

## Lloyd Institute of Engineering & Technology, Greater Noida

**A Project Report On**

**“Transportation buddy**”

Submitted in partial fulfillment of the requirement for the award of the degree of Master of Computer Applications

By

### MD MUSARRAF PARWEZ

2301530140063

### Under the Supervision of

### Mr. Yash Sharma

### DR. A P J ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

(2024-25)

PROJECT REPORT ON

“Transportation buddy”

(KCA-451)

Session-2024-2025

Department of Master of Computer Applications (MCA)



**Submitted to: Submitted by:**

Mr. Yash Sharma MD MUSARRAF PARWEZ Roll no: 2301530140063

Semester: IV

## Lloyd Institute of Engineering & Technology, Greater Noida

Plot No.3, Knowledge Park 2, Greater Noida, Uttar Pardesh 201306

# CERTIFICATE OF ORIGINALITY

I hereby declare that my Project titled **Transportation Buddy** submitted to **Dr. APJ ABDUL KALAM TECHNICAL UNIVERSITY, Lucknow** for the partial fulfillment of the degree of Master of Computer Applications Session 2024-2025 from **Lloyd Institute of Engineering & Technology, Greater Noida** has not previously formed the basis for the award of any other degree, diploma or other title.

Place: Greater Noida (Delhi NCR) **Signature**:

Date: Student Name**: MD MUSARRAF PARWEZ**

# CERTIFICATE OF ACCEPTANCE

This is to certify that the project entitled, **Transportation Buddy** submitted by **Md Musarraf Parwez** a student of **Lloyd Institute of Engineering & Technology, Greater Noida** in partial fulfillment for the award of **Master of Computer Applications** affiliated to **Dr. APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW** during the year 2024-25. It is certified that all corrections, suggestions indicated as per Internal Assessment have been incorporate in the project.

To the best of our knowledge, the work embodied in this report is original and has not been submitted to any other degree of discipline. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

**[Sign and Name of Internal Guide] [Sign of External Examiner]**

# DECLARATION

I hereby declare that the project report entitled **Transportation Buddy** submitted by us to **Lloyd Institute of Engineering & Technology, Greater Noida** is the partial requirement for the award of the degree of the Master of Computer Application is a record of bonafide project work carried out by us under the guidance of “**Mr. Yash Sharma**”. I further declare that the work reported in this project has not been submitted and will not be submitted either in part or in full for the award.

Place: Greater Noida

Date: 24 May 2025 Signature of student

# ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success. We owe a great many thanks to great many people, who assisted and helped me during and till the end of the project.

We would like to express our gratitude towards **Gaurav Bhatia, Head of Department–Computer Science & Information Systems, Lloyd Institute of Engineering & Technology Greater Noida**

**,UP**, for his guidelines and scholarly encouragement.

We are indebted to **Mr. Yash Sharma – Assistant Professor, Computer Science & Information Systems** of **Lloyd Institute of Engineering & Technology Greater Noida ,UP** for their valuable comments and suggestions that have helped us to make it a success. The valuable and fruitful discussion with them was of immense help without which it would have been difficult to present this project in live.

We gratefully acknowledge and express our gratitude to all faculty members and friends who supported us in preparing this project report.

Finally, this acknowledgement is incomplete without extending our deepest – felt thanks and gratitude towards our parents whose moral support has been the source of nourishment for us at each stage of our life.

**Student Name: MD MUSARRAF PARWEZ Roll Number : 2301530140063**

# ABSTRACT

In today’s fast-paced world, the need for reliable and efficient transportation services is paramount for individuals, businesses, and industries. Traditional manual transport booking systems are plagued by inefficiencies such as paperwork errors, communication delays, lack of real-time tracking, and limited accessibility. The proposed project, **Transportation Buddy**, aims to address these challenges by developing a comprehensive and user-friendly web-based platform for booking and managing transportation vehicles for both goods and passengers.

The system streamlines the process of transport vehicle scheduling, driver allotment, booking confirmation, and real-time vehicle tracking. It offers modules for client registration, vehicle management, employee and driver records, online payment transactions, query resolution, and ratings and reviews. Admins can manage the entire ecosystem, while clients can easily search and book vehicles that meet their specific needs. Vehicle tracking through GPS integration enhances safety, transparency, and trust.

Developed using Java (JDK 17) with Eclipse IDE and MySQL as the backend database, the project is designed to ensure high availability, scalability, and ease of use. The architecture follows a modular and layered approach for better maintainability. Features like notification posting, search filters, and direct messaging contribute to a smooth user experience.

This project not only replaces outdated manual processes but also paves the way for future enhancements like mobile app integration, video conferencing with drivers, and AI-based vehicle recommendations. With a vision to promote smart, secure, and convenient transportation, *Transportation Buddy* stands as a modern solution for the dynamic logistics needs of today’s world.

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| DECLARATION |  |
| CERTIFICATE |  |
| ACKNOWLEDGEMENT |  |
| ABSTRACT |  |
| LIST OF FIGURES |  |
| LIST OF SYMBOLS AND ABBREVIATIONS |  |
| CHAPTER 1: INTRODUCTION   * 1. Introduction of the Project   2. Objective and Scope of the Project      1. Objective of the Project      2. Scope of the Project |  |
| CHAPTER 2: LITERATURE REVIEW   * 1. Existing Transport Booking System   2. Manual Booking System   3. Conclusion of Review |  |
| CHAPTER 3: DESIGN OF PROJECT MODEL   * 1. System Architecture   2. Use Case-Based Modular Design   3. Flow of Data   4. Data Flow Diagram (DFD)      1. Level 0 DFD      2. Level 1 DFD   5. Entity Relationship Diagram (ERD)   6. Grantt Chart   7. Agile Development Model   8. UI/UX Diagram |  |
| CHAPTER 4: EXPERIMENTS, SIMULATION & TESTING   * 1. Methodology   2. Hardware & Software used   3. Testing Technology used |  |

|  |  |
| --- | --- |
| CHAPTER 5: RESULT AND DISCUSSION |  |
| CHAPTER 6: CONCLUSION AND FUTURE SCOPE   * 1. Conclusion   2. Future Scope |  |
| *References* |  |
| BIOGRAPHY |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** |  |
| 1 | Use Case Diagram |  |
| 2 | Level 0 |  |
| 3 | Level 1 |  |
| 4 | Entity Relationship Diagram |  |
| 5 | Grantt Chart |  |
| 6 | Agile Model Diagram |  |

**LIST OF SYMBOLS AND ABBREVIATIONS**

|  |  |
| --- | --- |
| Symbol/Abbreviation | **Description** |
| DFD | Data Flow Diagram |
| IDE | Integerated Development Environment |
| CRUD | Create,Read,Update,Delete |
| SQL | Structured Query Language |
| ERD | Entity Relationship Diagram |
| DAO | Data Access Object |

**CHAPTER 1**

**INTRODUCTION**

#### Introduction of the Project

Transportation is a critical component of modern society, playing a vital role in the movement of people, goods, and services. With the growing demand for efficient, reliable, and accessible transportation systems, there is a need for platforms that can simplify and streamline the process of booking and managing transport vehicles. Traditional methods of transport booking often involve manual efforts, phone calls, and paperwork, which can lead to inefficiencies, delays, and errors.

**Transportation Buddy** is a web-based application developed to address these challenges by offering a smart, convenient, and user-friendly platform for booking vehicles for both personal and commercial use. The system is designed to serve a wide range of users, including individuals, businesses, and vendors, who require timely and cost-effective transportation solutions.

This project enables users to choose from a variety of vehicles such as cars, trucks, and buses, based on their specific requirements like size, capacity, and travel distance. It incorporates real- time vehicle tracking, driver allocation, online payment, and digital receipts, providing a fully automated experience. Additionally, it allows users to rate and review the service, helping improve transparency and trust.

The platform also benefits transportation companies by offering efficient vehicle and driver management, scheduling tools, and communication modules. With features like real-time tracking and secure online payments, **Transportation Buddy** aims to enhance both user satisfaction and operational efficiency.

This introduction outlines the motivation behind the project and sets the foundation for understanding the need for a digital transportation management solution that meets the expectations of today’s fast-paced, tech-driven world.

#### Objective and Scope of the Project

* + 1. **Objective of the Project:**

The primary objective of the **Transportation Buddy** project is to develop a centralized and efficient online platform that simplifies the booking and management of transportation vehicles for individuals and businesses. The application aims to reduce the time, cost, and manual effort involved in traditional transport systems and replace them with an automated, secure, and real- time solution.

The core objectives are:

* + - * To provide a user-friendly interface for booking vehicles based on specific needs (type, size, location, and timing).
      * To automate the process of vehicle allocation, driver assignment, and scheduling.
      * To enable real-time tracking of vehicles for transparency and security.
      * To offer secure online payment options and generate digital receipts.
      * To store and manage details of vehicles, customers, and employees effectively.
      * To allow users to rate and review services to ensure accountability and continuous improvement.
      * To establish smooth communication between users, drivers, and administrators via messaging or video conferencing.
    1. **Scope of the Project:**

The scope of the *Transportation Buddy* project includes a comprehensive set of features and functionalities that support both goods and passenger transport services. It is designed to be scalable, customizable, and accessible across web and mobile platforms.

Key aspects covered within the project scope are:

* + - * **Web-Based System:** Accessible via browsers, with the potential for mobile app integration.
      * **User Roles:** Supports multiple user types – Admin, Clients (individuals or companies), and Employees (drivers, staff).
      * **Vehicle Management:** Handles data for different types of vehicles, including availability, capacity, and tracking information.
      * **Booking & Scheduling:** Enables easy booking with scheduling features to optimize resource utilization.
      * **Tracking System:** Integrated GPS functionality to monitor vehicle movement in real- time.
      * **Payment Gateway:** Secure payment processing with e-bill generation and transaction history.
      * **Feedback System:** Users can provide feedback and ratings to improve service quality.
      * **Authentication & Security:** Ensures secure logins and access control for different roles.

# CHAPTER 2 LITERATURE REVIEW

The growth of digital technologies and web applications has significantly transformed the transportation and logistics industry. As organizations and individuals increasingly demand faster, more reliable, and transparent services, the need for automated vehicle booking and management platforms has become apparent. This literature review explores existing systems, research studies, and technological developments relevant to the design and implementation of the *Transportation Buddy* project.

* 1. **Existing Transport Booking Systems**

Online platforms like **Uber**, **Ola**, **RedBus**, and **Porter** have already established themselves in the domain of passenger and goods transportation. These applications offer various features such as real-time booking, fare estimation, route tracking, and driver ratings. However, most of these platforms focus on either passenger transport or goods delivery — rarely offering a unified solution for both. Moreover, the majority of such systems are app-based and lack comprehensive web portal interfaces for organizational use, especially for bulk or long-term bookings.

* 1. **Manual Booking Systems and Limitations**

Several transport service providers in semi-urban and rural areas still rely on manual booking systems involving paper records and phone calls. Studies have shown that manual systems are prone to errors, miscommunication, delays, and increased operational costs. In particular, a report by the **International Journal of Transportation Engineering and Technology (IJTET)** highlights that manual transport systems cannot cope with high demand and lack flexibility in managing booking schedules and driver assignments.

* 1. **Integration of Real-Time Technologies**

Modern transportation platforms are increasingly integrating **GPS tracking**, **cloud databases**, and **automated scheduling** to enhance user experience and operational efficiency. These technologies ensure live vehicle tracking, route optimization, and immediate updates for customers. According to **IEEE research** on intelligent transport systems, real-time tracking and communication significantly reduce delays, improve customer satisfaction, and enhance overall system transparency.

* 1. **Communication and Security Features**

Recent advancements also stress the importance of secure communication channels and real-time customer support. While some systems provide in-app messaging, very few platforms include features like **video conferencing with drivers** or **customer-admin chat interfaces**. Security and user trust are further strengthened through **OTP verification**, **encrypted payments**, and detailed driver profiles — features which *Transportation Buddy* aims to include.

* 1. **Identified Gaps and Proposed Solutions**

Despite the availability of various systems, there is a clear **gap** in unified platforms that cater to both passenger and goods transportation. Additionally, most systems lack the flexibility to handle dynamic pricing, vehicle scheduling, employee management, and bulk bookings by corporate clients. There is also limited support for **admin-controlled vehicle monitoring**, multi- role user access, and **custom notification systems**.

*Transportation Buddy* addresses these gaps by offering:

* + - A web-based platform accessible to both individuals and organizations.
    - Modules for admin, client, and driver roles.
    - Real-time vehicle tracking and booking management.
    - Rating, review, and feedback integration.
    - Online payment and digital invoice generation.
  1. **Conclusion of Review**

From the literature and existing systems surveyed, it is evident that while technology has made notable strides in transportation management, many challenges remain — especially in unified system development, role-based access control, and customer support. *Transportation Buddy* fills these voids by offering an inclusive, flexible, and efficient online platform for managing all aspects of transportation—both passengers and goods—in a secure and user-friendly manner.

# CHAPTER 3

**DESIGN OF PROJECT MODEL**

The design of the *Transportation Buddy* project follows a structured and modular approach that ensures scalability, maintainability, and ease of integration. The system is divided into multiple interconnected components, each responsible for handling a specific part of the transport booking and management process. The design includes architectural planning, user interface layouts, data flow mechanisms, and interaction diagrams to describe how each part of the system functions.

1. **System Architecture**

The overall architecture is based on a **three-tier architecture** consisting of the following layers:

* + **PresentationLayer(Front-End)**:

This is the user-facing layer developed using HTML, CSS, and JavaScript integrated into JSP or similar technology through the Eclipse IDE. It allows clients, admins, and drivers to interact with the system through login, booking, viewing, or updating data.

* + **BusinessLogicLayer(Back-End)**:

Developed using Java (JDK 17), this layer handles all the core functionality, including user authentication, booking logic, scheduling, driver allotment, vehicle management, and payment processing.

* + **DataLayer(Database)**:

This layer is managed using MySQL and is responsible for storing and retrieving persistent data such as vehicle information, user profiles, booking records, transaction logs, ratings, and feedback.

1. **Use Case-Based Modular Design**

The project is designed using a **modular approach**. Each module handles a specific function and communicates with other modules when needed:

* + **Client Module** – Registration, login, booking, payment, and tracking.
  + **Admin Module** – Manage vehicles, drivers, schedules, view feedback, post notifications.
  + **Driver Module** – View assigned schedules, route info, and update delivery status.
  + **Vehicle Management Module** – Handles addition, deletion, and updating of vehicle details.

The design of the *Transportation Buddy* project follows a structured and modular approach that ensures scalability, maintainability, and ease of integration. The system is divided into multiple interconnected components, each responsible for handling a specific part of the transport booking and management process. The design includes architectural planning, user interface layouts, data flow mechanisms, and interaction diagrams to describe how each part of the system functions.

1. **System Architecture**

The overall architecture is based on a **three-tier architecture** consisting of the following layers:

* + **PresentationLayer(Front-End)**:

This is the user-facing layer developed using HTML, CSS, and JavaScript integrated into JSP or similar technology through the Eclipse IDE. It allows clients, admins, and drivers to interact with the system through login, booking, viewing, or updating data.

* + **BusinessLogicLayer(Back-End)**:

Developed using Java (JDK 17), this layer handles all the core functionality, including user authentication, booking logic, scheduling, driver allotment, vehicle management, and payment processing.

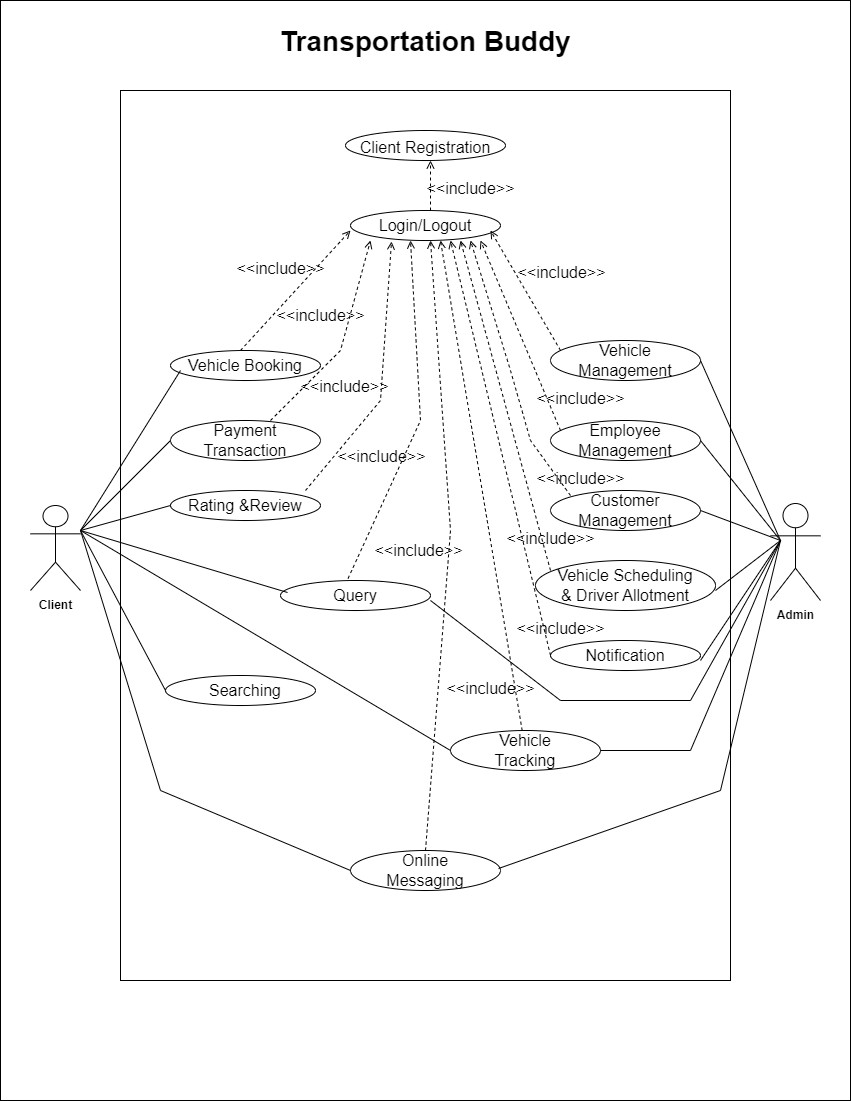
* + **DataLayer(Database)**:

This layer is managed using MySQL and is responsible for storing and retrieving persistent data such as vehicle information, user profiles, booking records, transaction logs, ratings, and feedback.

1. **Use Case-Based Modular Design**

The project is designed using a **modular approach**. Each module handles a specific function and communicates with other modules when needed:

* + **Client Module** – Registration, login, booking, payment, and tracking.
  + **Admin Module** – Manage vehicles, drivers, schedules, view feedback, post notifications.
  + **Driver Module** – View assigned schedules, route info, and update delivery status.
  + **Vehicle Management Module** – Handles addition, deletion, and updating of vehicle details.
  + **Query & Messaging Module** – Manages client queries and portal communication.
  + **Review & Rating Module** – Allows users to provide feedback on the service and drivers.



1. **Flow of Data**

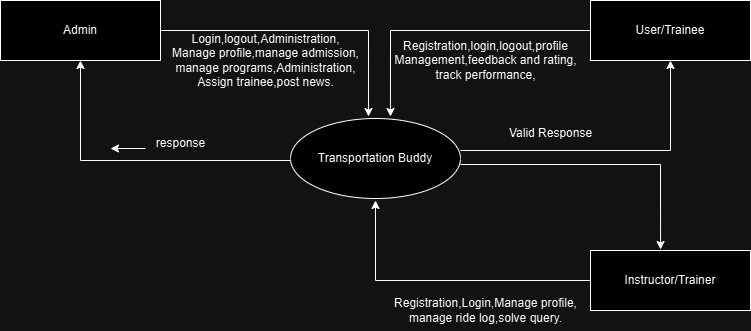
The flow of data between modules is managed efficiently using controllers and services:

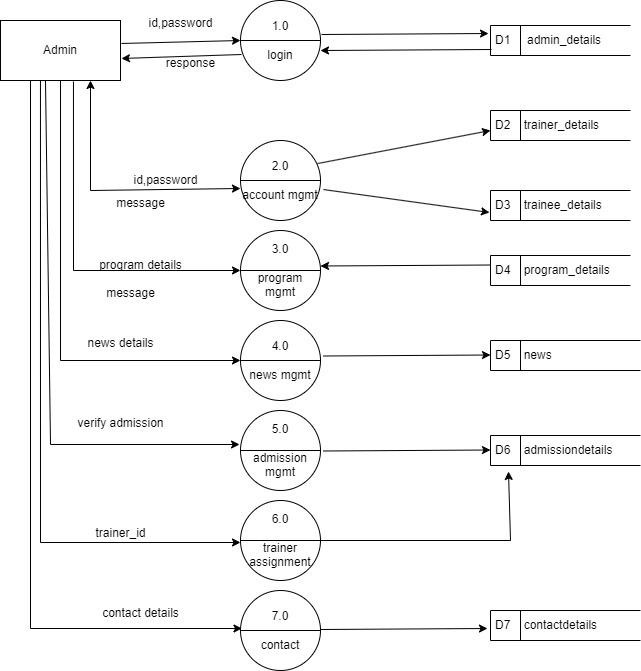
* + When a user books a vehicle, the front-end sends data to the backend.
  + The business logic layer checks availability, updates the schedule, assigns a driver, and stores the booking in the database.
  + Tracking data is fetched from the GPS module and shown to users through the frontend.
  + Ratings and reviews are submitted post-trip and stored for future viewing.

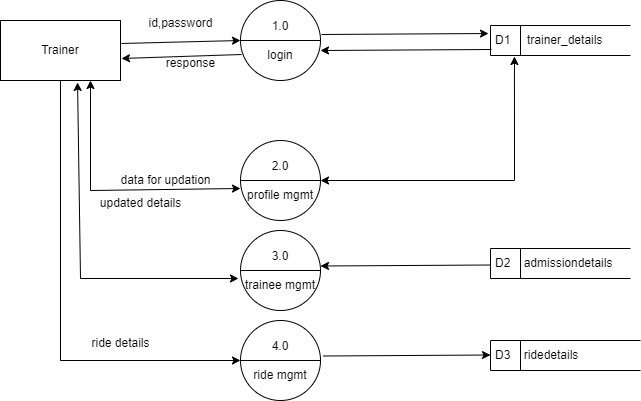
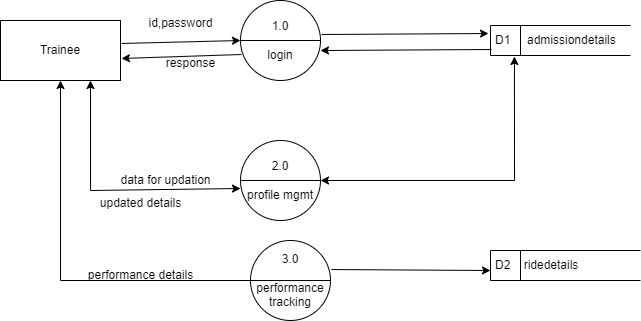
1. **Data Flow Diagram (DFD)**

The Data Flow Diagram (DFD) outlines how data moves throughout the system:

* + **Level 0** – High-level overview of interaction between the client, system, and admin.
  + **Level 1** – Details of booking, payment, and feedback processing.
  + **Level 2** – Includes driver allotment, vehicle updates, and notification broadcasting.

Level-0:

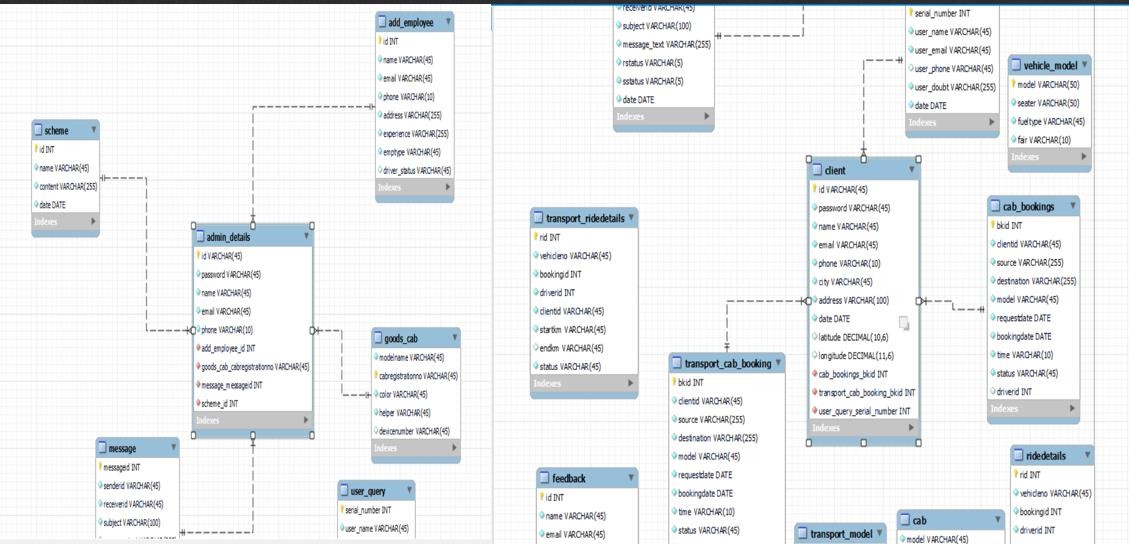
**Level-1:**

******Level-2:**

The ER diagram includes the following main entities and their relationships:

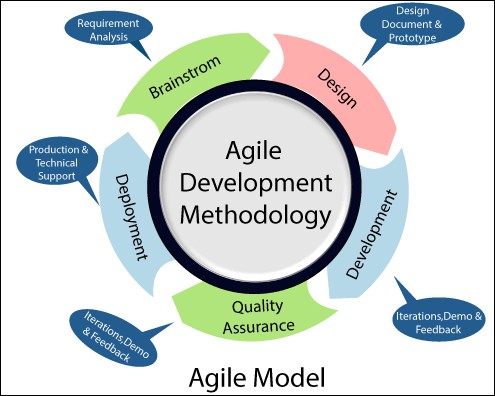
* + **Users** (Admin, Client, Driver)
  + **Vehicles**
  + **Bookings**
  + **Schedules**
  + **Payments**
  + **Ratings & Reviews**
  + **Queries**

Each entity has clearly defined attributes and relationships (e.g., one driver can be assigned to many bookings, one client can post many ratings, etc.).



**Grantt Chart**

**AGILE DEVELOPMENT MODEL**

****

1. **User Interface Design**

The interface is designed to be simple, responsive, and intuitive. Key UI components include:

* + **Login & Registration Forms**
  + **Booking Dashboard**
  + **Vehicle Search Page**
  + **Admin Control Panel**
  + **Driver Schedule View**
  + **Real-Time Tracking Map**

# CHAPTER 4 EXPERIMENTS, SIMULATION & TESTING

* 1. **Methodology**

The methodology adopted for the development of the **Transportation Buddy** web application is based on the **Agile Software Development Life Cycle (SDLC)** model. This approach allows for iterative development, continuous feedback, and flexibility in accommodating changes during the development process.

The development process is divided into the following phases:

* + - **Requirement Analysis**: Gather detailed functional and non-functional requirements from potential users such as individuals, vendors, and businesses. Identify the drawbacks of the manual booking system and define objectives for the online platform.
    - **System Design**: Design the architecture of the system, database structure using MySQL, and create a responsive and user-friendly interface. Define clear modules for each user role (Admin and Client).
    - **Module Development**: Develop each module such as registration, login, vehicle management, driver allotment, booking, payments, and tracking separately using Java in Eclipse IDE. All modules are integrated once developed.
    - **Testing**: Perform testing at various levels—unit testing, integration testing, and system testing—to ensure accuracy, security, and usability.
    - **Deployment**: Deploy the fully integrated application on a suitable server environment. Verify all features such as real-time vehicle tracking, online payments, and messaging are functioning correctly.
    - **Maintenance and Future Enhancements**: Collect user feedback post-deployment and provide regular updates, performance monitoring, and new feature integration like mobile app support and AI-based suggestions.
  1. **Hardware & Software Used Hardware Requirements:**
     + **Processor**: Dual Core or above
     + **RAM**: 4 GB or higher
     + **Hard Disk**: 500 GB
     + **Display**: High-resolution screen

**Software Requirements:**

* + - **IDE**: Eclipse IDE
    - **Programming Language**: Java (JDK 17)
    - **Database**: MySQL
    - **Operating System**: Windows 10 or later
    - **Web Technologies**: HTML, CSS, JavaScript (for UI design)

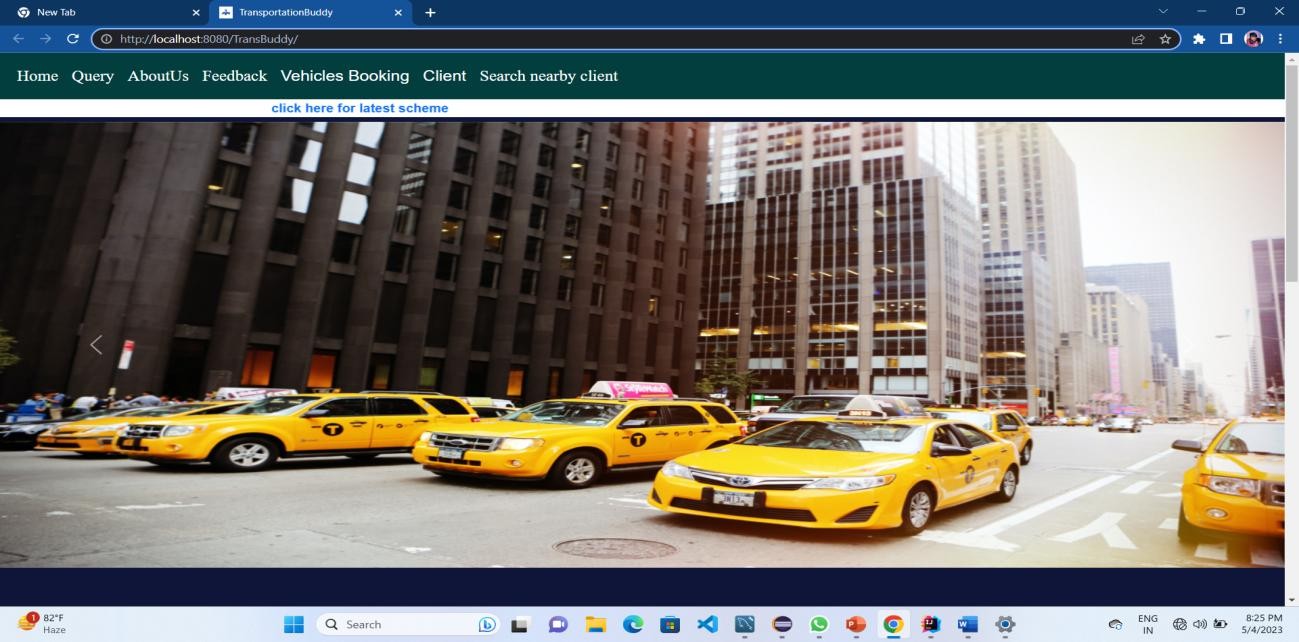
These tools are chosen for their efficiency, ease of use, and strong community support, making them ideal for developing a robust web-based transport booking system.

### Testing Technology Used

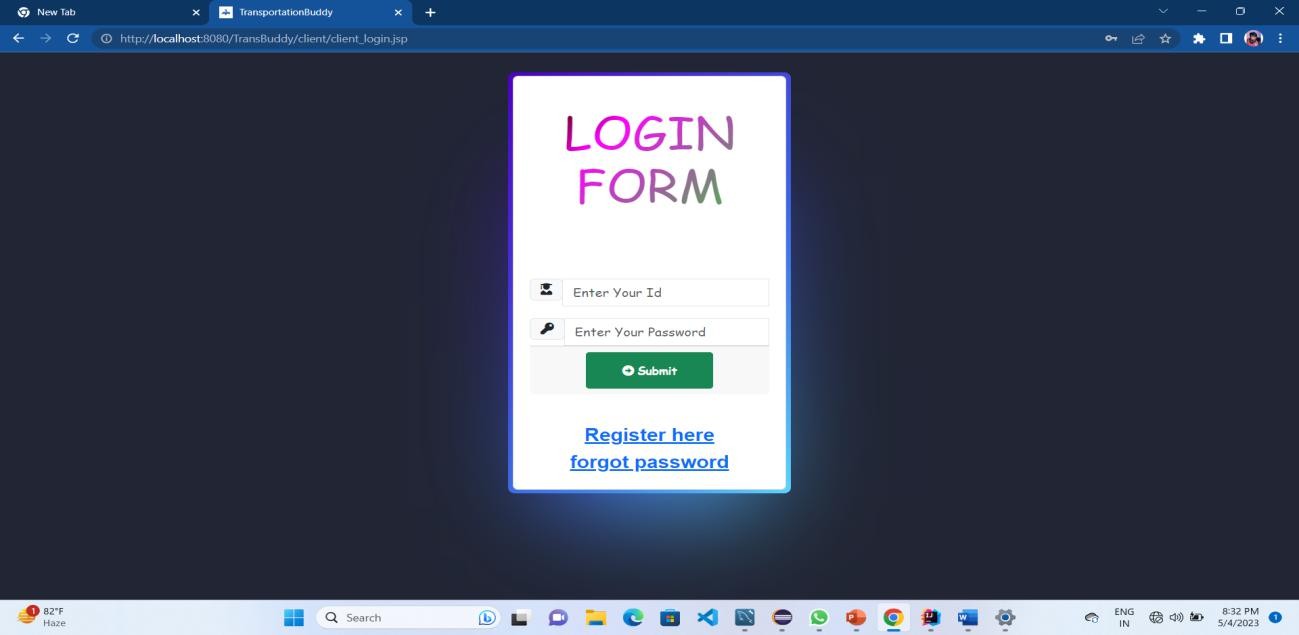
To ensure the reliability and performance of the **Transportation Buddy** application, various testing techniques and tools are used:

* **Unit Testing**: Performed on individual modules like login, booking, and payment to validate internal logic and functionality.
* **Integration Testing**: Used to check whether different modules such as vehicle booking and driver allotment work together correctly.
* **System Testing**: Comprehensive testing of the entire system using real-time scenarios to evaluate overall performance, usability, and correctness.
* **Manual Testing**: Most features are tested manually using different test cases to identify bugs and unexpected behaviors.
* **Database Testing**: MySQL database operations are tested to verify data integrity, accurate storage, and retrieval.
* **Security Testing**: Ensures that data access is restricted based on roles and that sensitive information (e.g., user credentials, payment details) is handled securely.

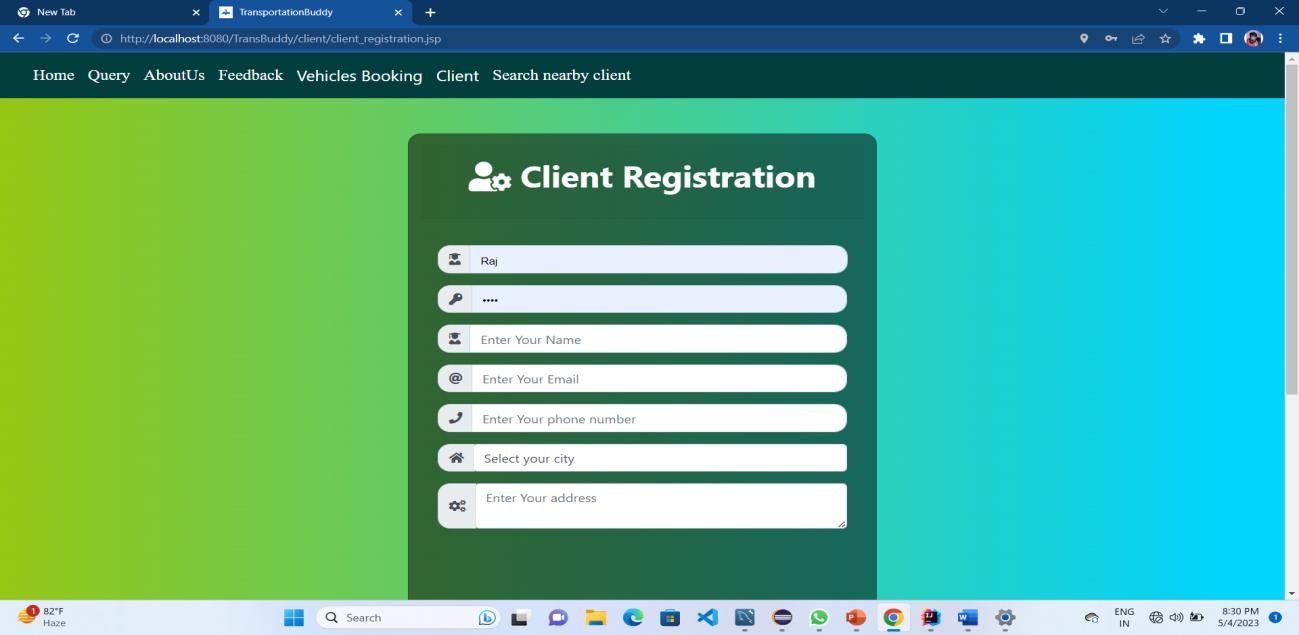
## Screenshots

Dashboard

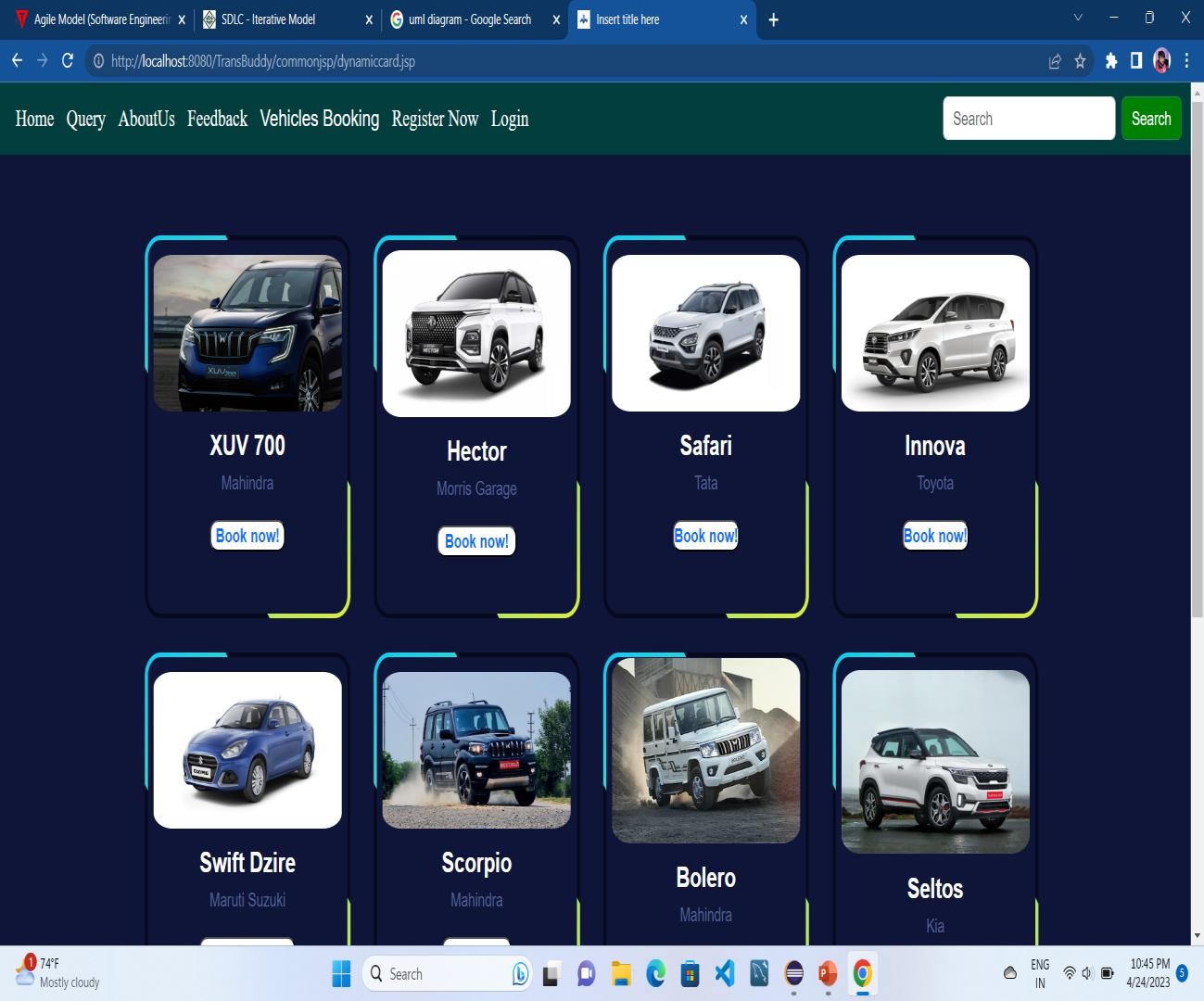
Admin Login:

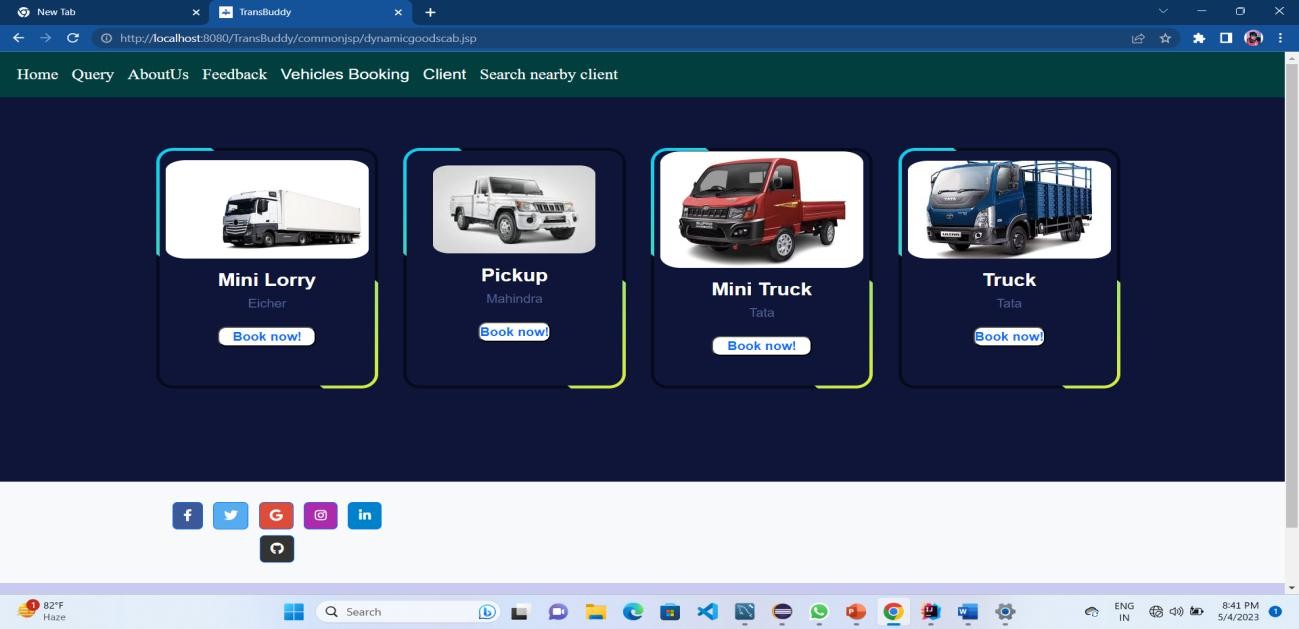


Registration Page:



Cab Booking:

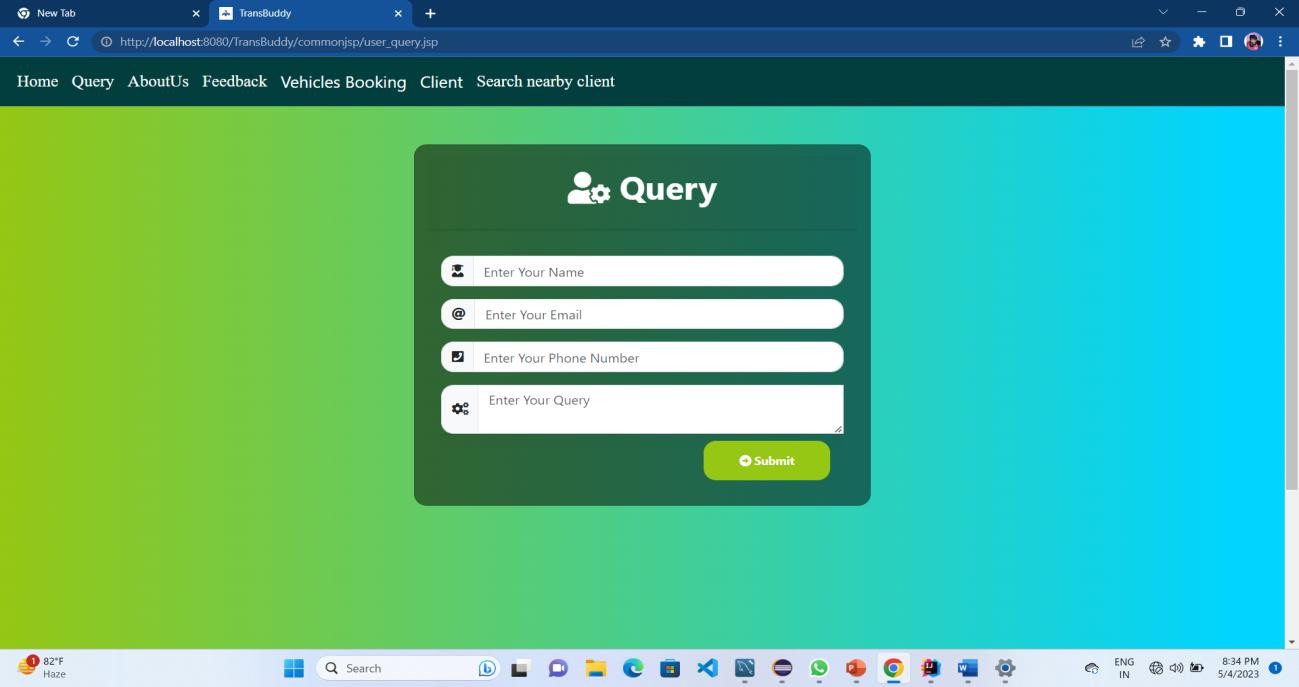


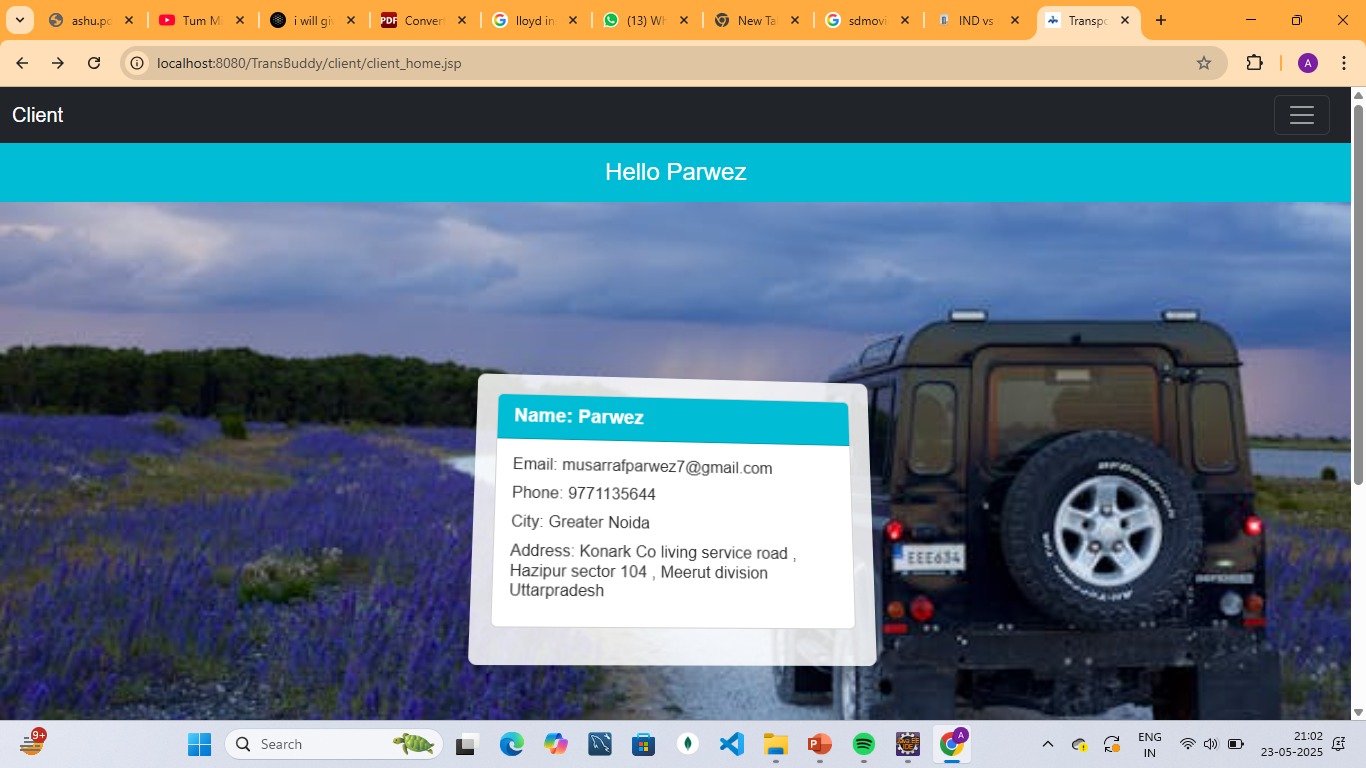
Transport Cab booking:

Riding Status:

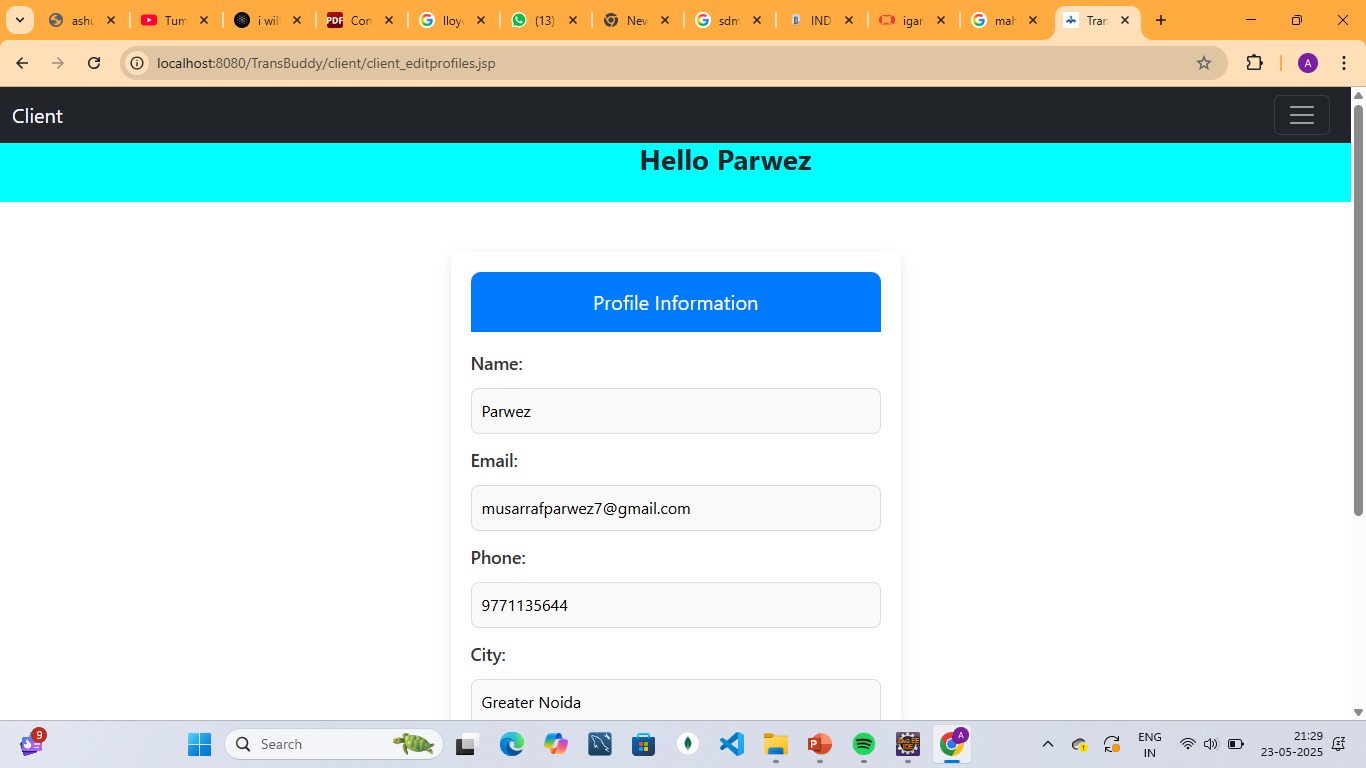


User Query:

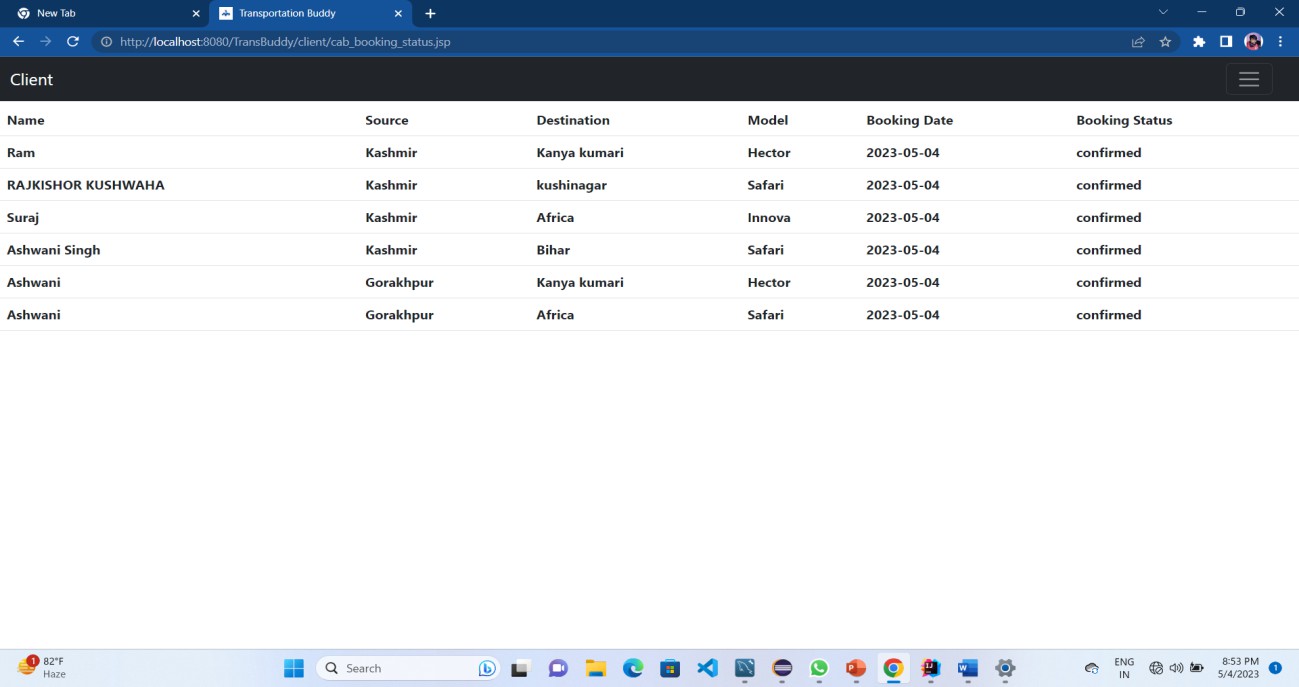




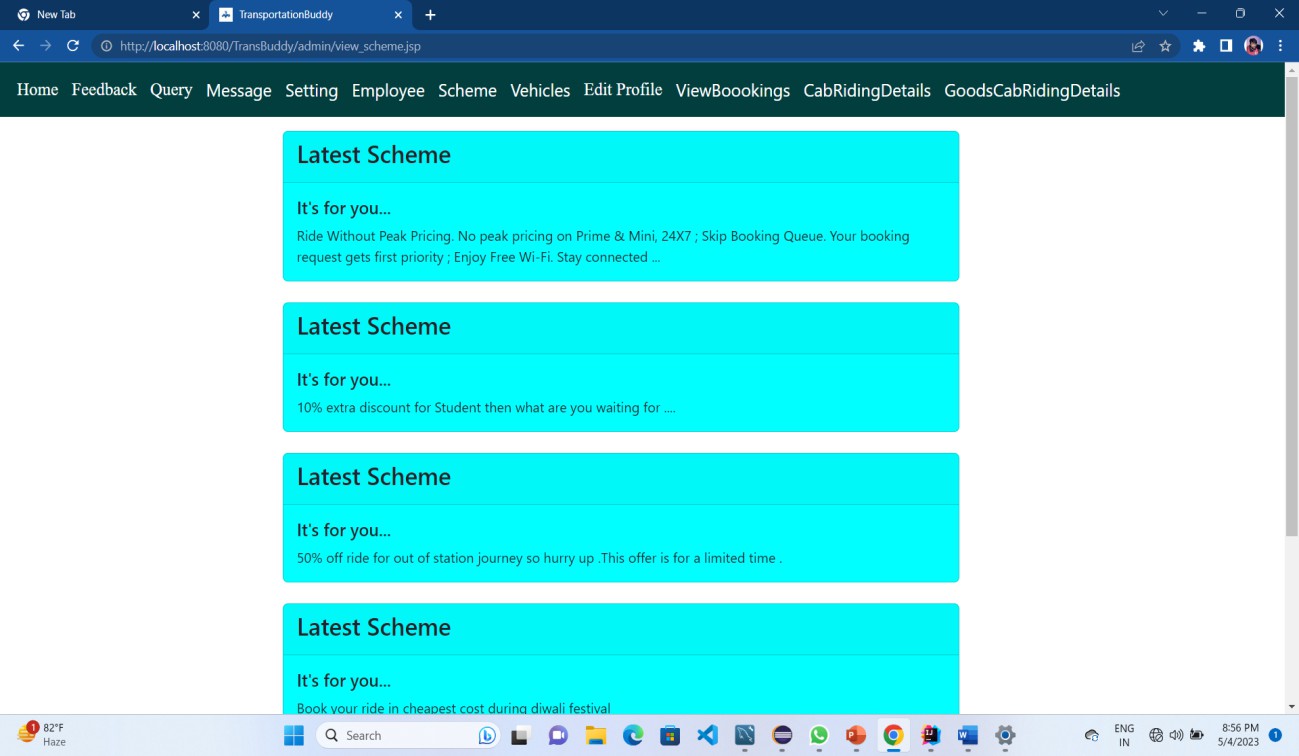
About Us:



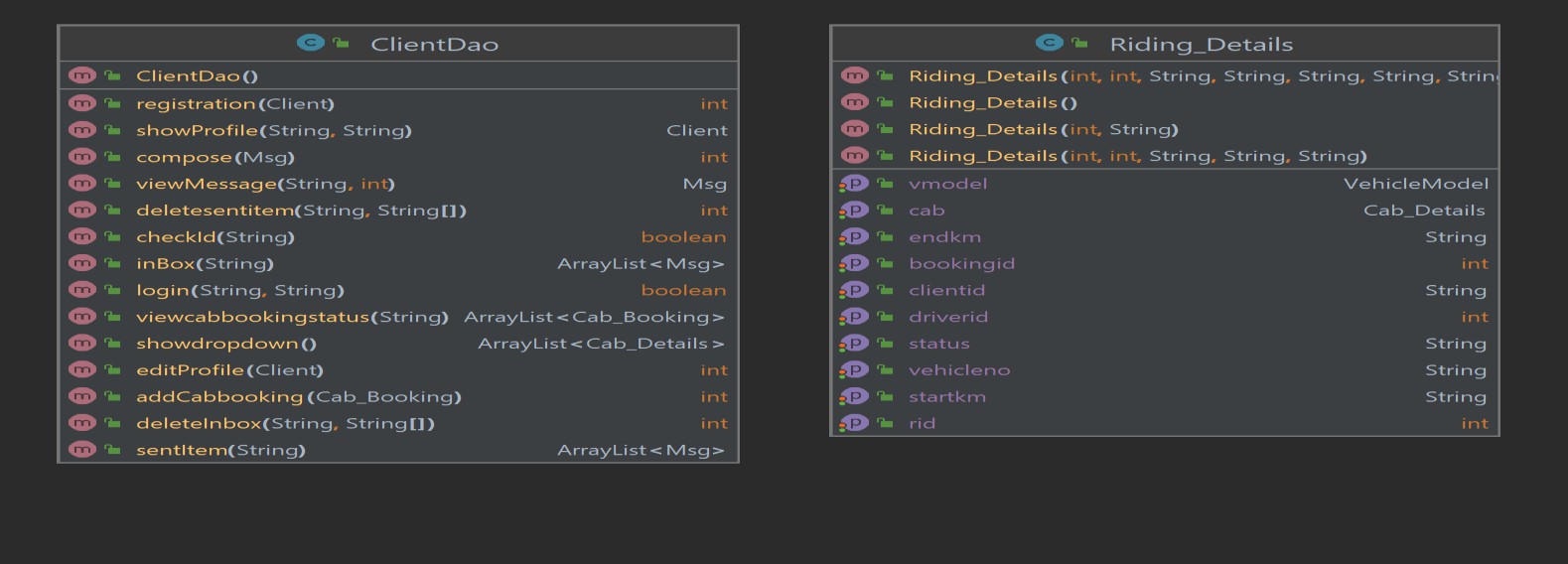
Booking Status :

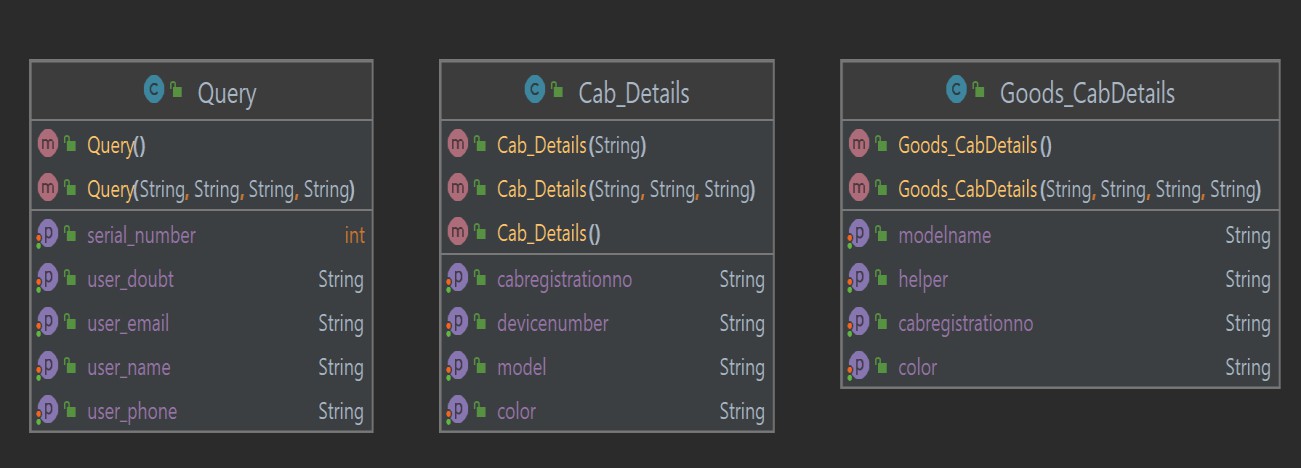
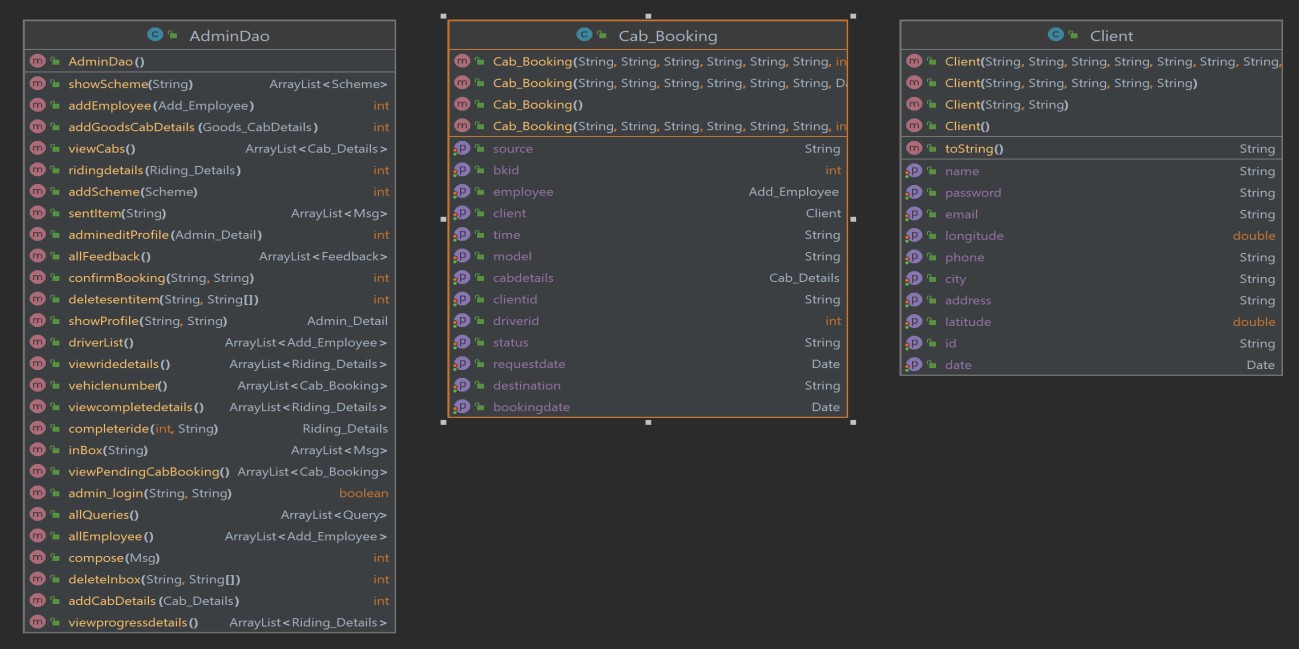


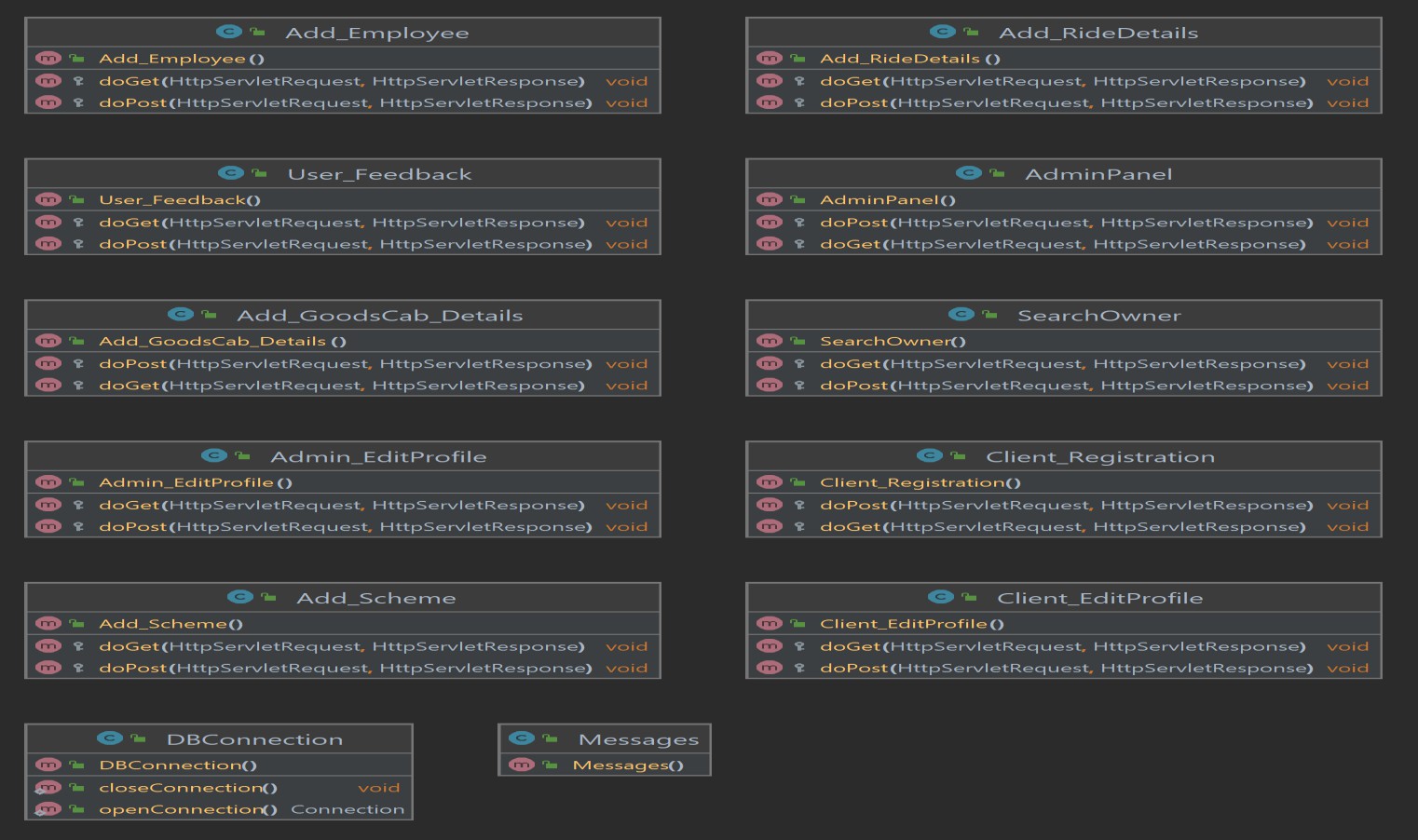
Notification:

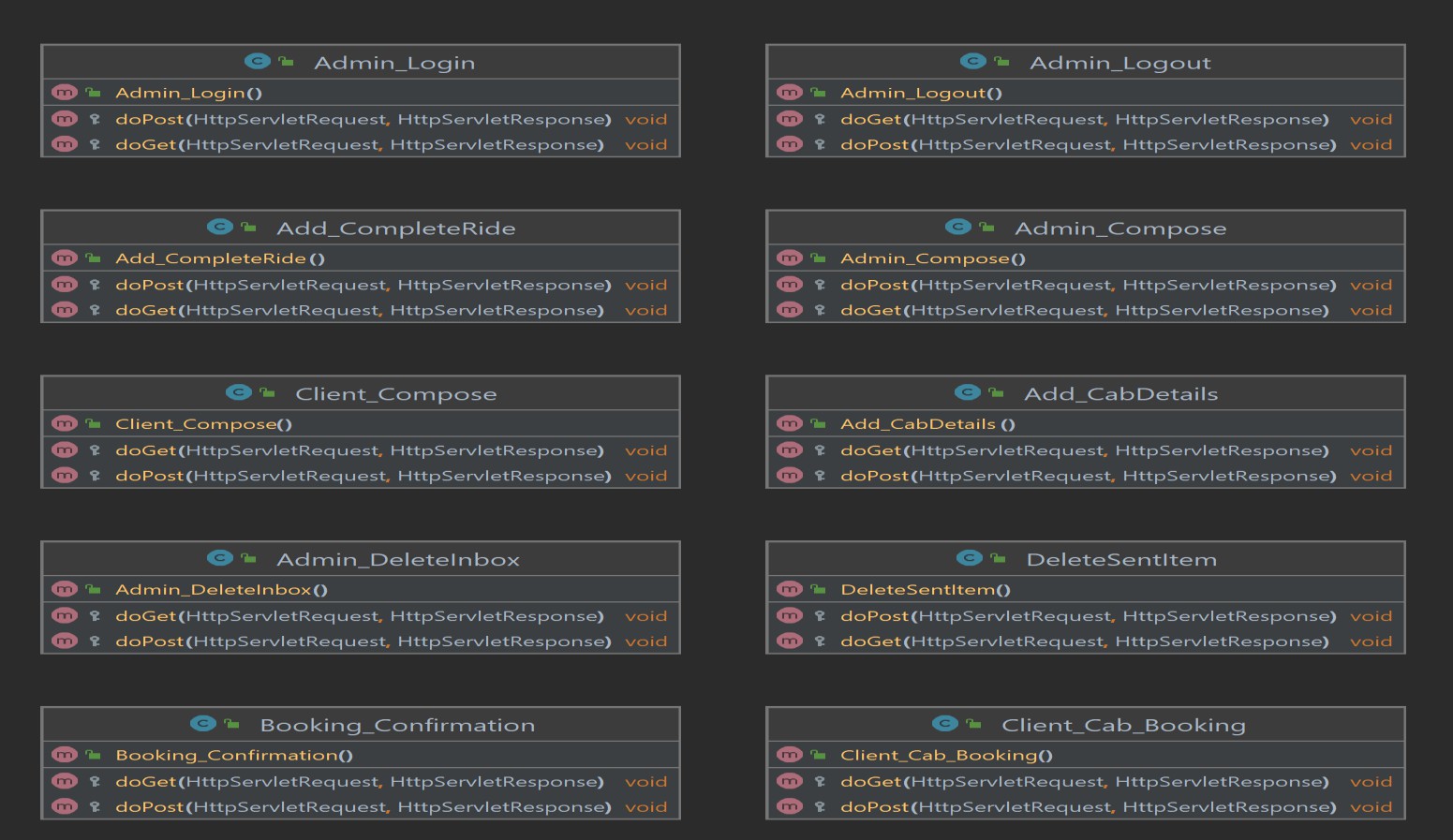


**Class Diagram**

****

****



****

**Project Code**

AdminDao:

package transbuddy.dao;

import transbuddy.beans.Add\_Employee; import transbuddy.beans.Cab\_Booking; import transbuddy.beans.Cab\_Details; import transbuddy.beans.Client;

import transbuddy.beans.Goods\_CabDetails; import transbuddy.beans.Msg;

import transbuddy.beans.Transport\_CabBooking; import transbuddy.dbutils.DBConnection;

import java.sql.\*;

import java.util.ArrayList; public class ClientDao {

private Connection connection;

// show message

public Msg viewMessage(String strsql, int messageid) { Msg message = null;

PreparedStatement ps = null; ResultSet rs = null;

try {

connection = DBConnection.openConnection();

ps = connection.prepareStatement(strsql);

ps.setInt(1, messageid); rs = ps.executeQuery(); rs.next();

message = new Msg(); message.setSenderid(rs.getString("senderid")); message.setDate(rs.getDate("date")); message.setMessage\_text(rs.getString("message\_text"));

} catch (SQLException se) { se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close(); if (rs != null)

rs.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return message;

}

public int registration(Client c) {

connection = DBConnection.openConnection();

String strinsert = "insert into client(id, password, name, email, phone, city, address, date, latitude, longitude)values(?,?,?,?,?,?,?,?,?,?)";

java.util.Date d = new java.util.Date();

long dt = d.getTime();// long value of today's date java.sql.Date sd = new java.sql.Date(dt); System.out.println("sql date is " + sd);

int status = 0; PreparedStatement ps = null;

try {

ps = connection.prepareStatement(strinsert); ps.setString(1, c.getId());

ps.setString(2, c.getPassword()); ps.setString(3, c.getName()); ps.setString(4, c.getEmail()); ps.setString(5, c.getPhone()); ps.setString(6, c.getCity()); ps.setString(7, c.getAddress()); ps.setDate(8, sd); ps.setDouble(9, c.getLatitude());

ps.setDouble(10, c.getLongitude()); System.out.println(ps);

status = ps.executeUpdate();

} catch (SQLException se) { se.printStackTrace();

} finally {

try {

ps.close(); connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return status;

}

public boolean login(String id, String password) { connection = DBConnection.openConnection();

String strsql = "select \* from client where id=? and password=?";

PreparedStatement ps = null; ResultSet rs = null;

try {

ps = connection.prepareStatement(strsql); ps.setString(1, id);

ps.setString(2, password); System.out.println(ps);

rs = ps.executeQuery(); if (rs.next()) {

return true;

}

} catch (SQLException se) { se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close(); if (rs != null)

rs.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return false;

}

// method to view profile

public Client showProfile(String sql, String id) { Client c = null;

connection = DBConnection.openConnection(); PreparedStatement ps = null;

ResultSet rs = null;

try {

ps = connection.prepareStatement(sql); ps.setString(1, id);

rs = ps.executeQuery();

rs.next();// to move the cursor in the data set String name = rs.getString("name");

String email = rs.getString("email"); String phone = rs.getString("phone"); String city = rs.getString("city");

String address = rs.getString("address");

c = new Client(name, email, phone, city, address);// object of bean class

} catch (SQLException se) {

se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close(); if (rs != null)

rs.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return c;

}// method closed

//method for updating profile

public int editProfile(Client c) {

connection = DBConnection.openConnection(); PreparedStatement ps = null;

int status = 0; try {

where id=?";

String strupdate = "update client set email=?,phone=?,city=?,address=? ps = connection.prepareStatement(strupdate);

ps.setString(1, c.getEmail()); ps.setString(2, c.getPhone()); ps.setString(3, c.getCity());

ps.setString(4, c.getAddress()); ps.setString(5, c.getId()); status = ps.executeUpdate();

} catch (SQLException se) { se.printStackTrace();

}

finally {

try {

if (ps != null)

ps.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return status;

}

public int compose(Msg m) {

connection = DBConnection.openConnection();

String strinsert = "insert into message(messageid, senderid, receiverid, subject, message\_text, rstatus, sstatus, date)values(?,?,?,?,?,?,?,?)";

java.util.Date d = new java.util.Date(); java.sql.Date sd = new java.sql.Date(d.getTime());

int status = 0; PreparedStatement ps = null;

try {

ps = connection.prepareStatement(strinsert); ps.setInt(1, m.getMessageid()); ps.setString(2, m.getSenderid()); ps.setString(3, m.getReceiverid()); ps.setString(4, m.getSubject()); ps.setString(5, m.getMessage\_text()); ps.setString(6, m.getRstatus()); ps.setString(7, m.getStatus());

ps.setDate(8, sd); System.out.println(ps); status = ps.executeUpdate();

} catch (SQLException se) { se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return status;

}

public ArrayList<Msg> inBox(String receiverid) { Msg message = null;

ArrayList<Msg> messageList = new ArrayList<Msg>();

String strsql = "select \* from message where receiverid=? and rstatus='true'"; connection = DBConnection.openConnection();

PreparedStatement ps = null;

// int messageid, String senderid, String receiverid, String subject, String

// message\_text, String rstatus,

// String status, Date date ResultSet rs = null;

try {

ps = connection.prepareStatement(strsql); ps.setString(1, receiverid);

rs = ps.executeQuery();

while (rs.next()) {

message = new Msg(rs.getInt("messageid"), rs.getString("senderid"), rs.getString("receiverid"),

rs.getString("subject"), rs.getString("message\_text"), rs.getString("rstatus"),

rs.getString("sstatus"), rs.getDate("date"));

messageList.add(message);// adding object in the arraylist

}

} catch (SQLException se) { se.printStackTrace();

}

// finally block;

finally {

try {

if (ps != null)

ps.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return messageList;

}

// sent item

public ArrayList<Msg> sentItem(String senderid) { Msg message = null;

ArrayList<Msg> messageList = new ArrayList<Msg>();

String strsql = "select \* from message where senderid=? and sstatus='true'"; connection = DBConnection.openConnection();

PreparedStatement ps = null;

// int messageid, String senderid, String receiverid, String subject, String

// message\_text, String rstatus,

// String status, Date date ResultSet rs = null;

try {

ps = connection.prepareStatement(strsql); ps.setString(1, senderid);

rs = ps.executeQuery();

while (rs.next()) {

message = new Msg(rs.getInt("messageid"), rs.getString("senderid"), rs.getString("receiverid"),

rs.getString("subject"), rs.getString("message\_text"), rs.getString("rstatus"),

rs.getString("sstatus"), rs.getDate("date"));

messageList.add(message);// adding object in the arraylist

}

} catch (SQLException se) { se.printStackTrace();

}

// finally block;

finally {

try {

if (ps != null)

ps.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return messageList;

}

public int deleteInbox(String userid, String[] ids) {

receiverid=?";

connection = DBConnection.openConnection();

String strupdate = "update message set rstatus=? where messageid=? and

PreparedStatement ps = null; int status = 0;

try {

ps = connection.prepareStatement(strupdate); for (int i = 0; i < ids.length; i++) {

ps.setString(1, "false");

ps.setInt(2, Integer.parseInt(ids[i])); ps.setString(3, userid); ps.addBatch();// store data into buffer System.out.println(ps);

}

int rows[] = ps.executeBatch();// batch processing

for (int i = 0; i < rows.length; i++) { if (rows[i] < 0) {

break;

} else

}

status = 1;

} catch (SQLException se) { try {

connection.rollback();

} catch (SQLException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return status;

}

public int deletesentitem(String userid, String[] ids) {

senderid=?";

connection = DBConnection.openConnection();

String strupdate = "update message set sstatus=? where messageid=? and

PreparedStatement ps = null; int status = 0;

try {

ps = connection.prepareStatement(strupdate); for (int i = 0; i < ids.length; i++) {

ps.setString(1, "false");

ps.setInt(2, Integer.parseInt(ids[i])); ps.setString(3, userid); ps.addBatch();// store data into buffer System.out.println(ps);

}

int rows[] = ps.executeBatch();// batch processing

for (int i = 0; i < rows.length; i++) { if (rows[i] < 0) {

break;

} else

}

status = 1;

} catch (SQLException se) { try {

connection.rollback();

} catch (SQLException e) {

// TODO Auto-generated catch block e.printStackTrace();

}

se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return status;

}

public int addCabbooking(Cab\_Booking cb) { connection = DBConnection.openConnection();

String strinsert = "insert into cab\_bookings(bkid, clientid, source, destination, model, requestdate, bookingdate, time, status)values(?,?,?,?,?,?,?,?,?)";

int status = 0; PreparedStatement ps = null;

try {

ps = connection.prepareStatement(strinsert); ps.setInt(1, cb.getBkid());

ps.setString(2, cb.getClientid()); ps.setString(3, cb.getSource()); ps.setString(4, cb.getDestination()); ps.setString(5, cb.getModel()); ps.setDate(6, cb.getRequestdate()); ps.setDate(7, cb.getBookingdate()); ps.setString(8, cb.getTime()); ps.setString(9, "pending");

System.out.println(ps); status = ps.executeUpdate();

} catch (SQLException se) { se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return status;

}

public ArrayList<Cab\_Details> showdropdown() { connection = DBConnection.openConnection();

String sql = "select \* from cab";

ArrayList<Cab\_Details> cablist = new ArrayList<Cab\_Details>(); Cab\_Details cd = null;

PreparedStatement ps = null; ResultSet rs = null;

try {

ps = connection.prepareStatement(sql); rs = ps.executeQuery();

while (rs.next()) {// to move the cursor in the data set String model = rs.getString("model");

System.out.println(model);

cd = new Cab\_Details(model);// object of bean class cablist.add(cd);

}

} catch (SQLException se) { se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close(); if (rs != null)

rs.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return cablist;

}// method closed

public ArrayList<Goods\_CabDetails> showgoodsdropdown() { connection = DBConnection.openConnection();

String sql = "select \* from goods\_cab";

ArrayList<Goods\_CabDetails> goodscablist = new ArrayList<Goods\_CabDetails>();

Goods\_CabDetails gd = null;

PreparedStatement ps = null; ResultSet rs = null;

try {

ps = connection.prepareStatement(sql); rs = ps.executeQuery();

while (rs.next()) {// to move the cursor in the data set String model = rs.getString("modelname");

System.out.println(model);

gd = new Goods\_CabDetails(model);// object of bean class goodscablist.add(gd);

}

} catch (SQLException se) { se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close(); if (rs != null)

rs.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return goodscablist;

}// method closed

public ArrayList<Cab\_Booking> viewcabbookingstatus(String id) { connection = DBConnection.openConnection();

ArrayList<Cab\_Booking> bookingstatuslist = new ArrayList<Cab\_Booking>();

Cab\_Booking cb = null; Add\_Employee emp = null;

String strsql = "Select cb.source,cb.destination,cb.model,cb.bookingdate,cb.status,emp.name from cab\_bookings cb,add\_employee emp where cb.clientid=? and emp.id=cb.driverid ";

java.util.Date d = new java.util.Date(); java.sql.Date sd = new java.sql.Date(d.getTime());

PreparedStatement ps = null; ResultSet rs = null;

try {

ps = connection.prepareStatement(strsql); ps.setString(1, id);

rs = ps.executeQuery();

// id, name, email, phone, feedback\_text, rating, date while (rs.next()) {

String source = rs.getString("source");

String destination = rs.getString("destination"); String model = rs.getString("model");

String bookingdate = rs.getString("bookingdate");

String status = rs.getString("status"); String name = rs.getString("name");

emp = new Add\_Employee(); emp.setName(name);

cb = new Cab\_Booking(); cb.setSource(source); cb.setDestination(destination); cb.setModel(model); cb.setBookingdate(sd);

cb.setStatus(status); cb.setEmployee(emp);

bookingstatuslist.add(cb);

}

}

catch (SQLException se) { se.printStackTrace();

}

return bookingstatuslist;

}// method closed

public ArrayList<Transport\_CabBooking> viewtransportcabbookingstatus(String id) { connection = DBConnection.openConnection(); ArrayList<Transport\_CabBooking> statuslist = new

ArrayList<Transport\_CabBooking>();

Transport\_CabBooking tb = null; Add\_Employee emp = null; Goods\_CabDetails g=null;

String strsql = "Select tb.source,tb.destination,tb.model,tb.bookingdate,tb.status,emp.name,g.helper from transport\_cab\_booking tb,add\_employee emp,goods\_cab g where tb.clientid=? and emp.id=tb.driverid and g.modelname=tb.model";

java.util.Date d = new java.util.Date(); java.sql.Date sd = new java.sql.Date(d.getTime());

PreparedStatement ps = null; ResultSet rs = null;

try {

ps = connection.prepareStatement(strsql); ps.setString(1, id);

rs = ps.executeQuery();

// id, name, email, phone, feedback\_text, rating, date while (rs.next()) {

String source = rs.getString("source");

String destination = rs.getString("destination"); String model = rs.getString("model");

String bookingdate = rs.getString("bookingdate");

String status = rs.getString("status"); String name = rs.getString("name");

String helper=rs.getString("helper");

emp = new Add\_Employee(); emp.setName(name);

g=new Goods\_CabDetails(); g.setHelper(helper);

tb = new Transport\_CabBooking(); tb.setSource(source); tb.setDestination(destination); tb.setModel(model); tb.setBookingdate(sd); tb.setStatus(status); tb.setEmployee(emp);

tb.setGoodscabdetails(g);

statuslist.add(tb);

}

}

catch (SQLException se) { se.printStackTrace();

}

return statuslist;

}// method closed

public boolean checkId(String id) {

connection = DBConnection.openConnection(); String strsql="select \* from client where id=?";

PreparedStatement ps = null; ResultSet rs = null;

try {

ps = connection.prepareStatement(strsql); ps.setString(1,id);;

rs = ps.executeQuery(); if (rs.next()) {

return true;

}

} catch (SQLException se) { se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close(); if (rs != null)

rs.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return false;

}

public int addTransportCabbooking(Transport\_CabBooking tb)

{

connection = DBConnection.openConnection();

String strinsert = "insert into transport\_cab\_booking(bkid, clientid, source, destination, model, requestdate, bookingdate, time, status)values(?,?,?,?,?,?,?,?,?)";

int status = 0; PreparedStatement ps = null;

try {

ps = connection.prepareStatement(strinsert); ps.setInt(1, tb.getBkid());

ps.setString(2, tb.getClientid()); ps.setString(3, tb.getSource());

ps.setString(4, tb.getDestination()); ps.setString(5, tb.getModel()); ps.setDate(6, tb.getRequestdate()); ps.setDate(7, tb.getBookingdate()); ps.setString(8, tb.getTime()); ps.setString(9, "pending");

System.out.println(ps); status = ps.executeUpdate();

} catch (SQLException se) { se.printStackTrace();

} finally {

try {

if (ps != null)

ps.close();

if (connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return status;

}

//forgot password

public int forgotpassword(String email)

{

connection=DBConnection.openConnection(); String strsql="select \* from client where email=?";

PreparedStatement ps=null; ResultSet rs=null;

try {

ps=connection.prepareStatement(strsql); ps.setString(1,email); System.out.println(ps); rs=ps.executeQuery();

if(rs.next())

return 1;

}

catch (SQLException se)

{

se.printStackTrace();

}

finally {

try {

}

if(ps!=null)

ps.close(); if(rs!=null)

rs.close(); if(connection !=null)

connection.close();

catch (SQLException e) {

}

}

return 0;

}

//update password

public int resetpassword(String email, String pass)

{

connection = DBConnection.openConnection();

PreparedStatement ps = null; int status=0;

try {

String strupdate="update client set password=? where email=?"; ps=connection.prepareStatement(strupdate);

ps.setString(1, pass); ps.setString(2, email); status=ps.executeUpdate();

}

catch (SQLException se) { se.printStackTrace();

}

finally {

try {

if(ps != null)

ps.close(); if(connection != null)

connection.close();

} catch (SQLException se) { se.printStackTrace();

}

}

return status;

}

}

Client login:

<%@ **page** language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!**DOCTYPE** html>

<**html**>

<**head**>

<**meta** charset=*"ISO-8859-1"*>

<**title**>TransportationBuddy</**title**>

<**link** rel=*"stylesheet"* href=*"/TransBuddy/css/logincss.css"*>

<**link** rel=*"stylesheet"* href=*"/TransBuddy/css/feedcss.css"*>

<%@**include** file=*"/commoncss/allcss.html"*%>

</**head**>

<**body** style="background-color: *#212534*">

<%-- <%@include file="/headers/common\_header.html" %> --%>

<%

String msg=(String)request.getAttribute("message"); if(msg!=null)

{

%>

<**div** class=*"alert alert-warning alert-dismissible fade show"*

role=*"alert"*

style="width: *25%*; height: *40px*; margin-left: *0%*; margin-top: *40px*">

<**strong**><%=msg %></**strong**>

<**button** type=*"button"* class=*"btn-close"* data-bs-dismiss=*"alert"* aria-label=*"Close"*></**button**>

</**div**>

<%} %>

<**div** class=*"card"*>

<**div** class=*"card-body"*>

<**div** class=*"container"*>

<**div** class=*"row"*>

<**div** class=*"col-md-12 text-center"*>

<**h3** class=*"animate-charcter"* style="margin-top: *20px*">LOGIN FORM</**h3**>

</**div**>

</**div**>

</**div**> Login Please

<**form** method=*"post"* action=*"/TransBuddy/Client\_Login"*

class=*"needs-validation"* novalidate>

<**div** id=*"dynamic\_container"*>

<**div** class=*"input-group mt-5"*>

<**div** class=*"input-group-prepend"*>

<**span** class=*"input-group-text br-15"*><**i**

class=*"fas fa-user-graduate"*></**i**></**span**>

</**div**>

<**input** type=*"text"* name=*"id"* placeholder=*"Enter Your Id"* required class=*"form-control"* />

<**div** class=*"invalid-feedback"*>Enter Your id please</**div**>

</**div**>

<**div** class=*"input-group mt-3"*>

<**div** class=*"input-group-prepend"*>

<**span** class=*"input-group-text br-15"*><**i** class=*"fas fa-key"*></**i**></**span**>

</**div**>

<**input** type=*"password"* name=*"password"* required placeholder=*"Enter Your Password"* class=*"form-control"* />

<**div** class=*"invalid-feedback"*>Enter Your Password please</**div**>

</**div**>

</**div**>

<**div** class=*"card-footer "*>

<!-- <a class="btn btn-secondary btn-sm" id="add\_more"><i class="fas fa-plus-circle"></i> Add</a>

<a class="btn btn-secondary btn-sm" id="remove\_more"><i class="fas fa-trash-alt"></i> Remove</a> -->

<**button** style="width: *150px*; height: *50px*" class=*"btn btn-success btn-sm float-right submit\_btn"*>

<**i** class=*"fas fa-arrow-alt-circle-right"*></**i**> <**b**>Submit</**b**>

</**button**>

</**div**>

</**form**>

Don't have an account?<**a** href=*"/TransBuddy/client/client\_registration.jsp"*>Register here</**a**><**br**>

<**a** href=*"/TransBuddy/client/forgetpassword.jsp"*>forgot password</**a**>

</**div**>

</**div**>

<**script** src=*"/TransBuddy/commonjs/custom\_validation.js"*></**script**>

<%@**include** file=*"/commonjs/alljs.html"*%>

</**body**>

</**html**>

# CHAPTER 5

# RESULT AND DISCUSSION

**5. Result and Discussion**

The development and implementation of the **Transportation Buddy** web application successfully address the core challenges of traditional manual transport booking systems. The system is designed to simplify and automate the process of booking vehicles for both passengers and goods, providing an efficient, user-friendly, and accessible platform that meets the diverse needs of individual users, businesses, and vendors.

One of the key results of the project is the creation of a centralized portal that offers real-time access to vehicle information, booking status, driver details, and payment history. This significantly reduces the dependence on manual processes such as paperwork, phone calls, and in-person visits. Users can now register on the portal, log in with their credentials, search for vehicles that meet their requirements, and complete the booking process in just a few clicks. The entire system operates seamlessly on a web browser and is compatible with different screen resolutions, ensuring accessibility across various devices.

The **vehicle tracking feature**, enabled by GPS integration, allows both customers and administrators to monitor the real-time location of vehicles. This has proven to be particularly useful in increasing transparency and reliability. Customers feel more secure when they can track their shipment or the vehicle they are about to board. The ability to view **driver details** such as name, contact information, and ratings also adds an extra layer of trust and safety.

Another notable result is the incorporation of a **rating and review system**, where users can provide feedback based on their experience. This feature not only helps improve the service quality but also assists new users in selecting the most reliable vehicles and drivers.

The **online payment module** has streamlined financial transactions, offering a secure and efficient method for completing bookings. E-bills are generated automatically, reducing the need for physical invoices and simplifying expense tracking for both users and transport companies.

The portal also supports **admin functionality**, including vehicle management, employee/driver management, scheduling, customer handling, and responding to user queries. This enables transport companies to handle a larger number of bookings efficiently and scale their operations as needed. Furthermore, the inclusion of **notification and messaging modules** has enhanced communication among users and system administrators, minimizing delays and improving customer support.

During testing, the system performed efficiently in handling simultaneous bookings, maintaining accurate data logs, and ensuring that communication between modules (e.g., booking and driver allotment) was smooth. Manual testing and integration testing confirmed the reliability of each module, while user feedback indicated high satisfaction with ease of use, responsiveness, and the informative interface.

In conclusion, the **Transportation Buddy** web application meets its core objectives by delivering a comprehensive, modern transport booking system that is secure, scalable, and user-focused. It not only resolves the inefficiencies of manual booking systems but also introduces new standards for transparency, convenience, and customer satisfaction. With potential for future enhancements like mobile app integration and AI-based vehicle recommendations, the platform is well-positioned for broader adoption and impact in the transportation sector.

# CHAPTER 6

# CONCLUSION AND FUTURE SCOPE

* 1. **Conclusion**

The goal of the **Transportation Buddy** website is to provide a convenient, reliable, and user- friendly platform where individuals and organizations can effortlessly book vehicles for their transportation needs. By offering a wide range of vehicle options—including cars, vans, buses, and even specialized transport—the website ensures that it caters to a diverse audience. Whether it's a solo traveler needing a quick ride, a family planning a weekend getaway, or a corporate group arranging large-scale travel, Transportation Buddy is designed to accommodate all types of requirements.

The platform aims to streamline the entire booking process by incorporating features such as real-time availability, transparent pricing, and flexible scheduling. Users can browse, compare, and book vehicles based on their specific preferences, such as size, budget, fuel type, or duration of use. Moreover, the website emphasizes ease of access through a responsive design that works smoothly across desktops, tablets, and smartphones.

With its focus on efficiency and versatility, Transportation Buddy not only saves users time and effort but also enhances their overall travel experience. The ultimate vision is to become a trusted go-to platform that empowers users to find the right transportation solution, every time, with just a few clicks.

* 1. **Future Scope**
     + Integration with mobile apps:

The portal can be integrated with a mobile app for improved accessibility and convenience.

* + - Real-time communication:

Video conferencing with drivers can be added to enhance security and communication.

**REFERENCES**

* Sommerville, I. (2016). *Software Engineering* (10th ed.). Pearson Education.

→ Used for understanding SDLC models and best practices in system development.

* Pressman, R. S., & Maxim, B. R. (2020). *Software Engineering: A Practitioner's Approach*

(9th ed.). McGraw-Hill Education.

→ Provided guidance on designing modular applications and methodologies.

* Patel, K., & Desai, M. (2020). "Online Vehicle Booking System." *International Journal of Research and Analytical Reviews (IJRAR)*, Vol. 7, Issue 2, pp. 320-325.

→ Reference for system structure and features similar to vehicle booking platforms.

* Oracle. (2023). *Java Platform, Standard Edition Documentation*. Retrieved from: <https://docs.oracle.com/javase/>

→ Used for implementing Java-based functionalities in backend development.

* MySQL Documentation. (2023). *MySQL 8.0 Reference Manual*. Retrieved from: <https://dev.mysql.com/doc/refman/8.0/en/>

→ For designing and managing the system’s database structure.

* W3Schools. (2024). *HTML, CSS, JavaScript Tutorial*. Retrieved from: <https://www.w3schools.com/>

→ Used for developing frontend components of the application.

* Google Maps API Documentation. (2023). Retrieved from: https://developers.google.com/maps/documentation

→ For understanding GPS tracking and vehicle location integration concepts.

* Stack Overflow Developer Community. (n.d.). Retrieved from: <https://stackoverflow.com/>

→ For resolving programming and integration issues during development.

* Mishra, A. & Gupta, R. (2021). “Vehicle Management System using Web Technologies.”

*International Journal of Computer Applications (IJCA)*, Vol. 179, No. 39, pp. 1–5.

→ Provided ideas related to real-time vehicle tracking and user feedback systems.