**L** **LM DEVELOPMENT TOOLKIT: BUILDING A CTSE LECTURE NOTES CHATBOT**    
  
  
SE4010 - Current Trends in Software Engineering

Saraf MMMS   
  
IT21297854  
  
  
  
B.Sc. (Hons) Degree in Information Technology specialized in

Software Engineering  
  
  
  
Department of Software Engineering  
  
Sri Lanka Institute of Information Technology  
Sri Lanka  
  
  
  
May 2025

Contents

[Introduction 3](#_Toc197952239)

[Justification of LLM Choice 4](#_Toc197952240)

[Justification of Development Approach 6](#_Toc197952241)

[System Design (Optional Diagram) 9](#_Toc197952242)

[Challenges and Lessons Learned 10](#_Toc197952243)

[Use of GenAI Tools (Prompts and Outputs) 12](#_Toc197952244)

[Technologies Used 14](#_Toc197952245)

[Conclusion 15](#_Toc197952246)

[Source Links 16](#_Toc197952247)

**Figures**

[Figure 1 : System Design 9](#_Toc197949118)

[Figure 2: Generated Question and answers 13](#_Toc197949119)

[Figure 3 : Extracted files and lecture notes used in project 15](#_Toc197949120)

[Figure 4 : Prompt engineering 15](#_Toc197949121)

# Introduction

The rapid and transformative advancement of artificial intelligence (AI) has revolutionized multiple sectors, and one of the most notable areas where this progress has been made is in the field of natural language processing (NLP). Language models, particularly Large Language Models (LLMs), have drastically changed the way we interact with and utilize technology for everyday tasks. These models have proven capable of performing complex tasks such as answering questions, summarizing long passages of text, translating languages, generating creative content, and even automating conversations.

In the context of education, LLMs have immense potential to reshape how students engage with learning materials. Traditional methods of learning and revision often require students to manually go through extensive lecture notes or textbooks, which can be time-consuming and inefficient. This is where AI can make a meaningful difference. By integrating an LLM into educational workflows, we can automate the extraction and processing of lecture material, enabling a more interactive and dynamic learning experience.

This project delves into the application of LLMs, specifically OpenAI’s GPT models, to create an intelligent system designed to assist in the learning process. The system is built to read content from lecture slides, presentations, and other educational documents, extract meaningful information, and generate relevant questions and answers that will aid students in reviewing and understanding the material more effectively. The primary aim is to design a robust system that can digest complex topics and generate concise, accurate, and contextually appropriate responses based on user input. By automating this process, the system offers a unique way for students to interact with their educational content and receive real-time assistance when studying.

Through this project, we aim to demonstrate the practical applications of AI in education, showcasing how technology can enhance the learning experience by making it more efficient, personalized, and engaging. This initiative is not just about automating study processes but also about enhancing the way students comprehend and interact with educational content in a way that is both intuitive and accessible.

# Justification of LLM Choice

Choosing the right Large Language Model (LLM) for this project is a critical decision that directly impacts the system's performance, accuracy, and user experience. The model must not only be capable of handling complex educational content but also be efficient enough to generate meaningful responses in real-time. After careful consideration of several available models, we chose OpenAI’s GPT models, particularly GPT-3.5-turbo, as the best option for this project.

One of the main reasons for selecting GPT-3.5-turbo is its exceptional proficiency in understanding and generating natural language. This model has proven to be one of the most reliable and robust language models available, with the capability to process and produce high-quality text output across a wide variety of contexts. Whether it is answering factual questions, explaining complex concepts, or generating coherent dialogue, GPT-3.5-turbo has demonstrated remarkable versatility in tasks that require deep comprehension of language and context. In the context of this project, such capabilities are essential for understanding the diverse and often intricate educational content, such as lecture slides, textbooks, and PowerPoint presentations, which are typically written in a combination of structured and unstructured formats.

The content that this system needs to process is inherently diverse in nature. Lecture slides and presentations, for example, often contain bulleted lists, diagrams, code snippets, definitions, and other forms of condensed information. GPT-3.5-turbo excels at processing this variety, as it is trained on vast amounts of data and has a sophisticated understanding of both formal and informal language structures. It can effectively analyze and generate responses from content that varies from straightforward definitions to more complex explanations. This makes GPT-3.5-turbo well-suited for educational applications where the text can range from simple terminology to intricate subject matter requiring deeper contextualization.

Another compelling reason for choosing GPT-3.5-turbo is its ability to generate contextually relevant and accurate answers. When a user asks a question related to a particular lecture topic, the system must be able to process the relevant lecture content, understand the underlying context, and provide a concise and informative response. GPT-3.5-turbo's performance in these areas is exceptional, as it is trained on a broad spectrum of text sources and has developed the ability to discern subtle nuances in language. This ability is essential for ensuring that the answers provided by the system are not only relevant but also well-aligned with the context of the lecture material.

Furthermore, OpenAI’s API provides easy access to GPT models, making them highly practical for integration into this project. The extensive documentation and straightforward interface make it simple to send requests to the model and retrieve results, which is critical for maintaining the seamless user experience we aimed for in this project. The API’s scalability ensures that as the project grows or as the system is used by more students, it can handle the increased load without compromising performance. The accessibility of OpenAI’s API also allows for easy fine-tuning of the model in the future, should the need arise to tailor its responses further to suit specific educational domains or content types.

Additionally, OpenAI has consistently demonstrated leadership in developing AI models that are both ethical and transparent, which aligns with the project's goals of providing reliable and responsible educational tools. The company's commitment to fairness and safety in AI deployment provides a solid foundation for building a trustworthy system that users can rely on for accurate and unbiased information.

In summary, GPT-3.5-turbo was chosen for its advanced capabilities in natural language understanding and generation, its adaptability to various forms of educational content, its efficiency in producing relevant and detailed responses, and its ease of integration via OpenAI’s API. These qualities make it an ideal choice for powering an intelligent, user-friendly system designed to assist students in comprehending and interacting with educational materials. The choice of GPT-3.5-turbo is essential for ensuring that the system meets its objectives of enhancing learning and providing real-time, accurate answers to student queries.

# Justification of Development Approach

The development approach for this project was carefully designed to ensure that the system would be both functional and scalable, while also providing a seamless and intuitive user experience. At the heart of this approach is the goal of helping students effectively engage with educational content, particularly lecture slides and presentations. The system focuses on automating the process of extracting, analyzing, and summarizing content from lecture materials, followed by generating relevant questions and answers that can assist students in their review and understanding.

One of the main design principles for this project was to break down the process into manageable, modular components. This modular approach ensures that each stage of the system—whether it’s text extraction, chunking, embedding, or generating questions and answers—functions independently but works seamlessly as part of the larger system. The benefit of this approach is twofold: not only does it make the system more maintainable and flexible, but it also allows for easier modifications or improvements in the future without disrupting the entire process.

The first step in the system’s operation is the extraction of content from diverse formats, such as PDFs and PPTX files. Lecture slides often exist in these two common formats, and ensuring that the system can handle both is crucial for its versatility. By using a combination of Python libraries like PyPDF2 for PDFs and python-pptx for PowerPoint presentations, we were able to create a robust text extraction pipeline that efficiently captures the content from these different formats. The text extraction phase ensures that the information from lecture slides is accurately captured and ready for further analysis.

Once the text is extracted, the next step involves chunking and processing the content. To manage the large volume of text that could result from long lecture slides, we utilized Langchain’s text-splitting capabilities, allowing us to break the text into smaller, more manageable chunks. This is essential for working within the token limitations of the LLMs, ensuring that the model can process the content efficiently without running into issues related to context length. Langchain helps in segmenting the content into logical parts that are easier for the system to analyze and understand, making it an ideal tool for this task.

Next, the system uses OpenAI’s GPT models to process and generate responses based on the extracted text. We leveraged OpenAI's API for its powerful natural language understanding and generation capabilities. The model is responsible for analyzing the chunks of text, understanding the context, and generating meaningful and accurate answers to student queries. By using GPT-3.5-turbo, we were able to tap into a model that excels in answering questions and generating detailed, contextually relevant information, making it a perfect fit for educational applications. The system's ability to generate accurate answers in real-time is a key feature, providing students with instant assistance while they interact with the content.

**Jupyter Notebook** played an essential role throughout the development and testing phases of the project. It served as the interactive environment for prototyping, debugging, and experimenting with various components of the system. The notebook’s ability to execute code in real-time allowed for quick adjustments to the text extraction and chunking processes, enabling seamless integration with OpenAI's API for generating answers. Additionally, Jupyter’s real-time feedback and visualization features helped to efficiently monitor the performance and debug issues that arose during development. Using Jupyter Notebook made it easier to visualize the flow of data and outputs, making the process more transparent and manageable.

User interaction was another focal point of the development approach. We designed the system to be simple and intuitive to ensure that anyone—regardless of technical expertise—could easily navigate the process. After the text is extracted and processed, users are prompted to select a topic of interest from a list of available options. This user-centered design ensures that students can quickly find the area of content they want to explore. Once a topic is selected, users can ask specific questions related to that topic, and the system generates an answer based on the relevant lecture content.

The interaction flow was designed with simplicity and accessibility in mind. Rather than overwhelming users with complex inputs or interactions, the system provides a clear, guided process where users can easily select topics, ask questions, and receive answers. This design approach maximizes the efficiency of the system, ensuring that students can quickly access the information they need without unnecessary barriers.

Overall, the chosen development approach ensures that each component of the system is well-structured, easy to maintain, and scalable for future improvements. The modular design enables flexibility, while the intuitive user interface promotes accessibility for all students. This approach ultimately creates a user-friendly, efficient, and effective system for helping students interact with and comprehend educational materials, ultimately enhancing the learning experience.

# System Design (Optional Diagram)

The system design focuses on a straightforward flow that allows users to input lecture slide data and receive valuable insights in the form of questions and answers. At a high level, the system consists of several key components:

1. **Text Extraction**: The system reads the lecture slides (both PDFs and PPTX files) and extracts text.
2. **Text Chunking**: The extracted text is then split into smaller chunks for easier processing and embedding.
3. **Embedding and Indexing**: The chunks are processed using embeddings, allowing the system to efficiently search and retrieve relevant content.
4. **User Interaction**: The system prompts the user to select a topic and ask a question related to that topic.
5. **Answer Generation**: Based on the selected content, the system generates an answer using OpenAI’s GPT model.

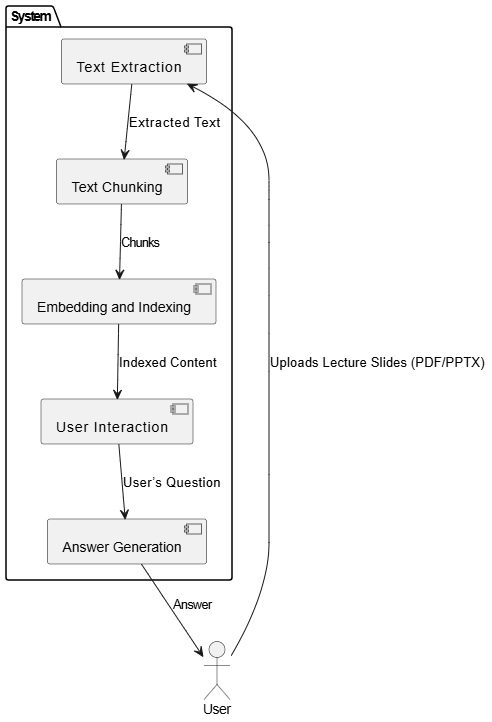


Figure 1 : System Design

# Challenges and Lessons Learned

The development of this system was a complex and multifaceted task, and as with any ambitious project, several challenges emerged that required innovative solutions. These challenges not only tested our technical skills but also provided valuable learning experiences that shaped the final outcome. Below are the key challenges faced during the project, along with the lessons learned from each.

One of the primary challenges encountered was ensuring the accuracy and relevance of the extracted content from both PowerPoint slides and PDFs. Lecture slides often come in various formats, and both PowerPoint and PDF files posed unique hurdles. For PowerPoint slides, the main issue was dealing with fragmented and sometimes disorganized information. Slides often contain bullet points, images, and text boxes, and it was important to extract only the relevant text while ignoring unnecessary elements. This required careful parsing of the slide content to ensure that meaningful and cohesive information was captured. On the other hand, PDFs presented their own challenges, especially when dealing with inconsistent text encoding. Some PDFs had poor text extraction quality due to embedded fonts or scanned images, which made it difficult to extract the text accurately. To address this, we had to implement specific error handling and pre-processing strategies to improve text extraction reliability across both formats.

Another significant challenge was managing the context size that could be processed by OpenAI’s GPT models. The language models come with a token limit, which is essentially a restriction on the amount of text that can be processed in a single request. This limitation posed a challenge when dealing with long lecture slides, as there was a need to carefully chunk the extracted text into manageable pieces that could fit within the token constraints. Balancing the amount of context to include in the prompt while ensuring that the text remained both comprehensive and concise was a key challenge. To solve this, we employed chunking techniques to split the content into smaller sections and prioritized the most relevant information, ensuring that the model could process it efficiently without losing important context.

Another key challenge was the user interaction aspect of the system. Ensuring that users could easily navigate the system, select topics, and ask relevant questions was critical to the overall user experience. While the system was designed to be simple, we faced challenges in maintaining an intuitive interface, especially when it came to selecting topics from large sets of lecture slides. We had to ensure that the user interface was both responsive and easy to use, even when dealing with extensive lecture content. Additionally, managing the flow of interaction—allowing users to ask clear and concise questions while still providing enough context for the model—required constant iteration and testing.

One of the most important lessons learned during the development process was the significance of pre-processing and context management. It became clear that preparing the data in advance—by splitting it into manageable chunks, indexing it correctly, and ensuring that only the most relevant content was used—was critical for the success of the model. Pre-processing not only improved the accuracy of the responses but also made the system more efficient and responsive. In addition, understanding the token constraints of the language model was essential. By being mindful of the model’s limitations, we were able to structure the data in a way that maximized its effectiveness while preventing issues related to excessive context length.

Moreover, the project highlighted the importance of iterative testing and feedback. Throughout the development process, we learned that even small changes to the system (e.g., altering chunk sizes or adjusting the prompt structure) could have a significant impact on the quality of the generated answers. Therefore, continuous testing and refining of the system based on user feedback were crucial to ensuring its performance met the desired standards.

Overall, the project not only deepened our understanding of how to work with LLMs and natural language processing techniques but also reinforced the importance of effective data preparation, context management, and user-centered design. The lessons learned will undoubtedly inform future work in this area, as we continue to explore how AI can enhance learning experiences and provide valuable support to students.

# Use of GenAI Tools (Prompts and Outputs)

At the core of this project is the utilization of OpenAI’s language models, which serve as the engine for generating human-like responses based on the educational content provided by the lecture slides. By feeding carefully extracted and processed chunks of text as context, the system can generate insightful, accurate, and contextually relevant answers to a variety of user questions. This ability to interpret and respond intelligently to user queries is what makes the project valuable in an educational context.

The process begins by selecting relevant content from the lecture slides, which serves as the "context" for generating answers. These chunks of text are then integrated into a prompt, along with a user’s question, in order to guide the language model in producing a precise and informative response. The prompt structure typically looks like this:

**Prompt:**  
"Answer the following question based on the context:\n\nContext: [Extracted content from selected topic]\n\nQuestion: What is the CAP Theorem?"

This prompt is crafted to ensure that the model has all the necessary information from the lecture slides to generate an answer that is relevant to the specific question asked by the user. The context might include key definitions, concepts, and examples related to the topic, ensuring that the model has a comprehensive understanding of the subject matter before generating an answer.

**Example Output:**  
**Answer:** "The CAP Theorem, also known as Brewer's Theorem, states that it is impossible for a distributed system to simultaneously achieve all three of the following guarantees: Consistency, Availability, and Partition Tolerance. This theorem is crucial in understanding the trade-offs that distributed systems must make in order to function effectively in real-world scenarios."

In this example, the system not only explains the key concept of the CAP Theorem but also provides additional context, such as its alternative name and the significance of the theorem in distributed systems. This level of detail helps students gain a deeper understanding of the topic.

By utilizing this approach, the system enables users to engage with complex topics in a more accessible and intuitive manner. It offers clear, well-structured answers that distill critical information into easily digestible pieces. This interaction model reflects how AI can be used to enhance learning, improve knowledge retention, and offer personalized, on-demand educational assistance.

Moreover, the output quality is highly dependent on the quality and relevance of the context provided in the prompt. Ensuring that the context is comprehensive yet concise is essential to obtaining accurate and coherent answers from the model. This highlights the importance of proper content extraction, chunking, and prompt engineering to optimize the performance of the language model in generating meaningful responses.

This application of OpenAI's language models demonstrates the transformative potential of AI in education. It not only empowers students to quickly grasp difficult concepts but also provides them with the tools to actively engage with the material, reinforcing their learning experience.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 2: Generated Question and answers

# Technologies Used

* **Jupyter Notebook**: Jupyter Notebook was used for the interactive development, testing, and prototyping of the system. It provided a flexible environment where we could experiment with different methods for text extraction, chunking, and integration with OpenAI’s API. The notebook’s real-time code execution and visualization capabilities made it an ideal tool for refining the system and debugging various components.
* **Python**: Python was the primary programming language used for this project. It is known for its readability and vast ecosystem of libraries, which made it well-suited for tasks like text extraction, file handling, and API interaction. Python’s extensive support for machine learning and natural language processing (NLP) also contributed to the project’s efficiency.
* **OpenAI GPT-3.5 Turbo**: OpenAI's GPT-3.5 turbo model was used for generating meaningful answers and providing insights based on the lecture slides' content. The model’s ability to understand context and generate coherent responses made it the perfect choice for automating question answering based on educational materials.
* **Langchain**: Langchain was employed to split the extracted text into manageable chunks for easier processing. It also helped facilitate the interaction with OpenAI’s API by creating structured prompts based on the extracted content and user input. Langchain’s flexibility in handling text processing tasks made it an integral part of the system.
* **PyPDF2**: PyPDF2 was used to extract text from PDF files. It allowed the system to parse PDFs and retrieve the lecture content in text format, making it suitable for further processing and analysis.
* **python-pptx**: The python-pptx library was used to extract text from PowerPoint slides. This library enabled the system to parse PowerPoint files and retrieve slide content, which was then processed for question generation and answering.
* **TQDM**: TQDM was used to add progress bars to the data processing tasks. It provided real-time feedback during text extraction and chunking, ensuring that users could monitor the system’s progress in real time.
* **Python-dotenv**: Python-dotenv was used to load environment variables, such as the OpenAI API key, from a .env file. This helped securely manage API credentials and other sensitive information while keeping them separate from the codebase.
* **Glob**: The Glob module was used to handle file paths and search for specific file types (such as .pdf and .pptx) in the directory. It helped the system locate and process the lecture slides and presentations efficiently.

# Conclusion

This project showcases the potential of using Large Language Models (LLMs) like OpenAI’s GPT to enhance educational experiences. By extracting content from lecture slides and generating questions and answers, we’ve created a system that can help students better engage with their learning materials. While there were several challenges related to text extraction, chunking, and prompt management, the solution successfully provides valuable insights to users. The project not only highlights the capabilities of LLMs but also offers a glimpse into the future of AI-powered education, where students can easily access information and receive instant help through conversational interfaces.

In conclusion, the use of AI in education is an exciting field, and this project demonstrates how powerful these tools can be in helping students review and understand complex subjects.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer code

AI-generated content may be incorrect.

Figure 3 : Extracted files and lecture notes used in project

Figure 4 : Prompt engineering

# Source Links

Github Link : <https://github.com/it21297854/CTSE_Assignment_02.git>

Youtube Link : <https://youtu.be/HwcM-nlB7VU>