

Project : University Course Registration System

Groups Members Details:

Name: Shakib Alom Fahim Student

ID: 2222866042

1. Proposed Functional Requirements:

Functional Requirements/Specifications

The system will include the following core functionalities:

1. User Authentication and Role Management

- Secure login for students, faculty, and administrators.
- Role-based access to different system features.

2. Course Browsing and Registration

- Students can search for courses based on subject, department, or instructor.
- Real-time seat availability updates.
- Pre-requisite verification before course enrollment.

3. Faculty and Administrator Features

- Faculty can manage student enrollments and view class lists.
- Administrators can approve course overload requests and adjust enrollment limits.

4. Timetable Management

- Automated scheduling to prevent time conflicts.
- Students can view their registered courses in a weekly timetable format.

5. Waitlist Management

- When a course is full, students can join a waitlist.

Completed Functional Requirements:

Screenshot of code index.php:

```
<?php
<html lang="en">
<body>
    <h1 class="university-name">Slim Shady University</h1>
    <p class="motto">"Will the Real Graduates Please Stand Up"</p>

    <div class="container">
        <!-- Login Form -->
        <div class="login-container">
            <h2 class="form-title">Login</h2>
            <?php if ($error && isset($_POST['action'])) && $_POST['action'] === 'login'): ?>
                <div class="error"><?= htmlspecialchars(string: $error) ?></div>
            <?php endif; ?>
            <form method="POST">
                <input type="hidden" name="action" value="login">
                <div class="form-group">
                    <label for="role">Select Role:</label>
                    <select id="role" name="role" required>
                        <option>--Choose--</option>
                        <option value="student">Student</option>
                        <option value="instructor">Instructor</option>
                        <option value="admin">Admin</option>
                    </select>
                </div>
                <div class="form-group">
                    <label for="username">Username:</label>
                    <input type="text" id="username" name="username" required>
                </div>
                <div class="form-group">
                    <label for="password">Password:</label>
                    <input type="password" id="password" name="password" required>
                </div>
                <button type="submit" class="btn">Login</button>
            </form>
        </div>
        <!-- Registration Form -->
        <div class="register-container">
            <h2 class="form-title">Register</h2>
            <?php if ($success): ?>

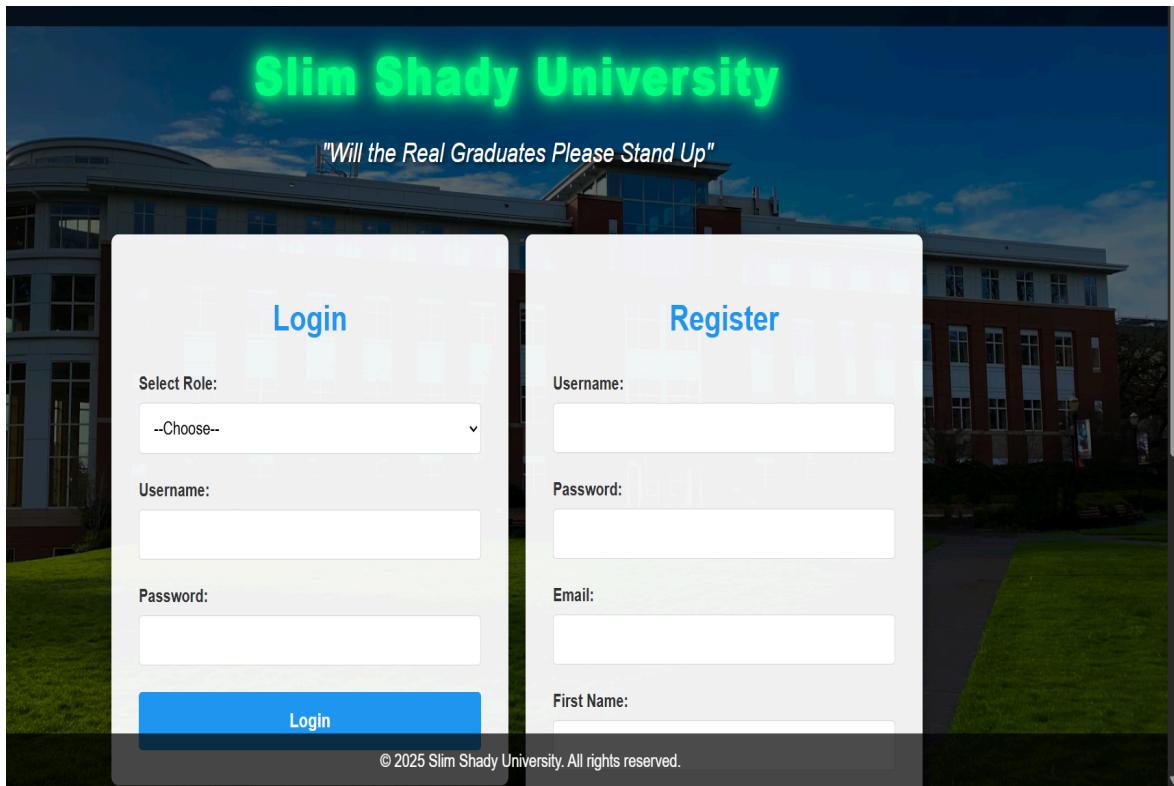
```

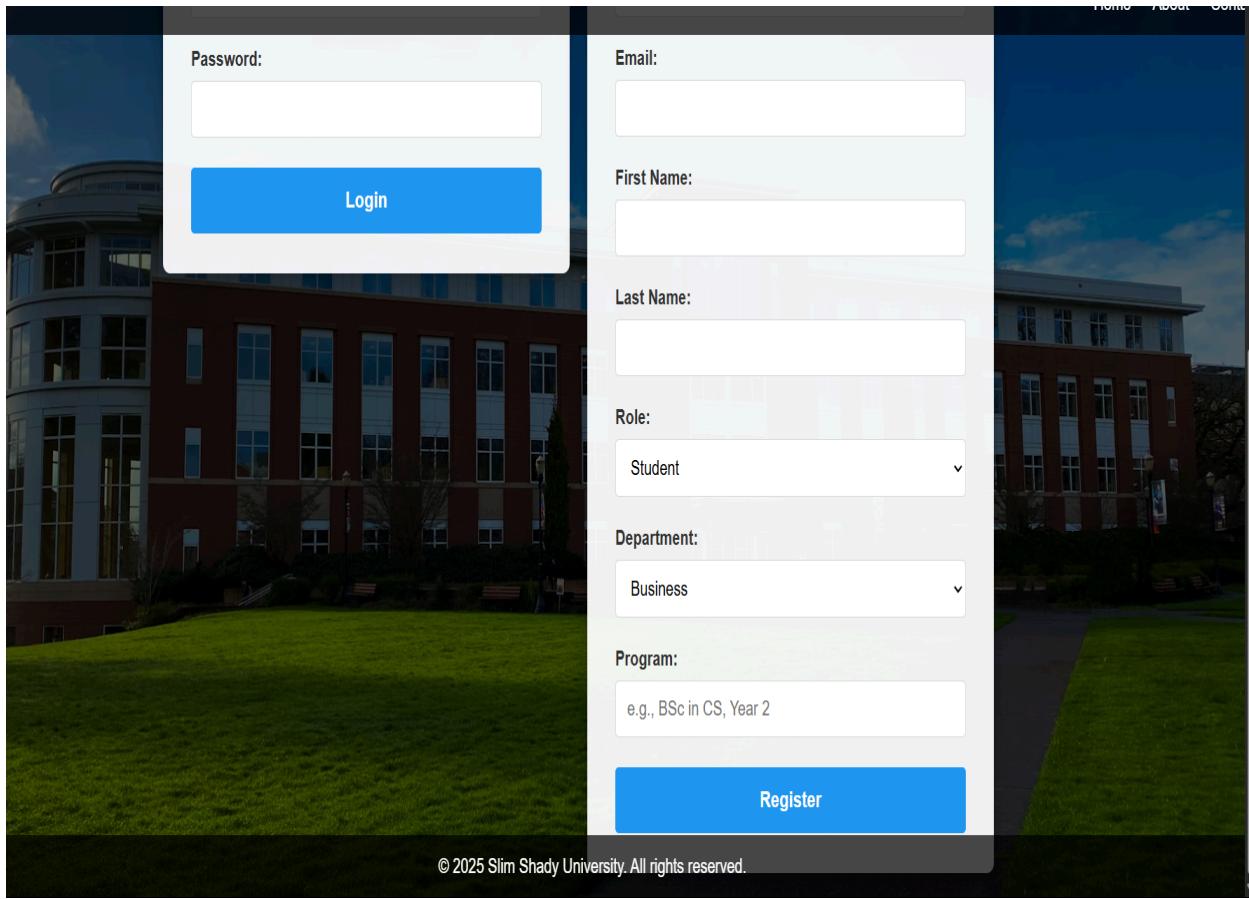
itory

Ln 393, Col 41 Spaces: 4 UTF-8 CRLF {} PHP ⌂ ⌂

```
<body>
<div class="container">
<div class="register-container">
<?php if ($success): ?>
<div class="success"><?= htmlspecialchars(string: $success) ?></div>
<?php endif; ?>
<?php if ($error && isset($_POST['action']) && $_POST['action'] === 'register'): ?>
<div class="error"><?= htmlspecialchars(string: $error) ?></div>
<?php endif; ?>
<form method="POST">
<input type="hidden" name="action" value="register">
<div class="form-group">
<label for="reg_username">Username:</label>
<input type="text" id="reg_username" name="username" required>
</div>
<div class="form-group">
<label for="reg_password">Password:</label>
<input type="password" id="reg_password" name="password" required>
</div>
<div class="form-group">
<label for="email">Email:</label>
<input type="email" id="email" name="email" required>
</div>
<div class="form-group">
<label for="first_name">First Name:</label>
<input type="text" id="first_name" name="first_name" required>
</div>
<div class="form-group">
<label for="last_name">Last Name:</label>
<input type="text" id="last_name" name="last_name" required>
</div>
<div class="form-group">
<label for="reg_role">Role:</label>
<select id="reg_role" name="role" required onchange="toggleProgramField()">
<option value="student">Student</option>
<option value="instructor">Instructor</option>
<option value="admin">Admin</option>
</select>
...</div>
```

Screenshot of UI:





The image shows a registration form overlaid on a photograph of a university campus. The background features a large, modern brick building with many windows and a circular tower on the left, set against a backdrop of a clear blue sky with some clouds. In the foreground, there is a well-maintained green lawn.

Registration Form Fields:

- Password:** Input field
- Email:** Input field
- First Name:** Input field
- Last Name:** Input field
- Role:** Select dropdown menu currently set to "Student"
- Department:** Select dropdown menu currently set to "Business"
- Program:** Input field with placeholder "e.g., BSc in CS, Year 2"
- Register** Button

Page Header: Home | About | Contact

Page Footer: © 2025 Slim Shady University. All rights reserved.

Screenshot of Code of Admin.php:

```
Admin.php
1 <?php
2 session_start();
3
4 // Check if user is logged in and is an admin
5 if (!isset($_SESSION['role']) || $_SESSION['role'] != 'admin') {
6     header("Location: index.php");
7     exit;
8 }
9
10 // Database connection
11 $conn = new mysqli(hostname: 'localhost', username: 'root', password: '', database: 'course_registration');
12 if ($conn->connect_error) {
13     die("Connection failed: " . $conn->connect_error);
14 }
15
16 // Handle course creation
17 if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['create_course'])) {
18     $course_code = $_POST['course_code'];
19     $course_name = $_POST['course_name'];
20     $department_id = $_POST['department_id'];
21     $instructor_id = $_POST['instructor_id'];
22     $credits = $_POST['credits'];
23     $max_students = $_POST['max_students'];
24     $schedule_day = $_POST['schedule_day'];
25     $schedule_time = $_POST['schedule_time'];
26     $room = $_POST['room'];
27     $semester = $_POST['semester'];
28     $year = $_POST['year'];
29
30     $stmt = $conn->prepare(query: "INSERT INTO courses (course_code, course_name, department_id, instructor_id, credits, max_students, schedule_day, schedule_time");
31     $stmt->bind_param(types: "ssiiissssi", var: &$course_code, vars: &$course_name, $department_id, $instructor_id, $credits, $max_students, $schedule_day, $schedule_time);
32
33     if ($stmt->execute()) {
34         $_SESSION['success_message'] = "Course created successfully!";
35     } else {
36         $_SESSION['error_message'] = "Error creating course: " . $conn->error;
37     }
38     $stmt->close();
39 }
40
41 // Redirect back to the course management section
42 header(header: "Location: admin.php#course-management");
43 exit;
44
45 // Handle course editing
46 if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['edit_course'])) {
47     $course_id = $_POST['course_id'];
48     $course_code = $_POST['edit_course_code'];
49     $course_name = $_POST['edit_course_name'];
50     $department_id = $_POST['edit_department_id'];
51     $instructor_id = $_POST['edit_instructor_id'];
52     $schedule_day = $_POST['edit_schedule_day'];
53     $schedule_time = $_POST['edit_schedule_time'];
54     $room = $_POST['edit_room'];
55     $max_students = $_POST['edit_max_students'];
56
57     $stmt = $conn->prepare(query: "UPDATE courses SET course_code = ?, course_name = ?, department_id = ?, instructor_id = ?, schedule_day = ?, schedule_time = ?, room = ?");
58     $stmt->bind_param(types: "ssiiisss", var: &$course_code, vars: &$course_name, $department_id, $instructor_id, $schedule_day, $schedule_time, $room, $max_students);
59
60     if ($stmt->execute()) {
61         $_SESSION['success_message'] = "Course updated successfully!";
62     } else {
63         $_SESSION['error_message'] = "Error updating course: " . $conn->error;
64     }
65     $stmt->close();
66
67     // Redirect back to the course management section
68     header(header: "Location: admin.php#course-management");
69     exit;
70 }
```

```
Admin.php
17 if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['create_course'])) {
18     if ($stmt->execute()) {
19     } else {
20         $_SESSION['error_message'] = "Error creating course: " . $conn->error;
21     }
22     $stmt->close();
23
24     // Redirect back to the course management section
25     header(header: "Location: admin.php#course-management");
26     exit;
27 }
28
29 // Handle course editing
30 if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['edit_course'])) {
31     $course_id = $_POST['course_id'];
32     $course_code = $_POST['edit_course_code'];
33     $course_name = $_POST['edit_course_name'];
34     $department_id = $_POST['edit_department_id'];
35     $instructor_id = $_POST['edit_instructor_id'];
36     $schedule_day = $_POST['edit_schedule_day'];
37     $schedule_time = $_POST['edit_schedule_time'];
38     $room = $_POST['edit_room'];
39     $max_students = $_POST['edit_max_students'];
40
41     $stmt = $conn->prepare(query: "UPDATE courses SET course_code = ?, course_name = ?, department_id = ?, instructor_id = ?, schedule_day = ?, schedule_time = ?, room = ?");
42     $stmt->bind_param(types: "ssiiisss", var: &$course_code, vars: &$course_name, $department_id, $instructor_id, $schedule_day, $schedule_time, $room, $max_students);
43
44     if ($stmt->execute()) {
45         $_SESSION['success_message'] = "Course updated successfully!";
46     } else {
47         $_SESSION['error_message'] = "Error updating course: " . $conn->error;
48     }
49     $stmt->close();
50
51     // Redirect back to the course management section
52     header(header: "Location: admin.php#course-management");
53     exit;
54 }
```

Screenshot of UI:

Admin Dashboard

Student Management Course Management Logout

Enrolled Students

ID	Name	Department	Email	Actions
8	sakib fahim	Computer Science	fahimsakib774@gmail.com	<button>View Details</button>
12	Rifat alom	Mathematics	rifat21@gmail.com	<button>View Details</button>
13	saif alom	Computer Science	saif01@gmail.com	<button>View Details</button>

Student Management Course Management

Create New Course

Course List

Code	Name	Department	Instructor	Schedule	Room	Enrolled/Max	Actions
cse327	Software Engineering	Computer Science	Rafsun islam	Thursday 15:02	SAC302	2/30	<button>Edit</button> <button>Delete</button>
Mat120	Introduction to integration	Mathematics	Rafsun islam	17:45	SAC311	2/30	<button>Edit</button> <button>Delete</button>
MAT350	Social life instructor	Business	Hasib Rahim	13:02	Nac311	0/20	<button>Edit</button> <button>Delete</button>
CSE115	Introduction to C programming	Computer Science	Hasib Rahim	Wednesday 16:32	Nac232	3/25	<button>Edit</button> <button>Delete</button>

Screenshot of Code of Instructor.php:

```
instructor.php
53  <html lang="en">
54  <head>
55  </head>
56  <body>
57      <div class="sidebar">
58          <h1>Instructor Dashboard</h1>
59          <ul class="nav-links">
60              <li><a href="#" onclick="showSection('profile')">Profile</a></li>
61              <li><a href="#" onclick="showSection('courses')">My Courses</a></li>
62              <li><a href="#" onclick="showSection('students')">Manage Students</a></li>
63              <li><a href="#" onclick="showSection('grades')">Grade Management</a></li>
64              <li><a href="#" onclick="showSection('schedule')">Schedule</a></li>
65              <li><a href="logout.php">Logout</a></li>
66          </ul>
67      </div>
68
69      <div class="main-content">
70          <div id="notification" class="notification"></div>
71
72          <!-- Profile Section -->
73          <div id="profile" class="content-section active">
74              <div class="profile-info">
75                  <h2>Instructor Profile</h2>
76                  <div class="info-item">
77                      <strong>Name:</strong> <?php echo htmlspecialchars(string: $instructor_name); ?>
78                  </div>
79                  <div class="info-item">
80                      <strong>ID:</strong> <?php echo htmlspecialchars(string: $instructor_id); ?>
81                  </div>
82                  <div class="info-item">
83                      <strong>Department:</strong> <?php echo htmlspecialchars(string: $instructor_department); ?>
84                  </div>
85                  <div class="info-item">
86                      <strong>Email:</strong> <?php echo htmlspecialchars(string: $instructor_email); ?>
87                  </div>
88              </div>
89          </div>
90
91          <!-- Courses Section -->
92          <div id="courses" class="content-section">
93              <h2>My Courses</h2>
94              <table>
95                  <thead>
96                      <tr>
97                          <th>Course Code</th>
98                          <th>Course Name</th>
99                          <th>Schedule</th>
100                         <th>Room</th>
101                         <th>Current Students</th>
102                         <th>Max Students</th>
103                     </tr>
104                 </thead>
105                 <tbody>
106                     <?php while($course = $courses_result->fetch_assoc()): ?>
107                     <tr>
108                         <td><?php echo htmlspecialchars(string: $course['course_code']); ?></td>
109                         <td><?php echo htmlspecialchars(string: $course['course_name']); ?></td>
110                         <td><?php echo htmlspecialchars(string: $course['schedule_day'] . ' ' . $course['schedule_time']); ?></td>
111                         <td><?php echo htmlspecialchars(string: $course['room']); ?></td>
112                         <td><?php echo htmlspecialchars(string: $course['current_students']); ?></td>
113                         <td><?php echo htmlspecialchars(string: $course['max_students']); ?></td>
114                     </tr>
115                 <?php endwhile; ?>
116             </tbody>
117         </table>
118     </div>
119
120     <!-- Students Section -->
121     <div id="students" class="content-section">
122         <h2>Manage Students</h2>
123         <table>
124             <thead>
125                 <tr>
126                     <th>Student Name</th>
127                 </tr>
128             </thead>
129             <tbody>
130                 <tr>
131                     <td><?php echo htmlspecialchars(string: $student_name); ?></td>
132                 </tr>
133             </tbody>
134         </table>
135     </div>
136 
```

inventor

Ln 232, Col 32 Spaces: 4 UTF-8 CRLF {} PHP ⚙ Go Live 8.2 ⚙ Prettier

```
instructor.php
53  <html lang="en">
54  <body>
55      <div class="main-content">
56
57          <!-- Courses Section -->
58          <div id="courses" class="content-section">
59              <h2>My Courses</h2>
60              <table>
61                  <thead>
62                      <tr>
63                          <th>Course Code</th>
64                          <th>Course Name</th>
65                          <th>Schedule</th>
66                          <th>Room</th>
67                          <th>Current Students</th>
68                          <th>Max Students</th>
69                      </tr>
70                  </thead>
71                  <tbody>
72                      <?php while($course = $courses_result->fetch_assoc()): ?>
73                      <tr>
74                          <td><?php echo htmlspecialchars(string: $course['course_code']); ?></td>
75                          <td><?php echo htmlspecialchars(string: $course['course_name']); ?></td>
76                          <td><?php echo htmlspecialchars(string: $course['schedule_day'] . ' ' . $course['schedule_time']); ?></td>
77                          <td><?php echo htmlspecialchars(string: $course['room']); ?></td>
78                          <td><?php echo htmlspecialchars(string: $course['current_students']); ?></td>
79                          <td><?php echo htmlspecialchars(string: $course['max_students']); ?></td>
80                      </tr>
81                  <?php endwhile; ?>
82              </tbody>
83          </table>
84      </div>
85
86      <!-- Students Section -->
87      <div id="students" class="content-section">
88          <h2>Manage Students</h2>
89          <table>
90              <thead>
91                  <tr>
92                      <th>Student Name</th>
93                  </tr>
94              </thead>
95              <tbody>
96                  <tr>
97                      <td><?php echo htmlspecialchars(string: $student_name); ?></td>
98                  </tr>
99              </tbody>
100         </table>
101     </div>
102 
```

inventor

Ln 232, Col 32 Spaces: 4 UTF-8 CRLF {} PHP ⚙ Go Live 8.2 ⚙ Prettier

Screenshot of UI:

Instructor Profile

Name: Hasib Rahim

ID: 9

Department: Computer Science

Email: fahimsakib00@gmail.com

My Courses

Course Code	Course Name	Schedule	Room	Current Students	Max Students
MAT350	Social life instructor	0 13:02	Nac311	0	20
CSE115	Introduction to C programming	Wednesday 16:32	Nac232	3	25

Manage Students

Student Name	Course	Email	Program	Status
Rifat alom	CSE115 - Introduction to C programming	rifat21@gmail.com	BSc in mathematics	Enrolled
saif alom	CSE115 - Introduction to C programming	saif01@gmail.com	Bsc in Cs.year 1	Enrolled
sakib fahim	CSE115 - Introduction to C programming	fahimsakib774@gmail.com	Bsc in Cs.year 2	Enrolled

Grade Management

Enter grades as A, B, C, D, or F

Student Name	Course	Current Grade	Action
Rifat alom	CSE115	B	<button>Save Grade</button>
saif alom	CSE115	F	<button>Save Grade</button>
sakib fahim	CSE115	A	<button>Save Grade</button>

Teaching Schedule

MAT350

Social life instructor

0 13:02

Room: Nac311

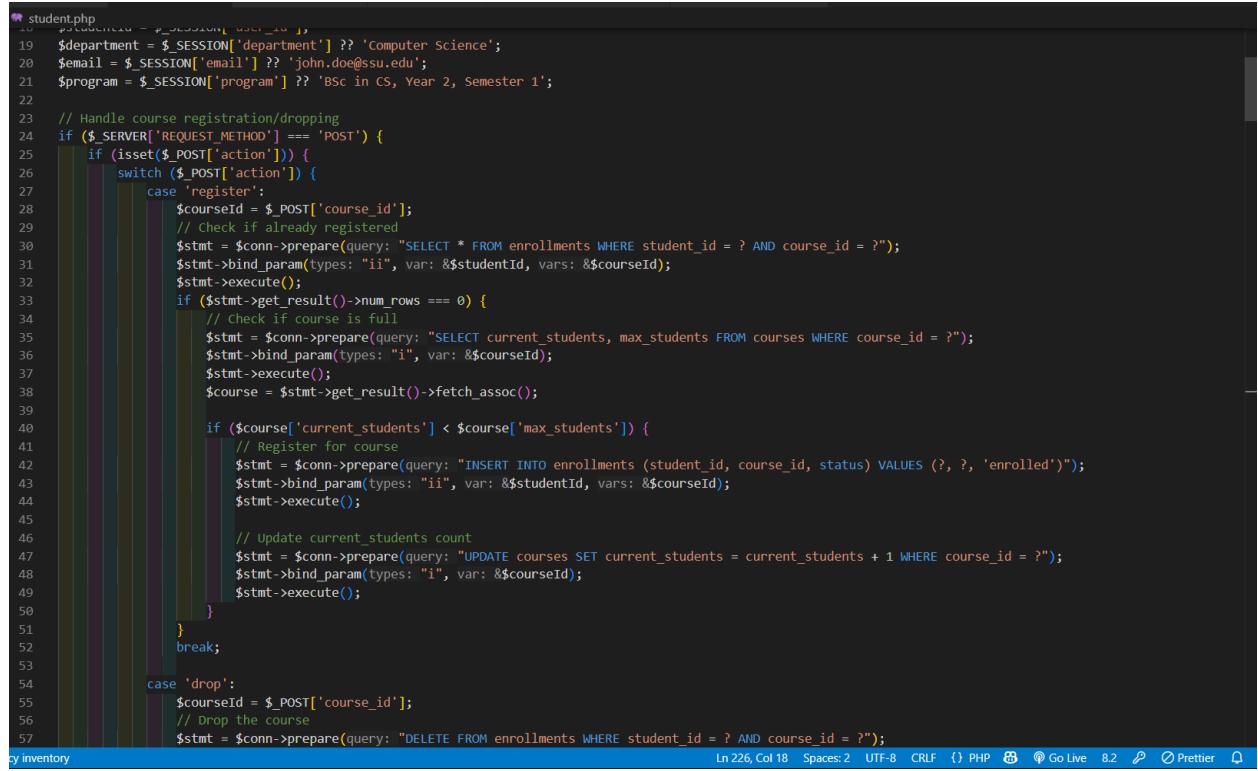
CSE115

Introduction to C programming

Wednesday 16:32

Room: Nac232

Screenshot of Code of Student.php:



```
student.php
1 $studentId = $_SESSION['user_id'];
2 $department = $_SESSION['department'] ?? 'Computer Science';
3 $email = $_SESSION['email'] ?? 'john.doe@ssu.edu';
4 $program = $_SESSION['program'] ?? 'BSc in CS, Year 2, Semester 1';
5
6 // Handle course registration/dropping
7 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
8     if (isset($_POST['action'])) {
9         switch ($_POST['action']) {
10             case 'register':
11                 $courseId = $_POST['course_id'];
12                 // Check if already registered
13                 $stmt = $conn->prepare("SELECT * FROM enrollments WHERE student_id = ? AND course_id = ?");
14                 $stmt->bind_param(types: "ii", var: &$amp;$studentId, vars: &$courseId);
15                 $stmt->execute();
16                 if ($stmt->get_result()->num_rows === 0) {
17                     // check if course is full
18                     $stmt = $conn->prepare("SELECT current_students, max_students FROM courses WHERE course_id = ?");
19                     $stmt->bind_param(types: "i", var: &$courseId);
20                     $stmt->execute();
21                     $course = $stmt->get_result()->fetch_assoc();
22
23                     if ($course['current_students'] < $course['max_students']) {
24                         // Register for course
25                         $stmt = $conn->prepare("INSERT INTO enrollments (student_id, course_id, status) VALUES (?, ?, 'enrolled')");
26                         $stmt->bind_param(types: "ii", var: &$studentId, vars: &$courseId);
27                         $stmt->execute();
28
29                         // Update current_students count
30                         $stmt = $conn->prepare("UPDATE courses SET current_students = current_students + 1 WHERE course_id = ?");
31                         $stmt->bind_param(types: "i", var: &$courseId);
32                         $stmt->execute();
33                     }
34                 }
35             break;
36
37             case 'drop':
38                 $courseId = $_POST['course_id'];
39                 // Drop the course
40                 $stmt = $conn->prepare("DELETE FROM enrollments WHERE student_id = ? AND course_id = ?");
41             }
42         }
43     }
44 }
45
46
47
48
49
50
51
52
53
54
55
56
57
```

Inventory

Ln 226 Col 18 Spaces: 2 UTF-8 CRLF {} PHP ⚡ Go Live 8.2 ⚡ Prettier ⚡

```
student.php
100  <html lang="en">
192  <body>
193  <nav>
195  <a onclick="showSection('profile')">Profile</a>
196  <a onclick="showSection('registration')">Register/Drop Course</a>
197  <a onclick="showSection('courses')">View Courses</a>
198  <a onclick="showSection('instructors')">Instructors</a>
199  <a onclick="showSection('grades')">View Grades</a>
200  <a href="logout.php">Logout</a>
201 </nav>
202
203 <div class="container">
204 <section id="profile" class="active">
205 <h2>Student Profile</h2>
206 <p><strong>Name:</strong> <?= htmlspecialchars(string: $username) ?></p>
207 <p><strong>ID:</strong> <?= htmlspecialchars(string: $studentID) ?></p>
208 <p><strong>Department:</strong> <?= htmlspecialchars(string: $department) ?></p>
209 <p><strong>Email:</strong> <?= htmlspecialchars(string: $email) ?></p>
210 <p><strong>Program:</strong> <?= htmlspecialchars(string: $program) ?></p>
211 </section>
212
213 <section id="registration">
214 <h2>Course Registration</h2>
215 <div class="card">
216 <h3>Available Courses</h3>
217 <table>
218 <thead>
219 <tr>
220 <th>Course Code</th>
221 <th>Course Name</th>
222 <th>Credits</th>
223 <th>Available Slots</th>
224 <th>Instructor</th>
225 <th>Action</th>
226 </tr>
227 </thead>
228 <tbody>
229 <?php while($course = $availableCourses->fetch_assoc()): ?>
230 <tr>
231 <td><?= htmlspecialchars(string: $course['course_code']) ?></td>
```

Screenshot of UI:

Student Profile

Name: fahim

ID: 8

Department: Computer Science

Email: fahimsakib774@gmail.com

Program: Bsc in Cs,year 2

Course Registration

Available Courses

Course Code	Course Name	Credits	Available Slots	Instructor	Action
cse327	Software Engineering	3	28	Rafsun	<button>Register</button>
Mat120	Introduction to integration	3	28	Rafsun	<button>Register</button>
MAT350	Social life instructor	3	20	Hasib	<button>Register</button>
CSE115	Introduction to C programming	3	22	Hasib	<button>Register</button>

My Registered Courses

Course Code	Course Name	Credits	Instructor	Action
cse327	Software Engineering	3	Rafsun	<button>Drop</button>
CSE115	Introduction to C programming	3	Hasib	<button>Drop</button>

My Courses

Course Code	Course Name	Credits	Schedule	Instructor
cse327	Software Engineering	3	TBD	Rafsun
CSE115	Introduction to C programming	3	TBD	Hasib

Instructor Information

Course	Instructor	Email
cse327	Rafsun	Rafsun123@gmail.com
Mat120	Rafsun	Rafsun123@gmail.com
MAT350	Hasib	fahimsakib00@gmail.com
CSE115	Hasib	fahimsakib00@gmail.com

Grades

Course	Grade	Credits
cse327 - Software Engineering	A	3
CSE115 - Introduction to C programming	A	3

[Download Grade Report](#)

Grade Report

Course	Grade	Credits
cse327 - Software Engineering	A	3
CSE115 - Introduction to C programming	A	3

2. Proposed Non-Functional Requirement:

Non-Functional Requirements/Specifications

The system must meet the following non-functional requirements:

1. Performance

- The system should support a large number of concurrent users, especially during peak registration periods.
- Response times should be minimal for browsing and enrollment actions.

2. Scalability

- The architecture should allow easy expansion to accommodate growing student populations and additional courses.

3. Security

- Data encryption for student information.
- Multi-factor authentication for sensitive administrative functions.

4. Availability

- The system should maintain **99.9% uptime** to ensure uninterrupted access during registration periods.

5. Usability

- The UI should be intuitive and accessible on both desktop and mobile devices.
- The design should comply with accessibility standards for users with disabilities.

6. Backup and Recovery

- Regular data backups should be scheduled to prevent data loss.
- The system should support disaster recovery mechanisms.

Completed Non-Functional Requirement:

Screenshot of Code of User.php:

```
rc > classes > User.php > ...
3   abstract class User {
4       1 reference
5       protected $id;
6       2 references
7       protected $username;
8       3 references
9       protected $email;
10      1 reference
11      protected $password;
12
13      0 references | 0 overrides
14      public function __construct($username, $email, $password) {
15          $this->username = $username;
16          $this->email = $email;
17          $this->password = password_hash(password: $password, algo: PASSWORD_DEFAULT);
18      }
19
20      0 references | 0 overrides
21      public function getId(): mixed {
22          return $this->id;
23      }
24
25      0 references | 0 overrides
26      public function getUsername(): mixed {
27          return $this->username;
28      }
29
30      0 references | 0 overrides
31      public function getEmail(): mixed {
32          return $this->email;
33      }
34
35      0 references | 0 overrides
36      public function setEmail($email): void {
37          $this->email = $email;
38      }
39
40      0 references | 0 overrides
41      abstract public function login();
42      0 references | 0 overrides
43      abstract public function logout();
```

inventory Ln 1, Col 1 Spaces: 4 UTF-8 CRLF {} PHP ⚙

```
    public function setEmail($email): void {
        $this->validateEmail($email);
        $this->email = $email;
    }

    /**
     * Update password with validation and hashing
     * @param string $newPassword
     * @return void
     */
    public function updatePassword($newPassword): void
    {
        $this->validatePassword($newPassword);
        $this->password = password_hash($newPassword,
    }

    /**
     * Track login attempts and handle account locking
     * @param string $password
     * @return bool
     */
    protected function handleLoginAttempt($password):
        if ($this->accountLocked) {
            throw new RuntimeException("Account is locked");
        }

        if ($this->verifyPassword($password)) {
            // Reset login attempts on successful login
            $this->loginAttempts = 0;
            $this->lastLogin = new DateTime();
            return true;
        }

        // Increment failed login attempts
        $this->loginAttempts++;

        // Lock account if max attempts reached
        if ($this->loginAttempts >= self::MAX_LOGIN_ATTEMPTS) {
            $this->lockAccount();
        }
    }
}
```

```
/*
public function isAccountLocked(): bool {
    return $this->accountLocked;
}

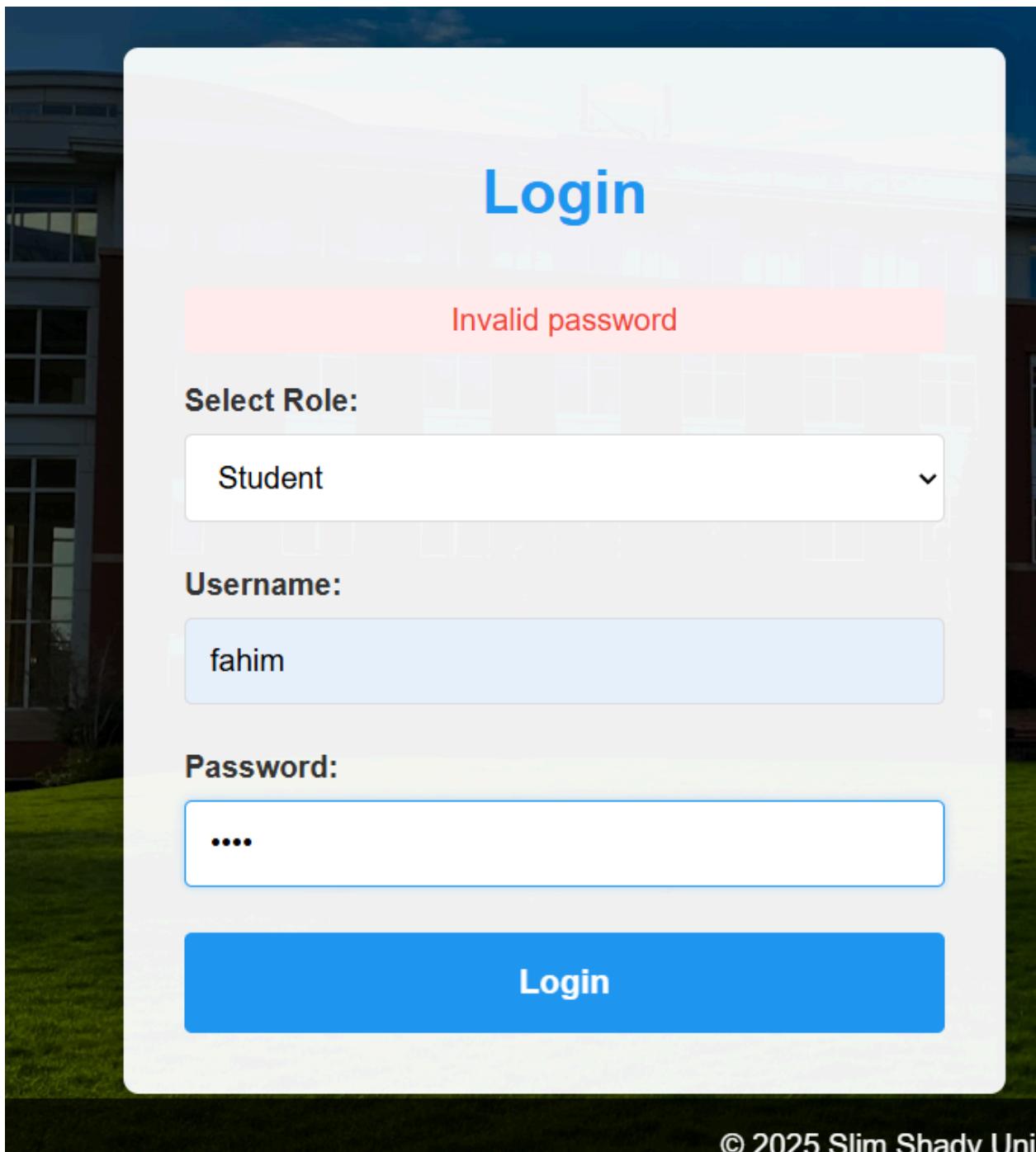
/**
 * Unlock account
 * @return void
 */
public function unlockAccount(): void {
    $this->accountLocked = false;
    $this->loginAttempts = 0;
}

/**
 * Abstract login method to be implemented by child classes
 * @param string $password
 * @return bool
 */
abstract public function login($password): bool;

/**
 * Abstract logout method to be implemented by child classes
 * @return void
 */
abstract public function logout(): void;

/**
 * Audit log for security tracking
 * @param string $action
 * @param string $details
 * @return void
 */
protected function logActivity($action, $details)
{
    $timestamp = new DateTime();
    $logEntry = sprintf(
        "[%s] User %s (%s): %s - %s\n",
        $timestamp->format('Y-m-d H:i:s'),
```

Screenshot of UI:



Security:Hash password used

	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	user_id	username	password
	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	8	fahim	\$2y\$10\$USwahbaVmwoHN/EaKiS5HuuhiOmX7p/ZsN.QA8WKnUt...
	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	9	Hasib	\$2y\$10\$vdgjyQWmmJqy40S2mYByE.hihuRX3TaKCmwMzJBwf9Y...
	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	10	Raiyan	\$2y\$10\$fS/yhlnGZxE10Qy8MmE2LOSLNodcbkrb/W1f0fZQHvH...
	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	11	Rafsun	\$2y\$10\$KO.0zCaTYm/3MDgnYDfhNuYKFsbZ67HbTyVfdSfpX6f...
	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	12	Rifat	\$2y\$10\$.UVGeo6AKrlPEH1UGwjLuQWMLK6A.dFSTh8TgdUun6...
	<input type="checkbox"/>	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	13	Saif	\$2y\$10\$Eht0fUYNH2siyxBWrZ5x4OSSV/eJYZkNMlriZi.8GV6Z...



Check all

With selected:



Large number of Users:

Login

Invalid password

Select Role:

Student

Username:

Password:

Hasib
Rafsun
Rifat
Saif
Raiyan
fahim

Register

Home

Username:

Password:

Email:

First Name:

Last Name:

Role:

Student

© 2025 Slim Shady University. All rights reserved.

1.CourseFactory design pattern:

```
src > classes > UserFactory.php > ...
    2 references | 0 implementations
6   class UserFactory {
    2 references | 0 overrides
7     public static function createUser($type, $userData): Admin|Instructor|Student {
8       switch ($type) {
9         case 'student':
10           return new Student(
11             $userData['username'],
12             $userData['email'],
13             $userData['password'],
14             $userData['studentId'],
15             $userData['major']
16           );
17
18         case 'instructor':
19           return new Instructor(
20             $userData['username'],
21             $userData['email'],
22             $userData['password'],
23             $userData['department']
24           );
25
26         case 'admin':
27           return new Admin(
28             $userData['username'],
29             $userData['email'],
30             $userData['password']
31           );
32
33         default:
34           throw new Exception(message: "Invalid user type: {$type}");
35
36       }
37     }
```

2.StudentObserver Design pattern:

```
1 <?php
2
3     1 reference | 1 implementation
4     interface CourseSubject {
5         0 references | 1 override
6         public function registerStudent(Student $student);
7         0 references | 1 override
8         public function removeStudent(Student $student);
9         1 reference | 1 override
10        public function notifyStudents($message);
11    }
```

src > classes > interfaces > 🏫 StudentObserver.php > ...

```
1 <?php
2
3     0 references | 0 implementations
4     interface StudentObserver {
5         0 references | 0 overrides
6         public function update($message);
7     }
```

3.DatabaseSingleton design pattern:

```
3   class DatabaseSingleton {
4       private function __construct() {
5           '
6           $this->connection->set_charset(charset: "utf8mb4");
7       } catch (Exception $e) {
8           error_log(message: "Database Connection Error: " . $e->getMessage());
9           throw $e;
10      }
11  }
12
13  6 references | 0 overrides
14  public static function getInstance(): DatabaseSingleton {
15      if (self::$instance === null) {
16          self::$instance = new self();
17      }
18      return self::$instance;
19  }
20
21
22  6 references | 0 overrides
23  public function getConnection(): mysqli {
24      return $this->connection;
25  }
26
27  // Prevent cloning
28  0 references
29  private function __clone(): void {}
30
31  // Prevent unserialization
32  0 references
33  private function __wakeup(): void {}
34
35
36  0 references | 0 overrides
37  public function __destruct() {
38      if ($this->connection) {
39          $this->connection->close();
40      }
41  }
42 }
```

4.CourseDecorator:

```
design_patterns > CourseDecorator.php > ...
35  abstract class CourseDecorator implements CourseInterface {
36
37      3 references | 2 overrides
38      public function getCost(): float {
39          return $this->course->getCost();
40      }
41
42      2 references | 1 override
43      public function getCredits(): int {
44          return $this->course->getCredits();
45      }
46
47  }
48
49
50
51
52
53
54
55  class LabComponent extends CourseDecorator {
56      3 references | 0 overrides | prototype
57      public function getDescription(): string {
58          return $this->course->getDescription() . " + Laboratory";
59      }
60
61      3 references | 0 overrides | prototype
62      public function getCost(): float {
63          return $this->course->getCost() + 500.00;
64      }
65
66      2 references | 0 overrides | prototype
67      public function getCredits(): int {
68          return $this->course->getCredits() + 1;
69      }
70
71
72
73
74
75
76  class OnlineMaterialsComponent extends CourseDecorator {
77      3 references | 0 overrides | prototype
78      public function getDescription(): string {
79          return $this->course->getDescription() . " + Online Materials";
80      }
81
82      3 references | 0 overrides | prototype
83      public function getCost(): float {
84          return $this->course->getCost() + 200.00;
85      }
86
87
88
89
90
91
92
93
94
95
96
```

In 1, Col 1 - Space

3.Explanation:

1. Singleton Pattern:

- Used for database connection management
- Ensures only one database instance exists throughout the application
- Implemented in `DatabaseSingleton.php`.
- Provides global access point while preventing multiple instances

2. Decorator Pattern:

- Used for adding features to courses dynamically
- Base course can be enhanced with lab components or online materials
- Implemented in `CourseDecorator.php`.
- Allows flexible combination of course features without subclassing

3. Factory Pattern:

- Used for creating different types of courses
- Centralizes course creation logic in one place
- Implemented in `CourseFactory.php`
- Supports creation of standard and lab courses with specific requirements

4. Observer Pattern:

- Used for course-related notifications and logging
- Automatically notifies interested parties about course changes
- Implemented in `CourseObserver.php`
- Currently supports email notifications and database logging

All design patterns in the system are integrated through a single database connection managed by the Singleton pattern. The Factory pattern is responsible for creating base course objects, which the Decorator pattern then enhances with additional features. Any changes made to the courses trigger the

Observer pattern, which sends notifications to relevant users. Throughout this process, all database operations—such as creating, updating, and logging course data—are handled through the Singleton-managed connection. Error handling is distributed among the patterns: the Factory validates course parameters, the Decorator checks for compatible feature combinations, the Observer ensures proper notification delivery, and the Singleton manages any connection issues. The system is easily extensible; new course types can be added using the Factory, new features through the Decorator, and additional notification types via the Observer. From a user's perspective, Admins create courses, features are applied automatically, and notifications are delivered—all seamlessly connected through the unified database setup.

4. Class and Sequence diagram matches with my implementation:

Screenshot of Code:

1.Student.php:

```
classes > Student.php > Student > requestCourseRegistration()
2  class Student {
3      1 reference
4      private $email;
5
6      0 references | 0 overrides
7      public function getStudentInfo(): array {
8          return [
9              'id' => $this->studentId,
10             'name' => $this->name,
11             'email' => $this->email
12         ];
13     }
14
15     0 references | 0 overrides
16     public function requestCourseRegistration($courseId): bool {
17         // Step 1 in sequence diagram
18         return true;
19     }
20
21     0 references | 0 overrides
22     public function viewRegisteredCourses(): array {
23         return []; // Returns list of registered courses
24     }
25
26     0 references | 0 overrides
27     public function getGrades(): array {
28         // Step 12 in sequence diagram
29         return []; // Returns grades
30     }
31
32     0 references | 0 overrides
33     public function viewSchedule(): array {
34         return []; // Returns student's schedule
35     }
36 }
```

2.Instructor.php:

```
classes > 🐾 Instructor.php > ...
1  <?php
2  class Instructor {
3      private $instructorId;
4      private $name;
5      private $email;
6
7      0 references | 0 overrides
8      public function getInstructorInfo(): array {
9          return [
10             'id' => $this->instructorId,
11             'name' => $this->name,
12             'email' => $this->email
13         ];
14     }
15
16     0 references | 0 overrides
17     public function assignGrades($studentId, $courseId, $grade): bool {
18         // Step 11 in sequence diagram
19         return true;
20     }
21
22     0 references | 0 overrides
23     public function viewAssignedCourses(): array {
24         return []; // Returns list of assigned courses
25     }
26
27     0 references | 0 overrides
28     public function getSchedule(): array {
29         return []; // Returns instructor's schedule
30     }
31
32     0 references | 0 overrides
33     public function requestCourseAssignment($courseId): bool {
34         // Step 8 in sequence diagram
35         return true;
36     }
```

3.Admin.php:

```
classes > Admin.php > ...
2  class Admin {
3      private $adminId;
4      0 references
5      private $name;
6
7          0 references | 0 overrides
8      public function validateStudentDetails($studentId): bool {
9          // Step 2 in sequence diagram
10         return true;
11     }
12
13     0 references | 0 overrides
14     public function assignInstructor($instructorId, $courseId): bool {
15         // Step 9 in sequence diagram
16         return true;
17     }
18
19     0 references | 0 overrides
20     public function getSchedule($courseId): array {
21         return []; // Returns course schedule
22     }
23
24     0 references | 0 overrides
25     public function getDepartmentDetails($departmentId): array {
26         // Step 14 in sequence diagram
27         return [];
28     }
29
30     0 references | 0 overrides
31     public function manageDepartmentDetails($departmentId, $details): bool {
32         return true;
33     }
34
35     0 references | 0 overrides
36     public function updateCourseDetails($courseId, $details): bool {
37         // Step 16 in sequence diagram
38         return true;
39     }
40
41     0 references | 0 overrides
42     public function notifyInstructor($instructorId, $message): bool {
```

Ln 1, Col 1 Spaces: 4 UTF-8

4.Registration.php:

```
classes > Registration.php > ...
1  <?php
2  0 references | 0 implementations
3  class Registration {
4      0 references
5      private $registrationId;
6      0 references
7      private $status;
8
9      0 references | 0 overrides
10     public function checkCourseAvailability($courseId): bool {
11         // Step 3 in sequence diagram
12         return true;
13     }
14
15     0 references | 0 overrides
16     public function verifyScheduleAvailability($courseId, $studentId): bool {
17         // Step 4 in sequence diagram
18         return true;
19     }
20
21     0 references | 0 overrides
22     public function updateRegistrationStatus($studentId, $courseId, $status): bool {
23         // Step 6 in sequence diagram
24         return true;
25     }
26
27     0 references | 0 overrides
28     public function confirmRegistration($studentId, $courseId): bool {
29         // Step 7 in sequence diagram
30         return true;
31     }
32
33     0 references | 0 overrides
34     public function getRegistrationDetails($registrationId): array {
35         return [];
36     }
37 }
```

5.Course.php:

```
classes > Course.php > ...
1  <?php
2  1 reference | 0 implementations
3  class Course {
4      3 references
5      private $courseId;
6      3 references
7      private $courseName;
8      1 reference
9      private $description;
10
11     0 references | 0 overrides
12     public function getCourseDetails(): array {
13         return [
14             'id' => $this->courseId,
15             'name' => $this->courseName,
16             'description' => $this->description
17         ];
18     }
19
20     0 references | 0 overrides
21     public function getSchedule(): array {
22         return []; // Returns course schedule
23     }
24
25     0 references | 0 overrides
26     public function updateCourse($details): bool {
27         return true;
28     }
29
30     0 references | 0 overrides
31     public function offerCourse(): bool {
32         // Step 15 in sequence diagram
33         return true;
34     }
35
36     0 references | 0 overrides
37     public function updateCoursePlan($plan): bool {
38         // Step 16 in sequence diagram
39         return true;
40     }
41
42     0 references | 0 overrides
43     public function cancelOffer(): bool {
44         // Step 17 in sequence diagram
45         return true;
46     }
47
48     0 references | 0 overrides
49     public function addStudent($studentId): bool {
50         // Step 18 in sequence diagram
51         return true;
52     }
53
54     0 references | 0 overrides
55     public function removeStudent($studentId): bool {
56         // Step 19 in sequence diagram
57         return true;
58     }
59
60     0 references | 0 overrides
61     public function updateStudentGrade($grade): bool {
62         // Step 20 in sequence diagram
63         return true;
64     }
65
66     0 references | 0 overrides
67     public function printReportCard(): string {
68         // Step 21 in sequence diagram
69         return "Report Card Generated";
70     }
71
72     0 references | 0 overrides
73     public function generateBill(): string {
74         // Step 22 in sequence diagram
75         return "Bill Generated";
76     }
77
78     0 references | 0 overrides
79     public function updateBillStatus($status): bool {
80         // Step 23 in sequence diagram
81         return true;
82     }
83
84     0 references | 0 overrides
85     public function printBill(): string {
86         // Step 24 in sequence diagram
87         return "Bill Printed";
88     }
89
90     0 references | 0 overrides
91     public function updateBillStatus($status): bool {
92         // Step 25 in sequence diagram
93         return true;
94     }
95
96     0 references | 0 overrides
97     public function printBill(): string {
98         // Step 26 in sequence diagram
99         return "Bill Printed";
100    }
101
102   0 references | 0 overrides
103   public function updateBillStatus($status): bool {
104       // Step 27 in sequence diagram
105       return true;
106   }
107
108   0 references | 0 overrides
109   public function printBill(): string {
110       // Step 28 in sequence diagram
111       return "Bill Printed";
112   }
113
114   0 references | 0 overrides
115   public function updateBillStatus($status): bool {
116       // Step 29 in sequence diagram
117       return true;
118   }
119
120   0 references | 0 overrides
121   public function printBill(): string {
122       // Step 30 in sequence diagram
123       return "Bill Printed";
124   }
125
126   0 references | 0 overrides
127   public function updateBillStatus($status): bool {
128       // Step 31 in sequence diagram
129       return true;
130   }
131
132   0 references | 0 overrides
133   public function printBill(): string {
134       // Step 32 in sequence diagram
135       return "Bill Printed";
136   }
137
138   0 references | 0 overrides
139   public function updateBillStatus($status): bool {
140       // Step 33 in sequence diagram
141       return true;
142   }
143
144   0 references | 0 overrides
145   public function printBill(): string {
146       // Step 34 in sequence diagram
147       return "Bill Printed";
148   }
149
150   0 references | 0 overrides
151   public function updateBillStatus($status): bool {
152       // Step 35 in sequence diagram
153       return true;
154   }
155
156   0 references | 0 overrides
157   public function printBill(): string {
158       // Step 36 in sequence diagram
159       return "Bill Printed";
160   }
161
162   0 references | 0 overrides
163   public function updateBillStatus($status): bool {
164       // Step 37 in sequence diagram
165       return true;
166   }
167
168   0 references | 0 overrides
169   public function printBill(): string {
170       // Step 38 in sequence diagram
171       return "Bill Printed";
172   }
173
174   0 references | 0 overrides
175   public function updateBillStatus($status): bool {
176       // Step 39 in sequence diagram
177       return true;
178   }
179
180   0 references | 0 overrides
181   public function printBill(): string {
182       // Step 40 in sequence diagram
183       return "Bill Printed";
184   }
185
186   0 references | 0 overrides
187   public function updateBillStatus($status): bool {
188       // Step 41 in sequence diagram
189       return true;
190   }
191
192   0 references | 0 overrides
193   public function printBill(): string {
194       // Step 42 in sequence diagram
195       return "Bill Printed";
196   }
197
198   0 references | 0 overrides
199   public function updateBillStatus($status): bool {
200       // Step 43 in sequence diagram
201       return true;
202   }
203
204   0 references | 0 overrides
205   public function printBill(): string {
206       // Step 44 in sequence diagram
207       return "Bill Printed";
208   }
209
210   0 references | 0 overrides
211   public function updateBillStatus($status): bool {
212       // Step 45 in sequence diagram
213       return true;
214   }
215
216   0 references | 0 overrides
217   public function printBill(): string {
218       // Step 46 in sequence diagram
219       return "Bill Printed";
220   }
221
222   0 references | 0 overrides
223   public function updateBillStatus($status): bool {
224       // Step 47 in sequence diagram
225       return true;
226   }
227
228   0 references | 0 overrides
229   public function printBill(): string {
230       // Step 48 in sequence diagram
231       return "Bill Printed";
232   }
233
234   0 references | 0 overrides
235   public function updateBillStatus($status): bool {
236       // Step 49 in sequence diagram
237       return true;
238   }
239
240   0 references | 0 overrides
241   public function printBill(): string {
242       // Step 50 in sequence diagram
243       return "Bill Printed";
244   }
245
246   0 references | 0 overrides
247   public function updateBillStatus($status): bool {
248       // Step 51 in sequence diagram
249       return true;
250   }
251
252   0 references | 0 overrides
253   public function printBill(): string {
254       // Step 52 in sequence diagram
255       return "Bill Printed";
256   }
257
258   0 references | 0 overrides
259   public function updateBillStatus($status): bool {
260       // Step 53 in sequence diagram
261       return true;
262   }
263
264   0 references | 0 overrides
265   public function printBill(): string {
266       // Step 54 in sequence diagram
267       return "Bill Printed";
268   }
269
270   0 references | 0 overrides
271   public function updateBillStatus($status): bool {
272       // Step 55 in sequence diagram
273       return true;
274   }
275
276   0 references | 0 overrides
277   public function printBill(): string {
278       // Step 56 in sequence diagram
279       return "Bill Printed";
280   }
281
282   0 references | 0 overrides
283   public function updateBillStatus($status): bool {
284       // Step 57 in sequence diagram
285       return true;
286   }
287
288   0 references | 0 overrides
289   public function printBill(): string {
290       // Step 58 in sequence diagram
291       return "Bill Printed";
292   }
293
294   0 references | 0 overrides
295   public function updateBillStatus($status): bool {
296       // Step 59 in sequence diagram
297       return true;
298   }
299
299 1 in 1 Col 1 Spaces
```

6.Department.php:

```
classes > 🐘 Department.php > ...
1  <?php
2  0 references | 0 implementations
3  class Department {
4      1 reference
5      private $departmentId;
6      1 reference
7      private $name;
8
9          0 references | 0 overrides
10     public function getDepartmentDetails(): array {
11         return [
12             'id' => $this->departmentId,
13             'name' => $this->name
14         ];
15     }
16
17     0 references | 0 overrides
18     public function updateDepartment($details): bool {
19         return true;
20     }
21
22     0 references | 0 overrides
23     public function offerNewCourses($courses): bool {
24         // Step 15 in sequence diagram
25         return true;
26     }
27
28     0 references | 0 overrides
29     public function updateCoursePlan($courseId, $plan): bool {
30         // Step 16 in sequence diagram
31         return true;
32     }
33 }
```

7.Schedule.php:

```
classes > 🐾 Schedule.php > ...
1  <?php
2  0 references | 0 implementations
3  class Schedule {
4      1 reference
5      private $scheduleId;
6      1 reference
7      private $day;
8      1 reference
9      private $time;
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
```

0 references | 0 overrides

```
    public function getScheduleDetails(): array {
        return [
            'id' => $this->scheduleId,
            'day' => $this->day,
            'time' => $this->time
        ];
    }

0 references | 0 overrides
    public function setSchedule($details): bool {
        return true;
    }

0 references | 0 overrides
    public function confirmSchedule(): bool {
        // Step 5 in sequence diagram
        return true;
    }

0 references | 0 overrides
    public function updateSchedule($details): bool {
        return true;
    }
```

Explanation:

Matches with Implementation:

- 1. Class Structure** - Admin, Student, and Instructor classes are maintained - Course and Department entities are present - Schedule management is integrated - Basic relationships between classes are preserved
- 2. Core Functionality** - Course registration process - Grade management system - Department management - Course scheduling system

Changes/Additions in Implementation:

- 1. Design Pattern Integration** - Added `DatabaseSingleton` (not in original diagram) - Introduced `CourseDecorator` for extended course features - Added `CourseFactory` for course creation - Implemented `CourseObserver` for notifications
- 2. Sequence Flow Modifications** - Added more robust validation through Factory pattern - Enhanced course creation with Builder pattern - Expanded notification system with Observer pattern - Centralized database operations with Singleton
- 3. Additional Features** - Course feature decoration (lab components, online materials) - More sophisticated notification system - Centralized course creation - Enhanced error handling

I need to Update Diagrams:

- 1. Class Diagram Updates Needed** - Add design pattern classes - Show relationships between pattern implementations - Include new interfaces - Update method signatures
- 2. Sequence Diagram Updates Needed** - Add notification flows - Include course creation steps - Show database singleton usage - Include decorator pattern interactions

5.Github link:

<https://github.com/fahimsakib007/University-Course-Registration-System.git>