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.

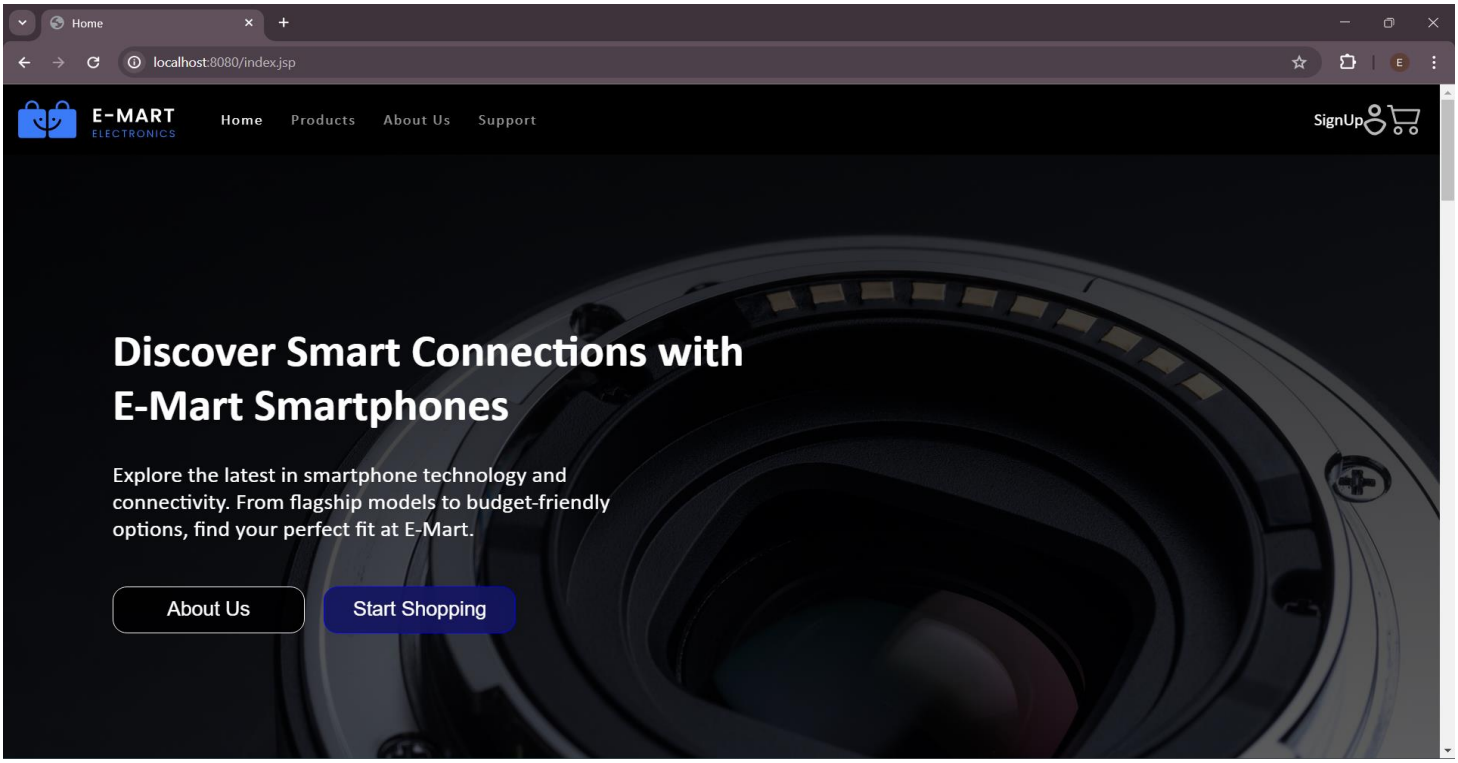
Content **(Click to navigate directly to any section)**

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

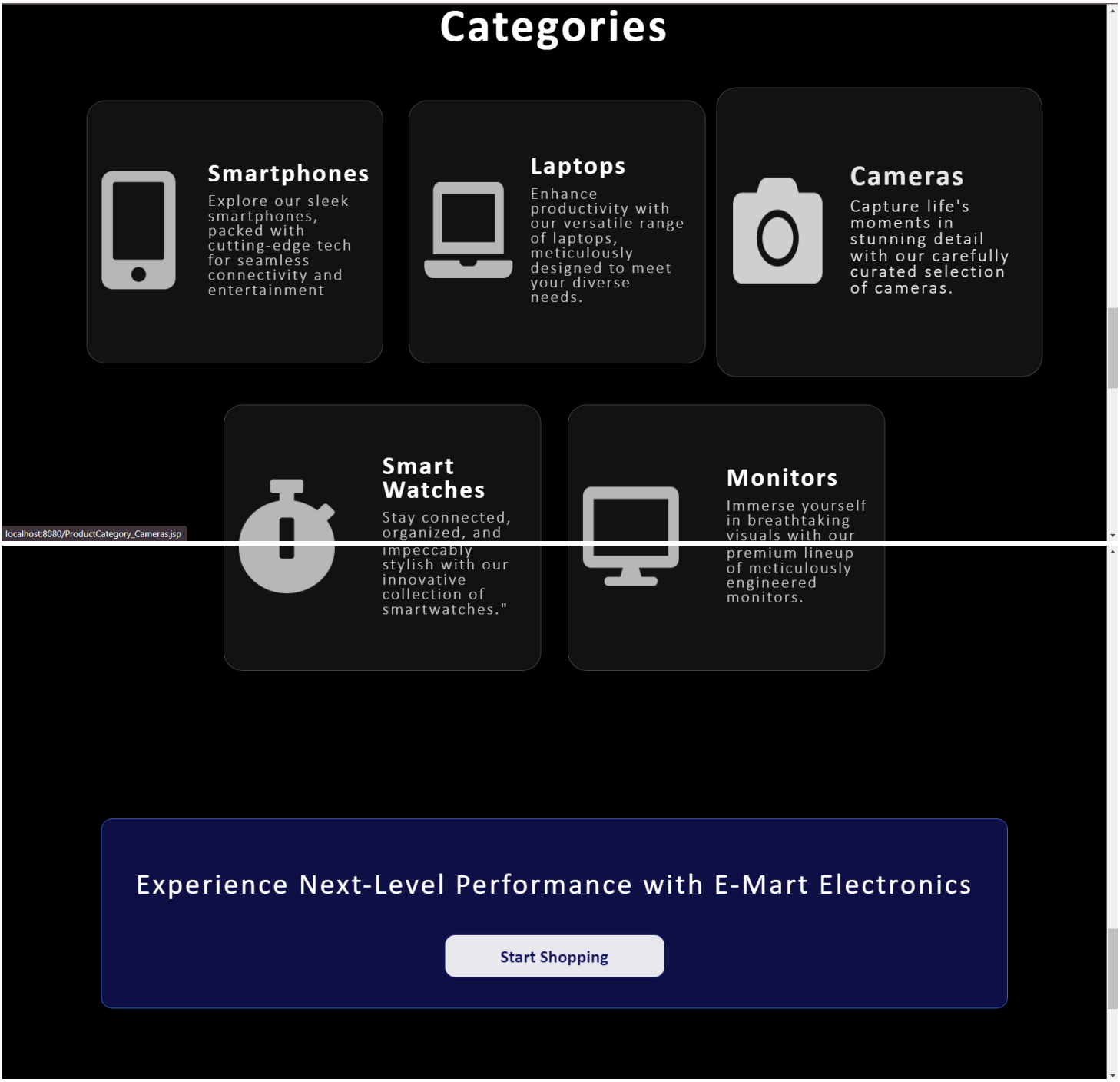
New Arrivals

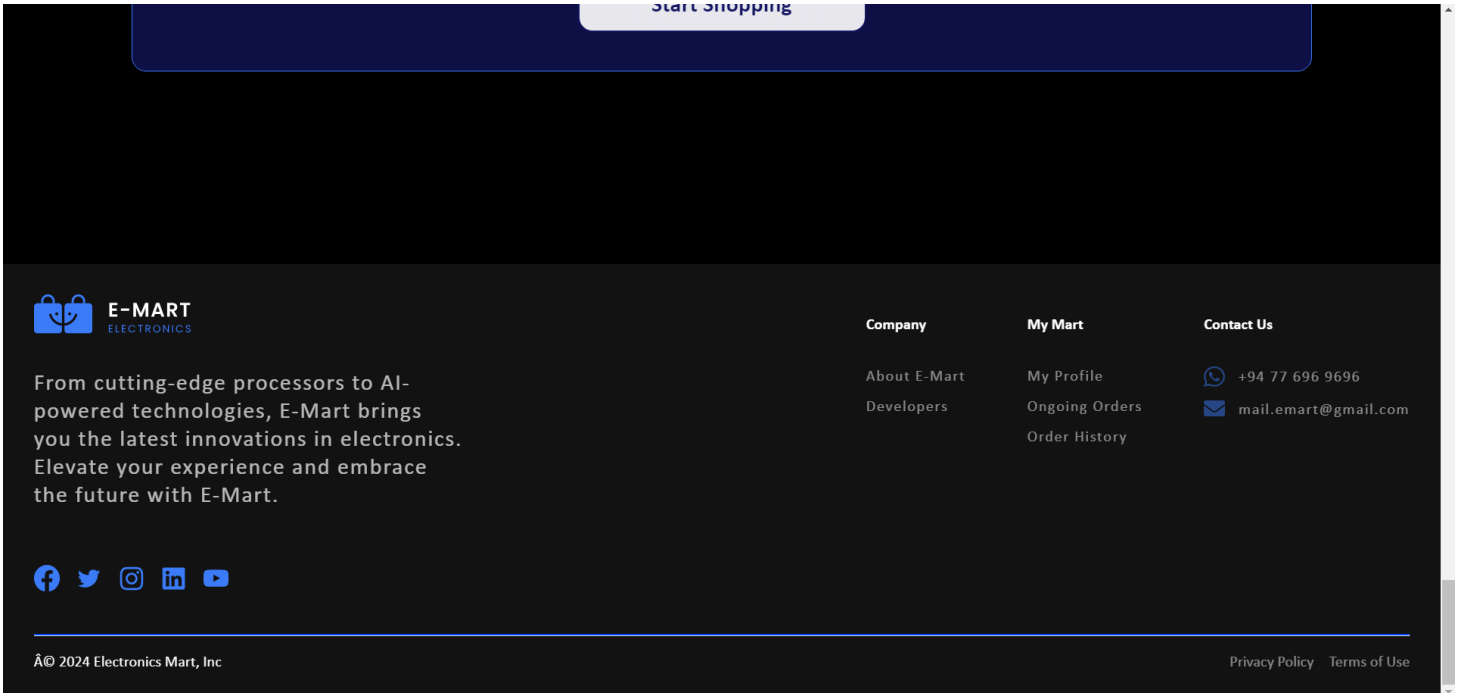
name
LKR 56546

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.
localhost:8080/ProductCategory_Cameras.jsp
Experience Next-Level Performance with E-Mart Electronics
Start Shopping



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

```jsp
<%--
    Document   : Home
    Created on : Apr 11, 2024, 3:53:02 PM
    Author     : Esanki Lakvindee
--%>

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@page import="java.util.List"%>
<%@page import="Model.DAO"%>
<%@page import="Model.NewestProductObj"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Home</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/Home.css" rel="stylesheet" type="text/css"/>
    </head>

    <body>
        <%@ include file="Navbar.jsp" %>

        <% DAO.getAllCartItems(); %>
        <% DAO.getAllSmartphones(); %>
        <% DAO.getNewArrivals(); %>

        <!-- Home page UI -->
        <div class="container">
            <img src="Images/Home/lens.jpg" alt="Background Image" class="background-image">
            <div class="gradient-overlay"></div>
            <div class="content">
                <h1>Discover Smart Connections with<br>E-Mart Smartphones</h1>
                <p>Explore the latest in smartphone technology and<br>connectivity. From flagship models to budget-friendly<br>
                    options, find your perfect fit at E-Mart.</p>
                <div class="buttons">
                    <a href="AboutUs.jsp">
                        <button class="button1">About Us</button>
                    </a>
                    <a href="ProductCategory_Cameras.jsp">
                        <button class="button2">Start Shopping</button>
                    </a>
                </div>
            </div>
        </div>

        <!-- Second Section -->
        <div class="second-section">
            <h2 class="Topic">Best Sellers</h2>
            <div class="rectangle-container">
                <div class="rectangle">
                    <a href="ProductCategory_Phones.jsp">
                        <div class="img-container">
                            <img src="Images/Home/S24.png" alt="Product Image">
                        </div>
                        <span class="text1">Samsung Galaxy S24</span>
                        <span class="text2">LKR 400,000</span>
                    </a>

                </div>

                <div class="rectangle">
                    <a href="ProductCategory_Phones.jsp">
                        <div class="img-container">
                            <img src="Images/Home/S22.png" alt="Product Image">
                        </div>
                        <span class="text1">Samsung Galaxy S22</span>
                        <span class="text2">LKR 300,000</span>
                    </a>

                </div>
```

```
74                  <div class="rectangle">
75                      <a href="ProductCategory_Phones.jsp">
76                          <div class="img-container">
77                              <img src="Images/Home/iphone14.png" alt="Product Image">
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                              <img src="Images/Home/iphone15.png" alt="Product Image">
89                          </div>
90                          <span class="text1">Iphone 15</span>
91                          <span class="text2">LKR 500,000</span>
92                      </a>
94                  </div>
95              </div>
96          </div>
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 newItems = newArrivals.get(i);
106                 %>
107                 <div class="rectangle">
108                     <a href="ProductCategory_Cameras.jsp">
109                         <div class="img-container">
110                             <img src="<%= newItems.getIconPath() %>" alt="Product Image">
111                         </div>
112                         <span class="text1"><%= newItems.getProductName() %></span>
113                         <span class="text2">LKR <%= newItems.getProductPrice() %></span>
114                     </a>
115                 </div>
116                 <% } %>
117             </div>
118
119             <!--                 <div class="rectangle">
120                                 <a href="abc.jsp">
121                                     <img src="Images/Home/SONY alpha.png" alt="Product Image">
122                                     <span class="text1">SONY Alpha 9 II</span>
123                                     <span class="text2">LKR 300,000</span>
124                                 </a>
125
126                             </div>
127
128                             <div class="rectangle">
129                                 <a href="abc.jsp">
130                                     <img src="Images/Home/SONY RX10.png" alt="Product Image">
131                                     <span class="text1">SONY RX10 IV</span>
132                                     <span class="text2">LKR 450,000</span>
133                                 </a>
134
135                             </div>
136
137                             <div class="rectangle">
138                                 <a href="abc.jsp">
139                                     <img src="Images/Home/iphone15.png" alt="Product Image">
```

```
                                <span class="text1">Iphone 15</span>
                                <span class="text2">LKR 500,000</span>
                            </a>

                        </div>-->
        </div>

        <!--Fourth section -->
        <div class="categories-text">Categories</div>

        <div class="category-container">
            <a href="ProductCategory_Phones.jsp" class="category1">
                <div class="column1">
                    <img src="Images/Home/phone.png" alt="Image 1" style="height: 160px; width: 100px;">
                </div>
                <div class="column2">
                    <div class="line1">Smartphones</div>
                    <div class="line2">Explore our sleek smartphones, packed with cutting-edge tech for seamless connectivity and enter
                </div>
            </a>

            <a href="ProductCategory_Laptops.jsp" class="category2">
                <div class="column1">
                    <img src="Images/Home/laptop.png" alt="Image 2" style="height: 130px; width: 120px;">
                </div>
                <div class="column2">
                    <div class="line1">Laptops</div>
                    <div class="line2">Enhance productivity with our versatile range of laptops, meticulously designed to meet your div
                </div>
            </a>

            <a href="ProductCategory_Cameras.jsp" class="category3">
                <div class="column1">
                    <img src="Images/Home/camera.png" alt="Image 3" style="height: 130px; width: 110px;">
                </div>
                <div class="column2">
                    <div class="line1">Cameras</div>
                    <div class="line2">Capture life's moments in stunning detail with our carefully curated selection of cameras.</div>
                </div>
            </a>
        </div>

        <!-- Fifth section -->
        <div class="nextcategory-container">
            <a href="ProductCategory_Smartwatches.jsp" class="category4">
                <div class="column1">
                    <img src="Images/Home/watch.png" alt="Image 3" style="height: 150px; width: 130px;">
                </div>
                <div class="column2">
                    <div class="line1">Smart<br><br>Watches</div>
                    <div class="line2">Stay connected, organized, and impeccably stylish with our innovative collection of smartwatches
                </div>
            </a>

            <a href="ProductCategory_Monitors.jsp" class="category5">
                <div class="column1">
                    <img src="Images/Home/monitor.png" alt="Image 3" style="height: 130px; width: 130px;">
                </div>
                <div class="column2">
                    <div class="line1">Monitors</div>
                    <div class="line2">Immerse yourself in breathtaking visuals with our premium lineup of meticulously engineered moni
                </div>
            </a>
        </div>


        <!-- Sixth section -->
        <div class="longrectangle-container">
            <div class="longrectangle">
                <p class="sentence">Experience Next-Level Performance with E-Mart Electronics</p>
                <button class="start-shopping" onclick="window.location.href = 'ProductCategory_Phones.jsp';">Start Shopping</button>
            </div>
        </div>
        <%@ include file="Footer.html" %>
        <script src="JS/Common.js"></script>
    </body>
</html>
```

Home.css

Start Page × DAO.java × index.jsp × Home.css × NewestProductObj.java ×

Source History

```
/*
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 11, 2024, 3:53:39 PM
    Author     : Esanki Lakvindee
*/

/*Global styles*/
body, html {
    margin: 0;
    padding: 0;
    font-family: Calibri, sans-serif;
    color: #FFF;
    background-color: black;
    height: 100%;
}

.gradient-overlay {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background-image: linear-gradient(180deg, rgba(0,0,0,0.8) 0%, rgba(0,0,0,0.6) 100%);
    z-index: 1; /* Ensure the gradient overlay is behind the content but on top of the background image */
}

.container {
    position: relative;
    width: 100%;
    height: 815px; /* Adjust height as needed */
    overflow: hidden; /* Ensure that content beyond the height of the container is hidden */
    background-color: black; /* Set background color as desired */
}

.background-image {
    width: 100%;
    height: 815px; /* Adjust height as needed */
    object-fit: cover;
}

.content {
    position: absolute;
    top: 149px;
    left: 114px;
    text-align: left;
    z-index: 2;
}

h1, p {
    font-size: 3rem;
    margin-bottom: 2rem;
}

p {
    font-size: 1.5rem;
    margin-bottom: 3rem;
}

.buttons {
    display: flex;
    justify-content: flex-start;
    align-items: center;
    gap: 20px;
```

```
button {
    width: 200px;
    height: 50px;
    border-radius: 15px;
    border: 0.5px solid #FFF;
    background: rgba(0, 0, 0, 0.00);
    font-size: 21px;
    cursor: pointer;
    color: #FFF;
}

.button1 {
    background: rgba(0, 0, 0, 0.90); /* Black color background */
}

.button2 {
    background: rgba(21, 22, 99, 0.90) !important;
    backdrop-filter: blur(50px);
    border: 0.5px solid blue;
}


/*Second Section*/
.second-section {
    position: relative;
    margin-top: 141px; /* Placed after 141px after the image lens */
    text-align: center;
}

.Topic {
    color: #FFF;
    font-family: Calibri;
    font-size: 64px;
    font-style: normal;
    font-weight: 700;
    line-height: normal;
    letter-spacing: 3.2px;
}

.rectangle-container {
    display: flex;
    justify-content: space-between;
    margin-left: 114px; /* Space from left page border */
    margin-right: 107px; /* Space from right page border */
    gap: 35px; /* Gap between rectangles */
}

.rectangle {
    width: 394px;
    height: 523px;
    border-radius: 25px;
    border: 4px solid rgba(255, 255, 255, 0.40);
    background: linear-gradient(0deg, rgba(255, 255, 255, 0.07) 0%, rgba(255, 255, 255, 0.07) 100%), linear-gradient(209deg, rgba(0, 0,
    overflow: hidden; /* Ensures that the border-radius is applied to the background */
}


.rectangle img {
    width: 90%;
    margin-top: 41px;
    max-height: 100%;
}

.text1, .text2 {
    display: block;
```

```
    color: #FFF;
    font-family: Calibri;
    font-style: normal;
    line-height: 16px;
    letter-spacing: 2.24px;
}

.text1 {
    font-size: 28px;
    font-weight: 700;
    margin-top: 50px;
}

.text2 {
    font-size: 32px;
    font-weight: 400;
    color: rgba(255, 255, 255, 0.70);
    margin-top: 26px;
}

/*Fourth Section*/
.categories-text {
    text-align: center;
    color: #FFF;
    font-family: Calibri;
    font-size: 64px;
    font-weight: 700;
    line-height: 16px;
    letter-spacing: 3.2px;
    margin-top: 171px;
}
```

```
.category-container {
    display: flex;
    justify-content: space-between;
    margin-left: 114px; /* Space from left page border */
    margin-right: 107px; /* Space from right page border */
    margin-top: 93px;
}

.category-container .category1,
.category-container .category2,
.category-container .category3 {
    width: 542px; /* Adjust width as needed */
    height: 314px; /* Adjust height as needed */
    border-radius: 25px;
    border: 1px solid rgba(255, 255, 255, 0.26);
    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
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 20px;
    margin-right: 35px; /* Gap between each category */
}

.category-container .category1 {
    margin-left: 0;
}

.category-container .category3 {
    margin-right: 0;
}
```

```css
.category-container .column1 img {
    width:40%;
}

.category-container .column2 {
    width: 60%; /* Each takes half of the container */
}

.category-container .line1 {
    color: #FFF;
    font-family: Calibri;
    font-size: 35px;
    font-weight: 700;
    line-height: 16px;
    letter-spacing: 2.4px;
    margin-bottom: 20px;
}

.category-container .line2 {
    color: rgba(255, 255, 255, 0.80);
    font-family: Calibri;
    font-size: 22px;
    font-weight: 300;
    line-height: 20px;
    letter-spacing: 2.4px;
}

/* Fifth section */
.nextcategory-container {
    display: flex;
    justify-content: space-between;
    margin-left: 300px;
    margin-right: 300px;
    margin-top: 56px;
}

.nextcategory-container .category4,
.nextcategory-container .category5 {
    width: 542px; /* Adjust width as needed */
    height: 314px; /* Adjust height as needed */
    border-radius: 25px;
    border: 1px solid rgba(255, 255, 255, 0.26);
    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
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 20px;
    margin-right: 36px; /* Gap between each category */
}

.nextcategory-container .category4 {
    margin-left: 0;
}

.nextcategory-container .category5 {
    margin-right: 0;
}
```

```css
.nextcategory-container .column1 img {
    height: 176px; /* Image height */
    width: 141px; /* Image width */
}

.nextcategory-container .column2 {
    width: 50%; /* Each takes half of the container */
}

.nextcategory-container .line1 {
    color: #FFF;
    font-family: Calibri;
    font-size: 35px;
    font-weight: 700;
    line-height: 16px;
    letter-spacing: 2.4px;
    margin-bottom: 20px;
}

.nextcategory-container .line2 {
    color: rgba(255, 255, 255, 0.80);
    font-family: Calibri;
    font-size: 22px;
    font-weight: 300;
    line-height: 20px;
    letter-spacing: 2.4px;
}

/* Sixth section */
.longrectangle-container {
    display: flex;
    justify-content: center;
    margin-left: 134px;
    margin-right: 134px;
    margin-top: 200px;
    margin-bottom: 200px;
}

.longrectangle {
    width: 1652px;
    height: 255px;
    border-radius: 15px;
    border: 1px solid #3B7BF8;
    background: linear-gradient(209deg, rgba(21, 22, 99, 0.70) 17.86%, rgba(21, 22, 99, 0.70) 82.12%);
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}

.sentence {
    color: #FFF;
    font-family: Calibri;
    font-size: 40px;
    font-weight: 400;
    line-height: 16px; /* 33.333% */
    letter-spacing: 3px;
    margin-bottom: 60px; /* Adjust as needed */
}
```

```css
.start-shopping {
    width: 300px;
    height: 60px;
    border-radius: 15px;
    border: 2px solid #151663;
    background: rgba(255, 255, 255, 0.90);
    backdrop-filter: blur(50px);
    font-family: Calibri;
    font-size: 24px;
    font-weight: 700;
    color: #151663;
    cursor: pointer;
}

/*Clicking parts of second, third sections*/
.rectangle-container .rectangle {
    position: relative;
}

.rectangle-container .rectangle a {
    display: block;
    position: relative;
}

.rectangle-container .rectangle a img {
    display: block;
    transition: filter 0.3s ease; /* Smooth transition for the hover effect */
}

.rectangle-container .rectangle a:hover img {
    filter: brightness(70%);
}

.rectangle-container .rectangle a:hover .text1,
.rectangle-container .rectangle a:hover .text2 {
    color: rgba(255, 255, 255, 0.9);
}


/*Clicking part of fourth,fifth sections*/
.category-container a,
.nextcategory-container a {
    display: block;
    text-decoration: none; /* Remove underline from links */
    transition: transform 0.3s ease; /* Smooth transition for the hover effect */
}

.category-container a:hover,
.nextcategory-container a:hover {
    transform: scale(1.1); /* Increase size by 10% when hovered */
}

.category-container a:hover .line1,
.category-container a:hover .line2,
.nextcategory-container a:hover .line1,
.nextcategory-container a:hover .line2 {
    color: rgba(255, 255, 255, 0.9); /* Change text color when hovered */
}

a {
    text-decoration: none;
}

.img-container{
    height: 350px;
    display: flex;
    flex-direction: column;
    align-content: center;
    justify-content: center;
    align-items: center;
}

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

DAO.java – getNewArrivals()

Start Page × DAO.java × index.jsp × Home.css × NewestProductObj.java ×

Source History

```java
/*
 * 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 = "";

    public static Connection getConnection() throws SQLException {
        return DriverManager.getConnection(URL, USERNAME, PASSWORD);
    }
```

Source History

```java

    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 × | DAO.java × | index.jsp × | Home.css × | NewestProductObj.java ×

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:

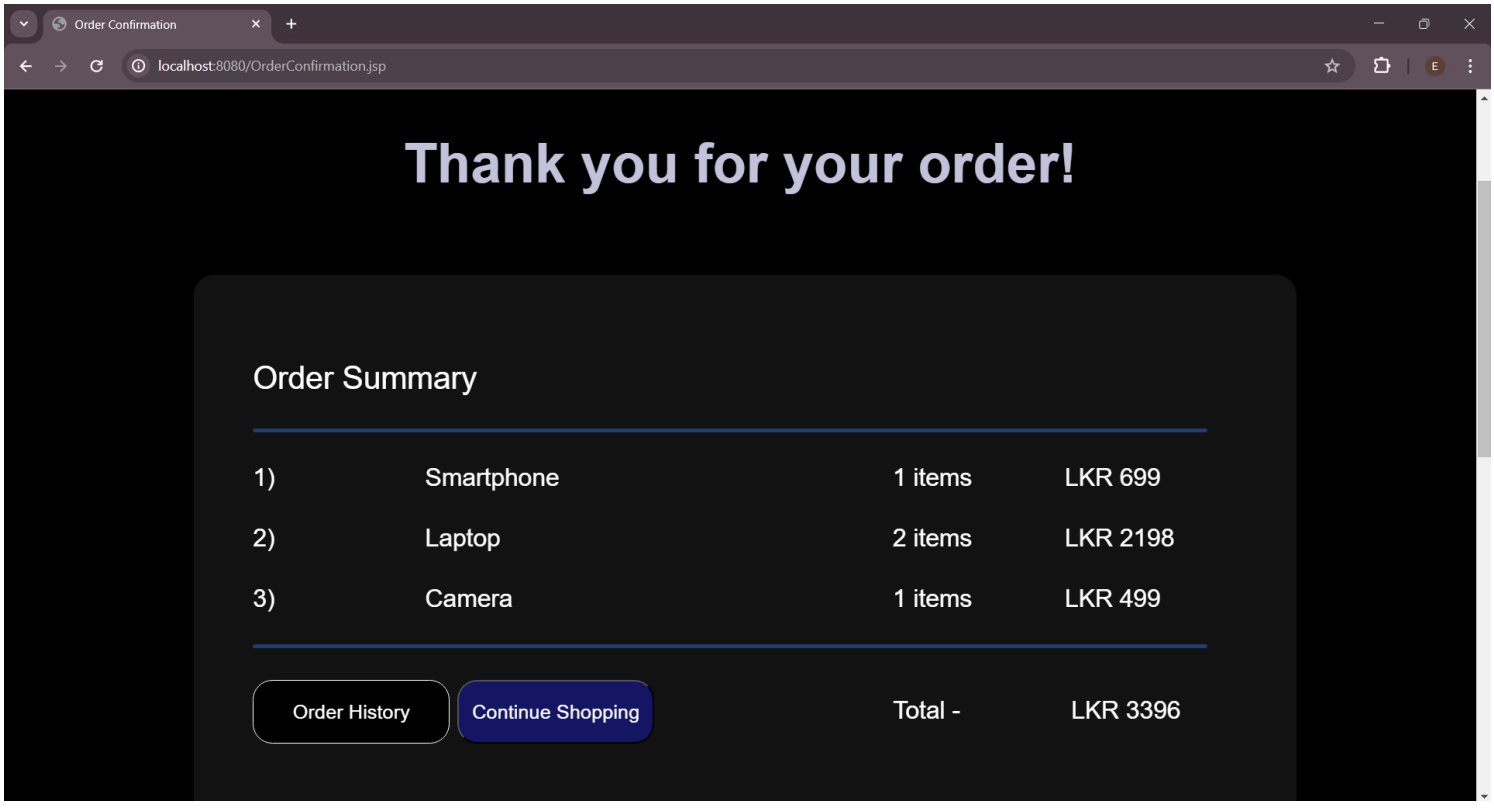
The Home Page is the website's main entry point, effectively communicating the brand's message, showcasing featured products, highlighting new arrivals, and providing easy navigation options. The Home Page's meticulous design and dynamic content integration aims to engage users and encourage them to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

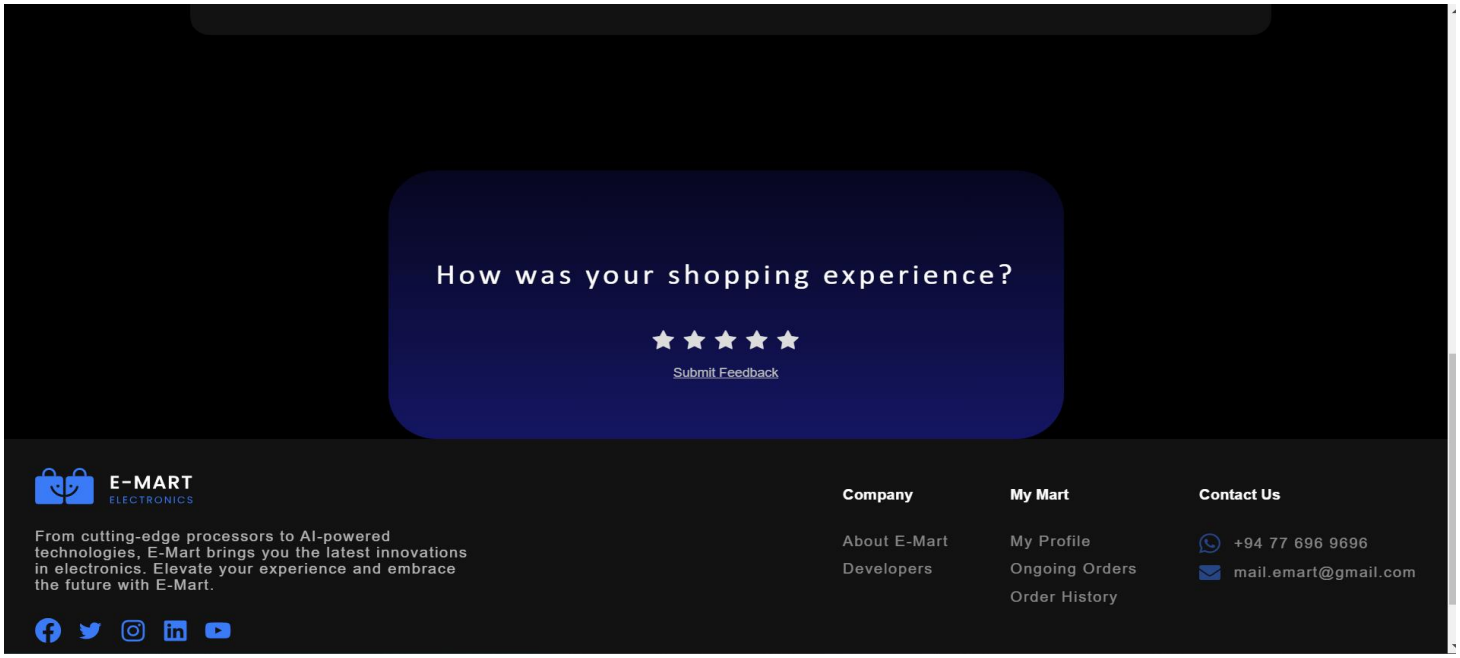
1.2 Order Confirmation page

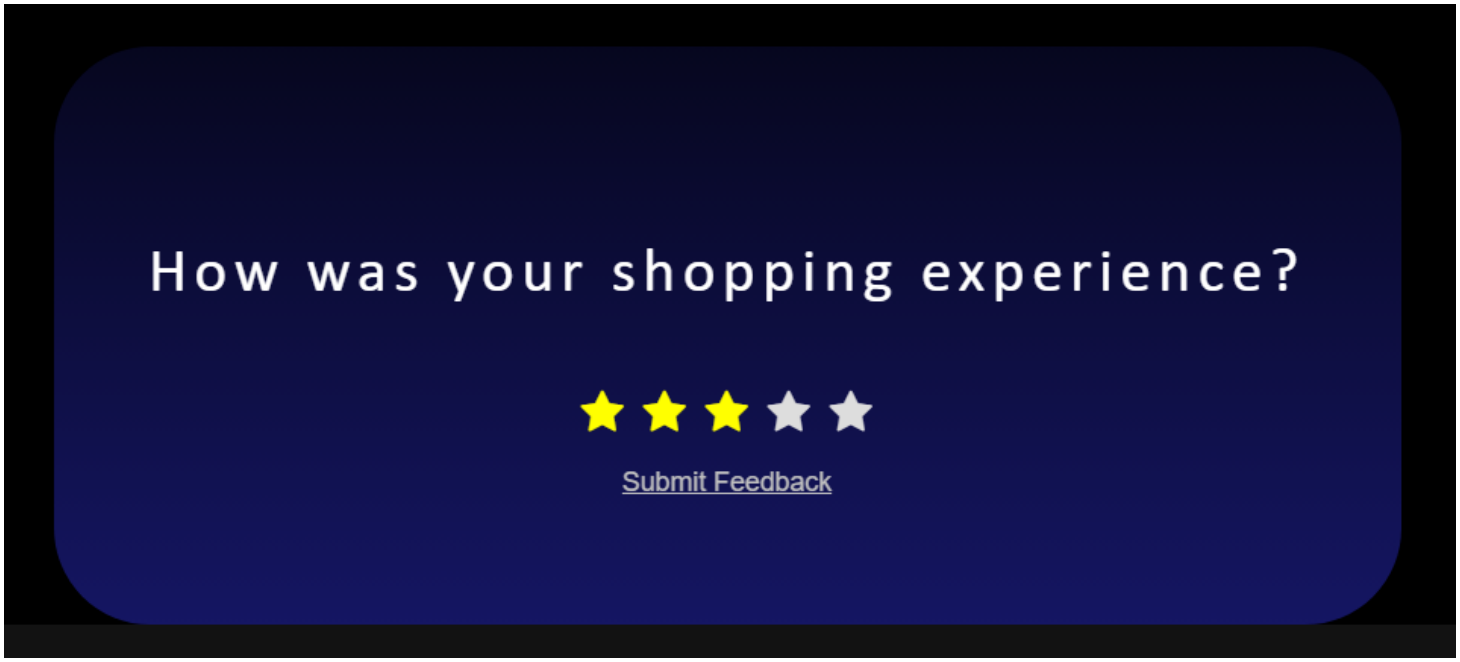
Done by : E. L. Thambawita

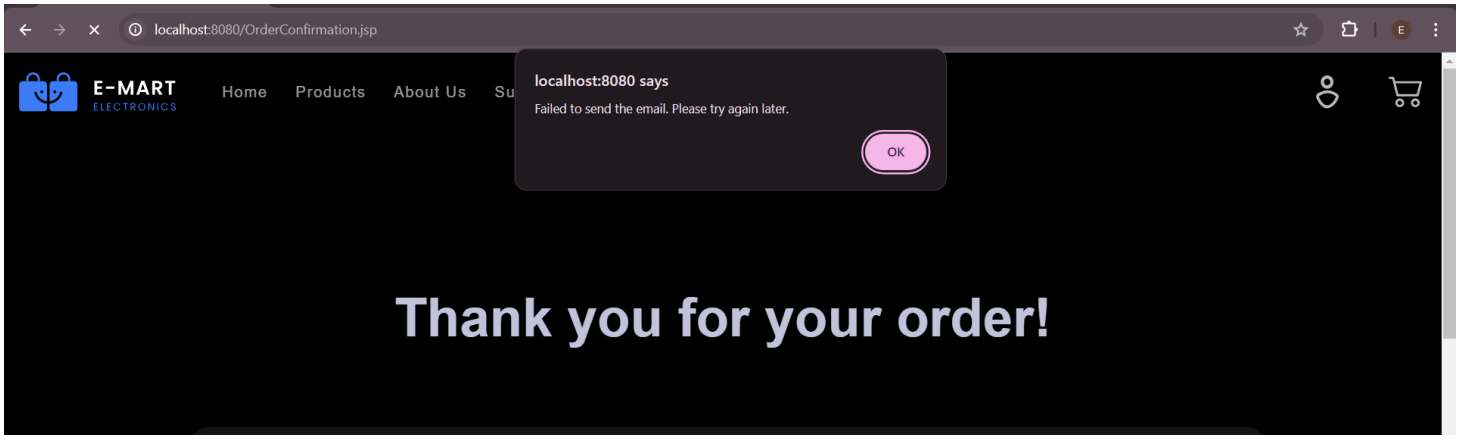
Student ID : 29186

1.2.1 Web page




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


How was your shopping experience?
Submit Feedback


localhost:8080/OrderConfirmation.jsp
E-MART
ELECTRONICS
Home
Products
About Us
localhost:8080 says
Failed to send the email. Please try again later.
OK
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

```jsp
<%--
    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>
```

```jsp
        <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">
```

```
                                    <span class="number">3)</span>
                                    <span class="name">Iphone 14 pro</span>
                                    <span class="amount">3 items</span>
                                    <span class="price">LKR 300,000</span>
                                </div>-->


                    <div class="line"></div>

                    <div class="button-container">
                        <button class="button button1" onclick="window.location.href = 'OrderHistory.jsp';">Order History</button>
                        <button class="button button2" onclick="window.location.href = 'ProductCategory_Phones.jsp';">Continue Shopping
                        <span class="text1">Total -</span>
                        <%
                                int total = 0;
                                for (ShoppingCartObj item : confirmItems) {
                                    total += item.getTotalPrice();
                                }
                        %>
                        <span class="text2">LKR <%= total %></span>
                    </div>


                </div>
            </div>


            <div class="feedback">
                <h3>How was your shopping experience?</h3>

                <!-- Star rating section -->
                <div class="stars">
                    <fieldset class="rating">
```

```
                        <input type="radio" id="star5" name="rating" value="5"/><label for="star5" class="full" title="Awesome"></label
                        <input type="radio" id="star4" name="rating" value="4"/><label for="star4" class="full"></label>
                        <input type="radio" id="star3" name="rating" value="3"/><label for="star3" class="full"></label>
                        <input type="radio" id="star2" name="rating" value="2"/><label for="star2" class="full"></label>
                        <input type="radio" id="star1" name="rating" value="1"/><label for="star1" class="full"></label>
                    </fieldset>
                </div>
                <!-- Submit button -->
                <div>
                    <%
                        List<AdminOrderObj> orders = DAO.getAdminOrders();
                        int lastOrderId = 0; // Initialize with 0 or any other default value
                        int lastIndex = orders.size() - 1;
                        if (lastIndex >= 0) {
                            // Get the orderId of the last order
                            lastOrderId = orders.get(lastIndex).getOrderNumber();
                        }
                    %>

<button type="button" class="submit-feedback" onclick="submitFeedback()" data-orderid="<%= lastOrderId %>">
    Submit Feedback
</button>

                </div>

            </div>


        </div>

```

```
<%String apiKey = "mlsn.1e09da5329b79f474aff9bc3ccb8ddc06df779561f248f8fc202c9764dea4a33";
String senderEmail = "MS_1iz77E@trial-k68zxl2enw9lj905.mlsender.net";
String recipientEmail = "esankilakvindee2000@gmail.com";

String url = "https://api.mailersend.com/v1/email";
String payload = "{\n" +
        "    \"from\": {\n" +
        "        \"email\": \"" + senderEmail + "\"\n" +
        "    },\n" +
        "    \"to\": [\n" +
        "        {\n" +
        "            \"email\": \"" + recipientEmail + "\"\n" +
        "        }\n" +
        "    ],\n" +
        "    \"subject\": \"Hello from MailerSend!\",\n" +
        "    \"text\": \"Greetings from the team, you got this message through MailerSend.\",\n" +
        "    \"html\": \"Greetings from the team, you got this message through MailerSend.\"\n" +
        "}";

try {
    URL obj = new URL(url);
    HttpURLConnection con = (HttpURLConnection) obj.openConnection();

    con.setRequestMethod("POST");
    con.setRequestProperty("Content-Type", "application/json");
    con.setRequestProperty("X-Requested-With", "XMLHttpRequest");
    con.setRequestProperty("Authorization", "Bearer " + apiKey);

    con.setDoOutput(true);
    DataOutputStream wr = new DataOutputStream(con.getOutputStream());
    wr.writeBytes(payload);
    wr.flush();
```

```
    wr.close();

    int responseCode = con.getResponseCode();
    System.out.println("Response Code : " + responseCode);

    BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
    String inputLine;
    StringBuilder MailResponse = new StringBuilder();

    while ((inputLine = in.readLine()) != null) {
        MailResponse.append(inputLine);
    }
    in.close();
} catch (Exception e) {
}%>



<%@ include file="Footer.html" %>
<script src="JS/Common.js"></script>
<script src="JS/Email.js"></script>
<script src="JS/Feedback-Rating.js"></script>
</body>
</html>
```

OrderConfirmation.css

Start Page × DAO.java × OrderConfirmation.jsp [-/M] × Feedback-Rating.js × Email.js × OrderConfirmation.css ×

Source History

```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 11, 2024, 4:43:57 PM
    Author     : Esanki Lakvindee
*/

/*Import styles for stars*/
@import url(https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.css);

body {
  margin: 0;
  padding: 0;
  font-family: 'Inter', sans-serif;
  background-color: #000; /* Updated background color */
}

.container {
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
}

.confirmation {
  text-align: center;
  color: #C2C3DB;
  margin-bottom: 81px;
  margin-top: 124px;
  font-size: 28px;
}

.order-summary {
  width: calc(100% - 254px); /* Adjusted width */
  max-width: 1000px;
  height: 500px;
  margin: 0 127px; /* Added margin */
  border-radius: 20px;
  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,
  box-shadow: 0px 0px 20px 0px rgba(0, 0, 0, 0.25);
  backdrop-filter: blur(50px);
  padding: 60px;
  margin-top: 81px;
  text-align: left;
}

.order-summary h2 {
  color: #FFF;
  font-size: 32px;
  font-weight: 400;
  margin-bottom: 32px;
}

.line{
  width: 970px;
  height: 3px;
  background: rgba(59, 123, 248, 0.40);
  margin-bottom: 32px;
  margin-left: 20;
  margin-right: 20px;


}
```

```css
.item {
  display: flex;
  align-items: center;
  margin-bottom: 32px;
  font-size: 28px;
}

.item span {
  flex: 1;
  text-align: left;
  font-size: 25px;
  color: #FFF;
}

.name{
    margin-right: 300px;
}

.button {
  height: 80.294px;
  flex-shrink: 0;
  border-radius: 20px;
}

button:hover{
    cursor: pointer !important;
}

```

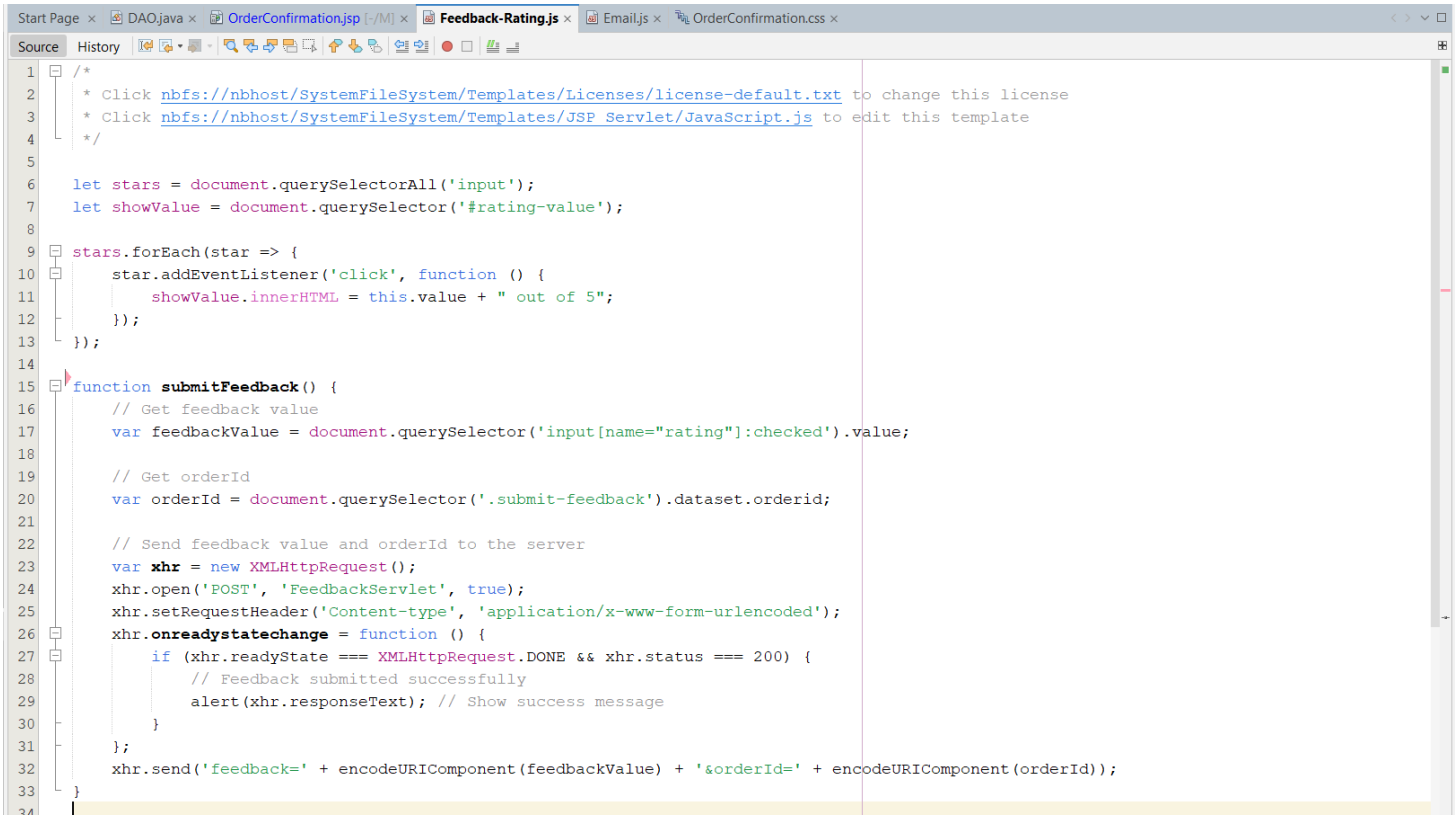
```css
.button1 {
  width: 200px;
  height: 65px; /* Added height */
  flex-shrink: 0;
  border: 1px solid #FFF;
  background: #000;
}

.button2 {
  width: 200px;
  height: 65px;
  flex-shrink: 0;
  border-radius: 20px;
  background: #151663;
  margin-right: 235px;
}

.button1, .button2 {
  color: #FFF;
  font-size: 20px;
}

.text1, .text2 {
  font-size: 25px;
  color: #FFF;
}

.text1{
    margin-right: 105px;
}

```

```css
.feedback {
  width: 700px;
  height: 300px;
  border-radius: 50px;
  background: linear-gradient(180deg, #000 -44.82%, #151663 100%);
  display: flex;
  flex-direction: column;
  justify-content: center;
  align-items: center;
  margin-top: 70px;
}

.feedback h3 {
  color: #FFF;
  font-family: 'Calibri', sans-serif;
  font-size: 32px;
  font-weight: 400;
  letter-spacing: 3.84px;
}

/* stars*/
.rating {
    border: none;
    float: left;
}

.rating > input {
    display: none;
}
```

```css
.rating > label:before {
    content: '\f005';
    font-family: FontAwesome;
    margin: 5px;
    font-size: 1.5rem;
    display: inline-block;
    cursor: pointer;
}

.rating > label {
    color: #ddd;
    float: right;
    cursor: pointer;
}

.rating > input:checked ~ label,
.rating:not(:checked) > label:hover,
.rating:not(:checked) > label:hover ~ label {
    color: yellow;
}

.rating > input:checked + label:hover,
.rating > input:checked ~ label:hover,
.rating > label:hover ~ input:checked ~ label,
.rating > input:checked ~ label:hover ~ label {
    color: yellow;
}
```

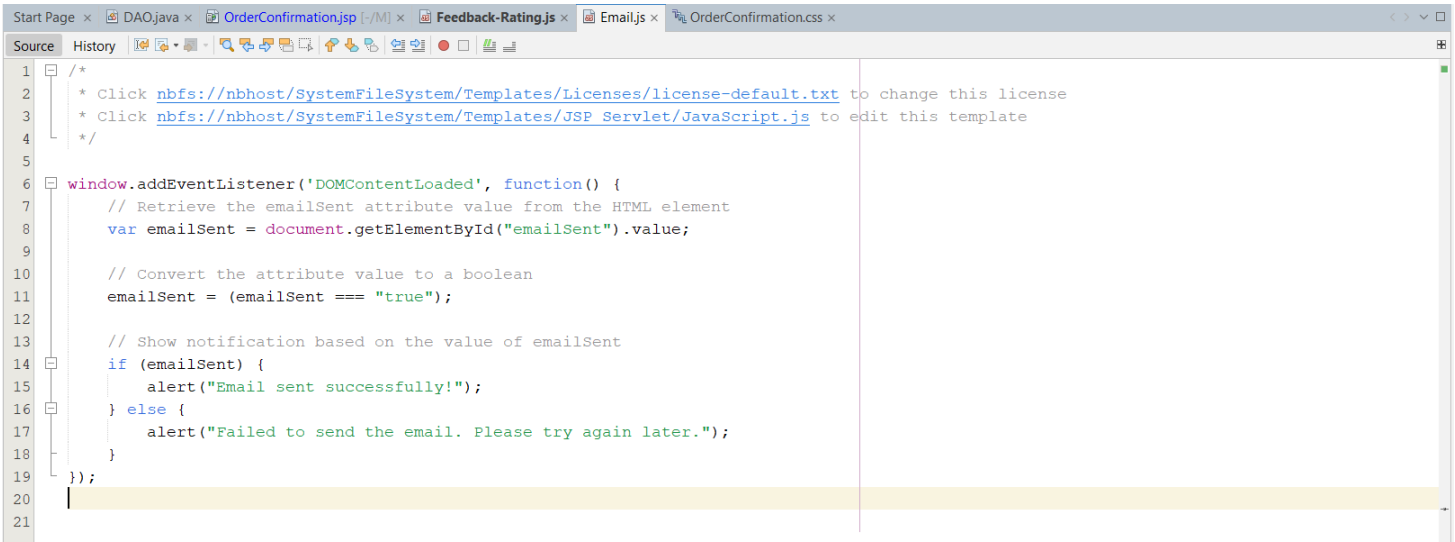
```css
/*submit button*/
.submit-feedback {
    background: none; /* Remove background */
    border: none; /* Remove border */
    color: #C0C0C0; /* Text color */
    font-size: 14px; /* Font size */
    cursor: pointer; /* Cursor on hover */
    text-decoration: underline; /* Underline text */
    transition: color 0.3s; /* Smooth transition for text color */
}

.submit-feedback:hover {
    color:#A9A9A9; /* Darken text color on hover */
}
```

Feedback-Rating.js



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/JavaScript.js to edit this template
 */

let stars = document.querySelectorAll('input');
let showValue = document.querySelector('#rating-value');

stars.forEach(star => {
    star.addEventListener('click', function () {
        showValue.innerHTML = this.value + " out of 5";
    });
});

function submitFeedback() {
    // Get feedback value
    var feedbackValue = document.querySelector('input[name="rating"]:checked').value;

    // Get orderId
    var orderId = document.querySelector('.submit-feedback').dataset.orderid;

    // Send feedback value and orderId to the server
    var xhr = new XMLHttpRequest();
    xhr.open('POST', 'FeedbackServlet', true);
    xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
    xhr.onreadystatechange = function () {
        if (xhr.readyState === XMLHttpRequest.DONE && xhr.status === 200) {
            // Feedback submitted successfully
            alert(xhr.responseText); // Show success message
        }
    };
    xhr.send('feedback=' + encodeURIComponent(feedbackValue) + '&orderId=' + encodeURIComponent(orderId));
}
```

Email.js



```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/JavaScript.js to edit this template
 */

window.addEventListener('DOMContentLoaded', function() {
    // Retrieve the emailSent attribute value from the HTML element
    var emailSent = document.getElementById("emailSent").value;

    // Convert the attribute value to a boolean
    emailSent = (emailSent === "true");

    // Show notification based on the value of emailSent
    if (emailSent) {
        alert("Email sent successfully!");
    } else {
        alert("Failed to send the email. Please try again later.");
    }
});
```

FeedbackServlet.java

Start Page | DAO.java | OrderConfirmation.jsp [-/M] | Feedback-Rating.js [-/M] | Email.js | OrderConfirmation.css | index.jsp | Products.jsp | FeedbackServlet.java [-/M]

Source | History

```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 java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
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 Esanki Lakvindee
 */
@WebServlet(name = "FeedbackServlet", urlPatterns = {"/FeedbackServlet"})
public class FeedbackServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        /*
        // Assuming you have a method to fetch order data from the database
        List<Order> orders = fetchOrdersFromDatabase();

        // Assuming each Order object has a feedback attribute
        // Retrieve feedback data for each order
```

```java
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Retrieve feedback value and orderId from request
        String feedbackValue = request.getParameter("feedback");
        String orderId = request.getParameter("orderId");

        // Insert feedback into the database along with orderId
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            try (Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/emart", "root", "")) {
                PreparedStatement pst = con.prepareStatement("UPDATE orders SET feedback = ? WHERE orderId = ?");
                pst.setString(1, feedbackValue);
                pst.setString(2, orderId);
                pst.executeUpdate();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Send response back to client
        response.setContentType("text/plain");
        PrintWriter out = response.getWriter();
        out.print("Feedback saved successfully");
        out.flush();
    }

}
```

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                 java.util.Date orderDate = resultSet.getDate("orderDate");
504                 String orderStatus = resultSet.getString("orderStatus");
505                 int totalPrice = resultSet.getInt("totalPrice");
506                 String shippingAddress = resultSet.getString("address");
507                 String customerName = resultSet.getString("name");
508                 String feedback = resultSet.getString("feedback");
509                 String email = resultSet.getString("email");
510                 String productName = resultSet.getString("productName"); // Retrieve product name
511 
512                 // Creating Smartphone object and adding it to the list
513                 AdminOrderObj order = new AdminOrderObj(orderDate, orderNumber, orderStatus, productId, quantity,
514                         totalPrice, shippingAddress, customerName, feedback, email);
515                 order.setProductName(productName);
516                 orders.add(order);
517             }
518 
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         return orders;
539     }
```

DAO.java – getOrderConfirmation()

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

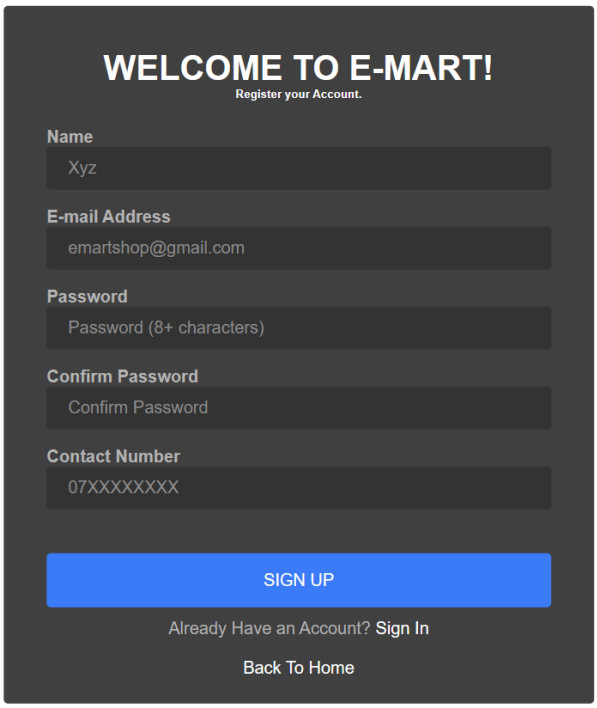
Conclusion

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



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>
```

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

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

    <label>07XXXXXXXX</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);
  }
```

```css
.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;
}
```

```css
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(string: "username");
        String email = request.getParameter(string: "email");
        String password = request.getParameter(string: "password");
        String confirmPassword = request.getParameter(string: "confirmPassword");
        String contactNumber = request.getParameter(string: "ContactNumber");

        // If Validation fails
        if (!validation(username, email, password, confirmPassword, contactNumber)) {
            request.getRequestDispatcher(string: "SignUp.jsp").forward(sr: request, sr1: 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(string: "username", o:username.trim());
            session.setAttribute(string: "email", o:email.trim());
            session.setAttribute(string: "password", o:password.trim());
            session.setAttribute(string: "contactNumber", o:contactNumber.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);
```

```
        }

    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet 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(anObject: confirmPassword)) {
            return false;
        }

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

}
```

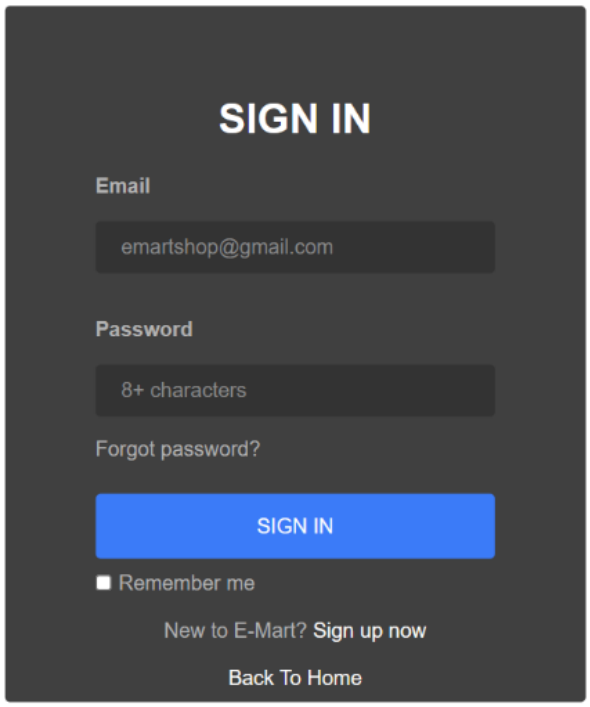
userDAO.java

```
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(x:"Record inserted");
        } catch (SQLException ex) {
            System.out.println(x: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);
        }
    }
```

1.4 Sign In page

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

1.4.1 Web page



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>emartshop@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>

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

```
        <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>
          <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);
}
```

```css
.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%);
}
```

```css
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;
}
```

```css
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;
}
```

```css
@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

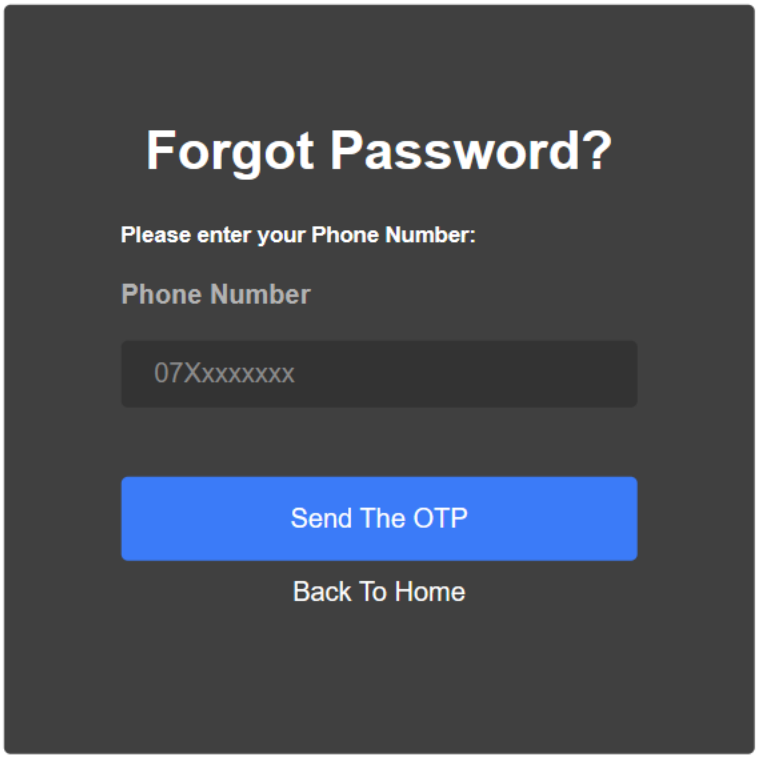
```java
package Controller;

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(string: "email");
        String password = request.getParameter(string: "password");

        // Validate email and password
        if (email == null || email.isEmpty() || password == null || password.isEmpty()) {
            out.println(x:"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(i: 1, string: email.trim());
            statement.setString(i: 2, string: password.trim());

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

ForgetPassword.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>
    <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 OTP</button>
  <p style="text-align: center;"><a href="Home.jsp">Back To Home</a>


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

ForgotPassword.css

```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;
}
```

```css
.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

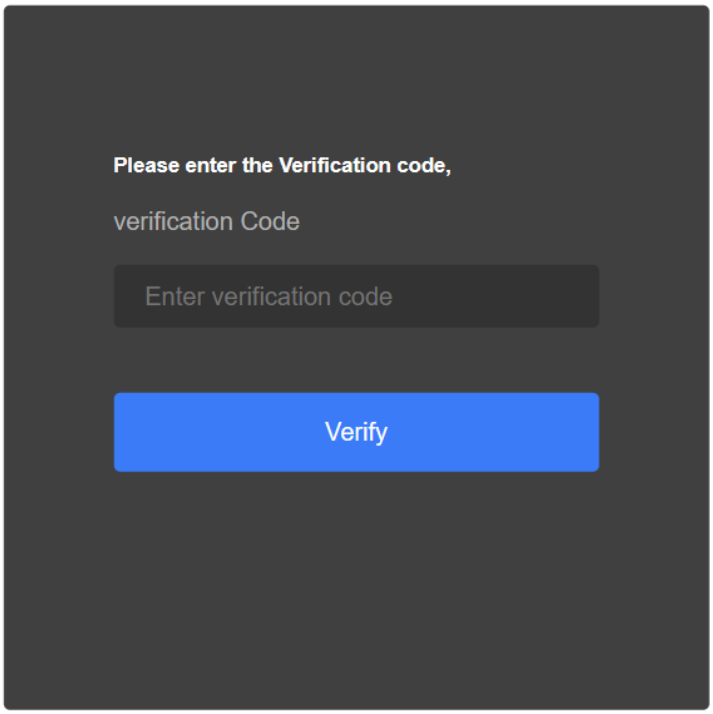
```
package Controller;

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(string: "mobileNumber");

          String API_KEY = "942f32b7";
        String API_SEC = "pkYTozjlXO4qAJeK";
        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(method: "GET");
            StringBuilder apiResponse = new StringBuilder();
            BufferedReader reader = new BufferedReader(new InputStreamReader(in: connection.getInputStream()));
            String line;
            while ((line = reader.readLine()) != null) {
                apiResponse.append(str:line);
            }

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


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

        // Forward the request to verifycode.jsp
        request.getRequestDispatcher(string: "verify.jsp").forward(sr: request, sr1: response);


            } catch (Exception ex) {
                System.out.println(x: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: "\"request_id\":\"(.*?)\"");
            Matcher matcher = pattern.matcher(input: response);
            if (matcher.find()) {
                return matcher.group(group: 1);
            }
            return response;

        }

        /**
         * Returns a short description of the servlet.
         *
         * @return a String containing servlet 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 {
```

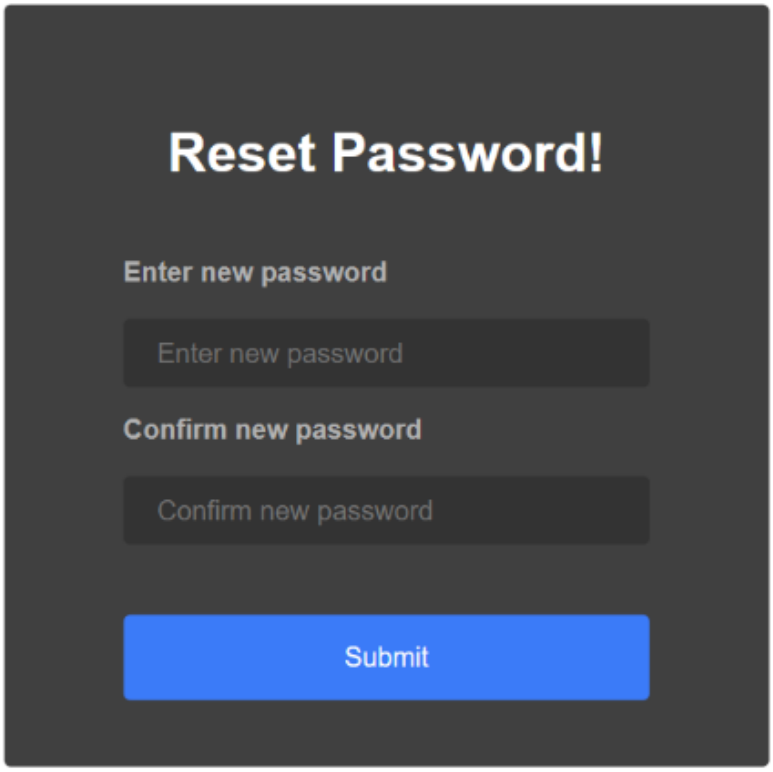
```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //processRequest(request, response);\
        // Get the user's entered OTP from the request
String userEnteredOTP = request.getParameter(string: "otp");

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

// Nexmo API credentials
    String API_KEY = "942f32b7";
    String API_SEC = "pkYTozjIXO4qAJeK";

// 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(spec: URL_TO_VERIFY);
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    connection.setRequestMethod(method: "GET");

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

```
    // Check if the response contains patterns indicating success or failure
    if (apiResponse.toString().contains(s: "\"status\": \"0\"")) {
        // Verification successful
        // Perform the desired action (e.g., allow access, redirect to a success page)
        response.sendRedirect(string: "resetPassword.jsp");
    } else {
        // Verification failed
        // Handle the error (e.g., display an error message, redirect to a failure page)
        response.sendRedirect(string: "verify.jsp");
    }
} catch (Exception ex) {
    System.out.println(x: 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(name: 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);
    // 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(name: 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);
    // 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(name: ChangePasswordController.class.getName()).log(level:Level.SEVERE, msg: null, thrown: ex);
    }


}
```

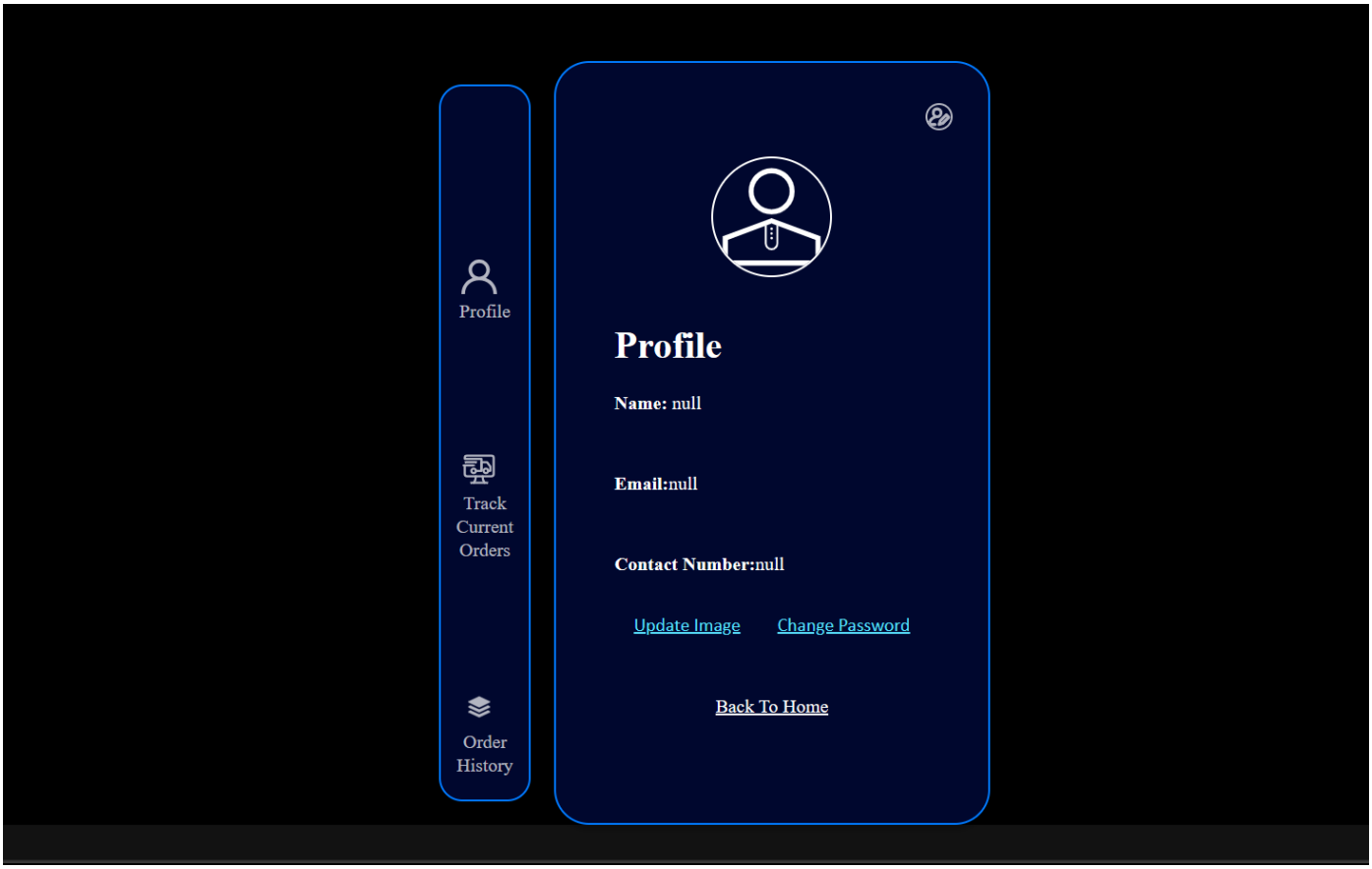
1.8 Profile page

(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">
              <img src="Images/Profile/Profile.svg">
              <span>Profile</span>
            </a>
          </div>
          <div class="sidebar-item">
            <a href="TrackOrder.jsp">
              <img src="Images/Profile/TrackOrder.svg">
              <span>Track Current Orders</span>
            </a>
          </div>
```

```
            <div class="sidebar-item">
              <a href="TrackOrder.jsp">
                <img src="Images/Profile/TrackOrder.svg">
                <span>Track Current Orders</span>
              </a>
            </div>
            <div class="sidebar-item">
              <a href="OrderHistory.jsp">
                <img src="Images/Profile/OrderHistory.svg">
                <span>Order History</span>
              </a>
            </div>
          </div>

          <div class="form-container profile-form" action="ProfilePictureServlet" method="post" enctype="multipart/form-data" style="border: 2px solid #007bff; height: 530px; margin-top: 30px">
            <div class="user-image-container" style="position: relative;">
              <img src="Images/Profile/ProfilePlaceholder.svg" id="profile-pic">
              <a href="EditDetails.jsp">
                <div class="edit-profile-tooltip" style="position: absolute; top: -45px; right: -20px; opacity: 0.7; transition: opacity 0.3s ease;" onmouseover="this.style.opacity = '1';" onmouseout="this.style.opacity = '0.7';">

                  <img src="Images/Profile/EditProfile.svg" alt="Edit Details" class="edit-icon" title="Edit Profile" style="width: 20px; height: 20px; color: #59E3FF;">
                </div>
              </a>
            </div>
            <br>
```

```
            <h1>Profile</h1>
            <%
              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">
              <table>
                <tr>
                  <td>
                    <div class="button-row change-email">
                      <label for="input-file" style="color: #59E3FF; text-decoration: underline; cursor: pointer;">Update Image</label>
                      <input type="file" accept="image/jpeg, image/png, image/jpg" id="input-file" style="display: none;">
                    </div>
                  </td>
                  <td>
                    <a href="ChangePassword.jsp" style="color: #59E3FF;">Change Password</a>
                  </td>
                </tr>
              </table>
            </form>
              <br>
```

```
            <p style="text-align: center;"><a href="index.jsp" style="color: white;">Back To Home</a></p>
            <script>
              window.onload = function () {
                let profilePic = document.getElementById("profile-pic");
                let inputFile = document.getElementById("input-file");
                let chooseFile = document.getElementById("choose-file");

                inputFile.onchange = function () {
                  if (validateFile(inputFile)) {
                    profilePic.src = URL.createObjectURL(inputFile.files[0]);
                  } else {
                    input.value = ""; // Clear the file input field if validation fails
                  }
                };

                document.querySelector('label[for="input-file"]').addEventListener('click', function () {
                  chooseFile.click();
                });
              };

              function validateFile(input) {
                var file = input.files[0];
                var allowedTypes = ["image/jpeg", "image/png", "image/jpg"]; // Allowed image types

                if (file && !allowedTypes.includes(file.type)) {
                  // File is not an image, show an error message
                  alert("Please select a valid image file (JPEG, PNG, JPG).");
                  return false;
                }
                return true;
              }
            </script>
          </div>
        </div>
      </div>
      <%@ include file="Footer.html" %>
      <script src="JS/Common.js"></script>
    </body>
  </html>
```

Profile.css

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

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

.sidebar-item img {
  width: 30px; /* Adjust the width as needed */
  height: 30px; /* Adjust the height as needed */
  margin-right: 10px; /* Optional: Add some spacing between the icon and text */
}

/* styles for sidebar item text */
.sidebar-item span {
  vertical-align: middle; /* Align the text vertically with the icon */
}

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

```css
.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: #ffffff; /* 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: 20px 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: #ffffff; /* Change font color */
}
```

```css
.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;
}
```

```css
.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 */
}
```

ProfileServlet.java

```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(string: "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(string: "user", o:user);

        // Forward the request to the profile JSP
        request.getRequestDispatcher(string: "Profile.jsp").forward(sr: request, sr1: 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

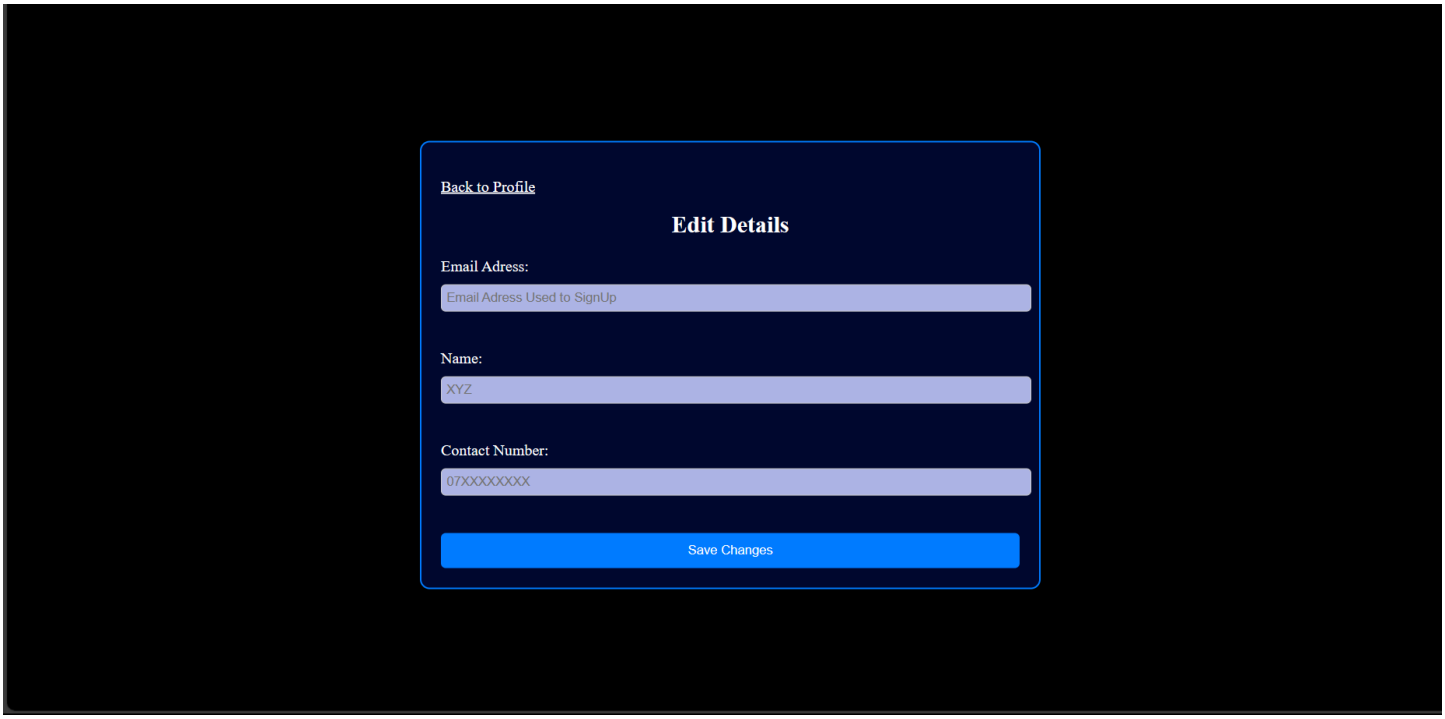
```
    public SignUpUser getUserByEmail(String email) {
        connectToDB();
        String query = "SELECT * FROM users WHERE email = '" + email + "'";
        ResultSet rs = null;
        SignUpUser user = null;

        try {
            rs = st.executeQuery(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(x:ex.getMessage());
        } finally {
            try {
                if (rs != null) {
                    rs.close();
                }
            } catch (SQLException ex) {
                System.out.println(x: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(string: "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(b:buffer)) != -1) {
        output.write(b:buffer, off:0, len: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(string: "profilePicture", o:fileName);

    // Redirect back to the profile page or any other desired page
    response.sendRedirect(string: "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

```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;
}
```

```css
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: 20px 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);
}
```

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

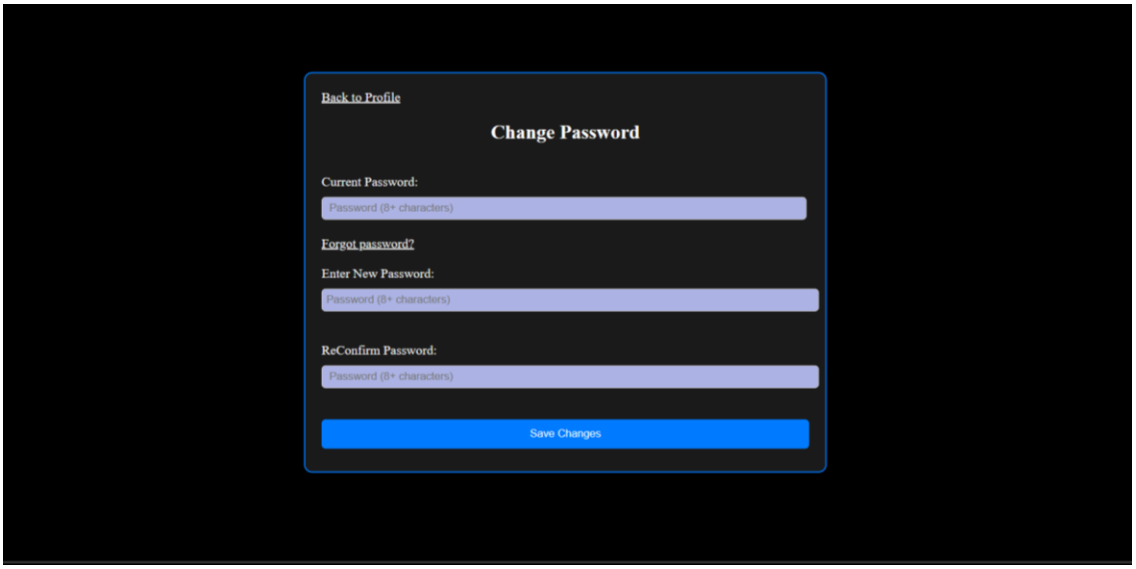
```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //processRequest(request, response);
    // Retrieve form data
    String email = request.getParameter(string: "email");
    String newusername = request.getParameter(string: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

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

    response.sendRedirect(string: "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(string: checkQuery);
        checkStatement.setString(i: 1, string: email);
        rs = checkStatement.executeQuery();
        emailExists = rs.next(); // Check if there's a result
    } catch (SQLException ex) {
        System.out.println(x:ex.getMessage());
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException ex) {
            System.out.println(x: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(i: 1, string: newusername);
            updateStatement.setString(i: 2, string: newcontactNumber);
            updateStatement.setString(i: 3, string: email);
            updateStatement.executeUpdate();
            System.out.println(x:"Details updated successfully.");
        } catch (SQLException ex) {
            System.out.println(x:ex.getMessage());
        }
    } else {
        System.out.println(x:"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

```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;
}
```

```css
.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

```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(name: ChangePasswordController.class.getName()).log(level: Level.SEVERE, msg: null, thrown: ex);
        }
    }
```

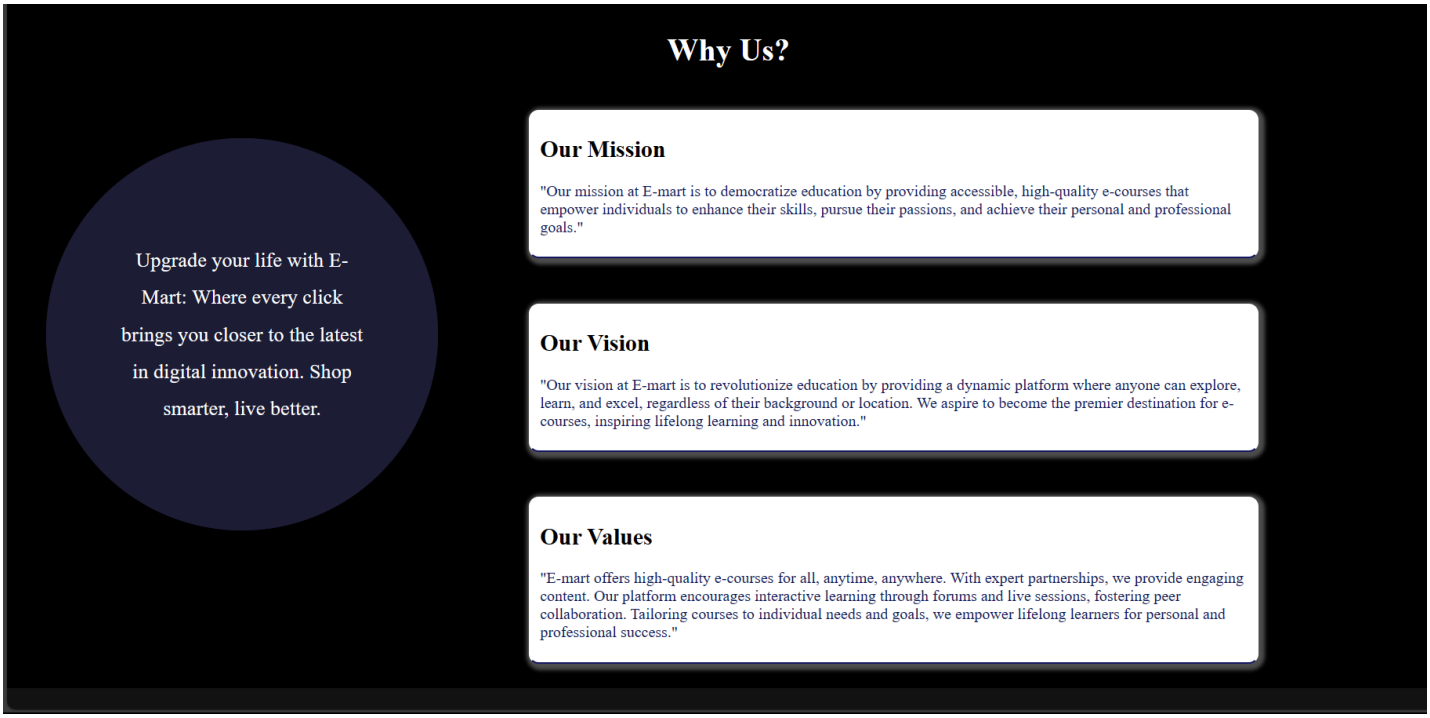
```
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    //processRequest(request, response);
    response.setContentType(string: "text/html");
    PrintWriter out = response.getWriter();


    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(name: 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



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

AboutUs.css

```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;
}
```

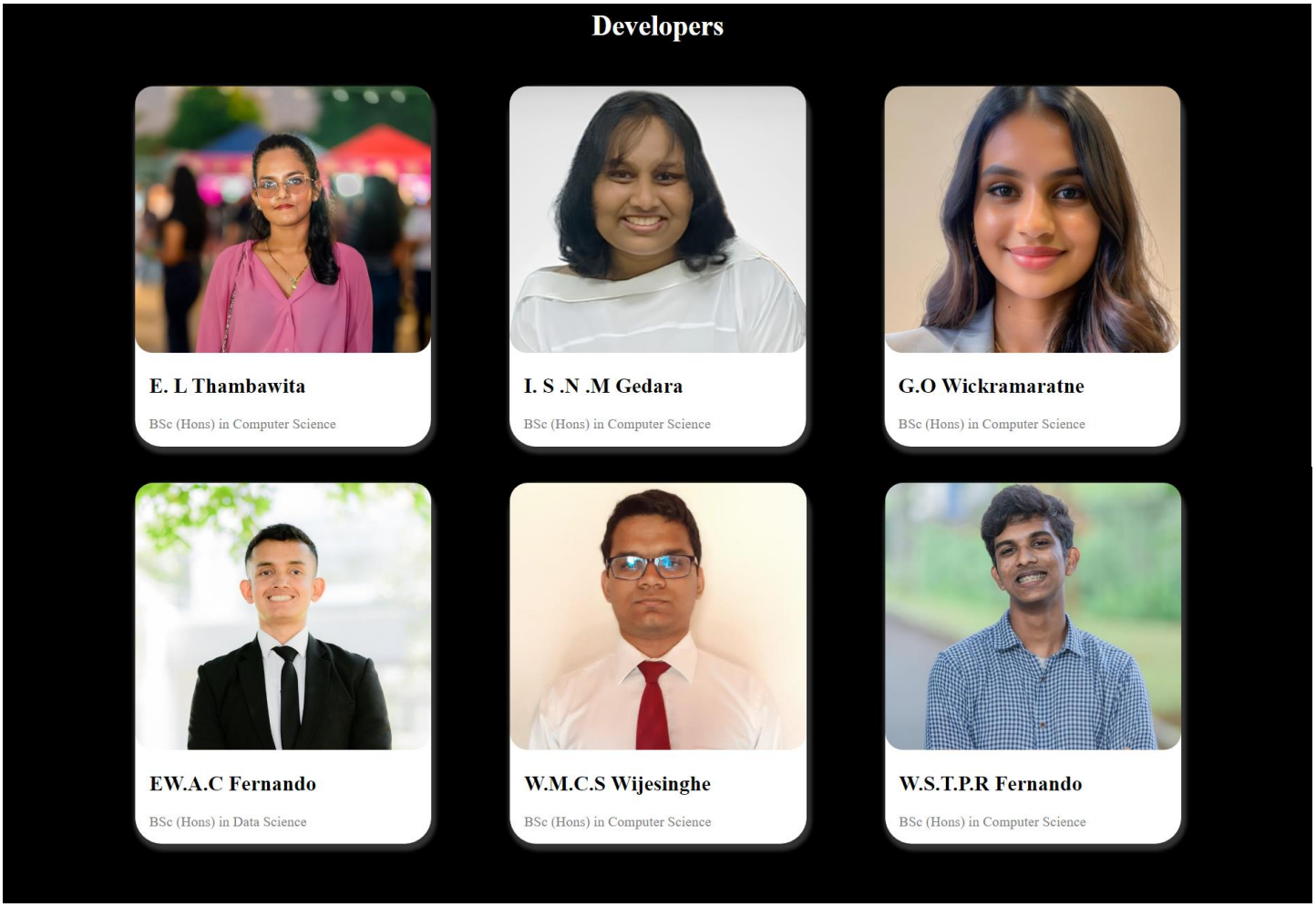
```css
.container-right {
  width: 55%;
}

.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;
}
```

```css
.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



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">
            <img src="Images/Developers/Esanki.jpg" alt="E. L Thambawita" style="width:100%"></a>
          <div class="container">
            <h2>E. L Thambawita</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/imalsha-sandali-54b01326a">
            <img src= "Images/Developers/sandali.jpg"  alt="I. S .N .M Gedara" style="width:100%"></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">
            <img src= "Images/Developers/oneli.jpg"  alt="G.O Wickramaratne" style="width:100%"></a>
          <div class="container">
            <h2>G.O Wickramaratne</h2>
            <p class="title">BSc (Hons) in Computer Science</p>
          </div>
        </div>
      </div>
    </div>
    <!--Second Row-->
    <div class="row">
      <div class="column">
        <div class="card">
          <a href="https://www.linkedin.com/in/adheesha-fernando-5b8975230">
            <img src= "Images/Developers/Adeesha.jpg"  alt="EW.A.C Fernando" style="width:100%"></a>
          <div class="container">
            <h2>EW.A.C Fernando</h2>
            <p class="title">BSc (Hons) in Data Science</p>
          </div>
        </div>
      </div>
```

```
        <div class="column">
          <div class="card">
            <a href="https://www.linkedin.com/in/chanuka-wijesinghe-83968825a?trk=blended-typeahead">
              <img src= "Images/Developers/Chanuka.jpg"  alt="W.M.C.S Wijesinghe" style="width:100%"></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">
              <img src= "Images/Developers/Rovin.jpg"  alt="W.S.T.P.R Fernando" style="width:100%"></a>
            <div class="container">
              <h2>W.S.T.P.R Fernando</h2>
              <p class="title">BSc (Hons) in Computer Science</p>
            </div>
          </div>
        </div>
      </div>
      <%@ include file="Footer.html" %>
      <script src="JS/Common.js"></script>
    </body>

</html>
```

Developers.css

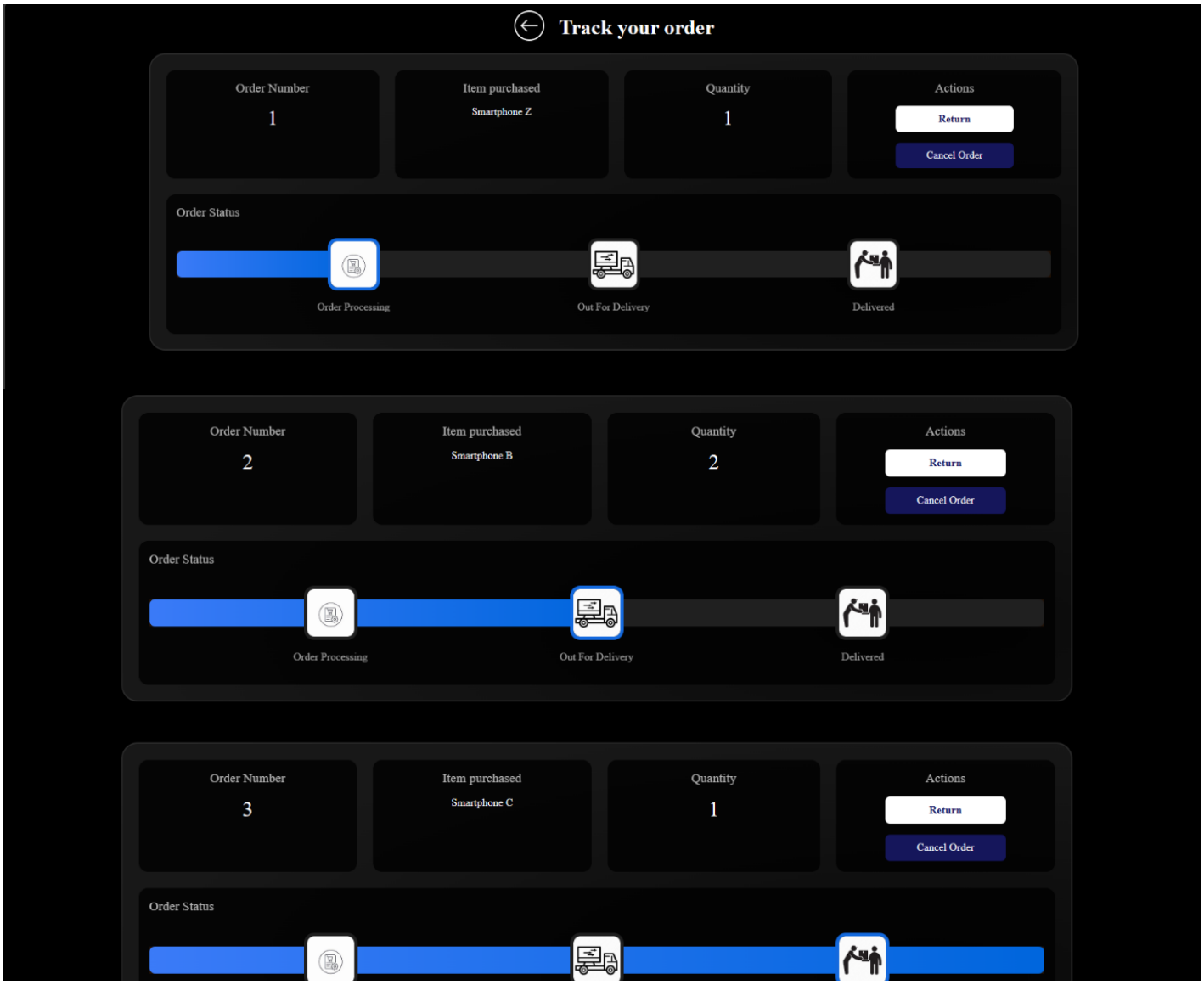
```
footer{
    margin-top: 100px;
}

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:#fff;
}

.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



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 .
- 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;">
        <img class="returnButton" src="Images/TrackOrder/returnButton.png" alt="Return" style="cursor: pointer;">
      </a>

      <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>
          <%= order.getProductName() %>
          <div class="order-detail-col-content">

          </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 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>

          <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>
```

```css
.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;
}
```

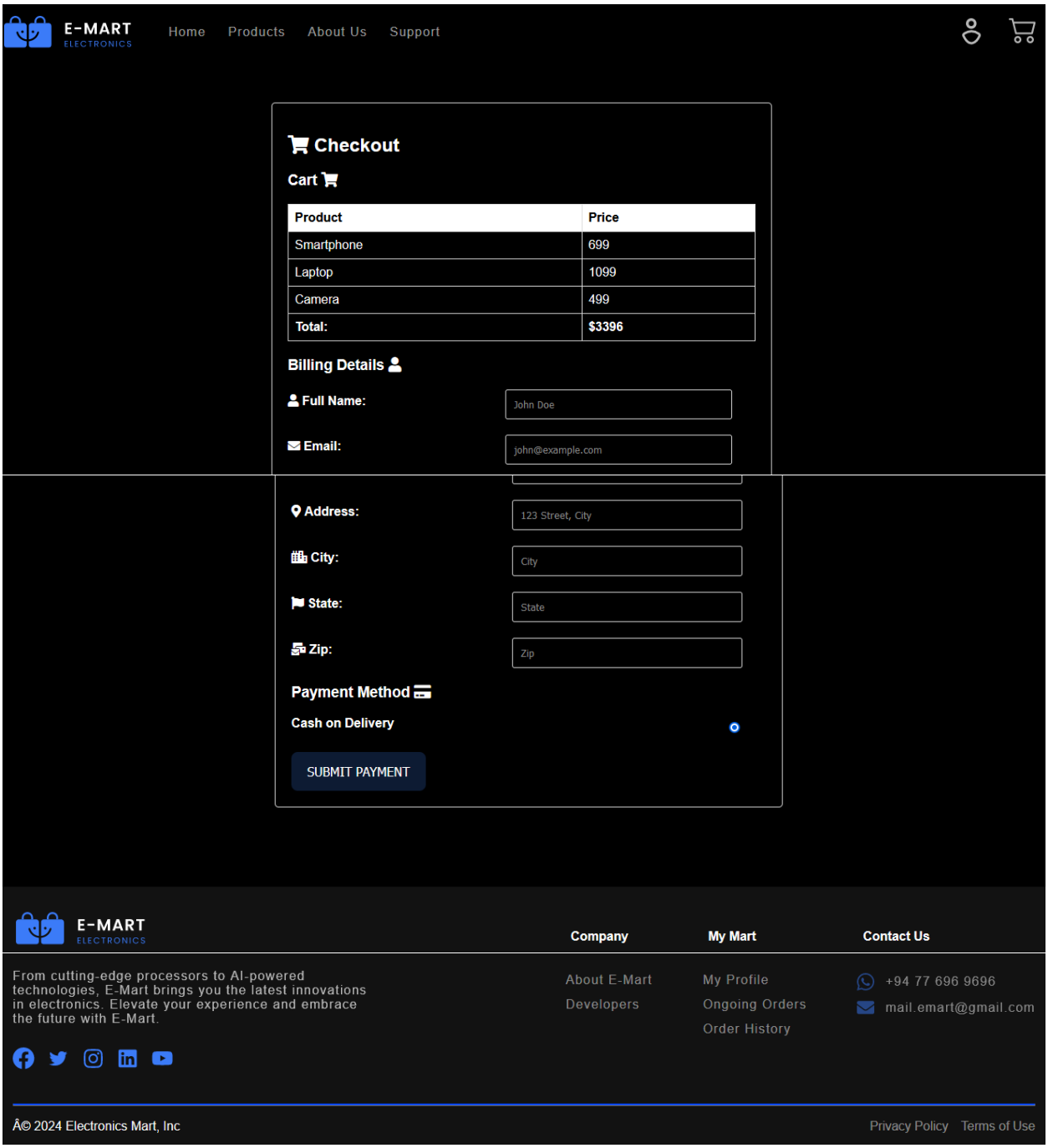
```css
.order-details-col p {
  font-size: xx-large;
}

.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: 000px 15%;
  margin-top: -60px;
  bottom: 0;
  display: flex;
  justify-content: space-between;
}
```

```css
.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



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

```jsp
<%--
    Document   : Checkout
    Created on : Apr 12, 2024, 10:05:16?AM
    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 -->
            <h3>Payment Method <i class="fas fa-credit-card"></i></h3>
            <div class="form-group">
                <label for="cashOnDelivery">Cash on Delivery</label>
                    <input type="radio" id="cashOnDelivery" name="paymentMethod" value="cod" checked>
                </div>

                <input type="submit" value="Submit Payment" class="btn btn-darkblue">
            </form>
        </div>



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

Checkout.css

```
    */

body {
    font-family: Arial, sans-serif;
    background-color: #000000;
    color: #ffffff;
}

.container {
    background-color: #000000;
    border: 1px solid #dddddd;
    border-radius: 5px;
    padding: 20px;
    margin: 20px auto;
    max-width: 600px;
    margin-bottom: 100px;
    margin-top: 50px;

}

h2, h3 {
    color: #ffffff;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
}
```

```css
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,
```

```css
.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: #0d1b33;
    color: white;
    padding: 15px 20px;
    border: none;
    border-radius: 8px;
    cursor: pointer;
    font-size: 16px;
    text-transform: uppercase;
}

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

```css
.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

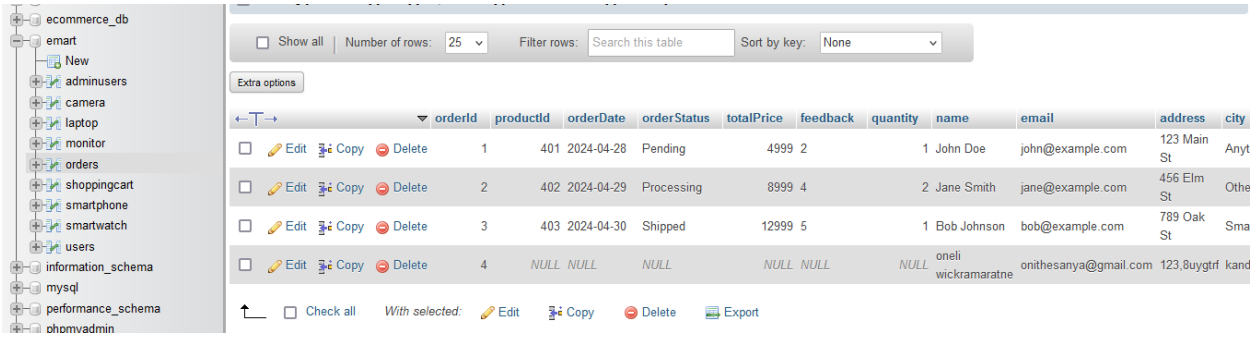
```
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String name=request.getParameter("fullName");
        String mail=request.getParameter("email");
        String add=request.getParameter("address");
        String citys=request.getParameter("city");
        String states=request.getParameter("state");
        int zips=Integer.parseInt(request.getParameter("zip"));

        Checkout c= new Checkout();
         c.addCheckout(name,mail,add,citys,states,zips);

         // Redirect to orderConfirmation.jsp
    response.sendRedirect("OrderConfirmation.jsp");

       // processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
```

Checkout.java

```
 */
package Model;

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 hp
 */
public class Checkout {
      Statement st;
    public void addCheckout(String name, String mail, String add, String citys, String states, int zips) {
       connectToDB();
        String query = "INSERT INTO orders(name,email,address,city,state,zip) VALUES('"+name+"','"+mail+"','"+add+"','"+citys+"','"+states+"','"+zips+"')";
        try {
            st.executeUpdate(query);
        } catch (SQLException ex) {
          Logger.getLogger(Checkout.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void connectToDB() {
        String driver = "com.mysql.cj.jdbc.Driver";
```

```
            String driver = "com.mysql.cj.jdbc.Driver";
            String url = "jdbc:mysql://localhost:3306/emart";

            try {
                Class.forName(driver);
                Connection con = DriverManager.getConnection(url, "root", "");
                st = con.createStatement();

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


        }

    }
```

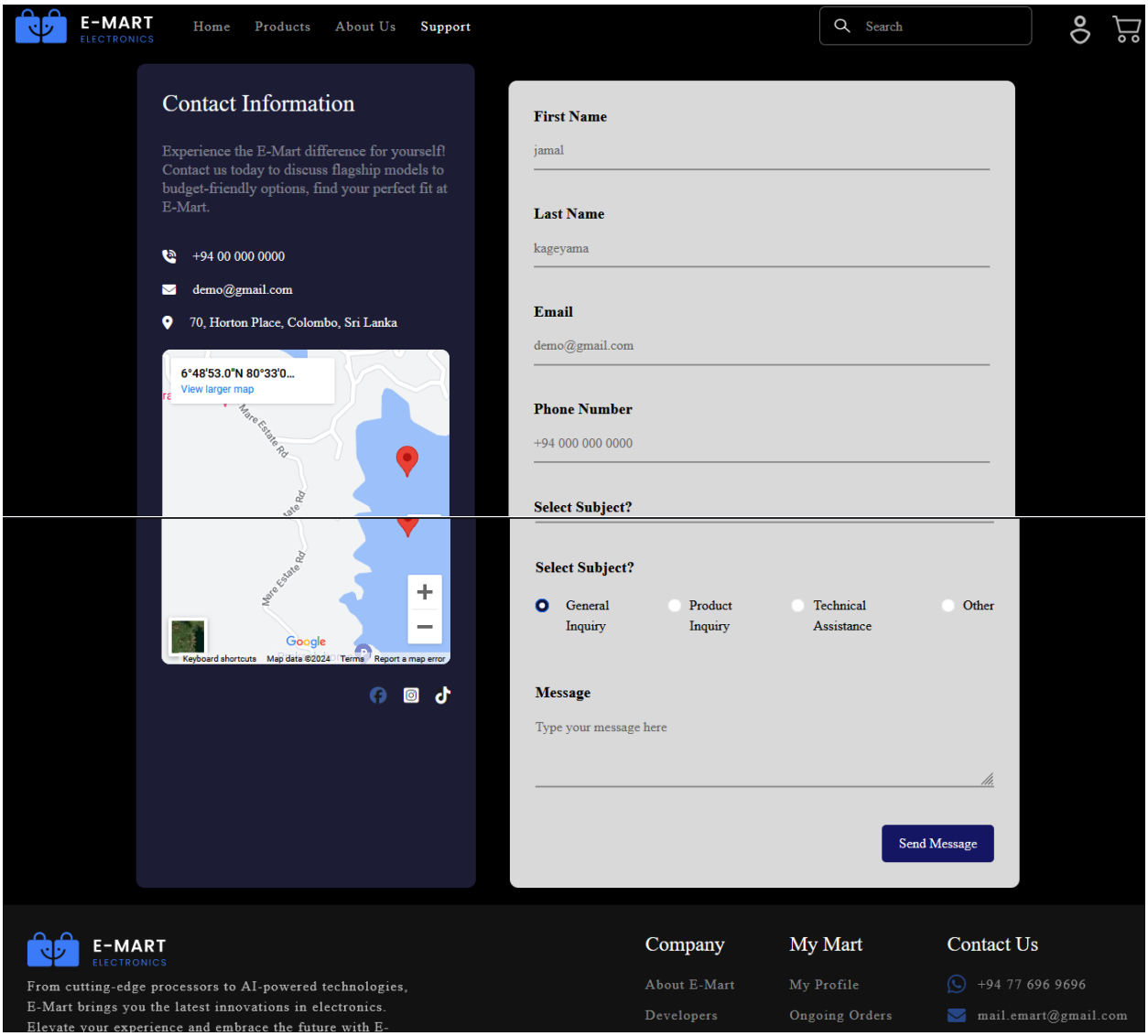
Values being entered to the database.



1.13 Contact page

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

1.13.1 Web page



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

```jsp

<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Contact Page</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/Contact.css" rel="stylesheet" type="text/css"/>
        <script src="https://kit.fontawesome.com/7942e54de0.js" crossorigin="anonymous"></script>
    </head>
    <%@ include file="Navbar.html" %>
    <body>
        <div class="wrapper-center contact-page-content">
            <img src="/letter_send 1.png" alt="" class="bg-graphic">
            <div class="content-cards">
                <div class="content-card bg-accent-2">
                    <h3 class="c-card-tittle">Contact Information</h3>
                    <p class="description">Experience the E-Mart difference for yourself! Contact us today to discuss flagship models to budget-friendly op
                    <div class="contactlinks">
                        <div class="contact-link">
                            <i class="fa-solid fa-phone-volume"></i>
                            <span class="c-link-text">+94 00 000 0000</span>
```

```
        </div>
        <div class="contact-link">
            <i class="fa-solid fa-envelope"></i>
            <span class="c-link-text">demo@gmail.com</span>
        </div>
        <div class="contact-link">
            <i class="fa-solid fa-location-dot"></i>
            <span class="c-link-text">70, Horton Place, Colombo, Sri Lanka</span>
        </div>
    </div>
    <div class="map">
        <iframe src="https://www.google.com/maps/embed?pb=!1m17!1m12!1m3!1d3961.630551289352!2d80.54815007589576!3d6.814710993182925!2m3!1f0!2f0!3f0!3m2!1i1
            width="400" height="225" style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade"></iframe>
    </div>

    <div class="socialButtons">
        <i class="fa-brands fa-facebook"></i>
        <i class="fa-brands fa-square-instagram"></i>
        <i class="fa-brands fa-tiktok"></i>
    </div>
</div>

<div class="c-contact-form">
    <form action="" class="contact-form">
        <div class="contact-form-row">
            <div class="contact-input-area">
                <label for="firstName" class="form-label">First Name</label>
                <input type="text" class="contact-form-control" id="firstName" placeholder="jamal">
            </div>
```

```
            <div class="contact-input-area">
                <label for="lastName" class="form-label">Last Name</label>
                <input type="text" class="contact-form-control" id="lastName" placeholder="kageyama">
            </div>
        </div>
        <div class="contact-form-row">
            <div class="contact-input-area">
                <label for="email" class="form-label">Email</label>
                <input type="email" class="contact-form-control" id="email" placeholder="demo@gmail.com">
            </div>
            <div class="contact-input-area">
                <label for="phoneNumber" class="form-label">Phone Number</label>
                <input type="text" class="contact-form-control" id="phoneNumber" placeholder="+94 000 000 0000">
            </div>
        </div>
        <div class="contact-form-row">
            <div class="contact-input-area">
                <label for="subject" class="form-label">Select Subject?</label>
                <div class="subjects">
                    <div class="radio-input-area">
                        <input class="form-check-input" type="radio" checked name="gridRadios" id="gridRadios" value="option2">
                        <span class="checkmark"></span>
                        <label class="form-check-label" for="gridRadios">General Inquiry</label>
                    </div>
                    <div class="radio-input-area">
                        <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios1" value="option2">
                        <label class="form-check-label" for="gridRadios1">Product Inquiry</label>
                    </div>
```

```
                        <label class="form-check-label" for="gridRadios1">Product Inquiry</label>
                    </div>
                    <div class="radio-input-area">
                        <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios2" value="option2">
                        <label class="form-check-label" for="gridRadios2">Technical Assistance</label>
                    </div>
                    <div class="radio-input-area">
                        <input class="form-check-input" type="radio" name="gridRadios" id="gridRadios3" value="option2">
                        <label class="form-check-label" for="gridRadios3">Other</label>
                    </div>
                </div>
            </div>
        </div>
        <div class="contact-form-row">
            <div class="contact-input-area">
                <label for="message" class="form-label">Message</label>
                <textarea class="contact-form-control" id="message" rows="3" placeholder="Type your message here"></textarea>
            </div>
        </div>
        <div class="contact-form-row c-action">
            <button id="sendMessageBtn" type="button" class="btn primary-btn order-btn">Send Message</button>
        </div>
    </form>
</div>
v>

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

```
        </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:oneliwickramaratne@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;
    bottom: 0;
```

```css
.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;
```

```css
.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  }
```

```css
.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;
```

```css
.c-action {
    width: 100%;
    display: flex;
    align-items: flex-end;
    justify-content: flex-end;
}

.radio-input-area input:checked {
    background-color: #15133a;
}
/* Add this CSS for the button styling */
.primary-btn {
    background-color: #151663; /* Dark blue background color */
    color: white; /* White text color */
    border: none;
    padding: 10px 20px;
    border-radius: 5px;
    cursor: pointer;
    transition: background-color 0.3s; /* Smooth transition on hover */
}

.primary-btn:hover {
    background-color: transparent; /* Change background color to transparent on hover */
    color: #151663; /* Change text color to dark blue on hover */
    border: 2px solid #151663; /* Add border on hover */
}

.primary-btn:focus {
    outline: none; /* Remove outline on focus */
}
```

```css
/* CSS for responsiveness */
/* Your existing CSS rules */

.socialButtons i {
    background-color: #1B1B1B;
    font-size: 20px;
    border-radius: 50%;
    color: white; /* Set the icon color to white */
}

/* Add this CSS to specifically target the Facebook icon */
.socialButtons .fa-facebook {
    color: #3b5998; /* Facebook blue color */
}
/* Set the color of phone, email, and location icons to white */
.contact-link i {
    color: white;
}

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

1.14 Product Category page

Done by : W.M.C.S.Wijesinghe
Student ID: 30143

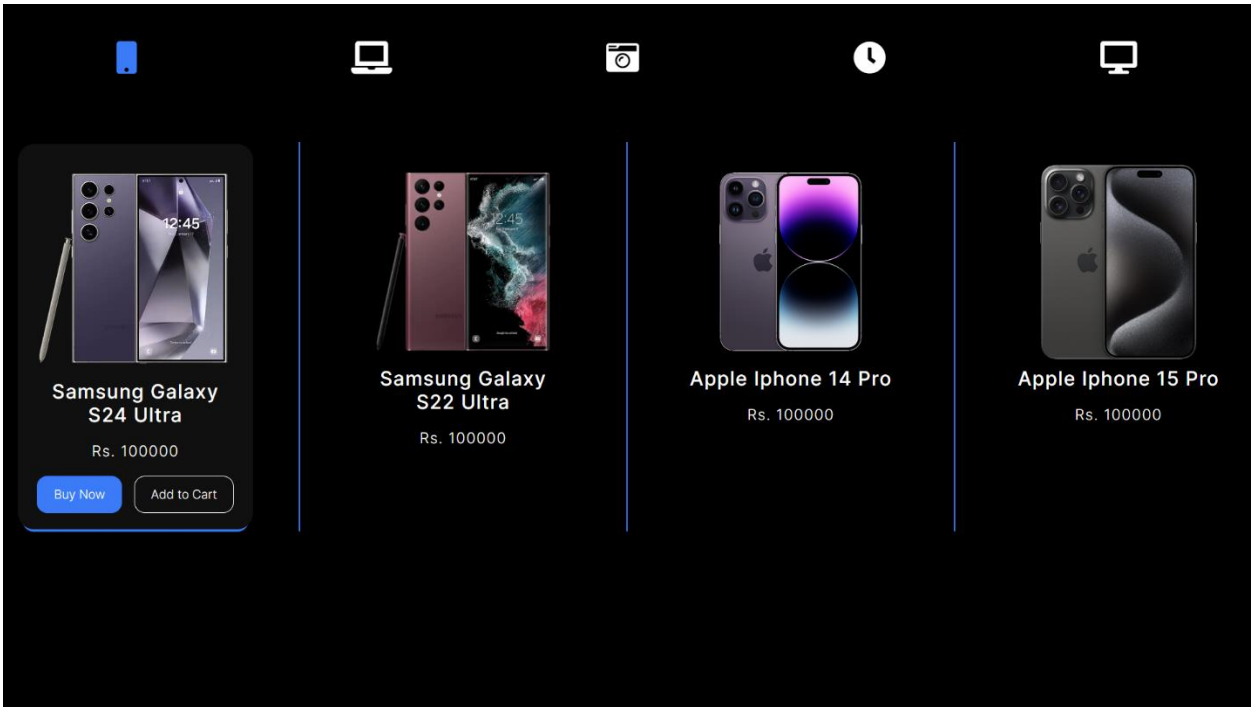


Figure 1: Front End Design for the Product Category Page

- 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.

```html
<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

```css
/*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.

```html
<div class="card">
    <form id="productForm" action="ProductServlet" method="get">
        <input type="hidden" name="id" value="1">
        <a href="#" onclick="document.getElementById('productForm').submit();">
            <img src="Images/ProductCategory&Details/Phones/s24u.webp" alt="product 1">
        </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 Javasctipt 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( s: "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

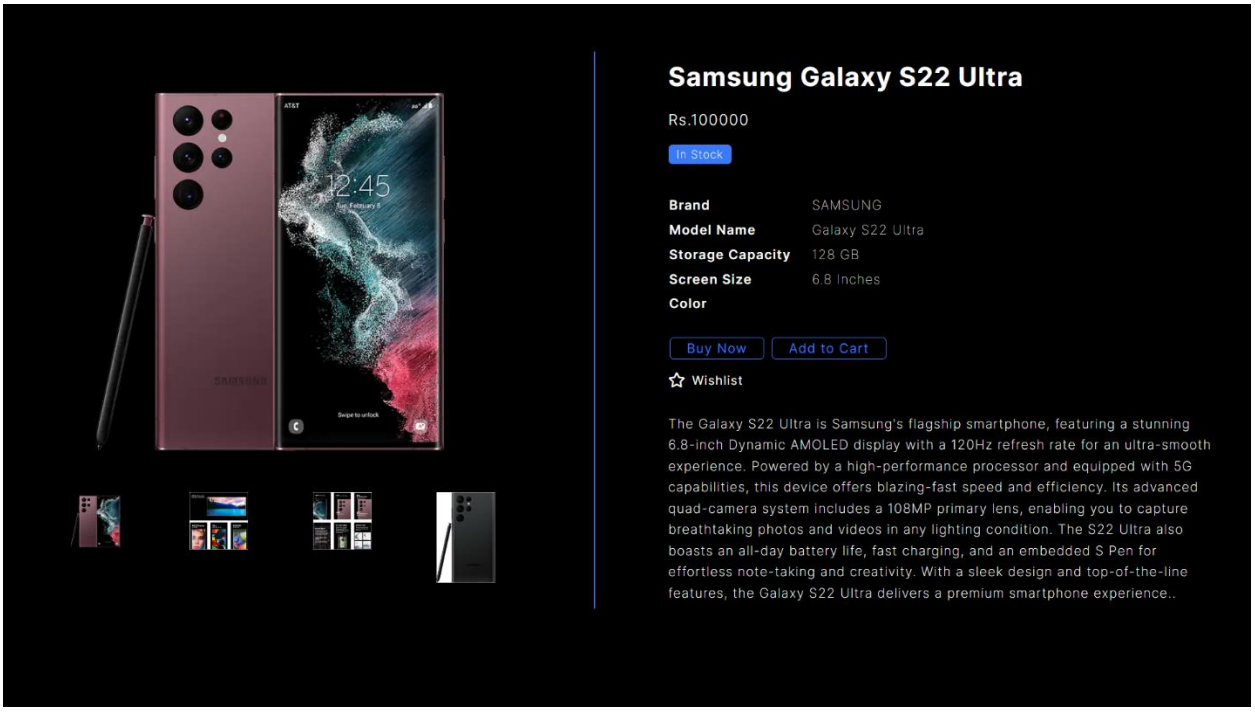
- Above is the opening code snippet to the ProductInfoServlet. The ProductInfoServlet will connect to the database using JDBC.
- Then it runs SQL queries to retrieve all the necessary data from the database. The productID is used as the identifier.

```
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




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

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">
    <img src="<%= photo1 %>" class="thumbnail" id="ProductImg">

    <div class="small-img-row">
        <div class="small-img-col">
            <img src="<%= photo1 %>" width="100%" class="small-img">
        </div>
        <div class="small-img-col">
            <img src="<%= photo2 %>" width="100%" class="small-img">
        </div>
        <div class="small-img-col">
            <img src="<%= photo3 %>" width="100%" class="small-img">
        </div>
        <div class="small-img-col">
            <img src="<%= photo4 %>" width="100%" class="small-img">
        </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 <img> tags with the src attribute set to the value of photos.

```
<!--Right product info and stuff-->
<div class="right-column">
    <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( s: "ID");
        String action = request.getParameter( s: "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, msg: 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");
```

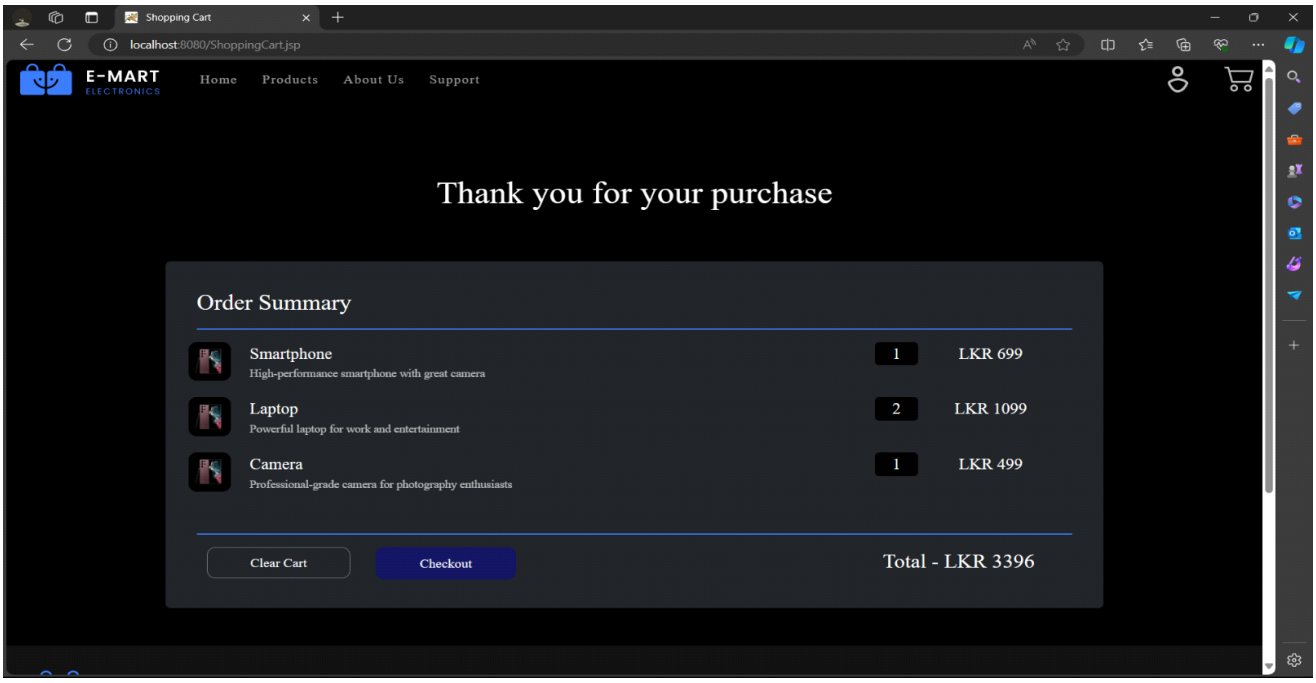
Figure 13: Code snippet to retrieve data from the table and update table

- 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

```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>
```

```jsp
  <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>
          <hr>
          <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="varient"><%= 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 qty-remove"></i>
                                    </div>-->
                <span class="qtyAmount"><%= item.getQuantity() %></span>
                <!--                //      <div class="qty">
                                      <i class="fa-solid fa-add button-qty bg-accent qty-add"></i>
                                    </div>-->
              </div>

              <div class="col col-lg-2">
                LKR <%= item.getProductPrice() %>
              </div>
            </div>
            <% } %>
          </div>

          <hr>
```

```jsp
          <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>
      </div>
    </div>
```

```
    <%@ 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;
}
```

```css
.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;
}
```

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

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

.bg-accent{
  background-color: #3B7BF8;
}

.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;
}
```

```css
.varient{
  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
        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(e: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(e: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(e: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 Smartphone 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(e: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(e: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(string: "orderId");
            int productId = resultSet.getInt(string: "productId");
            int quantity = resultSet.getInt(string: "quantity");
            java.util.Date orderDate = resultSet.getDate(string: "orderDate");
            String orderStatus = resultSet.getString(string: "orderStatus");
            int totalPrice = resultSet.getInt(string: "totalPrice");
            String shippingAddress = resultSet.getString(string: "address");
            String customerName = resultSet.getString(string: "name");
            String feedback = resultSet.getString(string: "feedback");
            String email = resultSet.getString(string: "email");
            String productName = resultSet.getString(string: "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(e: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(string: "DELETE FROM orders WHERE orderId = ?")) {
        statement.setInt(i: 1, i1: 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(string: sql);
```

```
            // Set parameters
            statement.setInt(i: 1, i1: quantity);
            statement.setDouble(i: 2, d: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: screenSize);
            statement.setString(i: 9, string: color);
            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, 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(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 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(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 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(i: 1, string: productId);
        statement.setString(i: 2, string: productName);
        statement.setString(i: 3, string: category);
        statement.setInt(i: 4, i1: quantity);
        statement.setInt(i: 5, i1: price);
        statement.setString(i: 6, string: photo1);
        statement.setString(i: 7, string: photo2);
        statement.setString(i: 8, string: photo3);
        statement.setString(i: 9, string: photo4);
        statement.setString(i: 10, string: brand);
        statement.setString(i: 11, string: modelName);
        statement.setString(i: 12, string: productDescription);
        statement.setString(i: 13, string: storageCapacity);
        statement.setString(i: 14, string: screenSize);
        statement.setString(i: 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, i1: 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, i1:  quantity);
        statement.setDouble(i: 2, d: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, thrown: 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( i: 1, string: productId);
        statement.setString( i: 2, string: productName);
        statement.setString( i: 3, string: category);
        statement.setInt( i: 4, i1: quantity);
        statement.setInt( i: 5, i1: price);
        statement.setString( i: 6, string: photo1);
        statement.setString( i: 7, string: photo2);
        statement.setString( i: 8, string: photo3);
        statement.setString( i: 9, string: photo4);
        statement.setString( i: 10, string: brand);
        statement.setString( i: 11, string: modelName);
        statement.setString( i: 12, string: productDescription);
        statement.setString( i: 13, string: storageCapacity);
        statement.setString( i: 14, string: cpu);
        statement.setString( i: 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(i: 1, i1: quantity);
        statement.setDouble(i: 2, d:price);
        statement.setString(i: 3, string: productName);
        statement.setString(i: 4, string: category);
        statement.setString(i: 5, string: photo1);
        statement.setString(i: 6, string: photo2);
        statement.setString(i: 7, string: photo3);
        statement.setString(i: 8, string: photo4);
        statement.setString(i: 9, string: brand);
        statement.setString(i: 10, string: modelName);
        statement.setString(i: 11, string: productDescription);
        statement.setString(i: 12, string: formFactor);
        statement.setString(i: 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(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 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(i: 1, i1: quantity);
    statement.setDouble(i: 2, d:price);
    statement.setString(i: 3, string: productName);
    statement.setString(i: 4, string: category);
    statement.setString(i: 5, string: photo1);
    statement.setString(i: 6, string: photo2);
    statement.setString(i: 7, string: photo3);
    statement.setString(i: 8, string: photo4);
    statement.setString(i: 9, string: brand);
    statement.setString(i: 10, string: modelName);
    statement.setString(i: 11, string: productDescription);
    statement.setString(i: 12, string: screenSize);
    statement.setString(i: 13, string: refreshRate);
    statement.setString(i: 14, string: resolution);
    statement.setString(i: 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(i: 1, string: productId);
        statement.setInt(i: 2, i1: quantity);
        statement.setDouble(i: 3, d:price);
        statement.setString(i: 4, string: productName);
        statement.setString(i: 5, string: category);
        statement.setString(i: 6, string: photo1);
        statement.setString(i: 7, string: photo2);
        statement.setString(i: 8, string: photo3);
        statement.setString(i: 9, string: photo4);
        statement.setString(i: 10, string: brand);
        statement.setString(i: 11, string: modelName);
        statement.setString(i: 12, string: productDescription);
        statement.setString(i: 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(i: 1, string: productId);
            statement.setInt(i: 2, i1: quantity);
            statement.setDouble(i: 3, d:price);
            statement.setString(i: 4, string: productName);
            statement.setString(i: 5, string: category);
            statement.setString(i: 6, string: photo1);
            statement.setString(i: 7, string: photo2);
            statement.setString(i: 8, string: photo3);
            statement.setString(i: 9, string: photo4);
            statement.setString(i: 10, string: brand);
            statement.setString(i: 11, string: modelName);
            statement.setString(i: 12, string: productDescription);
            statement.setString(i: 13, string: screenSize);
            statement.setString(i: 14, string: refreshRate);
            statement.setString(i: 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(i: 1, i1: quantity);
        statement.setDouble(i: 2, d:price);
        statement.setString(i: 3, string: productName);
        statement.setString(i: 4, string: category);
        statement.setString(i: 5, string: photo1);
        statement.setString(i: 6, string: photo2);
        statement.setString(i: 7, string: photo3);
        statement.setString(i: 8, string: photo4);
        statement.setString(i: 9, string: brand);
        statement.setString(i: 10, string: modelName);
        statement.setString(i: 11, string: productDescription);
        statement.setString(i: 12, string: screenSize);
        statement.setString(i: 13, string: color);
        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, 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(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 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(i: 1, string: productId);
    statement.setInt(i: 2, i1: pieces);
    statement.setDouble(i: 3, d:price);
    statement.setString(i: 4, string: productName);
    statement.setString(i: 5, string: category);
    statement.setString(i: 6, string: photo1);
    statement.setString(i: 7, string: photo2);
    statement.setString(i: 8, string: photo3);
    statement.setString(i: 9, string: photo4);
    statement.setString(i: 10, string: brand);
    statement.setString(i: 11, string: modelName);
    statement.setString(i: 12, string: productDescription);
    statement.setString(i: 13, string: screenSize);
    statement.setString(i: 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(sr: request, sr1: response);
  }
```

Navbar

Navbar.java

```
<nav id="topNav" class="collapse">
  <div class="nav-content">
    <img class="nav-brand" src="Images/Navbar/E-MartLogo.svg" alt="E-Mart">
    <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">
        <img src="Images/Navbar/profile.svg" alt="User" class="nav-button-icon">
      </a>
      <% } else { // User is not logged in %>
      <a href="SignUp.jsp" class="nav-button sign">
        <span class="indicator">SignUp</span>
        <img src="Images/Navbar/profile.svg" alt="User" class="nav-button-icon">
      </a>
      <% } %>
      <a href="ShoppingCart.jsp" class="nav-button">
        <img src="Images/Navbar/cart.svg" alt="" class="nav-button-icon">
      </a>
    </div>
  </div>
  <div class="hamburger" onclick="expandNavBar();">
    <div class="line"></div>
    <div class="line"></div>
    <div class="line"></div>
  </div>
</nav>
```

Navbar.css

```
<nav id="topNav" class="collapse">
  <div class="nav-content">
    <img class="nav-brand" src="Images/Navbar/E-MartLogo.svg" alt="E-Mart">
    <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">
        <img src="Images/Navbar/profile.svg" alt="User" class="nav-button-icon">
      </a>
      <% } else { // User is not logged in %>
      <a href="SignUp.jsp" class="nav-button sign">
        <span class="indicator">SignUp</span>
        <img src="Images/Navbar/profile.svg" alt="User" class="nav-button-icon">
      </a>
      <% } %>
      <a href="ShoppingCart.jsp" class="nav-button">
        <img src="Images/Navbar/cart.svg" alt="" class="nav-button-icon">
      </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;
}
```

```css
.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;
}
```

```css
@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;
  }
}
```

```css
.collaspe{
  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

Footer.html

```
<footer>
  <div class="footer-content">
    <div class="footer-details">
      <img src="Images/Navbar/E-MartLogo.svg" alt="EMart" class="footer-logo">
      <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/?_rdc=1&_rdr">
        <img src="Images/Footer/facebook.svg" alt="facebook" class="social-icon-link">
        </a>
        <a href="https://twitter.com/?lang=en">
        <img src="Images/Footer/twitter.svg" alt="twitter" class="social-icon-link">
        </a>
        <a href="https://www.instagram.com/?hl=en">
        <img src="Images/Footer/insta.svg" alt="instagram" class="social-icon-link">
        </a>
        <a href="https://www.linkedin.com/">
        <img src="Images/Footer/linkedin.svg" alt="linkedin" class="social-icon-link">
        </a>
        <a href="https://www.youtube.com/">
        <img src="Images/Footer/youtube.svg" alt="youtube" class="social-icon-link">
        </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">
          <img src="Images/Footer/whatsapp.svg" alt="" class="footer-contact-icon">
          <span>+94 77 696 9696</span>
        </a>
        <a href="https://mail.google.com/" class="footer-link footer-contact-link">
          <img src="Images/Footer/mail.svg" alt="" class="footer-contact-icon">
          <span>mail.emart@gmail.com</span>
        </a>
      </div>
    </div>
    <hr class="seperator"/>
    <div class="footer-extra">
      <span class="footer-extra-text">© 2024 Electronics Mart, Inc </span>

      <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>
  </div>
</footer>
```

Footer.css

```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, 0.42) 17.86%, #000000 82.12%), linear-gradient(93.07deg, rgba(0, 0, 0, 0.24) 0%, rgba(0, 0, 0, 0.24) 100%);
    box-shadow: 0px 0px 20px rgba(0, 0, 0, 0.25);

}

.footer-content {
    display: flex;
    flex-direction: row;
    justify-content: stretch;
    align-items: stretch;
    gap: 4rem;
}

.seperator {
    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;
    }
}
```

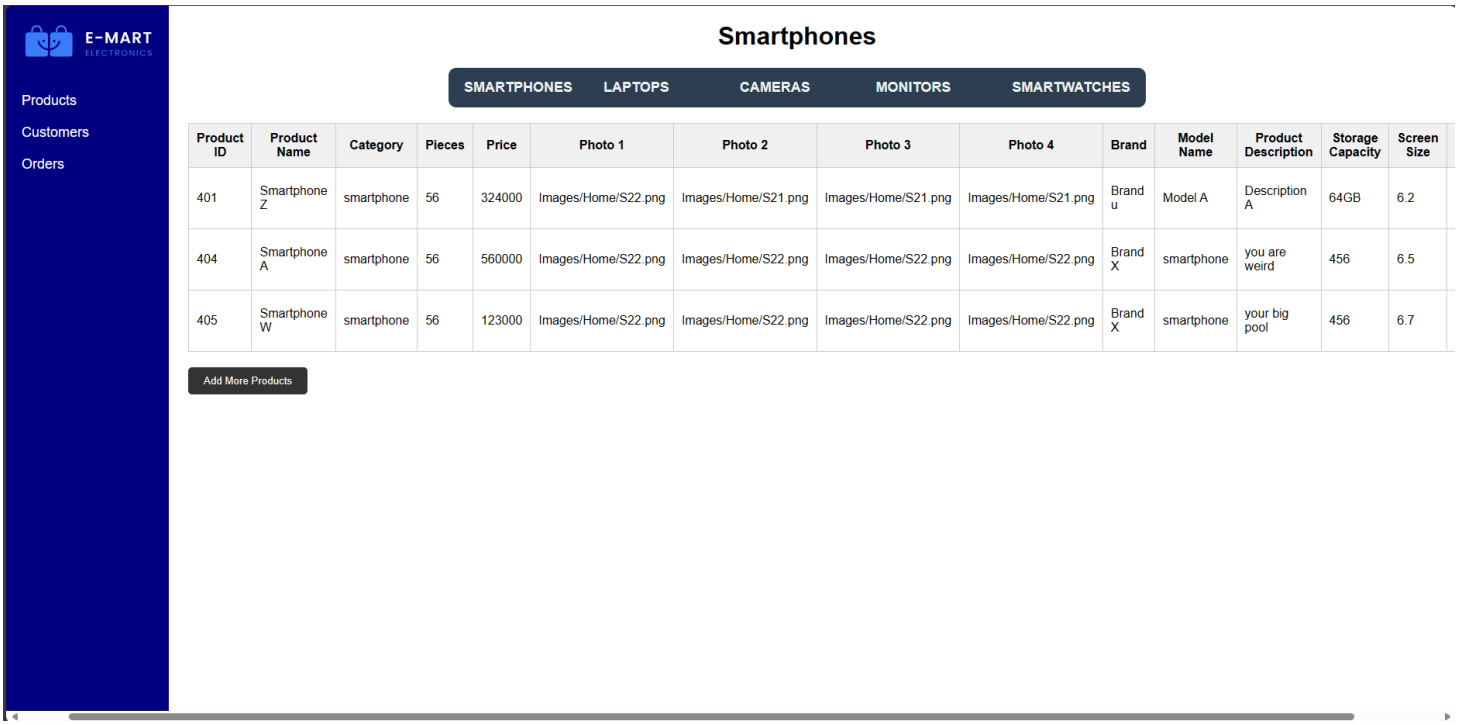
2. Admin Panel

- 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.

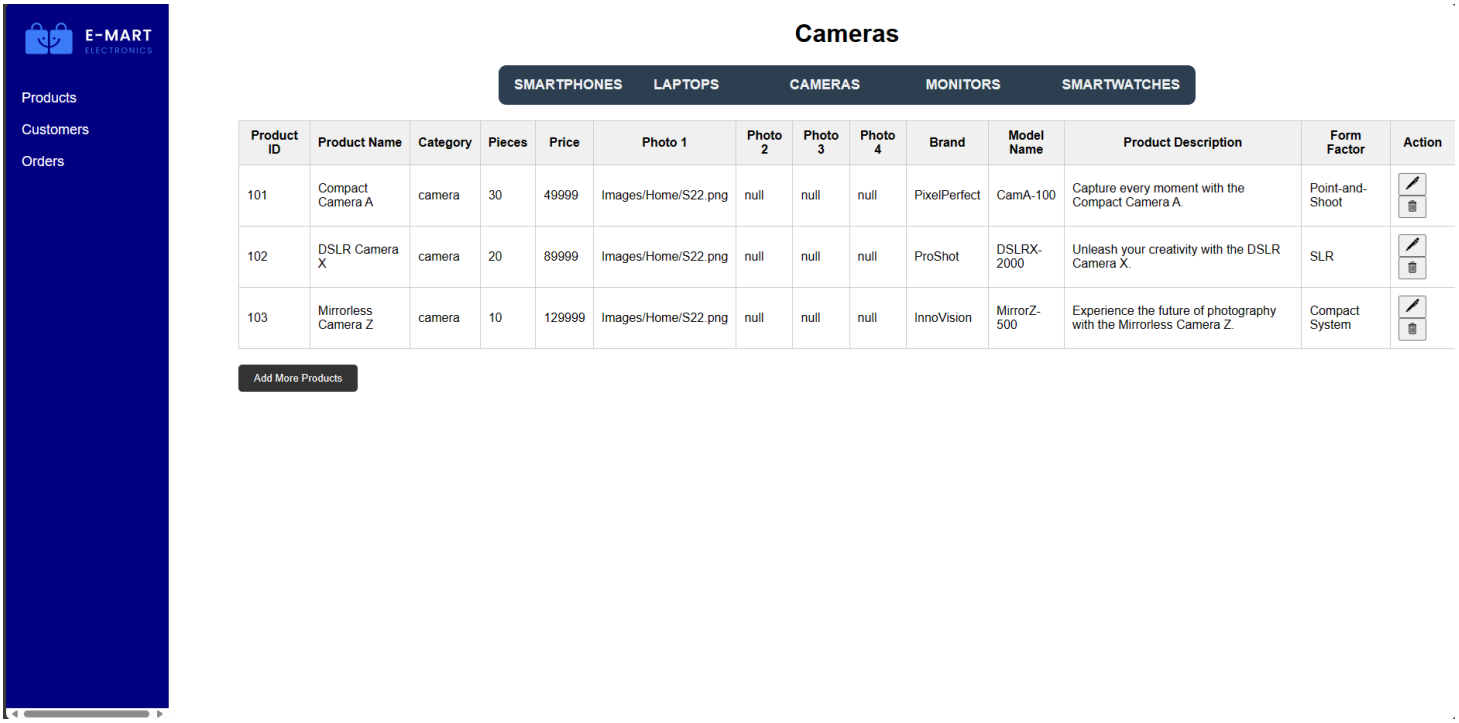
2.1 Products page (index page)

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

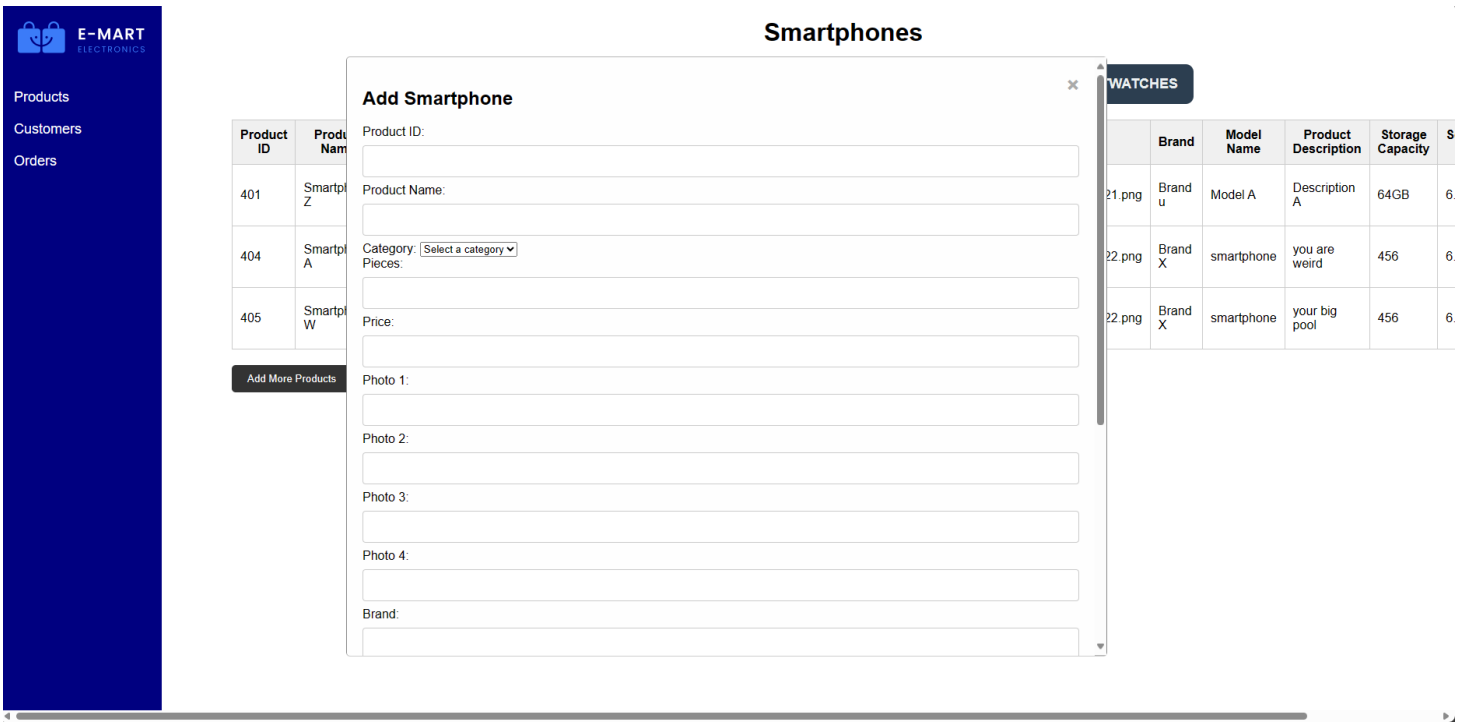
2.1.1 Web page



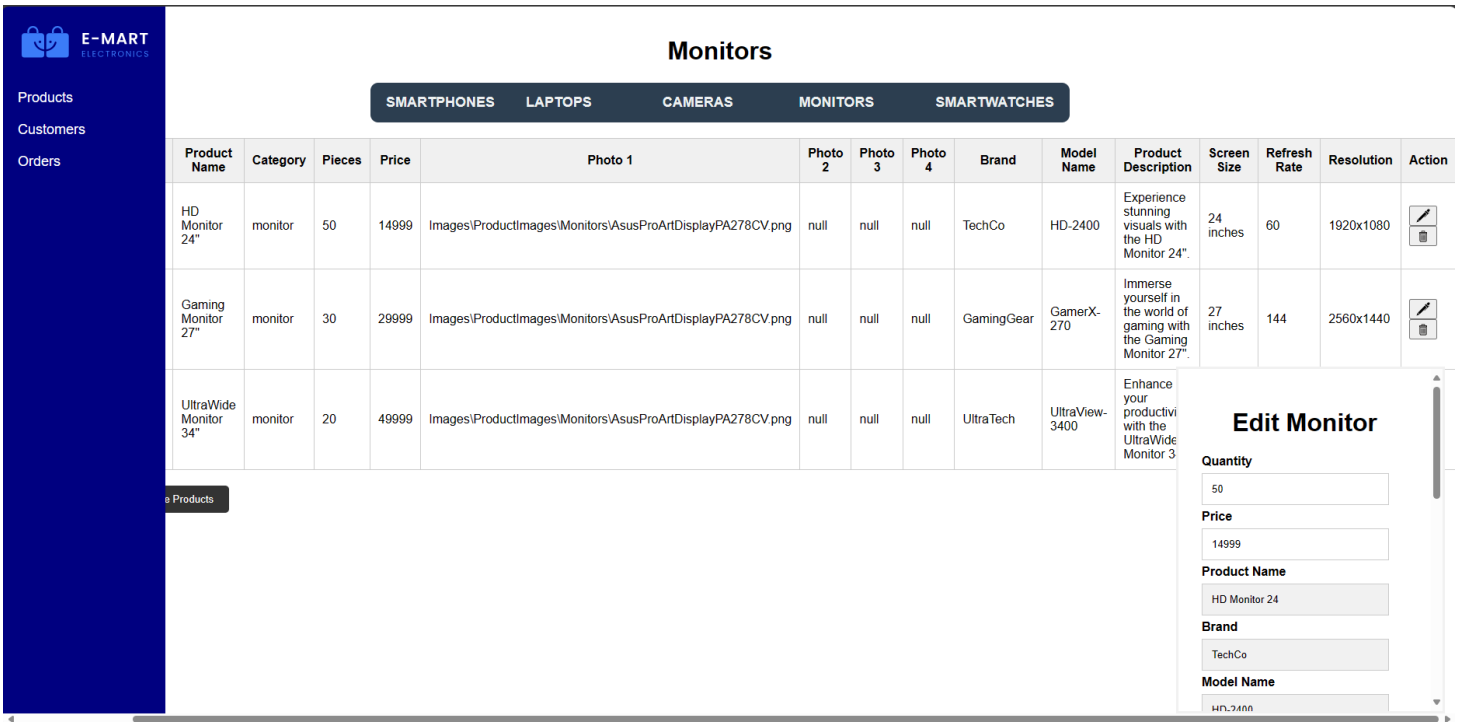
Product ID	Product Name	Category	Pieces	Price	Photo 1	Photo 2	Photo 3	Photo 4	Brand	Model Name	Product Description	Storage Capacity	Screen Size
401	Smartphone Z	smartphone	56	324000	Images/Home/S22.png	Images/Home/S21.png	Images/Home/S21.png	Images/Home/S21.png	Brand u	Model A	Description A	64GB	6.2
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



Add products form:



Edit product info - form :



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

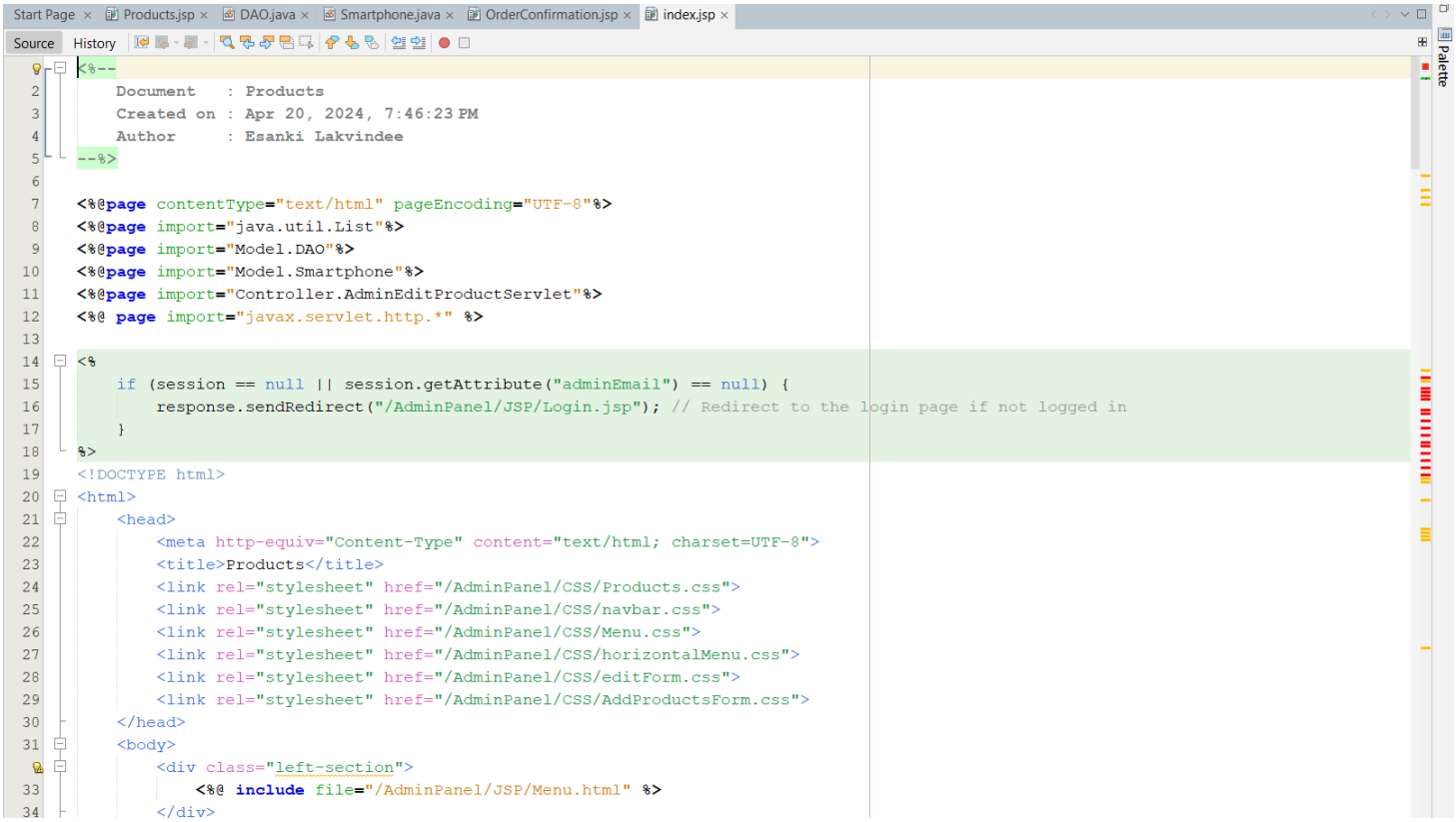
DAO.java: This file includes database-related methods like updateSmartphone(), deleteSmartphone(), and addSmartphone(). These methods perform database operations such as updating, deleting, and adding smartphone products.

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

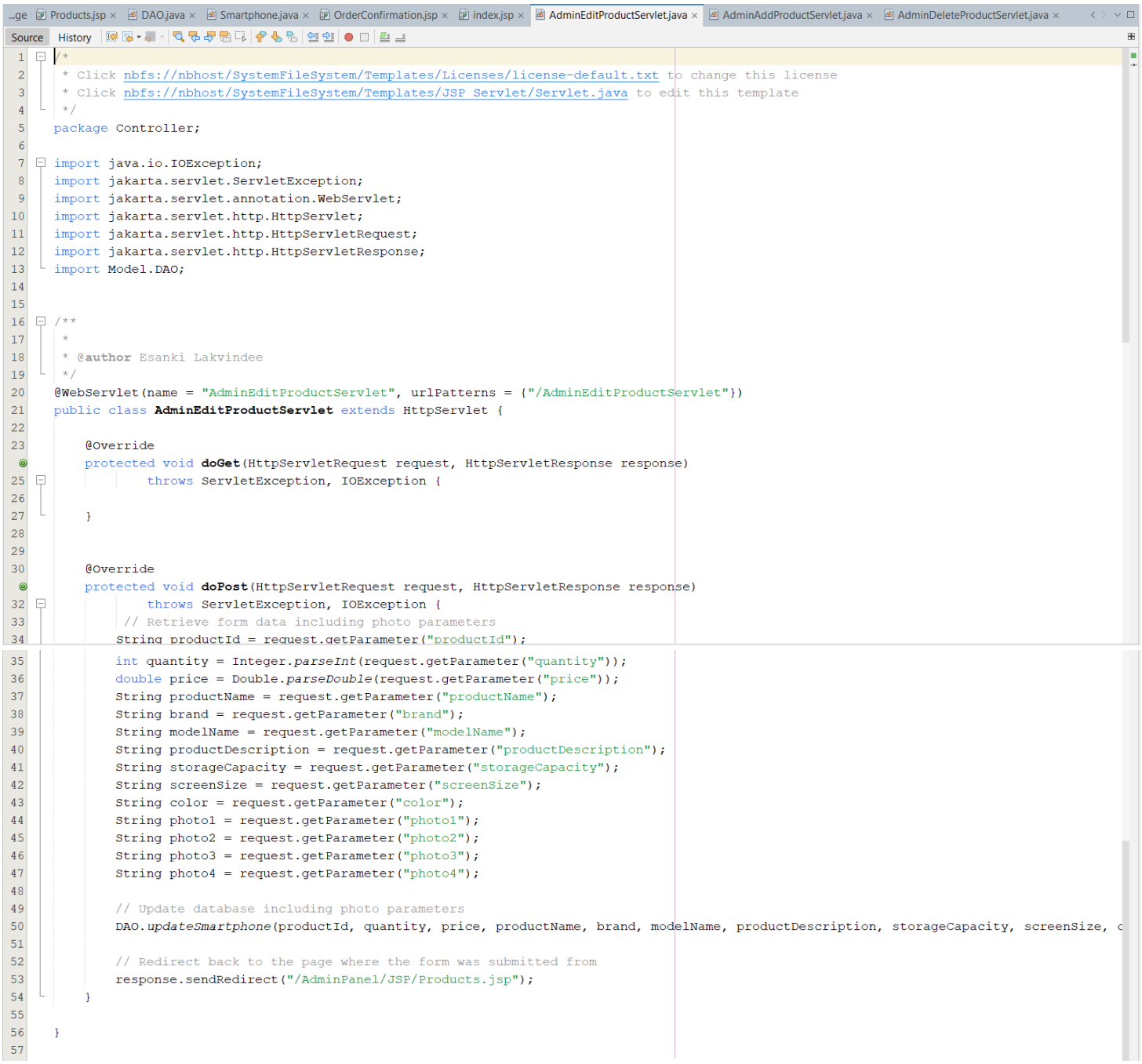


```
        <div class="right-section">

            <div class="container">
                <h1>Smartphones</h1>
                <ul class="horizontal-menu">
                    <li><a href="/AdminPanel/JSP/Products.jsp">Smartphones</a></li>
                    <li><a href="/AdminPanel/JSP/Laptop.jsp">Laptops</a></li>
                    <li><a href="/AdminPanel/JSP/Camera.jsp">Cameras</a></li>
                    <li><a href="/AdminPanel/JSP/Monitor.jsp">Monitors</a></li>
                    <li><a href="/AdminPanel/JSP/Smartwatch.jsp">Smartwatches</a></li>
                </ul>

                <table id="product-table">
                    <thead>
                        <tr>
                            <th>Product ID</th>
                            <th>Product Name</th>
                            <th>Category</th>
                            <th>Pieces</th>
                            <th>Price</th>
                            <th>Photo 1</th>
                            <th>Photo 2</th>
                            <th>Photo 3</th>
                            <th>Photo 4</th>
                            <th>Brand</th>
                            <th>Model Name</th>
                            <th>Product Description</th>
                            <th>Storage Capacity</th>
                            <th>Screen Size</th>
                            <th>Color</th>
                            <th>Action</th>
                        </tr>
                    </thead>
                    <tbody>
                        <%
                        List<Smartphone> products = DAO.getAllSmartphones();
                        for (Smartphone item : products) {
                        %>
                        <tr>
                            <td><%= item.getProductId() %></td>
                            <td><%= item.getProductName() %></td>
                            <td><%= item.getCategory() %></td>
                            <td><%= item.getQuantity() %></td>
                            <td><%= item.getPrice() %></td>
                            <td><%= item.getPhoto1() %></td>
                            <td><%= item.getPhoto2() %></td>
                            <td><%= item.getPhoto3() %></td>
                            <td><%= item.getPhoto4() %></td>
                            <td><%= item.getBrand() %></td>
                            <td><%= item.getModelName() %></td>
                            <td><%= item.getProductDescription() %></td>
                            <td><%= item.getStorageCapacity() %></td>
                            <td><%= item.getScreenSize() %></td>
                            <td><%= item.getColor() %></td>
                            <td>
                                <button class="button1" onclick="openForm('<%= item.getProductId() %>')">
                                    <img src="/AdminPanel/Images/pen.svg" alt="Edit" style="width: 20px; height: 20px;">
                                </button>
                                <div class="form-popup" id="myForm_<%= item.getProductId() %>">
                                    <form action="/AdminEditProductServlet" class="form-container" method="post">
                                        <h1>Edit Smartphone</h1>
                                        <label for="pieces_<%= item.getProductId() %>"><b>Quantity</b></label>
                                        <input type="number" placeholder="Enter quantity" name="quantity" value="<%= item.getQuantity()
                                        <label for="price_<%= item.getProductId() %>"><b>Price</b></label>
                                        <input type="number" step="0.01" placeholder="Enter Price" name="price" value="<%= item.getPric
                                        <input type="hidden" name="productId" value="<%= item.getProductId() %>">
```

```
                                <input type="hidden" name="category" value="<%= item.getCategory() %>">
                                <label for="productName_<%= item.getProductId() %>"><b>Product Name</b></label>
                                <input type="text" placeholder="Enter product name" name="productName" value="<%= item.getProdu
                                <label for="brand_<%= item.getProductId() %>"><b>Brand</b></label>
                                <input type="text" placeholder="Enter brand" name="brand" value="<%= item.getBrand() %>" requir
                                <label for="modelName_<%= item.getProductId() %>"><b>Model Name</b></label>
                                <input type="text" placeholder="Enter model name" name="modelName" value="<%= item.getModelName
                                <label for="productDescription_<%= item.getProductId() %>"><b>Product Description</b></label>
                                <textarea placeholder="Enter product description" name="productDescription" required><%= item.g
                                <label for="storageCapacity_<%= item.getProductId() %>"><b>Storage Capacity</b></label>
                                <input type="text" placeholder="Enter storage capacity" name="storageCapacity" value="<%= item.
                                <label for="screenSize_<%= item.getProductId() %>"><b>Screen Size</b></label>
                                <input type="text" placeholder="Enter screen size" name="screenSize" value="<%= item.getScreenS
                                <label for="color_<%= item.getProductId() %>"><b>Color</b></label>
                                <input type="text" placeholder="Enter color" name="color" value="<%= item.getColor() %>" requir
                                <label for="photo1_<%= item.getProductId() %>"><b>Photo 1</b></label>
                                <input type="text" placeholder="Enter URL for photo 1" name="photo1" value="<%= item.getPhoto1(
                                <label for="photo2_<%= item.getProductId() %>"><b>Photo 2</b></label>
                                <input type="text" placeholder="Enter URL for photo 2" name="photo2" value="<%= item.getPhoto2(
                                <label for="photo3_<%= item.getProductId() %>"><b>Photo 3</b></label>
                                <input type="text" placeholder="Enter URL for photo 3" name="photo3" value="<%= item.getPhoto3(
                                <label for="photo4_<%= item.getProductId() %>"><b>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() %>')">Clos
                            </form>
                        </div>


                        <button class="button2" onclick="deleteProduct('<%= item.getProductId() %>')">
                            <img src="/AdminPanel/Images/delete.svg" alt="Delete" style="width: 20px; height: 20px;">
                        </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>
```

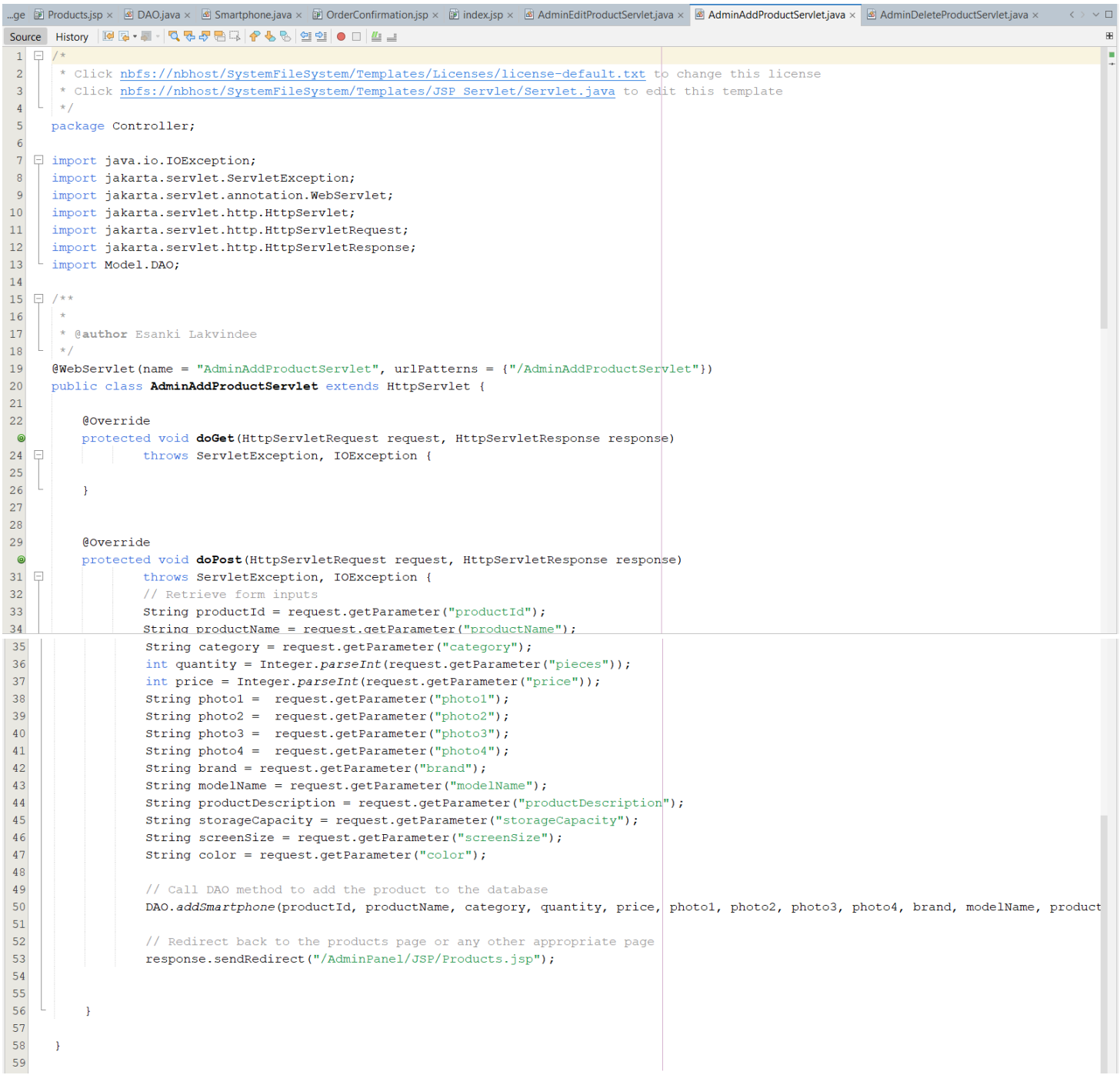
```
                        <label for="productDescription">Product Description:</label><br>
                        <textarea id="productDescription" name="productDescription" required></textarea><br>
                        <label for="storageCapacity">Storage Capacity:</label>
                        <input type="text" id="storageCapacity" name="storageCapacity" required><br>
                        <label for="screenSize">Screen Size:</label>
                        <input type="text" id="screenSize" name="screenSize" required><br>
                        <label for="color">Color:</label>
                        <input type="text" id="color" name="color" required><br>
                        <button type="submit" onclick="saveProduct()">Save</button>
                        <span class="close" onclick="closetheForm()">&times;</span>
                    </form>
                </div>
            </div>


            <!--<div class="pagination">
                <span>Pages:</span>
                <a href="#">1</a>
                <a href="#">2</a>
                <a href="#">3</a>
            </div>-->
        </div>
    </div>

    <script>
        function openForm(productId) {
            var formId = "myForm_" + productId;
            document.getElementById(formId).style.display = "block";
        }

        function closeForm(productId) {
            var formId = "myForm_" + productId;
            document.getElementById(formId).style.display = "none";
        }

        function opentheForm() {
            document.getElementById("addProductModal").style.display = "block";
        }

        function closetheForm() {
            document.getElementById("addProductModal").style.display = "none";
        }

    </script>

    <script src="/AdminPanel/JS/delete.js"></script>

</body>
</html>
```

AdminEditProductServlet.java



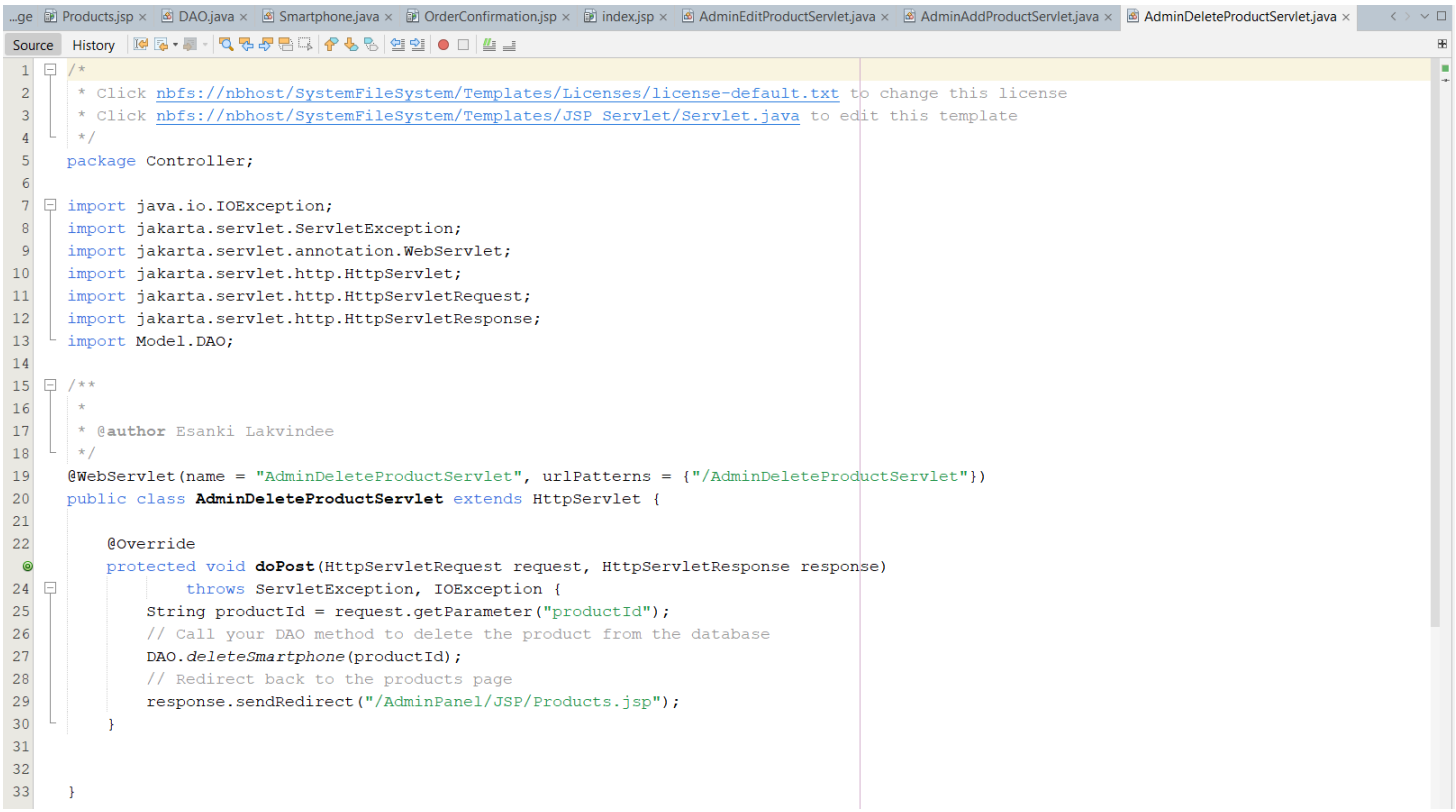
```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template
 */
package Controller;

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 = "AdminEditProductServlet", urlPatterns = {"/AdminEditProductServlet"})
public class AdminEditProductServlet 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 data including photo parameters
        String productId = request.getParameter("productId");
        int quantity = Integer.parseInt(request.getParameter("quantity"));
        double price = Double.parseDouble(request.getParameter("price"));
        String productName = request.getParameter("productName");
        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");
        String photo1 = request.getParameter("photo1");
        String photo2 = request.getParameter("photo2");
        String photo3 = request.getParameter("photo3");
        String photo4 = request.getParameter("photo4");

        // Update database including photo parameters
        DAO.updateSmartphone(productId, quantity, price, productName, brand, modelName, productDescription, storageCapacity, screenSize, c

        // Redirect back to the page where the form was submitted from
        response.sendRedirect("/AdminPanel/JSP/Products.jsp");
    }

}
```

AdminAddProductServlet.java



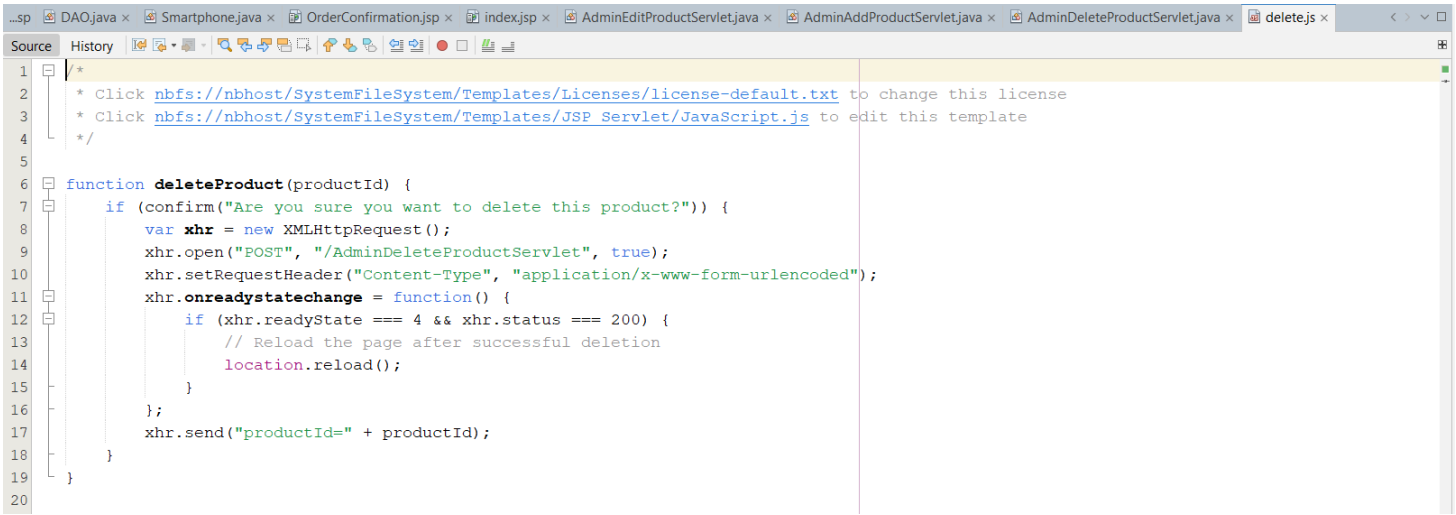
```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template
 */
package Controller;

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, product

            // Redirect back to the products page or any other appropriate page
            response.sendRedirect("/AdminPanel/JSP/Products.jsp");


    }

}
```

AdminDeleteProductServlet.java



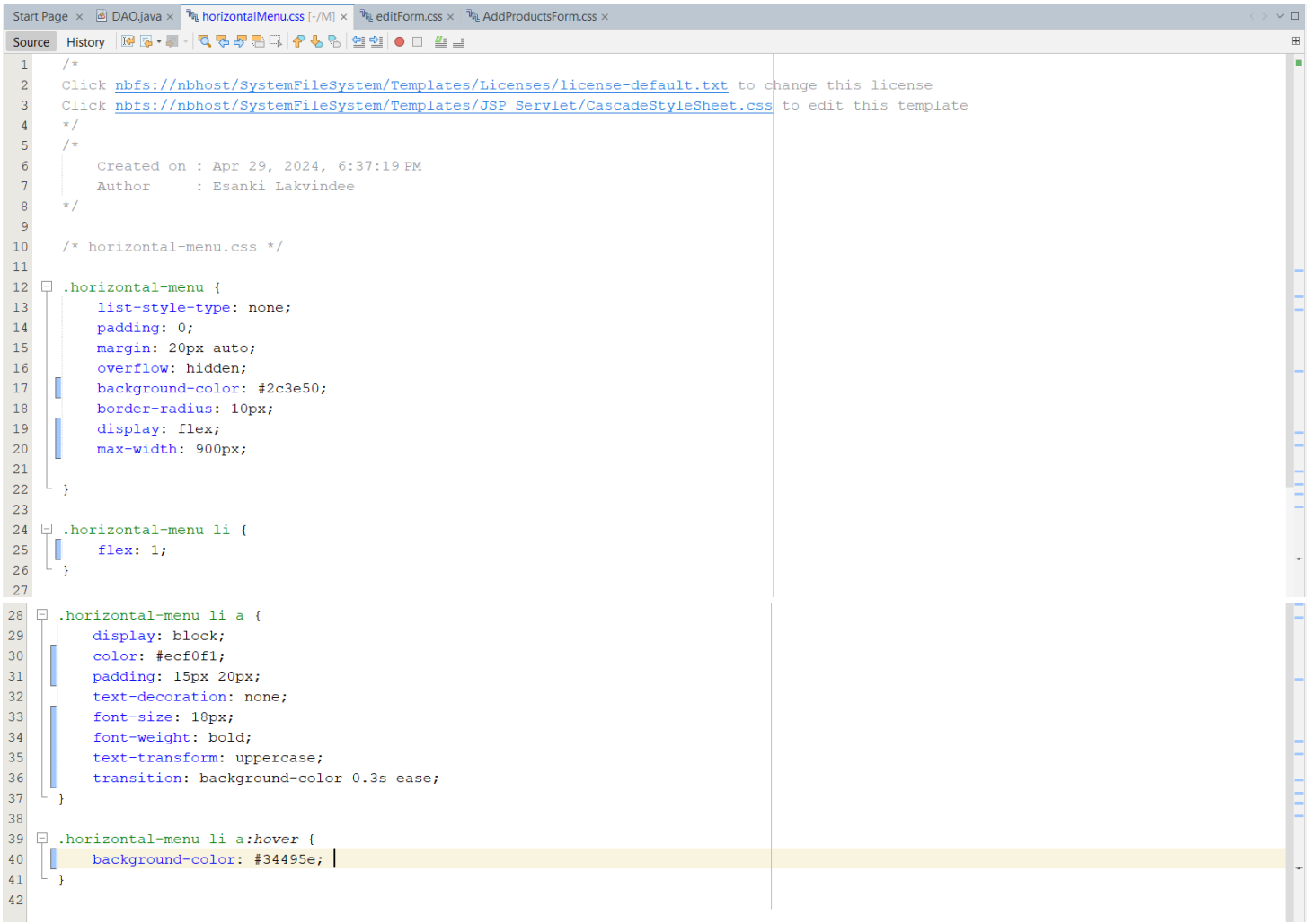
```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/Servlet.java to edit this template
 */
package Controller;

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 = "AdminDeleteProductServlet", urlPatterns = {"/AdminDeleteProductServlet"})
public class AdminDeleteProductServlet extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        String productId = request.getParameter("productId");
        // Call your DAO method to delete the product from the database
        DAO.deleteSmartphone(productId);
        // Redirect back to the products page
        response.sendRedirect("/AdminPanel/JSP/Products.jsp");
    }


}
```

delete.js



```javascript
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/JSP_Servlet/JavaScript.js to edit this template
 */

function deleteProduct(productId) {
    if (confirm("Are you sure you want to delete this product?")) {
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "/AdminDeleteProductServlet", true);
        xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
        xhr.onreadystatechange = function() {
            if (xhr.readyState === 4 && xhr.status === 200) {
                // Reload the page after successful deletion
                location.reload();
            }
        };
        xhr.send("productId=" + productId);
    }
}
```

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

Start Page × DAO.java × horizontalMenu.css [-/M] × editForm.css × AddProductsForm.css ×

Source History

```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 30, 2024, 8:56:00 PM
    Author     : Esanki Lakvindee
*/

/* Popup container */
.form-popup {
  display: none;
  position: fixed;
  bottom: 0;
  right: 15px;
  border: 3px solid #f1f1f1;
  z-index: 9;
  background-color: white;
  max-width: 300px;
  max-height: 400px; /* Set a maximum height for scrolling */
  overflow-y: auto; /* Enable vertical scrolling */
  padding: 20px;
}

/* Full-width input fields */
.form-popup input[type=text],
.form-popup input[type=number],
.form-popup textarea {
  width: calc(100% - 24px); /* Adjust for padding */
  padding: 12px;
  margin: 6px 0;
  display: inline-block;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

/* Set a style for the submit/save button */
.form-popup .btn {
  background-color: #4CAF50;
  color: white;
  padding: 14px 20px;
  border: none;
  cursor: pointer;
  width: 100%;
  margin-bottom: 10px;
}

/* Add a red background color to the cancel button */
.form-popup .btn.cancel {
  background-color: #f44336;
}

/* Add some hover effects to buttons */
.form-popup .btn:hover, .open-button:hover {
  opacity: 0.8;
}
```

```css
/* Position and style the close button (top right corner) */
.form-popup .btn.close {
  position: absolute;
  top: 0;
  right: 0;
  padding: 12px 16px;
  margin: -20px -20px 0 0;
}

/* Add animation (fade in the popup) */
@keyframes fadeIn {
  from {opacity: 0;}
  to {opacity: 1;}
}

.form-popup {
  animation: fadeIn 0.3s ease-in-out;
}

```

AddProductsForm.css

Start Page × DAO.java × horizontalMenu.css [-/M] × editForm.css × AddProductsForm.css ×

Source History

```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 : May 1, 2024, 9:35:41 AM
    Author     : Esanki Lakvindee
*/

/* Popup form content */

.modal-content {
    background-color: #fefefe;
    position: fixed;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    padding: 20px;
    border: 1px solid #888;
    width: 50%; /* You can adjust the width as needed */
    border-radius: 5px;
    max-height: 80%; /* Set maximum height */
    overflow-y: auto; /* Enable vertical scrolling */
    z-index: 9999; /* Ensure modal is on top of other content */
}

```

```css
/* Center input fields */
.modal-content input[type="text"],
.modal-content input[type="number"] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    display: inline-block;
    border: 1px solid #ccc;
    border-radius: 4px;
    box-sizing: border-box;
}

/* Center buttons */
.modal-content button[type="submit"] {
    width: 100%;
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

.modal-content button[type="submit"]:hover {
    background-color: #45a049;
}


.close {
    color: #aaa;
    float: right;
    font-size: 28px;
    font-weight: bold;
}

.close:hover,
.close:focus {
    color: black;
    text-decoration: none;
    cursor: pointer;
}

```

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=?, storageC
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 
732             // Create PreparedStatement
733             statement = connection.prepareStatement(query);
734 
735             // Set parameters
736             statement.setString(1, productId);
737             statement.setString(2, productName);
738             statement.setString(3, category);
739             statement.setInt(4, quantity);
740             statement.setInt(5, price);
741             statement.setString(6, photo1);
742             statement.setString(7, photo2);
743             statement.setString(8, photo3);
744             statement.setString(9, photo4);
745             statement.setString(10, brand);
746             statement.setString(11, modelName);
747             statement.setString(12, productDescription);
748             statement.setString(13, storageCapacity);
749             statement.setString(14, screenSize);
750             statement.setString(15, color);
751 
```

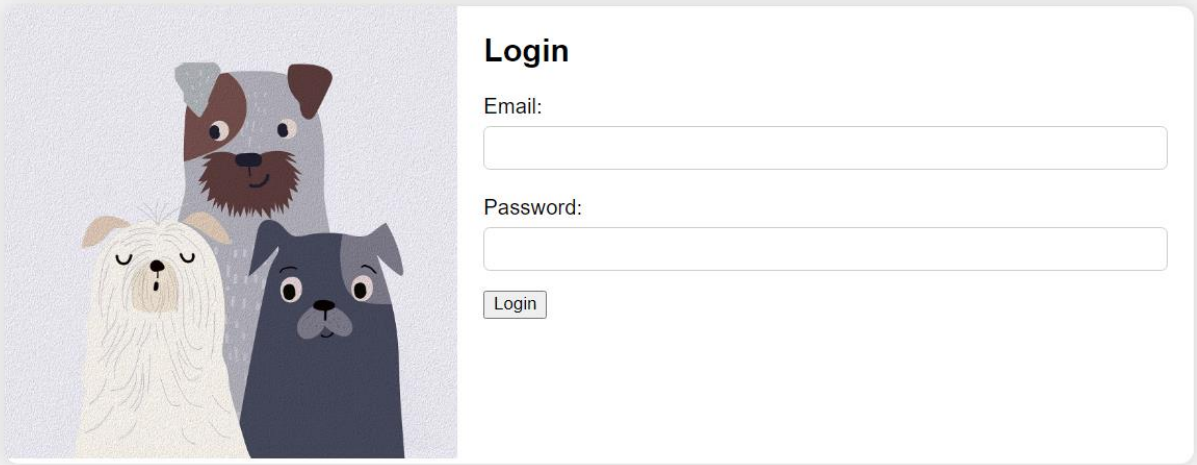
```
752             // Execute the query
753             statement.executeUpdate();
754         } catch (SQLException e) {
            e.printStackTrace(); // Handle or log the exception appropriately
756         } finally {
757             // Close the connection and statement
758             try {
759                 if (statement != null) {
760                     statement.close();
761                 }
762                 if (connection != null) {
763                     connection.close();
764                 }
765             } catch (SQLException e) {
                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

```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">
          <img src="/AdminPanel/Images/cute.jpg" alt="Login Image">
        </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>
    </div>
  </body>
</html>
```

AdminLoginServlet.java

```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(string: "email");
        String Password = request.getParameter(string: "password");

        // Check if the password is correct
        if ("emart24051".equals(anObject: Password)) {
            // Create a session
            HttpSession session = request.getSession();
            // Store user information in session
            session.setAttribute(string: "email", o:Email);
            // Redirect to profile page
            response.sendRedirect(request.getContextPath() + "/AdminPanel/JSP/Profile.jsp");
        } else {
            // Password is incorrect, display an error message
            request.setAttribute(string: "errorMessage", o:"Incorrect password");
            request.getRequestDispatcher(string: "/AdminPanel/JSP/Login.jsp").forward(sr: request, sr1: response);
        }

        AdminLogin login = new AdminLogin();
        login.insertloginData(Email, Password);

    }
```

AdminLogin.java

```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(string: query);
                System.out.println(x:"Record inserted");
            } catch (SQLException ex) {
                System.out.println(x:ex.getMessage());
            }
        } else {
            System.out.println(x:"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(string: query);
            System.out.println(x:"Record inserted");
        } catch (SQLException ex) {
            System.out.println(x: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);
        }
    }
```

```
    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(x: "Record inserted");
        } catch (SQLException ex) {
            System.out.println(x: 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(i: 1, string: email);
        rs = checkStatement.executeQuery();
        emailExists = rs.next(); // Check if there's a result
    } catch (SQLException ex) {
        System.out.println(x: ex.getMessage());
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException ex) {
            System.out.println(x: 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( i: 1, string: newusername);
            updateStatement.setString( i: 2, string: newcontactNumber);
            updateStatement.setString( i: 3, string: email);
            updateStatement.executeUpdate();
            System.out.println( x: "Details updated successfully.");
        } catch (SQLException ex) {
            System.out.println( x: ex.getMessage());
        }
    } else {
        System.out.println( x: "Email address not found. Update failed.");
        // You can send an error message to the user here
    }
}
```

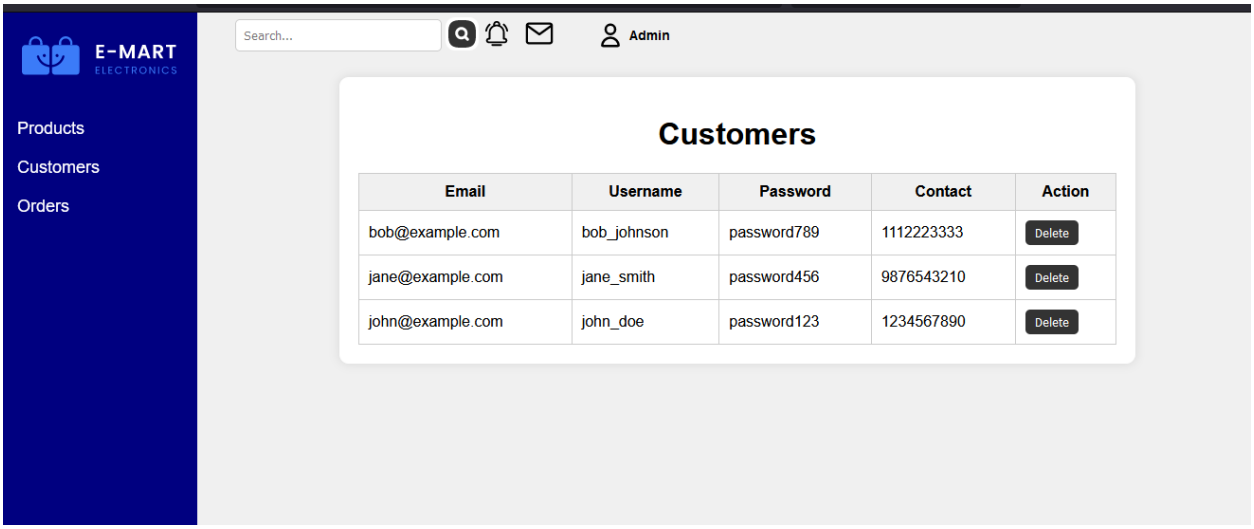
```
public SignUpUser getUserByEmail(String email) {
    connectToDB();
    String query = "SELECT * FROM users WHERE email = '" + email + "'";
    ResultSet rs = null;
    SignUpUser user = null;

    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( x: ex.getMessage());
    } finally {
        try {
            if (rs != null) {
                rs.close();
            }
        } catch (SQLException ex) {
            System.out.println( x: 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( x: "Password updated successfully.");
        } catch (SQLException ex) {
            System.out.println( x: ex.getMessage());
        }
    }

}
```

2.3 Customers page

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

2.3.1 Web page



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>
                    </form>
                   </td>
              </tr>
              <%
                  }
                  rs.close();
                  stmt.close();
                  conn.close();
              } catch (SQLException e) {
                  e.printStackTrace();
              }
              %>
              </tbody>
          </table>
    </div>
    </div>

    <!-- Popup Form -->
    <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>
```

```
                        <button type="button" id="cancelBtn">Cancel</button>
                    </form>
                </div>
            </div>

            <script src="/AdminPanel/JS/Customers.js"></script>

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

Customers.js

```
*/

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

```css
body {
    margin: 0;
    padding: 0;
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
}

.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: 800px;
    margin: 20px auto;
    padding: 20px;
    background-color: #fff;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
    text-align: center;
}
```

```css
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

```
 */
package Controller;
import Model.CustomerDAO;

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;

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


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

    /**
     * Handles the HTTP <code>POST</code> method.
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        // Get the customer email parameter from the request
        String customeremail = request.getParameter("email");

        // Call your DAO method to delete the customer from the database
        CustomerDAO.deleteCustomer(customeremail);

        // Redirect back to the customers page
        response.sendRedirect("/AdminPanel/JSP/Customers.jsp");
    }
    // processRequest(request, response);


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

CustomerDAO.java

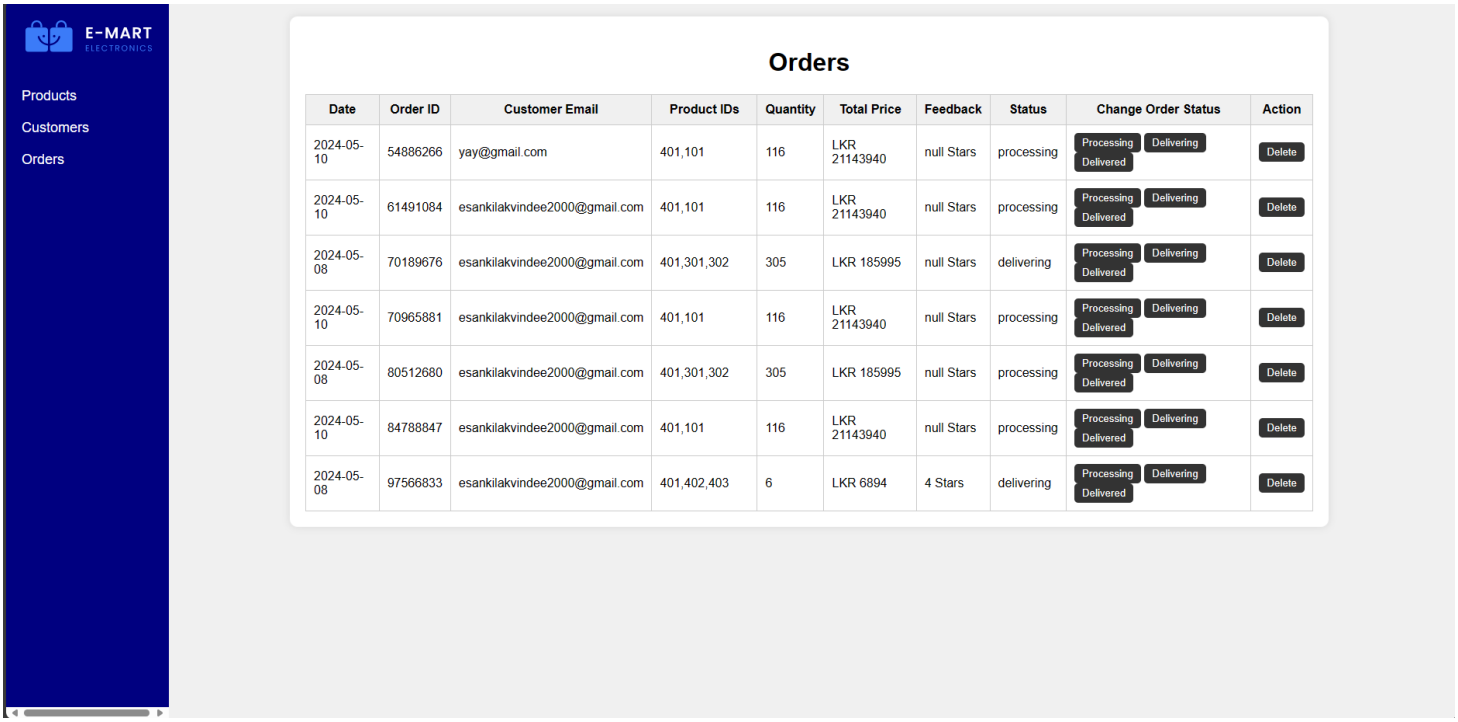
```
 */
package Model;

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

public class CustomerDAO {
    private static final String URL = "jdbc:mysql://localhost:3306/emart";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "";

    // Method to delete a customer record from the database
    public static void deleteCustomer(String customeremail) {
        Connection conn = null;
        PreparedStatement stmt = null;

        try {
            // Get a database connection
            conn = DriverManager.getConnection(URL, USERNAME, PASSWORD);

            // SQL statement to delete customer
            String sql = "DELETE FROM users WHERE email = ?";

            // Create prepared statement
            stmt = conn.prepareStatement(sql);

            // Set customer email parameter
            stmt.setString(1, customeremail);

```

```
            stmt = conn.prepareStatement(sql);

            // Set customer email parameter
            stmt.setString(1, customeremail);

            // Execute the delete operation
            stmt.executeUpdate();
        } catch (SQLException e) {
            // Handle any SQL errors
            e.printStackTrace();
        } finally {
            // Close resources
            try {
                if (stmt != null) {
                    stmt.close();
                }
                if (conn != null) {
                    conn.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

2.4 Admin Orders page

Done by : T.P.R. Fernando
Student ID: 26888

2.4.1 Web page



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 Delivering Delivered	Delete
2024-05-10	61491084	esankilakvindee2000@gmail.com	401,101	116	LKR 21143940	null Stars	processing	Processing Delivering Delivered	Delete
2024-05-08	70189676	esankilakvindee2000@gmail.com	401,301,302	305	LKR 185995	null Stars	delivering	Processing Delivering Delivered	Delete
2024-05-10	70965881	esankilakvindee2000@gmail.com	401,101	116	LKR 21143940	null Stars	processing	Processing Delivering Delivered	Delete
2024-05-08	80512680	esankilakvindee2000@gmail.com	401,301,302	305	LKR 185995	null Stars	processing	Processing Delivering Delivered	Delete
2024-05-10	84788847	esankilakvindee2000@gmail.com	401,101	116	LKR 21143940	null Stars	processing	Processing Delivering Delivered	Delete
2024-05-08	97566833	esankilakvindee2000@gmail.com	401,402,403	6	LKR 6894	4 Stars	delivering	Processing Delivering 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>LKR <%= 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>
</body>
</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;
}
```

```css
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

```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	Front end and back end of : • Home page • Order Confirmation page	Front end of : • Index page (products pages) • Customers page • Orders page • Log-in page • Nav bar Back end of : • Products pages (index) which include five different pages in total for each category. · Smartphones · Laptops · Camera · Monitors · Smartwatches
26888	T.P.R. Fernando	Front end and back end of • Shopping cart page Others : • Navbar • Footer • Database	• Back end of Orders page
29165	S.N.M. Gedara	Front end and back end of : • Sign In page · Forgot password · Verify OTP · Reset password • Sign up page • About Us page • Developers page • Track Order page Back end of • Profile page	• Back end of Log In page

28039	G.O. Wickramaratne	Front end and back end of : • Check out page • Contact page	• Back end of Customers page
30143	W.M.C.S. Wijesinghe	Front end and back end of • Product category pages • Product information pages	_____
26545	W.A.C. Fernando	• Front end of profile page	_____