**Online Shopping Cart System**

**Project Based Learning (PBL) Report**

**For the course**

**Object Oriented Programming-20CS21002**

**Database Management System-20CS21003**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**By**

A.MEENAKSHI – 24R15A0501

G.VARSHA – 23R11A0518

K.ASHWITHA – 23R11A0521

**under the guidance of**

G Krishna Lava Kumar

D Jeevan

****

**Department of Computer Science and Engineering**

**Accredited by NBA**

**Geetanjali College of Engineering and Technology**

**(UGC Autonomous)**

(Affiliated to J.N.T.U, Approved by AICTE, New Delhi)

Cheeryal (V), Keesara (M), Medchal.Dist- 501 301

**December-2024**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Contents** | **Page No** |
| **1** | **Introduction** | **3** |
| **2** | **System Design** | **5** |
| **3** | **Implementation** | **7** |
| **4** | **Conclusion** | **34** |
| **5** | **References** | **35** |

**Introduction**

**About the Project**

This project is an online shopping cart application designed to provide users with an easy and efficient way to browse and purchase products. The system is built with a clear user interface using Java Swing and a MySQL database to manage data. It consists of multiple components such as product management, cart functionality, billing, and order management. Users can browse available products, select items to add to their cart, and review their selections in the cart frame. When ready to checkout, users can proceed to the billing frame to finalize their purchase. The orders, along with the corresponding details, are saved in the database for tracking and reference. The system aims to deliver a seamless shopping experience, allowing users to interact with various frames such as login, sign-in, add to cart, cart, billing, and orders with ease.

**Project Outcomes and Objectives**

* **Objectives**:
  + **User-Friendly Interface**: Provide a simple and intuitive interface using Java Swing for easy navigation between different frames.
  + **Product Selection**: Allow users to browse available products and select items to add to their shopping cart.
  + **Cart Management**: Enable users to view, modify, and manage the contents of their shopping cart.
  + **Billing Process**: Implement a billing frame for users to finalize their purchases and enter payment information.
  + **Order Management**: Track user orders by storing order details in the database for future reference.
  + **Secure User Authentication**: Ensure secure login and sign-in processes for users to access their accounts.
  + **Database Integration**: Use MySQL for efficient data management of products, users, orders, and billing information.
  + **Seamless Checkout**: Provide a smooth checkout process from adding products to the cart to completing the order.
* **Outcomes**:
  + **Operational Online Shopping Cart**: A fully functional online shopping cart system that allows users to browse products, add them to the cart, and complete purchases with ease.Integration with a database backend for persistent data storage.
  + **Efficient Product and Order Management**: Seamless management of product listings, user orders, and billing information, with all data securely stored in the MySQL database.
  + **Improved User Experience**: A user-friendly interface with smooth navigation, enabling customers to easily interact with different frames like login, cart, and billing.
  + **Secure Transactions**: A secure system for user authentication and order processing, ensuring privacy and protection of user data.
  + **Database-Driven Application**: Integration of MySQL for managing product data, user information, order details, and billing, leading to better organization and retrieval of information.
  + **Scalable System**: A flexible and scalable architecture that can be extended with additional features like product search, payment gateways, or user reviews.
  + **Real-Time Order Tracking**: Users can track their orders and manage them through the system, ensuring better customer service and satisfaction.

**System Design**

**System Architecture**

The system follows a 3-layered architecture:

1. Presentation Layer:
   * Implements the GUI using Swing components for user interaction.
   * Provides various frames (login, sign-in, cart, add to cart, billing, orders) for user interaction.
   * Handles user inputs, displays product listings, manages the shopping cart, and allows order processing.
2. Business Logic Layer:
   * Contains services that handle the core functionalities of the shopping cart system:
   * **Product Service**: Manages product selection, display, and updates.
   * **Cart Service**: Handles adding/removing products to/from the cart, as well as calculating the total price.
   * **Billing Service**: Processes the billing information, including applying discounts or taxes.
   * **Order Service**: Manages order creation, saving order details to the database, and tracking the status of orders.
3. Data Layer:
   * Connects to a **MySQL** database using **JDBC** for persistent storage.
   * Manages tables for **product**, **billing**, **orders**, **order\_details**, **users**, and **addToCart**.
   * Responsible for retrieving and storing data such as product information, user details, cart items, order history, and billing data.

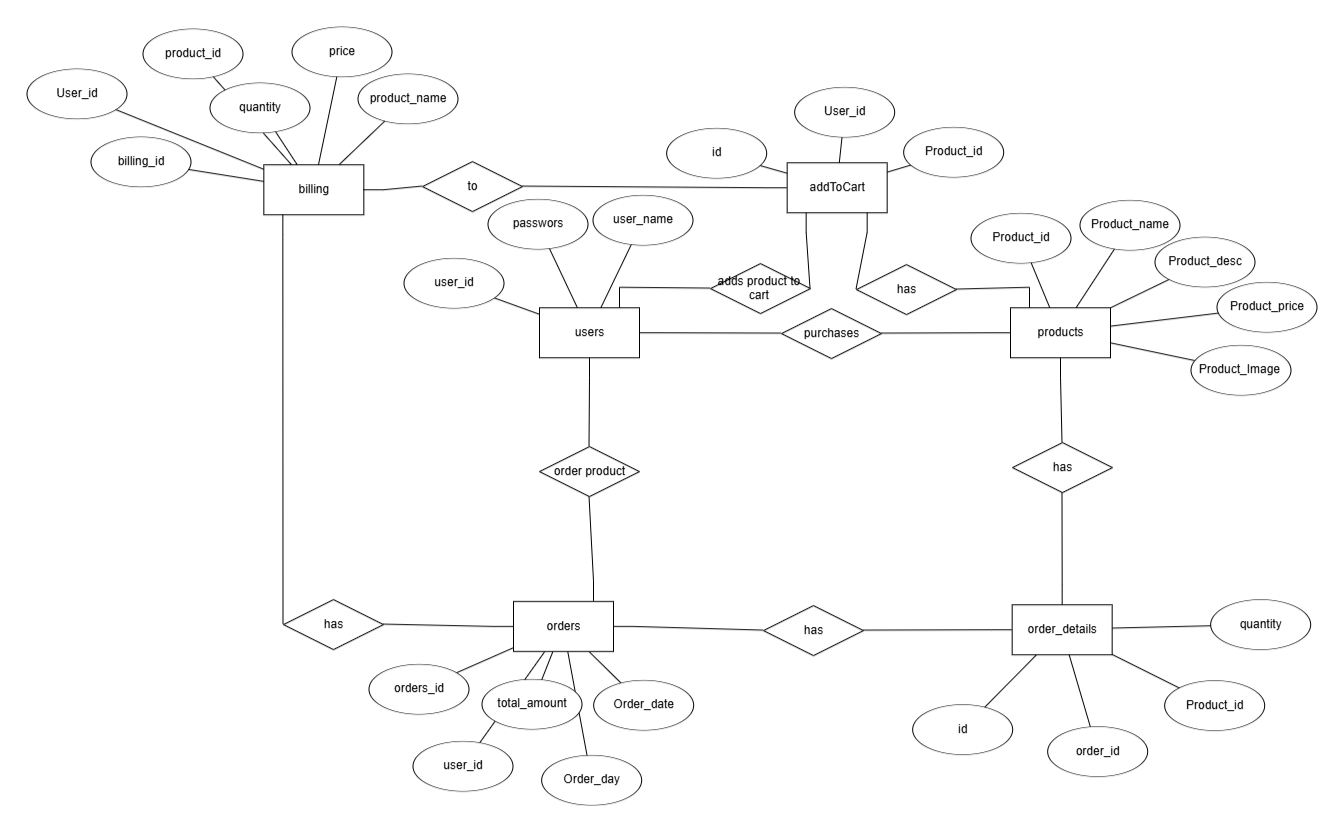
**Modules**

1. User Authentication Module:
   * Functionality: Manage user login and registration details.
   * Details: Validate username and password for login; allow retry for incorrect attempts.
2. Product Management Module:
   * Functionality: Display products across categories such as Electronics, Clothing, Jewellery, etc.
   * Details: Organize products into categories; enable selection and viewing of product details.
3. Cart Management Module:
   * Functionality: Add, update, delete, and view items in the cart.
   * Details: Store items added by the user, including quantity and price; calculate the total cart value.
4. Checkout Module:
   * Functionality: Process payments and finalize purchases.
   * Details: Display total cost; prompt for payment confirmation; generate a success message.

### ****Backend Design****

* **Database:**
  + **Tables:**
    - **users**: (username, password, email).
    - **products**: (product\_id, name, category, price, stock).
    - **cart**: (cart\_id, username, product\_id, quantity).
    - **orders**: (order\_id, username, total\_amount, order\_date).
    - **order\_details**: (order\_id, product\_id, quantity, price\_per\_unit).
  + **Relationships**:
    - **users (Primary Key: username)** links to **cart** and **orders**.
    - **products (Primary Key: product\_id)** links to **cart** and **order\_details**.
    - **orders (Primary Key: order\_id)** links to **order\_details**.

This structure ensures modular functionality with clear relationships among users, products, cart items, and orders.

****

**Implementation**

**Modules Implementation**

**Add to Cart Module:**

* + Accepts product selection and user details (user ID, product ID) through a dialog
  + Validates if the product is available and the user is logged in.
  + Adds the product to the cart for the user in the addtocart table.

**View Cart Module:**

* + Fetches all items from the user's cart by joining addtocart and products tables.
  + Displays selected products, quantity, price, and total amount in a table.

**Update Cart Module:**

* + Allows users to modify the quantity of items in their cart.
  + Updates the addtocart table with the new quantity and recalculates the total amount.

**Delete from Cart Module:**

* + Deletes a product from the user’s cart based on product ID.
  + Removes the corresponding record from the addtocart table

**Billing Module:**

1. Generate Bill:
   * Accepts user and cart details (products, quantity) from the cart.
   * Calculates total cost, taxes, and any discounts.
   * Stores billing information (user ID, product ID, quantity, price, total) in the billing table.

1. View Billing Details:
   * Displays the complete billing details (product names, quantities, prices, total amount) for the user.
   * Fetches data from the billing table.
2. Update Billing Information:
   * Allows users to modify their billing details before finalizing the order.
   * Updates the billing table with any changes to the quantities or prices.

### ****Order Module:****

1. **Place Order:**
   * Accepts user details and finalized billing data.
   * Creates a new order record in the orders table with the total amount, order date, and user ID.
   * Generates corresponding order details in the order\_details table, linking products with their quantities.
2. **View Orders:**
   * Displays all orders placed by a user, including order ID, total amount, order date, and order status.
   * Fetches data from the orders and order\_details tables.
3. **Update Order Status:**
   * Allows the admin or system to update the status of an order (e.g., "Processing", "Shipped", "Delivered").
   * Updates the orders table with the new status.
4. **Cancel Order:**
   * Allows users to cancel an order before it’s processed.
   * Removes the order from the orders and order\_details tables, or marks it as "Cancelled".

### ****Product Management Module:****

1. **Add Product:**
   * Admin inputs product details (name, description, price, image URL) through a form.
   * Validates the data and adds it to the products table.
2. **View Products:**
   * Displays all products in a list with details like product name, description, and price.
   * Allows users to browse and select products to add to their cart.
3. **Update Product:**
   * Allows the admin to modify existing product details (e.g., price, description).
   * Updates the products table with new information.
4. **Delete Product:**
   * Allows the admin to remove a product from the system.
   * Deletes the product from the products table.

### ****User Authentication Module:****

1. **Register User:**
   * Accepts user details (username, password) and stores them in the users table.
   * Validates input (e.g., checks if username is unique).
2. **Login User:**
   * Authenticates users based on their username and password.
   * Grants access to the main system if login is successful.
3. **Update User Profile:**
   * Allows users to update their personal information (e.g., password, email).
   * Updates the users table with new information.
4. **Delete User:**
   * Allows users to delete their account.
   * Removes the user record from the users table.

**Sample code**

**//contains only main frame**

public class CartMain1 extends JFrame {

private static final long serialVersionUID = 1L;

private JPanel contentPane;

private JPanel productPanel; // Panel for products

private List<Product> products;

private List<JCheckBox> checkboxes;

private CartFrame cartFrame;

private int userId;

private List<Product> cartItems = new ArrayList<>();

BillingFrame billingFrame = new BillingFrame(userId);

private static String currentUser;

//private JButton addToCart;

public static void main(String[] args) {

EventQueue.invokeLater(new Runnable() {

public void run() {

try {

CartMain1 frame = new CartMain1();

frame.setVisible(true);

} catch (Exception e) {

e.printStackTrace();

}

}

});

}

public CartMain1() {

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setBounds(100, 100, 900, 600);

contentPane = new JPanel();

contentPane.setBackground(new Color(0, 198, 198)); // Updated background color

contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

getContentPane().setLayout(new BorderLayout());

setContentPane(contentPane);

contentPane.setLayout(null);

// ADD TO CART button

JButton addToCart = new JButton("ADD TO CART");

addToCart.setBounds(692, 149, 150, 23);

contentPane.add(addToCart);

addToCart.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

addSelectedProductsToCart();

}

});

// BILL button

JButton billing = new JButton("BILL");

billing.setBounds(692, 234, 152, 23);

contentPane.add(billing);

billing.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (getCurrentUser() == null || getCurrentUser().trim().isEmpty()) {

// User is not logged in, prompt them to log in

int choice = JOptionPane.showConfirmDialog(

null,

"You are not logged in. Would you like to log in?",

"Login Required",

JOptionPane.YES\_NO\_OPTION

);

if (choice == JOptionPane.YES\_OPTION) {

loginUser(); // Call the login method

if (getCurrentUser() != null && !getCurrentUser().trim().isEmpty()) {

// Login successful, proceed with opening the BillingFrame

try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/shopping\_cart\_sys3", "root", "200601")) {

int userId = getUserId(getCurrentUser(), connection); // Fetch the user ID for the logged-in user

BillingFrame billingFrame = new BillingFrame(userId);

billingFrame.loadOrderDetails(); // Optionally load user order details

billingFrame.setVisible(true);

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "Failed to connect to the database.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

} else {

// User is already logged in, proceed with opening the BillingFrame

try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/shopping\_cart\_sys3", "root", "200601")) {

int userId = getUserId(getCurrentUser(), connection); // Fetch the user ID for the logged-in user

BillingFrame billingFrame = new BillingFrame(userId);

billingFrame.loadOrderDetails(); // Optionally load user order details

billingFrame.setVisible(true);

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "Failed to connect to the database.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

});

// ORDERS button

JButton ordersButton = new JButton("ORDERS");

ordersButton.setBounds(692, 324, 152, 23); // Adjust the position as needed

contentPane.add(ordersButton);

ordersButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if (getCurrentUser() == null || getCurrentUser().trim().isEmpty()) {

// User is not logged in, prompt them to log in

int choice = JOptionPane.showConfirmDialog(

null,

"You are not logged in. Would you like to log in?",

"Login Required",

JOptionPane.YES\_NO\_OPTION

);

if (choice == JOptionPane.YES\_OPTION) {

loginUser(); // Call the login method

if (getCurrentUser() != null && !getCurrentUser().trim().isEmpty()) {

// Login successful, proceed with opening the OrdersFrame

try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/shopping\_cart\_sys3", "root", "200601")) {

int userId = getUserId(getCurrentUser(), connection); // Fetch the user ID for the logged-in user

OrdersFrame ordersFrame = new OrdersFrame(userId);

ordersFrame.setVisible(true);

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "Failed to connect to the database.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

} else {

// User is already logged in, proceed with opening the OrdersFrame

try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/shopping\_cart\_sys3", "root", "200601")) {

int userId = getUserId(getCurrentUser(), connection); // Fetch the user ID for the logged-in user

OrdersFrame ordersFrame = new OrdersFrame(userId);

ordersFrame.setVisible(true);

} catch (SQLException ex) {

ex.printStackTrace();

JOptionPane.showMessageDialog(null, "Failed to connect to the database.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

}

});

// Product Panel

productPanel = new JPanel();

productPanel.setBackground(new java.awt.Color(255, 128, 128)); // Background color changed

JScrollPane scrollPane = new JScrollPane(productPanel);

scrollPane.setBounds(10, 52, 641, 511);

productPanel.setLayout(new GridLayout(1, 0, 0, 0));

scrollPane.setPreferredSize(new Dimension(600, 400)); // Adjust as needed

getContentPane().add(scrollPane, BorderLayout.CENTER);

contentPane.add(scrollPane);

//logout

JButton logout = new JButton("LOGOUT");

logout.setBounds(692, 415, 152, 23);

contentPane.add(logout);

logout.addActionListener(e -> logoutUser());

//tool bar

JToolBar toolBar = new JToolBar();

toolBar.setBounds(10, 11, 841, 30);

toolBar.setRollover(true);

toolBar.setBackground(new Color(255, 128, 128)); // Toolbar color

contentPane.add(toolBar);

JLabel lblNewLabel = new JLabel("MINNY CART ");

lblNewLabel.setFont(new Font("Times New Roman", Font.BOLD, 32));

lblNewLabel.setHorizontalAlignment(SwingConstants.CENTER);

toolBar.add(lblNewLabel);

// SIGN IN button

JButton signIn = new JButton("SIGN IN");

signIn.setBackground(new Color(255, 128, 128)); // Updated color

toolBar.add(signIn);

signIn.setFont(new Font("Tahoma", Font.BOLD, 11));

signIn.addActionListener(e -> signInUser());

// LOGIN button

JButton login = new JButton("LOGIN");

login.setBackground(new Color(255, 128, 128)); // Updated color

toolBar.add(login);

login.setFont(new Font("Tahoma", Font.BOLD, 11));

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

JButton cartDetails = new JButton("CART");

cartDetails.setBackground(new Color(255, 128, 128)); // Updated color

cartDetails.setHorizontalAlignment(SwingConstants.LEADING);

toolBar.add(cartDetails);

cartDetails.setFont(new Font("Tahoma", Font.BOLD, 11));

cartDetails.addActionListener(e -> viewCart());

loadProducts();

}

private void loadProducts() {

products = fetchProductsFromDatabase(); // Fetch products from DB

checkboxes = new ArrayList<>();

GridBagConstraints gbc = new GridBagConstraints();

gbc.insets = new Insets(10, 10, 10, 10); // Spacing between items

gbc.gridx = 0;

gbc.gridy = 0;

gbc.fill = GridBagConstraints.HORIZONTAL; // Allow components to stretch horizontally

for (Product product : products) {

JPanel productItem = new JPanel();

productItem.setLayout(new GridBagLayout());

productItem.setBackground(new Color(245, 245, 220)); // Added background color

productItem.setBorder(new LineBorder(Color.GRAY, 1)); // Added border

GridBagConstraints itemGbc = new GridBagConstraints();

itemGbc.insets = new Insets(15, 15, 15, 15);

itemGbc.gridx = 0;

itemGbc.gridy = 0;

itemGbc.gridwidth = 2; // Span across multiple columns for the image

itemGbc.fill = GridBagConstraints.HORIZONTAL; // Allow the image to stretch horizontally

// Add product image with border

try {

File file = new File(product.getImageUrl());

if (file.exists()) {

ImageIcon imageIcon = new ImageIcon(product.getImageUrl());

java.awt.Image img = imageIcon.getImage();

java.awt.Image scaledImg = img.getScaledInstance(200, 200, java.awt.Image.SCALE\_SMOOTH); // Adjust the size here

JLabel imageLabel = new JLabel(new ImageIcon(scaledImg));

imageLabel.setBorder(new LineBorder(Color.DARK\_GRAY, 2)); // Border for the image

productItem.add(imageLabel, itemGbc);

} else {

JLabel errorLabel = new JLabel("Image not found");

productItem.add(errorLabel, itemGbc);

}

} catch (Exception e) {

e.printStackTrace();

JLabel errorLabel = new JLabel("Image load error");

productItem.add(errorLabel, itemGbc);

}

// Add product name with increased font size

itemGbc.gridy++; // Move to next row

itemGbc.gridwidth = 1; // Reset gridwidth

JLabel nameLabel = new JLabel(product.getName());

nameLabel.setFont(new Font("Arial", Font.BOLD, 16)); // Increased font size

productItem.add(nameLabel, itemGbc);

// Add product price with increased font size

itemGbc.gridy++; // Move to next row

JLabel priceLabel = new JLabel("Price: RS" + product.getPrice());

priceLabel.setFont(new Font("Arial", Font.PLAIN, 14)); // Increased font size

productItem.add(priceLabel, itemGbc);

// Add product description with increased font size

itemGbc.gridy++; // Move to next row

JLabel descriptionLabel = new JLabel("<html>" + product.getDescription() + "</html>"); // For wrapping long description text

descriptionLabel.setFont(new Font("Arial", Font.ITALIC, 12)); // Increased font size

productItem.add(descriptionLabel, itemGbc);

// Add product checkbox

itemGbc.gridy++; // Move to next row

JCheckBox checkBox = new JCheckBox();

checkboxes.add(checkBox);

productItem.add(checkBox, itemGbc);

// Add productItem to productPanel

productPanel.add(productItem, gbc);

gbc.gridy++; // Move to the next row in the main panel

}

productPanel.revalidate(); // Refresh UI

productPanel.repaint();

}

// Dynamically adjust columns when the window is resized

@Override

public void setSize(int width, int height) {

super.setSize(width, height);

// Recalculate the columns based on the new window size

loadProducts(); // Reload products with updated layout

}

private void viewCart() {

if (getCurrentUser() == null || getCurrentUser().isEmpty()) {

JOptionPane.showMessageDialog(this, "You must be logged in to view the cart.", "Login Required", JOptionPane.WARNING\_MESSAGE);

return;

}

// Fetch cart items from the database and assign it to the instance-level cartItems

cartItems = fetchCartItemsForUser(getCurrentUser()); // Assign to the instance variable

if (cartItems.isEmpty()) {

JOptionPane.showMessageDialog(this, "Your cart is empty.", "No Items in Cart", JOptionPane.INFORMATION\_MESSAGE);

} else {

if (this.cartFrame != null) {

this.cartFrame.dispose(); // Dispose existing cartFrame to avoid multiple instances

}

this.cartFrame =new CartFrame(cartItems, billingFrame);

// Assign to the instance variable

this.cartFrame.setVisible(true);

}

}

private List<Product> fetchProductsFromDatabase() {

List<Product> products = new ArrayList<>();

try {

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/shopping\_cart\_sys3", "root", "200601");

String query = "SELECT \* FROM products";

PreparedStatement statement = connection.prepareStatement(query);

ResultSet resultSet = statement.executeQuery();

while (resultSet.next()) {

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

String name = resultSet.getString("name");

String description = resultSet.getString("description");

String imageUrl = resultSet.getString("image\_url");

double price = resultSet.getDouble("price");

products.add(new Product(id, name, description, imageUrl, price));

}

resultSet.close();

statement.close();

connection.close();

} catch (Exception e) {

e.printStackTrace();

}

return products;

}

private void logoutUser() {

if (getCurrentUser() != null) {

String username = getCurrentUser();

setCurrentUser(null);

resetCartAndUI();

// Show a logout success message

JOptionPane.showMessageDialog(this, "Successfully logged out from account: " + username,

"Logout Successful", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(this, "No user is currently logged in.",

"Logout Failed", JOptionPane.ERROR\_MESSAGE);

}

}

private void resetCartAndUI() {

// Clear the cart items

if (cartItems != null) {

cartItems.clear(); // Clear the cart list

}

// Debug: Check cartFrame status

if (cartFrame != null) {

System.out.println("Disposing of cart frame...");

cartFrame.dispose(); // Close the cart window

cartFrame = null; // Remove the reference to avoid memory leaks

} else {

System.out.println("Cart frame is already null.");

}

// Reset all product checkboxes

if (checkboxes != null) {

for (JCheckBox checkbox : checkboxes) {

checkbox.setSelected(false); // Uncheck each checkbox

}

}

// Notify user

JOptionPane.showMessageDialog(this, "Cart and UI have been reset successfully.", "Reset Complete", JOptionPane.INFORMATION\_MESSAGE);

}

private void signInUser() {

String username = JOptionPane.showInputDialog(this, "Enter New Username:", "Sign In", JOptionPane.PLAIN\_MESSAGE);

String password = JOptionPane.showInputDialog(this, "Enter New Password:", "Sign In", JOptionPane.PLAIN\_MESSAGE);

if (username == null || username.trim().isEmpty() || password == null || password.trim().isEmpty()) {

JOptionPane.showMessageDialog(this, "Username and password cannot be empty!", "Sign Up Failed", JOptionPane.ERROR\_MESSAGE);

return;

}

// Call signUpUser to add the new user to the database

if (signUpUser(username, password)) {

JOptionPane.showMessageDialog(this, "Sign Up Successful!", "Sign In", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(this, "Username already exists. Please choose another one.", "Sign In", JOptionPane.ERROR\_MESSAGE);

}

}

private boolean signUpUser(String username, String password) {

// Validate input (username and password should not be empty)

if (username == null || username.trim().isEmpty() || password == null || password.trim().isEmpty()) {

JOptionPane.showMessageDialog(this, "Username and password cannot be empty!", "Sign Up Failed", JOptionPane.ERROR\_MESSAGE);

return false;

}

try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/shopping\_cart\_sys3", "root", "200601")) {

// Check if the username already exists

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

PreparedStatement checkStatement = connection.prepareStatement(checkQuery);

checkStatement.setString(1, username);

ResultSet resultSet = checkStatement.executeQuery();

if (resultSet.next()) {

// Username already exists

JOptionPane.showMessageDialog(this, "Username already exists. Please choose another one.", "Sign Up Failed", JOptionPane.ERROR\_MESSAGE);

resultSet.close();

return false;

}

// Username is unique, proceed with insertion

String insertQuery = "INSERT INTO users (username, password) VALUES (?, ?)";

PreparedStatement insertStatement = connection.prepareStatement(insertQuery);

insertStatement.setString(1, username);

insertStatement.setString(2, password);

int rowsAffected = insertStatement.executeUpdate();

insertStatement.close();

if (rowsAffected > 0) {

// Sign up successful

return true;

} else {

// Insert failed

JOptionPane.showMessageDialog(this, "Sign up failed. Please try again.", "Sign Up Failed", JOptionPane.ERROR\_MESSAGE);

return false;

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error occurred while connecting to the database.", "Database Error", JOptionPane.ERROR\_MESSAGE);

return false;

}

}

private void addSelectedProductsToCart() {

// Check if the user is logged in

if (getCurrentUser() == null || getCurrentUser().isEmpty()) {

int response = JOptionPane.showConfirmDialog(this, "You must be logged in to add items to the cart. Do you want to log in?", "Login Required", JOptionPane.YES\_NO\_OPTION);

if (response == JOptionPane.YES\_OPTION) {

loginUser();

} else {

return; // Stop if user chooses not to log in

}

}

// List to hold added products

List<Product> addedProducts = new ArrayList<>();

try {

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/shopping\_cart\_sys3", "root", "200601");

int userId = getUserId(getCurrentUser(), connection);

if (userId == -1) {

JOptionPane.showMessageDialog(this, "User not found!", "Error", JOptionPane.ERROR\_MESSAGE);

return;

}

String checkQuery = "SELECT \* FROM addToCart WHERE user\_id = ? AND product\_id = ?";

PreparedStatement checkStatement = connection.prepareStatement(checkQuery);

String insertQuery = "INSERT INTO addToCart (user\_id, product\_id) VALUES (?, ?)";

PreparedStatement insertStatement = connection.prepareStatement(insertQuery);

boolean itemAdded = false;

for (int i = 0; i < checkboxes.size(); i++) {

if (checkboxes.get(i).isSelected()) {

Product selectedProduct = products.get(i);

checkStatement.setInt(1, userId);

checkStatement.setInt(2, selectedProduct.getId());

ResultSet resultSet = checkStatement.executeQuery();

if (!resultSet.next()) {

insertStatement.setInt(1, userId);

insertStatement.setInt(2, selectedProduct.getId());

insertStatement.executeUpdate();

addedProducts.add(selectedProduct); // Add the product to the cart list

itemAdded = true;

// Show a dialog box with the product name and price

JOptionPane.showMessageDialog(this,

"Item Added: " + selectedProduct.getName() + "\nPrice: " + selectedProduct.getPrice(),

"Item Added to Cart",

JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(this, selectedProduct.getName() + " is already in the cart.", "Duplicate Item", JOptionPane.WARNING\_MESSAGE);

}

resultSet.close();

}

}

if (itemAdded) {

JOptionPane.showMessageDialog(this, "Products added to cart successfully!", "Cart Updated", JOptionPane.INFORMATION\_MESSAGE);

} else {

JOptionPane.showMessageDialog(this, "No new products were added to the cart.", "No Changes", JOptionPane.WARNING\_MESSAGE);

}

checkStatement.close();

insertStatement.close();

connection.close();

resetCheckboxes();

} catch (Exception e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Error adding products to cart.", "Error", JOptionPane.ERROR\_MESSAGE);

}

}

private void loginUser() {

// Check if the user is already logged in

if (getCurrentUser() != null && !getCurrentUser().trim().isEmpty()) {

JOptionPane.showMessageDialog(this, "You are already logged in as " + getCurrentUser() + ". Please log out first if you want to log in with a different account.", "Already Logged In", JOptionPane.INFORMATION\_MESSAGE);

return; // Exit the method if the user is already logged in

}

String username = JOptionPane.showInputDialog(this, "Enter Username:", "Login", JOptionPane.PLAIN\_MESSAGE);

String password = JOptionPane.showInputDialog(this, "Enter Password:", "Login", JOptionPane.PLAIN\_MESSAGE);

// Check if the username or password is empty

if (username == null || username.trim().isEmpty() || password == null || password.trim().isEmpty()) {

JOptionPane.showMessageDialog(this, "Login Failed: Username and Password cannot be empty.", "Login Failed", JOptionPane.ERROR\_MESSAGE);

return; // Exit the method if the input is invalid

}

// Authenticate user

if (authenticateUser(username, password)) {

setCurrentUser(username);

JOptionPane.showMessageDialog(this, "Login Successful!", "Success", JOptionPane.INFORMATION\_MESSAGE);

// Get userId from the authenticated username

try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/shopping\_cart\_sys3", "root", "200601")) {

int userId = getUserId(username, connection); // Get userId using the username

if (userId == -1) {

JOptionPane.showMessageDialog(this, "Error: User not found.", "Error", JOptionPane.ERROR\_MESSAGE);

} else {

// Optionally store the userId for later use

// You can open the BillingFrame at a different point in the app

// BillingFrame billingFrame = new BillingFrame(userId);

// billingFrame.setVisible(true);

}

} catch (SQLException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(this, "Database connection error.", "Error", JOptionPane.ERROR\_MESSAGE);

}

} else {

JOptionPane.showMessageDialog(this, "Invalid credentials. Please try again.", "Login Failed", JOptionPane.ERROR\_MESSAGE);

}

}

// Example of your authenticateUser method, which should query the database for validation

private boolean authenticateUser(String username, String password) {

try {

// Here we will check the credentials in the users database

Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/shopping\_cart\_sys3", "root", "200601");

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

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, username);

statement.setString(2, password);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

resultSet.close();

statement.close();

connection.close();

return true; // User authenticated

} else {

resultSet.close();

statement.close();

connection.close();

return false; // Invalid credentials

}

} catch (SQLException e) {

e.printStackTrace();

return false; // Error in database connection or query

}

}

// Helper method to get user\_id based on the username

private int getUserId(String username, Connection connection) {

try {

String query = "SELECT id FROM users WHERE username = ?";

PreparedStatement statement = connection.prepareStatement(query);

statement.setString(1, username);

ResultSet resultSet = statement.executeQuery();

if (resultSet.next()) {

int userId = resultSet.getInt("id");

resultSet.close();

statement.close();

return userId;

}

resultSet.close();

statement.close();

return -1; // User not found

} catch (Exception e) {

e.printStackTrace();

}

return -1; // In case of error

}

// Method to reset the checkboxes after adding products to cart

private void resetCheckboxes() {

for (JCheckBox checkbox : checkboxes) {

checkbox.setSelected(false); // Uncheck all checkboxes

}

}

private List<Product> fetchCartItemsForUser(String username) {

List<Product> cartItems = new ArrayList<>();

try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/shopping\_cart\_sys3", "root", "200601")) {

String userQuery = "SELECT id FROM users WHERE username = ?";

PreparedStatement userStmt = connection.prepareStatement(userQuery);

userStmt.setString(1, username);

ResultSet userResult = userStmt.executeQuery();

if (userResult.next()) {

int userId = userResult.getInt("id");

String cartQuery = "SELECT p.\* FROM addToCart c JOIN products p ON c.product\_id = p.id WHERE c.user\_id = ?";

PreparedStatement cartStmt = connection.prepareStatement(cartQuery);

cartStmt.setInt(1, userId);

ResultSet cartResult = cartStmt.executeQuery();

while (cartResult.next()) {

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

String name = cartResult.getString("name");

String description = cartResult.getString("description");

String imageUrl = cartResult.getString("image\_url");

double price = cartResult.getDouble("price");

cartItems.add(new Product(id, name, description, imageUrl, price));

}

cartResult.close();

cartStmt.close();

}

userResult.close();

userStmt.close();

} catch (Exception e) {

e.printStackTrace();

}

return cartItems;

}

public static String getCurrentUser() {

return currentUser;

}

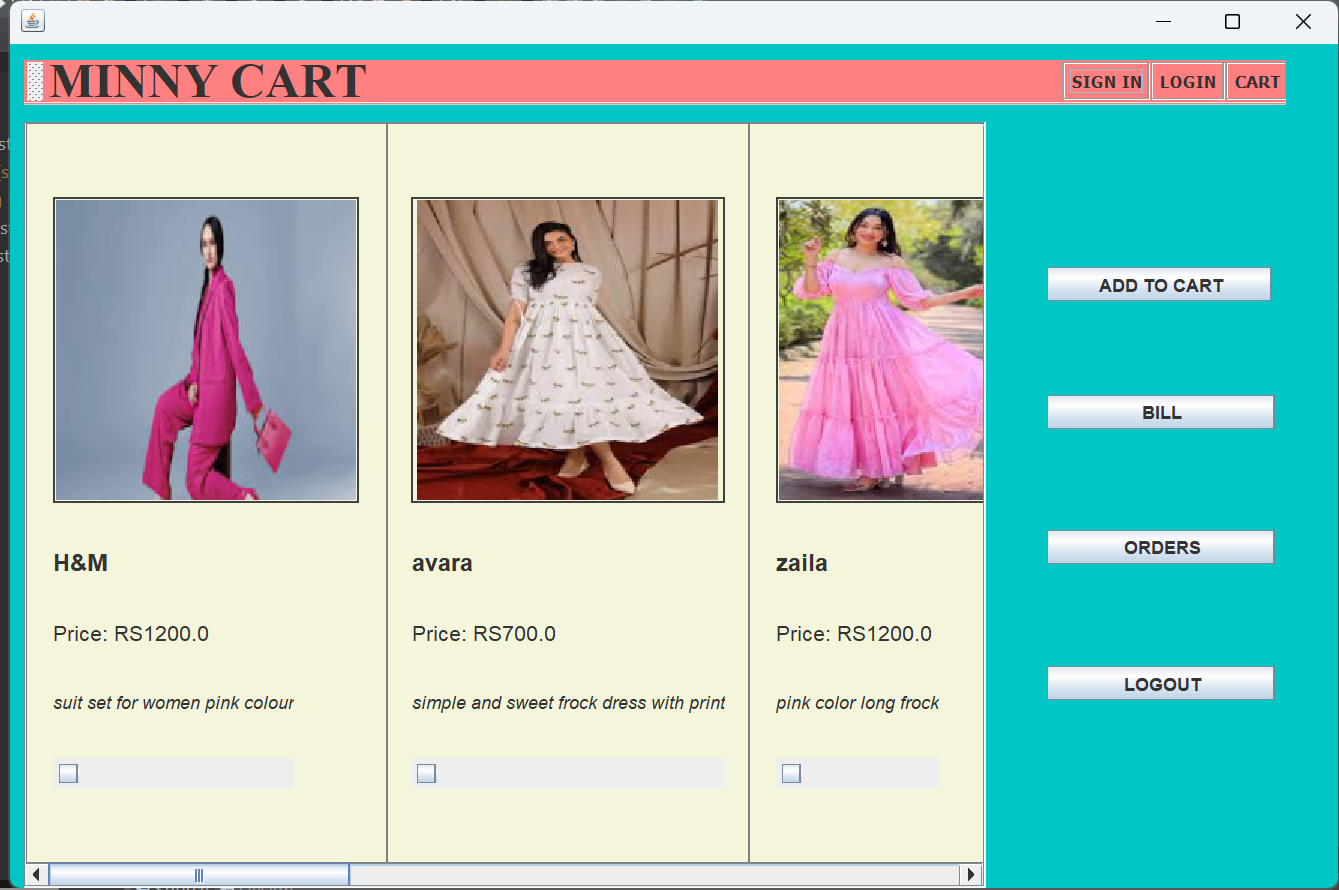
public void setCurrentUser(String currentUser) {

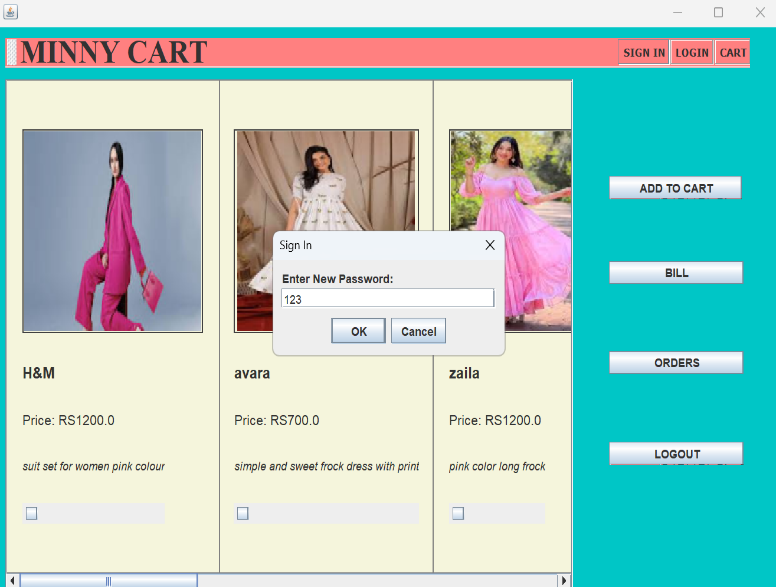
this.currentUser = currentUser;

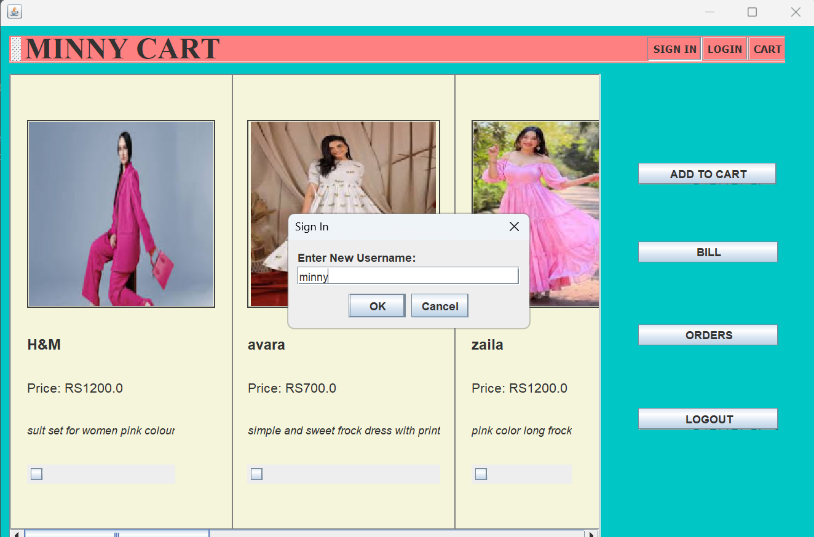
}

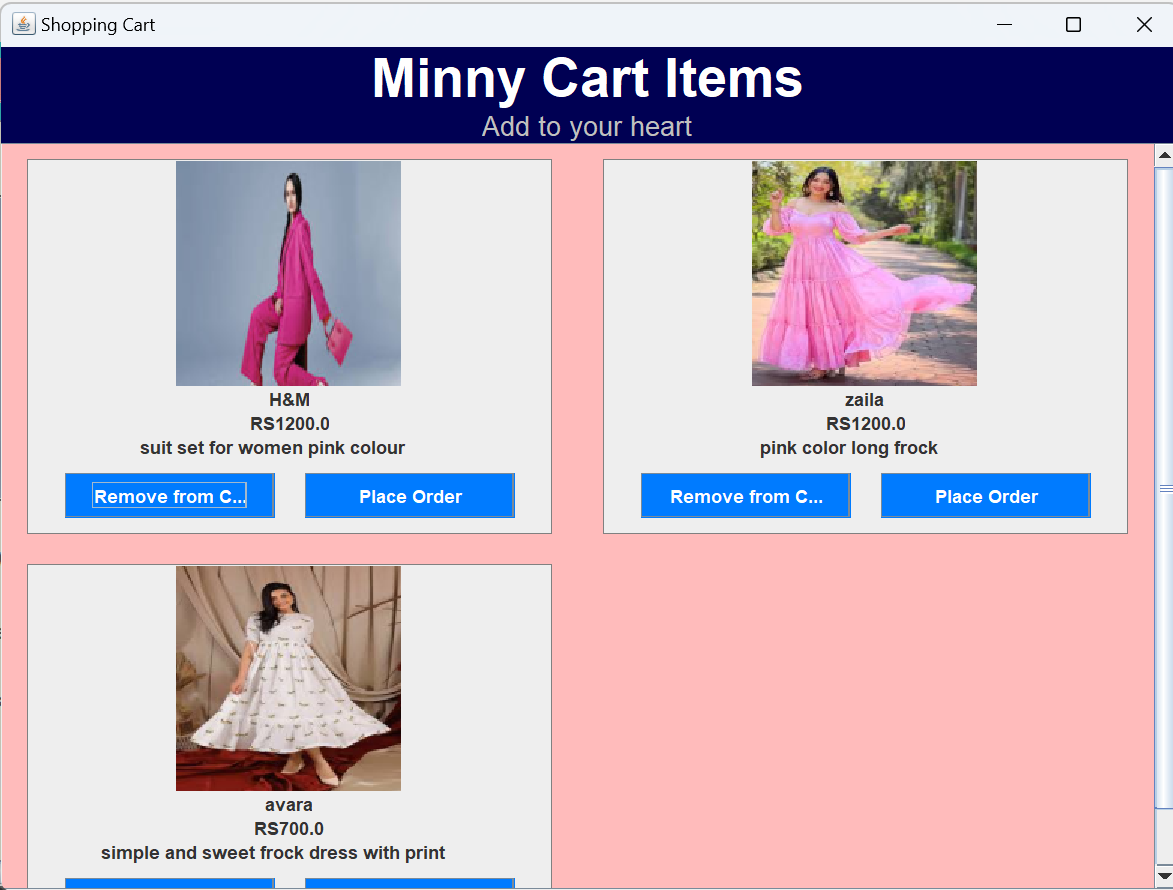
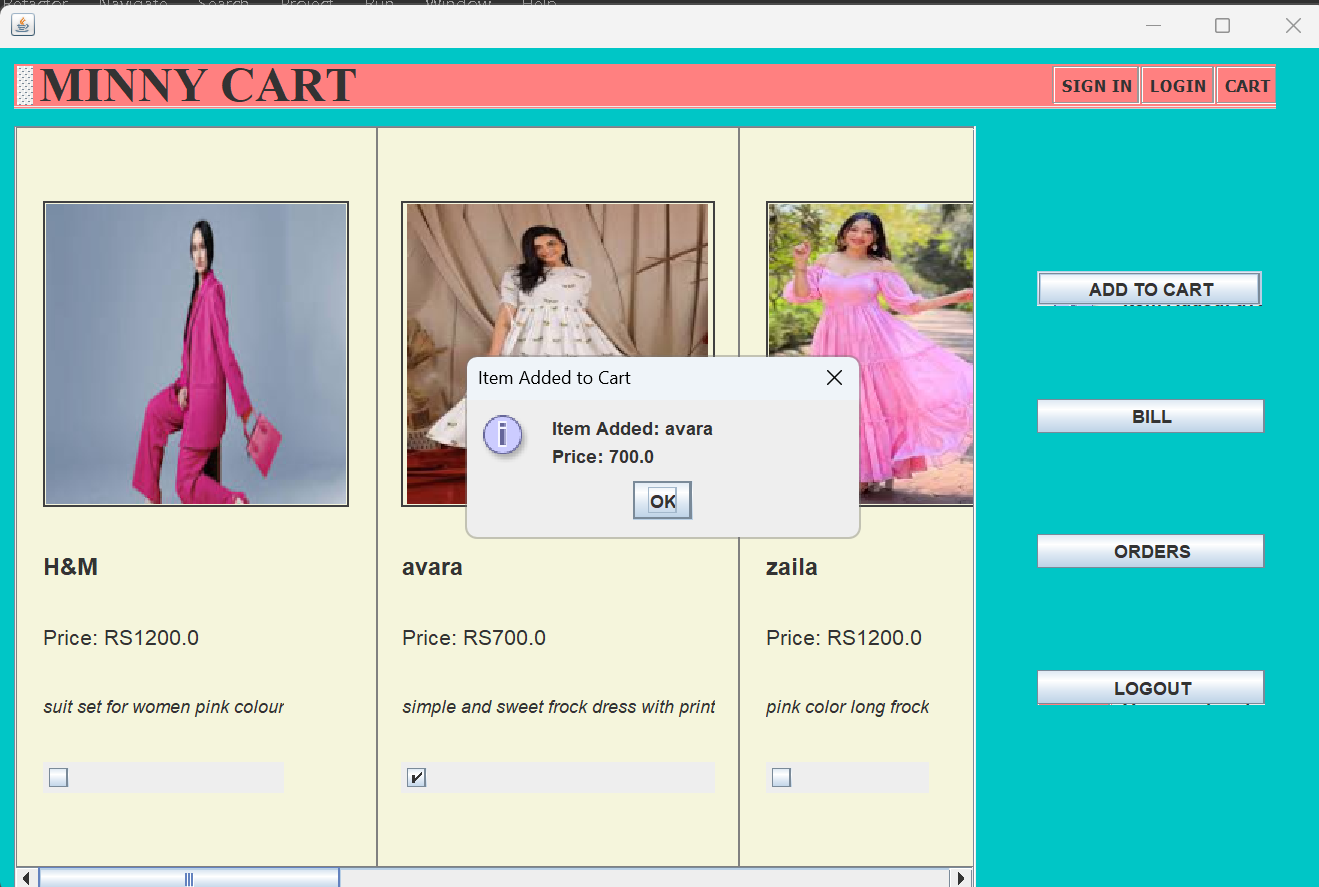
}

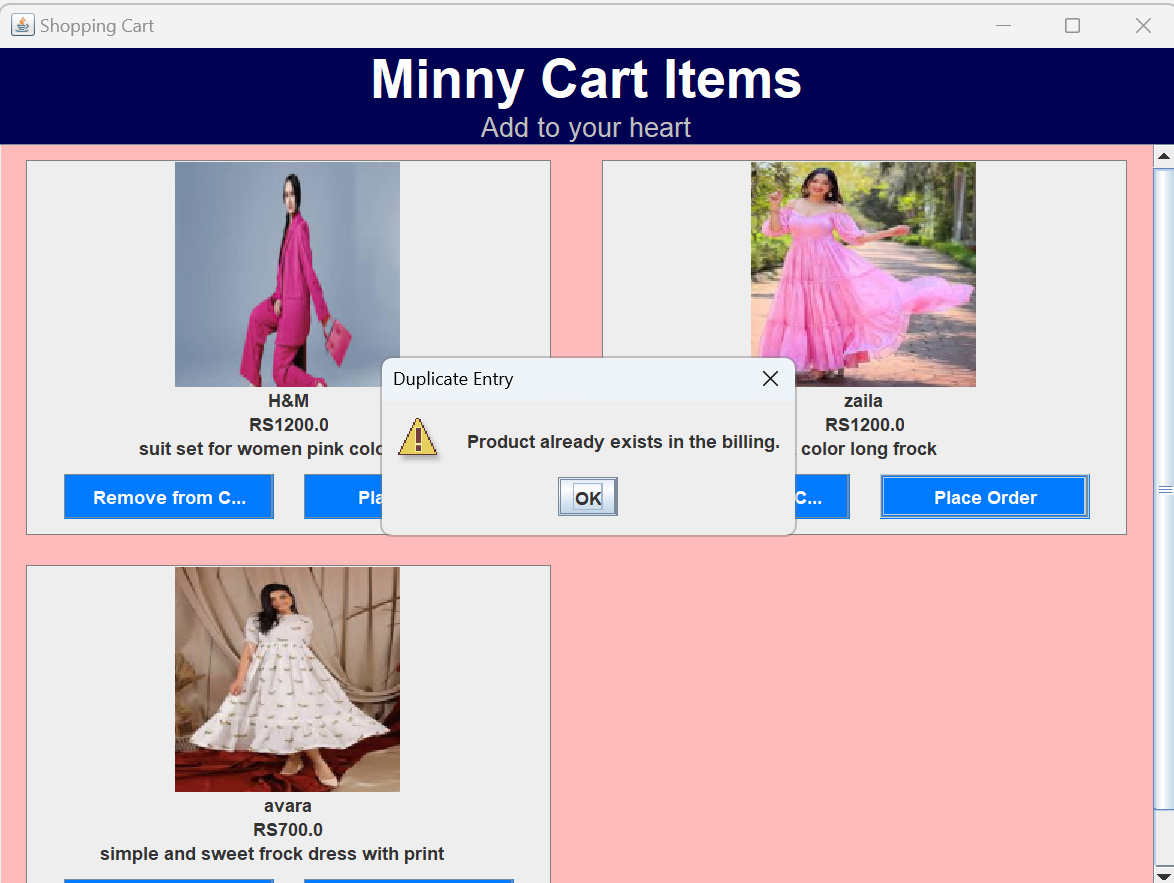
**Output**

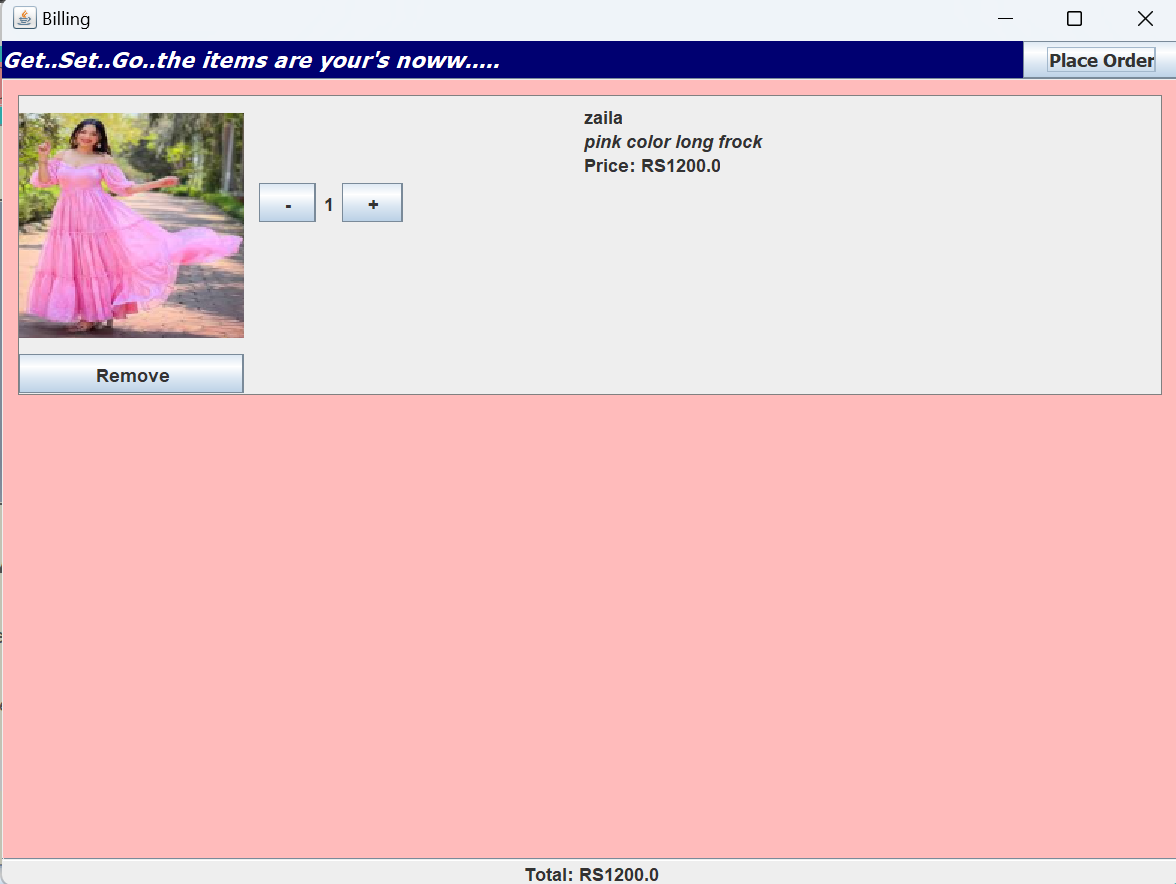


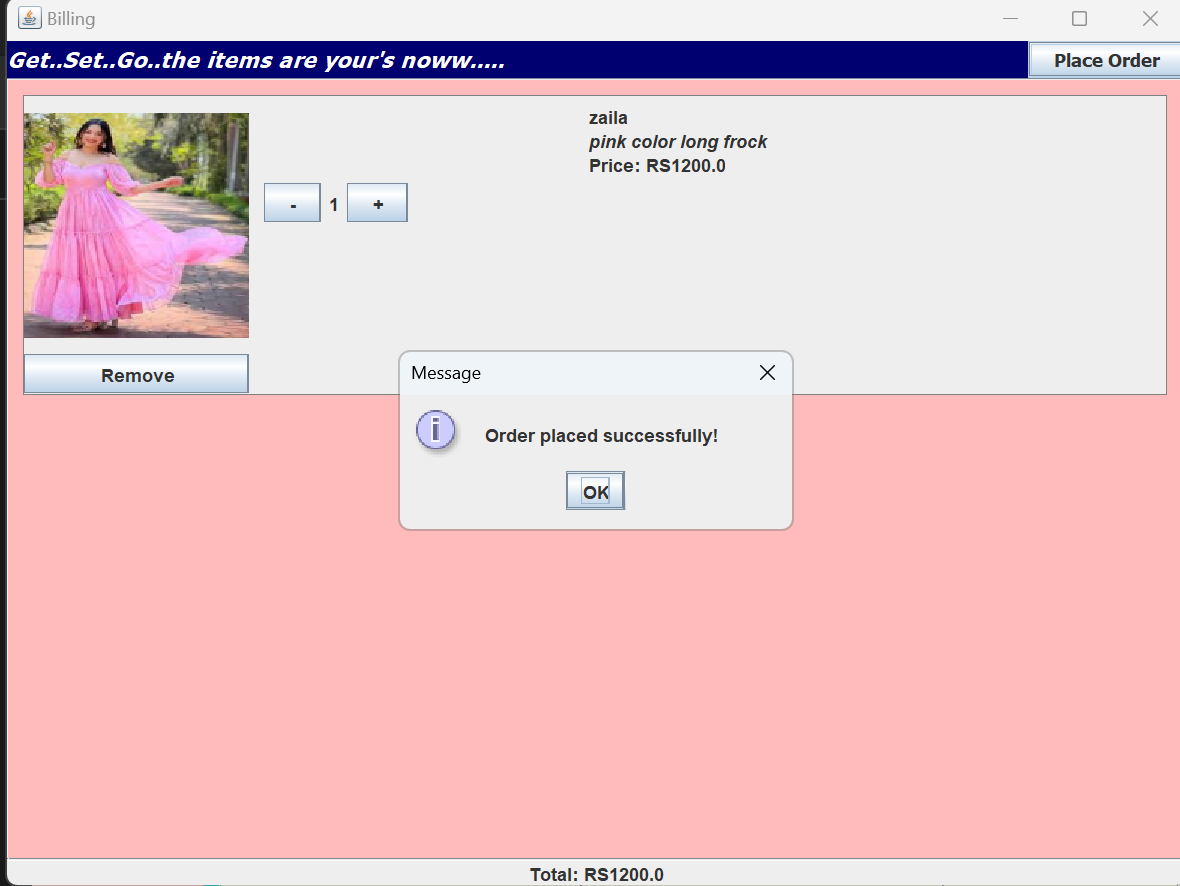














**Conclusion**

The online shopping cart system successfully provides a seamless and efficient shopping experience for users, with key features such as product browsing, cart management, billing, and order processing. The system is designed with a clear separation of concerns, using a 3-layered architecture that ensures smooth interaction between the user interface, business logic, and database layers.

Through the use of Java Swing for the front-end and MySQL for back-end data management, the application ensures fast performance, secure transactions, and reliable data storage. Users can easily manage their shopping cart, finalize orders, and keep track of their purchases, while administrators can efficiently manage products, users, and order details.

The system is scalable, allowing for future enhancements such as integrating payment gateways, adding product reviews, or improving search functionality. Overall, the project meets the core objectives of providing a functional, secure, and user-friendly online shopping platform.

**References**

<https://stackoverflow.com/questions/44971999/change-jframe-in-windowbuilder->

<https://www.geeksforgeeks.org/introduction-to-java-swing/>

eclipseTutorialsPoint - <https://www.tutorialspoint.com/>

OpenAI - <https://openai.com/>