DATABASE CODE

```
CREATE DATABASE showroom_db
use showroom_db
CREATE TABLE bikes (
    bike_id INT AUTO_INCREMENT PRIMARY KEY,
    bike_name VARCHAR(100),
    bike_price DECIMAL(10, 2),
    stock_quantity INT
);
CREATE TABLE customers (
    customer_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100),
    phone VARCHAR(15),
    email VARCHAR(100)
);
CREATE TABLE sales (
    sale_id INT AUTO_INCREMENT PRIMARY KEY,
    bike_id INT,
    customer_id INT,
    sale_date DATE,
    quantity INT,
    total_amount DECIMAL(10, 2),
    FOREIGN KEY (bike_id) REFERENCES bikes(bike_id),
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);
CREATE TABLE orders (
    order_id INT AUTO_INCREMENT PRIMARY KEY,
```

```
    customer_name VARCHAR(255),
    model_name VARCHAR(255),
    contact_number VARCHAR(20),
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
create table model2(model_id int,model_name varchar(500),price float,stock
int);
INSERT INTO model2(model_id, model_name, price, stock)
VALUES
(1, 'Royal Enfield Classic 350', 190000.00, 10),
(2, 'Royal Enfield Meteor 350', 210000.00, 5),
(3, 'Royal Enfield Interceptor 650', 300000.00, 3),
(4, 'Royal Enfield Continental GT 650', 310000.00, 2),
(5, 'Royal Enfield Bullet 350', 180000.00, 15),
(6, 'Royal Enfield Himalayan', 230000.00, 8);
```

JAVA CODE

```
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);
}
});
}
}
```