RIDE ALONG

A Project Report

Submitted in partial fulfilment of the Requirements for the award of the Degree of

BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)
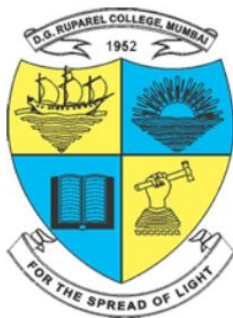
By

Ms. Tanvi Dasharath Naik

IT-1143

Under the esteemed guidance of

Mr. Mandar Bhave

Course Coordinator

DEPARTMENT OF INFORMATION TECHNOLOGY

D. G. RUPAREL COLLEGE OF ARTS, SCIENCE & COMMERCE

(Affiliated to University of Mumbai)

SENAPATI BAPAT MARG, MAHIM, MUMBAI, 400 016

MAHARASHTRA

2020-21

PROFORMA FOR THE APPROVAL PROJECT PROPOSAL

PNR No**.: 2019016401222987**   Roll no**: <u>IT-1143</u>**

1.   Name of the Student

        <u>Tanvi Dasharath Naik</u>

2.   Title of the Project

        <u>RideAlong</u>

3.   Name of the Guide

        <u>Mr. Mandar Bhave</u>

4.   Is this your first submission?   Yes ☑   No ☐

Signature of the Student

Date: …………………

D. G. RUPAREL COLLEGE OF ARTS, SCIENCE & COMMERCE

(Affiliated to University of Mumbai)

SENAPATI BAPAT MARG, MAHIM, MUMBAI, Maharashtra 400 016.

DEPARTMENT OF INFORMATION TECHNOLOGY

<u>CERTIFICATE</u>

This is to certify that the project entitled, **"<u>RideAlong</u>"**, is bonafied work of **<u>Ms. Tanvi Dasharath Naik</u>** bearing Seat.No: **<u>IT-1143</u>** submitted in partial fulfillment of the requirements for the award of degree of BACHELOR OF SCIENCE in INFORMATION TECHNOLOGY from University of Mumbai.

Place: Mumbai

Date:

Mr. Mandar Bhave.

(Course Coordinator)

ABSTRACT

Carpooling also commonly known as car-sharing, ride-sharing and lift sharing, is the sharing of car journeys so that more than one person can travel in a car. With the enormous increase in number of vehicles on road, people around the country especially in metro cities have started facing problem now due to increase in traffic which added an hour or so to their daily travelling time. Carpooling is seen as a more environmentally friendly and sustainable way to travel as sharing journeys reduces carbon emissions, traffic congestion on the roads, and the need for parking spaces. This method is very useful as it has great value and great use in normal life. Car sharing aims at solving this problem by targeting the empty seats in the private cars.

Sometimes higher authorities encourages people to use carpool during the period whenever there is a hike in the prices of petroleum products or when the pollution level of that state is going beyond the maximum limits. The main objective of this carpooling website is to enable different persons living in an area to use this system to minimize monthly expenses and no tension of hiring a car at higher cost. This will enable people using this website to share expense, not worry about hiring a cab and making new connections. This project is intended to better utilize the vacancy seat in the passengers cars.

ACKNOWLEDGEMENT

It gives me great pleasure to present this project report on "**RideAlong**". It's a great pleasure and moment of immense satisfaction for me to express my profound gratitude to my Project Guide, **Prof. Mandar Bhave** and **Prof. Manasi Rajapurkar** whose constant encouragement enabled me to work enthusiastically. His perpetual motivation, patience and excellent expertise in discussion during the progress of the project report work have benefited me to an extent, which is beyond expression. I am highly indebted to his invaluable guidance and ever-ready support in the successful completion of this project report in time. Working under his guidance has been a fruitful and unforgettable experience.

I would like to thank all people who contributed to the successful completion of the project. I would also like to thank the entire Information Technology department of **D. G. Ruparel College of Arts, Science & Commerce** and all Library staff for their co-operation, which helped me to complete this project report.

Last but not least I would also wish to thank all teaching and non-teaching staff and my friends; it is really impossible to repay the debt of all people who have directly or indirectly helped me for performing the project. To the people who have helped me with their valuable suggestions and guidance has been very helpful in various phases of the completion of the project.

Tanvi Dasharath Naik

DECLARATION

I hereby declare that the project entitled, "**RideAlong**" done at **D.G. Ruparel College of Arts, Science & Commerce**, has not been in any case duplicated to submit to any other university for the award of any degree. To the best of my knowledge other than me, no one has submitted to any other university.

The project is done in partial fulfilment of the requirements for the award of degree of **BACHELOR OF SCIENCE (INFORMATION TECHNOLOGY)** to be submitted as final semester project as part of our curriculum.

Tanvi Dasharath Naik

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1
Introduction

1.1 Background

Transportation is one of major issue which is affecting day to day life of people. Roadways traffic is one of the most commonly used network for moving from one place to another place. In road traffic the majority of road transport consists of the single passenger car. These private cars are mostly used with only a single ride for single person.

With the increase of environmental concerns and the congestion of roads, carpooling has gained a lot of popularity when it comes to environment-friendly and cheap ways of travelling. Carpooling is when two or more persons share a ride in one of their personal cars. Carpooling reduces pollution since we have fewer cars on the road. It's also economic since the travel expenses are shared among the riders. Travelling alone may be stressful, so having other persons with you on a trip reduces the stress and is also the occasion to socialize and make the trip funnier.

Finding people to share a ride with is the challenge of carpooling as it is difficult to find a person going to the same place as you at a given time. Many websites and applications have been developed to help people meet to share rides. Those applications enable users to create and share their trips and find passengers. The downside of those applications is most of them do not have a feature for direct communication between the passenger and rider through the website. The purpose of this project is to develop an application that tries to overcome the disadvantages of the other available applications.

1.2 Objectives

The project aims to reduce overall traffic congestion by reducing the number of vehicles and increasing the share mode of transport for people. RideAlong will allow a secure and safe ride for a user in a user-friendly online system. It will also help to manage the details of the users, vehicles, payment of rides, booking, etc. which will help in reducing the manual work for recording this information.

Benefits of the carpooling system:

- The system is fully functional and flexible.

- Easy to use, time and money saver system.

- Eco-friendly system.

- 24/7 open system that increases the quality of service to the user in less span of time.

1.3 Purpose, Scope, & Applicability

1.3.1 Purpose

The goal of this project is to reduce overall traffic congestion by reducing the number of vehicles on the road and increasing the percentage of people who use public transportation. The following are some of the project's advantages:

- Reducing single-occupancy car trips by implementing a carpooling system.

- Reduce peak hour congestion.

- Reducing overall traffic congestion on the roads

- Save money by sharing the cost of driving one car.

- Reduce the number of cars on the road.

- Reduce pollution and carbon dioxide emissions.

- Reduces driving-related stress for participants

- Provide social connections in society.

1.3.2 Scope

Scope of the project is the part of the project planning process that determines and documents what are the project deliverables, goals, tasks, costs and deadlines. It is what needs to be achieved and the work that must be done to deliver a project.

The reasons people choose to carpool include saving money on commuting by sharing the cost of fuel, helping to reduce traffic congestion during peak travel hours, and contributing to lower levels of air pollution. But in many cases, people don't carpool because of the difficulty of finding someone else with the same location and schedule.

RideAlong can help people to find other people to share the ride through the website which will encourage more people to choose carpool. This website will allow the rider to communicate with the passenger through chat. Paying for the ride can also be done through the website.

1.3.3 Applicability

Traffic is not just a nuisance for drivers: It's also a public health hazard and bad news for the economy. Transportation studies put the annual cost of congestion at $160 billion, which includes 7 billion hours of time lost to sitting in traffic and an extra 3 billion gallons of fuel burned.

One way to improve traffic is through ride-sharing — and a new MIT study suggests that using carpooling options could reduce the number of vehicles on the road by a factor of three without significantly impacting travel time.

Benefits for the drivers:
• The driver who will already make that trip on that day, will reduce his/her travel costs.
• For the ones that do not like traveling alone, will have the chance to find a travel-mate.
• Shared driving carpooling can also reduce driving stress.

Benefits for the passengers:

• Passengers will have the chance to travel at lower costs than train or bus.

• They will make their trip with the comfort of an automobile.

• Avoiding lonely trips also applies to the passenger.

Benefits to the environment and economy:

• Carpooling was encouraged to save oil. In reducing the number of cars on the road, carpooling decreases pollution and the need for parking space, and from a global perspective, reduces greenhouse gas emissions.

• When there are fewer cars on roads, the traffic jam in especially large cities will decrease.

1.4   Achievements

I learnt a lot about the technologies I utilised for this project, such as JavaScript, Node, and others, while working on it. I was able to put what I had learned in class into practice. Making this project has aided in the development of my abilities and the acquisition of new knowledge. It has also boosted my self-assurance and experience.

I demonstrated my abilities to come up with a solid solution to develop an application in this assignment. This project also served as a showcase for the degree of effort I put into implementing it and then testing it for any flaws. The end result was good in that I was able to become familiar with and then apply new technologies such as Nodejs, Reactjs, and others effectively.

Chapter 2
Survey of Technologies

2.1   Existing System

This system includes sharing our ride with another person who is going to the same location. The driver and the passenger need to meet and discuss the ride timings and locations. They have to keep a record of the ride and payment.

Disadvantages of Existing System:

- Finding a person going to the same location at the same time is difficult.
- Maintaining a record of ride-shares is time-consuming.
- It is limited to a certain area only.
- There are no valid records of payment.

2.2   Proposed System

RideAlong website will help to overcome the above issues by providing a user friendly interface. A platform as a link between supply and demand is provided creating a new mobility service. The ride fare is fixed by the car owner within the range of kilometers traveled and will even take care of all administrative issues. User can find or offer a ride without meeting in person which will save the time of both- driver and passenger. This system will help to reduce the paper-work required to store the records. Payment gateway will assure the proper payment of the ride.

2.3 Justification of Platform

2.3.1 JavaScript

This project is made using JavaScript Language. One of the most popular scripting languages, JavaScript is used in all the web applications for validation, rendering dynamic content, interactive graphics and maps, and much more. Along with HTML and CSS, JS has the power to build complete, robust web applications. Because of JS, the user can interact with a web page and see all the interesting elements on the page.

Following are the important features of JS:

- Used on both client-side and server-side to create interactive web content.
- Greatly improves user experience by providing dynamic functionality.
- Light-weight language having object-oriented capabilities.
- Interpreted, open and cross-platform language.
- Seamless integration with Java and HTML.

2.3.2 React.js

React is a free and open-source front-end JavaScript library for building user interfaces or UI components. React makes it painless to create interactive UIs. Design simple views for each state in the application and React will efficiently update and render just the right components when your data changes. React.js can be used to create user interfaces in JavaScript for different platforms. We can use ReactDOM for web applications, React Native for mobile app development (sharing the majority of code between Android and iOS), and cross-platform hybrid desktop applications with Electron. Recently, Microsoft has also released React Native for Windows. Compared to other frontend frameworks, the React code is easier to maintain and is flexible due to its

modular structure. This flexibility, in turn, saves huge amount of time and cost to businesses.

2.3.3 Node.js

Node.js is an open source server environment. It allows us to run JavaScript on the server.

Advantages of Node.js:

- **The ability to scale up quickly**- Each of the nodes in Node.js is based around an "event." The amount of nodes you can add to your core programming function are nearly limitless. This means you can scale vertically, adding new capability paths that lead back to your core application code.
- **Speed and Performance** - Its non-blocking, input-output operations make the environment one of the speediest options available. Code runs quickly, and that enhances the entire run-time environment.
- **Flexibility** - When you make a change in Node.js, only that node is affected. Where other run time environments or frameworks may require you to make changes all the way back to the core programming, it doesn't require anything more than a change to the node.

2.3.4 Express.js

Express is a fast, assertive, essential and moderate web framework of Node.js. We can assume express as a layer built on the top of the Node.js that helps manage a server and routes. It provides a robust set of features to develop web and mobile applications. The core features of Express framework:

- It can be used to design single-page, multi-page and hybrid web applications.
- It allows to setup middlewares to respond to HTTP Requests.

- It defines a routing table which is used to perform different actions based on HTTP method and URL.
- It allows to dynamically render HTML Pages based on passing arguments to templates.

2.3.5 MongoDB

MongoDB is a document-oriented database software that is open source and cross-platform. MongoDB is a NoSQL database software that uses JSON-like documents with schemas that may be customised. MongoDB is a database developed by MongoDB Inc. and distributed under the terms of the Server Side Public License (SSPL). At a high level, MongoDB enables developers that use data (which is most of us) to build easily, adapt quickly, and scale reliably.

- **Flexible Document Schemas**: MongoDB's document model allows virtually any kind of data structure to be modeled and manipulated easily. MongoDB's BSON data format, inspired by JSON, allows you to have objects in one collection have different sets of fields

- **Code-native data access**: Most databases force you to use heavy wrappers, like ORMs (Object Relational Mappers), to get data into Object form for use in programs. MongoDB's decision to store and represent data in a document format means that you can access it from any language, in data structures that are native to that language.

- **Change-friendly design**: We spend a lot of time and effort designing efficient processes, and learning from our mistakes, but typically the database is slowing the

whole thing down. There's no downtime required to change schemas, and you can start writing new data to MongoDB at any time, without disrupting its operations.

- **Easy horizontal scale-out**: MongoDB is designed from the ground up to be a distributed database. Create clusters with real-time replication, and shard large or high-throughput collections across multiple clusters to sustain performance and scaler horizontally.

Chapter 3
Requirements and Analysis

3.1   Problem definition

As we live in a modern society, individuals prefer to use their own car on the road for every minor task. The number of cars on the road has increased significantly in recent years, causing traffic congestion on the roadways. As the number of cars on the road grows, it causes dozens of new issues, including an increase in pollution levels in the surrounding area. Because fuel costs are a nonrenewable supply of energy, the more the number of cars on the road, the higher the fuel consumption, which leads to an increase in gasoline prices. Because pollution from these cars has negative impacts on people's health, a car sharing system is a viable option, but it comes with its own set of problems and issues, such as security and trust. A web-based carpool system that allows a new user to guarantee the security and safety of the journey can address the problem. Short- and long-distance travels must be included in the Carpooling System, whether they are within the city or outside of the city of the user's intended location.

3.2   Requirement Specification

Requirement analysis is significant and essential activity after elicitation. We analyze, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements. Requirements analysis encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product or project, taking account of the possibly conflicting requirements of the various stakeholders, analyzing, documenting, validating and managing software or system requirements. The following section contains the user and system requirements for the carpooling website.

The website is a meeting point for carpoolers, both drivers and passengers. Users can share and find rides. The access of the application is only granted to authorized users.

- **Functional Requirements:**

  - ➢ **Users:** The users of the application are travelers and commuters who want to go from one place to another or users that are driving a trip and want to find passengers. Users can act as both passengers and drivers while using an website. The users can use their social media accounts in order to log in the application. Any user of the application can act as:

    - ▪ **Rider (Driver):** A driver is any person that owns a car and wants to go from one place to another and publishes his trip on the website in order to find passengers to share the ride with.
    - ▪ **Passenger:** A passenger is any person that doesn't own a car and wants to join a driver in a trip he posted and agrees to all the conditions specified (price and general behavior).

  - ➢ **Login:** Since all the operations that can be done using the application requires both the driver and passenger to be logged in, they can use the login forms of either Google or Facebook. For this matter, the user is prompted to connect the website to his account and then proceed for sign in/up. If he has never logged to the application before, a new account is created for him.

  - ➢ **Rate driver/passenger:** Both the driver and passenger can rate each other in other to gain a reputation. The importance of the rating is to encourage users to be helpful and nice during the trip so that they gain popularity in the application. It is also a way to ensure users of who can be trusted or not. The ratings represent a relative guarantee for the users to trust each other.

- ➢ **Chat:** Both the users: rider and passenger can interact and fix the meeting schedule through chat. Deciding the payment amount can also be done in the chat.

- ➢ **Map:** After the driver user selects the source and destination location it will be shown in the map which can be viewed by the passenger user as well as the driver.

- • **Non-Functional Requirements:**

  - ➢ **Performance:** The application has to offer a very quick response time as the meeting between the driver and passengers is done through the website. The application should handle 1000 users sending queries at the same time.

  - ➢ **Scalability:** The application should respond properly to a high increase of users. It should be able to handle from 1,000 users to 10,000 users.

  - ➢ **Privacy and Security:** The application should ensure the privacy of the users including the trips they take part in and their accounts. The login system should also be robust where only authorized users can post and edit their own information.

  - ➢ **Maintainability:** Since the application may be developed in the future by adding other features, it should be easily maintainable.

3.3   Planning and Scheduling

3.3.1 Gantt Chart

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity. When you set up a Gantt chart, you need to think through all of the tasks involved in your project. As part of this process, you'll work out who will be responsible for each task, how long each task will take, and what problems your team may encounter. It enables us to see at a glance:

- What the various activities are
- When each activity begins and finishes
- How long each activity is scheduled to run
- Where activities overlap with other activities, and by how much
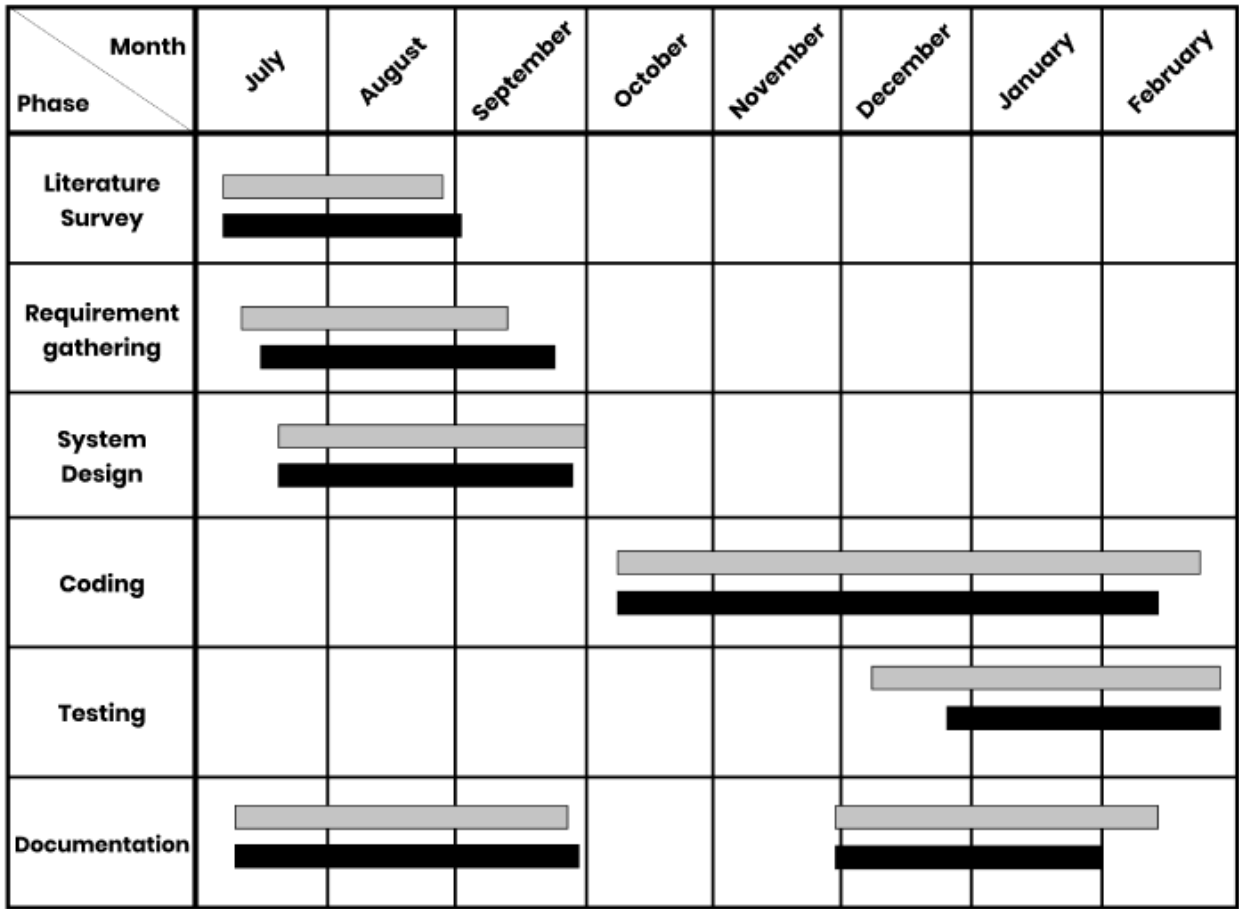- The project's start and completion dates.

The vertical axis of a Gantt chart shows the tasks that need to be completed, while the horizontal axis represents the project timeline. Most Gantt diagrams are created in Excel or with project management software, which is sometimes referred to as Gantt chart software.

In other words, a gantt chart is a super-simple way to communicate what it will take to deliver a project on time and budget. That means it's a whole lot easier to keep your project team and stakeholders on the same page from the get-go.

Reading a gantt chart really comes down to understanding how the different elements come together to make a gantt chart work.

Elements of a gantt chart:

- **Task list**: Runs vertically down the left of the gantt chart to describe project work and may be organized into groups and subgroups

- **Timeline**: Runs horizontally across the top of the gantt chart and shows months, weeks, days, and years

- **Dateline**: A vertical line that highlights the current date on the gantt chart

- **Bars**: Horizontal markers on the right side of the gantt chart that represent tasks and show progress, duration, and start and end dates

- **Milestones**: Yellow diamonds that call out major events, dates, decisions, and deliverables

- **Dependencies**: Light gray lines that connect tasks that need to happen in a certain order

- **Progress**: Shows how far along work is and may be indicated by percent complete and/or bar shading

- **Resource assigned**: Indicates the person or team responsible for completing a task

Month / Phase	July	August	September	October	November	December	January	February
Literature Survey	▓▓▓▓	▓▓						
	████	████						
Requirement gathering	▓▓▓▓	▓▓▓▓						
	████	████	████					
System Design		▓▓▓▓	▓▓▓▓					
		████	████					
Coding				▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓▓▓	▓▓
				████	████	████	████	
Testing						▓▓	▓▓▓▓	▓▓
						██	████	████
Documentation	▓▓	▓▓▓▓	▓▓			▓▓	▓▓▓▓	
	████	████	████			████	████	

Estimated Time ▓▓▓▓▓▓▓▓▓▓

Actual Time ██████████

Figure 1: Gantt Chart

3.4   Software and Hardware requirements

3.4.1 Hardware Requirements

To run this software smoothly there are some hardware requirements which are as follows:

- Processor: Intel Core 2 duo processor or higher
- RAM: Minimum 4 GB
- HARD DISK: 500 GB

3.4.2   Software Requirements

- Operating System: Windows 7 or above
- Technologies: HTML, CSS, JavaScript, React, Node, MongoDB
- Browsers: Google Chrome, Mozilla Firefox, Microsoft Edge, etc
- Editor: VSCode

3.5 Preliminary Product Description

3.5.1 Product Perspective

Carpooling, also known as ride-sharing or lift-sharing, implies sharing a vehicle and it gathers in the same vehicle, users that want to do the same trip and are willing to share their vehicles and costs. This method makes use of private vehicles and splits the travel and maintenance costs among all its passengers. Its goal is to increase vehicle sharing among several users and consequently reduce the number of vehicles that only has one person in it. Some groups of people search and adopt this solution in their daily life, usually it is a closed group from a specific workplace, or a group of students from a university campus or even a mix of the two. Several groups of people use carpooling to make the not so common trip to another city every now and then. It is very easy to find people to carpool with when going, for example, from Mumbai to Delhi and vice-versa in the social networks or other carpooling website.

3.5.2 Product Function

- **Rider**

  The rider is the one who drives his own vehicle during the journey. This person uploads the ride information and decides the fee.

- **Passenger**

  The passenger is the one who selects a suitable ride from the list of rides offered by other riders. They may chat with the rider and pay for the journey.

- **Admin**

  The admin is the one who will add location related data to the database

3.6   SDLC Model

- **Incremental Development Model**

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

Advantage of Incremental Model

- Errors are easy to be recognized.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it handled during its iteration.
- The Client gets important functionality early.

The various phases of incremental model are as follows:

1. **Requirement analysis**: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

2. **Design & Development:** In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

3. **Testing**: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behavior of each task.

4. **Implementation**: Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

Thus we have many models with which we can develop software and achieve the required objective. We have seen about once such model called as Incremental Model with it's characteristics, application and advantages. The main application whether we use this model is where we have clear understanding of requirement and 100% objective of the software is expected.

STAGED / INCREMENTAL MODEL OF SDLC
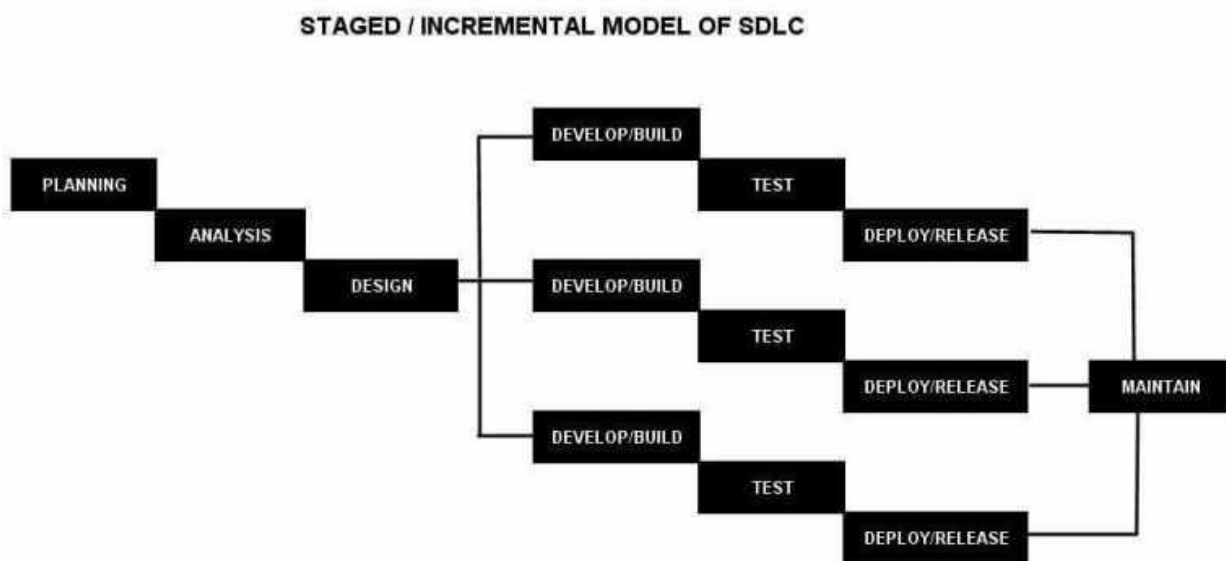
Figure 2: Incremental Development Model

Chapter 4
System Design

4.1   Basic Modules

- **SignUp/Login:**

  In this module the user is requested to register by giving all of the needed information. If the user already has an account, they can use it to log in.

- **Rider:**

  This module include users who are willing to share a ride in their own car. The Rider user has to post the ride details including the locations, timings, number of seats, etc.

- **Passenger:**

  This module includes users who are willing to share a ride in someone else's car. They may look over rides listed by other riders and select one that is suitable for them.

- **Chat:**

  This is the module where rider and passengers can communicate through chat.

- **Payment:**

  In this module, the passenger user can pay the decided amount to the rider user.

- **Feedback:**

  After the ride is complete, the passenger and rider can give feedbacks to each other in this module.

- **Profile:**

  This is the profile of the user which will display all the details about the user. The history of rides and payments will be displayed in this module.

- **Admin:**

  Admin can verify users from their documents and add city details like city name, latitude, longitude.

4.2 Data Design

4.2.1 Schema Design

- **User Table**

Column Name	Data Type	Constraints
_id	ObjectId	Primary Key
name	String	Not Null
email	String	Not Null
Username	String	Not Null
Gender	String	
password	String	Not Null
Encry_password	String	
Salt	String	
vehicle	ObjectId	Foreign Key

rides	Array	
payments	Array	
feedbacks	Array	
Profile_pic	String	
Document	String	Not Null
contactNumber	Number	Not Null
verificationStatus	Boolean	

Table 1: User Schema

- **Ride Table**

Column Name	Data Type	Constraints
_id	ObjectId	Primary Key
driverUser	ObjectId	Foreign Key
Passengers	Array	
sourceLocation	Array	
destinationLocation	Array	
Requests	Array	
vehicle	ObjectId	Foreign Key
seats	Number	
startTime	Date	
fare	Number	

Paymenrs	Array	

Table 2: Ride Schema

- **Vehicle Table**

Column Name	Data Type	Constraints
_id	ObjectId	Primary Key
Owner	ObjectId	Foreign Key
namePlate	String	
model	String	
numberOfSeats	Number	
License	String	
vehicleInsurance	String	
vehicleRC	String	

Table 3: Vehicle Schema

- **City Table**

Column Name	Data Type	Constraints
_id	ObjectId	Primary Key
name	String	
Latitude	Number	

longitude	Number	

Table 4: City Schema

- **Invoice Table**

Column Name	Data Type	Constraints
_id	ObjectId	Primary Key
sender	ObjectId	Foreign Key
receiver	ObjectId	Foreign Key
ride	ObjectId	Foreign Key
invoiceAmount	Number	

Table 5: Invoice Schema

- **Feedback Table**

Column Name	Data Type	Constraints
_id	ObjectId	Primary Key
Feedbacker	ObjectId	Foreign Key
receiver	ObjectId	Foreign Key
feedbackText	String	
Rating	Number	

Table 6: Feedback Table

- **Conversation Table**

Column Name	Data Type	Constraints
_id	ObjectId	Primary Key
Members	Array	

Table 7: Conversation Table

- **Message Table**

Column Name	Data Type	Constraints
_id	ObjectId	Primary Key
conversationId	ObjectId	Foreign Key
Sender	ObjectId	Foreign Key
Content	String	

Table 8: Message Table

4.3   ER Diagram

ER Diagram stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database. In other words, ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships. ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships. The purpose of ER Diagram is to represent the entity framework infrastructure.

Components of ER Diagram:

- **Entity, Entity Type, Entity Set :** An Entity is an object of Entity Type and set of all entities is called as entity set.

- **Attribute(s):** Attributes are the properties which define the entity type.

- **Composite Attribute:** An attribute composed of many other attribute is called as composite attribute.

- **Multivalued Attribute:** An attribute consisting more than one value for a given entity.

- **Derived Attribute:** An attribute which can be derived from other attributes of the entity type is known as derived attribute.

The number of times an entity of an entity set participates in a relationship set is known as cardinality. Cardinality can be of different types:

- One-to-One
- Many-to-Many
- Many-to-One
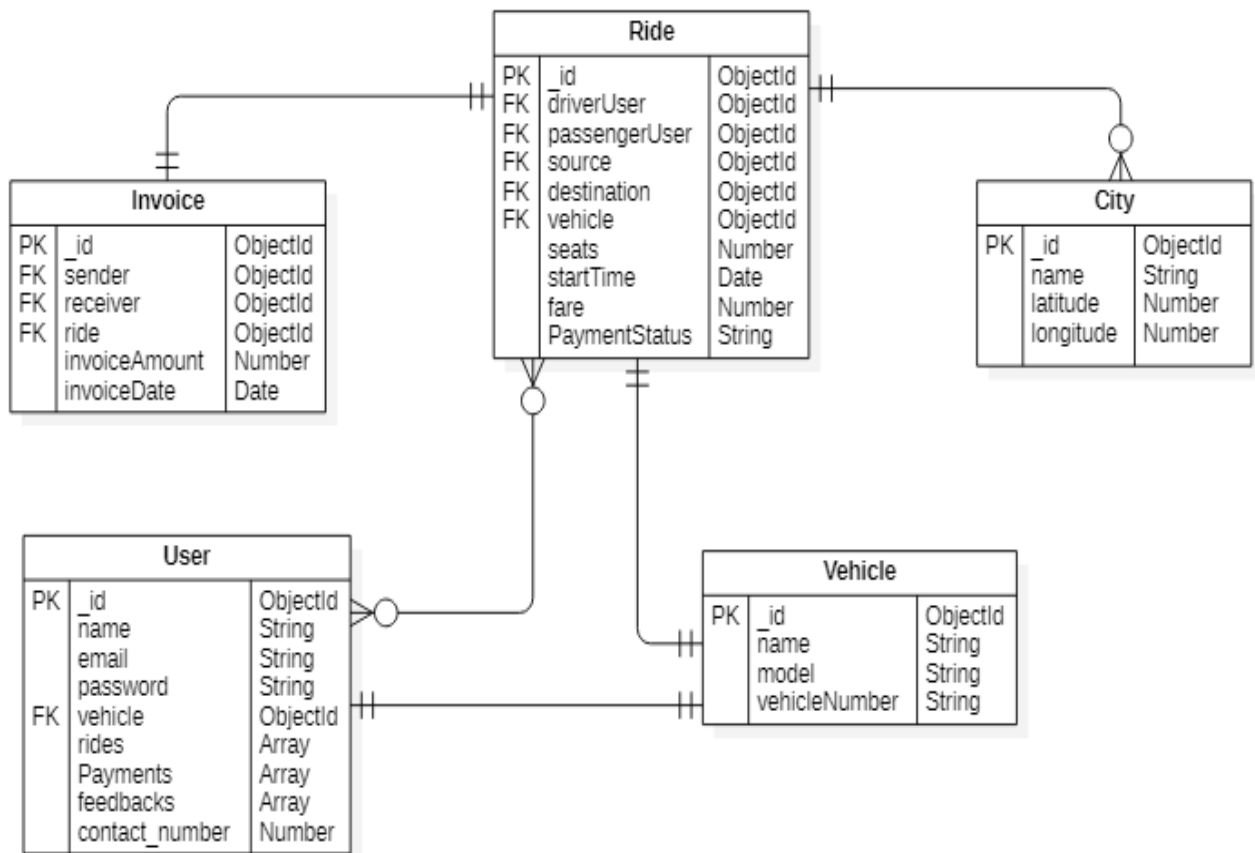
Figure 3: ER Diagram

4.4   UML

4.4.1 Usecase Diagram

A **Use Case Diagram** is a form of behavioural diagram specified by and generated from a Use-case analysis in the Unified Modeling Language (UML). Its goal is to offer a graphical representation of a system's functioning in terms of actors. A use case diagram in UML is used to show the various ways in which a user may interact with a system.

Some of the symbols used in Usecase Diagrams are:

- **Use cases:** Horizontally shaped ovals that represent the different uses that a user might have.

- **Actors:** Stick figures that represent the people actually employing the use cases.

- **Associations:** A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.

- **System boundary boxes:** A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho Killer is outside the scope of occupations in the chainsaw example found below.

- **Packages:** A UML shape that allows you to put different elements into groups. Just as with component diagrams, these groupings are represented as file folders.
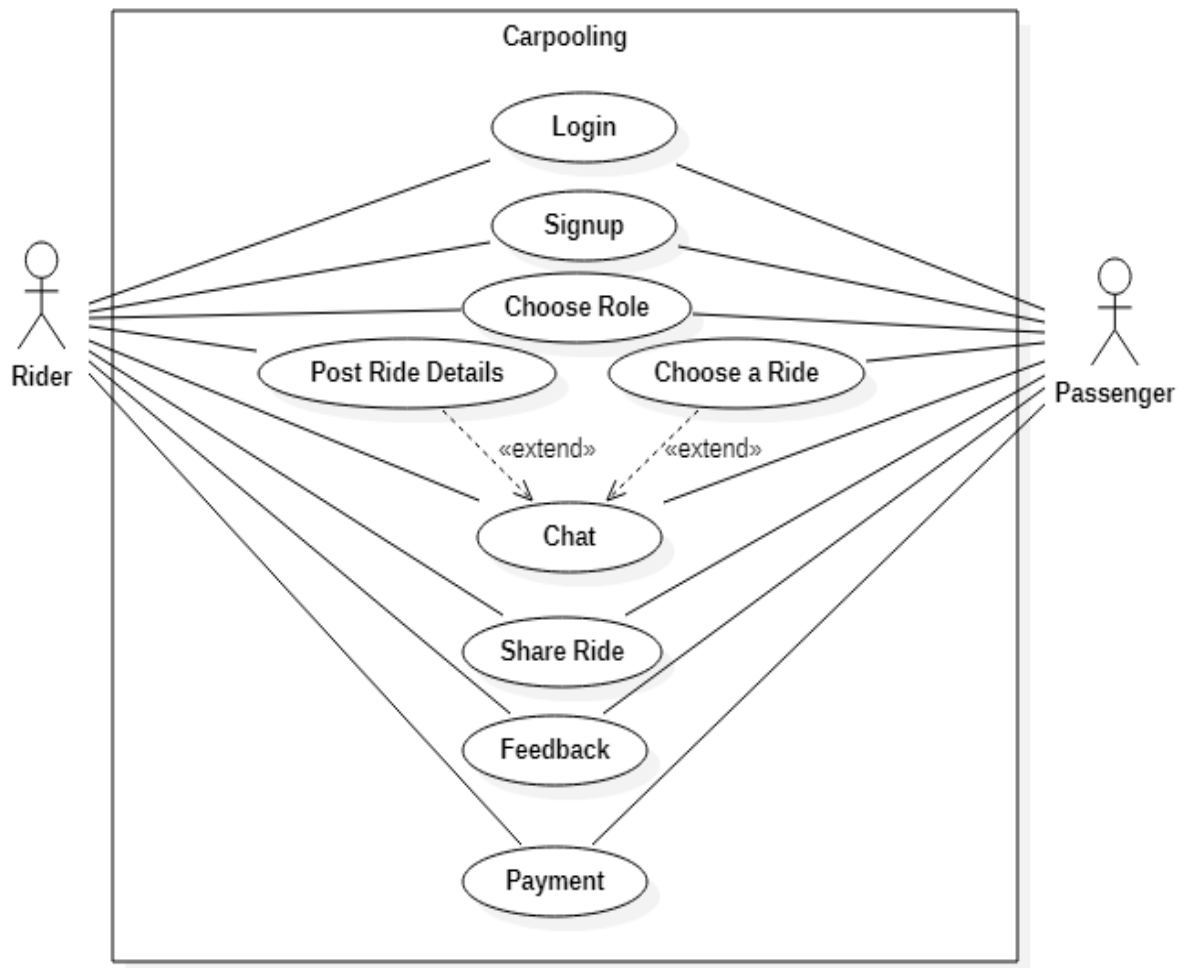
Figure 4: Usecase Diagram

This use case diagram is a visual representation of the process required for carpooling. Riders and passengers are the two types of users. Each user's various usecases are depicted in the figure.

4.4.2 Activity Diagram

Activity Diagram is an important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

Some of the most common components of an activity diagram include:

- **Action:** A step in the activity wherein the users or software perform a given task. In Lucidchart, actions are symbolized with round-edged rectangles.

- **Decision node:** A conditional branch in the flow that is represented by a diamond. It includes a single input and two or more outputs.

- **Control flows:** Another name for the connectors that show the flow between steps in the diagram.

- **Start node:** Symbolizes the beginning of the activity. The start node is represented by a black circle.

- **End node:** Represents the final step in the activity. The end node is represented by an outlined black circle.
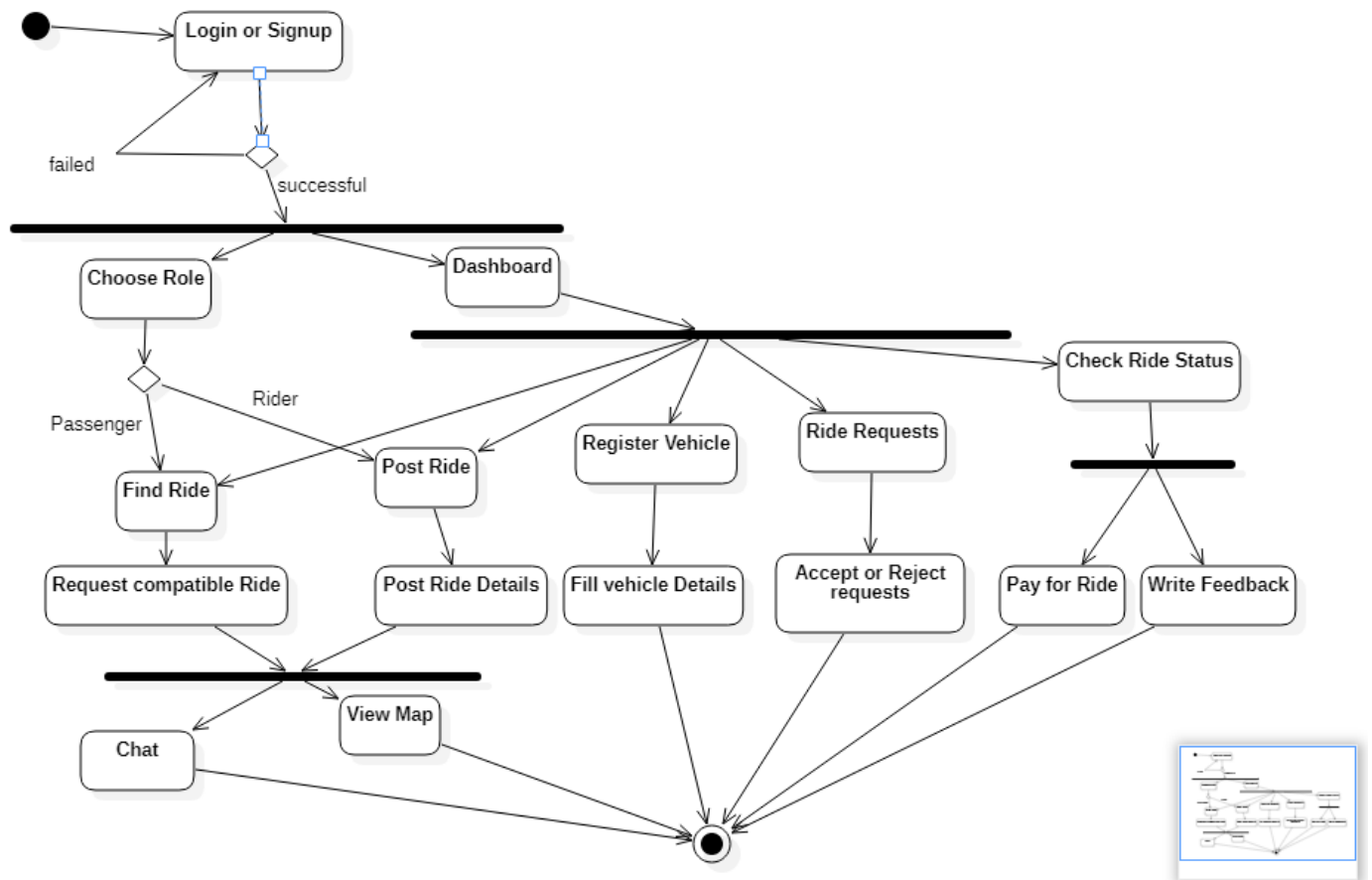
Figure 5: Activity Diagram

This is the RideAlong activity UML diagram, which depicts the information flow between multiple users' activities, especially rider and passenger.

4.4.3 Sequence Diagram

A **Sequence Diagram** shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

Sequence diagrams are made up of the following icons and elements:

- **Object symbol**: Represents a class or object in UML. The object symbol demonstrates how an object will behave in the context of the system. Class attributes should not be listed in this shape.

- **Activation box**: Represents the time needed for an object to complete a task. The longer the task will take, the longer the activation box becomes.

- **Actor symbol**: Shows entities that interact with or are external to the system.

- **Lifeline symbol**: Represents the passage of time as it extends downward. This dashed vertical line shows the sequential events that occur to an object during the charted process. Lifelines may begin with a labeled rectangle shape or an actor symbol.

- **Rider**
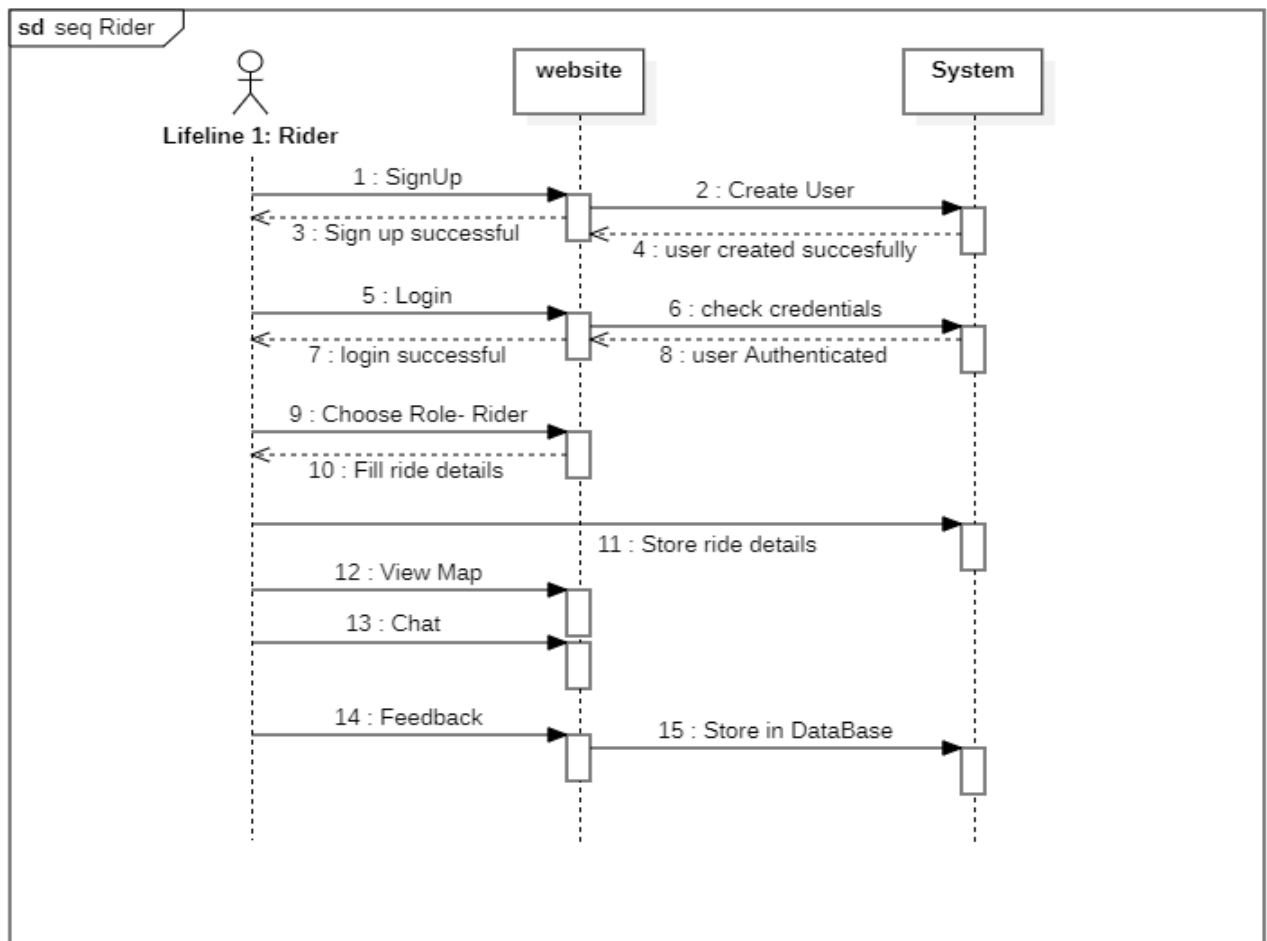
Figure 6: Sequence Diagram- Rider

The interaction between the rider, the website, and the system is depicted in the diagram above.

- **Passenger**
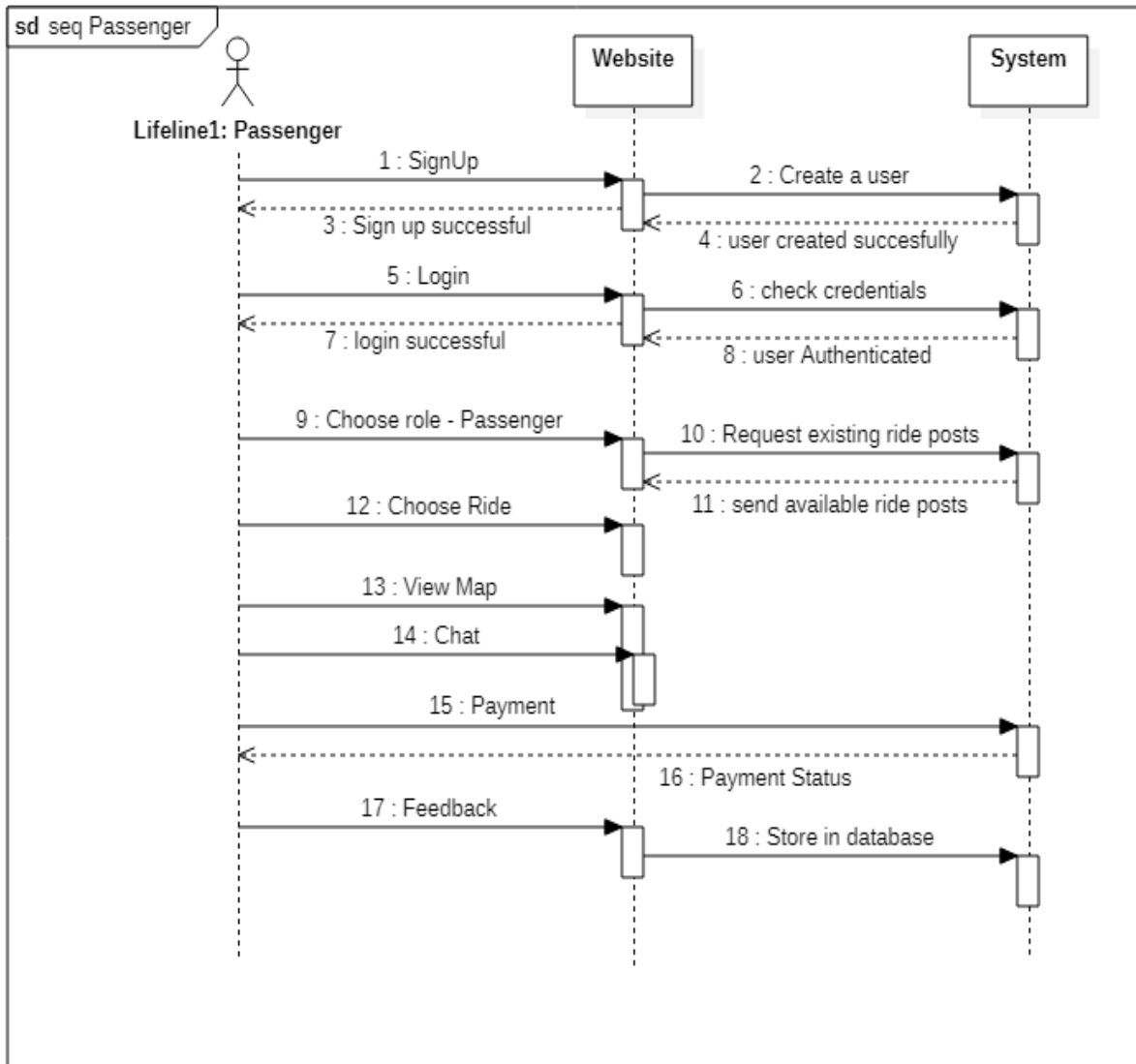
Figure 7: Sequence Diagram- Passenger

The interaction between the passenger, the website, and the system is depicted in the diagram above.

- **Admin**
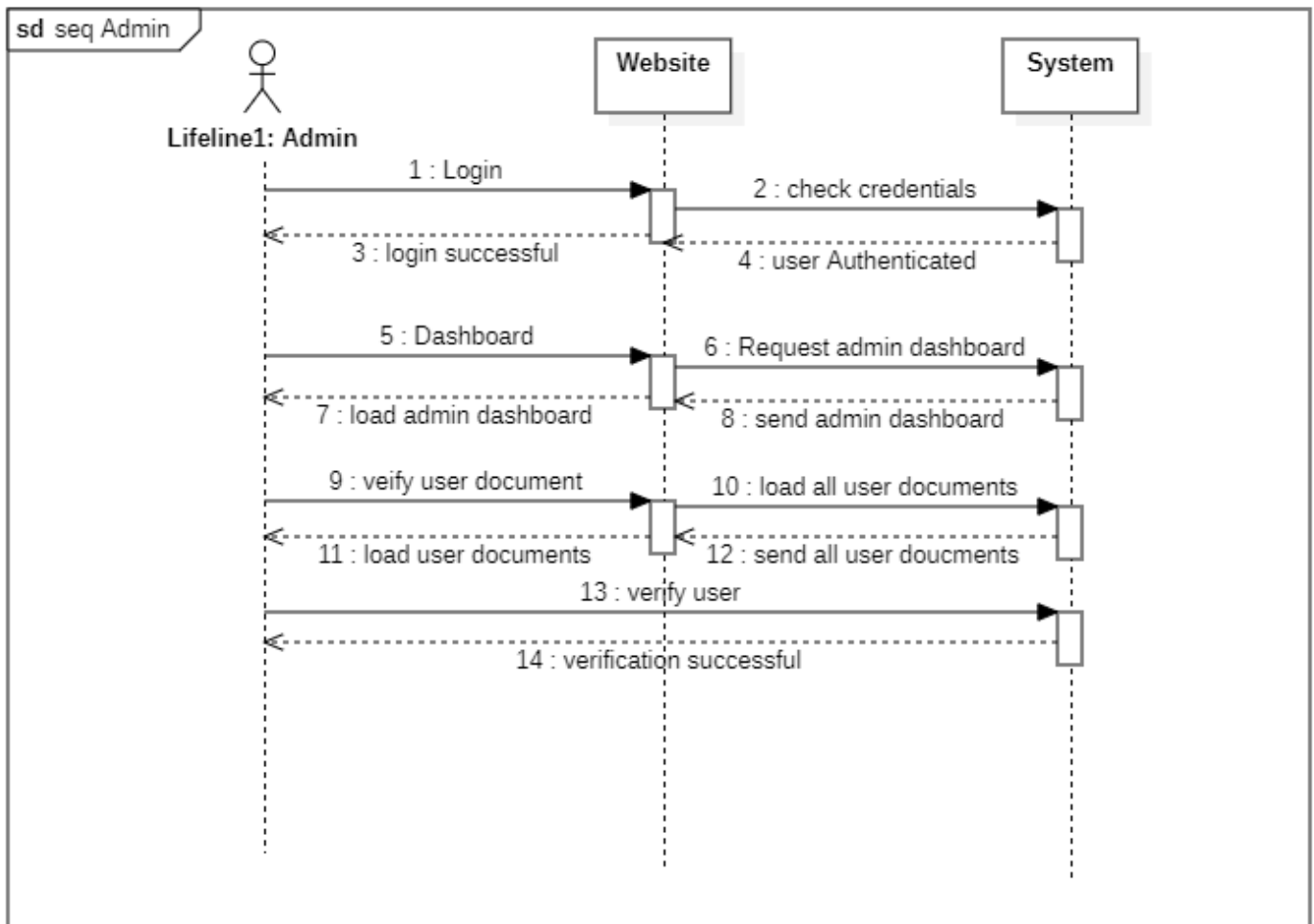
The role of admin and interaction between the admin, the website, and the system is depicted in the diagram above.

4.4.4 Class Diagram

A **Class Diagram** in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, and the relationships between the classes. The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code.

Class diagrams are made up of the following icons and elements:

- **Classes**: A template for creating objects and implementing behavior in a system. In UML, a class represents an object or a set of objects that share a common structure and behavior. They're represented by a rectangle that includes rows of the class name, its attributes, and its operations.

    o **Name**: The first row in a class shape.

    o **Attributes**: The second row in a class shape. Each attribute of the class is displayed on a separate line.

    o **Methods**: The third row in a class shape. Also known as operations, methods are displayed in list format with each operation on its own line.

- **Data types**: Classifiers that define data values. Data types can model both primitive types and enumerations.

- **Packages**: Shapes designed to organize related classifiers in a diagram. They are symbolized with a large tabbed rectangle shape.
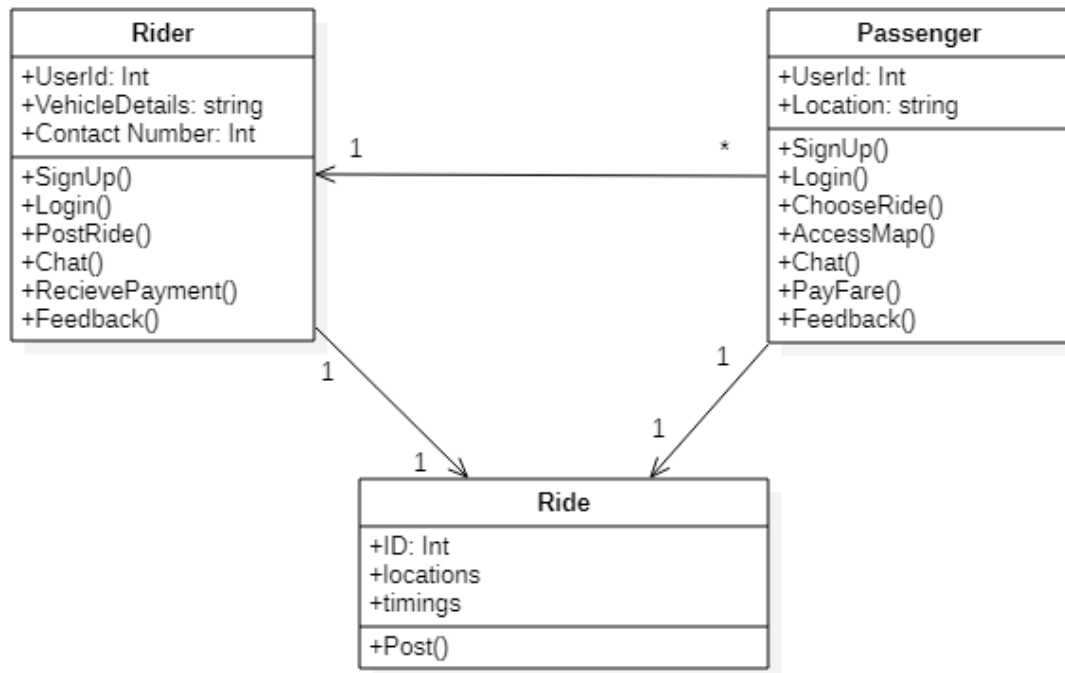
Figure 8: Class Diagram

The given diagram depicts several classes, as well as their properties and operations, to offer a useful overview of the Carpooling website.

4.4.5 Object Diagram

An object diagram shows this relation between the instantiated classes and the defined class, and the relation between these objects in the system. They are be useful to explain smaller portions of your system, when your system class diagram is very complex, and also sometimes modeling recursive relationship in diagram.

- Every object is actually symbolized like a rectangle, that offers the name from the object and its class underlined as well as divided with a colon.
- Similar to classes, you are able to list object attributes inside a separate compartment. However, unlike classes, object attributes should have values assigned for them.
- Links tend to be instances associated with associations. You can draw a link while using the lines utilized in class diagrams.

The major elements of an object diagram are:

- **Objects**: Objects are instances of a class.
- **Class titles**: Class titles are the specific attributes of a given class. In the family tree object diagram, class titles include the name, gender, and age of the family members.

Rider

+UserId: int
+vehicleDetails
+ContactDetails
+rides
+feedbacks

Passenger

+UserID: int
+location
+rides
+feedbacks

Ride

+Id: int
+location
+time
+vehicleType
+vehicleNumber
+seats: int
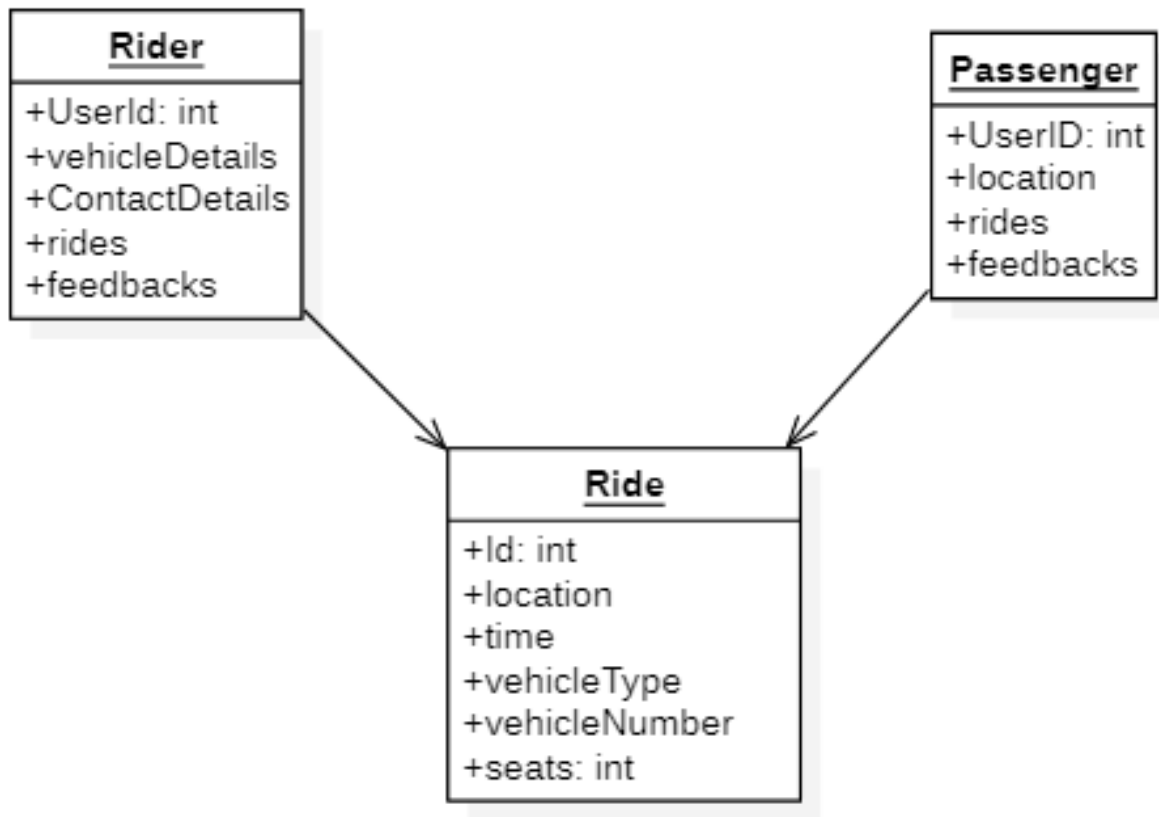
Figure 9: Object Diagram

The diagram above depicts a more detailed representation of the project's classes and their properties.

4.4.6 Component Diagram

UML Component diagrams are used in modeling the physical aspects of object-oriented systems that are used for visualizing, specifying, and documenting component-based systems and also for constructing executable systems through forward and reverse engineering. Component diagrams are essentially class diagrams that focus on a system's components that often used to model the static implementation view of a system. A component diagram breaks down the actual system under development into various high levels of functionality. Each component is responsible for one clear aim within the entire system and only interacts with other essential elements on a need-to-know basis. A component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment.

The following are shape types that are will commonly encountered when reading and building component diagrams:

- **Component**: An entity required to execute a stereotype function. A component provides and consumes behavior through interfaces, as well as through other components.
- **Node**: Represents hardware or software objects, which are of a higher level than components.
- **Interface**: Shows input or materials that a component either receives or provides. Interfaces can be represented with textual notes or symbols, such as the lollipop, socket, and ball-and-socket shapes.
- **Package**: Groups together multiple elements of the system and is represented by file folders
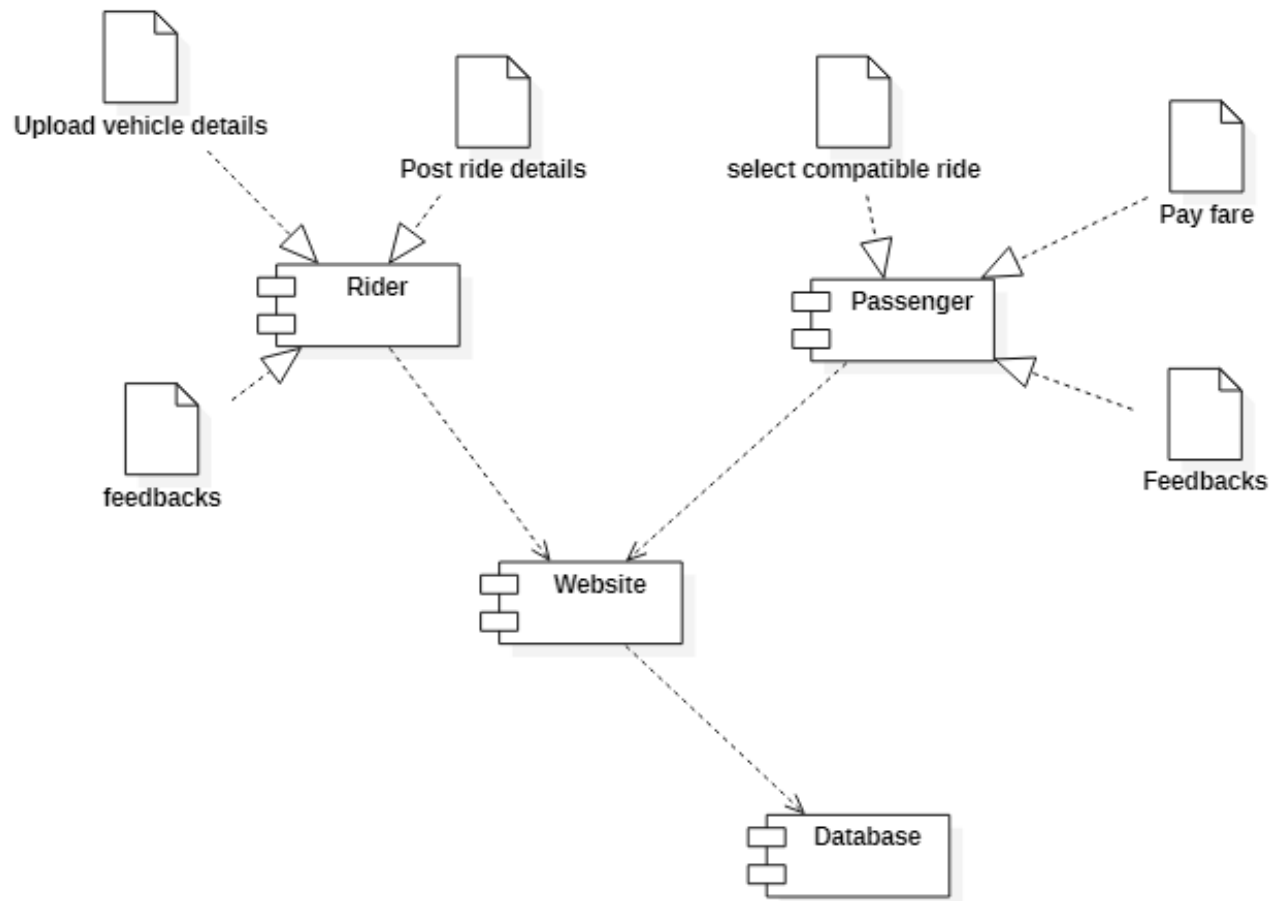
Figure 10: Component Diagram

The diagram above depicts how the project's various components function and behave. The collection of needed interfaces that a component realises or implements is provided by a component.

4.4.7 Deployment Diagram

A UML deployment diagram is a diagram that shows the configuration of run time processing nodes and the components that live on them. Deployment diagram is a kind of structure diagram used in modeling the physical aspects of an object-oriented system. They are often be used to model the static deployment view of a system (topology of the hardware). They show the structure of the run-time system.

They capture the hardware that will be used to implement the system and the links between different items of hardware. They model physical hardware elements and the communication paths between them. They can be used to plan the architecture of a system. They are also useful for Document the deployment of software components or nodes.

The basic elements used in Deployment diagram are:

- **Artifact**: A product developed by the software, symbolized by a rectangle with the name and the word "artifact" enclosed by double arrows.
- **Association**: A line that indicates a message or other type of communication between nodes.
- **Component**: A rectangle with two tabs that indicates a software element.
- **Dependency**: A dashed line that ends in an arrow, which indicates that one node or component is dependent on another.
- **Interface**: A circle that indicates a contractual relationship. Those objects that realize the interface must complete some sort of obligation.
- **Node**: A hardware or software object, shown by a three-dimensional box.
- **Node as container**: A node that contains another node inside of it—such as in the example below, where the nodes contain components.
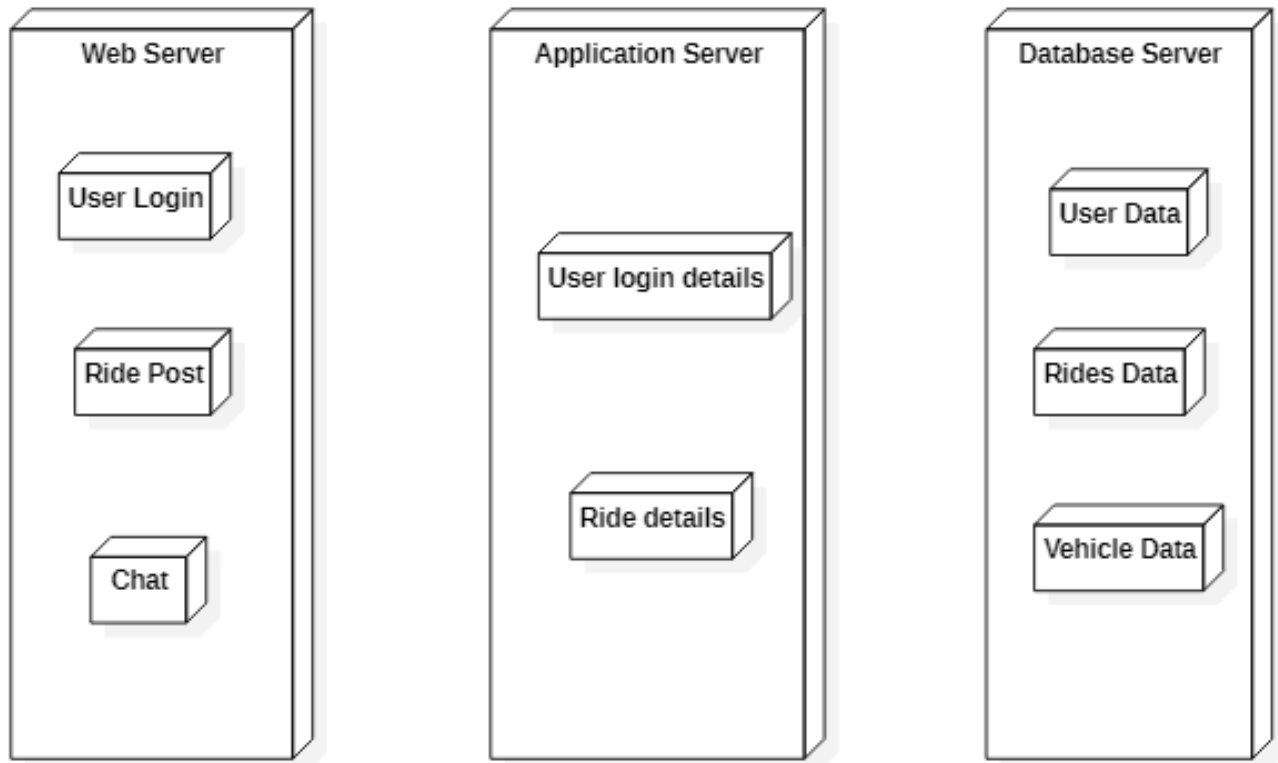
Figure 11: Deployment Diagram

This diagram depicts the project's basic deployment diagram. The user accesses the website on an application server, a database server, and a web server.

4.5   User Interface Design

1. Login Page:

This is the homepage where user can login with their username and password.
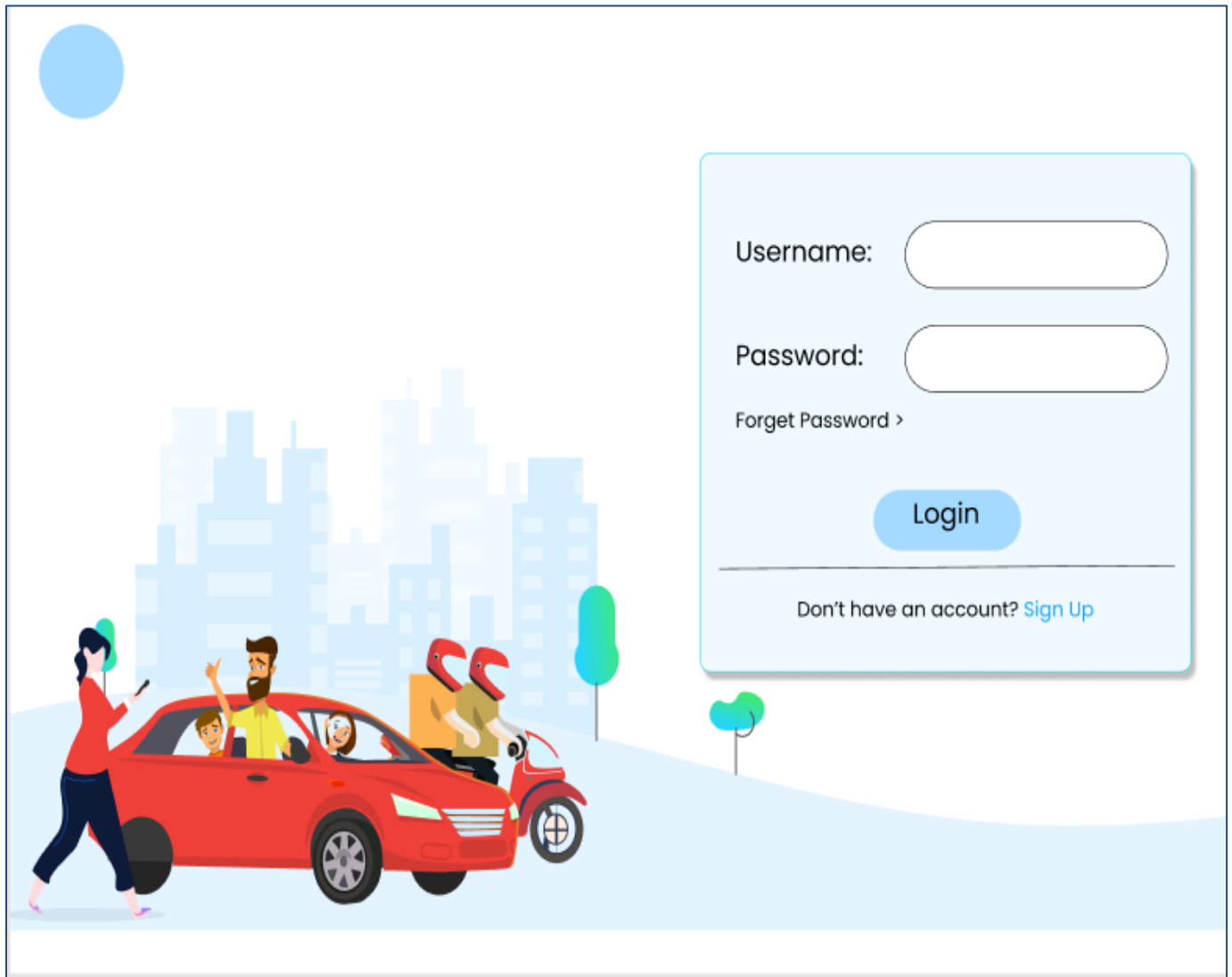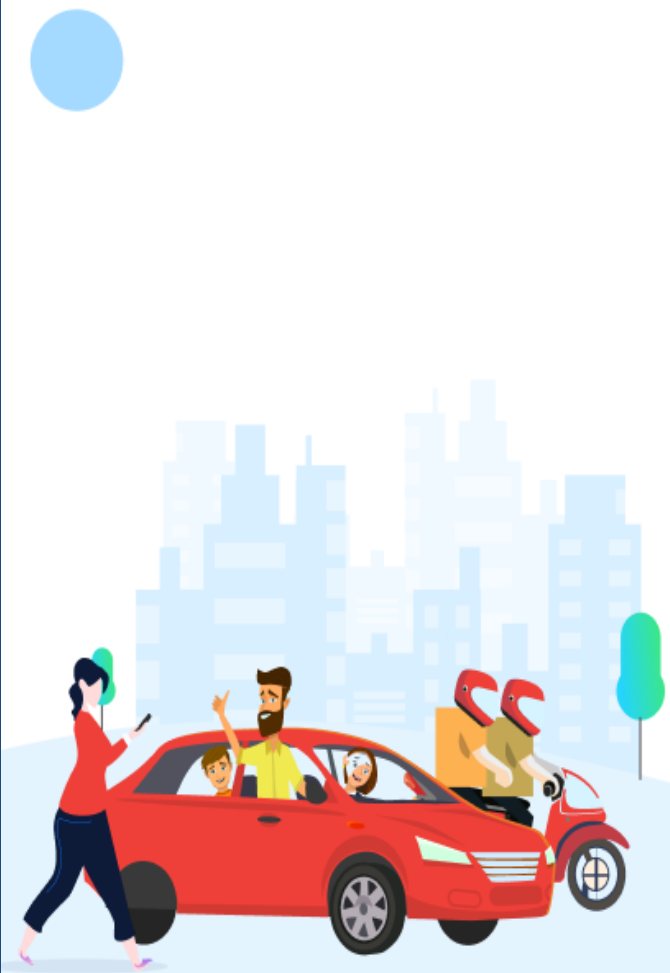
Figure 12: Login Page

2. SignUp Page:

If user does not have an account, he/she can create a new account using the following sign-up form.

Figure 13: Login Page

3. User Profile:

This is the user profile, which contains all of the user's information. The user's feedback will be displayed on their profile. The history of the rides will be highlighted as well.
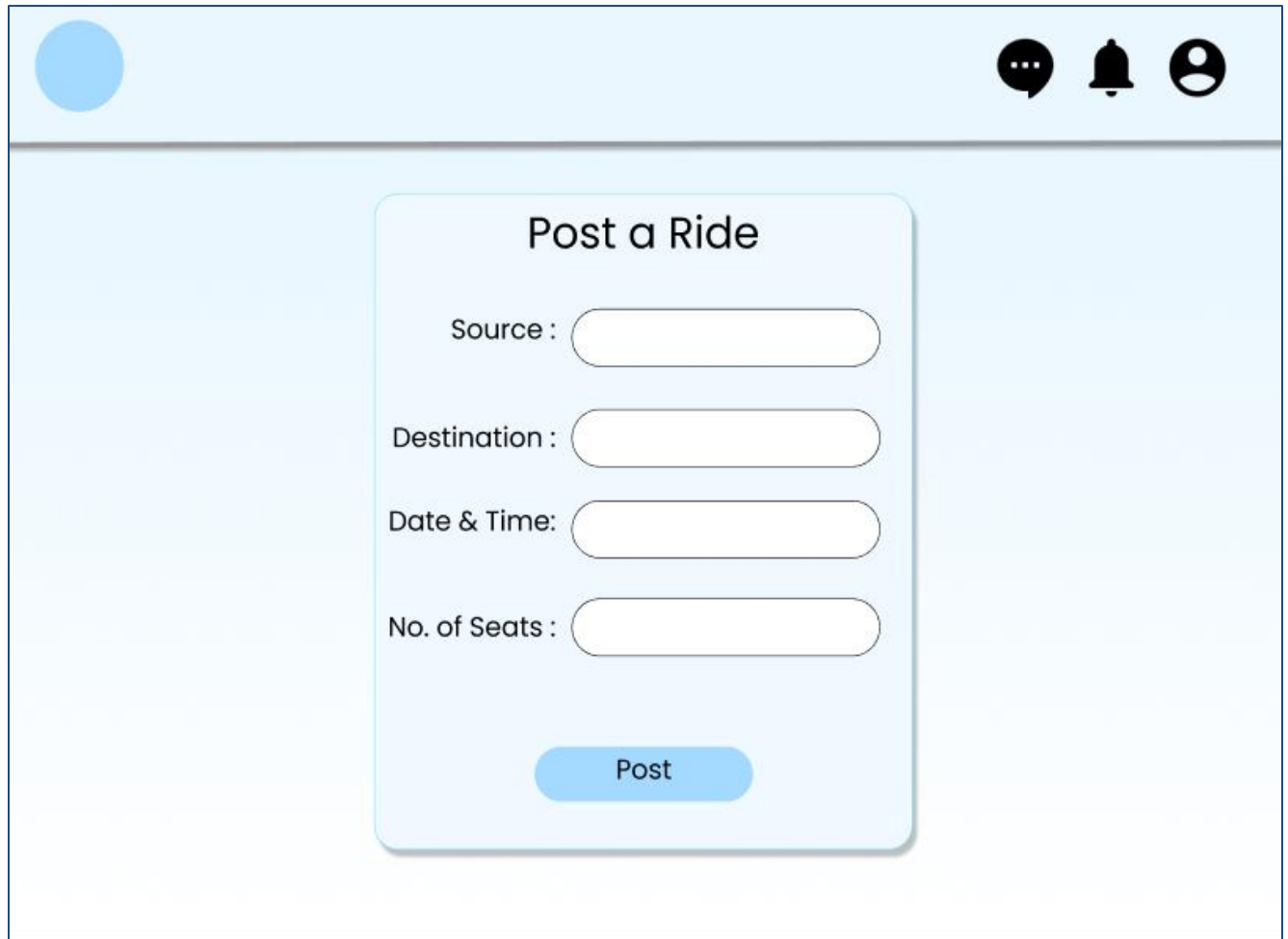
Figure 14: User Profile Page

4. Posting a Ride:

If user chooses to be a Rider he/she has to post the ride details which includes source and destination location along with date & time and number of seats.

Figure 15: Posting Page

5. **Finding a Ride:**

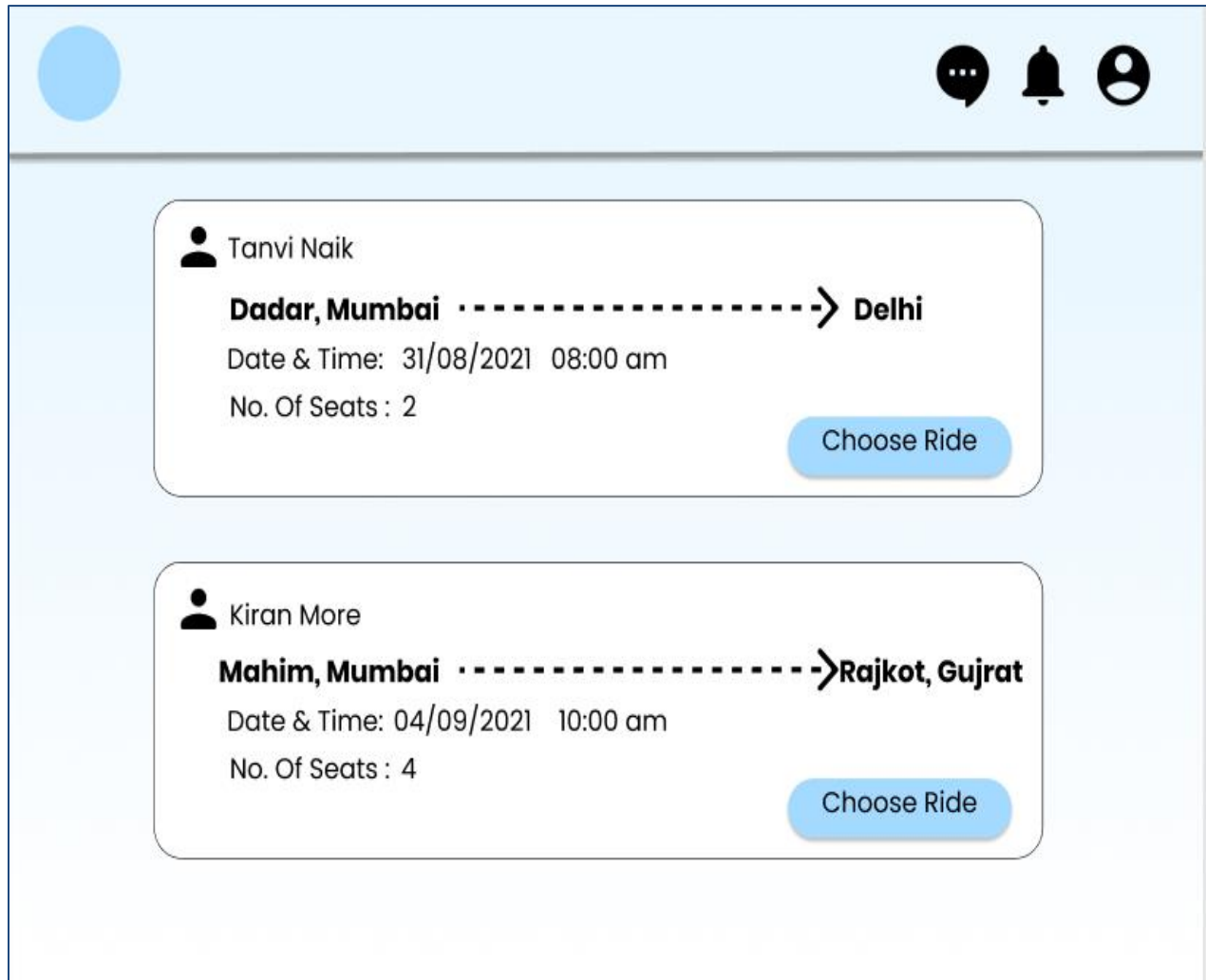   If user chooses Passenger role, he/she can select the compatible ride among the rides posted by the rider users.

Figure 16: Finding-Ride Page

6. Chat:

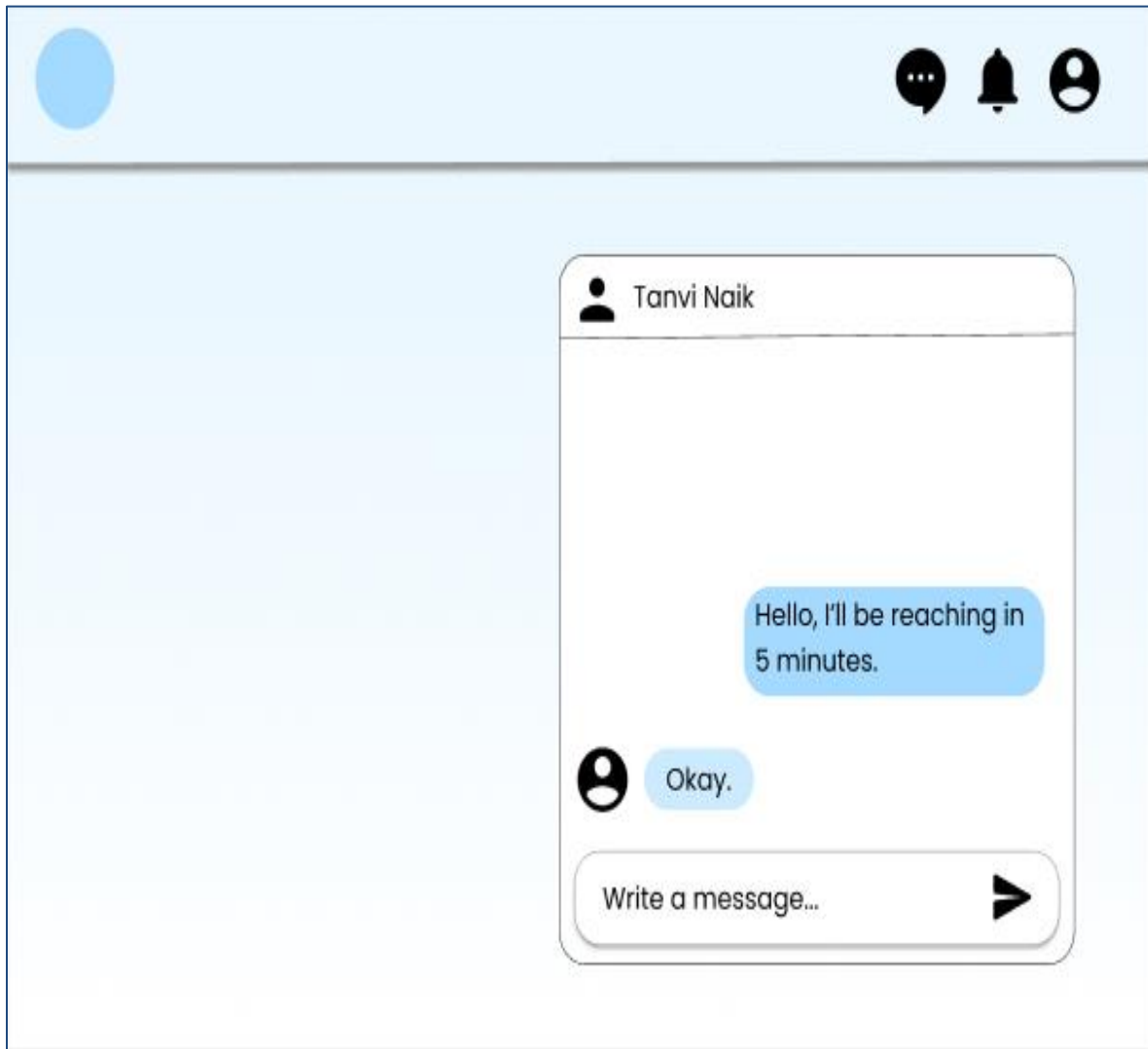Here the passenger user and rider user can communicate through chat.
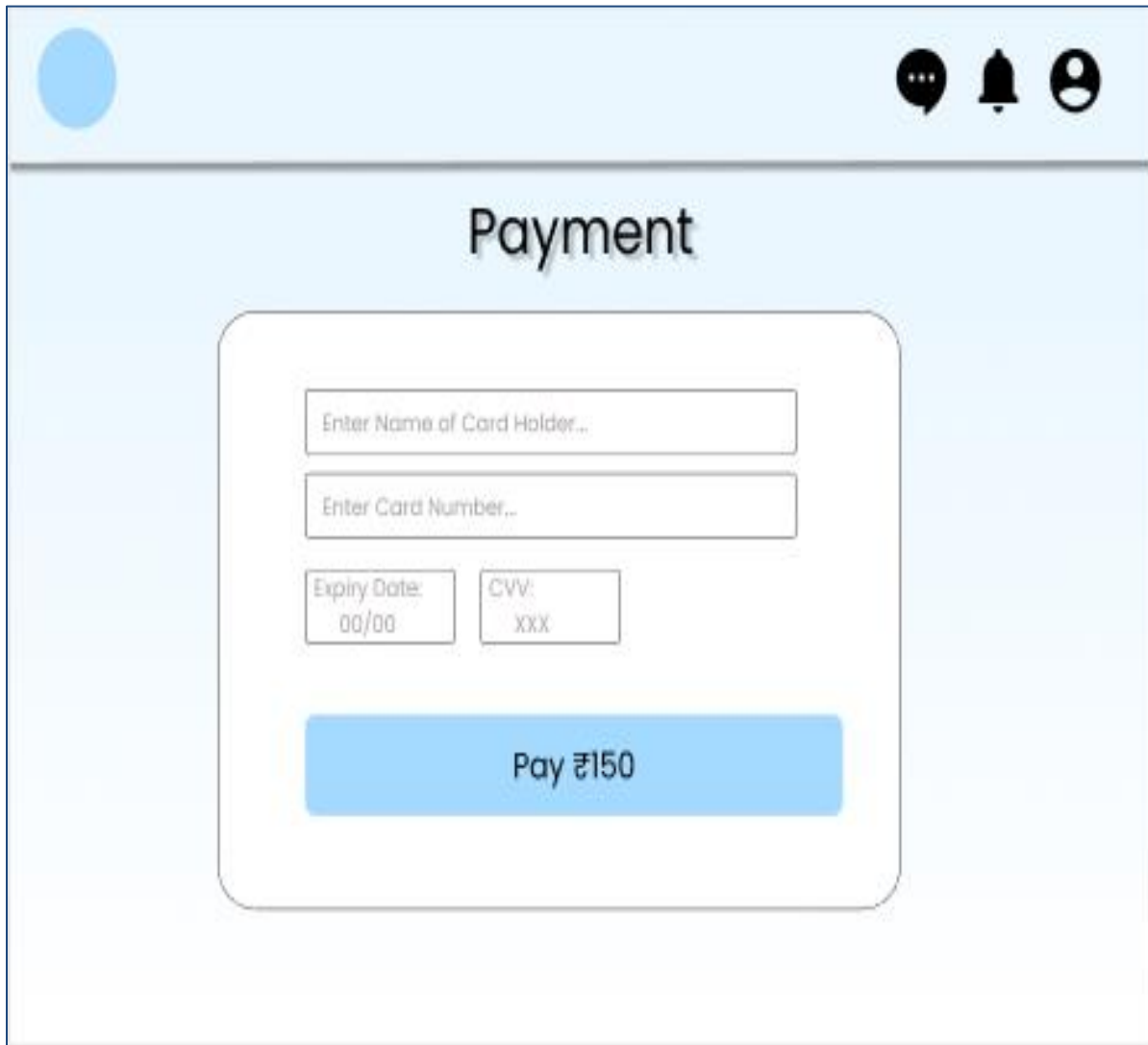
Figure 17: Chat

7. Payment:

Here the passenger can pay the decided amount to the rider.
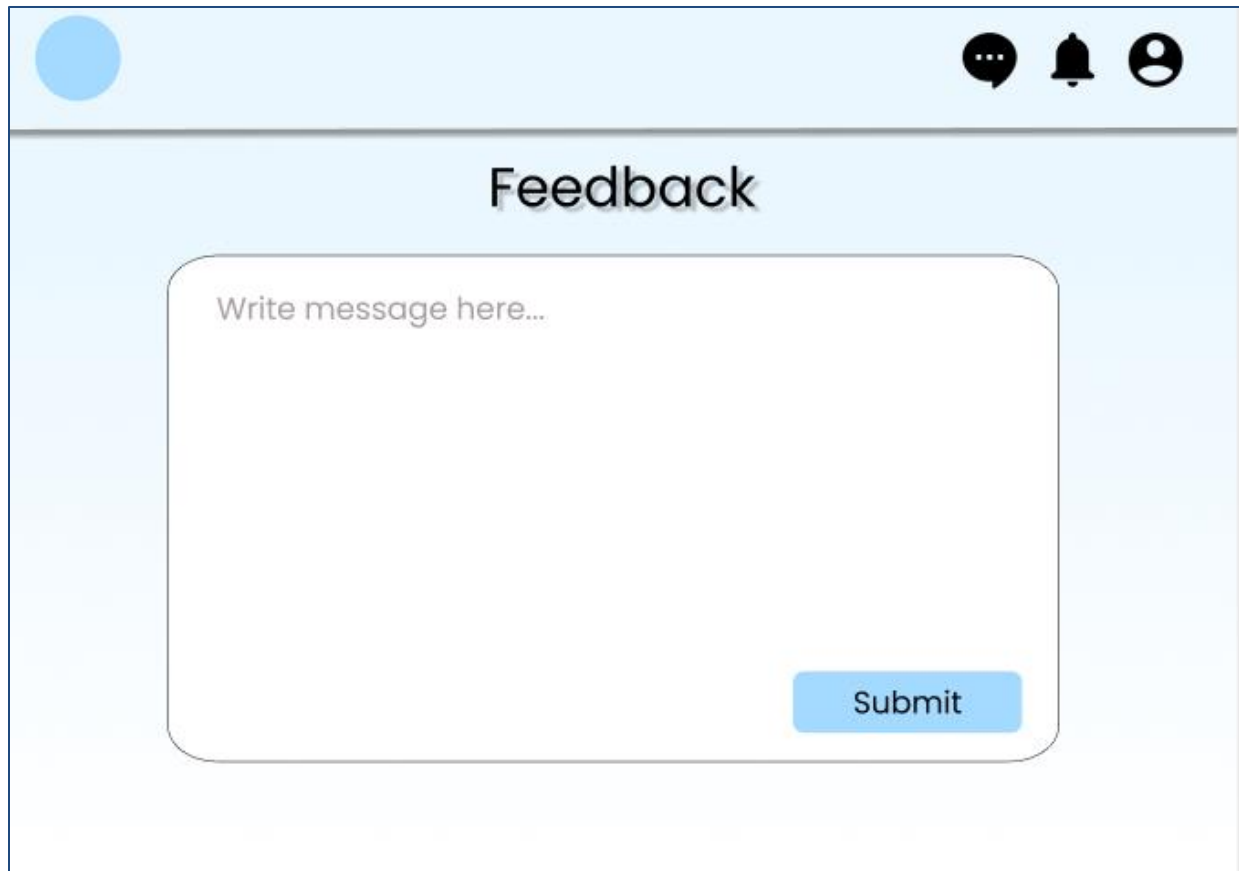
Figure 18: Payment Page

8. Feedback:

After the ride, feedback option will be given to both the users (Rider and Passenger)
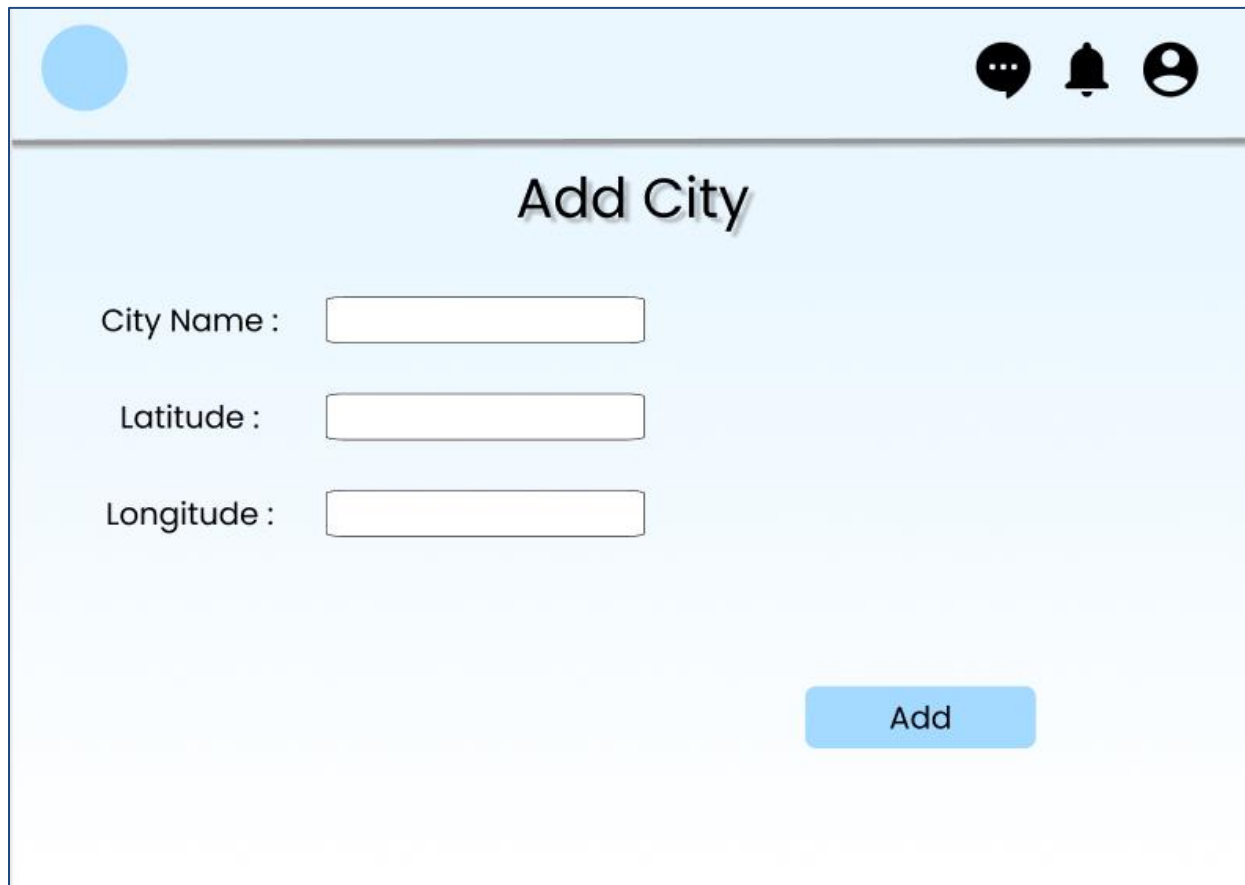
Figure 19: Feedback Page

9. Add city:

On this page, admin can add new cities to the database.

Figure 20: Add city Page

4.6   Security Issues

Data security means protecting your precious data from unauthorized access. Data in database should be kept secure and safe to unauthorized modifications. Only authorized users should have the grant to access the database. There is a username set for all the users who access the database with password so that no other guy can access this information. DBMS always keep database tamperproof, secure and theft free.

Issues	Solutions
User data	Data will be kept in encrypted form in the database and accessible only by users.
User password	Access will be given only to the valid users
Payment	All the data will be secured and safe with the provided software.
Chat	Chat will be stored in encrypted format

Table 9: Security Issues

4.7 Event table

Sr No.	Event	Trigger	Source	Activity	Response	Destination
1.	User visits the website	Request to signup	User	Validate all fields	Signup Page	Database
2.	User request to login into the website	Request for Login	User	Verify username and password	Homepage of RideAlong	Database
3.	User request rider role	Request from Main Page	User	Provider role Rider	Allow user to post	Database
4.	User request to post ride details	Request for posting	Rider	Posting ride details	Post success	Database
5.	Rider request to view Map	Request for Map	Rider	Ride Post	Map	Database
6.	Rider request to chat	Request for Chat	Rider	Chat	Chat page	Database
7.	Rider request to give feedback	Request for feedback	Rider	Feedback	Feedback page	Database
8.	Passenger requests to view rides	Request for rides page	Passenger	Rides	Rides Page	Database
9.	Passenger requests to chat	Request for chat	Passenger	Chat	Chat page	Database
10.	Passenger requests to pay	Request to payment page	Passenger	Payment Page Details	Payment Page	Database

11.	User requests to view profile	Request to view profile	User	User profile	Profile Page	Database
12.	Admin requests to add city	Request to add city page	Admin	Add city	Add city page	Database
13.	User requests for Logout	Request for Logout	User	User Logged out	Login/Home Page	Database

Table 10: Event Table

4.8 Test Cases Design

a) SignupForm

Test Case ID	Test Scenario	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result
TC_SIGNUP_001	Verify the registration of User	Enter Valid Name, Username, Email, Mobile No, Gender,Password	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 4) Enter Password 5) Confirm Password 6) Mobile Number 7) Select Gender 8) Click 'SignUp' Button	\<Valid Name\> \<Valid Username\> \<Valid Email\> \<Valid Password\> \<Matched Password\> \<Valid Mobile number\> \<Selected Gender\>	Successful Signup
TC_SIGNUP_002	Verify the registration of User	Enter null Name and valid	Visit the Signup Page	1) Enter Name 8) Click 'SignUp' Button	\<Invalid Name\>	A message 'Name must be atleast 3
TC_SIGNUP_003	Verify the registration of User	Enter Valid Name, Email,	Visit the Signup Page	1) Enter Name 2) Enter UserName 8) Click 'SignUp' Button	\<Valid Name\> \<Invalid Username\>	A message 'Username must be alphanumeric
TC_SIGNUP_004	Verify the registration of User	Enter Valid Name, Username, Mobile No,	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 8) Click 'SignUp' Button	\<Valid Name\> \<Valid Username\> \<Invalid Email\>	A message 'Enter valid email-id' is shown
TC_SIGNUP_005	Verify the registration of User	Enter Valid Name, Username, Email, Mobile No, Gender and Password of 6	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 4) Enter Password 8) Click 'SignUp' Button	\<Valid Name\> \<Valid Username\> \<Valid Email\> \<Invalid Password\>	A message 'Password must be minimum 8 characters and alphanumeric' is
TC_SIGNUP_006	Verify the registration of User	Enter Valid Name, Username, Email, Mobile No, Gender,Password and different	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 4) Enter Password 5) Confirm Password 8) Click 'SignUp' Button	\<Valid Name\> \<Valid Username\> \<Valid Email\> \<Valid Password\> \<Unmatched Password\>	A message 'Password and Confirm Password must match' is shown
TC_SIGNUP_007	Verify the registration of User	Enter Valid Name, Username, Email, Mobile No,Password and Null Gender	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 4) Enter Password 5) Confirm Password 6) Mobile Number 7) Select Gender 8) Click 'SignUp' Button	\<Valid Name\> \<Valid Username\> \<Valid Email\> \<Valid Password\> \<Matched Password\> \<Valid Mobile number\> \<No Gender Selected \>	A message 'Gender is Required' is shown
TC_SIGNUP_008	Verify the registration of User	Enter Name and valid	Visit the Signup Page	1) Enter Name 8) Click 'SignUp' Button	\<Invalid Name\>	A message 'Name must be atleast 3
TC_SIGNUP_009	Verify the registration of User	Enter Valid Name, Username, Email, Mobile No, Gender, Password and empty Mobile Number	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 4) Enter Password 5) Confirm Password 6) Mobile Number 8) Click 'SignUp' Button	\<Valid Name\> \<Valid Username\> \<Valid Email\> \<Valid Password\> \<Matched Password\> \<Invalid Mobile number\>	A message 'Contact Number must be 10 digits' is shown
TC_SIGNUP_010	Verify the	Enter 2 charactered	Visit the	1) Enter Name	\<Invalid Name\>	A message 'Name

b)   Login Form

Test Case ID	Test Scenario	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result
TC_Login_001	Verify the login of User	Enter Valid Email and Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	\<Valid Email\> \<Valid Password\>	Successful Login
TC_Login_002	Verify the login of Admin	Enter Valid Email and Password	Need a Valid Admin Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	\<Valid Admin Email\> \<Valid Admin Password\>	Successful Login
TC_Login_003	Verify the login of User	Enter wrong Email and Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	\<Invalid Email\> \<Invalid Password\>	A message 'Bad Credentials' is shown
TC_Login_004	Verify the login of User	Enter Invalid Email and Valid Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	\<Invalid Email\> \<Valid Password\>	A message 'Enter valid email' is shown
TC_Login_005	Verify the login of User	Enter Valid Email and Empty Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	\<Valid Email\> \<Empty Password\>	A message 'Password field is required' is shown
TC_Login_006	Verify the login of User	Enter null Email and Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	\<Invalid Email\> \<Invalid Password\>	A message 'Bad Credentials' is shown
TC_Login_007	Verify the login of User	Enter null Email and Valid Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	\<Invalid Email\> \<Valid Password\>	A message 'Enter valid email' is shown
TC_Login_008	Verify the login of User	Enter Valid Email and wrong Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	\<Valid Email\> \<Empty Password\>	A message 'Bad Credentials' is shown

c) Add vehicle form

Test Case ID	Test Scenario	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result
TC_addVehicle_001	Add User's vehicle	Enter Valid Vehicle details	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Valid No. of Seats> <Documents Uploaded>	A message "Vehicle added successfully" is shown
TC_addVehicle_002	Add User's vehicle	Enter empty model and other	Visit the add-vehicle page	1) Enter Model 5) Click 'Add Vehicle' Button	<Invalid Model>	A message "Please provide
TC_addVehicle_003	Add User's vehicle	Enter numeric nameplate and other valid	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 5) Click 'Add Vehicle' Button	<Valid Model> <Invalid nameplate>	A message "Enter valid nameplate number" is shown
TC_addVehicle_004	Add User's vehicle	Enter empty number of seats and other Valid Vehicle details	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Empty No. of Seats>	A message "Number of seats can't be empty" is shown
TC_addVehicle_005	Add User's vehicle	Enter Valid Vehicle details but no documents	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Valid No. of Seats> <No Documents Uploaded>	A message "Vehicle Documents are required" is shown
TC_addVehicle_006	Add User's vehicle	Enter Invalid model and other	Visit the add-vehicle page	1) Enter Model 5) Click 'Add Vehicle' Button	<Invalid Model>	A message "Please provide
TC_addVehicle_007	Add User's vehicle	Enter empty nameplate and other valid	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 5) Click 'Add Vehicle' Button	<Valid Model> <Invalid nameplate>	A message "Enter valid nameplate number" is shown
TC_addVehicle_008	Add User's vehicle	Enter 1 number of seats and other Valid Vehicle details	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Empty No. of Seats>	A message "Number of seats can't be empty" is shown
TC_addVehicle_009	Add User's vehicle	Enter Valid Vehicle details but no vehicle Rc documents	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Valid No. of Seats> <No Documents Uploaded>	A message "Vehicle Documents are required" is shown
TC_addVehicle_010	Add User's vehicle	Enter Valid Vehicle details but no Driving License documents	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Valid No. of Seats> <No Documents Uploaded>	A message "Driving license are required" is shown

Chapter 5
Implementation and Testing

5.1   Implementation Approaches

This is the phase in the software life cycle where the actual software is implemented. The result of this phase consists of source code, together with documentation to make the code reliable. Implementation is the action that must follow any preliminary in order for something to actually happen. It encompasses all the processes involved in getting new software and hardware operating properly in its environment, including installation, configuration, and running, testing and making necessary changes. The word deployment is sometimes used to mean the same thing. Implementation refers to post sales process of guiding a client from purchase to use of the software or hardware that was purchased. This includes Requirement Analysis, Scope Analysis, Customizations, System Integrations, User Policies, User Training and Delivery. These steps are often overseen by a Project Management Methodologies set forth in the Project Management Body of Knowledge. Software Implementations involve several professionals that are relatively new to the knowledge based economy such as business Analysts, Technical Analysts, Solution Architect and Project Managers

5.2   Coding Details and Code Efficiency

5.2.1 Code Efficiency

- Naming convention have been utilized for user element and that of variables and functions which help for better understanding of the application code.
- The application does not contain any type of redundant code which might lead to slowing down of applications response time.
- Comments have been declared whenever required for further references.
- The code is written according to the logic which was required for the application to present itself and hence avoiding any type of unwanted lines of code.

- Simplicity has been maintained throughout the application code by making use of sections and proper formatting of the code.
- Validations have been provided in order to prevent from any type of misuse of the application by any of the user.

5.2.2 Coding Details

- **Validation for signup**

```
router.post("/signup", upload.fields([{
  name: 'pp', maxCount: 1
}, {
  name: 'document', maxCount: 1
}]), [
  check("name")
  .isLength({min: 3})
  .withMessage("Name should be atleast 3 characters"),

  check("username")
  .isAlphanumeric()
  .withMessage("Username must be alphanumeric and  atleast 6 characters")
  .isLength({min:6})
  .withMessage("Username must be alphanumeric and  atleast 6 characters"),

  check("email")
  .isEmail()
  .withMessage("Enter valid email-id"),

  check("password")
```

```
.isLength({min: 8})

.withMessage("Password must be minimum 8 characters and alphanumeric")

.isAlphanumeric()

.withMessage("Password must be minimum 8 characters and alphanumeric"),


check("gender")

.isLength({min: 1})

.withMessage("Gender is Required"),



check("contact_number")

.isLength({min: 10, max: 10})

.withMessage('Mobile number should be 10 digits only')


] , signup )
```

- **Encrypting user password**

```
userSchema
  .virtual("password")
  .set(function (password) {
   this._password = password;
   this.salt = uuidv4();
   this.encry_password = this.securePassword(password);
  })
  .get(function () {
   return this._password;
  });

userSchema.methods = {
  authenticate: function (plainPassword) {
   return this.securePassword(plainPassword) === this.encry_password
     ? true
     : false;
```

```
      },
       securePassword: function (plainPassword) {
        if (!plainPassword) return "";
        try {
          return crypto
            .createHmac("sha256", this.salt)
            .update(plainPassword)
            .digest("hex");
        } catch (err) {
          return "";
        }
       }
      };
```

- **Code for Posting a ride (backend)**

```
createRide = (req,res)=>{

  const errors = validationResult(req);

  // checking for validation errors
  if(!errors.isEmpty()){
    return res.status(422).json({
      error: errors.array()[0].msg
    })//422- Unprocessable entity
  }
  if(req.profile.verificationStatus !== true){
    return res.status(400).json({
      error: "Documents needs to be verified by admin before posting a ride"
    })
  }

  const ride = new Ride(req.body);
```

```
    ride.driverUser = req.profile._id



ride.save((err, ride) =>{
   if(err){
      return res.status(400).json({
         error: `${err}`
      })
   }
   req.ride = ride;


   User.findByIdAndUpdate(req.profile._id,{
      $push: {
         "rides": ride._id
      }
   },
   {new: true, useFindAndModify: false },
   (error, user)=>{
      if(error){
         return res.status(400).json({
            err: "Unable to add ride in user profile"
         })
      }
      return res.json({
         message: "Ride posted successfully"
      })
   })
})
}
```

- **Code for posting Ride (frontend)**

```javascript
const createRide = (ride, userId, token) => {
return fetch(`/api/createRide/${userId}`, {
    method: "POST",
    headers: {
        Accept: "application/json",
        "Content-Type": "application/json",
        Authorization: `Bearer ${token}`
    },
    body: JSON.stringify(ride)
}).then( response => {
    return response.json();
})
.catch( err => console.log(err))
}

const onSubmit = (event) =>{
    event.preventDefault();
    setValues({...values, loading: true, error:""})
    console.log(destinationLocation)

    createRide({ sourceLocation, destinationLocation, startTime, vehicle, fare, seats},
user._id, token).then(data => {
        if(data.error){
            setValues({...values, error: data.error, loading: false})
        }else{
            setValues({...values, loading: false, success: data.message})
        }
    }).catch(err => {
        console.log("Failed to post the ride")
    })
}
```

- **Display all rides based on conditions**

```javascript
getAllRides().then(data => {
        let rides=[]
```

```
        let today = new Date()
        data.rides.map(ride => {
            let rideDate =  ride.startTime.split('T')[0]
            let rideTime =ride.startTime.split('T')[1].split('.')[0]
            let todayDate = today.getFullYear() + '-' + today.getMonth() + '-' +
today.getDate()
            let todayTime = today.getHours() + ':' + today.getMinutes() + ':' +
today.getSeconds()


            rides.push(ride)
            if(
            (!ride.passengers.includes(user._id)) &&
            (!ride.requests.includes(user._id)) &&
            (! (ride.driverUser._id == user._id)) &&
            (! (ride.passengers.length == ride.seats)) &&
            ( (rideDate < todayDate ? (rideTime > todayTime ? true : false) : false)
            ))
            {
              rides.push(ride)
            }
        })


        setValues({...values, loading:false, rides: rides})
      })
```

- **Implementing mapbox geocoder:**

```
const geocoder = new MapboxGeocoder({
    // Initialize the geocoder
    accessToken: mapboxgl.accessToken, // Set the access token
    mapboxgl: mapboxgl, // Set the mapbox-gl instance
    marker: true, // Do not use the default marker style
    placeholder: 'Search for a location', // Placeholder text for the search bar
  });

  useEffect(()=>{
    const map = new mapboxgl.Map({
        container: 'map', // Container ID
        style: 'mapbox://styles/mapbox/streets-v11', // Map style to use
        center: [longitude, latitude], // Starting position [lng, lat]
```

```
    zoom: 12, // Starting zoom level
  });


// Add the geocoder to the map
map.addControl(geocoder);
map.on('load', () => {
map.addSource('single-point', {
  type: 'geojson',
  data: {
    type: 'FeatureCollection',
    features: []
  }
});

// Listen for the `result` event from the Geocoder
// `result` event is triggered when a user makes a selection
//  Add a marker at the result's coordinates
geocoder.on('result', (event) => {
    console.log(event.result)
    // setLatitude(event.result.geometry.coordinates[1])
    // setLongitude(event.result.geometry.coordinates[0])
    const place = event.result.place_name.split(",")


onLocationChange(event.result.geometry.coordinates[1],event.result.geometry.coordinat
es[0],  event.result.place_name)

    map.getSource('single-point').setData(event.result.geometry);
  });
});
},[])


return (
  <div id='map' className='map'>
  </div>
)
```

5.3 Testing Approach

5.3.1 Unit Testing

Unit testing is a level of software testing process where individual units or components of a software system are tested. The purpose of unit testing is to validate that each unit of the software performs as designed. A unit is the smallest testable part of the software. It usually has one or a few inputs and usually a single output. In object-oriented programming, a unit is often an entire interface, such as a class, but can be an individual method. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. In most programming languages, that is a function, a subroutine, a method or property.

Unit Testing - Advantages:

- Reduces Defects in the Newly developed features or reduces bugs when changing the existing functionality.

- Reduces Cost of Testing as defects are captured in very early phase.

- Improves design and allows better refactoring of code.

- Unit Tests, when integrated with build gives the quality of the build as well.

Unit Testing Techniques:

- Black Box Testing - Using which the user interface, input and output are tested.

- White Box Testing - used to test each one of those functions behaviour is tested.

- Gray Box Testing - Used to execute tests, risks and assessment methods.

5.3.2 Integrated Testing

Integration testing is a level of software testing where individual units are combined and tested to verify if they are working properly. Integration testing carries a lot of significance as it helps testers in determining the effectiveness as well as the functionality of the software.

Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as 'I & T' (Integration and Testing), 'String Testing' and sometimes 'Thread Testing'. Integration tests determine if independently developed units of software work correctly when they are connected to each other.

The advantages of integration testing include the following:
- It makes sure that integrated modules work properly as intended.
- The tester can start testing once the modules to be tested are available.
- Integration testing also detects the errors of the individual modules.
- It increases the test coverage and improves the reliability of test.

Software Engineering defines variety of strategies to execute Integration testing, viz.
- Big Bang Approach :
- Incremental Approach: which is further divided into the following
- Top Down Approach
- Bottom Up Approach
- Sandwich Approach – Combination of Top Down and Bottom Up

5.3.3 Beta Testing

Beta testing is one of the types of User acceptance testing. The main goal of user acceptance testing is to check whether the developed software product fulfills the user requirements. Beta testing is performed in order to access the product by exposing it to real end users. After that, the feedback is taken from the users and the defects are fixed. It helps the software product to provide better user experience

There is no standard for what a beta test should look like and how to set up beta testing. The actual testing procedure should be relevant to your testing goals.

Features of beta testing are:

- Beta testing used in a real environment at the user's site. Beta testing helps in providing the actual position of the quality.
- Testing performed by the client, stakeholder, and end-user.
- Beta testing always is done after the alpha testing, and before releasing it into the market.
- Beta testing is black-box testing.
- Beta testing performs in the absence of tester and the presence of real users
- Beta testing is performed after alpha testing and before the release of the final product.
- Beta testing generally is done for testing software products like utilities, operating systems, and applications, etc.

5.4   Modifications and Improvements

- Modifications made in the database schema for easy storage and access of data
- Integrated Stripe Payment Gateway
- Added various types of validations to authenticate a user for security purpose

5.5 Test Cases

a)   Signup form

Test Case ID	Test Scenario	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Post-Condition	Actual Result	Status (PASS/FAIL)
TC_SIGNUP_001	Verify the registration of User	Enter Valid Name, Username, Email, Mobile No, Gender,Password	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 4) Enter Password 5) Confirm Password 6) Mobile Number 7) Select Gender 8) Click 'SignUp' Button	<Valid Name> <Valid Username> <Valid Email> <Valid Password> <Matched Password> <Valid Mobile number> <Selected Gender>	Successful Signup	Redirect to Login Page	Successful Signup	Pass
TC_SIGNUP_002	Verify the registration of User	Enter null Name and valid	Visit the Signup Page	1) Enter Name 8) Click 'SignUp' Button	<Invalid Name>	A message 'Name must be atleast 3	Stays on Signup Page	A message 'Name must be atleast 3 characters' is shown	Pass
TC_SIGNUP_003	Verify the registration of User	Enter Valid Name, Email,	Visit the Signup Page	1) Enter Name 2) Enter UserName 8) Click 'SignUp' Button	<Valid Name> <Invalid Username>	A message 'Username must be alphanumeric	Stays on Signup Page	A message 'Username must be alphanumeric and atleast 6 characters' is shown	Pass
TC_SIGNUP_004	Verify the registration of User	Enter Valid Name, Username, Mobile No,	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 8) Click 'SignUp' Button	<Valid Name> <Valid Username> <Invalid Email>	A message 'Enter valid email-id' is shown	Stays on Signup Page	A message 'Enter valid email-id' is shown	Pass
TC_SIGNUP_005	Verify the registration of User	Enter Valid Name, Username, Email, Mobile No, Gender and Password of 6	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 4) Enter Password 8) Click 'SignUp' Button	<Valid Name> <Valid Username> <Valid Email> <Invalid Password>	A message 'Password must be minimum 8 characters and alphanumeric' is	Stays on Signup Page	A message 'Password must be minimum 8 characters and alphanumeric' is shown	Pass
TC_SIGNUP_006	Verify the registration of User	Enter Valid Name, Username, Email, Mobile No, Gender,Password and different	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 4) Enter Password 5) Confirm Password 8) Click 'SignUp' Button	<Valid Name> <Valid Username> <Valid Email> <Valid Password> <Unmatched Password>	A message 'Password and Confirm Password must match' is shown	Stays on Signup Page	A message 'Password and Confirm Password must match' is shown	Pass
TC_SIGNUP_007	Verify the registration of User	Enter Valid Name, Username, Email, Mobile No,Password and Null Gender	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 4) Enter Password 5) Confirm Password 6) Mobile Number 7) Select Gender 8) Click 'SignUp' Button	<Valid Name> <Valid Username> <Valid Email> <Valid Password> <Matched Password> <Valid Mobile number> <No Gender Selected >	A message 'Gender is Required' is shown	Stays on Signup Page	A message 'Gender is Required' is shown	Pass
TC_SIGNUP_008	Verify the registration of User	Enter Name and valid	Visit the Signup Page	1) Enter Name 8) Click 'SignUp' Button	<Invalid Name>	A message 'Name must be atleast 3	Stays on Signup Page	A message 'Name must be atleast 3 characters' is shown	Pass
TC_SIGNUP_009	Verify the registration of User	Enter Valid Name, Username, Email, Mobile No, Gender, Password and empty Mobile Number	Visit the Signup Page	1) Enter Name 2) Enter UserName 3) Enter Email 4) Enter Password 5) Confirm Password 6) Mobile Number 8) Click 'SignUp' Button	<Valid Name> <Valid Username> <Valid Email> <Valid Password> <Matched Password> <Invalid Mobile number>	A message 'Contact Number must be 10 digits' is shown	Stays on Signup Page	A message 'Contact Number must be 10 digits' is shown	Pass
TC_SIGNUP_010	Verify the	Enter 2 charactered	Visit the	1) Enter Name	<Invalid Name>	A message 'Name	Stays on	A message 'Name must be	Pass

b) Login Form

Test Case ID	Test Scenario	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Post- Condition	Actual Result	Status (PASS/FAIL)
TC_Login_001	Verify the login of User	Enter Valid Email and Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	<Valid Email> <Valid Password>	Successful Login	Redirected to "Choose-Role" page	Successful Login	Pass
TC_Login_002	Verify the login of Admin	Enter Valid Email and Password	Need a Valid Admin Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	<Valid Admin Email> <Valid Admin Password>	Successful Login	Redirected to Admin dashboard	Successful Login	Pass
TC_Login_003	Verify the login of User	Enter wrong Email and Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	<Invalid Email> <Invalid Password>	A message 'Bad Credentials' is shown	Stays on Signin page	A message 'Bad Credentials' is shown	Pass
TC_Login_004	Verify the login of User	Enter Invalid Email and Valid Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	<Invalid Email> <Valid Password>	A message 'Enter valid email' is shown	Stays on Signin page	A message 'Enter valid email' is shown	Pass
TC_Login_005	Verify the login of User	Enter Valid Email and Empty Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	<Valid Email> <Empty Password>	A message 'Password field is required' is shown	Stays on Signin page	A message 'Password field is required' is shown	Pass
TC_Login_006	Verify the login of User	Enter null Email and Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	<Invalid Email> <Invalid Password>	A message 'Bad Credentials' is shown	Stays on Signin page	A message 'Bad Credentials' is shown	Pass
TC_Login_007	Verify the login of User	Enter null Email and Valid Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	<Invalid Email> <Valid Password>	A message 'Enter valid email' is shown	Stays on Signin page	A message 'Enter valid email' is shown	Pass
TC_Login_008	Verify the login of User	Enter Valid Email and wrong Password	Need a Valid Account to Login	1) Enter Email 2) Enter Password 3) Click 'Login' Button	<Valid Email> <Empty Password>	A message 'Bad Credentials' is shown	Stays on Signin page	A message 'Bad Credentials' is shown	Pass

c) Add-Vehicle Form

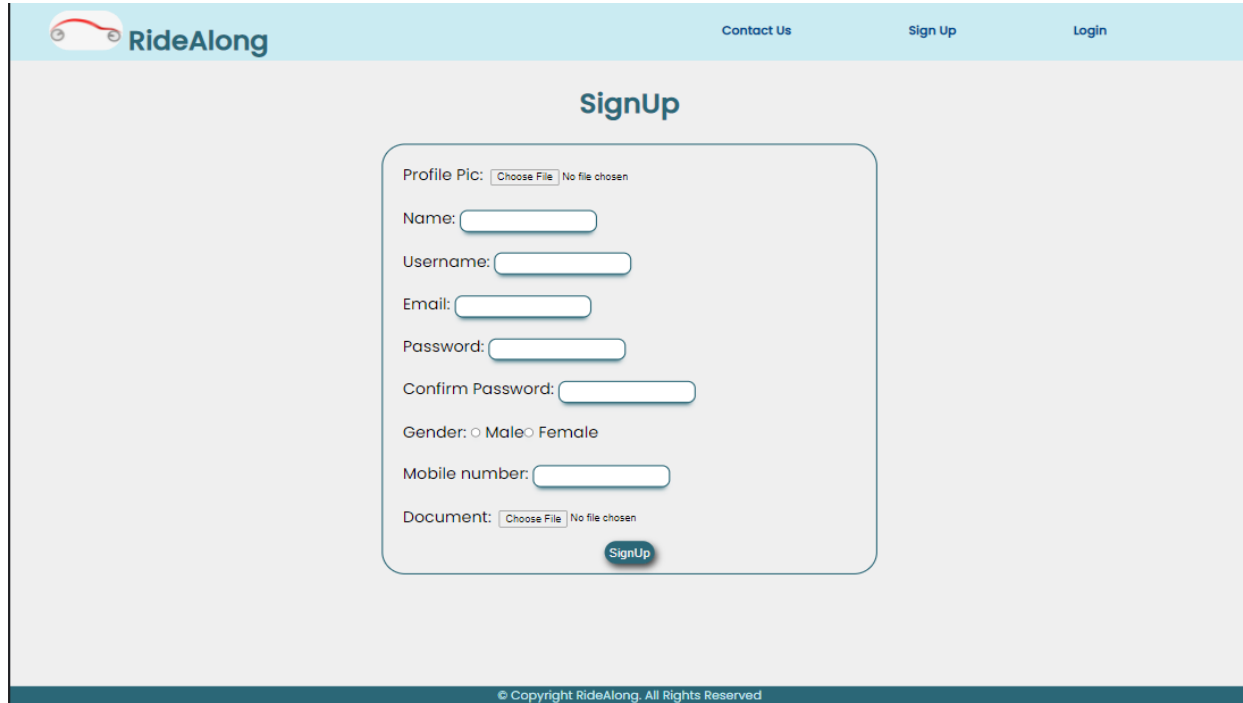| Test Case ID	Test Scenario	Test Case	Pre-Condition	Test Steps	Test Data	Expected Result	Post- Condition	Actual Result	Status (PASS/FAIL)
TC_addVehicle_001	Add User's vehicle	Enter Valid Vehicle details	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Valid No. of Seats> <Documents Uploaded>	A message "Vehicle added successfully" is shown	Successful	A message "Vehicle added successfully" is shown	Pass
TC_addVehicle_002	Add User's vehicle	Enter empty model and other Valid Vehicle details	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Invalid Model> <Valid nameplate> <Valid No. of Seats> <Documents Uploaded>	A message "Please provide model name" is shown	Stays on Add-Vehicle Page	A message "Please provide model name" is shown	Pass
TC_addVehicle_003	Add User's vehicle	Enter numeric nameplate and other valid Vehicle details	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Invalid nameplate> <Valid No. of Seats> <Documents Uploaded>	A message "Enter valid nameplate number" is shown	Stays on Add-Vehicle Page	A message "Enter valid nameplate number" is shown	Pass
TC_addVehicle_004	Add User's vehicle	Enter empty number of seats and other Valid Vehicle details	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Empty No. of Seats> <Documents Uploaded>	A message "Number of seats can't be empty" is shown	Stays on Add-Vehicle Page	A message "Number of seats can't be empty" is shown	Pass
TC_addVehicle_005	Add User's vehicle	Enter Valid Vehicle details but no documents	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Valid No. of Seats> <No Documents Uploaded>	A message "Vehicle Documents are required" is shown	Stays on Add-Vehicle Page	A message "Vehicle Documents are required" is shown	Pass
TC_addVehicle_006	Add User's vehicle	Enter Invalid model and other Valid Vehicle details	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Invalid Model> <Valid nameplate> <Valid No. of Seats> <Documents Uploaded>	A message "Please provide model name" is shown	Stays on Add-Vehicle Page	A message "Please provide model name" is shown	Pass
TC_addVehicle_007	Add User's vehicle	Enter empty nameplate and other valid Vehicle details	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Invalid nameplate> <Valid No. of Seats> <Documents Uploaded>	A message "Enter valid nameplate number" is shown	Stays on Add-Vehicle Page	A message "Enter valid nameplate number" is shown	Pass
TC_addVehicle_008	Add User's vehicle	Enter 1 number of seats and other Valid Vehicle details	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Empty No. of Seats> <Documents Uploaded>	A message "Number of seats can't be empty" is shown	Stays on Add-Vehicle Page	A message "Number of seats can't be empty" is shown	Pass
TC_addVehicle_009	Add User's vehicle	Enter Valid Vehicle details but no vehicle Rc documents	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Valid No. of Seats> <No Documents Uploaded>	A message "Vehicle Documents are required" is shown	Stays on Add-Vehicle Page	A message "Vehicle Documents are required" is shown	Pass
TC_addVehicle_010	Add User's vehicle	Enter Valid Vehicle details but no Driving License documents	Visit the add-vehicle page	1) Enter Model 2)Enter nameplate 3)Enter No. of Seats 4)Upload Documents 5) Click 'Add Vehicle' Button	<Valid Model> <Valid nameplate> <Valid No. of Seats> <No Documents Uploaded>	A message "Driving license are required" is shown	Stays on Add-Vehicle Page	A message "Driving license are required" is shown	Pass

Chapter 6

Results and Discussion

6.1   Test Reports

Test Report						
Test Cycle	**System Test**					
EXECUTED	PASSED			28		
	FAILED			0		
	(Total) TESTS EXECUTED				**28**	
PENDING					0	
IN PROGRESS					0	
BLOCKED					0	
(Sub-Total) TEST PLANNED					**28**	
(PENDING + IN PROGRESS + BLOCKED + TEST EXECUTED)						

Functions	Description	% TCs Executed	% TCs Passed	TCs pending	Priority	Remarks
Verify User by Login	Verify User Account by Login	100%	100%	0	*High*	
New User by Signing up	Check User Account is added by signing up	100%	100%	0	*High*	
Verify Admin by Login	Verify Admin Account by Login	100%	100%	0	*High*	
Add vehicle by User	Verify Vehicle Added by User	100%	100%	0	*High*	

6.2   User Documentation
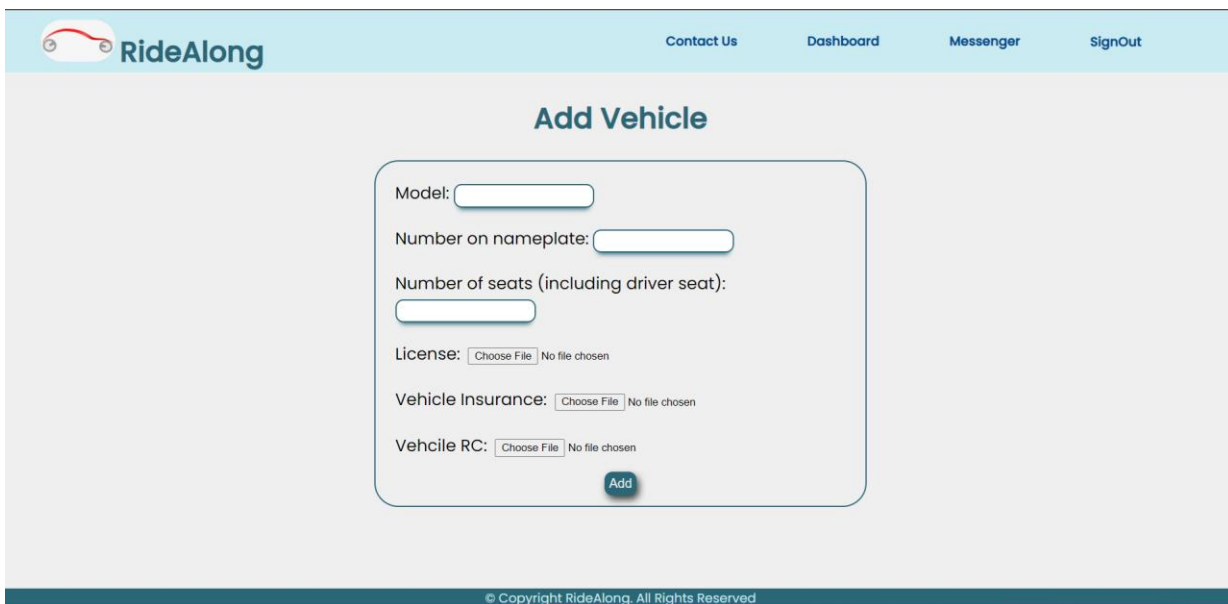
1. Steps to Register:
   a. Go to Signup page from navigation bar
   b. Fill the signup form with your details
   c. Upload the verification document to get verified by the admin
   d. Click on 'Signup' button

2. Steps to Login:
   a. Go to homepage or Login page from navigation bar
   b. Enter correct credentials
   c. Click on 'Login'

3. Steps to register your Vehicle:
   a. Go to 'Dashboard'
   b. Click on 'Add Vehicle'
   c. Fill vehicle details properly
   d. Upload the required documents
   e. Click on 'Register'

4. Steps for posting a Ride:
   a. To post a ride, your documents must be verified by the admin and you must register a vehicle first
   b. If the documents are verified go to 'Dashboard' from navigation bar
   c. Click on 'Post Ride' button
   d. To select source location, search location in map and then click on 'select' button besides source location, repeat same for destination Location
   e. Select the vehicle for ride
   f. Fill all details and click on 'Post'
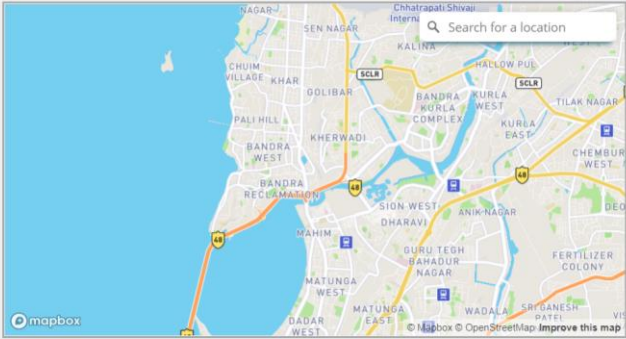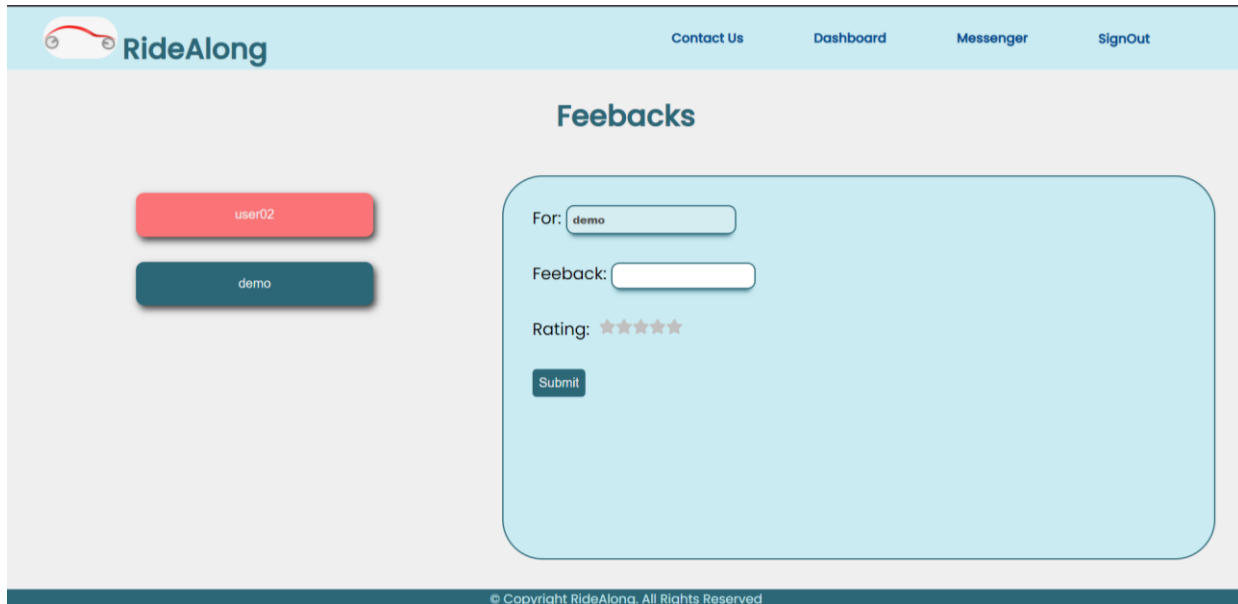
5. Steps to Find a Ride:
   a. Go to 'Dashboard'
   b. Click on 'Find a Ride' button
   c. Choose the ride compatible for you
   d. You can view the locations on map by clicking on 'View Map' button on a ride
   e. You can send Request for ride by clicking on 'Request' button
   f. You can chat with the driver by clicking on 'Chat' button
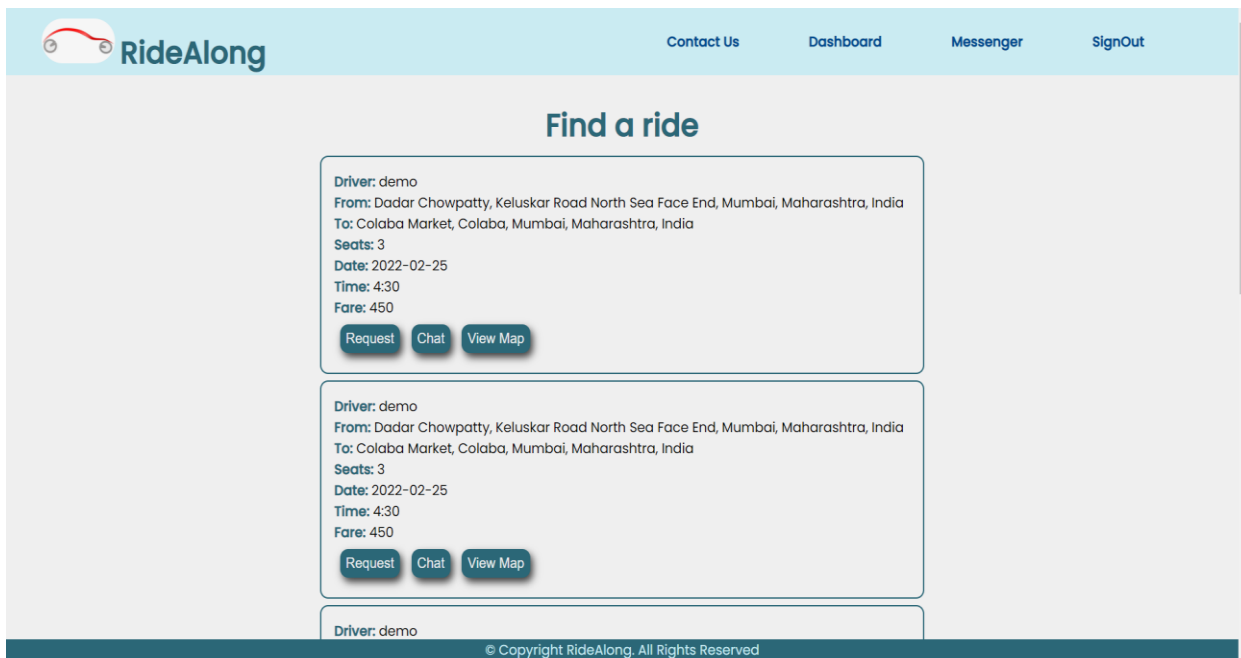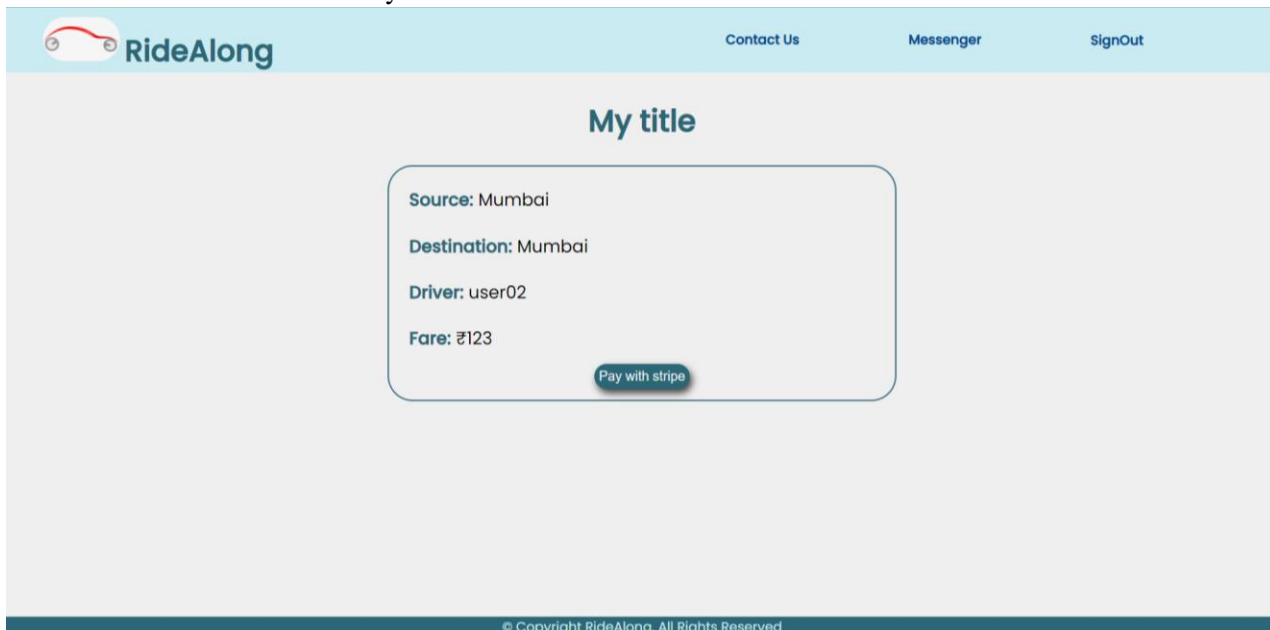
6. Steps to give feedback:
   a. Go to 'Dashboard'
   b. Click on 'Check Ride Status'
   c. Click on "Feedback"
   d. Choose user
   e. Fill feedback form
   f. Submit

7. Steps for Payment:
   a. Go to Dashboard
   b. Click on 'Check Ride Status'
   c. Click on Pay and fill the details

Chapter 7
Conclusions

7.1 Conclusions

       Carpooling is a very efficient way to minimise pollution and traffic congestion in cities. It also provides an environmentally beneficial mode of transportation. It also gives you the chance to meet new individuals. Because of the delays in public transportation and the comforts given by private automobiles, most individuals today choose to travel by private vehicle. Pre-registration guarantees that only identifiable individuals enter the car, allowing for the establishment of confidence. As a result, the proposed carpooling system will be beneficial in lowering pollution levels.

       This carpooling application is designed to be performing, scalable, extensible, and highly available. It also ensures the privacy of the user's data and secures its access. Given that it may be improved in many ways, the application is also easily maintainable. The result achieved in this project is a working Web application and server that perform the requirements stated in this document.

7.1.1 Significance of the System

- The proposed system's major goal is to implement carpooling in India and avail the benefits.
- The system is easy to use as it is user friendly
- The user information will be properly stored in the database and the passwords will also be in encrypted form
- The working of the system is very organized form

7.2 Limitations of the System

- There is no way to track the position of a car in real time.
- The accessibility of the website for color-blind individuals is not taken into account.

7.3 Future Scope of the Project

- Can add realtime tracking of ride.
- The ability to arrange monthly rides is a useful feature for users that travel to work on a daily basis can be introduced.
- Pooling system can be introduced for transportation of goods in sharing manner (Truck Pooling).

References

1. **Web References**

   - http://www.youtube.com
   - http://www.w3schools.com
   - http://www.google.com
   - www.tutorialspoint.com

2. **Documentation**

   - https://docs.mongodb.com/
   - https://nodejs.org/en/docs/
   - https://express-validator.github.io/docs/
   - https://reactjs.org/