**Assignment 3**

David Rutledge

The University of Arizona Global Campus

CST499: Capstone for Computer Software Technology

Charmelia Butler

January 13, 2025

**Introduction**

The Registration App is built using PHP, CSS, and MySQL coding machine languages and runs on a localhost using XAMPP. The project involves creating the required page layouts and integrating a MySQL for secure data storage. This paper explains key process elements for running PHP files in XAMPP, developing the web pages, creating and connecting a database, and implementing the registration functionality.

**How to Run a PHP File in XAMPP**

To run a PHP file in XAMPP, first you must download and install XAMPP from the website. Follow the instructions for Mac or Windows, depending on your computer’s operating system. Once XAMPP is installed, open the XAMPP control panel and start the Apache server by clicking “Start”.

I am using Mac, so I will continue with instructions for Mac OS. Open a terminal. In the finder, drag the htdocs folder into the Terminal, (Ghost Together, 2022). Type ‘cd’ before the htdocs to create a new directory. You can now check the content by using the ‘ls’ command. Use the mkdir command followed by a name for your new folder. In our case we used ‘class’. Type ‘cd class’ to enter that folder’s directory. Enter touch index.php to create an empty file.

Next, go to VSCode and open the ‘class’ file we just created. Now you are ready to type your PHP script. When you are finished scripting, save your PHP file to index.php. You can then open a browser, type http://localhost/index.php in the address bar, and press Enter. The PHP file should execute.

By following these steps, you can test and execute PHP scripts locally, ensuring that they function as intended before deploying them to a live server.

**Developing the Landing, Login, and Registration Pages**

**Landing/Login Page**

When the user types in the URL for the Registration App, they are brought to the landing and login page. I chose to combine these two pages, so that users can login immediately, as is the standard for most websites. Password are encrypted, and the user can access their account securely.

Figure 1.

A screenshot of a login screen

Description automatically generated

**Registration Page**

If a user does not have an account, they are directed to the registration page from the landing/login page. The registration page allows the user to input their relevant personal information, such as username, email, and password, to create an account.

The registration page contains the following fields: username, password, email, first name, last name, and phone number. If a field is not completed, the user will not be able to submit the form and is given a prompt to complete the missing field.

The backend verifies the form is submitted through a POST request, (Nixon, 2012). The server-side script ensure that the required fields are completed. Passwords are hashed using PHP’s password\_hash function prior to being stored for security.

The user receives feedback from success or error messages for cases such as duplicate user name or email. When registration is completed successfully,

To provide feedback, the page includes mechanisms to display success or error messages. For instance, if a user attempts to register with a duplicate username or email, the system alerts them with a specific error message. Conversely, upon successful registration, users are notified and encouraged to log in to their new account.

Figure 2.

A screenshot of a login form

Description automatically generated

**Database Design**

It is important to have a sound database structure for storing user data securely and to ensure all interactions are smooth. The users table includes fields for unique identification (user\_id), username, hashed password, email, first name, last name, phone number, and timestamps for creation and any updates. The simple design requests basic information required for most registration systems, which should prove to be familiar and comfortable for users.

The username and email fields do not allow duplicate entries from other users. Each entry into these fields must be unique. This will prevent potential conflicts at login.

Figure 3.

A screenshot of a computer

Description automatically generated

Figure 4.

A screenshot of a software application

Description automatically generated

Figure 5.

A screenshot of a computer

Description automatically generated

**Steps for Implementation**

Several key steps were taken to build the Registration App with a focus on security, functionality, and user experience. Here are the steps:

1. **Database Setup**: First, the users table was created in a MySQL database using appropriate data types. The PHP application establishes a connection to the database with connection credentials configured in the database.php file.
2. **Form Design and Validation**: The registration form collects user information. Once submitted, validates the input data and checks for empty fields. Feedback is provided for errors or success.
3. **Password Security**: The system uses password\_hash to encrypt passwords before being stored in the database, ensuring passwords remain secure.
4. **Database Interaction**: PHP Data Objects are used for database interaction. This provides protection from injection attacks and ensures there are no duplicate records inserted.

**Source Code**

index.php

<?php

session\_start();

// Redirect if already logged in

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

header("Location: index.php");

exit();

}

require\_once 'config/database.php';

require\_once 'classes/User.php';

$database = new Database();

$db = $database->getConnection();

$user = new User($db);

$message = '';

$messageType = '';

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

$username = $\_POST['username'] ?? '';

$password = $\_POST['password'] ?? '';

if (empty($username) || empty($password)) {

$message = "Please fill in all fields";

$messageType = "error";

} else {

if ($user->login($username, $password)) {

// Set session variables

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

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

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

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

// Redirect to dashboard/home page

header("Location: index.php");

exit();

} else {

$message = "Invalid username or password";

$messageType = "error";

}

}

}

include 'includes/header.php';

?>

<div class="form-container">

<h2>Login</h2>

<?php if ($message): ?>

<div class="alert alert-<?php echo $messageType; ?>">

<?php echo $message; ?>

</div>

<?php endif; ?>

<form action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" method="post">

<div class="form-group">

<label for="username" class="form-label">Username</label>

<input type="text" name="username" id="username" class="form-input"

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

</div>

<div class="form-group">

<label for="password" class="form-label">Password</label>

<input type="password" name="password" id="password" class="form-input" required>

</div>

<button type="submit" class="btn btn-primary">Login</button>

</form>

<p class="form-footer">

Don't have an account? <a href="register.php">Register here</a><br>

<a href="forgot-password.php">Forgot Password?</a>

</p>

</div>

<?php include 'includes/footer.php'; ?>

login.php

<?php

session\_start();

// Redirect if already logged in

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

header("Location: index.php");

exit();

}

require\_once 'config/database.php';

require\_once 'classes/User.php';

$database = new Database();

$db = $database->getConnection();

$user = new User($db);

$message = '';

$messageType = '';

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

$username = $\_POST['username'] ?? '';

$password = $\_POST['password'] ?? '';

if (empty($username) || empty($password)) {

$message = "Please fill in all fields";

$messageType = "error";

} else {

if ($user->login($username, $password)) {

// Set session variables

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

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

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

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

// Redirect to dashboard/home page

header("Location: index.php");

exit();

} else {

$message = "Invalid username or password";

$messageType = "error";

}

}

}

include 'includes/header.php';

?>

<div class="form-container">

<h2>Login</h2>

<?php if ($message): ?>

<div class="alert alert-<?php echo $messageType; ?>">

<?php echo $message; ?>

</div>

<?php endif; ?>

<form action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" method="post">

<div class="form-group">

<label for="username" class="form-label">Username</label>

<input type="text" name="username" id="username" class="form-input"

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

</div>

<div class="form-group">

<label for="password" class="form-label">Password</label>

<input type="password" name="password" id="password" class="form-input" required>

</div>

<button type="submit" class="btn btn-primary">Login</button>

</form>

<p class="form-footer">

Don't have an account? <a href="register.php">Register here</a><br>

<a href="forgot-password.php">Forgot Password?</a>

</p>

</div>

<?php include 'includes/footer.php'; ?>

register.php

<?php

session\_start();

require\_once 'config/database.php';

require\_once 'classes/User.php';

$database = new Database();

$db = $database->getConnection();

$user = new User($db);

$message = '';

$messageType = '';

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

// Validate input

$errors = [];

if (empty($\_POST['username'])) {

$errors[] = "Username is required";

} elseif ($user->username = $\_POST['username'] && $user->usernameExists()) {

$errors[] = "Username already exists";

}

if (empty($\_POST['email'])) {

$errors[] = "Email is required";

} elseif (!filter\_var($\_POST['email'], FILTER\_VALIDATE\_EMAIL)) {

$errors[] = "Invalid email format";

} elseif ($user->email = $\_POST['email'] && $user->emailExists()) {

$errors[] = "Email already exists";

}

if (empty($\_POST['password'])) {

$errors[] = "Password is required";

} elseif (strlen($\_POST['password']) < 6) {

$errors[] = "Password must be at least 6 characters";

}

if ($\_POST['password'] !== $\_POST['confirm\_password']) {

$errors[] = "Passwords do not match";

}

if (empty($\_POST['first\_name'])) {

$errors[] = "First name is required";

}

if (empty($\_POST['last\_name'])) {

$errors[] = "Last name is required";

}

// If no errors, proceed with registration

if (empty($errors)) {

$user->username = $\_POST['username'];

$user->password = $\_POST['password'];

$user->email = $\_POST['email'];

$user->first\_name = $\_POST['first\_name'];

$user->last\_name = $\_POST['last\_name'];

$user->phone = $\_POST['phone'];

if ($user->create()) {

$message = "Registration successful! Please login.";

$messageType = "success";

} else {

$message = "Registration failed. Please try again.";

$messageType = "error";

}

} else {

$message = implode("<br>", $errors);

$messageType = "error";

}

}

include 'includes/header.php';

?>

<div class="form-container">

<h2>Create Account</h2>

<?php if ($message): ?>

<div class="alert alert-<?php echo $messageType; ?>">

<?php echo $message; ?>

</div>

<?php endif; ?>

<form action="<?php echo htmlspecialchars($\_SERVER["PHP\_SELF"]); ?>" method="post">

<div class="form-group">

<label for="username" class="form-label">Username</label>

<input type="text" name="username" id="username" class="form-input"

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

</div>

<div class="form-group">

<label for="email" class="form-label">Email</label>

<input type="email" name="email" id="email" class="form-input"

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

</div>

<div class="form-group">

<label for="first\_name" class="form-label">First Name</label>

<input type="text" name="first\_name" id="first\_name" class="form-input"

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

</div>

<div class="form-group">

<label for="last\_name" class="form-label">Last Name</label>

<input type="text" name="last\_name" id="last\_name" class="form-input"

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

</div>

<div class="form-group">

<label for="phone" class="form-label">Phone Number</label>

<input type="tel" name="phone" id="phone" class="form-input"

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

</div>

<div class="form-group">

<label for="password" class="form-label">Password</label>

<input type="password" name="password" id="password" class="form-input" required>

</div>

<div class="form-group">

<label for="confirm\_password" class="form-label">Confirm Password</label>

<input type="password" name="confirm\_password" id="confirm\_password" class="form-input" required>

</div>

<button type="submit" class="btn btn-primary">Register</button>

</form>

<p class="form-footer">

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

</p>

</div>

<?php include 'includes/footer.php'; ?>

**References**

[Ghost Together]. (2022, August 7). How To Run PHP Code in Visual Studio Code on Mac | VSCode | Vs code | Localhost Xampp [Video]. YouTube. <https://www.youtube.com/watch?v=aHIuIaxEU4Y>

Shah, V. (2024, February 29). *Building Dynamic Websites with PHP and MySQL*. WPWEB Infotech. <https://wpwebinfotech.com/blog/php-and-mysql/>

Nixon, R. (2012). *Learning PHP, MySQL, JavaScript, and CSS: A step-by-step guide to creating dynamic websites*. " O'Reilly Media, Inc.".