JAVA WITH OOPS – MINI PROJECT

VEHICLE RENTAL SYSTEM

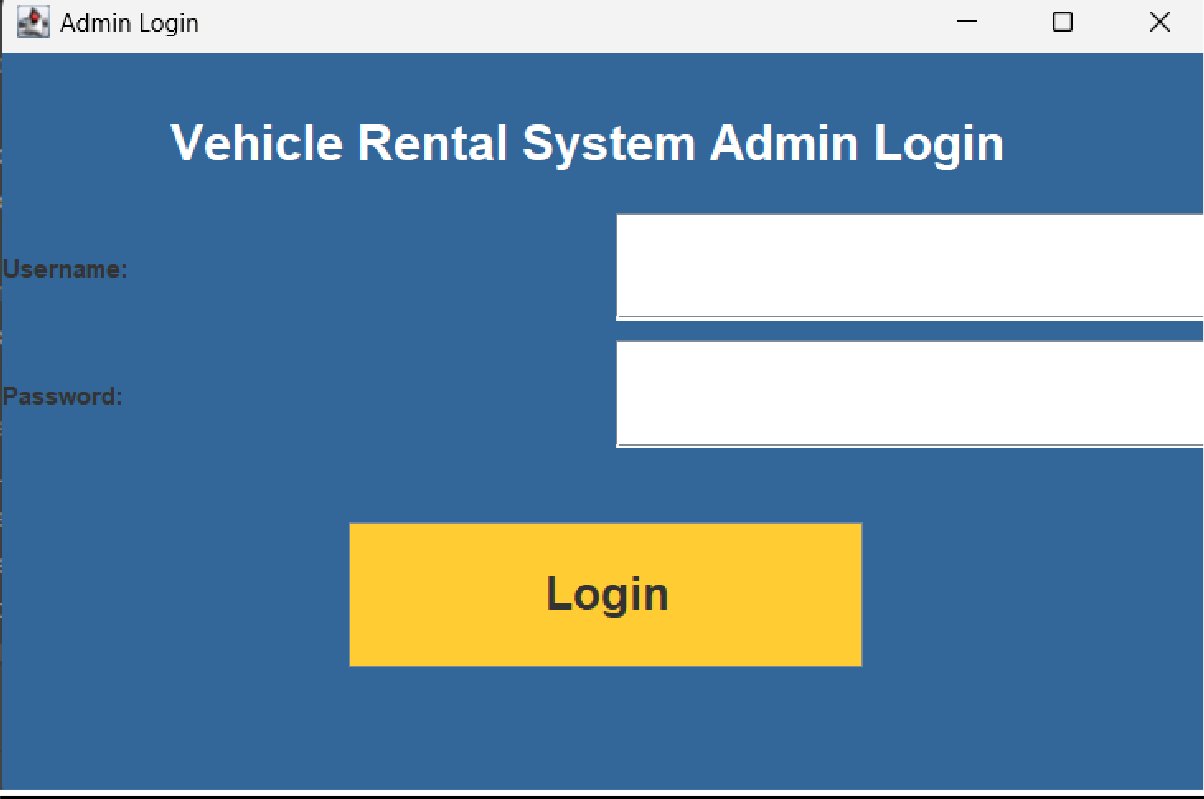
INTRODUCTION/OVERVIEW:

The **Vehicle Rental System** is a comprehensive Java-based project developed to simplify and automate vehicle rental operations. Designed with a robust Swing- based graphical user interface, it provides secure admin login and an intuitive dashboard for managing key functionalities such as vehicle availability,

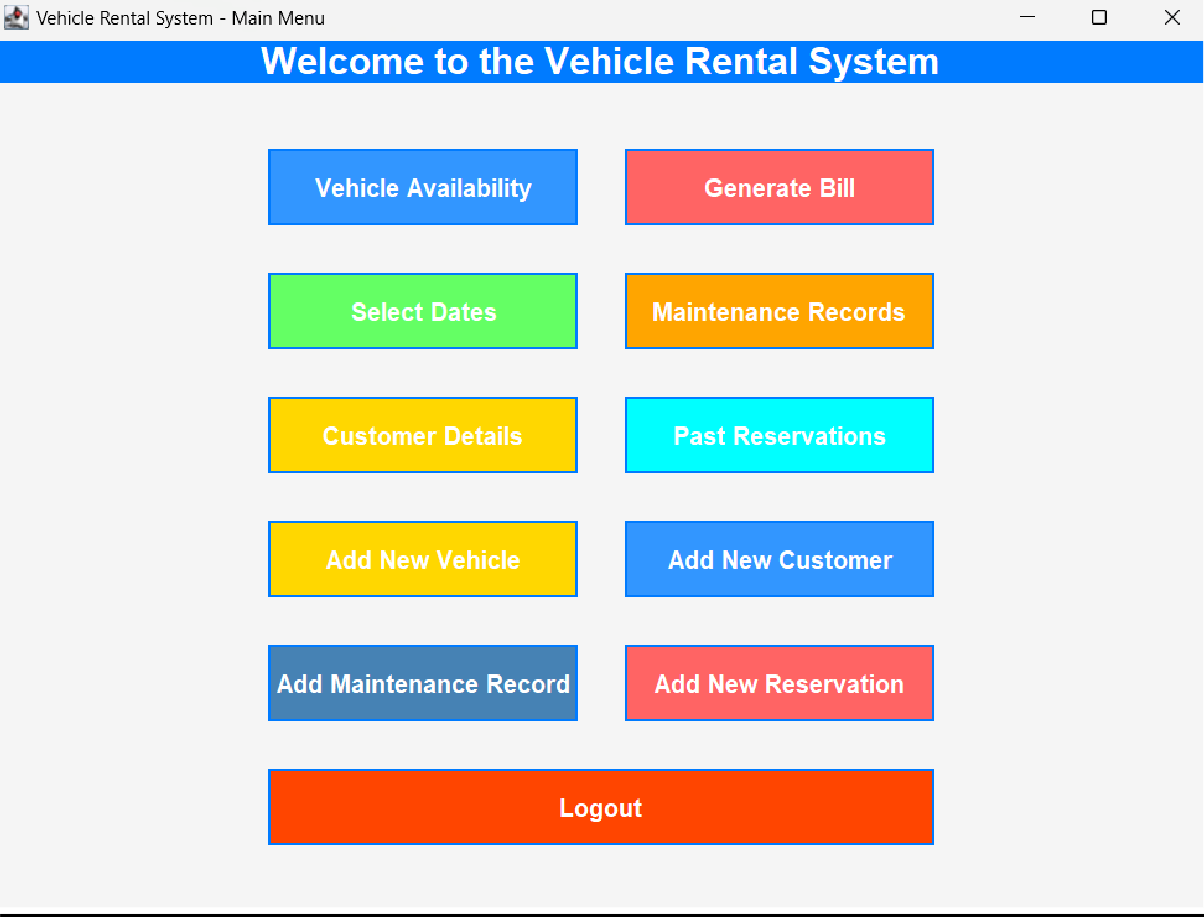
reservations, maintenance records, customer details, and billing. By integrating MySQL for backend data management, the system ensures reliable storage and retrieval of information. Its user-friendly design and well-organized menu options enable smooth navigation, making it an efficient tool for rental businesses to enhance their operational workflow and customer experience.

# SCREENSHOTS OF PROJECT:

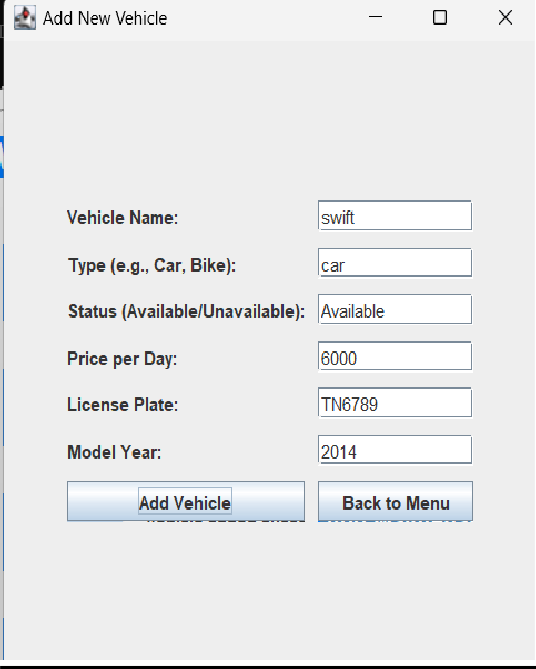
**Login page:**



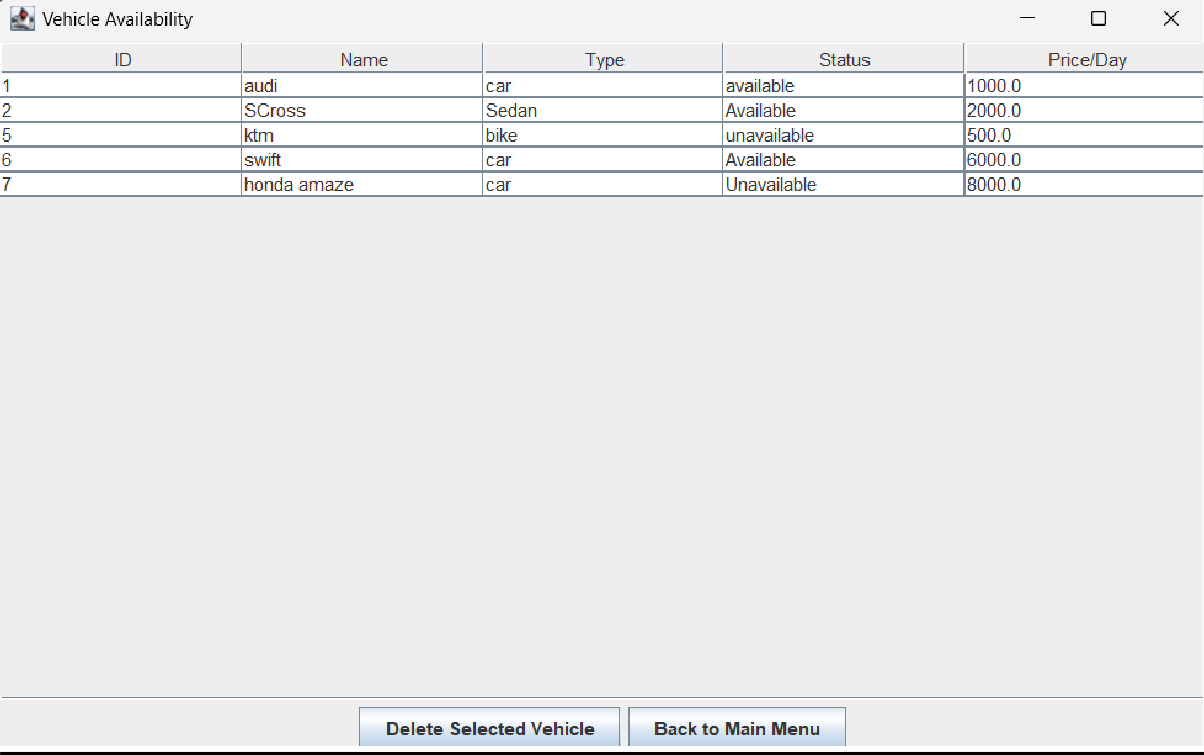
**Main menu:**



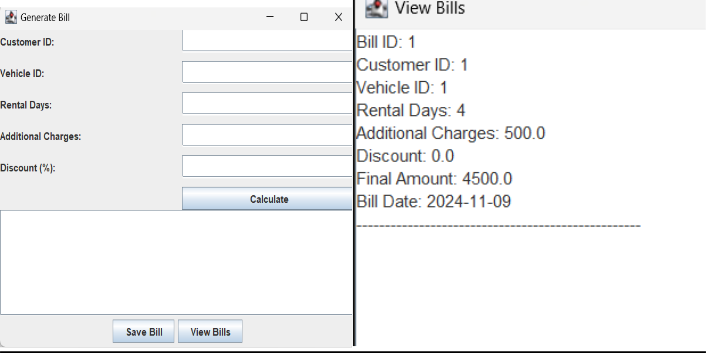
**Adding vehicle:**



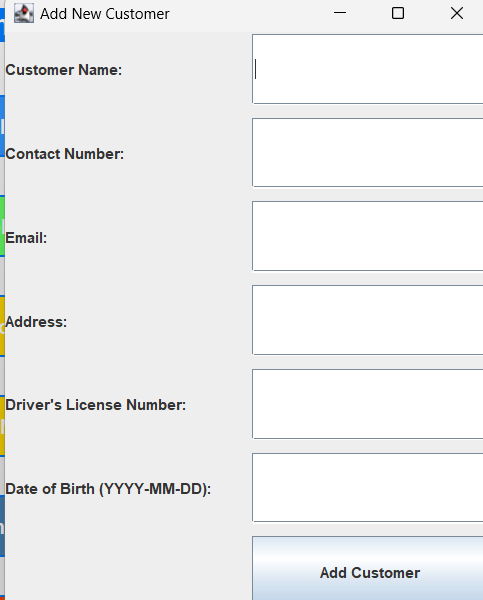
# VEHICLE AVAILABLITY:



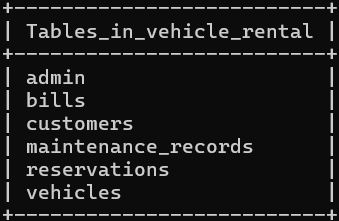
**BILL GENERATING:**



**NEW CUSTOMER:**



**MYSQL TABLES USED:**



# PROJECT CODE:

import javax.swing.\*; import java.awt.\*;

import java.awt.event.\*; import java.sql.\*;

import java.util.Date;

import javax.swing.table.DefaultTableModel; import com.toedter.calendar.JDateChooser;

import java.awt.event.ActionListener; import java.awt.event.MouseAdapter; import java.awt.event.MouseEvent;

import java.text.SimpleDateFormat; public class VehicleRentalSystem {

private JFrame mainFrame;

private Connection connection; public VehicleRentalSystem() {

// Establish the database connection try {

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

connection = DriverManager.*getConnection*("jdbc:mysql://localhost:3306/vehicle\_rental", "root", "janu0309");

System.*out*.println("Database connected successfully.");

} catch (Exception e) { e.printStackTrace();

}

// Initialize the GUI createAdminLogin();

}

private void createAdminLogin() {

// Create the main frame and set its size

JFrame frame = new JFrame("Admin Login");

frame.setSize(600, 400); // Increased size for better UI

frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*); frame.setLayout(new BorderLayout());

// Create a panel to hold the form elements JPanel panel = new JPanel();

panel.setLayout(new BoxLayout(panel, BoxLayout.*Y\_AXIS*)); // Use BoxLayout for vertical alignment

panel.setBackground(new Color(51, 102, 153)); // Set the background color

// Add spacing at the top

panel.add(Box.*createVerticalStrut*(30)); // Space between the top and title

// Title label with a larger font size and centered

JLabel titleLabel = new JLabel("Vehicle Rental System Admin Login", JLabel.*CENTER*); titleLabel.setForeground(Color.*WHITE*);

titleLabel.setFont(new Font("Arial", Font.*BOLD*, 24)); // Larger font for the title panel.add(titleLabel);

// Add space between the title and input fields panel.add(Box.*createVerticalStrut*(20));

// Create the input fields for username and password JTextField userText = new JTextField(20);

JPasswordField passwordText = new JPasswordField(20); userText.setFont(new Font("Arial", Font.*PLAIN*, 16));

passwordText.setFont(new Font("Arial", Font.*PLAIN*, 16));

// Input panel for username and password, aligned centrally JPanel inputPanel = new JPanel();

inputPanel.setLayout(new GridLayout(2, 2, 10, 10)); // Grid layout with padding inputPanel.setBackground(new Color(51, 102, 153));

inputPanel.add(new JLabel("Username:")); inputPanel.add(userText); inputPanel.add(new JLabel("Password:")); inputPanel.add(passwordText);

// Add the input panel to the main panel panel.add(inputPanel);

// Add space before the login button panel.add(Box.*createVerticalStrut*(30));

// Create a stylish login button with increased size and color JButton loginButton = new JButton("Login");

loginButton.setBackground(new Color(255, 204, 51)); // Bright yellow color for the button loginButton.setFont(new Font("Arial", Font.*BOLD*, 22)); // Larger font for better visibility loginButton.setPreferredSize(new Dimension(250, 70)); // Larger button size

loginButton.setFocusPainted(false); // Remove focus border

// Add button hover effect

loginButton.addMouseListener(new MouseAdapter() { public void mouseEntered(MouseEvent e) {

loginButton.setBackground(new Color(255, 140, 0)); // Darker color on hover

}

public void mouseExited(MouseEvent e) {

loginButton.setBackground(new Color(255, 204, 51)); // Original color

}

});

// Center the login button by creating a panel with a FlowLayout JPanel buttonPanel = new JPanel();

buttonPanel.setLayout(new FlowLayout(FlowLayout.*CENTER*)); // Center-align the button buttonPanel.setBackground(new Color(51, 102, 153)); // Set the background color of the button

panel

buttonPanel.add(loginButton);

// Add the button panel to the main panel panel.add(buttonPanel);

// Add the panel to the frame and center it frame.add(panel, BorderLayout.*CENTER*);

frame.setLocationRelativeTo(null); // Center the frame on the screen frame.setVisible(true);

// Action listener for the login button loginButton.addActionListener(e -> {

String username = userText.getText();

String password = new String(passwordText.getPassword()); if (authenticateAdmin(username, password)) {

JOptionPane.*showMessageDialog*(null, "Login successful!"); frame.dispose(); // Close the login window

showMainMenu(); // Navigate to the main menu

} else {

JOptionPane.*showMessageDialog*(null, "Invalid credentials!");

}

});

}

private boolean authenticateAdmin(String username, String password) { try {

String query = "SELECT \* FROM admin WHERE username = ? AND password = ?"; PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, username); statement.setString(2, password);

ResultSet resultSet = statement.executeQuery();

return resultSet.next(); // Return true if credentials match

} catch (SQLException e) { e.printStackTrace(); return false;

}

}

private void showMainMenu() {

mainFrame = new JFrame("Vehicle Rental System - Main Menu"); mainFrame.setSize(800, 600);

mainFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*); mainFrame.setLayout(new BorderLayout());

// Title Panel with a gradient background JPanel titlePanel = new JPanel();

titlePanel.setLayout(new BorderLayout());

titlePanel.setBackground(new Color(0, 123, 255)); // Stylish blue color

JLabel titleLabel = new JLabel("Welcome to the Vehicle Rental System", JLabel.*CENTER*); titleLabel.setFont(new Font("Arial", Font.*BOLD*, 24));

titleLabel.setForeground(Color.*WHITE*);

titlePanel.add(titleLabel, BorderLayout.*CENTER*);

// Add a decorative icon or logo

JLabel logoLabel = new JLabel(new ImageIcon("path\_to\_logo.png")); // Add logo image here titlePanel.add(logoLabel, BorderLayout.*WEST*); // Position logo on the left

// Main Panel with GridBagLayout for better control over the layout JPanel mainPanel = new JPanel(new GridBagLayout());

mainPanel.setBackground(new Color(245, 245, 245)); // Light background for the grid

GridBagConstraints gbc = new GridBagConstraints(); gbc.fill = GridBagConstraints.*HORIZONTAL*;

gbc.insets = new Insets(15, 15, 15, 15); // Add padding around each button

// Create buttons with larger size, different colors, and hover effects

JButton availabilityButton = createStyledButton("Vehicle Availability", new Color(50, 150, 255), new Color(30, 100, 180), e -> showAvailability());

JButton billButton = createStyledButton("Generate Bill", new Color(255, 100, 100), new Color(200, 70, 70), e -> generateBill());

JButton calendarButton = createStyledButton("Select Dates", new Color(100, 255, 100), new Color(70, 200, 70), e -> selectDates());

JButton maintenanceButton = createStyledButton("Maintenance Records", new Color(255, 165, 0), new Color(200, 130, 0), e -> showMaintenanceRecords());

JButton customerButton = createStyledButton("Customer Details", new Color(255, 215, 0), new Color(220, 170, 0), e -> showCustomerDetails());

JButton reservationsButton = createStyledButton("Past Reservations", new Color(0, 255, 255), new Color(0, 200, 200), e -> showPastReservations());

JButton addVehicleButton = createStyledButton("Add New Vehicle", new Color(255, 215, 0), new Color(220, 170, 0), e -> addNewVehicle());

JButton addCustomerButton = createStyledButton("Add New Customer", new Color(50, 150, 255), new Color(30, 100, 180), e -> addNewCustomer());

JButton maintenanceRecordButton = createStyledButton("Add Maintenance Record", new Color(70, 130, 180), new Color(50, 100, 150), e -> addMaintenanceRecord());

JButton addReservationButton = createStyledButton("Add New Reservation", new Color(255, 100, 100), new Color(200, 70, 70), e -> addNewReservation());

// Add buttons to the main panel using GridBagLayout gbc.gridx = 0;

gbc.gridy = 0;

mainPanel.add(availabilityButton, gbc);

gbc.gridx = 1;

gbc.gridy = 0;

mainPanel.add(billButton, gbc);

gbc.gridx = 0;

gbc.gridy = 1;

mainPanel.add(calendarButton, gbc);

gbc.gridx = 1;

gbc.gridy = 1;

mainPanel.add(maintenanceButton, gbc);

gbc.gridx = 0;

gbc.gridy = 2; mainPanel.add(customerButton, gbc);

gbc.gridx = 1;

gbc.gridy = 2;

mainPanel.add(reservationsButton, gbc);

gbc.gridx = 0;

gbc.gridy = 3; mainPanel.add(addVehicleButton, gbc);

gbc.gridx = 1;

gbc.gridy = 3; mainPanel.add(addCustomerButton, gbc);

gbc.gridx = 0;

gbc.gridy = 4;

mainPanel.add(maintenanceRecordButton, gbc);

gbc.gridx = 1;

gbc.gridy = 4;

mainPanel.add(addReservationButton, gbc);

// Add title panel and main panel to the frame mainFrame.add(titlePanel, BorderLayout.*NORTH*); mainFrame.add(mainPanel, BorderLayout.*CENTER*); mainFrame.setVisible(true);

JButton logoutButton = createStyledButton("Logout", new Color(255, 69, 0), new Color(200, 50, 0), e -> {

int confirm = JOptionPane.*showConfirmDialog*(mainFrame, "Are you sure you want to logout?", "Logout Confirmation", JOptionPane.*YES\_NO\_OPTION*);

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

mainFrame.dispose(); // Close the main menu window createAdminLogin(); // Show the login screen again

}

});

// Add the Logout button to the bottom of the main panel gbc.gridx = 0;

gbc.gridy = 5;

gbc.gridwidth = 2; // Span the button across two columns mainPanel.add(logoutButton, gbc);

}

// Method to create a stylish button with hover effects, custom colors, and larger size

private JButton createStyledButton(String text, Color backgroundColor, Color hoverColor, ActionListener action) {

JButton button = new JButton(text);

button.setFont(new Font("Arial", Font.*BOLD*, 16)); // Increased font size for bigger buttons button.setPreferredSize(new Dimension(200, 50)); // Increased button size

button.setBackground(backgroundColor); // Custom background color button.setForeground(Color.*WHITE*);

button.setFocusPainted(false); // Remove focus outline

button.setBorder(BorderFactory.*createLineBorder*(new Color(0, 123, 255), 2)); // Blue border

// Add hover effect

button.addMouseListener(new MouseAdapter() { public void mouseEntered(MouseEvent e) {

button.setBackground(hoverColor); // Change background on hover

}

public void mouseExited(MouseEvent e) {

button.setBackground(backgroundColor); // Reset to original color

}

});

button.addActionListener(action); return button;

}

// Method to show vehicle availability private void showAvailability() {

JFrame availabilityFrame = new JFrame("Vehicle Availability");

availabilityFrame.setSize(800, 500);

availabilityFrame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);

// Panel to hold the table and buttons

JPanel panel = new JPanel(new BorderLayout());

// Table columns

String[] columnNames = {"ID", "Name", "Type", "Status", "Price/Day"}; DefaultTableModel model = new DefaultTableModel(columnNames, 0); JTable availabilityTable = new JTable(model);

JScrollPane scrollPane = new JScrollPane(availabilityTable);

// Fetch vehicle data from the database and populate the table try {

String query = "SELECT \* FROM vehicles";

Statement statement = connection.createStatement(); ResultSet resultSet = statement.executeQuery(query);

while (resultSet.next()) {

Object[] row = new Object[5]; row[0] = resultSet.getInt("id");

row[1] = resultSet.getString("name"); row[2] = resultSet.getString("type"); row[3] = resultSet.getString("status");

row[4] = resultSet.getDouble("price\_per\_day"); model.addRow(row);

}

resultSet.close(); statement.close();

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

JOptionPane.*showMessageDialog*(availabilityFrame, "Error fetching vehicle data.");

}

// Button to delete the selected vehicle

JButton deleteButton = new JButton("Delete Selected Vehicle"); deleteButton.addActionListener(e -> {

int selectedRow = availabilityTable.getSelectedRow(); if (selectedRow != -1) { // Ensure a row is selected

int vehicleId = (int) model.getValueAt(selectedRow, 0); // Get the vehicle ID from the first

column

// Confirm deletion

int confirm = JOptionPane.*showConfirmDialog*( availabilityFrame,

"Are you sure you want to delete this vehicle?", "Confirm Deletion",

JOptionPane.*YES\_NO\_OPTION*

);

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

// Delete the selected vehicle from the database try {

String deleteQuery = "DELETE FROM vehicles WHERE id = ?";

PreparedStatement deleteStatement = connection.prepareStatement(deleteQuery); deleteStatement.setInt(1, vehicleId);

deleteStatement.executeUpdate(); deleteStatement.close();

// Remove the row from the table model.removeRow(selectedRow);

JOptionPane.*showMessageDialog*(availabilityFrame, "Vehicle deleted successfully.");

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

JOptionPane.*showMessageDialog*(availabilityFrame, "Error deleting vehicle.");

}

}

} else {

JOptionPane.*showMessageDialog*(availabilityFrame, "Please select a vehicle to delete.");

}

});

// Button to go back to the main menu

JButton backButton = new JButton("Back to Main Menu");

backButton.addActionListener(e -> availabilityFrame.dispose());

// Panel for buttons

JPanel buttonPanel = new JPanel(); buttonPanel.add(deleteButton); buttonPanel.add(backButton);

// Add components to the main panel panel.add(scrollPane, BorderLayout.*CENTER*); panel.add(buttonPanel, BorderLayout.*SOUTH*);

availabilityFrame.add(panel);

availabilityFrame.setVisible(true);

}

// Method to generate a bill

// Declare finalAmount as a class-level variable

private double finalAmount = 0.0; // Default value

public void generateBill() {

JFrame billFrame = new JFrame("Generate Bill"); billFrame.setSize(500, 400);

billFrame.setLayout(new BorderLayout());

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

JLabel customerLabel = new JLabel("Customer ID:"); JTextField customerField = new JTextField();

JLabel vehicleLabel = new JLabel("Vehicle ID:"); JTextField vehicleField = new JTextField();

JLabel rentalDaysLabel = new JLabel("Rental Days:"); JTextField rentalDaysField = new JTextField();

JLabel additionalChargesLabel = new JLabel("Additional Charges:"); JTextField additionalChargesField = new JTextField();

JLabel discountLabel = new JLabel("Discount (%):"); JTextField discountField = new JTextField();

JButton calculateButton = new JButton("Calculate"); JButton saveButton = new JButton("Save Bill");

JButton viewBillsButton = new JButton("View Bills"); // New button to view saved bills

JTextArea billArea = new JTextArea(); billArea.setEditable(false);

// Add fields to panel inputPanel.add(customerLabel); inputPanel.add(customerField); inputPanel.add(vehicleLabel); inputPanel.add(vehicleField); inputPanel.add(rentalDaysLabel); inputPanel.add(rentalDaysField);

inputPanel.add(additionalChargesLabel); inputPanel.add(additionalChargesField); inputPanel.add(discountLabel); inputPanel.add(discountField);

inputPanel.add(new JLabel()); // Empty space inputPanel.add(calculateButton);

// Add panels to frame

billFrame.add(inputPanel, BorderLayout.*NORTH*);

billFrame.add(new JScrollPane(billArea), BorderLayout.*CENTER*); JPanel buttonPanel = new JPanel(); buttonPanel.add(saveButton);

buttonPanel.add(viewBillsButton); // Add view bills button to panel billFrame.add(buttonPanel, BorderLayout.*SOUTH*);

// Action listener to calculate bill calculateButton.addActionListener(e -> {

try {

int vehicleID = Integer.*parseInt*(vehicleField.getText());

int rentalDays = Integer.*parseInt*(rentalDaysField.getText());

double additionalCharges = Double.*parseDouble*(additionalChargesField.getText()); double discountPercentage = Double.*parseDouble*(discountField.getText());

// Fetch the vehicle's daily price from database

String query = "SELECT price\_per\_day FROM vehicles WHERE id = ?"; PreparedStatement statement = connection.prepareStatement(query); statement.setInt(1, vehicleID);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

double dailyPrice = resultSet.getDouble("price\_per\_day"); double totalRentalCost = rentalDays \* dailyPrice;

double discountAmount = totalRentalCost \* (discountPercentage / 100);

finalAmount = totalRentalCost + additionalCharges - discountAmount; // Store finalAmount

// Display the bill details

StringBuilder billBuilder = new StringBuilder();

billBuilder.append("Customer ID: ").append(customerField.getText()).append("\n"); billBuilder.append("Vehicle ID: ").append(vehicleID).append("\n");

billBuilder.append("Rental Days: ").append(rentalDays).append("\n"); billBuilder.append("Daily Price: ").append(dailyPrice).append("\n");

billBuilder.append("Total Rental Cost: ").append(totalRentalCost).append("\n");

billBuilder.append("Additional Charges: ").append(additionalCharges).append("\n"); billBuilder.append("Discount: ").append(discountAmount).append("\n");

billBuilder.append("Final Amount: ").append(finalAmount).append("\n");

billArea.setText(billBuilder.toString());

} else {

JOptionPane.*showMessageDialog*(billFrame, "Vehicle not found.");

}

resultSet.close(); statement.close();

} catch (NumberFormatException ex) {

JOptionPane.*showMessageDialog*(billFrame, "Please enter valid numbers.");

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

}

});

// Action listener to save the bill to the database saveButton.addActionListener(e -> {

try {

java.sql.Date billDate = new java.sql.Date(new java.util.Date().getTime());

String insertQuery = "INSERT INTO bills (customer\_id, vehicle\_id, rental\_days, additional\_charges, discount, final\_amount, bill\_date) VALUES (?, ?, ?, ?, ?, ?, ?)";

PreparedStatement insertStatement = connection.prepareStatement(insertQuery);

insertStatement.setInt(1, Integer.*parseInt*(customerField.getText())); insertStatement.setInt(2, Integer.*parseInt*(vehicleField.getText())); insertStatement.setInt(3, Integer.*parseInt*(rentalDaysField.getText()));

insertStatement.setDouble(4, Double.*parseDouble*(additionalChargesField.getText())); insertStatement.setDouble(5, Double.*parseDouble*(discountField.getText()));

insertStatement.setDouble(6, finalAmount); insertStatement.setDate(7, billDate);

insertStatement.executeUpdate(); JOptionPane.*showMessageDialog*(billFrame, "Bill saved successfully!");

insertStatement.close();

} catch (SQLException | NumberFormatException ex) { ex.printStackTrace();

JOptionPane.*showMessageDialog*(billFrame, "Error saving the bill.");

}

});

// Action listener for the View Bills button viewBillsButton.addActionListener(e -> {

try {

// Fetch all saved bills from the database String viewQuery = "SELECT \* FROM bills";

PreparedStatement viewStatement = connection.prepareStatement(viewQuery); ResultSet resultSet = viewStatement.executeQuery();

StringBuilder billsBuilder = new StringBuilder(); while (resultSet.next()) {

billsBuilder.append("Bill ID: ").append(resultSet.getInt("bill\_id")).append("\n"); billsBuilder.append("Customer ID:

").append(resultSet.getInt("customer\_id")).append("\n");

billsBuilder.append("Vehicle ID: ").append(resultSet.getInt("vehicle\_id")).append("\n"); billsBuilder.append("Rental Days:

").append(resultSet.getInt("rental\_days")).append("\n"); billsBuilder.append("Additional Charges:

").append(resultSet.getDouble("additional\_charges")).append("\n");

billsBuilder.append("Discount: ").append(resultSet.getDouble("discount")).append("\n"); billsBuilder.append("Final Amount:

").append(resultSet.getDouble("final\_amount")).append("\n");

billsBuilder.append("Bill Date: ").append(resultSet.getDate("bill\_date")).append("\n"); billsBuilder.append(" \n");

}

JTextArea billsArea = new JTextArea(); billsArea.setEditable(false);

billsArea.setText(billsBuilder.toString());

// Create a new frame to display the bills

JFrame viewBillsFrame = new JFrame("View Bills"); viewBillsFrame.setSize(600, 400);

viewBillsFrame.add(new JScrollPane(billsArea)); viewBillsFrame.setVisible(true);

resultSet.close();

viewStatement.close();

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

JOptionPane.*showMessageDialog*(billFrame, "Error fetching bills.");

}

});

billFrame.setVisible(true);

}

// Method to show select dates page with start and end date pickers private void selectDates() {

JFrame dateFrame = new JFrame("Select Rental Dates"); dateFrame.setSize(400, 300);

dateFrame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(3, 2, 10, 10));

// Create labels for Start Date and End Date JLabel startLabel = new JLabel("Start Date:"); JLabel endLabel = new JLabel("End Date:");

// Create date pickers (JDateChooser) for selecting dates JDateChooser startDateChooser = new JDateChooser(); JDateChooser endDateChooser = new JDateChooser();

// Create button to proceed with the selected dates JButton proceedButton = new JButton("Proceed");

proceedButton.addActionListener(new ActionListener() { public void actionPerformed(ActionEvent e) {

Date startDate = startDateChooser.getDate(); Date endDate = endDateChooser.getDate();

// Validate if the dates are selected and if end date is after start date if (startDate != null && endDate != null) {

if (endDate.after(startDate)) {

SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd"); String startDateStr = dateFormat.format(startDate);

String endDateStr = dateFormat.format(endDate);

// Display the selected dates or you could process them (e.g., check availability) JOptionPane.*showMessageDialog*(dateFrame, "Start Date: " + startDateStr + "\nEnd

Date: " + endDateStr);

} else {

JOptionPane.*showMessageDialog*(dateFrame, "End Date must be after Start Date!");

dates.");

}

} else {

JOptionPane.*showMessageDialog*(dateFrame, "Please select both start and end

}

}

});

// Add components to the panel panel.add(startLabel); panel.add(startDateChooser); panel.add(endLabel); panel.add(endDateChooser);

panel.add(new JLabel()); // Empty label for spacing panel.add(proceedButton);

// Add the panel to the frame dateFrame.add(panel);

dateFrame.setVisible(true);

}

// Method to show maintenance records with a table view private void showMaintenanceRecords() {

JFrame maintenanceFrame = new JFrame("Maintenance Records"); maintenanceFrame.setSize(1000, 600);

maintenanceFrame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);

// Panel to hold the table

JPanel panel = new JPanel(new BorderLayout());

// Updated table columns based on your MySQL table structure String[] columnNames = {

"ID", "Vehicle ID", "Service Date", "Description", "Cost", "Mechanic Name", "Next Service Date"

};

DefaultTableModel model = new DefaultTableModel(columnNames, 0); JTable maintenanceTable = new JTable(model);

maintenanceTable.setFillsViewportHeight(true);

JScrollPane scrollPane = new JScrollPane(maintenanceTable);

// Fetch maintenance records from the database try {

String query = "SELECT id, vehicle\_id, service\_date, description, cost, mechanic\_name, next\_service\_date FROM maintenance\_records";

Statement statement = connection.createStatement(); ResultSet resultSet = statement.executeQuery(query);

while (resultSet.next()) {

Object[] row = new Object[7]; row[0] = resultSet.getInt("id");

row[1] = resultSet.getInt("vehicle\_id");

row[2] = resultSet.getDate("service\_date"); row[3] = resultSet.getString("description"); row[4] = resultSet.getDouble("cost");

row[5] = resultSet.getString("mechanic\_name"); row[6] = resultSet.getDate("next\_service\_date"); model.addRow(row);

}

resultSet.close(); statement.close();

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

JOptionPane.*showMessageDialog*(maintenanceFrame, "Error fetching maintenance records.");

}

// Add the table to the panel

panel.add(scrollPane, BorderLayout.*CENTER*);

// Adding a "Back" button to return to the main menu

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.*RIGHT*)); JButton backButton = new JButton("Back to Main Menu");

backButton.addActionListener(e -> maintenanceFrame.dispose()); buttonPanel.add(backButton);

panel.add(buttonPanel, BorderLayout.*SOUTH*);

// Setting up the frame with an enhanced appearance maintenanceFrame.add(panel);

maintenanceFrame.setVisible(true);

}

private void addMaintenanceRecord() {

JFrame maintenanceFrame = new JFrame("Add Maintenance Record"); maintenanceFrame.setSize(600, 500);

maintenanceFrame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*); maintenanceFrame.setLayout(new GridLayout(8, 2, 10, 10));

// Labels and input fields

JLabel vehicleIdLabel = new JLabel("Vehicle ID:"); JTextField vehicleIdField = new JTextField();

JLabel serviceDateLabel = new JLabel("Service Date (YYYY-MM-DD):"); JTextField serviceDateField = new JTextField();

JLabel descriptionLabel = new JLabel("Description:"); JTextArea descriptionArea = new JTextArea(3, 20);

JScrollPane descriptionScroll = new JScrollPane(descriptionArea);

JLabel costLabel = new JLabel("Cost:"); JTextField costField = new JTextField();

JLabel mechanicLabel = new JLabel("Mechanic Name:"); JTextField mechanicField = new JTextField();

JLabel nextServiceDateLabel = new JLabel("Next Service Date (YYYY-MM-DD):"); JTextField nextServiceDateField = new JTextField();

JButton saveButton = new JButton("Save Record"); JButton clearButton = new JButton("Clear");

// Add components to the frame maintenanceFrame.add(vehicleIdLabel); maintenanceFrame.add(vehicleIdField); maintenanceFrame.add(serviceDateLabel); maintenanceFrame.add(serviceDateField); maintenanceFrame.add(descriptionLabel); maintenanceFrame.add(descriptionScroll); maintenanceFrame.add(costLabel); maintenanceFrame.add(costField);

maintenanceFrame.add(mechanicLabel); maintenanceFrame.add(mechanicField); maintenanceFrame.add(nextServiceDateLabel); maintenanceFrame.add(nextServiceDateField); maintenanceFrame.add(saveButton); maintenanceFrame.add(clearButton);

// Action listener for the Save button saveButton.addActionListener(e -> {

String vehicleId = vehicleIdField.getText();

String serviceDate = serviceDateField.getText(); String description = descriptionArea.getText(); String cost = costField.getText();

String mechanicName = mechanicField.getText();

String nextServiceDate = nextServiceDateField.getText();

// Validate input fields

if (vehicleId.isEmpty() || serviceDate.isEmpty() || description.isEmpty() || cost.isEmpty()) { JOptionPane.*showMessageDialog*(maintenanceFrame, "Please fill in all required fields.",

"Error", JOptionPane.*ERROR\_MESSAGE*); return;

}

// Save data to the database try {

String query = "INSERT INTO maintenance\_records (vehicle\_id, service\_date, description, cost, mechanic\_name, next\_service\_date) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query); statement.setInt(1, Integer.*parseInt*(vehicleId));

statement.setDate(2, java.sql.Date.*valueOf*(serviceDate)); statement.setString(3, description);

statement.setDouble(4, Double.*parseDouble*(cost)); statement.setString(5, mechanicName);

statement.setDate(6, java.sql.Date.*valueOf*(nextServiceDate));

int rowsInserted = statement.executeUpdate(); if (rowsInserted > 0) {

JOptionPane.*showMessageDialog*(maintenanceFrame, "Maintenance record added successfully!");

maintenanceFrame.dispose();

}

statement.close();

} catch (Exception ex) { ex.printStackTrace();

JOptionPane.*showMessageDialog*(maintenanceFrame, "Error saving record: " + ex.getMessage());

}

});

// Action listener for the Clear button clearButton.addActionListener(e -> {

vehicleIdField.setText("");

serviceDateField.setText(""); descriptionArea.setText(""); costField.setText(""); mechanicField.setText("");

nextServiceDateField.setText("");

});

maintenanceFrame.setVisible(true);

}

// Method to create a date picker for selecting dates

// Method to show customer details

// Method to show customer details with a table view

// Method to show customer details with a table view private void showCustomerDetails() {

JFrame customerFrame = new JFrame("Customer Details"); customerFrame.setSize(800, 600);

customerFrame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);

// Panel to hold the table

JPanel panel = new JPanel(new BorderLayout());

// Table to display customer data

String[] columnNames = {"ID", "Name", "Contact Number", "Email", "Address", "Driver's License Number", "Date of Birth"};

DefaultTableModel model = new DefaultTableModel(columnNames, 0); JTable customerTable = new JTable(model);

JScrollPane scrollPane = new JScrollPane(customerTable);

// Fetch customer data from the database

try {

String query = "SELECT \* FROM customers"; // Make sure "drivers\_licence\_number" exists in the table

Statement statement = connection.createStatement(); ResultSet resultSet = statement.executeQuery(query);

while (resultSet.next()) {

Object[] row = new Object[7]; row[0] = resultSet.getInt("id");

row[1] = resultSet.getString("name");

row[2] = resultSet.getString("contact\_number"); row[3] = resultSet.getString("email");

row[4] = resultSet.getString("address");

row[5] = resultSet.getString("drivers\_license\_number"); // Updated column name row[6] = resultSet.getDate("date\_of\_birth"); // java.sql.Date to display in a table cell model.addRow(row);

}

resultSet.close(); statement.close();

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

JOptionPane.*showMessageDialog*(customerFrame, "Error fetching customer details.");

}

// Add the table to the panel

panel.add(scrollPane, BorderLayout.*CENTER*);

// Back button to go back to main menu

JButton backButton = new JButton("Back to Main Menu");

backButton.addActionListener(e -> customerFrame.dispose()); // Close the customer details window

panel.add(backButton, BorderLayout.*SOUTH*);

customerFrame.add(panel); customerFrame.setVisible(true);

}

// Method to show past reservations private void showPastReservations() {

JFrame reservationsFrame = new JFrame("Past Reservations"); reservationsFrame.setSize(800, 600);

reservationsFrame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);

// Panel to hold the table

JPanel panel = new JPanel(new BorderLayout());

// Define column names based on your reservations table structure

String[] columnNames = {"Reservation ID", "Customer ID", "Vehicle ID", "Start Date", "End Date", "Total Cost"};

DefaultTableModel model = new DefaultTableModel(columnNames, 0); JTable reservationsTable = new JTable(model);

reservationsTable.setFillsViewportHeight(true);

JScrollPane scrollPane = new JScrollPane(reservationsTable);

// Fetch past reservations from the database try {

String query = "SELECT reservation\_id, customer\_id, vehicle\_id, start\_date, end\_date, total\_cost FROM reservations";

Statement statement = connection.createStatement(); ResultSet resultSet = statement.executeQuery(query);

while (resultSet.next()) {

Object[] row = new Object[6];

row[0] = resultSet.getInt("reservation\_id"); row[1] = resultSet.getInt("customer\_id"); row[2] = resultSet.getInt("vehicle\_id"); row[3] = resultSet.getDate("start\_date"); row[4] = resultSet.getDate("end\_date"); row[5] = resultSet.getDouble("total\_cost"); model.addRow(row);

}

resultSet.close(); statement.close();

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

JOptionPane.*showMessageDialog*(reservationsFrame, "Error fetching past reservations.");

}

// Add the table to the panel

panel.add(scrollPane, BorderLayout.*CENTER*);

// Adding a "Back" button to return to the main menu

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.*RIGHT*)); JButton backButton = new JButton("Back to Main Menu");

backButton.addActionListener(e -> reservationsFrame.dispose()); buttonPanel.add(backButton);

panel.add(buttonPanel, BorderLayout.*SOUTH*);

reservationsFrame.add(panel);

reservationsFrame.setVisible(true);

}

private void addNewReservation() {

JFrame reservationFrame = new JFrame("Add New Reservation");

reservationFrame.setSize(400, 300);

reservationFrame.setDefaultCloseOperation(JFrame.*DISPOSE\_ON\_CLOSE*);

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(7, 2, 10, 10));

// Labels and inputs

JLabel vehicleLabel = new JLabel("Vehicle ID:"); JTextField vehicleField = new JTextField();

JLabel customerLabel = new JLabel("Customer ID:"); JTextField customerField = new JTextField();

JLabel startDateLabel = new JLabel("Start Date:"); JDateChooser startDateChooser = new JDateChooser(); startDateChooser.setDateFormatString("yyyy-MM-dd");

JLabel endDateLabel = new JLabel("End Date:"); JDateChooser endDateChooser = new JDateChooser(); endDateChooser.setDateFormatString("yyyy-MM-dd");

JLabel totalPriceLabel = new JLabel("Total Price:"); JTextField totalPriceField = new JTextField();

totalPriceField.setEditable(false);

// Calculate total price when dates are selected

startDateChooser.addPropertyChangeListener("date", e -> { if (!vehicleField.getText().isEmpty()) {

int vehicleId = Integer.*parseInt*(vehicleField.getText());

calculateTotalPrice(startDateChooser, endDateChooser, totalPriceField, vehicleId);

}

});

endDateChooser.addPropertyChangeListener("date", e -> { if (!vehicleField.getText().isEmpty()) {

int vehicleId = Integer.*parseInt*(vehicleField.getText());

calculateTotalPrice(startDateChooser, endDateChooser, totalPriceField, vehicleId);

}

});

JButton saveButton = new JButton("Save Reservation"); saveButton.addActionListener(e -> {

try {

int vehicleId = Integer.*parseInt*(vehicleField.getText());

int customerId = Integer.*parseInt*(customerField.getText());

java.sql.Date startDate = new java.sql.Date(startDateChooser.getDate().getTime()); java.sql.Date endDate = new java.sql.Date(endDateChooser.getDate().getTime()); double totalPrice = Double.*parseDouble*(totalPriceField.getText());

// Insert reservation into the database

String query = "INSERT INTO reservations (vehicle\_id, customer\_id, start\_date, end\_date,total\_cost) VALUES (?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(query); statement.setInt(1, vehicleId);

statement.setInt(2, customerId); statement.setDate(3, startDate); statement.setDate(4, endDate); statement.setDouble(5, totalPrice); statement.executeUpdate();

statement.close();

JOptionPane.*showMessageDialog*(reservationFrame, "Reservation added successfully!");

} catch (SQLException | NumberFormatException ex) { ex.printStackTrace();

JOptionPane.*showMessageDialog*(reservationFrame, "Error adding reservation.");

}

});

// Add components to the panel panel.add(vehicleLabel); panel.add(vehicleField); panel.add(customerLabel); panel.add(customerField); panel.add(startDateLabel); panel.add(startDateChooser); panel.add(endDateLabel); panel.add(endDateChooser); panel.add(totalPriceLabel); panel.add(totalPriceField); panel.add(saveButton);

reservationFrame.add(panel);

reservationFrame.setVisible(true);

}

private void calculateTotalPrice(JDateChooser startDateChooser, JDateChooser endDateChooser, JTextField totalPriceField, int vehicleId) {

if (startDateChooser.getDate() != null && endDateChooser.getDate() != null) { long diffInMillies = endDateChooser.getDate().getTime() -

startDateChooser.getDate().getTime();

long diffInDays = diffInMillies / (1000 \* 60 \* 60 \* 24);

// Fetch price per day from the database

double pricePerDay = getVehiclePricePerDay(vehicleId); double totalPrice = diffInDays \* pricePerDay;

totalPriceField.setText(String.*format*("%.2f", totalPrice));

}

}

private double getVehiclePricePerDay(int vehicleId) { double pricePerDay = 0.0;

try {

String query = "SELECT price\_per\_day FROM vehicles WHERE id = ?"; PreparedStatement statement = connection.prepareStatement(query); statement.setInt(1, vehicleId);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

pricePerDay = resultSet.getDouble("price\_per\_day");

}

resultSet.close(); statement.close();

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

JOptionPane.*showMessageDialog*(null, "Error fetching vehicle price.");

}

return pricePerDay;

}

// Method to add a new vehicle

// Method to add a new vehicle with detailed input private void addNewVehicle() {

JFrame vehicleFrame = new JFrame("Add New Vehicle"); vehicleFrame.setSize(400, 450);

vehicleFrame.setLayout(new BorderLayout());

JPanel inputPanel = new JPanel(new GridBagLayout()); GridBagConstraints gbc = new GridBagConstraints(); gbc.insets = new Insets(5, 5, 5, 5);

gbc.fill = GridBagConstraints.*HORIZONTAL*;

// Labels and TextFields

JLabel nameLabel = new JLabel("Vehicle Name:"); JTextField nameField = new JTextField();

JLabel typeLabel = new JLabel("Type (e.g., Car, Bike):"); JTextField typeField = new JTextField();

JLabel statusLabel = new JLabel("Status (Available/Unavailable):"); JTextField statusField = new JTextField();

JLabel priceLabel = new JLabel("Price per Day:"); JTextField priceField = new JTextField();

JLabel licenseLabel = new JLabel("License Plate:");

JTextField licenseField = new JTextField(); JLabel yearLabel = new JLabel("Model Year:"); JTextField yearField = new JTextField();

// Buttons

JButton addButton = new JButton("Add Vehicle"); JButton backButton = new JButton("Back to Menu");

// Add components to the panel with proper alignment gbc.gridx = 0; gbc.gridy = 0;

inputPanel.add(nameLabel, gbc); gbc.gridx = 1;

inputPanel.add(nameField, gbc);

gbc.gridx = 0; gbc.gridy = 1; inputPanel.add(typeLabel, gbc); gbc.gridx = 1;

inputPanel.add(typeField, gbc);

gbc.gridx = 0; gbc.gridy = 2; inputPanel.add(statusLabel, gbc); gbc.gridx = 1; inputPanel.add(statusField, gbc);

gbc.gridx = 0; gbc.gridy = 3; inputPanel.add(priceLabel, gbc); gbc.gridx = 1;

inputPanel.add(priceField, gbc);

gbc.gridx = 0; gbc.gridy = 4; inputPanel.add(licenseLabel, gbc); gbc.gridx = 1; inputPanel.add(licenseField, gbc);

gbc.gridx = 0; gbc.gridy = 5; inputPanel.add(yearLabel, gbc); gbc.gridx = 1;

inputPanel.add(yearField, gbc);

gbc.gridx = 0; gbc.gridy = 6; inputPanel.add(addButton, gbc);

gbc.gridx = 1; gbc.gridy = 6; inputPanel.add(backButton, gbc);

vehicleFrame.add(inputPanel, BorderLayout.*CENTER*);

// Action listener to add vehicle to the database

addButton.addActionListener(e -> { try {

// Retrieve input data

String name = nameField.getText(); String type = typeField.getText(); String status = statusField.getText();

double price = Double.*parseDouble*(priceField.getText()); String licensePlate = licenseField.getText();

int modelYear = Integer.*parseInt*(yearField.getText());

if (name.isEmpty() || type.isEmpty() || status.isEmpty() || licensePlate.isEmpty()) { JOptionPane.*showMessageDialog*(vehicleFrame, "Please fill out all the fields."); return;

}

// Check for duplicates

String checkQuery = "SELECT COUNT(\*) FROM vehicles WHERE license\_plate = ?"; PreparedStatement checkStatement = connection.prepareStatement(checkQuery); checkStatement.setString(1, licensePlate);

ResultSet resultSet = checkStatement.executeQuery(); resultSet.next();

int count = resultSet.getInt(1); checkStatement.close();

exists.");

if (count > 0) {

JOptionPane.*showMessageDialog*(vehicleFrame, "Vehicle with this license plate already

return;

}

// Insert into database

String insertQuery = "INSERT INTO vehicles (name, type, status, price\_per\_day, license\_plate, model\_year) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(insertQuery); statement.setString(1, name);

statement.setString(2, type); statement.setString(3, status); statement.setDouble(4, price);

statement.setString(5, licensePlate); statement.setInt(6, modelYear);

int rowsAffected = statement.executeUpdate(); if (rowsAffected > 0) {

JOptionPane.*showMessageDialog*(vehicleFrame, "Vehicle added successfully!");

}

statement.close();

} catch (NumberFormatException ex) {

JOptionPane.*showMessageDialog*(vehicleFrame, "Please enter valid numbers for price and

model year.");

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

JOptionPane.*showMessageDialog*(vehicleFrame, "Error adding vehicle.");

}

});

// Action listener for "Back to Menu" button

backButton.addActionListener(e -> vehicleFrame.dispose());

vehicleFrame.setVisible(true);

}

// Method to add a new customer

// Method to add a new customer with detailed input

// Method to add a new customer with detailed input private void addNewCustomer() {

JFrame customerFrame = new JFrame("Add New Customer"); customerFrame.setSize(400, 500);

customerFrame.setLayout(new BorderLayout());

JPanel inputPanel = new JPanel(new GridLayout(7, 2, 10, 10));

JLabel nameLabel = new JLabel("Customer Name:"); JTextField nameField = new JTextField();

JLabel contactLabel = new JLabel("Contact Number:"); JTextField contactField = new JTextField();

JLabel emailLabel = new JLabel("Email:"); JTextField emailField = new JTextField();

JLabel addressLabel = new JLabel("Address:"); JTextField addressField = new JTextField();

JLabel licenseLabel = new JLabel("Driver's License Number:"); JTextField licenseField = new JTextField();

JLabel dobLabel = new JLabel("Date of Birth (YYYY-MM-DD):"); JTextField dobField = new JTextField();

JButton addButton = new JButton("Add Customer"); inputPanel.add(nameLabel);

inputPanel.add(nameField);

inputPanel.add(contactLabel);

inputPanel.add(contactField); inputPanel.add(emailLabel); inputPanel.add(emailField); inputPanel.add(addressLabel); inputPanel.add(addressField); inputPanel.add(licenseLabel); inputPanel.add(licenseField); inputPanel.add(dobLabel); inputPanel.add(dobField);

inputPanel.add(new JLabel()); // Empty space inputPanel.add(addButton);

customerFrame.add(inputPanel, BorderLayout.*CENTER*);

// Action listener to add customer to the database addButton.addActionListener(e -> {

try {

String name = nameField.getText(); String contact = contactField.getText(); String email = emailField.getText(); String address = addressField.getText(); String license = licenseField.getText();

java.sql.Date dob = java.sql.Date.*valueOf*(dobField.getText()); // Use java.sql.Date.valueOf()

String insertQuery = "INSERT INTO customers (name, contact\_number, email, address, drivers\_license\_number, date\_of\_birth) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement statement = connection.prepareStatement(insertQuery);

statement.setString(1, name); statement.setString(2, contact); statement.setString(3, email); statement.setString(4, address); statement.setString(5, license); statement.setDate(6, dob);

int rowsAffected = statement.executeUpdate(); if (rowsAffected > 0) {

JOptionPane.*showMessageDialog*(customerFrame, "Customer added successfully!");

}

statement.close();

} catch (IllegalArgumentException ex) {

JOptionPane.*showMessageDialog*(customerFrame, "Please enter the date in YYYY-MM-DD format.");

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

JOptionPane.*showMessageDialog*(customerFrame, "Error adding customer.");

}

});

customerFrame.setVisible(true);

}

public static void main(String[] args) { new VehicleRentalSystem();

}

}

TEAM MEMBERS:

1. JANANY M-231901012
2. LAKSHMI PRIYA-231901025
3. KAVYA VARSHINI-231901021