#### VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELGAUM



Mini Project report on

## “RENTARIDE”

Submitted in partial fulfillment of the requirements for the 5th semester

**BACHELOR OF ENGINEERING IN**

#### INFORMATION SCIENCE AND ENGINEERING

By

**SANDESH KUNWAR(1SP22IS050)**

**PRATIKSHA THAPA(1SP22IS043)**

**ANKITA PRIYADARSHINI(1SP22IS009)**

**Under the guidance Of**

#### Mrs. Sangeetha K R

##### Professor Dept. of ISE



**Department of Information Science And Engineering**

## S.E.A. COLLEGE OF ENGINEERING AND TECHNOLOGY

**BENGALURU-560049 2023-2024**

## S.E.A COLLEGE OF ENGINEERING ANDTECHNOLOGY

EktaNagar, Basavanpura, Virgonagar Post, K.R.Puram, Bengaluru, Karnataka 560049



#### DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

## CERTIFICATE

This is to certify the project work entitled “**RENTARIDE”** has been successfully implemented by **Mr.SANDESH KUNWAR, Mrs.PRATIKSHA THAPA, Mrs.ANKITA PRIYADARSHINI** bearing **USN 1SP22IS050**, **1SP22IS043** and **1SP22IS009** of V semester in partial fulfillment for the award of **Bachelor of Engineering in Information Science And Engineering** of the **Visvesvaraya Technological University, Belgaum** during the year **2023-24**.The project report has been approved as it satisfied the academic requirement in respect of the mini project work prescribed for Bachelor of Engineering.

Signature of Co-ordinator Signature of HOD

**Prof. D ARUNA Dr.NIJAGUNA G S**

Signature of Internal Examiner Signature of External Examiner

## ACKNOWLEDGEMENT

Firstly, I thank the Management late **Shri A Krishnappa**, Chairman SEA College of Engineering and Technology for Providing Necessary infrastructure and creating good environment.

I would like to express my profound thanks to our respected principal **Dr B VENKATANARAYAN** for the encouragement and support given by him.

I would like to express my profound thanks to our respected Director **Dr DESHPANDEY**

for the encouragement and support given by him

I would like to express my sincere thanks to our respected **Dr NIJAGUNA G S, HOD OF INFORMATION SCIENCE AND ENGINEERING** department, for his assistance and guidance.

I am thankful for the support rendered by my Project guide **Prof. Sangeetha K R** and coordinator **Prof. D ARUNA** for their valuable suggestions.

I am also obliged, to the faculty members of ISE Department who rendered their valuable assistance for the Project.

And, I would like to express my heart full gratitude to my parents who have extended their help throughout my Project.

And finally, I would like to express my heart full gratitude to my friends and all those who have extended their help throughout my Project.

**SANDESH KUNWAR(1SP22IS050)**

**PRATIKSHA THAPA(1SP22IS043)**

**ANKITA PRIYADARSHINI(1SP22IS009)**

## INDEX

#### CHAPTER CONTENTS PAGE NO

###### INTRODUCTION 1

###### LITERATURE SURVEY 2

###### ANALYSIS & REQUIREMENT SPECIFICATION 4

###### SOURCECODE 6

1. SNAPSHOTS 43
2. CONCLUSION 47
3. FUTURE ENHANCEMENT30 48
4. REFERENCE 49

**Chapter 1**

**INTRODUCTION**

**Introduction:**

RentaRide is a convenient and user-friendly app designed to simplify the process of booking bikes and cars for your travel needs. Whether you're looking to rent a bike for a quick ride around town or need a car for a longer journey,it offers a wide selection of vehicles to suit your preferences. The app ensures a seamless experience with secure payment options, allowing you to book and pay with confidence. Additionally, it values user feedback, offering a feature where you can rate your experience and provide reviews, helping future customers make informed decisions.

**Objective:**

The objective of the RentaRide app is to provide a seamless platform for users to rent two-wheelers and four-wheelers conveniently. It offers vehicle booking, rental period selection, pricing details, user feedback, and ratings to enhance the rental experience.

**Functionality:**

* User-Friendly Interface: To design an intuitive and easy-to-navigate interface that allows customers to rent vehicles without hassle.
* Real-Time Availability: To provide real-time updates on the availability of vehicles and bus services, ensuring customers can make informed decisions.
* Secure Payment Processing: To integrate secure payment gateways for smooth transactions during bookings for bikes, cars, and buses.
* Vehicle and Service Management: To enable efficient management of available vehicles (bikes, cars), including tracking, updates, and maintenance.
* Customer Account Management: To allow users to manage their profiles, track booking history, and save preferences for future rentals and bookings.
* Optimized Travel Options: To offer customers multiple choices of transportation modes, enhancing flexibility in selecting the most suitable options based on cost, convenience, and travel requirements.

**Benefits:**

1. Centralized Platform For renting bikes and cars.2. User Satisfaction and feedback.3. Simple UI to provide comfort to user while using it.

**Chapter 2**

**LITERATURE SURVEY**

**Purpose of a RentaRide App:**

The purpose of the RentaRide app is to simplify vehicle rentals by providing an easy-to-use platform for users to book bikes and cars as per their needs. It aims to offer a hassle-free rental experience with transparent pricing, flexible booking options, and user feedback to ensure reliability and customer satisfaction.

**Key Features and Functionalities:**

The literature survey focuses on existing transportation systems related to bike rentals and car rentals services and highlights their features, technologies, and limitations:

* Bike and Car Rental SystemsExisting Systems like Uber, Rapido and Ola Bikes offer vehicle rentals with real-time tracking, booking, and payment integration through mobile apps.Key Technologies include GPS tracking, dynamic pricing, and secure payment gateways.
* Limitations: These platforms are often limited to specific areas (mostly urban), have high operational costs, and lack diverse transportation options like electric vehicles or short-distance rentals.

Vehicle rental systems have evolved from traditional manual processes to digital platforms:

* Traditional Rental Systems: Involve physical offices where customers complete paperwork for renting vehicles (e.g., Hertz, Avis).
* Online Rental Platforms: Digital platforms like Zoomcar and Turo enable users to book vehicles via apps or websites. These systems provide automation, real-time availability, and online payments, improving convenience and efficiency.

**Challenges in Vehicle Rental System development:**

* High Maintenance Costs: Vehicles require regular servicing and maintenance.
* User Verification & Security: Preventing fraudulent bookings and vehicle misuse.
* Regulatory Compliance: Meeting legal requirements for rentals in different regions.
* Competition & Market Saturation: Differentiating services from existing platforms.

**Case Studies:**

* Zoomcar: It revolutionized self-drive rentals in India, offering users short-term and long-term car rentals. It tackled challenges like regulatory issues and fleet maintenance by implementing GPS tracking, keyless entry, and flexible subscription models.
* Turo – Peer-to-Peer Car Sharing:Turo operates like Airbnb for cars, enabling individuals to rent out their vehicles. It addressed trust and insurance challenges by integrating identity verification, comprehensive damage coverage, and secure payment processing.

**Technologies Used:**

**JAVA:**

Java is a high-level, object-oriented programming language known for its platform independence (Write Once, Run Anywhere). It is widely used for developing web applications, enterprise software, mobile apps, and backend systems due to its robustness, scalability, and security.

We have used Java Swing for user interface. Java Swing is a GUI (Graphical User Interface) toolkit in Java, part of the Java Foundation Classes (JFC). It provides components like buttons, text fields, tables, and frames to create desktop applications. Swing offers a rich set of UI elements, supports event-driven programming, and allows customization using the MVC (Model-View-Controller) architecture.

**SQL DATABASE:**

SQL is a standard language for storing, manipulating and retrieving data in databases. Our SQL tutorial will teach you how to use SQL in: MySQL, SQL Server, MS Access, Oracle, Sybase, Informix, Postgres, and other database systems.

**Chapter 3**

**ANALYSIS & SPECIFICATION RQUIREMENT**

**3.1 Purpose:**

A vehicle rental application aims to provide a convenient, cost-effective, and flexible solution for users to rent two-wheelers and four-wheelers without the need for ownership. It simplifies the booking process, offers real-time vehicle availability, ensures transparent pricing, and enhances user experience with features like GPS tracking, online payments, and customer reviews. The application benefits both renters and service providers by streamlining operations, reducing paperwork, and improving accessibility to rental vehicles.

**3.2 Scope:**

RentaRide is designed to provide a hassle-free vehicle rental experience, offering users an easy way to rent two-wheelers and four-wheelers. The platform enables users to browse available vehicles, check real-time pricing, and make secure bookings with flexible rental durations. Features like GPS tracking, automated invoicing, and a user rating system enhance convenience and trust, ensuring a seamless rental process.

From a technical standpoint, RentaRide is built using Java and MySQL, ensuring robust backend management and secure data handling. The application efficiently processes user requests, manages vehicle availability, and maintains transaction records. Real-time data retrieval and optimized database management ensure smooth performance and scalability, making it capable of handling a growing user base and fleet size.

From a business perspective, RentaRide offers a profitable model by providing rental services to both individual users and corporate clients. It supports dynamic pricing, subscription-based rentals, and seasonal promotions to attract and retain customers. Additionally, collaborations with vehicle service providers and insurance companies can enhance service offerings. Future expansion into new markets and the integration of electric vehicles can further strengthen its position in the rental industry.

**3.3 Functional Requirements:**

The functional requirements of RentaRide include user registration and authentication to ensure secure access, along with a vehicle search and selection feature that allows users to browse available two-wheelers and four-wheelers. The booking and rental management system enables users to reserve vehicles, specify rental durations, and calculate charges. A secure payment processing system supports multiple transaction options, ensuring seamless payments. Users can provide feedback and ratings to enhance service quality, while an admin dashboard facilitates fleet and user management. The application also includes real-time vehicle availability tracking, dynamic pricing and billing, automated notifications for booking confirmations and reminders, and comprehensive report generation for monitoring user activity, revenue, and vehicle usage.

**Module Details:**

**Admin Module:** The admin module in RentaRide is designed to manage users, vehicles, and bookings efficiently. It allows administrators to oversee user accounts, verify registrations, and handle complaints or disputes. The system provides tools for vehicle management, including adding, updating, or removing vehicles, monitoring availability, and scheduling maintenance. Admins can also manage pricing, apply discounts, and configure rental policies.

**User Module:** The user module in RentaRide is designed to provide a seamless rental experience by allowing users to register, log in, and manage their profiles. Users can search for available two-wheelers and four-wheelers, view rental prices, and select vehicles based on their preferences. The booking system enables users to reserve vehicles, choose rental durations, and make secure payments. Additionally, users receive booking confirmations, notifications, and reminders. The module also includes features for providing feedback and ratings, viewing rental history, and managing cancellations or refunds. With a user-friendly interface, the module ensures a smooth and hassle-free rental process.

**Database Module:** The Database module, powered by SQL database , is the backbone of the application’s data storage and management system. The SQL database module in RentaRide is responsible for managing and storing all essential data related to users, vehicles, bookings, and transactions. It maintains user records, including login credentials, personal details, and rental history. Vehicle data such as availability, pricing, and maintenance schedules are stored and updated in real time. The system also handles booking records, ensuring accurate tracking of reservations, rental durations, and payment statuses. Additionally, it supports transaction management for secure payment processing and refund handling. The database module ensures data integrity, efficient retrieval, and smooth backend operations for seamless app performance.

**3.4 Non-Functional Requirements:**

RentaRide's non-functional requirements ensure the system's reliability, performance, and usability. The application must be secure, implementing encryption and authentication measures to protect user data and transactions. It should be scalable, capable of handling an increasing number of users, vehicles, and transactions without performance degradation. The system must offer high availability and reliability, ensuring minimal downtime and seamless booking experiences. Performance efficiency is crucial, with fast response times for searches, bookings, and payments. The interface should be user-friendly, providing an intuitive and smooth experience across devices. Additionally, the system should support data backup and recovery to prevent data loss and ensure business continuity.

### Chapter 4

### SOURCE CODE

**RentaRide1.java:**

package app;

public class RentaRide1 extends javax.swing.JFrame {

public RentaRide1() {

initComponents();

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jbtnRegister = new javax.swing.JButton();

jbtnLogin = new javax.swing.JButton();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jButton1 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE\_ON\_CLOSE);

setBackground(new java.awt.Color(255, 51, 51));

setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT\_CURSOR));

jLabel1.setBackground(new java.awt.Color(51, 51, 0));

jLabel1.setFont(new java.awt.Font("Tempus Sans ITC", 3, 36)); // NOI18N

jLabel1.setForeground(new java.awt.Color(102, 0, 153));

jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

jLabel1.setText("RENTARIDE");

jLabel1.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color(0, 0, 0)));

jbtnRegister.setBackground(new java.awt.Color(102, 0, 204));

jbtnRegister.setFont(new java.awt.Font("Sylfaen", 3, 18)); // NOI18N

jbtnRegister.setText("Register");

jbtnRegister.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jbtnRegister.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jbtnRegisterActionPerformed(evt);

}

});

jbtnLogin.setBackground(new java.awt.Color(102, 0, 204));

jbtnLogin.setFont(new java.awt.Font("Sylfaen", 3, 18)); // NOI18N

jbtnLogin.setText("Log In");

jbtnLogin.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jbtnLogin.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jbtnLoginActionPerformed(evt);

}

});

jLabel2.setText("If you are new, please register!");

jLabel3.setText("Already registered? Please LogIn");

jButton1.setBackground(new java.awt.Color(102, 0, 204));

jButton1.setFont(new java.awt.Font("Yet R", 1, 18)); // NOI18N

jButton1.setText("admin");

jButton1.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(112, 112, 112)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 428, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 114, Short.MAX\_VALUE)

.addComponent(jButton1))

.addGroup(layout.createSequentialGroup()

.addComponent(jbtnRegister, javax.swing.GroupLayout.PREFERRED\_SIZE, 136, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(177, 177, 177)

.addComponent(jbtnLogin, javax.swing.GroupLayout.PREFERRED\_SIZE, 112, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 0, Short.MAX\_VALUE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 168, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(89, 89, 89)

.addComponent(jLabel3)

.addGap(179, 179, 179)))

.addContainerGap())

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(17, 17, 17)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jButton1)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 97, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 125, Short.MAX\_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 43, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jLabel3))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jbtnRegister, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jbtnLogin, javax.swing.GroupLayout.DEFAULT\_SIZE, 42, Short.MAX\_VALUE))

.addGap(150, 150, 150))

);

pack();

}// </editor-fold>

private void jbtnRegisterActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

this.dispose();

new RegisterGUI().setVisible(true);

}

private void jbtnLoginActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

this.dispose();

new LogInGUI\_1().setVisible(true);

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// TODO add your handling code here:

this.dispose();

new Admin().setVisible(true);

}

public static void main(String args[]) {

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(RentaRide1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(RentaRide1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(RentaRide1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(RentaRide1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new RentaRide1().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JButton jbtnLogin;

private javax.swing.JButton jbtnRegister;

// End of variables declaration

}

**Usersession.java:**

package usersession;

public class UserSession {

private static int loggedInUserId = -1;

// Set the user ID after a successful login

public static void setLoggedInUserId(int userId) {

loggedInUserId = userId;

}

// Get the current logged-in user's ID

public static int getLoggedInUserId() {

return loggedInUserId;

}

// Check if there is a logged-in user

public static boolean isUserLoggedIn() {

return loggedInUserId != -1;

}

// Clear the session (log out)

public static void logout() {

loggedInUserId = -1;

}

}

**AdminPage.java:**

package app;

import usersession.UserSession;

public class AdminPage extends javax.swing.JFrame {

public AdminPage() {

initComponents();

}

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jButton3 = new javax.swing.JButton();

jButton4 = new javax.swing.JButton();

jButton5 = new javax.swing.JButton();

jButton6 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

jButton1.setText("Manage Bikes");

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jButton2.setText("Manage Cars");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

jButton3.setText("Payment Details");

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

jButton4.setText("Maintenance Vehicle ");

jButton4.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton4ActionPerformed(evt);

}

});

jButton5.setText("FeedBacks");

jButton5.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton5ActionPerformed(evt);

}

});

jButton6.setText("response");

jButton6.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton6ActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(227, 227, 227)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jButton1, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton2, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton4, javax.swing.GroupLayout.DEFAULT\_SIZE, 173, Short.MAX\_VALUE)

.addComponent(jButton5, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addContainerGap(316, Short.MAX\_VALUE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton6)

.addGap(38, 38, 38))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(96, 96, 96)

.addComponent(jButton1)

.addGap(30, 30, 30)

.addComponent(jButton2)

.addGap(33, 33, 33)

.addComponent(jButton3)

.addGap(36, 36, 36)

.addComponent(jButton4)

.addGap(32, 32, 32)

.addComponent(jButton5)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

.addComponent(jButton6)

.addContainerGap(7, Short.MAX\_VALUE))

);

pack();

}

.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill in all the fields.", "Input Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

String sql = "UPDATE bikes SET model = ?, brand = ?, rate\_per\_hour = ?, rate\_per\_day = ?, availability = ? WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(sql);

pstmt.setString(1, model);

pstmt.setString(2, brand);

pstmt.setBigDecimal(3, new java.math.BigDecimal(ratePerHour));

pstmt.setBigDecimal(4, new java.math.BigDecimal(ratePerDay));

pstmt.setBoolean(5, availability);

pstmt.setInt(6, id);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Bike updated successfully!");

clearInputFields();

loadBikes();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error updating bike.", "Database Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void deleteBike() {

int selectedRow = bikesTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a bike to delete.", "Selection Error", JOptionPane.ERROR\_MESSAGE);

return;

}

int id = (int) tableModel.getValueAt(selectedRow, 0);

int confirm = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this bike?", "Confirm Deletion", JOptionPane.YES\_NO\_OPTION);

if (confirm != JOptionPane.YES\_OPTION) {

return;

}

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

String sql = "DELETE FROM bikes WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(sql);

pstmt.setInt(1, id);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Bike deleted successfully!");

loadBikes();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error deleting bike.", "Database Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void sendBikeForMaintenance() {

int selectedRow = bikesTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a bike to send for maintenance.", "Selection Error", JOptionPane.ERROR\_MESSAGE);

return;

}

int id = (int) tableModel.getValueAt(selectedRow, 0);

String model = tableModel.getValueAt(selectedRow, 1).toString();

String brand = tableModel.getValueAt(selectedRow, 2).toString();

// Store the current date for maintenance

java.sql.Date maintenanceDate = new java.sql.Date(System.currentTimeMillis());

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

// Insert maintenance record into the maintenance table

String insertSql = "INSERT INTO maintenance\_bikes (model, brand, maintenance\_date) VALUES (?, ?, ?)";

PreparedStatement insertPstmt = conn.prepareStatement(insertSql);

insertPstmt.setString(1, model);

insertPstmt.setString(2, brand);

insertPstmt.setDate(3, maintenanceDate);

insertPstmt.executeUpdate();

// Update availability to false (bike is under maintenance)

String updateSql = "UPDATE bikes SET availability = false WHERE id = ?";

PreparedStatement updatePstmt = conn.prepareStatement(updateSql);

updatePstmt.setInt(1, id);

updatePstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Bike sent for maintenance successfully!");

loadBikes();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error sending bike for maintenance.", "Database Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void clearInputFields() {

modelField.setText("");

brandField.setText("");

ratePerHourField.setText("");

ratePerDayField.setText("");

availabilityCheckBox.setSelected(false);

}

public static void main(String[] args) {

SwingUtilities.invokeLater(BikeManager::new);

}

}

**BikePrice.java:**

package app;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class BikePrice extends JFrame {

private JFrame frame;

private JSpinner hourSpinner;

private JSpinner daySpinner;

private JLabel totalLabel;

private JRadioButton hoursRadioButton;

private JRadioButton daysRadioButton;

private ButtonGroup rentalTypeGroup;

private int vehicleId;

private String vehicleModel;

private String vehicleBrand;

private double ratePerHour;

private double ratePerDay;

public BikePrice(int vehicleId, String vehicleModel, String vehicleBrand, double ratePerHour, double ratePerDay) {

this.vehicleId = vehicleId;

this.vehicleModel = vehicleModel;

this.vehicleBrand = vehicleBrand;

this.ratePerHour = ratePerHour;

this.ratePerDay = ratePerDay;

frame = new JFrame("Bike Rental System");

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frame.setSize(600, 400);

frame.setLayout(new BorderLayout());

JPanel inputPanel = new JPanel();

inputPanel.setLayout(new GridBagLayout());

inputPanel.setBackground(new Color(245, 245, 245)); // Light gray background for the panel

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(10, 10, 10, 10);

hoursRadioButton = new JRadioButton("Rental Hours");

hoursRadioButton.setSelected(true); // Default option

hoursRadioButton.setFont(new Font("Arial", Font.PLAIN, 14));

daysRadioButton = new JRadioButton("Rental Days");

daysRadioButton.setFont(new Font("Arial", Font.PLAIN, 14));

rentalTypeGroup = new ButtonGroup();

rentalTypeGroup.add(hoursRadioButton);

rentalTypeGroup.add(daysRadioButton);

hourSpinner = new JSpinner(new SpinnerNumberModel(0, 0, 1000, 1)); // Default to 0

hourSpinner.setFont(new Font("Arial", Font.PLAIN, 14));

daySpinner = new JSpinner(new SpinnerNumberModel(0, 0, 1000, 1)); // Default to 0

daySpinner.setFont(new Font("Arial", Font.PLAIN, 14));

JButton calculateButton = new JButton("Calculate Total");

calculateButton.setBackground(new Color(34, 139, 34)); // Green button

calculateButton.setForeground(Color.WHITE);

calculateButton.setFont(new Font("Arial", Font.BOLD, 14));

calculateButton.setPreferredSize(new Dimension(150, 35));

calculateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

calculateTotal();

}

});

inputPanel.add(totalLabel, gbc);

JButton bookButton = new JButton("Pay Rent");

bookButton.setBackground(new Color(255, 69, 0)); // Orange button

bookButton.setForeground(Color.WHITE);

bookButton.setFont(new Font("Arial", Font.BOLD, 14));

bookButton.setPreferredSize(new Dimension(150, 35));

bookButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

openPaymentInterface();

}

});

frame.add(inputPanel, BorderLayout.CENTER);

frame.add(bookButton, BorderLayout.SOUTH);

hoursRadioButton.addActionListener(e -> {

hourSpinner.setEnabled(true);

daySpinner.setEnabled(false);

daySpinner.setValue(0);

});

daysRadioButton.addActionListener(e -> {

daySpinner.setEnabled(true);

hourSpinner.setEnabled(false);

hourSpinner.setValue(0);

});

// Initialize the spinners based on default selection

if (hoursRadioButton.isSelected()) {

hourSpinner.setEnabled(true);

daySpinner.setEnabled(false);

} else {

hourSpinner.setEnabled(false);

daySpinner.setEnabled(true);

}

frame.setVisible(true);

}

private void calculateTotal() {

double totalCost = 0;

double discount = 0;

if (hoursRadioButton.isSelected()) {

int hours = (int) hourSpinner.getValue();

int days = hours / 24;

hours = hours % 24;

double baseCost = (days \* ratePerDay) + (hours \* ratePerHour);

if (days > 7) discount = baseCost \* 0.2;

totalCost = baseCost - discount;

} else if (daysRadioButton.isSelected()) {

int days = (int) daySpinner.getValue();

double baseCost = days \* ratePerDay;

if (days > 7) discount = baseCost \* 0.2;

totalCost = baseCost - discount;

}

totalLabel.setText(String.format("Total Amount: $%.2f (Discount: $%.2f)", totalCost, discount));

}

private void openPaymentInterface() {

String totalAmount = totalLabel.getText().split(":")[1].trim();

String duration = hoursRadioButton.isSelected() ? hourSpinner.getValue() + " Hours" : daySpinner.getValue() + " Days";

new PaymentInterface(vehicleId, vehicleModel, vehicleBrand, totalAmount, duration).setVisible(true);

}

}

**BikesTable.java:**

package app;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import javax.swing.table.JTableHeader;

public class BikesTable extends JFrame {

private static final String DB\_URL = "jdbc:mysql://localhost:3306/vehiclerental";

private static final String DB\_USER = "root"; // Replace with your database username

private static final String DB\_PASSWORD = "#Sandy@25k"; // Replace with your database password

private JFrame frame;

private JTable table;

private DefaultTableModel tableModel;

public BikesTable() {

frame = new JFrame("Bike Rental System");

frame.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

frame.setSize(900, 600);

JLabel headerLabel = new JLabel("Available Bikes for Rent", SwingConstants.CENTER);

headerLabel.setFont(new Font("Arial", Font.BOLD, 26));

headerLabel.setForeground(new Color(34, 139, 34)); // Dark green color

frame.add(headerLabel, BorderLayout.NORTH);

tableModel = new DefaultTableModel(new String[]{

"ID", "Model", "Brand", "Rate Per Hour", "Rate Per Day", "Availability"

}, 0) {

@Override

public boolean isCellEditable(int row, int column) {

return false; // Make table cells non-editable

}

};

table = new JTable(tableModel);

table.setFont(new Font("Arial", Font.PLAIN, 14));

table.setRowHeight(30);

table.setSelectionBackground(new Color(173, 216, 230)); // Light blue color for selection

JTableHeader tableHeader = table.getTableHeader();

tableHeader.setFont(new Font("Arial", Font.BOLD, 16));

tableHeader.setBackground(new Color(70, 130, 180)); // Steel blue header

tableHeader.setForeground(Color.WHITE);

JScrollPane scrollPane = new JScrollPane(table);

JButton bookButton = new JButton("Book Selected Bike");

bookButton.setBackground(new Color(34, 139, 34)); // Green button

bookButton.setForeground(Color.WHITE);

bookButton.setFont(new Font("Arial", Font.BOLD, 16));

bookButton.setFocusPainted(false);

bookButton.setPreferredSize(new Dimension(200, 40));

bookButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

int selectedRow = table.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(frame, "Please select a bike to book!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

int bikeId = (int) tableModel.getValueAt(selectedRow, 0);

String bikeModel = (String) tableModel.getValueAt(selectedRow, 1);

String bikeBrand = (String) tableModel.getValueAt(selectedRow, 2);

double ratePerHour = (double) tableModel.getValueAt(selectedRow, 3);

double ratePerDay = (double) tableModel.getValueAt(selectedRow, 4);

String availability = (String) tableModel.getValueAt(selectedRow, 5);

if (availability.equals("Not Available")) {

JOptionPane.showMessageDialog(frame, "This vehicle is not available for booking!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

new BikePrice(bikeId, bikeModel, bikeBrand, ratePerHour, ratePerDay).setVisible(true);

}

});

// Panel for table and button

JPanel contentPanel = new JPanel(new BorderLayout());

contentPanel.setBackground(new Color(245, 245, 245)); // Light gray background

contentPanel.add(scrollPane, BorderLayout.CENTER);

// Footer panel for button

JPanel footerPanel = new JPanel();

footerPanel.setBackground(new Color(245, 245, 245));

footerPanel.add(bookButton);

frame.add(contentPanel, BorderLayout.CENTER);

frame.add(footerPanel, BorderLayout.SOUTH);

frame.setVisible(true);

loadData();

}

private void loadData() {

try (Connection conn = DriverManager.getConnection(DB\_URL, DB\_USER, DB\_PASSWORD);

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM bikes")) {

tableModel.setRowCount(0); // Clear existing rows

while (rs.next()) {

int id = rs.getInt("id");

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

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

double ratePerHour = rs.getDouble("rate\_per\_hour");

double ratePerDay = rs.getDouble("rate\_per\_day");

boolean availability = rs.getBoolean("availability");

tableModel.addRow(new Object[]{

id, model, brand, ratePerHour, ratePerDay, availability ? "Available" : "Not Available"

});

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(frame, "Error loading data from database.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

public static void main(String[] args) {

try {

Class.forName("com.mysql.cj.jdbc.Driver");

} catch (ClassNotFoundException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "MySQL JDBC Driver not found.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

SwingUtilities.invokeLater(BikesTable::new);

}

}

**CarMaintain.java:**

package app;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class CarMaintain extends JFrame {

private JTable maintenanceTable;

private DefaultTableModel tableModel;

private JButton completeMaintenanceButton;

public CarMaintain() {

setTitle("Car Maintenance Manager");

setSize(1000, 600);

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout(10, 10));

JLabel titleLabel = new JLabel("Car Maintenance Management System", JLabel.CENTER);

titleLabel.setFont(new Font("Arial", Font.BOLD, 24));

titleLabel.setOpaque(true);

titleLabel.setBackground(new Color(51, 153, 255));

titleLabel.setForeground(Color.WHITE);

titleLabel.setBorder(BorderFactory.createEmptyBorder(10, 0, 10, 0));

add(titleLabel, BorderLayout.NORTH);

tableModel = new DefaultTableModel(new String[]{"ID", "Model", "Brand", "Maintenance Date"}, 0);

maintenanceTable = new JTable(tableModel);

maintenanceTable.setRowHeight(25);

maintenanceTable.setFont(new Font("Arial", Font.PLAIN, 14));

maintenanceTable.setSelectionBackground(new Color(51, 153, 255));

maintenanceTable.setSelectionForeground(Color.WHITE);

JScrollPane tableScrollPane = new JScrollPane(maintenanceTable);

completeMaintenanceButton = new JButton("Complete Maintenance");

completeMaintenanceButton.setBackground(new Color(0, 204, 102));

completeMaintenanceButton.setForeground(Color.WHITE);

completeMaintenanceButton.setFont(new Font("Arial", Font.BOLD, 14));

completeMaintenanceButton.setPreferredSize(new Dimension(180, 30));

completeMaintenanceButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

completeMaintenance();

}

});

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 10));

buttonPanel.add(completeMaintenanceButton);

add(tableScrollPane, BorderLayout.CENTER);

add(buttonPanel, BorderLayout.SOUTH);

loadMaintenanceData();

setVisible(true);

}

private void loadMaintenanceData() {

tableModel.setRowCount(0);

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

String sql = "SELECT \* FROM maintenance\_cars"; // Fetching from maintenance\_cars table

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

tableModel.addRow(new Object[]{

rs.getInt("id"),

rs.getString("model"),

rs.getString("brand"),

rs.getDate("maintenance\_date")

});

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private void completeMaintenance() {

int[] selectedRows = maintenanceTable.getSelectedRows();

if (selectedRows.length == 0) {

JOptionPane.showMessageDialog(this, "Please select at least one row to complete maintenance.", "Selection Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

for (int selectedRow : selectedRows) {

int maintenanceId = (int) tableModel.getValueAt(selectedRow, 0);

String model = tableModel.getValueAt(selectedRow, 1) != null ? tableModel.getValueAt(selectedRow, 1).toString() : "";

String brand = tableModel.getValueAt(selectedRow, 2) != null ? tableModel.getValueAt(selectedRow, 2).toString() : "";

String vehicleType = "Car";

String checkCarSql = "SELECT COUNT(\*) FROM cars WHERE model = ? AND brand = ?";

PreparedStatement checkCarStmt = conn.prepareStatement(checkCarSql);

checkCarStmt.setString(1, model);

checkCarStmt.setString(2, brand);

ResultSet carResult = checkCarStmt.executeQuery();

carResult.next();

if (carResult.getInt(1) == 0) {

JOptionPane.showMessageDialog(this, "Car not found in the cars table.", "Data Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String updateMaintenanceSql = "UPDATE maintenance\_cars SET model = ?, brand = ?, maintenance\_date = ? WHERE id = ?";

PreparedStatement updateMaintenanceStmt = conn.prepareStatement(updateMaintenanceSql);

updateMaintenanceStmt.setString(1, model);

updateMaintenanceStmt.setString(2, brand);

updateMaintenanceStmt.setDate(3, new java.sql.Date(System.currentTimeMillis())); // Setting current date as maintenance date

updateMaintenanceStmt.setInt(4, maintenanceId);

updateMaintenanceStmt.executeUpdate();

String updateCarAvailabilitySql = "UPDATE cars SET availability = true WHERE model = ? AND brand = ?";

PreparedStatement updateCarStmt = conn.prepareStatement(updateCarAvailabilitySql);

updateCarStmt.setString(1, model);

updateCarStmt.setString(2, brand);

updateCarStmt.executeUpdate();

String deleteMaintenanceSql = "DELETE FROM maintenance\_cars WHERE id = ?";

PreparedStatement deleteMaintenanceStmt = conn.prepareStatement(deleteMaintenanceSql);

deleteMaintenanceStmt.setInt(1, maintenanceId);

deleteMaintenanceStmt.executeUpdate();

}

JOptionPane.showMessageDialog(this, "Maintenance completed successfully!");

loadMaintenanceData();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error completing maintenance.", "Database Error", JOptionPane.ERROR\_MESSAGE);

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(CarMaintain::new);

}

}

**CarManager.java:**

package app;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class CarManager extends JFrame {

private JTextField modelField, brandField, ratePerHourField, ratePerDayField;

private JCheckBox availabilityCheckBox;

private JButton addButton, editButton, deleteButton, maintenanceButton;

private JTable carsTable;

private DefaultTableModel tableModel;

public CarManager() {

setTitle("Manage Cars");

setSize(1000, 800);

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLayout(new BorderLayout(10, 10));

JLabel titleLabel = new JLabel("Car Management System", JLabel.CENTER);

titleLabel.setFont(new Font("Arial", Font.BOLD, 24));

titleLabel.setOpaque(true);

titleLabel.setBackground(new Color(51, 153, 255));

titleLabel.setForeground(Color.WHITE);

titleLabel.setBorder(BorderFactory.createEmptyBorder(10, 0, 10, 0));

add(titleLabel, BorderLayout.NORTH);

JPanel inputPanel = new JPanel(new GridLayout(6, 2, 10, 10));

inputPanel.setBorder(BorderFactory.createTitledBorder("Enter Car Details"));

inputPanel.setBackground(new Color(230, 230, 250));

inputPanel.add(new JLabel("Brand:", JLabel.RIGHT));

brandField = new JTextField();

brandField.setPreferredSize(new Dimension(200, 30));

inputPanel.add(brandField);

inputPanel.add(new JLabel("Rate Per Hour:", JLabel.RIGHT));

ratePerHourField = new JTextField();

ratePerHourField.setPreferredSize(new Dimension(200, 30));

inputPanel.add(ratePerHourField);

inputPanel.add(new JLabel("Rate Per Day:", JLabel.RIGHT));

ratePerDayField = new JTextField();

ratePerDayField.setPreferredSize(new Dimension(200, 30));

inputPanel.add(ratePerDayField);

inputPanel.add(new JLabel("Availability:", JLabel.RIGHT));

availabilityCheckBox = new JCheckBox("Available");

availabilityCheckBox.setBackground(new Color(230, 230, 250));

inputPanel.add(availabilityCheckBox);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 10));

addButton = new JButton("Add");

editButton = new JButton("Edit");

deleteButton = new JButton("Delete");

maintenanceButton = new JButton("Maintenance");

customizeButton(addButton, new Color(0, 153, 76));

customizeButton(editButton, new Color(255, 153, 51));

customizeButton(deleteButton, new Color(204, 0, 0));

customizeButton(maintenanceButton, new Color(102, 102, 255));

buttonPanel.add(addButton);

buttonPanel.add(editButton);

buttonPanel.add(deleteButton);

buttonPanel.add(maintenanceButton);

tableModel = new DefaultTableModel(new String[]{"ID", "Model", "Brand", "Rate/Hour", "Rate/Day", "Availability"}, 0);

carsTable = new JTable(tableModel);

carsTable.setRowHeight(25);

carsTable.setFont(new Font("Arial", Font.PLAIN, 14));

carsTable.setSelectionBackground(new Color(51, 153, 255));

carsTable.setSelectionForeground(Color.WHITE);

JScrollPane tableScrollPane = new JScrollPane(carsTable);

addButton.addActionListener(e -> addCar());

editButton.addActionListener(e -> editCar());

deleteButton.addActionListener(e -> deleteCar());

maintenanceButton.addActionListener(e -> sendCarForMaintenance());

carsTable.getSelectionModel().addListSelectionListener(e -> selectCarFromTable());

// Add components to the frame

add(inputPanel, BorderLayout.WEST);

add(buttonPanel, BorderLayout.CENTER);

add(tableScrollPane, BorderLayout.SOUTH);

loadCars();

setVisible(true);

}private void customizeButton(JButton button, Color bgColor) {

button.setBackground(bgColor);

button.setForeground(Color.WHITE);

button.setFocusPainted(false);

button.setFont(new Font("Arial", Font.BOLD, 14));

button.setPreferredSize(new Dimension(120, 30));

}

private void loadCars() {

tableModel.setRowCount(0);

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

String sql = "SELECT \* FROM cars";

Statement stmt = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql);

while (rs.next()) {

tableModel.addRow(new Object[] {

rs.getInt("id"),

rs.getString("model"),

rs.getString("brand"),

rs.getBigDecimal("rate\_per\_hour"),

rs.getBigDecimal("rate\_per\_day"),

rs.getBoolean("availability") ? "Yes" : "No"

});

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private void selectCarFromTable() {

int selectedRow = carsTable.getSelectedRow();

if (selectedRow != -1) {

modelField.setText(tableModel.getValueAt(selectedRow, 1).toString());

brandField.setText(tableModel.getValueAt(selectedRow, 2).toString());

ratePerHourField.setText(tableModel.getValueAt(selectedRow, 3).toString());

ratePerDayField.setText(tableModel.getValueAt(selectedRow, 4).toString());

availabilityCheckBox.setSelected(tableModel.getValueAt(selectedRow, 5).toString().equals("Yes"));

}

}

private void addCar() {

String model = modelField.getText();

String brand = brandField.getText();

String ratePerHour = ratePerHourField.getText();

String ratePerDay = ratePerDayField.getText();

boolean availability = availabilityCheckBox.isSelected();

if (model.isEmpty() || brand.isEmpty() || ratePerHour.isEmpty() || ratePerDay.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill in all the fields.", "Input Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

String sql = "INSERT INTO cars (model, brand, rate\_per\_hour, rate\_per\_day, availability) VALUES (?, ?, ?, ?, ?)";

PreparedStatement pstmt = conn.prepareStatement(sql);

pstmt.setString(1, model);

pstmt.setString(2, brand);

pstmt.setBigDecimal(3, new java.math.BigDecimal(ratePerHour));

pstmt.setBigDecimal(4, new java.math.BigDecimal(ratePerDay));

pstmt.setBoolean(5, availability);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Car added successfully!");

clearInputFields();

loadCars();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error adding car.", "Database Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void editCar() {

int selectedRow = carsTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a car to edit.", "Selection Error", JOptionPane.ERROR\_MESSAGE);

return;

}

int id = (int) tableModel.getValueAt(selectedRow, 0);

String model = modelField.getText();

String brand = brandField.getText();

String ratePerHour = ratePerHourField.getText();

String ratePerDay = ratePerDayField.getText();

boolean availability = availabilityCheckBox.isSelected();

if (model.isEmpty() || brand.isEmpty() || ratePerHour.isEmpty() || ratePerDay.isEmpty()) {

JOptionPane.showMessageDialog(this, "Please fill in all the fields.", "Input Error", JOptionPane.ERROR\_MESSAGE);

return;

}

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

String sql = "UPDATE cars SET model = ?, brand = ?, rate\_per\_hour = ?, rate\_per\_day = ?, availability = ? WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(sql);

pstmt.setString(1, model);

pstmt.setString(2, brand);

pstmt.setBigDecimal(3, new java.math.BigDecimal(ratePerHour));

pstmt.setBigDecimal(4, new java.math.BigDecimal(ratePerDay));

pstmt.setBoolean(5, availability);

pstmt.setInt(6, id);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Car updated successfully!");

clearInputFields();

loadCars();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error updating car.", "Database Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void deleteCar() {

int selectedRow = carsTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a car to delete.", "Selection Error", JOptionPane.ERROR\_MESSAGE);

return;

}

int id = (int) tableModel.getValueAt(selectedRow, 0);

int confirm = JOptionPane.showConfirmDialog(this, "Are you sure you want to delete this car?", "Confirm Deletion", JOptionPane.YES\_NO\_OPTION);

if (confirm != JOptionPane.YES\_OPTION) {

return;

}

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

String sql = "DELETE FROM cars WHERE id = ?";

PreparedStatement pstmt = conn.prepareStatement(sql);

pstmt.setInt(1, id);

pstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Car deleted successfully!");

loadCars();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error deleting car.", "Database Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void sendCarForMaintenance() {

int selectedRow = carsTable.getSelectedRow();

if (selectedRow == -1) {

JOptionPane.showMessageDialog(this, "Please select a car to send for maintenance.", "Selection Error", JOptionPane.ERROR\_MESSAGE);

return;

}

int id = (int) tableModel.getValueAt(selectedRow, 0);

String model = tableModel.getValueAt(selectedRow, 1).toString();

String brand = tableModel.getValueAt(selectedRow, 2).toString();

java.sql.Date maintenanceDate = new java.sql.Date(System.currentTimeMillis());

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

String insertSql = "INSERT INTO maintenance\_cars (model, brand, maintenance\_date) VALUES (?, ?, ?)";

PreparedStatement insertPstmt = conn.prepareStatement(insertSql);

insertPstmt.setString(1, model);

insertPstmt.setString(2, brand);

insertPstmt.setDate(3, maintenanceDate);

insertPstmt.executeUpdate();

String updateSql = "UPDATE cars SET availability = false WHERE id = ?";

PreparedStatement updatePstmt = conn.prepareStatement(updateSql);

updatePstmt.setInt(1, id);

updatePstmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Car sent for maintenance successfully!");

loadCars();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error sending car for maintenance.", "Database Error", JOptionPane.ERROR\_MESSAGE);

}

} private void clearInputFields() {

modelField.setText("");

brandField.setText("");

ratePerHourField.setText("");

ratePerDayField.setText("");

availabilityCheckBox.setSelected(false);

}public static void main(String[] args) {

SwingUtilities.invokeLater(CarManager::new);

}

}

**LogInGUI\_1.java:**

package app;

import usersession.UserSession;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.\*;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.swing.JOptionPane;

public class LogInGUI\_1 extends javax.swing.JFrame {

static final String JDBC\_URL = "jdbc:mysql://localhost:3306/VehicleRental";

static final String DB\_USERNAME = "root"; //

static final String DB\_PASSWORD = "#Sandy@25k";

private Object username;

public LogInGUI\_1() {

initComponents();

}

@SuppressWarnings("unchecked")

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

uname1 = new javax.swing.JTextField();

jButton1 = new javax.swing.JButton();

pass1 = new javax.swing.JPasswordField();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE\_ON\_CLOSE);

setTitle("User LogIn");

jLabel1.setText("Please enter details for log in:");

jLabel2.setText("Enter Username:");

jLabel3.setText("Enter password:");

uname1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

uname1ActionPerformed(evt);

}

});

jButton1.setBackground(new java.awt.Color(153, 153, 153));

jButton1.setFont(new java.awt.Font("STXinwei", 3, 18)); // NOI18N

jButton1.setForeground(new java.awt.Color(102, 0, 204));

jButton1.setText("Log In");

jButton1.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(155, 155, 155)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 179, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGroup(layout.createSequentialGroup()

.addGap(69, 69, 69)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT\_SIZE, 114, Short.MAX\_VALUE)

.addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)

.addComponent(uname1)

.addComponent(pass1, javax.swing.GroupLayout.DEFAULT\_SIZE, 165, Short.MAX\_VALUE)))

.addGroup(layout.createSequentialGroup()

.addGap(138, 138, 138)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 123, javax.swing.GroupLayout.PREFERRED\_SIZE)))

.addContainerGap(161, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(29, 29, 29)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 32, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED\_SIZE, 27, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(uname1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(39, 39, 39)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED\_SIZE, 28, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(pass1, javax.swing.GroupLayout.PREFERRED\_SIZE, javax.swing.GroupLayout.DEFAULT\_SIZE, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(44, 44, 44)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 41, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(88, Short.MAX\_VALUE))

);

pack();

}

private void uname1ActionPerformed(java.awt.event.ActionEvent evt) {

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

String un1 = uname1.getText();

String password1 = pass1.getText();

try {

Class.forName("com.mysql.cj.jdbc.Driver");

Connection conn1 = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k");

Statement stmt = conn1.createStatement();

String query = "SELECT \* FROM users WHERE username = ?";

PreparedStatement pstmt = conn1.prepareStatement(query);

pstmt.setString(1, un1); // Set the username parameter to match the entered username

ResultSet rs = pstmt.executeQuery();

boolean isValid = false; // Flag to check if login is successful

while (rs.next()) {

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

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

if (un1.equals(username) && password1.equals(password)) {

isValid = true;

int userId = rs.getInt("id"); // Fetch the userId when login is successful

UserSession.setLoggedInUserId(userId); // Set the session

break; // Exit the loop as soon as a match is found

}

}

if (isValid) {

JOptionPane.showMessageDialog(LogInGUI\_1.this, "Log In Successful...!");

new ChoiceFrame().setVisible(true); // Open the Vehicle Interface

dispose(); // Close the login window

} else {

JOptionPane.showMessageDialog(null, "Incorrect username or password");

}

conn1.close();

} catch (Exception er) {

JOptionPane.showMessageDialog(null, "Error: " + er.getMessage());

}

}

public static void main(String args[]) {

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(LogInGUI\_1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(LogInGUI\_1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(LogInGUI\_1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(LogInGUI\_1.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new LogInGUI\_1().setVisible(true);

}

});

}

private javax.swing.JButton jButton1;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JPasswordField pass1;

private javax.swing.JTextField uname1;

}

**ChoiceFrame.java:**

package app;

import javax.swing.JOptionPane;

import usersession.UserSession;

public class ChoiceFrame extends javax.swing.JFrame {

public ChoiceFrame() {

initComponents();

}

@SuppressWarnings("unchecked")

private void initComponents() {

jLabel1 = new javax.swing.JLabel();

jButton1 = new javax.swing.JButton();

jButton2 = new javax.swing.JButton();

jButton3 = new javax.swing.JButton();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE\_ON\_CLOSE);

setBackground(new java.awt.Color(153, 153, 0));

jLabel1.setFont(new java.awt.Font("Sylfaen", 1, 24)); // NOI18N

jLabel1.setText("Welcome to RentaRide! ");

jButton1.setBackground(new java.awt.Color(153, 0, 204));

jButton1.setFont(new java.awt.Font("Sylfaen", 3, 18)); // NOI18N

jButton1.setText("Book ");

jButton1.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jButton2.setBackground(new java.awt.Color(153, 0, 204));

jButton2.setFont(new java.awt.Font("Sylfaen", 3, 18)); // NOI18N

jButton2.setText("Return");

jButton2.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

jButton3.setBackground(new java.awt.Color(255, 204, 51));

jButton3.setText("Help");

jButton3.setBorder(new javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));

jButton3.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND\_CURSOR));

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addGap(97, 97, 97)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 93, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 214, Short.MAX\_VALUE)

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 110, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(139, 139, 139))

.addGroup(layout.createSequentialGroup()

.addGap(168, 168, 168)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 272, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE))

.addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()

.addContainerGap(javax.swing.GroupLayout.DEFAULT\_SIZE, Short.MAX\_VALUE)

.addComponent(jButton3)

.addGap(124, 124, 124))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addGap(61, 61, 61)

.addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 63, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 118, Short.MAX\_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

.addComponent(jButton1, javax.swing.GroupLayout.PREFERRED\_SIZE, 35, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addComponent(jButton2, javax.swing.GroupLayout.PREFERRED\_SIZE, 31, javax.swing.GroupLayout.PREFERRED\_SIZE))

.addGap(50, 50, 50)

.addComponent(jButton3)

.addContainerGap())

);

pack();

} private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

new VehicleInterface().setVisible(true); // TODO add your handling code here:

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

int userId = UserSession.getLoggedInUserId();

if (userId != -1) {

new ReturnInterface().setVisible(true);

dispose(); // Close the current choice interface

} else {

JOptionPane.showMessageDialog(this, "No user is logged in!", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

int userId = UserSession.getLoggedInUserId();

new CustomerReq(userId).setVisible(true);

}

public static void main(String args[]) {

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(ChoiceFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(ChoiceFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(ChoiceFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(ChoiceFrame.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new ChoiceFrame().setVisible(true);

}

}); }

**FeedbackForm.java:**

package app;

import usersession.UserSession;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

public class FeedbackForm extends JFrame {

private JFrame frame;

private JComboBox<Integer> ratingComboBox;

private JTextArea feedbackTextArea;

private JButton submitButton;

private JLabel feedbackLabel;

private JLabel ratingLabel;

private static final String DB\_URL = "jdbc:mysql://localhost:3306/vehiclerental"; // Change to your database URL

private static final String DB\_USER = "root"; // Database username

private static final String DB\_PASSWORD = "#Sandy@25k"; // Database password

public FeedbackForm() {

int loggedInUserId = UserSession.getLoggedInUserId();

if (loggedInUserId == -1) {

JOptionPane.showMessageDialog(this, "No user is logged in.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

frame = new JFrame("User Feedback");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 300);

frame.setLayout(new BorderLayout());

JPanel inputPanel = new JPanel();

inputPanel.setLayout(new GridBagLayout());

inputPanel.setBackground(new Color(245, 245, 245)); // Light gray background for the panel

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(10, 10, 10, 10);

// Rating label and combo box (1-5 stars)

ratingLabel = new JLabel("Select Rating (1 to 5):");

ratingLabel.setFont(new Font("Arial", Font.PLAIN, 14));

ratingComboBox = new JComboBox<>();

for (int i = 1; i <= 5; i++) {

ratingComboBox.addItem(i); // Adding rating options 1 to 5

}

ratingComboBox.setFont(new Font("Arial", Font.PLAIN, 14));

// Feedback label and text area

feedbackLabel = new JLabel("Enter your Feedback:");

feedbackLabel.setFont(new Font("Arial", Font.PLAIN, 14));

feedbackTextArea = new JTextArea(5, 20);

feedbackTextArea.setFont(new Font("Arial", Font.PLAIN, 14));

feedbackTextArea.setLineWrap(true);

feedbackTextArea.setWrapStyleWord(true);

JScrollPane scrollPane = new JScrollPane(feedbackTextArea);

submitButton = new JButton("Submit Feedback");

submitButton.setBackground(new Color(34, 139, 34)); // Green button

submitButton.setForeground(Color.WHITE);

submitButton.setFont(new Font("Arial", Font.BOLD, 14));

submitButton.setPreferredSize(new Dimension(150, 35));

submitButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

submitFeedback(loggedInUserId);

dispose();// Pass the user ID to submit feedback

}

});

**ReturnBikeInterface.java:**

import usersession.UserSession;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class ReturnBikeInterface extends JFrame {

private JComboBox<String> bookedBikesComboBox;

private JButton returnButton;

public ReturnBikeInterface() {

int userId = UserSession.getLoggedInUserId(); // Fetch the logged-in user ID

if (userId == -1) {

JOptionPane.showMessageDialog(this, "No user is logged in.", "Error", JOptionPane.ERROR\_MESSAGE);

dispose(); // Close the frame

return;

}

setTitle("Return Booked Bike");

setLayout(new BorderLayout());

setSize(500, 300);

setLocationRelativeTo(null);

setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

JPanel panel = new JPanel();

panel.setLayout(new GridBagLayout());

panel.setBackground(new Color(245, 245, 245)); // Light gray background

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(10, 10, 10, 10);

gbc.gridx = 0;

gbc.gridy = 0;

panel.add(new JLabel("Select Booked Bike:"), gbc);

// ComboBox to display booked bikes

gbc.gridx = 1;

bookedBikesComboBox = new JComboBox<>();

panel.add(bookedBikesComboBox, gbc);

gbc.gridx = 0;

gbc.gridy = 1;

gbc.gridwidth = 2; // Span across both columns

returnButton = new JButton("Return Bike");

returnButton.setBackground(new Color(34, 139, 34)); // Green button

returnButton.setForeground(Color.WHITE);

returnButton.setFont(new Font("Arial", Font.BOLD, 16));

returnButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

returnBike(userId);

}

});

panel.add(returnButton, gbc);

add(panel, BorderLayout.CENTER);

boolean hasBookedBikes = loadBookedBikes(userId);

if (hasBookedBikes) {

// Add a stylish header if booked bikes exist

JPanel headerPanel = new JPanel();

headerPanel.setBackground(new Color(255, 69, 0)); // Orange header

JLabel headerLabel = new JLabel("Return Booked Bike");

headerLabel.setForeground(Color.WHITE);

headerLabel.setFont(new Font("Arial", Font.BOLD, 20));

headerPanel.add(headerLabel);

add(headerPanel, BorderLayout.NORTH); // Add header to the frame

}

setVisible(true);

}

private boolean loadBookedBikes(int userId) {

List<String> bookedBikes = new ArrayList<>();

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

// Query to fetch all booked bikes by the user

PreparedStatement stmt = conn.prepareStatement(

"SELECT bb.bike\_id, b.model, b.brand, bb.booking\_date " +

"FROM booked\_bikes bb " +

"JOIN bikes b ON bb.bike\_id = b.id " +

"WHERE bb.user\_id = ? AND bb.status = 'Booked'"

);

stmt.setInt(1, userId);

ResultSet rs = stmt.executeQuery();

while (rs.next()) {

int bikeId = rs.getInt("bike\_id");

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

String bikeBrand = rs.getString("brand");

String bookingDate = rs.getDate("booking\_date").toString();

bookedBikes.add("Bike ID: " + bikeId + ", " + bikeBrand + " " + bikeModel + " (Booked on: " + bookingDate + ")");

}

if (!bookedBikes.isEmpty()) {

for (String bikeDetails : bookedBikes) {

bookedBikesComboBox.addItem(bikeDetails);

}

return true; // Booked bikes found

} else {

JOptionPane.showMessageDialog(this, "No booked bikes found.", "Error", JOptionPane.ERROR\_MESSAGE);

dispose(); // Close the frame if no bikes are found

return false; // No booked bikes found

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error fetching booked bikes!", "Error", JOptionPane.ERROR\_MESSAGE);

dispose(); // Close the frame if an error occurs

return false;

}

}

private void returnBike(int userId) {

String selectedBike = (String) bookedBikesComboBox.getSelectedItem();

if (selectedBike == null) {

JOptionPane.showMessageDialog(this, "Please select a bike to return.", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String[] bikeDetails = selectedBike.split(",");

String bikeIdStr = bikeDetails[0].split(":")[1].trim();

int bikeId = Integer.parseInt(bikeIdStr);

try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/vehiclerental", "root", "#Sandy@25k")) {

PreparedStatement deleteBookingStmt = conn.prepareStatement(

"DELETE FROM booked\_bikes WHERE bike\_id = ? AND user\_id = ?"

);

deleteBookingStmt.setInt(1, bikeId);

deleteBookingStmt.setInt(2, userId);

int rowsDeleted = deleteBookingStmt.executeUpdate();

if (rowsDeleted > 0) {

PreparedStatement updateAvailabilityStmt = conn.prepareStatement(

"UPDATE bikes SET availability = TRUE WHERE id = ?"

);

updateAvailabilityStmt.setInt(1, bikeId);

updateAvailabilityStmt.executeUpdate();

JOptionPane.showMessageDialog(this, "Bike returned successfully!\nThank you for using Rentaride");

new FeedbackForm().setVisible(true);

} else {

JOptionPane.showMessageDialog(this, "No booking found for the selected bike.", "Error", JOptionPane.ERROR\_MESSAGE);

}

// Close the interface

dispose();

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error returning the bike!", "Error", JOptionPane.ERROR\_MESSAGE);

dispose(); // Close the frame if an error occurs

}

}

}

**VehicleRental.sql:**

CREATE DATABASE VehicleRental;

USE VehicleRental;

CREATE TABLE users (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255) NOT NULL,

phone VARCHAR(15),

license\_number VARCHAR(255),

password VARCHAR(255) NOT NULL

);

CREATE TABLE bikes (

id INT PRIMARY KEY AUTO\_INCREMENT,

model VARCHAR(50),

brand VARCHAR(50),

rate\_per\_hour DECIMAL(10, 2),

rate\_per\_day DECIMAL(10, 2),

availability BOOLEAN

);

INSERT INTO bikes (model, brand, rate\_per\_hour, rate\_per\_day, availability)

VALUES

('CBR 500', 'Honda', 15.00, 75.00, TRUE),

('Ninja 400', 'Kawasaki', 20.00, 90.00, TRUE),

('Ducati Monster', 'Ducati', 25.00, 120.00, TRUE);

CREATE TABLE cars (

id INT PRIMARY KEY AUTO\_INCREMENT,

model VARCHAR(50),

brand VARCHAR(50),

rate\_per\_hour DECIMAL(10, 2),

rate\_per\_day DECIMAL(10, 2),

availability BOOLEAN

);

INSERT INTO cars (model, brand, rate\_per\_hour, rate\_per\_day, availability)

VALUES

('Model S', 'Tesla', 30.00, 150.00, TRUE),

('Civic', 'Honda', 20.00, 100.00, TRUE),

('Mustang', 'Ford', 50.00, 200.00, TRUE);

CREATE TABLE payment\_details (

id INT AUTO\_INCREMENT PRIMARY KEY,

username VARCHAR(255),

phone\_number VARCHAR(15),

model VARCHAR(255),

brand VARCHAR(255),

duration VARCHAR(255), -- This could be "X hours" or "X days"

total\_cost DOUBLE,

payment\_method VARCHAR(255)

);

CREATE TABLE booked\_bikes (

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

bike\_id INT NOT NULL,

booking\_date DATE, -- No default value, will be set by trigger

start\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

end\_time TIMESTAMP,

total\_price DECIMAL(10, 2) NOT NULL,

status VARCHAR(20) DEFAULT 'Booked',

CONSTRAINT fk\_user FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk\_bike FOREIGN KEY (bike\_id) REFERENCES bikes(id) ON DELETE CASCADE ON UPDATE CASCADE

);

DELIMITER //

CREATE TRIGGER before\_booked\_bikes\_insert

BEFORE INSERT ON booked\_bikes

FOR EACH ROW

BEGIN

SET NEW.booking\_date = CURRENT\_DATE;

END //

DELIMITER ;

CREATE TABLE booked\_cars (feedbacks

id INT AUTO\_INCREMENT PRIMARY KEY,

user\_id INT NOT NULL,

car\_id INT NOT NULL,

booking\_date DATE, -- Booking date (can be set via trigger or defaults)

start\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

end\_time TIMESTAMP,

total\_price DECIMAL(10, 2) NOT NULL,

status VARCHAR(20) DEFAULT 'Booked',

CONSTRAINT fk\_booked\_cars\_user FOREIGN KEY (user\_id) REFERENCES users(id) ON DELETE CASCADE ON UPDATE CASCADE,

CONSTRAINT fk\_booked\_cars\_car FOREIGN KEY (car\_id) REFERENCES cars(id) ON DELETE CASCADE ON UPDATE CASCADE

);

CREATE TRIGGER before\_booked\_cars\_insert

BEFORE INSERT ON booked\_cars

FOR EACH ROW

BEGIN

SET NEW.booking\_date = CURRENT\_DATE;

END //

DELIMITER ;

CREATE TABLE maintenance\_bikes (

id INT AUTO\_INCREMENT PRIMARY KEY,

model VARCHAR(255) NOT NULL,

brand VARCHAR(255) NOT NULL,

maintenance\_date DATE NOT NULL

);

CREATE TABLE maintenance\_cars (

id INT AUTO\_INCREMENT PRIMARY KEY,

model VARCHAR(255) NOT NULL,

brand VARCHAR(255) NOT NULL,

maintenance\_date DATE NOT NULL

);

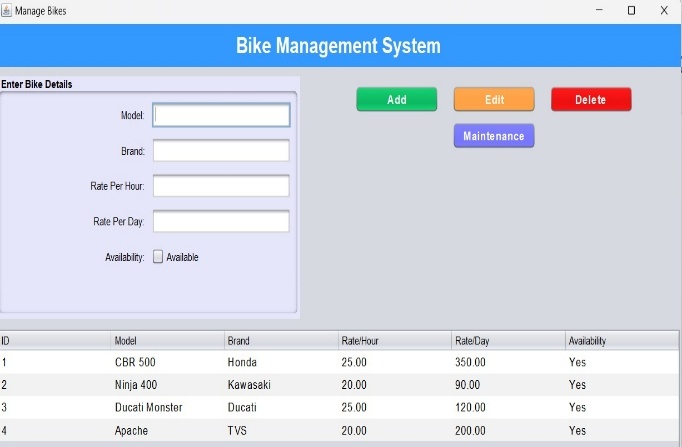
### Chapter 5

### SNAPSHOTS



Figure 1. Initial Interface

Figure 2 manage bikes



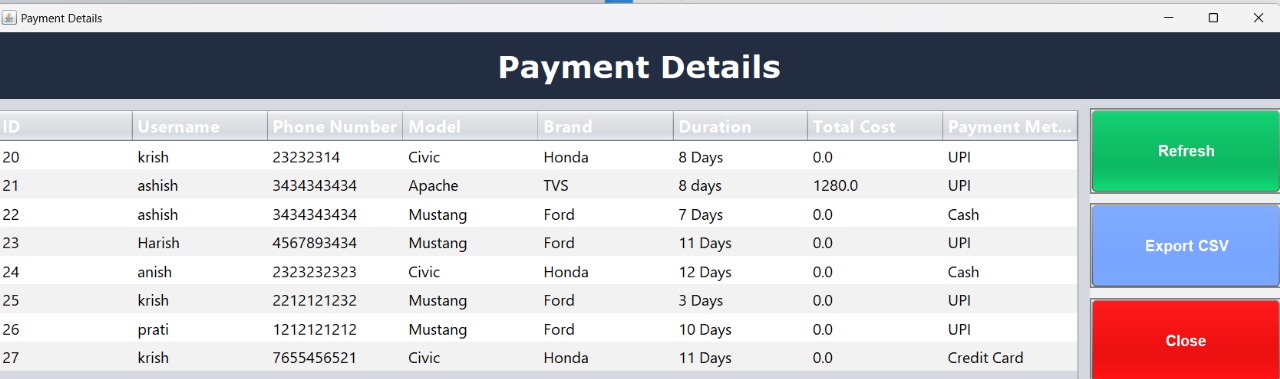
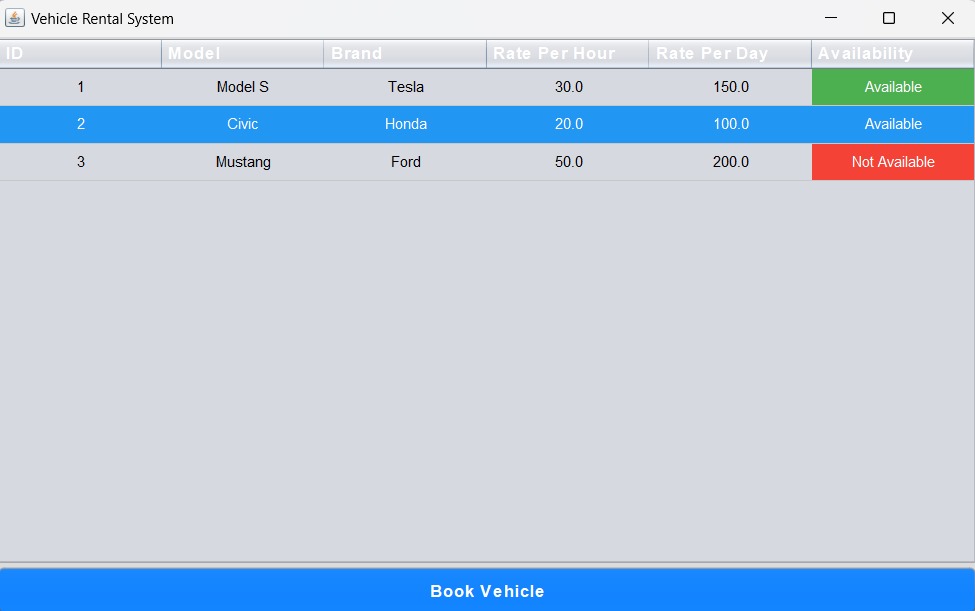
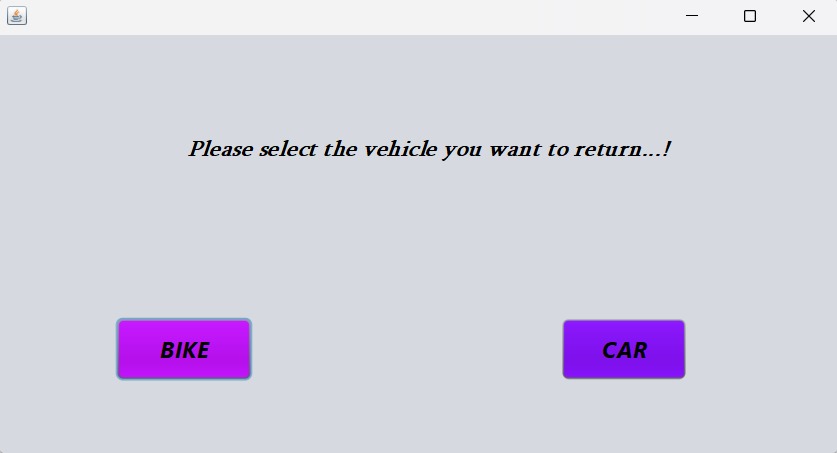
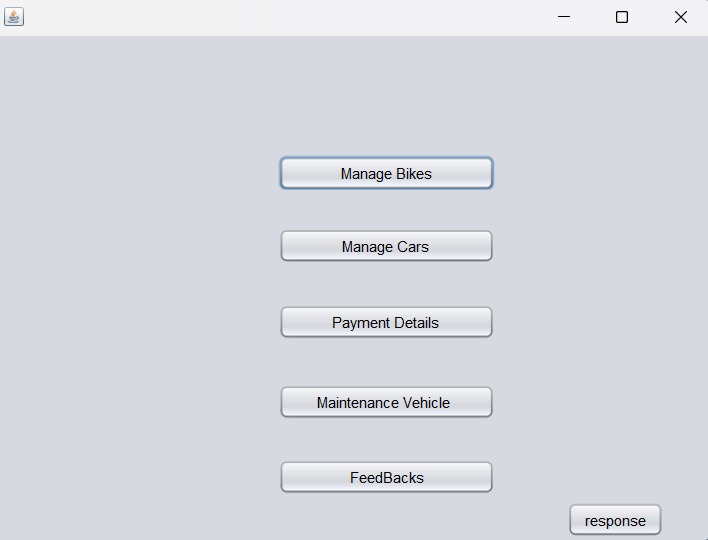
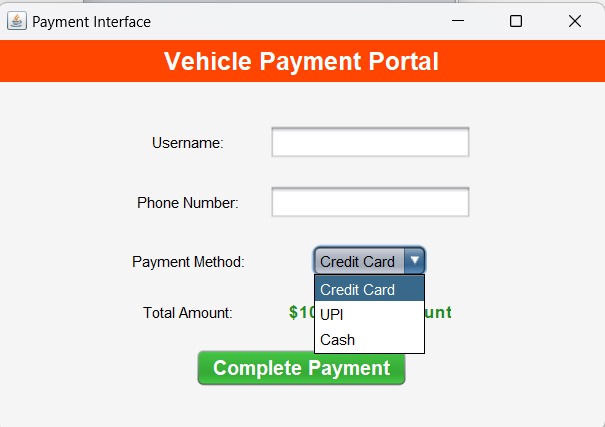


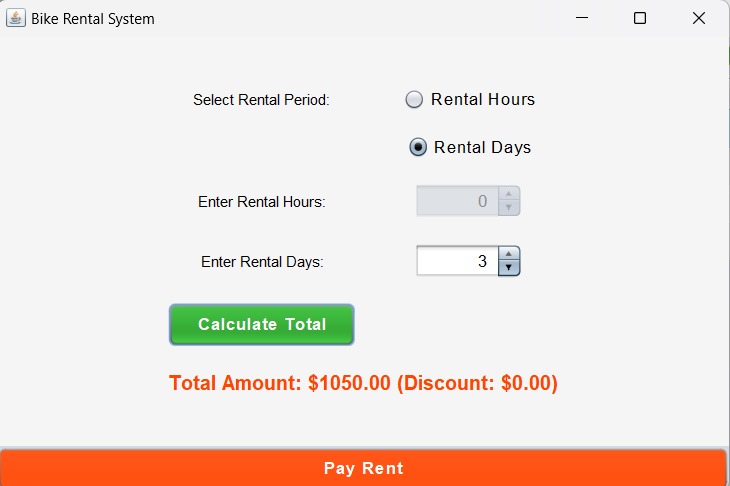
Figure 3 Payment details

Figure 5 Selection



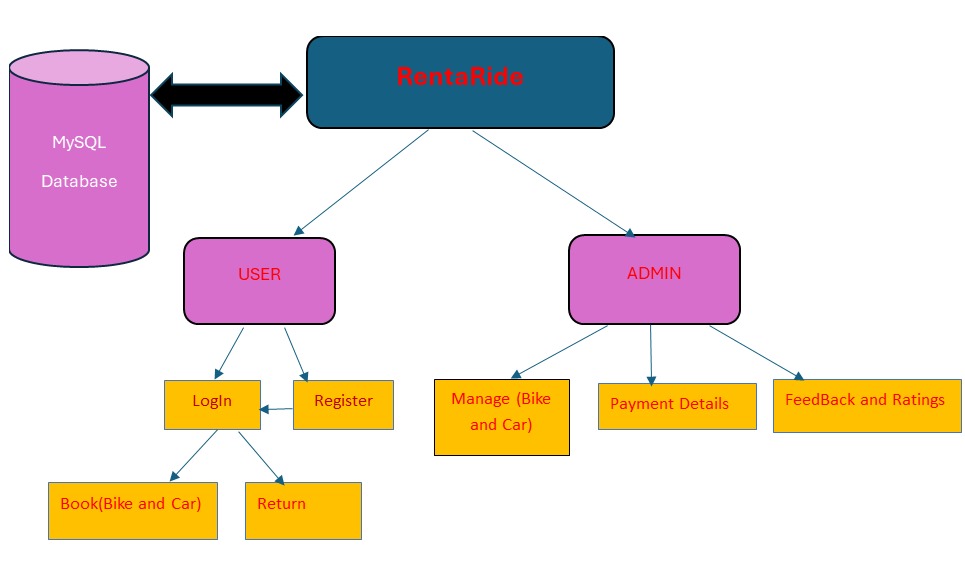






*Fig: Payments and pricing*

*Fig: Admin Interface*



*Fig: Architecture*

### Chapter 6

# CONCLUSION

**Conclusion:**

The RentaRide is a comprehensive application designed to streamline the process of renting vehicles (cars and bikes) for short-term or long-term use. This system leverages Java for backend development and an SQL database for efficient data management, aiming to provide a seamless experience for both customers and administrators. The application caters to users looking for a reliable, easy-to-use platform to rent vehicles for their personal or business needs. It offers a wide range of features such as vehicle availability checks, booking management and user-friendly interfaces. The system ensures reliability through a well-structured database, allowing real-time updates, conflict-free bookings, and secure payments.

### Chapter 7

# FUTURE ENHANCEMENT

To further enhance the RentaRide app and improve both user and owner experience, several advanced features can be integrated. Implementing a real-time demand-based pricing model can optimize rental rates dynamically, ensuring competitive pricing for users while maximizing earnings for vehicle owners. A ride insurance option can provide users with accident coverage, increasing trust and safety. Introducing multi-city rentals will allow users to rent vehicles across different locations, expanding the service reach. Additionally, vehicle maintenance alerts can be provided to owners, ensuring timely servicing and better vehicle performance. A user rating and behavior analysis system can help owners assess reliable renters and encourage responsible usage. Lastly, a gamification reward system—offering discounts, points, or cashback for frequent bookings—can boost customer engagement and retention. These features will enhance convenience, security, and profitability, making RentaRide a more efficient and user-friendly platform.

### Chapter 8

# REFERENCES

**Reference and Acknowledgement**

The RentaRide application was developed with the assistance of ChatGPT, leveraging AI-driven guidance for coding, database design, and feature implementation. ChatGPT provided solutions for Java and MySQL integration, as well as enhancements such as user authentication, booking systems, and AI-powered customer support.This project showcases the potential of AI-assisted development, where ChatGPT played a crucial role in shaping the application's core functionalities, troubleshooting errors, and optimizing performance.