Database Development and Class Registration

Carlon Bullock

The University of Arizona Global Campus

CST 499: Capstone for Computer Software Technology

Professor: Joseph Rangitsch

May 5, 2025

**Introduction**

Developing a dynamic course registration platform is a multi-phased challenge that demands technical acumen and adaptability to evolving software engineering practices. Throughout this process, I built upon earlier foundational coursework, notably CST310, and returned to tools and technologies that once felt daunting—such as MAMP, MySQL, and PHP. Working on macOS, I again utilized MAMP due to its seamless compatibility with Apache and MySQL on a Mac-based system. Unlike previous experiences where setup and connectivity issues stalled progress, navigating these environments this time was more intuitive and deliberate, reinforcing how far I've come to understand the full-stack development lifecycle.

This implementation phase represents a culmination of the system design work documented in earlier weeks, particularly the software requirements specification (SRS) and UML modeling. It focuses on deploying functional backend components such as database table creation, user session handling, and class registration workflows. These deliverables reflect core principles of implementation from the software engineering lifecycle, including consistency, reusability, and maintainability (Tsui, Karam, & Bernal, 2022, Ch. 9). Additionally, validation through manual SQL inspection and live browser testing reinforces the alignment between specification and behavior—a key emphasis in modern QA practices (Tsui et al., 2022, Ch. 10). This paper documents the full implementation of an online course registration system, outlining the creation of backend database tables, user interaction pages, dynamic class registration and deletion, and the supporting logic that ties the system together through secure and maintainable code.

**Revisions from Week 3 Implementation**

While the core logic of the registration system remained consistent with the Week 3 design, several technical adjustments were made during the implementation phase to improve functionality, maintainability, and cross-platform compatibility. One of the first necessary revisions involved transitioning all .html files to .php format. This allowed server-side code execution within MAMP and enabled dynamic page rendering based on user session states. In particular, the landing page was converted from landing.html to landing.php and updated to display navigation options conditionally for authenticated users. The final version gave users links to register for classes, view their current enrollments, or drop a class if they are already logged in. At the same time, unauthenticated visitors were only given options to log in or register a new account.

Another essential revision was aligning the file structure to meet MAMP's URL access rules. Because MAMP interprets project folders under htdocs/, the initial folder name (ocres\_project) required normalization and proper linking using URL-encoded paths (e.g., http://localhost:8888/Ocres%20Project/landing.php). Adjustments like these reflect the implementation-level detail that must align with deployment requirements—a central takeaway from the textbook's importance on environment-aware configuration and early validation of runtime context (Tsui, Karam, & Bernal, 2022, Ch. 9). Furthermore, a significant structural revision was the creation of a centralized configuration file (db\_config.php) for database connections using PHP Data Objects (PDO). This move reduced duplication across all scripts interacting with MySQL, enforced stronger security via prepared statements, and reflected the maintainability and scalability principles discussed in Chapter 9 of the course text. Though not required by the initial SRS, these improvements demonstrate the iterative nature of software implementation and the value of refactoring even in small-scale projects.

A screen shot of a computer

AI-generated content may be incorrect.

Figure 1. Landing page with dynamic session-based links

**MySQL Tables**

The Online Course Registration and Enrollment System (OCRES) database design consists of three main tables: users, classes, and registrations. These tables were created per standard normalization principles to avoid redundancy, enforce data integrity, and simplify query logic (Tsui, Karam, & Bernal, 2022, Ch. 9). Together, they support the key user workflows identified in the SRS: registration, login, class enrollment, and class withdrawal. The users' table stores personal and login credentials for each student. It includes fields such as name, email, phone, and a securely hashed password. Passwords are encrypted using PHP's password\_hash() function, aligning with best practices in secure credential storage.

The class table defines the academic courses offered. Each entry includes a unique class\_id, a course\_code, a course\_name, an instructor, and the schedule. These fields provide all the necessary information for a student to evaluate and enroll in a class. The registrations table is a junction table to record many-to-many relationships between students and classes. Each row maps a user\_id to a class\_id, effectively representing a student's enrollment in a specific course. This schema supports efficient querying for a student's schedule or a class roster, and its simplicity ensures straightforward maintenance and scalability (Tsui et al., 2022, Ch. 9). **Figure 2** shows the database structure, and **Figure 3** displays the classes table populated with sample data.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 2. Complete database schema with all three tables

A screenshot of a computer program

AI-generated content may be incorrect.

Figure 3. Classes table with populated data

**Feature Implementation**

The system's feature set was implemented through individual PHP scripts interacting with the MySQL database using prepared statements and session-based user management. Each page fulfills a specific aspect of the user experience outlined in the original software requirements specification, including class enrollment, viewing schedules, and withdrawing from classes.

**Register for Classes**

The register\_class.php page allows logged-in users to browse available classes and select one to enroll in. The course list is dynamically generated using a SELECT \* FROM classes query. Upon submission, the system uses a prepared INSERT INTO registrations query to associate the user\_id with a class\_id. Once a class is selected and registered, the confirmation message confirms that the operation was successful. This feature reflects the structured, modular programming style advocated in Chapter 9 of the course text, where each function is responsible for a clearly defined task, and input validation ensures robustness (Tsui, Karam, & Bernal, 2022). Security was enforced through PDO and parameter binding, preventing SQL injection even if the form is tampered with.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4. Class registration page after successful submission

**View Registered Classes**

The my\_classes.php page displays all the courses a user has registered for by joining the classes and registrations tables on class\_id. A friendly message is returned if no classes are found for the user. This functionality reinforces dynamic content rendering principles and demonstrates a black-box testing strategy: the user provides no direct input, but the output must reflect the state of the database based on prior user actions (Tsui et al., 2022, Ch. 10).

A screenshot of a computer

AI-generated content may be incorrect.

Figure 5. List of registered classes shown dynamically

**Drop a Class**

The drop\_class.php page completes the student interaction cycle by allowing users to remove a registered class. A simple DELETE FROM registrations WHERE user\_id =? AND class\_id = ? query is executed when the user selects a class to drop. Once dropped, the updated class list reflects the change, and the user is notified. This feature aligns with the system's maintainability goals using shared configuration (db\_config.php) and session logic. It also aligns with the system testing stage of software quality assurance by verifying end-to-end data deletion (Tsui et al., 2022, Ch. 10).

A screenshot of a computer

AI-generated content may be incorrect.

Figure 6. Drop class interface with selectable courses

**Registration Table: Before and After Operations**

To validate the effectiveness of the class registration and drop logic, the registrations table was reviewed before and after each core operation. This table functions as the relational glue between users and classes, recording each unique registration event. A manual SQL INSERT was used during early testing to simulate enrollment before the interface logic was finalized. Once the system pages were functional, live user registration via the web interface updated this table seamlessly.

**Before Class Registration**

Initially, the registrations table was empty, indicating no students enrolled in any course. This clean state provided a baseline for testing and aligned with unit testing practices, where test cases begin with a predictable setup (Tsui, Karam, & Bernal, 2022, Ch. 10).

A screenshot of a computer

AI-generated content may be incorrect.

Figure 7. Empty the registrations table before class selection

**After Class Registration**

A new row appeared in the registrations table after a student selected a course and submitted the registration form. This row included a valid user\_id and class\_id, confirming that the many-to-many relationship between students and courses was functioning correctly. This operation passed the system-level test by validating front-end interaction and backend data handling.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 8. Registrations table populated after successful enrollment

**Implementation Experience Summary**

Implementing the OCRES project marked a noticeable turning point in my development confidence. Early in the software engineering program, even basic PHP and SQL integration felt overwhelming, especially within local environments like MAMP. However, through this project, those once-daunting tasks became a matter of structured logic and repeatable problem-solving. Configuring the environment, troubleshooting session variables, managing relational keys in MySQL, and ensuring form data security felt more like the expected course of software development rather than roadblocks.

One of the most valuable lessons reinforced during implementation was the role of prepared statements for secure database communication. Using PHP Data Objects (PDO) improved the system's maintainability through shared connection logic (db\_config.php) and eliminated potential SQL injection vulnerabilities. Likewise, password\_hash() and password\_verify() elevated the project's credential handling to a level that aligns with industry standards. These implementations map directly to software quality metrics such as correctness, security, and maintainability—metrics accentuated throughout the software lifecycle in Tsui et al. (2022, Ch. 9).

From a testing standpoint, verifying user session behavior and database changes constituted an informal but effective acceptance testing strategy. By visually confirming that registration actions updated the registrations table and drop actions deleted the expected row, I gained a deeper understanding of system-level validation (Tsui et al., 2022, Ch. 10). In essence, each feature served not only as a product function but as a live test case demonstrating proper data flow. Generally, this project reinforced the value of modular code, consistent database schema design, and meaningful validation through user interaction. It highlighted how practical implementation gives weight to theoretical principles—closing the loop between what we design and what we deliver.

**Conclusion**

The final implementation of the Online Course Registration and Enrollment System (OCRES) represents the successful convergence of system design, database architecture, and secure web development practices. From user account creation and login functionality to dynamic course registration and withdrawal, each component was built and tested to reflect the specifications outlined in earlier project phases. Leveraging PHP with MySQL and MAMP as a local development environment allowed for seamless full-stack integration on macOS, and the modular structure of the codebase made testing, debugging, and enhancement both logical and repeatable.

Beyond the functionality, this project reinforced key principles of implementation and quality assurance. Secure password handling, session-based user management, and prepared statements aligned with modern best practices in backend development. Additionally, real-time database validation and iterative refinement mirrored core concepts from the course text, particularly the importance of maintainability, correctness, and traceability in the software lifecycle. Ultimately, this implementation phase did more than complete a technical build—it provided tangible evidence of growth as a software engineer, linking prior learning to practical, testable outcomes. The project has met the specified requirements and demonstrated the value of structured thinking, careful debugging, and thoughtful design in building reliable systems.

**References**

Gemino, A., Reich, B. H., & Serrador, P. (2020). Agile, traditional, and hybrid approaches to project success: Is hybrid a poor second choice? Project Management Journal, 51(2), 177–194. <https://doi.org/10.1177/8756972820973082>

Nweke, L. O., & Eze, O. P. (2023a). A holistic review on optimization strategies for C & K metrics in object-oriented testing. International Journal of Innovative Computing, 24(3), 154–165.

Nweke, L. O., & Eze, O. P. (2023b). A novel approach for software effort estimation using optimized C&K metrics. Journal of Systems and Software Engineering, 17(2), 88–97.

Tsui, F., Karam, O., & Bernal, B. (2022). Essentials of software engineering (5th ed.). Jones & Bartlett Learning.