COMPUTER AND ARTIFICIAL INTELLIGENCE

GRADUATION PROJECT

NeuraLearn Academy

Khaled Mohamad Fathy
Sherif Ahmed Shehata Hammam
Mahmoud Ahmad Zitony
Basem Yahia Abdelatif Ahmed
Karim abdelazim Mohamed shehtata
Mohamed Ashraf Mohamed Badwy
Beshoy Tag Makram

Supervisor
Dr. Gehad HASSAN

June 24, 2024

Abstract

NeuraLearn Academy is an innovative educational platform leveraging generative AI and Large Language Models (LLMs) to enhance online learning experiences.This platform introduces new workflows utilizing AI technologies, benefiting both instructors and students. Despite the rapid growth of online education, traditional platforms often struggle with issues such as limited interactivity, lack of personal engagement, and reliance on digital assessments. The primary objective of our proposed Model NeuraLearn Academy is to revolutionize online learning by integrating generative AI and Large Language Models (LLMs). For instructors, the platform provides deep learning systems to facilitate the generation of insightful questions and summaries, enhancing teaching effectiveness. Meanwhile, students benefit from the Retrieval-Augmented Generation (RAG) architecture, enabling interactive educational chats that foster deeper understanding and engagement with course material.our proposed Model have demonstrated exceptional performance in question generation and summarization tasks. Our question generation model achieved ROUGE scores of 40% in ROUGE-1, 20% in ROUGE-2, and 39% in ROUGE-L. Similarly, our summarization model excelled on the BookSum benchmark with scores of 33% in ROUGE-1, 5.2% in ROUGE-2, 23% in ROUGE-L, and 29.977% in ROUGE-LSUM. These results affirm the effectiveness of our approach in enhancing educational interactions and content comprehension in online learning environments.

Acknowledgements

We would like to take this opportunity to express our sincere appreciation and best wishes to our devoted supervisor, Dr. Jehad, for her great leadership and ongoing motivation during the completion of this project. The assistance, and guidance she occasionally provided will help us get far along in the life's journey we are about to take.

We also take this occasion to thank Faculty of Computers and Artificial Intelligence Council for their support and work in getting us all the resources we need.

We would especially like to thank the FCAI members for the useful information they gave in their respective domains. Finally, we express our gratitude to Almighty, our parents, and everyone who has provided assistance, encouragement, and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

1.1.1 Problem Definition

Over the past decade, online learning has experienced significant growth, capitalizing on the fusion of the internet and education to offer individuals the opportunity to acquire new skills. The COVID-19 pandemic has further propelled online learning into a central position in people's lives, compelling educational institutions and businesses to embrace remote work and education. This surge in demand has given rise to a multitude of online learning platforms like Udemy, Coursera, Lynda, Skillshare, and Udacity, serving millions of users and evolving based on different user needs. Additionally, prestigious universities such as Stanford and Harvard are contributing to the democratization of education by providing accessible online courses spanning computer science, engineering, mathematics, business, art, and personal development.

Advantages of Online Learning

- **Efficiency**: Online learning enables teachers to efficiently deliver lessons using tools such as videos and PDFs.
- **Accessibility of Time and Place**: Students can attend classes from any location, breaking geographical barriers and reaching a broader network of students.
- **Affordability**: Online education proves to be more cost-effective compared to traditional learning methods.
- **Suits a Variety of Learning Styles**: Online learning caters to diverse learning styles, accommodating visual, auditory, and independent learners.

Disadvantages of Online Learning

- **Lack of Personal Interaction**: Online learning lacks face-to-face interaction between students and instructors, missing the dynamic of traditional classrooms.
- **Limited Hands-On Experience**: Certain disciplines require hands-on experience, which online learning may struggle to provide adequately.
- **Interactivity**: While online education offers interactive elements like discussion forums and virtual classrooms, the level of engagement may vary.
- **Assessment Methods**: Digital assessments in online education may be convenient, but traditional education often incorporates a mix of digital and traditional assessment methods.

1.1.2 The Role of Generative AI

Generative Artificial Intelligence (AI) has become a revolutionary force in education, particularly in online learning. Harnessing the advancements in technology, educators are leveraging generative AI to create personalized, engaging experiences for students. In this article, we will explore how generative AI transforms student interaction with online materials, fostering a more dynamic and effective learning environment.

1.2 Introducing NeuraLearnAcademy

The project aims to develop a new generation of learning platforms that address the challenges of online learning using Generative AI. NeuraLearnAcademy merges the power of Generative AI with an online platform, offering a solution to the drawbacks associated with traditional online education. This innovative approach aims to enhance interactivity, overcome the lack of personal interaction, and provide a more immersive and effective learning experience for students.

1.2.1 Problem Solution

Generative Artificial Intelligence (Generative AI) refers to a subset of artificial intelligence that focuses on creating and generating new content, such as images, text, or other data types. Unlike traditional AI models that rely on pre-existing data for classification or prediction tasks, generative models have the capability to generate novel and coherent output. These models learn patterns and structures from training data and use that knowledge to create new, similar content.

A language model is a type of artificial intelligence that is specifically designed to understand and generate human-like language. Language models learn the structure, grammar, and context of language from large datasets and can be utilized for various natural language processing tasks, such as machine translation, text summarization, and text completion. One notable example of a language model is OpenAI's GPT (Generative Pre-trained Transformer), which has demonstrated remarkable proficiency in understanding and generating coherent text.

1.3 Generating Questions and Summarizing Video Transcripts

Transcribing videos and extracting valuable information from them can be a powerful method for knowledge extraction. This process can be further enhanced by utilizing language model techniques for generating questions and summarizing the content.

Once you have identified key information from the transcripts, you can use language models to generate questions automatically. These questions can be crafted based on the content, aiming to cover essential topics, clarify ambiguities, or prompt further exploration. Question generation models can be trained on existing datasets or fine-tuned for specific domains. There are various approaches to summarizing textual content, and they can be adapted to video transcripts as well. Extractive summarization involves selecting the most important sentences or phrases from the transcript, while abstractive summarization generates a concise summary in the model's own words.

1.4 Integration with Platform

Integrate the trained question-answering model seamlessly into your educational platform. Design a user-friendly interface that allows users to submit their questions or feedback related to the video content. Enable the question-answer model to generate dynamic responses based on the context of the questions and feedback. The model should consider the entire transcript, relevant sections, and any additional information available to provide accurate and context-aware responses.

Implement interactive platforms or chatbots that can engage with users based on the generated questions. These platforms can provide a more dynamic and personalized learning experience.

1.5 Motivation

In today's education scene, Machine Learning isn't just a passing trend; it is a powerful tool that can fundamentally transform the way we teach and learn. By integrating Machine Learning into our project, we want to use its capabilities to revolutionize traditional educational methods and improve the overall learning experience.

1.5.1 Comprehensive Understanding

- **Goal**: Ensure every student comprehensively understands each chapter of the course.
- **How**: Implement a post-chapter feedback mechanism where students interact with ChatGPT, expressing what they've grasped and clarifying any uncertainties. This iterative process ensures a solid foundation before progressing to subsequent chapters.

1.5.2 Personalized Knowledge Assessment

- **Goal**: Tailor assessments to each student's understanding and learning pace.
- **How**: Develop a dynamic quiz model that adapts to individual progress, assessing not only factual knowledge but also the ability to connect concepts across chapters. This personalized approach enhances the learning experience and identifies areas that may need additional focus.

1.5.3 Summarization for Reinforcement

- **Goal**: Reinforce learning through concise summarization.
- **How**: Implement a summarization model that extracts key points from each chapter, generating a PDF file containing both a textual summary and a visual representation of important concepts discussed in the video. This resource aids in reinforcement and serves as a quick reference for students.

1.5.4 Iterative Learning Journey

- **Goal**: Facilitate a structured and iterative learning journey.

– **How**: Restructure the course format to unlock subsequent chapters only after the student submits an understanding summary for the previous chapter. This ensures a stepwise progression, solidifying knowledge before advancing.

1.5.5 Continuous Improvement Feedback Loop

– **Goal**: Establish a continuous improvement feedback loop.

– **How**: Gather feedback from students on the effectiveness of the ChatGPT interactions, summarization model, and quizzes. Utilize this feedback to enhance the platform, ensuring its responsiveness to the diverse needs and learning styles of users.

By aligning these goals and objectives, NeuraLearnAcademy aims to create a dynamic and adaptive learning environment that not only imparts knowledge but also actively engages students in the learning process, fostering a deeper and more sustained understanding of course material.

1.6 Project Limitations

The primary limitation lies in the accuracy of assessing students to gauge their comprehension of specific course sections. Ensuring precise evaluation of understanding within the platform is a critical focus, recognizing potential challenges in achieving absolute accuracy due to inherent complexities in subjective assessments.

Chapter 2

Project Planning

2.1 Why Planning ?

In project management, planning is crucial for successful project implementation. so we should carefully planning and choose the methodology that fit our project and specify, prioritize, manage the main project tasks, as well as identify and manage potential risks. Furthermore, effective planning can significantly enhance project performance and success rates, leading to cost and time savings. Additionally, it improves team communication, ensuring the best utilization of resources making it easier to track project goals and outcomes.

2.2 Scrum Overview

What is Scrum? Scrum is an agile way to manage work. Ever few weeks (typically two to four), teams deliver a fully functional chunk of work (an increment). Teams and the business use the feedback from each delivery to determine what to build next, or how to adapt what they've already built. Scrum works through a series of events that happen over a defined period of time: that time period is called a sprint. Sprints are short timeboxes during which the team turns ideas into working product. The events then repeat every sprint.

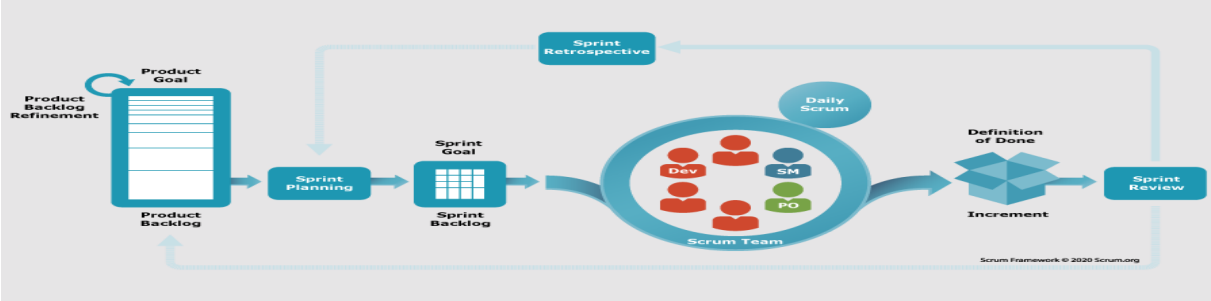


Figure 2.1: Scrum Overview

2.3 Scrum Activities and Events

- **Sprint Planning.**

  Each sprint begins with a sprint planning meeting, where the team leader presents the top items on the product backlog to the team, and team members figure out how much work they can commit to during the coming sprint.

- **The Sprint.**

  During each sprint, the team takes a small set of features from idea to fully implemented and tested functionality. At the end, these features are done and could potentially be released.

- **Daily Scrum.**

  On each day of the sprint, all team members attend a daily scrum meeting. Daily scrums are a way for team members to synchronize their work and collaborate to move that work to done. The daily meetings last no more than 15 minutes and are intended to give the team a time to share what they worked on the prior day, will work on that day, and identify any impediments to progress.

- **Sprint Review.**

  At the end of a sprint, the team conducts a sprint review during which the team demonstrates the new functionality to the stakeholder who wishes to provide feedback that could influence the next sprint. It's critical that the sprint review remains informal and doesn't become its own task, distracting from the work itself.

2.4 Why Scrum?

1. **Responsive team:** with scrum, our teams will be more responsive in their productivity, especially as changes and pivots are required. The scrum discipline requires frequent reviewing of progress, which often demands changes to prevent a project from failing.

2. **More accurate planning:** by using Scrum, our plans will be less apt to fail. Why? Because our teams are constantly putting in the effort to keep them on track by shifting and changing as needed. And because of the way scrum is designed, our teams will constantly be reflecting how things are going and can make small or large adjustments to the plans, according to the winds of change. By adhering to scrum artifacts and events, our plans are far less likely to fail.

3. **Everyone in sync:** when using scrum, a project's stakeholders are always in sync. And because the scrum methodology prioritizes individuals and interactions over all else, keeping everyone involved in sync is actually built into the process.

4. **Flexible priorities:** with scrum, it's very easy to prioritize and re-prioritize as the project moves through the process. With this ability, our developer team become more flexible and our project becomes more agile. This also makes it possible to easily (and quickly) adjust short-term goals while still adhering to the overall strategy of the project.

5. **More control:** finally, we will have more control over the entire project. That's not to say we will be able to better control our staff. No. Instead, we have more control over the direction and flow of the development process. And when we have consistent input from developers and other stakeholders, it lends a level of cohesion to the process you wouldn't otherwise have.

2.5 Main tasks of our project

- Design UI and UX
- Design database diagrams
- Implement the frontend
- Implement the backend
- Test and debug the project
- Writing the documentation

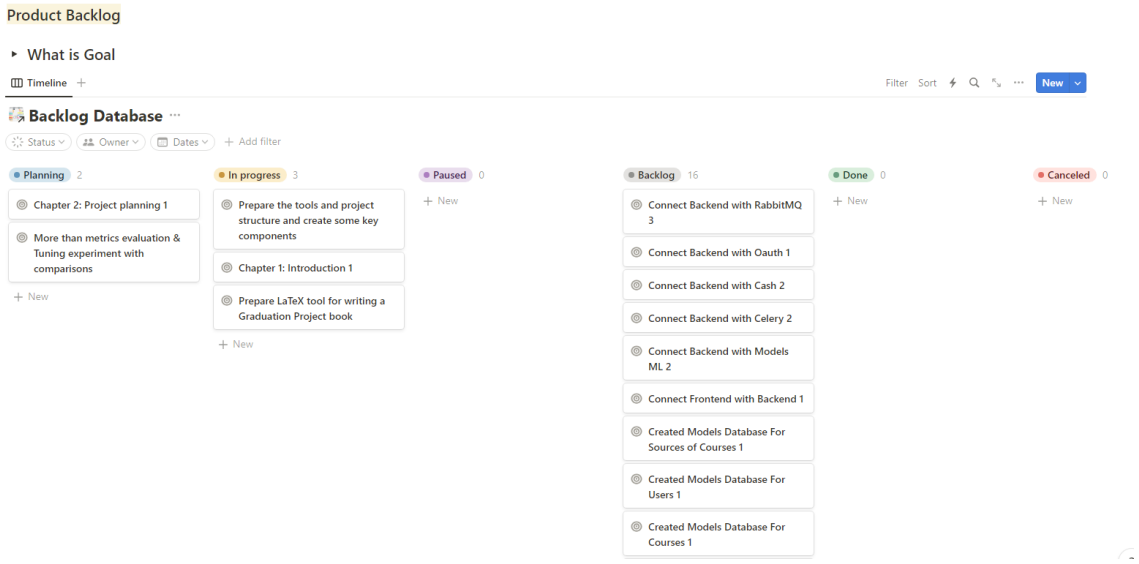


Figure 2.2: Snapshot Of backlog

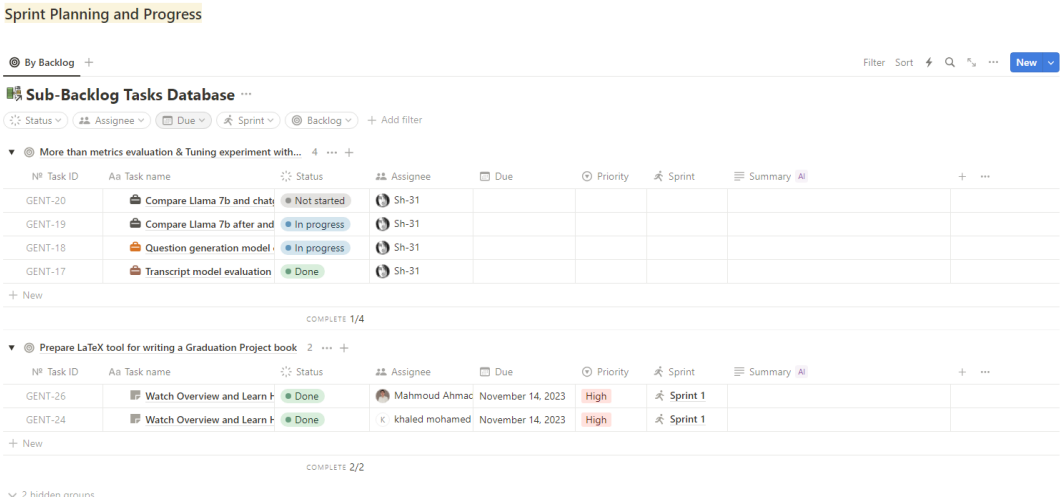


Figure 2.3: Snapshot Of sprint planning

2.6 Risk Identification

Project Risk Management includes the processes of conducting risk management planning, identification, analysis, response planning, and controlling risk on a project. The objectives of project risk management are to increase the likelihood and impact of positive events and decrease the likelihood and impact of negative events in the project. The key step in risk management planning is project risk identification. Risk identification identifies the risks that could have an impact on the project and lists their characteristics. However, as recommended, we should avoid devoting too much time to risk identification.

2.6.1 Technology Dependencies

Risk: Dependencies on third-party technologies or frameworks may undergo updates or discontinuation, impacting the platform's functionality.
Mitigation: Regularly update and maintain dependencies, conduct thorough compatibility checks, and have contingency plans for potential disruptions.

2.6.2 Data Security and Privacy

Risk: Security breaches or data privacy issues could compromise user information and erode trust.
Mitigation: Implement robust security measures, adhere to data protection regulations, and conduct regular security audits to identify and address vulnerabilities.

2.6.3 User Adoption

Risk: Users may find the platform challenging to navigate, leading to low adoption rates.
Mitigation: Implement a user-friendly onboarding process, provide clear documentation, and offer responsive customer support to ensure users can easily navigate and utilize the platform.

2.6.4 Scalability Challenges

Risk: An unexpected surge in user traffic may lead to server overload, causing system slowdowns or crashes.
Mitigation: Utilize scalable cloud services, conduct load testing, and have a scalable infrastructure in place to accommodate a growing user base.

2.6.5 Content Quality

Risk: Inaccuracies or insufficient quality in course content may impact the effectiveness of the learning experience.
Mitigation: Establish a thorough content review process, actively seek user feedback on content quality, and iterate based on user input.

2.6.6 Technological Obsolescence

Risk: Rapid advancements in technology may render certain components of the platform obsolete.
Mitigation: Regularly update the technological stack, monitor emerging technologies, and plan for phased upgrades to avoid obsolescence.

2.6.7 Adaptability to Learning Styles

Risk: The platform may not fully cater to the diverse learning styles and preferences of users.
Mitigation: Gather user feedback on learning experiences, conduct usability testing with a diverse user base, and iterate the platform design to enhance adaptability.

2.6.8 Integration with External Systems

Risk: Challenges in integrating seamlessly with external platforms or tools may hinder collaboration opportunities.
Mitigation: Conduct thorough compatibility testing with external systems, establish robust APIs, and foster collaborations through ongoing communication with potential partners.

2.6.9 Regulatory Compliance

Risk: Changes in educational or data protection regulations may necessitate adjustments to the platform.
Mitigation: Stay informed about regulatory changes, conduct regular compliance checks, and adapt the platform accordingly to ensure alignment with current standards.

2.6.10 User Technical Proficiency

Risk: Users with limited technological skills may struggle to navigate and utilize the platform effectively.
Mitigation: Provide comprehensive user guides, tutorials, and responsive customer support to assist users with varying levels of technical proficiency.

Chapter 3

System Analysis

3.1 System Requirements

A requirement is simply a statement of what the system must do or what characteristics it needs to have. During a systems development project, requirements will be created that describe what the business needs (business requirements), what the users need to do (user requirements), what the software should do (functional requirements), characteristics the system should have (nonfunctional requirements), and how the system should be built (system requirements).

3.1.1 Functional Requirements

Functional requirements are the specifications of the product's functions (features). In another words, functional requirements define what precisely a software must do and how the system must respond to inputs. Functional requirements define the software's goals, meaning that the software will not work if these requirements are not met.

1. **Authentication**

    1.1. The system will allow users to create an account.

    1.2. The system must validate users credentials to login.

    1.3. Users should have the ability to reset their passwords in case of forgotten credentials.

2. **Enroll Courses**

    2.1. Students can enroll courses and see course content We will extend this to enroll paid course wit credit card, debit card etc. in the future.

3. **Course Managment**

    3.1. Instructors can create a course and upload to our website.

    3.2. Instructors can upload various types of course content, including text, multimedia, and documents.

    3.3. Only authorized instructors should have the ability to modify or update course content.

3.1.2 Non-Functional Requirements

Non-functional Requirements define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs. Also known as system qualities, non-functional requirements are just as critical as functional requirements. They ensure the usability and effectiveness of the entire system. They specify criteria that judge the operation of a system, rather than specific behaviors.

- **Performance**

  The application should be able to handle large numbers of concurrent users.

- **Security**

  The application should protect user data from unauthorized access or theft.

- **Scalability**

  The application should be able to handle an increasing number of users and services.

- **User-friendliness**

  The application should be easy to use and navigate, with clear instructions and explanations of the analysis process.

- **Privacy**

  The application should have a clear privacy policy and should not retain user data

3.2 Exploring Trade-offs Architectures

- Sequence to sequence models (LSTM-GRU-RNN):
  - Difficulty in Incorporating Domain Knowledge
  - Limited Interpretability
  - Handling Varied Output Lengths
- Encoder-Decoder Transformer:
  - Transformer architecture, with self-attention mechanisms, has been widely used in Seq2Seq tasks.
  - It does come with some potential disadvantages, including high training costs.
  - Alternative way Fine-Tuning and Transfer Learning.

Model	Parameters	Jump Factor	Chinchilla Tokens (B)	Jump Factor	CS-2 Config	Days To Train	Jump Factor	Price To Train	Jump Factor	Cost Per 1M Parameters
GPT-3XL	1.3		26		4 * CS-2	0.4		$2,500		$1.92
GPT-J	6	4.6 X	120	4.6 X	4 * CS-2	8	20.0 X	$45,000	18.0 X	$7.50
GPT-3 6.7B	6.7	1.1 X	134	1.1 X	4 * CS-2	11	1.4 X	$40,000	0.9 X	$5.97
T-5 11B	11	1.6 X	*34*	0.3 X	4 * CS-2	9	0.8 X	$60,000	1.5 X	$5.45
GPT-3 13B	13	1.2 X	260	7.6 X	4 * CS-2	39	4.3 X	$150,000	2.5 X	$11.54
GPT NeoX	20	1.5 X	400	1.5 X	4 * CS-2	47	1.2 X	$525,000	3.5 X	$26.25
GPT NeoX	*20*	*1.5 X*	***400***	*1.5 X*	***16 * CS-2***	***11.1***	*0.3 X*	***$656,250***	*4.4 X*	***$32.81***
GPT 70B	70	3.5 X	1,400	3.5 X	4 * CS-2	85	1.8 X	$2,500,000	4.8 X	$35.71
GPT 70B	*70*	*3.5 X*	***1,400***	*3.5 X*	***16 * CS-2***	***21.3***	*0.3 X*	***$3,125,000***	*6.0 X*	***$44.64***
GPT 175B	175	2.5 X	3,500	2.5 X	4 * CS-2	***110.5***	1.3 X	***$8,750,000***	3.5 X	***$50.00***
GPT 175B	*175*	*2.5 X*	***3,500***	*2.5 X*	***16 * CS-2***	***27.6***	*0.3 X*	***$10,937,500***	*4.4 X*	***$62.50***

Figure 3.1: Transformer

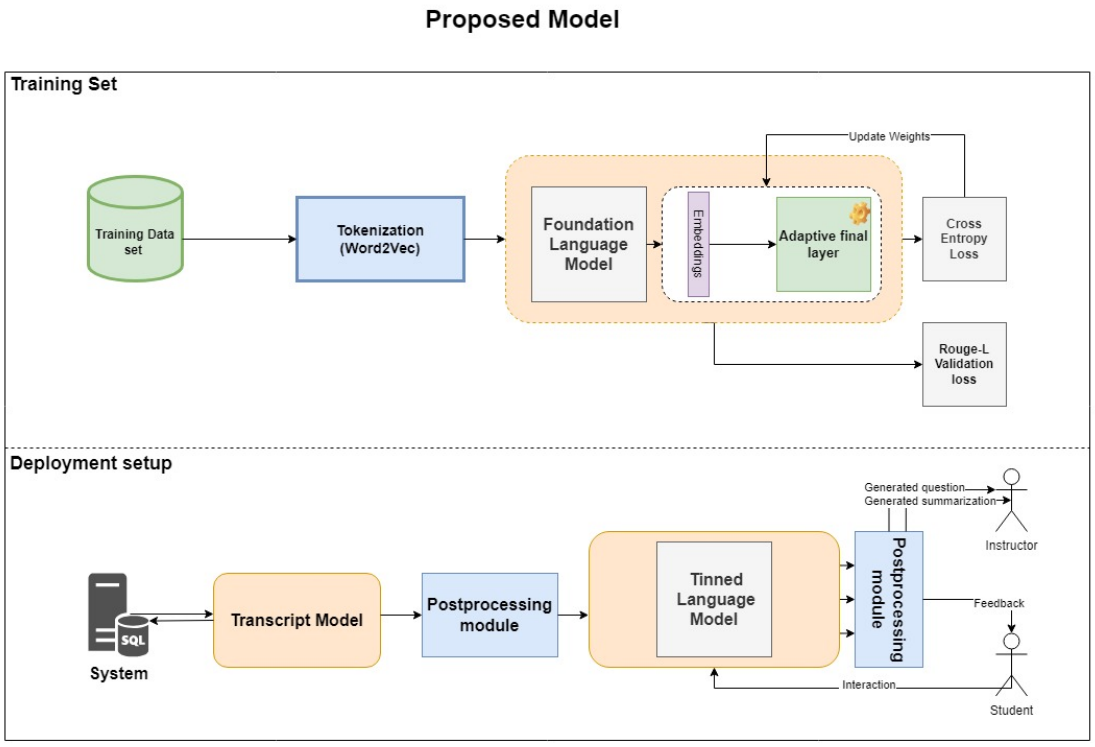


Figure 3.2: Proposed Model

The motivation behind designing a large language model with adaptive layers lies in the pursuit of creating a model that can efficiently and effectively adapt to the complexities and nuances of different datasets and tasks

3.2.1 Flexibility Across Diverse Tasks:

Adaptive layers enhance the model's flexibility to handle various natural language processing tasks. Whether it's text classification, language translation, sentiment analysis, or any other task, the adaptive layers allow the model to dynamically adjust its parameters based on the specific requirements of each task.

3.2.2 Optimizing for Different Data Distributions:

Different datasets may exhibit varying data distributions and characteristics. Adaptive layers provide a mechanism for the model to adapt its learning rates or normalization parameters dynamically, optimizing its performance for the specific patterns present in the data.

3.2.3 Enhanced Convergence and Training Speed:

Adaptive learning rate methods contribute to faster convergence during training. By adapting learning rates for individual parameters, the model can converge

more quickly, leading to reduced training times and improved overall efficiency.

3.3 Use Case Analysis

A Use Case is a type of behavioral representation that shows the interactions between a system and its users, or actors. It is used to represent and model the functionality of a system. The main purpose of a use case diagram is to capture the functional requirements of a system.

There are two formats to represent use cases:

1. Use case diagram.
2. Use case specifications or scenario in textual format.

3.4 Use Case Diagram

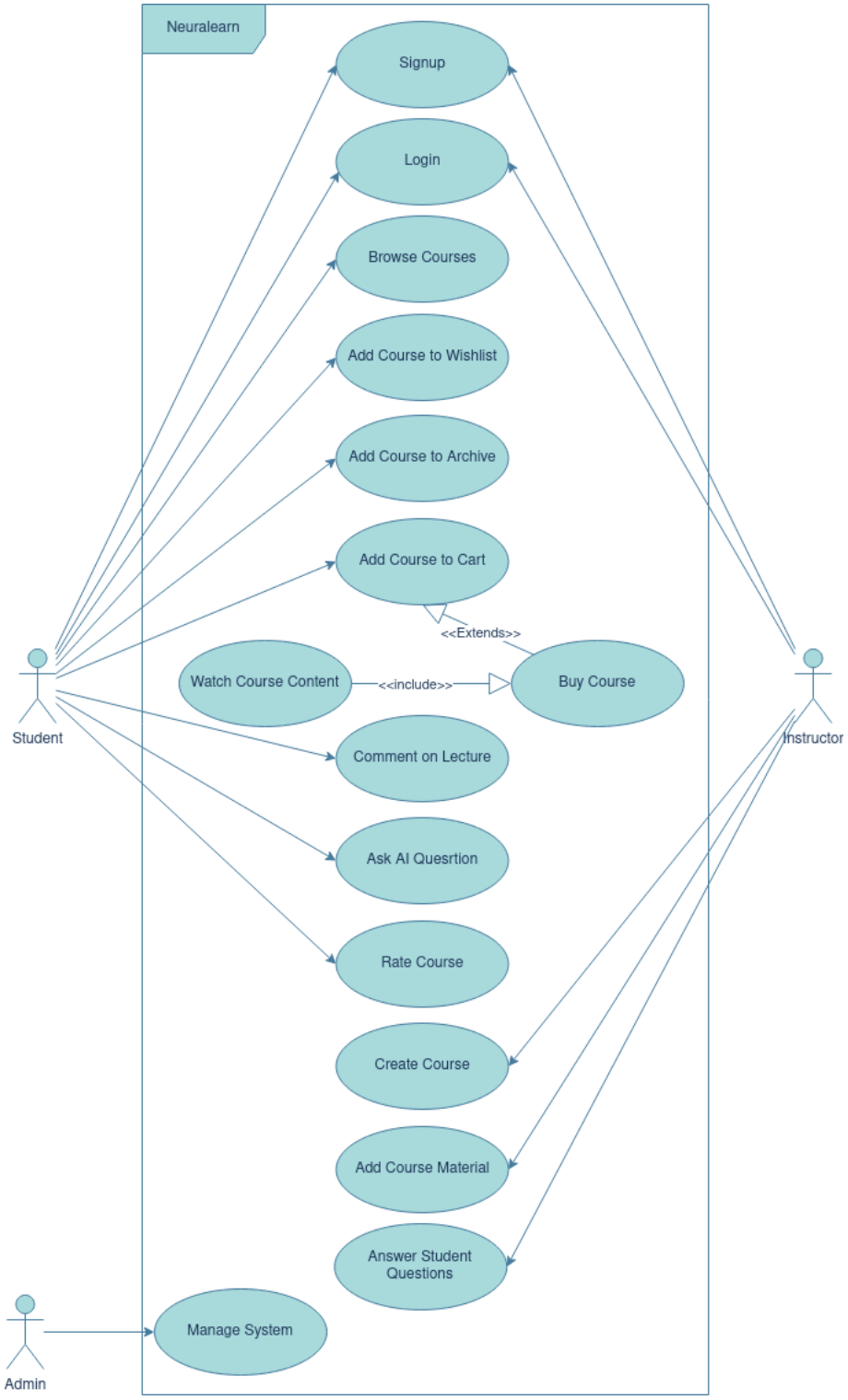


Figure 3.3: UML Use Case Diagram

Table 3.1: Use Case: Sign-Up

Use Case Name	Sign-Up
Number	1
Actor	Student, Instructor, and Admin
Preconditions	• Full name • Email address • Date of birth • Username • Password (and password confirmation)
Flow of Events	• The student clicks on the sign-up button, leading to the registration form page. • The form prompts the student to enter their full name, email address, date of birth, username, and password. • The student fills in the required information and clicks the ”Submit” button.
Exception	If the verification email is not received, the student may have the option to request a new verification.

Table 3.2: Use Case: Login

Use Case Name	Login
Number	2
Actor	Student, Instructor, and Admin
Preconditions	• The student has successfully registered an account on the platform. • The student has a valid username/email and password.
Flow of Events	• The student clicks on the sign-up button, leading to the registration form page. • The form prompts the student to enter their full name, email address, date of birth, username, and password. • The student fills in the required information and clicks the ”Submit” button.
Postcondition	• The student is successfully logged into the platform. • The platform displays the student's personalized dashboard with relevant information. • Can Browse Courses • Add Course to wishlist • Add Course to Archive • Add Course to Cart • Comment on lecture • Rate Courses • Ask Question
Exception	Invalid Credentials: If the entered credentials are invalid (e.g., incorrect password or username), the platform displays an error message and prompts the student to re-enter the information.

Table 3.3: Use Case: Add Course to Cart

Use Case Name	Add Course to Cart
Number	3
Actor	Student
Preconditions	• The student is logged into their account. • The student has navigated to the course catalogue or details page.
Flow of Events	• The student browses the available courses in the platform's catalogue. • The student selects a course they are interested in by clicking on its title or a designated "Add to Cart" button. • The platform adds the selected course to the student's shopping cart. • Optionally, the student may choose to view their cart to review the selected course and its details. • The student may choose to continue browsing and adding more courses to the cart.
Postcondition	The selected course is successfully added to the student's shopping cart.
Exception	• If the student tries to add a course to the cart that is already present, the platform recognizes this as a duplicate request. • In case of technical issues, such as server downtime or network problems, during the course addition to the cart, the platform informs the student about the problem.

Table 3.4: Use Case: Buy Course

Use Case Name	Buy Course
Number	4
Actor	Student
Preconditions	• The student has added at least one course to their shopping cart.
Flow of Events	• The student decides to proceed with the purchase. • The student views the contents of their shopping cart, confirming the courses they want to purchase. • The student clicks on the ”Proceed to Checkout” button. • The platform prompts the student to provide billing information such as credit card details or select a saved payment method. • The student reviews the order summary, including the selected course(s) and the total cost. • The student confirms the purchase by clicking on the ”Confirm Purchase” or ”Place Order” button. • The platform processes the payment using the provided billing information. • Successful payment, the platform displays a purchase confirmation message.
Postcondition	• The purchased course is now accessible in the student’s account. • The student may receive a confirmation email with details of the purchase.
Exception	• If the student attempts to purchase a course but does not have sufficient funds in their account, the platform notifies the student about the insufficient balance. • The student may be prompted to try the purchase again or choose an alternative payment method. • In the event of a technical issue or a failure in processing the payment transaction, the platform informs the student about the problem.

Table 3.5: Use Case: Add Course Material

Use Case Name	Add Course Material
Number	5
Actor	Instructor
Preconditions	• The instructive actor is logged into the platform. • The instructive actor has the necessary permissions to add course materials. • A course has been created, and the instructive actor has access to it.
Flow of Events	• The instructive actor navigates to the course dashboard or management section. • The instructive actor selects the specific course for which they want to add course materials. • Within the course management interface, the instructive actor finds and clicks on the ”Materials” or ”Content” section. • The instructive actor selects the type of material they want to add and uploads the file or provides the necessary information. • Can use AI to generate Questions or summarization for a section on the course.
Postcondition	• The course material is successfully added and available within the specified course. • Students enrolled in the course can access the added material based on the visibility settings.
Exception	• If there is an issue with uploading the material (e.g., file format not supported, network error), the system provides an error message and prompts the instructive actor to try again.

3.5 Sequence Diagram

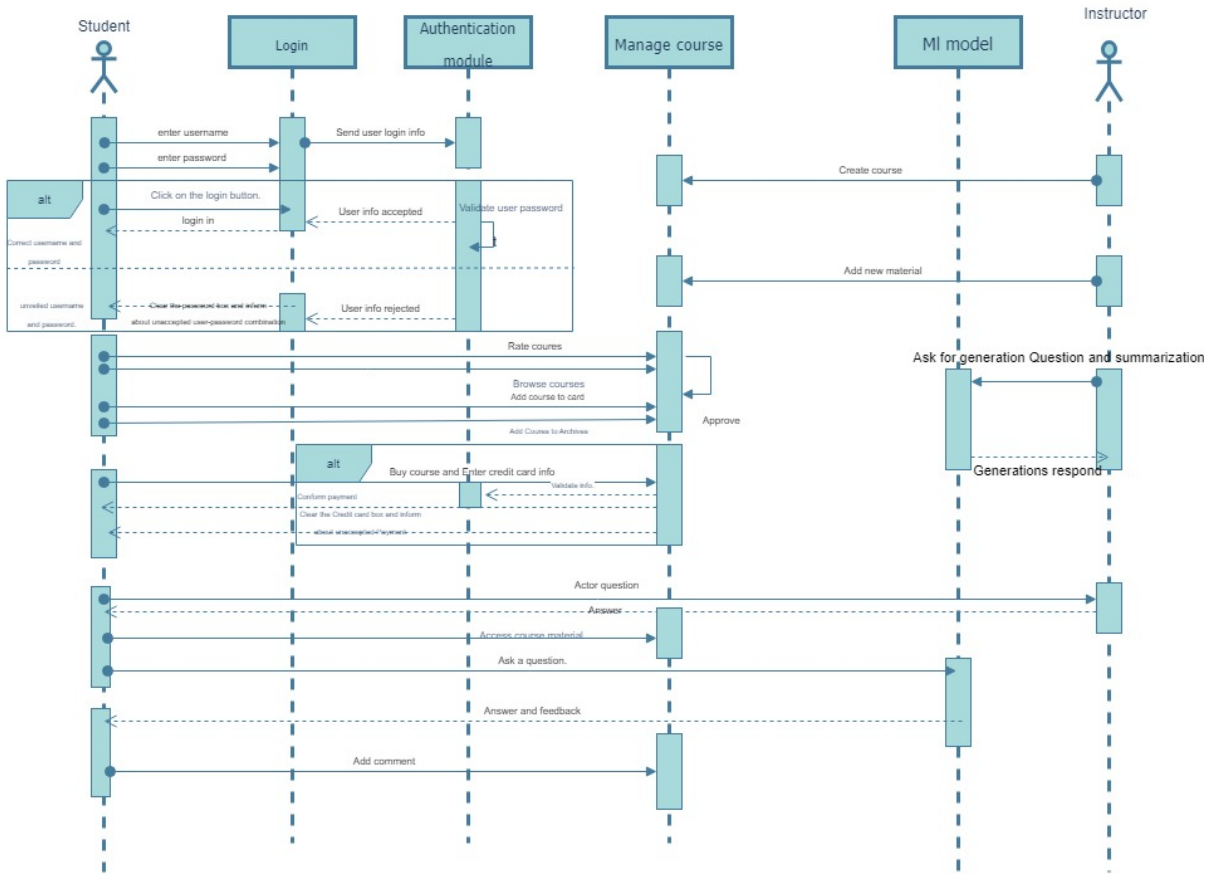


Figure 3.4: Sequence Diagram

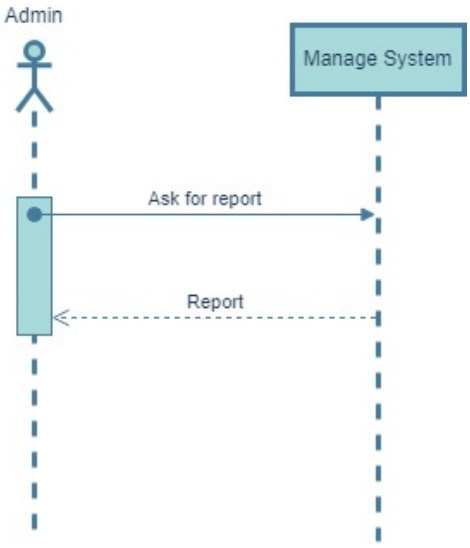


Figure 3.5: Sequence Diagram

3.6 Context Diagram

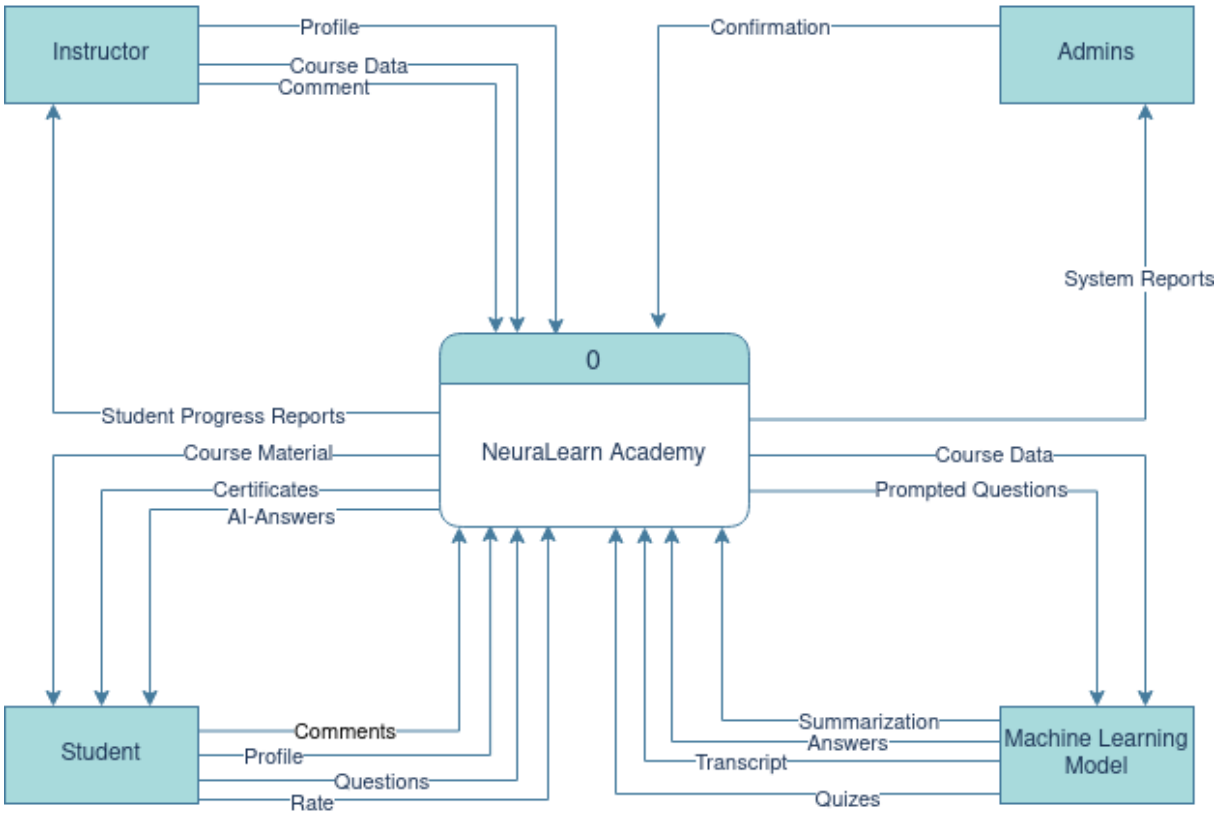


Figure 3.6: Context Diagram

3.7 Data Flow Diagram

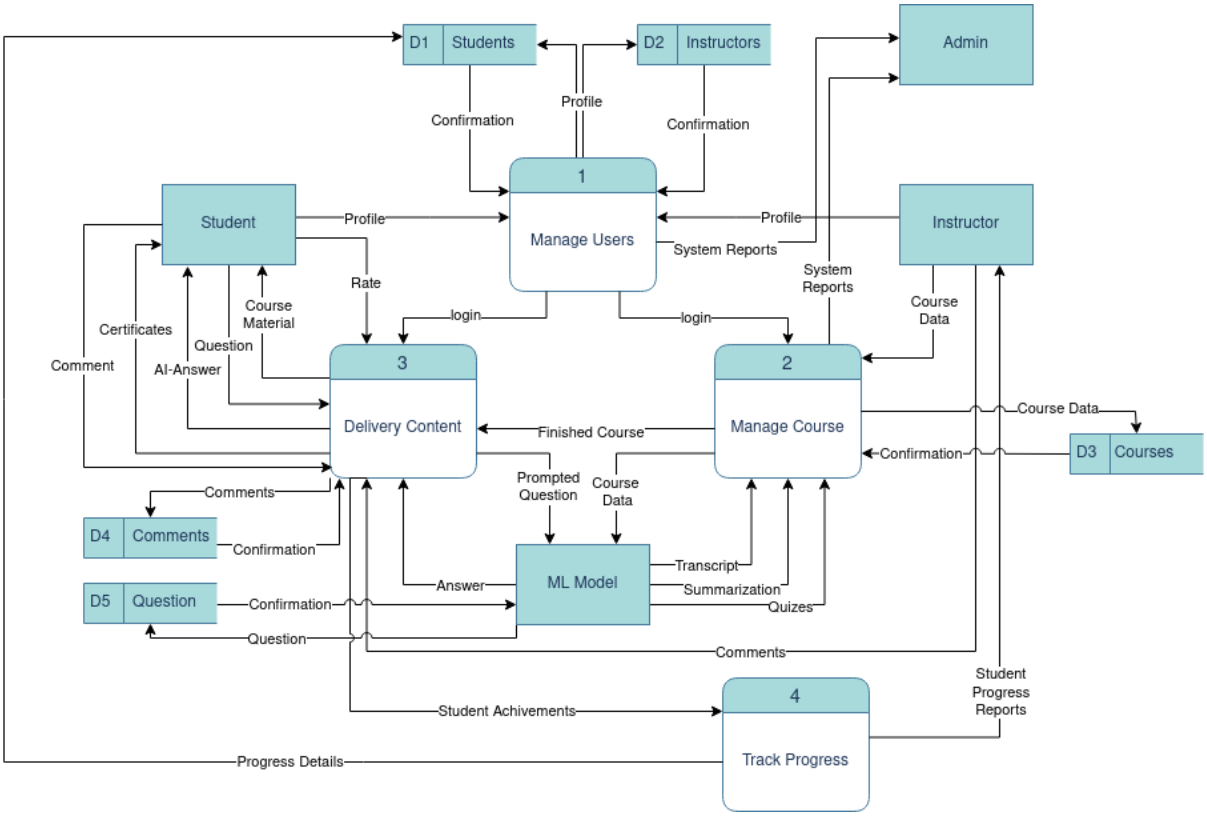


Figure 3.7: Data flow Diagram

3.8 Class Diagram

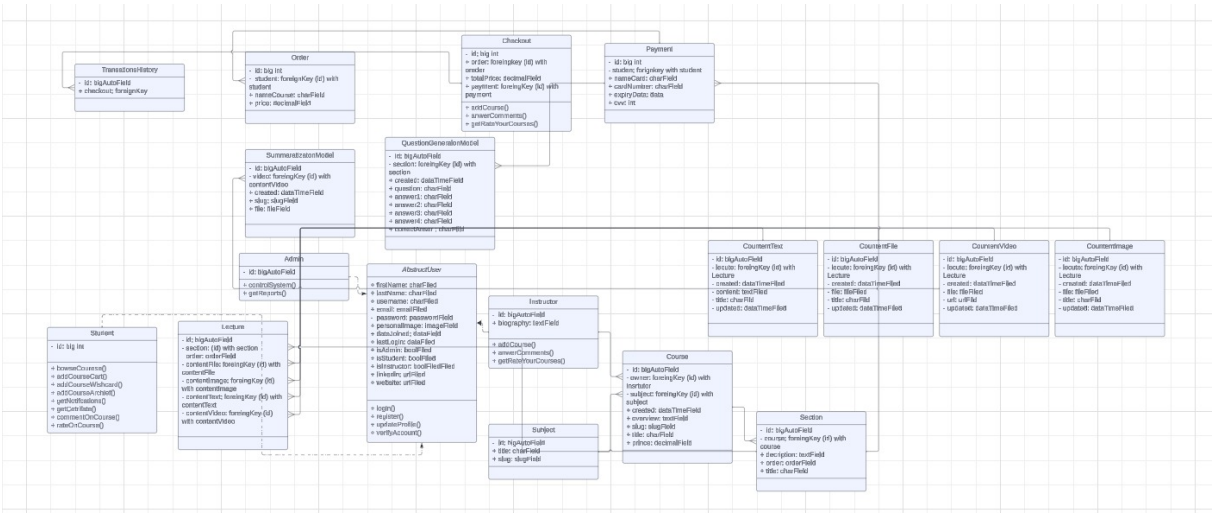
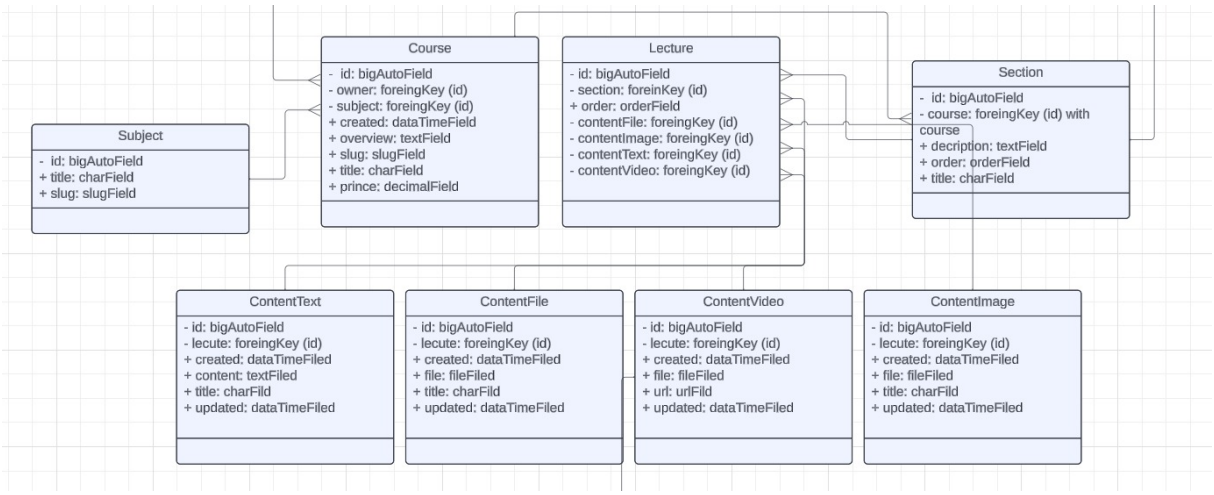
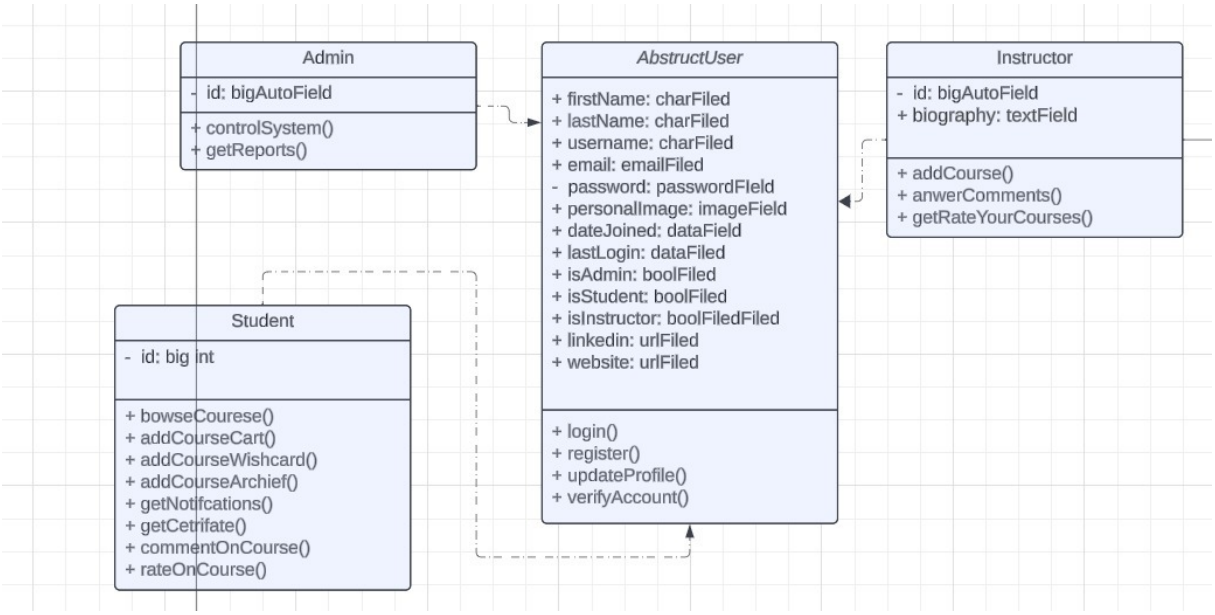
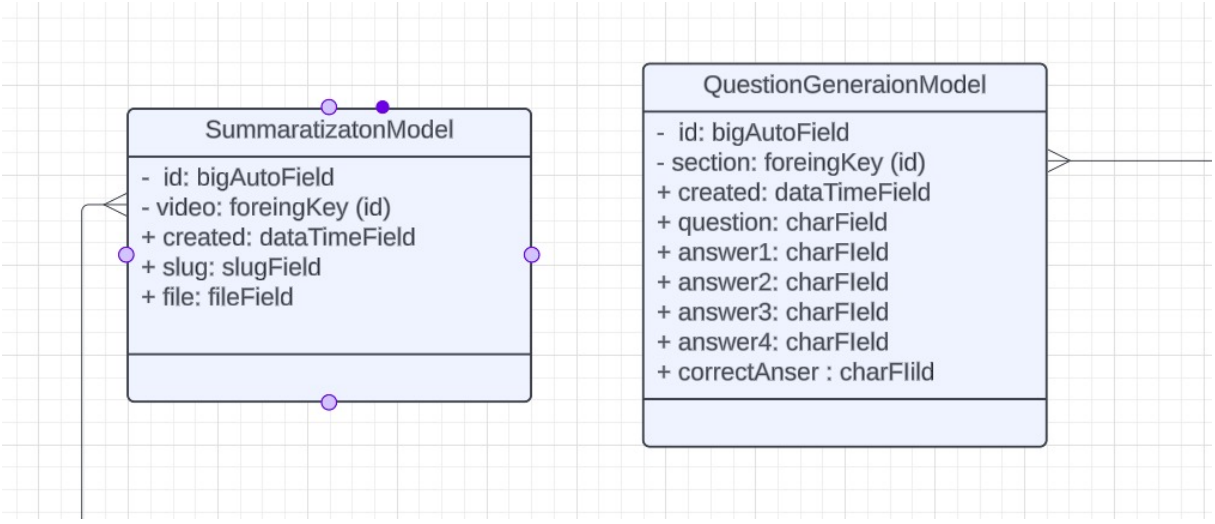
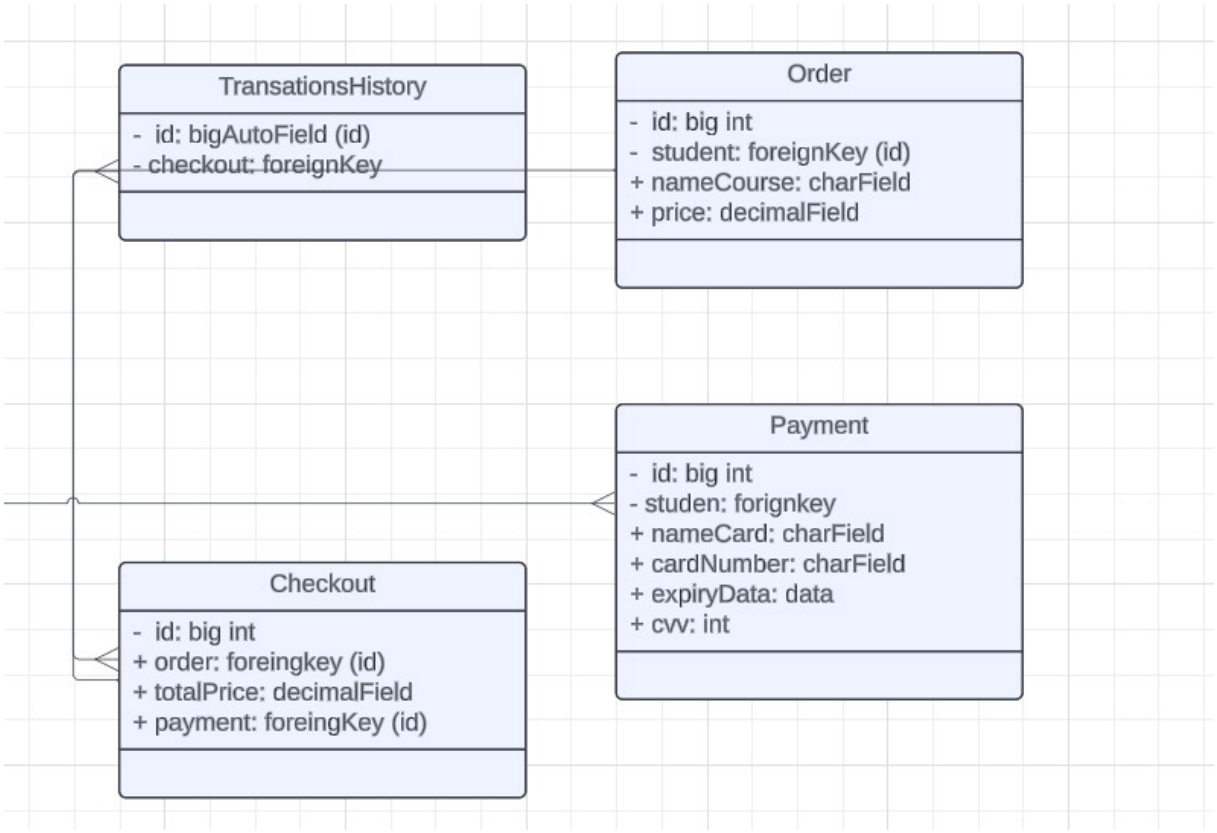


Figure 3.8: Class Diagram




Admin
- id: bigAutoField
+ controlSystem()
+ getReports()
AbstructUser
+ firstName: charFiled
+ lastName: charFiled
+ username: charFiled
+ email: emailFiled
- password: passwordFIeld
+ personalImage: imageField
+ dateJoined: dataField
+ lastLogin: dataFiled
+ isAdmin: boolFiled
+ isStudent: boolFiled
+ isInstructor: boolFiledFiled
+ linkedin: urlFiled
+ website: urlFiled
+ login()
+ register()
+ updateProfile()
+ verifyAccount()
Instructor
- id: bigAutoField
+ biography: textField
+ addCourse()
+ anwerComments()
+ getRateYourCourses()
Student
- id: big int
+ bowseCourese()
+ addCourseCart()
+ addCourseWishcard()
+ addCourseArchief()
+ getNotifcations()
+ getCetrifate()
+ commentOnCourse()
+ rateOnCourse()


SummaratizatonModel
- id: bigAutoField
- video: foreingKey (id)
+ created: dataTimeField
+ slug: slugField
+ file: fileField
QuestionGeneraionModel
- id: bigAutoField
- section: foreingKey (id)
+ created: dataTimeField
+ question: charField
+ answer1: charFIeld
+ answer2: charFIeld
+ answer3: charFIeld
+ answer4: charFIeld
+ correctAnser : charFIild


TransationsHistory
- id: bigAutoField (id)
- checkout: foreignKey
Order
- id: big int
- student: foreignKey (id)
+ nameCourse: charField
+ price: decimalField
Payment
- id: big int
- studen: forignkey
+ nameCard: charField
+ cardNumber: charField
+ expiryData: data
+ cvv: int
Checkout
- id: big int
+ order: foreingkey (id)
+ totalPrice: decimalField
+ payment: foreingKey (id)

Chapter 4

Data Collection

4.1 General Data Requirements

Creating a quiz requires meticulous data collection and a clear understanding of the requirements to ensure that the questions generated are relevant, challenging, and beneficial for users.

1. **Data Quality and Validation**

    1.1. Accuracy: Ensure the correctness of the data collected.

    1.2. Relevance: The data must be pertinent to the topics covered.

    1.3. Completeness: All necessary data points should be collected to generate comprehensive questions.

    1.4. Consistency: Standardize the format and structure of data for uniformity.

    1.5. Verification: Cross-check data with multiple sources and validate through SMEs.

4.1.1 Question generation data schema

To effectively generate and manage quiz questions, a well-defined schema is essential. Below is a detailed schema that aligns with the requirements provided. This schema includes fields for context, difficulty level, question type, the question itself, choices (for MCQs), and the answer.

Data Schema Overview

Context **Description:** The reference or source from which the question is derived. This must cover a diversity of topics to ensure a broad range of subjects.

Type: String

Level **Description:** Indicates the difficulty level of the question.

Type: Enum (hard, medium, easy)

Type **Description:** Specifies the format of the question.

Type: Enum (MCQ, Bool, Open)

Question **Description:** The actual question text.

Type: String

Choices **Description:** Possible answers for Multiple Choice Questions (MCQs).

Type: Array of Strings

Note: This field is required only if the **Type** is ”MCQ”.

Answer **Description:** The correct answer to the question.

Type: String

Example: ”To declare a block-scoped variable”

Note: The format may vary depending on the **Type** of question.

Schema Definition

Here is the schema definition in JSON format:

```
{
    "Context": {
      "type": "string",
      "description": "The reference or source from which the question is derived and Videos
transcript",
      "example": "Introduction to TypeScript..."
    },
    "Level": {
      "type": "string",
      "enum": ["easy", "medium", "hard"],
      "description": "Indicates the difficulty level of the question.",
      "example": "medium",
      "required": false
    },
    "Type": {
      "type": "string",
      "enum": ["MCQ", "Bool", "Open"],
      "description": "Specifies the format of the question.",
      "example": "MCQ",
    },
    "Question": {
      "type": "string",
      "description": "The actual question text.",
      "example": "What is the purpose of the 'let' keyword in JavaScript?"
    },
    "Choices": {
      "type": "array",
      "items": {
        "type": "string"
      },
      "description": "Possible answers for Multiple Choice Questions (MCQs).",
      "example": ["To declare a block-scoped variable",
                  "To declare a function",
                  "To declare a constant",
                  "To declare a global variable"],
      "required": false
    },
    "Answer": {
      "type": "string",
      "description": "The correct answer to the question.",
      "example": "To declare a block-scoped variable"
    }
  }
}
```

Figure 4.1: Schema of data in json format

Example Entries

Multiple Choice Question (MCQ)


```
{
  "Context": "This is the Human Papilloma Virus, which causes
  "Type": "MCQ",
  "Question": "What makes viral stis more dangerous than othe
  "Choices": [
    "A. Jim.",
    "B. Steven.",
    "C. Green.",
    "D. Tom."
  ],
  "Answer": "B"
}
```


True/False Question


```
{
  "Context": " AR-15s in California The Roberti-Roos Assault
  "Type": "bool",
  "Question": "is it legal to own an ar15 in california.",
  "Answer": "false"
}
```


Written Question


```
{
  "Context": "Dan Dierdorf, 1974 to 1978 seasons: From 1974 t
  "Type": "open",
  "Question": "What position did he play?",
  "Answer": "Dierdorf started every game at right tackle."
}
```

4.1.2 Datasets

QUAC-CQA [1] Location: Local

Type: Open

Size: 74,977 examples

Link: https://huggingface.co/datasets/tilyupo/quac_cqa?row=4

RACE (Hugging Face) Location: Hosted

Type: MCQ

Size: 195,374 examples

Description: Collected from English exams for Chinese students aged 12 to 18, covering diverse topics.

Link: https://huggingface.co/datasets/race?row=26

SQuAD (Hugging Face) [16] Location: Hosted

Type: Open

Size: 98,169 examples

Description: Collection of question-answer pairs derived from Wikipedia articles, allowing any sequence of tokens as correct answers.

Link: https://huggingface.co/datasets/squad?row=31

Boolean Questions [5](Google Research) Location: Local

Type: Boolean

Size: 15,699 examples

Link: https://github.com/google-research-datasets/boolean-questions

XQuAD (Google DeepMind) [2] Location: Local

Type: Open

Description: Subset of 240 paragraphs and 1190 QA pairs from SQuAD v1.1.

Link: https://github.com/google-deepmind/xquad

SciQ [15] (AllenAI) **Location:** Local

Type: MCQ

Size: 13,679 examples

Description: Crowdsourced science exam questions covering Physics, Chemistry, and Biology.

Link: `https://allenai.org/data/sciq`

ClarQ [12] (Vaibhav4595) **Location:** Local

Size: 4GB

Link: `https://github.com/vaibhav4595/ClarQ`

QA Schema Summary

The QA schema includes the following components:

- **Context:** The reference or source from which the question is derived.
- **Question Level:** Indicates the difficulty level (easy, medium, hard).
- **Question Type:** Specifies the format of the question (MCQ, Boolean, Open).
- **Question:** The actual question text.
- **Answer Choices:** Possible answers for MCQs.
- **Correct Answer:** The correct answer to the question.

Dataset Statistics

The datasets collectively contain a total of 398,199 examples, broken down as follows:

- **MCQ Questions:** 209,053 examples
- **Open Questions:** 173,146 examples
- **Boolean Questions:** 16,000 examples

These datasets cover a wide range of topics and were collected from various sources, providing a comprehensive collection for question generation and answering tasks.

4.2 Text Summarization Requirements

Text summarization involves specific requirements to ensure effective processing and generation of concise summaries. These requirements typically include:

4.2.1 Text Summarization Dataset Requirements

- **Size**: Datasets should be sufficiently large to train robust models. For example:
- **Quality**: Summaries should be accurate and relevant to the original texts, ensuring meaningful abstraction.
- **Diversity**: Datasets should cover a wide range of topics and domains to generalize well.

Text Summarization Datasets

CCDV Datasets

- Government Report Summarization
  - Location: hosted
  - Size: 19,463 records
  - Token Limit: 9,000 characters / 500 tokens
- PubMed Summarization
  - Location: hosted
  - Size: 266,430 records
  - Token Limit: 3,000 characters / 215 tokens
- ArXiv Summarization
  - Location: hosted
  - Size: 215,037 records
  - Token Limit: 6,000 characters / 300 tokens

Chapter 5

Modeling

5.1 Background and Literature review

5.1.1 Literature review

The literature review examines several papers published between 2017-2021 based on Automatic question generation and answer assessment: a survey [7] that address different aspects of automatic question generation and answer assessment, Here's an expanded overview:

1. **Deena et al. (2020) - "Developing the assessment questions automatically to determine the cognitive level of the E-learner using NLP techniques": [8]** This paper proposed a method for generating subjective questions dynamically using Natural Language Processing (NLP) and Bloom's taxonomy. The approach aimed to reduce memory occupation and generate questions that could assess different cognitive levels of e-learners.

2. **Das et al. (2019) - "Automatic generation of fill-in-the-blank question with corpus-based distractors for e-assessment to enhance learning": [6]** This study focused on generating fill-in-the-blank questions automatically. It used corpus-based distractors to enhance the quality of the questions for e-assessment. The method aimed to improve learning by providing more effective automatically generated questions.

3. **Klein and Nabi (2019) - "Learning to Answer by Learning to Ask: Getting the Best of GPT-2 and BERT Worlds": [11]** This paper explored the use of advanced language models, specifically GPT-2 and BERT, for question generation and answering. It aimed to combine the strengths of these two models to improve both the quality of generated questions and the accuracy of answers.

4. **Carmichael et al. (2018) - "Assessing the impact of educational video on student engagement, critical thinking and learning the Current State of Play": [4]** This paper explored the use of advanced language models, specifically GPT-2 and BERT, for question generation and answering. It aimed to combine the strengths of these two models to improve both the quality of generated questions and the accuracy of answers.

5. **Du et al. (2017) - "Learning to Ask: Neural Question Generation for Reading Comprehension":[10]** This paper presented a neural approach to question generation for reading comprehension. It used an encoder-decoder sequence learning framework to generate questions from given input text, aiming to improve the quality and relevance of automatically generated questions.

The main challenges All of these architectures are based on the servery:

- **Informative-simple-sentence extraction:**
  - Difficulty in extracting informative sentences that can generate quality questions.
  - Complexity in simplifying compound and complex sentences.
  - Lack of generic techniques for extracting informative-simple-sentences from text.
- **Question generation from multiple sentences:**
  - Challenges in generating questions that require understanding relationships between multiple sentences.
  - Need for natural language understanding to grasp the context across sentences.
- **Short and long-type answer assessment:**
  - Difficulty in automatically evaluating subjective answers reliably.
  - Challenges in assessing the depth of knowledge and critical reasoning in long answers.
  - Need for more accurate and practical systems for grading both short and long-type answers.

- **Answer assessment standard:**
  - Lack of a standardized approach to evaluate and compare learners' knowledge.
  - Variability in scoring scales and evaluation techniques across different systems.
- **Question generation and assessment from video lectures:**
  - Complexity in generating questions from audio-visual content.
  - Need for techniques to assess learning from video-based materials.
- **Question generation and assessment using machine learning:**
  - Challenges in applying advanced NLP techniques like natural language understanding (NLU) and natural language generation (NLG).
  - Need for more research in using deep learning and neural networks for question generation and answer assessment.
- **Visual Question-Answer Generation:**
  - Difficulties in generating questions from images and visual content.
  - Challenges in understanding and reasoning about visual information.
- **Automated evaluation of subjective questions:**
  - Unreliable results in computer-based assessment of subjective questions.
  - Difficulty in fully automating the evaluation of writing skills and critical reasoning.

Paper title	Published year	Challenges in Question Generation and Answer Assessment
Developing the assessment questions automatically to determine the cognitive level of the E-learner using NLP techniques.	2020	• Informative-simple-sentence extraction. • Question generation from multiple sentences. • Short and long-term answer assessment. • Answer assessment standard. • Question generation and assessment from video lectures. • Question generation and assessment using machine learning. • Visual Question-Answer Generation.
Automatic generation of fill-in-the-blank question with corpus distractors for e-assessment to enhance learning.	2019	
Learning to Answer by Learning to Ask: Getting the Best of GPT-2 and BERT Worlds.	2019	
Assessing the impact of educational video on student engagement, The Current State of Play.	2018	
Learning to Ask: Neural Question Generation for Reading Comprehension.	2017	
Our Proposed Model		Advantages and Disadvantages of our proposed mode
Proposed model (NeuraLearn Academy)	2024	Advantages: • Question generation and assessment from video lectures. • Answer assessment standard. • Question generation three types of questions (MCQ-True/False-Written). • Short and long-term answer assessment. • Question generation from multiple sentences. • More generalized (can generate meaningful questions from a different domain). Disadvantages: • Token generating Latency. • Model generation affected by the quality of the transcript.

Figure 5.1: Literature review

Advantages of our proposed model:

- **Question generation and assessment from video lectures:**
  - By using video transcripts for question generation and assessment from video lectures.
- **Answer assessment standard.**
  - By using An instruction fine-tuning technique.
- **Question generation three types of questions (MCQ-True/False-Written).**
- **Short and long-type answer assessment.**
- **Question generation from multiple sentences.**
- **More generalize (can generate meaningful questions from a different domain).**

Disadvantages of our proposed model:

- **Token generating Latency.**
- **Model generation affected by the quality of the transcript.**

5.1.2 Model architecture

The Transformer architecture, introduced by Vaswani et al. in 2017 [17], has become a pivotal model in natural language processing tasks.

Encoder:
The Encoder processes an input sequence using multi-head self-attention and position-wise feed-forward networks.

- **Multi-Head Self-Attention:** This mechanism allows each word in the input sequence to attend to all other words simultaneously, capturing dependencies and relationships effectively.
- **Position-wise Feed-Forward Networks:** After attention computation, a position-wise fully connected feed-forward network is applied independently to each position.

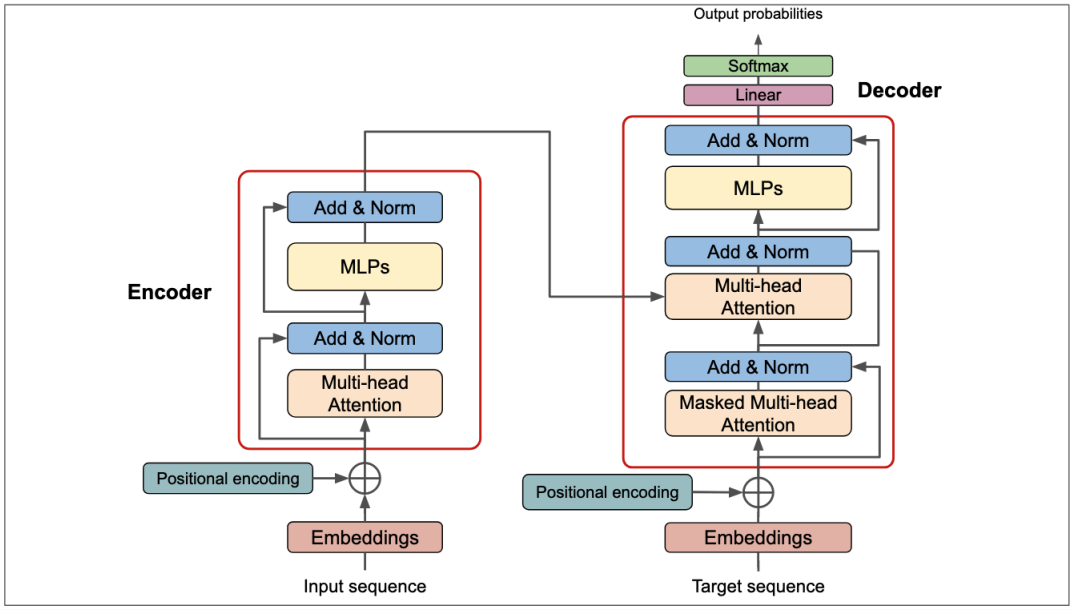


Figure 5.2: Transformer Architecture

Decoder:
The Decoder generates an output sequence by attending to the Encoder's output and performing masked self-attention and cross-attention over the input sequence.

- **Masked Multi-Head Self-Attention:** During training, masking is applied to prevent future positions from being used, allowing each word to attend only to previous positions.

- **Multi-Head Cross-Attention:** This sub-layer attends to the encoded input sequence, helping the decoder focus on relevant parts of the input sequence when generating each word in the output sequence.
- **Position-wise Feed-Forward Networks:** A position-wise fully connected feed-forward network is applied to each position after attention computation.

Key Innovations:

- **Self-Attention Mechanism:** Enables capturing dependencies across the entire input sequence.
- **Positional Encoding:** Provides information about the order of tokens in input sequences.
- **Residual Connections and Layer Normalization:** Address vanishing gradient problem and improve convergence speed in deep models.

5.1.3 LLMs

(LLMs)

LLMs such as GPT (Generative Pre-trained Transformer) are pivotal in natural language processing. These models are trained on large-scale datasets using unsupervised learning techniques before fine-tuning on specific tasks.

Training Process:

LLMs are typically pre-trained using a variant of the Transformer architecture. The training process involves:

- **Pre-training on Large Corpora:** LLMs are trained on vast amounts of text data to learn general language patterns and representations. This phase helps the model capture semantic relationships and syntactic structures.
- **Objective Functions:** During pre-training, objectives like language modeling (predicting the next word in a sequence) and masked language modeling (predicting masked-out words) are used to teach the model about language coherence and context understanding.
- **Fine-tuning:** After pre-training, LLMs can be fine-tuned on smaller, task-specific datasets to adapt to specific applications such as question answering, summarization, or language translation.

Key Details:

LLMs leverage the Transformer's self-attention mechanism, allowing them to process and generate text while considering global dependencies efficiently. Key aspects include:

- **Self-Attention:** Enables the model to weigh the significance of each word in the context of the entire input sequence.
- **Positional Encodings:** Provides information about the order of tokens in input sequences, compensating for the lack of sequential information in the Transformer's architecture.
- **Fine-grained Representations:** LLMs produce contextual embeddings that capture intricate linguistic nuances, making them versatile for various natural language processing tasks.

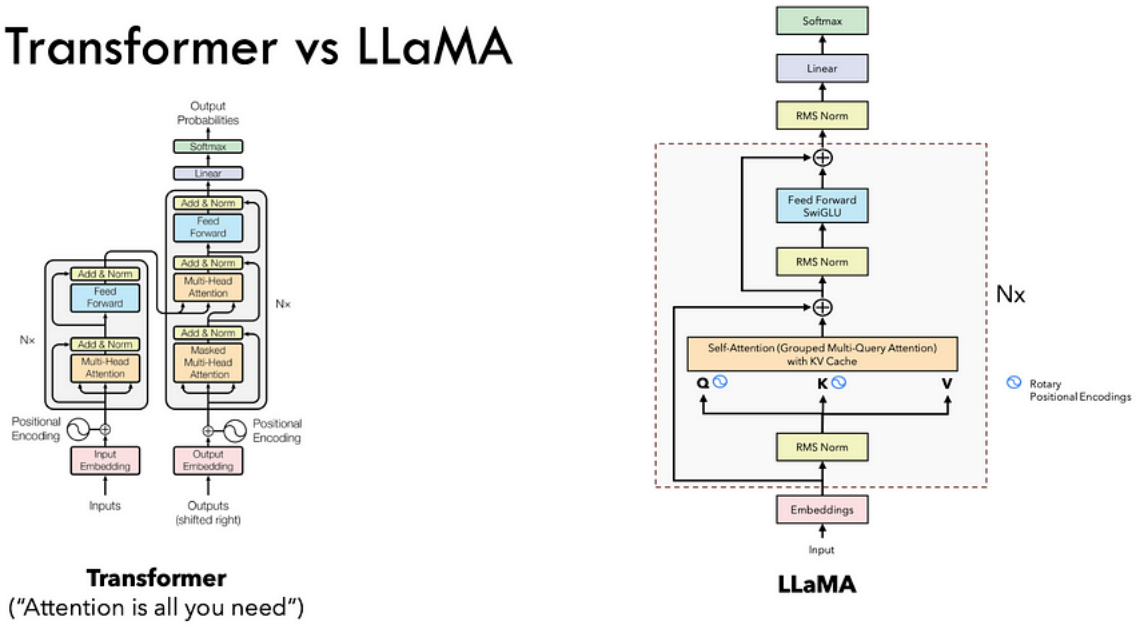


Figure 5.3: Transformer vs LLAMA

Challenges of Training Large Language Models (LLMs)

Training Large Language Models (LLMs) such as GPT (Generative Pre-trained Transformer) poses several challenges due to the scale and complexity of the models involved:

- **Computational Resources:** LLMs require substantial computational resources, including high-performance GPUs or TPUs, to handle the massive amounts of data and computations involved in training.
- **Data Efficiency:** Despite their size, LLMs still require vast amounts of labeled or unlabeled data for effective training. Acquiring and preprocessing such large datasets can be challenging.
- **Training Time:** Training LLMs is time-consuming, often taking days or even weeks on powerful hardware. This extended duration can hinder rapid experimentation and model iteration.
- **Fine-tuning Challenges:** While pre-training is effective, fine-tuning LLMs for specific tasks requires careful tuning of hyperparameters and regularization techniques to prevent overfitting and ensure optimal performance.
- **Memory and Scalability:** Managing memory usage and ensuring scalability across distributed systems are critical challenges, especially as models grow larger and more complex.
- **Ethical Considerations:** The sheer size and capability of LLMs raise

ethical concerns related to bias, fairness, and the potential misuse of generated content.

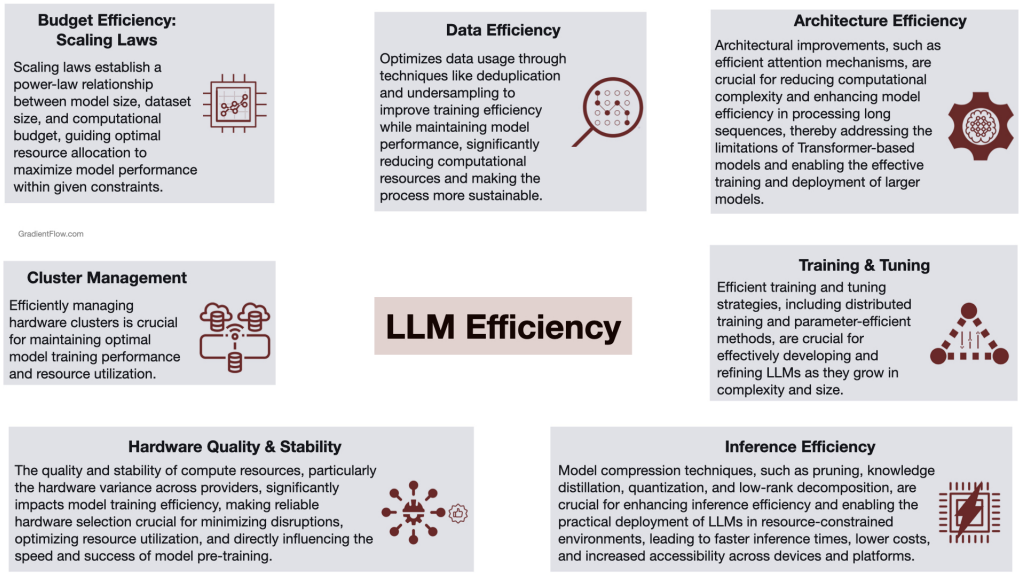


Figure 5.4: challenges of training LLMs

5.1.4 Instruction Fine-tuning

What is instruction tuning?

Instruction tuning is a subset of the broader category of fine-tuning techniques used to adapt pre-trained foundation models for downstream tasks. Foundation models can be fine-tuned for a variety of purposes, from style customization to supplementing the core knowledge and vocabulary of the pre-trained model to optimizing performance for a specific use case. Though fine-tuning is not exclusive to any specific domain or artificial intelligence model architecture, it has become an integral part of the LLM lifecycle. For example, Meta's Llama 2 model family is offered (in multiple sizes) as a base model, as a variant fine-tuned for dialogue (Llama-2-chat) and as a variant fine-tuned for coding (Code Llama).

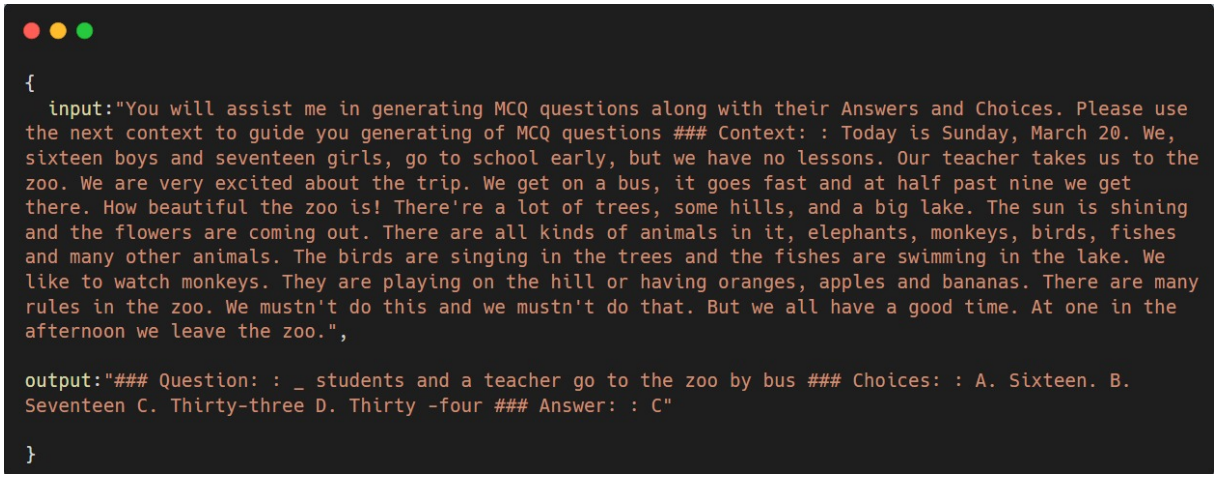
Instruction tuning is not mutually exclusive with other fine-tuning techniques. For example, chat models often undergo both instruction tuning and reinforcement learning from human feedback (RLHF), a fine-tuning technique that aims to improve abstract qualities like helpfulness and honesty; models fine-tuned for coding often undergo both instruction tuning (to broadly optimize responses for instruction following) and additional fine-tuning on programming-specific data (to augment the model's knowledge of coding syntax and vocabulary).

Why instruction tune LLMs?

The pre-training process for autoregressive language models—LLMs used for generating text, like Meta's Llama 2, OpenAI's GPT, Google's Gemini or IBM's Granite—optimizes these LLMs to simply predict the next word(s) in a given sequence until it's complete.

LLMs are pre-trained using self-supervised learning on a massive corpus of written content. In pre-training, autoregressive models are provided the beginning of a text sample and repeatedly tasked with predicting the next word in the sequence until the end of the excerpt. For each prediction, the actual next word of the original sample sentence serves as "ground truth." Through optimization algorithms like gradient descent that iteratively adjust model parameters—the varying weights and biases applied to the mathematical operations occurring at each node in a neural network—in a way that brings the model's predictions closer to the original text, the model "learns" the linguistic patterns in its training data (and, by extension, the "knowledge" conveyed in those linguistic patterns).

Example of instruction promote



```
{
  input:"You will assist me in generating MCQ questions along with their Answers and Choices. Please use
the next context to guide you generating of MCQ questions ### Context: : Today is Sunday, March 20. We,
sixteen boys and seventeen girls, go to school early, but we have no lessons. Our teacher takes us to the
zoo. We are very excited about the trip. We get on a bus, it goes fast and at half past nine we get
there. How beautiful the zoo is! There're a lot of trees, some hills, and a big lake. The sun is shining
and the flowers are coming out. There are all kinds of animals in it, elephants, monkeys, birds, fishes
and many other animals. The birds are singing in the trees and the fishes are swimming in the lake. We
like to watch monkeys. They are playing on the hill or having oranges, apples and bananas. There are many
rules in the zoo. We mustn't do this and we mustn't do that. But we all have a good time. At one in the
afternoon we leave the zoo.",

output:"### Question: : _ students and a teacher go to the zoo by bus ### Choices: : A. Sixteen. B.
Seventeen C. Thirty-three D. Thirty -four ### Answer: : C"

}
```

Figure 5.5: Example of instruction promote

Here is the link to the Dataset repository: `https://huggingface.co/datasets/shredder-31/QG_QusestionsData`

5.1.5 QLORA: Efficient Finetuning of Quantized LLMs

QLORA: Efficient Finetuning of Quantized LLMs

"QLORA: Efficient Finetuning of Quantized LLMs" [9] is a research paper that focuses on optimizing the fine-tuning process for quantized large language models (LLMs). Here's a detailed explanation of the key concepts and contributions of QLORA:

Large Language Models (LLMs) like GPT (Generative Pre-trained Transformer) have achieved significant success in various natural language processing (NLP) tasks. However, these models are computationally intensive and memory-intensive, making them challenging to deploy in resource-constrained environments like mobile devices or edge computing platforms.

Quantization is a technique used to address these challenges. It involves reducing the precision of model parameters (e.g., from 32-bit floating point to 8-bit integers) to reduce memory usage and improve inference speed. However, quantization often leads to a drop in performance unless the model is fine-tuned properly.

Key Challenges Addressed by QLORA

1. **Efficient Fine-tuning**: Fine-tuning a quantized LLM involves adjusting the model weights to perform well on a specific downstream task while maintaining the benefits of quantization. This process is crucial because quantization can degrade model accuracy if not managed properly.

2. **Optimal Hyperparameter Search**: Finding the right hyperparameters (e.g., learning rate, batch size) for fine-tuning quantized models is non-trivial due to the interaction between quantization and model training dynamics.

Contributions of QLORA

QLORA proposes several techniques to improve the efficiency and effectiveness of fine-tuning quantized LLMs:

1. **Layer-wise Quantization-aware Optimization**: QLORA introduces a method to optimize the learning rate and weight decay for each layer of the quantized model separately. This approach accounts for the different sensitivities of layers to quantization errors, improving overall performance.

2. **Dynamic Range Scaling**: QLORA employs dynamic range scaling dur-

ing fine-tuning, which adjusts the quantization parameters (such as scale and zero-point) based on the statistics of the input data. This adaptive scaling helps mitigate the impact of quantization on model accuracy.

3. **Efficient Search for Hyperparameters**: QLORA incorporates techniques for efficiently searching the hyperparameter space, such as Bayesian optimization or grid search tailored for quantized models. This ensures that the fine-tuning process is both effective and resource-efficient.

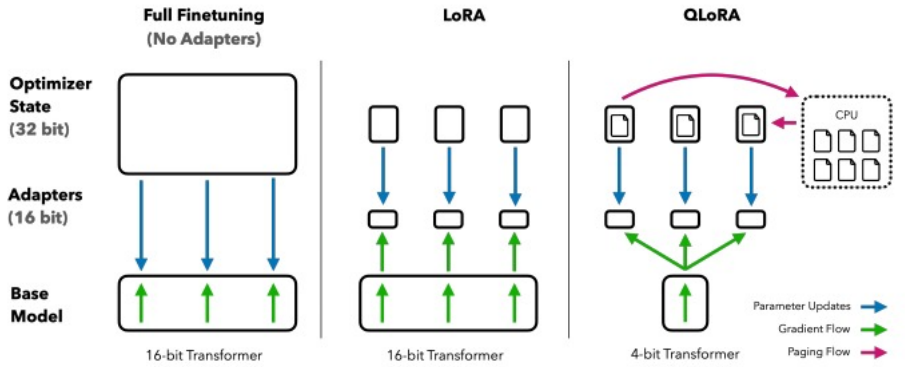


Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.

Figure 5.6: Qlora

5.1.6 RAG: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Retrieval-Augmented Generation (RAG) represents a significant advancement in natural language processing [13], particularly for knowledge-intensive tasks. This approach integrates both retrieval-based and generation-based models to leverage external knowledge sources effectively.

The workflow of Retrieval-Augmented Generation typically involves the following steps:

- **Retrieval Phase:**
  - *Query Formation:* The input query or prompt is formulated.
  - *Retrieval:* A retrieval model searches through external knowledge sources (like Wikipedia, databases, or custom corpora) to extract relevant passages or documents related to the query.
- **Integration Phase:**
  - *Candidate Selection:* The retrieved documents or passages are filtered to select the most relevant ones based on similarity metrics or other criteria.
  - *Context Integration:* The selected documents are integrated into the context of the generation model. This can involve concatenating retrieved text with the original input or dynamically attending to relevant parts during generation.
- **Generation Phase:**
  - *Text Generation:* The integrated context is fed into a generation model (often a transformer-based model) to produce a coherent and contextually relevant output.
  - *Fine-Tuning:* The model may be fine-tuned on task-specific data to improve performance on particular NLP tasks, ensuring the generated outputs are accurate and contextually appropriate.

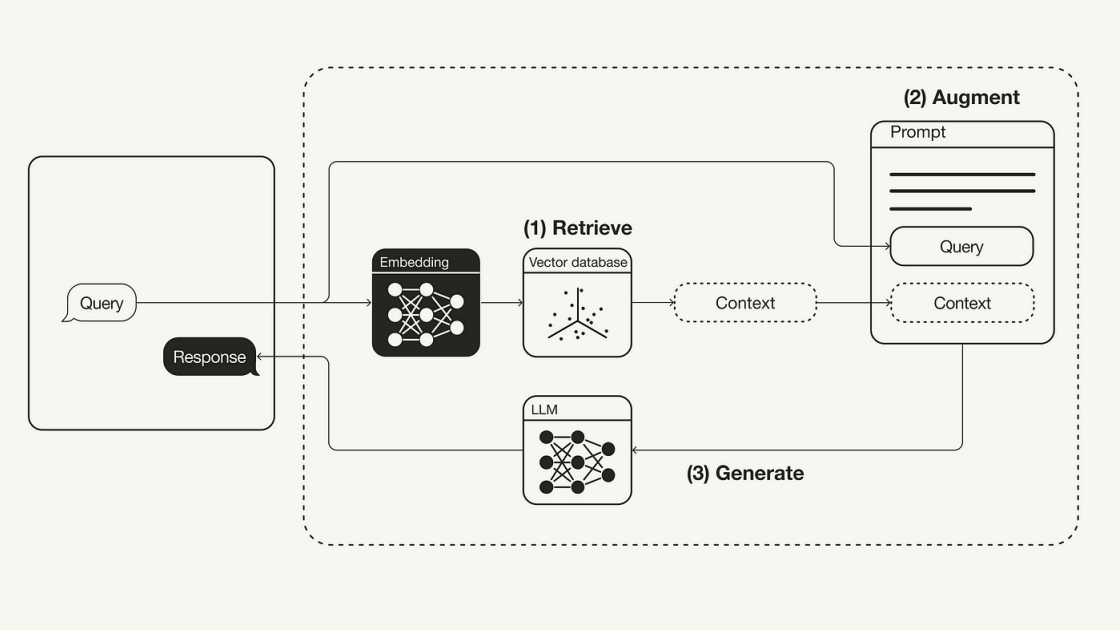


Figure 5.7: Retrieval Augmented Generation

Retrieval-Augmented Generation is particularly beneficial for knowledge-intensive tasks where accurate and comprehensive knowledge retrieval is crucial. Some notable applications include:

- **Question Answering:** Enhancing the ability to answer complex questions by leveraging a broader range of external knowledge.
- **Dialogue Systems:** Improving the responses of chatbots by integrating real-time information retrieval.
- **Summarization:** Generating concise summaries enriched with up-to-date information from external sources.
- **Content Creation:** Facilitating the creation of informative and well-researched content by incorporating diverse perspectives and facts.

The advantages of Retrieval-Augmented Generation include:

- *Accuracy:* Integrating external knowledge sources helps in providing more accurate and relevant responses.
- *Coverage:* It expands the scope of information that can be accessed beyond what is inherently encoded in the model.
- *Flexibility:* The approach is flexible and adaptable to different domains and tasks by selecting appropriate retrieval sources.

However, there are challenges associated with RAG:

- *Efficiency:* Retrieval can be computationally expensive, especially with large knowledge bases.
- *Integration Complexity:* Harmonizing retrieved knowledge with the generation model output without losing coherence can be challenging.
- *Evaluation:* Assessing the quality of generated outputs that depend on retrieved knowledge poses evaluation challenges.

To understand the technical architecture of RAG models, consider the following components:

- **Retriever Model:**
  - *Indexing:* The retriever model first indexes a large corpus of documents or knowledge base.
  - *Query Encoding:* When a query is input, the retriever encodes it into a dense vector representation.
  - *Similarity Search:* The encoded query is compared with the indexed document vectors to retrieve the most relevant documents.
- **Generator Model:**
  - *Contextual Embeddings:* The retrieved documents are transformed into contextual embeddings, which are combined with the query embeddings.
  - *Attention Mechanisms:* Advanced attention mechanisms allow the generator to attend to relevant parts of the retrieved documents while generating the response.
  - *Sequence Generation:* Using transformer-based architectures, the generator produces the output text, conditioned on the combined context of the query and retrieved documents.

An example workflow for a knowledge-intensive task, such as answering a detailed historical question, might involve:

- **Input Query:** "What were the key factors that led to the fall of the Roman Empire?"
- **Retrieval Phase:**
  - *Query Encoding:* Encode the query using a model like BERT.
  - *Similarity Search:* Retrieve relevant passages from a historical database or Wikipedia.
  - *Selected Passages:* Identify key documents discussing economic troubles, military losses, and political corruption.
- **Integration Phase:**
  - *Contextual Integration:* Integrate the retrieved passages into the generation context.
- **Generation Phase:**
  - *Response Generation:* Generate a comprehensive answer that weaves together information from the retrieved passages, highlighting economic, military, and political factors.
- **Scalability:** Developing more efficient retrieval algorithms and data structures to handle ever-larger knowledge bases.
- **Personalization:** Tailoring retrieval and generation processes to individual user preferences and contexts.
- **Multimodal Retrieval:** Extending RAG to handle multimodal data, incorporating text, images, and other media types.
- **Evaluation Metrics:** Creating better evaluation metrics to assess the factual accuracy, relevance, and coherence of generated content.

Practical applications and case studies of RAG include:

- **Chat-bot support:**
  - *Scenario:* A support chatbot for a tech company.
  - *Implementation:* Integrate a knowledge base of product manuals and troubleshooting guides. Use RAG to retrieve relevant sections and generate tailored responses to customer queries.
- **Educational Tools:**
  - *Scenario:* An interactive learning platform for history students.
  - *Implementation:* Access a large repository of historical texts. RAG retrieves pertinent excerpts and generates explanatory content or answers to student questions.

Here is the link to the RAG Pipline repository: `https://github.com/Sh-31/NeuraLearn-ML-Server-QG-RAG-Pipeline`

5.2 Training and Evaluation

5.2.1 Question Generation

There was a several implementation for question generation model I will present the final model and do comparison of variance architecture and number of parameter in evaluation section.

Training Arguments:

```
trainer = SFTTrainer(
    model = model,
    tokenizer = tokenizer,
    train_dataset = train_dataset,
    eval_dataset = vaild_dataset,
    dataset_text_field = "text",
    max_seq_length = max_seq_length,
    dataset_num_proc = 2,
    packing = False,
    args = TrainingArguments(
        per_device_train_batch_size = 4,
        per_device_eval_batch_size=   2,
        gradient_accumulation_steps = 4,
        warmup_steps = 5,
        max_steps = -1,
        num_train_epochs=1,
        learning_rate = 2e-4,
        fp16 = not torch.cuda.is_bf16_supported(),
        bf16 = torch.cuda.is_bf16_supported(),
        logging_steps = 10,
        optim = "adamw_8bit",
        weight_decay = 0.01,
        lr_scheduler_type = "linear",
        seed = 3407,
        output_dir = "outputs",
        report_to = "tensorboard",
    ),
)
```

Figure 5.8: Question Generation Training Arguments

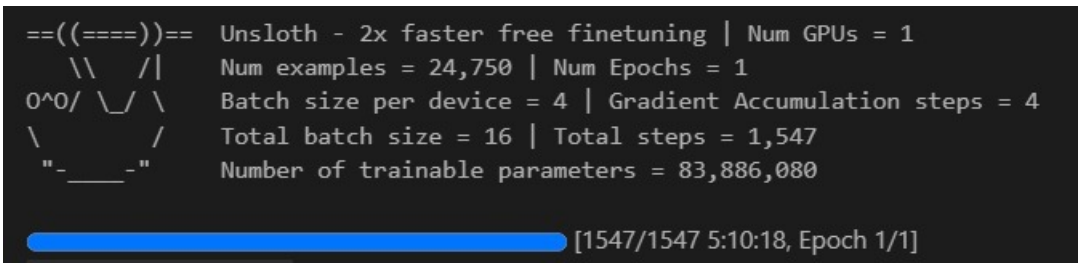


Figure 5.9: Number of parameter

Training loss vs steps :

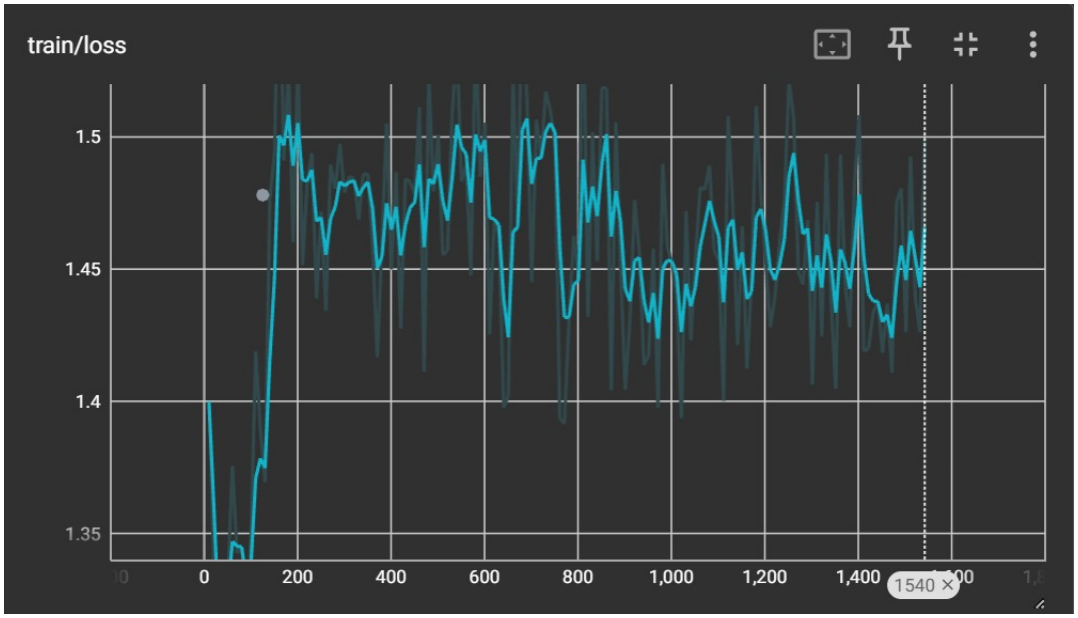


Figure 5.10: Question Generation Training Loss vs Steps

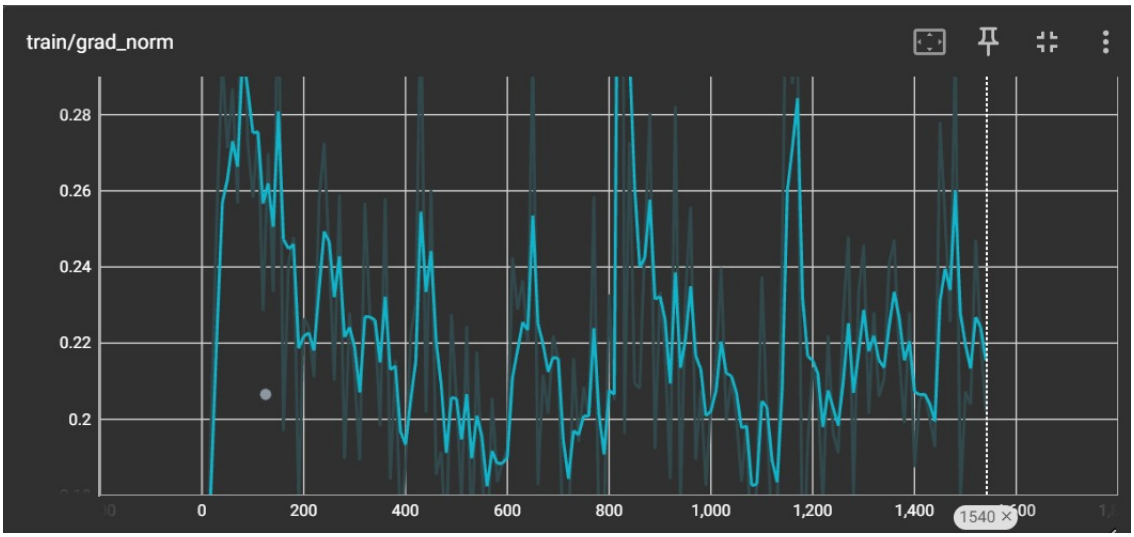


Figure 5.11: Question Generation Training grad norm vs Steps

Metrics evaluation:

How to evaluation:

Based on ”Are Large Language Models Fit For Guided Reading?” Paper [14] looks at the ability of large language models to participate in educational guided reading. We specifically, evaluate their ability to generate meaningful questions from the input text, generate diverse questions.

It’s suggest use ROUGE-L and BERTScore as evaluation metric.

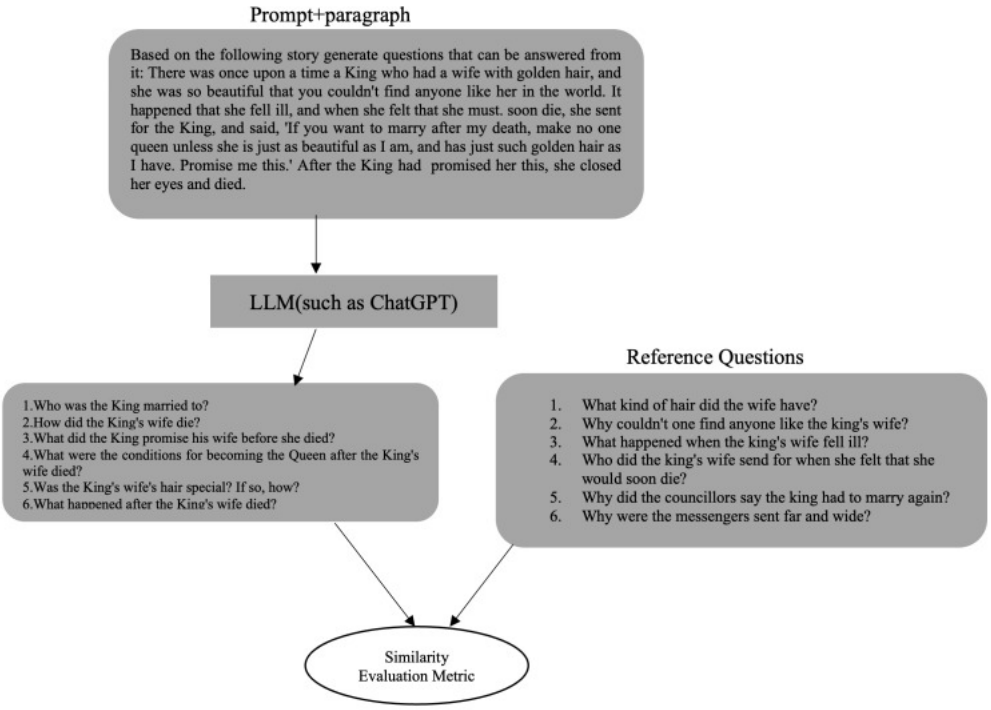


Figure 5.12: Prompting LLM to generate questions from a given text input

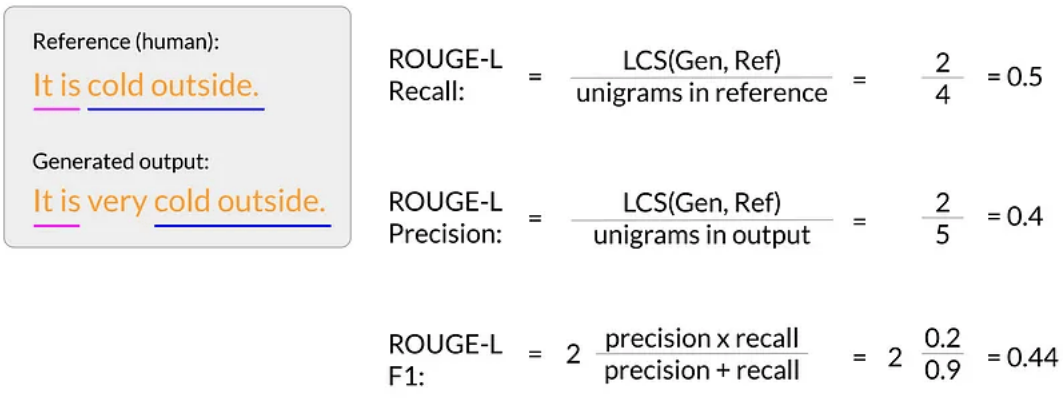


Figure 5.13: ROUGE-L

Model	Adapter Rank	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-Lsum
LLaMA-3 8B	32	0.4024	0.1956	0.3790	0.3381
LLaMA-3 8B	16	0.3637	0.1025	0.2997	0.3546
Gemma 2B	32	0.3642	0.0998	0.3068	0.3536
Gemma 2B	16	0.3003	0.2111	0.2368	0.3234

Table 5.1: Comparison of ROUGE scores for different models and adapter ranks

What's a good ROUGE score?:

A good ROUGE score varies by task and metric. ROUGE-1 scores are excellent around 0.5, with scores above 0.5 considered good and 0.4 to 0.5 moderate. For ROUGE-2, scores above 0.4 are good, and 0.2 to 0.4 are moderate.

ROUGE-L scores are good around 0.4 and low at 0.3 to 0.4. While ROUGE scores are useful, they don't account for semantic or syntactic quality and should be complemented with other metrics and human evaluation for a complete assessment.

Important Note: we didn't use any type of beam search algorithm to enhance the result because of competition power resource is limited.

ROUGE Metric	Excellent	Good	Moderate
ROUGE-1	0.5+	>0.5	0.4-0.5
ROUGE-2	-	>0.4	0.2-0.4
ROUGE-L	-	~0.4	0.3-0.4

Figure 5.14: What's a good ROUGE score?

```python
model = peftconfig.get_peft_model(
    model,
    r = 32,
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 32,
    lora_dropout = 0,
    bias = "none",
    use_gradient_checkpointing = "unsloth",
    random_state = 3407,
)
```

Figure 5.15: Adapter config rank 32

```python
model = peftconfig.get_peft_model(
    model,
    r = 16,
    target_modules = ["q_proj", "k_proj", "v_proj", "o_proj",
                      "gate_proj", "up_proj", "down_proj",],
    lora_alpha = 16,
    lora_dropout = 0,
    bias = "none",
    use_gradient_checkpointing = "unsloth",
    random_state = 3407,
    use_rslora = False,
    loftq_config = None,
)
```

Figure 5.16: Adapter config rank 16

Here is the link to the model repository: `https://huggingface.co/shredder-31/Llamma-3_QG_V.3.0`

5.2.2 Text summarization

Our text summarization model is based of [3] longformer is a variant of the transformer architecture designed to handle long-range dependencies more effectively than traditional Transformers, which typically have a quadratic dependency on the sequence length due to their self-attention mechanism.

Training Arguments:

```
batch_size = 2

training_args = Seq2SeqTrainingArguments(
    predict_with_generate=True,
    training_strategy="steps",
    per_device_train_batch_size=batch_size,
    fp16=True,
    fp16_backend="apex",
    learning_rate = 2e-3,
    output_dir="./",
    logging_steps=250,
    save_steps=500,
    warmup_steps=1500,
    save_total_limit=2,
    gradient_accumulation_steps=4,
)
```

Figure 5.17: Summarizing Training Arguments

Training loss vs steps :

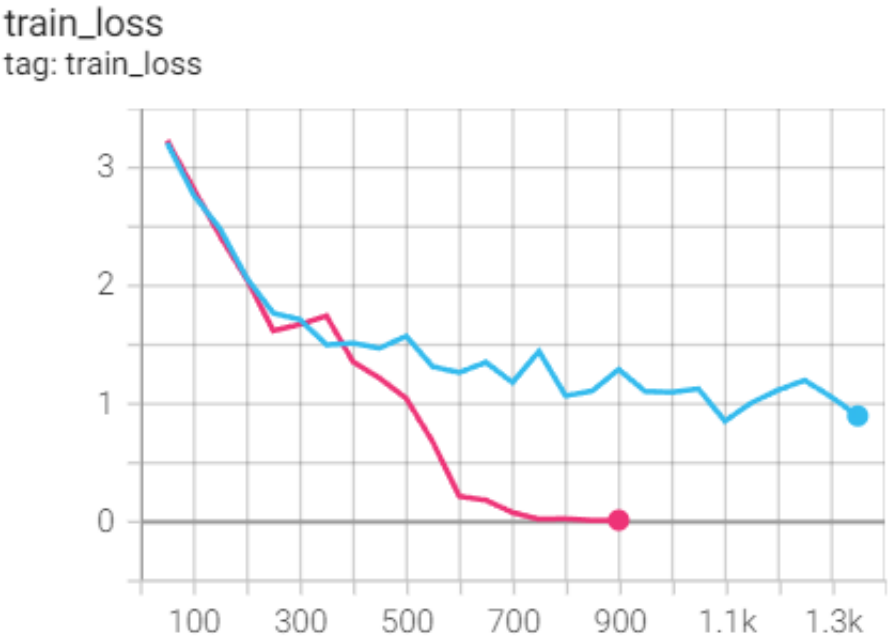


Figure 5.18: Summarizing Training Loss vs Steps

Val loss vs steps :

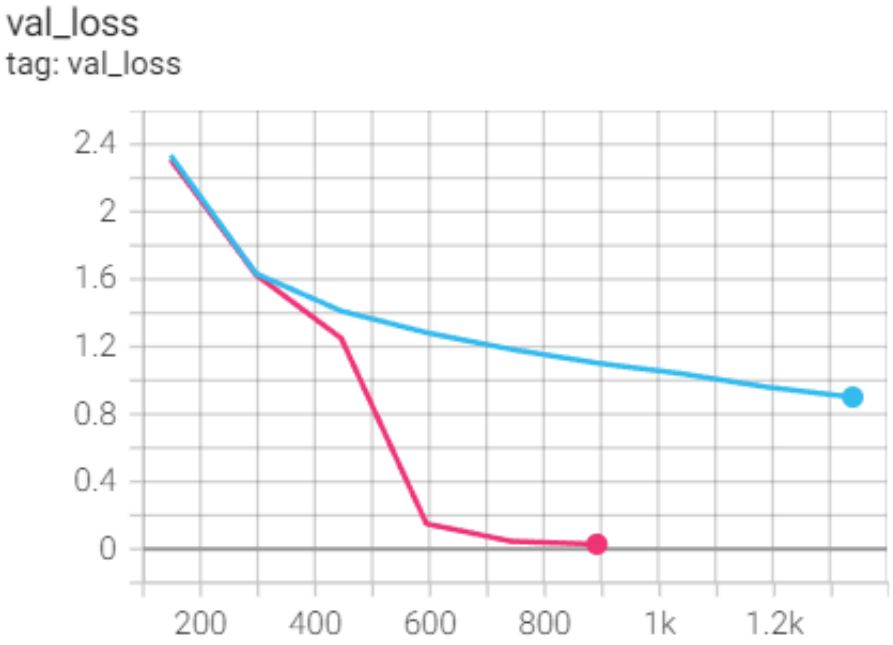


Figure 5.19: Summarizing Validation Loss vs Steps

Metrics evaluation:

Metric	Value
ROUGE-1	0.32
ROUGE-2	0.6
ROUGE-L	0.16
ROUGE-LSUM	0.29

Table 5.2: Performance Metrics on booksum benchmark

Metric	Value
ROUGE-1	0.30
ROUGE-2	0.13
ROUGE-L	0.19
ROUGE-LSUM	0.28

Table 5.3: Performance Metrics on samsum dataset

Metric	Value
ROUGE-1	0.36
ROUGE-2	0.15
ROUGE-L	0.23
ROUGE-LSUM	0.30

Table 5.4: Performance Metrics on billsum dataset

Here is the link to the model repository: `https://huggingface.co/shredder-31/Summarization_Model_led_base_book_summary`

Chapter 6

System Designing

6.1 System Requirments to activate the system

The system architecture is divided into three main components:

- **Frontend:** The frontend is developed using React. It provides an interactive user interface for end-users to interact with the system.
- **Backend:** The backend is built with Django, a high-level Python web framework. It handles the application logic, user authentication, and serves as an intermediary between the frontend and the machine learning server.
- **Machine Learning:** The machine learning server is implemented using FastAPI. It processes and analyzes data, and exposes machine learning models via RESTful APIs.

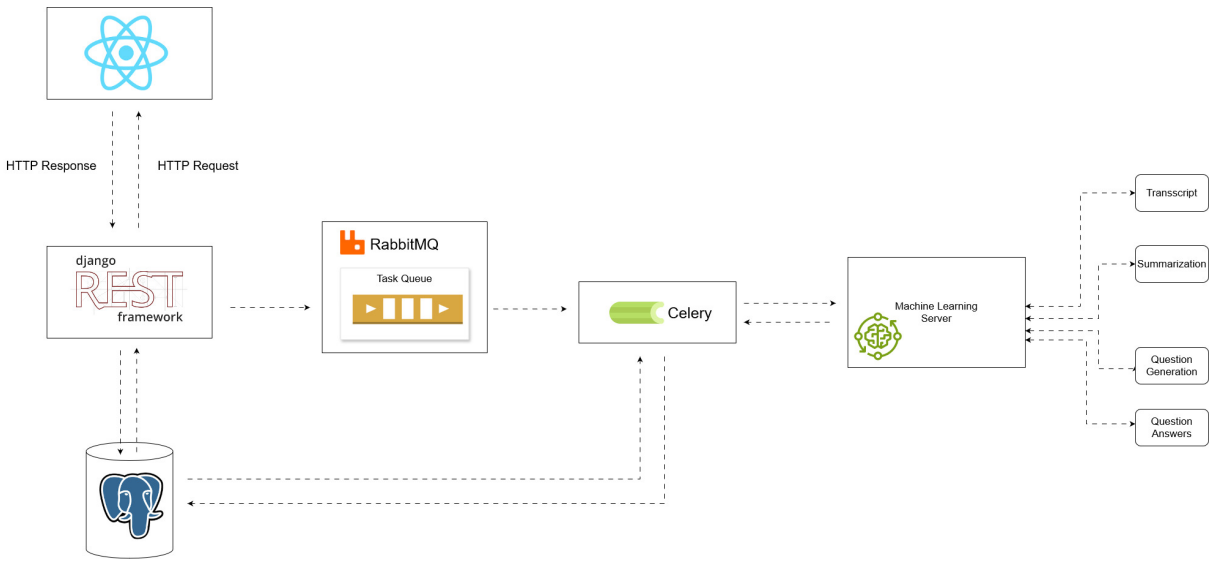


Figure 6.1: System Architecture

6.1.1 Software Requirements

- **Node.js and npm:** Node.js and npm should be installed. They are required for building and running the React application.
- **React Build:** The React application must be built using the command `npm run build` to generate static files for deployment.
- **Python and Django:** Python and Django should be installed. The Django application must be properly configured and tested.
- **Database:** Set up the database (PostgreSQL) and should be configured correctly in the Django settings.
- **FastAPI:** FastAPI should be installed for serving ML models. The machine learning server must be properly configured and tested.
- **API Endpoints:** API endpoints should functioning correctly and are accessible by the backend.

6.1.2 Hardware Requirements

There are two sets of hardware requirements for the backend/frontend and the machine learning server.

- **Backend and Frontend Server:**
  - **Disk Space:** 1 TB (SSD class disk recommended)
  - **Memory:** 4 GB
  - **Processor:** 2 cores
- **Machine Learning Server:**
  - **Disk Space:** 1 TB (SSD class disk recommended)
  - **Memory:** 16 GB
  - **Processor:** 16 cores
  - **GPU:** 24 GB

6.2 Code Screenshots Description

6.2.1 Machine Learning

The implementation and code description of machine learning models is described in Chapter 5

6.2.2 Main Backend Funtions

```python
@extend_schema(tags=['Courses'])
class CourseListAPIView(CourseOwnerMixin, generics.ListAPIView):
    queryset = Course.objects.all()
    serializer_class = ManageCourseSerializer
    permission_classes = [IsInstructorPermission]

@extend_schema(tags=['Courses'])
class CourseCreateAPIView(generics.CreateAPIView):
    queryset = Course.objects.all()
    serializer_class = CourseSerializer
    permission_classes = [IsInstructorPermission]

    def perform_create(self, serializer):
        serializer.save(owner=self.request.user)

@extend_schema(tags=['Courses'])
class CoursePublishView(generics.GenericAPIView):
    permission_classes = [IsInstructorPermission]

    def put(self, request, slug, format=None):
        user = self.request.user
        course = get_object_or_404(Course, slug=slug, owner=user)
        if course.available:
            return Response({"Error": "Course is already published"},
                            status=status.HTTP_400_BAD_REQUEST)
        course.available = True
        course.save()
        serializer = CourseAvailableSerializer(course)
        return Response(serializer.data)
```

Figure 6.2: Course Creationg and publishing

```python

@extend_schema(tags=['Courses'])
class CourseDetailAPIView(CourseOwnerMixin, generics.RetrieveAPIView):
    queryset = Course.objects.all()
    serializer_class = CourseDetailSerializer
    permission_classes = [permissions.IsAdminUser, IsInstructorPermission]
    lookup_field = 'slug'

@extend_schema(tags=['Courses'])
class CourseUpdateAPIView(CourseOwnerMixin, generics.UpdateAPIView):
    queryset = Course.objects.all()
    serializer_class = CourseSerializer
    permission_classes = [permissions.IsAdminUser, IsInstructorPermission]
    lookup_field = 'slug'

@extend_schema(tags=['Courses'])
class CourseDeleteAPIView(CourseOwnerMixin, generics.DestroyAPIView):
    queryset = Course.objects.all()
    serializer_class = CourseSerializer
    permission_classes = [permissions.IsAdminUser, IsInstructorPermission]
    lookup_field = 'slug'

```

Figure 6.3: Course CRUD

```python

@shared_task
def transcript_video(id, *args, **kwargs):
    print(id)
    try:
        obj = Video.objects.get(id=id)

        if obj.transcript:
            return

        video_path = f"{MEDIA_ROOT}/{obj.file}"
        extracted_audio_path =  f"{MEDIA_ROOT}/audios/{obj.id}.mp3"
        convert_video_to_audio(video_path, extracted_audio_path)
        with open(extracted_audio_path, "rb") as f:
            data = f.read()
        for i in range(5):
            r = requests.post(URL, data=data)
            response = r.json()
            if 'error' not in response:
                obj.transcript = response['text']
                obj.save()
                break
            else:
                logger.error(f"Transcription error: {response['error']}")
            sleep(10)
    except Exception as e:
        logger.exception(f"Failed to transcribe video {id}: {e}")
```

Figure 6.4: Transcript Integration

```python
class Summarizer(APIView):
    permission_classes = []
    serializer_class = Transcripts

    def post(self, request):
        serializer = Transcripts(data=request.data)
        if serializer.is_valid():
            data = serializer.validated_data['text']
            # try 5 times if one request is valid then return response
            for i in range(5):
                response = requests.post(url="http://127.0.0.1:8080/neuarlearn/ml/summaizer", json={"text":data, "min_length": 50, "max_length": 250})
                print(type(response.json()))
                data = response.json()[0]
                if response.status_code == 200 and not data.get('error'):
                    output_serializer = SummarizerSerializer(data={"summary": data.get('generated_text')})
                    if output_serializer.is_valid():
                        return Response(output_serializer.data)
                    return Response(output_serializer.errors, status=400)
                sleep(20)
            return Response(response.json(), status=response.status_code)
        return Response(serializer.errors, status=400)

```

Figure 6.5: Summarizer Integration

```python

@extend_schema(
    tags=['Question Answer (chatbot)'],
    request=ChatBotRequestSerializer,
    responses={200: ChatBotResponseSerializer(many=True)}
)
class ChatBotAPIView(APIView):
    def post(self, request):
        serializer = ChatBotRequestSerializer(data=request.data)
        if serializer.is_valid():
            context = serializer.validated_data['context']
            question = serializer.validated_data['question']
            chat_history = serializer.validated_data['chat_history']
            k = serializer.validated_data['k']

            # Send data to the model server
            payload = {
                "context": context,
                "question": question,
                "chat_history": chat_history ,
                "k" : k
            }
            response = generate_answer(payload)
            if response.status_code == 200:
                response_data = response.json()
                return Response(response_data, status=status.HTTP_200_OK)
            else:
                return Response({"error": "Model server error"}, status=status.HTTP_500_INTERNAL_SERVER_ERROR)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)


```

Figure 6.6: Question Answer Integration

6.2.3 API Endpoints

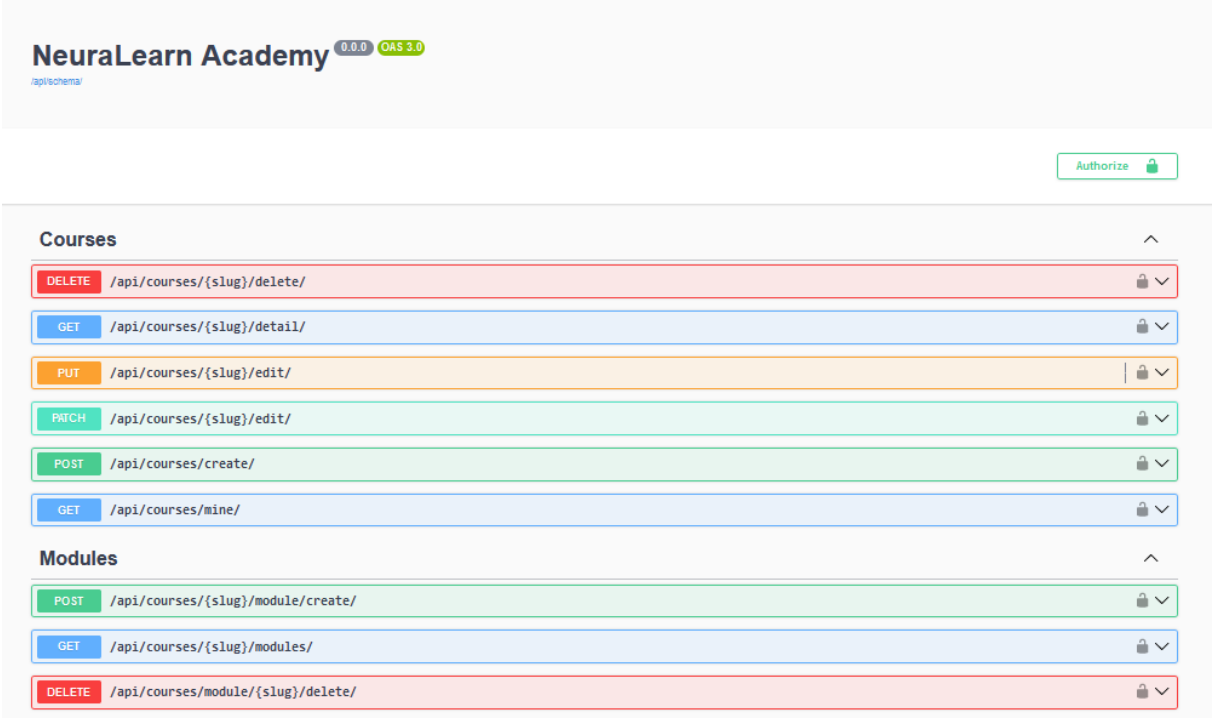


Figure 6.7: API endpoints Course Managment System

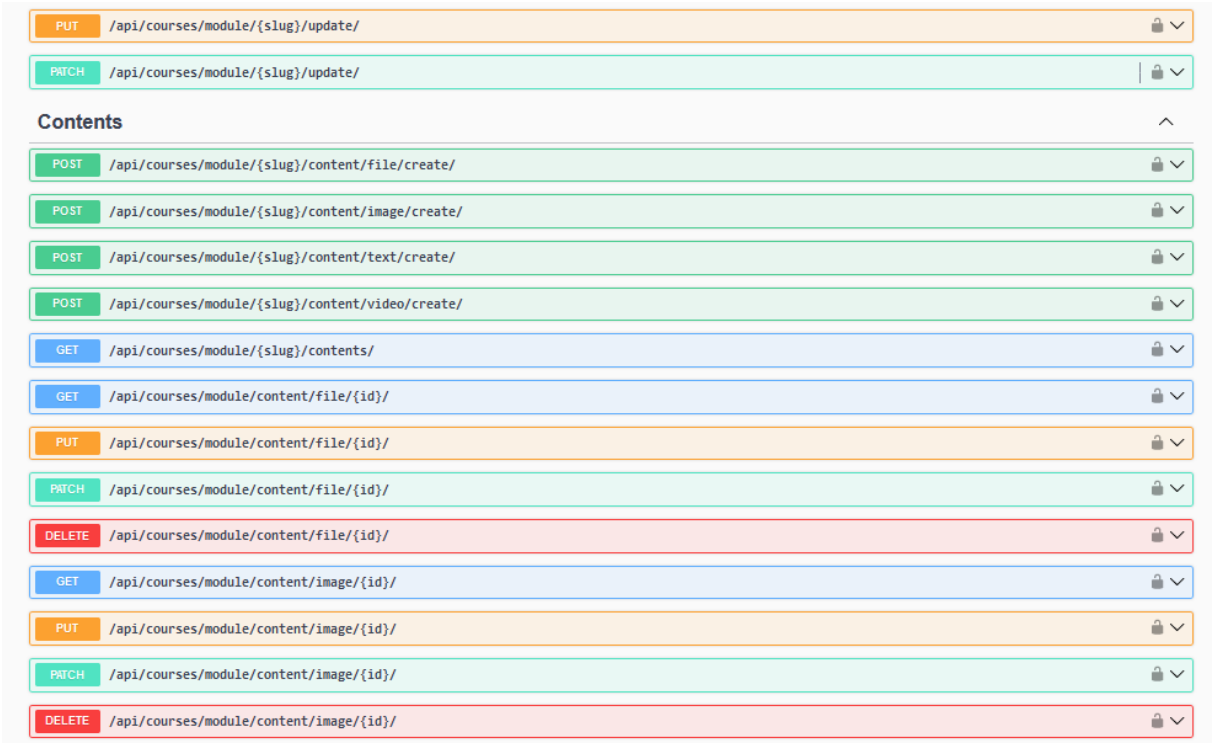


Figure 6.8: API endpoints Course Managment System 2

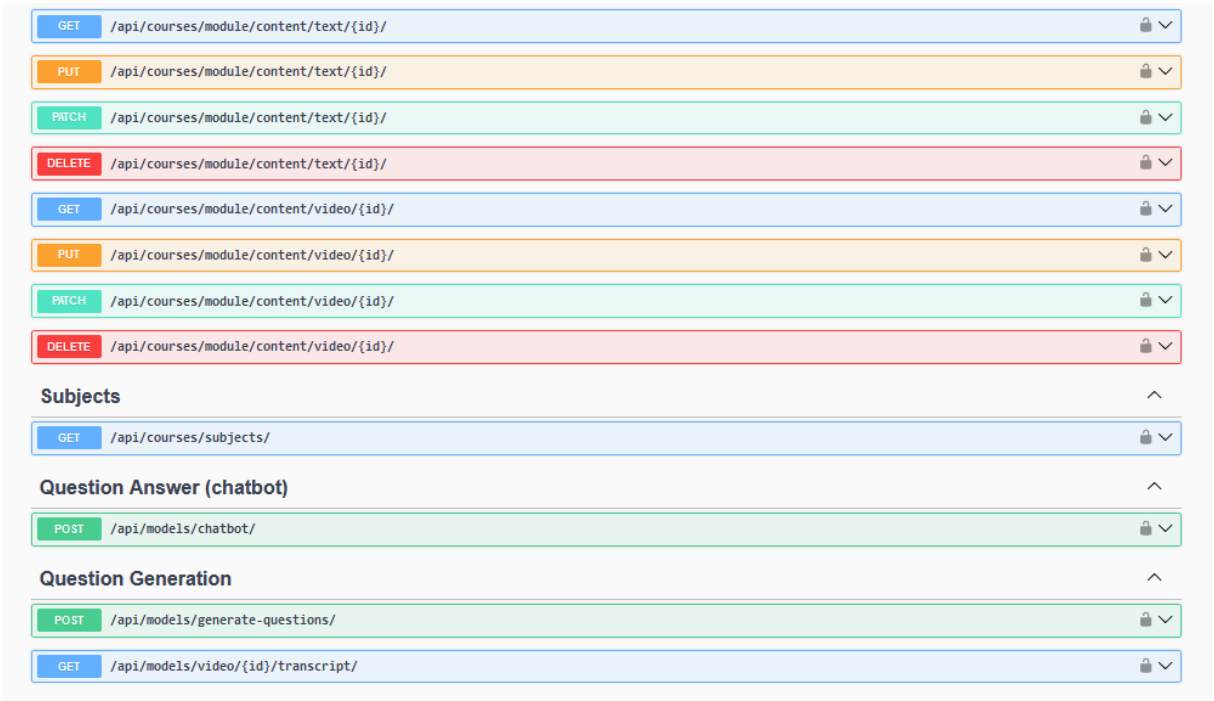


Figure 6.9: API endpoints CMS & ML models

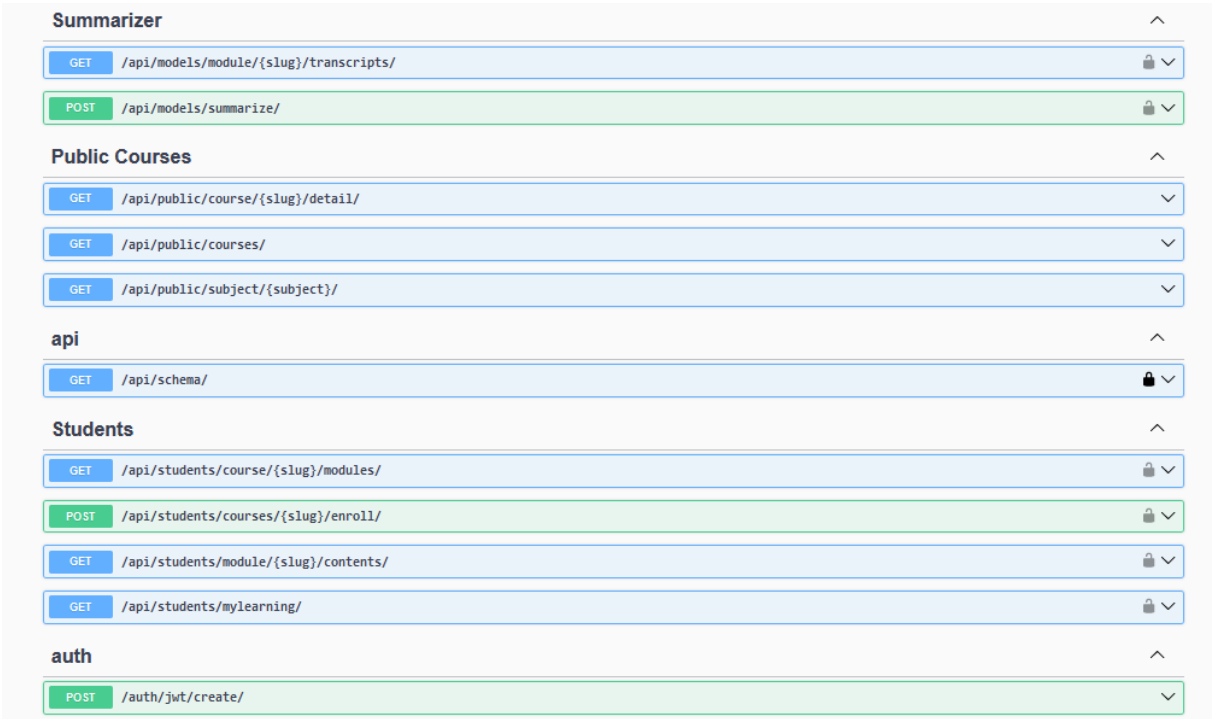


Figure 6.10: API endpoints ML models and students

6.3 Application Screenshots

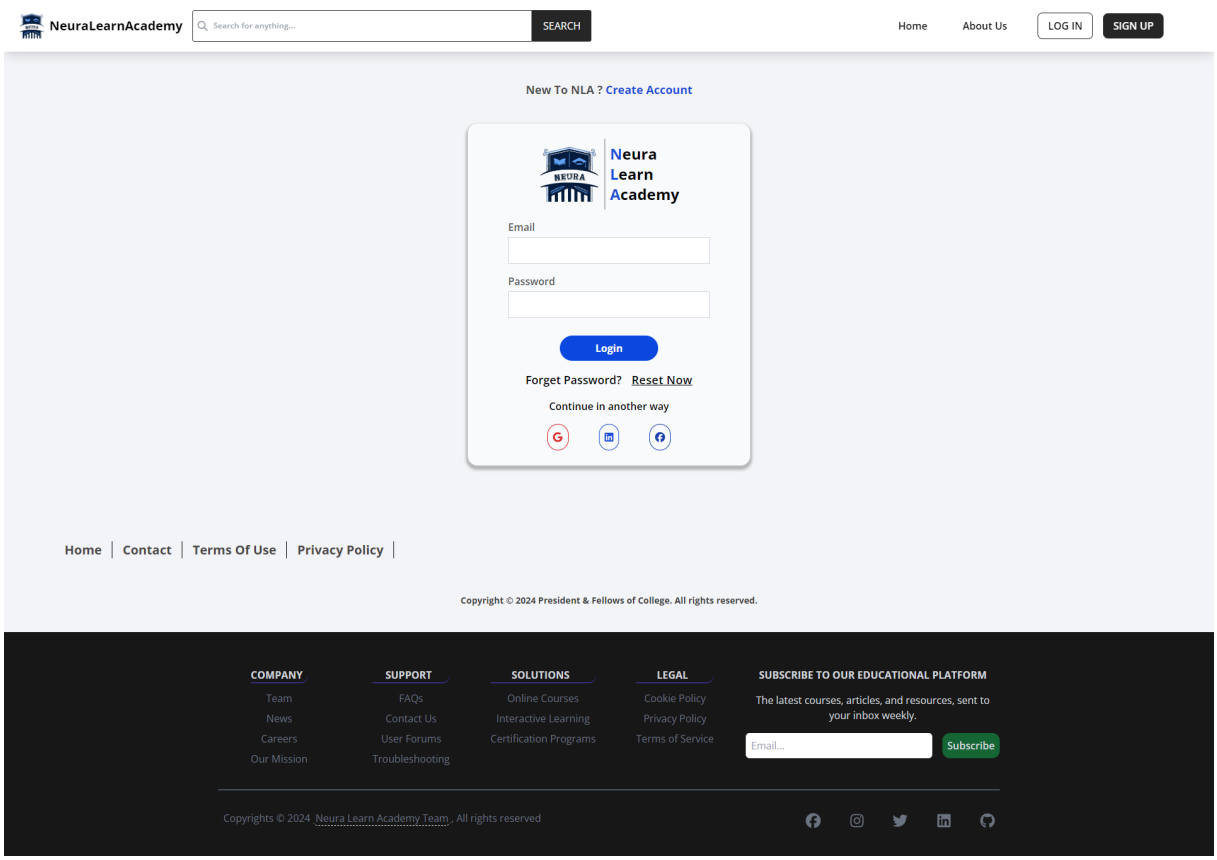


Figure 6.11: Login

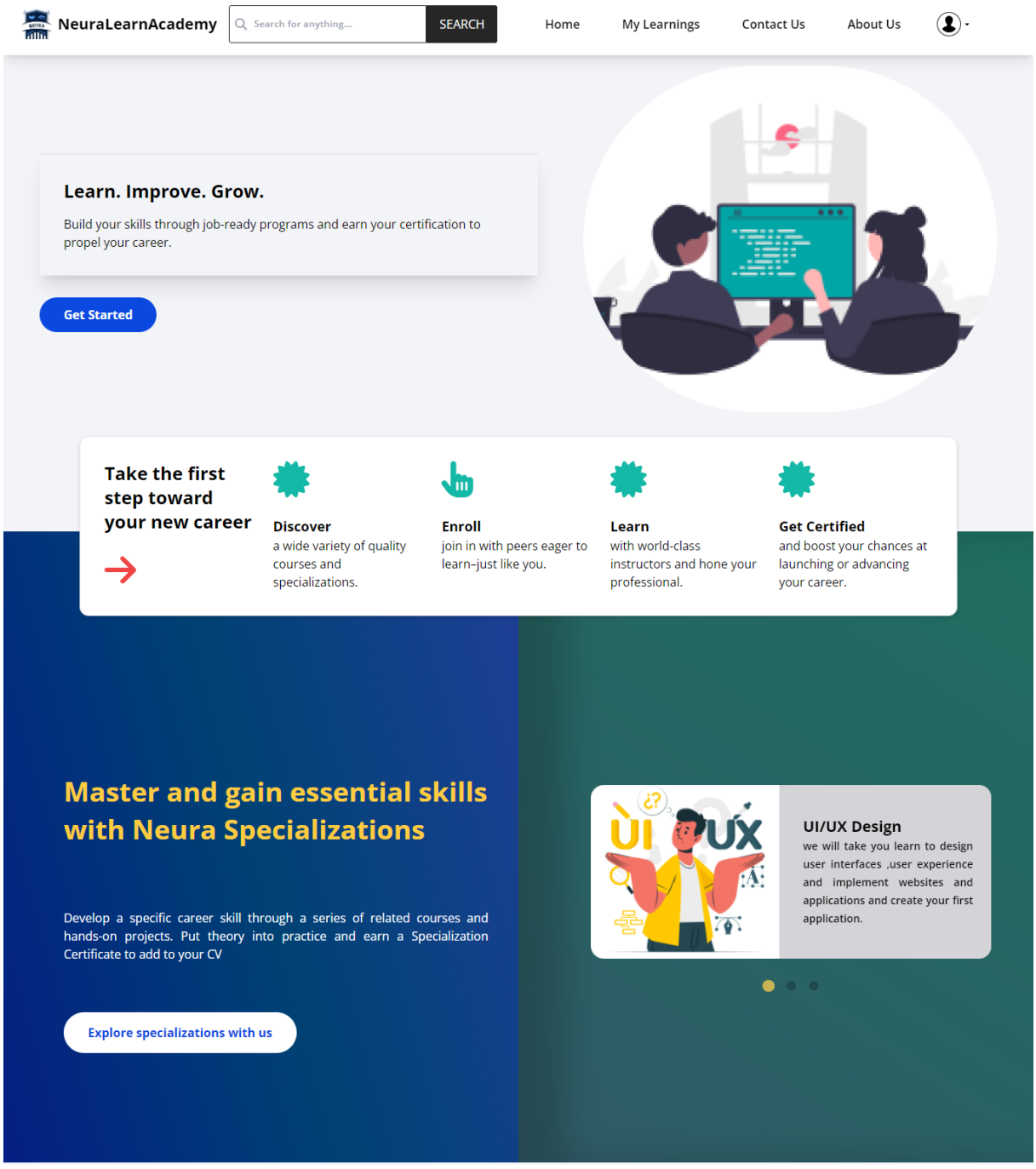


Figure 6.12: Home page

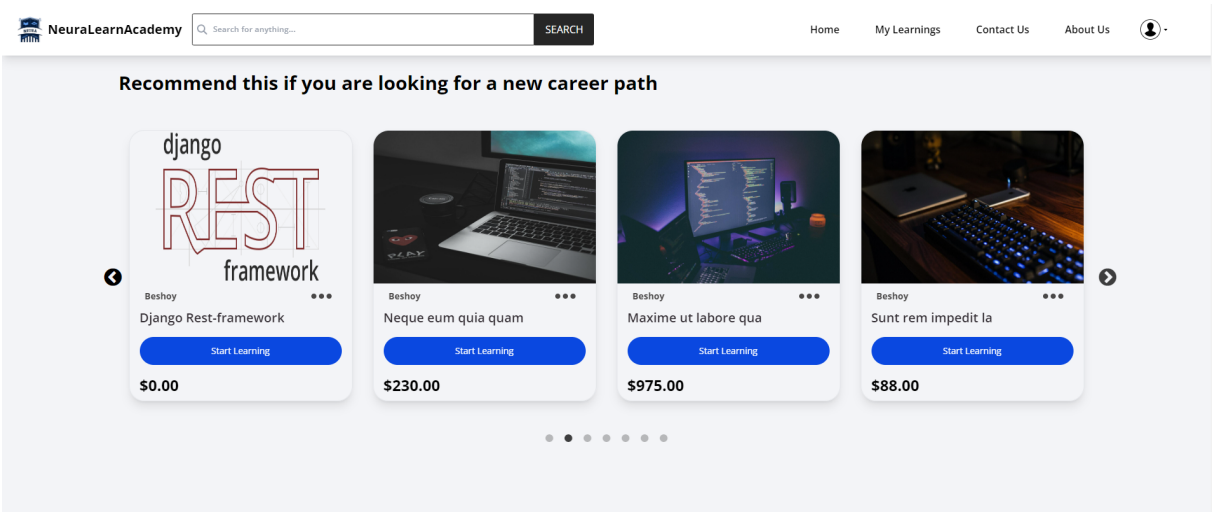


Figure 6.13: Home page 2

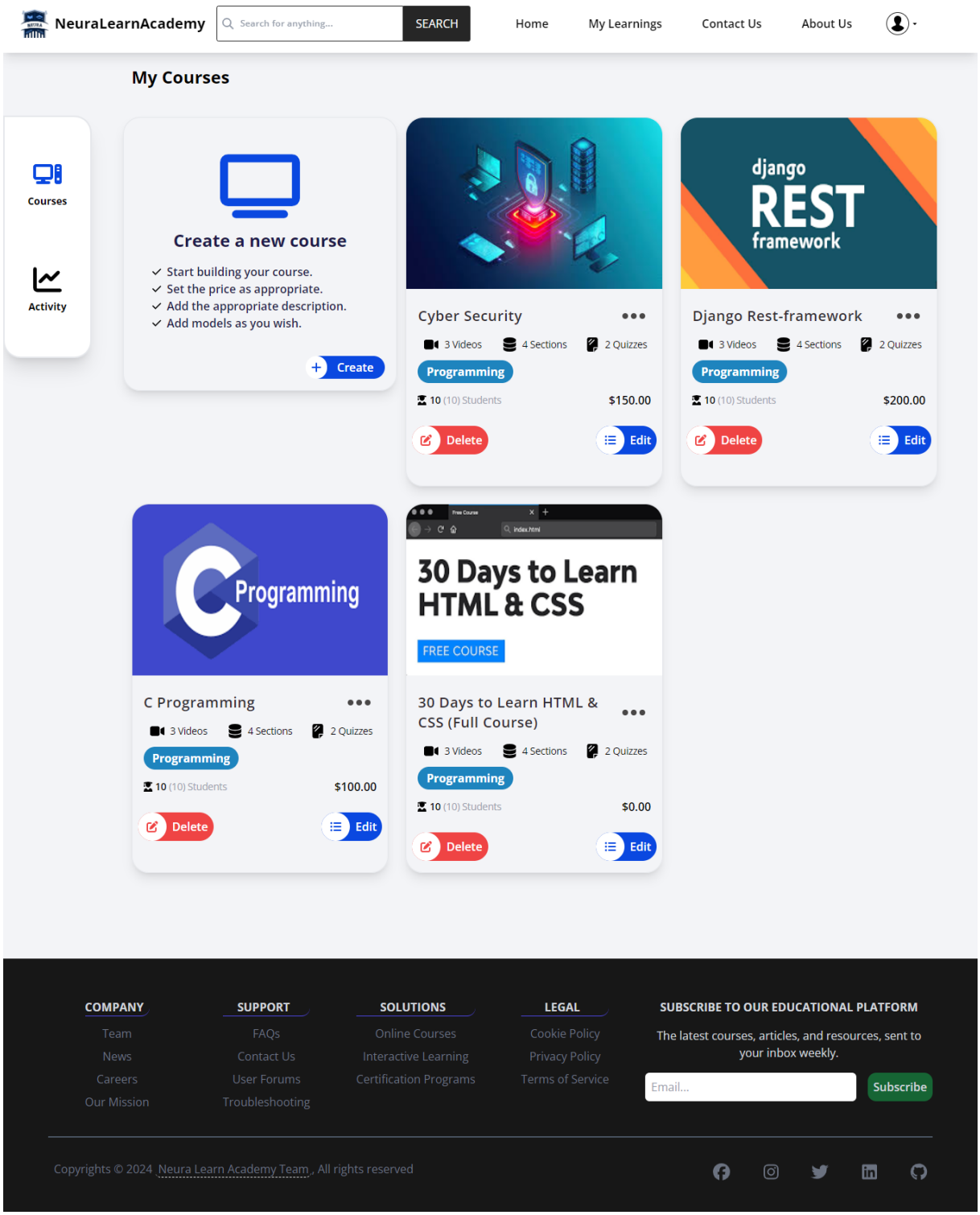


Figure 6.14: Instructor Courses

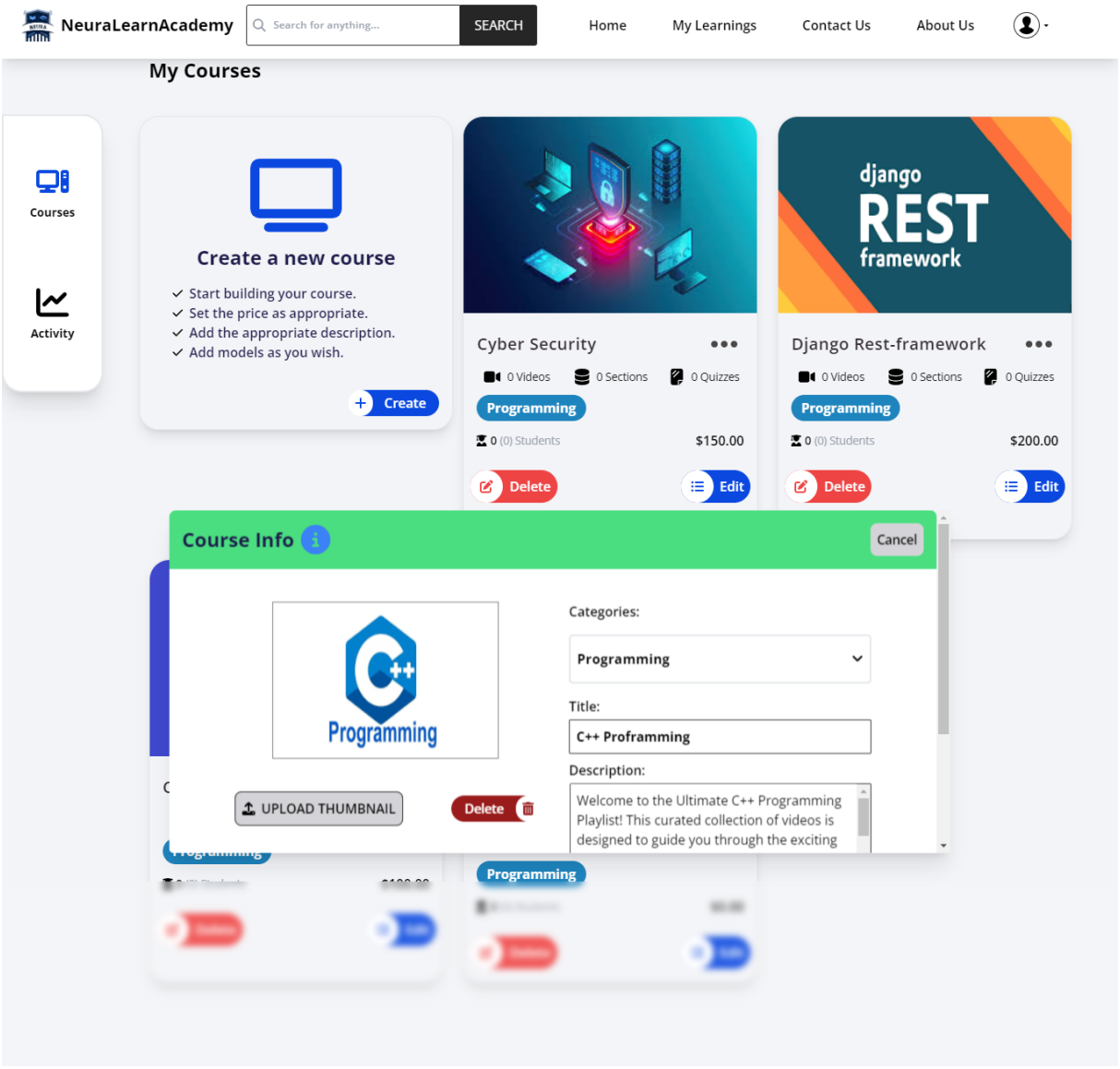


Figure 6.15: Instructor Create course form

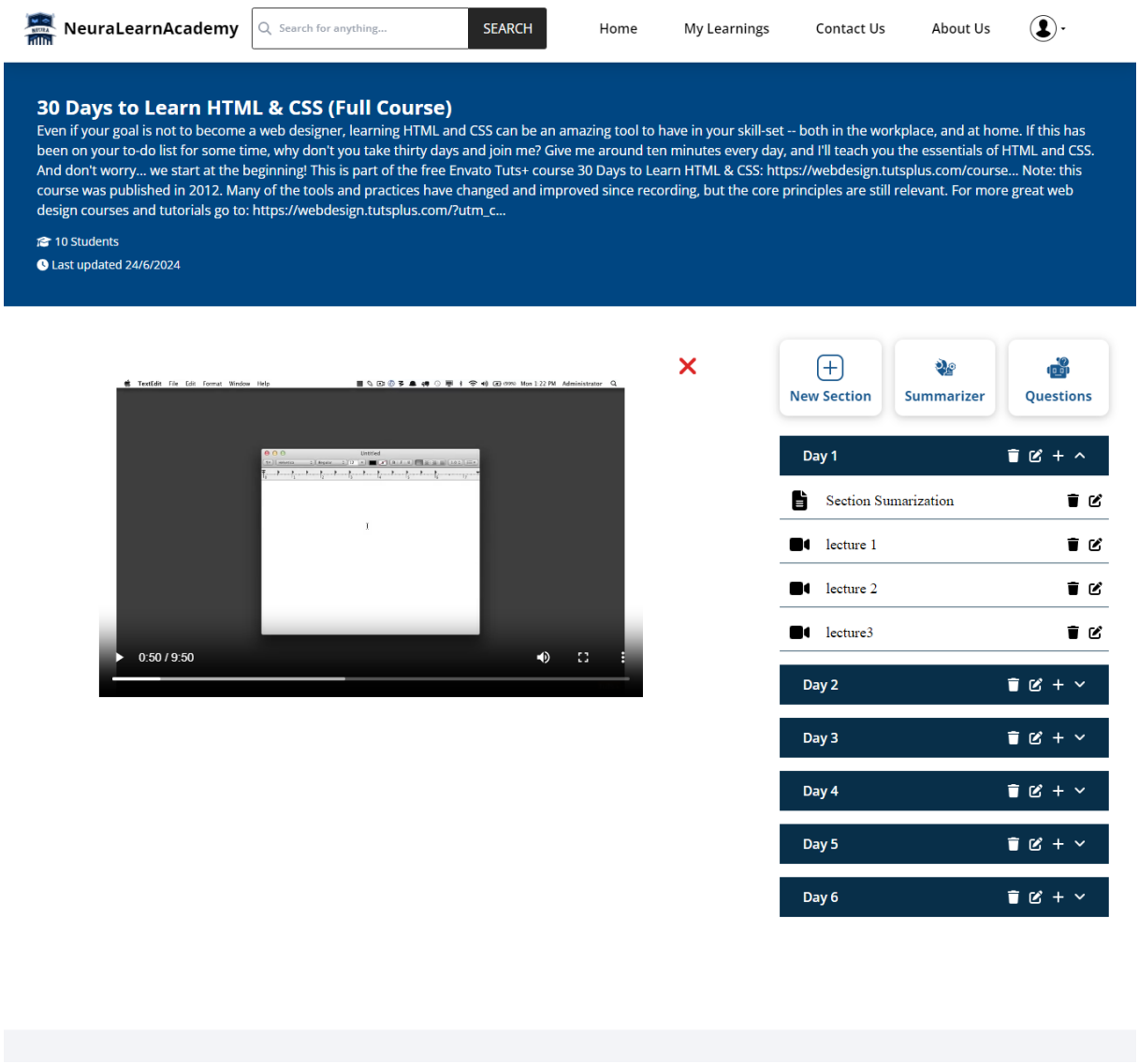


Figure 6.16: Instructor Course Managment System Dashboard

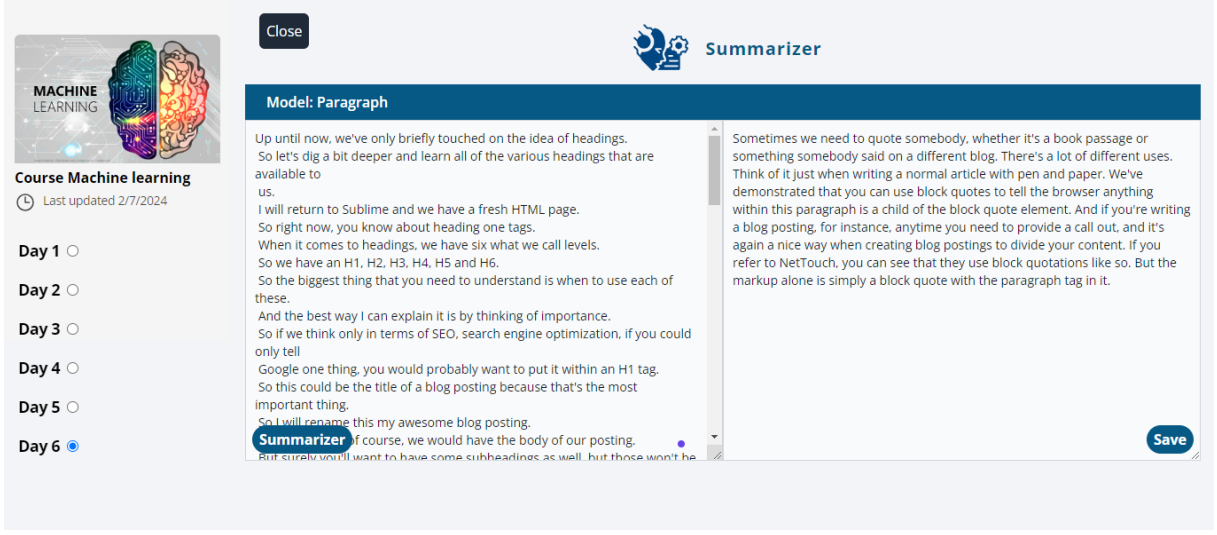


Figure 6.17: Summarizer Page

Chapter 7

Conclusion and future work

7.1 Project conclusion

7.2 Future work

Our future goals include the following:

7.2.1 Machine Learning

- Find new ways to reduce Token generation latency and speed up Inference.
- Develop new architecture to handle visual question generation Such as combining Diffusion and autoregressive language Modeling.
- Develop a new evaluation metric to measure question generation accuracy.
- Develop new techniques to increase the quality of transcripts.

7.2.2 Frontend

- Improve user interface design for a more intuitive user experience.
- Enhance the usability and accessibility of the system.
- Add support for dark mode and customizable themes.
- Integrate more interactive elements such as drag-and-drop features.
- Support the Arabic language.

7.2.3 Backend

- Implement online payment methods.
- Support the Arabic language.
- Implement real-time notifications for user activities.
- Enhance security measures to protect user data.
- Implement caching mechanisms to reduce server load.
- Implement scalable architecture to handle increased traffic.

References

[1] Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi. Open-domain question answering goes conversational via question rewriting. *arXiv preprint arXiv:2010.04898*, 2020.

[2] Mikel Artetxe, Sebastian Ruder, and Dani Yogatama. On the cross-lingual transferability of monolingual representations. *arXiv preprint arXiv:1910.11856*, 2019.

[3] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[4] Michael Carmichael, A Reid, and Jeffrey D Karpicke. Assessing the impact of educational video on student engagement, critical thinking and learning. *A SAGE white paper*, 2018.

[5] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*, 2019.

[6] Bidyut Das, Mukta Majumder, Santanu Phadikar, and Arif Ahmed Sekh. Automatic generation of fill-in-the-blank question with corpus-based distractors for e-assessment to enhance learning. *Computer Applications in Engineering Education*, 27(6):1485–1495, 2019.

[7] Bidyut Das, Mukta Majumder, Santanu Phadikar, and Arif Ahmed Sekh. Automatic question generation and answer assessment: a survey. *Research and Practice in Technology Enhanced Learning*, 16(1):5, 2021.

[8] G Deena, K Raja, Nizar Banu PK, and K Kannan. Developing the assessment questions automatically to determine the cognitive level of the e-learner using nlp techniques. *International Journal of Service Science, Management, Engineering, and Technology (IJSSMET)*, 11(2):95–110, 2020.

[9] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

[10] Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. *arXiv preprint arXiv:1705.00106*, 2017.

[11] Tassilo Klein and Moin Nabi. Learning to answer by learning to ask: Getting the best of gpt-2 and bert worlds. *arXiv preprint arXiv:1911.02365*, 2019.

[12] Vaibhav Kumar et al. Clarq: A large-scale and diverse dataset for clarification question generation. *arXiv preprint arXiv:2006.05986*, 2020.

[13] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[14] Peter Ochieng. Are large language models fit for guided reading? *arXiv preprint arXiv:2305.10645*, 2023.

[15] C Pedersen, M Otokiak, I Koonoo, J Milton, E Maktar, A Anaviapik, M Milton, G Porter, A Scott, C Newman, et al. Sciq: an invitation and recommendations to combine science and inuit qaujimajatuqangit for meaningful engagement of inuit communities in research. *Arctic Science*, 6(3):326–339, 2020.

[16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.