Software Requirements Specification

for

< Database Development

and Class Registration >

Version 1.0 approved

Prepared by <Nikko Amaya>

<UAGC CST499 >

<Capstone for Computer Software Technology>

<Prof. Charmelia Butler >

<9/1/25>

**Implementation Phase of the Student Registration System**

**Introduction**

The implementation phase of the Student Registration System marks a critical step in translating design artifacts and requirements into a functioning application environment. In the previous weeks, foundational components such as the system design, requirements specification, and the initial registration and login pages were developed and tested to ensure successful integration with the XAMPP environment. Building upon that foundation, this phase emphasizes the full development of the database schema, connection between the user interface and the backend, and the creation of essential functions such as class registration, course listing, course addition, and course deletion. This paper provides an overview of the work completed in this phase, detailing the creation of database tables, the integration of the backend, the development of the necessary PHP pages, and reflections on the overall experience of carrying out the implementation process.

**Revision of Previous Work**

Before advancing further into development, the primary focus of any revisions made was to improve code organization and ensure consistency between the landing page, login functionality, and registration page. Minor structural changes were introduced to the registration form, particularly around field validation and error handling, to ensure data integrity. Additionally, the database connection class was refined to provide more robust error messages, which streamlined debugging. These revisions created a stable base from which the remaining functions of the project could be built.

**Creation of Database Tables**

The next major step in this phase was the creation of the required MySQL tables that supported the full system design. In addition to the user authentication table built earlier, new tables were designed and implemented to represent courses, student schedules, and registration records. The courses table stores essential details such as course ID, course name, capacity, and enrollment count. The registrations table links students to their selected courses, ensuring relational integrity by referencing both the user ID and the course ID. The schema was created with normalization in mind to minimize redundancy and maintain efficiency in data management. Screenshots of these tables before and after data manipulation operations demonstrate the accuracy of the implementation and confirm that the database is functioning as intended.

**Backend Database Connectivity**

Once the tables were created, each PHP page was connected to the backend through the database connection class. This step ensured that user inputs on the frontend pages were consistently captured and stored in the MySQL database. The database connection class was designed with reusable functions, allowing for consistent queries across multiple pages such as class registration, schedule listing, and course deletion. This architecture promotes maintainability and scalability, enabling future system enhancements with minimal code duplication.

**Class Registration Functionality**

The most critical feature implemented in this phase was the ability for students to register for classes. This functionality integrates front end user inputs with backend logic to validate course availability and prevent enrollment once a course reaches its maximum capacity. The system also checks for duplicate enrollments, ensuring that a student cannot register for the same course more than once. By combining input validation, database queries, and session management, the registration page provides a smooth and accurate process for students to enroll in their selected courses. Screenshots of the database before and after registration illustrate how records are properly inserted into the registrations table, updating course enrollment counts in real time.

**Listing Registered Classes**

Another core feature developed was the ability for students to view the courses they are currently registered for. This function queries the database and displays relevant course details in a user-friendly format. The listing page dynamically retrieves data from the registrations and courses tables, ensuring that users always see up-to-date information about their enrollment. This page is essential for transparency and allows students to confirm their schedules without relying on manual verification.

**Adding and Deleting Courses**

To provide flexibility, additional functionality was implemented to allow students to add more courses to their schedule or remove classes they no longer wish to attend. The add function mirrors the registration process but enables subsequent course selection for a given semester. The delete function, on the other hand, carefully removes the course record from the student’s schedule while decrementing the course enrollment count. This ensures that the course capacity is accurately reflected, allowing other students on the waitlist to be notified if a seat becomes available. Database screenshots captured before and after deletion operations illustrate the correct decrement in enrollment counts and the removal of registration records.

**Experience with the Implementation Phase**

The implementation phase of the Student Registration System presented both challenges and valuable learning opportunities. One of the most significant aspects of this stage was bridging the gap between theoretical design artifacts and practical execution in code. While creating tables and basic forms was straightforward, ensuring that each function integrated seamlessly with the backend required careful debugging and testing. Incremental testing was conducted after each feature was developed, which minimized the complexity of error resolution. Additionally, the modular structure of the database connection class and the use of consistent query methods highlighted the importance of reusable code.

Another key takeaway was the importance of capturing database states through screenshots before and after operations. This documentation provided concrete evidence of functionality and facilitated a clear understanding of how queries affected the stored data. Furthermore, the iterative process of implementing registration, listing, adding, and deleting courses reinforced the importance of user experience, ensuring that each function was intuitive and efficient. Overall, this phase emphasized the practical application of database design, backend integration, and software development best practices.

**Conclusion**

The implementation phase successfully translated the system’s design and requirements into a fully functional Student Registration System capable of supporting user authentication, course registration, schedule management, and database integration. By creating and linking the necessary MySQL tables, establishing backend connectivity, and developing features such as registration, listing, adding, and deleting courses, the system now provides a comprehensive environment for managing student enrollments. The process highlighted the importance of revising prior work based on feedback, documenting operations through screenshots, and building scalable code. Through this phase, the project demonstrated how design principles and requirements evolve into a working solution, marking an essential milestone in the development of the Student Registration System.

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 screen

AI-generated content may be incorrect.

**Canceling an Enrollment**

A screen shot of a computer

AI-generated content may be incorrect.

A blue and white text

AI-generated content may be incorrect.

A screenshot of a computer program

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

AI-generated content may be incorrect.

**Student Notification Page**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A blue and white screen with text

AI-generated content may be incorrect.**

**A blue and white screen with black text

AI-generated content may be incorrect.**

A screenshot of a computer

AI-generated content may be incorrect.

A computer screen shot of text

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>