1.1.1 Functional Requirements

The following subsections will outline in detail the functional requirements of the system. Each set of requirements is categorized based on the user roles and functionalities:

1. **Guest User:** The first part will list the functional requirements related to guest users.
2. **Registered User:** Next, the focus will shift to the functional requirements for registered users.
3. **Instructor (Registered User):** This section will outline the functional requirements for instructors.
4. **Student (Registered User):** Finally, the functional requirements for students will be detailed.

1.1.1.1 Guest User Functional Requirements

FR1	The system shall allow a guest user to register by providing the required information: first name, last name, email address, role, and password.

1.1.1.2 Registered User Functional Requirements

FR1	The system shall allow a registered user to log in to their account.
FR2	The system shall allow a logged-in registered user to log out of their session.

1.1.1.3 Instructor (Registered User) Functional Requirements

FR1	The system shall provide a courses page where an instructor can view their created courses.
FR2	The system shall allow an instructor to create a course.
FR3	The system shall allow an instructor to edit any of their courses.
FR4	The system shall allow an instructor to delete any of their courses.
FR5	The system shall allow an instructor to view detailed information about their course.
FR6	The system shall allow an instructor to edit any of their courses.
FR7	The system shall allow an instructor to remove a student from any of their courses.
FR8	The system shall allow an instructor to add students during the course creation process or while editing any of their existing courses by importing a file containing the students' IDs.
FR9	The system shall allow an instructor to add students during the course creation process or while editing any of their existing courses by manually adding the students' IDs.
FR10	The system shall allow an instructor to remove previously uploaded files from a course.
FR11	The system shall allow an instructor to upload a file during the course creation process or while editing any of their existing courses.
FR12	The system shall allow an instructor to interact with the content of the uploaded file in a course through a chat interface.
FR13	The system shall allow an instructor to download chat responses as files.

1.1.1.4 Student (Registered User) Functional Requirements

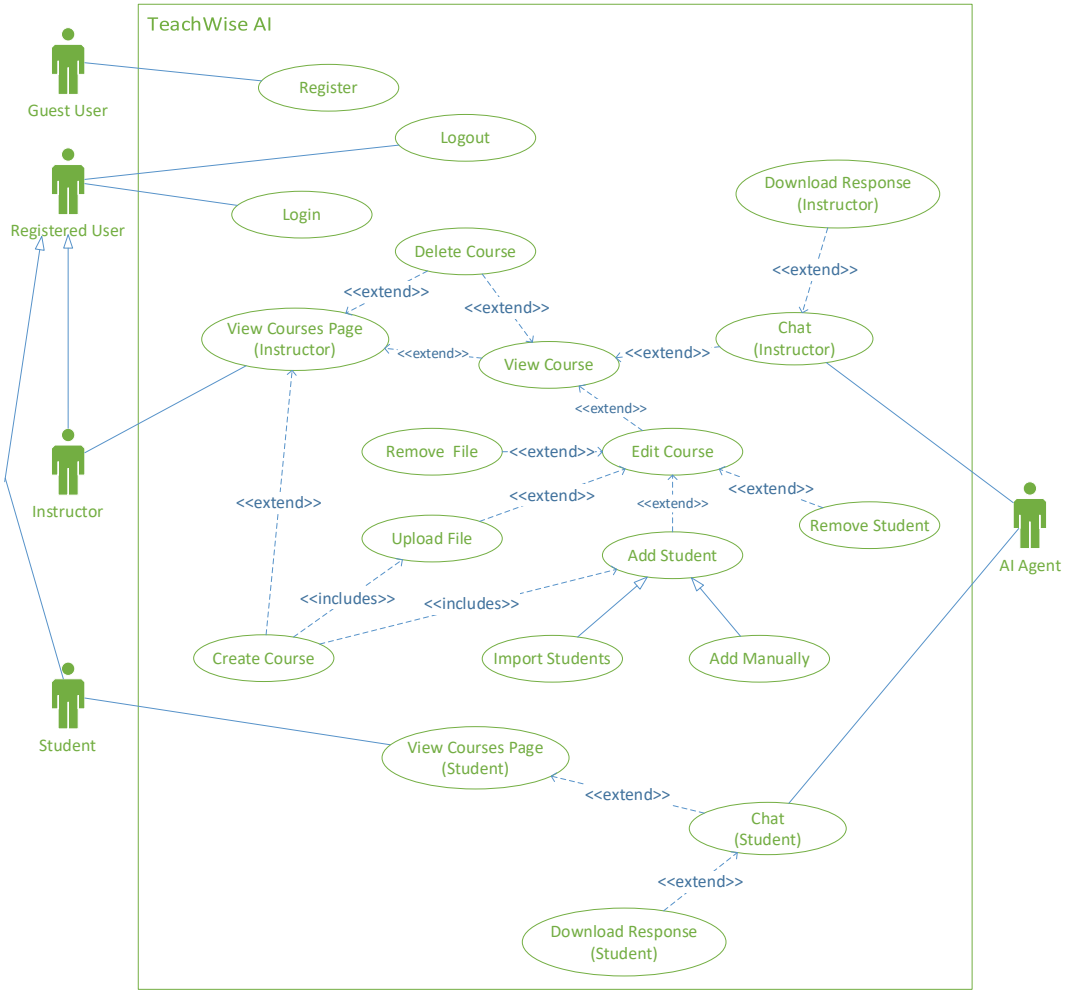
FR1	The system shall provide a courses page where a student can view their enrolled courses.
FR2	The system shall allow a student to interact with an uploaded file content in an enrolled course through a chat interface.
FR3	The system shall allow a student to download chat responses as files.

1.1.2 Non-Functional Requirements

The non-functional requirements are the constraints or limitations on the services and functionalities provided by the system [17]. In other words, they are the characteristics or qualities of a system rather than the provided services [16].

NFR1 Privacy	The system must protect the privacy of user's personal information.
NFR2 Correctness	The system must ensure that the generated responses during the chat interactions with course files are accurate and reflect correct information from the course materials.

1.2 Use Case Diagram



1.3 Use Case Description Tables

The use case description table has the following fields:

- **Use case name:** Title of the represented use case.
- **Use case description:** A brief description of the represented use case.
- **Inputs:** Data required to perform the use-case (actor inputs and system inputs).
- **Pre-conditions:** The conditions that must be true before the use case can be executed.
- **Sources:** Represents the entity (actor or system) that initiates the interaction.
- **Destination:** Represents the entity (actor or system) that receives the interaction sent by the source.
- **Steps:** The main flow of events or steps that occur during the execution to accomplish the purpose of the use case.
- **Outputs:** Represents the result or the response that will be displayed after executing the use case.

- **Post-conditions:** The conditions that are expected to be true after the successful completion and execution of the use case.
- **Exceptions & Alternative Flows:** Exceptional events that occur, preventing the successful flow from executing or alternative flow of events that occur during the execution of the use case.

1.3.1 Register

Use-Case Description	
Use-Case Name	Register
Use-Case Description	Enables individuals to create an account on the TeachWise AI platform.
Inputs	- First Name - Last Name - Email Address - Role - Password
Pre-Conditions	No existing account with the provided email address.
Source	Guest User
Steps	1. This use case starts when the guest user navigates to the registration page. 2. The system presents the registration form. 3. The guest user submits the form with their credentials. 4. The system sends a verification link to the guest user. 5. The guest user verifies their account through the link. 6. The system redirects the user to the home page.
Outputs	None
Post-Conditions	If the basic flow steps are successful, then the home page is displayed, and the guest user is registered in the platform.
Destination	TeachWise AI System
Exception & Alternative flows	In step 3, if the guest user entered an/a: - Invalid input formats: Display an error message. - Email domain that doesn't match the allowed universities domains: Display an error message. - Student email contains non-numeric characters: Display an error message. - Account that already exists: Display an error message.

1.3.2 Login

Use-Case Description	
Use-Case Name	Login
Use-Case Description	Enables registered users to access their accounts.
Inputs	- Email - Password
Pre-Conditions	- The user is registered on the platform. - The registered user isn't already logged in.
Source	Registered User (Instructor, Student)
Steps	1. This use case starts when the registered user navigates to the login page. 2. The system presents the login form. 3. The registered user submits the form with their credentials. 4. The system redirects the registered user to the home page.
Outputs	None
Post-Conditions	If the basic flow steps are successful, then the registered user is logged in and the home page is displayed.
Destination	TeachWise AI System
Exception & Alternative flows	In step 3, if the registered user submits invalid credentials: Display an error message.

1.3.3 Logout

Use-Case Description	
Use-Case Name	Logout
Use-Case Description	Enables registered users to log out of their current session.
Inputs	None
Pre-Conditions	The registered user is logged in.
Source	Registered User (Instructor, Student)
Steps	1. This use case starts when the registered user selects the logout option. 2. The system logs out the registered user from their account.
Outputs	None
Post-Conditions	If the basic flow steps are successful, then the registered user no longer has access to their account until logged in and is redirected to the login page.
Destination	TeachWise AI System
Exception & Alternative flows	None

1.3.4 View Courses Page (Instructor)

Use-Case Description	
Use-Case Name	View Courses Page
Use-Case Description	Enables instructors to view their created courses.
Inputs	None
Pre-Conditions	The user is authenticated and has the instructor role.
Source	Instructor
Steps	1. This use case starts when the instructor navigates to the courses page. 2. The system loads their created courses. 3. If the instructor wants to view a course, then go to View Course use case. 4. If the instructor wants to create a course, then go to Create Course use case. 5. If the instructor wants to delete a course, then go to Delete Course use case.
Outputs	Created courses
Post-Conditions	If the basic flow steps are successful, then the instructor should be able to view their created courses.
Destination	TeachWise AI System
Exception & Alternative flows	In step 2, if there are no created courses: Display a message.

1.3.5 View Course

Use-Case Description	
Use-Case Name	View Course
Use-Case Description	Enables instructors to view detailed information about their created course.
Inputs	None
Pre-Conditions	- The user is authenticated and has the instructor role. - Course is available.
Source	Instructor
Steps	1. This use case starts when the instructor selects the desired course. 2. The system loads the course details. 3. If the instructor wants to delete the course, then go to Delete Course use case. 4. If the instructor wants to edit the course, then go to Edit Course use case. 5. If the instructor wants to chat, then go to Chat (Instructor) use case.
Outputs	Course details
Post-Conditions	If the basic flow steps are successful, then the instructor should be able to view the selected course details.
Destination	TeachWise AI System
Exception & Alternative flows	In step 1, if the selected course isn't available: Redirect the instructor to the courses page.

1.3.6 Create Course

Use-Case Description	
Use-Case Name	Create Course
Use-Case Description	Enables instructors to create a course.
Inputs	- Course Title - Course Description
Pre-Conditions	The user is authenticated and has the instructor role.
Source	Instructor
Steps	1. This use case starts when the instructor selects the create course option. 2. The system displays the form. 3. The instructor fills in the necessary information. 4. Execute Upload File use case. 5. Execute Add Student use case. 6. The instructor submits the form. 7. The system updates the view.
Outputs	Course Created
Post-Conditions	If the basic flow steps are successful, then the course is created and can be viewed by the instructor.
Destination	TeachWise AI System
Exception & Alternative flows	In step 3, if the instructor entered invalid course attributes input formats: Display an error message. In step 6, if the instructor didn't add students or didn't upload course material : Display an error message.

1.3.7 Delete Course

Use-Case Description	
Use-Case Name	Delete Course
Use-Case Description	Enables instructors to delete their created courses.
Inputs	None
Pre-Conditions	- The user is authenticated and has the instructor role. - Course is available. - Course is created by this instructor.
Source	Instructor
Steps	1. This use case starts when the instructor selects the delete course option at the desired course. 2. The system updates the view.
Outputs	Course deleted
Post-Conditions	If the basic flow steps are successful, then the desired course is deleted and can no longer be accessed by students nor the instructor.
Destination	TeachWise AI System
Exception & Alternative flows	None

1.3.8 Edit Course

Use-Case Description	
Use-Case Name	Edit Course
Use-Case Description	Enables instructors to view create a course.
Inputs	- Course Title - Course Description
Pre-Conditions	- The user is authenticated and has the instructor role. - Course is available.
Source	Instructor
Steps	1. This use case starts when the instructor selects the edit course option. 2. The system displays the form. 3. The instructor fills the necessary information. 4. if the instructor wants to upload files, then go to Upload File use case. 5. if the instructor wants to remove files, then go to Remove File use case. 6. If the instructor wants to add students, then go to Add Student use case. 7. If the instructor wants to remove students, then go to Remove Student use case. 8. The instructor submits the form. 9. The system updates the view.
Outputs	Course updated
Post-Conditions	If the basic flow steps are successful, then the course is updated, and changes can be viewed by the instructor.
Destination	TeachWise AI System
Exception & Alternative flows	In step 3, if the instructor entered invalid course attributes input formats: Display an error message.

1.3.9 Upload File

Use-Case Description	
Use-Case Name	Upload File
Use-Case Description	Enables instructors to upload files to their courses.
Inputs	One or multiple files
Pre-Conditions	- The user is authenticated and has the instructor role. - The instructor is in the process of creating a new course or editing an existing course.
Source	Instructor
Steps	1. This use case starts when the instructor selects the upload file option. 2. The system prompts the instructor to add their desired files. 3. The instructor adds the desired files. 4. The system updates the view.
Outputs	Uploaded files
Post-Conditions	If the basic flow steps are successful, then the uploaded files are added to the course and can be viewed by the instructor.
Destination	TeachWise AI System
Exception & Alternative flows	In step 2, if the instructor uploads a file in a disallowed format: Display an error message. In step 2, if the instructor uploads a file exceeding the allowed size: Display an error message.

1.3.10 Remove File

Use-Case Description	
Use-Case Name	Remove File
Use-Case Description	Enables instructors to remove desired files from their courses.
Inputs	None
Pre-Conditions	- The user is authenticated and has the instructor role. - File is available. - File is uploaded by this instructor. - The instructor is in the process of editing their course.
Source	Instructor
Steps	1. This use case starts when the instructor selects the remove files option at the desired file. 2. The system updates the view.
Outputs	Files removed
Post-Conditions	If the basic flow steps are successful, then the desired files are removed from the course.
Destination	TeachWise AI System
Exception & Alternative flows	None

1.3.11 Import Students

Use-Case Description	
Use-Case Name	Import Students
Use-Case Description	Allows instructors to import a file with student IDs to enrol them in the course.
Inputs	File with student IDs
Pre-Conditions	- The user is authenticated and has the instructor role. - The instructor is in the process of creating a new course or editing an existing course.
Source	Instructor
Steps	1. This use case starts when the instructor selects the import students' option. 2. The system prompts the instructor to upload the file. 3. The instructor uploads the file. 4. The system updates the view.
Outputs	Uploaded file
Post-Conditions	If the basic flow steps are successful, then the file with student IDs is uploaded to the course.
Destination	TeachWise AI System
Exception & Alternative flows	In step 2, If the instructor uploads a file in a disallowed format: Display an error message.

1.3.12 Add Manually

Use-Case Description	
Use-Case Name	Add Manually
Use-Case Description	Allows instructors to manually add student ID to be enrolled in the course.
Inputs	Student ID
Pre-Conditions	- The user is authenticated and has the instructor role. - The instructor is in the process of creating a new course or editing an existing course.
Source	Instructor
Steps	1. This use case starts when the instructor selects add students' option. 2. The instructor fills in the student's ID. 3. The system updates the view.
Outputs	Added student ID
Post-Conditions	If the basic flow steps are successful, then the student ID is added to the course.
Destination	TeachWise AI System
Exception & Alternative flows	- In step 2, If the instructor fills in an invalid format: Display an error message.

1.3.13 Remove Student

Use-Case Description	
Use-Case Name	Remove Student
Use-Case Description	Enables instructors to remove students from courses.
Inputs	None
Pre-Conditions	- The user is authenticated and has the instructor role. - Course is available. - Course is created by this instructor. - Student is available. - Student in the course. - The instructor is in the process of editing their course.
Source	Instructor
Steps	1. This use case starts when the instructor selects the remove student option for the desired student. 2. The system updates the view.
Outputs	Student removed
Post-Conditions	If the basic flow steps are successful, then the desired student is removed from the course.
Destination	TeachWise AI System
Exception & Alternative flows	None

1.3.14 Chat (Instructor)

Use-Case Description	
Use-Case Name	Chat (Instructor)
Use-Case Description	Allows instructors to chat with a desired course content.
Inputs	Prompt
Pre-Conditions	- The user is authenticated and has the instructor role. - The course has content. - The course is created by the instructor.
Source	Instructor
Steps	1. This use case starts when the instructor selects the chat option. 2. The system displays the chat. 3. The instructor sends their prompt. 4. The system displays the response. 5. If the instructor wants to download the response, then go to the Download Response use case. 6. If the instructor wants to continue chatting, repeat the process.
Outputs	AI Agent Response
Post-Conditions	If the basic flow steps are successful, then a response is displayed to the instructor.
Destination	AI Agent
Exception & Alternative flows	In step 4, if the instructor requests information outside the course content: Display a message informing them the requested information doesn't exist in the course content.

1.3.15 Download Response (Instructor)

Use-Case Description	
Use-Case Name	Download Response (Instructor)
Use-Case Description	Allows instructors to download a response from a course chat.
Inputs	None
Pre-Conditions	- The user is authenticated and has the instructor role. - The response is listed in the list of downloadable responses.
Source	Instructor
Steps	1. This use case starts when the instructor selects the download option. 2. The system formats the response in a file. 3. The system downloads the file.
Outputs	Response file
Post-Conditions	If the basic flow steps are successful, then the response is downloaded in a file format.
Destination	TeachWise AI System
Exception & Alternative flows	None

1.3.16 View Courses Page (Student)

Use-Case Description	
Use-Case Name	View Courses Page (Student)
Use-Case Description	Enables students to view their enrolled courses.
Inputs	None
Pre-Conditions	The user is authenticated and has the student role.
Source	Student
Steps	1. This use case starts when the student navigates to the courses page. 2. The system loads their enrolled courses. 3. If the student wants to leave a course, then go to Leave Course use case. 4. If the student wants to chat with course content, then go to Chat (Student) use case.
Outputs	Enrolled courses
Post-Conditions	If the basic flow steps are successful, then the student should be able to view their enrolled courses.
Destination	TeachWise AI System
Exception & Alternative flows	In step 2, if there are no enrolled courses: Display a message.

1.3.17 Chat (Student)

Use-Case Description	
Use-Case Name	Chat (Student)
Use-Case Description	Allows students to chat with a desired course content.
Inputs	Prompt
Pre-Conditions	- The user is authenticated and has the student role. - The course has content. - The student is enrolled in the course.
Source	Student
Steps	1. This use case starts when the student selects the chat option. 2. The system displays the chat. 3. The student sends their prompt. 4. The system displays the response. 5. If the student wants to download the response, then go to the Download Response use case. 6. If the student wants to continue chatting, repeat the process.
Outputs	AI Agent Response
Post-Conditions	If the basic flow steps are successful, then a response is displayed to the instructor.
Destination	AI Agent
Exception & Alternative flows	In step 4, if the student requests information outside the course content: Display a message informing them that the requested information doesn't exist in the course content.

1.3.18 Download Response (Student)

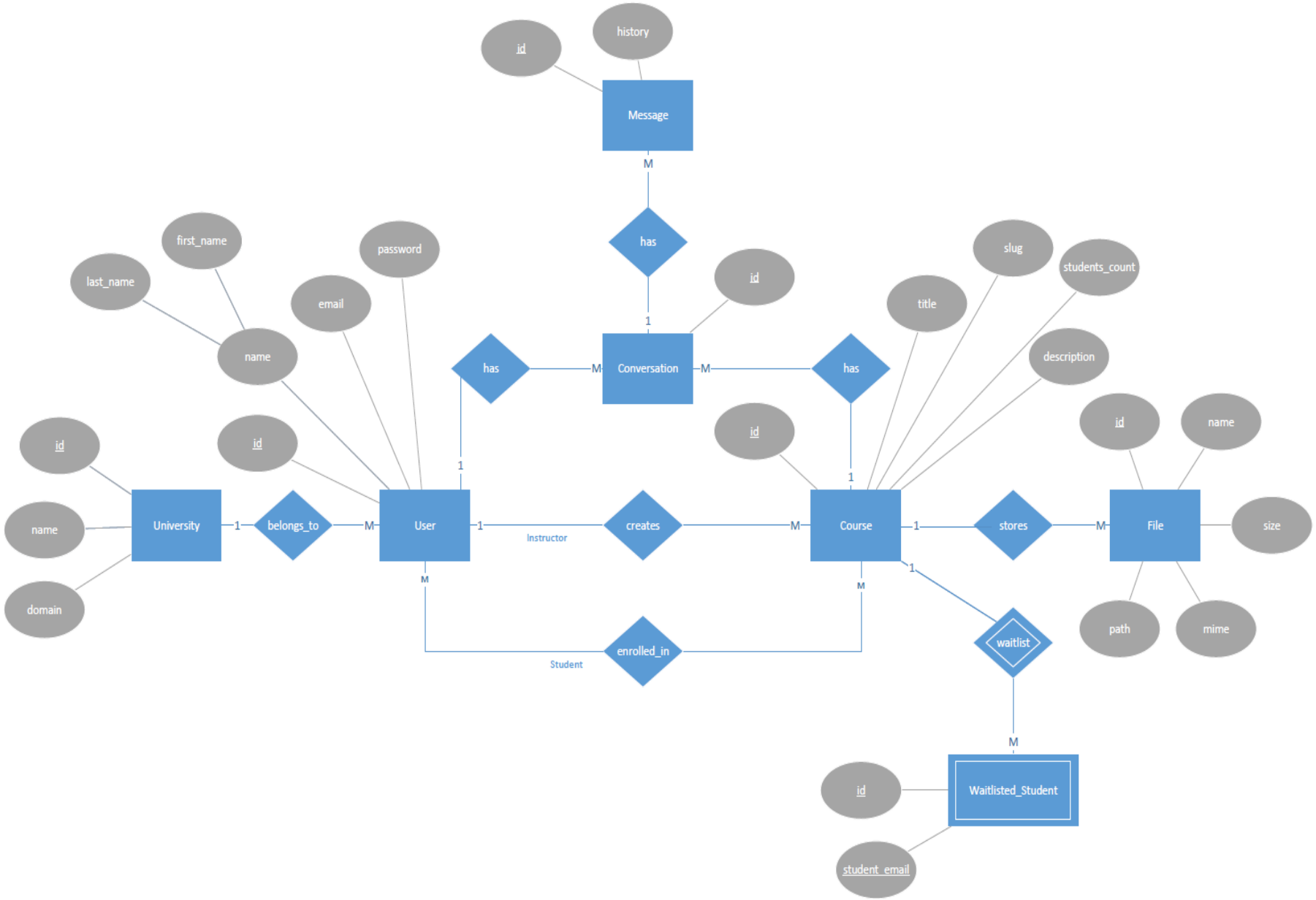
Use-Case Description	
Use-Case Name	Download Response (Student)
Use-Case Description	Allows students to download a response from a course chat.
Inputs	None
Pre-Conditions	- The user is authenticated and has the student role. - The response is listed in the list of downloadable responses.
Source	Student
Steps	1. This use case starts when the student selects the download option. 2. The system formats the response in a file. 3. The system downloads the file.
Outputs	Response file
Post-Conditions	If the basic flow steps are successful, then the response is downloaded in a file format.
Destination	TeachWise AI System
Exception & Alternative flows	None

CHAPTER 2

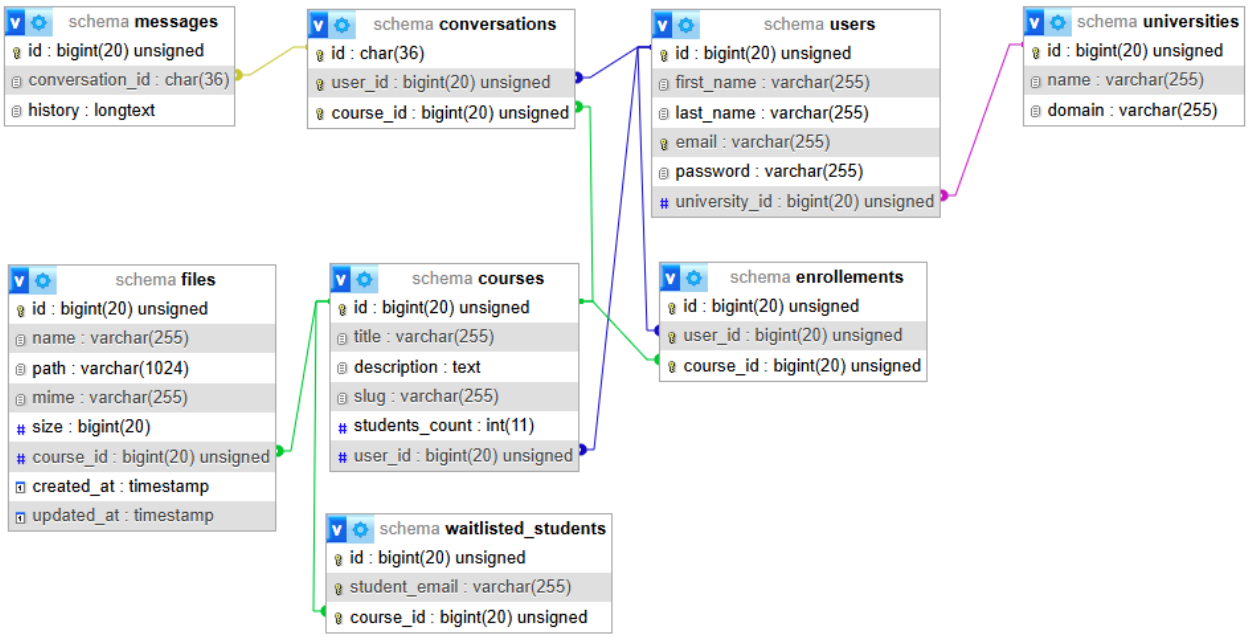
ARCHITECTURE AND DESIGN

2.1.1 Entity-Relationship Diagram (ERD)

.


id
history
Message
M
has
1
id
first_name
password
last_name
email
name
has
M
Conversation
M
has
id
title
slug
students_count
description
id
id
name
1
1
id
name
University
1
belongs_to
M
User
1
Instructor
creates
M
Course
1
stores
M
File
size
domain
M
M
1
path
mime
waitlist
enrolled_in
Student
M
id
Waitlisted_Student
student_email

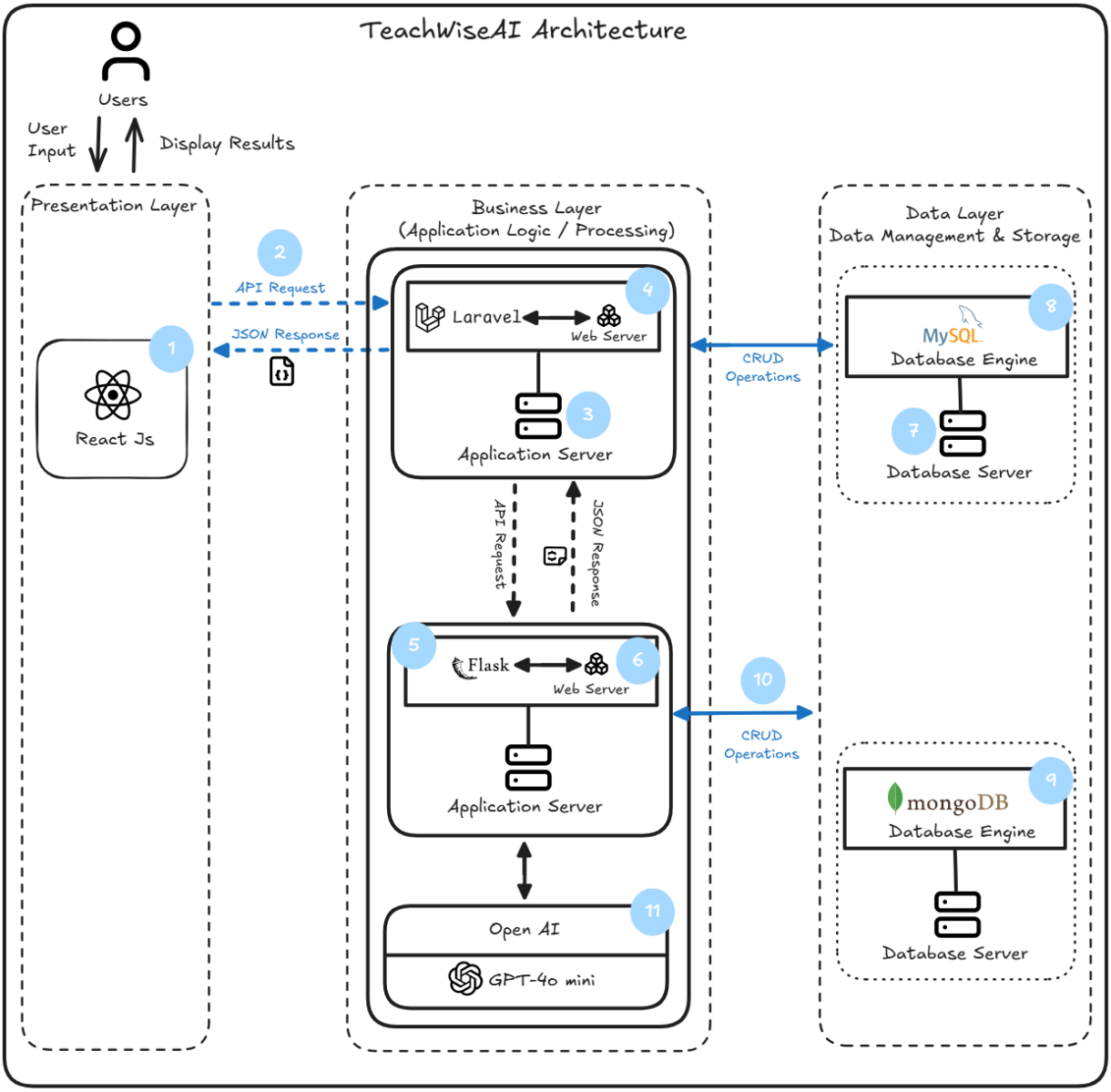
2.1.2 Database Schema



2.2 Architectural Design

2.2.1 TeachWiseAI Architecture

“TeachWiseAI” will combine SPA and 3-Tier Client-Server architectures to produce a unique, flexible, and engaging user interface in addition to efficient backend management. The front end will be an SPA, to allow dynamic interactions without full page reloads. The backend will follow the 3-Tier architecture, by splitting the logic between the servers. This separation ensures a modular design where making a change or an update to one tier doesn’t affect others. Moreover, it allows us to use different technologies for each tier. Thus, it improves maintainability, scalability, and performance.



The Presentation Layer contains the front-end framework, the user's browser uses to interact with the system.

1) In our system, the used front-end framework is React JS, a famous JavaScript library used for user interfaces, known for its component-based architecture and efficient rendering. The reason we chose this framework is that it allows the building of reusable components, which makes the development more modular and maintainable.

2) Since our architecture follows the SPA architecture, the Presentation Layer communicates with the Business Layer by making API calls and receives JSON responses, which are used to dynamically update the user interface as described in SPA. Moreover, the Presentation Layer is allowed to only communicate with the Laravel server without being able to make any API calls to the Flask Server.

In the Business Layer, Laravel and Flask will be used to handle application logic.

3) The application server provides an environment to run and manage dynamic web applications, and it is responsible for processing the business logic. In our system, there are two application servers, each containing a web server, meaning that it is operating within the application server.
4) The first application server uses the Laravel framework, a PHP-based web app framework that uses the MVC (Model-View-Controller) architecture. Laravel provides a convenient structure for handling routing, authentication, and database management, making it a solid foundation for our application. Laravel is only allowed to communicate & perform CRUD operations (refer to point 10) on the database server that is hosting the MySQL database engine.
5) The second application server uses Flask, a micro-web framework for Python that enables developers to create applications quickly. Flask is ideal for building smaller, specialized services that require Python (such as APIs, Data Processing, and Machine Learning). It allows you to handle specific requirements without the overhead of a large framework. It is allowed to perform CRUD operations on both database servers.
6) The web server's function is to manage HTTP requests and deliver content to users. Our system uses two web servers to provide routing requests for each Laravel and Flask application server.
7) The database server is used to host and manage a database, making the hosted database available for CRUD operations (refer to point 10). It is used to run a database management system (DBMS), which is software that allows and facilitates the creation, definition, manipulation, management, and interaction with databases. Our system will have two database servers to fulfill its functionality.

8) The first database engine is MySQL, it is a reliable & well-established system capable of scaling & storing structured data, and handling user data and transactions with ease.
9) The second database engine is MongoDB, a NoSQL database that stores data in a flexible, document-oriented format. Unlike traditional relational databases (like MySQL), which store data in tables with rows and columns, MongoDB stores data in JSON-like documents. This format allows for more flexibility in the data's structure. It is optimized for high performance, capable of reading and writing large amounts of data efficiently & effectively, and extremely useful for applications with big data or high traffic.
10) **CRUD** operations refer to the primary actions used for data interaction: **C**reate, **R**ead, **U**pdate, and **D**elete. These operations allow users to add new data, retrieve existing data, modify their records, and remove data as necessary.
11) The RAG architecture will employ ChatGPT 4o Mini as an independent service, which can only be used via the Flask application server. This LLM model is selected for its lightweight nature, performance optimization, and suitability for quick, cost-efficient responses.

2.2.2 Retrieval-Augmented Generation (RAG) Architecture

Our system will use RAG architecture to generate responses based on user inputs and context. RAG is the process of improving the efficacy of LLM by authorizing knowledge based on its training data sources before generating a response. RAG extends the LLM capabilities without retraining the model or changing its parameters.

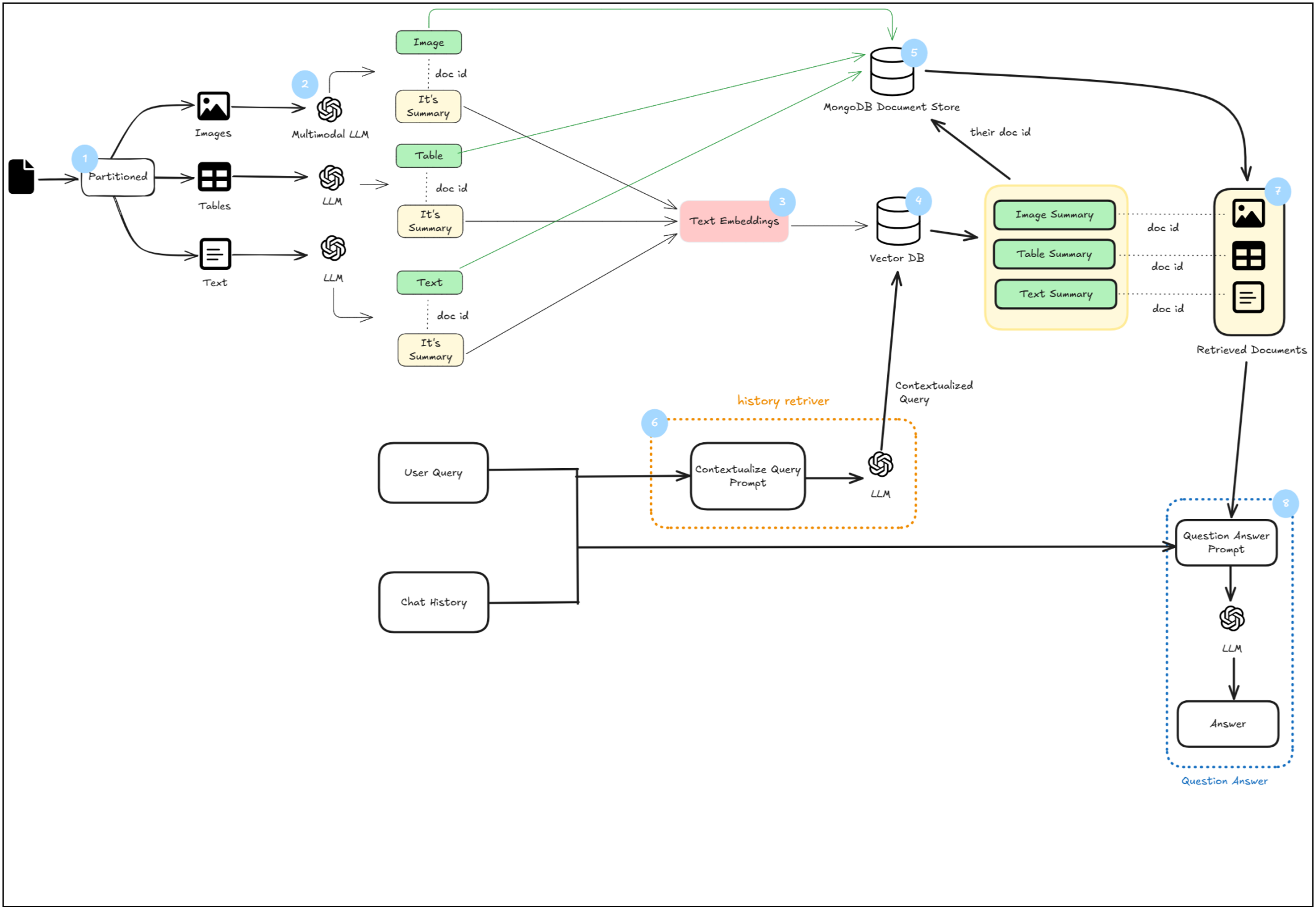
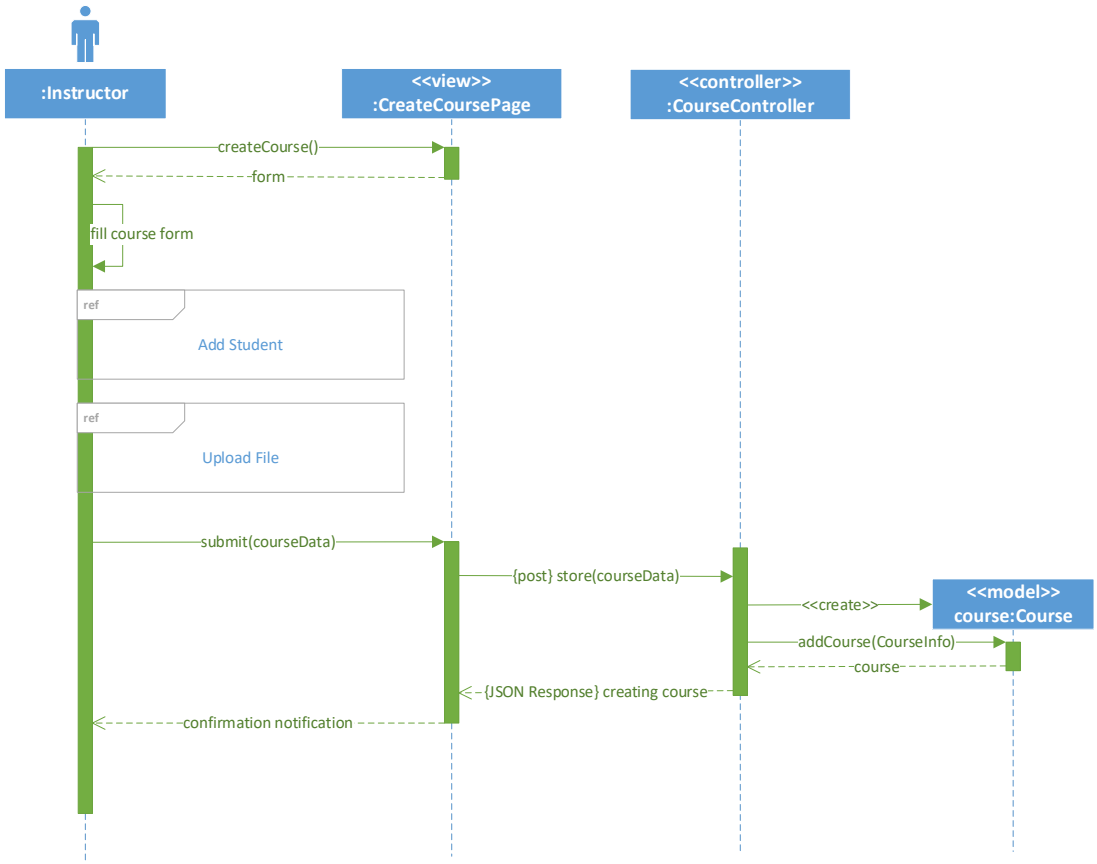

1
Partitioned
Images
Tables
Text
2
Multimodal LLM
LLM
LLM
Image
doc id
It's Summary
Table
doc id
It's Summary
Text
doc id
It's Summary
3
Text Embeddings
4
Vector DB
5
MongoDB Document Store
their doc id
Image Summary
Table Summary
Text Summary
doc id
doc id
doc id
7
Retrieved Documents
history retriver
6
User Query
Contextualize Query Prompt
LLM
Contextualized Query
Chat History
8
Question Answer Prompt
LLM
Answer
Question Answer

Figure 4.5 illustrates how our RAG architecture operates and how data flows through various stages.

1. The uploaded files are partitioned into chunks of text, images, and tables. This is achieved using the UNSTRUCTURED library. The library extracts and processes texts from unstructured document formats of all sorts (PDFs, Word, HTML, Images). This allows us to extract information effectively, fulfilling our application's requirements.
2. Afterwards, each chunk is transferred to the LLM for summarization, preparing it for the next step. Image chunks require a multimodal LLM to understand the text and images. After the chunk preparation, each chunk is given a Doc ID, matched to its associated summary.
3. Each of the text, images, and tables that were summarized is then turned into embeddings, which are vector representations of real-world data in a digestible format for our LLM to extract useful information.
4. Embeddings are then stored in a vector database, storing them as a collection of mathematical representations. The Chroma vector database is used, which is optimized for high-performance searches for vector and large-scale embeddings efficiently.
5. The original chunks are stored in a document store, an efficient system designed for managing, indexing, and retrieving documents efficiently. This will be achieved using MongoDB as discussed previously.
6. The chat history gets retrieved and sent to the history retriever phase the moment the user sends a message. It uses the given prompt to reformulate the user's question to ensure proper data collection and retrieval to correctly retrieve it from the vector database while adding the chat history to the prompt template to ensure the question matches what was asked earlier. After the prompt is ready, it is given to the LLM to generate the query, then it is used by the retriever to retrieve it from the vector database.
7. Once the data is retrieved from the vector database, it will be mapped to the original chunk, then made into a document for the next step.
8. Lastly, the user's question, chat history, and the retrieved documents are entered into a prompt template to instruct the LLM to answer the user's question properly. Afterwards, it is given to the LLM, which generates answers based on the provided context.

2.3 Interaction Diagrams

2.3.1 Sequence Diagram



Create Course Sequence Diagram