CourseMind

System Design Document

Yashodhan Rajesh Jaltare
Rucha Dilip Keni
Parshwa Shah
Foram Shah
Hitakshi Shirude

December 7, 2024

Contents

1 Project Overview

We are developing a web-based application to help professors manage their courses more efficiently by assisting them with course planning, quiz and assignment creation. The application will provide features to outline the course, auto-generate quizzes from the context and track project submissions, and manage deadlines in a centralized interface.

1.1 Introduction

Professors often spend a significant amount of time creating course outlines, assignments, quizzes, and projects. Managing these tasks manually can be inefficient, especially for professors handling multiple courses. The need for a tool that simplifies these tasks has never been greater, as it would allow educators to focus more on teaching and student interaction rather than administrative tasks. This project aims to build an application that assists professors in planning their courses, generating questions, and organizing small projects in a more streamlined manner. The tool will simplify the creation of quizzes, assignments, given the course outlines and materials, saving professors time while ensuring consistency and quality in their content delivery.

1.2 Context Diagram

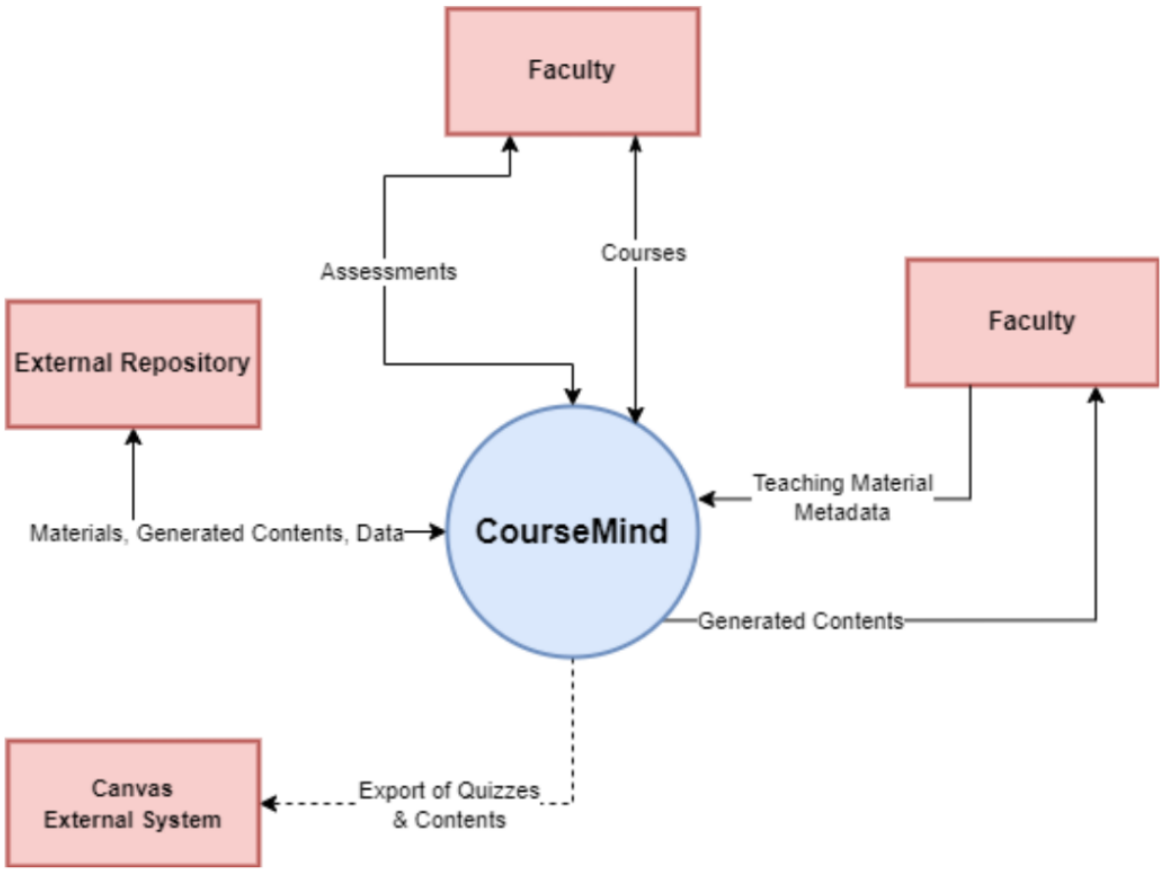

Figure 1: Context diagram highlighting scope of project and association of internal/external factors

1.3 Use case Diagram

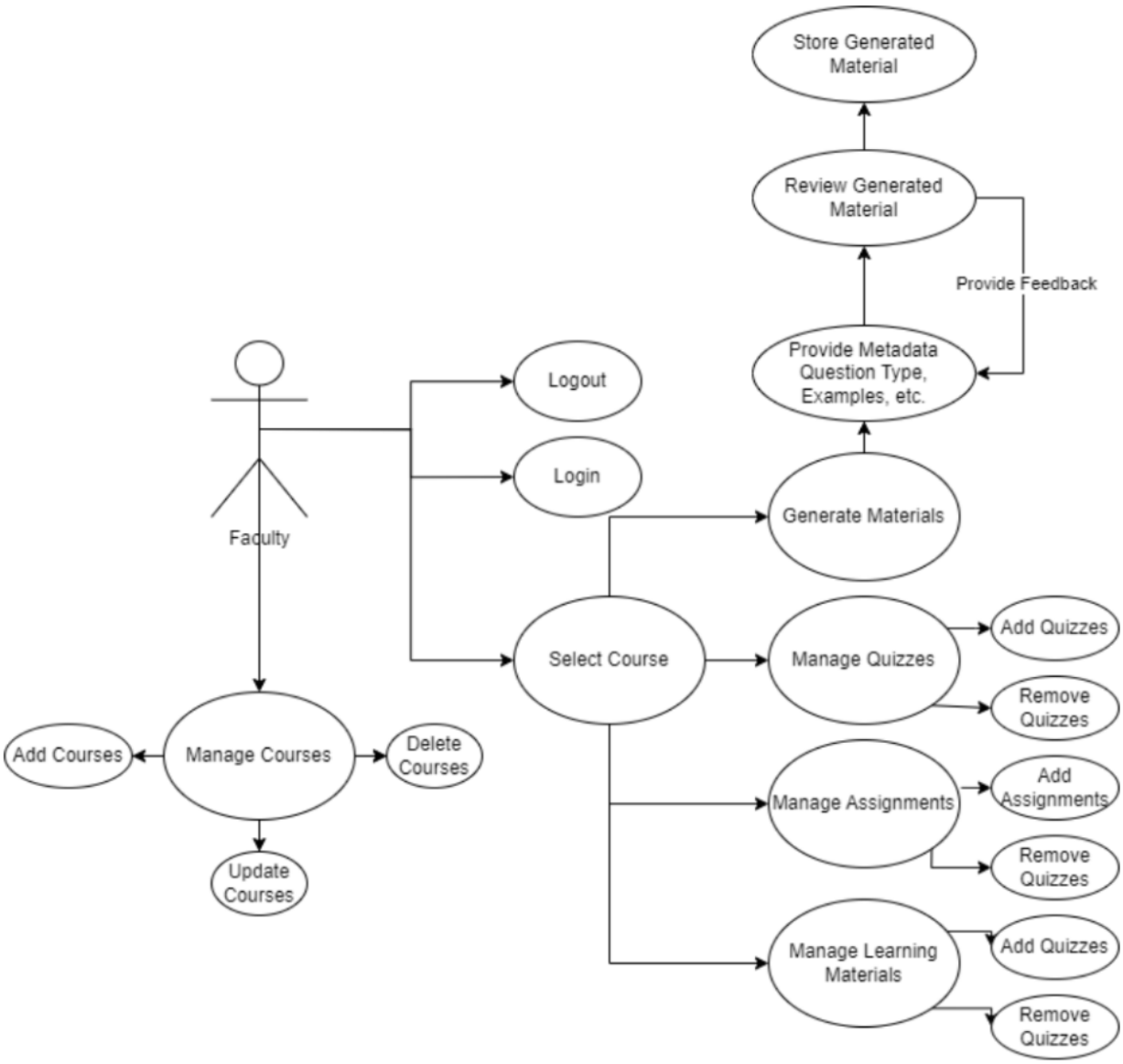

Figure 2: Use Case diagram describing various use cases associated with CourseMind from user's perspective

1.4 Project Scope

1. Develop a user-friendly interface for professors to create and manage their courses.
2. Build a question bank for quizzes and assignments with the ability for auto-generation.

3. Enable professors to add, update, and delete new course assignments and quizzes.
4. Allow course content and assignments to be exported to share with others.

2 Project Architecture

CourseMind architecture balances simplicity with flexibility. Clearly defined layers, secure data storage, and responsive control flow, the system performs its job of supporting professors in course management, assignments, and quizzes without technical complexities. The application will be highly accessible, reliable, and scalable in such a setting; it will be an easy and useful tool for educators.

2.1 Subsystem Architecture

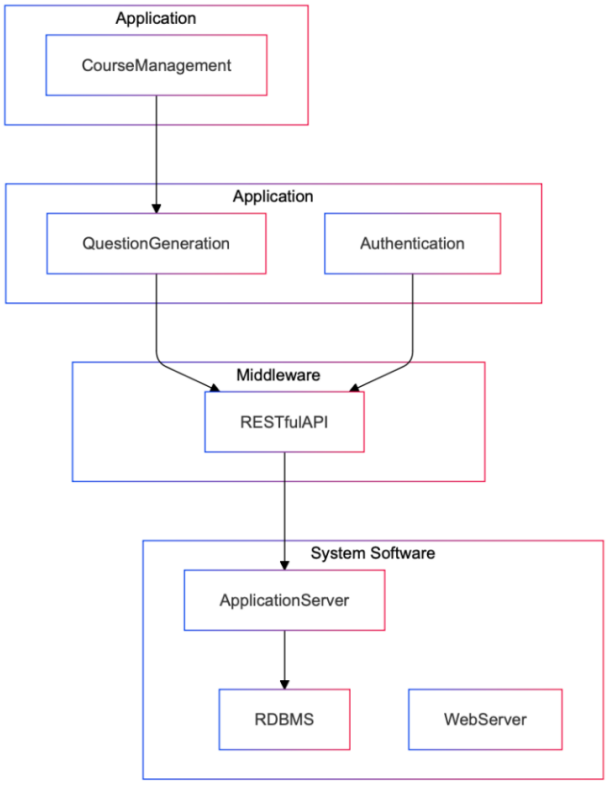

Figure 3: CourseMind system architecture overview and control flow

This section explains the system's layers and why they're organized this way, focusing on flexibility and ease of maintenance. The architecture followed is the multi tier architecture; it has divided the system into layers. This separation between presentation layer, business layer, and data storage makes the system very much scalable and maintainable. Yes, we did think over the monolithic design, but we saw that in such a design, scalability may be a

problem, and the system probably can't handle many users at once. For the architectural overview, we would typically include a subsystem package diagram.

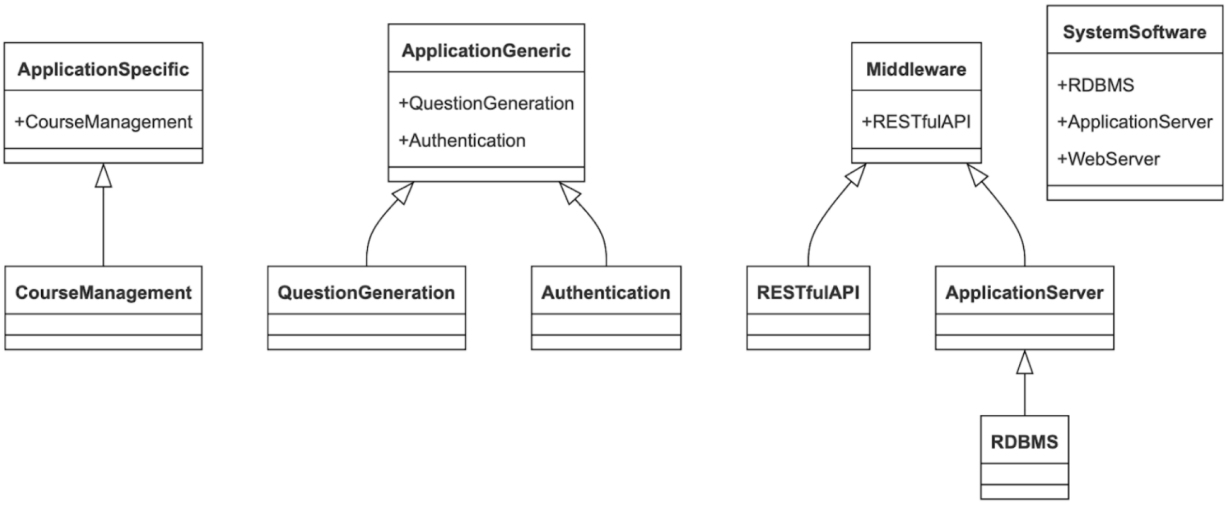

Figure 4: CourseMind system architecture overview and control flow

Layered Structure:

1. Application-Specific: Handles course management, quiz generation, and assignment modules, providing core functionalities for professors.
2. Application-Generic: Comprises reusable components like user interfaces and authentication services along with managing common features like question generation and authentication
3. Middleware: The REST API acts as the communication layer between the frontend and backend
4. System Software: This is the foundation, including the operating system and the web server where the backend operates

Reason for this Architecture: This layered approach keeps things organized and makes future updates easier. By separating layers, we can improve one layer (like the user interface) without affecting other layers.

2.2 Deployment Architecture

The deployment diagram maps out where each part of the "CourseMind" application will run and how they communicate. Basically, it shows how different parts of the system are physically distributed and communicate with each other. In this case, we have separated the front-end, back-end, and database layers to enhance scalability, maintainability, and security.

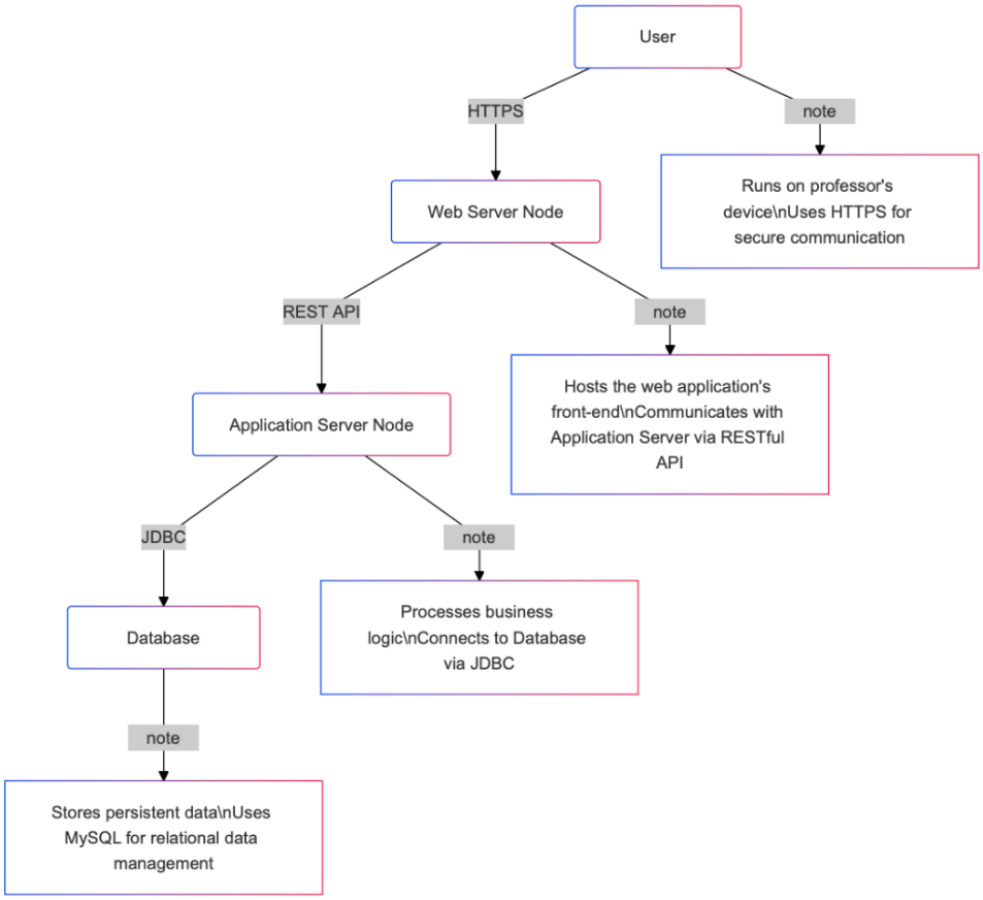

Figure 5: CourseMind system architecture back end and related flow

Detailed Explanation:

1. User Devices: This represents user devices (laptops, desktops, tablets, or smartphones) used by professors where the application front end will be accessed through web browsers and facilitating professors' interaction with the system over a secure HTTPS connection and accessibility without requiring installation.

2. Backend Server: This server hosts the core functionality that is UI components, handling authentication, course management, and quiz/ assignment generation.

3. WebServerNode: Contains the main logic for course management and content generation. The server can handle multiple requests simultaneously, enabling professors to create and manage course content concurrently.

4. Central Database: Stores course content, quizzes, assignments, and user credentials. Communication between the back end and database occurs via JDBC or SQL protocols for reliable data transactions.

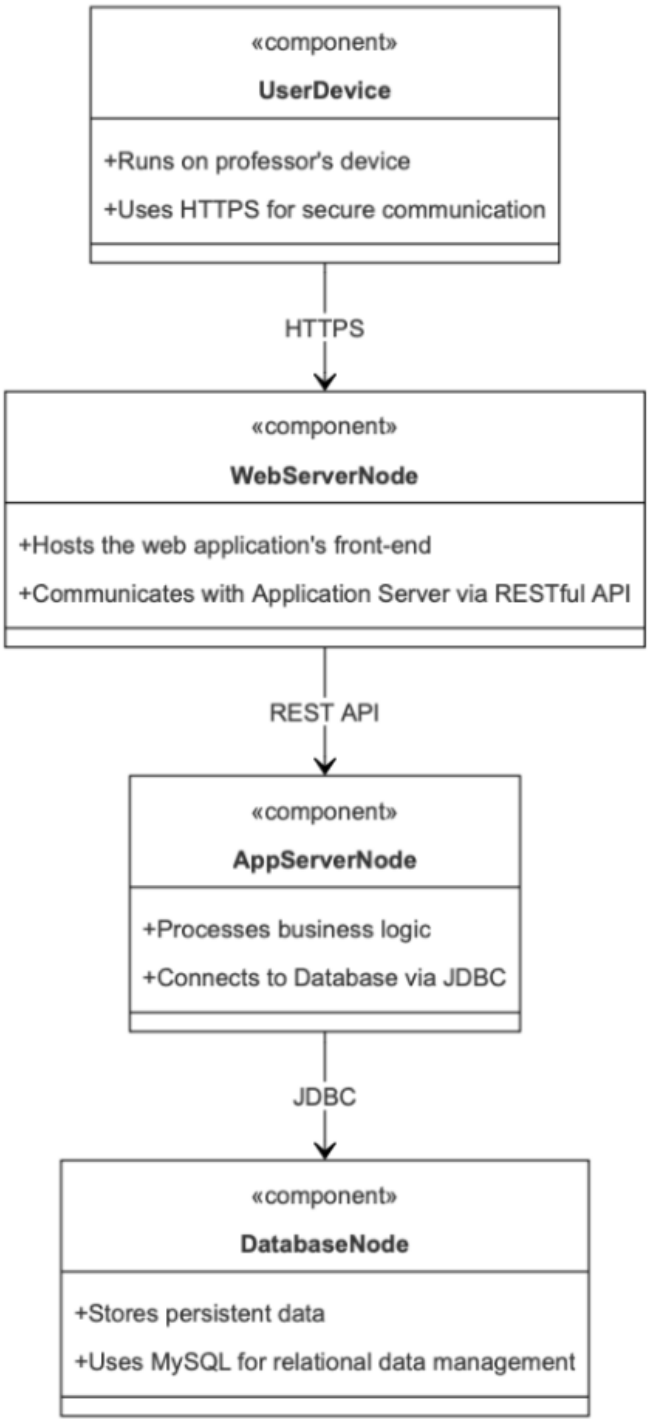

Figure 6: System backend API structure

Communication:
The front end (user's device) and back end server communicate via HTTP/HTTPS

for security. The back end uses SQL or similar database protocols to interact with the database.

2.3 Persistent Data Storage

The persistent data storage diagram shows the database schema, outlining how different pieces of data are stored and related. The relational structure ensures data integrity and allows complex querying. Since the application requires persistent storage for course information, assignments, and user details, we use a relational database.

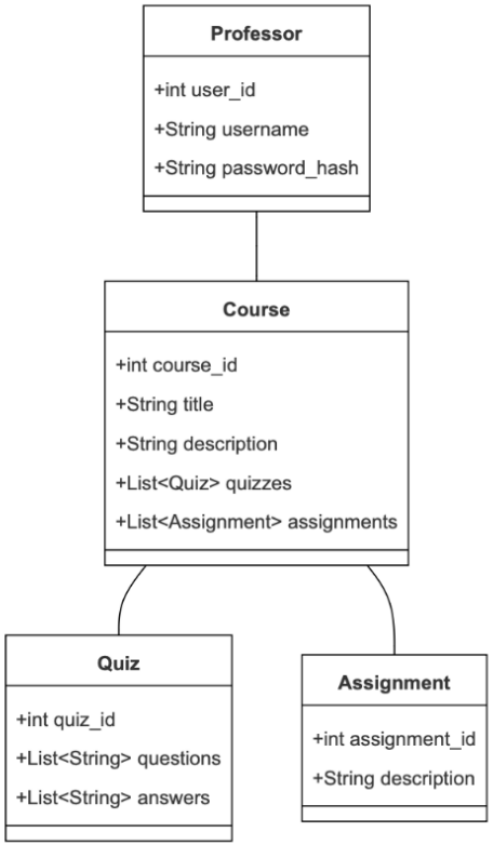

Figure 7: Document database models and schemas

Database Schema Explanation:

1. COURSE: Stores course details, including title, description, and schedule. Each course can have multiple quizzes and assignments.
2. QUIZ and ASSIGNMENT: Stores quiz and assignment data, each linked to a course.
3. USER: Maintains user credentials and roles. Each user has unique id and role based access is implemented to ensure only professors can manage course data.

Reason for Relational Database: A relational database structure is used here because it allows organized data storage and quick retrieval while handling relationships (like linking courses to their quizzes and assignments).

2.4 Global Control Flow

The global control flow defines how the application's functions are triggered and handled, based on user interactions and asynchronous processing The system's execution is primarily event-driven. Here's a breakdown based on control flow characteristics:

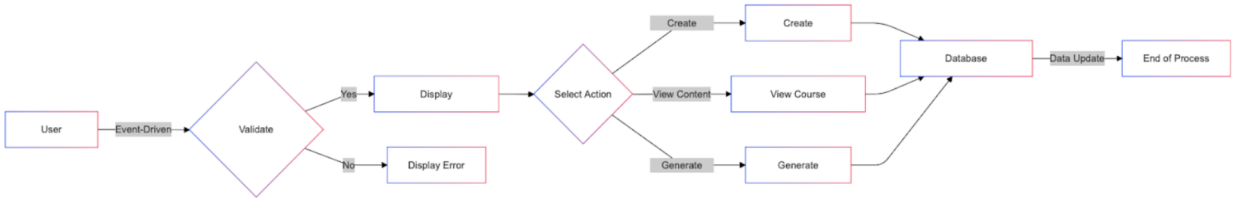

Figure 8: Global control flow diagram

1. Event-Driven: The system responds to user actions, like when a professor creates a course or generates a quiz and each action triggers a backend process.
2. Time Dependency: The system operates asynchronously without strict time constraints, allowing professors to schedule tasks but not relying on real-time operations.
3. Concurrency: While primary operations run on the server, user interactions trigger asynchronous events. Components like the quiz generator may use threads to handle processing without blocking the main flow.

Why This Flow: The event-driven approach gives professors freedom in how they interact with the app, while concurrency ensures smooth operation, even with multiple users.

3 System Design Details

This section includes both the static and dynamic views, describing the structure of classes, their attributes, and the interactions needed for core functions.

3.1 Static View

Each class (User, Course, Quiz, Assignment) has specific tasks:

1. User Class: Manages user credentials and sessions. The login() and logout() methods handle user authentication, ensuring secure access to the application.

2. Course Class: Represents a course, with methods for creation, updates, and deletion. It allows professors to add new courses or modify existing ones.

3. Quiz Class: Contains details of the quizzes, including the questions. The generateQuiz() method automates quiz creation, leveraging course material to populate questions.

4. Assignment Class: Represents assignments within a course. It has methods for creation and submission, supporting the management of assignments within the course context.

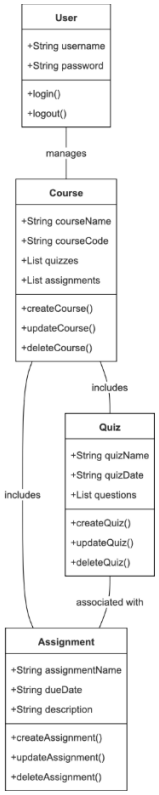

Figure 9: Global control flow diagram

3.2 Dynamic View

It shows the interactions for the quiz generation process, mapping out the sequential flow and responsibilities of each component.

1. Professor Initiates Quiz Creation: The professor uses the front-end interface to request quiz creation for a specific course.

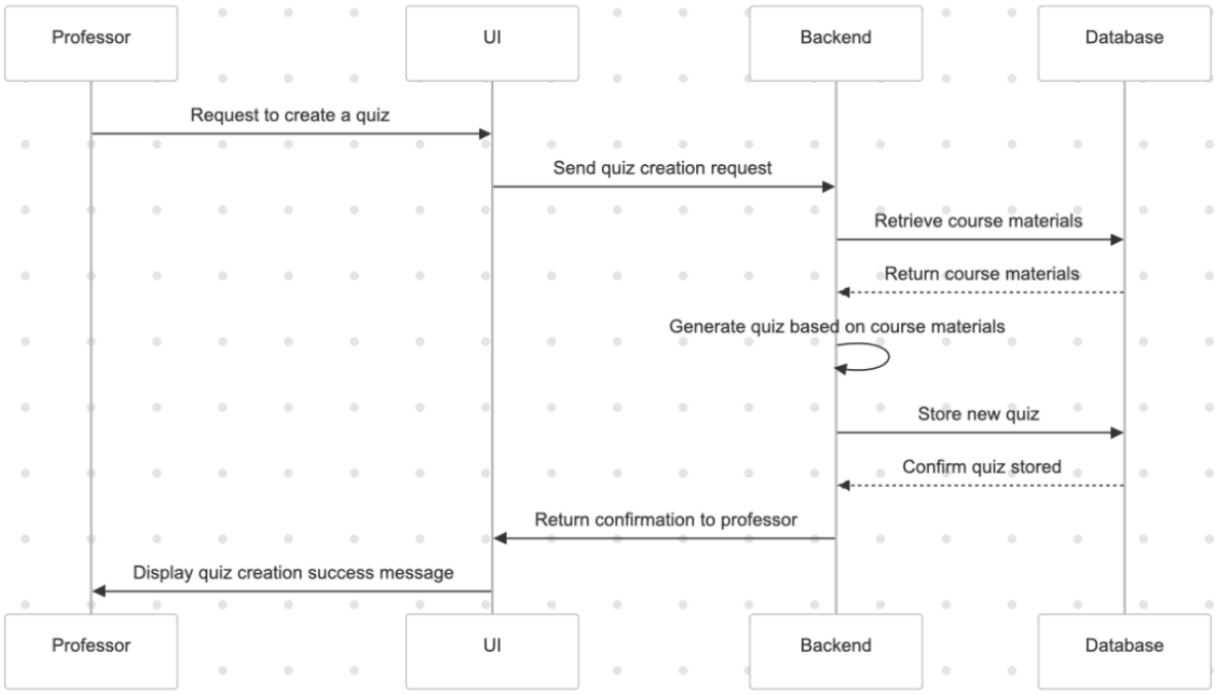

Figure 10: Global control flow diagram

2. Front End Sends Request to Backend: The front end sends an HTTP request to the backend, containing relevant course details.

3. Backend Fetches Course Data: The backend queries the database to retrieve course content that will serve as the basis for quiz questions.

4. Quiz Generation and Storage: Using the retrieved course content, the backend generates quiz questions, then stores the new quiz in the database.

5. Confirmation: Finally, the backend sends a success message to the front end, confirming that the quiz was created. This feedback is displayed to the professor

Purpose of this Design: The class structure keeps functions separate and organized. The sequence flow shows how each component contributes to tasks, making the process efficient and easy to follow.

A Appendix

A.1 Tasks

1. Authentication - UI
2. Dashboard with list of courses - UI
3. Inner page for list of assignments and quizzes for each course - UI
4. Create a new course - UI
5. Add new materials for course - UI
6. Create new quizzes - UI
7. Create new assignment - UI
8. Create database schema - backend (structuring)
9. Authentication - backend API
10. List of courses - backend API
11. CRUD for courses - backend API
12. CRUD for materials for course - backend API
13. CRUD for quizzes - backend API
14. CRUD for assignment - backend API
15. Get a model to create questions from given context - AI Model (POC)
16. Convert document to data chunks - AI Model
17. Use data chunks to find key information in chunks - AI Model
18. Use key information and chunks to create answers - AI Model
19. Get a model to generate assignments or short projects problems for course - AI Model
20. Integration of Authentication UI & API
21. Integration of Courses CRUD UI & API
22. Integration of Quizzes CRUD UI & API
23. Integration of Assignments CRUD UI & API
24. Integration of AI Model to feed new materials
25. Integration of AI Model to generate new things

A.2 Use Cases

USE CASE NAME:	Generating Examples from teaching materials	**USE CASE TYPE**
USE CASE ID:	1	Business Requirements: □
PRIORITY:	1	System Analysis: □
SOURCE:	Teacher	System Design: ☑
PRIMARY BUSINESS ACTOR	Teacher who wants to teach some topic to students	
PRIMARY SYSTEM ACTOR	CourseMind System (working on generating examples)	
OTHER PARTICIPATING ACTORS:	Generative AI Model (part of CourseMind)	
OTHER INTERESTED STAKEHOLDERS:		
DESCRIPTION:	Focuses on generating examples to teach students from given teaching materials	
PRE-CONDITION:	Teacher has uploaded some teaching materials for the selected course from which Teacher wants CourseMind to create e	
TRIGGER:	Teacher wants to teach something and seeks to generate examples	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: Teacher navigates to selected course	**Step 4**: Gives examples for faculty to filt
	Step 2: Selects uploaded material from which teacher wants to create examples	
	Step 3: Gives some details about what type of examples teacher wants	
	Step 5: Selects and exports these examples in JSON	
ALTERNATE COURSES:	If Teacher doesn't like the examples generated by CourseMind, teacher can provide feedback to CourseMind which allow	
	If user CourseMind doesn't get any good examples it would return with error message	
CONCLUSION:	Teacher will have examples for particular topic ready to help students with studying	
POST-CONDITION:	Teachers will have examples generated and stored into a database as well as JSON of all examples exported.	
BUSINESS RULES	Teacher should be logged in	
	Should have uploaded some reference material	
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	Only authorized users can access	
	It creates examples on 1 topic at a time	
ASSUMPTIONS:	Teachers will provide high quality teaching material to create examples from	
OPEN ISSUES:		

USE CASE NAME:	Generating Question of different types from teaching materials	**USE CASE TYPE**
USE CASE ID:	2	Business Requirements: □
PRIORITY:	1	System Analysis: □
SOURCE:	Teacher	System Design: ☑
PRIMARY BUSINESS ACTOR	Teacher who wants to assess student's knowledge on some topic	
PRIMARY SYSTEM ACTOR	CourseMind System (working on generating examples)	
OTHER PARTICIPATING ACTORS:	Generative AI Model (part of CourseMind)	
OTHER INTERESTED STAKEHOLDERS:		
DESCRIPTION:	This feature allows teachers to generate questions from a given teaching materials so that they can assess students' performance	
PRE-CONDITION:	Teacher has uploaded some teaching materials for the selected course from which Teacher wants CourseMind to create questions	
TRIGGER:	Teacher wants to generate quiz questions to assess students	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: Teacher navigates to selected course	**Step 4**: CourseMind will give question t
	Step 2: Selects uploaded material from which the teacher wants to create questions from.	
	Step 3: Gives type of question and number of questions to be generated.	
	Step 5: Selects and exports the complete quiz in QTI format	
ALTERNATE COURSES:	If Teacher doesn't like the questions generated by CourseMind, teacher can provide feedback to CourseMind which allows teacher t	
	If user CourseMind doesn't get the required number of questions it will give however many it was able to generate and add an error	
CONCLUSION:	Teacher will have a quiz which can be used to assess students' knowledge on certain topic	
POST-CONDITION:	Teacher will have questions generated and stored into a database as well as exported to Canvas for assessment.	
BUSINESS RULES	Teacher should be logged in	
	Should have uploaded some reference material	
	Should provide valid type of question	
IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS	Only authorized users can access	
ASSUMPTIONS:	Teachers will provide high quality teaching material to create questions from	
OPEN ISSUES:		

USE CASE NAME:	Generating Assignments for courses from teaching materials		USE CASE TYPE
USE CASE ID:	3		Business Requirements: □
PRIORITY:	1		System Analysis: □
SOURCE:	Teacher		System Design: ☑
PRIMARY BUSINESS ACTOR	Teacher who wants to give students personalized assignment for particular topic		
PRIMARY SYSTEM ACTOR	CourseMind System (working on generating examples)		
OTHER PARTICIPATING ACTORS:	Generative AI Model (part of CourseMind)		
OTHER INTERESTED STAKEHOLDERS:			
DESCRIPTION:	This feature allows teachers to generate different assignments from a given teaching materials so that they can assess students' per		
PRE-CONDITION:	Teacher should have some topic and content to create short personalized assignments for students		
TRIGGER:	Teacher wants to generate quiz questions to assess students		
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**	
	Step 1: Teacher navigates to selected course	**Step 4**: Generates assignments based o	
	Step 2: Selects uploaded material from which the teacher wants to create questions from.		
	Step 3: Gives the idea and sample of assignment and number of assignments required		
	Step 5: Selects and exports the JSON of created assignments		
ALTERNATE COURSES:	If Teacher doesn't like the assignments generated by CourseMind, teacher can provide feedback to CourseMind which allows teach		
	If user CourseMind doesn't get the required number of assignments it will give however many it was able to generate and add an err		
CONCLUSION:	Teacher will get examples generated for particular topic ready to help students with studying using existing teaching materials		
POST-CONDITION:	Teachers will have questions generated and stored into a database as well as exported to Canvas for assessment.		
BUSINESS RULES	Teacher should be logged in		
	Should have uploaded some reference material		
	Should provide valid type of question		
IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS	Only authorized users can access		
ASSUMPTIONS:	Teachers will provide high quality teaching material to create assignments from		
OPEN ISSUES:			

USE CASE NAME:	Authentication		USE CASE TYPE
USE CASE ID:	4		Business Requirements: □
PRIORITY:	1		System Analysis: □
SOURCE:	Teacher		System Design: ☑
PRIMARY BUSINESS ACTOR	The person uploading and managing course content.		
PRIMARY SYSTEM ACTOR	The system where the materials are stored and managed.		
OTHER PARTICIPATING ACTORS:	Teaching Assistants and Canvas		
OTHER INTERESTED STAKEHOLDERS:	None		
DESCRIPTION:	This feature allows teacher to login and keep their data separated from others so that it doesn't get mixed up		
PRE-CONDITION:	Teacher has internet access and a compatible device		
TRIGGER:	To create an account or log into existing one		
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**	
	Step 1: Teacher navigates to login portal	**Step 2**: System validates the credentials	
	Types in his credentials in the system or register with all the details	Saves their account details and allows	
	Teacher can logout of their account	Clears the cookies and authentication k	
ALTERNATE COURSES:	If the Teacher email does not exist, it will show the SignUp page.		
	If CourseMind can not validate the credentials it will return an error saying Invalid Credentials.		
CONCLUSION:	Atlast, the teacher will be able to perform CRUD requests for their courses		
POST-CONDITION:	If the credentials were correct then the teacher would be redirected to the dashboard with their data.		
BUSINESS RULES	Teacher must have an account or sign up with all the details		
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	All the details provided by the user, mainly emailId, must be correct.		
ASSUMPTIONS:	Only users logged in/ registered can be authenticated		
OPEN ISSUES:	None		

USE CASE NAME:	Manage all courses	
USE CASE ID:	5	
PRIORITY:	1	
SOURCE:	Teacher	
PRIMARY BUSINESS ACTOR	Teacher who wants to manage all his courses	
PRIMARY SYSTEM ACTOR	CourseMind (The backend system for managing all the user courses and CRUD requests)	
OTHER PARTICIPATING ACTORS:	Teaching Assistants and Canvas	
OTHER INTERESTED STAKEHOLDERS:	None	
DESCRIPTION:	This feature allows teachers to create, update, read and delete all their courses.	
PRE-CONDITION:	Teacher has internet access and a compatible device and is authenticated.	
TRIGGER:	To manage their courses in one page	
TYPICAL COURSE OF EVENTS:	**Actor Action**	
	Step 1: Teacher sees the dashboard and their added courses(if any)	
	Adds new course name, code, description and start and end date and save	
	Teachers can edit any listed course, make changes and save.	
	Teacher can delete any course and all his materials. A confirmation dropbox will popup for delete.	
ALTERNATE COURSES:	If Teacher is not authenticated, it will show the LogIn/SignUp page.	
	If the added course details are of incorrect type or missing, it will return an error message and won't add it to the dataset.	
	If CourseMind faces any error in deleting the course or course material from the database, it will return with error message	
CONCLUSION:	Atlast, the teacher will be able to perform CRUD requests for their courses	
POST-CONDITION:	Teacher will have all the courses added, updated and listed on the dashboard.	
BUSINESS RULES	User/Teacher/Professor must be logged in	
	Course that is being added should have all required information	
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	Only authorized users are granted read permission	
ASSUMPTIONS:	Only users seeing this page must be authorized	
OPEN ISSUES:	None	

USE CASE NAME:	Selecting course from a list of courses I am teaching and taught in the past	**USE CASE TYPE**
USE CASE ID:	6	Business Requirements: □
PRIORITY:	1	System Analysis: □
SOURCE:	Teacher/Professor	System Design: ☑
PRIMARY BUSINESS ACTOR	Teacher/Professor (who wants to enter into an individual course from portal)	
PRIMARY SYSTEM ACTOR	CourseMind (The backend system)	
OTHER PARTICIPATING ACTORS:		
OTHER INTERESTED STAKEHOLDERS:		
DESCRIPTION:	Focuses on selecting a course from existing courses list for opening course material	
PRE-CONDITION:	User has internet access and a compatible device and has at least one course added in the database	
TRIGGER:	User wants to enter into an individual course webpage for further work	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: User navigates to the Courses page	**Step 3**: Upon successful completion of
	Step 2: User clicks on "Open" button and enters into a course by filling	
ALTERNATE COURSES:	If error occurs during the course select process, the error message will be displayed.	
CONCLUSION:		
POST-CONDITION:	User will view individual course webpage, displaying course notes, assignments and quizzes	
BUSINESS RULES	User/Teacher/Professor must be logged in	
	There should be at least one course added in the system	
IMPLEMENTATION CONTRAINTS AND SPECIFICATIONS	Only authorized users are granted read permission	
ASSUMPTIONS:	Only authorized users are granted read permission	
OPEN ISSUES:	Courses are already present in the database to begin with	

USE CASE NAME:	Managing Assignments for courses from teaching materials	**USE CASE TYPE**
USE CASE ID:	7	Business Requirements: □
PRIORITY:	1	System Analysis: □
SOURCE:	Teacher	System Design: ☑

PRIMARY BUSINESS ACTOR	Teacher who is responsible for managing course assignments	
PRIMARY SYSTEM ACTOR	CourseMind System (helps teachers to manage course assignments)	
OTHER PARTICIPATING ACTORS:	Generative AI Model (part of CourseMind)	
OTHER INTERESTED STAKEHOLDERS:		
DESCRIPTION:	This feature focuses on how teachers manage the course assignments for a particular course including the ability to add, remove and review assignments.	
PRE-CONDITION:	Teacher should have successfully logged into the system and has selected a course for which they want to manage assignments.	
TRIGGER:	Teacher wants to manage assignments for students	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: Teacher navigates to selected course's dashboard	ii) Remove Assignment: Teacher selects an existing assignment The system asks for confirmation befor
	Step 2: Selects "Manage Assignments" option and the system displays all the existing assignments for selected course	iii) Review Assignment: Teacher selects an existing assignment The system displays the details of the s
	Step 3: Teacher can choose to do	
	i) Add Assignments: Teacher selects Add Assignment The system prompts for assignment details (title, due date, description, etc)	
ALTERNATE FLOW:	If Teacher attempts to add an assignment without providing all required information (e.g., missing title or due date), the system will prompt them to complete the mis	
	If Teacher tries to remove an assignment that has active student submissions, the CourseMind system will prompt them with a warning that removal will also delete	
CONCLUSION:	Teacher will be able to add, delete and update assignments to keep the course materials up to date and relevant	
POST-CONDITION:	If the assignment is added or removed successfully, the course assignments list is updated accordingly. If there is an issue with adding or removing assignments (e.g., network error), an error message is displayed to the faculty member.	
BUSINESS RULES	Teacher should be logged in Should have assignments to be managed	
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	Only authorized users can access	
ASSUMPTIONS:	Teachers will perform high quality management to manage assignments	
OPEN ISSUES:		

USE CASE NAME:	Managing Quizzes for courses from teaching materials	**USE CASE TYPE**
USE CASE ID:	8	Business Requirements: □
PRIORITY:	1	System Analysis: □
SOURCE:	Teacher	System Design: ☑
PRIMARY BUSINESS ACTOR	Teacher who is responsible for managing course quizzes	
PRIMARY SYSTEM ACTOR	CourseMind System (helps teachers to manage course quizzes)	
OTHER PARTICIPATING ACTORS:	Generative AI Model (part of CourseMind)	
OTHER INTERESTED STAKEHOLDERS:		
DESCRIPTION:	This feature focuses on how teachers manage the course quizzes for a particular course including the ability to add, remove and review assignments.	
PRE-CONDITION:	Teacher should have successfully logged into the system and has selected a course for which they want to manage quizzes.	
TRIGGER:	Teacher wants to manage assignments for students	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: Teacher navigates to selected course's dashboard	ii) Remove Quizzes: Teacher selects an existing quiz to remo The system asks for confirmation befor
	Step 2: Selects "Manage Assignments" option and the system displays all the existing assignments for selected course	iii) Review Quiz: Teacher selects an existing quiz to revie The system displays the details of the
	Step 3: Teacher can choose to do	
	i) Add Assignments: Teacher selects Add Assignment The system prompts for assignment details (title, due date, description, etc)	
ALTERNATE FLOW:	If Teacher attempts to add a quiz without providing all required information (e.g., missing title or due date), the system will prompt them to complete the missing field	
	If Teacher tries to remove a quiz that has active student submissions, the CourseMind system will prompt them with a warning that removal will also delete the subm	
CONCLUSION:	Teacher will be able to add, delete and update quizzes to keep the course materials up to date and relevant	
POST-CONDITION:	If the quiz is added or removed successfully, the course quiz list is updated accordingly. If there is an issue with adding or removing quiz (e.g., network error), an error message is displayed to the faculty member.	
BUSINESS RULES	Teacher should be logged in Should have quizzes to be managed	
IMPLEMENTATION CONSTRAINTS AND SPECIFICATIONS	Only authorized users can access	
ASSUMPTIONS:	Teachers will perform high quality management to manage quizzes	
OPEN ISSUES:		