STM JOURNALS

International Journal of
Electronics Automation

Volume 1, Issue 2, 2023
July–December
DOI (Journal): 10.37591/IJEA

https://journals.stmjournals.com/ijea

Research IJEA

LLM-based Chatbot for Course-based Question Answering

Samyuktha M.S.[1,*], S. Charumathi[2], Darsana R.[3], S. Lovelyn Rose[4]

Abstract

The "LLM based Chatbot for Course-Based Question Answering" project addresses the pressing need for tailored and efficient learning tools in education. By using a state-of-the-art Large Language Model (LLM) with a diverse dataset, including textbooks, professor slides, web scraping data, the chatbot offers accurate and contextually enriched responses to students' course-related queries. Using recent advances in language modeling, this work presents a long former-based Language Model (LLM) for constructing a smart chatbot optimized for course-based question answering (CBQA). The suggested LLM-based chatbot has a user-friendly interface where students can submit questions on various subjects, topics, or concepts presented in their courses. The chatbot uses natural language understanding techniques to extract relevant information and generate responses to these requests. The LLM-based chatbot's key feature is its capacity to answer a wide range of question kinds, including factual, conceptual, and problem-solving queries. Through fine-tuning on individual course materials, the chatbot adapts to the unique vocabulary and content of different disciplines, ensuring accurate and contextually relevant responses. It also features a function for extracting chapter summaries and generating concise study notes, facilitating efficient revision. The project's outcomes include providing students with a robust tool for course clarification, enabling them to practice with relevant materials, and streamlining the revision process, ultimately enhancing the learning experience within the specified college.

Keywords: LLM based Chatbot, Large Language Model (LLM), Natural Language Processing (NLP), Question Answering (QA) Systems, processing, Chapter summaries, Efficient revision, Course clarification, Academic enhancement.

INTRODUCTION

The development of the "LLM based Chatbot for Course-Based Question Answering" system is driven by the dynamic landscape of education, where the demand for tailored and efficient learning tools has surged. In this rapidly evolving environment, it has become imperative to harness advanced language models to create intelligent systems that cater to the specific needs of students. Providing course-based question-answering support is of paramount importance as it directly addresses the challenges faced by students in comprehending and navigating their course materials. The chatbot's ability to deliver accurate and contextually enriched responses not only dispels doubts but also nurtures deeper understanding, thereby fostering a more effective learning experience. The problem statement lies in the unmet need for a comprehensive and interactive platform that assists students in resolving queries, exam preparation, and overall learning enhancement. To address this, the core objectives of the chatbot project include fine-tuning a state-of-the-art Large Language Model

***Author for Correspondence**
Samyuktha M.S.
E-mail: samyuktha1262@gmail.com

[1-3]Student, Department of Computer Science & Engineering, PSG College of Technology, Coimbatore, Tamil Nadu, India
[4]Associate Professor, Department of Computer Science & Engineering, PSG College of Technology, Coimbatore, Tamil Nadu, India

Received Date: March 15, 2024
Accepted Date: March 30, 2024
Published Date: April 10, 2024

Citation: Samyuktha M.S., S. Charumathi, Darsana R., S. Lovelyn Rose. LLM-based Chatbot for Course-based Question Answering. International Journal of Electronics Automation. 2023; 1(2): 30–41p.

(LLM) with a diverse dataset and integrating features like PDF processing for efficient revision, aiming to provide a holistic solution to students' educational needs.

LITERATURE REVIEW

Large language models (LLMs) have attracted significant attention for their transformative potential across diverse domains. Gao, Baptista-Hon, and Zhang [1] delve into the implications of LLMs in medicine and research, particularly focusing on clinical natural language processing (NLP) tasks. Augmentation techniques have been proposed to enhance LLM performance, as demonstrated in the LLM-Augmenter system [2]. Petersen et al. [3] showcase the adaptability of LLMs in educational contexts through their system for classifying course discussion board questions. Moreover, domain specialization for LLMs is explored in "Beyond One-Model-Fits-All" [4], emphasizing the necessity of tailored models for specific application domains. Yager [5] illustrates the feasibility of domain-specific chatbots powered by LLMs, highlighting their potential to provide informative responses and accelerate research in the physical sciences. These studies collectively underscore the multifaceted utility of LLMs in advancing various fields [6].

METHODOLOGY

Selection and Overview of Language Models

Two advanced language models, GPT-3.5 and Llama2, were chosen for their superior natural language processing capabilities.

GPT-3.5

GPT-3.5, an advanced iteration developed by OpenAI, embodies a transformer-based architecture trained on a vast corpus of diverse textual data. Its adeptness in generating coherent and contextually relevant responses to a wide array of queries makes it an ideal candidate for the project's objectives. Leveraging GPT-3.5's expansive pre-training, the chatbot can comprehend and respond to nuanced inquiries with remarkable accuracy and depth.

Llama2

Llama 2, a family of pre-trained and fine-tuned large language models (LLMs) developed by Meta AI, has been instrumental in enhancing the dialogue capabilities of the chatbot within our project. With a parameter range spanning from 7B to 70B, Llama 2 offers scalability and versatility tailored for dialogue-centric applications, positioning it as a formidable contender in the landscape of open-source chat models. Performance evolution of GPT 3.5 Vs Llama 2 shown in below Table 1.

Performance Evaluation

Performance evaluation after using above mentioned methodology is performed and elaborated below:

Database Utilization

Two Databases are used to store the vector embeddings, one each for the Language model chosen according to the dimensions of the embeddings generated.

Chroma

Chroma DB is an open-source vector storage system (vector database) designed for the storing and retrieving of vector embeddings. Its primary function is to store embeddings with associated metadata for subsequent use by extensive language models. It provides a lightweight wrapper around OpenAI embeddings. By default, Chroma utilizes the Sentence Transformers model "all-MiniLM-L6-v2" for generating embeddings. This model can create embeddings for sentences and documents, offering versatility across a range of tasks [6]. The embedding process is executed locally on the user's machine, and may prompt the download of model files, which occurs automatically. The maximum dimension of the vector embeddings is 1536.

Table 1. Performance Evaluation of GPT 3.5 Vs. Llama 2.

Benchmark (Shots)	GPT 3.5	Llama 2
MMLU (5-Shot)	70.0	68.9
GSM8K (8-Shot)	57.1	56.8
HumanEval (0-Shot)	48.1	29.9

Source: Llama 2: Open Foundation and Fine-Tuned Chat Models by Meta

Table 2. Characteristics of OpenAI Embeddings.

Characteristics	Range
Rough Pages Per Dollar	12,500
Example Performance On Mteb Eval	61.0%
Maximum Input Tokens	8191
Output Dimensions	1536

Table 3. Characteristics of Hugging Face Embeddings

Characteristics	Range
Maximum Input Tokens	256
Output Dimensions	384
Avg. Performance	58.80
Encoding Speed	14200

FAISS

The Facebook AI Similarity Search (Faiss) library is designed for efficient similarity search and clustering of dense vectors. It features algorithms capable of searching in vector sets of varying sizes, including those that may exceed available RAM capacity. Faiss enables developers to perform multimedia document searches that are either inefficient or unfeasible using standard database engines like SQL. The library includes implementations for nearest-neighbor searches on datasets ranging from millions to billions of vectors, optimizing the tradeoff between memory usage, speed, and accuracy. Faiss aims to deliver state-of-the-art performance across all operational scenarios. Implemented primarily in C++, Faiss also offers an optional Python interface and supports GPU acceleration via CUDA, with several key algorithms available for GPU usage. The dimensions of vectors used for FAISS in this chatbot is 384.

Embeddings Generation

OpenAI and Hugging Face Embeddings are chosen for the Language models GPT 3.5 and Llama 2 respectively.

OpenAI Embeddings

The OpenAI Embeddings as shown below Table 2 used in this chatbot is generated using the text-embedding-ada-002 model with cl100k_base tokenizer. Ada002 embeddings capture intricate semantic relationships and contextual nuances within the text, enabling a deeper understanding of the input data. The embeddings generated by this model are scalable, capable of handling both short and long text sequences effectively, making them suitable for processing large volumes of textual data. Metrics of Embeddings generated using text-embedding-ada-002 model.

Hugging Face Embeddings

The Hugging Face embeddings utilized in this chatbot are produced by the pre-trained MiniLM-L6-H384-uncased model and fine-tuned on a dataset comprising 1 billion sentence pairs. Primarily designed as a sentence and short paragraph encoder, this model aims to extract semantic information from input text. The resulting vector serves various purposes such as information retrieval, clustering, or tasks related to sentence similarity. The metrics of the embeddings are summarized below in Table 3.

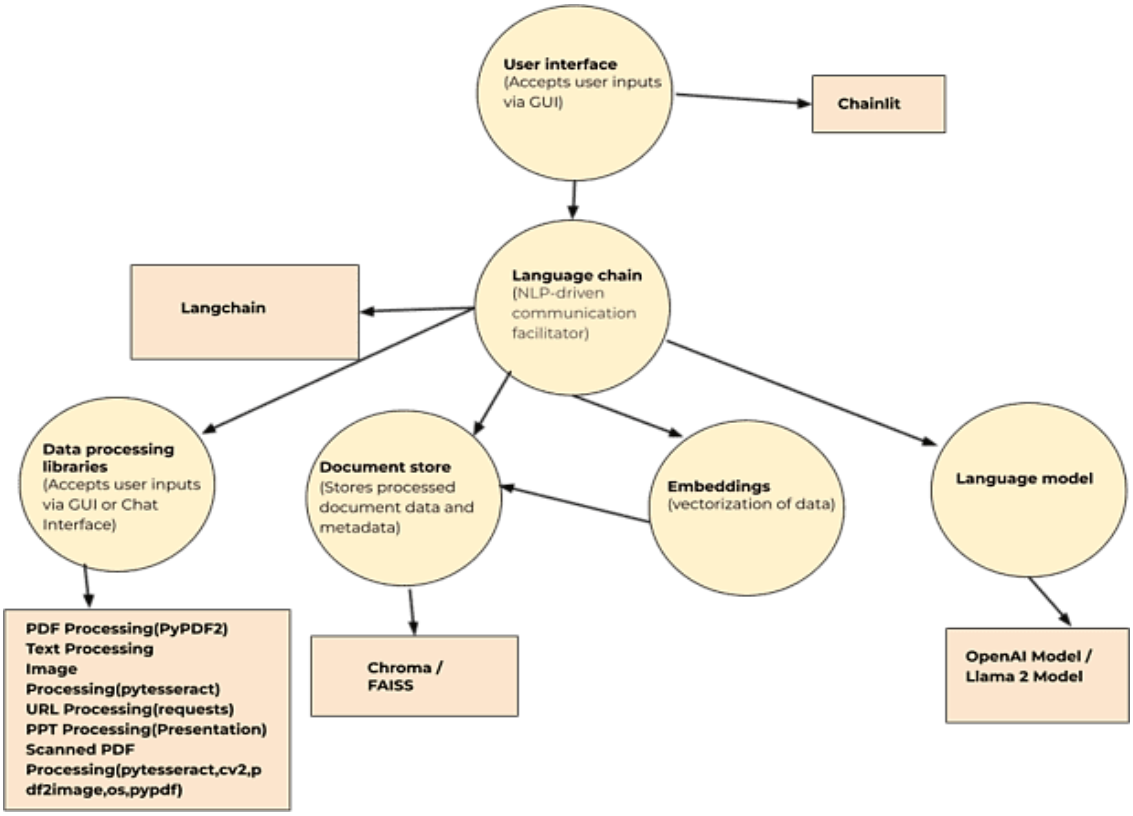


Figure 1. System Architecture.

SYSTEM DESCRIPTION

System Architecture

The chatbot system is powered by a finely tuned Large Language Model (LLM) at its core. This LLM is trained on a diverse dataset comprising textbooks, professor slides, web scraping data, Images, and Scanned PDF enabling it to provide accurate and contextually enriched responses to user queries. The system includes an Input Processor to interpret user queries and uploaded PDFs, a Knowledge Base with comprehensive course materials, and a PDF Processing Module for extracting chapter summaries and generating study notes [7]. The Response Generator crafts clear, concise, and relevant responses, while the User Interface serves as the interaction point. A module for AI ethics and data privacy ensures secure handling of student data, and a feedback mechanism allows for continuous improvement. System Architecture is shown in Figure 1.

Components

User Interface: Implemented using ChainLit, it enables interactive interactions with the chatbot via graphical elements and chat-based communication, ensuring a seamless user experience. (Appendix-I)

Language Chain: Implemented with LangChain, it serves as a comprehensive NLP framework for tasks like data retrieval, metadata management, creating embeddings, and working with Language Model Models (LLMs) for accurate responses.

Data Processing Libraries:

- *PDF Processing (PyPDF2):* Extracts text and metadata from PDFs for analysis.
- *Text Processing:* Handles tasks like parsing, keyword extraction, and manipulation for effective information retrieval.
- *Image Processing (Pytesseract):* Extracts text from images using OCR techniques.
- *URL Processing (Requests):* Retrieves data from web resources for analysis.

- *PPT Processing (Presentation):* Handles data from PowerPoint presentations.
- *Scanned PDF Processing:* Extracts text from scanned PDFs.

Document Store:

- *Chroma Data Store:* Efficiently manages processed document data and metadata, ensuring seamless storage and retrieval of vectorized data representations.
- *Faiss Data Store:* Handles large-scale data retrieval and analysis, particularly for complex similarity searches and clustering.

Embeddings: Generates vector representations for processed data using Open AI Embeddings and Hugging Face Embeddings, enhancing data analysis, retrieval, and context-based operations [8].

Language Model: Utilizes advanced NLP techniques and algorithms to process user inputs and generate contextually relevant responses, serving as the core component of the system. Llama and OpenAI models are used based on user preference (Appendix-VII-VIII).

USER INTERFACE

Chain Lit is utilized to develop the user interface, which includes:

Settings Panel

Allows users to choose the LLM model, select Embeddings, set temperature, define Chunk size and Chunk Overlap, and specify the number of files (Appendix-VI).

Text Input Area

Enables users to input questions, serving as the gateway for LLM interaction, facilitating efficient querying, receiving responses, and accessing information.

Session History Section

Keeps a record of interactions and conversations, providing a log of past interactions with LLM for reference, context, and continuity in the dialogue.

Light and Dark Mode

Allows users to switch between contrasting display themes for optimal visibility and aesthetics, enhancing user experience based on preference and environment (Appendix-II-III).

Chain of Thought

Enables users to view the reasoning process used by the LLM to arrive at an answer. CoT prompting guides the LLM to think step by step by providing a few-shot example outlining the reasoning process, allowing the model to follow a similar chain of thought when responding to prompts (Appendix-IV-V).

IMPLEMENTATION

PDF Processing

PDF Processing (Figure 2) module employs the PyPDF2 library to handle PDF files. It first opens the input PDF and then extracts text content using PyPDF2's text extraction capabilities. The process involves loading the PDF, iterating through its pages, and collecting the text from each page. The extracted text is then returned to the user.

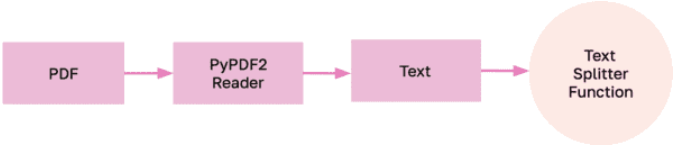


Figure 2. PDF Processing.

Scanned PDF Processing

Scanned PDF Processing (Figure 3). This component is designed to handle scanned PDFs, which typically consist of images. It utilizes PyPDF to extract images from the input PDF. Once the images are extracted, the system uses Optical Character Recognition (OCR) through the Pytesseract library to convert the image content into text. The extracted text is then made available to the user.

Image Processing

Image Processing System (Figure 4) In this function, the system accepts an image as input and leverages Pytesseract, a Python wrapper for Google's Tesseract OCR engine, to recognize and extract text from the image. The image is first preprocessed to enhance text visibility, and then Pytesseract is applied to extract textual content from the image.

URL Processing

URL Processing (Figure 5) module handles a list of comma-separated URLs. It employs the Requests library to fetch the webpage content from each URL. After obtaining the HTML content, it parses and extracts the textual information using various parsing techniques like Beautiful Soup or regular expressions. The module aggregates and returns the text content from all the specified URLs.

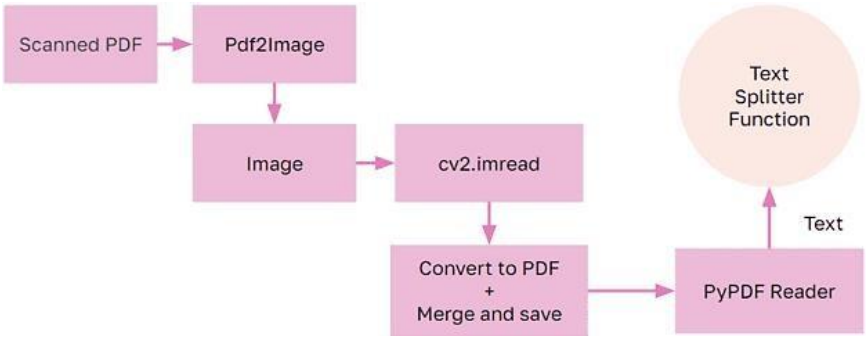


Figure 3. Scanned PDF Processing.

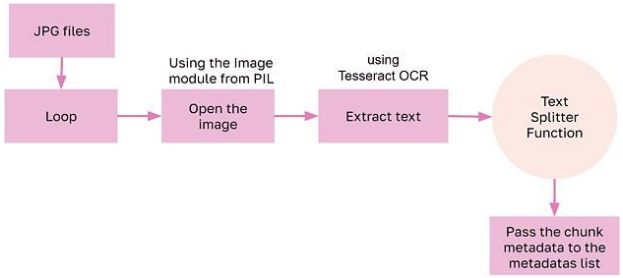


Figure 4. Image Processing.

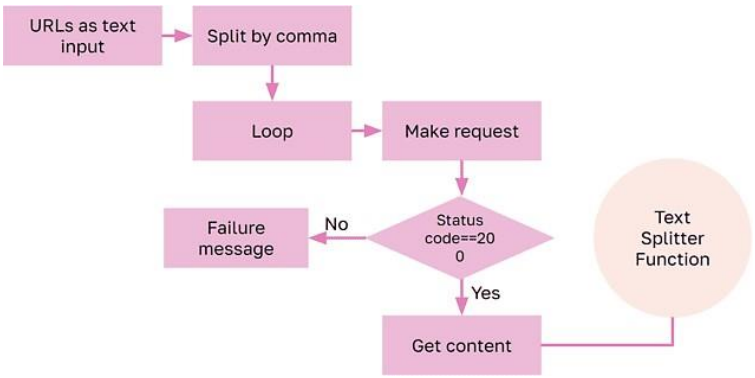


Figure 5. URL Processing.

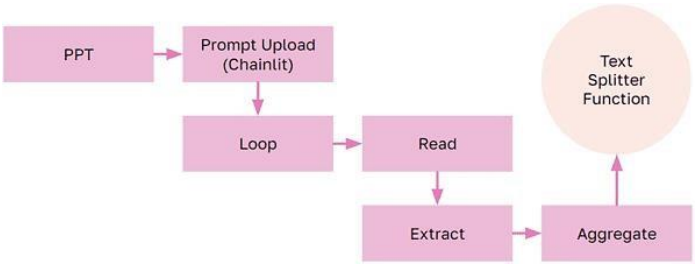


Figure 6. PPT Processing.

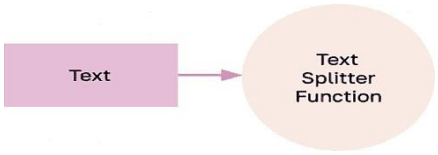


Figure 7. Text Processing.

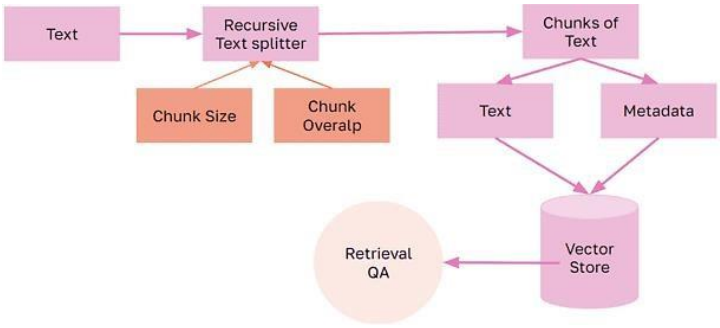


Figure 8. Recursive Text Splitter.

PPT Processing

The PPT processing (Figure 6) function uses a library like python-pptx to open the input PowerPoint presentation. It iterates through the slides, extracting text from each one. The text from each slide is collected and returned as the output. This function does not include image or media extraction, focusing solely on text content.

Text Processing

Text Processing (Figure 7) module is straightforward and is used for processing plain text documents. It takes the text document as input and directly returns its contents without any additional processing. It is suitable for files in formats like .txt or .docx.

Text Splitting Module

Recursive Text Splitter (Figure 8)

The Recursive Character Text Splitter takes a large text and splits it based on a specified chunk size. It does this by using a set of characters. The default characters provided to it are [“\n\n”, “\n”, “ “, ““].

It takes in the large text then tries to split it by the first character \n\n. If the first split by \n\n is still large, then it moves to the next character which is \n and tries to split by it. If it is still larger than our specified chunk size it moves to the next character in the set until we get a split that is less than our specified chunk size.

SYSTEM PERFORMANCE

While formal testing was not conducted to measure specific metrics, a qualitative assessment of the chatbot's performance was performed based on user interactions and feedback. The chatbot

demonstrated satisfactory accuracy in providing responses to course-related queries, with users reporting a high level of satisfaction with the system's ability to clarify course concepts and provide relevant information. Additionally, the chatbot exhibited efficient response times, ensuring timely assistance to users seeking academic support. User engagement with the chatbot was consistently positive, indicating its effectiveness in facilitating learning and academic enhancement. Anecdotal evidence suggests that the chatbot effectively serves its intended purpose of aiding students in course clarification and revision. Further quantitative evaluation may be warranted in future iterations to provide more robust insights into the system's performance [9].

DISCUSSION

The "LLM-based Chatbot for Course-Based Question Answering" promises to transform students' learning by delivering tailored responses with deep contextual understanding, aiding comprehension. Integration of past question papers enhances exam readiness, while a summarization feature streamlines revision. The chatbot's accessibility and prompt responses empower students, potentially improving their academic performance and satisfaction. Leveraging advanced language models paves the way for a more engaging and effective learning mode. In essence, it serves as a crucial tool for creating a dynamic educational environment.

The following are the limitations and areas of future improvement:

- *Specialized Language Model:* Integrating a fine-tuned open-source language model specialized for education enhances answer accuracy by adapting to course-specific language and terminology.
- *Handwriting Recognition:* Extending OCR to recognize handwritten notes broadens the chatbot's material processing capabilities.
- *Voice Search:* Implementing voice recognition enables users to ask questions verbally, enhancing accessibility and user-friendliness.

Our project represents a significant contribution to the evolving landscape of educational technology by harnessing advanced language models to provide personalized and accessible academic support. It addresses the growing demand for intelligent systems that enhance learning, making education more efficient, interactive, and engaging [10].

CONCLUSION

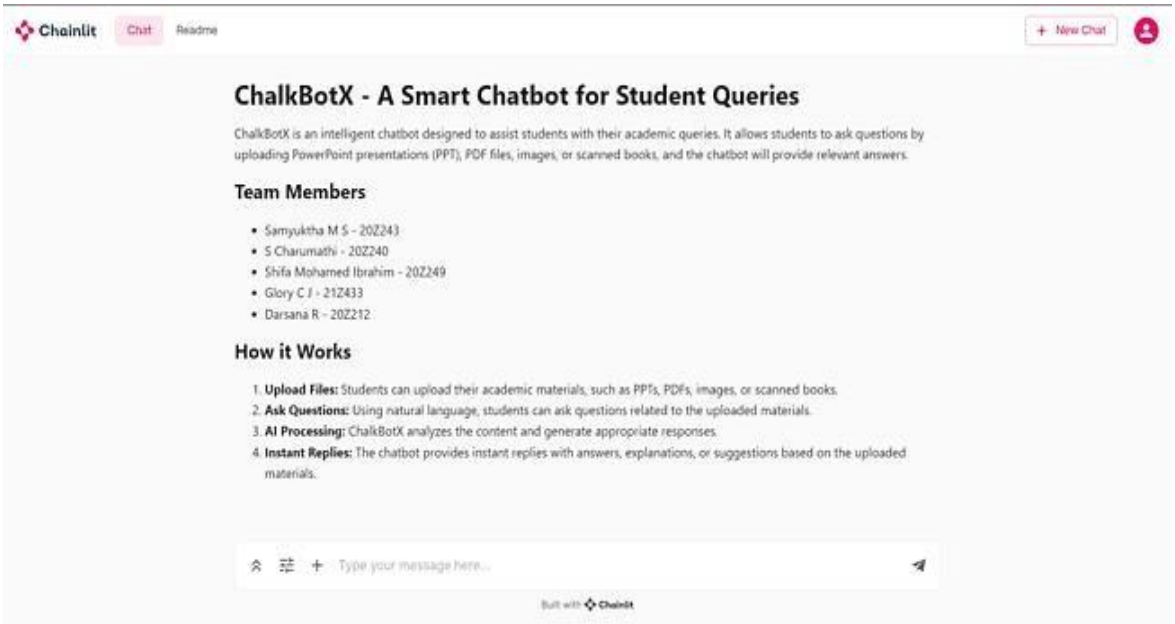
In conclusion, the "LLM Based Chatbot for Course-Based Question Answering" project offers a versatile and accessible educational tool. With the ability to accept various input formats and provide context-specific responses, it enhances learning experiences. The integration of past question papers and the generation of concise study notes streamline exam preparation and revision. Its user-friendly interface fosters self-directed learning, while ethical data handling ensures responsible use. This project exemplifies a timely and impactful endeavor, addressing the demand for personalized and streamlined educational support. It holds significant potential for enhancing the learning experience of students and contributes to the broader field of educational technology.

REFERENCES

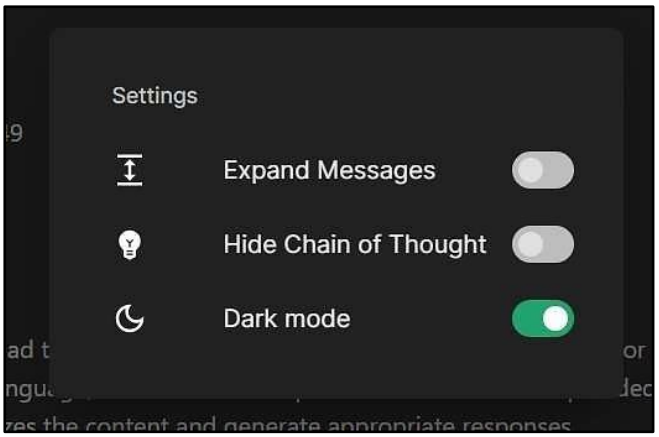
1. Gao Y, Baptista-Hon DT, Zhang K. The inevitable transformation of medicine and research by large language models: the possibilities and pitfalls. MEDCOMM-Future Medicine. 2023 Jun 1;2(2).
2. Peng B, Galley M, He P, Cheng H, Xie Y, Hu Y, Huang Q, Liden L, Yu Z, Chen W, Gao J. Check your facts and try again: Improving large language models with external knowledge and automated feedback. arXiv preprint arXiv:2302.12813. 2023 Feb 24.
3. Zhang P, Jaipersaud B, Ba J, Petersen A, Zhang L, Zhang MR. Classifying Course Discussion Board Questions using LLMs. InProceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 2 2023 Jun 29 (pp. 658-658).
4. Ling C, Zhao X, Lu J, Deng C, Zheng C, Wang J, Chowdhury T, Li Y, Cui H, Zhao T. Beyond one-model-fits-all: A survey of domain specialization for large language models. arXiv preprint

arXiv:2305.18703. 2023;1.
5. Yager KG. Domain-specific chatbots for science using embeddings. Digital Discovery. 2023;2(6):1850-61.
6. Chen YH, Senk SL, Thompson DR, Voogt K. Examining psychometric properties and level classification of the van Hiele Geometry Test Using CTT and CDM Frameworks. Journal of Educational Measurement. 2019 Dec;56(4):733-56.
7. Tripathi KP. A study of interactivity in human computer interaction. International Journal of Computer Applications. 2011 Feb 28;16(6):1-3.
8. Dehelean T, Nafornita C, Isar A. Estimate's Statistics in the Performance Evaluation of Extended Object Tracker. In2019 International Symposium on Signals, Circuits and Systems (ISSCS) 2019 Jul 11 (pp. 1-4). IEEE.
9. Dev A, Parmanand B. A Novel Feature Extraction Technique for Speaker Identification. International Journal of Computer Applications. 2011 Feb;16(6):0975-8887.
10. Jeon J, Lee S. Large language models in education: A focus on the complementary relationship between human teachers and ChatGPT. Education and Information Technologies. 2023 Dec;28(12):15873-92.

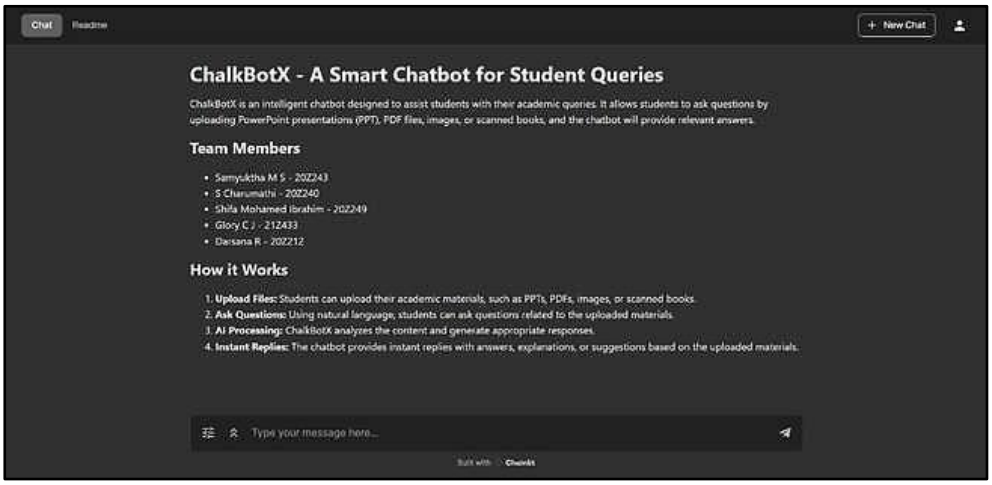
APPENDIX



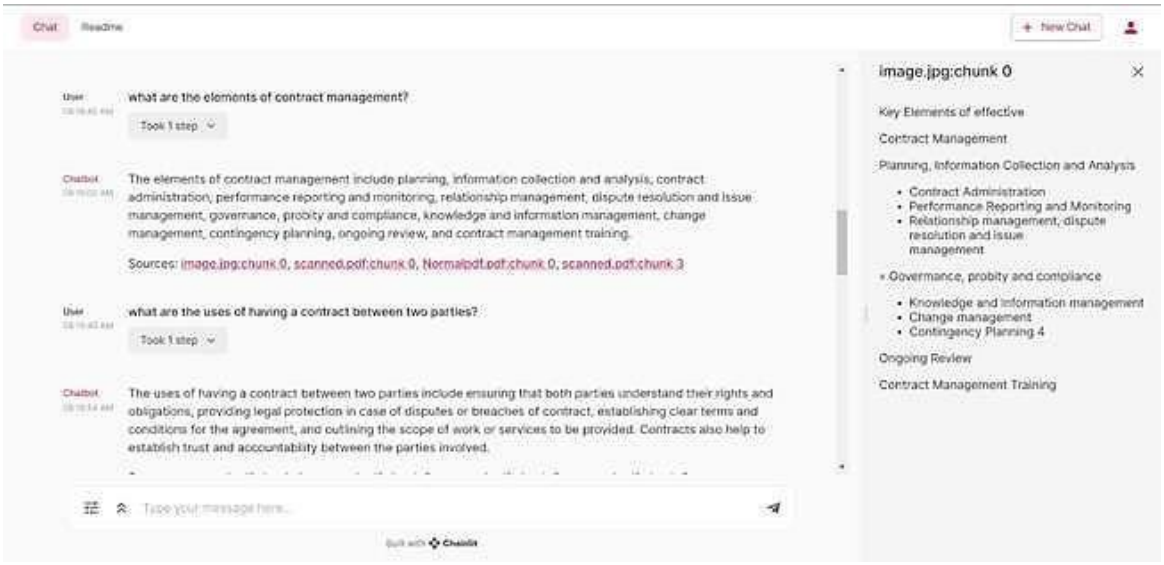
Appendix I. User Interface in Light Mode



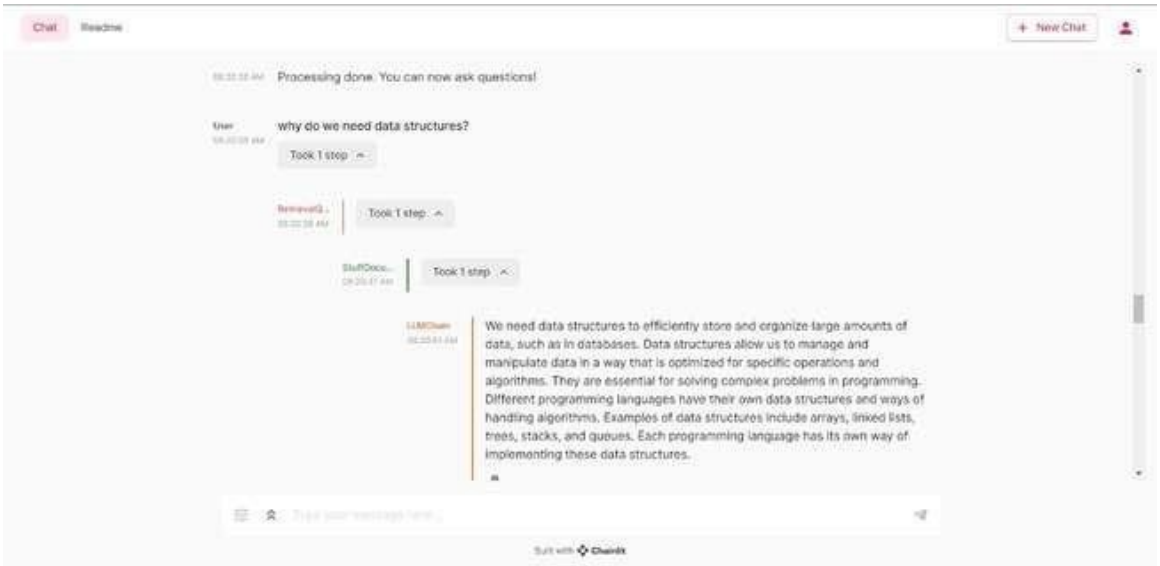
Appendix II. User Settings



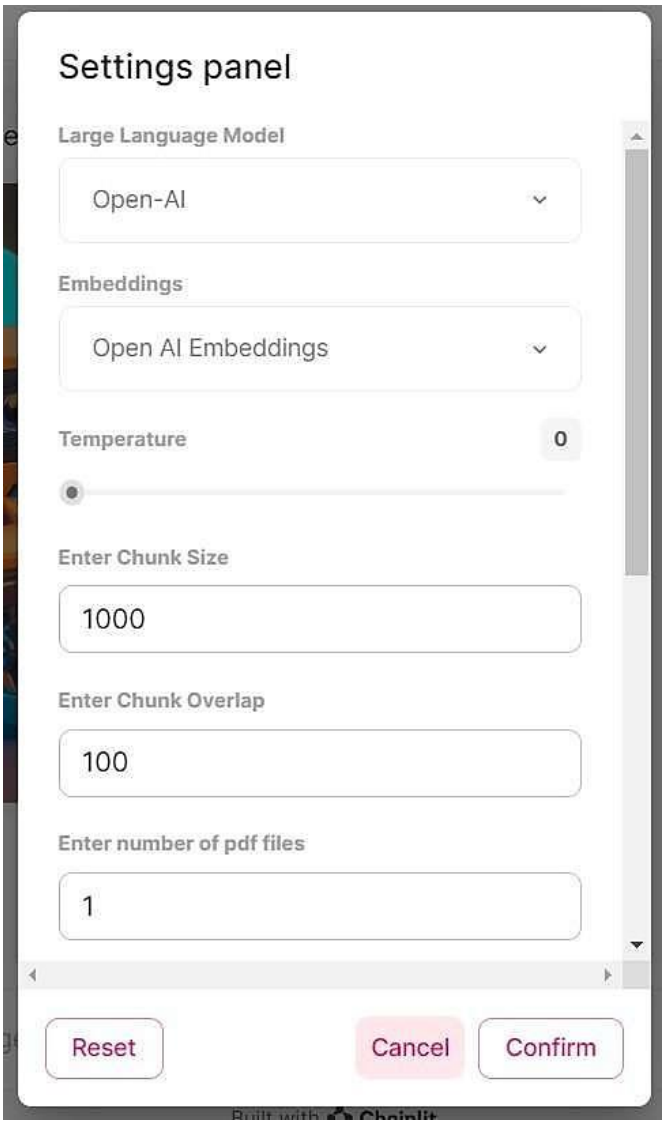
Appendix III. User Interface in Dark mode



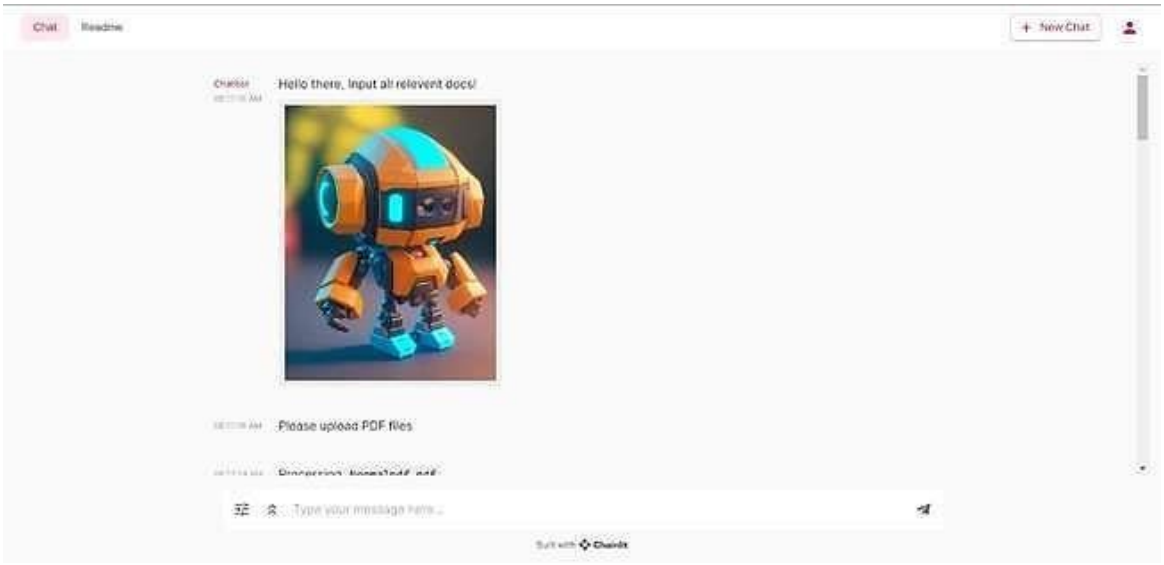
Appendix IV. Sources Displayed



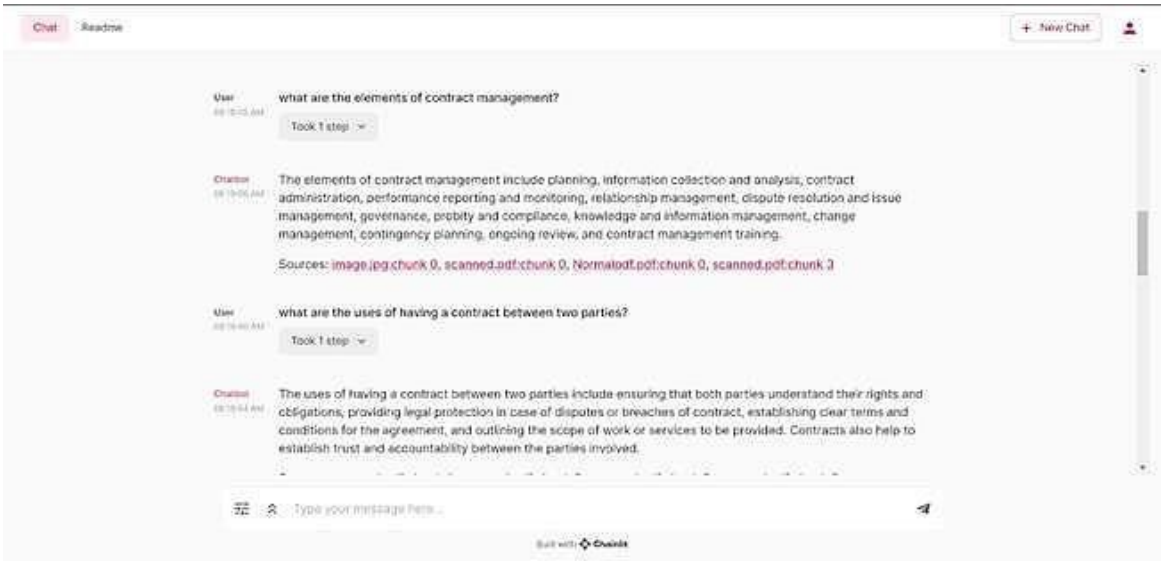
Appendix V. Chain of Thought



Appendix VI. Settings Panel



Appendix VII. ChatBot Starting Point



Appendix VIII. Chatbot Response

