Source Code

Frontend (java) code

```
import java.awt.*;
import java.sql.*;
import java.util.ArrayList;
import javax.swing.*;

class FoodItem {
    int id;
    String name;
    double price;

    FoodItem(int id, String name, double price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    @Override
    public String toString() {
        return id + ". " + name + " - ₹" + price;
    }
}

class CartItem {
    FoodItem foodItem;
    int quantity;
```

```
    CartItem(FoodItem foodItem, int quantity) {
        this.foodItem = foodItem;
        this.quantity = quantity;
    }

    double getTotalPrice() {
        return foodItem.price * quantity;
    }

    @Override
    public String toString() {
        return foodItem.name + " x " + quantity + " = ₹" + getTotalPrice();
    }
}

public class FoodDeliveryAppSwing extends JFrame {
    private static Connection connection;
    private static ArrayList<FoodItem> menu = new ArrayList<>();
    private static ArrayList<CartItem> cart = new ArrayList<>();
    private static String loggedInUsername;

    private JList<String> menuList;
    private DefaultListModel<String> menuListModel;
    private JList<String> cartList;
    private DefaultListModel<String> cartListModel;
    private JButton addToCartButton, editCartButton, deleteCartButton,
placeOrderButton, viewProfileButton;
```

```
    private JLabel totalAmountLabel;

    // New components for Login and Signup
    private JTextField usernameField, signUpUsernameField, signUpNameField, signUpAddressField, signUpEmailField;
    private JPasswordField passwordField, signUpPasswordField;

    public FoodDeliveryAppSwing() {
        // Initialize JFrame
        setTitle("Food Delivery App - Sign Up");
        setSize(400, 300);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null); // Center the window

        // Sign-Up Page UI
        JLabel nameLabel = new JLabel("Name: ");
        JLabel addressLabel = new JLabel("Address: ");
        JLabel emailLabel = new JLabel("Email: ");
        JLabel usernameLabel = new JLabel("Username: ");
        JLabel passwordLabel = new JLabel("Password: ");
        signUpNameField = new JTextField();
        signUpAddressField = new JTextField();
        signUpEmailField = new JTextField();
        signUpUsernameField = new JTextField();
        signUpPasswordField = new JPasswordField();
        JButton signUpSubmitButton = new JButton("Sign Up");
        JButton loginButton = new JButton("Already registered? Login");
```

```
    signUpSubmitButton.addActionListener(e -> signUpUser());
    loginButton.addActionListener(e -> showLoginPage());

    JPanel signUpPanel = new JPanel(new GridLayout(6, 2));
    signUpPanel.add(nameLabel);
    signUpPanel.add(signUpNameField);
    signUpPanel.add(addressLabel);
    signUpPanel.add(signUpAddressField);
    signUpPanel.add(emailLabel);
    signUpPanel.add(signUpEmailField);
    signUpPanel.add(usernameLabel);
    signUpPanel.add(signUpUsernameField);
    signUpPanel.add(passwordLabel);
    signUpPanel.add(signUpPasswordField);
    signUpPanel.add(loginButton);
    signUpPanel.add(signUpSubmitButton);

    add(signUpPanel, BorderLayout.CENTER);
}

private void showLoginPage() {
    // Remove Sign-Up Page UI before showing Login page
    getContentPane().removeAll();
    revalidate();
    repaint();

    JFrame loginFrame = new JFrame("Login");
```

```
    loginFrame.setSize(400, 300);
    loginFrame.setLocationRelativeTo(null);

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

    loginSubmitButton.addActionListener(e -> loginUser());

    JPanel loginPanel = new JPanel(new GridLayout(3, 2));
    loginPanel.add(usernameLabel);
    loginPanel.add(usernameField);
    loginPanel.add(passwordLabel);
    loginPanel.add(passwordField);
    loginPanel.add(new JLabel());
    loginPanel.add(loginSubmitButton);

    loginFrame.add(loginPanel);
    loginFrame.setVisible(true);
}

private void loginUser() {
    String username = usernameField.getText();
    String password = new String(passwordField.getPassword());
```

```
        if (username.isEmpty() || password.isEmpty()) {
            showMessage("Error", "Please enter both username and password.");
            return;
        }

        try {
            String query = "SELECT * FROM users WHERE username = ? AND
password = ?";
            PreparedStatement pst = connection.prepareStatement(query);
            pst.setString(1, username);
            pst.setString(2, password);
            ResultSet rs = pst.executeQuery();

            if (rs.next()) {
                loggedInUsername = username; // Set the logged-in user
                showMessage("Success", "Login successful!");
                showDashboard(); // Show the dashboard after login
            } else {
                showMessage("Error", "Invalid credentials.");
            }
        } catch (SQLException e) {
            showMessage("Error", "Error checking user credentials: " +
e.getMessage());
        }
    }

    private void signUpUser() {
        String name = signUpNameField.getText();
```

```
        String address = signUpAddressField.getText();
        String email = signUpEmailField.getText();
        String username = signUpUsernameField.getText();
        String password = new String(signUpPasswordField.getPassword());

        if (name.isEmpty() || address.isEmpty() || email.isEmpty() ||
username.isEmpty() || password.isEmpty()) {
            showMessage("Error", "Please fill in all the fields.");
            return;
        }

        try {
            String query = "INSERT INTO users (name, address, email, username,
password) VALUES (?, ?, ?, ?, ?)";
            PreparedStatement pst = connection.prepareStatement(query);
            pst.setString(1, name);
            pst.setString(2, address);
            pst.setString(3, email);
            pst.setString(4, username);
            pst.setString(5, password);
            int result = pst.executeUpdate();

            if (result > 0) {
                showMessage("Success", "Sign up successful! You can now login.");
                showLoginPage(); // Show login page after successful sign-up
            } else {
                showMessage("Error", "Error signing up. Please try again.");
            }
```

```
    } catch (SQLException e) {
        showMessage("Error", "Error signing up: " + e.getMessage());
    }
}

private void showDashboard() {
    // Remove previous UI before showing the dashboard
    getContentPane().removeAll();
    revalidate();
    repaint();

    JFrame dashboardFrame = new JFrame("Dashboard");
    dashboardFrame.setSize(600, 400);
    dashboardFrame.setLocationRelativeTo(null);

    // Add buttons to view menu and profile
    JButton viewMenuButton = new JButton("View Menu");
    JButton viewProfileButton = new JButton("Profile");

    viewMenuButton.addActionListener(e -> showMenuPage());
    viewProfileButton.addActionListener(e -> showUserProfile());

    JPanel dashboardPanel = new JPanel(new GridLayout(2, 1));
    dashboardPanel.add(viewMenuButton);
    dashboardPanel.add(viewProfileButton);

    dashboardFrame.add(dashboardPanel, BorderLayout.CENTER);
```

```
    dashboardFrame.setVisible(true);
}

private void showUserProfile() {
    // Fetch user profile details
    try {
        String query = "SELECT * FROM users WHERE username = ?";
        PreparedStatement pst = connection.prepareStatement(query);
        pst.setString(1, loggedInUsername);
        ResultSet rs = pst.executeQuery();

        if (rs.next()) {
            String name = rs.getString("name");
            String address = rs.getString("address");
            String email = rs.getString("email");

            // Profile page with Logout button
            JFrame profileFrame = new JFrame("User Profile");
            profileFrame.setSize(400, 300);
            profileFrame.setLocationRelativeTo(null);

            JLabel nameLabel = new JLabel("Name: " + name);
            JLabel addressLabel = new JLabel("Address: " + address);
            JLabel emailLabel = new JLabel("Email: " + email);

            JButton logoutButton = new JButton("Logout");
            logoutButton.addActionListener(e -> logoutUser());
```

```
            JPanel profilePanel = new JPanel(new GridLayout(4, 1));
            profilePanel.add(nameLabel);
            profilePanel.add(addressLabel);
            profilePanel.add(emailLabel);
            profilePanel.add(logoutButton);

            profileFrame.add(profilePanel, BorderLayout.CENTER);
            profileFrame.setVisible(true);
        }
    } catch (SQLException e) {
        showMessage("Error", "Error fetching user profile: " + e.getMessage());
    }
}

private void logoutUser() {
    loggedInUsername = null;

    // Close all open frames except for the main application frame
    Window[] windows = Window.getWindows();
    for (Window window : windows) {
        if (window instanceof JFrame && window.isShowing()) {
            window.dispose();
        }
    }

    // Show a success message
```

```
    showMessage("Success", "You have been logged out.");

    // Open the login page
    showLoginPage();
}


private void showMenuPage() {
    // Remove previous UI before showing the menu
    getContentPane().removeAll();
    revalidate();
    repaint();

    JFrame menuFrame = new JFrame("Menu");
    menuFrame.setSize(600, 400);
    menuFrame.setLocationRelativeTo(null);

    menuListModel = new DefaultListModel<>();
    cartListModel = new DefaultListModel<>();
    menuList = new JList<>(menuListModel);
    cartList = new JList<>(cartListModel);
    totalAmountLabel = new JLabel("Total Amount: ₹0");

    addToCartButton = new JButton("Add to Cart");
    editCartButton = new JButton("Edit Cart");
    deleteCartButton = new JButton("Delete Cart");
    placeOrderButton = new JButton("Place Order");
```

```
    addToCartButton.addActionListener(e -> addItemToCart());
    editCartButton.addActionListener(e -> editCartItem());
    deleteCartButton.addActionListener(e -> deleteCartItem());
    placeOrderButton.addActionListener(e -> placeOrder());

    // Load Menu items
    loadMenu();

    JPanel menuPanel = new JPanel(new GridLayout(1, 2));
    menuPanel.add(new JScrollPane(menuList));
    menuPanel.add(new JScrollPane(cartList));

    JPanel buttonPanel = new JPanel(new FlowLayout());
    buttonPanel.add(addToCartButton);
    buttonPanel.add(editCartButton);
    buttonPanel.add(deleteCartButton);
    buttonPanel.add(placeOrderButton);

    menuFrame.add(menuPanel, BorderLayout.CENTER);
    menuFrame.add(buttonPanel, BorderLayout.SOUTH);
    menuFrame.setVisible(true);
}

private void loadMenu() {
    try {
        String query = "SELECT * FROM menu";
```

```
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery(query);

        menu.clear();
        menuListModel.clear();

        while (rs.next()) {
            int id = rs.getInt("id");
            String name = rs.getString("name");
            double price = rs.getDouble("price");
            FoodItem item = new FoodItem(id, name, price);
            menu.add(item);
            menuListModel.addElement(item.toString());
        }
    } catch (SQLException e) {
        showMessage("Error", "Error loading menu: " + e.getMessage());
    }
}

private void addItemToCart() {
    int selectedIndex = menuList.getSelectedIndex();
    if (selectedIndex == -1) {
        showMessage("Error", "Please select a food item to add to the cart.");
        return;
    }

    FoodItem selectedItem = menu.get(selectedIndex);
```

```
        String quantityString = JOptionPane.showInputDialog("Enter Quantity: ");
        try {
            int quantity = Integer.parseInt(quantityString);
            if (quantity > 0) {
                cart.add(new CartItem(selectedItem, quantity));
                updateCart();
            } else {
                showMessage("Error", "Invalid quantity.");
            }
        } catch (NumberFormatException e) {
            showMessage("Error", "Please enter a valid number for quantity.");
        }
    }

    private void editCartItem() {
        int selectedIndex = cartList.getSelectedIndex();
        if (selectedIndex == -1) {
            showMessage("Error", "Please select an item from the cart to edit.");
            return;
        }

        CartItem cartItem = cart.get(selectedIndex);
        String newQuantityString = JOptionPane.showInputDialog("Enter new
quantity for " + cartItem.foodItem.name + ": ");
        try {
            int newQuantity = Integer.parseInt(newQuantityString);
            if (newQuantity > 0) {
                cartItem.quantity = newQuantity;
```

```
            updateCart();
        } else {
            showMessage("Error", "Invalid quantity.");
        }
    } catch (NumberFormatException e) {
        showMessage("Error", "Please enter a valid number for quantity.");
    }
}

private void deleteCartItem() {
    int selectedIndex = cartList.getSelectedIndex();
    if (selectedIndex != -1) {
        cart.remove(selectedIndex);
        updateCart();
    }
}

private void updateCart() {
    cartListModel.clear();
    double totalAmount = 0;

    for (CartItem cartItem : cart) {
        cartListModel.addElement(cartItem.toString());
        totalAmount += cartItem.getTotalPrice();
    }

    totalAmountLabel.setText("Total Amount: ₹" + totalAmount);
```

```
    }

    private void placeOrder() {
        if (cart.isEmpty()) {
            showMessage("Error", "Your cart is empty.");
            return;
        }

        StringBuilder orderSummary = new StringBuilder("Your Order Summary:\n");
        double totalAmount = 0;
        for (CartItem cartItem : cart) {
            orderSummary.append(cartItem.toString()).append("\n");
            totalAmount += cartItem.getTotalPrice();
        }
        orderSummary.append("Total Amount: ₹").append(totalAmount);

        // Show message that the order has been placed
        JOptionPane.showMessageDialog(this, orderSummary.toString() + "\n\nYour order has been placed.", "Order Placed", JOptionPane.INFORMATION_MESSAGE);
        cart.clear();  // Clear cart after placing the order
        updateCart();
    }

    private void showMessage(String title, String message) {
        JOptionPane.showMessageDialog(this, message, title, JOptionPane.INFORMATION_MESSAGE);
    }
```

```
    public static void main(String[] args) {
        try {
            // Establish the database connection
            connection =
DriverManager.getConnection("jdbc:mysql://localhost/fooddelivery", "root",
"Dharani@2006");


            // Show the Sign-Up page initially
            SwingUtilities.invokeLater(() -> new
FoodDeliveryAppSwing().setVisible(true));
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Backend (SQL) code

```
CREATE DATABASE fooddelivery;
```

```
USE fooddelivery;

CREATE TABLE menu (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    price DOUBLE NOT NULL
);

CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    address VARCHAR(255) NOT NULL,
    total DOUBLE NOT NULL,
    order_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE order_details (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    food_id INT NOT NULL,
    quantity INT NOT NULL,
    total_price DOUBLE NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (food_id) REFERENCES menu(id)
);
```

```
INSERT INTO menu (name, price) VALUES
('Burger', 60.00),
('Pizza', 120.00),
('Pasta', 40.00),
('Salad', 55.00),
('Soda', 30.00);

select * from orders;
ALTER TABLE order_details ADD COLUMN price DOUBLE;
ALTER TABLE order_details MODIFY total_price DOUBLE NULL;
describe order_details;
ALTER TABLE order_details DROP COLUMN price;

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    name VARCHAR(255),
    email VARCHAR(255),
    address VARCHAR(255) NOT NULL
);
```