 Emergency Alert Handling . . . 46
5.5 Integration with Supabase . . . 46
5.6 Testing and Deployment . . . 47
5.6.1 Testing . . . 47
5.6.2 Deployment . . . 47
5.7 Security Measures . . . 47

**6 System Testing and Evaluation 48**
6.1 Importance of System Testing . . . 48
6.2 Graphical User Interface (GUI) Testing . . . 48
6.3 Usability Testing . . . 49
6.4 Software Performance Testing . . . 49
6.5 Compatibility Testing . . . 49
6.6 Exception Handling . . . 49
6.7 Test Cases . . . 50
6.7.1 Test Case: Login . . . 50
6.7.2 Test Case: Ride Posting . . . 51
6.7.3 Test Case: Driver Ride Offer . . . 51
6.7.4 Test Case: Emergency Alert . . . 52
6.7.5 Test Case: Admin Dashboard . . . 52
6.7.6 Security Test Scenario . . . 53

**7 Conclusion 54**
7.1 Summary . . . 54

7.2 Future Work . . . . . 54

**References** **55**

# List of Figures


3.1 Use Case Diagram . . . . . 15

4.1 System Architecture Design . . . . . 22
4.2 Agile Methodology . . . . . 24
4.3 User Login Sequence Diagram Using Flutter and Supabase Authentication 26
4.4 Ride Posting Sequence Diagram . . . . . 27
4.5 Offer Management Sequence Diagram . . . . . 28
4.6 Emergency Alert Handling Sequence Diagram . . . . . 29
4.7 Post Ride Process Activity Diagram . . . . . 30
4.8 Accept Offer Process Activity Diagram . . . . . 31
4.9 Initial screen displayed when the application is installed . . . . . 32
4.10 Sign Up Page . . . . . 33
4.11 Admin Page . . . . . 34
4.12 Login Page . . . . . 35
4.13 User Profile . . . . . 36
4.14 User Post Ride . . . . . 37
4.15 Map-Based Location Selection Screen . . . . . 38
4.16 Ride Sharing Details . . . . . 39
4.17 Driver Offer a Ride . . . . . 40
4.18 Passenger Dashboard . . . . . 41
4.19 History Dashboard . . . . . 42

# List of Tables


2.1 Summary of Studies on Ride-Sharing . . . . . 10

3.1 Use Case for User Login . . . . . 16
3.2 Use Case for Sign Up . . . . . 16
3.3 Use Case for Post Ride . . . . . 17
3.4 Use Case for Ride Search and Offer Management . . . . . 17
3.5 Use Case for Ratings and Reviews . . . . . 18
3.6 Use Case for Spending Summary . . . . . 18
3.7 Use Case for Emergency Feature . . . . . 19
3.8 Use Case for Admin Monitoring . . . . . 19

6.1 Login Test Case . . . . . 50
6.2 Test Case for Ride Posting . . . . . 51
6.3 Driver Ride Offer and Acceptance . . . . . 51
6.4 Test Case for Emergency Alert Trigger . . . . . 52
6.5 Test Case for Admin Dashboard and Ride Statistics . . . . . 52
6.6 Negative and Security Test Results . . . . . 53

# Acronyms and Abbreviations

| | |
|---|---|
| **JWT** | JSON Web Token |
| **API** | Application Programming Interface |
| **GUI** | Graphical User Interface |
| **UI** | User Interface |
| **VS Code** | Visual Studio Code |
| **IDE** | Integrated Development Environment |
| **RSL** | Ride Sharing Layer |
| **Supabase SQL** | Supabase Structured Query Language |

# Chapter 1

# Introduction

The increasing interest in sustainable modes of transport and cost effective commuting has motivated the creation of grand ride-sharing concepts in the global context. Wheel Share is a community-based priority ride-sharing system which offers a secure, scalable, and easy experience to customers and the passengers who travel in the same direction. Will using Flutter as the frontend and Supabase (PostgreSQL) as the backend and Google API for location. The system promotes car sharing that saves gas and amounts of traffic growth, traffic congestion and encourage the sustainability of the environment. Ride-Sharing in a period when gas prices are increasing and where the issue of environmental awareness is on the rise. Ride-Sharing has turned to be a viable and greener way of transport. However, most existing ride sharing services are commercialized and have high commissions, which makes it community-based, affordable and centralized alternative. Through this platform, users can is used in order to post available rides. Passenger can also search for and send offers to join. In is also provided on the platform built in controls on financial overviews, rating schemes on critiques, and an emergency option which guaranties safety of the user. It uses cross platform flexibility of Flutter and Supabase real time backend services, Wheel Share makes data management and communication transparent as well as reliable and efficient performance. It does not only create trust and user-to-user collaboration, but it also is part of the overall objective of sustainable shared mobility [1].

## 1.1 Background

Transportation remains a daily challenge for millions of individuals, especially in urban areas where congestion, fuel prices, and time delays create significant inconvenience. The idea of carpooling or ride sharing emerged to address these issues by allowing people traveling in the same direction to share a single vehicle, thereby reducing costs and emissions. However, the local communities, in spite of the introduction of the ride sharing

services into the global community. We already do not have easy access to low-cost and safe prospects that can address their needs. These challenges have been such that Wheel Share has been created to handle them. The project integrates current mobile and web technologies to turn drivers and a flawless experience as well for passengers. The user can create profiles, add rides, search, since it has a user-friendly interface to match in journeys, make and receive offers. Both user type, two types of drivers, passenger and administrator is a role that occurs, which has both role and access privileges to assure ordered functionality and data security. Moreover, it has the emergency alert system, rating, review and capabilities.The earnings spending dashboards also help individuals save time and money besides offering assurance to the users. Wheel Share will,therefore, be a revolution a change of commuting culture to a more ecological and intermediate kind.

## 1.2 Objective

It will aim at providing a safe convenient along with reliable car-pooling environmentment by integrating the real time communication, open financial track and security features.The main aims will be the creation of a system that will provide the driver and the passengers with the opportunity to communicate and operate rides effectively and with the aim of developing trust, provide feedbacks in a smooth manner[4]. It also dedicated to the empowerment of the users and providing them with the entire control over their rides and trade, and at the same time enable administrators to trace activities and ensure that the platform possesses integrity.

Lastly, Wheel Share will target combination of convenience, cost efficiency and security, provide- Development of the alternative to the traditional commuting which will be a win-win to the users and the environment.

## 1.3 Problem Description

Numerous low-cost modes of transportation are being used as a result of the rising need for affordable, effective travel and transportation.Travel costs, expensive forms of transportation, and lack of reliable car-sharing services are all things that people dislike. The market offers a wide range of solutions.The data is heavy, commercial, or geographically restricted, making it impractical for personal or community use.User comments and market observations can help identify some of these. Through market observation and user feedback, several core issues were identified:

i. Lack of localized, affordable ride sharing solutions tailored to community use.

ii. Absence of a transparent system that allows users to view and compare rides based on city, address, and date.

iii. Inadequate financial tracking and transaction monitoring between drivers and passengers.

iv. Safety concerns due to insufficient identity verification and lack of emergency protocols.

v. Lack of comprehensive dashboards to analyze travel spending and earnings performance.

## 1.4 Project Objectives

The Wheel Share project has been designed with a clear set of objectives to ensure effective implementation and user satisfaction. The objective is to develop a cross-platform mobile application using Flutter that connects drivers and passengers in an efficient, secure, and transparent manner.

The system intends to implement role-based authentication through Supabase to ensure that each user type driver, passenger, and admin has specific access rights and responsibilities. Drivers will be able to post available rides with relevant details and manage offers from users. Passengers will be able to search for rides by city, address, and date, send offers, and manage accepted rides. Additionally, both parties will have access to dashboards that summarize earnings and expenses respectively, helping them maintain financial clarity.

Another key objective is the incorporation of a rating and review system that fosters accountability and builds trust among users. The inclusion of an emergency feature enhances safety by allowing users to alert the admin in case of unforeseen circumstances[3]. The admin panel provides tools to monitor user activity, manage rides, handle emergencies, and remove inappropriate users or content.

Together, these objectives ensure that Wheel Share becomes a holistic and user driven platform that encourages shared mobility while maintaining transparency, safety, and convenience.

## 1.5 Project Scope

The production of a ride sharing application is part of the Wheel Share combines safe authentication, tools of managing and communication cohesive system. The project will

be directed at providing a stable platform that contributes to provision posting of rides, searching and managing and ensuring that it is safe and transparent to all users. The system has three modules namely driver, passenger and administrator. The driver module provides their users with a chance to post rides, offers, and track it, view, and access their ride history. The passenger module allows the ride searching to take place and offers an offer dispatching, and cost tracking. The administration module provides management functionality, monitoring of processes on the platform and management of emergencies and the efficient work of the operation of the system. The user interface is Flutter and the Database and Authentication is Supabase services, Wheel Share will offer data synchrony, data reliability and data efficiency at real time. The concept of analysing it, in the end, all gives to the ride it has made a ride that is both community based and functional Sharing application that promotes price effectiveness, safety and sustainability. In conclusion, Wheel Share system is capable of realizing the recent technology and community partnership that will provide the new generation, dependable, and green ride-sharing solution. Helping to establish trustful relations and open transactions, and it enhances the experience of commuting to the entirety and generates the commuting experience and tend to smarter and better networked cities.

## 1.6 Solution Application Areas

The Wheel Share system is designed to serve a wide range of users who seek affordable and reliable commuting options. It provides an intelligent ride sharing platform suitable for daily commuters, students, employees, and individuals seeking intercity travel options. The system can be applied in both urban and suburban contexts, where it enables users to share rides, reduce travel expenses, and optimize vehicle usage.

The application also provides significant benefits for environmental sustainability, as it reduces traffic congestion and lowers carbon emissions through shared mobility. Its role-based dashboards empower different stakeholders drivers, passengers, and administrators by providing access to relevant tools and data insights[8]. For drivers, the platform acts as an earnings management tool, while passengers benefit from transparency in travel spending. Administrators can utilize the platform to monitor safety alerts, manage user activities, and ensure smooth operational flow.

## 1.7 Risk Involved

### 1.7.1 Technical Challenges

One significant technical challenges in creating Wheel Share is guaranteeing scalability and performance efficiency. The system needs to handle a high volume of ride data while ensuring real-time synchronization among numerous users. Possible risks include problems with integrating Supabase, performance lags in Flutter interfaces, and difficulties in handling multiple concurrent user sessions. These technical challenges, if not swiftly addressed, could affect system reliability and user satisfaction.Therefore, proactive testing, code refinement, and a strong backend design are essential to reduce technical risks.

### 1.7.2 User Adoption

Another risk is the user adoption. Ride-sharing in which the communication is a significant aspect is also of importance.The most important secret of success is participation that involves the amount of users. Without proper marketing and awareness, the platform will have low participation of drivers or passengers. Local promotions, strategic outreach programs to control this and referrals as incentives can also help to improve the uptake and continued usage of the Wheel Share application [9].

### 1.7.3 Legal Compliance

The project must also be able to meet the requirements of laws of transportation and data privacy at regional level.Confidence of the legislation of the check of users, protection of data and ride sharing policies should be introduced to avoid the potential legalization. Moreover, maintaining the field of transparency in user identity check and ride histories will help the platform to achieve control and earn customer trust.

# Chapter 2

# Literature Review

## 2.1 Ride-Sharing Systems

One of the most revolutionary developments in urban mobility is ride-sharing, which is changing the way people commute by allowing travelers on comparable routes to share a vehicle. Mitropoulos et al. (2021) claim that ride-sharing services act as a middleman between drivers and passengers, providing economical and ecologically friendly travel options while resolving issues with pollution and urban congestion[1].

Alonso-González et al. (2020) looked at the elements that influence users' propensity to share rides and discovered that social comfort, cost savings, journey time, and privacy all have a significant impact on participation decisions[2]. In a similar vein, Gangadharaiah et al. (2023) introduced the Pooled Rideshare Acceptance Model (PRAM), highlighting the importance of convenience, perceived safety, and trust for the long-term uptake of ridesharing services[3].

Even though ride-sharing services like Uber, Lyft, and BlaBlaCar have their benefits, they frequently work under centralised commercial frameworks that put making money first through commissions and spike pricing.Ashkrof et al. (2022) noted that driver acceptance and behavior toward ride offers are also influenced by dynamic pricing and operational flexibility[4].

### 2.1.1 User Factors and Barriers in Ride-Sharing Adoption

A major body of research highlights user psychology and behavioral factors as critical determinants in ride-sharing adoption. Merat et al. (2017) examined the human factors and user acceptance challenges in ride sharing, particularly for automated and digitalized vehicles, noting that perceived control and safety significantly influence user willingness

to participate [5].

Abouelela et al. (2022) found that the frequency of pooled versus solo ride-hailing use depends on demographic characteristics, trip purpose, and time flexibility[6]. Dastani et al. (2024) extended this by proposing a mathematical model that integrates transfer convenience, waiting time, and social comfort into ride-sharing preference predictions[7].

Participation in ride-sharing is also influenced by social and geographic differences. Users in low-income or less digitally connected neighbourhoods are less likely to use pooled ride-hailing, according to Soria and Stathopoulos (2021), underscoring the significance of equitable design in mobility platforms[8]. According to Compostella et al. (2025), future driverless and automated ride-sharing scenarios will increase consumer perceptions of safety, control, and comfort[9].

### 2.1.2 Passenger Safety

Adoption of ride-sharing is still heavily influenced by safety concerns. According to Chaudhry et al. (2018), ride-sharing systems should have emergency warning capabilities, user rating systems, and identity verification to guarantee accountability since passenger safety is a major factor in determining trust and retention[10]. Similarly, Acheampong et al. (2021) emphasized that digital mobility platforms should incorporate transparent user data handling and behavioral monitoring to mitigate risks related to harassment, fraud, and privacy breaches[11].

According to Mageto et al. (2025), users' propensity to utilise ride-hailing services is greatly reduced by perceived hazards, such as a lack of driver screening and ambiguity about route safety. The design of Wheel Share, which has an emergency feature, user authentication system, and rating and review systems to promote community trust and safety, is directly influenced by these insights [12].

### 2.1.3 Trust Mechanisms in Ride-Sharing

Hussain et al. (2025) explored commuters' preferences for future ride-sharing and concluded that a combination of safety assurance, technological reliability, and social responsibility drives long-term engagement[13].

### 2.1.4 Technological Advancements in Ride-Sharing Systems

In terms of scalability, accessibility, and real-time response, emerging technologies have completely changed how ride-sharing systems are implemented. According to Taiebat et al.

(2022), data-driven insights can optimize matching and pricing algorithms by analyzing user behavior and predicting ride-sharing preferences using machine learning models[14].

In order to promote justice while maintaining platform operations, Bujak and Kucharski (2024) emphasised the significance of striking a balance between profit and traveler acceptance in ride-pooling through customised fare schemes. This idea can be used to community-oriented systems like Wheel Share [15].

### 2.1.5 Automated Ride Monitoring Safety Insights

In 2023, R. Taylor et al. presented the Experian Hunter system, originally designed to analyze large datasets and identify unusual patterns. In a ride-sharing context, similar analytical techniques can help understand user behavior,improve platform reliability, and support better decision making while ensuring data privacy and security[16].

### 2.1.6 Experian Hunter

The Experian Hunter system, a data-driven analytical tool initially created to evaluate and analyse massive data sets for anomalous patterns, was introduced by R. Taylor et al. in 2023. Similar analytical techniques can be applied to a ride-sharing setting to examine ride patterns, user activity, and platform interactions in order to guarantee consistency, boost operational effectiveness, and facilitate trustworthy decision-making[16]. These tools assist ride-sharing services improve service quality and uphold operational and data protection regulations by offering immediate insights and predictive capabilities.

### 2.1.7 Intelligent Ride Pattern Monitoring

In 2025, J. Smith et al. discussed the Actimize system, which uses artificial intelligence and predictive analytics to identify unusual patterns and optimize operational decision making[17]. Similar AI-based analytical techniques can be adapted for ride-sharing platforms to improve user safety, ensure reliable matching, and strengthen overall platform trust. By analyzing behavioral patterns including unusual ride cancellations, duplicate ride postings, or repeated payment disputes predictive models can identify potential misuse or suspicious activity. Real-time alerts and automated risk scoring can then notify administrators for early intervention.

### 2.1.8 FICO Falcon Fraud Manager

n 2022 A.Lee et al enhanced that FICO Falcon Fraud Manager operates through machine learning to analyze insurance payments for detecting irregular patterns through its artificial intelligence capabilities. Insured party conduct data alongside existing claim statistics combined with payment documents allows FICO Falcon Fraud Manager to discover potential fraudulent activities. Insurers benefit from real-time fraud detection through the system which develops security systems and minimizes false claims by offering data-based choices to reduce insured losses[18].

### 2.1.9 ClearSale

In 2019 B. Johson et al presented that ClearSale combines artificial intelligence with human expertise to use its fraud prevention platform for insurance claims verification so it can stop fraudulent activities. Machine learning algorithms analysis within the platform examines claims data to discover system abnormalities that result in assessment report generation. The user-invoked review process improves both fraud detection quality and enables rapid and seamless handling of genuine claims[19].

### 2.1.10 Comparison Table

| Ref | Year | Author(s) | Approach | Contribution | Limitation |
|---|---|---|---|---|---|
| [1] | 2021 | Mitropoulos et al. | Study of ride-sharing's role in urban mobility | Highlights cost effectiveness and environmental benefits of shared commuting | Limited focus on safety and trust mechanisms |
| [2] | 2020 | Alonso-González et al. | Behavioral analysis of ride-sharing adoption | Identifies cost, time, and privacy as key adoption factors | Does not address fraud or security issues |
| [3] | 2023 | Gangadharaiah et al. | Pooled Rideshare Acceptance Model (PRAM) | Establishes safety, trust, and convenience as central to adoption | Focused mainly on user perception, not implementation |

*Continued on next page*

| Ref | Year | Author(s) | Approach | Contribution | Limitation |
|---|---|---|---|---|---|
| [10] | 2018 | Chaudhry et al. | Passenger safety in digital mobility | Proposes safety and emergency alert mechanisms for trust building | Does not consider real-time fraud or misuse detection |
| [11] | 2021 | Acheampong et al. | Data transparency and accountability in ride-sharing | Promotes secure data handling and user behavior monitoring | Focus limited to privacy rather than fraud prevention |
| [16] | 2023 | R. Taylor et al. | Experian Hunter automated Ride Monitoring | Safety insights analytics for real-time risk assessment | Initially designed for safety; needs adaptation for ride-sharing data |
| [17] | 2025 | J. Smith et al. | Intelligent ride pattern monitoring and predictive analytics | Detects anomalies through behavioral and transactional data patterns | Industry-specific model; lacks mobility-related parameters |
| [18] | 2022 | A. Lee et al. | FICO Falcon Fraud Manager using machine learning | Provides real-time analysis of payment irregularities | Limited to financial datasets; not optimized for ride-sharing context |
| [19] | 2019 | B. Johnson et al. | ClearSale AI and human review hybrid model | Enhances fraud detection accuracy through human-in-the-loop verification | Dependent on manual review, which may reduce scalability |
| [13] | 2025 | Hussain et al. | Trust mechanisms in ride-sharing systems | Identifies safety and social responsibility as long-term engagement drivers | Conceptual focus; lacks technical model implementation |

Table 2.1: Summary of Studies on Ride-Sharing

# Chapter 3

# Requirement Specifications

## 3.1 Existing System

Currently, majority of the ride-sharing coordination is done manually via phone calls, messaging app, or a social media platform. These old approaches are disjointed and inefficient and lead to communication failure, absence of transparency, and safety[10]. As an example, the riders and drivers find it hard to make schedules or confirm actions of the rides because there is no centralized system to handle the booking, confirming, and cancellations. Also, the users are not able to trace their past rides, sums spent, and revenues effectively. The other major drawback is that there is lack of emergency management leaving the riders unprotected in case of accidents or suspicious cases. Administrators also do not have real-time monitoring dashboard and tools and it is hard to monitor the work of the platform, identify improper behavior, or to be accountable. In this way, the existing model is not organized, not trustworthy, and cannot effectively meet the increasing need of the safe and community-focused ride-sharing models.

## 3.2 Proposed Solution

The new Wheel Share system will bring a monolithic, automated and convenient system that ensures that all ride-sharing functions are incorporated within one mobile and web application. Wheel Share uses Flutter because it is a cross-platform platform or stacks and Supabase because it has real-time background distribution assurance to provide its customers with smoother functionality, high safety, and engaging access of its applications. Riders, Drivers, and Admins of the system have role-based dashboards to make the system more user-friendly and secure. Users will be able to post rides, find the available ones by city, address or date, and deal with the driver. Drive through an in-built dashboard is

capable of sending offers, accepting rides and managing their overall earnings[11]. The system will also have building trust ratings and reviews under its system, emergency alert mechanisms to release the safety of the system and Administration Module to oversee the system, manage and monitor emergency activities. It also has a search and filtering tool of rides and transactions to enhance effectiveness. The suggested solution is projected to lessen the communication limitations, improve the levels of trust and transparency, and offer a safe and effective setting of shared mobility.

## 3.3 Functional Requirements

**User Authentication:**

The system allows individuals to create and log in using email and password credentials through Supabase Auth, and manage the sessions and change passwords and personal profiles.

**Ride Management:**

The drivers are offered with option of posting the rides with destination by address of the city and date. Riders can locate rides according to the various filters and invite to join any available rides. Obstacles that have been posted can be edited and erased when the need arises.

**Offer Management:**

The drivers who seek to take up rides may send offers and these offers may be rejected or accepted by the riders. This has contributed to this being flexible and in control by the two experiencing ride coordination.

**Ratings and Reviews:**

Following a ride, the users can do a review and comment upon each other. Such ratings help in maintaining reliability, improve quality of service and foster transparency.

**Emergency Feature:**

The system also has irritating alarm feature that allows people (riders or drivers) to call the admin instantly in case of emergency to have the required action taken in time and remain safe.

**Admin Module:**

The admin will be able to handle the total user activities, ride management and the rating issue, delete the unnecessary user account and also track the emergencies.

Adequate accountability may also be done by admins who can address reports and system logs.

## 3.4 Non-Functional Requirements

i. **User Interface:** The Wheel Share application provides an easy to use interface in Flutter version based on Web, Android and iOS applications giving users of any ability level ease in navigating the functionality.

ii. **Security:** The platform uses a high security future and Supabase Row-Level Security (RLS) to protect the credentials of the users, the rides, and the payment details.

### 3.4.1 Performance:

Basic functions like posting of rides, searching and offer response should be done within 2-3 seconds and this renders the users with quick and interactive interactions.

i. **Reliability:** The system will be built to guarantee 99.9

ii. **Scalability:** The system will be built to guarantee 99.9

iii. **Compatibility:** The app can support the recent models of the mobile operating system and web browsers which proves to be device-friendly.

### 3.4.2 Audit and Compliance:

All big operations such as ride posts, offers, approvals and emergency alerts are real-time logged to provide transparency, accountability and auditing.

## 3.5 Use Cases

A UML use case diagram are graphical illustrations that show how the users interact with what the system is able to do. When applied to Wheel Share, these diagrams show the interaction between Riders, Drivers, and the Admins and different modules of the platform. The Rider is able to modify and create rides, transfer and receive offers and rate the rides. The Driver has an opportunity to find rides, earns, and accept offers to work. Admin is responsible of the running of all system activities such as monitoring of users and tracking of the ride, as well as dealing with emergency alerts. The diagrams allow

developers to gain an overview of the scope of the system and simplify how the system functions were viewed by a user. They are also used as a basis in the definition of test cases, refinement of requirements and also in ensuring that the final implementation is in line with the expectations of the stakeholders.

### 3.5.1 Use Case Diagram of WheelShare System

This use case diagram illustrates the interaction between the main actors of the WheelShare system, including Rider, Driver, and Admin. It shows the core functionalities such as ride posting, ride searching, sending and accepting offers, and triggering emergency alerts. The diagram also represents administrative operations like user management, ride monitoring, and performance evaluation. Overall, it provides a high-level view of how different users interact with the system to support safe and efficient ride sharing.

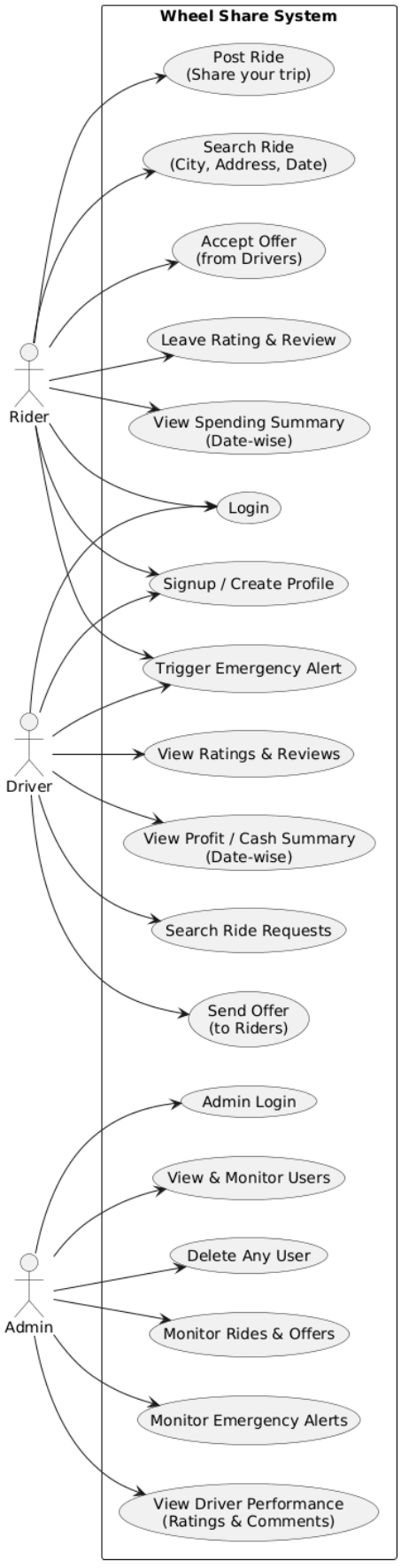


Figure 3.1: Use Case Diagram

## 3.6 Use Case Specifications

### 3.6.1 Use Case: User Login

| | |
|---|---|
| **Use Case ID** | UC001 |
| **Actor** | User, Driver, Admin |
| **Trigger** | User submits credentials. |
| **Pre-Condition** | Registered user account exists. |
| **Post-Condition** | User successfully authenticated. |
| **Flow** | 1. User opens the app.<br>2. User selects the login option.<br>3. User enters login details.<br>4. System verifies the account.<br>5. User completes the login process. |

Table 3.1: Use Case for User Login

**Description:** This use case describes the authentication process where users log in using valid credentials and are redirected to their respective dashboards.

### 3.6.2 Use Case: Sign Up

| | |
|---|---|
| **Use Case ID** | UC002 |
| **Actor** | Rider, Driver |
| **Trigger** | User opens the sign-up page. |
| **Pre-Condition** | Email is not already registered. |
| **Post-Condition** | Account is created and profile is saved. |
| **Flow** | 1. User enters personal details.<br>2. User verifies email.<br>3. User creates profile.<br>4. User logs into the system. |

Table 3.2: Use Case for Sign Up

**Description:** This use case explains the registration process for new users to create an account and access the system.

### 3.6.3 Use Case: Post Ride

| | |
|---|---|
| **Use Case ID** | UC003 |
| **Actor** | Rider |
| **Pre-Condition** | Rider is logged into the system. |
| **Post-Condition** | Ride is stored and visible to drivers. |
| **Flow** | 1. Rider enters ride details.<br>2. Rider submits the ride.<br>3. Ride becomes available for joining. |

Table 3.3: Use Case for Post Ride

**Description:** This use case allows riders to post ride details so that drivers can view and join the ride.

### 3.6.4 Use Case: Search Ride / Send Offer / Accept Offer

| | |
|---|---|
| **Use Case ID** | UC004 |
| **Actors** | Rider, Driver |
| **Pre-Condition** | User is logged in and rides are available. |
| **Post-Condition** | Ride is confirmed if offer is accepted. |
| **Flow** | 1. Driver searches available rides.<br>2. Driver sends an offer.<br>3. Rider accepts or rejects the offer. |

Table 3.4: Use Case for Ride Search and Offer Management

**Description:** This use case defines the interaction between riders and drivers for ride matching and confirmation.

### 3.6.5 Use Case: Ratings and Reviews

| | |
|---|---|
| **Use Case ID** | UC005 |
| **Actors** | Rider, Driver |
| **Trigger** | Completion of a ride. |
| **Pre-Condition** | Ride must be completed successfully. |
| **Post-Condition** | Feedback is saved and displayed. |
| **Flow** | 1. User selects completed ride.<br>2. User submits rating and review. |

Table 3.5: Use Case for Ratings and Reviews

**Description:** This use case allows users to rate each other and provide feedback after a completed ride.

### 3.6.6 Use Case: Spending Summary

| | |
|---|---|
| **Use Case ID** | UC006 |
| **Actors** | Rider, Driver |
| **Trigger** | User opens summary module. |
| **Pre-Condition** | Ride history exists. |
| **Post-Condition** | Summary is displayed. |
| **Flow** | 1. User selects date range.<br>2. System displays earnings or spending. |

Table 3.6: Use Case for Spending Summary

**Description:** This use case enables users to track their earnings or spending within a selected time period.

### 3.6.7 Use Case: Emergency Feature

| | |
|---|---|
| **Use Case ID** | UC007 |
| **Actors** | Rider, Driver, Admin |
| **Trigger** | User presses emergency button. |
| **Pre-Condition** | Ride is ongoing. |
| **Post-Condition** | Emergency alert is logged. |
| **Flow** | 1. User triggers alert.<br>2. System notifies admin. |

Table 3.7: Use Case for Emergency Feature

**Description:** This use case ensures user safety by allowing emergency alerts during an active ride.

### 3.6.8 Use Case: Admin Monitoring

| | |
|---|---|
| **Use Case ID** | UC008 |
| **Actors** | Admin |
| **Trigger** | Admin logs into the system. |
| **Pre-Condition** | Admin authentication successful. |
| **Post-Condition** | System activities are monitored. |
| **Flow** | 1. Admin monitors users and rides.<br>2. Admin manages emergency alerts. |

Table 3.8: Use Case for Admin Monitoring

**Description:** This use case allows the administrator to monitor system activities and manage users.

## 3.7 Project Feasibility

The Wheel Share solution is most likely to be realized since it is based on open source technologies that prove to be inexpensive and have a scalable architecture. Flutter and

Supabase can be used to dramatically decrease development cost and provide high-performing and cross-platform compatibility. The modularity of the application helps to make the future development of the app easier and manageable in terms of maintenance. Its friendly interface, which is based on the secure backend management, makes it applicable in the real world application. Besides, it covers the issues of affordability, convenience and safety thus finding its place in the area of community ride-sharing which makes the system sustainable in terms of urban mobility.

# Chapter 4

# Design

The architecture of the Wheel Share system is intended at developing scalable, secure, user-friendly ride sharing service that may be used by multiple users Rider, Driver, and Admin to serve in each of the mobile and web environments. The chapter gives a summary of the architectural structure, design limitations, and approach to be used in order to provide an unhindered user experience, performance of the system, and flexibility to experience future expansion. Since Wheel Share incorporates real-time data processing and cloud-based systems, it was stressed that proper planning and modular design were necessary to ensure that the system was efficient, had a high level of maintainability, and reliability of all the components[1].

## 4.1 System Architecture

Wheel Share system is based on a service-oriented architecture that is modular to guarantee flexibility and scalability. It uses Flutter to develop cross-platform front-end forms, which could serve both mobile and web consumers. The client connects to Supabase services Authentication, PostgreSQL Database, and Storage, and Realtime channels via secure HTTP connections via JWT-modified sessions. The PostgreSQL uses Row-Level Security (RLS) policies to provide Role-Based Access Control (RBAC) to guarantee that users can only retrieve data that they are allowed to access by their role.

A database has information that is important (user profiles, ride details, offers, emergency alerts, and so on) whereas Supabase storage has media files such as ride pictures or payment receipts. The use of this architecture provides secure communication, reliable data storage and real time synchronization of rides, offers and emergency notifications throughout the platform[14].

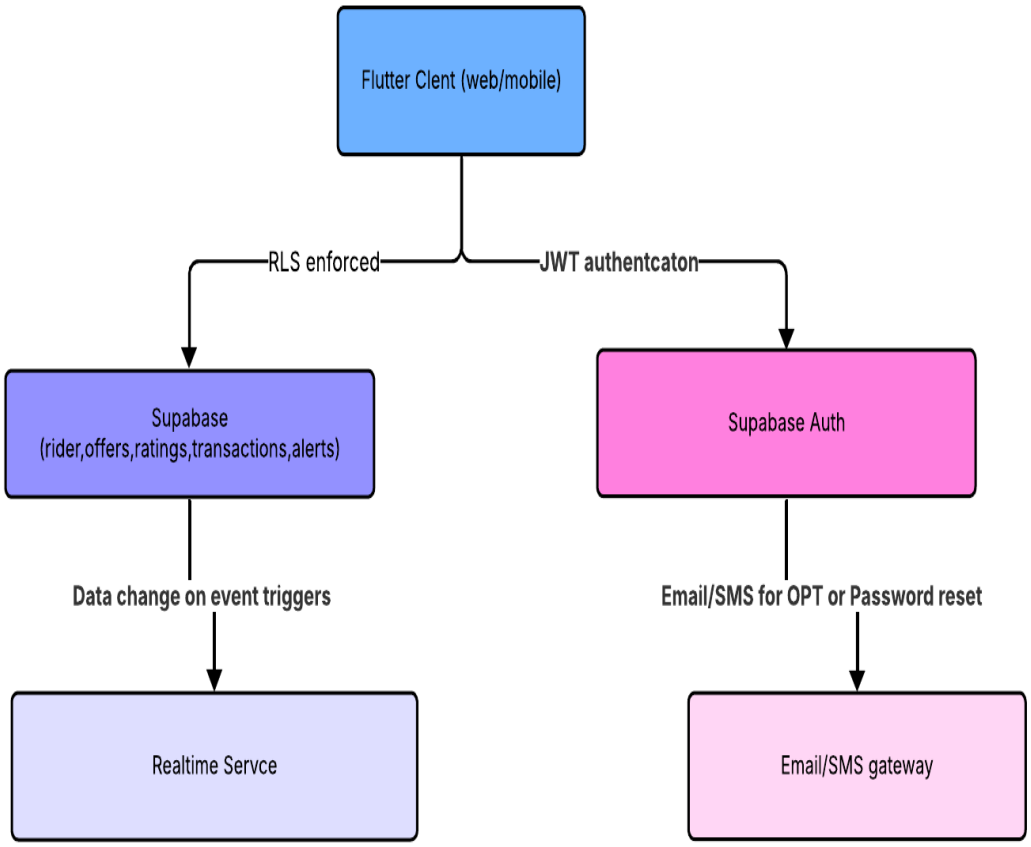


Figure 4.1: System Architecture Design

**Description:** This architecture illustrates the interaction between the Flutter client, Supabase backend, and authentication services. The Flutter web/mobile client communicates with Supabase using JWT-based authentication, where Row Level Security (RLS) ensures controlled data access. Supabase Auth manages user authentication and handles email based OTP and password reset through an external gateway. Any data changes in the Supabase database trigger real-time services, enabling instant updates for rides, offers, ratings, transactions, and emergency alerts within the system.

## 4.2 Design Constraints

When designing Wheel Share, there were a number of technical, environmental, and operational factors that were used to design the system in a manner that satisfies its operation in the real-life state.

- **Software Requirements:** The browser Wheel Share should be compatible with latest software like Google Chrome or Microsoft Edge or Android mobile platforms.

The communication with Supabase services through HTTPS requires a stable internet connection. Moreover, authentication and password recovery messages will also need an organization email account.

- **Development Environment Requirements:** The programming languages needed in the front-end development are Flutter SDK (greater than 3.x) and Dart programming language. IDEs: the IDEs may be VS Code or Android Studio. It can be run with the aid of the backend using Supabase, and formatted with the help of SQL migration tools with the help of web-based Supabase Dashboard

- **Frameworks and Programming Language:** The front-end Flutter Dart of the system is reactive and has the successful state management. It also has a Supabase-based backend which contains PostgreSQL and structured data storage, Authentication and secure user management, storage and media management, and real-time updates. In addition, emergency alerts and receive notifications both will be the secure workflows that will be implemented on serverless Edge Functions.

- **Technical Limitations:** Storage space and network latency of mobile browsers are also restricted by limitations of system performance, therefore, can have an impact on real-time ride updates and notifications. Besides, real time capabilities scalability depends on Supabase project which are the number of concurrent WebSocket connections and the number of concurrent requests.

- **Assumptions:** I believe the user familiar with the cardinal ride-sharing is involved in the activities of posting rides, making and accepting offers and rating rides. The fact that the drivers are supposed to be aware of the availability of the rides also questions them assuming that the emergency alerts will be monitored by the administrators at any given time.

## 4.3 Design Methodology

The Wheel Share development and design process is modular and structure based which is premised on an **Agile development methodology**. The system is divided into logical modules that are easy to understand and they can be maintained at the same time as they can be developed.

- **Partitioning:** The system is separated by features into Rider, Driver, and Admin modules and has some common services like Authentication, Ride Management, and Notifications. The database is partitioned to use separate tables that contain the data of the Users, Rides, Offers, Ratings, Transactions, and Emergency Alerts. Time

based partitioning is used to manage asynchronous jobs such as the summary of earnings or expenditure and the delivery of timely notifications.

- **Communication Requirements:** Communication between the client and Supabase is fully complete via the HTTPS protocol using tokens through in the authorization header. Instant synchronization, e.g. new offers, ride confirmations and emergency alerts are handled through WebSocket channels.
- **Agglomeration Strategy:** Shared ride-list, offer management and rating display UI widgets are used to achieve design efficiency and re-usability of code. Database views are developed to make complex queries easier.
- **Mapping:** Flutter has route guards that are mapped to role based permissions to ensure that only Riders, Drivers and admin can gain access of an authorized section. Database entities like Rides and Offers are implemented with RLS policies, and Edge Functions are implemented to automatically notify the administrator and emergency contacts in case an alert is fired.

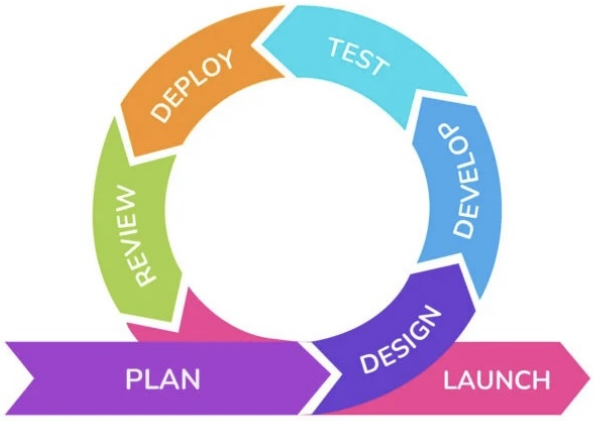


Figure 4.2: Agile Methodology

## 4.4 High-Level Design

The high level design of Wheel Share names the key elements of the system and their interaction process to attain the desired functionality. It focuses primarily on the database design and entity relationships, which guarantee efficient storage, retrieval and connectivity of the information of the user and the ride.

- **User:** Keeps information about customers like user-Id, name, email, role-Rider/Driver and profile description.

- **Ride:** Ride data are stored in the stores and include ride related data such as source, destination, city, address, date and rider identification.
- **Provide:** Relates rides/drivers and offers four fields of offer-Id, ride-Id, driver-Id and offer-Status.
- **Rating:** : It is used in rating the rides and is contained of rating-Id, ride-Id, reviewer-Id, score, and comments.
- **Transaction:** : This table records all the transactions by transaction-Id, ride-Id, user-Id, amount and type (earning/ spending).
- **Emergency Alert:** : Responsible of handling alerts like alert-Id,ride-Id,timestamp and status.

### 4.4.1 User Login Authentication and Authorization Flow

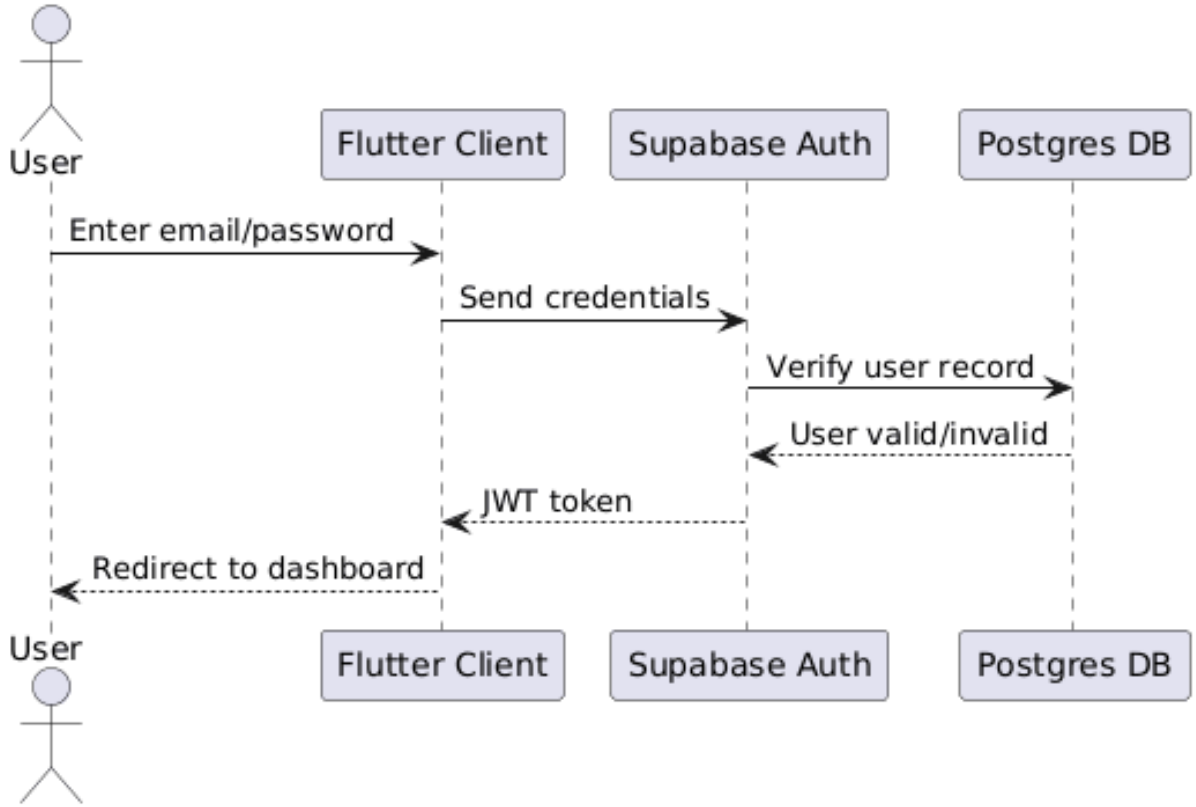


Figure 4.3: User Login Sequence Diagram Using Flutter and Supabase Authentication

**Description:** This sequence diagram shows the user login process in the system. The user enters credentials through the Flutter client, which are sent to Supabase Authentication for verification. Supabase Auth validates the user using the PostgreSQL database. After successful authentication, a JWT token is generated and the user is redirected to the dashboard.

### 4.4.2 Ride Posting and Real-Time Notification Flow

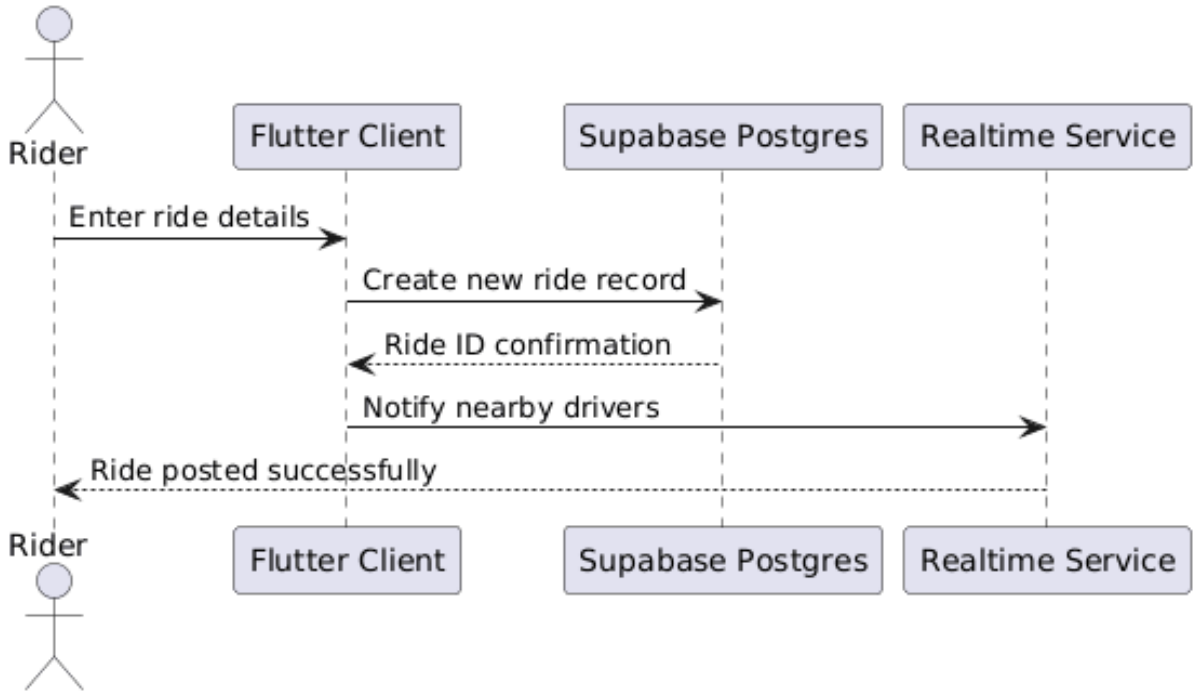


Figure 4.4: Ride Posting Sequence Diagram

**Description:** This sequence diagram illustrates the ride posting process performed by a rider. The rider enters ride details through the Flutter client, which creates a new ride record in the Supabase PostgreSQL database. Once the ride is successfully stored, a ride ID is generated. The real-time service then notifies nearby drivers about the newly posted ride.

### 4.4.3 Offer Management and Ride Confirmation Flow

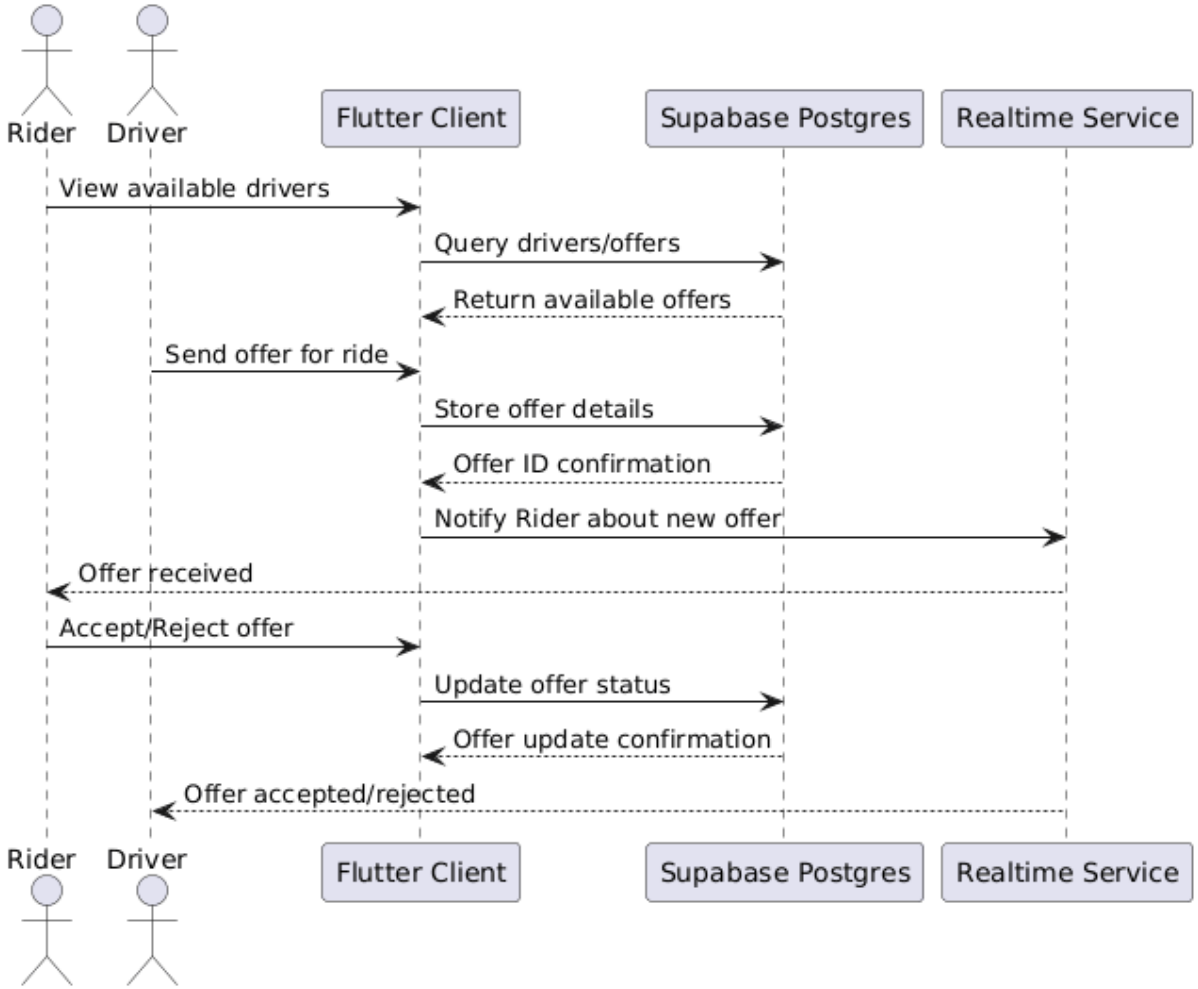


Figure 4.5: Offer Management Sequence Diagram

**Description:** This sequence diagram shows the offer management process between riders and drivers. Drivers view available rides and send offers through the Flutter client, which are stored in the Supabase PostgreSQL database. The real-time service notifies riders about new offers. Riders can accept or reject offers, and the system updates the offer status accordingly.

### 4.4.4 Emergency Alert Handling and Notification Flow

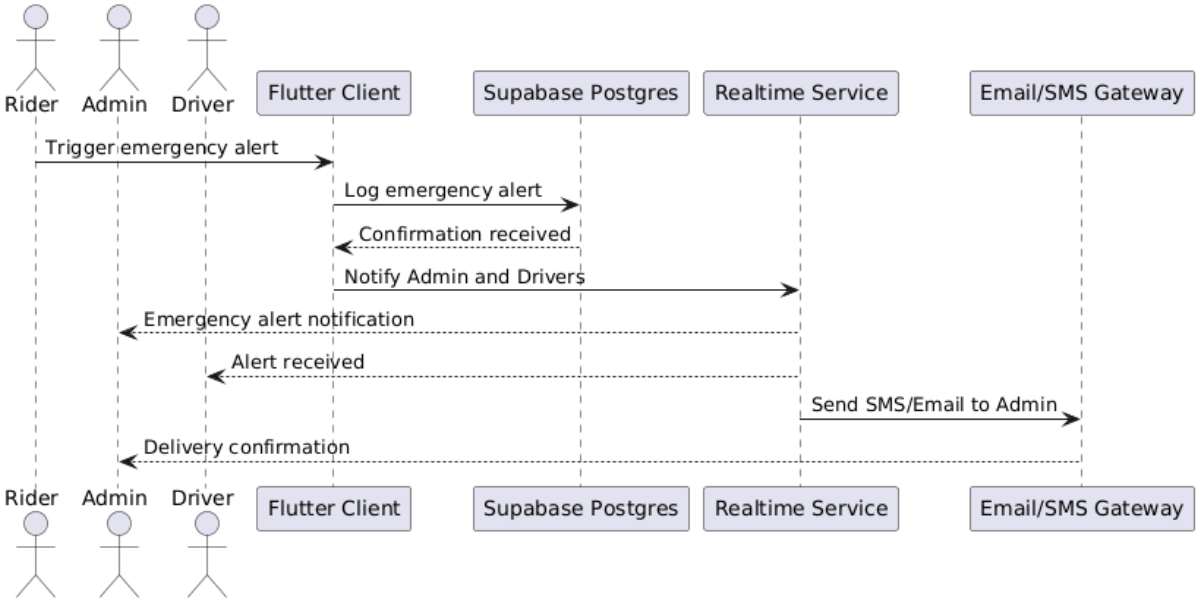


Figure 4.6: Emergency Alert Handling Sequence Diagram

**Description:** This sequence diagram illustrates how emergency alerts are handled within the system. When a rider triggers an emergency alert, the Flutter client logs the alert in the Supabase PostgreSQL database. The real-time service notifies the admin and drivers instantly, while the Email/SMS gateway sends notifications to ensure timely response and user safety.

## 4.5 Low-Level Design

Each system component's specific structure is specified in the WheelShare Low-Level Design. With distinct interfaces and internal logic, it divides the program into smaller parts, including ride posting, ride search, offer handling, emergency alarms, admin monitoring, and authentication. In addition to offering a clear implementation plan, this modular approach guarantees that the code is simple to create, maintain, and reuse.

### 4.5.1 Post Ride Submission and Validation Workflow

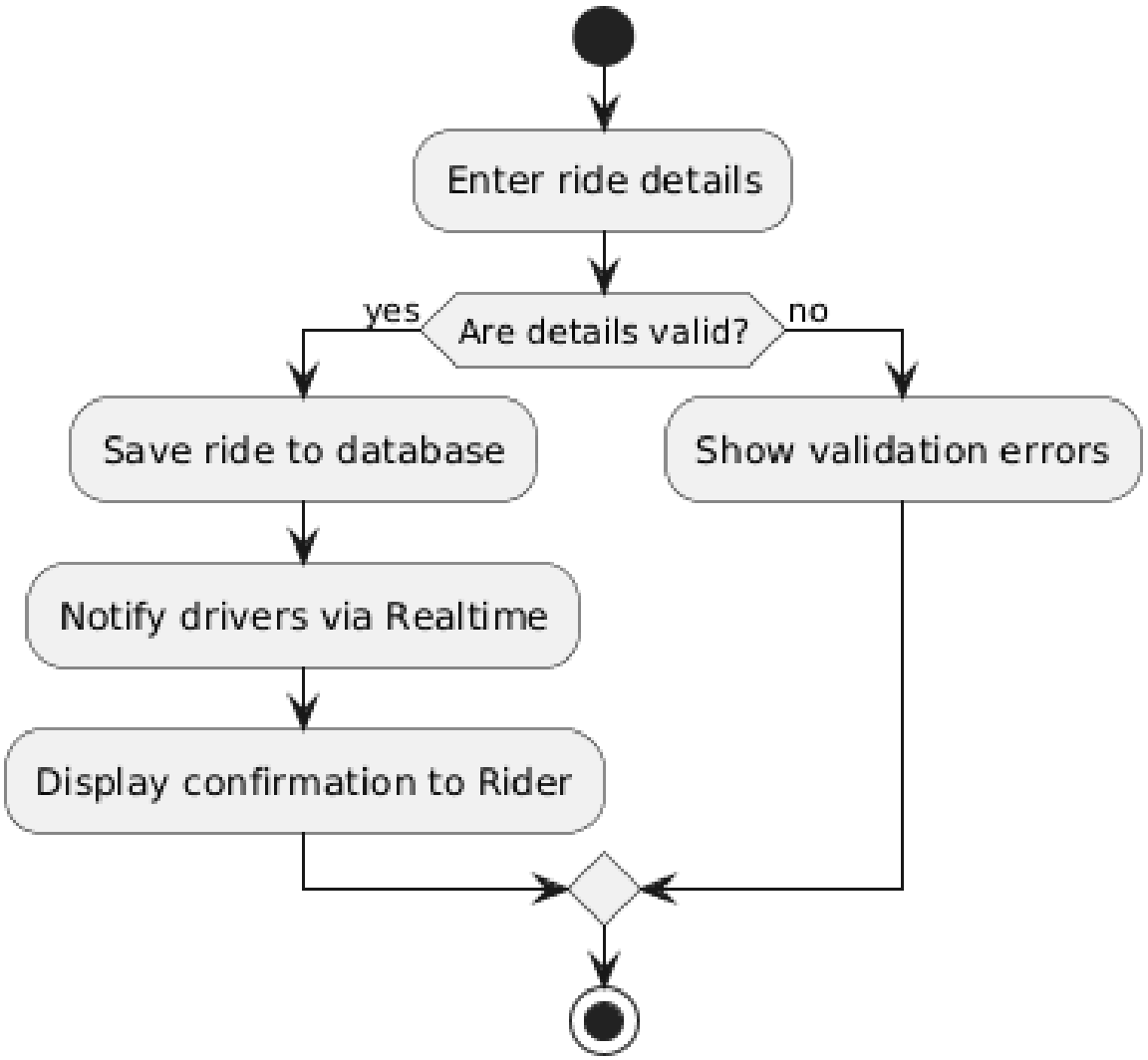


Figure 4.7: Post Ride Process Activity Diagram

**Description:** This activity diagram represents the workflow of posting a ride in the system. The rider enters ride details, which are first validated by the system. If the details are valid, the ride is saved to the database and nearby drivers are notified through the real-time service; otherwise, validation errors are displayed to the rider.

### 4.5.2 Ride Offer Acceptance and Rejection Workflow

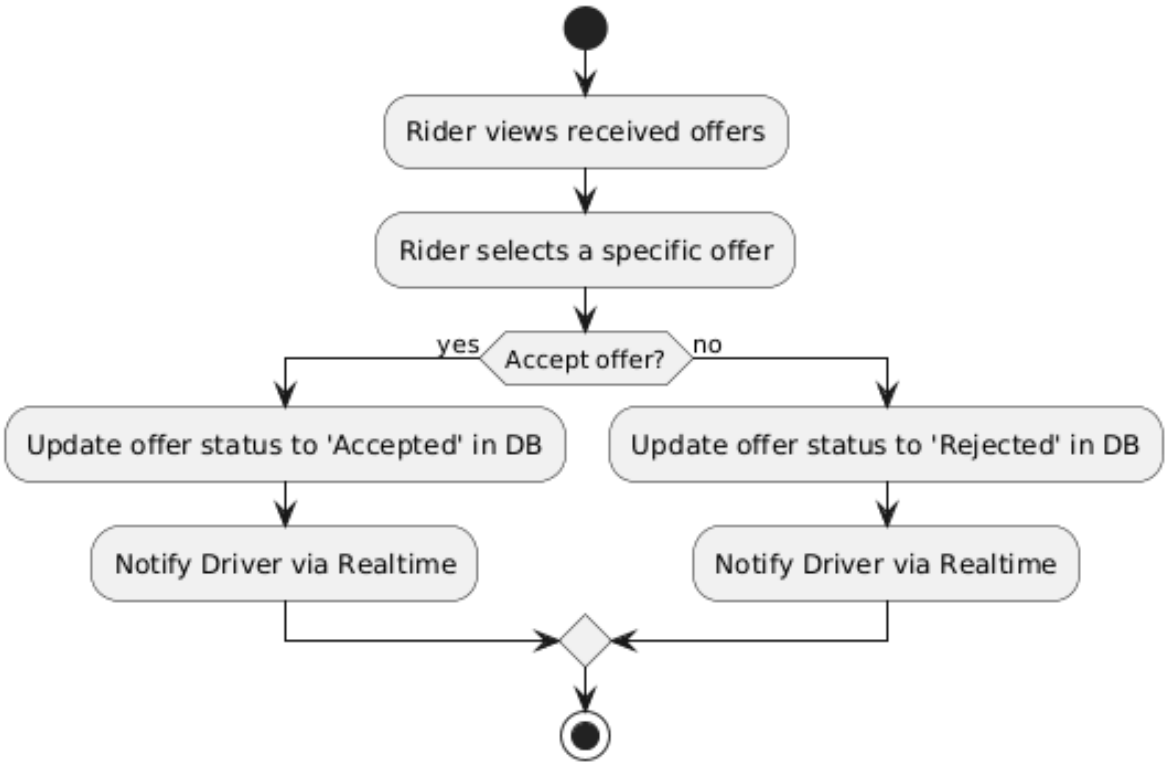


Figure 4.8: Accept Offer Process Activity Diagram

**Description:** This activity diagram illustrates the process of handling ride offers by the rider. The rider views received offers and selects a specific offer to accept or reject. Based on the decision, the system updates the offer status in the database and notifies the driver in real time.

## 4.6 GUI Design

Wheel Share is built with Flutter as its Graphical User Interface (GUI) to provide a similar and dynamic experience in the mobile and web interfaces. The interface is centered upon the clarity, access and ease of navigation. A bottom navigation bar or a layout based on tabs enables the users to have a quick reach to the basic functions such as posting rides, looking at offers, checking the earnings or spending, and causing emergency notifications. It is minimalistic and easy to use in design: there are no icons or labels, the color schemes are balance-oriented, and the input forms fit on a mobile screen. The system can also incorporate real-time notification of offers, confirmation and alerting to ensure continuous information to the users.

## 4.7 Mobile App Interface

The Flutter-based GUI was developed with a focus on:

### 4.7.1 Initial App Launch and Location Permission Screen

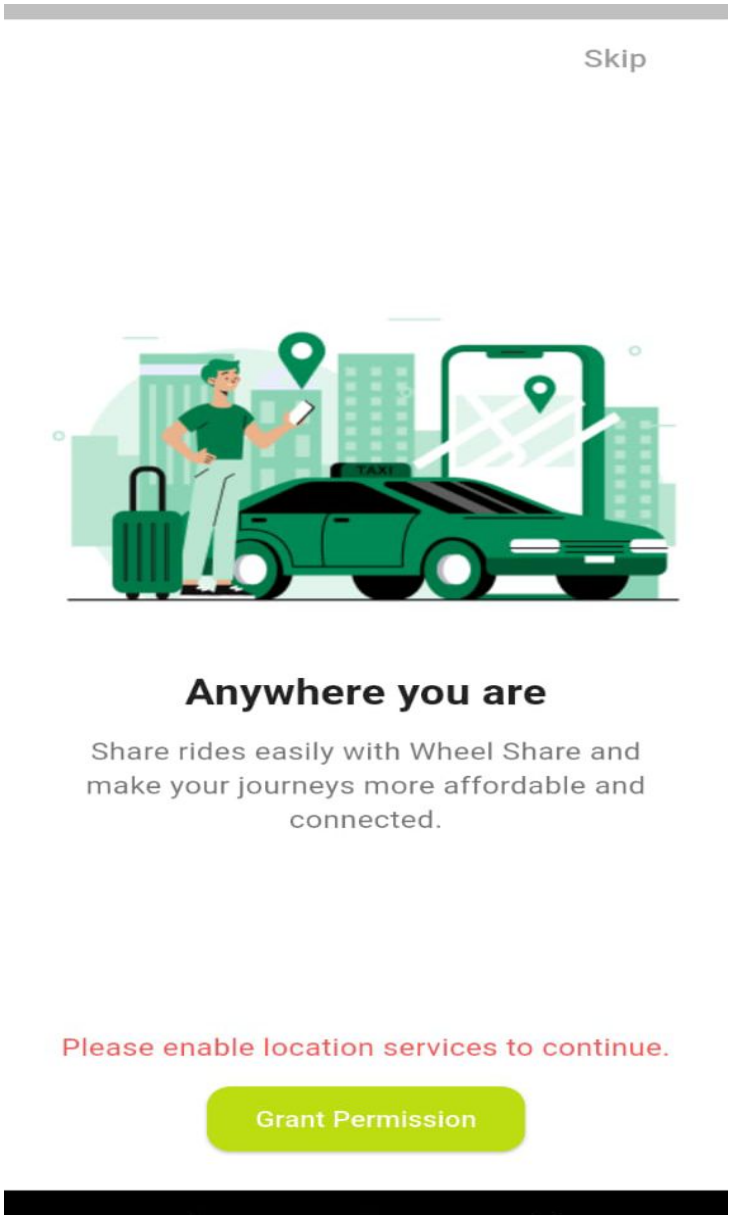


Figure 4.9: Initial screen displayed when the application is installed

**Description:** This screen is displayed when the application is launched for the first time after installation. It requests location permission to enable nearby ride discovery and navigation features.

### 4.7.2 User Registration (Sign Up) Interface

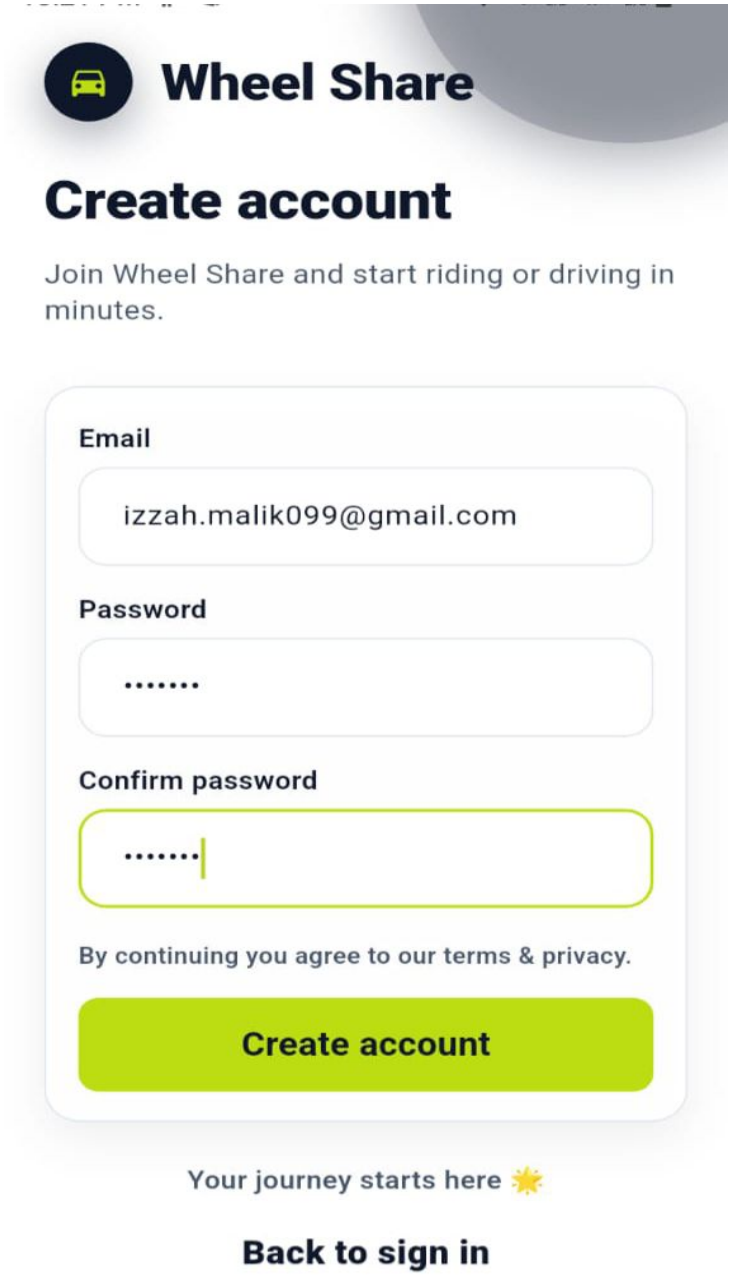


Figure 4.10: Sign Up Page

**Description:** This interface enables new users to register by providing their email and password information. Upon successful registration, users gain access to the core functionalities of the application.

### 4.7.3 Admin Dashboard and System Monitoring Interface


Figure 4.11: Admin Page

**Description:** This dashboard allows the admin to monitor system activity, including users, rides, and revenue statistics. It also provides analytical insights through graphs to support system management and decision-making.

### 4.7.4 User Authentication (Login) Interface

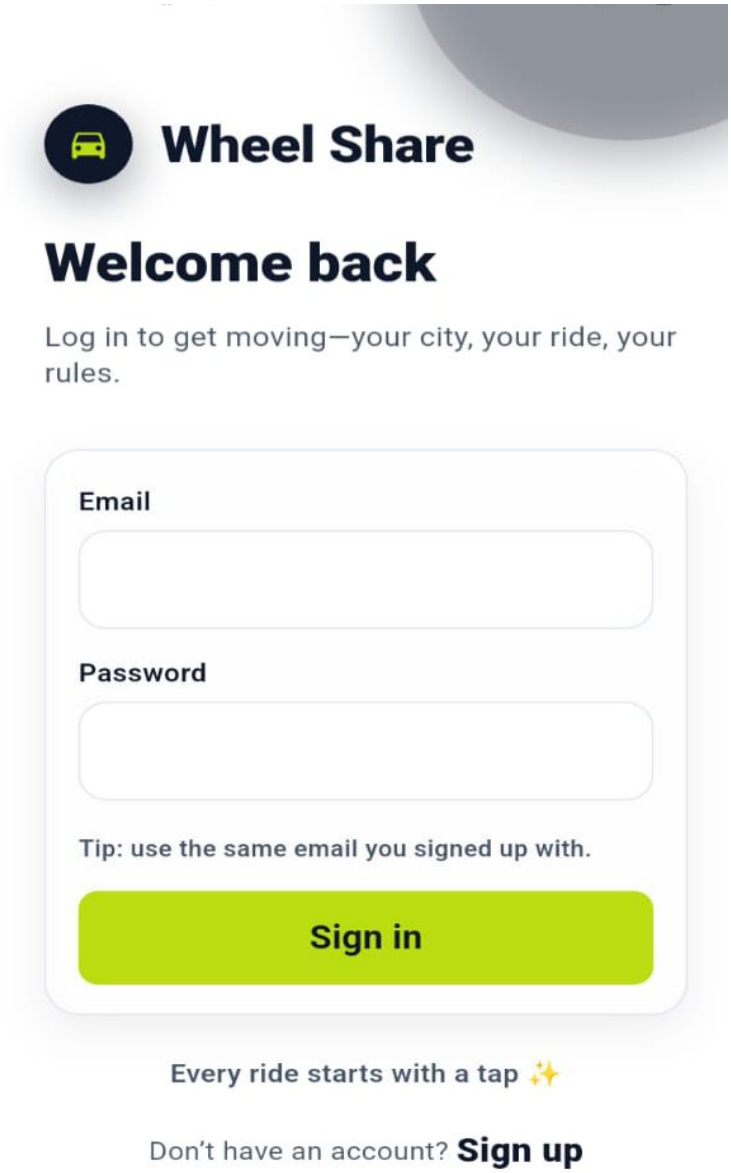


Figure 4.12: Login Page

**Description:** This screen allows registered users to log in by entering their email and password credentials. Successful authentication grants access to personalized dashboards and ride-sharing features.

### 4.7.5 User Profile and Account Information Interface

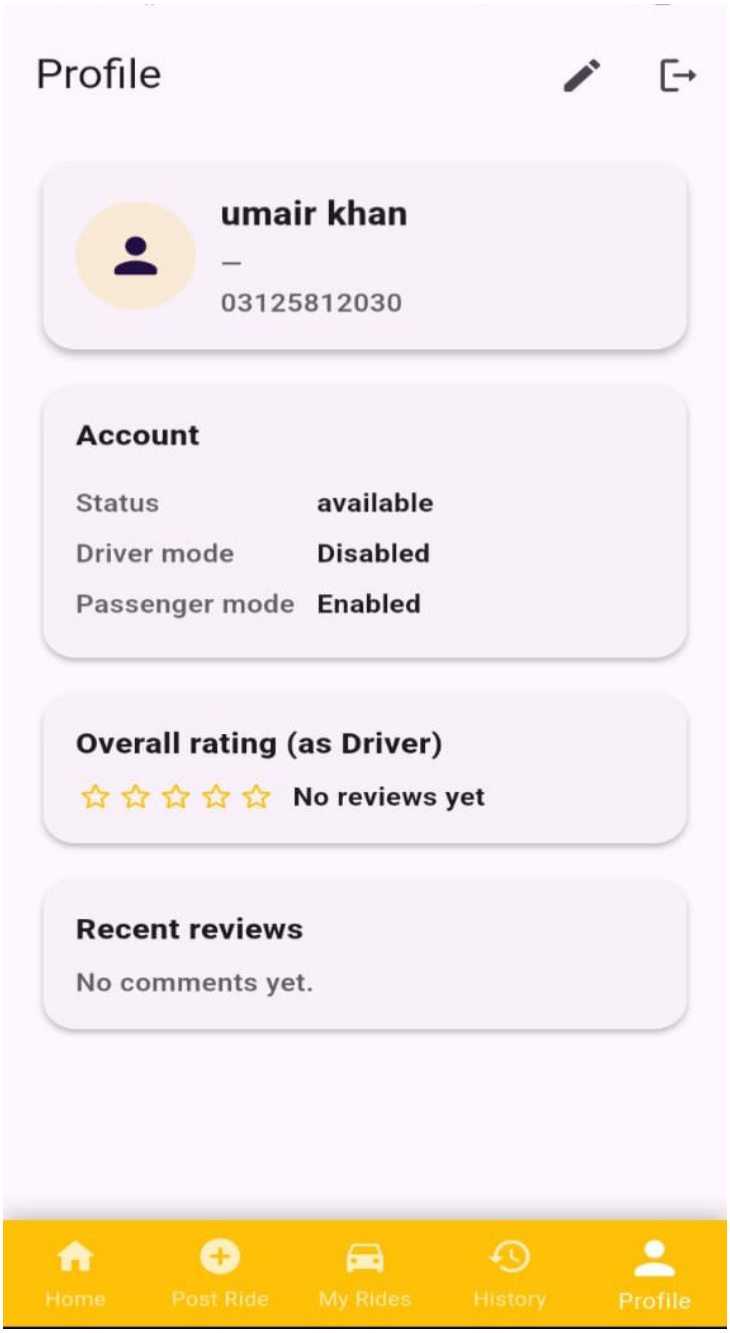


Figure 4.13: User Profile

**Description:** This screen displays the user's personal details, account status, and role preferences. It also shows driver ratings and recent reviews to support trust and transparency within the system.

### 4.7.6 Ride Posting Interface

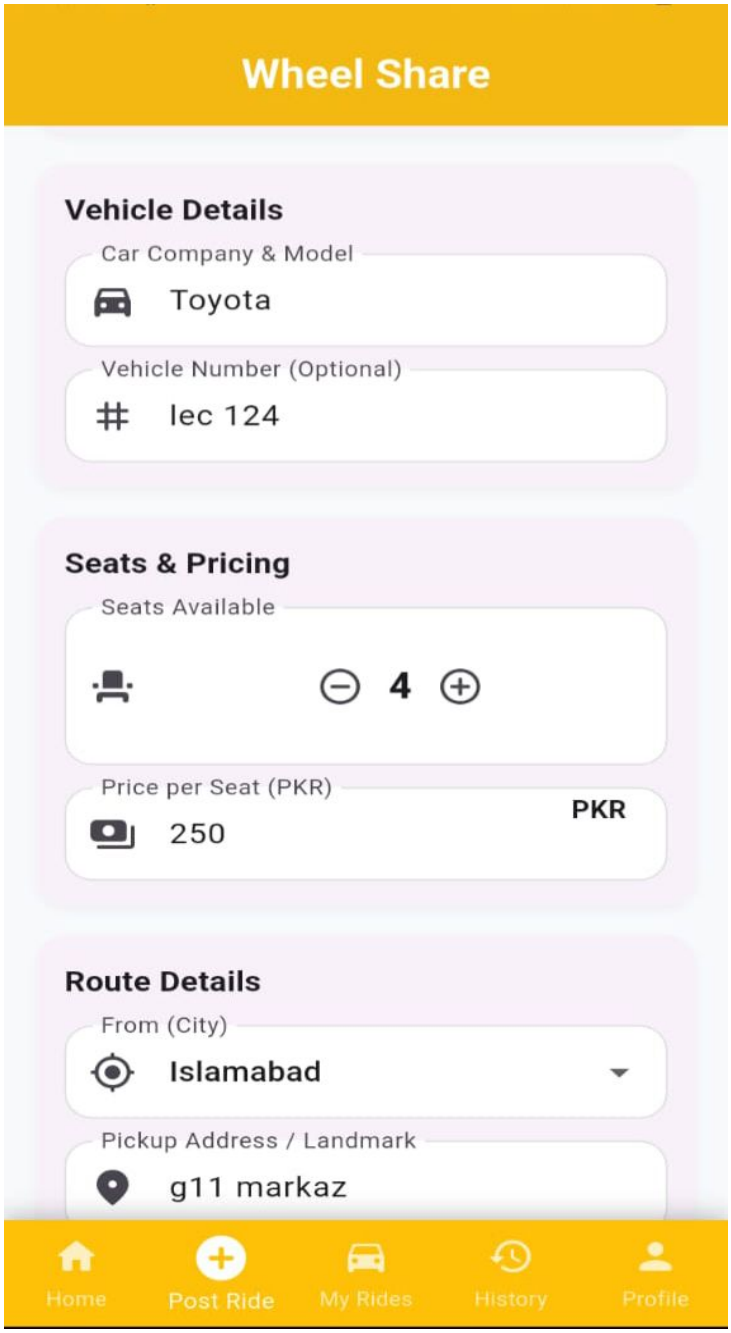


Figure 4.14: User Post Ride

**Description:** This interface allows users to post a ride by entering vehicle, pricing, and route details. Once submitted, the ride becomes available for drivers or passengers to view and join.

### 4.7.7 Location Selection for Ride Posting

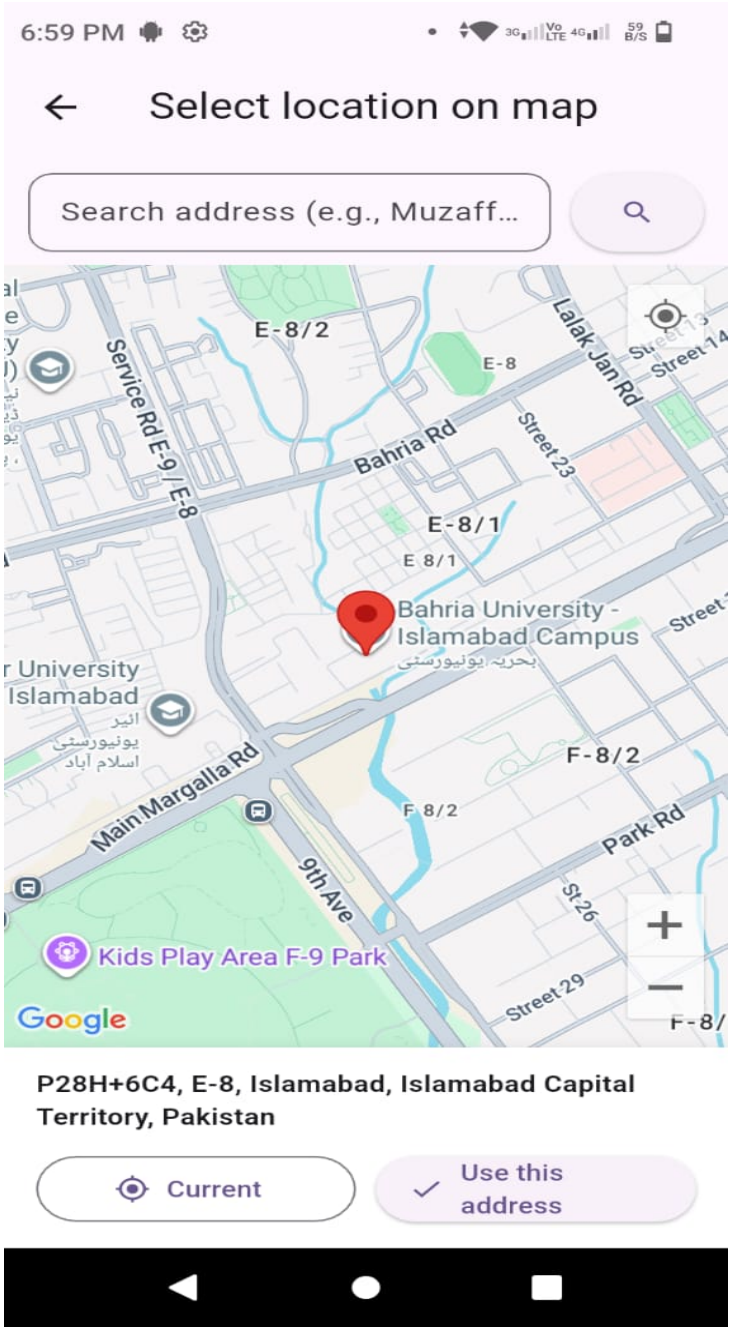


Figure 4.15: Map-Based Location Selection Screen

**Description:** This screen allows users to select the pickup or destination location directly from the map. The selected address is used to ensure accurate routing and ride matching within the system.

### 4.7.8 Ride Details and Availability Interface

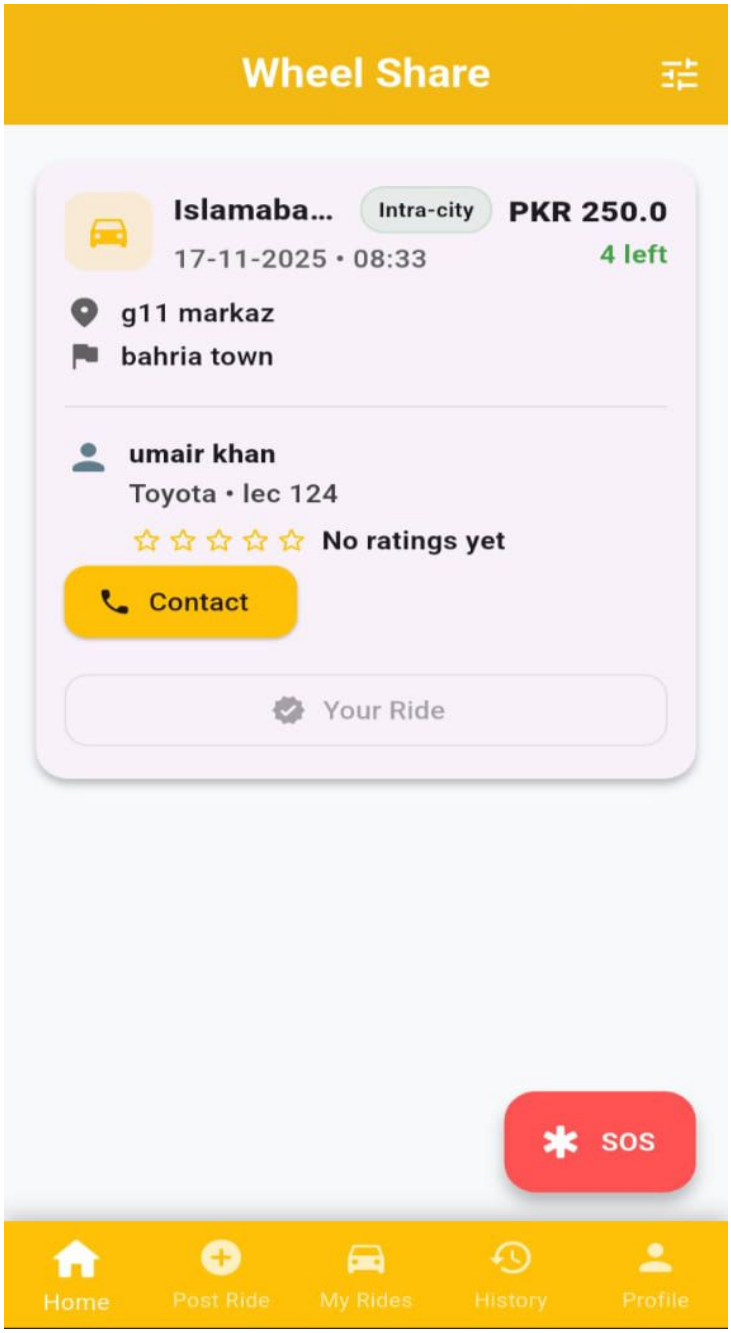


Figure 4.16: Ride Sharing Details

**Description:** This screen displays detailed information about an available ride, including route, time, price, and driver details. Users can contact the driver or view ride status directly from this interface.

### 4.7.9 Driver Ride Management Interface

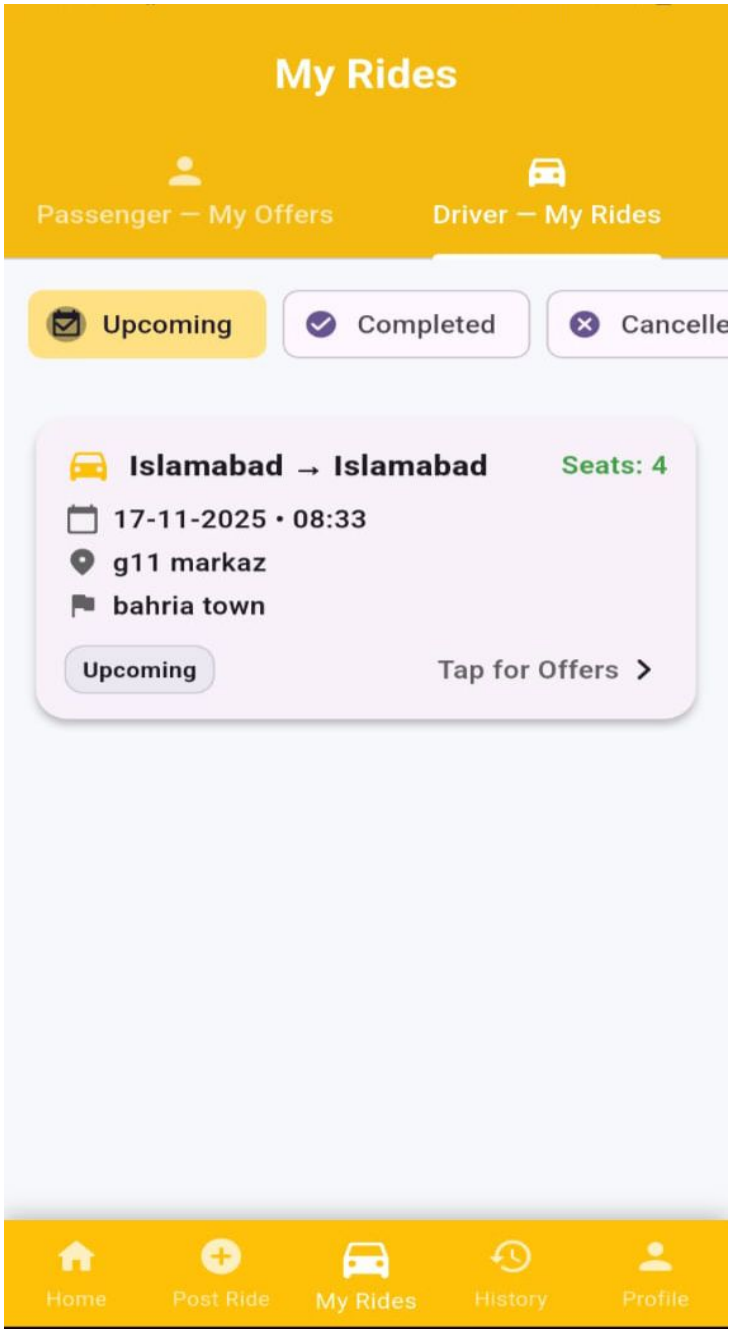


Figure 4.17: Driver Offer a Ride

**Description:** This screen allows drivers to view and manage their posted rides, including upcoming and completed trips. Drivers can also review ride details and track available seats from this interface.

### 4.7.10 Passenger Ride Offers Dashboard

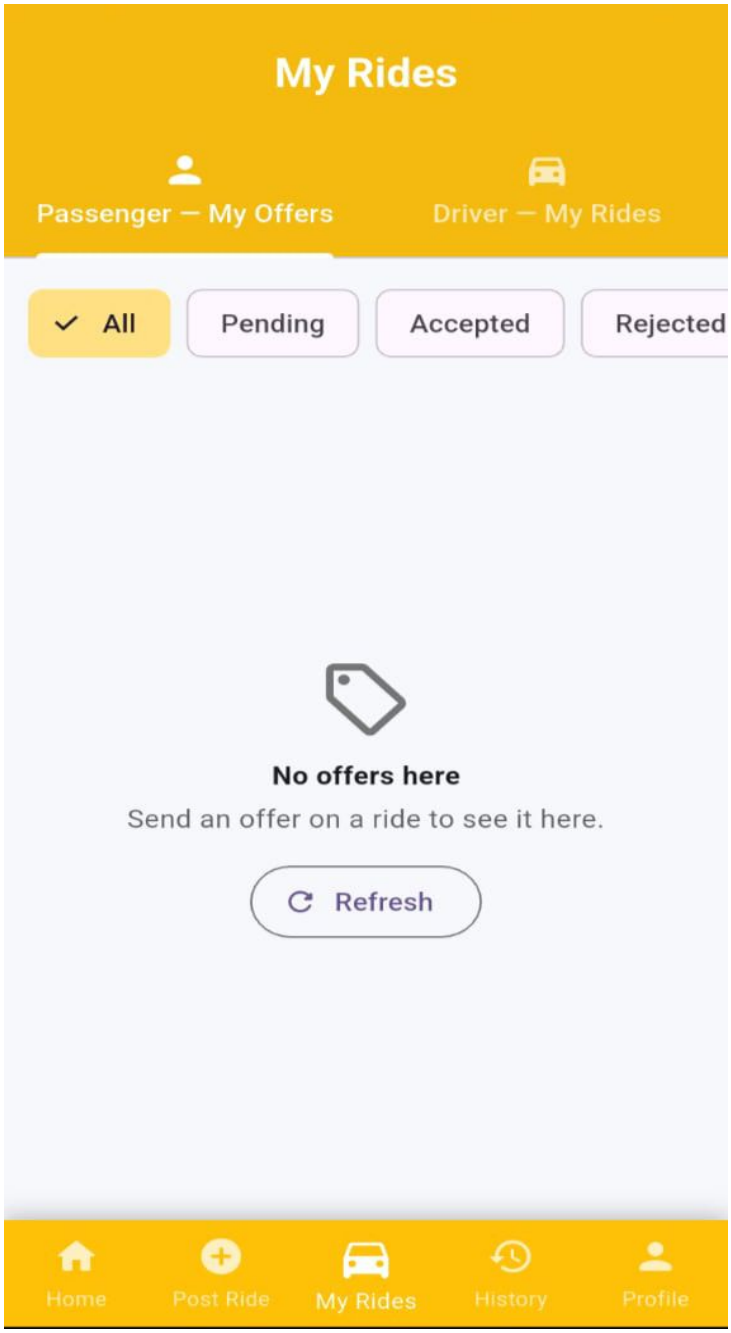


Figure 4.18: Passenger Dashboard

**Description:** This dashboard allows passengers to view and manage ride offers they have sent or received. Offers can be filtered by status such as pending, accepted, or rejected for easy tracking.

### 4.7.11 History Dashboard

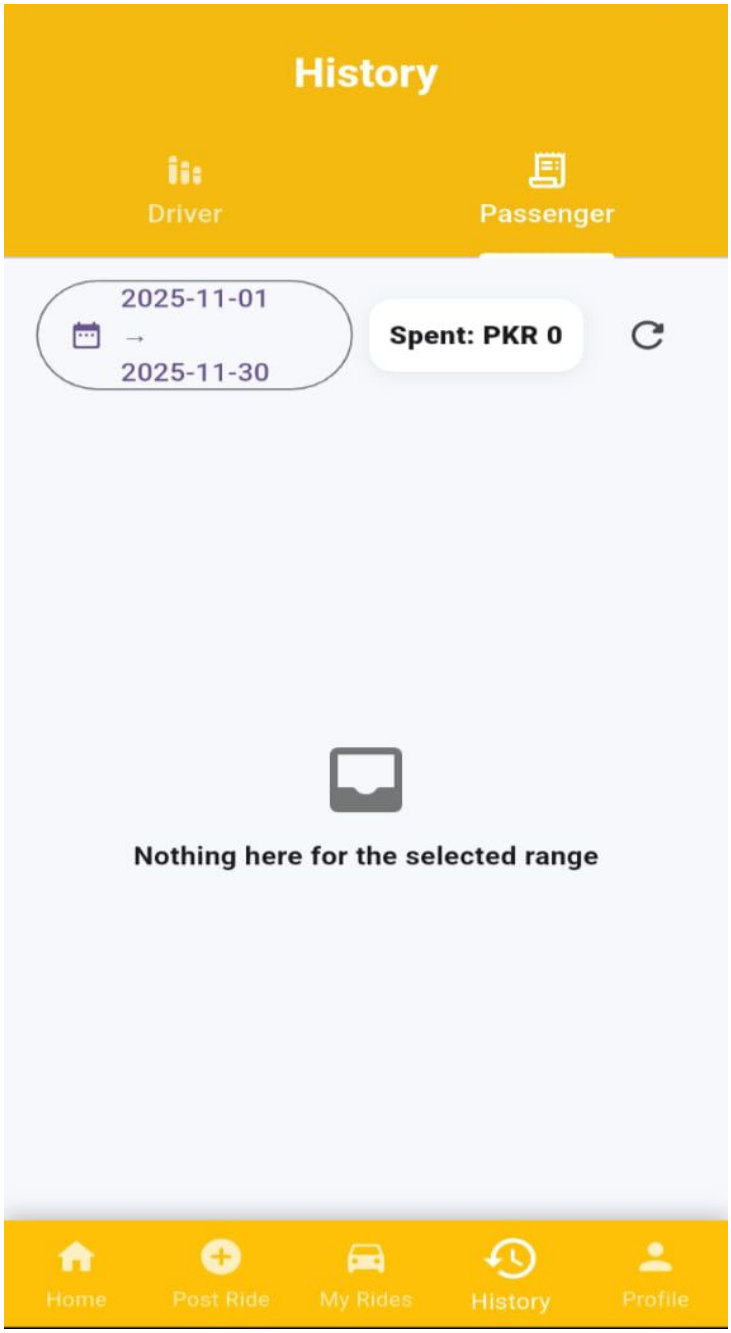


Figure 4.19: History Dashboard

**Description:** This dashboard enables users to view their ride history within a selected date range and track total spending. If no rides exist for the chosen period, the system clearly indicates the absence of records.

## 4.8 External Interfaces

To provide a very efficient, safe and smooth experience Wheel Share works with various external services. Supabase is an authentication that provides user authentication, transactions, and passwords. Supabase database is a postgreSQL that handles operations related to ride, offers, transactions, and log emergency in addition to providing RLS as protection. The ride-related images, documents or receipts are stored securely by Supabase storage on signed URLs that provide limited access. The Realtime Service allows the use of the WebSocket channel on two-way communication to immediately update status, and the Email Gateways serve to notify about the offer acceptance, ride confirmations, and emergency messages. All external communication is carried out through the HTTPS or web socket protocols and the authentication is used to have secure and role-based access to data and integrity.

# Chapter 5

# System Implementation

The chapter explains the steps that the WheelShare system implementation process will follow, how the conceptual design may be transformed into a completely effective, intelligent carpooling service. It explains the development of the frontend, backend and database parts, their integration, and testing to address the nonfunctional and functional requirements. Its building system is based on Flutter and cross-platform mobile and web application but Supabase is the backend service, which supports authentication, storage, real-time updates, and database management.

## 5.1 System Architecture

WheelShare architecture is a programmed system to be implemented as a modular and scalable system, whereby there is an efficient communication between components. It is applicable to visualization in form of three major layers:

- **Frontend (Flutter):** Its front is backed up by Flutter and provides role based dashboards to Riders, Drivers and Administrators. It enjoys simple structure; thus navigation and real-time updates of rides and responsive platforms of Android platforms, and Web platform can be easily navigated.
- **Backend Services (Supabase):** Its back-end is Supabase that is an engine and hosted by the following fundamental services: authentication, database management (PostgreSQL with Row-Level Security), file storage and realtime sync.
- **Resource (Individual channels):** Live-time communication layer will deliver the live time information on the ride offers or confirmations, emergency notification and the state of the completion of the ride, which will give the user a perfect experience and data synchronization, in real-time.

## 5.2 Tools and Technologies

The WheelShare solution is based on a new technology stack to provide efficiency, performance, and scalability.

- **Front-end Development:** Flutter (for Android, and Web)
- **Backend Services:** Supabase (Auth, PostgreSQL, Storage, Realtime)
- **Database:** Secure unencrypted role-based data access in postgreSQL using Row-Level security (RLS).
- **Programming Languages:** Dart (Frontend), SQL (Backend Policies
- **IDE:** Visual Studio Code / Android Studio

## 5.3 Development Environment

To ensure compatibility and stable performance, the following environment was used during development and testing:

- **Operating System:** Windows 11
- **Testing Devices:** Android Emulator
- **SDKs:** Flutter 3.x, Dart 3.x
- **Database:** Supabase PostgreSQL 15.x
- **APIs:** Supabase REST endpoints and Realtime subscriptions
- **Authentication:** Email and Password-based authentication via Supabase Auth

## 5.4 Processing Logic / Algorithms

The internal working of WheelShare follows several logical processes and algorithms that power its main features.

### 5.4.1 Login and Role Routing

1. Validate user credentials via Supabase Auth.
2. Retrieve user role from the Profiles table.
3. Redirect the user to the corresponding dashboard Rider, Driver, or Admin.

### 5.4.2 Ride Posting and Offer Management

1. Rider creates a new ride post including source, destination, city, address, and date.
2. Drivers can search available rides and send offers to Riders.
3. Riders can accept or reject offers; once confirmed, rides are updated in the database.
4. Realtime notifications are triggered upon any change in offer status.

### 5.4.3 Earnings and Spending Summary

1. Driver earnings and Rider spending are extracted from ride transaction records.
2. Summaries are computed based on a selected date range.

### 5.4.4 Emergency Alert Handling

1. The user is able to activate an emergency alarm inside the app.
2. The notification is immediately forwarded to Administrators and close registered contacts through Realtime notifications.
3. All the alerts are documented in the database to be audited and analysed in future.

## 5.5 Integration with Supabase

All the parts of the system interact with each other smoothly using the Flutter client SDK of Supabase. Supabase replaces a custom backend server and handles:

- Authentication
- Database operations,File storage
- Realtime synchronization

## 5.6 Testing and Deployment

WheelShare system was thoroughly tested to ensure it is stable, secure and easy to use.

### 5.6.1 Testing

To guarantee system dependability, testing was done at several levels. Workflows for ride posting, offer acceptance, and authentication were all tested. Integration testing confirmed the accuracy of earnings and spending summaries as well as the processing of emergency alerts. UI testing concentrated on search filter efficacy, dashboard accessibility, and navigation.

### 5.6.2 Deployment

Android platforms are used for mobile deployment. WheelShare's online version is hosted by Firebase Hosting or Supabase Hosting. GitHub Actions is used to manage Continuous Integration and Deployment (CI/CD), which guarantees automated builds, upgrades, and seamless delivery of new features.

## 5.7 Security Measures

WheelShare uses a number of security procedures to guarantee data safety. To ensure secrecy, sensitive data including login credentials is encrypted. While Row Level Security (RLS) stops unwanted access at the database layer, authentication and role-based access control limit users to the functions permitted for their roles. Additionally, the system keeps track of important activities like ride postings and warnings in real time for auditing purposes. Furthermore, secure HTTPS encrypted APIs are used for all connection between the app and Supabase to guarantee secure data transfer.

# Chapter 6

# System Testing and Evaluation

In this chapter, the testing and evaluation process of the Wheel Share system will be introduced so that it should be effective and will address all the requirements of the user. Various forms of testing that include GUI testing, usability testing, performance testing, compatibility testing and exception handling were carried out in order to be assured that the system operates as desired by both the Riders, Drivers and Admins.

## 6.1 Importance of System Testing

The tests on the system are critical to the quality and reliability of the Wheel Share application. It also checks that all the parts, such as posting of rides or accepting offers and emergency notifications, are functioning as desired. Systematic testing was also used to detect possible errors and rectify before real deployment. This would ensure that there is a user friendly and safe user experience and the system develops the confidence of the user in the performance and stability of the system[10].

## 6.2 Graphical User Interface (GUI) Testing

Graphical User Interface (GUI) testing will deal with confirming the visual and interactive interface of the Wheel Share app. All the screens and dashboards have been tested to launch menus, icons, buttons and flow of navigation both on the web and mobile platform. Responsiveness, consistency of layouts and accuracy of design were given special consideration. The interface was verified to be clean, intuitive and easy to use and offered a smooth navigation through the Riders, Drivers and Admins.

## 6.3 Usability Testing

The usability testing was performed to determine the ease with which this application (Wheel Share) could be navigated and used by the users. A team of Riders, Drivers, and Admins were consulted to execute more routine duties including adding rides, taking in offers, and making emergency alerts. The findings revealed that the application was not only easy to navigate but was also user friendly with the instructions being very simple and very little process was involved to accomplish each task. The user-friendly design of the app enhanced the use experiences and satisfaction.

## 6.4 Software Performance Testing

The performance testing was done to assess the behaviour of the Wheel Share system under different load directions. They tested the speed with which they could respond to the actions of ride search, offer submission and dashboard loading. The system did not have a decline in performance even upon having more users of the system simultaneously. The average search time of the ride took less than two seconds and the dashboard summary took an average of three seconds to load. These are the findings indicating the necessary requirements of scalability and speed of the application.

## 6.5 Compatibility Testing

The compatibility testing was made to ensure that Wheel Share system will work across various devices, operating systems and browsers. The app has been made to work with Android, and web browsers, including Chrome, Edge, and Safari. It worked effectively in every platform it had similar graphic effects and functionality. This is to make sure that users can use Wheel Share in any modern gadget without any performance problems and disfiguration on the interface.

## 6.6 Exception Handling

Exception handling test was done to ensure that the system responds to the wrong or unexpected messages. The system showed relevant error messages when the users typed wrong credentials or left some required fields blank or even trying to do actions without authorization. This increased stability and avoided crashing of the systems so that all the users had a secure and stable environment.

## 6.7 Test Cases

Various test cases were carried out to test the basic functionality of the system. These comprised of login authentication, Ride creation, offer management, emergency alert triggering, earnings and spending summaries. Every test case was verified to attest the proper functioning and system work. The system successfully passed the majority of test cases, having more than 95 percent accuracy on function test and integration test.

### 6.7.1 Test Case: Login

| | |
|---|---|
| **Test Case ID** | 0001 |
| **Test Case** | Login with valid credentials (Rider/Driver/Admin) |
| **Pre-Conditions** | User exists, email verified, active status |
| **Steps** | 1. Open login screen.<br>2. Enter email and password.<br>3. Submit the login form. |
| **Expected Result** | User is redirected to the respective dashboard and a secure session is established. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.1: Login Test Case

**Description:** This test case validates that authorized users can successfully log into the system using valid credentials and access their respective dashboards securely.

### 6.7.2 Test Case: Ride Posting

| | |
|---|---|
| **Test Case ID** | 0002 |
| **Test Case** | Rider: Post a Ride Request |
| **Pre-Conditions** | Rider logged in; ride form available |
| **Steps** | 1. Enter ride details (source, destination, date, time).<br>2. Submit the ride request. |
| **Expected Result** | Ride request is saved successfully, becomes visible to drivers in the same city, and a notification is sent. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.2: Test Case for Ride Posting

**Description:** This test case verifies that a rider can successfully post a ride request and that the request is properly stored, displayed to relevant drivers, and notifications are triggered.

### 6.7.3 Test Case: Driver Ride Offer

| | |
|---|---|
| **Test Case ID** | 0003 |
| **Test Case** | Driver: Offer Ride |
| **Pre-Conditions** | Driver logged in; rides available |
| **Steps** | 1. Select a ride post.<br>2. Send ride offer.<br>3. Rider accepts or rejects the offer. |
| **Expected Result** | Offer status is updated, real-time notifications are sent, and the ride is confirmed upon acceptance. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.3: Driver Ride Offer and Acceptance

**Description:** This test case ensures that drivers can successfully offer rides to riders and that the system correctly handles offer status updates, notifications, and ride confirmation.

### 6.7.4 Test Case: Emergency Alert

| | |
|---|---|
| **Test Case ID** | 0004 |
| **Test Case** | Emergency Alert Trigger |
| **Pre-Conditions** | User logged in; network connected |
| **Steps** | 1. Tap the emergency alert button.<br>2. Confirm the emergency alert. |
| **Expected Result** | Notification is sent to the admin and emergency contacts, and the alert is logged in the database. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.4: Test Case for Emergency Alert Trigger

**Description:** This test case verifies that users can successfully trigger an emergency alert and that the system promptly notifies relevant authorities and records the alert for safety purposes.

### 6.7.5 Test Case: Admin Dashboard

| | |
|---|---|
| **Test Case ID** | 0005 |
| **Test Case** | Admin: View Dashboard & Ride Statistics |
| **Pre-Conditions** | Admin logged in; rides and users seeded |
| **Steps** | 1. Load the admin dashboard.<br>2. Apply filters by city or date.<br>3. Drill down into ride details. |
| **Expected Result** | Dashboard aggregates match the database records, and drill-down views display correct ride information. |
| **Actual Result** | As expected |
| **Status** | Pass |

Table 6.5: Test Case for Admin Dashboard and Ride Statistics

**Description:** This test case ensures that the admin can view accurate dashboard analytics and ride statistics, apply filters, and access detailed ride information correctly.

### 6.7.6 Security Test Scenario

| **Test Scenario** | Result / Enforcement |
|---|---|
| **Unauthorized role attempts to accept rides on behalf of others** | Action denied by role-level security (RLS) |
| **Cross-city ride access attempt** | No records returned due to access restrictions |
| **Unsigned GET request to private profile images** | Request blocked with 403 Forbidden response |
| **SQL/JavaScript injection attempts via form inputs** | No impact due to parameterized queries and input validation |
| **Fake or modified emergency alert submission** | Request denied by security policy enforcement |

Table 6.6: Negative and Security Test Results

**Description:** This security test scenario verifies that the system enforces role-based access control and protects against unauthorized access and common security vulnerabilities.

# Chapter 7

# Conclusion

## 7.1 Summary

The Wheel Share has effectively provided a secure, scalable, user-friendly service to the citizens of the contemporary city, namely ride-sharing. The system is built to enable a modular structure and developed on Flutter to render cross-platform compatibility and Supabase as the backend service, which offers a single platform to the Riders, Drivers, and Administrators. It handles the necessity functionality like posting of rides, offering, real-time alerts, and emergency notifications[1]. There is also a seamless integration of Supabase to serve as an authentication tool, database management and real time communication tool to guarantee smooth data synchronization and security. In general, Wheel Share achieves its goals of establishing the safe, convenient, and efficient ride-sharing as well as building trust and reliability among consumers. The system creates a foundation on which scalable and data driven urban mobility solutions to accommodate future technological changes can fit through centralization of core operations into one framework.

## 7.2 Future Work

To extend the performance and possibilities of the Wheel Share system, it can be seen that a number of improvements will be considered in its further development. Caching mechanisms, asynchronous query processing, and read replicas can be included in order to scalability and achieve good performance in order to maintain low-latency responses even when there is a high traffic. The application of AI-based automation may also enhance the precision of ride-matching, add dynamic fare suggestions, and also predictive notifications to foster more user interaction[15].

# References


[1] L. Mitropoulos, A. Kortsari, and G. Ayfantopoulou, *A systematic literature review of ride-sharing platforms, user factors and barriers*, European Transport Research Review, 13(61), 2021. (Cited on pp. 1, 6, 9, 21, and 54)

[2] M.J. Alonso-González et al., *Determinants of willingness to share rides in ride-hailing*, PLoS ONE, 15(9), 2020. (Cited on pp. 6 and 9)

[3] R. Gangadharaiah et al., *Development of the pooled rideshare acceptance model (PRAM)*, Safety (MDPI), 9(3):61, 2023. (Cited on pp. 3, 6, and 9)

[4] P. Ashkrof et al., *Ride acceptance behaviour of ride-sourcing drivers*, Transportation Research Part A: Policy and Practice, 160:182–197, 2022. (Cited on pp. 2 and 6)

[5] N. Merat, R. Madigan, and S. Nordhoff, *Human factors and user acceptance of ride-sharing in automated vehicles*, ITF Discussion Paper 2017-10, OECD/ITF, 2017. (Cited on p. 7)

[6] M. Abouelela et al., *Adoption and frequency of pooled vs. solo ride-hailing*, Transportation Research Part A, 156:27–39, 2022. (Cited on p. 7)

[7] Z. Dastani et al., *User preferences in ride-sharing: Mathematical model integrating transfer preferences*, Scientific Reports, 14:78469, 2024. (Cited on p. 7)

[8] J. Soria and A. Stathopoulos, *Socio-spatial differences in solo vs pooled rides*, Transportation Research Interdisciplinary Perspectives, 2021. (Cited on pp. 4 and 7)

[9] J. Compostella et al., *Factors affecting pooled ride-hailing in driverless scenarios*, Transportation Research Part A, 2025. (Cited on pp. 5 and 7)

[10] B. Chaudhry et al., *Passenger safety in ride-sharing services*, Procedia Computer Science, 138:624–629, 2018. (Cited on pp. 7, 10, 11, and 48)

[11] R.A. Acheampong et al., *Societal impacts of smart digital mobility services: Safety and security*, Transportation Research Interdisciplinary Perspectives, 11:100391, 2021. (Cited on pp. 7, 10, and 12)

[12] J. Mageto et al., *Perceived risks and intention to use ride-hailing*, Journal of Transport and Supply Chain Management, 19:1173, 2025. (Cited on p. 7)

[13] I. Hussain et al., *Commuters' preferences and future intentions for ride-sharing*, Transport Policy, 2025. (Cited on pp. 7 and 10)

[14] M. Taiebat, E. Amini, and M. Xu, *Sharing behavior in ride-hailing trips: A machine learning inference*, arXiv preprint, 2022. (Cited on pp. 8 and 21)

[15] M. Bujak and R. Kucharski, *Balancing profit and traveller acceptance in personalized ride-pooling fares*, arXiv preprint, 2024. (Cited on pp. 8 and 54)

[16] R. Taylor et al., *Experian Hunter: Real-time fraud detection using predictive analytics*, Journal of Insurance Technology, 34(2):210–221, 2023. (Cited on pp. 8 and 10)

[17] J. Smith et al., *Actimize Fraud Management: Predictive fraud detection in insurance*, Journal of Fraud Prevention Technologies, 12(3):115–130, 2025. (Cited on pp. 8 and 10)

[18] A. Lee et al., *FICO Falcon Fraud Manager: Real-time fraud detection*, Journal of Fraud Prevention Technologies, 14(1):50–65, 2022. (Cited on pp. 9 and 10)

[19] B. Johnson et al., *ClearSale: AI + human expertise for fraud prevention*, Journal of Fraud Management and Prevention, 20(2):88–102, 2019. (Cited on pp. 9 and 10)