



# Online Supermarket Web Application

PUSL2024-SOFTWARE ENGINEERING-2  
COURSEWORK FINAL DOCUMENT  
GROUP A49

SUBMISSION DATE: 31/12/2023  
UNIVERSITY OF PLYMOUTH

Name: Wedamulla Madinage Thisara Madusanka

Student Reference Number: 10899603

Module Code: PUSL2024

Module Name: Software Engineering-2

Coursework Title: Coursework Final Document

Deadline Date: 31<sup>st</sup> of December

Member of staff responsible for coursework: Mr. Chaminda Wijesinghe

Programme: BSc (Hons) Software Engineering

Please note that University Academic Regulations are available under Rules and Regulations on the University website [www.plymouth.ac.uk/studenthandbook](http://www.plymouth.ac.uk/studenthandbook).

Group work: please list all names of all participants formally associated with this work and state whether the work was undertaken alone or as part of a team. Please note you may be required to identify individual responsibility for component parts.

10899603 – Wedamulla Madinage Thisara Madusanka

10899621- Chathupraba Devindi Munasinghe

10899521 – Navindu Nimsara Gamage

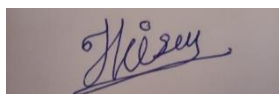
10899556- Yaddehi Kishal Sankalpa Jayalath

10899685- Kihaduwege Diduli Wijini Sahasra

10899600- Senanayake Dasili Liyanage Sameepa Pramuditha

***We confirm that we have read and understood the Plymouth University regulations relating to Assessment Offences and that we are aware of the possible penalties for any breach of these regulations. We confirm that this is the independent work of the group.***

Signed on behalf of the group:



Individual assignment: ***I confirm that I have read and understood the Plymouth University regulations relating to Assessment Offences and that I am aware of the possible penalties for any breach of these regulations. I confirm that this is my own independent work.***

Signed :

Use of translation software: failure to declare that translation software or a similar writing aid has been used will be treated as an assessment offence.

I \*have used/not used translation software.

If used, please state name of software.....

Overall mark \_\_\_\_\_% Assessors Initials \_\_\_\_\_ Date \_\_\_\_\_

## **Table of Contents**

Introduction.....	5
Identified Processes of the Scenario .....	6
Use Case Diagram.....	9
Class Diagram.....	10
Sequence Diagram .....	11
ER Diagram .....	12
Front-End Designs .....	13
registration.jsp.....	13
login.jsp:.....	14
index.jsp:.....	15
product.jsp:.....	16
about.jsp:.....	17
review.jsp:.....	18
cart.jsp:.....	19
billing.jsp: .....	20
Used Library Files : .....	21
Back-End Codes.....	22
Navigation Bar :.....	22
Footer: .....	22
index.jsp:.....	23
login.jsp:.....	24
registration.jsp:.....	25
product.jsp:.....	26
review.jsp: .....	27
about.jsp:.....	28
cart.jsp:.....	29
billing.jsp: .....	30
DbCon.Java Class: .....	31
ProductDao.Java Class:.....	31
UserDao.Java Class: .....	33
Login Servlet:.....	34
Registration Servlet:.....	35

Cart.Java Class:.....	36
Product.Java Class: .....	37
User.Java Class: .....	38
Login Servlet:.....	39
Logout Servlet:.....	40
Review Servlet: .....	41
AddToCart Servlet: .....	42
QuantityIncDec Servlet:.....	43
RemoveFromCart Servlet: .....	44
ProcessPaypalPayment Servlet: .....	45
References .....	47
Work-Load Matrix .....	48

## **Introduction**

Efficient and intuitive supermarket management systems are essential for improving the entire shopping experience in the quickly changing e-commerce sector. A complete web-based system known as the "Green Supermarket Management System" was created to safeguard transactions, expedite the shopping process, and collect insightful customer feedback. The NetBeans IDE's implementation of Java Server Pages (JSP) and Servlet technologies is used by this system to give users and administrators a responsive and easy-to-use platform.

Customers can purchase online with ease and efficiency thanks to this Green Supermarket Management System, which also gives administrators the tools they need to handle orders and inventory effectively. By incorporating user feedback, an ecosystem of responsive and customer-focused green supermarkets is created, guaranteeing a cycle of continual improvement. This initiative helps customers have a smooth and pleasurable online purchasing experience by bridging the gap between technology and business.

Outfitted with formidable instruments, administrators effectively oversee orders and inventories, guaranteeing optimal resource distribution and reducing wastage. A responsive ecosystem where users' requirements and preferences are actively taken into consideration is established by the system's integration of consumer input. An atmosphere where the green supermarket develops in step with consumer expectations and new technology is fostered by this iterative process of feedback and adaptation, which feeds into a cycle of continuous development.

The project is an example of how technology and business can be combined, showing how an online grocery store may go beyond its transactional function and become a force for good. Through its adoption of sustainability principles, the Green Supermarket Management System not only makes online shopping easy and enjoyable, but it also sparks a revolution in consumer behavior toward eco-friendly shopping. With this forward-thinking strategy, the platform is positioned as a leader in the field of green e-commerce, opening the door for a day when technology advances both commercial and environmental goals.

## **Identified Processes of the Scenario**

Several recognized procedures are involved in the Green Supermarket Management System in order to promote user involvement and efficient functioning. The main procedures seen in this situation are listed below:

### **1. Registering as a User:**

Description: In order to access customized features, new users must first register on the platform by giving the required information and creating an account.

Actions:

- Go to the page where registration is requested.
- Complete the registration form by entering your name, email, password, etc.
- Send in your registration information.
- Get a confirmation of your account.

### **2. User Sign-in:**

Description: Order tracking, transaction history maintenance, and customizable features are all available to registered users when they log in to their accounts.

Actions:

- Go to the login screen.
- Enter your password and registered email address.
- Verify the user's credentials.
- Get access to feature exclusive to your account.

### **3. Choosing Products and Organizing Carts:**

Description: Customers peruse the virtual grocery, choose what they want, and put it in their virtual shopping cart.

Actions:

- Check out the product catalogue.
- Choose items.
- Delete or add things to the cart.
- View and modify the items of the cart.

#### 4. User opinions and reviews:

Description: Using the special review page, users voice their opinions, recommendations, and grievances.

Actions:

- Go to the review page.
- Send comments to the administrators.
- Administrators examine user comments and take necessary actions.

#### 5. Order Placement

Description: Once on the order page, users may check the things they've chosen, pick a payment option, and finalize their purchase.

Actions:

- Go to the order page.
- Examine the contents of the cart.
- Verify the order and checkout.

#### 6. Constant Enhancement:

Administrators review user comments and adjust the system in response to recommendations and grievances.

Actions:

- Review user comments on a regular basis.
- Determine what needs to be improved.
- Apply upgrades and improvements to the system.

#### 7. Confirmation and Notification:

About new registrations and order submissions, alerts and confirmations are sent to both users and administrators.

Tasks:

- emails to users automatically confirming orders.
- Users' confirmation of registration notifications

#### 8. Processing Payments and Ensuring Security:

Description: During the checkout process, the system makes sure that payment information is handled securely.

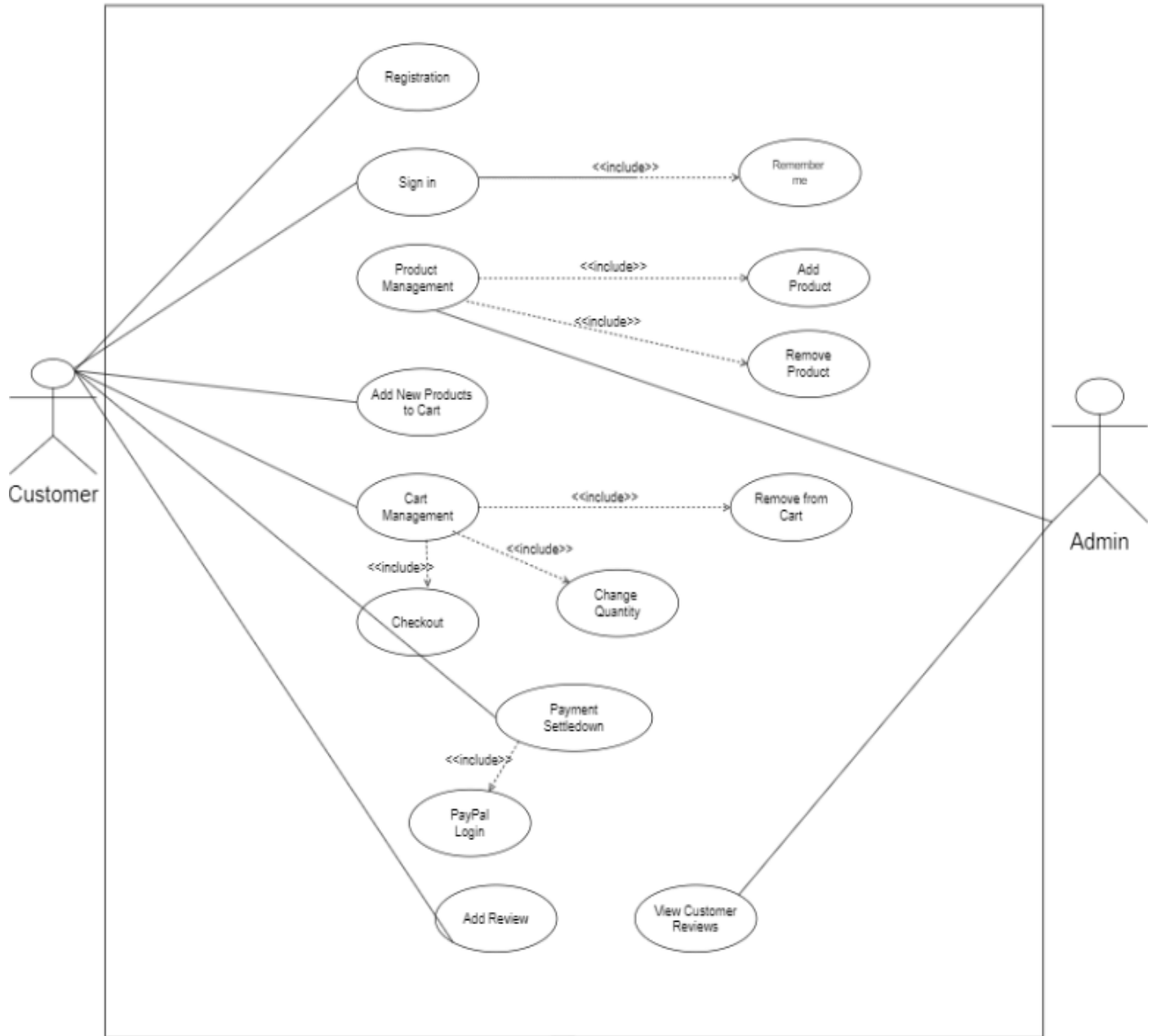
Actions:

- Combining safe payment gateway integration
- Payment information encryption

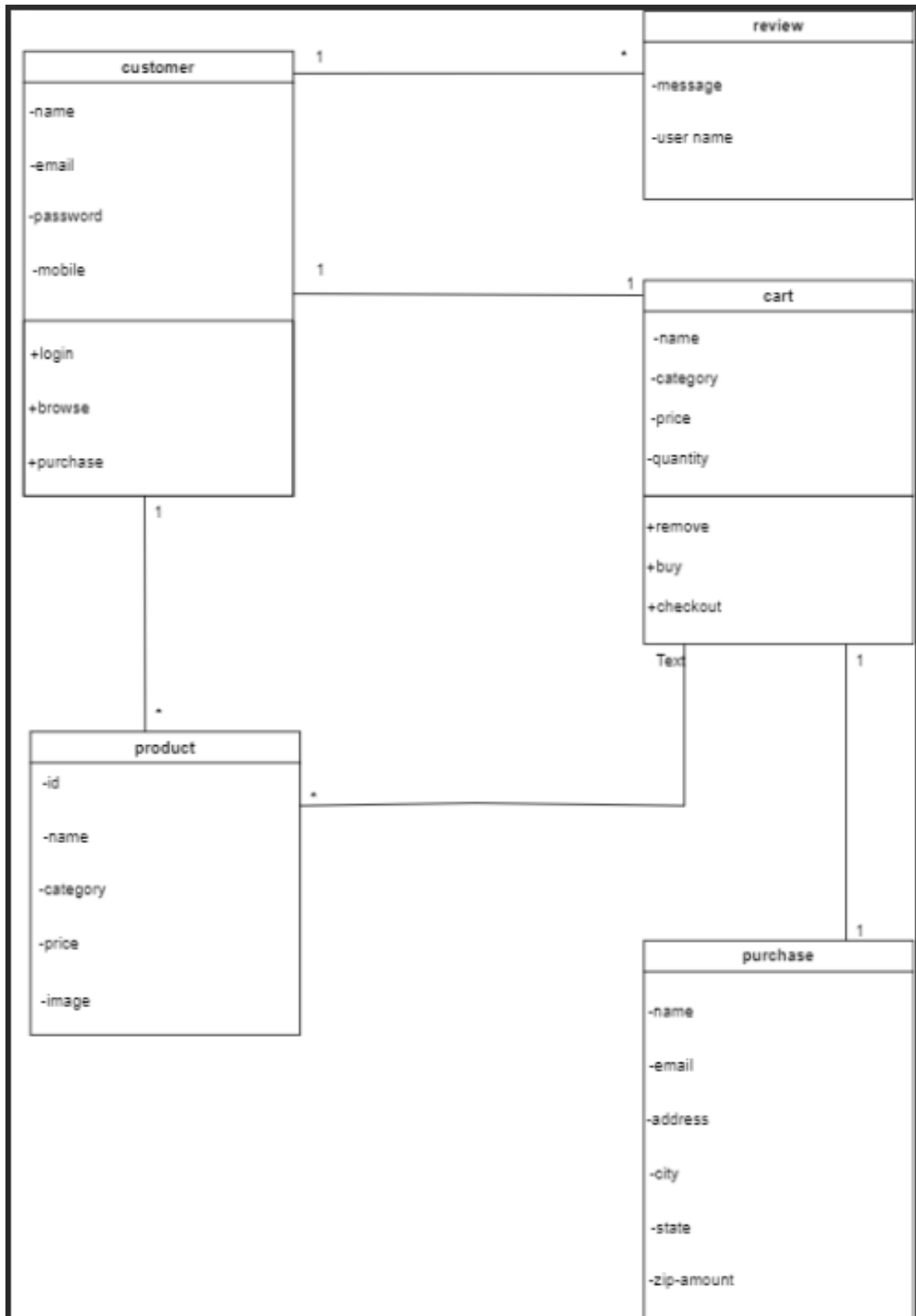


## Use Case Diagram

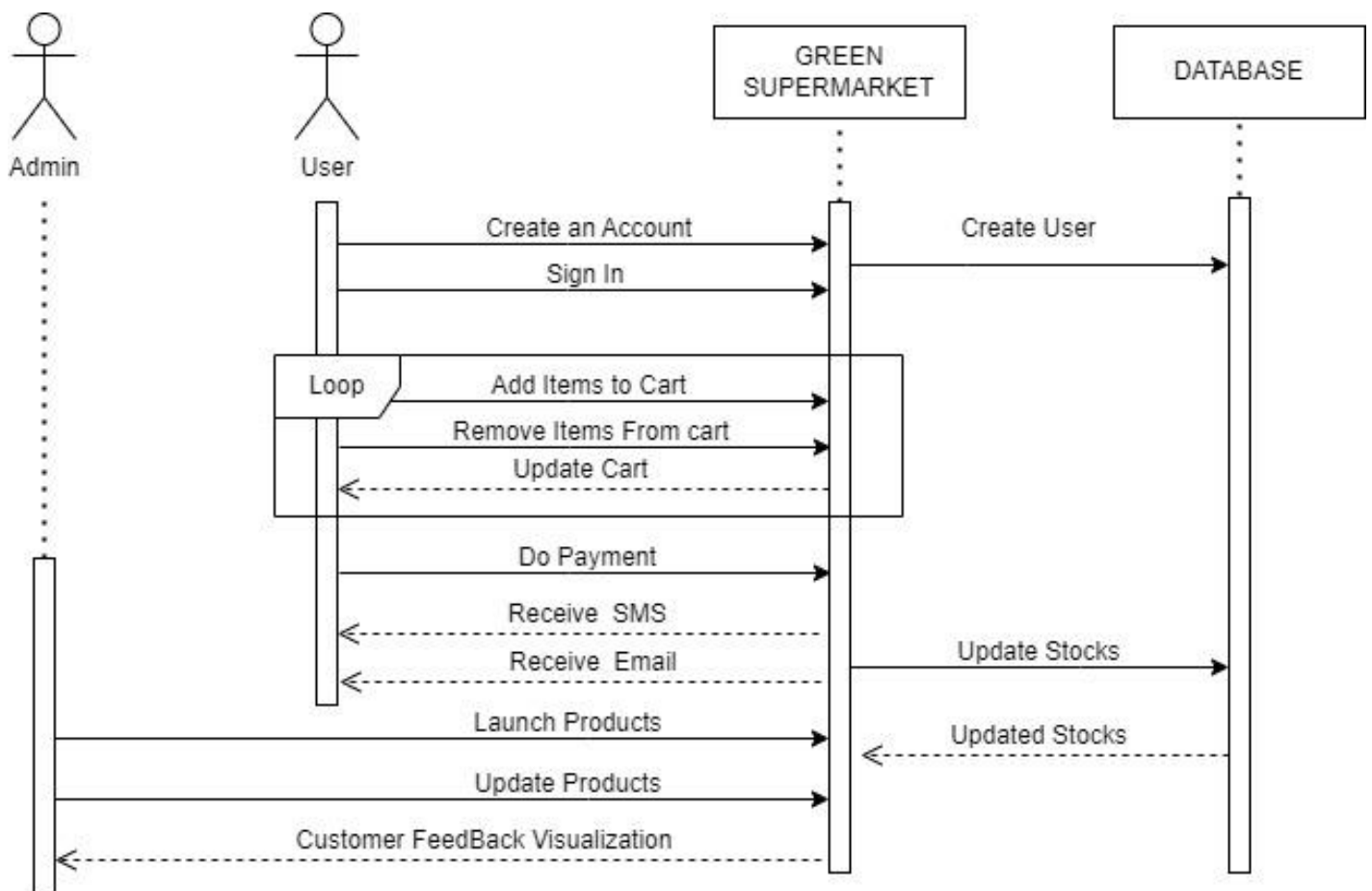
### Green Supermarket Management System



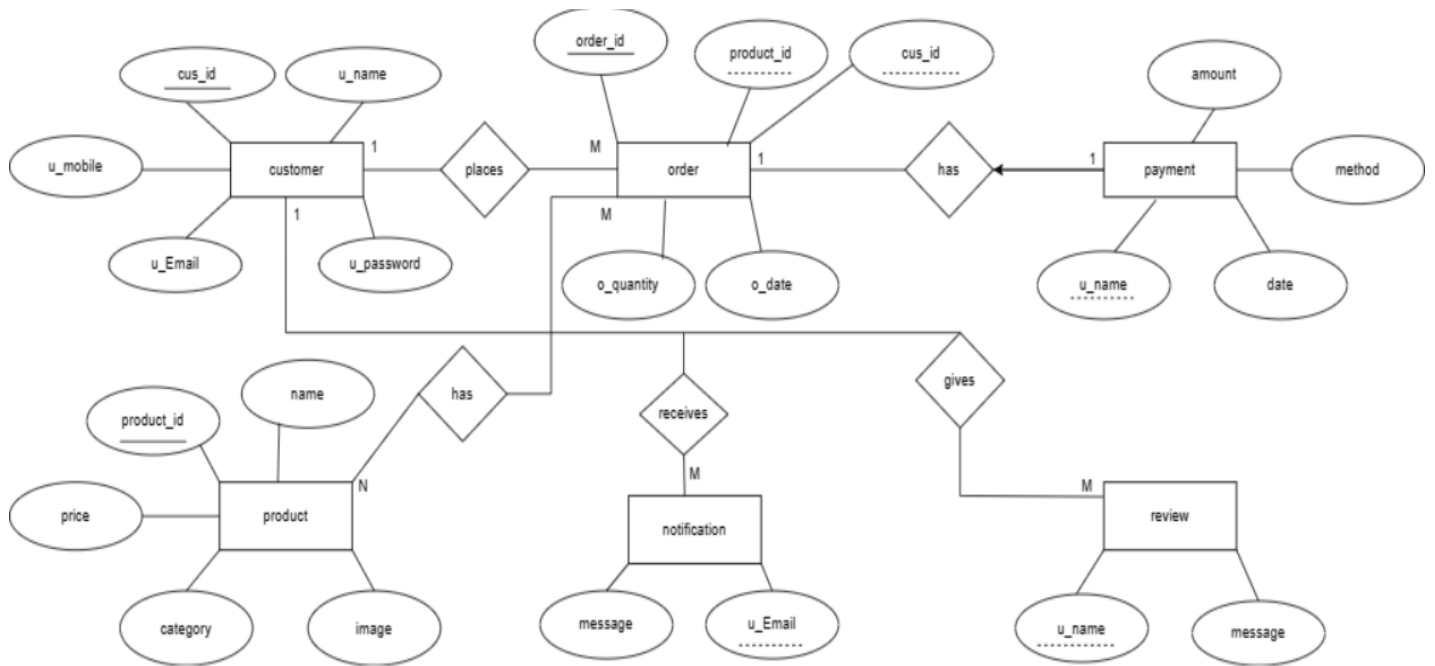
## Class Diagram



## Sequence Diagram

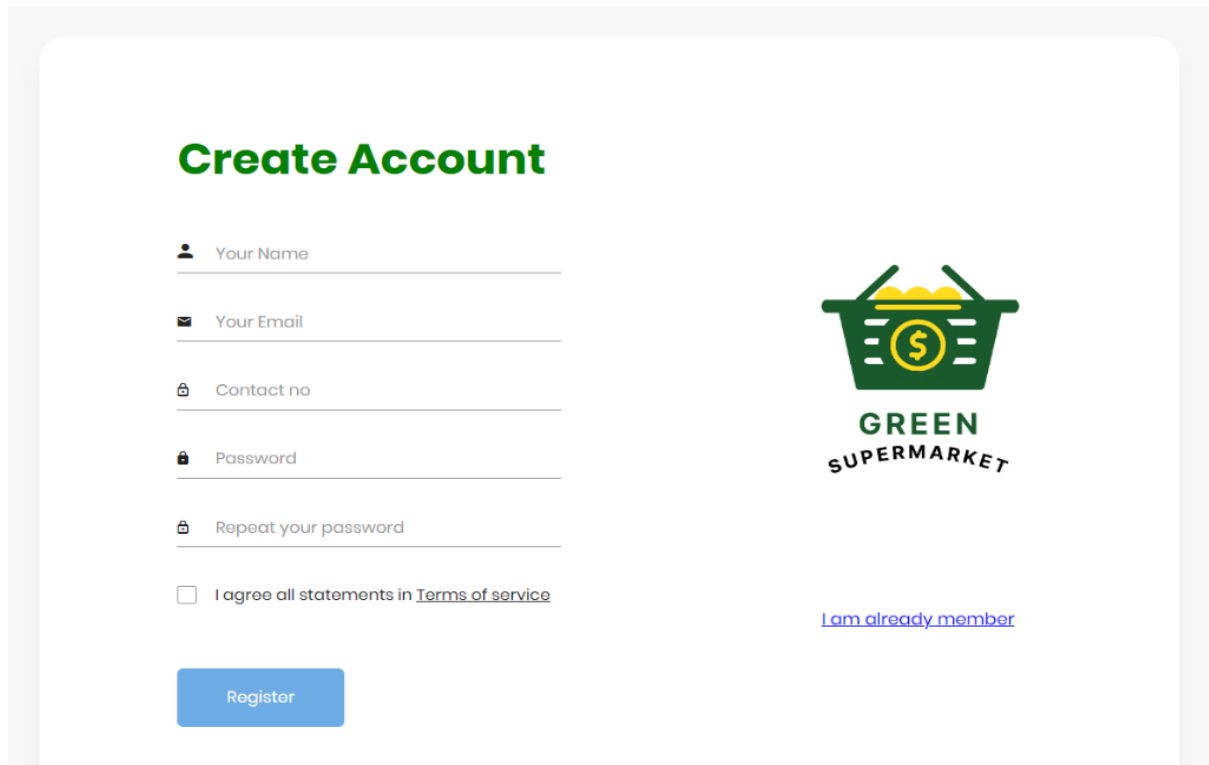


## ER Diagram



## Front-End Designs

### registration.jsp



The image shows a web form titled "Create Account" for "GREEN SUPERMARKET". The form is contained within a light gray rounded rectangle. It features five input fields, each with a small icon to its left: a person icon for "Your Name", an envelope icon for "Your Email", a phone icon for "Contact no", a padlock icon for "Password", and another padlock icon for "Repeat your password". Below these fields is a checkbox labeled "I agree all statements in [Terms of service](#)". At the bottom left of the form is a blue "Register" button. To the right of the form, there is a logo for "GREEN SUPERMARKET" which consists of a green shopping basket with a yellow dollar sign inside, and the text "GREEN SUPERMARKET" below it. Below the logo is a blue link that says "I am already member".

**Create Account**

Your Name

Your Email

Contact no

Password

Repeat your password


☐ I agree all statements in [Terms of service](#)

[I am already member](#)

Register

The user registration form for the website is represented by this jsp page. The user's name, email, phone number, and password are all requested in the form fields. Every input area has an icon, and the design is responsive and contemporary. The servlet "Register" manages the registration process via an asynchronous JavaScript (AJAX) function, enabling form submission without requiring a page reload. Further features include a client-side registration status check and an attractive alert library that presents success or failure notifications in an eye-catching way. In addition to offering a link for current members to log in, the website promotes user involvement.

**login.jsp:**





**GREEN  
SUPERMARKET**

Still Don't Have an Account?  
[Create an account](#)

# Welcome

## Sign in

 Your Name

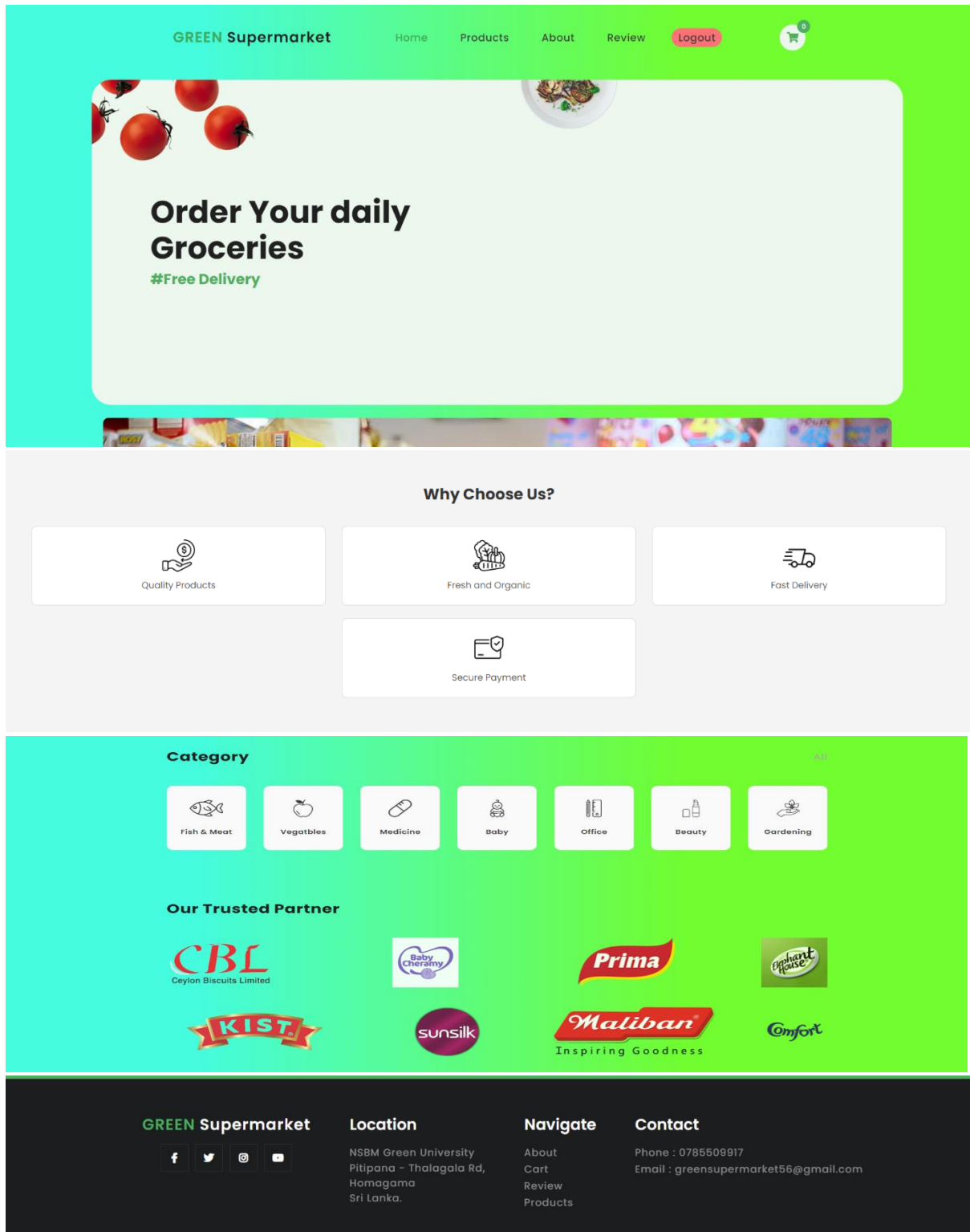
 Password

☐ Remember me

Log in

The user login interface on this jsp website has a vivid green backdrop. It has a form with a "Remember me" button and fields for inputting a username and password. The website offers a clear "Create an account" link and a greeting for new visitors. Furthermore, it uses JavaScript to send a charming warning in the event that the login attempt fails, offering an error message that is easy to understand. The aesthetically pleasing design encourages a smooth login process.

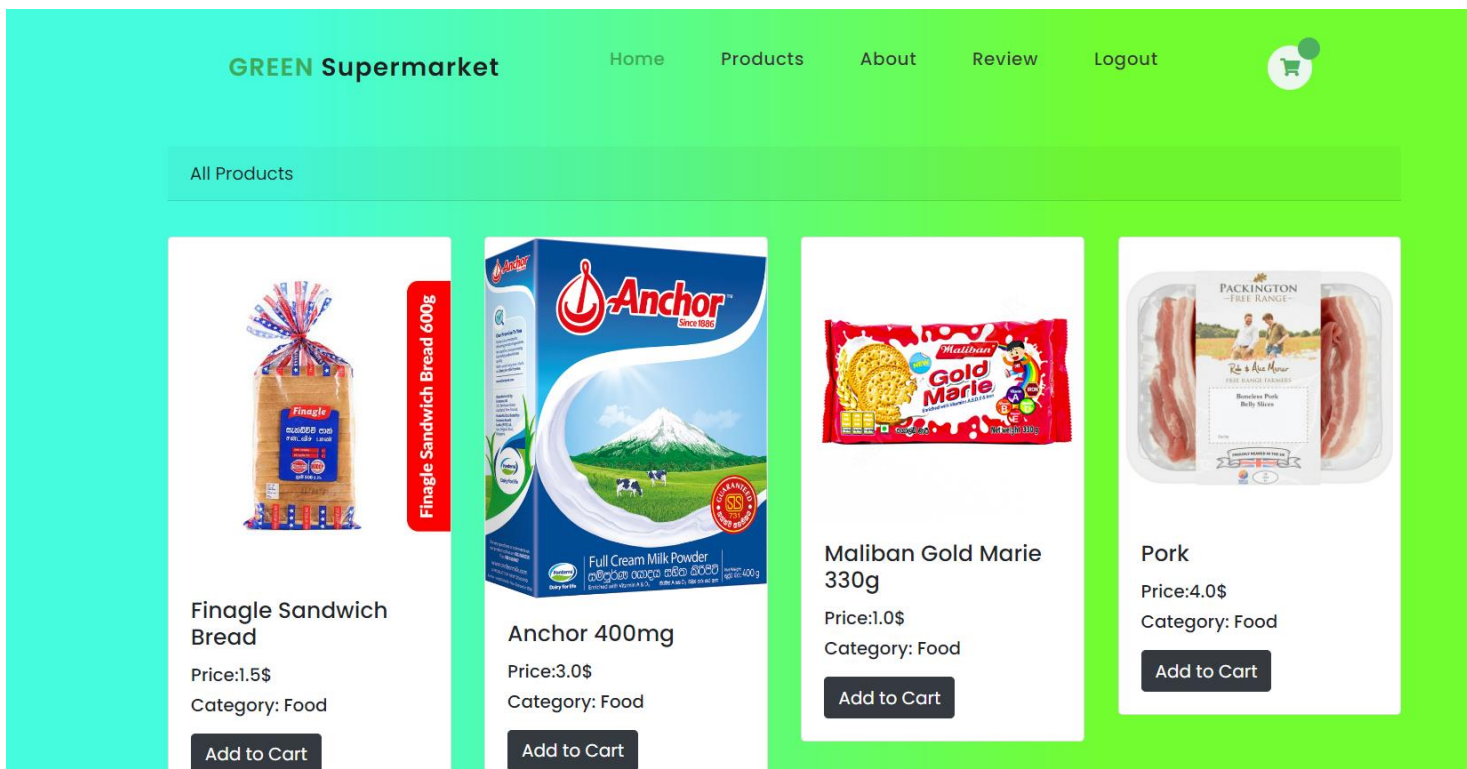
index.jsp:



The front end of an online Green Supermarket website is displayed on this jsp page. It has a striking design with a color palette centered around green. The page has a carousel with several photos shown, a search banner that is prominent, and a responsive navigation bar. The items are arranged by category, and the "Why Choose Us" section emphasizes important

features. Important elements are included in the footer, such as the location of the supermarket, links for navigation, and contact information. Credibility is increased with social media icons and reliable partner branding. The page has an easy-to-use layout that facilitates interaction and navigation, and the usage of dynamic content improves the entire purchasing experience.

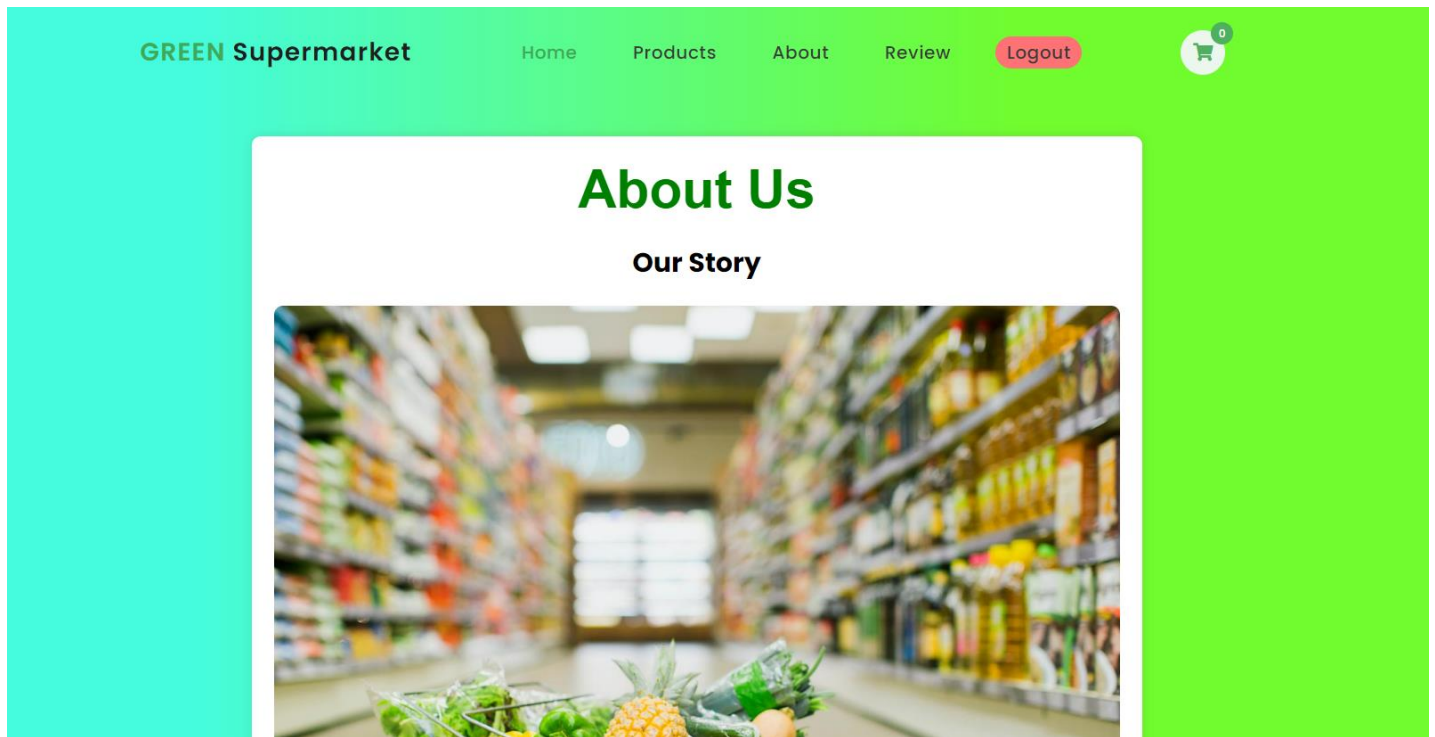
## product.jsp:



This Java Server Page (JSP) file is an online supermarket's product listing page. Product details, including image, name, price, and category, are shown on product cards that are dynamically retrieved from the database. Customers can utilize a specific "Add to Cart" button to add items to their shopping basket. The page has an easy-to-use navigation bar, a footer with important links and contact details, and a responsive design. For style, Bootstrap and Font Awesome are used. Additionally, the code verifies if a user is authorized; if so, the user's information is shown on the page.

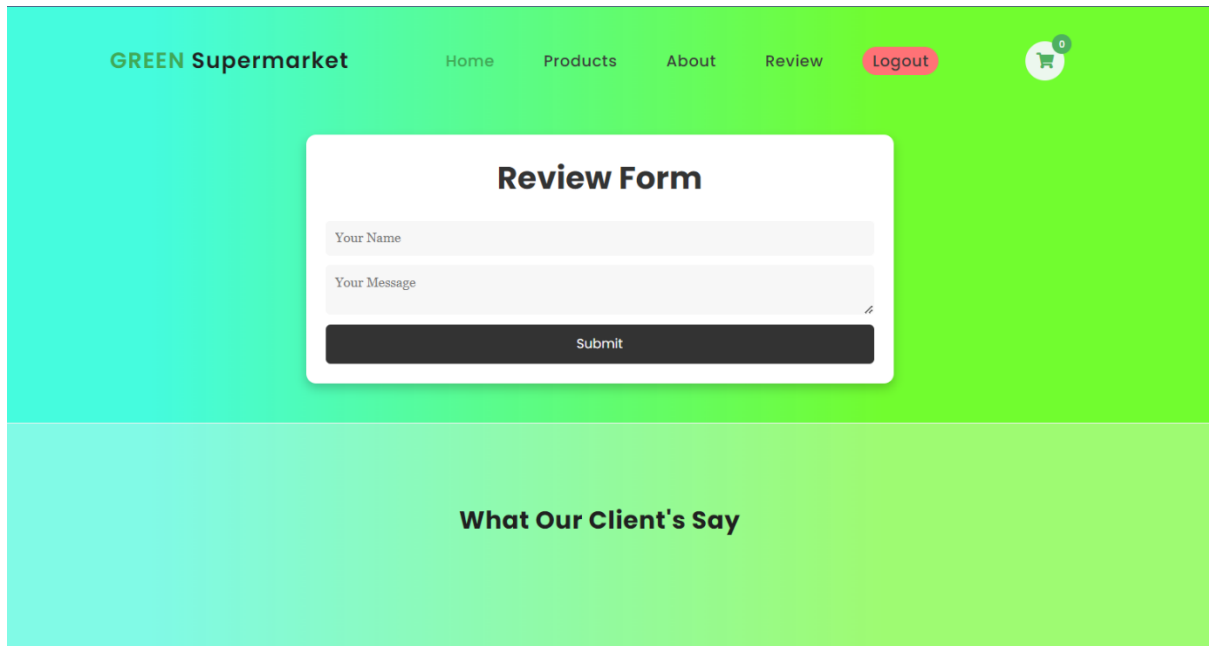


**about.jsp:**



The "About Us" page for GREEN Supermarket is represented by this JavaServer Pages (JSP) file. The page has a navigation bar, a simple, adaptable design, and dynamic content that tells the supermarket's history, goal, and ambition. The visual appeal is improved by the use of embedded styles and photos, which provide a thorough summary of GREEN Supermarket's dedication to sustainability, community involvement, and openness. The page highlights feature like online payment and improved user interfaces while outlining the difficulties the supermarket's website is facing and imagining a digital makeover to bring it into line with current standards. Important links and contact details are included in the bottom, which completes the informational and well-organized webpage.

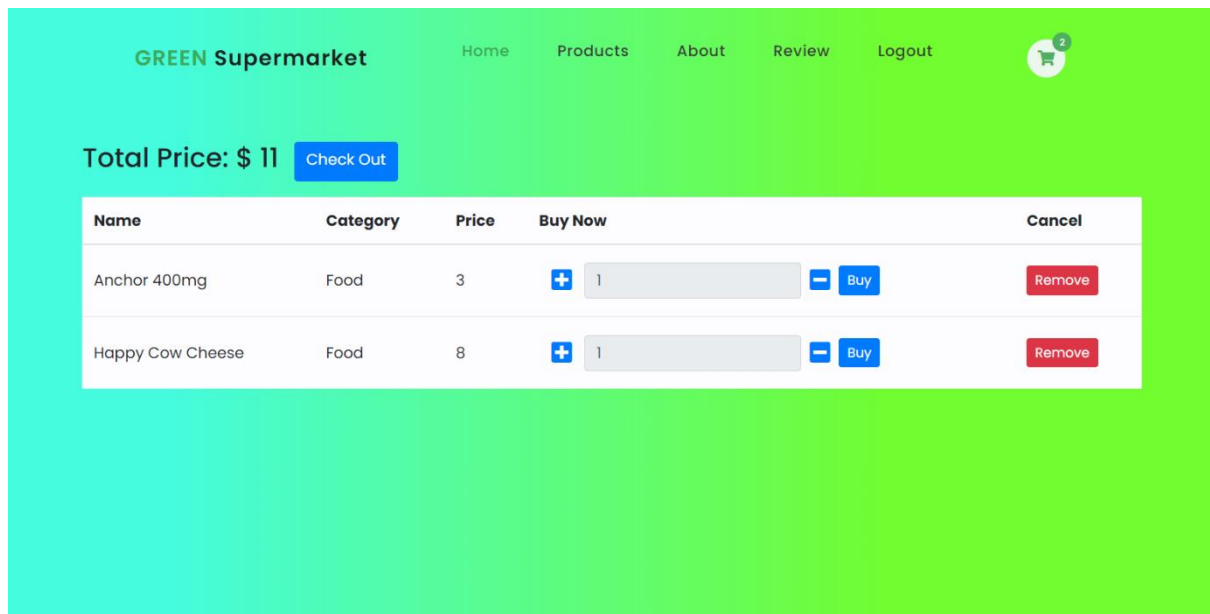
**review.jsp:**



The screenshot displays the 'Review Form' on the GREEN Supermarket website. The page has a green header with the logo 'GREEN Supermarket' and navigation links: Home, Products, About, Review, and a pink 'Logout' button. A shopping cart icon with a '0' badge is in the top right. The form itself is a white box with a black border, containing a title 'Review Form', a 'Your Name' text input, a 'Your Message' text area, and a black 'Submit' button. Below the form is a green section titled 'What Our Client's Say'.

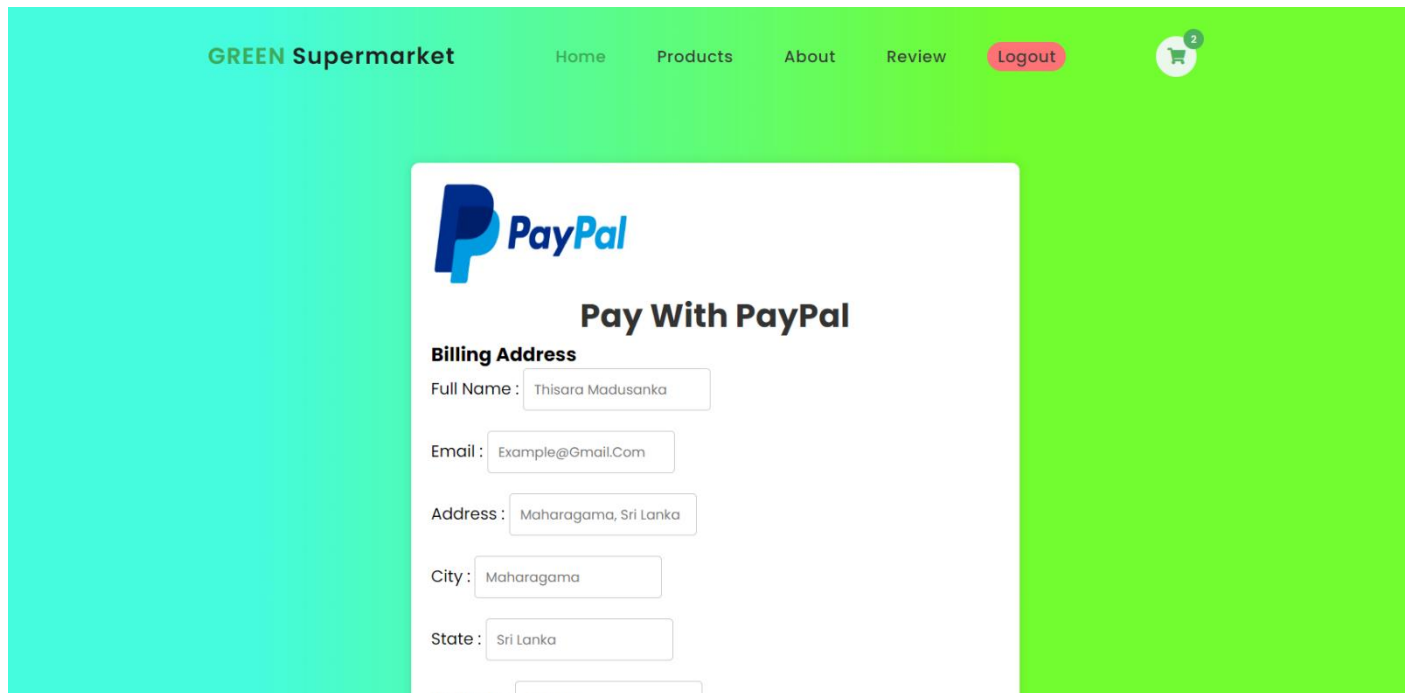
The "Review" portion of the GREEN Supermarket website is represented by this Java Server Pages (JSP) file. The page has a user-friendly, dynamic client review form that uses JavaScript to process form submissions asynchronously. Without having to reload the website, users may add their names and messages and post reviews. Furthermore, current client reviews that are kept in a MySQL database are dynamically retrieved and shown. With a well-organized layout and a dynamic navigation bar, the website keeps its visual coherence. The visual appeal is improved by the use of Google Fonts and FontAwesome icons, which also give users an interactive platform to post and read opinions about GREEN Supermarket.

**cart.jsp:**



The shopping cart page for the GREEN Supermarket website is represented by this JavaServer Pages (JSP) file. The product selections made by the user are dynamically shown, with the option to modify the amount and obtain the total price. The page updates and retrieves cart information by integrating with a MySQL database. Before checking out, users can change the quantity of products they want to buy. The design of the site remains responsive and includes a navigation bar that leads to different parts. The visual attractiveness is improved by the use of Bootstrap and Font Awesome icons, which also offer an effective and user-friendly interface for managing and evaluating chosen products prior to completing a transaction.

## billing.jsp:

















The screenshot displays the 'billing.jsp' page of the GREEN Supermarket website. The page features a green header with the site name 'GREEN Supermarket' and navigation links: 'Home', 'Products', 'About', 'Review', and a red 'Logout' button. A shopping cart icon with a '2' badge is in the top right. The main content area is white and contains the PayPal logo, the heading 'Pay With PayPal', and a 'Billing Address' section. This section includes input fields for 'Full Name' (filled with 'Thisara Madusanka'), 'Email' (filled with 'Example@Gmail.Com'), 'Address' (filled with 'Maharagama, Sri Lanka'), 'City' (filled with 'Maharagama'), and 'State' (filled with 'Sri Lanka'). A 'Zip Code' field is partially visible at the bottom.






















The GREEN Supermarket website's payment processing page, which incorporates PayPal as a payment option, is represented by this JSP (JavaServer Pages) file. The user-friendly PayPal interface enables users to safely pay for the products they have chosen. A billing address form with the user's name, email address, address, city, state, zip code, and the amount due is displayed on the website. Custom CSS styles are used to achieve a visually appealing layout. The payment procedure is started with the PayPal logo and a "Pay with PayPal" button. The "processPayPalPayment" servlet or a related backend component most likely houses the backend functionality for handling PayPal payments.

For testing reasons, this website incorporates PayPal payments in a sandbox setting. It has a checkout form with fields to enter the money and billing information. An endpoint called "processPayPalPayment" receives the form. The precise processing logic is not specified, although it probably entails managing the payment confirmation and using PayPal REST API credentials. The "Pay with PayPal" button click results in the activation of a confirmation alert. CSS is used in the form's stylistic design to create an eye-catching layout. More information on the "processPayPalPayment" servlet is needed for complete implementation, and testing in the PayPal sandbox environment should be done thoroughly before releasing to production.

## Used Library Files :

- ▼ Libraries
  - >  mysql-connector-j-8.2.0.jar
  - >  javax.mail-1.6.2.jar
  - >  javax.mail-api-1.6.2.jar
  - >  square-36.0.0.20231213.jar
  - >  rest-api-sdk-1.14.0.jar
  - >  slf4j-api-2.0.9.jar
  - >  logback-classic-1.4.9.jar
  - >  log4j-api-2.22.1.jar
  - >  log4j-core-2.22.1.jar
  - >  gson-2.2.2.jar
  - >  activation-1.1.1.jar
  - >  commons-email-1.6.0.jar
  - >  JDK 20 (Default)
  - >  Apache Tomcat or TomEE

## Servlets and Java Classes Used :

- ▼ Source Packages
  - ▼  cn.develop.connection
    -  DbCon.java
  - ▼  cn.develop.dao
    -  ProductDao.java
    -  UserDao.java
  - ▼  cn.develop.model
    -  Cart.java
    -  Product.java
    -  User.java
  - ▼  cn.develop.servlet
    -  AddToCartServlet.java
    -  LoginServlet.java
    -  LogoutServlet.java
    -  ProcessPaypalPayment.java
    -  QuantityIncDecServlet.java
    -  RemoveFromCartServlet.java
    -  ReviewServlet.java
  - ▼  com.uniquedeveloper.registration
    -  RegistrationServlet.java
    -  RegistrationServletImpl.java
    -  login.java

## Back-End Codes

## Navigation Bar :

```
<a class="navigation">
  <a href="index.jsp" class="logo"> <span>GREEN&nbsp;</span>Supermarket </a>
  <!--menu-btn---->
  <input type="checkbox" class="menu-btn" id="menu-btn" />
  <label for="menu-btn" class="menu-icon">
    <span class="nav-icon"></span>
  </label>
  <!--menu----->
  <ul class="menu">
    <li><a href="index.jsp" class="active">Home</a></li>
    <li><a href="product.jsp">Products</a></li>
    <li><a href="about.jsp">About</a></li>
    <li><a href="review.jsp">Review</a></li>
    <li><a style="background-color: #FF7276; border-radius: 30px;" href="log-out">Logout</a></li>
  </ul>
  <!--right-nav- (cart-like)-->
  <div class="right-nav">
    <!--cart----->
    <a href="cart.jsp" class="cart">
      <i class="fas fa-shopping-cart"></i>
      <span><%= (session.getAttribute("cart-list") != null) ? ((ArrayList<Cart>) session.getAttribute("cart-list")).size() : 0 %></span>
    </a>
  </div>
</nav>
```

## Footer:

```

<footer>
<div class="footer-container">
<div class="footer-logo">
<a href="index.jsp" class="logo"> <span>GREEN&nbsp;</span>Supermarket </a>
<!--social----->
<div class="footer-social">
<a href="#"><i class="fab fa-facebook-f"></i></a>
<a href="#"><i class="fab fa-twitter"></i></a>
<a href="#"><i class="fab fa-instagram"></i></a>
<a href="#"><i class="fab fa-youtube"></i></a>
</div>
</div>
<div class="footer-links">
<strong>Location</strong>
<ul>
<li><a href="#">NSBM Green University<br>Pitipana - Thalagala Rd,<br>Homagama<br>Sri Lanka.</a></li>
</ul>
</div>
<div class="footer-links">
<strong>Navigate</strong>
<ul>
<li><a href="about.jsp">About</a></li>
<li><a href="cart.jsp">Cart</a></li>
<li><a href="review.jsp">Review</a></li>
<li><a href="product.jsp">Products</a></li>
</ul>
</div>
<div class="footer-links">
<strong>Contact</strong>
<ul>
<li><a href="#">Phone : 0785509917</a></li>
<li><a href="#">Email : greensupermarket56@gmail.com</a></li>
</ul>
</div>
</div>
</footer>

```

# index.jsp:

```
index.jsp x
132 </head>
158
159 <section id="search-banner">
160
161 
162 
163
164 <div class="search-banner-text">
165 <h1>Order Your daily Groceries</h1>
166 <strong>#Free Delivery</strong>
167 </div>
168 </section>
169
170 <section class="carousel-container">
171 <div class="carousel-arrow left-arrow" onclick="moveCarousel('left')">&#9665;</div>
172 <div class="carousel-images" id="carouselImages">
173 <!-- Your carousel images here -->
174 <div class="carousel-image"></div>
175 <div class="carousel-image"></div>
176 <div class="carousel-image"></div>
177 <div class="carousel-image"></div>
178 <div class="carousel-image"></div>
179 <div class="carousel-image"></div>
180 <div class="carousel-image"></div>
181 <div class="carousel-image"></div>
182 <div class="carousel-image"></div>
183 <!-- Add more images as needed -->
184 </div>
185 <div class="carousel-arrow right-arrow" onclick="moveCarousel('right')">&#9665;</div>
186 </section>
187
188
189 <div class="why-choose-us-section">
190 <h2 class="why-choose-us-title">Why Choose Us?</h2>
191 <div class="why-item">
192 
193 <p>Quality Products</p>
194 </div>
195 <div class="why-item">
196 
197 <p>Fresh and Organic</p>
198 </div>
199 <div class="why-item">
200 
201 <p>Fast Delivery</p>
202 </div>
203 <div class="why-item">
204 
205 <p>Secure Payment</p>
206 </div>
207 </div>
208 <section id="category">
209 <div class="category-heading">
210 <h2>Category</h2>
211 <span>All</span>
212 </div>
213 <div class="category-container">
214 <a href="#" class="category-box">
215 
216 <span>Fish & Meat</span>
217 </a>
218 <a href="#" class="category-box">
219 
220 <span>Vegetables</span>
221 </a>
222 <a href="#" class="category-box">
223 
224 <span>Medicine</span>
225 </a>
226 </div>
227
228 <div class="partner-heading">
229 <h3>Our Trusted Partner</h3>
230 </div>
231
232 <div class="logo-container">
233 
234 
235 
236 
237 </div>
238 <div class="logo-container">
239 
240 
241 
242 
243 </div>
244 </section>
```

# login.jsp

```
login.jsp x
Source History
15 </head>
16 <body style="background: #90EE90">
17     <input type="hidden" id="status" value="<%=request.getAttribute("status")%>">
18
19     <div class="main" style="background: #90EE90">
20
21         <!-- Sing in Form -->
22         <section class="sign-in">
23             <div class="container">
24                 <div class="signin-content">
25                     <div class="signin-image">
26                         <figure>
27                             
28                         </figure>
29                         <h1 style="font-size:10px">Still Don't Have an Account?</h1>
30                         <a href="registration.jsp" class="signup-image-link" style="color: blue;font-size: 15px">Create an
31                             account</a>
32                     </div>
33
34                     <div class="signin-form">
35                         <h1 style="color:green; font-size:50px">Welcome</h1>
36
37                         <h2 class="form-title">Sign in</h2>
38                         <form method="post" action="login" class="register-form"
39                             id="login-form">
40                             <div class="form-group">
41                                 <label for="username"><i
42                                     class="zmdi zmdi-account material-icons-name"></i></label> <input
43                                     type="text" name="username" id="username"
44                                     placeholder="Your Name" />
45                             </div>
46                             <div class="form-group">
47                                 <label for="password"><i class="zmdi zmdi-lock"></i></label> <input
48                                     type="password" name="password" id="password"
49                                     placeholder="Password" />
50                             </div>
51                         </div>
52
53                         <div class="form-group">
54                             <input type="checkbox" name="remember-me" id="remember-me"
55                                 class="agree-term" /> <label for="remember-me"
56                                 class="label-agree-term"><span><span></span></span>Remember
57                                 me</label>
58                         </div>
59                         <div class="form-group form-button">
60                             <input type="submit" name="signin" id="signin"
61                                 class="form-submit" value="Log in" />
62                         </div>
63                     </form>
64                 </div>
65             </div>
66         </section>
67     </div>
68
69     <!-- JS -->
70     <script src="vendor/jquery/jquery.min.js"></script>
71     <script src="js/main.js"></script>
72     <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
73     <link rel="stylesheet" href="alert/dist/sweetalert.css">
74
75     <script type="text/javascript">
76         var status=document.getElementById("status").value;
77         if(status === "failed"){
78             swal("Sorry","Wrong username or password","failed");
79         }
80     </script>
81 </body>
```



## registration.jsp:

```
<body>

    <input type="hidden" id="status" value="<%=request.getAttribute("status")%>">

    <div class="main">

        <!-- Sign up form -->
        <section class="signup">
            <div class="container">
                <div class="signup-content">
                    <div class="signup-form">
                        <h2 class="form-title" style="color: green">Create Account</h2>

                        <form method="post" action="Register" class="register-form"
                            id="register-form">
                            <div class="form-group">
                                <label for="name"><i class="zmdi zmdi-account material-icons-name"></i></label> <input
                                    class="zmdi zmdi-account material-icons-name"></i></label> <input
                                    type="text" name="name" id="name" placeholder="Your Name" />
                                </div>
                                <div class="form-group">
                                    <label for="email"><i class="zmdi zmdi-email"></i></label> <input
                                        type="email" name="email" id="email" placeholder="Your Email" />
                                    </div>
                                    <div class="form-group">
                                        <label for="contact"><i class="zmdi zmdi-lock-outline"></i></label>
                                        <input type="text" name="contact" id="contact"
                                            placeholder="Contact no" />
                                    </div>
                                    <div class="form-group">
                                        <label for="pass"><i class="zmdi zmdi-lock"></i></label> <input
                                            type="password" name="pass" id="pass" placeholder="Password" />
                                    </div>
                                    <div class="form-group">
                                        <label for="re-pass"><i class="zmdi zmdi-lock-outline"></i></label>
                                        <input type="password" name="re_pass" id="re_pass"
                                            placeholder="Repeat your password" />
                                    </div>
                                </div>
                                <div class="form-group form-button">
                                    <input type="submit" name="signup" id="signup"
                                        class="form-submit" value="Register" onclick="registerUser()" />
                                </div>
                            </form>
                        </div>
                        <div class="signup-image">
                            <figure>
                                
                            </figure>
                            <a href="login.jsp" class="signup-image-link" style="color: blue">I am already
                                member</a>
                        </div>
                    </div>
                </div>
            </section>
        </div>

        <!-- JS -->
        <script src="vendor/jquery/jquery.min.js"></script>
        <script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>

        <script src="js/main.js"></script>
        <script src="https://unpkg.com/sweetalert/dist/sweetalert.min.js"></script>
        <link rel="stylesheet" href="alert/dist/sweetalert.css">

        <script type="text/javascript">
            var status=document.getElementById("status").value;
            if(status === "success"){
                swal("Congrats","Account Created Successfully","success");
            }
        </script>
    </body>
```

**product.jsp:**

```
product.jsp x
Source History
<%@page import="cn.develop.connection.DbCon"%>
<%@page import="cn.develop.dao.ProductDao"%>
<%@page import="cn.develop.model.*"%>
<%@page import="com.uniquedeveloper.registration.*"%>
<%@page import="java.util.*"%>
<% page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%
User auth=(User)request.getSession().getAttribute("auth");
if(auth!=null){
    request.setAttribute("person",auth);
}
ProductDao pd = new ProductDao(DbCon.getConnection());
List<Product> products = pd.getAllProducts();
ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list");
if (cart_list != null) {
    request.setAttribute("cart_list", cart_list);
}
%>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta http-equiv="X-UA-Compatible" content="IE=edge" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<link rel="stylesheet" href="css/styleSheet.css" />
<link rel="shortcut icon" href="images/Green.jpg" />
<link
rel="stylesheet"
href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
integrity="sha384-AyMc384W5cVb32cuHtOA93w35dYTvvhLPVnYs9eStHFgJvOvKxVfELGroGkvsg+p"
crossorigin="anonymous"
/>
</div>
<div class="container">
<div class="card-header my-3">All Products</div>
<div class="row">
<%
if (!products.isEmpty()) {
for (Product p : products) {
%>
<div class="col-md-3 my-3">
<div class="card w-100">

<div class="card-body">
<h5 class="card-title"><%=p.getName() %></h5>
<h6 class="price">Price:<%=p.getPrice() %></h6>
<h6 class="category">Category: <%=p.getCategory() %></h6>
<div class="mt-3 d-flex justify-content-between">
<a class="btn btn-dark" href="add-to-cart?id=<%=p.getId() %>">Add to Cart</a>
</div>
</div>
</div>
</div>
<%
} else {
out.println("There is no products");
}
%>
</div>
</div>
<div class="card-footer">
</div>
```

# review.jsp:

```
review.jsp X
Source History
1 <%@ page import="java.sql.*" %>
2 <%@ page import="java.io.*" %>
3 <%@ page contentType="text/html;charset=UTF-8" language="java" %>
4 <%@page import="java.util.ArrayList"%>
5 <%@page import="cn.develop.model.Cart"%>
6
7 <%
8     if (session.getAttribute("name")==null){
9         response.sendRedirect("login.jsp");
10    }
11 %>
12
13 <html lang="en">
14 <head>
15     <!-- Your existing head content -->
16     <!-- ... -->
17     <script src="https://kit.fontawesome.com/80224b9ef7.js" crossorigin="anonymous"></script>
18     <link rel="stylesheet" href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
19         integrity="sha384-AyMcEC3Yw5cVb32cuHtOA93w35dYTvvhLPVnYs9eStHfGJvOVKXVfELGroGkvsg+p"
20         crossorigin="anonymous"/>
21     <link rel="stylesheet" href="css/styleSheet.css" />
22     <link rel="stylesheet" href="css/review.css" />
23     <link rel="preconnect" href="https://fonts.googleapis.com" />
24     <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
25     <link href="https://fonts.googleapis.com/css2?family=Poppins:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400" rel="stylesheet"/>
26     <title>GREEN Supermarket</title>
27
28 </head>
29
30 <script>
31     function submitForm() {
32         var username = document.getElementById("username").value;
33         var message = document.getElementById("message").value;
34
35         // Basic validation (you may want to enhance this)
36         if (username.trim() == "" || message.trim() == "") {
37             alert("Please enter both username and message.");
38             return;
39         }
40
41         var xhr = new XMLHttpRequest();
42         xhr.onreadystatechange = function () {
43             if (xhr.readyState == 4 && xhr.status == 200) {
44                 document.getElementById("client-box-container").innerHTML += xhr.responseText;
45                 document.getElementById("username").value = "";
46                 document.getElementById("message").value = "";
47             }
48         };
49         xhr.open("POST", "ReviewServlet", true);
50         xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
51         xhr.send("username=" + encodeURIComponent(username) + "&message=" + encodeURIComponent(message));
52     }
53 </script>
54 </head>
55 <body>
56
57 <div class="c_us" padding="20px">
58     <h1>Review Form</h1>
59     <form onsubmit="event.preventDefault(); submitForm();">
60         <input type="text" id="username" placeholder="Your Name" required>
61         <textarea id="message" placeholder="Your Message" required></textarea>
62         <button type="submit">Submit</button>
63     </form>
64 </div>
65
66 <section id="clients">
67     <div class="client-heading">
68         <h3>What Our Client's Say</h3>
69     </div>
70     <div class="client-box-container" id="client-box-container">
71
72 </div>
```

```

review.jsp
Source History
100
101
102 Connection connection = null;
103 PreparedStatement preparedStatement = null;
104 ResultSet resultSet = null;
105
106 try {
107     // Assuming you have a database named "your_database" and a table named "reviews"
108     String url = "jdbc:mysql://localhost:3306/esupermarket";
109     String dbUser = "admin";
110     String dbPassword = "8645";
111
112     Class.forName("com.mysql.cj.jdbc.Driver");
113     connection = DriverManager.getConnection(url, dbUser, dbPassword);
114
115     // Retrieve messages from the "reviews" table
116     String query = "SELECT username, message FROM reviews";
117     preparedStatement = connection.prepareStatement(query);
118     resultSet = preparedStatement.executeQuery();
119
120     // Display the existing messages
121     while (resultSet.next()) {
122         String username = resultSet.getString("username");
123         String message = resultSet.getString("message");
124     }
125
126     <div class="client-box">
127     <div class="client-profile">
128     <div class="profile-text">
129     <strong><%= username %></strong>
130     </div>
131     </div>
132     <p><%= message %></p>
133     </div>
134     <%= // End of while loop %>
135     <!-- Close database resources -->
136     <%=
137
138     } catch (Exception e) {
139         e.printStackTrace();
140     }
141
142 }
143
144
145
146

```

## about.jsp:

```

about.jsp
Source History
95
96
97
98
99 <h1>About Us</h1>
100
101
102
103 <h2>Our Story</h2>
104 
105 <p>Green Supermarket is a trailblazing entity in the supermarket industry, positioned at the forefront of eco-conscious retail. Over the years, the
106 <p>The concept of the Green Supermarket revolves around the principles of sustainability, environmental responsibility, and a dedication to providin
107 <p>One of the core advantages of the Green Supermarket concept is its contribution to environmental conservation. By prioritizing eco-friendly produ
108 <p>Additionally, the Green Supermarket concept fosters community engagement and supports local economies. By emphasizing the importance of locally s
109 <p>Furthermore, the Green Supermarket distinguishes itself through its dedication to transparency and ethical business practices. The supermarket en
110 <p>To address the challenges posed by the obsolete website, the Green Supermarket envisions a digital transformation that aligns with its core value
111 <p>In conclusion, the Green Supermarket concept represents a pioneering approach to retail, placing sustainability, community engagement, and transp
112 <br><br>
113 <h2>Our Mission</h2>
114 <p>The mission of Green Supermarket is to redefine the landscape of retail by embodying a steadfast commitment to sustainability, environmental resp
115
116 
117
118
119
120 </section>
121
122 <footer>
123 <div class="footer-container">
124 <!--logo-container----->
125 <div class="footer-logo">
126 <a href="index.jsp" class="logo"> <span>GREEN&nbsp;</span></span>Supermarket </a>
127 <!--social----->

```

## cart.jsp:

```
cart.jsp x
Source History
<%@page import="cn.develop.connection.DbCon"%>
<%@page import="com.uniquedeveloper.registration.*"%>
<%@page import="cn.develop.model.*"%>
<%@page import="cn.develop.servlet.*"%>
<%@page import="cn.develop.dao.*"%>
<%@page import="java.util.*"%>
<%@page import="java.text.DecimalFormat"%>
<%@page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%
DecimalFormat dcf = new DecimalFormat("###.##");
request.setAttribute("dcf", dcf);
User auth = (User) request.getSession().getAttribute("auth");
if (auth != null) {
    request.setAttribute("person", auth);
}
ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list");
List<Cart> cartProduct = null;
if (cart_list != null) {
    ProductDao pDao = new ProductDao(DbCon.getConnection());
    cartProduct = pDao.getCartProducts(cart_list);
    double total = pDao.getTotalCartPrice(cart_list);
    request.setAttribute("total", total);
    request.setAttribute("cart_list", cart_list);
}
%>
<!DOCTYPE html>
<html>
<head>

88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
<div class="container my-3">
    <div class="d-flex py-3"><h3>Total Price: $ ${total>0?dcf.format(total):0} </h3> <a class="mx-3 btn btn-primary" href="billing.jsp">Check
    <table class="table table-light">
        <thead>
            <tr>
                <th scope="col">Name</th>
                <th scope="col">Category</th>
                <th scope="col">Price</th>
                <th scope="col">Buy Now</th>
                <th scope="col">Cancel</th>
            </tr>
        </thead>
        <tbody>
            <%
            if (cart_list != null) {
                for (Cart c : cartProduct) {
            %>
            <tr>
                <td><%=c.getName() %></td>
                <td><%=c.getCategory() %></td>
                <td><%= dcf.format(c.getPrice()) %></td>
                <td>
                    <form action="order-now" method="post" class="form-inline">
                        <input type="hidden" name="id" value="<%= c.getId() %>" class="form-input">
                        <div class="form-group d-flex justify-content-between">
                            <a class="btn btn-sm btn-incre" href="quantity-inc-dec?action=inc&id=<%=c.getId() %>"><i cla
                            <input type="text" name="quantity" class="form-control" value="<%=c.getQuantity() %>" readonly
                            <a class="btn btn-sm btn-decre" href="quantity-inc-dec?action=dec&id=<%=c.getId() %>"><i cla
                        </div>
                    </form>
                    <a type="submit" class="btn btn-primary btn-sm" href="billing.jsp">Buy</a>
                </td>
                <td><a href="remove-from-cart?id=<%=c.getId() %>" class="btn btn-sm btn-danger">Remove</a></td>
            </tr>
        </tbody>
    </table>
    </div>
    </div>
    </div>
```

## billing.jsp:

```
billing.jsp x
Source History
<%@page import="cn.develop.dao.ProductDao"%>
<%@page import="cn.develop.connection.DbCon"%>
<%@page import="java.util.List"%>
<%@page import="cn.develop.model.User"%>
<%@page import="java.text.DecimalFormat"%>
<%@page import="java.util.ArrayList"%>
<%@page import="cn.develop.model.Cart"%>
<% page import="com.paypal.api.payments.*" %>
<% page import="com.paypal.base.rest.*" %>
<% page import="org.apache.commons.mail.*" %>
<%
    if (session.getAttribute("name")==null){
        response.sendRedirect("login.jsp");
    }
%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%
    DecimalFormat dcf = new DecimalFormat("###.##");
    request.setAttribute("dcf", dcf);
    User auth = (User) request.getSession().getAttribute("auth");
    if (auth != null) {
        request.setAttribute("person", auth);
    }
    ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute("cart-list");
    List<Cart> cartProduct = null;
    if (cart_list != null) {
        ProductDao pDao = new ProductDao(DbCon.getConnection());
        cartProduct = pDao.getCartProducts(cart_list);
        double total = pDao.getTotalCartPrice(cart_list);
        request.setAttribute("total", total);
        request.setAttribute("cart_list", cart_list);
    }
%>
<%@page import="java.util.Properties" %>
<%@page import="javax.mail.*" %>
<%@page import="javax.mail.internet.*" %>
<%
    String userEmail = request.getParameter("userEmail");
    if (userEmail != null && !userEmail.isEmpty()) {
        userEmail = userEmail.toLowerCase();

        Properties properties = new Properties();
        properties.put("mail.smtp.host", "smtp.gmail.com");
        properties.put("mail.smtp.port", "587");
        properties.put("mail.smtp.auth", "true");
        properties.put("mail.smtp.starttls.enable", "true");

        // Set up the session
        final String username = "greensupermarket56@gmail.com";
        final String password = "nume hmif eaga lesb";

        Session s1 = Session.getInstance(properties, new Authenticator() {
            protected PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(username, password);
            }
        });

        try {
            MimeMessage message = new MimeMessage(s1);
            message.setFrom(new InternetAddress(username));

            message.addRecipient(Message.RecipientType.TO, new InternetAddress(userEmail));
            message.setSubject("Payment Confirmation");
            message.setText("Thank you for your payment. Your order is confirmed.");

            Transport.send(message);
        } catch (MessagingException e) {
            // Handle exception
        }
    }
%>
<h3 class="title">billing address</h3>
<div class="inputBox">
    <span>full name :</span>
    <input type="text" placeholder="Thisara Madusanka" required />
</div>
<div class="inputBox">
    <span>email :</span>
    <input type="email" placeholder="example@gmail.com" required />
</div>
<div class="inputBox">
    <span>address :</span>
    <input type="text" placeholder="Maharagama, Sri Lanka" required />
</div>
<div class="inputBox">
    <span>city :</span>
    <input type="text" placeholder="Maharagama" required />
</div>
<div class="flex">
    <div class="inputBox">
        <span>state :</span>
        <input type="text" placeholder="Sri Lanka" required />
    </div>
    <div class="inputBox">
        <span>zip code :</span>
        <input type="text" placeholder="500 600" required />
    </div>
</div>
<label for="amount">Amount:</label>
<input type="text" name="amount" id="amount" value="<%= request.getAttribute("total") %>" required readonly />
<button type="submit" onclick="alert('Continue with Paypal');">Pay with PayPal</button>
</form>
</div>
</body>
```

## DbCon.Java Class:

```
package cn.develop.connection;

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

public class DbCon {
    private static Connection connection = null;
    public static Connection getConnection() throws ClassNotFoundException, SQLException{
        if(connection == null){
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection=DriverManager.getConnection(url: "jdbc:mysql://localhost:3306/esupermarket", user: "admin", password: "8645");
            System.out.print(s: "connected");
        }
        return connection;
    }
}
```

The supplied Java code specifies the DbCon class, which is in charge of creating a database connection, under the package cn.develop.connection. To guarantee that there is only one instance of the connection maintained, it makes use of the Singleton design pattern. The JDBC driver is used by the getConnection function to establish a MySQL database connection. It loads the driver, establishes the connection, and outputs a "connected" message if it hasn't already been established. The database URL, "jdbc:mysql://localhost:3306/esupermarket," and the login information, "admin" as the username and "8645" as the password, are included in the connection details. The code encourages modularity and reusability by encapsulating the database connectivity functionality.

## ProductDao.Java Class:

```
ProductDao.java x
Source History

1 package cn.develop.dao;
2
3 import cn.develop.connection.DbCon;
4 import java.sql.*;
5 import java.util.*;
6
7 import cn.develop.model.Cart;
8 import cn.develop.model.Product;
9
10 public class ProductDao {
11     private final Connection con;
12
13     private String query;
14     private PreparedStatement pst;
15     private ResultSet rs;
16
17     public ProductDao(Connection con) {
18         super();
19         this.con = con;
20     }
21
22
23     public List<Product> getAllProducts() {
24         List<Product> book = new ArrayList<>();
25         try {
26
27             query = "select * from products";
28             pst = this.con.prepareStatement(string: query);
29             rs = pst.executeQuery();
30
31             while (rs.next()) {
32                 Product row = new Product();
33                 row.setId(id: rs.getInt(string: "id"));
34                 row.setName(name: rs.getString(string: "name"));
35                 row.setCategory(category: rs.getString(string: "category"));
36                 row.setPrice(price: rs.getDouble(string: "price"));
37             }
38         }
39     }
40 }
```

```
ProductDao.java x
Source History
50
51 public Product getSingleProduct(int id) {
52     Product row = null;
53     try {
54         query = "select * from products where id=? ";
55
56         pst = this.con.prepareStatement(string:query);
57         pst.setInt(1, id);
58         ResultSet rs = pst.executeQuery();
59
60         while (rs.next()) {
61             row = new Product();
62             row.setId(id: rs.getInt(string:"id"));
63             row.setName(name: rs.getString(string:"name"));
64             row.setCategory(category: rs.getString(string:"category"));
65             row.setPrice(price: rs.getDouble(string:"price"));
66             row.setImage(image: rs.getString(string:"image"));
67         }
68     } catch (SQLException e) {
69         System.out.println(x: e.getMessage());
70     }
71
72     return row;
73 }
74
75 public double getTotalCartPrice(ArrayList<Cart> cartList) {
76     double sum = 0;
77     try {
78         if (!cartList.isEmpty()) {
79             for (Cart item : cartList) {
80                 query = "select price from products where id=?";
81                 pst = this.con.prepareStatement(string:query);
82                 pst.setInt(1, item.getId());
83                 rs = pst.executeQuery();
84                 while (rs.next()) {
85                     sum+=rs.getDouble(string:"price")*item.getQuantity();
86                 }
87             }
88         }
89     } catch (SQLException e) {
90         System.out.println(x: e.getMessage());
91     }
92     return sum;
93 }
94
95
96
97
98 public List<Cart> getCartProducts(ArrayList<Cart> cartList) {
99     List<Cart> book = new ArrayList<>();
100     try {
101         if (!cartList.isEmpty()) {
102             for (Cart item : cartList) {
103                 query = "select * from products where id=?";
104                 pst = this.con.prepareStatement(string:query);
105                 pst.setInt(1, item.getId());
106                 rs = pst.executeQuery();
107                 while (rs.next()) {
108                     Cart row = new Cart();
109                     row.setId(id: rs.getInt(string:"id"));
110                     row.setName(name: rs.getString(string:"name"));
111                     row.setCategory(category: rs.getString(string:"category"));
112                     row.setPrice(rs.getDouble(string:"price")*item.getQuantity());
113                     row.setQuantity(quantity: item.getQuantity());
114                     book.add(e: row);
115                 }
116             }
117         }
118     } catch (SQLException e) {
119         System.out.println(x: e.getMessage());
120     }
121     return book;
122 }
123
124
125
126
```

Product-related database activities are handled by the ProductDao class in the cn.develop.dao package. During instantiation, it accepts a Connection object as an argument, which makes database interaction easier. The class contains methods for handling product-related data and running SQL queries.

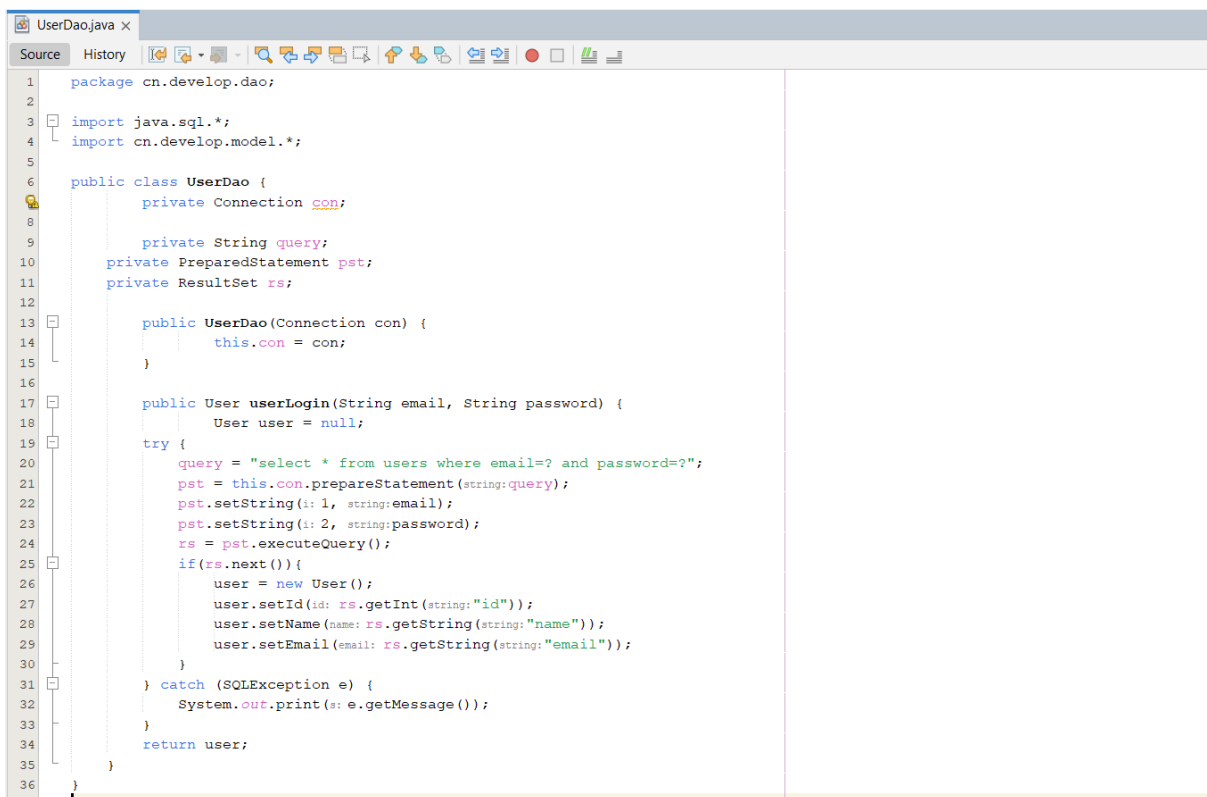


The given Java code is a component of the cn.develop.dao package's ProductDao class, which manages database operations for products in an e-commerce system. The class has methods for calculating the total cost of the goods in a shopping cart, retrieving all products, fetching a specific product by its ID, and obtaining comprehensive product information.

- A list of Product objects containing characteristics like ID, name, category, price, and image is created by the getAllProducts method, which pulls every product from the database.
- The getSingleProduct function returns a single Product object after retrieving a particular product from the database using its ID.
- The getTotalCartPrice function queries the database for each item's price and sums it up to get the total cost of the items in a shopping cart.

A list of Cart objects containing additional information, including computed price and quantity, is created by the getCartProducts method, which obtains comprehensive data about the items in the shopping cart. The code generates error warnings and handles SQL errors correctly.

## UserDao.Java Class:



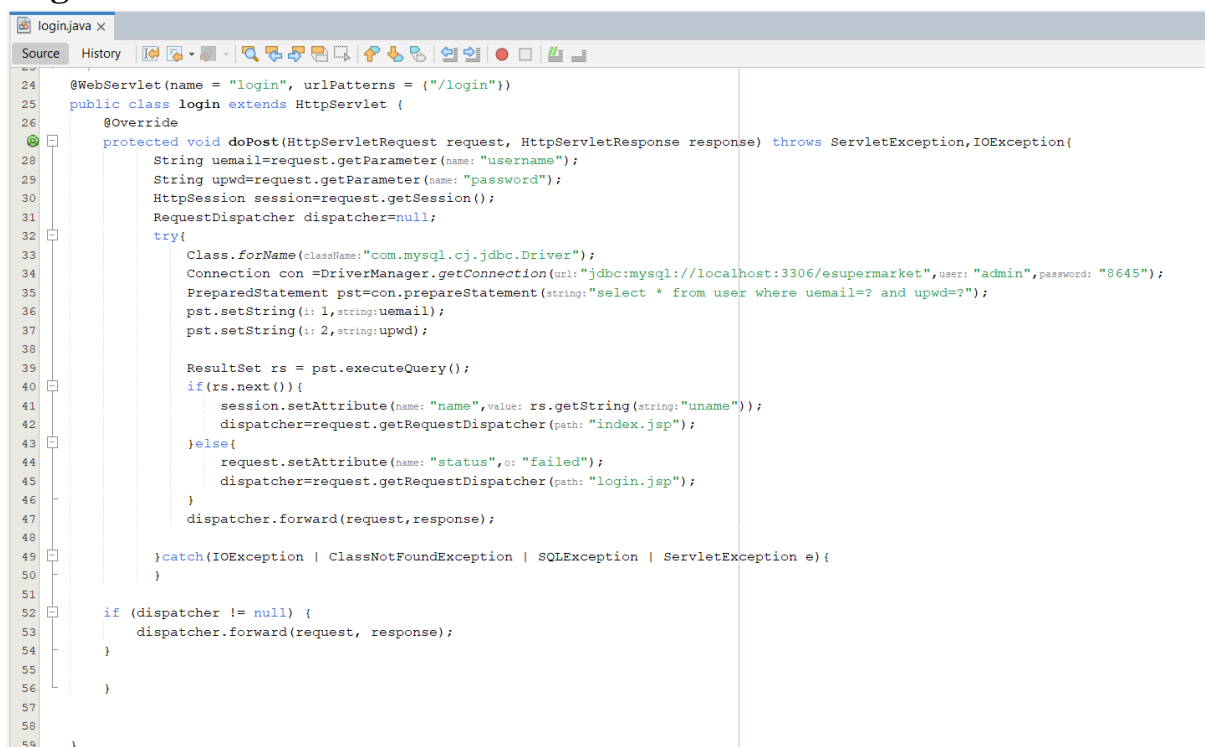
```
1 package cn.develop.dao;
2
3 import java.sql.*;
4 import cn.develop.model.*;
5
6 public class UserDao {
7     private Connection con;
8
9     private String query;
10    private PreparedStatement pst;
11    private ResultSet rs;
12
13    public UserDao(Connection con) {
14        this.con = con;
15    }
16
17    public User userLogin(String email, String password) {
18        User user = null;
19        try {
20            query = "select * from users where email=? and password=?";
21            pst = this.con.prepareStatement(string:query);
22            pst.setString(1, string:email);
23            pst.setString(2, string:password);
24            rs = pst.executeQuery();
25            if(rs.next()){
26                user = new User();
27                user.setId(id: rs.getInt(string:"id"));
28                user.setName(name: rs.getString(string:"name"));
29                user.setEmail(email: rs.getString(string:"email"));
30            }
31        } catch (SQLException e) {
32            System.out.print(s: e.getMessage());
33        }
34        return user;
35    }
36 }
```

The above Java code is a component of a UserDao class in the cn.develop.dao package that manages database activities pertaining to users, namely user authentication. The class has a userLogin function that, when called with email and password as inputs, searches the database for a user record that matches and, if it does, provides an instance of the user.

The SQL query looks for a user with the given email address and password in the "users" database. A new User object is constructed with characteristics like ID, name, and email, and is then returned if a matching record is discovered.

The code serves as an example of a basic yet crucial user authentication system that protects against SQL errors. It encourages modularity and maintainability in a bigger program by encapsulating user-related database operations.

## Login Servlet:



```
24 @WebServlet(name = "login", urlPatterns = {"/login"})
25 public class login extends HttpServlet {
26     @Override
27     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
28         String uemail=request.getParameter(name: "username");
29         String upwd=request.getParameter(name: "password");
30         HttpSession session=request.getSession();
31         RequestDispatcher dispatcher=null;
32         try{
33             Class.forName(className:"com.mysql.cj.jdbc.Driver");
34             Connection con =DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/esupermarket",user: "admin",password: "8645");
35             PreparedStatement pst=con.prepareStatement(string:"select * from user where uemail=? and upwd=?");
36             pst.setString(i: 1,string:uemail);
37             pst.setString(i: 2,string:upwd);
38
39             ResultSet rs = pst.executeQuery();
40             if(rs.next()){
41                 session.setAttribute(name: "name",value: rs.getString(string:"uname"));
42                 dispatcher=request.getRequestDispatcher(path: "index.jsp");
43             }else{
44                 request.setAttribute(name: "status",s: "failed");
45                 dispatcher=request.getRequestDispatcher(path: "login.jsp");
46             }
47             dispatcher.forward(request,response);
48         }catch(IOException | ClassNotFoundException | SQLException | ServletException e){
49         }
50
51         if (dispatcher != null) {
52             dispatcher.forward(request, response);
53         }
54     }
55 }
56
57
58
59 }
```

For user authentication, a servlet called "login" is defined using the Java code that is given. After connecting to a MySQL database and obtaining user credentials via a POST request, it looks for a matching user in the "user" table. In the event that login is unsuccessful, it sets a status property and returns to "login.jsp"; otherwise, it sets a session attribute and forwards to "index.jsp". Web-based user authentication is achieved through the use of Java Servlet technology in the code that wraps database connectivity and login functionality.

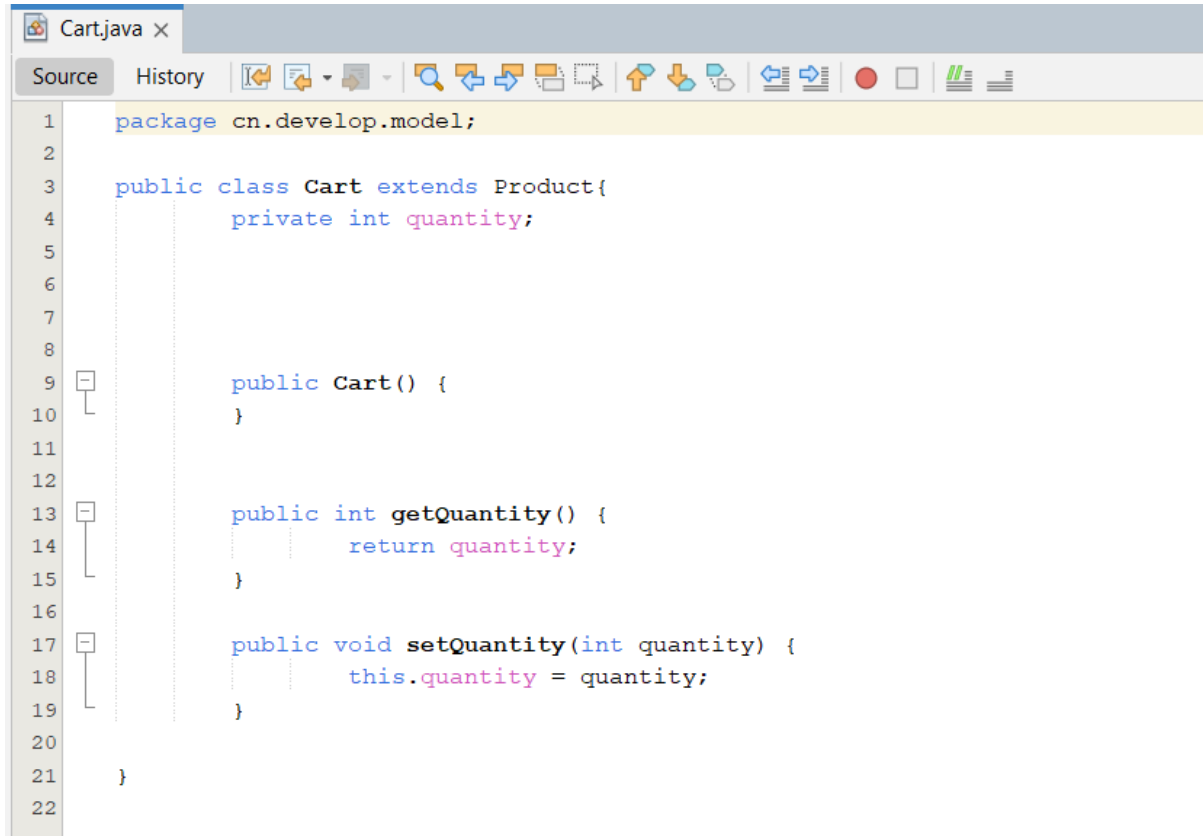
## Registration Servlet:

```
RegistrationServlet.java x
Source History
28 @WebServlet(name = "RegistrationServlet", urlPatterns = {"/Register"})
29 public class RegistrationServlet extends HttpServlet {
30
31     @Override
32     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
33
34         String uname=request.getParameter(name: "name");
35         String uemail=request.getParameter(name: "email");
36         String upwd=request.getParameter(name: "pass");
37         String umobile=request.getParameter(name: "contact");
38         RequestDispatcher dispatcher = null;
39         Connection con=null;
40
41         try{
42             Class.forName(className:"com.mysql.cj.jdbc.Driver");
43             con = DriverManager.getConnection(url:"jdbc:mysql://localhost:3306/esupermarket", user: "admin", password: "8645");
44             PreparedStatement pst = con.prepareStatement(string:"insert into user(uname,upwd,uemail,umobile) values(?,?,?,?)");
45             {
46                 pst.setString(i: 1,string:uname);
47                 pst.setString(i: 2,string:upwd);
48                 pst.setString(i: 3,string:uemail);
49                 pst.setString(i: 4,string:umobile);
50
51                 int rowCount = pst.executeUpdate();
52
53                 String userEmail=request.getParameter(name: "uemail");
54                 sendRegistrationEmail(userEmail);
55
56                 dispatcher=request.getRequestDispatcher(path: "registration.jsp");
57                 if(rowCount>0){
58                     request.setAttribute(name: "status",o: "success");
59                 }else{
60                     request.setAttribute(name: "status",o: "failed");
61                 }
62             }
63
64         }
65     }
66
67     private void sendRegistrationEmail(String userEmail) {
68         if (userEmail != null && !userEmail.isEmpty()) {
69             final String username = "greensupermarket56@gmail.com";
70             final String password = "supermarket81455682";
71
72             try {
73                 // Create the email
74                 HtmlEmail email = new HtmlEmail();
75                 email.setHostName(hostName: "smtp.gmail.com");
76                 email.setSmtpPort(portNumber: 587);
77                 email.setAuthenticator(new DefaultAuthenticator(userName: username, password));
78                 email.setStartTlsEnabled(startTlsEnabled: true);
79                 email.setFrom(email: username, name: "greensupermarket56@gmail.com");
80                 email.addTo(email: userEmail, name: "Recipient Name");
81                 email.setSubject(subject: "Registration Successful");
82
83                 // Set HTML content
84                 String htmlContent = "Dear User,<br><br>"
85                     + "Thank you for registering with GREEN Supermarket. "
86                     + "Your registration was successful. Welcome to our community!";
87                 email.setHtmlMsg(html: htmlContent);
88
89                 // Send the email
90                 email.send();
91
92                 System.out.println(x: "Email sent successfully!");
93             } catch (EmailException e) { // Log the exception
94                 // Log the exception
95                 System.out.println("Email sending failed: " + e.getMessage());
96             }
97         } else {
98             // Handle the case where userEmail is null or empty
99             System.out.println(x: "Email address is null or empty");
100         }
101     }
102 }
103
104 }
105
106 }
107
108 }
109
110 }
111 }
```

A web application's user registration capability is represented by the Java servlet code that is delivered. Through a POST request, the servlet obtains user registration information, including name, email, password, and phone number. After connecting to a MySQL database and entering user information, it uses the Apache Commons Email library to send an email confirming registration. An HTML welcome message is included in the email. Potential exceptions pertaining to database operations and email sending are handled by the code. In

general, this servlet offers a fundamental framework for user registration in a web application by integrating database storage, email notification, and user registration.

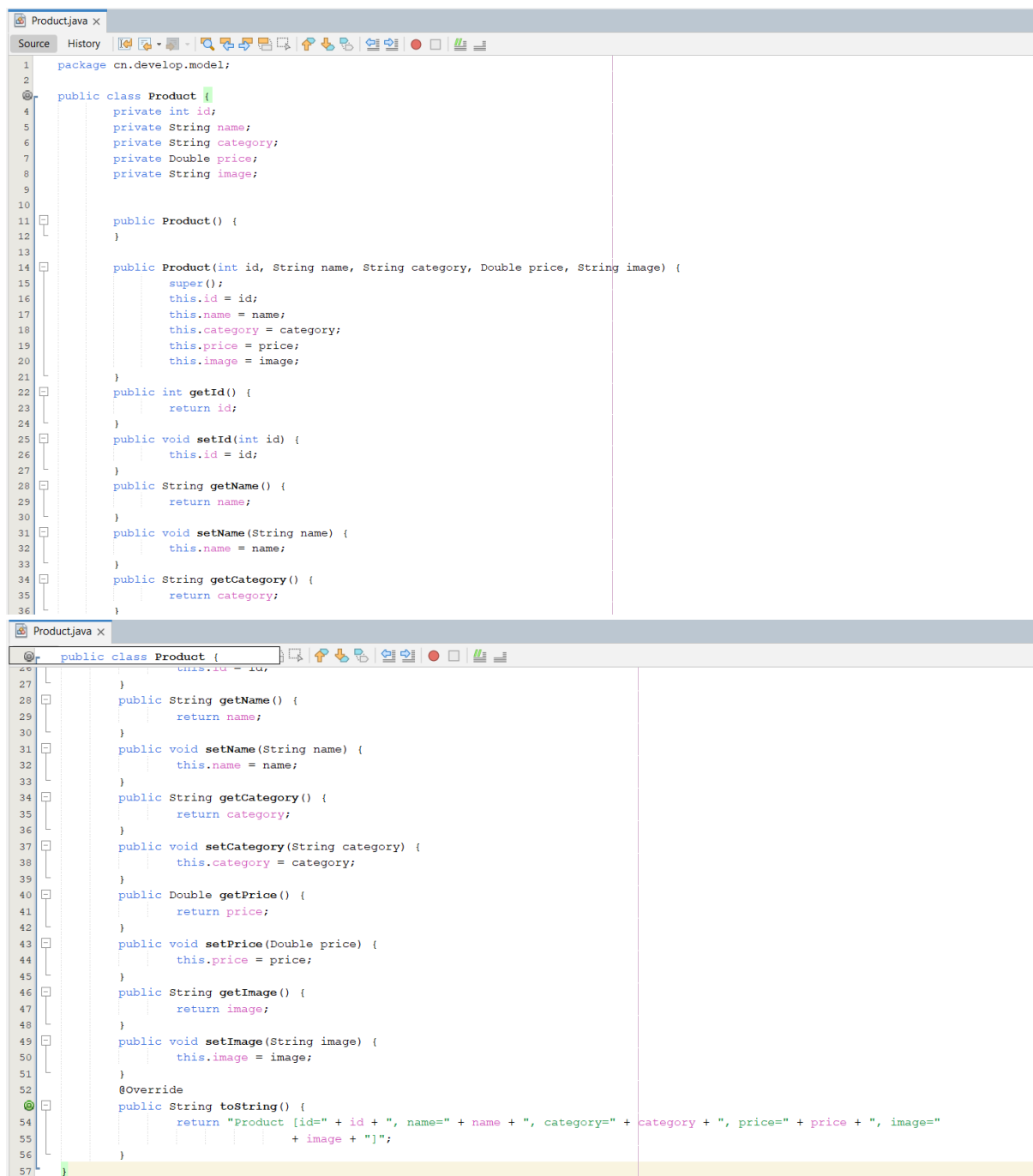
## Cart.Java Class:



```
1 package cn.develop.model;
2
3 public class Cart extends Product{
4     private int quantity;
5
6
7
8
9     public Cart() {
10    }
11
12
13    public int getQuantity() {
14        return quantity;
15    }
16
17    public void setQuantity(int quantity) {
18        this.quantity = quantity;
19    }
20
21 }
22
```

The "Product" class is extended by the given Java class, "Cart," indicating a connection between product representation and shopping cart functionality. The class adds a new attribute called "quantity," which probably indicates how much of a certain item has been added to the shopping cart. Within the context of a shopping cart, this design enables the recording of both product descriptions and their corresponding amounts. Cart objects may be instantiated by default without requiring any special initialization. This class, taken as a whole, is a component of a model that depicts a shopping cart system, integrating properties specific to products to make managing products and quantities in an e-commerce or related application easier.

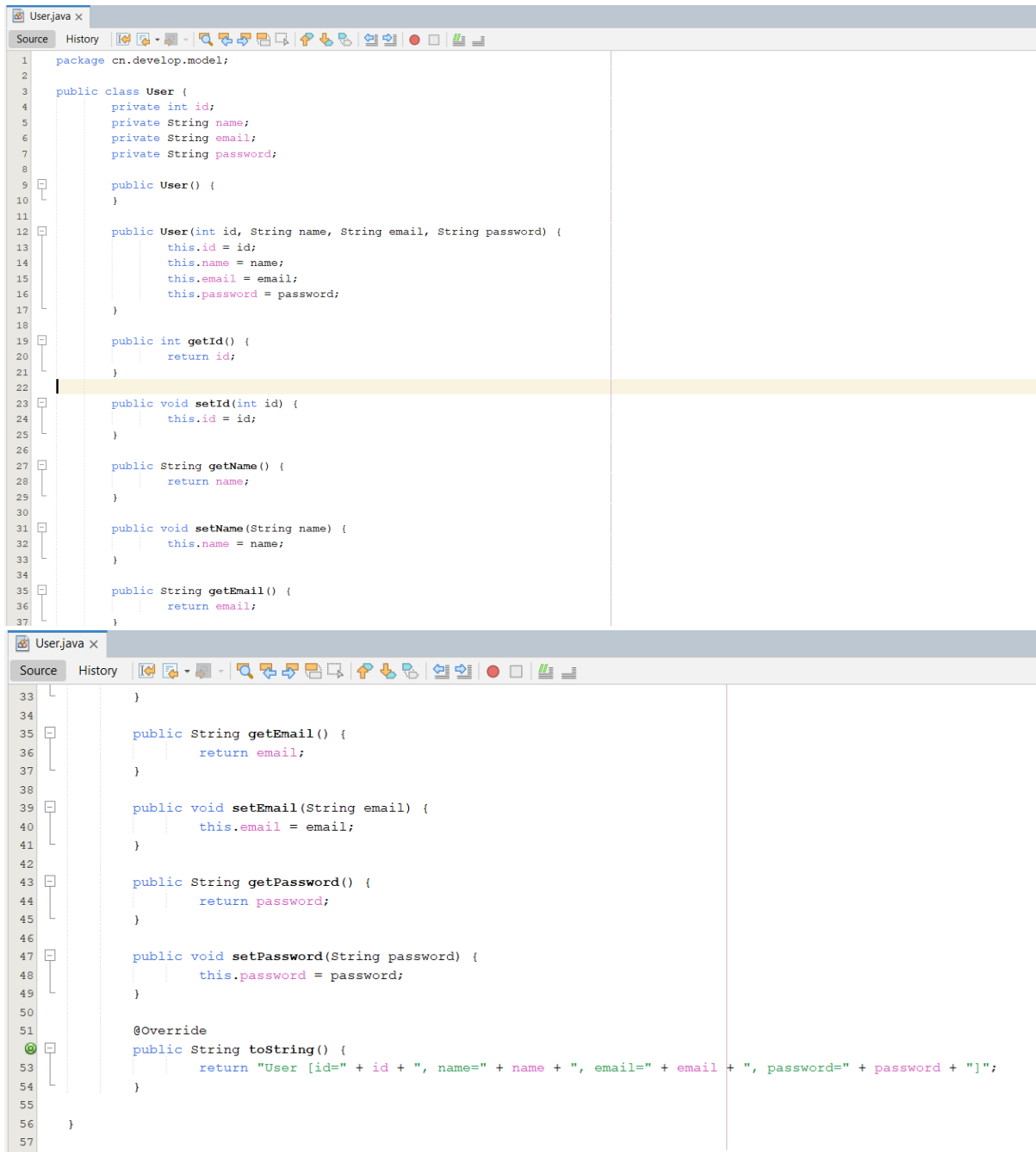
## Product.Java Class:



```
1 package cn.develop.model;
2
3
4 public class Product {
5     private int id;
6     private String name;
7     private String category;
8     private Double price;
9     private String image;
10
11     public Product() {
12     }
13
14     public Product(int id, String name, String category, Double price, String image) {
15         super();
16         this.id = id;
17         this.name = name;
18         this.category = category;
19         this.price = price;
20         this.image = image;
21     }
22     public int getId() {
23         return id;
24     }
25     public void setId(int id) {
26         this.id = id;
27     }
28     public String getName() {
29         return name;
30     }
31     public void setName(String name) {
32         this.name = name;
33     }
34     public String getCategory() {
35         return category;
36     }
37
38     public String getId() {
39         return id;
40     }
41     public void setId(String id) {
42         this.id = id;
43     }
44     public String getName() {
45         return name;
46     }
47     public void setName(String name) {
48         this.name = name;
49     }
50     public String getCategory() {
51         return category;
52     }
53     public void setCategory(String category) {
54         this.category = category;
55     }
56     public Double getPrice() {
57         return price;
58     }
59     public void setPrice(Double price) {
60         this.price = price;
61     }
62     public String getImage() {
63         return image;
64     }
65     public void setImage(String image) {
66         this.image = image;
67     }
68     @Override
69     public String toString() {
70         return "Product [id=" + id + ", name=" + name + ", category=" + category + ", price=" + price + ", image="
71             + image + "]\n";
72     }
73 }
```

An example of modeling product entities in a system is the supplied Java class "Product," which is probably going to be used in an inventory management or e-commerce application. It contains all of the important product details, including "id," "name," "category," "price," and "image." The class has two constructors: one parameterized for initializing the attributes during object creation, and the other default for instantiating objects. To create a string representation of the product for logging or debugging, the "toString" function is modified. Encapsulation is adhered to by this class, which offers getter and setter methods for accessing and changing its characteristics.

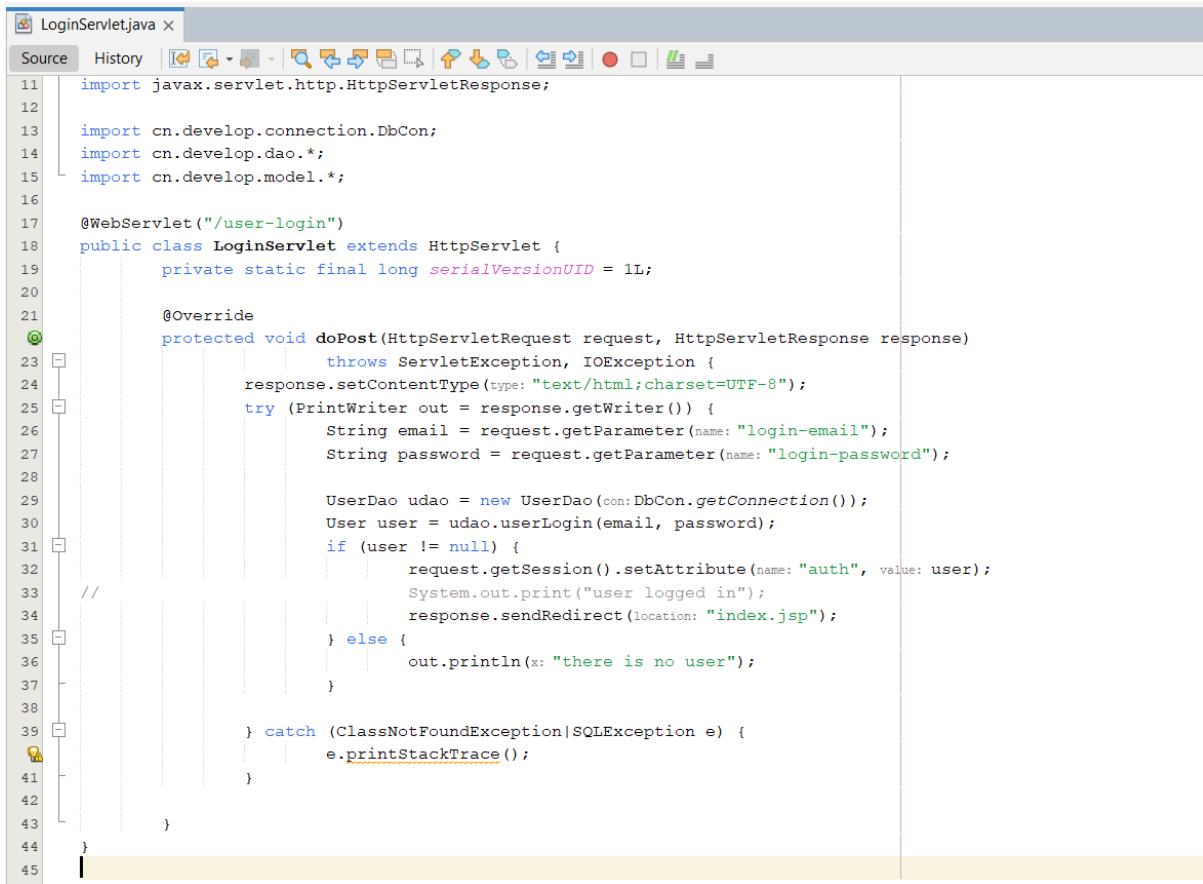
## User.Java Class:



```
1 package cn.develop.model;
2
3 public class User {
4     private int id;
5     private String name;
6     private String email;
7     private String password;
8
9     public User() {
10    }
11
12    public User(int id, String name, String email, String password) {
13        this.id = id;
14        this.name = name;
15        this.email = email;
16        this.password = password;
17    }
18
19    public int getId() {
20        return id;
21    }
22
23    public void setId(int id) {
24        this.id = id;
25    }
26
27    public String getName() {
28        return name;
29    }
30
31    public void setName(String name) {
32        this.name = name;
33    }
34
35    public String getEmail() {
36        return email;
37    }
38
39    public void setEmail(String email) {
40        this.email = email;
41    }
42
43    public String getPassword() {
44        return password;
45    }
46
47    public void setPassword(String password) {
48        this.password = password;
49    }
50
51    @Override
52    public String toString() {
53        return "User [id=" + id + ", name=" + name + ", email=" + email + ", password=" + password + "]\n";
54    }
55
56 }
57
```

The Java class "User," which is being shown, symbolizes a user object in a software system, frequently linked to user management and authentication features. It contains all of the necessary user information, such as the "id," "name," "email," and "password." The class has two constructors: one parameterized for initializing user properties during object creation, and the other default for instantiating objects. Encapsulation principles are followed via getter and setter methods, which grant regulated access to the class's private properties. To create a string representation of the user for debugging or logging, the "toString" function is modified.

## Login Servlet:



```
11 import javax.servlet.http.HttpServletResponse;
12
13 import cn.develop.connection.DbCon;
14 import cn.develop.dao.*;
15 import cn.develop.model.*;
16
17 @WebServlet("/user-login")
18 public class LoginServlet extends HttpServlet {
19     private static final long serialVersionUID = 1L;
20
21     @Override
22     protected void doPost(HttpServletRequest request, HttpServletResponse response)
23         throws ServletException, IOException {
24         response.setContentType(type: "text/html;charset=UTF-8");
25         try (PrintWriter out = response.getWriter()) {
26             String email = request.getParameter(name: "login-email");
27             String password = request.getParameter(name: "login-password");
28
29             UserDao udao = new UserDao(con: DbCon.getConnection());
30             User user = udao.userLogin(email, password);
31             if (user != null) {
32                 request.getSession().setAttribute(name: "auth", value: user);
33                 System.out.print("user logged in");
34                 response.sendRedirect(location: "index.jsp");
35             } else {
36                 out.println(x: "there is no user");
37             }
38
39         } catch (ClassNotFoundException | SQLException e) {
40             e.printStackTrace();
41         }
42     }
43 }
44
45
```

The LoginServlet class, which manages user login functionality for a web application, is defined by this Java Servlet code. The email and password arguments are extracted from a POST request when it is received. The userLogin function is then called using a UserDao object in an effort to authenticate the user. The user object is saved in the session property "auth," and they are forwarded to the "index.jsp" page if the authentication is successful. A notice stating that there is no user is printed if authentication is unsuccessful. Database operation exceptions are also handled by the code. All in all, it offers a web application's fundamental user login feature.

## Logout Servlet:

```
LogoutServlet.java x
Source History
5
6 import javax.servlet.ServletException;
7 import javax.servlet.annotation.WebServlet;
8 import javax.servlet.http.HttpServlet;
9 import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12
13 @WebServlet("/log-out")
14 public class LogoutServlet extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17
18     @Override
19     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
20         response.setContentType("text/html;charset=UTF-8");
21         try (PrintWriter out = response.getWriter()) {
22             // TODO Auto-generated method stub
23             response.sendRedirect(location: "login.jsp");
24         }
25     }
26
27
28     @Override
29     protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
30         // TODO Auto-generated method stub
31         doGet(request, response);
32     }
33
34 }
35
36
37
```

The LogoutServlet class in this Java Servlet code is in charge of managing user logout in a web application. After receiving a GET request, the doGet method changes the response content type and sends the user to the "login.jsp" page. The code simulates a logout action by making sure the user session is ended successfully. All that the doPost function does is pass calls to the doGet method. All things considered, this servlet offers a simple method of logging users out by sending them to the login page and ending their session.



## Review Servlet:

```
ReviewServlet.java x
Source History
53
54 @Override
55 protected void doPost(HttpServletRequest request, HttpServletResponse response)
56     throws ServletException, IOException {
57     String username = request.getParameter(name: "username");
58     String message = request.getParameter(name: "message");
59
60     // Store the data in the database
61     storeData(username, message);
62
63     // Prepare HTML content for the newly submitted message
64     String newMessageHtml = "<div class=\"client-box\">\n"
65         + "    <div class=\"client-profile\">\n"
66         + "        <div class=\"profile-text\">\n"
67         + "            <strong>" + username + "</strong>\n"
68         + "        </div>\n"
69         + "    </div>\n"
70         + "    <p>" + message + "</p>\n"
71         + "</div>\n";
72
73     // Send the HTML content back to the client
74     response.setContentType(type: "text/html;charset=UTF-8");
75     try (PrintWriter out = response.getWriter()) {
76         out.write(s: newMessageHtml);
77     }
78 }
79

ReviewServlet.java x
Source History
80 private void storeData(String username, String message) {
81     Connection connection = null;
82     PreparedStatement preparedStatement = null;
83
84     try {
85         // Assuming you have a database named "your_database" and a table named "reviews"
86         String url = "jdbc:mysql://localhost:3306/esupermarket";
87         String dbUser = "admin";
88         String dbPassword = "8645";
89
90         Class.forName(className: "com.mysql.cj.jdbc.Driver");
91         connection = DriverManager.getConnection(url, user: dbUser, password: dbPassword);
92
93         // Insert data into the "reviews" table
94         String query = "INSERT INTO reviews (username, message) VALUES (?, ?)";
95         preparedStatement = connection.prepareStatement(string: query);
96         preparedStatement.setString(i: 1, string: username);
97         preparedStatement.setString(i: 2, string: message);
98         preparedStatement.executeUpdate();
99     } catch (ClassNotFoundException | SQLException e) {
100     } finally {
101         try {
102             if (preparedStatement != null) {
103                 preparedStatement.close();
104             }
105             if (connection != null) {
106                 connection.close();
107             }
108         } catch (SQLException e) {
109         }
110     }
111 }
112
113 }
```

ReviewServlet is a Java Servlet that manages review submissions for a web application. The username and message arguments are retrieved from the HTTP request by the doPost method. The data is then stored in a MySQL database called "esupermarket" in the "reviews" table by using the storeData function. The code contains the login credentials (password, username, and URL) for the database connection.

The "reviews" table is updated with the username and message by the storeData method, which also closes the database resources after establishing a JDBC connection to the database. In order to handle possible exceptions during database operations, the code additionally provides error handling.

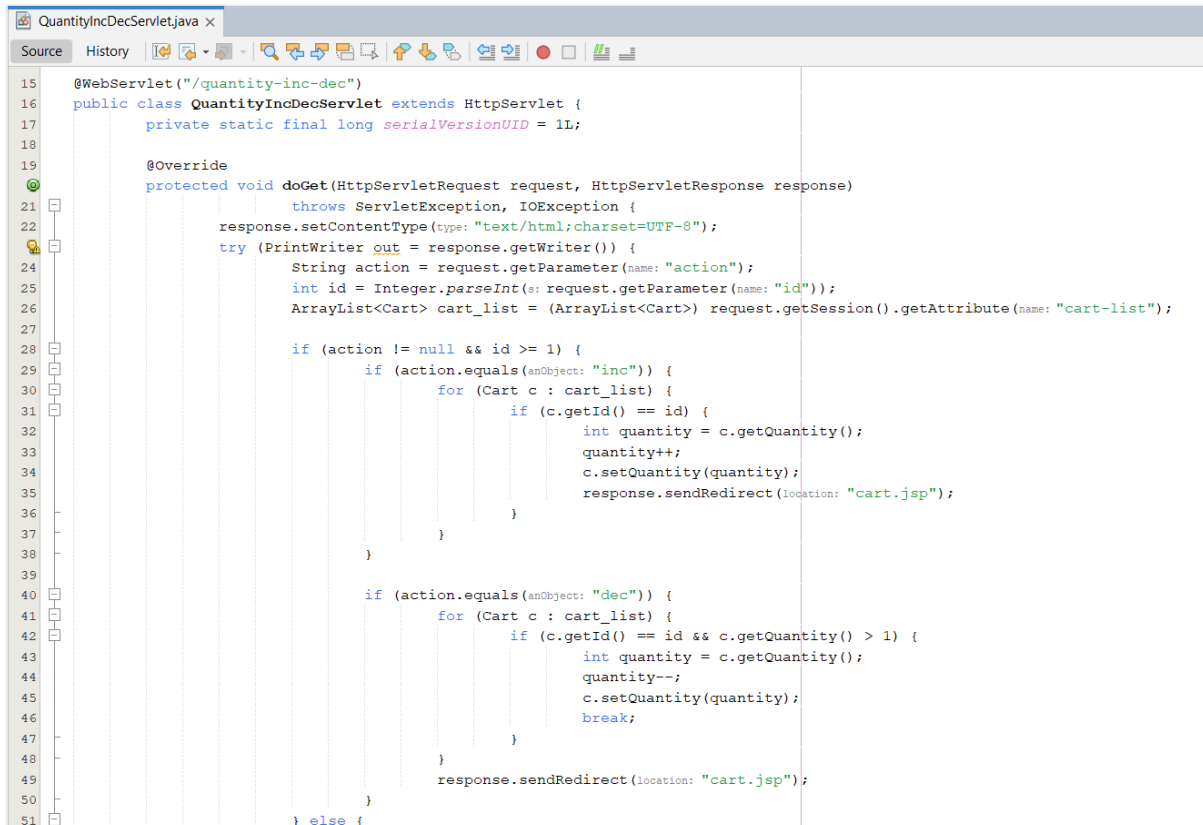
## AddToCart Servlet:

```
17 @WebServlet(name = "AddToCartServlet", urlPatterns = "/add-to-cart")
18 public class AddToCartServlet extends HttpServlet {
19     private static final long serialVersionUID = 1L;
20
21     @Override
22     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
23         response.setContentType("text/html;charset=UTF-8");
24
25         try (PrintWriter out = response.getWriter()) {
26             //
27
28             ArrayList<Cart> cartList = new ArrayList<>();
29             int id = Integer.parseInt(request.getParameter("id"));
30             Cart cm = new Cart();
31             cm.setId(id);
32             cm.setQuantity(quantity: 1);
33             HttpSession session = request.getSession();
34             ArrayList<Cart> cart_list = (ArrayList<Cart>) session.getAttribute(name: "cart-list");
35             if (cart_list == null) {
36                 cartList.add(cm);
37                 session.setAttribute(name: "cart-list", value: cartList);
38                 response.sendRedirect(location: "product.jsp");
39             } else {
40                 cartList = cart_list;
41
42                 boolean exist = false;
43                 for (Cart c : cart_list) {
44                     if (c.getId() == id) {
45                         exist = true;
46                         out.println(x: "<h3 style='color:crimson; text-align: center;'>Item Already in Cart. <a href='cart.jsp'>GO to Cart Page</a></h3>");
47                     }
48                 }
49
50                 if (!exist) {
51                     cartList.add(cm);
52                     response.sendRedirect(location: "product.jsp");
53                 }
54             }
55         }
56     }
57 }
```

The AddToCartServlet Java Servlet controls the process of adding products to a web application's shopping cart. The product ID parameter is retrieved from the request and the return content type is first specified in the doGet function. Next, it checks to see whether there is already a shopping cart in the user's session before creating a Cart object with a quantity of 1.

The product is added, a new cart is created if one doesn't already exist, and the user is sent to the "product.jsp" page. It determines whether the product is already in the cart if one already exists. If that's the case, it notifies the user; if not, it places the item in the basket and sends them to the "product.jsp" page. Information about the cart is kept in the user's session.

## QuantityIncDec Servlet:

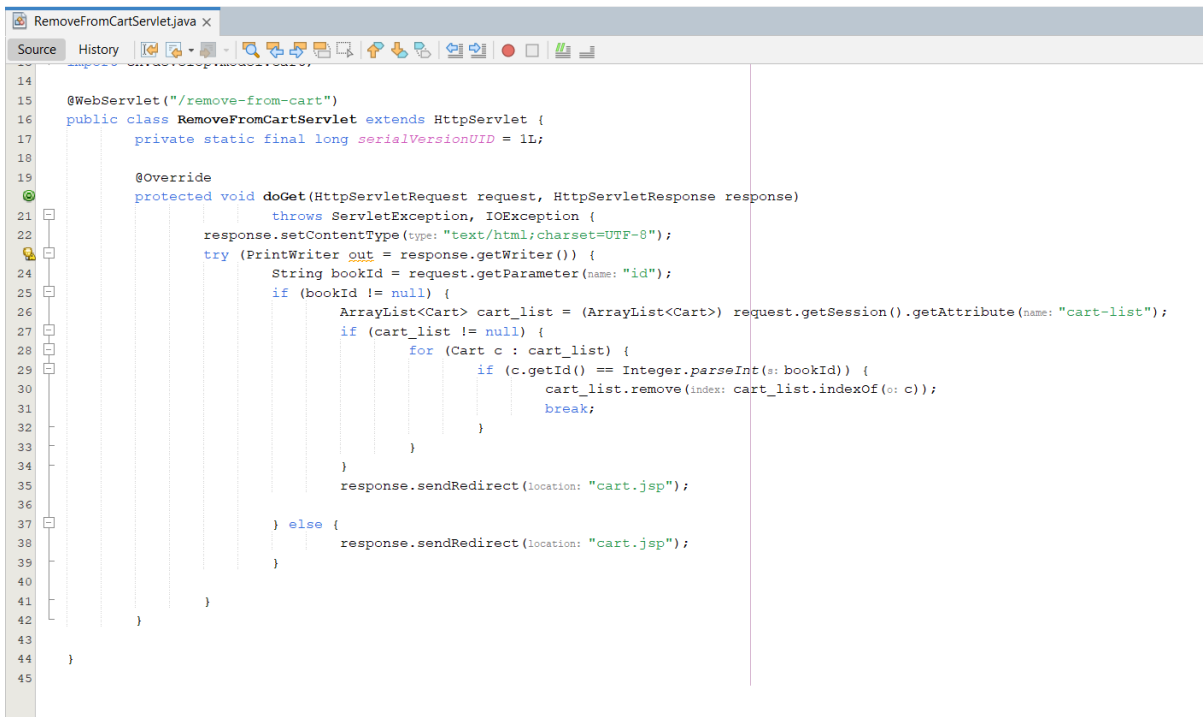


```
15 @WebServlet("/quantity-inc-dec")
16 public class QuantityIncDecServlet extends HttpServlet {
17     private static final long serialVersionUID = 1L;
18
19     @Override
20     protected void doGet(HttpServletRequest request, HttpServletResponse response)
21         throws ServletException, IOException {
22         response.setContentType("text/html;charset=UTF-8");
23         try (PrintWriter out = response.getWriter()) {
24             String action = request.getParameter("action");
25             int id = Integer.parseInt(request.getParameter("id"));
26             ArrayList<Cart> cart_list = (ArrayList<Cart>) request.getSession().getAttribute("cart-list");
27
28             if (action != null && id >= 1) {
29                 if (action.equals("inc")) {
30                     for (Cart c : cart_list) {
31                         if (c.getId() == id) {
32                             int quantity = c.getQuantity();
33                             quantity++;
34                             c.setQuantity(quantity);
35                             response.sendRedirect("cart.jsp");
36                         }
37                     }
38                 }
39                 if (action.equals("dec")) {
40                     for (Cart c : cart_list) {
41                         if (c.getId() == id && c.getQuantity() > 1) {
42                             int quantity = c.getQuantity();
43                             quantity--;
44                             c.setQuantity(quantity);
45                             break;
46                         }
47                     }
48                     response.sendRedirect("cart.jsp");
49                 }
50             }
51         } else {
```

The QuantityIncDecServlet Java Servlet controls the addition and subtraction of item amounts in a web application's shopping cart. The doGet function receives the action and product ID parameters from the request, sets the return content type, and gets the current shopping cart from the user's session.

The quantity of the designated product in the cart is increased if the action is marked as "inc" (increment). The quantity is decreased if the action is "dec" (decrement), making sure it doesn't drop below 1. The user is subsequently sent to the "cart.jsp" page when the modified cart data has been saved in the session. The user gets forwarded to the "cart.jsp" page if the action is incorrect or the product ID is less than 1.

## RemoveFromCart Servlet:



```
14
15 @WebServlet("/remove-from-cart")
16 public class RemoveFromCartServlet extends HttpServlet {
17     private static final long serialVersionUID = 1L;
18
19     @Override
20     protected void doGet(HttpServletRequest request, HttpServletResponse response)
21         throws ServletException, IOException {
22         response.setContentType("text/html;charset=UTF-8");
23         try (PrintWriter out = response.getWriter()) {
24             String bookId = request.getParameter("id");
25             if (bookId != null) {
26                 ArrayList<Cart> cart_list = (ArrayList<Cart>) request.getSession().getAttribute("cart-list");
27                 if (cart_list != null) {
28                     for (Cart c : cart_list) {
29                         if (c.getId() == Integer.parseInt(bookId)) {
30                             cart_list.remove(index: cart_list.indexOf(c));
31                             break;
32                         }
33                     }
34                 }
35                 response.sendRedirect(location: "cart.jsp");
36             } else {
37                 response.sendRedirect(location: "cart.jsp");
38             }
39         }
40     }
41 }
42
43
44
45
```

The RemoveFromCartServlet Java Servlet manages the process of removing products from a web application's shopping cart. The product ID parameter is retrieved from the request, the current shopping cart is obtained from the user's session, and the return content type is specified in the doGet function.

Iterating through the cart items, it locates the matched product and deletes it from the list if the product ID is supplied. The user is subsequently sent to the "cart.jsp" page when the modified cart data has been saved in the session. The user is sent straight to the "cart.jsp" page if the product ID is not supplied. The functionality for managing a user's shopping cart's contents is provided by this servlet.

# ProcessPaypalPayment Servlet:

```
ProcessPaypalPayment.java x
Source History
5 package cn.develop.servlet;
6
7 import com.paypal.api.payments.*;
8 import com.paypal.base.rest.*;
9
10 import java.io.IOException;
11 import java.util.ArrayList;
12 import java.util.List;
13 import javax.servlet.ServletException;
14 import javax.servlet.annotation.WebServlet;
15 import javax.servlet.http.HttpServlet;
16 import javax.servlet.http.HttpServletRequest;
17 import javax.servlet.http.HttpServletResponse;
18
19 /**
20 *
21 * @author asust
22 */
23 @WebServlet(name = "ProcessPaypalPayment", urlPatterns = {"/processPayPalPayment"})
24 public class ProcessPaypalPayment extends HttpServlet {
25
26     @Override
27     protected void doPost(HttpServletRequest request, HttpServletResponse response)
28         throws ServletException, IOException {
29
30         String clientId = "AS3T7t0N_xWqwsj0qID2lCSVocGoalir4JDPgyYzwiXFpoOUxE1CMIJqS6zydfkSnTg5T4z1vdXapHPm";
31         String clientSecret = "ECfzEEYwlyOzUnl00qWN525EicILFbXjgAqUJ6U_kCSaO7SxmbHWIzHkFP4nADmG2ZHHCaNnhyLh1hKh";
32
33         APIContext apiContext = new APIContext(clientId: clientId, clientSecret, mode: "sandbox");
34
35         // Get amount from the form submission
36         String amount = request.getParameter(name: "amount");
37
38         // Create a Payment object and set up the transaction details
39         Amount transactionAmount = new Amount();
40         transactionAmount.setCurrency(currency: "USD");
41         transactionAmount.setTotal(total: amount);
42
43         Transaction transaction = new Transaction();
44         transaction.setDescription(description: "Payment for your product or service");
45         transaction.setAmount(amount: transactionAmount);
46
47         List<Transaction> transactions = new ArrayList<>();
48         transactions.add(e: transaction);
49
50         Payer payer = new Payer();
51         payer.setPaymentMethod(paymentMethod: "paypal");
52
53         Payment payment = new Payment();
54         payment.setIntent(intent: "sale");
55         payment.setPayer(payer);
56         payment.setTransactions(transactions);
57
58         // Redirect URLs for PayPal to redirect the user after the payment
59         RedirectUrls redirectUrls = new RedirectUrls();
60         redirectUrls.setReturnUrl(returnUrl: "index.jsp");
61         redirectUrls.setCancelUrl(cancelUrl: "billing.jsp");
62         payment.setRedirectUrls(redirectUrls);
63
64         // Create the payment
65         try {
66             Payment createdPayment = payment.create(apiContext);
67
68             // Get the approval URL
69             String approvalURL = createdPayment.getLinks().get(index: 1).getHref();
70
71             // Redirect the user to PayPal for approval
72             response.sendRedirect(location: approvalURL);
73         } catch (PayPalRESTException e) {
74             // Handle exception (log it or display an error message)
75             response.getWriter().write(s: "Error processing PayPal payment.");
76         }
77     }
78 }
79
```

```

81     }
82
83     String userEmail = request.getParameter(name: "userEmail");
84
85
86     if (userEmail != null && !userEmail.isEmpty()) {
87
88         Properties properties = new Properties();
89         properties.put(key: "mail.smtp.host", value: "smtp.gmail.com");
90         properties.put(key: "mail.smtp.port", value: "587");
91         properties.put(key: "mail.smtp.auth", value: "true");
92         properties.put(key: "mail.smtp.starttls.enable", value: "true");
93
94         final String username = "greensupermarket56@gmail.com";
95         final String password = "numehmifeaqalesb";
96
97         Session session = Session.getInstance(props: properties, new Authenticator() {
98             @Override
99             protected PasswordAuthentication getPasswordAuthentication() {
100                 return new PasswordAuthentication(username, password);
101             }
102         });
103
104         try {
105
106             MimeMessage message = new MimeMessage(session);
107
108             message.setFrom(new InternetAddress(address: username));
109
110             message.addRecipient(type: Message.RecipientType.TO, new InternetAddress(address: userEmail));
111
112             message.setSubject(subject: "Payment Confirmation");
113
114             message.setText(text: "Thank you for your payment. Your order is confirmed.");
115
116             Transport.send(msg: message);
117         } catch (MessagingException e) {

```

The PayPal payment processing for a web application is made easier by this Java Servlet called `ProcessPaypalPayment`. The client ID and secret are set up together with the PayPal API credentials in the `doPost` function. After that, an `APIContext` object is created so that API calls may be made in the PayPal sandbox. The payment amount is retrieved by the servlet from the submitted form.

It creates a `Payment` object containing the transaction details—amount, currency, and description—by utilizing the PayPal API. PayPal is listed as the payment option, and redirect URLs are put up for both successful and unsuccessful payments.

The servlet makes an effort to use the PayPal API to create a payment. If it is successful, the user is sent to PayPal for approval once the approval URL is retrieved from the made payment. An error message is written to the response in the event that an exception occurs during the PayPal API call (which is handled by `PayPalRESTException`).

All things considered, this servlet combines with the PayPal API to start a payment process, enabling users to safely utilize PayPal to make payments within the web application. By redirecting the user to PayPal's approval page, the application's e-commerce capability is improved.

## References

- *Font awesome* (no date) *Font Awesome*. Available at: <https://fontawesome.com/>  
(Accessed: 30 December 2023).
- (No date) *Google fonts / google for developers*. Available at:  
<https://developers.google.com/fonts> (Accessed: 30 December 2023).
- (No date a) *Maven Repository: Central*. Available at:  
<https://mvnrepository.com/repos/central> (Accessed: 30 December 2023).

## Work-Load Matrix

	<b>Plymouth ID</b>	<b>Name (As appeared on DLE)</b>	<b>Contributed section</b>
<b>1</b>	<b>10899603</b>	<b>Thisara Madusanka</b>	<ul style="list-style-type: none"><li>• <b>Final Report Finalization.</b></li><li>• <b>Create product.jsp page and relevant configurations using Servlet files.</b></li><li>• <b>Create product.jsp page and relevant configurations using Servlet files.</b></li><li>• <b>Connect Paypal Payment system through Sandbox.</b></li></ul>
<b>2</b>	<b>10899621</b>	<b>Chathupraba Munasinghe</b>	<ul style="list-style-type: none"><li>• <b>Working with Database Configurations and Connections</b></li><li>• <b>Create review.jsp page and relevant configurations using servlet files</b></li></ul>
<b>3</b>	<b>10899521</b>	<b>Navindu Nimsara</b>	<ul style="list-style-type: none"><li>• <b>Drawing Er Diagram</b></li><li>• <b>Create index.jsp page and about.jsp pages.</b></li><li>• <b>Connect Email Confirmation Service and relevant configurations</b></li></ul>
<b>4</b>	<b>10899685</b>	<b>Kihaduwege Sahasra</b>	<ul style="list-style-type: none"><li>• <b>Drawing Class Diagram</b></li><li>• <b>Create login.jsp page and relevant configurations using servlet files</b></li></ul>
<b>5</b>	<b>10899600</b>	<b>Senanayake Liyanage</b>	<ul style="list-style-type: none"><li>• <b>Drawing Use case Diagram</b></li><li>• <b>Create registration.jsp page and relevant configurations using servlet files</b></li></ul>
<b>6</b>	<b>10899556</b>	<b>Kishal Sankalpa Jayalath</b>	<ul style="list-style-type: none"><li>• <b>Drawing Sequence Diagram</b></li><li>• <b>Create cart.jsp page and relevant configurations using servlet files</b></li></ul>