UNFILTERED NEWS PORTAL

Author: PAUL MWANIKI WAINAINA
Date: 4/JULY/2025

// FLATIRON SCHOOL

LAMP Stack

LAMP STACK Information

Description:

The acronym LAMP stands for Linux (operating system), Apache (web server), MySQL (database server), and PHP (programming language). These components work together in layers: Linux provides the foundation, Apache handles web requests, PHP processes dynamic content by interacting with MySQL, and the final output is sent back to the user's browser.

System Specs:

- **Operating System:** Ubuntu 24.04 LTS
- **Apache Version:** Apache/2.4.58 (Ubuntu)
- **Database:** mysql Ver 15.1 Distrib 10.11.13-MariaDB
- **PHP:** PHP 8.3.6

Index.php

Index.php Information

Description:

The index page serves as the main entry point for the "Unfiltered News Portal" website, providing users with a welcoming introduction and easy navigation to key sections such as login, news, and profile pages. Its purpose is to guide visitors efficiently to the content or features they want to access, acting as a centralized hub. By offering clear links and a brief message, the index page enhances user experience and helps organize the website's structure

Key Points about the Page:

- Title and Branding: The page is titled "Unfiltered News Portal" and prominently displays the site name in the header.
- Navigation Menu: Contains links to essential pages — Login, News Page, and Profile Page — for easy user access.
- User Prompt: Encourages users to log in to access personalized news and profile features.
- Styling: Links to an external CSS file (css/global.css) for consistent styling across the site.

Index.php Code Screenshot

<!DOCTYPE html>

Declares the document type and version of HTML (HTML5), ensuring the browser renders the page correctly.

<title>Unfiltered News Portal</title>

Defines the title of the webpage shown on the browser tab

<link rel="stylesheet" href="css/global.css" />

Links an external CSS file (global.css) for styling the webpage, ensuring consistent design.

`<h1>Unfiltered News Portal</h1>` displays the main heading or site title prominently.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Unfiltered News Portal</title>
        <link rel="stylesheet" href="css/global.css" />
</head>
<body>
    <header>
        <h1>Unfiltered News Portal</h1>
    </header>

<div class="container">
        <p>Welcome to the latest news and updates</p>
    <nav >
        <a href="login.php">Login</a>
        <a href="news.php">News Page</a>
        <a href="profile.php">Profile Page</a>
    </nav>

    <main>
        <p>Please <a href="login.php">log in</a> to access your personalized news and profile.</p>
    </main>
</div>
</body>
</html>
```

Index.php Code Screenshot

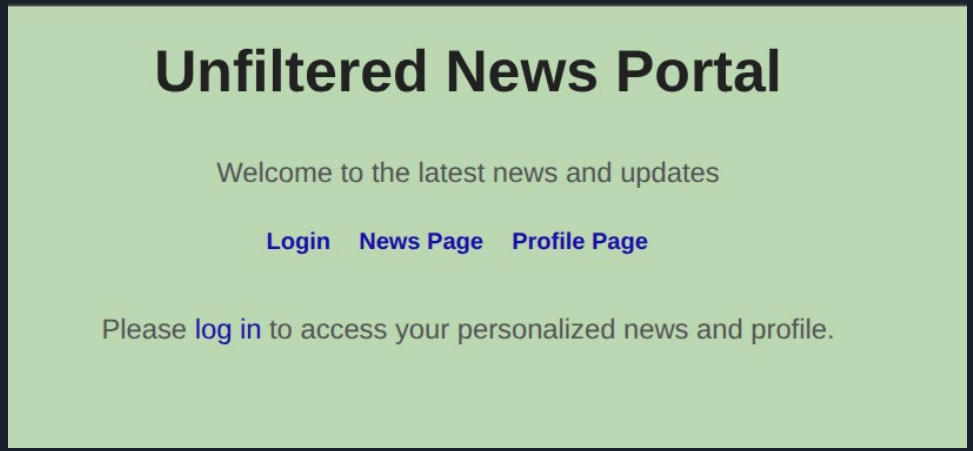
Defines the navigation section with links to important pages:
Login — Link to the login page.
News Page — Link to the news content.
Profile Page — Link to the user's profile.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Unfiltered News Portal</title>
        <link rel="stylesheet" href="css/global.css" />
</head>
<body>
    <header>
        <h1>Unfiltered News Portal</h1>
    </header>

<div class="container">
        <p>Welcome to the latest news and updates</p>
    <nav >
        <a href="login.php">Login</a>
        <a href="news.php">News Page</a>
        <a href="profile.php">Profile Page</a>
    </nav>

    <main>
        <p>Please <a href="login.php">log in</a> to access your personalized news and profile.</p>
    </main>
</div>
</body>
</html>
```

PHP Webpage Screenshot

- Description:

The index page serves as the main entry point for the "Unfiltered News Portal" website, providing users with a welcoming introduction and easy navigation to key sections such as login, news, and profile pages. Its purpose is to guide visitors efficiently to the content or features they want to access, acting as a centralized hub. By offering clear links and a brief message, the index page enhances user experience and helps organize the website's structure



Database Information

Database Information

Description:

The database used is essential because it stores and manages user-specific data that enables personalized content delivery based on individual interests . By tracking each user's preferences and interactions, the database allows the site to serve tailored news articles that match their unique tastes, improving user engagement and satisfaction . This personalization helps reduce information overload by filtering out irrelevant content and highlighting what matters most to each user.

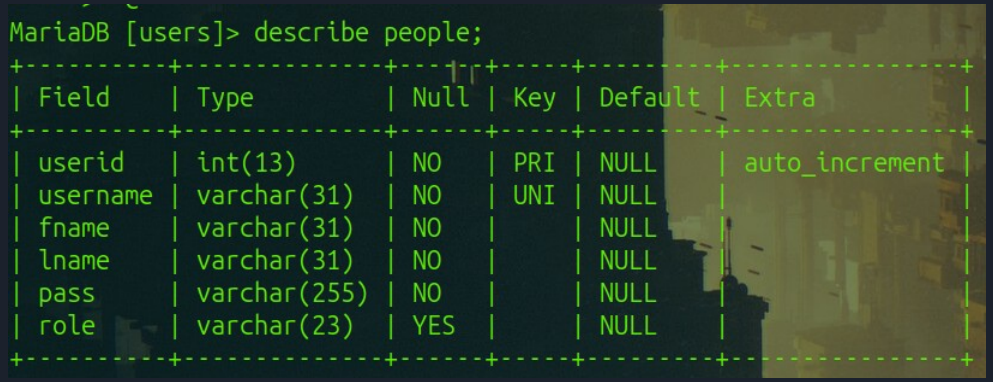
Key Points about the Page:

Handling Many-to-Many Relationships: A junction table had to be added to store each users specific interests .

unique constraint on the username column. This feature had to to be added so that every username added was distinct across all user records, preventing duplicate entries and ensuring data integrity.

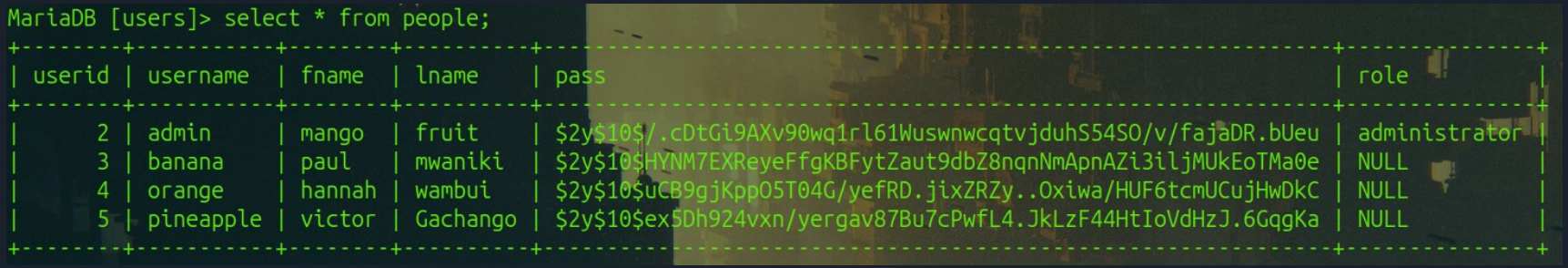
Database Screenshot

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



```
MariaDB [users]> describe people;
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| userid   | int(13)      | NO   | PRI | NULL    | auto_increment |
| username | varchar(31)  | NO   | UNI | NULL    |                |
| fname    | varchar(31)  | NO   |     | NULL    |                |
| lname    | varchar(31)  | NO   |     | NULL    |                |
| pass     | varchar(255) | NO   |     | NULL    |                |
| role     | varchar(23)  | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```

Screenshot of the database with all the new users added



```
MariaDB [users]> select * from people;
+--------+----------+--------+----------+--------------------------------------------------------------+---------------+
| userid | username | fname  | lname    | pass                                                         | role          |
+--------+----------+--------+----------+--------------------------------------------------------------+---------------+
|      2 | admin    | mango  | fruit    | $2y$10$/.cDtGi9AXv90wq1rl61WuswnwcqtvjduhS54SO/v/fajaDR.bUeu | administrator |
|      3 | banana   | paul   | mwaniki  | $2y$10$HYNM7EXReyeFfgKBFytZaut9dbZ8nqnNmApnAZi3iljMUkEoTMa0e | NULL          |
|      4 | orange   | hannah | wambui   | $2y$10$uCB9gjKppO5T04G/yefRD.jixZRZy..Oxiwa/HUF6tcmUCujHwDkC | NULL          |
|      5 | pineapple| victor | Gachango | $2y$10$ex5Dh924vxn/yergav87Bu7cPwfL4.JkLzF44HtIoVdHzJ.6GqgKa | NULL          |
+--------+----------+--------+----------+--------------------------------------------------------------+---------------+
```

phptest.php

phptest.php Information

Description:

A phptest.php page is typically used as a simple script to verify that PHP is properly installed and running on a web server. It often contains basic PHP code that outputs a message or performs a small function, allowing developers to confirm the server correctly processes PHP code. Beyond this, such a page can also serve as a testing ground for new PHP features or snippets before integrating them into the main website.

Key Points about the Page:

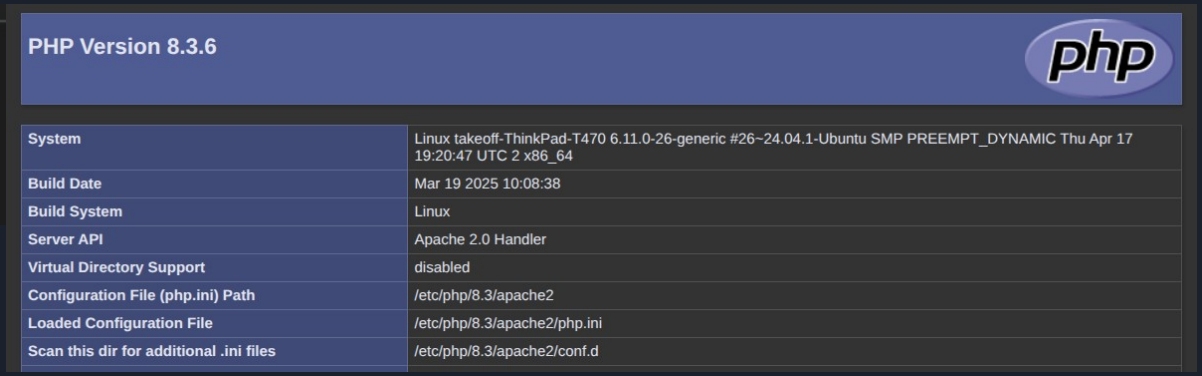
Security Risk if Publicly Accessible: Because it reveals sensitive information about the server and PHP configuration, leaving this page accessible to unauthorized users can pose a security risk. It’s best to restrict access or remove the page after testing.

phptest.php Code Screenshot

Code Screenshot

```
<?php phpinfo(); ?>
~
~
~
```

Webpage Screenshot



PHP Version 8.3.6

System	Linux takeoff-ThinkPad-T470 6.11.0-26-generic #26~24.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Apr 17 19:20:47 UTC 2 x86_64
Build Date	Mar 19 2025 10:08:38
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.3/apache2
Loaded Configuration File	/etc/php/8.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.3/apache2/conf.d

Connect.php page

Connect Page Information

Description:

A connect.php page is a dedicated PHP script responsible for establishing and managing the connection between a website and its database. Its primary purpose is to centralize the database connection code—such as server name, username, password, and database name—so that other pages can easily include this file to access the database without repeating connection details. This approach promotes code reusability, simplifies maintenance, and ensures consistent database access across the website. Handling connection errors and setting up the database link in one place improves the reliability and security of the application.

Key Points about the Page:

Use of PDO with Prepared Statements: So as to secure database interactions, the connection.php file utilizes PHP Data Objects (PDO) with prepared statements to prevent SQL injection attacks and improve compatibility across different database systems

PHP Code Screenshot

```php
<?php
$servername = "localhost";
$dbname = 'users';
$dbuser = 'root';
$dbpass = 'banana';

$dsn = "mysql:host=$servername;dbname=$dbname;charset=utf8";
$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::ATTR_EMULATE_PREPARES => false,
];

try {
    $pdo = new PDO($dsn, $dbuser, $dbpass, $options);
} catch (\PDOException $e) {
    throw new \PDOException($e->getMessage(), (int)$e->getCode());
}
?>
~
```

PHP Code Explanation

Defining variables for the database connection parameters:

- `$servername`: The hostname where the database server is located (usually "localhost" for local servers).
- `$dbname`: The name of the database to connect to (`users`).
- `$dbuser`: The username for authenticating with the database (`root`).
- `$dbpass`: The password associated with the database user (`banana`).

Creating the Data Source Name (DSN) string which is used by PDO to specify the database driver (mysql), host, database name, and character set (utf8), ensuring proper encoding.
$dsn = "mysql:host=$servername;dbname=$dbname;charset=utf8";

Defining an array of options for the PDO connection:
PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION: Configures PDO to throw exceptions on database errors, which helps with error handling.
PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC: Sets the default fetch mode to associative arrays, making it easier to work with query results.
PDO::ATTR_EMULATE_PREPARES => false: Disables emulated prepared statements, forcing the use of native prepared statements for better security.

PHP Code Explanation

```
try {
    $pdo = new PDO($dsn, $dbuser, $dbpass, $options);
} catch (\PDOException $e) {
    throw new \PDOException($e->getMessage(), (int)$e->getCode());
}
```

Attempts to create a new PDO instance to connect to the database using the DSN, username, password, and options.

If the connection fails, it catches the PDOException and rethrows it with the error message and code, allowing the calling code to handle or log the error appropriately.

Account Creation Web Page

Account Creation Page Information

Description:
An account creation page allows new users to register on a website by providing essential information such as username, email, and password. Its purpose is to collect and validate this data, then securely store it in the database to create a unique user profile. This page often includes checks to prevent duplicate usernames or emails and ensures passwords are handled safely, typically by hashing before storage. By enabling users to create accounts, the website can offer personalized experiences

Key Points about the Page:

Basic validation ensures all required fields (username, fname, lname, pass) are filled.

Before creating a new account, the code checks if the username already exists in the database using a prepared statement.

Passwords are hashed securely using password_hash() with the default algorithm (usually bcrypt).

All database queries use prepared statements ($pdo->prepare()), which protect against SQL injection attacks by separating SQL logic from user input.

PHP Code Screenshot

```php
<?php
session_start();

if (isset($_SESSION['user_id'])) {
    header("Location: profile.php");
    exit;
}

include_once 'connect.php';

$message = "";

if (isset($_POST['sca'])) {
    $username = trim($_POST['username']);
    $fname = trim($_POST['fname']);
    $lname = trim($_POST['lname']);
    $pass = trim($_POST['pass']);

    // Basic validation (add more as needed)
    if (empty($username) || empty($fname) || empty($lname) || empty($pass)) {
        $message = "Please fill in all fields.";
    } else {
        // Check if username already exists
        $stmt = $pdo->prepare("SELECT userid FROM people WHERE username = ?");
        $stmt->execute([$username]);
        if ($stmt->fetch()) {
            $message = "Username already taken. Please choose another.";
        } else {
            // Hash password securely
            $passwordHash = password_hash($pass, PASSWORD_DEFAULT);

            $query = "INSERT INTO people (username, fname, lname, pass) VALUES (?, ?, ?, ?)";
            $stmt = $pdo->prepare($query);
            $success = $stmt->execute([$username, $fname, $lname, $passwordHash]);

            if ($success) {
                // Registration successful, redirect to login
                header("Location: login.php?registered=1");
                exit;
            } else {
                $message = "Registration failed. Please try again.";
            }
        }
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Register</title>
    <link rel="stylesheet" href="css/global.css" />
    <style>
        body { font-family: Arial, sans-serif; max-width: 400px; margin: 2rem auto; }
        .message { color: red; margin-bottom: 1rem; }
        form { display: flex; flex-direction: column; }
        label { margin-top: 10px; }
        input[type="text"], input[type="password"] { padding: 8px; font-size: 1rem; }
        input[type="submit"] { margin-top: 20px; padding: 10px; font-size: 1rem; cursor: pointer; }
        a { color: #1a0dab; text-decoration: none; }
        a:hover { text-decoration: underline; }
    </style>
</head>
<body>
```

PHP Code Screenshot

```php
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Register</title>
    <link rel="stylesheet" href="css/global.css" />
    <style>
        body { font-family: Arial, sans-serif; max-width: 400px; margin: 2rem auto; }
        .message { color: red; margin-bottom: 1rem; }
        form { display: flex; flex-direction: column; }
        label { margin-top: 10px; }
        input[type="text"], input[type="password"] { padding: 8px; font-size: 1rem; }
        input[type="submit"] { margin-top: 20px; padding: 10px; font-size: 1rem; cursor: pointer; }
        a { color: #1a0dab; text-decoration: none; }
        a:hover { text-decoration: underline; }
    </style>
</head>
<body>

<h1>Create Account</h1>

<?php if (!empty($message)): ?>
    <div class="message"><?php echo htmlspecialchars($message, ENT_QUOTES, 'UTF-8'); ?></div>
<?php endif; ?>

<form action="register.php" method="post" autocomplete="off">
    <label for="username">Username:</label>
    <input type="text" name="username" id="username" required />

    <label for="fname">First Name:</label>
    <input type="text" name="fname" id="fname" required />

    <label for="lname">Last Name:</label>
    <input type="text" name="lname" id="lname" required />
```

```php
        a { color: #1a0dab; text-decoration: none; }
        a:hover { text-decoration: underline; }
    </style>
</head>
<body>

<h1>Create Account</h1>

<?php if (!empty($message)): ?>
    <div class="message"><?php echo htmlspecialchars($message, ENT_QUOTES, 'UTF-8'); ?></div>
<?php endif; ?>

<form action="register.php" method="post" autocomplete="off">
    <label for="username">Username:</label>
    <input type="text" name="username" id="username" required />

    <label for="fname">First Name:</label>
    <input type="text" name="fname" id="fname" required />

    <label for="lname">Last Name:</label>
    <input type="text" name="lname" id="lname" required />

    <label for="pass">Password:</label>
    <input type="password" name="pass" id="pass" required />

    <input type="submit" name="sca" value="Create Account" />
</form>

<p>Already have an account? <a href="login.php">Login here</a>.</p>

</body>
</html>
```

PHP Code explanation

```
session_start();
if (isset($_SESSION['user_id'])) {
    header("Location: profile.php");
    exit;
}
```

Starts a session to track user data.

Checks if a user is already logged in by verifying if `$_SESSION['user_id']` exists.

If logged in, redirects the user to the profile page to prevent re-registration.

```
include_once 'connect.php';
```

Includes the external file connect.php which contains the database connection setup using PDO.

PHP Code explanation

```
if (isset($_POST['sca'])) {
    $username = trim($_POST['username']);
    $fname = trim($_POST['fname']);
    $lname = trim($_POST['lname']);
    $pass = trim($_POST['pass']);
```

Checks if the form has been submitted by looking for the sca POST parameter.
Retrieves and trims user input values for username, first name, last name, and password

```
if (empty($username) || empty($fname) || empty($lname) || empty($pass)) {
    $message = "Please fill in all fields.";
}
```

Ensures none of the required fields are empty.
If any field is empty, sets an error message.

PHP Code explanation

```
$stmt = $pdo->prepare("SELECT userid FROM people WHERE username = ?");
$stmt->execute([$username]);
if ($stmt->fetch()) {
    $message = "Username already taken. Please choose another.";
}
```

Uses a prepared statement to query the database for the submitted username.
If a record is found, sets an error message indicating the username is taken.

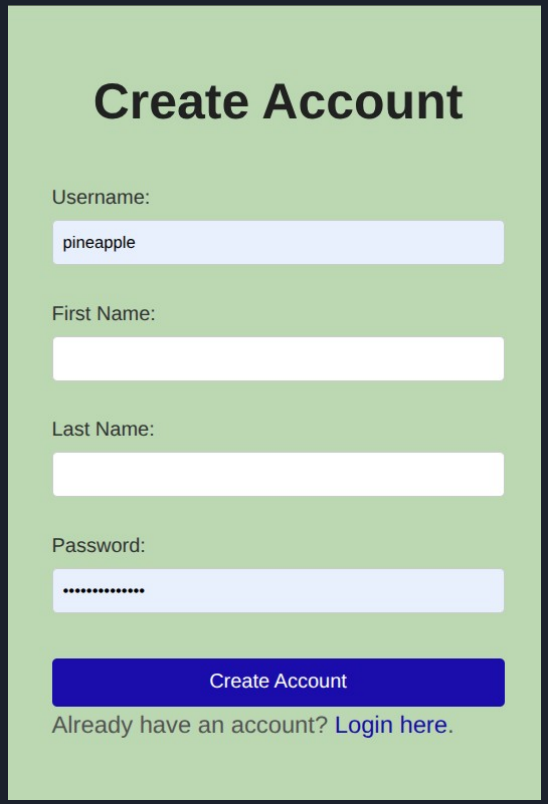
```
$passwordHash = password_hash($pass, PASSWORD_DEFAULT);
$query = "INSERT INTO people (username, fname, lname, pass) VALUES (?, ?, ?, ?)";
$stmt = $pdo->prepare($query);
$success = $stmt->execute([$username, $fname, $lname, $passwordHash]);
```

Hashes the password securely using PHP's password_hash() function.
Prepares and executes an SQL INSERT statement to add the new user to the database.

PHP Code explanation

The HTML part displays the registration form with fields for username, first name, last name, and password.
If there is a message (error or otherwise), it is safely displayed using htmlspecialchars() to prevent XSS attacks.
The form uses method="post" to securely send data to the same PHP page (register.php).

PHP Webpage Screenshot



PHP Webpage Screenshot explanation

The webpage includes several important features designed to provide a secure and user-friendly registration experience:

User Input Validation: The form requires all fields (username, first name, last name, password) to be filled, with server-side checks to ensure no empty submissions, enhancing data integrity.

Username Uniqueness Check: Before creating an account, the system checks the database to ensure the username is not already taken, preventing duplicate accounts and conflicts.

Secure Password Handling: Passwords are hashed securely using PHP's password_hash() function before storage, protecting user credentials against theft.

Use of Prepared Statements: All database queries use prepared statements to prevent SQL injection attacks, a critical security measure

Clean and Responsive UI: Simple CSS styling ensures the form is readable and user-friendly, with a clear call to action and navigation link for existing users to log in.

Login Webpage

Login Page Information

Description:

A login page is a critical component of a website that allows users to authenticate themselves by entering their username and password. Its primary purpose is to verify these credentials against stored data usually in a database to grant access to protected areas, such as user profiles or personalized content

Key Points about the Page:

Session Management: Properly start and manage sessions (session_start()) to maintain user login state across pages while preventing session hijacking.
Redirects After Login: Redirect users to appropriate pages after successful login to improve user experience and security.
Secure Password Verification: Use password_verify() to compare the submitted password with the hashed password stored in the database, ensuring secure authentication.

PHP Code Screenshot

```php
<?php
session_start();

include_once 'connect.php';

// --- CSRF Token Generation ---
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}

// --- Rate Limiting ---
if (!isset($_SESSION['login_attempts'])) {
    $_SESSION['login_attempts'] = 0;
    $_SESSION['last_attempt_time'] = time();
}

$max_attempts = 5;
$lockout_time = 300; // 5 minutes

$locked = false;
if ($_SESSION['login_attempts'] >= $max_attempts) {
    $elapsed = time() - $_SESSION['last_attempt_time'];
    if ($elapsed < $lockout_time) {
        $locked = true;
        $remaining = $lockout_time - $elapsed;
    } else {
        $_SESSION['login_attempts'] = 0; // Reset after lockout period
        $locked = false;
    }
}

$message = "";
```

```php
$message = "";

if (isset($_SESSION['user'])) {
    header("Location: index.php");
    exit;
}

if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['sca'])) {
    // --- CSRF Check ---
    if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== $_SESSION['csrf_token']) {
        $message = "Invalid session. Please refresh and try again.";
    } elseif ($locked) {
        $message = "Too many failed login attempts. Please try again in " . ceil($remaining/60) . " minutes.";
    } else {
        $username = trim($_POST['username']);
        $pass = trim($_POST['pass']);

        $query = "SELECT userid, username, pass FROM people WHERE username = ?";
        $stmt = $pdo->prepare($query);
        $stmt->execute([$username]);
        $row = $stmt->fetch(PDO::FETCH_ASSOC);

        // --- Password Verification ---
        if ($row && password_verify($pass, $row['pass'])) {
            $_SESSION['user'] = $row['userid'];
            $_SESSION['username'] = $row['username'];
            $_SESSION['login_attempts'] = 0; // Reset on success
            header("Location: profile.php");
            exit;
        } else {
            $_SESSION['login_attempts'] += 1;
            $_SESSION['last_attempt_time'] = time();
            // --- Generic error message (does not reveal if user exists) ---
```

PHP Code Screenshot

```php
        } else {
            $_SESSION['login_attempts'] += 1;
            $_SESSION['last_attempt_time'] = time();
            // --- Generic error message (does not reveal if user exists) ---
            $message = "Invalid username or password.";
        }
    }
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>Login</title>
    <link rel="stylesheet" href="css/global.css" />
    <style>
        body { font-family: Arial, sans-serif; max-width: 400px; margin: 2rem auto; }
        .message { color: red; margin-bottom: 1rem; }
        form { display: flex; flex-direction: column; }
        label { margin-top: 10px; }
        input[type="text"], input[type="password"] { padding: 8px; font-size: 1rem; }
        input[type="submit"] { margin-top: 20px; padding: 10px; font-size: 1rem; cursor: pointer; }
        a { color: #1a0dab; text-decoration: none; }
        a:hover { text-decoration: underline; }
    </style>
</head>
<body>
    <h1>Login</h1>

    <?php if (!empty($message)): ?>
        <div class="message"><?php echo htmlspecialchars($message, ENT_QUOTES, 'UTF-8'); ?></div>
    <?php endif; ?>
```

```php
    </style>
</head>
<body>
    <h1>Login</h1>

    <?php if (!empty($message)): ?>
        <div class="message"><?php echo htmlspecialchars($message, ENT_QUOTES, 'UTF-8'); ?></div>
    <?php endif; ?>

    <?php if ($locked): ?>
        <div class="message">
            Too many failed login attempts. Please try again in <?php echo ceil($remaining/60); ?> minu
        </div>
    <?php else: ?>
        <form action="login.php" method="post" autocomplete="off">
            <label for="username">Username:</label>
            <input type="text" name="username" id="username" required />

            <label for="pass">Password:</label>
            <input type="password" name="pass" id="pass" required />

            <!-- CSRF Token -->
            <input type="hidden" name="csrf_token" value="<?php echo htmlspecialchars($_SESSION['csrf_t


            <input type="submit" name="sca" value="Login" />
        </form>
        <p>Don't have an account? <a href="register.php">Create one here</a>.</p>
    <?php endif; ?>

</body>
</html>
```

PHP Code explanation

```
include_once 'connect.php';
```

Includes the database connection script (connect.php) to enable database operations using PDO.

```
if (empty($_SESSION['csrf_token'])) {
    $_SESSION['csrf_token'] = bin2hex(random_bytes(32));
}
```

Generates a cryptographically secure CSRF token (if not already set) and stores it in the session.

```
if (!isset($_SESSION['login_attempts'])) {
    $_SESSION['login_attempts'] = 0;
    $_SESSION['last_attempt_time'] = time();
}
```

Initializes login attempt tracking variables in the session to limit brute-force login attempts.

PHP Code explanation

```
$max_attempts = 5;
$lockout_time = 300; // 5 minutes

$locked = false;
if ($_SESSION['login_attempts'] >= $max_attempts) {
    $elapsed = time() - $_SESSION['last_attempt_time'];
    if ($elapsed < $lockout_time) {
        $locked = true;
        $remaining = $lockout_time - $elapsed;
    } else {
        $_SESSION['login_attempts'] = 0; // Reset after lockout period
        $locked = false;
    }
}
```

Defines a maximum number of login attempts (5) and a lockout period (5 minutes).

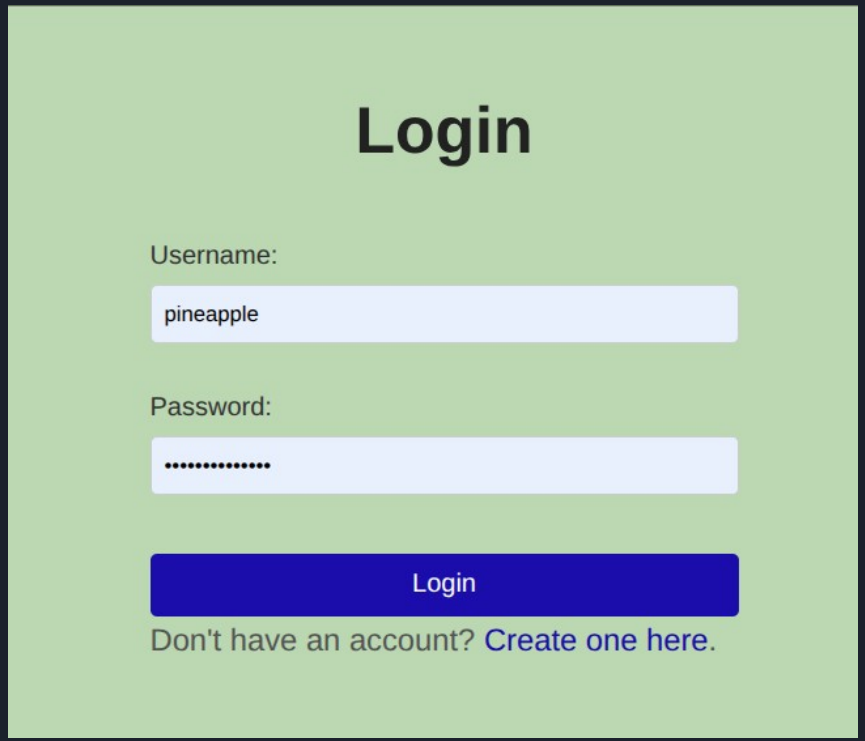
```
if (!isset($_POST['csrf_token']) || $_POST['csrf_token'] !== $_SESSION['csrf_token']) {
    $message = "Invalid session. Please refresh and try again.";
}
```

Validates the CSRF token submitted with the form against the session token.
If invalid, sets an error message to prevent CSRF attacks.

The page includes a simple, clean form styled with embedded CSS for usability and readability.

<input type="hidden" name="csrf_token" value="<?php echo htmlspecialchars($_SESSION['csrf_token']); ?>" />
The page also includes:
Username and password input fields (required).
A hidden input field containing the CSRF token.
A link to the registration page for new users.

PHP Webpage Screenshot



PHP Webpage Screenshot Description

This login webpage incorporates a comprehensive set of features designed to ensure security, usability, and a smooth user experience:

CSRF Protection: Implements a robust Cross-Site Request Forgery token mechanism to safeguard the login form against unauthorized submissions from external sites.

Rate Limiting and Lockout: Tracks failed login attempts and temporarily locks the user out after a configurable number of failures, protecting against brute-force attacks.

Secure Password Handling: Uses PHP's password_verify() function to safely compare hashed passwords, ensuring password security.

User Feedback: Provides clear, sanitized error messages and lockout notifications to inform users without exposing sensitive information.

Responsive and Accessible UI: Features a clean, simple form layout with CSS styling for readability and ease of use, including proper labels and input focus.

Navigation Links: Offers direct navigation to the registration page for new users, facilitating account creation.

Profile.php page

Profile Page Information

Description:

A **profile.php** page serves as the personalized dashboard for a logged-in user, displaying their account information and relevant data stored in the database. Its primary purpose is to provide users with access to their profile details, such as username, name, preferences, or other personalized content, and often allows them to update or manage this information.

Key Points about the Page:

User Authentication and Access Control: Ensure the page checks if a user is logged in before displaying profile data to prevent unauthorized access

Secure Data Retrieval: Use prepared statements when querying the database to fetch user details, protecting against SQL injection.

Error Handling: Gracefully handle cases where user data is missing or the database connection fails, providing user-friendly messages without exposing sensitive information.

PHP Code Screenshot

```php
<?php
ob_start();
session_start();
require_once 'connect.php';

if (!isset($_SESSION['user'])) {
    header("Location: index.php");
    exit;
}

$userid = $_SESSION['user'];

// Fetch user info
try {
    $query = "SELECT * FROM people WHERE userid = :userid";
    $stmt = $pdo->prepare($query);
    $stmt->execute([':userid' => $userid]);
    $userRow = $stmt->fetch(PDO::FETCH_ASSOC);

    if (!$userRow) {
        // User not found, log out or handle error
        header("Location: logout.php");
        exit;
    }

    // Fetch existing interests for the user
    $sql = "SELECT interest_id FROM user_interests WHERE people_userid = :userid";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([':userid' => $userid]);
    $existing_interests = $stmt->fetchAll(PDO::FETCH_COLUMN, 0); // Fetch interest_id as array

    // Handle form submission
    if ($_SERVER["REQUEST_METHOD"] === "POST" && !empty($_POST['interests']) && is_array($_POST['interests'])) {
        $interest_ids = $_POST['interests'];

        // Prepare insert statement once
        $insertSql = "INSERT INTO user_interests (people_userid, interest_id) VALUES (:userid, :interest_id)";
        $insertStmt = $pdo->prepare($insertSql);

        foreach ($interest_ids as $interest_id) {
            $interest_id = (int)$interest_id; // Cast to int for safety

            if (!in_array($interest_id, $existing_interests)) {
                $insertStmt->execute([
                    ':userid' => $userid,
                    ':interest_id' => $interest_id
                ]);
            }
        }

        // Redirect to avoid form resubmission and refresh existing interests
        header("Location: " . $_SERVER['PHP_SELF']);
        exit;
    }

    // Fetch all available interests
    $sql = "SELECT id, name FROM interests ORDER BY name";
    $stmt = $pdo->query($sql);
    $allInterests = $stmt->fetchAll(PDO::FETCH_ASSOC);

} catch (PDOException $e) {
    // Handle database errors (log them in production)
    die("Database error: " . htmlspecialchars($e->getMessage()));
}
?>
```

PHP Code Screenshot

```
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>You are logged in!</title>
        <link rel="stylesheet" href="css/global.css" />

</head>
<body>
    <p style="text-align: center;">Welcome to the protected page, <?php echo htmlspecialchars($userRow['fname'], ENT_QUOTES, 'UTF-8'); ?>!</p>
        <p style="text-align: center;"><a href="news.php" >View Your Personalized News</a></p>

    <?php if (isset($userRow['role']) && $userRow['role'] === "administrator"): ?>
        <p style="text-align: center;"><a href="edit.php">Edit password here </a></p>
    <?php endif; ?>

    <p style="text-align: center;"><a href="logout.php">Logout</a></p>

    <h1>Add Your Interests</h1>

    <?php if (empty($existing_interests)): ?>
        <?php if (!empty($allInterests)): ?>
            <form method="POST" action="">
                <label for="interests">Select your interests:</label><br>
                <?php foreach ($allInterests as $interest): ?>
                    <input type="checkbox"
                           name="interests[]"
                           value="<?php echo htmlspecialchars($interest['id'], ENT_QUOTES, 'UTF-8'); ?>"
                           id="interest-<?php echo htmlspecialchars($interest['id'], ENT_QUOTES, 'UTF-8'); ?>">
                    <label for="interest-<?php echo htmlspecialchars($interest['id'], ENT_QUOTES, 'UTF-8'); ?>">
                        <?php echo htmlspecialchars($interest['name'], ENT_QUOTES, 'UTF-8'); ?>
```

PHP Code Screenshot

```
    <?php endif; ?>

    <p style="text-align: center;"><a href="logout.php">Logout</a></p>

    <h1>Add Your Interests</h1>

    <?php if (empty($existing_interests)): ?>
        <?php if (!empty($allInterests)): ?>
            <form method="POST" action="">
                <label for="interests">Select your interests:</label><br>
                <?php foreach ($allInterests as $interest): ?>
                    <input type="checkbox"
                           name="interests[]"
                           value="<?php echo htmlspecialchars($interest['id'], ENT_QUOTES, 'UTF-8'); ?>"
                           id="interest-<?php echo htmlspecialchars($interest['id'], ENT_QUOTES, 'UTF-8'); ?>">
                    <label for="interest-<?php echo htmlspecialchars($interest['id'], ENT_QUOTES, 'UTF-8'); ?>">
                        <?php echo htmlspecialchars($interest['name'], ENT_QUOTES, 'UTF-8'); ?>
                    </label><br>
                <?php endforeach; ?>
                <br>
                <input type="submit" value="Save Interests">
            </form>
        <?php else: ?>
            <p>No interests available.</p>
        <?php endif; ?>
    <?php else: ?>
        <p style="text-align: center;">You have already added your interests.</p>
    <?php endif; ?>
</body>
</html>

<?php ob_end_flush(); ?>
```

PHP Code Screenshot Description

session_start() initializes the session to access session variables like the logged-in user ID.

require_once 'connect.php';
Includes the database connection file to use the PDO instance ($pdo) for database operations.

```
if (!isset($_SESSION['user'])) {
    header("Location: index.php");
    exit;
}
$userid = $_SESSION['user'];
```

Checks if the user is logged in by verifying the presence of $_SESSION['user'].
If not logged in, redirects to the homepage (index.php).

PHP Code Screenshot Description

```
try {
    $query = "SELECT * FROM people WHERE userid = :userid";
    $stmt = $pdo->prepare($query);
    $stmt->execute([':userid' => $userid]);
    $userRow = $stmt->fetch(PDO::FETCH_ASSOC);
    if (!$userRow) {
        header("Location: logout.php");
        exit;
    }
```

Prepares and executes a query to fetch all user details from the people table using the user ID.

```
    $sql = "SELECT interest_id FROM user_interests WHERE people_userid = :userid";
    $stmt = $pdo->prepare($sql);
    $stmt->execute([':userid' => $userid]);
    $existing_interests = $stmt->fetchAll(PDO::FETCH_COLUMN, 0);
```

Retrieves all interest IDs associated with the current user from the user_interests table.

PHP Code Screenshot Description

```
$sql = "SELECT id, name FROM interests ORDER BY name";
$stmt = $pdo->query($sql);
$allInterests = $stmt->fetchAll(PDO::FETCH_ASSOC);
```

Retrieves all interests from the interests table, ordered alphabetically by name.
These will be displayed as options for the user to select.

PHP Code Screenshot Description

```
<?php if (empty($existing_interests)): ?>
    <?php if (!empty($allInterests)): ?>
        <form method="POST" action="">
            <label for="interests">Select your interests:</label><br>
            <?php foreach ($allInterests as $interest): ?>
                <input type="checkbox"
                    name="interests[]"
                    value="<?php echo htmlspecialchars($interest['id'], ENT_QUOTES, 'UTF-8'); ?>"
                    id="interest-<?php echo htmlspecialchars($interest['id'], ENT_QUOTES, 'UTF-8'); ?>">
                <label for="interest-<?php echo htmlspecialchars($interest['id'], ENT_QUOTES, 'UTF-8'); ?>">
                    <?php echo htmlspecialchars($interest['name'], ENT_QUOTES, 'UTF-8'); ?>
                </label><br>
            <?php endforeach; ?>
            <br>
            <input type="submit" value="Save Interests">
        </form>
    <?php else: ?>
        <p>No interests available.</p>
    <?php endif; ?>
<?php else: ?>
    <p style="text-align: center;">You have already added your interests.</p>
<?php endif; ?>
```

If the user has not added any interests yet, displays a form with checkboxes for all available interests.
Each interest is safely output with proper escaping.
If no interests exist in the database, informs the user.
If the user already has interests, displays a message instead of the form.

Logout.php page

Logout Page Information

Description:

A logout.php page is designed to securely end a user's session on a website, effectively logging them out. Its primary purpose is to clear all session data associated with the user, such as authentication tokens or user identifiers, to prevent unauthorized access after logout. The logout page helps protect user privacy and maintain the security of the application. Overall, it ensures that once a user chooses to log out, their session is fully closed and their account remains secure.

Key Points about the Page:

Complete Session Destruction: Ensure that all session data is fully cleared using session_unset() and session_destroy() to prevent any residual data from lingering.

Redirect After Logout: Immediately redirect users to a safe page (e.g., login or homepage) after logout to prevent access to protected content via the back button.

Cookie Management: If sessions use cookies, explicitly delete the session cookie by setting its expiration time in the past to fully invalidate the session on the client side.

PHP Code Screenshot

```php
<?php
session_start();
unset($_SESSION['user']);
session_unset();
session_destroy();
header("Location: index.php");
exit;
?>
~
```

PHP Code Screenshot Description

unset($_SESSION['user']);
Removes the specific session variable user that typically stores the logged-in user's ID, effectively logging out the user.

session_unset();
Frees all session variables currently registered, clearing the session data array.

session_destroy();
Destroys the entire session data on the server, invalidating the session ID and making the session unusable.

header("Location: index.php");
Sends an HTTP header to redirect the browser to the homepage (index.php), guiding the user away from protected content.

exit;
Stops further script execution to ensure the redirect happens immediately and no additional code runs.

404 Error Page

404 Error Page Page Information

Description:
A **404 page** is a special webpage displayed when a user tries to access a URL that does not exist on the server. Its purpose is to inform visitors that the requested resource cannot be found, often due to a mistyped URL, a broken link, or a page that has been moved or deleted.

Key Points about the Page

Minimal Information Disclosure: Ensure the 404 page does not reveal server details, file paths, or software versions that could assist attackers in reconnaissance.

Using a custom 404 page improves user experience and allows control over the content, but it must be carefully coded to avoid introducing security vulnerabilities.

PHP Code Screenshot

```php
<?php
// Set the HTTP response status code to 404
http_response_code(404);
?>
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>404 Not Found</title>
        <link rel="stylesheet" href="css/global.css">
</head>
<body>
    <h1>404</h1>
        <div class="container">
            <p>Oops! The page you are looking for cannot be found.</p>
            <p><a href="index.php">Go back to Home Page</a></p>
        </div>
</body>
</html>
```

PHP Code Screenshot explanation

http_response_code(404);
This line explicitly sets the HTTP response status code to **404 Not Found**, informing browsers and search engines that the requested resource does not exist.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>404 Not Found</title>
                <link rel="stylesheet" href="css/global.css">
</head>
<body>
```

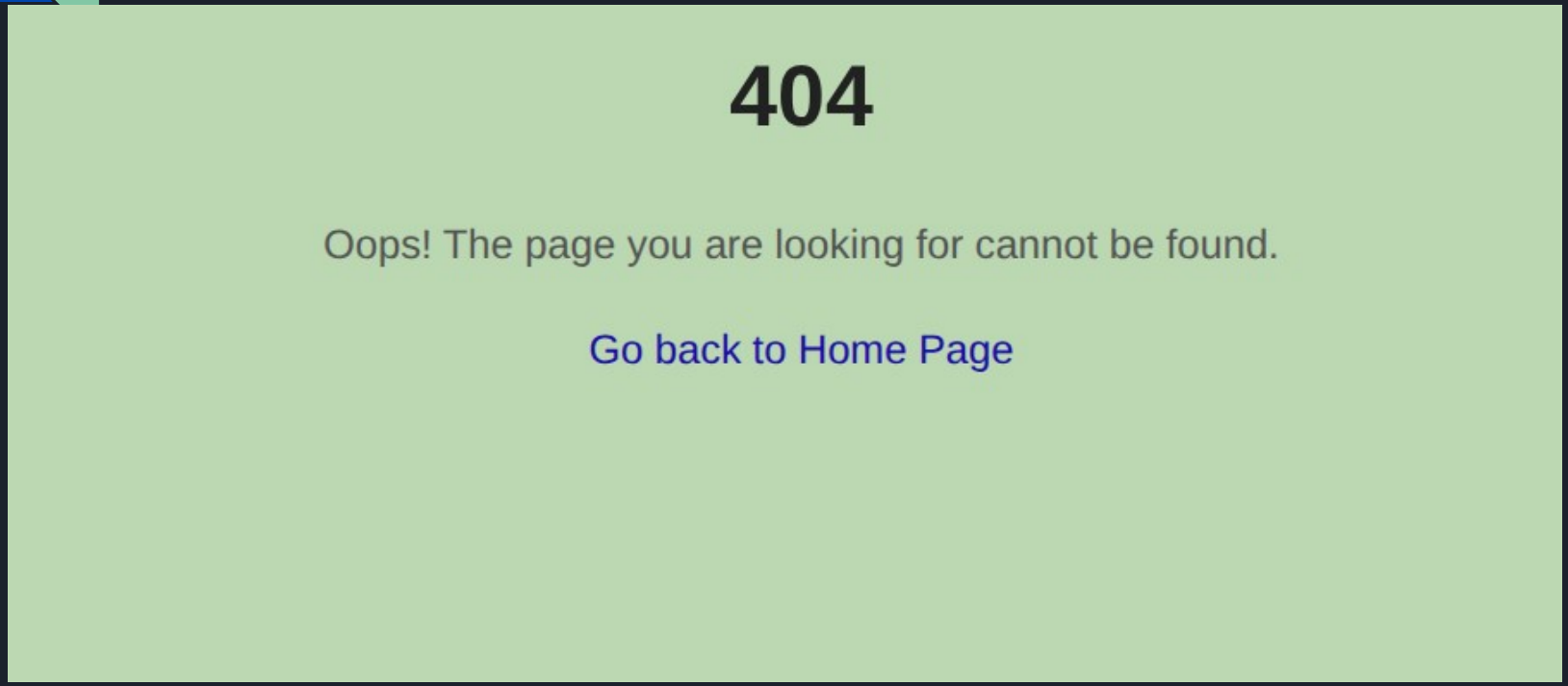
A link to an external CSS stylesheet (css/global.css) for consistent styling.
Declares the HTML5 document type and sets the language attribute to English.

PHP Code Screenshot Description

```
    <h1>404</h1>
                <div class="container">
                    <p>Oops! The page you are looking for cannot be found.</p>
                    <p><a href="index.php">Go back to Home Page</a></p>
                </div>
</body>
</html>
```

Displays a large heading "404" to clearly indicate the error.
Contains a message informing the user that the page cannot be found.
Provides a helpful link to return to the website's homepage (index.php), improving navigation and user experience.

PHP Webpage Screenshot



PHP Webpage Screenshot Description

Clear HTTP 404 Status Code: The page explicitly sends a 404 status to browsers and search engines, ensuring proper recognition of the missing resource.
Navigation Link: A prominent link back to the homepage helps users easily continue browsing the site instead of leaving.
Consistent Styling: The page uses an external CSS file (css/global.css) to maintain visual consistency with the rest of the website.