Department of Mathematics and Computer Science

Faculty of Data Analysis

UNIME

Web Programming Report

Sport Nutrition

E-Commerce

Student:Zhumabekov Murad (537909)

Student: Zhengis Sanat (537902)

Professor: Armando Ruggeri

Academic Year 2023/2024

Table of Contents

1. Project Overview

1.1. Objective

The objective of this project is to develop an online store specializing in sports nutrition. The platform will enable users to browse, select, and purchase various sports nutrition products. The backend development will utilize PHP and MySQL to handle database interactions, user authentication, and order processing. The frontend will be designed using HTML, CSS, and JavaScript to create an intuitive and interactive user experience.

1.2. Scope

The online store will cater to customers interested in purchasing sports nutrition products. It will feature a user-friendly interface where visitors can browse products, view detailed descriptions, add items to their cart, and complete the checkout process. Additionally, the platform will include administrative functionality for managing products, orders, and user accounts. The project scope encompasses both frontend and backend development, with a focus on creating a seamless shopping experience for users.

2. Technology Stack

The development of the online store will leverage the following technologies:

- Frontend: HTML, CSS, JavaScript
- Backend: PHP
- Database: MySQL

The frontend technologies will be utilized to design the layout, style, and interactivity of the online store interface. PHP will power the backend logic, handling user authentication, product management, and order processing. MySQL will serve as the database management system for storing product information, user data, and order details.

3. Approach

In developing the online store, we aim to prioritize simplicity, usability, and performance. The frontend design will focus on creating a visually appealing and intuitive user interface, optimizing for ease of navigation and seamless shopping experience. We will implement responsive design techniques to ensure compatibility across various devices and screen sizes.

On the backend, we will adhere to best practices in PHP development to ensure robustness, security, and scalability. We will implement secure authentication mechanisms to protect user accounts and sensitive data. Additionally, we will optimize database queries and performance to ensure efficient data retrieval and processing.

Throughout the development process, we will adopt an iterative approach, continuously refining and improving the online store based on user feedback and testing results. Regular testing and debugging will be conducted to identify and address any issues or bugs promptly.

By employing a comprehensive technology stack and following sound development principles, we aim to deliver a high-quality online store that meets the needs and expectations of our target audience.

3. Database Structure

The database consists of five tables: cart, orders, products, users and wishlist.

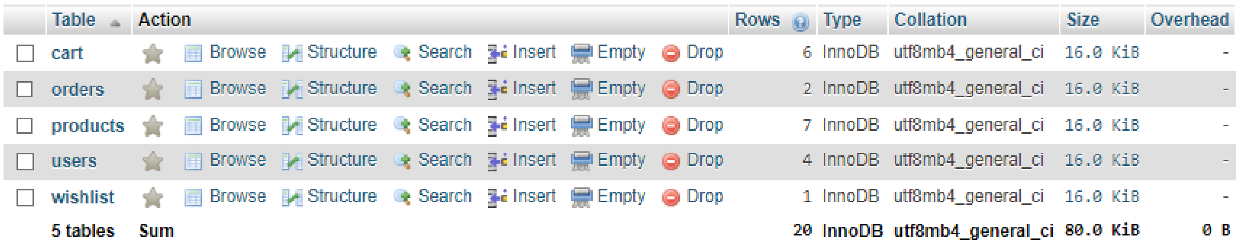


Table	Action	Rows	Type	Collation	Size	Overhead
cart	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_general_ci	16.0 KiB	-
orders	Browse Structure Search Insert Empty Drop	2	InnoDB	utf8mb4_general_ci	16.0 KiB	-
products	Browse Structure Search Insert Empty Drop	7	InnoDB	utf8mb4_general_ci	16.0 KiB	-
users	Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_general_ci	16.0 KiB	-
wishlist	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_general_ci	16.0 KiB	-
5 tables	Sum	20	InnoDB	utf8mb4_general_ci	80.0 KiB	0 B

3.1. Cart Table

The cart table serves to track the items added by users for potential purchase. It includes attributes such as:

- id: A unique identification number for each cart item.

- user_id: The identification number of the user associated with the cart.

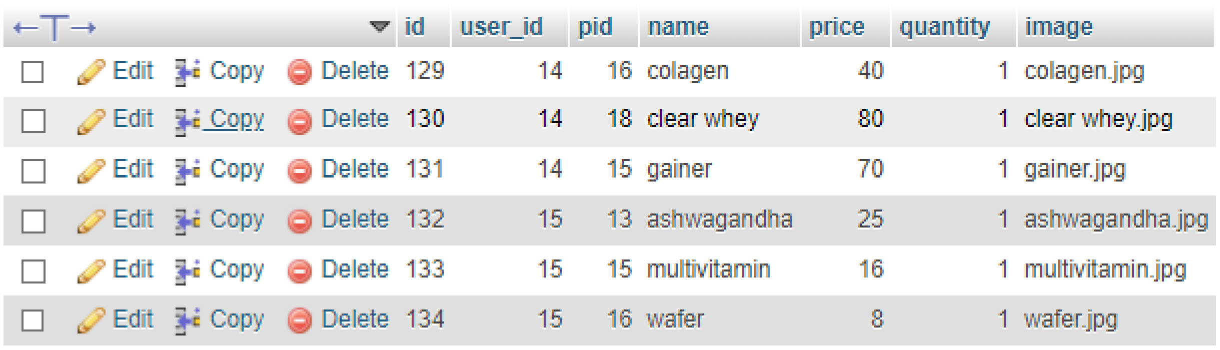
- pid: Represents the product ID associated with the cart entry. It links the cart entry to a specific product in the products table.

- name: Stores the name of the product in the cart. It provides a human-readable representation of the product associated with the cart entry.

- price: Represents the price of the product in the cart. It stores the cost associated with a particular product in the user's cart.

- quantity: The quantity of the product added to the cart.

- image: Stores the filename or path of the image associated with the product in the cart. This attribute facilitates displaying the product image when viewing the cart.



←T→	id	user_id	pid	name	price	quantity	image
Edit Copy Delete	129	14	16	colagen	40	1	colagen.jpg
Edit Copy Delete	130	14	18	clear whey	80	1	clear whey.jpg
Edit Copy Delete	131	14	15	gainer	70	1	gainer.jpg
Edit Copy Delete	132	15	13	ashwagandha	25	1	ashwagandha.jpg
Edit Copy Delete	133	15	15	multivitamin	16	1	multivitamin.jpg
Edit Copy Delete	134	15	16	wafer	8	1	wafer.jpg

3.2. Orders Table

The orders table stores information about completed orders placed by users. It includes attributes such as:

- id: A unique identification number for each order.
- user_id: The identification number of the user who placed the order.
- order_date: The date and time when the order was placed.
- total_price: The total amount of the order.
- payment_status: The status of the order (e.g., pending, processing, shipped).

3.3. Products Table

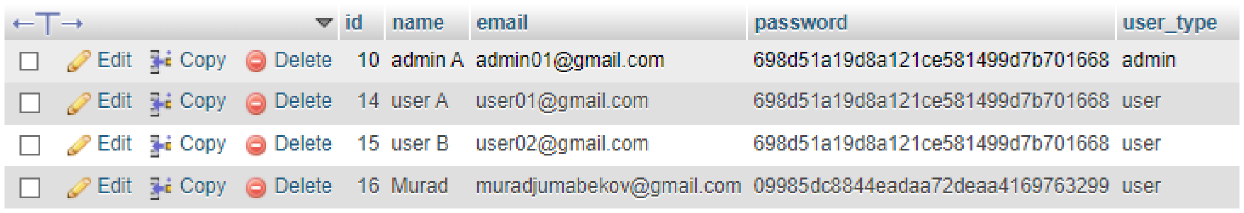
The products table contains details about the available sports nutrition products. It includes attributes such as:

- id: A unique identification number for each product.
- name: The name of the product.
- details: Additional information about the product.
- price: The price of the product.
- image: The image of the product.

3.4. Users Table

The users table stores information about registered users of the online store. It includes attributes such as:

- id: A unique identification number for each user.
- name: The name of the user.
- email: The email address of the user.
- password: The hashed password of the user for security measures.
- user_type : A flag indicating whether the user is an administrator or a guest account.



	id	name	email	password	user_type
Edit Copy Delete	10	admin A	admin01@gmail.com	698d51a19d8a121ce581499d7b701668	admin
Edit Copy Delete	14	user A	user01@gmail.com	698d51a19d8a121ce581499d7b701668	user
Edit Copy Delete	15	user B	user02@gmail.com	698d51a19d8a121ce581499d7b701668	user
Edit Copy Delete	16	Murad	muradjumabekov@gmail.com	09985dc8844eadaa72deaa4169763299	user

3.5. Wishlist Table

The wishlist table allows users to save products they are interested in for future reference. It includes attributes such as:

- id: A unique identification number for each wishlist item.

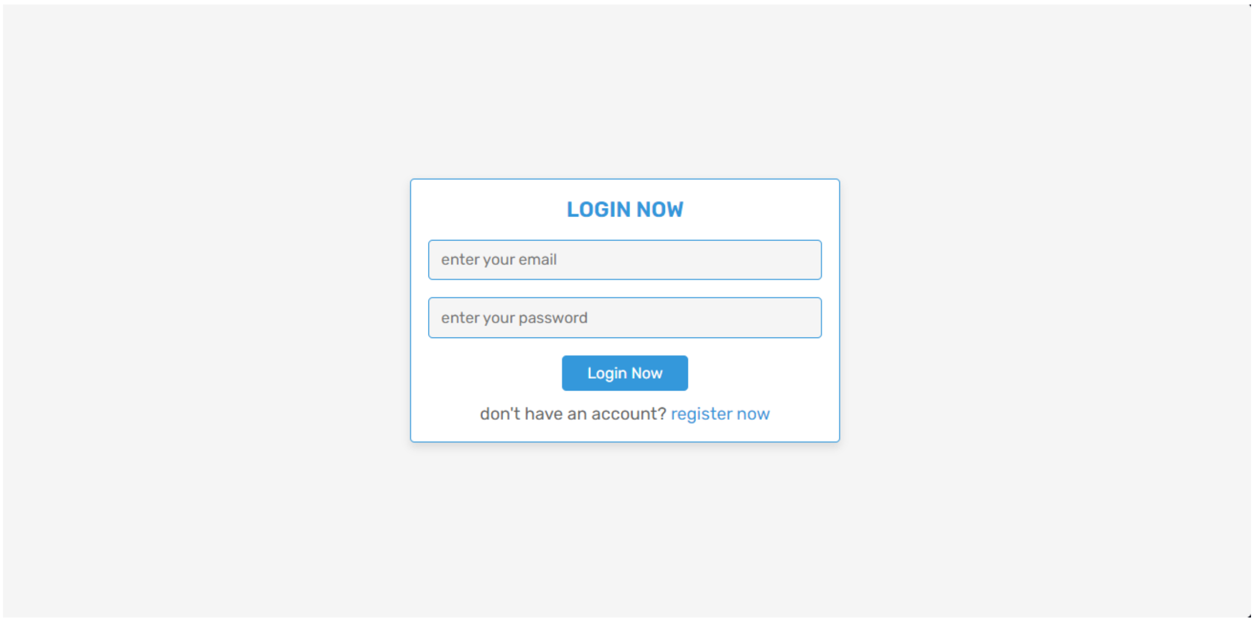
- user_id: The identification number of the user who added the product to the wishlist.

- price: The price of the product added to the wishlist.

- image: The image of the product added to the wishlist.

By organizing the database into these tables, we can efficiently manage user accounts, product inventory, shopping carts, orders, and wishlists within the online store application.

4. Web Pages



1. Login Button:

   - This button is used to initiate the login process. Users click on it after entering their credentials (email and password) to access the system.

2. Email and Password Fields:

   - These fields allow users to input their login credentials.

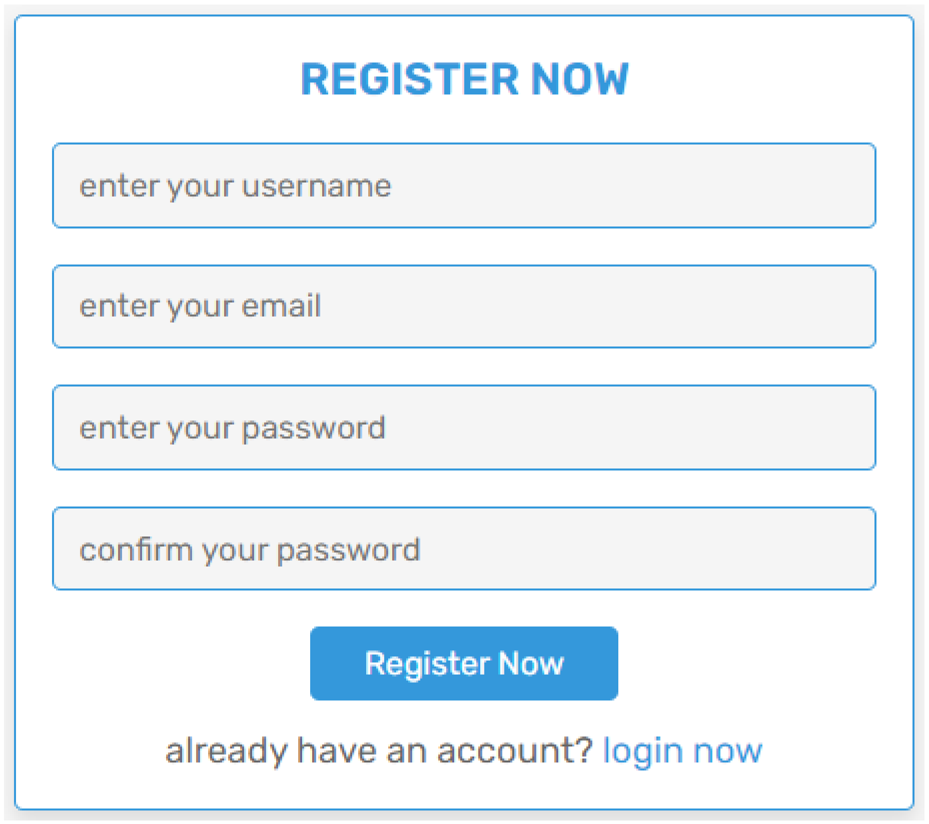
   - The email field typically serves as the username or identifier for the user.

   - The password field is where users enter their secret password for authentication purposes.

3. "Register Now" Link:

   - This link provides users with the option to register for a new account if they don't already have one.

   - Clicking on this link would typically redirect users to a registration page where they can fill out a form to create a new account by providing necessary information such as name, email, and password.



1. Enter Your Username:

- This field allows users to choose a username or identifier that they will use to log in to the system. It's often a unique name that distinguishes them from other users.

2. Enter Your Email:

- Users provide their email address in this field. The email serves as a means of communication and often as a unique identifier for the user within the system.

3. Enter Your Password:

- Here, users input the password they wish to use for accessing their account. Passwords are typically required to meet certain criteria for security reasons, such as containing a combination of letters, numbers, and special characters.

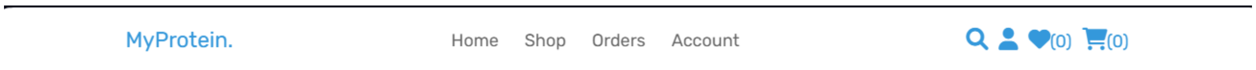
4. Confirm Your Password:

- This field asks users to re-enter their chosen password to ensure they've typed it correctly and to prevent errors during registration.

5. Register Now Button:

- Clicking on this button submits the information provided in the form for account creation.

- After clicking the button, the system typically processes the data, checks for any errors or missing information, and then registers the user if everything is in order.



1. Header Buttons:

- Home: This button redirects users to the main page of the website.

- Shop: Clicking here takes users to the product catalog, where they can explore and purchase sport nutrition items.

- Orders: Users can track past orders, monitor current ones, and manage order history by clicking on this button.

- Account: Access to personal account features, including updating information and managing orders, is available by selecting this button.

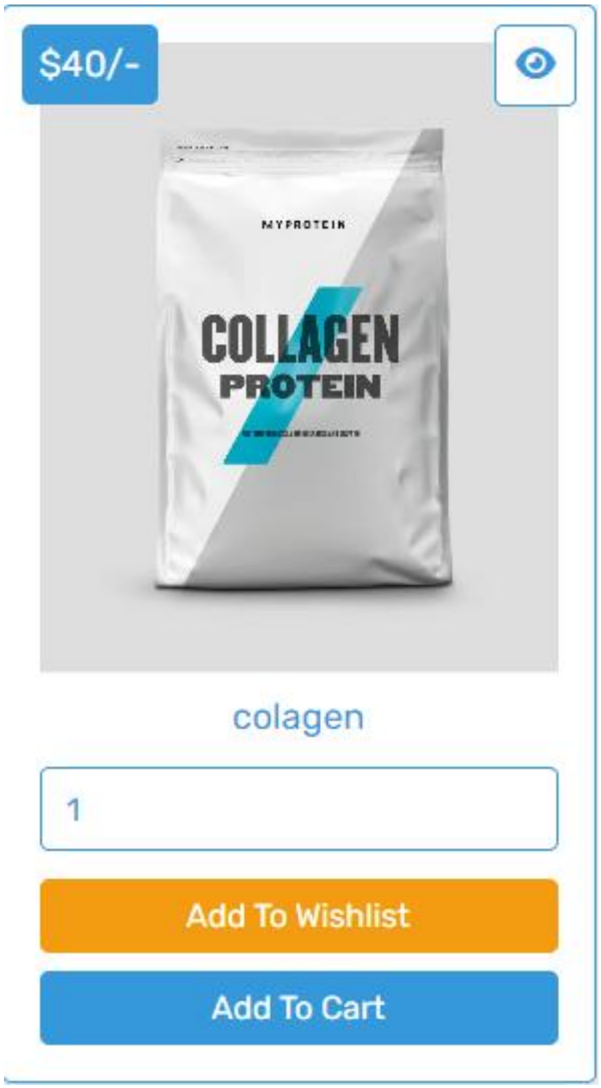
2. Symbols on the Right Corner:

- Search Symbol: Allows users to search for specific products, categories, or information across the website.

- Profile Symbol: Redirects users to their personal account page for managing profile details and tracking orders.

- Wishlist Symbol: Provides access to the wishlist feature, allowing users to track favorite items.

- Cart Symbol: Displays the number of items in the shopping cart and allows users to review selections and proceed to checkout.



1. Price:

- The price of each product is clearly indicated on its respective card, providing users with transparency and clarity regarding pricing information.

2. Quantity Regulation Button:

- This button allows users to adjust the quantity of the product they wish to purchase before adding it to their cart.

3. "Add to Wishlist" Button:

- Users can add products to their wishlist by clicking on this button, enabling them to save items for future consideration or purchase.

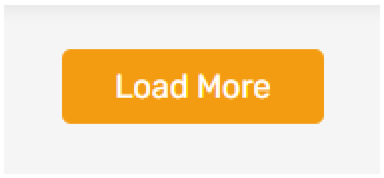
4. "Add to Cart" Button:

   - Clicking on this button adds the selected product to the user's shopping cart, allowing for easy and convenient checkout.

5. Eye Symbol (Preview Button):

   - The eye symbol serves as a preview button, providing users with a quick glance at the product and additional details without navigating away from the current page.

Upon clicking the eye symbol for a specific product, users can access a preview window showcasing more detailed information about the product, aiding in their decision-making process.

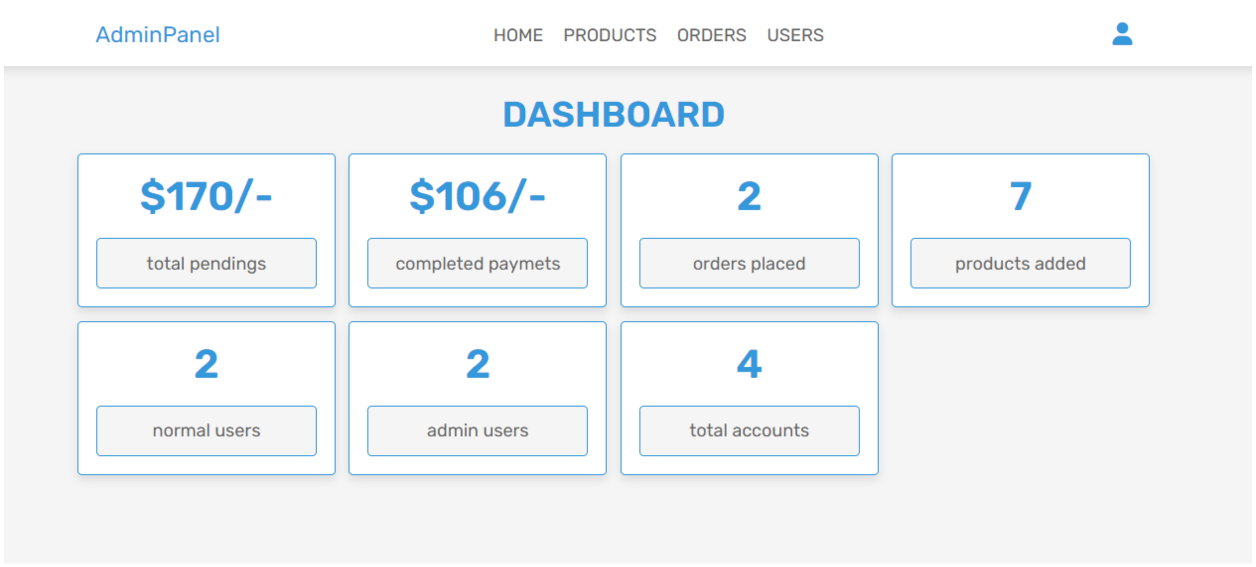


"Load More" Button:

- The "Load More" button is a user interface element designed to facilitate the loading of additional products on the page without navigating to a new page or refreshing the current one.

- When users reach the end of the current product listing, they have the option to click the "Load More" button to dynamically load more products onto the page, expanding the selection and providing users with further options to explore.

- This feature enhances the user experience by allowing for seamless browsing and discovery of products without disruptions or page reloads.


AdminPanel

HOME PRODUCTS ORDERS USERS

DASHBOARD

$170/- total pendings

$106/- completed paymets

2 orders placed

7 products added

2 normal users

2 admin users

4 total accounts

Within the admin panel of our online shop, accessible exclusively by administrators, it consists of several navigation options:

1. Home: This link likely serves as a quick way to return to the main dashboard or landing page of the admin panel.

2. Products: This section provides access to manage the products available for sale on the platform. Administrators can add, edit, or remove products from the catalog using this feature.

3. Orders: Administrators can monitor and manage orders placed by customers through this section. It allows for tracking order statuses, managing shipments, and handling any customer inquiries related to orders.

4. Users: This section enables administrators to manage user accounts and permissions. It includes features such as creating new user accounts, modifying existing accounts, and managing user roles and access levels.

5. Dashboard: The dashboard serves as a centralized hub for administrators to access key statistics and insights about the performance of the website. It provides a snapshot of various metrics, including:

- Total pendings: The number of pending orders or tasks that require action.

- Completed payments: The number of payments successfully processed.

- Orders placed: The total number of orders received.

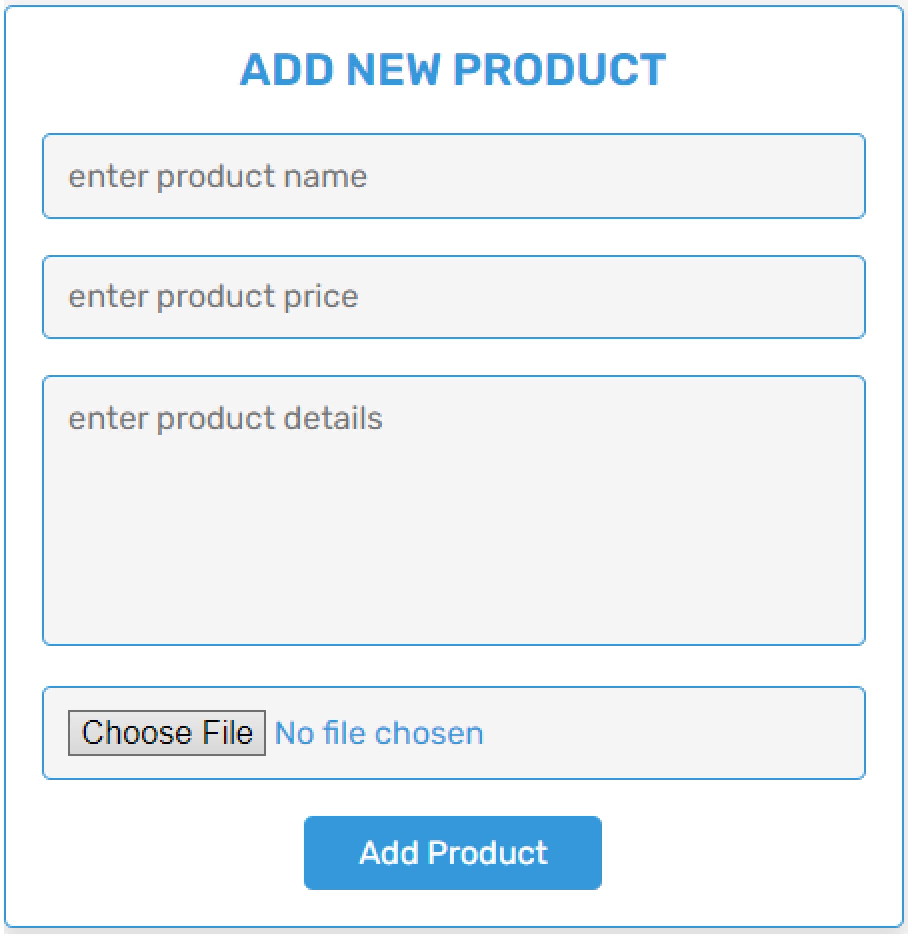
- Products added: The number of new products added to the catalog.

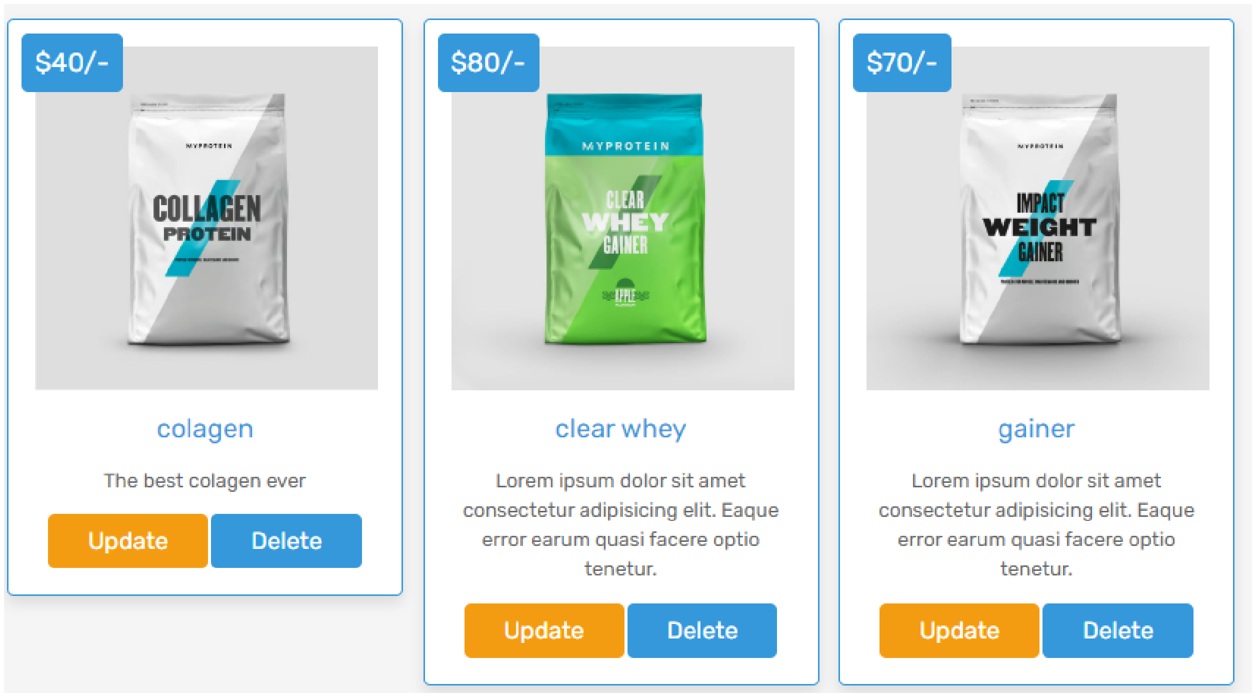
- Normal users: The count of regular user accounts registered on the platform.

- Admin users: The count of administrator accounts with access to the admin panel.

- Total accounts: The overall number of user accounts registered on the platform, including both normal users and administrators.

This dashboard feature offers administrators valuable insights into the performance and activity of the online shop, allowing for informed decision-making and effective management of the platform.




$40/-

colagen

The best colagen ever

$80/-

clear whey

Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque error earum quasi facere optio tenetur.

$70/-

gainer

Lorem ipsum dolor sit amet consectetur adipisicing elit. Eaque error earum quasi facere optio tenetur.

In the "Products" tab within the admin panel, administrators have the capability to perform various actions related to managing programs. These actions typically include:

1. Adding Programs:

  - Administrators can add new programs to the catalog by providing relevant details such as program name, description, price, and any other pertinent information.

  - They may also upload images or multimedia content to enhance the program listing and attract user interest.

2. Updating Programs:

  - Admins have the ability to update existing program listings with any necessary modifications or changes.

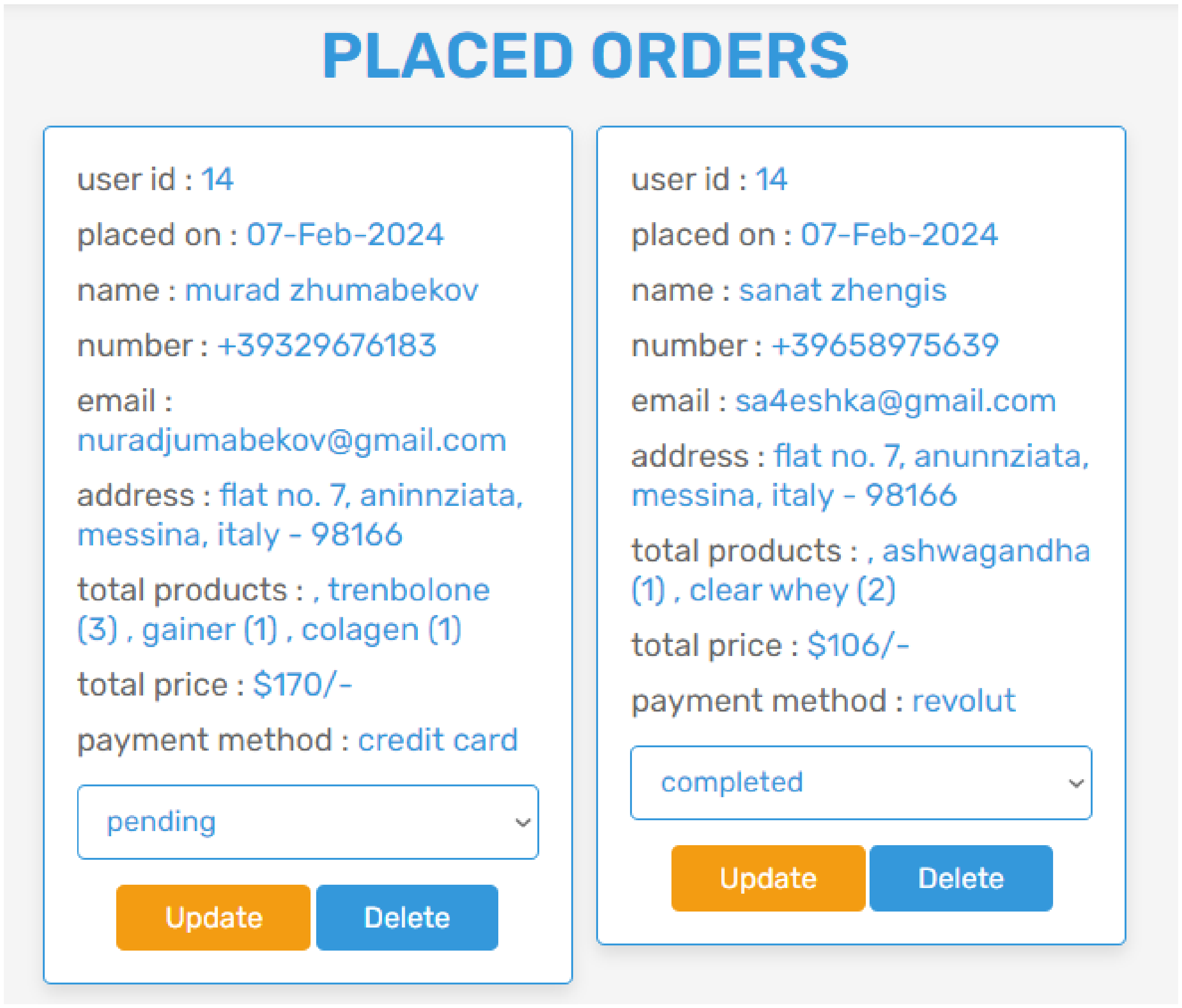
  - This could involve editing program details, adjusting pricing, updating descriptions, or modifying other attributes to ensure accuracy and relevance.

3. Deleting Programs:

  - In cases where a program is no longer offered or relevant, administrators can delete it from the catalog.

- Deleting a program removes it from the available listings and prevents users from accessing or purchasing it.

Through these functionalities, administrators maintain control over the programs offered within the online shop, ensuring that the catalog remains up-to-date, accurate, and reflective of the current offerings. This level of control allows administrators to effectively manage the program inventory and tailor the selection to meet the needs and preferences of users.



PLACED ORDERS

user id : 14
placed on : 07-Feb-2024
name : murad zhumabekov
number : +39329676183
email : nuradjumabekov@gmail.com
address : flat no. 7, aninnziata, messina, italy - 98166
total products : , trenbolone (3) , gainer (1) , colagen (1)
total price : $170/-
payment method : credit card

user id : 14
placed on : 07-Feb-2024
name : sanat zhengis
number : +39658975639
email : sa4eshka@gmail.com
address : flat no. 7, anunnziata, messina, italy - 98166
total products : , ashwagandha (1) , clear whey (2)
total price : $106/-
payment method : revolut

In the "Orders" tab of the admin panel, administrators have access to comprehensive information regarding each order placed on the online shop. This information typically includes:

1. Order Details:

- Administrators can view specific details about each order, such as order number, date and time of purchase, and order status (e.g., pending, processing, shipped, completed).

2. Customer Information:

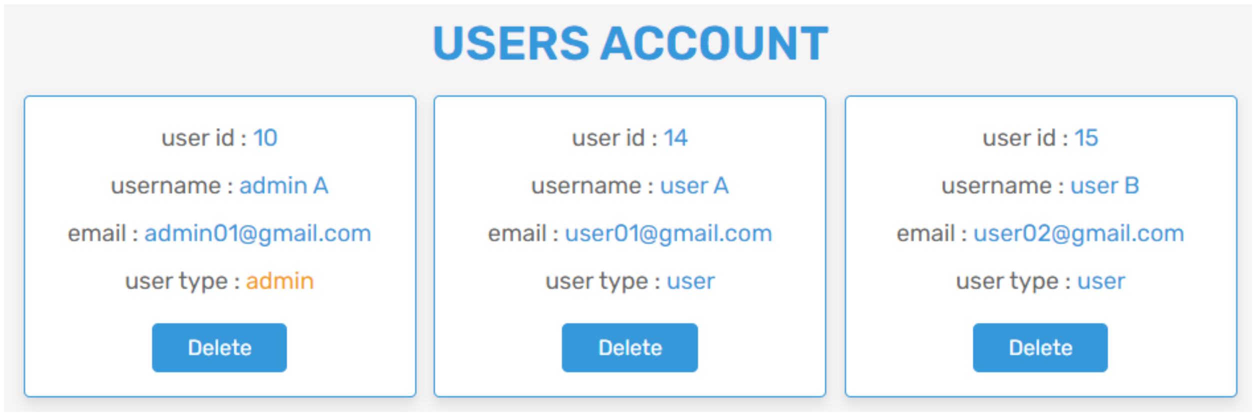
- The order information typically includes details about the customer who placed the order, including their name, contact information, and shipping address.

3. Product Details:

- Administrators can see a breakdown of the products included in each order, including product names, quantities ordered, and prices.

4. Payment Information:

- Details about the payment method used for the order, such as credit card information or payment gateway details, may also be available.


USERS ACCOUNT

user id : 10
username : admin A
email : admin01@gmail.com
user type : admin

user id : 14
username : user A
email : user01@gmail.com
user type : user

user id : 15
username : user B
email : user02@gmail.com
user type : user

In the "Users" tab of the admin panel, administrators have access to a comprehensive list of all users registered on the online platform, including other administrators. Here's what administrators typically can see and do within this tab:

1. User Listing:

- Administrators can view a list of all registered users, including both regular users and other administrators with access to the admin panel.

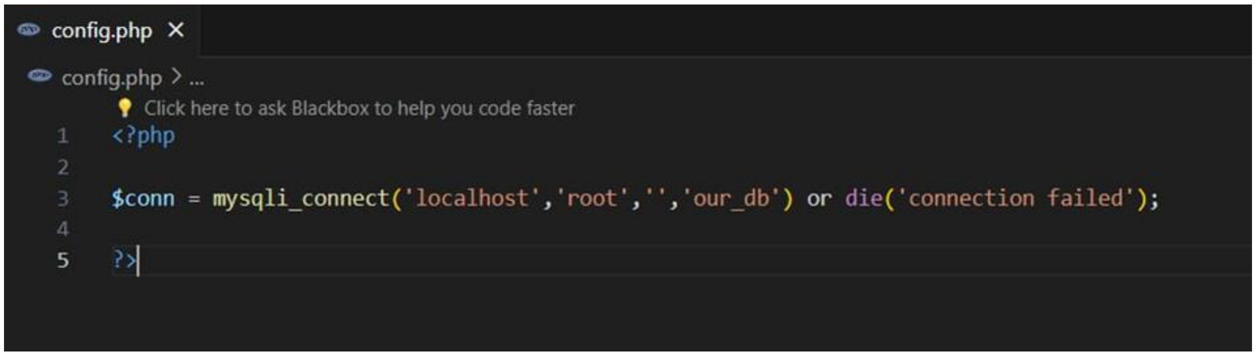
2. User Details:

 - For each user, administrators can access detailed information such as username, email address, contact information, and account creation date

5 Key Features

5.1 Database connection

This .php file is designed to establish a connection with the database named "our_db." The provided code follows a standard practice for connecting to a specific database hosted on localhost. In the subsequent development of the project, other .php files will utilize this central file to validate the connection to the database, eliminating the need to repeatedly include this code block in each individual file. This approach not only conserves space but also enhances the efficiency of coding and overall project design.



```php
<?php

$conn = mysqli_connect('localhost','root','','our_db') or die('connection failed');

?>
```

5.2 Login

The provided PHP code manages the login functionality within the HTML index page. It connects to the database, processes user sessions, and handles form submissions. Here's an overview:

PHP Code:

- Database Connection:

 - The config.php file is included to establish a database connection using the mysqli extension.

- Session Management:

  - session_start() initiates a session to manage user sessions.

- Form Submission Handling:

  - Checks if the form is submitted (isset($_POST['submit'])).

  - Filters and sanitizes user inputs (email and password).

  - Queries the database to validate the user's credentials.

  - Sets session variables based on the user type (admin or user).

  - Redirects users to their respective pages (admin_page.php or home.php) upon successful login.

  - Displays appropriate messages for different scenarios (e.g., incorrect email or password).

HTML Code:

- Head Section:

  - Declares the doctype and includes metadata for character set, viewport settings, and IE compatibility.

  - Links external stylesheets for font awesome and a custom style.css file.

- Body Section:

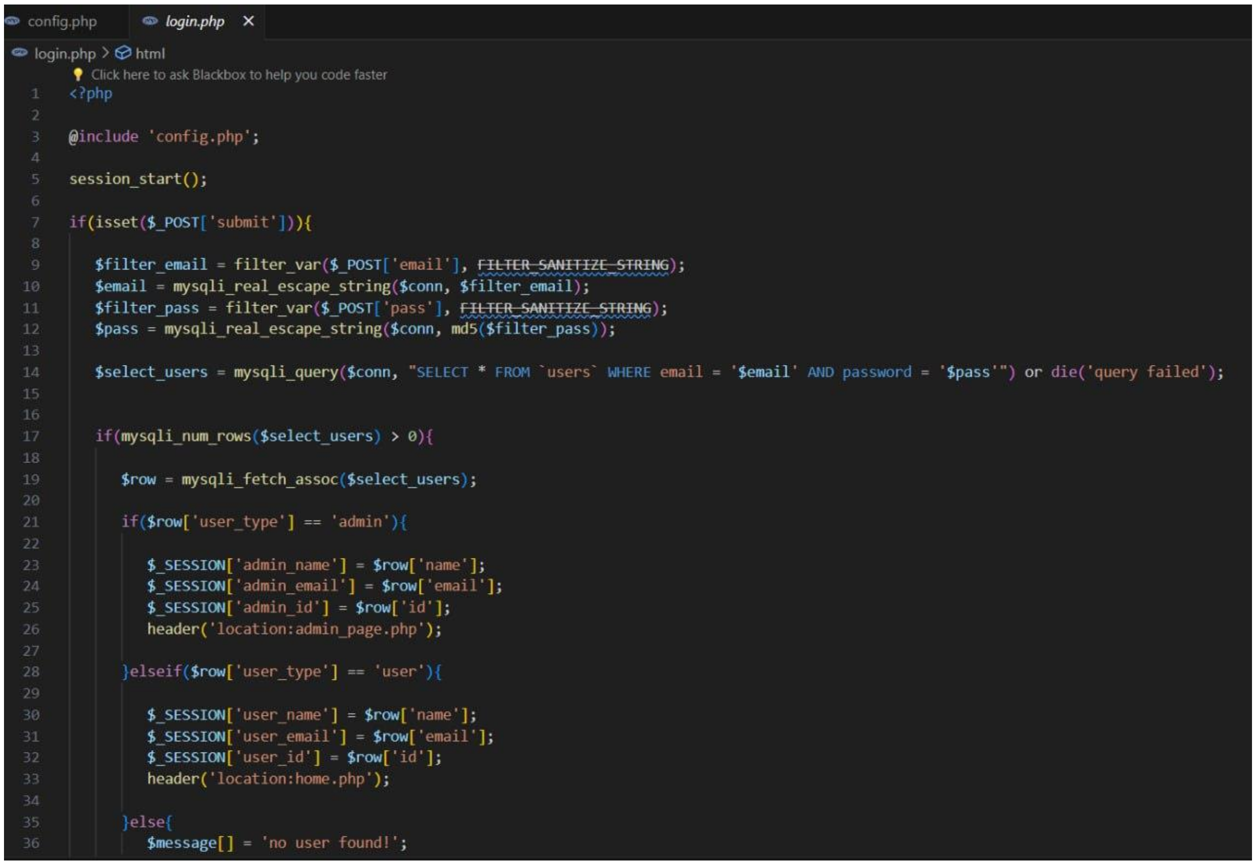
  - Displays error messages if any exist.

  - Defines the HTML form to collect user credentials using the POST method.

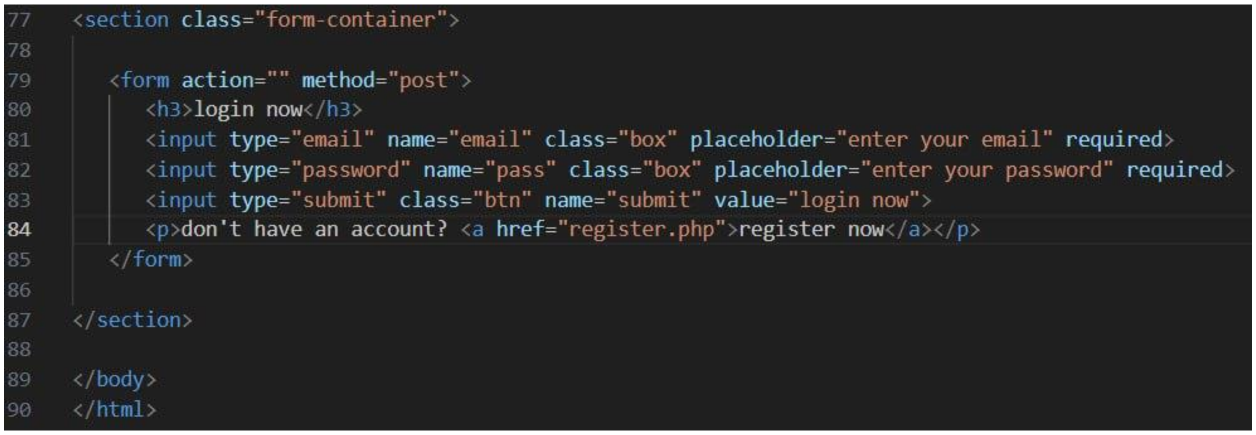
  - Processes user input with the embedded PHP code.

  - Provides a link for users without accounts to navigate to the registration page.

Overall, the login page combines HTML and PHP effectively to create a functional and secure authentication mechanism. The separation of concerns, with PHP handling backend logic and HTML managing the presentation, enhances code maintainability and organization.


config.php login.php ✕
login.php > html
Click here to ask Blackbox to help you code faster

```php
<?php

@include 'config.php';

session_start();

if(isset($_POST['submit'])){

   $filter_email = filter_var($_POST['email'], ~~FILTER_SANITIZE_STRING~~);
   $email = mysqli_real_escape_string($conn, $filter_email);
   $filter_pass = filter_var($_POST['pass'], ~~FILTER_SANITIZE_STRING~~);
   $pass = mysqli_real_escape_string($conn, md5($filter_pass));

   $select_users = mysqli_query($conn, "SELECT * FROM `users` WHERE email = '$email' AND password = '$pass'") or die('query failed');


   if(mysqli_num_rows($select_users) > 0){

      $row = mysqli_fetch_assoc($select_users);

      if($row['user_type'] == 'admin'){

         $_SESSION['admin_name'] = $row['name'];
         $_SESSION['admin_email'] = $row['email'];
         $_SESSION['admin_id'] = $row['id'];
         header('location:admin_page.php');

      }elseif($row['user_type'] == 'user'){

         $_SESSION['user_name'] = $row['name'];
         $_SESSION['user_email'] = $row['email'];
         $_SESSION['user_id'] = $row['id'];
         header('location:home.php');

      }else{
         $message[] = 'no user found!';
```



```html
<section class="form-container">

   <form action="" method="post">
      <h3>login now</h3>
      <input type="email" name="email" class="box" placeholder="enter your email" required>
      <input type="password" name="pass" class="box" placeholder="enter your password" required>
      <input type="submit" class="btn" name="submit" value="login now">
      <p>don't have an account? <a href="register.php">register now</a></p>
   </form>

</section>

</body>
</html>
```

5.3 Registration

PHP Code:

- Database Connection:

  - The config.php file is included to establish a database connection using the mysqli extension.

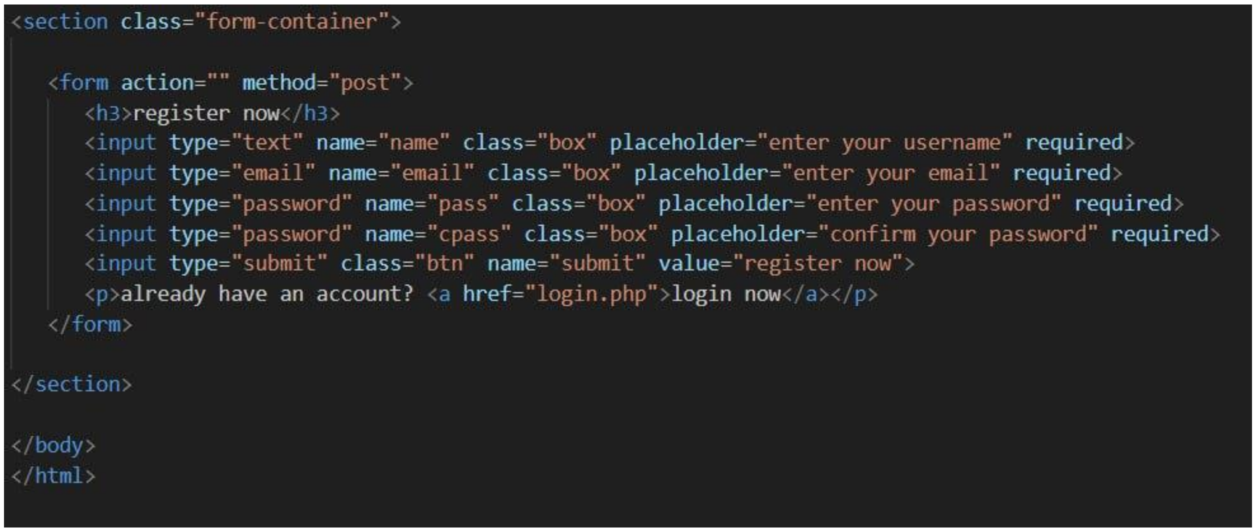
- Form Submission Handling:
  - Checks if the form is submitted (isset($_POST['submit'])).
  - Filters and sanitizes user inputs (name, email, password, confirm password).
  - Queries the database to check if the email already exists.
  - Compares the entered password with the confirmed password.
  - Inserts user information into the database if all conditions are met.
  - Displays appropriate messages for different scenarios (e.g., user already exists, password mismatch, successful registration).
  - Redirects users to the login page upon successful registration.

HTML Code:

- Head Section:
  - Declares the doctype and includes metadata for character set, viewport settings, and IE compatibility.
  - Links external stylesheets for font awesome and a custom style.css file.

- Body Section:
  - Displays error messages if any exist.
  - Defines the HTML form to collect user registration details using the POST method.
  - Processes user input with the embedded PHP code.
  - Provides a link for users with existing accounts to navigate to the login page.

Overall, the registration page combines HTML and PHP effectively to create a secure user registration mechanism. The separation of concerns, with PHP handling backend logic and HTML managing the presentation, enhances code maintainability and organization.



```
<section class="form-container">

   <form action="" method="post">
      <h3>register now</h3>
      <input type="text" name="name" class="box" placeholder="enter your username" required>
      <input type="email" name="email" class="box" placeholder="enter your email" required>
      <input type="password" name="pass" class="box" placeholder="enter your password" required>
      <input type="password" name="cpass" class="box" placeholder="confirm your password" required>
      <input type="submit" class="btn" name="submit" value="register now">
      <p>already have an account? <a href="login.php">login now</a></p>
   </form>

</section>

</body>
</html>
```

5.4 Admin privileges

JavaScript Code:

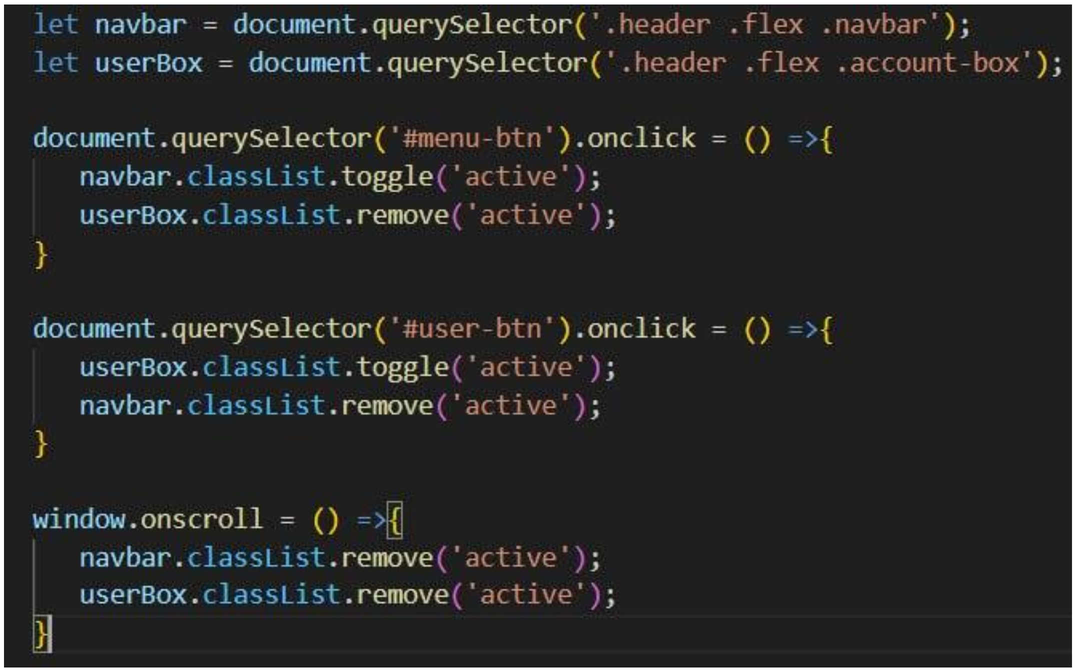
1. Navbar and User Box Interaction:

  - The script responds to user actions on the webpage.

  - When the menu button (menu-btn) is clicked, it toggles the visibility of the navigation bar (navbar), making it active or inactive.

  - Clicking on the user button (user-btn) toggles the user box (userBox) visibility, allowing users to interact with their account settings.

  - When the user scrolls down the page, both the navbar and user box are set to inactive to provide a clean and user-friendly experience.



```javascript
let navbar = document.querySelector('.header .flex .navbar');
let userBox = document.querySelector('.header .flex .account-box');

document.querySelector('#menu-btn').onclick = () =>{
   navbar.classList.toggle('active');
   userBox.classList.remove('active');
}

document.querySelector('#user-btn').onclick = () =>{
   userBox.classList.toggle('active');
   navbar.classList.remove('active');
}

window.onscroll = () =>{
   navbar.classList.remove('active');
   userBox.classList.remove('active');
}
```

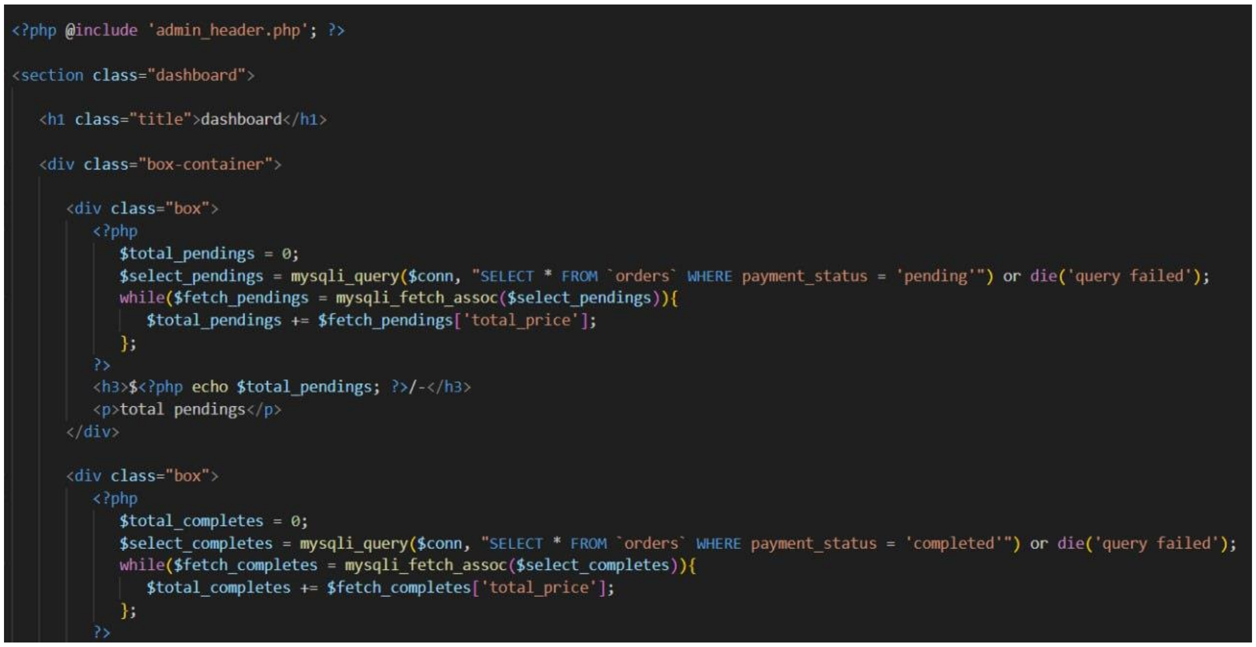
PHP Code (admin_update_product.php):

1. Admin Authorization Check:

- This PHP block ensures that only users with admin privileges can access the update product page.
- If a user lacks admin privileges, they are redirected to the login page for security reasons.

2. Update Product Handling:

- The PHP script retrieves the details of a specific product based on the provided update ID from the URL.
- It presents a form to the admin for updating the product details, including name, price, details, and image.
- Upon submitting the form:
    - Product information is updated in the database.
    - The script manages the upload and update of product images.
    - Success or error messages are displayed to inform the admin about the outcome.



```
<?php @include 'admin_header.php'; ?>

<section class="dashboard">

   <h1 class="title">dashboard</h1>

   <div class="box-container">

      <div class="box">
         <?php
            $total_pendings = 0;
            $select_pendings = mysqli_query($conn, "SELECT * FROM `orders` WHERE payment_status = 'pending'") or die('query failed');
            while($fetch_pendings = mysqli_fetch_assoc($select_pendings)){
               $total_pendings += $fetch_pendings['total_price'];
            };
         ?>
         <h3>$<?php echo $total_pendings; ?>/-</h3>
         <p>total pendings</p>
      </div>

      <div class="box">
         <?php
            $total_completes = 0;
            $select_completes = mysqli_query($conn, "SELECT * FROM `orders` WHERE payment_status = 'completed'") or die('query failed');
            while($fetch_completes = mysqli_fetch_assoc($select_completes)){
               $total_completes += $fetch_completes['total_price'];
            };
         ?>
```

PHP Code (admin_users.php):

1. Admin Authorization Check:

   - This PHP block checks if the current user has admin privileges.

   - If not, the user is redirected to the login page to maintain secure access control.

2. User Deletion Handling:

   - The script fetches a list of users from the database.

   - It displays user information within a container, providing options to delete individual users.

   - Clicking the delete button triggers a SQL DELETE query to remove the selected user from the database.

PHP Code (dashboard.php):

1. Admin Authorization Check:

   - Ensures that only users with admin privileges can access the dashboard.

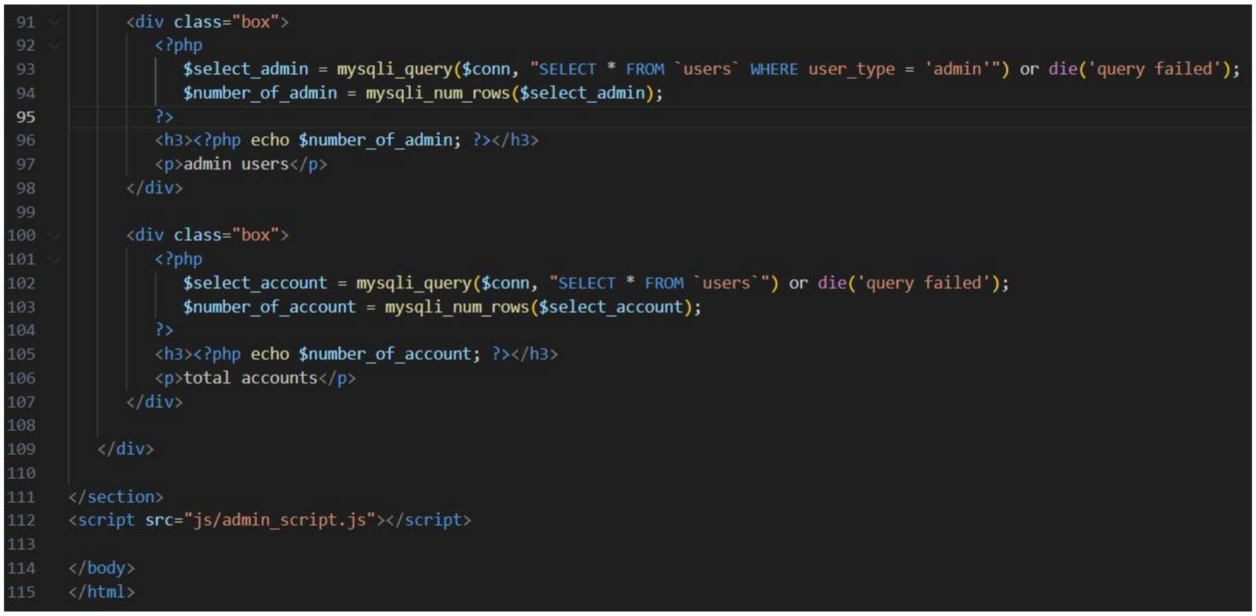
   - If a user lacks admin privileges, they are redirected to the login page for security reasons.

2. Dashboard Statistics:

- Fetches and presents various statistics:

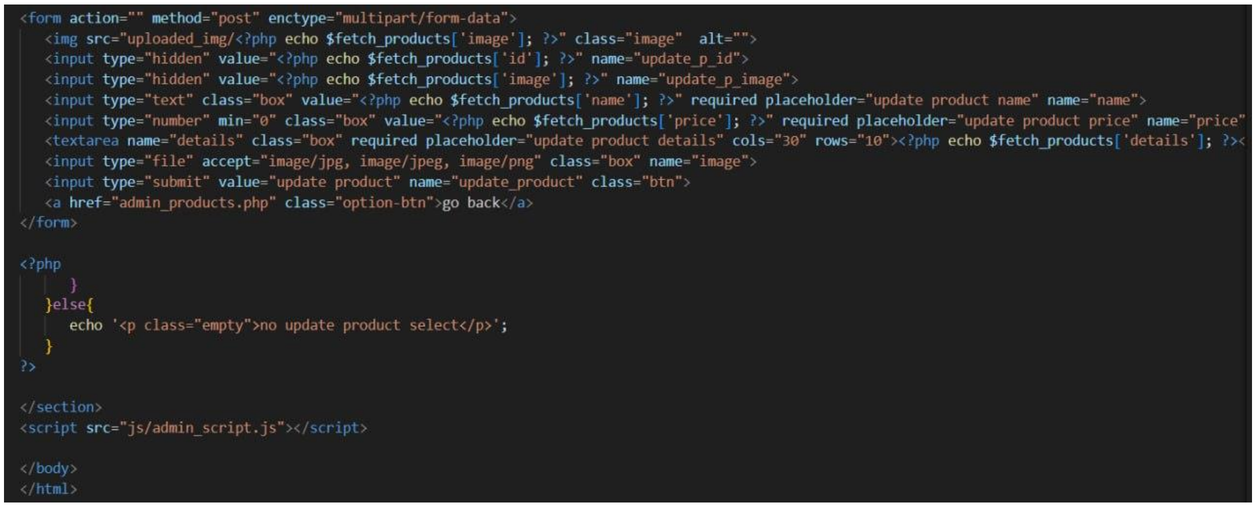
  - Total pending payments, completed payments, orders placed, products added, normal users, admin users, and total accounts.

- The displayed statistics offer a quick overview of the website's performance and user activities.



```
      <div class="box">
         <?php
            $select_admin = mysqli_query($conn, "SELECT * FROM `users` WHERE user_type = 'admin'") or die('query failed');
            $number_of_admin = mysqli_num_rows($select_admin);
         ?>
         <h3><?php echo $number_of_admin; ?></h3>
         <p>admin users</p>
      </div>

      <div class="box">
         <?php
            $select_account = mysqli_query($conn, "SELECT * FROM `users`") or die('query failed');
            $number_of_account = mysqli_num_rows($select_account);
         ?>
         <h3><?php echo $number_of_account; ?></h3>
         <p>total accounts</p>
      </div>

   </div>

</section>
<script src="js/admin_script.js"></script>

</body>
</html>
```

This approach ensures a seamless and secure experience for admin users, with organized and informative statistics on the dashboard. The division of functionality enhances code clarity and maintenance.



```
<form action="" method="post" enctype="multipart/form-data">
   <img src="uploaded_img/<?php echo $fetch_products['image']; ?>" class="image"  alt="">
   <input type="hidden" value="<?php echo $fetch_products['id']; ?>" name="update_p_id">
   <input type="hidden" value="<?php echo $fetch_products['image']; ?>" name="update_p_image">
   <input type="text" class="box" value="<?php echo $fetch_products['name']; ?>" required placeholder="update product name" name="name">
   <input type="number" min="0" class="box" value="<?php echo $fetch_products['price']; ?>" required placeholder="update product price" name="price"
   <textarea name="details" class="box" required placeholder="update product details" cols="30" rows="10"><?php echo $fetch_products['details']; ?><
   <input type="file" accept="image/jpg, image/jpeg, image/png" class="box" name="image">
   <input type="submit" value="update product" name="update_product" class="btn">
   <a href="admin_products.php" class="option-btn">go back</a>
</form>

<?php
      }
   }else{
      echo '<p class="empty">no update product select</p>';
   }
?>

</section>
<script src="js/admin_script.js"></script>

</body>
</html>
```

5.5 User's Home page

Here we delve into the functionality of the E-commerce platform, specifically focusing on user interactions and product management. The PHP and JavaScript code presented here facilitates a smooth user experience, allowing users to interact with products by adding them to their wishlist or cart. Additionally, the chapter covers the display of the latest products on the home and shop pages.

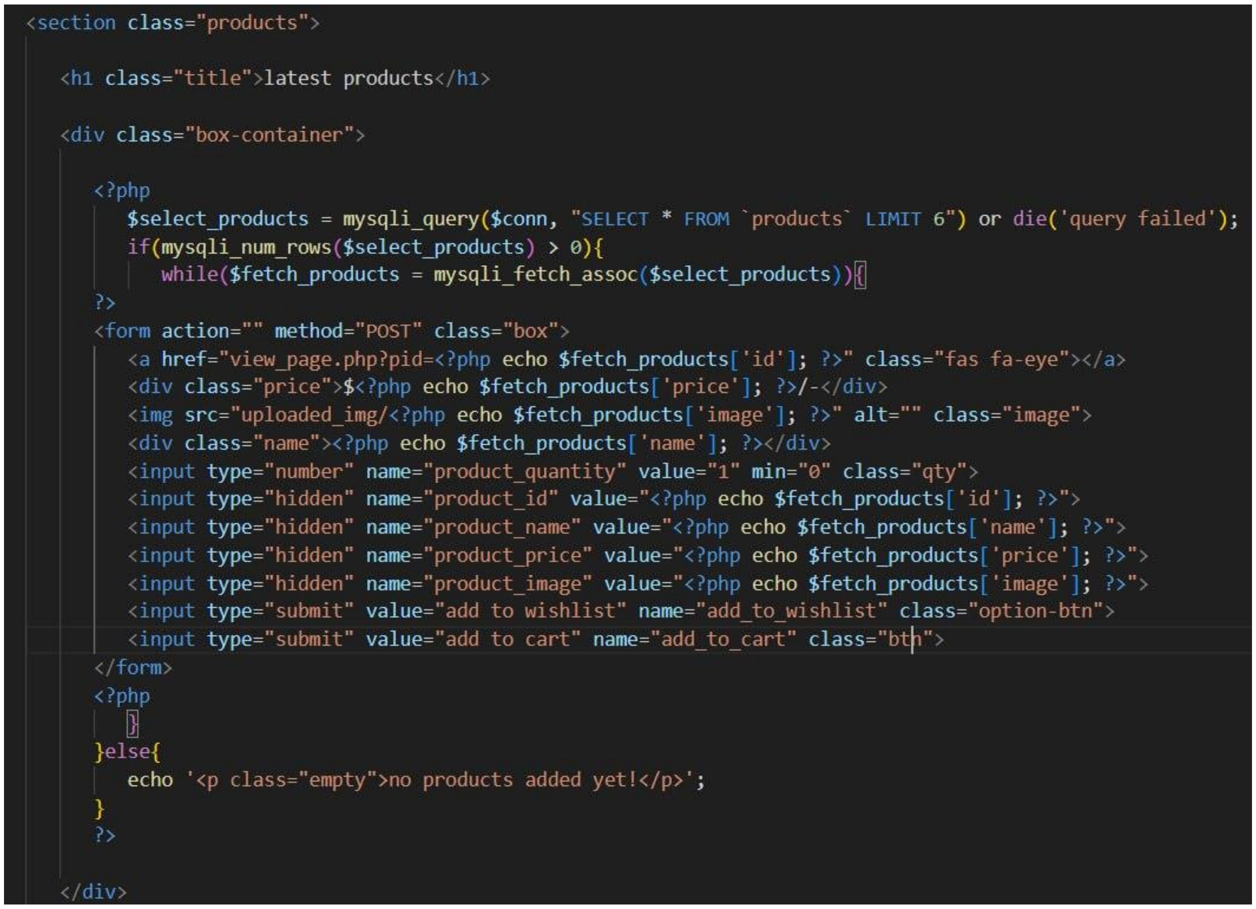
The PHP script initiates with session management and user authorization, ensuring a secure environment. Users are redirected to the login page if not authenticated. The script then manages wishlist and cart interactions, preventing duplicate entries and providing informative messages for user feedback.

The HTML structure defines sections for home and product display. The home section introduces new collections, enhancing the user experience with a brief description. The latest products section dynamically fetches information from the database, presenting a variety of products. Each product is accompanied by a form for users to view details, specify quantity, and add the product to their wishlist or cart.

The user interface is enriched with Font Awesome icons for actions such as viewing product details. CSS styling ensures a visually appealing and responsive design, contributing to an engaging platform.

JavaScript is employed for user interface enhancements, allowing for the toggling of the navigation bar and user box visibility in response to user clicks and scroll events.

In conclusion, this chapter provides an overview of the PHP and JavaScript code that governs user interactions and product management on the E-commerce platform. The integration of wishlist and cart functionalities, coupled with an attractive display of the latest products, collectively enhances the overall user experience.



```
<section class="products">

   <h1 class="title">latest products</h1>

   <div class="box-container">

      <?php
         $select_products = mysqli_query($conn, "SELECT * FROM `products` LIMIT 6") or die('query failed');
         if(mysqli_num_rows($select_products) > 0){
            while($fetch_products = mysqli_fetch_assoc($select_products)){
      ?>
      <form action="" method="POST" class="box">
         <a href="view_page.php?pid=<?php echo $fetch_products['id']; ?>" class="fas fa-eye"></a>
         <div class="price">$<?php echo $fetch_products['price']; ?>/-</div>
         <img src="uploaded_img/<?php echo $fetch_products['image']; ?>" alt="" class="image">
         <div class="name"><?php echo $fetch_products['name']; ?></div>
         <input type="number" name="product_quantity" value="1" min="0" class="qty">
         <input type="hidden" name="product_id" value="<?php echo $fetch_products['id']; ?>">
         <input type="hidden" name="product_name" value="<?php echo $fetch_products['name']; ?>">
         <input type="hidden" name="product_price" value="<?php echo $fetch_products['price']; ?>">
         <input type="hidden" name="product_image" value="<?php echo $fetch_products['image']; ?>">
         <input type="submit" value="add to wishlist" name="add_to_wishlist" class="option-btn">
         <input type="submit" value="add to cart" name="add_to_cart" class="btn">
      </form>
      <?php
         }
      }else{
         echo '<p class="empty">no products added yet!</p>';
      }
      ?>

   </div>
```

5.6 Search page

The Search Page functionality within the user interface of our E-commerce platform. The HTML architecture is meticulously organized into three pivotal sections: 'heading,' 'search-form,' and 'products.' Each section plays a crucial role in delivering a seamless and interactive experience for users engaged in product searches and exploration.

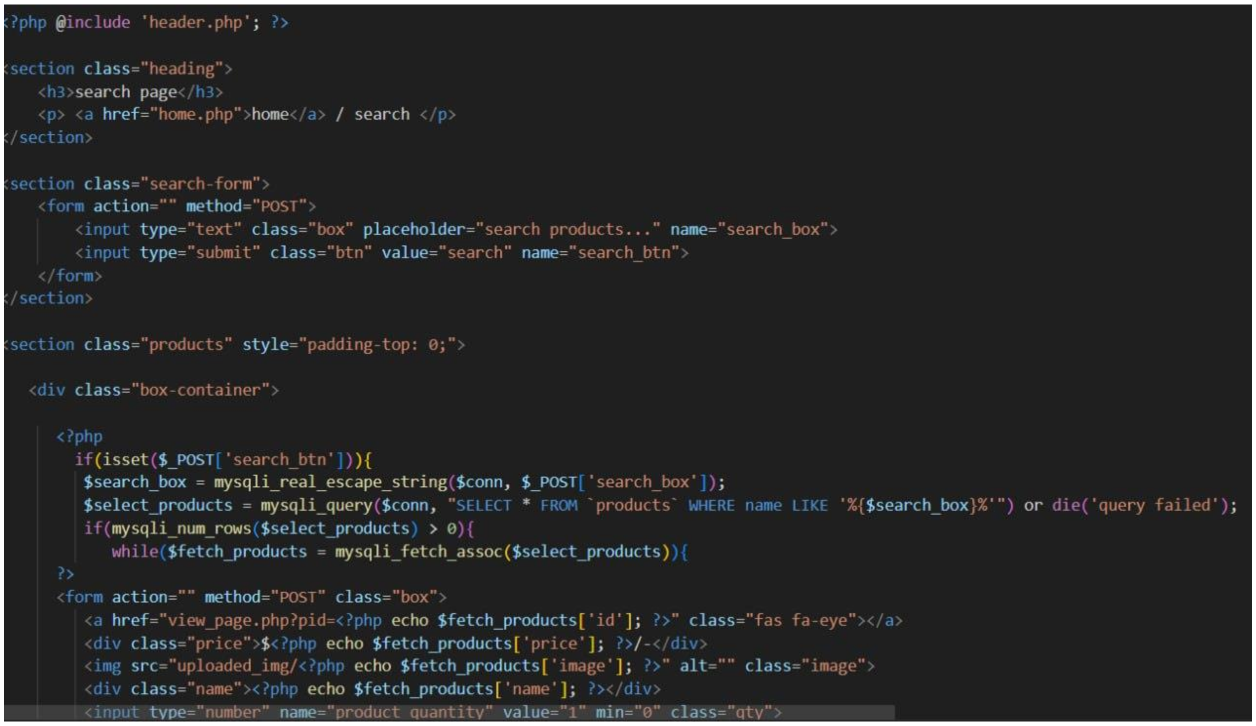
The 'heading' section serves as the gateway to the Search Page, offering users a clear and concise title alongside a navigation link that effortlessly guides them back to the home page. This strategic placement ensures users can navigate through the platform intuitively, enhancing overall user experience.

The 'search-form' section is designed with user-friendliness in mind, featuring a straightforward search input and a responsive button. Users can easily input their desired search queries, initiating product searches with a single click. This section serves as the starting point for users to explore the vast array of products available on our platform.

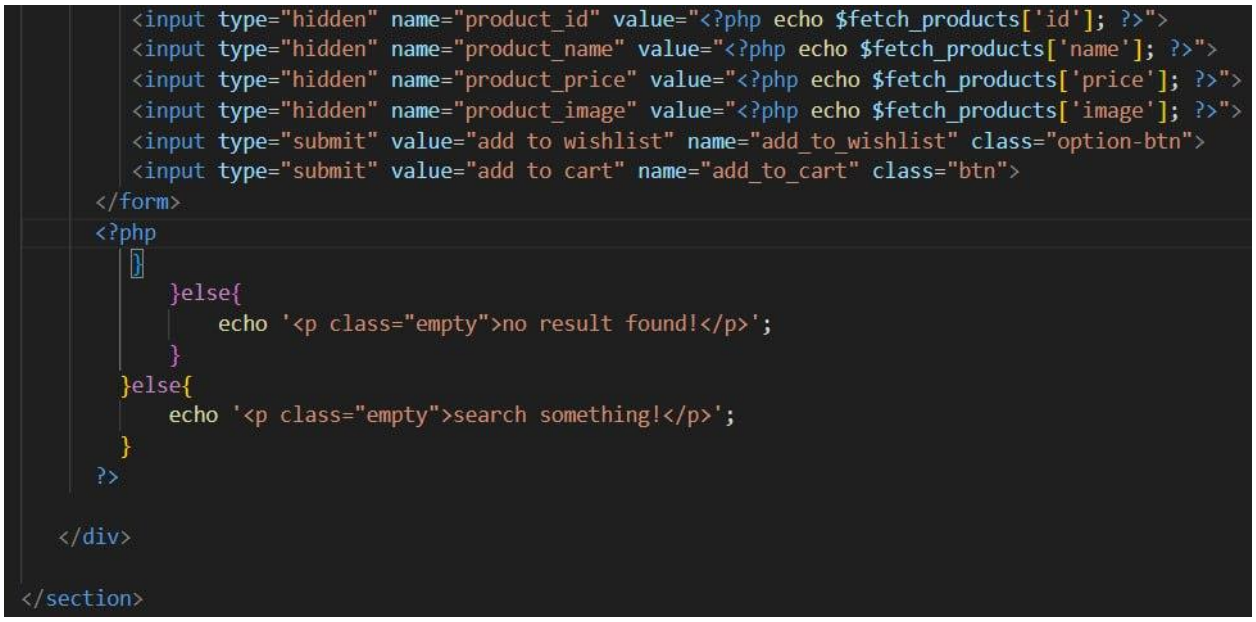
The 'products' section is the dynamic heart of the Search Page. Through PHP scripting, it dynamically retrieves and showcases products relevant to the user's search query. This real-time interaction empowers users to delve into product details, fostering an engaging environment. Additionally, users can seamlessly add discovered products to their wishlist or cart, contributing to a streamlined and efficient shopping experience.

This trio of sections is seamlessly integrated, incorporating Font Awesome icons for visual aesthetics and CSS styling to ensure a responsive and visually appealing design. JavaScript is employed to enhance the overall user interface, providing a platform that is not only functional but also visually captivating.

In summary, this chapter illuminates the intricate design and functionality of the Search Page, elucidating how its three main sections collaboratively contribute to an immersive and user-centric E-commerce experience.



```
<?php @include 'header.php'; ?>

<section class="heading">
    <h3>search page</h3>
    <p> <a href="home.php">home</a> / search </p>
</section>

<section class="search-form">
    <form action="" method="POST">
        <input type="text" class="box" placeholder="search products..." name="search_box">
        <input type="submit" class="btn" value="search" name="search_btn">
    </form>
</section>

<section class="products" style="padding-top: 0;">

   <div class="box-container">

      <?php
        if(isset($_POST['search_btn'])){
         $search_box = mysqli_real_escape_string($conn, $_POST['search_box']);
         $select_products = mysqli_query($conn, "SELECT * FROM `products` WHERE name LIKE '%{$search_box}%'") or die('query failed');
         if(mysqli_num_rows($select_products) > 0){
            while($fetch_products = mysqli_fetch_assoc($select_products)){
      ?>
      <form action="" method="POST" class="box">
         <a href="view_page.php?pid=<?php echo $fetch_products['id']; ?>" class="fas fa-eye"></a>
         <div class="price">$<?php echo $fetch_products['price']; ?>/-</div>
         <img src="uploaded_img/<?php echo $fetch_products['image']; ?>" alt="" class="image">
         <div class="name"><?php echo $fetch_products['name']; ?></div>
         <input type="number" name="product_quantity" value="1" min="0" class="qty">
```



```
            <input type="hidden" name="product_id" value="<?php echo $fetch_products['id']; ?>">
            <input type="hidden" name="product_name" value="<?php echo $fetch_products['name']; ?>">
            <input type="hidden" name="product_price" value="<?php echo $fetch_products['price']; ?>">
            <input type="hidden" name="product_image" value="<?php echo $fetch_products['image']; ?>">
            <input type="submit" value="add to wishlist" name="add_to_wishlist" class="option-btn">
            <input type="submit" value="add to cart" name="add_to_cart" class="btn">
        </form>
        <?php
            }
              }else{
                  echo '<p class="empty">no result found!</p>';
              }
          }else{
              echo '<p class="empty">search something!</p>';
          }
        ?>

    </div>

</section>
```

5.7 Shopping cart

The comprehensive overview of the functionality embedded in the 'cart.php' page for users on our E-commerce platform. The HTML structure is encapsulated within a 'shopping-cart' section, orchestrating a cohesive and visually engaging interface designed to showcase the products added to the user's cart.

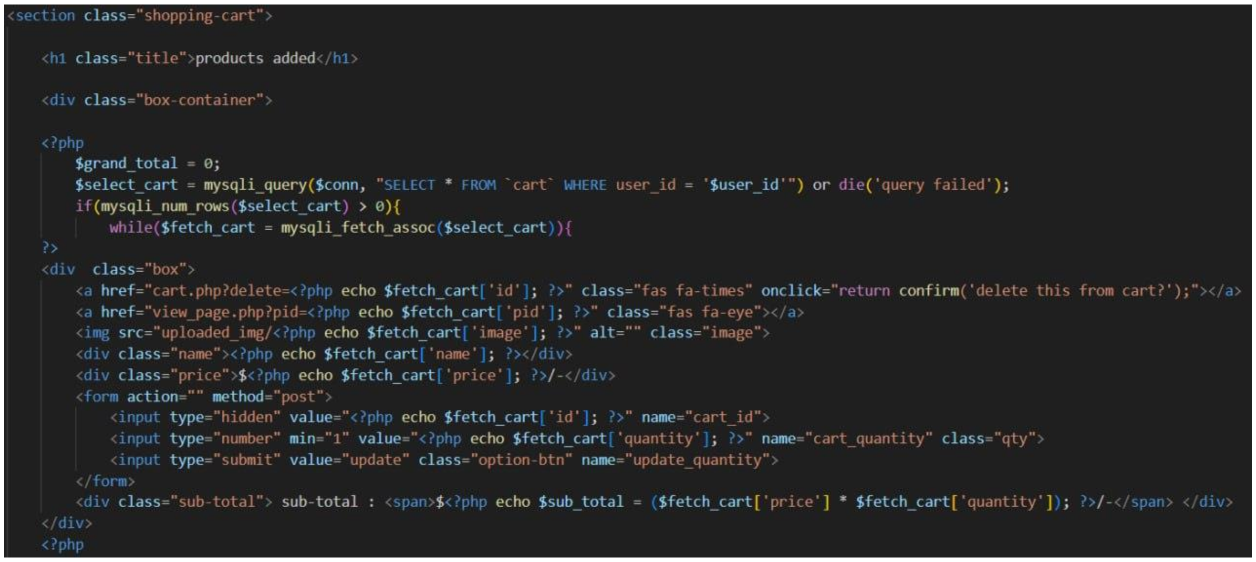
The centerpiece of this chapter is the PHP-driven section responsible for rendering the user's cart items. The 'box-container' encapsulates each product's details, fostering an organized and aesthetically pleasing display. The dynamic nature of this section ensures that user-specific cart data is retrieved from the database, offering a tailored view of the products added.

Within each 'box,' users encounter essential product details, including an image, product name, unit price, and quantity. Additionally, users have the flexibility to update the quantity of each product or remove it entirely from the cart. The 'sub-total' for each product dynamically calculates based on the product's unit price and quantity, contributing to a transparent display of costs.

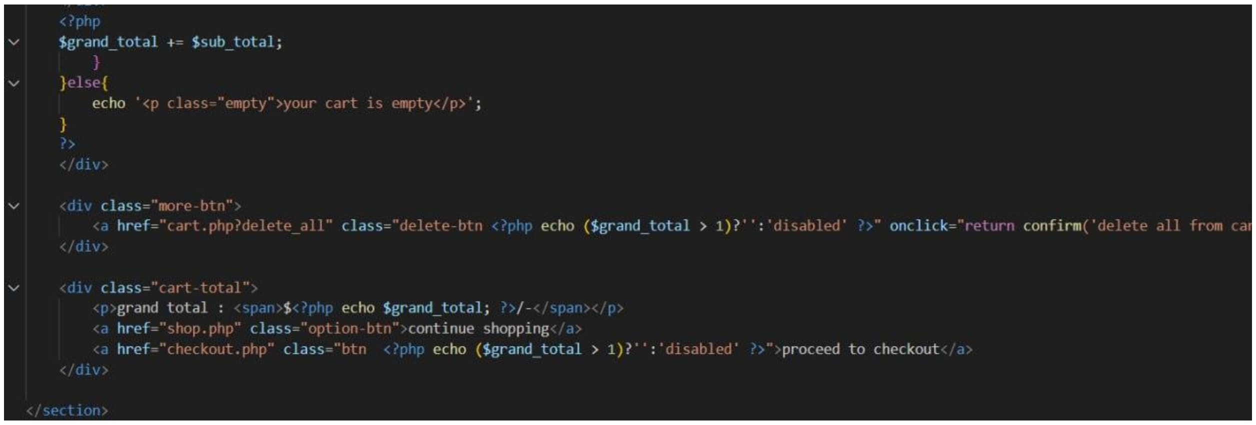
To enhance user control and convenience, the 'delete all' button is featured, enabling users to clear their entire cart. A conditional 'disabled' class ensures that users can only execute this action if their cart contains one or more items, preventing accidental deletion.

The 'cart-total' section offers a summarized view of the user's cart, displaying the grand total cost. Users are prompted to continue shopping or proceed to checkout based on the presence of items in their cart. Notably, the 'proceed to checkout' button is conditionally disabled if the cart is empty, guiding users intuitively through the shopping process.

In conclusion, this chapter illuminates the core functionalities of the 'cart.php' page, providing users with a user-friendly and informative interface to manage their shopping cart effectively. The integration of dynamic PHP scripting ensures a personalized and responsive user experience, underscoring the platform's commitment to a seamless E-commerce journey.



```php
<section class="shopping-cart">

    <h1 class="title">products added</h1>

    <div class="box-container">

    <?php
        $grand_total = 0;
        $select_cart = mysqli_query($conn, "SELECT * FROM `cart` WHERE user_id = '$user_id'") or die('query failed');
        if(mysqli_num_rows($select_cart) > 0){
            while($fetch_cart = mysqli_fetch_assoc($select_cart)){
    ?>
    <div  class="box">
        <a href="cart.php?delete=<?php echo $fetch_cart['id']; ?>" class="fas fa-times" onclick="return confirm('delete this from cart?');"></a>
        <a href="view_page.php?pid=<?php echo $fetch_cart['pid']; ?>" class="fas fa-eye"></a>
        <img src="uploaded_img/<?php echo $fetch_cart['image']; ?>" alt="" class="image">
        <div class="name"><?php echo $fetch_cart['name']; ?></div>
        <div class="price">$<?php echo $fetch_cart['price']; ?>/-</div>
        <form action="" method="post">
            <input type="hidden" value="<?php echo $fetch_cart['id']; ?>" name="cart_id">
            <input type="number" min="1" value="<?php echo $fetch_cart['quantity']; ?>" name="cart_quantity" class="qty">
            <input type="submit" value="update" class="option-btn" name="update_quantity">
        </form>
        <div class="sub-total"> sub-total : <span>$<?php echo $sub_total = ($fetch_cart['price'] * $fetch_cart['quantity']); ?>/-</span> </div>
    </div>
    <?php
```



```php
    <?php
    $grand_total += $sub_total;
        }
    }else{
        echo '<p class="empty">your cart is empty</p>';
    }
    ?>
    </div>

    <div class="more-btn">
        <a href="cart.php?delete_all" class="delete-btn <?php echo ($grand_total > 1)?'':'disabled' ?>" onclick="return confirm('delete all from car
    </div>

    <div class="cart-total">
        <p>grand total : <span>$<?php echo $grand_total; ?>/-</span></p>
        <a href="shop.php" class="option-btn">continue shopping</a>
        <a href="checkout.php" class="btn  <?php echo ($grand_total > 1)?'':'disabled' ?>">proceed to checkout</a>
    </div>

</section>
```

5.8 Wishlist

Here is insightful exploration of the core functionalities embedded in the 'wishlist.php' page for users on our E-commerce platform. The HTML structure is encapsulated within a 'wishlist' section, fostering a visually appealing and user-friendly interface tailored for managing products in the user's wishlist.

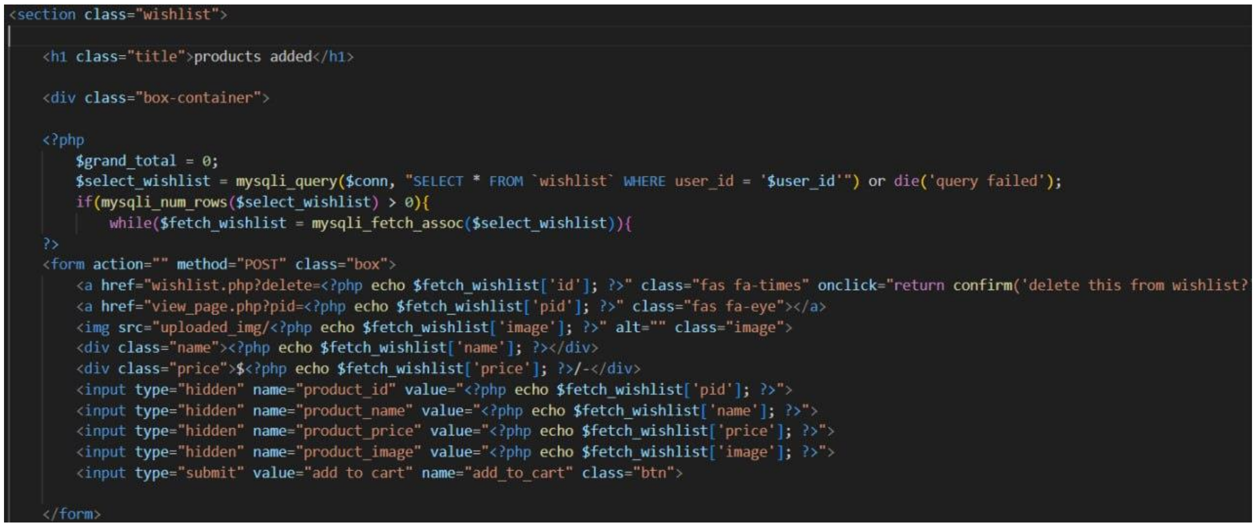
The focal point of this chapter is the dynamic PHP-driven section responsible for rendering the user's wishlist items. Within the 'box-container,' each product is elegantly presented, featuring essential details such as an image, product name, and price. Users can initiate actions such as deleting individual items or adding them directly to their shopping cart.

The 'box' format ensures a structured and organized display, creating an intuitive user experience. For each product, users can effortlessly navigate to the corresponding product page or promptly remove the item from their wishlist. The 'add to cart' button facilitates a seamless transition, allowing users to effortlessly move desired items from the wishlist to their shopping cart.

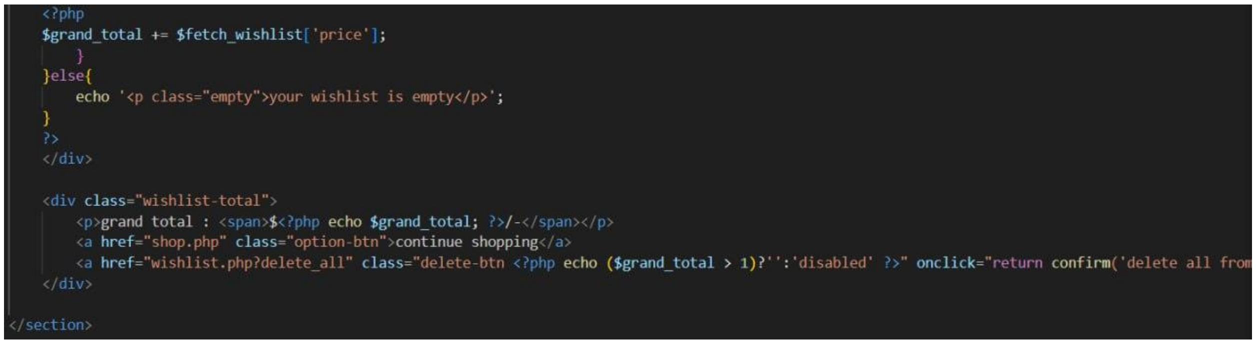
To enhance user control and convenience, the 'delete all' button is featured, enabling users to clear their entire wishlist. The conditional application of a 'disabled' class ensures that users can only execute this action if their wishlist contains one or more items, avoiding accidental deletion.

The 'wishlist-total' section provides users with a summarized view of the total cost of items in their wishlist. Users are presented with the option to continue shopping or efficiently remove all items from their wishlist based on individual preferences. Notably, the 'delete all' button is conditionally disabled if the wishlist is empty, streamlining the user experience.

In conclusion, this chapter sheds light on the primary functionalities of the 'wishlist.php' page, showcasing the platform's commitment to providing users with a seamless and personalized E-commerce journey. The integration of dynamic PHP scripting ensures real-time updates and user interactions, reinforcing the user-centric design of the platform.



```
<section class="wishlist">

    <h1 class="title">products added</h1>

    <div class="box-container">

    <?php
        $grand_total = 0;
        $select_wishlist = mysqli_query($conn, "SELECT * FROM `wishlist` WHERE user_id = '$user_id'") or die('query failed');
        if(mysqli_num_rows($select_wishlist) > 0){
            while($fetch_wishlist = mysqli_fetch_assoc($select_wishlist)){
    ?>
    <form action="" method="POST" class="box">
        <a href="wishlist.php?delete=<?php echo $fetch_wishlist['id']; ?>" class="fas fa-times" onclick="return confirm('delete this from wishlist?
        <a href="view_page.php?pid=<?php echo $fetch_wishlist['pid']; ?>" class="fas fa-eye"></a>
        <img src="uploaded_img/<?php echo $fetch_wishlist['image']; ?>" alt="" class="image">
        <div class="name"><?php echo $fetch_wishlist['name']; ?></div>
        <div class="price">$<?php echo $fetch_wishlist['price']; ?>/-</div>
        <input type="hidden" name="product_id" value="<?php echo $fetch_wishlist['pid']; ?>">
        <input type="hidden" name="product_name" value="<?php echo $fetch_wishlist['name']; ?>">
        <input type="hidden" name="product_price" value="<?php echo $fetch_wishlist['price']; ?>">
        <input type="hidden" name="product_image" value="<?php echo $fetch_wishlist['image']; ?>">
        <input type="submit" value="add to cart" name="add_to_cart" class="btn">

    </form>
```



```
    <?php
    $grand_total += $fetch_wishlist['price'];
        }
    }else{
        echo '<p class="empty">your wishlist is empty</p>';
    }
    ?>
    </div>

    <div class="wishlist-total">
        <p>grand total : <span>$<?php echo $grand_total; ?>/-</span></p>
        <a href="shop.php" class="option-btn">continue shopping</a>
        <a href="wishlist.php?delete_all" class="delete-btn <?php echo ($grand_total > 1)?'':'disabled' ?>" onclick="return confirm('delete all from
    </div>

</section>
```

6. Folders Structures

For our Web Programming class project, we've structured the project into dedicated folders, each with a specific function:

- css:

- This folder contains cascading style sheets (CSS) for styling the web application. It encompasses all styles related to the user interface, ensuring a consistent and visually appealing design throughout the project.

- js:

- Within the js (JavaScript) folder, you'll find scripts that enhance the interactivity and functionality of the web application. These scripts handle client-side operations, providing a dynamic and responsive user experience.

- uploaded_images:

- The uploaded_img folder serves as a storage location for images uploaded by users. It ensures an organized approach to managing and retrieving user-uploaded images within the application.

- PHP:

- Unlike other folders, we've kept these PHP files directly within the project root for convenient access. These files contain server-side logic, support backend functionalities, and contribute to the seamless operation of the web application.

- database:

- The database folder is dedicated to managing database-related operations. It includes PHP files responsible for connecting to the database, executing queries, and handling data transactions. This section serves as the backend for storing and retrieving data.

This well-structured folder arrangement promotes clarity, separation of concerns, and scalability in our project. Each folder serves a specific purpose, contributing to a modular and efficient architecture, making development and maintenance more straightforward.