Week 3: Landing, Login, and Enrollment Pages Development

Nkosinomusa Gwalla

CST499: Capstone for Computer Software Technology

Professor Joseph Rangistch

April 7, 2025

Introduction

This paper outlines the development of a dynamic PHP and MySQL-based web application built using XAMPP for local hosting. The goal for Week 3 was to create the foundational components of a user-authenticated website, including a landing page, login page, and registration page. Each component integrates with a backend MySQL database using a custom PHP database class. The interface is styled with Bootstrap for mobile responsiveness and professional design. The project demonstrates the implementation of secure password storage, input validation, and PHP session handling. This report includes source code explanations and serves as a foundation for future application features. The purpose of this paper is to illustrate how core login and registration features were successfully built and tested in a local development environment.

Running PHP Files in XAMPP

To run PHP files on XAMPP:  
1. Launch the XAMPP Control Panel and start Apache and MySQL.  
2. Save all project files in `C:\xampp\htdocs\Week3\_php\_files\`.  
3. Visit `http://localhost/Week3\_php\_files/` in your browser.

This setup allows testing of PHP scripts and database queries in a safe local environment before deployment.

A screenshot of a computer

AI-generated content may be incorrect.

Website Pages Implemented

Landing Page (index.php)

This page checks session variables to determine if the user is logged in. It offers navigation to the login and registration pages or greets the logged-in user.

A white background with black text

AI-generated content may be incorrect.Login Page (login.php)

Users input their credentials. If verified using PHP’s `password\_verify()` function against hashed values in the database, a session is started and they are redirected.

A screenshot of a computer

AI-generated content may be incorrect.  
A screenshot of a computer

AI-generated content may be incorrect.

Registration Page (register.php)

New users fill out a form that validates username and password input. If validation passes and the username is unique, the password is hashed and saved to the `users` table.

A screenshot of a computer

AI-generated content may be incorrect.  
A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a login form

AI-generated content may be incorrect.

MySQL Database and Users Table

The `users` table was created to store user credentials:

CREATE TABLE users (  
 id INT AUTO\_INCREMENT PRIMARY KEY,  
 username VARCHAR(50) NOT NULL UNIQUE,  
 password VARCHAR(255) NOT NULL,  
 created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP  
);

The table stores hashed passwords using `password\_hash()` and ensures that usernames are unique.

A screenshot of a computer

AI-generated content may be incorrect. A screenshot of a computer

AI-generated content may be incorrect.

Custom Database Connection Class (Database.php)

A PHP class was created to manage MySQLi database connections. This class centralizes the connection logic, improves code reusability, and supports prepared statements:

<?php  
class Database {  
 private $conn;

public function \_\_construct($host = "localhost", $user = "root", $pass = "", $dbname = "mydatabase") {  
 $this->conn = new mysqli($host, $user, $pass, $dbname);  
 if ($this->conn->connect\_error) {  
 die("Connection failed: " . $this->conn->connect\_error);  
 }  
 }

public function prepare($sql) {  
 return $this->conn->prepare($sql);  
 }

public function query($sql) {  
 return $this->conn->query($sql);  
 }

public function close() {  
 $this->conn->close();  
 }  
}  
?>

A screenshot of a computer program

AI-generated content may be incorrect.Process of Creating and Using Registration Page

1. Form Layout: A Bootstrap-styled form collects username and password input.  
2. Validation: PHP checks for empty fields and ensures password is at least six characters.  
3. Uniqueness Check: A SELECT query ensures the username doesn’t already exist.  
4. Password Hashing: The password is hashed with `password\_hash()` before insertion.  
5. Database Insert: A prepared INSERT query saves the user to the `users` table.

This process ensures a secure and user-friendly registration experience.

Conclusion

The Week 3 development phase successfully delivered a functional PHP web application with user login and registration capabilities. The pages interact with a MySQL database using prepared statements and a custom database class. Passwords are securely stored, and input validation ensures reliable user interaction. These foundational features are critical for expanding the application in future weeks. Screenshots and structured code demonstrate the use of modern security practices and responsive design principles.

References

Elmasri, R., & Navathe, S. B. (2017). \*Fundamentals of database systems\* (7th ed.). Pearson.

Schach, S. R. (2011). \*Essentials of software engineering\* (3rd ed.). McGraw-Hill Education.

PHP Documentation. (2023). password\_hash - PHP Manual. https://www.php.net/manual/en/function.password-hash.php

Tutorial Republic. (n.d.). PHP MySQL Login System Tutorial. https://www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php

HTTPLocalhost Info. (2024). Introduction to XAMPP. http://httplocalhost.info/xampp