BARRETT'S PROJECT PORTFOLIO

AUTHOR: BARRETT CAYWOOD

DATE: 4-1-24

LAMP STACK

LAMP STACK Information

Description:

The LAMP stack is a popular open-source web platform used for building and hosting dynamic websites and web applications. It consists of four main components: Linux (the operating system), Apache (the web server), MySQL (the database management system), and PHP (the programming language). Together, these components provide a powerful and flexible environment for developing and deploying web-based projects.

System Specs:

CentOS Linux 7 (Core)

Apache/2.4.6

5.5.68-MariaDB MariaDB Server

PHP 5.4.16 (cli)

INDEX.PHP

Index.php Information

Description:

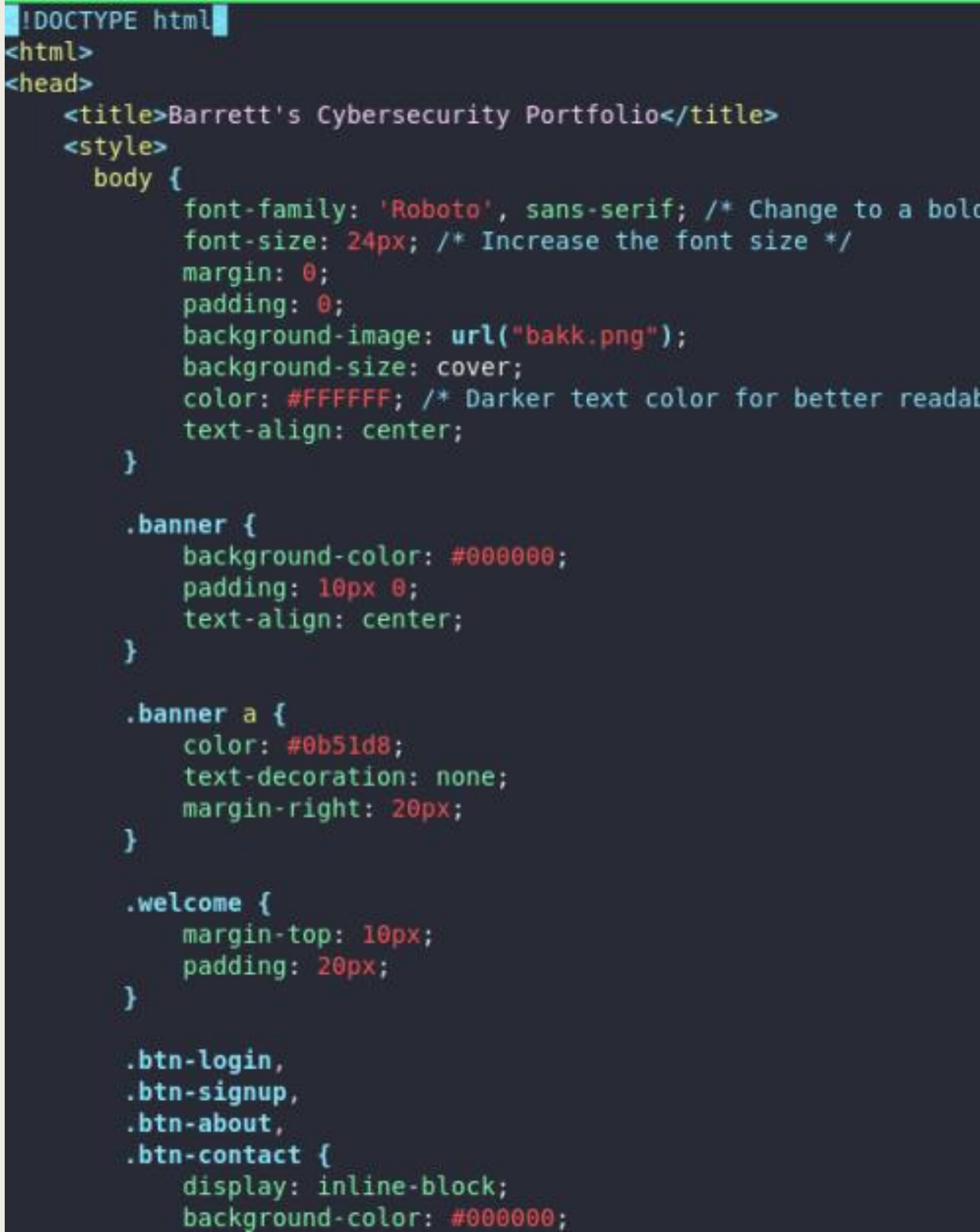
The purpose of my index page is to serve as the initial landing page for visitors, providing them with essential navigation options and a brief overview of the website's content. It welcomes visitors to my portfolio and encourages them to explore the projects and cybersecurity experience showcased on the site. This page pushes the unauthenticated users to login so that they can be redirected to the home page.

Key Points:

- The page is designed with a bold Roboto font, a dark color scheme for better readability, and a background image for aesthetic appeal
- The message encourages visitors to login to explore my projects, indicating the purpose of the website
- The page includes links for logging in, signing up, viewing projects, and accessing the about page, making it easy for visitors to navigate the site

Index.php Code

- body: Contains the content of the page, such as text, the background and other styling options
- .banner: Styles the banner section, which contains links for Login, Sign Up, Projects, and About
- .welcome: Styles the welcome section, which contains the welcome message and a prompt to login
- .btn-login, .btn-signup, etc.: Styles the Login, Sign Up, Projects, and About buttons.



```
<!DOCTYPE html>
<html>
<head>
    <title>Barrett's Cybersecurity Portfolio</title>
    <style>
      body {
            font-family: 'Roboto', sans-serif; /* Change to a bolc
            font-size: 24px; /* Increase the font size */
            margin: 0;
            padding: 0;
            background-image: url("bakk.png");
            background-size: cover;
            color: #FFFFFF; /* Darker text color for better readab
            text-align: center;
        }

        .banner {
            background-color: #000000;
            padding: 10px 0;
            text-align: center;
        }

        .banner a {
            color: #0b51d8;
            text-decoration: none;
            margin-right: 20px;
        }

        .welcome {
            margin-top: 10px;
            padding: 20px;
        }

        .btn-login,
        .btn-signup,
        .btn-about,
        .btn-contact {
            display: inline-block;
            background-color: #000000;
```

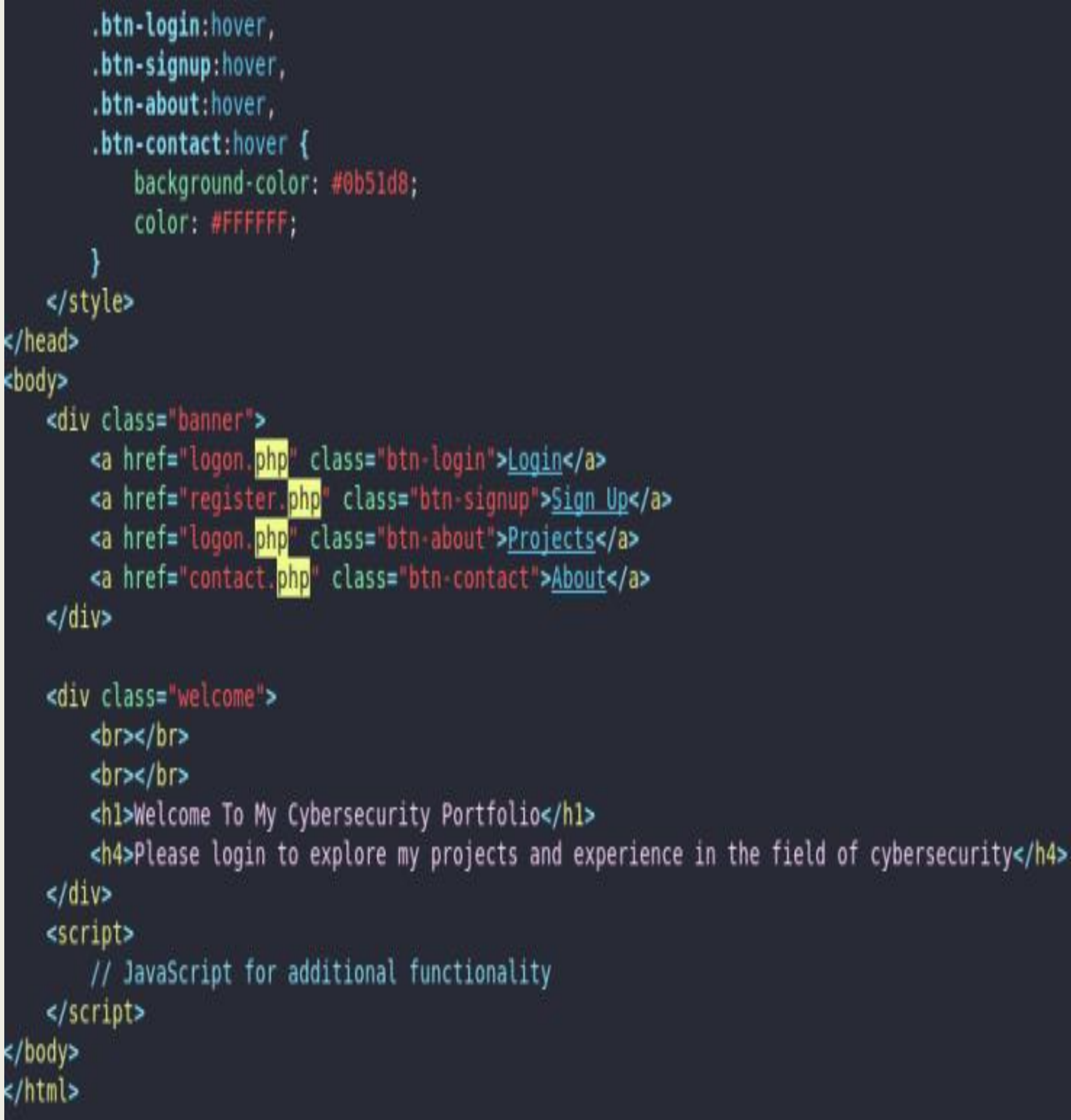
Index.php Code

- .btn-login:hover, .btn-signup:hover, etc.: Styles the buttons to change when hovered over

- <a>: Defines a hyperlink which is used for the corresponding buttons in the banner section

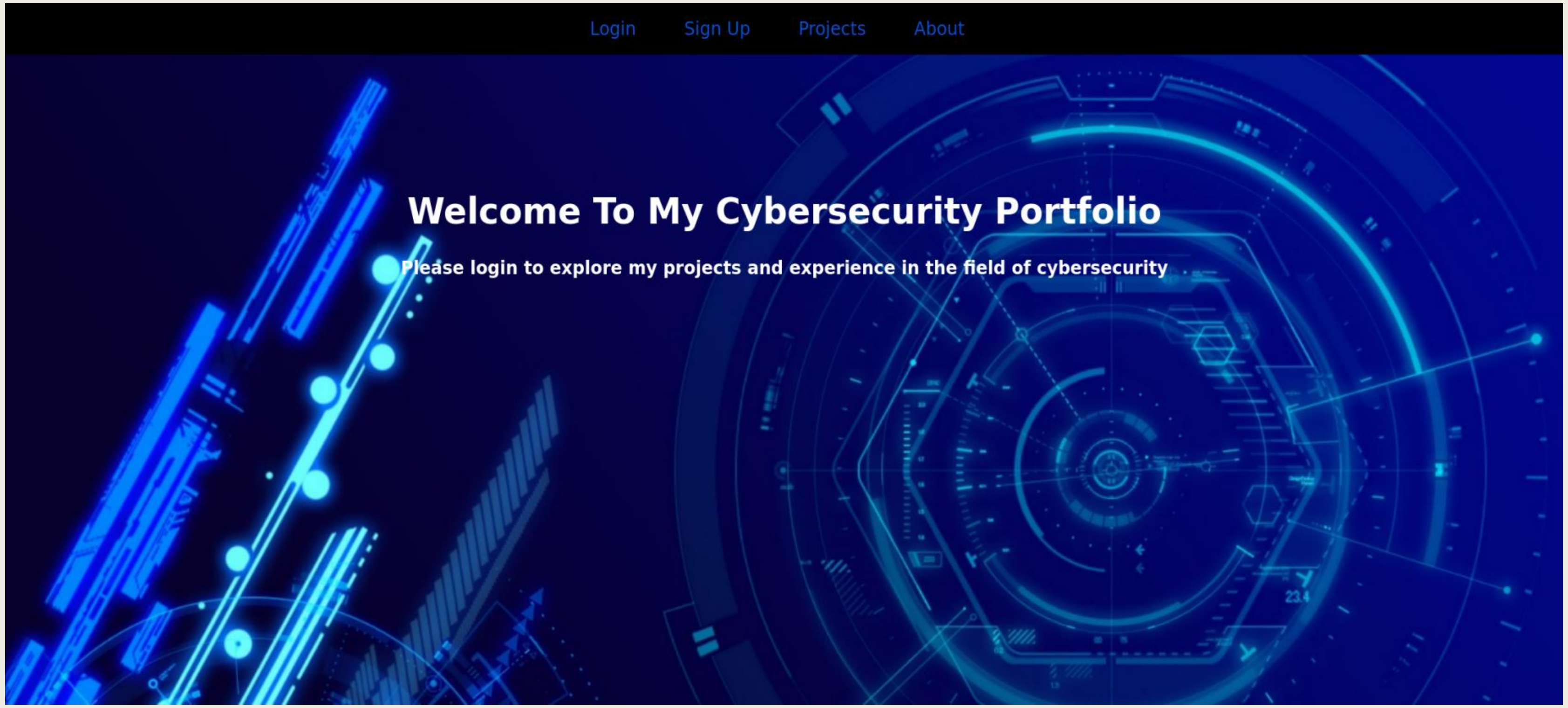
- <h1> <h4>: These make up the body of the webpage which is styled by the welcome class as seen above in the code

- <script>: Here I plan to add more JavaScript to make the website more fluid and dynamic



```
        .btn-login:hover,
        .btn-signup:hover,
        .btn-about:hover,
        .btn-contact:hover {
            background-color: #0b51d8;
            color: #FFFFFF;
        }
    </style>
</head>
<body>
    <div class="banner">
        <a href="logon.php" class="btn-login">Login</a>
        <a href="register.php" class="btn-signup">Sign Up</a>
        <a href="logon.php" class="btn-about">Projects</a>
        <a href="contact.php" class="btn-contact">About</a>
    </div>

    <div class="welcome">
        <br></br>
        <br></br>
        <h1>Welcome To My Cybersecurity Portfolio</h1>
        <h4>Please login to explore my projects and experience in the field of cybersecurity</h4>
    </div>
    <script>
        // JavaScript for additional functionality
    </script>
</body>
</html>
```

Index.php Webpage



Welcome To My Cybersecurity Portfolio

Please login to explore my projects and experience in the field of cybersecurity

Here users can choose to login if they already have an account or sign up if they have yet to make an account. Users can also choose to visit my projects or about page but must login first. Upon logging in they will be redirected to the home page where they can access any of the previous mentioned pages and more.

DATABASE INFORMATION

Database Information

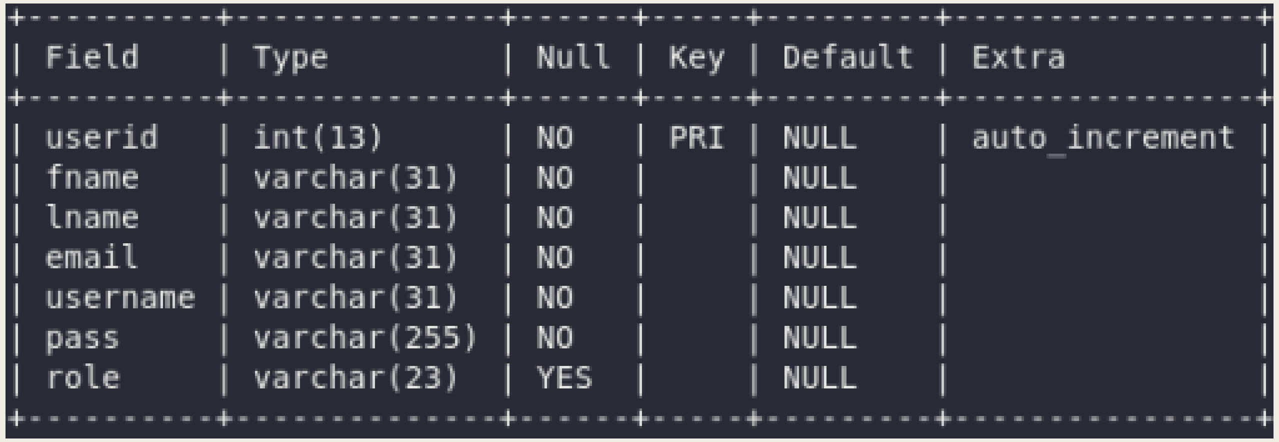
Description:

The database used on my website serves as a crucial component for managing user authentication and storing user-related information. It stores user credentials securely for login purposes, ensuring that only authenticated users can access certain pages. The database also contains user-specific data, such as contact information and roles, which is utilized to personalize the user experience and provide tailored content.

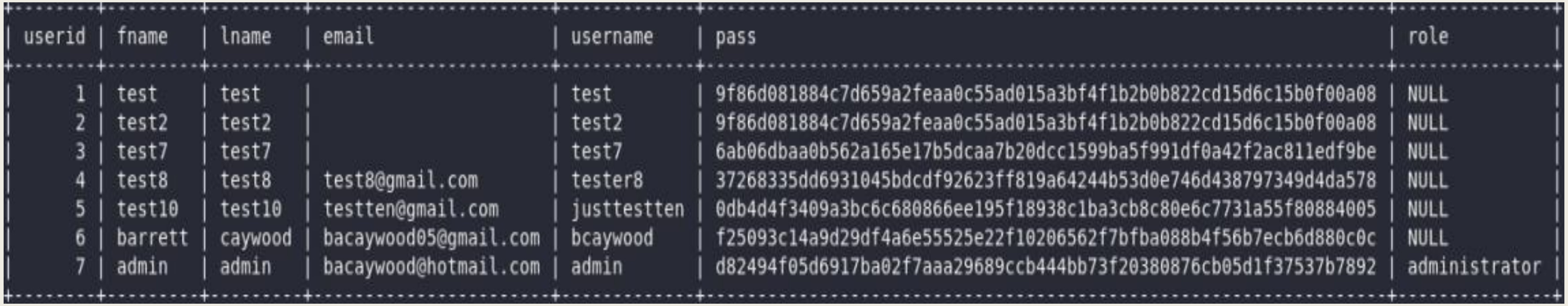
Key Points:

- The database manages user roles (e.g., admin, regular user) to control access to certain parts of the website and enforce security measures
- It is designed to be scalable, capable of handling a growing number of users and data as the website expands
- The database ensures the integrity and security of user data through encryption, access control, and other security measures to protect user privacy

Database



```
+----------+--------------+------+-----+---------+----------------+
| Field    | Type         | Null | Key | Default | Extra          |
+----------+--------------+------+-----+---------+----------------+
| userid   | int(13)      | NO   | PRI | NULL    | auto_increment |
| fname    | varchar(31)  | NO   |     | NULL    |                |
| lname    | varchar(31)  | NO   |     | NULL    |                |
| email    | varchar(31)  | NO   |     | NULL    |                |
| username | varchar(31)  | NO   |     | NULL    |                |
| pass     | varchar(255) | NO   |     | NULL    |                |
| role     | varchar(23)  | YES  |     | NULL    |                |
+----------+--------------+------+-----+---------+----------------+
```



```
+--------+---------+---------+-----------------------+-------------+------------------------------------------------------------------+---------------+
| userid | fname   | lname   | email                 | username    | pass                                                             | role          |
+--------+---------+---------+-----------------------+-------------+------------------------------------------------------------------+---------------+
|      1 | test    | test    |                       | test        | 9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08 | NULL          |
|      2 | test2   | test2   |                       | test2       | 9f86d081884c7d659a2feaa0c55ad015a3bf4f1b2b0b822cd15d6c15b0f00a08 | NULL          |
|      3 | test7   | test7   |                       | test7       | 6ab06dbaa0b562a165e17b5dcaa7b20dcc1599ba5f991df0a42f2ac811edf9be | NULL          |
|      4 | test8   | test8   | test8@gmail.com       | tester8     | 37268335dd6931045bdcdf92623ff819a64244b53d0e746d438797349d4da578 | NULL          |
|      5 | test10  | test10  | testten@gmail.com     | justtestten | 0db4d4f3409a3bc6c680866ee195f18938c1ba3cb8c80e6c7731a55f80884005 | NULL          |
|      6 | barrett | caywood | bacaywood05@gmail.com | bcaywood    | f25093c14a9d29df4a6e55525e22f10206562f7bfba088b4f56b7ecb6d880c0c | NULL          |
|      7 | admin   | admin   | bacaywood@hotmail.com | admin       | d82494f05d6917ba02f7aaa29689ccb444bb73f20380876cb05d1f37537b7892 | administrator |
+--------+---------+---------+-----------------------+-------------+------------------------------------------------------------------+---------------+
```

PHPTEST.PHP

Phptest.php Information

Description:

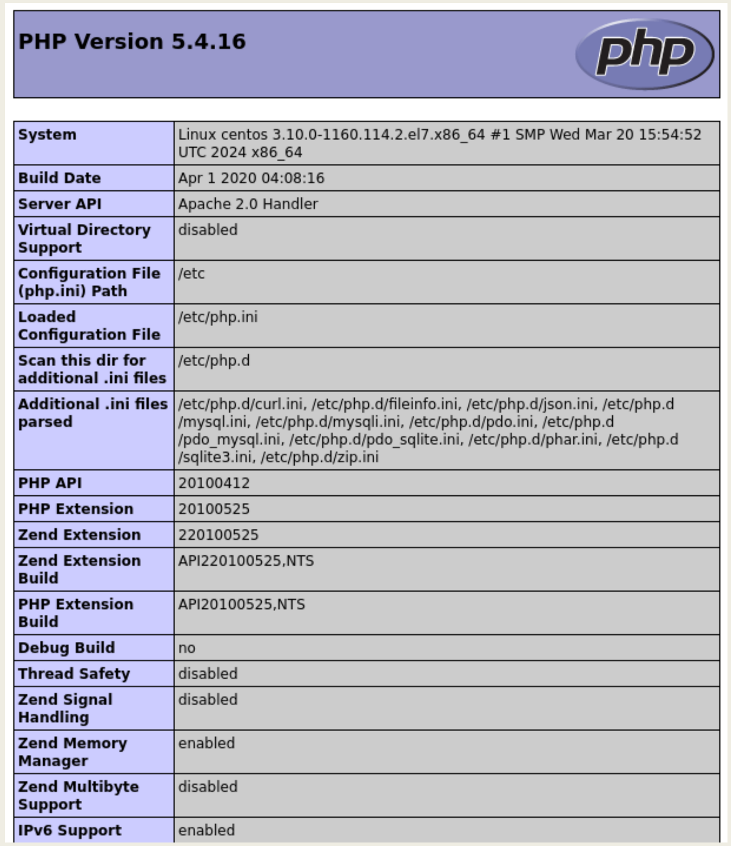
This page contains the phpinfo() function, which outputs a comprehensive overview of the PHP environment on the server where the script is executed. This includes information such as PHP version, configuration settings, modules, and more. It's primarily used for debugging and troubleshooting purposes, providing developers with detailed insights into the PHP configuration and environment.

Key Points:

- This page is not accessible to the public, as it can expose sensitive server information.

Phptest.php Code

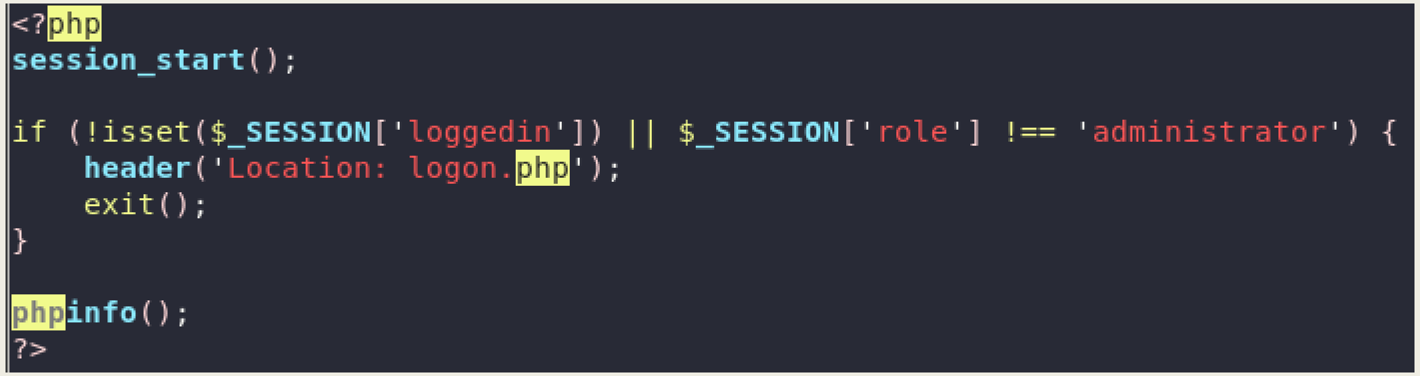
This is one of the many pages included with the phpinfo() function



PHP Version 5.4.16

System	Linux centos 3.10.0-1160.114.2.el7.x86_64 #1 SMP Wed Mar 20 15:54:52 UTC 2024 x86_64
Build Date	Apr 1 2020 04:08:16
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/curl.ini, /etc/php.d/fileinfo.ini, /etc/php.d/json.ini, /etc/php.d/mysql.ini, /etc/php.d/mysqli.ini, /etc/php.d/pdo.ini, /etc/php.d/pdo_mysql.ini, /etc/php.d/pdo_sqlite.ini, /etc/php.d/phar.ini, /etc/php.d/sqlite3.ini, /etc/php.d/zip.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled

- This makes sure that only administrators can view this page
- Anyone else who tries to access this page will be redirected to the login page
- This is what makes the request for the info page



```php
<?php
session_start();

if (!isset($_SESSION['loggedin']) || $_SESSION['role'] !== 'administrator') {
    header('Location: logon.php');
    exit();
}

phpinfo();
?>
```

CONNECT.PHP

Connect.php Information

Description:

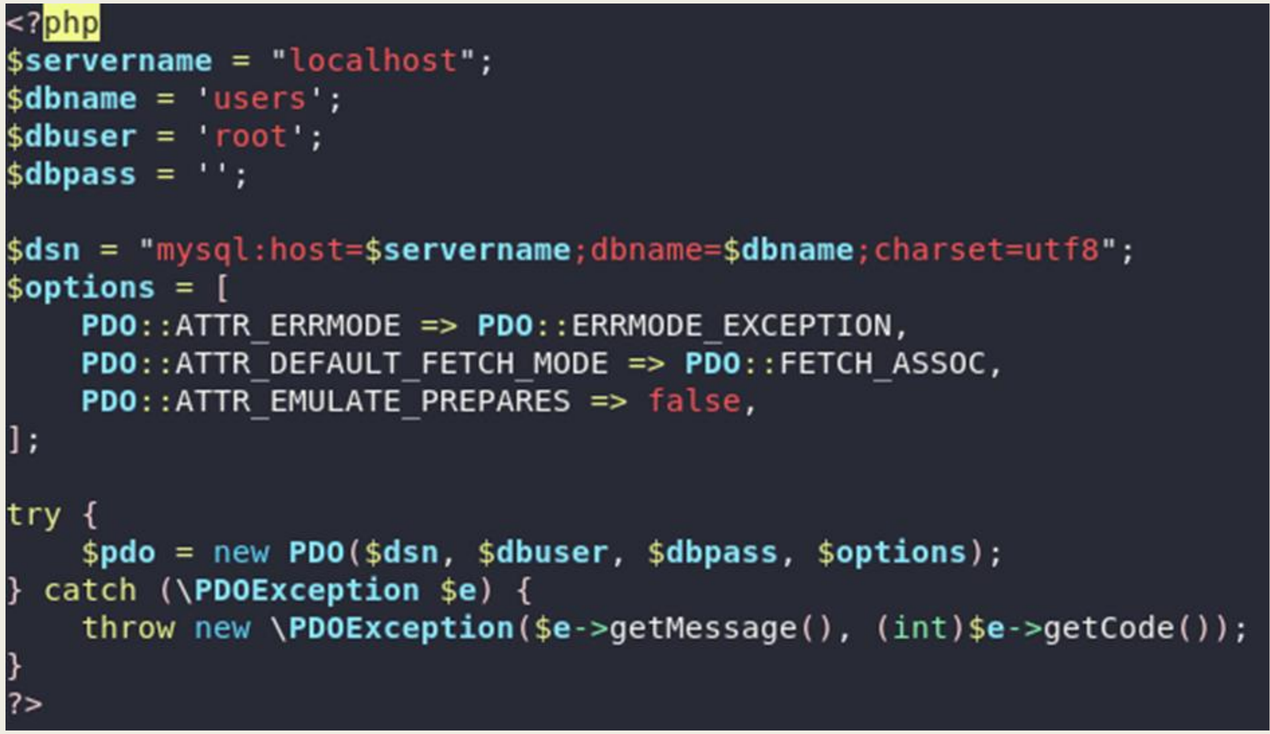
This PHP script establishes a connection to the MySQL database named 'users' on the same server. The purpose of this page is to connect to the database to perform various operations such as retrieving user information, updating user details, or inserting new user data.

Key Points:

- This webpage uses PHP Data Objects to connect to the MySQL database, which provides a secure and efficient way to interact with databases in PHP
- The $options array is used to set attributes for error handling, default fetch mode, and emulation of prepared statements
- The try-catch block is used to handle any exceptions that may occur during the database connection process, ensuring that errors are caught and properly managed

Connect.php Code

- These variables store the server's name, database name, database username, and database password
- $dsn: This variable stores the Data Source Name for the database connection
- $options: This variable is an array that configures the PDO connection.
- try: This block of code tries to create a new PDO instance with the specified options.



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

$dsn = "mysql:host=$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());
}
?>
```

REGISTER.PHP

(Account Creation Web Page)

Register.php Information

Description:

This PHP script creates a user registration form and handles the insertion of new user data into a MySQL database. It includes client-side validation for the form fields and redirects the user to the login page upon successful registration.

Key Points:

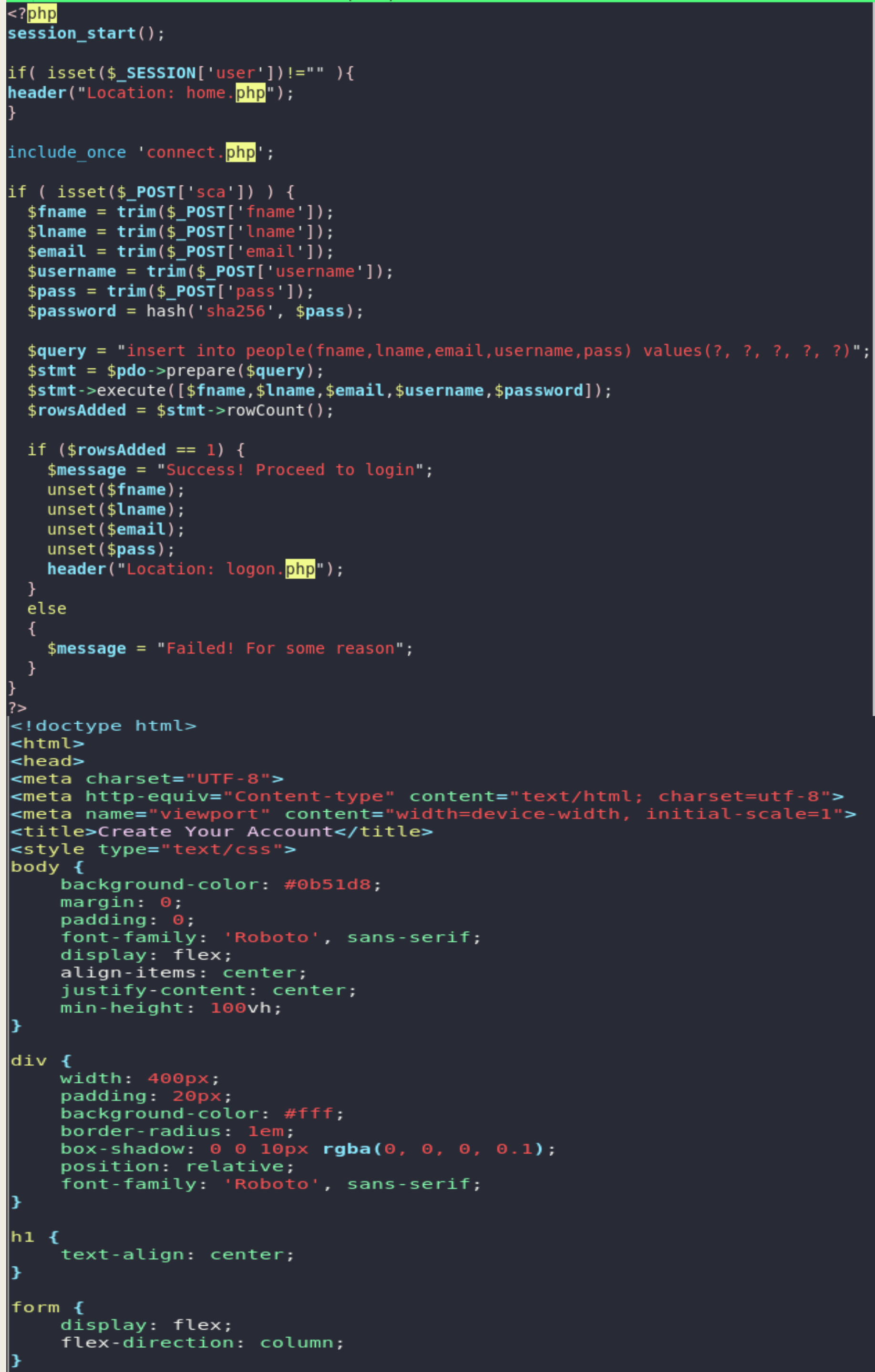
- Uses PHP's PDO extension to connect to a MySQL database and insert user data securely
- Implements client-side validation using JavaScript to ensure that the form is filled out correctly before submission
- Utilizes session management to prevent logged-in users from accessing the registration page
- Provides feedback to the user about the success or failure of the registration process

Register.php Code

- session_start();: Starts a new session or resumes an existing session

- Then it checks if the 'user' session variable is set and redirects to the home.php page if so

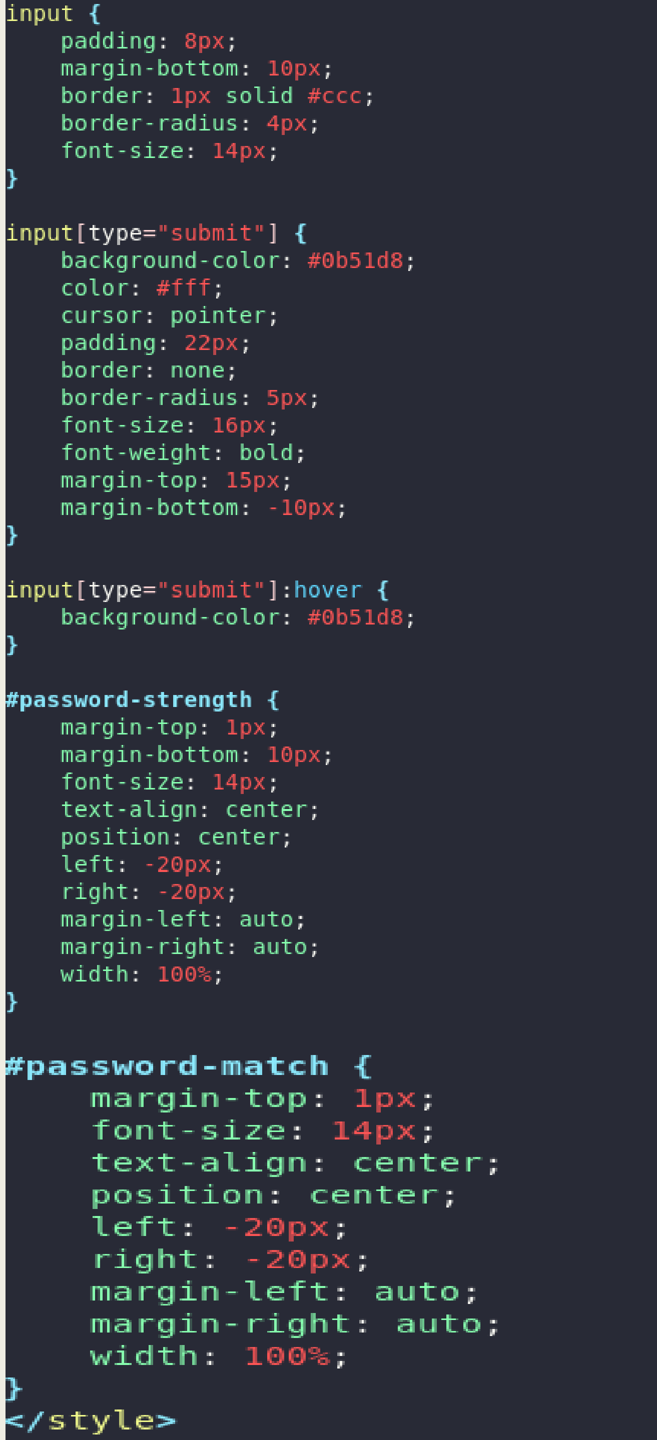
- Include_once : Includes the 'connect.php' file, which contains the PDO database connection



```php
<?php
session_start();

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

include_once 'connect.php';

if ( isset($_POST['sca']) ) {
  $fname = trim($_POST['fname']);
  $lname = trim($_POST['lname']);
  $email = trim($_POST['email']);
  $username = trim($_POST['username']);
  $pass = trim($_POST['pass']);
  $password = hash('sha256', $pass);

  $query = "insert into people(fname,lname,email,username,pass) values(?, ?, ?, ?, ?)";
  $stmt = $pdo->prepare($query);
  $stmt->execute([$fname,$lname,$email,$username,$password]);
  $rowsAdded = $stmt->rowCount();

  if ($rowsAdded == 1) {
    $message = "Success! Proceed to login";
    unset($fname);
    unset($lname);
    unset($email);
    unset($pass);
    header("Location: logon.php");
  }
  else
  {
    $message = "Failed! For some reason";
  }
}
?>
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<meta http-equiv="Content-type" content="text/html; charset=utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Create Your Account</title>
<style type="text/css">
body {
    background-color: #0b51d8;
    margin: 0;
    padding: 0;
    font-family: 'Roboto', sans-serif;
    display: flex;
    align-items: center;
    justify-content: center;
    min-height: 100vh;
}

div {
    width: 400px;
    padding: 20px;
    background-color: #fff;
    border-radius: 1em;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    position: relative;
    font-family: 'Roboto', sans-serif;
}

h1 {
    text-align: center;
}

form {
    display: flex;
    flex-direction: column;
}
```

Register.php Code

- This provides the layout and styling for the submit button
- Here it styles the password strength meter
- This is the styling for the prompt that tells the user if the password they entered match



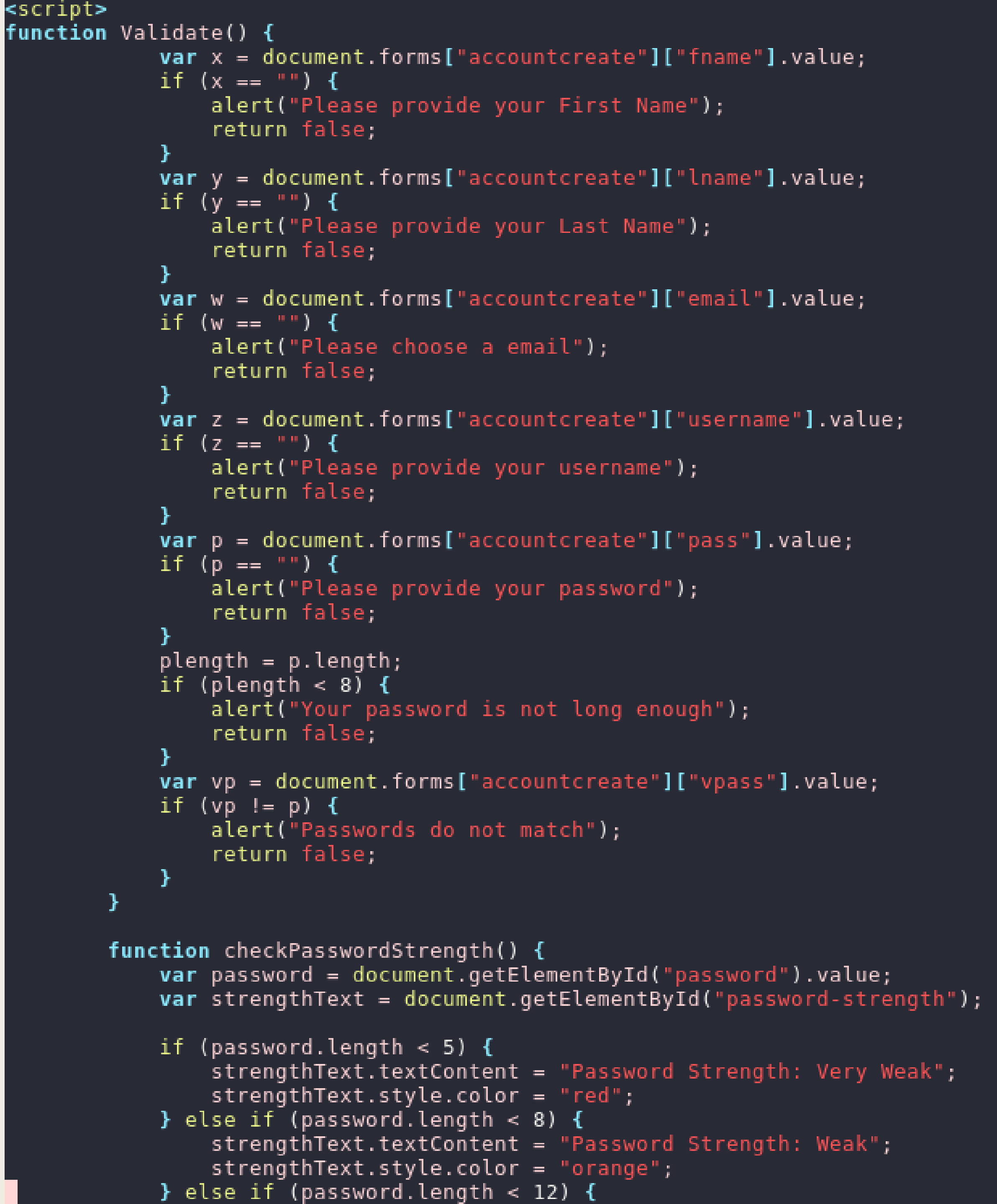
```
input {
    padding: 8px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
    font-size: 14px;
}

input[type="submit"] {
    background-color: #0b51d8;
    color: #fff;
    cursor: pointer;
    padding: 22px;
    border: none;
    border-radius: 5px;
    font-size: 16px;
    font-weight: bold;
    margin-top: 15px;
    margin-bottom: -10px;
}

input[type="submit"]:hover {
    background-color: #0b51d8;
}

#password-strength {
    margin-top: 1px;
    margin-bottom: 10px;
    font-size: 14px;
    text-align: center;
    position: center;
    left: -20px;
    right: -20px;
    margin-left: auto;
    margin-right: auto;
    width: 100%;
}

#password-match {
    margin-top: 1px;
    font-size: 14px;
    text-align: center;
    position: center;
    left: -20px;
    right: -20px;
    margin-left: auto;
    margin-right: auto;
    width: 100%;
}
</style>
```

Register.php Code

- Here the input fields for the account creation are made

- Unlike the other fields the password field requires a minimum length for security purposes

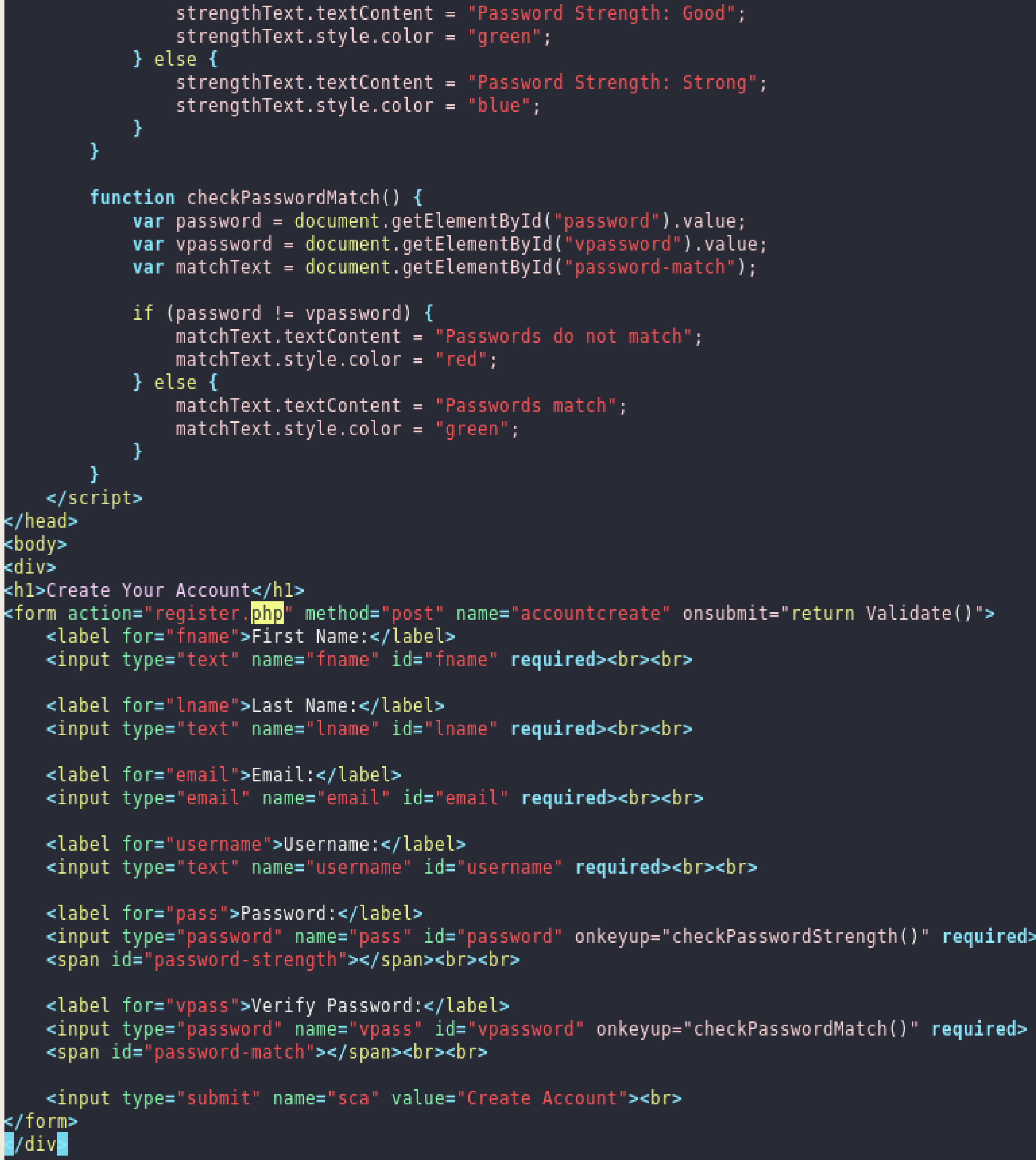
- This function checks the passwords length and displays the strength level to the user



```
<script>
function Validate() {
            var x = document.forms["accountcreate"]["fname"].value;
            if (x == "") {
                alert("Please provide your First Name");
                return false;
            }
            var y = document.forms["accountcreate"]["lname"].value;
            if (y == "") {
                alert("Please provide your Last Name");
                return false;
            }
            var w = document.forms["accountcreate"]["email"].value;
            if (w == "") {
                alert("Please choose a email");
                return false;
            }
            var z = document.forms["accountcreate"]["username"].value;
            if (z == "") {
                alert("Please provide your username");
                return false;
            }
            var p = document.forms["accountcreate"]["pass"].value;
            if (p == "") {
                alert("Please provide your password");
                return false;
            }
            plength = p.length;
            if (plength < 8) {
                alert("Your password is not long enough");
                return false;
            }
            var vp = document.forms["accountcreate"]["vpass"].value;
            if (vp != p) {
                alert("Passwords do not match");
                return false;
            }
        }

        function checkPasswordStrength() {
            var password = document.getElementById("password").value;
            var strengthText = document.getElementById("password-strength");

            if (password.length < 5) {
                strengthText.textContent = "Password Strength: Very Weak";
                strengthText.style.color = "red";
            } else if (password.length < 8) {
                strengthText.textContent = "Password Strength: Weak";
                strengthText.style.color = "orange";
            } else if (password.length < 12) {
```

Register.php Code

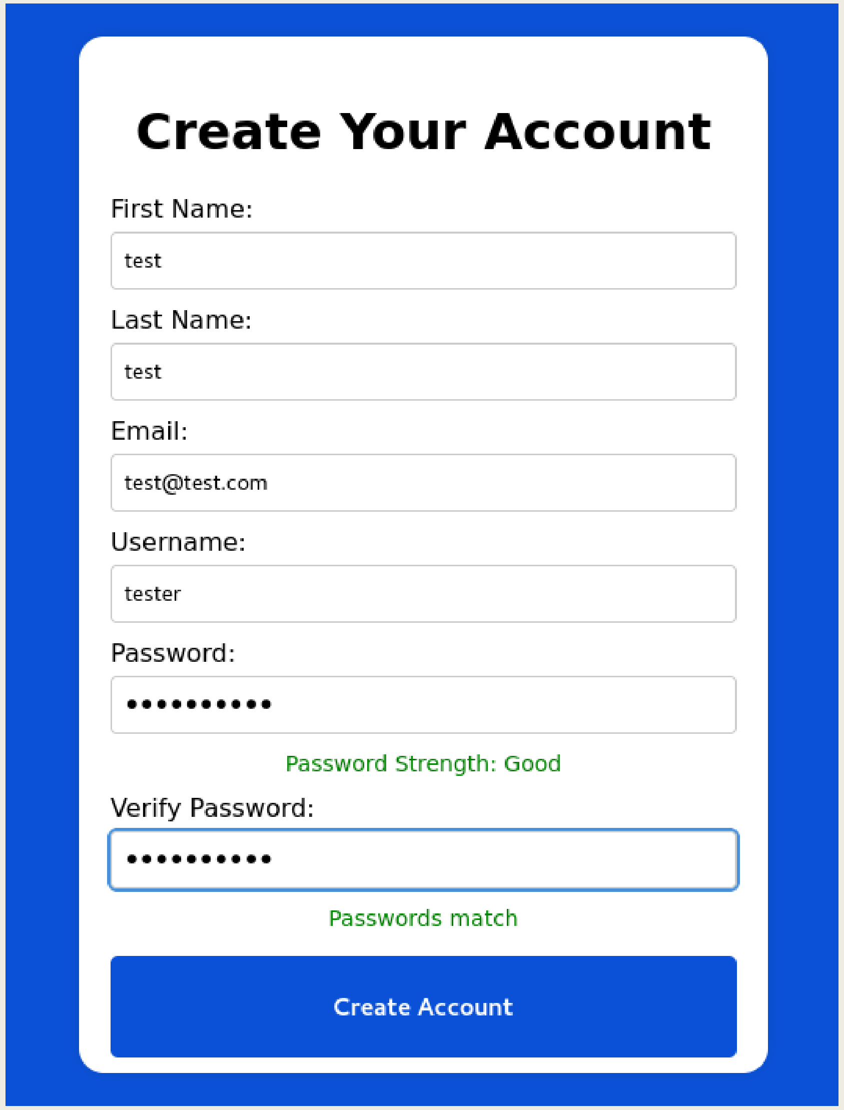
- This function assures that passwords entered by the user match before insertion to the database
- Here all the input fields and functions covered are put into there place



```
                strengthText.textContent = "Password Strength: Good";
                strengthText.style.color = "green";
            } else {
                strengthText.textContent = "Password Strength: Strong";
                strengthText.style.color = "blue";
            }
        }

        function checkPasswordMatch() {
            var password = document.getElementById("password").value;
            var vpassword = document.getElementById("vpassword").value;
            var matchText = document.getElementById("password-match");

            if (password != vpassword) {
                matchText.textContent = "Passwords do not match";
                matchText.style.color = "red";
            } else {
                matchText.textContent = "Passwords match";
                matchText.style.color = "green";
            }
        }
    </script>
</head>
<body>
<div>
<h1>Create Your Account</h1>
<form action="register.php" method="post" name="accountcreate" onsubmit="return Validate()">
    <label for="fname">First Name:</label>
    <input type="text" name="fname" id="fname" required><br><br>

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

    <label for="email">Email:</label>
    <input type="email" name="email" id="email" required><br><br>

    <label for="username">Username:</label>
    <input type="text" name="username" id="username" required><br><br>

    <label for="pass">Password:</label>
    <input type="password" name="pass" id="password" onkeyup="checkPasswordStrength()" required>
    <span id="password-strength"></span><br><br>

    <label for="vpass">Verify Password:</label>
    <input type="password" name="vpass" id="vpassword" onkeyup="checkPasswordMatch()" required>
    <span id="password-match"></span><br><br>

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

Register.php Webpage

Here new users can create an account. It includes fields for the user's first name, last name, email, username, password, and password verification. Client-side JavaScript functions are used for form validation, including checking the password strength and verifying that the passwords match. If registration is successful, the user is redirected to the login page.



Create Your Account

First Name: test

Last Name: test

Email: test@test.com

Username: tester

Password: •••••••••••

Password Strength: Good

Verify Password: •••••••••••

Passwords match

LOGON.PHP

(Login Webpage)

Logon.php Information

Description:

Here users can log into their account so that they can view my projects and other pages . It includes fields for entering a username and password, along with a submit button to log in. The form is validated using JavaScript to ensure both fields are filled out before submission. If the login is successful, the user is redirected to the home.php page. Otherwise, an error message will be displayed.

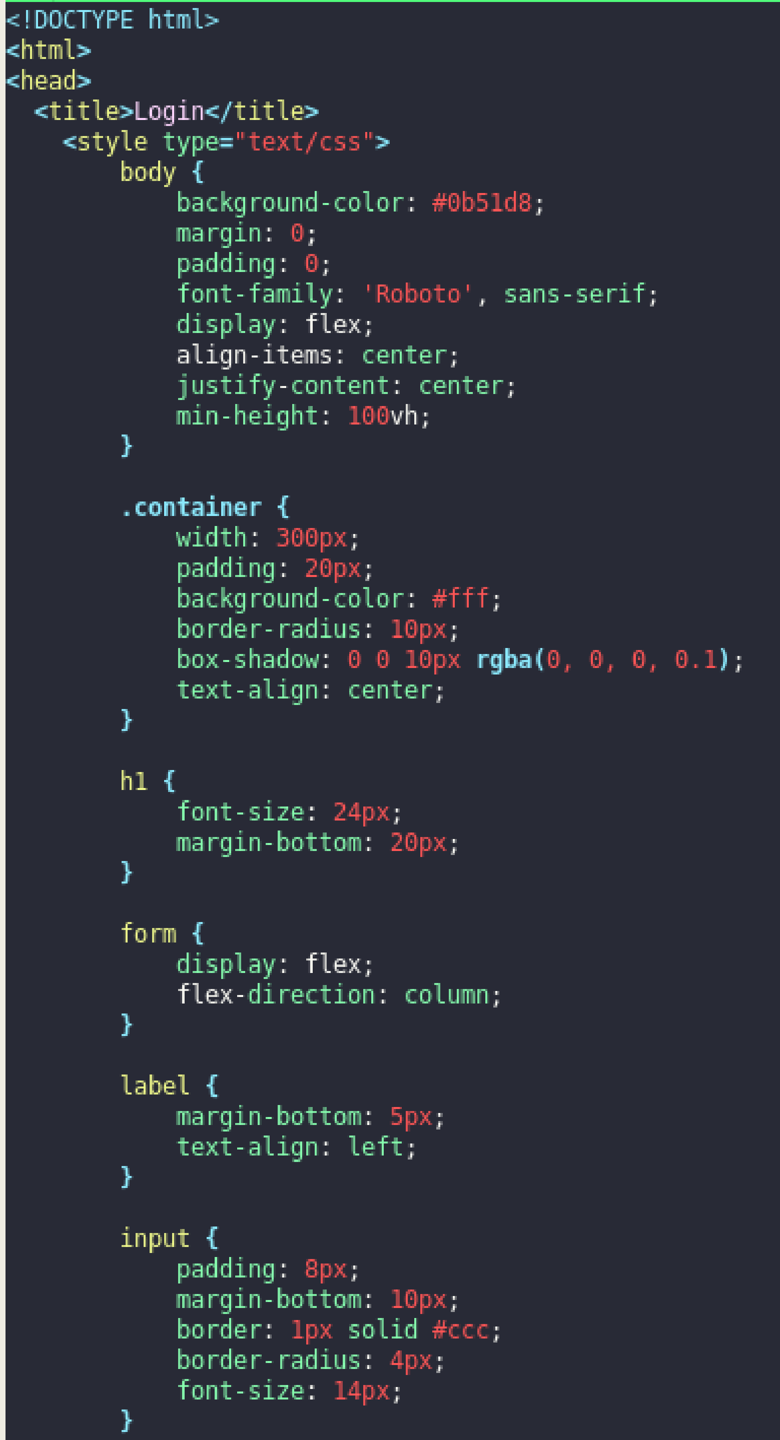
Key Points:

- Passwords are hashed using the SHA-256 algorithm before being stored
- Provides a link to create a new account if the user doesn't have one
- Displays an error message if the login is unsuccessful

Logon.php Code

- This sets the blue background and other styling choices for the webpage

- This defines the styling for the body, container, form, labels, inputs, and error message elements box shadow, and text alignment.

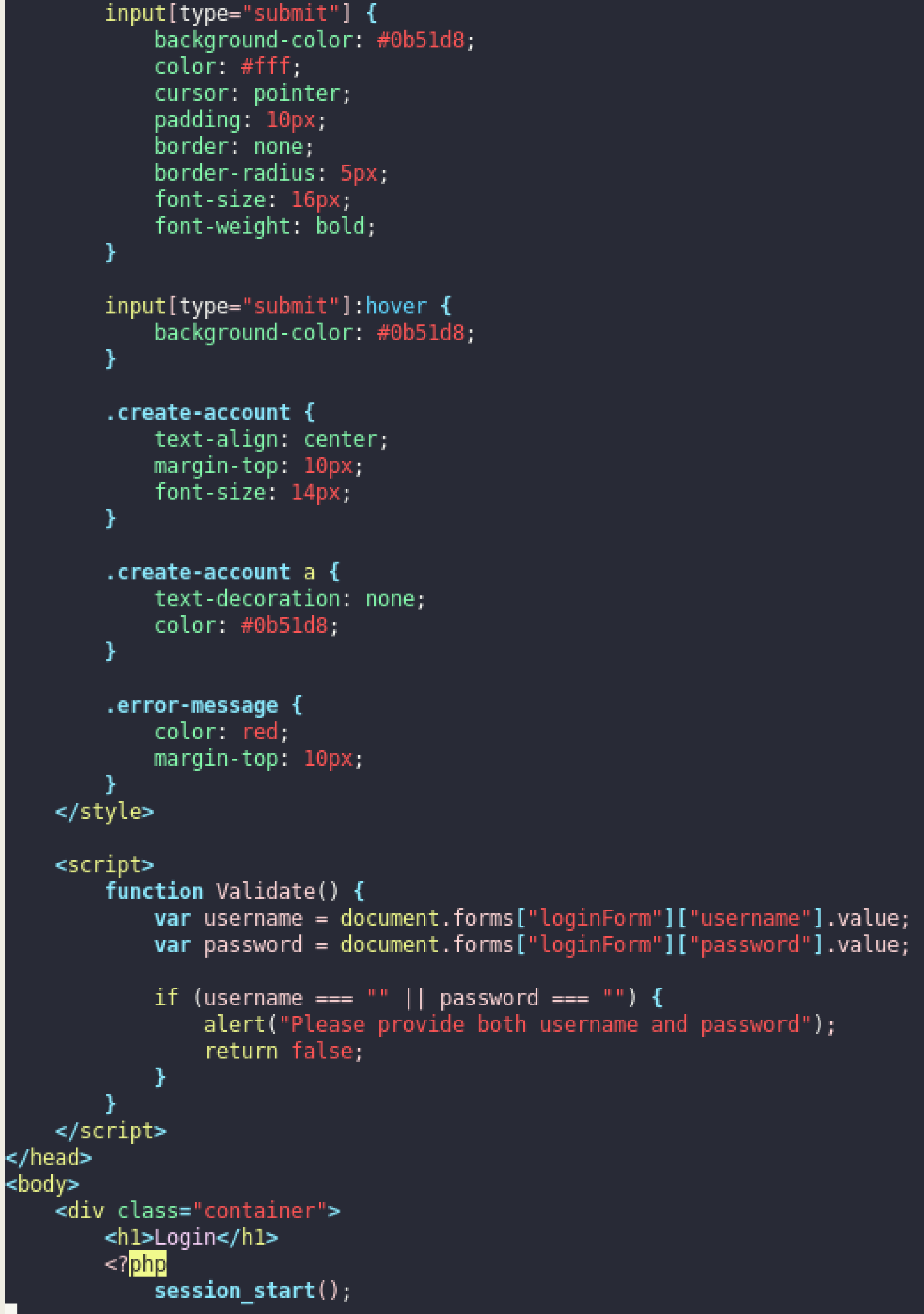


```
<!DOCTYPE html>
<html>
<head>
  <title>Login</title>
    <style type="text/css">
        body {
            background-color: #0b51d8;
            margin: 0;
            padding: 0;
            font-family: 'Roboto', sans-serif;
            display: flex;
            align-items: center;
            justify-content: center;
            min-height: 100vh;
        }

        .container {
            width: 300px;
            padding: 20px;
            background-color: #fff;
            border-radius: 10px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            text-align: center;
        }

        h1 {
            font-size: 24px;
            margin-bottom: 20px;
        }

        form {
            display: flex;
            flex-direction: column;
        }

        label {
            margin-bottom: 5px;
            text-align: left;
        }

        input {
            padding: 8px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 4px;
            font-size: 14px;
        }
```

Logon.php Code

- This styles the submission button and the option to create and account instead

- This function validates that user has provided both the username and password

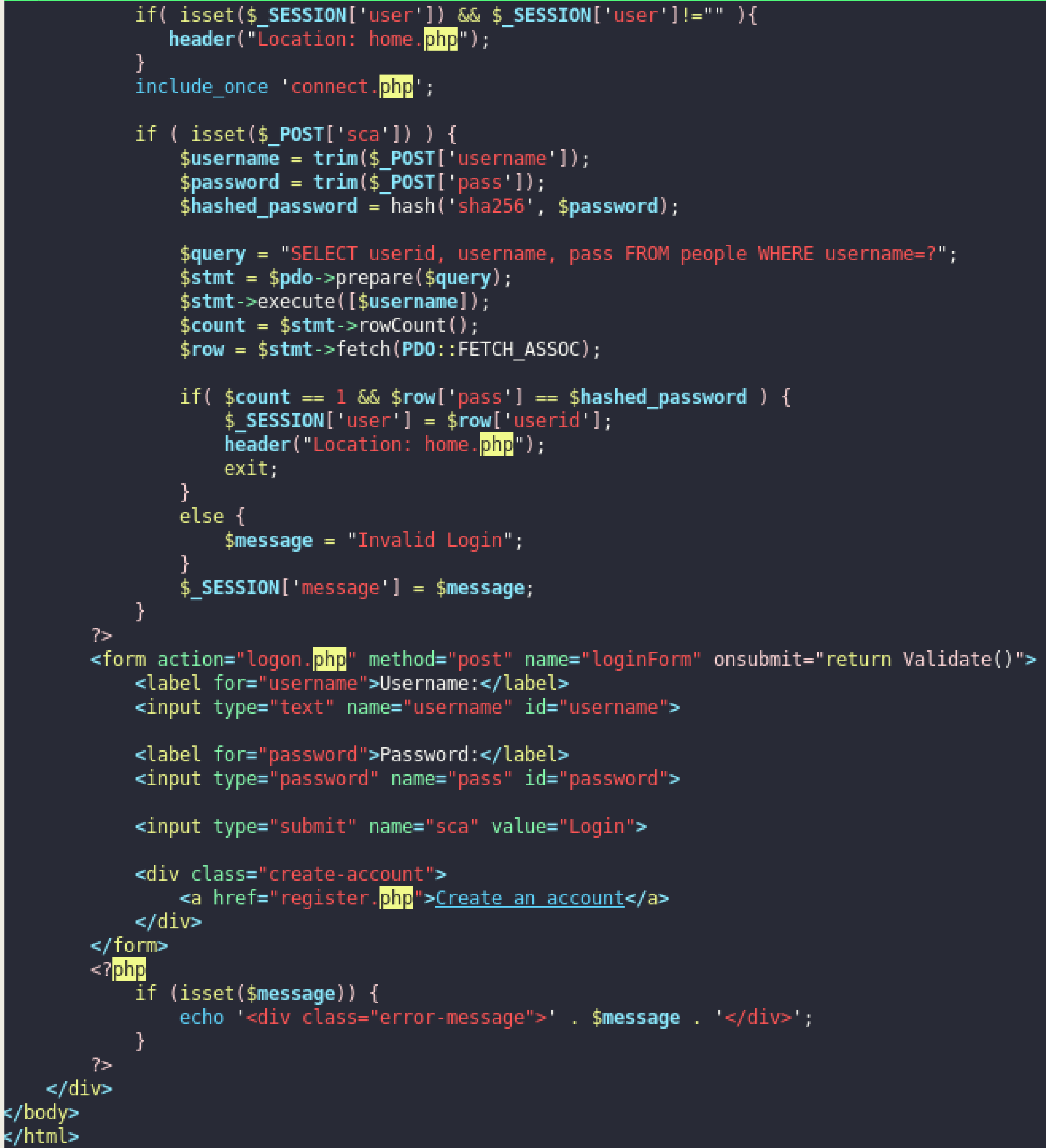


```
        input[type="submit"] {
            background-color: #0b51d8;
            color: #fff;
            cursor: pointer;
            padding: 10px;
            border: none;
            border-radius: 5px;
            font-size: 16px;
            font-weight: bold;
        }

        input[type="submit"]:hover {
            background-color: #0b51d8;
        }

        .create-account {
            text-align: center;
            margin-top: 10px;
            font-size: 14px;
        }

        .create-account a {
            text-decoration: none;
            color: #0b51d8;
        }

        .error-message {
            color: red;
            margin-top: 10px;
        }
    </style>

    <script>
        function Validate() {
            var username = document.forms["loginForm"]["username"].value;
            var password = document.forms["loginForm"]["password"].value;

            if (username === "" || password === "") {
                alert("Please provide both username and password");
                return false;
            }
        }
    </script>
</head>
<body>
    <div class="container">
        <h1>Login</h1>
        <?php
            session_start();
```

Logon.php Code

- Here the login function is connected to the database to make the corresponding changes

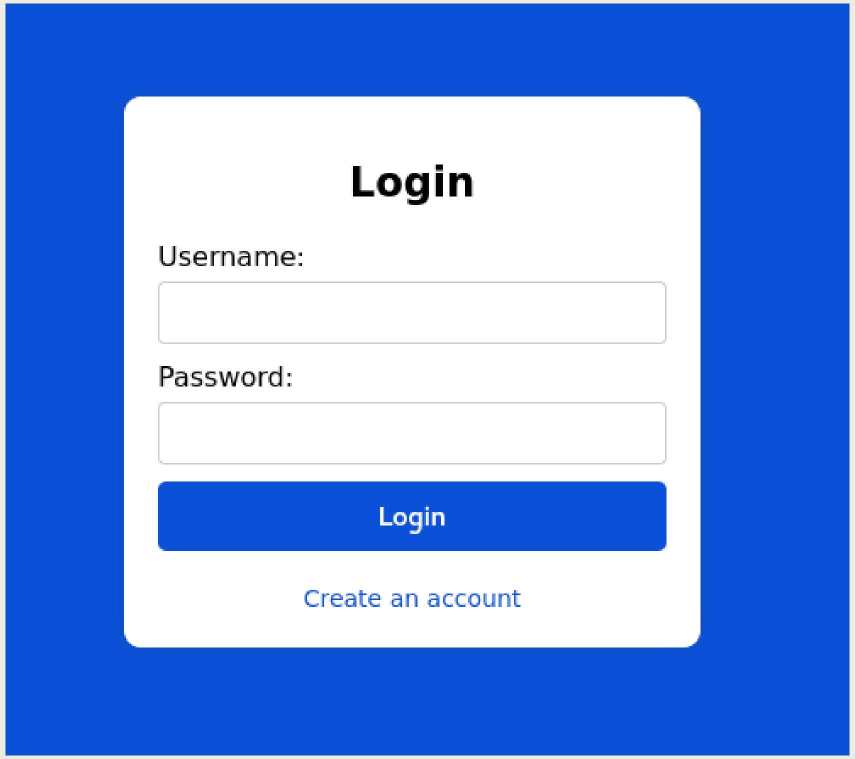
- If the credentials are found in the database, the user is redirected to home.php



```
        if( isset($_SESSION['user']) && $_SESSION['user']!="" ){
          header("Location: home.php");
        }
        include_once 'connect.php';

        if ( isset($_POST['sca']) ) {
            $username = trim($_POST['username']);
            $password = trim($_POST['pass']);
            $hashed_password = hash('sha256', $password);

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

            if( $count == 1 && $row['pass'] == $hashed_password ) {
                $_SESSION['user'] = $row['userid'];
                header("Location: home.php");
                exit;
            }
            else {
                $message = "Invalid Login";
            }
            $_SESSION['message'] = $message;
        }
    ?>
    <form action="logon.php" method="post" name="loginForm" onsubmit="return Validate()">
        <label for="username">Username:</label>
        <input type="text" name="username" id="username">

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

        <input type="submit" name="sca" value="Login">

        <div class="create-account">
            <a href="register.php">Create an account</a>
        </div>
    </form>
    <?php
        if (isset($message)) {
            echo '<div class="error-message">' . $message . '</div>';
        }
    ?>
  </div>
</body>
</html>
```

Logon.php

The webpage features a login form where it employs client-side form validation with JavaScript to ensure both the username and password fields are completed. User authentication is managed through PHP sessions, with user credentials retrieved from the MySQL database. Passwords are hashed using the SHA-256 algorithm before storage for enhanced security. Upon successful login, users are redirected to the home.php page. If the login attempt fails, an error message is displayed. Additionally, a link is provided for users to create a new account if they do not have one.



HOME.PHP

Home.php Information

Description:

This is the actual project portfolio page that includes a header with a welcome message, title, and the logout, about me and settings buttons. The page also contains a container with buttons linking to various projects.

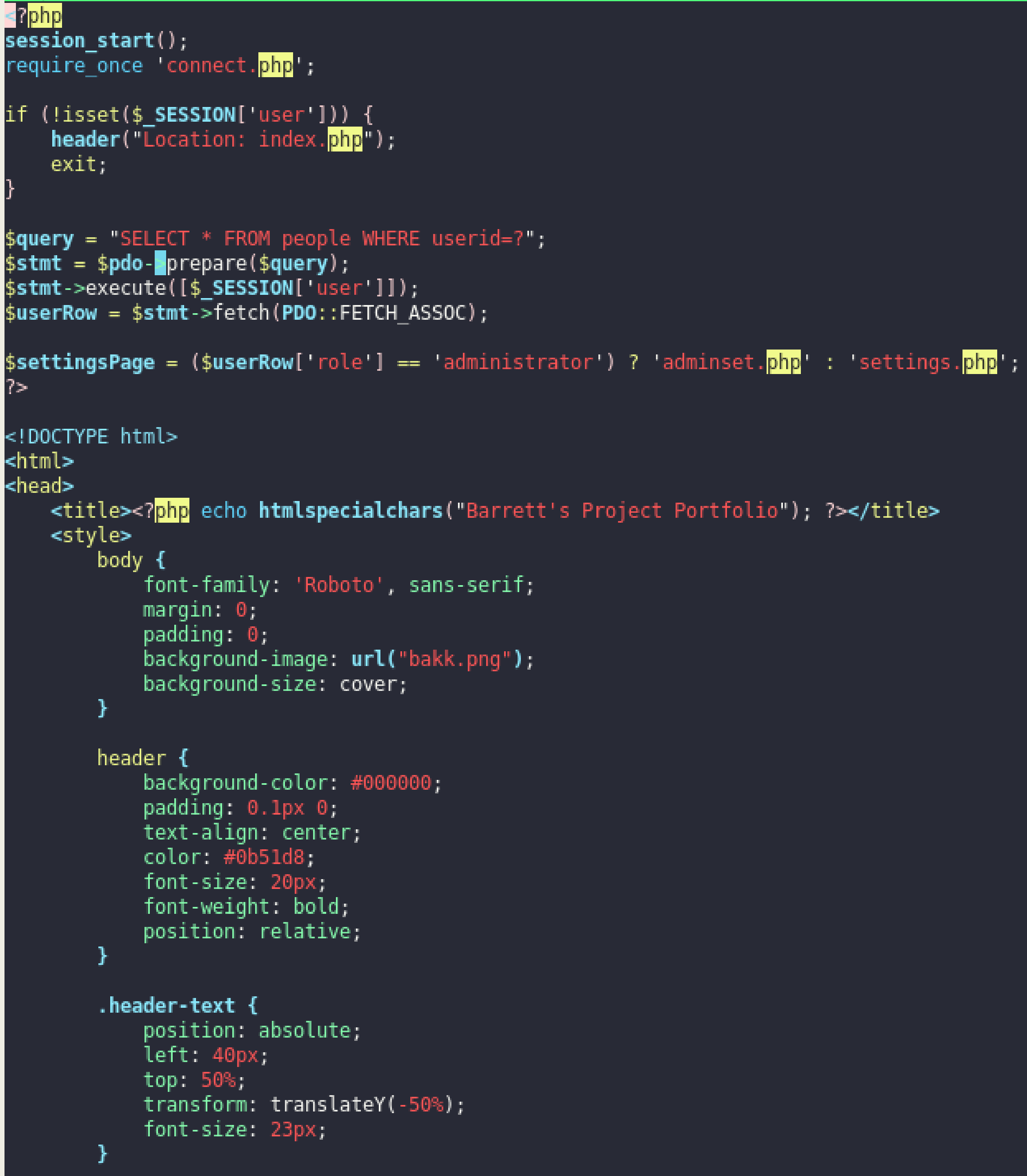
Key Points:

- The page starts a session to manage user authentication. If a user is not logged in, they are redirected to the index.php page.
- The page interacts with the database to fetch user information based on the session user ID. It uses this information to determine the user's role and display the appropriate links in the subheader.
- The subheader includes links for logging out, accessing the contact page (About Me), and accessing settings (either adminset.php for administrators or setting.php for other users)

s if t

Home.php Code

- First the session is started and connected to the database
- Then it checks if the user is not logged in and redirects to the index.php page if not.
- Here it starts to define the styles for the elements of the page

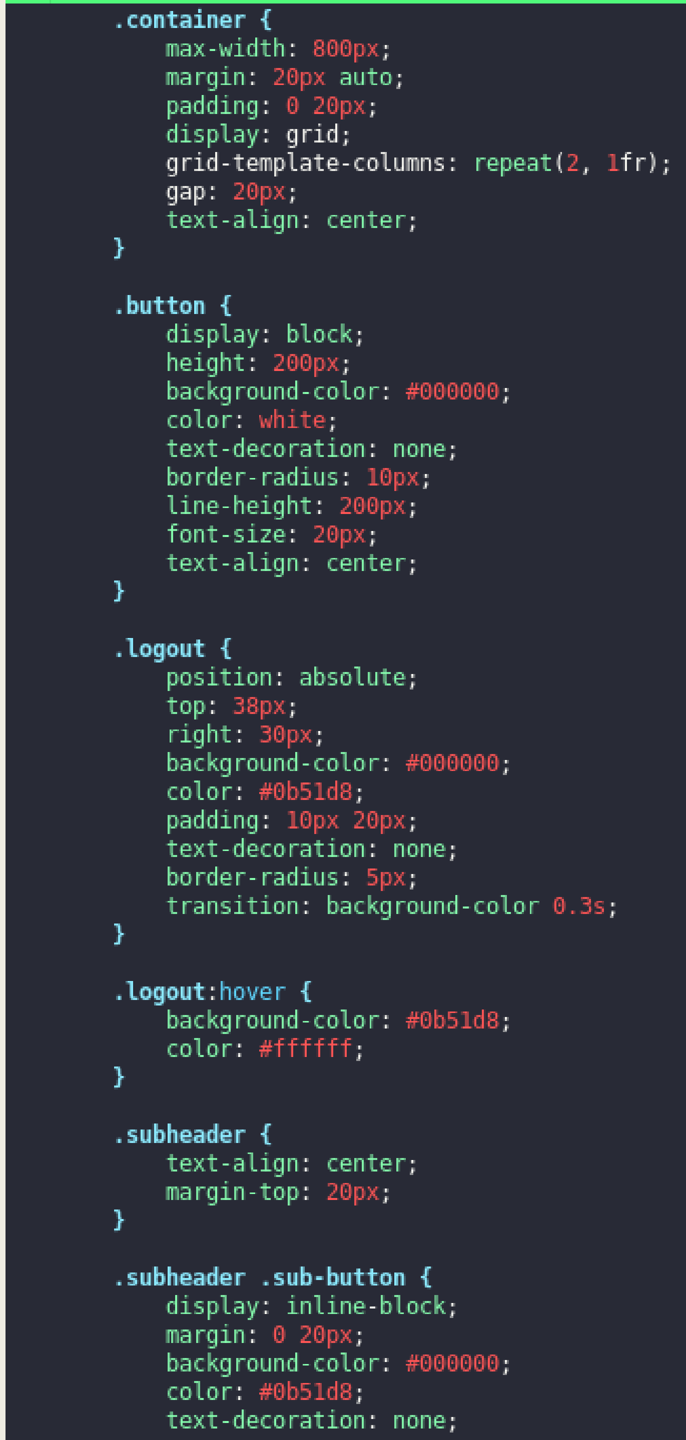


```
<?php
session_start();
require_once 'connect.php';

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

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

$settingsPage = ($userRow['role'] == 'administrator') ? 'adminset.php' : 'settings.php';
?>

<!DOCTYPE html>
<html>
<head>
    <title><?php echo htmlspecialchars("Barrett's Project Portfolio"); ?></title>
    <style>
        body {
            font-family: 'Roboto', sans-serif;
            margin: 0;
            padding: 0;
            background-image: url("bakk.png");
            background-size: cover;
        }

        header {
            background-color: #000000;
            padding: 0.1px 0;
            text-align: center;
            color: #0b51d8;
            font-size: 20px;
            font-weight: bold;
            position: relative;
        }

        .header-text {
            position: absolute;
            left: 40px;
            top: 50%;
            transform: translateY(-50%);
            font-size: 23px;
        }
```

Home.php Code

- Here the styles for the logout and about buttons are defined

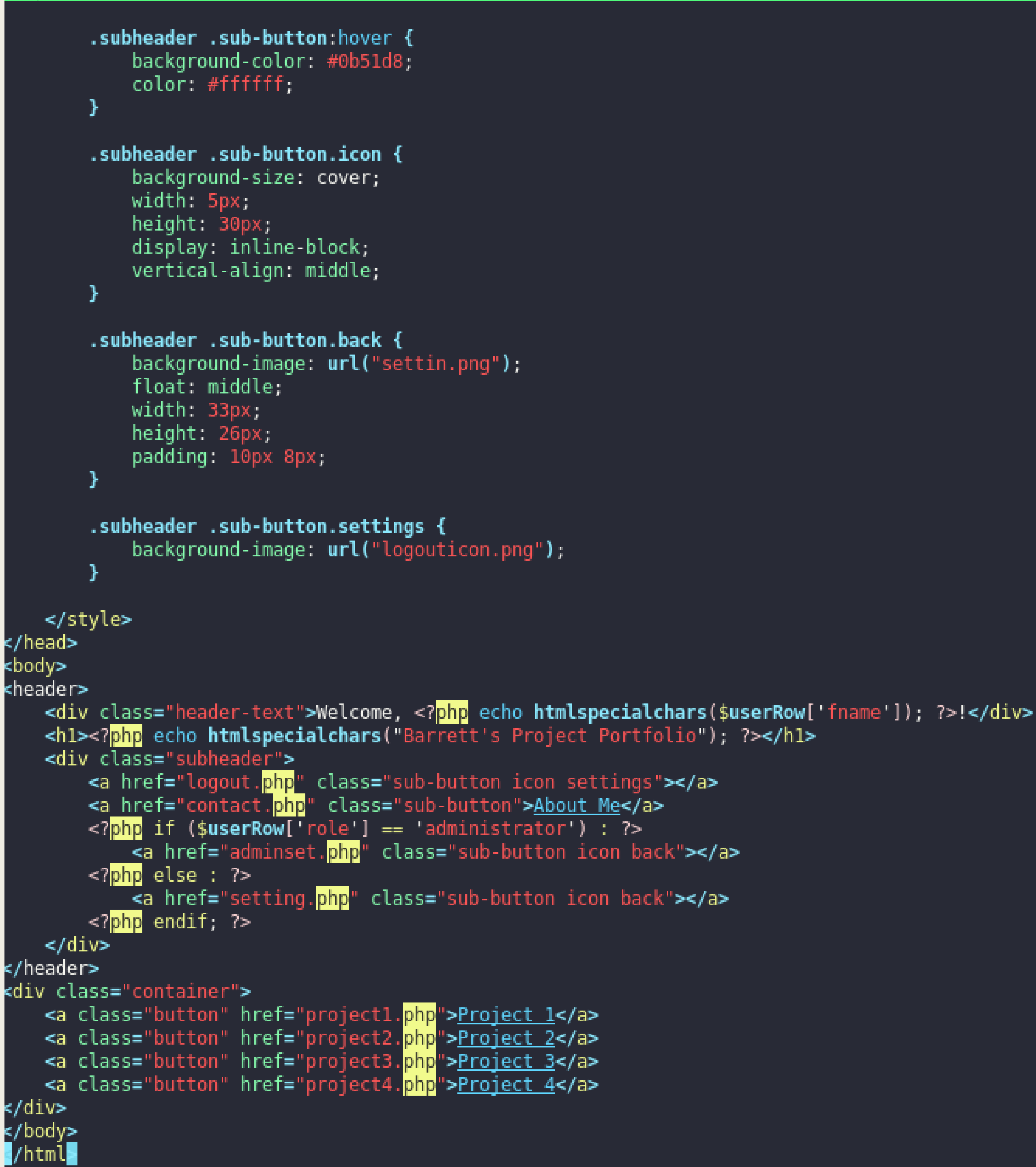
- Here it defines styles for the subheader, including text alignment and margin.



```
.container {
    max-width: 800px;
    margin: 20px auto;
    padding: 0 20px;
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 20px;
    text-align: center;
}

.button {
    display: block;
    height: 200px;
    background-color: #000000;
    color: white;
    text-decoration: none;
    border-radius: 10px;
    line-height: 200px;
    font-size: 20px;
    text-align: center;
}

.logout {
    position: absolute;
    top: 38px;
    right: 30px;
    background-color: #000000;
    color: #0b51d8;
    padding: 10px 20px;
    text-decoration: none;
    border-radius: 5px;
    transition: background-color 0.3s;
}

.logout:hover {
    background-color: #0b51d8;
    color: #ffffff;
}

.subheader {
    text-align: center;
    margin-top: 20px;
}

.subheader .sub-button {
    display: inline-block;
    margin: 0 20px;
    background-color: #000000;
    color: #0b51d8;
    text-decoration: none;
```

Home.php Code

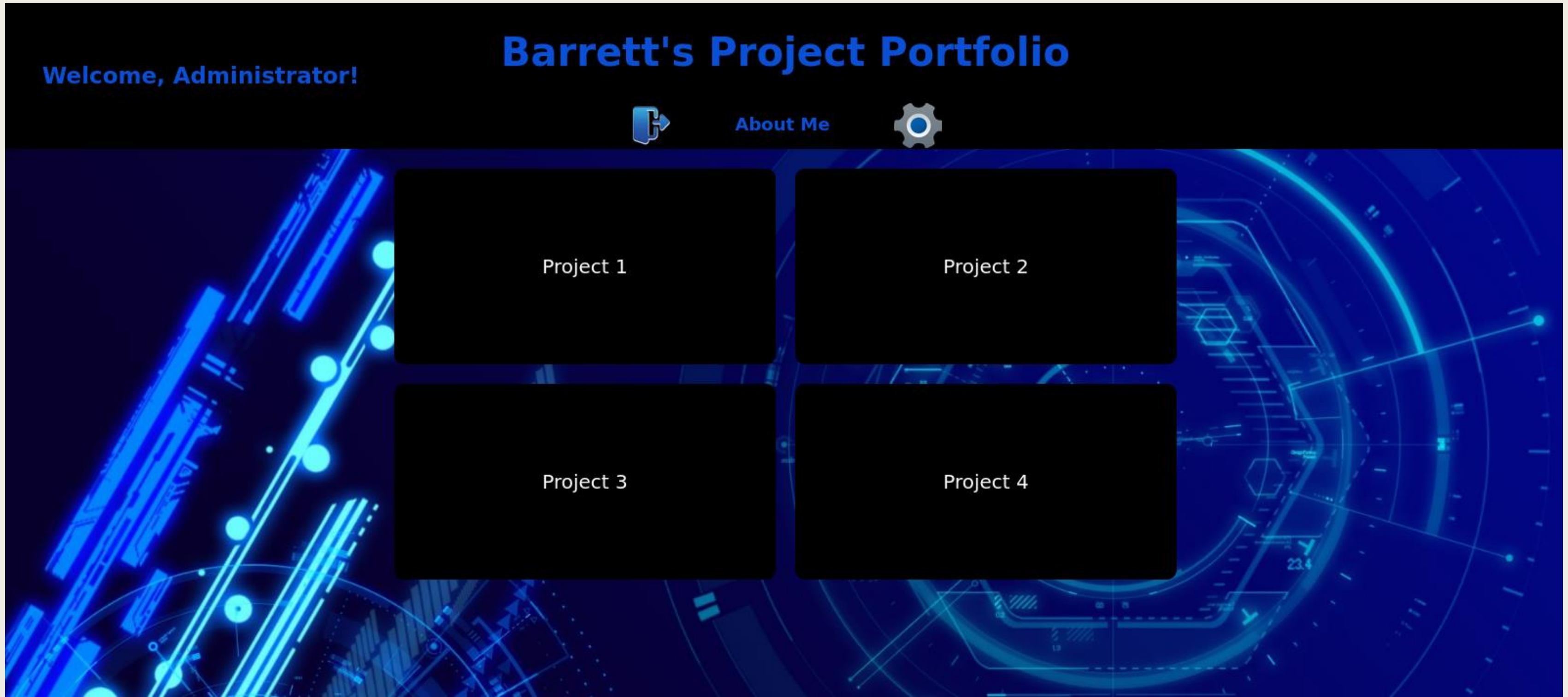
- Sub header and corresponding buttons styles are continued
- Here the page structure starts to be defined
- Included in the header content is the header with the welcome message, project portfolio title, and buttons that link to other pages.
- Here is where the projects will be put so that users can see my work



```
        .subheader .sub-button:hover {
            background-color: #0b51d8;
            color: #ffffff;
        }

        .subheader .sub-button.icon {
            background-size: cover;
            width: 5px;
            height: 30px;
            display: inline-block;
            vertical-align: middle;
        }

        .subheader .sub-button.back {
            background-image: url("settin.png");
            float: middle;
            width: 33px;
            height: 26px;
            padding: 10px 8px;
        }

        .subheader .sub-button.settings {
            background-image: url("logouticon.png");
        }

    </style>
</head>
<body>
<header>
    <div class="header-text">Welcome, <?php echo htmlspecialchars($userRow['fname']); ?>!</div>
    <h1><?php echo htmlspecialchars("Barrett's Project Portfolio"); ?></h1>
    <div class="subheader">
        <a href="logout.php" class="sub-button icon settings"></a>
        <a href="contact.php" class="sub-button">About Me</a>
        <?php if ($userRow['role'] == 'administrator') : ?>
            <a href="adminset.php" class="sub-button icon back"></a>
        <?php else : ?>
            <a href="setting.php" class="sub-button icon back"></a>
        <?php endif; ?>
    </div>
</header>
<div class="container">
    <a class="button" href="project1.php">Project 1</a>
    <a class="button" href="project2.php">Project 2</a>
    <a class="button" href="project3.php">Project 3</a>
    <a class="button" href="project4.php">Project 4</a>
</div>
</body>
</html>
```

Home.php Webpage

The project portfolio includes a header that features a personalized welcome message and title in the header. The navigation is intuitive, with a subheader offering links to different sections based on the user's role, such as the contact page and settings. The main content area showcases the user's projects with large, visually appealing buttons, making it easy for visitors to explore.



CONTACT.PHP

Contact.php Information

Description:

This webpage serves as an "About Me" section, providing visitors with a brief insight into my professional background. Below is my contact information which includes my email and links to my LinkedIn and GitHub pages.

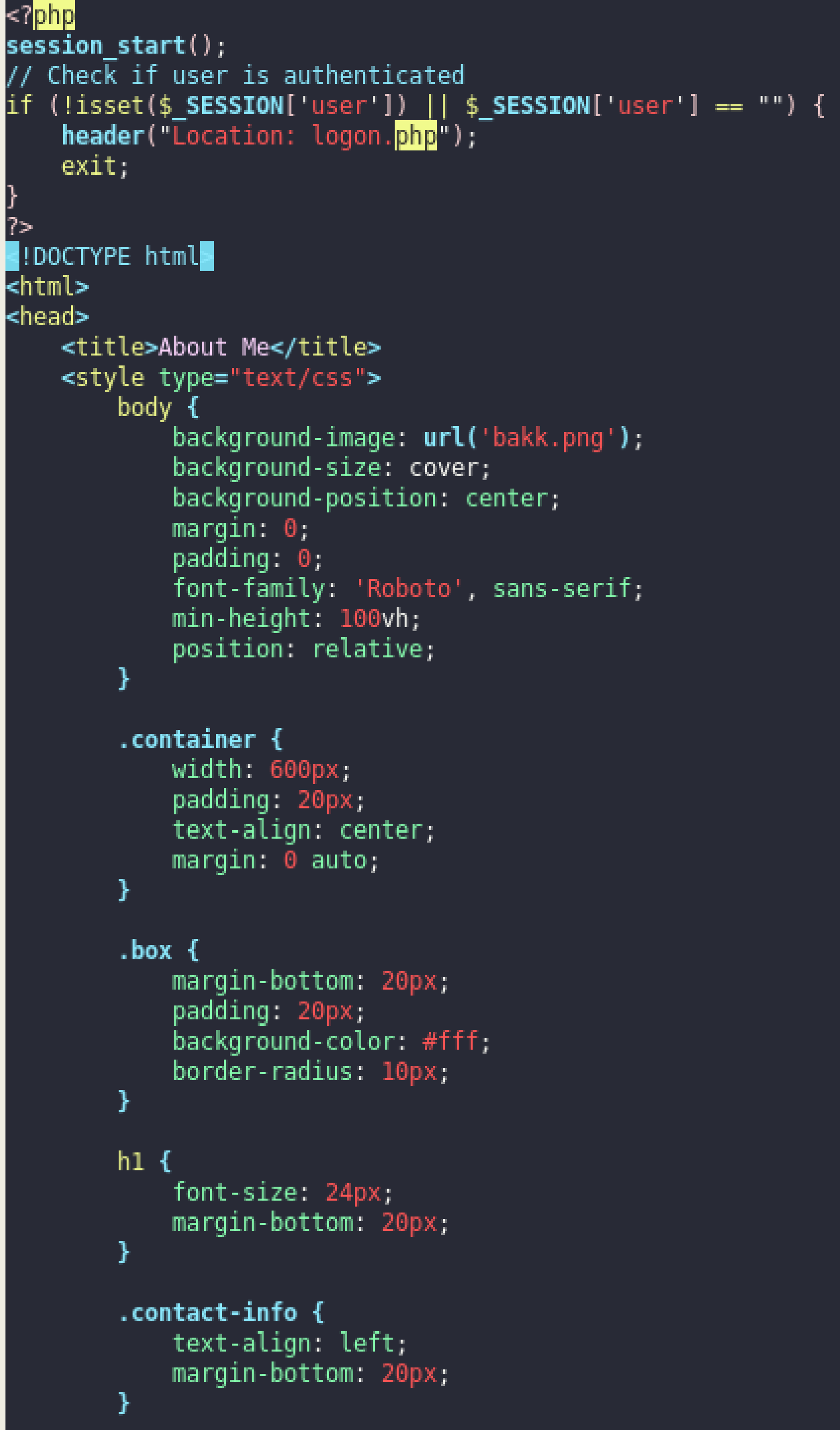
Key Points:

- The page features a home button with an icon, providing easy navigation back to the home page
- The page checks if the user is authenticated before allowing access, ensuring that only logged-in users can view the content
- The page has a clean and organized layout, with a container for content and styled boxes for different sections

Contact.php Code

- Here a new session is started or resumes an existing one, if the user is not logged in yet they are redirected to the login page

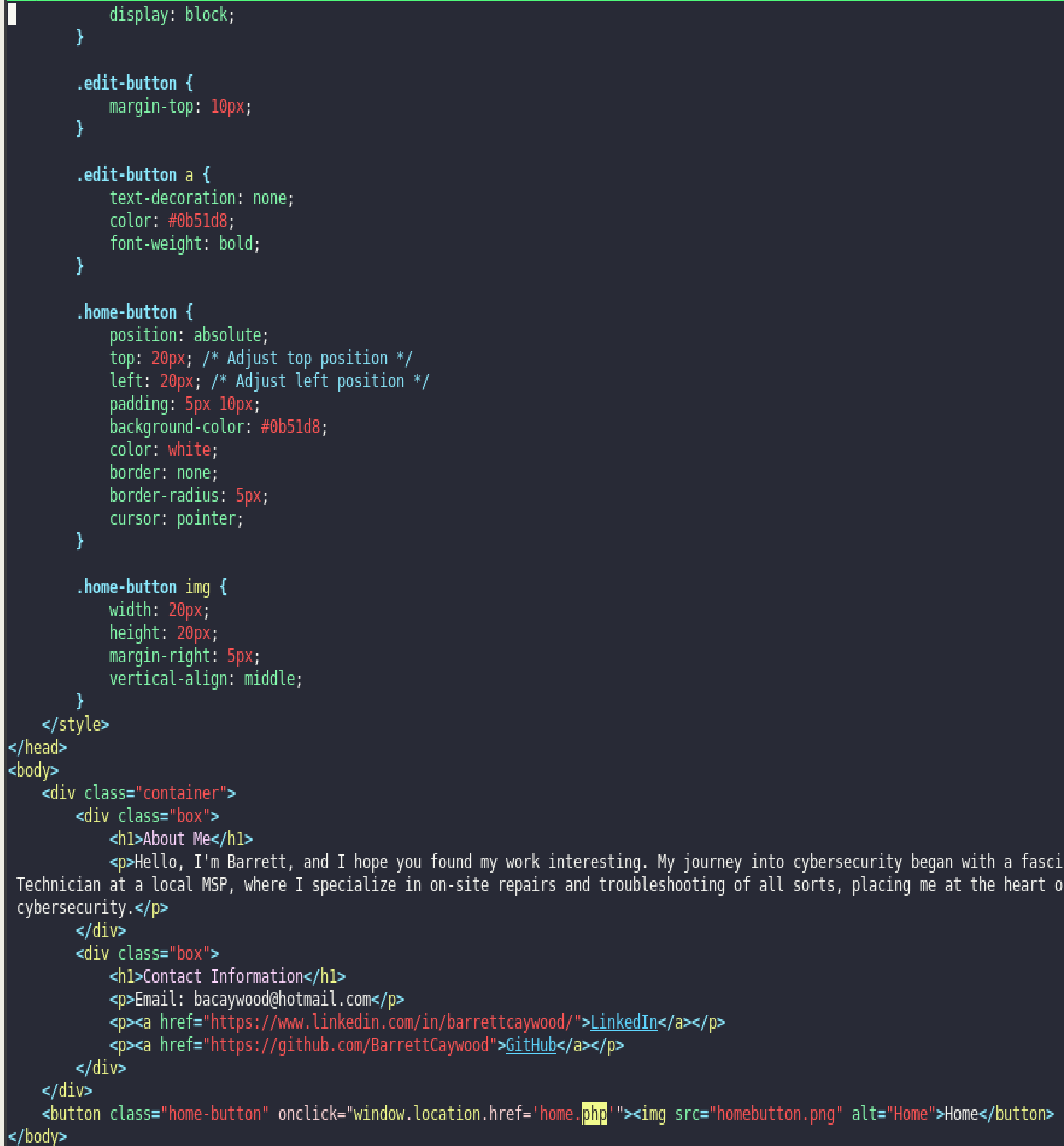
- Styles are defined for the entire document, including the body, container, boxes, headings, contact information, and buttons



```
<?php
session_start();
// Check if user is authenticated
if (!isset($_SESSION['user']) || $_SESSION['user'] == "") {
    header("Location: logon.php");
    exit;
}
?>
<!DOCTYPE html>
<html>
<head>
    <title>About Me</title>
    <style type="text/css">
        body {
            background-image: url('bakk.png');
            background-size: cover;
            background-position: center;
            margin: 0;
            padding: 0;
            font-family: 'Roboto', sans-serif;
            min-height: 100vh;
            position: relative;
        }

        .container {
            width: 600px;
            padding: 20px;
            text-align: center;
            margin: 0 auto;
        }

        .box {
            margin-bottom: 20px;
            padding: 20px;
            background-color: #fff;
            border-radius: 10px;
        }

        h1 {
            font-size: 24px;
            margin-bottom: 20px;
        }

        .contact-info {
            text-align: left;
            margin-bottom: 20px;
        }
```

Contact.php Code

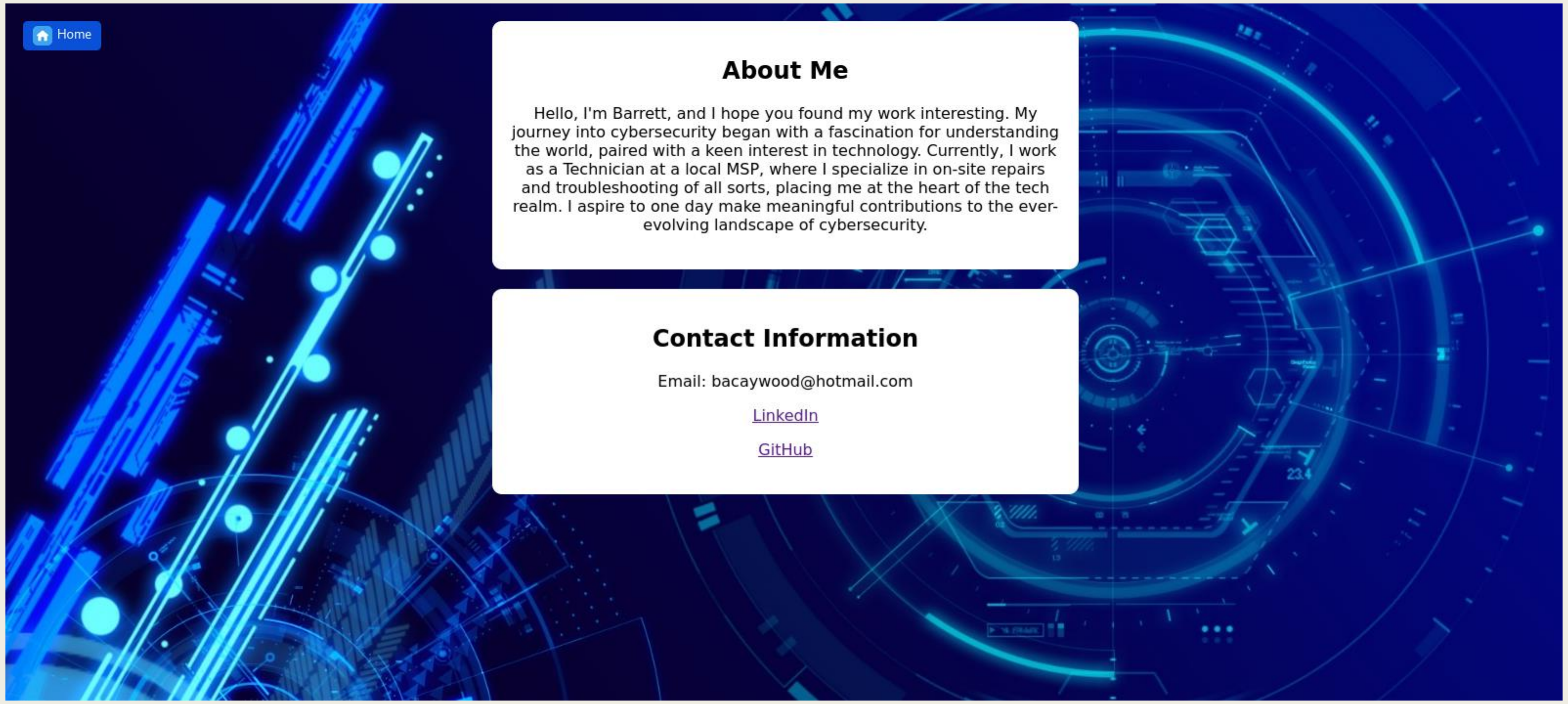
- Stylings for the contact information and about section of the page are defined
- Styling for the home button is defined here
- Here the page is structured with about me at the top with my contact information under



```
            display: block;
        }

        .edit-button {
            margin-top: 10px;
        }

        .edit-button a {
            text-decoration: none;
            color: #0b51d8;
            font-weight: bold;
        }

        .home-button {
            position: absolute;
            top: 20px; /* Adjust top position */
            left: 20px; /* Adjust left position */
            padding: 5px 10px;
            background-color: #0b51d8;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
        }

        .home-button img {
            width: 20px;
            height: 20px;
            margin-right: 5px;
            vertical-align: middle;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="box">
            <h1>About Me</h1>
            <p>Hello, I'm Barrett, and I hope you found my work interesting. My journey into cybersecurity began with a fasci
 Technician at a local MSP, where I specialize in on-site repairs and troubleshooting of all sorts, placing me at the heart o
 cybersecurity.</p>
        </div>
        <div class="box">
            <h1>Contact Information</h1>
            <p>Email: bacaywood@hotmail.com</p>
            <p><a href="https://www.linkedin.com/in/barrettcaywood/">LinkedIn</a></p>
            <p><a href="https://github.com/BarrettCaywood">GitHub</a></p>
        </div>
    </div>
    <button class="home-button" onclick="window.location.href='home.php'"><img src="homebutton.png" alt="Home">Home</button>
</body>
```

Contact.php Webpage

This page presents my professional background and contact information in a visually appealing and user-friendly manner, showcasing my skills and interests in cybersecurity. It also includes a “Home” button that redirects users back to the home page.



About Me

Hello, I'm Barrett, and I hope you found my work interesting. My journey into cybersecurity began with a fascination for understanding the world, paired with a keen interest in technology. Currently, I work as a Technician at a local MSP, where I specialize in on-site repairs and troubleshooting of all sorts, placing me at the heart of the tech realm. I aspire to one day make meaningful contributions to the ever-evolving landscape of cybersecurity.

Contact Information

Email: bacaywood@hotmail.com

LinkedIn

GitHub

SETTING.PHP

Setting.php Information

Description:

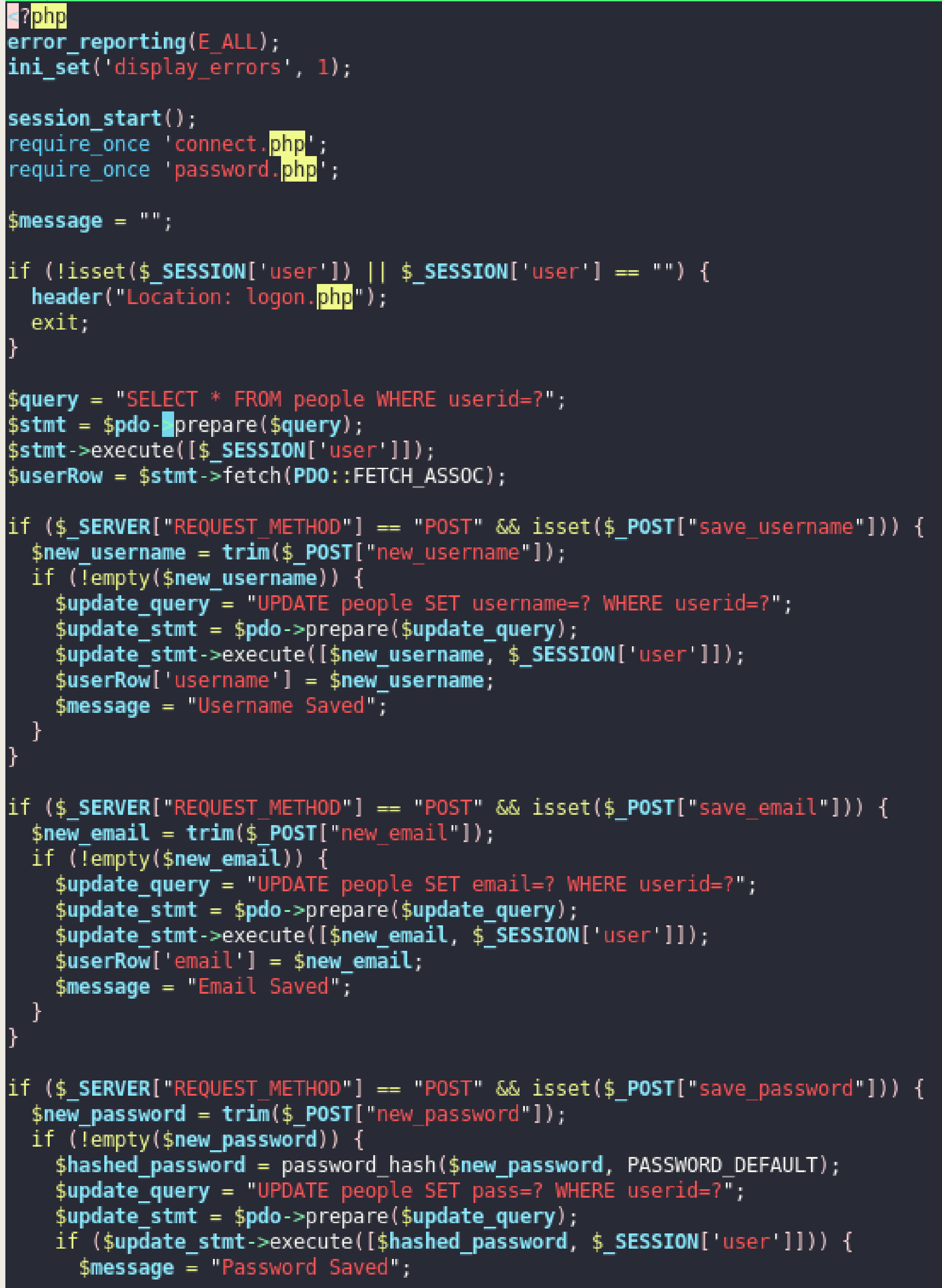
The settings page is a user-friendly interface for updating account information which includes username, email, and password. Each section has a "Save" button for submitting updates, and a message area provides feedback on the outcome of the operations. A "Home" button allows users to easily return back to the home page.

Key Points:

- A message displays feedback on the outcome of the update operations (e.g., "Username Saved")
- Each input form has a "Save" button to submit the changes
- The "Home" button at the top left corner allows the user to return back to the main page

Setting.php Code

- The code begins with error reporting and session start to handle any potential errors and manage user
- Soon after a statement includes a file that establishes a connection to the MySQL
- The code checks if a user is logged in by verifying the existence of a session user variable.
  - If not, it redirects the user to the login page

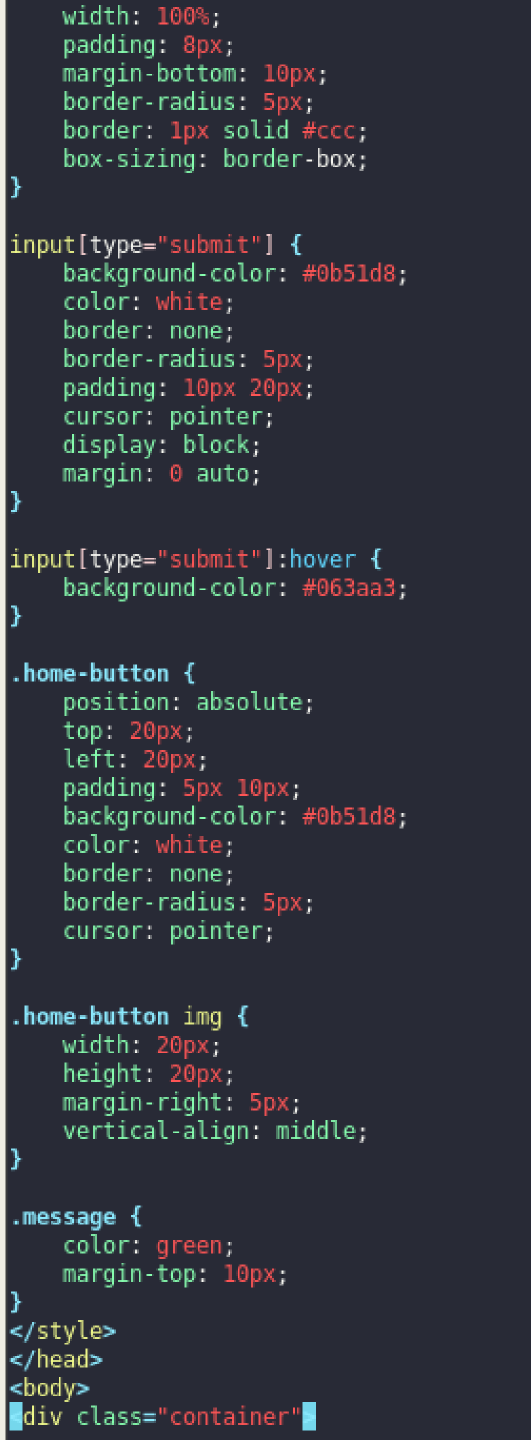


```php
<?php
error_reporting(E_ALL);
ini_set('display_errors', 1);

session_start();
require_once 'connect.php';
require_once 'password.php';

$message = "";

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

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

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["save_username"])) {
  $new_username = trim($_POST["new_username"]);
  if (!empty($new_username)) {
    $update_query = "UPDATE people SET username=? WHERE userid=?";
    $update_stmt = $pdo->prepare($update_query);
    $update_stmt->execute([$new_username, $_SESSION['user']]);
    $userRow['username'] = $new_username;
    $message = "Username Saved";
  }
}

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["save_email"])) {
  $new_email = trim($_POST["new_email"]);
  if (!empty($new_email)) {
    $update_query = "UPDATE people SET email=? WHERE userid=?";
    $update_stmt = $pdo->prepare($update_query);
    $update_stmt->execute([$new_email, $_SESSION['user']]);
    $userRow['email'] = $new_email;
    $message = "Email Saved";
  }
}

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["save_password"])) {
  $new_password = trim($_POST["new_password"]);
  if (!empty($new_password)) {
    $hashed_password = password_hash($new_password, PASSWORD_DEFAULT);
    $update_query = "UPDATE people SET pass=? WHERE userid=?";
    $update_stmt = $pdo->prepare($update_query);
    if ($update_stmt->execute([$hashed_password, $_SESSION['user']])) {
      $message = "Password Saved";
```

Setting.php Code

- The background image, font, and home and submit button styles are defined to enhance the overall look and feel of the page

- The styling for the message that updates the user on their changes is also defined here



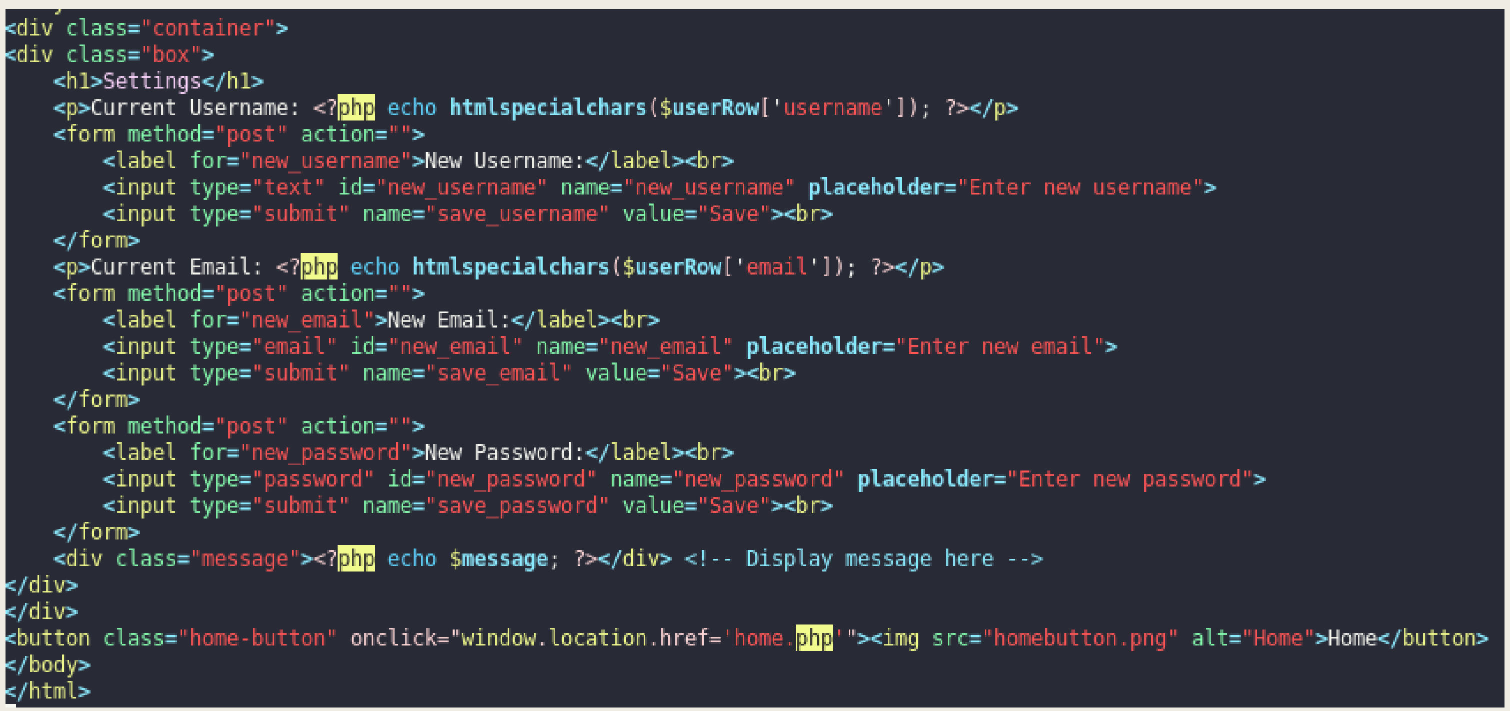
```
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
    box-sizing: border-box;
}

input[type="submit"] {
    background-color: #0b51d8;
    color: white;
    border: none;
    border-radius: 5px;
    padding: 10px 20px;
    cursor: pointer;
    display: block;
    margin: 0 auto;
}

input[type="submit"]:hover {
    background-color: #063aa3;
}

.home-button {
    position: absolute;
    top: 20px;
    left: 20px;
    padding: 5px 10px;
    background-color: #0b51d8;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

.home-button img {
    width: 20px;
    height: 20px;
    margin-right: 5px;
    vertical-align: middle;
}

.message {
    color: green;
    margin-top: 10px;
}
</style>
</head>
<body>
div class="container"
```

Setting.php Code

- The HTML section includes forms for updating username, email, and password. Each form contains input fields for the new value and a submit button.

- The message area displays feedback messages (e.g., "Username Saved") to inform the user of the outcome of their actions.

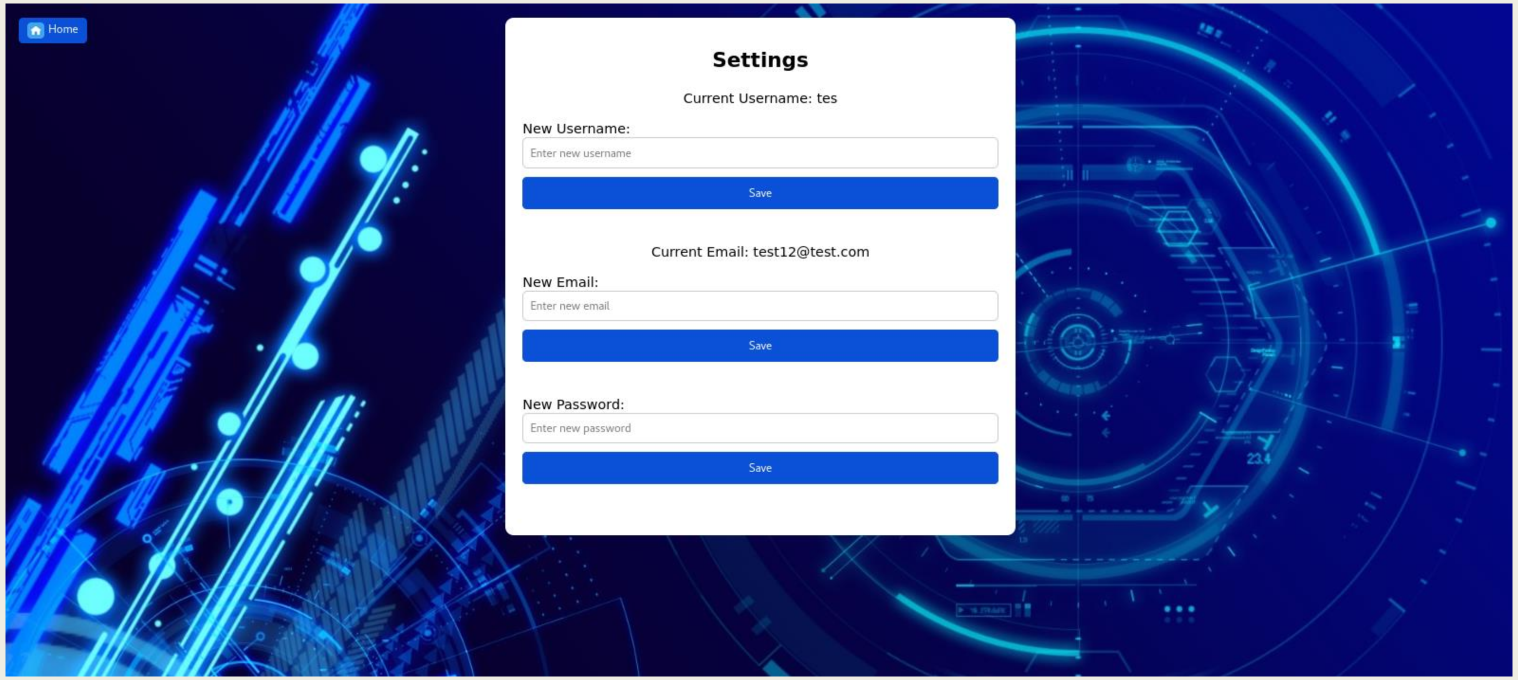
- A "Home" button is added a the bottom of the page, allowing users to return to the home page easily.



```
<div class="container">
<div class="box">
    <h1>Settings</h1>
    <p>Current Username: <?php echo htmlspecialchars($userRow['username']); ?></p>
    <form method="post" action="">
        <label for="new_username">New Username:</label><br>
        <input type="text" id="new_username" name="new_username" placeholder="Enter new username">
        <input type="submit" name="save_username" value="Save"><br>
    </form>
    <p>Current Email: <?php echo htmlspecialchars($userRow['email']); ?></p>
    <form method="post" action="">
        <label for="new_email">New Email:</label><br>
        <input type="email" id="new_email" name="new_email" placeholder="Enter new email">
        <input type="submit" name="save_email" value="Save"><br>
    </form>
    <form method="post" action="">
        <label for="new_password">New Password:</label><br>
        <input type="password" id="new_password" name="new_password" placeholder="Enter new password">
        <input type="submit" name="save_password" value="Save"><br>
    </form>
    <div class="message"><?php echo $message; ?></div> <!-- Display message here -->
</div>
</div>
<button class="home-button" onclick="window.location.href='home.php'"><img src="homebutton.png" alt="Home">Home</button>
</body>
</html>
```

Setting.php Webpage

The webpage is a user-friendly interface for updating account information, including username, email, and password. It features a simple layout with input fields for each type of information and a "Save" button to submit changes. Feedback messages are displayed to inform users of their changes. Additionally, a "Home" button is provided for easy navigation back to the home page.



Current Username: tes

New Username:

Enter new username

Current Email: test12@test.com

New Email:

Enter new email

New Password:

Enter new password

ADMINSET.PHP

(Administrator Settings)

Adminset.php Information

Description:

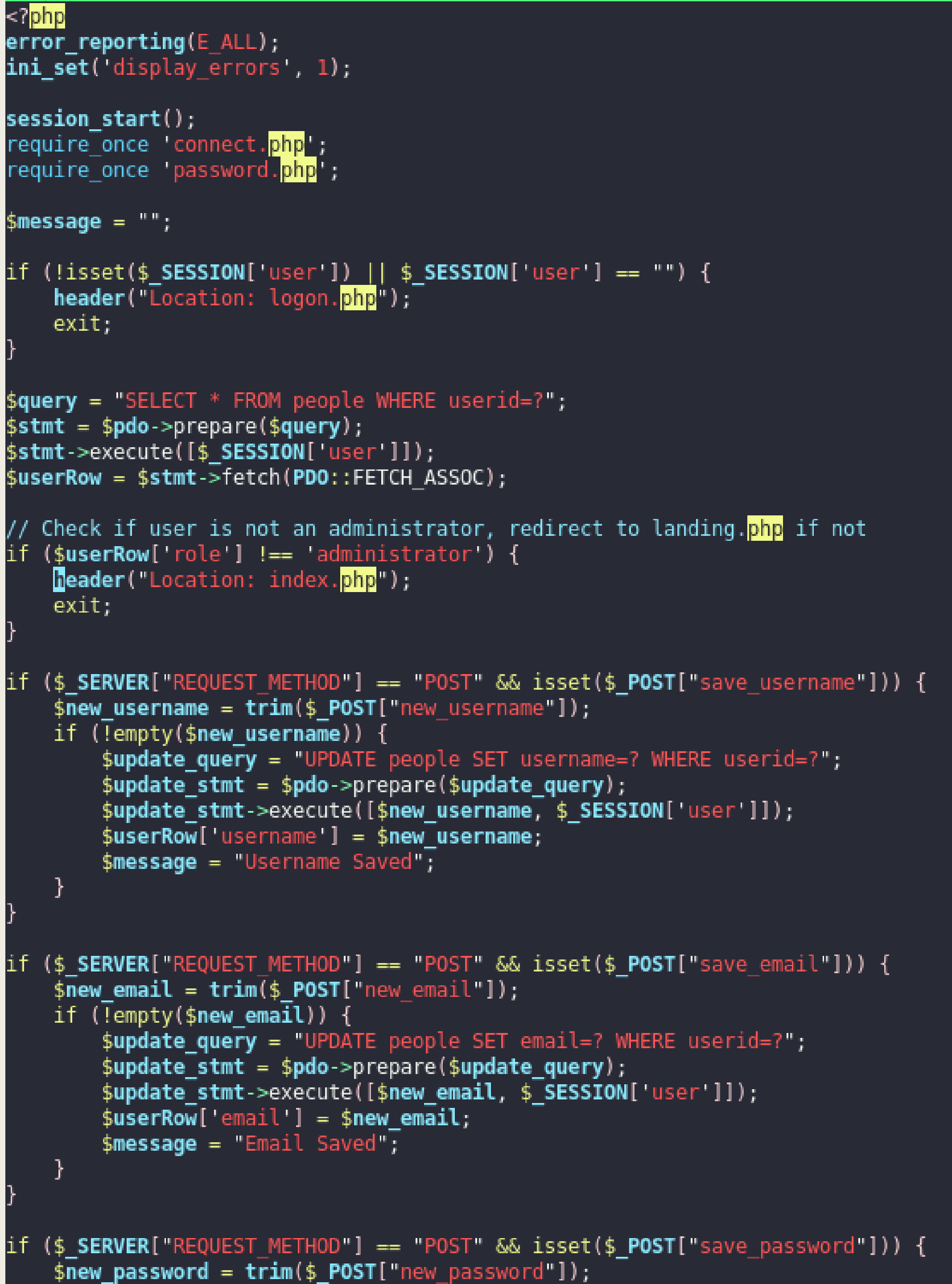
This webpage is an administrator settings page designed for managing user accounts. It includes functionality to update usernames, emails, passwords, and user roles. The page features a clean and intuitive layout with input fields and dropdown menus for making changes, along with feedback messages to inform administrators of the outcomes of their actions. A "Home" button is also included to allow for easy navigation back to the home page.

Key Points:

- The code checks if the logged-in user is an administrator. If not, it redirects them to the landing page
- In addition to updating the username, email, and password, administrators can also change a user's role between "User" and "Administrator”
- Feedback messages are displayed to inform administrators of the outcome of their actions

Adminset.php Code

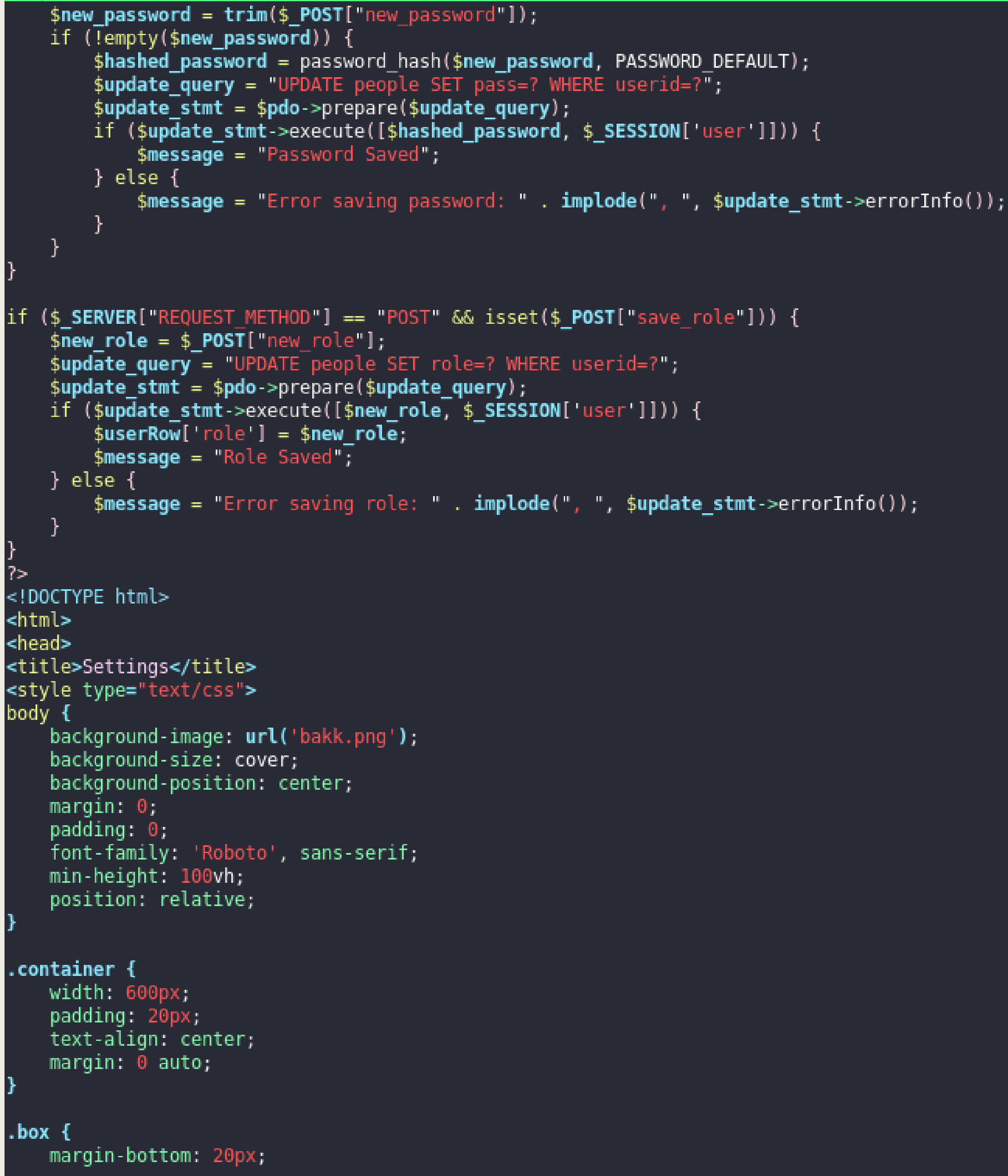
- The code starts with error reporting and session start to handle errors and manage user sessions
- Then establishes a connection to the MySQL database
- It checks if a user is logged in and redirects to the login page if not
- It redirects non-administrators to the landing page to restrict access to the administrator settings page



```php
<?php
error_reporting(E_ALL);
ini_set('display_errors', 1);

session_start();
require_once 'connect.php';
require_once 'password.php';

$message = "";

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

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

// Check if user is not an administrator, redirect to landing.php if not
if ($userRow['role'] !== 'administrator') {
    header("Location: index.php");
    exit;
}

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["save_username"])) {
    $new_username = trim($_POST["new_username"]);
    if (!empty($new_username)) {
        $update_query = "UPDATE people SET username=? WHERE userid=?";
        $update_stmt = $pdo->prepare($update_query);
        $update_stmt->execute([$new_username, $_SESSION['user']]);
        $userRow['username'] = $new_username;
        $message = "Username Saved";
    }
}

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["save_email"])) {
    $new_email = trim($_POST["new_email"]);
    if (!empty($new_email)) {
        $update_query = "UPDATE people SET email=? WHERE userid=?";
        $update_stmt = $pdo->prepare($update_query);
        $update_stmt->execute([$new_email, $_SESSION['user']]);
        $userRow['email'] = $new_email;
        $message = "Email Saved";
    }
}

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["save_password"])) {
    $new_password = trim($_POST["new_password"]);
```

Adminset.php Code

- It hashes the new password and updates it in the database, displaying a feedback message based on the outcome
- Then it updates the user's role in the database based on the selection made in the dropdown menu
- Background image, font, and button styles are used to enhance the overall design.



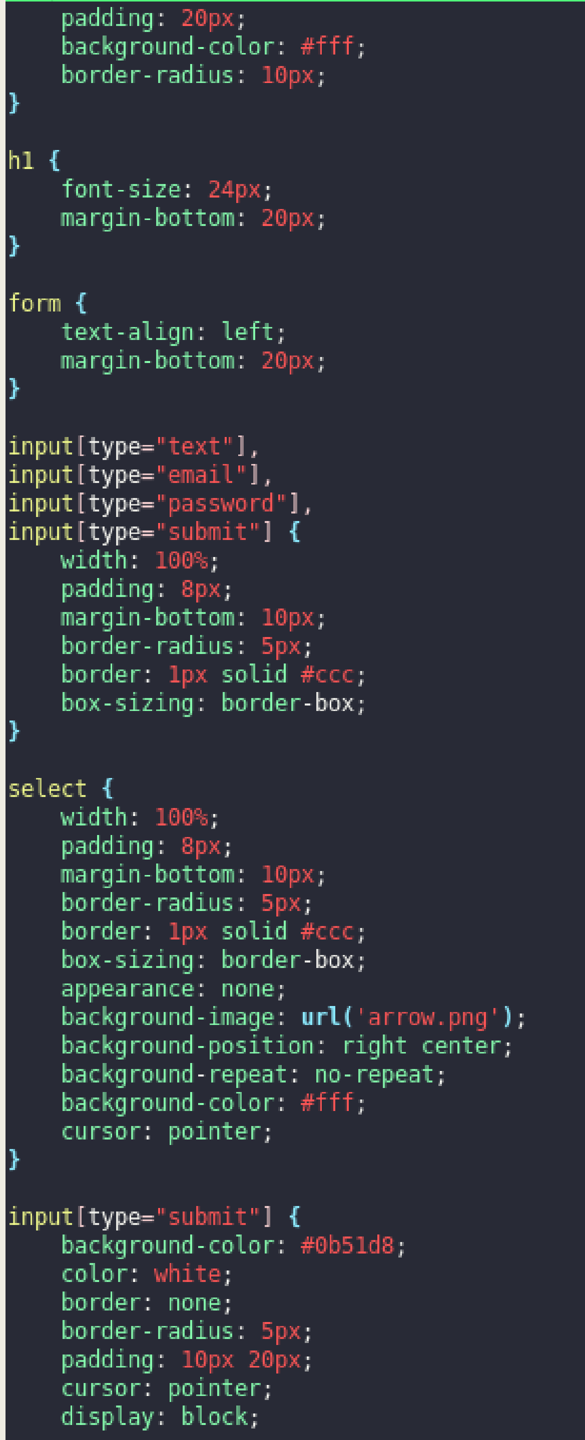
```
    $new_password = trim($_POST["new_password"]);
    if (!empty($new_password)) {
        $hashed_password = password_hash($new_password, PASSWORD_DEFAULT);
        $update_query = "UPDATE people SET pass=? WHERE userid=?";
        $update_stmt = $pdo->prepare($update_query);
        if ($update_stmt->execute([$hashed_password, $_SESSION['user']])) {
            $message = "Password Saved";
        } else {
            $message = "Error saving password: " . implode(", ", $update_stmt->errorInfo());
        }
    }
}

if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST["save_role"])) {
    $new_role = $_POST["new_role"];
    $update_query = "UPDATE people SET role=? WHERE userid=?";
    $update_stmt = $pdo->prepare($update_query);
    if ($update_stmt->execute([$new_role, $_SESSION['user']])) {
        $userRow['role'] = $new_role;
        $message = "Role Saved";
    } else {
        $message = "Error saving role: " . implode(", ", $update_stmt->errorInfo());
    }
}
?>
<!DOCTYPE html>
<html>
<head>
<title>Settings</title>
<style type="text/css">
body {
    background-image: url('bakk.png');
    background-size: cover;
    background-position: center;
    margin: 0;
    padding: 0;
    font-family: 'Roboto', sans-serif;
    min-height: 100vh;
    position: relative;
}

.container {
    width: 600px;
    padding: 20px;
    text-align: center;
    margin: 0 auto;
}

.box {
    margin-bottom: 20px;
```

Adminset.php Code

- This section defines styles for the body, container, input fields, buttons, and update message displayed to users

- Here the style for the drop-down menu to choose the users role is defined

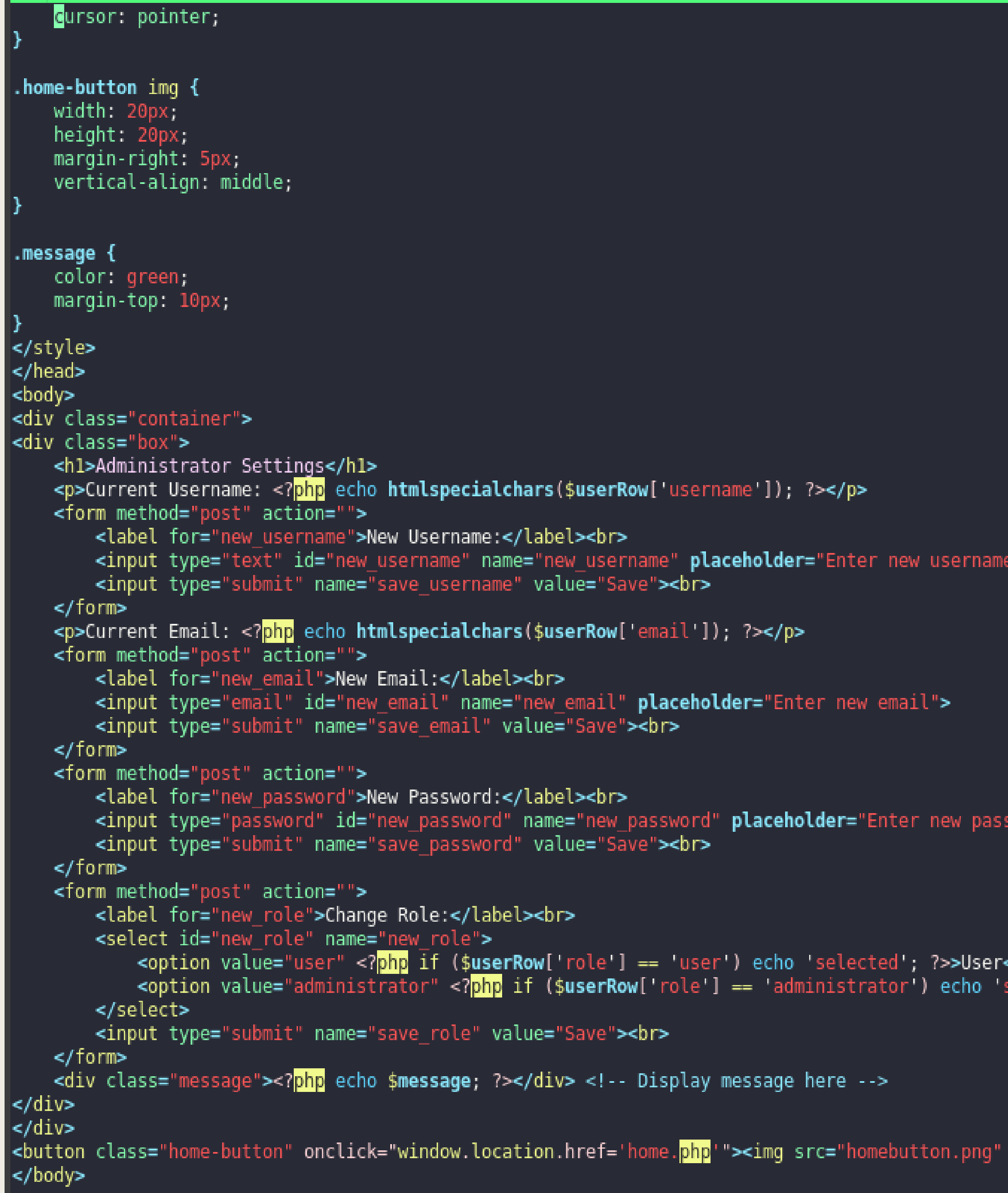
- The save button style is defined here



```
    padding: 20px;
    background-color: #fff;
    border-radius: 10px;
}

h1 {
    font-size: 24px;
    margin-bottom: 20px;
}

form {
    text-align: left;
    margin-bottom: 20px;
}

input[type="text"],
input[type="email"],
input[type="password"],
input[type="submit"] {
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
    box-sizing: border-box;
}

select {
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
    box-sizing: border-box;
    appearance: none;
    background-image: url('arrow.png');
    background-position: right center;
    background-repeat: no-repeat;
    background-color: #fff;
    cursor: pointer;
}

input[type="submit"] {
    background-color: #0b51d8;
    color: white;
    border: none;
    border-radius: 5px;
    padding: 10px 20px;
    cursor: pointer;
    display: block;
```

Adminset.php Code

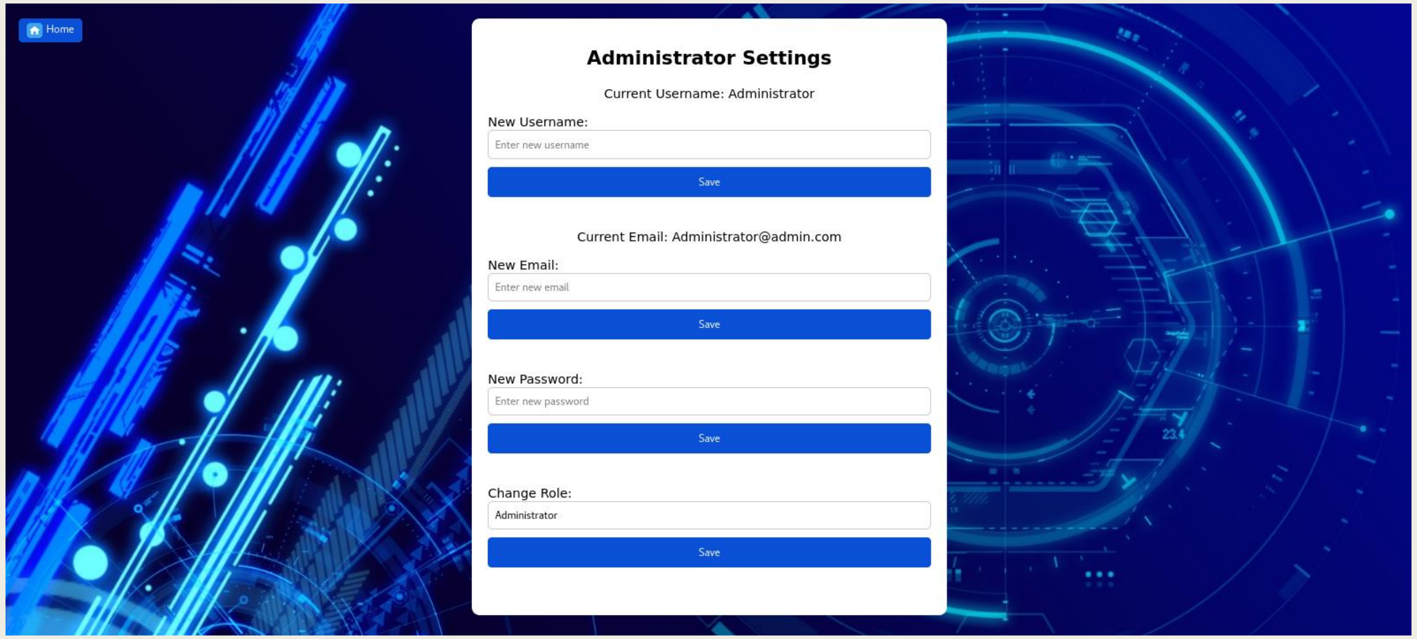
- The HTML section includes forms for updating username, email, password, and role, each with input fields or dropdown menus and a "Save" button
- Here the message area displays feedback messages (e.g., "Username Saved") to inform administrators of the outcome of their actions
- A "Home" button is added at the bottom of the page, allowing administrators to return to the home page easily



```
    cursor: pointer;
}

.home-button img {
    width: 20px;
    height: 20px;
    margin-right: 5px;
    vertical-align: middle;
}

.message {
    color: green;
    margin-top: 10px;
}
</style>
</head>
<body>
<div class="container">
<div class="box">
    <h1>Administrator Settings</h1>
    <p>Current Username: <?php echo htmlspecialchars($userRow['username']); ?></p>
    <form method="post" action="">
        <label for="new_username">New Username:</label><br>
        <input type="text" id="new_username" name="new_username" placeholder="Enter new username
        <input type="submit" name="save_username" value="Save"><br>
    </form>
    <p>Current Email: <?php echo htmlspecialchars($userRow['email']); ?></p>
    <form method="post" action="">
        <label for="new_email">New Email:</label><br>
        <input type="email" id="new_email" name="new_email" placeholder="Enter new email">
        <input type="submit" name="save_email" value="Save"><br>
    </form>
    <form method="post" action="">
        <label for="new_password">New Password:</label><br>
        <input type="password" id="new_password" name="new_password" placeholder="Enter new pass
        <input type="submit" name="save_password" value="Save"><br>
    </form>
    <form method="post" action="">
        <label for="new_role">Change Role:</label><br>
        <select id="new_role" name="new_role">
            <option value="user" <?php if ($userRow['role'] == 'user') echo 'selected'; ?>>User
            <option value="administrator" <?php if ($userRow['role'] == 'administrator') echo '
        </select>
        <input type="submit" name="save_role" value="Save"><br>
    </form>
    <div class="message"><?php echo $message; ?></div> <!-- Display message here -->
</div>
</div>
<button class="home-button" onclick="window.location.href='home.php'"><img src="homebutton.png"
</body>
```

Adminset.php Webpage

This webpage is a secure and user-friendly platform designed for administrators to manage user accounts and roles. It features error reporting and session handling for secure authentication, ensuring only authorized users can access and make changes. The layout includes sections for updating usernames, emails, passwords, and user roles. Feedback messages are displayed to inform administrators of the outcome of their actions.



Administrator Settings

Current Username: Administrator

New Username:

Current Email: Administrator@admin.com

New Email:

New Password:

Change Role:

Administrator

LOGOUT.PHP

Logout.php Information

Description:

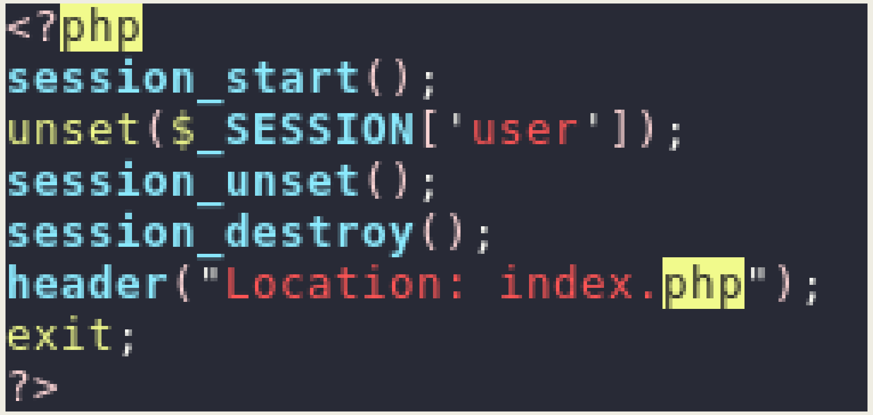
This PHP script serves as a logout page for a website. When accessed, it initiates a session, unsets and destroys the 'user' session variable, logging the user out. It then redirects the user to the landing page.

Key Points:

- Utilizes sessions to manage user authentication.
- Unsets and destroys the session variable, ensuring the user is logged out.
- Redirects the user to the homepage after logout.
- Ensures a secure logout process to protect user data.

Logout.php Code

- It begins by starting the session and then unsetting the 'user' session variable, logging the user out.
- The next function is called to remove all session variables, and after it is used to destroy the session data.
- After the logout process is complete, the script redirects the user to the landing page.



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

404 ERROR PAGE

404error.php Information

Description:

This webpage is a custom 404 error page, which is displayed when a user tries to access a page that does not exist on the website.

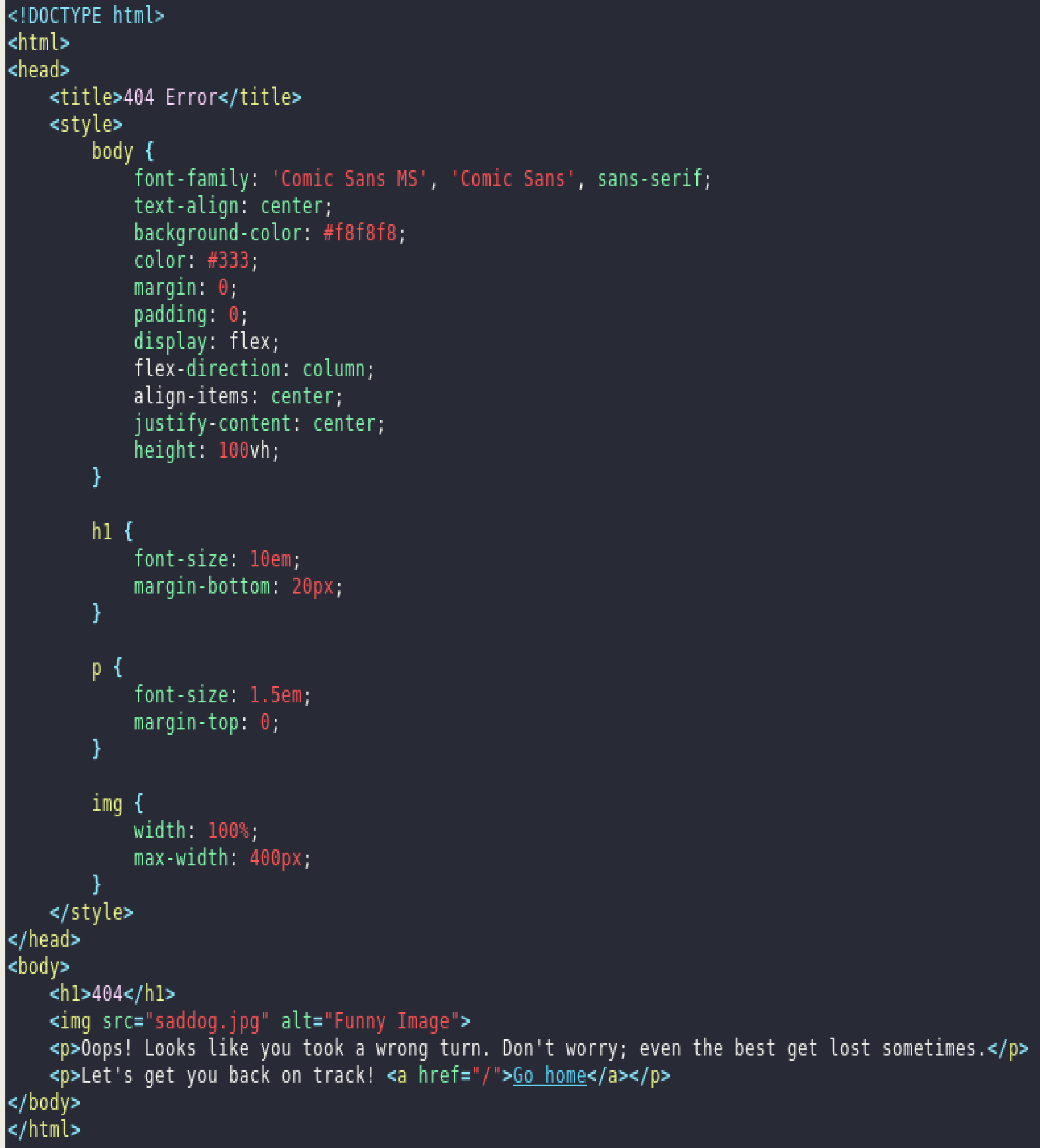
Key Points:

- It features a large "404" heading to clearly indicate the error, along with a message reassuring the user and offering a link to return to the homepage

- The page also features Comic Sans MS font and a humorous image of a sad dog

404error.php Code

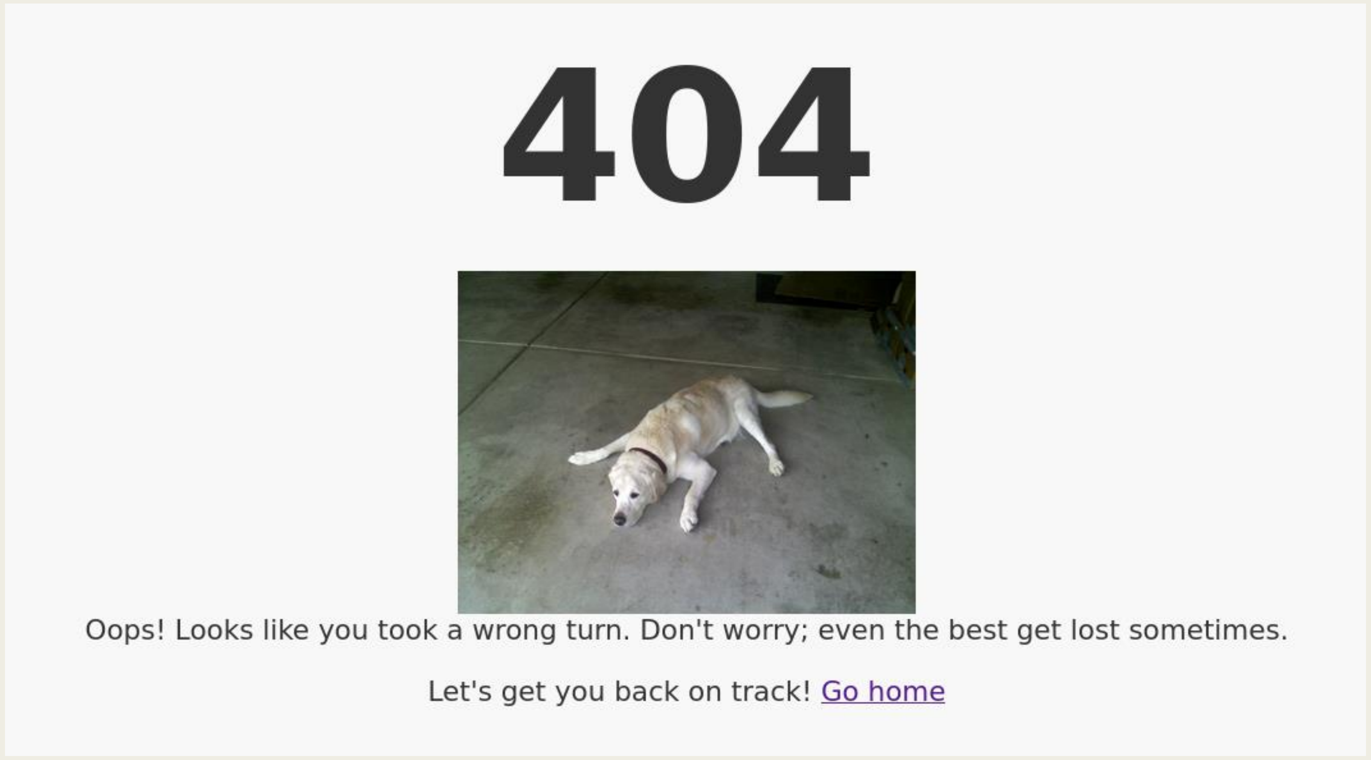
- Here the styling for the elements on the page is defined
- The heading section includes a large "404" heading to clearly indicate the error
- The sad dog image is applied for visual appeal, as well as a message that reassures the user with a humorous touch



```html
<!DOCTYPE html>
<html>
<head>
    <title>404 Error</title>
    <style>
        body {
            font-family: 'Comic Sans MS', 'Comic Sans', sans-serif;
            text-align: center;
            background-color: #f8f8f8;
            color: #333;
            margin: 0;
            padding: 0;
            display: flex;
            flex-direction: column;
            align-items: center;
            justify-content: center;
            height: 100vh;
        }

        h1 {
            font-size: 10em;
            margin-bottom: 20px;
        }

        p {
            font-size: 1.5em;
            margin-top: 0;
        }

        img {
            width: 100%;
            max-width: 400px;
        }
    </style>
</head>
<body>
    <h1>404</h1>
    <img src="saddog.jpg" alt="Funny Image">
    <p>Oops! Looks like you took a wrong turn. Don't worry; even the best get lost sometimes.</p>
    <p>Let's get you back on track! <a href="/">Go home</a></p>
</body>
</html>
```

404error.php Webpage

This webpage is a custom 404 error page, which is displayed when a user tries to access a page that does not exist on the website.



404

Oops! Looks like you took a wrong turn. Don't worry; even the best get lost sometimes.

Let's get you back on track! Go home