

Abstract

In an era where digital technology is transforming education, the rapid advancement of digital tools such as simulators, virtual labs, video conferencing, and electronic learning (E-learning) platforms underscores the necessity for flexible and accessible learning environments.

Our project aims to create an E-Learning Web Application, named Student Portal, for students at university. It offers both the student and the teacher experiences, ensuring that each student receives his respective courses, with a modern, user-friendly, and accessible interface. The frontend is developed using the React framework and Tailwind providing a simple and clear user interface with efficient rendering. On the backend, Node.js and Express.js deliver a scalable, non-blocking environment for handling real-time requests, while MongoDB offers flexible, schema-less data management for evolving application needs. Lastly, it is created with security as its core basis, implementing the latest authentication and web security technologies utilizing JSON Web Tokens, Cross-origin resource sharing (CORS), and bcryptJs.

By providing a modern, adaptable, and secure solution, Student Portal enhances the accessibility and efficiency of educational delivery, meeting the diverse needs of today's learners.

Keywords: HTML, CSS, Javascript, React, NodeJS, ExpressJS, UML, Web developement, E-learning, Coures.

Dedication

To my parents, who planted the seed of knowledge in my mind and nurtured it.

To my dear siblings, Karim, Yasmine, and Mohamed.

To my friends who believed in me when I didn't and stood by my side.

To the pursuit of knowledge and the ones who make it possible, my teachers.

To everyone who lent a hand, each act of your kindness and support made a difference.

To myself, for showing up, pushing through, and finishing what I started.

Acknowledgment

First and foremost, I would like to praise Allah the Almighty, the Most Gracious, and the Most Merciful for His blessing given to me during my study path and in completing this project.

I would also like to express our sincere gratitude to my supervisor Dr. Sadouki Leila for her continuous support, assistance, patience, and valuable feedback. Her guidance was priceless for the realization of this project.

My deep appreciation also goes to all who supported me through my educational journey at the Institute of Electrical and Electronics Engineering of the University M'Hamed BOUGARA – Boumerdes.

Table of Contents

Abstract.....	I
Dedication.....	II
Acknowledgment.....	III
Table of Contents.....	IV
List of Figures.....	VII
List of tables.....	VIII
List of Abbreviations	IX
General Introduction	1
Chapter 1: Web Development and E-learning Overview	3
1.1 Introduction.....	3
1.2 Web Development Overview	3
1.2.1 Definition	3
1.2.2 Frontend and Backend	4
1.2.3 Web Server and Web Client.....	4
1.2.4 Web Request	5
1.2.5 Web Hosting	6
1.2.6 Web Development Fundamental Technologies	7
1.2.7 Data and Databases	7
1.2.8 Libraries and Frameworks	8
1.2.9 Application Programming Interfaces (APIs) and REST API.....	9
1.2.10 The Document Object Model (DOM).....	9
1.2.11 Static and Dynamic Websites	10
1.2.12 Single Page Websites	10
1.3 Student Portal and E-Learning.....	11
1.3.1 Definition	11
1.3.2 Existing E-Learning Platforms.....	11
1.4 Conclusion	14

Chapter 2: Tools and Technologies	15
2.1 Introduction.....	15
2.2 Modeling Tools	15
2.2.1 Modeling Language	15
2.2.2 Unified Modeling Language (UML).....	15
2.3 Development Tools	16
2.3.1 Visual Studio Code	16
2.3.2 Node Packet Manager (NPM).....	16
2.3.3 Web Browser	16
2.3.4 Git and GitHub.....	17
2.4 Front-End Tools	17
2.4.1 Hyper Text Markup Language (HTML).....	17
2.4.2 Cascading Style Sheets (CSS).....	18
2.4.3 Tailwind CSS	18
2.4.4 JavaScript.....	19
2.4.5 React	19
2.4.6 Redux	19
2.4.7 Axios	20
2.5 Back-End Tools	20
2.5.1 NodeJs.....	20
2.5.2 ExpressJs.....	20
2.5.3 MongoDB	21
2.5.4 Mongoose.....	22
2.5.5 Multer.....	22
2.5.6 JavaScript Object Notation Web Token.....	22
2.5.7 BcryptJs.....	23
2.5.8 Cross-Origin Resource Sharing (CORS)	23
2.6 Conclusion	23
Chapter 3: System Design.....	24

3.1	Introduction.....	24
3.2	System Modeling	24
3.2.1	System Actors	24
3.2.2	Use Case Description	24
3.2.3	Use Case Identification	25
3.2.4	Use Case Diagrams	27
3.3	Database Design.....	29
3.3.1	Database Description	29
3.3.2	Relationships Between Documents.....	35
3.4	Conclusion	38
Chapter 4: Implementation		39
4.1	Introduction.....	39
4.2	Application Interfaces	39
4.2.1	Login Interface.....	39
4.2.2	Dashboard Interface	41
4.2.3	Profile Interface.....	42
4.2.4	Courses Interface.....	45
4.2.5	Course Interface	48
4.2.6	Edit Course Interface	50
4.2.7	Lecture Interface	55
4.3	Conclusion	57
Future Enhancements.....		58
General Conclusion.....		60
References.....		61

List of Figures

Figure 1-1: Web Server and Web Clients Relationship	5
Figure 1-2: Web request.....	5
Figure 1-3: Document Object Model structure example	10
Figure 1-4: EducateMe website	12
Figure 1-5: Google Classroom website.....	13
Figure 1-6: Moodle Platform	13
Figure 2-1: HyperText Markup Language	18
Figure 2-2: Cascading Style Sheets	18
Figure 2-3: JavaScript	19
Figure 2-4: MongoDB data storage	21
Figure 3-1: Teacher use case diagram	28
Figure 3-2: Student use case diagram	29
Figure 3-3: Database collections interface.....	30
Figure 3-4: User document example.....	31
Figure 3-5: Course document example	33
Figure 3-6: Announcement document example	34
Figure 3-7: Embedded data in a document	36
Figure 3-8: Referenced data in a document	36
Figure 3-9: Entity Relationship Diagram.....	38
Figure 4-1: Login initial page	40
Figure 4-2: Login form	40
Figure 4-3: Login page errors	41
Figure 4-4: Dashboard page.....	42
Figure 4-5: User profile dropdown	42
Figure 4-6: Profile details page for a teacher.....	43
Figure 4-7: Profile details page for a student.....	43
Figure 4-8: Edit profile page.....	44
Figure 4-9: Edit profile page after filling in the new information	44
Figure 4-10: Profile page after submitting the changes	45
Figure 4-11: Courses page for a student	45
Figure 4-12: Courses page for a teacher	46

Figure 4-13: Add a course modal.....	46
Figure 4-14: Add a course modal showing errors.....	47
Figure 4-15: Add a course modal when submitting a new course	47
Figure 4-16: Courses page after a new course is added.....	48
Figure 4-17: Course page for the student.....	49
Figure 4-18: Course page for the teacher.....	49
Figure 4-19: Delete course popup.....	50
Figure 4-20: Edit course page	51
Figure 4-21: Edit course details form	51
Figure 4-22: Edit course form after submission	52
Figure 4-23: Delete lecture popup	52
Figure 4-24: Add lecture modal.....	53
Figure 4-25: Add announcement modal	53
Figure 4-26: Lecture form upload.....	54
Figure 4-27: Lecture form successful upload	54
Figure 4-28: Edit course page after the refresh.....	55
Figure 4-29: Lecture page.....	55
Figure 4-30: Lecture page after pressing the "Mark as done" button	56
Figure 4-31: Lecture page displaying a video lecture.....	56

List of tables

Table 3-1: Use Case description table	25
Table 3-2: Use Case diagram notations	27
Table 3-3: Entity relationship diagram notations.....	37

List of Abbreviations

API: Application Programming Interface

CORS: Cross-Origin Resource Sharing

CLI: Command Line Interface

CSS: Cascading Style Sheets

DOM: Document Object Model

HTTP: Hypertext Transfer Protocol

I/O: Input/Output

JWT: JSON Web Token

JSON: JavaScript Object Notation

LMS: Learning Management System

MOOC: Massive Open Online Course

NPM: Node Package Manager

NoSQL: Not Only SQL

REST: Representational State Transfer

RGB: Red Green Blue

SQL: Structured Query Language

UML: Unified Modeling Language

URL: Uniform Resource Locator

VPS: Virtual Private Server

XSRF: Cross-Site Request Forgery

ODM: Object Data Modeling

General Introduction

Programs are the building blocks of modern civilization [1]. Recent years have witnessed an evolution of digital technologies in the field of education. From simulators and virtual labs, video conferencing tools, and educational games, to e-learning platforms, the development rate was increasingly expanding due to the growing demand for flexible and accessible learning solutions, geographical constraints, and most importantly, the COVID-19 pandemic.

E-learning platforms provide online environments where learners can access educational content, interact with their instructors and classmates, and complete assignments and assessments remotely. From massive open online courses (MOOCs) to specialized learning management systems (LMS), e-learning platforms offer a wide range of options for individuals seeking to acquire new knowledge and skills.

Many students face geographical constraints that prevent them from attending traditional on-campus courses, whether due to distance, transportation issues, or work commitments. These challenges have been worsened by the COVID-19 pandemic, which has forced educational institutions worldwide to adapt to remote learning environments to ensure the safety of students and staff. As a result, there is an urgent need to overcome these geographical constraints by offering online courses through e-learning platforms, enabling students to pursue higher education regardless of their location. The transition to e-learning has become a necessity to ensure the continuity of education during these times and to provide students with the flexibility and accessibility they need to continue their academic pursuits amidst the challenges posed by the pandemic.

A web application is the most suited solution for addressing this need because it's easy to use, works on any device with an internet connection, and doesn't require installation. Students can access their courses from desktops, laptops, tablets, or smartphones. Moreover, website updates are automatic, so everyone always has the latest materials. Web apps can handle large numbers of users without extra costs, making them ideal for schools. Overall, they offer a convenient and adaptable platform for today's learners.

The "Student Portal" project aims to develop a comprehensive e-learning web application tailored for university students. It offers an intuitive, user-friendly platform that facilitates remote education through features like course management, announcements, and lectures. Built using cutting-edge web development technologies, namely the React and ExpressJs frameworks, this project ensures compatibility across all devices, while maintaining security measures aligned with university policies, and safeguarding user data through advanced authentication and web security protocols.

This report is divided into four main chapters. It begins with Chapter 1, offering a comprehensive overview of the project, including insights into web development and the significance of a student portal for e-learning. Chapter 2 dives into the tools and technologies utilized, ranging from development tools to the programming languages used. Additionally, Chapter 3 explores system modeling and database design, emphasizing the significance of effective data management. Chapter 4 provides a detailed account of the implementation process. We close our report with Future Enhancements, which outlines potential avenues for project development, and a Conclusion that reflects on key findings contributions.

Chapter 1: Web Development and E-learning Overview

1.1 Introduction

In today's digital era, technology has remarkably impacted education, reshaping traditional learning methods and offering new opportunities for knowledge sharing. This chapter begins with an overview of web development and its essential building blocks. Additionally, it discusses the importance of E-learning in addressing modern educational challenges. This introduction lays the groundwork for further exploration of our project's objectives and outcomes in subsequent chapters.

1.2 Web Development Overview

1.2.1 Definition

web development is a discipline that encompasses all the activities involved in creating, designing, putting online, and marketing maintenance of a website or web application. This can range from the creation of a simple static page (composed solely of text and images) to the creation of interactive sites with databases and advanced user interfaces [2].

Building a website includes the following stages:

- **Needs analysis Development:** the development team must first determine the objectives and expected functionalities of the site or application to be developed.
- **Design:** Once the first stage has been completed, the developers draw up the design and technical architecture of the website in line with the needs expressed above.
- **Development Programming:** web developers then work on programming and writing the code needed to implement the site's functionalities.
- **Testing and integration:** Before putting the site online, it is crucial to test that it works properly and to integrate the graphic elements created by the designers.
- **Going into production Development:** once the project has been finalized, the developers put the site or application online. This stage generally requires a good knowledge of web servers and hosts.

- **Maintenance and upgrading:** Finally, a good web developer must be able to maintain his work and develop it in line with user needs and new technologies [2].

1.2.2 Frontend and Backend

Frontend and Backend are the two most popular terms used in web development. Essentially, the difference between them is that the frontend of a website is what you see and interact with on your browser. Also referred to as client-side, it includes everything the user experiences directly: from text and colors to buttons, images, and navigation menus. And the backend (or “server-side”) is the portion of the website you don’t see. It’s responsible for storing and organizing data, and ensuring everything on the client side works. While these two types of programming are certainly distinct from one another, they’re also like two sides of the same coin. A website’s functionality relies on each side communicating and operating effectively with the other as a single unit [3]. The languages used for the frontend are HTML, CSS, and JavaScript while those used for the backend include Java, Ruby, Python, and Node.js.

1.2.3 Web Server and Web Client

The primary function of the Web Server is the storage, processing, and administration of websites. It is a computer system capable of storing both the website (HTML, JS, CSS, and media files, ...etc) and the user data as databases. Web servers handle incoming web requests and provide security to web applications through encryption and authentication mechanisms.

The Web Client, on the other hand, is a software application that communicates with the web server. By using Hypertext Transfer Protocol (HTTP) to perform the web request, it allows the end-users to interact with the website to request or send data. The web client can be either a browser, a phone application, or a desktop app.

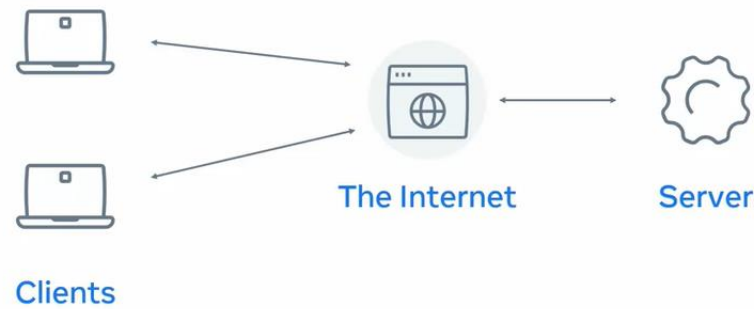


Figure 1-1: Web Server and Web Clients Relationship

1.2.4 Web Request

A web request is a request made by a client, such as a web browser, to a server in order to retrieve a web page or other resource. Web requests are sent using the Hypertext Transfer Protocol (HTTP), which is a standard protocol for transmitting data on the World Wide Web. When a user enters a URL into a web browser or clicks on a hyperlink, the browser sends a web request to the website's server. The server then responds by sending the requested resource back to the client [4].

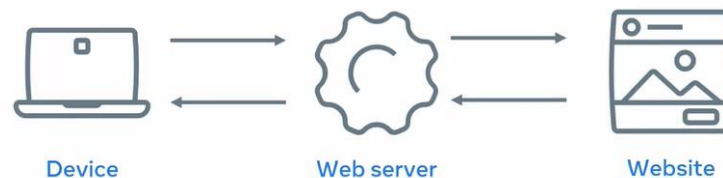


Figure 1-2: Web request

a) HTTP Request and Response

HTTP is the core operational protocol of the Internet. It is the set of rules governing the communication between the web server and the web clients. It is a request-response protocol. As for the HTTP request, there are five types as follows:

- **GET:** This type of request is made on behalf of a client when it's seeking data from a specified resource. Once the client has requested a resource, the server will process the request, get the information or data, and send it back to the client.

- **POST:** The purpose of POST is to send data or information to the server. It is often used when uploading a file or sending messages.
- **PUT:** This method allows the target to be entirely replaced with a new resource. PUT should be utilized to replace or overwrite a resource that the client is clearly aware of.
- **PATCH:** This method is simply used to modify a specific part of the resource. While it is similar to the PUT method, PATCH aims only to update or modify rather than replace.
- **DELETE:** This method sends a request to the server to delete a resource. While this is a possibility, it is not the most preferred choice [4].

A web request contains three parts, the request line, the request header, and the body.

A request line can be something like:

GET /home.html HTTP/1.1

On the other hand, the HTTP response contains a status code and message that explains the response type and acts as feedback to the HTTP request. The categories of the response are listed below:

- **Code Range [100, 199] Information:** Provides provisional information.
- **Code Range [200, 299] Successful:** The web request was successfully processed.
- **Code Range [300, 399] Redirect:** Moves the web page to a different path.
- **Code Range [400, 499] Client Error:** Either bad syntax or content.
- **Code Range [500, 599] Server Error:** Failure in server.

A web response also contains three parts, the response line, the header, and the body.

A response line may look like:

HTTP/1.1 200 OK

1.2.5 Web Hosting

Web hosting is an online service that enables you to publish your website or web application on the internet, namely on a web server. It can be either of the following cases:

- **Shared Hosting:** It is when many accounts are hosted in the same web server, which makes all websites share performance and storage. This is especially good for small websites and practice environments, and it can be free with the cost of running ads.
- **VPS Hosting:** VPS stands for Virtual Private Surface. This hosting grants each account a fixed amount of resources on the server, which allows the performance to stay stable. This web hosting is more expensive than the previous one.
- **Dedicated Hosting:** In this type of hosting, the account gets the full resources of the web server as its website is being hosted alone. It is the most expensive server hosting.
- **Cloud Hosting:** Cloud computing is the new trend of the internet, in which the website is hosted in a cloud environment (i.e., a collection of many virtual and physical servers). If one of the servers fails, the other works. It offers no limits on resources, which makes it have a relatively higher cost depending on the resources used.

1.2.6 Web Development Fundamental Technologies

The three fundamental building blocks of web development are HTML, CSS, and JavaScript. Think of a shop, HTML is the actual building, the structure, CSS is the interior decoration and landscaping outside, and JavaScript is just like the business, the services offered and the people coming in and out.

HTML stands for Hypertext Markup Language. It does not matter how big or complicated your website is going to be; you will always start with HTML. It is the standard language to create structures for the web [5]. Cascading Style Sheets or commonly known as CSS is the presentation part of a web page. HTML creates a structure, and CSS converts it into an attractive and more readable version. It is all about the color, style, and responsiveness of the web page. Lastly, JavaScript is a scripting language that allows you to control the behavior of your website and responses to user interactions. It is used on client-side as well as server-side. Further information will be discussed in the following chapter.

1.2.7 Data and Databases

The data layer is a large information warehouse. It includes a database repository that collects and saves data from the front end to the back end. It is where all the dynamic

information is stored. To manage this information effectively, it can be stored in different types of databases, which are primarily categorized into Relational and Non-Relational Models.

- **Relational and Non-Relational Databases**

A relational database, also called a Relational Database Management System (RDBMS) or SQL database, stores data in tables and rows also referred to as records. A relational database works by linking information from multiple tables through the use of “keys.” A primary key is a unique identifier for a row in one table, and when this key is added to a related record in another table, it becomes a foreign key. The connection between the primary and foreign key then creates the “relationship” between records contained across multiple tables. Some of the more popular SQL databases are MySQL, PostgreSQL, SQLite, and MariaDB [6].

The non-relational database, or NoSQL database, stores data. However, unlike the relational database, there are no tables, rows, primary keys, or foreign keys. Instead, the non-relational database uses a storage model optimized for specific requirements of the type of data being stored. There are five popular non-relational types: document store, column-oriented database, key-value store, and graph database. Often combinations of these types are used for a single application. Some of the more popular NoSQL databases are MongoDB, Apache Cassandra, Redis, and Apache HBase [6].

1.2.8 Libraries and Frameworks

It is possible to create a web page from the ground up, but it would take a long time, especially if additional complexity is needed to be added. JavaScript frameworks can be helpful in this situation. A framework for the website is similar to a pre-packaged structure of pre-written code that defines how programs should interact. Frameworks help develop faster and more efficiently. One of the most popular JavaScript frameworks is React [7].

Libraries, however, are reusable code that other developers have written to perform specific tasks or functions. Unlike frameworks, libraries do not impose a particular structure or architecture on the project, thus they are easier to replace, but instead, provide utility functions or modules that can be integrated into the code as needed. A Notable JavaScript Library is Mongoose.

1.2.9 Application Programming Interfaces (APIs) and REST API

API, or an Application Programming Interface, is an interface that connects one application to another through the end-points that the second offers. The first uses a key for each end-point to either access data or get a job done. APIs allow frontend developers to collect, modify, and delete data from a backend database.

An API is... I like to think of it as a digital librarian. Imagine having books stored in a library and you need to use these books for a report. So, you go to the librarian and ask to check out a book. You need to tell the librarian certain keywords (let's say the name of the book and the name of the author) and have the right authorization (in this case, a library card) to get this book. Now replace 'library' with 'server', 'books' with 'data', 'report' with 'website', and 'librarian' with 'API' [8].

Common types of APIs encompass browser APIs, RESTful APIs, and sensor-based APIs. a REST API (Representational State Transfer API) is a specific type of API that follows the Representational State Transfer guidelines. It is a simple, uniform interface that is used to make data, content, media, and other digital resources available through web URLs. REST API defines how clients can request and manipulate data from a server using standard HTTP methods like GET, POST, PUT, and DELETE.

1.2.10 The Document Object Model (DOM)

The DOM stands for Document Object Model. When the HTML file is loaded into a browser, a tree-like structure is created. This structure has various nodes, and these nodes represent various elements of the HTML document. The DOM structure of the HTML document will look like the following structure [5].

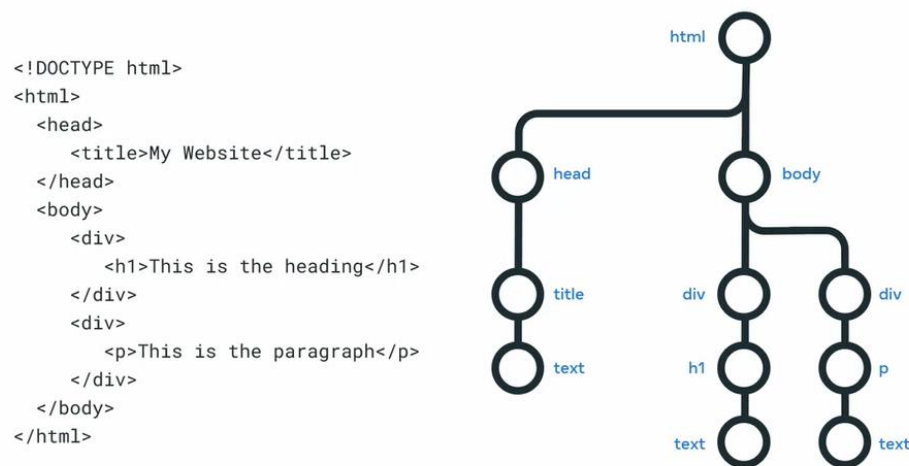


Figure 1-3: Document Object Model structure example

With JavaScript, it is possible to manipulate almost everything in HTML such as content and styles, and even add new elements and remove the existing ones, which helps the website be dynamic [5].

1.2.11 Static and Dynamic Websites

When you open a website, a web server sends the website's content to your browser. The content can be static or dynamic. Static content is files that the server transfers just as they are stored on the web server, such as videos or images. Dynamic content, on the other hand, is generated when the HTTP request is made. For example, the content may be generated based on input from a user, or when you visit a news website, it would be based on the current date. What actually happens, is that the web server communicates with another kind of server, called an application server or a back-end. The application server generates the dynamic content that the web server sends back to the user's browser. This process allows different users to see different contents on the web pages, which marks the website to be dynamic.

1.2.12 Single Page Websites

The best request is no request, both for the client and the server [9]. Single-page web applications are a relatively new idea. Instead of a website requiring a network request every time the user navigates to a different page, a single-page web application downloads the entire site (or a good chunk of it) to the client's browser. After that initial download, navigation is faster because there is little or no communication with the server. Single-page

application development is facilitated by the use of popular frameworks such as React or Angular [10].

1.3 Student Portal and E-Learning

1.3.1 Definition

E-learning is a fast and efficient way of providing and sharing knowledge with learners in different parts of the world. It uses the Internet or other digital content for learning and education activities and takes full advantage of modern educational technology to provide a new mechanism for communication and a learning environment rich in resources to achieve a new way of learning [11] [12].

1.3.2 Existing E-Learning Platforms

- **EducateMe**

EducateMe is a collaborative LMS platform designed to help instructors create hitch-free learning sessions and students learn simultaneously. For effective collaboration, EducateMe offers great peer-review options, group assignments, and in-built engagement tools. Besides, it has progressive cohort management and reporting, as you can track all student assignments from the unified dashboard, displaying deadlines and the progress of students [13].

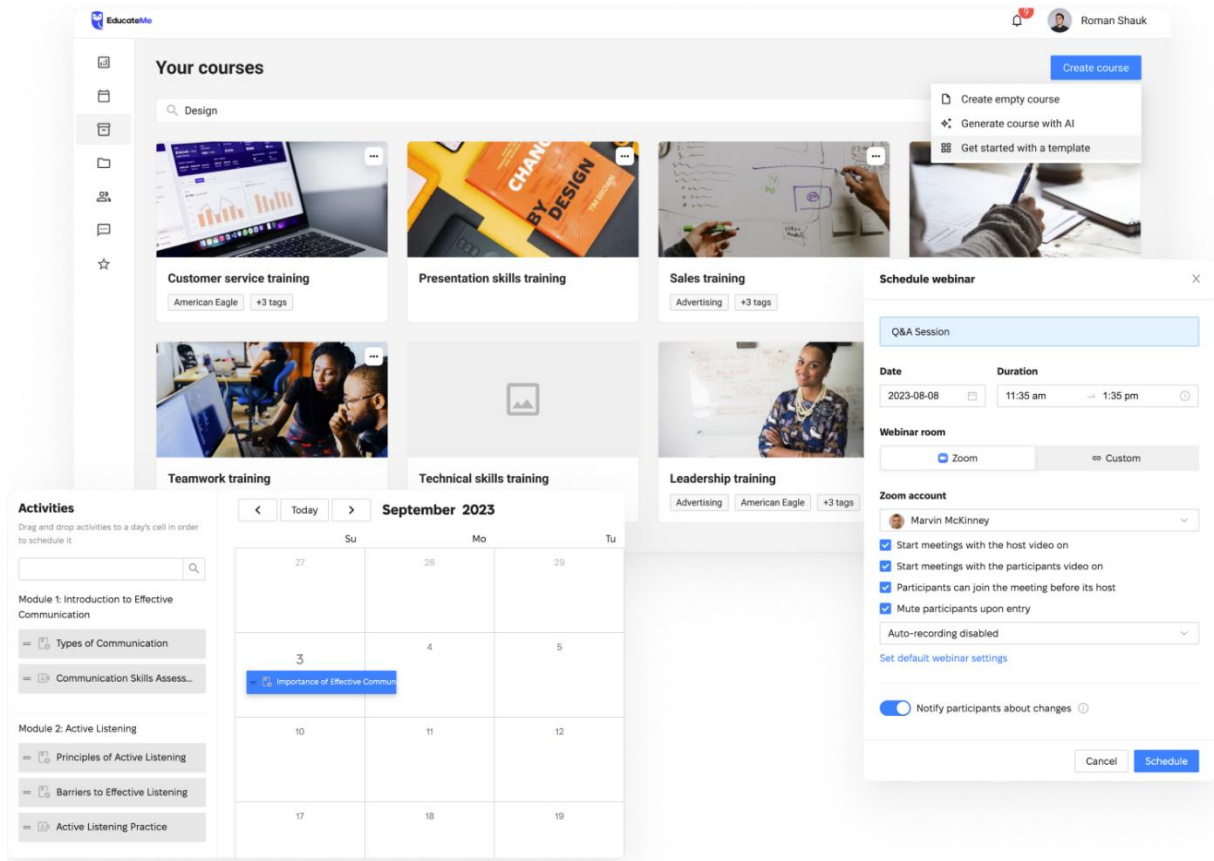


Figure 1-4: EducateMe website

- **Google Classroom**

Google Classroom is a free teaching and learning service for schools, non-profits, and anyone with a Google account. The focus on assignments and collaboration makes it suitable for high school and college classrooms, as well as for distance learning and flipped classrooms. For students, Google Classroom is a convenient tool to access their assignments and course materials, submit papers, and communicate with their teachers and classmates [13].

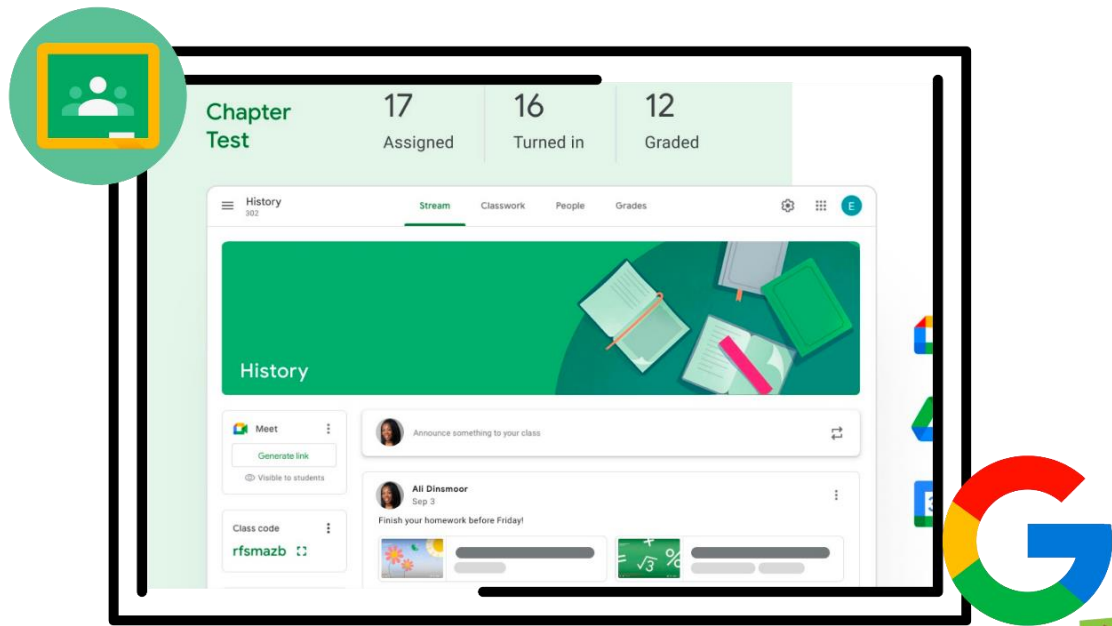


Figure 1-5: Google Classroom website

- **Moodle Platform**

Moodle is an open-source, feature-rich, secure, and scalable learning management system that integrates seamlessly with other platforms and can be customized for any teaching or training method you choose. The software is used by over 30 million students around the world. Moodle allows trainers to create their course materials and add them to a virtual library [13].

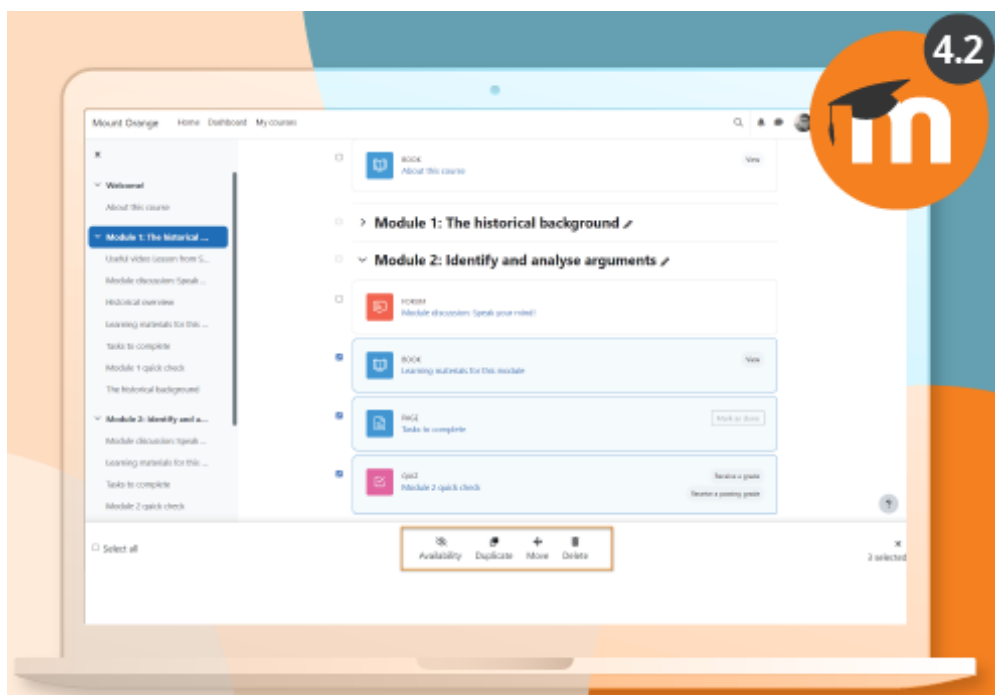


Figure 1-6: Moodle Platform

1.4 Conclusion

In this chapter, we had an overview of web development and its essential building blocks. Furthermore, we discussed the importance of E-learning in addressing modern educational challenges. This chapter served as a foundation for the next chapter, in which we will cover the specific tools used in this project.

Chapter 2: Tools and Technologies

2.1 Introduction

This chapter introduces the tools and technologies used in the project. It covers development tools like Visual Studio Code, NPM, Git, and GitHub, along with front-end technologies such as HTML, CSS, React, and Redux. It also explores back-end technologies, including NodeJs, ExpressJs, MongoDB, and security tools like JSON Web Token and BcryptJs.

2.2 Modeling Tools

2.2.1 Modeling Language

A modeling language is any artificial language that can be used to express information, knowledge, or systems in a structure that is defined by a consistent set of rules. The rules are used for interpretation of the meaning of components in the structure. A modeling language can be graphical or textual. Graphical modeling languages use diagram techniques with named symbols that represent concepts and lines that connect the symbols and that represent relationships and various other graphical annotations to represent constraints. Textual modeling languages typically use standardized keywords accompanied by parameters to make computer-interpretable expressions [14].

2.2.2 Unified Modeling Language (UML)

Unified Modeling Language (UML) is a general-purpose modeling language that is an industry standard for specifying software-intensive systems. UML 2.0, the current version, supports thirteen different diagram techniques and has widespread tool support [14]. UML diagrams are used to portray the behavior and structure of a system. They can be categorized into two main types:

- a) **Structural Diagrams:** These represent the static aspects of a system, such as the organization of system components and their relationships. They include Class Diagrams and Object Diagrams.

- b) **Behavioral Diagrams:** These focus on the dynamic aspects of the system, such as how it responds to inputs and the flow of control and data. Examples include Use Case Diagrams and Sequence Diagrams.

2.3 Development Tools

2.3.1 Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor that runs on your desktop and is available for Windows, macOS, and Linux. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other languages and runtimes (such as C++, C#, Java, Python, Go, .NET) [15]. It is the most used code editor for web development.

2.3.2 Node Packet Manager (NPM)

NPM is the world's largest software registry. Open-source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well [16]. Simply put, NPM is like a giant online store for software tools and libraries that developers use to build websites and applications.

npm consists of three distinct components:

- **The website:** Use the website to discover packages, set up profiles, and manage other aspects of your npm experience. For example, you can set up organizations to manage access to public or private packages.
- **The Command Line Interface (CLI):** The CLI runs from a terminal, and is how most developers interact with npm.
- **The registry:** The registry is a large public database of JavaScript software and the meta-information surrounding it [16].

2.3.3 Web Browser

A web browser is a software application used to access information on the World Wide Web. It allows users to view web pages, browse websites, and interact with online content. Web browsers use the Hypertext Transfer Protocol (HTTP) to request web pages

from web servers and display them on the user's device. They interpret HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and JavaScript code to render web pages with text, images, videos, and interactive elements. Popular web browsers include Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, and Opera.

2.3.4 Git and GitHub

Version control, also known as source control, is the practice of tracking and managing changes to software code. Version control systems are software tools that help software teams manage changes to source code over time. As development environments have accelerated, version control systems help software teams work faster and smarter [17].

Git, the most widely used modern version control system, is essential for tracking and controlling code changes, as well as facilitating collaboration. It manages projects with repositories¹ and allows one to work on different parts and versions of a project by offering branches and a merging ability. GitHub, a cloud-based platform that extends Git's functionality, provides additional tools for project management, enabling developers to store repositories online, collaborate more easily, and manage multiple project versions efficiently.

2.4 Front-End Tools

2.4.1 Hyper Text Markup Language (HTML)

The basic programming language for web creation is HTML. It contains the essential elements of a website, such as words, titles, images, links, and paragraphs. HTML is made up of a set of pre-defined tags that represent various functions and subsequently "translate" into understandable information on the screen. These tags are always written between angle brackets [7].

¹ Repositories are central locations in which data is stored and managed.



Figure 2-1: HyperText Markup Language

2.4.2 Cascading Style Sheets (CSS)

Cascading Style Sheets, or CSS, is a style sheet that describes how HTML components appear on a page. CSS is used to manage your website's appearance, style, and formatting, including RGB values, border colors, background pictures, and more. CSS files specify a set of rules for defining a set of properties and their values [7].



Figure 2-2: Cascading Style Sheets

Additionally, CSS plays a crucial role in making websites responsive, ensuring that web pages adapt and display properly on various devices and screen sizes. By using CSS media queries, flexible grid, and fluid images, developers can create layouts that adjust dynamically to provide optimal viewing experiences across desktops, laptops, tablets, and smartphones.

2.4.3 Tailwind CSS

Tailwind CSS is a utility-first CSS framework that streamlines web development by providing a set of pre-designed utility classes. These classes enable rapid styling without writing custom CSS, promoting consistency and scalability. Tailwind's approach shifts focus from traditional CSS components to functional classes, empowering developers to efficiently build responsive and visually appealing interfaces with minimal effort [18].

2.4.4 JavaScript

JavaScript is a scripting language that allows you to control the behavior of your website. It is one of the most widely used programming languages in the world, with a low entry barrier and instant results based on your code's success. By manipulating different HTML and CSS elements, JavaScript makes web pages interactive. Using JavaScript, a user may click a button, scroll to the bottom of a page, or see pictures in a rolling carousel [7].



Figure 2-3: JavaScript

2.4.5 React

React is a free and open-source front-end JavaScript framework for building user interfaces based on components. It is maintained by Meta and a community of individual developers and companies.

React lets the developer build user interfaces out of individual pieces called components. Create his own React components like Thumbnail, DeleteButton, and Modals. Then combine them into entire screens, pages, and apps [19]. Whether he works on his own or with thousands of other developers, using React feels the same. It is designed to let him seamlessly combine components written by independent people, teams, and organizations [20].

2.4.6 Redux

Redux is a JS library for predictable and maintainable global state² management. It helps you write applications that behave consistently, run in different environments (client,

² State refers to an object that holds the data or information that a component or application needs to manage and render. State is dynamic, meaning it can change over time based on user interactions, API responses, or other factors.

server, and native), and are easy to test [21]. Redux provides a structured way to manage states across the entire application, making it easier to handle complex state interactions and ensuring that state changes are reflected on various parts of the application.

2.4.7 Axios

Axios is a client HTTP API and library based on the “XMLHttpRequest” interface provided by browsers. It allows developers to make simple requests to either their server or a third-party server to fetch data while ensuring wide browser support. It can be used to intercept HTTP requests and responses and enables client-side protection against XSRF, and it also can cancel requests. The key difference between using Axios and using the native API lies in the level of abstraction and ease of use in addition to the additional features Axios offers.

2.5 Back-End Tools

2.5.1 NodeJs

Node.js is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools, and scripts [22]. Node.js runs JavaScript by utilizing V8, Google’s fast JavaScript engine designed for Chrome. This allows Node.js to create a runtime environment that pushes JavaScript from the server to the client quickly. Node.js has an event-driven architecture capable of asynchronous I/O that is provided by the libuv C library. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications [22].

2.5.2 ExpressJs

Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications [23]. It is the underlying library for several other popular Node.js frameworks. It provides mechanisms to:

- Write handlers for requests with different HTTP verbs (methods) at different URL paths (routes).
- Integrate with "view" rendering engines to generate responses by inserting data into templates.

- Add additional request processing "middleware" at any point within the request handling pipeline.
- Scale applications by organizing routes, middleware, and logic into modular components, making it ideal for building both small APIs and large, complex web applications [24].

2.5.3 MongoDB

MongoDB is an open-source document-oriented non-relational database that is designed to store a large scale of data and also allows the user to work with that data very efficiently. It is a powerful and flexible solution for handling modern data needs. Unlike traditional relational databases, MongoDB's document-oriented architecture allows for greater agility and scalability, making it a preferred choice for businesses and developers aiming to handle large volumes of unstructured or semi-structured data. Because of its NoSQL model, the data is stored in collections and documents. Hence the database, collection, and documents are related to each other as shown below [25]:

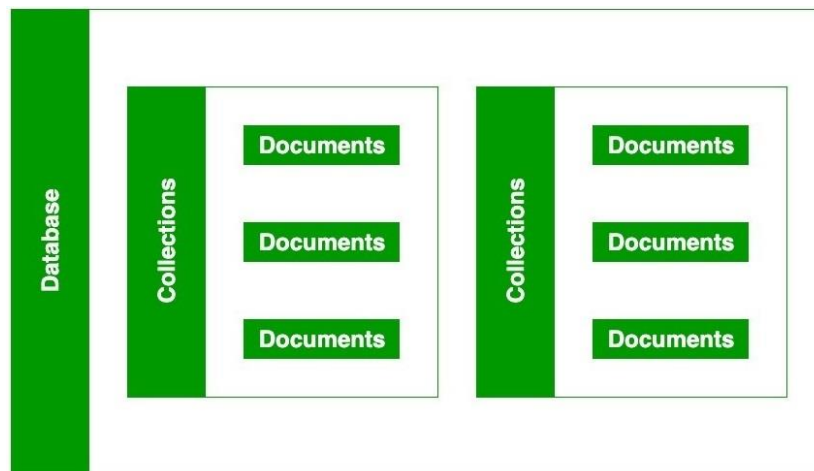


Figure 2-4: MongoDB data storage

MongoDB was chosen for this project for its ability to handle unstructured and dynamic data, offering flexibility without predefined schemas. This is ideal for educational platforms, which deal with diverse and dynamic data like courses, assignments, multimedia, and user interactions. Additionally, MongoDB's scalability and support for horizontal scaling make it ideal for handling large volumes of data or high-traffic applications. It is optimized for high-performance operations, making it well-suited for the real-time data processing needs of an e-learning website, such as submitting assignments, participating in discussions, or accessing course materials.

2.5.4 Mongoose

Mongoose is a JavaScript ODM (Object Data Modeling) library that creates a connection between MongoDB and the Node.js JavaScript runtime environment. It provides a straightforward, schema-based solution to model application data [26].

2.5.5 Multer

Multer is a Node.js middleware for handling multipart/form-data. It simplifies the process of handling file uploads in Node.js making it an essential tool for developers dealing with user-generated content [27]. It can process both single and multiple file uploads. It also provides options to filter files, set size limits, and manage file storage.

2.5.6 JavaScript Object Notation Web Token

JSON Web Token, or JWT, is an open standard (RFC 7519³) used to create compact, self-contained tokens for securely transmitting information between different applications or services. These tokens are typically used for authentication and authorization, as they can contain information that verifies the identity of a user, and their permissions. In terms of authentication, the information stored in the JWT is used to help servers establish trust between an unknown client and themselves. JWTs work by encoding a set of claims into a compact, URL-safe string. This string can be easily transmitted over the network and verified by the receiver. The three main components of a JWT are the:

- Header
- Payload
- Signature

With these three components, JWTs allow developers to build a stateless authentication or authorization flow that is easily scalable and eliminates the need for servers to maintain session information. All three of these parts are Base64Url encoded strings concatenated with periods ('.') [28].

³ RFC stands for Request for Comments. It is a series of documents that describe various specifications, protocols, and standards related to the Internet and computer networking published by the Internet Engineering Task Force (IETF). RFC 7519 is a technical specification document that outlines the standards and guidelines for JSON Web Tokens (JWT).

2.5.7 BcryptJs

Bcrypt is a password-hashing⁴ function designed by Niels Provos and David Mazières, based on the Blowfish cipher and presented at USENIX in 1999. The bcrypt hashing function allows developers to build a password security platform that scales with computation power so it remains resistant to brute-force search attacks. And always hashes every password with a salt to protect against rainbow table attacks.

2.5.8 Cross-Origin Resource Sharing (CORS)

Cross-Origin Resource Sharing, or CORS, is an HTTP-header-based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources. It helps prevent unauthorized access to resources from websites that a browser shouldn't trust by default [29]. CORS is also a node.js package for providing an Express middleware that can be used to enable the CORS mechanism with various options.

2.6 Conclusion

This chapter provided an overview of the key tools and technologies used in the development process. By covering both front-end and back-end frameworks, along with essential development tools, it established a solid foundation for building and managing the project's infrastructure.

⁴ Password hashing is the practice of algorithmically transforming plain-text passwords into secure, fixed-length hashes that are difficult to reverse-engineer, as a means of blocking against the threat of hacking.

Chapter 3: System Design

3.1 Introduction

This chapter focuses on the design phase of the system. Beginning with system modeling, it introduces key elements such as system actors, use-case descriptions, identifications, and diagrams to visually represent system interactions. The chapter then transitions to database design, providing a detailed description of the database structure and the relationships between documents. This sets the stage for understanding the system's functionality and data flow.

3.2 System Modeling

3.2.1 System Actors

System actors are the entities that interact with the system. In our system, the primary actors include:

- a) **Teacher:** The teacher creates and manages courses and their content, creates assignments, and grades students' submissions.
- b) **Student:** The student accesses his courses and their content, views his assignments, and submits his work for evaluation.

Actors are typically represented in use case diagrams in UML, where they show how these external entities engage with the system's functionality.

3.2.2 Use Case Description

Use cases represent the actions that the system offers to actors to help them accomplish specific objectives. The table below summarizes all the use cases for each actor, based on their respective roles within the system.

Table 3-1: Use Case description table

Actors	Use Cases
Teacher	<ul style="list-style-type: none"> a) Login: Authentication b) View profile information: Full name, profile image, role, e-mail, and teacher ID. c) Edit profile: E-mail, password, and profile Image. d) View courses: View his own courses. e) Manage courses: Create edit and delete courses. f) View course materials: Lectures, announcements, and assignments. g) View students: View students for each course he teaches. h) Manage lectures: Create and delete lectures. i) Manage announcements: Create and delete announcements. j) Manage assignments: Create and delete assignments. k) Evaluate assignment submissions: View student submissions and give each one a mark.
Student	<ul style="list-style-type: none"> a) Login: Authentication. b) View profile information: Full name, profile image, role, e-mail, student ID, academic year, academic level, major, and group. c) Edit profile: E-mail, password, and profile Image. d) View courses: View the courses he studies during the current academic year. e) View course materials: Lectures, announcements., and assignments. f) View classmates: View his classmates in each course. g) Mark lectures as done: Mark what he has already studied as done. h) Submit work on assignments: Submit his answers to the assignment questions. i) View Marks: View his marks on assignments.

3.2.3 Use Case Identification






- a) **Login:** The user needs to enter his e-mail and password correctly to get authenticated and access the website.
- b) **View profile information:** The user can view his information that was pre-registered on the website by the administration. This information includes his full name, profile picture, role as a teacher or student, e-mail, and user ID, while the student can also view his academic year, academic level, major, and group.
- c) **Edit profile:** The user is allowed to edit his login credentials i.e., e-mail and password, and his profile picture.

- d) **View courses:** The teacher can view all the courses he teaches, and the student can see the courses he studies during the current academic year.
- e) **Manage courses:** The teacher can create new courses by giving them a title and description and specifying to which academic year, level, and major they are for. The teacher can also edit this information at any time, or delete any course he owns from the database.
- f) **View course materials:** The user can see the lectures, announcements, and assignments of the courses he can access. He can also view the lecture content that is registered in either a video (.mp4, .avi, or .mov) or a pdf (.pdf) format.
- g) **Mark lectures as done:** The student can mark lectures that he has studied through the platform as done so that he can ... his progress.
- h) **View students/classmates:** The teacher can see the student names and profile pictures of each course he teaches, and the students can see their classmates as well.
- i) **Manage lectures:** The teacher can add new lectures to any of his courses by submitting the lecture file in either a video (.mp4, .avi, or .mov) or a pdf (.pdf) format, accompanied by a title and a description. He can also delete them anytime.
- j) **Manage announcements:** The teacher can add new announcements about the course and delete them anytime.
- k) **Manage assignments:** The teacher can create assignments by specifying a title, description, due date, and any related resources or attachments in PDF format. He can also delete assignments as needed.
- l) **Evaluate assignment submissions:** The teacher can review students' submitted assignments, provide feedback, and assign grades. This evaluation will be recorded in the system for students to view.
- m) **Submit work on assignments:** Students must submit their assignments by uploading the required files (in PDF format) before the deadline. They can still submit late though. They can also review their submissions.
- n) **View Marks:** Students can view the grades and feedback provided by the teacher for their assignments.

3.2.4 Use Case Diagrams

Use case diagrams highlight the relationship between actors and their respective use cases. A single use case diagram represents a specific functionality of the system. These diagrams are especially important in organizing and modeling the behaviors of a system. Notations used when representing use case diagrams include the following:

Table 3-2: Use Case diagram notations

Figure	Name	Explanation
	Actor	The entities (users or systems) that interact with the system.
	Use Case	The specific functions or actions the system performs.
	Association	The lines connecting actors to use cases, indicate their interaction.
	Include	A relationship between use cases, it is used when a use case always calls or depends on another use case as part of its execution.
	Extend	A relationship between use cases, it is used when a use case conditionally adds extra behavior to another use case.

The following figures show the use case diagram for each actor and the overall use case for the system.

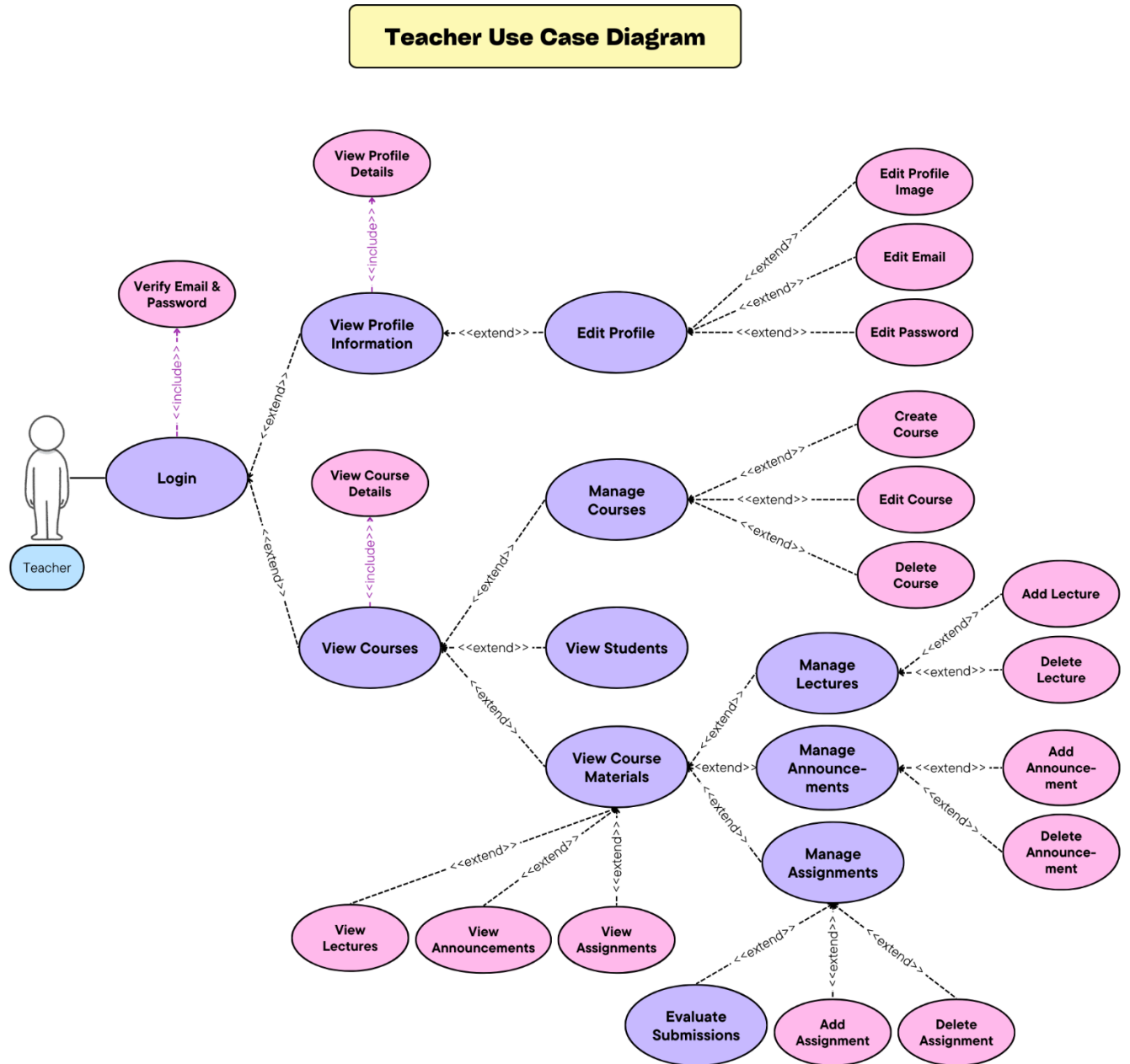


Figure 3-1: Teacher use case diagram

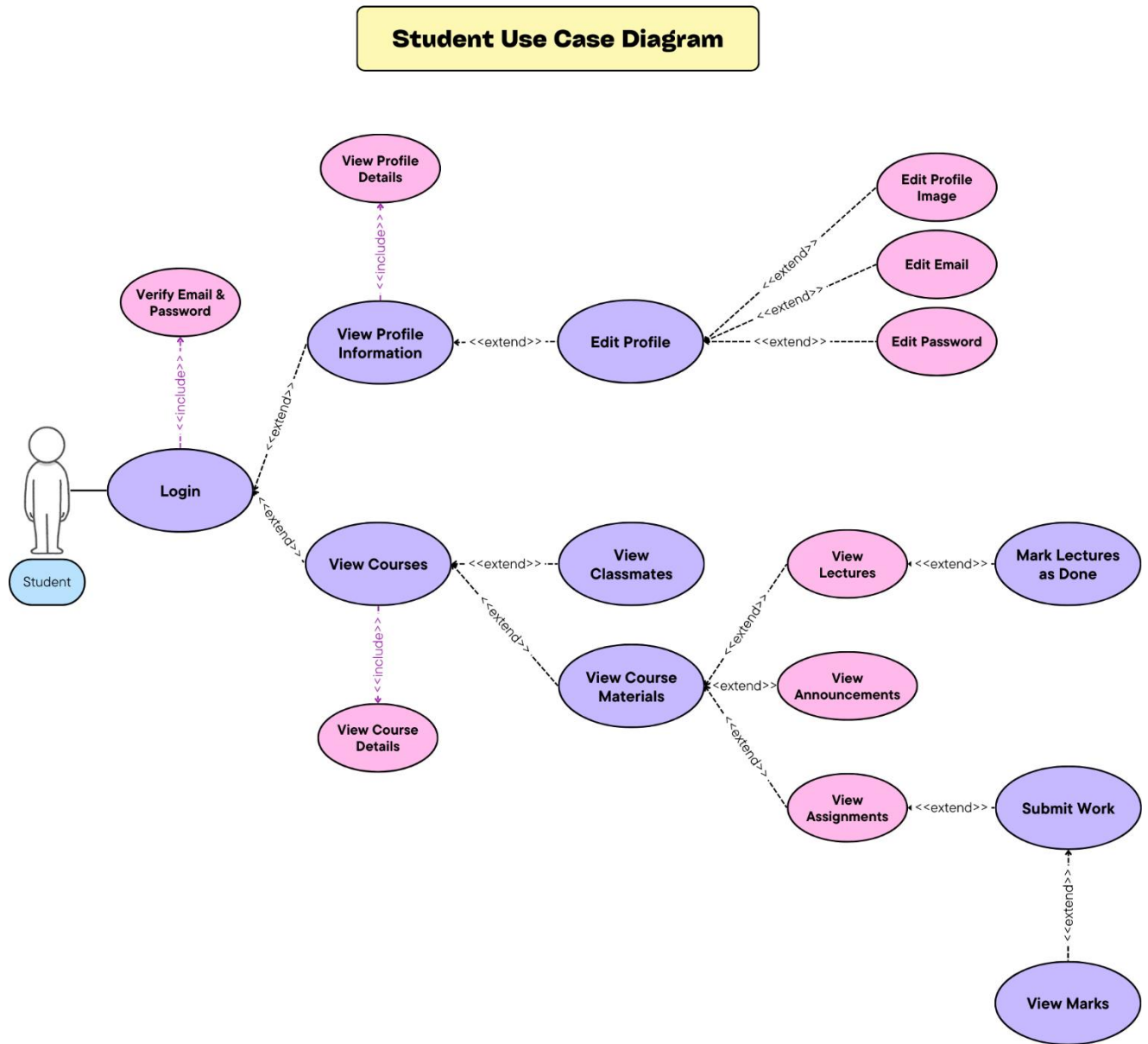


Figure 3-2: Student use case diagram

3.3 Database Design

3.3.1 Database Description

Our database system contains five collections, which are the user, course, announcements, assignments, and submissions collections.

Collection Name	Documents	Logical Data Size	Avg Document Size	Storage Size	Indexes	Index Size	Avg Index Size
announcements	3	490B	164B	36KB	1	36KB	36KB
assignments	0	0B	0B	4KB	1	4KB	4KB
courses	6	5.48KB	936B	36KB	1	36KB	36KB
submissions	0	0B	0B	4KB	1	4KB	4KB
users	11	4.2KB	391B	36KB	3	108KB	36KB

Figure 3-3: Database collections interface

a) User Collection

Documents in this collection contain twelve attributes⁵:

- **_id**: A unique identifier for each document in a collection auto-generated by the database.
- **firstName**: The first name of the user.
- **lastName**: The last name of the user.
- **userID**: The user's personal ID given by the university.
- **email**: The email of the user.
- **password**: The password of the user.
- **role**: The role of the user in the university; either a student or a teacher.
- **academicLevel**: The academic level of the user; either Licence or Master.
- **academicYear**: The academic year the user is currently in.
- **major**: The major of the user if he was a Master's student.
- **group**: The user's group within the current academic program.

⁵ There is in fact an additional attribute in all of the collections called `__v` that represents the version of the document. It's automatically added by Mongoose, and when documents are updated, it increments to help prevent concurrent data modification issues.

- **profileImage:** The profile picture of the user.

```
_id: ObjectId('669fb9ecbf059b0da5fcc5d4')
firstName: "Imane"
lastName: "Otmanine"
email: "otmanine.imane@gmail.com"
password: "$2a$10$MyBHTQl75hU01DXEATzzo03YBSUmmWZ43BgIrtqig/snonoRgMpuu"
role: "student"
__v: 0
academicLevel: "Master"
academicYear: 2
group: 1
major: "Computer Engineering"
profileImage: "https://img.freepik.com/free-photo/young-beautiful-woman-pink-warm-swe..."
userID: "191932037991"
```

Figure 3-4: User document example

b) Course Collection

Documents in this collection contain eleven attributes:

- **_id:** A unique identifier for each document in a collection, auto-generated by the database.
- **title:** The title of the course.
- **description:** A brief description of the course.
- **teacher:** An id reference to the teacher collection, it indicates the teacher of the course.
- **academicLevel:** The intended educational level for the course.
- **academicYear:** The academic year the course is designated for.
- **major:** The major the course is associated with.
- **files:** An array of the lectures that belong to the course. Each file contains the following attributes:
 - **_id:** A unique identifier for each file in the files array.
 - **fileName:** The name of the file attached to the lecture.

- **fileType:** The format of the file attached to the lecture. It can be either “application/pdf” or “video/mp4”.
- **lectureName:** The name of the lecture.
- **description:** A brief description of the lecture.
- **completionStatus:** An array of statuses, one for each student, indicating whether he has completed studying a specific lecture.
 - **_id:** A unique identifier for each status in the completionStatus array.
 - **student:** An id reference to the student collection, indicating the student associated with the completion status for this lecture.
 - **completed:** The status of completion of this lecture, can be either True or False.
 - **markedAt:** The time when the student marks the lecture as completed.
- **created_at:** The time when the lecture was created.
- **announcements:** An array of the announcements that belong to the course. Each announcement is an id reference to the announcements collection, indicating an announcement that is part of this course.
- **created_at:** The time at which the course was created, automatically added by the database.
- **updated_at:** The time at which the course was modified, automatically updated by the database.

```

_id: ObjectId('66d74a3193f12b7316fb05c2')
title: "Active Devices I"
description: "The main goal of this course is to provide students with an understand..."
teacher: ObjectId('66bce166cd9fe0c65b23019c')
academicLevel: "Licence"
academicYear: 2
▼ announcements: Array (2)
  0: ObjectId('66e72cb4b7e56a409cc6d35d')
  1: ObjectId('66e72ed6b7e56a409cc6d49b')
▼ files: Array (16)
  ▼ 0: Object
    lectureName: "Introduction to semiconductor materials"
    filename: "1726425889886-Semiconductor materials.pdf"
    fileType: "application/pdf"
    description: "This lecture is an introduction to semiconductor materials."
    _id: ObjectId('66e72b22b7e56a409cc6d27a')
    created_at: 2024-09-15T18:44:50.072+00:00
    ▼ completionStatus: Array (4)
      ▼ 0: Object
        student: ObjectId('66aa14a02d3e746c265ab957')
        completed: true
        _id: ObjectId('66e8556dad2db3430eb25120')
        markedAt: 2024-09-16T15:57:33.981+00:00
      ▶ 1: Object
      ▶ 2: Object
      ▶ 3: Object
    ▶ 1: Object
    ▶ 2: Object
    ▶ 3: Object
    ▶ 4: Object
    ▶ 5: Object
    ▶ 6: Object
    ▶ 7: Object
    ▶ 8: Object
    ▶ 9: Object
    ▶ 10: Object
    ▶ 11: Object
    ▶ 12: Object
    ▶ 13: Object
    ▶ 14: Object
    ▶ 15: Object
  createdAt: 2024-09-03T17:41:05.842+00:00
  updatedAt: 2024-09-16T16:05:18.409+00:00
  __v: 26

```

Figure 3-5: Course document example

c) Announcements Collection

Documents in this collection contain five attributes:

- **_id:** A unique identifier for each document in a collection auto-generated by the database.

- **content:** The content of the announcement.
- **teacher:** An id reference to the teacher collection, indicates the teacher who made the announcement.
- **courses:** An array of the courses where the announcement should be posted. In this application typically it is an array of one element.
- **created_at:** The time when the announcement was made. It is automatically added by the database.

```
_id: ObjectId('66e72ed6b7e56a409cc6d49b')
content: "This Thursday we will have the control correction session at 2:30 PM i..."
teacher: ObjectId('66bce166cd9fe0c65b23019c')
▼ courses: Array (2)
  0: ObjectId('66d74a3193f12b7316fb05c2')
  1: null
created_at: 2024-09-15T19:00:38.578+00:00
__v: 0
```

Figure 3-6: Announcement document example

d) Assignments Collection

Documents in this collection contain eight attributes:

- **_id:** A unique identifier for each document in a collection auto-generated by the database.
- **title:** The title of the assignment.
- **description:** The description of the assignment.
- **fileName:** The attached file to the assignment, contains the assignment questions.
- **deadline:** The date and time by which the assignment must be completed and submitted.
- **course:** An id reference to the course collection, indicates the course to which this assignment belongs.
- **teacher:** An id reference to the teacher collection, indicates the teacher who posted this assignment.

- **created_at:** The time at which the assignment was added, automatically recorded by the database.

e) Submissions Collection

Documents in this collection contain eight attributes:

- **_id:** A unique identifier for each document in a collection auto-generated by the database.
- **assignment:** An id reference to the assignment collection, indicating the assignment to which the submission pertains.
- **student:** An id reference to the student collection, indicating the student who submitted this answer.
- **fileName:** The file submitted by the student containing his answers to the assignment. It must be in “application/pdf” format.
- **mark:** The mark the teacher gives the student on his submitted work.
- **notes:** The teacher’s additional notes on the answer of the student.
- **submitted_at:** The time when the student submits his work on the assignment.
- **marked_at:** The time when the teacher gives a mark to the student for his work.

3.3.2 Relationships Between Documents

In MongoDB, relationships define how different documents are logically connected. These relationships can be modeled using either the Embedded or Referenced approaches and can represent various types, such as 1:1, 1:N, N:1, or N: M.

- a) **Embedded Relationships:** Embedded documents store related data in a single document structure. A document can contain arrays and sub-documents with related data [30]. The following is an example of embedded data in a document.



Figure 3-7: Embedded data in a document

b) Referenced Relationships: References store relationships between data by including links, called references, from one document to another. For example, a teacher field in a course collection indicates a reference to a document in a user's collection [30]. The following is an example of referenced data in a document.

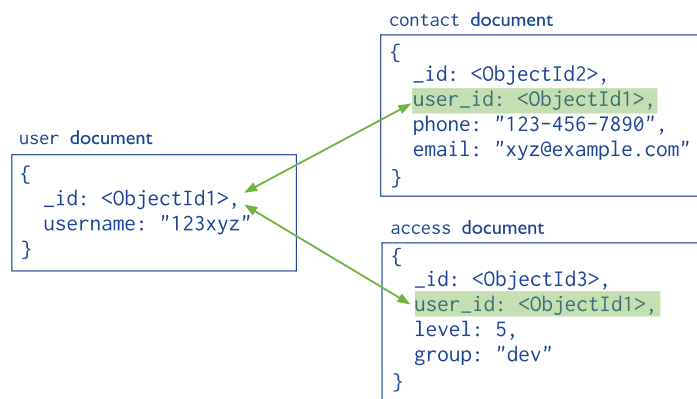


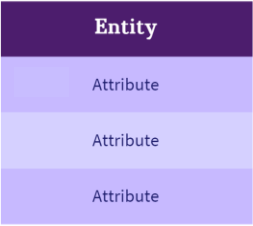



Figure 3-8: Referenced data in a document

• Entity Relationship Diagram

An Entity Relationship Diagram visualizes the relationships between entities, such as people, objects, or concepts, within a database. It typically illustrates the entities, their attributes, and the connections between them, providing a clear view of the database's logical structure.

The following table summarises the notations used in the entity relationship diagram.

Table 3-3: Entity relationship diagram notations

Symbol	Name	Description
	Entity	A rectangle represents a collection of documents and their attributes. The attributes underlined illustrate the primary key of a document and the ones that have * sign are foreigner id references. Each document within the collection represents an instance of that entity.
	One and only one	A straight line with “1..1” on top indicates a single instance in a relationship.
	Zero or many	A line with “0..*” on top shows multiple instances in a relationship.
	Generalization/ Specification	An arrow indicates a generalization or specification between two entities illustrating the inheritance.

The entity relationship diagram of this system is represented in the figure on the next page.

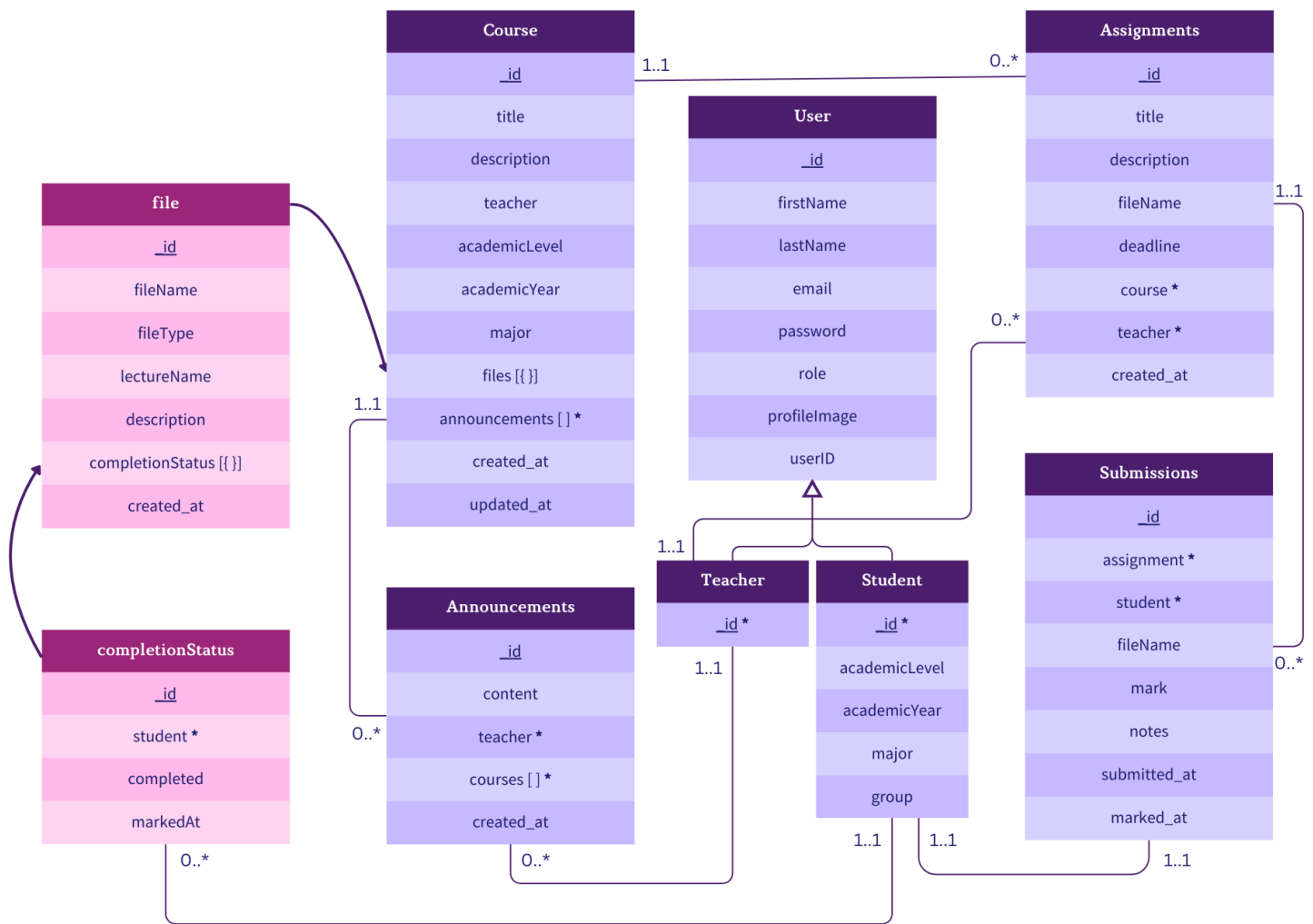


Figure 3-9: Entity Relationship Diagram

3.4 Conclusion

In conclusion, this chapter presented the foundational elements of system design, including system modeling through use case diagrams, and a detailed approach to database design. These components are critical for structuring the system and ensuring proper data management, forming the basis for the project's implementation in the next chapter.

Chapter 4: Implementation

4.1 Introduction

This chapter focuses on the implementation of the application interfaces, detailing the key components that make up the user experience. It begins with the login interface, followed by the dashboard, profile, and various course-related interfaces. Each section outlines the design and functionality of these interfaces, which play a crucial role in how users interact with the application.

4.2 Application Interfaces

Using the tools outlined in the previous chapters, we have implemented the application to address the problem introduced at the start of this report. The following section presents the different interfaces that users will engage with.

The frontend of this website is hosted on Github Pages and is accessible through the link: “<https://imonyaa.github.io/Student-Portal>”. In contrast, the backend is hosted on a VPS provided by DigitalOcean, maintaining the necessary server-side functionality and database management to support the application's features.

The available users who can access the platform include the following:

Student e-mails: otmanine.imane@gmail.com, roseonii@gmail.com, tod@gmail.com.

Teacher e-mails: leila@gmail.com, ivan@gmail.com, racha@gmail.com.

Passwords: password123

4.2.1 Login Interface

Initially, when the user first enters the website, he encounters the login page shown on the next page.

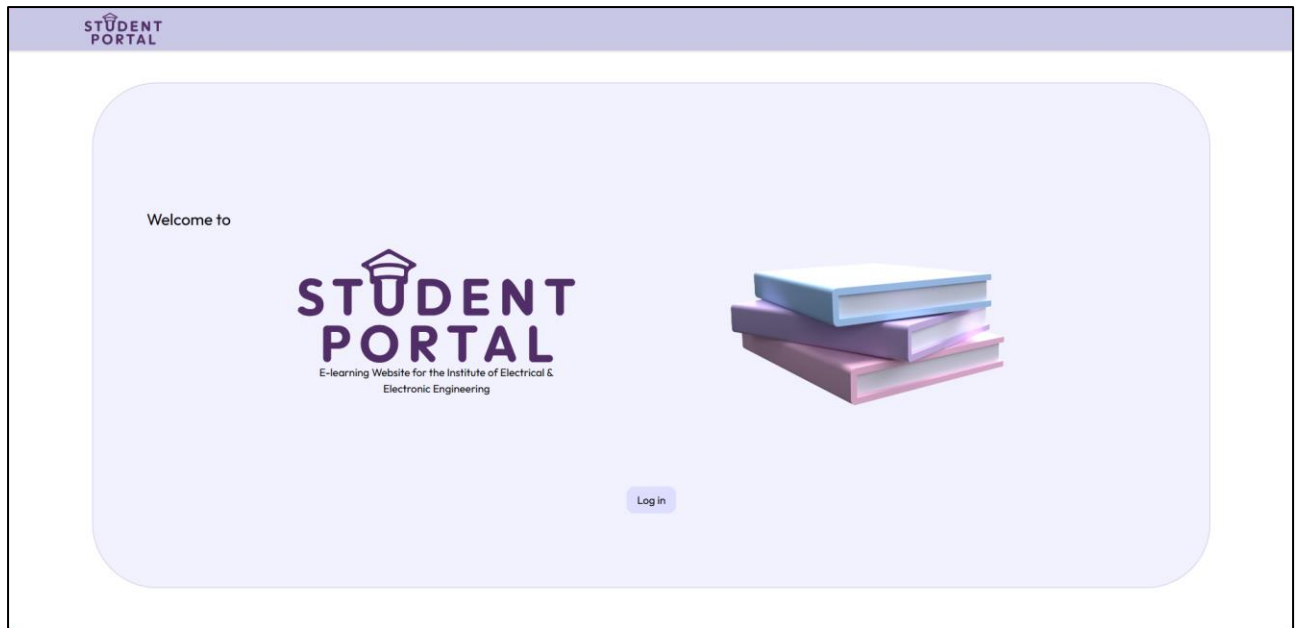


Figure 4-1: Login initial page

After pressing the "Log in" button, the user will see the login form. Since the users' collection documents are pre-filled by the administration, users are pre-registered, and their information is already stored in the database, allowing them to log in seamlessly with the provided credentials.

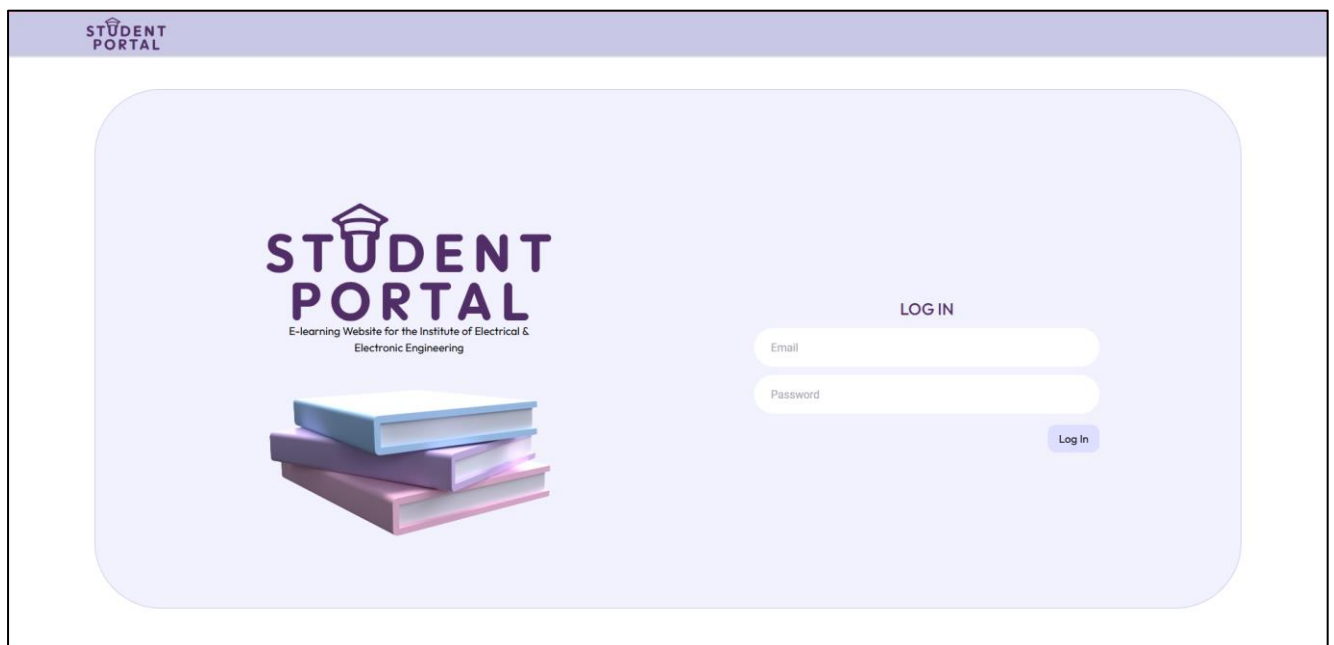


Figure 4-2: Login form

The users must enter both their email and password correctly to log in to their account. Upon submitting the login form, the application sends an HTTP request to the backend, containing the user's credentials. The backend then queries the database to verify if

the provided email and password belong to a pre-registered user. If the credentials are correct, the backend responds to the frontend with an access token, which is stored in cookies, granting the user access to their account. However, if either the email or password is incorrect, the system triggers an error response. An error toaster will appear at the top of the screen, and an error message will be displayed below the input fields, informing the user of the mistake and prompting them to try again as shown in the figure on the next page.

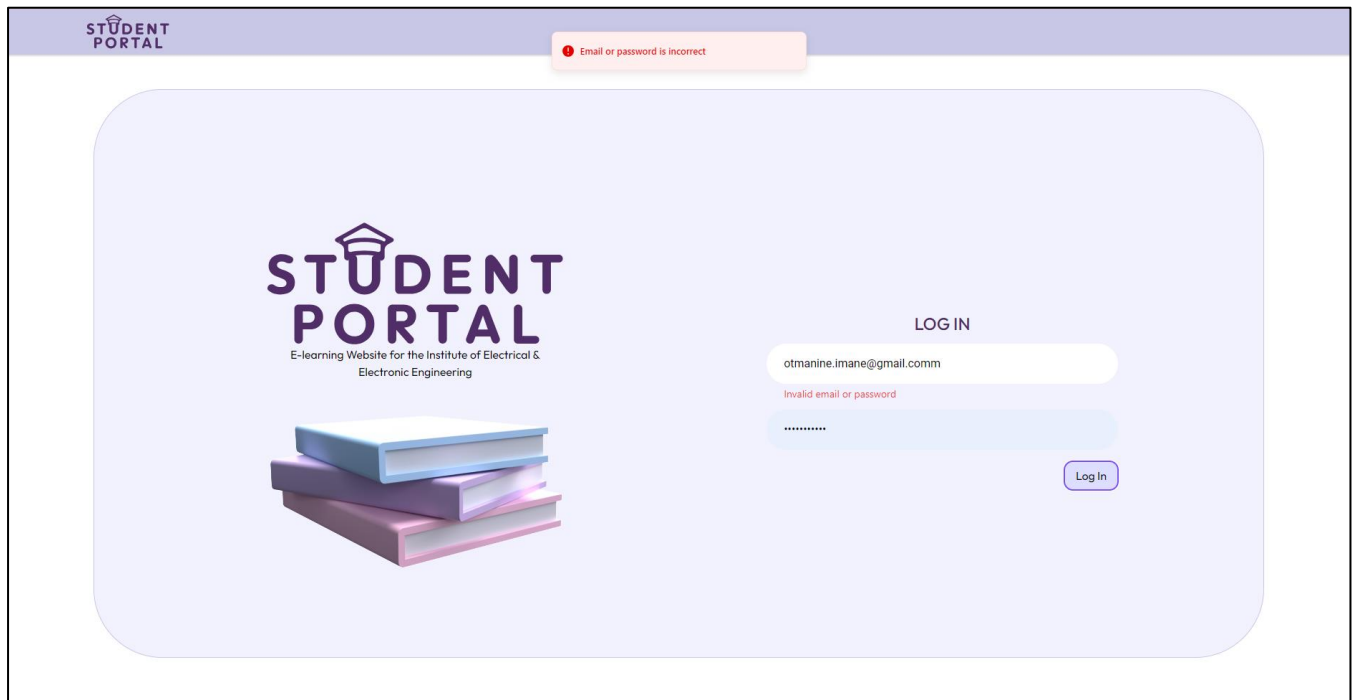


Figure 4-3: Login page errors

Once the backend responds to the authentication request, it also sends the user information, which is then stored in the Redux store, making it accessible across all pages of the website. The user is then redirected to the dashboard page.

4.2.2 Dashboard Interface

After the student logs in he meets the dashboard page. On the left of the page, he can see a sidebar illustrating five tabs, dashboard, courses, assignments, calendar, and grades. On the dashboard page, he can see the last three lectures posted, the assignments he has to do, and the announcements his teachers have made for the past 15 days. He can click on lectures and assignments to quickly access them, and he can see the latest events through the announcements and open the corresponding courses. The teacher can see the same information on this page.

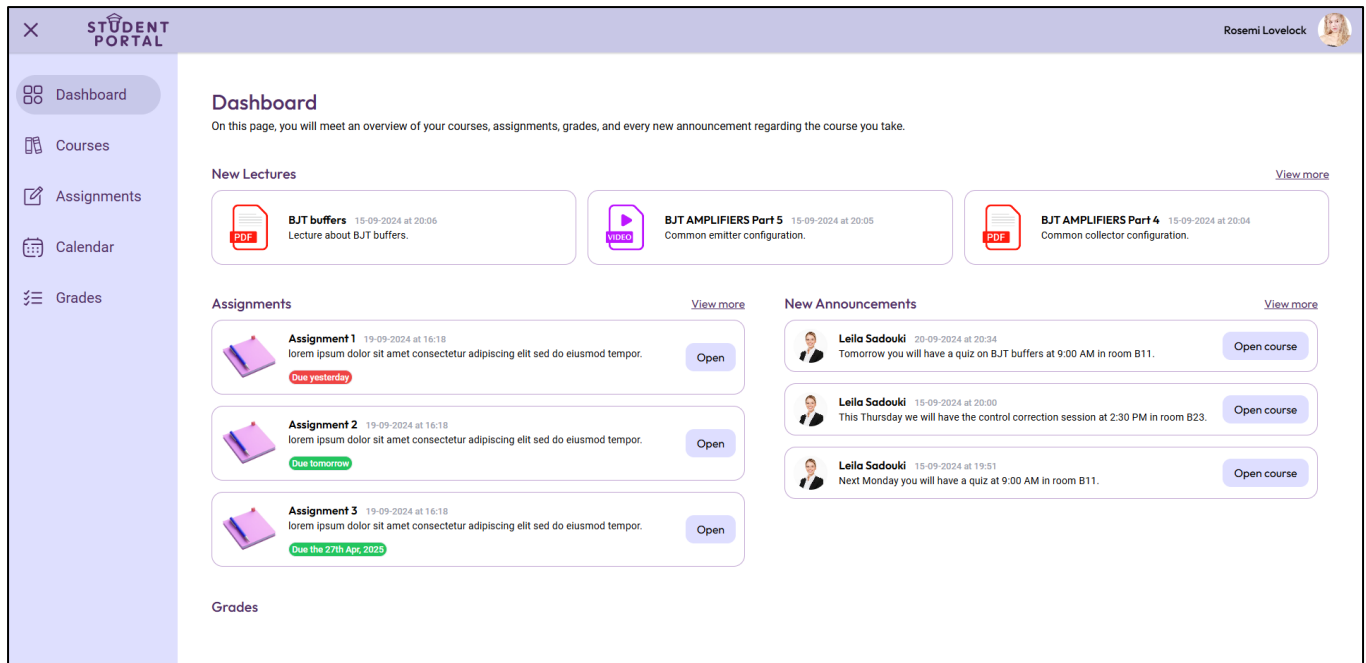


Figure 4-4: Dashboard page

4.2.3 Profile Interface

The users can see their profile and personal information by clicking on their profile image on the top right of the website, and then clicking on the “View profile” button. This will take them to the profile page. They can also log out by clicking on the “Logout” button. By doing that the website will clear the access token from the cookies and redirect them to the login page.

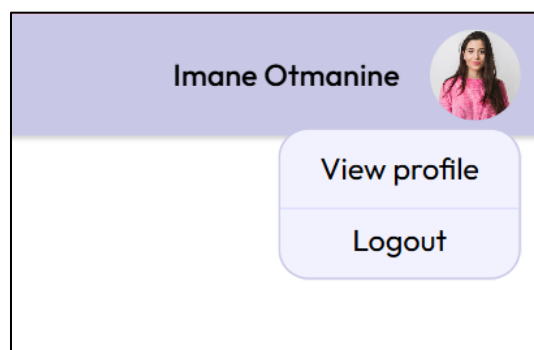


Figure 4-5: User profile dropdown

When students navigate to their profile page, they are presented with their personal information, including their full name, profile picture, email, user ID, academic year, major, and group. Teachers, on the other hand, see their full name, email, and user ID. These data are retrieved from the Redux store and dynamically displayed based on the user’s role.

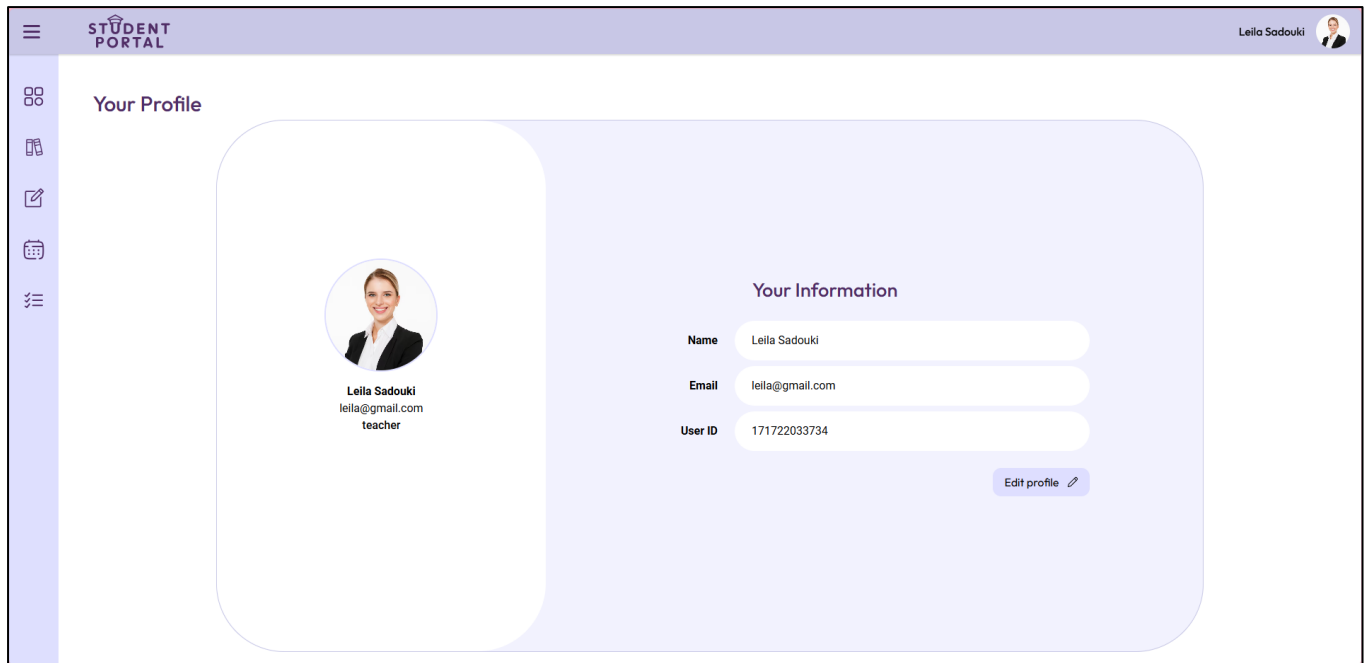


Figure 4-6: Profile details page for a teacher

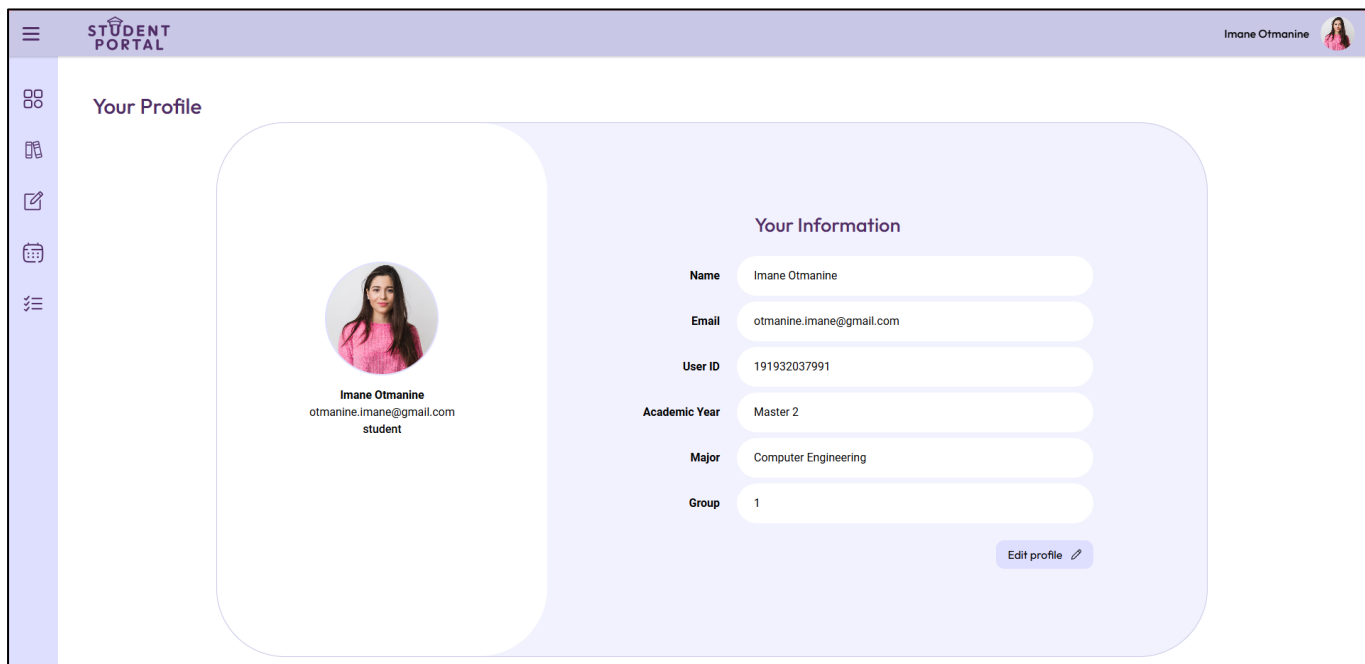


Figure 4-7: Profile details page for a student

If the user wants to change his profile picture, email, or password, they can do that by clicking on the “Edit profile” button. By clicking they open a form that they need to fill with the new information they want to update. Also, the profile picture now becomes clickable, and once clicked, it allows the user to upload a new image to replace the existing one.

STUDENT PORTAL Imane Otmanine

Dashboard Courses Assignments Calendar Grades

Your Profile

Imane Otmanine
otmanine.imane@gmail.com
student

Edit Your Profile

Email

Current Password

Password

Confirm Password

Cancel Save

Figure 4-8: Edit profile page

STUDENT PORTAL Imane Otmanine

Dashboard Courses Assignments Calendar Grades

Your Profile

Imane Otmanine
otmanine.imane@gmail.com
student

Edit Your Profile

Email

Current Password

Password

Confirm Password

Cancel Save

Figure 4-9: Edit profile page after filling in the new information

After submitting the changes, the updated information is sent to the backend, where the current password is verified to ensure it is correct before allowing any updates. After saving the new information in the database, a response is sent back containing the updated details. The redux store gets updated with this information, and lastly, the user's profile page is refreshed to reflect the new changes.

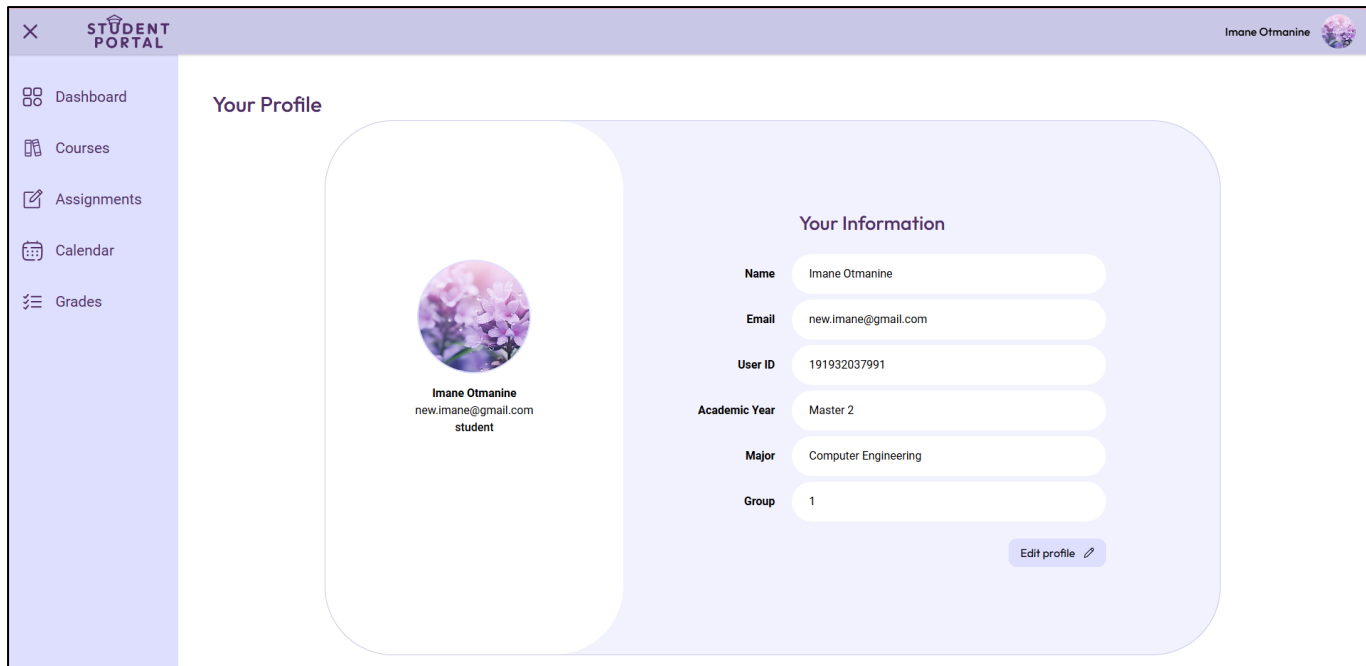


Figure 4-10: Profile page after submitting the changes

4.2.4 Courses Interface

When a user clicks on the “Courses” tab on the sidebar, the web page navigates to the courses page. The application requests the available courses for the user, in the case of a student, the courses they are enrolled in are fetched, while for a teacher, the courses they manage are retrieved. Once received they are displayed in the form of course cards consisting of a course picture, the course name and description, and a button to view the course’s page.

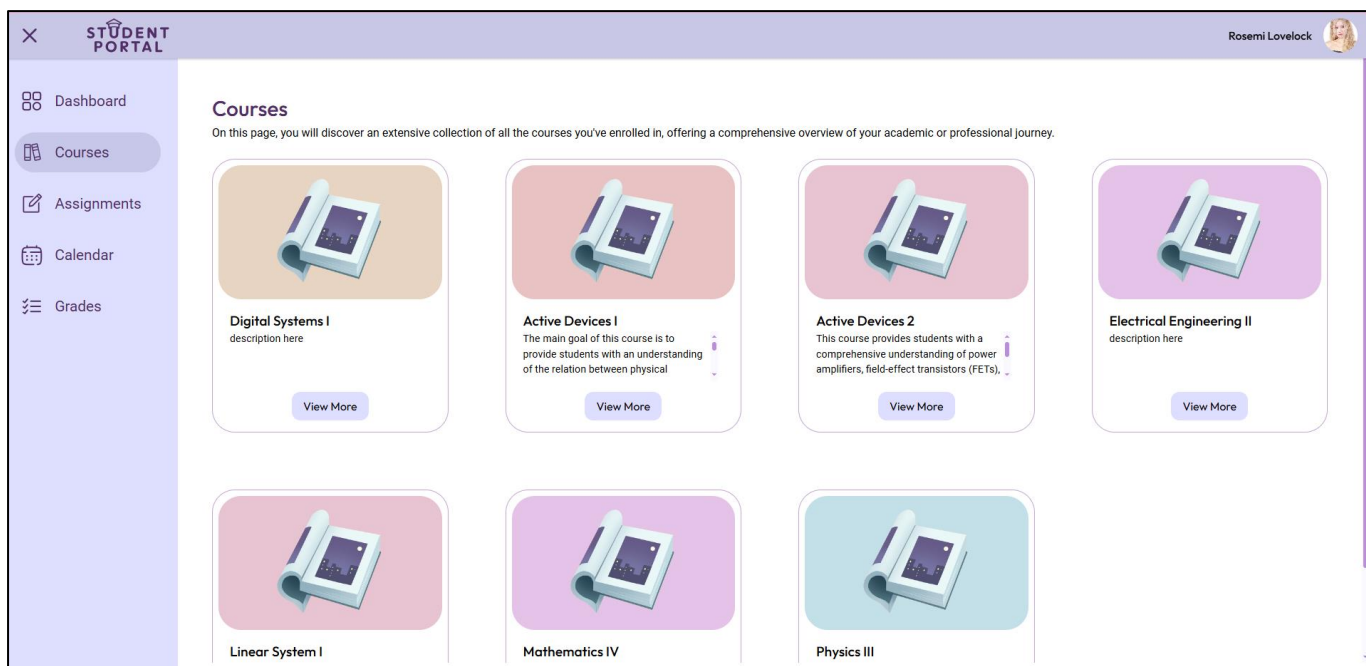


Figure 4-11: Courses page for a student

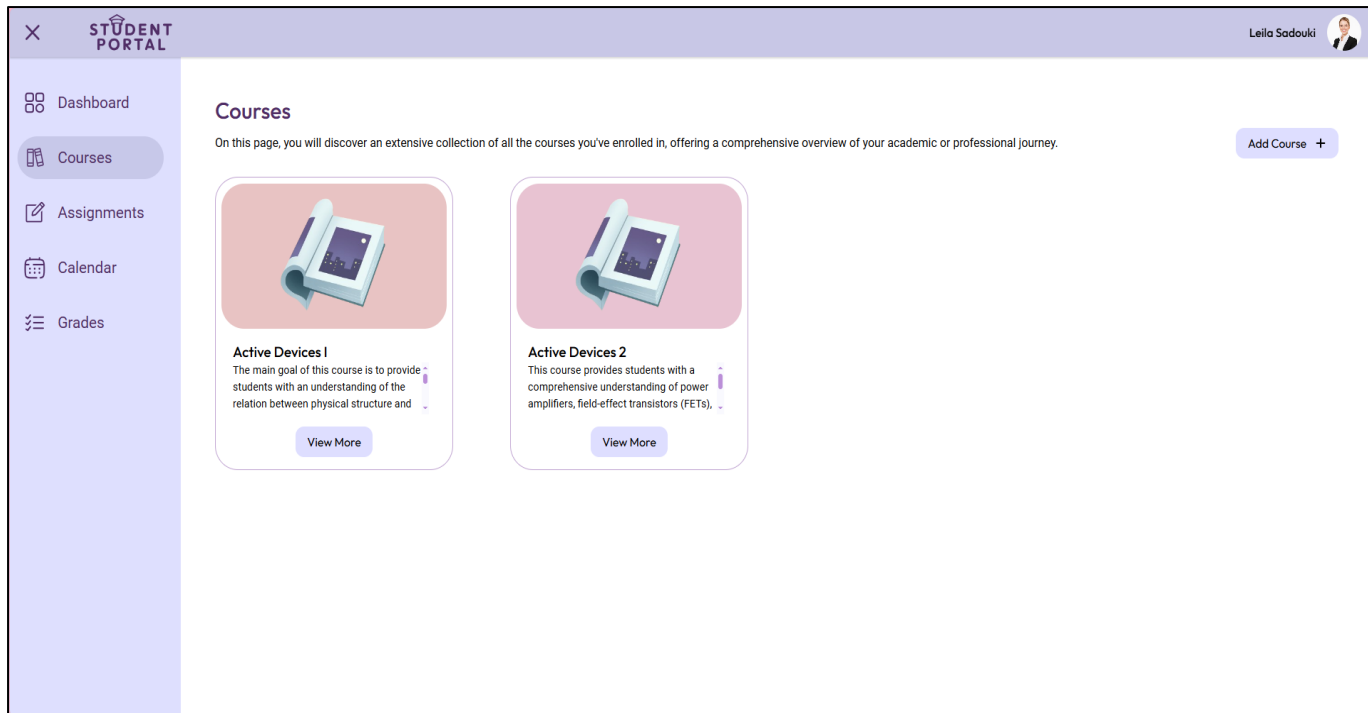


Figure 4-12: Courses page for a teacher

For the teacher, an “Add Course” button is also displayed allowing the teacher to create a new course. Once clicked, a modal opens containing a form in which they can fill in the course name and description, and select which academic level, year, and major the course is for.

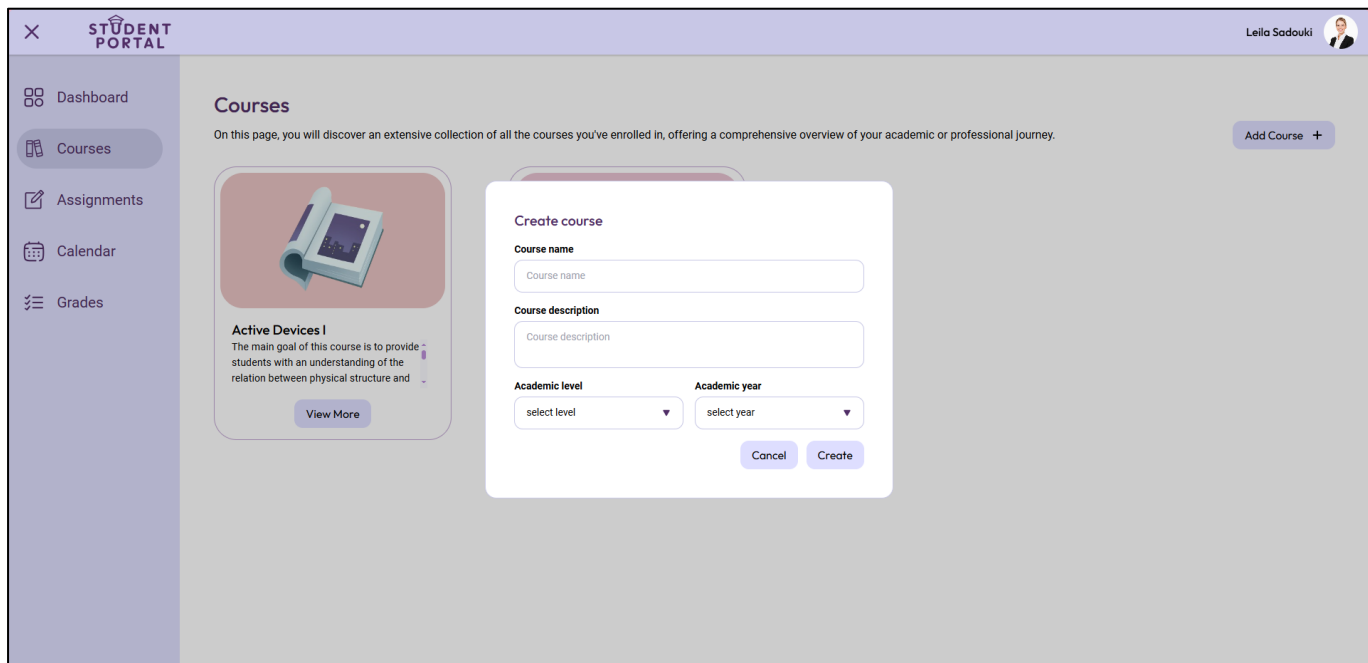


Figure 4-13: Add a course modal

Create course

Course name

 You must give the course a name

Course description

 You must add a description

Academic level

Academic year

 You must add an academic year

Course major

 You must add a major

Cancel Create

Figure 4-14: Add a course modal showing errors

When the teacher submits the course information, a new course gets created in the database with it, and a success toaster is displayed on the top of the screen before it refreshes to show the added new course.

STUDENT PORTAL

Leila Sadouki

Courses

On this page, you will discover an extensive collection of all the courses you've enrolled in, offering a comprehensive overview of your academic or professional journey.

Create course

Course name

Course description

Academic level

Academic year

Course major

Cancel Create

Figure 4-15: Add a course modal when submitting a new course

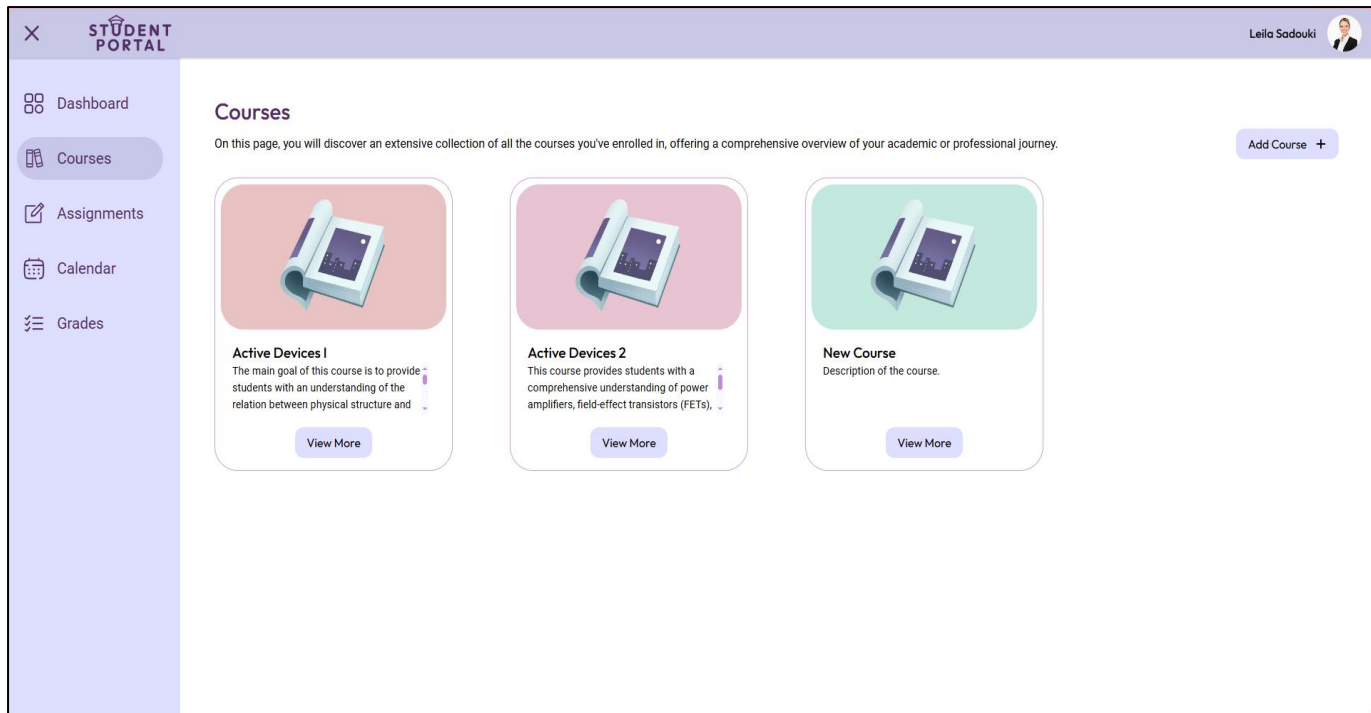


Figure 4-16: Courses page after a new course is added

4.2.5 Course Interface

When this page is rendered, all the information about the course is retrieved from the database. The course page is divided into two parts. On the right, there's a sidebar for the course name, description, teacher, and students attending the course. On the left, the application displays the list of lectures and announcements sorted. Lectures are clickable and they redirect the user to the specific lecture page. Announcements consist of the teacher's profile image, their name, date of posting, and the content of the announcement. Lectures, on the other hand, consist of an image illustrating if the lecture is a video or a PDF, the lecture's title, a description of it, and the posting time.

For students, they also have a check box on each lecture card displaying if they have completed studying the lecture or not.

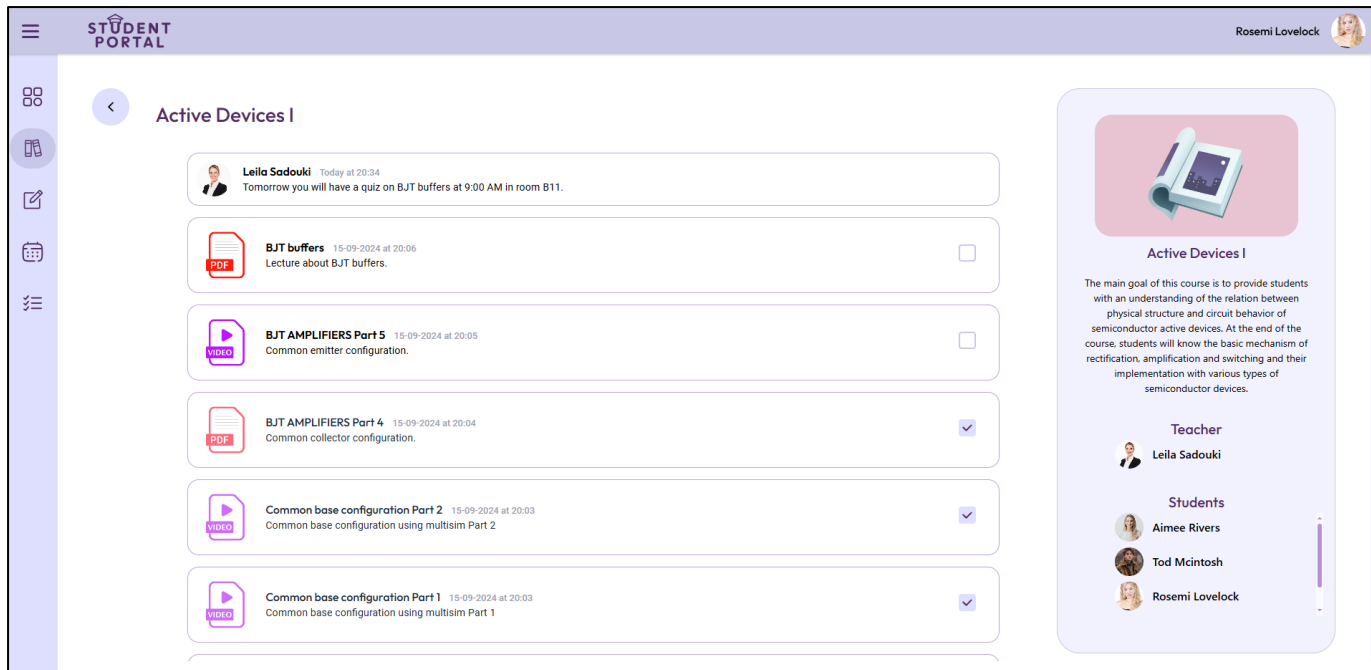


Figure 4-17: Course page for the student

For teachers, however, there are two additional buttons: "Delete" and "Edit Course." The Delete button allows the teacher to remove the course entirely from the database, while the Edit Course button redirects the teacher to the course editing page.

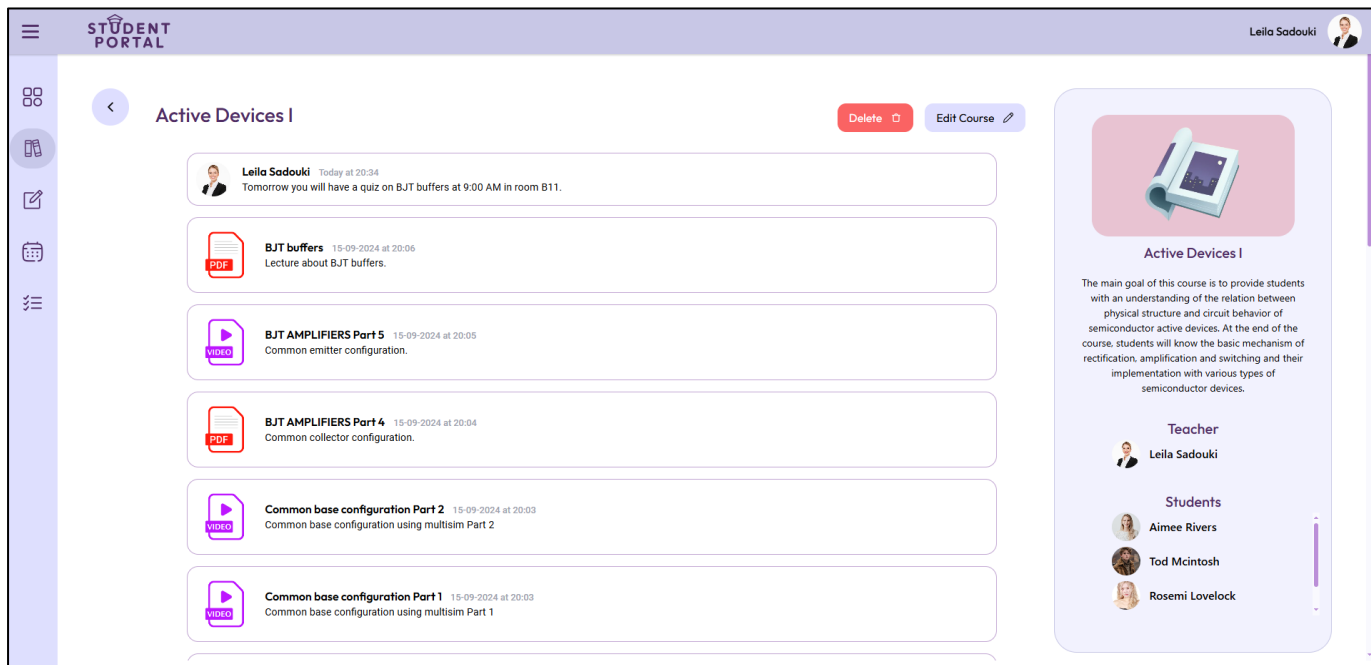


Figure 4-18: Course page for the teacher

When the teacher clicks the "Delete" button, a confirmation popup appears, asking for verification before proceeding with the deletion.

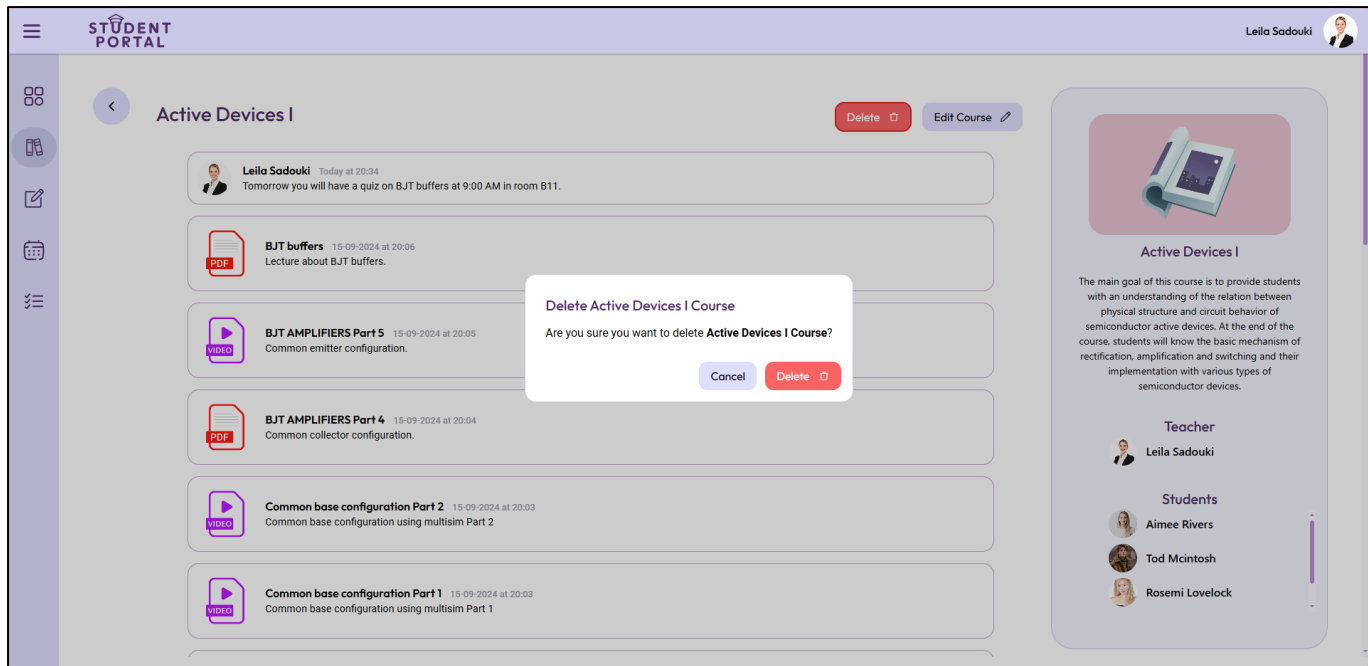


Figure 4-19: Delete course popup

4.2.6 Edit Course Interface

This page is divided into two sections; the first one is designated for editing the course details, and the second is designated for editing course materials.

The Edit Course section consists of the course details: Course picture, course name, description, academic level and year, and major (if the academic year is Master's). It also has a button to enable editing them.

The Course Materials section consists of three tabs: Lectures, Announcements, and Assignments. Each tab has a list of the respective items, along with a delete button for each item.

The screenshot shows the 'Edit course' page in the Student Portal. The page has a purple header with the 'STUDENT PORTAL' logo and the user's name 'Leila Sadouki'. A sidebar on the left contains icons for home, edit, calendar, and list. The main content area is titled 'Edit course' and contains two sections: 'Course Details' and 'Course materials'.

Course Details: This section includes a course image placeholder, a 'Course name' field with the value 'Active Devices I', a 'Description' field with the text 'The main goal of this course is to provide students with an understanding of the relation between physical structure and circuit behavior of semiconductor active devices. At the end of the course, students will know the basic mechanism of rectification, amplification and switching and their implementation with various types of semiconductor devices.', a 'Course academic level' dropdown with the value 'Licence', and a 'Course academic year' dropdown with the value '2'. There is an 'Edit details' button with a pencil icon.

Course materials: This section has tabs for 'Lectures', 'Announcements', and 'Assignments'. It shows two materials: 'BJT buffers' (a PDF file) and 'BJT AMPLIFIERS Part 5' (a video file). Each material has a 'Delete' button with a trash icon.

Figure 4-20: Edit course page

In the first section, when the teacher clicks on the “Edit details” button the course details show in a form allowing the teacher to edit them. Once he finishes making edits he can either press save or cancel the changes.

The screenshot shows the 'Edit course details form' in the Student Portal. The page has a purple header with the user's name 'Leila Sadouki'. A sidebar on the left contains icons for home, edit, calendar, and list. The main content area is titled 'Edit course' and contains a 'Course Details' section.

Course Details: This section includes a course image placeholder, a 'Course name' field with the value 'Updated: Active Devices I', a 'Description' field with the text 'Updated: The main goal of this course is to provide students with an understanding of the relation between physical structure and circuit behavior of semiconductor active devices. At the end of the course, students will know the basic mechanism of rectification, amplification and switching and their implementation with various types of semiconductor devices.', a 'Course academic level' dropdown with the value 'Master', a 'Course academic year' dropdown with the value '2', and a 'Course major' dropdown with the value 'Computer Engineering'. There are 'Cancel' and 'Save' buttons.

Figure 4-21: Edit course details form

When the teacher clicks on “Save” a web request is sent to the backend with the new course information. The backend then stores the information in the database and sends back a successful HTTP response. A success toast notification appears at the top of the screen before the page refreshes to reflect the new changes.

Leila Sadouki

Course details updated successfully

< Edit course

Course Details

Course name

Updated: Active Devices I

Course academic level

Master

Course academic year

2

Course major

Computer Engineering

Edit details

Description

Updated: The main goal of this course is to provide students with an understanding of the relation between physical structure and circuit behavior of semiconductor active devices. At the end of the course, students will know the basic mechanism of rectification, amplification and switching and their implementation with various types of semiconductor devices.

Figure 4-22: Edit course form after submission

In the second section of this page, Once the teacher clicks on delete on one of the lecture cards, a confirmation popup appears, asking for verification before proceeding with the deletion. If they click on delete again the course will be deleted from the database.

STUDENT PORTAL

Leila Sadouki

Dashboard

Courses

Assignments

Calendar

Grades

< Edit course

Course Details

Course name

Active Devices I

Course academic level

Licence

Course academic year

2

Edit details

Description

Delete BJT buffers

Are you sure you want to delete BJT buffers?

Cancel Delete

Course materials

Lectures

Announcements

Assignments

Add material +

BJT buffers 15-09-2024 at 20:06

Lecture about BJT buffers.

Delete

BJT AMPLIFIERS Part 5 15-09-2024 at 20:05

Common emitter configuration.

Delete

Figure 4-23: Delete lecture popup

In the same section, when the teacher presses on the “Add material” button, an “Add course material” appears to let the teacher add new course materials either a lecture, an announcement, or an assignment. They can choose one of the tabs to select what material to add. The Lecture tab consists of three input fields: Lecture title, Lecture description, and Lecture content.

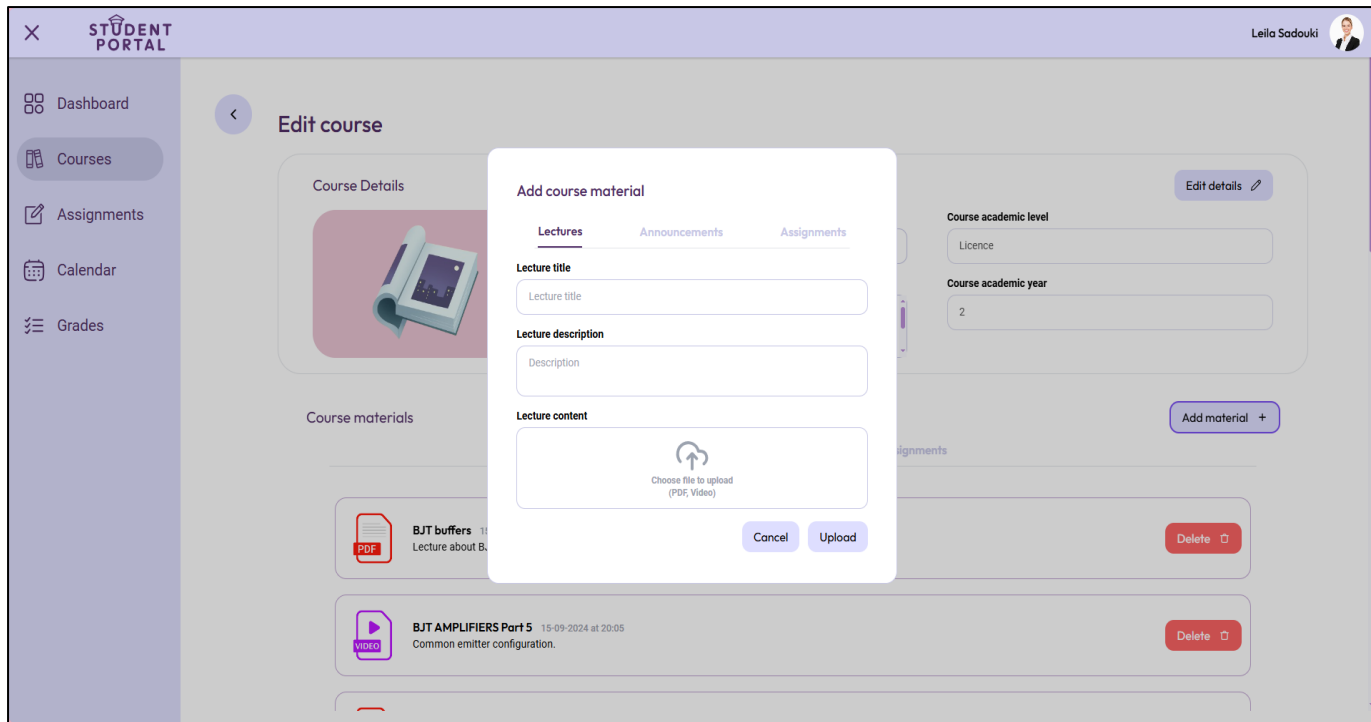


Figure 4-24: Add lecture modal

The Lecture tab consists of one input field for the announcement content.

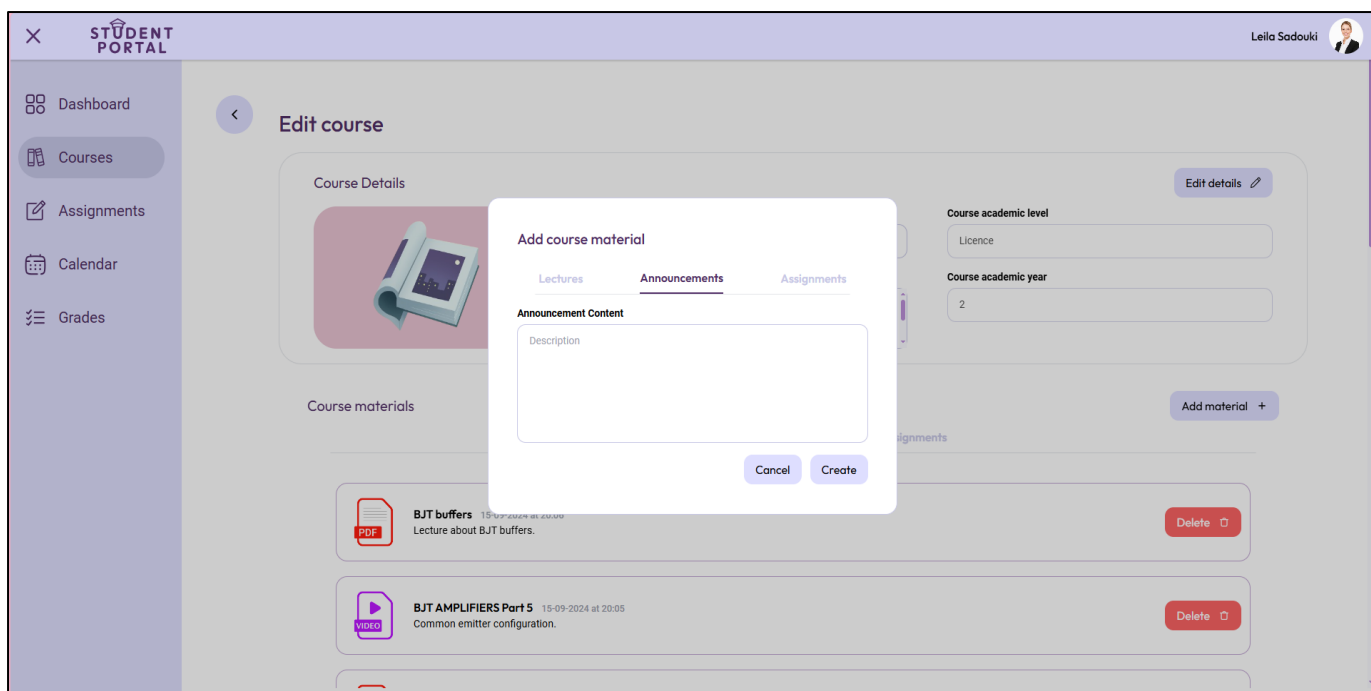


Figure 4-25: Add announcement modal

On the lecture tab, when the user clicks on the upload area the file explorer opens, allowing the teacher to choose a file to upload. Following his selection of the file and clicking the “Create” button, the same changes to “Uploading” with a spinner next to it. In the

meantime, the file and the form details get sent to the backend; the first is stored in the server and the second is stored in the database.

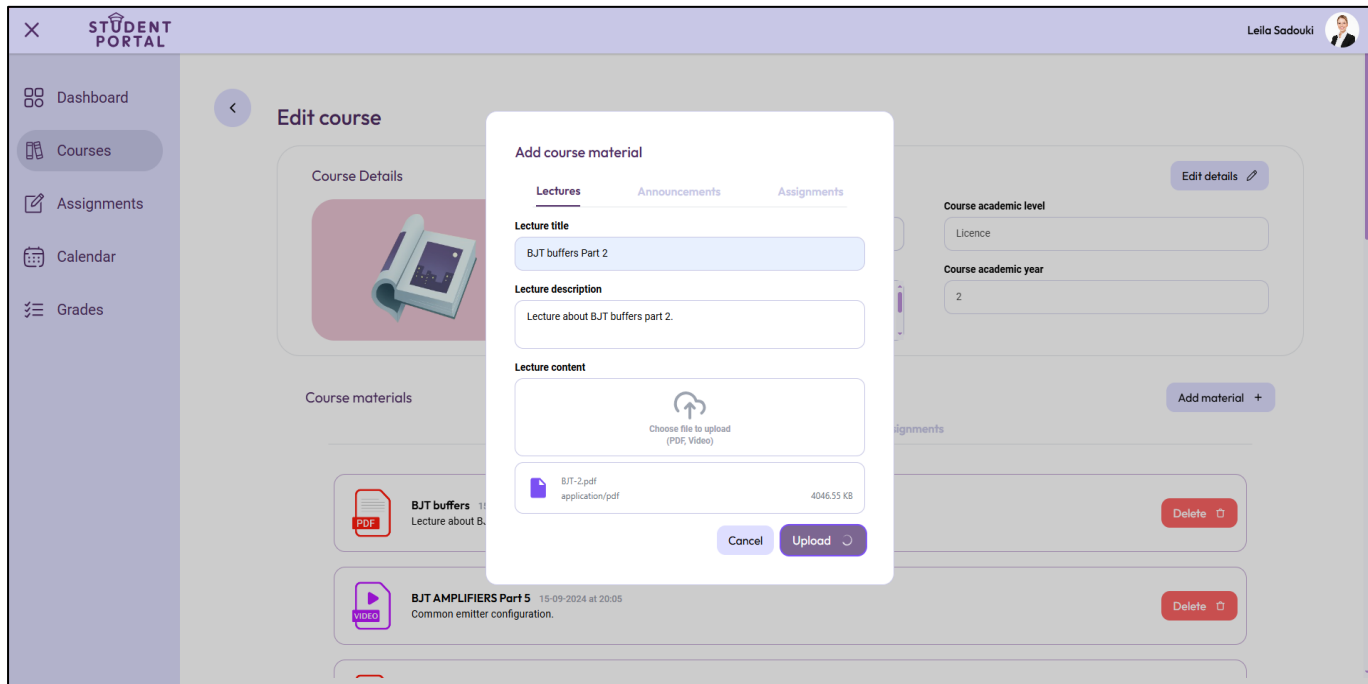


Figure 4-26: Lecture form upload

Once the lecture is saved, the backend responds with a success and a success toast notification appears before the page auto-refreshes to reflect the new changes.

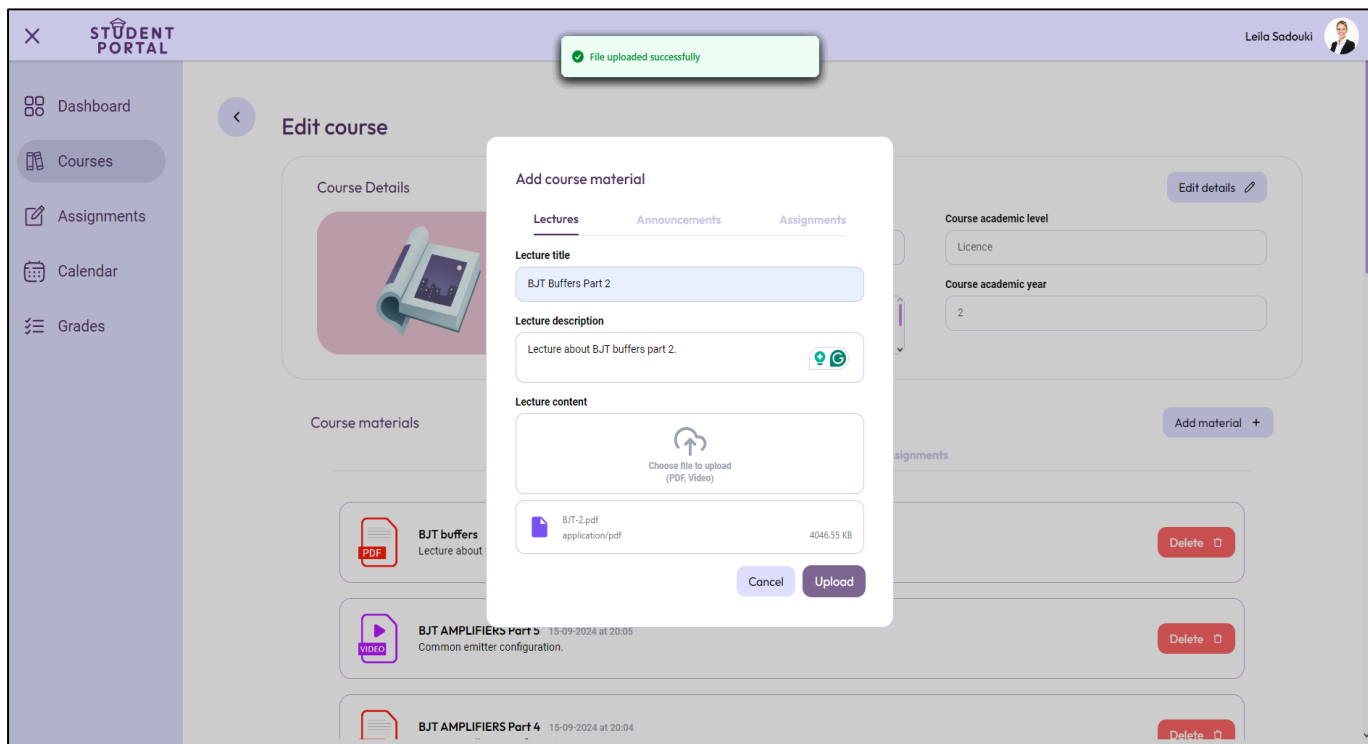


Figure 4-27: Lecture form successful upload

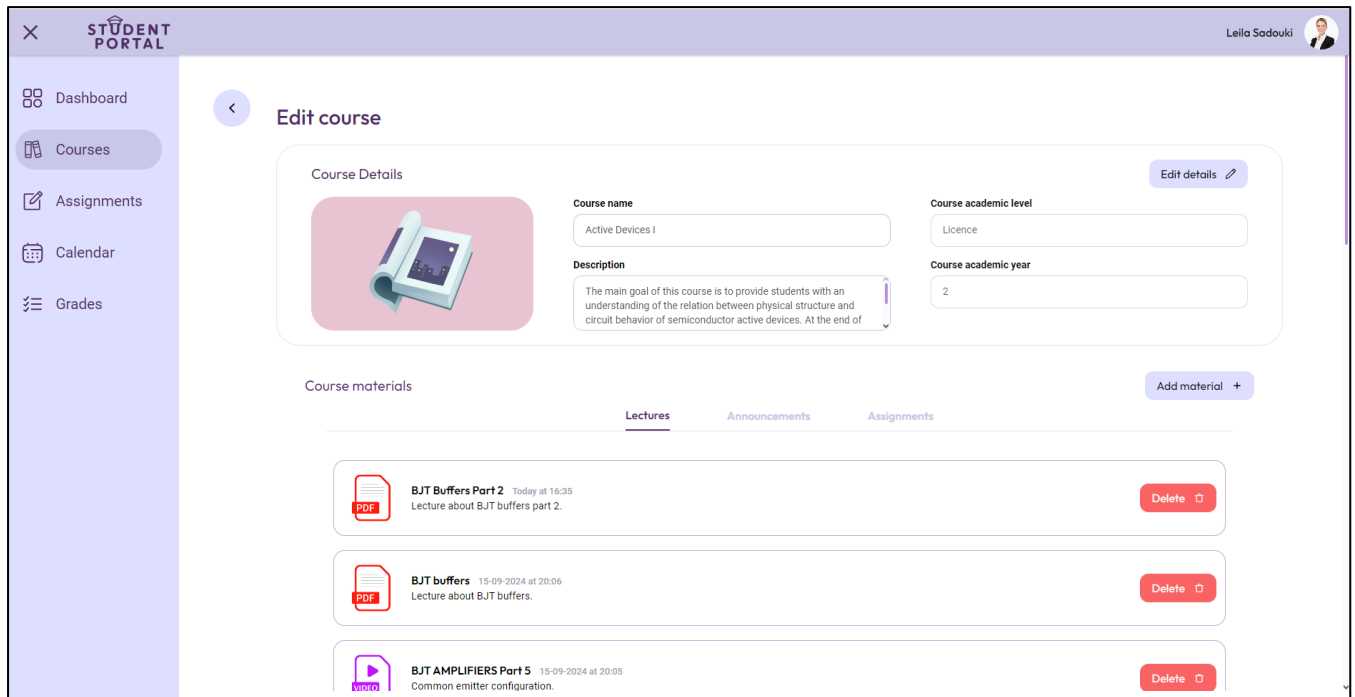


Figure 4-28: Edit course page after the refresh

4.2.7 Lecture Interface

The lecture page is divided into two parts: the right sidebar displays a list of other lectures, while the left side features a PDF/video reader that shows the lecture content, which was received from the backend server during page rendering. Below the file reader, the title and description of the lecture are presented.

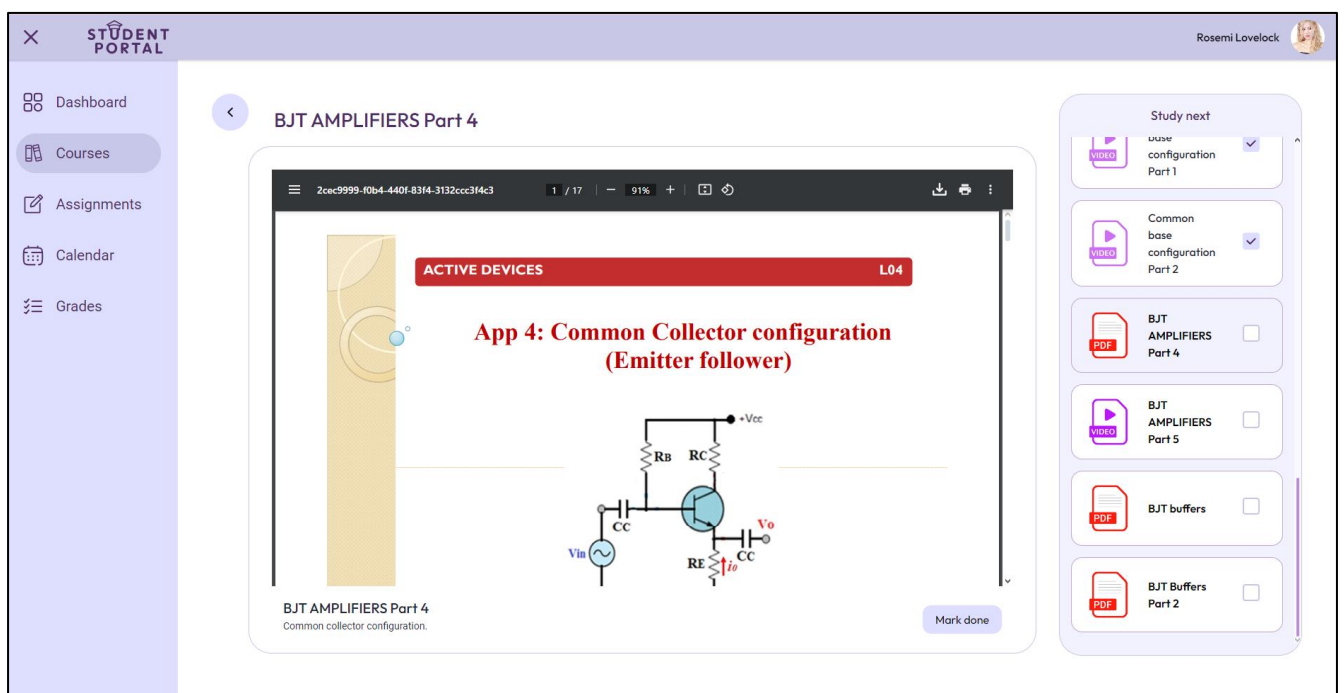


Figure 4-29: Lecture page

If the user is a student and hasn't yet completed the lecture, a “Mark as Done” button is also displayed. Upon clicking it, the lecture will be marked as completed both in the frontend interface and the backend, and the button changes to indicate the status update, allowing the user to easily track their progress. A new button also appears for the user to allow him to easily move to the next lecture page.

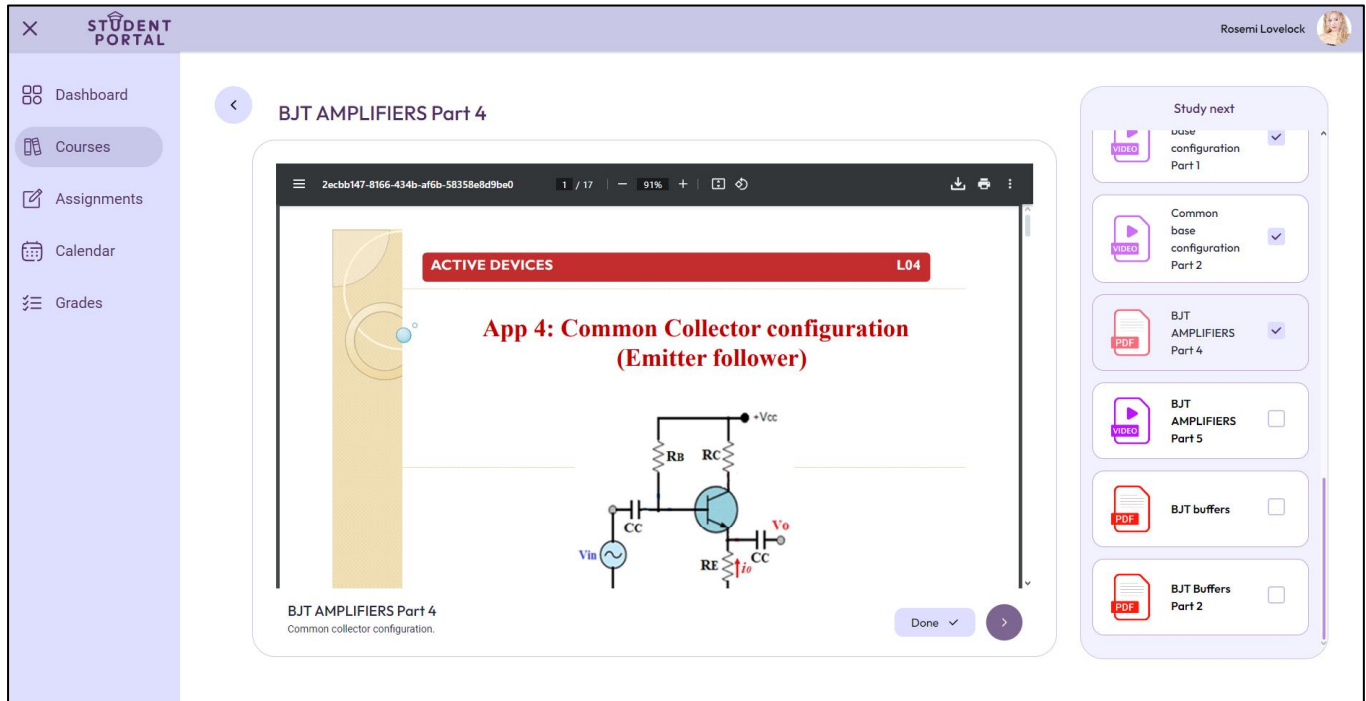


Figure 4-30: Lecture page after pressing the "Mark as done" button

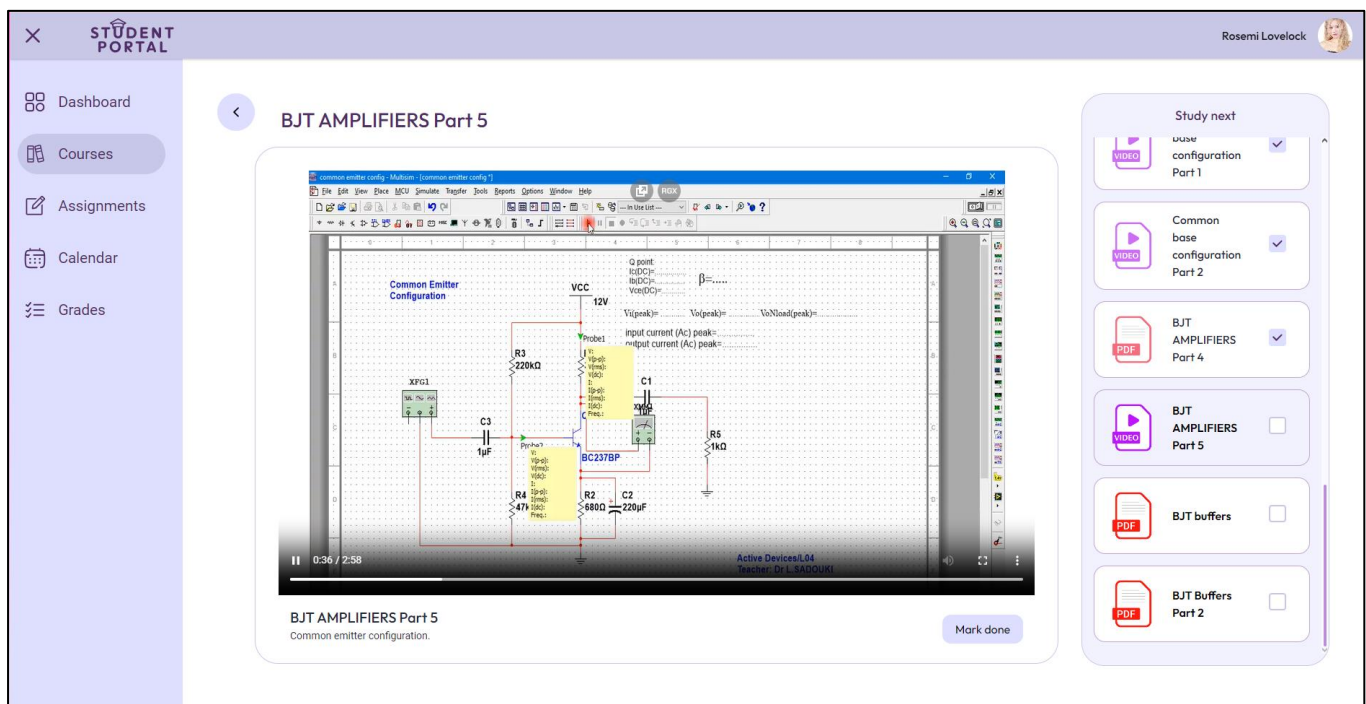


Figure 4-31: Lecture page displaying a video lecture

4.3 Conclusion

In conclusion, this chapter provided a comprehensive overview of the application's interfaces, highlighting the implementation of the login, dashboard, profile, and course-related screens. These interfaces form the core of the user experience, ensuring smooth navigation and interaction within the application.

Future Enhancements

As we look to expand and improve the Student Portal platform, several key enhancements are proposed to enrich user experience and functionality:

a) Assignment, Grades, and Calendar Pages

To facilitate better management of academic tasks, dedicated pages for assignments and grades will be added. These features will allow students to receive assignments, submit their work, and view their marks. A calendar page will provide an overview of important dates, including deadlines and events, enhancing students' ability to plan their study schedules effectively and easily track due dates.

b) Comments Feature on Lectures

To promote interaction and facilitate communication, a comments feature will be added to lecture pages. This will allow students to engage with content, ask questions, and share insights directly on lecture materials, fostering a collaborative learning atmosphere.

c) Support for Additional File Formats

Currently limited to PDF and video files, the platform will expand its file upload and display capabilities to support a wider range of formats, including PowerPoint presentations, Word documents, and others. This enhancement will allow instructors to share diverse types of resources, enriching the learning experience.

d) Cloud Integration for File Storage

Currently, files are stored directly on the server. To improve scalability and accessibility, the platform will transition to cloud storage for file management. This will provide more efficient storage solutions, enabling users to access materials from anywhere, while reducing server load and ensuring better data redundancy and security.

e) Live Chat Between Students and Teachers

Implementing a live chat feature will enable real-time communication between students and teachers. This functionality will support immediate assistance, clarification of

concepts, and enhance the overall engagement, making the learning process more dynamic and interactive.

f) Laboratory Page with Embedded Simulator Software

To provide practical experience in the electrical and electronic fields, a dedicated laboratory page will be introduced, by embedding a simulator software to the website. This feature will enable students to conduct lab experiments and simulations through the platform and be graded on them.

g) Arabic Language Support

The platform will include support for the Arabic language, making it accessible for high school students. This feature will allow users to switch the interface language, ensuring that students who are more comfortable in Arabic can effectively navigate and utilize the platform.

These enhancements aim to create a more robust and user-friendly platform, ultimately supporting better educational outcomes and a more enjoyable learning experience for all users.

General Conclusion

In summary, the development of the Student Portal e-learning platform represents a significant advancement in the delivery of educational resources and support. By integrating essential web technologies and modern development tools, the project successfully creates an environment that fosters effective learning and collaboration among students and teachers.

Throughout this project, we have explored the finer points of web development, from the underlying technologies to the design and implementation of user interfaces. The emphasis on user experience has been a guiding principle, ensuring that the platform is intuitive and accessible to all users.

Looking forward, the proposed enhancements will further enrich the platform's capabilities, addressing the evolving needs of students and educators. By incorporating features such as assignment tracking, diverse file support, interactive laboratories, and improved communication tools, we aim to create a comprehensive educational ecosystem that promotes engagement and academic success.

Ultimately, this project not only demonstrates the potential of modern web technologies in education but also sets the stage for future innovations that can transform the learning experience.

References

- [1] M. Haverbeke, Eloquent JavaScript, No Starch Press, 2024.
- [2] Web Agency Optimize 360, “Web development: diving into the world of website creation,” [Online]. Available: <https://www.optimize360.fr/en/agence-creation-sitesweb/web-definitions/web-development/>. [Accessed Mars 2024].
- [3] N. Ferguson, “What's the Difference Between Frontend vs Backend Web Development?,” 6 February 2024. [Online]. Available: www.careerfoundry.com/en/blog/web-development/whats-the-difference-between-frontend-and-backend. [Accessed Mars 2024].
- [4] Source Defense, “What is a Web Request?,” [Online]. Available: <https://sourcedefense.com/glossary/web-request>. [Accessed Mars 2024].
- [5] W. B. Mastery, Web Development for beginners, Independently published, 2020.
- [6] T. Pattinson, “Relational vs. Non-Relational Databases,” 9 November 2022. [Online]. Available: <https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases>. [Accessed March 2024].
- [7] Northell Team, A Complete Web Development Guide, northell.design, 2021.
- [8] J. Abu, “5 Coding Projects You Should Include in Your Front End Portfolio,” 8 February 2021. [Online]. Available: <https://www.freecodecamp.org/news/coding-projects-to-include-in-your-frontend-portfolio>. [Accessed Mars 2024].
- [9] S. Souders, High Performance Web Sites, Sebastopol: O'Reilly Media, Inc., 2007.
- [10] E. Brown, Web Development With Node & Express, O'Reilly Media, Inc., 2014.
- [11] L. Ching-Hong, “The comparison of learning effectiveness between traditional face-to-face learning and e-learning among goal-oriented users,” *Multimedia Technology and its Applications (IDC)*, no. 6th International Conference on. 2010. IEEE., 2010.

- [12] M. Shirzad, A. Hoseinpanah and M. H. Ahma, “Using Cloud Computing in e-learning Systems,” in *Cloud Computing Technologies, Applications and Management (ICCCTAM)*, 2012.
- [13] R. Gryshuk, “15 Great Google Classroom Alternatives,” 17 May 2024. [Online]. Available: <https://www.educate-me.co/blog/google-classroom-alternatives>. [Accessed August 2024].
- [14] Saylor Academy, “Introduction to Software Engineering/Methodology,” 10 June 2024. [Online]. Available: <https://learn.saylor.org/mod/book/view.php?id=65499&chapterid=58136>. [Accessed September 2024].
- [15] Visual Studio Code Team, “Documentation of Visual Studio Code,” Microsoft, [Online]. Available: <https://code.visualstudio.com/docs>. [Accessed Mars 2024].
- [16] L. Karrys, M. Rienstra, M. Borins and E. Thomson, “About npm,” npm, Inc., 23 October 2023. [Online]. Available: <https://docs.npmjs.com/about-npm>. [Accessed Mars 2024].
- [17] Atlassian Team, “What is version control?,” 2024. [Online]. Available: <https://www.atlassian.com/git/tutorials/what-is-version-control#:~:text=Version%20control%2C%20also%20known%20as,to%20source%20code%20over%20time..> [Accessed August 2024].
- [18] Geeksforgeeks, “Introduction to Tailwind CSS,” 7 June 2024. [Online]. Available: <https://www.geeksforgeeks.org/introduction-to-tailwind-css>. [Accessed August 2024].
- [19] React Team, “React,” Meta, 2024. [Online]. Available: <https://react.dev>. [Accessed Mars 2024].
- [20] R. Wieruch, *The Road to React*, Leanpub, 2016.
- [21] Redux documentation authors, “Getting Started with Redux,” Redux, 31 Mar 2024. [Online]. Available: <https://redux.js.org/introduction/getting-started>. [Accessed August 2024].

- [22] NodeJs Team, “Run JavaScript Everywhere,” OpenJS Foundation © , [Online]. Available: <https://nodejs.org/en>. [Accessed Mars 2024].
- [23] ExpressJs Team, “Express,” OpenJs Foundation, [Online]. Available: <https://expressjs.com/>. [Accessed August 2024].
- [24] MDN contributors, “Express/Node introduction,” 25 July 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction. [Accessed August 2024].
- [25] A. Saini, “MongoDB – Working and Features,” 2 July 2024. [Online]. Available: <https://www.geeksforgeeks.org/what-is-mongodb-working-and-features/>. [Accessed August 2024].
- [26] Mongoose Team, “Mongoose,” [Online]. Available: <https://mongoosejs.com/>. [Accessed August 2024].
- [27] C. Miner, “Handling File Uploads Using Multer In Node Js Express,” 5 April 2024. [Online]. Available: <https://medium.com/learn-to-earn/handling-file-uploads-using-multer-in-node-js-cbe0389f4f9e>. [Accessed August 2024].
- [28] Descope Inc, “What Is a JWT & How It Works,” Descope Inc, 30 March 2024. [Online]. Available: <https://www.descope.com/learn/post/jwt>. [Accessed August 2024].
- [29] MDN contributors, “Cross-Origin Resource Sharing (CORS),” 26 July 2024. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>. [Accessed August 2024].
- [30] MongoDB Team, “Data Modeling,” MongoDB, Inc., 2024. [Online]. Available: <https://www.mongodb.com/docs/manual/data-modeling/>. [Accessed August 2024].