Software Requirements Specification

for

< Landing, Login, and Enrollment

Pages Development >

Version 1.0 approved

Prepared by <Nikko Amaya>

<UAGC CST499 >

<Capstone for Computer Software Technology>

<Prof. Charmelia Butler >

<8/25/25>

**Building a PHP Registration System with XAMPP and MySQL**

**Introduction**

This project centers on the development of a student registration system where users can create accounts in order to enroll in classes. The system is designed to provide students with a secure and streamlined way to register, log in, and access available course offerings. The implementation makes use of PHP for server-side scripting and MySQL for database management, all hosted in a local development environment through XAMPP. The project includes the creation of a landing page, a login page, and a registration page, along with the underlying database tables and PHP logic required to manage student accounts. This paper provides a comprehensive explanation of the steps taken to design and implement the system, including establishing the server environment, developing the database schema, and creating a functional registration page capable of saving user information into the database.

**Running PHP Files in XAMPP**

To execute PHP files, a local server environment is required. XAMPP provides this environment by integrating Apache, MySQL, PHP, and phpMyAdmin into one package. After installation, the Apache and MySQL modules must be activated from the XAMPP Control Panel. Once active, PHP files are placed into the htdocs directory of the XAMPP installation folder. For this project, a folder named student\_portal was created inside htdocs, containing all relevant PHP and HTML files. To run the project, the browser is directed to http://localhost/student\_portal/filename.php. This setup allows the system to mimic a live server environment, enabling immediate testing and debugging of the PHP scripts and database connections.

**Creating the Landing Page, Login Page, and Registration Page**

The landing page serves as the first point of entry for the system, giving users a clear overview of available options and navigation links to register or log in. The login page allows existing students to access their accounts by validating credentials against stored records in the database. The registration page is specifically designed for new users who wish to enroll in classes, and it collects essential personal information such as full name, email, phone number, and password. Each page was developed with structured HTML and enhanced with minimal CSS for clarity and ease of navigation. The design separates responsibilities: guiding new students to create accounts, allowing returning users to log in, and funneling all users through the landing page to maintain a consistent workflow.

**Creating the MySQL Database and Tables**

The foundation of the registration system lies in the database. Using phpMyAdmin, a database named student\_portal was created to hold all user-related information. Within it, a users table was built to store student details. The fields included an auto-incremented id as the primary key, fullname, email, phone, and password. Each field was carefully defined with data types to ensure proper storage and retrieval, such as VARCHAR for text fields and INT for the primary key. Constraints were applied where necessary, such as making the email field unique to prevent duplicate accounts. This structure ensures that the database maintains data integrity while supporting secure and efficient operations.

**MySQL Database Functions and the Custom Connection Class**

To establish communication between the PHP application and the MySQL database, the mysqli extension was used. Functions such as mysqli\_connect(), mysqli\_query(), and mysqli\_error() played a key role in opening connections, running SQL statements, and capturing errors. To simplify repeated operations and ensure consistency across multiple pages, a custom database connection class was implemented. This class handled the connection setup and provided methods for executing queries, centralizing database logic in one place. If an error occurred during the connection, the class returned meaningful feedback, allowing for smoother debugging. This modular approach improved maintainability and ensured the project followed best practices in software development.

**Developing the Registration Page Layout**

The registration page was designed with both usability and accessibility in mind. It includes input fields for full name, email, phone number, and password, each labeled for clarity. The form also contains a “Register” button and a link to the login page for students who already have accounts. HTML5 attributes such as required and type="email" were used to enforce correct data formats before submission. Minimal CSS styling was added to align fields, create spacing, and ensure a professional appearance. By keeping the layout simple and intuitive, students can register with minimal confusion while the system ensures valid input.

**Developing the Registration Page PHP Source Code**

The backend PHP code for the registration page processes form submissions and inserts new student data into the users table. Upon submission, the script validates all input fields on the server side to ensure none are empty and that formats are correct. For security, the password is encrypted using PHP’s password\_hash() function before being saved in the database. After validation, the script executes an INSERT query to add the student’s information to the users table. Any errors, such as attempting to register with an already existing email, are caught and returned as meaningful messages to the user. This ensures that the system not only saves data securely but also provides feedback to enhance user experience.

**Developing the User Information Table in the Database**

The users table was designed with a schema tailored for registration data:

* id – INT, AUTO\_INCREMENT, PRIMARY KEY
* fullname – VARCHAR(100)
* email – VARCHAR(100), UNIQUE
* phone – VARCHAR(15)
* password – VARCHAR(255)

The choice of VARCHAR(255) for the password field ensures compatibility with encrypted values generated by password\_hash(). The combination of primary keys, unique constraints, and adequate field sizes ensures data is stored securely, efficiently, and without duplication.

**Steps Taken to Create the Registration Page and Save User Information**

The development of the registration functionality followed a structured workflow. First, the HTML form was created with fields for student details. Second, PHP scripts were written to handle form submission, validate the data, and secure the password using hashing. Third, a database connection was established through the custom connection class, and SQL queries were executed to insert the data. Fourth, error handling mechanisms were implemented to prevent duplicate registrations and provide user feedback. Finally, successful tests were conducted by registering multiple students and verifying that their information was correctly stored in the users table. This systematic approach ensured functionality, usability, and security.

**Conclusion**

The student registration system developed for this project demonstrates the integration of PHP, MySQL, and XAMPP in building a functional web application. By creating a landing page, login page, and registration page, students are given clear pathways to create accounts and enroll in classes. The database schema, supported by constraints and secure password hashing, ensures safe storage of student data. The custom database connection class improved maintainability, while the registration page design emphasized simplicity and usability. Together, these elements formed a secure, efficient, and user-friendly system. The methods applied here can be extended to more complex student information systems, course scheduling platforms, or other web applications requiring secure user registration and data handling.

**Developed Screenshots**

A screenshot of a computer

AI-generated content may be incorrect.

A blue and white screen with white text

AI-generated content may be incorrect.

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 computer

AI-generated content may be incorrect.

**DATABASE TABLES**

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 computer

AI-generated content may be incorrect.**

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 computer

AI-generated content may be incorrect.

**SOURCE CODE SCREEN SHOTS**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer screen

AI-generated content may be incorrect.**

A screenshot of a computer program

AI-generated content may be incorrect.

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 computer program

AI-generated content may be incorrect.A white and blue screen

AI-generated content may be incorrect.

**REFERENCES**

Tsui, F., Karam, O., & Bernal, B. (2018). [*Essentials of software engineering*](https://uagc.instructure.com/courses/149470/modules/items/7628542) (4th ed.). Jones & Bartlett Learning.

ExamSnap. (n.d.). *How to run PHP files locally using XAMPP: Step-by-step tutorial for beginners*. <https://www.examsnap.com/certification/how-to-run-php-files-locally-using-xampp-step-by-step-tutorial-for-beginners/>

MySQL. (n.d.). *13.2.10 USE statement*. In *MySQL 8.0 reference manual*. Oracle. Retrieved August 25, 2025, from <https://dev.mysql.com/doc/refman/8.0/en/database-use.html>