**Assignment 4**

David Rutledge

The University of Arizona Global Campus

CST499: Capstone for Computer Software Technology

Charmelia Butler

January 20, 2025

**Registration App**

Implementation of the Registration App has provided valuable insight into full-stack web application development. The implementation has seen some challenges that have triggered some important learning experiences.

The first challenge was the creation of the database. The relationships between users, courses, enrollments, and waitlist tables required thoughtful planning to ensure proper data integrity. Using foreign keys and appropriate indexes were important factors in ensuring performance and data consistency requirements were met.

Security was a has been a primary concern throughout the development process. The Registration App features password hashing, input validation, session management, and form validation, (Novikava, 2025).

* Password Hashing: Prevents password theft and credential exposure
* Input Validation: Prevents injection attacks and protects business logic
* Session Management: Creates and terminates session, maintains user authentication without requiring password re-entry
* Form Validation: Ensures data integrity, prevents malicious data submissions.

The user interface design is focused on creating an intuitive experience for users. Common button configuration, functionality, and colors were used to create clear indicators for course availability. For example, there is a distinct separation between current courses, which can be enrolled in, and future courses, which cannot. The design is also responsive to suit various screen sizes.

Several challenges were revealed during implementation. The waitlist management required consideration of race conditions and data consistency.

**Race Management Function:**

public function processWaitlist($enrollment\_id) {

// Get the course\_id from the dropped enrollment

$query = "SELECT course\_id FROM enrollments WHERE enrollment\_id = :enrollment\_id";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(':enrollment\_id', $enrollment\_id);

$stmt->execute();

$course = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($course) {

// Start transaction to prevent race conditions

$this->conn->beginTransaction();

try {

// Get the next person on the waitlist

$query = "SELECT w.\*, u.email

FROM waitlist w

JOIN users u ON w.user\_id = u.user\_id

WHERE w.course\_id = :course\_id

AND w.status = 'Waiting'

ORDER BY w.position ASC

LIMIT 1

FOR UPDATE"; // Lock the row to prevent concurrent access

$stmt = $this->conn->prepare($query);

$stmt->bindParam(':course\_id', $course['course\_id']);

$stmt->execute();

$waitlist = $stmt->fetch(PDO::FETCH\_ASSOC);

if ($waitlist) {

// Update waitlist status to 'Notified'

$query = "UPDATE waitlist

SET status = 'Notified'

WHERE waitlist\_id = :waitlist\_id";

$stmt = $this->conn->prepare($query);

$stmt->bindParam(':waitlist\_id', $waitlist['waitlist\_id']);

$stmt->execute();

// Commit the transaction

$this->conn->commit();

}

} catch (Exception $e) {

// If anything goes wrong, rollback the changes

$this->conn->rollBack();

throw $e;

}

}

}

**Technical Growth**

This project has provided a variety of hands-on experiences. Here are some of the concepts covered in the development of the Registration App:

* PHP backend development
* MySQL database construction and management
* Session management
* Frontend design
* Security best practices
* Transaction management
* Row locking
* Ordered processing
* Row Locking

Figure 1.

A screenshot of a computer

AI-generated content may be incorrect.

users table with one entry.

Figure 2.

A screenshot of a computer

AI-generated content may be incorrect.

enrollments table with 2 classes added and one class dropped.

Figure 3.

A screenshot of a computer

AI-generated content may be incorrect.

Waitlist page.

Figure 5.

A screenshot of a computer

AI-generated content may be incorrect.

My Courses page showing 3 enrolled classes.

Figure 6.

A screenshot of a computer

AI-generated content may be incorrect.

My Courses page with confirmation modal.

<form method="POST" class="drop-form" onsubmit="return confirm('Are you sure you want to drop this course?');">

<input type="hidden" name="enrollment\_id" value="<?php echo $row['enrollment\_id']; ?>">

<button type="submit" name="drop" class="btn btn-danger">Drop Course</button>

</form>

Figure 7.

A screenshot of a computer

AI-generated content may be incorrect.

My Courses page reflecting dropped class.

Figure 8.

A screenshot of a web page

AI-generated content may be incorrect.

Course Registration page.

Figure 9.

A screenshot of a web page

AI-generated content may be incorrect.

Dashboard.

Figure 10.

A screenshot of a login page

AI-generated content may be incorrect.

**Scripts**

**dashboard.php**

A computer screen shot of a program code

AI-generated content may be incorrect.

A computer screen shot of a computer code

AI-generated content may be incorrect.A black background with white spots

AI-generated content may be incorrect.

**mycourses.php**

A computer screen with many colorful text

AI-generated content may be incorrect.

A computer screen with many colorful text

AI-generated content may be incorrect.A computer screen with many colorful text

AI-generated content may be incorrect.

**waitlist.php**

A computer screen with many colorful text

AI-generated content may be incorrect.

**References**

Novikava, A. (2025, January 8). *Top software development security best practices*. NordLayer. <https://nordlayer.com/blog/software-development-security-best-practices/>

Ak, A. F. (2024, September 8). *Understanding race conditions in software development*. DevOps.dev. <https://blog.devops.dev/understanding-race-conditions-in-software-development-805e771bb660>