ROYALENFIELD BOOKING MANAGEMENT SYSTEM

A MINI-PROJECT BY:

R.Siva Santhosh 230701318

R.V.Someash 230701324

in partial fulfillment of the award of the degree

OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI

An Autonomous Institute

CHENNAI

NOVEMBER 2024

BONAFIDECERTIFICATE

Certified that this project **"ROYALENFIELD BOOKING MANAGEMENT SYSTEMS"** is the bonafide work of **"SIVA SANTHSOH, SOMEASH"** who carried out the project work under my supervision.

Submitted for the practical examination held on _____

SIGNATURE

Mr.G.Saravana Gokul
Assistant Professor (SS),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

SIGNATURE

Ms.V.JANANEE
Assistant Professor(SG),
Computer Science and Engineering,
Rajalakshmi Engineering College
(Autonomous),
Thandalam,Chennai-602105

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLEOFCONTENTS

ABSTRACT

The Royal Enfield Booking Management System is a Java Swing-based application integrated with MySQL to streamline the booking, order management, and customer services for Royal Enfield motorcycles. It allows customers to place orders, view order details, update or delete existing orders, and ensure the availability of models. The system offers a user-friendly interface, with various tabs for managing different operations such as placing, updating, and viewing orders, as well as maintaining stock levels. The application ensures data integrity, secure authentication, and seamless management of bike orders, with real-time updates to the database.

INTRODUCTION

1.1 INTRODUCTION

In the modern world, managing customer bookings for motorcycle sales and services is a crucial aspect of business operations. The Royal Enfield Booking Management System leverages the power of Java Swing for the user interface and MySQL for database management to provide a comprehensive solution for booking, tracking, and managing orders for Royal Enfield motorcycles.
The system provides administrators with an intuitive platform to manage customer orders, bike stock, and service schedules. It also offers functionalities to place, update, delete, and view orders, ensuring a smooth and seamless experience for both the customer and the staff

1.2 IMPLEMENTATION

The implementation of the Royal Enfield Booking Management System involves several steps and components, which include:

1. **Login Authentication:**
    - The system starts with a login screen where the admin must authenticate using a predefined username and password.
    - Only authorized users (admins) can access the core functionalities of the application, ensuring security.
2. **Main Dashboard:**
    - After login, users are directed to the main dashboard, which includes multiple tabs for operations such as:
        - **Place Order**: Allows customers to book a motorcycle model by providing their details.
        - **View Orders**: Displays a list of all placed orders along with details like customer name, contact, model, and order date.
        - **Update Order**: Allows administrators to modify the details of existing orders such as customer name, model name, or contact number.
        - **Delete Order**: Provides functionality to delete a specific order using its order ID.
3. **Database Interaction:**
    - The system integrates with a MySQL database that stores essential information such as bike models, customer details, and order records.
    - The database schema includes tables such as bikes, customers, sales, and orders, with necessary foreign key relationships ensuring data integrity.

4. **SQL Operations:**
    - The system executes SQL queries for CRUD operations (Create, Read, Update, Delete) on the orders table.
    - The placeOrder method checks the stock of a bike before allowing the customer to place an order. If the model is in stock, the order is placed into the orders table, and the stock is updated accordingly.
    - The viewOrders method retrieves all existing orders and displays them in a user-friendly format.
    - The updateOrder and deleteOrder methods allow changes or removal of existing orders based on the given order ID.
5. **User Interface:**
    - The user interface is designed using Java Swing, providing a clean and responsive experience with tabbed navigation.
    - The layout uses grid and border layouts for better organization, with labels, text fields, buttons, and output areas for real-time data display.
6. **Security Features:**
    - The application includes basic login authentication to restrict access to the system.
    - Passwords and sensitive information are handled securely to ensure the safety of user data.

1.3 SCOPE OF THE PROJECT

The scope of the Royal Enfield Booking Management System is as follows:

- **Customer Interaction**: Customers can place orders for different Royal Enfield models, track their orders, and view available bikes.
- **Order Management**: Administrators can manage the lifecycle of orders — from creation to deletion. This includes placing, updating, and viewing orders.
- **Stock Management**: The system ensures that the stock of bikes is updated as orders are placed or modified, preventing over-selling of available bikes.
- **Real-Time Data**: Data such as the customer's order history and available bike models is stored and fetched from the database, providing real-time information.
- **User Roles**: Only authorized users (admins) have the ability to add, delete, or modify orders, providing role-based access control.
- **Scalability**: The application can be extended to include more features, such as generating reports, managing service appointments, or integrating payment systems.
- **Technology Integration**: Java Swing provides the graphical interface, while MySQL is used for managing backend data.

SYSTEMSPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS:

PROCESSOR : Intel i5

MEMORY SIZE : 4GB(Minimum)

HARD DISK : 500 GB of free space

2.2 SOFTWARE SPECIFICATIONS:

PROGRAMMING LANGUAGE : Java,

MySQL FRONT-END : Java

BACK-END : MySQL

OPERATING SYSTEM : Windows 11

PROGRAM

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;

public class RoyalEnfieldShowroomSystemSwing extends JFrame {

private static final String url = "jdbc:mysql://localhost:3306/showroom_db";
private static final String username = "root";
private static final String password = "Vigshan@2116";

private JTextField modelNameField, customerNameField, contactNumberField,
orderIdField;
private JTextArea outputArea;
private Connection connection;
private JPanel loginPanel, mainPanel;

public RoyalEnfieldShowroomSystemSwing() {
setTitle("Royal Enfield Showroom Management System");
setSize(500, 500);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

// Create the login screen first
createLoginScreen();
}

private void createLoginScreen() {
// Create the login panel
loginPanel = new JPanel();
loginPanel.setLayout(new GridLayout(3, 2));

JLabel usernameLabel = new JLabel("Username:");
JTextField usernameField = new JTextField();
JLabel passwordLabel = new JLabel("Password:");
JPasswordField passwordField = new JPasswordField();
JButton loginButton = new JButton("Login");
```

```
loginButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
String inputUsername = usernameField.getText().trim();
String inputPassword = new String(passwordField.getPassword()).trim();

// Validate the credentials (hardcoded for now)
if (inputUsername.equals("admin") && inputPassword.equals("admin123")) {
// If credentials are valid, move to the main system
loginPanel.setVisible(false); // Hide login screen
createMainScreen(); // Show the main screen
setContentPane(mainPanel); // Update the content pane to the main panel
revalidate(); // Revalidate the frame to show the new panel
} else {
JOptionPane.showMessageDialog(RoyalEnfieldShowroomSystemSwing.this, "Invalid
credentials", "Login Failed", JOptionPane.ERROR_MESSAGE);
}
}
});

// Add components to the login panel
loginPanel.add(usernameLabel);
loginPanel.add(usernameField);
loginPanel.add(passwordLabel);
loginPanel.add(passwordField);
loginPanel.add(loginButton);

// Set the login screen as the content pane
setContentPane(loginPanel);
}

private void createMainScreen() {
mainPanel = new JPanel();
mainPanel.setLayout(new BorderLayout());

// Create a tabbed pane to organize operations
JTabbedPane tabbedPane = new JTabbedPane();

// Create tabs for different operations
tabbedPane.addTab("Place Order", createPlaceOrderTab());
tabbedPane.addTab("View Orders", createViewOrdersTab());
tabbedPane.addTab("Update Order", createUpdateOrderTab());
```

```
tabbedPane.addTab("Delete Order", createDeleteOrderTab());

// Add the tabbed pane to the main panel
mainPanel.add(tabbedPane, BorderLayout.CENTER);

// Output Area to show results
outputArea = new JTextArea(10, 20);
JScrollPane scrollPane = new JScrollPane(outputArea);
mainPanel.add(scrollPane, BorderLayout.SOUTH);
}

// Create Place Order Tab
private JPanel createPlaceOrderTab() {
JPanel orderPanel = new JPanel();
orderPanel.setLayout(new GridLayout(4, 2));

JLabel customerNameLabel = new JLabel("Customer Name:");
customerNameField = new JTextField();
JLabel modelNameLabel = new JLabel("Model Name:");
modelNameField = new JTextField();
JLabel contactNumberLabel = new JLabel("Contact Number:");
contactNumberField = new JTextField();

JButton placeOrderButton = new JButton("Place Order");
placeOrderButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
placeOrder();
}
});

orderPanel.add(customerNameLabel);
orderPanel.add(customerNameField);
orderPanel.add(modelNameLabel);
orderPanel.add(modelNameField);
orderPanel.add(contactNumberLabel);
orderPanel.add(contactNumberField);
orderPanel.add(placeOrderButton);

return orderPanel;
}
```

```
// Create View Orders Tab
private JPanel createViewOrdersTab() {
JPanel viewPanel = new JPanel();
viewPanel.setLayout(new BorderLayout());

JButton viewButton = new JButton("View Orders");
viewButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
viewOrders();
}
});

viewPanel.add(viewButton, BorderLayout.CENTER);
return viewPanel;
}

// Create Update Order Tab
private JPanel createUpdateOrderTab() {
JPanel updatePanel = new JPanel();
updatePanel.setLayout(new GridLayout(5, 2));

JLabel orderIdLabel = new JLabel("Order ID:");
orderIdField = new JTextField();
JLabel newCustomerNameLabel = new JLabel("New Customer Name:");
JTextField newCustomerNameField = new JTextField();
JLabel newModelNameLabel = new JLabel("New Model Name:");
JTextField newModelNameField = new JTextField();
JLabel newContactNumberLabel = new JLabel("New Contact Number:");
JTextField newContactNumberField = new JTextField();

JButton updateButton = new JButton("Update Order");
updateButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
// Get the input data from the fields
String orderId = orderIdField.getText().trim();
String newCustomerName = newCustomerNameField.getText().trim();
String newModelName = newModelNameField.getText().trim();
String newContactNumber = newContactNumberField.getText().trim();
```

```
if (orderId.isEmpty() || newCustomerName.isEmpty() || newModelName.isEmpty() ||
newContactNumber.isEmpty()) {
outputArea.setText("Please fill all fields");
} else {
// Call the method to update the order with the provided data
updateOrder(orderId, newCustomerName, newModelName, newContactNumber);
}
}
});

updatePanel.add(orderIdLabel);
updatePanel.add(orderIdField);
updatePanel.add(newCustomerNameLabel);
updatePanel.add(newCustomerNameField);
updatePanel.add(newModelNameLabel);
updatePanel.add(newModelNameField);
updatePanel.add(newContactNumberLabel);
updatePanel.add(newContactNumberField);
updatePanel.add(updateButton);

return updatePanel;
}

// Create Delete Order Tab
private JPanel createDeleteOrderTab() {
JPanel deletePanel = new JPanel();
deletePanel.setLayout(new BorderLayout());

JLabel orderIdLabel = new JLabel("Enter Order ID to delete:");
JTextField orderIdToDeleteField = new JTextField();
JButton deleteButton = new JButton("Delete Order");

deleteButton.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent e) {
deleteOrder(orderIdToDeleteField.getText());
}
});

deletePanel.add(orderIdLabel, BorderLayout.NORTH);
deletePanel.add(orderIdToDeleteField, BorderLayout.CENTER);
deletePanel.add(deleteButton, BorderLayout.SOUTH);
```

```
return deletePanel;
}

// Place Order Method
private void placeOrder() {
String customerName = customerNameField.getText();
String modelName = modelNameField.getText();
String contactNumber = contactNumberField.getText();

// Check if the bike model is in stock and fetch its price
String sqlCheckStock = "SELECT price, stock FROM models WHERE model_name = ?";
try (PreparedStatement stmtCheckStock = connection.prepareStatement(sqlCheckStock)) {
stmtCheckStock.setString(1, modelName);
ResultSet rs = stmtCheckStock.executeQuery();

if (rs.next()) {
double price = rs.getDouble("price");
int stock = rs.getInt("stock");

if (stock > 0) {
// Place the order if stock is available
placeOrderInDatabase(customerName, modelName, contactNumber, price);
} else {
outputArea.setText("Sorry, the bike model '" + modelName + "' is out of stock.");
}
} else {
outputArea.setText("Sorry, the bike model '" + modelName + "' is not available.");
}

} catch (SQLException e) {
e.printStackTrace();
outputArea.setText("Error occurred while checking stock.");
}
}

// Method to actually insert the order into the database
private void placeOrderInDatabase(String customerName, String modelName, String
contactNumber, double price) {
String sql = "INSERT INTO orders (customer_name, model_name, contact_number)
VALUES (?, ?, ?)";
```

```
try (PreparedStatement stmt = connection.prepareStatement(sql)) {
stmt.setString(1, customerName);
stmt.setString(2, modelName);
stmt.setString(3, contactNumber);

int rowsAffected = stmt.executeUpdate();
if (rowsAffected > 0) {
outputArea.setText("Order placed successfully! The cost of the bike is: ₹" + price);
} else {
outputArea.setText("Order failed.");
}
} catch (SQLException e) {
e.printStackTrace();
outputArea.setText("Error occurred while placing the order.");
}
}

// View Orders Method
private void viewOrders() {
String sql = "SELECT order_id, customer_name, model_name, contact_number, order_date
FROM orders";

try (Statement stmt = connection.createStatement();
ResultSet rs = stmt.executeQuery(sql)) {

StringBuilder result = new StringBuilder();
result.append("Order ID | Customer Name | Model Name | Contact Number | Order Date\n");
result.append(" \n");

while (rs.next()) {
int orderId = rs.getInt("order_id");
String customerName = rs.getString("customer_name");
String modelName = rs.getString("model_name");
String contactNumber = rs.getString("contact_number");
String orderDate = rs.getTimestamp("order_date").toString();

result.append(orderId).append(" | ")
.append(customerName).append(" | ")
.append(modelName).append(" | ")
.append(contactNumber).append(" | ")
.append(orderDate).append("\n");
}
```

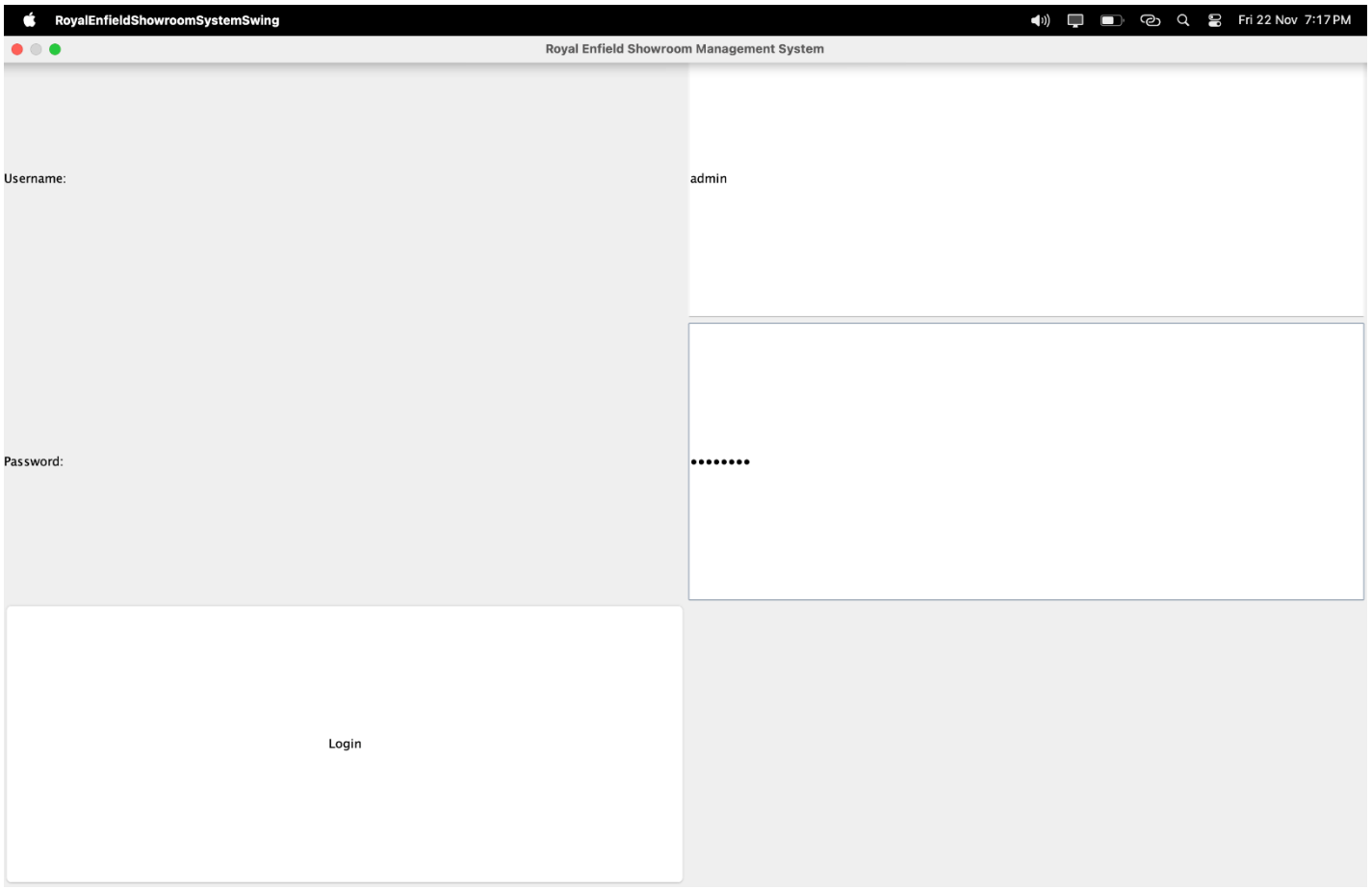
```
outputArea.setText(result.toString());
} catch (SQLException e) {
e.printStackTrace();
outputArea.setText("Error occurred while fetching orders.");
}
}

// Update Order Method
private void updateOrder(String orderId, String newCustomerName, String newModelName,
String newContactNumber) {
String sql = "UPDATE orders SET customer_name = ?, model_name = ?, contact_number =
? WHERE order_id = ?";

try (PreparedStatement stmt = connection.prepareStatement(sql)) {
stmt.setString(1, newCustomerName);
stmt.setString(2, newModelName);
stmt.setString(3, newContactNumber);
stmt.setInt(4, Integer.parseInt(orderId)); // Convert order ID to integer

int rowsAffected = stmt.executeUpdate();
if (rowsAffected > 0) {
outputArea.setText("Order updated successfully.");
} else {
outputArea.setText("Failed to update order. Make sure the order ID is correct.");
}
} catch (SQLException e) {
e.printStackTrace();
outputArea.setText("Error occurred while updating the order.");
}
}

// Delete Order Method
private void deleteOrder(String orderId) {
String sql = "DELETE FROM orders WHERE order_id = ?";

try (PreparedStatement stmt = connection.prepareStatement(sql)) {
stmt.setInt(1, Integer.parseInt(orderId)); // Convert order ID to integer

int rowsAffected = stmt.executeUpdate();
if (rowsAffected > 0) {
outputArea.setText("Order deleted successfully.");
```

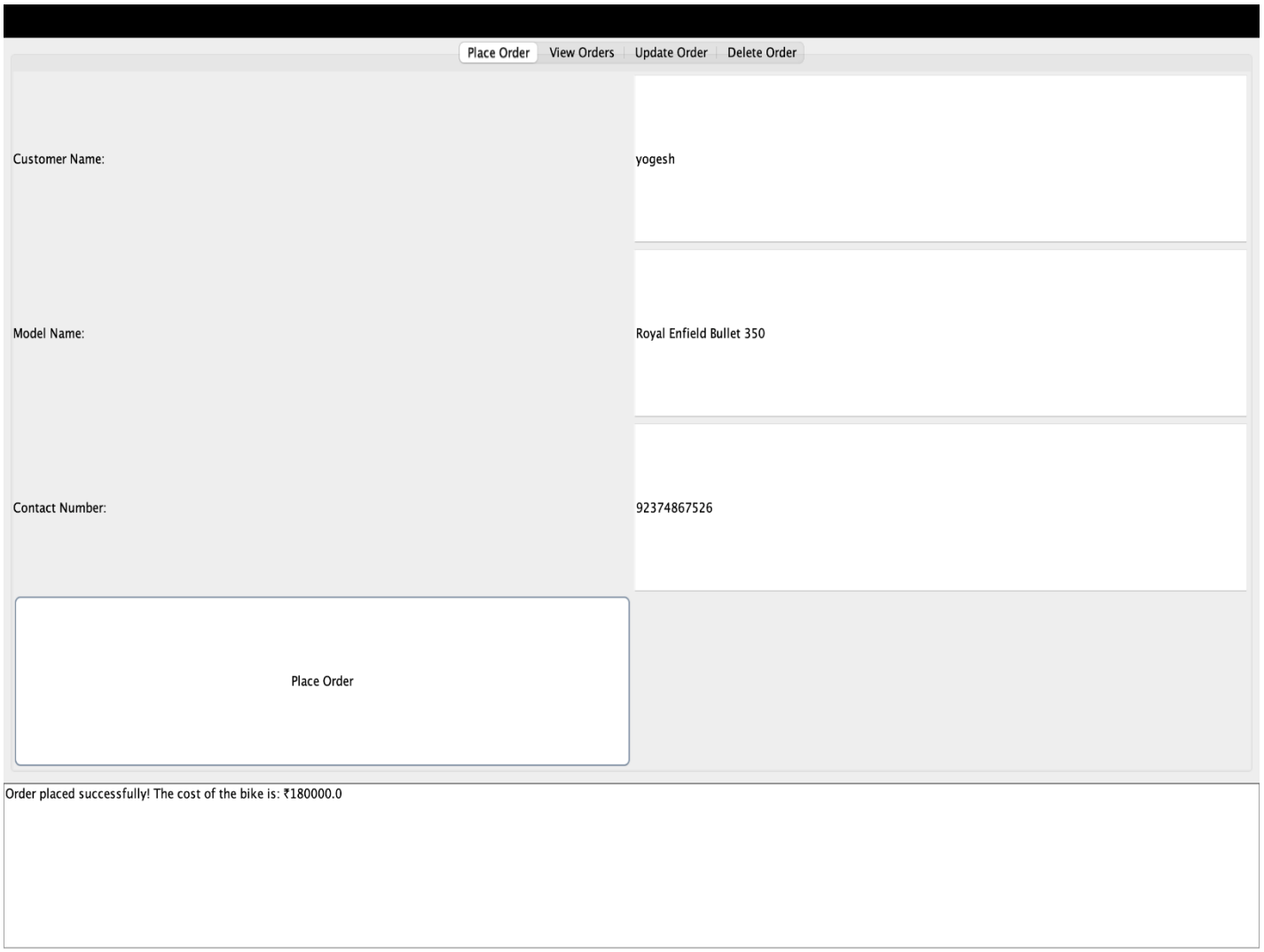
```
} else {
outputArea.setText("Failed to delete order. Make sure the order ID is correct.");
}
} catch (SQLException e) {
e.printStackTrace();
outputArea.setText("Error occurred while deleting the order.");
}
}

public static void main(String[] args) {
SwingUtilities.invokeLater(new Runnable() {
@Override
public void run() {
new RoyalEnfieldShowroomSystemSwing().setVisible(true);
}
});
}
}
```

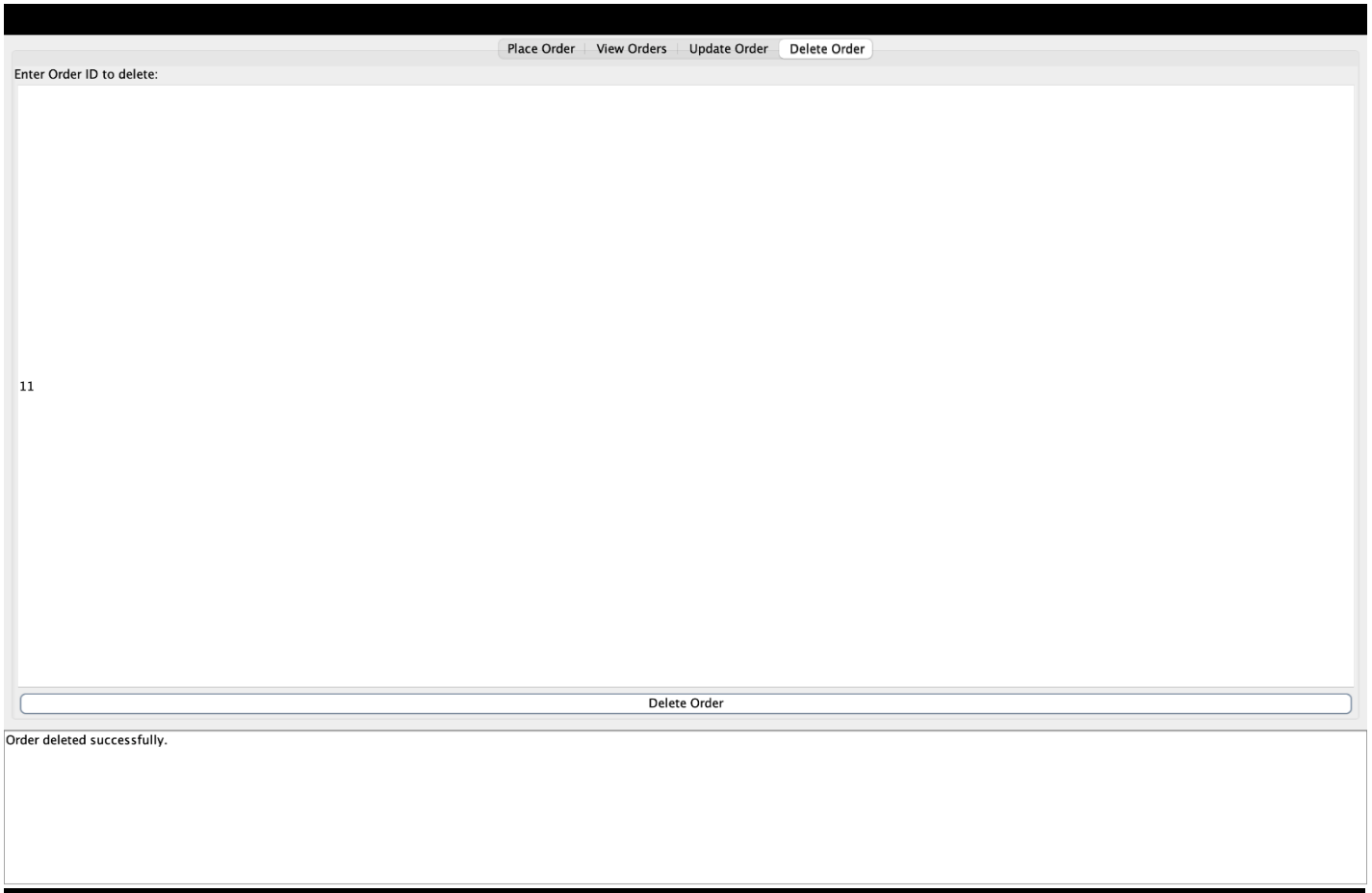
JDBC CONNECTION

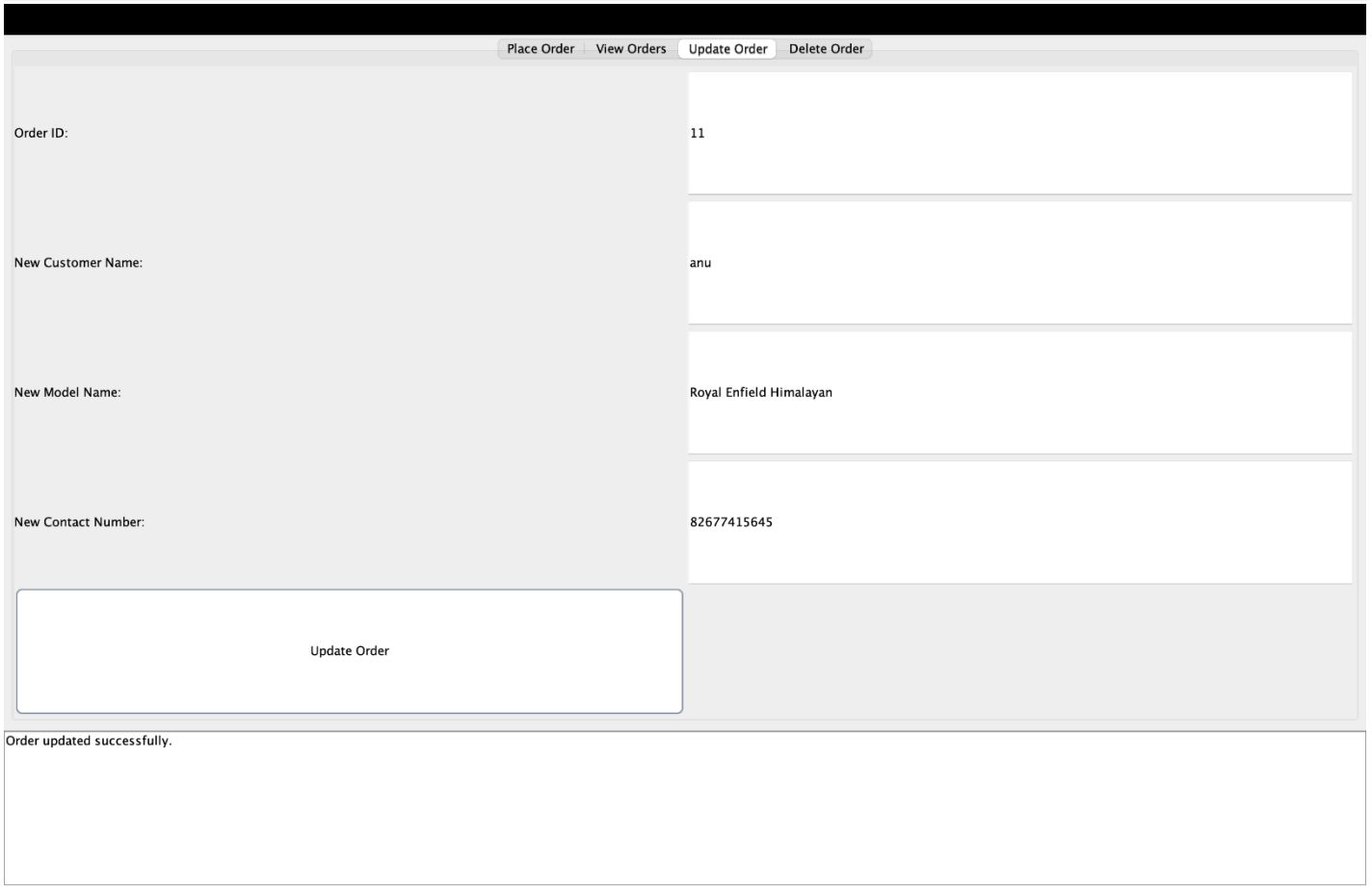
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
    private static final String URL =
"jdbc:mysql://localhost:3306/doctor_patient_system";
    private static final String USERNAME = "root"; private
    static final String PASSWORD = "Shankysathu2!";
    public static Connection connect() {
        try {
            return DriverManager.getConnection(URL, USERNAME, PASSWORD);
        } catch (SQLException e) {
            System.err.println("Database connection
            failed!"); e.printStackTrace(); return null;
        }
    }
}
```

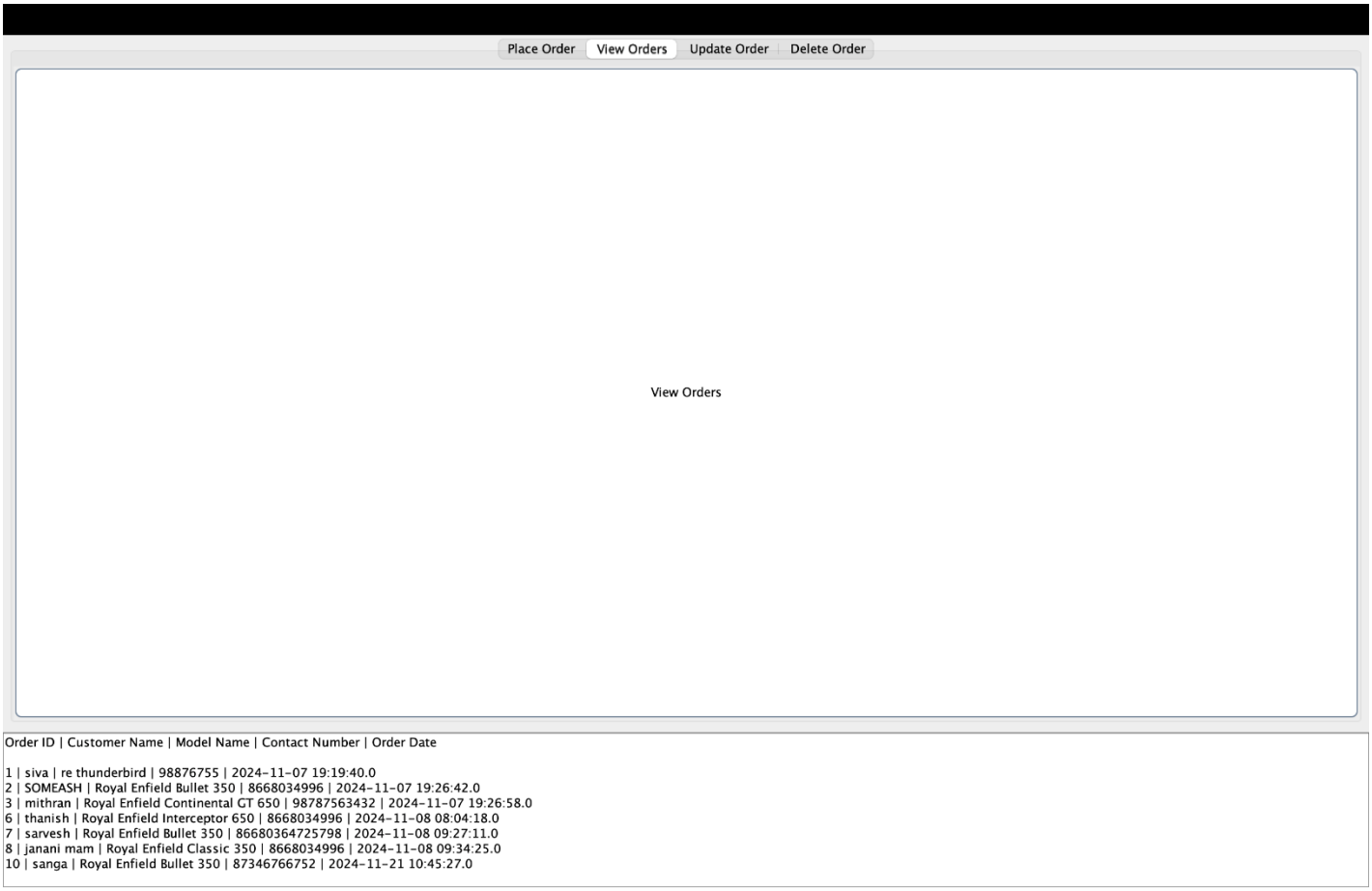
JAVA UI PICTURES




Place Order
View Orders
Update Order
Delete Order
Customer Name:
yogesh
Model Name:
Royal Enfield Bullet 350
Contact Number:
92374867526
Place Order
Order placed successfully! The cost of the bike is: ₹180000.0


Place Order
View Orders
Update Order
Delete Order
Enter Order ID to delete:
11
Delete Order
Order deleted successfully.


Place Order
View Orders
Update Order
Delete Order
Order ID:
11
New Customer Name:
anu
New Model Name:
Royal Enfield Himalayan
New Contact Number:
82677415645
Update Order
Order updated successfully.



Order ID | Customer Name | Model Name | Contact Number | Order Date

1 | siva | re thunderbird | 98876755 | 2024-11-07 19:19:40.0
2 | SOMEASH | Royal Enfield Bullet 350 | 8668034996 | 2024-11-07 19:26:42.0
3 | mithran | Royal Enfield Continental GT 650 | 98787563432 | 2024-11-07 19:26:58.0
6 | thanish | Royal Enfield Interceptor 650 | 8668034996 | 2024-11-08 08:04:18.0
7 | sarvesh | Royal Enfield Bullet 350 | 86680364725798 | 2024-11-08 09:27:11.0
8 | janani mam | Royal Enfield Classic 350 | 8668034996 | 2024-11-08 09:34:25.0
10 | sanga | Royal Enfield Bullet 350 | 87346766752 | 2024-11-21 10:45:27.0

CONCLUSION

The Royal Enfield Showroom Management System is a powerful tool designed to automate the management of customer orders and bike stock. By utilizing Java Swing for the user interface and MySQL for the backend, the system offers a responsive and secure solution for showroom operations. The implementation of features like order placement, viewing, updating, and deletion improves the overall workflow and provides a reliable solution for showroom staff. The project also offers potential for future expansion, making it a flexible and scalable solution for motorcycle dealerships.

REFERENCES

1. **https://www.javatpoint.com/java-tutorial**
2. https://www.wikipedia.org/
3. https://www.w3schools.com/sql/
4. SQL | Codecademy
5. https://chatgpt.com/