PHP Project

Author: Eloi Rochard Eliamar

Date: 9/26/2025



All RED font should be replaced

Code + word version : https://github.com/rochardEloi/php\_project.git

All lines of code in the code screenshots must be readable.

If your code is too big, chunk the code into multiple slides.

# LAMP Stack

## LAMP STACK Information

#### Description:

<Include a 3-5 sentence paragraph explaining the LAMP STACK.>

#### System Specs:

<Fill in in the below information about your specific LAMP STACK.>

* **Operating System:** Distributor ID: Kali,Description: Kali GNU/Linux Rolling Release: 2025.3 Codename: kali-rolling
* **Apache Version: Server version: Apache/2.4.65 (Debian)**
* **Server built: 2025-08-15T08:30:58**
* **Database:** mysql from 11.8.3-MariaDB, client 15.2 for debian-linux-gnu (x86\_64) using EditLine wrapper
* **PHP: PHP 8.4.11 (cli) (built: Aug 15 2025 23:34:18) (NTS)**

# Index.php

## Index.php Information

##### Description:

The index page serves as the main landing page for the PHP Project website, providing a clean and welcoming interface for visitors. It features a responsive navigation menu that adapts to different screen sizes and includes links to all major sections of the site. The page displays an overview of the project's features and includes call-to-action buttons that guide users to either register for an account or log in if they already have one.

##### Key Points about the Page:

* Css
* Responsive

## Index.php Code Screenshot

Line Description here

**### Chunk 1: HTML Structure and Navigation**

```php

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>PHP Project</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <header>

        <nav class="navbar">

            <div class="nav-container">

                <div class="nav-logo">

                    <h1>PHP Project</h1>

                </div>

                <div class="nav-menu">

                    <a href="index.php" class="nav-link active">Home</a>

                    <a href="register.php" class="nav-link">Register</a>

                    <a href="login.php" class="nav-link">Login</a>

                    <?php

                    session\_start();

                    if (isset($\_SESSION['user\_id'])) {

```

**\*\*Explanation:\*\*** This chunk sets up the basic HTML structure with proper meta tags and links to the CSS file. It creates the header navigation with the site logo and menu items. The PHP code starts a session and conditionally shows different navigation options based on whether a user is logged in or not.

**### Chunk 2: Features and About Sections**

```php

    <section class="features">

        <div class="container">

            <h2>Features</h2>

            <div class="features-grid">

                <div class="feature-card">

                    <h3>User Registration</h3>

                    <p>Create an account with username, email, and secure password hashing.</p>

                </div>

                <!-- More feature cards... -->

            </div>

        </div>

    </section>

    <section class="about">

        <div class="container">

            <h2>About This Project</h2>

            <p>This is a demonstration of a complete LAMP stack application featuring:</p>

            <ul>

                <li>Modern PHP development practices</li>

                <li>Secure user authentication system</li>

                <!-- More list items... -->

            </ul>

        </div>

    </section>

</main>

<footer>

    <div class="container">

        <p>&copy; 2024 PHP Project. All rights reserved.</p>

    </div>

</footer>

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

</body>

</html>

```

**\*\*Explanation:\*\*** These sections showcase the application features in a grid layout and provide information about the project. The features section highlights key capabilities while the about section lists the technologies and practices used. The footer provides copyright information and includes the JavaScript file.

Code Screenshot Goes Here

A black screen with white lines

AI-generated content may be incorrect.

A black screen with white lines

AI-generated content may be incorrect.

### PHP Webpage Screenshot

Webpage Description here

A simple presentation page that display all the features of the website

Webpage Screenshot Goes Here

A screenshot of a computer

AI-generated content may be incorrect.

# Database Information

## Database Information

##### Description:

The database\_setup.sql file contains the SQL commands needed to create the MySQL database structure for the project. It defines two main tables: users for storing user account information and user\_sessions for managing user login sessions. The file includes sample data and proper indexing for optimal database performance.

## Database Screenshot

Screenshot of **describe people;** from the database

A screenshot of a computer

AI-generated content may be incorrect.

Screenshot of the database with all the new users added

A screenshot of a computer

AI-generated content may be incorrect.

# phptest.php

## phptest.php Information

##### Description:

The PHP test page demonstrates basic PHP functionality including version information, server details, and database connectivity. It shows simple examples of PHP variables, arrays, and database queries in action. The page serves as a basic diagnostic tool to verify that PHP and the database connection are working properly.

##### Key Points about the Page:

<List any tricky or special points on the phptest.php page>

* Point 1
* Point 2
* Point 3

## phptest.php Code Screenshot

Code Screenshot Goes Here Webpage Screenshot Goes Here

A screenshot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

# Connect.php page

## Connect Page Information

##### Description:

The connect.php file establishes and manages the connection between the PHP application and the MySQL database. It contains the database server configuration including hostname, username, password, and database name. The file also includes error handling to gracefully manage connection failures.

##### Key Points about the Page:

<List any tricky or special points on the php page>

* Point 1
* Point 2
* Point 3

## PHP Code Screenshot

Line Description here

<?php

// Database connection configuration

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "php\_project";

// Create connection

$conn = new mysqli($servername, $username, $password, $dbname);

// Check connection

if ($conn->connect\_error) {

    die("Connection failed: " . $conn->connect\_error);

}

// Set charset to utf8

$conn->set\_charset("utf8");

// Function to safely close the connection

function closeConnection() {

    global $conn;

    if ($conn) {

        $conn->close();

    }

}

?>

Code Screenshot Goes Here

A screen shot of a computer program

AI-generated content may be incorrect.

# Account Creation Web Page

## Account Creation Page Information

##### Description:

The registration page allows new users to create an account by filling out a form with their personal information including first name, last name, username, email, and password. The form includes validation to ensure all required fields are completed and that the password meets minimum security requirements. Once submitted, the system checks for duplicate usernames or emails and provides feedback to the user about the success or failure of their registration attempt.

##### Key Points about the Page:

<List any tricky or special points on the php page>

* Point 1
* Point 2
* Point 3

## PHP Code Screenshot

Line Description here

// Check if username or email already exists

$stmt = $conn->prepare("SELECT id FROM users WHERE username = ? OR email = ?");

$stmt->bind\_param("ss", $username, $email);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows > 0) {

    $errors[] = "Username or email already exists";

}

if (empty($errors)) {

    // Hash password

    $hashed\_password = password\_hash($password, PASSWORD\_DEFAULT);

    // Insert user

    $stmt = $conn->prepare("INSERT INTO users (username, email, password, first\_name, last\_name) VALUES (?, ?, ?, ?, ?)");

    $stmt->bind\_param("sssss", $username, $email, $hashed\_password, $first\_name, $last\_name);

    if ($stmt->execute()) {

        echo '<div class="message success">Registration successful! <a href="login.php">Login here</a></div>';

    } else {

        echo '<div class="message error">Registration failed. Please try again.</div>';

    }

}

```

**\*\*Explanation:\*\*** This code performs database operations during user registration. It first checks for existing users with prepared statements to prevent SQL injection, then securely hashes the password and inserts the new user record. The use of prepared statements ensures security and prevents SQL injection attacks.

**### User Login Database Operations (login.php)**

```php

// Get user from database

$stmt = $conn->prepare("SELECT id, username, password, first\_name, last\_name FROM users WHERE username = ?");

$stmt->bind\_param("s", $username);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows === 1) {

    $user = $result->fetch\_assoc();

    if (password\_verify($password, $user['password'])) {

        // Login successful

        $\_SESSION['user\_id'] = $user['id'];

        $\_SESSION['username'] = $user['username'];

        $\_SESSION['first\_name'] = $user['first\_name'];

        $\_SESSION['last\_name'] = $user['last\_name'];

        // Generate session token

        $session\_token = bin2hex(random\_bytes(32));

        $\_SESSION['session\_token'] = $session\_token;

        // Store session in database

        $stmt = $conn->prepare("INSERT INTO user\_sessions (user\_id, session\_token, expires\_at) VALUES (?, ?, DATE\_ADD(NOW(), INTERVAL 24 HOUR))");

        $stmt->bind\_param("is", $user['id'], $session\_token);

        $stmt->execute();

        echo '<div class="message success">Login successful! Redirecting...</div>';

        header("refresh:2;url=index.php");

        exit();

    } else {

        $errors[] = "Invalid username or password";

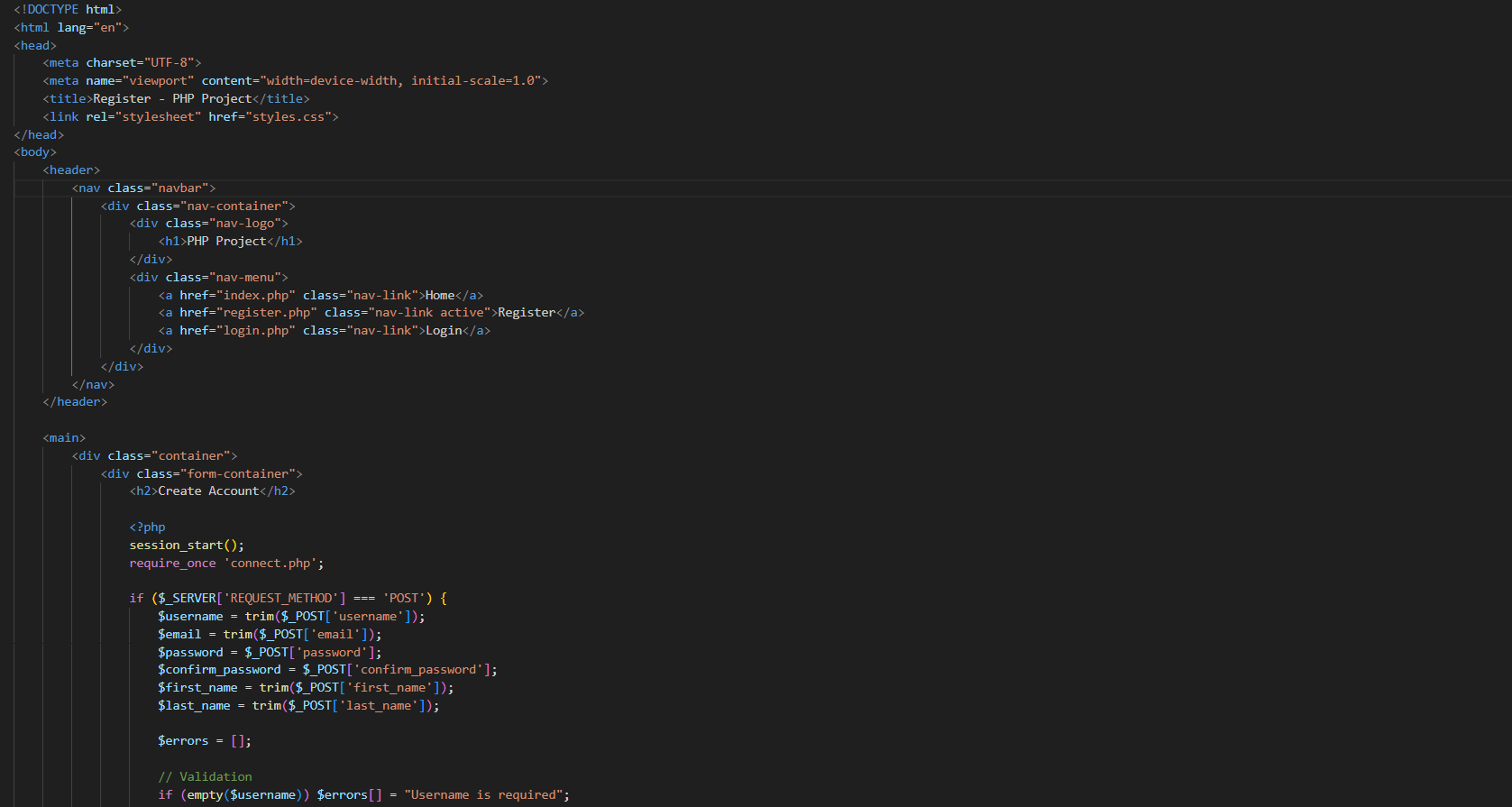
    }

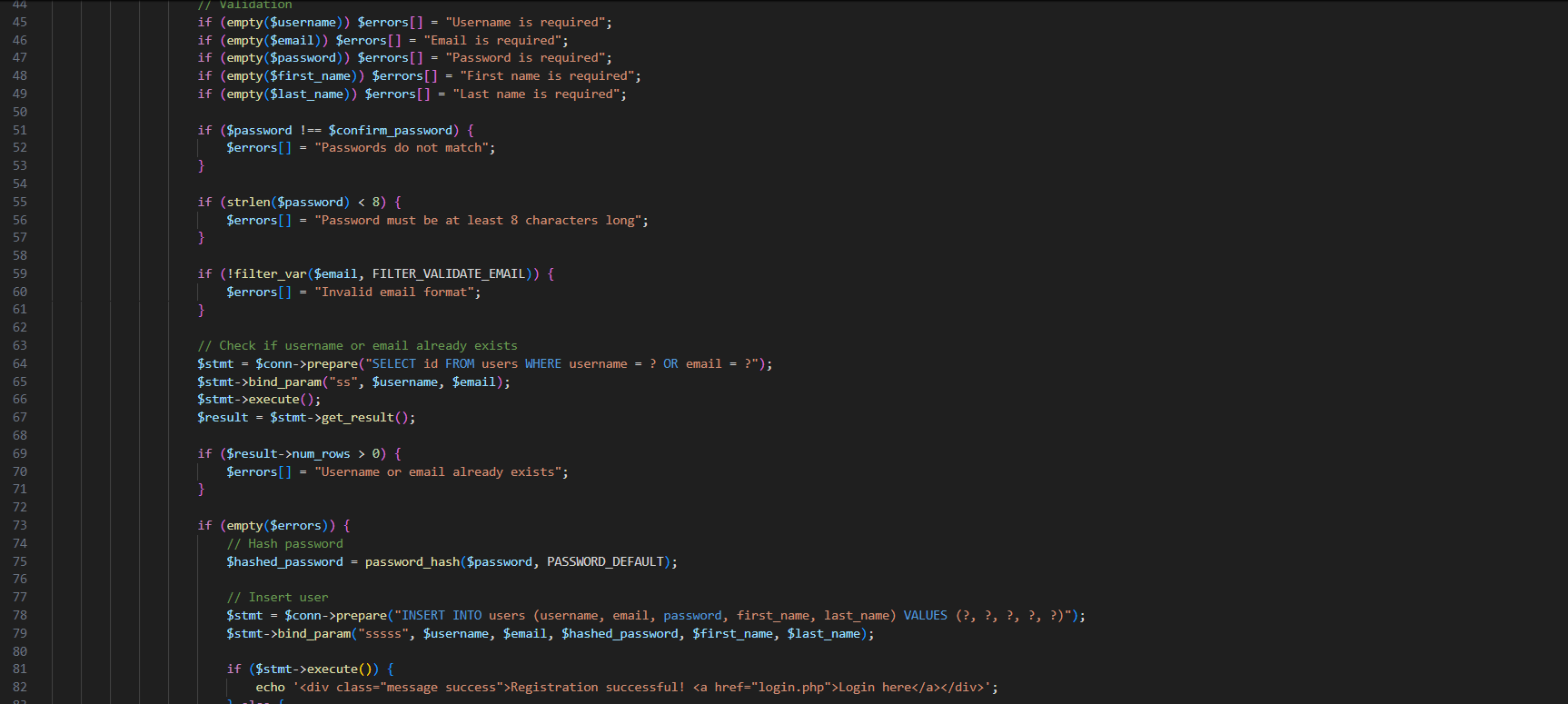
}

Code Screenshot Goes Here

A black background with red and white lights

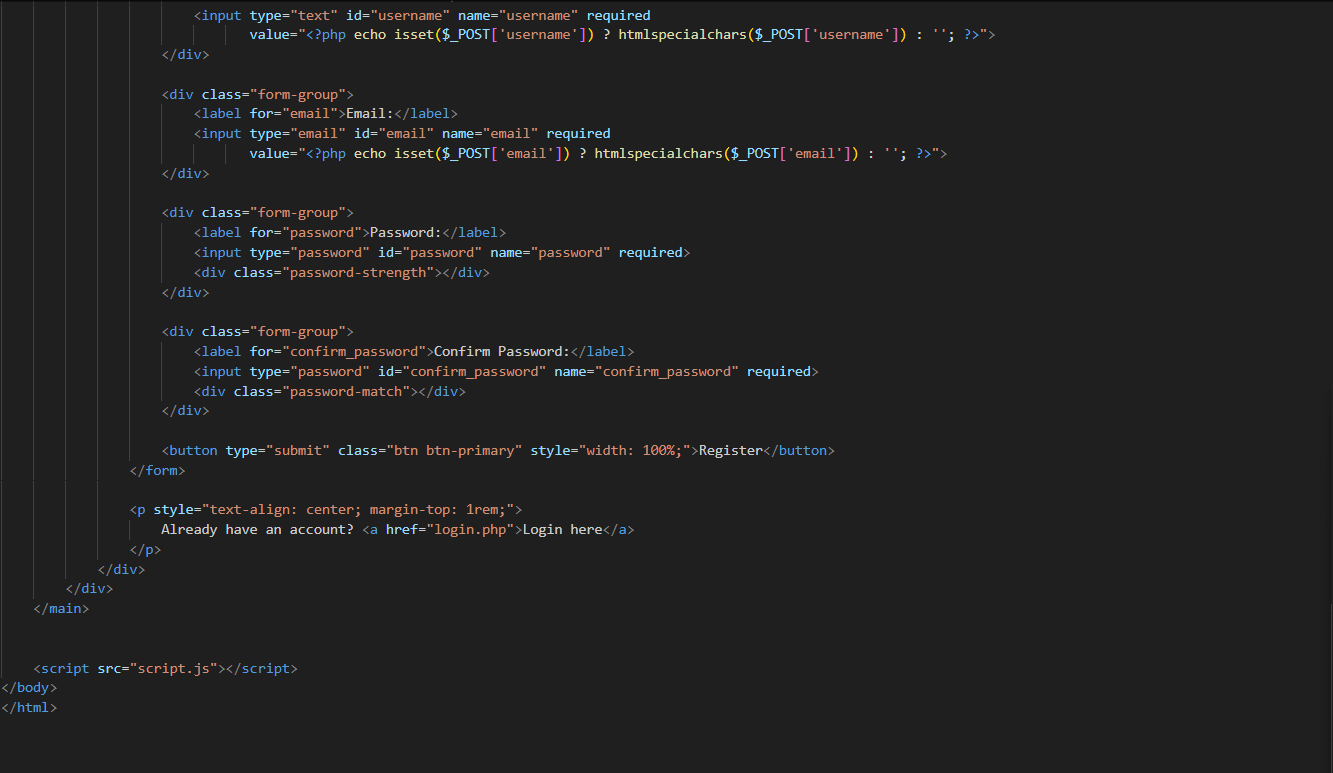
AI-generated content may be incorrect.





A screen shot of a computer

AI-generated content may be incorrect.



## PHP Webpage Screenshot

Webpage Description here

 Features of this page:

    1. User registration with username, email, password, first name, and last name

    2. Validation of username, email, password, first name, and last name

    3. Password hashing using password\_hash()

    4. Checking if username or email already exists in the database

    5. Inserting user into the database

    6. Displaying success or error messages

Webpage Screenshot Goes Here

A screenshot of a computer

AI-generated content may be incorrect.

# Login Webpage

## Login Page Information

##### Description:

The login page provides a simple interface for existing users to authenticate themselves by entering their username and password. The system validates the credentials against the database and creates a user session upon successful authentication. If the login attempt fails, the page displays an error message to help users understand what went wrong with their login attempt.

##### Key Points about the Page:

<List any tricky or special points on the php page>

* Point 1
* Point 2
* Point 3

## PHP Code Screenshot

Line Description here

**### Chunk 1: Authentication Logic**

```php

if ($\_SERVER['REQUEST\_METHOD'] === 'POST') {

    $username = trim($\_POST['username']);

    $password = $\_POST['password'];

    $errors = [];

    if (empty($username)) $errors[] = "Username is required";

    if (empty($password)) $errors[] = "Password is required";

    if (empty($errors)) {

        // Get user from database

        $stmt = $conn->prepare("SELECT id, username, password, first\_name, last\_name FROM users WHERE username = ?");

        $stmt->bind\_param("s", $username);

        $stmt->execute();

        $result = $stmt->get\_result();

        if ($result->num\_rows === 1) {

            $user = $result->fetch\_assoc();

            if (password\_verify($password, $user['password'])) {

                // Login successful

                $\_SESSION['user\_id'] = $user['id'];

                $\_SESSION['username'] = $user['username'];

                // ... set more session variables ...

                // Generate session token

                $session\_token = bin2hex(random\_bytes(32));

                $\_SESSION['session\_token'] = $session\_token;

                // Store session in database

                $stmt = $conn->prepare("INSERT INTO user\_sessions (user\_id, session\_token, expires\_at) VALUES (?, ?, DATE\_ADD(NOW(), INTERVAL 24 HOUR))");

                $stmt->bind\_param("is", $user['id'], $session\_token);

                $stmt->execute();

                echo '<div class="message success">Login successful! Redirecting...</div>';

                header("refresh:2;url=index.php");

                exit();

            } else {

                $errors[] = "Invalid username or password";

            }

        } else {

            $errors[] = "Invalid username or password";

        }

    }

    foreach ($errors as $error) {

        echo '<div class="message error">' . htmlspecialchars($error) . '</div>';

    }

}

```

**\*\*Explanation:\*\*** This chunk processes login attempts by validating credentials against the database. It uses prepared statements for security, verifies passwords with password\_verify(), creates user sessions, and generates secure session tokens stored in the database. Upon successful login, it redirects to the home page.

**### Chunk 2: Login Form**

```php

<form method="POST" action="">

    <div class="form-group">

        <label for="username">Username:</label>

        <input type="text" id="username" name="username" required

               value="<?php echo isset($\_POST['username']) ? htmlspecialchars($\_POST['username']) : ''; ?>">

    </div>

    <div class="form-group">

        <label for="password">Password:</label>

        <input type="password" id="password" name="password" required>

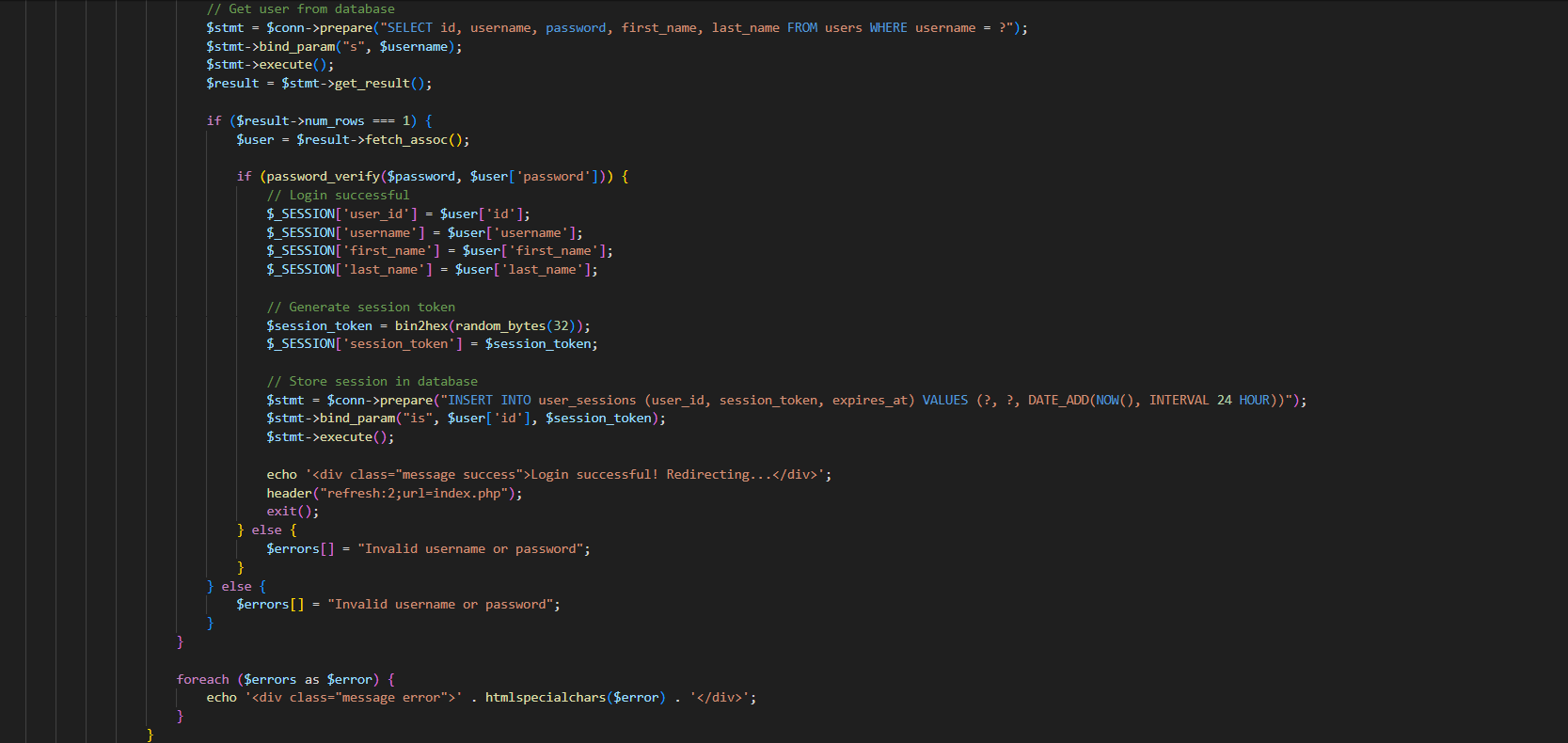
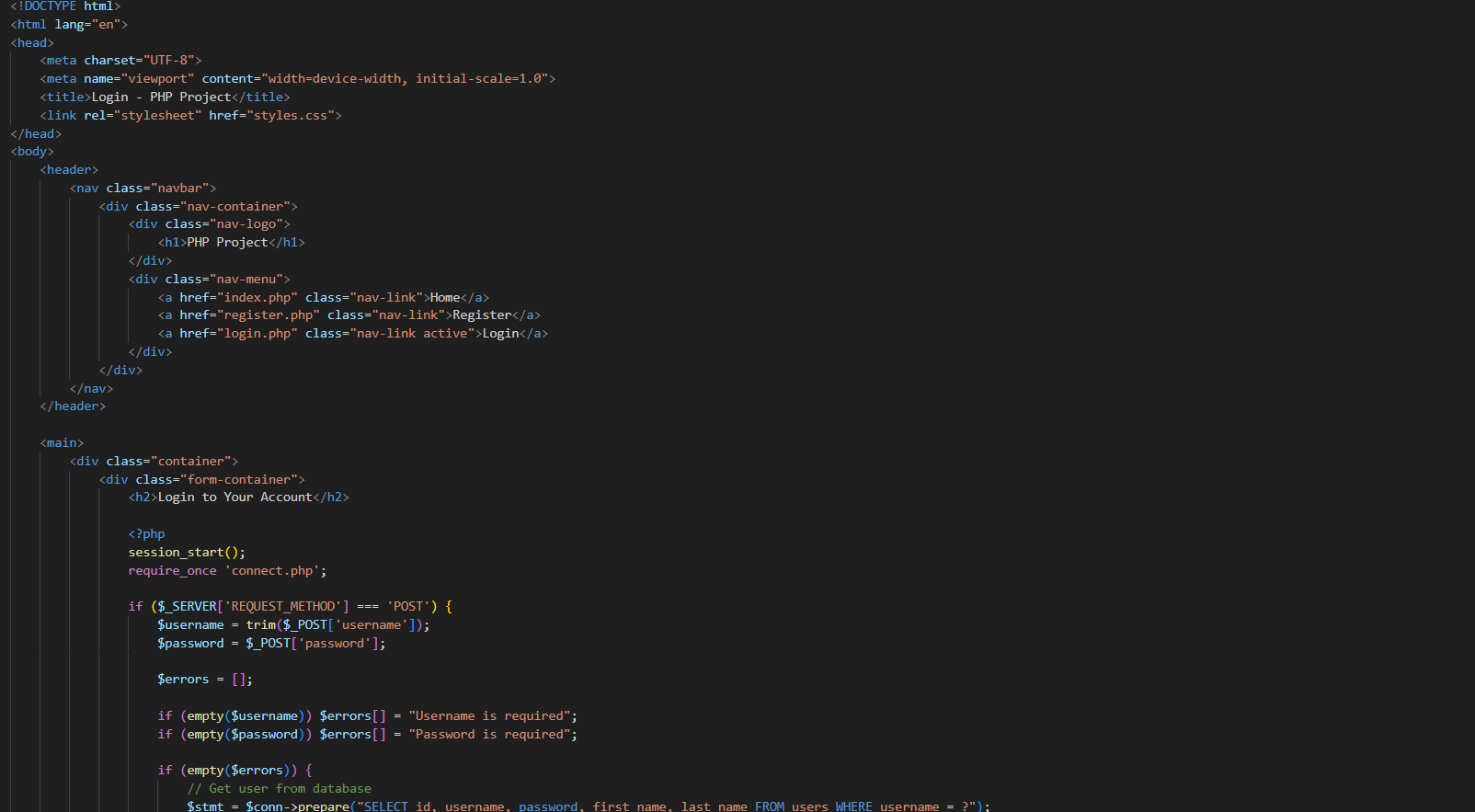
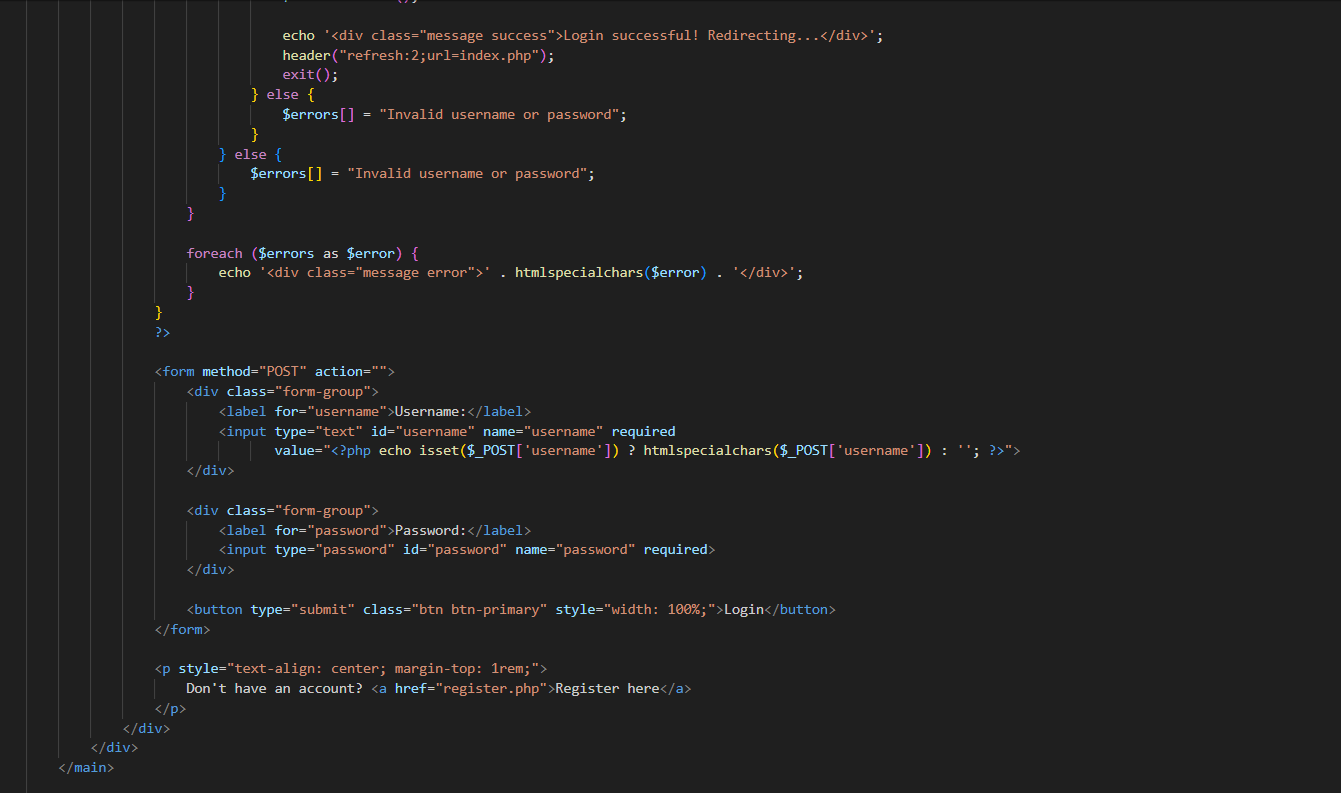
    </div>

    <button type="submit" class="btn btn-primary" style="width: 100%;">Login</button>

</form>

```

**\*\*Explanation:\*\*** This creates a simple login form with username and password fields. The form maintains user input on validation errors and uses appropriate HTML5 form attributes for better user experience.

### PHP Webpage Screenshot

Webpage Description here

- User registration with username, email, and secure password hashing

- Secure login with session management and secure password verification

- User profiles with personal information and account details

- Responsive design for mobile and desktop devices

This section should list all the features included on the webpage.

Webpage Screenshot Goes Here

A screenshot of a login form

AI-generated content may be incorrect.

# Profile.php page

## Profile Page Information

##### Description:

The profile page displays the user's personal information and provides functionality to update their account details. Users can view their username, email, full name, and account creation date in a clean, organized layout. The page includes a form that allows users to modify their first name, last name, and email address, with the changes being immediately saved to the database.

##### Key Points about the Page:

<List any tricky or special points on the php page>

* Point 1
* Point 2
* Point 3

## PHP Code Screenshot

Line Description here

**### Chunk 1: User Data Retrieval**

```php

// Check if user is logged in

if (!isset($\_SESSION['user\_id'])) {

    header("Location: login.php");

    exit();

}

$user\_id = $\_SESSION['user\_id'];

// Get user information

$stmt = $conn->prepare("SELECT username, email, first\_name, last\_name, created\_at, updated\_at FROM users WHERE id = ?");

$stmt->bind\_param("i", $user\_id);

$stmt->execute();

$result = $stmt->get\_result();

if ($result->num\_rows === 1) {

    $user = $result->fetch\_assoc();

} else {

    echo '<div class="message error">User not found.</div>';

    exit();

}

```

**\*\*Explanation:\*\*** This chunk ensures the user is authenticated before accessing the profile page. It retrieves the user's complete information from the database using a prepared statement and handles cases where the user might not be found.

**### Chunk 2: Profile Update Logic**

```php

if ($\_SERVER['REQUEST\_METHOD'] === 'POST' && isset($\_POST['update\_profile'])) {

    $first\_name = trim($\_POST['first\_name']);

    $last\_name = trim($\_POST['last\_name']);

    $email = trim($\_POST['email']);

    $errors = [];

    if (empty($first\_name)) $errors[] = "First name is required";

    // ... more validation ...

    if (empty($errors)) {

        $stmt = $conn->prepare("UPDATE users SET first\_name = ?, last\_name = ?, email = ? WHERE id = ?");

        $stmt->bind\_param("sssi", $first\_name, $last\_name, $email, $user\_id);

        if ($stmt->execute()) {

            $\_SESSION['first\_name'] = $first\_name;

            $\_SESSION['last\_name'] = $last\_name;

            echo '<div class="message success">Profile updated successfully!</div>';

            // Refresh user data

            $user['first\_name'] = $first\_name;

            $user['last\_name'] = $last\_name;

            $user['email'] = $email;

        } else {

            echo '<div class="message error">Failed to update profile. Please try again.</div>';

        }

    } else {

        foreach ($errors as $error) {

            echo '<div class="message error">' . htmlspecialchars($error) . '</div>';

        }

    }

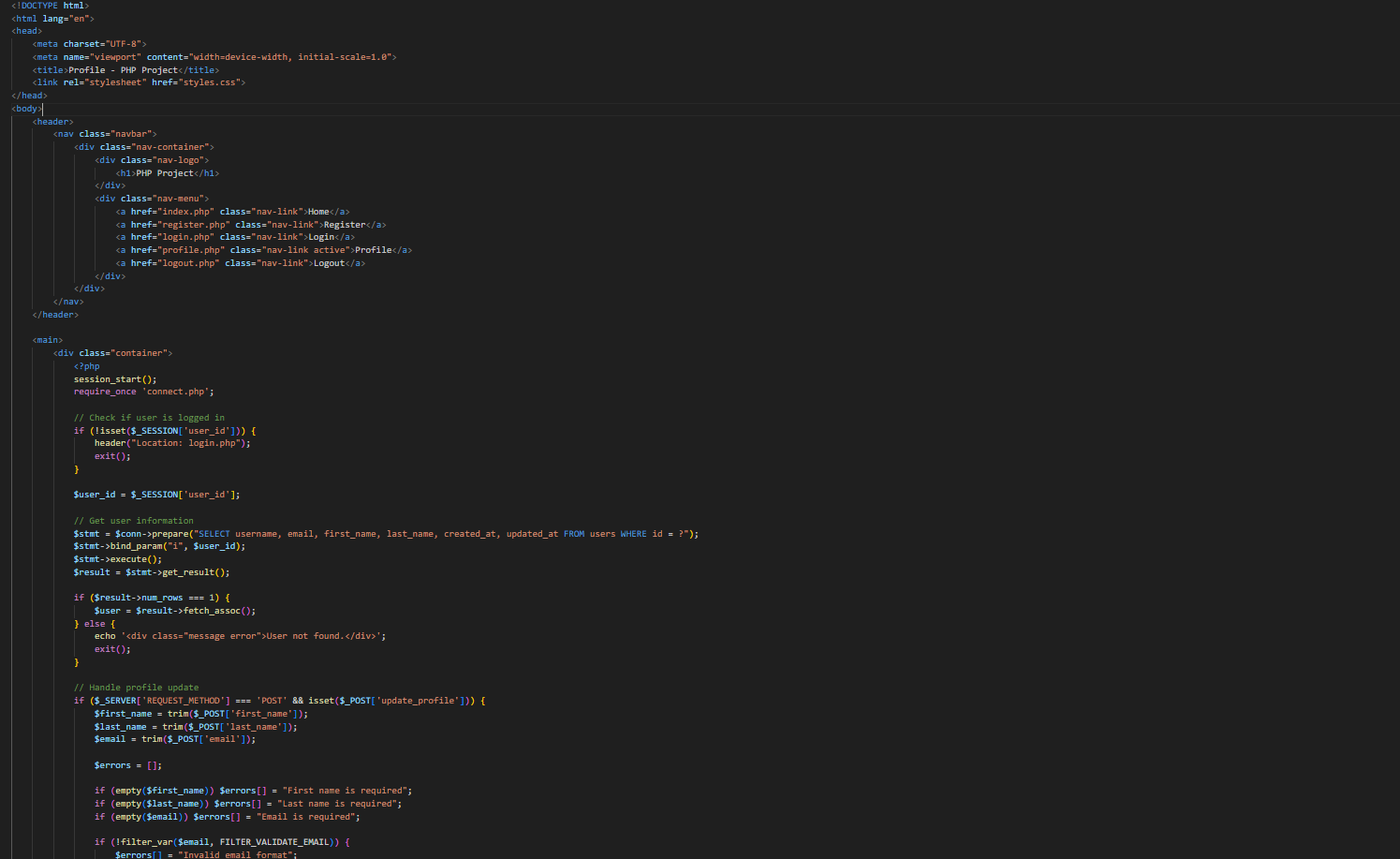
}

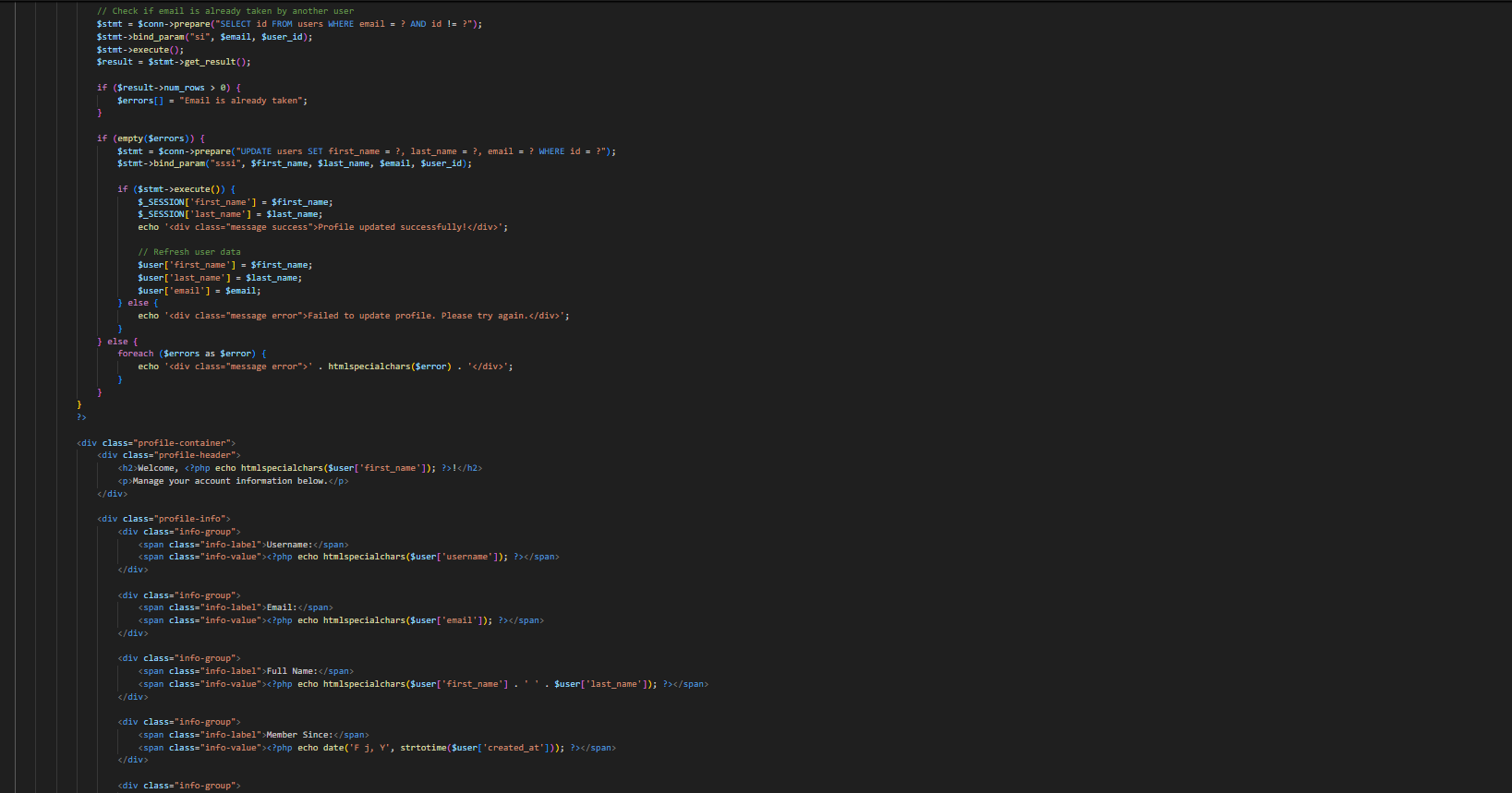
```

**\*\*Explanation:\*\*** This handles profile update requests by validating the new information, checking for email conflicts, and updating the database. It also updates the user session data and refreshes the displayed user information after successful updates.

This section should include an explanation of each chunk of code in your php page

Code Screenshot Goes Here





A screen shot of a computer

AI-generated content may be incorrect.

# Logout.php page

## Logout Page Information

##### Description:

The logout page handles the secure termination of user sessions by destroying session data and removing session records from the database. It provides a confirmation message to the user that they have been successfully logged out. The page automatically redirects users back to the home page after a brief delay to complete the logout process.

##### Key Points about the Page:

<List any tricky or special points on the php page>

* Point 1
* Point 2
* Point 3

## PHP Code Screenshot

Line Description here

**### Chunk 1: Session Cleanup**

```php

if (isset($\_SESSION['user\_id'])) {

    $user\_id = $\_SESSION['user\_id'];

    $session\_token = $\_SESSION['session\_token'] ?? null;

    // Remove session from database

    if ($session\_token) {

        $stmt = $conn->prepare("DELETE FROM user\_sessions WHERE session\_token = ?");

        $stmt->bind\_param("s", $session\_token);

        $stmt->execute();

    }

    // Destroy session

    session\_unset();

    session\_destroy();

    echo '<div class="message success">You have been successfully logged out.</div>';

    echo '<p style="text-align: center; margin-top: 1rem;">Redirecting to home page...</p>';

    // Redirect after 2 seconds

    header("refresh:2;url=index.php");

} else {

    echo '<div class="message info">You are not currently logged in.</div>';

    echo '<p style="text-align: center; margin-top: 1rem;"><a href="index.php">Return to Home</a></p>';

}

```

**\*\*Explanation:\*\*** This chunk handles the complete logout process by removing the session token from the database, destroying the PHP session, and providing user feedback. It includes automatic redirection to the home page after successful logout.

This section should include an explanation of each chunk of code in your php page

Code Screenshot Goes Here

A screenshot of a computer

AI-generated content may be incorrect.

# 404 Error Page

## 404 Error Page Page Information

##### Description:

The 404 error page provides a user-friendly response when visitors attempt to access non-existent URLs on the website. It displays a clear error message explaining that the requested page could not be found. The page includes helpful navigation links to guide users back to valid sections of the website.

##### Key Points about the Page:

<List any tricky or special points on the php page>

* Point 1
* Point 2
* Point 3

## PHP Code Screenshot

Line Description here

**### Chunk 1: HTML Structure and Navigation**

**```php**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta name="viewport" content="width=device-width, initial-scale=1.0">**

**<title>404 - Page Not Found - PHP Project</title>**

**<link rel="stylesheet" href="styles.css">**

**<style>**

**.error-container {**

**text-align: center;**

**padding: 4rem 0;**

**}**

**.error-code {**

**font-size: 8rem;**

**color: #e74c3c;**

**font-weight: bold;**

**margin-bottom: 1rem;**

**}**

**.error-title {**

**font-size: 2rem;**

**color: #2c3e50;**

**margin-bottom: 1rem;**

**}**

**.error-message {**

**font-size: 1.2rem;**

**color: #7f8c8d;**

**margin-bottom: 2rem;**

**}**

**<!-- More CSS styles... -->**

**</style>**

**</head>**

**<body>**

**<header>**

**<nav class="navbar">**

**<div class="nav-container">**

**<div class="nav-logo">**

**<h1>PHP Project</h1>**

**</div>**

**<div class="nav-menu">**

**<a href="index.php" class="nav-link">Home</a>**

**<a href="register.php" class="nav-link">Register</a>**

**<a href="login.php" class="nav-link">Login</a>**

**</div>**

**</div>**

**</nav>**

**</header>**

**```**

**\*\*Explanation:\*\* This chunk sets up the basic HTML structure for the 404 error page with proper meta tags and links to the CSS file. It includes custom CSS styles specifically for the error page layout and creates the header navigation. The CSS defines the visual styling for the error code, title, and message elements.**

**### Chunk 2: Error Content Display**

**```php**

**<main>**

**<div class="container">**

**<div class="error-container">**

**<div class="error-code">404</div>**

**<h1 class="error-title">Page Not Found</h1>**

**<p class="error-message">Sorry, the page you are looking for doesn't exist or has been moved.</p>**

**<div class="error-actions">**

**<a href="index.php" class="btn btn-primary">Go Home</a>**

**<a href="javascript:history.back()" class="btn btn-secondary">Go Back</a>**

**</div>**

**```**

**\*\*Explanation:\*\* This section creates the main error content with a large 404 error code, clear title, and descriptive message. It provides two action buttons - one to return to the home page and another to go back to the previous page using JavaScript's history.back() function.**

**### Chunk 3: Navigation Links and Technical Details**

**```php**

**<script src="script.js"></script>**

**</body>**

**</html>**

**```**

**\*\*Explanation:\*\* This chunk adds helpful navigation links to popular pages and technical debugging information. The helpful links section provides quick access to main site areas, while the technical details section shows server information and the requested URL for debugging purposes. The page maintains the standard footer and includes the JavaScript file.**

This section should include an explanation of each chunk of code in your php page

Code Screenshot Goes Here

A screenshot of a computer program

AI-generated content may be incorrect.

## PHP Webpage Screenshot

Webpage Description here

  Feature: Helpful links at the bottom of the page for easier navigation

Webpage Screenshot Goes Here

A screenshot of a error message

AI-generated content may be incorrect.