

**AN INTERNSHIP PROJECT REPORT ON**

# **UBERIZATION SYSTEM**

**SUBMITTED TOWARDS THE  
PARTIAL FULFILLMENT OF THE REQUIREMENTS OF**

**Larson and Toubro Pvt. Ltd.**

**BY**

Pratima Pore

**Under The Guidance of**

ISD Team Transportation Infrastructure IC

# Contents

<b>1</b>	<b>Abstract and Introduction</b>	<b>4</b>
<b>2</b>	<b>Motivation</b>	<b>5</b>
2.1	Efficiency . . . . .	5
2.2	Accessibility . . . . .	5
2.3	Safety . . . . .	5
2.4	Technological Advancement . . . . .	6
2.5	Customer Experience . . . . .	6
<b>3</b>	<b>OBJECTIVES</b>	<b>7</b>
3.1	User-Friendly Interface . . . . .	7
3.2	Secure Authentication and Registration . . . . .	7
3.3	Scalability and Performance . . . . .	7
3.4	Driver and Fleet Management . . . . .	7
3.5	Analytics and Reporting . . . . .	8
<b>4</b>	<b>Software Requirements Specifications (SRS)</b>	<b>9</b>
4.1	Functional Requirements . . . . .	9
4.1.1	Admin . . . . .	9
4.1.2	Normal User . . . . .	10
4.1.3	Driver . . . . .	11
4.2	Non-functional Requirements . . . . .	11
4.2.1	Usability . . . . .	11
4.2.2	Reliability . . . . .	11
4.2.3	Availability . . . . .	11
4.2.4	Performance . . . . .	12
4.2.5	Security . . . . .	12
4.3	CONSTRAINTS . . . . .	12
4.4	ASSUMPTIONS . . . . .	12
4.5	GLOSSARY . . . . .	13

<b>5</b>	<b>Design</b>	<b>15</b>
5.1	Use Case Diagram . . . . .	15
5.2	Collaboration Diagram . . . . .	17
5.3	Sequence Diagram . . . . .	18
<b>6</b>	<b>Implementation</b>	<b>20</b>
6.1	Frontend Snippets . . . . .	20
6.1.1	Landing Page . . . . .	20
6.1.2	Login Page . . . . .	21
6.1.3	User Pages . . . . .	21
6.1.4	Admin Pages . . . . .	23
6.2	Backend Snippets . . . . .	28
6.2.1	Database . . . . .	28
6.2.2	Backend Routes . . . . .	29
<b>7</b>	<b>Future Scope</b>	<b>30</b>

# Chapter 1

## Abstract and Introduction

The purpose of this project is to develop a ride-hailing application similar to Uber, aiming to provide a convenient, reliable, and efficient transportation solution for Larson and Toubro Employees. This application connects passengers with available drivers through a user-friendly interface, allowing for real-time booking and ride tracking.

The project involved a comprehensive design and development process, utilizing modern technologies such as React for the frontend, Node.js for the backend, and a MongoDB Database system for data management. Key features of the application include user registration and authentication, ride request and acceptance, GPS-based navigation, and a rating system for drivers and passengers.

The application was rigorously tested to ensure reliability, security, and scalability. Performance metrics were evaluated under various conditions to validate the application's ability to handle high traffic volumes and provide a seamless user experience. Feedback from beta testing was incorporated to enhance functionality and user satisfaction.

This project demonstrates the potential of technology to transform urban mobility, offering insights into the challenges and solutions in developing a ride-hailing platform. The outcomes highlight the importance of integrating user-centered design, advanced technology, and robust testing in creating applications that meet modern transportation needs.

# Chapter 2

## Motivation

The motivation behind developing a ride-hailing application like Uber stems from the need to address several critical challenges in urban transportation. Traditional vehicle request services, currently offered in company, often suffer from inefficiencies, lack of transparency, and limited availability, leading to frustration among commuters. This project aims to leverage modern technology to provide a more reliable, convenient, and efficient solution for urban mobility. These include:

### 2.1 Efficiency

Traditional request services often struggle with inefficiencies such as long wait times and inconsistent availability. A ride-hailing app can optimize the matching process between passengers and drivers, ensuring faster response times and more reliable service.

### 2.2 Accessibility

Employees require a transportation solution that is easily accessible and convenient. By developing a ride-hailing application, we can offer users the ability to book rides from their smartphones, reducing the time and effort needed to find transportation and wait for long time.

### 2.3 Safety

Safety is a primary concern for passengers and drivers. The application incorporates features like real-time ride tracking and driver and passenger

ratings to enhance safety and build trust among users.

## **2.4 Technological Advancement**

Developing a ride-hailing application allows for the exploration and implementation of cutting-edge technologies such as GPS navigation, real-time data processing, and secure payment systems. This project serves as an opportunity to innovate and push the boundaries of what is possible in the transportation sector.

## **2.5 Customer Experience**

Last but not least, we're committed to enhancing the overall customer experience. From user-friendly mobile web apps for passengers to intuitive dashboards for administrators, our system prioritizes convenience, reliability, and satisfaction at every touch point. By addressing these challenges and opportunities, our bus management system aims to not only improve the efficiency and effectiveness of bus services but also contribute to creating smarter, more sustainable, and more inclusive cities and communities.

# Chapter 3

## OBJECTIVES

### 3.1 User-Friendly Interface

Design and implement an intuitive and easy-to-use interface for both passengers and drivers, ensuring a seamless experience for booking rides, tracking trips, and managing accounts.

### 3.2 Secure Authentication and Registration

Implement secure user authentication and registration processes to protect user data and ensure a trustworthy user base.

### 3.3 Scalability and Performance

Ensure that the application is scalable to handle a growing number of users and can perform efficiently.

### 3.4 Driver and Fleet Management

Provide tools for drivers to manage their profiles, schedules, and earnings, and for fleet managers to oversee driver performance and vehicle maintenance.

## **3.5 Analytics and Reporting**

Develop a dashboard for administrators to monitor system performance, user requests, and other key metrics, facilitating data-driven decision-making and continuous improvement.



# Chapter 4

## Software Requirements Specifications (SRS)

### 4.1 Functional Requirements

#### 4.1.1 Admin

##### 4.1.1.1 Add and View Vehicles

- The system shall allow administrators to add new vehicles by entering details such as vehicle number, capacity, model, manufacturer, and insurance details.
- Upon addition, the system shall validate the uniqueness of the vehicle number and ensure that all mandatory fields are filled.
- Administrators should also be able to view all the details of vehicles along with their availability for approvals.

##### 4.1.1.2 Add and View Driver

- The system shall allow administrators to add new drivers by entering details such as contact number, name, and license details.
- Upon addition, the system shall validate the uniqueness of the driver and ensure that all mandatory fields are filled.
- Administrators should also be able to view all the details of drivers along with their availability for approvals.

#### **4.1.1.3 Approve Requests**

- The system shall allow administrators to approve requests sent by users.
- Upon approval, the availability fields of vehicle and driver should be marked unavailable respectively.

#### **4.1.1.4 Track Vehicle**

- Administrators shall be able to track distance travelled by vehicle from one location to another.
- The system shall open leaflet map with pointer to the location of the tracked device and estimated position.
- The system shall validate the feasibility of routes based on geographical constraints and regulations for optimized routing.

### **4.1.2 Normal User**

#### **4.1.2.1 New Request**

- End users should be able to request for vehicles by filling a request form.
- The system shall validate the details filled by the user in the form and update the database and also notify the admin about the requests.

#### **4.1.2.2 View Status of requests**

- End users shall the status of the requests they have sent, whether they are approved or not and if approved, the details of vehicle and driver contact details.
- Any changes to status of requests shall trigger notifications to respective users.

#### **4.1.2.3 Track Vehicle**

- End user should be able to live track the vehicle allocated to them by the admin.

### **4.1.3 Driver**

#### **4.1.3.1 View Details of Assignment**

- Driver will be able to view the details of the request they are assigned to, like the vehicle they would drive, pickup location, date and time, and drop location.
- The system shall enforce notification and reminder prevent inconvenience to the end users and drivers.

#### **4.1.3.2 View Location**

- Driver should be able to find the location of end user they are assigned to.

## **4.2 Non-functional Requirements**

### **4.2.1 Usability**

- The user interface shall be intuitive, responsive, and accessible across devices and screen sizes.
- User interactions shall be straightforward, with clear instructions and error messages provided where necessary.

### **4.2.2 Reliability**

- The system shall be highly reliable, with a minimal risk of failures or errors impacting operations.
- Data integrity shall be ensured through robust validation mechanisms and transaction controls.
- The system shall have built-in redundancy and fail over mechanisms to minimize downtime and data loss.

### **4.2.3 Availability**

- The system shall be available 24/7, with minimal scheduled downtime for maintenance purposes.
- Redundant server architecture shall be implemented to ensure high availability and fault tolerance.

- . Load balancing techniques shall be employed to distribute incoming traffic evenly across servers.

#### **4.2.4 Performance**

- The system shall be capable of handling a large number of concurrent users and transactions without significant performance degradation.
- Performance tests shall be conducted regularly to identify and address bottlenecks.
- Load balancing techniques Response times for user interactions shall be within acceptable limits, even during peak usage hours.

#### **4.2.5 Security**

- The system shall implement robust authentication and authorization mechanisms to ensure secure access to sensitive data and functionalities.
- Data encryption shall be employed to protect user information during transmission and storage.
- Regular security audits and vulnerability assessments shall be conducted to identify and address potential security risks.

### **4.3 CONSTRAINTS**

- Technology Stack: The system will be developed using a combination of React.js in the frontend with Tailwind CSS as styling, Node.js and Express.js at the backend along with MongoDB as the Database.
- Regulatory Compliance: The system shall comply with relevant regulations and standards in the transportation industry, including data protection regulations.

### **4.4 ASSUMPTIONS**

- Users will have access to the internet to use the system.
- Vehicles will be equipped with GPS devices for real-time tracking.

- Users will have basic knowledge of using web-based applications and mobile devices.
- Adequate infrastructure for data storage, processing, and backup shall be provided.
- The system shall support integration with external systems for maintaining vehicle schedules and routing information.

## 4.5 GLOSSARY

- Administrator: User with privileged access rights to manage the system.
- End User: Any individual who interacts with the system for requesting or tracking purposes.
- Route: A predefined path along which vehicle travels from pickup location to drop location mentioned by the end user.
- Vehicle: Any vehicle used for transportation of end users.
- Schedule: A plan outlining the pickup and drop times for a specific route.
- Request: A reservation for a vehicle for a specific date and time.
- Fleet: A collection of vehicles managed by a transportation company.
- Tracking: Monitoring the current location and status of a vehicle or driver in real-time.
- Notifications: Alerts sent to users about delays, approvals, or other important updates.
- Reporting: Generation of reports detailing various aspects of the transportation system.
- Analytics: Analysis of key performance indicators to provide insights and trends.
- Data Encryption: Encoding of data to protect it during transmission and storage.

- Authentication: Proving the identity of a user to grant access to the system.
- Authorization: Granting or denying access to specific functionalities based on user roles.
- Load Balancing: Distribution of incoming traffic evenly across servers to ensure optimal performance.
- Redundancy: Duplication of system components to ensure high availability and fault tolerance.
- Fail over: Automatic switchover to a backup system in the event of a failure.
- Localization: Adaptation of the user interface for different languages and regions.
- Integration: Connecting the system with external systems or services for seamless data exchange.
- Performance Testing: Evaluation of the system's ability to handle large numbers of concurrent users and transactions.
- Data Integrity: Ensuring the accuracy and consistency of data within the system.
- Validation: Checking user input and system data to ensure correctness and completeness.
- Transaction Controls: Ensuring the atomicity, consistency, isolation, and durability of database transactions.
- Geographical Constraints: Limitations on route definition based on physical distance, road conditions, and regulations.

# Chapter 5

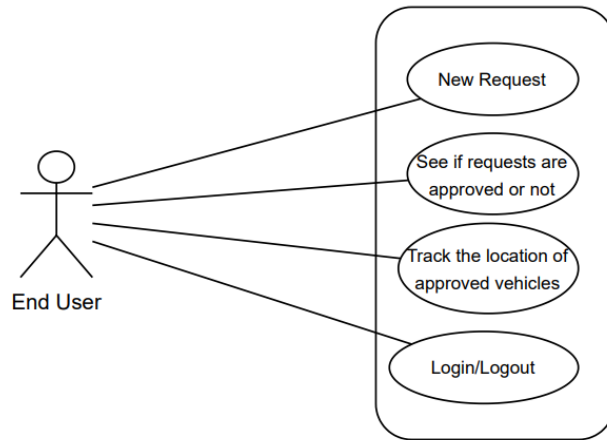
## Design

### 5.1 Use Case Diagram

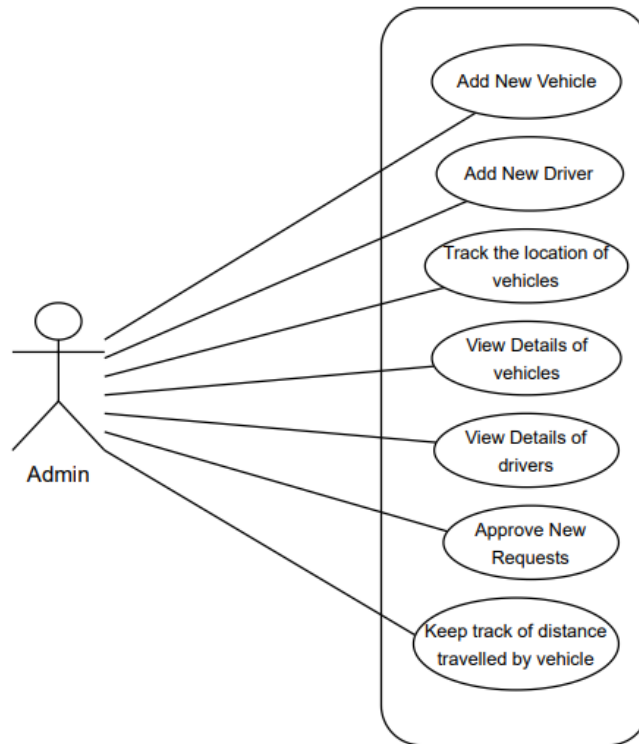
Use Case Diagrams are a type of behavioral diagram in the Unified Modeling Language (UML) that represent the functionality of a system from the user's perspective. They help visualize the interactions between users (actors) and the system, illustrating what the system should do (use cases).

1. Actors: Identify users and external systems.
2. Use Cases: Define system functionalities.
3. Relationships: Connect actors to use cases.
4. System Boundary: Separate system from external entities.
5. Include and Extend: Show dependencies between use cases.
6. Use Case Descriptions: Provide brief summaries.
7. Multiplicity: Indicate actor participation levels.
8. Annotations: Add notes for clarity.
9. Simplicity: Keep the diagram clear and focused.

1. NORMAL USER:



2. ADMIN

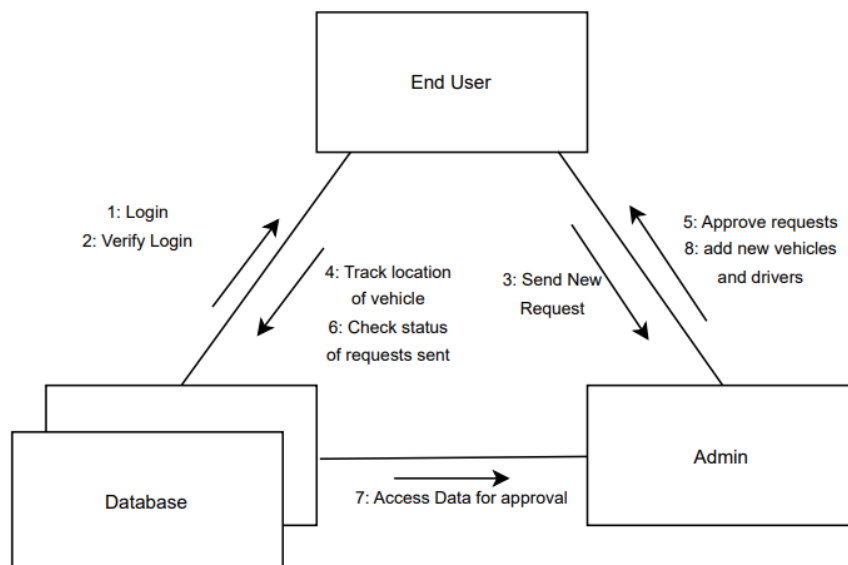




## 5.2 Collaboration Diagram

A Collaboration Diagram, also known as a Communication Diagram, is a type of interaction diagram in UML that shows how objects interact with each other within a system. It focuses on the structural organization of objects and their interactions to accomplish a specific task.

- Objects
  1. Represent instances of classes that interact in the system.
  2. Denoted by rectangles with underlined names.
- Links
  1. Depict the connections between objects, indicating a relationship or association.
  2. Shown as lines connecting objects.
- Messages
  1. Indicate the communication between objects.
  2. Represented by labeled arrows along the links, showing the sequence and direction of the messages.
  3. Labeled with the message name and a sequence number to denote the order of interactions.

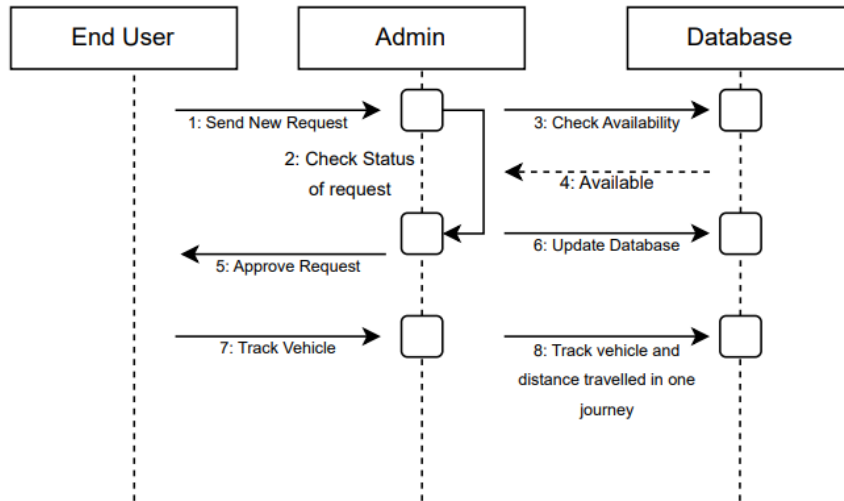


## 5.3 Sequence Diagram

A Sequence Diagram is a type of interaction diagram in UML that shows how objects interact in a particular sequence. It focuses on the chronological order of messages exchanged between objects to carry out a specific functionality or process.

- Lifelines
  1. Represent individual participants or objects in the interaction.
  2. Depicted as vertical dashed lines.
- Actors
  1. Represent external entities that interact with the system.
  2. Shown as stick figures or rectangles.
- Messages
  1. Show communication between lifelines.
  2. Represented by arrows indicating the direction of the message.
  3. Can be synchronous (solid line with filled arrowhead) or asynchronous (solid line with open arrowhead).
- Activation Bars
  1. Indicate the period an object is performing an action or active in the process.
  2. Depicted as thin rectangles on a lifeline.
- Conditions and Loops
  1. Conditional interactions are shown with a guard condition.
  2. Loops are depicted using a loop fragment with a loop keyword and a guard condition.
- Fragments
  1. Combined fragments define control structures such as loops, alternatives, and parallel executions.

2. Types include alt (alternative), opt (optional), loop, par (parallel), etc.



# Chapter 6

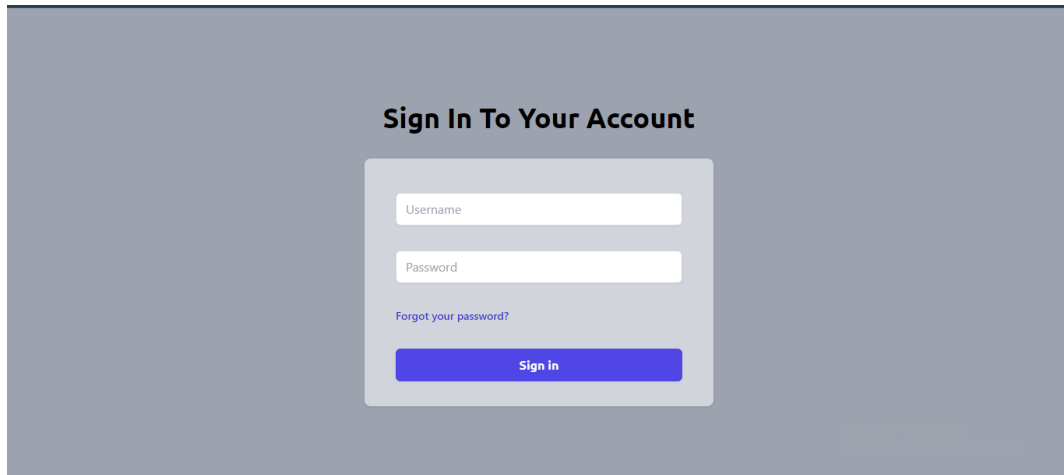
## Implementation

### 6.1 Frontend Snippets

#### 6.1.1 Landing Page



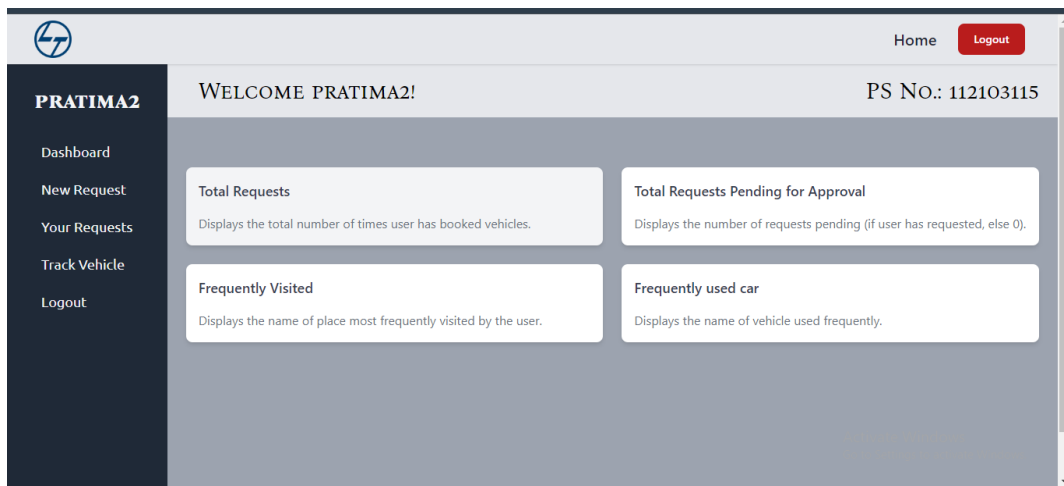
## 6.1.2 Login Page



The image shows a login page with a dark gray background. In the center, there is a light gray rounded rectangle containing the login form. The form has the title "Sign In To Your Account" in bold black text. Below the title, there are two input fields: "Username" and "Password". Below the "Password" field, there is a link "Forgot your password?" in blue text. At the bottom of the form, there is a blue button with the text "Sign in" in white.

## 6.1.3 User Pages

### 6.1.3.1 Dashboard



The image shows a user dashboard for "PRATIMA2". The dashboard has a dark blue sidebar on the left with the following menu items: "Dashboard", "New Request", "Your Requests", "Track Vehicle", and "Logout". The main content area has a light gray header with a "Home" link and a "Logout" button. Below the header, the dashboard is divided into four sections: "Total Requests" (Displays the total number of times user has booked vehicles.), "Total Requests Pending for Approval" (Displays the number of requests pending (if user has requested, else 0).), "Frequently Visited" (Displays the name of place most frequently visited by the user.), and "Frequently used car" (Displays the name of vehicle used frequently.).

6.1.3.2 New Request Form

REQUEST FORM

Name

pratima2

PS Number

112103115

Phone Number

9594066383

Pickup Location

Drop Location

Required From

dd-mm-yyyy

Required To

dd-mm-yyyy

Pickup Time

--:--

Drop Time

--:--

☐ Return

REQUEST

Figure 6.1: Without return

REQUEST FORM

Name

pratima2

PS Number

112103115

Phone Number

9594066383

Pickup Location

Drop Location

Required From

dd-mm-yyyy

Required To

dd-mm-yyyy

Pickup Time

--:--

Drop Time

--:--

☒ Return

Halt Time

Return Pickup Location

Return Drop Location

REQUEST

Figure 6.2: When wants to return

6.1.3.3 Your Request

PRATIMA2

Dashboard

New Request

Your Requests

Track Vehicle

Logout

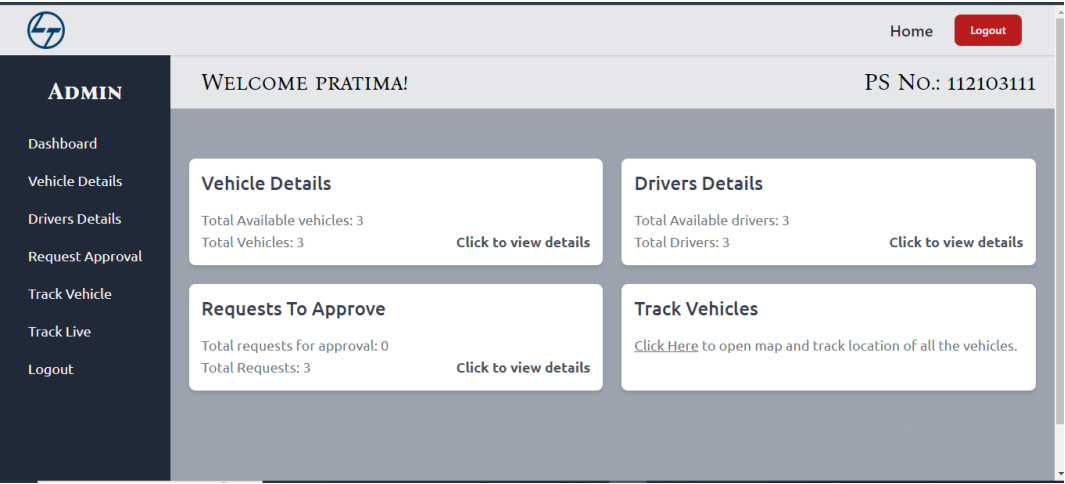
PickUp Location: kharghar  
Drop Location: andheri  
Vehicle Allocated: XUV700  
Vehicle Number: MH01AB1234  
Driver Assigned: Patrick  
Driver Contact: 9876543210  
Approved  
☒ Mark as Completed

PickUp Location: juhu  
Drop Location: andheri  
Vehicle Allocated: Traveller  
Vehicle Number: MH02AB1234  
Driver Assigned: John  
Driver Contact: 9876543210  
Approved  
☒ Mark as Completed

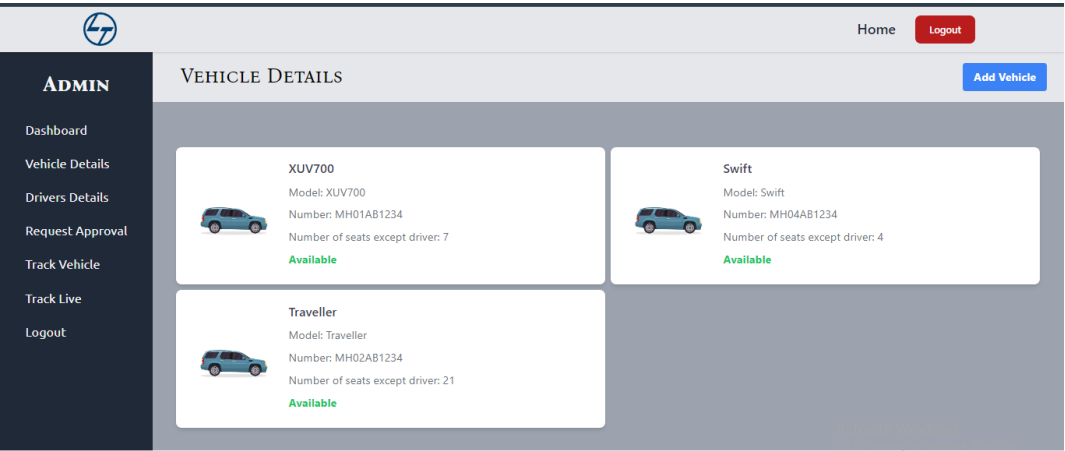
PickUp Location: andheri  
Drop Location: powai  
Vehicle Allocated: Swift  
Vehicle Number: MH04AB1234  
Driver Assigned: John  
Driver Contact: 9876543210  
Approved  
☒ Mark as Completed

# 6.1.4 Admin Pages

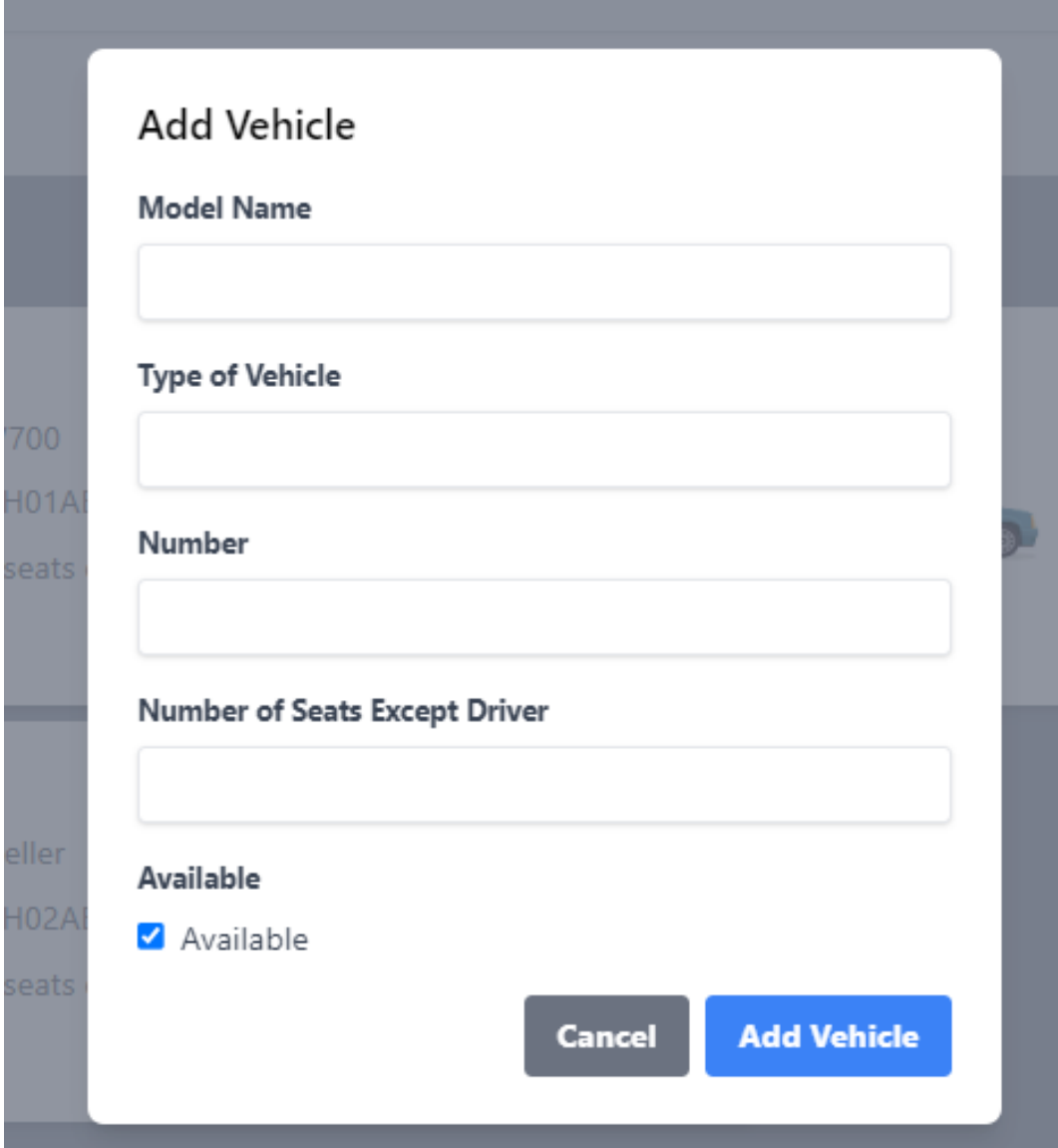
## 6.1.4.1 Dashboard



## 6.1.4.2 View Vehicles



#### 6.1.4.3 Add Vehicles

A screenshot of a web application showing a modal form titled "Add Vehicle". The form is white with rounded corners and is centered on a dark gray background. It contains five input fields: "Model Name", "Type of Vehicle", "Number", and "Number of Seats Except Driver", each with a light gray border. Below these is a checkbox labeled "Available" which is checked. At the bottom right are two buttons: "Cancel" (dark gray) and "Add Vehicle" (blue).

**Add Vehicle**

**Model Name**

**Type of Vehicle**

**Number**

**Number of Seats Except Driver**

**Available**

☒ Available

**Cancel** **Add Vehicle**



#### 6.1.4.4 View Drivers

The screenshot shows a web application interface for an admin dashboard. On the left is a dark sidebar with the 'ADMIN' header and a list of menu items: Dashboard, Vehicle Details, Drivers Details (highlighted), Request Approval, Track Vehicle, Track Live, and Logout. The main content area is titled 'DRIVER DETAILS' and includes an 'Add Driver' button in the top right. It displays three driver profiles in white cards. Each card features a blue circular profile icon, the driver's name, their ID number (9876543210), their license number (12345), and a green 'Available' status label.

Name	Number	License	Status
John	9876543210	12345	Available
James	9876543210	12345	Available
Patrick	9876543210	12345	Available

#### 6.1.4.5 Add Drivers

The screenshot shows a modal form titled 'Add Driver'. It contains four input fields: 'Name', 'Phone Number', and 'License'. Below these is a checkbox labeled 'Available', which is currently checked. At the bottom right of the form are two buttons: 'Cancel' and 'Add Driver'.

**Add Driver**

**Name**

**Phone Number**

**License**

**Available**

☒ Available

**Cancel** **Add Driver**

6.1.4.6 Requests For Approval

REQUESTS FOR APPROVAL

pratima2

PS Number: 112103115

Contact Number: 9594066383

PickUp Location: kharghar

Drop Location: andheri

Vehicle Allocated: XUV700

Vehicle Number: MH01AB1234

Driver Assigned: Patrick

Driver Contact: 9876543210

Approved

Completed

pratima2

PS Number: 112103115

Contact Number: 9594066383

PickUp Location: andheri

Drop Location: powai

Vehicle Allocated: Swift

Vehicle Number: MH04AB1234

Driver Assigned: John

Driver Contact: 9876543210

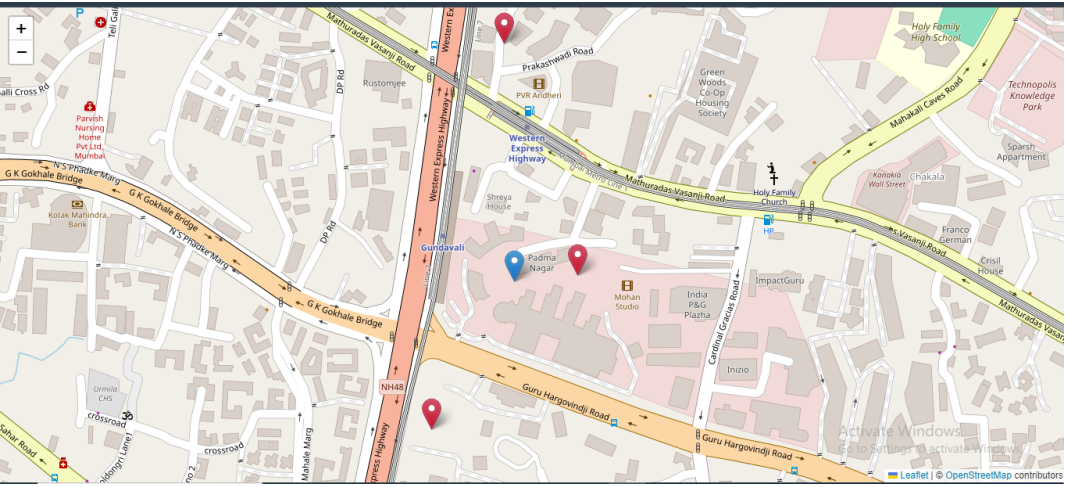
Approved

Completed

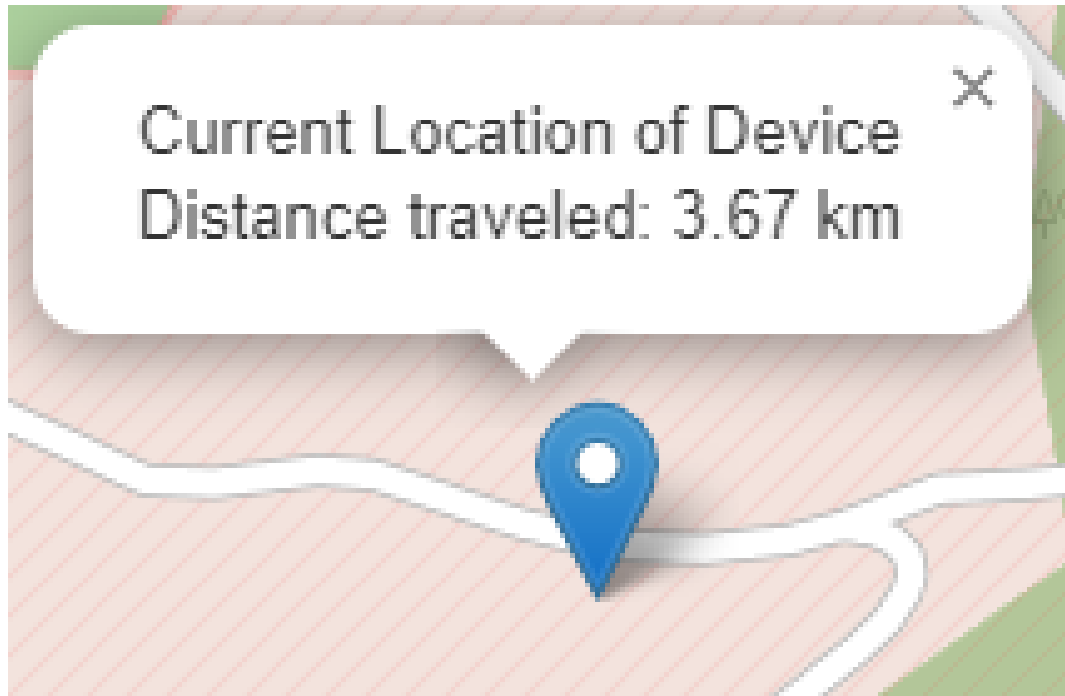
pratima2

PS Number: 112103115

6.1.4.7 Track Vehicle



#### 6.1.4.8 Track Vehicle with Distance travelled



## 6.2 Backend Snippets

### 6.2.1 Database

```
_id: ObjectId('668cc015bb20128ff672d272')
name: "pratima"
psno: "112103111"
phno: "9594066383"
email: "pratima@gmail.com"
password: "123456"
isadmin: true
__v: 0

_id: ObjectId('668cc034bb20128ff672d274')
name: "pratima2"
psno: "112103115"
phno: "9594066383"
email: "pratima2@yahoo.com"
password: "123456"
isadmin: false
__v: 0
```

Figure 6.3: User Database

```
_id: ObjectId('668d0780a0a83cddd1829a3a')
name: "pratima2"
psno: "112103115"
phno: "9594066383"
pickLocation: "kharghar"
dropLocation: "andheri"
return: false
returnPick: ""
returnDrop: ""
haltTime: ""
__v: 0
completed: false

_id: ObjectId('668e22a25fe9c42d643ee183')
name: "pratima2"
psno: "112103115"
phno: "9594066383"
pickLocation: "andheri"
dropLocation: "powai"
return: false
returnPick: ""
```

Figure 6.4: Request Form Database

```
_id: ObjectId('667d41d83bb8c503e01e9c95')
name: "John"
phno: "9876543210"
license: "12345"
available: true
__v: 0

_id: ObjectId('667d42043bb8c503e01e9c97')
name: "James"
phno: "9876543210"
license: "12345"
available: true
__v: 0

_id: ObjectId('668f69e0c97b25cd81c28428')
name: "Patrick"
phno: "9876543210"
license: "12345"
available: true
__v: 0
```

Figure 6.5: Driver Database

```
_id: ObjectId('6683cdc2d80495ee64fce0cf')
model_name: "XUV700"
type_of_vehicle: "SUV"
number: "MH01AB1234"
number_of_seats_except_driver: 7
available: true
__v: 0

_id: ObjectId('6683cde0d80495ee64fce0d1')
model_name: "Swift"
type_of_vehicle: "Sedan"
number: "MH04AB1234"
number_of_seats_except_driver: 4
available: true
__v: 0

_id: ObjectId('6683ce09d80495ee64fce0d3')
model_name: "Traveller"
type_of_vehicle: "Mini Bus"
number: "MH03AB1234"
```

Figure 6.6: Vehicle Database

## 6.2.2 Backend Routes

```
// definition of routes
const loginRouter = require("./routes/login");
const newUserRouter = require("./routes/newUser");
const formRouter = require("./routes/newForm");
const driverRouter = require("./routes/newDriver");
const newVehicleRouter = require("./routes/newVehicle");
const findVehicleRouter = require("./routes/findVehicle");
const findDriverRouter = require("./routes/findDriver");
const requestApprovalRouter = require("./routes/requestApproval");
const getReqApprovalRouter = require("./routes/getReqApproval");
const getUserRequestsRouter = require("./routes/findRequests");
const updateRequestRouter = require("./routes/updateRequest");
const findDetailsRouter = require("./routes/findDetails");
const updateCompletedRequestRouter = require("./routes/updateCompletedRequest");

// routes
app.use(loginRouter);
app.use(newUserRouter);
app.use(formRouter);
app.use(driverRouter);
app.use(newVehicleRouter);
app.use(findVehicleRouter);
app.use(findDriverRouter);
app.use(requestApprovalRouter);
app.use(getReqApprovalRouter);
app.use(getUserRequestsRouter);
app.use(updateRequestRouter);
app.use(findDetailsRouter);
app.use(updateCompletedRequestRouter);
```

# Chapter 7

## Future Scope

1. Allow private vehicle owners to register.
2. Allow sharing of vehicles between more than 2 employees.
3. Use google maps and APIs for live location tracking.
4. Personalized Services using data analytics
5. Integrate with existing transportation services like Uber and OLA for better options.
6. Focus on Environmental Sustainability with electric buses and renewable energy
7. Implementation of Health and Safety Measures
8. Data Sharing and Collaboration for interoperability
9. Continued Innovation and Adaptation to meet evolving needs.