

E-Commerce Application Documentation

E-Mart

Website link : <http://40.82.152.93/>



About the project

We proudly introduce E-mart, our first E-Commerce application built with Java technologies. Our project demonstrates our dedication to creating a seamless electronic shopping experience. Here's how we launched E-mart:

- **User-Centric Profiles:** To accommodate individual preferences, we've implemented customized user accounts.
- **Extensive Product Catalog:** With our platform's wide selection of electronic products, users can easily find what they need.
- **Effortless Cart Management:** For a seamless shopping experience, we've simplified the process of adding and removing items from the cart.
- **Secure Transaction Process:** Ensuring secure transactions is our top priority in order to give customers peace of mind when they check out.
- **Real-Time Order Tracking:** Customers can follow the progress of their orders in real time and remain informed at all times.
- **Comprehensive Order History:** For customers' convenience, we've included a thorough rundown of previous purchases.
- **Direct Support Channel:** Users can contact us directly at any time to request help via our messaging system.
- **Automated Email Confirmations:** Upon checkout, users receive instant order confirmation emails for their convenience.
- **Feedback Integration:** To continuously develop and improve the platform, we actively welcome user feedback.

Content

About the project	2
Emart Website.....	6
1.1 Home page	6
1.1.1 Web page	6
1.1.2 Documentation: Home Page Implementation.....	9
1.1.3 Codes.....	11
1.2 Order Confirmation page	22
1.2.1 Web page	22
1.2.2 Documentation: Order Confirmation Page Implementation.....	24
1.2.3 Codes.....	27
1.3 Sign Up page	36
1.3.1 Web page	36
1.3.2 Documentation: Sign Up Page Implementation	37
1.3.3 Codes.....	38
1.4 Sign In page	44
1.4.1 Web page	44
1.4.2 Documentation : Sign In page implementation	44
1.4.3 Codes.....	45
1.5 Forgot Password page.....	51
1.5.1 Web page	51
1.5.2 Documentation : Forgot password page implementation.....	51
1.5.3 Codes.....	52
1.6 Verify OTP page.....	57
1.6.1 Web page	57
1.6.2 Documentation : VerifyOTP page implementation	57
1.6.3 Codes.....	58
1.7 Reset password page	60
1.7.1 Web page	60
1.7.2 Documentation : Reset password page implementation	60
1.7.3 Codes.....	61
1.8 Profile page	63

1.8.1 Web page	63
1.8.2 Documentation : Profile page implementation	63
1.8.3 Codes.....	64
1.9 About Us page.....	83
1.9.1 Web page	83
1.9.2 Documentation : About Us page implementation.....	84
1.9.3 Codes.....	85
1.10 Developers page	88
1.10.1 Web page	88
1.10.2 Documentation : Developers page implementation	89
1.10.3 Codes.....	90
1.11 Track Order page.....	93
1.11.1 Web page	93
1.11.2 Documentation : Track Order page implementation.....	94
1.11.3 Codes.....	95
1.12 Checkout page.....	100
1.12.1 Web page	100
1.12.2 Documentation : Checkout page implementation	101
1.12.3 Codes.....	103
1.13 Contact page	112
1.13.1 Web page	112
1.13.2 Documentation : Contact page implementation	112
1.13.3 Codes.....	114
1.14 Product Category page.....	122
1.14 Product Information page.....	126
1.15 Shopping Cart page	131
1.15.1 Web page	131
1.14.2 Documentation : Shopping Cart page implementation	131
1.14.3 Codes.....	135
Navbar.....	158
Footer.....	162
2. Admin Panel	165
2.1 Products page (index page).....	165
2.1.1 Web page	165

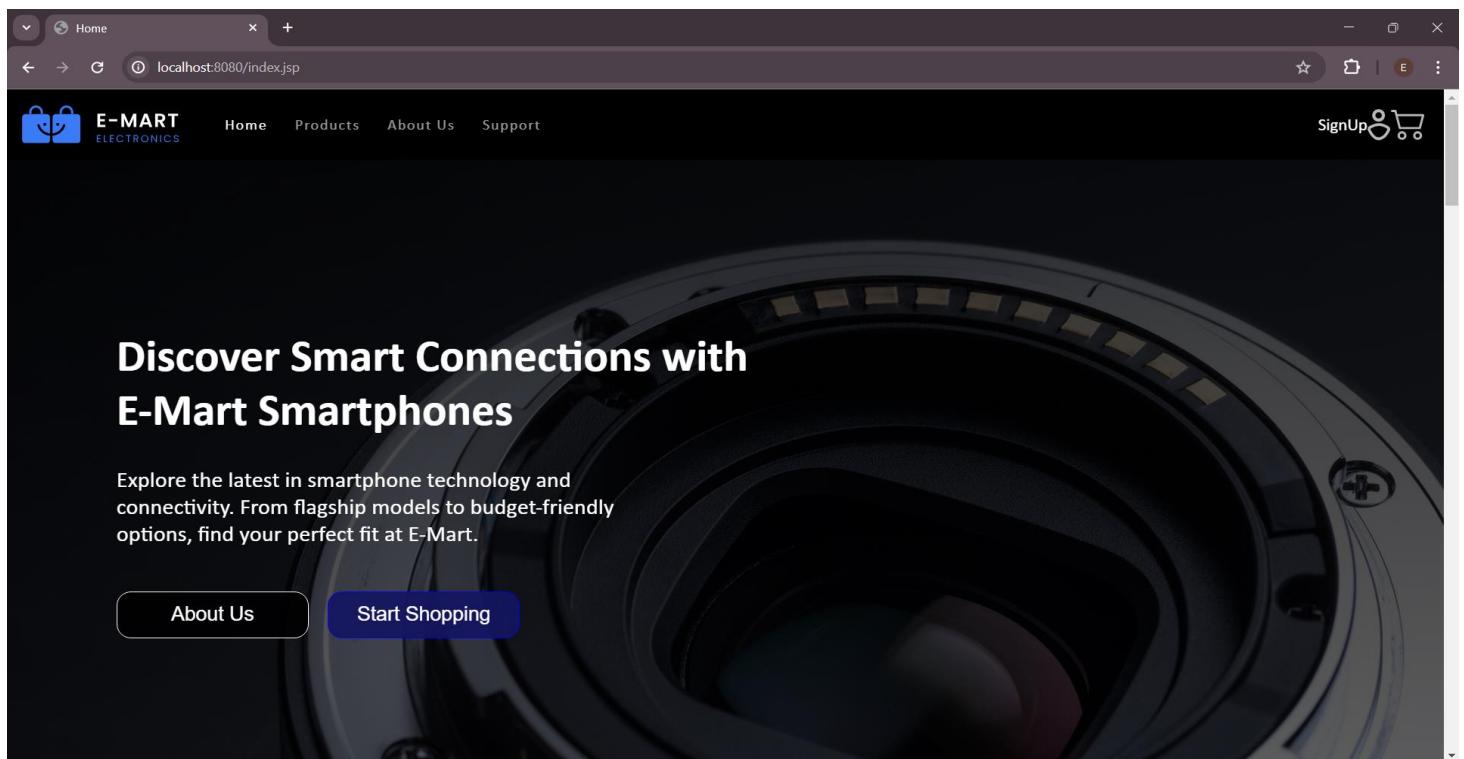
2.1.2 Documentation	167
2.1.3 Codes.....	169
2.2 Log In page	182
2.2.1 Web page.....	182
2.2.2 Documentation	183
2.2.3 Codes.....	183
2.3 Customers page	189
2.3.1 Web page	189
2.3.2 Documentation	189
2.3.3 Codes.....	190
2.4 Admin Orders page	198
2.4.1 Web page	198
2.4.2 Documentation	198
2.4.3 Codes.....	199
Contribution of team members	203

1. Emart Website

1.1 Home page

Done by : E. L. Thambawita
Student ID : 29186

1.1.1 Web page



Best Sellers



Samsung Galaxy S24
LKR 400,000



Samsung Galaxy S22
LKR 300,000



iPhone 14
LKR 350,000



iPhone 15
LKR 500,000

New Arrivals



name
LKR 56546



name
LKR 694200



iPhone 12
LKR 5345



Smartphone C
LKR 32432

Categories



Smartphones

Explore our sleek smartphones, packed with cutting-edge tech for seamless connectivity and entertainment.



Laptops

Enhance productivity with our versatile range of laptops, meticulously designed to meet your diverse needs.



Cameras

Capture life's moments in stunning detail with our carefully curated selection of cameras.



Smart Watches

Stay connected, organized, and impeccably stylish with our innovative collection of smartwatches."



Monitors

Immerse yourself in breathtaking visuals with our premium lineup of meticulously engineered monitors.

Experience Next-Level Performance with E-Mart Electronics

[Start Shopping](localhost:8080/ProductCategory_Cameras.jsp)

The screenshot shows the homepage of E-Mart Electronics. At the top, there's a dark blue header bar with a "Start Shopping" button. Below the header is a large, dark background image of a smartphone. In the upper left corner of the main content area, there's a logo for "E-MART ELECTRONICS" featuring a blue shopping bag icon. To the right of the logo is a brief promotional text about the latest innovations in electronics. Further down, there are social media sharing icons for Facebook, Twitter, Instagram, LinkedIn, and YouTube. On the far right, there are links for "Company", "My Mart", and "Contact Us", along with contact information including a phone number and email address. At the bottom, there's a copyright notice for 2024 and links for "Privacy Policy" and "Terms of Use".

1.1.2 Documentation: Home Page Implementation

Introduction:

As the primary landing page, the website's Home Page gives visitors an overview of what's available and entices them to explore further. The functionality and implementation specifics of the Home Page are explained in this documentation.

Background and overview:

The Home Page has an aesthetically pleasing background image that is overlaid with a gradient to increase visual appeal.

It gives a brief overview of the website's primary focus - smartphones - and invites visitors to learn more.

Buttons and Navigation:

Two prominent buttons, "About Us" and "Start Shopping," are displayed, with each linking to relevant pages on the site.

The "About Us" button takes users to a page that contains information about the website and its mission. The "Start Shopping" button takes users to the product categories page, promoting immediate exploration and interaction.

Featured products:

The Home Page displays best-selling products in a visually appealing format.

Each product is presented in a rectangular container that includes an image, product name, and price. Clicking on a product takes users to the corresponding product category page for further exploration.

New Arrivals:

A section dedicated to showcasing the most recent product arrivals is prominently displayed.

These new arrivals are dynamically fetched from the database and displayed in rectangular containers, much like the featured products section.

Each new arrival includes an image, product name, and price, as well as links to the product category pages.

Product Categories:

The Home Page contains direct links to all the product categories, such as smartphones, laptops, cameras, smartwatches, and monitors.

Each category is represented by an icon, a category name, and a brief description, encouraging users to explore specific product ranges.

Call to Action:

Toward the bottom of the Home Page, a prominent call-to-action encourages users to start shopping. Clicking the "Start Shopping" button takes users to the smartphones category page, allowing for easy navigation and interaction.

1.1.3 Codes

index.jsp

```
1  <%-- Document      : Home           Next Bookmark
2  Created on : Apr 11, 2024, 3:53:02 PM
3  Author       : Esanki Lakvindee
4  --%>
5
6
7  <%@page contentType="text/html" pageEncoding="UTF-8"%>
8  <%@page import="java.util.List"%
9  <%@page import="Model.DAO"%
10 <%@page import="Model.NewestProductObj"%
11 <!DOCTYPE html>
12 <html>
13   <head>
14     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
15     <title>Home</title>
16     <link href="CSS/Common.css" rel="stylesheet" type="text/css"/>
17     <link href="CSS/Navbar.css" rel="stylesheet" type="text/css"/>
18     <link href="CSS/Footer.css" rel="stylesheet" type="text/css"/>
19     <link href="CSS/Home.css" rel="stylesheet" type="text/css"/>
20   </head>
21
22   <body>
23     <%@ include file="Navbar.jsp" %>
24
25     <% DAO.getAllCartItems(); %>
26     <% DAO.getAllSmartphones(); %>
27     <% DAO.getNewArrivals(); %>
28
29     <!-- Home page UI --&gt;
30     &lt;div class="container"&gt;
31       &lt;img src="Images/Home/lens.jpg" alt="Background Image" class="background-image"&gt;
32       &lt;div class="gradient-overlay"&gt;&lt;/div&gt;
33       &lt;div class="content"&gt;
34         &lt;h1&gt;Discover Smart Connections with&lt;br&gt;E-Mart Smartphones&lt;/h1&gt;
35
36         &lt;p&gt;Explore the latest in smartphone technology and&lt;br&gt;connectivity. From flagship models to budget-friendly&lt;br&gt;
37           options, find your perfect fit at E-Mart.&lt;/p&gt;
38         &lt;div class="buttons"&gt;
39           &lt;a href="AboutUs.jsp"&gt;
40             &lt;button class="button1"&gt;About Us&lt;/button&gt;
41           &lt;/a&gt;
42           &lt;a href="ProductCategory_Cameras.jsp"&gt;
43             &lt;button class="button2"&gt;Start Shopping&lt;/button&gt;
44           &lt;/a&gt;
45         &lt;/div&gt;
46       &lt;/div&gt;
47
48     &lt;!-- Second Section --&gt;
49     &lt;div class="second-section"&gt;
50       &lt;h2 class="Topic"&gt;Best Sellers&lt;/h2&gt;
51       &lt;div class="rectangle-container"&gt;
52         &lt;div class="rectangle"&gt;
53           &lt;a href="ProductCategory_Phones.jsp"&gt;
54             &lt;div class="img-container"&gt;
55               &lt;img src="Images/Home/S24.png" alt="Product Image"&gt;
56             &lt;/div&gt;
57             &lt;span class="text1"&gt;Samsung Galaxy S24&lt;/span&gt;
58             &lt;span class="text2"&gt;LKR 400,000&lt;/span&gt;
59           &lt;/a&gt;
60
61         &lt;/div&gt;
62
63         &lt;div class="rectangle"&gt;
64           &lt;a href="ProductCategory_Phones.jsp"&gt;
65             &lt;div class="img-container"&gt;
66               &lt;img src="Images/Home/S22.png" alt="Product Image"&gt;
67             &lt;/div&gt;
68             &lt;span class="text1"&gt;Samsung Galaxy S22&lt;/span&gt;
69             &lt;span class="text2"&gt;LKR 300,000&lt;/span&gt;
70           &lt;/a&gt;
71
72       &lt;/div&gt;
73     &lt;/div&gt;</pre>
```

```

74   <div class="rectangle">
75     <a href="ProductCategory_Phones.jsp">
76       <div class="img-container">
77         
78       </div>
79       <span class="text1">Iphone 14</span>
80       <span class="text2">LKR 350,000</span>
81     </a>
82
83   </div>
84
85   <div class="rectangle">
86     <a href="ProductCategory_Phones.jsp">
87       <div class="img-container">
88         
89       </div>
90       <span class="text1">Iphone 15</span>
91       <span class="text2">LKR 500,000</span>
92     </a>
93
94   </div>
95 </div>
96
97
98  <!-- Third Section -->
99  <div class="second-section">
100    <h2 class="Topic">New Arrivals</h2>
101    <div class="rectangle-container">
102      <%
103        List<NewestProductObj> newArrivals = DAO.getNewArrivals();
104        for (int i = 0; i < 4 && i < newArrivals.size(); i++) {
105          NewestProductObj newItem = newArrivals.get(i);
106        %}
107
108      <div class="rectangle">
109        <a href="ProductCategory_Cameras.jsp">
110          <div class="img-container">
111            
112          </div>
113          <span class="text1"><%= newItem.getProductName() %></span>
114          <span class="text2">LKR <%= newItem.getProductPrice() %></span>
115        </a>
116      </div>
117      <% } %>
118    </div>
119
120    <!--
121      <div class="rectangle">
122        <a href="abc.jsp">
123          
124          <span class="text1">SONY Alpha 9 II</span>
125          <span class="text2">LKR 300,000</span>
126        </a>
127      </div>
128
129      <div class="rectangle">
130        <a href="abc.jsp">
131          
132          <span class="text1">SONY RX10 IV</span>
133          <span class="text2">LKR 450,000</span>
134        </a>
135      </div>
136
137      <div class="rectangle">
138        <a href="abc.jsp">
139          

```

```

140 |           <span class="text1">Iphone 15</span>
141 |           <span class="text2">LKR 500,000</span>
142 |       </a>
143 |   </div>
144 |   </div>-->
145 | 
146 | 
147 | <!--Fourth section -->
148 | <div class="categories-text">Categories</div>
149 | 
150 | <div class="category-container">
151 |     <a href="ProductCategory_Phones.jsp" class="category1">
152 |         <div class="column1">
153 |             
154 |         </div>
155 |         <div class="column2">
156 |             <div class="line1">Smartphones</div>
157 |             <div class="line2">Explore our sleek smartphones, packed with cutting-edge tech for seamless connectivity and enter
158 |         </div>
159 |     </a>
160 | 
161 |     <a href="ProductCategory_Laptops.jsp" class="category2">
162 |         <div class="column1">
163 |             
164 |         </div>
165 |         <div class="column2">
166 |             <div class="line1">Laptops</div>
167 |             <div class="line2">Enhance productivity with our versatile range of laptops, meticulously designed to meet your div
168 |         </div>
169 |     </a>
170 | 
171 |     <a href="ProductCategory_Cameras.jsp" class="category3">
172 |         <div class="column1">
173 |             
174 |         </div>
175 |         <div class="column2">
176 |             <div class="line1">Cameras</div>
177 |             <div class="line2">Capture life's moments in stunning detail with our carefully curated selection of cameras.</div>
178 |         </div>
179 |     </a>
180 | </div>
181 | 
182 | <!-- Fifth section -->
183 | <div class="nextcategory-container">
184 |     <a href="ProductCategory_Smartwatches.jsp" class="category4">
185 |         <div class="column1">
186 |             
187 |         </div>
188 |         <div class="column2">
189 |             <div class="line1">Smart<br><br>Watches</div>
190 |             <div class="line2">Stay connected, organized, and impeccably stylish with our innovative collection of smartwatches
191 |         </div>
192 |     </a>
193 | 
194 |     <a href="ProductCategory_Monitors.jsp" class="category5">
195 |         <div class="column1">
196 |             
197 |         </div>
198 |         <div class="column2">
199 |             <div class="line1">Monitors</div>
200 |             <div class="line2">Immerse yourself in breathtaking visuals with our premium lineup of meticulously engineered moni
201 |         </div>
202 |     </a>
203 | </div>
204 | 
205 | 
206 | <!-- Sixth section -->
207 | <div class="longrectangle-container">
208 |     <div class="longrectangle">
209 |         <p class="sentence">Experience Next-Level Performance with E-Mart Electronics</p>
210 |         <button class="start-shopping" onclick="window.location.href = 'ProductCategory_Phones.jsp';">Start Shopping</button>
211 |     </div>
212 |     </div>
213 |     <%@ include file="Footer.html" %>
214 |     <script src="JS/Common.js"></script>
215 | </body>
216 | </html>
217 |

```

Home.css

```
Start Page × DAO.java × index.jsp × Home.css × NewestProductObj.java ×
Source History  New File Open Recent Save All Refresh Find Replace        

```

1 /*
2 Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3 Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/CascadeStyleSheet.css to edit this template
4 */
5 /*
6 Created on : Apr 11, 2024, 3:53:39 PM
7 Author : Esanki Lakvindee
8 */
9
10 /*Global styles*/
11 body, html {
12 margin: 0;
13 padding: 0;
14 font-family: Calibri, sans-serif;
15 color: #FFF;
16 background-color: black;
17 height: 100%;
18 }
19
20 .gradient-overlay {
21 position: absolute;
22 top: 0;
23 left: 0;
24 width: 100%;
25 height: 100%;
26 background-image: linear-gradient(180deg, rgba(0,0,0,0.8) 0%, rgba(0,0,0,0.6) 100%);
27 z-index: 1; /* Ensure the gradient overlay is behind the content but on top of the background image */
28 }
29
30 .container {
31 position: relative;
32 width: 100%;
33 height: 815px; /* Adjust height as needed */
34 overflow: hidden; /* Ensure that content beyond the height of the container is hidden */
35 background-color: black; /* Set background color as desired */
36 }
37
38 .background-image {
39 width: 100%;
40 height: 815px; /* Adjust height as needed */
41 object-fit: cover;
42 }
43
44 .content {
45 position: absolute;
46 top: 149px;
47 left: 114px;
48 text-align: left;
49 z-index: 2;
50 }
51
52 h1, p {
53 font-size: 3rem;
54 margin-bottom: 2rem;
55 }
56
57 p {
58 font-size: 1.5rem;
59 margin-bottom: 3rem;
60 }
61
62 .buttons {
63 display: flex;
64 justify-content: flex-start;
65 align-items: center;
66 gap: 20px;

```


```

```

68
69  button {
70    width: 200px;
71    height: 50px;
72    border-radius: 15px;
73    border: 0.5px solid #FFF;
74    background: rgba(0, 0, 0, 0.00);
75    font-size: 21px;
76    cursor: pointer;
77    color: #FFF;
78  }
79
80  .button1 {
81    background: rgba(0, 0, 0, 0.90); /* Black color background */
82  }
83
84  .button2 {
85    background: rgba(21, 22, 99, 0.90) !important;
86    backdrop-filter: blur(50px);
87    border: 0.5px solid blue;
88  }
89
90
91  /*Second Section*/
92  .second-section {
93    position: relative;
94    margin-top: 141px; /* Placed after 141px after the image lens */
95    text-align: center;
96  }
97
98  .Topic {
99    color: #FFF;
100   font-family: Calibri;
101
102   font-size: 64px;
103   font-style: normal;
104   font-weight: 700;
105   line-height: normal;
106   letter-spacing: 3.2px;
107 }
108
109 .rectangle-container {
110   display: flex;
111   justify-content: space-between;
112   margin-left: 114px; /* Space from left page border */
113   margin-right: 107px; /* Space from right page border */
114   gap: 35px; /* Gap between rectangles */
115 }
116
117 .rectangle {
118   width: 394px;
119   height: 523px;
120   border-radius: 25px;
121   border: 4px solid rgba(255, 255, 255, 0.40);
122   background: linear-gradient(0deg, rgba(255, 255, 255, 0.07) 0%, rgba(255, 255, 255, 0.07) 100%), linear-gradient(209deg, rgba(0, 0, 0, 0.05) 0%, rgba(0, 0, 0, 0.05) 100%);
123   overflow: hidden; /* Ensures that the border-radius is applied to the background */
124 }
125
126 .rectangle img {
127   width: 90%;
128   margin-top: 41px;
129   max-height: 100%;
130 }
131
132 .text1, .text2 {
133   display: block;

```

```

134     color: #FFF;
135     font-family: Calibri;
136     font-style: normal;
137     line-height: 16px;
138     letter-spacing: 2.24px;
139   }
140
141   .text1 {
142     font-size: 28px;
143     font-weight: 700;
144     margin-top: 50px;
145   }
146
147   .text2 {
148     font-size: 32px;
149     font-weight: 400;
150     color: rgba(255, 255, 255, 0.70);
151     margin-top: 26px;
152   }
153
154   /*Fourth Section*/
155   .categories-text {
156     text-align: center;
157     color: #FFF;
158     font-family: Calibri;
159     font-size: 64px;
160     font-weight: 700;
161     line-height: 16px;
162     letter-spacing: 3.2px;
163     margin-top: 171px;
164   }
165
166   .category-container {
167     display: flex;
168     justify-content: space-between;
169     margin-left: 114px; /* Space from left page border */
170     margin-right: 107px; /* Space from right page border */
171     margin-top: 93px;
172   }
173
174   .category-container .category1,
175   .category-container .category2,
176   .category-container .category3 {
177     width: 542px; /* Adjust width as needed */
178     height: 314px; /* Adjust height as needed */
179     border-radius: 25px;
180     border: 1px solid rgba(255, 255, 255, 0.26);
181     background: linear-gradient(0deg, rgba(255, 255, 255, 0.07) 0%, rgba(255, 255, 255, 0.07) 100%), linear-gradient(209deg, rgba(0, 0, 0, 0.05) 0%, transparent 100%);
182     display: flex;
183     justify-content: space-between;
184     align-items: center;
185     padding: 20px;
186     margin-right: 35px; /* Gap between each category */
187   }
188
189   .category-container .category1 {
190     margin-left: 0;
191   }
192
193   .category-container .category3 {
194     margin-right: 0;
195   }
196

```

```

197 .category-container .column1 img {
198   width:40%;
199 }
200
201 .category-container .column2 {
202   width: 60%; /* Each takes half of the container */
203 }
204
205 .category-container .line1 {
206   color: #FFF;
207   font-family: Calibri;
208   font-size: 35px;
209   font-weight: 700;
210   line-height: 16px;
211   letter-spacing: 2.4px;
212   margin-bottom: 20px;
213 }
214
215 .category-container .line2 {
216   color: rgba(255, 255, 255, 0.80);
217   font-family: Calibri;
218   font-size: 22px;
219   font-weight: 300;
220   line-height: 20px;
221   letter-spacing: 2.4px;
222 }
223
224 /* Fifth section */
225 .nextcategory-container {
226   display: flex;
227   justify-content: space-between;
228   margin-left: 300px;
229   margin-right: 300px;
230   margin-top: 56px;
231 }
232
233 .nextcategory-container .category4,
234 .nextcategory-container .category5 {
235   width: 542px; /* Adjust width as needed */
236   height: 314px; /* Adjust height as needed */
237   border-radius: 25px;
238   border: 1px solid rgba(255, 255, 255, 0.26);
239   background: linear-gradient(0deg, rgba(255, 255, 255, 0.07) 0%, rgba(255, 255, 255, 0.07) 100%), linear-gradient(209deg, rgba(0, 0, 0, 0.05) 0%, transparent 100%);
240   display: flex;
241   justify-content: space-between;
242   align-items: center;
243   padding: 20px;
244   margin-right: 36px; /* Gap between each category */
245 }
246
247 .nextcategory-container .category4 {
248   margin-left: 0;
249 }
250
251 .nextcategory-container .category5 {
252   margin-right: 0;
253 }
254

```

```
255 .nextcategory-container .column1 img {  
256     height: 176px; /* Image height */  
257     width: 141px; /* Image width */  
258 }  
259  
260 .nextcategory-container .column2 {  
261     width: 50%; /* Each takes half of the container */  
262 }  
263  
264 .nextcategory-container .line1 {  
265     color: #FFF;  
266     font-family: Calibri;  
267     font-size: 35px;  
268     font-weight: 700;  
269     line-height: 16px;  
270     letter-spacing: 2.4px;  
271     margin-bottom: 20px;  
272 }  
273  
274 .nextcategory-container .line2 {  
275     color: rgba(255, 255, 255, 0.80);  
276     font-family: Calibri;  
277     font-size: 22px;  
278     font-weight: 300;  
279     line-height: 20px;  
280     letter-spacing: 2.4px;  
281 }  
282  
283 /* Sixth section */  
284 .longrectangle-container {  
285     display: flex;  
286     justify-content: center;  
287     margin-left: 134px;  
288     margin-right: 134px;  
289     margin-top: 200px;  
290     margin-bottom: 200px;  
291 }  
292  
293 .longrectangle {  
294     width: 1652px;  
295     height: 255px;  
296     border-radius: 15px;  
297     border: 1px solid #3B7BF8;  
298     background: linear-gradient(209deg, rgba(21, 22, 99, 0.70) 17.86%, rgba(21, 22, 99, 0.70) 82.12%);  
299     display: flex;  
300     flex-direction: column;  
301     align-items: center;  
302     justify-content: center;  
303 }  
304  
305 .sentence {  
306     color: #FFF;  
307     font-family: Calibri;  
308     font-size: 40px;  
309     font-weight: 400;  
310     line-height: 16px; /* 33.333% */  
311     letter-spacing: 3px;  
312     margin-bottom: 60px; /* Adjust as needed */  
313 }  
314 }
```

```

315  .start-shopping {
316    width: 300px;
317    height: 60px;
318    border-radius: 15px;
319    border: 2px solid #151663;
320    background: rgba(255, 255, 255, 0.90);
321    backdrop-filter: blur(50px);
322    font-family: Calibri;
323    font-size: 24px;
324    font-weight: 700;
325    color: #151663;
326    cursor: pointer;
327  }
328
329  /*Clicking parts of second, third sections*/
330  .rectangle-container .rectangle {
331    position: relative;
332  }
333
334  .rectangle-container .rectangle a {
335    display: block;
336    position: relative;
337  }
338
339  .rectangle-container .rectangle a img {
340    display: block;
341    transition: filter 0.3s ease; /* Smooth transition for the hover effect */
342  }
343
344  .rectangle-container .rectangle a:hover img {
345    filter: brightness(70%);
346  }
347
348  .rectangle-container .rectangle a:hover .text1,
349  .rectangle-container .rectangle a:hover .text2 {
350    color: rgba(255, 255, 255, 0.9);
351  }
352
353
354  /*Clicking part of fourth,fifth sections*/
355  .category-container a,
356  .nextcategory-container a {
357    display: block;
358    text-decoration: none; /* Remove underline from links */
359    transition: transform 0.3s ease; /* Smooth transition for the hover effect */
360  }
361
362  .category-container a:hover,
363  .nextcategory-container a:hover {
364    transform: scale(1.1); /* Increase size by 10% when hovered */
365  }
366
367  .category-container a:hover .line1,
368  .category-container a:hover .line2,
369  .nextcategory-container a:hover .line1,
370  .nextcategory-container a:hover .line2 {
371    color: rgba(255, 255, 255, 0.9); /* Change text color when hovered */
372  }
373
374  a {
375    text-decoration: none;
376  }
377
378  .img-container{
379    height: 350px;
380    display: flex;
381    flex-direction: column;
382    align-content: center;
383    justify-content: center;
384    align-items: center;
385  }
386
387  .nav-link.home-link {
388    opacity: 1;
389  }

```

DAO.java – getNewArrivals()

The screenshot shows two code editors side-by-side, both titled "DAO.java".

The top editor (lines 1-25) contains the definition of a class named DAO:

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package Model;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * @author robin
 */
public class DAO {

    private static final String URL = "jdbc:mysql://localhost:3306/emart";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "";
```

The bottom editor (lines 418-450) contains the implementation of the getNewArrivals() method:

```
public static List<NewestProductObj> getNewArrivals() {
    List<NewestProductObj> newitems = new ArrayList<>();
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;
    try {
        Class.forName("com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);

        // Creating SQL statement
        statement = connection.createStatement();

        // SQL query to fetch shopping cart items
        String query = "SELECT * FROM smartphone ORDER BY productId DESC";

        // Executing the query
        resultSet = statement.executeQuery(query);

        // Iterating over the result set and adding items to the shopping cart list
        while (resultSet.next()) {
            String productName = resultSet.getString("productName");
            int productPrice = resultSet.getInt("price");
            String iconPath = resultSet.getString("photo1");

            // Creating ShoppingCartItem object and adding it to the list
            NewestProductObj items = new NewestProductObj(productName, productPrice, iconPath);
            newitems.add(items);
        }
    } catch (Exception e) {
        System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
    }
}
```

```
451     } finally {
452         // Closing the connection, statement, and result set
453         try {
454             if (resultSet != null) {
455                 resultSet.close();
456             }
457             if (statement != null) {
458                 statement.close();
459             }
460             if (connection != null) {
461                 connection.close();
462             }
463         } catch (SQLException e) {
464             System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
465         }
466     }
467     return newitems;
468 }
```

NewestProductObj.java

```
Start Page x DAO.java x index.jsp x Home.css x NewestProductObj.java x
Source History |          
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
4   */
5  package Model;
6
7  /**
8   *
9   * @author robin
10  */
11 public class NewestProductObj {
12
13     private String productName;
14     private int productPrice;
15     private String iconPath;
16
17     public NewestProductObj(String productName, int productPrice, String iconPath) {
18         this.productName = productName;
19         this.productPrice = productPrice;
20         this.iconPath = iconPath;
21     }
22
23
24     public String getProductName() {
25         return productName;
26     }
27
28     public void setProductName(String productName) {
29         this.productName = productName;
30     }
31
32     public int getProductPrice() {
33         return productPrice;
34     }
35
36     public void setProductPrice(int productPrice) {
37         this.productPrice = productPrice;
38     }
39
40     public String getIconPath() {
41         return iconPath;
42     }
43
44     public void setIconPath(String iconPath) {
45         this.iconPath = iconPath;
46     }
47
48 }
49
```

Conclusion:

The Home Page is the website's main entry point, effectively communicating the brand's message, showcasing featured products, highlighting new arrivals, and providing easy navigation options. The Home Page's meticulous design and dynamic content integration aims to engage users and encourage them to explore and interact with the website's offerings.

1.2 Order Confirmation page

Done by : E. L. Thambawita

Student ID : 29186

1.2.1 Web page

A screenshot of a web browser window showing the 'Order Confirmation' page. The URL in the address bar is 'localhost:8080/OrderConfirmation.jsp'. The page has a dark background with white text. At the top center, it says 'Thank you for your order!'. Below that is a 'Order Summary' section with a table of items. At the bottom, there are buttons for 'Order History' and 'Continue Shopping', and a total amount of 'LKR 3396'.

Order Summary			
1)	Smartphone	1 items	LKR 699
2)	Laptop	2 items	LKR 2198
3)	Camera	1 items	LKR 499

Total - LKR 3396

A screenshot of a smartphone displaying a feedback survey overlay. The overlay has a dark blue background with white text. It asks "How was your shopping experience?" and features a five-star rating scale. Below the stars is a "Submit Feedback" button. At the bottom of the screen, there is a navigation bar with the E-Mart logo, social media icons, and links for Company, My Mart, and Contact Us.

How was your shopping experience?

★★★★★

Submit Feedback

E-MART
ELECTRONICS

From cutting-edge processors to AI-powered technologies, E-Mart brings you the latest innovations in electronics. Elevate your experience and embrace the future with E-Mart.

Company My Mart Contact Us

About E-Mart My Profile +94 77 696 9696
Developers Ongoing Orders
Order History mail.emart@gmail.com

[Facebook](#) [Twitter](#) [Instagram](#) [LinkedIn](#) [YouTube](#)

A screenshot of a smartphone displaying a feedback survey overlay. The overlay has a dark blue background with white text. It asks "How was your shopping experience?" and features a four-star rating scale. Below the stars is a "Submit Feedback" button. The rest of the screen is black, indicating the background of the phone.

How was your shopping experience?

★★★★★

Submit Feedback

A screenshot of a smartphone displaying an order confirmation page. The page has a dark background with white text. It says "Thank you for your order!" and includes a message from "localhost:8080" stating "Failed to send the email. Please try again later." There is an "OK" button at the bottom of the message box. At the top, there is a navigation bar with the E-Mart logo and links for Home, Products, About Us, and Support. On the right side, there are icons for user profile, cart, and more.

localhost:8080 says

Failed to send the email. Please try again later.

OK

E-MART
ELECTRONICS

Home Products About Us Support

Thank you for your order!

1.2.2 Documentation: Order Confirmation Page Implementation

Introduction:

The Order Confirmation Page is an important part of the e-commerce platform because it gives users a summary of their most recent purchases and prompts them to provide feedback on their shopping experience. This documentation describes the Order Confirmation Page's functionality and implementation details, including the structure of the JSP file, associated CSS styles, and JavaScript scripts.

Overview of OrderConfirmation.jsp:

The OrderConfirmation.jsp file renders the Order Confirmation Page interface. It uses JavaServer Pages (JSP) technology to dynamically generate HTML content from server-side data. The OrderConfirmation.jsp file contains the following key features:

Dynamic Order Summary: The page uses a DAO (Data Access Object) class to dynamically retrieve order details from the server and iterate over the list of items in the user's shopping cart. Each item is listed with its name, quantity, and total price.

Navigation Buttons: Users can view their order history or continue shopping. These buttons use JavaScript to redirect users to their respective pages.

Method calls:

- DAO.getAdminOrders(): Fetches the list of admin orders from the database.
- DAO.getOrderConfirmationDetails(): Fetches the order confirmation details, typically shopping cart items, from the database.

Feedback section: The page contains a feedback section where users can rate their shopping experience using a star rating system. The selected rating is forwarded to the server for further processing.

Email Integration: The page uses the MailerSend API to send email notifications to users once their orders are confirmed. This functionality is implemented using server-side Java code.

Special Imports:

```
<%@ page import="java.io.BufferedReader" %>
<%@ page import="java.io.DataOutputStream" %>
<%@ page import="java.io.InputStreamReader" %>
<%@ page import="java.net.HttpURLConnection" %>
<%@ page import="java.net.URL" %>
```

These imports are used in conjunction with the MailerSend API integration to send email notifications to users upon order confirmation. The 'HttpURLConnection' and 'URL' classes are used to connect to the MailerSend API endpoint, while the 'DataOutputStream' and 'BufferedReader' classes send the request payload and receive

the response, respectively. Overall, these imports make it easier to communicate with external services via HTTP, allowing you to seamlessly integrate additional functionality into your web application.

OrderConfirmation.css

The OrderConfirmation.css file specifies the visual styling for the Order Confirmation Page, resulting in a consistent and visually appealing user experience. The key styling elements are:

Container Layout: Styles are used to create a centered layout for page content, which improves readability and visual consistency.

Order Summary: The order summary section is designed to present information in a clear and organized format, with emphasis on typography, spacing, and color contrast.

Navigation Buttons: Styling is used to improve the visual appeal of the navigation buttons and to provide clear user interaction options.

Feedback Section: The feedback section is visually distinct from the rest of the page, with tailored styles for the star rating interface and submit button.

Imports: @import url(<https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css>);

This line imports icon styles from the Font Awesome library which we have used to get the star icons for the feedback part.

Feedback-Rating.js

The Feedback-Rating.js file manages the client-side behavior of the order confirmation page's feedback rating feature. It allows users to provide feedback by selecting a rating and sending it to the server asynchronously.

Rating Display : When users interact with the star rating interface, the JavaScript code dynamically updates the displayed rating value. This feature improves the user experience by providing real-time feedback on the selected rating.

Feedback Submission : After users select a rating and click the submit button, the submitFeedback() function is called. This function carries out the following actions:

- Retrieve Feedback Value: This method retrieves the selected feedback value from the star rating interface.
- Retrieve Order ID: It retrieves the order ID for the current order confirmation from the HTML element's dataset.

- Send Feedback to the Server: The JavaScript function uses an XMLHttpRequest to send an asynchronous POST request to the server-side endpoint (FeedbackServlet), passing the feedback value and order ID as parameters.
- Handle Server Response: When the server returns a response, the JavaScript code can handle it appropriately. If the feedback submission is successful, the server will respond with a confirmation message.

Interaction with Feedback Servlet

The submitFeedback() function communicates with the FeedbackServlet backend endpoint to manage the feedback submission process. This is how it works.

- Data Preparation: The JavaScript function collects the feedback value and order ID from the client side.
- Asynchronous Request: It sends an asynchronous POST request to the FeedbackServlet endpoint with the feedback value and order ID as parameters.
- Server-Side Processing: The FeedbackServlet accepts the POST request, extracts the feedback value and order ID, and stores it in the database.
- Response Handling: After successfully processing the feedback submission, the servlet returns a response to the client-side JavaScript code, confirming the successful submission.

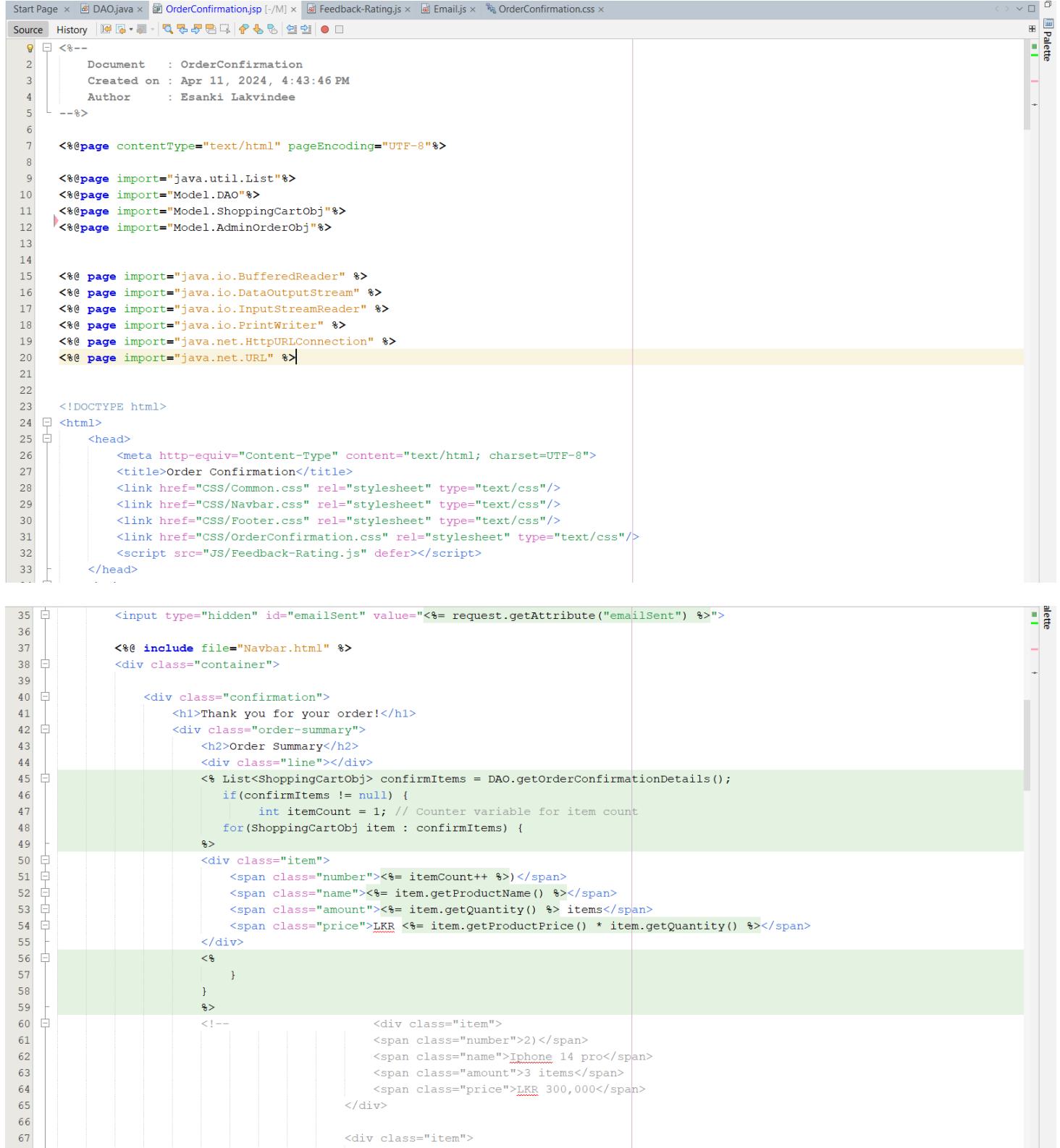
Email.js

The Email.js file contains client-side JavaScript code responsible for handling email notification functionality. Key functionalities include:

Email Notification Handling: The script retrieves the emailSent attribute value from the HTML element and displays a corresponding notification message to the user based on the success or failure of the email sending process.

1.2.3 Codes

OrderConfirmation.jsp



The screenshot shows the Java code for the OrderConfirmation.jsp page. The code includes imports for Java.util.List, Model.DAO, Model.ShoppingCartObj, Model.AdminOrderObj, and various Java.io and java.net classes. It sets the page content type to text/html and imports JavaScript files like Feedback-Rating.js and Email.js. The page starts with a DOCTYPE html and an HTML head section containing meta tags, title, and links to CSS and JS files. The body section contains an input field for emailSent, an include directive for Navbar.html, and a container div. Inside the container, there's a confirmation message and an order summary table. The table lists items with columns for number, name, amount, and price. The code uses JSP scriptlets to interact with the DAO layer to get order confirmation details.

```
<%--  
    Document : OrderConfirmation  
    Created on : Apr 11, 2024, 4:43:46 PM  
    Author : Esanki Lakvindee  
--%>  
  
<%@page contentType="text/html" pageEncoding="UTF-8"%>  
  
<%@page import="java.util.List"%>  
<%@page import="Model.DAO"%>  
<%@page import="Model.ShoppingCartObj"%>  
<%@page import="Model.AdminOrderObj"%>  
  
<%@ page import="java.io.BufferedReader" %>  
<%@ page import="java.io.DataOutputStream" %>  
<%@ page import="java.io.InputStreamReader" %>  
<%@ page import="java.io.PrintWriter" %>  
<%@ page import="java.net.HttpURLConnection" %>  
<%@ page import="java.net.URL" %>  
  
<!DOCTYPE html>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
        <title>Order Confirmation</title>  
        <link href="CSS/Common.css" rel="stylesheet" type="text/css"/>  
        <link href="CSS/Navbar.css" rel="stylesheet" type="text/css"/>  
        <link href="CSS/Footer.css" rel="stylesheet" type="text/css"/>  
        <link href="CSS/OrderConfirmation.css" rel="stylesheet" type="text/css"/>  
        <script src="JS/Feedback-Rating.js" defer></script>  
    </head>  
  
    <input type="hidden" id="emailSent" value="<%= request.getAttribute("emailSent") %>">  
    <%@ include file="Navbar.html" %>  
    <div class="container">  
  
        <div class="confirmation">  
            <h1>Thank you for your order!</h1>  
            <div class="order-summary">  
                <h2>Order Summary</h2>  
                <div class="line"></div>  
                <% List<ShoppingCartObj> confirmItems = DAO.getOrderConfirmationDetails();  
                    if(confirmItems != null) {  
                        int itemCount = 1; // Counter variable for item count  
                        for(ShoppingCartObj item : confirmItems) {  
                            %>  
                            <div class="item">  
                                <span class="number"><%= itemCount++ %></span>  
                                <span class="name"><%= item.getProductName() %></span>  
                                <span class="amount"><%= item.getQuantity() %> items</span>  
                                <span class="price">LKR <%= item.getProductPrice() * item.getQuantity() %></span>  
                            </div>  
                            <%  
                            }  
                            %>  
                            <!--  
                                <div class="item">  
                                    <span class="number">2</span>  
                                    <span class="name">Iphone 14 pro</span>  
                                    <span class="amount">3 items</span>  
                                    <span class="price">LKR 300,000</span>  
                                </div>  
                                <div class="item">  
                            -->  
            </div>  
        </div>  
    </div>
```

```

68 |           <span class="number">3</span>
69 |           <span class="name">Iphone 14 pro</span>
70 |           <span class="amount">3 items</span>
71 |           <span class="price">LKR 300,000</span>
72 |       </div>-->
73 |
74 |
75 |     <div class="line"></div>
76 |
77 |     <div class="button-container">
78 |         <button class="button button1" onclick="window.location.href = 'OrderHistory.jsp';">Order History</button>
79 |         <button class="button button2" onclick="window.location.href = 'ProductCategory_Phones.jsp';">Continue Shopping</button>
80 |         <span class="text1">Total -</span>
81 |     <%
82 |         int total = 0;
83 |         for (ShoppingCartObj item : confirmItems) {
84 |             total += item.getTotalPrice();
85 |         }
86 |     %>
87 |     <span class="text2">LKR <%= total %></span>
88 |   </div>
89 |
90 |   </div>
91 |
92 |   </div>
93 |
94 |   <div class="feedback">
95 |     <h3>How was your shopping experience?</h3>
96 |
97 |     <!-- Star rating section -->
98 |     <div class="stars">
99 |       <fieldset class="rating">
100 |         <%
101 |             <input type="radio" id="star5" name="rating" value="5"/><label for="star5" class="full" title="Awesome"></label>
102 |             <input type="radio" id="star4" name="rating" value="4"/><label for="star4" class="full"></label>
103 |             <input type="radio" id="star3" name="rating" value="3"/><label for="star3" class="full"></label>
104 |             <input type="radio" id="star2" name="rating" value="2"/><label for="star2" class="full"></label>
105 |             <input type="radio" id="star1" name="rating" value="1"/><label for="star1" class="full"></label>
106 |         </fieldset>
107 |       </div>
108 |     <!-- Submit button -->
109 |     <div>
110 |       <%
111 |           List<AdminOrderObj> orders = DAO.getAdminOrders();
112 |           int lastorderId = 0; // Initialize with 0 or any other default value
113 |           int lastIndex = orders.size() - 1;
114 |           if (lastIndex >= 0) {
115 |               // Get the orderId of the last order
116 |               lastorderId = orders.get(lastIndex).getOrderNumber();
117 |           }
118 |       %>
119 |     <button type="button" class="submit-feedback" onclick="submitFeedback()" data-orderid="<%= lastorderId %>">
120 |       Submit Feedback
121 |     </button>
122 |
123 |   </div>
124 |
125 | </div>
126 |
127 |
128 | </div>
129 |
130 |

```

```

131
132     <%String apiKey = "mlsn.1e09da5329b79f474aff9bc3ccb8ddc06df779561f248f8fc202c9764dea4a33";
133     String senderEmail = "MS_liz77E@trial-k68zx12enw9l1j905.mlsender.net";
134     String recipientEmail = "esankilakvindee2000@gmail.com";
135
136     String url = "https://api.mailersend.com/v1/email";
137     String payload = "{\n" +
138         "   \"from\": {\n" +
139             "     \"email\": \"" + senderEmail + "\"\n" +
140             "   },\n" +
141             "   \"to\": [\n" +
142                 "\n" +
143                     "       \"email\": \"\" + recipientEmail + "\"\n" +
144                     "   ]\n" +
145             "   ],\n" +
146             "   \"subject\": \"Hello from MailerSend!\",\n" +
147             "   \"text\": \"Greetings from the team, you got this message through MailerSend.\",\n" +
148             "   \"html\": \"Greetings from the team, you got this message through MailerSend.\n\" +
149         "}";
150
151     try {
152         URL obj = new URL(url);
153         HttpURLConnection con = (HttpURLConnection) obj.openConnection();
154
155         con.setRequestMethod("POST");
156         con.setRequestProperty("Content-Type", "application/json");
157         con.setRequestProperty("X-Requested-With", "XMLHttpRequest");
158         con.setRequestProperty("Authorization", "Bearer " + apiKey);
159
160         con.setDoOutput(true);
161         DataOutputStream wr = new DataOutputStream(con.getOutputStream());
162         wr.writeBytes(payload);
163         wr.flush();
164
165         wr.close();
166
167         int responseCode = con.getResponseCode();
168         System.out.println("Response Code : " + responseCode);
169
170         BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
171         String inputLine;
172         StringBuilder MailResponse = new StringBuilder();
173
174         while ((inputLine = in.readLine()) != null) {
175             MailResponse.append(inputLine);
176         }
177         in.close();
178     } catch (Exception e) {
179     }%>
180
181
182
183     <%@ include file="Footer.html" %>
184     <script src="JS/Common.js"></script>
185     <script src="JS/Email.js"></script>
186     <script src="JS/Feedback-Rating.js"></script>
187     </body>
188 </html>
189

```

OrderConfirmation.css

```
Start Page × DAO.java × OrderConfirmation.jsp [-/M] × Feedback-Rating.js × Email.js × OrderConfirmation.css ×
Source History |         
1  /*
2  Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/CascadeStyleSheet.css to edit this template
4  */
5  /*
6      Created on : Apr 11, 2024, 4:43:57 PM
7      Author      : Esanki Lakvindee
8  */
9
10 /*Import styles for stars*/
11 @import url(https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css);
12
13 body {
14     margin: 0;
15     padding: 0;
16     font-family: 'Inter', sans-serif;
17     background-color: #000; /* Updated background color */
18 }
19
20 .container {
21     display: flex;
22     justify-content: center;
23     align-items: center;
24     flex-direction: column;
25 }
26
27 .confirmation {
28     text-align: center;
29     color: #C2C3DB;
30     margin-bottom: 81px;
31     margin-top: 124px;
32     font-size: 28px;
33 }
34
35 .order-summary {
36     width: calc(100% - 254px); /* Adjusted width */
37     max-width: 1000px;
38     height: 500px;
39     margin: 0 127px; /* Added margin */
40     border-radius: 20px;
41     background: linear-gradient(0deg, rgba(255, 255, 255, 0.07) 0%, rgba(255, 255, 255, 0.07) 100%), linear-gradient(209deg, rgba(0, 0, 0, 0.25) 0px 0px 20px 0px rgba(0, 0, 0, 0.25));
42     box-shadow: 0px 0px 20px 0px rgba(0, 0, 0, 0.25);
43     backdrop-filter: blur(50px);
44     padding: 60px;
45     margin-top: 81px;
46     text-align: left;
47 }
48
49 .order-summary h2 {
50     color: #FFF;
51     font-size: 32px;
52     font-weight: 400;
53     margin-bottom: 32px;
54 }
55
56 .line{
57     width: 970px;
58     height: 3px;
59     background: rgba(59, 123, 248, 0.40);
60     margin-bottom: 32px;
61     margin-left: 20px;
62     margin-right: 20px;
63
64
65
66 }
```

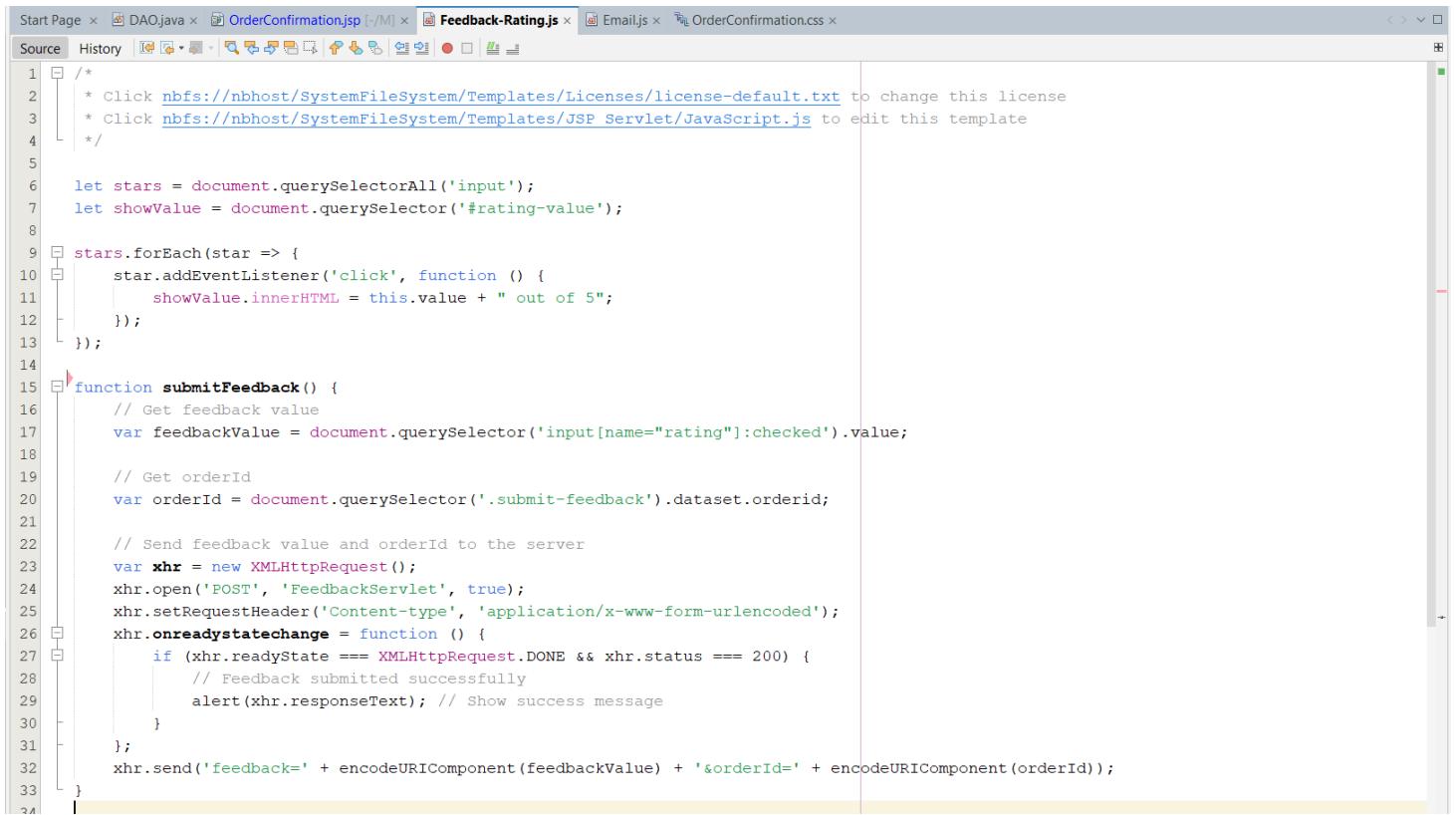
```
67  .item {
68    display: flex;
69    align-items: center;
70    margin-bottom: 32px;
71    font-size: 28px;
72  }
73
74  .item span {
75    flex: 1;
76    text-align: left;
77    font-size: 25px;
78    color: #FFF;
79  }
80
81  .name{
82    margin-right: 300px;
83  }
84
85  .button {
86    height: 80.294px;
87    flex-shrink: 0;
88    border-radius: 20px;
89  }
90
91  button:hover{
92    cursor: pointer !important;
93  }
94
95  .button1 {
96    width: 200px;
97    height: 65px; /* Added height */
98    flex-shrink: 0;
99    border: 1px solid #FFF;
100   background: #000;
101  }
102
103 .button2 {
104   width: 200px;
105   height: 65px;
106   flex-shrink: 0;
107   border-radius: 20px;
108   background: #151663;
109   margin-right: 235px;
110  }
111
112 .button1, .button2 {
113   color: #FFF;
114   font-size: 20px;
115  }
116
117 .text1, .text2 {
118   font-size: 25px;
119   color: #FFF;
120  }
121
122 .text1{
123   margin-right: 105px;
124  }
```

```

126 .feedback {
127   width: 700px;
128   height: 300px;
129   border-radius: 50px;
130   background: linear-gradient(180deg, #000 -44.82%, #151663 100%);
131   display: flex;
132   flex-direction: column;
133   justify-content: center;
134   align-items: center;
135   margin-top: 70px;
136 }
137
138 .feedback h3 {
139   color: #FFF;
140   font-family: 'calibri', sans-serif;
141   font-size: 32px;
142   font-weight: 400;
143   letter-spacing: 3.84px;
144 }
145
146 /* stars*/
147 .rating {
148   border: none;
149   float: left;
150 }
151
152 .rating > input {
153   display: none;
154 }
155
156 .rating > label:before {
157   content: '\f005';
158   font-family: FontAwesome;
159   margin: 5px;
160   font-size: 1.5rem;
161   display: inline-block;
162   cursor: pointer;
163 }
164
165 .rating > label {
166   color: #ddd;
167   float: right;
168   cursor: pointer;
169 }
170
171 .rating > input:checked ~ label,
172 .rating:not(:checked) > label:hover,
173 .rating:not(:checked) > label:hover ~ label {
174   color: yellow;
175 }
176
177 .rating > input:checked + label,
178 .rating > input:checked ~ label:hover,
179 .rating > label:hover ~ input:checked ~ label,
180 .rating > input:checked ~ label:hover ~ label {
181   color: yellow;
182 }
183
184 /*submit button*/
185 .submit-feedback {
186   background: none; /* Remove background */
187   border: none; /* Remove border */
188   color: #C0C0C0; /* Text color */
189   font-size: 14px; /* Font size */
190   cursor: pointer; /* Cursor on hover */
191   text-decoration: underline; /* Underline text */
192   transition: color 0.3s; /* Smooth transition for text color */
193 }
194
195 .submit-feedback:hover {
196   color:#A9A9A9; /* Darken text color on hover */
197 }
198
199

```

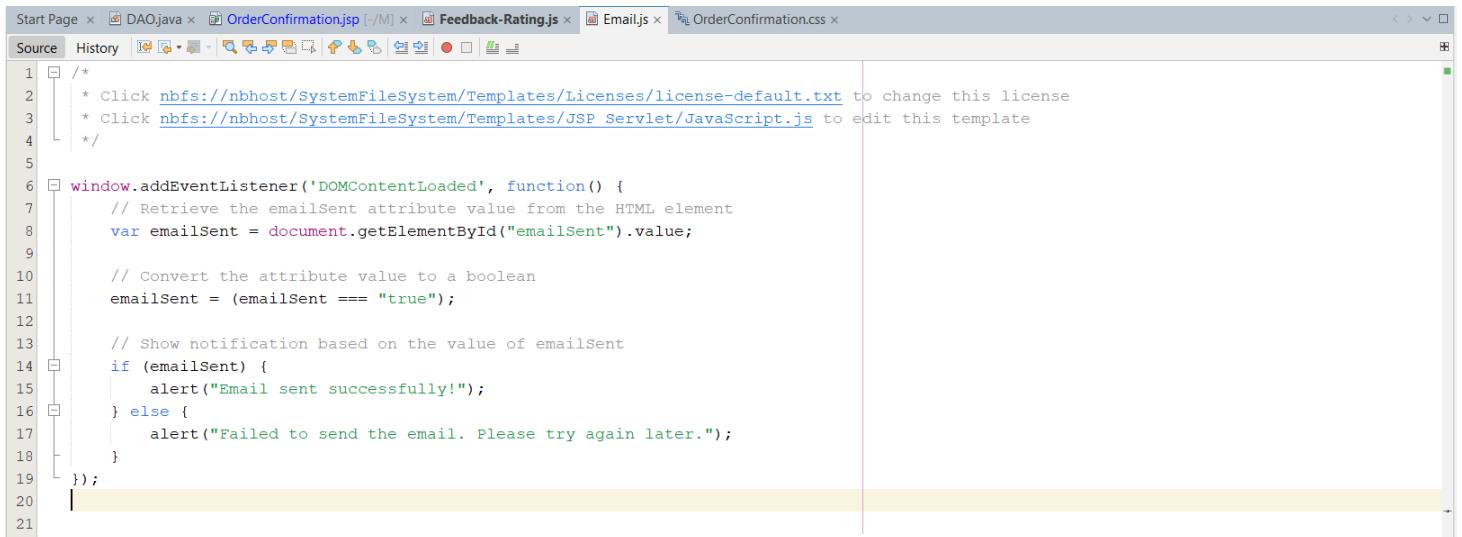
Feedback-Rating.js



The screenshot shows a Java IDE interface with multiple tabs at the top: Start Page, DAO.java, OrderConfirmation.jsp, Feedback-Rating.js (which is the active tab), Email.js, and OrderConfirmation.css. The Feedback-Rating.js tab contains the following code:

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/JavaScript.js to edit this template
4   */
5
6  let stars = document.querySelectorAll('input');
7  let showValue = document.querySelector('#rating-value');
8
9  stars.forEach(star => {
10    star.addEventListener('click', function () {
11      showValue.innerHTML = this.value + " out of 5";
12    });
13  });
14
15  function submitFeedback() {
16    // Get feedback value
17    var feedbackValue = document.querySelector('input[name="rating"]:checked').value;
18
19    // Get orderId
20    var orderId = document.querySelector('.submit-feedback').dataset.orderid;
21
22    // Send feedback value and orderId to the server
23    var xhr = new XMLHttpRequest();
24    xhr.open('POST', 'FeedbackServlet', true);
25    xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
26    xhr.onreadystatechange = function () {
27      if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {
28        // Feedback submitted successfully
29        alert(xhr.responseText); // Show success message
30      }
31    };
32    xhr.send('feedback=' + encodeURIComponent(feedbackValue) + '&orderId=' + encodeURIComponent(orderId));
33  }
34
```

Email.js



The screenshot shows a Java IDE interface with multiple tabs at the top: Start Page, DAO.java, OrderConfirmation.jsp, Feedback-Rating.js, Email.js (which is the active tab), and OrderConfirmation.css. The Email.js tab contains the following code:

```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/JavaScript.js to edit this template
4   */
5
6  window.addEventListener('DOMContentLoaded', function() {
7    // Retrieve the emailSent attribute value from the HTML element
8    var emailSent = document.getElementById("emailSent").value;
9
10   // Convert the attribute value to a boolean
11   emailSent = (emailSent === "true");
12
13   // Show notification based on the value of emailSent
14   if (emailSent) {
15     alert("Email sent successfully!");
16   } else {
17     alert("Failed to send the email. Please try again later.");
18   }
19 });
20
21
```

FeedbackServlet.java

The screenshot shows a Java IDE interface with the file `FeedbackServlet.java` open. The code implements a `HttpServlet` to handle feedback submission. It uses JDBC to update a MySQL database. The code includes annotations for web methods (`@WebServlet`, `@Override`, `@throws`) and imports various Java and Jakarta packages.

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template
4  */
5  package Controller;
6
7  import java.io.IOException;
8  import java.io.PrintWriter;
9  import java.sql.Connection;
10 import java.sql.DriverManager;
11 import java.sql.PreparedStatement;
12 import jakarta.servlet.ServletException;
13 import jakarta.servlet.annotation.WebServlet;
14 import jakarta.servlet.http.HttpServlet;
15 import jakarta.servlet.http.HttpServletRequest;
16 import jakarta.servlet.http.HttpServletResponse;
17
18 /**
19  *
20  * @author Esanki Lakvindee
21  */
22 @WebServlet(name = "FeedbackServlet", urlPatterns = {"/FeedbackServlet"})
23 public class FeedbackServlet extends HttpServlet {
24
25     @Override
26     protected void doGet(HttpServletRequest request, HttpServletResponse response)
27             throws ServletException, IOException {
28         /*
29         // Assuming you have a method to fetch order data from the database
30         List<Order> orders = fetchOrdersFromDatabase();
31
32         // Assuming each Order object has a feedback attribute
33         // Retrieve feedback data for each order
34
35         * @throws ServletException if a servlet-specific error occurs
36         * @throws IOException if an I/O error occurs
37         */
38     @Override
39     protected void doPost(HttpServletRequest request, HttpServletResponse response)
40             throws ServletException, IOException {
41         // Retrieve feedback value and orderId from request
42         String feedbackValue = request.getParameter("feedback");
43         String orderId = request.getParameter("orderId");
44
45         // Insert feedback into the database along with orderId
46         try {
47             Class.forName("com.mysql.cj.jdbc.Driver");
48             try (Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/emart", "root", "")) {
49                 PreparedStatement pst = con.prepareStatement("UPDATE orders SET feedback = ? WHERE orderId = ?");
50                 pst.setString(1, feedbackValue);
51                 pst.setString(2, orderId);
52                 pst.executeUpdate();
53             }
54         } catch (Exception e) {
55             e.printStackTrace();
56         }
57
58         // Send response back to client
59         response.setContentType("text/plain");
60         PrintWriter out = response.getWriter();
61         out.print("Feedback saved successfully");
62         out.flush();
63     }
64 }
65 }
```

DAO.java – getAdminOrders()

```
470  public static List<AdminOrderObj> getAdminOrders() {
471      List<AdminOrderObj> orders = new ArrayList<>();
472      Connection connection = null;
473      Statement statement = null;
474      ResultSet resultSet = null;
475      try {
476          Class.forName("com.mysql.cj.jdbc.Driver");
477
478          // Establishing connection to the database
479          connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
480
481          // Creating SQL statement
482          statement = connection.createStatement();
483
484          // SQL query to fetch from smartphone
485          String query = "SELECT o.*," +
486              + "CASE "
487              + " WHEN o.productId IN (SELECT productId FROM camera) THEN (SELECT productName FROM camera WHERE productId = o.pro
488              + " WHEN o.productId IN (SELECT productId FROM smartphone) THEN (SELECT productName FROM smartphone WHERE productId
489              + " WHEN o.productId IN (SELECT productId FROM smartwatch) THEN (SELECT productName FROM smartwatch WHERE productId
490              + " WHEN o.productId IN (SELECT productId FROM laptop) THEN (SELECT productName FROM laptop WHERE productId = o.pro
491              + " WHEN o.productId IN (SELECT productId FROM monitor) THEN (SELECT productName FROM monitor WHERE productId = o.p
492              + "END AS productName "
493              + "FROM orders o";
494
495          // Executing the query
496          resultSet = statement.executeQuery(query);
497
498          // Iterating over the result set and adding items from smartphone
499          while (resultSet.next()) {
500              int orderNumber = resultSet.getInt("orderId");
501              int productId = resultSet.getInt("productId");
502              int quantity = resultSet.getInt("quantity");
503
504              java.util.Date orderDate = resultSet.getDate("orderDate");
505              String orderStatus = resultSet.getString("orderStatus");
506              int totalPrice = resultSet.getInt("totalPrice");
507              String shippingAddress = resultSet.getString("address");
508              String customerName = resultSet.getString("name");
509              String feedback = resultSet.getString("feedback");
510              String email = resultSet.getString("email");
511              String productName = resultSet.getString("productName"); // Retrieve product name
512
513              // Creating Smartphone object and adding it to the list
514              AdminOrderObj order = new AdminOrderObj(orderDate, orderNumber, orderStatus, productId, quantity,
515                  totalPrice, shippingAddress, customerName, feedback, email);
516              order.setProductName(productName);
517              orders.add(order);
518          }
519
520      } catch (Exception e) {
521          System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
522      } finally {
523          // Closing the connection, statement, and result set
524          try {
525              if (resultSet != null) {
526                  resultSet.close();
527              }
528              if (statement != null) {
529                  statement.close();
530              }
531              if (connection != null) {
532                  connection.close();
533              }
534          } catch (SQLException e) {
535              System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
536          }
537      }
538  }
```

DAO.java – getOrderConfirmation()

```
415  public static List<ShoppingCartObj> getOrderConfirmationDetails() {  
416      return getAllCartItems();  
417  }
```

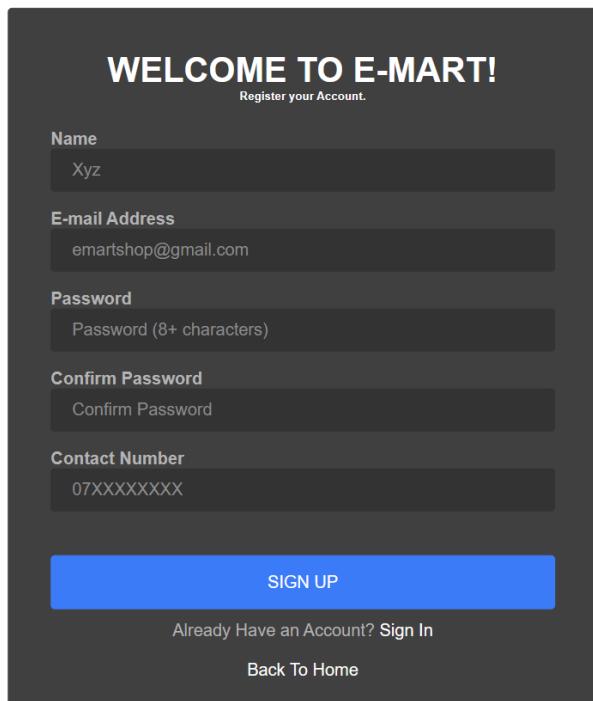
Conclusion

The Order Confirmation Page is essential for providing users with a smooth and informative post-purchase experience. The page improves user satisfaction and engagement with the e-commerce platform by generating dynamic content, providing intuitive user interface elements, and integrating with external services such as email notifications.

1.3 Sign Up page

Done by : S.N.M. Gedara
Student ID : 29165

1.3.1 Web page



The screenshot shows the sign-up form for the E-Mart platform. The form is titled "WELCOME TO E-MART!" and includes a sub-instruction "Register your Account." Below the title, there are five input fields: "Name" (with "Xyz" entered), "E-mail Address" (with "emartshop@gmail.com" entered), "Password" (with "Password (8+ characters)" placeholder), "Confirm Password" (with "Confirm Password" placeholder), and "Contact Number" (with "07XXXXXXXX" entered). At the bottom of the form is a large blue "SIGN UP" button. Below the button, there is a link "Already Have an Account? [Sign In](#)" and a link "Back To Home".

1.3.2 Documentation: Sign Up Page Implementation

Sign Up Servlet (SignUpServlet.java):

- This servlet handles user sign-up functionality.
- It receives form data from a sign-up form (e.g., username, email, password, contact number) via HTTP POST request.
- Validates the input data to ensure it meets certain criteria (e.g., not empty, password length, matching passwords, valid contact number).
- If validation passes, it inserts the user details into a database using the userDao class.
- Generates a unique UserID using UUID.
- Sets session attributes with user details and creates a cookie with the UserID.
- Finally, redirects the user to the profile page.

User Data Access Object (userDao.java):

- This class handles interactions with the database for user-related operations.
- It provides methods to insert user details into the database.
- Establishes a connection to the MySQL database using JDBC.

Frontend Styling (CSS):

- The CSS code provided styles the sign-up form.
- It sets up a responsive layout for the form elements, ensuring they look consistent across different screen sizes.
- Defines styles for form controls like input fields, buttons, labels, etc.
- Uses background images and colors to enhance the visual appeal of the form.

1.3.3 Codes

SignUp.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>E-mart SignUp</title>
    <link href="CSS/SignUp.css" rel="stylesheet" type="text/css"/>
    <script>

function validation() {
    var username = document.getElementById("username").value;
    var email = document.getElementById("email").value;
    var password = document.getElementById("password").value;
    var confirmPassword = document.getElementById("confirmPassword").value;
    var contactNumber = document.getElementById("ContactNumber").value;

    if (!username || !email || !password || !confirmPassword || !contactNumber) {
        alert("All fields must be filled out.");
        return false;
    }

    if (password.length < 8) {
        alert("Password must be at least 8 characters long.");
        return false;
    }

    if (password !== confirmPassword) {
        alert("Passwords do not match.");
        return false;
    }

    if (!/\d{10}/.test(contactNumber)) {
        alert("Contact number must be a 10-digit number.");
        return false;
    }

    return true;
}

</script>
</head>
<body>
<div class="form-wrapper">
    <h2 style="text-align: center;">WELCOME TO E-MART!</h2>
    <h6 style="color: white; text-align: center;">Register your Account.</h6>

    <form action="SignUpServlet" onsubmit="return validation()" method="post">
        <h4> <label for="username">Name</label></h4>
        <div class="form-control">
            <input type="text" id="username" name="username" required>
            <label>Xyz</label>
        </div>

        <h4> <label for="email">E-mail Address</label></h4>
        <div class="form-control">
            <input type="text" id="email" name="email" required>
            <label>emartshop@gmail.com</label>
        </div>

        <h4> <label for="password">Password</label></h4>
        <div class="form-control">
            <input type="password" id="password" name="password" required>
            <label>Password (8+ characters)</label>
        </div>

        <h4> <label for="Confirmpassword">Confirm Password</label></h4>
        <div class="form-control">
            <input type="password" id="confirmPassword" name="confirmPassword" required>
            <label>Confirm Password</label>
        </div>
    </form>
</div>
```

```

<h4><label for="ContactNumber">Contact Number</label></h4>

<div class="form-control">
<input type="text" id="ContactNumber" name="ContactNumber" required>
<label>07XXXXXXX</label>
</div>

<button class="" type="submit">SIGN UP</button>
<br>
<p style="text-align: center;">Already Have an Account? <a href="Signin.jsp">Sign In</a>
</p>
<br>
<p style="text-align: center;"><a href="index.jsp">Back To Home</a>
</p>
</form>
</div>

</body>
</html>

```

SignUp.css

```

* {
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: 'Roboto', sans-serif;
}

body {
background: #fff;
}

body::before {
content: "";
position: absolute;
left: 0;
top: 0;
opacity: 80%;
width: 100%;
height: 100%;
background: url("images/device-img.jpg") no-repeat;
background-position: center center;
background-size: cover;
filter: brightness(30%);
}

.form-wrapper {
position: absolute;
left: 50%;
top: 50%;
border-radius: 4px;
padding: 40px;
width: 550px;
height: 650px;
transform: translate(-50%, -50%);
background: rgba(0, 0, 0, .75);
}

```

```
.form-wrapper h2 {  
  color: #fff;  
  font-size: 2rem;  
}  
  
.form-wrapper form {  
  margin: 25px 0 65px;  
}  
  
form .form-control {  
  height: 40px;  
  position: relative;  
  margin-bottom: 16px;  
}  
  
.form-control input {  
  height: 100%;  
  width: 100%;  
  background: #333;  
  border: none;  
  outline: none;  
  border-radius: 4px;  
  color: #fff;  
  font-size: 1rem;  
  padding: 0 20px;  
}  
  
.form-control input:focus,  
.form-control input:valid {  
  background: #444;  
  padding: 16px 20px 0;  
}  
  
.form-control label {  
  position: absolute;  
  left: 20px;  
  top: 50%;  
  transform: translateY(-50%);  
  font-size: 1rem;  
  pointer-events: none;  
  color: #8c8c8c;  
  transition: all 0.1s ease;  
}  
  
.form-control input:focus-label,  
.form-control input:valid-label {  
  font-size: 0.75rem;  
  transform: translateY(-130%);  
}  
  
form button {  
  width: 100%;  
  padding: 16px 0;  
  font-size: 1rem;  
  background: #3B7BF8;  
  color: #fff;  
  font-weight: 500;  
  border-radius: 4px;  
  border: none;  
  outline: none;  
  margin: 25px 0 10px;  
  cursor: pointer;  
  transition: 0.1s ease;  
}
```

```
form button:hover{  
background:#151663;  
}  
  
.form-wrapper a{  
text-decoration: none;  
}  
  
.form-wrapper a:hover{  
text-decoration: underline;  
}  
  
.form-wrapper :where(label, p, small, a){  
color:#b3b3b3;  
}  
  
form .form-help{  
display: flex;  
justify-content: space-between;  
}  
  
form .remember-me{  
display: flex;  
}  
  
form .remember-me input{  
margin-right:5px;  
accent-color:#b3b3b3;  
}  
  
form .form-help :where(label, a){  
font-size: 0.9rem;  
}  
  
.form-wrapper p a{  
color: #fff;  
}  
  
.form-wrapper small{  
display: block;  
margin-top: 15px;  
color: #b3b3b3;  
}  
  
.form-wrapper small a{  
color: #0071eb;  
}  
  
@media (max-width: 740px){  
body::before{  
display: none;  
}  
  
nav, form-wrapper{  
padding: 20px;  
}  
  
nav a img{  
width: 140px;  
}  
  
.form-wrapper{  
width: 100%;  
top: 43%;  
}  
  
.form-wrapper form{  
margin: 25px 20px 40px;  
}  
}
```

SignUpServlet.java

```
package Controller;

import Model.SignUpUser;
import Model.userDao;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.UUID;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
@WebServlet(name = "SignUpServlet", urlPatterns = {"/SignUpServlet"})
public class SignUpServlet extends HttpServlet {

    /**
     * @Override
     protected void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
        //processRequest(request, response);

        //Get data
        String username = request.getParameter("username");
        String email = request.getParameter("email");
        String password = request.getParameter("password");
        String confirmPassword = request.getParameter("confirmPassword");
        String contactNumber = request.getParameter("ContactNumber");

        // If Validation fails
        if (isValid(username, email, password, confirmPassword, contactNumber)) {
            request.getRequestDispatcher("SignUp.jsp").forward(request, response);
        } else {

            // Create user object
            SignUpUser user = new SignUpUser(username, password, contactNumber);
            // Save to the database
            userDao userDao = new userDao(); // Create an instance of userDao
            userDao.insertDetails(email, username, password, contactNumber);
            // Generate unique UserID
            String userID = UUID.randomUUID().toString();
            // Set session attributes
            HttpSession session = request.getSession();
            session.setAttribute("username", username.trim());
            session.setAttribute("email", email.trim());
            session.setAttribute("password", password.trim());
            session.setAttribute("contactNumber", contactNumber.trim());

            // Set cookie with UserID
            Cookie cookie = new Cookie("UserID", userID);
            cookie.setMaxAge(24 * 60 * 60);

            // Redirect to profile page
            response.sendRedirect("ProfileServlet?email=" + email);
        }
    }
}
```

```

        }

    }

    /**
     * Returns a short description of the serverlet.
     *
     * @return a String containing serverlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    } // </editor-fold>

    private boolean validation(String username, String email, String password, String confirmPassword, String contactNumber) {
        if (username.isEmpty() || email.isEmpty() || password.isEmpty() || confirmPassword.isEmpty() || contactNumber.isEmpty()) {
            return false;
        }

        if (password.length() < 8) {
            return false;
        }

        if (!password.equals(confirmPassword)) {
            return false;
        }

        return contactNumber.matches(regex: "\d{10}");
    }
}

```

userDAO.java

```

package Model;

/**
 *
 * @author DELL
 */
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.sql.*;
public class userDao {

    static Statement st;
    private static Connection con;

    public static void insertDetails(String email, String username, String password, String contactNumber) {
        connectToDB();
        String query = "INSERT INTO users(email,username,password,contactNumber) VALUES('" + email + "','" + username + "','" + password + "','" + contactNumber + "')";
        try {
            st.executeUpdate(string: query);
            System.out.println("Record inserted");
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }

    public static void connectToDB() {
        String driver = "com.mysql.cj.jdbc.Driver";
        String url = "jdbc:mysql://localhost:3306/emart";

        try {
            Class.forName(className: driver);
            // Establish the connection
            con = DriverManager.getConnection(url, user: "root", password: "");
            // Statement st=con.createStatement();
            st = con.createStatement();
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(userDao.class.getName()).log(level: Level.SEVERE, msg: null, throws: ex);
        }
    }
}

```

1.4 Sign In page

Done by : S.N.M. Gedara
Student ID : 29165

1.4.1 Web page

The screenshot shows a dark-themed sign-in form titled "SIGN IN". It has two input fields: "Email" containing "emartshop@gmail.com" and "Password" with a placeholder "8+ characters". Below the password field is a link "Forgot password?". A large blue "SIGN IN" button is centered below the inputs. To its left is a checkbox labeled "Remember me". At the bottom, there are links for new users: "New to E-Mart? Sign up now" and "Back To Home".

1.4.2 Documentation : Sign In page implementation

Sign In JSP (SignIn.jsp):

- This JSP file represents the sign-in form.
- It collects user input for email and password.
- Upon submission, it sends a POST request to the SignInServlet.
- It also includes links for password recovery and new user registration.

Sign In Servlet (SignInServlet.java):

- This servlet handles user sign-in requests.
- Upon receiving a POST request from the sign-in form, it retrieves the email and password parameters.
- It validates whether both email and password are provided.

- It establishes a connection to the database using JDBC, queries the database to check if the provided credentials match any user records.
- If a matching user is found, it generates a unique UserID using UUID, sets session attributes with the user's email, and creates a cookie with the UserID.
- It then redirects the user to the profile page.
- If no matching user is found, it redirects the user back to the sign-in page with a message indicating failure.

1.4.3 Codes

SignIn.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>E-mart SignIn</title>
    <link href="CSS/SignIn.css" rel="stylesheet" type="text/css"/>
</head>

<div class="form-wrapper">
    <h2 style="text-align: center;">SIGN IN</h2>
    <form action="SignInServlet" method="post">
        <h4><label for="email">Email</label></h4>
        <br>
        <div class="form-control">
            <input type="text" id="email" name="email" required>
            <label>smartshop@gmail.com</label>
        </div>
        <br>
        <h4><label for="password">Password</label></h4>
        <br>
        <div class="form-control">
            <input type="password" id="password" name="password" required>
            <label>8+ characters</label>
        </div>
        <a href="ForgotPassword.jsp">Forgot password?</a>
        <br>
        <button class="" type="submit">SIGN IN</button>
    </form>
</div>
```

```

<button class="" type="submit">SIGN IN</button>

<div class="remember-me">
  <input type="checkbox" id="remember-me">
  <label for="remember-me">Remember me</label>
</div>
<br>
<p style="text-align: center;">New to E-Mart? <a href="SignUp.jsp">Sign up now</a></p>
<br>
<p style="text-align: center;"><a href="Home.jsp">Back To Home</a></p>
<br>
</form>
</div>
</body>
</html>

```

SignIn.css

```

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: 'Roboto', sans-serif;
}

body
{
  background: #fff;
}

body::before
{
  content: "";
  position: absolute;
  left: 0;
  top: 0;
  opacity: 70%;
  width: 100%;
  height: 100%;
  background-position: center center;
  background-size: cover;
  filter: brightness(50%);
}

.form-wrapper
{
  position: absolute;
  left: 50%;
  top: 50%;
  border-radius: 4px;
  padding: 70px;
  width: 450px;
  height: 540px;
  transform: translate(-50%, -50%);
  background: rgba(0, 0, 0, .75);
}

```

```
.form-wrapper h2 {
  color: #fff;
  font-size: 2rem;
}

.form-wrapper form
{
  margin: 25px 0 65px;
}

form.form-control
{
  height: 40px;
  position: relative;
  margin-bottom: 16px;
}
form.remember-me
{
  display: flex;
}

form.remember-me input
{
  margin-right: 5px;
  accent-color: #b3b3b3;
}
.form-control input
{
  height: 100%;
  width: 100%;
  background: #333;
  border: none;
  outline: none;
  border-radius: 4px;
  color: #fff;
  font-size: 1rem;
  padding: 0 20px;
}

.form-control input:focus,
.form-control input:valid
{
  background: #444;
  padding: 16px 20px 0;
}

.form-control label
{
  position: absolute;
  left: 20px;
  top: 50%;
  transform: translateY(-50%);
  font-size: 1rem;
  pointer-events: none;
  color: #8c8c8c;
  transition: all 0.1s ease;
}

.form-control input:focus-label,
.form-control input:valid-label
{
  font-size: 0.75rem;
  transform: translateY(-130%);
}
```

```
form button
{
    width: 100%;
    padding: 16px 0;
    font-size: 1rem;
    background: #3B7BF8;
    color: #fff;
    font-weight: 500;
    border-radius: 4px;
    border: none;
    outline: none;
    margin: 25px 0 10px;
    cursor: pointer;
    transition: 0.1s ease;
}

form button:hover
{
    background: #151663;
}

.form-wrapper a
{
    text-decoration: none;
}

.form-wrapper a:hover
{
    text-decoration: underline;
}

.form-wrapper .where(label, p, small, a)
{
    color: #b3b3b3;
}

form .form-help
{
    display: flex;
    justify-content: space-between;
}

form .remember-me
{
    display: flex;
}

form .remember-me input
{
    margin-right: 5px;
    accent-color: #b3b3b3;
}

form .form-help .where(label, a)
{
    font-size: 0.9rem;
}

.form-wrapper p a
{
    color: #fff;
}
```

```
@media (max-width: 740px)
{
    body::before {
        display: none;
    }

    nav, .form-wrapper {
        padding: 20px;
    }

    nav a img {
        width: 140px;
    }

    .form-wrapper {
        width: 100%;
        top: 43%;
    }

    .form-wrapper form {
        margin: 25px 20px 40px;
    }
}
```

SignInServlet.java

```
package Controller;

import java.sql.Connection;
import jakarta.resource.cci.ResultSet;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.PreparedStatement;
import java.util.UUID;

/**
 * 
 * @author DELL
 */
@WebServlet(name = "SignInServlet", urlPatterns = {"/SignInServlet"})
public class SignInServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        // Retrieve email and password from the form
        String email = request.getParameter("email");
        String password = request.getParameter("password");

        // Validate email and password
        if (email == null || email.isEmpty() || password == null || password.isEmpty()) {
            out.println("Please provide both email and password.");
            return;
        }
    }
}
```

```

// Database connection parameters
String url = "jdbc:mysql://localhost:3306/emart"; // database name

try {
    // Load the MySQL driver
    Class.forName(className: "com.mysql.jdbc.Driver");

    // Establish the database connection
    Connection con = DriverManager.getConnection(url, user: "root", password: "");

    // Prepare the SQL statement
    String sql = "SELECT * FROM users WHERE email=? AND password=?";
    PreparedStatement statement = con.prepareStatement(string: sql);
    statement.setString(1, string: email.trim());
    statement.setString(2, string: password.trim());

    // Execute the query
    var rs = statement.executeQuery();
    out.println(rs);

    if (rs.next()) {
        String userID = UUID.randomUUID().toString();
        // User exists, you can perform further actions here like setting session attributes or redirecting to a different page.
        HttpSession session = request.getSession();
        session.setAttribute(string: "email", o:email.trim());

        // Set cookie with UserID
        Cookie cookie = new Cookie(name: "UserID", value: userID);
        cookie.setMaxAge(24 * 60 * 60);

        // Redirect to profile page
        response.sendRedirect("ProfileServlet?email=" + email);
    } else {
        // User doesn't exist or provided credentials are incorrect.
        response.sendRedirect(string: "SignIn.jsp?message=fails");
    }

    // Close resources
    rs.close();
    statement.close();
    con.close();
} catch (ClassNotFoundException e) {
    out.println("MySQL JDBC Driver not found.");
} catch (SQLException e) {
    out.println("SQL Exception: " + e.getMessage());
}
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return "Short description";
} // </editor-fold>
}

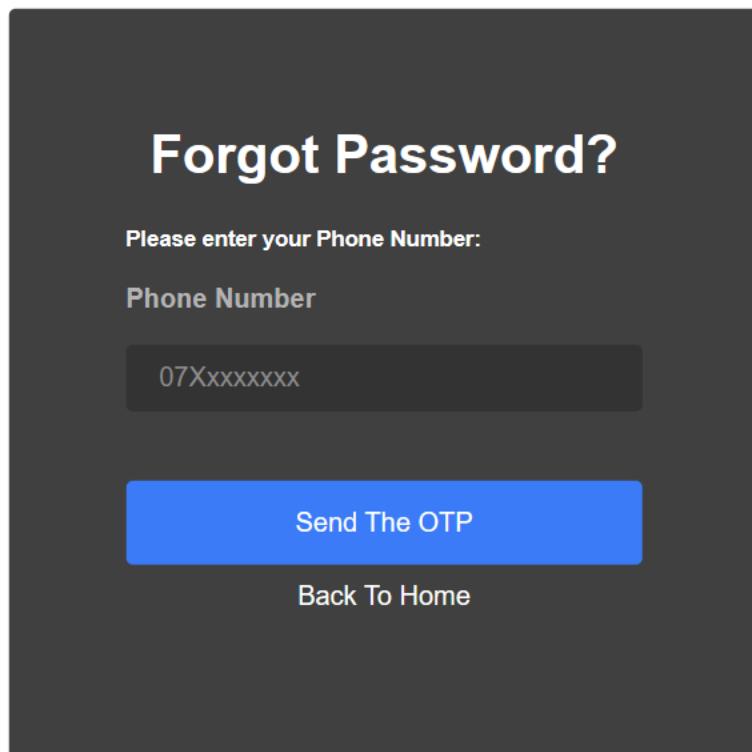
```

1.5 Forgot Password page

Done by : S.N.M. Gedara

Student ID : 29165

1.5.1 Web page



1.5.2 Documentation : Forgot password page implementation

Forget Password JSP (ForgetPassword.jsp):

- This JSP file displays a form where users can enter their email address or phone number to receive a verification code for resetting their password.
- It includes client-side validation using JavaScript to ensure that the entered email address is valid.
- Upon submission, the form sends a POST request to the SendVerificationCodeServlet.

Forget Password CSS (ForgotPassword.css):

- This CSS file contains styles to enhance the appearance of the forget password form.

Send Verification Code Servlet (SendVerificationCodeServlet.java):

- This servlet handles the logic for sending a verification code to the user's email or phone number.
- It receives the phone number from the submitted form.
- It constructs a URL with the necessary parameters (API key, API secret, user's phone number) to send a verification code using the Nexmo API.
- It makes a GET request to the Nexmo API endpoint to send the verification code.
- It extracts the request ID from the API response using regex.
- It sets the request attribute with the request ID and forwards the request to a JSP page (verifyOTP.jsp) where users can enter the verification code.

1.5.3 Codes

ForgotPassword.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Forgot Password</title>
    <link href="CSS/ForgotPassword.css" rel="stylesheet" type="text/css"/>
    <script type="text/javascript">

$(document).ready(function() {
  $("#resetForm").validate({
    rules: {
      email: {
        required: true,
        email: true
      },
      messages: {
        email: {
          required: "Please enter email",
          email: "Please enter a valid email address"
        }
      }
    });
  });
</script>
</head>
<div class="form-wrapper">
  <h2 style="text-align: center;">Forgot Password?</h2>
  <form action="SendVerificationCodeServlet" method="post">

    <h5 style="color:white;">
      Please enter your Phone Number:</h5>
    </h5>
    <br>
    <h4> <label for="mobileNumber" name="mobileNumber">Phone Number</label></h4>
    <br>
    <div class="form-control">
      <input type="text" id="mobileNumber" name="mobileNumber" required>
      <label>07xxxxxxxx</label>
    </div>

    <button class="" type="submit">Send The QIP</button>
    <p style="text-align: center;"><a href="Home.jsp">Back To Home</a>

  </form>
</div>
</body>
</html>
```

ForgotPassword.css

```
* {  
    margin: 0;  
    padding: 0;  
    box-sizing: border-box;  
    font-family: 'Roboto', sans-serif;  
}
```

```
body {  
    background: #fff;  
}
```

```
.form-wrapper
```

```
{  
    position: absolute;  
    left: 50%;  
    top: 50%;  
    border-radius: 4px;  
    padding: 70px;  
    width: 450px;  
    height: 450px;  
    transform: translate(-50%, -50%);  
    background: rgba(0, 0, 0, .75);  
}
```

```
.form-wrapper h2 {  
    color: #fff;  
    font-size: 2rem;  
}
```

```
.form-wrapper form {  
    margin: 25px 0 65px;  
}
```

```
form .form-control {  
    height: 40px;  
    position: relative;  
    margin-bottom: 16px;  
}
```

```
form .remember-me {  
    display: flex;  
}
```

```
form .remember-me input {  
    margin-right: 5px;  
    accent-color: #b3b3b3;  
}
```

```
.form-control input {  
    height: 100%;  
    width: 100%;  
    background: #333;  
    border: none;  
    outline: none;  
    border-radius: 4px;  
    color: #fff;  
    font-size: 1rem;  
    padding: 0 20px;  
}
```

```
.form-control input:focus,
```

```
.form-control input:valid {  
    background: #444;  
    padding: 16px 20px 0;  
}
```

```
.form-control label
{
  position: absolute;
  left: 20px;
  top: 50%;
  transform: translateY(-50%);
  font-size: 1rem;
  pointer-events: none;
  color: #8c8c8c;
  transition: all 0.1s ease;
}

.form-control input:focus-label,
.form-control input:valid-label
{
  font-size: 0.75rem;
  transform: translateY(-130%);
}

form button
{
  width: 100%;
  padding: 16px 0;
  font-size: 1rem;
  background: #3B7BF8;
  color: #fff;
  font-weight: 500;
  border-radius: 4px;
  border: none;
  outline: none;
  margin: 25px 0 10px;
  cursor: pointer;
}

form button:hover
{
  background: #151663;
}

.form-wrapper a
{
  text-decoration: none;
}

.form-wrapper a:hover
{
  text-decoration: underline;
}

.form-wrapper .where(label, p, small, a)
{
  color: #b3b3b3;
}

form .form-help
{
  display: flex;
  justify-content: space-between;
}

form .remember-me
{
  display: flex;
}

form .remember-me input
{
  margin-right: 5px;
  accent-color: #b3b3b3;
}
```

```
form .form-help .where(label,a)
{
    font-size:0.9rem;
}

.form-wrapper p a
{
    color: #fff;
}

@media (max-width: 740px)
{
    body::before {
        display: none;
    }

    nav, .form-wrapper {
        padding: 20px;
    }

    nav a img {
        width: 140px;
    }

    .form-wrapper {
        width: 100%;
        top: 43%;
    }

    .form-wrapper form {
        margin: 25px 20px 40px;
    }
}
```

SendVerificationCodeServlet.java

```
package Controller;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * @author DELL
 */
@WebServlet(name = "SendVerificationCodeServlet", urlPatterns = {"~/SendVerificationCodeServlet"})
public class SendVerificationCodeServlet extends HttpServlet {
```

```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
] throws ServletException, IOException {
//processRequest(request, response);
// Get the user's number from the request
String userNumber = request.getParameter("mobileNumber");

String API_KEY = "942f32b7";
String API_SEC = "pkYTozjIXO4qAjeK";
String UsersNumber = "94722646386";

String URL_TO_VERIFY = "https://api.nexmo.com/verify/json?&api_key=" + API_KEY + "&api_secret=" + API_SEC + "&number=" + UsersNumber + "&brand=AcmeInc";

try {
URL url = new URL(spec.URL_TO_VERIFY);

HttpURLConnection connection = (HttpURLConnection) url.openConnection();
connection.setRequestMethod("GET");
StringBuilder apiResponse = new StringBuilder();
BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));
String line;
while ((line = reader.readLine()) != null) {
apiResponse.append(line);
}

// Extract request_id from response
System.out.println(response.toString());
// Extract request_id from response
String requestId = extractRequestId(response.toString());
System.out.println("Request ID: " + requestId);

// Set request attributes
request.setAttribute("requestId", requestId);

// Forward the request to verifycode.jsp
request.getRequestDispatcher("verify.jsp").forward(request, response);

} catch (Exception ex) {
System.out.println(ex);
}
}

// Method to extract request ID from Nexmo API response
private static String extractRequestId(String response) {
// Using regex to extract request_id
Pattern pattern = Pattern.compile(regex: "\r\nrequest_id": "(.*?)" );
Matcher matcher = pattern.matcher(response);
if (matcher.find()) {
return matcher.group(1);
}
return response;
}

/**
 * Returns a short description of the server.
 *
 * @return a String containing server description
 */
@Override
public String getServletInfo() {
return "Short description";
}// </editor-fold>

}

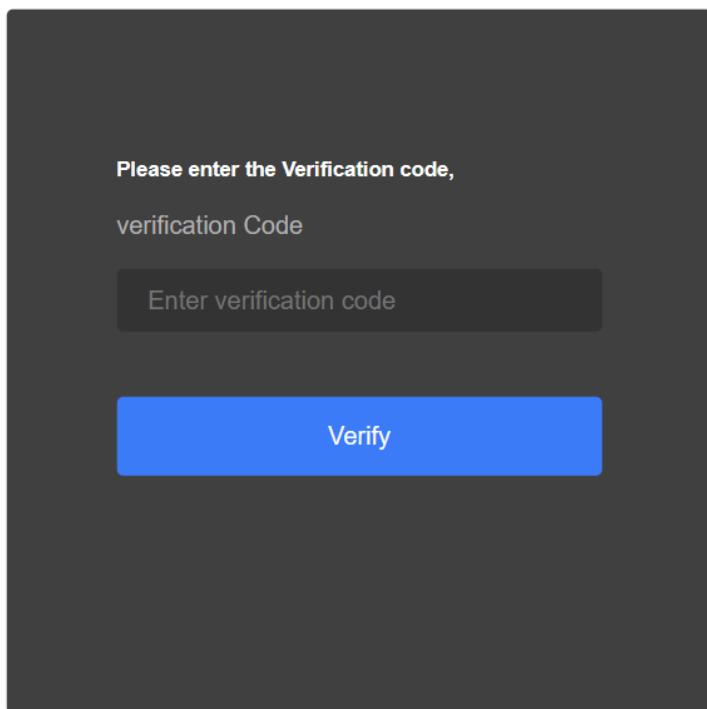
```

1.6 Verify OTP page

Done by : S.N.M. Gedara

Student ID : 29165

1.6.1 Web page



1.6.2 Documentation : VerifyOTP page implementation

Verify OTP JSP (VerifyOTP.jsp):

- This JSP file displays a form where users can enter the verification code received via SMS.
- It includes a form field for entering the verification code.
- Upon submission, the form sends a POST request to the VerifyCodeServlet.

Verify Code Servlet (VerifyCodeServlet.java):

- This servlet handles the logic for verifying the OTP entered by the user.
- It retrieves the user-entered OTP from the request parameters.
- It retrieves the stored request ID (received during OTP generation) from the session.

- It constructs a URL with the necessary parameters (API key, API secret, request ID, and user-entered OTP) to verify the OTP using the Nexmo API.
- It makes a GET request to the Nexmo API endpoint to verify the OTP.
- It checks the response from the Nexmo API to determine whether the verification was successful or not.
- If the verification is successful, it typically redirects the user to a page where they can reset their password (resetPassword.jsp). Otherwise, it may redirect the user back to the verification page (verifyOTP.jsp) or display an error message.

1.6.3 Codes

VerifyOTP.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Forgot Password</title>
    <link href="CSS\ForgotPassword.css" rel="stylesheet" type="text/css"/>
  </head>
  <body>

    <% String requestId = (String) request.getAttribute("requestId"); %>

    <div class="form-wrapper">

      <form action="VerifyCodeServlet" method="post">

        <h5 style="color:white;">
          Please enter the Verification code,
        </h5>
        <br>
        <label for="text" name="verificationCode">verification Code</label></h4>
        <br>
        <br>
        <div class="form-control">
          <input type="text" name="verificationCode" placeholder="Enter verification code" required>
        </div>

        <button class="" type="submit">Verify</button>

      </form>
    </div>
  </body>
</html>
```

VerifyCodeServlet.java

```
package Controller;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.URL;

/**
 * 
 * @author DELL
 */
@WebServlet(name = "VerifyCodeServlet", urlPatterns = {"/VerifyCodeServlet"})
public class VerifyCodeServlet extends HttpServlet {
```

```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //processRequest(request, response);
    // Get the user's entered OTP from the request
    String userEnteredOTP = request.getParameter("otp");

    // Get the stored request ID from the session
    String storedRequestId = (String) request.getSession().getAttribute("requestId");

    // Nexmo API credentials
    String API_KEY = "942f32b7";
    String API_SEC = "pkYTozjXO4qAjeK";

    // Construct the URL to verify the OTP with Nexmo
    String URL_TO_VERIFY = "https://api.nexmo.com/verify/check/json?" +
        "&api_key=" + API_KEY +
        "&api_secret=" + API_SEC +
        "&request_id=" + storedRequestId +
        "&code=" + userEnteredOTP;

    try {
        URL url = new URL(URL_TO_VERIFY);
        HttpURLConnection connection = (HttpURLConnection) url.openConnection();
        connection.setRequestMethod("GET");

        // Get the response from the Nexmo API
        StringBuilder apiResponse = new StringBuilder();
        try (BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()))) {
            String line;
            while ((line = reader.readLine()) != null) {
                apiResponse.append(line);
            }
        }

        // Check if the response contains patterns indicating success or failure
        if (apiResponse.toString().contains("status": "0")) {
            // Verification successful
            // Perform the desired action (e.g., allow access, redirect to a success page)
            response.sendRedirect("resetPassword.jsp");
        } else {
            // Verification failed
            // Handle the error (e.g., display an error message, redirect to a failure page)
            response.sendRedirect("verify.jsp");
        }
    } catch (Exception ex) {
        System.out.println(ex);
    }
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return "Short description";
} // </editor-fold>
}

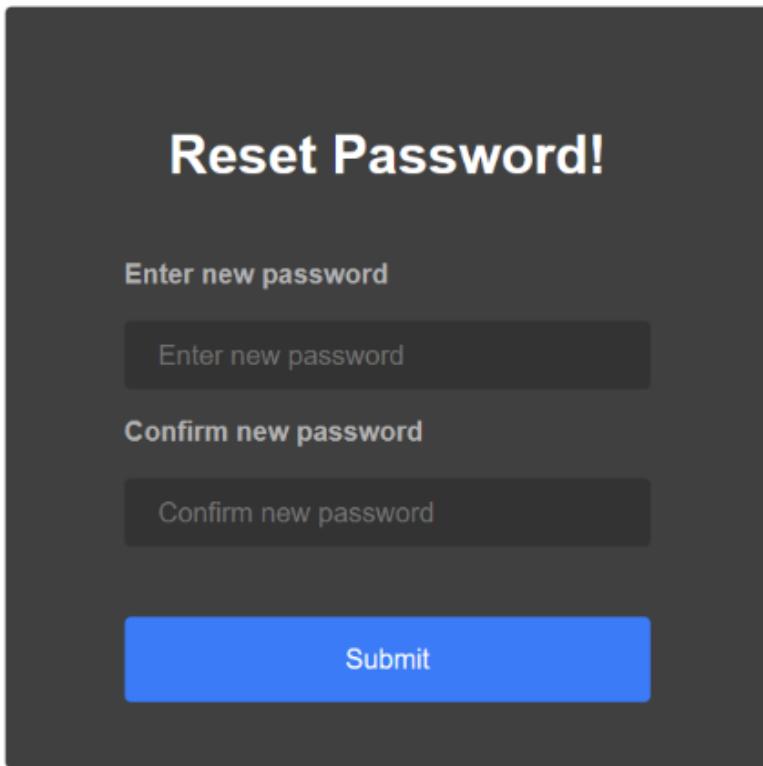
```

1.7 Reset password page

Done by : S.N.M. Gedara

Student ID : 29165

1.7.1 Web page



1.7.2 Documentation : Reset password page implementation

Reset Password JSP (resetPassword.jsp):

- This JSP file displays a form where users can enter their new password and confirm it.
- It includes form fields for entering the new password and confirming it.
- Upon submission, the form sends a POST request to the UpdatePasswordServlet.

Update Password Servlet (UpdatePasswordServlet.java):

- This servlet handles the logic for updating the user's password in the database.
- It retrieves the user's email from the session (assuming it was set during the verification process).
- It retrieves the new password and confirmed new password from the request parameters.
- It constructs an SQL query to update the user's password in the database.
- It executes the SQL query to update the password.

1.7.3 Codes

resetPassword.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>reset Password</title>
        <link href="CSS\ForgotPassword.css" rel="stylesheet" type="text/css"/>
    </head>
    <div class="form-wrapper">
        <h2 style="text-align: center;">Reset Password!</h2>
        <form action="UpdatePasswordServlet" method="post">
            <br>
            <h4> <label for="mobileNumber" name="mobileNumber">Enter new password</label></h4>
            <br>
            <div class="form-control">
                <input type="password" name="password" placeholder="Enter new password" required>
            </div>
            <h4> <label for="Confirmnewpassword" name="Confirmnewpassword">Confirm new password</label></h4>
            <br>
            <div class="form-control">
                <input type="password" name="confirmPassword" placeholder="Confirm new password" required>
            </div>
            <button class=" " type="submit">Submit</button>
        </form>
    </div>
</body>
</html>
```

UpdatePasswordServlet.java

```
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
/*
 * @author DELL
 */
@WebServlet(urlPatterns = {" /UpdatePasswordServlet"})
public class UpdatePasswordServlet extends HttpServlet {
```

```

public Statement st; // Declare Statement as a member variable

@Override
public void init() throws ServletException {
    super.init();
    try {
        // Initialize the Statement in the init method
        String driver = "com.mysql.cj.jdbc.Driver";
        String url = "jdbc:mysql://localhost:3306/emart";
        Class.forName(className: driver);
        Connection con = DriverManager.getConnection(url, user: "root", password: "");
        st = con.createStatement();
    } catch (ClassNotFoundException | SQLException ex) {
        Logger.getLogger(ChangePasswordController.class.getName()).log(Level.SEVERE, msg: null, thrown: ex);
    }
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //processRequest(request, response);
    // Retrieve form data
    String email = (String) request.getSession().getAttribute(string: "email");

    String Newpassword = request.getParameter(string: "Newpassword");
    String Confirmnewpassword = request.getParameter(string: "Confirmnewpassword");

    try {
        init();
        String q1 = "UPDATE users SET password = '" + Newpassword + "' WHERE email = '" + email + "'";
        int x = st.executeUpdate(string: q1);
    } catch (SQLException ex) {
        Logger.getLogger(ChangePasswordController.class.getName()).log(Level.SEVERE, msg: null, thrown: ex);
    }
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //processRequest(request, response);
    // Retrieve form data
    String email = (String) request.getSession().getAttribute(string: "email");

    String Newpassword = request.getParameter(string: "Newpassword");
    String Confirmnewpassword = request.getParameter(string: "Confirmnewpassword");

    try {
        init();
        String q1 = "UPDATE users SET password = '" + Newpassword + "' WHERE email = '" + email + "'";
        int x = st.executeUpdate(string: q1);
    } catch (SQLException ex) {
        Logger.getLogger(ChangePasswordController.class.getName()).log(Level.SEVERE, msg: null, thrown: ex);
    }
}

```

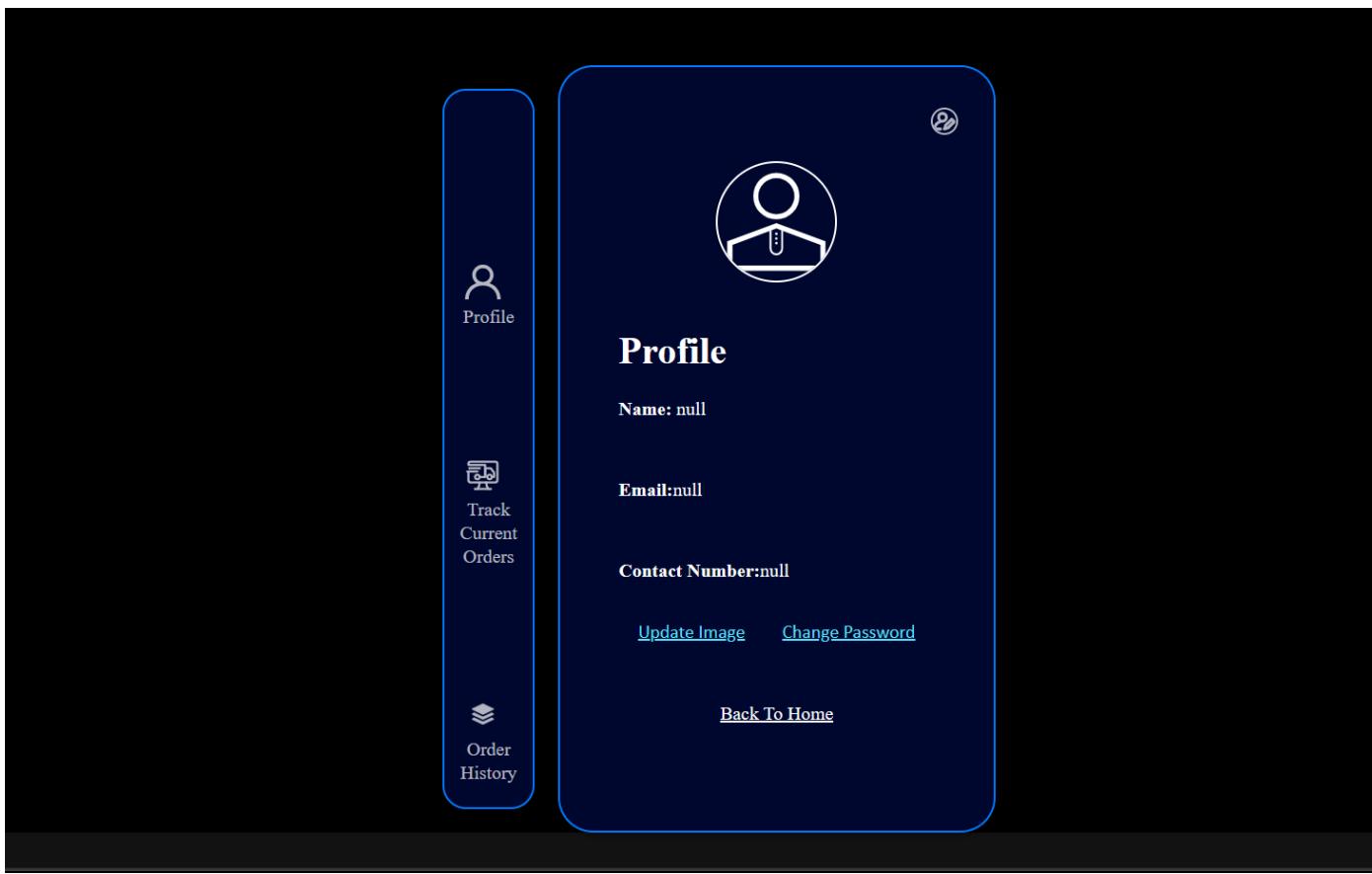
1.8 Profile page

(Includes these pages: Edit details, Change password)

Done by : S.N.M. Gedara

Student ID : 29165

1.8.1 Web page



1.8.2 Documentation : Profile page implementation

Profile JSP (Profile.jsp):

- This JSP file displays the user's profile information, including their name, email, and contact number.
- It includes a sidebar with links to different sections of the user's profile, such as tracking orders and order history.
- Users can update their profile picture by clicking on the "Update Image" label and selecting a file.
- There's also a link to change the user's password.
- JavaScript is used to handle the file input and display the selected profile picture.

Profile Servlet (ProfileServlet.java):

- This servlet is responsible for fetching user data from the database and forwarding it to the profile JSP.
- In the doGet method, it retrieves the user's email from the request parameters and uses it to fetch the corresponding user object from the database.
- It sets the user object as an attribute in the request and forwards the request to the profile JSP.
- The getUserByEmail method in the userDao class retrieves user data from the database based on the provided email.

User Model (SignUpUser.java):

- This class represents a user object with properties such as email, username, password, and contact number.
- It provides getter methods to retrieve these properties.

ProfilePictureServlet.java:

- Receives the uploaded file using request.getPart("profile-picture").Extracts the file name using filePart.getSubmittedFileName().Defines the upload directory and constructs the file path.
- Reads the file content as an input stream and writes it to the specified path.Stores the file name in the user's session.Redirects the user back to the profile page.

1.8.3 Codes

Profile.jsp

```
<%@page import="Model.SignUpUser" %>
<%@page import="Model.userDao" %>
<%@page import="java.util.*" %>
<%@page import="java.sql.*" %>
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Profile Page</title>
    <link href="CSS/Common.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/Navbar.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/Footer.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/Profile.css" rel="stylesheet" type="text/css"/>
  </head>
  <body>
    <%@ include file="Navbar.jsp" %>
    <div class="body-wrapper">
      <div class="container">
        <div class="sidebar" style="border: 2px solid #007bff;><br>
          <div class="sidebar-item">
            <a href="Profile.jsp">
              
              <span>Profile</span>
            </a>
          </div>
          <div class="sidebar-item">
            <a href="TrackOrder.jsp">
              
              <span>Track Current Orders</span>
            </a>
          </div>
        </div>
      </div>
    </div>
```

```


!\[\]\(Images/Profile/TrackOrder.svg\)
Track Current Orders



!\[\]\(Images/Profile/OrderHistory.svg\)
Order History






!\[Edit Details\]\(Images/Profile/EditProfile.svg "Edit Profile"\)


<br>

```

```


# Profile


<%
SignUpUser user = (SignUpUser) request.getAttribute("user");
if (user != null) {
%>
<p>Name: <%= user.getUsername() %></p>
<br>
<p>Email: <%= user.getEmail() %></p>
<br>
<p>Contact Number:<%= user.getContactNumber() %></p>
<!-- Add more fields as needed -->
<% } else { %>
<p><span style="font-weight: bold;">Name:</span> <%= session.getAttribute("username") %></p>
<br>
<p><span style="font-weight: bold;">Email:</span> <%= session.getAttribute("email") %></p>
<br>
<p><span style="font-weight: bold;">Contact Number:</span> <%= session.getAttribute("contactNumber") %></p>
<% } %>
<form action="ProfilePictureServlet" method="post" enctype="multipart/form-data">


|                                                                                                                                                                                                                                                                                         |                                                                          |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|
| <div class="button-row change-email"> &lt;label for="input-file" style="color: #59E3FF; text-decoration: underline; cursor: pointer;"&gt;Update Image&lt;/label&gt; &lt;input type="file" accept="image/jpeg, image/png, image/jpg" id="input-file" style="display: none;" /&gt; </div> | <a href="ChangePassword.jsp" style="color: #59E3FF;">Change Password</a> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------|


</form>
<br>

```

```

1 <p style="text-align: center;"><a href="index.jsp" style="color: white;">Back To Home</a></p>
2 <script>
3   window.onload = function () {
4     let profilePic = document.getElementById("profile-pic");
5     let inputFile = document.getElementById("input-file");
6     let chooseFile = document.getElementById("choose-file");
7
8     inputFile.onchange = function () {
9       if (validateFile(inputFile)) {
10         profilePic.src = URL.createObjectURL(inputFile.files[0]);
11       } else {
12         inputFile.value = ""; // Clear the file input field if validation fails
13       }
14     };
15
16     document.querySelector('label[for="input-file"]').addEventListener('click', function () {
17       chooseFile.click();
18     });
19   };
20
21   function validateFile(input) {
22     var file = input.files[0];
23     var allowedTypes = ["image/jpeg", "image/png", "image/jpg"]; // Allowed image types
24
25     if(file && !allowedTypes.includes(file.type)) {
26       // File is not an image, show an error message
27       alert("Please select a valid image file (JPEG, PNG, JPG).");
28       return false;
29     }
30     return true;
31   }
32 </script>
33 </div>
34 </div>
35 </div>
36 <%@ include file="Footer.html" %>
37 <script src="JS/Common.js"></script>
38 </body>
39 </html>

```

Profile.css

```

1 body {
2   background-color: #000000; /* Black background */
3   height: 100vh; /* Set the height of the viewport */
4 }
5
6 .body-wrapper {
7   display: flex;
8   flex-direction: column;
9   align-items: center;
10 }
11
12 .sidebar-item img {
13   width: 30px; /* Adjust the width as needed */
14   height: 30px; /* Adjust the height as needed */
15   margin-right: 10px; /* Optional: Add some spacing between the icon and text */
16 }
17
18 /* styles for sidebar item text */
19 .sidebar-item span {
20   vertical-align: middle; /* Align the text vertically with the icon */
21 }
22
23 /* Styling for the form */
24 .container {
25   display: flex;
26 }

```

```
.sidebar {
  width: 55px;
  max-height: 600px; /* Adjust as needed */
  height: 570px;
  background-color: rgba(0, 11, 69, 0.67);
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-between;
  padding: 20px 10px;
  border-radius: 20px; /* Rounded corners for the entire sidebar */
  margin-top: 50px; /* Adjust the top margin to move the sidebar up */
}

.sidebar-item {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 10px;
  text-align: center;
}

.sidebar-icon {
  font-size: 24px;
}
.sidebar-item span {
  color: #fffff; /* Set text color to white */
  font-size: medium;
}
.sidebar-item a {
  text-decoration: none;
}
.sidebar-item .sidebar-icon,
.sidebar-item span {
  color: rgba(255, 255, 255, 0.7); /* Set initial text color with reduced opacity */
  opacity: 3.0; /* Set initial opacity */
  transition: color 0.3s ease, opacity 0.3s ease; /* Smooth transition */
}
.sidebar-item:hover .sidebar-icon,
.sidebar-item:hover span {
  color: #fff; /* Increase the text color brightness on hover */
  opacity: 1; /* Increase opacity on hover */
}
.sidebar-item img {
  opacity: 0.7; /* Set initial opacity */
  transition: opacity 0.3s ease; /* Smooth transition */
}

.sidebar-item:hover img {
  opacity: 1; /* Increase opacity on hover */
}

.form-container {
  margin-left: 20px; /* Create space between sidebar and form */
  max-width: 800px;
  margin-top: 20px;
  height: 500px;
  padding: 60px 50px;
  border: 2px solid #000000; /* Increased border width */
  border-radius: 30px;
  background: rgba(0, 11, 69, 0.67);
  box-shadow: 0px 4px 4px 0px rgba(0, 0, 0, 0.25);
}

.profile-form table {
  width: 100%;
  border-radius: 50px;
}

.profile-form th, .profile-form td {
  padding: 15px;
  text-align: center;
  font-family: Calibri, Calibri; /* Change font family */
  color: #fffff; /* Change font color */
}
```

```
.profile-form .line {
    border-top: 1px solid #ccc;
}

.user-image-container {
    display: flex;
    justify-content: center; /* Center horizontally */
    margin-top: 20px; /* Adjust as needed */
}

.user-image-container img {
    width: 100px; /* Adjust the width of the image */
    height: 100px; /* Adjust the height of the image */
    border-radius: 50%; /* Make it a circle */
    border: 2px solid #fff; /* Add a border around the circle */
}

input[type="text"] {
    display: block;
    width: 100%;
    border: 0;
    border-bottom: 2px solid #fff;
    background: transparent;
    outline: none;
    padding: 5px 2px;
    position: relative;
    z-index: 10;
}

input[type="password"] {
    display: block;
    width: 100%;
    border: 0;
    border-bottom: 2px solid #fff;
    background: transparent;
    outline: none;
    padding: 5px 2px;
    position: relative;
    z-index: 10;
}

input[type="email"] {
    display: block;
    width: 100%;
    border: 0;
    border-bottom: 2px solid #fff;
    background: transparent;
    outline: none;
    padding: 5px 2px;
    position: relative;
    z-index: 10;
}

input[type="tel"] {
    display: block;
    width: 100%;
    border: 0;
    border-bottom: 2px solid #fff;
    background: transparent;
    outline: none;
    padding: 5px 2px;
    position: relative;
    z-index: 10;
}

.change-password a {
    color: #59E3FF; /* Change to the desired color */
    text-decoration: underline;
}
```

```

choose-file-button, .save-button {
    background-color: #22A1C9; /* Change to the desired color */
    color: white; /* Text color */
    padding: 8px 12px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
}

choose-file-button:hover, .save-button:hover {
    background-color: #0056b3; /* Change to the desired hover color */
}

#choose-file {
    display: none;
}

.edit-icon {
    width: 20px;
    height: 20px;
    margin-left: 5px; /* Adjust the spacing between the icon and the link */
    fill: #59E3F; /* Change the color of the icon */
}

.form-container {
    position: relative; /* Set position relative for proper positioning of children */
}

```

ProfileServlet.java

```

package Controller;

import Model.SignUpUser;
import Model.userDao;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * @author DELL
 */
@WebServlet(name = "ProfileServlet", urlPatterns = {"~/ProfileServlet"})
public class ProfileServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        //processRequest(request, response);
        // Fetch data from the database
        String email = request.getParameter("email"); // Assuming email is used as a unique identifier
        userDao userDAO = new userDao();
        SignUpUser user = userDAO.getUserByEmail(email);

        // Set the user object as an attribute in the request
        request.setAttribute("user", user);

        // Forward the request to the profile JSP
        request.getRequestDispatcher("Profile.jsp").forward(request, response);
    }
}

```

SignUpUser.java

```
package Model;

public class SignUpUser {
    private String email;
    private String username;
    private String password;
    private String contactNumber;

    public SignUpUser(String username, String password, String contactNumber, String email) {
        this.email = email;
        this.username = username;
        this.password = password;
        this.contactNumber = contactNumber;
    }

    public Object getEmail() {
        return email;
    }

    public String getUsername() {
        return username;
    }

    public String getPassword() {
        return password;
    }

    public String getContactNumber() {
        return contactNumber;
    }
}
```

UserDAO.java

```
public SignUpUser getUserByEmail(String email) {
    connectToDB();
    String query = "SELECT * FROM users WHERE email = '" + email + "'";
    ResultSet rs = null;
    SignUpUser user = null;

    try {
        rs = st.executeQuery(query);
        if (rs.next()) {
            user = new SignUpUser(rs.getString("username"), rs.getString("email"), rs.getString("password"), rs.getString("contactNumber"));
        }
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }
    return user;
}
```

ProfilePictureServlet.java

```
package Controller;

import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import jakarta.servlet.http.Part;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.OutputStream;

public class ProfilePictureServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        // Get the uploaded file from the request
        Part filePart = request.getPart("input-file");
        String fileName = filePart.getSubmittedFileName();

        // Define the directory to save the uploaded file
        String uploadDir = "/path/to/upload/directory";
        String filePath = uploadDir + File.separator + fileName;

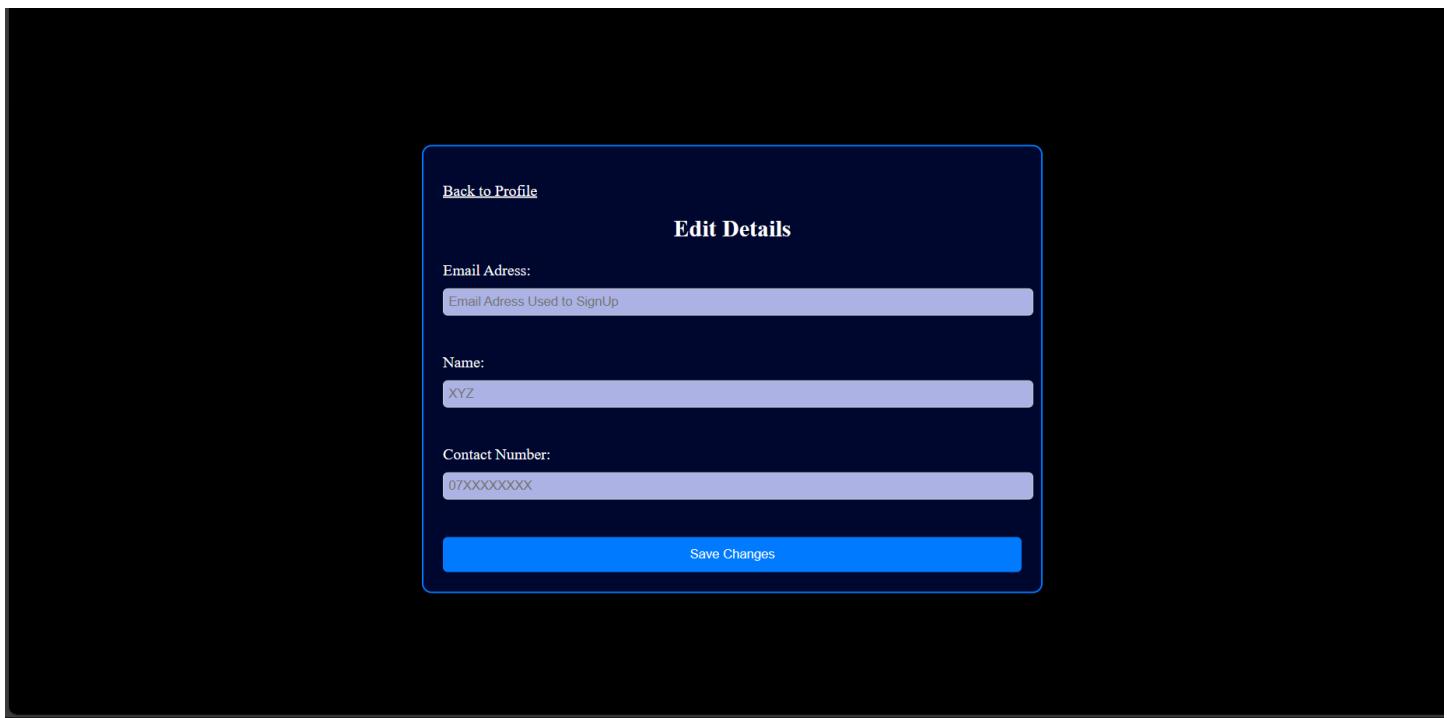
        // Save the uploaded file to the server
        try (InputStream input = filePart.getInputStream();
            OutputStream output = new FileOutputStream(name: filePath)) {
            byte[] buffer = new byte[1024];
            int bytesRead;
            while ((bytesRead = input.read(buffer)) != -1) {
                output.write(buffer, 0, bytesRead);
            }
        } catch (IOException e) {
            // Handle file upload error
            e.printStackTrace();
        }

        // Store the file name in the user's session or associated data structure
        HttpSession session = request.getSession();
        session.setAttribute("profilePicture", fileName);

        // Redirect back to the profile page or any other desired page
        response.sendRedirect("Profile.jsp");
    }
}
```

Edit Details

Web page



Documentation : Edit Details part implementation

Form Submission Handling:

- When the user submits the form in EditDetails.jsp, the browser sends a POST request to the server with the form data.
- This servlet listens for POST requests, so it overrides the doPost method to handle these requests specifically.
- Inside the doPost method, it retrieves the form data using request.getParameter("parameterName"). In this case, it retrieves the email, username, and contact number that the user entered in the form.

Updating User Details:

- After retrieving the form data, the servlet calls the updateDetails method from the userDao class.
- This method takes the new email, username, and contact number as parameters and updates the corresponding fields in the database for the user with the specified email.
- The updateDetails method contains the SQL logic to execute the database update query. It first checks if the email exists in the database, and if it does, it updates the user's details accordingly.

Session Attribute Update:

- Once the database update is successful, the servlet sets the updated email, username, and contact number in the user's session attributes.
- This ensures that the updated information is stored in the session and can be accessed across multiple requests during the user's session. It's particularly useful for maintaining user state and providing personalized experiences.

Redirect to Profile Page:

- Finally, after updating the database and session attributes, the servlet redirects the user back to the profile page (Profile.jsp) using response.sendRedirect("Profile.jsp").
- This ensures that the user is redirected to the appropriate page after completing the form submission.

Codes

EditDetails.jsp

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Details</title>
    <link href="CSS/Common.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/Navbar.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/Footer.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/EditDetails.css" rel="stylesheet" type="text/css"/>
</head>
<body>
    <%@ include file="Navbar.html" %>
    <div class="body-wrapper">

        <form id="editForm" action="SaveDetailsServlet" method="post" style="border: 2px solid #007bff; padding: 20px;">
            <p><a href="Profile.jsp" style="color: white;">Back to Profile</a></p><h2 style="text-align: center; margin-top: 0;">Edit Details</h2>

            <label for="email">Email Adress:</label>
            <input type="email" id="email" name="email" placeholder="Email Adress Used to SignUp" value=""><br><br>

            <label for="username">Name:</label>
            <input type="text" id="username" name="username" placeholder="XYZ" value=""><br><br>

            <label for="ContactNumber">Contact Number:</label>
            <input type="tel" id="ContactNumber" name="contactNumber" placeholder="07XXXXXXXX" value=""><br><br>

            <button type="submit">Save Changes</button>
        </form>
    </div>
</body>
</html>
```

EditDetails.css

```
body {
    background-color: #000000; /* Black background */
    height: 100vh; /* Set the height of the viewport */
    color: white;
}

.body-wrapper{
    display: flex;
    flex-direction: column;
    align-items: center;
}

form {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: #333; /* Change to desired background color */
    padding: 20px;
    border-radius: 10px;
}

/* Align labels and inputs */
label {
    display: block;
    margin-bottom: 10px;
    color: white; /* Set label text color */
}

/* Center the form and set width */
form {
    position: absolute;
    top: 55%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: rgba(0, 11, 69, 0.67); /* Change to desired background color */
    padding: 20px;
    border-radius: 10px;
    width: 400px; /* Adjust the width as needed */
    max-width: 100%; /* Ensure the form does not exceed the viewport width */
}

.h2 {
    text-align: center;
    font-family: sans-serif; /* Change the font family to Arial or any other desired font */
    margin-top: 30px; /* Adjust the margin-top value to move the heading down */
    margin-bottom: 20px; /* Adjust the margin-bottom value for spacing between the heading and the form */
}

/* Adjust form width and center it */
form {
    width: 40%; /* Adjust the width of the form */
    margin: 0 auto; /* Center the form horizontally */
}

input[type="text"],
input[type="email"],
input[type="password"],
input[type="tel"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
}
```

```
button[type="submit"]{
    width: 100%;
    background-color: #007bff; /* Default background color */
    color: white;
    padding: 10px;
    color:white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease; /* Transition for smooth color change */
}

button[type="submit"]:hover{
    background-color: #0056b3; /* Darker background color on hover */
}

/* Styling for the form */
.container {
    display: flex;
}

.form-container{
    margin-left: 20px; /* Create space between sidebar and form */
    max-width: 800px;
    margin-top: 20px;
    height:500px;
    padding: 60px 50px;
    border: 2px solid #000000; /* Increased border width */
    border-radius: 70px;
    background: rgba(0, 11, 69, 0.67);
    box-shadow: 0px 4px 4px 0px rgba(0, 0, 0, 0.25);
}
```

```
.profile-form th, .profile-form td {
    padding: 15px;
    text-align: center;
    font-family: Calibri,Calibri; /* Change font family */
    color: #fffff; /* Change font color */
}

.profile-form .line {
    border-top: 1px solid #ccc;
}

.user-image-container {
    display: flex;
    justify-content: center; /* Center horizontally */
    margin-top: 20px; /* Adjust as needed */
}

.user-image-container img {
    width: 100px; /* Adjust the width of the image */
    height: 100px; /* Adjust the height of the image */
    border-radius: 50%; /* Make it a circle */
    border: 2px solid #fff; /* Add a border around the circle */
}

input[type="text"]{
    background-color: #acb3e4;
    border: 1px solid #ccc;
    padding: 5px;
}

input[type="password"]{
    background-color: #acb3e4;
    border: 1px solid #ccc;
    padding: 5px;
}
```

```

input[type="email"]{
    background-color: #acb3e4;
    border: 1px solid #ccc;
    padding: 5px;
}
input[type="tel"]{
    background-color: #acb3e4;
    border: 1px solid #ccc;
    padding: 5px;
}
.change-password a {
    color: #59E3FF; /* Change to the desired color */
    text-decoration: underline;
}
.choose-file-button, save-button {
    background-color: #22A1C9; /* Change to the desired color */
    color: white; /* Text color */
    padding: 8px 12px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
}

.choose-file-button:hover, save-button:hover {
    background-color: #0056b3; /* Change to the desired hover color */
}
#choose-file {
    display: none;
}
.edit-icon {
    width: 20px;
    height: 20px;
    margin-left: 5px; /* Adjust the spacing between the icon and the link */
    fill: #59E3FF; /* Change the color of the icon */
}
.form-container {
    position: relative; /* Set position relative for proper positioning of children */
}

```

SaveDetailsServlet.java

```

package Controller;

import Model.SignUpUser;
import Model.userDao;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;
import java.io.PrintWriter;

/*
 * 
 * @author DELL
 */
@WebServlet(name = "SaveDetailsServlet", urlPatterns = {"/SaveDetailsServlet"})
public class SaveDetailsServlet extends HttpServlet {

```

```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //processRequest(request, response);
    // Retrieve form data
    String email = request.getParameter("email");
    String newusername = request.getParameter("username");
    String newcontactNumber = request.getParameter("contactNumber");
    userDao.updateDetails(email, newusername, newcontactNumber);

    HttpSession session = request.getSession();
    session.setAttribute("email", email.trim());
    session.setAttribute("username", newusername.trim());
    session.setAttribute("contactNumber", newcontactNumber.trim());

    response.sendRedirect("Profile.jsp");
}

```

UserDAO.java

```

public static void updateDetails(String email, String newusername, String newcontactNumber) {
    connectToDB();
    // Check if email exists
    String checkQuery = "SELECT * FROM users WHERE email = ?";
    ResultSet rs = null;
    boolean emailExists = false;

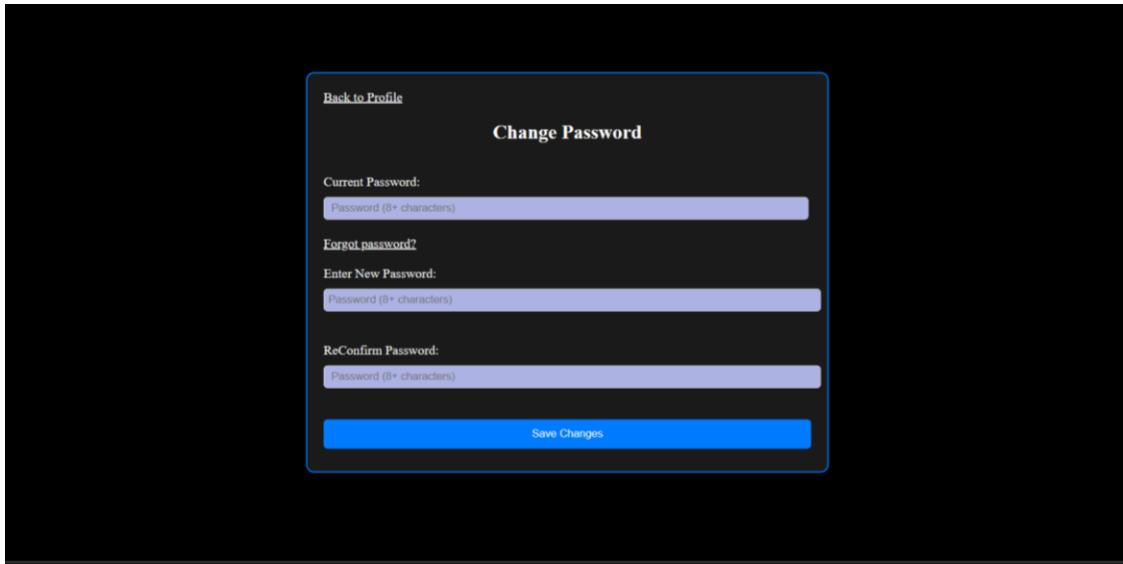
    try {
        PreparedStatement checkStatement = con.prepareStatement(checkQuery);
        checkStatement.setString(1, email);
        rs = checkStatement.executeQuery();
        emailExists = rs.next(); // Check if there's a result
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }
}

// Update details only if email exists
if (emailExists) {
    String updateQuery = "UPDATE users SET username = ?, contactNumber = ? WHERE email = ?";
    try {
        PreparedStatement updateStatement = con.prepareStatement(updateQuery);
        updateStatement.setString(1, newusername);
        updateStatement.setString(2, newcontactNumber);
        updateStatement.setString(3, email);
        updateStatement.executeUpdate();
        System.out.println("Details updated successfully.");
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
} else {
    System.out.println("Email address not found. Update failed.");
    // You can send an error message to the user here
}
}

```

Change password

Web page



Documentation : Change password part implementation

Form for Changing Password:

- The HTML form (ChangePassword.jsp) allows users to change their password.
- It includes fields for entering the current password (oldPassword), the new password (NewPassword), and confirming the new password (ReConfirmPassword).
- JavaScript validation (validation() function) ensures that all fields are filled out, the new password is at least 8 characters long, and the new password matches the confirmation.

Error Handling:

- The JSP page checks for error messages passed via the request parameter error.
- If there's an error (e.g., incorrect current password or mismatched new passwords), it displays an appropriate error message above the form.

Servlet Handling Form Submission:

- The servlet (ChangePasswordController) handles the form submission using the doPost method.
- It retrieves the old password, new password, and confirmation from the request parameters.
- Then it initializes a connection to the database in the init method using a Statement object.

Updating Password in the Database:

- The servlet constructs an SQL query to update the password in the database.
- It uses a PreparedStatement to prevent SQL injection and execute the update query.
- If the update is successful, it redirects the user to the profile page (Profile.jsp), indicating that the password change was successful.

Error Handling in Servlet:

- Error handling is limited in the servlet. If there's an SQL exception during the password update, it logs the exception but doesn't provide feedback to the user on the webpage.
- For better user experience, you could enhance error handling to inform the user if the password update fails due to an SQL error.

Codes

ChangePassword.jsp

```
<%@ page import="Model.SignUpUser" %>
<%@ page import="Model.userDao" %>

<%
String error = request.getParameter("error");
String errorMessage = "";
if(error != null && error.equals("1")) {
    errorMessage = "Incorrect current password. Please try again.";
} else if(error != null && error.equals("2")){
    errorMessage = "New password and confirm new password do not match.";
}
%>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Change Password</title>
    <link href="CSS/Common.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/Navbar.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/Footer.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/ChangePassword.css" rel="stylesheet" type="text/css"/>
```

```

<script>
function validation() {
    var NewPassword = document.getElementById("NewPassword").value;
    var ReConfirmPassword = document.getElementById("ReConfirmPassword").value;

    if( !NewPassword || !ReConfirmPassword) {
        alert("All fields must be filled out.");
        return false;
    }

    if(NewPassword.length<8){
        alert("Password must be at least 8 characters long.");
        return false;
    }

    if(NewPassword != ReConfirmPassword){
        alert("Passwords do not match.");
        return false;
    }

    return true;
}
</script>
</head>
<body>

```

```

<%@ include file="Navbar.html" %>
] <div class="body-wrapper">
] <div style="color: red;"><%= errorMessage %></div>
] <form action="ChangePasswordController" onsubmit="return validation()" method="post" style="border: 2px solid #007bff; background-color: #1a1a1a; width: 600px; height: 450px;">
<!-- Rest of your form -->
<a href="Profile.jsp" style="color: white;">Back to Profile</a>
<h2 style="text-align: center;">Change Password</h2>

<br><label for="oldPassword">Current Password:</label>
<input type="password" id="oldPassword" name="oldPassword" placeholder="Password (8+ characters)" style="width: 585px;">
<a href="ForgotPassword.jsp" style="color: white;">Forgot password?</a>
<br><br>
<label for="newpassword">Enter New Password:</label>
<input type="password" id="NewPassword" name="NewPassword" placeholder="Password (8+ characters)"><br><br>

<label for="ReConfirmPassword">ReConfirm Password:</label>
<input type="password" id="ReConfirmPassword" name="ReConfirmPassword" placeholder="Password (8+ characters)"><br><br>
<button type="submit">Save Changes</button>
</form>
</div>
</body>
</html>

```

ChangePassword.css

```
body {
    background-color: #000000; /* Black background */
    height: 100vh; /* Set the height of the viewport */
    color: white;
}

.body-wrapper {
    display: flex;
    flex-direction: column;
    align-items: center;
}

form {
    position: absolute;
    top: 55%;
    left: 50%;
    transform: translate(-50%, -50%);
    background-color: rgba(0, 11, 69, 0.67); /* Change to desired background color */
    padding: 20px;
    border-radius: 10px;
    width: 400px; /* Adjust the width as needed */
    max-width: 100%; /* Ensure the form does not exceed the viewport width */
}

/* Align labels and inputs */
label {
    display: block;
    margin-bottom: 10px;
    color: white; /* Set label text color */
}

input[type="text"],
input[type="email"],
input[type="password"],
input[type="tel"] {
    width: 100%;
    padding: 10px;
    margin-bottom: 20px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

button[type="submit"] {
    width: 100%;
    background-color: #007bff; /* Default background color */
    color: white;
    padding: 10px;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s ease; /* Transition for smooth color change */
}

button[type="submit"]:hover {
    background-color: #0056b3; /* Darker background color on hover */
}

.h2 {
    text-align: center;
    font-family: Arial, sans-serif; /* Change the font family to Arial or any other desired font */
    margin-top: 30px; /* Adjust the margin-top value to move the heading down */
    margin-bottom: 50px; /* Adjust the margin-bottom value for spacing between the heading and the form */
}

/* Styling for the form */
.container {
    display: flex;
}
```

```

.form-container {
    margin-left: 20px;
    max-width: 800px;
    margin-top: 20px;
    height: 500px;
    padding: 60px 50px;
    border: 30px solid #000000; /* Increased border width */
    border-radius: 70px;
    background: #5D6D7E;
    box-shadow: 0px 4px 4px 0px rgba(0, 0, 0, 0.25);
}

.form-line {
    border-top: 1px solid #ccc;
}

input[type='password'] {
    background-color: #acb3e4;
    border: 1px solid #ccc;
    padding: 5px;
}

.change-password a {
    color: #59E3FF; /* Change to the desired color */
    text-decoration: underline;
}

.form-container {
    position: relative; /* Set position relative for proper positioning of children */
}

```

ChangePasswordController.java

```

package Controller;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 * 
 * @author DELL
 */
@WebServlet(name = "ChangePasswordController", urlPatterns = {"/ChangePasswordController"})
public class ChangePasswordController extends HttpServlet {

    public Statement st; // Declare Statement as a member variable

    @Override
    public void init() throws ServletException {
        super.init();
        try {
            // Initialize the Statement in the init method
            String driver = "com.mysql.cj.jdbc.Driver";
            String url = "jdbc:mysql://localhost:3306/emart";
            Class.forName(className: driver);
            Connection con = DriverManager.getConnection(url, user: "root", password: "");
            st = con.createStatement();
        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(ChangePasswordController.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
        }
    }
}

```

```

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    //processRequest(request, response);
    response.setContentType(string: "text/html");
    PrintWriter out = response.getWriter();

    String oldPassword=request.getParameter(string: "oldPassword");
    String newPassword = request.getParameter(string: "NewPassword");
    String confirmNewPassword = request.getParameter(string: "ReConfirmPassword");

    try {
        init();
        String q1 = "UPDATE users SET password = '" + newPassword + "' WHERE password = '" + oldPassword + "'";
        int x = st.executeUpdate(string: q1);

    } catch (SQLException ex) {
        Logger.getLogger(ChangePasswordController.class.getName()).log(level:Level.SEVERE, msg: null, thrown: ex);
    }

    // Password updated successfully, redirect to a success page
    response.sendRedirect(string: "Profile.jsp");
}

```

1.9 About Us page

Done by : S.N.M. Gedara

Student ID : 29165

1.9.1 Web page

Upgrade your life with E-Mart: Where every click brings you closer to the latest in digital innovation. Shop smarter, live better.

Why Us?

Our Mission

"Our mission at E-mart is to democratize education by providing accessible, high-quality e-courses that empower individuals to enhance their skills, pursue their passions, and achieve their personal and professional goals."

Our Vision

"Our vision at E-mart is to revolutionize education by providing a dynamic platform where anyone can explore, learn, and excel, regardless of their background or location. We aspire to become the premier destination for e-courses, inspiring lifelong learning and innovation."

Our Values

"E-mart offers high-quality e-courses for all, anytime, anywhere. With expert partnerships, we provide engaging content. Our platform encourages interactive learning through forums and live sessions, fostering peer collaboration. Tailoring courses to individual needs and goals, we empower lifelong learners for personal and professional success."

1.9.2 Documentation : About Us page implementation

AboutUs.jsp :

- This page is structured using jsp, with sections for the header, main content, and footer.
- It includes CSS and JavaScript files for styling and functionality.

Header and Navigation:

- The header includes a navigation bar (Navbar.jsp) that allows users to navigate between different sections of the website.
- The navigation bar likely includes links to other pages such as the home page, profile page, and more.

Main Content:

- The main content of the "About Us" page focuses on three key aspects: mission, vision, and values.
- Each aspect is presented in a separate section (div.container-right) for better organization and readability.

Mission, Vision, and Values:

- Each section (div.column) contains a card (div.card) with a title (h2) and a description (p.title).
- The description outlines the organization's mission, vision, and values, providing users with insights into its goals, aspirations, and principles.
- The content emphasizes accessibility, quality, innovation, and inclusivity, reflecting the organization's commitment to its stakeholders.

Styling and Responsiveness:

- The CSS (AboutUs.css) defines styles for different elements, ensuring a visually appealing layout and consistent design across devices.
- Media queries are used to adjust the layout and styling based on the screen size, enhancing responsiveness and usability on various devices.

1.9.3 Codes

AboutUs.jsp

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>About Us</title>
    <link href="CSS/Common.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/Navbar.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/Footer.css" rel="stylesheet" type="text/css"/>
    <link href="CSS/AboutUs.css" rel="stylesheet" type="text/css"/>
  </head>

  <body>
    <%@ include file="Navbar.jsp" %>
    <!-- <div id="wave"></div> -->

    <h1 class="center">Why Us?</h1>
    <div style="display:flex;">
      <div class="container-left">
        <div class="circle"></div>
        <div class="circle"></div>
        <div class="text">Upgrade your life with E-Mart: Where every click brings you closer to the latest in digital innovation. Shop smarter, live better.</div>
      </div>
    </div>

    <div class="container-right">
      <div class="row">
        <!-- our mission -->
        <div class="column">
          <div class="card">
            <div class="container">
              <h2>Our Mission </h2>
              <p class="title">"Our mission at E-mart is to democratize education by providing accessible, high-quality e-courses that empower individuals to enhance their skills, pursue their passions, and achieve their personal and professional goals."</p>
            </div>
          </div>
        </div>
      </div>
    </div>

    <div class="column">
      <div class="card">
        <div class="container">
          <h2>Our Vision</h2>
          <p class="title">"Our vision at E-mart is to revolutionize education by providing a dynamic platform where anyone can explore, learn, and excel, regardless of their background or location. We aspire to become the premier destination for e-courses, inspiring lifelong learning and innovation."</p>
        </div>
      </div>
    </div>

    <div class="column">
      <div class="card">
        <div class="container">
          <h2>Our Values </h2>
          <p class="title">"E-mart offers high-quality e-courses for all, anytime, anywhere. With expert partnerships, we provide engaging content. Our platform encourages interactive learning through forums and live sessions, fostering peer collaboration. Tailoring courses to individual needs and goals, we empower lifelong learners for personal and professional success."</p>
        </div>
      </div>
    </div>

    </div>
    </div>
    <!-- <div id="wave"></div> -->
    <%@ include file="Footer.html" %>
    <script src="JS/Common.js"></script>
  </body>
</html>
```

AboutUs.css

```
.center {  
    text-align: center;  
    color: #fff;  
}  
  
.html {  
    box-sizing: border-box;  
}  
  
.body{  
    background: #000;  
    margin: 20px;  
}  
  
.container-left {  
    width: 500px;  
    height: 500px;  
    position: relative;  
    right: 10px;  
    margin-bottom: 40px;  
}  
  
.circle {  
    width: 400px;  
    height: 400px;  
    border-radius: 50%;  
    background-color: #1C1D35;  
    position: absolute;  
    top: 50%;  
    left: 50%;  
    transform: translate(-50%, -50%);  
    margin: 0 auto;  
}  
  
.text {  
    color: #fff;  
    font-size: 21px;  
    text-align: center;  
    line-height: 1.8;  
    position: absolute;  
    top: 50%;  
    left: 50%;  
    transform: translate(-50%, -50%);  
    z-index: 1;  
}  
  
.line {  
    width: 40%;  
    border-color:#fff;  
    margin: 0 auto;  
    position: absolute;  
    top: 70%;  
    left: 50%;  
    transform: translateX(-50%);  
    z-index: 2;  
    border-bottom: none;  
    border-left: none;  
    border-right: none;  
}  
  
.line::after {  
    content: "E-MART";  
    display: block;  
    font-size: 24px;  
    color:#fff;  
    text-align: center;  
    margin-top: 10px;  
}
```

```
.container-right {
    width: 55%;
}

.row {
    display: flex;
    flex-wrap: wrap;
    justify-content: center;
    margin: 0px -0px;
}

.column {
    color: black;
    width: calc(100% - 5px);
    margin: 1px 1px;
    margin-bottom: 5px;
    padding: 20px 30px;
    box-sizing: border-box;
}

.card {
    border-radius: 10px;
    box-shadow: 2px 2px 4px 4px rgba(255, 255, 255, 0.3);
    background-color: white;
    border: 2px solid transparent;
    border-bottom-color: #151663;
}

.container {
    padding: 5px 10px;
}

.container::after, .row::after {
    content: "";
    clear: both;
    display: table;
}

.title {
    color: #131c5e;
    /*font-weight: bold;*/
}

@media screen and (max-width: 650px) {
    .container-left,
    .container-right {
        width: 100%;
    }
}

body {
    margin: 0;
}

.nav-link.aboutUs-link {
    opacity: 1;
}
```

1.10 Developers page

Done by : S.N.M. Gedara
Student ID : 29165

1.10.1 Web page

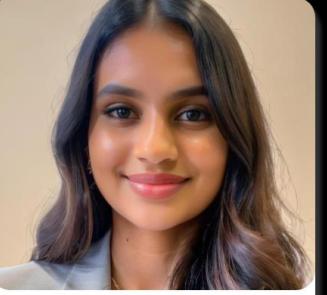
Developers



E. L Thambawita
BSc (Hons) in Computer Science



I. S .N .M Gedara
BSc (Hons) in Computer Science



G.O Wickramaratne
BSc (Hons) in Computer Science



EW.A.C Fernando
BSc (Hons) in Data Science



W.M.C.S Wijesinghe
BSc (Hons) in Computer Science



W.S.T.P.R Fernando
BSc (Hons) in Computer Science

1.10.2 Documentation : Developers page implementation

Developers.jsp :

- This page is structured using HTML, with sections for the header, main content, and footer.
- It includes CSS and JavaScript files for styling and functionality.

Header and Navigation:

- Similar to other pages, the header includes a navigation bar (Navbar.html) for easy navigation.

Main Content:

- The main content of the "Developers" page features a list of developers organized into rows and columns.
- Each developer is represented by a card (div.card) containing their name, image, and qualifications.

Developer Cards:

- Each developer card is structured with an image (img) and a container (div.container) for the developer's name and qualifications.
- The developer's name and qualifications are displayed as headings (h2 and p.title).

Developer Information:

- The image in each card serves as a hyperlink to the developer's LinkedIn profile, allowing users to learn more about them and connect professionally.

Styling and Responsiveness:

- The CSS (Developers.css) defines styles for different elements, ensuring a visually appealing layout and consistent design across devices.
- Media queries are used to adjust the layout and styling based on the screen size, making the page responsive and accessible on various devices.

Footer:

- Similar to other pages, the footer (Footer.html) includes links to additional resources, contact information, and social media profiles.

1.10.3 Codes

Developers.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>

    <head>
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Developers</title>
        <link href="CSS/Common.css" rel="stylesheet" type="text/css"/>
        <link href="CSS/Navbar.css" rel="stylesheet" type="text/css"/>
        <link href="CSS/Footer.css" rel="stylesheet" type="text/css"/>
        <link href="CSS/Developers.css" rel="stylesheet" type="text/css"/>
    </head>

    <body>
        <%@ include file="Navbar.html" %>
        <h1 class="center">Developers</h1>
        <br>

        <div class="row">
            <div class="column">
                <div class="card">
                    <a href="https://www.linkedin.com/in/esanki-thambawita-350947256">
                        </a>
                    <div class="container">
                        <h2>E. L Thambawita</h2>
                        <p class="title">BSc (Hons) in Computer Science</p>
                    </div>
                </div>
            </div>
        </div>

        <div class="column">
            <div class="card">
                <a href="https://www.linkedin.com/in/imalsha-sandali-54b01326a">
                    </a>
                <div class="container">
                    <h2>I. S .N .M Gedara</h2>
                    <p class="title">BSc (Hons) in Computer Science</p>
                </div>
            </div>
        </div>

        <div class="column">
            <div class="card">
                <a href="www.linkedin.com/in/oneli-wickramaratne">
                    </a>
                <div class="container">
                    <h2>G.O Wickramaratne</h2>
                    <p class="title">BSc (Hons) in Computer Science</p>
                </div>
            </div>
        </div>
        <!--Second Row-->
        <div class="row">
            <div class="column">
                <div class="card">
                    <a href="https://www.linkedin.com/in/adheeshafernando-5b8975230">
                        </a>
                    <div class="container">
                        <h2>EW.A.C Fernando</h2>
                        <p class="title">BSc (Hons) in Data Science</p>
                    </div>
                </div>
            </div>
        </div>
    </body>

```

```

<div class="column">
  <div class="card">
    <a href="https://www.linkedin.com/in/chanuka-wijesinghe-83968825a?trk=blended-typeahead">
      </a>
    <div class="container">
      <h2>W.M.C.S Wijesinghe</h2>
      <p class="title">BSc (Hons) in Computer Science</p>
    </div>
  </div>
</div>

<div class="column">
  <div class="card">
    <a href="https://www.linkedin.com/in/esanki-thambawita-350947256">
      </a>
    <div class="container">
      <h2>W.S.T.P.R Fernando</h2>
      <p class="title">BSc (Hons) in Computer Science</p>
    </div>
  </div>
</div>

<%@ include file="Footer.html" %>
<script src="JS/Common.js"></script>
</body>

</html>

```

Developers.css

```

footer{
  margin-top: 100px;
}

html {
  box-sizing: border-box;
}

*, *:before, *:after{
  box-sizing: inherit;
}

body {
  background-color: black; /* Set the background color of the body to black */
  color: white; /* Set text color to white for better visibility */
}

.row {
  display: flex;
  flex-wrap: wrap;
  justify-content: center; /* Center the columns horizontally */
  margin: 0 -15px; /* Adjust the negative margin to create equal spacing on both sides */
}

.column {
  width: calc(30% - 60px); /* Adjust the width to include padding */
  margin: 0 15px; /* Adjust the margin to create space between columns */
  margin-bottom: 16px;
  padding: 10px 30px;
  box-sizing: border-box;
}

.center {
  text-align: center;
}

```

```
□ @media screen and (max-width: 650px) {  
    □ .column {  
        width: 100%;  
        display: block;  
    }  
}  
  
□ .card img {  
    border-radius: 20px;  
    width: 100%;  
    height: 300px; /* Adjust the height as needed */  
    object-fit: cover;  
}  
  
□ .card {  
    border-radius: 30px;  
    box-shadow: 2px 4px 4px 4px rgba(255, 255, 255, 0.2);  
    background-color: #ffff;  
}  
  
□ .container {  
    padding: 0 16px;  
}  
  
□ .container::after, .row::after {  
    content: "";  
    clear: both;  
    display: table;  
}  
  
□ .title {  
    color: grey;  
}  
□ h2 {  
    color: black;  
}
```

1.11 Track Order page

Done by : S.N.M. Gedara
Student ID : 29165

1.11.1 Web page

The image displays three separate instances of a web-based order tracking interface, each showing a different order. Each instance consists of four main sections: Order Number, Item purchased, Quantity, and Actions. Below these are three status icons representing the progression of the order: Order Processing, Out For Delivery, and Delivered.

Order 1:
Order Number: 1
Item purchased: Smartphone Z
Quantity: 1
Actions: Return (white button), Cancel Order (dark blue button)
Order Status: Order Processing (blue bar), Out For Delivery (grey bar), Delivered (grey bar)

Order 2:
Order Number: 2
Item purchased: Smartphone B
Quantity: 2
Actions: Return (white button), Cancel Order (dark blue button)
Order Status: Order Processing (blue bar), Out For Delivery (blue bar), Delivered (grey bar)

Order 3:
Order Number: 3
Item purchased: Smartphone C
Quantity: 1
Actions: Return (white button), Cancel Order (dark blue button)
Order Status: Order Processing (grey bar), Out For Delivery (blue bar), Delivered (blue bar)

1.11.2 Documentation : Track Order page implementation

- The page imports necessary classes like TrackOrderServlet, List, ShoppingCartObj, DAO, and AdminOrderObj.
- The header includes a title "Track your order" along with a return button.
- It includes a navigation bar (Navbar.html) for easy navigation within the website.
- The main content is wrapped within a <div class="body-wrapper">.
- Orders are retrieved using DAO.getAdminOrders() and displayed using a loop.
- Each order is displayed within an order-tracking-card containing order details and tracking information.
- For each order, details like order number, purchased items, and quantity are displayed.
- Action buttons are provided for returning or canceling the order.
- The order status is visually represented using milestone icons and a progress bar.
- Milestone icons indicate different order statuses such as processing, out for delivery, and delivered.
- JavaScript code dynamically updates the milestone icons and progress bar based on the order status.
- JavaScript code is used to update the appearance of milestone icons and the width of the progress bar based on the order status.
- The updateProgressBar function adjusts the width of the progress bar.
- CSS files (Common.css, Navbar.css, Footer.css, TrackOrder.css) are linked to provide styling to various elements of the page.
- The footer (Footer.html) includes links to additional resources, contact information, and social media profiles.

Order Tracking Milestones:

- A visual representation of the order status progression is provided using milestone icons and a progress bar.
- Icons representing different order statuses (processing, out for delivery, delivered) are displayed along with their corresponding descriptions.
- The progress bar (div.progress-bar-progress) visually indicates the progress of the order.
- JavaScript Functionality:
- JavaScript is used to dynamically update the order status icons and progress bar based on the current status of each order.

- The updateProgressBar function adjusts the width of the progress bar to reflect the progress of the order.
- The status icons (tracking-mile-icon) change their appearance (from inactive to active) based on the order status.

Styling:

- The CSS (TrackOrder.css) provides styling for various elements, ensuring a visually appealing layout and consistent design.
- Different classes are used to style components such as order tracking cards, order details, and milestone icons

1.11.3 Codes

TrackOrder.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ page import="Controller.TrackOrderServlet" %>
<%@ page import="java.util.List" %>
<%@ page import="Model.ShoppingCartObj" %>
<%@page import="Model.DAO" %>
<%@page import="Model.AdminOrderObj" %>

<!DOCTYPE html>
<html>

    <head>
        <meta charset="utf-8" />
        <meta name="viewport" content="initial-scale=1, width=device-width" />

        <link rel="stylesheet" href="CSS/Common.css">
        <link rel="stylesheet" href="CSS/Navbar.css">
        <link rel="stylesheet" href="CSS/Footer.css">
        <link href="CSS\TrackOrder.css" rel="stylesheet" />
    </head>
    <body>
        <%@ include file="Navbar.html" %>
        <!-- Start Body Wrapper -->
        <div class="body-wrapper">

            <div class="headingTitle">
                <a href="Profile.jsp" style="display: inline-block;">
                    
                </a>
            </div>

            <h1 class="section-title">
                Track your order
            </h1>
        </div>
        <%
        List<AdminOrderObj> orders = DAO.getAdminOrders();
        for (AdminOrderObj order : orders) {
        %>
            <div class="order-tracking-card">
                <div class="order-details-row">
                    <div class="order-details-col">
                        <h2 class="order-detail-col-title">Order Number </h2>
                        <p class="order-detail-col-content"> <%= order.getOrderNumber() %> </p>
                    </div>
                    <div class="order-details-col">
                        <h2 class="order-detail-col-title">Item purchased </h2>
                        <p class="order-detail-col-content"> <%= order.getProductName() %> </p>
                    </div>
                </div>
                <div class="order-details-col">
                    <h2 class="order-detail-col-title">Quantity </h2>
                    <p class="order-detail-col-content"> <%= order.getQuantity() %> </p>
                </div>
            </div>
        <%>
    </div>
```

```

<div class="order-details-col">
  <h2 class="order-detail-col-title">Actions</h2>
  <a href="MyReturns.jsp" class="button-white">Return</a>
  <a href="" class="button-acrylic">Cancel Order</a>
</div>
</div>
<div class="order-tracking-row">
  <div class="tracking-header-row">
    <h2 class="order-detail-col-title title">Order Status</h2>
  </div>

  <div class="progress-visual">
    <div class="tracking-progress">
      <div class="progress-bar">
        <div class="progress-bar-progress" id="progressBar_<%= order.getOrderNumber() %>"></div>
      </div>
    </div>
    <div class="tracking-milestones">
      <input type="hidden" name="orderStatus" id="orderStatus_<%= order.getOrderNumber() %>" value="<%= order.getOrderStatus() %>">
      <div class="tracking-mile">
        <img class="tracking-mile-icon inactive" id="processingIcon_<%= order.getOrderNumber() %>" src="Images/TrackOrder/OrderProcessingIcon.png" alt="Processing">
        <p class="tracking-mile-des">Order Processing</p>
      </div>
      <div class="tracking-mile">
        <img class="tracking-mile-icon inactive" id="deliveryIcon_<%= order.getOrderNumber() %>" src="Images/TrackOrder/OutForDeliveryIcon.png" alt="Delivery">
        <p class="tracking-mile-des">Out For Delivery</p>
      </div>
      <div class="tracking-mile">
        <img class="tracking-mile-icon inactive" id="deliveredIcon_<%= order.getOrderNumber() %>" src="Images/TrackOrder/DeliveredIcon.png" alt="Delivered">
        <p class="tracking-mile-des">Delivered</p>
      </div>
    </div>
  </div>
</div>

<script>
document.addEventListener("DOMContentLoaded", function () {
  // Get the order status from the hidden input field
  var orderStatus = document.getElementById("orderStatus_<%= order.getOrderNumber() %>").value;

  // Update the class of the tracking-mile-icon elements based on the order status
  switch (orderStatus.toLowerCase()) {
    case "processing":
      document.getElementById("processingIcon_<%= order.getOrderNumber() %>").classList.remove("inactive");
      document.getElementById("processingIcon_<%= order.getOrderNumber() %>").classList.add("active");
      updateProgressBar(<%= order.getOrderNumber() %>, 20); // Set progress to 20%
      break;
    case "outfordelivery":
      document.getElementById("deliveryIcon_<%= order.getOrderNumber() %>").classList.remove("inactive");
      document.getElementById("deliveryIcon_<%= order.getOrderNumber() %>").classList.add("active");
      updateProgressBar(<%= order.getOrderNumber() %>, 51); // Set progress to 51%
      break;
    case "delivered":
      document.getElementById("deliveredIcon_<%= order.getOrderNumber() %>").classList.remove("inactive");
      document.getElementById("deliveredIcon_<%= order.getOrderNumber() %>").classList.add("active");
      updateProgressBar(<%= order.getOrderNumber() %>, 100); // Set progress to 100%
      break;
    default:
      break;
  }
});

// Function to update the width of the progress bar
function updateProgressBar(orderNumber, progress) {
  var progressBar = document.getElementById("progressBar_" + orderNumber);
  if (progressBar) {
    progressBar.style.width = progress + "%";
  }
}
</script>
</div>
</div>
</div>

```

```

body-wrapper {
  min-height: 55.2vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: flex-start;

  padding-top: 6vh;
  padding-bottom: 6vh;
}

.headingTitle{
  display: flex !important;
  flex-direction: row !important;
  align-items: center;
  gap: 20px;
}

.order-tracking-card {
  display: flex;
  flex-direction: column;
  gap: 1.5rem;

  flex-grow: 1;

  width: 100%;

  max-width: 1366px;
  height: 400px;

  padding: 1.5rem;

  background: radial-gradient(81.34% 81.34% at 84.42% 75.76%, #0A0A0A 0%, #181818 100%);
  border-radius: 24px;
  border: 2px solid rgba(255, 255, 255, 0.089);
  overflow: hidden;
  margin-bottom: 60px;
}

.order-details-row,
.order-tracking-row {
  display: flex;
  flex-direction: row;
  gap: 1.5rem;
}

.order-details-row {}

.order-tracking-row {
  flex-grow: 1;

  display: flex;
  flex-direction: column;
  gap: 1rem;

  padding: 1rem;

  background: rgba(0, 0, 0, 0.79);
  border-radius: 14px;
}

.order-details-col {
  flex-grow: 1;

  min-width: calc(25% - 5.3rem);
  max-width: calc(25% - 3rem);

  padding: 1rem;
  background: rgba(0, 0, 0, 0.79);
  border-radius: 14px;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 1rem;
}

```

```
.order-details-col p {  
  font-size: xx-large;  
}  
  
.order-detail-col-title {  
  font-size: larger;  
  font-weight: 400;  
  opacity: 0.7;  
  text-align: center;  
  margin: 0;  
}  
  
.order-detail-col-content {  
  display: flex;  
  flex-grow: 0;  
  margin: 0;  
  gap: 1rem;  
  font-size: 21px;  
  max-width: 200px;  
}  
  
.order-item-image {  
  height: 70px;  
}  
  
.tracking-header-row {  
  min-width: 100px;  
}  
  
.tracking-header-row .title {  
  text-align: left;  
}  
  
.progress-visual {  
  margin-top: 2rem;  
}  
  
.tracking-progress {  
  background-color: rgba(139, 69, 19, 0.226);  
  min-width: 100px;  
  flex-grow: 1;  
  
  display: flex;  
  flex-direction: column;  
  justify-content: center;  
}  
  
.progress-bar {  
  background-color: #202020;  
  min-width: 100px;  
  min-height: 40px;  
  
  border-radius: 8px;  
}  
  
.progress-bar-progress {  
  min-width: 0;  
  min-height: inherit;  
  background: linear-gradient(90deg, #3B7BF8 0%, #0066DD 100%);  
  border-radius: 8px;  
  width: 0%;  
  transition: all 2s;  
}  
  
.tracking-milestones {  
  margin: 0 0 15%;  
  margin-top: -60px;  
  bottom: 0;  
  display: flex;  
  justify-content: space-between;  
}
```

```
.tracking-mile {  
    min-width: 140px;  
    min-height: 80px;  
    z-index: 1;  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
}  
  
.tracking-mile-icon {  
    background-color: #181818;  
    border-radius: 15px;  
    transition: all 2s;  
}  
  
.tracking-mile-des {  
    opacity: 0.6;  
}  
  
.tracking-mile-icon.active {  
    border: 5px solid #1369de;  
}  
  
.tracking-mile-icon.inactive {  
    border: 5px solid #1f1f1f;  
}
```

1.12 Checkout page

Done by : G.O.Wickramaratne
Student ID : 28039

1.12.1 Web page

The screenshot shows the checkout process on the E-Mart Electronics website. At the top, there's a navigation bar with the logo 'E-MART ELECTRONICS', 'Home', 'Products', 'About Us', and 'Support'. On the right side of the header are icons for user profile and shopping cart.

The main content area is titled 'Checkout' with a sub-section 'Cart'. It displays a table of items:

Product	Price
Smartphone	699
Laptop	1099
Camera	499
Total:	\$3396

Below the cart, the 'Billing Details' section contains fields for 'Full Name' (John Doe) and 'Email' (john@example.com).

The next section, 'Address', includes fields for 'Address' (123 Street, City), 'City', 'State', and 'Zip'.

The 'Payment Method' section offers 'Cash on Delivery' as the selected option (indicated by a blue dot). A 'SUBMIT PAYMENT' button is at the bottom of this section.

At the very bottom of the page, there's a footer with the E-Mart logo, a brief description about the latest innovations in electronics, links to 'Company' (About E-Mart, Developers), 'My Mart' (My Profile, Ongoing Orders, Order History), 'Contact Us' (+94 77 696 9696, mail.emart@gmail.com), social media links (Facebook, Twitter, Instagram, LinkedIn, YouTube), and copyright information (Â© 2024 Electronics Mart, Inc) along with links to 'Privacy Policy' and 'Terms of Use'.

1.12.2 Documentation : Checkout page implementation

Header Section:

- Includes the document title "Checkout" and imports necessary Java classes.
- Links to external CSS files for styling.

Navigation Bar:

- Utilizes an included HTML file Navbar.html to display navigation links for easy website traversal.

Main Content Section:

- Contains a container div with a class container for layout.
- Displays a heading indicating the checkout process.

Cart Summary:

- Lists the items in the shopping cart along with their prices.
- Retrieves cart items using Java backend operations and iterates through them to display product names and prices.
- Calculates the total price of all items in the cart.

Billing Details Form:

- Provides input fields for the user to enter their full name, email, address, city, state, and zip code.
- Utilizes Font Awesome icons for visual representation of form fields.
- Submits the form data to a servlet (CheckoutServlet) for further processing.

Payment Method:

- Offers a radio button option for "Cash on Delivery" payment method.
- Allows the user to select their preferred payment method.

Submit Payment Button:

- Triggers form submission to initiate the checkout process.

Footer Section:

- Includes an included HTML file Footer.html for footer content.

JavaScript Functionality:

- Includes a script tag linking to Common.js for common JavaScript functionalities, though the script itself is not provided in the code snippet.

Servlet (CheckoutServlet):

- The CheckoutServlet class is responsible for handling HTTP requests related to the checkout process.
- It overrides the doPost method to handle POST requests, which typically occur when the user submits the checkout form.

- In the doPost method, it retrieves the user's input data (name, email, address, city, state, zip) from the request parameters.
- It then creates an instance of the Checkout class and calls the addCheckout method to add the checkout details to the database.

After successfully adding the checkout details, it redirects the user to an order confirmation page (OrderConfirmation.jsp).

Java Class (Checkout):

- The Checkout class contains a method addCheckout which adds checkout details (name, email, address, city, state, zip) to the database.
- It establishes a connection to the database using JDBC and inserts the checkout details into the orders table.

Integration with "Checkout" Page Description:

- The CheckoutServlet serves as the backend handler for the form submission in the "Checkout" page.
- When the user submits the checkout form, the CheckoutServlet retrieves the form data, processes it, and stores it in the database using the Checkout class.
- After successful submission, the user is redirected to an order confirmation page to confirm their order.
- Overall, this integration adds functionality to the "Checkout" page, enabling users to submit their billing details and complete the checkout process successfully.

1.12.3 Codes

Checkout.jsp

```
<%--  
Document : Checkout  
Created on : Apr 12, 2024, 10:05:16AM  
Author : hp  
--%>  
  
<%@ page import="java.util.Map" %>  
<%@ page import="java.util.HashMap" %>  
<%@ page import="java.util.Map.Entry" %>  
<%@page import="java.util.List"%>  
<%@page import="Model.DAO"%>  
<%@page import="Model.ShoppingCartObj"%>  
  
<!DOCTYPE html>  
<html>  
<head>  
<title>Checkout</title>  
<link href="CSS/Common.css" rel="stylesheet" type="text/css"/>  
<link href="CSS/Navbar.css" rel="stylesheet" type="text/css"/>  
<link href="CSS/Footer.css" rel="stylesheet" type="text/css"/>  
<link rel="stylesheet" type="text/css" href="CSS/Checkout.css"/>  
<!-- Add Font Awesome CDN -->  
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css">  
</head>  
<body style="background-color: #000000; color: #ffffff;">  
<%@ include file="Navbar.html" %>  
<div class="container">  
<h2><i class="fas fa-shopping-cart"></i> Checkout</h2>  
<h3>Cart <i class="fas fa-shopping-cart"></i></h3>  
<table>  
<tr>  
<th>Product</th>  
<th>Price</th>  
</tr>  
<%  
    List<ShoppingCartObj> cartItems = DAO.getAllCartItems();  
    for (ShoppingCartObj item : cartItems) {  
%>  
<tr>  
<td><%= item.getProductName() %></td>  
<td><%= item.getProductPrice() %></td>  
</tr>  
<% } %>  
<tr>  
<%  
    int total = 0;  
    for (ShoppingCartObj item : cartItems) {  
        total += item.getTotalPrice();  
    }  
%>  
<td><b>Total:</b></td>  
<td><b>$<%= total %></b></td>  
</tr>  
</table>  
  
<h3>Billing Details <i class="fas fa-user"></i></h3>  
<form action="CheckoutServlet" method="post">  
<div class="form-group">  
<label><i class="fas fa-user"></i> Full Name:</label><br>  
<input type="text" name="fullName" required placeholder="John Doe"><br>  
</div>
```

```

<h3>Billing Details <i class="fas fa-user"></i></h3>
<form action="CheckoutServlet" method="post">
    <div class="form-group">
        <label><i class="fas fa-user"></i> Full Name:</label><br>
        <input type="text" name="fullName" required placeholder="John Doe"><br>
    </div>
    <div class="form-group">
        <label><i class="fas fa-envelope"></i> Email:</label><br>
        <input type="email" name="email" required placeholder="john@example.com"><br>
    </div>
    <div class="form-group">
        <label><i class="fas fa-map-marker-alt"></i> Address:</label><br>
        <input type="text" name="address" required placeholder="123 Street, City"><br>
    </div>
    <div class="form-group">
        <label><i class="fas fa-city"></i> City:</label><br>
        <input type="text" name="city" required placeholder="City"><br>
    </div>
    <div class="form-group">
        <label><i class="fas fa-flag"></i> State:</label><br>
        <input type="text" name="state" required placeholder="State"><br>
    </div>
    <div class="form-group">
        <label><i class="fas fa-mail-bulk"></i> Zip:</label><br>
        <input type="text" name="zip" required placeholder="Zip"><br>
    </div>
    <input type="hidden" name="action" value="checkout" /> <!-- Hidden field for action --&gt;
&lt;h3&gt;Payment Method &lt;i class="fas fa-credit-card"&gt;&lt;/i&gt;&lt;/h3&gt;
&lt;div class="form-group"&gt;
    &lt;label for="cashOnDelivery"&gt;Cash on Delivery&lt;/label&gt;
    &lt;input type="radio" id="cashOnDelivery" name="paymentMethod" value="cod" checked&gt;
&lt;/div&gt;

    &lt;input type="submit" value="Submit Payment" class="btn btn-darkblue"&gt;
&lt;/form&gt;
&lt;/div&gt;

&lt;%@ include file="Footer.html" %&gt;
&lt;script src="JS/Common.js"&gt;&lt;/script&gt;
&lt;/body&gt;
&lt;/html&gt;
</pre>

```

Checkout.css

```
 1  /* CSS */  
 2  
 3  
 4  body {  
 5      font-family: Arial, sans-serif;  
 6      background-color: #000000;  
 7      color: #ffffff;  
 8  }  
 9  
10 .container {  
11     background-color: #000000;  
12     border: 1px solid #dddddd;  
13     border-radius: 5px;  
14     padding: 20px;  
15     margin: 20px auto;  
16     max-width: 600px;  
17     margin-bottom: 100px;  
18     margin-top: 50px;  
19  }  
20  
21  
22 h2, h3 {  
23     color: #ffffff;  
24  }  
25  
26  
27 table {  
28     width: 100%;  
29     border-collapse: collapse;  
30     margin-bottom: 20px;  
31  }
```

```
 th, td {  
    border: 1px solid #dddddd;  
    padding: 8px;  
    text-align: left;  
}  
  
 th {  
    background-color: #ffffff;  
    color: #000000;  
}  
  
 form {  
    margin-top: 20px;  
}  
  
 .form-group {  
    margin-bottom: 20px;  
    width: 95%; /* Adjust form-group width */  
}  
  
 label {  
    font-weight: bold;  
    width: 100%; /* Adjust label width */  
    display: inline-block; /* Make labels behave like inline elements */  
    margin-bottom: 8px;  
    color: #ffffff; /* Set label text color to white */  
}  
  
.form-group input[type="text"],  
.form-group input[type="email"],  
.form-group select,
```

```
.form-group select,  
.form-group textarea {  
    width: 100%; /* Adjust input, select, and textarea width */  
    padding: 10px;  
    border-radius: 3px;  
    border: 1px solid #dddddd;  
    background-color: #000000; /* Set background color to black */  
    color: #ffffff; /* Set text color to white */  
}  
  
select {  
    padding: 10px;  
    border-radius: 3px;  
    border: 1px solid #dddddd;  
}  
  
.btn {  
    background-color: #0dlb33;  
    color: white;  
    padding: 15px 20px;  
    border: none;  
    border-radius: 8px;  
    cursor: pointer;  
    font-size: 16px;  
    text-transform: uppercase;  
}  
  
.btn:hover {  
    background-color: #303f59;  
}
```

```
    .btn:active {
        background-color: #303f59;
    }

    .btn:focus {
        outline: none;
    }

    .row {
        display: flex;
        flex-wrap: wrap;
        margin-left: -10px;
        margin-right: -10px;
    }

    .col-25, .col-50, .col-75 {
        padding-left: 10px;
        padding-right: 10px;
    }

    .col-25 {
        flex: 25%;
    }

    .col-50 {
        flex: 50%;
    }

    .col-75 {
        flex: 75%;
    }

    @media screen and (max-width: 600px) {
        .row {
            flex-direction: column;
        }
    }

    .form-group {
        display: flex;
        align-items: center;
    }

    label {
        margin-right: 10px; /* Adjust as needed */
    }
```

CheckoutServlet.java

```
* @param request servlet request
* @param response servlet response
* @throws ServletException if a servlet-specific error occurs
* @throws IOException if an I/O error occurs
*/
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
        /* TODO output your page here. You may use following sample code. */
        out.println("<!DOCTYPE html>");
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet CheckoutServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet CheckoutServlet at " + request.getContextPath() + "</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
/**
 * Handles the HTTP <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
/*
package Controller;

import Model.Checkout;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

/**
 * @author hp
 */
@WebServlet(name = "CheckoutServlet", urlPatterns = {"/CheckoutServlet"})
public class CheckoutServlet extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
}
```

ava

```

66     *
67     * @param request servlet request
68     * @param response servlet response
69     * @throws ServletException if a servlet-specific error occurs
70     * @throws IOException if an I/O error occurs
71     */
72     @Override
73     protected void doPost(HttpServletRequest request, HttpServletResponse response)
74         throws ServletException, IOException {
75         String name=request.getParameter("fullName");
76         String mail=request.getParameter("email");
77         String add=request.getParameter("address");
78         String citys=request.getParameter("city");
79         String states=request.getParameter("state");
80         int zips=Integer.parseInt(request.getParameter("zip"));
81
82         Checkout c= new Checkout();
83         c.addCheckout(name,mail,add,citys,states,zips);
84
85         // Redirect to orderConfirmation.jsp
86         response.sendRedirect("OrderConfirmation.jsp");
87
88         // processRequest(request, response);
89     }
90
91     /**
92      * Returns a short description of the servlet.
93      *
94      * @return a String containing servlet description
95      */
96     @Override

```

Checkout.java

```

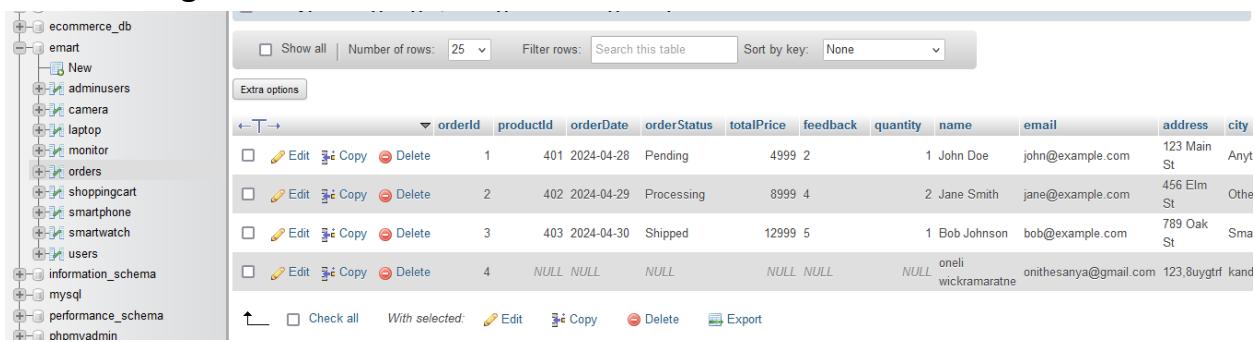
4  | /*
5   * package Model;
6
7   import java.sql.Connection;
8   import java.sql.DriverManager;
9   import java.sql.SQLException;
10  import java.sql.Statement;
11  import java.util.logging.Level;
12  import java.util.logging.Logger;
13
14  /**
15   * 
16   * @author hp
17  */
18  public class Checkout {
19      Statement st;
20  public void addCheckout(String name, String mail, String add, String citys, String states, int zips) {
21      connectToDB();
22      String query = "INSERT INTO orders(name,email,address,city,state,zip) VALUES('"+name+"','"+mail+"','"+add+"','"+citys+"','"+states+"','"+zips+"')";
23      try {
24          st.executeUpdate(query);
25      } catch (SQLException ex) {
26          Logger.getLogger(Checkout.class.getName()).log(Level.SEVERE, null, ex);
27      }
28  }
29
30  private void connectToDB() {
31      String driver = "com.mysql.cj.jdbc.Driver";

```

```

1   String driver = "com.mysql.cj.jdbc.Driver";
2   String url = "jdbc:mysql://localhost:3306/emart";
3
4   try {
5       Class.forName(driver);
6       Connection con = DriverManager.getConnection(url, "root", "");
7       st = con.createStatement();
8
9   } catch (ClassNotFoundException | SQLException ex) {
10      Logger.getLogger(Checkout.class.getName()).log(Level.SEVERE, null, ex);
11  }
12
13
14 }
15
16 }
```

Values being entered to the database.



	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	orderId	productId	orderDate	orderStatus	totalPrice	feedback	quantity	name	email	address	city
	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	1	401	2024-04-28	Pending	4999	2	1	John Doe	john@example.com	123 Main St	Anyt
	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	2	402	2024-04-29	Processing	8999	4	2	Jane Smith	jane@example.com	456 Elm St	Othe
	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	3	403	2024-04-30	Shipped	12999	5	1	Bob Johnson	bob@example.com	789 Oak St	Sma
	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	4	NULL	NULL	NULL	NULL	NULL	NULL	oneli wickramaratne	onithesanya@gmail.com	123,8uygrtf kand	

Up ▲ Check all With selected: Edit Copy Delete Export

1.13 Contact page

Done by : G.O.Wickramaratne
Student ID : 28039

1.13.1 Web page

The screenshot displays the E-Mart contact page with two side-by-side views: a desktop version on the left and a mobile version on the right.

Desktop View (Left):

- Contact Information:** A section encouraging users to experience the E-Mart difference, contact them for flagship models, and provides a phone number (+94 00 000 0000), email (demo@gmail.com), and address (70, Horton Place, Colombo, Sri Lanka).
- Map:** A Google map showing the location of E-Mart at 70, Horton Place, Colombo. The map includes coordinates (6°48'53.0"N 80°33'0"E) and a "View larger map" link.
- Social Media:** Icons for Facebook, Instagram, and TikTok.

Mobile View (Right):

- Form Fields:** First Name (jamal), Last Name (kageyama), Email (demo@gmail.com), and Phone Number (+94 000 000 0000).
- Select Subject:** Radio buttons for General Inquiry (selected), Product Inquiry, Technical Assistance, and Other.
- Message Area:** A text input field for the message and a "Send Message" button.

Footer (Bottom):

- E-Mart Logo:** featuring a shopping bag icon and the text "E-MART ELECTRONICS".
- Text:** "From cutting-edge processors to AI-powered technologies, E-Mart brings you the latest innovations in electronics. Elevate your experience and embrace the future with E-Mart."
- Links:** Company (About E-Mart, Developers), My Mart (My Profile, Ongoing Orders), and Contact Us (+94 77 696 9696, mail.emart@gmail.com).

1.13.2 Documentation : Contact page implementation

Header Section:

- Includes the document title "Contact Page".
- Imports necessary CSS and JavaScript libraries, such as Bootstrap and FontAwesome.
- Links to external CSS files for styling.

Navigation Bar:

- Utilizes an included HTML file Navbar.html to display navigation links for easy website traversal.

Main Content Section:

- Features a central wrapper div with a class wrapper-center for positioning.
- Contains a graphic image (/letter_send 1.png) in the background for visual appeal.

Contact Information:

- Provides contact details such as phone number, email address, and physical address for E-Mart.
- Utilizes FontAwesome icons for visual representation of contact information.

Google Map Integration:

- Embeds a Google Maps iframe to display the location of E-Mart.

Social Media Links:

- Displays icons for Facebook, Instagram, and TikTok, possibly linking to E-Mart's social media profiles.

Contact Form:

- Allows users to fill out their first name, last name, email, phone number, select a subject from predefined options (like General Inquiry, Product Inquiry, Technical Assistance, or Other), and write a message.
- The "Send Message" button triggers a JavaScript function to generate a mailto link with the user's input data and opens the default email client to compose an email to the specified address (oneliwickramaratne@gmail.com).

Footer Section:

- Includes an included HTML file Footer.html for footer content, which might contain additional navigation links, copyright information, etc.

JavaScript Functionality:

- Defines an event listener for the "Send Message" button to gather form input data and construct a mailto link for emailing the message to a specified email address.
- The form data (first name, last name, email, phone number, selected subject, and message) are encoded and appended to the email's subject and body in the mailto link.

1.13.3 Codes

Contact.jsp

```
6  <%@page contentType="text/html" pageEncoding="UTF-8"%>
7  <!DOCTYPE html>
8
9  <html>
10 <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>Contact Page</title>
13     <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
14     <link href="CSS/Common.css" rel="stylesheet" type="text/css"/>
15     <link href="CSS/Navbar.css" rel="stylesheet" type="text/css"/>
16     <link href="CSS/Footer.css" rel="stylesheet" type="text/css"/>
17     <link href="CSS/Contact.css" rel="stylesheet" type="text/css"/>
18     <script src="https://kit.fontawesome.com/7942e54de0.js" crossorigin="anonymous"></script>
19 </head>
20 <%@ include file="Navbar.html" %>
21 <body>
22     <div class="wrapper-center contact-page-content">
23         
24         <div class="content-cards">
25             <div class="content-card bg-accent-2">
26                 <h3 class="c-card-title">Contact Information</h3>
27                 <p class="description">Experience the E-Mart difference for yourself! Contact us today to discuss flagship models to budget-friendly options. We're here to help you find the perfect fit for your needs.</p>
28                 <div class="contactlinks">
29                     <div class="contact-link">
30                         <i class="fa-solid fa-phone-volume"></i>
31                         <span class="c-link-text">+94 00 000 0000</span>
```

```

32         </div>
33     <div class="contact-link">
34         <i class="fa-solid fa-envelope"></i>
35         <span class="c-link-text">demo@gmail.com</span>
36     </div>
37     <div class="contact-link">
38         <i class="fa-solid fa-location-dot"></i>
39         <span class="c-link-text">70, Horton Place, Colombo, Sri Lanka</span>
40     </div>
41 </div>
42 <div class="map">
43     <iframe src="https://www.google.com/maps/embed?pb=!1m7!1m2!1d3961.630551289352!2d80.54815007589576!3d6.814710993182925!2m3!1f0!2f0!3f0!3m2!1i1
44     width="400" height="225" style="border:0;" allowfullscreen="" loading="lazy" referrerPolicy="no-referrer-when-downgrade"></iframe>
45 </div>
46 <div class="socialButtons">
47     <i class="fa-brands fa-facebook"></i>
48     <i class="fa-brands fa-square-instagram"></i>
49     <i class="fa-brands fa-tiktok"></i>
50 </div>
51 </div>
52 </div>
53 <div class="c-contact-form">
54     <form action="" class="contact-form">
55         <div class="contact-form-row">
56             <div class="contact-input-area">
57                 <label for="firstName" class="form-label">First Name</label>
58                 <input type="text" class="contact-form-control" id="firstName" placeholder="jamal">
59             </div>
60             <div class="contact-input-area">
61                 <label for="lastName" class="form-label">Last Name</label>
62                 <input type="text" class="contact-form-control" id="lastName" placeholder="kageyama">
63             </div>
64             <div class="contact-form-row">
65                 <div class="contact-input-area">
66                     <label for="email" class="form-label">Email</label>
67                     <input type="email" class="contact-form-control" id="email" placeholder="demo@gmail.com">
68                 </div>
69                 <div class="contact-input-area">
70                     <label for="phoneNumber" class="form-label">Phone Number</label>
71                     <input type="text" class="contact-form-control" id="phoneNumber" placeholder="+94 000 000 0000">
72                 </div>
73             </div>
74             <div class="contact-form-row">
75                 <div class="contact-input-area">
76                     <label for="subject" class="form-label">Select Subject?</label>
77                     <div class="subjects">
78                         <div class="radio-input-area">
79                             <input class="form-check-input" type="radio" checked name="gridRadios" id="gridRadios" value="option2">
80                             <span class="checkmark"></span>
81                             <label class="form-check-label" for="gridRadios">General Inquiry</label>
82                         </div>
83                         <div class="radio-input-area">
84                             <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios1" value="option2">
85                             <label class="form-check-label" for="gridRadios1">Product Inquiry</label>
86                         </div>
87                         <div class="radio-input-area">
88                             <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios2" value="option2">
89                             <label class="form-check-label" for="gridRadios2">Technical Assistance</label>
90                         </div>
91                         <div class="radio-input-area">
92                             <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios3" value="option2">
93                             <label class="form-check-label" for="gridRadios3">Other</label>
94                         </div>
95                     </div>
96                 </div>
97             </div>
98             <div class="contact-form-row">
99                 <div class="contact-input-area">
100                     <label for="message" class="form-label">Message</label>
101                     <textarea class="contact-form-control" id="message" rows="3" placeholder="Type your message here"></textarea>
102                 </div>
103             </div>
104             <div class="contact-form-row c-action">
105                 <button id="sendMessageBtn" type="button" class="btn primary-btn order-btn">Send Message</button>
106             </div>
107         </div>
108     </form>
109     <div>
110         <script src="Footer.html" ></script>
111         <script src="JS/Common.js" ></script>
112         <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" ></script>
113     </div>

```

```

        ...
    </div>
    <% include file="Footer.html" %>
    <script src="JS/Common.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script>

    document.getElementById("sendMessageBtn").addEventListener("click", function () {
        var firstName = document.getElementById("firstName").value;
        var lastName = document.getElementById("lastName").value;
        var email = document.getElementById("email").value;
        var phoneNumber = document.getElementById("phoneNumber").value;
        var subject = document.querySelector('input[name="gridRadios"]:checked').nextElementSibling.textContent;
        var message = document.getElementById("message").value;

        var mailtoLink = "mailto:oneiliwickramarathne@gmail.com"
            + "?subject=" + encodeURIComponent(subject)
            + "&body=" + encodeURIComponent("First Name: " + firstName + "\n"
            + "Last Name: " + lastName + "\n"
            + "Email: " + email + "\n"
            + "Phone Number: " + phoneNumber + "\n"
            + "Subject: " + subject + "\n"
            + "Message: " + message);

        window.location.href = mailtoLink;
    });
</script>
</body>
</html>

```

Contact.css

```

/* CSS for responsiveness */
.bg-accent-1 {
    background-color: #192639;
}

.bg-accent-2 {
    background-color: #1c1d35;
}

.card-bg {
    background-color: #d9d9d9;
}

.bunnton-bg {
    background-color: #151663;
}

body {
    overflow-x: hidden; /* Only hide horizontal overflow */
    width: 100vw; /* Fix typo */
    height: 100vh; /* Fix typo */
}

.contact-page-content {
    display: flex;
    min-height: 100vh;
    flex-direction: column;
}

.bg-graphic {
    position: absolute;
    top: -10px;
    left: -10px;
    width: 20px;
    height: 20px;
    border-radius: 50%;
    background-color: #192639;
    transform: rotate(45deg);
}

```

```
.bg-graphic {
  position: absolute;
  bottom: 0;
  right: 0;
  left: 0;
  margin: auto;
}

.content-cards {
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
  gap: 40px;
  padding: 20px;
}

.content-card {
  border-radius: 10px;
  padding: 30px;
  display: flex;
  flex-direction: column;
  gap: 20px;
  max-width: 400px; /* Set a maximum width for content cards */
}

.content-card .description {
  font-family: 'Inter';
  font-style: normal;
  font-weight: 400;
```

```
[-] .contactlinks {
    display: flex;
    flex-direction: column;
    gap: 15px;
}

[-] .contact-link {
    display: flex;
    flex-direction: row;
    align-items: center;
    gap: 20px;
}

[-] .c-link-text {
    font-family: 'Poppins';
    font-style: normal;
    font-weight: 400;
    font-size: 16px;
    line-height: 24px;
    color: #FFFFFF;
}

[-] .map iframe {
    width: 100%;
    height: 300px;
    border-radius: 10px;
}

[-] .socialButtons {
    display: flex;
```

```
.16  
.17  .socialButtons i {  
.18      background-color: #1B1B1B;  
.19      font-size: 20px;  
.20      border-radius: 50%;  
.21  }  
.22  
.23  .c-contact-form {  
.24      display: flex;  
.25      flex-direction: column;  
.26      padding: 30px;  
.27      border-radius: 10px;  
.28      margin-top: 20px;  
.29      background-color: #d9d9d9;  
.30      color: black;  
.31      max-width: 600px; /* Set a maximum width for the form */  
.32      width: 100%; /* Allow the form to take full width */  
.33  }  
.34  
.35  .contact-form {  
.36      display: flex;  
.37      flex-direction: column;  
.38      gap: 40px;  
.39  }  
.40  
.41  .contact-form-row {  
.42      display: flex;  
.43      flex-wrap: wrap;  
.44      gap: 40px;  
.45  }
```

```
.contact-input-area {
    flex-grow: 1;
}

.radio-input-area {
    display: flex;
    gap: 10px;
    padding: 15px 0px;
}

.subjects {
    display: flex;
    gap: 30px;
}

.form-label {
    font-family: 'Poppins';
    font-style: normal;
    font-weight: 700;
    font-size: 18px;
    line-height: 20px;
    color: #000000;
}

.contact-form-control {
    width: 100%;
    padding: 10px 0;
    background-color: transparent;
    border: 0;
    border-bottom: 2px solid #8D8D8D;
```

```
79  .c-action {
80      width: 100%;
81      display: flex;
82      align-items: flex-end;
83      justify-content: flex-end;
84  }
85
86  .radio-input-area input:checked {
87      background-color: #15133a;
88  }
89  /* Add this CSS for the button styling */
90  .primary-btn {
91      background-color: #151663; /* Dark blue background color */
92      color: white; /* White text color */
93      border: none;
94      padding: 10px 20px;
95      border-radius: 5px;
96      cursor: pointer;
97      transition: background-color 0.3s; /* Smooth transition on hover */
98  }
99
100 .primary-btn:hover {
101     background-color: transparent; /* Change background color to transparent on hover */
102     color: #151663; /* Change text color to dark blue on hover */
103     border: 2px solid #151663; /* Add border on hover */
104 }
105
106 .primary-btn:focus {
107     outline: none; /* Remove outline on focus */
108 }
109
110 /* CSS for responsiveness */
111 /* Your existing CSS rules */
112
113 .socialButtons i {
114     background-color: #1B1B1B;
115     font-size: 20px;
116     border-radius: 50%;
117     color: white; /* Set the icon color to white */
118 }
119
120 /* Add this CSS to specifically target the Facebook icon */
121 .socialButtons .fa-facebook {
122     color: #3b5998; /* Facebook blue color */
123 }
124 /* Set the color of phone, email, and location icons to white */
125 .contact-link i {
126     color: white;
127 }
128
129 .nav-link.support-link {
130     opacity: 1;
131 }
```

1.14 Product Category page

Done by : W.M.C.S.Wijesinghe

Student ID: 30143

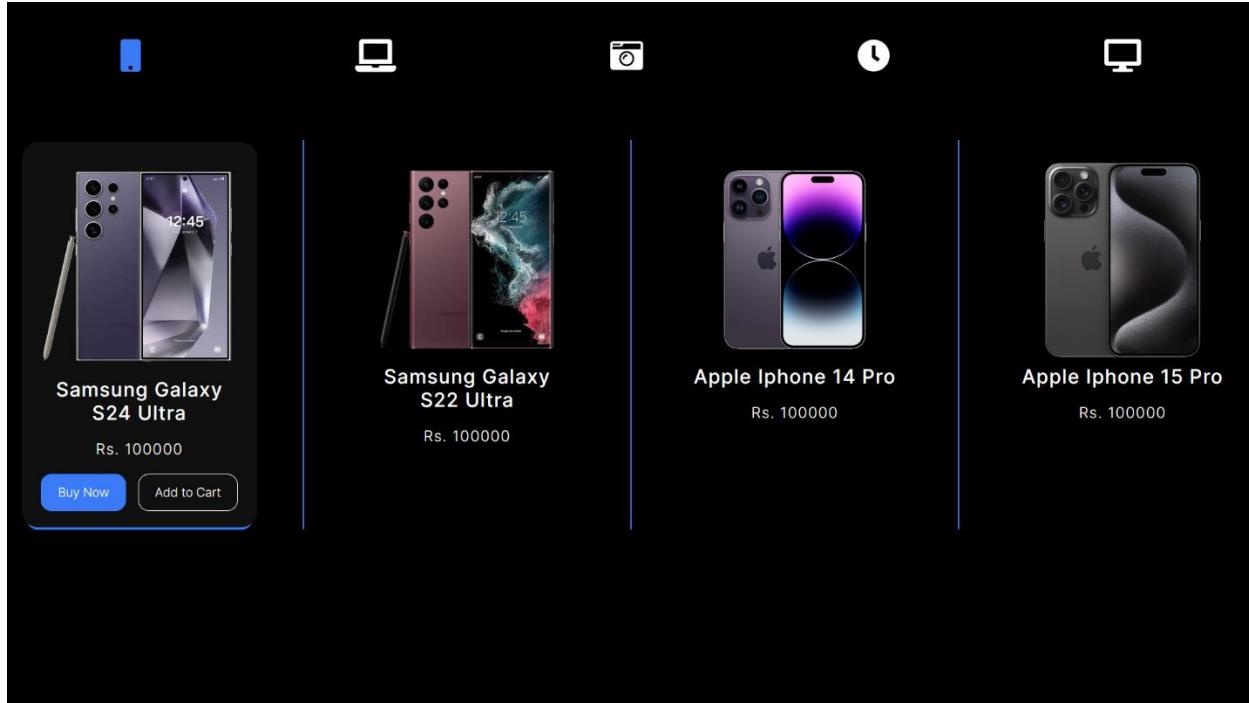


Figure 1: Front End Design for the Product Category Page

- Top row of the page was designated for category selection. FontIcons were used for this purpose, and they have CSS properties attached which change the colour and size upon hovering.

```

<div class="category-icons">
    <div>
        <i class="fas fa-mobile" style="color: #3b7bf8;"></i>
    </div>
    <div>
        <a href="ProductCategory_Laptops.jsp">
            <i class="fas fa-laptop"></i>
        </a>
    </div>
    <div>
        <a href="ProductCategory_Cameras.jsp">
            <i class="fas fa-camera-retro"></i>
        </a>
    </div>
    <div>
        <a href="ProductCategory_Smartwatches.jsp">
            <i class="fas fa-clock"></i>
        </a>
    </div>
    <div>
        <a href="ProductCategory_Monitors.jsp">
            <i class="fas fa-desktop"></i>
        </a>
    </div>
</div>

```

Figure 2: HTML code for the Font Icons

```

/*Category Icons hover effect*/
.category-icons i:hover {
    color: #3b7bf8;
    transform: translateY(-5px);
}

```

Figure 3: CSS code for changing the colour and size upon hovering.

- Each product is displayed as an HTML card. CSS code is written to change the background colour, and Javascript code is written to direct the User to a servlet. This allows us to create dynamic web pages for each product.

```

<div class="card">
    <form id="productForm" action="ProductServlet" method="get">
        <input type="hidden" name="id" value="1">
        <a href="#" onclick="document.getElementById('productForm').submit();">
            
        </a>
    </form>
    <h2>Samsung Galaxy S24 Ultra</h2>
    <h3>Rs. 100000</h3>

    <form id="buttonForm" action="#" method="post">
        <input type="hidden" id="productID" name="id" value="1">
        <div class="btn-container">
            <button class="buy" onclick="document.getElementById('buttonForm').action='CheckoutServlet';">Buy Now</button>
            <button class="cart" onclick="document.getElementById('buttonForm').action='CartServlet';">Add to Cart</button>
        </div>
    </form>
</div>

```

Figure 4: Product Card and associated Javascript

- Above is the Code Snippet associated with the Buy-Now and Add-to-Cart buttons. Each button has Javascript code attached to them, which will direct the User to the relevant pages.

- Buttons are enclosed within <form> tags and has a hidden Input, which is used to pass the product ID to the Shopping Servlet.
- Upon clicking the card, the user is then directed to the ProductInfoServlet

```
@WebServlet(name = "ProductInfoServlet", value = "/ProductInfoServlet")
public class ProductInfoServlet extends HttpServlet {
    @Override no usages
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        int productID = Integer.parseInt(request.getParameter("productID"));

        //Establishing a connection
        String url = "jdbc:mysql://localhost:3306/emart";
        String user = "root";
        String password = "";

        Connection conn = null;
```

Figure 5: Code snippet used to connect to the Database in the ProductInfoServlet

- Above is the opening code snippet to the ProductInfoServlet. The ProductInfoServlet will connect to the database using JDBC.
- Then it runs SQL queries to retrieve all the necessary data from the database. The productID is used as the identifier.

```

try{
    Class.forName("com.mysql.jdbc.Driver");
    conn = DriverManager.getConnection(url, user, password);

    //Retrieve the data from the database
    String query = "select * from smartphone where productID = ?";
    PreparedStatement st = conn.prepareStatement(query);
    st.setInt(1, productID);
    ResultSet rs = st.executeQuery();

    //Forward the result to the JSP
    if(rs.next()) {
        request.setAttribute("productName", rs.getString("productName"));
        request.setAttribute("price", rs.getInt("price"));
        request.setAttribute("brand", rs.getString("brand"));
        request.setAttribute("modelName", rs.getString("modelName"));
        request.setAttribute("storageCapacity", rs.getString("storageCapacity"));
        request.setAttribute("screenSize", rs.getDouble("screenSize"));
        request.setAttribute("color", rs.getString("color"));
        request.setAttribute("productDescription", rs.getString("productDescription"));
        request.setAttribute("photo1", rs.getString("photo1"));
        request.setAttribute("photo2", rs.getString("photo2"));
        request.setAttribute("photo3", rs.getString("photo3"));
        request.setAttribute("photo4", rs.getString("photo4"));
    }
    request.getRequestDispatcher("ProductInfoTemplate.jsp").forward(request, response);

    rs.close();
    st.close();
    conn.close();
}

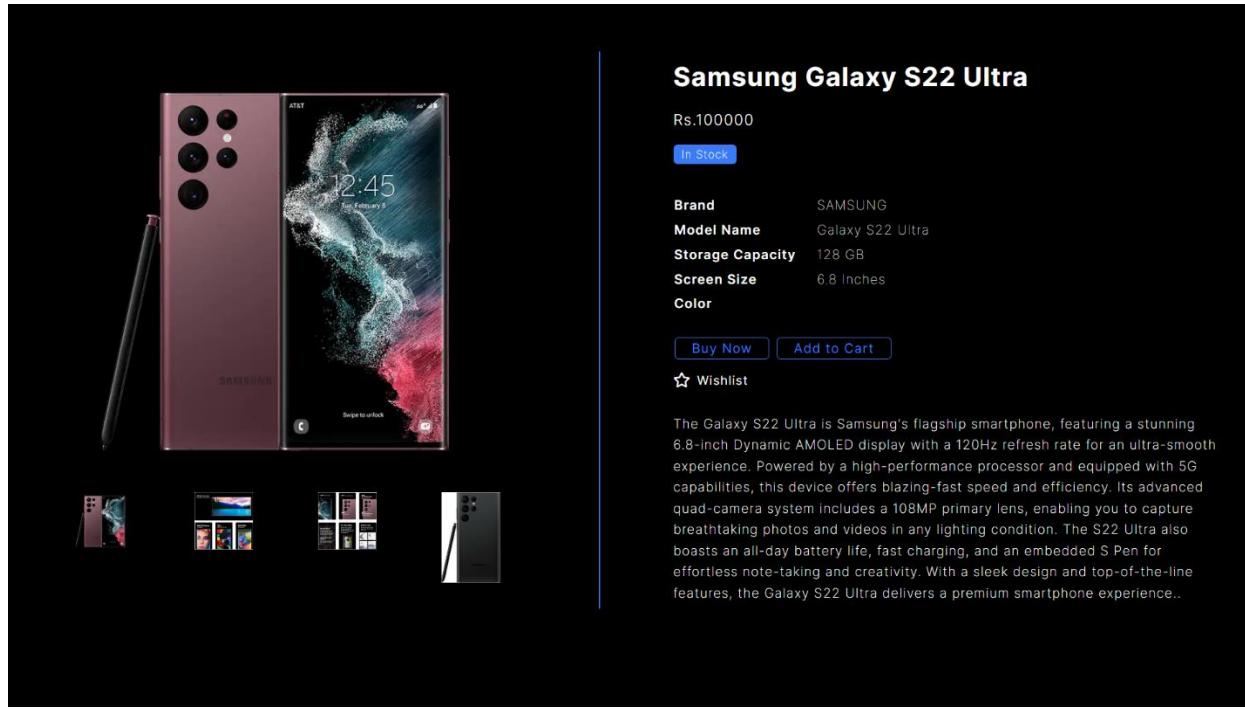
}catch(ClassNotFoundException | SQLException e){
    e.printStackTrace();
}

```

Figure 6: JDBC code for retrieving and passing the data

- As shown in the above image, JDBC is used to connect and then retrieve the appropriate data from the database.
- If the query returns any results, it extracts various attributes of the product. These attributes are then set as attributes in the HTTP request object.
- The request object is then forwarded to a JSP page that's used as the template for displaying product information.

1.14 Product Information page



```
<%\nString productName = (String) request.getAttribute("productName");\nint price = (int) request.getAttribute("price");\nString brand = (String) request.getAttribute("brand");\nString modelName = (String) request.getAttribute("modelName");\nString storageCapacity = (String) request.getAttribute("storageCapacity");\ndouble screenSize = (double) request.getAttribute("screenSize");\nString color = (String) request.getAttribute("color");\nString productDescription = (String) request.getAttribute("productDescription");\nString photo1 = (String) request.getAttribute("photo1");\nString photo2 = (String) request.getAttribute("photo2");\nString photo3 = (String) request.getAttribute("photo3");\nString photo4 = (String) request.getAttribute("photo4");\n%>
```

Figure 7: Attribute retrieval in the JSP page

- JSP code for the Template Page, retrieves the various attributes that were forwarded from the ProductInfoServlet.
- Then they are casted into respective data types and are assigned to variables.

```
<!--Left gallery-->
<div class="left-column">
    

    <div class="small-img-row">
        <div class="small-img-col">
            
        </div>
        <div class="small-img-col">
            
        </div>
        <div class="small-img-col">
            
        </div>
        <div class="small-img-col">
            
        </div>
    </div>
</div>
```

Figure 8: Product Gallery code snippet

- Above is the code snippet used to create an image gallery to display images. It is used to display the product images using `` tags with the `src` attribute set to the value of photos.

```

<!--Right product info and stuff-->


<h1 class="name"><%= productName %></h1>
    <h3 class="price"><%= price %></h3>

    <!--In stock/out of stock square thingy-->
    <div class="availability">In Stock</div>

    <!--Product details table-->
    <table class="info-table">
        <tr>
            <td>Brand</td>
            <td><%= brand %></td>
        </tr>
        <tr>
            <td>Model Name</td>
            <td><%= modelName %></td>
        </tr>
        <tr>
            <td>Storage Capacity</td>
            <td><%= storageCapacity %></td>
        </tr>
        <tr>
            <td>Screen Size</td>
            <td><%= screenSize %></td>
        </tr>
        <tr>
            <td>Color</td>
            <td><%= color %></td>
        </tr>
    </table>


```

Figure 9: Product Information code snippet

- Within the right column of the webpage, various details about the product are displayed. Variables that were assigned previously are used to display the values.

```

<form action="ShoppingServlet" method="post">
    <input type="hidden" name="ID" value=<%= productID %>>
    <div class="option-buttons">
        <button type="submit" class="buy-now" name="action" value="buy_now">Buy Now</button>
        <button type="submit" class="add-to-cart" name="action" value="add_to_cart">Add to Cart</button>
    </div>
</form>

```

Figure 10: Code snippet for the buy now and add to cart buttons

- Buy-Now and Add-to-Cart buttons are enclosed within `<form>` tags and ‘submit’ type buttons.
- The form contains a hidden input variable with the `productID`, which is passed onto the `ShoppingServlet`
- Additionally, this page is also linked to a Javascript page which is used to handle the product gallery.

```

//Gallery Script
var ProductImg = document.getElementById("ProductImg");
var SmallImg = document.getElementsByClassName("small-img");

SmallImg[0].onclick = function () {
    ProductImg.src = SmallImg[0].src;
}

SmallImg[1].onclick = function () {
    ProductImg.src = SmallImg[1].src;
}

SmallImg[2].onclick = function () {
    ProductImg.src = SmallImg[2].src;
}

SmallImg[3].onclick = function () {
    ProductImg.src = SmallImg[3].src;
}

```

Figure 11: JS code for the gallery

- The JavaScript code for the image gallery selects the main product element by its ID and assigns it to a variable, along with all elements with class small-img.
- For each small image element, an onclick event handler is assigned. Upon clicking the image, the source of the main product image is updated.
- Essentially, when the small image is clicked it triggers a function that replaces the source of the main product image with the source of the clicked image.

- The buttons within the template product are connected to the ShoppingServlet, and they operate as follows.

```

@WebServlet(name = "ShoppingServlet", value = "/ShoppingServlet")
public class ShoppingServlet extends HttpServlet {
    @Override no usages
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String productID = request.getParameter("ID");
        String action = request.getParameter("action");

        String driver = "com.mysql.jdbc.Driver";
        String url = "jdbc:mysql://localhost:3306/";
        String user = "root";
        String password = "";
        Statement statement;
        ResultSet resultSet;
    }
}

```

Figure 12: Code snippet to connect to the database.

- The Servlets requests the ID and the action of the buttons as parameters and assigns them to separate variables.

- Then it uses JDBC to connect to the database

```

String selectQuery = "select * from smartphone where productid = " + productId + "";
resultSet = statement.executeQuery(selectQuery);

//inserting data to the ShoppingCart table
while (resultSet.next()){
    String insertQuery = "INSERT INTO shoppingcart (recordId, email, productName, productPrice, quantity, description, category, iconPath, productId) VALUES
        resultSet.getInt( columnLabel: "recordId") + "'"
        resultSet.getString( columnLabel: "email") + "'"
        resultSet.getString( columnLabel: "productName") + "'"
        resultSet.getInt( columnLabel: "productPrice") + "'"
        resultSet.getInt( columnLabel: "quantity") + "'"
        resultSet.getString( columnLabel: "description") + "'"
        resultSet.getString( columnLabel: "category") + "'"
        resultSet.getString( columnLabel: "iconPath") + "'"
        resultSet.getInt( columnLabel: "productId") +
    ")";
    statement.executeUpdate(insertQuery);
}

} catch (ClassNotFoundException | SQLException ex){
    Logger.getLogger(ShoppingServlet.class.getName()).log(Level.SEVERE, null, ex);
}

if ("buy_now".equals(action)) {
    // Redirect to the "checkout" JSP page
    response.sendRedirect( s: "checkout.jsp");
} else if ("add_to_cart".equals(action)) {
    // Redirect to the "add to cart" JSP page
    response.sendRedirect( s: "ShoppingCart.jsp");
}

```

Figure 13: Code snippet to retrieve data from the table and update table

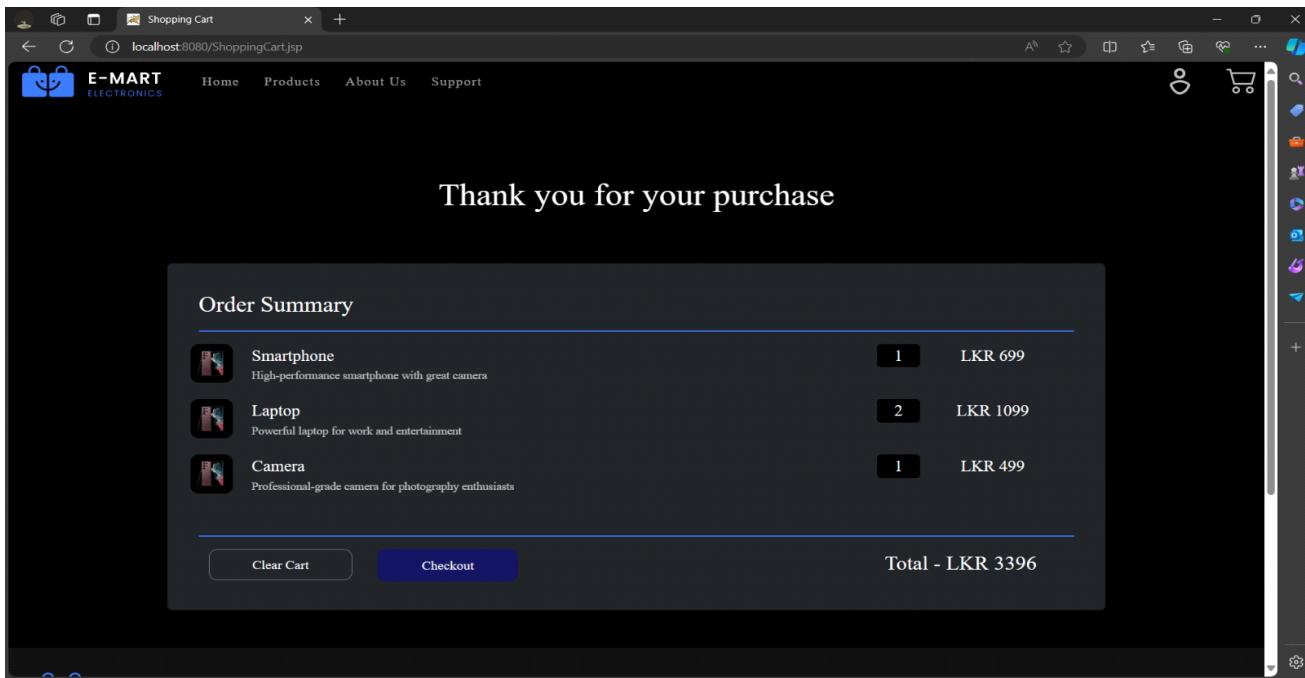
- It executes a SQL SELECT query to retrieve data from the smartphone table based on a given productID. The retrieved data is stored in a ResultSet.
- SQL INSERT query is executed and each of those retrieved data is inserted into the shoppingcart table.

1.15 Shopping Cart page

Done by : T.P.R. Fernando

Student ID: 26888

1.15.1 Web page



1.14.2 Documentation : Shopping Cart page implementation

- The page imports necessary Java classes and objects from the Model package, likely representing data objects and database access objects.
- JSP structure of the page includes the necessary meta tags, title, and links to external CSS and JavaScript files, including Bootstrap and Font Awesome libraries.
- The header section includes the navigation bar (Navbar.html), which is likely a reusable component for navigation.
- The main content consists of a title "Thank you for your purchase" and an order summary section.
- The order summary section displays a list of items in the shopping cart along with their details like product name, description, quantity, price, and total price.
- Iteration over the list of shopping cart items is done using a JSP scriptlet (`<% %>`) to retrieve items from the database and display them dynamically.

Order Actions:

- Users are provided with options to clear the cart or proceed to checkout.
- The "Clear Cart" button submits a form to a servlet named "ClearCart".

- The "Checkout" button redirects the user to a JSP page named "Checkout.jsp".
 - The total price of the items in the cart is calculated by summing up the individual total prices of each item.
 - The total price is displayed at the end of the order summary section.
 - **Footer:** The footer section includes a footer component (Footer.html), which is likely another reusable component for displaying footer information.
-
- A JavaScript function confirmClear() is defined to confirm clearing the cart before submission.
 - External JavaScript files are included at the end of the document for additional functionality (Bootstrap and Common.js).
 - Custom CSS styles are provided to style various elements of the page, including buttons, order summary, and item details.
 - DAO class is used for communicating with the database mainly about retrieving, inserting and updating products from the database.
Methods like getAllSmartphones(), getAllLaptops(), getAllCamera(), getAllSmartwatches(), and getAllMonitor() fetch product data from their respective tables in the database.

Managing Shopping Cart:

- getAllCartItems(): Retrieves all items in the shopping cart from the database, creating ShoppingCartObj instances for each item.
- removeAllCartItems(): Deletes all items from the shopping cart table in the database

Home.Jsp:

- getNewArrivals(): Retrieves the newest products, possibly smartphones, from the database and returns a list of NewestProductObj instances.

Managing Admin Orders:

- The getAdminOrders() method retrieves order details from the database, including product information derived from multiple tables using a complex SQL query.
- removeOrderItem(int orderId): Deletes a specific order item from the orders table in the database.

Admin Product Operations:

- updateLaptop(): Updates details of a laptop product in the database.
- addLaptop(): Adds a new laptop product to the database.
- deleteLaptop(): Deletes a laptop product from the database.
- updateCamera(): Updates details of a camera product in the database.
- addCamera(): Adds a new camera product to the database.
- deleteCamera(): Deletes a camera product from the database.
- updateMonitor(): Updates details of a monitor product in the database.
- addMonitor(): Adds a new monitor product to the database.
- deleteMonitor(): Deletes a monitor product from the database.
- updateSmartwatch(): Updates details of a smartwatch product in the database.
- addSmartwatch(): Adds a new smartwatch product to the database.
- deleteSmartwatch(): Deletes a smartwatch product from the database
- updateSmartphone(): Updates details of a smartphone product in the database.

- `addSmartphone()`: Adds a new smartphone product to the database.
- `deleteSmartphone()`: Deletes a smartphone product from the database.

ClearCart.java:

- The servlet imports necessary classes from the Model package and the servlet API for handling HTTP requests and responses.
- Annotations: The `@WebServlet` annotation indicates that this servlet is mapped to the URL pattern `"/ClearCart"`.
- `processRequest()`: This method is generated by default but not used in this servlet. It generates an HTML response with a title indicating the servlet's name and the request context path.
- `doGet()`: This method handles HTTP GET requests. It delegates the request processing to the `processRequest()` method.
- `doPost()`: This method handles HTTP POST requests, which are typically used for form submissions. It removes all items from the shopping cart by calling the `removeAllCartItems()` method from the DAO class. Then, it forwards the request to the "ShoppingCart.jsp" page to display the updated cart.
- `getServletInfo()`: This method returns a short description of the servlet.
- When a POST request is sent to this servlet (e.g., when the "Clear Cart" button is clicked on the shopping cart page), it removes all items from the cart by calling the `removeAllCartItems()` method from the DAO class.
- After clearing the cart, it forwards the request to the "ShoppingCart.jsp" page to display the updated cart.

Navbar:

Navigation Bar (nav):

- Positioned at the top of the webpage.
- Contains three main sections: nav-content, nav-actions, and hamburger.

Nav-content:

- Contains the brand logo and navigation links.
- The brand logo is an image.
- Navigation links include links to different pages such as Home, Products, About Us, and Support.

Nav-actions:

- Contains action buttons like Profile, Sign Up, and Shopping Cart.
- Profile button redirects to the user's profile page if logged in; otherwise, it redirects to the Sign Up page.
- Shopping Cart button redirects to the Shopping Cart page.

Hamburger:

- Represents the mobile menu icon.
- Consists of three horizontal lines.
- Used for expanding the navigation menu on smaller screens.

- CSS Styling:
- Styled as a flex container with space between its children.
- Transitions are applied for smooth animations.
- Max-height is set to 100vh for mobile responsiveness.
- Styled as a flex container with gaps between items.
- Nav links are styled with white color, opacity, and hover effects.
- Styled as a flex container with gaps between items.
- Action buttons like Profile and Shopping Cart are styled as clickable icons.
- Styled with a black background, rounded corners, and transitions for animation effects.
- Consists of three lines vertically stacked with gaps between them.
- Display set to none by default, but appears when the screen width is below 1072px (responsive design).

Media Queries:

- Adjusts the layout and styles for screens with a maximum width of 1072px.
- Changes the flex-direction of the nav and its children for better mobile responsiveness.
- Adjusts the width of the input area and other elements for smaller screens.

Footer:

- Contains various sections such as footer-details, footer-about-links, footer-page-links, footer-contact-links, and footer-extra.
- Includes social media links, company information, links to different sections/pages, and contact information.
- CSS Styling:
 - Padding is applied to provide space around the content.
 - Background is styled with a linear gradient and a box shadow to create a visually appealing effect.
 - Styled as a flex container with row direction and equal spacing between items.

1.14.3 Codes

ShoppingCart.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.util.List">
<%@page import="Model.DAO"%>
<%@page import="Model.ShoppingCartObj"%>
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Shopping Cart</title>
        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
        <link href="CSS/Common.css" rel="stylesheet" type="text/css"/>
        <link href="CSS/Navbar.css" rel="stylesheet" type="text/css"/>
        <link href="CSS/Footer.css" rel="stylesheet" type="text/css"/>
        <link href="CSS/ShoppingCart.css" rel="stylesheet" type="text/css"/>
        <script src="https://kit.fontawesome.com/7942e54de0.js" crossorigin="anonymous"></script>
    </head>

    <body>
        <%@ include file="Navbar.html" %>
        <div class="wrapper-center d-flex mt-5 pt-5 mb-5">
            <h1 class="title">Thank you for your purchase</h1>

            <div class="card bg-dark order-card">
                <div class="card-body text-white">
                    <h3 class="card-title">Order Summary</h3>
                    <br>
                    <div class="container text-center" id="orderItemsList">
                        <%
                            List<ShoppingCartObj> cartItems = DAO.getAllCartItems();
                            for (ShoppingCartObj item : cartItems) {
                        %>
                            <div class="row">
                                <div class="col-md-auto item-icon">
                                    <img alt=<%= item.getIconPath() %> src=<%= item.getIconPath() %> />
                                </div>
                                <div class="col text-start">
                                    <div class="phoneDetails">
                                        <span class="model"><%= item.getProductName() %></span>
                                        <span class="variant"><%= item.getDescription() %></span>
                                    </div>
                                </div>
                                <div class="col col-lg-2 qty-group">
                                    <!-->
                                    <div class="qty">
                                        <i class="fa-solid fa-minus button-qty bg-gray" data-qty="remove"></i>
                                    </div>-->
                                    <span class="qtyAmount"><%= item.getQuantity() %></span>
                                    <!-->
                                    <div class="qty">
                                        <i class="fa-solid fa-plus button-qty bg-accent" data-qty="add"></i>
                                    </div>-->
                                </div>
                            </div>
                            <div class="col col-lg-2">
                                LKR <%= item.getProductPrice() %>
                            </div>
                        <% } %>
                    </div>
                    <hr>
                    ...
                </div>
            <div class="container">
                <div class="row">
                    <div class="col order-btns">
                        <form action="ClearCart" method="POST">
                            <button type="submit" class="btn btn-outline-secondary text-white order-btn" onclick="return confirmClear()">Clear Cart</button>
                        </form>
                        <button type="button" class="btn primary-btn order-btn" onclick="window.location.href = 'Checkout.jsp';">Checkout</button>
                    </div>
                    <div class="col text-end order-total">
                        <span>Total - </span>
                        <%
                            int total = 0;
                            for (ShoppingCartObj item : cartItems) {
                                total += item.getTotalPrice();
                            }
                        %>
                        <span>LKR <%= total %></span>
                    </div>
                </div>
            </div>
        </div>
    </body>

```

```

<%@ include file="Footer.html" %>
<script>
    function confirmClear() {
        var confirmed = confirm("Are you sure you want to clear the cart?");
        if(confirmed) {
            return true; // Proceed with form submission
        } else {
            return false; // Cancel form submission
        }
    }
</script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
<script src="JS/Common.js"></script>
</body>
</html>

```

ShoppingCart.css

```

.br-0{
    border-right: 0 !important;
}

.title{
    margin: auto;
}

.wrapper-center{
    flex-direction: column;
    align-items: center;
}

.order-card{
    margin-top: 4rem;
    width: 1100px;
    max-width: 1200px;
    padding: 20px;
}

hr{
    border-top: 2px solid #3B7BF8 !important;
    opacity: 1 !important;
}

#orderItemsList{
    display: flex;
    flex-direction: column;
    gap: 20px;
    margin-left: -10px;
    min-height: 230px;
    max-height: 300px;
}

#orderItemsList{
    font-size: 20px;
}

```

```

body{
    background: black !important;
    color: white !important;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}

.primary-btn{
    background: #151663 !important;
    color: #fff !important;
    border: 2px solid #151663 !important;
}

.order-btn{
    padding: 7px 50px !important;
    border-radius: 10px !important;
}

.order-btns{
    display: flex;
    gap: 30px;
}

.order-total{
    margin-right: 2rem;
    font-size: 25px;
}
.tr-text{
    background-color: transparent !important;
}

.bl-0{
    border-left: 0 !important;
}

```

```
□ .qty-group{
  display: flex;
  align-items: center;
  gap: 0;
  background-color: #010101;
  width: fit-content;
  height: fit-content;
  padding: 0;
  border-radius: 5px;
}

□ .button-qty{
  padding: 10px;
  cursor: pointer;
  transition: all 0.6s ease;
}

□ .qty-add{
  border-radius: 0px 5px 5px 0;
}

□ .qty-remove{
  border-radius: 5px 0px 0px 5px;
}

  .qty-add:hover,
  .qty-remove:hover{
    transform: scale(105%);
  }

□ .qty-remove:hover{
  background-color: #46010d;
}

□ .qty-add:hover{
  background-color: #012146;
}
```

```
□ .qtyAmount{
  padding: 0 20px;
}

□ .bg-gray{
  background-color: #606060;
}

□ .bg-accent{
  background-color: #387BF8;
}

□ .item-icon{
  background-color: black;
  border-radius: 10px;
  width: 50px;
  height: 50px;
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
}

  .item-icon img{
    height: 40px;
    width: auto;
  }

□ .phoneDetails{
  display: flex;
  flex-direction: column;
  align-items: start;
  justify-content: flex-start;
  margin-left: 10px;
}


```

```
□ .variant{
  font-size: 0.7em;
  opacity: 0.7;
}
```

ShoppingCartObj.java

```
public class ShoppingCartObj {  
    private String productName;  
    private int productPrice;  
    private int quantity;  
    private String description;  
    private ProductCategory category;  
    private String iconPath;  
    private int recordId;  
    private String email;  
  
    public ShoppingCartObj(String productName, int productPrice, int quantity, String description, ProductCategory category, String iconPath, int recordId, String email) {  
        this.productName = productName;  
        this.productPrice = productPrice;  
        this.quantity = quantity;  
        this.description = description;  
        this.category = category;  
        this.iconPath = iconPath;  
        this.recordId = recordId;  
        this.email = email;  
    }  
  
    public String getProductName() {  
        return productName;  
    }  
  
    public void setProductName(String productName) {  
        this.productName = productName;  
    }  
  
    public int getProductPrice() {  
        return productPrice;  
    }  
  
    public void setProductPrice(int productPrice) {  
        this.productPrice = productPrice;  
    }  
  
    public int getQuantity() {  
        return quantity;  
    }  
  
    public void setQuantity(int quantity) {  
        this.quantity = quantity;  
    }  
  
    public int getTotalPrice() {  
        return quantity * productPrice;  
    }  
  
    public String getDescription() {  
        return description;  
    }  
  
    public void setDescription(String description) {  
        this.description = description;  
    }  
  
    public ProductCategory getCategory() {  
        return category;  
    }  
  
    public void setCategory(ProductCategory category) {  
        this.category = category;  
    }  
  
    public String getIconPath() {  
        return iconPath;  
    }  
  
    public void setIconPath(String iconPath) {  
        this.iconPath = iconPath;  
    }  
}
```

```
    public int getRecordid() {
        return recordid;
    }

    public void setRecordid(int recordid) {
        this.recordid = recordid;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

DAO.java

```
package Model;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/*
 *
 * @author robin
 */
public class DAO {

    private static final String URL = "jdbc:mysql://localhost:3306/emart";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "";
```

```

public static Connection getConnection() throws SQLException {
    return DriverManager.getConnection(url:URL, user:USERNAME, password:PASSWORD);
}

public static List<Smartphone> getAllSmartphones() {
    List<Smartphone> products = new ArrayList<>();
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;
    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url:URL, user:USERNAME, password:PASSWORD);

        // Creating SQL statement
        statement = connection.createStatement();

        // SQL query to fetch from smartphone
        String query = "SELECT * FROM smartphone";

        // Executing the query
        resultSet = statement.executeQuery(string: query);

        // Iterating over the result set and adding items from smartphone
    }

    // Iterating over the result set and adding items from smartphone
    while (resultSet.next()) {
        int productId = resultSet.getInt(string: "productId");
        String productName = resultSet.getString(string: "productName");
        int price = resultSet.getInt(string: "price");
        int quantity = resultSet.getInt(string: "quantity");
        String brand = resultSet.getString(string: "brand");
        String modelName = resultSet.getString(string: "modelName");
        String productDescription = resultSet.getString(string: "productDescription");
        String category = resultSet.getString(string: "category");
        String storageCapacity = resultSet.getString(string: "storageCapacity");
        Double screenSize = resultSet.getDouble(string: "screenSize");
        String color = resultSet.getString(string: "color");
        String photo1 = resultSet.getString(string: "photo1");
        String photo2 = resultSet.getString(string: "photo2");
        String photo3 = resultSet.getString(string: "photo3");
        String photo4 = resultSet.getString(string: "photo4");

        // Creating Smartphone object and adding it to the list
        Smartphone smartphone = new Smartphone(productId, productName, price, quantity, photo1, photo2, photo3, photo4,
            brand, modelName, productDescription, category, storageCapacity, screenSize, color);
        products.add(smartphone);
    }

} catch (Exception e) {
    System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
}

System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
} finally {
    // Closing the connection, statement, and result set
    try {
        if (resultSet != null) {
            resultSet.close();
        }
        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.close();
        }
    } catch (SQLException e) {
        System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
    }
}
return products;
}

```

```

public static List<Laptop> getAllLaptops() {
    List<Laptop> products = new ArrayList<>();
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;
    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);

        // Creating SQL statement
        statement = connection.createStatement();

        // SQL query to fetch from smartphone
        String query = "SELECT * FROM laptop";

        // Executing the query
        resultSet = statement.executeQuery(string: query);

        // Iterating over the result set and adding items from smartphone
        while (resultSet.next()) {
            int productId = resultSet.getInt(string: "productId");
            String productName = resultSet.getString(string: "productName");
            int price = resultSet.getInt(string: "price");
            int quantity = resultSet.getInt(string: "quantity");
            String photo1 = resultSet.getString(string: "photo1");
            String photo2 = resultSet.getString(string: "photo2");
            String photo3 = resultSet.getString(string: "photo3");
            String photo4 = resultSet.getString(string: "photo4");
            String brand = resultSet.getString(string: "brand");
            String modelName = resultSet.getString(string: "modelName");
            String productDescription = resultSet.getString(string: "productDescription");
            String category = resultSet.getString(string: "category");
            String storageCapacity = resultSet.getString(string: "storageCapacity");
            String CPU = resultSet.getString(string: "CPU");
            String memory = resultSet.getString(string: "memory");

            // Creating Smartphone object and adding it to the list
            Laptop laptop = new Laptop(productId, productName, price, quantity, photo1, photo2, photo3, photo4,
                brand, modelName, productDescription, category, storageCapacity, CPU, memory);
            products.add(laptop);
        }
    } catch (Exception e) {
        System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
    } finally {
        // Closing the connection, statement, and result set
        try {
            if(resultSet != null) {
                resultSet.close();
            }
            if(statement != null) {
                statement.close();
            }
            if(connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
        }
    }
    return products;
}

```

```

public static List<Camera> getAllCamera() {
    List<Camera> products = new ArrayList<>();
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;
    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);

        // Creating SQL statement
        statement = connection.createStatement();

        // SQL query to fetch from smartphone
        String query = "SELECT * FROM camera";

        // Executing the query
        resultSet = statement.executeQuery(string: query);

        // Iterating over the result set and adding items from smartphone
        while (resultSet.next()) {
            int productId = resultSet.getInt(string: "productId");
            String productName = resultSet.getString(string: "productName");
            int price = resultSet.getInt(string: "price");
            int quantity = resultSet.getInt(string: "quantity");
            String photo1 = resultSet.getString(string: "photo1");
            String photo2 = resultSet.getString(string: "photo2");
            String photo3 = resultSet.getString(string: "photo3");
            String photo4 = resultSet.getString(string: "photo4");
            String brand = resultSet.getString(string: "brand");
            String modelName = resultSet.getString(string: "modelName");
            String productDescription = resultSet.getString(string: "productDescription");
            String category = resultSet.getString(string: "category");
            String formFactor = resultSet.getString(string: "formFactor");

            // Creating Smartphone object and adding it to the list
            Camera camera = new Camera(productId, productName, price, quantity, photo1, photo2, photo3, photo4,
                brand, modelName, productDescription, category, formFactor);
            products.add(.camera);
        }
    } catch (Exception e) {
        System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
    } finally {
        // Closing the connection, statement, and result set
        try {
            if (resultSet != null) {
                resultSet.close();
            }
            if (statement != null) {
                statement.close();
            }
            if (connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
        }
    }
    return products;
}

```

```

public static List<Smartwatch> getAllSmartwatches() {
    List<Smartwatch> products = new ArrayList<>();
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;
    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");
        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);
        // Creating SQL statement
        statement = connection.createStatement();
        // SQL query to fetch from smartphone
        String query = "SELECT * FROM smartwatch";
        // Executing the query
        resultSet = statement.executeQuery(string: query);
        // Iterating over the result set and adding items from smartphone
        while (resultSet.next()) {
            int productId = resultSet.getInt(string: "productId");
            String productName = resultSet.getString(string: "productName");
            int price = resultSet.getInt(string: "price");
            int quantity = resultSet.getInt(string: "quantity");
            String photo1 = resultSet.getString(string: "photo1");
            String photo2 = resultSet.getString(string: "photo2");
            String photo3 = resultSet.getString(string: "photo3");
            String photo4 = resultSet.getString(string: "photo4");
            String brand = resultSet.getString(string: "brand");
            String modelName = resultSet.getString(string: "modelName");
            String productDescription = resultSet.getString(string: "productDescription");
            String category = resultSet.getString(string: "category");
            String screenSize = resultSet.getString(string: "screenSize");
            String color = resultSet.getString(string: "color");
            // Creating Smartwatch object and adding it to the list
            Smartwatch smartwatch = new Smartwatch(productId, productName, price, quantity, photo1, photo2, photo3, photo4,
                brand, modelName, productDescription, category, screenSize, color);
            products.add(smartwatch);
        }
    } catch (Exception e) {
        System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
    } finally {
        // Closing the connection, statement, and result set
        try {
            if (resultSet != null) {
                resultSet.close();
            }
            if (statement != null) {
                statement.close();
            }
            if (connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
        }
    }
    return products;
}

```

```

public static List<Monitor> getAllMonitor() {
    List<Monitor> products = new ArrayList<>();
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;
    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url:URL, user:USERNAME, password:PASSWORD);

        // Creating SQL statement
        statement = connection.createStatement();

        // SQL query to fetch from smartphone
        String query = "SELECT * FROM monitor";

        // Executing the query
        resultSet = statement.executeQuery(string: query);

        // Iterating over the result set and adding items from smartphone
        while (resultSet.next()) {
            int productId = resultSet.getInt(string: "productId");
            String productName = resultSet.getString(string: "productName");
            int price = resultSet.getInt(string: "price");
            int quantity = resultSet.getInt(string: "quantity");
            String photo1 = resultSet.getString(string: "photo1");
            String photo2 = resultSet.getString(string: "photo2");
            String photo3 = resultSet.getString(string: "photo3");
            String photo4 = resultSet.getString(string: "photo4");
            String brand = resultSet.getString(string: "brand");
            String modelName = resultSet.getString(string: "modelName");
            String productDescription = resultSet.getString(string: "productDescription");
            String category = resultSet.getString(string: "category");
            String screenSize = resultSet.getString(string: "screenSize");
            int refreshRate = resultSet.getInt(string: "refreshRate");
            String resolution = resultSet.getString(string: "resolution");

            // Creating Smartphone object and adding it to the list
            Monitor monitor = new Monitor(productId, productName, price, quantity, photo1, photo2, photo3, photo4,
                brand, modelName, productDescription, category, refreshRate, screenSize, resolution);
            products.add(monitor);
        }
    } catch (Exception e) {
        System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
    } finally {
        // Closing the connection, statement, and result set
        try {
            if(resultSet != null){
                resultSet.close();
            }
            if(statement != null){
                statement.close();
            }
            if(connection != null){
                connection.close();
            }
        } catch (SQLException e) {
            System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
        }
    }
    return products;
}

```

```

public static List<ShoppingCartObj> getAllCartItems() {
    List<ShoppingCartObj> currentCartSnap = new ArrayList<>();
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;
    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);

        // Creating SQL statement
        statement = connection.createStatement();

        // SQL query to fetch shopping cart items
        String query = "SELECT * FROM ShoppingCart";

        // Executing the query
        resultSet = statement.executeQuery(string: query);

        // Iterating over the result set and adding items to the shopping cart list
        while (resultSet.next()) {
            String productName = resultSet.getString(string: "productName");
            int productPrice = resultSet.getInt(string: "productPrice");
            int quantity = resultSet.getInt(string: "quantity");
            String description = resultSet.getString(string: "description");
            ProductCategory category = StringToProductCategoryConverter.convert(categoryString: resultSet.getString(string: "category"));
            String iconPath = resultSet.getString(string: "iconPath");
            int recordId = resultSet.getInt(string: "recordId");
            String email = resultSet.getString(string: "email");

            // Creating ShoppingCartItem object and adding it to the list
            ShoppingCartObj cartItem = new ShoppingCartObj(productName, productPrice, quantity, description,
                category, iconPath, recordId, email);
            currentCartSnap.add(e: cartItem);
        }
    } catch (Exception e) {
        System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
    } finally {
        // Closing the connection, statement, and result set
        try {
            if (resultSet != null) {
                resultSet.close();
            }
            if (statement != null) {
                statement.close();
            }
            if (connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
        }
    }
    return currentCartSnap;
}

public static void removeAllCartItems() {
    try (Connection connection = getConnection(); PreparedStatement statement = connection.prepareStatement(string: "DELETE FROM shoppingcart")) {
        statement.executeUpdate();
    } catch (SQLException e) {
        System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
    }
}

public static List<ShoppingCartObj> getOrderConfirmationDetails() {
    return getAllCartItems();
}

public static List<NewestProductObj> getNewArrivals() {
    List<NewestProductObj> newItems = new ArrayList<>();
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;
}

```

```

try{
    Class.forName(className: "com.mysql.cj.jdbc.Driver");

    // Establishing connection to the database
    connection = DriverManager.getConnection(url:URL, user:USERNAME, password: PASSWORD);

    // Creating SQL statement
    statement = connection.createStatement();

    // SQL query to fetch shopping cart items
    String query = "SELECT * FROM smartphone ORDER BY productId DESC";

    // Executing the query
    resultSet = statement.executeQuery(string: query);

    // Iterating over the result set and adding items to the shopping cart list
    while (resultSet.next()) {
        String productName = resultSet.getString(string: "productName");
        int productPrice = resultSet.getInt(string: "price");
        String iconPath = resultSet.getString(string: "photo1");

        // Creating ShoppingCartItem object and adding it to the list
        NewestProductObj items = new NewestProductObj(productName, productPrice, iconPath);
        newItems.add(e:items);
    }
} catch (Exception e){
    System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
} finally {
    // Closing the connection, statement, and result set
    try{
        if(resultSet!= null){
            resultSet.close();
        }
        if(statement != null){
            statement.close();
        }
        if(connection != null){
            connection.close();
        }
    } catch (SQLException e){
        System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
    }
}
return newItems;
}

public static List<AdminOrderObj> getAdminOrders() {
    List<AdminOrderObj> orders = new ArrayList<>();
    Connection connection = null;
    Statement statement = null;
    ResultSet resultSet = null;
    try{
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url:URL, user:USERNAME, password: PASSWORD);

        // Creating SQL statement
        statement = connection.createStatement();

        // SQL query to fetch from smartphone
        String query = "SELECT o.*," +
                      + "CASE" +
                      + " WHEN o.productId IN (SELECT productId FROM camera) THEN (SELECT productName FROM camera WHERE productId = o.productId)" +
                      + " WHEN o.productId IN (SELECT productId FROM smartphone) THEN (SELECT productName FROM smartphone WHERE productId = o.productId)" +
                      + " WHEN o.productId IN (SELECT productId FROM smartwatch) THEN (SELECT productName FROM smartwatch WHERE productId = o.productId)" +
                      + " WHEN o.productId IN (SELECT productId FROM laptop) THEN (SELECT productName FROM laptop WHERE productId = o.productId)" +
                      + " WHEN o.productId IN (SELECT productId FROM monitor) THEN (SELECT productName FROM monitor WHERE productId = o.productId)" +
                      + "END AS productName" +
                      + "FROM orders o";

        // Executing the query
        resultSet = statement.executeQuery(string: query);
    }
}

```

```

// Iterating over the result set and adding items from smartphone
while (resultSet.next()) {
    int orderNumber = resultSet.getInt("orderId");
    int productId = resultSet.getInt("productId");
    int quantity = resultSet.getInt("quantity");
    java.util.Date orderDate = resultSet.getDate("orderDate");
    String orderStatus = resultSet.getString("orderStatus");
    int totalPrice = resultSet.getInt("totalPrice");
    String shippingAddress = resultSet.getString("address");
    String customerName = resultSet.getString("name");
    String feedback = resultSet.getString("feedback");
    String email = resultSet.getString("email");
    String productName = resultSet.getString("productName"); // Retrieve product name

    // Creating Smartphone object and adding it to the list
    AdminOrderObj order = new AdminOrderObj(orderDate, orderNumber, orderStatus, productId, quantity,
        totalPrice, shippingAddress, customerName, feedback, email);
    order.setProductName(productName);
    orders.add(order);
}

} catch (Exception e) {
    System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
} finally {
    // Closing the connection, statement, and result set
    try {
        if (resultSet != null) {
            resultSet.close();
        }
        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.close();
        }
    } catch (SQLException e) {
        System.err.println("Error while retrieving shopping cart items: " + e.getMessage());
    }
}
return orders;
}

public static void removeOrderItem(int orderId) {
    try (Connection connection = getConnection(); PreparedStatement statement = connection.prepareStatement("DELETE FROM orders WHERE orderId = ?")) {
        statement.setInt(1, orderId);
        statement.executeUpdate();
    } catch (SQLException e) {
        System.err.println("Error while deleting order: " + e.getMessage());
    }
}

public static void updateSmartphone(String productId, int quantity, double price, String productName, String brand, String modelName, String productDescription, String storageCapacity, String screenSize,
    String color, String photo1, String photo2, String photo3, String photo4) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);

        // Prepare SQL statement
        String sql = "UPDATE smartphone SET quantity=?, price=?, productName=?, brand=?, modelName=?, productDescription=?, storageCapacity=?, screenSize=?, color=?, photo1=?, photo2=?, photo3=?, photo4=?"
            + "WHERE productId=?";
        statement = connection.prepareStatement(sql);
    }
}

```

```

// Set parameters
statement.setInt(1, int quantity);
statement.setDouble(2, double price);
statement.setString(3, string productName);
statement.setString(4, string brand);
statement.setString(5, string modelName);
statement.setString(6, string productDescription);
statement.setString(7, string storageCapacity);
statement.setString(8, string screenSize);
statement.setString(9, string color);
statement.setString(10, string photo1);
statement.setString(11, string photo2);
statement.setString(12, string photo3);
statement.setString(13, string photo4);
statement.setString(14, string productId);

// Execute update
statement.executeUpdate();

} catch (SQLException e) {
    Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: e);
} catch (ClassNotFoundException ex) {
    Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
} finally {
    // Close resources
    try {
        if(statement != null) {
            statement.close();
        }
        if(connection != null) {
            connection.close();
        }
    } catch (SQLException e) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: e);
    }
}
}

public static void deleteSmartphone(String productId) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        // Establishing connection to the database
        connection = getConnection();

        // Prepare SQL statement to delete the product
        String sql = "DELETE FROM smartphone WHERE productId = ?";
        statement = connection.prepareStatement(string: sql);

        // Set the productId parameter
        statement.setString(1, string: productId);

        // Execute the delete statement
        statement.executeUpdate();

    } catch (SQLException e) {
        System.err.println("Error while deleting product: " + e.getMessage());
    } finally {
        // Close resources
        try {
            if(statement != null) {
                statement.close();
            }
            if(connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            System.err.println("Error while closing connection: " + e.getMessage());
        }
    }
}

```

```

public static void deleteLaptop(String productId) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        // Establishing connection to the database
        connection = getConnection();

        // Prepare SQL statement to delete the product
        String sql = "DELETE FROM laptop WHERE productId = ?";
        statement = connection.prepareStatement(string: sql);

        // Set the productId parameter
        statement.setString(1, string: productId);

        // Execute the delete statement
        statement.executeUpdate();
    } catch (SQLException e) {
        System.err.println("Error while deleting product: " + e.getMessage());
    } finally {
        // Close resources
        try {
            if(statement != null) {
                statement.close();
            }
            if(connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            System.err.println("Error while closing connection: " + e.getMessage());
        }
    }
}

public static void addSmartphone(String productId, String productName, String category, int quantity, int price, String photo1, String photo2, String photo3, String photo4,
String brand, String modelName, String productDescription, String storageCapacity, String screenSize, String color) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        // Get connection
        connection = getConnection();

        // Create SQL query
        String query = "INSERT INTO smartphone (productId, productName, category, quantity, price, photo1, photo2, photo3, photo4, brand, modelName, productDescription, "
        + "storageCapacity, screenSize, color) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

        // Create PreparedStatement
        statement = connection.prepareStatement(string: query);

        // Set parameters
        statement.setString(1, string: productId);
        statement.setString(2, string: productName);
        statement.setString(3, string: category);
        statement.setInt(4, int: quantity);
        statement.setInt(5, int: price);
        statement.setString(6, string: photo1);
        statement.setString(7, string: photo2);
        statement.setString(8, string: photo3);
        statement.setString(9, string: photo4);
        statement.setString(10, string: brand);
        statement.setString(11, string: modelName);
        statement.setString(12, string: productDescription);
        statement.setString(13, string: storageCapacity);
        statement.setString(14, string: screenSize);
        statement.setString(15, string: color);

        // Execute the query
        statement.executeUpdate();
    } catch (SQLException e) {
        e.printStackTrace(); // Handle or log the exception appropriately
    } finally {
        // Close the connection and statement
        try {
            if(statement != null) {
                statement.close();
            }
            if(connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            e.printStackTrace(); // Handle or log the exception appropriately
        }
    }
}

```

```

public static void UpdateOrderStatus(int orderId, String orderStatus) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        // Establishing connection to the database
        connection = DriverManager.getConnection(url:URL, user:USERNAME, password: PASSWORD);

        // Prepare SQL statement
        String sql = "UPDATE orders SET orderStatus=? WHERE orderId=?";
        statement = connection.prepareStatement(string: sql);

        // Set parameters
        statement.setString(i: 1, string: orderStatus);
        statement.setInt(i: 2, int: orderId);

        // Execute update
        statement.executeUpdate();
    } catch (SQLException e) {
        // Log the exception with additional context information
        Logger.getLogger(name: DAO.class.getName()).log(level:Level.SEVERE, msg: "Error updating order status", thrown: e);
    } finally {
        // Close resources in reverse order of acquisition
        try {
            if(statement != null) {
                statement.close();
            }
            if(connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            // Log the exception with additional context information
            Logger.getLogger(name: DAO.class.getName()).log(level:Level.SEVERE, msg: "Error closing resources", thrown: e);
        }
    }
}

public static void updateLaptop(String productId, int quantity, double price, String productName, String brand, String modelName,
String productDescription, String storageCapacity, String cpu, String memory, String photo1, String photo2, String photo3, String photo4) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url:URL, user:USERNAME, password: PASSWORD);

        // Prepare SQL statement
        String sql = "UPDATE laptop SET quantity=?, price=?, productName=?, brand=?, modelName=?, productDescription=?, storageCapacity=?, cpu=?, memory=?, photo1=?, photo2=?, photo3=?, photo4=? WHERE productId=?";
        statement = connection.prepareStatement(string: sql);

        // Set parameters
        statement.setInt(i: 1, int: quantity);
        statement.setDouble(i: 2, double: price);
        statement.setString(i: 3, string: productName);
        statement.setString(i: 4, string: brand);
        statement.setString(i: 5, string: modelName);
        statement.setString(i: 6, string: productDescription);
        statement.setString(i: 7, string: storageCapacity);
        statement.setString(i: 8, string: cpu);
        statement.setString(i: 9, string: memory);
        statement.setString(i: 10, string: photo1);
        statement.setString(i: 11, string: photo2);
        statement.setString(i: 12, string: photo3);
        statement.setString(i: 13, string: photo4);
        statement.setString(i: 14, string: productId);

        // Execute update
        statement.executeUpdate();
    } catch (SQLException e) {
        Logger.getLogger(name: DAO.class.getName()).log(level:Level.SEVERE, msg: null, thrown: e);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(name: DAO.class.getName()).log(level:Level.SEVERE, msg: null, thrown: ex);
    }
}

```

```

} finally{
    // Close resources
    try{
        if(statement != null){
            statement.close();
        }
        if(connection != null){
            connection.close();
        }
    } catch (SQLException e) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, throws: e);
    }
}

public static void addLaptop(String productId, String productName, String category, int quantity, int price, String photo1,
String photo2, String photo3, String photo4, String brand, String modelName, String productDescription, String storageCapacity, String cpu, String memory) {
Connection connection = null;
PreparedStatement statement = null;

try{
    // Get connection
    connection = getConnection();

    // SQL query to insert laptop details into the database
    String query = "INSERT INTO laptop (productId, productName, category, quantity, price, photo1, photo2, photo3, photo4, brand, modelName, productDescription, storageCapacity, cpu, memory) " +
    "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

    // Create a prepared statement
    statement = connection.prepareStatement(string: query);

    // Set parameters for the query
    statement.setString(1, string: productId);
    statement.setString(2, string: productName);
    statement.setString(3, string: category);
    statement.setInt(4, int: quantity);
    statement.setInt(5, int: price);
    statement.setString(6, string: photo1);
    statement.setString(7, string: photo2);
    statement.setString(8, string: photo3);
    statement.setString(9, string: photo4);
    statement.setString(10, string: brand);
    statement.setString(11, string: modelName);
    statement.setString(12, string: productDescription);
    statement.setString(13, string: storageCapacity);
    statement.setString(14, string: cpu);
    statement.setString(15, string: memory);

    // Execute the query
    statement.executeUpdate();

} catch (SQLException e) {
    e.printStackTrace();
}
}

```

```

public static void updateCamera(String productId, int quantity, double price, String productName, String category, String photo1,
String photo2, String photo3, String photo4, String brand, String modelName, String productDescription, String formFactor) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);

        // Prepare SQL statement
        String sql = "UPDATE camera SET quantity=?, price=?, productName=?, category=?, photo1=?, photo2=?, photo3=?, photo4=?, brand=?, modelName=?, productDescription=?, formFactor=? WHERE productId=?";
        statement = connection.prepareStatement(string: sql);

        // Set parameters
        statement.setInt(1, int: quantity);
        statement.setDouble(2, double: price);
        statement.setString(3, string: productName);
        statement.setString(4, string: category);
        statement.setString(5, string: photo1);
        statement.setString(6, string: photo2);
        statement.setString(7, string: photo3);
        statement.setString(8, string: photo4);
        statement.setString(9, string: brand);
        statement.setString(10, string: modelName);
        statement.setString(11, string: productDescription);
        statement.setString(12, string: formFactor);
        statement.setString(13, string: productId);

        // Execute update
        statement.executeUpdate();
    } catch (SQLException e) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: e);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    } finally {
        // Close resources
        try {
            if(statement != null){
                statement.close();
            }
            if(connection != null){
                connection.close();
            }
        } catch (SQLException e) {
            Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: e);
        }
    }
}

public static void deleteCamera(String productId) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        // Establishing connection to the database
        connection = getConnection();

        // Prepare SQL statement to delete the product
        String sql = "DELETE FROM camera WHERE productId = ?";
        statement = connection.prepareStatement(string: sql);

        // Set the productId parameter
        statement.setString(1, string: productId);

        // Execute the delete statement
        statement.executeUpdate();
    } catch (SQLException e) {
        System.err.println("Error while deleting product: " + e.getMessage());
    } finally {
        // Close resources
        try {
            if(statement != null){
                statement.close();
            }
            if(connection != null){
                connection.close();
            }
        } catch (SQLException e) {
            System.err.println("Error while closing connection: " + e.getMessage());
        }
    }
}

```

```

public static void updateMonitor(String productId, int quantity, double price, String productName, String category, String photo1,
    String photo2, String photo3, String photo4, String brand, String modelName, String productDescription, String screenSize, String refreshRate, String resolution) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);

        // Prepare SQL statement
        String sql = "UPDATE monitor SET quantity=?, price=?, productName=?, category=?, photo1=?, photo2=?, photo3=?, photo4=?, brand=?, modelName=?," +
            "productDescription=?, screenSize=?, refreshRate=?, resolution=? WHERE productId=?";
        statement = connection.prepareStatement(string: sql);

        // Set parameters
        statement.setInt(1, id: quantity);
        statement.setDouble(2, id: price);
        statement.setString(3, string: productName);
        statement.setString(4, string: category);
        statement.setString(5, string: photo1);
        statement.setString(6, string: photo2);
        statement.setString(7, string: photo3);
        statement.setString(8, string: photo4);
        statement.setString(9, string: brand);
        statement.setString(10, string: modelName);
        statement.setString(11, string: productDescription);
        statement.setString(12, string: screenSize);
        statement.setString(13, string: refreshRate);
        statement.setString(14, string: resolution);
        statement.setString(15, string: productId);

        // Execute update
        statement.executeUpdate();
    } catch (SQLException e) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: e);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    } finally {
        // Close resources
        try {
            if(statement != null) {
                statement.close();
            }
            if(connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: e);
        }
    }
}

```

```

public static void addCamera(String productId, int quantity, double price, String productName, String category, String photo1,
    String photo2, String photo3, String photo4, String brand, String modelName, String productDescription, String formFactor) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);

        // Prepare SQL statement
        String sql = "INSERT INTO camera (productId, quantity, price, productName, category, photo1, photo2, photo3, photo4, brand, modelName, productDescription, formFactor) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        statement = connection.prepareStatement(string: sql);

        // Set parameters
        statement.setString(1, string: productId);
        statement.setInt(2, int: quantity);
        statement.setDouble(3, double: price);
        statement.setString(4, string: productName);
        statement.setString(5, string: category);
        statement.setString(6, string: photo1);
        statement.setString(7, string: photo2);
        statement.setString(8, string: photo3);
        statement.setString(9, string: photo4);
        statement.setString(10, string: brand);
        statement.setString(11, string: modelName);
        statement.setString(12, string: productDescription);
        statement.setString(13, string: formFactor);

        // Execute update
        statement.executeUpdate();
    } catch (SQLException e) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: e);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    }
}

public static void addMonitor(String productId, int quantity, double price, String productName, String category, String photo1, String photo2,
    String photo3, String photo4, String brand, String modelName, String productDescription, String screenSize, String refreshRate, String resolution) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);

        // Create SQL query
        String query = "INSERT INTO monitor (productId, quantity, price, productName, category, photo1, photo2, photo3, photo4, brand, "
            + "modelName, productDescription, screenSize, refreshRate, resolution) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

        // Create PreparedStatement
        statement = connection.prepareStatement(string: query);

        // Set parameters
        statement.setString(1, string: productId);
        statement.setInt(2, int: quantity);
        statement.setDouble(3, double: price);
        statement.setString(4, string: productName);
        statement.setString(5, string: category);
        statement.setString(6, string: photo1);
        statement.setString(7, string: photo2);
        statement.setString(8, string: photo3);
        statement.setString(9, string: photo4);
        statement.setString(10, string: brand);
        statement.setString(11, string: modelName);
        statement.setString(12, string: productDescription);
        statement.setString(13, string: screenSize);
        statement.setString(14, string: refreshRate);
        statement.setString(15, string: resolution);
    }
}

```

```

        // Execute update
        statement.executeUpdate();
    } catch (SQLException e) {
        Logger.getLogger(name: DAO.class.getName()).log(level:Level.SEVERE, msg: null, thrown: e);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(name: DAO.class.getName()).log(level:Level.SEVERE, msg: null, thrown: ex);
    }
}

public static void deleteMonitor(String productId) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        // Establishing connection to the database
        connection = getConnection();

        // Prepare SQL statement to delete the product
        String sql = "DELETE FROM monitor WHERE productId = ?";
        statement = connection.prepareStatement(string: sql);

        // Set the productId parameter
        statement.setString(i: 1, string: productId);

        // Execute the delete statement
        statement.executeUpdate();
    } catch (SQLException e) {
        System.err.println("Error while deleting product: " + e.getMessage());
    } finally {
        // Close resources
        try {
            if(statement != null){
                statement.close();
            }
            if(connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            System.err.println("Error while closing connection: " + e.getMessage());
        }
    }
}

```

```

public static void updateSmartwatch(String productId, int quantity, double price, String productName, String category, String photo1,
String photo2, String photo3, String photo4, String brand, String modelName, String productDescription, String screenSize, String color) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);

        // Prepare SQL statement
        String sql = "UPDATE smartwatch SET quantity=?, price=?, productName=?, category=?, photo1=?, photo2=?, photo3=?, photo4=?, brand=?," +
                    + modelName=?, productDescription=?, screenSize=?, color=? WHERE productId=?";
        statement = connection.prepareStatement(string: sql);

        // Set parameters
        statement.setInt(1, quantity);
        statement.setDouble(2, price);
        statement.setString(3, productName);
        statement.setString(4, category);
        statement.setString(5, photo1);
        statement.setString(6, photo2);
        statement.setString(7, photo3);
        statement.setString(8, photo4);
        statement.setString(9, brand);
        statement.setString(10, modelName);
        statement.setString(11, productDescription);
        statement.setString(12, screenSize);
        statement.setString(13, color);
        statement.setString(14, productId);

        // Execute update
        statement.executeUpdate();
    } catch (SQLException e) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: e);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    } finally {
        // Close resources
        try {
            if(statement != null) {
                statement.close();
            }
            if(connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: e);
        }
    }
}

public static void deleteSmartwatch(String productId) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        // Establishing connection to the database
        connection = getConnection();

        // Prepare SQL statement to delete the product
        String sql = "DELETE FROM smartwatch WHERE productId = ?";
        statement = connection.prepareStatement(string: sql);

        // Set the productId parameter
        statement.setString(1, productId);

        // Execute the delete statement
        statement.executeUpdate();
    } catch (SQLException e) {
        System.err.println("Error while deleting product: " + e.getMessage());
    } finally {
        // Close resources
        try {
            if(statement != null) {
                statement.close();
            }
            if(connection != null) {
                connection.close();
            }
        } catch (SQLException e) {
            System.err.println("Error while closing connection: " + e.getMessage());
        }
    }
}

```

```

public static void addSmartwatch(String productId, int pieces, double price, String productName, String category, String photo1,
    String photo2, String photo3, String photo4, String brand, String modelName, String productDescription, String screenSize, String color) {
    Connection connection = null;
    PreparedStatement statement = null;

    try {
        Class.forName(className: "com.mysql.cj.jdbc.Driver");

        // Establishing connection to the database
        connection = DriverManager.getConnection(url: URL, user: USERNAME, password: PASSWORD);

        // Prepare SQL statement
        String sql = "INSERT INTO smartwatch (productId, quantity, price, productName, category, photo1, photo2, photo3, photo4, brand, "
            + |modelName, productDescription, screenSize, color| VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";

        statement = connection.prepareStatement(string: sql);

        // Set parameters
        statement.setString(1, string: productId);
        statement.setInt(2, int: pieces);
        statement.setDouble(3, double: price);
        statement.setString(4, string: productName);
        statement.setString(5, string: category);
        statement.setString(6, string: photo1);
        statement.setString(7, string: photo2);
        statement.setString(8, string: photo3);
        statement.setString(9, string: photo4);
        statement.setString(10, string: brand);
        statement.setString(11, string: modelName);
        statement.setString(12, string: productDescription);
        statement.setString(13, string: screenSize);
        statement.setString(14, string: color);

        // Execute insert
        statement.executeUpdate();

    } catch (SQLException e) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: e);
    } catch (ClassNotFoundException ex) {
        Logger.getLogger(name: DAO.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    }
}
}

```

ClearCart.java

```

package Controller;

import Model.DAO;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

/*
 * @author robin
 */
@WebServlet(name = "ClearCart", urlPatterns = {"/*ClearCart"})
public class ClearCart extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        //processRequest(request, response);
        DAO.removeAllCartItems();
        request.getRequestDispatcher(string: "ShoppingCart.jsp").forward(er: request, sr1: response);
    }
}

```

Navbar



Navbar.java

```
<nav id="topNav" class="collapse">
    <div class="nav-content">
        
        <div class="nav-links">
            <a href="index.jsp" class="nav-link home-link">Home</a>
            <a href="ProductCategory_Cameras.jsp" class="nav-link products-link">Products</a>
            <a href="AboutUs.jsp" class="nav-link aboutUs-link">About Us</a>
            <a href="Contact.jsp" class="nav-link support-link">Support</a>
        </div>
    </div>
    <div class="nav-actions">
        <div class="nav-action-buttons">
            <%
                String loggedInUser = (String) session.getAttribute("email");
                if (loggedInUser != null) { // User is logged in
            %>
                <a href="Profile.jsp" class="nav-button">
                    
                </a>
            <% } else { // User is not logged in %>
                <a href="SignUp.jsp" class="nav-button sign">
                    <span class="indicator">SignUp</span>
                    
                </a>
            <% } %>
                <a href="ShoppingCart.jsp" class="nav-button">
                    
                </a>
            </div>
        </div>
        <div class="hamburger" onclick="expandNavBar();">
            <div class="line"></div>
            <div class="line"></div>
            <div class="line"></div>
        </div>
    </div>
</nav>
```

Navbar.css

```
<nav id="topNav" class="collapse">
  <div class="nav-content">
    
    <div class="nav-links">
      <a href="index.jsp" class="nav-link home-link">Home</a>
      <a href="ProductCategory_Cameras.jsp" class="nav-link products-link">Products</a>
      <a href="AboutUs.jsp" class="nav-link aboutUs-link">About Us</a>
      <a href="ContactJsp" class="nav-link support-link">Support</a>
    </div>
  </div>
  <div class="nav-actions">
    <div class="nav-action-buttons">
      <%>
        String loggedInUser = (String) session.getAttribute("email");
        if (loggedInUser != null) { // User is logged in
      %>
        <a href="Profile.jsp" class="nav-button">
          
        </a>
      <% } else { // User is not logged in %>
        <a href="SignUp.jsp" class="nav-button sign">
          <span class="indicator">Sign Up</span>
          
        </a>
      <% } %>
        <a href="ShoppingCart.jsp" class="nav-button">
          
        </a>
      </div>
    </div>
    <div class="hamburger" onclick="expandNavBar();">
      <div class="line"></div>
      <div class="line"></div>
      <div class="line"></div>
    </div>
  </nav>
  nav {
    display: flex;
    flex-direction: row;
    justify-content: space-between;
    align-items: center;
    padding: 1rem;
    transition: all 2s;
    max-height: 100vh;
  }
  .nav-content {
    display: flex;
    align-items: center;
    gap: 3rem;
  }
  .nav-brand {
  }
  .nav-links {
    display: flex;
    gap: 1.8rem;
  }
  .nav-link {
    text-decoration: none;
    color: white;
    opacity: 0.6;
    font-size: 16px;
    letter-spacing: 0.08em;
  }
```

```
.nav-actions {
  display: flex;
  align-items: center;
  gap: 2.5rem;
}

.input-area {
  display: flex;
  flex-direction: row;
  align-items: center;
  gap: 0.7rem;

  background-color: black;
  padding: 3px 17px;
  border-radius: 7px;
  border: 1px #eeeeee6c solid;
  min-width: 250px;
  transition: all 0.6s;
}

.input-area:hover{
  border-color: white;
}

.text-input-area:focus {
  border-color: transparent; /* Set the border color to transparent when focused */
  outline: none; /* Remove the outline that appears by default */
  background-color: black;
}

.input-area-icon {
}

.text-input-area {
  padding: 7px;
  background-color: black;
  border: 0;
  color: white;
}

.nav-action-buttons {
  display: flex;
  gap: 1rem;
}

.nav-button-icon {
}

.nav-button {
  height: 34px;
  width: 34px;
  padding: 5px;
  border-radius: 7px;
}

.hamburger{
  background-color: rgb(0, 0, 0);
  min-width: 40px;
  min-height: 40px;
  position: absolute;
  top: 0;
  right: 0;
  margin: 1.5rem;

  align-items: center;
  flex-direction: column;
  gap: 6px;
  justify-content: center;
  border-radius: 7px;
  opacity: 0.7;
  border: 2px solid transparent;
  transition: all 0.6s;
  display: none;
}

.line{
  display: block;
  background-color: rgb(255, 255, 255);
  min-height: 4px;
  border-radius: 2px;
  min-width: 27px;
}

.hamburger:active{
  border: 2px solid white;
}
```

```
@media screen and (max-width: 1072px) {  
  nav {  
    flex-direction: column;  
    gap: 2rem;  
    justify-content: flex-start;  
    align-items: flex-start;  
    overflow: hidden;  
  }  
  
  .nav-content {  
    flex-direction: column;  
    align-items: flex-start;  
  }  
  
  .nav-links {  
    flex-direction: column;  
    overflow: hidden;  
  }  
  
  .nav-actions {  
    overflow: hidden;  
    gap: 1rem;  
  }  
  
  .nav-action-buttons {  
    gap: 0.5rem;  
  }  
  
  .input-area {  
    min-width: 0;  
    width: 45%;  
  }  
  
  .hamburger {  
    display: flex;  
  }  
}
```

```
.collapse {  
  height: 50px;  
  max-height: 50px;  
  overflow: hidden;  
}  
  
.nav-button {  
  display: flex;  
  align-items: center; /* Align items vertically */  
}  
  
.nav-action-buttons {  
  display: flex;  
  gap: 2rem;  
  align-items: center; /* Align items vertically */  
}  
  
.indicator {  
  font-size: 18px;  
  color: white;  
}
```

Footer

 **E-MART**
ELECTRONICS

From cutting-edge processors to AI-powered technologies, E-Mart brings you the latest innovations in electronics. Elevate your experience and embrace the future with E-Mart.

[!\[\]\(e2723e6865291fbe505a7739c93fa2bf_img.jpg\)](#) [!\[\]\(491e9ba1d8a135f46c4e4cbdbdf9a113_img.jpg\)](#) [!\[\]\(3c5b221eae5b2d1113a1e702fb38075f_img.jpg\)](#) [!\[\]\(56b6ffda46ac5be8a94e2a69fecbe6e1_img.jpg\)](#) [!\[\]\(4b0c2cd7afca65275c2de19ecf626f42_img.jpg\)](#)

Â© 2024 Electronics Mart, Inc

[Privacy Policy](#) [Terms of Use](#)

Footer.html

```
<footer>
  <div class="footer-content">
    <div class="footer-details">
      
      <p class="footer-description">From cutting-edge processors to AI-powered technologies, E-Mart brings you the latest innovations in electronics. Elevate your experience and embrace the future with E-Mart.</p>
      <div class="footer-social-links">
        <a href="https://web.facebook.com/?_rdr=1&_rdr">
          
        </a>
        <a href="https://twitter.com/lang_ap">
          
        </a>
        <a href="https://www.instagram.com/2hl=en">
          
        </a>
        <a href="https://www.linkedin.com/">
          
        </a>
        <a href="https://www.youtube.com/">
          
        </a>
      </div>
    </div>
    <div class="footer-spacer"></div>
    <div class="footer-about-links">
      <h4 class="footer-heading">Company</h4>
      <a href="AboutUs.jsp" class="footer-link">About E-Mart</a>
      <a href="Developers.jsp" class="footer-link">Developers</a>
    </div>
    <div class="footer-spacer"></div>
    <div class="footer-about-links">
      <h4 class="footer-heading">Company</h4>
      <a href="AboutUs.jsp" class="footer-link">About E-Mart</a>
      <a href="Developers.jsp" class="footer-link">Developers</a>
    </div>
    <div class="footer-page-links">
      <h4 class="footer-heading">My Mart</h4>
      <a href="Profile.jsp" class="footer-link">My Profile</a>
      <a href="TrackOrder.jsp" class="footer-link">Ongoing Orders</a>
      <a href="OrderHistory.jsp" class="footer-link">Order History</a>
    </div>
    <div class="footer-contact-links">
      <h4 class="footer-heading">Contact Us</h4>
      <a href="https://web.whatsapp.com/" class="footer-link footer-contact-link">
        
        <span>+94 77 696 9696</span>
      </a>
      <a href="https://mail.google.com/" class="footer-link footer-contact-link">
        
        <span>mail.emart@gmail.com</span>
      </a>
    </div>
    <div class="separator"></div>
    <div class="footer-extra">
      <span class="footer-extra-text">© 2024 Electronics Mart, Inc </span>
    </div>
    <div class="footer-extra-links">
      <a href="AboutUs.jsp" class="footer-extra-link">Privacy Policy</a>
      <a href="AboutUs.jsp" class="footer-extra-link">Terms of Use</a>
    </div>
  </div>
</footer>
```

Footer.css

```
footer {
  padding: 2rem;
  background: linear-gradient(0deg, rgba(255, 255, 255, 0.07), rgba(255, 255, 255, 0.07)), linear-gradient(209.08deg, rgba(0, 0, 0.42) 17.86%, #000000 82.12%), linear-gradient(93.07deg, rgba(0, 0, 0.24) 0%, rgba(0, 0, 0.24) 100%);
}

.footer-content {
  display: flex;
  flex-direction: row;
  justify-content: stretch;
  align-items: stretch;
  gap: 4rem;
}

.separator {
  border-top: 2px solid rgb(0, 68, 255);
  margin-top: 2.5rem;
  margin-bottom: 1rem;
}

.footer-extra {
  min-height: 10px;
}

.footer-details {
  flex-grow: 1;
  max-width: 450px;
  letter-spacing: 0.08em;
  display: flex;
  flex-direction: column;
  gap: 0.7rem;
  align-items: flex-start;
  justify-content: flex-start;
}

.footer-description {
  opacity: 0.7;
}

.footer-social-links {
  display: flex;
  gap: 1rem;
}

.social-icon-link {
  height: 28px;
}

.footer-spacer {
  min-width: 50px;
  min-width: 50px;
  flex-grow: 1;
}

.footer-about-links,
.footer-page-links,
.footer-contact-links {
  display: flex;
  flex-direction: column;
  gap: 0.8rem;
}

.footer-link {
  text-decoration: none;
  color: white;
  opacity: 0.5;
  letter-spacing: 0.08em;
  transition: all 0.6s;
}

.footer-link:hover {
  opacity: 1;
}

.footer-contact-link {
  display: grid;
  grid-template-columns: 28px auto;
  gap: 0.5rem;
  align-items: center;
}

.footer-contact-icon {
  width: 22px;
}

.footer-extra {
  display: flex;
  justify-content: space-between;
}

.footer-extra-links {
  display: flex;
  gap: 1rem;
}

.footer-extra-link {
  text-decoration: none;
  color: white;
  opacity: 0.5;
  transition: all 0.6s;
}
```

```
□ .footer-extra-link:hover{  
    opacity: 1;  
}  
  
□ @media screen and (max-width: 600px) {  
    □ .footer-content {  
        flex-direction: column;  
    }  
  
    □ .footer-extra {  
        flex-direction: column;  
    }  
  
    □ .footer-extra-links {  
        margin-top: 2rem;  
        flex-direction: column;  
    }  
}
```

2. Admin Panel

- To access the Admin page. Type /AdminPanel in the URL followed by the website host URL.
- Enter password "emart1234" when asked for login details. Use a preferable email of your choice.

2.1 Products page (index page)

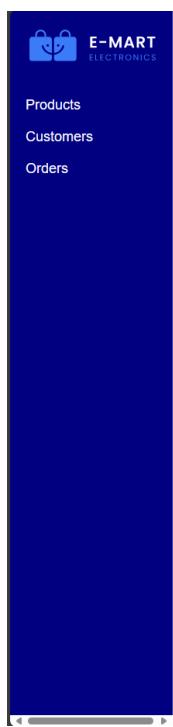
Done by : E. L. Thambawita

Student ID : 29186

2.1.1 Web page

The screenshot shows the E-Mart Admin Panel interface. On the left, there is a sidebar with icons for Products, Customers, and Orders. The main content area has a header 'Smartphones' with tabs for SMARTPHONES, LAPTOPS, CAMERAS, MONITORS, and SMARTWATCHES. Below the header is a table with columns: Product ID, Product Name, Category, Pieces, Price, Photo 1, Photo 2, Photo 3, Photo 4, Brand, Model Name, Product Description, Storage Capacity, and Screen Size. Three rows of smartphone data are listed: 401 (Smartphone Z), 404 (Smartphone A), and 405 (Smartphone W). At the bottom of the table is a button labeled 'Add More Products'.

Product ID	Product Name	Category	Pieces	Price	Photo 1	Photo 2	Photo 3	Photo 4	Brand	Model Name	Product Description	Storage Capacity	Screen Size
401	Smartphone Z	smartphone	56	324000	Images/Home/S22.png	Images/Home/S21.png	Images/Home/S21.png	Images/Home/S21.png	Brand u	Model A	Description A	64GB	6.2
404	Smartphone A	smartphone	56	560000	Images/Home/S22.png	Images/Home/S22.png	Images/Home/S22.png	Images/Home/S22.png	Brand X	smartphone	you are weird	456	6.5
405	Smartphone W	smartphone	56	123000	Images/Home/S22.png	Images/Home/S22.png	Images/Home/S22.png	Images/Home/S22.png	Brand X	smartphone	your big pool	456	6.7



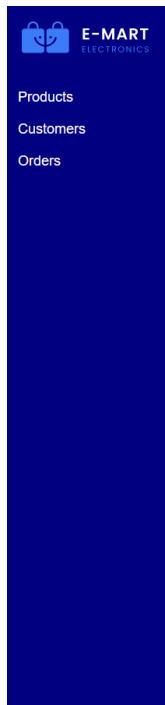
Cameras

SMARTPHONES LAPTOPS CAMERAS MONITORS SMARTWATCHES

Product ID	Product Name	Category	Pieces	Price	Photo 1	Photo 2	Photo 3	Photo 4	Brand	Model Name	Product Description	Form Factor	Action
101	Compact Camera A	camera	30	49999	Images/Home/S22.png	null	null	null	PixelPerfect	CamA-100	Capture every moment with the Compact Camera A.	Point-and-Shoot	
102	DSLR Camera X	camera	20	89999	Images/Home/S22.png	null	null	null	ProShot	DSLRX-2000	Unleash your creativity with the DSLR Camera X.	SLR	
103	Mirrorless Camera Z	camera	10	129999	Images/Home/S22.png	null	null	null	InnoVision	MirrorZ-500	Experience the future of photography with the Mirrorless Camera Z.	Compact System	

[Add More Products](#)

Add products form:



Smartphones

Add Smartphone

Product ID:	21.png	Brand u	Model A	Product Description	Storage Capacity	\$
401	Smartp Z			Description A	64GB	6.
404	Smartp A	22.png	Brand X	smartphone	you are weird	456
405	Smartp W	22.png	Brand X	smartphone	your big pool	456

Product ID:

Product Name:

Category:

Pieces:

Price:

Photo 1:

Photo 2:

Photo 3:

Photo 4:

Brand:

Edit product info - form :

Monitors																	
	SMARTPHONES			LAPTOPS		CAMERAS		MONITORS			SMARTWATCHES						
	Product Name	Category	Pieces	Price	Photo 1			Photo 2	Photo 3	Photo 4	Brand	Model Name	Product Description	Screen Size	Refresh Rate	Resolution	Action
	HD Monitor 24"	monitor	50	14999	Images\Product\Images\Monitors\AsusProArtDisplayPA278CV.png			null	null	null	TechCo	HD-2400	Experience stunning visuals with the HD Monitor 24".	24 inches	60	1920x1080	
	Gaming Monitor 27"	monitor	30	29999	Images\Product\Images\Monitors\AsusProArtDisplayPA278CV.png			null	null	null	GamingGear	GamerX-270	Immerse yourself in the world of gaming with the Gaming Monitor 27".	27 inches	144	2560x1440	
	UltraWide Monitor 34"	monitor	20	49999	Images\Product\Images\Monitors\AsusProArtDisplayPA278CV.png			null	null	null	UltraTech	UltraView-3400	Enhance your productivity with the UltraWide Monitor 34".				

Add Products

Edit Monitor

Quantity
50

Price
14999

Product Name
HD Monitor 24"

Brand
TechCo

Model Name
HD-2400

2.1.2 Documentation

Overview

The Products page in the admin panel provides functionality to manage various product categories such as smartphones, laptops, cameras, monitors, and smartwatches. Each category has its own separate page accessible through a horizontal menu. The page allows administrators to view, add, edit, and delete products within each category.

Common Features

Horizontal Menu: The horizontal menu at the top of the page allows you to easily navigate between product categories.

Table Display: Products are displayed in a tabular format, with detailed information such as Product ID, Name, Quantity, Price, Brand, Model Name, Description, Storage Capacity, Screen Size, Color, and Edit and Delete buttons. These differ according to the category page.

Edit and Delete: Each product entry in the table includes buttons for editing and deleting that product. The edit button opens a form for modifying product details, whereas the delete button prompts a confirmation before removing the product from the database.

Add Products: The "Add More Products" button displays a modal form in which administrators can add new products to the database. This form has fields for entering product information such as Product ID, Name,

Category, Quantity, Price, Brand, Model Name, Description, Storage Capacity, Screen Size, Color, and Photo URLs.

File Structure

Java Servlet:

AdminEditProductServlet.java: Edits product details. It accepts POST requests from client-side form submissions, processes the form data, and updates the product entry in the database accordingly.

AdminDeleteProductServlet.java: Handles product deletions. When triggered by a POST request from the client, it deletes the specified product from the database.

AdminAddProductServlet.java: Handles the addition of new products to the database. It handles POST requests containing form data, validates the input, and adds a new product entry to the database.

JavaScript file:

This JavaScript file, delete.js, contains functions for deleting products. It allows you to display a confirmation dialog before performing the deletion operation.

CSS files:

horizontalMenu.css: A stylesheet designed specifically for the horizontal menu at the top of the page. It specifies the layout, appearance, and behavior of menu items.

editForm.css: Contains styles for the popup form used to edit product details. It specifies the arrangement of input fields, buttons, and the overall appearance of the form.

AddProductsForm.css: This is the stylesheet for the modal form used to add new products. It determines the layout and appearance of input fields, buttons, and other form elements.

JavaServer Pages (JSP)

products.jsp: The primary JSP file that generates dynamic HTML content for displaying product information and forms. It contains logic for retrieving product data from the database and displaying it in a user-friendly format.
Other files:

Other files

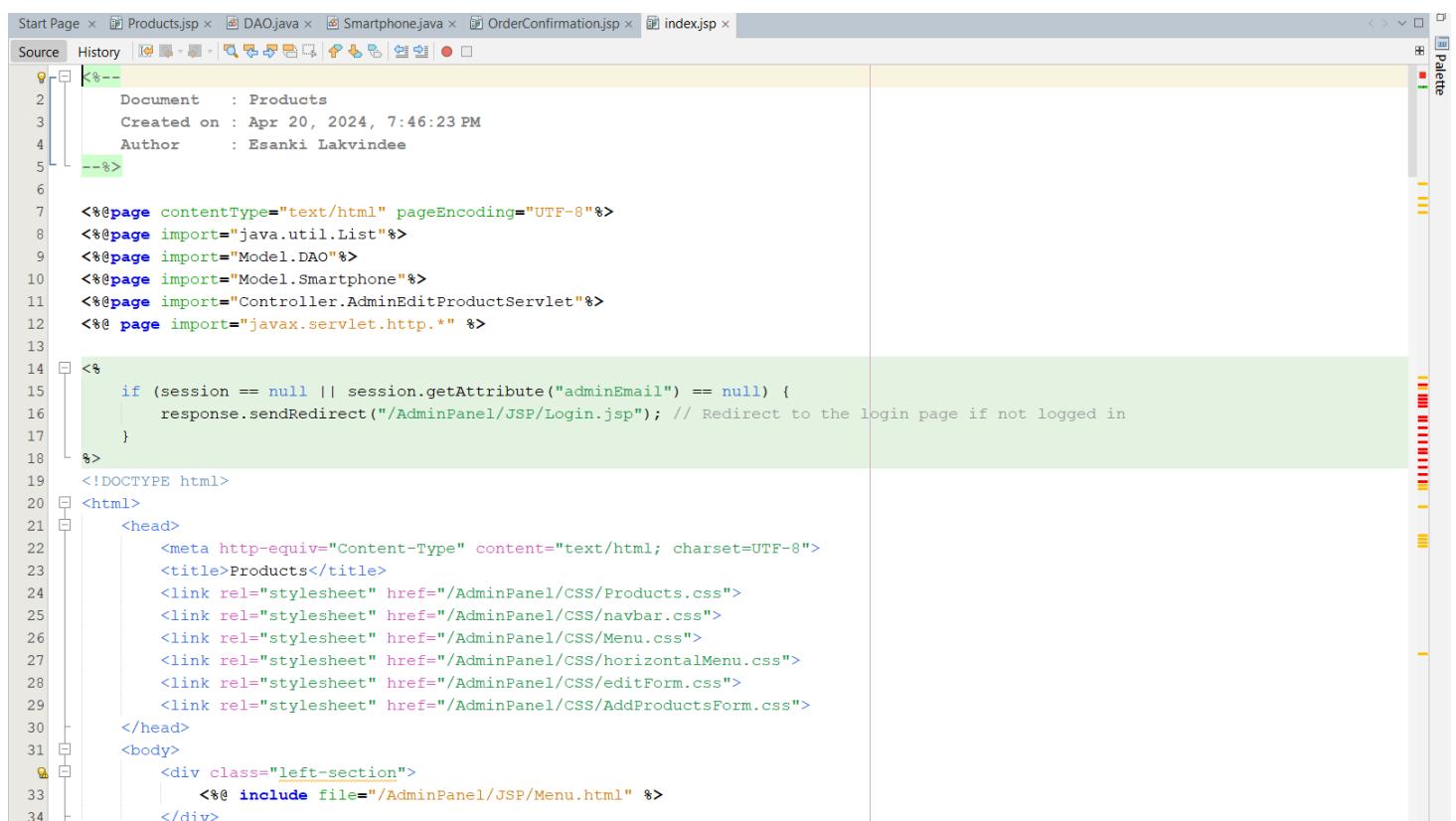
DAO.java: This file includes database-related methods like updateSmartphone(), deleteSmartphone(), and addSmartphone(). These methods perform database operations such as updating, deleting, and adding smartphone products.

Conclusion

The Products page in the admin panel has a file structure that includes Java Servlets, JavaScript files, CSS files, JavaServer Pages, and other supporting files. Each component is essential for managing product information, handling user interactions, and providing a consistent user experience. These files work together to create a cohesive system for managing and storing product data within the application.

2.1.3 Codes

Index.jsp



```
<%--  
1 Document      : Products  
2 Created on   : Apr 20, 2024, 7:46:23 PM  
3 Author        : Esanki Lakvindee  
4 --%>  
5  
6  
7 <%@page contentType="text/html" pageEncoding="UTF-8"%>  
8 <%@page import="java.util.List"%>  
9 <%@page import="Model.DAO"%>  
10 <%@page import="Model.Smartphone"%>  
11 <%@page import="Controller.AdminEditProductServlet"%>  
12 <%@ page import="javax.servlet.http.*" %>  
13  
14 <%  
15     if (session == null || session.getAttribute("adminEmail") == null) {  
16         response.sendRedirect("/AdminPanel/JSP/Login.jsp"); // Redirect to the login page if not logged in  
17     }  
18 <%>  
19 <!DOCTYPE html>  
20 <html>  
21     <head>  
22         <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
23         <title>Products</title>  
24         <link rel="stylesheet" href="/AdminPanel/CSS/Products.css">  
25         <link rel="stylesheet" href="/AdminPanel/CSS/navbar.css">  
26         <link rel="stylesheet" href="/AdminPanel/CSS/Menu.css">  
27         <link rel="stylesheet" href="/AdminPanel/CSS/horizontalMenu.css">  
28         <link rel="stylesheet" href="/AdminPanel/CSS/editForm.css">  
29         <link rel="stylesheet" href="/AdminPanel/CSS/AddProductsForm.css">  
30     </head>  
31     <body>  
32         <div class="left-section">  
33             <%@ include file="/AdminPanel/JSP/Menu.html" %>  
34         </div>
```

```

37
38
39     <div class="right-section">
40
41         <div class="container">
42             <h1>Smartphones</h1>
43             <ul class="horizontal-menu">
44                 <li><a href="/AdminPanel/JSP/Products.jsp">Smartphones</a></li>
45                 <li><a href="/AdminPanel/JSP/Laptop.jsp">Laptops</a></li>
46                 <li><a href="/AdminPanel/JSP/Camera.jsp">Cameras</a></li>
47                 <li><a href="/AdminPanel/JSP/Monitor.jsp">Monitors</a></li>
48                 <li><a href="/AdminPanel/JSP/Smartwatch.jsp">Smartwatches</a></li>
49             </ul>
50
51             <table id="product-table">
52                 <thead>
53                     <tr>
54                         <th>Product ID</th>
55                         <th>Product Name</th>
56                         <th>Category</th>
57                         <th>Pieces</th>
58                         <th>Price</th>
59                         <th>Photo 1</th>
60                         <th>Photo 2</th>
61                         <th>Photo 3</th>
62                         <th>Photo 4</th>
63                         <th>Brand</th>
64                         <th>Model Name</th>
65                         <th>Product Description</th>
66                         <th>Storage Capacity</th>
67                         <th>Screen Size</th>
68                         <th>Color</th>
69                         <th>Action</th>
70                 </tr>
71             </thead>
72             <tbody>
73                 <%
74                     List<Smartphone> products = DAO.getAllSmartphones();
75                     for (Smartphone item : products) {
76                         %>
77                         <tr>
78                             <td><%= item.getProductId() %></td>
79                             <td><%= item.getProductName() %></td>
80                             <td><%= item.getCategory() %></td>
81                             <td><%= item.getQuantity() %></td>
82                             <td><%= item.getPrice() %></td>
83                             <td><%= item.getPhoto1() %></td>
84                             <td><%= item.getPhoto2() %></td>
85                             <td><%= item.getPhoto3() %></td>
86                             <td><%= item.getPhoto4() %></td>
87                             <td><%= item.getBrand() %></td>
88                             <td><%= item.getModelName() %></td>
89                             <td><%= item.getProductDescription() %></td>
90                             <td><%= item.getStorageCapacity() %></td>
91                             <td><%= item.getScreenSize() %></td>
92                             <td><%= item.getColor() %></td>
93                             <td>
94                                 <button class="button1" onclick="openForm('<%= item.getProductId() %>')">
95                                     
96                                 </button>
97                                 <div class="form-popup" id="myForm_<%= item.getProductId() %>">
98                                     <form action="/AdminEditProductServlet" class="form-container" method="post">
99                                         <h1>Edit Smartphone</h1>
100                                         <label for="pieces_<%= item.getProductId() %>"><b>Quantity</b></label>
101                                         <input type="number" placeholder="Enter quantity" name="quantity" value="<%= item.getQuantity() %>">
102                                         <label for="price_<%= item.getProductId() %>"><b>Price</b></label>
103                                         <input type="number" step="0.01" placeholder="Enter Price" name="price" value="<%= item.getPrice() %>">
104                                         <input type="hidden" name="productId" value="<%= item.getProductId() %>">

```

```

102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167

```

--

```

<input type="hidden" name="category" value="<% item.getCategory() %>">
<label for="productName_<%= item.getProductId() %>">Product Name</b></label>
<input type="text" placeholder="Enter product name" name="productName" value="<% item.getProd
<label for="brand_<%= item.getProductId() %>">Brand</b></label>
<input type="text" placeholder="Enter brand" name="brand" value="<% item.getBrand() %>" requir
<label for="modelName_<%= item.getProductId() %>">Model Name</b></label>
<input type="text" placeholder="Enter model name" name="modelName" value="<% item.getModelName
<label for="productDescription_<%= item.getProductId() %>">Product Description</b></label>
<textarea placeholder="Enter product description" name="productDescription" required><%= item.
<label for="storageCapacity_<%= item.getProductId() %>">Storage Capacity</b></label>
<input type="text" placeholder="Enter storage capacity" name="storageCapacity" value="<% item.
<label for="screenSize_<%= item.getProductId() %>">Screen Size</b></label>
<input type="text" placeholder="Enter screen size" name="screenSize" value="<% item.getScreens
<label for="color_<%= item.getProductId() %>">Color</b></label>
<input type="text" placeholder="Enter color" name="color" value="<% item.getColor() %>" require
<label for="photo1_<%= item.getProductId() %>">Photo 1</b></label>
<input type="text" placeholder="Enter URL for photo 1" name="photo1" value="<% item.getPhoto1
<label for="photo2_<%= item.getProductId() %>">Photo 2</b></label>
<input type="text" placeholder="Enter URL for photo 2" name="photo2" value="<% item.getPhoto2
<label for="photo3_<%= item.getProductId() %>">Photo 3</b></label>
<input type="text" placeholder="Enter URL for photo 3" name="photo3" value="<% item.getPhoto3
<label for="photo4_<%= item.getProductId() %>">Photo 4</b></label>
<input type="text" placeholder="Enter URL for photo 4" name="photo4" value="<% item.getPhoto4
<button type="submit" class="btn">Save Changes</button>
<button type="button" class="btn cancel" onclick="closeForm('<%= item.getProductId() %>')>Close
</form>
</div>

<button class="button2" onclick="deleteProduct('<%= item.getProductId() %>')>
    
</button>
</td>
</tr>
<% } %>
</tbody>
</table>

<button class="add-products-btn" onclick="opentheForm()>Add More Products</button>
<div id="addProductModal" class="modal" style="display: none;">
    <div class="modal-content">
        <span class="close" onclick="closetheForm()>&times;</span>
        <h2>Add Smartphone</h2>
        <form id="addProductForm" method="POST" action="/AdminAddProductServlet">
            <label for="productId">Product ID:</label>
            <input type="text" id="productId" name="productId" required><br>
            <label for="productName">Product Name:</label>
            <input type="text" id="productName" name="productName" required><br>
            <label for="category">Category:</label>
            <input type="text" id="category" name="category" required><br>
            <label for="pieces">Pieces:</label>
            <input type="number" id="pieces" name="pieces" required><br>
            <label for="price">Price:</label>
            <input type="number" id="price" name="price" required><br>
            <label for="photo1">Photo 1:</label>
            <input type="text" id="photo1" name="photo1"><br>
            <label for="photo2">Photo 2:</label>
            <input type="text" id="photo2" name="photo2"><br>
            <label for="photo3">Photo 3:</label>
            <input type="text" id="photo3" name="photo3"><br>
            <label for="photo4">Photo 4:</label>
            <input type="text" id="photo4" name="photo4"><br>
            <label for="brand">Brand:</label>
            <input type="text" id="brand" name="brand" required><br>
            <label for="modelName">Model Name:</label>
            <input type="text" id="modelName" name="modelName" required><br>

```

```

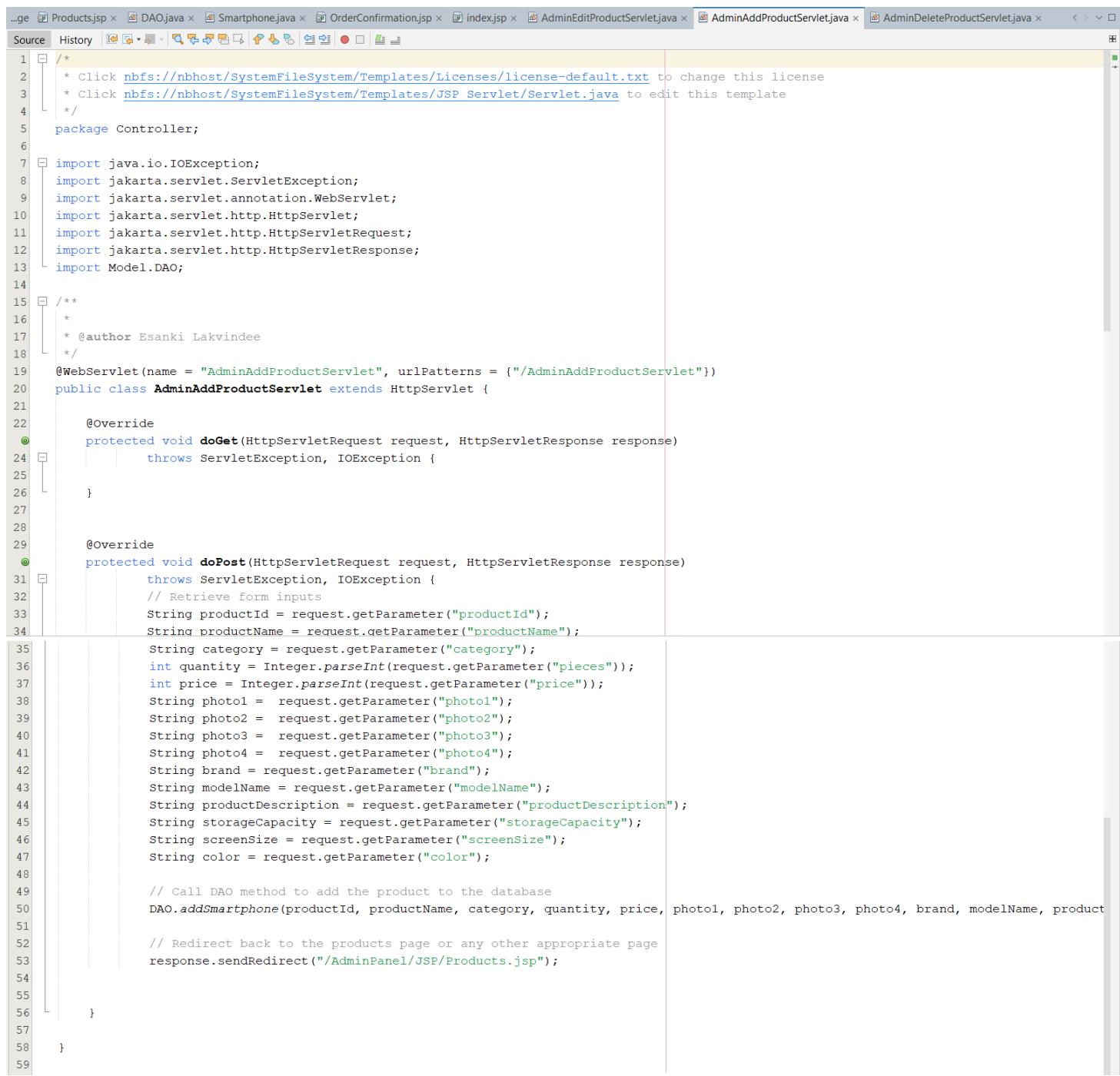
168     <label for="productDescription">Product Description:</label><br>
169     <textarea id="productDescription" name="productDescription" required></textarea><br>
170     <label for="storageCapacity">Storage Capacity:</label>
171     <input type="text" id="storageCapacity" name="storageCapacity" required><br>
172     <label for="screenSize">Screen Size:</label>
173     <input type="text" id="screenSize" name="screenSize" required><br>
174     <label for="color">Color:</label>
175     <input type="text" id="color" name="color" required><br>
176     <button type="submit" onclick="saveProduct()">Save</button>
177     <span class="close" onclick="closetheForm()">&times;

```

AdminEditProductServlet.java

```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template
4  */
5  package Controller;
6
7  import java.io.IOException;
8  import jakarta.servlet.ServletException;
9  import jakarta.servlet.annotation.WebServlet;
10 import jakarta.servlet.http.HttpServlet;
11 import jakarta.servlet.http.HttpServletRequest;
12 import jakarta.servlet.http.HttpServletResponse;
13 import Model.DAO;
14
15 /**
16  *
17  * @author Esanki Lakvindee
18  */
19
20 @WebServlet(name = "AdminEditProductServlet", urlPatterns = {"/AdminEditProductServlet"})
21 public class AdminEditProductServlet extends HttpServlet {
22
23     @Override
24     protected void doGet(HttpServletRequest request, HttpServletResponse response)
25             throws ServletException, IOException {
26     }
27
28
29
30     @Override
31     protected void doPost(HttpServletRequest request, HttpServletResponse response)
32             throws ServletException, IOException {
33         // Retrieve form data including photo parameters
34         String productId = request.getParameter("productId");
35
36         int quantity = Integer.parseInt(request.getParameter("quantity"));
37         double price = Double.parseDouble(request.getParameter("price"));
38         String productName = request.getParameter("productName");
39         String brand = request.getParameter("brand");
40         String modelName = request.getParameter("modelName");
41         String productDescription = request.getParameter("productDescription");
42         String storageCapacity = request.getParameter("storageCapacity");
43         String screenSize = request.getParameter("screenSize");
44         String color = request.getParameter("color");
45         String photo1 = request.getParameter("photo1");
46         String photo2 = request.getParameter("photo2");
47         String photo3 = request.getParameter("photo3");
48         String photo4 = request.getParameter("photo4");
49
50         // Update database including photo parameters
51         DAO.updateSmartphone(productId, quantity, price, productName, brand, modelName, productDescription, storageCapacity, screenSize, color, photo1, photo2, photo3, photo4);
52
53         // Redirect back to the page where the form was submitted from
54         response.sendRedirect("/AdminPanel/JSP/Products.jsp");
55     }
56 }
57 }
```

AdminAddProductServlet.java



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template
 */
package Controller;

import java.io.IOException;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import Model.DAO;

/**
 *
 * @author Esanki Lakvindee
 */
@WebServlet(name = "AdminAddProductServlet", urlPatterns = {" /AdminAddProductServlet"})
public class AdminAddProductServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
    }

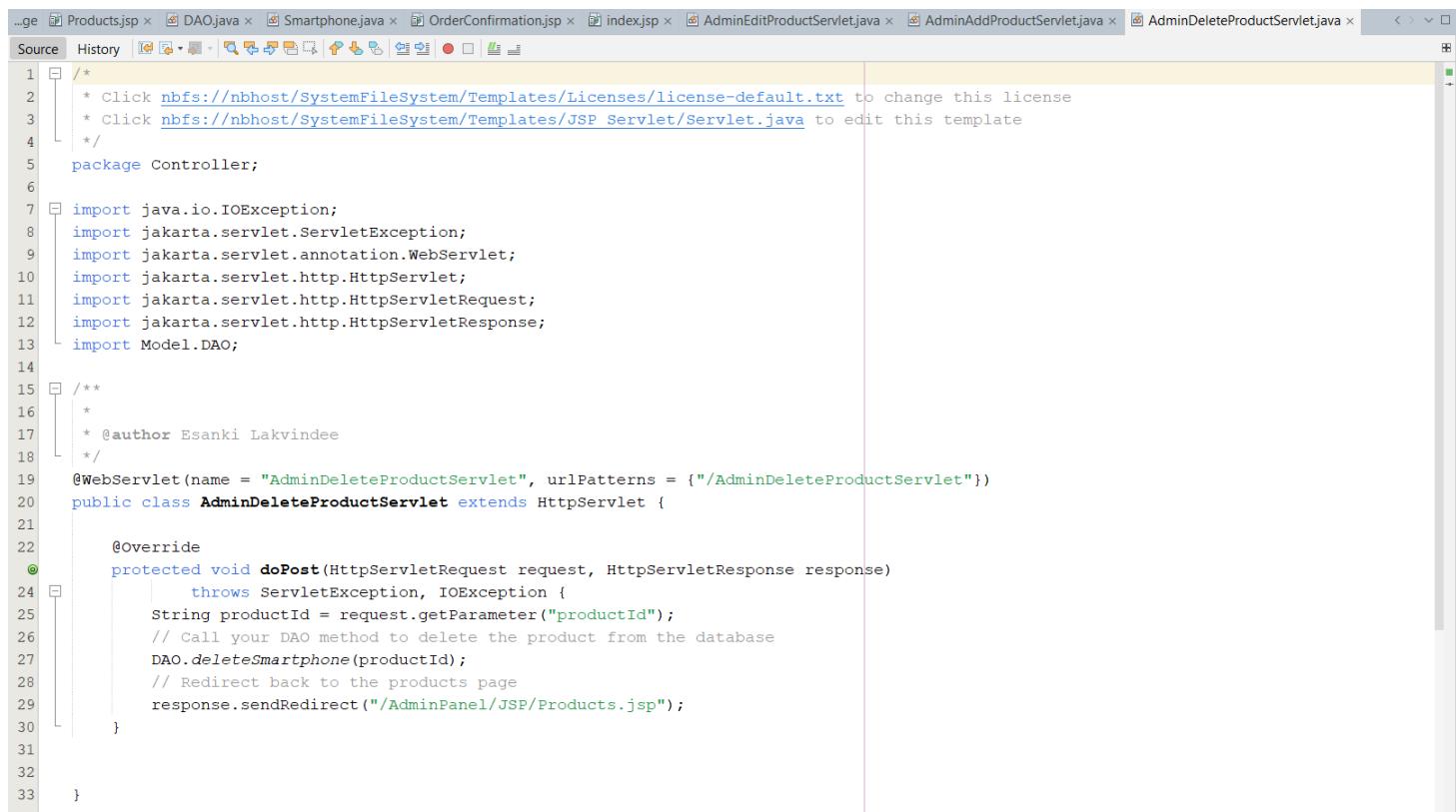
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Retrieve form inputs
        String productId = request.getParameter("productId");
        String productName = request.getParameter("productName");

        String category = request.getParameter("category");
        int quantity = Integer.parseInt(request.getParameter("pieces"));
        int price = Integer.parseInt(request.getParameter("price"));
        String photo1 = request.getParameter("photo1");
        String photo2 = request.getParameter("photo2");
        String photo3 = request.getParameter("photo3");
        String photo4 = request.getParameter("photo4");
        String brand = request.getParameter("brand");
        String modelName = request.getParameter("modelName");
        String productDescription = request.getParameter("productDescription");
        String storageCapacity = request.getParameter("storageCapacity");
        String screenSize = request.getParameter("screenSize");
        String color = request.getParameter("color");

        // Call DAO method to add the product to the database
        DAO.addSmartphone(productId, productName, category, quantity, price, photo1, photo2, photo3, photo4, brand, modelName, productDescription, storageCapacity, screenSize, color);

        // Redirect back to the products page or any other appropriate page
        response.sendRedirect(" /AdminPanel/JSP/Products.jsp");
    }
}
```

AdminDeleteProductServlet.java



```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/JSP Servlet/Servlet.java to edit this template
4   */
5  package Controller;
6
7  import java.io.IOException;
8  import jakarta.servlet.ServletException;
9  import jakarta.servlet.annotation.WebServlet;
10 import jakarta.servlet.http.HttpServlet;
11 import jakarta.servlet.http.HttpServletRequest;
12 import jakarta.servlet.http.HttpServletResponse;
13 import Model.DAO;
14
15 /**
16  *
17  * @author Esanki Lakvindee
18  */
19 @WebServlet(name = "AdminDeleteProductServlet", urlPatterns = {"/AdminDeleteProductServlet"})
20 public class AdminDeleteProductServlet extends HttpServlet {
21
22     @Override
23     protected void doPost(HttpServletRequest request, HttpServletResponse response)
24             throws ServletException, IOException {
25         String productId = request.getParameter("productId");
26         // Call your DAO method to delete the product from the database
27         DAO.deleteSmartphone(productId);
28         // Redirect back to the products page
29         response.sendRedirect("/AdminPanel/JSP/Products.jsp");
30     }
31
32 }
33 }
```

delete.js



```
1  /*
2   * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3   * Click nbfs://nbhost/SystemFileSystem/Templates/JavaScript.js to edit this template
4   */
5
6  function deleteProduct(productId) {
7      if (confirm("Are you sure you want to delete this product?")) {
8          var xhr = new XMLHttpRequest();
9          xhr.open("POST", "/AdminDeleteProductServlet", true);
10         xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
11         xhr.onreadystatechange = function() {
12             if (xhr.readyState === 4 && xhr.status === 200) {
13                 // Reload the page after successful deletion
14                 location.reload();
15             }
16         };
17         xhr.send("productId=" + productId);
18     }
19 }
```

horizontalMenu.css

```
/*
Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
Click nbfs://nbhost/SystemFileSystem/Templates/JSP Servlet/CascadeStyleSheet.css to edit this template
*/
/*
    Created on : Apr 29, 2024, 6:37:19 PM
    Author      : Esanki Lakvindee
*/
/* horizontal-menu.css */

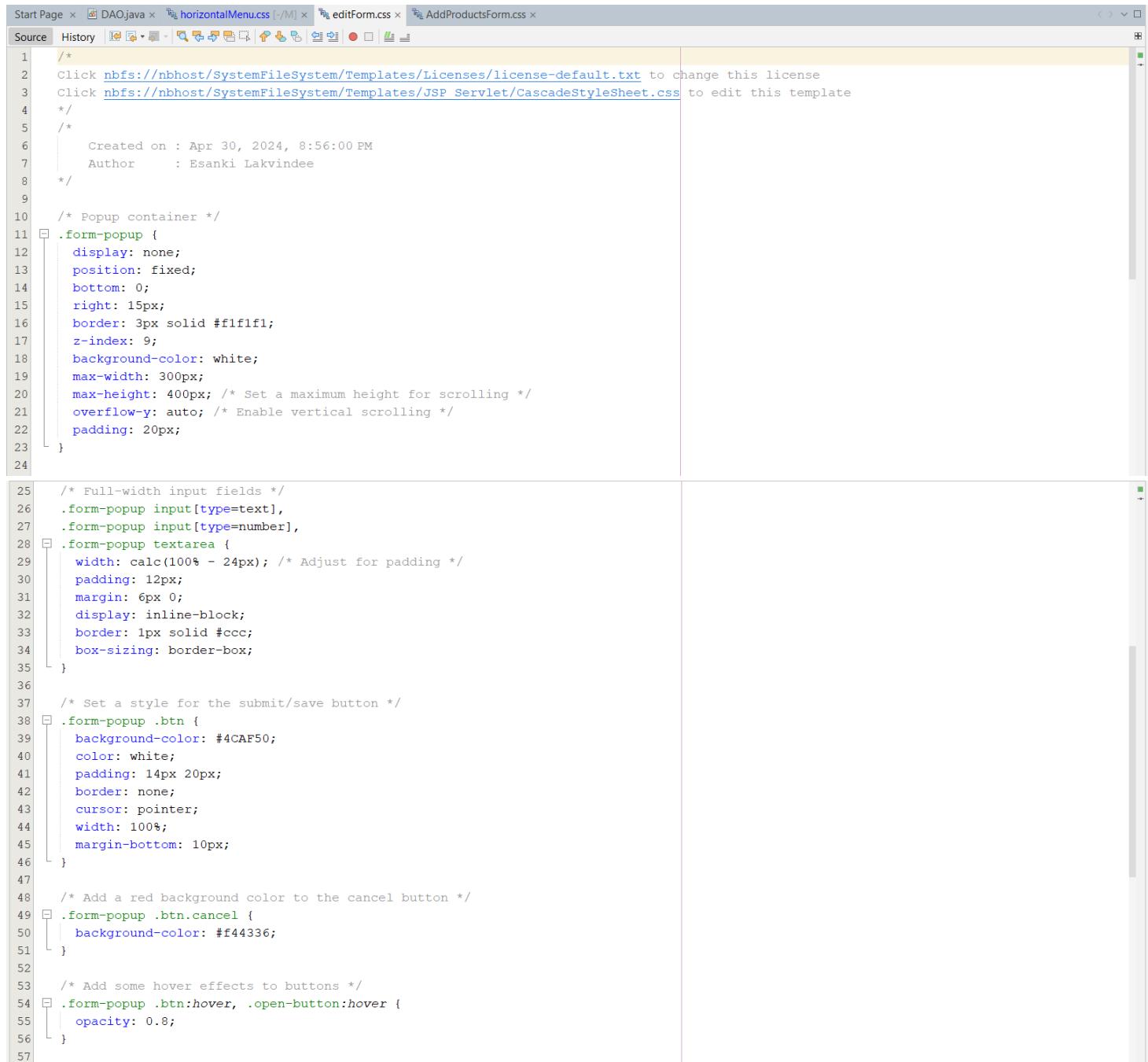
.horizontal-menu {
    list-style-type: none;
    padding: 0;
    margin: 20px auto;
    overflow: hidden;
    background-color: #2c3e50;
    border-radius: 10px;
    display: flex;
    max-width: 900px;
}

.horizontal-menu li {
    flex: 1;
}

.horizontal-menu li a {
    display: block;
    color: #ecf0f1;
    padding: 15px 20px;
    text-decoration: none;
    font-size: 18px;
    font-weight: bold;
    text-transform: uppercase;
    transition: background-color 0.3s ease;
}

.horizontal-menu li a:hover {
    background-color: #34495e;
}
```

editForm.css



The screenshot shows a code editor interface with the tab bar at the top displaying "editForm.css". The main area contains the CSS code for the "form-popup" class.

```
/*
2 Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3 Click nbfs://nbhost/SystemFileSystem/Templates/JSP Servlet/CascadeStyleSheet.css to edit this template
4 */
5 *
6     Created on : Apr 30, 2024, 8:56:00 PM
7     Author      : Esanki Lakvindee
8 */

/* Popup container */
9 .form-popup {
10    display: none;
11    position: fixed;
12    bottom: 0;
13    right: 15px;
14    border: 3px solid #f1f1f1;
15    z-index: 9;
16    background-color: white;
17    max-width: 300px;
18    max-height: 400px; /* Set a maximum height for scrolling */
19    overflow-y: auto; /* Enable vertical scrolling */
20    padding: 20px;
21 }
22 }

/* Full-width input fields */
23 .form-popup input[type=text],
24 .form-popup input[type=number],
25 .form-popup textarea {
26     width: calc(100% - 24px); /* Adjust for padding */
27     padding: 12px;
28     margin: 6px 0;
29     display: inline-block;
30     border: 1px solid #ccc;
31     box-sizing: border-box;
32 }
33

/* Set a style for the submit/save button */
34 .form-popup .btn {
35     background-color: #4CAF50;
36     color: white;
37     padding: 14px 20px;
38     border: none;
39     cursor: pointer;
40     width: 100%;
41     margin-bottom: 10px;
42 }
43

/* Add a red background color to the cancel button */
44 .form-popup .btn.cancel {
45     background-color: #f44336;
46 }
47

/* Add some hover effects to buttons */
48 .form-popup .btn:hover, .open-button:hover {
49     opacity: 0.8;
50 }
51
52
53
54
55
56
57 }
```

```

58  /* Position and style the close button (top right corner) */
59  .form-popup .btn.close {
60      position: absolute;
61      top: 0;
62      right: 0;
63      padding: 12px 16px;
64      margin: -20px -20px 0 0;
65  }
66
67  /* Add animation (fade in the popup) */
68  @keyframes fadeIn {
69      from {opacity: 0;}
70      to {opacity: 1;}
71  }
72
73  .form-popup {
74      animation: fadeIn 0.3s ease-in-out;
75  }
76

```

AddProductsForm.css

Start Page × DAOjava × horizontalMenu.css [-/M] × editForm.css × AddProductsForm.css ×

Source History

```

1  /*
2  Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
3  Click nbfs://nbhost/SystemFileSystem/Templates/JSP Servlet/CascadeStyleSheet.css to edit this template
4  */
5
6  /*
7     Created on : May 1, 2024, 9:35:41 AM
8     Author      : Esanki Lakvindee
9  */
10
11  /* Popup form content */
12  .modal-content {
13      background-color: #fefefe;
14      position: fixed;
15      top: 50%;
16      left: 50%;
17      transform: translate(-50%, -50%);
18      padding: 20px;
19      border: 1px solid #888;
20      width: 50%; /* You can adjust the width as needed */
21      border-radius: 5px;
22      max-height: 80%; /* Set maximum height */
23      overflow-y: auto; /* Enable vertical scrolling */
24      z-index: 9999; /* Ensure modal is on top of other content */
25  }
26
27

```

```
28  /* Center input fields */
29  .modal-content input[type="text"],
30  □ .modal-content input[type="number"] {
31      width: 100%;
32      padding: 12px 20px;
33      margin: 0px 0;
34      display: inline-block;
35      border: 1px solid #ccc;
36      border-radius: 4px;
37      box-sizing: border-box;
38  }
39
40  /* Center buttons */
41  □ .modal-content button[type="submit"] {
42      width: 100%;
43      background-color: #4CAF50;
44      color: white;
45      padding: 14px 20px;
46      margin: 0px 0;
47      border: none;
48      border-radius: 4px;
49      cursor: pointer;
50  }
51
52  □ .modal-content button[type="submit"]:hover {
53      background-color: #45a049;
54  }
55
56
57  □ .close {
58      color: #aaa;
59      float: right;
60      font-size: 28px;
61      font-weight: bold;
62  }
63
64  .close:hover,
65  □ .close:focus {
66      color: black;
67      text-decoration: none;
68      cursor: pointer;
69  }
70
```

DAO.java – updateSmartphone()

```
602     public static void updateSmartphone(String productId, int quantity, double price, String productName, String brand, String modelName) {
603         Connection connection = null;
604         PreparedStatement statement = null;
605
606         try {
607             Class.forName("com.mysql.cj.jdbc.Driver");
608
609             // Establishing connection to the database
610             connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);
611
612             // Prepare SQL statement
613             String sql = "UPDATE smartphone SET quantity=?, price=?, productName=?, brand=?, modelName=?, productDescription=?, storageCapacity=?, screenSize=?, color=?, photo1=?, photo2=?, photo3=?, photo4=? WHERE productId=?";
614             statement = connection.prepareStatement(sql);
615
616             // Set parameters
617             statement.setInt(1, quantity);
618             statement.setDouble(2, price);
619             statement.setString(3, productName);
620             statement.setString(4, brand);
621             statement.setString(5, modelName);
622             statement.setString(6, productDescription);
623             statement.setString(7, storageCapacity);
624             statement.setString(8, screenSize);
625             statement.setString(9, color);
626             statement.setString(10, photo1);
627             statement.setString(11, photo2);
628             statement.setString(12, photo3);
629             statement.setString(13, photo4);
630             statement.setString(14, productId);
631
632             // Execute update
633             statement.executeUpdate();
634         } catch (SQLException e) {
635             Logger.getLogger(DAO.class.getName()).log(Level.SEVERE, null, e);
636         } catch (ClassNotFoundException ex) {
637             Logger.getLogger(DAO.class.getName()).log(Level.SEVERE, null, ex);
638         } finally {
639             // Close resources
640             try {
641                 if (statement != null) {
642                     statement.close();
643                 }
644                 if (connection != null) {
645                     connection.close();
646                 }
647             } catch (SQLException e) {
648                 Logger.getLogger(DAO.class.getName()).log(Level.SEVERE, null, e);
649             }
650         }
651     }
```

DAO.java – deleteSmartphone()

```
653     public static void deletesmartphone(String productId) {  
654         Connection connection = null;  
655         PreparedStatement statement = null;  
656  
657         try {  
658             // Establishing connection to the database  
659             connection = getConnection();  
660  
661             // Prepare SQL statement to delete the product  
662             String sql = "DELETE FROM smartphone WHERE productId = ?";  
663             statement = connection.prepareStatement(sql);  
664  
665             // Set the productId parameter  
666             statement.setString(1, productId);  
667  
668             // Execute the delete statement  
669             statement.executeUpdate();  
670         } catch (SQLException e) {  
671             System.err.println("Error while deleting product: " + e.getMessage());  
672         } finally {  
673             // Close resources  
674             try {  
675                 if (statement != null) {  
676                     statement.close();  
677                 }  
678                 if (connection != null) {  
679                     connection.close();  
680                 }  
681             } catch (SQLException e) {  
682                 System.err.println("Error while closing connection: " + e.getMessage());  
683             }  
684         }  
685     }
```

DAO.java – AddSmartphone()

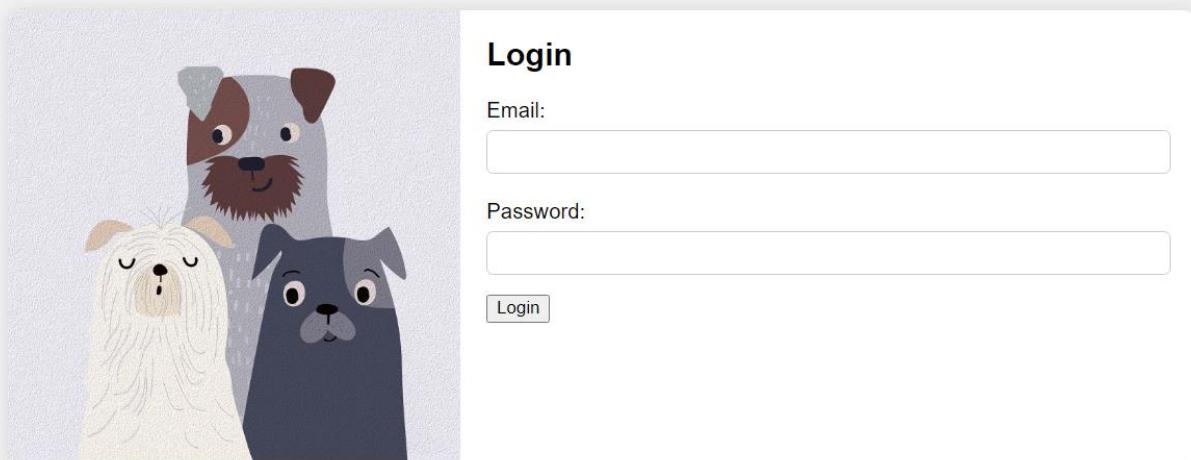
```
721     public static void addsmartphone(String productId, String productName, String category, int quantity, int price, String photo1, Stri  
722         Connection connection = null;  
723         PreparedStatement statement = null;  
724  
725         try {  
726             // Get connection  
727             connection = getConnection();  
728  
729             // Create SQL query  
730             String query = "INSERT INTO smartphone (productId, productName, category, quantity, price, photo1, photo2, photo3, photo4, b  
731             // Create PreparedStatement  
732             statement = connection.prepareStatement(query);  
733  
734             // Set parameters  
735             statement.setString(1, productId);  
736             statement.setString(2, productName);  
737             statement.setString(3, category);  
738             statement.setInt(4, quantity);  
739             statement.setInt(5, price);  
740             statement.setString(6, photo1);  
741             statement.setString(7, photo2);  
742             statement.setString(8, photo3);  
743             statement.setString(9, photo4);  
744             statement.setString(10, brand);  
745             statement.setString(11, modelName);  
746             statement.setString(12, productDescription);  
747             statement.setString(13, storageCapacity);  
748             statement.setString(14, screenSize);  
749             statement.setString(15, color);  
750  
751     }
```

```
752         // Execute the query
753         statement.executeUpdate();
754     } catch (SQLException e) {
755         e.printStackTrace(); // Handle or log the exception appropriately
756     } finally {
757         // Close the connection and statement
758         try {
759             if (statement != null) {
760                 statement.close();
761             }
762             if (connection != null) {
763                 connection.close();
764             }
765         } catch (SQLException e) {
766             e.printStackTrace(); // Handle or log the exception appropriately
767         }
768     }
769 }
770 }
```

2.2 Log In page

Done by : S.N.M. Gedara
Student ID : 29165

2.2.1 Web page



2.2.2 Documentation

- JSP includes input fields for email and password, along with a submit button.
- The page links to an external CSS file named "Login.css" to provide styling for the login form.
- The form is submitted to the servlet named "AdminLoginServlet" using the POST method.
- When the user clicks the login button, the form data is sent to the servlet for processing.

Servlet Handling (AdminLoginServlet):

- The servlet receives the form data (email and password) from the request.
- It checks if the password matches a predefined value (in this case, "emart24051").
- If the password is correct, a session is created, and the user's email is stored in the session.
- The servlet then redirects the user to the admin profile page ("AdminPanel/JSP/Profile.jsp").
- If the password is incorrect, an error message ("Incorrect password") is forwarded to the login page for display.
- After handling the login process, the servlet also inserts login data into the database using the insertloginData method of the AdminLogin class.
- The servlet is mapped to the URL pattern "/AdminLoginServlet" in the @WebServlet annotation.

2.2.3 Codes

Login.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Log In</title>
    <link rel="stylesheet" href="/AdminPanel/CSS/Login.css"/>
  </head>
  <body>

    <div class="container">
      <div class="card">
        <div class="image-column">
          
        </div>
        <div class="form-column">
          <h2>Login</h2>
          <form action="/AdminLoginServlet" method="POST" >
            <div class="input-group">
              <label for="email">Email:</label>
              <input type="email" id="email" name="email" required>
            </div>
            <div class="input-group">
              <label for="password">Password:</label>
              <input type="password" id="password" name="password" required>
            </div>
            <button type="submit" id="loginButton">Login</button>
          </form>
        </div>
      </div>
    </div>
  </body>
</html>
```

AdminLoginServlet.java

```
package Controller;

import Model.AdminLogin;
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;

/*
 * @author Esanki Lakvindee
 */
@WebServlet(name = "AdminLoginServlet", urlPatterns = {"/AdminLoginServlet"})
public class AdminLoginServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String Email = request.getParameter("email");
        String Password = request.getParameter("password");

        // Check if the password is correct
        if ("emart24051".equals(Email)) {
            // Create a session
            HttpSession session = request.getSession();
            // Store user information in session
            session.setAttribute("email", Email);
            // Redirect to profile page
            response.sendRedirect(request.getContextPath() + "/AdminPanel/JSP/Profile.jsp");
        } else {
            // Password is incorrect, display an error message
            request.setAttribute("errorMessage", "Incorrect password");
            request.getRequestDispatcher("AdminPanel/JSP/Login.jsp").forward(request, response);
        }

        AdminLogin login = new AdminLogin();
        login.insertloginData(Email, Password);
    }
}
```

AdminLogin.java

```
package Model;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;

public class AdminLogin {
    static Statement st;

    public void insertloginData(String Email, String Password) {
        // Connect to the database
        connectToDB();

        // Check if the combination already exists
        if (!isDuplicate(Email, Password)) {
            // If it doesn't exist, insert the data
            String query = "INSERT INTO adminusers(email, password) VALUES('" + Email + "', '" + Password + "')";
            try {
                st.executeUpdate(query);
                System.out.println("Record inserted");
            } catch (SQLException ex) {
                System.out.println(ex.getMessage());
            }
        } else {
            System.out.println("Record already exists");
        }
    }
}
```

```

public boolean isDuplicate(String Email, String Password) {
    if (st == null) {
        System.out.println("Database connection is not established");
        return false;
    }

    String query = "SELECT COUNT(*) FROM adminusers WHERE email = '" + Email + "' AND password = '" + Password + "'";
    try (ResultSet rs = st.executeQuery(query)) {
        if (rs.next()) {
            int count = rs.getInt(1);
            return count > 0; // If count > 0, combination exists
        }
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
    return false;
}

public void connectToDB() {
    String driver = "com.mysql.cj.jdbc.Driver";
    String url = "jdbc:mysql://localhost:3306/emart";

    try {
        Class.forName(driver);
        // Establish the connection
        Connection con = DriverManager.getConnection(url, "root", "");
        st = con.createStatement();
    } catch (ClassNotFoundException | SQLException ex) {
        Logger.getLogger(AdminLogin.class.getName()).log(Level.SEVERE, null, ex);
    }
}
}

```

- The connectToDB method establishes a connection to the MySQL database named "emart" running on localhost (port 3306).
- It loads the MySQL JDBC driver (com.mysql.cj.jdbc.Driver) and creates a connection using the DriverManager.

Inserting Login Data (insertloginData method):

- This method inserts admin login data (email and password) into the "adminusers" table of the database.
- Before insertion, it checks if the combination of email and password already exists in the table to prevent duplicate entries.
- If the combination doesn't exist, it constructs an SQL INSERT query and executes it to insert the data into the table.

Checking for Duplicate Entries (isDuplicate method):

- This method checks if a given combination of email and password already exists in the "adminusers" table.
- It constructs an SQL SELECT query to count the number of rows matching the provided email and password.
- If the count is greater than 0, it indicates that the combination exists, and the method returns true; otherwise, it returns false.

Exception Handling:

- The class handles exceptions related to database operations (e.g., SQLException, ClassNotFoundException) by logging the error messages using the Logger class.

UserDAO.java

```
package Model;

/*
 * @author DELL
 */

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import java.sql.*;

public class userDao {

    static Statement st;
    private static Connection con;

    public static void insertDetails(String email, String username, String password, String contactNumber) {
        connectToDB();
        String query = "INSERT INTO users(email,username,password,contactNumber) VALUES('" + email + "','" + username + "','" + password + "','" + contactNumber + "')";
        try {
            st.executeUpdate(query);
            System.out.println("Record inserted");
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }

    public static void connectToDB() {
        String driver = "com.mysql.cj.jdbc.Driver";
        String url = "jdbc:mysql://localhost:3306/emart";

        try {
            Class.forName(driver);
            // Establish the connection
            con = DriverManager.getConnection(url, "root", "");
            // Statement st=con.createStatement();
            st = con.createStatement();

        } catch (ClassNotFoundException | SQLException ex) {
            Logger.getLogger(userDao.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

```

static Statement st;
private static Connection con;

public static void insertDetails(String email, String username, String password, String contactNumber) {
    connectToDB();
    String query = "INSERT INTO users(email,username,password,contactNumber) VALUES('" + email + "','" + username + "','" + password + "','" + contactNumber + "')";
    try {
        st.executeUpdate(string: query);
        System.out.println("Record inserted");
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
}

public static void connectToDB() {
    String driver = "com.mysql.cj.jdbc.Driver";
    String url = "jdbc:mysql://localhost:3306/emart";

    try {
        Class.forName(className: driver);
        // Establish the connection
        con = DriverManager.getConnection(url, user: "root", password: "");
        // Statement st=con.createStatement();
        st = con.createStatement();

    } catch (ClassNotFoundException | SQLException ex) {
        Logger.getLogger(name: userDao.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
    }
}

public static void updateDetails(String email, String newusername, String newcontactNumber) {
    connectToDB();
    // Check if email exists
    String checkQuery = "SELECT * FROM users WHERE email = ?";
    ResultSet rs = null;
    boolean emailExists = false;

    try {
        PreparedStatement checkStatement = con.prepareStatement(string: checkQuery);
        checkStatement.setString(1, string: email);
        rs = checkStatement.executeQuery();
        emailExists = rs.next(); // Check if there's a result
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }
}

```

```

// Update details only if email exists
if(emailExists) {
    String updateQuery = "UPDATE users SET username = ?, contactNumber = ? WHERE email = ?";
    try {
        PreparedStatement updateStatement = con.prepareStatement(string: updateQuery);
        updateStatement.setString(1, string: newusername);
        updateStatement.setString(2, string: newcontactNumber);
        updateStatement.setString(3, string: email);
        updateStatement.executeUpdate();
        System.out.println("Details updated successfully.");
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
} else {
    System.out.println("Email address not found. Update failed.");
    // You can send an error message to the user here
}
}

```

```

public SignUpUser getUserByEmail(String email) {
    connectToDB();
    String query = "SELECT * FROM users WHERE email = '" + email + "'";
    ResultSet rs = null;
    SignUpUser user = null;

    try {
        rs = st.executeQuery(string: query);
        if(rs.next()) {
            user = new SignUpUser(username: rs.getString(string: "username"), password: rs.getString(string: "email"), contactNumber: rs.getString(string: "password"), email: rs.getString(string: "contactNumber"));
        }
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    } finally {
        try {
            if(rs != null) {
                rs.close();
            }
        } catch (SQLException ex) {
            System.out.println(ex.getMessage());
        }
    }

    return user;
}

```

```

public void updatePassword(String email, String newPassword) {
    connectToDB();
    String query = "UPDATE users SET password = '" + newPassword + "' WHERE email = '" + email + "'";
    try {
        st.executeUpdate(string: query);
        System.out.println("Password updated successfully.");
    } catch (SQLException ex) {
        System.out.println(ex.getMessage());
    }
}

```

2.3 Customers page

Done by: G.O.Wickramaratne

Student ID :28039

2.3.1 Web page

The screenshot shows a web application interface for 'E-MART ELECTRONICS'. On the left, there is a sidebar with a blue background containing icons for 'Products', 'Customers', and 'Orders'. The main content area has a white background. At the top, there is a navigation bar with a search bar, a magnifying glass icon, a bell icon, an envelope icon, and a user profile icon labeled 'Admin'. Below the navigation bar, the title 'Customers' is centered above a table. The table has columns for Email, Username, Password, Contact, and Action. There are three rows of data:

Email	Username	Password	Contact	Action
bob@example.com	bob_johnson	password789	1112223333	<button>Delete</button>
jane@example.com	jane_smith	password456	9876543210	<button>Delete</button>
john@example.com	john_doe	password123	1234567890	<button>Delete</button>

2.3.2 Documentation

Customers.jsp:

- This JSP file is responsible for displaying customer information in a tabular format.
- It retrieves customer data from the database and displays it in the table.
- Each row in the table represents a customer, displaying their email, username, password, and contact number.
- It includes a "Delete" button for each customer row, allowing the admin to delete a customer record from the database.
- Additionally, it includes a hidden form that submits customer email to the servlet for deletion.

AdminDeleteCustomerServlet.java:

- This servlet handles the deletion of customer records from the database.
- It overrides the doPost method to handle HTTP POST requests, which occur when the admin submits the delete form.

- It retrieves the customer email parameter from the request and calls the deleteCustomer method in the CustomerDAO class to delete the customer record from the database.
- After successful deletion, it redirects back to the "Customers.jsp" page to display the updated customer list.

CustomerDAO.java:

- This Java class contains database operations related to customer management.
- It includes a static method deleteCustomer to delete a customer record from the database.
- It establishes a database connection, prepares a SQL delete statement, sets the customer email parameter, and executes the delete operation.
- It handles any SQL exceptions that may occur during the delete operation and closes database resources properly.
- Overall, these components together provide functionality for managing customer information in the admin panel, including viewing customer details and deleting customer records when necessary.

2.3.3 Codes

Customers.jsp

```

<%@page import="java.sql.Connection"%>
<%@page import="java.sql.DriverManager"%>
<%@page import="java.sql.Statement"%>
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.SQLException"%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Customer Info</title>
        <link rel="stylesheet" href="/AdminPanel/CSS/Customers.css"/>
        <link rel="stylesheet" href="/AdminPanel/CSS/navbar.css"/>
        <link rel="stylesheet" href="/AdminPanel/CSS/Menu.css"/>
    </head>
    <body>
        <div class="left-section">
            <%@ include file="Menu.html" %>
        </div>

        <div class="right-section">
            <div class="navbar">
                <%@ include file="navbar.html" %>
            ...
        
```

```

</div>
<div class="container">
  <h1>Customers</h1>
  <table>
    <thead>
      <tr>
        <th>Email</th>
        <th>Username</th>
        <th>Password</th>
        <th>Contact</th>
        <th>Action</th>
      </tr>
    </thead>
    <tbody>
      <%
      try {
        Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/emart", "root", "");
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM users");
        while (rs.next()) {
%>
      <tr>
        <td><%= rs.getString("email") %></td>
        <td><%= rs.getString("username") %></td>
        <td><%= rs.getString("password") %></td>
        <td><%= rs.getString("contactNumber") %></td>
        <td>
          <form action="/AdminDeleteCustomerServlet" method="post">
            <input type="hidden" name="email" value="<%= rs.getString("email") %>">
            <button type="submit">Delete</button>
          </form>
        </td>
      </tr>
      <%
        }
        rs.close();
        stmt.close();
        conn.close();
      } catch (SQLException e) {
        e.printStackTrace();
      }
%>
    </tbody>
  </table>
</div>
</div>


<div id="editFormContainer" class="edit-form-container" style="display: none;">
  <div class="edit-form">
    <h2>Edit Customer</h2>
    <form id="editForm">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name">
      <label for="address">Address:</label>
      <input type="text" id="address" name="address">
      <label for="email">Email:</label>
      <input type="email" id="email" name="email">
      <label for="contact">Contact:</label>
      <input type="tel" id="contact" name="contact">
      <button type="button" id="saveBtn">Save</button>
    </form>
  </div>
</div>

```

```
        <button type="button" id="cancelBtn">Cancel</button>
    </form>
</div>
</div>

<script src="/AdminPanel/JS/Customers.js"></script>

</div>
</body>
</html>
```

Customers.js

```
L | */

document.addEventListener("DOMContentLoaded", function () {
    const editButtons = document.querySelectorAll(".editBtn");
    const editFormContainer = document.getElementById("editFormContainer");
    const editForm = document.getElementById("editForm");

    editButtons.forEach(function (button) {
        button.addEventListener("click", function () {
            // Show the edit form popup
            editFormContainer.style.display = "block";
        });
    });

    document.getElementById("cancelBtn").addEventListener("click", function () {
        // Hide the edit form popup when cancel button is clicked
        editFormContainer.style.display = "none";
    });

    document.getElementById("saveBtn").addEventListener("click", function () {
        // Handle form submission here (e.g., send data to server or update the DOM)
        // Once saved, hide the edit form popup
        editFormContainer.style.display = "none";
    });
});
```

Customers.css

```
1  body {
2      margin: 0;
3      padding: 0;
4      font-family: Arial, sans-serif;
5      background-color: #f0f0f0;
6  }
7
8  .left-section {
9      float: left;
10     width: 200px; /* Adjust width as needed */
11     height: 100%; /* Adjust height as needed */
12     background-color: #f0f0f0;
13 }
14
15 .right-section {
16     margin-left: 200px;
17 }
18
19 .container {
20     width: 800px;
21     margin: 20px auto;
22     padding: 20px;
23     background-color: #fff;
24     border-radius: 10px;
25     box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
26 }
27
28 h1 {
29     text-align: center;
30 }
```

```
] table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

] table th, table td {
    border: 1px solid #ccc;
    padding: 10px;
}

] table th {
    background-color: #f0f0f0;
}

] table td button {
    padding: 5px 10px;
    background-color: #333;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

] table td button:hover {
    background-color: #555;
}
```

AdminDeleteCustomerServlet.java

```
4   */
5  package Controller;
6  import Model.CustomerDAO;
7
8  import java.io.IOException;
9  import jakarta.servlet.ServletException;
10 import jakarta.servlet.annotation.WebServlet;
11 import jakarta.servlet.http.HttpServlet;
12 import jakarta.servlet.http.HttpServletRequest;
13 import jakarta.servlet.http.HttpServletResponse;
14
15 /**
16 *
17 * @author hp
18 */
19 @WebServlet(name = "AdminDeleteCustomerServlet", urlPatterns = {"/AdminDeleteCustomerServlet"})
20 public class AdminDeleteCustomerServlet extends HttpServlet {
21
22
23     @Override
24     protected void doGet(HttpServletRequest request, HttpServletResponse response)
25             throws ServletException, IOException {
26         //processRequest(request, response);
27     }
28
29     /**
30      * Handles the HTTP <code>POST</code> method.
31      */
32
33     @Override
34     protected void doPost(HttpServletRequest request, HttpServletResponse response)
35             throws ServletException, IOException {
36         // Get the customer email parameter from the request
37         String customeremail = request.getParameter("email");
38
39         // Call your DAO method to delete the customer from the database
40         CustomerDAO.deleteCustomer(customeremail);
41
42         // Redirect back to the customers page
43         response.sendRedirect("/AdminPanel/JSP/Customers.jsp");
44     }
45
46     // processRequest(request, response);
47
48
49     /**
50      * Returns a short description of the servlet.
51      *
52      * @return a String containing servlet description
53      */
54
55     @Override
56     public String getServletInfo() {
57         return "Short description";
58     }
59 // </editor-fold>
```

CustomerDAO.java

```
4  |  */
5  |  package Model;
6  |
7  |  import java.sql.Connection;
8  |  import java.sql.DriverManager;
9  |  import java.sql.PreparedStatement;
0  |  import java.sql.SQLException;
1  |
2  |  public class CustomerDAO {
3  |      private static final String URL = "jdbc:mysql://localhost:3306/emart";
4  |      private static final String USERNAME = "root";
5  |      private static final String PASSWORD = "";
6  |
7  |      // Method to delete a customer record from the database
8  |      public static void deleteCustomer(String customeremail) {
9  |          Connection conn = null;
0  |          PreparedStatement stmt = null;
1  |
2  |          try {
3  |              // Get a database connection
4  |              conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);
5  |
6  |              // SQL statement to delete customer
7  |              String sql = "DELETE FROM users WHERE email = ?";
8  |
9  |              // Create prepared statement
0  |              stmt = conn.prepareStatement(sql);
1  |
2  |              // Set customer email parameter
3  |              stmt.setString(1, customeremail);
4  |          }
```

```
0     stmt = conn.prepareStatement(sql);
1
2         // Set customer email parameter
3         stmt.setString(1, customeremail);
4
5         // Execute the delete operation
6         stmt.executeUpdate();
7     } catch (SQLException e) {
8         // Handle any SQL errors
9         e.printStackTrace();
10    } finally {
11        // Close resources
12        try {
13            if (stmt != null) {
14                stmt.close();
15            }
16            if (conn != null) {
17                conn.close();
18            }
19        } catch (SQLException e) {
20            e.printStackTrace();
21        }
22    }
23}
24
25}
```

2.4 Admin Orders page

Done by : T.P.R. Fernando

Student ID: 26888

2.4.1 Web page

The screenshot shows a web application interface for managing orders. On the left, there is a vertical sidebar with a dark blue background containing navigation links: 'Products', 'Customers', and 'Orders'. The 'Orders' link is highlighted. The main content area has a light gray background and features a title 'Orders' at the top. Below the title is a table with the following columns: Date, Order ID, Customer Email, Product IDs, Quantity, Total Price, Feedback, Status, Change Order Status, and Action. There are seven rows of data in the table, each representing an order with specific details like date (e.g., 2024-05-10), order ID (e.g., 54886266), customer email (e.g., yay@gmail.com), product ID (e.g., 401,101), quantity (e.g., 116), total price (e.g., LKR 21143940), and status (e.g., processing). The 'Change Order Status' column contains two buttons: 'Processing' and 'Delivered'. The 'Action' column contains a single 'Delete' button.

Date	Order ID	Customer Email	Product IDs	Quantity	Total Price	Feedback	Status	Change Order Status	Action
2024-05-10	54886266	yay@gmail.com	401,101	116	LKR 21143940	null Stars	processing	Processing Delivered	Delete
2024-05-10	61491084	esankilakvindee2000@gmail.com	401,101	116	LKR 21143940	null Stars	processing	Processing Delivered	Delete
2024-05-08	70189676	esankilakvindee2000@gmail.com	401,301,302	305	LKR 185995	null Stars	delivering	Processing Delivered	Delete
2024-05-10	70965881	esankilakvindee2000@gmail.com	401,101	116	LKR 21143940	null Stars	processing	Processing Delivered	Delete
2024-05-08	80512680	esankilakvindee2000@gmail.com	401,301,302	305	LKR 185995	null Stars	processing	Processing Delivered	Delete
2024-05-10	84788847	esankilakvindee2000@gmail.com	401,101	116	LKR 21143940	null Stars	processing	Processing Delivered	Delete
2024-05-08	97566833	esankilakvindee2000@gmail.com	401,402,403	6	LKR 6894	4 Stars	delivering	Processing Delivered	Delete

2.4.2 Documentation

- Imports necessary Java classes from the Model package and Controller package.
- Sets the content type and character encoding for the page.
- Includes external CSS files for styling.

Body Section:

- Divides the page into two sections: left-section and right-section.
- The left-section includes a menu component.
- The right-section displays a table of orders.
- The table contains columns for various order details like date, order ID, customer email, products, quantity, total price, feedback, order status, and action buttons.

Java Code:

- Utilizes JSP scriptlets (`<% %>`) to retrieve order data from the database using DAO methods and iterate over the list of orders to display them dynamically in the table.
- Each row in the table corresponds to an order object, and data is populated accordingly.

CSS Styles (CascadeStyleSheet.css):

Body Styling:

- Sets margin, padding, font-family, and background color for the body.
- Section Styling:
- Defines styling for the left and right sections of the page, including width, height, and background color.

Container Styling:

- Styles the container for the orders table, including width, margin, padding, background color, border radius, and box shadow.

Table Styling:

- Defines styling for the table, including width, border-collapse, and margin.
- Sets border and padding for table cells.

Button Styling:

- Styles the buttons within the table cells, including padding, background color, color, border, border-radius, and cursor.
- Specifies hover effects for buttons.

Java Class (AdminOrderObj.java):

- Represents an order object with properties like order date, order number, order status, product ID, quantity, total price, shipping address, customer name, feedback, email, and product name.
- Initializes the order object with provided data.

Getter and Setter Methods:

- Provides access to the properties of the order object.

2.4.3 Codes

Order.jsp

```
<%@page import="java.util.List"%>
<%@page import="Model.DAO"%>
<%@page import="Model.AdminOrderObj"%>
<%@page import="Controller.DeleteAdminOrder"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Orders</title>
        <link rel="stylesheet" href="/AdminPanel/CSS/navbar.css"/>
        <link rel="stylesheet" href="/AdminPanel/CSS/Menu.css"/>
        <link rel="stylesheet" href="/AdminPanel/CSS/Orders.css"/>
    </head>
    <body>
        <div class="left-section">
            <%@include file="Menu.html" %>
        </div>
```

```

<div class="right-section">
  <div class="container">
    <h1>Orders</h1>
    <table>
      <thead>
        <tr>
          <th>Date</th>
          <th>Order ID</th>
          <th>Customer Email</th>
          <th class="products">Products</th>
          <th>Quantity</th>
          <th>Total Price</th>
          <th>Feedback</th>
          <th>Status</th>
          <th>Change Order Status</th>
          <th>Action</th>
        </tr>
      </thead>
      <tbody>
        <%>
        List<AdminOrderObj> orders = DAO.getAdminOrders();
        for (AdminOrderObj order : orders) {
          %>
          <tr>
            <td><%= order.getOrderDate() %></td>
            <td><%= order.getOrderNumber() %></td>
            <td><%= order.getEmail() %></td>
            <td><%= order.getProductName() %></td>
            <td><%= order.getQuantity() %></td>
            <td>LRP <%= order.getTotalPrice() %></td>
            <td><%= order.getFeedback() %> Stars</td>
            <td><%= order.getOrderStatus() %></td>
            <td>
              <form action="/AdminEditOrderStatusServlet" method="post">
                <input type="hidden" name="orderId" value=<%= order.getOrderNumber() %>>
                <button type="submit" name="orderStatus" value="processing">Processing</button>
                <button type="submit" name="orderStatus" value="delivering">Delivering</button>
                <button type="submit" name="orderStatus" value="delivered">Delivered</button>
              </form>
            </td>
            <td><form action="/DeleteAdminOrder" method="post">
              <input type="hidden" name="orderId" value=<%= order.getOrderNumber() %>>
              <button type="submit">Delete</button>
            </form></td>
          </tr>
          <% } %>
        </tbody>
      </table>
    </div>
  </div>
</html>

```

Order.css

```

body {
  margin: 0;
  padding: 0;
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
  justify-content: center important;
  align-content: center important;
}

.left-section {
  float: left;
  width: 200px; /* Adjust width as needed */
  height: 100%; /* Adjust height as needed */
  background-color: #f0f0f0;
}

.right-section {
  margin-left: 200px;
}

.container {
  width: 1300px;
  margin: 20px auto;
  padding: 20px;
  background-color: #fff;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  justify-content: center important;
  align-content: center important;
}

```

```

    h1 {
        text-align: center;
    }

    table {
        width: 100%;
        border-collapse: collapse;
        margin-top: 20px;
        justify-content: center important;
        align-content: center important;
    }

    table th, table td {
        border: 1px solid #ccc;
        padding: 10px;
        justify-content: center important;
        align-content: center important;
    }

    table th {
        background-color: #f0f0f0;
    }

    /* Buttons */
    table td button {
        padding: 5px 10px;
        background-color: #333;
        color: #fff;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }

    table td button:hover {
        background-color: #555;
    }

    /* Make buttons horizontal */
    table td.b2 {
        margin-left: 3px;
    }

    .products{
        width: 115px;
    }

```

AdminOrderObj.java

```

package Model;

import java.util.Date;

/*
 * @author robin
 */
public class AdminOrderObj {

    private Date orderDate;
    private int orderNumber;
    private String orderStatus;
    private int productId;
    private int quantity;
    private int totalPrice;
    private String shippingAddress;
    private String customerName;
    private String feedback;
    private String email;
    private String productName;

    public AdminOrderObj(Date orderDate, int orderNumber, String orderStatus, int productId, int quantity, int totalPrice, String shippingAddress, String customerName, String feedback, String email) {
        this.orderDate = orderDate;
        this.orderNumber = orderNumber;
        this.orderStatus = orderStatus;
        this.productId = productId;
        this.quantity = quantity;
        this.totalPrice = totalPrice;
        this.shippingAddress = shippingAddress;
        this.customerName = customerName;
        this.feedback = feedback;
        this.email = email;
    }
}

```

```
public int getProductId() {
    return productId;
}

public void setProductId(int productId) {
    this.productId = productId;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public Date getOrderDate() {
    return orderDate;
}

public void setOrderDate(Date orderDate) {
    this.orderDate = orderDate;
}

public int getOrderNumber() {
    return orderNumber;
}

public void setOrderNumber(int orderNumber) {
    this.orderNumber = orderNumber;
}

public String getOrderStatus() {
    return orderStatus;
}

public void setOrderStatus(String orderStatus) {
    this.orderStatus = orderStatus;
}

public int getProductId() {
    return productId;
}

public void setProductId(int products) {
    this.productId = products;
}

public int getQuantity() {
    return quantity;
}

public void setQuantity(int quantity) {
    this.quantity = quantity;
}

public int getTotalPrice() {
    return totalPrice;
}

public void setTotalPrice(int totalPrice) {
    this.totalPrice = totalPrice;
}

public String getShippingAddress() {
    return shippingAddress;
}

public void setShippingAddress(String shippingAddress) {
    this.shippingAddress = shippingAddress;
}

public String getCustomerName() {
    return customerName;
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}

public String getFeedback() {
    return feedback;
}

public void setFeedback(String feedback) {
    this.feedback = feedback;
}

void setProductName(String productName) {
    this.productName = productName;
}

public String getProductName() {
    return productName;
}
```

Contribution of team members

Student ID	Name	Contribution to the Emart website	Contribution to the Admin Panel
29186	E.L. Thambawita	<p>Front end and back end of :</p> <ul style="list-style-type: none"> • Home page • Order Confirmation page 	<p>Front end of :</p> <ul style="list-style-type: none"> • Index page (products pages) • Customers page • Orders page • Log-in page • Nav bar <p>Back end of :</p> <ul style="list-style-type: none"> • Products pages (index) which include five different pages in total for each category. <ul style="list-style-type: none"> • Smartphones • Laptops • Camera • Monitors • Smartwatches
26888	T.P.R. Fernando	<p>Front end and back end of</p> <ul style="list-style-type: none"> • Shopping cart page <p>Others :</p> <ul style="list-style-type: none"> • Navbar • Footer • Database 	<ul style="list-style-type: none"> • Back end of Orders page
29165	S.N.M. Gedara	<p>Front end and back end of :</p> <ul style="list-style-type: none"> • Sign In page <ul style="list-style-type: none"> • Forgot password • Verify OTP • Reset password • Sign up page • About Us page • Developers page • Track Order page <p>Back end of</p> <ul style="list-style-type: none"> • Profile page 	<ul style="list-style-type: none"> • Back end of Log In page

28039	G.O. Wickramaratne	Front end and back end of : <ul style="list-style-type: none"> • Check out page • Contact page 	• Back end of Customers page
30143	W.M.C.S. Wijesinghe	Front end and back end of <ul style="list-style-type: none"> • Product category pages • Product information pages 	_____
26545	W.A.C. Fernando	• Front end of profile page	_____