



Inspiring Excellence

**CSE370 : Database Systems**  
**Project Report**  
**Project Title : Academic Gateway**

Group No : _08_, CSE370 Lab Section : _06_, Fall 2024		
ID	Name	Contribution
22301689	Tasnim Rahman Moumita	<ol style="list-style-type: none"><li>1. Database Connection Setup,</li><li>2. User login.</li><li>3. User logout</li><li>4. Grade Sheet management.</li></ol>
22301504	Abdul Kaium Choudhury	<ol style="list-style-type: none"><li>1. Database Connection</li><li>2. User Entity Connecting</li><li>3. Teacher, Student, Admin Dashboard</li><li>4. Registration</li><li>5. Teacher Dashboard</li><li>6. Admin Dashboard</li><li>7. Student Dashboard</li></ol>
22301163	Tamanna Islam Tazin	<ol style="list-style-type: none"><li>1. Dashboard</li><li>2. Courses</li><li>3. Exam</li><li>4. Database creation</li><li>5. Phpmyadmin tables creation</li></ol>

## Table of Contents

Section No	Content	Page No
1	Introduction	03
2	Project Features	03-05
3	ER/EER Diagram	06
4	Schema Diagram	07
5	Frontend Development	08-20
6	Backend Development	21-32
7	Source Code Repository	33
8	Conclusion	33
9	References	34

## **Introduction**

An educational institution's academic operations can be streamlined and centralized with the help of the **Academic Gateway** database management system. Facilitating effective management of student, teacher, and administrative data while guaranteeing smooth collaboration across diverse academic activities is its main objective.

For managing teaching assignments, course enrollment, and grades processing, this system is designed to offer reliable solutions.

## **Project Features**

### **1. User Management System**

- User registration and login functionality for Admin, Teacher, and Student roles.
- Profile management for all users, including name, email, password, and other attributes.

### **2. Role-Based Access Control**

- Admins can manage permissions for teachers and students.
- Teachers can manage courses and exams.
- Students can enroll in courses and access grades.

### **3. Registration to Login**

- Users can directly access to their role-based dashboard without re-login immediately after they register and have access to the features.

#### **4.Student Management**

- Add, edit, and view student details, including enrollment year, department, and courses they are enrolled in.
- View grades and academic performance through a grade sheet.

#### **5.Teacher Management**

- Add, edit, and view teacher details, including specialization and designation.
- Assign teachers to specific courses and exams.

#### **6.Course Management**

- Admins and teachers can create and manage courses, including course name, start date, end date, and description.
- Students can enroll in courses.

#### **7.Exam Management**

- Admins or teachers can schedule exams for specific courses, including exam name, date, and creator details.
- Link exams with courses for a structured flow.

#### **8.Grade Sheet Management**

- Manage grades for each student based on their performance in exams.
- Generate and display grade sheets showing grades for enrolled courses and exams.

#### **9.Permission Management**

- Admins can assign permissions for specific actions like course creation, exam management, and grade sheet access.

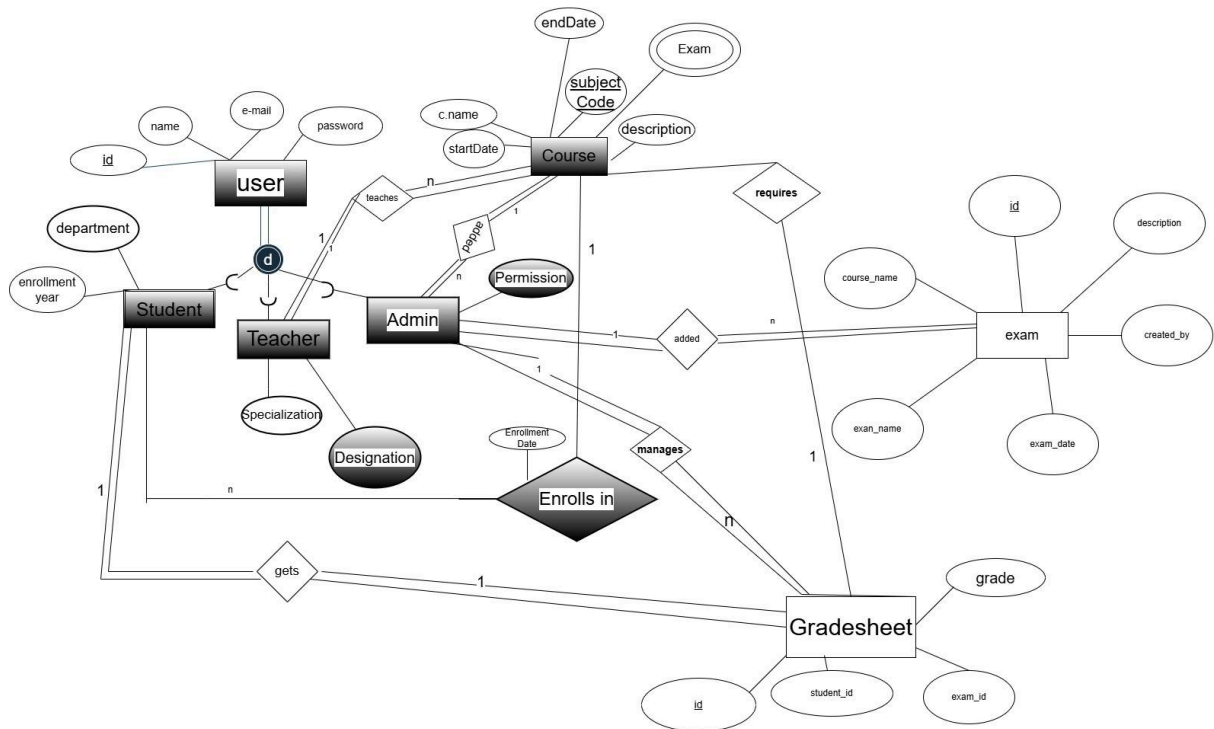
- Ensure secure access for role-specific functionalities.

## **10.Data Management**

- Comprehensive data storage for all entities, including users, courses, exams, and grades.
- Integration of relational data between students, teachers, courses, and exams.

## ER/EER Diagram

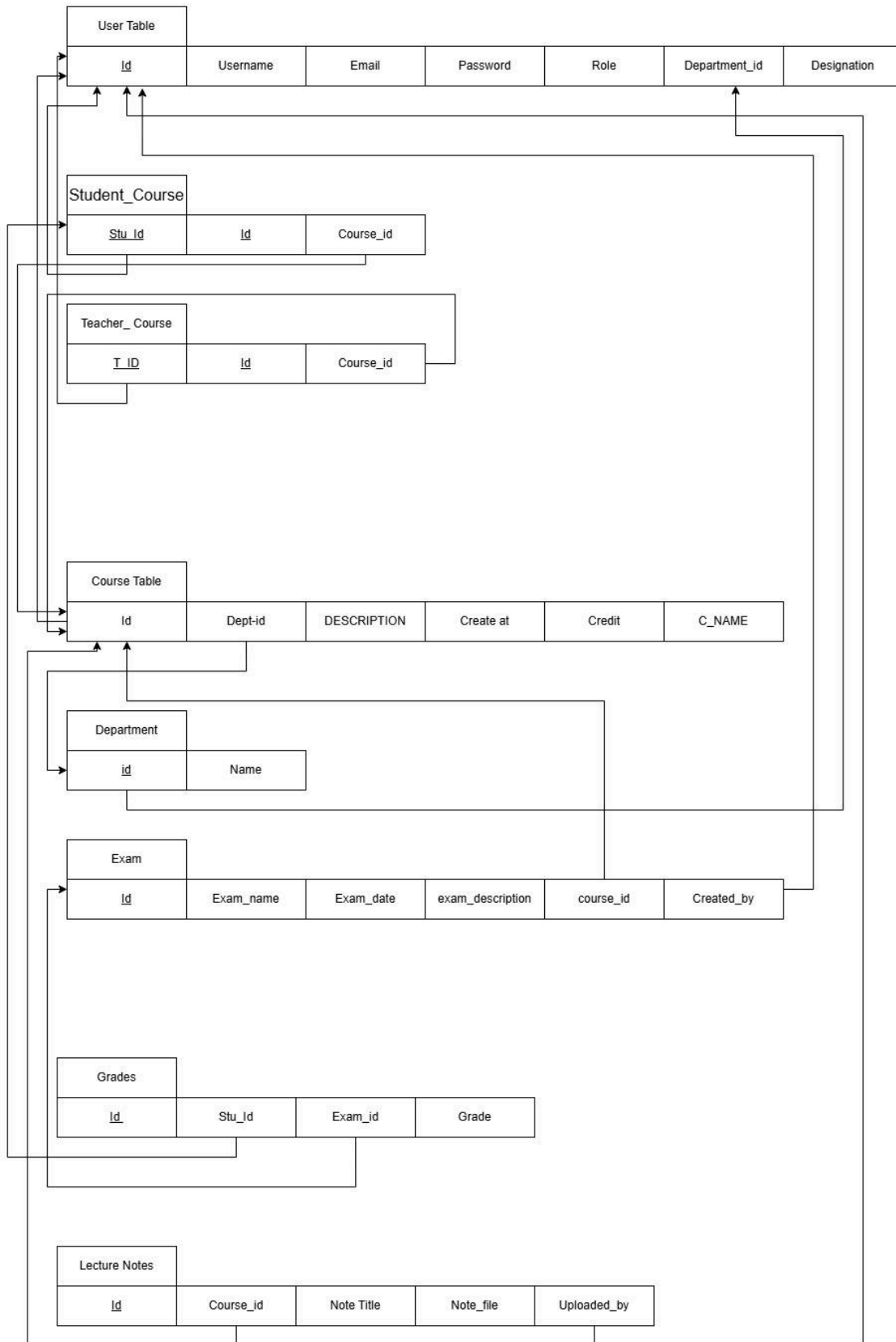
### Academic Gateway



**Figure : EER Diagram of "ACADEMIC GATEWAY" Database**

## Schema Diagram

### Academic Gateway Schema Representation



## Frontend Development

Briefly discuss about Frontend Development and add Screenshots by mentioning Individual Contributions

Contribution of ID : 22301689, Name : Tasnim Rahman Moumita

### Login :

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
  <title>Login</title>
</head>
<body>
  <h2 class="container">Login</h2>
  <form method="POST">
    <input type="text" name="username" placeholder="Username" required>
    <input type="password" name="password" placeholder="Password" required>
    <button type="submit">Login</button>
  </form>
</body>
</html>
```

This frontend focuses on the user interface (UI), enabling the user to interact with the login system.

### Key frontend features:

- **Structure and Styling:** The HTML markup structures the page, including a header (<h2>) for the title and a form for user input. The external stylesheet style.css is linked to enhance the visual presentation of the page.
- **User Input:** The form includes two input fields (<input>), one for the username and another for the password, both of which are required. The placeholder attributes provide hints for user input.
- **Interactivity:** A submit button (<button>) allows users to send the form data for processing.
- **Accessibility and Usability:** Simple, clean design ensures ease of use and accessibility for users.



This part ensures a responsive and user-friendly interface that communicates with the backend to complete the login process.

### **Log out:**

```
<style>
.logout-btn {
  background-color: red;
  color: white;
  border: none;
  padding: 10px 20px;
  cursor: pointer;
  border-radius: 5px;
}
.logout-btn:hover {
  background-color: darkred;
}
</style>
</head>
<body>
<div class="container">
  <h2>Student Dashboard</h2>
  <form method="POST" action="logout.php" style="display: inline-block;">
    <button class="logout-btn" type="submit">Logout</button>
  </form>
</div>
</body>
</html>
```

This section includes a Student Dashboard with a Logout button, styled with CSS:

- CSS Styling:
  - .logout-btn: Styles the button with a red background, white text, and rounded corners. It also includes a hover effect that changes the background to dark red when the button is hovered.
  - The cursor changes to a hand when hovering, indicating the button is clickable.
- HTML Structure:
  - The page contains a heading (<h2>Student Dashboard</h2>) and a form that sends a POST request to logout.php when the Logout button is clicked.
  - The Logout Button (<button class="logout-btn">Logout</button>) submits the form to log the user out.

When clicked, the button triggers a logout action by submitting the form to logout.php, ending the user session.

## Grade sheet:

```
<!-- Grade Upload Section -->
<h3>Upload Grades</h3>
<?php if (isset($grade_success_message)) echo "<p>$grade_success_message</p>"; ?>
<form method="POST">
    <select name="student_id" required>
        <option value="">Select Student</option>
        <?php
            // Fetch students from the database to populate the dropdown
            $students_stmt = $pdo->query("SELECT id, username FROM users WHERE role = 'student'");
            while ($student = $students_stmt->fetch()) {
                echo "<option value='" . $student['id'] . "'>" . htmlspecialchars($student['username']) . "</option>";
            }
        ?>
    </select><br>

    <select name="exam_id" required>
        <option value="">Select Exam</option>
        <?php
            // Fetch exams from the database to populate the dropdown
            $exams_stmt = $pdo->query("SELECT id, exam_name FROM exams");
            while ($exam = $exams_stmt->fetch()) {
                echo "<option value='" . $exam['id'] . "'>" . htmlspecialchars($exam['exam_name']) . "</option>";
            }
        ?>
    </select><br>

    <input type="number" name="grade" placeholder="Grade (e.g., 85.5)" step="0.01" required><br>
    <button type="submit">Upload Grade</button>
</form>

</body>
</html>
```

## Functionality:

### 1. Form Fields:

- **Student Dropdown:**
  - Dynamically populates with students retrieved from the users table (role = 'student').
- **Exam Dropdown:**
  - Dynamically populates with exams retrieved from the exams table.
- **Grade Input:**
  - Accepts the grade value in decimal format using the number input type with a step of 0.01.

- 2. Submit Button:** Triggers the POST request to submit the grade data to the backend for processing.

Contribution of ID : 22301504, Name : Abdul Kaium Choudhury

## Student Dashboard:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
  <title>Student Dashboard</title>
</head>
<body>
  <div class="container">
    <h2>Student Dashboard</h2>

    <h3>Enroll in a Course</h3>
    <form method="POST">
      <select name="course_id" required>
        <option value="">Select a Course</option>
        <?php foreach ($courses as $course): ?>
          <option value="= $course['id'] ?">= htmlspecialchars($course['course_name']) ?&gt;&lt;/option&gt;
        &lt;?php endforeach; ?&gt;
      &lt;/select&gt;
      &lt;button type="submit"&gt;Enroll&lt;/button&gt;
    &lt;/form&gt;

    &lt;h3&gt;Your Courses&lt;/h3&gt;
    &lt;ul&gt;
      &lt;?php foreach ($enrolled_courses as $course): ?&gt;
        &lt;li&gt;<?= htmlspecialchars($course['course_name']) ?&gt;&lt;/li&gt;
      &lt;?php endforeach; ?&gt;
    &lt;/ul&gt;

    &lt;h3&gt;Lecture Notes&lt;/h3&gt;
    &lt;ul&gt;
      &lt;?php
        $notes_stmt = $pdo-&gt;prepare("SELECT * FROM lecture_notes WHERE course_id IN (SELECT course_id FROM student_courses WHERE student_id = ?)");
        $notes_stmt-&gt;execute([$student_id]);
        $notes = $notes_stmt-&gt;fetchAll();

        foreach ($notes as $note): ?&gt;
          &lt;li&gt;
            &lt;strong&gt;<?= htmlspecialchars($note['note_title']) ?&gt;&lt;/strong&gt;
            &lt;a href="<?= htmlspecialchars($note['note_file']) ?" target="_blank">Download</a>
          </li>
        <?php endforeach; ?>
    </ul>
  </div>
</body>
</html>
```

Frontend part of Student Dashboard,

### 1. Page Structure:

- The HTML structure is simple, containing a header with the title "Student Dashboard" and different sections for course enrollment, displaying enrolled courses, and lecture notes.
- The page is styled using an external stylesheet (style.css).

### 2. Course Enrollment Form:

- The form allows the student to select a course from a dropdown list. The options for courses are dynamically populated by iterating

over the \$courses array (from the backend), where each course and course\_name are displayed.

- When a course is selected, and the form is submitted, the data is sent to the server via a POST request for processing in the backend.

### 3. Displaying Enrolled Courses:

- The section lists all courses the student is enrolled in by looping through the \$enrolled\_courses array. Each course name is displayed as a list item in an unordered list (<ul>).

### 4. Lecture Notes:

- Another section displays the student's lecture notes. The notes are fetched from the database based on the courses the student is enrolled in and displayed in a list.
- Each lecture note has a title and a link to download the file. The href attribute points to the file stored in the note\_file column, opening the file in a new tab (target="\_blank").

### Admin Dashboard:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      font-family: Arial, sans-serif;
      margin: 20px;
    }
    h2, h3 {
      color: #2c3e50;
    }
    form {
      margin-bottom: 20px;
      padding: 10px;
      border: 1px solid #ddd;
      border-radius: 5px;
    }
    input, textarea, select, button {
      display: block;
      margin-bottom: 10px;
      padding: 8px;
      width: 100%;
      border: 1px solid #ddd;
      border-radius: 5px;
    }
    button {
      background-color: #2c3e50;
      color: white;
      cursor: pointer;
    }
    button:hover {
      background-color: #34495e;
    }
    p {
      color: green;
    }
  </style>
  <title>Admin Dashboard</title>
</head>
<body>
  <h2>Admin Dashboard</h2>
```

Here, for admin frontend part,

### **1. User-Friendly Interface:**

- The admin dashboard is designed with a simple and clean interface using basic HTML and CSS.
- The layout is intuitive, making it easy for the admin to navigate and perform tasks.

### **2. Responsive Form Design**

- All forms are styled with consistent spacing, borders, and padding to improve usability.
- Input fields, text areas, and buttons have been styled to ensure a uniform appearance.

### **3. Interactive Button Effects**

- Buttons feature a hover effect that changes their background color, providing visual feedback to the user.

### **4. Organized Sections**

- The dashboard is divided into sections for uploading:
  - Courses
  - Exams
  - Grades
- Each section has a distinct header and a form, ensuring clarity and separation of tasks.

### **5. Success Messages**

- Success messages are displayed in green after a successful operation, providing immediate feedback to the admin.

### **6. Dynamic Dropdown Options**

- Forms dynamically fetch options for courses, exams, and students from the database, ensuring up-to-date data is always available for selection.

## Teacher Dashboard:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
  <title>Teacher Dashboard</title>
</head>
<body>
  <div class="container">
    <h2>Teacher Dashboard</h2>

    <h3>Assign Courses to Teach</h3>
    <form method="POST">
      <select name="course_id" required>
        <option value="">Select a Course</option>
        <?php foreach ($courses as $course): ?>
          <option value="<?=$course['id'] ?>"><?=$course['course_name'] ?></option>
        <?php endforeach; ?>
      </select>
      <button type="submit">Assign Course</button>
    </form>

    <h3>Your Courses</h3>
    <ul>
      <?php foreach ($assigned_courses as $course): ?>
        <li><?=$course['course_name'] ?></li>
      <?php endforeach; ?>
    </ul>

    <h3>Upload Lecture Notes</h3>
    <form method="POST" action="upload_notes.php" enctype="multipart/form-data">
      <select name="course_id" required>
        <option value="">Select a Course</option>
        <?php foreach ($assigned_courses as $course): ?>
          <option value="<?=$course['id'] ?>"><?=$course['course_name'] ?></option>
        <?php endforeach; ?>
      </select>
      <input type="text" name="note_title" placeholder="Note Title" required>
      <input type="file" name="note_file" accept=".pdf,.doc,.docx,.ppt,.pptx" required>
      <button type="submit">Upload</button>
    </form>
  </div>
</body>
</html>
```

For this part of frontend,

### 1. Assign Courses to Teach:

- **Functionality:** This section allows the teacher to assign themselves to a course they will be teaching. Teachers can choose from a list of available courses.
- **Form:** A dropdown menu (<select>) is provided to choose a course. Each course is dynamically populated from the available courses (\$courses) fetched from the backend. After selecting a course, the teacher clicks the "Assign Course" button to submit the form.
- **Backend Interaction:** Upon form submission, the course ID is sent to the server to associate the teacher with the selected course in the database (the backend handles this part via a POST request).

## **2. Courses:**

- **Functionality:** This section displays a list of courses the teacher is currently assigned to teach.
- **Display:** A simple unordered list (<ul>) is used to display the course names. The course names are fetched from the backend using the \$assigned\_courses array, which contains the list of courses the teacher is currently teaching.

## **3. Upload Lecture Notes:**

- **Functionality:** Teachers can upload lecture notes for the courses they are assigned to. This allows students to access relevant resources for their enrolled courses.
- **Form:** The form includes the following fields:
  - A dropdown menu to select the course to which the lecture notes belong.
  - An input field for the teacher to enter the title of the note.
  - A file input field to upload lecture notes, accepting file types like .pdf, .doc, .docx, .ppt, .pptx.
  - A submit button to upload the file.

## Registration:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
  <title>Register</title>
</head>
<body>
  <h2>Register</h2>
  <form method="POST">
    <input type="email" name="email" placeholder="Email" required>
    <input type="text" name="username" placeholder="Username" required>
    <input type="password" name="password" placeholder="Password" required>
    <select name="role" required>
      <option value="" disabled selected>Select your role</option>
      <option value="admin">Admin</option>
      <option value="teacher">Teacher</option>
      <option value="student">Student</option>
    </select>
    <button type="submit">Register</button>
  </form>
</body>
</html>
```

The frontend of this registration system is designed using HTML to allow users to register by providing their email, username, password, and role. It consists of the following key components:

### 1. Form Structure:

- The form uses the POST method to send data to the server when submitted.
- **It has four fields:**
  - Email: An input field where the user is required to enter a valid email address.
  - Username: A text input field where the user is required to choose a username.
  - Password: A password input field where the user can securely enter their password.
  - Role: A dropdown select field allowing the user to choose one of three roles: admin, teacher, or student.
- The submit button sends the form data to the server for processing.



## 2. Input Validation:

- Each of the fields (email, username, password, and role) is marked as required, meaning the form will not be submitted unless these fields are filled in.
- The email field is validated on the server side to ensure it contains a valid email format.

## 3. Styling:

- A link to the external style.css file is included to style the form, but details of this styling are not included in the provided code.

## 4. Action on Form Submission:

- When the form is submitted, it triggers a POST request to the same page, sending the form data to the server for validation and insertion into the database.

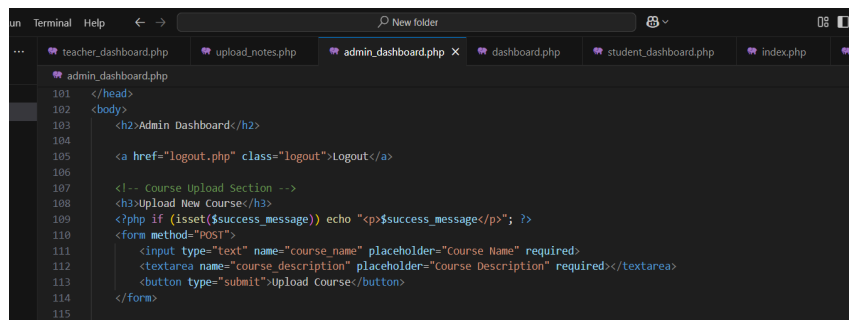
**Contribution of ID : 22301163, Name : Tamanna Islam Tazin**

## ● Admin Dashboard:

### i) Course Management

Front-End (What the admin sees and interacts with):

- A form with:
  - A text box for the course name.
  - A larger box for the course description.
  - A submit button to save the course.
- Success message displayed if the course is successfully added.



```
101 </head>
102 <body>
103 <h2>Admin Dashboard</h2>
104
105 <a href="logout.php" class="logout">Logout</a>
106
107 <!-- Course Upload Section -->
108 <h3>Upload New Course</h3>
109 <?php if (isset($success_message)) echo "<p>$success_message</p>"; ?>
110 <form method="POST">
111 <input type="text" name="course_name" placeholder="Course Name" required>
112 <textarea name="course_description" placeholder="Course Description" required></textarea>
113 <button type="submit">Upload Course</button>
114 </form>
115
```

## ii). Exam Management

Front-End (What the admin sees and interacts with):

- A form with:
  - A text box for the exam name.
  - A date picker to choose the exam date.
  - A larger box to describe the exam.
  - A dropdown list of courses (e.g., "Math 101") to link the exam to the right course.
  - A submit button to save the exam.
- Success message displayed if the exam is successfully added.

```
116 <!-- Exam Upload Section -->
117 <h3>Upload New Exam</h3>
118 <?php if (isset($exam_success_message)) echo "<p>$exam_success_message</p>"; ?>
119 <form method="POST">
120     <input type="text" name="exam_name" placeholder="Exam Name" required>
121     <input type="date" name="exam_date" required>
122     <textarea name="exam_description" placeholder="Exam Description" required></textarea>
123     <select name="course_id" required>
124         <option value="">Select Course</option>
125         <?php
126             // Fetch courses from the database
127             $courses_stmt = $pdo->query("SELECT id, course_name FROM courses");
128             while ($course = $courses_stmt->fetch()) {
129                 echo "<option value='" . $course['id'] . "'>" . htmlspecialchars($course['course_name']) . "</option>";
130             }
131         </select>
132     <button type="submit">Upload Exam</button>
133 </form>
```

## Student Dashboard:

### 1. Course Enrollment

Front-End (What the student sees and interacts with):

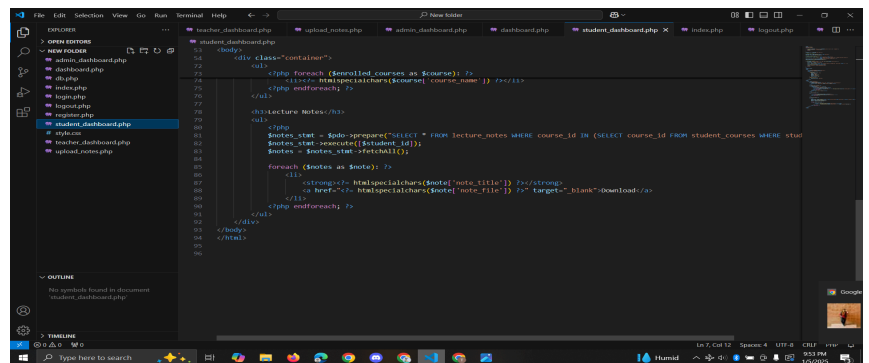
- Dropdown menu:
  - Shows all available courses (e.g., "Math 101," "Physics 101").
  - Students can select one to enroll in.
- Enroll button:
  - Allows students to submit their choice.
- List of enrolled courses:
  - Displays the courses the student is already enrolled in (e.g., "Math 101").
- Success feedback:
  - Once a course is successfully enrolled, the page refreshes, showing the updated list of enrolled courses.

```
<h3>Enroll in a course</h3>
<form method="POST">
    <select name="course_id" required>
        <option value="">Select a Course</option>
        <?php foreach ($courses as $course): ?>
            <option value="<?=$course['id'] ?>"><?=$course['course_name']
        </select>
    <button type="submit">Enroll</button>
</form>
```

## 2. Lecture Notes Access

Front-End (What the student sees and interacts with):

- List of lecture notes:
  - Displays all available notes for the courses the student is enrolled in.
  - For each note, students can:
    - See the title (ex: "Week 1 Notes").
    - Download the file (exp, "Week1Notes.pdf").
- User-friendly layout:
  - Notes are organized under the relevant course for easy navigation.



## Teacher Dashboard:

### 1. Course Assignment

Front-End (What the teacher sees and interacts with):

- Dropdown menu:
  - Lists all available courses (e.g., "Math 101," "Physics 101").
  - Teachers can select a course to assign to themselves.
- Assign button:
  - Submits the selected course for assignment.
- List of assigned courses:
  - Displays all the courses the teacher is already assigned to (e.g., "Math 101," "Physics 101").

```

<h3>Assign Courses to Teach</h3>
<form method="POST">
  <select name="course_id" required>
    <option value="">Select a Course</option>
    <?php foreach ($courses as $course): ?>
      <option value="{?= $course['id'] ?}">{?= htmlspecialchars($course['course_name']
    <?php endforeach; ?>
  </select>
  <button type="submit">Assign Course</button>
</form>

```

## 2. Lecture Note Upload

Front-End (What the teacher sees and interacts with):

- Form for uploading lecture notes:
  - Dropdown menu to select one of their assigned courses (e.g., "Math 101").
  - Input field to enter a title for the notes (e.g., "Week 1 Lecture Notes").
  - File upload field to attach lecture notes (e.g., PDFs, Word docs, or PowerPoint slides).
  - Submit button to upload the file.
- Success feedback:
  - Displays a success message once the notes are uploaded.

```

77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
<h3>Upload Lecture Notes</h3>
<form method="POST" action="upload_notes.php" enctype="multipart/form-data">
  <select name="course_id" required>
    <option value="">Select a Course</option>
    <?php foreach ($assigned_courses as $course): ?>
      <option value="{?= $course['id'] ?}">{?= htmlspecialchars($course['course_name']) ?></option>
    <?php endforeach; ?>
  </select>
  <input type="text" name="note_title" placeholder="Note Title" required>
  <input type="file" name="note_file" accept=".pdf,.doc,.docx,.ppt,.pptx" required>
  <button type="submit">Upload</button>
</form>
</div>
</body>
</html>

```

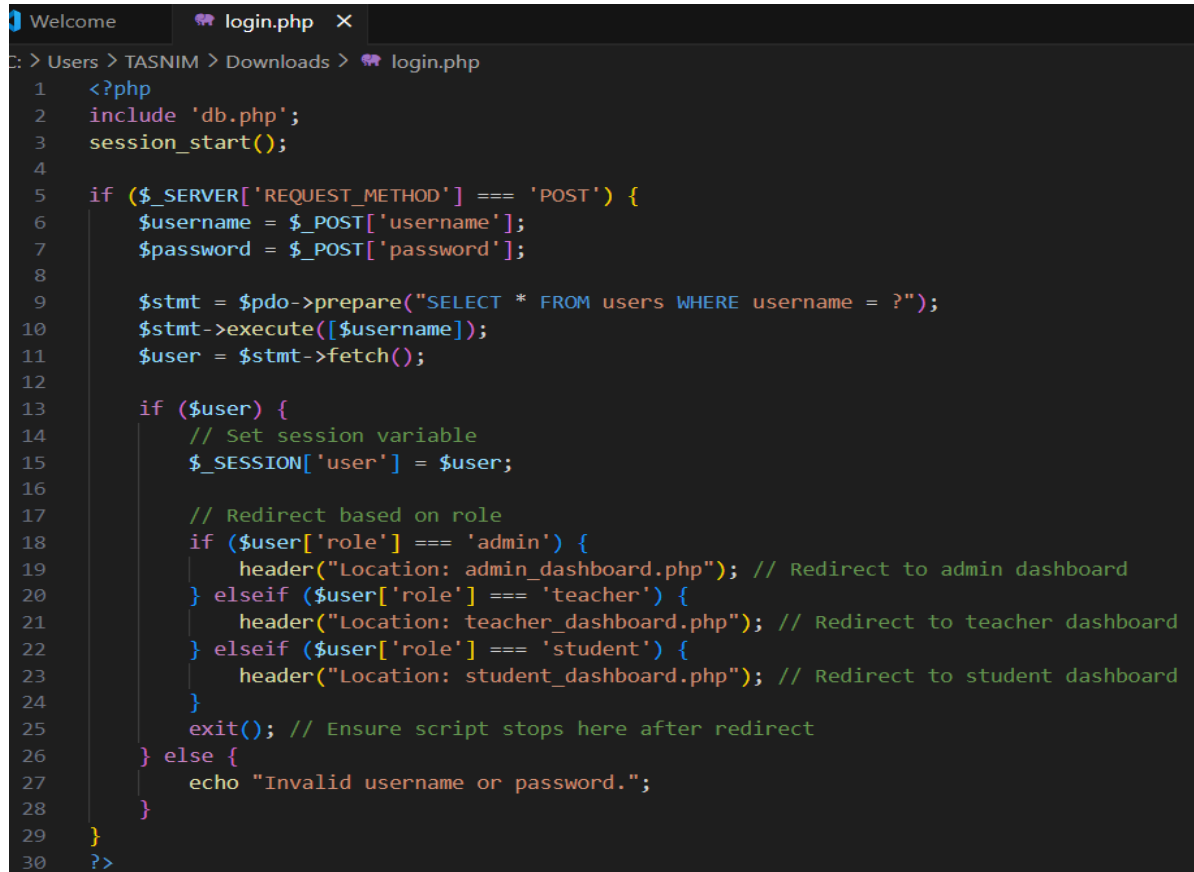
○

## Backend Development

Briefly discuss about Backend Development and add Screenshots by mentioning Individual Contributions

Contribution of ID : 22301689, Name : Tasnim Rahman Moumita

### Login:



```
1 <?php
2 include 'db.php';
3 session_start();
4
5 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
6     $username = $_POST['username'];
7     $password = $_POST['password'];
8
9     $stmt = $pdo->prepare("SELECT * FROM users WHERE username = ?");
10    $stmt->execute([$username]);
11    $user = $stmt->fetch();
12
13    if ($user) {
14        // Set session variable
15        $_SESSION['user'] = $user;
16
17        // Redirect based on role
18        if ($user['role'] === 'admin') {
19            header("Location: admin_dashboard.php"); // Redirect to admin dashboard
20        } elseif ($user['role'] === 'teacher') {
21            header("Location: teacher_dashboard.php"); // Redirect to teacher dashboard
22        } elseif ($user['role'] === 'student') {
23            header("Location: student_dashboard.php"); // Redirect to student dashboard
24        }
25        exit(); // Ensure script stops here after redirect
26    } else {
27        echo "Invalid username or password.";
28    }
29 }
30 ?>
```

It is responsible for processing the user input, authenticating the user, and determining the appropriate action based on the user's role.

Key backend features:

- **Session Management:** The script initializes a session using `session_start()`, allowing for persistent user data management across pages.
- **Database Interaction:** Using **PDO** for secure database operations, the script fetches the user details from the users table based on the provided username. The `prepare()` and `execute()` methods protect against **SQL injection**.

- **Role-Based Access Control:** Depending on the user's role (admin, teacher, or student), the script redirects the user to the corresponding dashboard page. This ensures users can only access features appropriate for their role.
- **Error Handling:** Basic error handling is implemented to inform users if their login attempt fails due to invalid credentials.
- **Security Measures:** The use of prepared statements and proper session handling enhances the security of the backend system.

The backend ensures secure and reliable handling of data, validates user credentials, and facilitates seamless communication between the user interface and the database.

### Grade Sheet:

```
// Grade Upload Section
if (isset($_POST['student_id']) && isset($_POST['exam_id']) && isset($_POST['grade'])) {
    $student_id = $_POST['student_id'];
    $exam_id = $_POST['exam_id'];
    $grade = $_POST['grade'];

    // Insert grade details into the grades table
    $stmt = $pdo->prepare("INSERT INTO grades (student_id, exam_id, grade) VALUES (?, ?, ?)");
    $stmt->execute([$student_id, $exam_id, $grade]);

    $grade_success_message = "Grade successfully uploaded.";
}
?>
```

The backend part of the grade sheet functionality is responsible for processing and storing the grade data in the database. Here's a detailed look:

### **Functionality:**

#### **1. Input Validation:**

- Checks if student\_id, exam\_id, and grade fields are set in the POST request.

#### **2. Database Interaction:**

- Prepares an SQL statement to insert the grade details into the grades table.
- Executes the statement with the provided values (student\_id, exam\_id, and grade).

### 3. Success Message:

- Provides feedback by setting a success message (\$grade\_success\_message).

### Database Tables:

- grades: Stores the association of grades with specific students and exams:
  - student\_id: References the student receiving the grade.
  - exam\_id: References the specific exam.
  - grade: The grade awarded to the student.

### Logout:

```
C: > Users > TASNIM > Downloads > 📄 logout.php
1  <?php
2  session_start();
3  session_destroy();
4  header("Location: ../index.php");
5  ?>
6
```

This PHP script is responsible for handling the user logout process. Here's a breakdown of its functionality:

#### 1. Session Start:

- session\_start(); is called to resume the current session, allowing access to session data (such as user information) that was previously stored.

#### 2. Destroying the Session:

- session\_destroy(); is used to terminate the session and remove all session variables. This effectively logs the user out by deleting any session data, including user authentication information.

#### 3. Redirecting to the Homepage:

- header("Location: ../index.php"); sends a header to the browser, instructing it to redirect the user to the homepage (index.php) after the session is destroyed. The header function must be called before any output is sent to the browser, ensuring the redirection occurs properly.

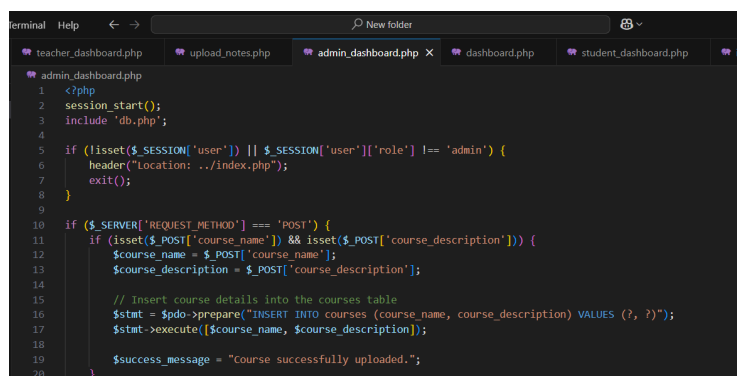
## Contribution of ID : 22301163, Name : Tamanna Islam Tazin

### 1. Admin Dashboard:

#### i)course management:

Back-End (What happens in the background):

- When the admin submits the form:
  - The course name and description are checked to make sure they're not empty.
  - The data is securely saved into the database using PHP (to a table named courses).
- Example: Saving a course called "Math 101" with its description into the database.



```
terminal Help ← → New folder
teacher_dashboard.php upload_notes.php admin_dashboard.php X dashboard.php student_dashboard.php ind
admin_dashboard.php
1 <?php
2 session_start();
3 include 'db.php';
4
5 if (!isset($_SESSION['user']) || $_SESSION['user']['role'] != 'admin') {
6     header("Location: ../index.php");
7     exit();
8 }
9
10 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
11     if (isset($_POST['course_name']) && isset($_POST['course_description'])) {
12         $course_name = $_POST['course_name'];
13         $course_description = $_POST['course_description'];
14
15         // Insert course details into the courses table
16         $stmt = $pdo->prepare("INSERT INTO courses (course_name, course_description) VALUES (?, ?)");
17         $stmt->execute([$course_name, $course_description]);
18
19         $success_message = "Course successfully uploaded.";
20     }
21 }
```

---

#### ii). Exam Management

Back-End (What happens in the background):

- When the admin submits the form:
  - The exam name, date, description, and selected course are checked to ensure they're valid.
  - The exam details are securely saved in the database using PHP (to a table named **exams**).



- Example: Adding an exam called "Midterm" on "2025-01-15" for "Math 101."

```

21
22 // Exam Upload Section
23 if (isset($_POST['exam_name']) && isset($_POST['exam_date']) && isset($_POST['exam_description']) && isset($_POST['course_id'])) {
24     $exam_name = $_POST['exam_name'];
25     $exam_date = $_POST['exam_date'];
26     $exam_description = $_POST['exam_description'];
27     $course_id = $_POST['course_id'];
28
29     // Insert exam details into the exams table
30     $teacher_id = $_SESSION['user']['id'];
31     $stmt = $pdo->prepare("INSERT INTO exams (exam_name, exam_date, exam_description, course_id, created_by) VALUES (?, ?, ?, ?, ?)");
32     $stmt->execute([$exam_name, $exam_date, $exam_description, $course_id, $teacher_id]);
33
34     $exam_success_message = "Exam successfully uploaded.";
35 }
36

```

## Student Dashboard:

### 1. Course Enrollment:

#### Back-End (What happens in the background):

- When a student selects a course and clicks "Enroll":
  - The system checks that the selected course exists.
  - It saves the student's ID and course ID into a database table (e.g., [student\\_courses](#)).
- Example: If Student ID 1 selects "Math 101" (Course ID 3), the system stores this link in the database.

```

21 // Enroll in a course
22 if ($SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['course_id'])) {
23     $course_id = $_POST['course_id'];
24
25     $stmt = $pdo->prepare("INSERT INTO student_courses (student_id, course_id) VALUES (?, ?)");
26     $stmt->execute([$student_id, $course_id]);
27
28     header("Location: student_dashboard.php");
29     exit();
30 }
31
32 // DOCTYPE html
33 <html>
34 <head>
35     <link rel="stylesheet" href="style.css">
36

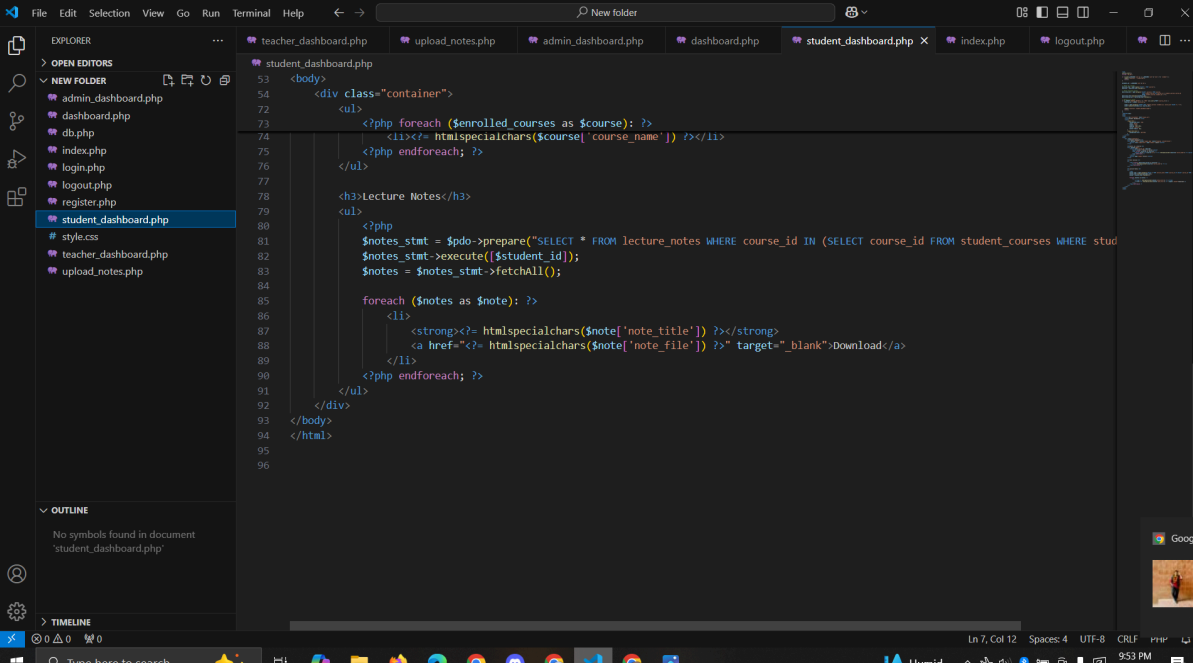
```

### 2. Lecture Notes Access:

#### Back-End (What happens in the background):

- The system checks which courses the student is enrolled in (from the [student\\_courses](#) table).

- It fetches the lecture notes for those courses from the database (e.g., **lecture\_notes** table).
- Example: If a student is enrolled in "Math 101," the system finds all notes linked to "Math 101" and displays them.



```

53 <body>
54 <div class="container">
72 <ul>
73 <?php foreach ($enrolled_courses as $course): ?>
74 <li><?php htmlspecialchars($course['course_name']) ?></li>
75 <?php endforeach; ?>
76 </ul>
77
78 <h3>Lecture Notes</h3>
79 <ul>
80 <?php
81 $notes_stmt = $pdo->prepare("SELECT * FROM lecture_notes WHERE course_id IN (SELECT course_id FROM student_courses WHERE stud
82 $notes_stmt->execute($student_id);
83 $notes = $notes_stmt->fetchAll();
84
85 foreach ($notes as $note): ?>
86 <li>
87 <strong><?php htmlspecialchars($note['note_title']) ?></strong>
88 <a href="<?php htmlspecialchars($note['note_file']) ?>" target="_blank">Download</a>
89 </li>
90 <?php endforeach; ?>
91 </ul>
92 </div>
93 </body>
94 </html>
95
96

```

## Teacher Dashboard:

### 1. Course Assignment

Back-End (What happens in the background):

- When a teacher selects a course and clicks "Assign":
  - The system checks that the selected course exists.
  - It saves the teacher's ID and the course ID into a database table (e.g., **teacher\_courses**).
- Example: If Teacher ID **5** selects "Math 101" (Course ID **3**), the system stores this link in the database.

```

23 // Assign a course to the teacher
24 if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['course_id'])) {
25     $course_id = $_POST['course_id'];
26
27     $stmt = $pdo->prepare("INSERT INTO teacher_courses (teacher_id, course_id) VALUES (?, ?)");
28     $stmt->execute([$teacher_id, $course_id]);
29
30     header("Location: teacher_dashboard.php");
31     exit();
32 }

```

## 2. Lecture Note Upload:

Back-End (What happens in the background):

- When a teacher uploads lecture notes:
  - The system checks that the selected course is assigned to the teacher.
  - Saves the title and uploaded file's path into a database table (e.g., **lecture\_notes**).
  - Links the notes to the corresponding course.
- Example: If Teacher ID **5** uploads "Week 1 Notes" for "Math 101," the system stores this information and file path in the **lecture\_notes** table.

```

53 <title>Admin Dashboard</title>
100 </head>
101 </body>
102 <body>
103     <h2>Admin Dashboard</h2>
104
105     <a href="logout.php" class="logout">Logout</a>
106
107     <!-- Course Upload Section -->
108     <h3>Upload New Course</h3>
109     <?php if (isset($success_message)) echo "<p>$success_message</p>"; ?>
110     <form method="POST">
111         <input type="text" name="course_name" placeholder="Course Name" required>
112         <textarea name="course_description" placeholder="Course Description" required></textarea>
113         <button type="submit">Upload Course</button>
114     </form>
115
116     <!-- Exam Upload Section -->
117     <h3>Upload New Exam</h3>
118     <?php if (isset($exam_success_message)) echo "<p>$exam_success_message</p>"; ?>
119     <form method="POST">
120         <input type="text" name="exam_name" placeholder="Exam Name" required>
121         <input type="date" name="exam_date" required>
122         <textarea name="exam_description" placeholder="Exam Description" required></textarea>
123         <select name="course_id" required>
124             <option value="">Select Course</option>
125             <?php
126                 // Fetch courses from the database
127                 $courses_stmt = $pdo->query("SELECT id, course_name FROM courses");
128                 while ($course = $courses_stmt->fetch()) {
129                     echo "<option value='" . $course['id'] . "'>" . htmlspecialchars($course['course_name']) . "</option>";
130                 }
131             <?php
132         </select>
133         <button type="submit">Upload Exam</button>
134     </form>
135
136     <!-- End of Exam Upload Section -->

```

## Contribution of ID : 22301504, Name : Abdul Kaium Choudhury

### Teacher Dashboard:

```
<?php
session_start();
include 'db.php';

if (!isset($_SESSION['user']) || $_SESSION['user']['role'] !== 'teacher') {
    header("Location: ../index.php");
    exit();
}

$teacher_id = $_SESSION['user']['id'];

// Fetch all courses
$courses_stmt = $pdo->query("SELECT * FROM courses");
$courses = $courses_stmt->fetchAll();

// Fetch courses assigned to this teacher
$assigned_stmt = $pdo->prepare("SELECT courses.* FROM courses
                                JOIN teacher_courses ON courses.id = teacher_courses.course_id
                                WHERE teacher_courses.teacher_id = ?");
$assigned_stmt->execute([$teacher_id]);
$assigned_courses = $assigned_stmt->fetchAll();

// Assign a course to the teacher
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['course_id'])) {
    $course_id = $_POST['course_id'];

    $stmt = $pdo->prepare("INSERT INTO teacher_courses (teacher_id, course_id) VALUES (?, ?)");
    $stmt->execute([$teacher_id, $course_id]);

    header("Location: teacher_dashboard.php");
    exit();
}
?>
```

In this backend for teacher's dashboard,

### 1. Session Management and User Authentication:

- Initiates a PHP session using `session_start()`.
- Includes the database connection file (`db.php`).
- Checks for the existence of a user session (`$_SESSION['user']`).
- Verifies if the user is a teacher (`$_SESSION['user']['role'] === 'teacher'`).
- Redirects unauthorized users (non-teachers or unauthenticated users) to the login page (`../index.php`).

### 2. Data Retrieval:

- Retrieves the teacher's ID from the session data (`$_SESSION['user']['id']`).
- Fetches all courses from the `courses` table using a simple `SELECT` query.
- Retrieves courses specifically assigned to the current teacher using a `JOIN` query between `courses` and `teacher_courses` tables.
- Stores the retrieved course data in variables (`$courses` and `$assigned_courses`).

### 3. Course Assignment (if POST Request):

- Checks if the request method is `POST` and a `course_id` is present in the `POST` data.
- Extracts the `course_id` from the `POST` data.
- Uses a prepared statement to insert a new record into the `teacher_courses` table, associating the `course_id` with the current teacher's ID.

- Redirects the user back to the teacher dashboard (teacher\_dashboard.php) after successful assignment.

## Student Dashboard:

```
<?php
session_start();
include 'db.php';

if (!isset($_SESSION['user']) || $_SESSION['user']['role'] !== 'student') {
    header("Location: ../index.php");
    exit();
}

$student_id = $_SESSION['user']['id'];

// Fetch all courses
$courses_stmt = $pdo->query("SELECT * FROM courses");
$courses = $courses_stmt->fetchAll();

// Fetch enrolled courses
$enrolled_stmt = $pdo->prepare("SELECT courses.* FROM courses
                                JOIN student_courses ON courses.id = student_courses.course_id
                                WHERE student_courses.student_id = ?");
$enrolled_stmt->execute([$student_id]);
$enrolled_courses = $enrolled_stmt->fetchAll();

// Enroll in a course
if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_POST['course_id'])) {
    $course_id = $_POST['course_id'];

    $stmt = $pdo->prepare("INSERT INTO student_courses (student_id, course_id) VALUES (?, ?)");
    $stmt->execute([$student_id, $course_id]);

    header("Location: student_dashboard.php");
    exit();
}
?>
```

For this backend part of student dashboard,

## Session Management and User Validation:

- The session\_start() function is called to start the session, which allows accessing session data across multiple pages.
- The code checks if the user is logged in and if their role is student. If either condition is not met, the user is redirected to the index.php page (likely a login page).
- The student\_id is fetched from the session data, which contains the ID of the logged-in student.

## Fetching Courses:

- The query SELECT \* FROM courses retrieves all the available courses from the database. This is done using a simple PDO::query method, which is executed without any user input (i.e., static).

## Fetching Enrolled Courses:

- A prepared SQL statement is used to fetch all courses in which the student is enrolled. It performs a join between courses and student\_courses tables where the

student\_courses.student\_id matches the logged-in student's ID. The result is stored in the \$enrolled\_courses array.

### **Course Enrollment:**

- If the form is submitted via a POST request, the student can enroll in a course by selecting one from the dropdown.
- The course ID from the form is then inserted into the student\_courses table (which links students to courses) using a prepared statement.
- After the course is successfully enrolled, the page reloads (redirects) to student\_dashboard.php.

### **Fetching and Displaying Lecture Notes:**

- A query retrieves the lecture notes for the courses the student is enrolled in by joining the student\_courses table with lecture\_notes. This is done using a subquery that fetches the course\_id from student\_courses based on the logged-in student's ID.
- The fetched notes are displayed with links to download the files.

## Registration:

```
1 <?php
2 include 'db.php';
3 session_start();
4
5 if ($_SERVER['REQUEST_METHOD'] === 'POST') {
6     $email = $_POST['email'];
7     $username = $_POST['username'];
8     $password = password_hash($_POST['password'], PASSWORD_DEFAULT);
9     $role = $_POST['role'];
10
11     // Step 1: Validate email
12     if (empty($email)) {
13         echo "<p>Email cannot be empty.</p>";
14         exit();
15     }
16
17     if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
18         echo "<p>Invalid email format.</p>";
19         exit();
20     }
21
22     // Step 2: Check if email already exists in the database
23     $check_email_stmt = $pdo->prepare("SELECT 1 FROM users WHERE email = ?");
24     $check_email_stmt->execute([$email]);
25
26     if ($check_email_stmt->rowCount() > 0) {
27         echo "<p>This email is already registered. Please use another email.</p>";
28         exit();
29     }
30
31     // Step 3: Insert user into the database
32     try {
33         $stmt = $pdo->prepare("INSERT INTO users (email, username, password, role) VALUES (?, ?, ?, ?)");
34         $stmt->execute([$email, $username, $password, $role]);
35
36         // Fetch the newly registered user
37         $user_stmt = $pdo->prepare("SELECT * FROM users WHERE email = ?");
38         $user_stmt->execute([$email]);
39         $user = $user_stmt->fetch();
40
41         // Automatically log in the user
42         $_SESSION['user'] = $user;
43
44         // Redirect to the appropriate dashboard based on role
45         if ($user['role'] === 'admin') {
46             header("Location: admin_dashboard.php");
47         } elseif ($user['role'] === 'teacher') {
48             header("Location: teacher_dashboard.php");
49         } elseif ($user['role'] === 'student') {
50             header("Location: student_dashboard.php");
51         }
52         exit();
53     } catch (Exception $e) {
54         echo "<p>Error: " . $e->getMessage() . "</p>";
55     }
56 }
```

The backend part of the code handles the form submission and manages the user registration process. Here's a breakdown of the steps:

### 1. Session Initialization:

- `session_start()`; is called to initialize a session that allows storing user data for login management. The session will be used to store the user's information after they successfully register.

### 2. Data Collection:

- The `$_POST` array is used to collect the form input data (email, username, password, and role).
- The password is hashed using `password_hash()` with the `PASSWORD_DEFAULT` algorithm to securely store it in the database.

### 3. Email Validation:

- The email field is checked to ensure it's not empty.
  - The `filter_var($email, FILTER_VALIDATE_EMAIL)` function is used to validate the email format. If the email is invalid, an error message is displayed.
4. Check for Existing Email:
- A SQL query is executed to check if the entered email already exists in the users table. If the email is already registered, an error message is displayed, and the script exits.
5. Insert User into Database:
- If the email is valid and does not already exist in the database, a SQL INSERT statement is prepared and executed to insert the user's information into the users table.
  - The database connection is assumed to be handled by the `db.php` file (which is not provided).
6. Redirect Based on Role:
- Based on the user's role (admin, teacher, or student), the script redirects the user to the appropriate dashboard:
    - Admin: Redirects to `admin_dashboard.php`
    - Teacher: Redirects to `teacher_dashboard.php`
    - Student: Redirects to `student_dashboard.php`
7. Error Handling:
- If there's an error during the user registration or insertion process (e.g., database issues), an exception is caught and an error message is displayed to the user.



## Source Code Repository

Upload source code to GitHub or Google Drive and share a publicly accessible link in this section of the report.

Google Drive Source Link :

<https://drive.google.com/drive/folders/1pl3E3BWW9JbOjTIDjRsHD0SHEc8qLtva?usp=sharing>

## Conclusion

An organized and thorough method of handling academic data is demonstrated by the creation of the "Academic Gateway" database system, which is depicted in the Enhanced Entity-Relationship (EER) diagram. Along with their complex interactions and characteristics, the system successfully represents the main entities, including students, instructors, administrators, courses, tests, and grades.

This study highlights how crucial a well-structured database is to expediting the management of academic procedures. The database design guarantees data consistency, integrity, and accessibility for everything from course distribution and exam administration to student enrollment and grade management.

Additionally noteworthy are the "Academic Gateway" database's scalability and flexibility, which enable future improvements like adding sophisticated analytics or expanding features to accommodate more academic procedures. The design guarantees strong security measures, protecting data, by concentrating on user roles and permissions.

All things considered, this project demonstrates our capacity to assess requirements, convert them into an optimal database design, and offer a solution that satisfies actual academic management requirements. By working together and solving problems, we were able to lay the groundwork for a dependable and effective database system, opening the door for a fully functional application in the following stages of development.

---

## References

Faisal,(n.d.).PHP and MySQL ( Database connection, Login, Insert and Retrieve data from database) [YouTube playlist].YouTube.

- 

[https://www.youtube.com/playlist?list=PLJW6cU20q-SOQB1Dqj2vF9mcNqgh4-z\\_j](https://www.youtube.com/playlist?list=PLJW6cU20q-SOQB1Dqj2vF9mcNqgh4-z_j)