Project Web Programming

THE BOOKNOOK

Presented by:

Ahmed Mohamed Sadek	***2305355***
Mariam Ahmed Elsayed	***2305300***
Shahd Akram Sherif	***2305398***
Zenab osama	***2305289***
Tasneem Yosry Khamis	***2305262***

THE EXPLANATION OF OUR PROJECT

This project serves as a foundational e-commerce system, allowing users to create accounts, browse and purchase products, and manage their profiles. It incorporates both front-end (HTML, CSS, JavaScript) and back-end (PHP, database) technologies to offer a complete, functional platform for managing user accounts, products, and purchases. The project also serves as an educational tool, demonstrating how to handle common e-commerce tasks such as user authentication, shopping cart management, and order processing

about.php

1.Header Inclusion:

The code starts by including header.php which likely contains the website's header section including navigation elements like logo, home link etc.

2.About Us Section:

It defines a heading "About Us" with a breadcrumb navigation link to the home page.
Then it includes a section with a flexbox layout containing an image on the left and content on the right describing the bookstore.
It highlights why users should choose this bookstore including focus on high-quality scientific books, affordability and catering to students, professionals and enthusiasts.
Finally, it includes a call to action button linking to the contact page.

about.php

3.Client Reviews Section:

It displays a heading "Client's Reviews".
Then it includes a box-container showcasing multiple client reviews with their image, text content and star ratings.

4.Great Authors Section:

Similar to Reviews section, it displays a heading "Great Authors" followed by a box-container showcasing images of different authors along with their social media links.

5.Footer Inclusion:

Similar to header, it includes footer.php which likely contains website footer elements like contact information, copyright etc.

6.Script Inclusion:

Finally, it includes a script file script.js which might contain JavaScript code for any dynamic functionalities on the webpage.

about.php

```
ection class="reviews">
  <h1 class="title">Client's Reviews</h1>
  <div class="box-container">
      <div class="box">
          <img src="people images/pic-1.png" alt="">
          <p>I've found so many great books on BookNook! The selection is diverse, and the site is easy to navigate. Definitely my new
          <div class="stars">
              <i class="fas fa-star"></i>
              <i class="fas fa-star"></i>
              <i class="fas fa-star"></i>
              <i class="fas fa-star"></i>
              <i class="fas fa-star-half"></i>
          </div>
          <h3>Abdelrahman Osama</h3>
      </div>
      <div class="box">
          <img src="people images/pic-2.png" alt="">
          <p>I love how easy it is to find my next read on BookNook. Whether I'm in the mood for a thrilling mystery or a heartwarming
          <div class="stars">
              <i class="fas fa-star"></i>
              <i class="fas fa-star"></i>
              <i class="fas fa-star"></i>
              <i class="fas fa-star"></i>
              <i class="fas fa-star"></i>
          </div>
          <h3>Ahmed Mohamed</h3>
      </div>
      <div class="box">
          <img src="people images/pic-3.png" alt="">
          <p>I love how quickly I can find my next read on BookNook. Their variety of genres keeps me coming back. Great prices and fas
          <div class="stars">
```

```
tion class="authors">
<h1 class="title">Great Authors</h1>
<div class="box-container">
    <div class="box">
        <img src="author images/author-1.png" alt="">
        <div class="share">
        <a href="https://www.facebook.com" target="_blank"> <i class="fab fa-facebook-f"></i>  </a>
    <a href="https://www.twitter.com" target="_blank"> <i class="fab fa-twitter"></i>  </a>
    <a href="https://www.instagram.com" target="_blank"> <i class="fab fa-instagram"></i>  </a>
    <a href="https://www.linkedin.com" target="_blank"> <i class="fab fa-linkedin"></i>  </a>
        </div>
        <h3>Richard Dawkins</h3>
    </div>
    <div class="box">
        <img src="author images/author-2.png" alt="">
        <div class="share">
        <a href="https://www.facebook.com" target="_blank"> <i class="fab fa-facebook-f"></i>  </a>
    <a href="https://www.twitter.com" target="_blank"> <i class="fab fa-twitter"></i>  </a>
    <a href="https://www.instagram.com" target="_blank"> <i class="fab fa-instagram"></i>  </a>
    <a href="https://www.linkedin.com" target="_blank"> <i class="fab fa-linkedin"></i> </a>
        </div>
        <h3>Ada Lovelace</h3>
    </div>
    <div class="box">
        <img src="author images/author-3.png" alt="">
        <div class="share">
        <a href="https://www.facebook.com" target="_blank"> <i class="fab fa-facebook-f"></i>  </a>
    <a href="https://www.twitter.com" target="_blank"> <i class="fab fa-twitter"></i>  </a>
    <a href="https://www.instagram.com" target="_blank"> <i class="fab fa-instagram"></i>  </a>
    <a href="https://www.linkedin.com" target="_blank"> <i class="fab fa-linkedin"></i> </a>
```

admin_contact.php

1.Security Check:

It starts by including config.php which likely contains database connection credentials.
It then initiates a PHP session and checks if an admin_id session variable is set. If not, it redirects the user to the login page. This ensures only authenticated admins can access this page.

2.Delete Message Functionality:

The code checks if a GET parameter named delete is set in the URL.
If yes, it extracts the message ID from the parameter and deletes the corresponding message from the message table in the database using a MySQL query.
After deletion, it redirects the admin back to the admin_contacts.php page.

admin_contact.php

3.Displaying Contact Messages:

It fetches all messages from the message table using a MySQL query.
If there are any messages found:
It loops through each message fetched from the database.
For each message, it displays details like name, phone number, email and message content within a box.
It also displays a "Delete Message" button next to each message. Clicking this button triggers the deletion functionality explained earlier (with confirmation).
If there are no messages found, it displays a message indicating that there are no messages.

4.Admin Header and Script Inclusion:
It includes admin_header.php which likely contains the header section for the admin panel.

Finally, it includes admin_script.js which might contain JavaScript code for any interactive elements on the webpage like the confirmation dialog before deleting a message.

admin_contact.php

```php
<?php
include 'config.php';
session_start();
$admin_id = $_SESSION['admin_id'];
if(!isset($admin_id)){
    header('location:login.php');
};
if(isset($_GET['delete'])){
    $delete_id = $_GET['delete'];
    mysqli_query($conn, "DELETE FROM `message` WHERE id = '$delete_id'") or die('query failed');
    header('location:admin_contacts.php');
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Messages</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css" integrity="
    <link rel="stylesheet" href="admin_style.css?v=time()">
</head>
<body>
    <?php include 'admin_header.php';?>
    <section class="messages">
        <h1 class="title">Messages</h1>
        <div class="box-container">
            <?php
                $select_message = mysqli_query($conn, "SELECT * FROM `message`") or die('query failed');
                if(mysqli_num_rows($select_message) > 0){
                    while($fetch_message = mysqli_fetch_assoc($select_message)){
```

admin_header.php

1.Displays Messages:

It checks if an array of messages ($message) exists.
If it does, it loops through each message and
displays it within a clickable box.
Clicking the "X" icon (provided by Font Awesome)
removes the entire message box.

2.Creates an Admin Header:

It builds the header for the admin panel.
This includes:
A logo linking to the main admin page.
A navigation bar with links to other admin sections
(e.g., Products, Orders, Users).
A user section displaying the admin's username,

```
        </div>
        ';
    }
}
?>

<header class="header">
    <div class="flex">
        <a href="admin_page.php" class="logo">Admin<span>Panel</span></a>
        <nav class="navbar">
            <a href="admin_page.php">Home</a>
            <a href="admin_products.php">Products</a>
            <a href="admin_orders.php">Orders</a>
            <a href="admin_users.php">Users</a>
            <a href="admin_contacts.php">Messages</a>
        </nav>
        <div class="icons">
            <div id="menu-btn" class="fas fa-bars"></div>
            <div id="user-btn" class="fas fa-user"></div>
        </div>
        <div class="account-box">
            <p>Username: <spam><?php echo $_SESSION['admin_name']; ?></spam></p>
            <p>Email: <spam><?php echo $_SESSION['admin_email']; ?></spam></p>
            <a href="logout.php" class="delete-btn">Log Out</a>
        </div>
    </div>
</header>
```

admin_orders.php

1.Authentication:

Security: The code begins by verifying the administrator's login status. It checks if the administrator is logged in by examining a session variable ($admin_id).

2.Order Updates:

Payment Status Modification: The code allows administrators to update the payment status of orders.

3.Order Deletion:

Removal of Orders: The code enables administrators to delete orders from the system.

4.Order Display:

Data Retrieval and Presentation: The code retrieves a list of all orders from the database. It then iterates through each order and displays key information such as:

Order ID
User ID
Order date
Customer details (name, address, contact information)

admin_orders.php

5.User Interface:

The code presents this information in a user-friendly manner, often within a tabular format or a series of cards.dministrative functionalities.

```
onfig.php';
art();
= $_SESSION['admin_id'];
$admin_id)){
('location:login.php');

_POST['update_order'])){
_update_id = $_POST['order_id'];
e_payment = $_POST['update_payment'];
_query($conn, "UPDATE `orders` SET payment_status = '$update_payment' WHERE id='$order_update_id'") or die('qu
ge[] = 'payment status has been updated!';


_GET['delete'])){
e_id = $_GET['delete'];
_query($conn, "DELETE FROM `orders` WHERE id = '$delete_id'") or die('query failed');
('location:admin_orders.php');
```

```
lass="box">
>User id : <span><?php echo $fetch_orders['user_id']; ?></span> </p>
>Placed On : <span><?php echo $fetch_orders['placed_on']; ?></span> </p>
>Name : <span><?php echo $fetch_orders['name']; ?></span> </p>
>Number : <span><?php echo $fetch_orders['number']; ?></span> </p>
>Email : <span><?php echo $fetch_orders['email']; ?></span> </p>
>Address : <span><?php echo $fetch_orders['address']; ?></span> </p>
>Total Products : <span><?php echo $fetch_orders['total_products']; ?></span> </p>
>Total Price : <span>$<?php echo $fetch_orders['total_price']; ?>/-</span> </p>
>Payment Method : <span><?php echo $fetch_orders['method']; ?></span> </p>
orm action="" method="post">
  <input type="hidden" name="order_id" value="<?php echo $fetch_orders['id']; ?>">
  <select name="update_payment">
      <option value="" selected disabled><?php echo $fetch_orders['payment_status']; ?></option>
      <option value="pending">pending</option>
      <option value="completed">completed</option>
  </select>
  <input type="submit" value="update" name="update_order" class="option-btn">
  <a href="admin_orders.php?delete=<?php echo $fetch_orders['id']; ?>" onclick="return confirm('delete this order?');" class="del
form>


  }

se{
  echo '<p class="empty">No Orders placed yet!</p>';
```

admin_page.php

1.Security

2.Data Retrieval and Display:

Queries the database to retrieve key metrics about the e-commerce store's performance. These metrics include:

- Total pending order value
- Total value of completed orders
- Total number of orders placed
- Total number of products listed
- Number of registered users
- Number of administrators
- Total number of users (both regular and admin)
- Number of new messages (likely customer inquiries or support tickets)

```php
<?php include 'admin_header.php'; ?>
<section class="dashboard">
    <h1 class="title">Dasboard</h1>
    <div class="box-container">
        <div class="box">
            <?php
                $total_pendings = 0;
                $select_pending = mysqli_query($conn, "SELECT total_price FROM `orders` WHERE payment_status = 'pending'") or die('query failed');
                if(mysqli_num_rows($select_pending)>0){
                    while($fetch_pendings = mysqli_fetch_assoc($select_pending)){
                        $total_price = $fetch_pendings['total_price'];
                        $total_pendings += $total_price;
                    };
                };
            ?>
            <h3><?php echo '$'. $total_pendings .'/-'; ?></h3>
            <p>Total Pendings</p>
        </div>
        <div class="box">
            <?php
                $total_completed = 0;
                $select_completed = mysqli_query($conn, "SELECT total_price FROM `orders` WHERE payment_status = 'completed'") or die('query failed
                if(mysqli_num_rows($select_completed)>0){
                    while($fetch_completed = mysqli_fetch_assoc($select_completed)){
                        $total_price = $fetch_completed['total_price'];
                        $total_completed += $total_price;
                    };
                };
            ?>
            <h3><?php echo '$'. $total_completed . '/-'; ?></h3>
            <p>Completed Payments</p>
```

admin_product.php

1.Maintain accurate product information:
Ensure product names, prices, and descriptions are up-to-date.

2.Enrich product offerings:
Add new products to the store's inventory.
Remove outdated or discontinued products: Keep the product catalog clean and relevant.

These capabilities are essential for effective e-commerce operations, contributing to a positive customer experience, increased sales, and overall business success.
1.Security measures

2.User experience:
Allows administrators to: add - modify- remove products
•Deletes the corresponding product record from the database.

3.Database Interaction:
The code interacts with a MySQL
File Handling: The code handles file uploads
User Interface: The code provides a basic user interface for administrators to interact with,

admin_product.php

```php
<?php
include 'config.php';
session_start();
$admin_id = $_SESSION['admin_id'];
if(!isset($admin_id)){
    header('location:login.php');
};
if(isset($_POST['add_product'])){
    $name = mysqli_real_escape_string($conn, $_POST['name']);
    $price = $_POST['price'];
    $image = $_FILES['image']['name'];
    $image_size = $_FILES['image']['size'];
    $image_tmp_name = $_FILES['image']['tmp_name'];
    $image_folder = 'uploaded_img/'.$image;
    $select_product_name = mysqli_query($conn, "SELECT name FROM `products` WHERE name = '$name'") or die('query failed');
    if(mysqli_num_rows($select_product_name)>0){
        $message[] = 'product name already added';
    }
    else{
        $add_product_query = mysqli_query($conn, "INSERT INTO `products`(name, price, image) VALUES('$name', '$price', '$image')") or die('query f
        if($add_product_query){
            if($image_size > 2000000){
                $message[] = 'image size is too large';
            }
            else{
                move_uploaded_file($image_tmp_name, $image_folder);
                $message[] = 'product added successfully';
            }
        }
        else{
            $message[] = 'product could not be added!';
        }
    }
}
```

```php
if(isset($_POST['update_product'])){
    $update_p_id = $_POST['update_p_id'];
    $update_name = $_POST['update_name'];
    $update_price = $_POST['update_price'];
    mysqli_query($conn, "UPDATE `products` SET name = '$update_name', price = '$update_price' WHERE id = '$update_p_id'") or die('query failed');
    $update_image = $_FILES['update_image']['name'];
    $update_image_tmp_name = $_FILES['update_image']['tmp_name'];
    $update_image_size = $_FILES['update_image']['size'];
    $update_folder = 'PHP Projects/'.$update_image;
    $update_old_image = $_POST['update_old_image'];
    if(!empty($update_image)){
        if($update_image_size > 2000000){
            $message[] = 'image file size is too large';
        }
        else{
            mysqli_query($conn, "UP @var mixed $update_old_image date_image' WHERE id = '$update_p_id'") or die('query failed');
            move_uploaded_file($upd ler);
            unlink('PHP Projects/'.$update_old_image);
        }
    }
    header('location:admin_products.php');
```

admin_script.js

1.Navbar and Account Box Toggling:

•Event Listeners: The code establishes event listeners for the "menu-btn") and the "user-btn" .

•Toggling Behavior: When the "menu-btn" is clicked, the navigation bar is toggled. This ensures that only one interactive element is active at a time, providing intuitive user interface.

2.Responsive Navigation:

•Scroll Event Handling: An event listener is attached to the scroll event of the window object.

•Auto-Closing: When the user scrolls the page, both the navbar and account box are automatically closed.

Form Handling:

•Canceling Product Edits: An event listener is attached to the "close-update" button .

•Form Closure: When the "close-update" button is clicked, the edit product form is hidden from the user's view, and the browser is redirected back to the main page

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

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

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

admin_style.css

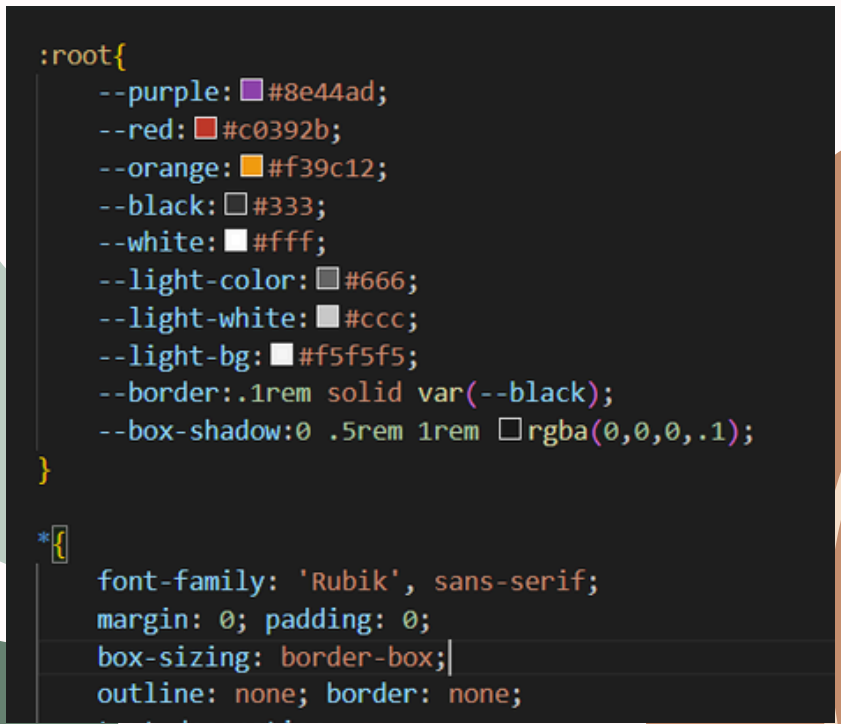
•Consistent Styling:
Utilizes CSS variables for colors, ensuring consistency throughout the interface.
Applies a consistent font family across the page

•Responsive Design:
Employs media queries to adjust the layout and font sizes for different screen sizes, ensuring optimal viewing on various devices .

•User-Friendly Interface:
Styles elements like buttons, input fields, and boxes to enhance user interaction and readability.
Provides clear visual cues

•Accessibility Considerations:
Uses sufficient color contrast for readability.

•Maintainability:
Easy to maintain and update



```
:root{
    --purple:#8e44ad;
    --red:#c0392b;
    --orange:#f39c12;
    --black:#333;
    --white:#fff;
    --light-color:#666;
    --light-white:#ccc;
    --light-bg:#f5f5f5;
    --border:.1rem solid var(--black);
    --box-shadow:0 .5rem 1rem rgba(0,0,0,.1);
}

*{
    font-family: 'Rubik', sans-serif;
    margin: 0; padding: 0;
    box-sizing: border-box;
    outline: none; border: none;
```

admin_users.php

- provide administrators with the ability to:
- View user information.
- Delete user accounts

- Security:

Authentication: The checks for a valid session variable ($admin_id)

- User Management:

User Retrieval: The code fetches a list of all registered users from the database.

User Display: It displays each user's information,

User Deletion: Provides a mechanism for administrators to delete user accounts. the selected user is removed from the database.

```php
include 'config.php';
session_start();
$admin_id = $_SESSION['admin_id'];
if(!isset($admin_id)){
    header('location:login.php');
}
if(isset($_GET['delete'])){
    $delete_id = $_GET['delete'];
    mysqli_query($conn, "DELETE FROM `users` WHERE id = '$delete_id'") or die('query failed');
    header('location:admin_users.php');
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Users</title>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css" integrity="sha512-iecdLmaskl7CVk
    <link rel="stylesheet" href="admin_style.css?v=time()">
</head>
<body>
    <?php include 'admin_header.php';?>
    <section class="users">
        <h1 class="title">User Accounts</h1>
        <div class="box-container">
            <?php
                $select_users = mysqli_query($conn, "SELECT * FROM `users`") or die('query failed');
                while($fetch_users = mysqli_fetch_assoc($select_users)){

            ?>
            <div class="box">
```

cart.php

1.User Authentication:

Ensures that only logged-in users can access their shopping carts by verifying the user's ID through a session variable.

2.Cart Item Management:

- Display: Retrieves and displays a list of products added to
- Quantity Updates: Allows users to adjust the quantity of each item in their cart.
- Item Deletion: Enables users to remove individual items from their cart.
- Cart Clearing: Provides an option to clear the entire cart with a confirmation prompt.
- Order Summary: Calculates and displays the total price of all items in the cart.
- Checkout Functionality: Includes a button to proceed to the checkout page, which is disabled if the cart is empty.

```
> cart.php > ...

config.php';
tart();
= $_SESSION['user_id'];
($user_id)){
r('location:login.php');

$_POST['update_cart'])){
_id = $_POST['cart_id'];
_quantity = $_POST['cart_quantity'];
i_query($conn, "UPDATE `cart` SET quantity = '$cart_quantity' WHERE id = '$cart_id'") or die('query failed');
age[] = 'cart quantity updated!';

$_GET['delete'])){
te_id = $_GET['delete'];
i_query($conn, "DELETE FROM `cart` WHERE id = '$delete_id'") or die('query failed');
r('location:cart.php');

$_GET['delete_all'])){
i_query($conn, "DELETE FROM `cart` WHERE user_id = '$user_id'") or die('query failed');
r('location:cart.php');
```

checkout.php

1.User Authentication:
Ensures that only logged-in users can access their shopping carts by verifying the user's ID through a session variable.

2.Cart Management:

- Display: Retrieves and displays a list of products added to the user's cart, This provides a clear overview of the items in the cart
- Quantity Updates: Allows users to dynamically adjust the quantity of each item within their cart.
- Item Deletion: Enables users to remove individual items from their cart.
- Cart Clearing: Offers the option to clear the entire cart, allowing users to start fresh or make significant changes to their order.

3.Order Summary and Checkout:
Calculates and displays the total price of all items in the cart, providing users with a clear overview order's total cost.
Includes a "Proceed to Checkout" button, this button is dynamically enabled/disabled preventing users from proceeding to checkout with an empty cart.

checkout.php

```php
include 'config.php';
session_start();
$user_id = $_SESSION['user_id'];
if(!isset($user_id)){
    header('location:login.php');
}
if(isset($_POST['order_btn'])){
    $name = mysqli_real_escape_string($conn, $_POST['name']);
    $number = $_POST['number'];
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $method = mysqli_real_escape_string($conn, $_POST['method']);
    $address = mysqli_real_escape_string($conn, 'flat no.'. $_POST['flat'].', '.$_POST['street'].', '.$_POST['city'].', '.$_POST['co
    $placed_on = date('D-M-Y');
    $cart_total = 0;
    $cart_products[] = '';
    $cart_query = mysqli_query($conn, "SELECT * FROM `cart` WHERE user_id = '$user_id'") or die('query failed');
    if(mysqli_num_rows($cart_query) > 0){
        while($cart_item = mysqli_fetch_assoc($cart_query)){
            $cart_products[] = $cart_item['name'].' ('.$cart_item['quantity'].') ';
            $sub_total = ($cart_item['price'] = $cart_item['quantity']);
            $cart_total += $sub_total;
        }
    }
    $total_products = implode(', ', $cart_products);
    $order_query = mysqli_query($conn, "SELECT * FROM `orders` WHERE name = '$name' AND number = '$number' AND email = '$email' AND
    if($cart_total == 0){
        $message[] = 'your cart is empty';
    }
    else{
        if(mysqli_num_rows($order_query) > 0){
            $message[] = 'order already placed!';
        }
        else{
            mysqli_query($conn, "INSERT INTO `orders`(user_id, name, number, email, method, address, total_products, total_price, pl
```

```php
<section class="checkout">
    <form action="" method="post">
        <h3>Place Your Order</h3>
        <div class="flex">
            <div class="inputBox">
                <span>Your Name :</span>
                <input type="text" name="name" required placeholder="Enter Your Name">
            </div>
            <div class="inputBox">
                <span>Your Number :</span>
                <input type="number" name="number" required placeholder="Enter Your Number">
            </div>
            <div class="inputBox">
                <span>Your Email :</span>
                <input type="email" name="email" required placeholder="Enter Your Email">
            </div>
            <div class="inputBox">
                <span>Payment Method :</span>
                <select name="method">
                    <option value="cash on delivery">Cash on Delivery</option>
                    <option value="credit card">Credit Card</option>
                    <option value="instapay">insatpay</option>
                </select>
            </div>
```

checkout.php

•This code connects to a MySQL database named "shop_db" running on the same server as the PHP script. If the connection is successful, the script can proceed to interact with the database . If the connection fails, the script will stop executing.

•Security:
Create a dedicated user with limited privileges for your application and use that user's credentials instead of the root user.

•Error Handling:
The or die() statement provides basic error handling, but it's generally better to use more robust error handling mechanisms, such as try...catch blocks, to gracefully handle potential connection issues.

```php
<?php

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

?>
```

contact.php

1.Database Connection:

The core function is mysqli_connect(), which establishes a connection to a MySQL server running on the local machine ('localhost').

- The provided credentials include the default username ('root') and an empty password
- The script specifies the name of the database to connect to ('shop_db').

2.Error Handling:

- The or die('connection failed'); clause provides basic error handling. If the connection attempt fails, the script execution is terminated, and the message "connection failed" is displayed.

☻ demonstrates the fundamental step of establishing a connection to a database in PHP. A successful database connection is crucial for any web application that needs to interact with data, such as:

- Storing and retrieving user data
- Managing product information
- Processing orders

```php
<?php
include 'config.php';
session_start();
$user_id = $_SESSION['user_id'];
if(!isset($user_id)){
    header('location:login.php');
}
if(isset($_POST['send'])){
    $name = mysqli_real_escape_string($conn, $_POST['name']);
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $number = $_POST['number'];
    $msg = mysqli_real_escape_string($conn, $_POST['message']);
    $select_message = mysqli_query($conn, "SELECT * FROM `message` WHERE name = '$name' AND email = '$email' AND number = '$number' AND message = '$msg'") or d
    if(mysqli_num_rows($select_message) > 0){
        $message[] = 'message sent already!';
    }
    else{
        mysqli_query($conn, "INSERT INTO `message`(user_id, name, email, number, message) VALUES('$user_id', '$name', '$email', '$number', '$msg')") or die('qu
        $message[] = 'message sent successfully!';
    }
}
```

footer.php

class "footer". This acts as a container for the entire footer content.

Individual Boxes:
"Quick Links" Contains links to essential pages
"Extra Links" :Includes links relevant to user accounts and shopping
Contact Info" : Displays contact details
"Follow Us" : Provides links to social media profiles
<p class="credit"> displays the copyright information.
<?php echo date('Y'); ?> dynamically inserts the current year using PHP.
The copyright notice is attributed to "BookNook Team

```
<section class="footer">
    <div class="box-container">
        <div class="box">
            <h3>Quick Links</h3>
            <a href="home.php">Home</a>
            <a href="about.php">About</a>
            <a href="shop.php">Shop</a>
            <a href="contact.php">Contact</a>
            <a href="orders.php">Orders</a>
        </div>
        <div class="box">
            <h3>Exrea Links</h3>
            <a href="login.php">Login</a>
            <a href="register.php">Register</a>
            <a href="cart.php">Cart</a>
            <a href="orders.php">Orders</a>
        </div>
        <div class="box">
            <h3>Contact Info</h3>
            <p><i class="fas fa-phone"></i> 01229646168</p>
            <p><i class="fas fa-phone"></i> 01555997951 </p>
            <p><i class="fas fa-envelope"></i> zenabosama5@gmail.com </p>
            <p><i class="fas fa-map-marker-alt"></i> Alexandria, Egypt</p>
        </div>
        <div class="box">
            <h3>Follow Us</h3>
            <a href="https://www.facebook.com" target="_blank"> <i class="fab fa-facebook-f"></i> Facebook </a>
            <a href="https://www.twitter.com" target="_blank"> <i class="fab fa-twitter"></i> Twitter </a>
```

header.php

1.Displaying Messages:

If there are any messages, they are displayed in boxes that can be closed by clicking an "X" icon.

2. Header Structure:

•Top Section:

*Contains social media icons .

*Provides links for "Login" and "Register".

•Main Section:

*Shows the website logo.

*Includes a navigation bar with links to "Home", "About", "Shop"....

•Displays icons for:

*Opening a menu .

*Searching the website.

*Accessing user options.

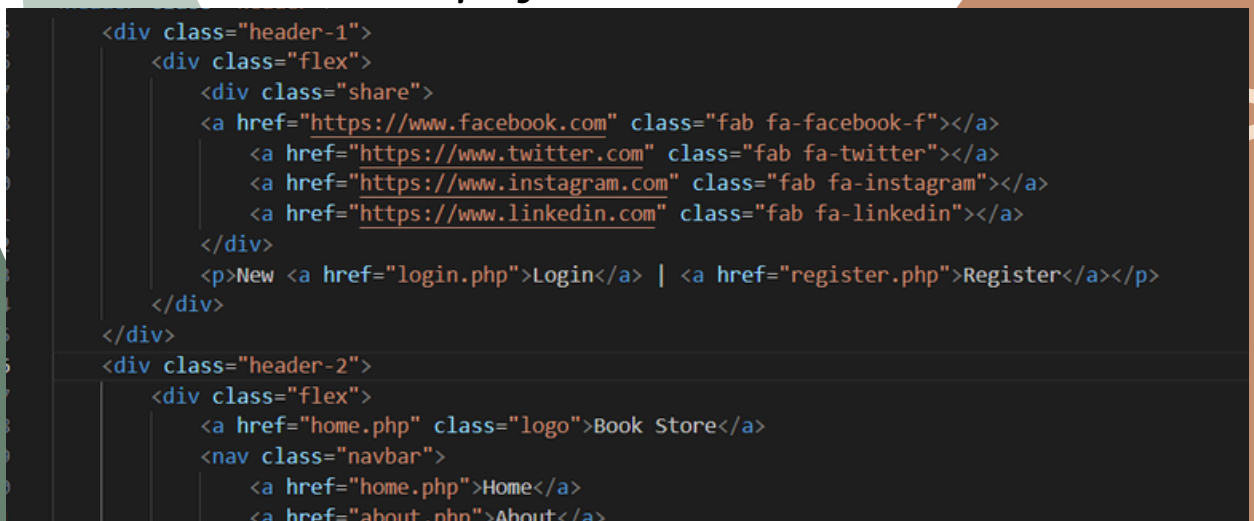
*Viewing the shopping cart with the number of items in it.

*If a user is logged in, it shows their username, email, and a "Log Out" button.

3. Behind the Scenes:

The code interacts with a database to get the number of items in the user's shopping cart.

It uses PHP to display info and handle user actions.



home.php

1.Setting Up:
It includes necessary files like config.php (likely containing database credentials) and the header (header.php) and footer (footer.php) sections.
It starts a PHP session (used to track user login status).
Checks if a user is logged in. If not, it redirects them to the login page.

2.Homepage Content:
Hero Section: Displays a title, description, and a "Discover More" button.
Latest Products:

3.About Us:
Shows an image and a description about the bookstore team

4.Contact Us:
Displays a message inviting users to contact the bookstore with any questions.

home.php

5.User Interaction:
The "add to cart" button it might add the selected product and quantity to the user's shopping cart.

6.Additional Features:
The code links stylesheets (style.css) and a JavaScript file (script.js) for styling and interactivity

```
36      <section class="home">
41          </div>
42      </section>
43      <section class="products">
44          <h1 class="title">Latest Products</h1>
45          <div class="box-container">
46              <?php
47                  $select_products = mysqli_query($conn, "SELECT * FROM `products` LIMIT 6") or die('query failed');
48                  if(mysqli_num_rows($select_products) > 0){
49                      while($fetch_products = mysqli_fetch_assoc($select_products)){
50              ?>
51              <form action="" method="post" class="box">
52                  <img src="<?php echo $fetch_products['image']; ?>" alt="">
53                  <div class="name"><?php echo $fetch_products['name']; ?></div>
54                  <div class="price">$<?php echo $fetch_products['price']; ?>/-</div>
55                  <input type="number" min="1" name="product_quantity" value="1" class="qty">
56                  <input type="hidden" name="product_name" value="<?php echo $fetch_products['name']; ?>">
57                  <input type="hidden" name="product_price" value="<?php echo $fetch_products['price']; ?>">
58                  <input type="hidden" name="product_image" value="<?php echo $fetch_products['image']; ?>">
59                  <input type="submit" value="add to cart" name="add_to_cart" class="btn">
60              </form>
61              <?php
```

login.php

1. User Input:

•The user enters their email and password into the login form.

When the "Login Now" button is clicked, the form data is sent for processing.

2. Data Processing:

The code checks if the form was submitted.

It retrieves the entered email and password.

It sanitizes the email to.

It encrypts the password .

3. Database Check:

The code attempts to find a matching user in the database using the provided email and encrypted password.

4. Login Success:

If a matching user is found:

•It checks the user's type, stores relevant user information then redirects the user to the appropriate page

•If no matching user is found, an error message is displayed

login.php

5. Displaying the Form:
The code displays the login form with input fields for email and password, along with a "Login Now" button. If any error messages exist, they are displayed above the form.

```php
include 'config.php';
session_start();
if(isset($_POST['submit'])){
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $pass = mysqli_real_escape_string($conn, md5($_POST['password']));
    $select_users = mysqli_query($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');
```

logout.php

This code effectively logs a user out by clearing their session data and redirecting them to the login page. This ensures that the user is no longer authenticated and requires them to log in again to access protected areas of the website.

```php
<?php
include 'config.php';
session_start();
session_unset();
session_destroy();
header('location:login.php');
?>
```

orders.php

1.Authentication:

•Checks if the user is logged in by verifying the presence of their ID in the session.

•Redirects to the login page if the user is not logged in.

2.Database Interaction:

•Retrieves the user's order information from the database based on their ID.

3.Order Display:

•If orders are found:

Displays each order's details (date, items, price, status) in a clear format.

•If no orders are found:

Displays a message indicating that the user has no orders.

```
</head>
<body>
    <?php include 'header.php';?>
    <div class="heading-6">
        <h3>Placed Orders</h3>
        <p><a href="home.php">Home</a> / Orders </p>
    </div>
    <section class="placed-orders">
        <h1 class="title">Placed Orders</h1>
        <div class="box-container">
            <?php
                $order_query = mysqli_query($conn, "SELECT * FROM `orders` WHERE user_id = '$user_id'") or die('query failed');
                if(mysqli_num_rows($order_query) > 0){
                    while($fetch_orders = mysqli_fetch_assoc($order_query)){
            ?>
            <div class="box">
                <p>Placed on : <span><?php echo $fetch_orders['placed_on']; ?></span></p>
                <p>Name: <span><?php echo $fetch_orders['name']; ?></span></p>
                <p>Number: <span><?php echo $fetch_orders['number']; ?></span></p>
                <p>Email: <span><?php echo $fetch_orders['email']; ?></span></p>
                <p>Address: <span><?php echo $fetch_orders['address']; ?></span></p>
                <p>Payment Method: <span><?php echo $fetch_orders['method']; ?></span></p>
                <p>Your Orders: <span><?php echo $fetch_orders['total_products']; ?></span></p>
```

register.php

1.User Input and Validation:

•collects user input (name, email, password) from a registration form.

It checks if the provided email is already registered in the database.

It verifies that the entered password and confirmed password match.

2. Data Sanitization and Security:

•mysqli_real_escape_string() prevents SQL injection attacks by escaping special characters in user input.

•It hashes the user's password using md5() before storing it in the database to enhance security.

3. Database Interaction:

•If the validation checks pass, the code inserts a new user record into the users table in the database.

4. Response and Redirection:

•Upon successful registration, the code displays a success message and redirects the user to the login page.

```php
include 'config.php';
if(isset($_POST['submit'])){
    $name = mysqli_real_escape_string($conn, $_POST['name']);
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $pass = mysqli_real_escape_string($conn, md5($_POST['password']));
    $cpass = mysqli_real_escape_string($conn, md5($_POST['cpassword']));
    $user_type = $_POST['user_type'];
    $select_users = mysqli_query($conn, "SELECT * FROM `users` WHERE email = '$email' AND password = '$pass'") or die('query failed');
    if(mysqli_num_rows($select_users) > 0){
        $message[] = 'user already exist!';
    }
    else{
        if($pass != $cpass){
            $message[] = 'confirmed password not matched!';
        }
        else{
            mysqli_query($conn, "INSERT INTO `users`(name, email, password, user_type) VALUES('$name', '$email', '$cpass', '$user_type')") or die('quer
            $message[] = 'registered successfully!';
            header('location:login.php');
        }
```

register.php

JavaScript code manages the interactive elements of a website header, including:

- User Menu: The user menu can be toggled by clicking a specific button. When the user menu is opened, the main navigation menu is automatically closed.
- Main Navigation Menu: Similar to the user menu, the main navigation menu can be toggled with its own button.
- Scroll Behavior: When the user scrolls down the page: Both the user menu and main navigation menu are closed. The header's style may change (e.g., become smaller or change color) if the user scrolls past a certain point.

```
let userBox = document.querySelector('.header .header-2 .user-box');
document.querySelector('#user-btn').onclick = () =>{
    userBox.classList.toggle('active');
    navbar.classList.remove('active');
}
let navbar = document.querySelector('.header .header-2 .navbar');
document.querySelector('#menu-btn').onclick = () =>{
    navbar.classList.toggle('active');
    userBox.classList.remove('active');
}
window.onscroll = () =>{
    userBox.classList.remove('active');
    navbar.classList.remove('active');
    if(window.scrollY > 60){
        document.querySelector('.header .header-2').classList.add('active');
    }
    else{
```

search_page.php

1•User Authentication: Ensures that only logged-in users can access the search functionality.

2•Product Search: Allows users to search for products based on their names.

3•Product Display: Displays search results with product images, names, prices, and an option to add products to the cart.

4•Add to Cart: Enables users to add selected products to their shopping cart.

5•Error Handling: Provides appropriate messages for cases like no search results found or when a product is already in the cart.

```
<body>
    <?php include 'header.php';?>
    <div class="heading-7">
        <h3>Search Page</h3>
        <p><a href="home.php">Home</a> / Search </p>
    </div>
    <section class="search-form">
        <form action="" method="post">
            <input type="text" name="search" placeholder="search products..." class="box">
            <input type="submit" name="submit" value="search" class="btn">
        </form>
    </section>
    <section class="products" style="padding-top: 0;">
        <div class="box-container">
            <?php
                if(isset($_POST['submit'])){
                    $search_item = $_POST['search'];
                    $select_products = mysqli_query($conn, "SELECT * FROM `products` WHERE name LIKE '%{$search_item}%'") or die('query failed');
                    if(mysqli_num_rows($select_products) > 0){
                        while($fetch_products = mysqli_fetch_assoc($select_products)){
            ?>
            <form action="" method="post" class="box">
                <img src="<?php echo $fetch_products['image']; ?>" alt="">
                <div class="name"><?php echo $fetch_products['name']; ?></div>
                <div class="price">$<?php echo $fetch_products['price']; ?>/-</div>
                <input type="number" min="1" name="product_quantity" value="1" class="qty">
                <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 cart" name="add_to_cart" class="btn">
            </form>
            <?php
```

shop.php

1.User Authentication

2.Display Products (PHP):

It retrieves all products from the products table in the database.

*If products are found:

•A loop iterates through each product and displays it within a product box , then showcases the product information and a quantity input field.

hidden form is included within the product box to add the selected product to the cart. .If no products are found, a message indicating "No Products Added Yet!" is displayed.

3.Add to Cart Functionality

•When the user clicks the "add to cart" button for a product: the code checks if the product is already in the user's cart • If the product is already in the cart, an error message ("already added to cart") .

•If the product is not already in the cart A success message ("product added to cart!")

```
<?php include 'header.php';?>
<div class="heading-2">
    <h3>Our Shop</h3>
    <p><a href="home.php">Home</a> / Shop</p>
</div>
<section class="products">
    <h1 class="title">Latest Products</h1>
    <div class="box-container">
        <?php
            $select_products = mysqli_query($conn, "SELECT * FROM `products`") 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">
            <img src="<?php echo $fetch_products['image']; ?>" alt="">
            <div class="name"><?php echo $fetch_products['name']; ?></div>
            <div class="price">$<?php echo $fetch_products['price']; ?>/-</div>
            <input type="number" min="1" name="product_quantity" value="1" class="qty">
            <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 cart" name="add_to_cart" class="btn">
```

shop_db.sql

- SQL allows you to define the structure of your database by creating new databases and tables within them.
- You can specify the columns (fields) in each table, their data types (e.g., text, numbers, dates), and constraints (e.g., primary keys, foreign keys, unique values).
- users: Stores user information like name, email, password, and user type (admin or user).
- products: Stores product information like name, price, and image.
- cart: Stores user cart items, including user ID, product name, price, quantity, and image.
- orders: Stores user order details like user ID, name, contact information, order items, total price, placed date, and payment status.
- message: Stores user messages, likely for contacting the website owner.

```sql
CREATE TABLE `cart` (
  `id` int(100) NOT NULL,
  `user_id` int(100) NOT NULL,
  `name` varchar(100) NOT NULL,
  `price` int(100) NOT NULL,
  `quantity` int(100) NOT NULL,
  `image` varchar(100) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

--
-- Dumping data for table `cart`
--

INSERT INTO `cart` (`id`, `user_id`, `name`, `price`, `quantity`, `image`) VALUES
(1, 2, 'A Game of Thrones', 419, 2, 'A song of ice and fire.jpeg'),
(2, 2, 'A Clash of Kings', 476, 3, 'A clash of kings.jpeg');

-- --------------------------------------------------------

--
-- Table structure for table `message`
--

CREATE TABLE `message` (
  `id` int(100) NOT NULL,
  `user_id` int(100) NOT NULL,
  `name` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `number` varchar(12) NOT NULL,
  `message` varchar(500) NOT NULL
```

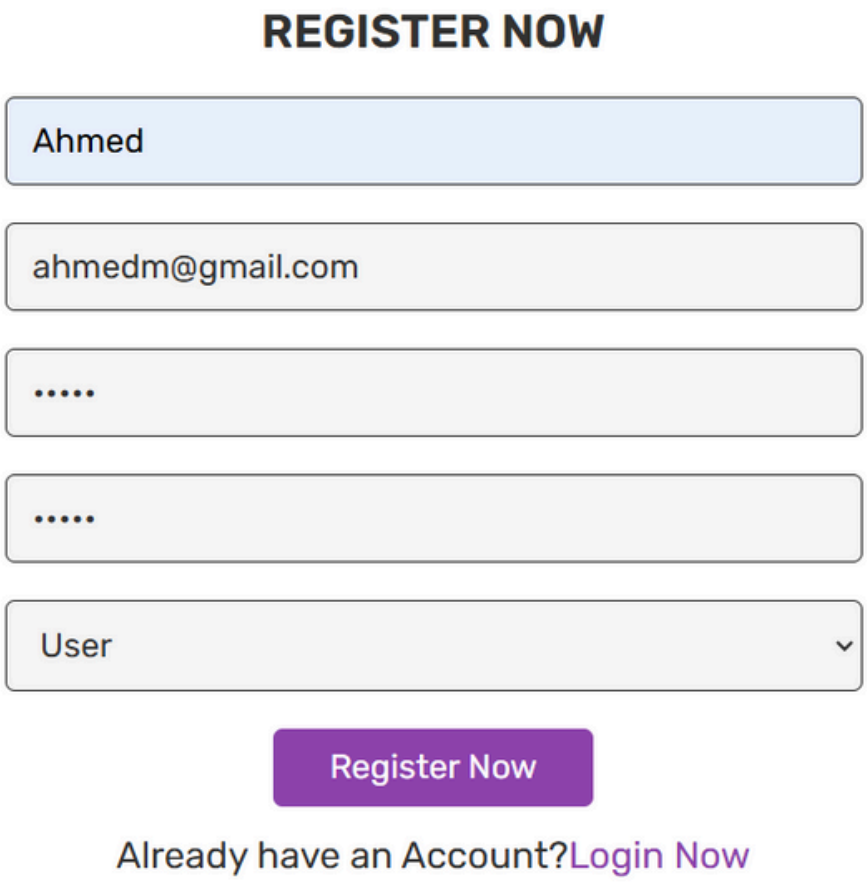
style.css

•CSS (Cascading Style Sheets) is the language used to style the presentation of content written in HTML. It controls the look and feel of a web page, dictating how a site is displayed on a browse

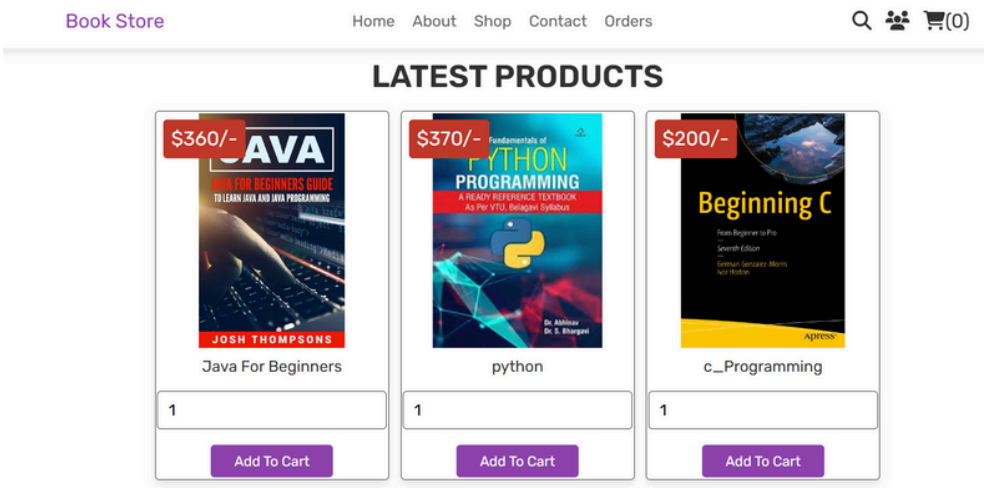
•Visual Appearance : Colors ,Fonts, Spacing, Layout
User Interface (UI) Enhancements : Hover Effects, Animations
Responsiveness Accessibility

•Maintaining Consistency : Reusability ,Separation of Concerns

Test

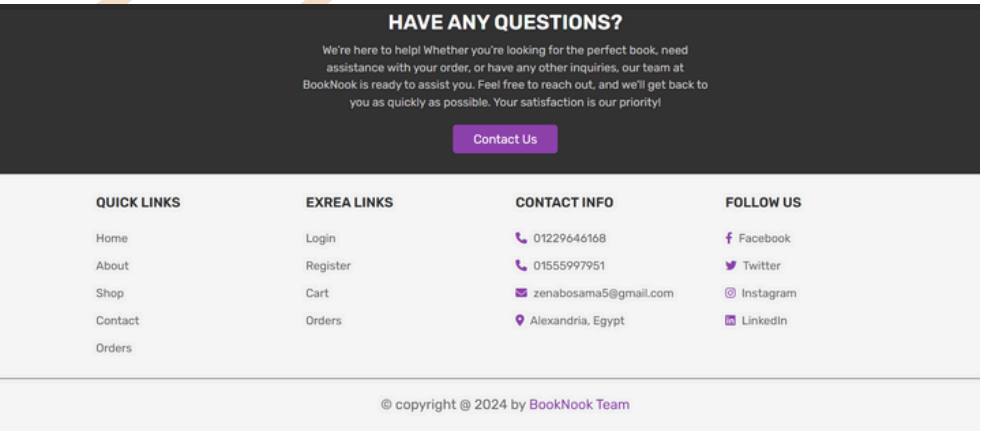


register.php

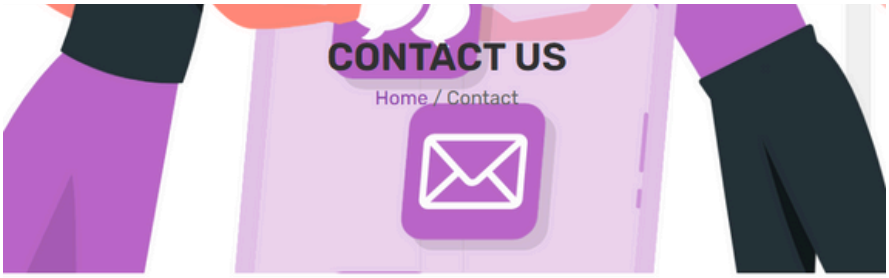


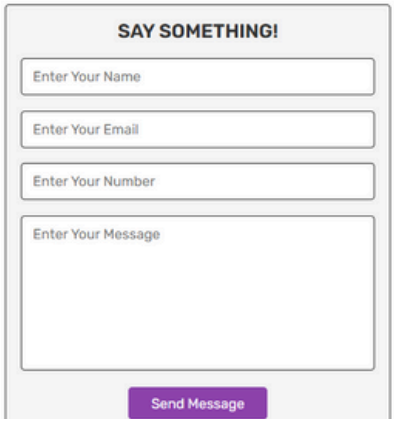
home.php

Test

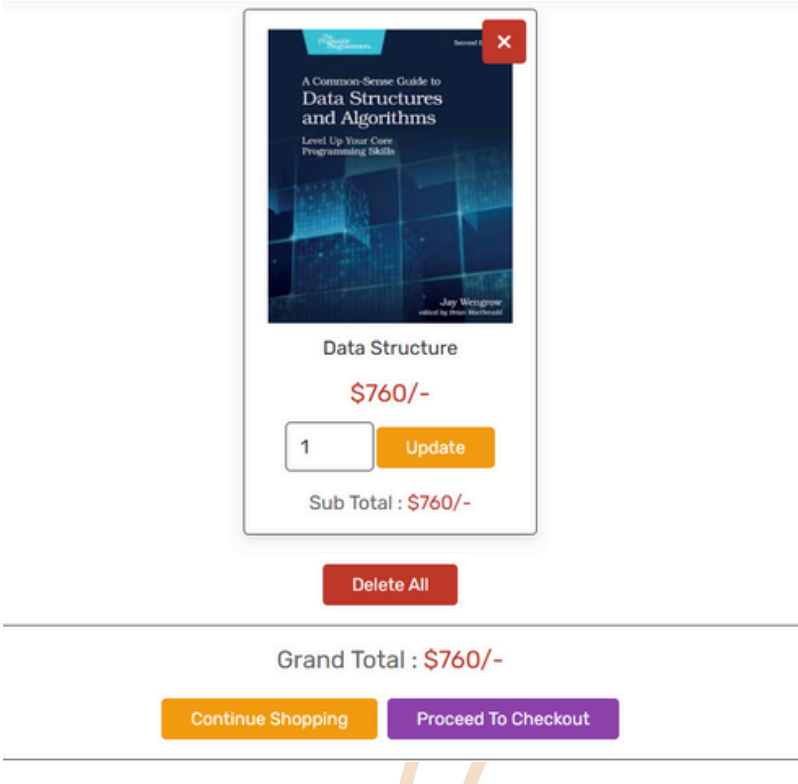


footer.php



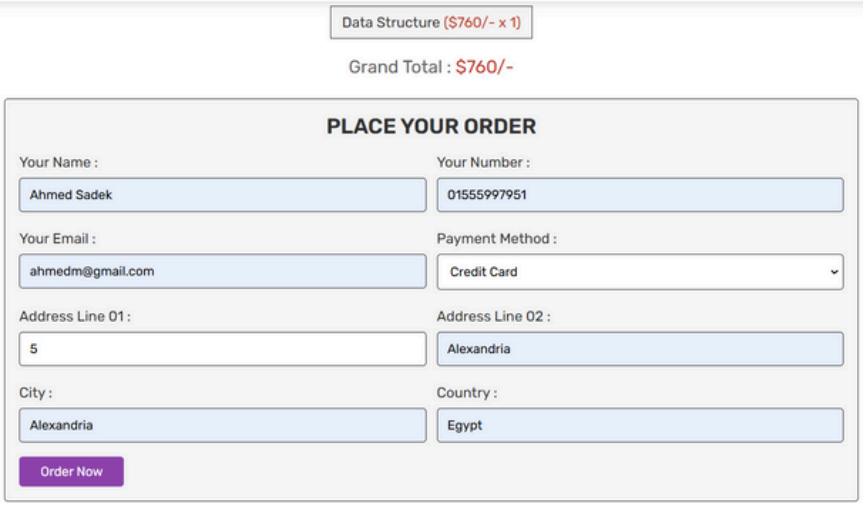


contact.php

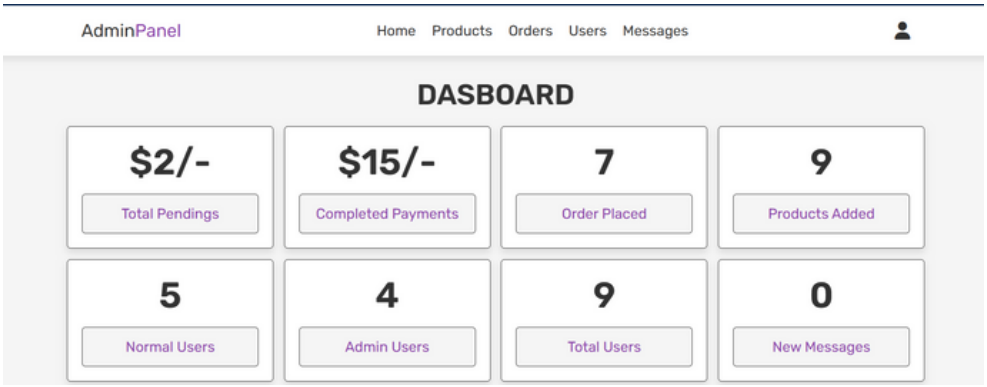


cart.php

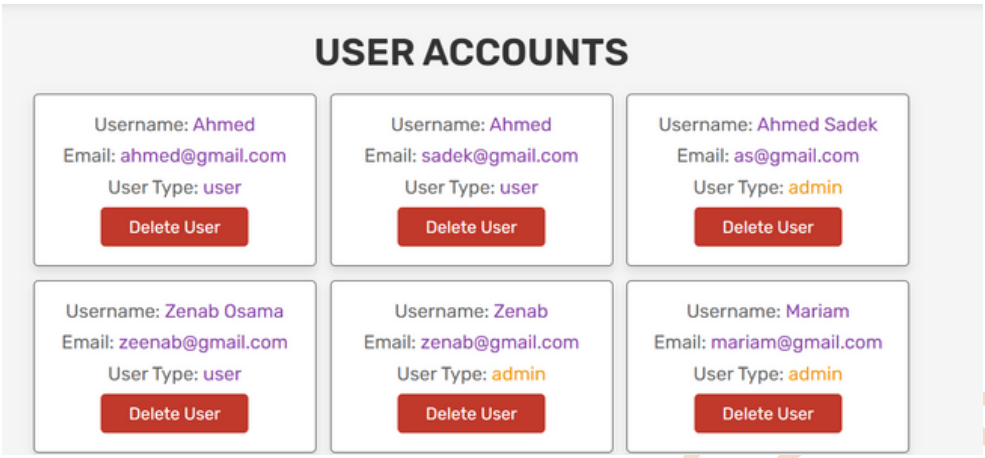
Test



checkout.php

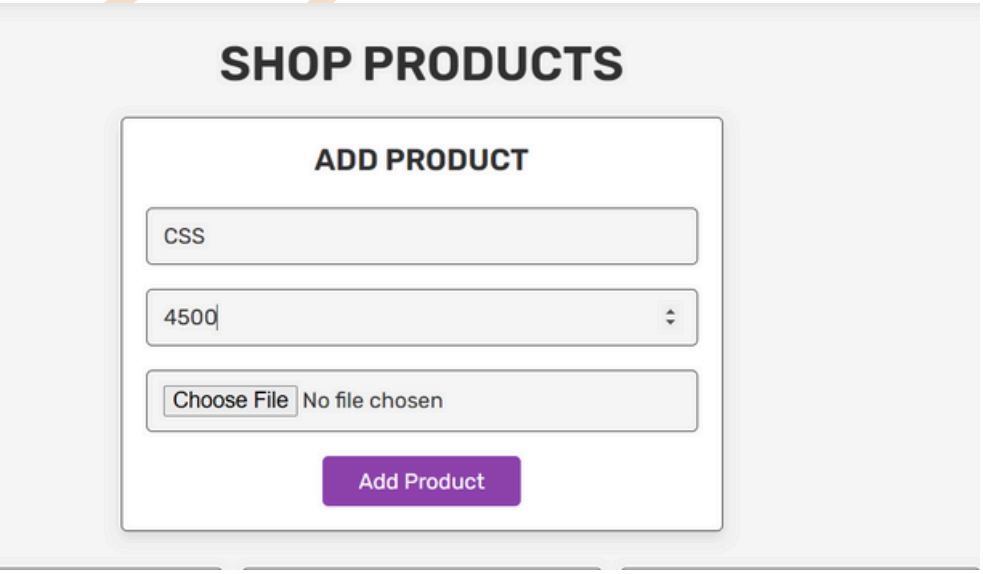


adminpanel.php

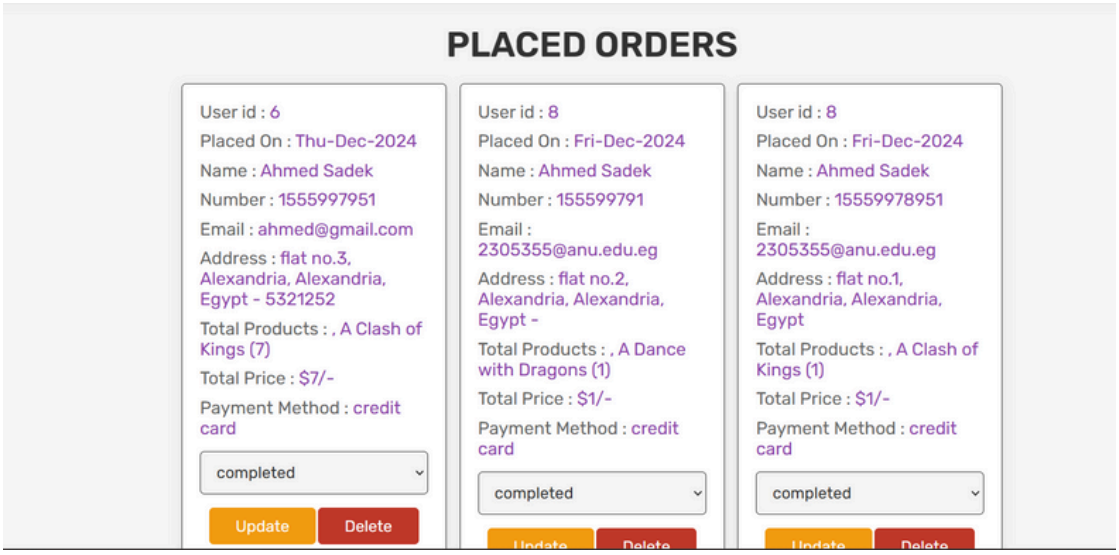


Users.php

Test



products.php



orders.php