

# Ride Share : New way of sharing rides



Submitted By:

B Darshan Reddy: AM.EN.U4CSE22113

M V Sai Nitheesh Reddy: AM.EN.U4CSE22134

P Mouli: AM.EN.U4CSE22140

**AMRITA SCHOOL OF COMPUTING**

*(Estd. U/S 3 of the UGC Act 1956)*

**AMRITAPURI CAMPUS, KOLLAM -690525**

**JUNE 2025**

# Abstract

Urban transportation is increasingly strained by traffic congestion, rising fuel costs, limited parking, and growing environmental concerns due to increasing carbon emissions. Many commuters struggle to find reliable, affordable, and convenient transportation options, while excessive dependence on private vehicles further aggravates pollution and urban traffic density. Public transportation systems often fall short of meeting the dynamic and flexible needs of today's urban population. In light of these challenges, RideShare has been developed as a comprehensive solution to facilitate efficient, secure, and environmentally sustainable carpooling.

**RideShare** is a web-based platform that connects drivers and passengers who are traveling along similar routes. The system allows users to create accounts with secure verification through email, phone numbers, or government IDs to ensure the safety and authenticity of its user base. Drivers can post ride details, including routes, schedules, pricing, and preferences, while passengers can search for rides matching their needs. Intelligent ride-matching algorithms analyze route compatibility, timing, user ratings, and pricing to suggest the best available options. To enhance coordination and communication, the platform includes real-time notifications and an in-app chat feature for discussing trip details, pickup points, and any special requests.

The platform also integrates secure payment processing with multiple options such as credit/debit cards, digital wallets, and UPI, along with automated fare-splitting to ensure transparent cost-sharing among riders. RideShare was developed following core software engineering principles, incorporating modular system design, UML-based modeling, iterative development, and thorough testing to ensure reliability, scalability, and long-term maintainability. With future plans for integrating features like GPS-based location tracking, real-time traffic data, and dynamic pricing, RideShare offers a robust, user-friendly solution that promotes safe, affordable, and eco-friendly commuting while reducing traffic congestion and lowering environmental impact.

# Contents

<b>Abstract</b>	<b>1</b>
<b>1 COMMUNICATION</b>	<b>4</b>
1.1 Client Information . . . . .	4
1.1.1 Meeting Minutes with Project Owner . . . . .	5
1.2 Problem Statement . . . . .	7
<b>2 PLANNING</b>	<b>9</b>
<b>PLANNING</b>	<b>9</b>
2.0.1 Deadline . . . . .	9
2.0.2 About The Team . . . . .	11
2.0.3 P Mouli (Team Lead & Authentication Developer): . . . . .	11
2.0.4 M V Sai Nitheesh Reddy (Backend & Documentation Lead): . . . . .	11
2.0.5 B Darshan Reddy(Frontend Developer & Scrum Master): . . . . .	11
2.0.6 Gopika Muraleedas (Faculty): . . . . .	12
<b>3 DESIGN</b>	<b>14</b>
3.0.1 System Architecture . . . . .	15
3.0.2 Data Flow Diagram Level 0: . . . . .	25
3.0.3 Level 1: . . . . .	25
3.0.4 Level 2 . . . . .	26
<b>4 IMPLEMENTATION</b>	<b>30</b>
<b>5 TESTING</b>	<b>42</b>
5.1 Testing . . . . .	42
5.2 Conclusion . . . . .	44

# CHAPTER 1

## COMMUNICATION

# COMMUNICATION

## 1.1 Client Information

The primary users (clients) of our TripMate system are individuals who need affordable, reliable, and flexible transportation options for their daily commuting needs. Urban commuters often face challenges such as high fuel prices, traffic congestion, and limited parking, while existing ride-sharing platforms may not fully address the needs of casual users, students, or those seeking peer-to-peer carpooling.

We also considered a few other people and groups as part of the system's users:

### 1. Commuters (Main Users)

- Daily travelers such as students, office workers, and casual commuters.
- Need access to affordable and flexible ride-sharing options.
- Prefer a secure platform with verified users and easy ride booking.

### 2. Drivers (Peer Drivers)

- Individuals who can offer rides while commuting to the same destination.
- Can share travel costs while utilizing empty vehicle seats.
- Seek a platform that ensures safety, transparency, and efficient matching with passengers.

### 3. Administrators (Platform Managers)

- Responsible for managing user verification, system maintenance, and overseeing platform operations.
- Monitor user feedback, resolve disputes, and ensure system security and data integrity.

Our goal was to create a user-friendly, secure, and trustworthy ride-sharing platform that allows users to act both as drivers and passengers. TripMate supports peer-to-peer carpooling, promotes community trust through OTP-based verification and ratings, and offers flexible commuting solutions while contributing to reduced traffic congestion and environmental impact.

### 1.1.1 Meeting Minutes with Project Owner

#### Attendees

#### Project Team Members

- **Palluru Mouli** – *Team Lead*
- **M. V. Sai Nitheesh Reddy** – *Developer*
- **B. Darshan Reddy** – *Business Analyst & Scrum Master*
- **Gopika Muraleedas** – *Project Owner*

#### *Main meetings*

##### Meeting 1

Date: February 21, 2025

Agenda: Problem Understanding and Research Gaps Discussed:

We discussed the main transportation challenges faced by urban commuters, such as rising fuel prices, heavy traffic congestion, and limited affordable ride-sharing options. We analyzed existing ride-sharing platforms and identified limitations like lack of peer-to-peer sharing, poor verification systems, and limited flexibility for users. The discussion helped us finalize our project idea to build a secure, peer-to-peer carpooling platform with dual roles and strong trust mechanisms.

##### Meeting 2

Date: March 21, 2025

Agenda: Planning the System and Feature Listing Points Discussed:

We listed the key features needed for TripMate, such as OTP-based registration, secure login, profile management, ride publishing and booking, real-time messaging, rating system, and secure payment options. Tasks were divided among team members, and a phased development plan was created for smooth progress.

##### Meeting 3

Date: April 22, 2025

Agenda: Design and Diagrams Points Discussed:

We worked on system design, creating UML diagrams such as use case, activity, class, and data flow diagrams. The database schema was designed, and we discussed how the frontend and backend components would interact. The architecture for user verification, ride management, and secure data handling was finalized.

#### **Meeting 4**

Date: May 9, 2025

Agenda: Development Phase Start Points Discussed:

We started developing key modules like user authentication, ride creation, profile setup, and OTP-based verification. The frontend team focused on building UI components using React and Tailwind CSS, while the backend team implemented APIs and database integration using Node.js, Express.js, and MongoDB.

#### **Meeting 5**

Date: May 23, 2025

Agenda: Ride Search and Booking Integration:

Integration of ride search, booking system, seat management, and in-app messaging was completed. Testing was conducted for search accuracy, seat updates, and chat features. Issues such as race conditions during booking and data synchronization bugs were identified and addressed.

#### **Meeting 6**

Date: June 6, 2025

Agenda: UI/UX Improvements and User Testing:

We gathered feedback from classmates and faculty. Based on the feedback, we improved the user interface, polished form validations, optimized error handling, and simplified navigation. Features like ride history, preferences management, and responsive design adjustments were also implemented.

#### **Meeting 7**

Date: June 13, 2025

Agenda: System Testing and Bug Fixing:

Comprehensive unit testing, API testing, and end-to-end tests were conducted. Minor functional and UI bugs were fixed. System performance and security were validated to ensure reliable behavior under different usage scenarios.

#### **Meeting 8**

Date: June 16, 2025

Agenda: Final Deployment, Documentation, and Viva Preparation:

The team completed final deployment tasks, cleaned up the codebase, updated all documentation including UML diagrams, and prepared the final report. The system was

fully validated, and demo preparation was completed for the final presentation and viva evaluation.

## 1.2 Problem Statement

Urban commuting today faces significant challenges such as rising fuel costs, heavy traffic congestion, limited parking, and increasing environmental concerns due to high carbon emissions. Commuters—including students, office workers, and casual travelers—struggle to find affordable, reliable, and flexible transportation options that suit their daily schedules. Existing ride-hailing platforms are often commercialized, with high surge pricing and a focus on professional drivers, excluding those who simply want to share rides for cost-saving and convenience. Carpool platforms that do exist often serve limited user groups and lack strong verification and trust mechanisms, making users hesitant to adopt peer-to-peer sharing.

There is a strong need for a secure, community-oriented ride-sharing platform that allows users to alternate between driver and passenger roles while ensuring trust, flexibility, and safety. The platform must include features like OTP-based user verification, secure authentication using JWT, real-time communication, transparent rating and review systems, and automated fare-splitting to encourage fair cost-sharing. RideShare addresses these gaps by offering a user-friendly, secure, and scalable peer-to-peer carpooling solution that promotes affordable, eco-friendly, and reliable urban commuting.



# CHAPTER 2

## PLANNING

# PLANNING

## 2.0.1 Deadline

The project was executed following a structured, phase-wise development plan considering the complexity of each module, available resources, and academic deadlines. The following table presents the complete timeline for the development of the RideShare system:

Phase	Description	Start Date	End Date
Initiation and Problem Understanding	Discussed urban commuting issues, studied existing carpool platforms, identified gaps, finalized project scope	20-04-2025	24-04-2025
Feature Listing and Planning	Defined system features: OTP-based authentication, ride publishing, booking system, messaging, ratings, payments; assigned roles and responsibilities	25-04-2025	30-04-2025
System Design	Designed UML diagrams (Use Case, Class, Activity, Sequence, DFD), database schema, and system architecture for frontend and backend integration	01-05-2025	08-05-2025
Frontend and Backend Setup	Set up React frontend, Node.js/Express.js backend, MongoDB database configuration, basic API structure, and state management setup	09-05-2025	15-05-2025
Core Module Development	Implemented user registration, OTP-based authentication, profile management, ride creation, and ride publishing modules	16-05-2025	22-05-2025
Ride Search and Booking System	Developed ride search algorithms, seat management, ride booking workflows, and in-app real-time chat system for user coordination	23-05-2025	30-05-2025

Integration Testing	Conducted integration testing of all modules: authentication, booking, messaging, ride management, and session handling using JWT	31-05-2025	06-06-2025
Feedback-Based Enhancements	Incorporated faculty and peer feedback: UI/UX improvements, error handling, responsive design adjustments, validation enhancements	07-06-2025	12-06-2025
Final Testing and Documentation	Performed full end-to-end system testing, finalized deployment preparations, updated all diagrams and reports, completed viva preparation	13-06-2025	16-06-2025

## 2.0.2 About The Team

- **Palluru Mouli** – *Team Lead*
- **M. V. Sai Nitheesh Reddy** – *Developer*
- **B. Darshan Reddy** – *Business Analyst & Scrum Master*
- **Gopika Muraleedas** – *Project Owner*

## 2.0.3 P Mouli (Team Lead & Authentication Developer):

- **Role:** As the Team Lead, guided the project from the initial planning stages through system design and feature selection. He was primarily responsible for implementing the complete user authentication system, including OTP-based email verification, secure login, session management, and password encryption using JWT and bcrypt. He handled critical backend components for secure user registration, login flows, and profile management, ensuring robust security and smooth user onboarding.

## 2.0.4 M V Sai Nitheesh Reddy (Backend & Documentation Lead):

- **Role:** As a backend developer, developed the core backend modules of the RideShare platform, including ride publishing, ride searching, booking system, seat management, real-time messaging, and ratings using Node.js, Express.js, and MongoDB. He ensured smooth API integration with the frontend and handled data consistency, input validation, and bug fixes. In addition, he took full responsibility for preparing comprehensive documentation, including all UML diagrams, technical reports, and deployment preparation.

## 2.0.5 B Darshan Reddy(Frontend Developer & Scrum Master):

- **Role:** As the Business Analyst and Scrum Master for the RideShare project, actively managing sprint planning, task assignments, and team coordination. He analyzed system requirements, identified user needs, and ensured that the business objectives were properly translated into technical features. Alongside his management responsibilities, Darshan contributed to frontend development, designing and implementing several UI components and pages using React and Tailwind CSS. He collaborated closely with backend developers to integrate APIs and ensured the user interface offered a smooth, responsive, and user-friendly experience.

### 2.0.6 Gopika Muraleedas (Faculty):

- **Role:** Gopika Muraleedas served as the Project Owner for the RideShare project, providing continuous support and overall oversight throughout the project lifecycle. She ensured that the project stayed aligned with its objectives, provided timely feedback during design and development phases, and reviewed progress regularly. Gopika participated in requirement discussions, offered valuable suggestions for user experience improvements, and assisted the team in refining features based on feedback. Her role was crucial in maintaining project vision, ensuring completeness of deliverables, and supporting the team during documentation and final presentation preparation.

# CHAPTER 3

## DESIGN

# DESIGN

## Modeling and Design Phase

The Modeling and Design phase played a crucial role in shaping the RideShare system. It involved transforming the functional requirements into a well-structured and scalable system architecture that could handle secure, real-time ride-sharing operations while maintaining high usability and performance standards.

The design process started with identifying key system components such as user management, OTP-based authentication, ride publishing and searching, booking workflows, real-time messaging, rating systems, and payment processing. These components were modeled to ensure clear interaction between users and system modules.

A layered architecture was adopted, separating responsibilities across three main layers:

- **Frontend Layer:** Developed using React, Tailwind CSS, and Vite, responsible for providing a responsive and interactive user interface.
- **Backend Layer:** Implemented using Node.js, Express.js, and TypeScript, managing business logic, authentication, data processing, and RESTful APIs.
- **Database Layer:** Built on MongoDB Atlas with Mongoose ODM, responsible for secure, cloud-based storage of users, rides, bookings, messages, and ratings.

To visualize and validate system behavior, multiple UML diagrams were developed:

- **Use Case Diagram** to capture user interactions and functional scope.
- **Class Diagram** to define system entities and their relationships.
- **Activity Diagram** to model key system workflows.
- **Sequence Diagram** to describe data flow between components.
- **Data Flow Diagram (DFD)** to map how information moves through the system.

By modeling the system early, potential challenges were identified in advance, allowing the team to design a robust solution that balanced security, performance, and user experience. This phase laid the foundation for successful development, integration, and deployment of the RideShare platform.

### 3.0.1 System Architecture

The architecture of the RideShare system is designed to deliver a secure, real-time, and user-friendly peer-to-peer carpooling experience. It combines robust backend processing with an interactive frontend interface, ensuring seamless coordination between drivers and passengers. The architecture follows a modular, layered design which promotes scalability, maintainability, and secure data handling.

Below is a detailed explanation of each component as represented in the architecture diagram:

1. **User Management** This module handles user registration, OTP-based email verification, secure login, and profile management. The system allows users to create and update personal profiles, switch between driver and passenger roles, and securely store user information in the database. OTP verification ensures the authenticity of users, reducing risks associated with fake accounts.
2. **Authentication and Security** Authentication is implemented using JWT (JSON Web Tokens) for secure session management, ensuring that only authorized users access protected resources. Passwords are hashed using bcrypt to prevent plaintext password storage. Middleware is used in the backend to protect routes and verify user roles and permissions dynamically.
3. **Ride Management** This core module enables drivers to create ride offers by entering route details, departure times, seat availability, pricing, and ride preferences. Passengers can search for rides using intelligent filters based on location, time, pricing, and driver ratings. The module manages real-time updates to ride availability and supports ride cancellation and modifications.
4. **Booking System** The booking module ensures accurate seat allocation, booking confirmations, and handles concurrency issues during simultaneous booking attempts. It maintains passenger lists, updates ride capacity in real-time, and provides booking status tracking for both drivers and passengers.
5. **Real-Time Messaging** A built-in messaging system allows drivers and passengers to communicate directly. Implemented using WebSocket or real-time RESTful API polling, this module allows users to coordinate pick-up points, discuss schedules, and handle last-minute changes efficiently, enhancing overall user experience.
6. **Ratings and Reviews** Post-ride, users can rate and review each other, contributing to a transparent reputation system. High-rated users gain credibility, encouraging safe and trustworthy ride-sharing. Reviews are stored securely in the database and are visible to other users during ride selection.



7. **Payment Integration (Future Scope)** This module, planned for future implementation, will handle automated fare calculation and secure payment transactions. It will support multiple payment gateways, including UPI, credit/debit cards, and digital wallets, ensuring fair and seamless cost sharing between riders.
8. **Testing** Comprehensive testing, including unit testing (using frameworks like Jest), API testing (using Postman), and end-to-end testing, is conducted to ensure that each component works reliably. Test automation scripts help validate system stability across different modules.
9. **Deployment** After successful development and testing, the system is deployed on cloud infrastructure for real-world access. The deployment process involves configuring backend servers, database hosting (MongoDB Atlas), and frontend hosting, ensuring high availability and scalability under production workloads.

## UML Use Case Diagram

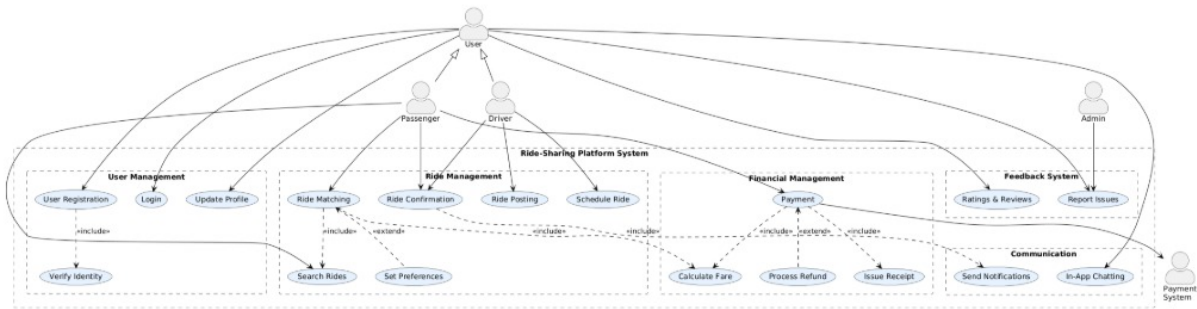


Figure 3.1: UML Use Case Diagram

## Class Diagram

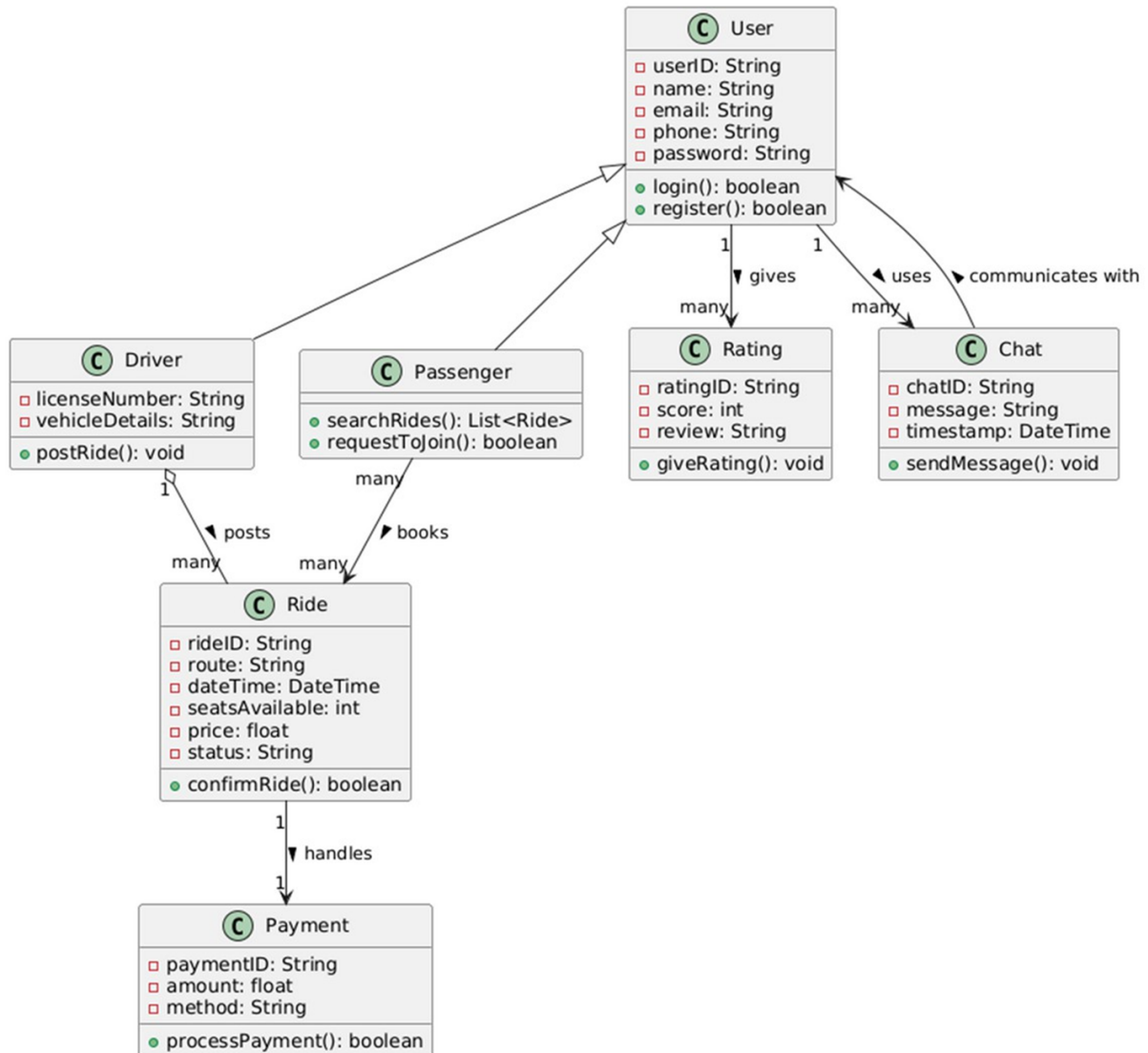


Figure 3.2: Class Diagram

## Activity Diagram

Activity diagram: Illustrates user interactions and administrative tasks within an AI Chatbot SaaS platform, highlighting sequential actions from registration and login to chatbot interaction and system management.

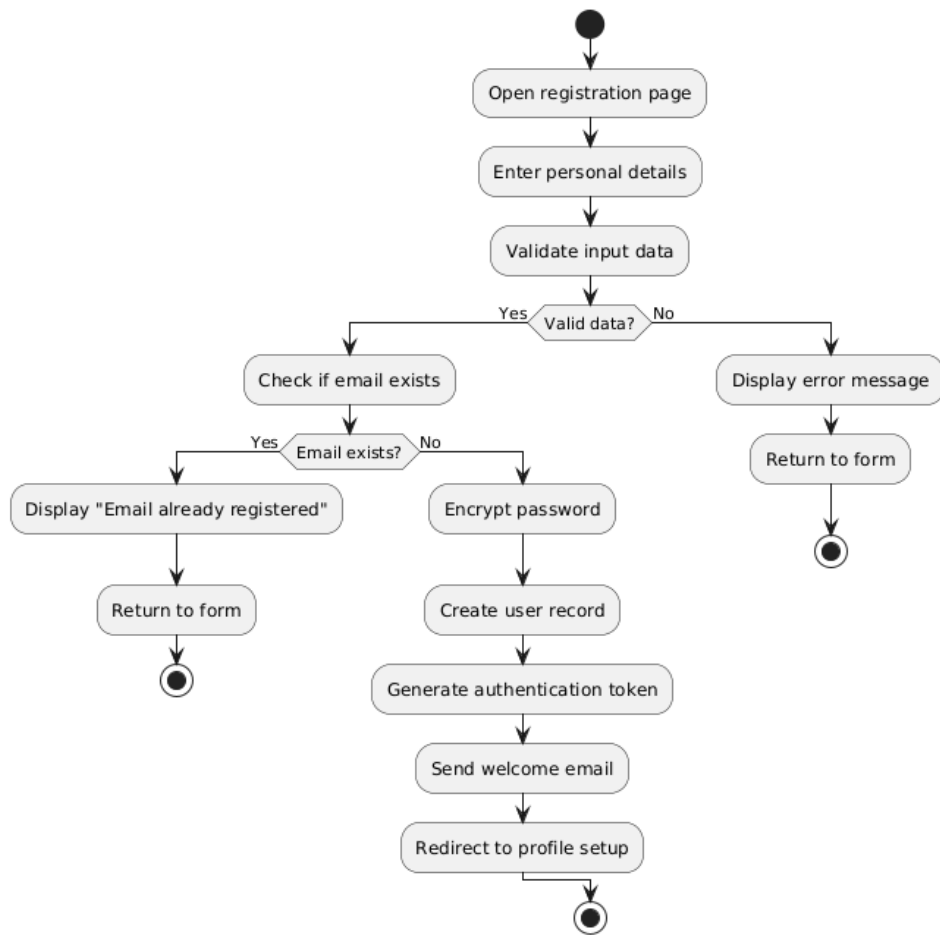


Figure 3.3: User Registration Process Activity Diagram

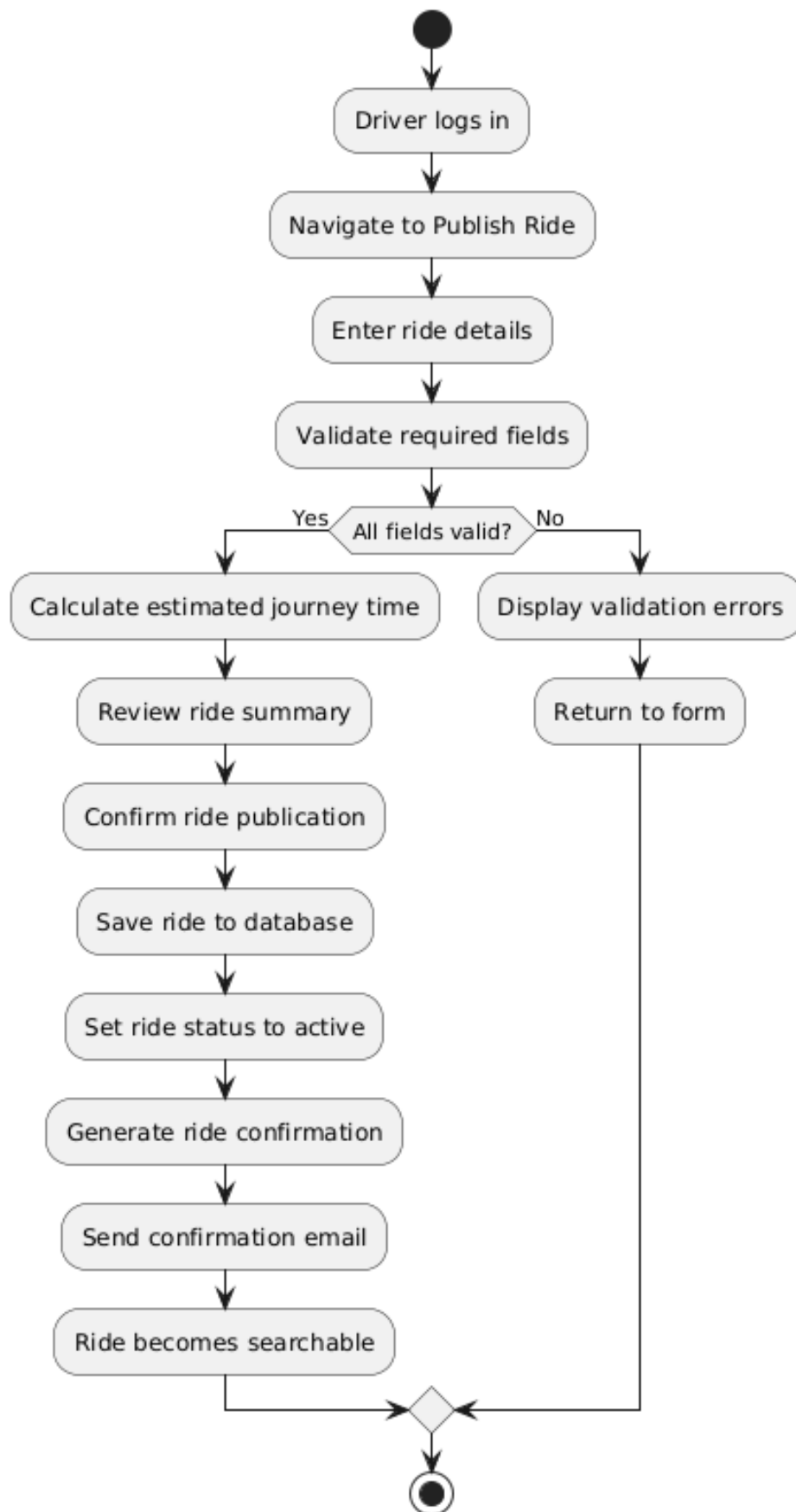
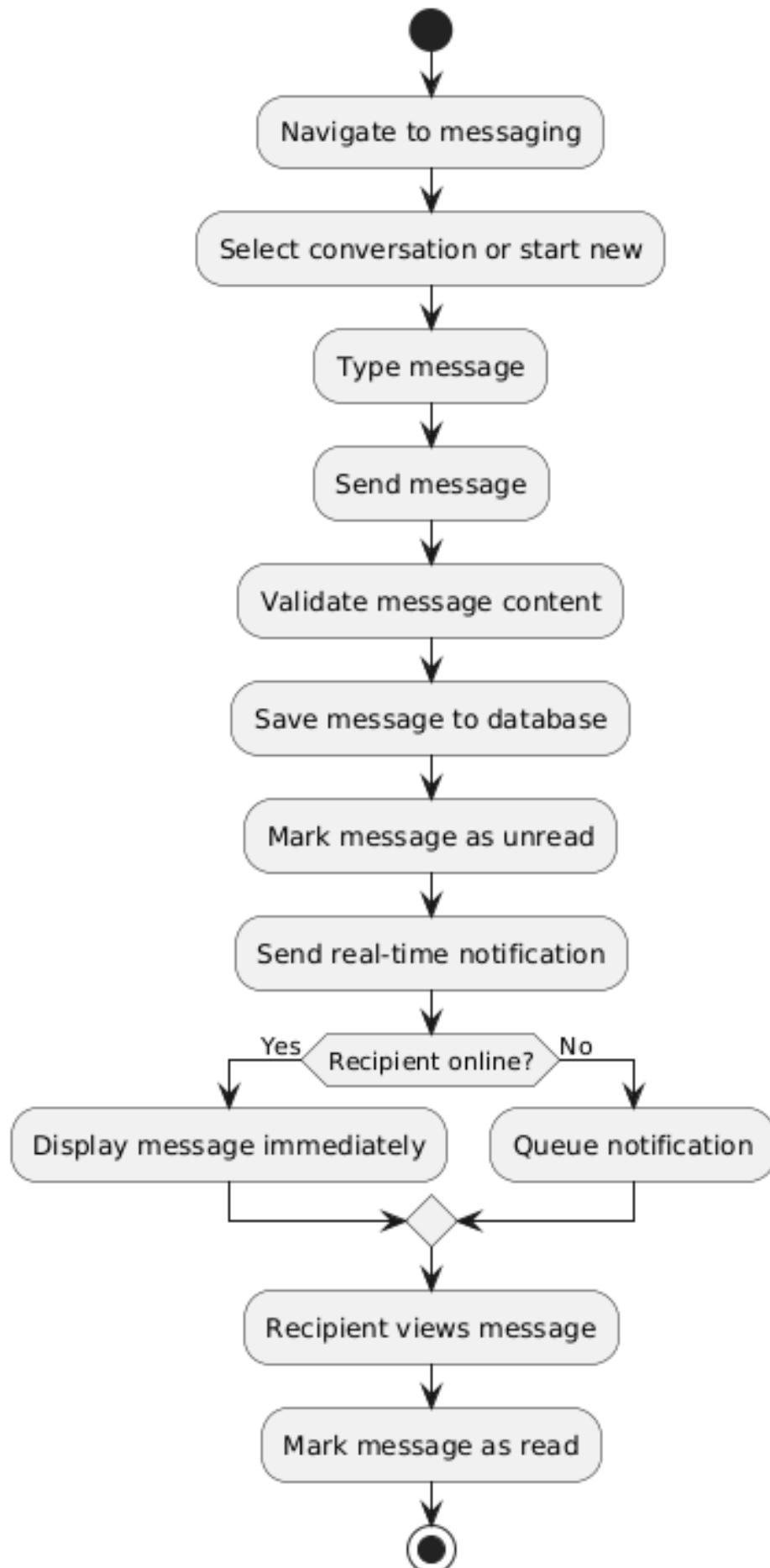


Figure 3.4: Ride Publishing Process Activity Diagram



Figure 3.5: Ride Booking Process Activity Diagram



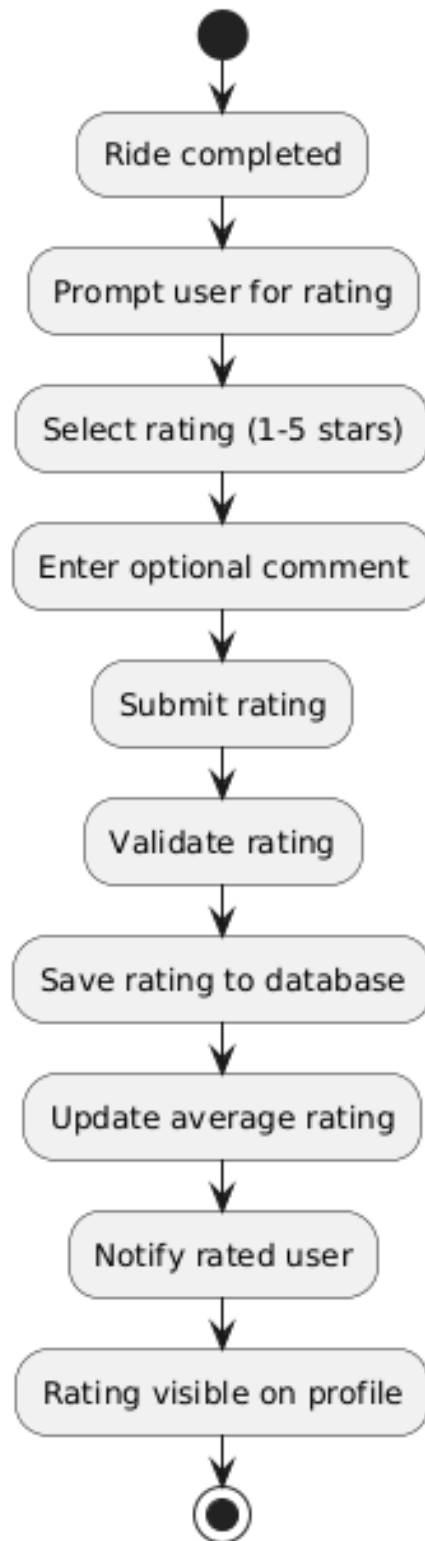


Figure 3.7: Rating System Activity Diagram



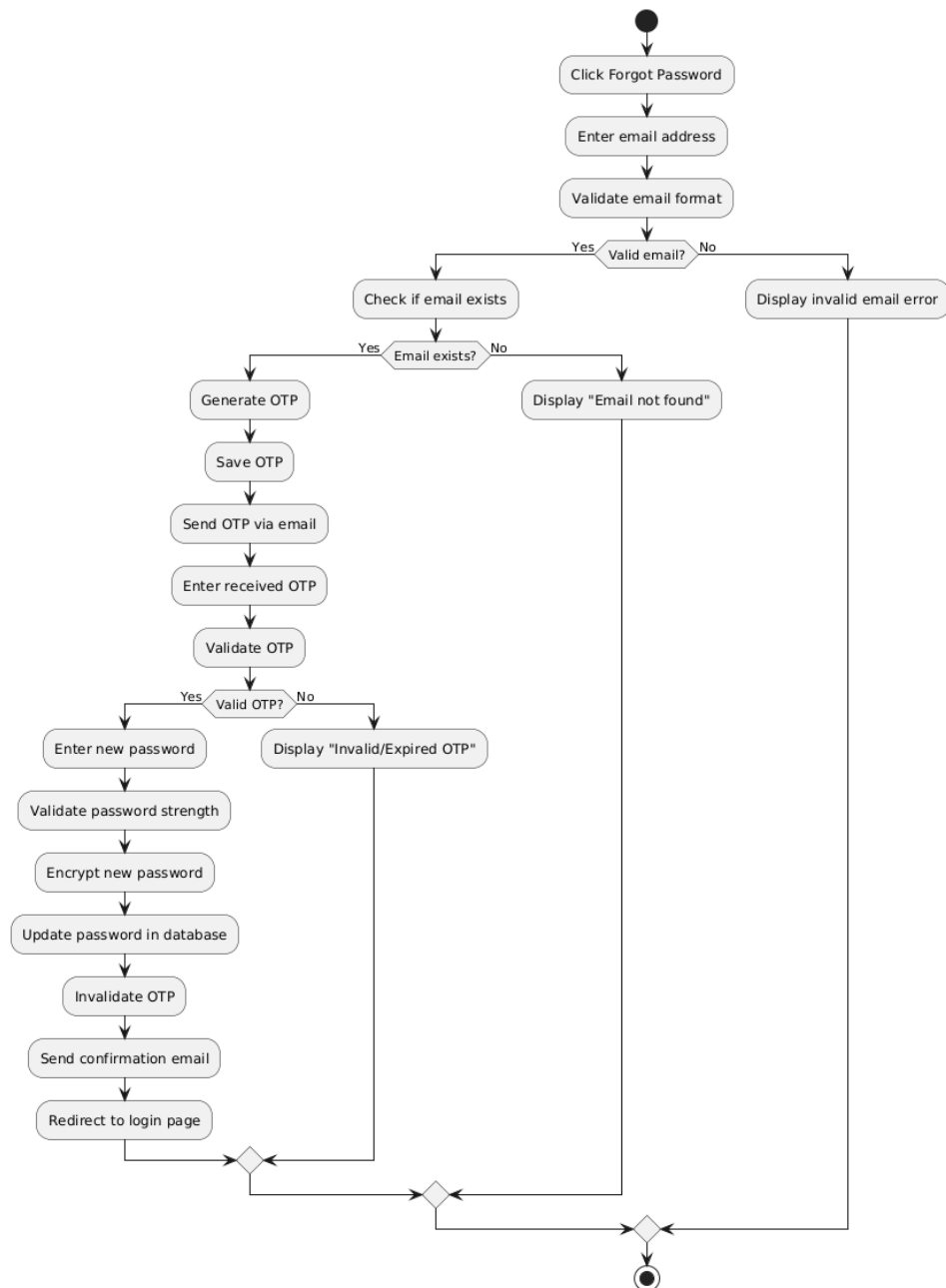


Figure 3.8: Password Reset Process Activity Diagram

### 3.0.2 Data Flow Diagram Level 0:

This diagram represents the Context-Level Data Flow Diagram (DFD) of the RideShare Connect system. It illustrates how the primary external entities—Guest Users, Registered Users, Drivers, Passengers, Email Service Provider, and Payment Gateway—interact with the system. Users can register, log in, search for rides, publish rides, book rides, and send messages. The system processes these requests, stores and retrieves relevant ride and user data, handles booking confirmations, and manages messaging and rating functionalities. It also communicates with external services for sending email notifications and processing payment transactions, ensuring smooth coordination between all involved actors.

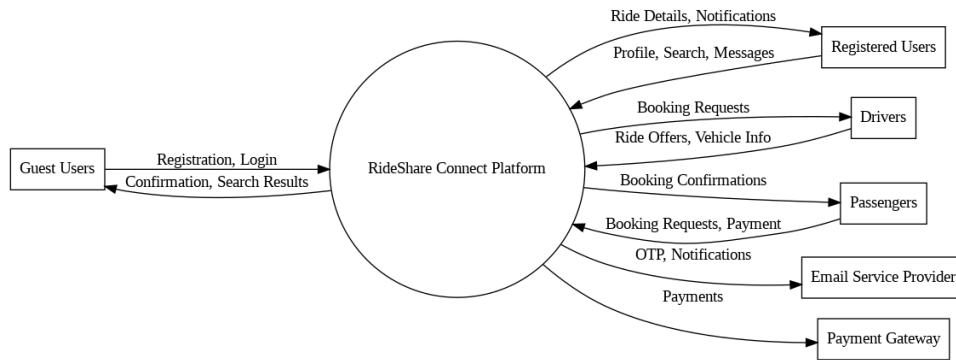


Figure 3.9: Data Flow Diagram Level 0

### 3.0.3 Level 1:

This is a Level 1 Data Flow Diagram (DFD) of the RideShare Connect system, which explains how the platform processes different user activities in detail. The process begins when users register, log in, or update their profiles through the User Management module, which stores and verifies their data. Drivers publish ride offers by entering details such as locations, timings, and seat availability into the Ride Management process. Passengers search for suitable rides, apply filters, and send booking requests, which are also handled within Ride Management. Once rides are booked, the Messaging System allows drivers and passengers to communicate directly regarding trip details. After ride completion, both parties submit their feedback through the Rating System, updating user ratings. The Notification Service ensures users receive timely email notifications, OTPs, and booking confirmations via integration with the Email Service Provider. All data exchanges are securely managed through dedicated databases for users, rides, messages, ratings, and OTPs, ensuring smooth and reliable system functionality.

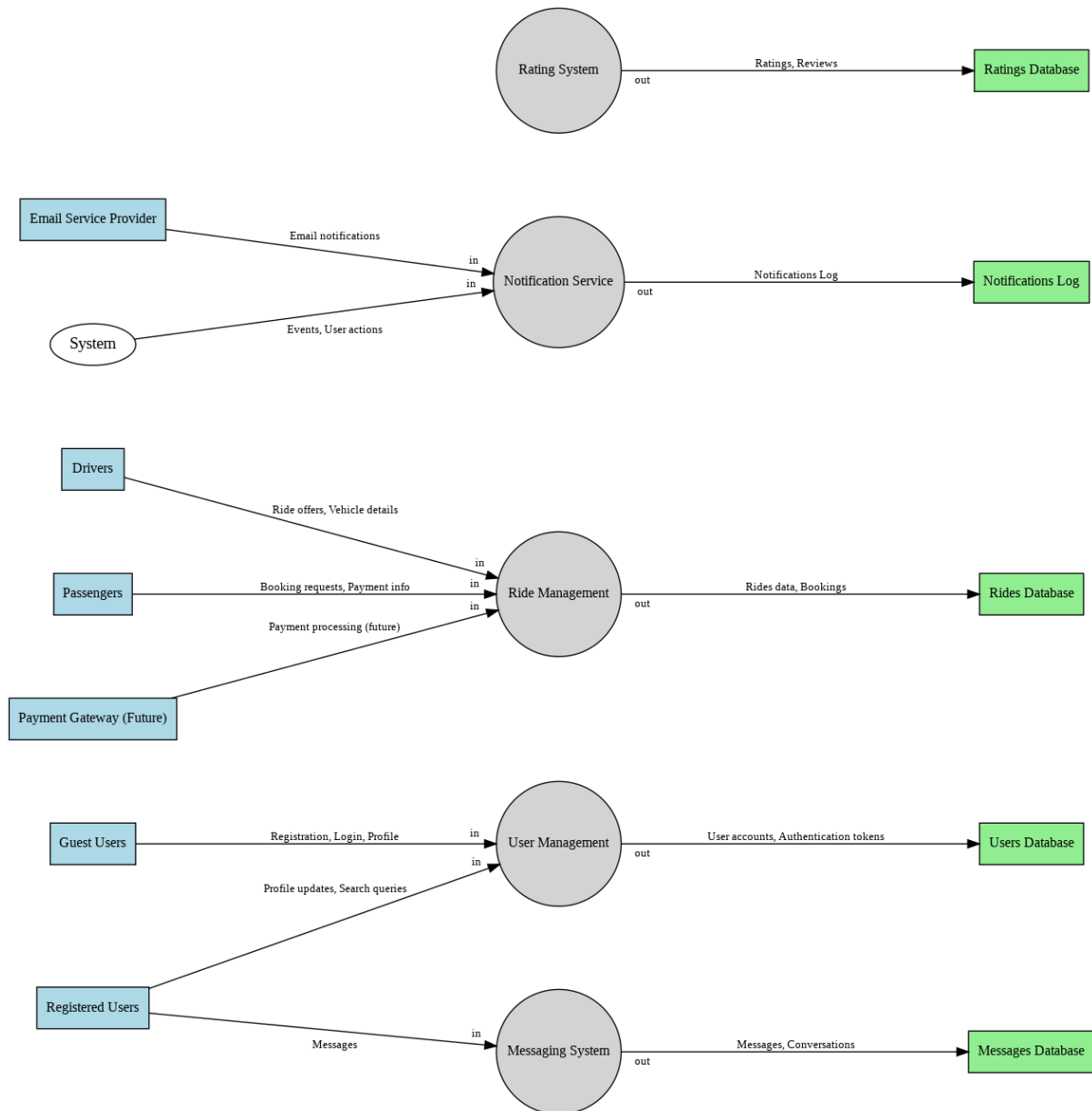


Figure 3.10: Data Flow Diagram Level 1

### 3.0.4 Level 2

This diagram shows a Level 2 Data Flow Diagram (DFD) for the RideShare Connect system, focusing in detail on how user activities are processed step-by-step within the platform.

The process begins when users register by providing their personal information. The system validates and securely stores this data in the User Database. For authentication, login credentials are verified, and successful logins generate secure user sessions. Profile Management allows users to update their information and upload verification documents.

For ride-related activities, drivers publish ride offers through the Ride Publishing module

by entering route, timing, seat availability, and vehicle details. Passengers use the Ride Searching module to filter and view available rides based on their preferences. When a passenger selects a ride, the Booking Management module handles booking requests, verifies seat availability, updates ride status, and sends confirmations to both driver and passenger.

Meanwhile, the Messaging System allows drivers and passengers to exchange messages for trip coordination. After ride completion, both parties submit feedback through the Rating Submission module, where ratings and reviews are stored and user averages are updated in the Ratings Database.

The Notification Service continuously monitors system events to trigger email alerts, OTP verifications, and booking confirmations, collaborating with the Email Service Provider. All processes interact with their dedicated databases to ensure accurate data storage, real-time updates, and smooth platform functionality.

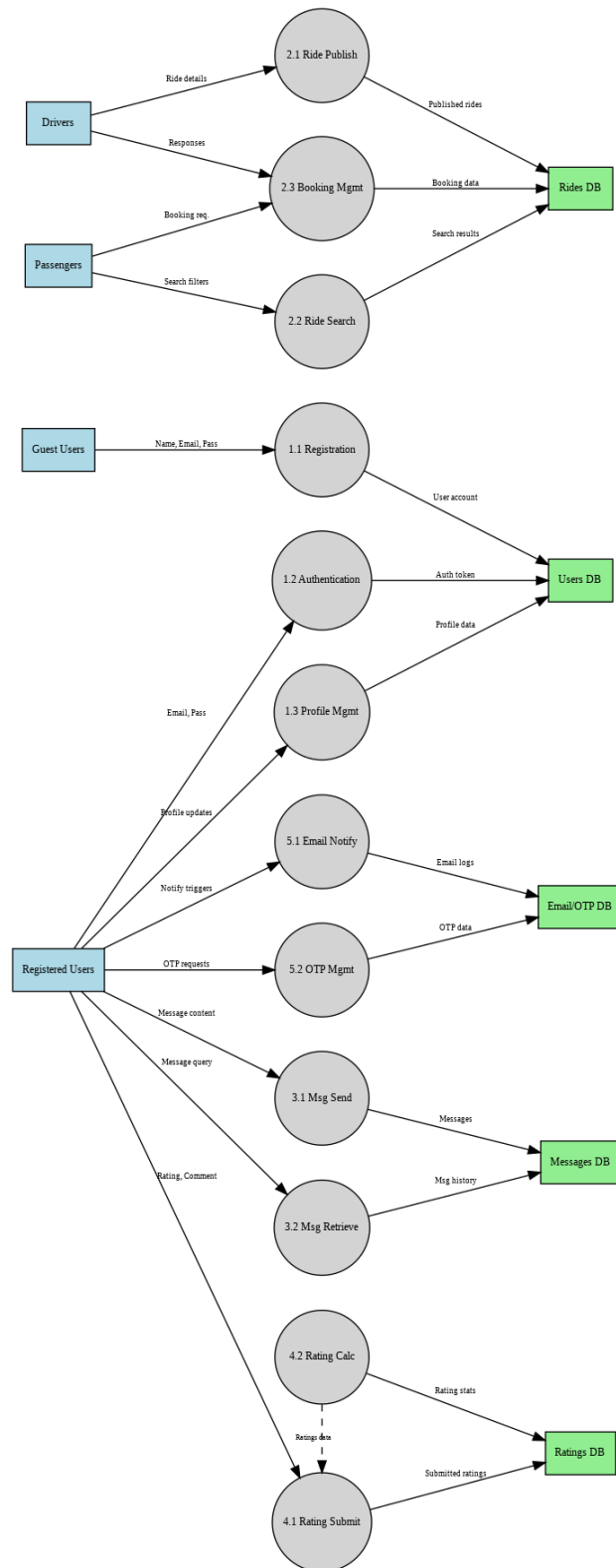


Figure 3.11: Data Flow Diagram Level 2

# CHAPTER 4

## IMPLEMENTATION

# IMPLEMENTATION

## IMPLEMENTATION

In the implementation phase of the RideShare Connect project, the system design is transformed into a fully functional web-based carpooling platform. The core modules for user registration, secure authentication, ride publishing, ride searching, booking management, messaging, ratings, and notifications are developed and integrated. Backend services connect to centralized databases to store user profiles, ride offers, bookings, messages, and ratings. Real-time messaging functionality is implemented to allow drivers and passengers to communicate seamlessly before and during rides. Payment integration modules are prepared for future enhancements. The system is designed with a user-friendly interface to ensure easy navigation for both drivers and passengers. Comprehensive testing is performed to verify system reliability, validate data flows, ensure proper handling of edge cases, and maintain smooth user experiences across various usage scenarios.

## Screenshots

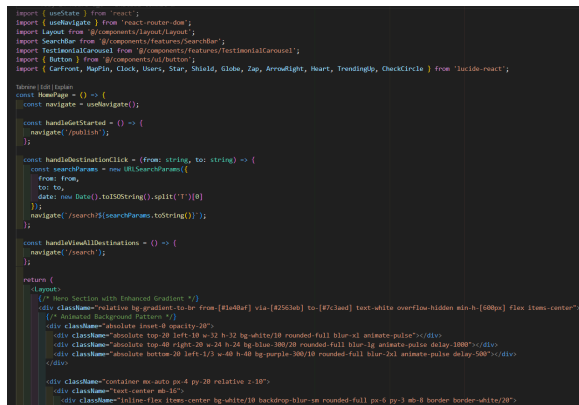


Figure 4.1: Implementation Screenshot 1

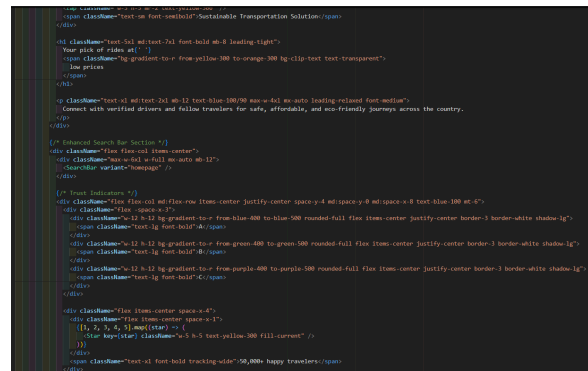


Figure 4.2: Implementation Screenshot 2

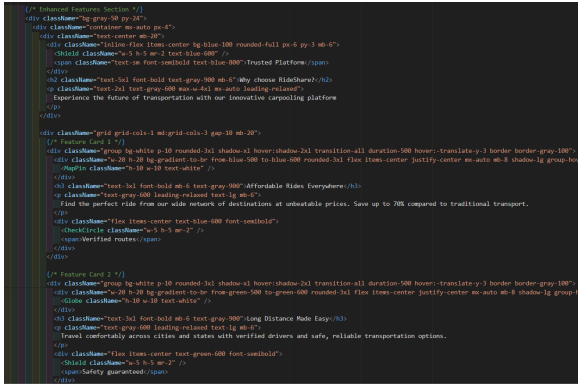


Figure 4.3: Implementation Screenshot 3

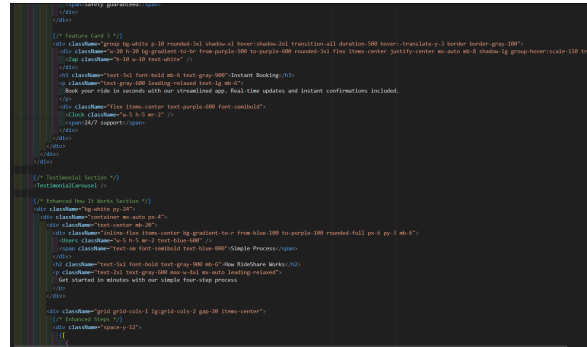


Figure 4.4: Implementation Screenshot 4

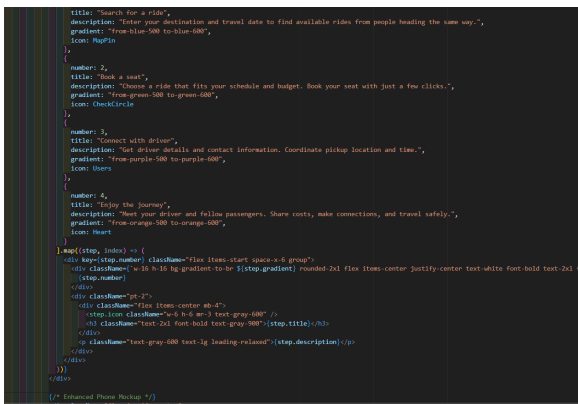


Figure 4.5: Implementation Screenshot 5

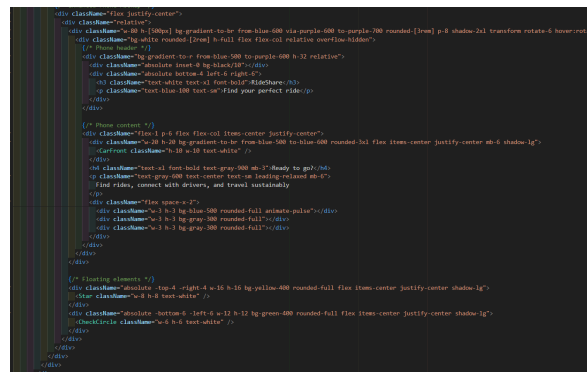


Figure 4.6: Implementation Screenshot 6

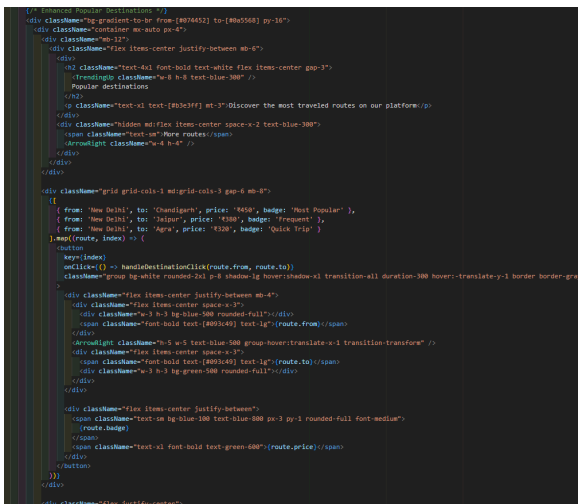


Figure 4.7: Implementation Screenshot 7

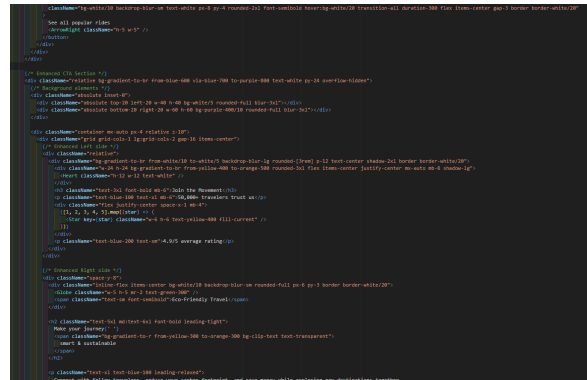


Figure 4.8: Implementation Screenshot 8



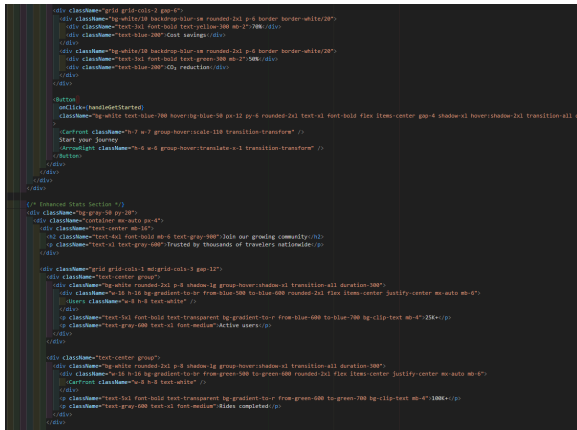


Figure 4.9: Implementation Screenshot 9

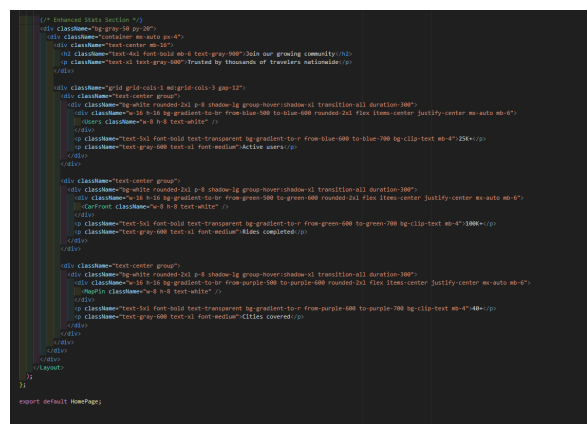


Figure 4.10: Implementation Screenshot 10

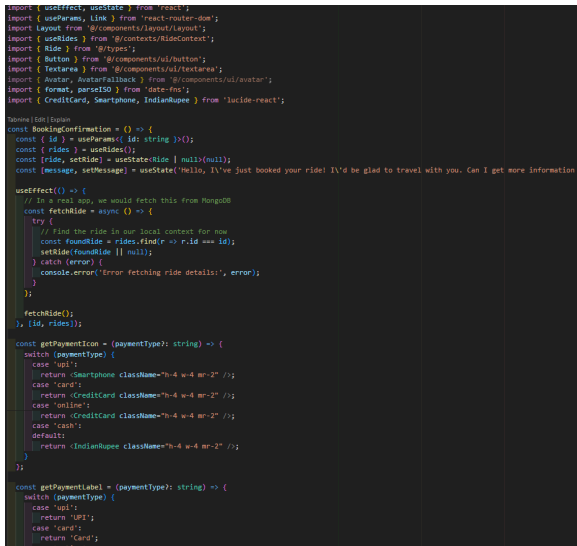


Figure 4.11: Implementation Screenshot 11

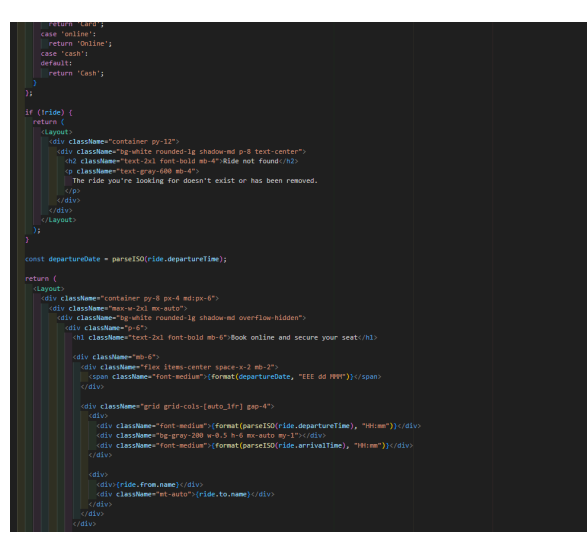


Figure 4.12: Implementation Screenshot 12



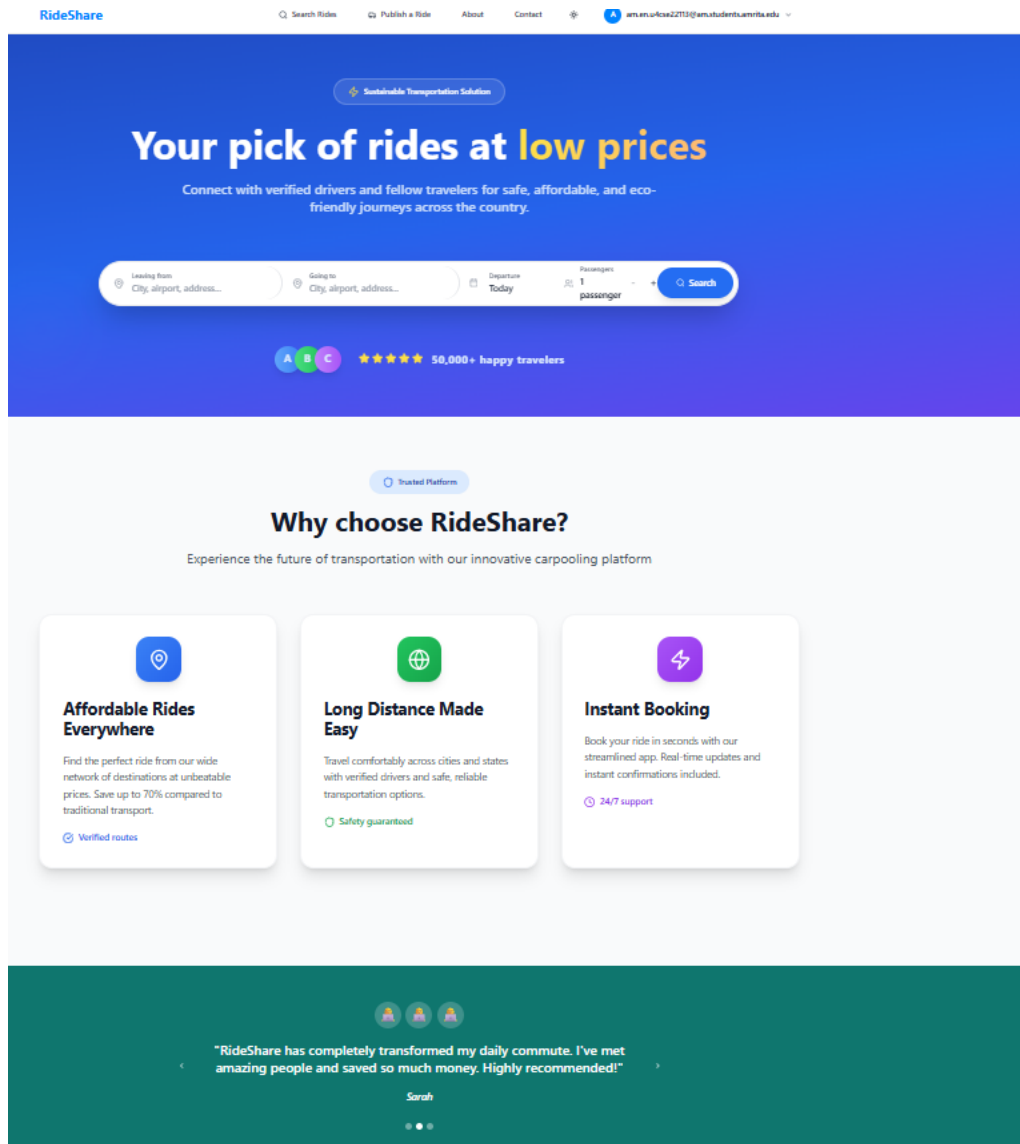


Figure 4.17: Implementation Screenshot 17

RideShare

Q Search Rides

Publish a Ride

About

Contact

am.en.u4cse22113@am.students.amrita.edu

Search Results

From

To

Passengers 1

6 rides available

Filters

Clear all

Sort by

Earliest departure

Lowest price

Close to departure

Close to arrival

Shortest duration

Departure time

06:00 - 12:00 1

12:01 - 18:00 3

18:01 - 00:00 1

00:01 - 06:00 1

Trust & Safety

Verified Drivers Only 3

Amenities

Instant Booking 6

Smoking allowed 0

Pets allowed 0

17:10 Kadapa 20:00 Kumool ₹530

M Mohan 3.7

Instant Booking

View details

17:30 Kadapa 20:20 Kumool ₹450

N Nalla 5

Instant Booking

View details

19:00 Kadapa 21:50 Kumool ₹450

V Vikram 4

Instant Booking

View details

02:00 Kayamkulam 04:00 Kochi, Kerala ₹700

Stopovers: Bellary, Karnataka

C Current User

Instant Booking

View details

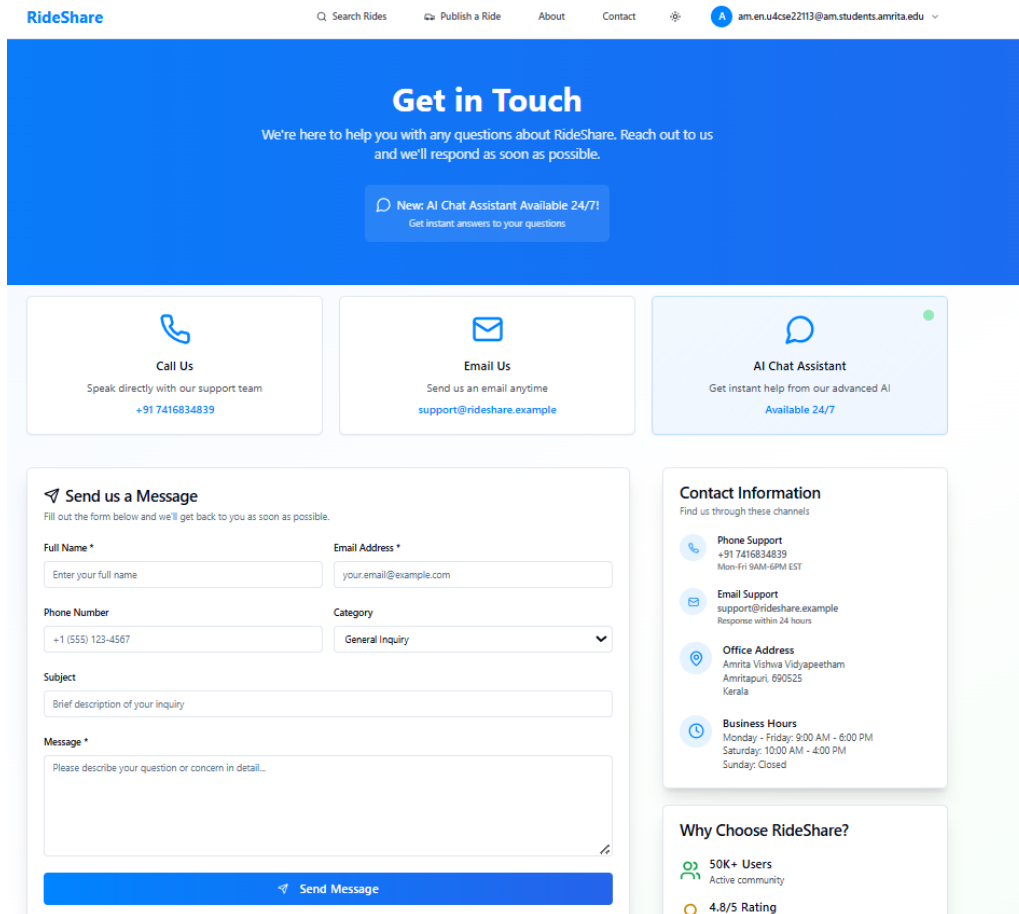
06:00 Kayamkulam 08:00 Kochi, Kerala ₹750

Stopovers: Anantapur, Andhra Pradesh

C Current User

UPI

Figure 4.18: Implementation Screenshot 18



The screenshot shows the 'Get in Touch' page of the RideShare application. The header includes the RideShare logo, search, publish, and contact links, and a user profile. The main section has a blue background with the heading 'Get in Touch' and a message: 'We're here to help you with any questions about RideShare. Reach out to us and we'll respond as soon as possible.' Below this is a button for the 'New AI Chat Assistant Available 24/7!'. The page is divided into three columns: 'Call Us' with a phone icon and number '+91 7416834839', 'Email Us' with an envelope icon and email 'support@rideshare.example', and 'AI Chat Assistant' with a chat bubble icon and 'Available 24/7'. A 'Send us a Message' form is on the left, and 'Contact Information' and 'Why Choose RideShare?' are on the right.

**Get in Touch**

We're here to help you with any questions about RideShare. Reach out to us and we'll respond as soon as possible.

**Call Us**  
Speak directly with our support team  
+91 7416834839

**Email Us**  
Send us an email anytime  
support@rideshare.example

**AI Chat Assistant**  
Get instant help from our advanced AI  
Available 24/7

**Send us a Message**  
Fill out the form below and we'll get back to you as soon as possible.

**Full Name \***  
Enter your full name

**Email Address \***  
your.email@example.com

**Phone Number**  
+1 (555) 123-4567

**Category**  
General Inquiry

**Subject**  
Brief description of your inquiry

**Message \***  
Please describe your question or concern in detail...

**Send Message**

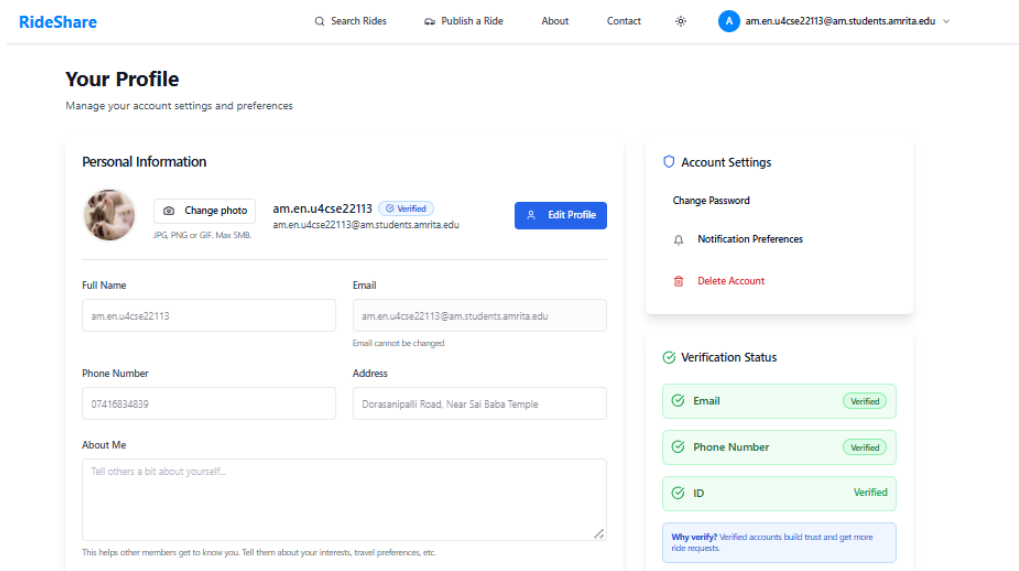
**Contact Information**  
Find us through these channels

- Phone Support**  
+91 7416834839  
Mon-Fri 9AM-6PM EST
- Email Support**  
support@rideshare.example  
Response within 24 hours
- Office Address**  
Amrita Vishwa Vidyapeetham  
Amritapuri, 690525  
Kerala
- Business Hours**  
Monday - Friday: 9:00 AM - 6:00 PM  
Saturday: 10:00 AM - 4:00 PM  
Sunday: Closed

**Why Choose RideShare?**

- 50K+ Users  
Active community
- 4.8/5 Rating


Figure 4.19: Implementation Screenshot 19



The screenshot shows the 'Your Profile' page of the RideShare application. The header is the same as Figure 4.19. The main section is titled 'Your Profile' with the subtitle 'Manage your account settings and preferences'. The page is divided into two columns: 'Personal Information' and 'Account Settings'. The 'Personal Information' column has a profile picture, a 'Change photo' button, and fields for 'Full Name', 'Email', 'Phone Number', and 'Address'. The 'Account Settings' column has buttons for 'Change Password', 'Notification Preferences', and 'Delete Account'. Below these is a 'Verification Status' section showing 'Email', 'Phone Number', and 'ID' as verified.

**Your Profile**  
Manage your account settings and preferences

**Personal Information**

 **Change photo** am.en.u4cse22113 **Verified** am.en.u4cse22113@am.students.amrita.edu **Edit Profile**

**Full Name**  
am.en.u4cse22113

**Email**  
am.en.u4cse22113@am.students.amrita.edu  
Email cannot be changed

**Phone Number**  
07416834839

**Address**  
Dorasaniipalli Road, Near Sai Baba Temple

**About Me**  
Tell others a bit about yourself...

**Account Settings**

- Change Password**
- Notification Preferences**
- Delete Account**

**Verification Status**

- Email** **Verified**
- Phone Number** **Verified**
- ID** **Verified**

**Why verify?** Verified accounts build trust and get more ride requests.

Figure 4.20: Implementation Screenshot 20

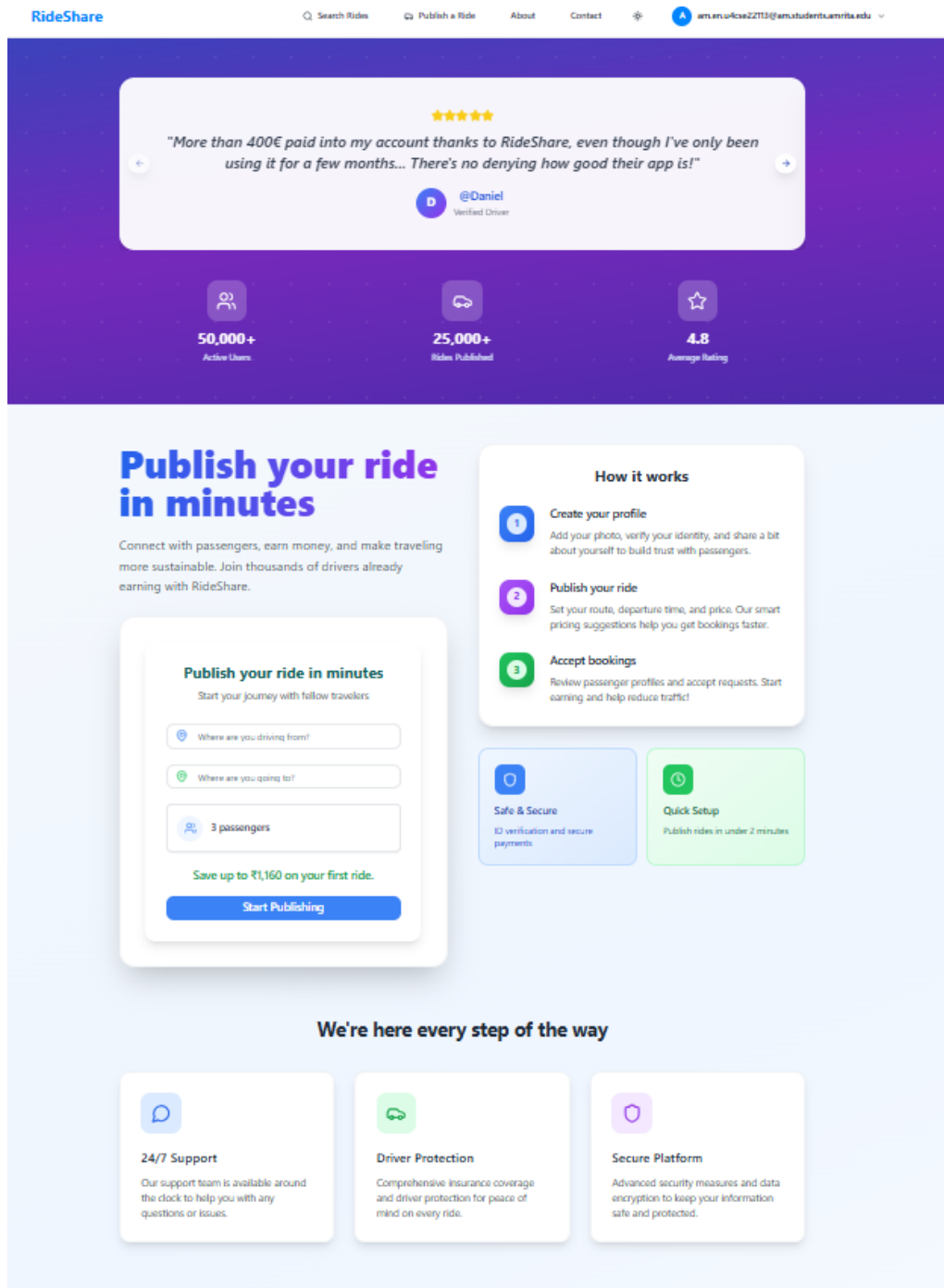


Figure 4.21: Implementation Screenshot 20

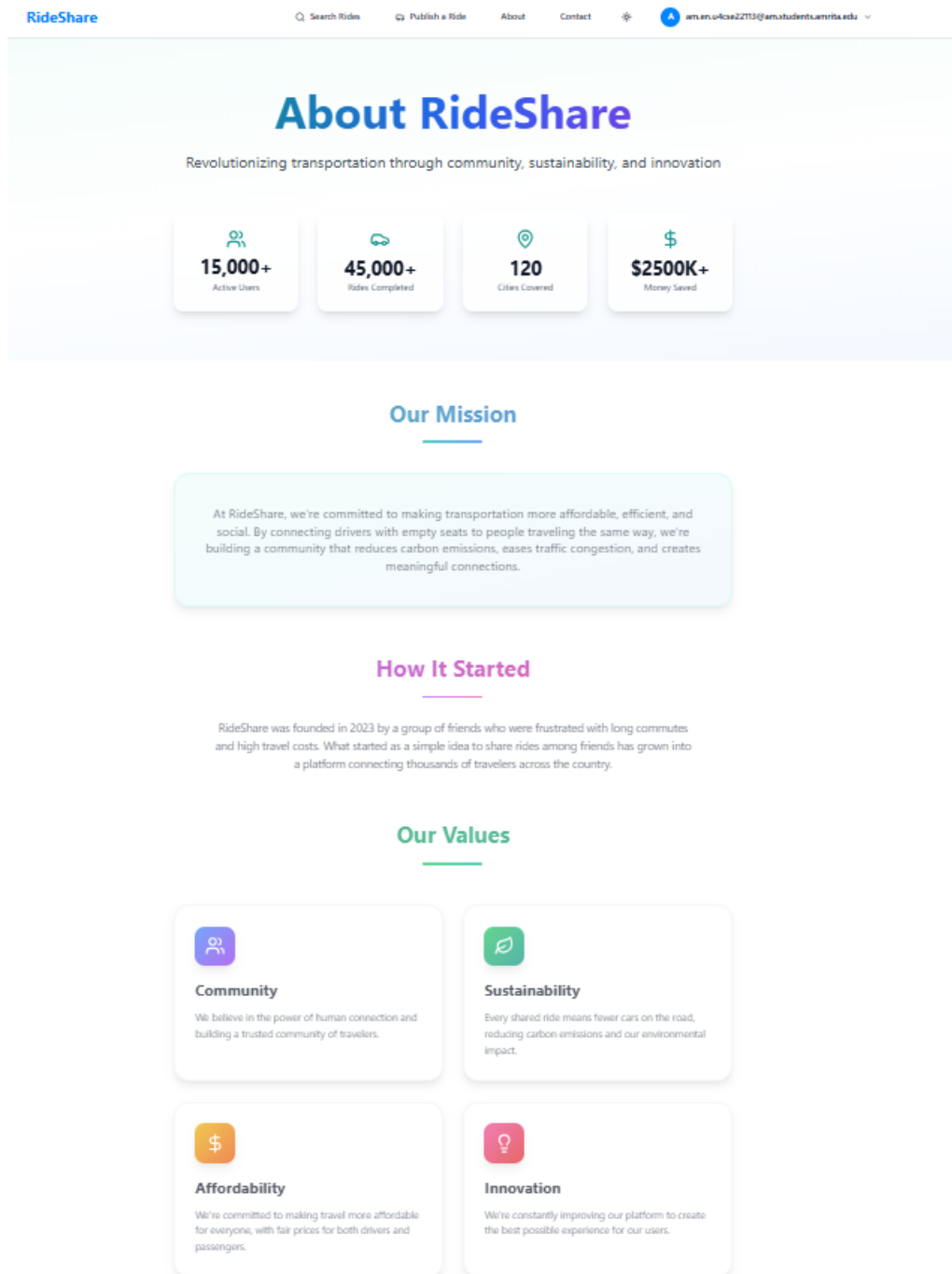


Figure 4.22: Implementation Screenshot 20

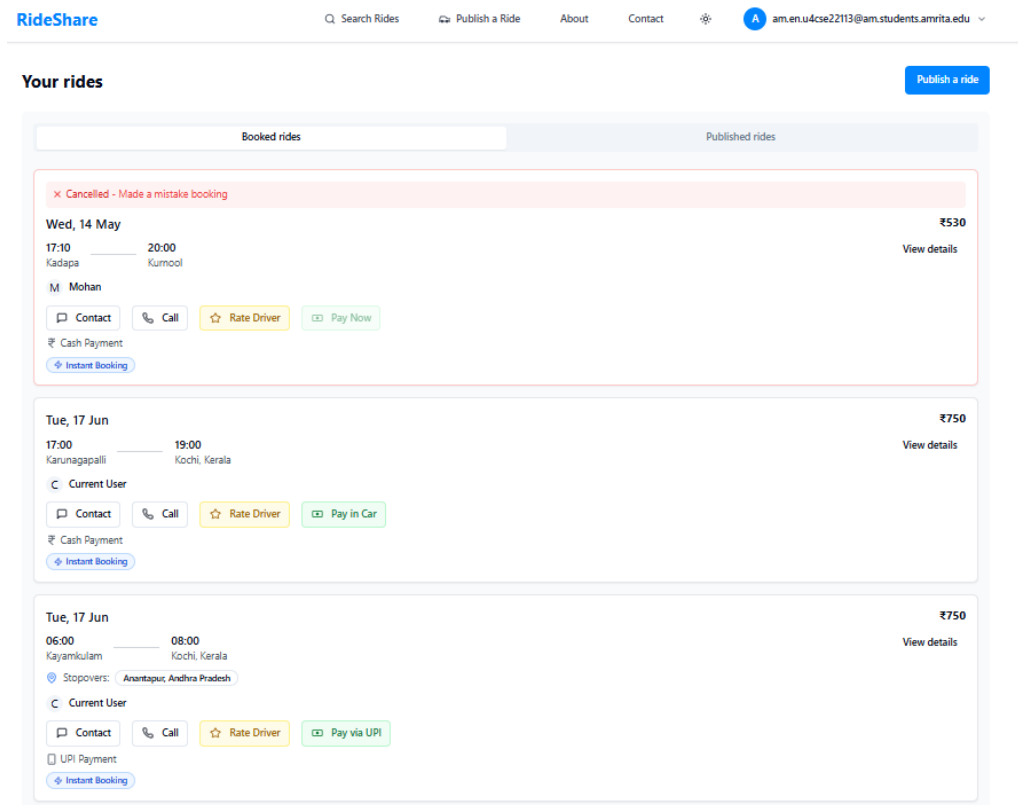


Figure 4.23: Implementation Screenshot 20

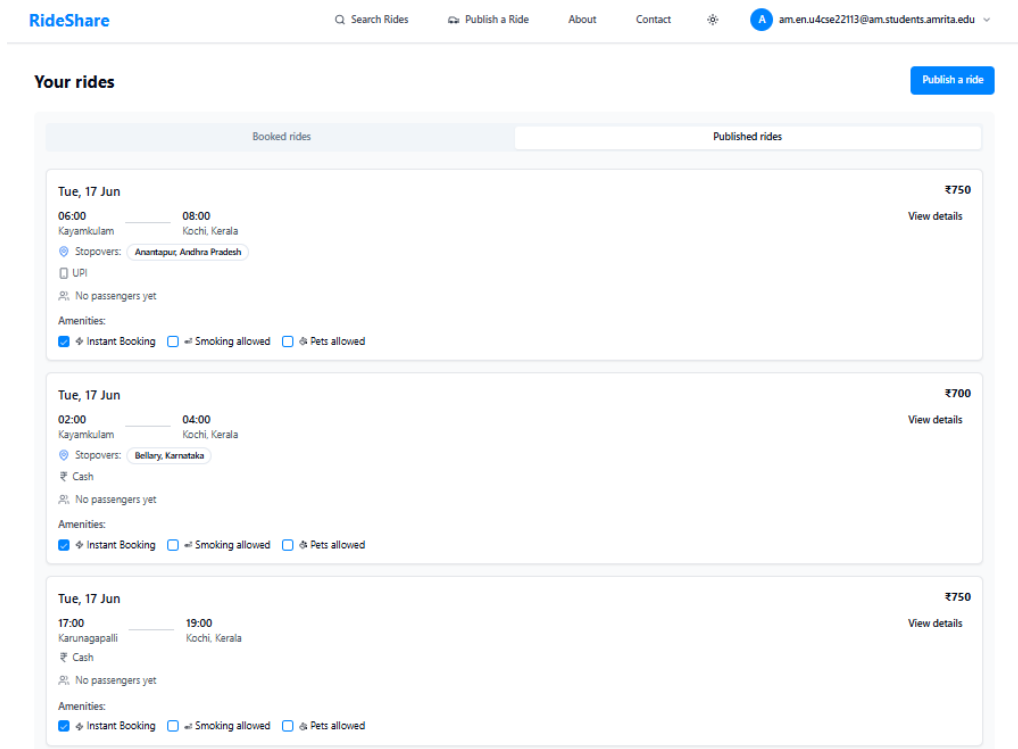


Figure 4.24: Implementation Screenshot 20



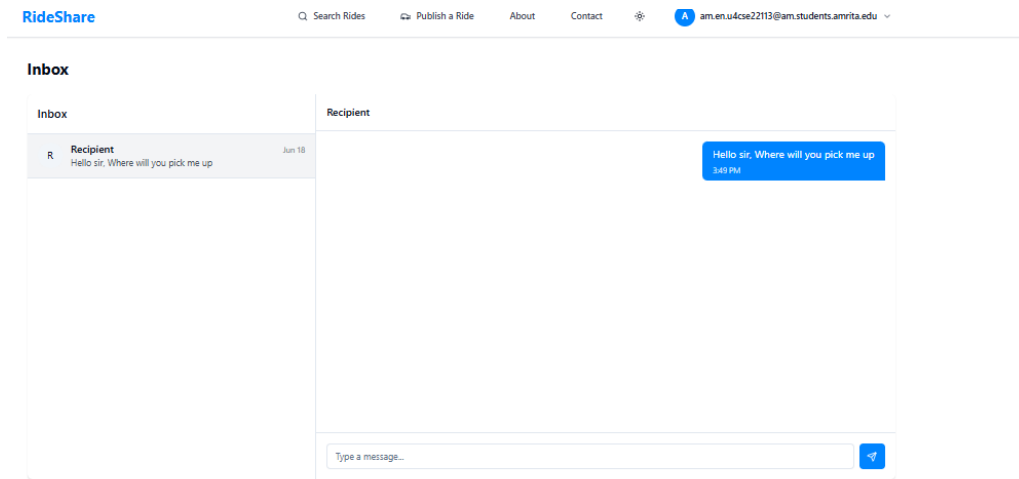


Figure 4.25: Implementation Screenshot 20

# CHAPTER 5

## TESTING

# TESTING

## 5.1 Testing

The testing phase played a vital role in ensuring the overall quality, stability, and reliability of the RideShare Connect system. Given the system's multiple interconnected modules — including user registration, authentication, ride publishing, ride searching, booking workflows, messaging, ratings, notifications, and payment handling (future scope) — it was essential to rigorously validate every module both individually and in combination. Our aim was to deliver a robust, seamless, and error-free platform that provides a smooth experience to both drivers and passengers under real-world operating conditions.

### Significance of Testing in RideShare Connect

- To ensure secure user registration, login authentication, and proper profile management functionality.
- To verify the correctness of ride creation, search, filtering, and booking operations.
- To confirm that booking requests, acceptances, cancellations, and ride completions are consistently handled across the system.
- To validate reliable delivery of driver-passenger communication through real-time messaging features.
- To accurately store, calculate, and display user ratings and feedback after ride completions.
- To guarantee that notification services such as OTP verification, email alerts, and booking confirmations are sent promptly and accurately.
- To ensure that any addition of new features does not disrupt existing system stability and functionality.

### *Comprehensive Testing Types Performed*

#### 1. Unit Testing

Each functional module such as user management, ride publishing, ride searching, booking confirmation, messaging, ratings, and notifications was tested in isolation. This phase helped us to identify early-stage defects, inconsistencies in business logic, and edge cases that may not have been captured during initial development.

## **2. Integration Testing**

After successfully passing unit tests, modules were integrated to verify full process flows — such as a user creating a ride, receiving booking requests, processing bookings, messaging passengers, completing rides, and submitting ratings. This ensured consistent data flow across all modules and accurate synchronization between interconnected subsystems.

## **3. Smoke Testing**

After every major code update or build, quick verification checks were performed on core functionalities like registration, login, ride publishing, ride searching, messaging, and notifications. This helped in early detection of significant flaws or system crashes after deployment of new changes.

## **4. Regression Testing**

As new features were progressively added — such as instant bookings, ride filtering, advanced booking cancellation policies, or ride preferences — previously implemented functionalities were re-tested systematically to verify they continued working as expected without being negatively impacted by the updates.

## **5. Black Box Testing**

From the end-user's perspective, real-world workflows were simulated by entering data without any knowledge of internal system logic. Use cases such as searching for rides, filtering results, booking rides, canceling bookings, contacting drivers, and providing ratings were tested extensively to ensure end-to-end usability, accuracy, and proper feedback delivery.

## **6. White Box Testing**

In this phase, internal code structures, algorithms, and logic flow were reviewed to validate data handling integrity, conditional checks, error management, authentication flows, database transactions, and backend consistency, especially during concurrent booking requests and user communication scenarios.

## **7. Performance Testing (Basic)**

Though large-scale production load simulation was outside this phase's scope, basic performance stress tests were conducted by simulating multiple concurrent users performing simultaneous bookings, messaging, and ride searches. This provided us with valuable insights into system responsiveness and scalability under moderate workload.

The comprehensive and multi-layered testing process played a key role in strengthening the system's performance, eliminating critical bugs, improving reliability, and delivering

a seamless user experience across diverse carpooling scenarios.

## 5.2 Conclusion

The development of the RideShare Connect platform was a highly enriching, insightful, and collaborative learning journey for our entire team. The central goal of the project was to build a highly functional, secure, and scalable web-based carpooling platform that seamlessly connects drivers and passengers while promoting ride-sharing, reducing traffic congestion, lowering fuel consumption, and contributing to environmental sustainability.

Throughout the entire development lifecycle, we systematically applied essential software engineering principles, encompassing requirement gathering, use case analysis, detailed system modeling using UML and DFD diagrams, software architecture design, iterative modular development, thorough multi-stage testing, and deployment planning. The creation of comprehensive design models like system architecture diagrams, data flow diagrams, class diagrams, and activity flows helped us structure our development approach clearly and logically.

Feature-wise, the platform successfully integrates secure OTP-based user registration and authentication, detailed ride publishing by drivers, advanced ride search and filtering by passengers, instant bookings, real-time messaging between drivers and passengers, booking status management, cancellation workflows, user ratings and reviews, and an automated notification system. All modules operate cohesively while ensuring data security, operational stability, and optimal usability for users.

Moreover, the systematic testing approach strengthened our ability to identify edge cases, correct logical inconsistencies, and ensure the system's integrity across a wide range of operational scenarios. This extensive practice also allowed us to enhance our understanding of backend integration, database management, error handling, and interface design under real-time conditions.

In conclusion, RideShare Connect demonstrates a fully functional, scalable, and practical carpooling solution that addresses modern commuting challenges while promoting sustainable transportation practices. Beyond achieving the technical project goals, the entire development cycle also enabled us to gain significant hands-on experience in collaborative team-based software development, real-world problem-solving, project coordination, and professional system design—preparing us for future software engineering challenges ahead.