solartis

9.Vehicle Sales Management System

Comprehensive Documentation and User Guide

By: MUHAMMED IRFAN KUZHYLANGATTIL

9. VEHICLE SALES MANAGEMENT SYSTEM

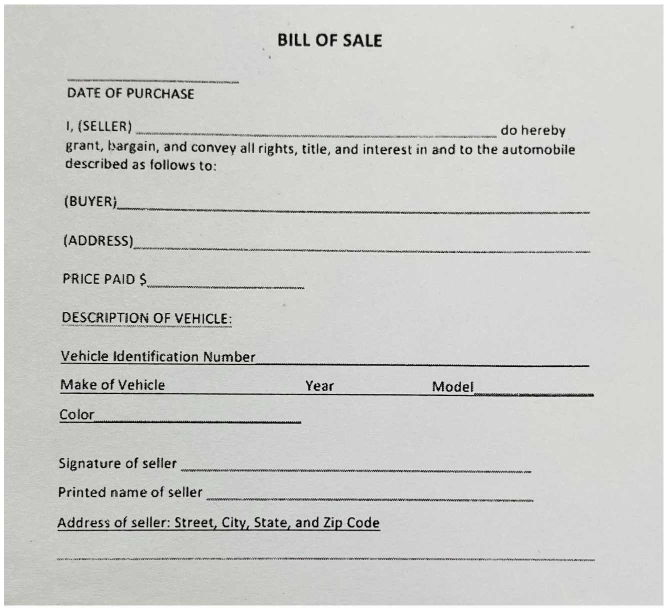
Name: Muhammed Irfan Kuzhylangattil

Project Repository Link: Github - Vehicle Sales Management

PROBLEM SATEMENT:

To create an application with the following functionalities:

1. 2 Roles - Manager, Sales
2. Manage Vehicles - Add/Edit/Remove - Sales
3. Manage Payment transactions - Manager
4. Generate bill in PDF and send to mail - Auto generation



BILL OF SALE

DATE OF PURCHASE

I, (SELLER) ____________ do hereby grant, bargain, and convey all rights, title, and interest in and to the automobile described as follows to:

(BUYER)____________

(ADDRESS)____________

PRICE PAID $____________

DESCRIPTION OF VEHICLE:

Vehicle Identification Number ____________

Make of Vehicle ____________ Year ____________ Model____________

Color____________

Signature of seller ____________

Printed name of seller ____________

Address of seller: Street, City, State, and Zip Code

5. PURCHASER HAS TWENTY (20) CALENDAR DAYS TO TRANSFER THE VEHICLE INTO HIS OR HER NAME WITHOUT PENALTIES AND INTEREST.
6. Profit calculation for each vehicle and then consolidate the amount for the given period - Auto calculation
7. Calculate vehicle score based on the vehicle assessment - Auto calculation - Salespeople enter information
8. Insurance coverage status (just data entry) - Sales
9. Additional Accessories details (just data entry) - Sales
10. Vehicle details management (Name, Make, Model, Year of manufacturing, Owner, Engine number, Chassis number....) - Sales

APPLICATION METHODOLOGY:

The vehicle sales management project is a comprehensive system designed to facilitate various aspects of managing vehicle sales, including inventory management, sales transactions, user management, and more. Here's a summary of how the project works:

1. **Database Setup**: The project relies on a relational database (MySQL in this case) to store information about vehicles, transactions, users, accessories, and scores. The database schema includes tables such as `vehicle`, `transaction`, `users`, `accessories`, and `score`.

2. **User Authentication**: The project includes a user authentication system where users can log in using their username and password. The system verifies the user credentials against the database records in the `users` table.

3. **Role-Based Access Control**: Upon login, users are directed to different pages based on their assigned role. For example, salespeople are directed to a sales dashboard (`sales.jsp`), while managers are directed to a manager dashboard (`manager.jsp`). Administrators have access to additional functionalities, such as adding, editing, and removing users.

4. **Vehicle Management**: Salespeople can add new vehicles to the inventory (`add_vehicle.jsp`), edit existing vehicle details (`edit_vehicle.jsp`), and remove vehicles from the inventory (`remove_vehicle.jsp`). Each vehicle record includes information such as make, model, year, price, owner, engine number, chassis number, and insurance coverage status.

5. **Transaction Management**: The system allows users to record sales transactions (`add_transaction.jsp`), including details such as the vehicle sold, the amount paid, and the transaction date. The transaction records are stored in the `transaction` table.

6. **Vehicle Assessment and Scoring**: The system provides functionality to calculate a vehicle's score based on various factors such as engine power, total mileage, safety rating, vehicle condition, year, and

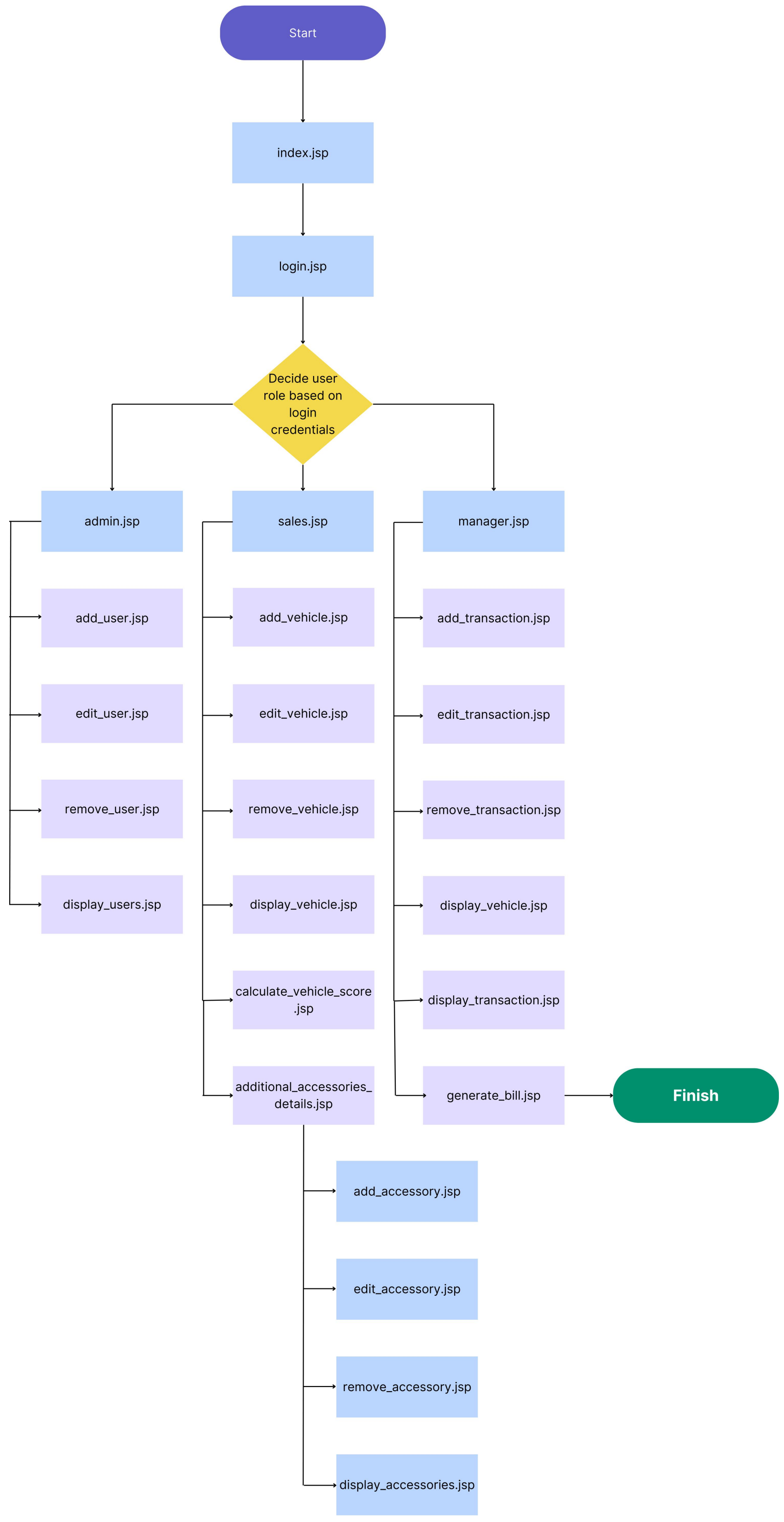
price. Users can input these factors through `calculate_vehicle_score.jsp`, and the system calculates the score using predefined weights.

7. **Additional Accessories Management**: Users can manage additional accessories for vehicles, such as floor mats, chrome accessories, GPS systems, and rearview cameras. The system allows users to add, edit, remove, and display accessory details.

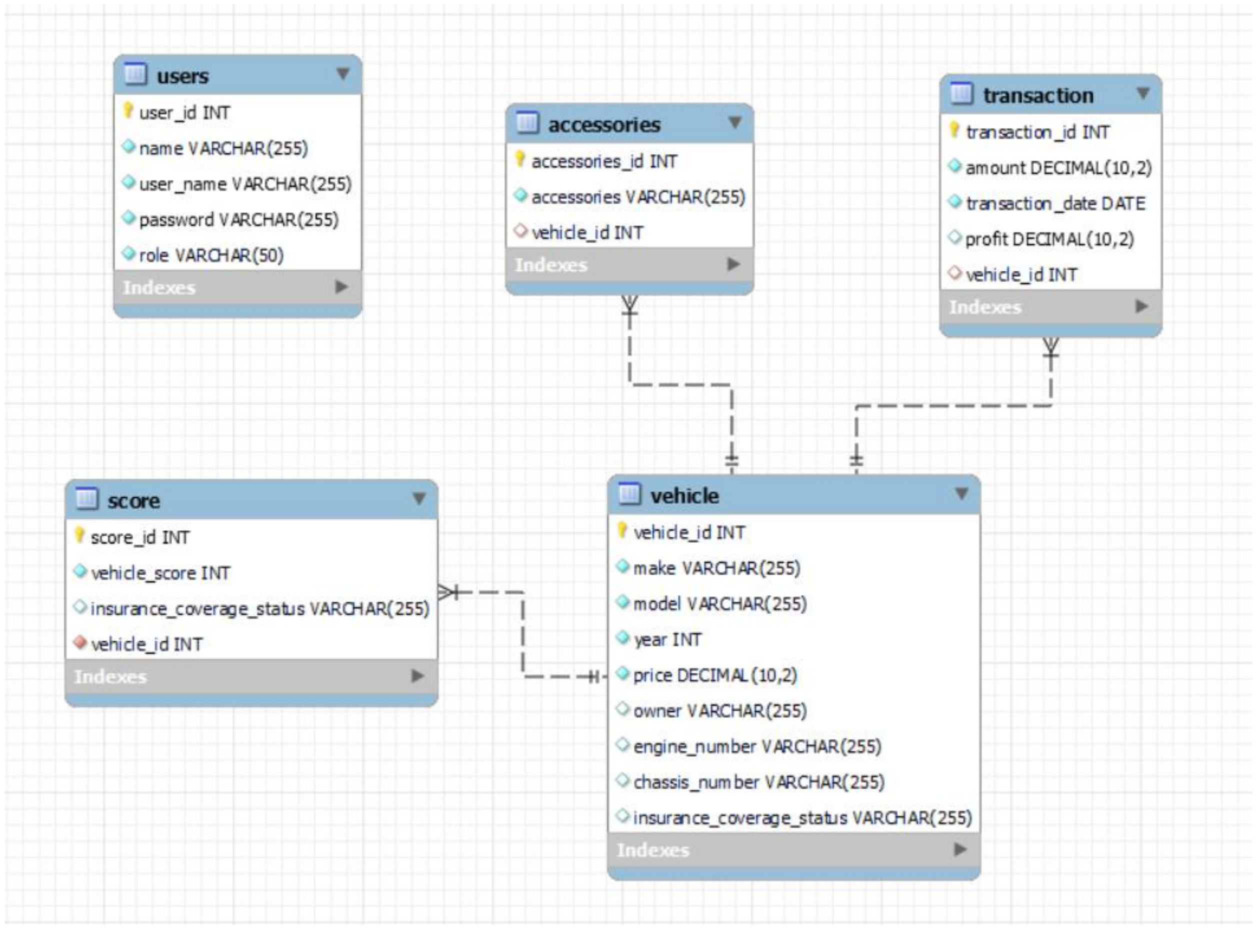
8. **Generating Documents**: The project includes functionality to generate documents such as bills of sale (`generate_bil.jsp`). It uses the Apache PDFBox library to create PDF documents dynamically with relevant information such as the seller's name, buyer's name, vehicle details, and price paid. The PDF is also attached with mail and sent to the recipient using javaxmail library.

Overall, the project streamlines various aspects of vehicle sales management, providing a user-friendly interface for salespeople, managers, and administrators to effectively manage inventory, transactions, user accounts, and related tasks.

APPLICATION WORKFLOW:



DATABASE DEFINITIONS AND RELATIONS:



Java code for the above table creations:

```
String createVehicleTableQuery = "CREATE TABLE IF NOT EXISTS vehicle ("
+ "vehicle_id INT AUTO_INCREMENT PRIMARY KEY,"
+ "make VARCHAR(255) NOT NULL,"
+ "model VARCHAR(255) NOT NULL,"
+ "year INT NOT NULL,"
+ "price DECIMAL(10,2) NOT NULL,"
+ "owner VARCHAR(255),"
+ "engine_number VARCHAR(255),"
+ "chassis_number VARCHAR(255),"
+ "insurance_coverage_status VARCHAR(255)"
+ ")";

String createTransactionTableQuery = "CREATE TABLE IF NOT EXISTS transaction ("
+ "transaction_id INT AUTO_INCREMENT PRIMARY KEY,"
+ "amount DECIMAL(10,2) NOT NULL,"
+ "transaction_date DATE NOT NULL,"
+ "profit DECIMAL(10,2),"
+ "vehicle_id INT,"
+ "FOREIGN KEY (vehicle_id) REFERENCES vehicle(vehicle_id)"
+ ")";

String createUsersTableQuery = "CREATE TABLE IF NOT EXISTS users ("
+ "user_id INT AUTO_INCREMENT PRIMARY KEY,"
+ "name VARCHAR(255) NOT NULL,"
+ "user_name VARCHAR(255) UNIQUE NOT NULL,"
+ "password VARCHAR(255) NOT NULL,"
+ "role VARCHAR(50) NOT NULL"
+ ")";

String query = "INSERT IGNORE INTO users (name, user_name, password, role) VALUES ('Admin', 'admin', 'admin', 'admin')";
PreparedStatement pstmt = conn.prepareStatement(query);
pstmt.executeUpdate(query);

String createAccessoriesTableQuery = "CREATE TABLE IF NOT EXISTS accessories ("
+ "accessories_id INT AUTO_INCREMENT PRIMARY KEY,"
+ "accessories VARCHAR(255) NOT NULL,"
+ "vehicle_id INT,"
+ "FOREIGN KEY (vehicle_id) REFERENCES vehicle(vehicle_id)"
+ ")";

String createScoreTableQuery = "CREATE TABLE IF NOT EXISTS score ("
+ "score_id INT AUTO_INCREMENT PRIMARY KEY,"
+ "vehicle_score INT NOT NULL,"
+ "insurance_coverage_status VARCHAR(255),"
+ "vehicle_id INT NOT NULL,"
+ "FOREIGN KEY (vehicle_id) REFERENCES vehicle(vehicle_id)"
+ ")";
```

CODE:

index.jsp

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Vehicle Sales Management</title>
<style>
body {
font-family: Arial, sans-serif;
margin: 0;
padding: 0;
background-color: #f0f0f0;
text-align: center;
}
.container {
max-width: 600px;
margin: 100px auto;
background-color: #fff;
border-radius: 10px;
padding: 20px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
h1 {
margin-bottom: 30px;
color: #333;
}
.btn {
display: inline-block;
padding: 10px 20px;
margin: 10px;
background-color: #007bff;
color: #fff;
text-decoration: none;
border: none;
border-radius: 5px;
cursor: pointer;
}
.btn:hover {
background-color: #0056b3;
}
```

```
</style>
</head>
<body>
<div class="container">
<h1>Vehicle Sales Management</h1>
<a href="login.jsp" class="btn">Login</a>
</div>
</body>
</html>
```

login.jsp

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login</title>
<style>
body {
font-family: Arial, sans-serif;
margin: 0;
padding: 0;
background-color: #f0f0f0;
text-align: center;
}
h2 {
margin-top: 50px;
color: #333;
}
form {
max-width: 300px;
margin: 20px auto;
padding: 20px;
background-color: #fff;
border-radius: 10px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}
label {
display: block;
margin-bottom: 10px;
font-weight: bold;
color: #333;
}
input[type="text"],
input[type="password"] {
width: 90%;
padding: 10px;
margin-bottom: 20px;
font-size: 16px;
border-radius: 5px;
border: 1px solid #ccc;
}
button {
display: block;
width: 100%;
padding: 10px;
background-color: #007bff;
color: #fff;
border: none;
border-radius: 5px;
cursor: pointer;
font-size: 16px;
}
button:hover {
background-color: #0056b3;
}
.btn {
display: inline-block;
padding: 10px 20px;
background-color: #007bff;
width: 87%;
color: #fff;
text-decoration: none;
border: none;
border-radius: 5px;
cursor: pointer;
}
.btn:hover {
background-color: #0056b3;
}
.failure-message {
color: red;
margin-top: 10px;
}
</style>
</head>
<body>
<h2>Login</h2>
<form action="sqlChecksServlet" method="post">
```

```
<label for="username">Username:</label>
<input type="text" id="username" name="username" required><br><br>
<label for="password">Password:</label>
<input type="password" id="password" name="password" required><br><br>
<button type="submit">Login</button>
<br>
<a href="index.jsp" class="btn">Back</a>
<div class="failure-message">
<% String failureMessage = (String) request.getAttribute("failureMessage"); %>
<% if (failureMessage != null) { %>
<%= failureMessage %>
<% } %>
</div>
</form>
</body>
</html>
```

sqlChecksServlet.java

```
package project;


import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.SQLException;

import java.sql.Statement;


import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;


/**
* Servlet implementation class sqlChecksServlet
*/
public class sqlChecksServlet extends HttpServlet {
private static final long serialVersionUID = 1L;


   protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
       response.setContentType("text/html");
       PrintWriter out = response.getWriter();


       String url = "jdbc:mysql://localhost:3306/";
       String dbName = "vehicle_sales_db";
       String driver = "com.mysql.cj.jdbc.Driver";
       String userName = "root";
       String password = "password";


       try {
           Class.forName(driver);
          Connection conn = DriverManager.getConnection(url, userName, password);
          Statement stmt = conn.createStatement();


          // Create database if not exists
          String createDatabaseQuery = "CREATE DATABASE IF NOT EXISTS " + dbName;
```

```
stmt.executeUpdate(createDatabaseQuery);


// Switch to the created or existing database

String useDatabaseQuery = "USE " + dbName;

stmt.executeUpdate(useDatabaseQuery);


// Create tables if not exist

String createVehicleTableQuery = "CREATE TABLE IF NOT EXISTS vehicle ("

        + "vehicle_id INT AUTO_INCREMENT PRIMARY KEY,"

        + "make VARCHAR(255) NOT NULL,"

        + "model VARCHAR(255) NOT NULL,"

        + "year INT NOT NULL,"

        + "price DECIMAL(10,2) NOT NULL,"

        + "owner VARCHAR(255),"

        + "engine_number VARCHAR(255),"

        + "chassis_number VARCHAR(255),"

        + "insurance_coverage_status VARCHAR(255)"

        + ")";

stmt.executeUpdate(createVehicleTableQuery);


String createTransactionTableQuery = "CREATE TABLE IF NOT EXISTS transaction ("

        + "transaction_id INT AUTO_INCREMENT PRIMARY KEY,"

        + "amount DECIMAL(10,2) NOT NULL,"

        + "transaction_date DATE NOT NULL,"

        + "profit DECIMAL(10,2),"

        + "vehicle_id INT,"
```

```
            + "FOREIGN KEY (vehicle_id) REFERENCES vehicle(vehicle_id)"
            + ")";
        stmt.executeUpdate(createTransactionTableQuery);

        String createUsersTableQuery = "CREATE TABLE IF NOT EXISTS users ("
            + "user_id INT AUTO_INCREMENT PRIMARY KEY,"
            + "name VARCHAR(255) NOT NULL,"
            + "user_name VARCHAR(255) UNIQUE NOT NULL,"
            + "password VARCHAR(255) NOT NULL,"
            + "role VARCHAR(50) NOT NULL"
            + ")";
        stmt.executeUpdate(createUsersTableQuery);

        String query = "INSERT IGNORE INTO users (name, user_name, password, role) VALUES ('Admin',
'admin', 'admin', 'admin')";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.executeUpdate(query);

        String createAccessoriesTableQuery = "CREATE TABLE IF NOT EXISTS accessories ("
            + "accessories_id INT AUTO_INCREMENT PRIMARY KEY,"
            + "accessories VARCHAR(255) NOT NULL,"
            + "vehicle_id INT,"
            + "FOREIGN KEY (vehicle_id) REFERENCES vehicle(vehicle_id)"
            + ")";
        stmt.executeUpdate(createAccessoriesTableQuery);
```

```
            String createScoreTableQuery = "CREATE TABLE IF NOT EXISTS score ("
                    + "score_id INT AUTO_INCREMENT PRIMARY KEY,"
                    + "vehicle_score INT NOT NULL,"
                    + "insurance_coverage_status VARCHAR(255),"
                    + "vehicle_id INT NOT NULL,"
                    + "FOREIGN KEY (vehicle_id) REFERENCES vehicle(vehicle_id)"
                    + ")";
            stmt.executeUpdate(createScoreTableQuery);


            // Forward the request to loginServlet.jsp
            request.getRequestDispatcher("loginServlet").forward(request, response);
            stmt.close();
            conn.close();
        } catch (ClassNotFoundException | SQLException e) {
            out.println("<html><body><b>Error: " + e.getMessage() + "</b></body></html>");
        }
    }
}
```

loginServlet.java

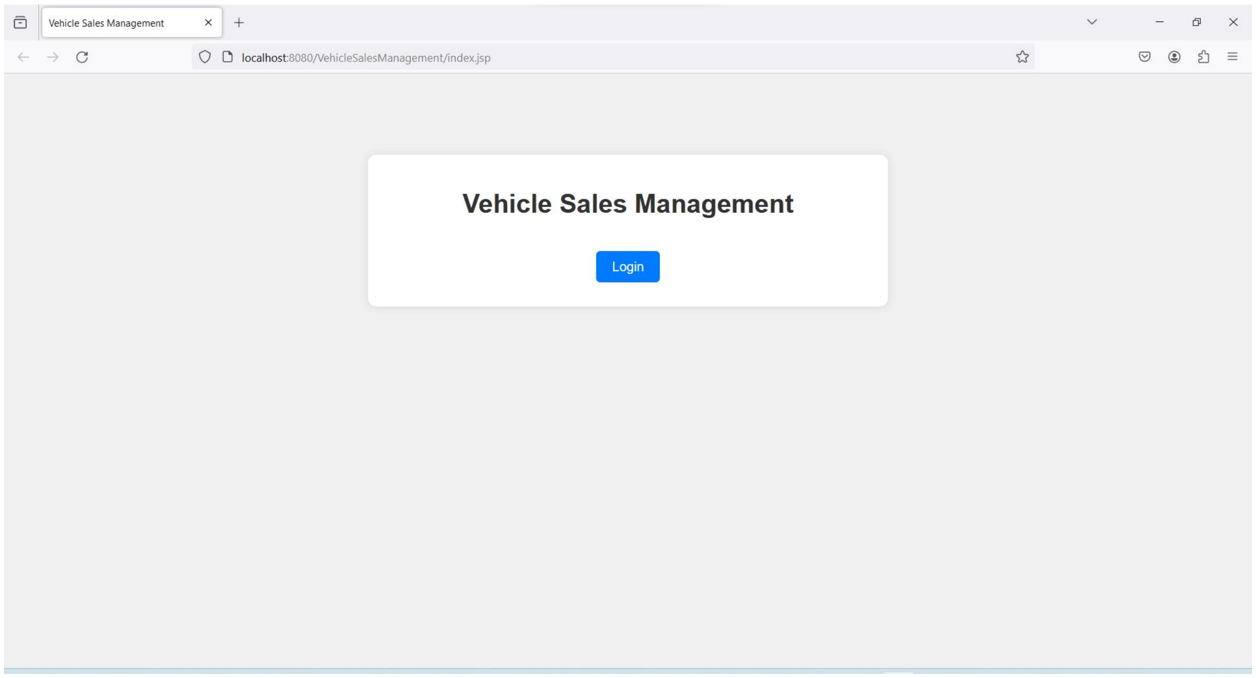
```
package project;


import java.io.*;

import java.sql.*;


import jakarta.servlet.ServletException;

import jakarta.servlet.annotation.WebServlet;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import java.io.IOException;


/**

* Servlet implementation class loginServlet

*/

public class loginServlet extends HttpServlet {

private static final long serialVersionUID = 1L;



protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    response.setContentType("text/html");

    PrintWriter out = response.getWriter();
```

```
        String url = "jdbc:mysql://localhost:3306/vehicle_sales_db";

        String dbName = "vehicle_sales_db";

        String driver = "com.mysql.cj.jdbc.Driver";

        String userName = "root";

        String password = "password";


        String username = request.getParameter("username");

        String pass = request.getParameter("password");


        try {

            Class.forName(driver);

            Connection conn = DriverManager.getConnection(url, userName, password);

            Statement stmt = conn.createStatement();

            ResultSet rs = stmt.executeQuery("SELECT * FROM users WHERE user_name='" + username + "'
AND password='" + pass + "'");


            if (rs.next()) {

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

                if (role.equals("sales")) {

                    request.getRequestDispatcher("sales.jsp").forward(request, response);

                } else if (role.equals("manager")) {

                    request.getRequestDispatcher("manager.jsp").forward(request, response);

                } else if (role.equals("admin")) {

                    request.getRequestDispatcher("admin.jsp").forward(request, response);

                }

            } else {
```

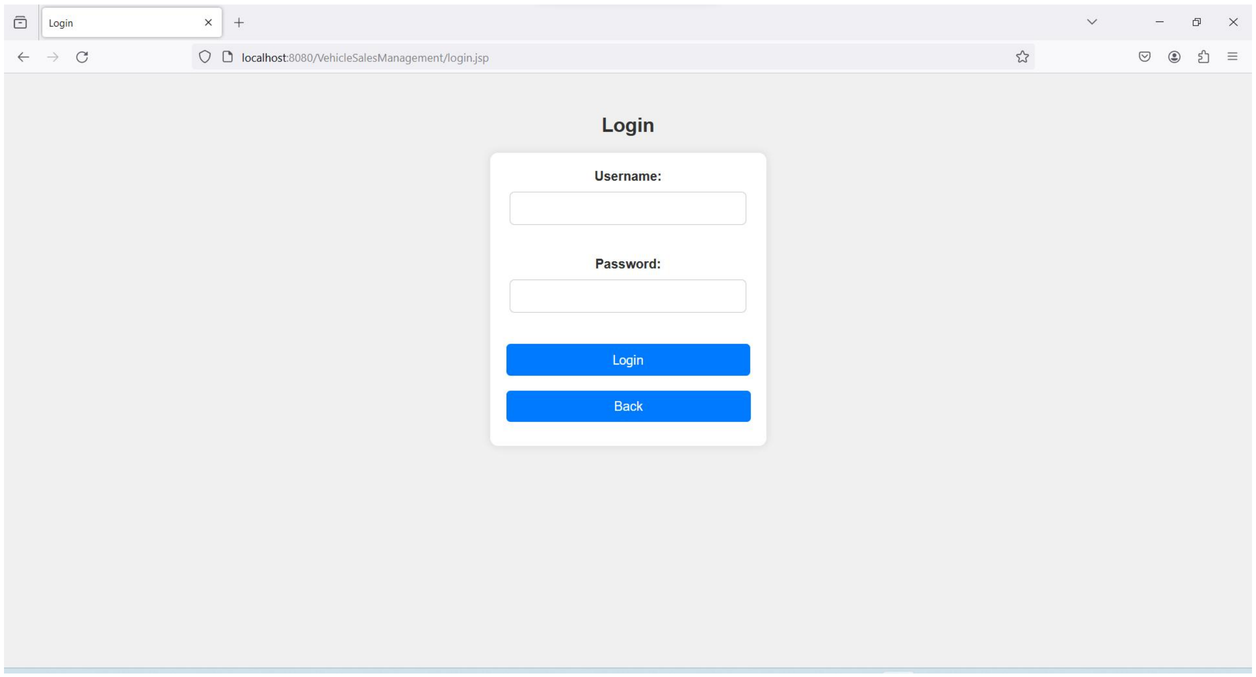
```
            request.setAttribute("failureMessage", "Incorrect username/password!");

            request.getRequestDispatcher("login.jsp").include(request, response);

        }


        rs.close();

        stmt.close();

        conn.close();

    } catch (ClassNotFoundException | SQLException e) {

        out.println("<html><body><b>Error: " + e.getMessage() + "</b></body></html>");

    }

  }

}
```

Kindly refer to the VehicleSalesManagement.zip file for the remaining code.

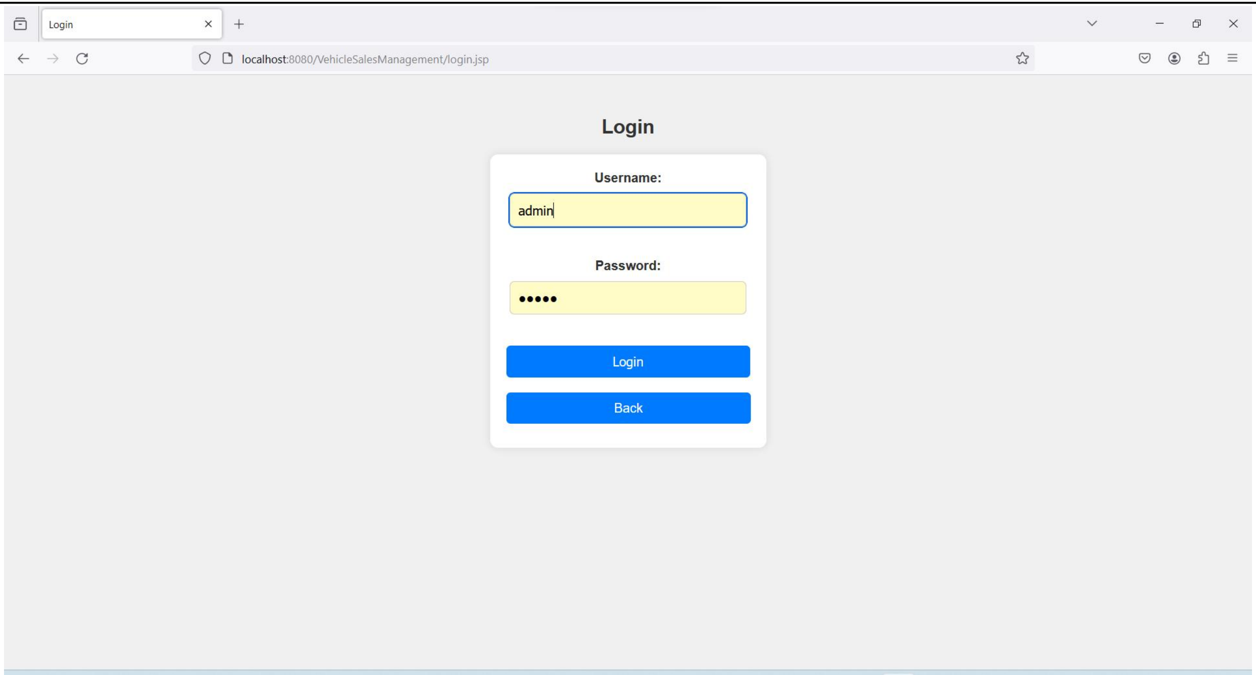
PROJECT OUTPUTS:

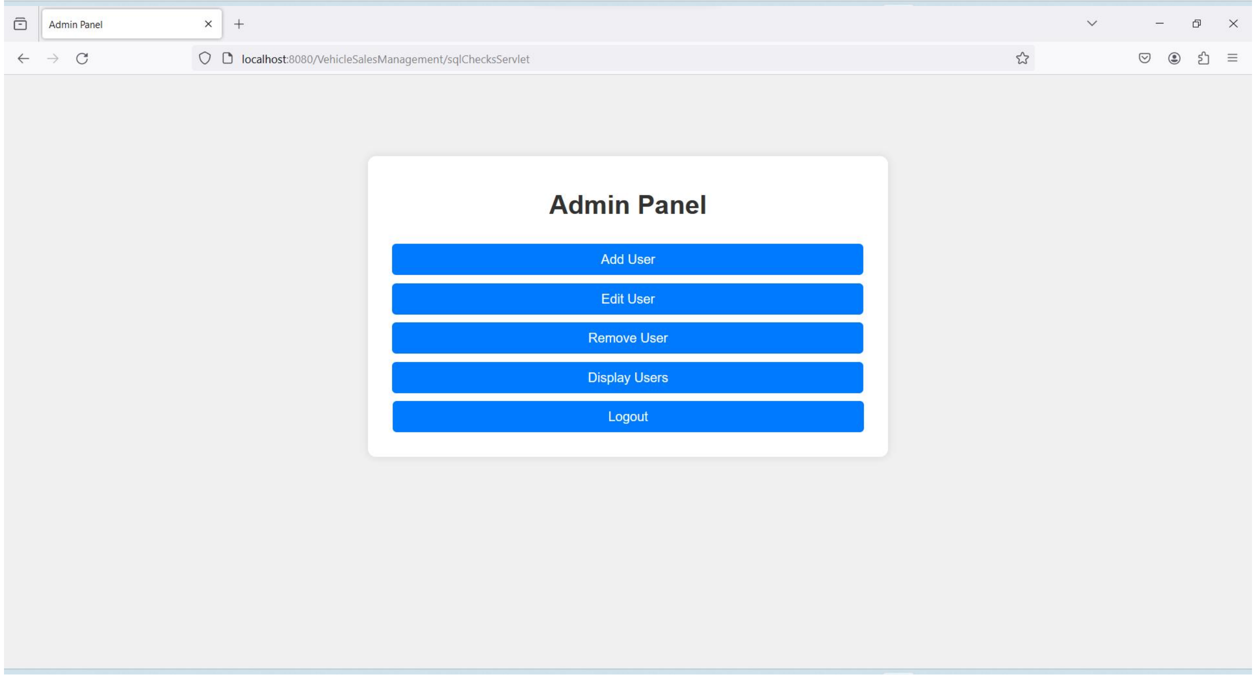


Above is the index.jsp file which upon clicking Login redirects to login.jsp.



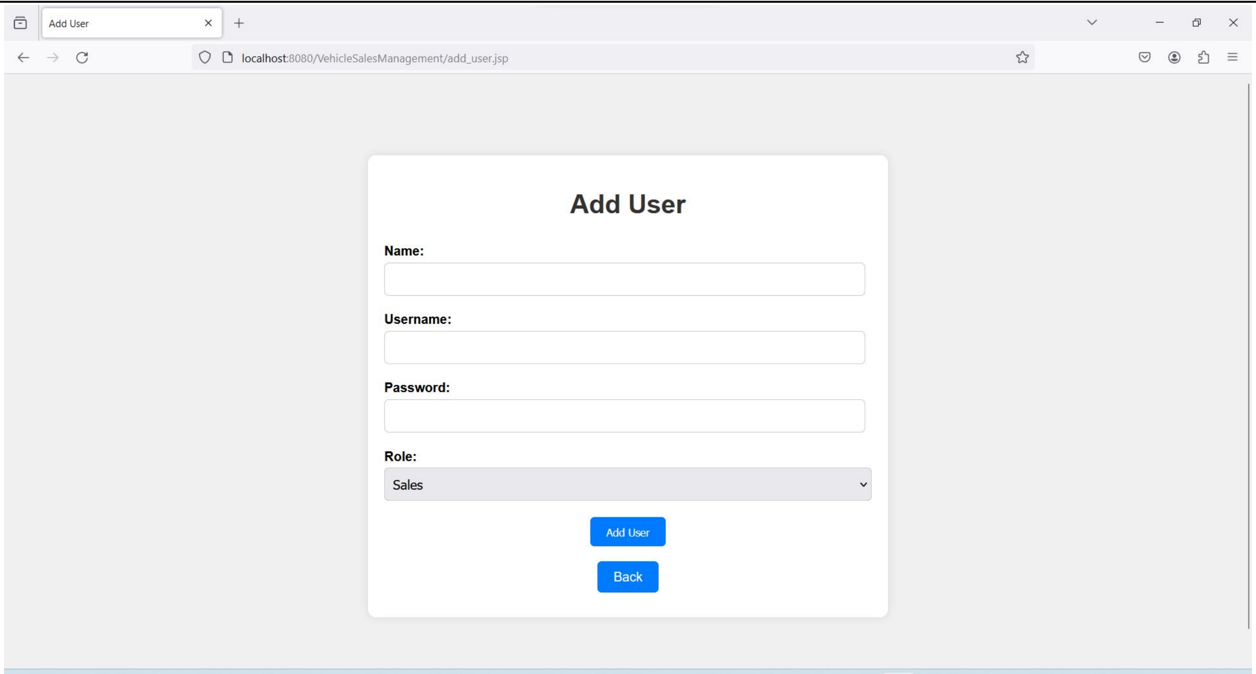
The above screenshot shows the login page where the user is redirected to their respective dashboard based on their user role which is retrieved from the users database.

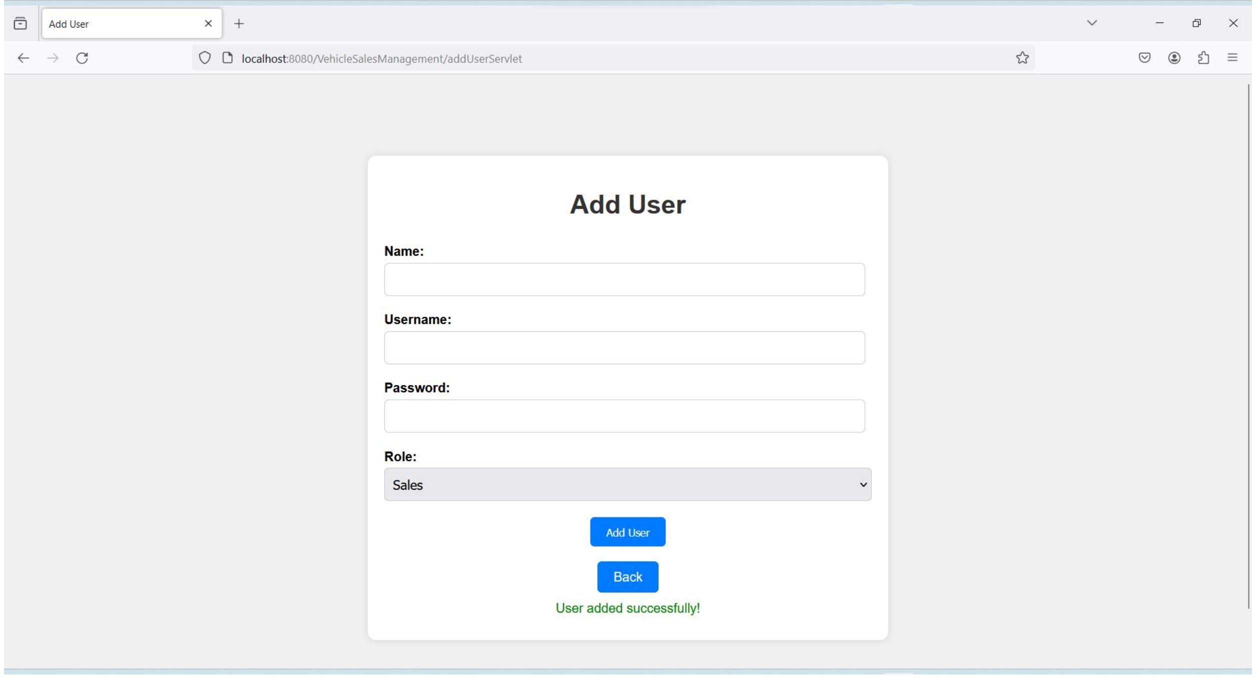




The admin has the functionalities:

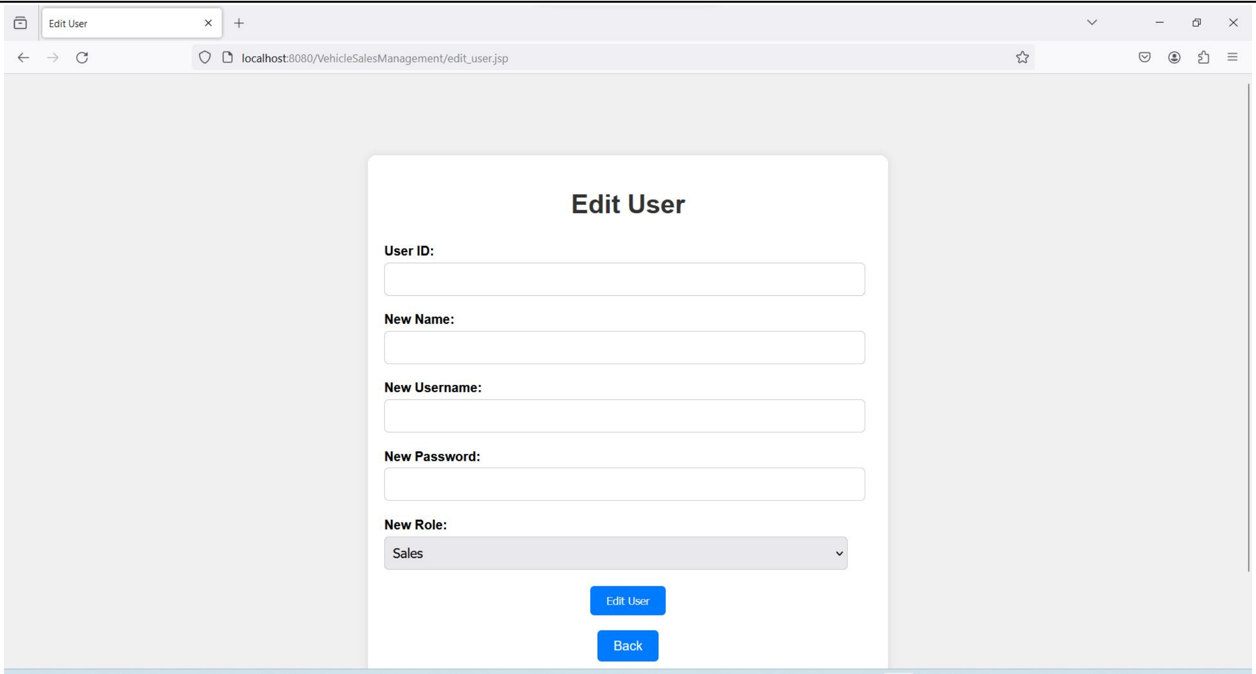
1. Add user
2. Edit user
3. Remove user
4. Display user
5. Logout





User added successfully!

Upon adding the user, a success message is displayed to notify the successful addition of the user. Similarly, a failure message will also be displayed upon unsuccessful addition of the user.


Edit User

localhost:8080/VehicleSalesManagement/edit_user.jsp

Edit User

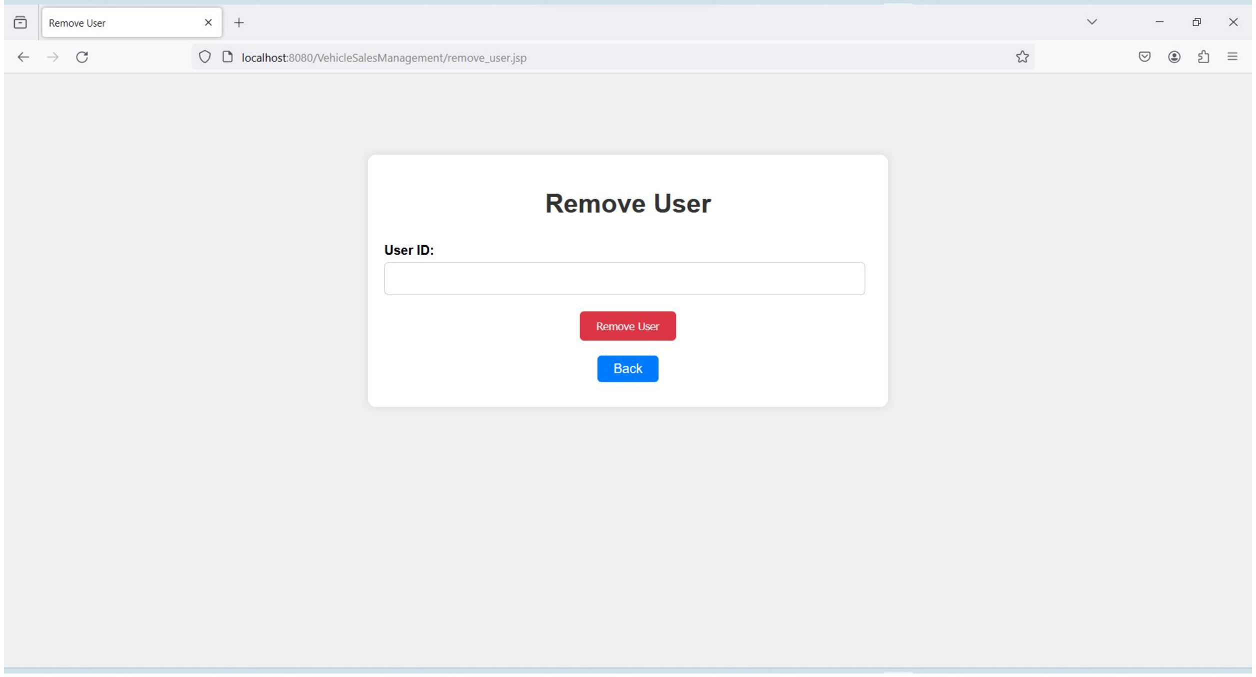
User ID:

New Name:

New Username:

New Password:

New Role:
Sales

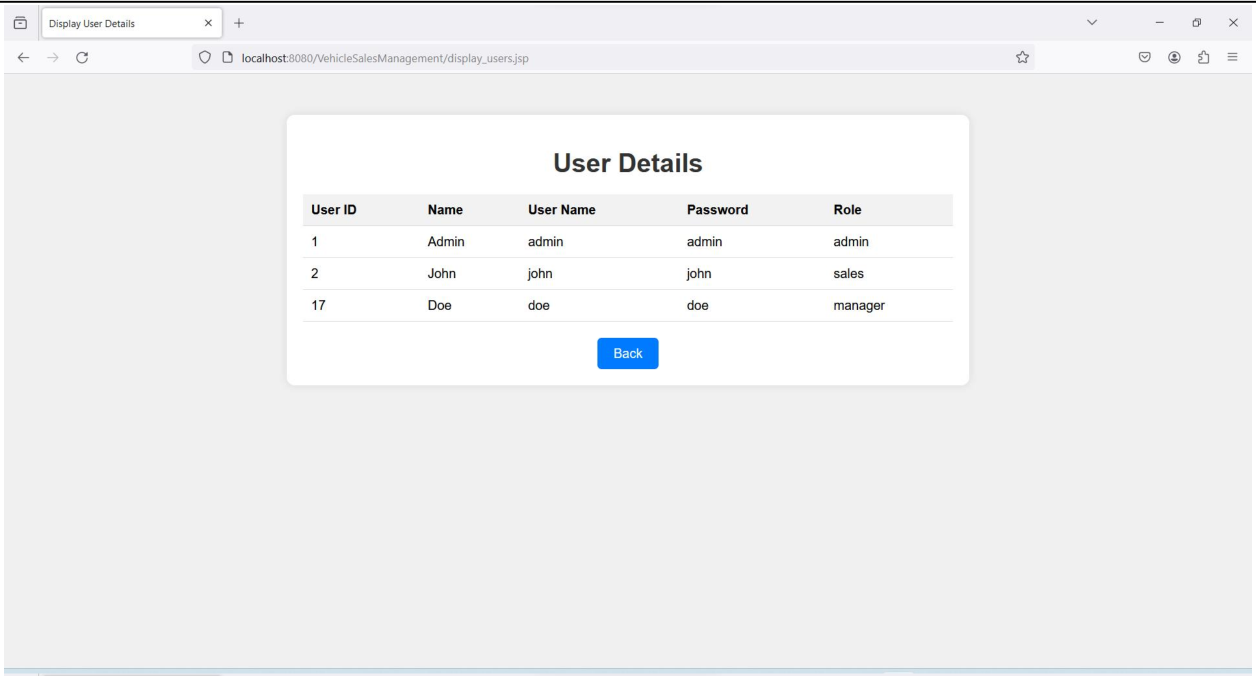

Remove User

localhost:8080/VehicleSalesManagement/remove_user.jsp

Remove User

User ID:

Removing a user is done by taking the user id from the admin and removing respective record from users table.

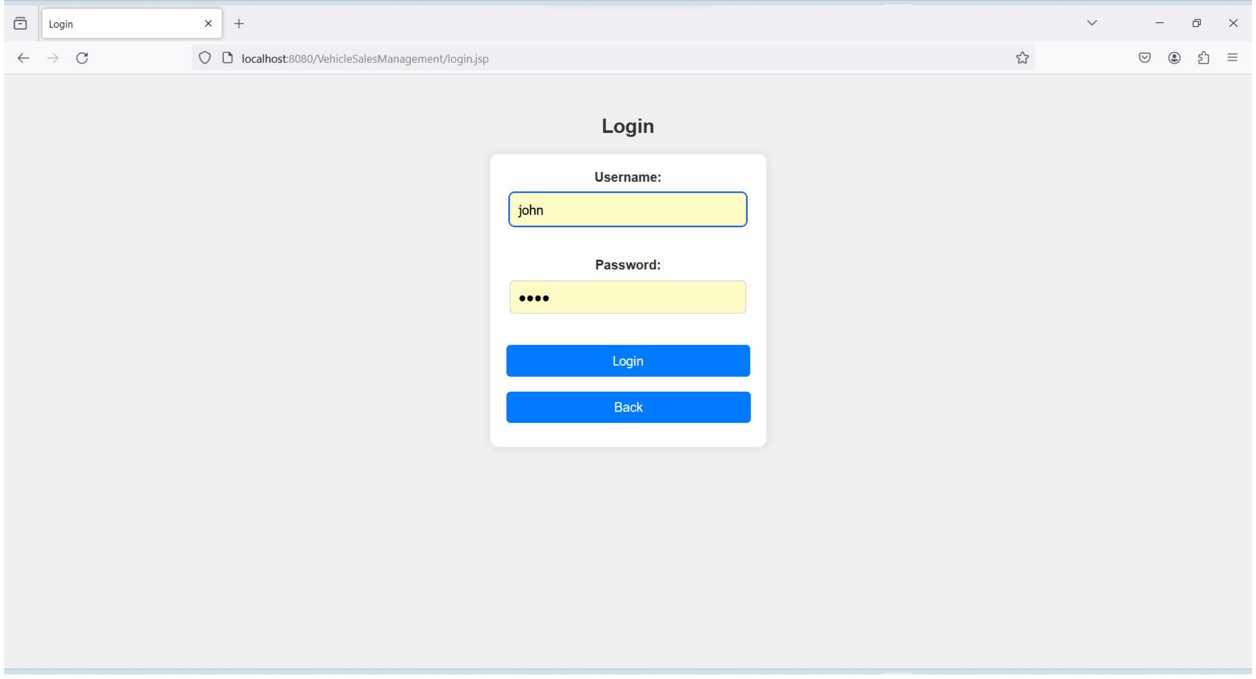


Display User Details

localhost:8080/VehicleSalesManagement/display_users.jsp

User Details

User ID	Name	User Name	Password	Role
1	Admin	admin	admin	admin
2	John	john	john	sales
17	Doe	doe	doe	manager



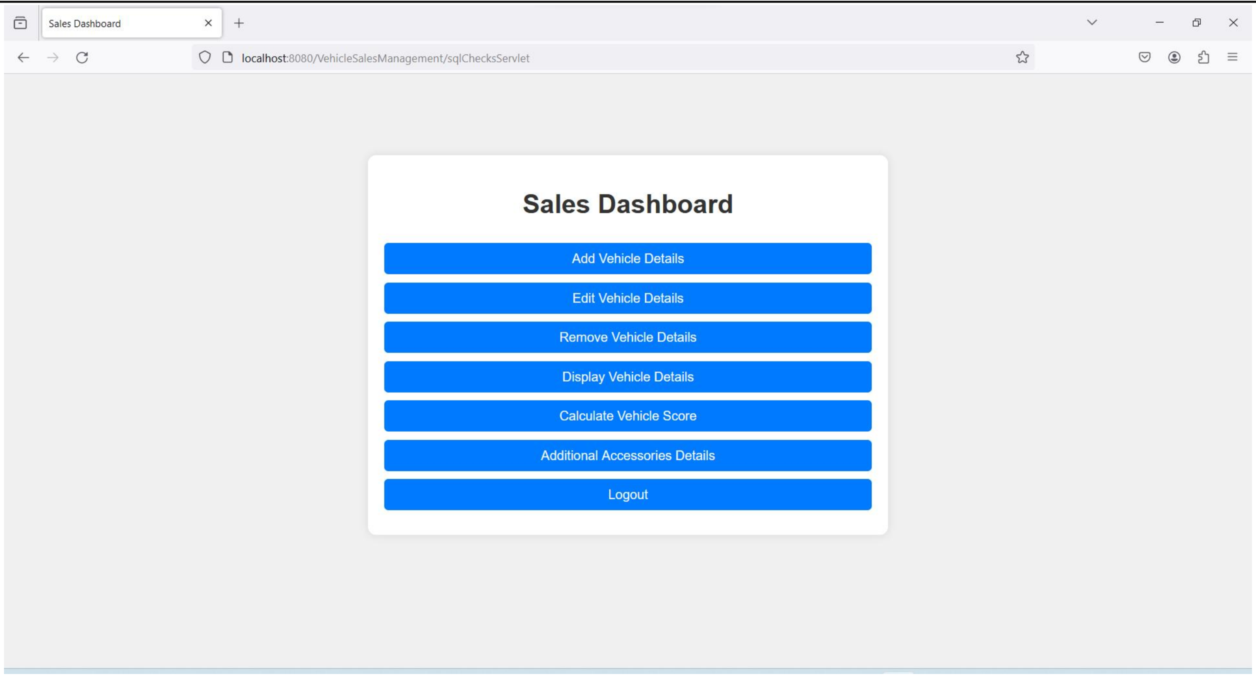
Login

localhost:8080/VehicleSalesManagement/login.jsp

Login

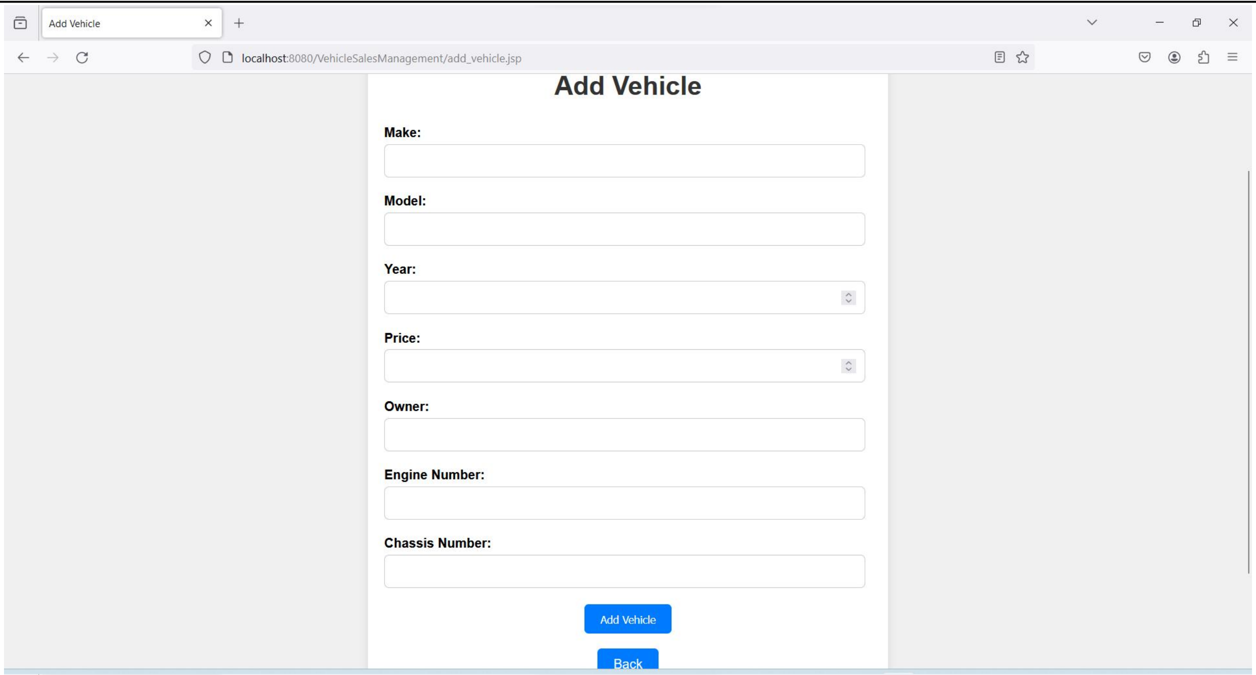
Username: john

Password: ••••



The sales have the functionalities:

1. Add vehicle details
2. Edit vehicle details
3. Remove vehicle details
4. Display vehicle details
5. Calculate vehicle score – based on which insurance coverage status changes to yes if score>90 and no if score<=90.
6. Additional accessories details
   a. Add accessory
   b. Edit accessory
   c. Remove accessory
   d. Display accessories
7. Logout


Add Vehicle

localhost:8080/VehicleSalesManagement/add_vehicle.jsp

Add Vehicle

Make:

Model:

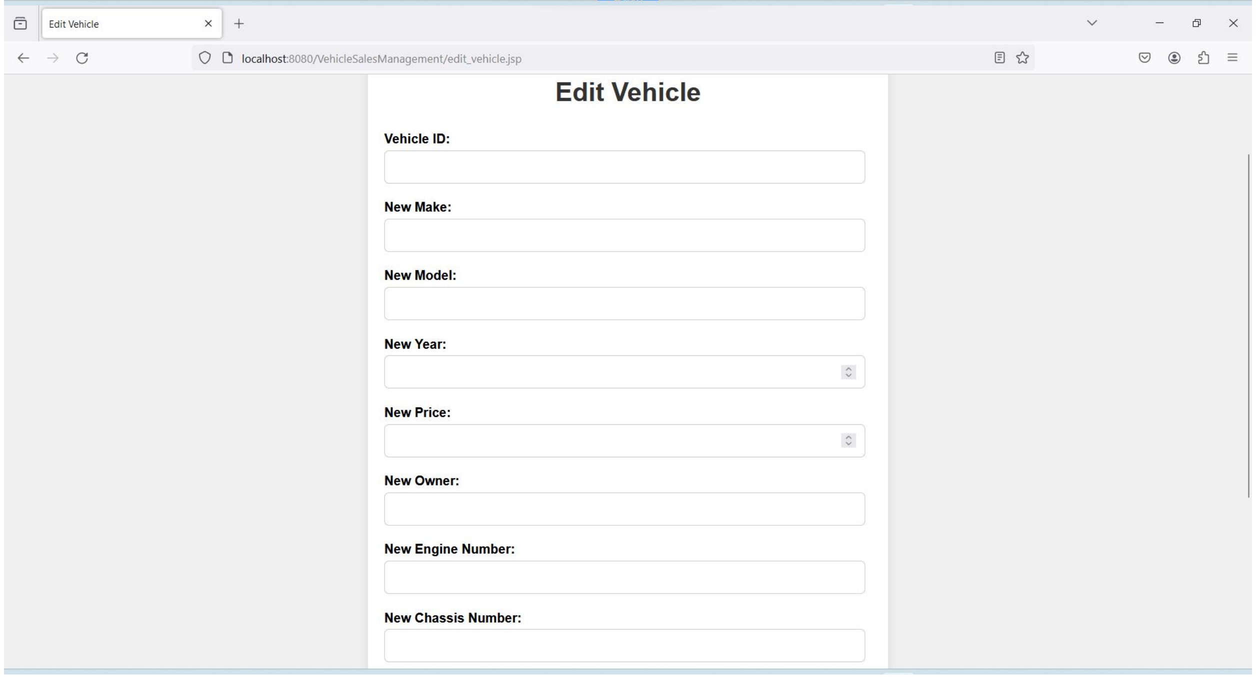
Year:

Price:

Owner:

Engine Number:

Chassis Number:


Edit Vehicle

localhost:8080/VehicleSalesManagement/edit_vehicle.jsp

Edit Vehicle

Vehicle ID:

New Make:

New Model:

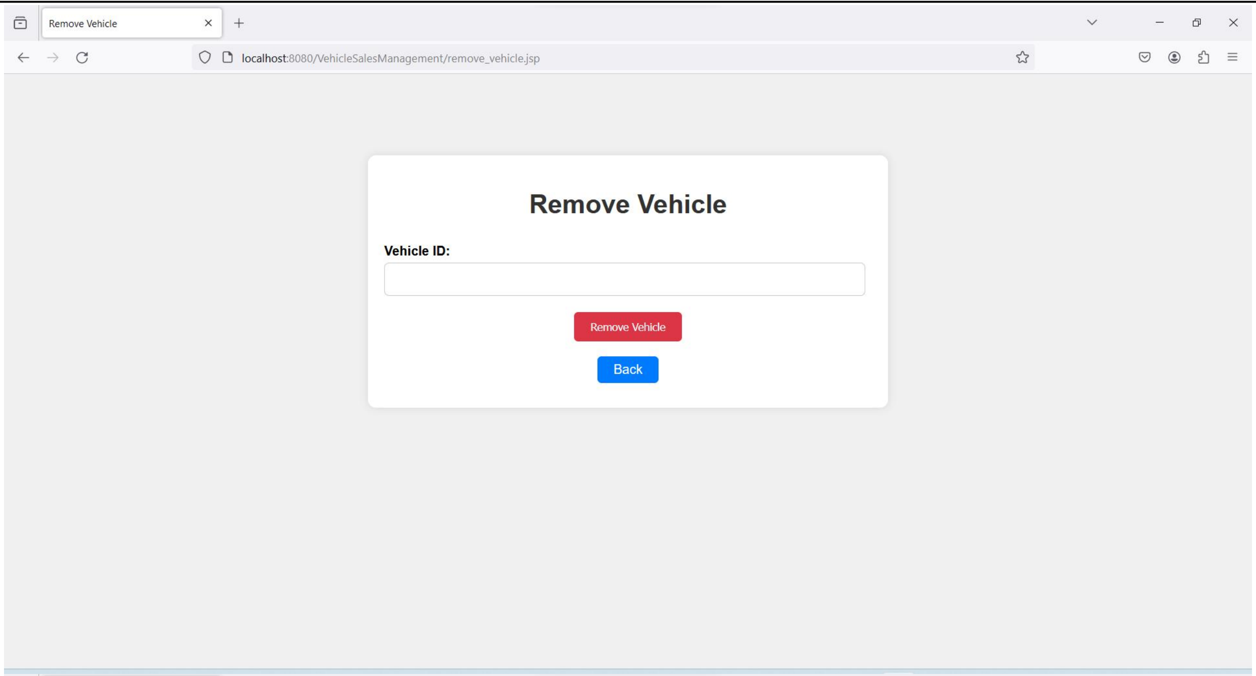
New Year:

New Price:

New Owner:

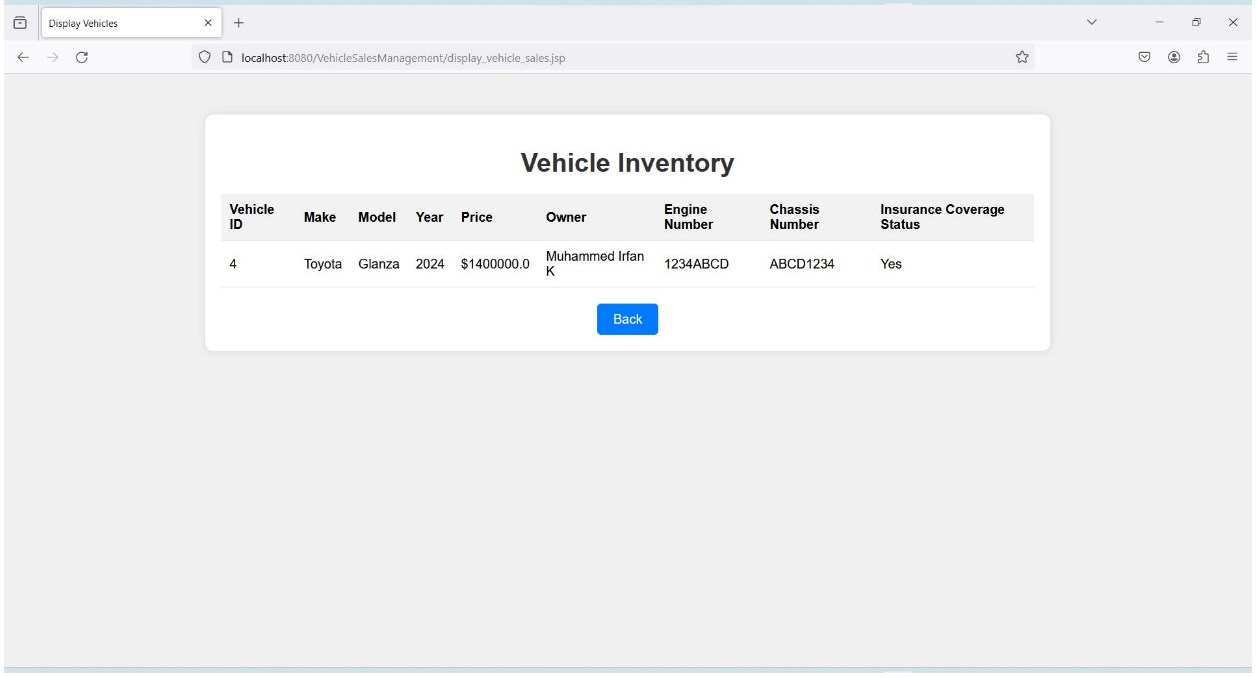
New Engine Number:

New Chassis Number:



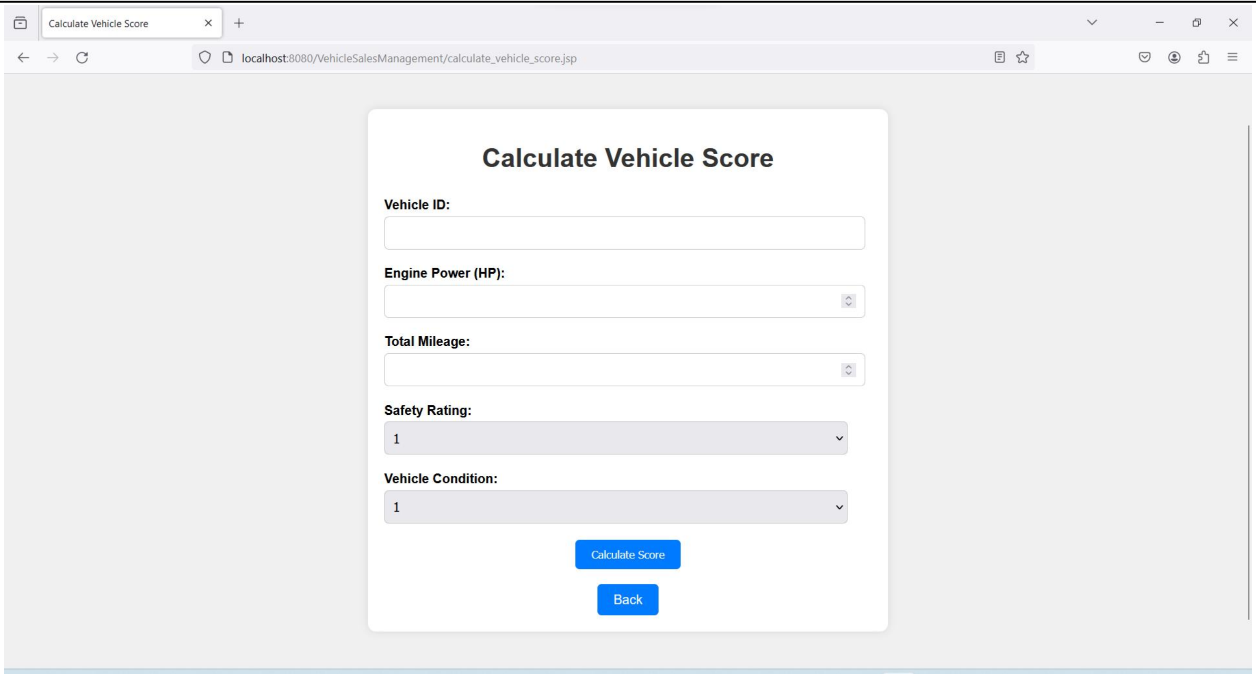
Remove Vehicle

Vehicle ID:


Display Vehicles
localhost:8080/VehicleSalesManagement/display_vehicle_sales.jsp

Vehicle Inventory

Vehicle ID	Make	Model	Year	Price	Owner	Engine Number	Chassis Number	Insurance Coverage Status
4	Toyota	Glanza	2024	$1400000.0	Muhammed Irfan K	1234ABCD	ABCD1234	Yes


Calculate Vehicle Score

localhost:8080/VehicleSalesManagement/calculate_vehicle_score.jsp

Calculate Vehicle Score

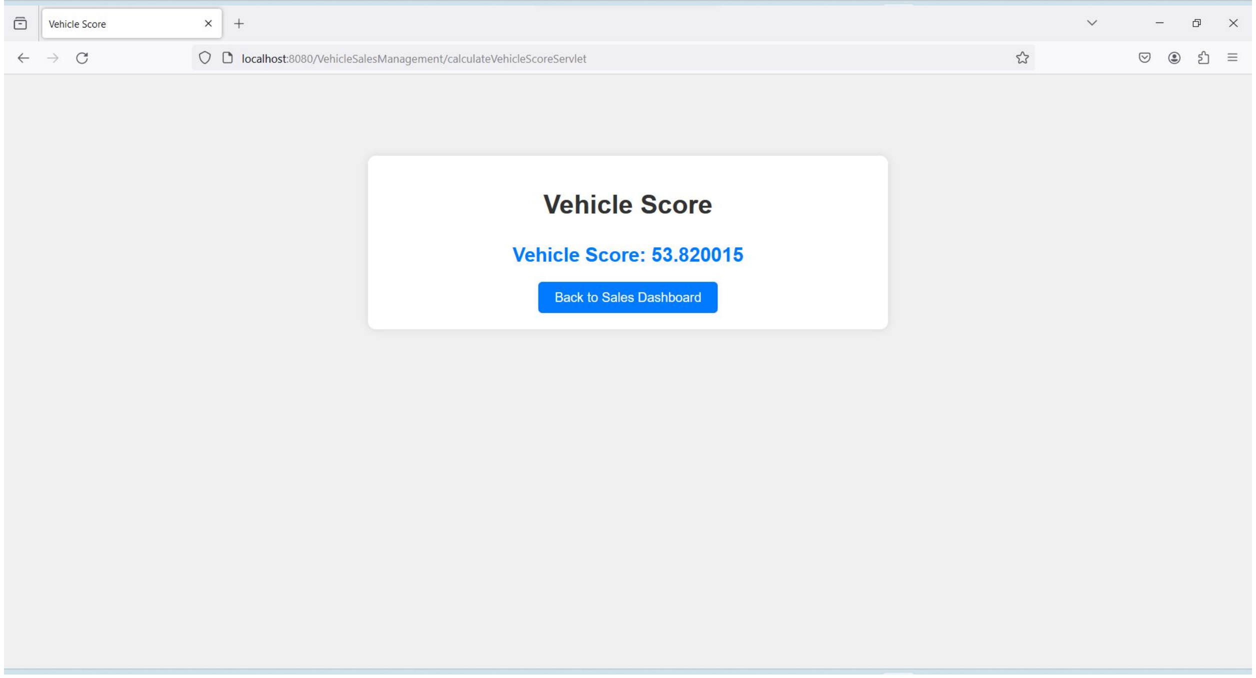
Vehicle ID:

Engine Power (HP):

Total Mileage:

Safety Rating: 1

Vehicle Condition: 1

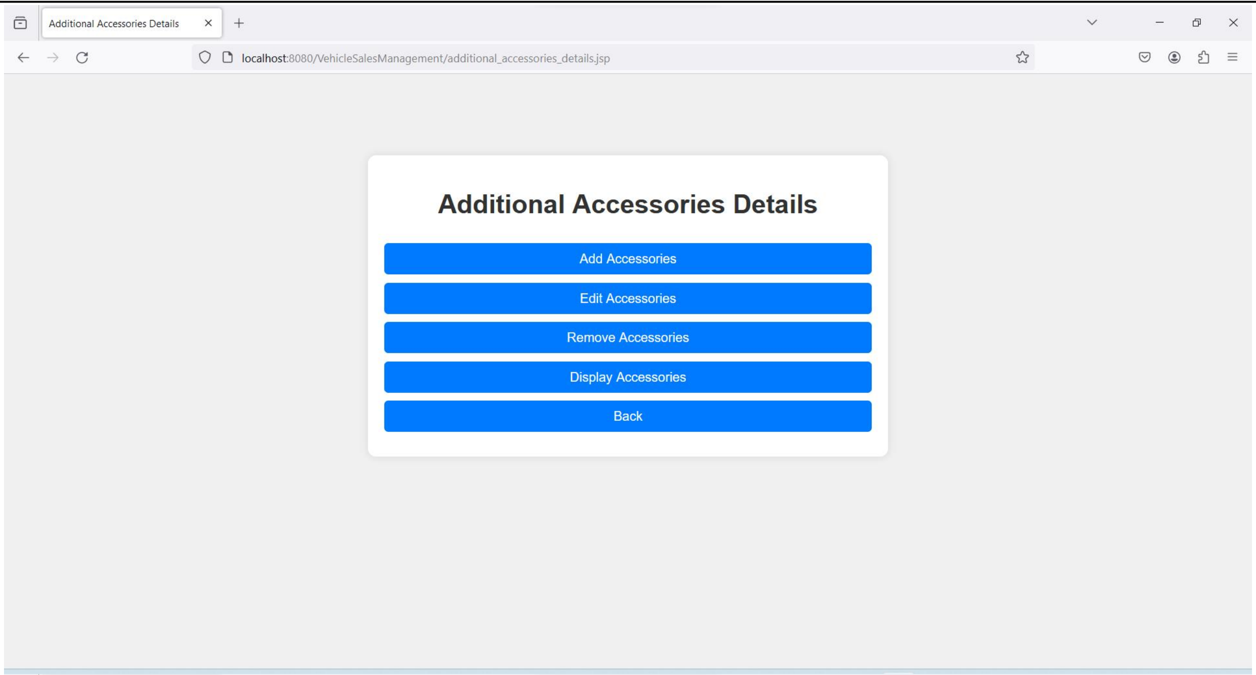

Vehicle Score

localhost:8080/VehicleSalesManagement/calculateVehicleScoreServlet

Vehicle Score

Vehicle Score: 53.820015

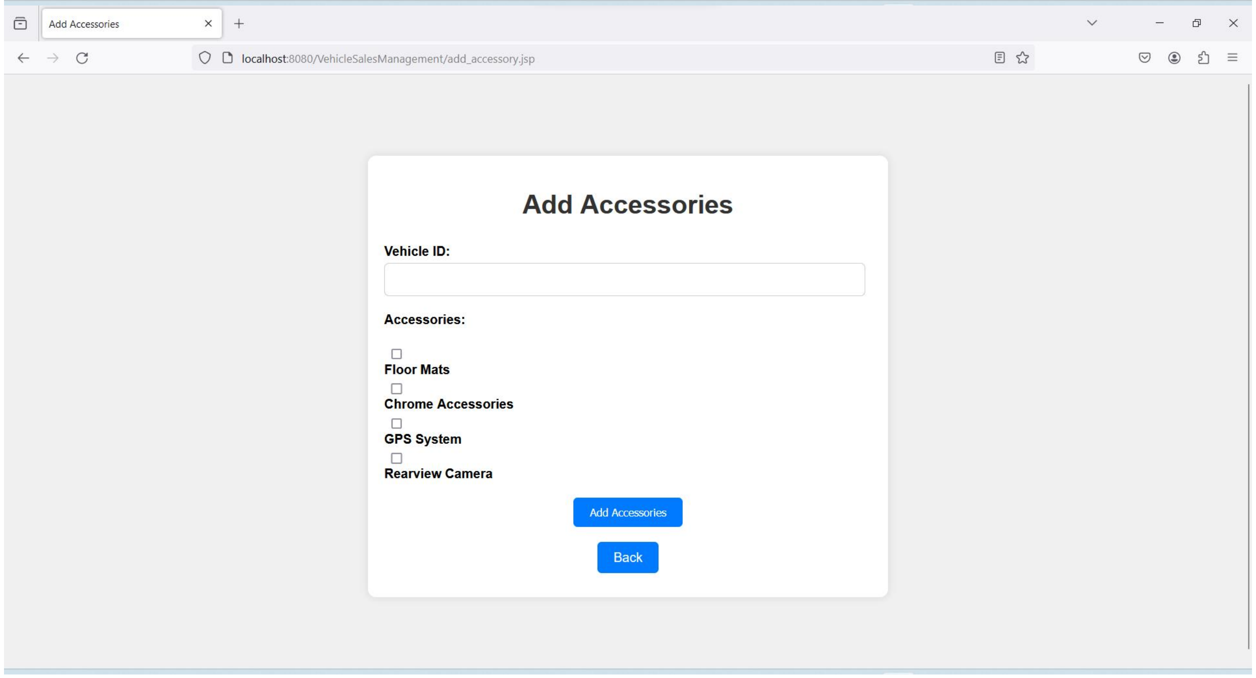
The vehicle score is calculated based on the vehicle's safety rating, year of manufacture, vehicle condition, price of the vehicle, engine power, and total mileage of the vehicle. Additionally, based on the vehicle score, the insurance coverage status changes to "Yes" or "No".



Additional Accessories Details

localhost:8080/VehicleSalesManagement/additional_accessories_details.jsp

Additional Accessories Details



Add Accessories

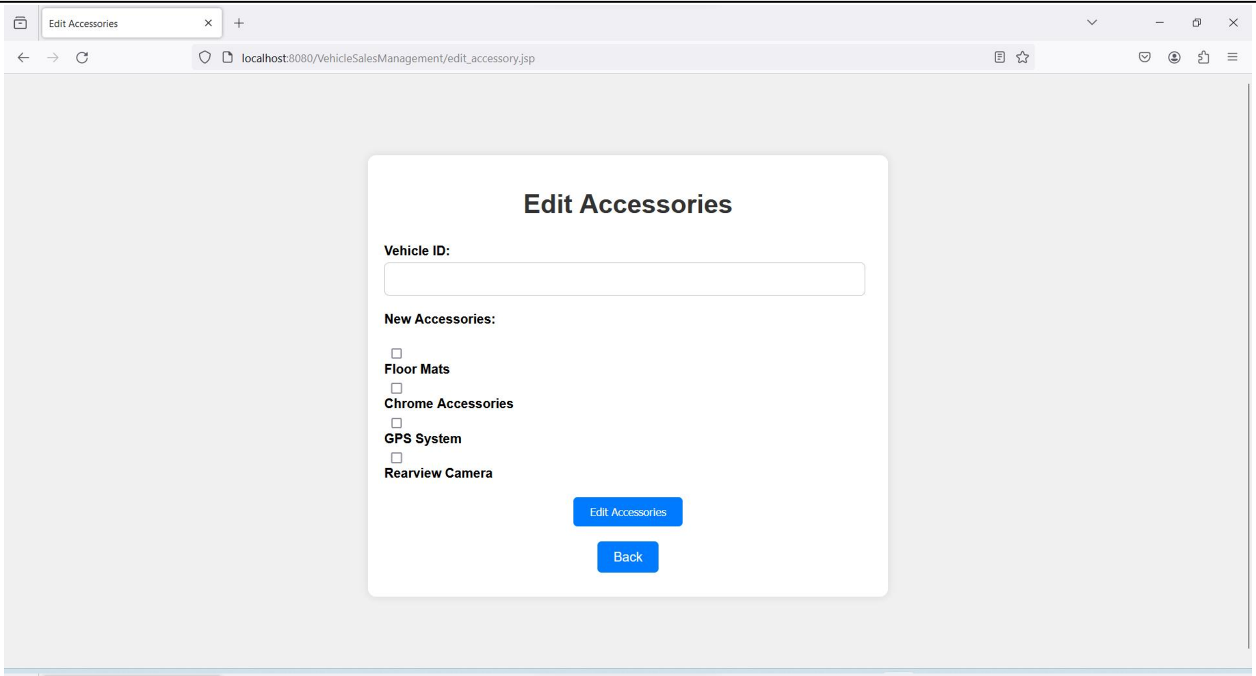
localhost:8080/VehicleSalesManagement/add_accessory.jsp

Add Accessories

Vehicle ID:

Accessories:

- [ ] Floor Mats
- [ ] Chrome Accessories
- [ ] GPS System
- [ ] Rearview Camera



Edit Accessories

localhost:8080/VehicleSalesManagement/edit_accessory.jsp

Edit Accessories

Vehicle ID:

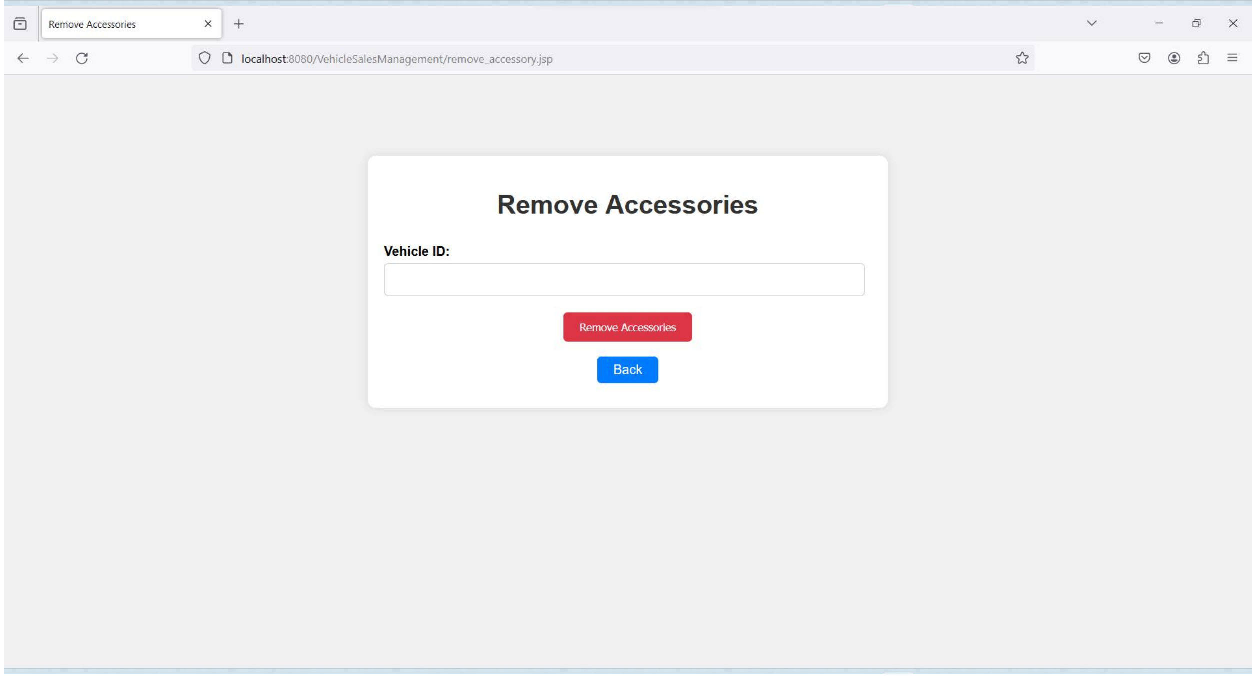
New Accessories:

☐ Floor Mats

☐ Chrome Accessories

☐ GPS System

☐ Rearview Camera

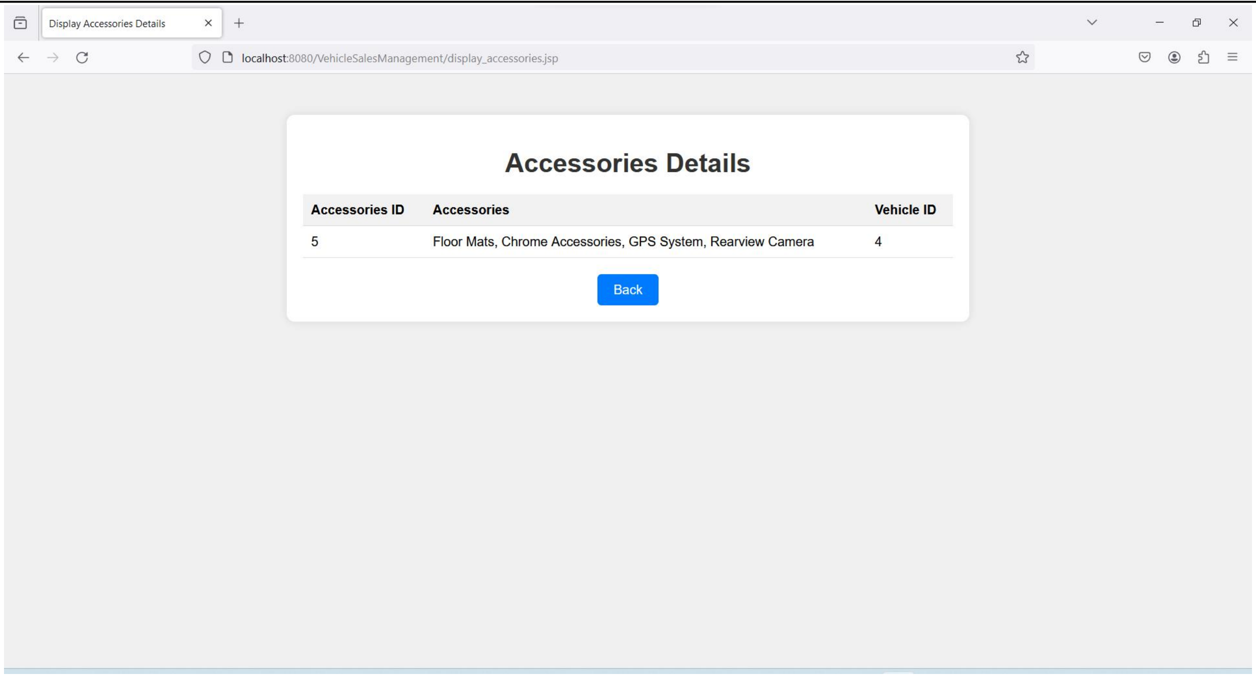


Remove Accessories

localhost:8080/VehicleSalesManagement/remove_accessory.jsp

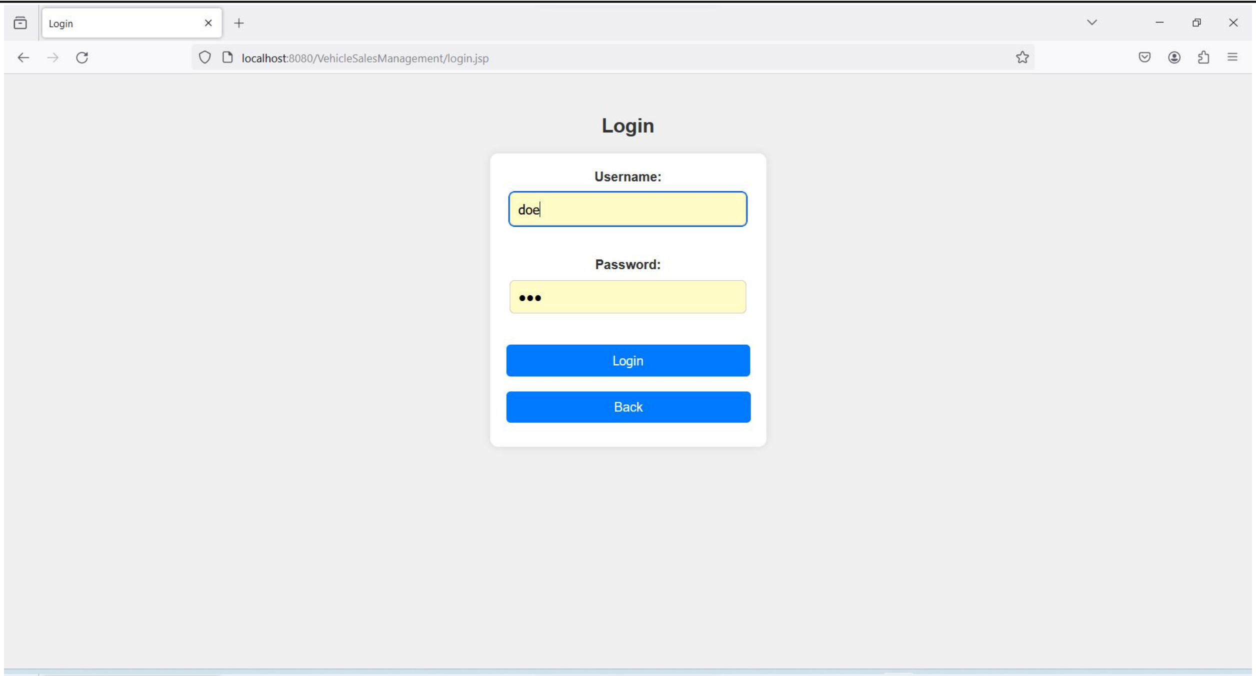
Remove Accessories

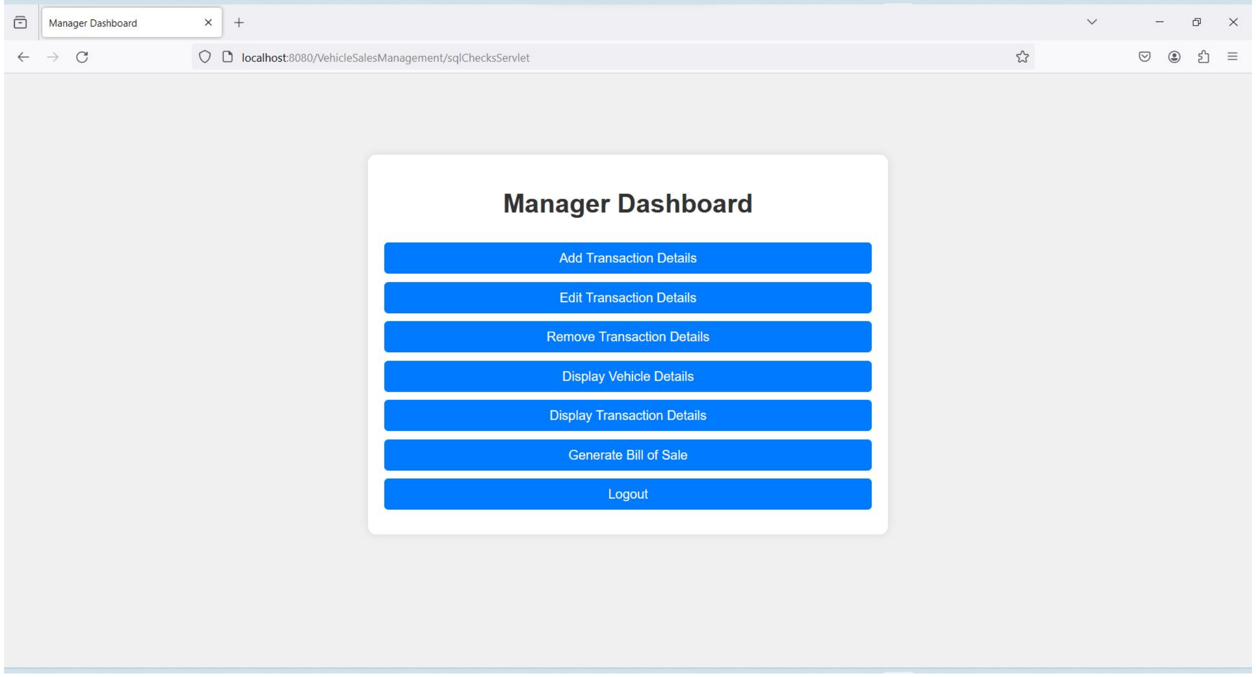
Vehicle ID:


Display Accessories Details
localhost:8080/VehicleSalesManagement/display_accessories.jsp

Accessories Details

Accessories ID	Accessories	Vehicle ID
5	Floor Mats, Chrome Accessories, GPS System, Rearview Camera	4

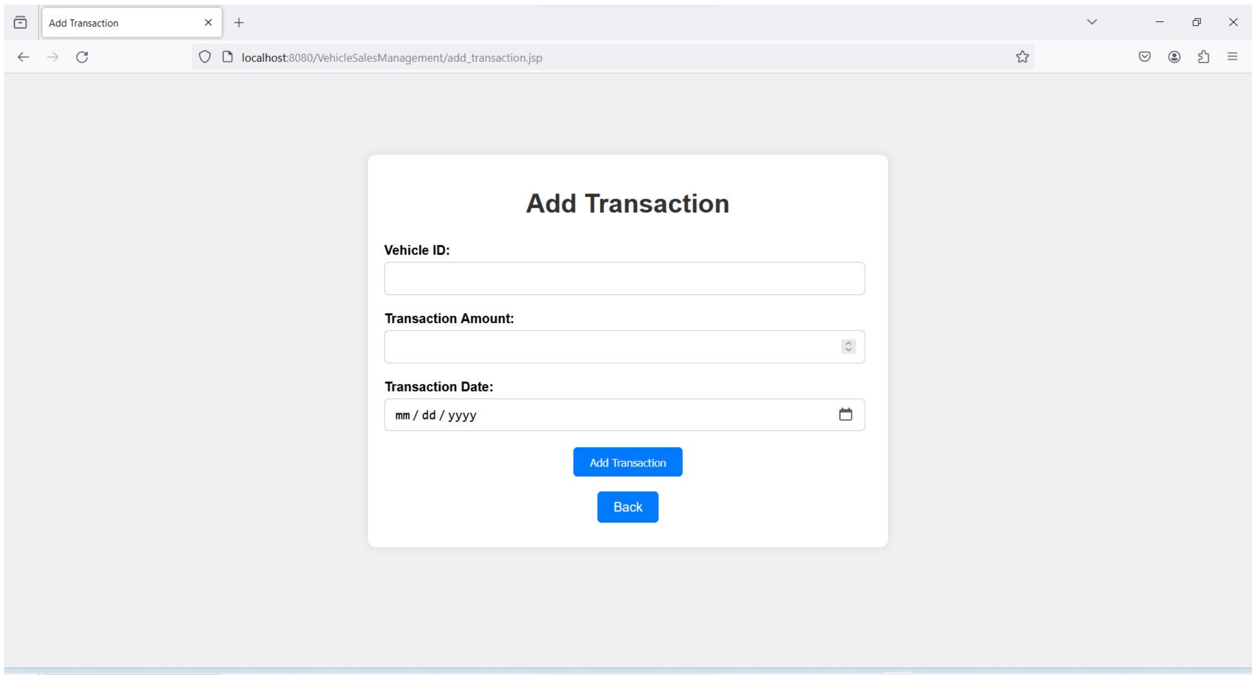


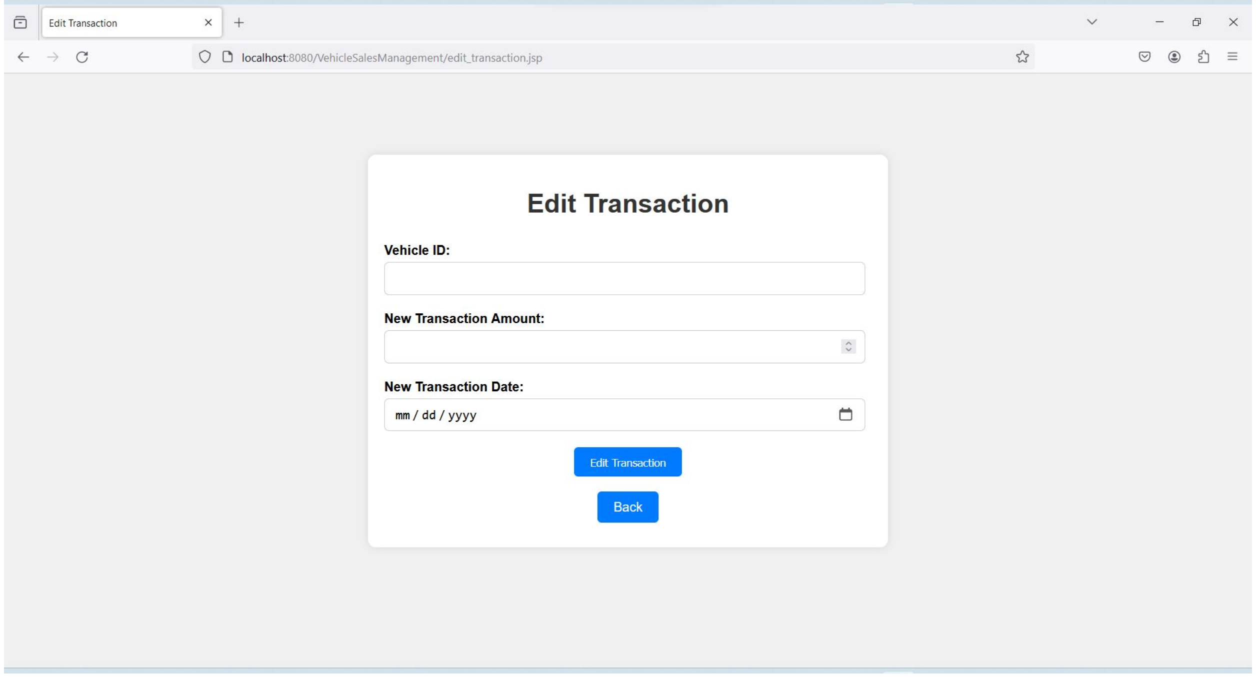


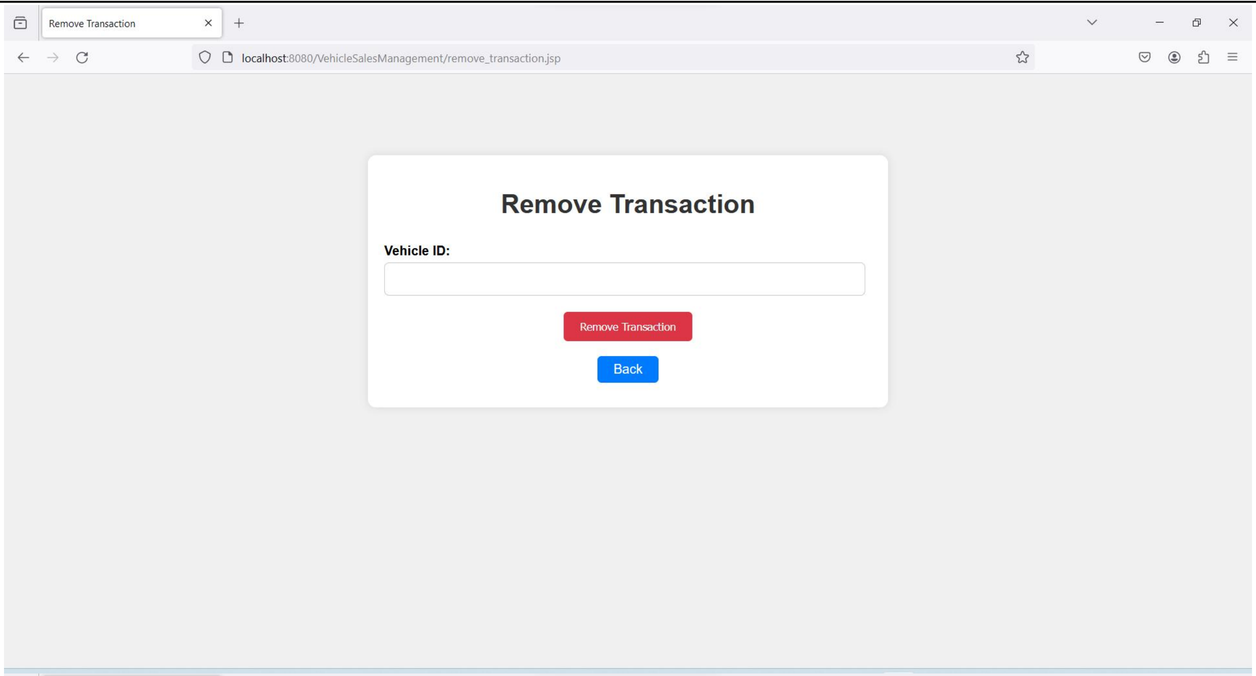
The manager has the functionalities:

1. Add transaction details
2. Edit transaction details
3. Remove transaction details
4. Display vehicle details
5. Display transaction details

6. Generate bill of sale – which creates PDF as well as sends an email to the recipient automatically.
7. Logout






Remove Transaction

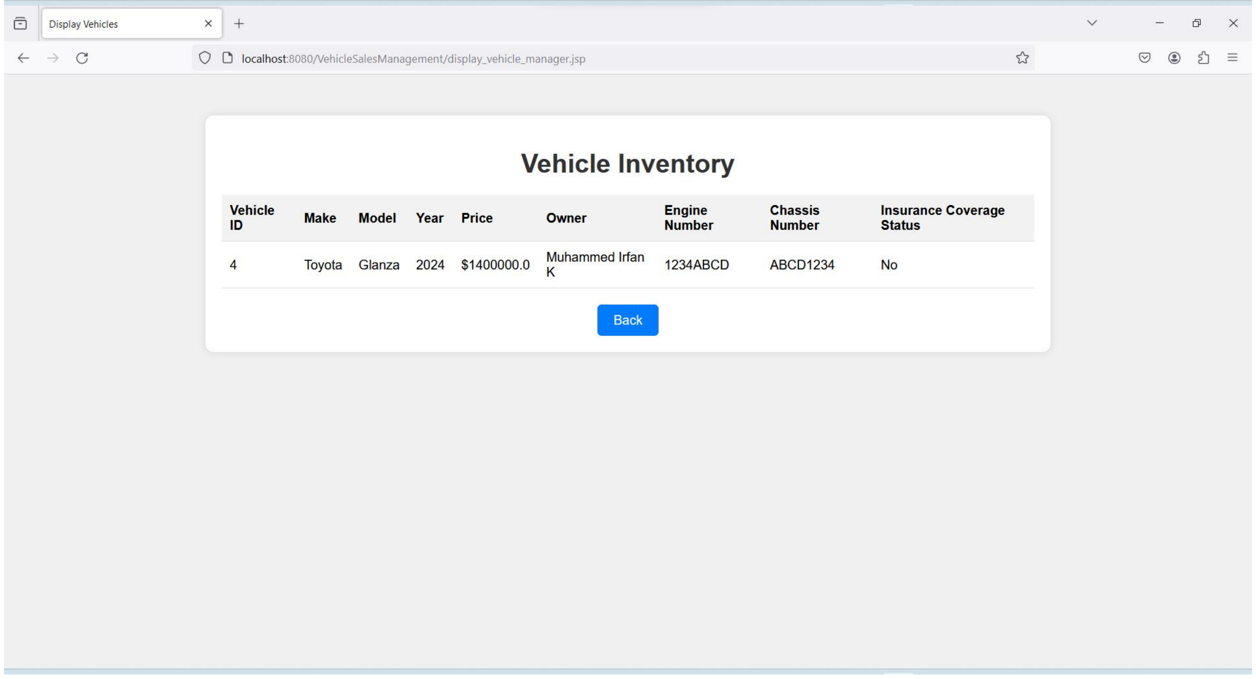
localhost:8080/VehicleSalesManagement/remove_transaction.jsp

Remove Transaction

Vehicle ID:

Remove Transaction

Back

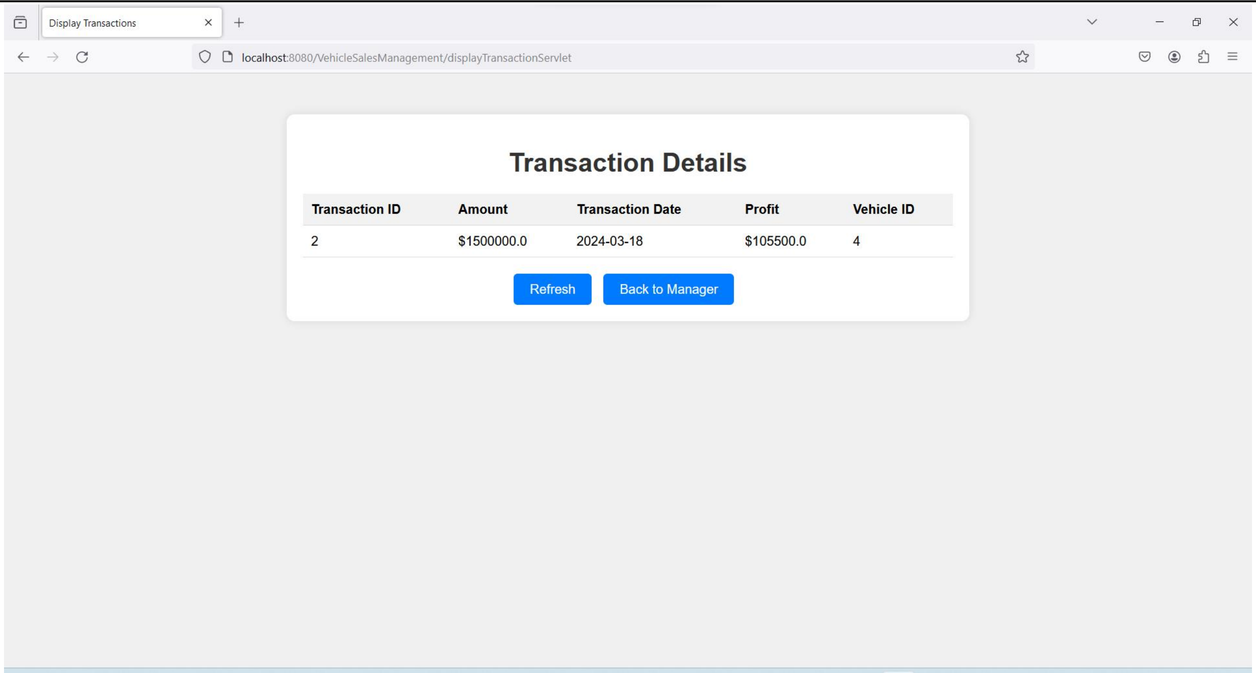

Display Vehicles

localhost:8080/VehicleSalesManagement/display_vehicle_manager.jsp

Vehicle Inventory

Vehicle ID	Make	Model	Year	Price	Owner	Engine Number	Chassis Number	Insurance Coverage Status
4	Toyota	Glanza	2024	$1400000.0	Muhammed Irfan K	1234ABCD	ABCD1234	No

Back

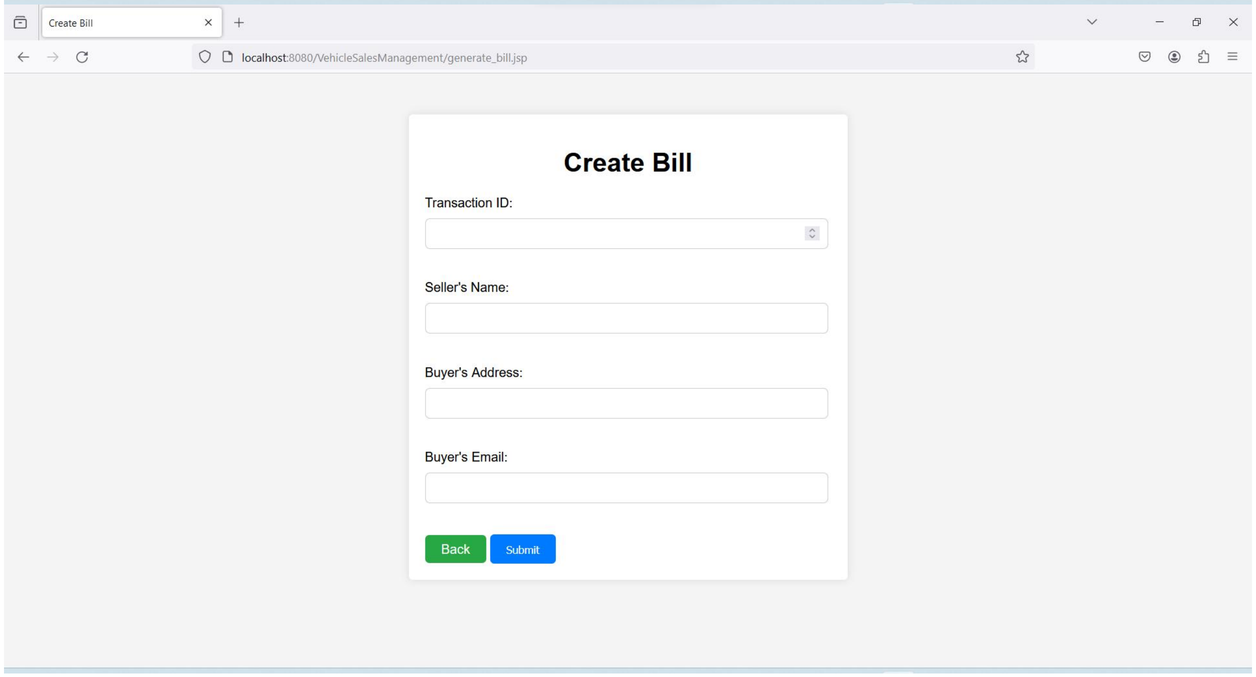


Display Transactions

localhost:8080/VehicleSalesManagement/displayTransactionServlet

Transaction Details

Transaction ID	Amount	Transaction Date	Profit	Vehicle ID
2	$1500000.0	2024-03-18	$105500.0	4



Create Bill

localhost:8080/VehicleSalesManagement/generate_bill.jsp

Create Bill

Transaction ID:

Seller's Name:

Buyer's Address:

Buyer's Email:

Create bill generates a PDF and sends an email to the buyer's email address automatically with the bill attached.

BILL OF SALE

2024-03-18
DATE OF PURCHASE

I, (SELLER) John Doe do hereby

grant, bargain, and convey all rights, title, and interest in and to the automobile

described as follows to:

(BUYER) Muhammed Irfan K

(ADDRESS) Porur, Chennai, TN

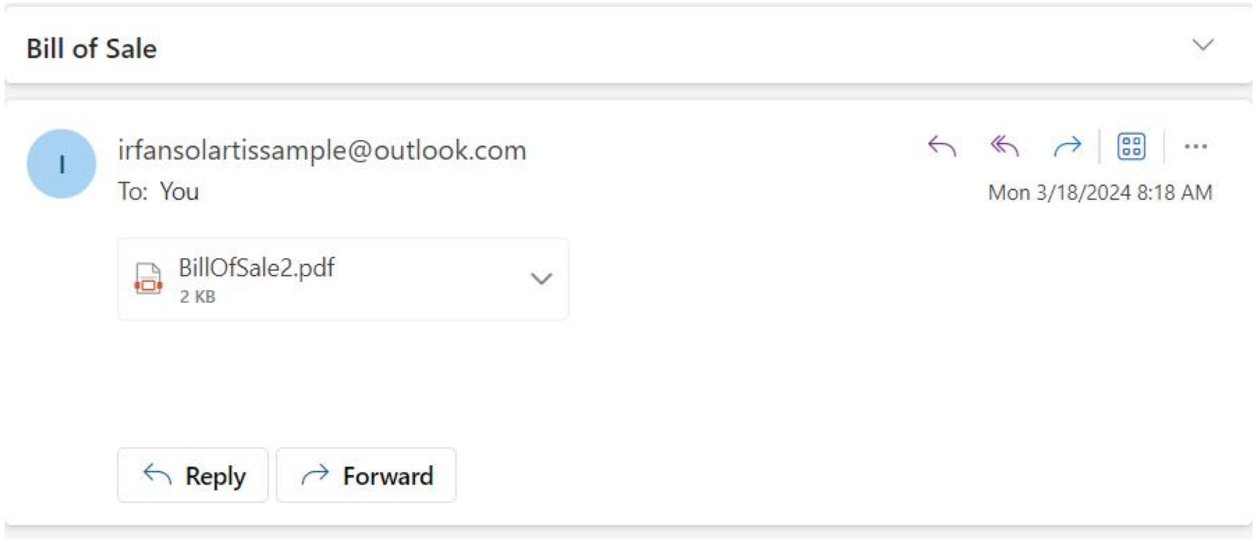
PRICE PAID $1500000.0

DESCRIPTION OF VEHICLE:

Engine Number: 1234ABCD

Chassis Number: ABCD1234

Vehicle (Make, Model, Year): Toyota Glanza (2024)


Bill of Sale

irfansolartissample@outlook.com

To: You

Mon 3/18/2024 8:18 AM

BillOfSale2.pdf
2 KB

CONCLUSION:

The vehicle sales management project offers a robust solution for streamlining the operations involved in managing vehicle sales. By leveraging a relational database, role-based access control, and a user-friendly interface, the system provides salespeople, managers, and administrators with the tools they need to efficiently handle inventory, transactions, user accounts, and more.

Through features such as user authentication, role-based access control, and comprehensive functionalities for vehicle and transaction management, the project empowers users to effectively navigate the complexities of the vehicle sales process. Additionally, the system's ability to calculate vehicle scores and manage additional accessories enhances its utility for optimizing sales strategies and enhancing customer satisfaction.

In summary, the vehicle sales management project not only simplifies day-to-day operations but also contributes to improved productivity, streamlined workflows, and enhanced decision-making capabilities within the context of vehicle sales. With its user-centric design and comprehensive functionalities, the project serves as a valuable asset for businesses and organizations involved in the automotive industry.