CSE370 : Database Systems
Project Report
Project Title: UniSystem (A Unified Task Management System)

Group No : 03, CSE370 Lab Section : 16, Fall 2025		
ID	**Name**	**Contribution**
24101544	Enan Mahmud	I built the **Admin Verification**, **Task Management**, and **Resources Sharing** systems, developed both the PHP backend logic and the corresponding user interfaces. I also built the majority of the backend of the Admin, Teacher, and Student dashboards. I used JavaScript for dynamic interactions, and integrated Google Fonts and Bootstrap Icons to enhance the UI/UX design for the features.
24101519	Akib Zawad	I have worked with **Role Based Access Systems**, **Student Enrollment Systems** and **Teacher Allocation Systems**. I designed the backend for them using PHP. I also designed the diagram and schema and the database of the project using MySQL.

Table of Contents

Introduction

Due to different courses and heavy workloads, keeping track of important academic dates has become difficult and overwhelming for both students and teachers. Moreover, the increasing demand for higher education has made it challenging for the university authority to supervise academic activities efficiently. Students often miss deadlines and search for important materials scattered across different media, while teachers have to manage multiple courses and maintain different communication channels to interact with students. At the same time, the efficiency of the authority is reduced due to excessive manual work.

Therefore, bringing all users under a single platform and creating a bridge among students, teachers, and the authority can significantly simplify and speed up the academic process. Keeping this scenario in mind and aiming to reduce the gaps between the authority, students, and teachers, we have designed a web-based management system named **UniSystem** (A Unified Task Management System) as our CSE370 Lab Project. By leveraging secure, role-based access control, the system ensures data integrity while providing a user-friendly interface for daily university operations.

The primary goal of this project is to decentralize communication channels (like WhatsApp, Discord, Google Classroom, Slack groups or email chains) in favor of a structured web platform.
Here, there are three types of users :

- Admin
- Teacher
- Student

Admin :

- Admin has a separate login page.
- Admin accounts are registered manually through the database.
- Admin can create courses and sections.
- Admin can enroll teachers into specific courses and sections.
- After registration, the admin verifies both students and teachers through the dashboard.
- Any academic material posted by a student requires admin approval to ensure academic consistency.

Teacher :

- Teachers need to register themselves for first-time access.
- After admin approval, teachers can:
    - View their assigned courses and sections
    - Post tasks and academic activities for students

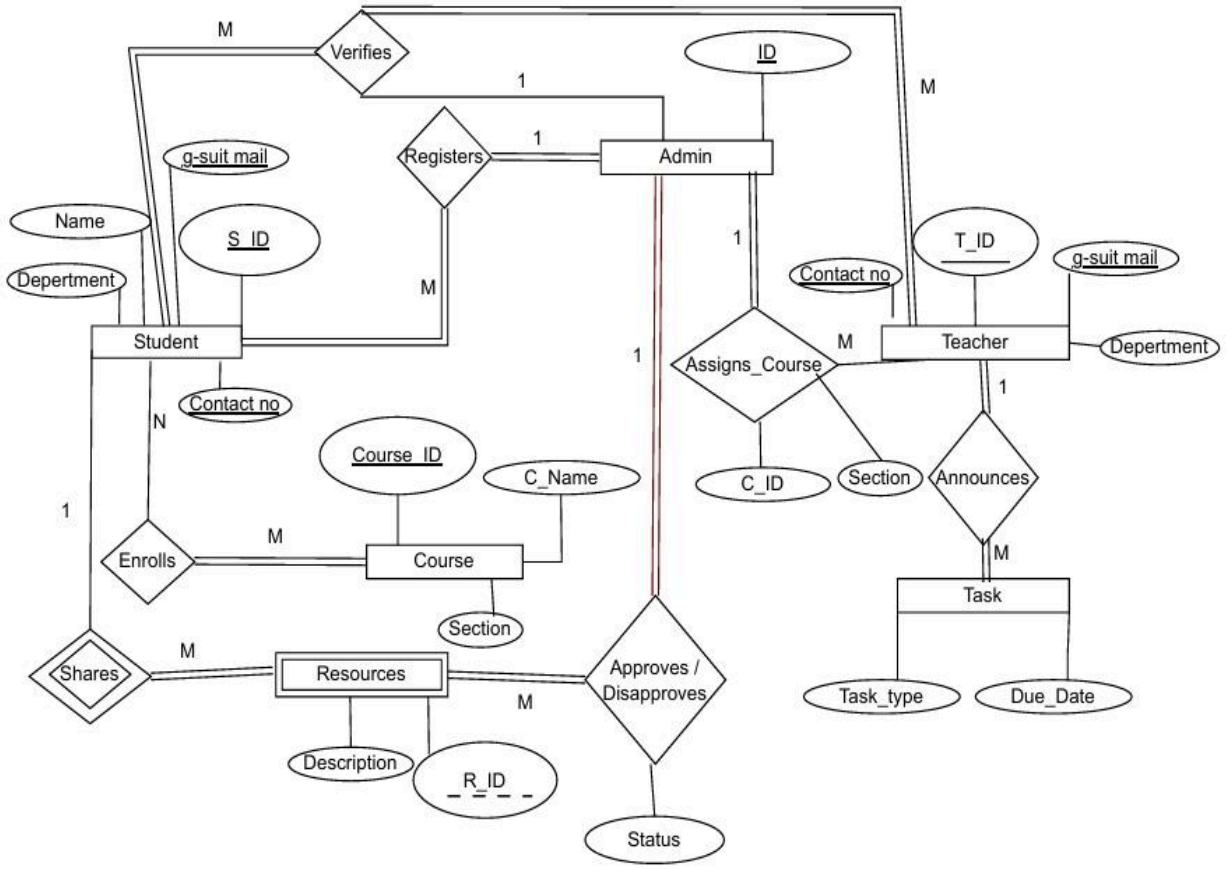
Students:

- Students also need to register themselves.
- After admin approval, students can:
    - Enroll in courses
    - View upcoming tasks and deadlines on their dashboard
    - Share academic materials in the form of text or links with others

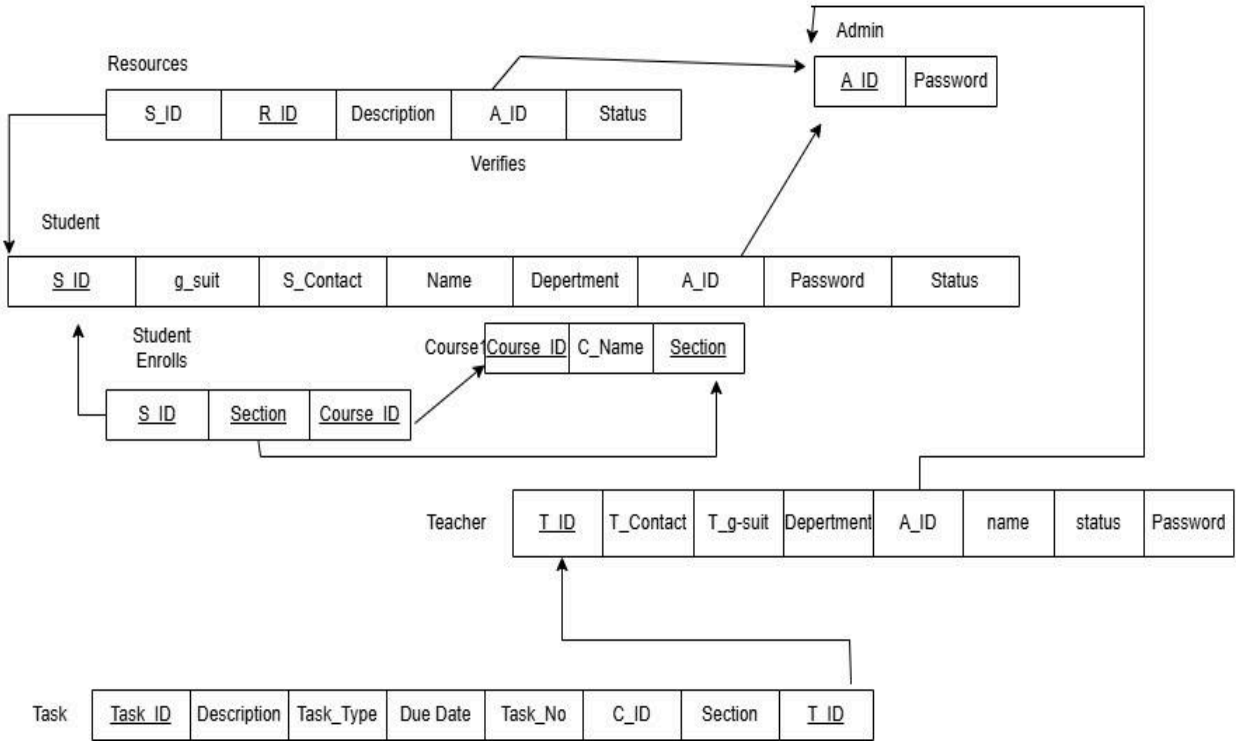
Project Features

ID, Name	Features [3 per member]	
24101544, Enan Mahmud	Ft 1	Admin User Verification System
	Ft 2	Task Management System
	Ft 3	Resources Sharing Center
24101519, Akib Zawad	Ft 1	Student Course Enrollment System
	Ft 2	Teacher Allocation System
	Ft 3	Role Based Access Control System

ER/EER Diagram



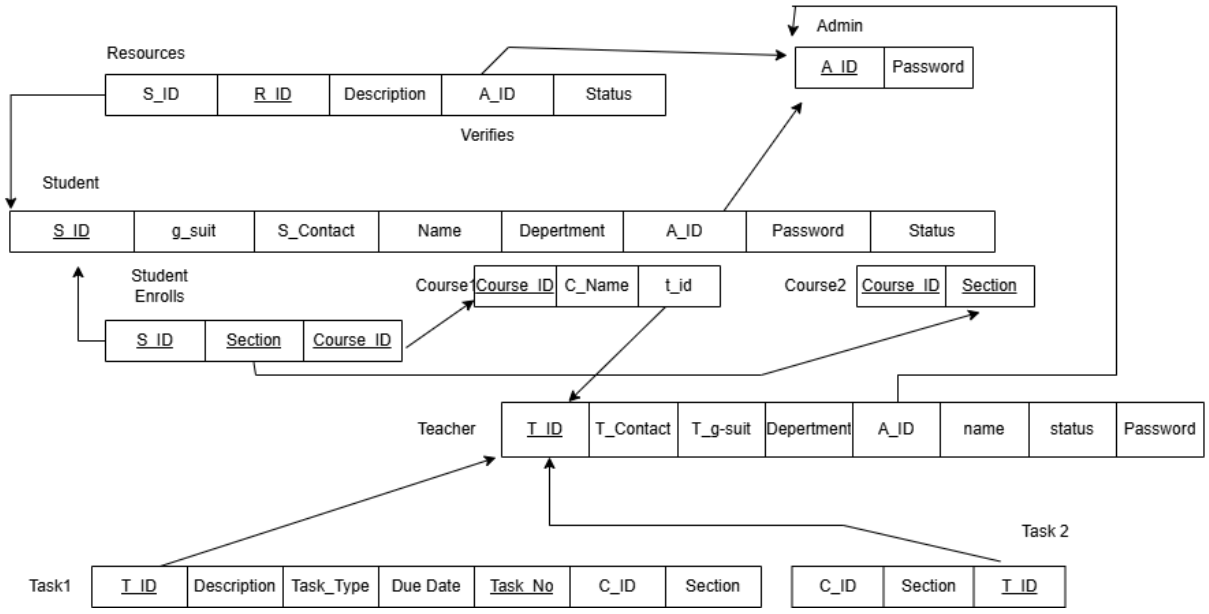
Schema Diagram



Normalization

a. Explain if your converted Schema is in 1NF or not. If not, decompose it to 1NF.
   - Yes, it is in 1NF
b. Explain if your converted Schema is in 2NF or not. If not, decompose it to 2NF. Can there be any partial functional dependencies in your relational schema?
   -The initial schema was not in 2NF because Section depended only on Course_ID (Partial Dependency). We decomposed the table to remove this, ensuring the final schema is in 2NF.
c. Explain if your converted Schema is in 3NF or not. If not, decompose it to 3NF. Can there be any transitive dependencies in your relational schema?
   -Yes, the converted schema is in 3NF. In the initial schema, in the task table, course_id depends on teacher_id and section depends on course_id, creating a transitive dependency. So, we had to decompose it into 3NF.

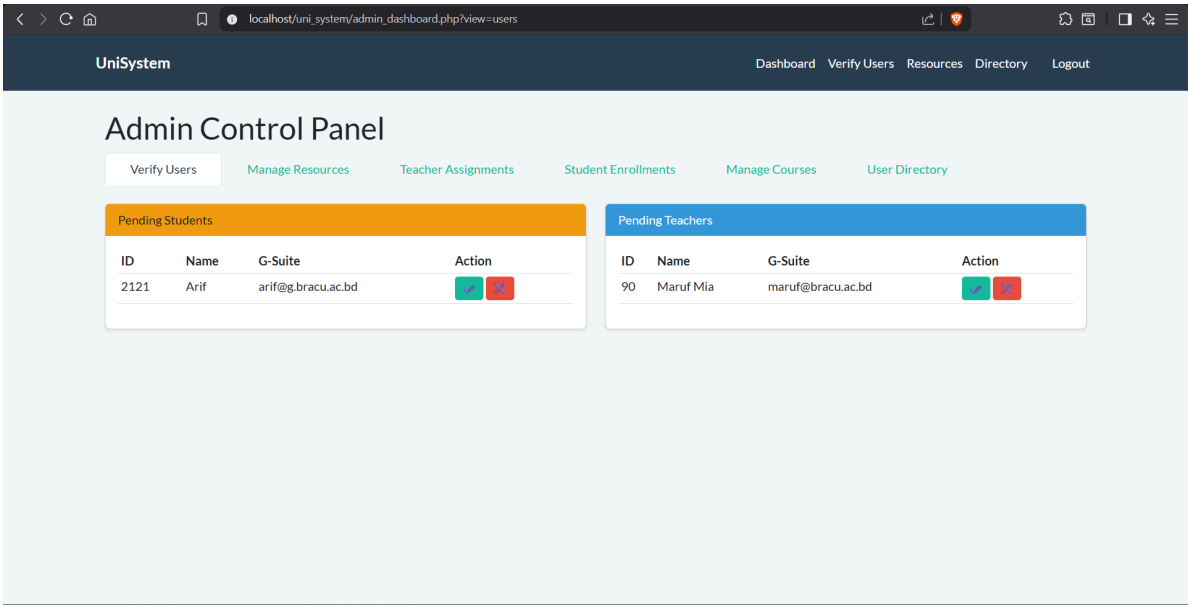
Normalized in 3NF:



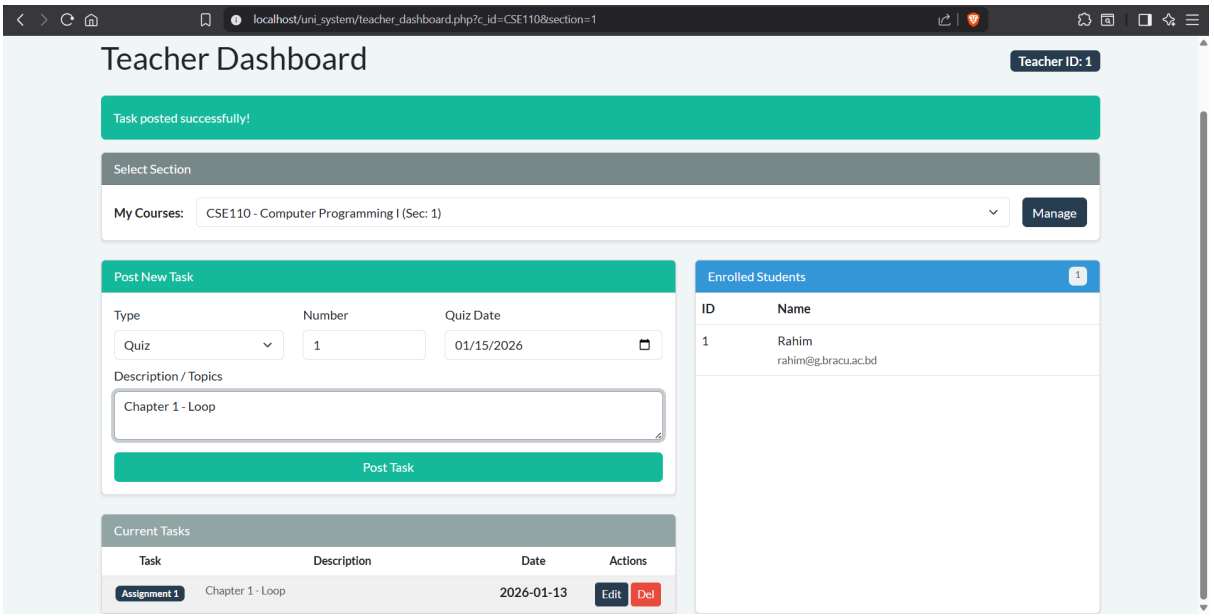
Frontend Development

Contribution of ID : 24101544, Name : Enan Mahmud

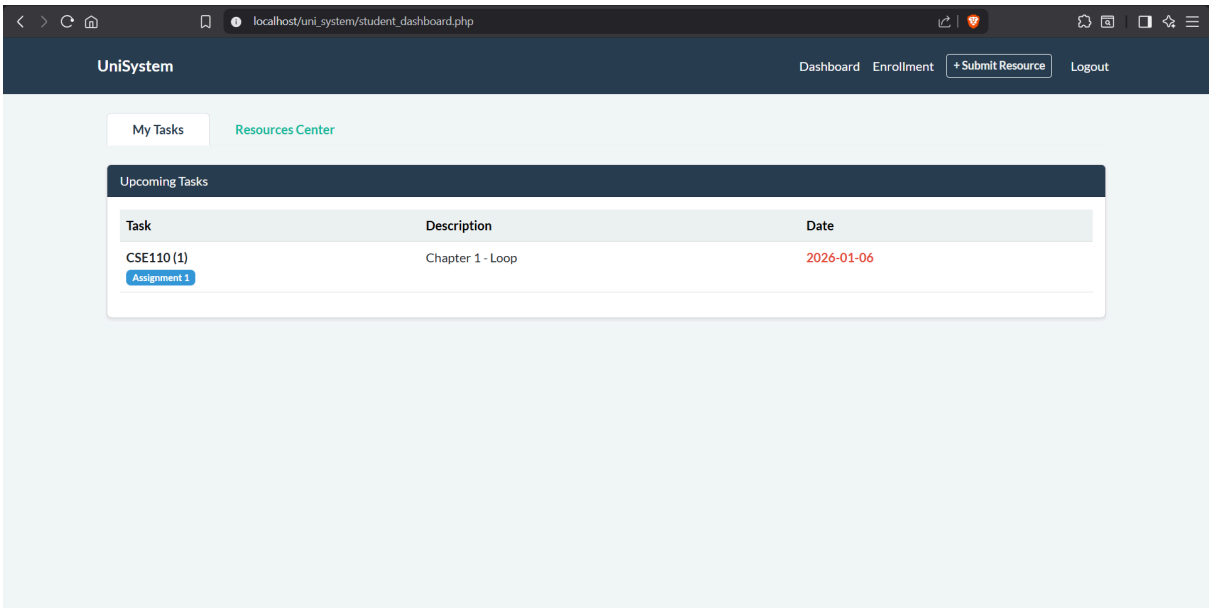
I designed the user interfaces for the **Admin User Verification**, **Task Management**, and **Resources Sharing Center**.
For these parts, I used HTML, CSS, and JavaScript, specifically implementing features like pop-up modals for file submissions. I applied the Bootswatch Flatly theme. I integrated Bootstrap Icons for the dashboard navigation and used Google Fonts.



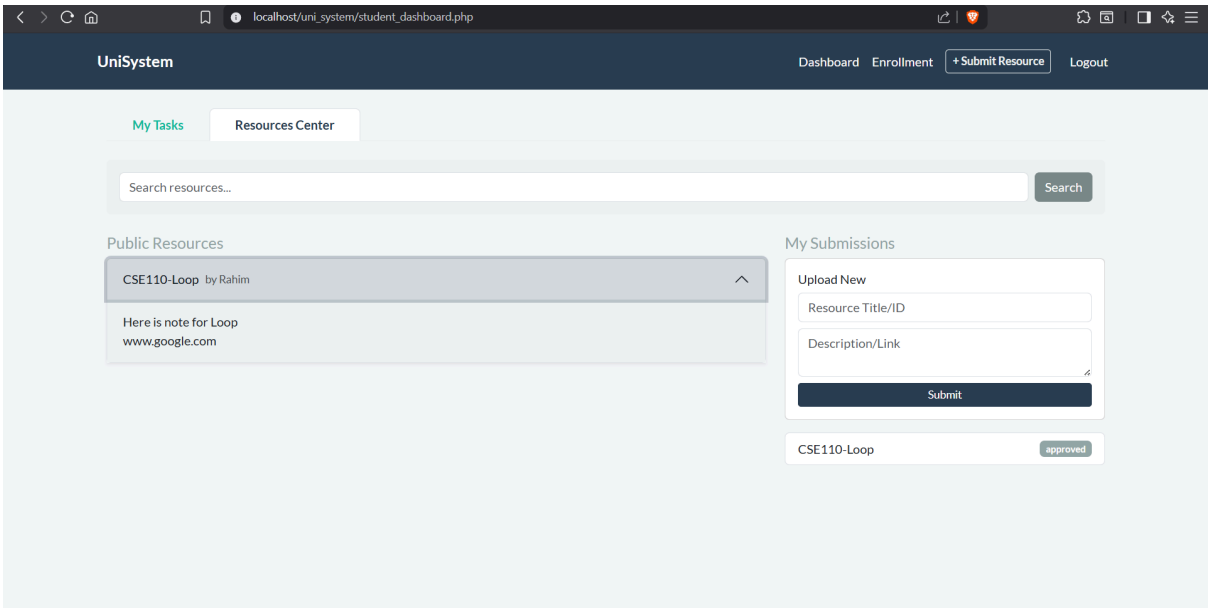
Admin User Verification



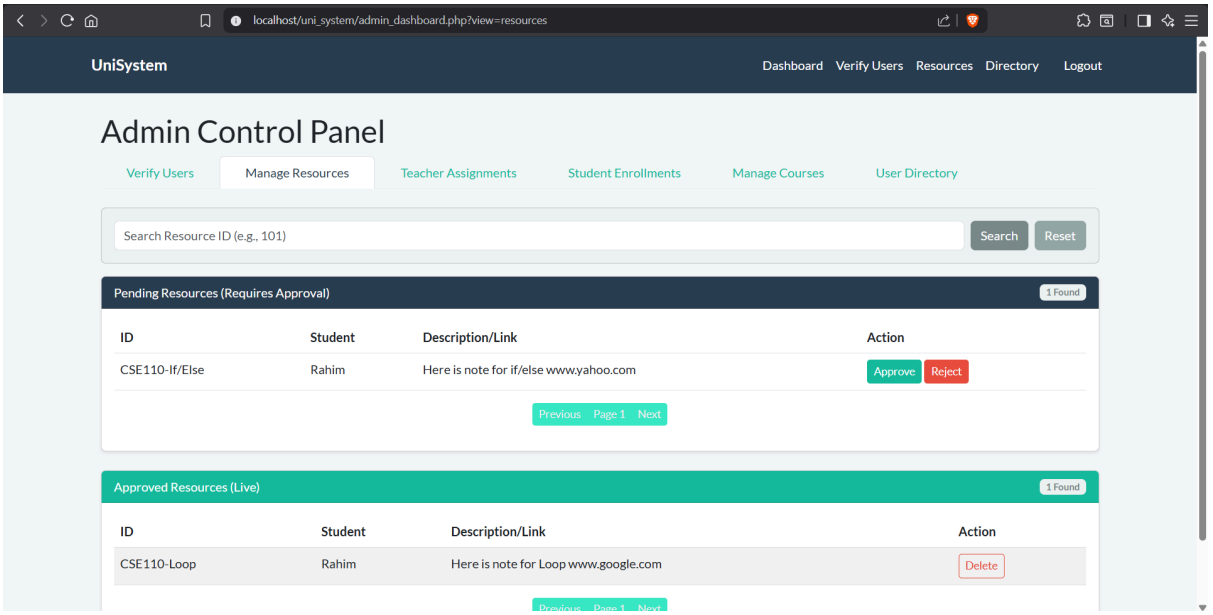
Task Management (Teacher dashboard)



Task Management (Student dashboard)



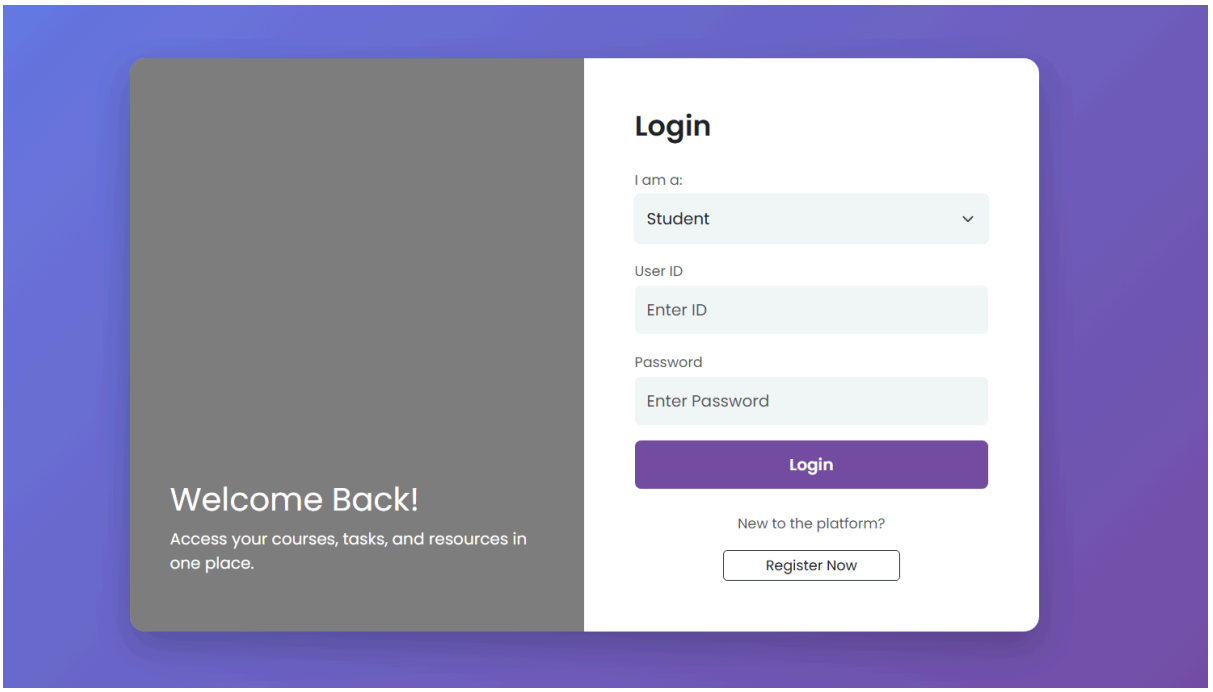
Resource Sharing (Student Dashboard)



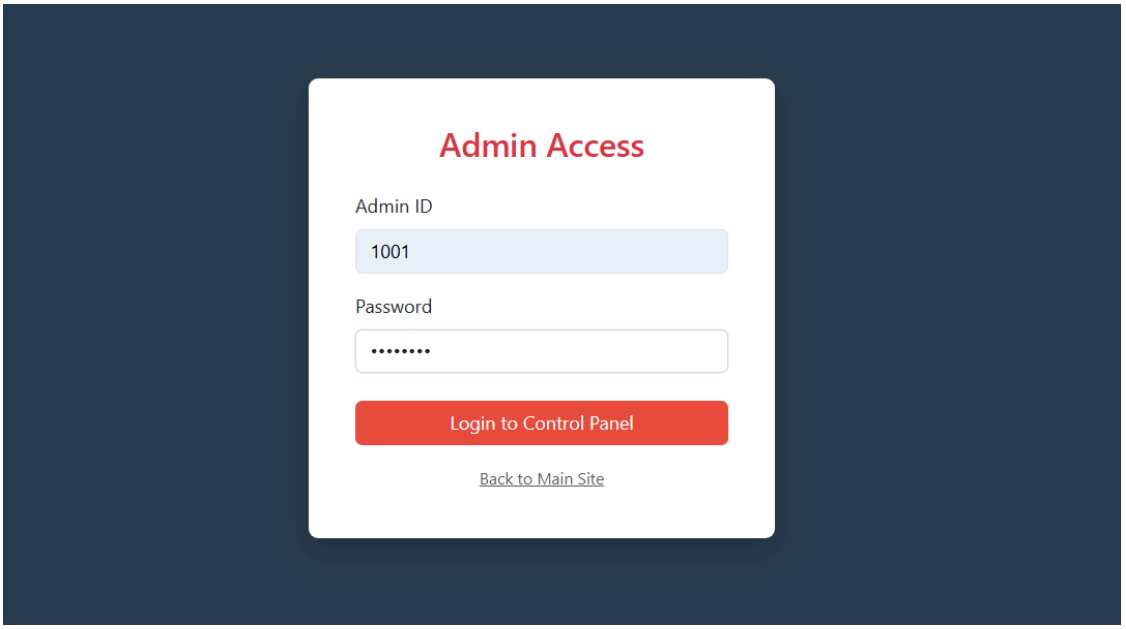
Resource Sharing (Admin Dashboard)

Contribution of ID : 24101519, Name : Akib Zawad

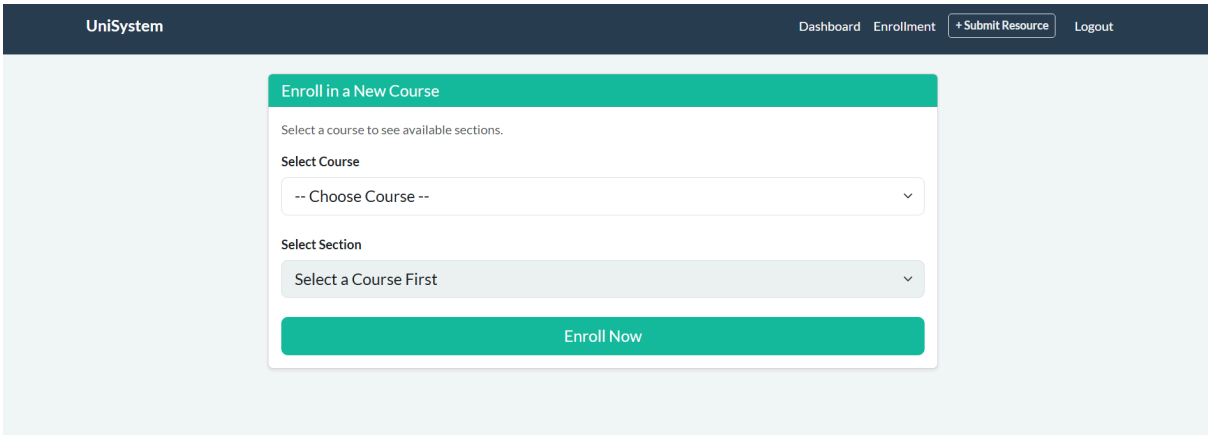
I have worked with **Students Course Enrollment**, **Teacher Allocation** and **Role Based Login System**. For these parts, we have used html, css framework bootstrap, javascript. Pre-built theme Bootswatch Flatly was used for UI design. For icons, we have used bootstrap icons. The font was designed using Google Fonts.



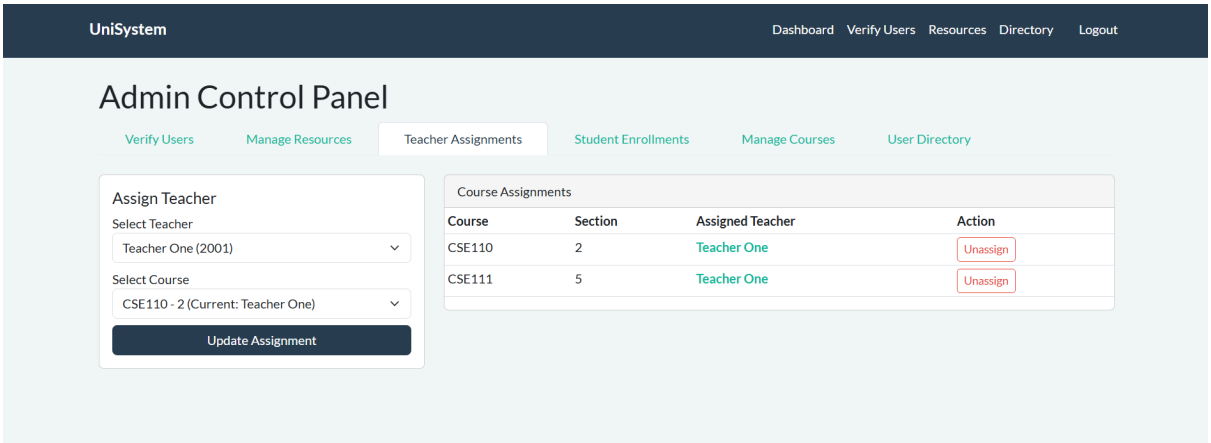
Role Based login system using Bootstrap



Admin login system using Bootstrap



Student Enrollment dashboard using Bootswatch (Theme name : Flatly)



Teacher allocation dashboard using Bootswatch (Theme name : Flatly)

Backend Development

For the backend of our project we have used **PHP** and **MySQL**. The connection file is **db_connect.php**, which connects our project files to the database named **university_db** using standard **MySQL credentials**.

Contribution of ID : 24101544, Name : Enan Mahmud

I developed the backend logic for the **User Verification**, **Resource Sharing**, and **Task Management** systems.
I also built the majority of the backend of the **Admin**, **Teacher**, and **Student dashboards**.

```php
if (isset($_POST['verify_user'])) {
    $target_id = $_POST['target_id'];
    $type = $_POST['type'];
    $status = $_POST['status'];
    $table = ($type == 'student') ? 'student' : 'teacher';
    $id_col = ($type == 'student') ? 's_id' : 't_id';

    $conn->query(query: "UPDATE $table SET account_status='$status', a_id='$admin_id' WHERE $id_col='$target_id'");
    $message = "<div class='alert alert-success'>User updated to $status!</div>";
}


if (isset($_POST['verify_resource'])) {
    $r_id = $_POST['r_id'];
    $status = $_POST['status'];
    $conn->query(query: "UPDATE resource SET status='$status', a_id='$admin_id' WHERE r_id='$r_id'");
    $message = "<div class='alert alert-success'>Resource updated!</div>";
}
```

User and Resources Verification (admin_dashboard.php)

User and Resources Verification System:
Here all new user registrations and resource uploads are initialized with a '*pending*' status in the database. The backend logic employs dynamic SQL UPDATE queries to toggle the account_status (for users) or status (for resources) columns to '*active*' or '*approved*' based on admin input.

```php
if (isset($_POST['upload_resource'])) {
    $r_id_input = trim(string: $_POST['r_id_text']);
    $desc = trim(string: $_POST['description']);

    if(empty($r_id_input) || empty($desc)) {
        $message = "<div class='alert alert-danger'>Error: All fields are required.</div>";
    } else {
        $check = $conn->query(query: "SELECT * FROM resource WHERE r_id = '$r_id_input'");
        if($check->num_rows > 0) {
            $r_id_input = $r_id_input . "-" . rand(min: 100, max: 999);
        }

        $sql = "INSERT INTO resource (r_id, s_id, description, status) VALUES ('$r_id_input', '$s_id', '$desc', 'pending')";

        if ($conn->query(query: $sql)) {
            $message = "<div class='alert alert-success'>Resource submitted!</div>";
        } else {
            $message = "<div class='alert alert-danger'>Database Error: " . $conn->error . "</div>";
        }
    }
}
```

Students Sharing Resources (student_dashboard.php)

Resource Sharing System:

I implemented a flag-based content moderation workflow that secures user uploads. When a resource is submitted, the backend performs an INSERT operation with a '*pending*' status. The student view uses a SELECT query with a WHERE status='*approved*', ensuring only admin-verified content is rendered.

```php
if (isset($_POST['post_task'])) {
    $c_id = $_POST['c_id'];
    $sec = $_POST['section'];

    $check_auth = $conn->query(query: "SELECT * FROM course WHERE c_id='$c_id' AND section='$sec' AND t_id='$t_id'");

    if ($check_auth->num_rows > 0) {
        $task_id = uniqid(prefix: "T");
        $type = $_POST['type'];
        $due = $_POST['due_date'];

        $task_num = $_POST['task_number'];
        $desc = $_POST['description'];

        $sql = "INSERT INTO task (task_id, teacher_id, c_id, section, due_date, type, task_number, description)
                VALUES ('$task_id', '$t_id', '$c_id', '$sec', '$due', '$type', '$task_num', '$desc')";

        if ($conn->query(query: $sql)) $message = "<div class='alert alert-success'>Task posted successfully!</div>";
        else $message = "<div class='alert alert-danger'>Error: " . $conn->error . "</div>";
    } else {
        $message = "<div class='alert alert-danger'>Security Alert: You are not assigned to this course.</div>";
    }
}
```

Teacher Sharing Task (teacher_dashboard.php)

```php
if ($page_view == 'dashboard') {
    $today = date(format: 'Y-m-d');

    $task_sql = "SELECT t.* FROM task t
                 JOIN student_enrolls se ON t.c_id = se.c_id AND t.section = se.section
                 WHERE se.s_id = '$s_id' AND t.due_date >= '$today'
                 ORDER BY t.due_date ASC";
    $tasks = $conn->query(query: $task_sql);
```

Student Viewing Task (student_dashboard.php)

Task Management System:

I designed a relational logic flow linking course owners to enrolled students. The teacher can post new tasks using the INSERT operation. The student dashboard uses an INNER JOIN between the task and student_enrolls tables, ensuring users only retrieve tasks corresponding to the specific course sections they are currently attending.

Contribution of ID : 24101519, Name : Akib Zawad

Role Based Access:

The login of admin is handled by **admin_login.php**.

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

$error = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $id = $_POST['a_id'];
    $pass = $_POST['password'];

    // Check ONLY Admin table
    $sql = "SELECT * FROM admin WHERE a_id='$id' AND password='$pass'";
    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        $_SESSION['user_id'] = $id;
        $_SESSION['role'] = 'admin';
        header("Location: admin_dashboard.php");
        exit();
    } else {
        $error = "Invalid Admin Credentials";
    }
}
?>
```

Admin Login Backend (admin_login.php)

For the login of the students and teachers the user type is stored in role variable from the dropdown in **index.php** file then it checks if the id and password matches with the table and if the approval status is active then it redirects to the user dashboard or it says "Login Failed! check your ID/Password or wait for Admin Approval."

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

$error = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $id = $_POST['user_id'];
    $pass = $_POST['password'];
    $role = $_POST['role'];


    if ($role == 'student') {
        $sql = "SELECT * FROM student WHERE s_id='$id' AND password='$pas
s' AND account_status='active'";
    } elseif ($role == 'teacher') {
        $sql = "SELECT * FROM teacher WHERE t_id='$id' AND password='$pas
s' AND account_status='active'";
    } else {

        die("Invalid role selected");
    }

    $result = $conn->query($sql);

    if ($result->num_rows > 0) {
        $_SESSION['user_id'] = $id;
        $_SESSION['role'] = $role;
        header("Location: " . $role . "_dashboard.php");
        exit();
    } else {
        $error = "Login Failed! check your ID/Password or wait for Admin A
pproval.";
    }
}
?>
```

Backend code for ensuring Student and Teacher role based access (index.php)

Student's Course Enrollment System:

The **student_enrolls** table links a student's ID (s_id) with a course ID (c_id) and section. Before enrollment, the backend runs a SELECT query to check if the student is already enrolled in that specific course section. If a record exists, it blocks the request to prevent duplicates. The **enroll.php** file contains logic to fetch the student's department and apply a course limit (7 for 'PHR'/'LLB' students, 5 for others). Finally if validations pass, an

INSERT INTO query adds the new record to the database.

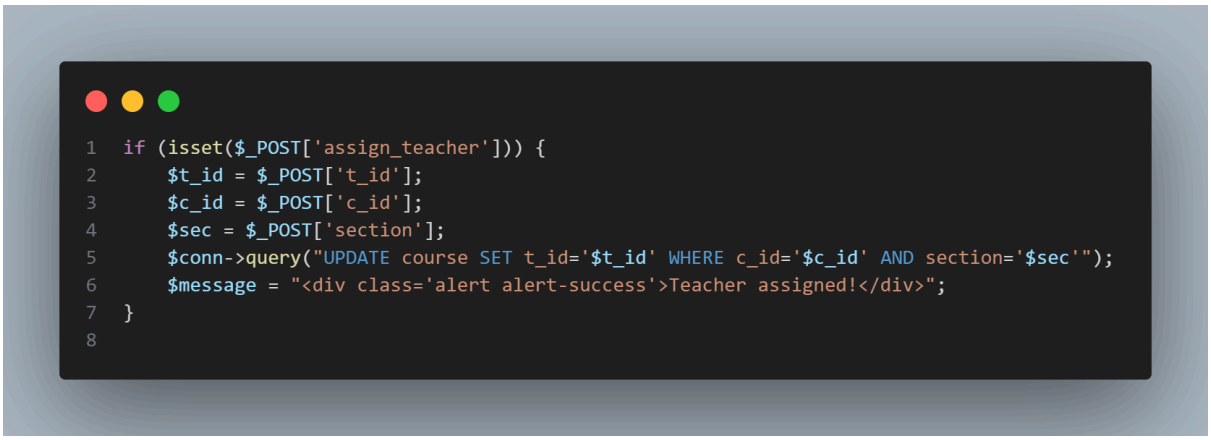
```php
<?php
include 'header.php';
include 'db_connect.php';

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

$s_id = $_SESSION['user_id'];
$message = "";


$s_info = $conn->query("SELECT department FROM student WHERE s_id='$s_id'")->fetch_assoc();
$dept = strtoupper($s_info['department']);

// course limit
$limit = ($dept == 'PHR' || $dept == 'LLB') ? 7 : 5;

//total enroll

$count_res = $conn->query("SELECT COUNT(*) as total FROM student_enrolls WHERE s_id='$s_id'");
$current_count = $count_res->fetch_assoc()['total'];

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $c_id = $_POST['c_id'];
    $section = $_POST['section'];


    if ($current_count >= $limit) {
        $message = "<div class='alert alert-danger'>Limit Reached! $dept students can max take $limit courses.</div>";
    } else {

        //duplicate course check
        $dup_check = $conn->query("SELECT * FROM student_enrolls WHERE s_id='$s_id' AND c_id='$c_id'");

        if ($dup_check->num_rows > 0) {
            $message = "<div class='alert alert-danger'>You are already enrolled in $c_id.
             You cannot take two sections of the same course.</div>";
        } else {

            $course_check = $conn->query("SELECT * FROM course WHERE c_id='$c_id' AND section='$section'");

            if ($course_check->num_rows > 0) {

                $sql = "INSERT INTO student_enrolls (s_id, c_id, section) VALUES ('$s_id', '$c_id', '$section')";
                if ($conn->query($sql)) {
                    $message = "<div class='alert alert-success'>Enrolled in $c_id ($section) successfully!</div>";
                    $current_count++;
                } else {
                    $message = "<div class='alert alert-danger'>Database Error.</div>";
                }
            } else {
                $message = "<div class='alert alert-warning'>Course/Section not found.</div>";
            }
        }
    }
}
?>
```

Backend code for student course enrollment (enroll.php)

Teacher Allocation System:

This feature allows Admins to link teachers to specific course sections, which subsequently unlocks permissions for those teachers.

The course table has a foreign key column t_id (Teacher ID) that can be null or linked to a specific teacher. When an admin assigns a teacher in admin_dashboard.php, the backend executes an UPDATE SQL query on the course table to set the t_id for that specific course and section. On the teacher_dashboard.php, the backend uses this link to verify authority.



```php
if (isset($_POST['assign_teacher'])) {
    $t_id = $_POST['t_id'];
    $c_id = $_POST['c_id'];
    $sec = $_POST['section'];
    $conn->query("UPDATE course SET t_id='$t_id' WHERE c_id='$c_id' AND section='$sec'");
    $message = "<div class='alert alert-success'>Teacher assigned!</div>";
}

```

Backend code for Teacher Allocation (admin_dashboard.php)

Source Code Repository

GitHub Repository Link : https://github.com/Akib-Zawad47/UniSystem

Conclusion

We developed the **UniSystem** (A Unified Task Management System) to solve the problems that we face in manual academic management by centralizing communication and resource sharing into a single, secure web-based platform. By integrating Role-Based Access Control, the system ensures data integrity and security, allowing Admins, Teachers, and Students to interact within their authorized scopes.

Throughout the development process, we successfully implemented core features such as the Admin Verification System, Task Management System, and Resource Sharing Center, all supported by a normalized database schema. We used PHP and MySQL for a reliable backend foundation, while the Bootswatch Flatly theme ensured a responsive and user-friendly frontend interface.

While the current version of UniSystem effectively decentralizes communication channels like Discord and Google Classroom, future iterations could include features such as real-time push notifications, assignment submission and grading. Ultimately, this project provided valuable hands-on experience in full-stack development and database architecture. The technical knowledge and problem-solving skills gained during this process will serve as a strong foundation for building scalable systems that solve real-world problems in the future.

References

For project setup and overall workflow idea:
https://youtube.com/playlist?list=PLm8sgxwSZofc8m-2n31yjAXzI_UKA7wOf&si=KMqoxvsrq-z73iNA

BootsWatch Flatly :
https://bootswatch.com/flatly/

For designing PHP code:
https://www.phptutorial.net/
https://www.w3schools.com/php/