VIETNAM NATIONAL UNIVERSITY – HO CHI MINH CITY

INTERNATIONAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

PRINCIPLES OF DATABASE MANAGEMENT

IT079IU

FINAL REPORT

Course by Assoc. Prof. Nguyen Van Sinh

MSc. Nguyen Quang Phu

TOPIC: LEARNING MANAGEMENT SYSTEM

Written By

Tran Ngoc Dang Khoi	ITCSIU21197
Do Duy Anh	ITCSIU21163

TABLE OF CONTENT

Project's Github: https://github.com/koitran14/Course-Management-Project.git

LIST OF FIGURES

Project's Github: https://github.com/koitran14/Course-Management-Project.git

Project's Github: https://github.com/koitran14/Course-Management-Project.git

LIST OF TABLES

Project's Github: https://github.com/koitran14/Course-Management-Project.git

I. Introduction

1. Abstract

The "Blackboard" project is meticulously crafted to serve as an advanced education management system, streamlining academic administration with its array of essential functionalities. Users, whether they be students or instructors, are empowered with seamless access to a range of features designed to enhance their educational experience.

The user-centric approach of the project is evident in its commitment to user-friendly interfaces, role-based access control, and timely notifications. This ensures that educational processes are not only efficient but also tailored to the specific needs of both students and irs. The system places a strong emphasis on empowering users to instructonteract with the platform effortlessly, contributing to a more enriched learning environment.

A critical aspect of the project is its focus on database design, adhering to the B.C. normal form. The logical design process involves meticulous planning, defining data entities, relationships, attributes, and constraints. By ensuring data integrity and eliminating anomalies, the project aims to optimize data retrieval and storage. This logical design is pivotal in guaranteeing the system's efficiency, reliability, and scalability as it evolves over time.

The inclusion of simulated data within the database for testing purposes showcases a commitment to robust system validation. This data can seamlessly be replaced with real-world data upon deployment, ensuring that the system is well-prepared for actual usage scenarios.

The user interface of the system is intentionally designed to be user-friendly, allowing easy navigation and access to various features. The login

function, a gateway to personalized experiences, enables users to interact with the database using their individual accounts. This personalized interaction further underscores the project's commitment to tailoring the educational experience to the unique needs and roles of each user.

In summary, the "Blackboard" project stands as a testament to the integration of advanced features, meticulous database design, and a user-centric approach. By addressing the specific needs of both students and instructors, the project aims to contribute significantly to the efficiency and richness of the educational experience.

2. System Overview

Student Information System (SIS) plays a central role in education, overseeing tasks such as student data management, enrollment, and grading. Modern systems go beyond basic functionalities, offering real-time updates and analytics to enhance decision-making.

Learning Management System (LMS) platforms complement the educational landscape by facilitating online learning, efficient course management, and streamlined content delivery. Effective communication is vital in education, and integration with communication tools like email and discussion forums is essential for fostering better interaction among teachers, students, and parents.

The incorporation of advanced analytics in these systems provides valuable insights into student performance and resource utilization. Predictive analytics further contribute by identifying at-risk students, enabling timely interventions to support their academic journey.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

A commitment to inclusivity is evident in these systems, as they strive to be accessible to individuals with diverse needs. Adhering to the principles of universal design, these platforms aim to create an inclusive learning environment for all.

3. Goal

- Design the database to suit the requirements of Normalization BC Normal Form
- Connect the front-end interface (application) to the back-end database
- Develop the functions such as login, account creation,profile edition, content distribution, interactive materials using complex queries.
- Take security measures in the database to ensure robust privacy for the users' data .

4. Technique & tool used

- ERDplus to build Entity-Relationship Diagrams.
- Work along with Express.js, or simply Express, is a back-end web application framework for building RESTful APIs with Node.js, released as free and open-source software under the MIT License.
- Next.js is used to simplify and optimize the development of React applications, providing features like server-side rendering, improved performance, simplified routing, and a great developer experience.
- Tailwind CSS is used to streamline the styling process in web development, providing a utility-first approach that allows for rapid prototyping, responsive design, and consistent styling across projects..
- GitHub to easily collaborate on the project.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

II. Task Timeline and Division

1. Contribution

Table 1. Individual responibility and contribution

Name	Role	Contribution (%)
Tran Ngoc Dang Khoi	Project Leader	55
Do Duy Anh	Member	45

2. Project Timeline & Task Division

Table 2. Sprint planning and task division for individuals and teams

Project's Github: https://github.com/koitran14/Course-Management-Project.git

Stage	Action	Member	Week
PLANNING	Giving ideas on tools and project structure for construction.	Dang Khoi	**1**
	Research for references and technical documentation	Dang Khoi	
	Determine scopes, learning objectives, and goals for the project	All	
	Establish development timeline	All	
	Determine proper technology stack and database management system	All	
	Propose necessary features and their priorities	All	
	Agree on communication, workflow, and tools to use	All	
	Week 1 progress report meeting	All	
CONCEPTUAL DESIGN	Identify use cases and actors of the system	All	**2**

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Specify types of information that are essential to be stored in the database	All	
	Design the relational models	Duy Anh	
	Prepare the E.R. diagram and Class diagram	Duy Anh	
	Design sample visual representation for the interface	All	
	Week 2 progress report meeting	All	
IMPLEMENTATIONS	Database creation and setup	Dang Khoi	**3 - 7**
	Setup database tables, theirrelationship, and constraints	All	
	Setup Client - Server model	Dang Khoi	
	Implement necessary queries for each functionality	All	
	Work on new user registration	Dang Khoi	
	Work on user login/logout and authentication	Dang Khoi	

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Develop application interface and client-side functionalities	All	
	Weekly progress report meeting	All	
TESTING	Code review and refactoring	All	
	Bug detection and fixing	All	
	Fix and modify the web until stable	All	
PRESENTATION	Final report	All	**8**
	Presentation slides	All	

III. Project Analysis

1. Requirement

- User Authentication and Authorization:

- Implement user registration and login functionalities.
- Apply role-based access control to distinguish between students and instructors.

- User Profiles:

  Allow users (students and instructors) to view and edit their profiles.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

- Educational Features:

- Provide course exploration functionalities for both students and instructors.
- Display comprehensive course details, including content, announcements, assignments, and alerts.

- Efficient User Interaction:

- Design user-friendly interfaces to ensure easy navigation and accessibility.
- Emphasize a seamless user experience to contribute to an enriched learning environment.

- Database Design:

- Adhere to the B.C. normal form in database design.
- Meticulously plan and define data entities, relationships, attributes, and constraints.
- Ensure data integrity and eliminate anomalies in the database.

- Data Retrieval and Storage Optimization:

  Optimize data retrieval and storage for system efficiency, reliability, and scalability.

- Notification System:

  Implement a notification system to provide timely alerts and announcements to users.

- Personalized User Experience:

- Design a personalized login function allowing users to interact with the database using individual accounts.
- Tailor the educational experience based on the unique needs and roles of each user.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

2. Approach Analysis

2.1. *Reviewed Materials*

- Learn Next.js

(URL : https://nextjs.org/learn?utm_source=next-site&utm_medium=homepage-cta&utm_campaign=home)

- Learn Nodejs

(URL : https://nodejs.org/en/learn/getting-started/introduction-to-nodejs)

- Express framework

(URL : Express - Node.js web application framework (expressjs.com))

- Learn TailwindCSS

(URL : https://tailwindcss.com/docs/installation)

2.2. *Research Approach*

Methodology & Process

The team's working method focuses on planning ahead for tasks, followed by flexible adjustments and updates weekly through progress reporting meetings. Every week, we conduct meetings to assess the project's progress and discuss necessary adjustments. Through these meetings, we reach consensus and implement timely adjustments and improvements to the project without affecting its organizational structure.

This method helps us maintain flexibility in the working process while ensuring that every team member is fully informed about the progress and any changes. Consensus through weekly meetings helps avoid conflicts within the project structure and provides a conducive environment for effective implementation of modifications. This approach allows us to uphold consistency and flexibility in team collaboration, optimizing both performance and the quality of the project.

3. System Analysis

3.1. *Database Design*

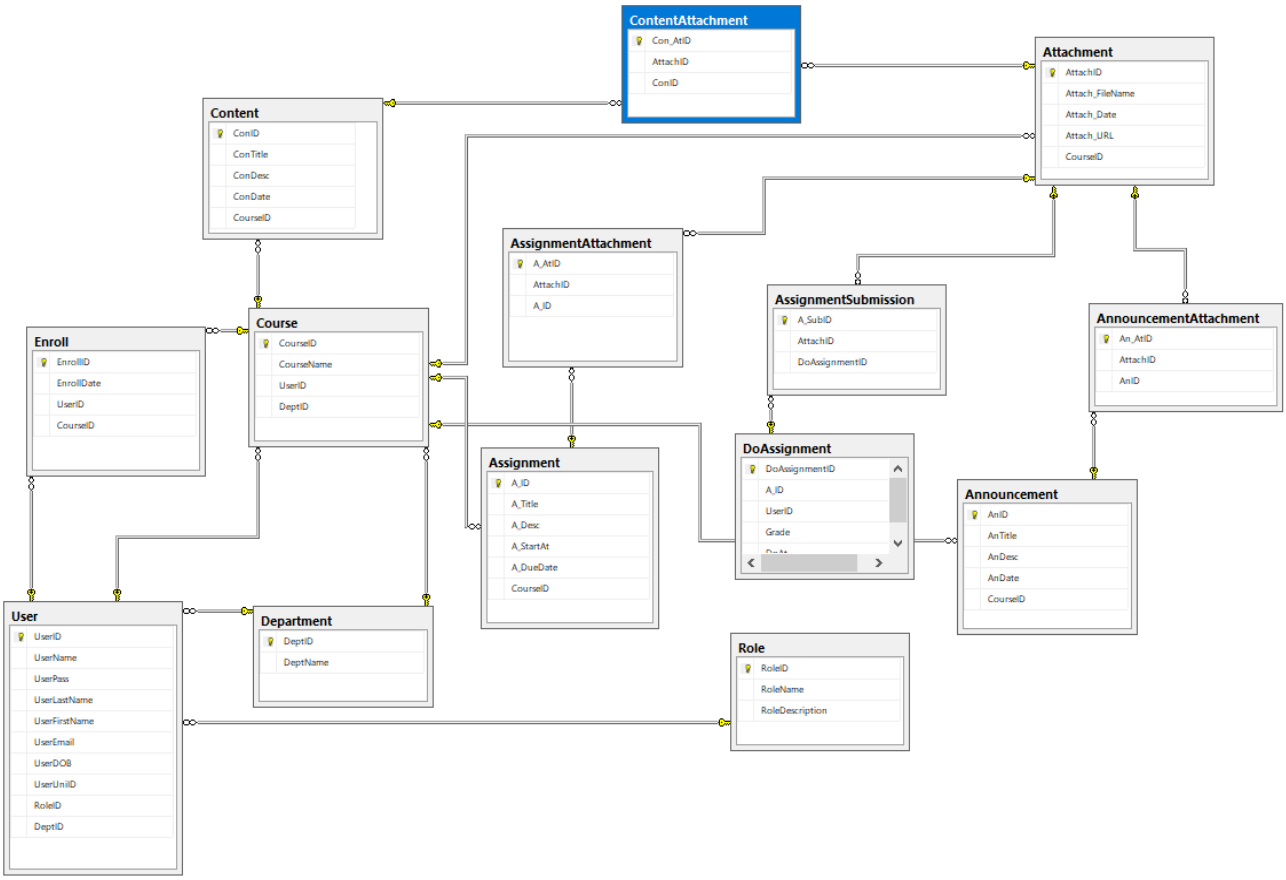

Figure 1. Entity-relationship diagram using SQL Server

The database has achieved the highest B.C normal form

Table 3. Normal form achievement

Normal Form	Description
1 N.F	No duplicated rows or columns. Each column has only one value for each row in the table.
2 N.F	Has no partial dependency, all non-key attributes are fully dependent

	on a primary key.
3 N.F	Have no transitive partial dependency.
BC N.F	Every non-trivial functional dependency in the database depends on a candidate key.

3.2. *Database and Table Creation*

a. Database Creation

```
CREATE DATABASE CourseDB;

USE CourseDB;
```

b. Tables Creation

The current version of the Entity- Relationship Diagram (**Figure 1**) has the following entities:

Table 4. Entities and Attributes

Table	Attributes
Role	RoleID VARCHAR(25) PRIMARY KEY RoleName varchar(25) RoleDescription varchar(1000)
User	UserID varchar(25) PRIMARY KEY UserName varchar(25) UserPass varchar(25) UserLastName VARCHAR(25) UserFirstName varchar(25) UserEmail varchar(50) UserDOB DATETIME UserUniID VARCHAR(25) RoleID VARCHAR(25) DeptID VARCHAR(25)

Project's Github: https://github.com/koitran14/Course-Management-Project.git

Department	DeptID varchar(25) PRIMARY KEY DeptName varchar(50)
Enroll	EnrollID varchar(25) PRIMARY KEY EnrollDate DATETIME UserID VARCHAR(25) CourseID VARCHAR(25)
Course	CourseID VARCHAR(25) PRIMARY KEY CourseName VARCHAR(50) UserID VARCHAR(25) DeptID VARCHAR(25)
Content	ConID VARCHAR(25) PRIMARY KEY ConTitle VARCHAR (50) ConDesc VARCHAR (1000) ConDate DATETIM, CourseID VARCHAR(25)
Announcement	AnID VARCHAR(25) PRIMARY KEY AnTitle VARCHAR(50) AnDesc VARCHAR(1000) AnDate DATETIME CourseID VARCHAR(25)
Assignment	A_ID VARCHAR(25) PRIMARY KEY A_Title VARCHAR(50) A_Desc VARCHAR(1000) A_StartAt DATETIME A_DueDate DATETIME CourseID VARCHAR(25)

Project's Github: https://github.com/koitran14/Course-Management-Project.git

Attachment	AttachID VARCHAR(25) PRIMARY KEY Attach_FileName VARCHAR(100) Attach_Date DATETIME Attach_URL VARCHAR(150) CourseID VARCHAR(25)
AnnouncementAttachment	An_AtID uniqueidentifier PRIMARY KEY DEFAULT NEWID() AttachID VARCHAR(25) AnID VARCHAR(25)
ContentAttachment	Con_AtID uniqueidentifier PRIMARY KEY DEFAULT NEWID() AttachID VARCHAR(25) ConID VARCHAR(25)
AssignmentAttachment	A_AtID uniqueidentifier PRIMARY KEY DEFAULT NEWID() AttachID VARCHAR(25) A_ID VARCHAR(25)
AssignmentSubmission	A_SubID uniqueidentifier PRIMARY KEY DEFAULT NEWID() AttachID VARCHAR(25) DoAssignmentID UNIQUEIDENTIFIER
DoAssignment	DoAssignmentID UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID() A_ID VARCHAR(25) UserID VARCHAR(25) Grade FLOAT DoAt DateTime

Foreign Key Relationships:

Various foreign key constraints are added to establish relationships between tables, ensuring data integrity and consistency.

For example, the RoleID in the User table references the Role (RoleID) table, connecting user roles.

Similarly, other foreign key constraints connect tables such as Enroll, Course, Content, Announcement, Assignment, Attachment, etc., to maintain relationships between entities.

Create Tables:

Role Table

```
CREATE TABLE Role (
    RoleID VARCHAR(25) PRIMARY KEY,
    RoleName varchar(25),
    RoleDescription varchar(1000),
);
```

User Table

```
CREATE TABLE [User] (
    UserID varchar(25) PRIMARY KEY,
    UserName varchar(25),
    UserPass varchar(25),
    UserLastName VARCHAR(25),
    UserFirstName varchar(25),
    UserEmail varchar(50),
    UserDOB DATETIME,
    UserUniID VARCHAR(25),
    RoleID VARCHAR(25),
    DeptID VARCHAR(25),
);
```

Department Table

```
CREATE TABLE Department(
    DeptID varchar(25) primary key,
    DeptName varchar(50),
);
```

Enroll Table

```
CREATE TABLE Enroll (
    EnrollID varchar(25) PRIMARY KEY,
    EnrollDate DATETIME,
    UserID VARCHAR(25),
    CourseID VARCHAR(25)
);
```

Course Table

```
CREATE TABLE Course (
    CourseID VARCHAR(25) PRIMARY KEY,
    CourseName VARCHAR(50),
    UserID VARCHAR(25),
    DeptID VARCHAR(25),
);
```

Content Table

```
CREATE TABLE Content (
    ConID VARCHAR(25) PRIMARY KEY,
    ConTitle VARCHAR (50),
    ConDesc VARCHAR (1000),
    ConDate DATETIME,
    CourseID VARCHAR(25),
);
```

Announcement Table

Project's Github: https://github.com/koitran14/Course-Management-Project.git

```
CREATE TABLE Announcement(
    AnID VARCHAR(25) PRIMARY KEY,
    AnTitle VARCHAR(50),
    AnDesc VARCHAR(1000),
    AnDate DATETIME,
    CourseID VARCHAR(25),
);
```

Assignment Table

```
CREATE TABLE Assignment (
    A_ID VARCHAR(25) PRIMARY KEY,
    A_Title VARCHAR(50),
    A_Desc VARCHAR(1000),
    A_StartAt DATETIME,
    A_DueDate DATETIME,
    CourseID VARCHAR(25),
);
```

Attachment Table

```
CREATE TABLE Attachment (
    AttachID VARCHAR(25) PRIMARY KEY,
    Attach_FileName VARCHAR(100),
    Attach_Date DATETIME,
    Attach_URL VARCHAR(150),
    CourseID VARCHAR(25)
);
```

AnnouncementAttachment Table

```
CREATE TABLE AnnouncementAttachment (
    An_AtID uniqueidentifier PRIMARY KEY DEFAULT NEWID(),
    AttachID VARCHAR(25),
    AnID VARCHAR(25)
);
```

ContentAttachment Table

```
CREATE TABLE ContentAttachment (
    Con_AtID uniqueidentifier PRIMARY KEY DEFAULT NEWID(),
    AttachID VARCHAR(25),
    ConID VARCHAR(25)
);
```


Project's Github: https://github.com/koitran14/Course-Management-Project.git

AssignmentAttachment Table

```
CREATE TABLE AssignmentAttachment (
    A_AtID uniqueidentifier PRIMARY KEY DEFAULT NEWID(),
    AttachID VARCHAR(25),
    A_ID VARCHAR(25)
);
```

AssignmentSubmission Table

```
CREATE TABLE AssignmentSubmission (
    A_SubID uniqueidentifier PRIMARY KEY DEFAULT NEWID(),
    AttachID VARCHAR(25),
    DoAssignmentID UNIQUEIDENTIFIER,
);
```

DoAssignment Table

```
CREATE TABLE DoAssignment(
    DoAssignmentID UNIQUEIDENTIFIER PRIMARY KEY DEFAULT NEWID(),
    A_ID VARCHAR(25),
    UserID VARCHAR(25),
    Grade FLOAT,
    DoAt DateTime,
)
```

Foreign Key:

Adding foreign key constraints to the [User] table

```
ALTER TABLE [User]
ADD CONSTRAINT FK_User_Role FOREIGN KEY (RoleID) REFERENCES Role(RoleID);

ALTER TABLE [User]
ADD CONSTRAINT FK_User_Dept FOREIGN KEY (DeptID) REFERENCES Department(DeptID);
```

Adding foreign key constraints to the Enroll table

```
ALTER TABLE Enroll
ADD CONSTRAINT FK_Enroll_User FOREIGN KEY (UserID) REFERENCES [User](UserID);

ALTER TABLE Enroll
ADD CONSTRAINT FK_Enroll_Course FOREIGN KEY (CourseID) REFERENCES Course(CourseID);
```

Adding foreign key constraints to the Course table

```
ALTER TABLE Course
ADD CONSTRAINT FK_Course_User FOREIGN KEY (UserID) REFERENCES [User](UserID);

ALTER TABLE Course
ADD CONSTRAINT FK_Course_Dept FOREIGN KEY (DeptID) REFERENCES Department(DeptID);
```

Adding foreign key constraints to the Content table

```
ALTER TABLE Content
ADD CONSTRAINT FK_Content_Course FOREIGN KEY (CourseID) REFERENCES Course(CourseID);
```

Adding foreign key constraints to the Announcement table

```
ALTER TABLE Announcement
ADD CONSTRAINT FK_Announcement_Course FOREIGN KEY (CourseID) REFERENCES Course(CourseID);
```

Adding foreign key constraints to the Assignment table

```
ALTER TABLE Assignment
ADD CONSTRAINT FK_Assignment_Course FOREIGN KEY (CourseID) REFERENCES Course(CourseID);
```

Adding foreign key constraints to the Attachment table

```
ALTER TABLE Attachment
ADD CONSTRAINT FK_Attachment_Course FOREIGN KEY (CourseID) REFERENCES Course(CourseID);

ALTER TABLE AssignmentAttachment
ADD CONSTRAINT FK_AA_Assignment FOREIGN KEY (A_ID) REFERENCES Assignment(A_ID);

ALTER TABLE AssignmentAttachment
ADD CONSTRAINT FK_AA_Attachment FOREIGN KEY (AttachID) REFERENCES Attachment(AttachID);

ALTER TABLE AnnouncementAttachment
ADD CONSTRAINT FK_AnA_Announcement FOREIGN KEY (AnID) REFERENCES Announcement(AnID);

ALTER TABLE AnnouncementAttachment
ADD CONSTRAINT FK_AnA_Attachment FOREIGN KEY (AttachID) REFERENCES Attachment(AttachID);

ALTER TABLE ContentAttachment
ADD CONSTRAINT FK_CAt_Content FOREIGN KEY (ConID) REFERENCES Content(ConID);

ALTER TABLE ContentAttachment
ADD CONSTRAINT FK_CAt_Attachment FOREIGN KEY (AttachID) REFERENCES Attachment(AttachID);

ALTER TABLE AssignmentSubmission
ADD CONSTRAINT FK_AssignmentSubmission_DoAssignment FOREIGN KEY (DoAssignmentID) REFERENCES DoAssignment(DoAssignmentID);

ALTER TABLE AssignmentSubmission
ADD CONSTRAINT FK_AssignmentSubmission_Attachment FOREIGN KEY (AttachID) REFERENCES Attachment(AttachID);
```

3.3. *Database Data Insertion*

Role Data

```
INSERT INTO Role (RoleID, RoleName, RoleDescription) VALUES
('R001', 'Teacher', 'Instructor responsible for course teaching'),
('R002', 'Student', 'Enrolled in courses');
```

User Data

```
INSERT INTO [User] (UserID, UserName, UserPass, UserLastName, UserFirstName, UserEmail, UserDOB, UserUniID, RoleID, DeptID) VALUES
('U001', 'teacher1', 'teacherpass1', 'Smith', 'John', 'teacher1@email.com', '1980-05-15', 'UNI123', 'R001', '101'),
('U002', 'teacher2', 'teacherpass2', 'Johnson', 'Emma', 'teacher2@email.com', '1975-10-18', 'UNI456', 'R001', '102'),
('U003', 'student1', 'studentpass1', 'Williams', 'Michael', 'student1@email.com', '1999-08-10', 'UNI789', 'R002', '101'),
('U004', 'student2', 'studentpass2', 'Brown', 'Alice', 'student2@email.com', '2000-02-20', 'UNI246', 'R002', '102');
```

Department Data

```
INSERT INTO Department (DeptID, DeptName) VALUES
('101', 'Computer Science'),
('102', 'Data Science');
```

Enroll Data

Project's Github: https://github.com/koitran14/Course-Management-Project.git

```
INSERT INTO Enroll (EnrollID, EnrollDate, UserID, CourseID) VALUES
('E001', '2023-01-15', 'U003', 'C001'),
('E002', '2023-02-20', 'U003', 'C002'),
('E003', '2023-03-10', 'U003', 'C003'),
('E004', '2023-04-05', 'U003', 'C004'),
('E005', '2023-04-15', 'U003', 'C005'),
('E006', '2023-01-25', 'U004', 'C011'),
('E007', '2023-02-28', 'U004', 'C012'),
('E008', '2023-03-15', 'U004', 'C013'),
('E009', '2023-04-20', 'U004', 'C014'),
('E010', '2023-05-05', 'U004', 'C015');
```

Course Data

```
INSERT INTO Course (CourseID, CourseName, UserID, DeptID) VALUES
('C001', 'Introduction to Programming', 'U001', '101'),
('C002', 'Algorithms', 'U001', '101'),
('C003', 'Data Structures', 'U001', '101'),
('C004', 'Database Management', 'U001', '101'),
('C005', 'Machine Learning Basics', 'U001', '101'),
('C006', 'Web Development', 'U001', '101'),
('C007', 'Computer Networks', 'U001', '101'),
('C008', 'Software Engineering', 'U001', '101'),
('C009', 'Cybersecurity', 'U001', '101'),
('C010', 'Operating Systems', 'U001', '101');
```

```
INSERT INTO Course (CourseID, CourseName, UserID, DeptID) VALUES
('C011', 'Data Mining', 'U002', '102'),
('C012', 'Big Data Analytics', 'U002', '102'),
('C013', 'Artificial Intelligence', 'U002', '102'),
('C014', 'Deep Learning', 'U002', '102'),
('C015', 'Natural Language Processing', 'U002', '102'),
('C016', 'Predictive Analytics', 'U002', '102'),
('C017', 'Computer Vision', 'U002', '102'),
('C018', 'Statistical Analysis', 'U002', '102'),
('C019', 'Cloud Computing', 'U002', '102'),
('C020', 'Quantum Computing', 'U002', '102');
```

Content Data

Project's Github: https://github.com/koitran14/Course-Management-Project.git

```
INSERT INTO Content (ConID, ConTitle, ConDesc, ConDate, CourseID) VALUES
('CO001', 'Variables in Programming', 'Introduction to variables in programming languages', '2023-01-20', 'C001'),
('CO002', 'Loop Structures', 'Understanding loop structures in programming', '2023-02-05', 'C001'),
('CO003', 'Functions and Methods', 'Exploring functions and methods in programming', '2023-02-18', 'C001'),
('CO004', 'Sorting Algorithms', 'Understanding various sorting algorithms', '2023-02-27', 'C002'),
('CO005', 'Graph Theory', 'Introduction to graph theory', '2023-03-10', 'C002'),
('CO006', 'Dynamic Programming', 'Understanding dynamic programming concepts', '2023-03-20', 'C002'),
('CO007', 'Stacks and Queues', 'Working with stacks and queues in data structures', '2023-03-25', 'C003'),
('CO008', 'Binary Trees', 'Understanding binary trees', '2023-04-05', 'C003'),
('CO009', 'Hash Tables', 'Working with hash tables in data structures', '2023-04-15', 'C003')
;
```

Announcement Data

```
INSERT INTO Announcement (AnID, AnTitle, AnDesc, AnDate, CourseID) VALUES
('A001', 'Assignment 1 Details', 'Details about the first assignment', '2023-01-25', 'C001'),
('A002', 'Midterm Exam Schedule', 'Details about the midterm exams', '2023-03-01', 'C001'),
('A003', 'Project Submission Guidelines', 'Guidelines for the project submission', '2023-03-20', 'C001'),
('A004', 'Assignment 2 Information', 'Details about the second assignment', '2023-02-10', 'C002'),
('A005', 'Midterm Quiz Schedule', 'Details about the midterm quiz', '2023-03-05', 'C002'),
('A006', 'Project Proposal Submission', 'Submission guidelines for the project proposal', '2023-03-25', 'C002'),
('A007', 'Assignment 3 Details', 'Details about the third assignment', '2023-03-15', 'C003'),
('A008', 'Final Exam Schedule', 'Details about the final exam', '2023-04-10', 'C003'),
('A009', 'Project Presentation Guidelines', 'Guidelines for the project presentation', '2023-04-20', 'C003');
```

Assignment Data

```
INSERT INTO Assignment (A_ID, A_Title, A_Desc, A_StartAt, A_DueDate, CourseID) VALUES
('AS001', 'Variables Exercise', 'Exercise on variables', '2023-01-25', '2023-02-05', 'C001'),
('AS002', 'Loop Structures Task', 'Task on loop structures', '2023-02-05', '2023-02-15', 'C001'),
('AS003', 'Functions Quiz', 'Quiz on functions and methods', '2023-02-20', '2023-02-28', 'C001'),
('AS004', 'Sorting Algorithms Project', 'Project on sorting algorithms', '2023-02-28', '2023-03-15', 'C002'),
('AS005', 'Graph Theory Assignment', 'Assignment on graph theory', '2023-03-10', '2023-03-25', 'C002'),
('AS006', 'Dynamic Programming Task', 'Task on dynamic programming concepts', '2023-03-20', '2023-04-05','C002'),
('AS007', 'Stacks and Queues Exercise', 'Exercise on stacks and queues', '2023-03-25', '2023-04-10', 'C003'),
('AS008', 'Binary Trees Quiz', 'Quiz on binary trees', '2023-04-05', '2023-04-20', 'C003'),
('AS009', 'Hash Tables Project', 'Project on hash tables', '2023-04-15', '2023-04-30', 'C003')
```

3.4. *Database Queries:*

a. Queries analysis

Table 5. SQL keywords

Keyword	Function
SELECT	**Retrieves** data from one or more **tables** in a database.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

JOIN	**Combines** rows from two or more **table**s based on a **related column** between them.
DELETE	**Removes** one or more **rows** from a **table**.
CONVERT	**Converts** a value from one **data type** to another.
DATEDIFF	**Calculates** the difference between two **dates** or **times**.
INSERT INTO VALUES	**Adds** one or more **records** to a **table**.
AND	**Combines** multiple **conditions** in a **WHERE** clause, ensuring that all conditions must be true.
DECLARE	**Defines** a **variable** in SQL.
UPDATE SET	**Modifies data** in a **table** by **updating existing records**.
SELECT DISTINCT	**Retrieves unique values** for a specified **column** or **columns**.
FROM	**Specifies** the **source table** or **tables** for a **query**.
WHERE	**Filters** the **result** set based on a **specified condition**.
INNER JOIN	**Returns** only the **rows** that have **matching values** in **both tables**.
JOIN	**Combines rows** from **two or more tables**
ORDER BY	**Sorts** the **result** set in **ascending** or **descending order** based on one or more **columns**.
AS	**Renames** a **column** or **table** using an **alias**.

b. MVC pattern:

MVC is a design pattern used to decouple user-interface (view), data (model), and application logic (controller). This pattern helps to achieve separation of concerns.

Using the MVC pattern for websites, requests are routed to a Controller that is responsible for working with the Model to perform actions and/or retrieve data. The Controller chooses the View to display and provides it with the Model. The View renders the final page, based on the data in the Model.

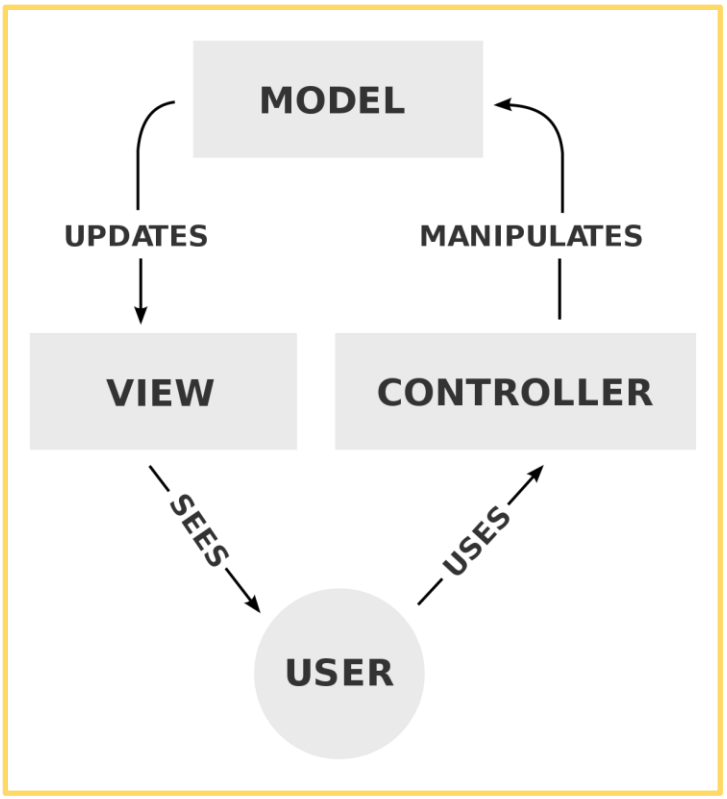

Figure 2. Diagram of interactions within one possible take on the MVC pattern

Project's Github: https://github.com/koitran14/Course-Management-Project.git

In this project,

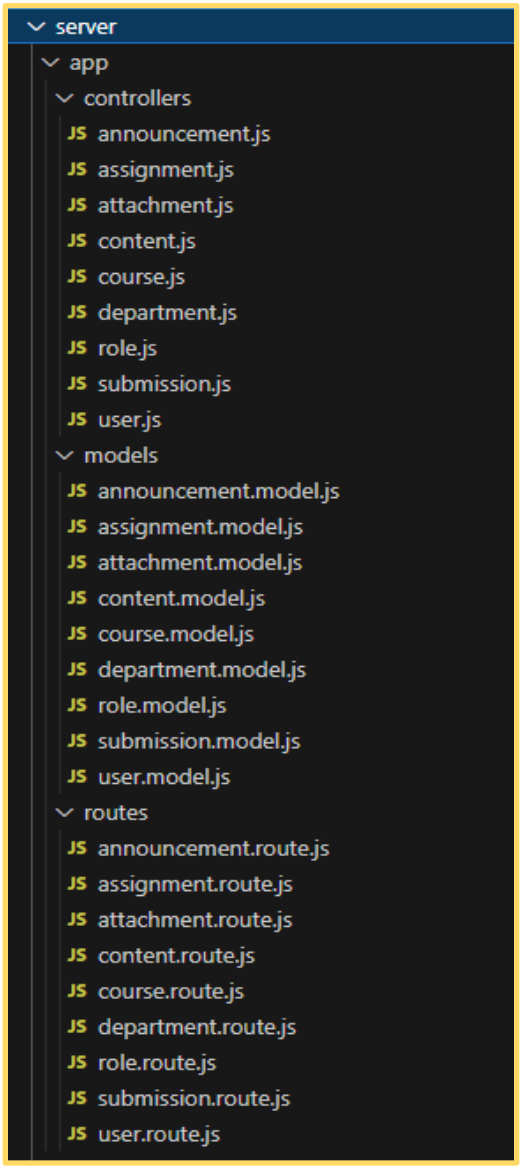

Figure 3. Project's server structure

Model (*.model.js*):

The classes define methods to interact with the data in the database.

It uses the conn object from ../../connect for database connection and performs CRUD (Create, Read, Update, Delete) operations.

Methods like getAll, getByID, getAllByUserID, getAllByCourseID, create, update, and delete,etc. handle the database interactions.

Below are code snippets taken as examples extracted from .model.js files.

```
const { conn, sql } = require('../../connect');

module.exports = class Announcement {
    async getAll(result) {
        var pool = await conn;
        var sqlString = 'Select * from Announcement';
        return await pool.request()
        .query(sqlString, function(err, data){
            if (data.recordset.length > 0){
                result(null, data.recordset);
            } else {
                result (true, null);
            }
        })
    }
```

From announcement.model.js

Project's Github: https://github.com/koitran14/Course-Management-Project.git

```
const { conn, sql } = require('../../connect');

module.exports = class Assignment {
    async getAll(result) {
        var pool = await conn;
        var sqlString = 'Select * from Assignment';
        return await pool.request()
        .query(sqlString, function(err, data){
            if (data.recordset.length > 0){
                result(null, data.recordset);
            } else {
                result (true, null);
            }
        })
    }
```

From assignment.model.js

```
async getByID(id, result) {
    var pool = await conn;
    var sqlString = "Select * from Role Where RoleID = @varID";
    return await pool.request()
    .input('varID', sql.NVarChar(25), id)
    .query(sqlString, function(err, data){
        if (data.recordset.length > 0){
            result(null, data.recordset[0]);
        } else {
            result (true, null);
        }
    })
}
```

From role.model.js

View (*.route.js*):

The routes defined in *.routes.js* act as the views in the MVC pattern.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

Each route corresponds to a specific action, such as getting all announcements, getting a course by ID, creating an assignment, etc.

These routes call the corresponding controller functions, passing control to the controller.

Below are code snippets taken as examples extracted from .route.js files.

```
const { getAllContent, getContentByID,getAllContentsByCourseID, createContent, updateContent, deleteContentByID } = require("../controllers/content");

module.exports = function(app){
    app.get('/api/content', getAllContent);

    app.get('/api/content/:id', getContentByID);

    app.get('/api/:id/content/', getAllContentsByCourseID);

    app.post('/api/content', createContent);

    app.put('/api/content/:id', updateContent);

    app.delete('/api/content/:id', deleteContentByID)

}
```

From content.route.js

```
const { getAllUsers, getUserByID, createUser, updateUser, deleteUserByID, getUserByUserName } = require("../controllers/user");

module.exports = function(app){
    app.get('/api/user', getAllUsers);

    app.get('/api/user/:id', getUserByID);

    app.get('/api/user/username/:name', getUserByUserName);

    app.post('/api/user', createUser);

    app.put('/api/user/:id', updateUser);

    app.delete('/api/user/:id', deleteUserByID)

}
```

From user.route.js

Controller (*announcement.controller.js*):

The controller functions in *.controller.js* handle the business logic and interact with the model to fetch or modify data.

Functions like getAllAnnouncement, getContentByID, etc., are responsible for handling HTTP requests, calling the corresponding model methods, and sending a response back to the client.

Below are code snippets taken as examples extracted from .controller.js files.

```
const Announcement = require('../models/announcement.model');
const model = new Announcement();

exports.getAllAnnouncement = async (req, res) => {
    model.getAll((err, data) => {
        res.send({result: data, error: err})
    })
};
```

From announcement.controller.js

```
const Course = require('../models/course.model');
const model = new Course();

exports.getAllCourses = async (req, res) => {
  model.getAll((err, data) => {
    res.send({result: data, error: err})
  })
}
```

From course.controller.js

c. All Queries used:

Table 6. SQL Queries

MODEL	ACTIONS	SQL QUERIES
ANNOUNCEMENT	*Get All Announcements*	SELECT * FROM Announcement;
	Get Announcement by ID	SELECT * FROM Announcement WHERE AnID = @varID;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Get All Announcements by UserID	SELECT A.* FROM Announcement A JOIN Enroll E ON A.CourseID = E.CourseID WHERE E.UserID = @varID;
	Get All Announcements by CourseID	SELECT *, DATEDIFF(day, GETDATE(), AnDate) AS DayDiff FROM Announcement WHERE CourseID = @varID ORDER BY DayDiff DESC;
	Create Announcement	INSERT INTO Announcement(AnID, AnTitle, AnDesc, AnDate, CourseID) VALUES(@AnID, @AnTitle, @AnDesc, @AnDate, @CourseID);
	Update Announcement	UPDATE Announcement SET AnTitle = @AnTitle, AnDesc = @AnDesc, AnDate = @AnDate WHERE AnID = @AnID;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Delete Announcement by ID	DECLARE @AnIDToDelete VARCHAR(25); SET @AnIDToDelete = @id; DELETE FROM AnnouncementAttachment WHERE AnID = @id; DELETE FROM Attachment WHERE AttachID IN (SELECT AttachID FROM AnnouncementAttachment WHERE AnID = @id); DELETE FROM Announcement WHERE AnID = @id;
ASSIGNMENT	*Get All Assignments*	SELECT * FROM Assignment;
	Get Assignment by ID	SELECT * FROM Assignment WHERE A_ID = @varID;

	Get Near Assignments By UserID	SELECT Assignment.*, CONVERT(VARCHAR, A_DueDate, 103) AS FormattedDueDate,DATEDIFF(DAY, GETDATE(), A_DueDate) AS DaysLeft FROM Assignment WHERE Assignment.CourseID = @varID AND DATEDIFF(DAY, GETDATE(), A_DueDate) < 4 AND DATEDIFF(DAY, GETDATE(), A_DueDate) > 0 ORDER BY A_DueDate ASC;

	Get Near Assignments By CourseID	SELECT Assignment.*, CONVERT(VARCHAR, A_DueDate, 103) AS FormattedDueDate,DATEDIFF(DAY, GETDATE(), A_DueDate) AS DaysLeft FROM Assignment WHERE Assignment.CourseID = @varID AND DATEDIFF(DAY, GETDATE(), A_DueDate) < 4 AND DATEDIFF(DAY, GETDATE(), A_DueDate) > 0 ORDER BY A_DueDate ASC;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Get All Assignments by UserID	SELECT DISTINCT Assign.* FROM Assignment Assign JOIN Course ON Assign.CourseID = Course.CourseID JOIN Enroll ON Course.CourseID = Enroll.CourseID JOIN [User] ON Enroll.UserID = [User].UserID WHERE [User].UserID = @varID AND DATEDIFF(DAY, GETDATE(), Assign.A_DueDate) > 0;
	Get All Assignments by CourseID	SELECT * FROM Assignment WHERE CourseID = @varID ORDER BY DATEDIFF(DAY, GETDATE(), A_DueDate) DESC;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Create Assignment	INSERT INTO Assignment(A_ID, A_Title, A_Desc, A_StartAt, A_DueDate, CourseID) VALUES(@A_ID, @A_Title, @A_Desc, @A_StartAt, @A_DueDate, @CourseID);
	Update Assignment	UPDATE Assignment SET A_Title = @A_Title, A_Desc = @A_Desc, A_StartAt = @A_StartAt, A_DueDate= @A_DueDate WHERE A_ID = @A_ID;
	Delete Assignment by ID	DELETE FROM AssignmentSubmission WHERE DoAssignmentID IN (SELECT DoAssignmentID FROM DoAssignment WHERE A_ID = @id); DELETE FROM Attachment WHERE AttachID IN (SELECT AttachID FROM AssignmentAttachment WHERE A_ID = @id);
ATTACHMENT	*Get All Attachments*	SELECT * FROM Attachment;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Get Attachment by ID	SELECT * FROM Attachment WHERE AttachID = @varID;
	Get Attachments by Announcement ID	SELECT Attachment.*
	Get Attachments by Assignment ID	SELECT Attachment.* FROM Attachment INNER JOIN AssignmentAttachment ON Attachment.AttachID = AssignmentAttachment.AttachID WHERE AssignmentAttachment.A_ID = @varA_ID;
	Get Attachments by Content ID	SELECT Attachment.* FROM Attachment JOIN ContentAttachment ON Attachment.AttachID = ContentAttachment.AttachID WHERE ContentAttachment.ConID = @varConID;

	Get Attachments by Submission ID (UserID and A_ID)	SELECT Attachment.* FROM Attachment INNER JOIN AssignmentSubmission ON Attachment.AttachID = AssignmentSubmission.AttachID INNER JOIN DoAssignment ON AssignmentSubmission.DoAssignmentID = DoAssignment.DoAssignmentID WHERE DoAssignment.UserID = @varUserID AND DoAssignment.A_ID = @varA_ID;
	Create Attachment	INSERT INTO Attachment (AttachID, Attach_FileName, Attach_Date, Attach_URL, CourseID) VALUES (@AttachID, @Attach_FileName, @Attach_Date, @Attach_URL, @CourseID);
	Create Announcement Attachment	INSERT INTO AnnouncementAttachment (AttachID, AnID) VALUES (@AttachID, @AnID);

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Create Assignment Attachment	INSERT INTO AssignmentAttachment (AttachID, A_ID) VALUES (@AttachID, @A_ID);
	Create Content Attachment	INSERT INTO ContentAttachment (AttachID, ConID) VALUES (@AttachID, @ConID);
	Update Announcement Attachment	UPDATE Attachment SET Attach_FileName = @Attach_FileName, Attach_Date = @Attach_Date, Attach_URL = @Attach_URL, CourseID = @CourseID WHERE AttachID = @AttachID; UPDATE AnnouncementAttachment SET AnID = @AnID WHERE AttachID = @AttachID;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Update Assignment Attachment	UPDATE Attachment SET Attach_FileName = @Attach_FileName, Attach_Date = @Attach_Date, Attach_URL = @Attach_URL, CourseID = @CourseID WHERE AttachID = @AttachID; UPDATE AssignmentAttachment SET A_ID = @A_ID WHERE AttachID = @AttachID;
	Update Content Attachment	UPDATE Attachment SET Attach_FileName = @Attach_FileName, Attach_Date = @Attach_Date, Attach_URL = @Attach_URL, CourseID = @CourseID WHERE AttachID = @AttachID; UPDATE ContentAttachment SET ConID = @ConID WHERE AttachID = @AttachID;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Delete Announcement Attachment	DELETE FROM AnnouncementAttachment WHERE AttachID = @AttachID; DELETE FROM Attachment WHERE AttachID = @AttachID;
	Delete Attachment by ID	DELETE FROM Attachment WHERE AttachID = @AttachID;
	Delete Assignment Attachment	DELETE FROM AssignmentAttachment WHERE AttachID = @AttachID; DELETE FROM Attachment WHERE AttachID = @AttachID;
	Delete Content Attachment	DELETE FROM ContentAttachment WHERE AttachID = @AttachID; DELETE FROM Attachment WHERE AttachID = @AttachID;
CONTENT	*Get All Contents*	SELECT * FROM Content;
	Get Content by ID	SELECT * FROM Content WHERE ConID = @varID;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Get All Contents by Course ID	SELECT *, DATEDIFF(DAY, ConDate, GETDATE()) AS DayDifference FROM Content WHERE CourseID = @varID ORDER BY DayDifference ASC;
	Create Content	INSERT INTO Content (ConID, ConTitle, ConDesc, ConDate, CourseID) VALUES (@ConID, @ConTitle, @ConDesc, @ConDate, @CourseID);
	Update Content	UPDATE Content SET ConTitle = @ConTitle, ConDesc = @ConDesc, ConDate = @ConDate WHERE ContentID = @ContentID;
	Delete Content by ID	DELETE FROM ContentAttachment WHERE ConID = @id; DELETE FROM Attachment WHERE AttachID IN (SELECT AttachID FROM ContentAttachment WHERE ConID = @id); DELETE FROM Content WHERE ConID = @id;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

COURSE	*Get All Courses*	SELECT * FROM Course;
	Get Course by ID	SELECT * FROM Course WHERE CourseID = @varID;
	Get All Courses by Teacher ID	SELECT Course.* FROM Course WHERE Course.UserID = @UserID;
	Get All Courses by Student ID	SELECT Course.* FROM Course WHERE Course.UserID = @UserID;
	Create Course	INSERT INTO Course (CourseID, CourseName, UserID, DeptID) VALUES (@CourseID, @CourseName, @UserID, @DeptID);
	Update Course	UPDATE Course SET CourseName = @CourseName, UserID = @UserID, DeptID = @DeptID WHERE CourseID = @CourseID;
	Delete Course	DELETE FROM Course WHERE CourseID = @id;
DEPARTMENT	*Get All Departments*	SELECT * FROM Department;
	Get Department by ID	SELECT * FROM Department WHERE DeptID = @varID;
ROLE	*Get All Roles*	SELECT * FROM Role;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Get Role by ID	SELECT * FROM Role WHERE RoleID = @varID;
	Create Role	SELECT * FROM Role WHERE RoleID = @varID;
	Update Role	UPDATE Role SET RoleName = @RoleName, RoleDescription = @RoleDescription WHERE RoleID = @RoleID;
	DeletE Role	DELETE FROM Role WHERE RoleID = @id;
SUBMISSION	*Get All Submissions*	SELECT * FROM AssignmentSubmission;
	Get All Submissions by Assignment ID	SELECT * FROM DoAssignment WHERE DoAssignment.A_ID = @varID;
	Get All Submissions by User ID	SELECT * FROM AssignmentSubmission WHERE DoAssignmentID IN (SELECT DoAssignmentID FROM DoAssignment WHERE UserID = @varID);

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Get Submission Information by User ID and Assignment ID	SELECT Attachment.AttachID, Attachment.Attach_FileName, Attachment.Attach_Date, Attachment.Attach_URL FROM Attachment JOIN AssignmentSubmission ON AssignmentSubmission.AttachID = Attachment.AttachID JOIN DoAssignment ON DoAssignment.DoAssignmentID = AssignmentSubmission.DoAssignmentID WHERE DoAssignment.A_ID = @varA_ID AND DoAssignment.UserID = @varUserID; SELECT DoAssignment.DoAssignmentID, DoAssignment.A_ID, DoAssignment.UserID, DoAssignment.Grade, DoAssignment.DoAt FROM Attachment JOIN AssignmentSubmission ON AssignmentSubmission.AttachID = Attachment.AttachID JOIN DoAssignment ON

		DoAssignment.DoAssignmentID = AssignmentSubmission.DoAssignmentID WHERE DoAssignment.A_ID = @varA_ID AND DoAssignment.UserID = @varUserID;
	Update Grade	UPDATE DoAssignment SET Grade = @varGrade WHERE A_ID = @varA_ID AND UserID = @varUserID;

	Create Submission	DECLARE @NewDoAssignmentID UNIQUEIDENTIFIER = NEWID(); INSERT INTO DoAssignment (DoAssignmentID, A_ID, UserID, Grade, DoAt) VALUES (@NewDoAssignmentID, @varA_ID, @varUserID, -1.0, @varDate); INSERT INTO Attachment (AttachID, Attach_FileName, Attach_Date, Attach_URL, CourseID) VALUES ('${attachment.AttachID}', '${attachment.Attach_FileName}', '${attachment.Attach_Date}', '${attachment.Attach_URL}', '${attachment.CourseID}'); INSERT INTO AssignmentSubmission (A_SubID, AttachID, DoAssignmentID) VALUES (NEWID(), '${attachment.AttachID}', @NewDoAssignmentID);

	Delete Submission	DECLARE @UserID VARCHAR(25) = @varUserID; DECLARE @AssignmentID VARCHAR(25) = @varA_ID; DECLARE @DoAssignmentID UNIQUEIDENTIFIER; SELECT @DoAssignmentID = DoAssignmentID FROM DoAssignment WHERE A_ID = @AssignmentID AND UserID = @UserID; DELETE FROM AssignmentSubmission WHERE DoAssignmentID = @DoAssignmentID; DELETE FROM DoAssignment WHERE DoAssignmentID = @DoAssignmentID;
USER	*Get All Users*	SELECT * FROM [User];
	Get User by ID	SELECT * FROM [User] WHERE UserID = @varID;
	Get User by UserName	SELECT * FROM [User] WHERE UserName = @name;

Project's Github: https://github.com/koitran14/Course-Management-Project.git

	Create User	INSERT INTO [User] (UserID, UserName, UserPass, UserFirstName, UserLastName, UserEmail, UserDOB, UserUniID, RoleID, DeptID) VALUES (@UserID, @UserName, @UserPass, @UserFirstName, @UserLastName, @UserEmail, @UserDOB, @UserUniID, @RoleID, @DeptID);
	Update User	UPDATE [User] SET UserName = @UserName, UserPass = @UserPass, UserFirstName = @UserFirstName, UserLastName = @UserLastName, UserEmail = @UserEmail, UserDOB = @UserDOB, UserUniID = @UserUniID, DeptID = @DeptID WHERE UserID = @UserID;
	Delete User	DELETE FROM [User] WHERE UserID = @id

Project's Github: https://github.com/koitran14/Course-Management-Project.git

4. Application Project structure

4.1. *Project structure*

Front-end (client):

public: Contains static resources such as images and CSS.

src: Contains the main source code of the front-end application.

components: components.

pages: Main pages of the application.

services: Handles communication with the back-end through APIs.

utils: Contains helper functions and utilities.

package.json: File describing information and dependencies for the front-end.

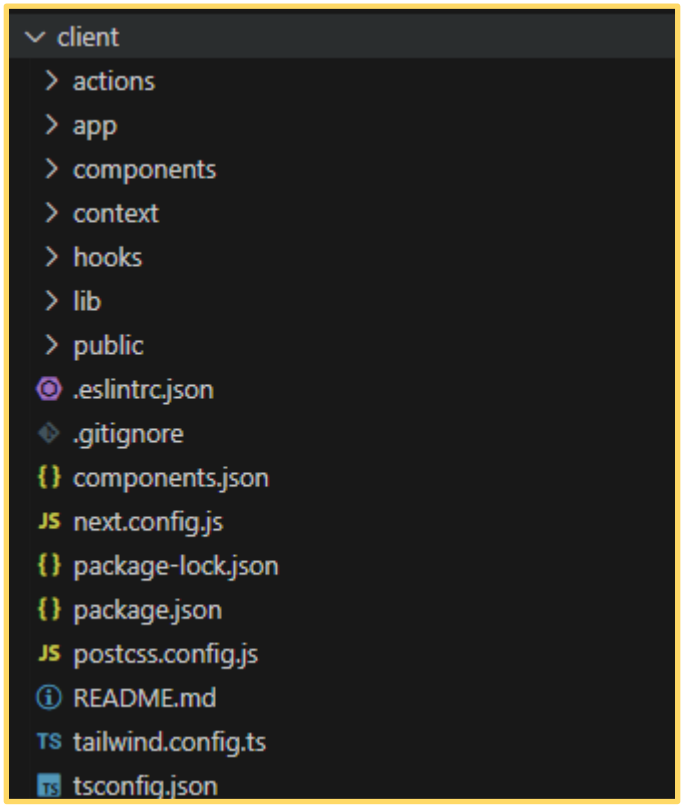

Figure 4. The Project's front-end structures in IDE

Project's Github: https://github.com/koitran14/Course-Management-Project.git

Back-end (server):

controllers: Business logic handling.

models: Defines objects and interacts with the database.

routes: Defines application routes.

server.js: sets up a basic Express.js server for a web application.

connect.js: sets up a connection pool to a Microsoft SQL Server database.

package.json: File describing information and dependencies for the back-end.

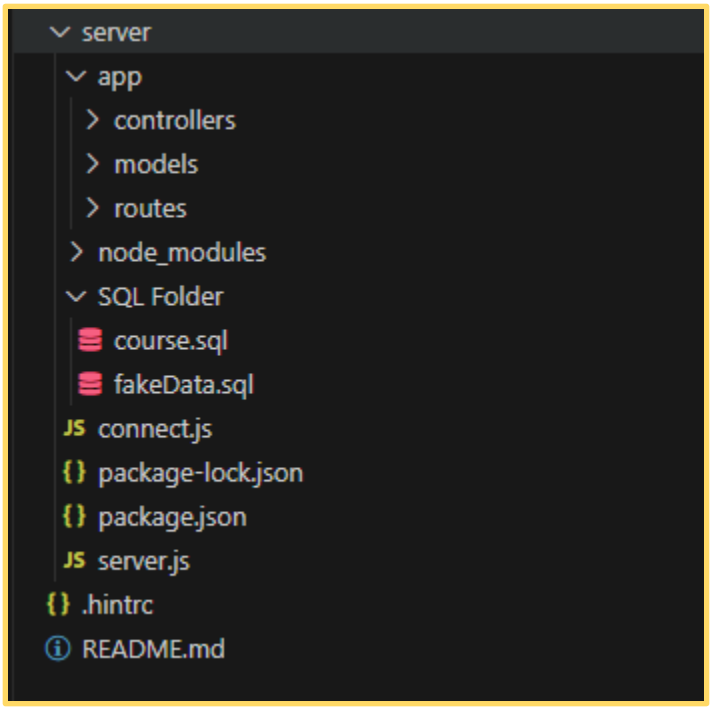

Figure 5. The Project's back-end structures in IDE

4.2. *Connection implementation*

Database Connection Configuration:

Project's Github: https://github.com/koitran14/Course-Management-Project.git

```
const mysql = require('mssql/msnodesqlv8');

var config = {
    server: "LAPTOP-J8Q1OE31\\SQLEXPRESS", //update me
    database: "CourseDB",
    options: {
        synchronize: true,
        trustServerCertificate: true,
        trustedConnection: true,
    },
    driver: "msnodesqlv8"
}
```

This section defines the configuration for the database connection. It includes the server address, database name, and various options related to the connection.

Connection Pool Creation:

```
const myconn = new mysql.ConnectionPool(config).connect().then(pool => {
    return pool;
})
```

This code creates a connection pool using ConnectionPool from the mssql library. The connect() method returns a promise that resolves to the connection pool.

Exporting Connection Pool and SQL Library:

```
module.exports = {
    conn: myconn,
    sql: mysql
}
```

This exports the connection pool (conn) and the SQL library (sql) so that other parts of the application can use them.

Server setup:

The project is a **Node.js** application utilizing **Express.js** to create an **API server**. It also uses the **cors** library to handle Same-Origin Policy issues and **body-parser** to process input data from **clients**.

- **Express.js:** It is a web framework for Node.js that facilitates the creation of web applications and APIs. It simplifies handling HTTP requests, routing, and middleware.
- **Body-parser:** A middleware used to parse input data from clients.
- **Cors (Cross-Origin Resource Sharing):** This is a mechanism to prevent issues when accessing resources from a different domain.

Import Modules:

```
const express = require('express');
const bodyParser = require('body-parser');

const cors = require('cors');
```

- **express:** Web framework for Node.js.
- **body-parser:** Middleware for parsing incoming request bodies.
- **cors:** Middleware for handling Cross-Origin Resource Sharing (CORS) issues.

Create Express Application:

```
const app = express();
```

Create an instance of the Express application.

Set the port for the server to listen on:

```
const PORT = 8080;
```

Define the port on which the server will listen.

CORS Configuration:

```
const corsOptions = {
  origin: 'http://localhost:3000',
  optionsSuccessStatus: 200 // some legacy browsers (IE11, various SmartTVs) choke on 204
};
```

```
app.use(cors(corsOptions));
```


Project's Github: https://github.com/koitran14/Course-Management-Project.git

Configuration object specifying allowed origins and options success status.

Enable CORS middleware with the specified options.

Body Parser Configuration:

```
app.use(bodyParser.json());
```

Enable body-parser middleware to parse incoming JSON data.

Route Configuration:

```
require('./app/routes/role.route')(app);
require('./app/routes/user.route')(app);
require('./app/routes/course.route')(app);
require('./app/routes/announcement.route')(app);
require('./app/routes/assignment.route')(app);
require('./app/routes/content.route')(app);
require('./app/routes/department.route')(app);
require('./app/routes/assignment.route')(app);
require('./app/routes/attachment.route')(app);
require('./app/routes/submission.route')(app);
```

Import and use route files for various endpoints.

Start Server:

```
app.listen(PORT, () => {
  console.log(`Server is running at site: http://localhost:${PORT}`);
});
```

Start the server and log a message when it begins listening.

4.3. *G.U.I Design*

4.3.1 Sketch the UI:

After completely determined the concepts and features of the website, we started having a try on finding and sketching the UI as individual imagination. During this process, Canva is a main platform for us to sketch and design our ideas.

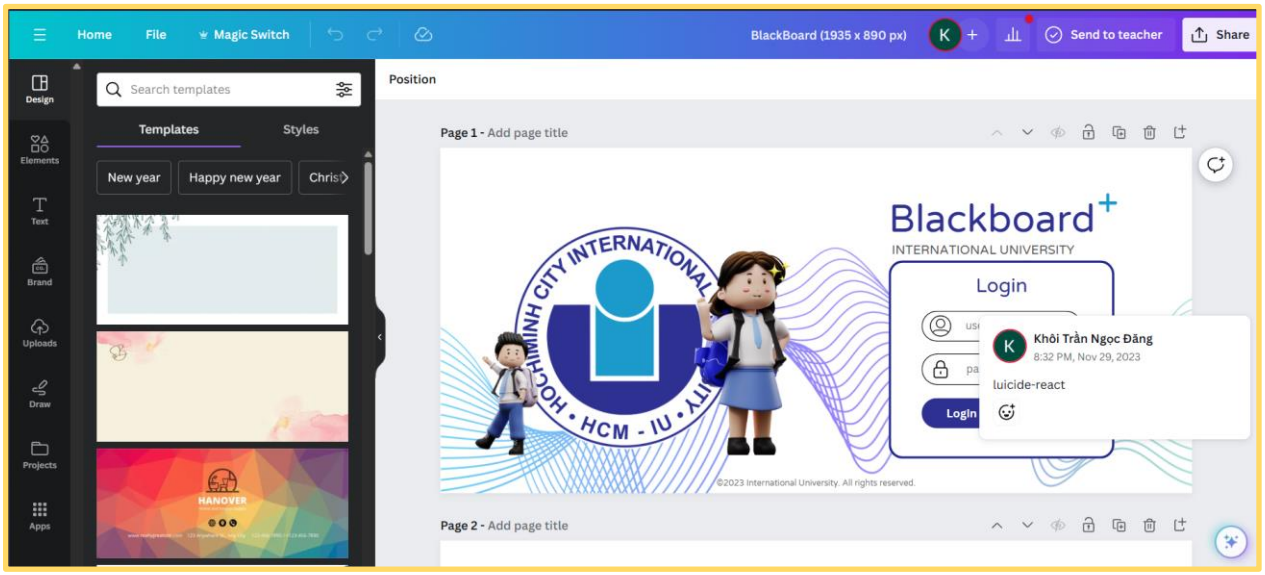

Figure 6. Canva platform and draft of planning and sketching UI process.

4.3.2 Main technique:

In this project, our team implemented Tailwind CSS as the main framework to build and create components. Firstly, we installed the framework in our project.

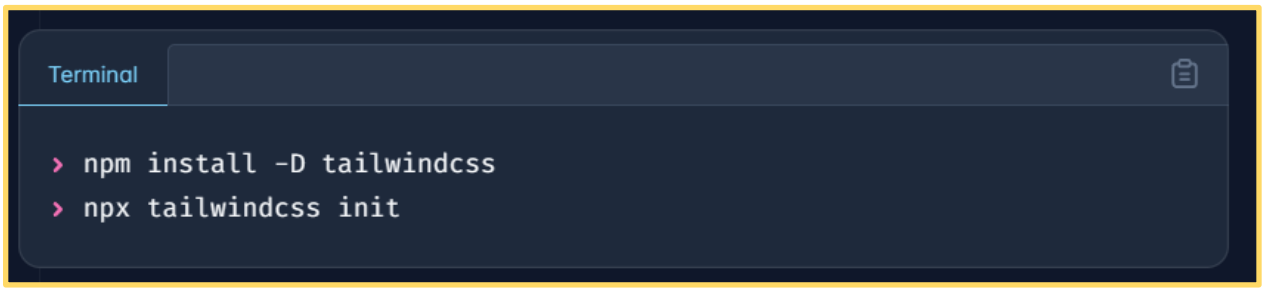

Figure 7. Installation of Tailwind CSS.

Then, we build the UI based on the finalized design using tags provided by Tailwind CSS.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

```
<Button  className="bg-indigo-700 hover:bg-indigo-800" disabled={!subAttachments}>
    Submit
</Button>
```

Figure 8. Example of using Tailwind CSS in building UI by calling its available name tags.

4.4. *Buttons implementation*

To integrate various functionalities, alongside executing SQL queries, buttons serve as the conduit for user interaction.

4.4.1. General:

When constructing the website, buttons are crucial for user actions and interactions within the system. To streamline the management and consistency of all buttons across the application, our approach begins with creating a universal button component. This method significantly improves code organization and cleanliness while implementing UI design, offering an efficient way to call and maintain buttons throughout the system.

To do that, it necessarily inherits and redesigns the `button` component of HTML.Element with some additional attributes and functions.

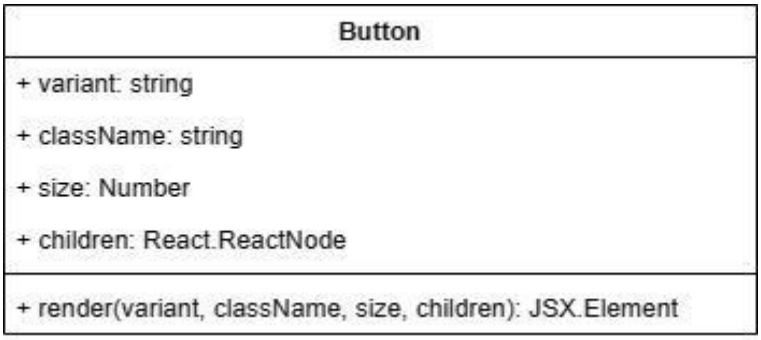

Figure 9. Class diagram of Button component

In 'Button' component, there are totally four attributes with three non-required attributes and the required one:

Table 7. Attributes and functionalities of 'Button' component

Attribute	Type	State	Functionality
variant	String	Optional	Determines the visual variant of buttons. Choices include: 'default', 'destructive', 'outline', 'secondary', 'ghost', 'link'. Each variant applies specific styles to the button, altering its appearance.
size	Number	Optional	Specifies the size of the button. Options include: 'default', 'sm' (small), 'lg' (large), 'icon', 'smallIcon'. Each size variant sets different height, width, and padding values for the button.
className	String	Optional	Additional CSS classes to be applied to the button, allowing for custom styling or extending existing styles.
children	React.ReactNode	Compulsory	The content rendered inside the button component. It can include text, other React components, or elements to be displayed within the button.

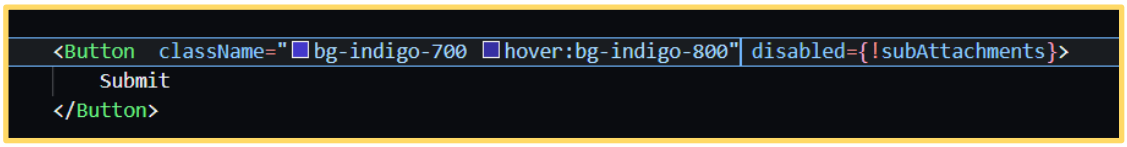

Figure 10. Example of 'Button' component implementation in rendering UI.

4.4.2. Submit buttons

Once the `Button` component is fully constructed, implementing it across various scenarios within the project becomes significantly easier. `Submitting` stands out as the primary functionality in user interactions, warranting initial consideration.

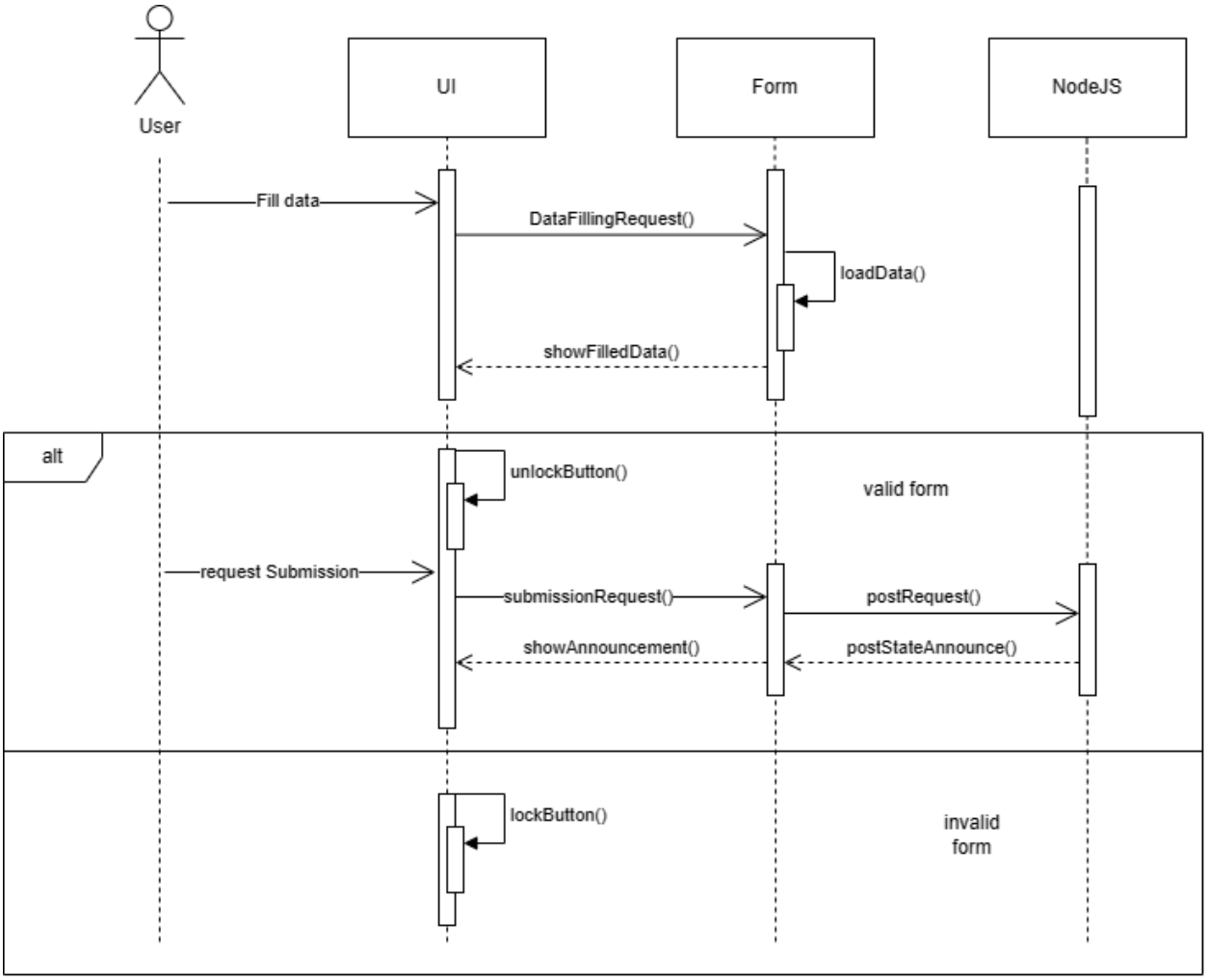

Figure 4.4.2.1. Sequence diagram of `Submit` button in posting form procedure.

The `handleSubmit` function encapsulates the critical logic for form submission within the application. It orchestrates several essential tasks, beginning with a validation check for the form's title, description, and end date. If these fields are valid, the function generates a unique ID for the assignment before attempting to create it via an API call to a local server endpoint.

As the example of *figure 4.4.2.2* below, upon successful creation of the assignment, the function provides visual feedback through a success toast and then navigates back to the previous page while refreshing the router. Additionally, if there are attachments associated with the assignment, each attachment undergoes a two-step process: first, it is created via an API call, and then a reference to this attachment is linked to the assignment.

Should any errors occur during these processes, error messages are displayed via toast notifications to alert the user. It is important to note that the function's design accommodates different form types, each with its own set of SQL queries tailored for SQL Server as pre-mentioned in title *3.4.Database Queries.*

```
const handleSubmit = async () => {
  if (!validTitle || !validDesc || !validEndAt) {
    return;
  }
  const uniqueID = await generateUniqueAssignment();
  const assignmentWithID = {
    ...assignment,
    A_ID: uniqueID,
  };
  try {
    const response = await axios.post('http://localhost:8080/api/assignment', assignmentWithID);
    console.log('Content created:', response.data);
    if (response.status === 200) {
      toast.success('Created successfully.');
      router.back();
      router.refresh();
    } else {
      toast.error("There's something wrong.");
    }

    if (attachments && attachments.length > 0) {
      for (const attachment of attachments) {
        try {
          const attachmentResponse = await axios.post(`${process.env.NEXT_PUBLIC_API_URL}/attachment`, attachment);
          console.log('Attachment created:', attachmentResponse);
        } catch (error) {
          console.error("Error creating attachment:", error)
        }
```

```
        try {
          const data = {
            A_ID: assignmentWithID.A_ID,
            AttachID: attachment.AttachID
          }
          const announcementAttachment = await axios.post(`${process.env.NEXT_PUBLIC_API_URL}/attachment/assignment`, data);
          console.log('Attachment for announcement created:', announcementAttachment);
        } catch (error) {
          console.error("Error creating attachment for assignment:", error)
        }
      }
    }
  } catch (error) {
    toast.error('Error: ' + error);
  }
};
```

Figure 11. Example of `handleSubmit` function under the `Submit` button.

4.4.3. Edit buttons

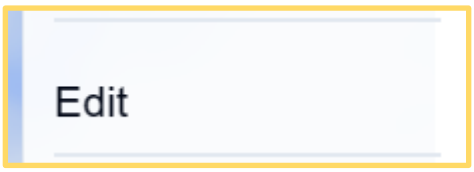

Figure 12. `Edit` navigation button in application

In other cases, `Button` component is also implemented in calling `Edit` function. Similar as `Submit` button, it contains the procedures under the button to process or lead to specific pages to edit. As the given example of figures below, `Edit` button is attached with the link, which is led users to 'Edit' page to fill the required form. When data is all valid, it reuses `submit` button to post and update the new data to the SQL system.

```
const NavigatorForMain = [
    {
        title: 'Profile',
        path: 'profile'
    },
    {
        title: 'Edit',
        path: 'edit'
    }
]
```

Figure 13. Navigation contents were firstly initialized with link (as `path`)

```
<Link href={`/${params.UserID}/${nav.path}`}>
    {nav.title}
</Link>
```

Figure 14. Load the array of Navigation contents with attached links.

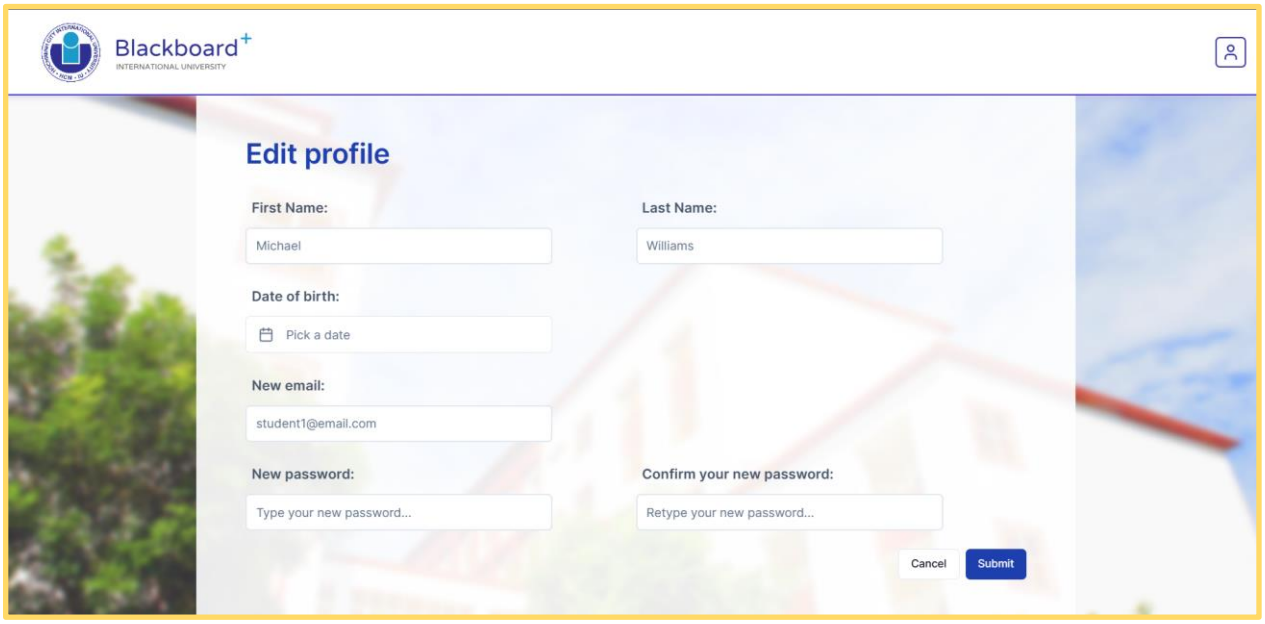

Figure 15. `Edit` form.

As mentioned before, similar with other buttons, it also has its own handler function.

```
const handleFormSubmit = async () => {
  try {
    const updatedUserData: Partial<User> = {
      UserFirstName: updatedUserFirstName || user?.UserFirstName || '',
      UserLastName: updatedUserLastName || user?.UserLastName || '',
      UserEmail: updatedUserEmail || user?.UserEmail || '',
      UserDOB: selectedDate,
    };
    if (newPassword || confirmNewPassword) {
      if (newPassword === confirmNewPassword) {
        updatedUserData.UserPass = newPassword;
      } else {
        setErrorMessage('Password mismatch');
        setSuccessMessage(''); // Clear success message
        return;
      }
    }
```


Project's Github: https://github.com/koitran14/Course-Management-Project.git

```
      const updatedUser = await updateUser(params.UserID as string, updatedUserData);
      if (updatedUser) {
        setUser(updatedUser);
        setSuccessMessage('Profile updated successfully');
        setErrorMessage('');
        console.log('User updated successfully:', updatedUser);
        toast.success("Completely updated.");
        router.refresh();
        router.back();
      } else {
        console.error('Failed to update user');
      }
    } catch (error) {
      console.error('Error updating user:', error);
    }
  };
```

Figure 16. `handleSubmit` function under `Submit` button of `Edit` form.

4.4.4. Cancel buttons

Inherited from the `Button` component, it is upgraded by working with `Router` object and `useRouter`, which are the supplied React hooks of Next.JS, to redirect users back to the previous page by user `back` function.

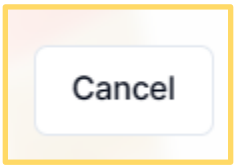

Figure 17. `Cancel` button in application.

As the example of *figure 4.4.4.1* below, it firstly imports the `useRouter` hook from `next/navigation` and initializes the `router` variable as the `Router` object in order to retrieve the function in usage.

```
import { useRouter } from "next/navigation";
```

```
const router = useRouter();
```

Figure 18. Initialization of `useRouter` hook in Next.JS

Project's Github: https://github.com/koitran14/Course-Management-Project.git

Then, the `handleCancel` function is easier to build after `useRouter` hook initialization is completed. As *figure 4.4.4.2* below, `back` function is retrieved directly by using the initialized `router` object from the `useRouter` hook.

```
const handleCancel = () => {
  router.back();
};
```

```
<Button variant="outline" onClick={() => handleCancel()}>
  Cancel
</Button>
```

Figure 19. `handleCancel` function and 'Cancel' button initialization

4.5. *Application Demo – Screenshots*

a. General

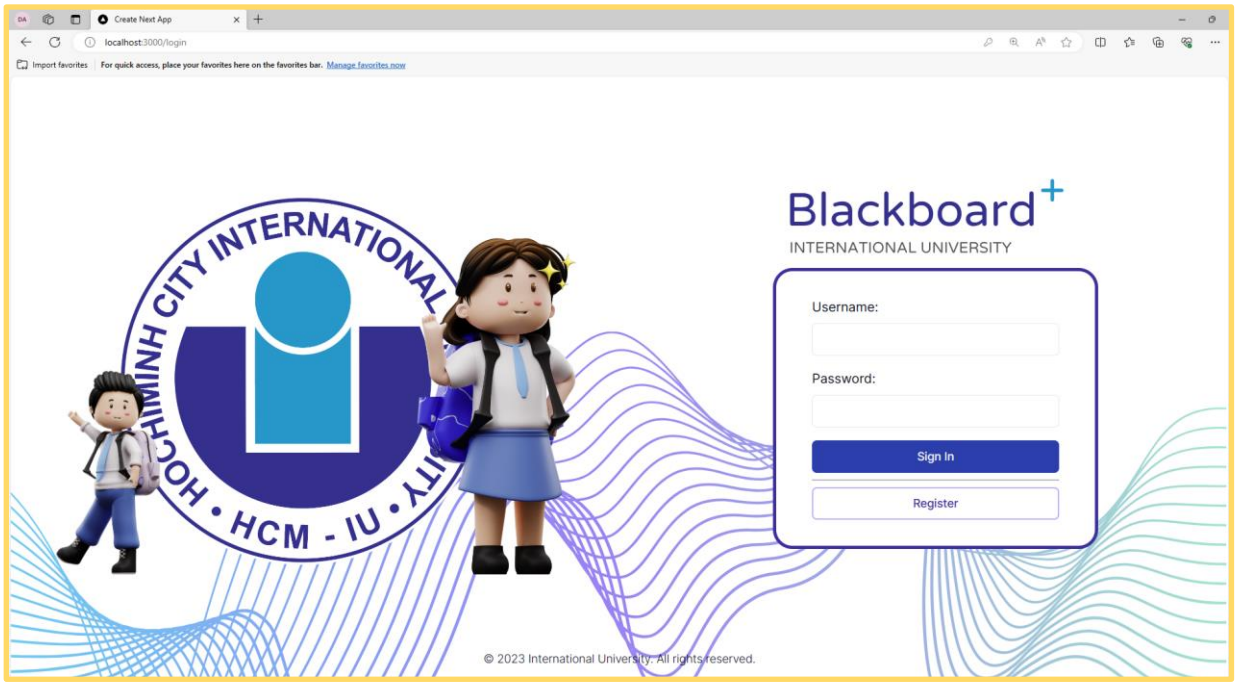

Figure 20. Login page

Project's Github: https://github.com/koitran14/Course-Management-Project.git

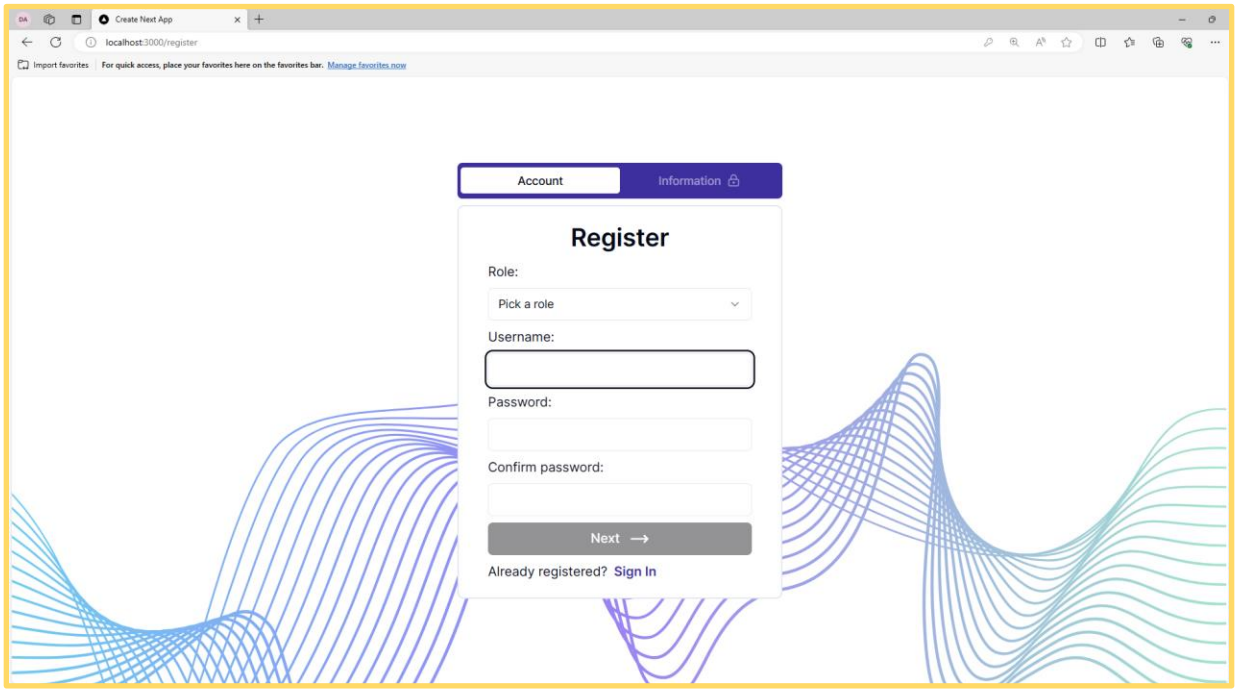

Figure 21. Account creation

Project's Github: https://github.com/koitran14/Course-Management-Project.git

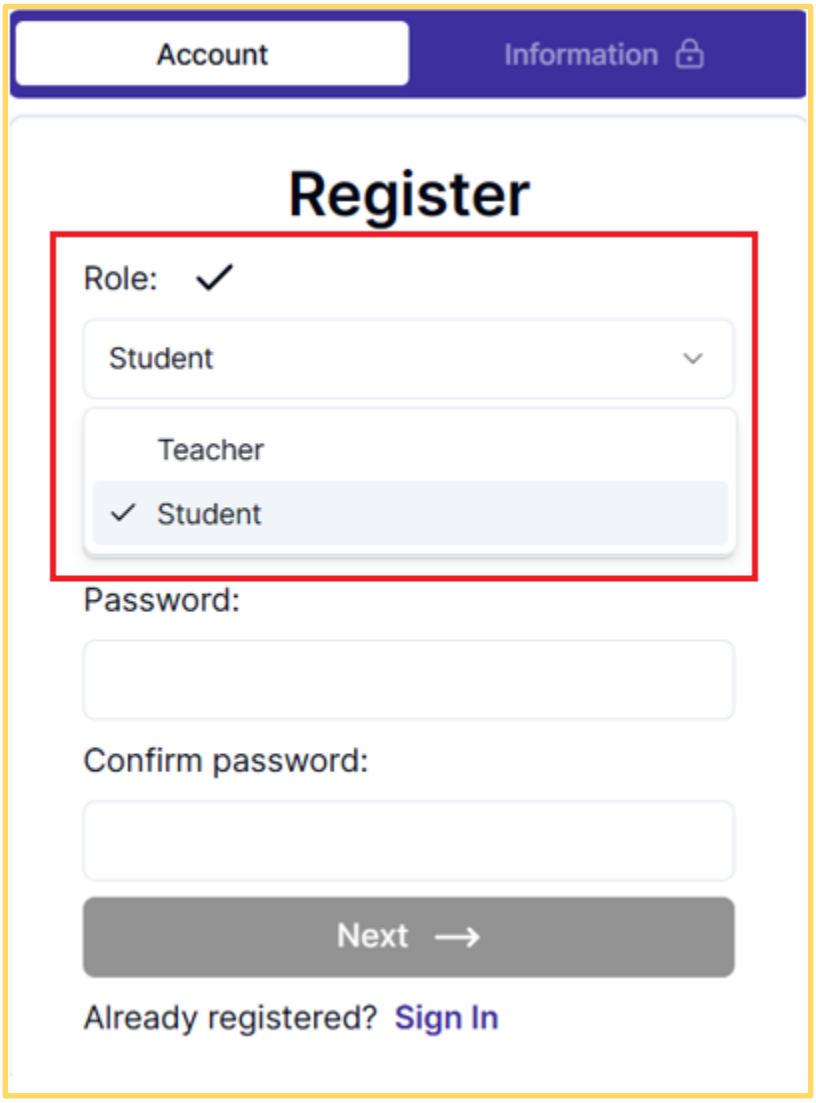

Figure 22. Role choose

The choice of role will determine your interface and permissions throughout the app usage process.

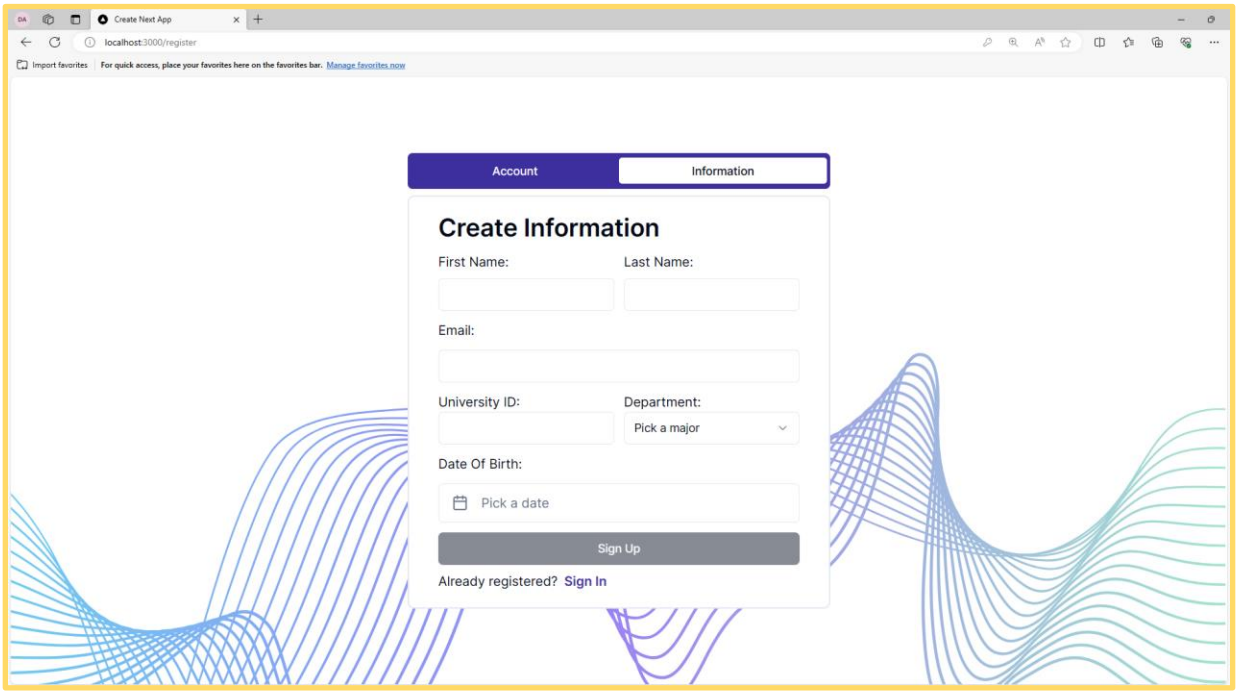

Figure 23. Add information

Project's Github: https://github.com/koitran14/Course-Management-Project.git

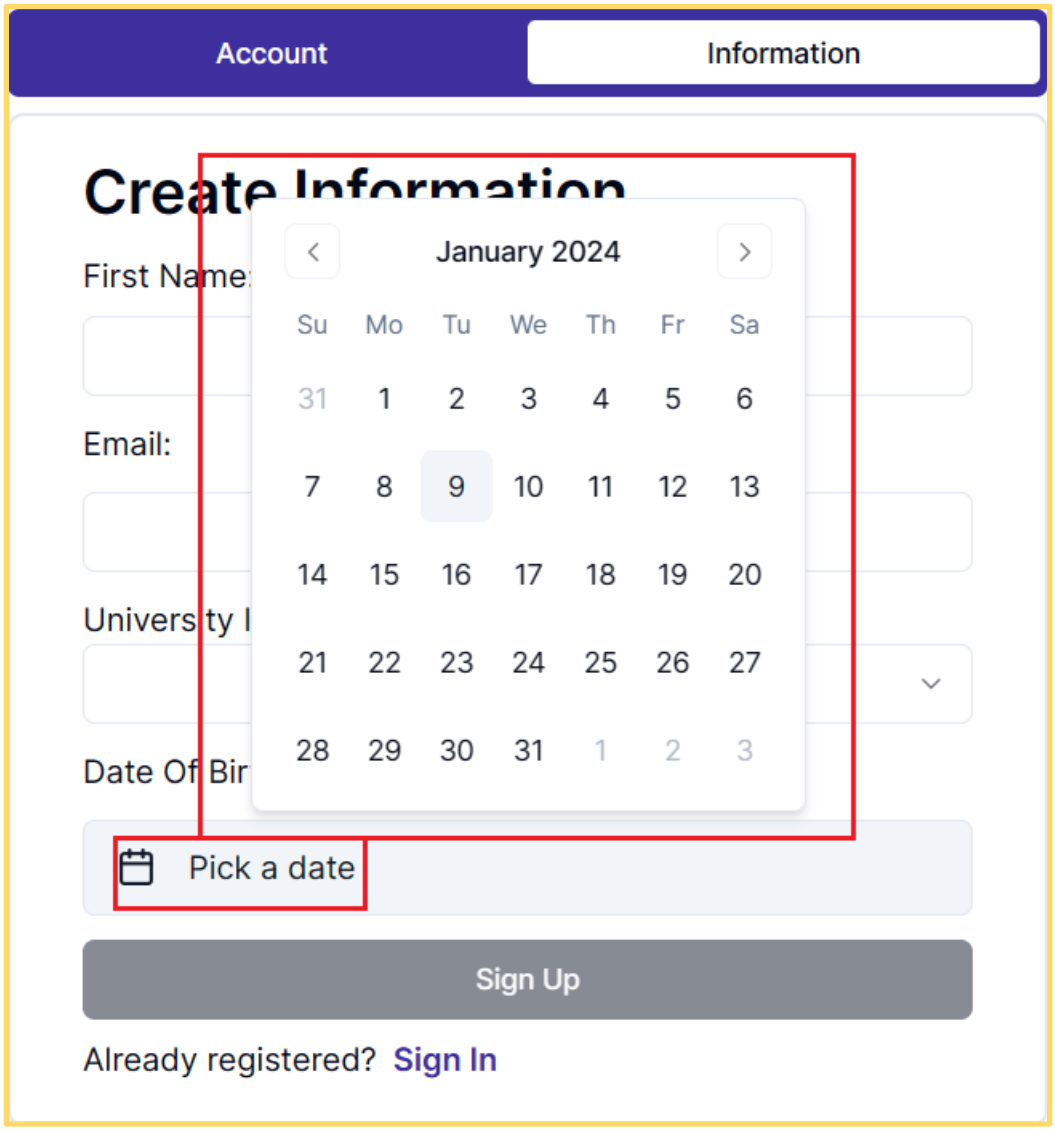

Figure 24. DOB Calendar

Project's Github: https://github.com/koitran14/Course-Management-Project.git

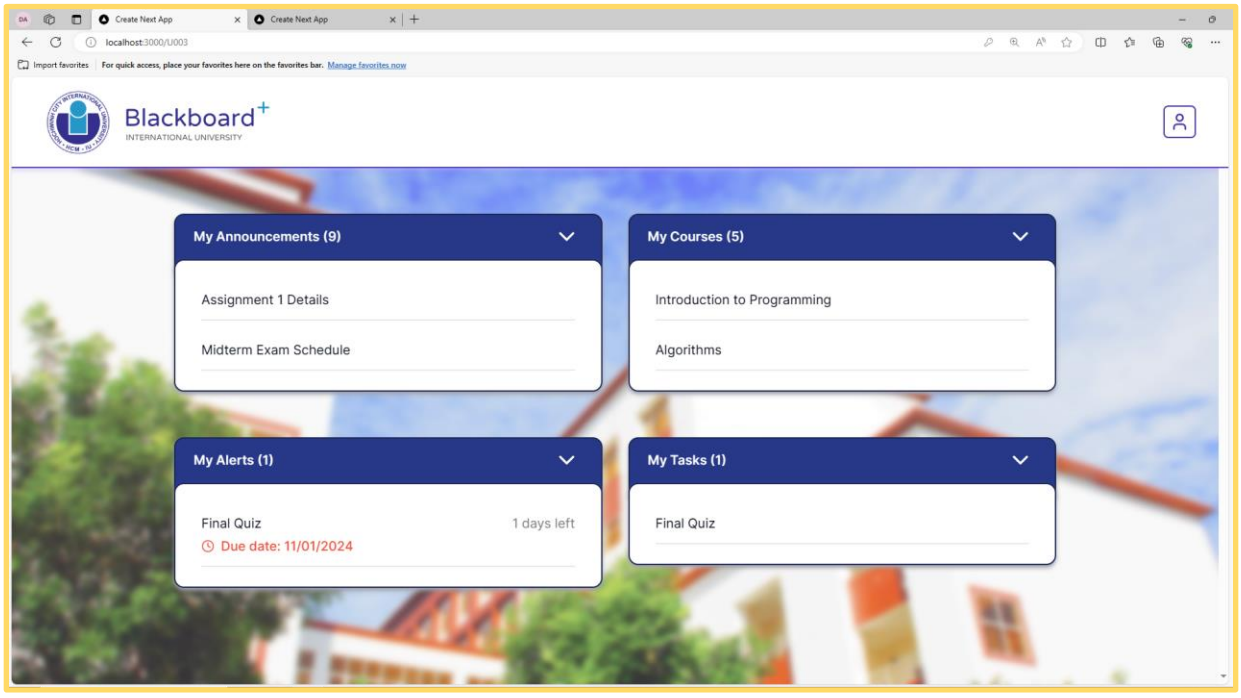

Figure 25. Homepage

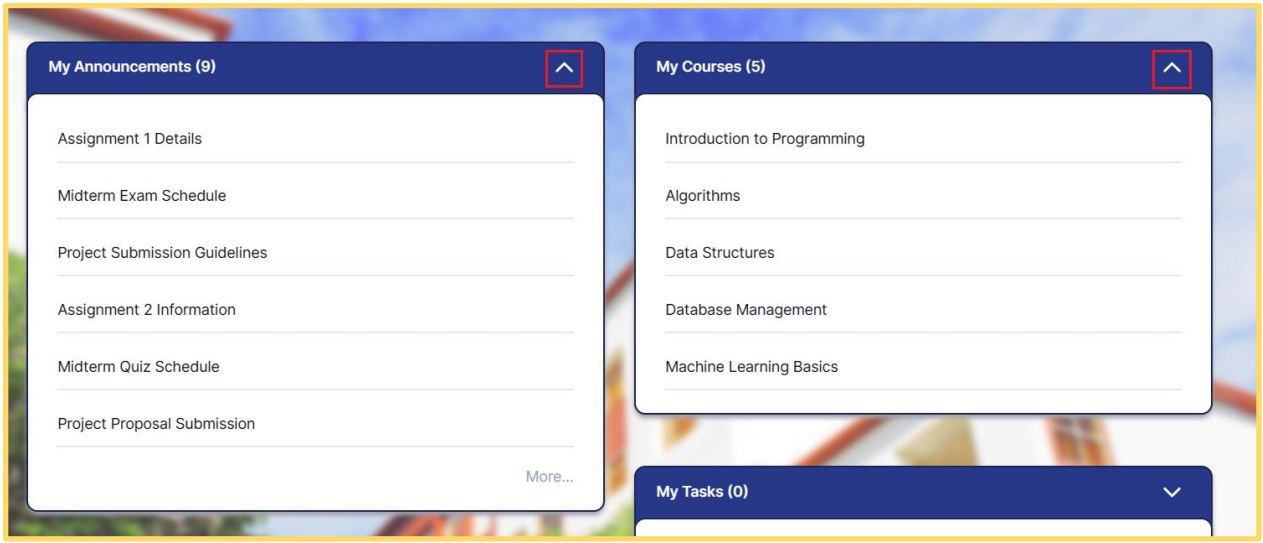

Figure 26. Adjustable Bars

Project's Github: https://github.com/koitran14/Course-Management-Project.git

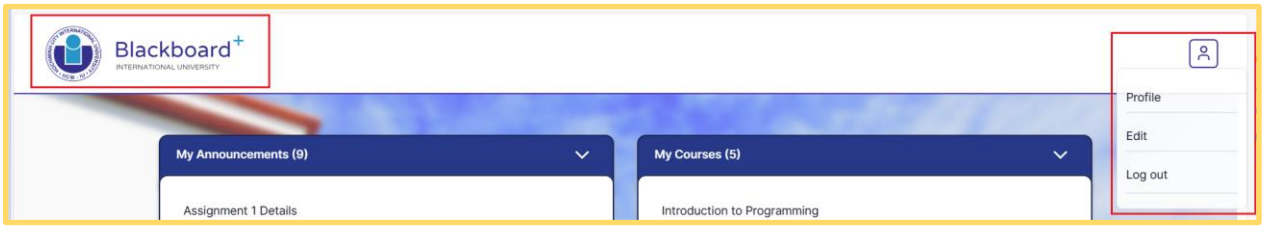

Figure 27. Homepage navigation bar

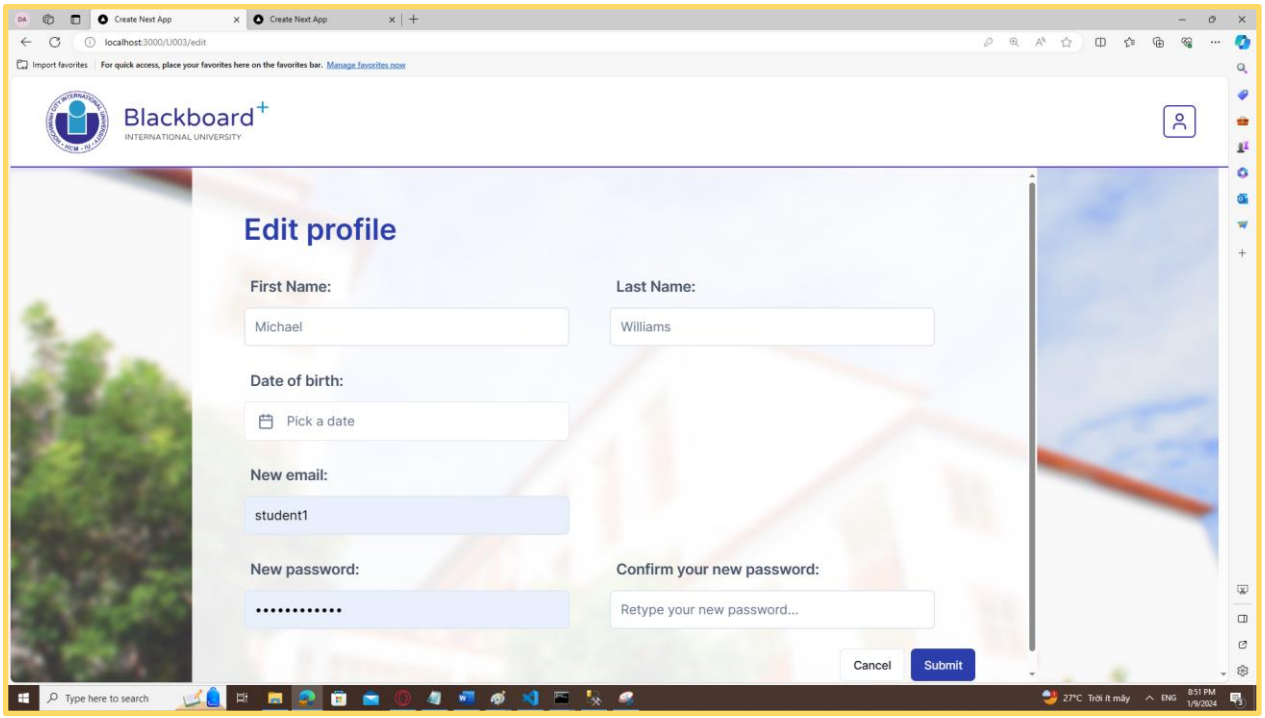

Figure 28. Edit profile page

b. Student mode

Project's Github: https://github.com/koitran14/Course-Management-Project.git

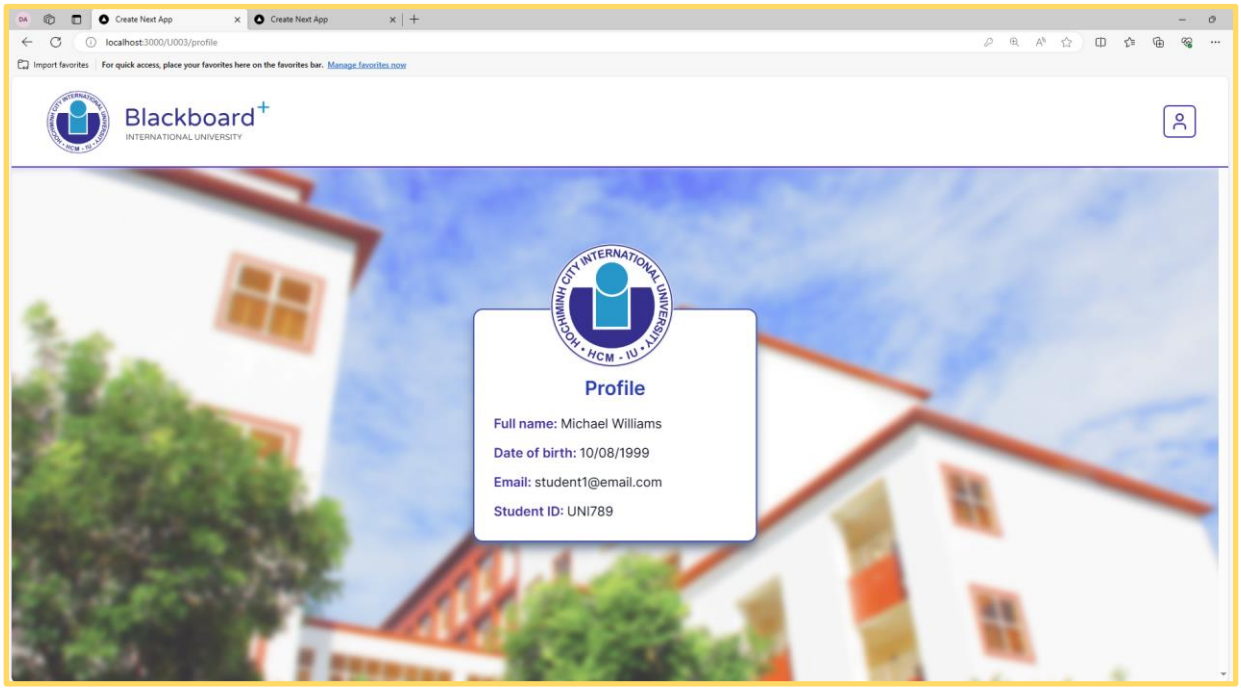

Figure 29. Student's Profile page

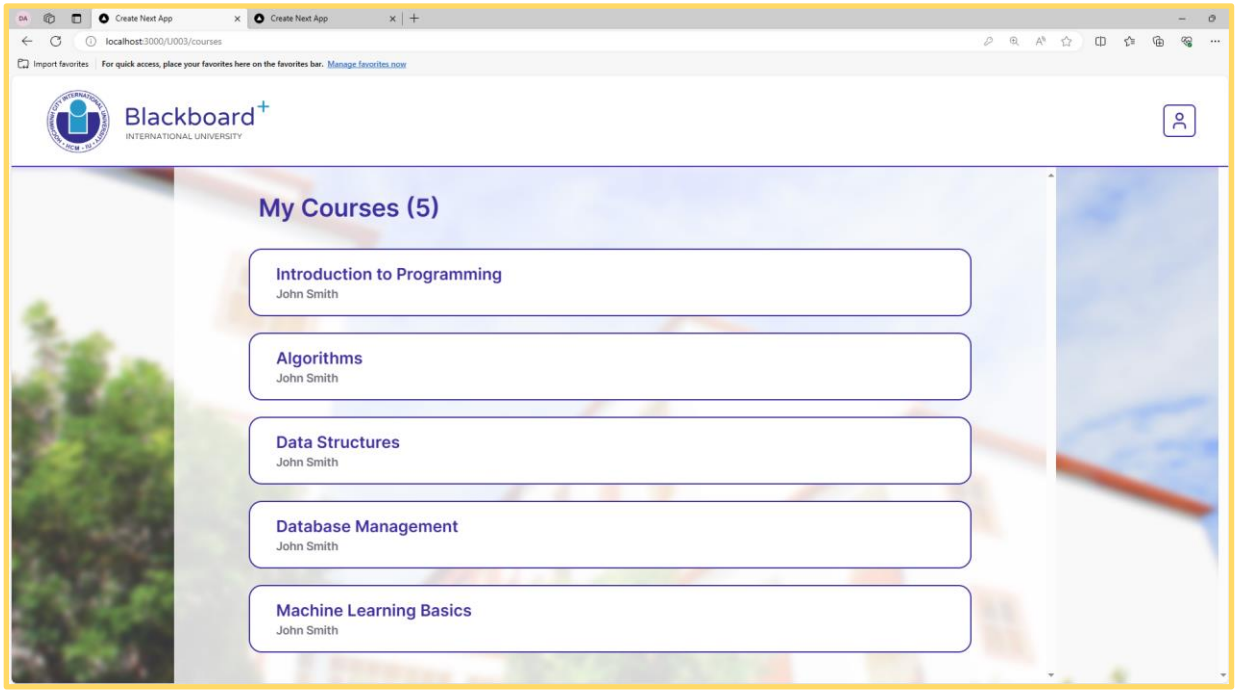

Figure 30. My Courses page

The instructor's name is displayed beneath the course title.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

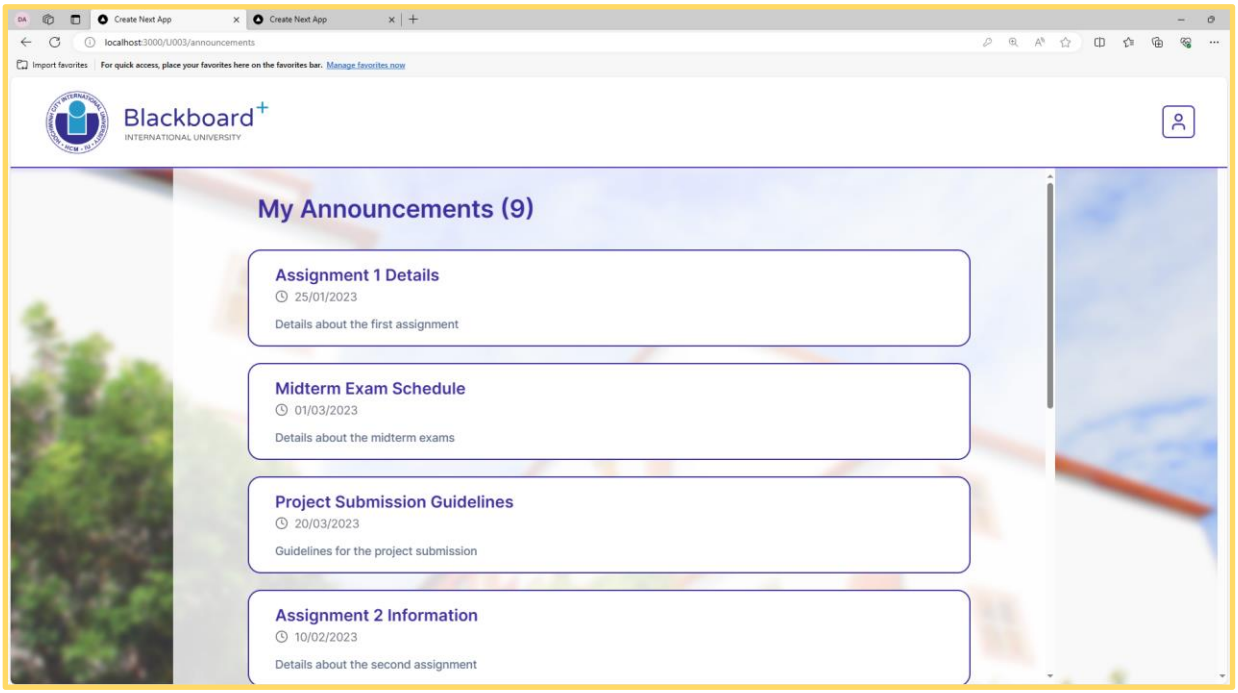

Figure 31. My Announcements page

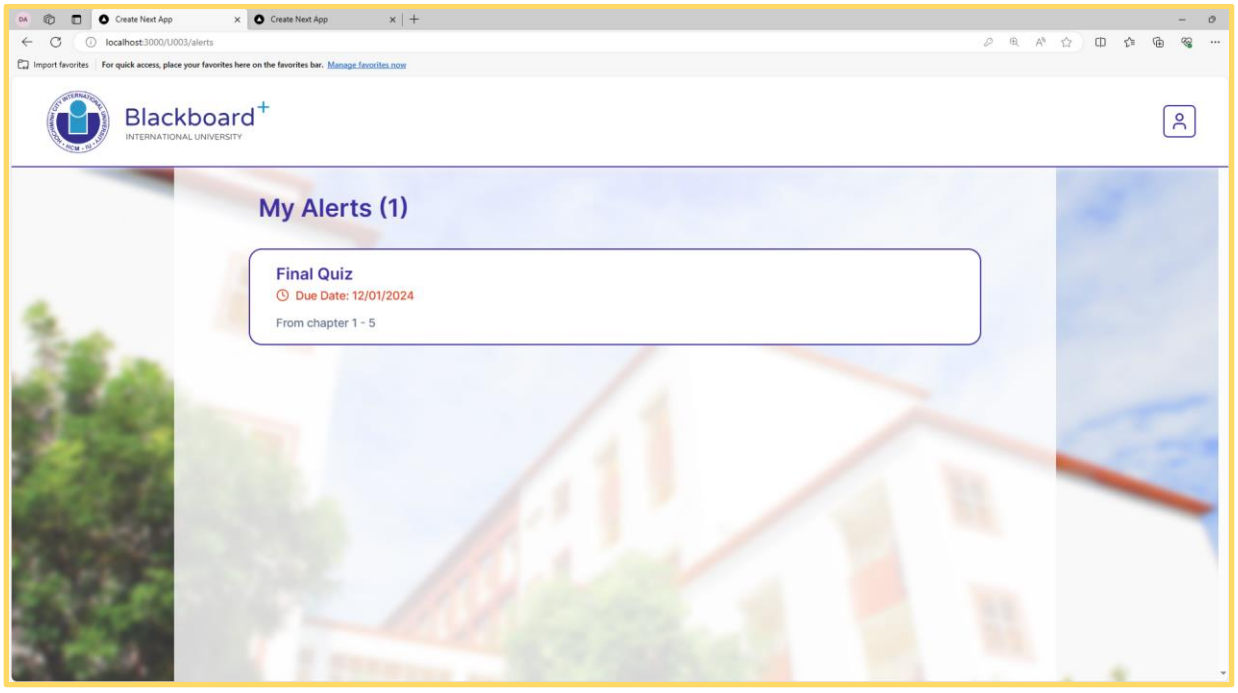

Figure 32. My Alerts page

Project's Github: https://github.com/koitran14/Course-Management-Project.git

Display assignments with due dates in the next 4 days.

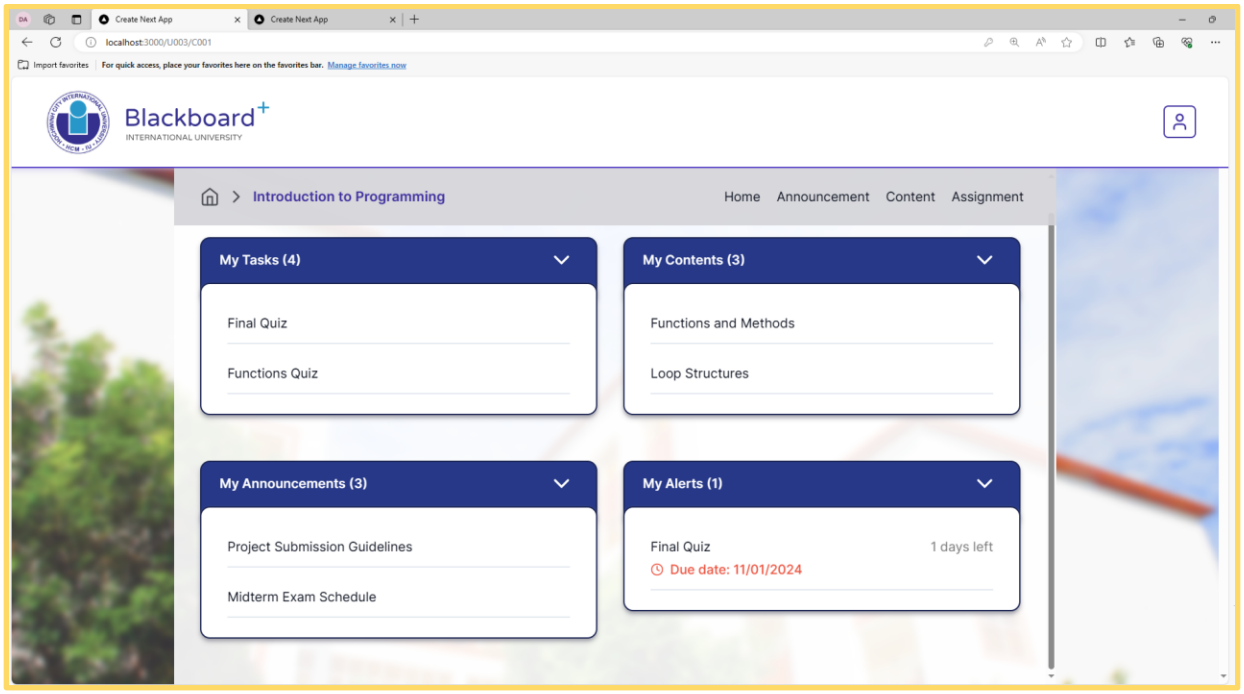

Figure 33. Specific Course page

Figure 34. Navigation Bar

Facilitating flexible navigation between items for users.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

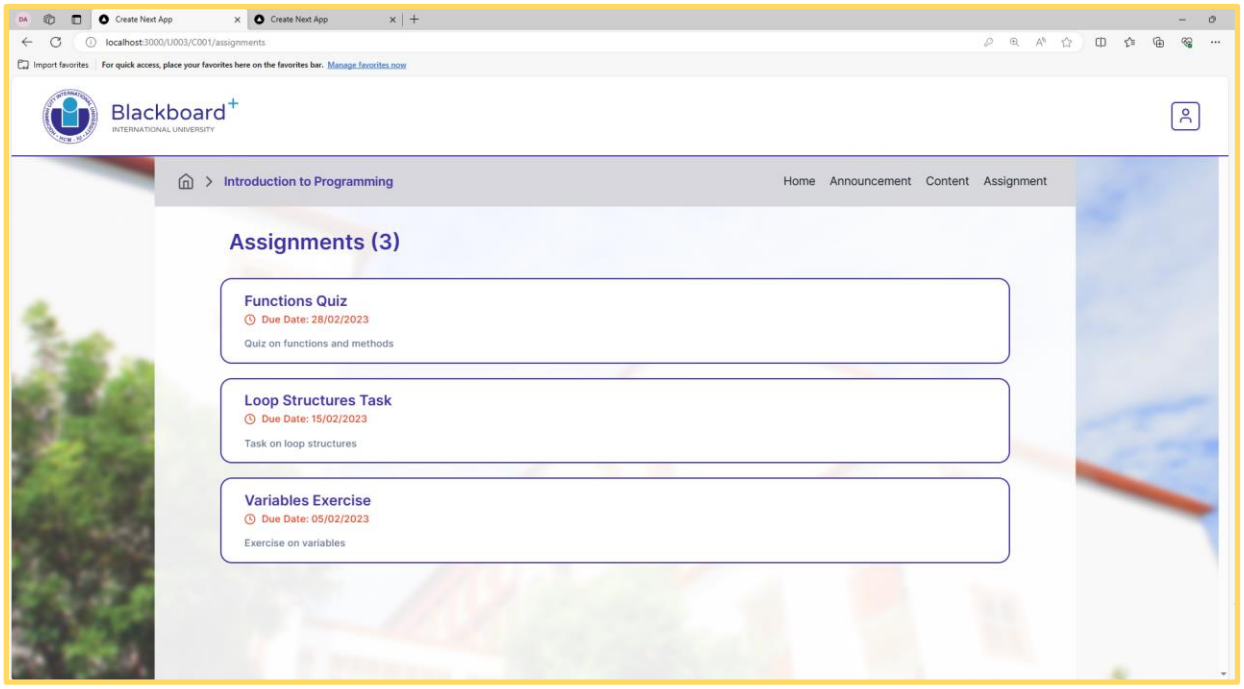

Figure 35. Specific Course's Assignments page

Display the assignment titles, deadlines and short descriptions.

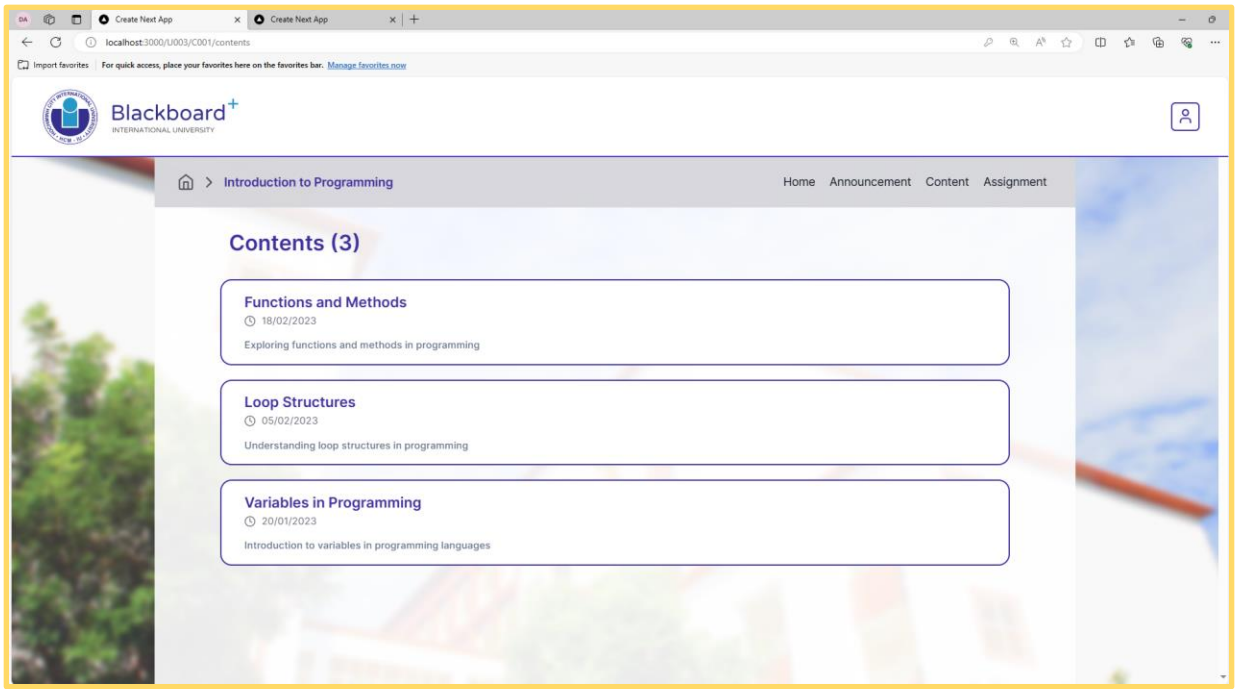

Figure 36. Specific Course's Contents page

Project's Github: https://github.com/koitran14/Course-Management-Project.git

Display the content titles, creation dates and short descriptions.

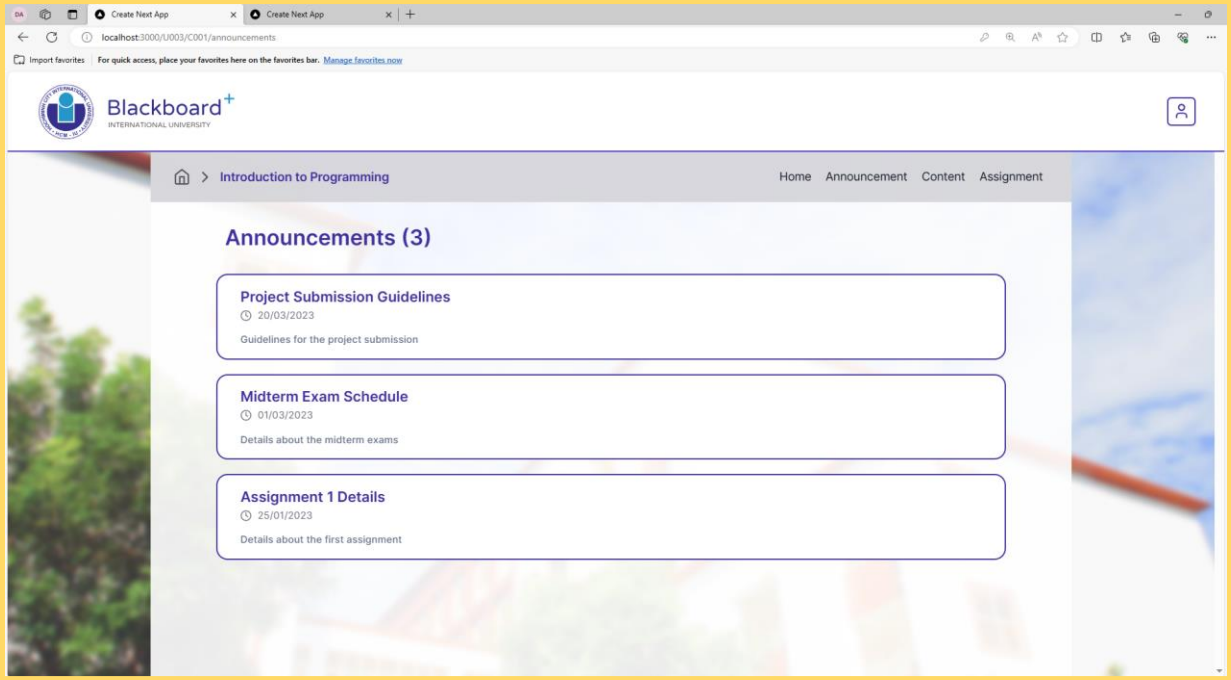

Figure 37. Specific Course's Announcements page

Display the announcement titles, creation dates and short descriptions.

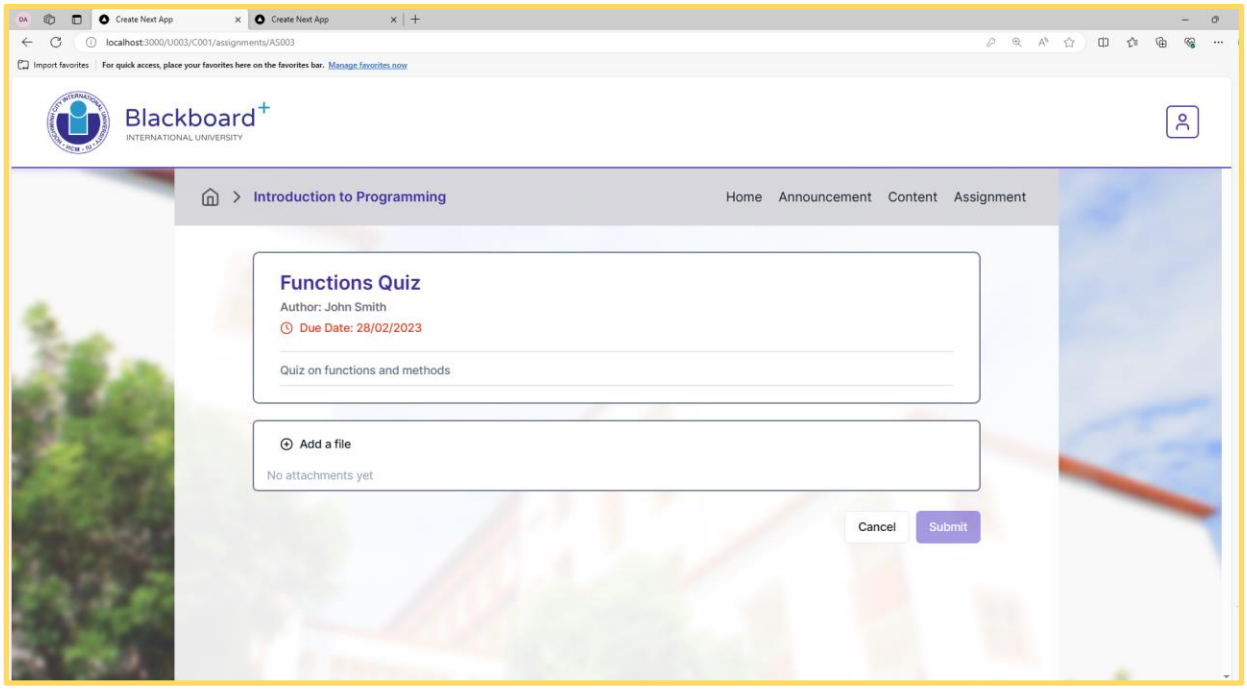

Figure 38. Specific Assignment page

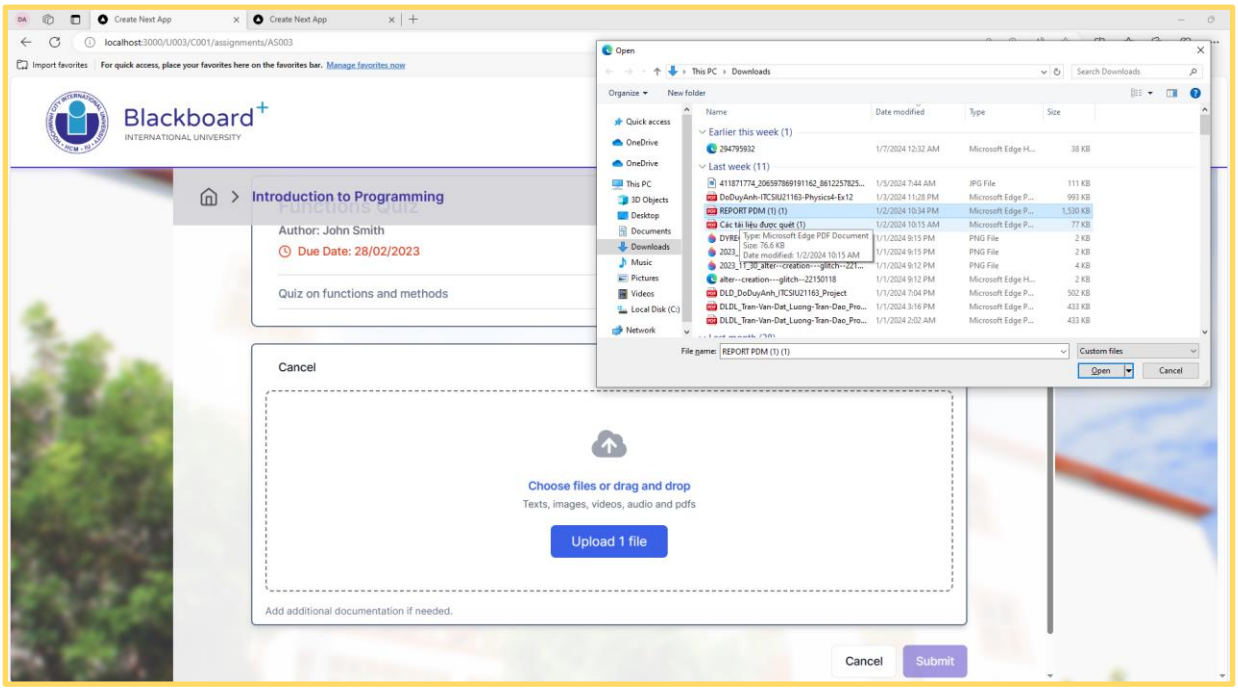

Figure 39. File Attachment (1)

Project's Github: https://github.com/koitran14/Course-Management-Project.git

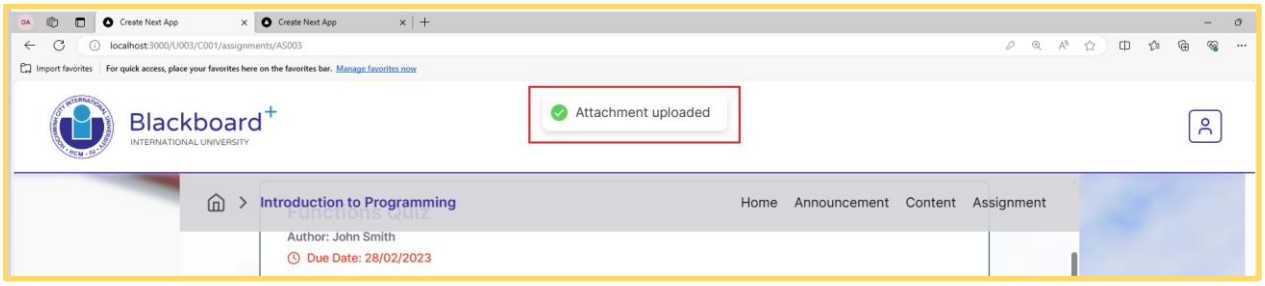

Figure 40. Attachment Uploaded notification

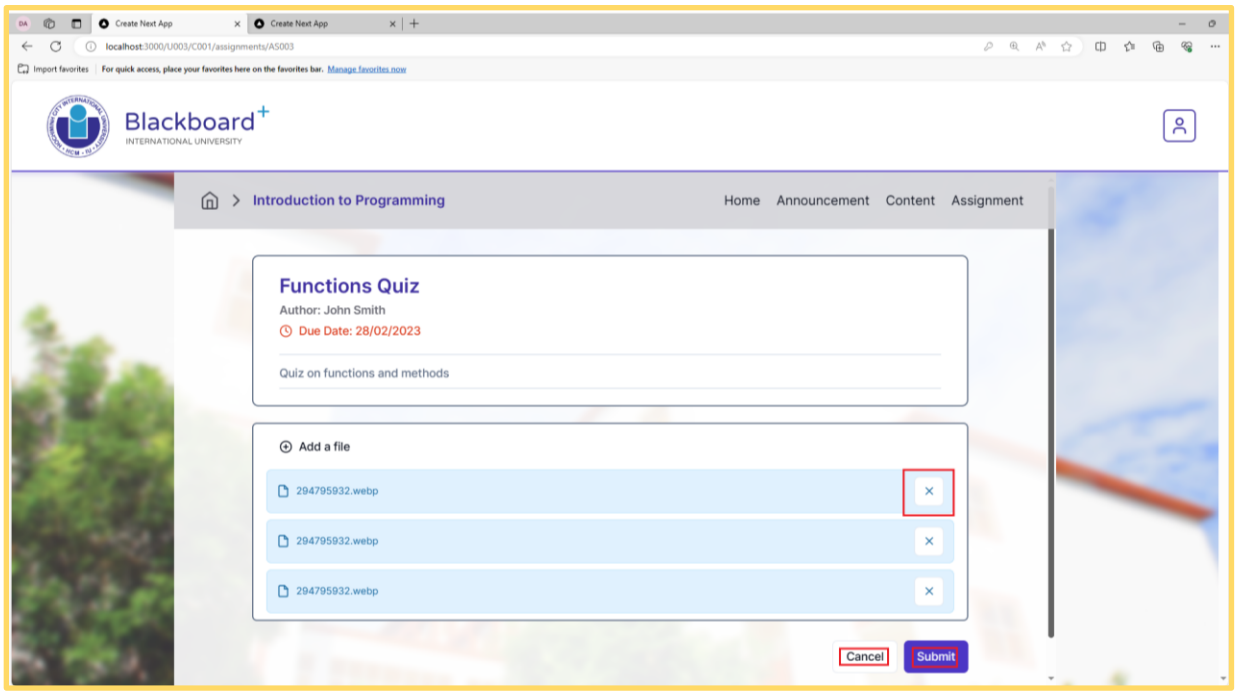

Figure 41. File Attachment (2)

Press the X button to detach the file.

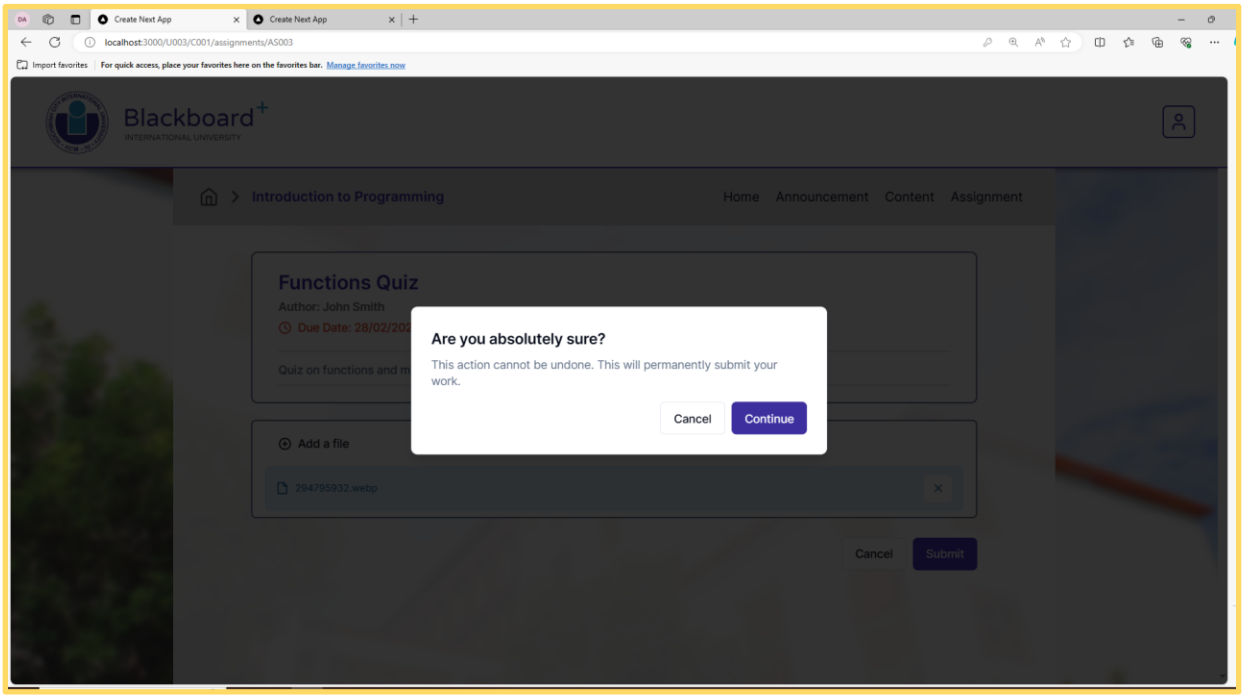

Figure 42.Confirmation prompt (Submit assignment)

Appears after the user clicks submit.

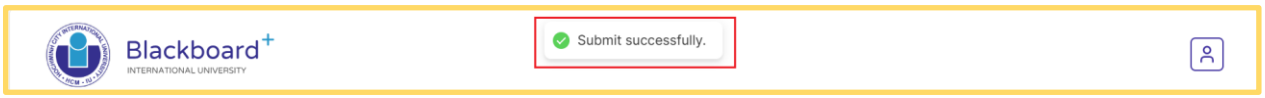

Figure 43. Successfully submit notification

Users will be redirected to the Specific Course Assignment page.

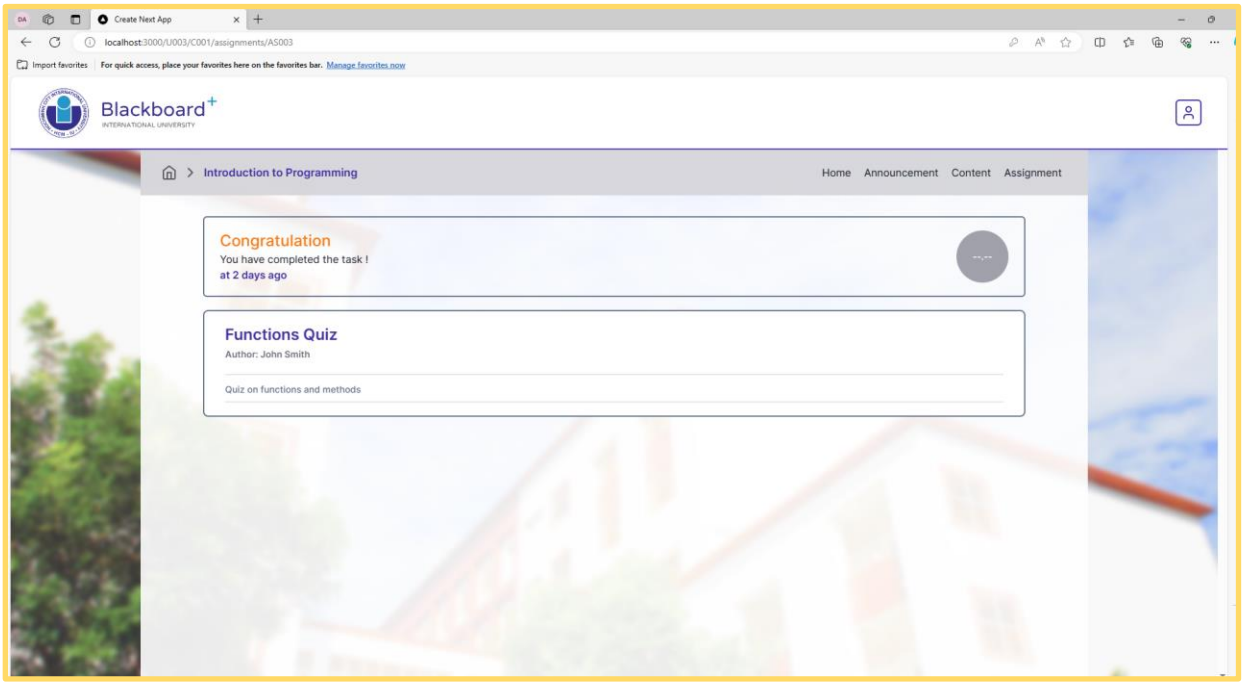

Figure 44. Specific Assignment page after submission (ungraded)

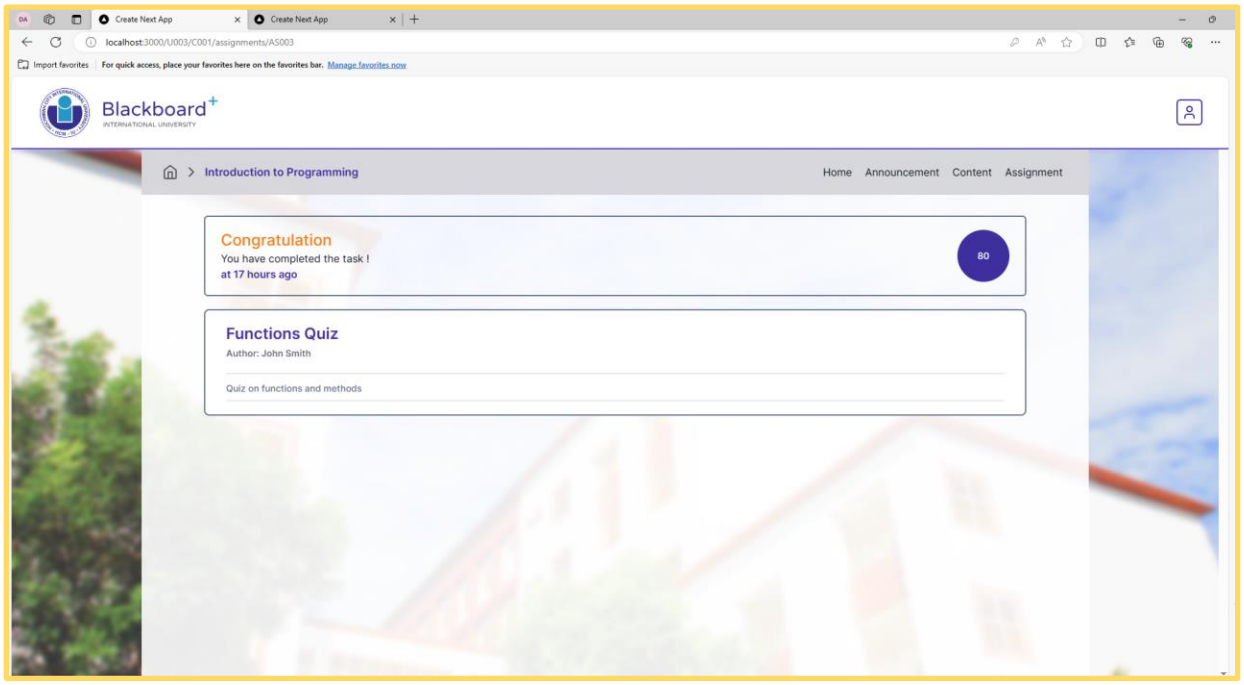

Figure 45. Specific Assignment page after submission (graded)

Project's Github: https://github.com/koitran14/Course-Management-Project.git

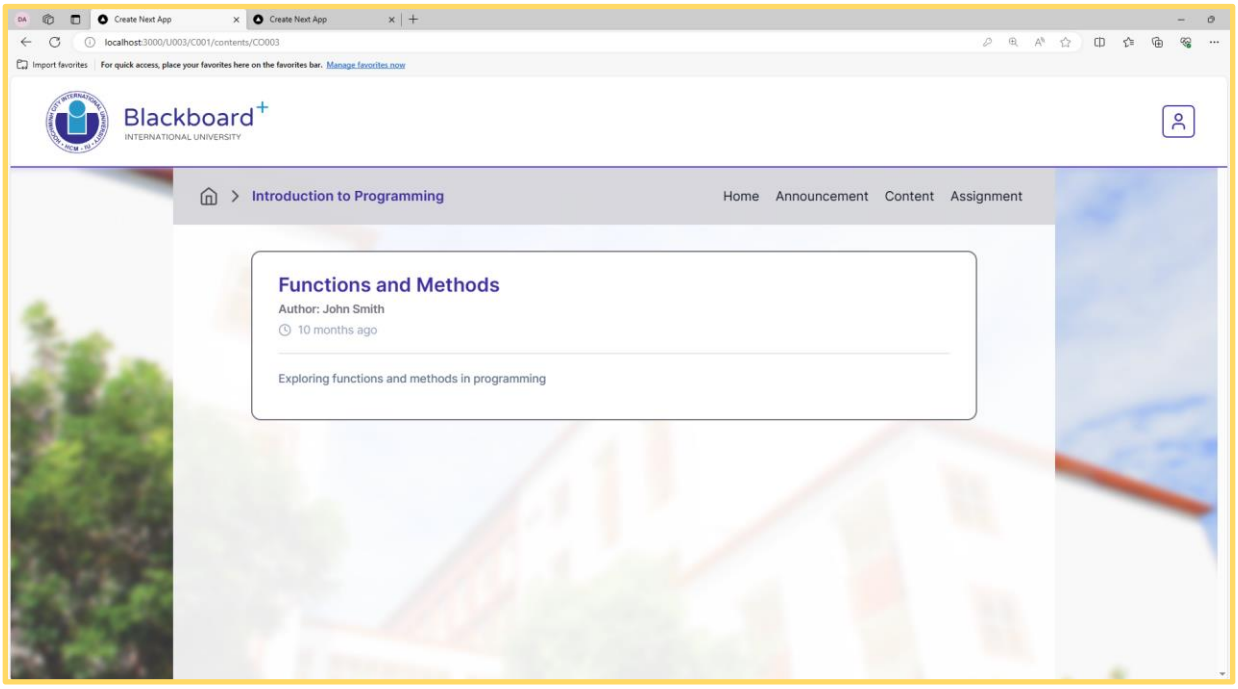

Figure 46. Specific Content page

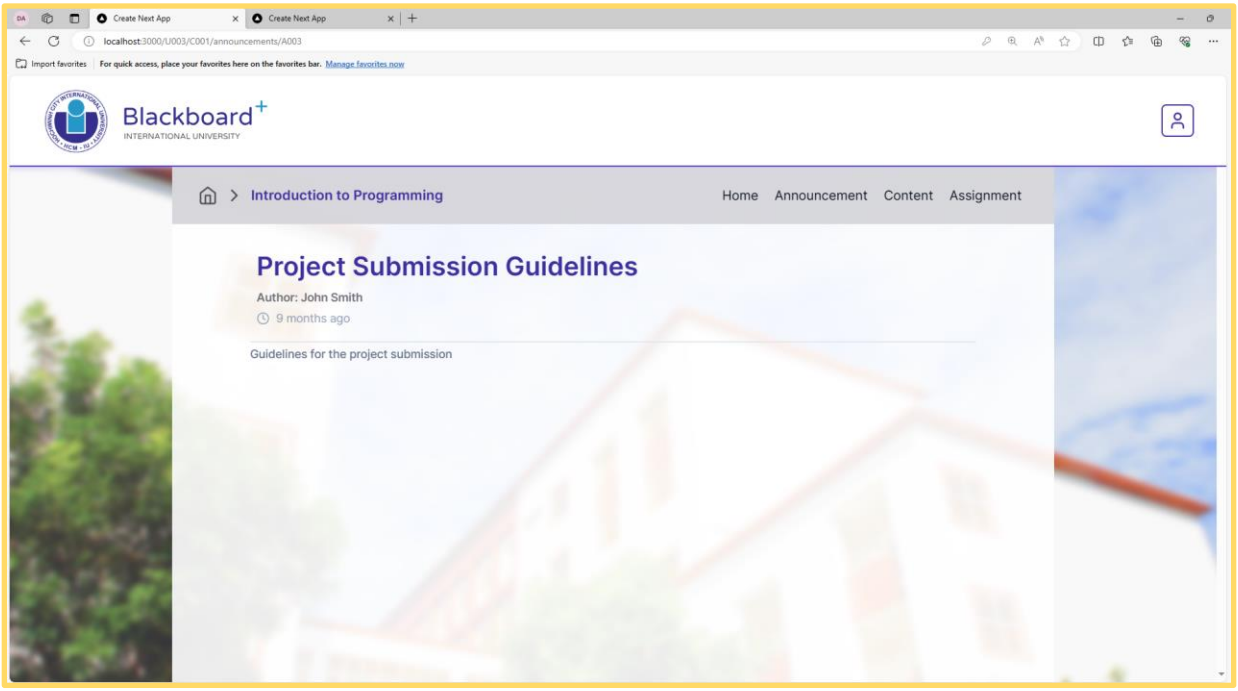

Figure 47. Specific Announcement page

Project's Github: https://github.com/koitran14/Course-Management-Project.git

c. Teacher mode

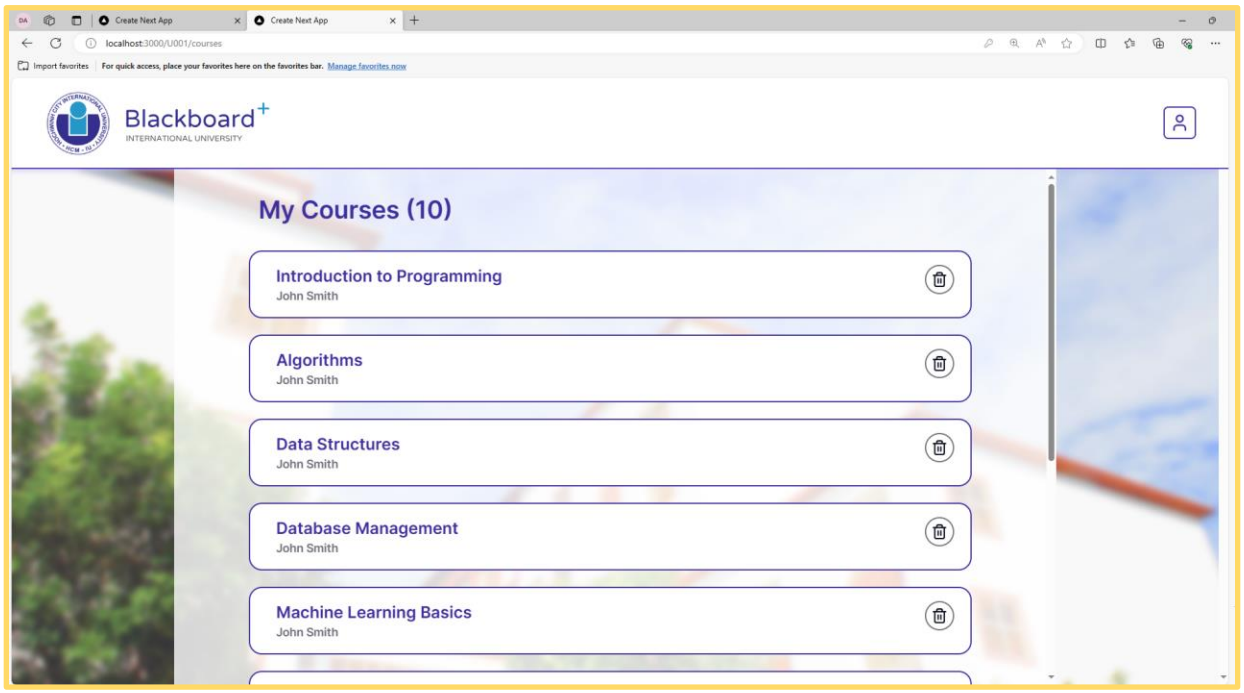

Figure 48. My Courses page

The course deletion option is available.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

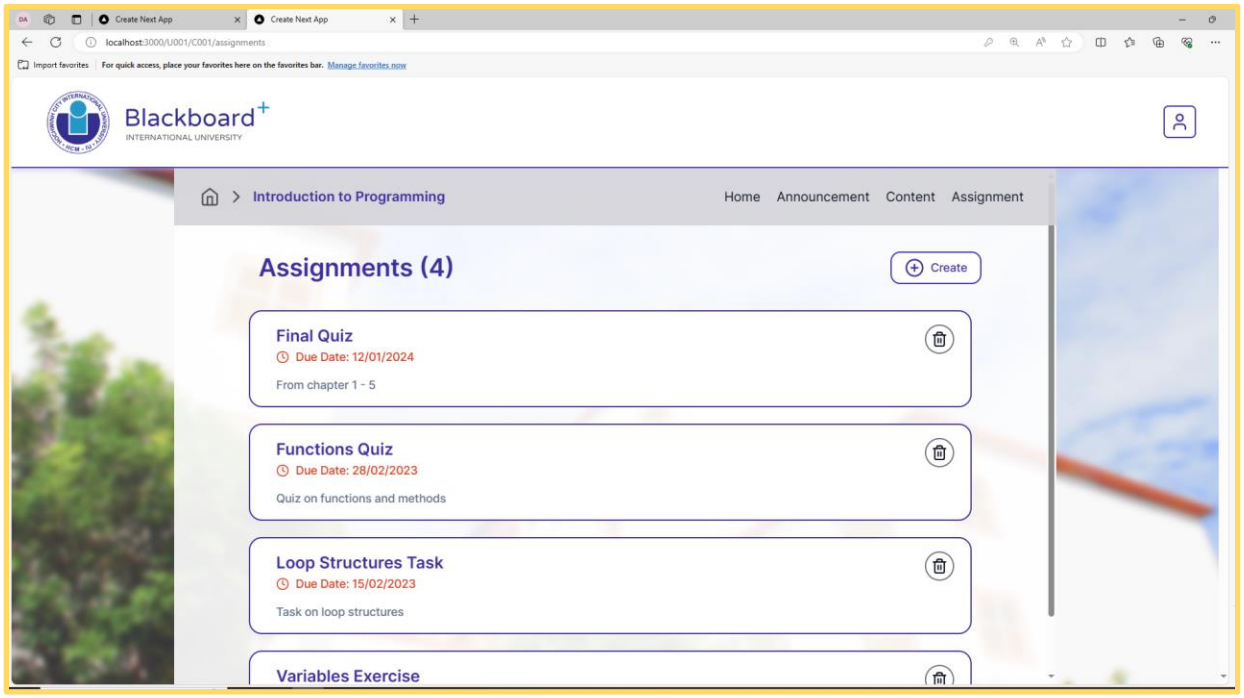

Figure 49. Specific Course's Assignment page

The assignment creation and deletion option are available.

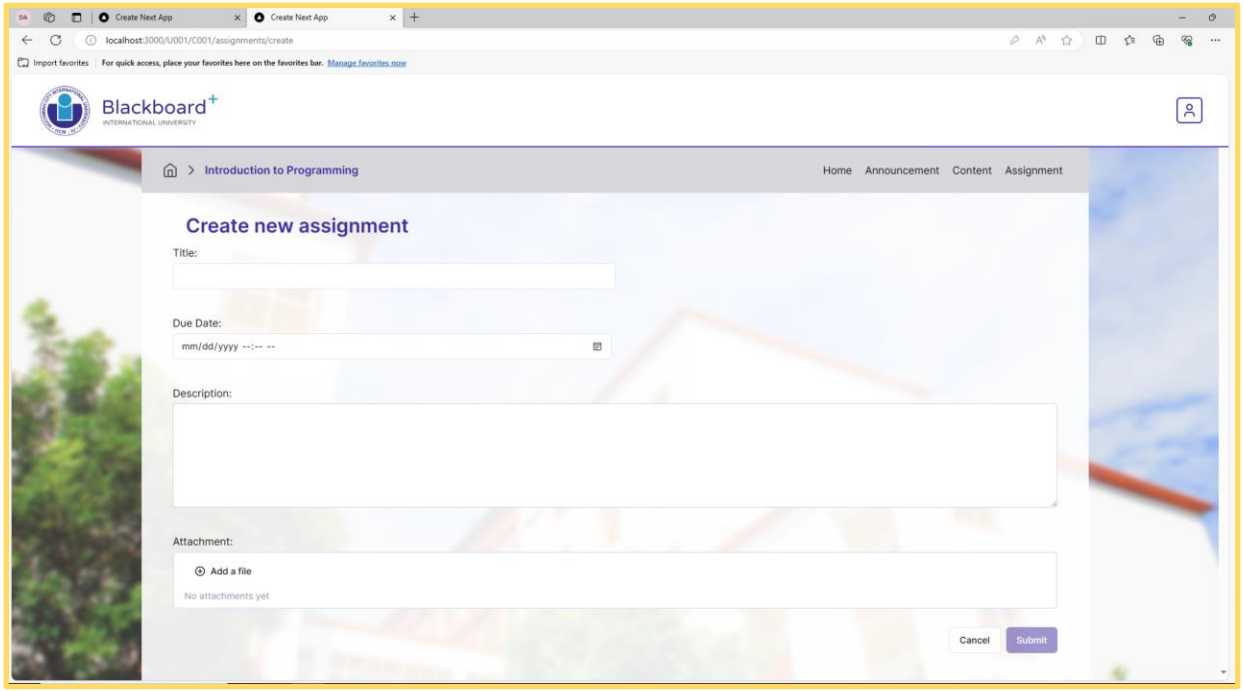

Figure 50. Create Assignment page

Project's Github: https://github.com/koitran14/Course-Management-Project.git

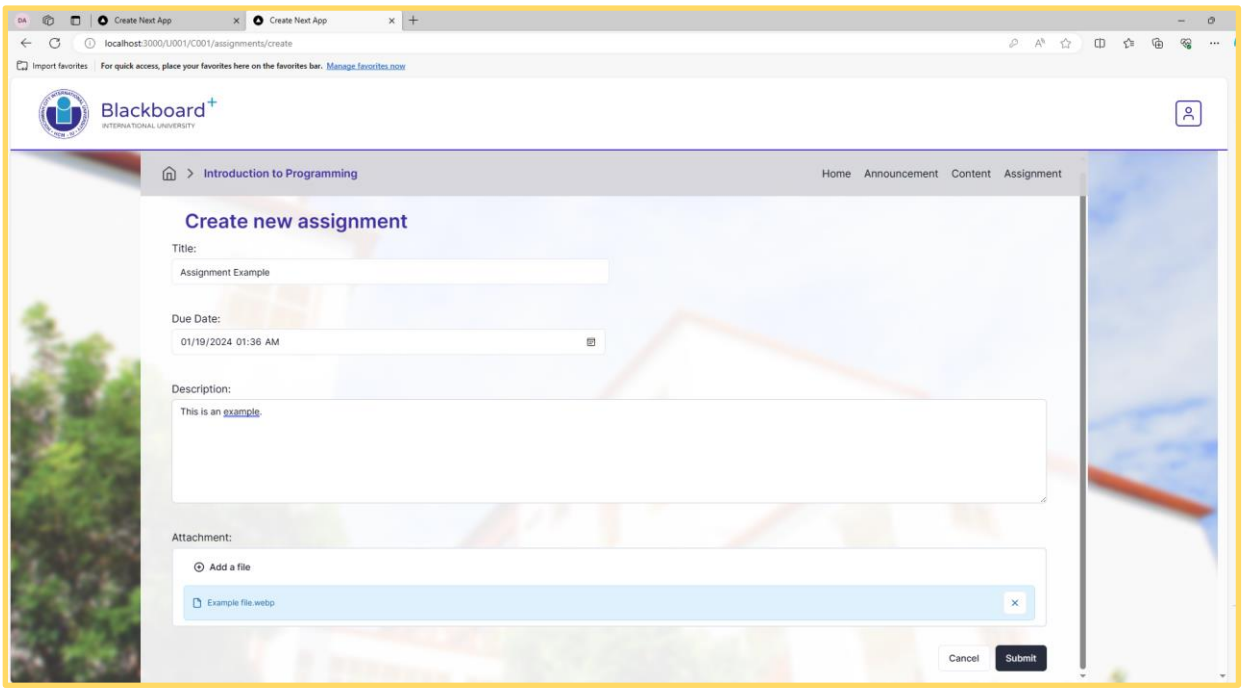

Figure 51. Create Example (assignment)

Similar steps as students submitting assignments.

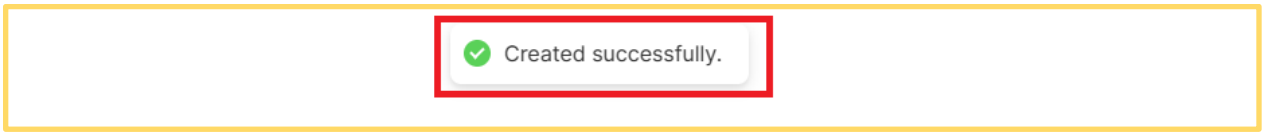

Figure 52. Successfully create notification

Similar for content and announcement.

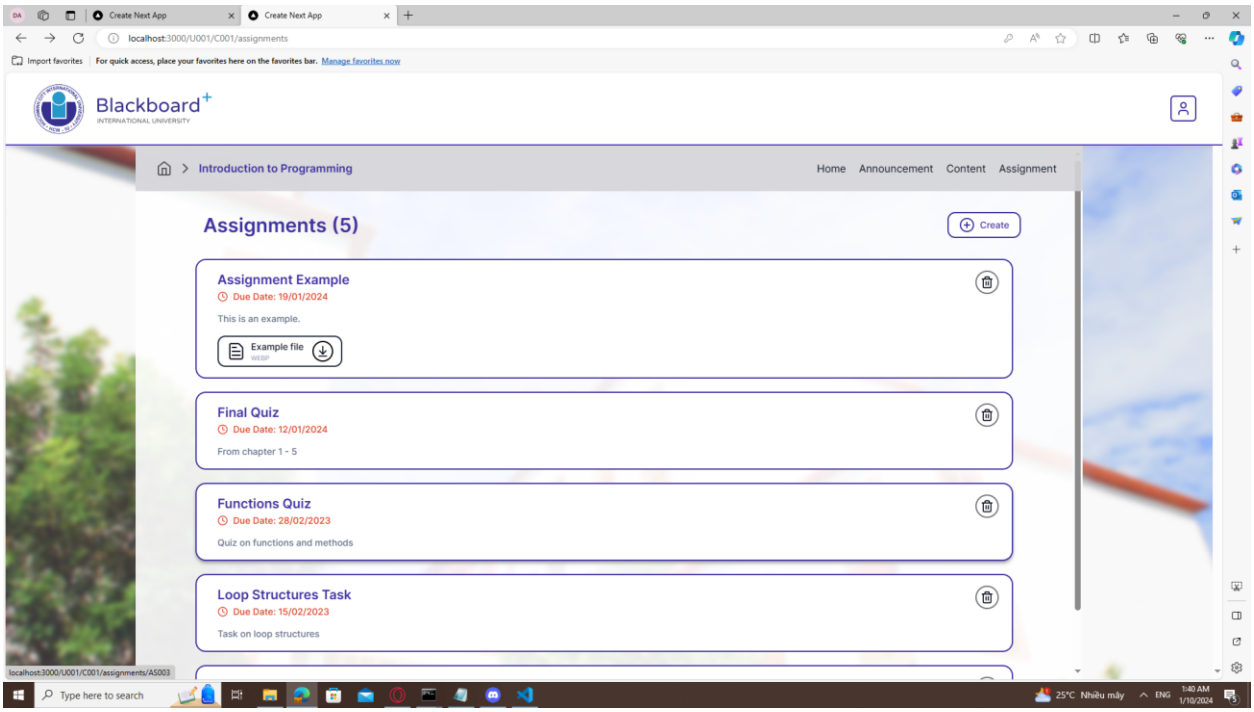

Figure 53. New assignment

Display attachment(s) for the assignment.

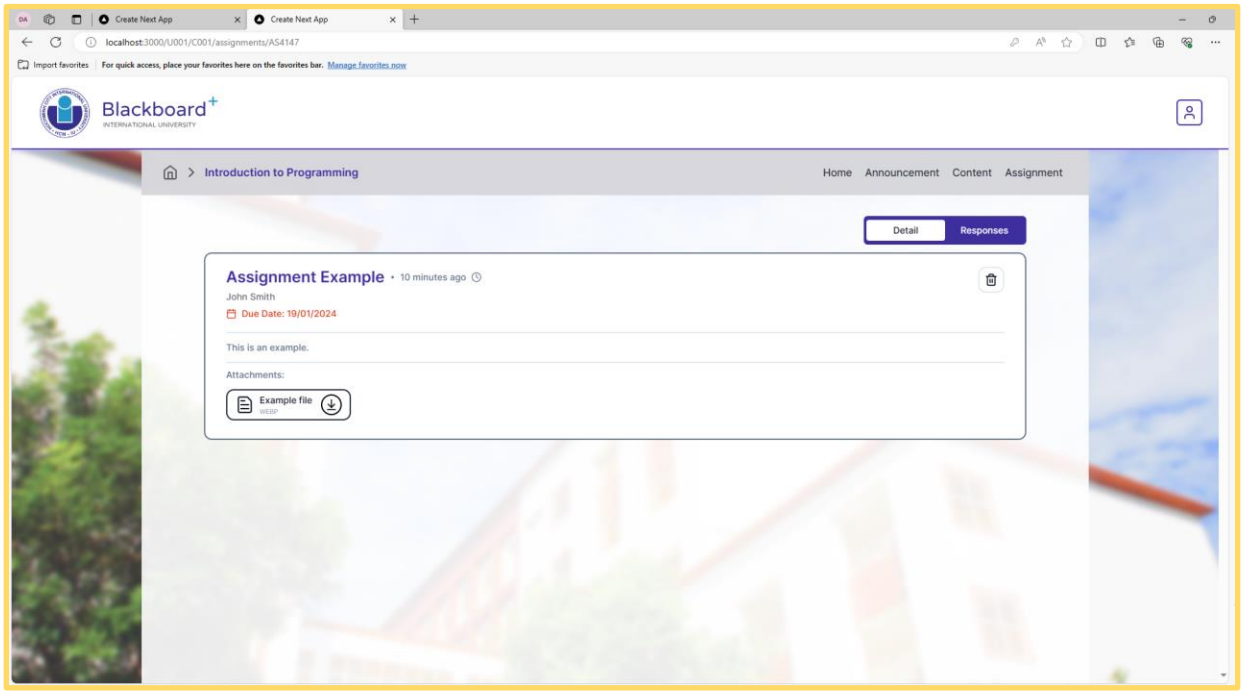

Figure 54. Specific Assignment page

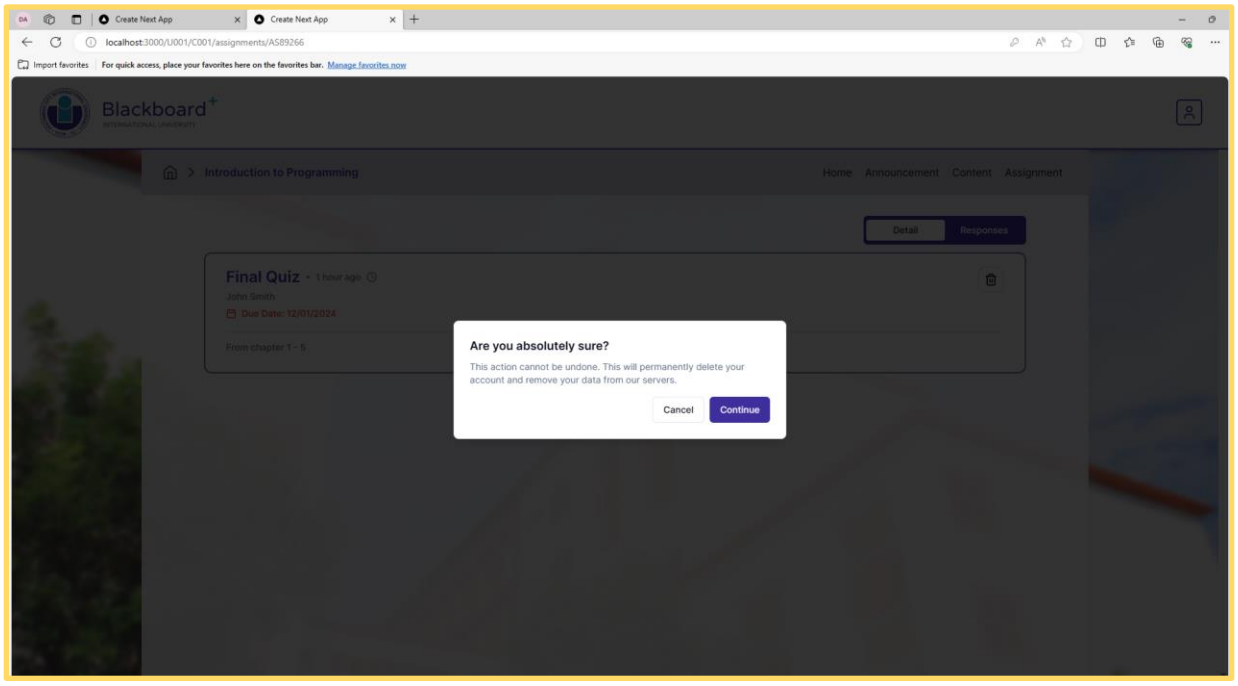

Figure 55. Confirmation prompt (Delete Assignment)

Similar for content and announcement.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

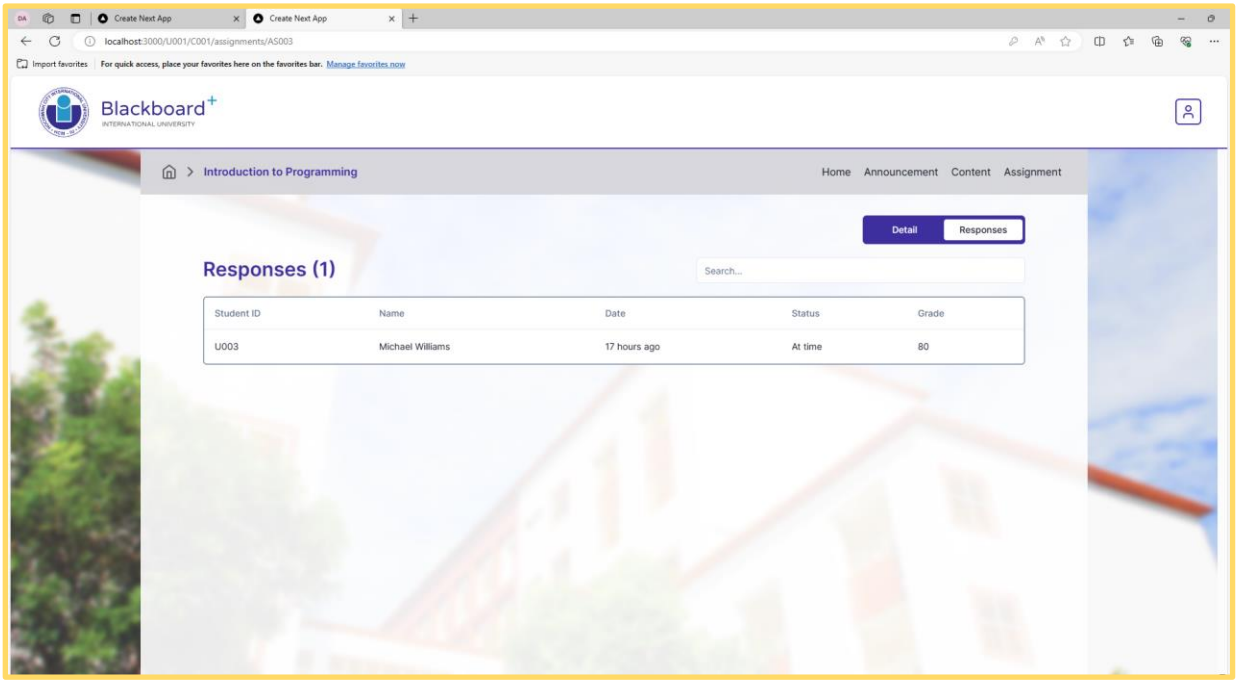

Figure 56. Specific Assignment's Responses

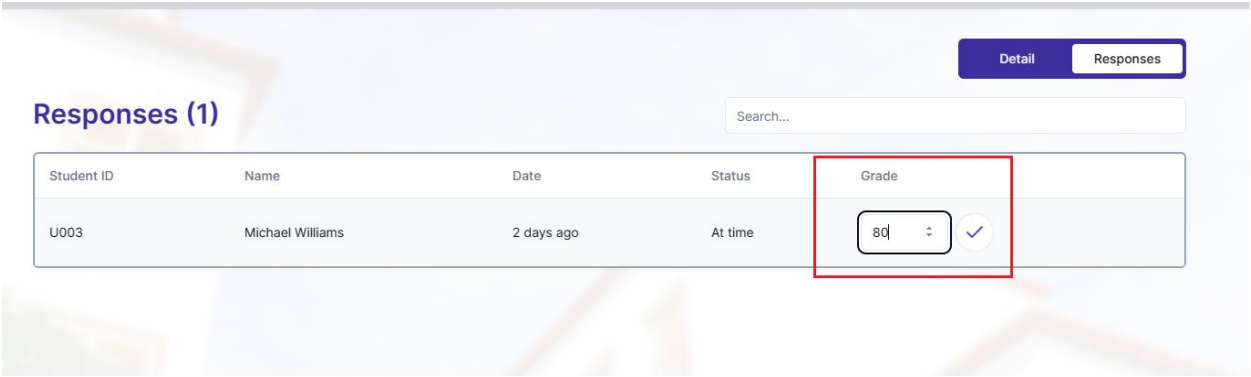

Figure 57. Specific Assignment's Responses Grading

Project's Github: https://github.com/koitran14/Course-Management-Project.git

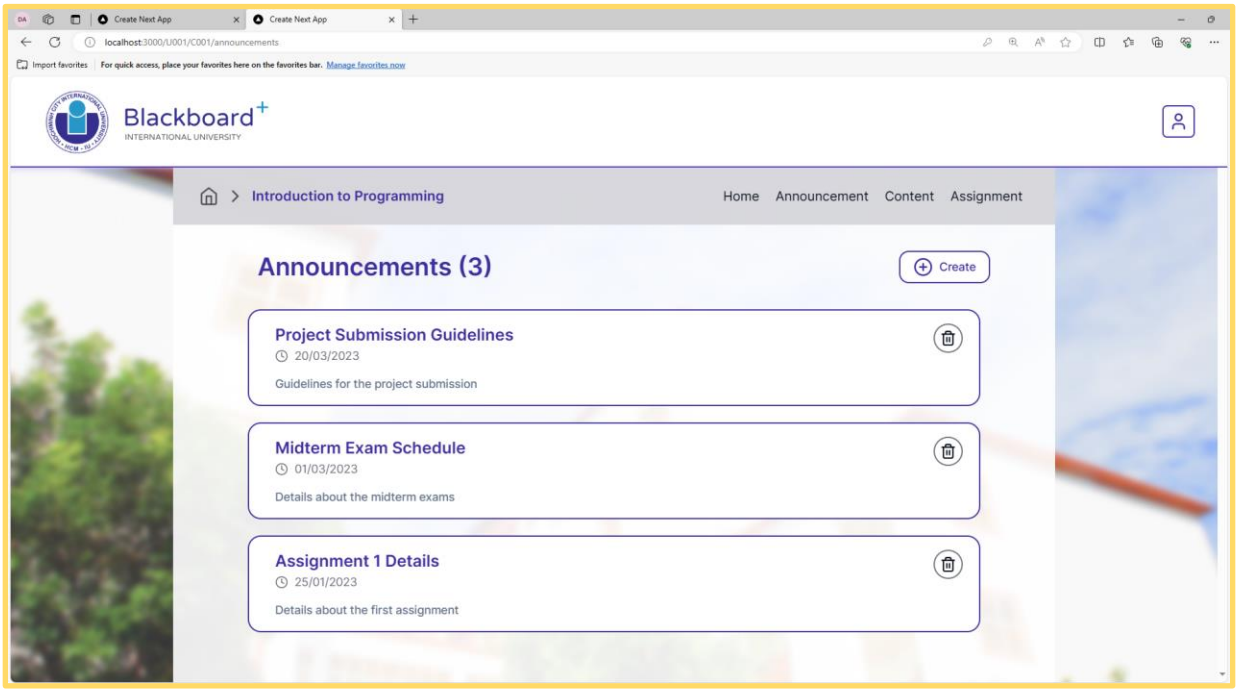

Figure 58. Specific Course's Announcement Page

The announcement creation and deletion option are available.

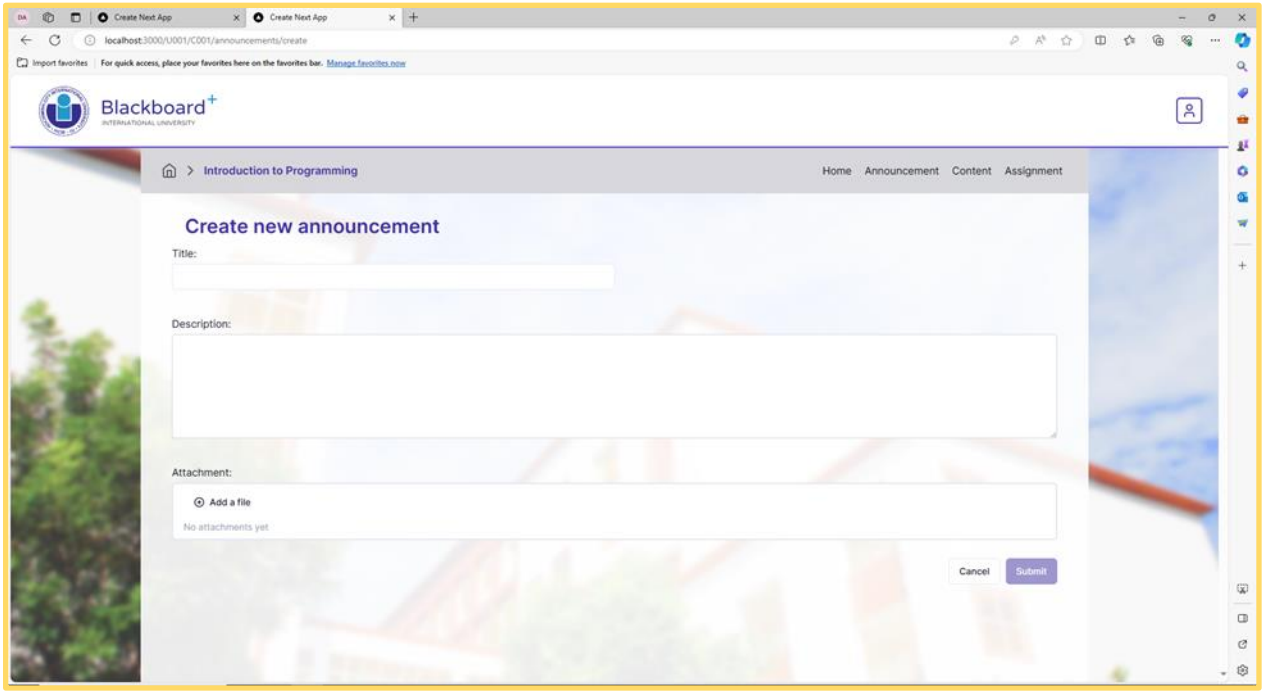

Figure 59. Create announce page

Project's Github: https://github.com/koitran14/Course-Management-Project.git

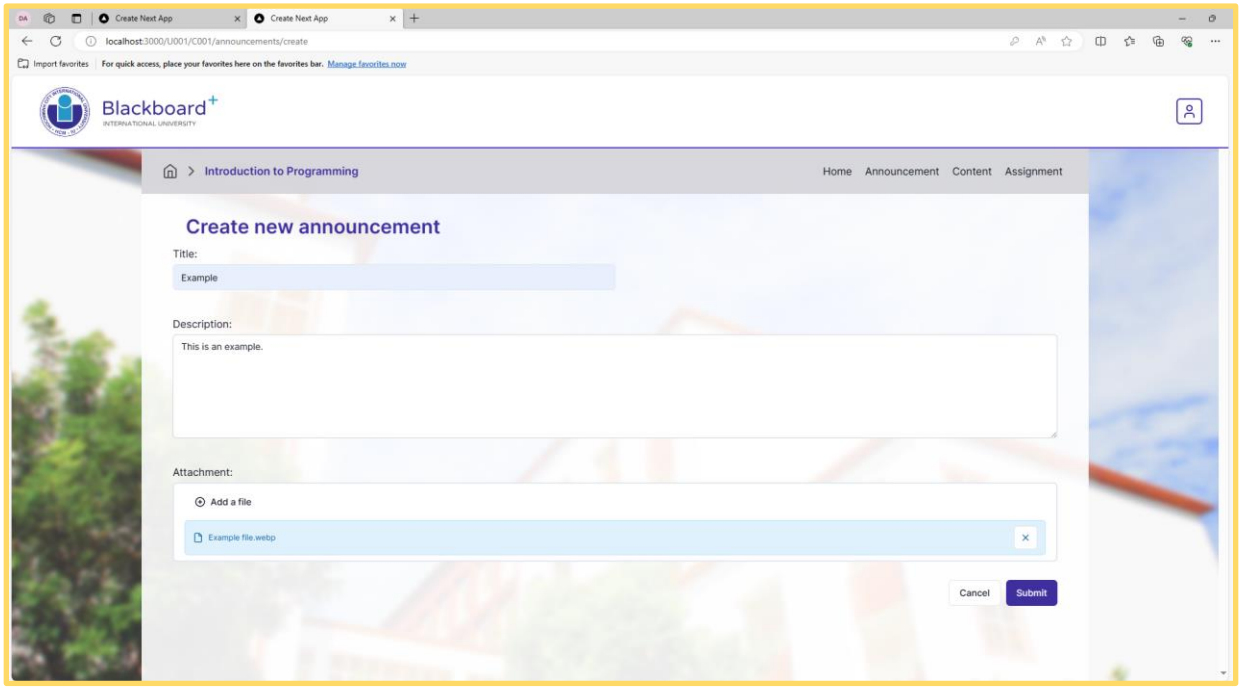

Figure 60. Create Example (announcement)

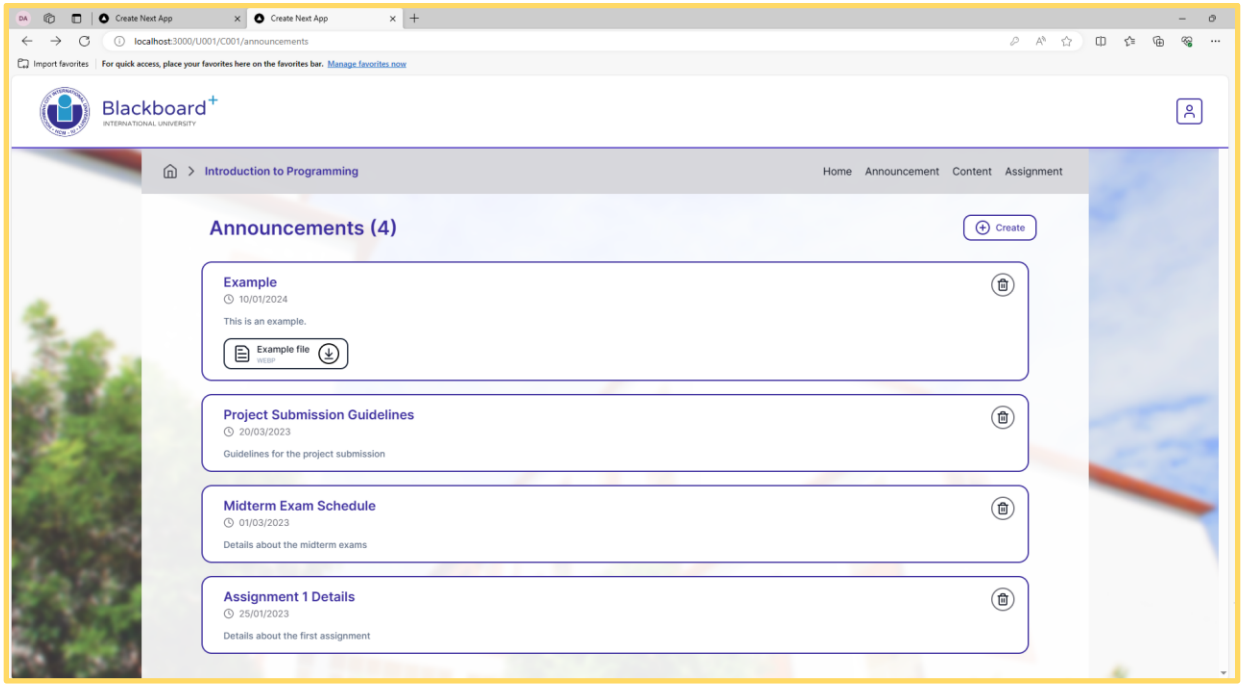

Figure 61. New announcement

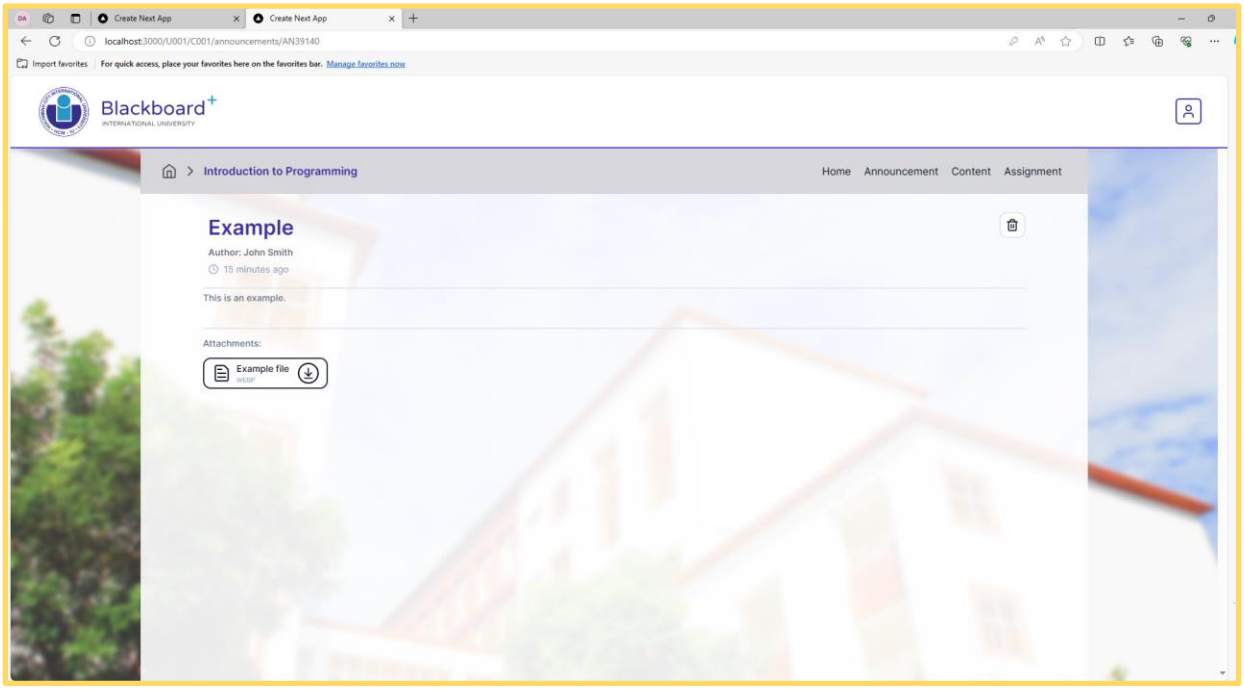

Figure 62. Specific Announcement page

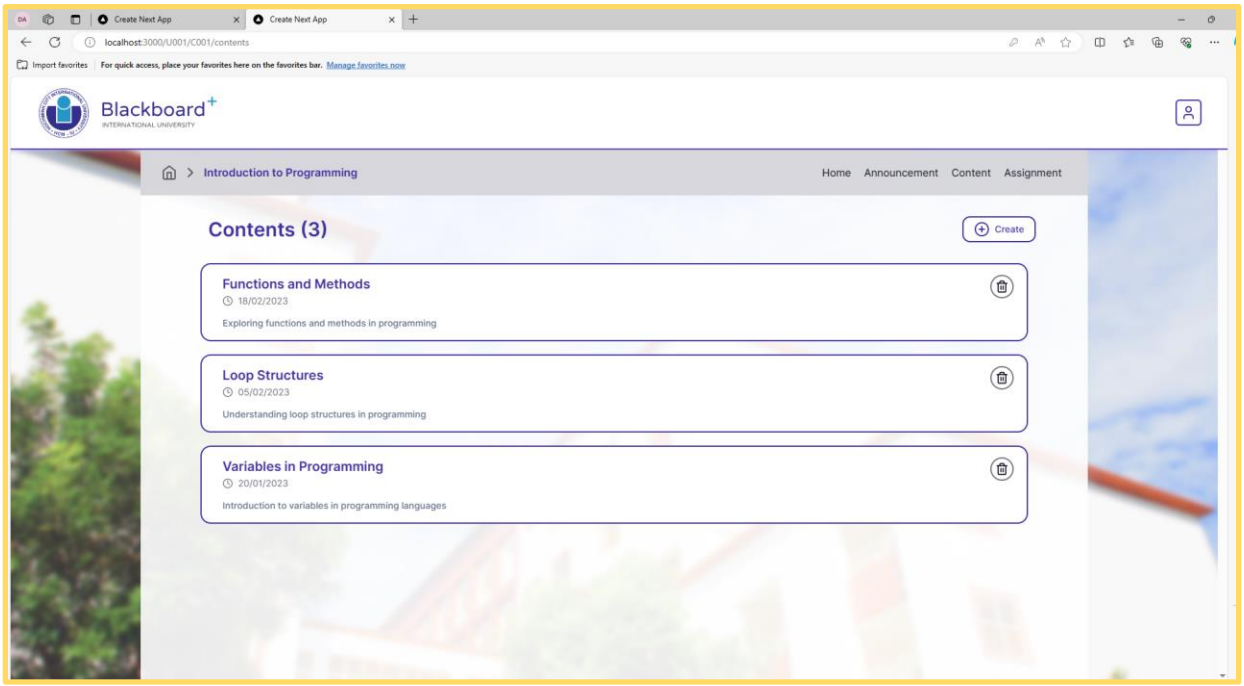

Figure 63. Specific Course's Content page

Project's Github: https://github.com/koitran14/Course-Management-Project.git

Content creation and deletion options are available.

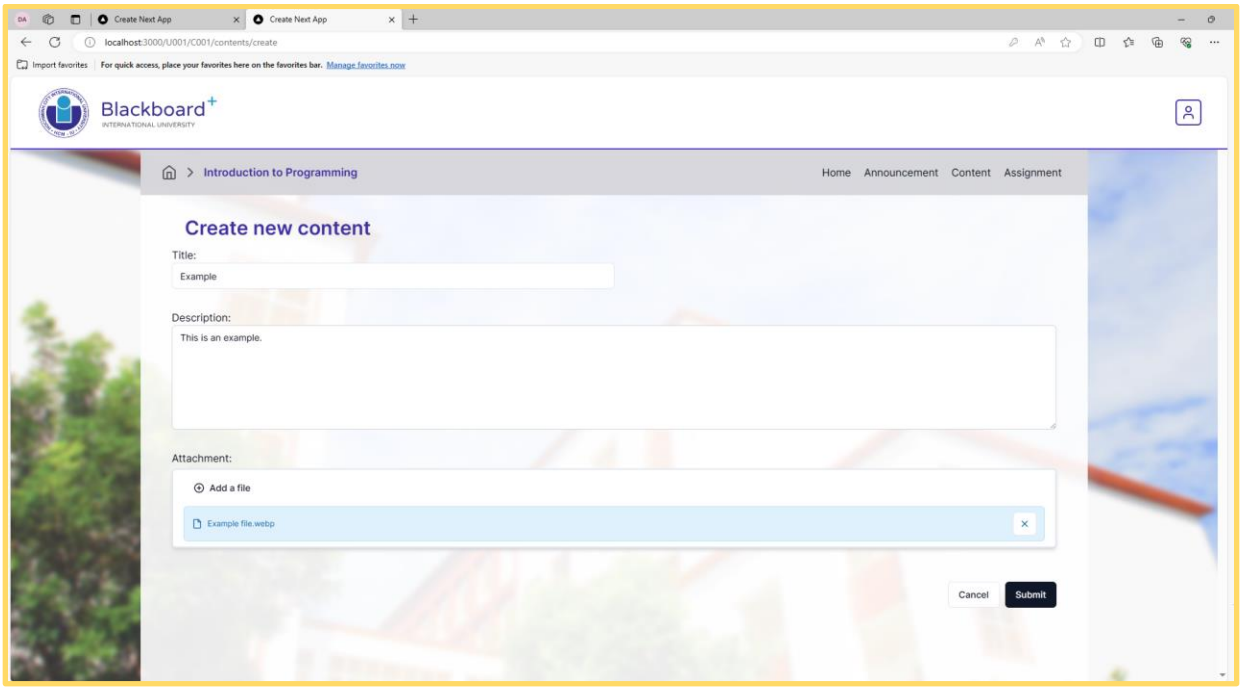

Figure 64. Create Content page

Project's Github: https://github.com/koitran14/Course-Management-Project.git

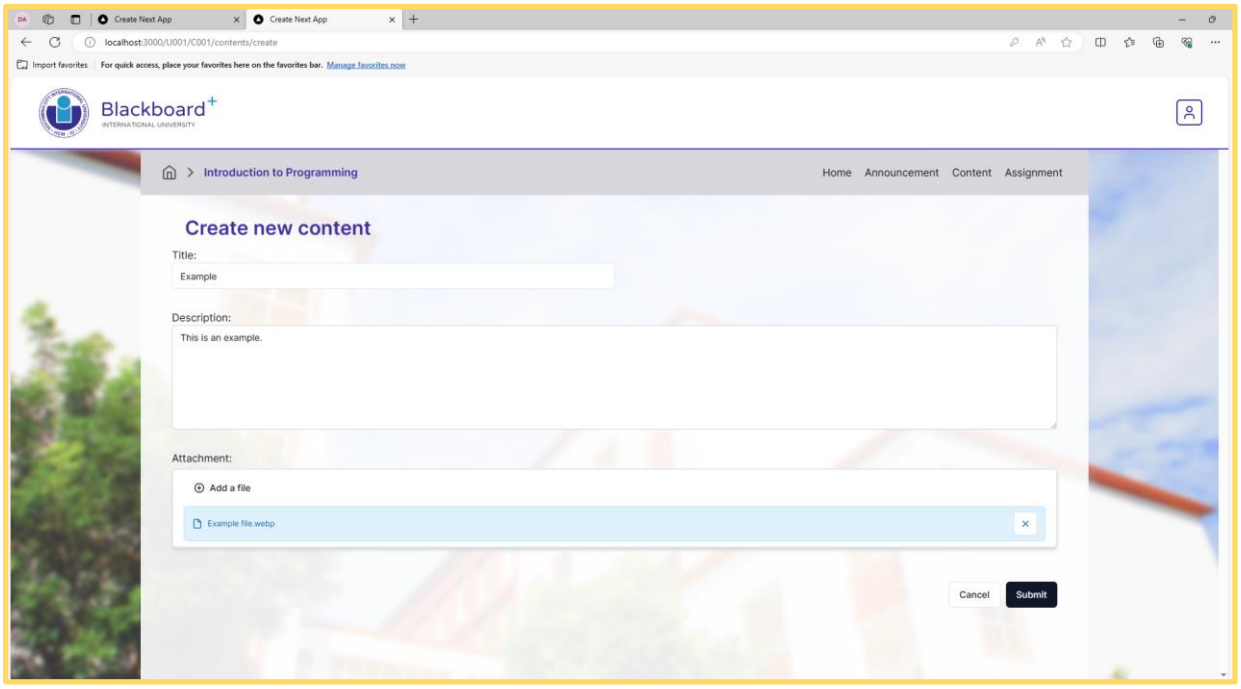

Figure 65. Create Example (content)

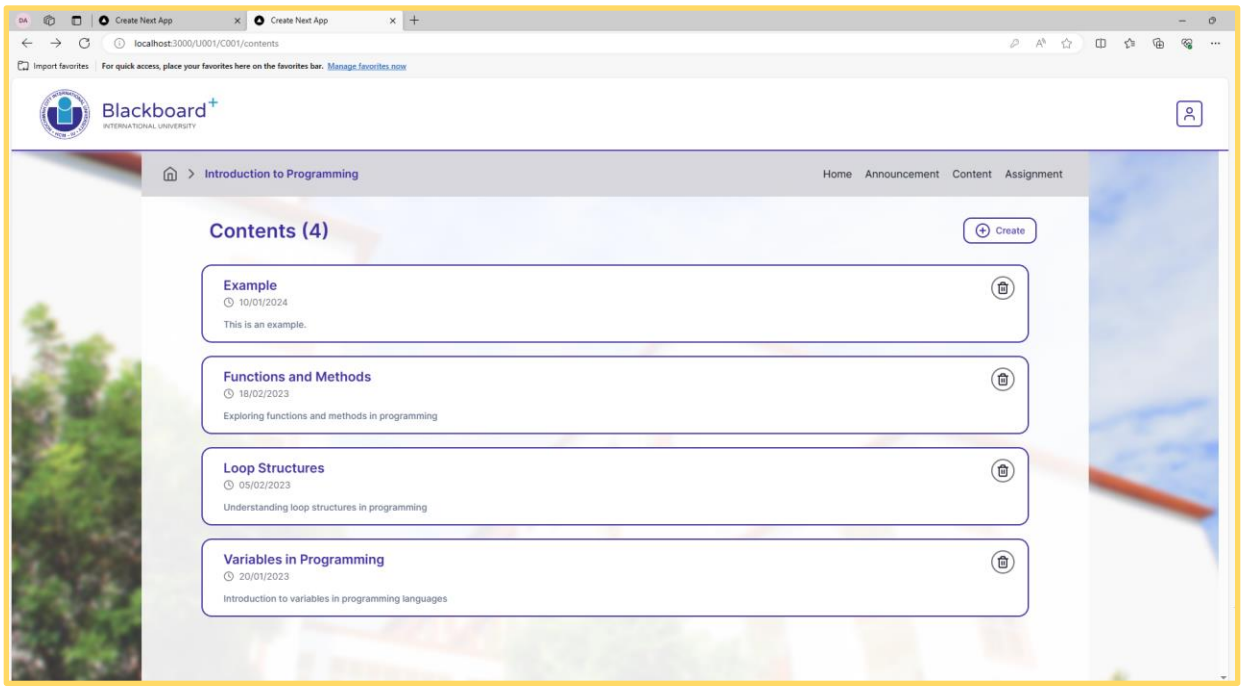

Figure 66. New content

Project's Github: https://github.com/koitran14/Course-Management-Project.git

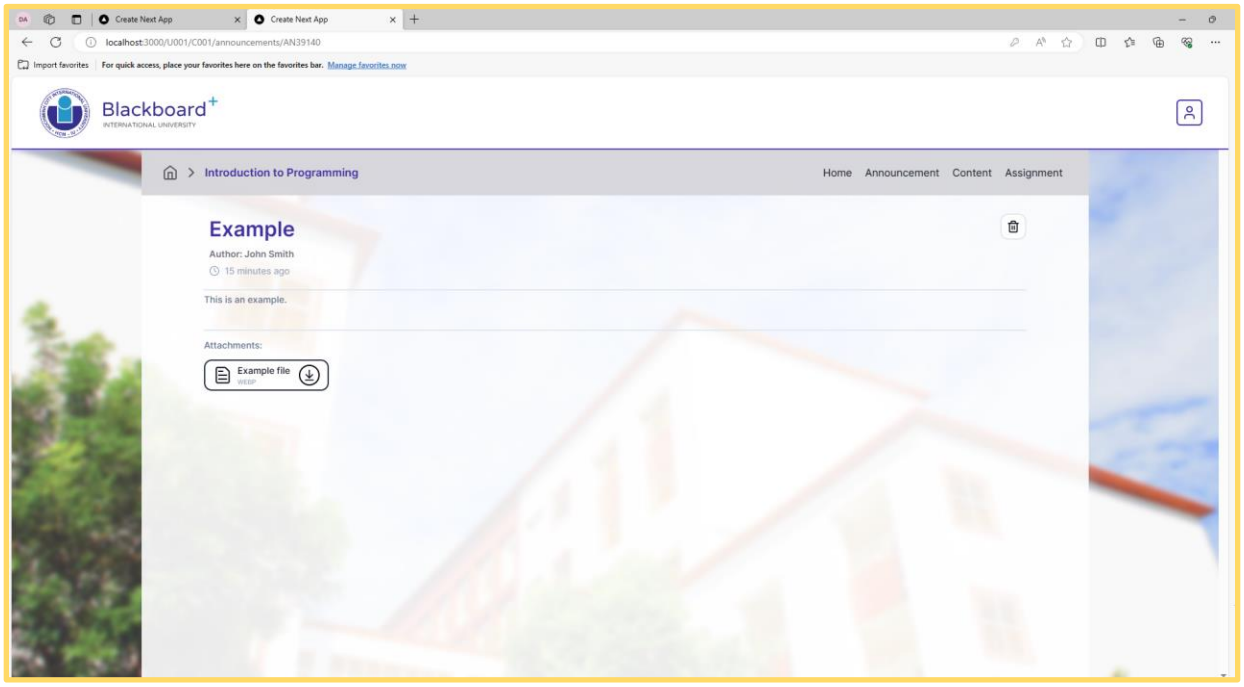

Figure 67. Specific Content page

IV. Conclusion

1. Achieve goals

During the project, our team successfully achieved the goal of developing the learning management system. Firstly, we design a database to adhere to the requirement of B.C Normal Form, ensuring efficient data storage and retrieval while eliminating data anomalies. Secondly, we connect the front-end interface of the application to the back-end database using Nodejs, establishing a seamless and secure connection between the authorization and authentication system, edit profile, add/ remove/edit course elements, access course elements and file attachments, implementing complex queries to manage various scenarios effectively.

Through completing these goals, our project has delivered a reliable and user-friendly learning management system that meets the requirements of modern database management, security, and multi-role user experience.

Project's Github: https://github.com/koitran14/Course-Management-Project.git

2. Future work

Having been implemented for a month, our project now aligns closely with the initial plan. However, there are several areas that we aim to enhance and maintain moving forward. These include:

+ Expanding editing capabilities for teachers to cater to various needs.
+ Allow teachers to create courses and manage student tasks - enabling actions like adding, editing, or removing students from courses.
+ Strengthening the authentication and authorization system with robust security measures, focusing on cookies and security protocols.
+ etc.

These improvements will further solidify the project's functionality and usability.

3. Concluding thoughts

Through this project, we have gained valuable knowledge in database design. Additionally, we have developed skills in planning, execution, and error handling while working in a team, which will be highly beneficial for future employment. The support from our instructors in clarifying our doubts has been indispensable. We greatly appreciate this opportunity and have put in our utmost effort to create the most comprehensive educational management system project possible. Our goal is to apply it in real-world scenarios at schools, with the hope that the system will continue to develop and contribute to the progress of our education.

V. References

1. NextJS: Next.js by Vercel - The React Framework (nextjs.org)
2. TailwindCSS: Tailwind CSS - Rapidly build modern websites without ever leaving your HTML.
3. Component UI Library Shadcn-ui: shadcn/ui
4. SQL Server with NODE.JS: (851) NodeJS - SQL Server bài 1 - Cài đặt nodejs - giới thiệu về npm tạo ứng dụng node đầu tiên - YouTube
5. Uploadthing documentation: What Is UploadThing? – uploadthing
6. Axios: axios - npm (npmjs.com)

Project's Github: https://github.com/koitran14/Course-Management-Project.git