MU Connect

Senior Project Presented to the Faculty of Science
Department of Computer Sciences
Al Maaref University

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science
Computer Sciences

Ali Awdi - 10121668
Kamel Faour -10121641

Spring 2025-2026

Dedication

In the name of Allah, the Most Gracious, the Most Merciful.

To Allah.

To our teachers and mentors, who guided us with their knowledge.

To our families and friends, for their patience and support throughout this journey.

To the martyrs — especially the martyrs of Al-Maaref University — who sacrificed their lives so that we could stand here today and present this work. This work is a humble effort to repay a part of the debt we owe them and to honor their sacrifice.

May this work stand as a reminder that their blood was not shed in vain, and that we, the new generation, will continue to uphold the values of justice, freedom, and dignity for which they fought.

May Allah accept this humble effort and guide us to use our knowledge and skills for the benefit of our community and our beloved homeland.

Acknowledgements

Firstly and foremost, we thank Allah, the merciful, for entrusting us with the power, patience, and guidance that would enable us to complete this task successfully. We'd like to express our sincere gratitude to our director and all our teachers at Al-Maaref Institute for their admirable leadership, unwavering encouragement, and generous support throughout the expedition. Our intelligence and advice have been a real light to us. Besides our sincere gratitude to our family for their endless love, patience, and understanding, we are also deeply grateful. Your support has given our team the motivation to continue traveling, even during the most difficult moments. Thank you, our friend and colleague, for your assistance, ideas, and cooperation:). A special part of our current expertise will always remain the long hours, discussions, and moments that we share. In the end, we admit everyone who contributes directly or implicitly to the success of the current function. This success has been made possible by your support and confidence in us. Could this project be a source of pride for our group and a step towards greater intelligence and assistance?

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ASP.Net - Active Server Pages Network Enabled Technologies.

HTML - Hypertext Markup Language

CSS - Cascading Style Sheets

UML - Unified Modeling Language

SQL - Structured Query Language

RDBMS - Relational Data Base Management System

LLM- Large Language Model

PHP - Hypertext Preprocessor

API – Application Programming Interface

UI – User Interface

ER- Entity Relational

Abstract

MU Connect is a web platform designed to enhance students' scholarly and social experiences near Al-Maaref University. It responds to common student demands by providing a central organization for equity education capital, producing a division exchange request, calculating classes, and being well informed about institution functions and the baseball club. Moreover, MU Connect extends a synergies level table to help students track their scholarly progress smoothly. The platform shall be built using Laravel as a backend and React JS as a frontend, ensuring a secure, scalable, and user-friendly interface. MU Connect aims at simplifying exchange and asset sharing among students while supporting their academic strategies and campus engagement by integrating these features into a single implementation. In order to promote a society of associates and to improve student admission to necessary scientific equipment and intelligence in an individual convenient setting, the current objectives of the undertaking are as follows. This report outlines the project's objectives, design, implementation processes, and anticipated benefits for the Al-Maaref University students community, highlighting its potential for broader adoption and impact.

Chapter1: Introduction

1.1 Overview

Students at Al Maaref University face significant challenges in their academic journey. Many are frequently overwhelmed by complex student affairs, often feeling lost when seeking clear answers and guidance. In addition, the limited availability of accessible study resources compounds their difficulties, leaving them without adequate support for their learning needs. Furthermore, the reliance on large, unwieldy PDF documents makes it particularly hard for students to efficiently extract and retain essential information, ultimately impeding their academic success.

1.2 Problem Statement

Students and professionals face significant challenges in the digital world, including phishing attacks that compromise personal data, difficulty in sharing and accessing reliable articles, and a lack of platforms to connect with experts for guidance. Fragmented tools and misinformation hinder efficient knowledge exchange, while the absence of secure communication channels limits collaborative learning. These issues create barriers to digital security, reliable information sharing, and professional growth, leaving users vulnerable and disconnected in an increasingly online-dependent environment.

1.3 Proposed Solution

To every challenge, there is a solution. Our website is designed to tackle the everyday problems faced by students at Al Maaref University—whether it's getting lost in complicated student affairs, struggling with limited study resources, or being overwhelmed by dense, lengthy course materials. We offer an all-in-one platform that brings together a supportive community of learners and a smart tool that transforms bulky academic content into clear, concise summaries.

Imagine a space where students can easily share their experiences, ask questions, and exchange helpful tips about everything from administrative issues to course content. With our platform, finding the key points in a huge PDF becomes simple. Our advanced course summarizing feature quickly distills complex material into digestible insights, so students can focus on what really matters without wasting time sifting through endless pages.

In addition, the platform encourages a collaborative learning environment. Students can contribute their own summaries, share study resources, and even create interactive quizzes based on these streamlined course materials. This not only makes studying more efficient but also builds a community where everyone works together to overcome academic challenges.

By turning overwhelming information into manageable, easily accessible content, our platform empowers students to study smarter and succeed in their academic journey.

1.4 Gantt Chart

A Gantt chart table provides a clear and structured visual summary of a project's timeline and task schedule. It simplifies intricate project plans into an easy-to-read

format, displaying tasks, their durations, and dependencies in a well-organized grid. This table outlines the essential tasks for our project, along with their durations in days and any dependencies between them.

Task No.	Task Description	Task Duration (days)	Dependencies
A	Project Initiation	5 days	None
B	System Design	10 days	A
C	Backend Development	30 days	B
D	Frontend Development	35 days	B
E	AI Integration	2 days	C&D
F	Testing & Quality Assurance	2 days	E
G	Deployment (Frontend & Backend)	2 days	C & D
H	Maintenance	8 days	G
I	Report	4 days	H

Table 1 Gantt Chart

1. Task A://todo we should remove this or do our own For our phishing detection website, we need to conduct research to collect accurate information about each feature and service provided on the platform. This task will take 15 days to complete and does not rely on any other tasks.

2. Task B: The Requirements Analysis task involves identifying and gathering the necessary requirements for our phishing detection website. This phase will take 15 days to complete and is dependent on Task A, which involves the research we previously conducted.

3. Task C: In this task, we will design the functionalities of our phishing detection website, including application modeling and overall design. This phase will take 20 days to complete and depends on the successful completion of Task B (Requirements Analysis).

4. Task D: The State-of-the-Art task (Task D) involves researching existing phishing detection websites to analyze similar platforms. This task will take 7 days to complete and depends on the completion of Task B (Requirements Gathering)

5. Task E: The Database Design task focuses on creating the schema and structure for our phishing detection website's database. This phase will take 15 days to complete and is dependent on the successful completion of Task C (Design Models and Functions).

6. Task F: The Database Implementation task focuses on building and deploying the designed database for our phishing detection website. This phase will take 18 days to complete and depends on the successful completion of Task E (Database Design).

7. Task G: The Front-End Implementation task involves developing the user interface and front-end components of the phishing detection website. This phase is expected to take 75 days and depends on the successful completion of Task C (Design Models and Functions).

8. Task H: The Back-End Implementation task focuses on developing the server-side logic and functionalities for the phishing detection website with APIs, ensuring dynamic operations and seamless integration with the front-end.

This phase is expected to take 120 days and depends on the successful completion of Task C (Design Models and Functions).

9. Task I: The AI Model Implementation task involves downloading the dataset and implementing a logistic regression model for binary classification. This phase will focus on setting up the necessary data, training the model, and ensuring its integration with the overall system. The task is expected to take 10 days and depends on the successful completion of the data preparation and feature engineering steps.

10. Task J: The Deployment task focuses on deploying the ASP.NET applications to Azure and the FastAPI app to AWS. It involves using Docker for containerization and setting up GitHub Actions workflows for automation. This task ensures that all applications are successfully deployed and run smoothly in their respective cloud environments. The task is expected to take 15 days to complete and depends on the successful completion of the front-end and back-end implementations.

11. Task K: The Testing and Quality Control task aims to assess the functionality, performance, and reliability of the phishing detection website. It involves creating and running test cases, identifying and fixing any bugs or issues, and ensuring the website meets all requirements and performs as expected. This task will take 12 days to complete and is dependent on the completion of both the front-end and back-end implementations.

12. Task L: The Report task focuses on preparing a detailed report for the phishing detection website project. It includes compiling findings, outlining the development process, and documenting the results. This task is expected to take 25 days, reflecting the significant time and effort required to gather data and create a thorough report.

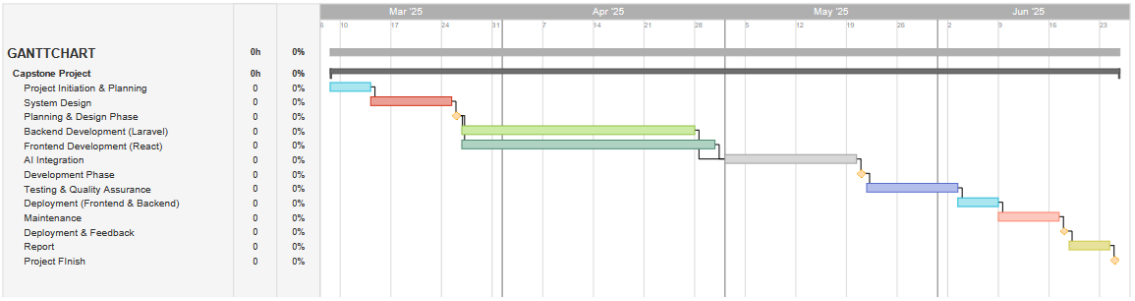


Figure 1 Gant Chart

Chapter 2: State of the Art

2.1 Introduction

In this chapter, we explore the current landscape of technologies and applications that relate to our project. Understanding what already exists helps us identify the strengths and limitations of similar platforms, and it allows us to position our work more effectively. We begin by looking at comparable applications that offer features similar to ours, followed by an overview of the technologies commonly used to build modern web applications. This includes database systems, backend frameworks, and frontend tools. By analyzing these elements, we can make more informed decisions in our own

development process and ensure that our project is aligned with current best practices.

2.2 Similar Applications

Our mission was to explore existing cybersecurity platforms with the goal of extracting valuable insights to inform the development of Security Guard. By thoroughly analyzing their strengths and weaknesses, we can identify opportunities to improve phishing detection and introduce enhanced features such as article sharing, group chat, and OCR for image phishing scanning. Through this process, we aim to craft a more comprehensive and effective solution that better serves users and provides superior protection against evolving phishing threats.

2.2.1 Zaker

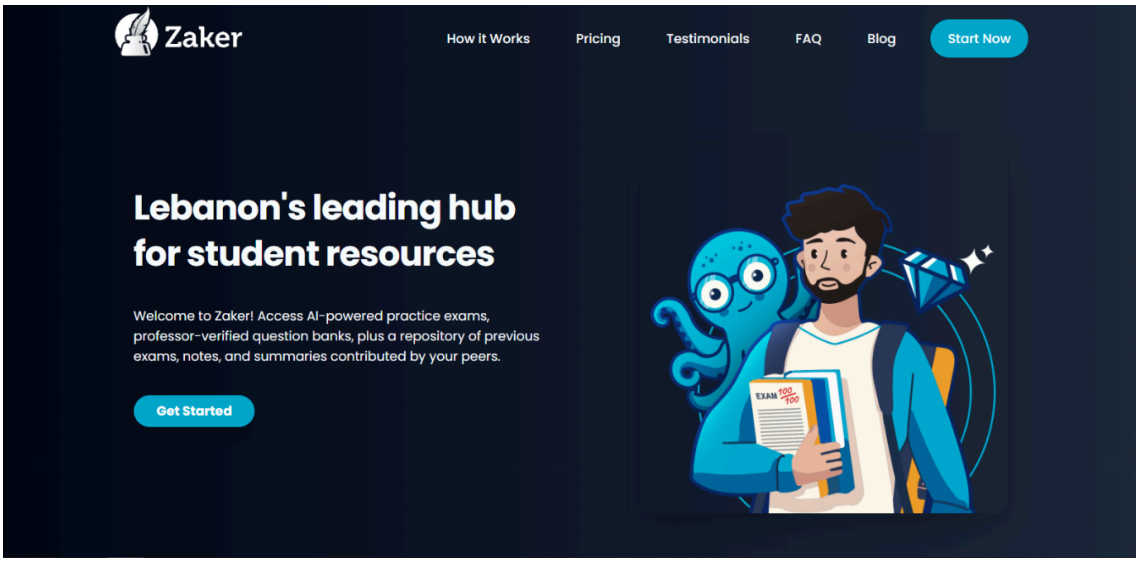


Figure 2 Zaker

Zaker is a platform allow students from different universities to share courses' documents, quizzes and summaries and uses the help of AI newly. According to their website description, Welcome to Zaker! Access AI-powered practice exams, professor-verified question banks, plus a repository of previous exams, notes, and summaries contributed by your peers. [1]

2.2.2 Preexam

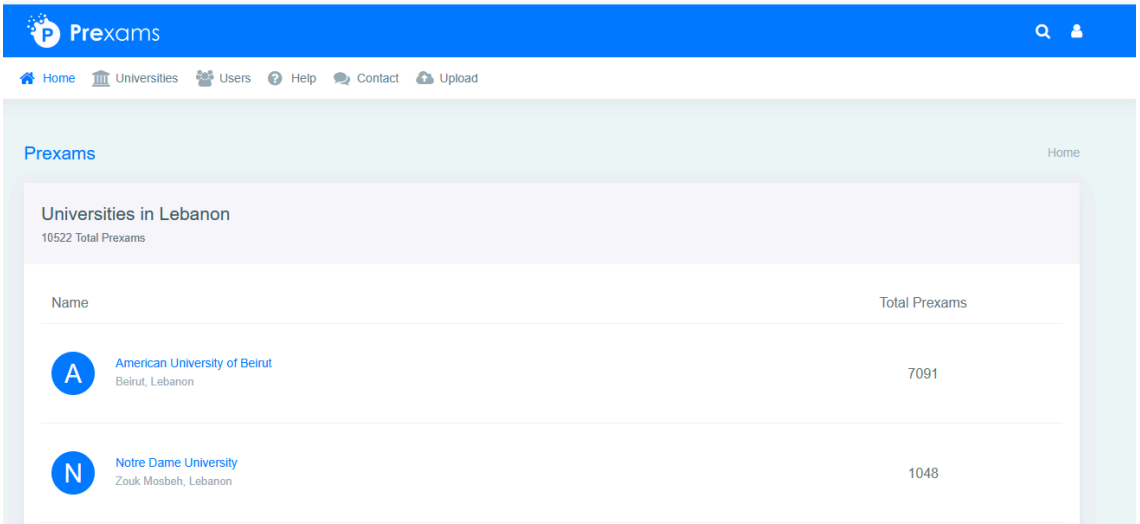


Figure 3 Preexam

Prexam is a platform where students can share test banks, summaries, and solutions, making it easier to prepare for exams. It fosters a collaborative learning environment

by allowing users to access and contribute study materials. This community-driven approach helps students enhance their knowledge and excel academically.[2]

2.2.3 CourseHero

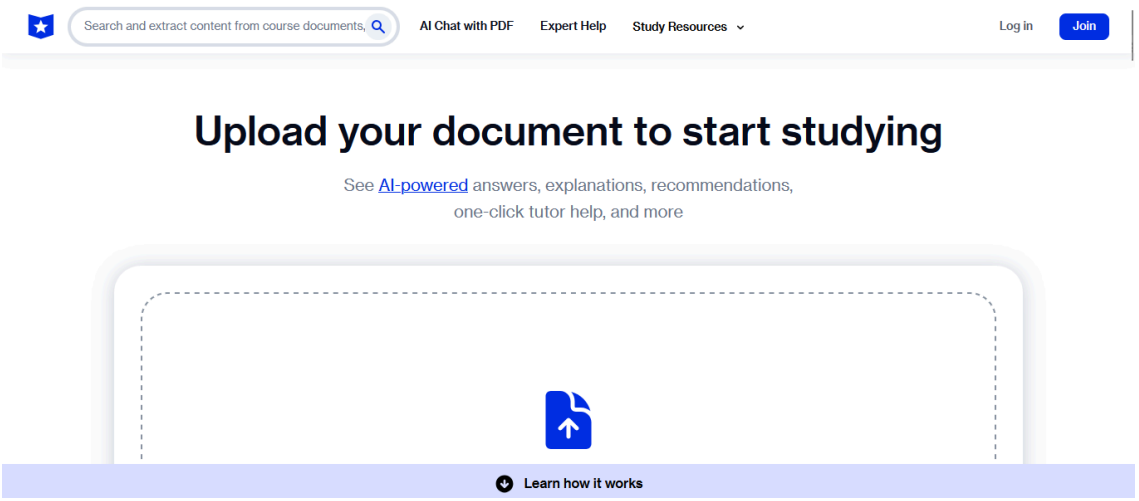


Figure 4 CourseHero

CourseHero is an online learning platform that provides educational resources to help students study and succeed in their academic pursuits. It offers access to a vast library of course-specific study materials, including lecture notes, textbook solutions, practice problems, essays, and study guides. [3]

Features	MU Connect	Zaker	Prexam	Course Hero
Register	✓	✓	✓	✓

Login/Logout	✓	✓	✓	✓
Upload Documents	✓	✓	✓	✓
Top Contributors	✓	✓	✓	✓
Free Version	✓	X	X	X
Review/ Favorites	✓	✓	X	✓
Filtering	✓	✓	✓	✓

Table 2 Similar Applications

2.3 Available Technologies

To build a modern and efficient web application, it's important to choose the right set of technologies. In this section, we explore the main tools and frameworks available for the database, backend, and frontend. Each of these layers plays a key role in how the application performs, scales, and interacts with users. By reviewing the most commonly used technologies, we can better understand which ones suit our project's needs and why we made certain technical choices.

2.3.1 Database Frameworks

Database frameworks are tools or sets of libraries that provide developers with an abstraction layer to interact with databases more efficiently. Here are some popular database frameworks:

a. **MySQL:**

MySQL is one of the most popular open-source relational database management systems (RDBMS). It is widely used in web development projects and integrates well with backend frameworks like Laravel. MySQL is known for its speed, reliability, and ease of use, especially for applications that require structured data and complex queries. [4]

b. **PostgreSQL:**

PostgreSQL is an advanced, open-source relational database system known for its robustness and support for complex queries, full ACID compliance, and extensibility. It is often chosen for projects that require high reliability, data integrity, and advanced features such as JSON support, stored procedures, and custom data types. [5]

c. **MongoDB:**

MongoDB is a NoSQL database that stores data in flexible, JSON-like documents. It's suitable for projects that need to handle large volumes of unstructured or semi-structured data and allows for rapid development and scaling. MongoDB is often used in modern web applications where flexibility and speed are priorities.[6]

2.3.2 Backend Frameworks

Backend frameworks are software frameworks designed to aid in the development of the server-side (backend) of a web application. Backend frameworks handle tasks such as database interactions, server-side logic, authentication, and more. Here are some popular backend frameworks:

a. ASP.Net Core:

An open-source web application framework developed by Microsoft. It is a cross-platform framework that enables developers to build high-performance and scalable web applications for multiple platforms. [7]

b. Laravel:

A PHP web application framework known for its elegant syntax and features. This technology is widely regarded as one of the top web frameworks and is considered the ideal software for backend development by many experts. [8]

c. Fast API:

Fast API is a modern web framework for building APIs with Python, known for its high performance, ease of use, and automatic interactive documentation. This technology is widely regarded as one of the top choices for developing fast and efficient backend applications, making it an ideal solution for many experts in API development [9].

2.3.3 Frontend Frameworks

Frontend frameworks are essential tools that help developers build interactive, responsive, and well-structured user interfaces. Instead of writing everything from scratch with plain HTML, CSS, and JavaScript, these frameworks provide ready-to-use components, structure, and best practices that speed up development and improve code quality. In this section, we'll look at some of the most commonly used frontend frameworks that could be considered for our project.

a. **Reactjs:**

React is a popular JavaScript library developed by Facebook for building dynamic user interfaces. It allows developers to create reusable UI components and efficiently update the UI when data changes. React is widely used for modern single-page applications (SPAs) and integrates well with backend APIs. [10]

b. **Angular:**

Angular is a full-fledged frontend framework developed by Google. It provides everything needed to build large-scale web applications, including routing, form validation, state management, and more. It uses TypeScript and enforces a more structured development approach, which is ideal for enterprise-level applications.[11]

c. **Vue.js**:

Vue is a progressive JavaScript framework that focuses on simplicity and flexibility. It's easy to integrate with existing projects and is great for building both small and large-scale applications. Vue offers two-way data binding, component-based architecture, and a gentle learning curve, making it beginner-friendly. [12]

2.3.3 AI Integration & Language Model APIs

Frontend frameworks are collections of pre-written code and tools that assist developers in building the user interface (UI) and user experience (UX) of a web application. The frontend refers to the part of a software application or a website that users interact with directly. Here are some popular frontend frameworks:

a. **OpenAI API (ChatGPT/ GPT Models)**

OpenAI provides powerful language models like GPT-4 and GPT-3.5 through an easy-to-use API. These models can be integrated into web applications to handle natural language processing tasks such as text generation, summarization, translation, question answering, and more. The API is flexible, with support for function calling and fine-tuning, and it's commonly used in chatbots and intelligent automation features [13].

b. **Google Cloud Vertex AI (Gemini API)**

Google's Vertex AI offers access to models like PaLM 2 and Gemini through the Generative AI Studio and APIs. It enables developers to build applications with

capabilities such as text understanding, summarization, image recognition, and more. It also integrates well with other Google services like Google Sheets, Drive, and Cloud Storage, making it useful for workflows involving document or image processing [14].

2.3 Conclusion

After exploring and comparing different technologies, we carefully selected a tech stack that balances ease of development, performance, and long-term maintainability.

For the **frontend**, we chose **React JS** due to its flexibility, component-based architecture, and strong community support. It allows us to build a dynamic and responsive user interface that delivers a smooth experience for our users.

On the **backend**, we used **Laravel (PHP)**, a powerful and elegant framework that offers built-in tools for authentication, routing, and database management. Laravel's ecosystem made it easy to structure our backend, handle API endpoints, and integrate with the frontend.

For **database management**, we used **SQLite** during the development phase because it's lightweight and requires no server setup—ideal for rapid testing. For the production environment, we switched to **MySQL**, a reliable and scalable relational database system that's well-suited for handling real-world traffic and data operations.

To add **AI functionality**, we explored multiple tools and settled on integrating **LLM APIs** such as **Google Gemini** for tasks like image-based data extraction and smart processing. These tools allowed us to automate and enhance certain parts of the user experience using advanced AI models.

This combination of tools allowed us to build a modern, scalable, and intelligent web application while keeping development efficient and focused. Each technology was chosen based on how well it fits our project goals, team skills, and future scalability needs.

Chapter 3: Requirement Analysis

This chapter outlines the requirements for the Capstone Project, a web-based platform designed to facilitate academic resource sharing, class exchange requests, study groups, and event management for university students and faculty. The requirements are derived from the system's codebase, folder structure, and implemented features.

3.1 High-level Business Objectives:

- **Centralize Academic Resources**
  Provide a unified digital platform where university students can easily upload, share, and access study materials like quizzes, summaries, and past exams.

- **Promote Collaboration and Peer Learning**
  Encourage knowledge sharing among students to improve academic performance and engagement across different faculties and years.

- **Improve Accessibility and Usability**
  Offer a modern, user-friendly interface (built with React) that makes it easy for

students to find and contribute content from any device.

- **Optimize Storage and Efficiency**
  Use a checksum-based deduplication system to reduce redundant file uploads and save server storage while maintaining data integrity.

- **Support Course Section Management**
  Help students manage their academic schedules by allowing section swap requests, minimizing registration issues and improving flexibility.

- **Enable Scalable Future Growth**
  Build a backend (Laravel) designed to support future features like AI quiz generation, recommendation systems, or mobile app integration.

3.2 Functional Requirements Table:

Table (6) shows the functional requirements of our project.

Functional Requirements
1. Email / password registration and login (A M S)
2. Google OAuth sign-in / sign-up flow (A M S)
3. Role-based access enforcement via stored JWT & localStorage.role (admin, moderator, student) (system)
4. Logout / token revocation (A M S)
5. View own profile details, avatar, roles and statistics (A M S).
6. Edit personal profile information & upload avatar (A M S)
7. Browse any user's public profile (A M S)
8. Admin dashboard: list, search and filter all users (A)

9. Toggle another user's active / suspended status (A)
10. Paginated, searchable course catalogue (A M S)
11. Course detail page with description, sections, faculty and resources (A M S)
12. Section-swap Requests module a. Create a swap request (current vs desired section) (S) b. View available requests & filter/sort (A M S) c. Apply to a request (S) d. Withdraw application (S) e. Request owner: view applicants, accept / decline (S) f. Cancel or edit own request (S) g. Delete any request (admin/moderator override) (A M)
13. Degree chart visualization: track completed & pending courses (A M S)
14. GPA / grade calculator tool (A M S)
15. Upload learning resources (files, links, notes) (S)
16. Edit / delete own resource (S)
17. Moderate or delete any resource (A M)
18. Resource catalogue: search, tag filters, pagination & sorts (A M S)
19. Detailed resource viewer with in-browser document preview (A M S)
20. Save / unsave resource to personal collection (A M S)
21. Clubs' module a. List all clubs and club posts (A M S) b. Vote in elections / polls when voting window open (S) c. Create / edit / delete club posts, schedules, polls (A M)
22. Study Group module a. Create study-group post (topic, schedule) (S)

b. Browse and join groups (A M S) c. Delete or edit any group post (A M)
23. Events module a. Calendar view of upcoming events (A M S) b. Create, edit, delete event (A M)
24. Notifications center: real-time alerts for requests, events, resources, etc. (A M S)
25. In-app comments on resources, events, or groups (present where components imported) (A M S)
26. Rich-text / markdown Create-Post modal (used across resources, clubs, study groups) (A M S according to parent module permissions)
27. Global dashboard summarizing notifications, saved resources, upcoming events (A M S)
28. Responsive UI with light/dark mode toggle (Chakra UI) (A M S)
29. Pagination, sorting and client-side filtering helpers across all list views (system)
30. Error handling & toast notifications for all CRUD operations (system)

Table 3 Functional Requirements

3.3 Non-Functional Requirements Tables:

Security

- User data must be protected using JWT-based authentication and HTTPS encryption.

- Uploaded files should be validated and scanned to prevent the upload of malicious content.

- Users should only be able to access resources based on their permissions/roles.

- The system shall enforce HTTPS for all client-server communication to ensure data integrity and confidentiality.

Usability

- The system should have an intuitive user interface, accessible to students with minimal technical knowledge.

- The user interface shall provide clear feedback (e.g., success, error messages) after user actions

- Navigation shall be consistent and user-friendly across all pages.

- The UI should be responsive and optimized for both desktop and mobile devices.

Maintainability

- The codebase is modular, with separation of concerns (controllers, services, components).
- Configuration and environment variables are managed via .env file

- Dependencies shall be managed via Composer (PHP) and npm/yarn (JavaScript).

- The codebase shall follow consistent naming conventions and coding standards.

- The application shall use version control (e.g., Git) with clear commit messages and branching strategy.

34. Conclusion:

In this chapter, we laid the foundation for our project by clearly identifying both the functional and non-functional requirements. By understanding what the system needs to do and how it should perform under different conditions, we were able to define a clear roadmap for development. From security and performance to usability and maintainability, each requirement was chosen to ensure the application is not only functional but also reliable, user-friendly, and scalable. This analysis will guide our technical decisions in the next phases and help us stay aligned with our project goals.

Chapter 4: Application Modeling and Design

4.1 Introduction

This chapter presents the design of our application using UML diagrams. These models help visualize how the system works and how users interact with it. We include use case diagrams, scenarios, activity diagrams, and, if applicable, a class diagram to guide development and ensure a clear structure.

4.2 UML Diagrams:

4.2.1 Use case Diagram:

The figure below shows our use-case without authentication for simplicity note that some functions need authentication. Our use case contains five actors, which are public user that serves as a viewer, User, Admin, Transportation Manager and Content Manager where all their functionalities are listed below in the figure.

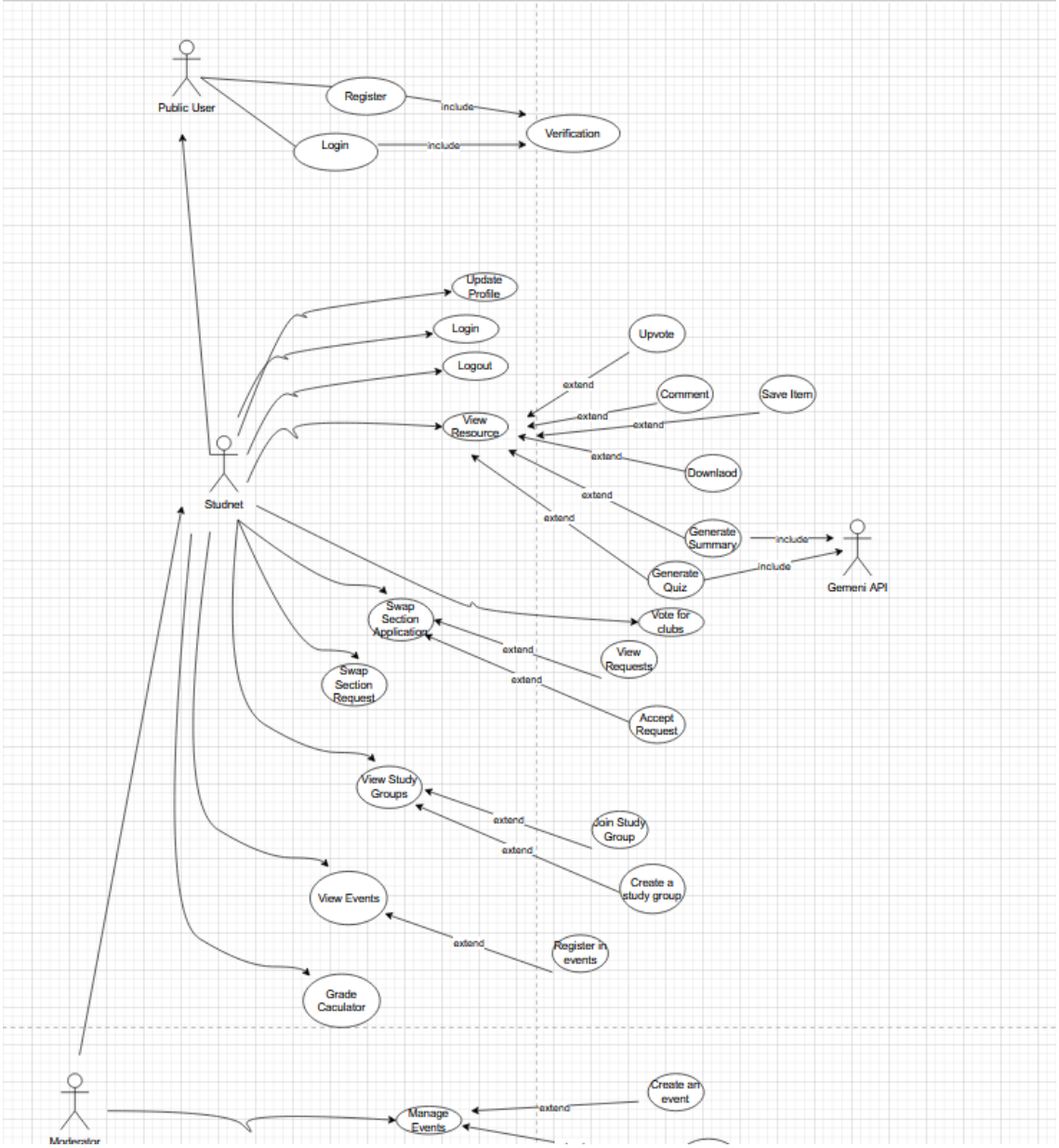


Figure 5 Use Case Diagram Part 1

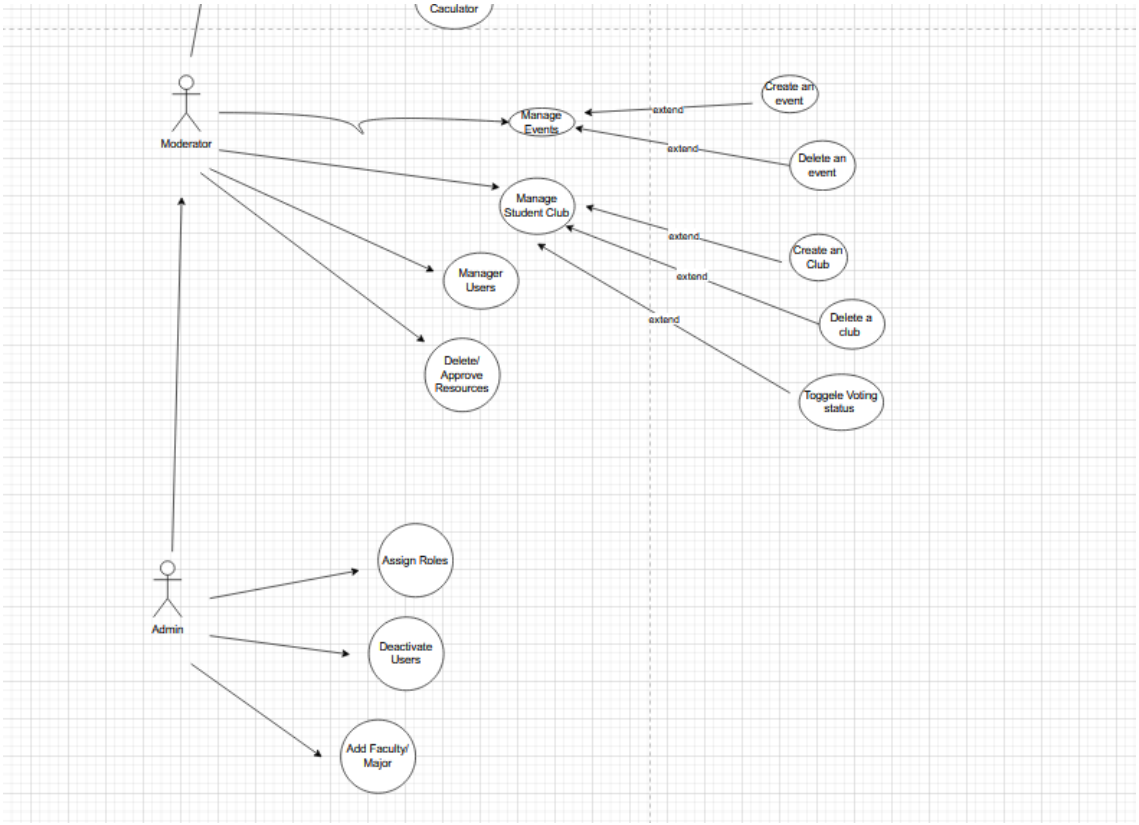


Figure 6 Use Case Diagram Part 2

4.2.2 Scenarios:

The tables that are presented below are some sample scenarios for our project: Post an article, and scan a link.

Name	Join a Study Group

Actors	Student
Description	This use case describes how a student can browser available study groups and join one if there is capacity
Pre-condition	-Student must be logged in -Study group exists and it is not full
The Flow of Events	• Student navigates to list of study groups. • Student selects a group. • System displays details. • Student clicks “Join”. • System checks capacity. • System adds student to study_group_user. • System confirms joining.
Post Condition	-Student is added to the study group -Study group member count is updated
Exceptions	• -If the group is full, the system displays an error message and prevents joining.

Table 4 Scenario 1 Join a Study Group

Name	Upload a Resource
Actors	Student
Description	This use case describes how a student uploads a resource (file) and links it to a course.
Pre-condition	-Student must be logged in -The course must exists

The Flow of Events	• Student opens “Upload Resource” page. • Fills in details (title, course, etc.). • Selects file(s). • System calculates checksum. • System checks for duplicates: • - If exists: link to existing attachment. • - Else: save new file. • System saves resource record. • System links resource and attachment(s). • System confirms upload.
Post Condition	- Resource is saved and visible to other students. - Files are stored with deduplication handled if needed.
Exceptions	• if file upload fails, show an error. • If no course is selected, system prompts for selection

Table 5 Scenario2 Upload a Resource

4.2.3 Activity Diagrams

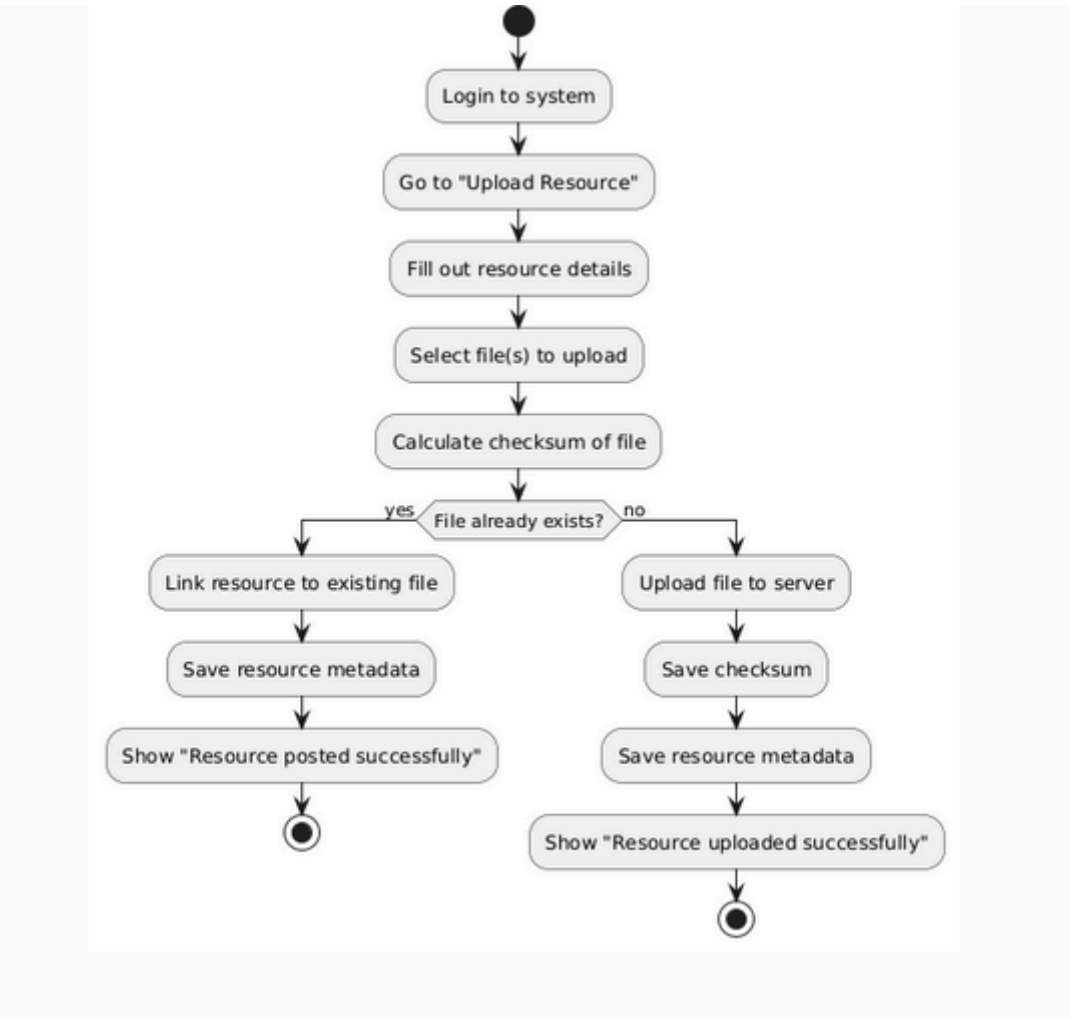


Figure 7 Upload a Resource Activity Diagram

This activity diagram illustrates the workflow a student follows to upload a resource (such as notes, summaries, or a quiz) to the system. The process starts with the student filling out resource details and selecting a file to upload. The system then calculates the file's checksum to check for duplicates. If the file already exists, the system links the new resource record to the existing file to avoid duplication. Otherwise, it uploads the new file, stores the checksum, and saves the resource metadata. The student then receives a confirmation message that the resource was successfully posted.

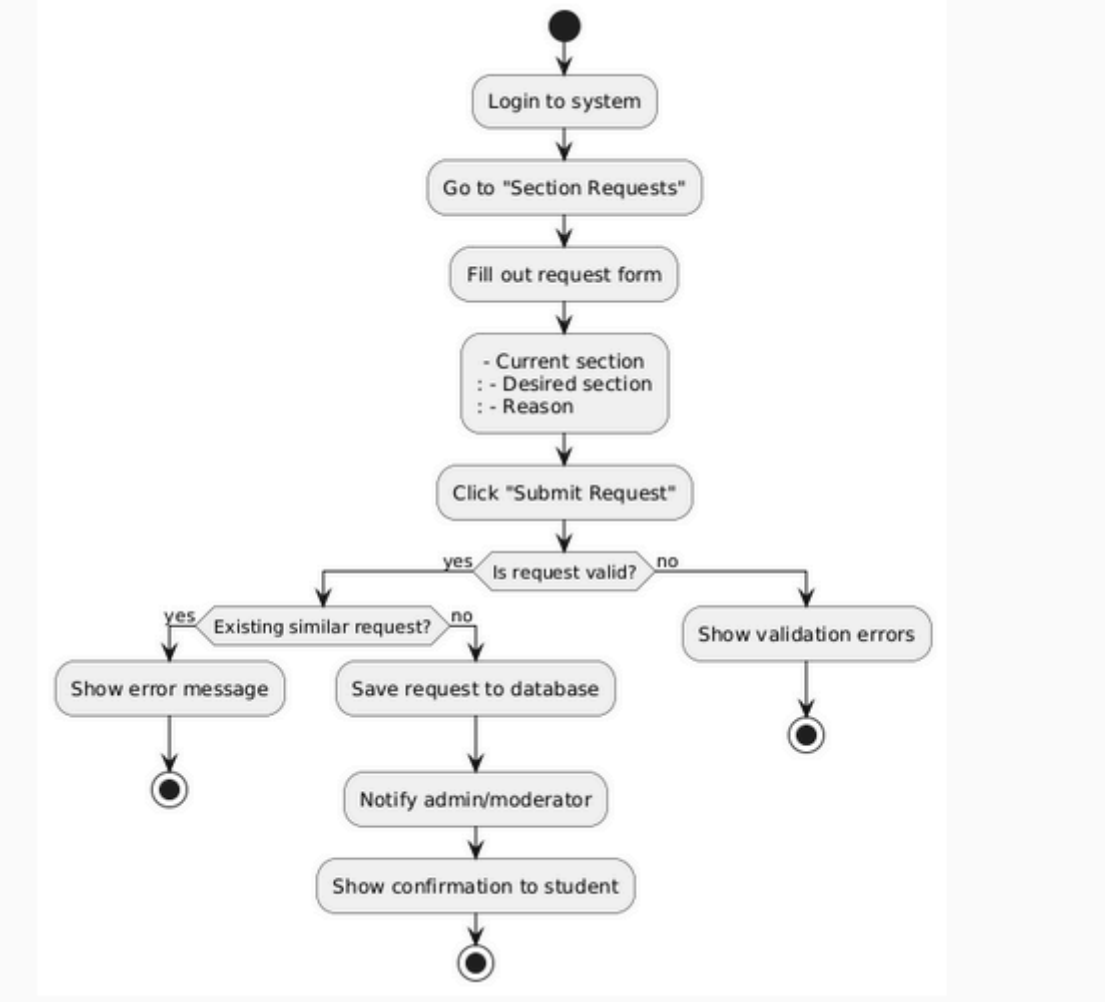


Figure 8 Make a Request to Change Section Activity Diagram

This activity diagram shows how a student submits a request to change their course section. The student logs in, navigates to the section request page, and completes a form indicating their current and desired sections along with a reason for the request. The system first validates the request details. If the request is valid and no similar request already exists, it saves the request in the database and notifies the relevant admin or moderator. If the request is invalid or a duplicate exists, the system displays an appropriate error message. The process ends by informing the student of the result.

4.2.4 Database Design

The database diagram is represented in Figure 11 that illustrates the relations with six tables.

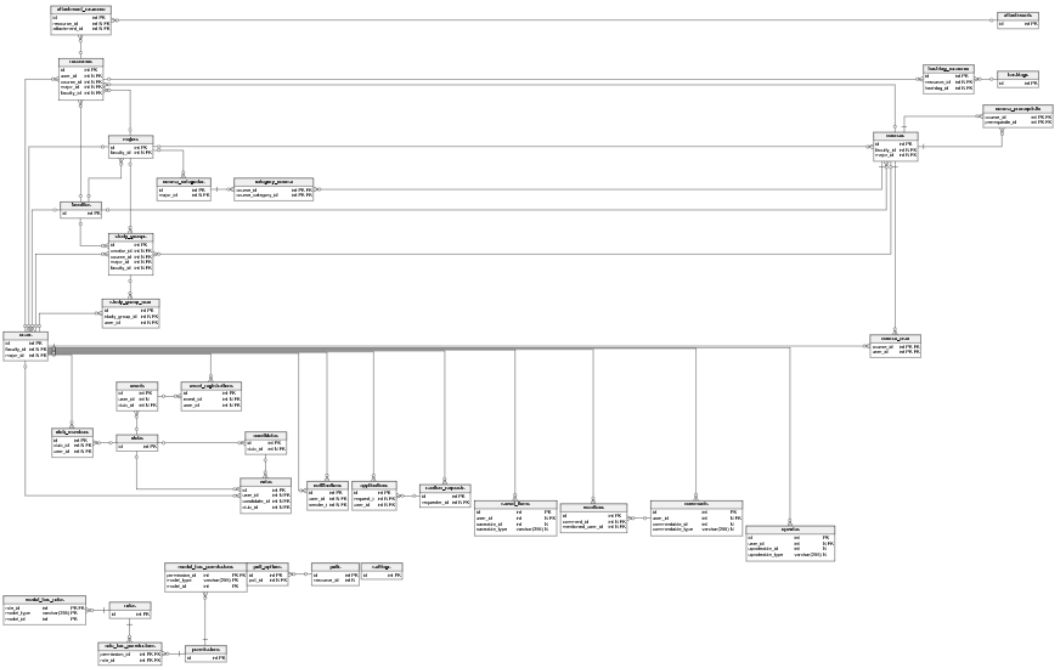


Figure 9 ER Diagram Diagram

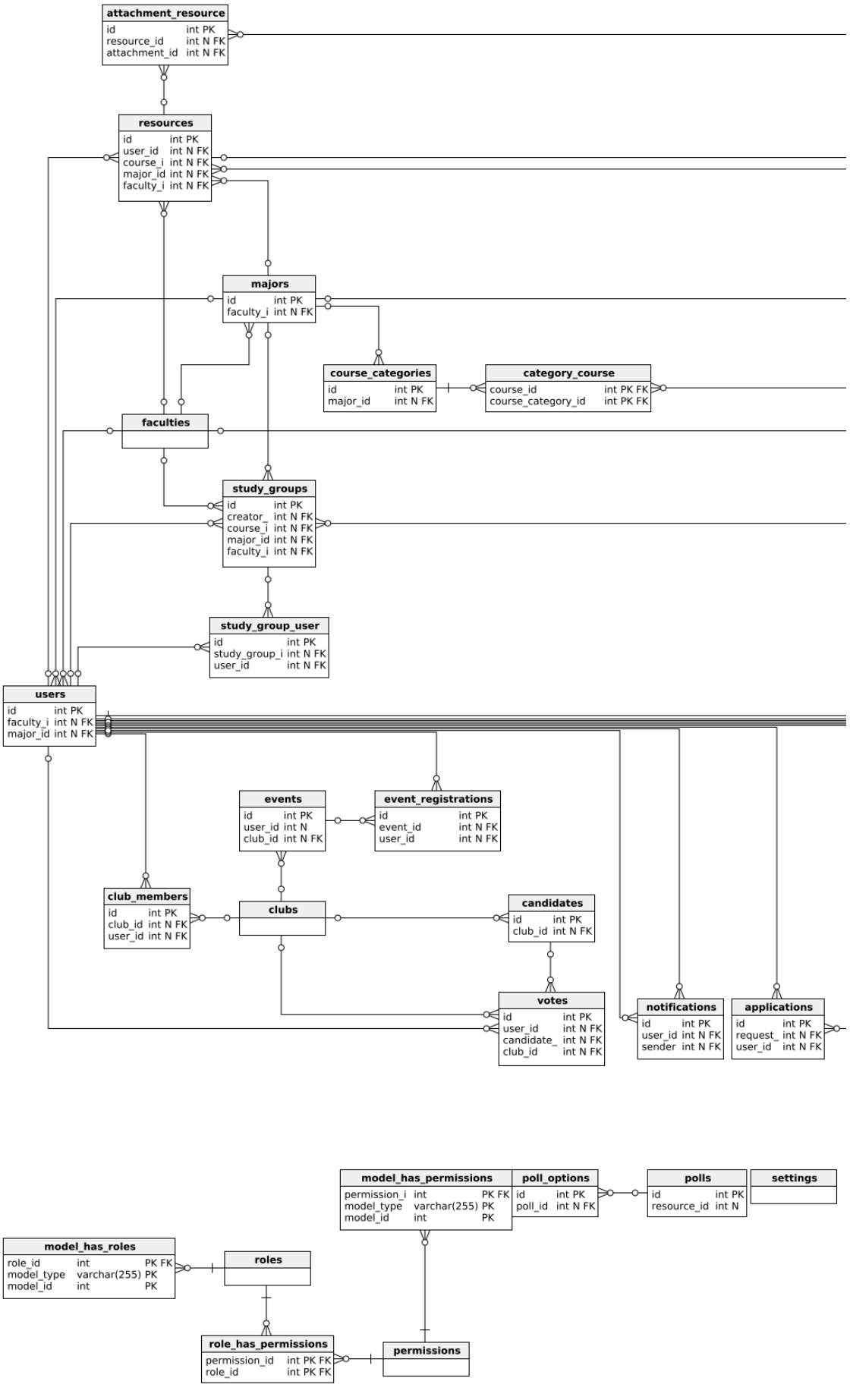

attachment_resource
id int PK
resource_id int N FK
attachment_id int N FK
resources
id int PK
user_id int N FK
course_i int N FK
major_id int N FK
faculty_i int N FK
majors
id int PK
faculty_i int N FK
course_categories
id int PK
major_id int N FK
category_course
course_id int PK FK
course_category_id int PK FK
faculties
study_groups
id int PK
creator_ int N FK
course_i int N FK
major_id int N FK
faculty_i int N FK
study_group_user
id int PK
study_group_i int N FK
user_id int N FK
users
id int PK
faculty_i int N FK
major_id int N FK
events
id int PK
user_id int N
club_id int N FK
event_registrations
id int PK
event_id int N FK
user_id int N FK
club_members
id int PK
club_id int N FK
user_id int N FK
clubs
candidates
id int PK
club_id int N FK
votes
id int PK
user_id int N FK
candidate_ int N FK
club_id int N FK
notifications
id int PK
user_id int N FK
sender int N FK
applications
id int PK
request_ int N FK
user_id int N FK
model_has_permissions
permission_i int PK FK
model_type varchar(255) PK
model_id int PK
poll_options
id int PK
poll_id int N FK
polls
id int PK
resource_id int N
settings
model_has_roles
role_id int PK FK
model_type varchar(255) PK
model_id int PK
roles
role_has_permissions
permission_id int PK FK
role_id int PK FK
permissions

Figure 10 ER Diagram Zoom in part 1

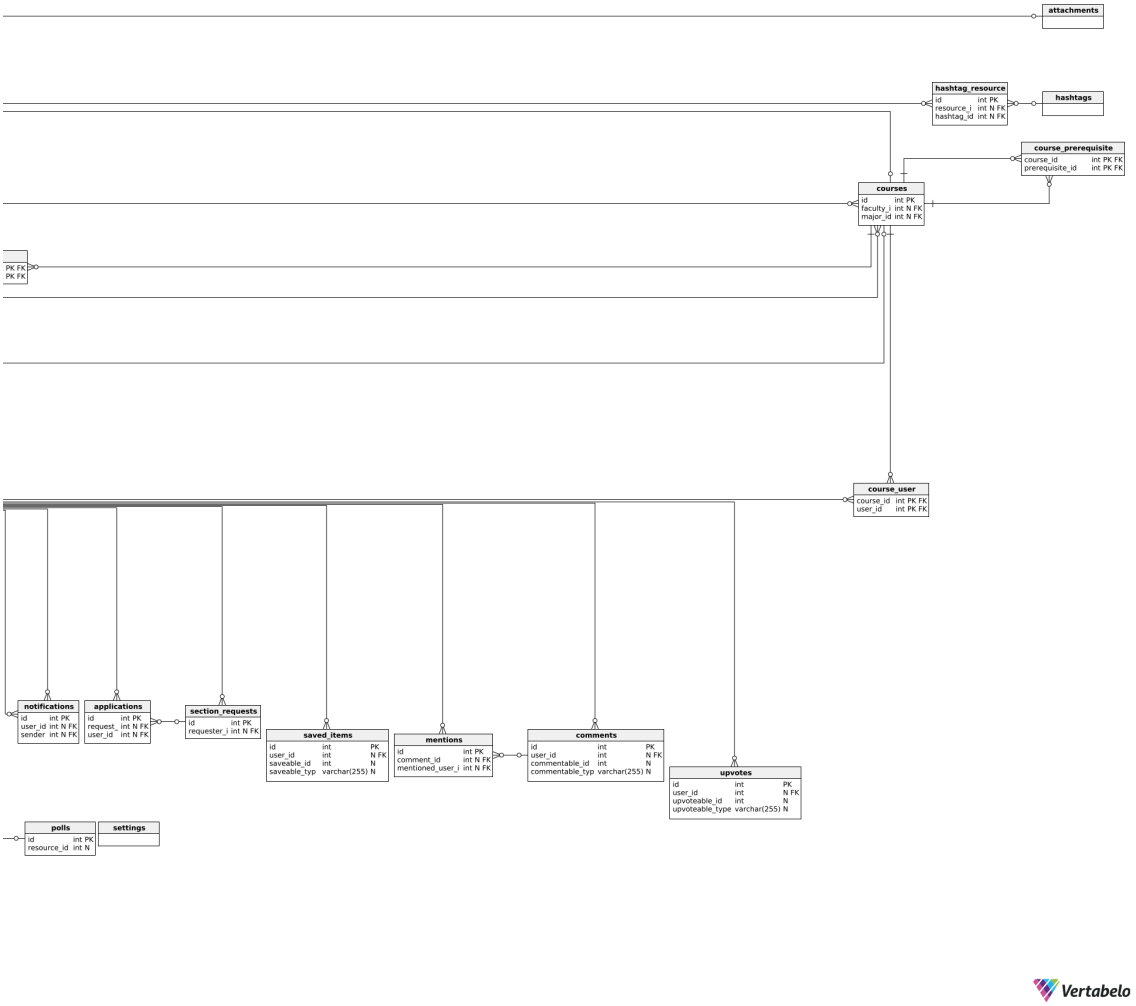


Figure 11 ER Diagram Zoom in part 2

4.3 Conclusion:

In conclusion, various techniques are employed in the development and management of websites, including use case diagrams, activity diagrams, and relational schemas. Use case diagrams, which illustrate system functionality and user requirements, guide the design process. Activity diagrams depict workflows, simplifying website operations. Relational schemas define the structure and organization of the website's database, ensuring data integrity and efficient storage and retrieval. Collectively, these tools provide a comprehensive approach to website creation, promoting effective project management, client satisfaction, and a functional system.

Chapter 5: Application Implementation

5.1 Introduction

This chapter presents the implementation phase of the application, highlighting the key technologies, frameworks, and tools used throughout the development process. It outlines the specific versions adopted to ensure compatibility and stability. The chapter also includes representative screenshots of the user interface and core sections of the source code to demonstrate how the application's functionality was brought to life. This practical insight bridges the gap between design and execution, providing a clear picture of the final product in action.

5.2 Used Technologies

There are several technologies that are available, but we aim to use the technologies that made our website more friendly and easy to use. We will provide you with the technologies we use that leads to this successful smooth website with perfect functionalities.

5.2.1 Used Backend Technologies

For the backend, we selected **Laravel (version 12)**, a robust PHP-based framework known for its security, scalability, and developer-friendly features. Laravel provides a solid foundation for building RESTful APIs, handling authentication, and managing application logic efficiently. During the development phase, we utilized **SQLite** for its simplicity and lightweight setup, while **MySQL** was adopted for the production environment to ensure reliability, scalability, and performance in managing structured data. To support AI-powered functionalities such as document summarization and quiz generation, we integrated the **Gemini API**, leveraging advanced machine learning capabilities to enhance user experience and streamline content processing.

5.2.2 Used Frontend Technologies

For the frontend, we have chosen **React JS (version 18)**, a widely adopted JavaScript library renowned for its ability to create dynamic, responsive, and maintainable user interfaces. Leveraging React's component-based architecture allows for better code organization and reusability. To enhance the visual appeal and usability of the application, we integrated **Chakra UI** alongside **Tailwind CSS**, combining flexibility with modern design principles. Communication between the frontend and backend is facilitated through **RESTful APIs**, ensuring efficient data exchange and a seamless user experience.

5.2.3 AI Integration Technologies

To enable intelligent, automated features within the application, we integrated the **Gemini 1.5 Flash API** by Google, a powerful large language model designed for high-speed and context-aware AI tasks. This integration allows the system to perform advanced natural language processing tasks such as **document summarization** and **quiz generation** based on user-uploaded content. By leveraging Gemini's capabilities,

we enhanced the overall functionality and interactivity of the application, providing users with valuable insights and content automation. The integration was achieved through secure API calls from the backend, ensuring smooth and efficient communication with the AI service.

5.3 Website Screenshots

The following figures are screenshots of our dynamic website that will show you several pages and how they appear. The screenshots serve as the visual appearance of the website and gives overview for the while website. Every figure represents a particular page, highlighting its unique layout, content organization, and interactive features.

Figure 12 Welcome Page Screenshot

The image displays the homepage of "MU Connect", alongside options to "Sign Up" or "Login".

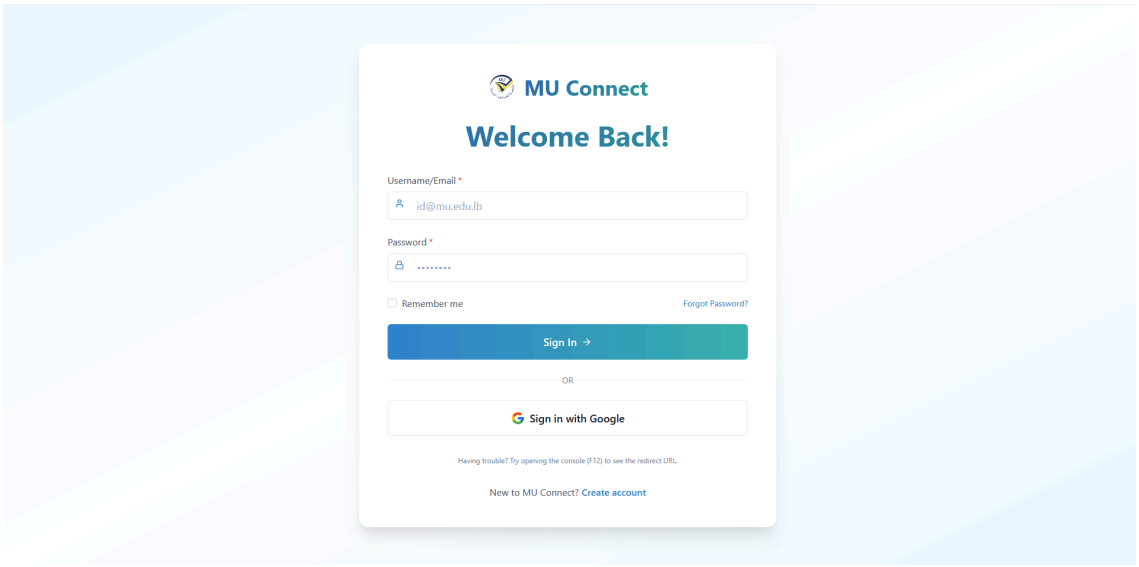


Figure 13 Login in PageScreenshot

The image displays the "MU Connect" login page with fields for username/email and password, along with sign-in options.

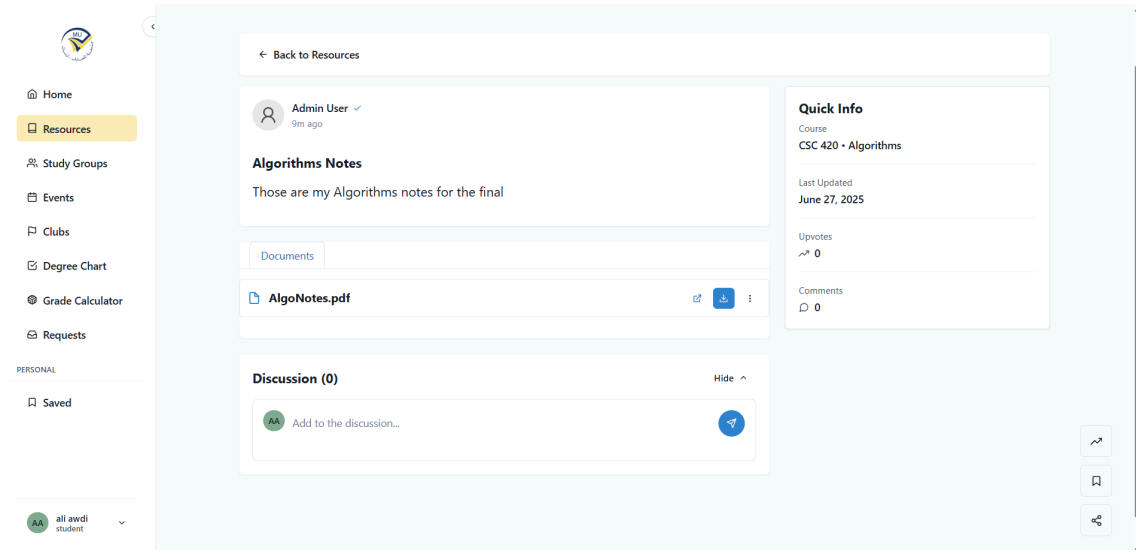


Figure 14 Resource View PageScreenshot

The image displays the "Resources" section of an online platform, showing "Algorithms Notes" posted by "Admin User" with an attached "AlgoNotes.pdf" file, and sections for "Quick Info," "Upvotes," "Comments," and a "Discussion" area.

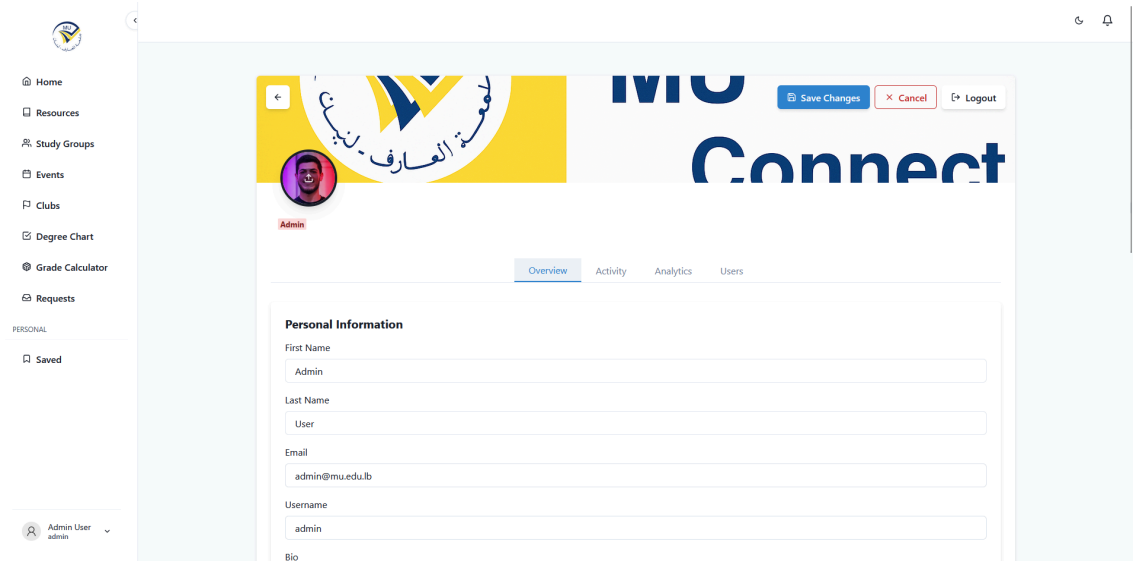


Figure 15 User Profile Update PageScreenshot

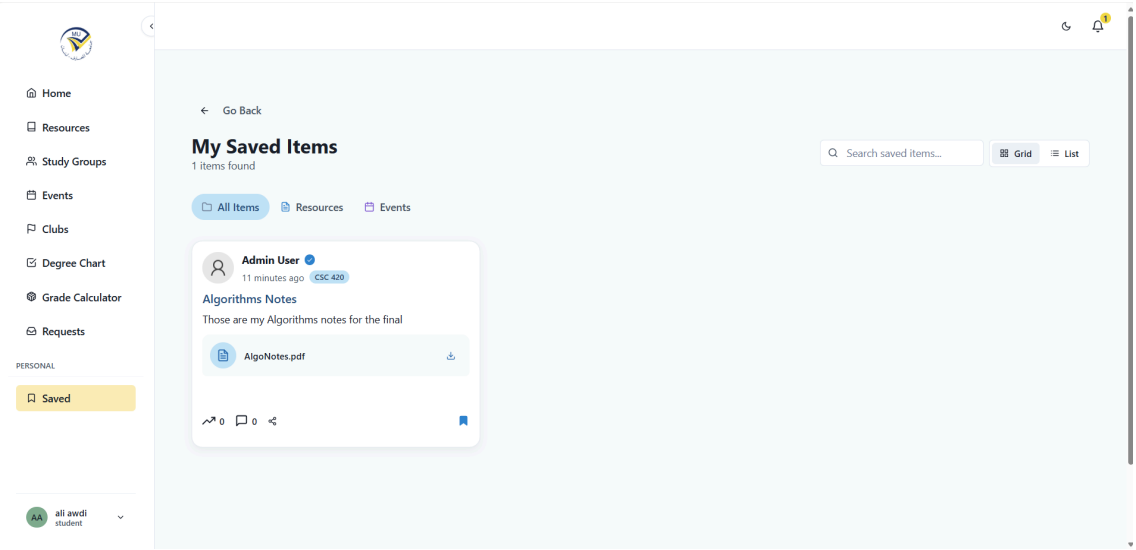


Figure 16 Saved Items Page Screenshot

This screenshot shows the saved item page for user, where user has a resource saved in the saved item Page

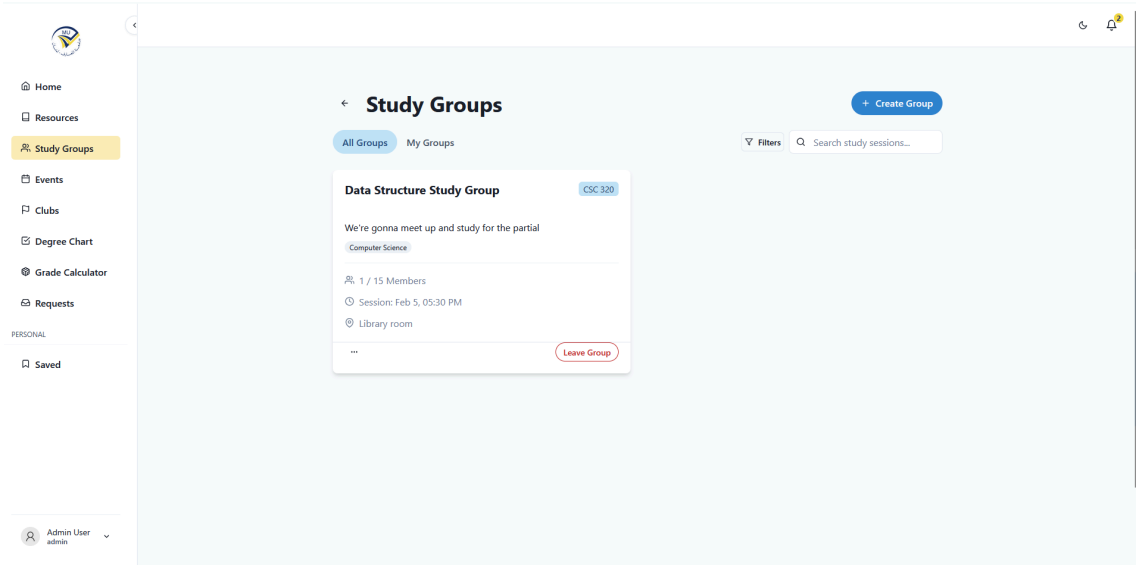


Figure 17 Study Group Page Screenshot

This page shows the Study Group page where a user created a data structure study group.

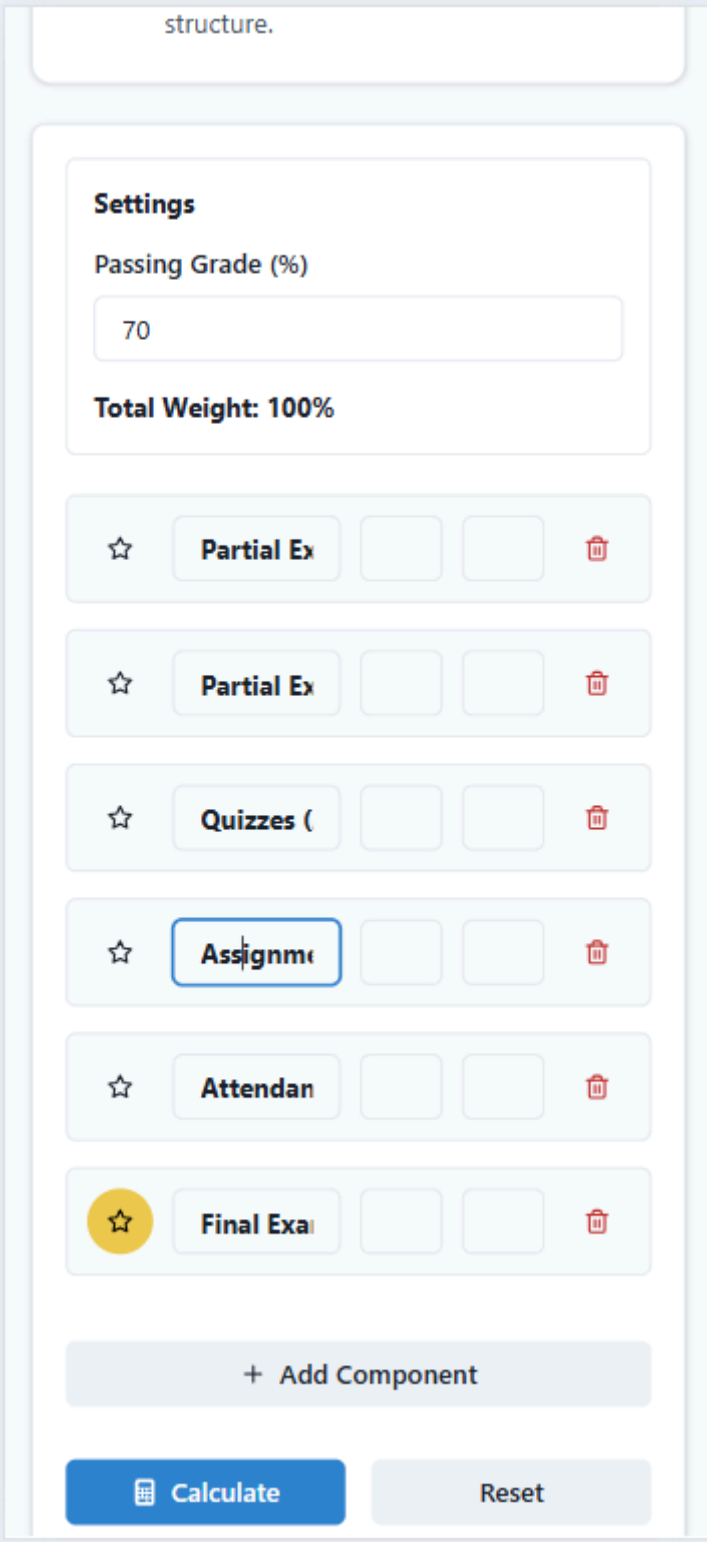


Figure 18 Grade Calculator Page Screenshot

The image displays a grade calculator page , showing a "Passing Grade (%)" of 70 and "Total Weight: 100%", with various grade components like "Partial Ex", "Quizzes", "Assignments", "Attendance", and "Final Exam" listed with input fields, a star icon, and a delete icon, along with "Add Component", "Calculate", and "Reset" buttons.

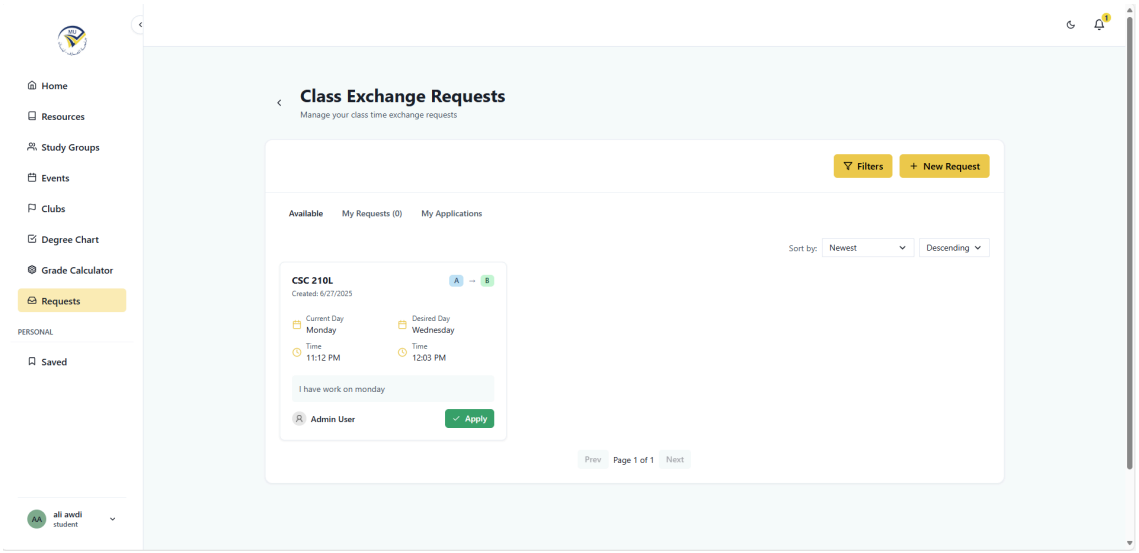


Figure 19 Request Page Screenshot

The image shows a "Class Exchange Requests" page with a sidebar menu, displaying an available request for "CSC 210L" with details about current and desired day/time, a message "I have work on Monday", and an "Apply" button.

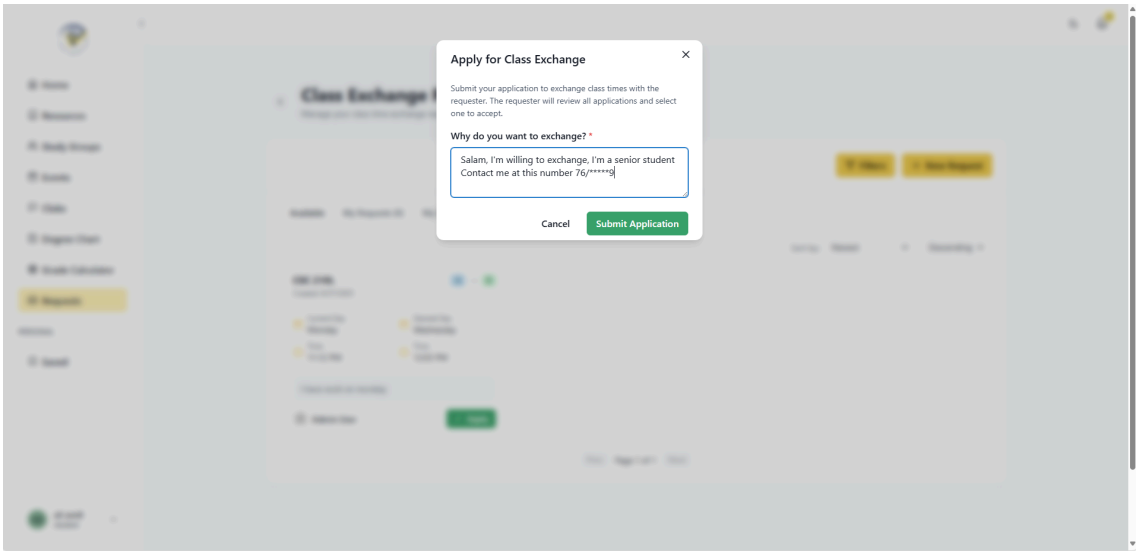


Figure 20 Request Page Screenshot

This Screenshot shows a user applying for application to exchange/swap classes with other user.

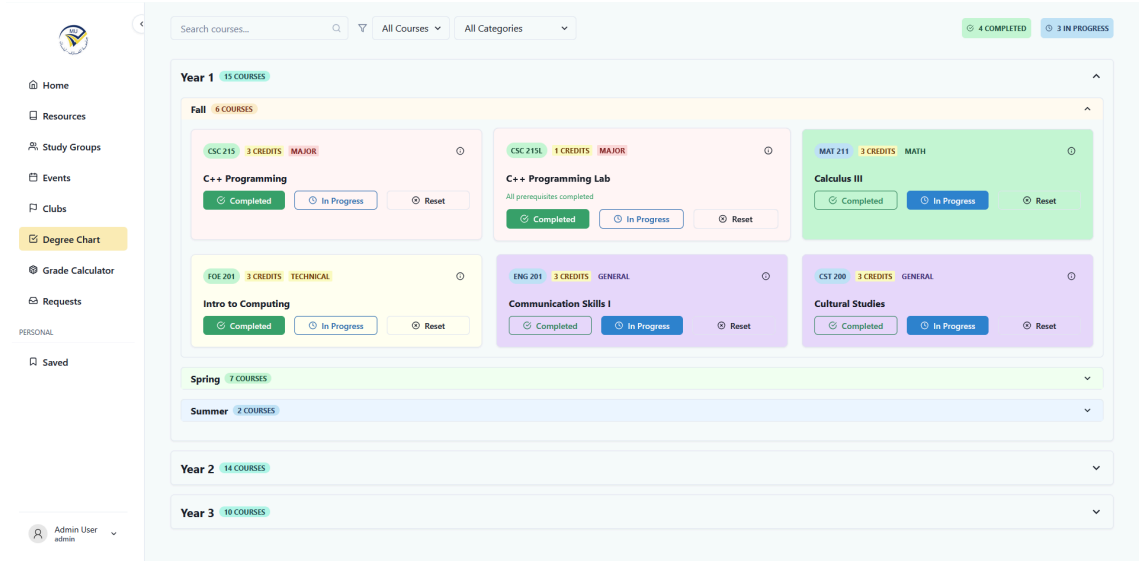


Figure 21 Degree Chart Page Screenshot

Figures 35 shows the Degree chart page showing courses organized by academic year (Year 1, Year 2, Year 3) and semester (Fall, Spring, Summer), with each course block indicating credits, type (e.g., Major, Math, Technical, General), and status options (Completed, In Progress, Reset).

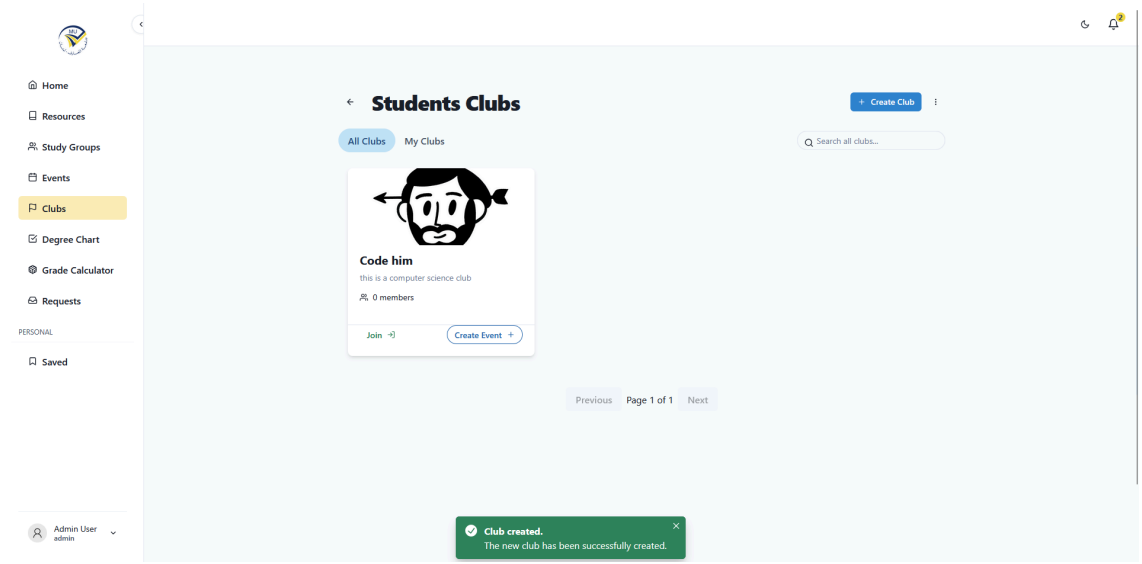


Figure 22 Clubs Page Screenshot

Here we got the clubs page, the user is an admin so he can create a club.

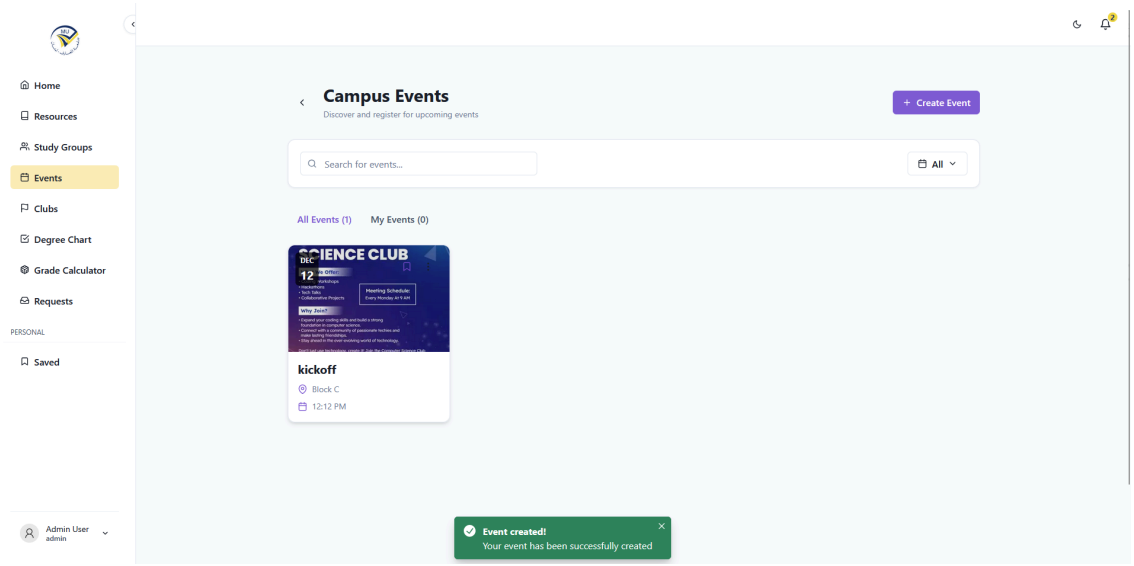


Figure 23 Event Page Screenshot

Figure 37 shows the event page, where the user here is an admin so he can create an event.

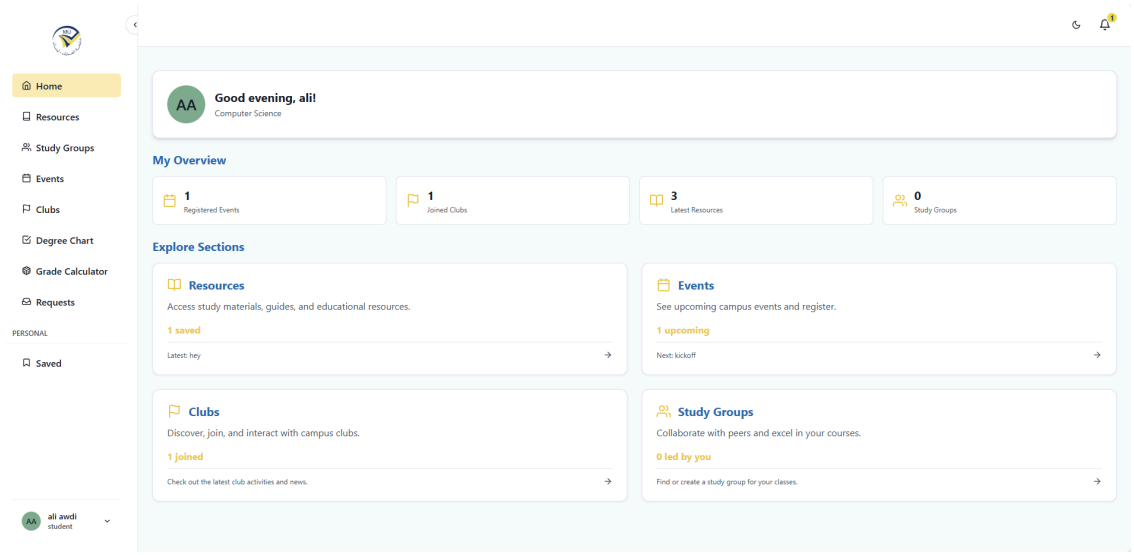


Figure 24 Dashboard page Screenshot

This screenshot shows dashboard, with an "My Overview" section summarizing registered events, joined clubs, latest resources, and study groups, followed by "Explore Sections" for Resources, Events, Clubs, and Study Groups with brief descriptions and quick links.

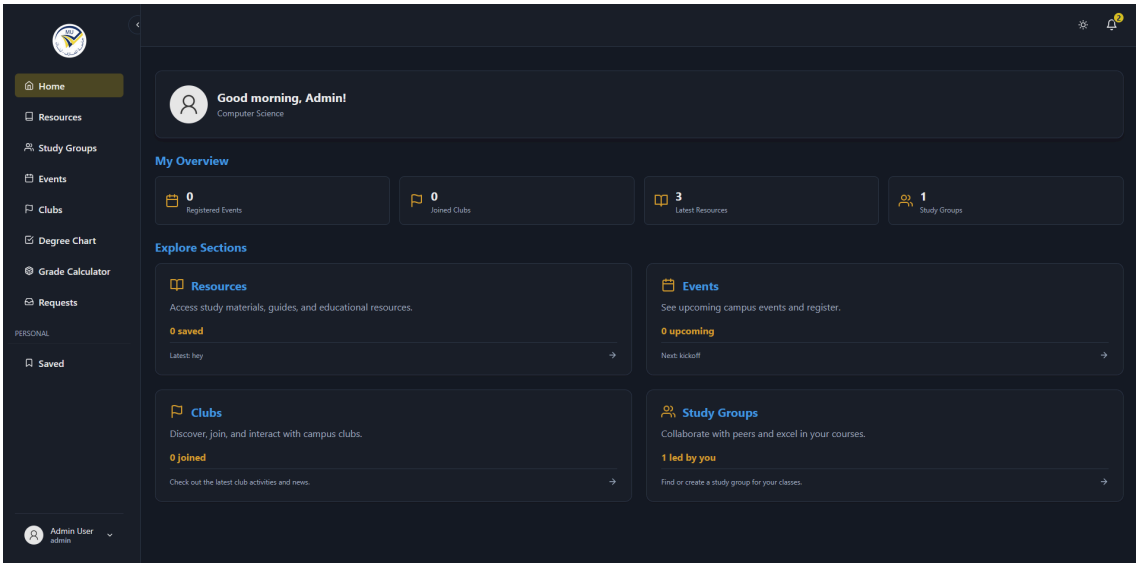


Figure 25 Dashboard in Dark Mode

Figure 39 shows the Dashboard in Dark Mode

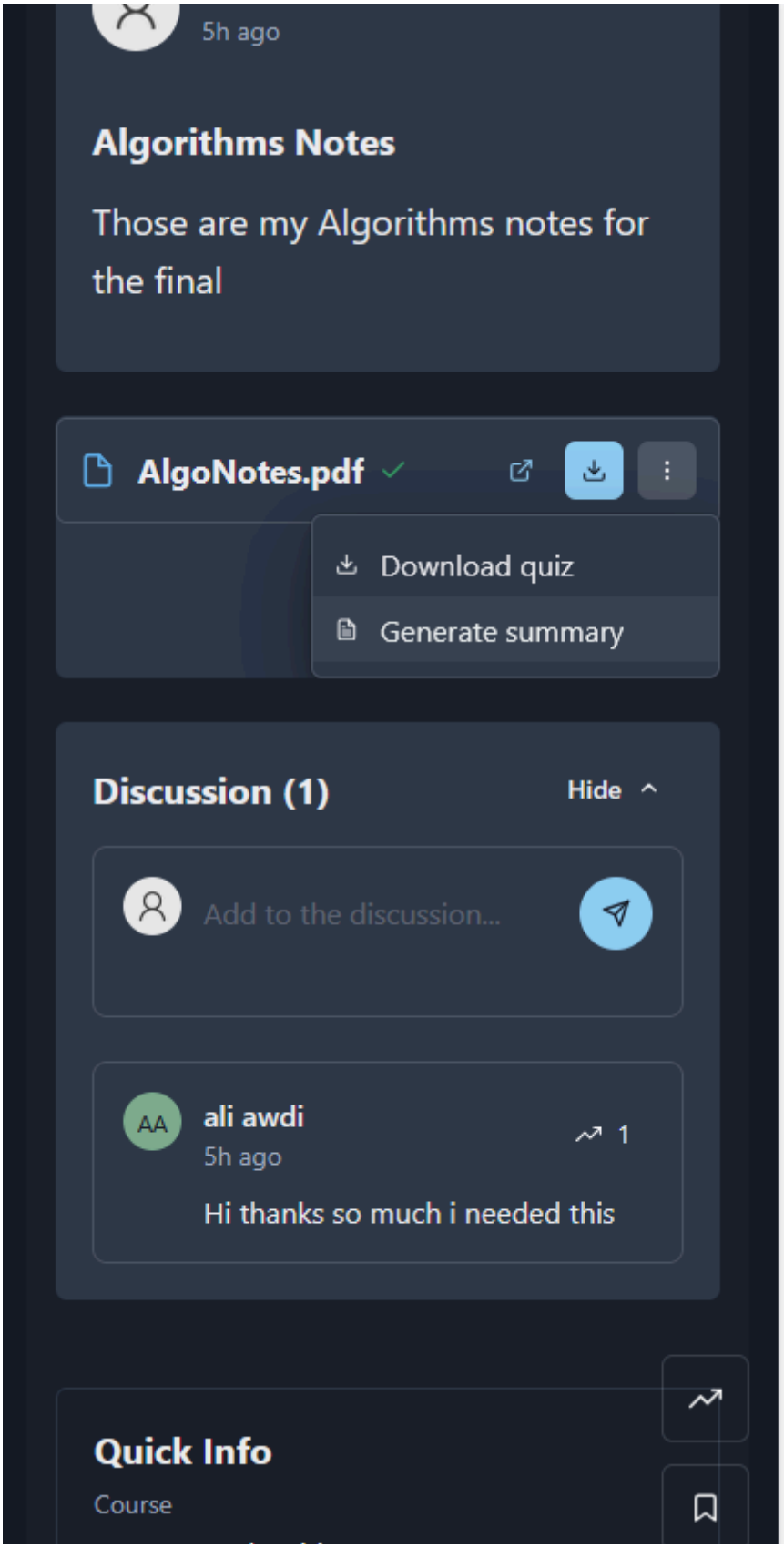
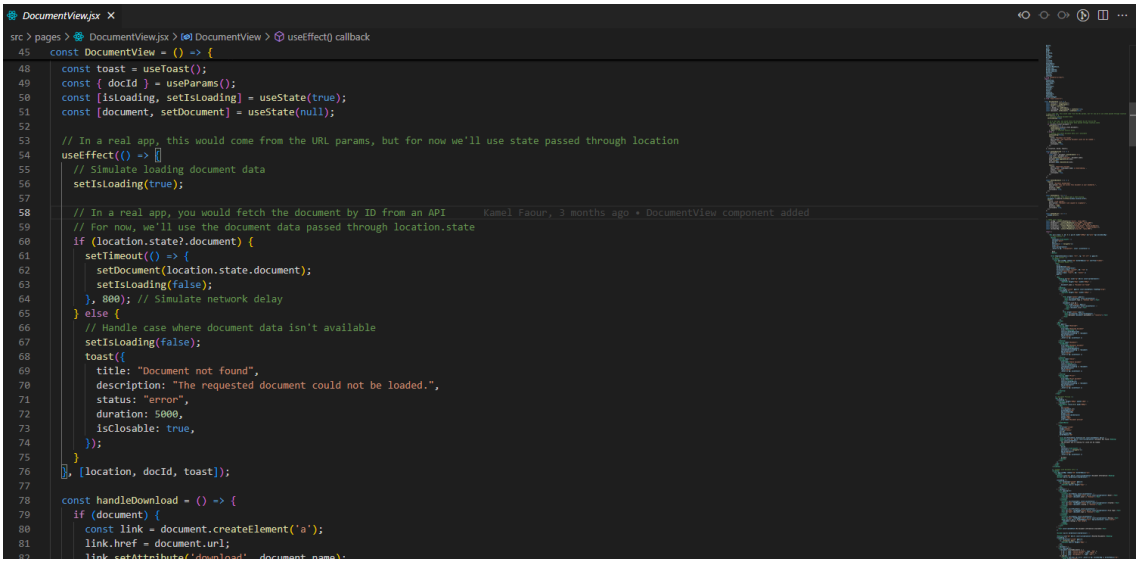


Figure 26 Resource AI Features Screenshot

Figures 26 shows the options of Resource with AI features like generating a quiz or generate a summary .

5.4 Sample Code Screenshots

5.4.1 Frontend Samples:



```jsx
const DocumentView = () => {
  const toast = useToast();
  const { docId } = useParams();
  const [isLoading, setIsLoading] = useState(true);
  const [document, setDocument] = useState(null);

  // In a real app, this would come from the URL params, but for now we'll use state passed through location
  useEffect(() => {
    // Simulate loading document data
    setIsLoading(true);

    // In a real app, you would fetch the document by ID from an API
    // For now, we'll use the document data passed through location.state
    if (location.state?.document) {
      setTimeout(() => {
        setDocument(location.state.document);
        setIsLoading(false);
      }, 800); // Simulate network delay
    } else {
      // Handle case where document data isn't available
      setIsLoading(false);
      toast({
        title: "Document not found",
        description: "The requested document could not be loaded.",
        status: "error",
        duration: 5000,
        isClosable: true,
      });
    }
  }, [location, docId, toast]);

  const handleDownload = () => {
    if (document) {
      const link = document.createElement('a');
      link.href = document.url;
```

Figure 27 DocumentView Code Screenshot

This code defines a React component called DocumentView that is responsible for displaying a document. It simulates fetching document data, either from location.state or, in a real application, from an API using a document ID. It handles loading states and displays an error message if the document is not found or cannot be loaded, along with providing a function to handle document downloads.

```jsx
import { Card, Box, Flex, Avatar, Input, Text } from "@chakra-ui/react";

const ResourcePostInput = ({ openPostModal, cardBg, borderColor, accentColor, mutedText }) => (
  <Card mb={6} bg={cardBg} cursor="pointer" onClick={openPostModal} border="2px solid" borderColor={accentColor} shadow="md">
    <Box p={{ base: 4, md: 6 }}>
      <Flex gap={4} direction={{ base: "column", md: "row" }} align="center">
        <Avatar size="md" src="https://bit.ly/dan-abramov" />
        <Input
          placeholder="Upload a document, share a quiz, or post a student-related question..."
          isReadOnly
          _placeholder={{ color: mutedText, fontWeight: 500 }}
          cursor="pointer"
          borderColor={borderColor}
          _hover={{ borderColor: accentColor }}
        />
      </Flex>
      <Text fontSize="xs" color={mutedText} mt={2} ml={{ md: 16 }}>
        Note that posts are under professional and academic moderation.
      </Text>
    </Box>
  </Card>
);

export default ResourcePostInput;
```

Figure 28 ResourcePost Input Code Screenshot

This code defines a React component called ResourcePostInput which provides an input field for users to upload documents, share quizzes, or post student-related questions. It is styled to resemble a card and, upon clicking, it likely triggers an "OpenPostModal" (a modal for creating a new post). The component also includes a disclaimer about professional and academic moderation.

5.4.2 Backend Samples

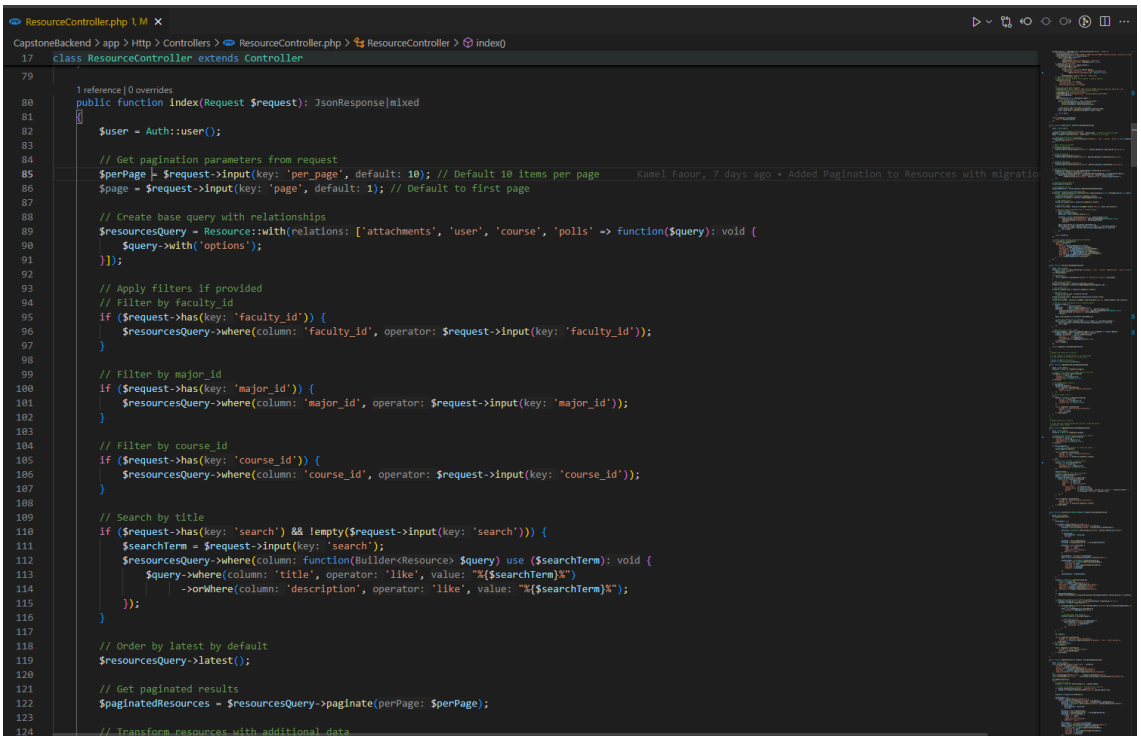
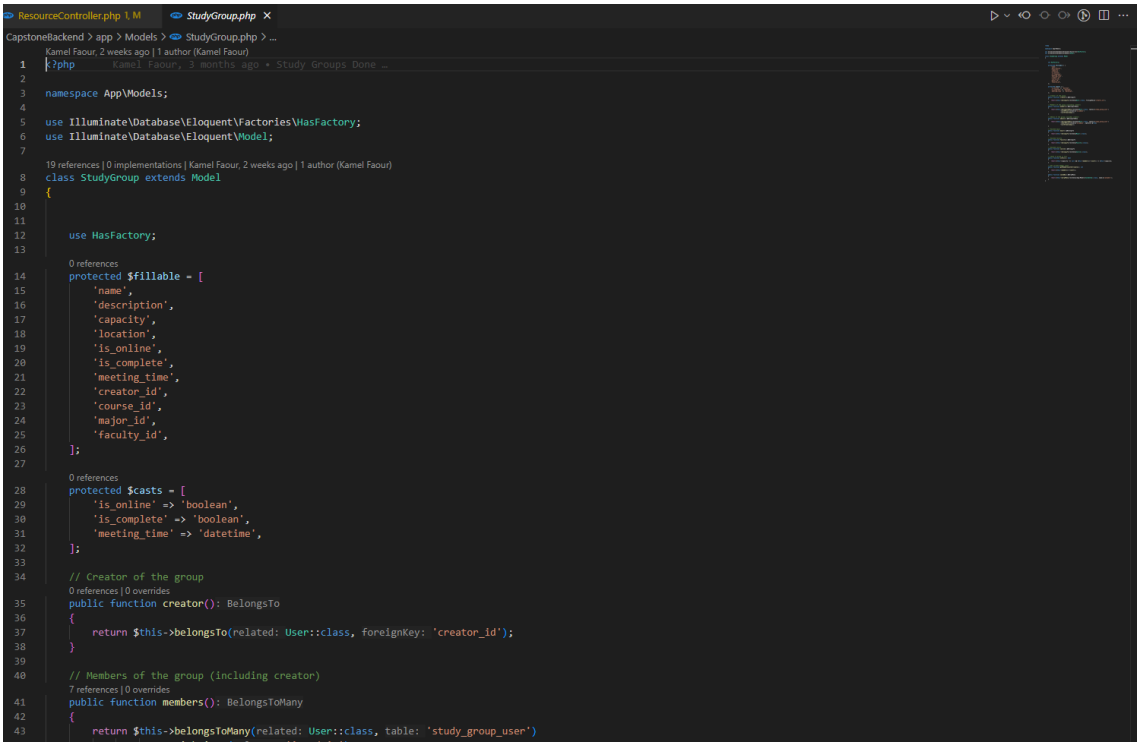


Figure 29 Index Function in ResourceController Code Screenshot

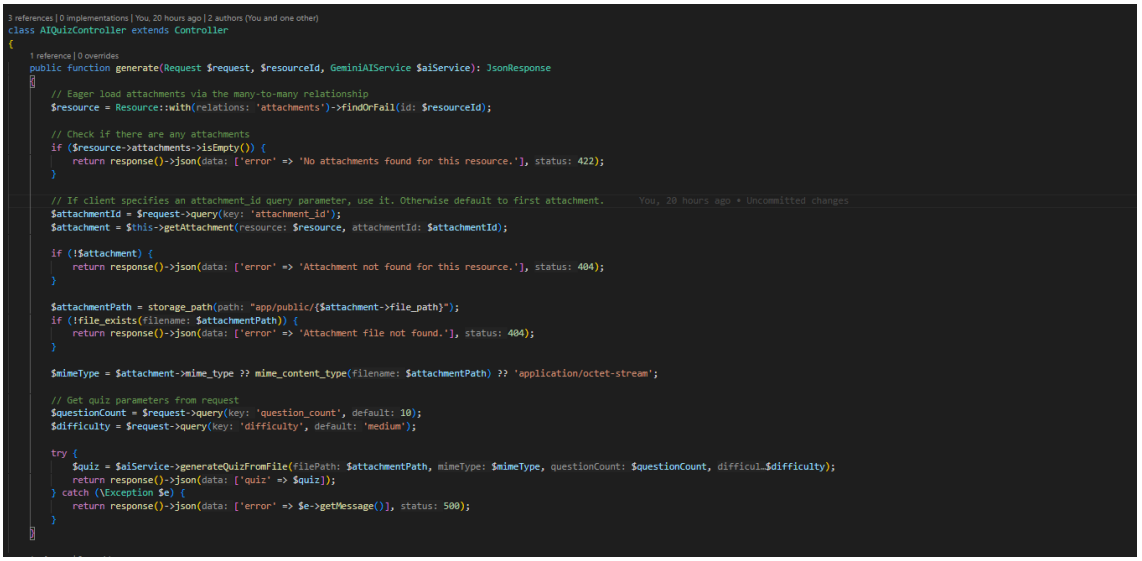
This screenshot shows a part of the index function written inside the resource controller, This index function retrieves a paginated list of resources, applying filters based on faculty_id, major_id, course_id, and a search term for title or description. For each resource, it includes related data like attachments, user, course, and poll information (including user's vote if applicable), along with upvote count, user's upvote status, and comment count. Finally, it returns the resources and pagination metadata as a JSON response.



```php
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class StudyGroup extends Model
{

    use HasFactory;

    protected $fillable = [
        'name',
        'description',
        'capacity',
        'location',
        'is_online',
        'is_complete',
        'meeting_time',
        'creator_id',
        'course_id',
        'major_id',
        'faculty_id',
    ];

    protected $casts = [
        'is_online' => 'boolean',
        'is_complete' => 'boolean',
        'meeting_time' => 'datetime',
    ];

    // Creator of the group
    public function creator(): BelongsTo
    {
        return $this->belongsTo(related: User::class, foreignKey: 'creator_id');
    }

    // Members of the group (including creator)
    public function members(): BelongsToMany
    {
        return $this->belongsToMany(related: User::class, table: 'study_group_user')
```

Figure 30 Study Group Model Screenshot

The above figure shows the Study Group model, StudyGroup class defines an Eloquent model for managing study groups. It specifies the fillable attributes for creating and updating study groups (e.g., name, description, capacity, location), casts certain attributes to appropriate data types (e.g., is_online and is_complete to boolean, meeting_time to datetime), and establishes relationships with other models such as User (for creator, members, and admins), Major, Faculty, and Course. Additionally, it includes helper methods to check if a group is full (isFull), get the current member count (getMemberCountAttribute), and manage saved study groups (savedBy).



```
3 references | 0 implementations | You, 20 hours ago | 2 authors (You and one other)
class AIQuizController extends Controller
{
    1 reference | 0 overrides
    public function generate(Request $request, $resourceId, GeminiAIService $aiService): JsonResponse
    {
        // Eager load attachments via the many-to-many relationship
        $resource = Resource::with(relations: 'attachments')->findOrFail(id: $resourceId);

        // Check if there are any attachments
        if ($resource->attachments->isEmpty()) {
            return response()->json(data: ['error' => 'No attachments found for this resource.'], status: 422);
        }

        // If client specifies an attachment_id query parameter, use it. Otherwise default to first attachment.      You, 20 hours ago • Uncommitted changes
        $attachmentId = $request->query(key: 'attachment_id');
        $attachment = $this->getAttachment(resource: $resource, attachmentId: $attachmentId);

        if (!$attachment) {
            return response()->json(data: ['error' => 'Attachment not found for this resource.'], status: 404);
        }

        $attachmentPath = storage_path(path: "app/public/{$attachment->file_path}");
        if (!file_exists(filename: $attachmentPath)) {
            return response()->json(data: ['error' => 'Attachment file not found.'], status: 404);
        }

        $mimeType = $attachment->mime_type ?? mime_content_type(filename: $attachmentPath) ?? 'application/octet-stream';

        // Get quiz parameters from request
        $questionCount = $request->query(key: 'question_count', default: 10);
        $difficulty = $request->query(key: 'difficulty', default: 'medium');

        try {
            $quiz = $aiService->generateQuizFromFile(filePath: $attachmentPath, mimeType: $mimeType, questionCount: $questionCount, difficul…$difficulty);
            return response()->json(data: ['quiz' => $quiz]);
        } catch (\Exception $e) {
            return response()->json(data: ['error' => $e->getMessage()], status: 500);
        }
    }
```

Figure 31 AI Quiz Controller Screenshot

This screenshot shows the AIQuiz Controller generate method.This generate function, part of a Laravel controller, handles the generation of a quiz from a resource's attachment using a GeminiAIService. It first retrieves the specified resource and its attachments, validates the existence of the attachment file, and determines its MIME type. It then extracts quiz parameters (question count and difficulty) from the request and, using the GeminiAIService, attempts to generate a quiz from the attachment's content. Finally, it returns the generated quiz as a JSON response or an error message if any issues occur (e.g., no attachments, file not found, or AI service errors).

5.5 Conclusion

In conclusion, this chapter provided a practical overview of how the application was actually built and what technologies were used to bring it to life. By including real screenshots of the application's interfaces and snippets of the underlying code, we aimed to demonstrate not just the final product but also the quality and structure of its implementation. This section ties together the design concepts with real-world execution, showing how initial plans were transformed into a working system. Overall, this implementation lays a strong foundation for future improvements and ensures that the application remains reliable, maintainable, and ready to scale as needed.

Chapter 6: General Conclusion

6.1 Summary

This project set out to solve a real-world problem through a practical application, moving from initial idea and requirements gathering to design, implementation, and testing. Throughout the process, we focused on creating a functional system that is user-friendly, secure, and scalable. Each chapter demonstrated how we translated theoretical concepts into a working solution, from modeling use cases and designing the database to integrating the frontend and backend. The result is an application that meets the core requirements while leaving room for further enhancements and optimizations. This journey has not only produced a usable product but also deepened our understanding of full-stack development and project management in a real-world context.

6.2 Challenges

Like any real-world project, this application came with its fair share of challenges. One of the biggest difficulties was balancing the project's scope with the time we had as a small team. Deciding which features were essential and which could be postponed was not always easy, especially when trying to keep the application practical yet not overly complicated.

Another challenge was integrating different technologies and making sure they worked together smoothly. Sometimes, unexpected bugs would appear during deployment or testing that required hours of debugging and research to fix.

Working as a team — just me and my partner — was also a valuable learning experience. We had to make sure we stayed on the same page, merged our code without conflicts, divided tasks fairly, and maintained clear communication throughout. It wasn't always simple, but it definitely made us better at collaborating and trusting each other's work.

Finally, there were moments of frustration when facing unfamiliar frameworks or libraries, but overcoming these obstacles taught us to read documentation more effectively and find creative solutions under pressure. Despite these difficulties, every challenge strengthened our skills, our teamwork, and our confidence for future projects.

6.3 Functionality Satisfaction

Overall, we're very satisfied with the functionalities we were able to implement in this project. Not only did we cover the core features that were part of our original idea, but we also managed to expand on that base with additional functionalities that made the application more practical and user-friendly.

Adding these extra features required careful planning and extra effort, but it helped us deliver a more complete and valuable system. Seeing our initial concept grow into something even more functional than we first imagined has been really rewarding.

We believe the current set of features strikes a good balance between usefulness and maintainability, and it gives us a strong foundation to build on in the future.

6.4 Future work and potential improvements

Looking ahead, there are several ways this project can continue to grow and improve. First, maintaining the application will be an ongoing responsibility. Regular updates, bug fixes, and keeping dependencies up to date will help ensure the system remains secure, stable, and relevant for its users.

We also see plenty of room for improving our codebase itself. Refactoring parts of the code to make it cleaner, more efficient, and easier to maintain is a priority. As we gain more experience, we can apply better design patterns and best practices to keep the project professional and future-proof.

Another exciting possibility is implementing a dedicated mobile app version. This would make the application more accessible and convenient for users on the go, and would allow us to expand its reach and usability.

All these ideas show that this journey doesn’t stop here — it’s really a continuous process of learning, maintaining, and enhancing what we’ve built. We’re excited to keep pushing it forward and seeing how far we can take it.

References:

[1] Zaker, https://zaker.io/

[2] Prexams, https://www.prexams.com/

[3] CourseHero, https://www.coursehero.com/

[4] MySQL, https://www.mysql.com/

[5] Postgres SQL,https://www.postgresql.org/

[6] MongoDB, https://www.mongodb.com/

[7] ASP.Net Core, http://dotnet.microsoft.com/en-us/apps/aspnet

[8] Laravel, https://laravel.com/

[9] FastAPI, https://fastapi.tiangolo.com/

[10] React, https://react.dev/

[11] Angular, https://angular.dev/

[12] VUE Js, https://vuejs.org/

[13]OpenAI API, https://openai.com/api/

[14] Gemini API, https://console.cloud.google.com/