Contents

ACKNOWLEDGEMENT

We would like to express our deepest appreciation to all those who provided us the opportunity to complete this graduation project. A special thanks to our supervisor **[Alshaimaa Mostafa]**, who guided us throughout the project with patience, expertise, and constant encouragement. His/her insights and feedback were invaluable in shaping the direction and development of our system.

We also extend our gratitude to the faculty members of the **[Computer Science Department]**, whose support and mentors' hip were crucial during the research and implementation phases. We are thankful to the students and instructors who participated in our surveys and testing sessions, providing real-world feedback that significantly enhanced the practicality of the platform.

Lastly, we sincerely thank our families and friends for their continued moral support, motivation, and understanding throughout the challenging phases of this project.

ABSTRACT

In recent years, the rapid shift towards digital learning environments has highlighted the importance of effective communication tools and content delivery systems between instructors and students. Traditional learning management systems (LMSs) often suffer from being overly complex, non-intuitive, or lacking in real-time interaction features, especially in local academic contexts.

This project presents the design and development of a web-based educational platform that streamlines the process of course material delivery, assignment submission, and result management. The system allows instructors to easily upload lectures and assignments for each subject, while students can access these materials, submit their solutions, and receive automated or manual feedback.

The platform is designed with usability, scalability, and academic efficiency in mind. It includes features such as role-based dashboards, notification systems, secure login, and structured data flow to enhance user experience. Our solution aims to bridge the gap between instructors and students, enabling a more productive and organized academic process.

Chapter 1: Introduction

In recent years, educational systems around the world have undergone a digital transformation, driven by the increasing need for flexible, accessible, and efficient learning environments. The rapid growth of internet access and mobile technologies has enabled new models of interaction between instructors and students, far beyond the boundaries of traditional classrooms.

However, despite this technological evolution, many institutions—particularly in developing regions—continue to rely on fragmented solutions or outdated manual processes when it comes to sharing lecture materials, managing assignments, and delivering timely feedback to students. These inefficiencies create unnecessary stress for both educators and learners and often result in decreased academic performance.

The COVID-19 pandemic further exposed the weaknesses in existing educational infrastructure and highlighted the need for platforms that can support hybrid or fully online learning. While some institutions adapted quickly using popular tools like Google Classroom or Microsoft Teams, others struggled due to the complexity or cost of such systems.

This project addresses these issues by proposing a dedicated, user-friendly platform tailored for academic environments. The system enables instructors to upload lectures and assignments for each course, and allows students to access the content, submit their work, and instantly receive results. The goal is to simplify academic communication, reduce manual work, and improve the overall learning experience.

This chapter provides a comprehensive overview of the project, including the problem it aims to solve, the objectives behind its development, the landscape of competing solutions, and the motivation and significance driving the project forward.

1.1 Problem Definition

In many universities and colleges, the process of delivering lectures and managing assignments is still inefficient, inconsistent, and prone to error. The most common issues include:

- Students struggling to find course materials distributed through various channels (email, Whats-app, LMS, etc.).
- Assignments being submitted via multiple methods, such as email or hand delivery, making it hard for instructors to track and organize submissions.
- Feedback and grades being delayed or shared informally, without a structured system to record performance history.
- Lack of centralized communication leading to missed deadlines, confusion, or lack of accountability.

From the instructors' point of view, the problems are just as frustrating:

- Difficulty in managing materials and student submissions across different platforms.
- Increased administrative work in collecting, grading, and returning assignments.
- No clear way to monitor student progress across multiple tasks.

These problems collectively limit the effectiveness of teaching and learning. A solution is needed to streamline the academic process and ensure that both students and instructors can focus on education, not logistics.

1.2 Aim of the Project

The primary aim of this project is to develop a centralized academic management platform that enhances the interaction between students and instructors. The platform is designed to:

- Allow instructors to upload and organize lecture materials by subject.
- Enable instructors to assign homework or quizzes with clear deadlines.
- Allow students to view lectures and submit their answers or files through the platform.
- Support both automatic and manual grading mechanisms.
- Deliver immediate feedback and results to students.
- Send notifications to users regarding upcoming deadlines or new content.
- Provide a user-friendly, responsive interface accessible via desktop or mobile devices.

Ultimately, the system will reduce workload, improve organization, and support a more productive academic experience.

1.3 Competitors

Several well-established platforms currently serve as educational tools or learning management systems (LMS), including:

- **Google Classroom:** Offers assignment submission, grading, and communication, but lacks customization options and may not scale well for complex university systems.
- **Moodle:** An open-source LMS with powerful features but often considered difficult to use for non-technical users.
- **Blackboard:** Popular in large universities, but expensive and complex to integrate.
- **Edmodo:** More suited for school-level education, with a focus on social interaction rather than structured academic management.

While these platforms provide important features, they may not address the specific needs of local institutions, especially those looking for a simpler and more cost-effective solution. Our system aims to fill this gap with a focused set of features tailored for university-level workflows, particularly in regions where resources or technical expertise are limited.

1.4 Motivation

The motivation behind this project arises from firsthand experience as students navigating disorganized academic systems. Many students miss assignment deadlines due to lack of clear communication or are unable to access lectures when needed. On the other hand, instructors often spend unnecessary time managing tasks that could easily be automated.

By observing these recurring problems, we recognized the need for a lightweight, adaptable solution. A platform that not only meets technical requirements but also understands the practical, day-to-day challenges faced by both students and staff. This project is our effort to contribute a meaningful, impactful tool to improve the learning experience in our educational community.

1.5 Significance of the Project

This project holds significance in both academic and practical dimensions:

- **Academic Value:** It provides a hands-on opportunity to apply software engineering principles such as system analysis, database design, and user interface development to a real-world problem.
- **Social Impact:** By improving communication and reducing academic confusion, the platform can enhance educational outcomes for hundreds or even thousands of students.

- **Scalability:** The system is designed to be scalable and customizable, allowing it to grow and adapt as institutional needs evolve.
- **Local Relevance:** Unlike many imported LMS solutions, this platform is designed with local academic structures and user behaviour in mind, making it more accessible and user-friendly for our target audience.

In short, this project offers a solution that is not only technically sound but also contextually appropriate for the educational environment it serves.

Chapter 2: System Analysis

2.1 Introduction

This chapter presents the analysis of our system (Web Platform): how we collect and organize the requirements, the functional and non-functional specifications, the system workflow, and the tools we used to design and understand the system. This includes the Gantt chart for project timeline, Block Diagram to represent the system's components, Data Flow Diagram (DFD) to show how data moves through the system, and the Use Case Diagram to demonstrate user interactions with the system.

2.2 Application Requirements

Platform: Web-based application
Supported Browsers: Google Chrome, Mozilla Firefox, Microsoft Edge (latest versions)
Operating System: Compatible with Windows, macOS, and Linux
Internet Connection: Required for all functionalities
User Roles:

- **Student (User):** Can view and download lectures and tasks, submit completed tasks.
- **Instructor:** Can upload lectures and tasks, view submissions, and evaluate students.
- **Admin:** Manages users, monitors platform activity.

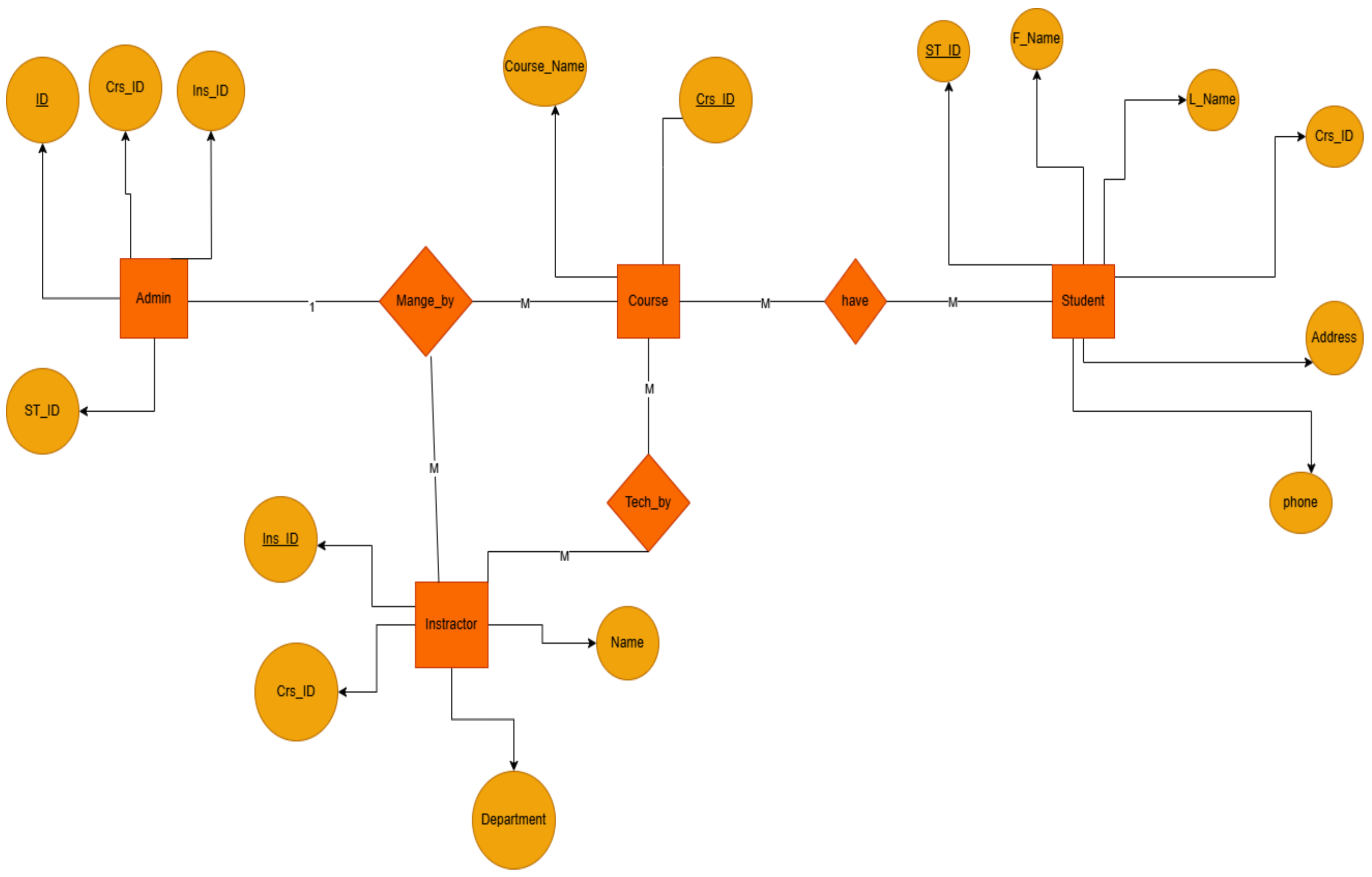
2.3 Development Tool

Used to	Tools
Framework	.NET MVC
Database	SQL Server
Frontend	HTML, CSS, JavaScript, Bootstrap
Backend	C#
Authentication	Identity Framework

2.4 Entity Relationship Diagram (ERD)

ER diagrams are used to design the database schema of the web platform, representing the structure of the system's data and the relationships between different entities such as Students, Instructors, Tasks, and Lectures. The ERD helps in visualizing how the system will manage, store, and connect data.

By analyzing the data requirements of the platform, we designed an ER diagram that reflects real-world interactions: students upload their task submissions, instructors evaluate them, and lectures are assigned to specific courses. This diagram serves as a blueprint for creating the relational database and implementing the logic necessary for data access and manipulation. Overall, the ER diagram is a crucial step in understanding and organizing the platform's data flow, ensuring that the database is both efficient and scalable.

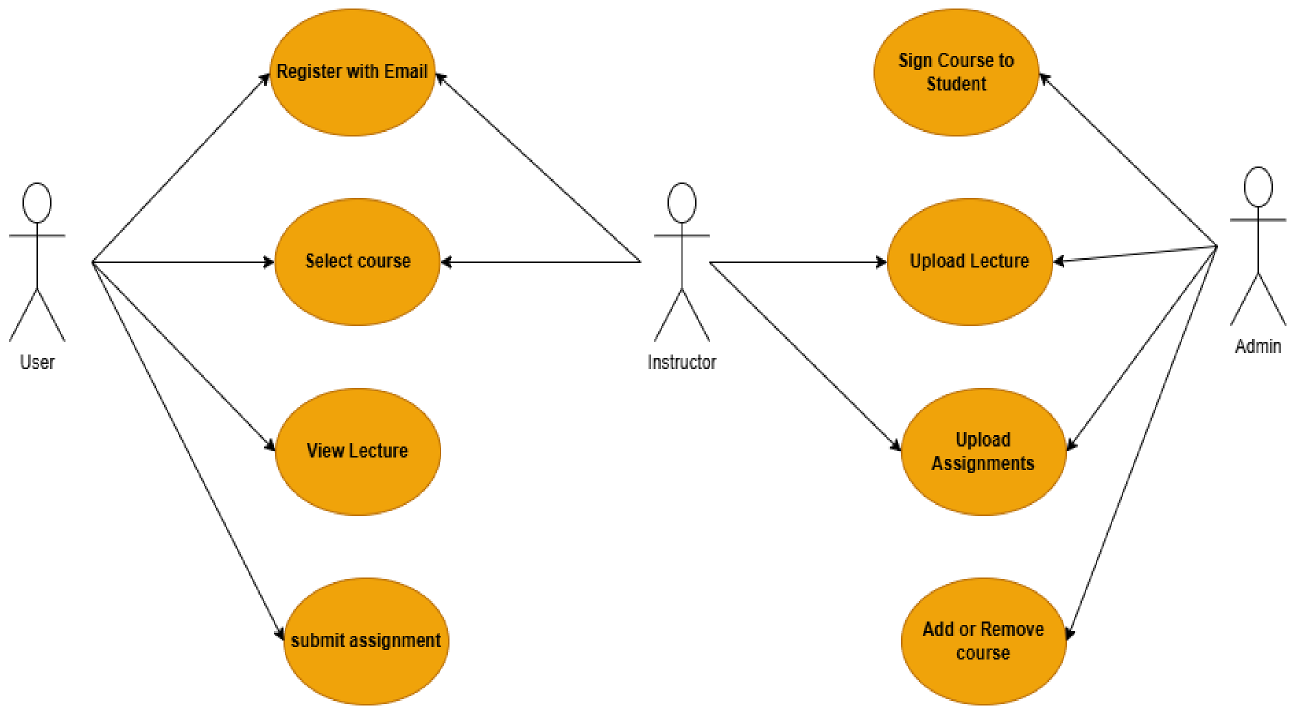


2.5 Use Case Diagram

A use case diagram is a fundamental part of system analysis that illustrates how different users interact with the system. In the context of this project, which aims to create a web-based platform for communication between instructors and students, the use case diagram helps visualize the core functionalities that the system provides for each type of user.

The diagram identifies the **actors** involved—primarily the **Instructor,** the **Student** and **Admin**—and the **use cases** such as uploading lectures, creating assignments, viewing lectures, and submitting tasks. This graphical representation ensures that all user interactions with the system are clearly defined and understood from the beginning of the development process.

The primary goal of the use case diagram in this project is to offer a high-level overview of the system's functional requirements. It serves as a tool for both communication and validation, allowing stakeholders to confirm that all necessary features are addressed. Additionally, it helps identify any gaps in functionality or potential conflicts early in the design phase.



2.6 DFD Diagram

A Data Flow Diagram (DFD) is a graphical tool that illustrates how data moves through the educational web platform designed to connect instructors and students. It shows the flow of information from external entities such as students and instructors into the system, how this data is processed internally, and how the results are presented back to the users.

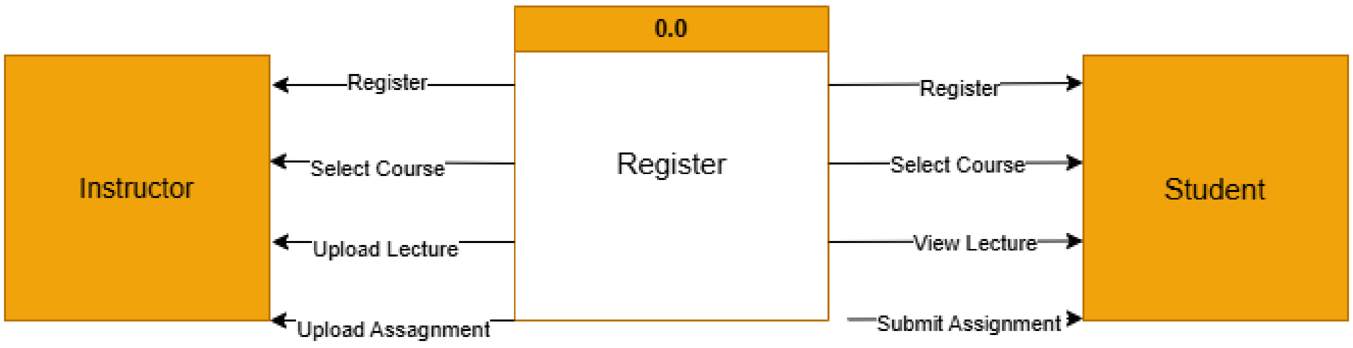
In our project, the DFD highlights the interaction between different system components:

- **External Entities** include the **Instructor** and the **Student**.

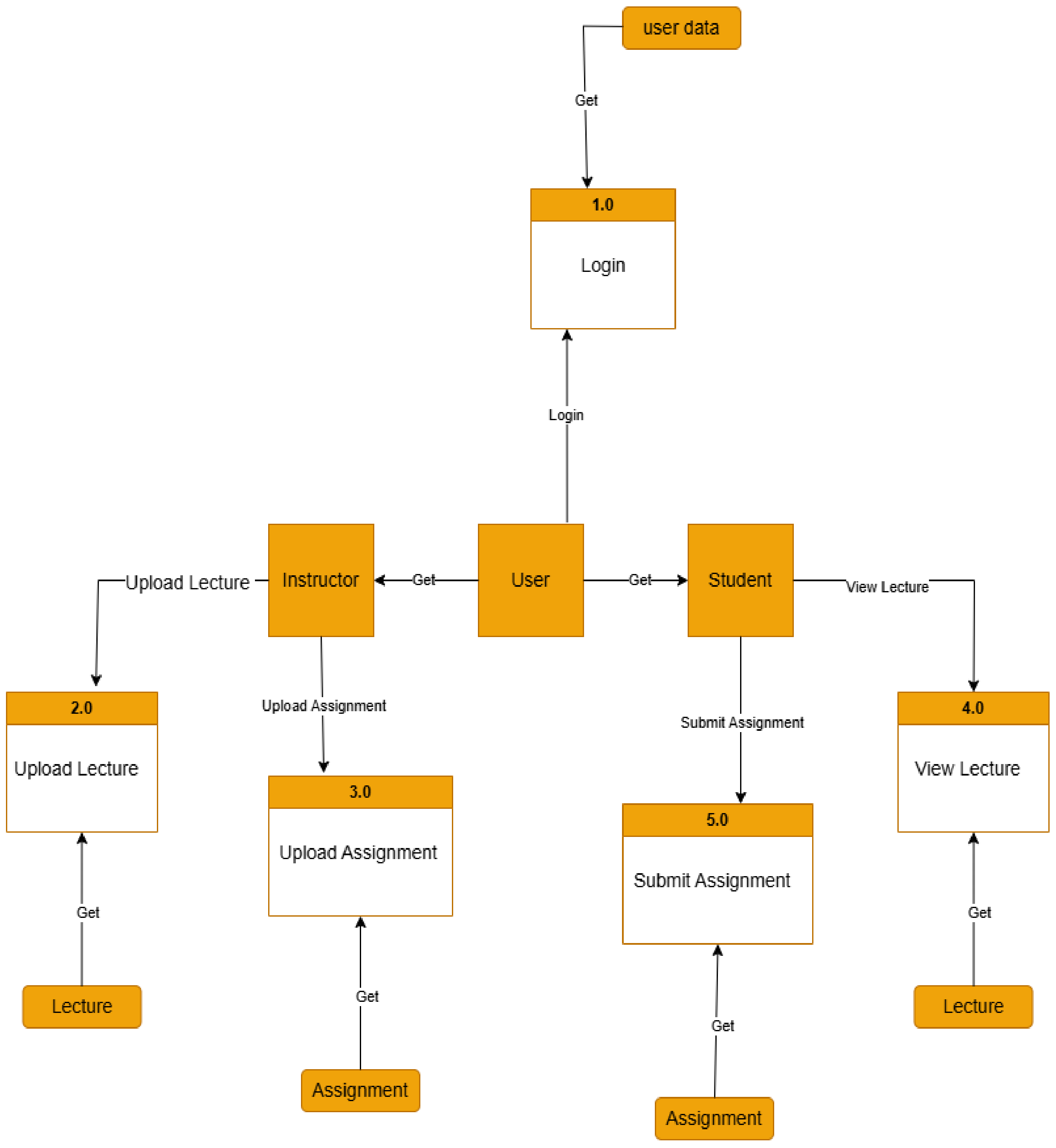
- **Processes** involve actions such as registration, login, course selection, lecture uploading, assignment submission, and viewing task status.
- **Data Flows** show how data moves between users and the system — for example, when a student uploads an assignment or when an instructor checks who submitted tasks.
- **Data Stores** represent saved content like user accounts, uploaded lectures, assignments, and student submissions.

This diagram helps clarify the overall system behavior and data interactions, making it easier to understand how information is managed and ensuring all functional requirements are met. The DFD is valuable during system analysis, design, testing, and implementation phases to ensure a smooth and efficient data structure.

2.6.1 DFD level zero



2.6.2 DFD level 1

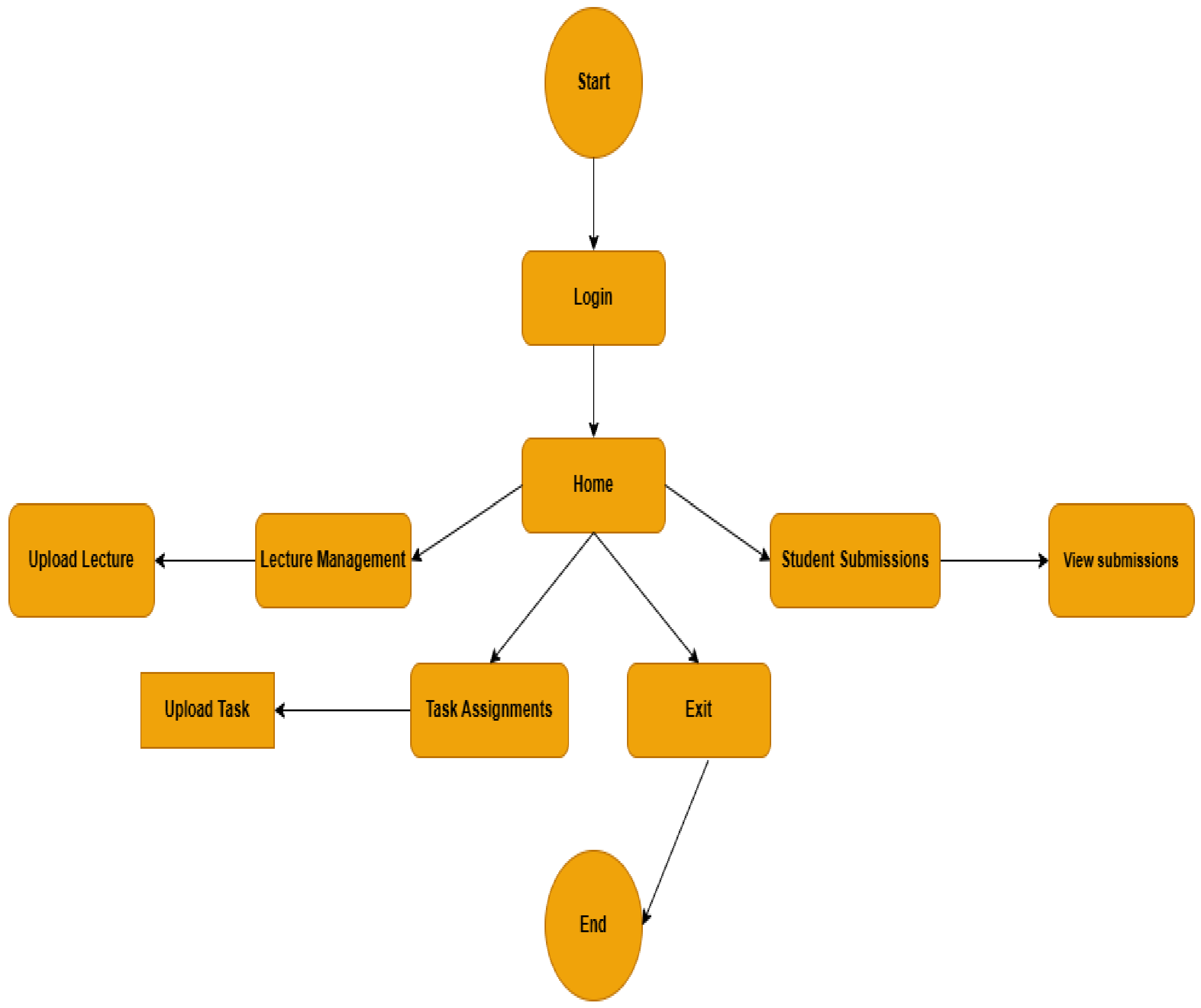


2.7 Flow Chart

A flowchart is a graphical representation used to illustrate the sequence of operations within the proposed educational platform. It is a valuable tool for visualizing the steps followed by users — including both students and instructors — while interacting with the system. Flowcharts use standardized symbols such as rectangles to represent processes, diamonds for decision points, and arrows to indicate the direction of data or process flow.

In the context of this project, the flowchart demonstrates key interactions such as user login, course selection, lecture uploads by instructors, assignment submissions by students, and feedback or evaluation processes. It helps identify the order of activities and highlights any possible bottlenecks or inefficiencies within the system.

Flowcharts play an essential role in **system analysis and design**, as they are used during requirements gathering, development, testing, and implementation stages. By providing a clear and concise overview of system processes, the flowchart facilitates better understanding among developers, stakeholders, and end-users. It also serves as a useful documentation and communication tool that enhances the effectiveness and accuracy of the software development process.


Start
Login
Home
Lecture Management
Upload Lecture
Student Submissions
View submissions
Task Assignments
Upload Task
Exit
End

Chapter 3: System Design

The System Design chapter in the documentation of EduBridge focuses on the design aspects of the system. It builds upon the analysis conducted in the previous chapter and translates the requirements into a well-defined and efficient system design. The key sections covered in this chapter are:

1- **System Architecture Design**

2- **Data Design**

3- **User Interface Design**

4- **Database Design**

3.1 System Architecture Design

The system architecture of **EduBridge** is designed to ensure scalability, modularity, and maintainability. It follows an API architecture consisting of server side and client side.

3.1.1 Class Diagram

The Class Diagram is a graphical representation used in object-oriented programming to illustrate the structure and behavior of a system or application. It showcases the classes, objects, and their relationships, providing a visual depiction of how they interact with each other.

Class diagrams are widely utilized in software engineering as a communication tool to convey system design to developers, stakeholders, and team members. They include class names, attributes, methods, and relationships such as association, inheritance, composition, and aggregation. Class diagrams play a crucial role in documenting and understanding the design of a system concisely and intuitively.

Authentication s User Management System

User Roles: The application defines different user roles

Admin: Responsible for managing user accounts, granting permissions, and overseeing the entire system.

Regular Users: Access the application with limited privileges.

Registration C Login: Only the admin can register new users.

Users log in using their credentials (username and password).

Upon successful login, the system generates an **access token** valid for **14 days**.

Password Constraints: The application enforces password policies (e.g., minimum length, complexity) to enhance security

Main View s Navigation System

EduBridge database management system Search Functionality, Users can search for stock items based on various parameters:

▪ **Name**: Search by item name.

▪ **Username**: Filter by the user who manages the stock.

▪ **Location Name**: Find items stored in specific locations.

▪ **Category Name**: Categorize items (e.g., electronics, office supplies).

▪ **Device ID**: Locate items by their unique identifiers.

User-Friendly Interface: The main view provides an intuitive interface for users to navigate through the application.

Clear menus and organized sections enhance usability

Backup s Logging System

Database Backup: Regular backups ensure data integrity and disaster recovery.

Admins can schedule automated backups or trigger them manually.

Logging: The system logs all interactions

Client-Side Requests: User queries, searches, and actions.

Server-Side Responses: System behavior, errors, and updates.

Logs are essential for auditing, troubleshooting, and monitoring system performance.

Admins can access the log files.

Device Specifications

This system manages additional information related to stock items

Device Type: Categorization (e.g., computers, peripherals, lab equipment).

Model: Specific model or version.

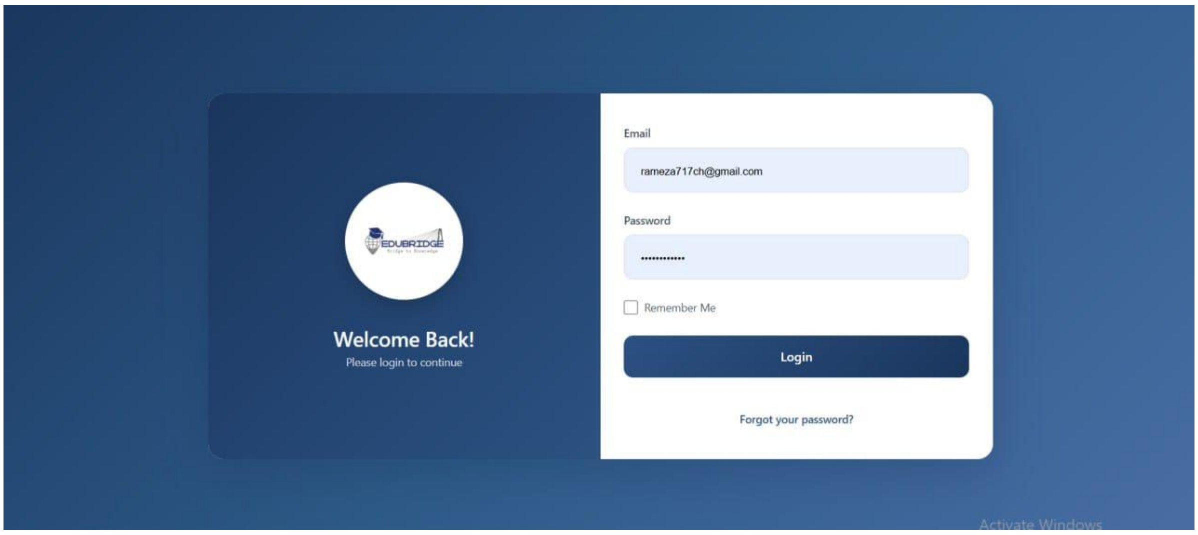
Serial Number: Unique identifier for each item.

Specifications: Technical details (e.g., RAM, storage capacity).

3.2 System interface design

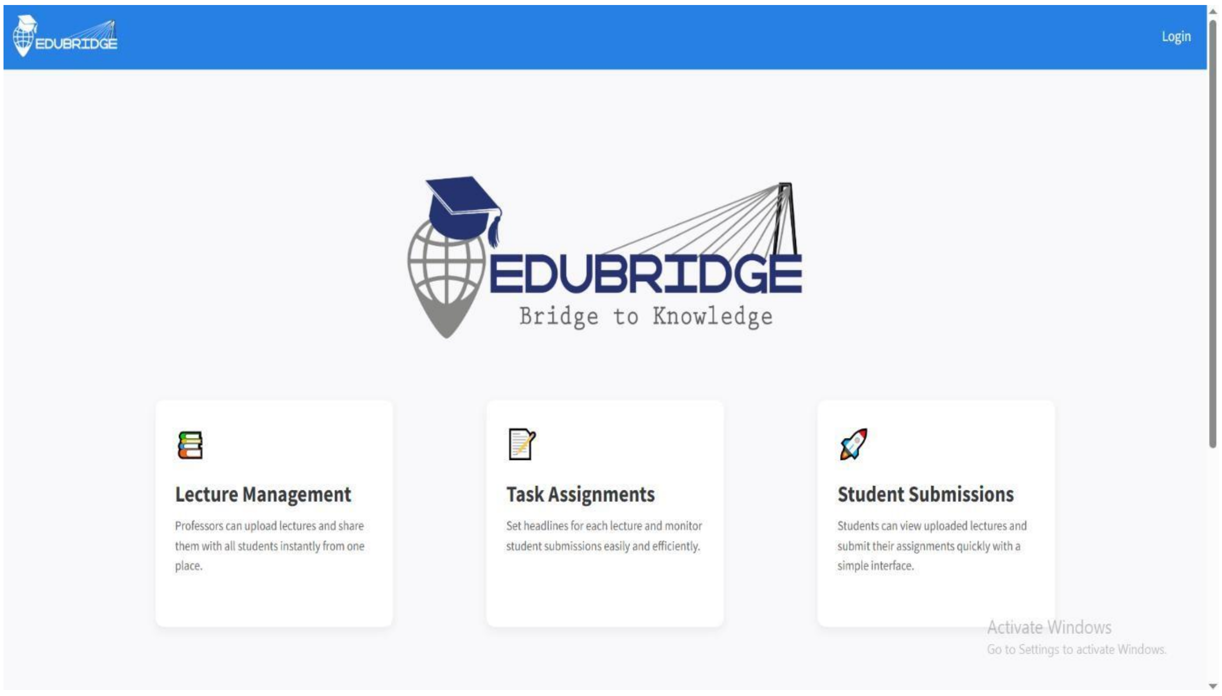
EduBridge application has the following screens:

3.2.1 Login page:



Here you can log in and register your username and password.

3.1.1 Home page:



Login

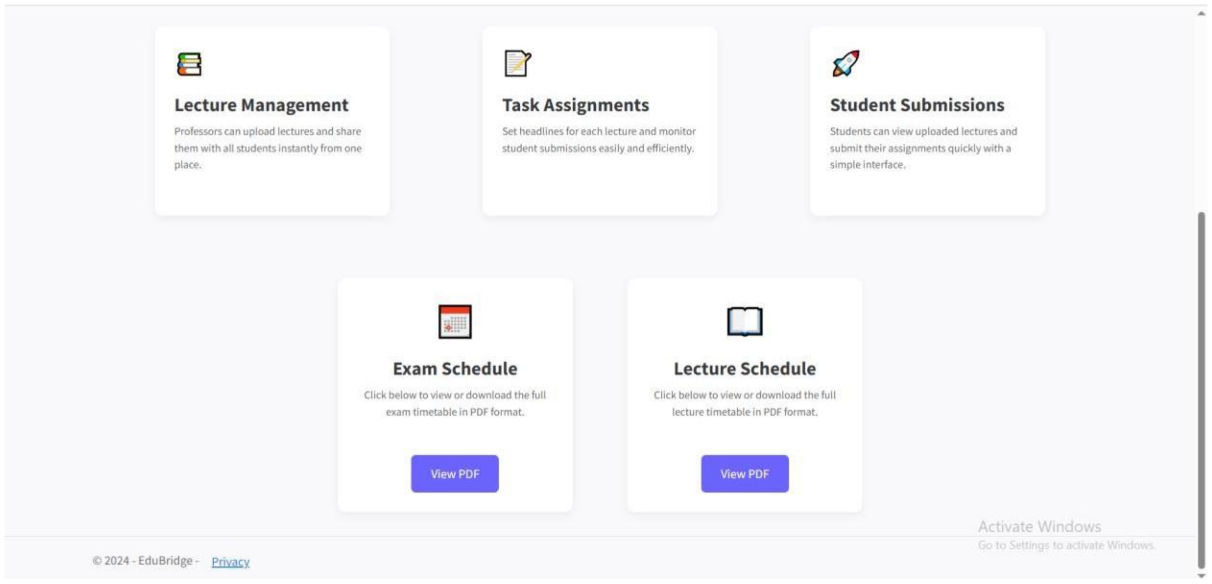
EDUBRIDGE
Bridge to Knowledge

Lecture Management
Professors can upload lectures and share them with all students instantly from one place.

Task Assignments
Set headlines for each lecture and monitor student submissions easily and efficiently.

Student Submissions
Students can view uploaded lectures and submit their assignments quickly with a simple interface.

Activate Windows
Go to Settings to activate Windows.



Lecture Management
Professors can upload lectures and share them with all students instantly from one place.

Task Assignments
Set headlines for each lecture and monitor student submissions easily and efficiently.

Student Submissions
Students can view uploaded lectures and submit their assignments quickly with a simple interface.

Exam Schedule
Click below to view or download the full exam timetable in PDF format.
View PDF

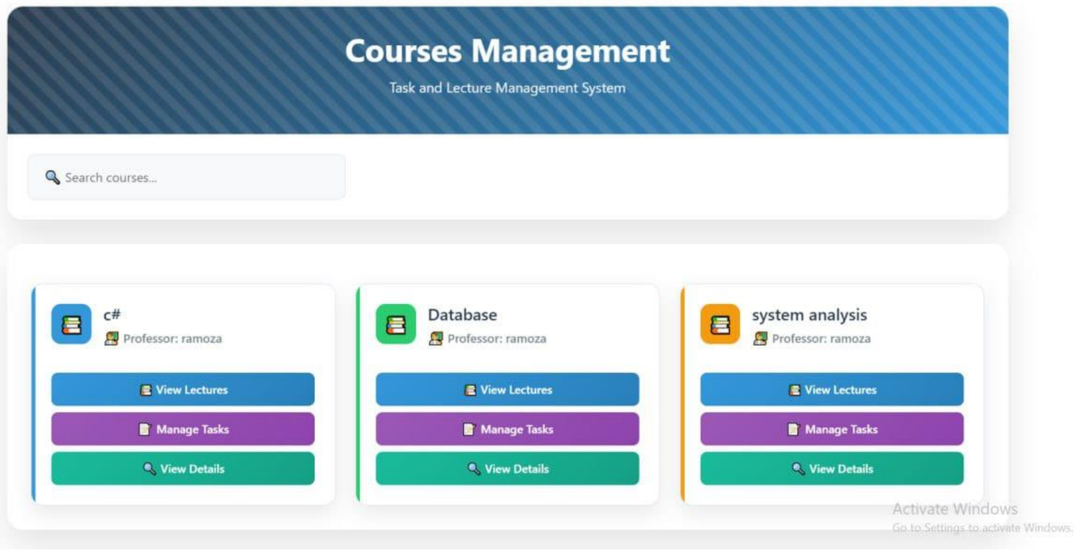
Lecture Schedule
Click below to view or download the full lecture timetable in PDF format.
View PDF

Activate Windows
Go to Settings to activate Windows.

© 2024 - EduBridge - Privacy

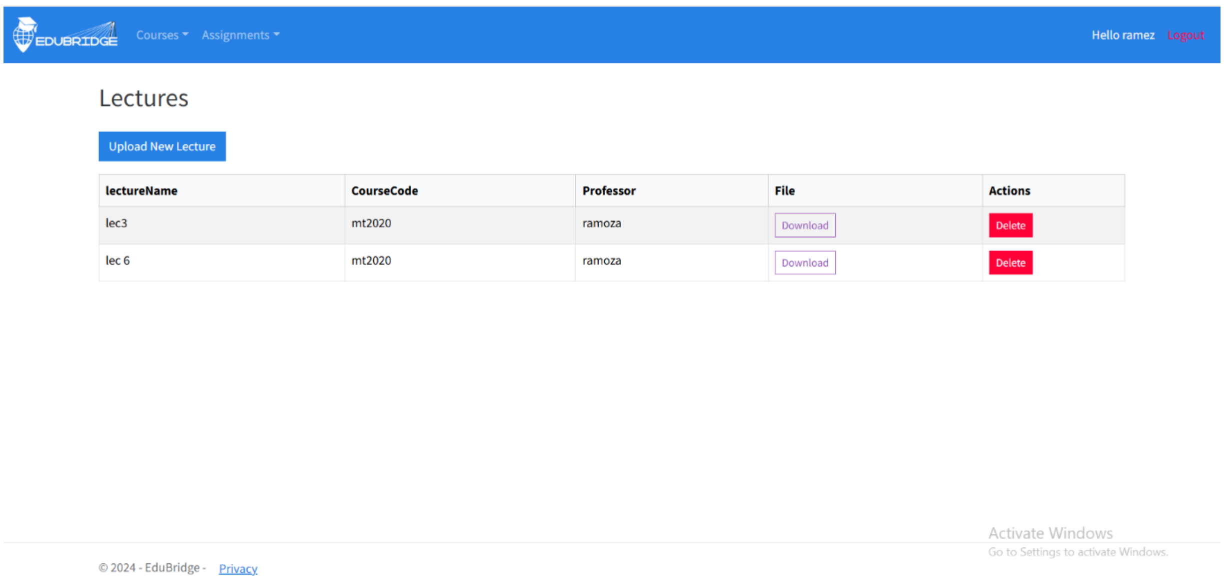
Here is the home page, which displays lectures, tasks, and all available options.

3.1.2 Doctor Courses page:



Here is the courses page, which contains the course lectures, tasks, and course specifications.

3.1.3 Doctor Lectures on specific course page:



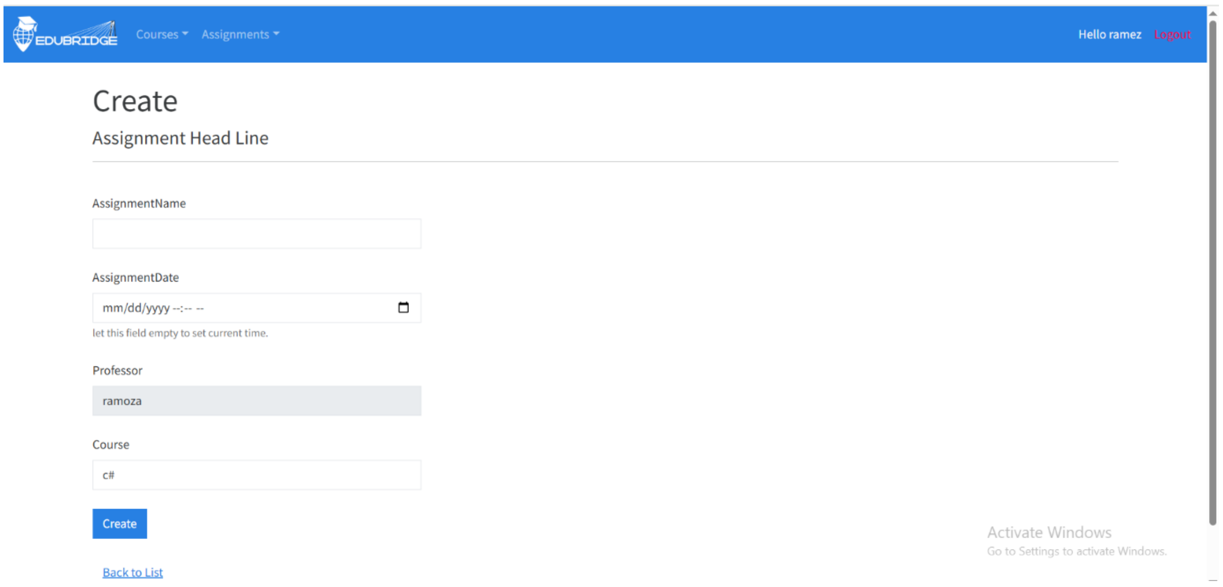
Lectures

lectureName	CourseCode	Professor	File	Actions
lec3	mt2020	ramoza	Download	Delete
lec 6	mt2020	ramoza	Download	Delete

© 2024 - EduBridge - Privacy

Lectures of the courses that have been uploaded are displayed here.

3.1.4 Task Upload page:



Create

Assignment Head Line

AssignmentName

AssignmentDate

mm/dd/yyyy --:-- --

let this field empty to set current time.

Professor

ramoza

Course

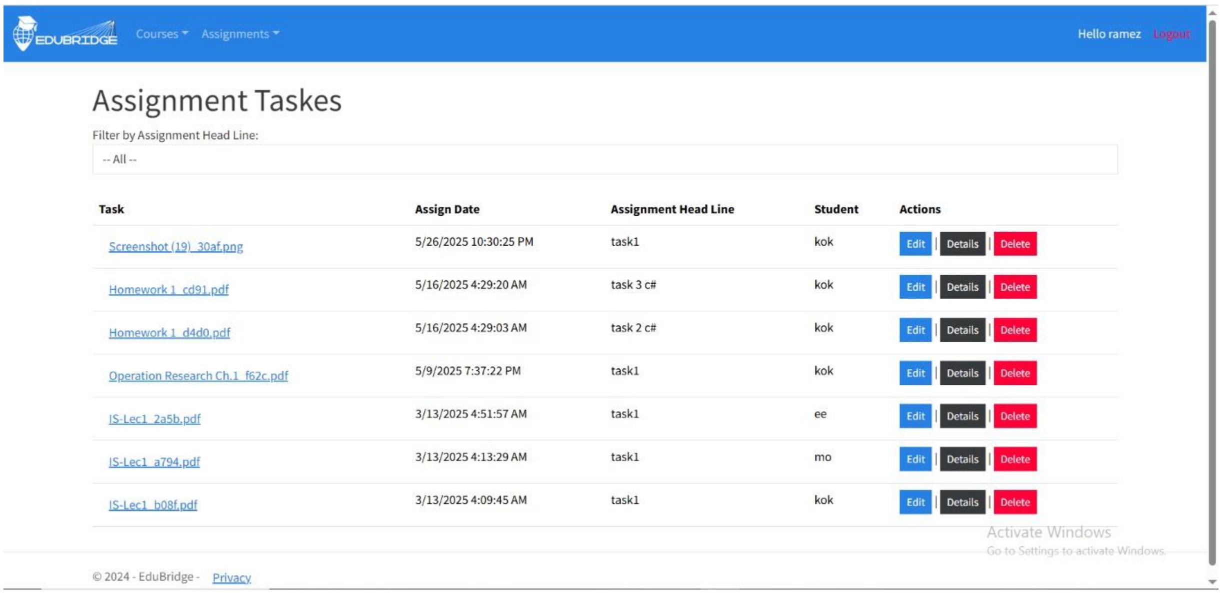
c#

Create

Back to List

Here he uploads the task he did for the course he chose.

3.1.5 Assignment Tasks page:



Assignment Taskes

Filter by Assignment Head Line:

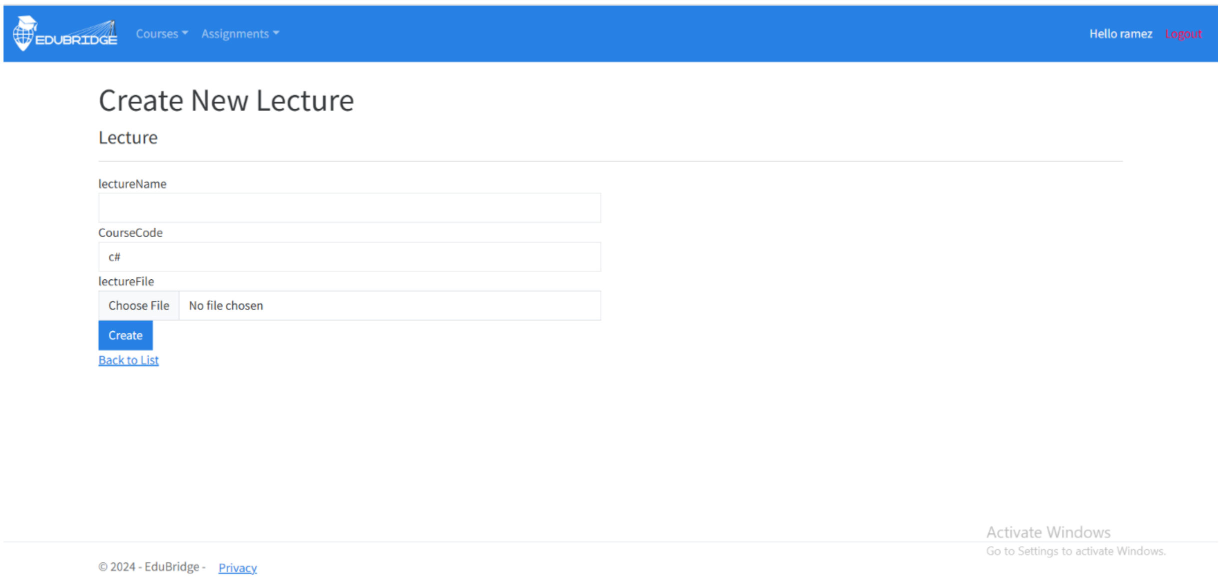
-- All --

Task	Assign Date	Assignment Head Line	Student	Actions
Screenshot (19)_30af.png	5/26/2025 10:30:25 PM	task1	kok	Edit | Details | Delete
Homework 1_cd91.pdf	5/16/2025 4:29:20 AM	task 3 c#	kok	Edit | Details | Delete
Homework 1_d4d0.pdf	5/16/2025 4:29:03 AM	task 2 c#	kok	Edit | Details | Delete
Operation Research Ch.1_f62c.pdf	5/9/2025 7:37:22 PM	task1	kok	Edit | Details | Delete
IS-Lec1_2a5b.pdf	3/13/2025 4:51:57 AM	task1	ee	Edit | Details | Delete
IS-Lec1_a794.pdf	3/13/2025 4:13:29 AM	task1	mo	Edit | Details | Delete
IS-Lec1_b08f.pdf	3/13/2025 4:09:45 AM	task1	kok	Edit | Details | Delete

© 2024 - EduBridge - Privacy

Here you will find the tasks that were submitted by students in the course he chose.

3.1.6 Uplode new lecture page:



Create New Lecture

Lecture

lectureName

CourseCode

c#

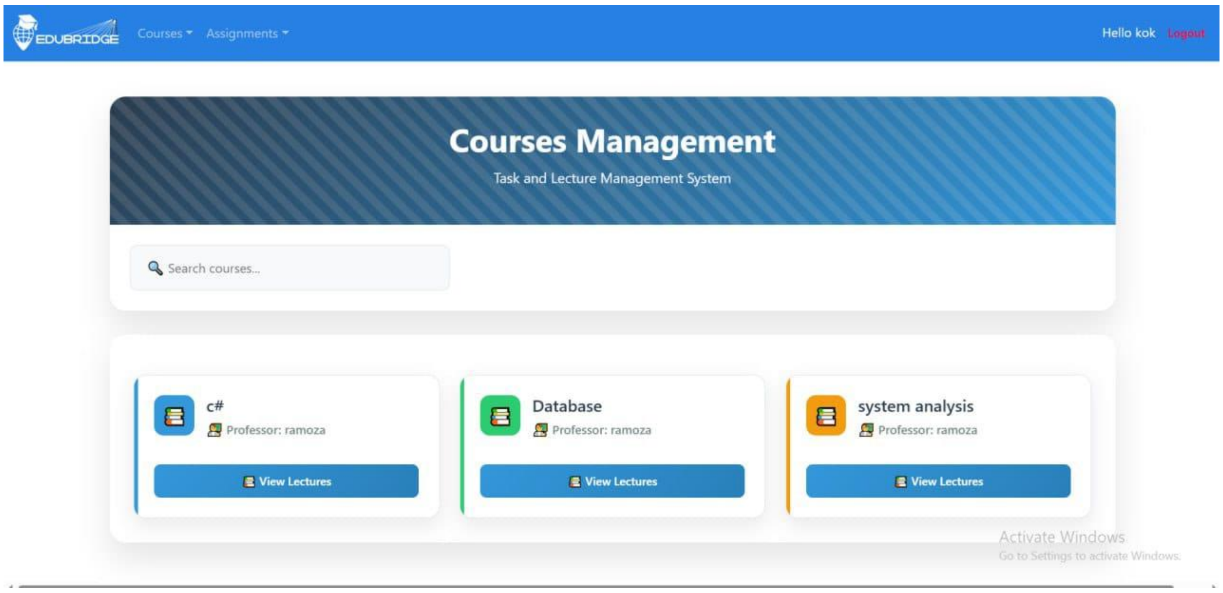
lectureFile

Choose File No file chosen

© 2024 - EduBridge - Privacy

Here he uploads the lecture for a specific course.

3.1.7 Student courses page:



Courses Management

Task and Lecture Management System

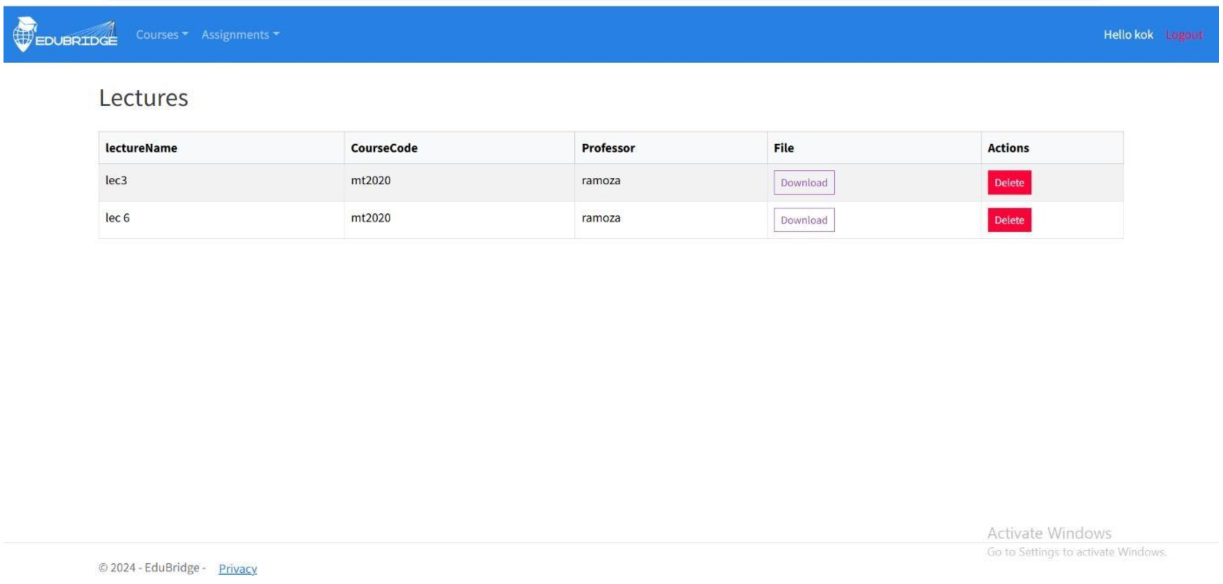
c#
Professor: ramoza

Database
Professor: ramoza

system analysis
Professor: ramoza

The student's registered courses are displayed here.

3.2.G Student Lectures page:



Lectures

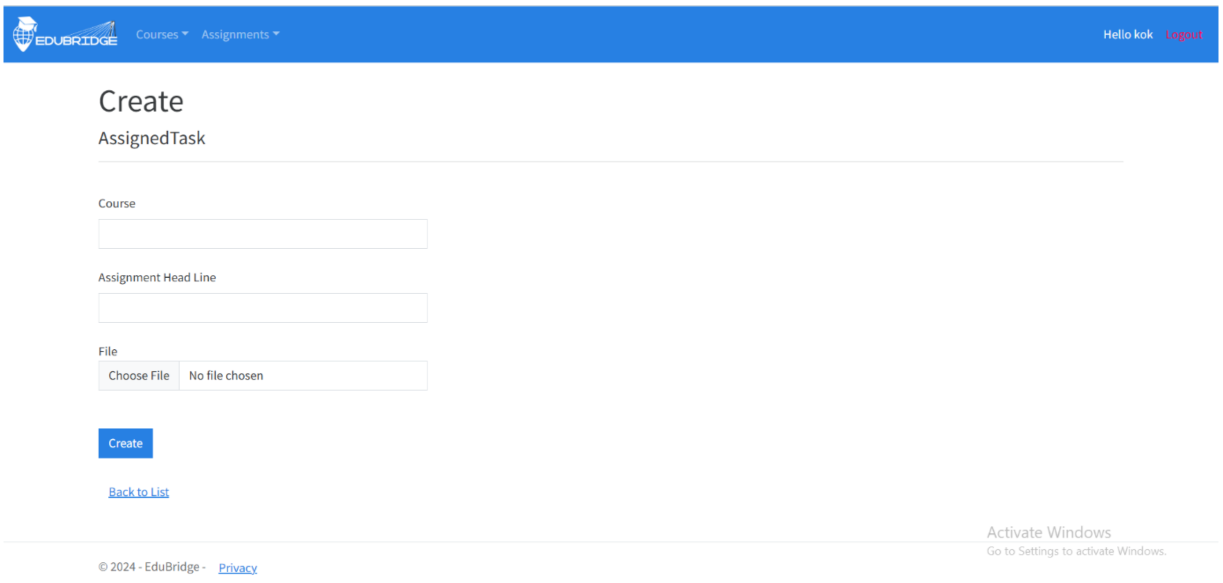
lectureName	CourseCode	Professor	File	Actions
lec3	mt2020	ramoza	Download	Delete
lec 6	mt2020	ramoza	Download	Delete

Activate Windows
Go to Settings to activate Windows.

© 2024 - EduBridge - Privacy

Here you will find lectures for a specific course.

3.2.10 Student Task Solution Uplode page:



Create

AssignedTask

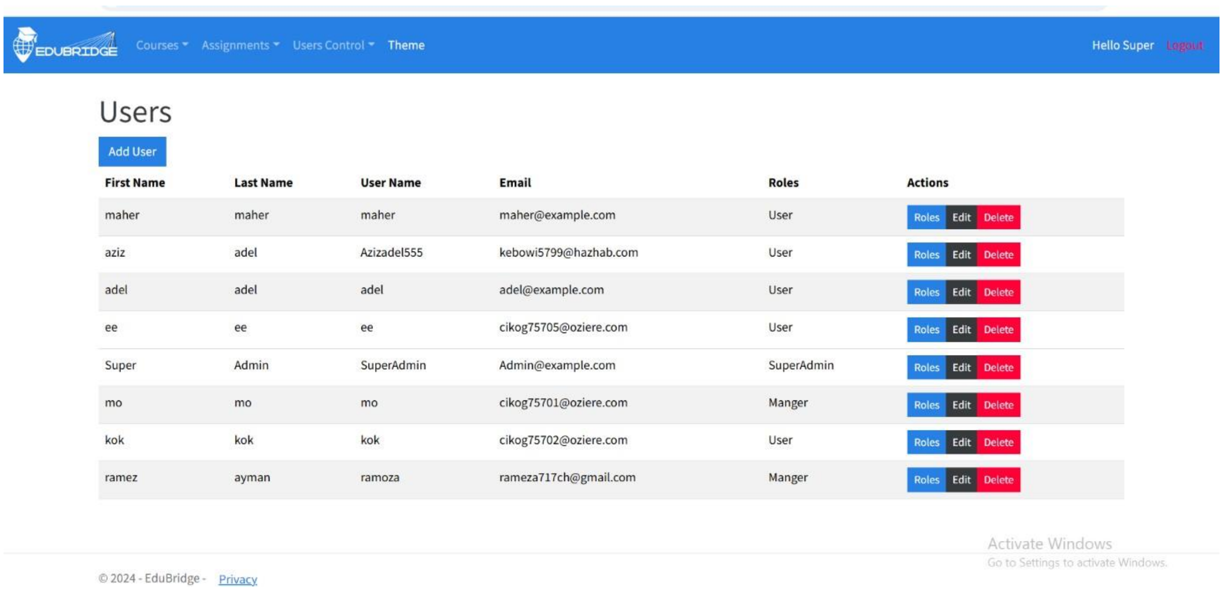
Course	
Assignment Head Line	
File	Choose File No file chosen

Activate Windows
Go to Settings to activate Windows.

© 2024 - EduBridge - Privacy

Here the student uploads his task solution.

3.2.11 Users page:



Users

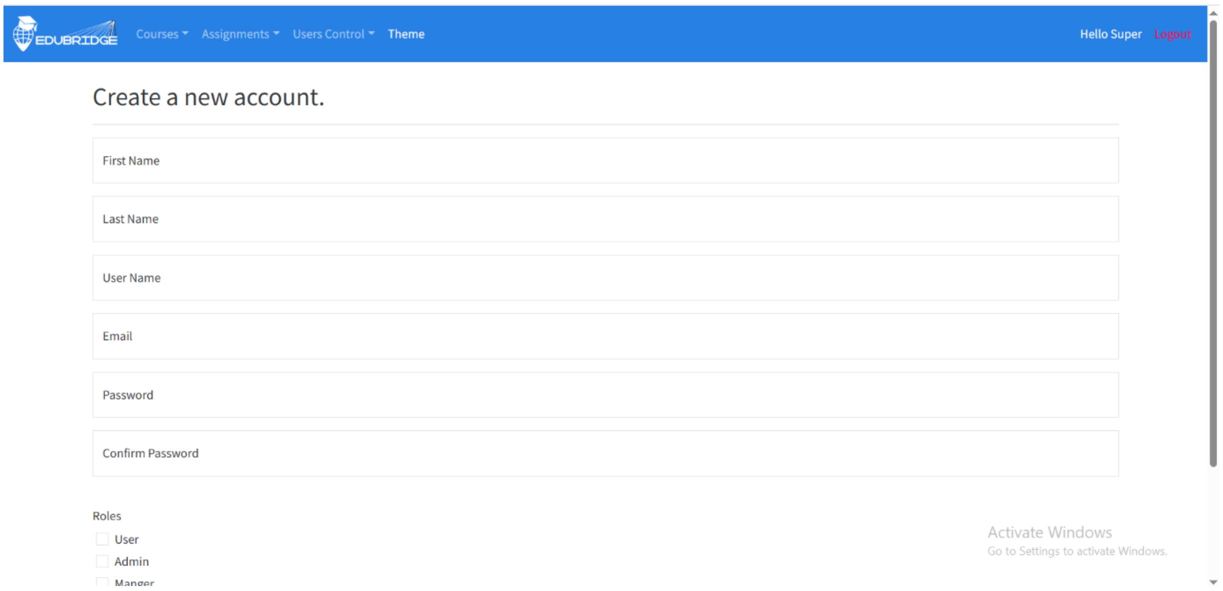
Add User

First Name	Last Name	User Name	Email	Roles	Actions
maher	maher	maher	maher@example.com	User	Roles Edit Delete
aziz	adel	Azizadel555	kebowi5799@hazhab.com	User	Roles Edit Delete
adel	adel	adel	adel@example.com	User	Roles Edit Delete
ee	ee	ee	cikog75705@oziere.com	User	Roles Edit Delete
Super	Admin	SuperAdmin	Admin@example.com	SuperAdmin	Roles Edit Delete
mo	mo	mo	cikog75701@oziere.com	Manger	Roles Edit Delete
kok	kok	kok	cikog75702@oziere.com	User	Roles Edit Delete
ramez	ayman	ramoza	rameza717ch@gmail.com	Manger	Roles Edit Delete

© 2024 - EduBridge - Privacy

Here you will see the names, user name and email of all users who use the site.

3.2.12 (Doctors / Student) Create page:



Create a new account.

First Name

Last Name

User Name

Email

Password

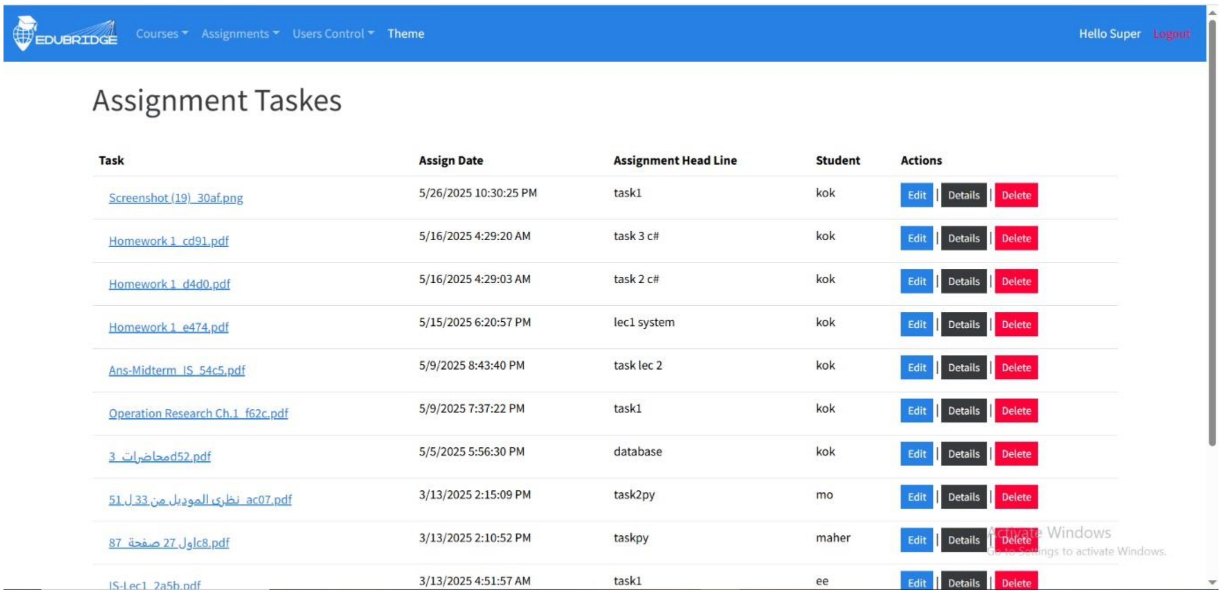
Confirm Password

Roles

- [ ] User
- [ ] Admin
- [ ] Manger

Here we can add a new student or doctor by adding some information, user name and password.

3.2.13 All Tasks page:

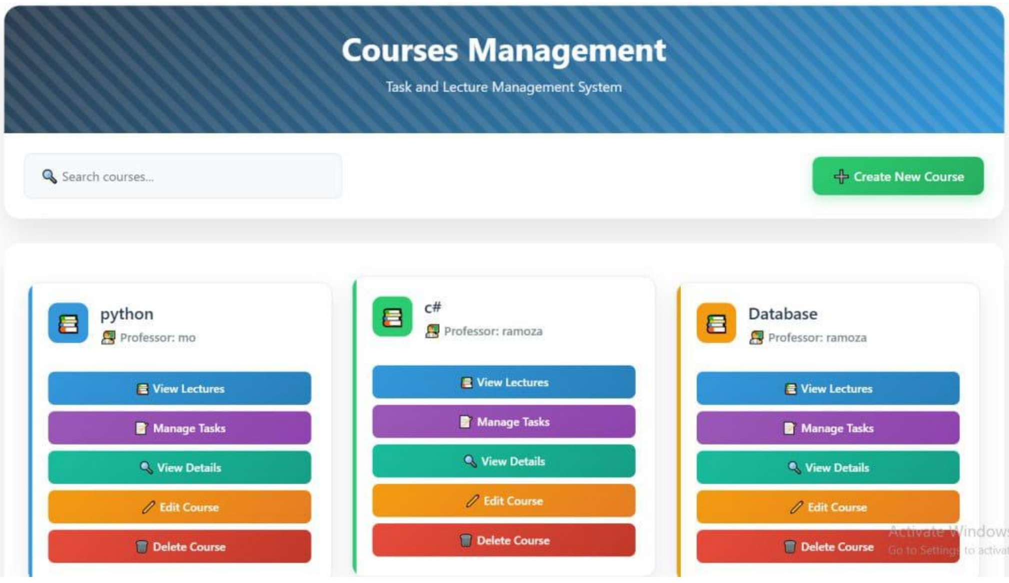


Assignment Taskes

Task	Assign Date	Assignment Head Line	Student	Actions
Screenshot (19)_30af.png	5/26/2025 10:30:25 PM	task1	kok	Edit | Details | Delete
Homework 1_cd91.pdf	5/16/2025 4:29:20 AM	task 3 c#	kok	Edit | Details | Delete
Homework 1_d4d0.pdf	5/16/2025 4:29:03 AM	task 2 c#	kok	Edit | Details | Delete
Homework 1_e474.pdf	5/15/2025 6:20:57 PM	lec1 system	kok	Edit | Details | Delete
Ans-Midterm_IS_54c5.pdf	5/9/2025 8:43:40 PM	task lec 2	kok	Edit | Details | Delete
Operation Research Ch.1_f62c.pdf	5/9/2025 7:37:22 PM	task1	kok	Edit | Details | Delete
3_محاضرات_d52.pdf	5/5/2025 5:56:30 PM	database	kok	Edit | Details | Delete
51 ل 33 نظرى الموديل من_ac07.pdf	3/13/2025 2:15:09 PM	task2py	mo	Edit | Details | Delete
87_صفحة 27 لاول_c8.pdf	3/13/2025 2:10:52 PM	taskpy	maher	Edit | Details | Delete
IS-Lec1_2a5b.pdf	3/13/2025 4:51:57 AM	task1	ee	Edit | Details | Delete

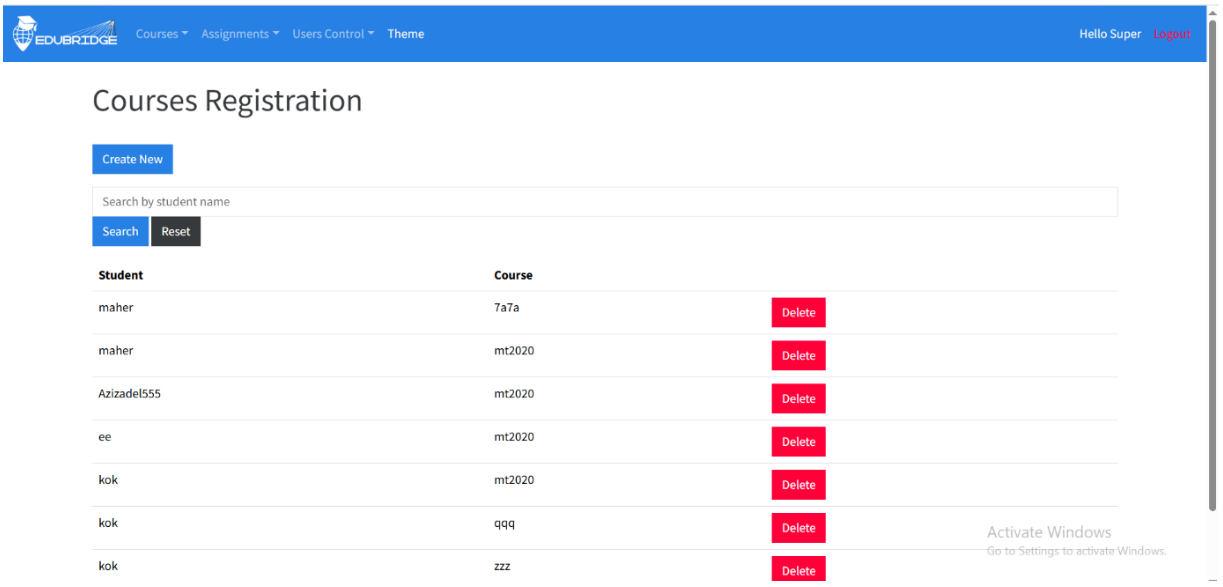
Here you will see all the tasks in the system and when it uploaded by student name.

3.2.14 All Courses page:



Here you will find the courses available in the system by its description.

3.2.15 Students Courses page:



Courses Registration

Create New

Search by student name

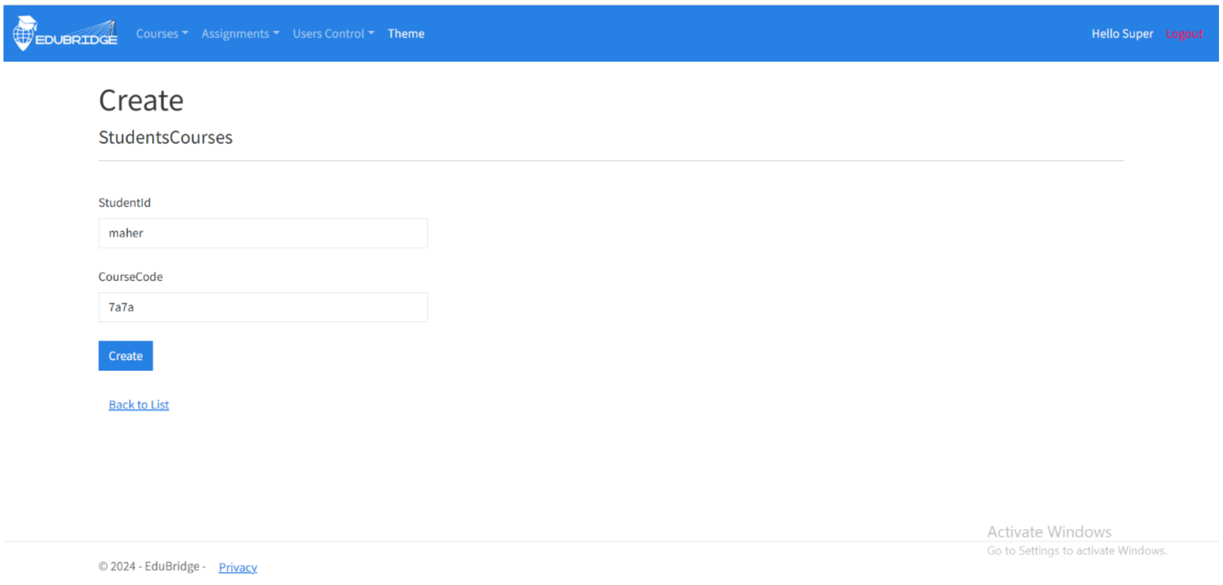
Search Reset

Student	Course	
maher	7a7a	Delete
maher	mt2020	Delete
Azizadel555	mt2020	Delete
ee	mt2020	Delete
kok	mt2020	Delete
kok	qqq	Delete
kok	zzz	Delete

Activate Windows
Go to Settings to activate Windows.

Here you will find the names of the students and the courses they are registered for.

3.2.16 Student Register page:



Create

StudentsCourses

StudentId: maher

CourseCode: 7a7a

Create

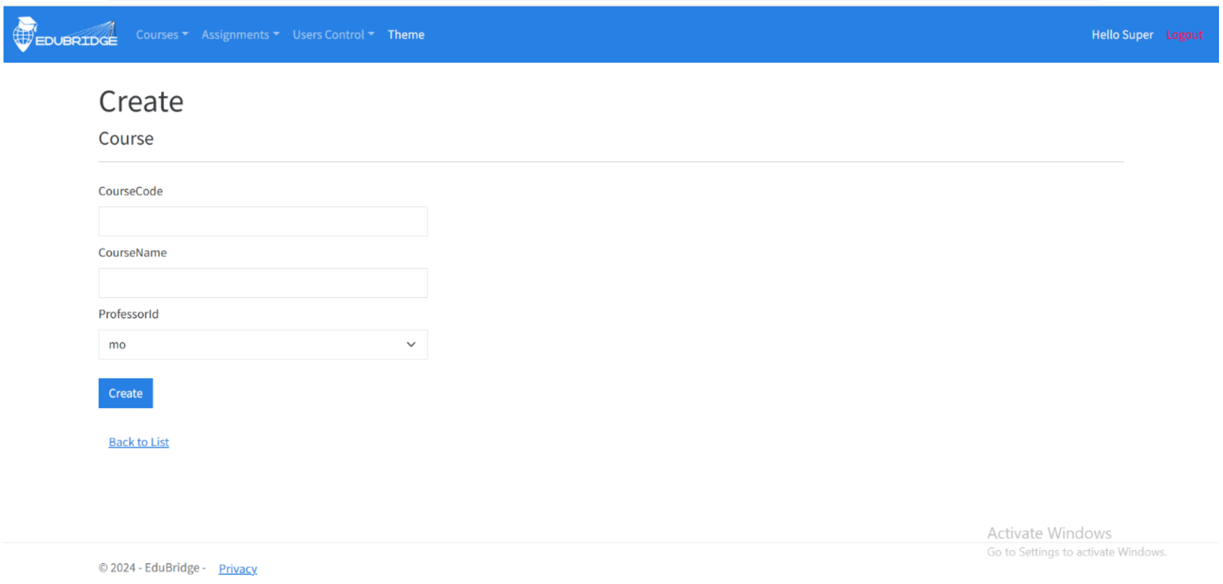
Back to List

Activate Windows
Go to Settings to activate Windows.

© 2024 - EduBridge - Privacy

Here we can register the course for the student by his name and course code.

3.2.17 Course Register page:



Create

Course

CourseCode

CourseName

ProfessorId

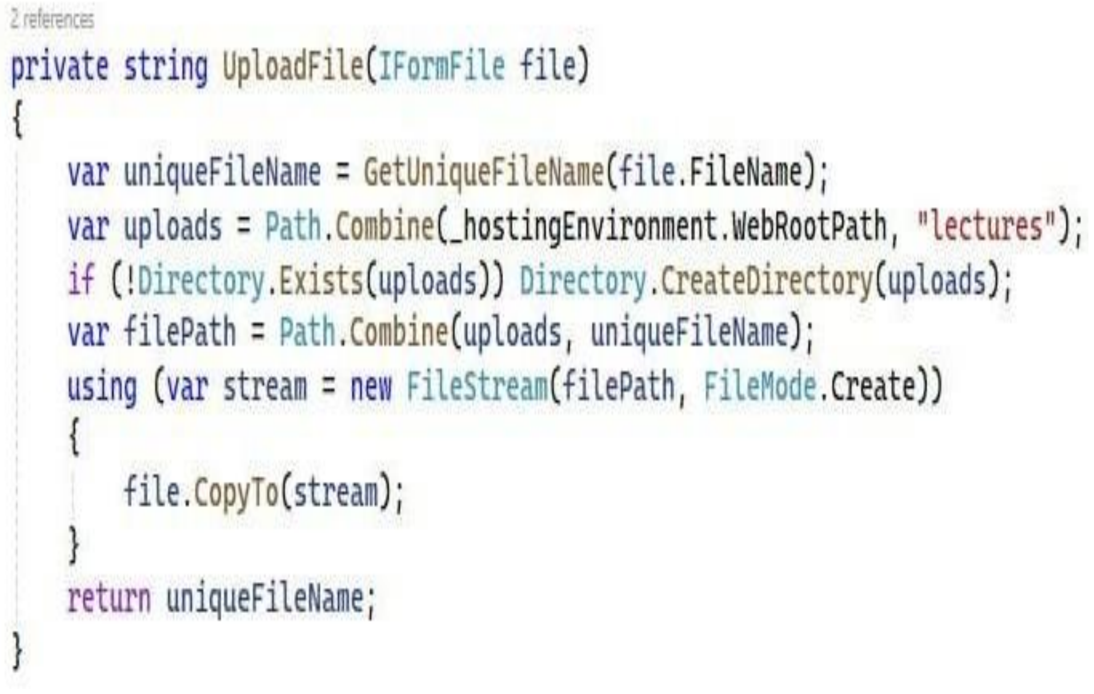
mo

© 2024 - EduBridge - Privacy

Here we can add a new course and who is the doctor who will teach the course.

Chapter 4: System Implementation

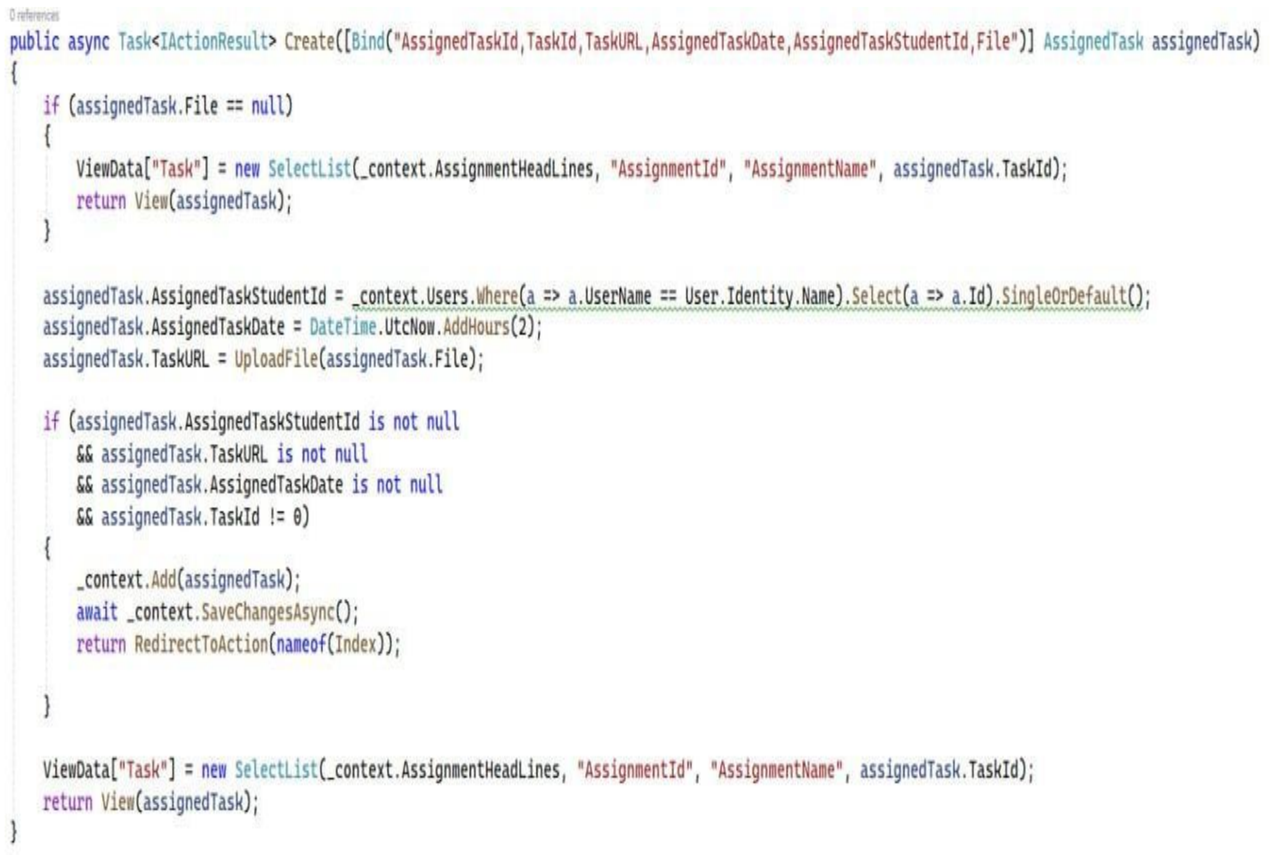
4.1-Upload Lecture Function:



```
2 references
private string UploadFile(IFormFile file)
{
    var uniqueFileName = GetUniqueFileName(file.FileName);
    var uploads = Path.Combine(_hostingEnvironment.WebRootPath, "lectures");
    if (!Directory.Exists(uploads)) Directory.CreateDirectory(uploads);
    var filePath = Path.Combine(uploads, uniqueFileName);
    using (var stream = new FileStream(filePath, FileMode.Create))
    {
        file.CopyTo(stream);
    }
    return uniqueFileName;
}
```

The **Upload-File** method is used to save an uploaded file (received from a web form) to the server in a specific folder called "lectures", located under the **www-root** directory. The method first generates a unique file name to avoid conflicts with existing files. It then checks if the target folder exists, and if not, it creates it. After that, it saves the file using a **File-Stream**, and finally, returns the unique file name for reference (e.g., saving in a database).

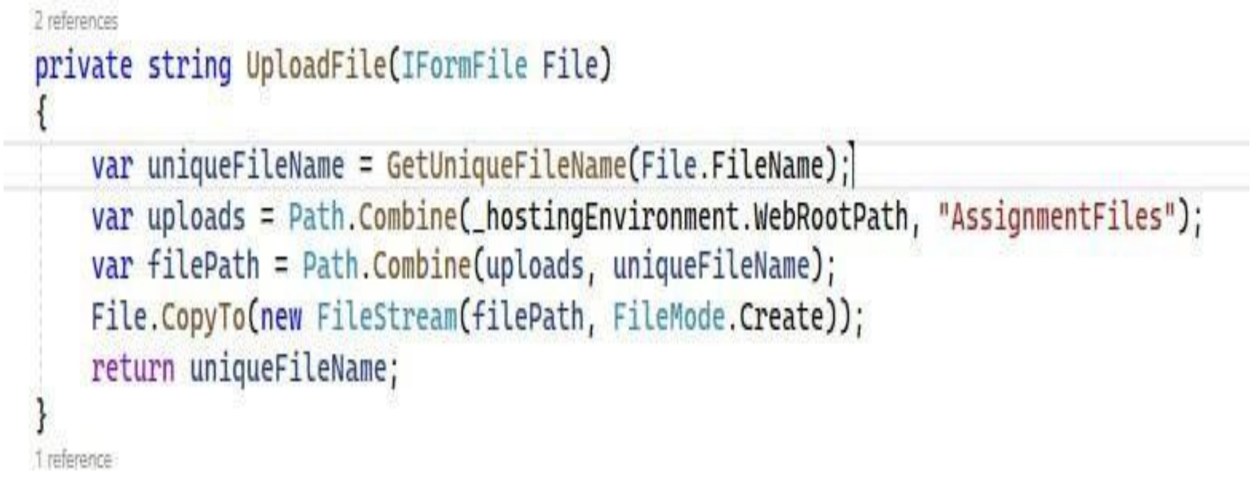
4.2-Upload Task Function:



```
0 references
public async Task<IActionResult> Create([Bind("AssignedTaskId,TaskId,TaskURL,AssignedTaskDate,AssignedTaskStudentId,File")] AssignedTask assignedTask)
{
    if (assignedTask.File == null)
    {
        ViewData["Task"] = new SelectList(_context.AssignmentHeadLines, "AssignmentId", "AssignmentName", assignedTask.TaskId);
        return View(assignedTask);
    }

    assignedTask.AssignedTaskStudentId = _context.Users.Where(a => a.UserName == User.Identity.Name).Select(a => a.Id).SingleOrDefault();
    assignedTask.AssignedTaskDate = DateTime.UtcNow.AddHours(2);
    assignedTask.TaskURL = UploadFile(assignedTask.File);

    if (assignedTask.AssignedTaskStudentId is not null
        && assignedTask.TaskURL is not null
        && assignedTask.AssignedTaskDate is not null
        && assignedTask.TaskId != 0)
    {
        _context.Add(assignedTask);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));

    }

    ViewData["Task"] = new SelectList(_context.AssignmentHeadLines, "AssignmentId", "AssignmentName", assignedTask.TaskId);
    return View(assignedTask);
}
```

This ASP.NET Core MVC Create action method handles the creation of a new assigned task. First, it checks if a file is uploaded. If not, it returns the view with a dropdown list of assignments (View-Data["Task"]). If a file exists, the method sets the **Assigned-Task-Student-Id** based on the currently logged-in user's ID, assigns the current UTC time plus 2 hours to **AssignedTask-Date,** and generates a **Task-URL** using a file upload function. It then checks if the **AssignedTask-Student-Id, Task-URL**, **Assigned-Task-Date** are not null, and if the Task Id is not 0. If all validations pass, the assigned task is added to the database, saved asynchronously, and the user is redirected to the Index page. If validation fails, the form is shown again with the appropriate assignment list.

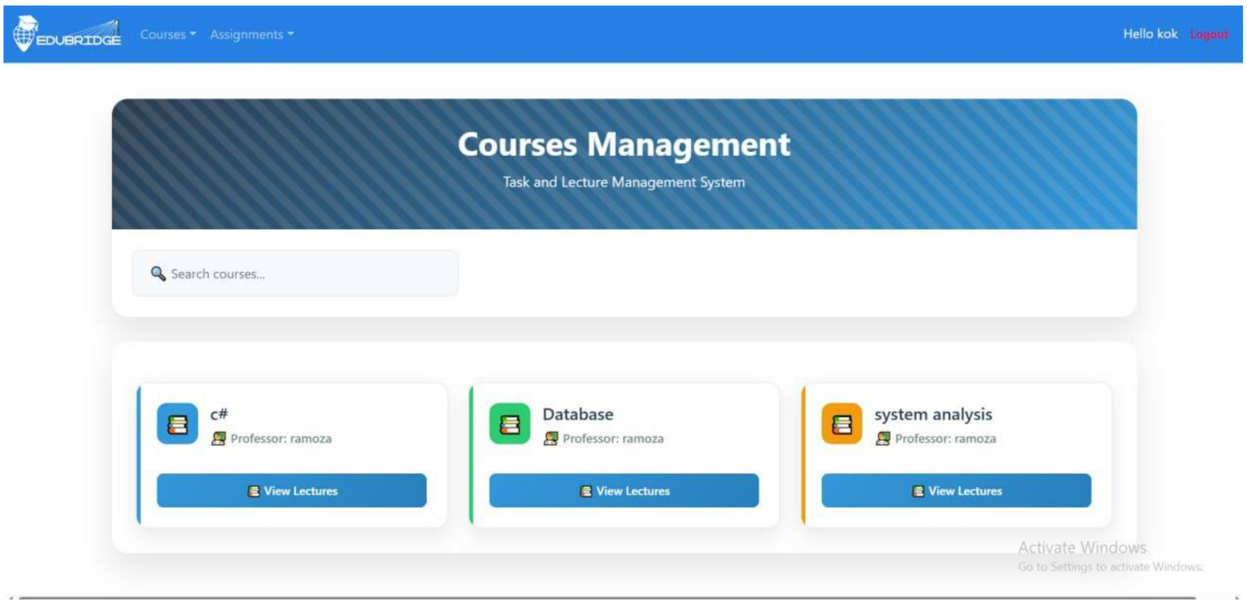
4.3-Submit Task Function:



```
2 references
private string UploadFile(IFormFile File)
{
    var uniqueFileName = GetUniqueFileName(File.FileName);
    var uploads = Path.Combine(_hostingEnvironment.WebRootPath, "AssignmentFiles");
    var filePath = Path.Combine(uploads, uniqueFileName);
    File.CopyTo(new FileStream(filePath, FileMode.Create));
    return uniqueFileName;
}
1 reference
```

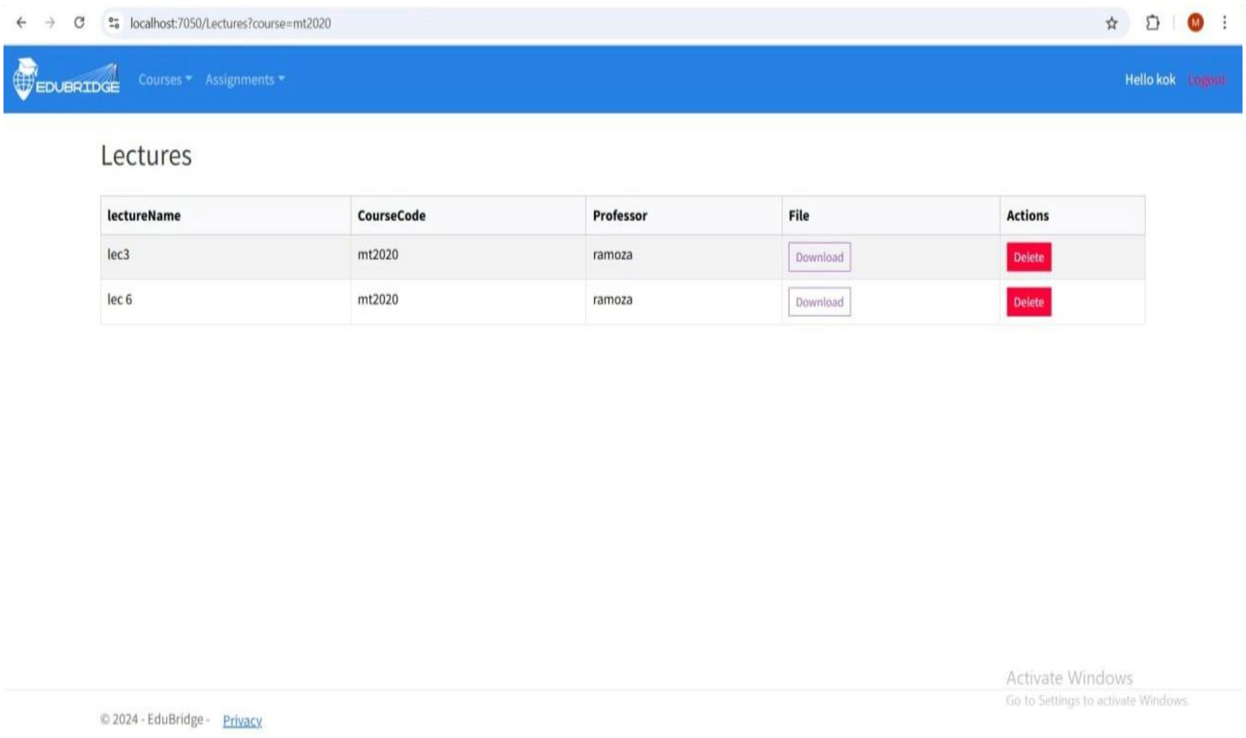
This method **Upload-File** handles uploading a file to the server. It first generates a unique file name using the original file name. Then, it constructs the upload path by combining the web root path with a folder named **"Assignment-Files".** After that, it combines the upload path and the unique file name to get the full file path. Finally, it copies the uploaded file to the specified location using a file stream and returns the unique file name.

4.4- Course Page (Student View):



Displays a list of courses the student is enrolled in. The student can select any course to view its related content.

4.5- Lecture Page (Student View):



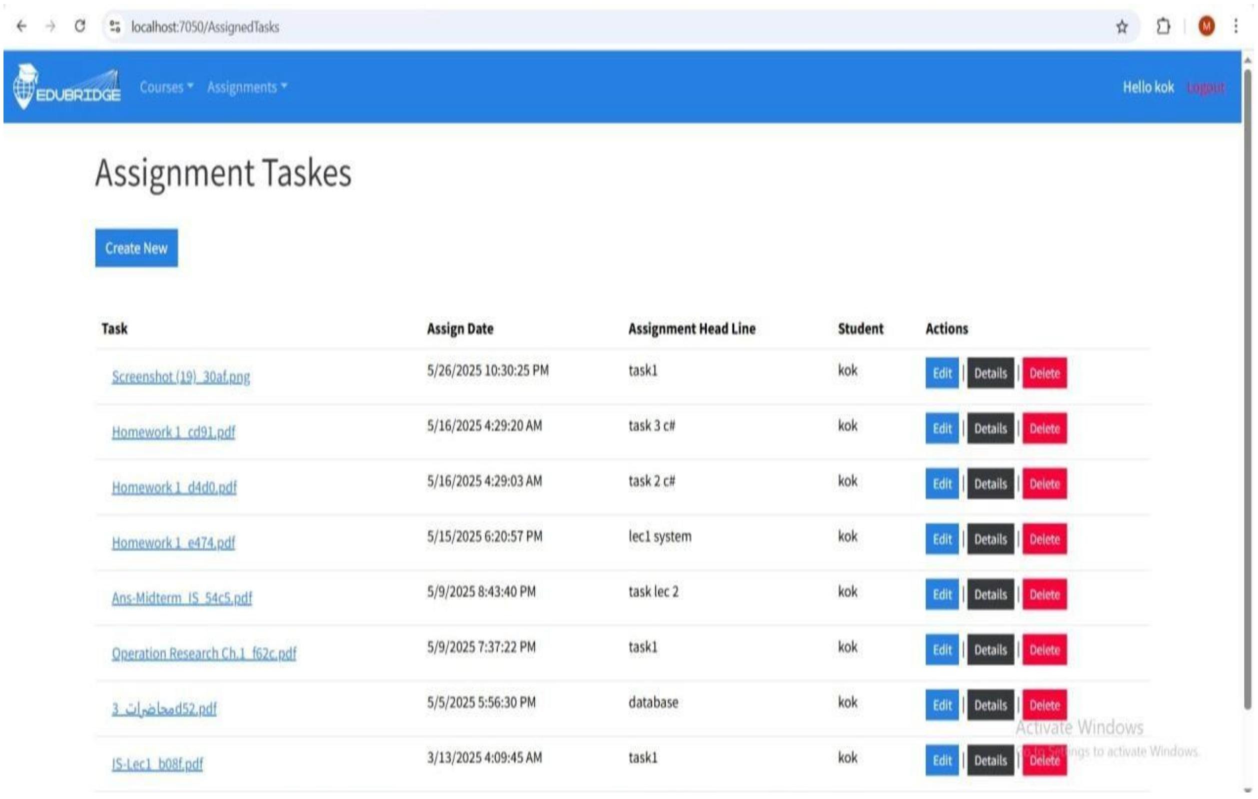
Lectures

lectureName	CourseCode	Professor	File	Actions
lec3	mt2020	ramoza	Download	Delete
lec 6	mt2020	ramoza	Download	Delete

© 2024 - EduBridge - Privacy

Shows all the lectures uploaded for the selected course, allowing the student to delete or download them.

4.6- Assignment Page (Student View):



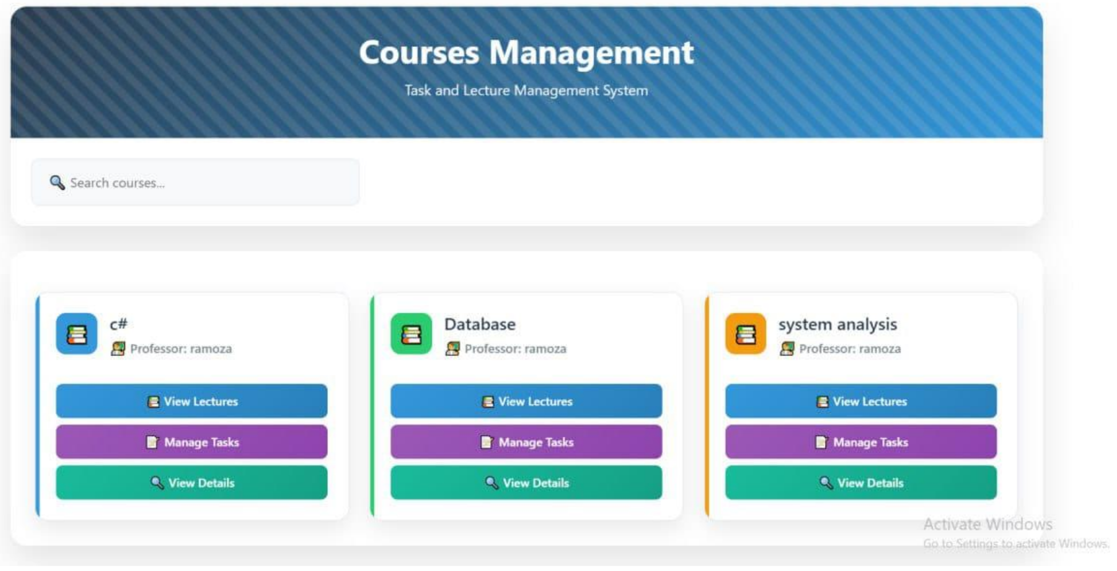
localhost:7050/AssignedTasks

Assignment Taskes

Task	Assign Date	Assignment Head Line	Student	Actions
Screenshot (19)_30af.png	5/26/2025 10:30:25 PM	task1	kok	Edit | Details | Delete
Homework 1_cd91.pdf	5/16/2025 4:29:20 AM	task 3 c#	kok	Edit | Details | Delete
Homework 1_d4d0.pdf	5/16/2025 4:29:03 AM	task 2 c#	kok	Edit | Details | Delete
Homework 1_e474.pdf	5/15/2025 6:20:57 PM	lec1 system	kok	Edit | Details | Delete
Ans-Midterm_IS_54c5.pdf	5/9/2025 8:43:40 PM	task lec 2	kok	Edit | Details | Delete
Operation Research Ch.1_f62c.pdf	5/9/2025 7:37:22 PM	task1	kok	Edit | Details | Delete
3_محاضراتd52.pdf	5/5/2025 5:56:30 PM	database	kok	Edit | Details | Delete
IS-Lec1_b08f.pdf	3/13/2025 4:09:45 AM	task1	kok	Edit | Details | Delete

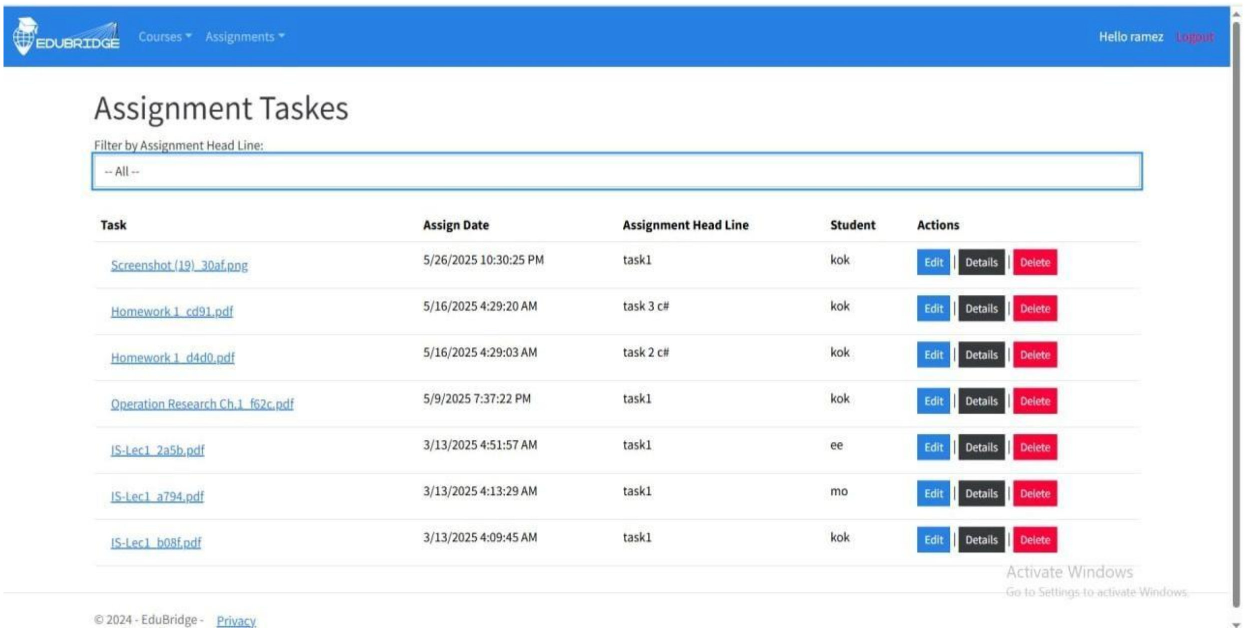
The Assignment Page enables students to edit, delete, and see details for their assignments through this page.

4.7- Course Page (Doctor View):



Allows the doctor to view the courses they are teaching, and access each course to manage its lectures and assignments.

4.8- Assignment Page (Doctor View):



Assignment Taskes

Filter by Assignment Head Line:

-- All --

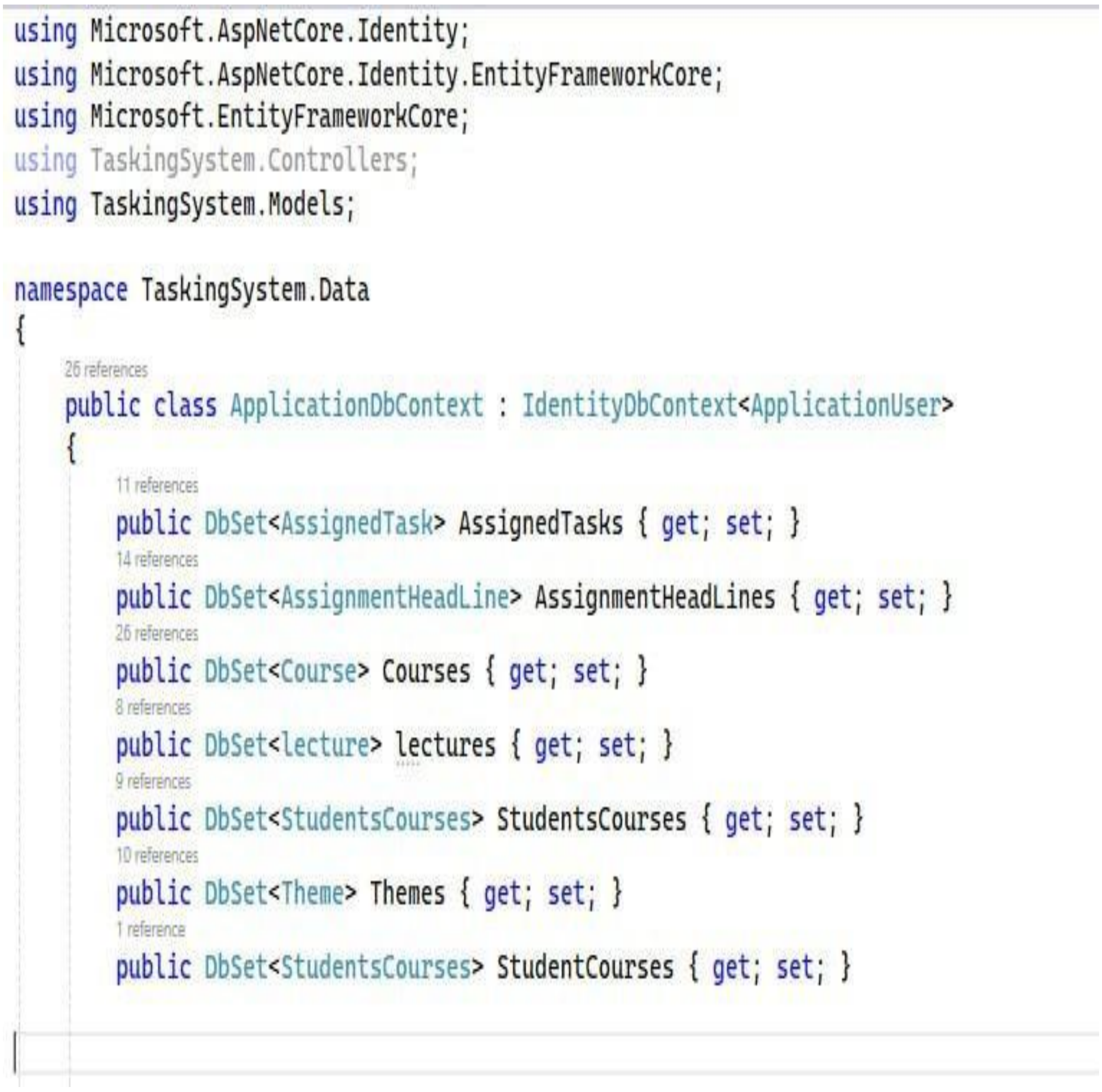
Task	Assign Date	Assignment Head Line	Student	Actions
Screenshot (19)_30af.png	5/26/2025 10:30:25 PM	task1	kok	Edit | Details | Delete
Homework 1_cd91.pdf	5/16/2025 4:29:20 AM	task 3 c#	kok	Edit | Details | Delete
Homework 1_d4d0.pdf	5/16/2025 4:29:03 AM	task 2 c#	kok	Edit | Details | Delete
Operation Research Ch.1_f62c.pdf	5/9/2025 7:37:22 PM	task1	kok	Edit | Details | Delete
IS-Lec1_2a5b.pdf	3/13/2025 4:51:57 AM	task1	ee	Edit | Details | Delete
IS-Lec1_a794.pdf	3/13/2025 4:13:29 AM	task1	mo	Edit | Details | Delete
IS-Lec1_b08f.pdf	3/13/2025 4:09:45 AM	task1	kok	Edit | Details | Delete

Activate Windows
Go to Settings to activate Windows.

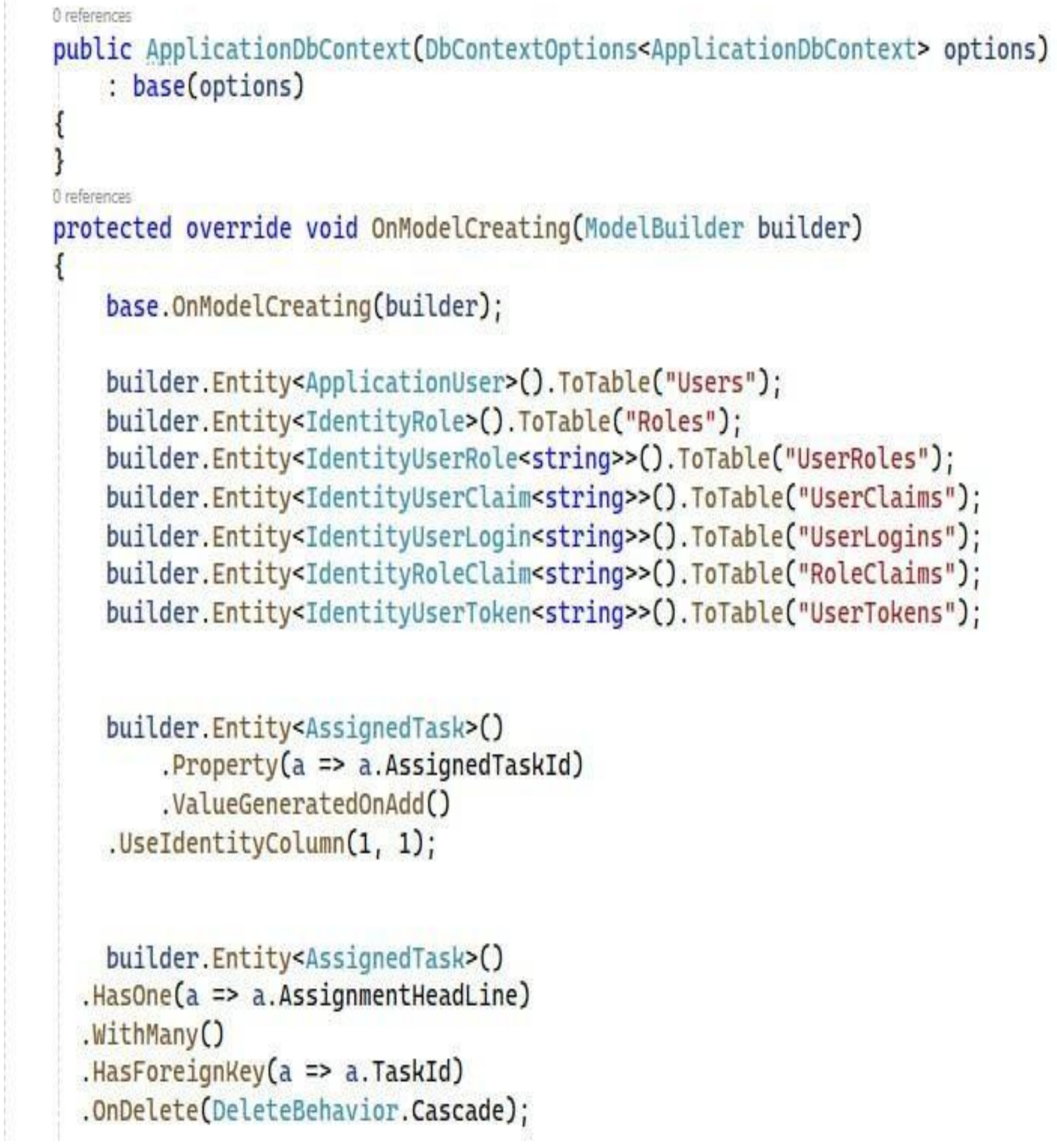
© 2024 - EduBridge - Privacy

Allows the doctor to Edit, Delete, and manage assignments for each course, including setting deadlines and uploading task details.

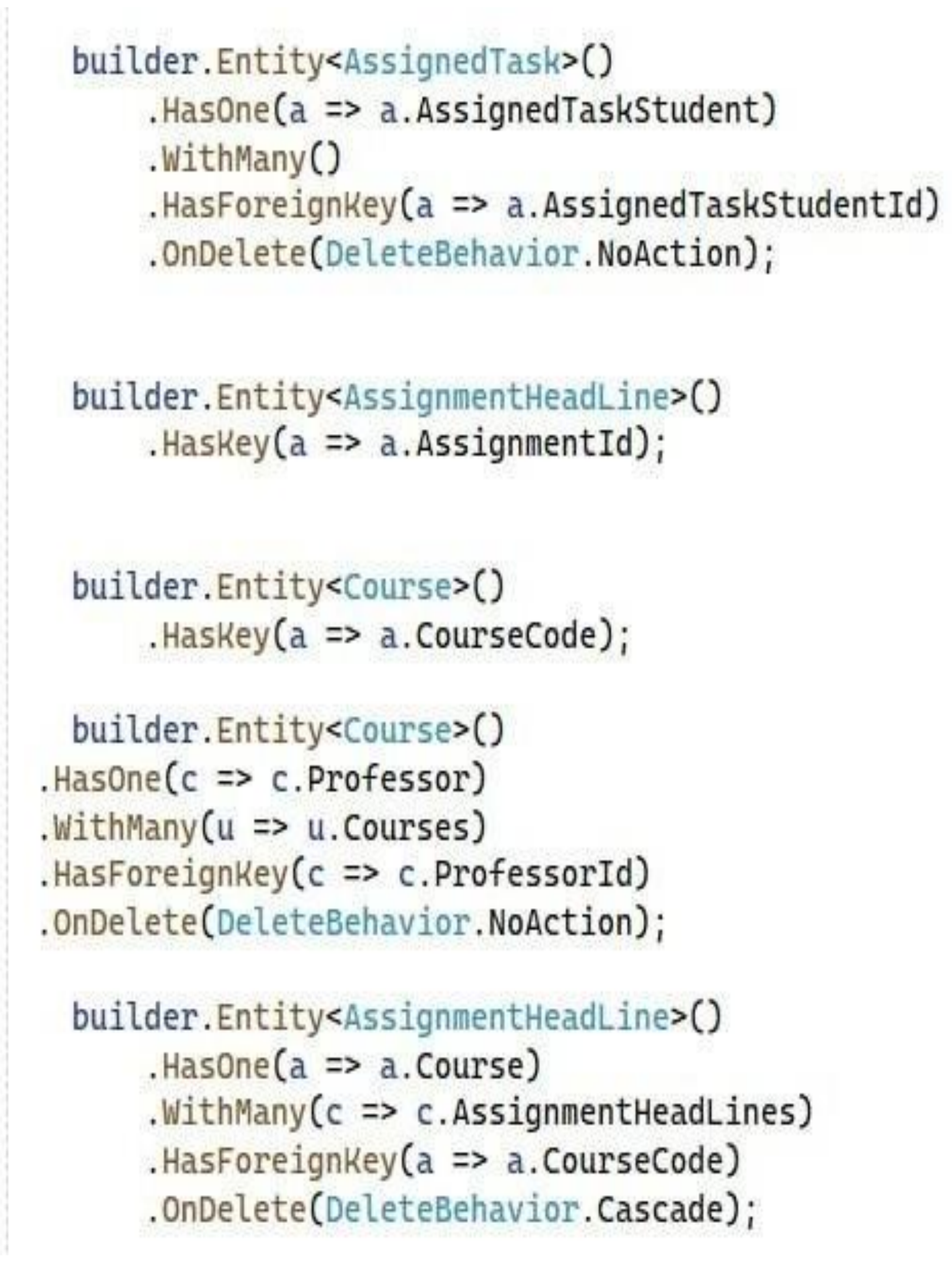
4.9-Database Context:



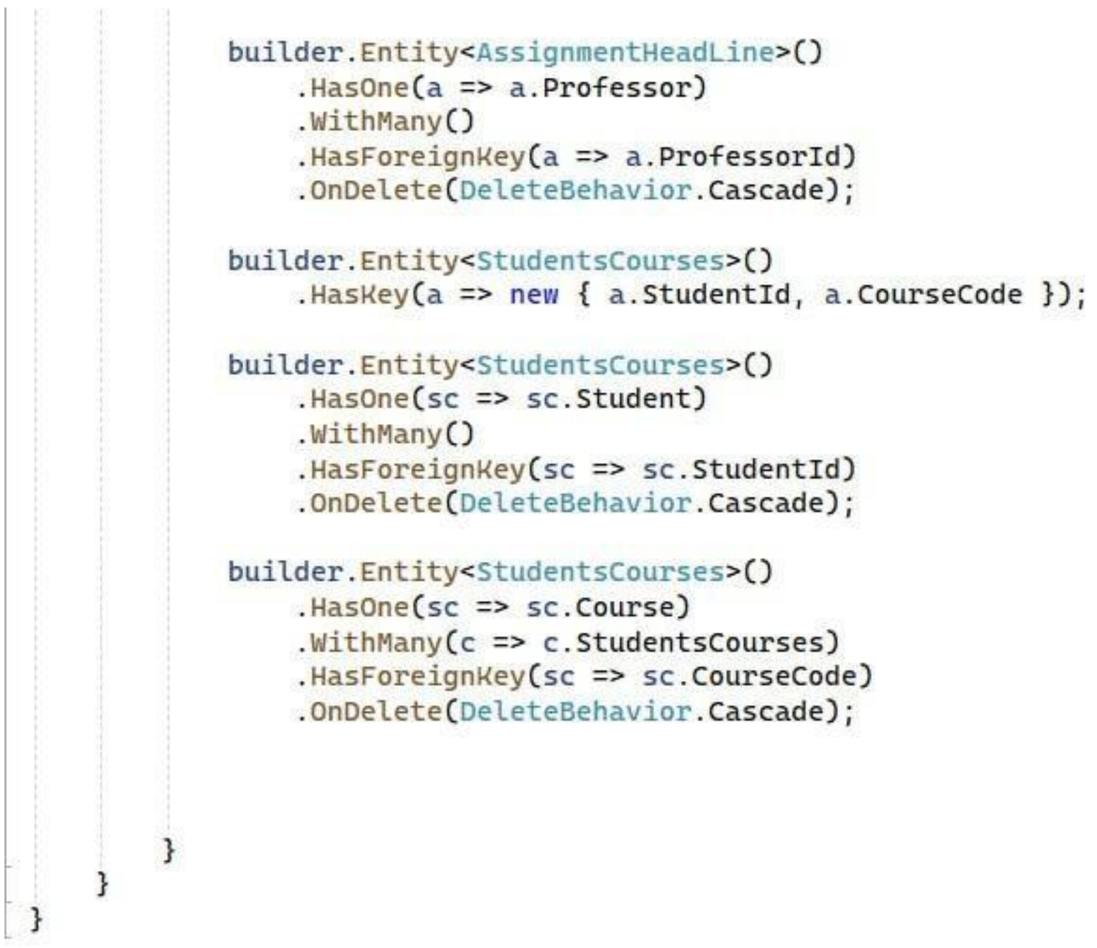
```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using TaskingSystem.Controllers;
using TaskingSystem.Models;

namespace TaskingSystem.Data
{
    26 references
    public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        11 references
        public DbSet<AssignedTask> AssignedTasks { get; set; }
        14 references
        public DbSet<AssignmentHeadLine> AssignmentHeadLines { get; set; }
        26 references
        public DbSet<Course> Courses { get; set; }
        8 references
        public DbSet<lecture> lectures { get; set; }
        9 references
        public DbSet<StudentsCourses> StudentsCourses { get; set; }
        10 references
        public DbSet<Theme> Themes { get; set; }
        1 reference
        public DbSet<StudentsCourses> StudentCourses { get; set; }
```



```
0 references
public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
    : base(options)
{
}
0 references
protected override void OnModelCreating(ModelBuilder builder)
{
    base.OnModelCreating(builder);

    builder.Entity<ApplicationUser>().ToTable("Users");
    builder.Entity<IdentityRole>().ToTable("Roles");
    builder.Entity<IdentityUserRole<string>>().ToTable("UserRoles");
    builder.Entity<IdentityUserClaim<string>>().ToTable("UserClaims");
    builder.Entity<IdentityUserLogin<string>>().ToTable("UserLogins");
    builder.Entity<IdentityRoleClaim<string>>().ToTable("RoleClaims");
    builder.Entity<IdentityUserToken<string>>().ToTable("UserTokens");


    builder.Entity<AssignedTask>()
        .Property(a => a.AssignedTaskId)
        .ValueGeneratedOnAdd()
    .UseIdentityColumn(1, 1);


    builder.Entity<AssignedTask>()
  .HasOne(a => a.AssignmentHeadLine)
  .WithMany()
  .HasForeignKey(a => a.TaskId)
  .OnDelete(DeleteBehavior.Cascade);
```



```
builder.Entity<AssignedTask>()
    .HasOne(a => a.AssignedTaskStudent)
    .WithMany()
    .HasForeignKey(a => a.AssignedTaskStudentId)
    .OnDelete(DeleteBehavior.NoAction);


builder.Entity<AssignmentHeadLine>()
    .HasKey(a => a.AssignmentId);


builder.Entity<Course>()
    .HasKey(a => a.CourseCode);

builder.Entity<Course>()
.HasOne(c => c.Professor)
.WithMany(u => u.Courses)
.HasForeignKey(c => c.ProfessorId)
.OnDelete(DeleteBehavior.NoAction);

builder.Entity<AssignmentHeadLine>()
    .HasOne(a => a.Course)
    .WithMany(c => c.AssignmentHeadLines)
    .HasForeignKey(a => a.CourseCode)
    .OnDelete(DeleteBehavior.Cascade);
```



```
            builder.Entity<AssignmentHeadLine>()
                .HasOne(a => a.Professor)
                .WithMany()
                .HasForeignKey(a => a.ProfessorId)
                .OnDelete(DeleteBehavior.Cascade);

            builder.Entity<StudentsCourses>()
                .HasKey(a => new { a.StudentId, a.CourseCode });

            builder.Entity<StudentsCourses>()
                .HasOne(sc => sc.Student)
                .WithMany()
                .HasForeignKey(sc => sc.StudentId)
                .OnDelete(DeleteBehavior.Cascade);

            builder.Entity<StudentsCourses>()
                .HasOne(sc => sc.Course)
                .WithMany(c => c.StudentsCourses)
                .HasForeignKey(sc => sc.CourseCode)
                .OnDelete(DeleteBehavior.Cascade);



        }
    }
}
```

This class defines the database tables and their relationships.

Conclusion

This graduation project successfully addressed the common challenges faced in academic environments related to course material delivery, assignment management, and communication between students and instructors. The developed web-based educational platform offers an intuitive, centralized system that simplifies these processes through user-friendly dashboards, secure login, real-time notifications, and performance tracking tools.

By focusing on usability, simplicity, and the specific needs of local universities, the system provides a practical solution that enhances the overall educational experience. Feedback from initial users has shown that the platform significantly reduces the administrative burden on faculty while improving students’ access to learning resources and timely feedback.

This project not only showcases the technical and analytical skills of the development team but also provides a foundation for scalable and impactful educational solutions tailored to our academic environment.

Future Work

While the platform currently meets essential educational needs, several enhancements can be considered for future development:

1- Mobile Application Integration: Creating dedicated Android and iOS apps for improved accessibility and ease of use on mobile devices.

2- AI-Based Feedback and Grading: Implementing artificial intelligence for automated grading and personalized feedback based on student performance.

3- Video Conferencing Tools: Integrating live class features such as virtual classrooms and video calls to support remote learning.

4- Data Analytics Dashboard: Providing detailed analytics for instructors and administrators to monitor class engagement and student progress.

5- Cloud-Based Storage: Enhancing storage capacity and reliability using scalable cloud services.

6- Gamification Features: Introducing badges, leaderboards, and rewards to boost student motivation and engagement.

By pursuing these future enhancements, the platform can evolve into a comprehensive learning management ecosystem that adapts to the evolving needs of modern education.