**The Database Development and Class Registration Paper**

Raul Machuca

## CST 499: Capstone for Computer Software Technology

Charmelia Butler

June 23, 2025

This phase of development expands on the foundation built in Week 3 by completing the database design and implementing full class registration functionality. I created additional pages allowing students to register for, drop, and view their courses. This paper presents an enhanced database schema, updated PHP logic, and screenshots demonstrating successful system operations. The goal is to implement a functional and scalable course registration system that connects a PHP front-end with a MySQL database using secure programming techniques (Connolly & Hoar, 2022).

To address feedback, I added SQL definitions for all tables and improved the database connection class by incorporating try/catch error handling using PHP Data Objects (PDO). This allows the application to catch connection errors gracefully and improves debugging (PHP Manual, n.d.-a). I also revised the project to modularize code for form handling and ensured consistent commenting across all PHP files to improve maintainability.

function connectDB() {

    try {

        $connString = "mysql:host=localhost;dbname=cst499";

        $user = "root";

        $pass = "";

        $pdo = new PDO($connString, $user, $pass);

        $pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

        return $pdo;

    } catch (PDOException $e) {

        die($e->getMessage());

    } }

The updated relational schema separates students, available courses, and class enrollments. This structure supports a scalable many-to-many relationship model and adheres to normalization principles (Setiawan et al., 2018).

CREATE TABLE students (

student\_id VARCHAR(10) PRIMARY KEY,

first\_name VARCHAR(50) NOT NULL,

last\_name VARCHAR(50) NOT NULL,

phone VARCHAR(20),

email VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL

);

CREATE TABLE coursetable (

course\_id VARCHAR(50) NOT NULL PRIMARY KEY,

course\_name VARCHAR(100) NOT NULL,

seat\_limit INT NOT NULL,

seat\_available INT NOT NULL

);

CREATE TABLE student\_courses (

id INT AUTO\_INCREMENT PRIMARY KEY,

student\_id VARCHAR(10),

course\_id VARCHAR(200),

semester VARCHAR(255),

enrollment\_date TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY (student\_id) REFERENCES students(student\_id),

FOREIGN KEY (course\_id) REFERENCES coursetable(course\_id)

);

This design supports real-time seat management, efficient joins, and protects integrity through foreign keys (Connolly & Hoar, 2022).

students

A screenshot of a computer

AI-generated content may be incorrect.

coursetable

A screenshot of a computer

AI-generated content may be incorrect.

student\_course

A screenshot of a computer

AI-generated content may be incorrect.

Registration Page

The registration.php script uses secure password hashing via password\_hash() and input validation to store new student records in the students table (PHP Manual).

A screen shot of a student registration form

AI-generated content may be incorrect.

Login Page

The login.php script validates hashed passwords using password\_verify() and initializes a session for the logged-in student (PHP Manual).

A screen shot of a login

AI-generated content may be incorrect.

Course View & Registration

The view\_courses.php page lists all available courses by querying the course table. Students can register by submitting a form to enrollment.php, which inserts a new record in student\_courses and updates seat availability.

A screen shot of a student course registration

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

The schedule.php page retrieves and displays the student’s enrolled courses using an SQL JOIN between the student\_courses and the course table. This approach allows the system to show detailed course information linked to the logged-in student. The page improves usability by presenting this data in a clean, tabular format and gives students real-time visibility into their current enrollment status.

Screenshots before and after key operations validate the functioning of the backend logic and seat updates. These are captured from phpMyAdmin and demonstrate correct state transitions.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

List from student\_courses

A screenshot of a computer

AI-generated content may be incorrect.

Request to unenroll

A screen shot of a computer

AI-generated content may be incorrect.

Class dropped

A screen shot of a computer

AI-generated content may be incorrect.

The table was updated to show seat availability.

A screenshot of a computer

AI-generated content may be incorrect.

As I continued through the implementation phase, I encountered a few challenges that made me step back and reassess how I initially structured the database. One of the first things I noticed was that the students table included extra fields like address, which were unnecessary for this system. Keeping the table simple with just the student’s name, phone number, email, and password made the registration and login process easier and more focused on functionality. Another key point I realized was that the relationships between the data were not set up in a way that supported the system I was trying to build. At first, I did not have a table to manage student enrollments. Once I got to the part where I needed to show what classes a student was registered for and update how many seats were available, I understood that I needed a third table. That is when I created the student\_courses table, which made it possible to track enrollments properly and support a many-to-many relationship between students and classes. Figuring out how to adjust the seat count when a student enrolled or dropped a class was one of the more complex parts of this phase and one of the most valuable learning experiences. Working with Bootstrap made the front-end a lot more manageable. Setting up the buttons and forms was straightforward, and they worked well when sending data to the backend, especially when it came to actions like dropping a class. Reviewing the Bootstrap documentation made designing clean, functional layouts (Bootstrap Documentation) simple.

This phase pushed the project closer to full implementation by integrating secure student registration, user authentication, course browsing, and dynamic class enrollment features. The revised database schema now cleanly supports many-to-many relationships between students and courses, and includes timestamped enrollment records to track activity. The PHP logic and Bootstrap components offer a functional interface that communicates directly with the MySQL backend through secure and modular code. Building out and testing these features strengthened my skills in secure PHP development, relational database modeling, and full-stack web application design. These enhancements addressed prior design gaps and resulted in a more maintainable, scalable, and user-friendly system.

database.php

A screen shot of a computer program

AI-generated content may be incorrect.

enrollment.phpA screen shot of a computer program

AI-generated content may be incorrect.

registration.php

A screen shot of a computer program

AI-generated content may be incorrect.

schedule.php

A screen shot of a computer program

AI-generated content may be incorrect.

profile.php

A screen shot of a computer program

AI-generated content may be incorrect.

**References**

Bootstrap Documentation. (n.d.). Forms. Retrieved from <https://getbootstrap.com/docs/4.0/components/forms/>

Bootstrap Documentation. (n.d.). Tables. Retrieved from <https://getbootstrap.com/docs/4.0/content/tables/>

Connolly, R., & Hoar, R. (2022). *Fundamentals of web development* (3rd ed.). Pearson.

PHP Manual. (n.d.). foreach loop. Retrieved from <https://www.php.net/manual/en/control-structures.foreach.php>

PHP Manual. (n.d.). password\_hash – Create a password hash. Retrieved from <https://www.php.net/manual/en/function.password-hash.php>

PHP Manual. (n.d.). password\_verify – Verifies that a password matches a hash. Retrieved from <https://www.php.net/manual/en/function.password-verify.php>

PHP Manual. (n.d.). PDO prepared statements. Retrieved from <https://www.php.net/manual/en/pdo.prepared-statements.php>

PHP Manual. (n.d.). PDOStatement::fetch. Retrieved from <https://www.php.net/manual/en/pdostatement.fetch.php>

PHP Manual. (n.d.). session\_start – Start a new or resume an existing session. Retrieved from <https://www.php.net/manual/en/function.session-start.php>

TraversyMedia. (2019). *How to create a simple user registration system using PHP & MySQL* [Video]. YouTube. Retrieved from <https://www.youtube.com/watch?v=oVJ0anq8yLA&ab_channel=TraversyMedia>

Tutorial Republic. (n.d.). PHP MySQL login system. Retrieved from <https://www.tutorialrepublic.com/php-tutorial/php-mysql-login-system.php>

W3Schools. (n.d.). HTML5 tags. Retrieved from <https://www.w3schools.com/tags/default.asp>

Setiawan, R., Rasjid, Z. E., & Effendi, A. (2018). Design metric indicator to improve quality software development. Procedia Computer Science, 135, 616–623. https://doi.org/10.1016/j.procs.2018.08.225