Project : University Course Registration System

Groups Members Details:

Name: Shakib Alom Fahim Student
ID: 2222866042

1.Proposed Functional Requirements:

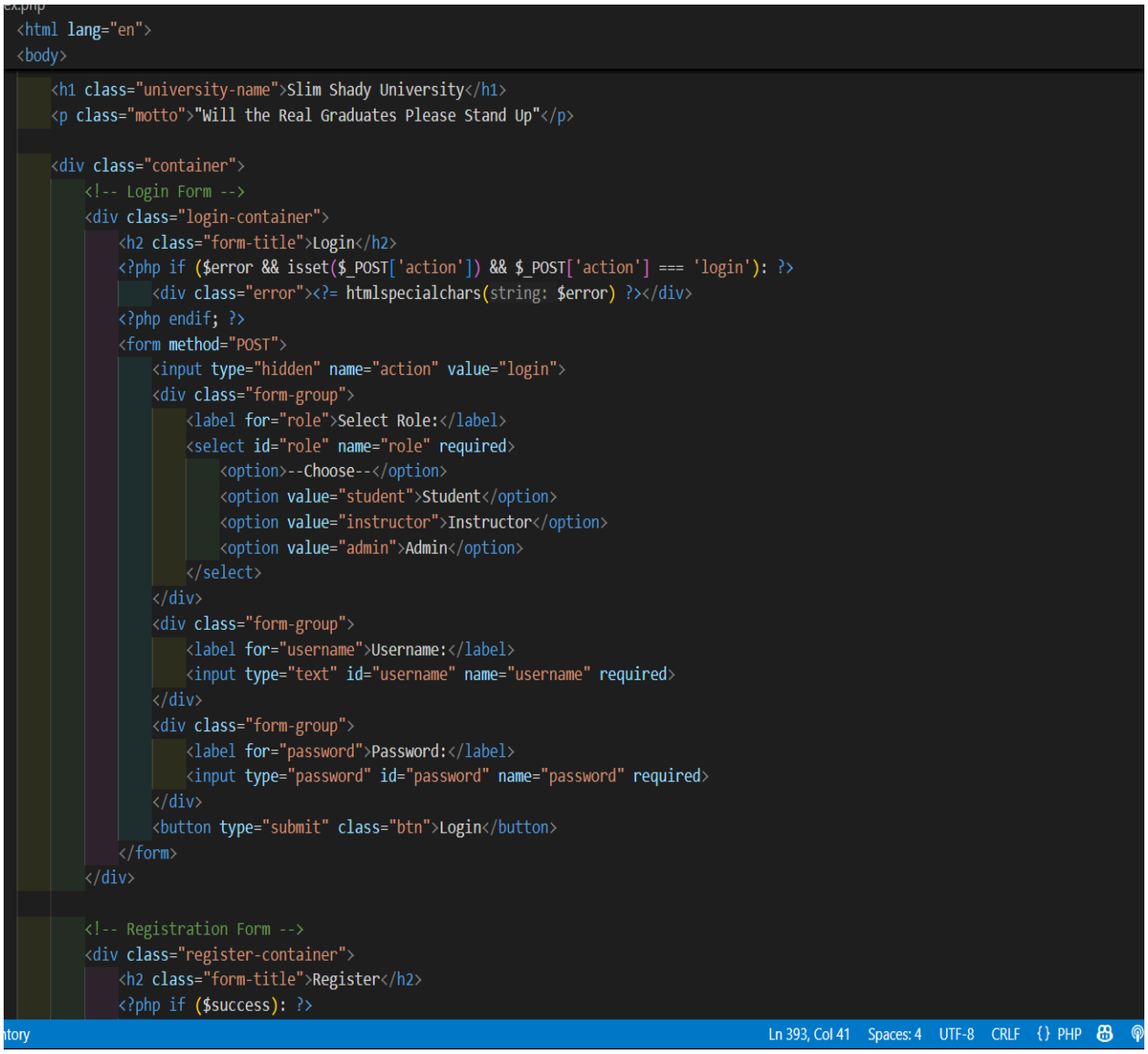
Functional Requirements/Specifications

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:

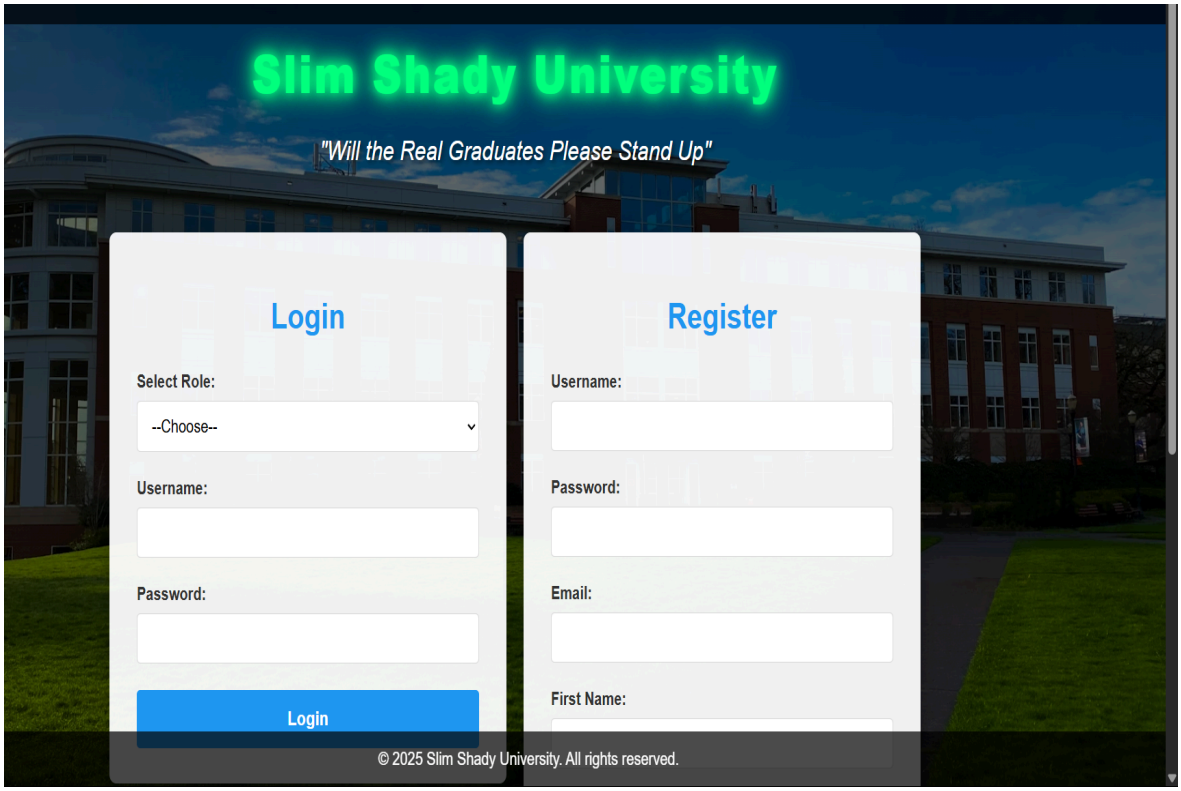
Screenshot of code index.php:

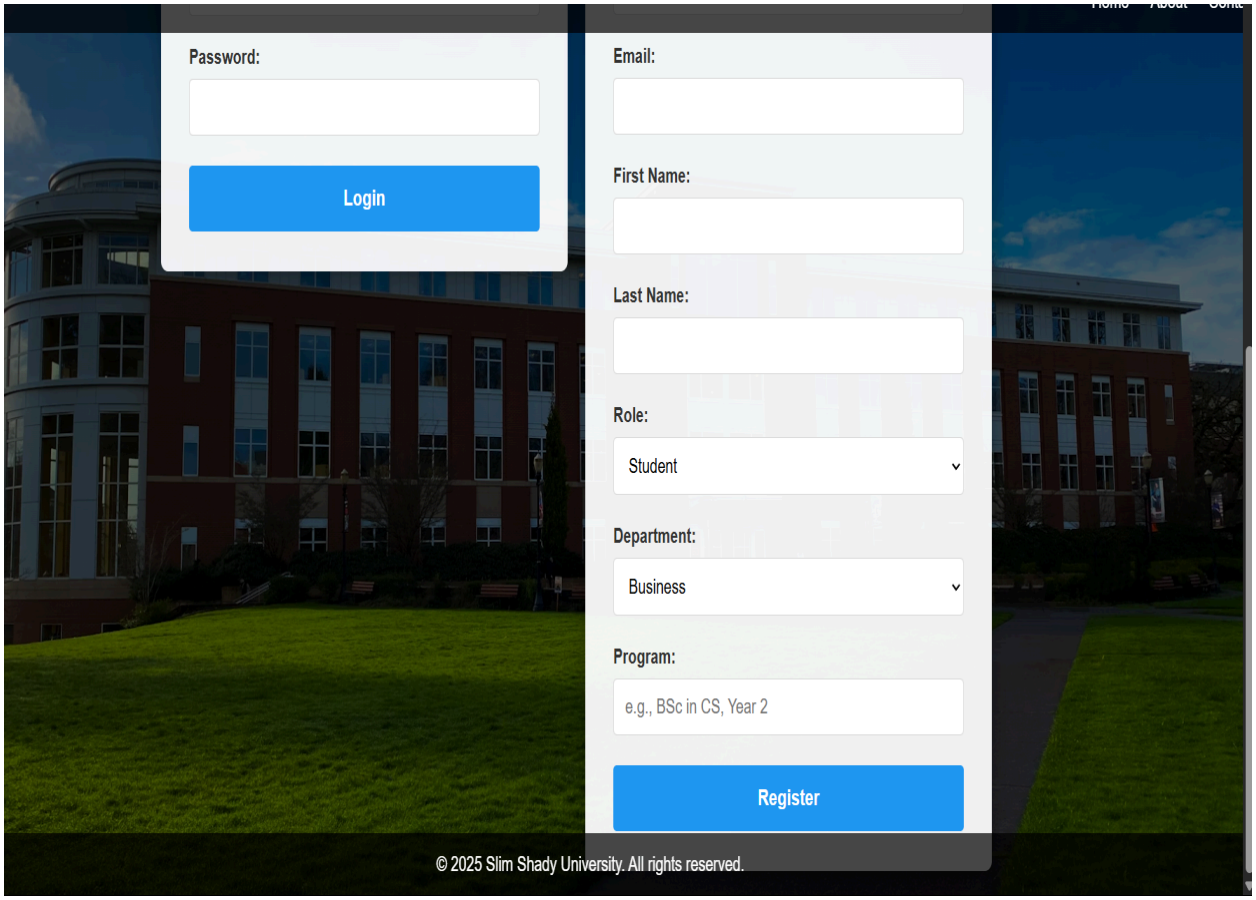


```
<html lang="en">
<body>
    <h1 class="university-name">Slim Shady University</h1>
    <p class="motto">"Will the Real Graduates Please Stand Up"</p>

    <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): ?>
```

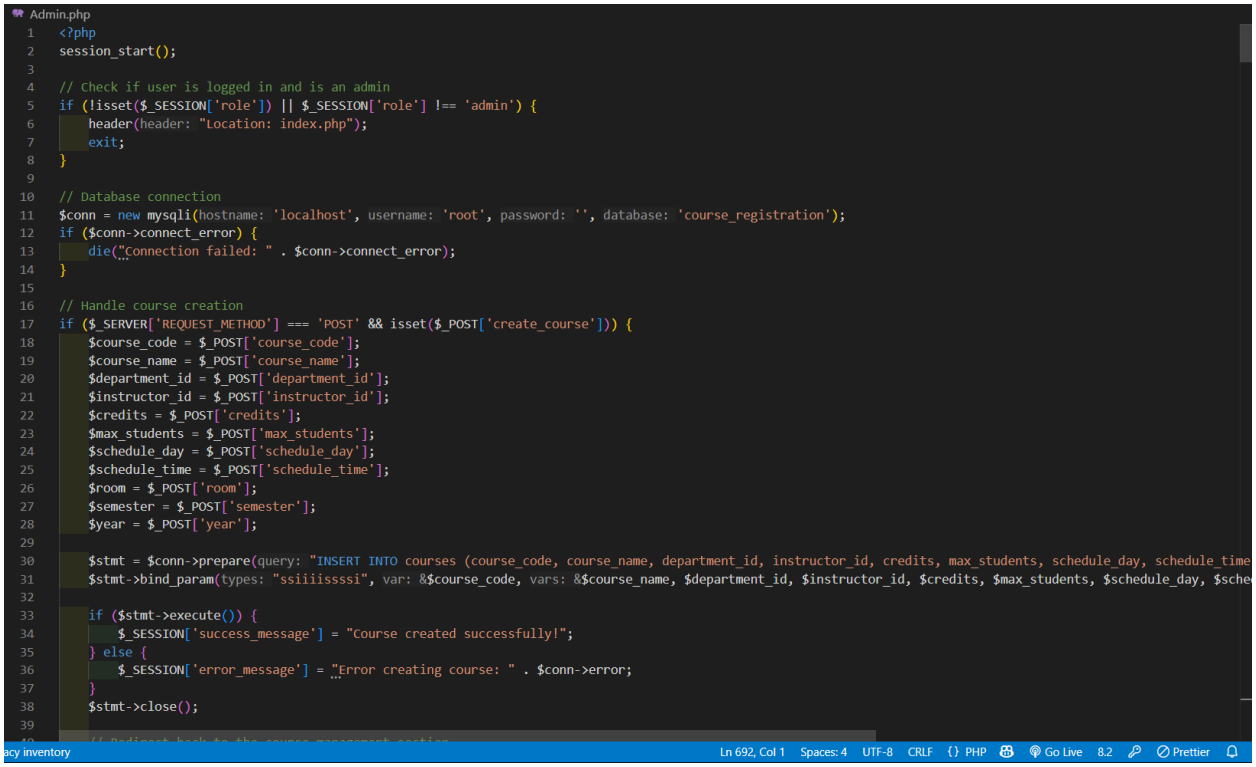
```
<body>
    <div class="container">
        <div class="register-container">
            <?php if ($success): ?>
                <div class="success"><?= htmlspecialchars(string: $success) ?></div>
            <?php endif; ?>
            <?php if ($error && isset($_POST['action']) && $_POST['action'] === '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>
```

Screenshot of UI:

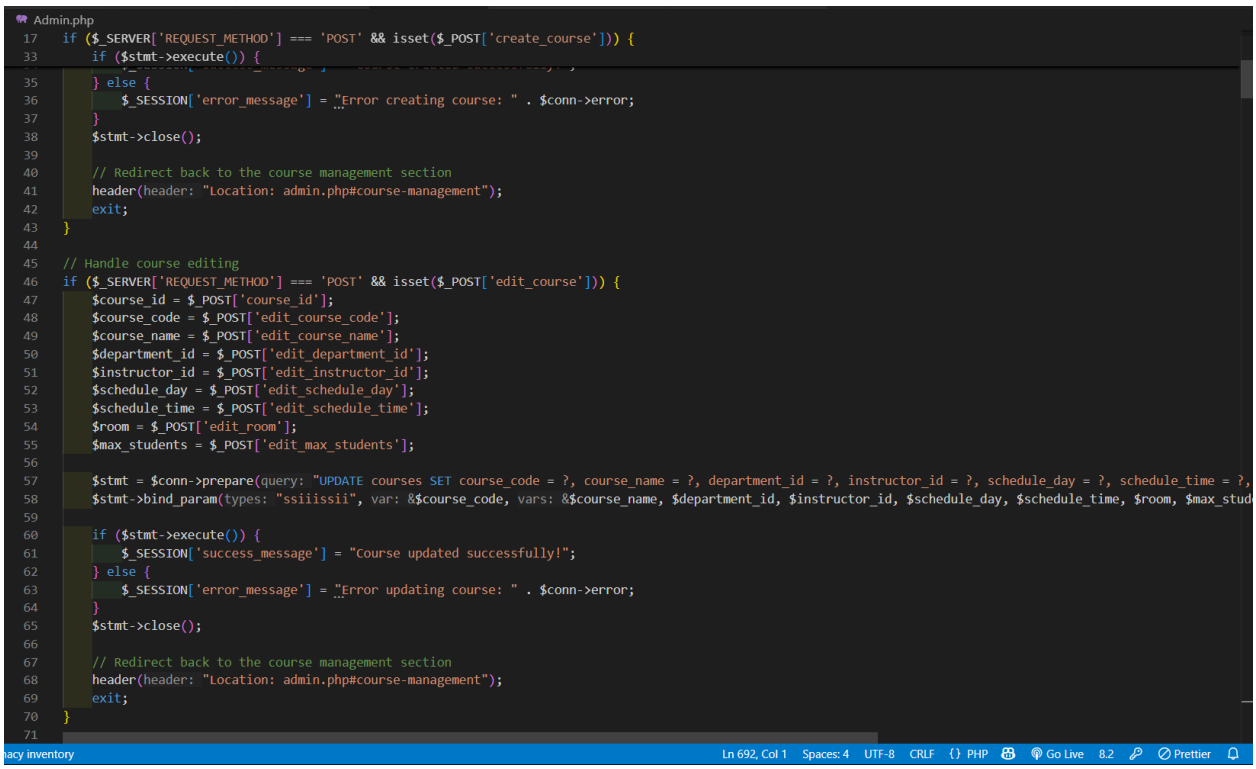



Password:
Login
Email:
First Name:
Last Name:
Role:
Student
Department:
Business
Program:
e.g., BSc in CS, Year 2
Register
© 2025 Slim Shady University. All rights reserved.

Screenshot of Code of Admin.php:

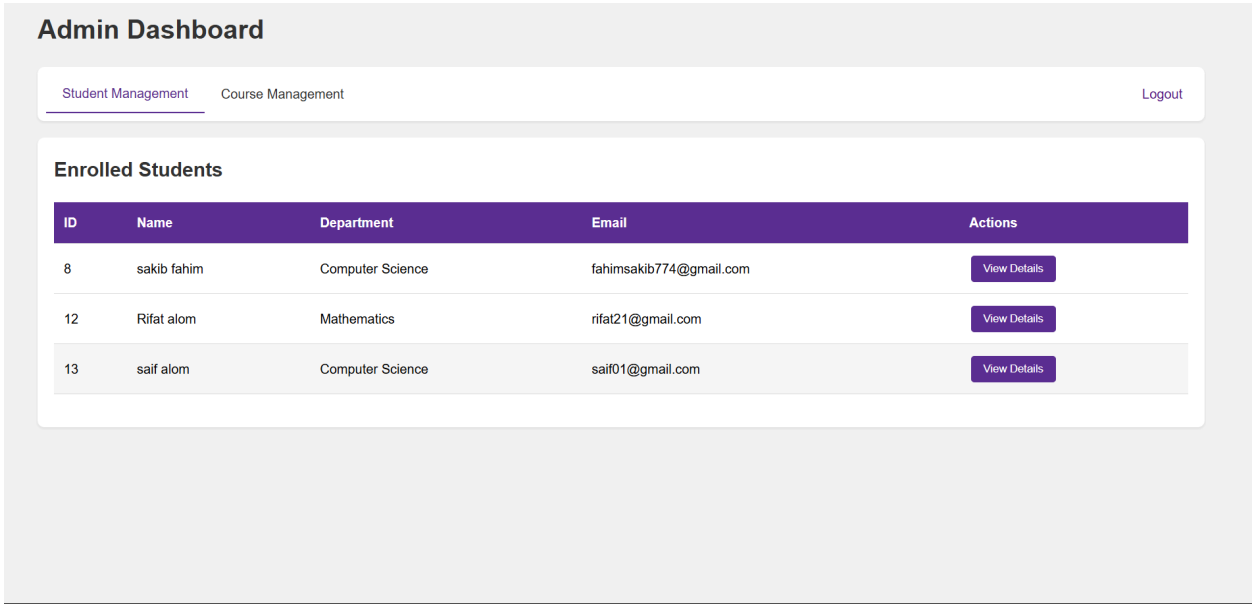


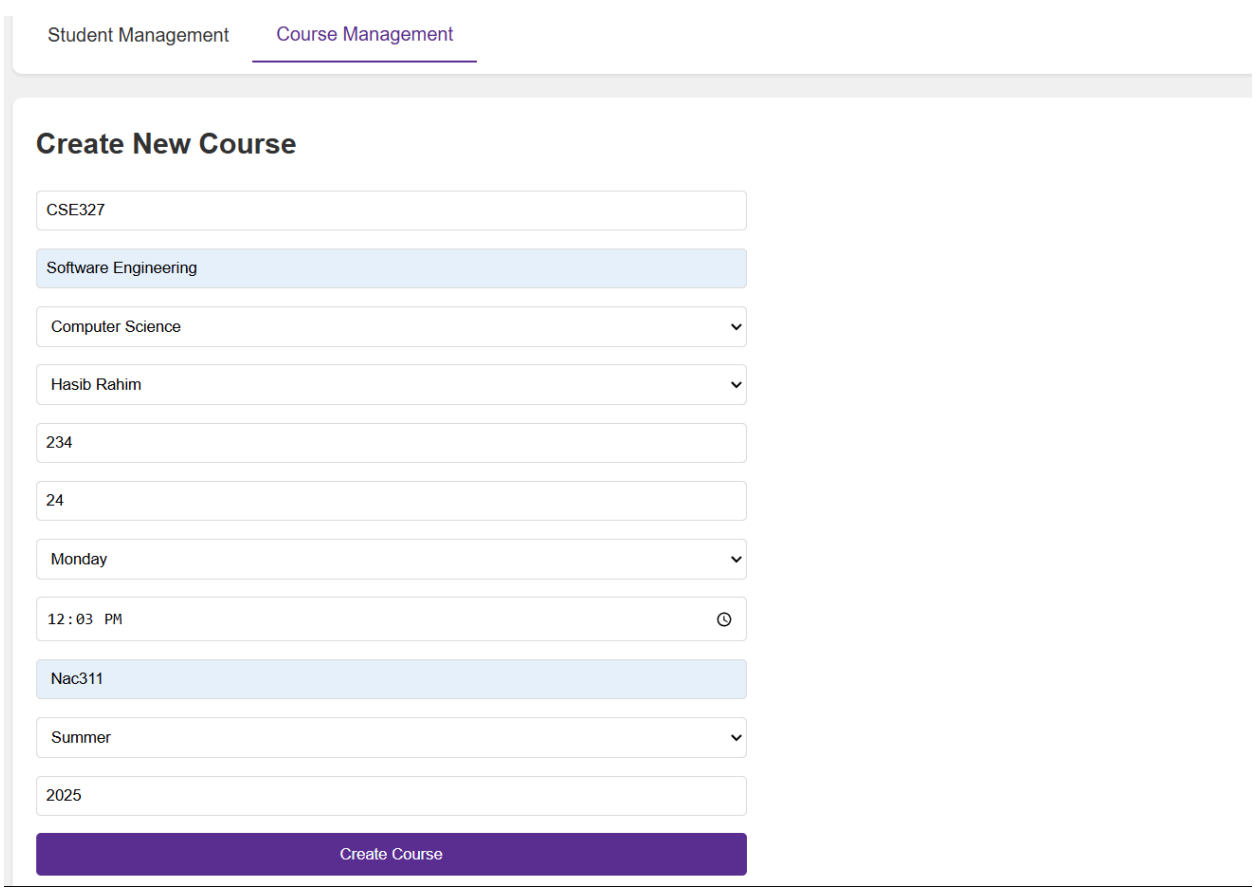
```php
<?php
session_start();

// Check if user is logged in and is an admin
if (!isset($_SESSION['role']) || $_SESSION['role'] !== 'admin') {
    header(header: "Location: index.php");
    exit;
}

// Database connection
$conn = new mysqli(hostname: 'localhost', username: 'root', password: '', database: 'course_registration');
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// Handle course creation
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['create_course'])) {
    $course_code = $_POST['course_code'];
    $course_name = $_POST['course_name'];
    $department_id = $_POST['department_id'];
    $instructor_id = $_POST['instructor_id'];
    $credits = $_POST['credits'];
    $max_students = $_POST['max_students'];
    $schedule_day = $_POST['schedule_day'];
    $schedule_time = $_POST['schedule_time'];
    $room = $_POST['room'];
    $semester = $_POST['semester'];
    $year = $_POST['year'];

    $stmt = $conn->prepare(query: "INSERT INTO courses (course_code, course_name, department_id, instructor_id, credits, max_students, schedule_day, schedule_time
    $stmt->bind_param(types: "ssiiiissssi", var: &$course_code, vars: &$course_name, $department_id, $instructor_id, $credits, $max_students, $schedule_day, $sche

    if ($stmt->execute()) {
        $_SESSION['success_message'] = "Course created successfully!";
    } else {
        $_SESSION['error_message'] = "Error creating course: " . $conn->error;
    }
    $stmt->close();
```

```php
Admin.php
    if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['create_course'])) {
        if ($stmt->execute()) {
    } else {
        $_SESSION['error_message'] = "Error creating course: " . $conn->error;
    }
    $stmt->close();

    // Redirect back to the course management section
    header(header: "Location: admin.php#course-management");
    exit;
}

// Handle course editing
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['edit_course'])) {
    $course_id = $_POST['course_id'];
    $course_code = $_POST['edit_course_code'];
    $course_name = $_POST['edit_course_name'];
    $department_id = $_POST['edit_department_id'];
    $instructor_id = $_POST['edit_instructor_id'];
    $schedule_day = $_POST['edit_schedule_day'];
    $schedule_time = $_POST['edit_schedule_time'];
    $room = $_POST['edit_room'];
    $max_students = $_POST['edit_max_students'];

    $stmt = $conn->prepare(query: "UPDATE courses SET course_code = ?, course_name = ?, department_id = ?, instructor_id = ?, schedule_day = ?, schedule_time = ?,
    $stmt->bind_param(types: "ssiiissii", var: &$course_code, vars: &$course_name, $department_id, $instructor_id, $schedule_day, $schedule_time, $room, $max_stud

    if ($stmt->execute()) {
        $_SESSION['success_message'] = "Course updated successfully!";
    } else {
        $_SESSION['error_message'] = "Error updating course: " . $conn->error;
    }
    $stmt->close();

    // Redirect back to the course management section
    header(header: "Location: admin.php#course-management");
    exit;
}
```

Screenshot of UI:

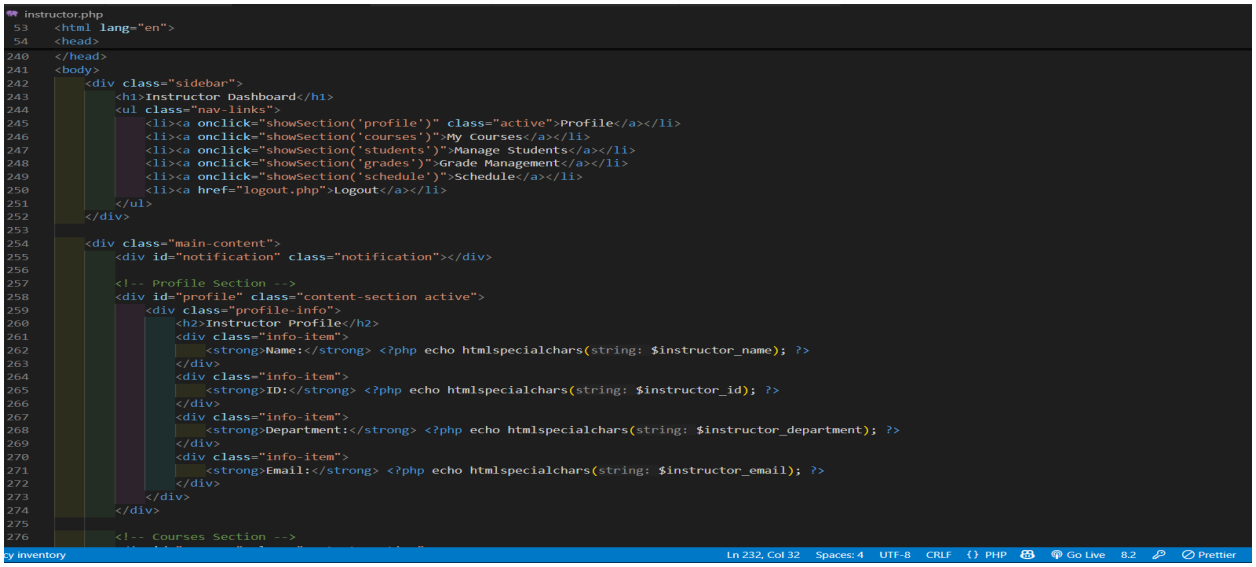




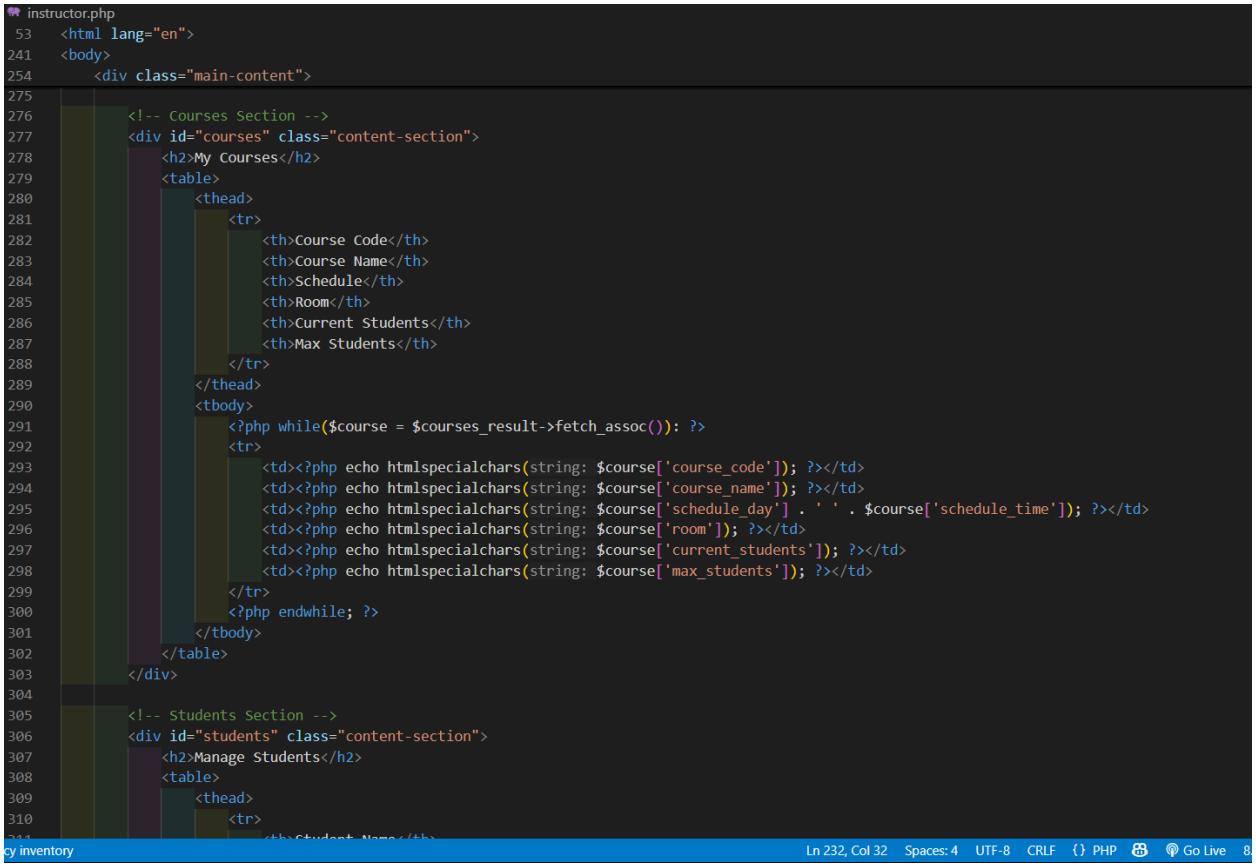
Course List

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

Screenshot of Code of Instructor.php:

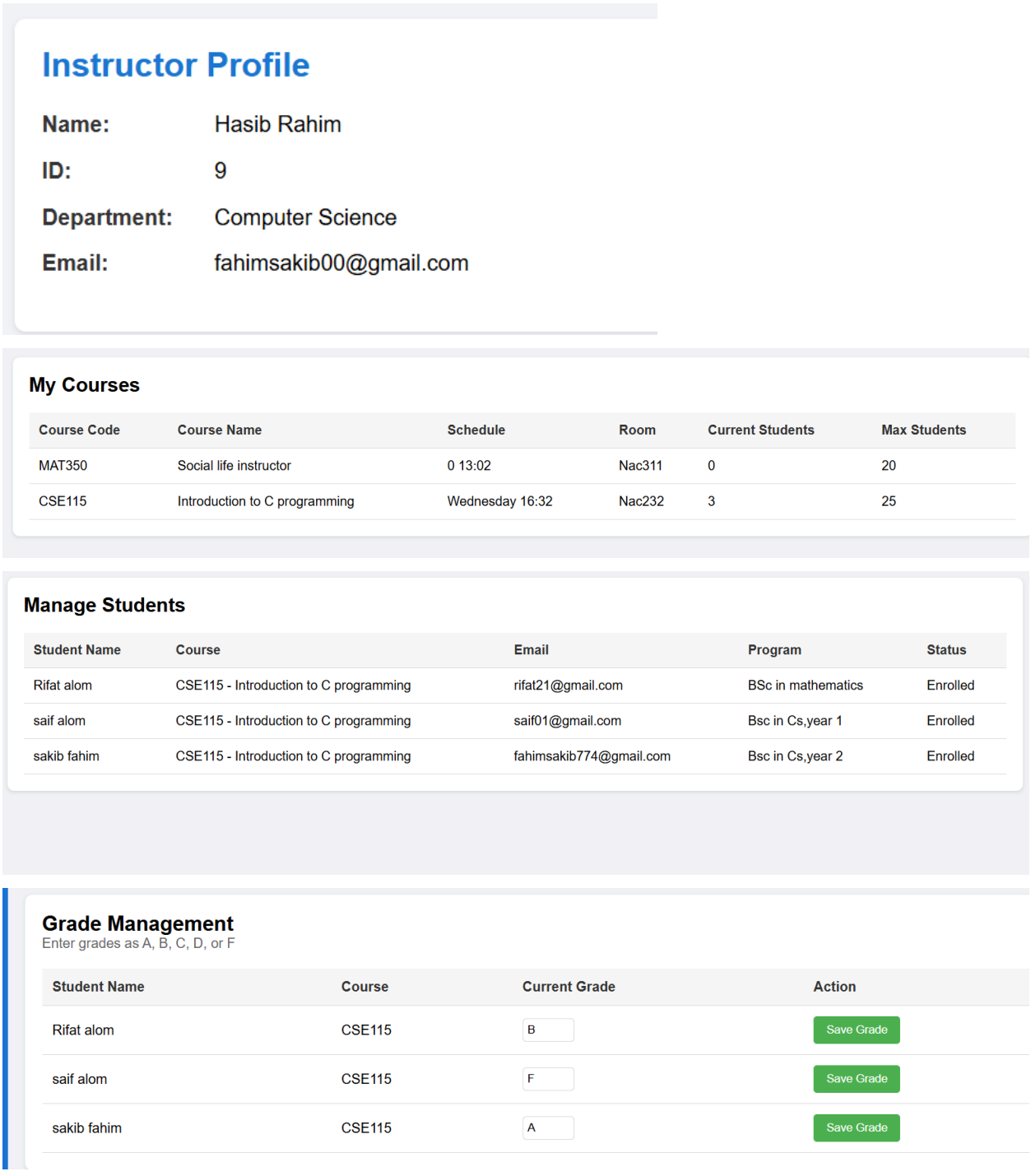


```
instructor.php
53    <html lang="en">
54    <head>
240   </head>
241   <body>
242       <div class="sidebar">
243           <h1>Instructor Dashboard</h1>
244           <ul class="nav-links">
245               <li><a onclick="showSection('profile')" class="active">Profile</a></li>
246               <li><a onclick="showSection('courses')">My Courses</a></li>
247               <li><a onclick="showSection('students')">Manage Students</a></li>
248               <li><a onclick="showSection('grades')">Grade Management</a></li>
249               <li><a onclick="showSection('schedule')">Schedule</a></li>
250               <li><a href="logout.php">Logout</a></li>
251           </ul>
252       </div>
253
254       <div class="main-content">
255           <div id="notification" class="notification"></div>
256
257           <!-- Profile Section -->
258           <div id="profile" class="content-section active">
259               <div class="profile-info">
260                   <h2>Instructor Profile</h2>
261                   <div class="info-item">
262                       <strong>Name:</strong> <?php echo htmlspecialchars(string: $instructor_name); ?>
263                   </div>
264                   <div class="info-item">
265                       <strong>ID:</strong> <?php echo htmlspecialchars(string: $instructor_id); ?>
266                   </div>
267                   <div class="info-item">
268                       <strong>Department:</strong> <?php echo htmlspecialchars(string: $instructor_department); ?>
269                   </div>
270                   <div class="info-item">
271                       <strong>Email:</strong> <?php echo htmlspecialchars(string: $instructor_email); ?>
272                   </div>
273               </div>
274           </div>
275
276           <!-- Courses Section -->
cy inventory                                  Ln 232, Col 32  Spaces: 4  UTF-8  CRLF  { } PHP  Go Live  8.2  Prettier
```



```
instructor.php
53    <html lang="en">
241   <body>
254       <div class="main-content">
275
276           <!-- Courses Section -->
277           <div id="courses" class="content-section">
278               <h2>My Courses</h2>
279               <table>
280                   <thead>
281                       <tr>
282                           <th>Course Code</th>
283                           <th>Course Name</th>
284                           <th>Schedule</th>
285                           <th>Room</th>
286                           <th>Current Students</th>
287                           <th>Max Students</th>
288                       </tr>
289                   </thead>
290                   <tbody>
291                       <?php while($course = $courses_result->fetch_assoc()): ?>
292                       <tr>
293                           <td><?php echo htmlspecialchars(string: $course['course_code']); ?></td>
294                           <td><?php echo htmlspecialchars(string: $course['course_name']); ?></td>
295                           <td><?php echo htmlspecialchars(string: $course['schedule_day'] . ' ' . $course['schedule_time']); ?></td>
296                           <td><?php echo htmlspecialchars(string: $course['room']); ?></td>
297                           <td><?php echo htmlspecialchars(string: $course['current_students']); ?></td>
298                           <td><?php echo htmlspecialchars(string: $course['max_students']); ?></td>
299                       </tr>
300                       <?php endwhile; ?>
301                   </tbody>
302               </table>
303           </div>
304
305           <!-- Students Section -->
306           <div id="students" class="content-section">
307               <h2>Manage Students</h2>
308               <table>
309                   <thead>
310                       <tr>
cy inventory                                  Ln 232, Col 32  Spaces: 4  UTF-8  CRLF  { } PHP  Go Live  8
```

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

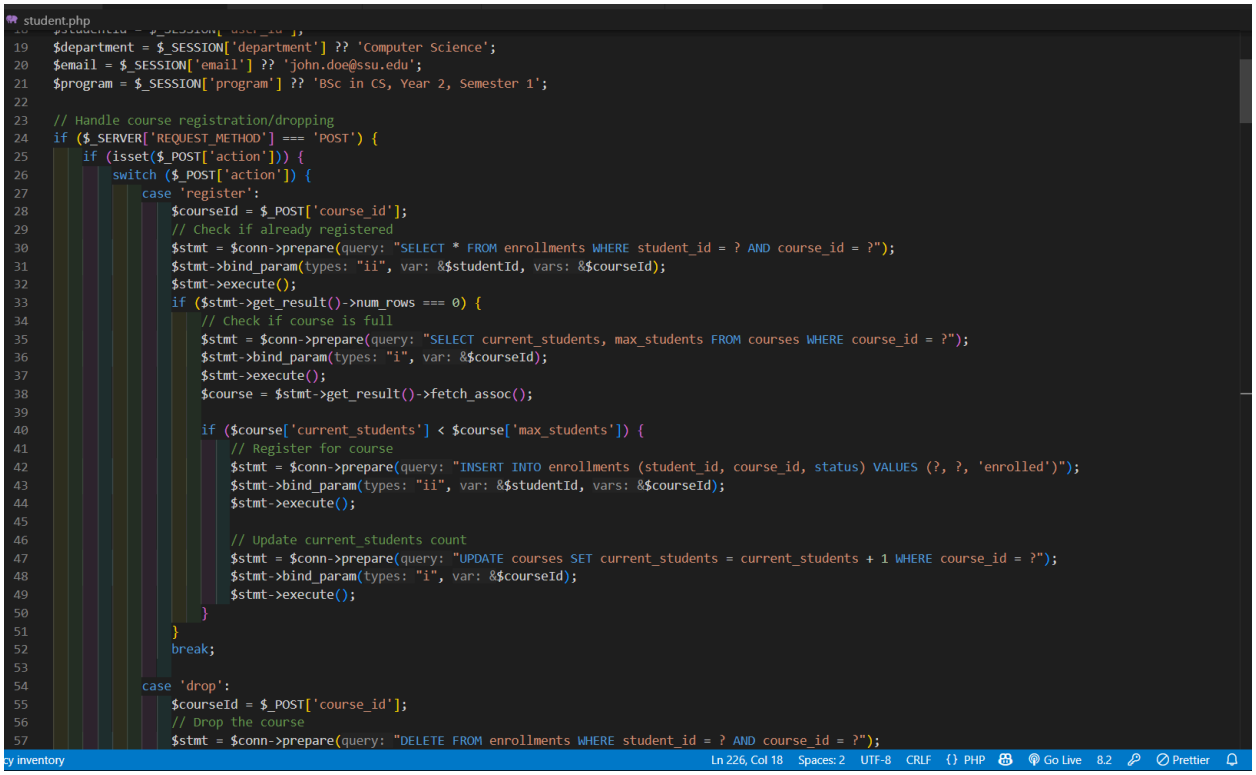
Student Name	Course	Current Grade	Action
Rifat alom	CSE115	B	Save Grade
saif alom	CSE115	F	Save Grade
sakib fahim	CSE115	A	Save Grade

Teaching Schedule

MAT350
Social life instructor
0 13:02
Room: Nac311

CSE115
Introduction to C programming
Wednesday 16:32
Room: Nac232

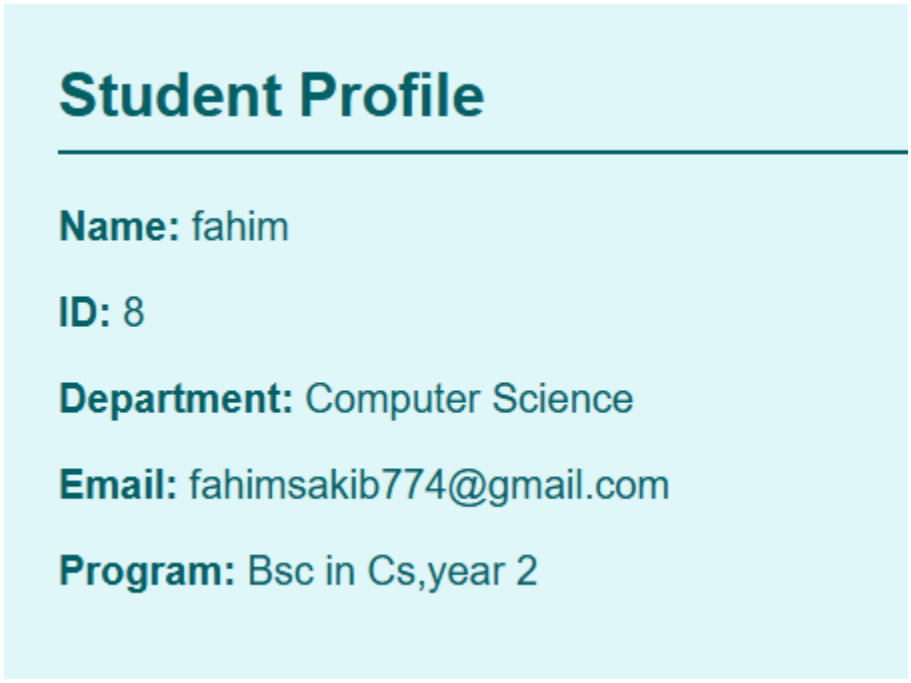
Screenshot of Code of Student.php:



student.php

```
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:



Course Registration

Available Courses

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

My Registered Courses

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

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:

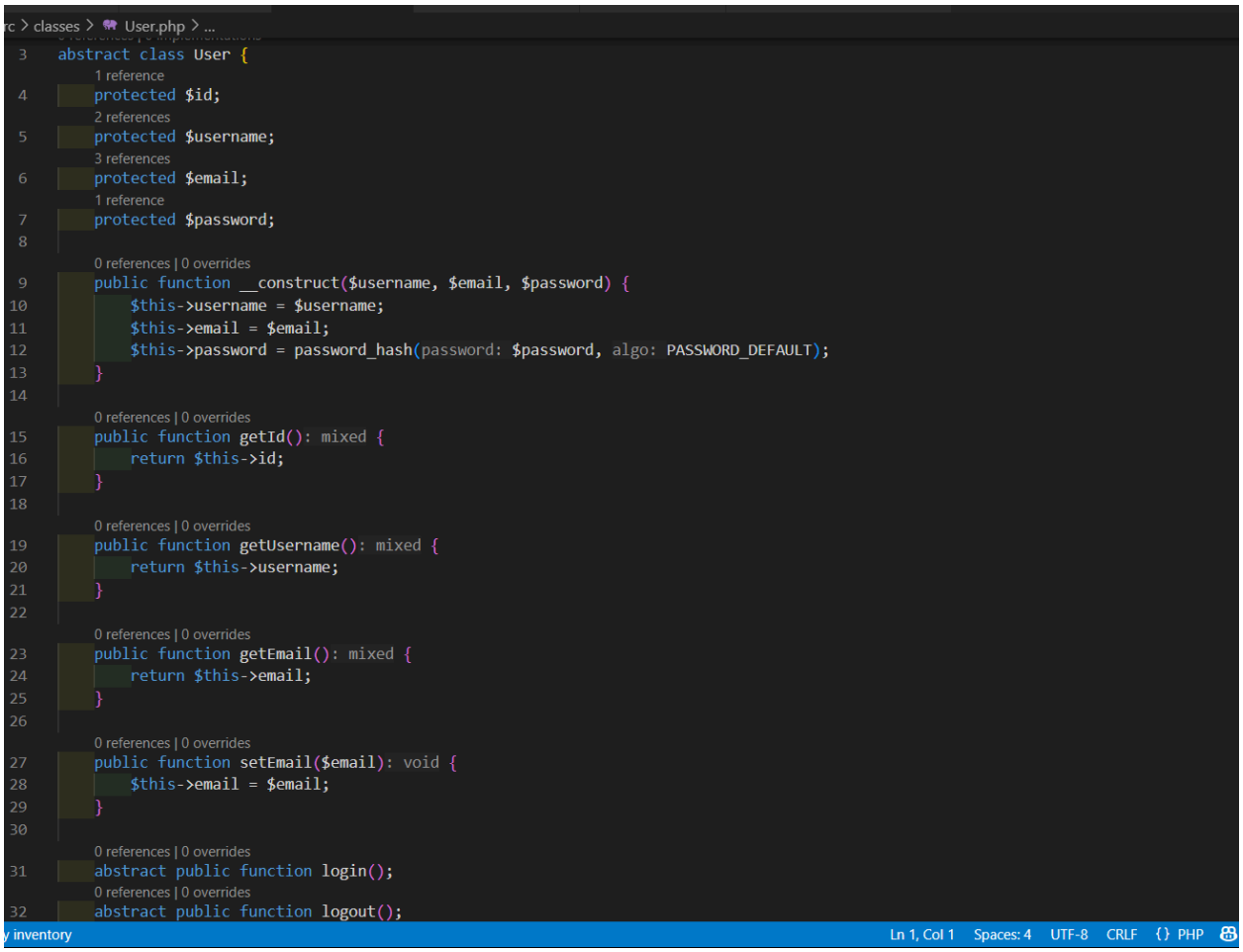
Non-Functional Requirements/Specifications

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:

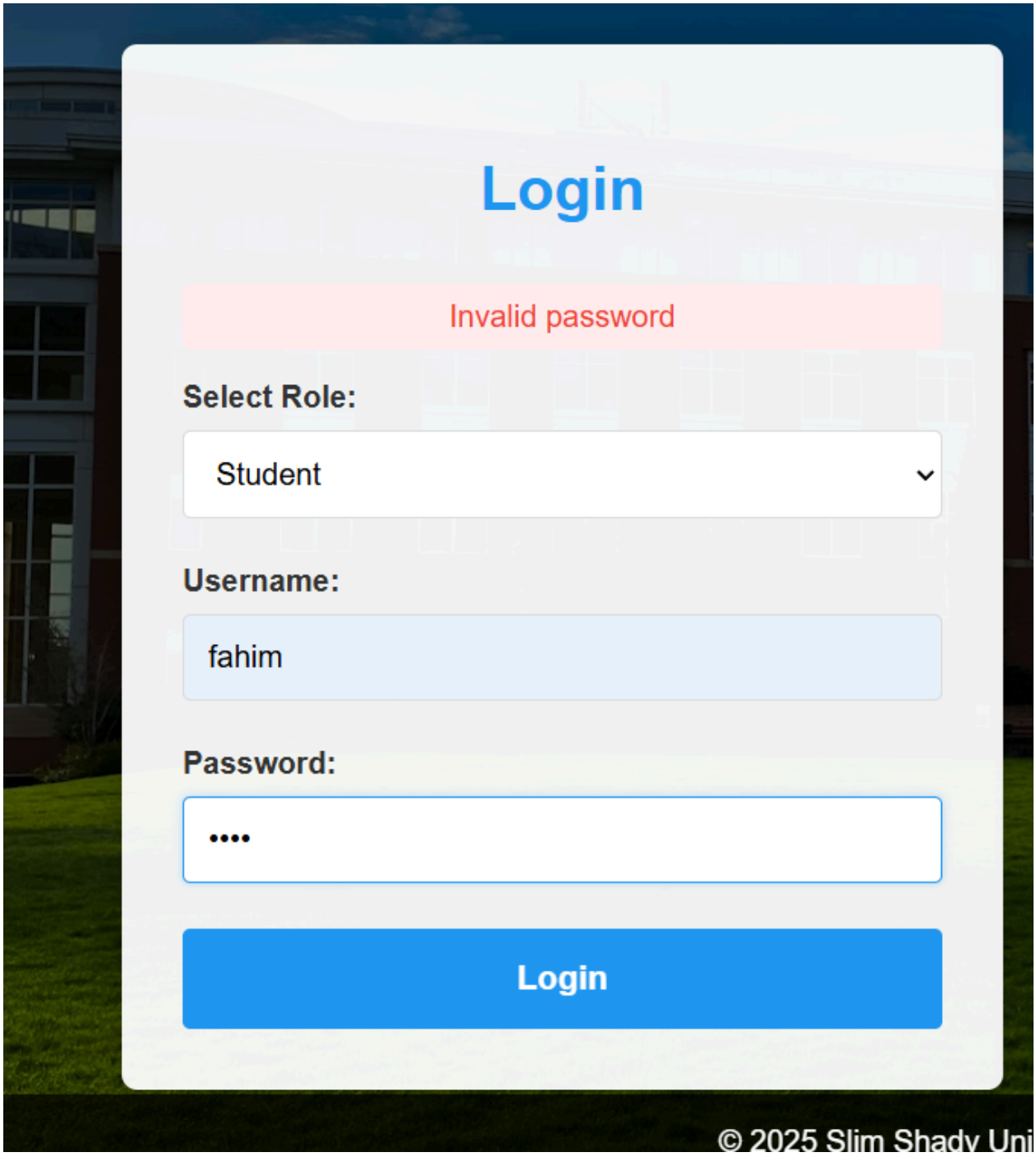


```
rc > classes > User.php > ...
3   abstract class User {
        1 reference
4       protected $id;
        2 references
5       protected $username;
        3 references
6       protected $email;
        1 reference
7       protected $password;
8
        0 references | 0 overrides
9       public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

```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): voi
    $this->validatePassword($newPassword);
    $this->password = password_hash($newPassword,
}

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

    if ($this->verifyPassword($password)) {
        // Reset login attempts on successful log
        $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_A
```

```php
     */
    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 chi
     * @param string $password
     * @return bool
     */
    abstract public function login($password): bool;

    /**
     * Abstract logout method to be implemented by ch
     * @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:

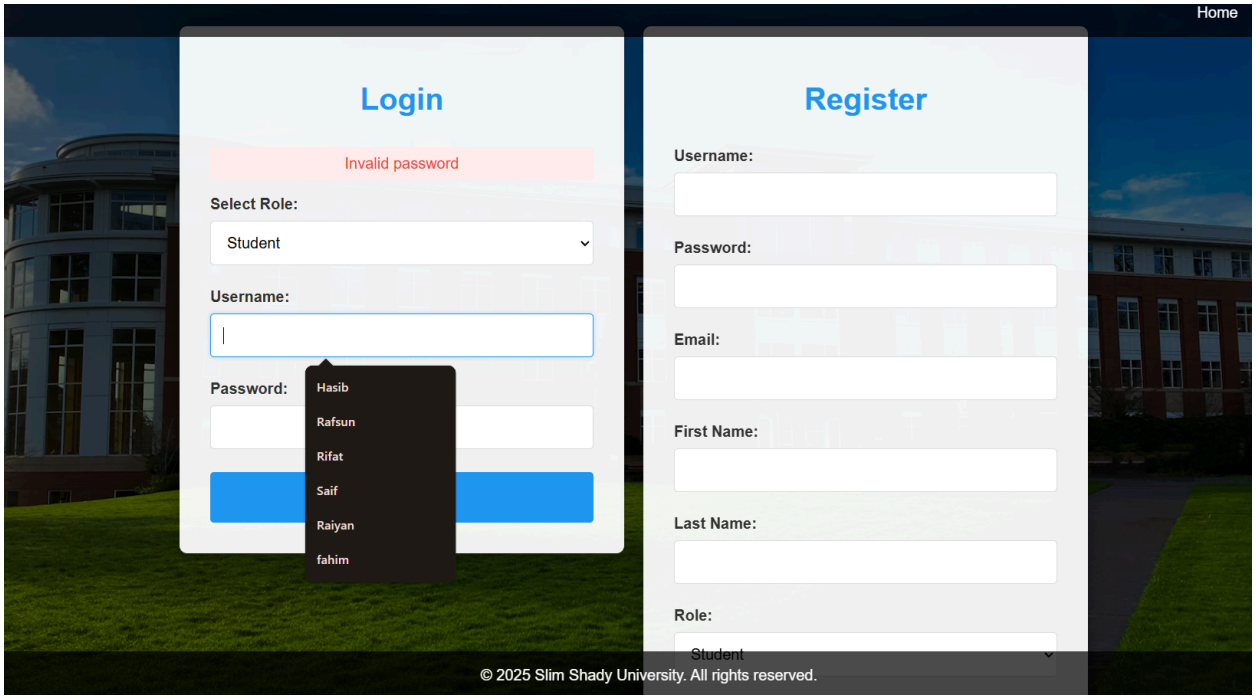


Security:Hash password used

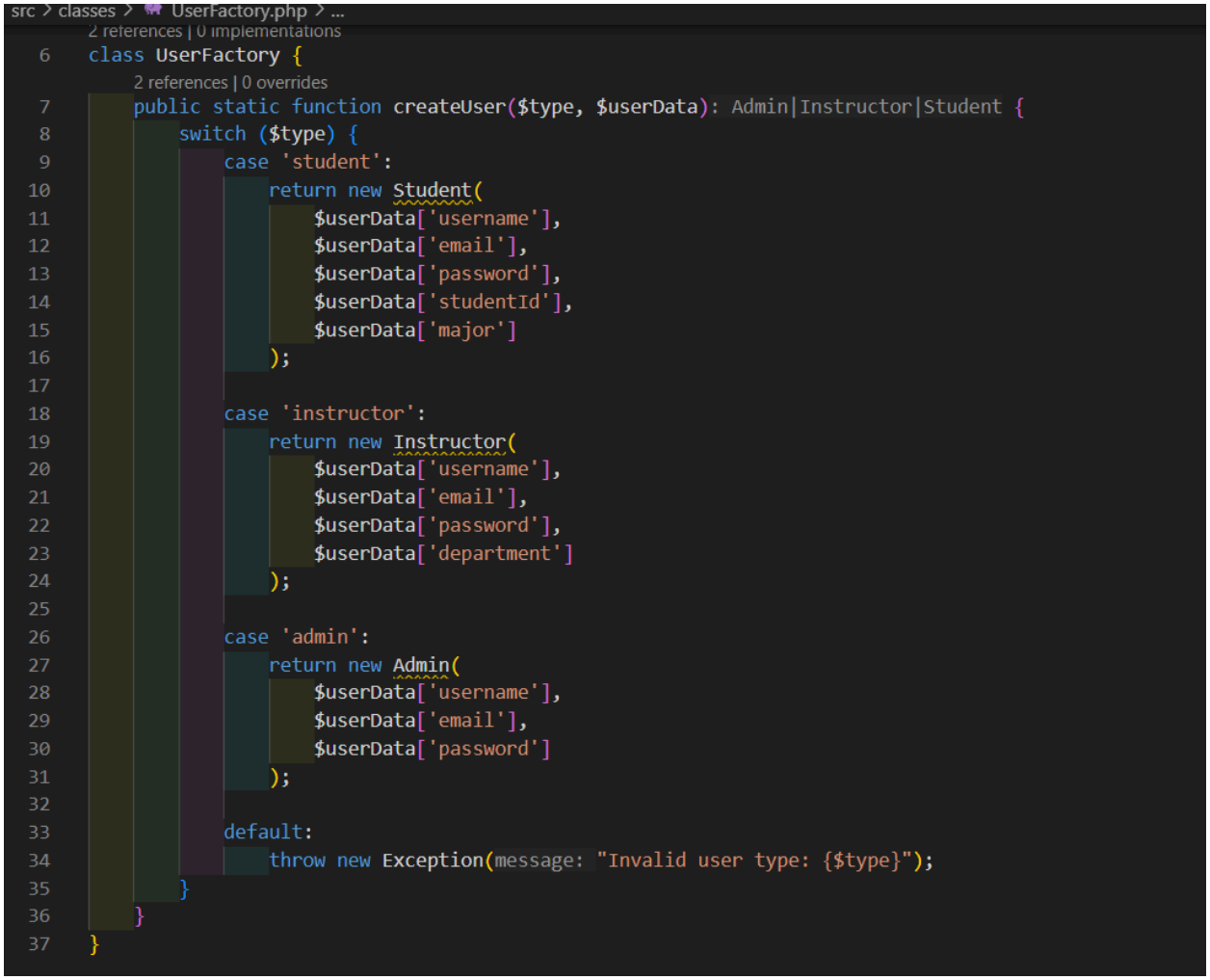
	user_id	username	password
Edit Copy Delete	8	fahim	$2y$10$USwahbaVmwOHN/EaKiS5HuuhiOmX7p/ZsN.QA8WKnUt...
Edit Copy Delete	9	Hasib	$2y$10$vdgjyQWmmJqy40S2mYByE.hihuRX3TaKCmwMzJBwf9Y...
Edit Copy Delete	10	Raiyan	$2y$10$fS/yhInGZxE10Qy8MmE2LOSLNodcbkrb/W1f0fZQHvH...
Edit Copy Delete	11	Rafsun	$2y$10$KO.0zCaTYm/3MDgnYDfhNuYKFsBZ67HbTyVfdSfpx6f...
Edit Copy Delete	12	Rifat	$2y$10$.UVGeo6AKrlLPEH1UGwjLuQWMLK6A.dFSTh8TgdUun6...
Edit Copy Delete	13	Saif	$2y$10$Eht0fUYNH2siyxBWrZ5x4OSSV/eJYZkNMlrZi.8GV6Z...

Check all *With selected:* Edit Copy Delete Export

Large number of Users:



1.CourseFactory design pattern:



2.StudentObserver Design pattern:

```php
<?php

1 reference | 1 implementation
interface CourseSubject {
    0 references | 1 override
    public function registerStudent(Student $student);
    0 references | 1 override
    public function removeStudent(Student $student);
    1 reference | 1 override
    public function notifyStudents($message);
}
```

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

```php
<?php

0 references | 0 implementations
interface StudentObserver {
    0 references | 0 overrides
    public function update($message);
}
```

3.DatabaseSingleton design pattern:

```
3   class DatabaseSingleton {
11      private function __construct() {
23
24              $this->connection->set_charset(charset: "utf8mb4");
25          } catch (Exception $e) {
26              error_log(message: "Database Connection Error: " . $e->getMessage(
27              throw $e;
28          }
29      }
30
            6 references | 0 overrides
31      public static function getInstance(): DatabaseSingleton {
32          if (self::$instance === null) {
33              self::$instance = new self();
34          }
35          return self::$instance;
36      }
37
            6 references | 0 overrides
38      public function getConnection(): mysqli {
39          return $this->connection;
40      }
41
42      // Prevent cloning
            0 references
43      private function __clone(): void {}
44
45      // Prevent unserialization
            0 references
46      private function __wakeup(): void {}
47
            0 references | 0 overrides
48      public function __destruct() {
49          if ($this->connection) {
50              $this->connection->close();
51          }
52      }
53  }
```

4.CourseDecorator:

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

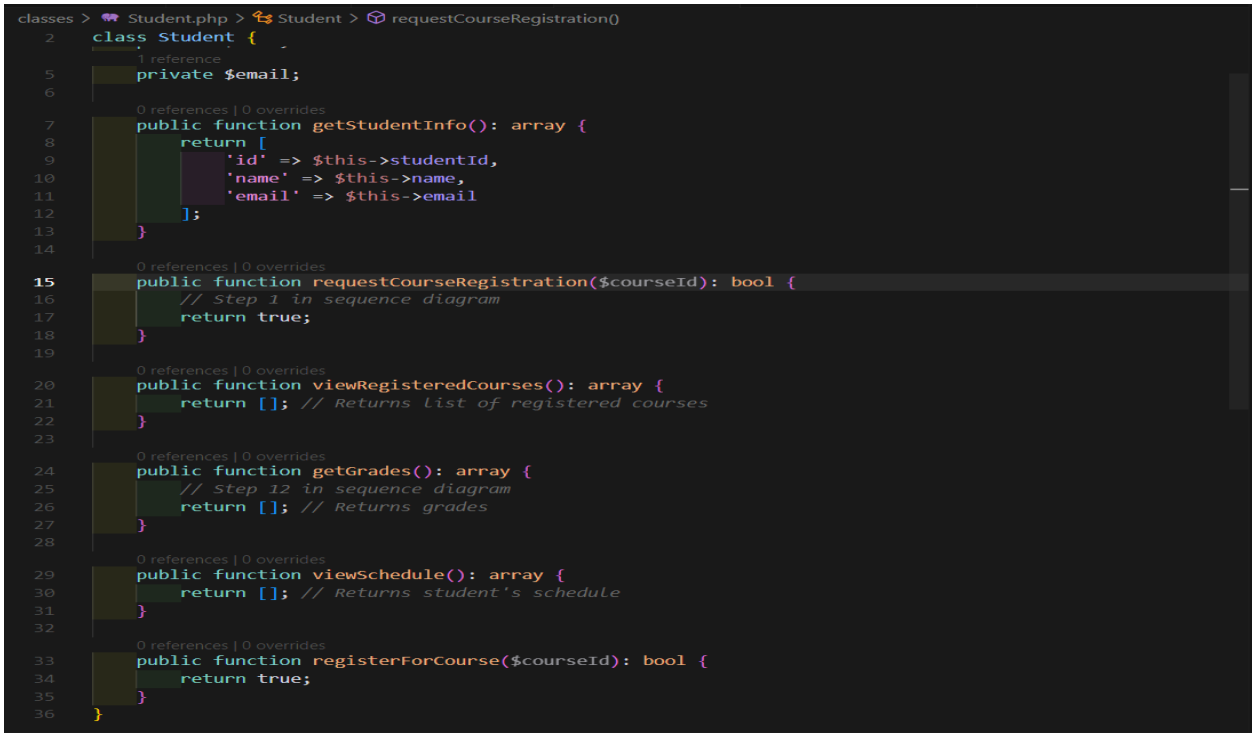
All design patterns in the system are integrated through a single database connection managed by the Singleton pattern. The Factory pattern is responsible for creating base course objects, which the Decorator pattern then enhances with additional features. Any changes made to the courses trigger the

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.

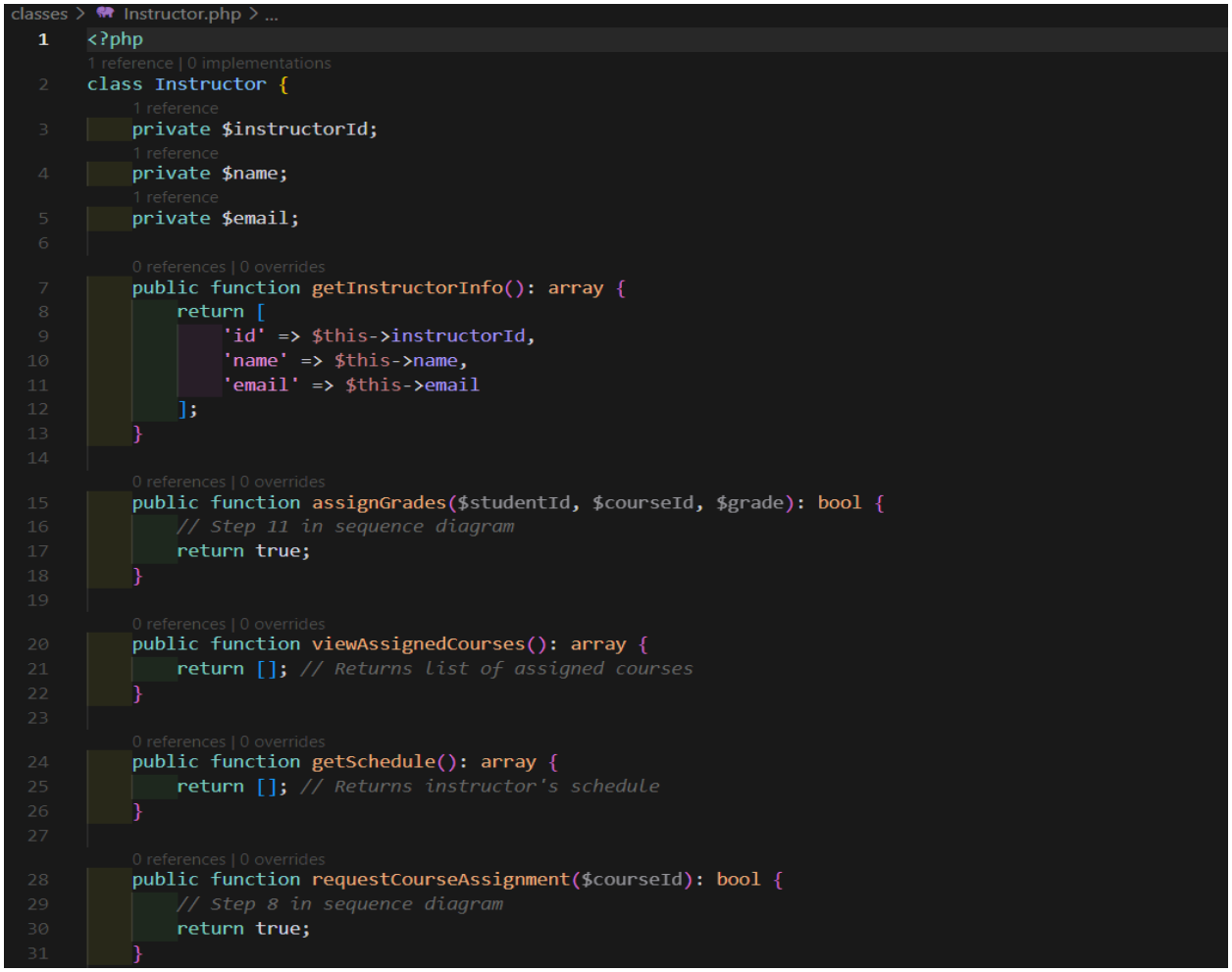
4. Class and Sequence diagram matches with my implementation:

Screenshot of Code:

1.Student.php:

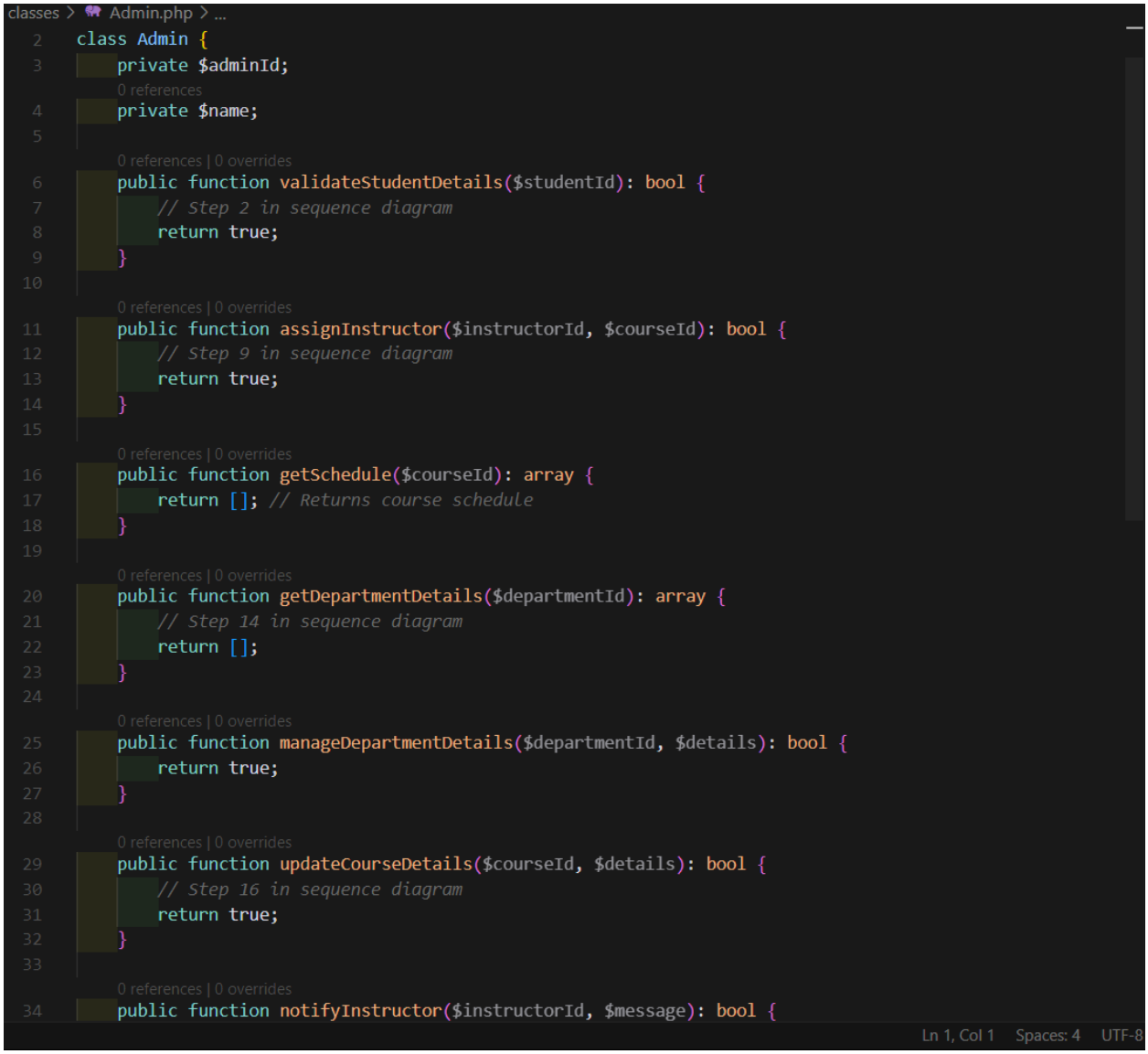


2.Instructor.php:



```
classes > Instructor.php > ...
<?php
1 reference | 0 implementations
class Instructor {
    1 reference
    private $instructorId;
    1 reference
    private $name;
    1 reference
    private $email;

    0 references | 0 overrides
    public function getInstructorInfo(): array {
        return [
            'id' => $this->instructorId,
            'name' => $this->name,
            'email' => $this->email
        ];
    }

    0 references | 0 overrides
    public function assignGrades($studentId, $courseId, $grade): bool {
        // Step 11 in sequence diagram
        return true;
    }

    0 references | 0 overrides
    public function viewAssignedCourses(): array {
        return []; // Returns list of assigned courses
    }

    0 references | 0 overrides
    public function getSchedule(): array {
        return []; // Returns instructor's schedule
    }

    0 references | 0 overrides
    public function requestCourseAssignment($courseId): bool {
        // Step 8 in sequence diagram
        return true;
    }
```

3.Admin.php:

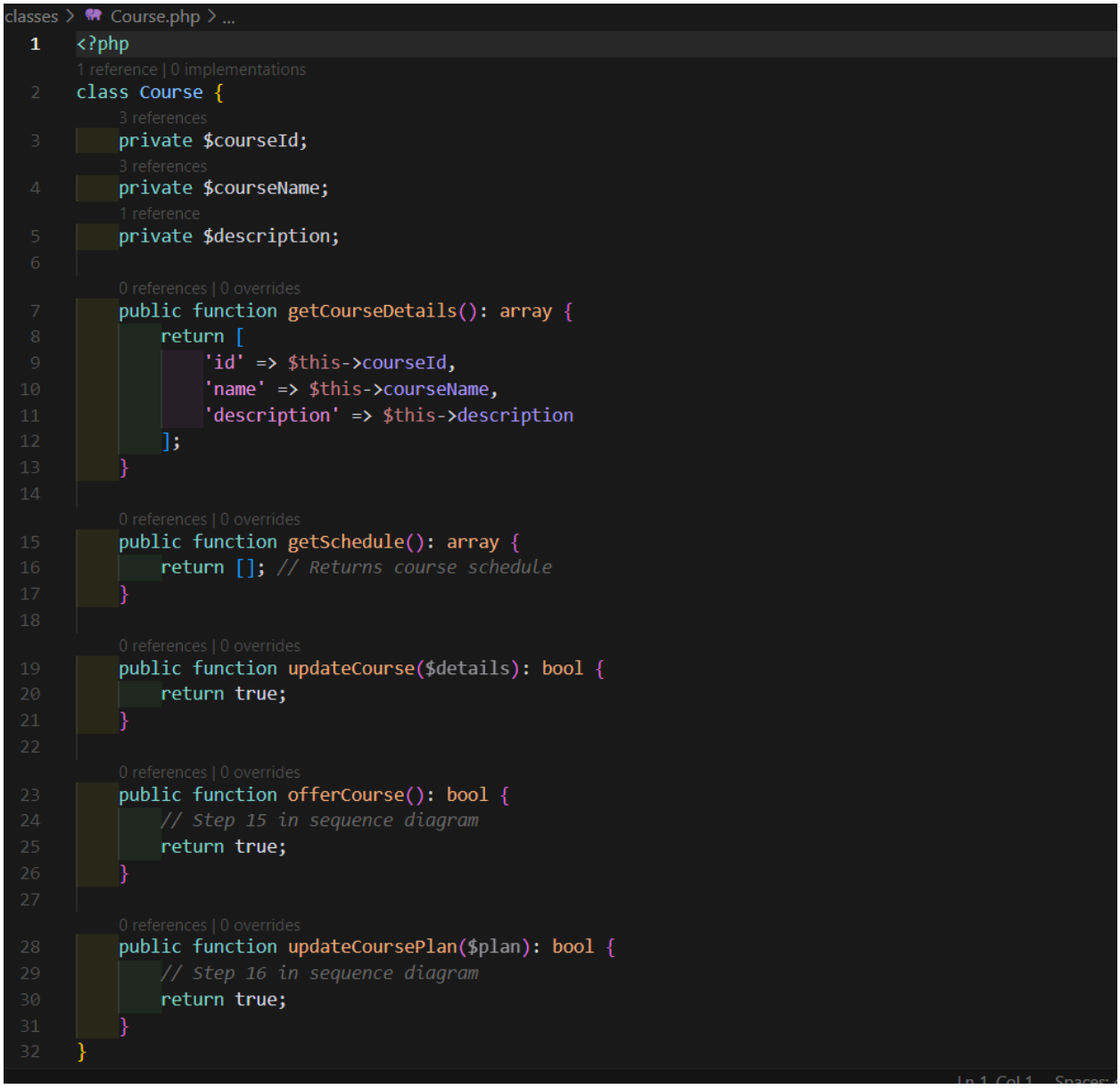

classes > Admin.php > ...
class Admin {
private $adminId;
0 references
private $name;
0 references | 0 overrides
public function validateStudentDetails($studentId): bool {
// Step 2 in sequence diagram
return true;
}
0 references | 0 overrides
public function assignInstructor($instructorId, $courseId): bool {
// Step 9 in sequence diagram
return true;
}
0 references | 0 overrides
public function getSchedule($courseId): array {
return []; // Returns course schedule
}
0 references | 0 overrides
public function getDepartmentDetails($departmentId): array {
// Step 14 in sequence diagram
return [];
}
0 references | 0 overrides
public function manageDepartmentDetails($departmentId, $details): bool {
return true;
}
0 references | 0 overrides
public function updateCourseDetails($courseId, $details): bool {
// Step 16 in sequence diagram
return true;
}
0 references | 0 overrides
public function notifyInstructor($instructorId, $message): bool {
Ln 1, Col 1 Spaces: 4 UTF-8

4.Registration.php:

classes > Registration.php > ...

```php
<?php
0 references | 0 implementations
class Registration {
    0 references
    private $registrationId;
    0 references
    private $status;

    0 references | 0 overrides
    public function checkCourseAvailability($courseId): bool {
        // Step 3 in sequence diagram
        return true;
    }

    0 references | 0 overrides
    public function verifyScheduleAvailability($courseId, $studentId): bool {
        // Step 4 in sequence diagram
        return true;
    }

    0 references | 0 overrides
    public function updateRegistrationStatus($studentId, $courseId, $status): bool {
        // Step 6 in sequence diagram
        return true;
    }

    0 references | 0 overrides
    public function confirmRegistration($studentId, $courseId): bool {
        // Step 7 in sequence diagram
        return true;
    }

    0 references | 0 overrides
    public function getRegistrationDetails($registrationId): array {
        return [];
    }
}
```

5.Course.php:


classes > Course.php > ...
<?php
1 reference | 0 implementations
class Course {
3 references
private $courseId;
3 references
private $courseName;
1 reference
private $description;
0 references | 0 overrides
public function getCourseDetails(): array {
return [
'id' => $this->courseId,
'name' => $this->courseName,
'description' => $this->description
];
}
0 references | 0 overrides
public function getSchedule(): array {
return []; // Returns course schedule
}
0 references | 0 overrides
public function updateCourse($details): bool {
return true;
}
0 references | 0 overrides
public function offerCourse(): bool {
// Step 15 in sequence diagram
return true;
}
0 references | 0 overrides
public function updateCoursePlan($plan): bool {
// Step 16 in sequence diagram
return true;
}
}

6.Department.php:

classes > Department.php > ...

```php
<?php
0 references | 0 implementations
class Department {
    1 reference
    private $departmentId;
    1 reference
    private $name;

    0 references | 0 overrides
    public function getDepartmentDetails(): array {
        return [
            'id' => $this->departmentId,
            'name' => $this->name
        ];
    }

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

    0 references | 0 overrides
    public function offerNewCourses($courses): bool {
        // Step 15 in sequence diagram
        return true;
    }

    0 references | 0 overrides
    public function updateCoursePlan($courseId, $plan): bool {
        // Step 16 in sequence diagram
        return true;
    }
}
```

7.Schedule.php:

classes > Schedule.php > ...

```php
<?php
0 references | 0 implementations
class Schedule {
    1 reference
    private $scheduleId;
    1 reference
    private $day;
    1 reference
    private $time;

    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