CST499: Capstone for Computer Software Technology – Week 5 Final Project

Name: Latonya Chambers

The University of Arizona Global Campus

Course Number: GEN499: Capstone for Computer Software Technology

Instructor's Name: Dr. Butler

Date Submitted: December 4th, 2025

Week 5 - Final Project

GitHub Link:

Settings · @LadyChambers1's CST499 Final Project

Week one

Software Requirements Specification
for
Online Course Registration System (OCRS)

Version 1.0 approved

Prepared by Latonya Chambers

University of Arizona Global Campus

November 9, 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Latonya Chambers		Initial SRS creation	1.0

Introduction

Purpose

This Software Requirements Specification (SRS) defines the requirements for the Online Course Registration System (OCRS). The document describes all functional and non-functional requirements for the system's initial release, focusing on user registration, course listing, semester management, enrollment, and waiting list functionality.

Document Conventions

This SRS follows IEEE formatting standards. Functional requirements appear as numbered "REQ-#" items. Bold text is used for section headings; italics may be used for emphasis. All priorities are explicitly stated for each system feature as High (H), Medium (M), or Low (L).

Intended Audience and Reading Suggestions

This document is intended for developers, project managers, quality assurance testers, system designers, and stakeholders.
Readers should begin with Sections 1 and 2 for foundational understanding, proceed to Section 3 for functional requirements, and then review Sections 4 and 5 for interface and non-functional specifications.

Project Scope

The Online Course Registration System provides secure registration, login, course browsing, enrollment, waiting list functions, and cancellation features. The system streamlines course registration, reduces administrative effort, and enhances student access to course information.

References

- IEEE SRS Standard 830-1998
- Tsui, F., Karam, O., & Bernal, B. (2018). *Essentials of software engineering* (4th ed.). Jones & Bartlett Learning.

Overall Description

Product Perspective

The OCRS is a new, standalone web-based system. It will replace manual registration processes currently used by students. The system does not rely on existing institutional software but may integrate with future student information systems.

Product Features

High-level capabilities include:

- New user registration with unique ID verification
- User profile creation
- Secure login
- Browsing courses by semester
- Enrollment in available courses
- Waiting list management
- Automatic notifications when seats become available
- Enrollment cancellation

User Classes and Characteristics

- Students**:** Primary users; basic computer literacy, require intuitive interface.
- Administrators**:** Manage course offerings; higher system privileges.

Operating Environment

- Browser-based application
- Compatible with Windows, macOS, iOS, Android
- Requires stable internet connection
- Works with modern browsers (Chrome, Edge, Safari, Firefox)

Design and Implementation Constraints

- User IDs must be unique
- Password encryption required
- System must comply with data privacy regulations
- Maximum class capacities must be strictly enforced
- Waiting lists must follow FIFO order

User Documentation

The system will provide:

- Online user guide
- Registration tutorial
- FAQ section
- Contextual help within forms

Assumptions and Dependencies

- Users provide valid personal information
- Admins enter accurate course data
- Email service availability for notifications
- Stable hosting and server availability

System Features

User Registration and Profile Management

2.1.1 Description and Priority

Allows new users to create accounts and profiles.
Priority: **High**

2.1.2 Stimulus/Response Sequences

1. User selects “Register”
2. System prompts for unique ID, password, and required profile fields
3. System validates information
4. System stores profile and confirms registration

2.1.3 Functional Requirements

REQ-1: The system shall allow users to register with a unique ID and password.
REQ-2: The system shall prevent duplicate IDs.
REQ-3: The system shall require users to enter name, phone, and email.
REQ-4: The system shall store registration data securely.

Login and Authentication

External Interface Requirements

User Interfaces

- Clean, accessible web UI
- Standard buttons: Submit, Cancel, Help
- Error messages displayed near input fields
- Dashboard with courses, schedule, and notifications

Hardware Interfaces

- No direct hardware interface required
- Runs on any device capable of running a modern browser

Software Interfaces

- Email service for notifications
- Database system for storing user and course data

Communications Interfaces

- HTTPS protocol required
- SSL encryption for data transmission

Other Nonfunctional Requirements

Performance Requirements

- System must support at least 500 concurrent users
- Course list must load within 3 seconds

Safety Requirements

- Data backups performed automatically daily
- System should restore from backup in case of failure

Security Requirements

- Passwords encrypted in storage
- Access control enforced for admin functions
- Sensitive data protected through secure protocols

Software Quality Attributes

- Usability: Simple and intuitive
- Reliability: 99% uptime
- Maintainability: Modular code structure
- Portability: Works across major browsers and devices

Other Requirements

- System must comply with FERPA-like guidelines for student data
- All stored data must be backed up in compliance with institutional policy

Appendix A: Glossary

- OCRS: Online Course Registration System
- User: Student accessing the system
- Admin: Manager of course offerings
- ID: Unique login identifier
- FIFO: First In, First Out

Appendix B: Analysis Models

N/A

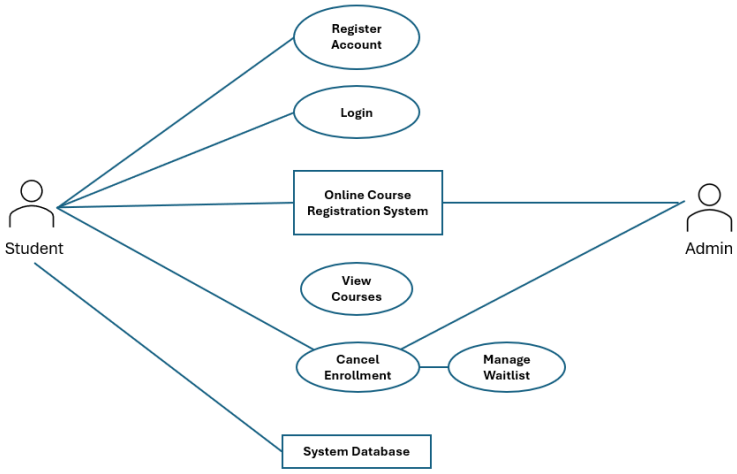
Week 2

1. Use Case Diagram

The Use Case Diagram identifies user interactions with the OCRS. Actors include Students, Administrators, and System Database. Primary use cases include Register Account, Login, View Courses, Enroll in Course, Cancel Enrollment, and Manage Waitlist.

Figure 1

Use Case Diagram



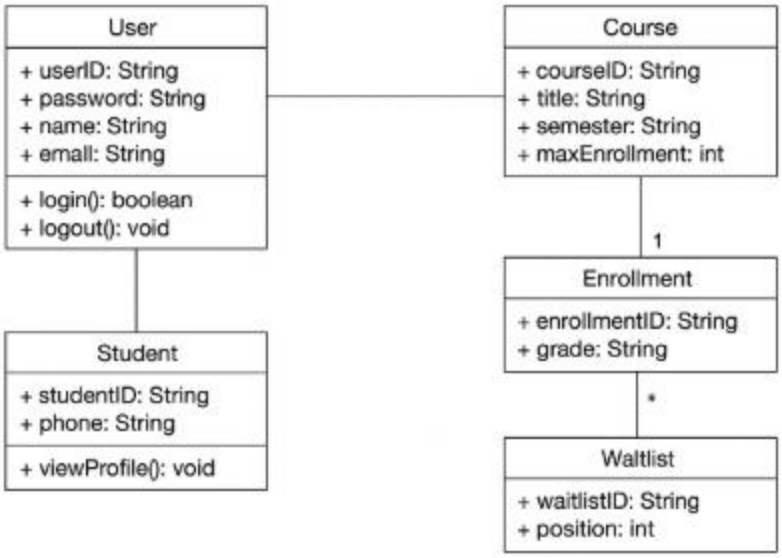
This diagram illustrates user-system relationships, clarifying system boundaries and helping ensure all functional requirements from the SRS are covered.

2. Class Diagram

The Class Diagram captures the system's static structure. Core classes include User, Student, Course, Enrollment, and Waitlist. The User class has attributes such as userID, password, name, and email. The Course class includes courseID, title, semester, and maxEnrollment. The Enrollment class associates users with courses, while the Waitlist class manages overflow requests.

Figure 2

Class Diagram



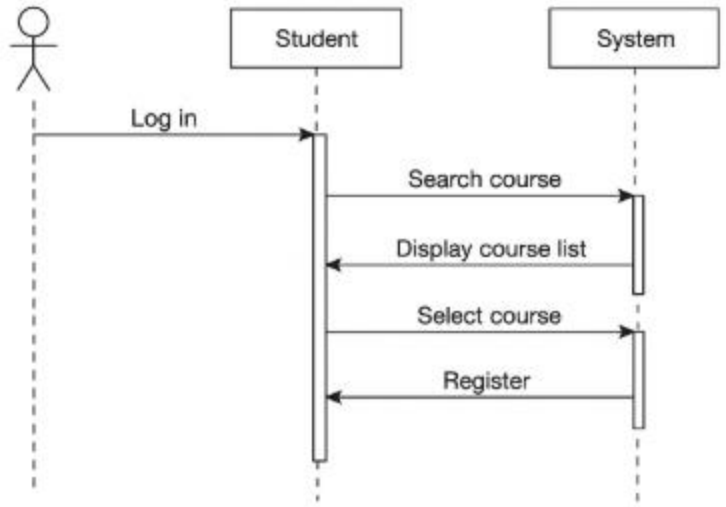
Relationships among these classes reflect one-to-many associations; for example, one course can have many enrolled students, and one waitlist can have multiple entries.

3. Sequence Diagram

The sequence diagram illustrates the dynamic flow when a user enrolls in a course. The student sends a request to CourseController, which interacts with Course and Database to verify availability. If the course is full, the system routes the user to the Waitlist.

Figure 3

Sequence Diagram



This diagram demonstrates how components communicate over time, emphasizing synchronous and asynchronous interactions.

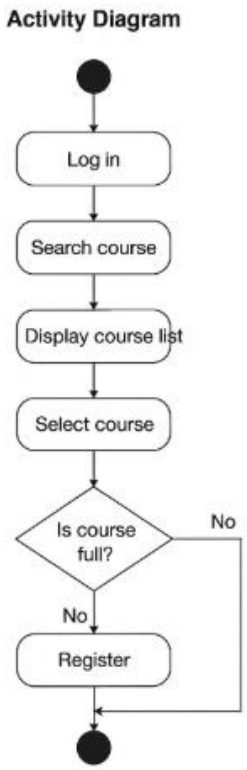
4. Activity Diagram

The activity diagram represents the workflow for course enrollment. Steps include:

1. Student logs in
2. System authenticates credentials
3. Student search for a course
4. If course is available → enroll
5. If full → add to waitlist
6. Confirmation message displayed

Figure 4

Activity Diagram



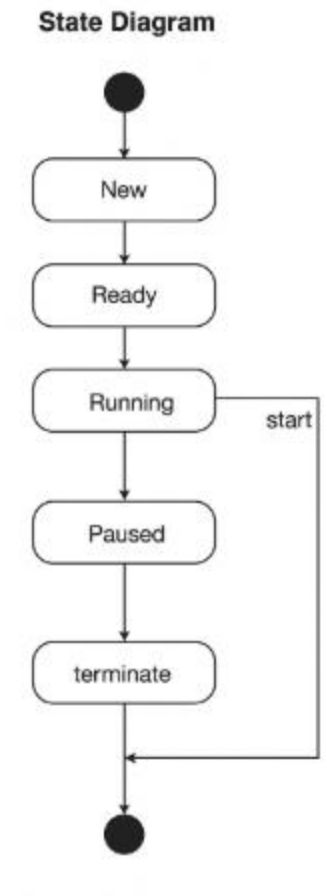
This model highlights decision points and parallel processes, ensuring the logical flow supports user requirements.

5. State Diagram

The State Diagram tracks a course's state transitions: Open, Full, Waitlisted, and Closed. When enrollment reaches capacity, the state changes to Full; if a student cancels, it transitions back to Open, notifying the next student on the waitlist.

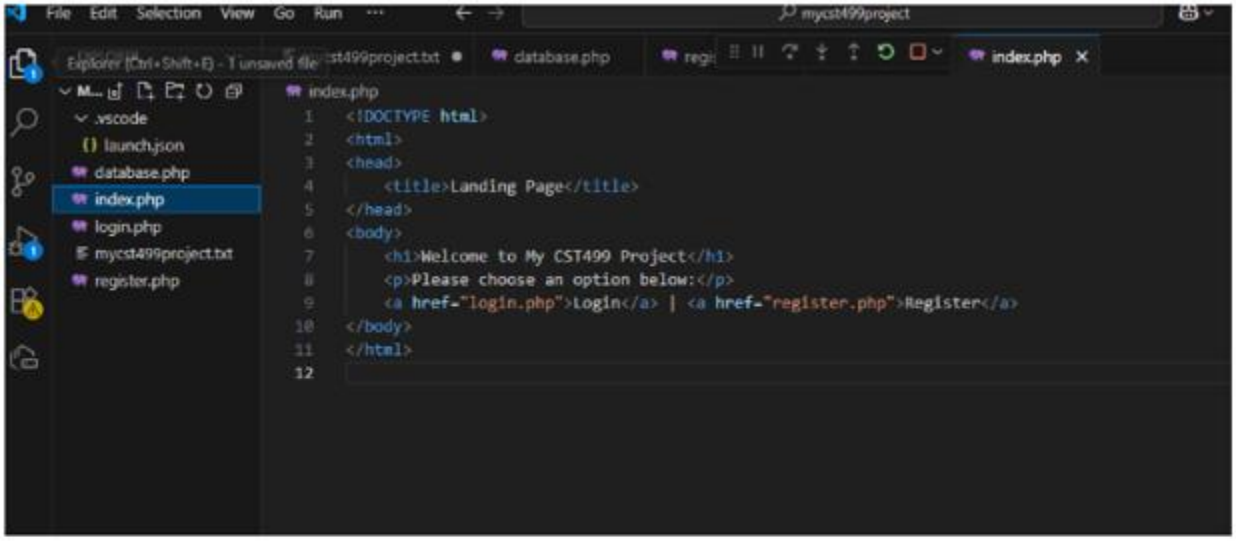
Figure 5

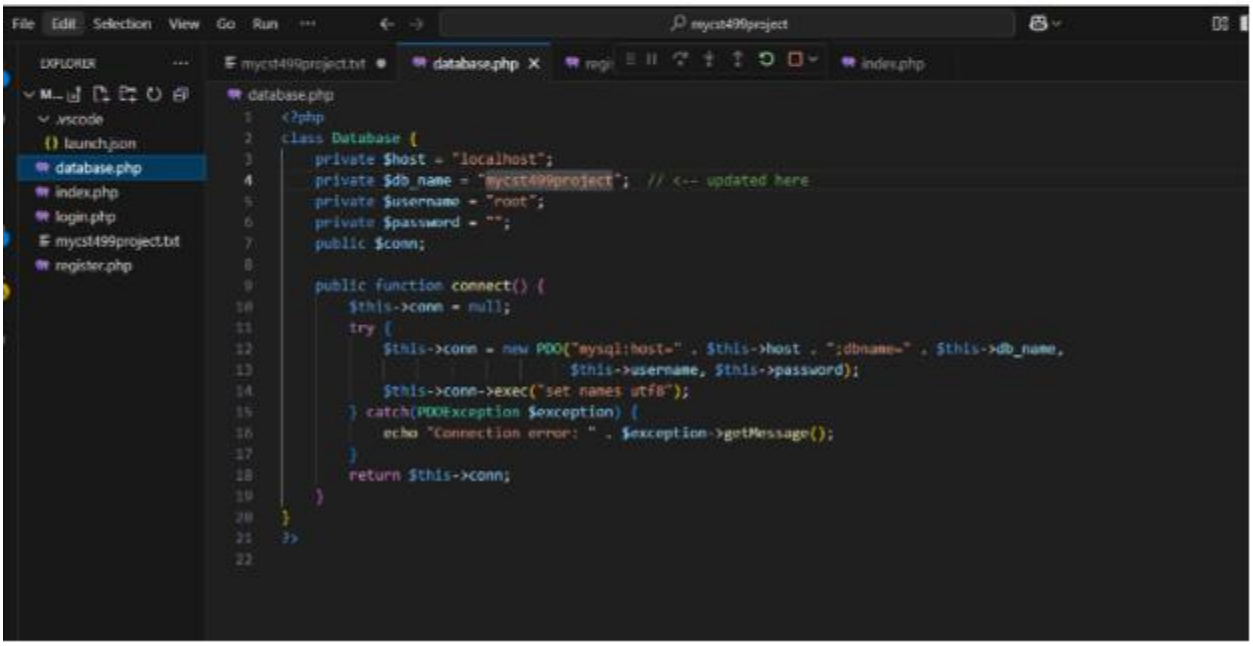
State Diagram

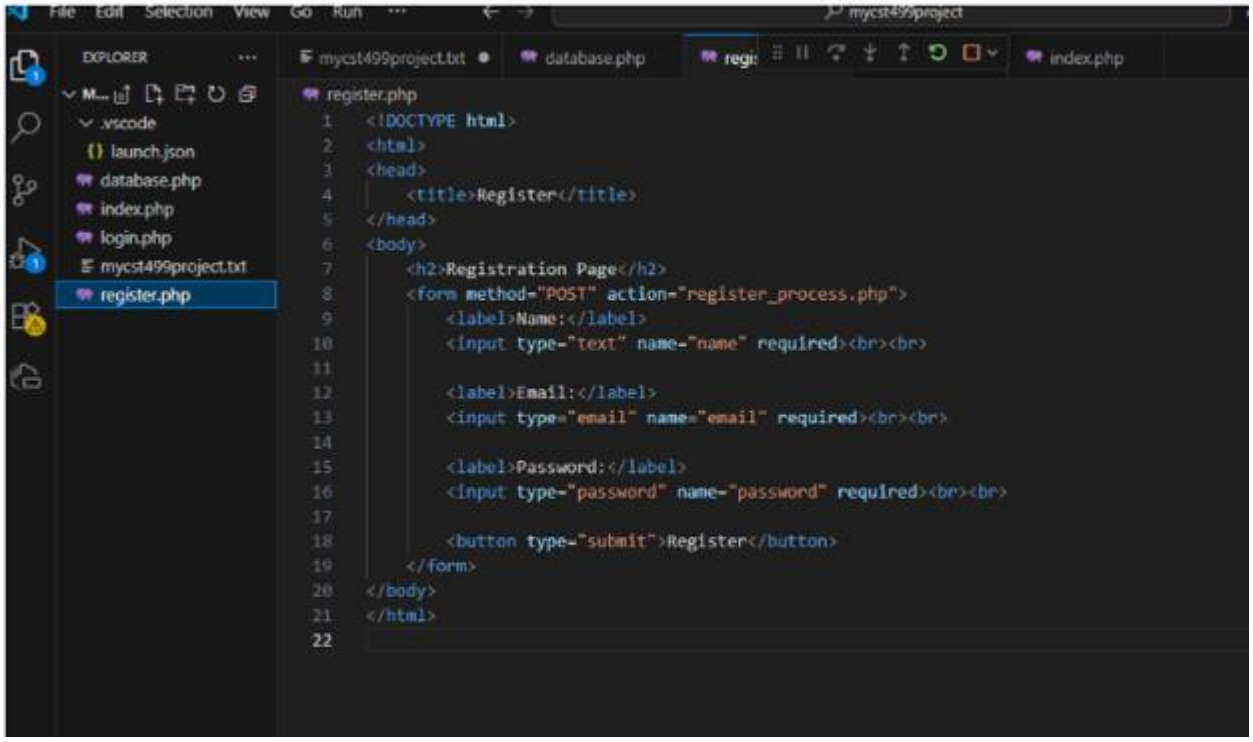


This diagram captures lifecycle behavior for dynamic components, particularly the course object.

Week 3


File Edit Selection View Go Run
mycst499project
index.php
.vscode
launch.json
database.php
index.php
login.php
mycst499project.txt
register.php
<!DOCTYPE html>
<html>
<head>
<title>Landing Page</title>
</head>
<body>
<h1>Welcome to My CST499 Project</h1>
Please choose an option below:
Login | Register
</body>
</html>

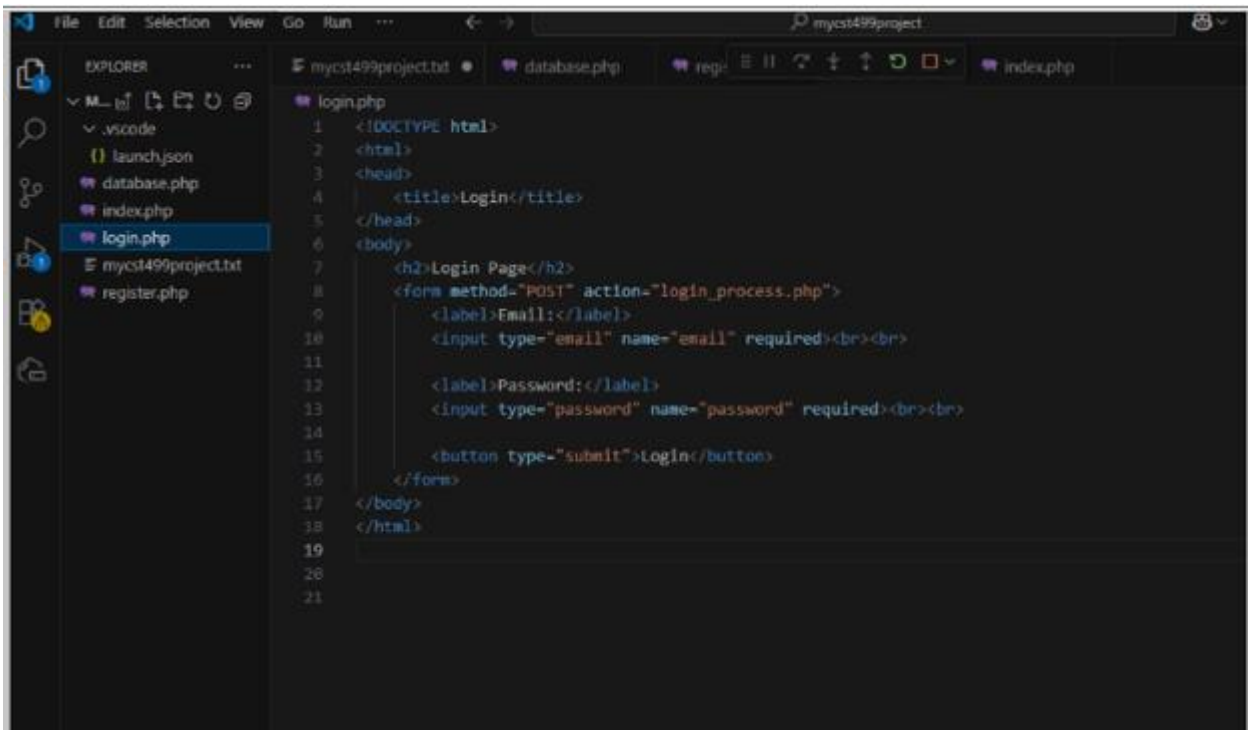

File Edit Selection View Go Run
mycst499project
EXPLORER
database.php
<?php
class Database {
private $host = "localhost";
private $db_name = "mycst499project"; // <-- updated here
private $username = "root";
private $password = "";
public $conn;
public function connect() {
$this->conn = null;
try {
$this->conn = new PDO("mysql:host=" . $this->host . ";dbname=" . $this->db_name,
$this->username, $this->password);
$this->conn->exec("set names utf8");
} catch(PDOException $exception) {
echo "Connection error: " . $exception->getMessage();
}
return $this->conn;
}
}
?>


File Edit Selection View Go Run
mycst499project
EXPLORER
mycst499project.txt
database.php
index.php
.vscode
launch.json
database.php
index.php
login.php
mycst499project.txt
register.php
register.php
<!DOCTYPE html>
<html>
<head>
<title>Register</title>
</head>
<body>
<h2>Registration Page</h2>
<form method="POST" action="register_process.php">
<label>Name:</label>
<input type="text" name="name" required>

<label>Email:</label>
<input type="email" name="email" required>

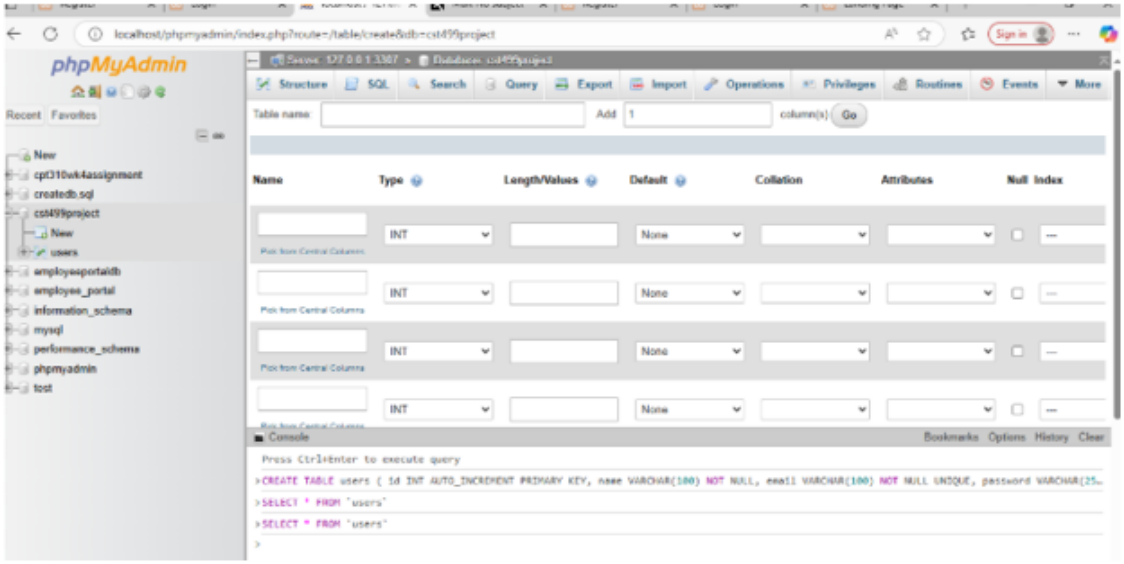
<label>Password:</label>
<input type="password" name="password" required>

<button type="submit">Register</button>
</form>
</body>
</html>


File Edit Selection View Go Run
mycst499project
EXPLORER
mycst499project.txt
database.php
index.php
.vscode
launch.json
database.php
index.php
login.php
mycst499project.txt
register.php
login.php
<!DOCTYPE html>
<html>
<head>
<title>Login</title>
</head>
<body>
<h2>Login Page</h2>
<form method="POST" action="login_process.php">
<label>Email:</label>
<input type="email" name="email" required>

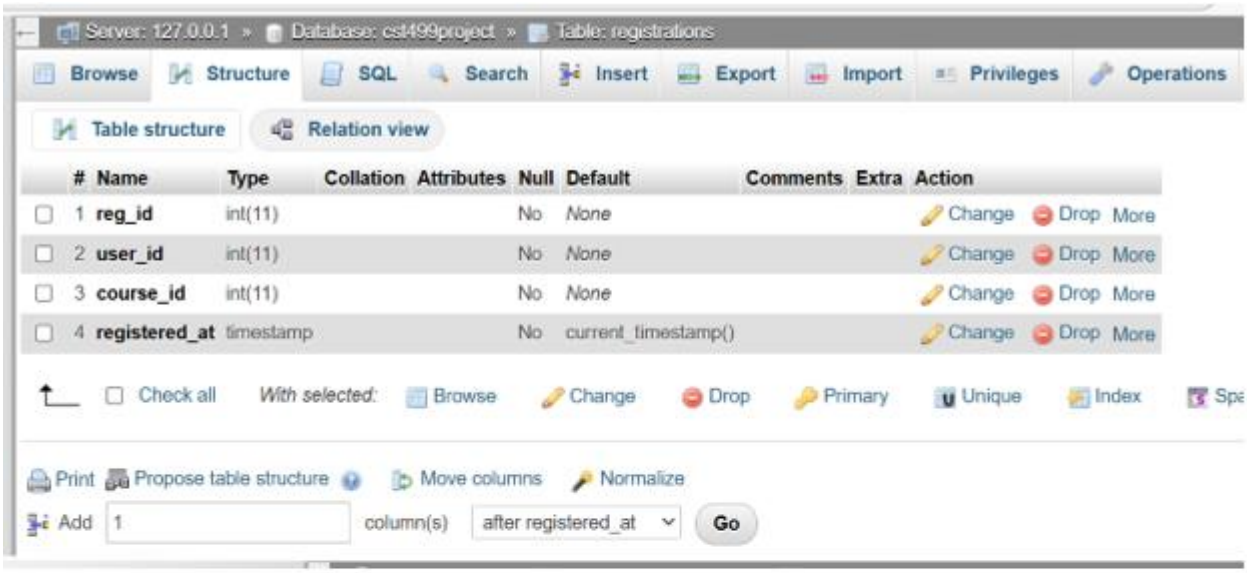
<label>Password:</label>
<input type="password" name="password" required>

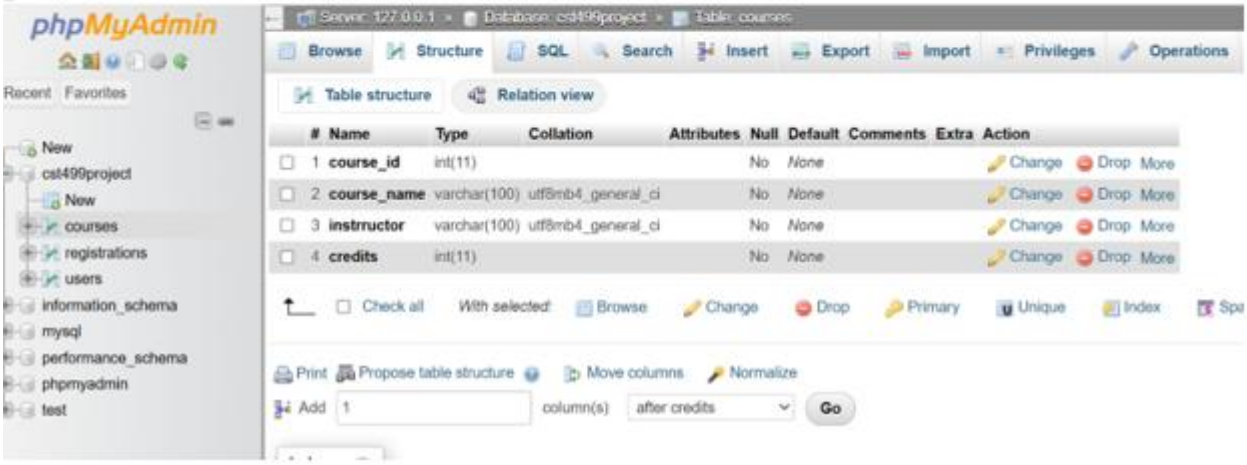
<button type="submit">Login</buttoe>
</form>
</body>
</html>

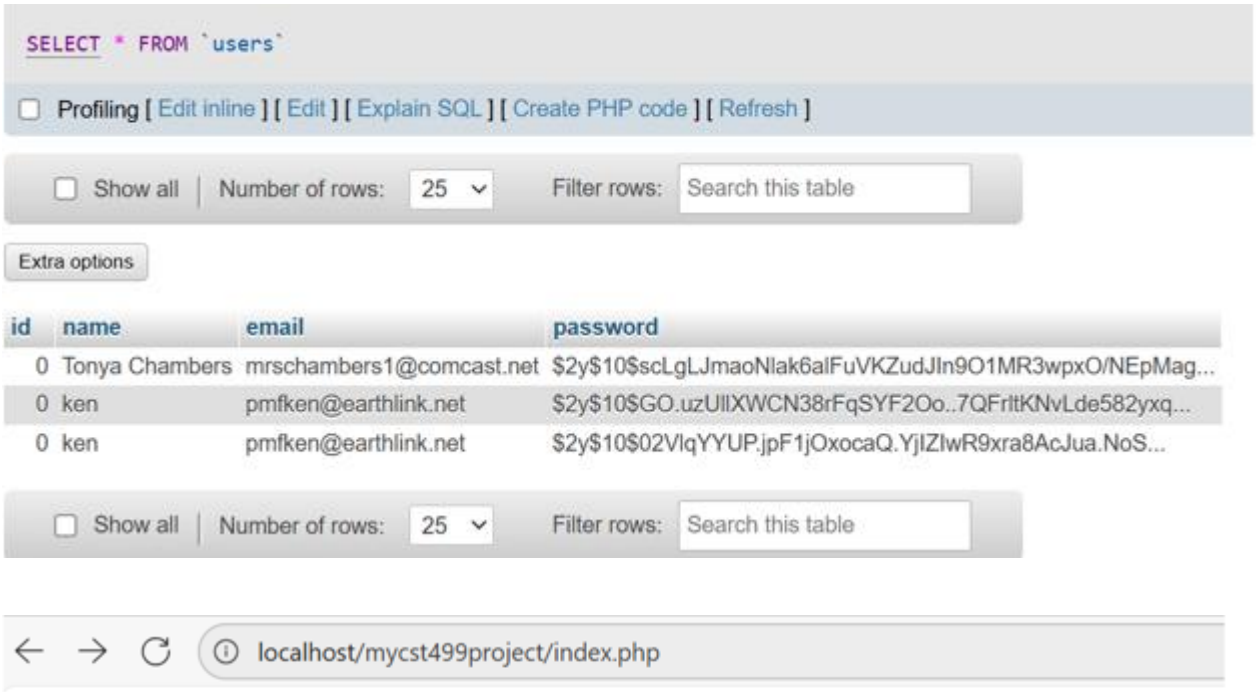


Week 4

Database Development and Class Registration


Server: 127.0.0.1 » Database: cst499project » Table: registrations
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Table structure
Relation view
Name Type Collation Attributes Null Default Comments Extra Action
1 reg_id int(11) No None Change Drop More
2 user_id int(11) No None Change Drop More
3 course_id int(11) No None Change Drop More
4 registered_at timestamp No current_timestamp() Change Drop More
Check all With selected: Browse Change Drop Primary Unique Index
Print Propose table structure Move columns Normalize
Add 1 column(s) after registered_at Go


phpMyAdmin
Recent Favorites
New
cst499project
New
courses
registrations
users
information_schema
mysql
performance_schema
phpmyadmin
test
Server: 127.0.0.1 » Database: cst499project » Table: courses
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Table structure
Relation view
Name Type Collation Attributes Null Default Comments Extra Action
1 course_id int(11) No None Change Drop More
2 course_name varchar(100) utf8mb4_general_ci No None Change Drop More
3 instructor varchar(100) utf8mb4_general_ci No None Change Drop More
4 credits int(11) No None Change Drop More
Check all With selected: Browse Change Drop Primary Unique Index
Print Propose table structure Move columns Normalize
Add 1 column(s) after credits Go



```
SELECT * FROM `users`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

Extra options

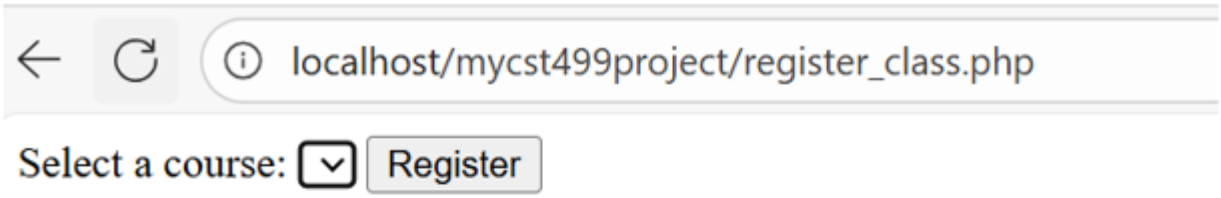
id	name	email	password
0	Tonya Chambers	mrschambers1@comcast.net	$2y$10$scLgLJmaoNlak6alFuVKZudJln9O1MR3wpxO/NEpMag...
0	ken	pmfken@earthlink.net	$2y$10$GO.uzUllXWCN38rFqSYF2Oo..7QFrltKNvLde582yxq...
0	ken	pmfken@earthlink.net	$2y$10$02VlqYYUP.jpF1jOxocaQ.YjIZlwR9xra8AcJua.NoS...

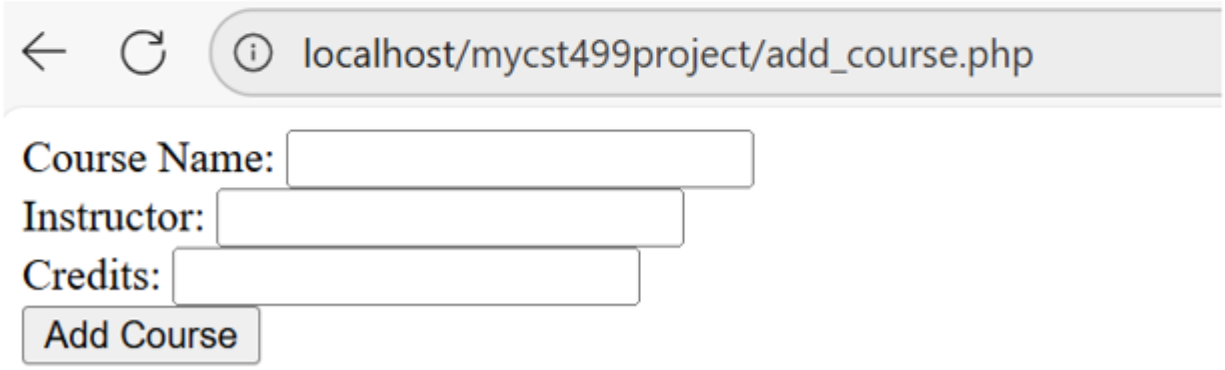
Show all | Number of rows: 25 Filter rows: Search this table

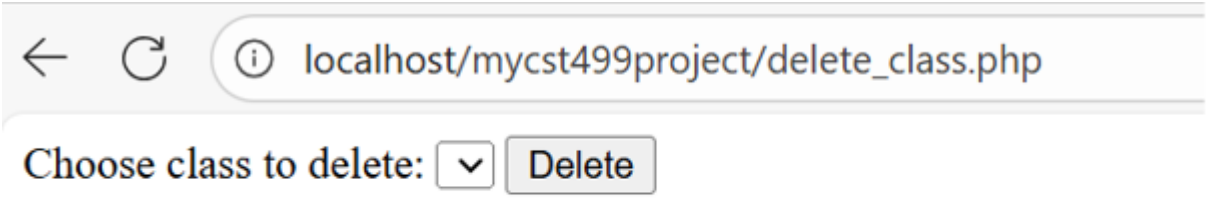
localhost/mycst499project/index.php

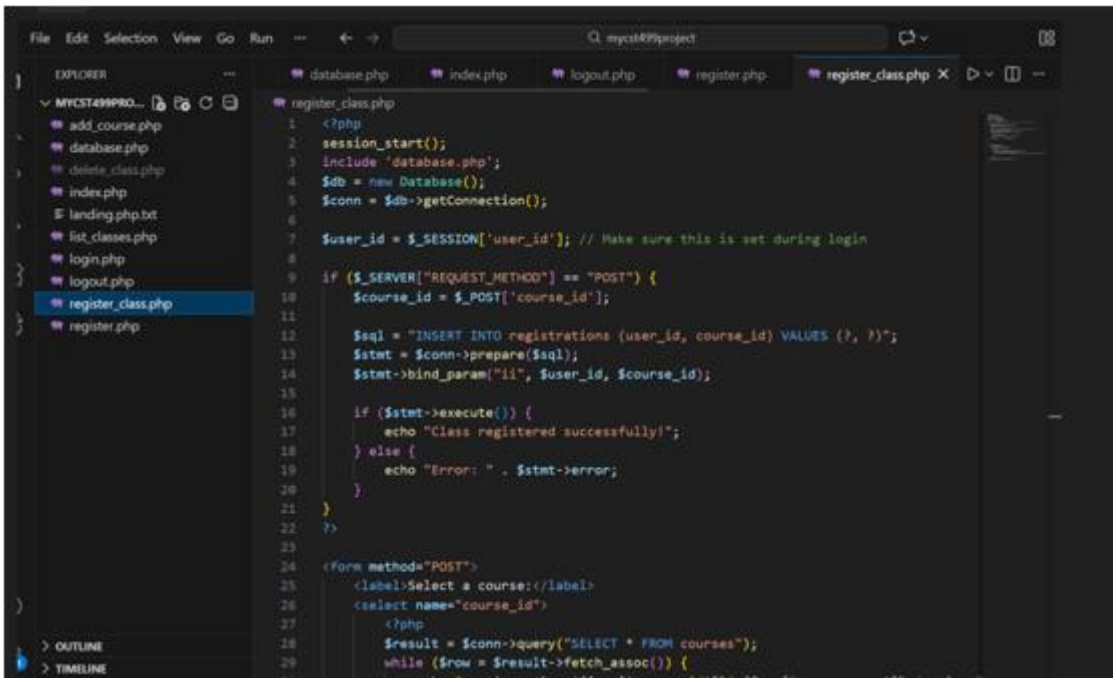
Welcome, ken!

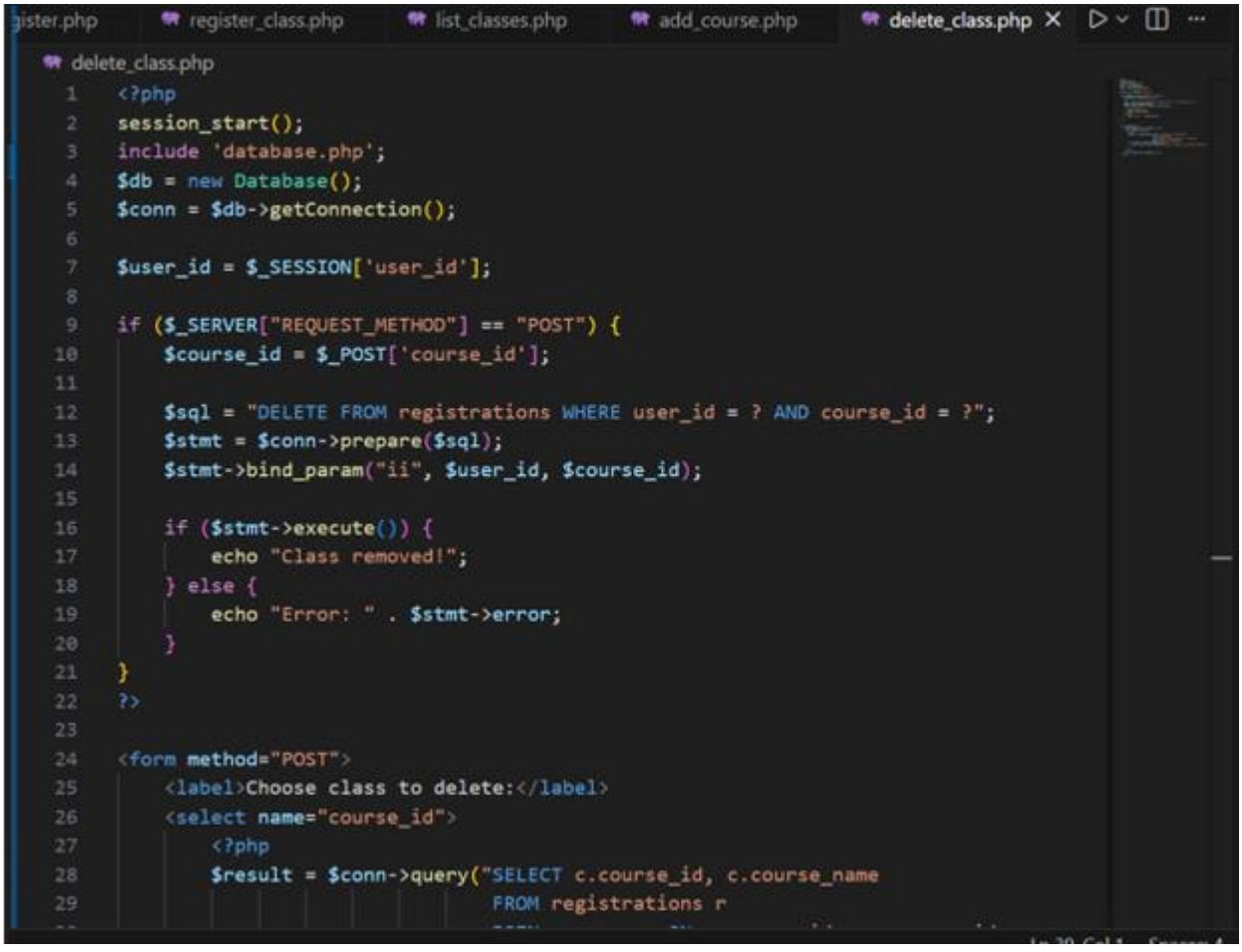
View Your Classes | Register for a Class | Add New Course | Delete a Class | Logout


localhost/mycst499project/register_class.php
Select a course:
Register


localhost/mycst499project/add_course.php
Course Name:
Instructor:
Credits:
Add Course


localhost/mycst499project/delete_class.php
Choose class to delete:
Delete


File Edit Selection View Go Run
EXPLORER
database.php
index.php
logout.php
register.php
register_class.php
add_course.php
database.php
delete_class.php
index.php
landing.php.txt
list_classes.php
login.php
logout.php
register_class.php
register.php
OUTLINE
TIMELINE
<?php
session_start();
include 'database.php';
$db = new Database();
$conn = $db->getConnection();
$user_id = $_SESSION['user_id']; // Make sure this is set during login
if ($_SERVER["REQUEST_METHOD"] == "POST") {
$course_id = $_POST['course_id'];
$sql = "INSERT INTO registrations (user_id, course_id) VALUES (?, ?)";
$stmt = $conn->prepare($sql);
$stmt->bind_param("ii", $user_id, $course_id);
if ($stmt->execute()) {
echo "Class registered successfully!";
} else {
echo "Error: " . $stmt->error;
}
}
?>
<form method="POST">
<label>Select a course:</label>
<select name="course_id">
<?php
$result = $conn->query("SELECT * FROM courses");
while ($row = $result->fetch_assoc()) {


register_class.php
list_classes.php
add_course.php
delete_class.php
delete_class.php
<?php
session_start();
include 'database.php';
$db = new Database();
$conn = $db->getConnection();
$user_id = $_SESSION['user_id'];
if ($_SERVER["REQUEST_METHOD"] == "POST") {
$course_id = $_POST['course_id'];
$sql = "DELETE FROM registrations WHERE user_id = ? AND course_id = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("ii", $user_id, $course_id);
if ($stmt->execute()) {
echo "Class removed!";
} else {
echo "Error: " . $stmt->error;
}
}
?>
<form method="POST">
<label>Choose class to delete:</label>
<select name="course_id">
<?php
$result = $conn->query("SELECT c.course_id, c.course_name
FROM registrations r

logout.php

```php
<?php
session_start();

// Destroy all session data
session_unset();
session_destroy();

// Redirect back to landing page
header("Location: index.php");
exit();
?>
```

list_classes.php

```php
<?php
session_start();
include 'database.php';
$db = new Database();
$conn = $db->getConnection();

$user_id = $_SESSION['user_id'];

$sql = "SELECT c.course_name, c.instructor, c.credits
        FROM registrations r
        JOIN courses c ON r.course_id = c.course_id
        WHERE r.user_id = ?";
$stmt = $conn->prepare($sql);
$stmt->bind_param("i", $user_id);
$stmt->execute();
$result = $stmt->get_result();

echo "<h2>Your Registered Classes</h2><ul>";
while ($row = $result->fetch_assoc()) {
    echo "<li>{$row['course_name']} - {$row['instructor']} ({$row['credits']} credits
}
echo "</ul>";
?>
```