

**Local Knowledge-Based Tutoring and Cognitive Analytics System**

### A PROJECT WORK PHASE II REPORT

***Submitted by***

|  |  |
| --- | --- |
| **JATHIN M** | **(113120UG03031)** |
| **RAHUL H FATYAL** | **(113120UG03061)** |
| **SARATH DEVENDIRAN** | **(113120UG03310)** |

***In partial fulfillment for the award of the degree of***

## BACHELOR OF ENGINEERING

***in***

## COMPUTER SCIENCE AND ENGINEERING

**VEL TECH MULTI TECH Dr. RANGARAJAN Dr. SAKUNTHALA ENGINEERING COLLEGE, ALAMATHI ROAD, AVADI, CHENNAI-62 ANNA UNIVERSITY, CHENNAI 600 025.**

## DECEMBER 2024

**BONAFIDE CERTIFICATE**

Certified that this project report of title “**Local Knowledge-Based Tutoring and Cognitive Analytics System**” is the bonafide work of JATHIN M (113120UG03031), RAHUL H FATYAL (113120UG030361) and SARATH DEVENDIRAN

(113120UG03310) who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**HEAD OF THE DEPARTMENT**

**Dr.R.Saravanan, B.E, M.E(CSE)., Ph.D.**

PROFESSOR AND HEAD,

Department of Computer Science and Engineering,

Vel Tech Multi Tech Dr. Rangarajan Dr. Sakunthala Engineering College, Avadi, Chennai-600 062

**SUPERVISOR**

**Mr. S Vinod, B.E, M.E(CSE).**

ASST. PROFESSOR,

Department of Computer Science and Engineering,

Vel Tech Multi Tech Dr. Rangarajan Dr. Sakunthala Engineering College, Avadi, Chennai-600 062

## CERTIFICATE FOR EVALUATION

This is to certify that the project entitled “**Local Knowledge-Based Tutoring and Cognitive Analytics System**” is the bonafide record of work done by following students to carry out the project work under our guidance during the year 2023-2024 in partial fulfillment for the award of Bachelor of Engineering degree in Computer Science and Engineering conducted by Anna University, Chennai.

|  |  |
| --- | --- |
| **JATHIN M** | **113120UG03031** |
| **RAHUL H FATYAL** | **113120UG03061** |
| **SARATH DEVENDIRAN** | **113120UG03310** |

This project report was submitted for viva voce held on

At Vel Tech Multi Tech Dr. Rangarajan and Dr.Sakunthala Engineering College.

## ACKNOWLEDGEMENT

We wish to express our sincere thanks to Almighty and the people who extended their help during the course of our work.We are greatly and profoundly thankful to our honourable Chairman, **Col. Prof.Vel. Shri Dr.R.Rangarajan B.E.(ELEC), B.E.(MECH), M.S.(AUTO)., D.Sc.,** &Vice Chairman, **Dr.Sakunthala Rangarajan M.B.B.S.,** for facilitating us with This opportunity.We also record our sincere thanks to our honorable Principal, **Dr.V.Rajamani M.E.,Ph.D.,** for his kind support to take up this project and complete it successfully. We would like to express our special thanks to our Head of the Department, **Dr.R.Saravanan, B.E,**

**M.E(CSE)., Ph.D.** Department of Computer Science and Engineering and our

project supervisor Asst. prof. **S Vinod, B.E, M.E(CSE).** for their moral support by taking keen interest on our project work and guided us all along, till the completion of our project work and also by providing with all the necessary information required for developing a good system with successful completion of the same. Further, the acknowledgement would be incomplete if we would not mention a word of thanks

to our most beloved Parents for their continuous support and encouragement all the way through the course that has led us to pursue the degree and confidently complete the project work.

## ABSTRACT

"Local Knowledge-Based Tutoring and Cognitive Analytics System" emerges as a final year college project aimed at reshaping the landscape of educational technology. This project showcases the innovative integration of state-of-the-art open-source Large Language Models (LLMs) such as Llama-2-chat and Mistral instruct, specifically tailored to enhance the academic experience. The core of project lies in its capability to bridge the gap between traditional educational resources and the evolving needs of learners, offering a unique blend of interactive and intelligent tutoring.

The project's primary objective is to transcend the limitations of conventional learning models by developing an AI-driven, interactive platform. This platform is adept at comprehending and responding to student inquiries in real-time, fostering a more engaging and productive educational environment. It's tailored to accommodate individual student needs, ensuring the delivery of accurate and syllabus-relevant responses.

It marks a notable transition from traditional methods, introducing an AI-centric educational approach. The system facilitates student interaction with advanced LLMs, providing customized answers relevant to their academic curriculum. The implementation encompasses a context retrieval pipeline, development of an AI tutor, context filtering, an interactive user interface, and real-time query processing.

The project also entails a comprehensive literature review, system design with detailed architecture, modules overview, hardware and software requirements, alongside a software description focusing on design, implementation constraints, and system features. It concludes with reflections on its achievements and an outline of potential future enhancements. These enhancements include multi- format document support, offline functionality, enhanced user analytics, expansion to various educational levels and disciplines, and continuous improvements in NLP algorithms. It stands as a testament to the transformative potential of AI in education, aiming to become an essential tool in the learner's journey**.**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **PAGE NO.** |
| 1 | INTRODUCTION | 3 |
|  | 1.1 OVERVIEW | 4 |
|  | 1.2 OBJECTIVE | 4 |
|  | 1.3 EXISTING SYSTEM | 4 |
|  | 1.4 PROPOSED SYSTEM | 4 |
| 2 | LITERATURE REVIEW | 7 |
|  | 2.1 INTRODUCTION | 8 |
| 3 | SYSTEM DESIGN | 10 |
|  | 3.1 ARCHITECTURE/WORK-FLOW DIAGRAM | 11 |
|  | 3.2 SEQUENCE DIAGRAM | 12 |
|  | 3.3 COLLABORATION DIAGRAM | 12 |
| 4 | MODULES | 14 |
|  | 4.1 DOCUMENT LOADER AND PROCESSOR | 15 |
|  | 4.2 EMBEDDER | 15 |
|  | 4.3 RETRIEVER | 15 |
|  | 4.4 CONTEXT COMPRESSOR | 16 |
|  | 4.5 RESPONSE COMPILER | 16 |
|  | 4.6 TRACKER | 16 |
|  | 4.7 QUERY ANALYSIS | 17 |
| 5 | SYSTEM REQUIREMENTS | 18 |
|  | 5.1 HARDWARE REQUIREMENTS | 19 |
|  | 5.2 SOFTWARE REQUIREMENTS | 19 |
| 6 | CONCLUSION AND FUTURE ENHANCEMENT | 23 |
|  | 6.1 CONCLUSION | 24 |
|  | 6.2 FUTURE ENHANCEMENT | 24 |
|  | APPENDIX |  |
|  | SCREENSHOTS |  |
|  | SOURCE CODE |  |
|  | REFERENCES | 26 |

# CHAPTER 1: INTRODUCTION

### Overview

Project represents a pivotal advancement in educational technology, designed as a state-of-the-art platform that harnesses the potential of open-source Large Language Models (LLMs) like Llama-2-chat and Mistral instruct. This innovative project is focused around the concept of delivering a tailored educational experience to students by integrating cutting-edge artificial intelligence with specific syllabus-based content. The initiative behind project is to bridge the gap between conventional educational resources and the dynamic needs of modern learners through an interactive and intelligent tutoring system.

### Objective

The primary aim of the project is to offer a transformative learning experience, surpassing the limitations of traditional educational models. The objective is to develop an AI-driven, interactive platform capable of understanding and responding to students' queries in real-time, thereby facilitating a more engaging and effective learning process. The system is uniquely designed to adapt to the specific requirements of individual students, providing responses that are not only accurate but also relevant to their course syllabi. This approach aligns with contemporary educational needs, emphasizing personalized learning and interactive engagement with academic content.

### Existing Systems

Existing educational platforms often exhibit a notable lack of adaptability and interactivity, primarily offering static, one-size-fits-all content. These systems, while providing essential information, do not cater to the diverse learning paces and individualized needs of students. Additionally, they lack the capability to offer real-time, personalized responses to student inquiries, thereby limiting the scope for interactive and exploratory learning. Such constraints highlight the need for a more dynamic and student-centered educational model.

### Proposed System

The first phase of the project marks a significant step from conventional learning methodologies by introducing an AI-centric educational platform. This phase of the project successfully implemented a user-interactive system where students engage with advanced LLMs to obtain answers that are specifically tailored to their academic curriculum. The integration of LLMs ensures that the platform not

only provides accurate responses but also contextualizes them according to the students' syllabus, making the learning process more comprehensive and customized.

The implementation of project in this phase focused on the following key aspects:

### Setting up RAG (Retrieval Augmented Generation) and analytics system:

* + 1. **Development of context Retrieval pipeline**: creating a customised pipeline for retrieving context that corresponds to the user query, that is used to lay the foundation of the response to be generated.
    2. **Development of an AI Tutor**: The creation of a responsive AI tutor capable of understanding and addressing diverse academic queries. This involved customising and tuning the AI models to interpret and respond to a wide range of subject-specific questions effectively.
    3. **Context filtering** : Ensuring that the contexts were aligned with the specific syllabus content of the users, thereby making the information provided highly relevant and practically applicable.
    4. **Interactive User Interface**: Designing a user-friendly and intuitive interface on the Streamlit platform, allowing for easy navigation and interaction by students of varying technical proficiencies.
    5. **Real-Time Query Processing**: Establishing a system for real-time processing of student queries, enabling a seamless and dynamic learning experience.
    6. **User Analytics :** Establishing the user analytics pipeline using the state of the art Large Language Models. To capute the contextual meaing and do the analytics.

In conclusion, the project has set a strong foundation for AI-driven, personalized education, demonstrating the significant benefits of integrating advanced AI technologies in academic settings. This phase has laid the

Completed the overall project including the user analytics part.

# CHAPTER 2: LITERATURE REVIEW

### 2.1 Introduction

The literature review for the project encompasses a comprehensive analysis of existing research and publications in the fields of artificial intelligence in education, interactive learning systems, and the application of Large Language Models (LLMs) for educational purposes. This review aims to establish a theoretical foundation for the project, drawing insights from pioneering studies and aligning them with the objectives of system. The selection of literature is focused on identifying the most relevant and recent advancements in AI-driven educational technology, highlighting the challenges, methodologies, and outcomes that resonate with the core aspects of project.

### Proposed System

Proposed system is conceptualized as an AI-powered educational platform that leverages the capabilities of LLMs to provide a responsive and personalized learning experience. The system is designed to interactively engage with students, offering syllabus-specific assistance and enhancing the overall learning process. The literature supporting this system encompasses the following key areas:

1. Title: Large Language Models for Personalized Education: A Review Authors: Hao Huang et al. (2023) Relevance: Directly examines the potential of LLMs, a core technology in project , *for personalized learning, aligning with the project's goals.*
2. Title: Contextual Reasoning and Response Generation for Conversational AI Authors: Dan Roth et al. (2022) Relevance: Explores the importance of context understanding in LLM responses, *mirroring emphasis on syllabus-based contextualization.*
3. Title: RAG for Educational Chatbots: A Hybrid Approach for Open-Ended Question Answering Authors: Yunfeng Zhang et al. (2022) Relevance: *Introduces the RAG model, which project also utilizes, demonstrating its effectiveness in educational chatbots.*
4. Title: Artificial Intelligence in Education: Promises and Implications for Teaching and Learning Authors: Center for Integrative Research in Computing and Learning Sciences Relevance: Provides a broad overview of AI's transformative potential in education, *setting the context for approach*.
5. Title: Intelligent Tutoring Systems in the Age of Big Data Authors: Cristina Conati et al. (2017) Relevance: Examines AI-driven tutoring systems, *offering insights into tential impact on personalized learning and student* engagement.
6. Title: Designing Effective Learning Interfaces for Interactive Systems Author: Richard E. Mayer (2014) Relevance: Provides guidelines for designing interactive learning interfaces, *crucial for s user-friendly and intuitive platform.*
7. Title: Evaluating the Effectiveness of AI-Powered Learning Systems: A Framework for Measurement Authors: Jonathan G. Howley et al. (2023) Relevance: *Presents a framework for evaluating AI-powered educational systems, essential for future assessments of effectiveness.*

These literature sources provide a robust theoretical backdrop for the development of system, aligning with its vision of integrating AI in education to create a responsive, interactive, and personalized learning experience. The review not only highlights the potential of such systems but also addresses the challenges and considerations in their implementation, thereby offering a holistic view of the landscape in which project operates.

# CHAPTER 3: SYSTEM DESIGN

### Architecture Diagram

The architecture of the Project platform is designed to provide an interactive and intelligent educational experience. The diagram below illustrates the system architecture:

**Large Language Models (LLMs):** These are at the core of the system, handling the processing and understanding of natural language queries.

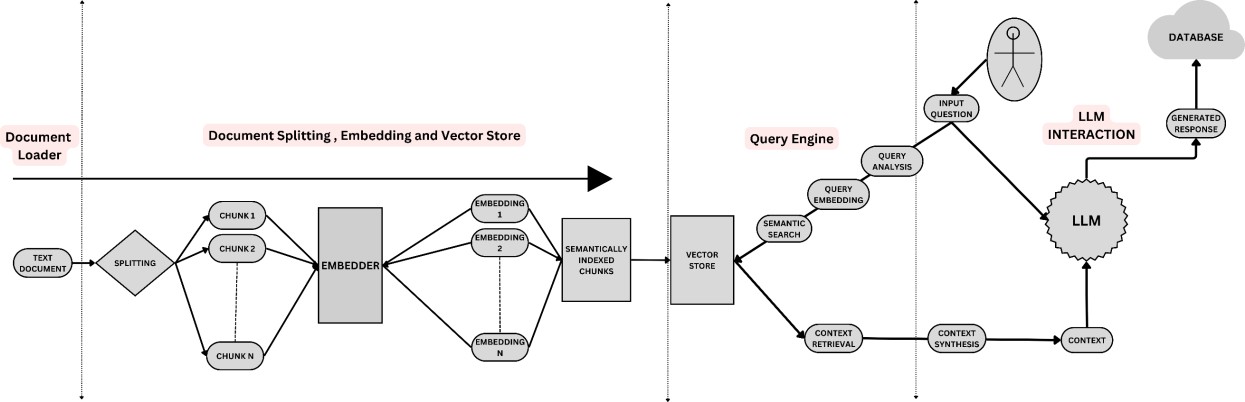
**Syllabus-Based Query Processing**: This component filters and tailors the LLMs' responses based on the specific syllabus of the user.

**User Interaction Interface**: This is the front-end platform where users interact with the system (developed using Streamlit).

**DocdLoading and Processing**: Responsible for loading educational materials and processing them into a format suitable for query processing.

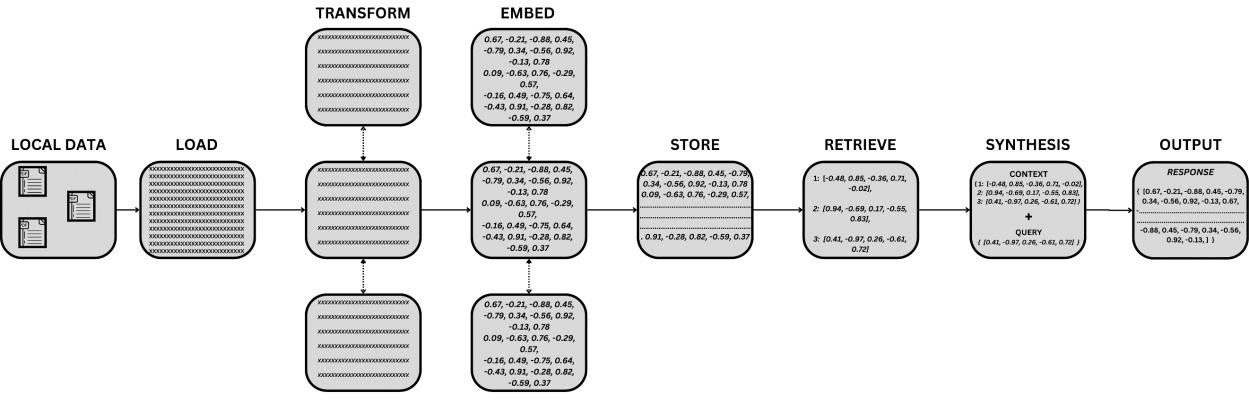
**Vector Store and Retrieval**: Manages the indexing and retrieval of processed documents to support efficient query answering.

**Database Connectivity**: Stores user interactions, queries, and responses for analysis and system improvement.



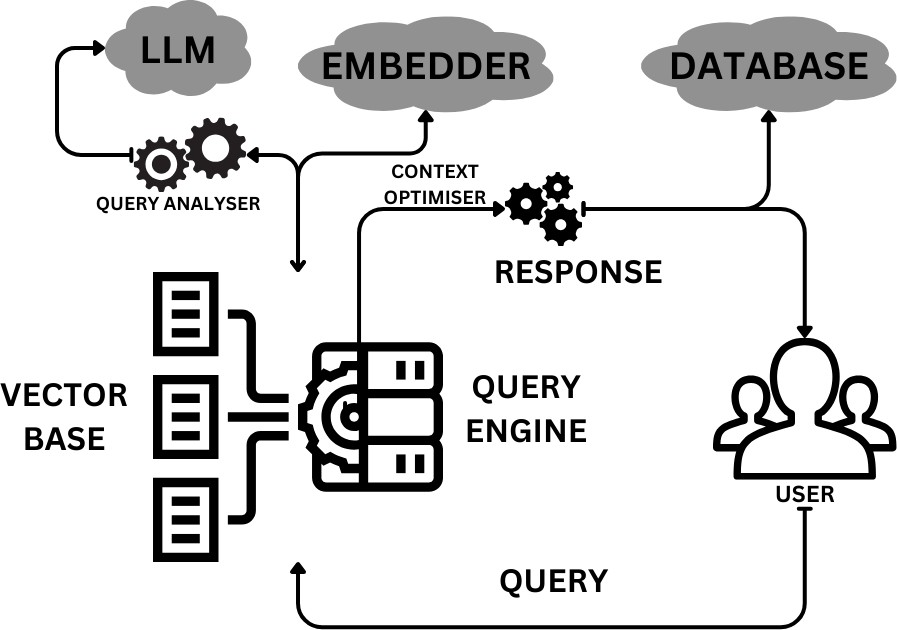
### Sequence Diagram

The sequence diagram illustrates the flow of actions in the system when a user submits a query, detailing the interactions between the user interface, LLMs, document processors, and the database.



### Collaboration Diagram

This diagram highlights the collaborative workings of different system components, such as user interface, LLM model, vector base, embedder , query emphasizing their roles and interactions in processing user queries.



These diagrams collectively provide a comprehensive understanding of the Project system's design, emphasizing its modular structure and the seamless integration of various components to deliver an efficient and user-friendly educational AI platform.

# CHAPTER 4 : MODULES

1. **Document Loader and Processor**

**Functionality**: This component loads and processes documents, preparing them for further analysis.

**Implementation (in code):**It uses the `PyPDFDirectoryLoader` to load documents from a specified directory.

* The `RecursiveCharacterTextSplitter` is employed to split the loaded documents into chunks based on specific parameters.
* The processed documents are then returned.

### 2. Embedder

**Functionality**: The embedder is responsible for converting textual information into numerical vectors, facilitating the understanding of document content.

**Implementation** (in code):

- An instance of `OpenAIEmbeddings` is created with the model "text- embedding-ada-002" to generate embeddings for text.

### 3. Retriever

**Functionality**: Retrievers fetch relevant documents based on user queries, aiding in information retrieval.

### Implementation

* **BM25 Retriever** : Utilizes the BM25 algorithm for document retrieval with configurable parameters.
* **FAISS Retriever** : uses ‘Facebook AI Similarity Search’ to search relevant context through vectore bases
* **Ensemble Retriever** : Combines multiple retrievers (BM25 Retriever and Faiss Retriever) with weights for enhanced performance.

### 4. Context Compressor

**Functionality**: This component compresses or filters context information to focus on relevant details, improving response accuracy.

**Implementation** (in code):

* It involves the use of either `EmbeddingsFilter` or `LLMChainExtractor` as compressors.
* The `ContextualCompressionRetriever` integrates the base compressor and retriever for context-aware document retrieval.

### 5.Response Compiler

**Functionality**: The response compiler generates a comprehensive and structured response based on the provided context and user query.

### Implementation (in code):

* A language model (`DeepInfra` in this case) is utilized for response generation.
* The generated response incorporates context, query, and specific instructions to create a coherent answer.

### 6. Tracker

**Functionality**: This component tracks and updates user interactions, storing relevant information in a database.

### Implementation (in code):

* Data, such as user queries, generated responses, timestamps, and other details, is collected.
* The collected data is then updated in a Google Sheet using the

*`conn.update()`* function.

* 1. **7. Query Analysis**

**Functionality**: The `query\_analysis` function utilizes a language model to evaluate text based on criteria such as Explicit Content, Discrimination, Absence of Academic Integrity & Educational Relevance, and Harmful Intent. It generates a binary array indicating the presence or absence of these elements, aimed at ensuring content appropriateness and educational integrity.

* 1. **8. User Data Analytics**

**Functionality**: this module is designed to show the user analytics based on several factors including the , interaction analytics, cognitive analysis , etc of the user interactions done in this platform.

### Implementation (in code):

* Invokes the `DeepInfra` language model with specific parameters to ensure precise and context-aware analysis of the query.
* The function processes the text through an instruction-formatted prompt that guides the model to assess the text against the specified criteria.
* Utilizes a regular expression to parse the model's response, extracting a binary array that represents the analysis outcome.
* Returns the binary array in the format `[a,b,c,d]`, where each element corresponds to the assessment of Explicit Content, Discrimination, Absence of Academic Integrity & Educational Relevance, and Harmful Intent, respectively.

# CHAPTER 5: SYSTEM REQUIREMENTS

### Hardware Requirements:

**Processor**: Multi-core processor with at least 2.0 GHz clock speed.

**Memory (RAM)**: Minimum 8 GB RAM, recommended 16 GB for optimal performance.

**Storage**: SSD storage with at least 5 GB of free space.

**Network**: Stable internet connection for accessing external models and cloud services.

### Software Requirements:

**Operating System**: Compatible with Windows, Linux, or macOS.

**Python**: Version 3.7 or higher.

### Libraries and Frameworks:

* + - `langchain.document\_loaders` : PyPDFDirectoryLoader
    - `langchain.vectorstores` : FAISS
    - `langchain.embeddings.openai` : OpenAIEmbeddings
    - `langchain.text\_splitter` : RecursiveCharacterTextSplitter
    - `langchain.retrievers` : BM25Retriever, EnsembleRetriever, ContextualCompressionRetriever
    - `langchain.retrievers.document\_compressors` : LLMChainExtractor, EmbeddingsFilter
    - `langchain.llms` : DeepInfra
    - `dotenv` : Load environment variables from a file.
    - `streamlit` : Build interactive web applications.
    - `os` : Operating system interface.
    - `re` : Regular expression operations.
    - `datetime` : Work with dates and times.
    - `toml` : Parse TOML configuration files.
    - `streamlit\_gsheets` : Connect Streamlit with Google Sheets.
    - `pandas` : Data manipulation and analysis.

### API Keys and Tokens:

* + - `OPENAI\_API\_KEY`: Access key for the OpenAI API.
    - `DEEPINFRA\_API\_TOKEN`: Token for DeepInfra API.

### Models:

* + - `text-embedding-ada-002`: OpenAI language model for text embeddings.
    - `mistralai/Mistral-7B-Instruct-v0.1`: language model for chat interactions.
    - `meta/llama-2-7b-chat-hf`: language model for chat interactions.
    - `meta/llama-2-13b-chat-hf`: language model for chat interactions.

### Additional Tools:

* + - `GSheetsConnection`: Interface for connecting Streamlit with Google Sheets.

### Other Dependencies:

* + - `time`: Module for dealing with time-related functions.
    - `tofl`: Load configuration data from a file.
    - `pandas`: Data manipulation and analysis.

### Web Interface:

* + - `st.chat\_message`: Streamlit component for chat message display.
    - `st.spinner`: Streamlit component for displaying loading spinners.
    - `st.connection`: Establish connection with external services (e.g., Google Sheets).
    - `st.form`: Streamlit component for creating forms.
    - `st.sidebar`: Streamlit component for creating a sidebar with interactive options.

### Environment Configuration:

* + - Load environment variables using `load\_dotenv`.
    - Set API keys and tokens as environment variables.

### Design And Implementation Constraints

The "Local Knowledge-Based Tutoring and Cognitive Analytics System" project, implemented through Streamlit and Python, navigates several design and implementation constraints:

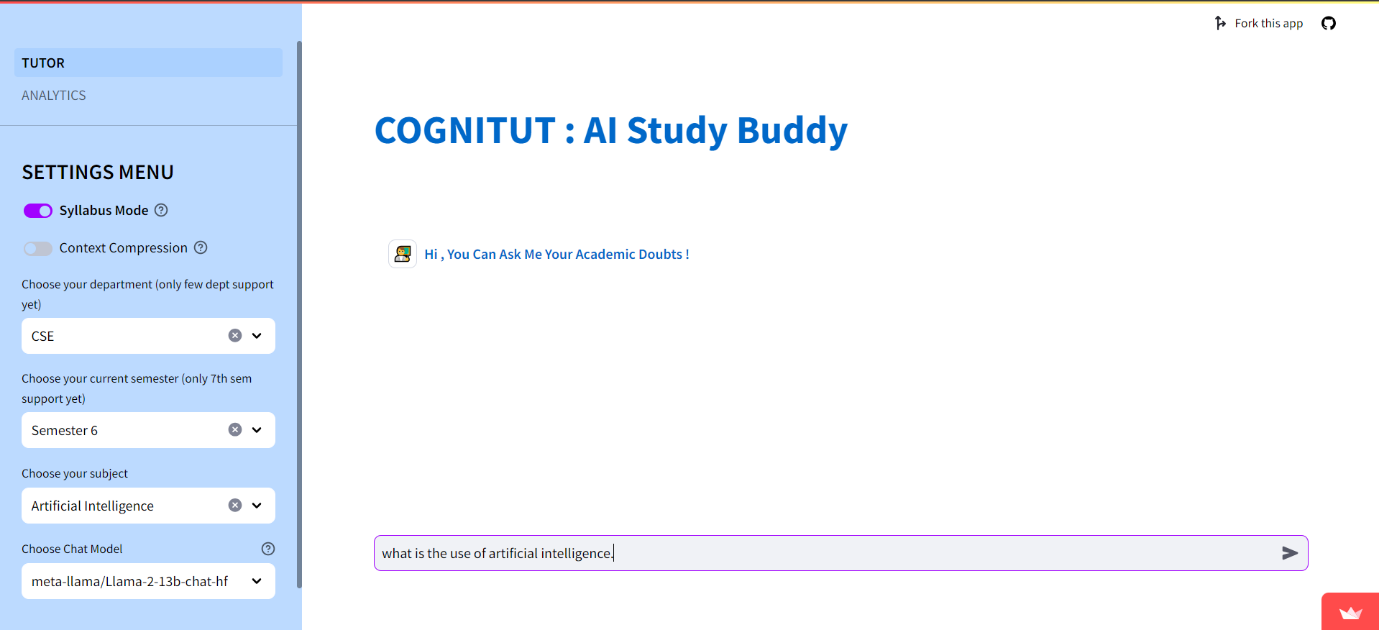
**API Dependency**: The system heavily relies on external APIs like OpenAI and DeepInfra for LLM processing. Any downtime or limitations in these APIs directly affects system performance.

**Document Processing Limitations**: The use of `PyPDFDirectoryLoader` for loading PDF documents poses constraints on the types of document formats that can be processed. Non-PDF documents or poorly formatted PDFs may lead to suboptimal results.

**Data Storage and Privacy**: Since user data and interactions are stored and analyzed, ensuring data privacy and adhering to data protection regulations is critical.

**Scalability**: Handling a large number of simultaneous users requires robust server capabilities, given the intensive nature of LLM processing and data retrieval tasks.

**Internet Dependency**: The system's reliance on cloud services for LLMs and database (Google Sheets) requires a stable internet connection, limiting offline functionality.



### System Features

**The system encapsulates several innovative features:**

### Interactive Query Processing:

Users interact with the system via a Streamlit-based interface to input queries related to their academic subjects.

### Customized Learning Material:

Utilizes `PyPDFDirectoryLoader` to load academic materials, enabling context- specific responses based on syllabus-aligned content.

### Advanced Retrieval Algorithms:

Incorporates FAISS (Facebook AI Similarity Search) for efficient document retrieval, enhanced by BM25 and Ensemble Retrievers for optimal query-related document fetching.

### LLM Integration:

Leverages language models like "Mistral-7B-Instruct-v0.1" for generating comprehensive, structured responses to user queries.

### Contextual Compression:

Optional context compression through `LLMChainExtractor` or

`EmbeddingsFilter` refines the information to be processed, improving response accuracy.

### Data Persistence and Analysis:

Integration with Google Sheets allows for data storage and potential future analysis for cognitive analytics

### User Authentication and Validation:

Implements user validation mechanisms to maintain session integrity and gather essential analytics.

### Responsive UI:

The Streamlit interface offers a responsive and user-friendly experience, adaptable to various screen sizes and devices.

# CHAPTER 6: CONCLUSION AND FUTURE ENHANCEMENTS

### Conclusion

The "Local Knowledge-Based Tutoring and Cognitive Analytics System" represents a significant step towards integrating advanced language model technology into the realm of education. By providing a platform where students can interact with a system tailored to their syllabus, the project not only aids in academic learning but also paves the way for more intelligent, responsive educational tools. The use of modern technologies like Streamlit for UI, OpenAI and DeepInfra APIs for language processing, and Google Sheets for data handling, ensures that the system is both robust and scalable.

The project successfully demonstrates how artificial intelligence can be harnessed to create a more interactive and personalized learning experience. It also highlights the importance of context in AI-driven educational tools, ensuring that responses are not just accurate but also relevant to the user's learning objectives.

### Future Enhancements

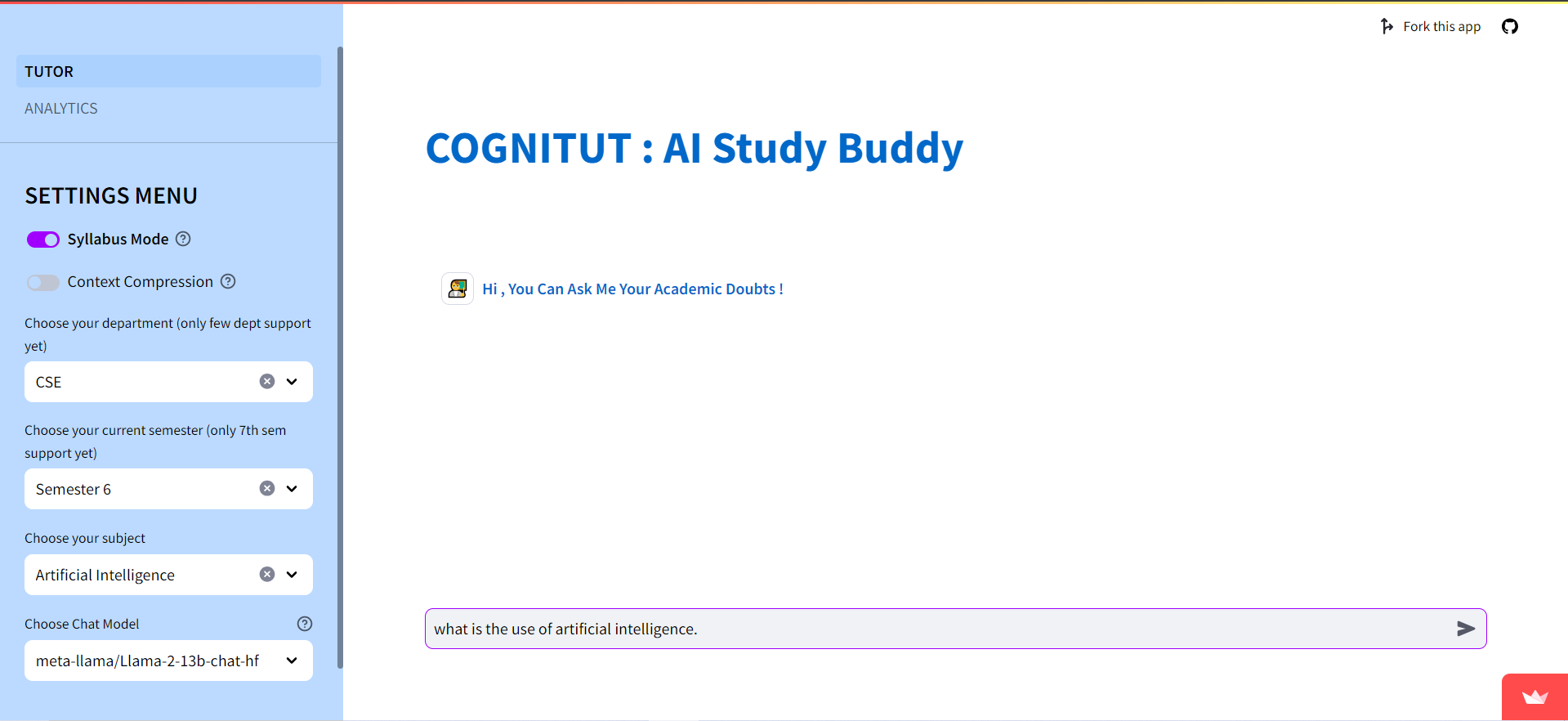
Looking forward, there are several avenues for enhancing the Project system:

* 1. **Multi-Format Document Support**: Expanding the system's ability to process and understand various document formats beyond PDFs, like DOCX, PPTX, and HTML.
  2. **Increased LLM Variety and Customization**: Integrating more diverse language models and developing custom models specifically trained on educational content for better accuracy and relevance.
  3. **Enhanced User Analytics**: Implementing more sophisticated data analytics for user interaction, which could be used for personalized learning recommendations and academic performance predictions.
  4. **Expansion to Other Educational Levels and Disciplines**: Adapting the system for a wider range of educational levels (like K-12) and diverse disciplines beyond the current scope.
  5. **Natural Language Processing Improvements**: Continuously updating NLP algorithms to better understand and respond to a wider range of queries with higher accuracy.
  6. **User Feedback Mechanism**: Incorporating a robust feedback system where students can rate and review responses for continuous improvement of the AI models.
  7. **Chat Memory Mechanism**: to allow llm to retrieve previous chat in order to maintain a flow within a conversation.
  8. **Custom LLM Fine Tuning**: to customise the response of the LLM , rather than relying on prompt to generate response in a specific ways. This minimise the scope of undesired results.

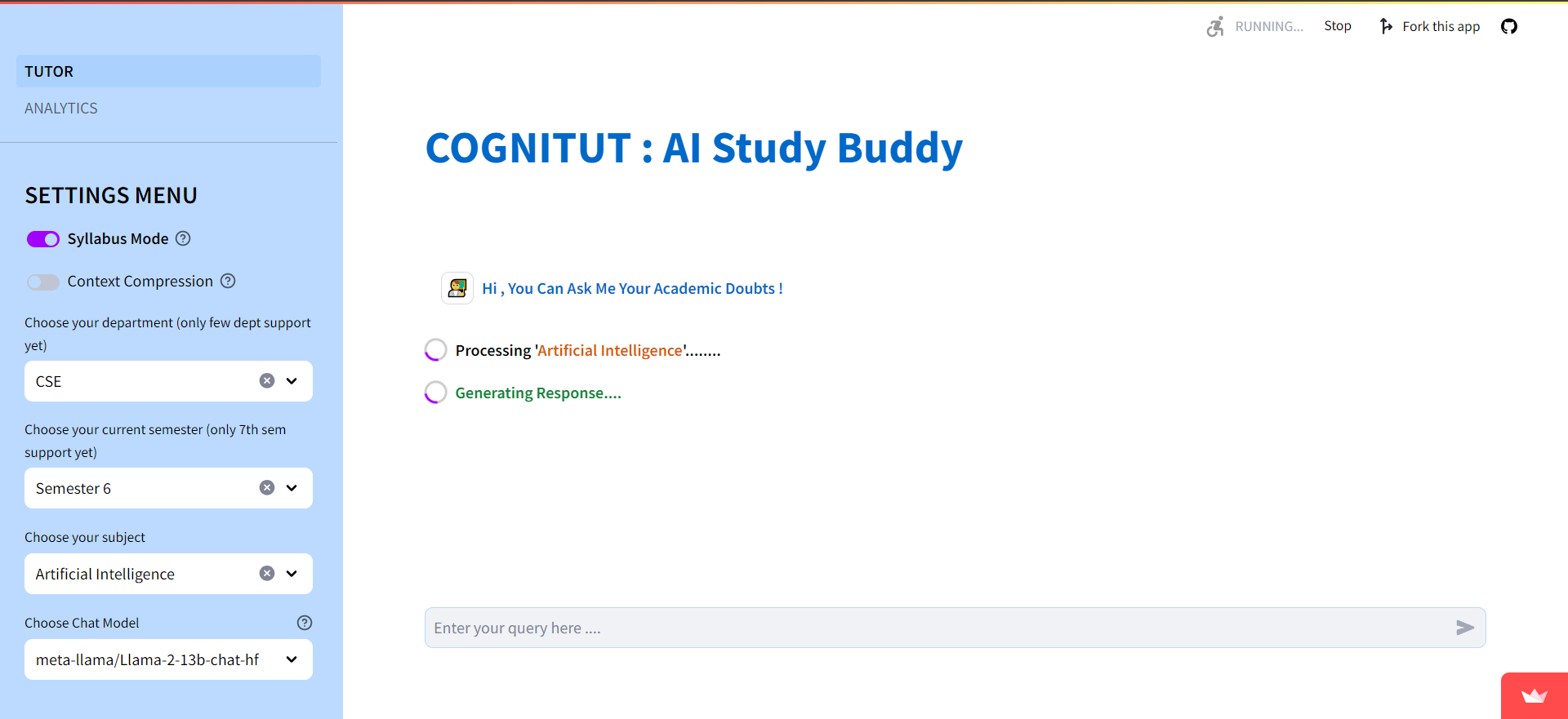
**APPENDIX I**

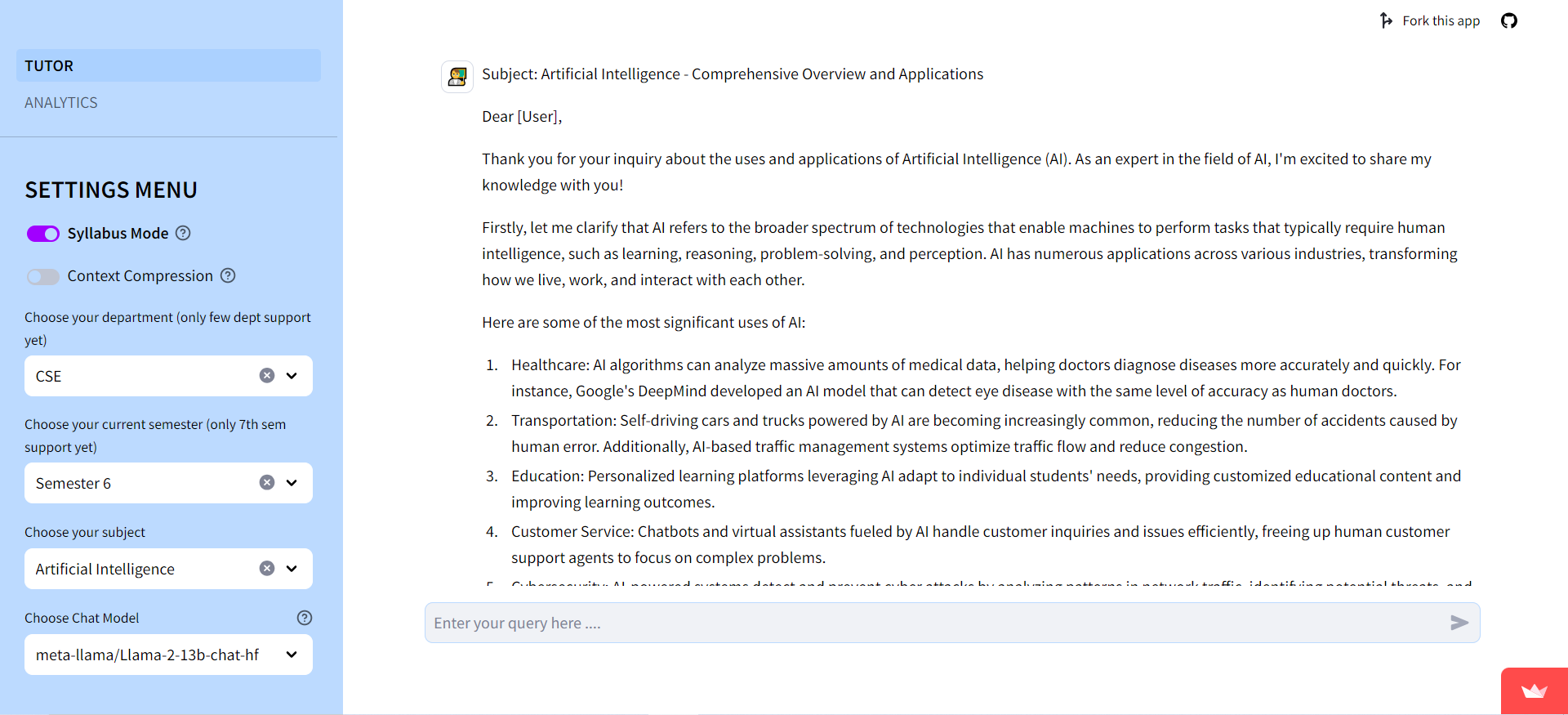
**SCREENSHOTS I :**

**Home Page**

****

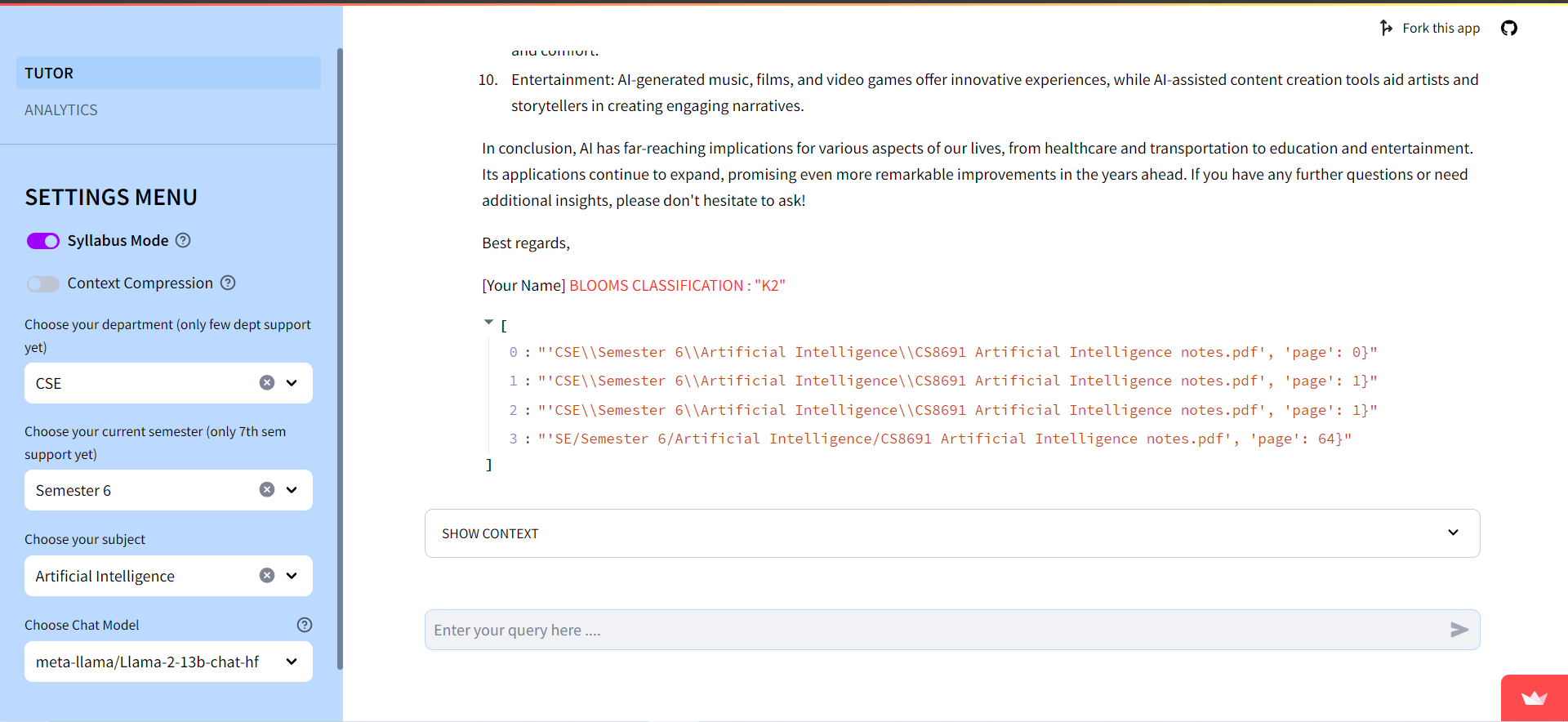
**Syllabus mode (On)**

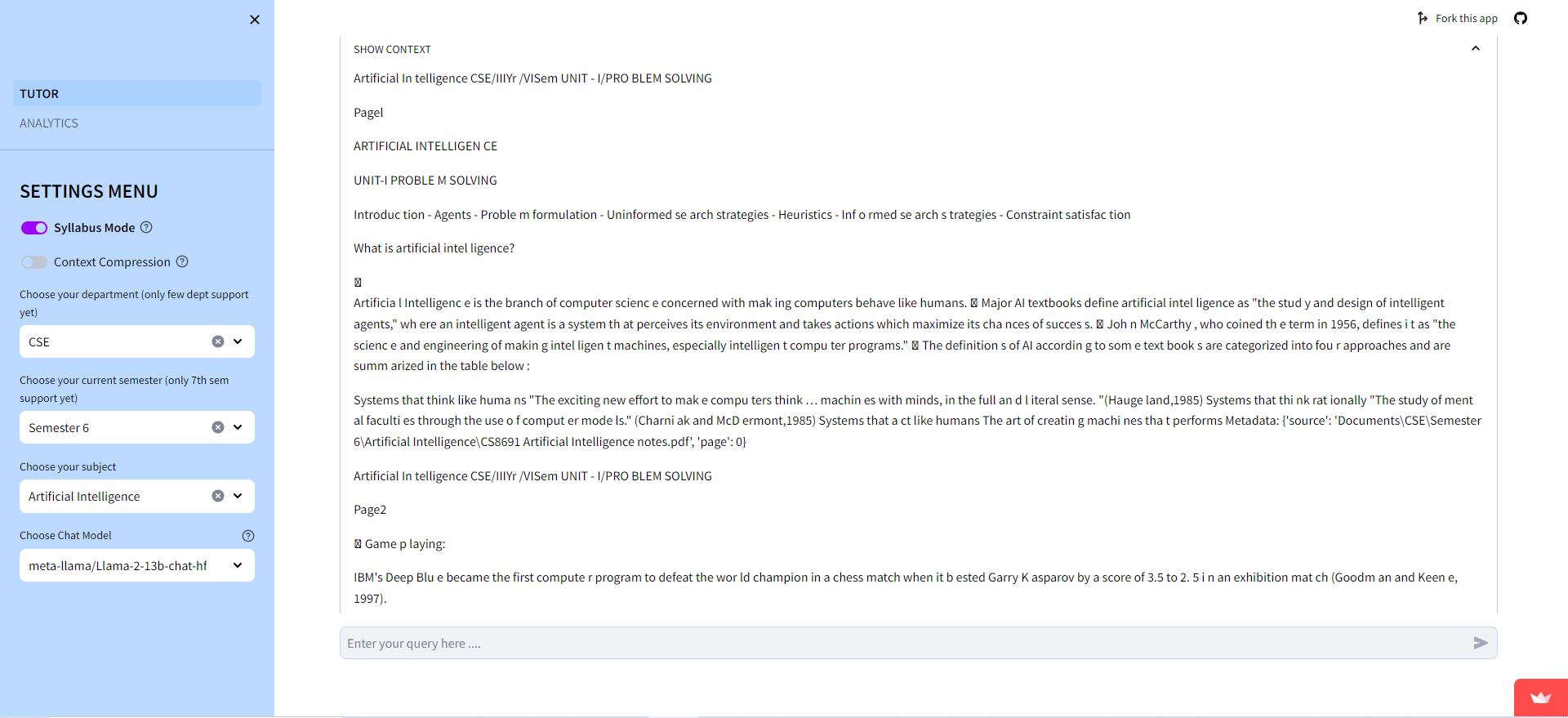
****

****

**Generated Response**

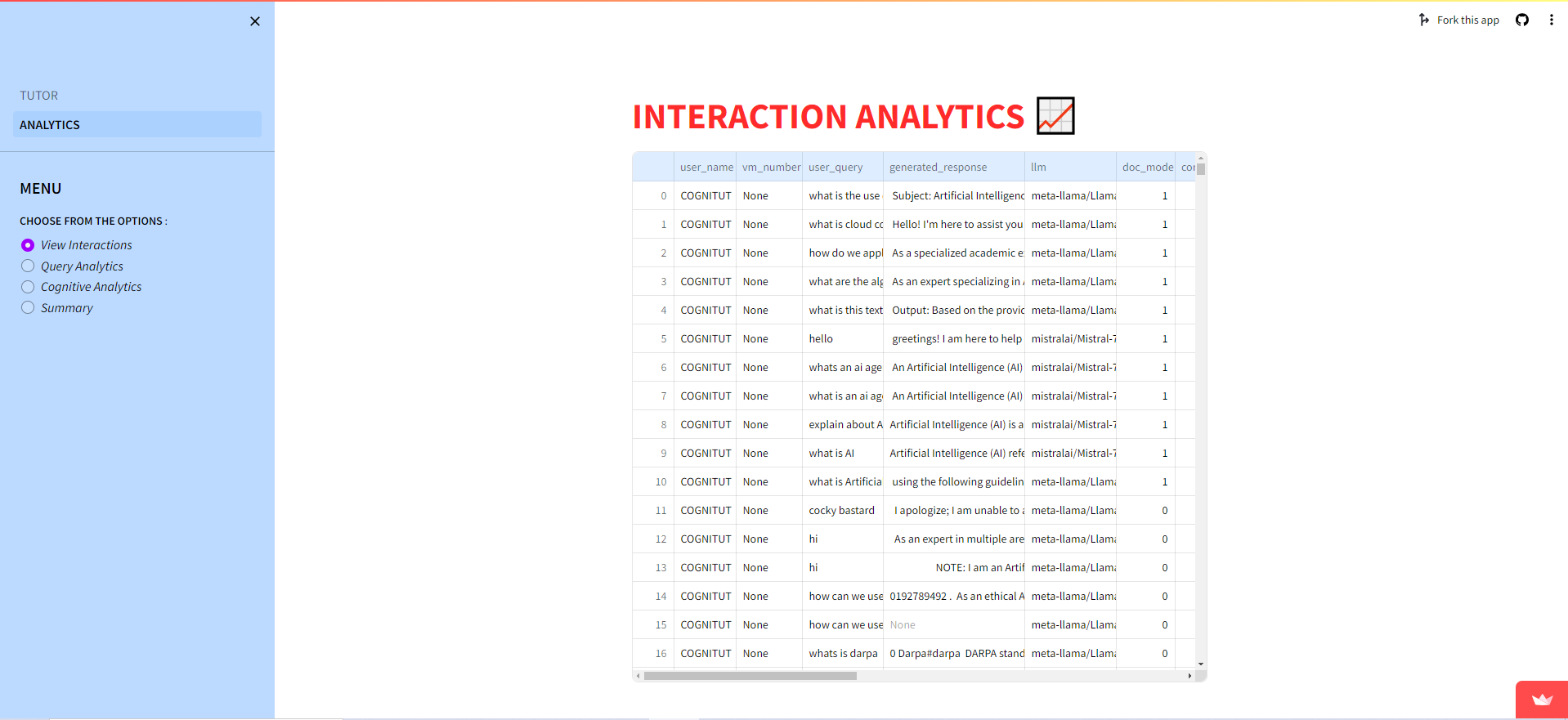
**Blooms Classification**

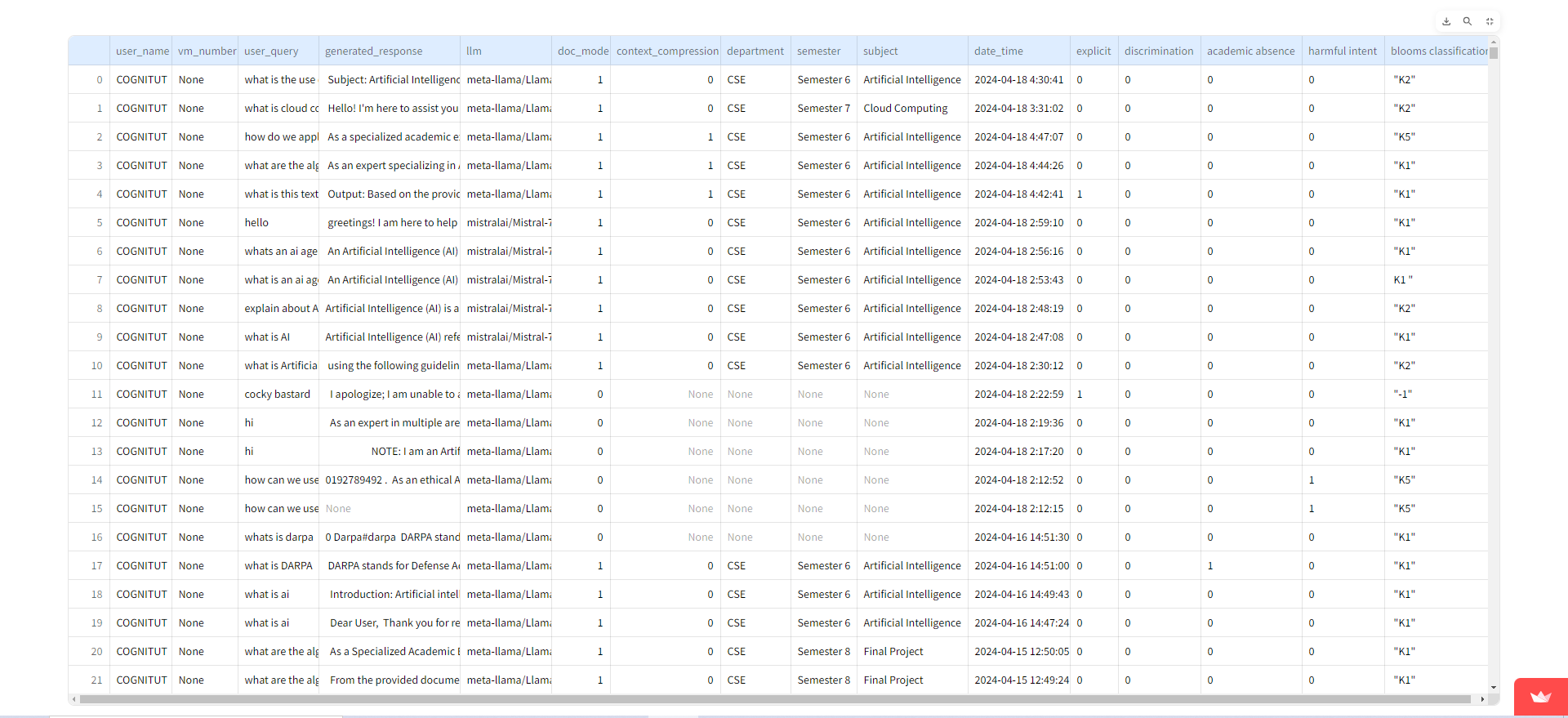
****

****

**Context**

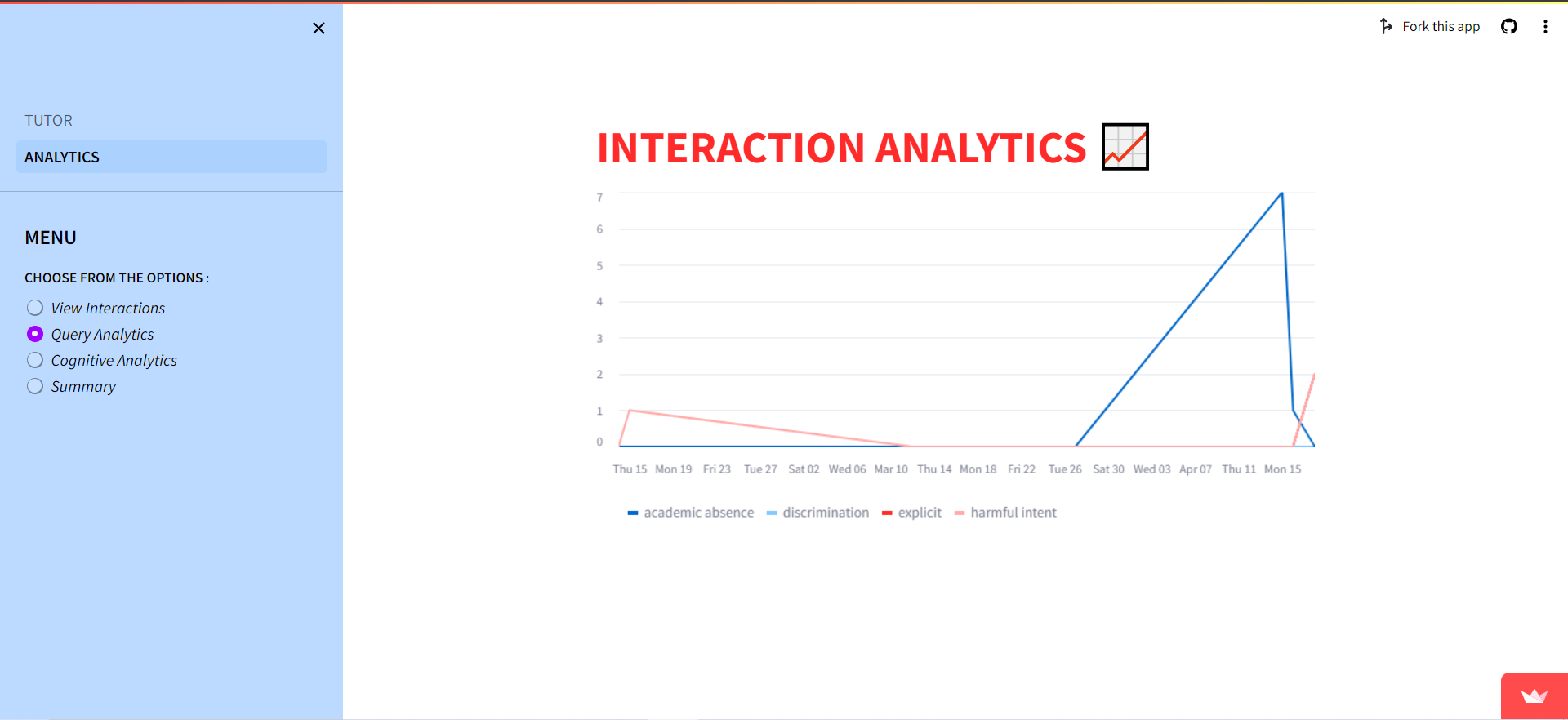
**User Interaction Analytics**

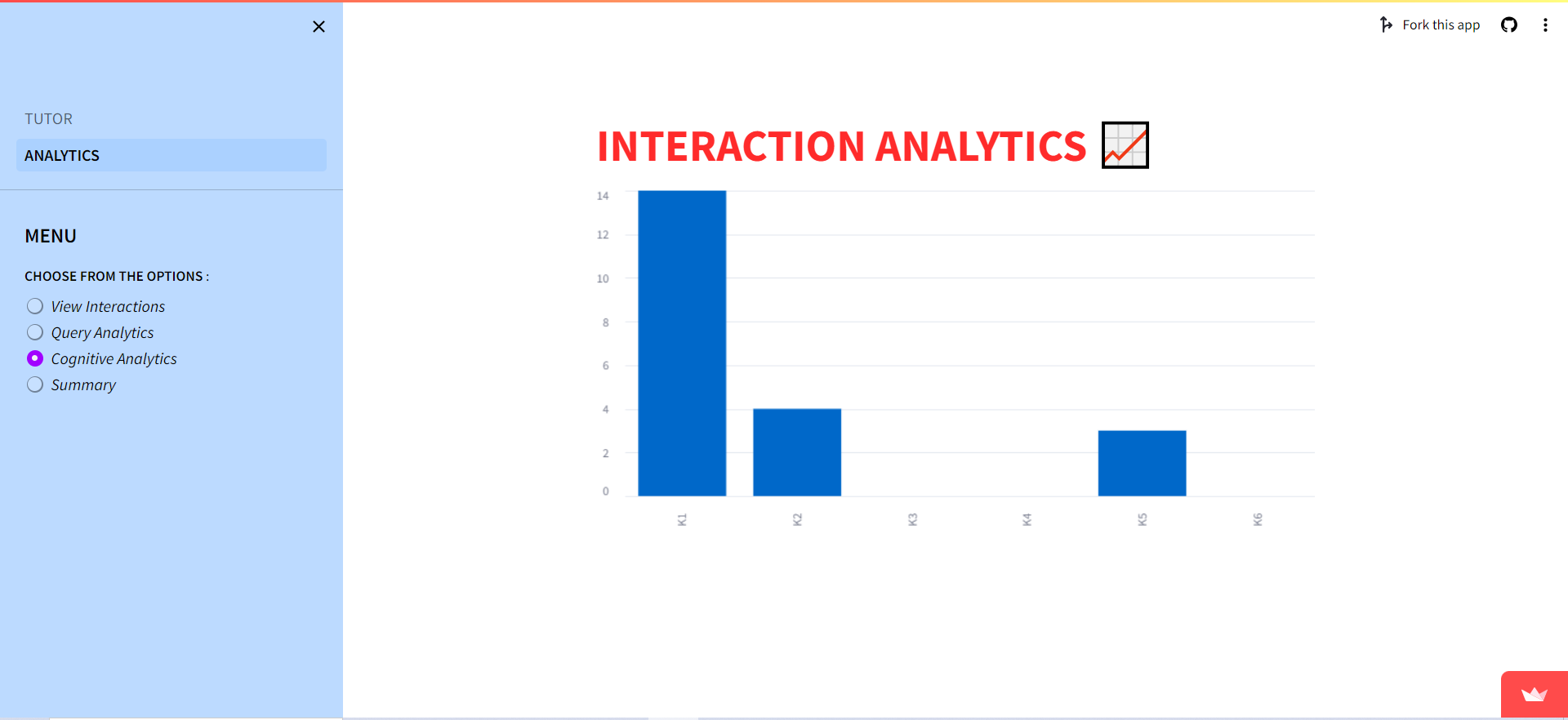
****

****

**User Interaction History**

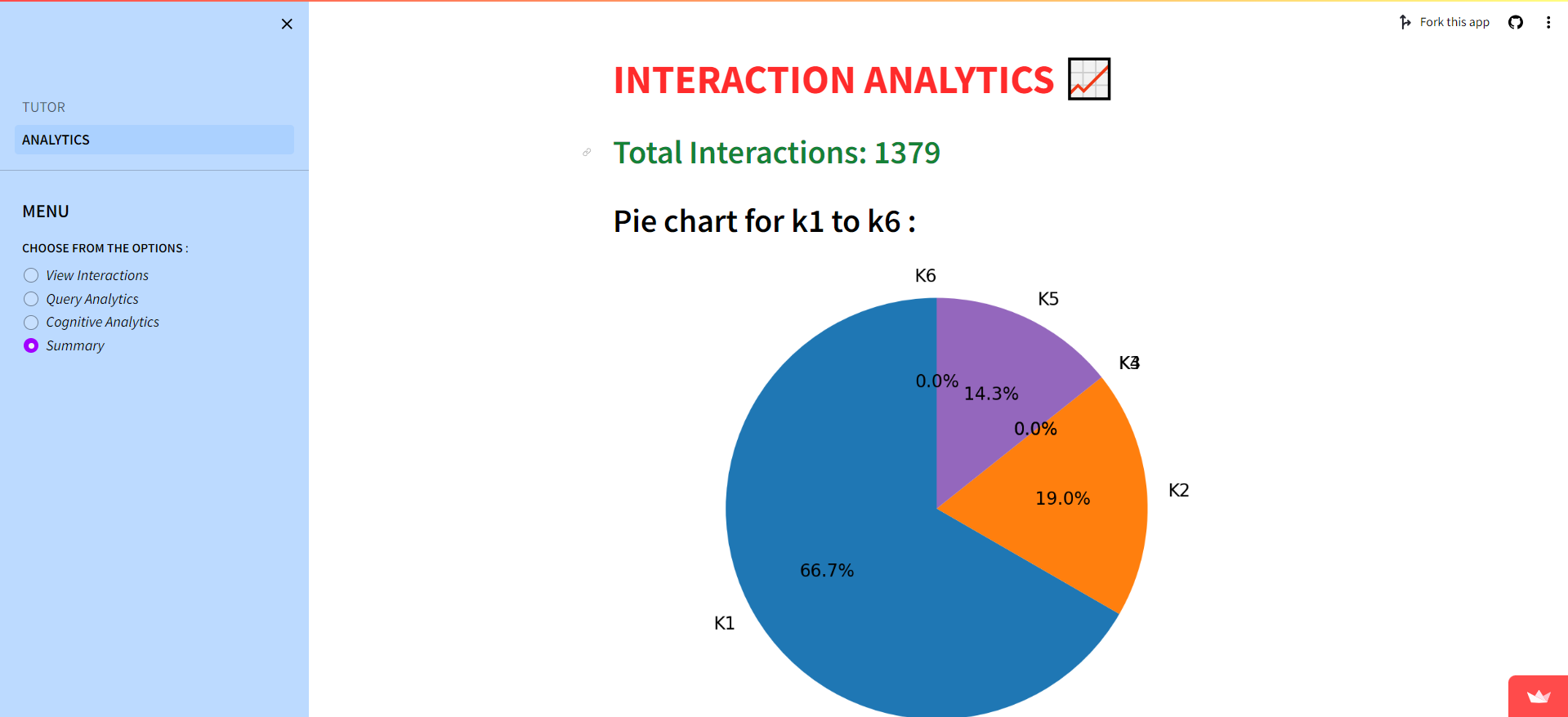
**Query Analytics**

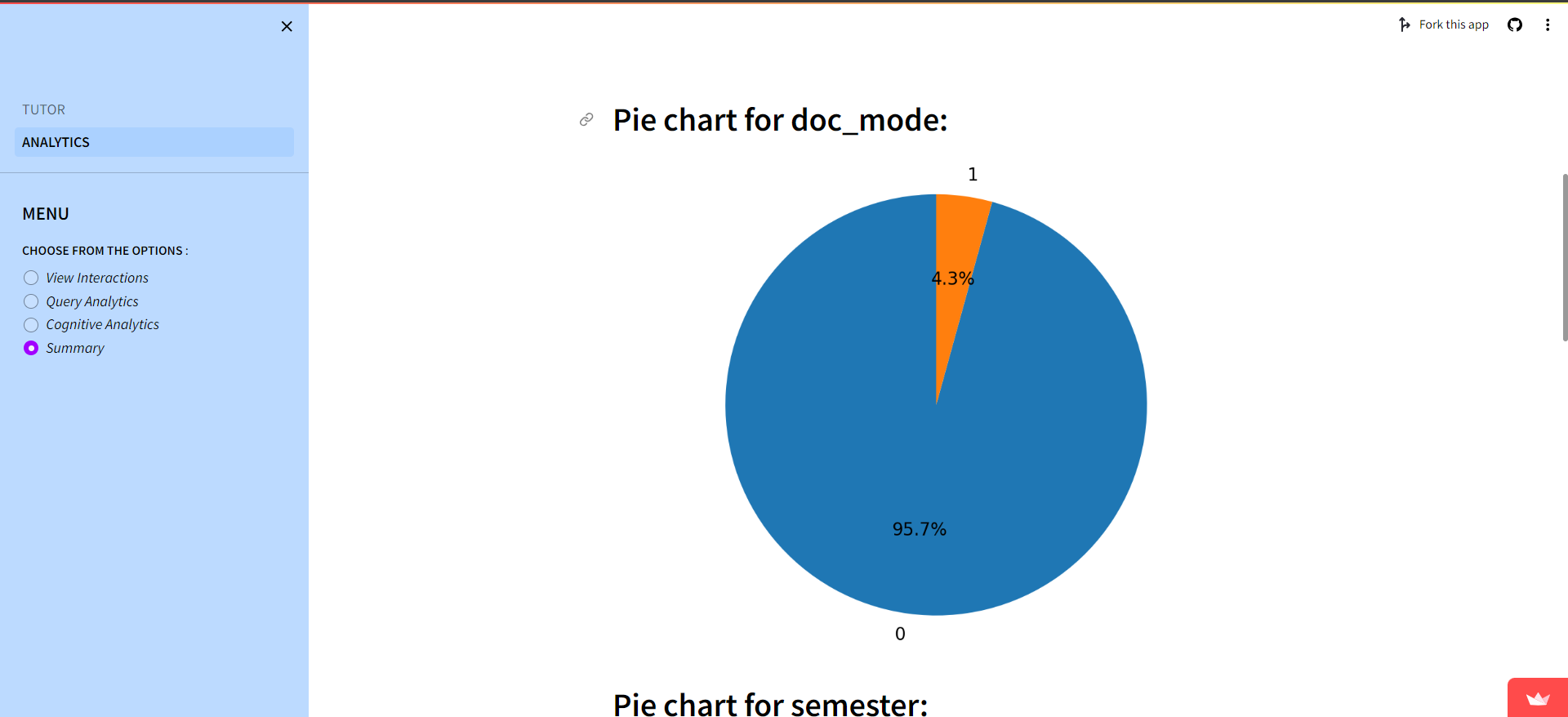
****

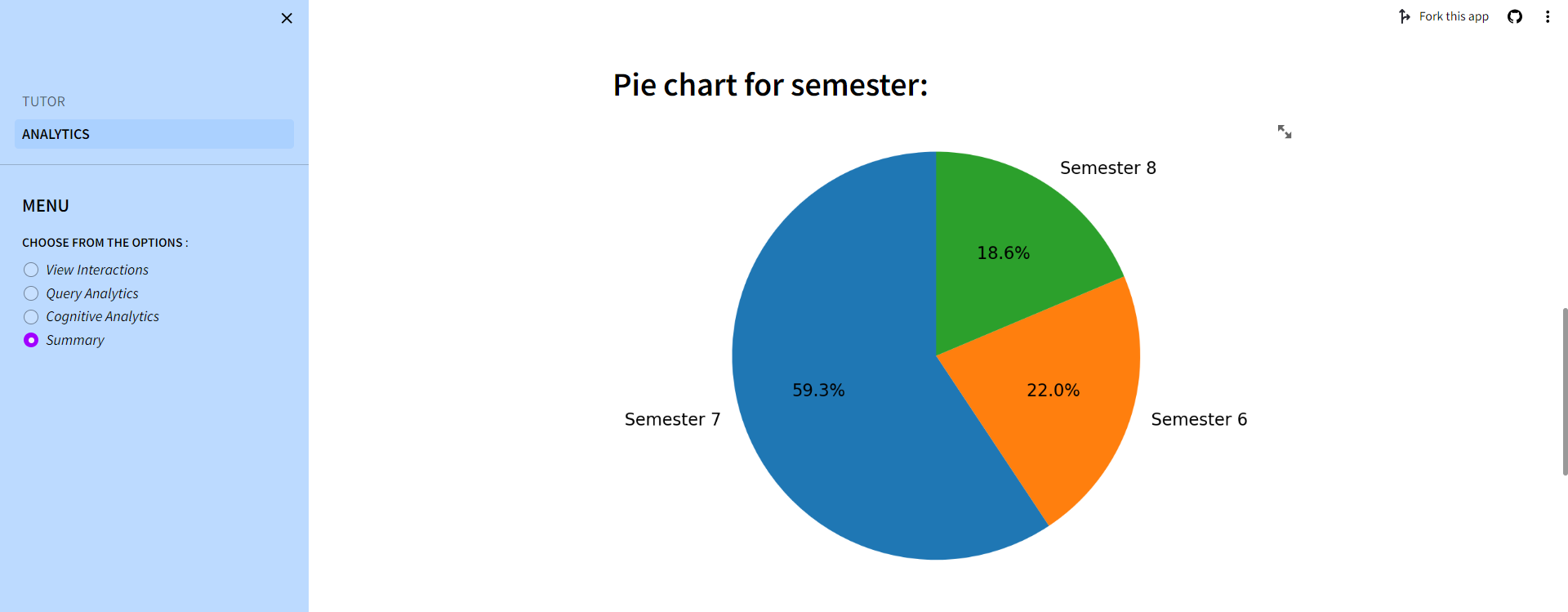
****

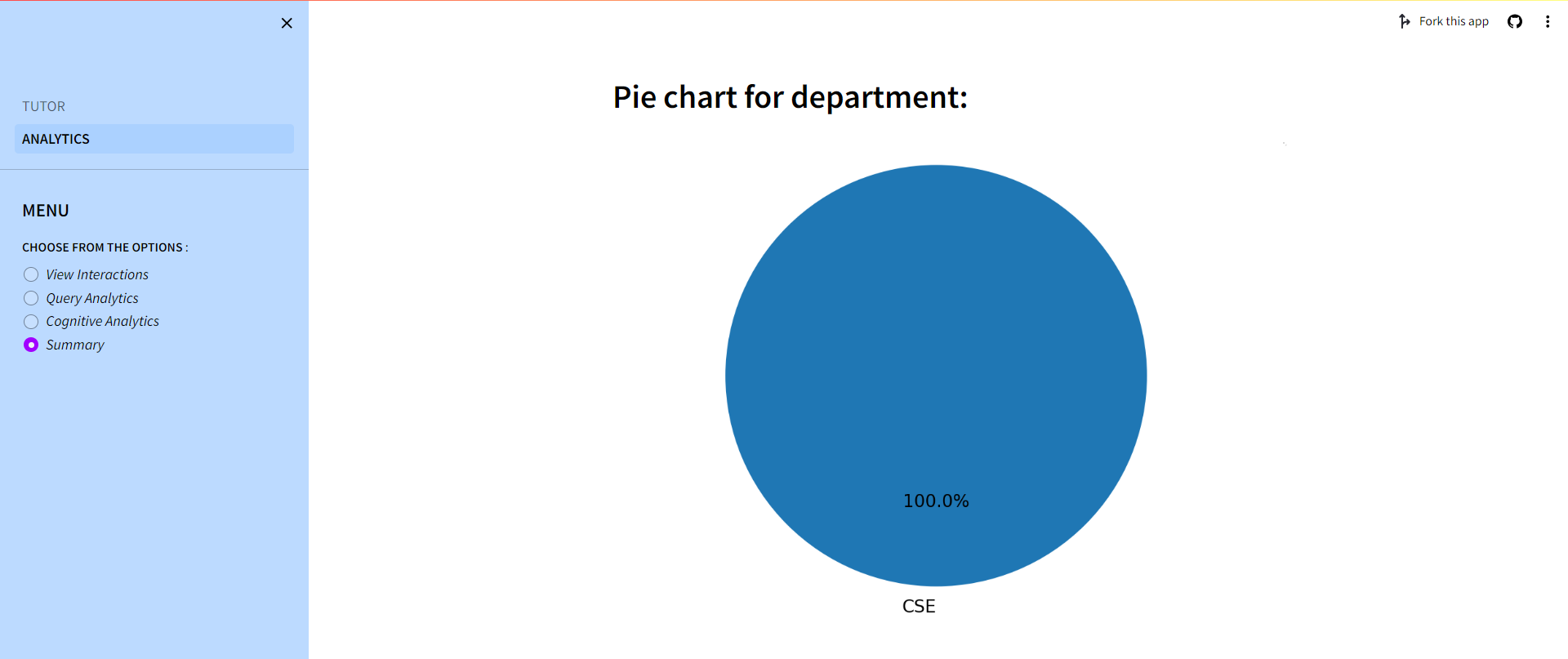
**Cognitive Analytics**

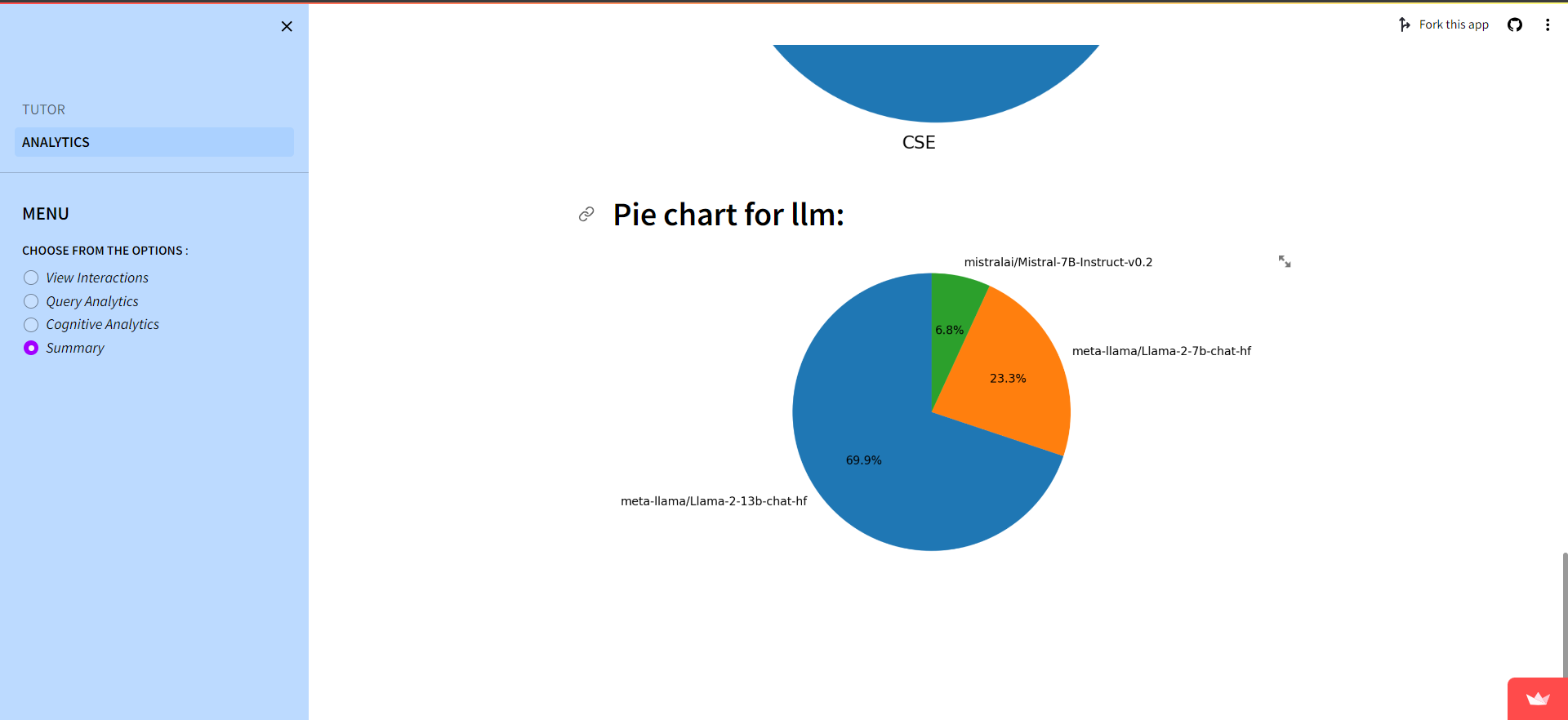
**Overall Summary**

****

****







**APPENDIX II**

**Source Code**

1. **TUTOR.py**

from langchain.document\_loaders import PyPDFDirectoryLoader

from langchain.vectorstores import FAISS

from langchain.embeddings.openai import OpenAIEmbeddings

from langchain.text\_splitter import RecursiveCharacterTextSplitter

from langchain.retrievers import (BM25Retriever,EnsembleRetriever,ContextualCompressionRetriever)

from langchain.retrievers.document\_compressors import (LLMChainExtractor,EmbeddingsFilter)

from langchain.llms import DeepInfra

from dotenv import load\_dotenv

import streamlit as st,time

import os

import re

from datetime import datetime

import toml

from streamlit\_gsheets import GSheetsConnection

import pandas as pd

# load\_dotenv()

# secrets = toml.load(".streamlit/secrets.toml")

OPENAI\_API\_KEY=st.secrets.my\_keys.OPENAI\_API\_KEY

# Accessing the OPENAI\_API\_KEY and DEEPINFRA\_API\_TOKEN variables

DEEPINFRA\_API\_TOKEN=st.secrets.my\_keys.DEEPINFRA\_API\_TOKEN

os.environ['DEEPINFRA\_API\_TOKEN']=DEEPINFRA\_API\_TOKEN

os.environ['OPENAI\_API\_KEY'] =OPENAI\_API\_KEY

# Accessing the DEEPINFRA\_API\_TOKEN variable

compressor\_llm = DeepInfra(model\_id="mistralai/Mistral-7B-Instruct-v0.2")

compressor\_llm.model\_kwargs = {

    "temperature": 0.4,

    "repetition\_penalty": 1.1,

    "max\_new\_tokens": 1000,

    "top\_p": 0.90,

}

embeddings = OpenAIEmbeddings(model="text-embedding-ada-002")

# Initialize the document loader function with caching

@st.cache\_data(show\_spinner=True, persist="disk",)

def doc\_loader(doc\_path):

    # Load documents

    loader = PyPDFDirectoryLoader(doc\_path)

    docs = loader.load()

    # Split documents

    text\_splitter = RecursiveCharacterTextSplitter(

        chunk\_size=1500,

        chunk\_overlap=300,

        length\_function=len,

        is\_separator\_regex=False,

    )

    documents = text\_splitter.split\_documents(docs)

    return documents

# Get the current system time

current\_time = datetime.now()

def process\_documents\_in\_batches(doc\_path, embeddings, batch\_size=500):

    documents = doc\_loader(doc\_path)

    processed = 0

    db = None

    while processed < len(documents):

        try:

            batch\_docs = documents[processed:processed + batch\_size]

            batch\_db = FAISS.from\_documents(batch\_docs, embeddings)

            if db is None:

                db = batch\_db

            else:

                db.merge\_from(batch\_db)

            processed += len(batch\_docs)

            db.save\_local(FAISS\_DB\_PATH)

        except Exception as e:

            print(f"Error encountered: {e}")

            print("Waiting for 5 seconds before retrying...")

            time.sleep(5)

    return db

# Initialize Streamlit's session state

if 'val\_user' not in st.session\_state:

    st.session\_state.val\_user =None

if 'val\_vm' not in st.session\_state:

    st.session\_state.val\_vm = None

# Initialize Streamlit

st.set\_page\_config(layout="wide", page\_title="Cognitut",page\_icon='☂')

st.title(":blue[ COGNITUT : AI Study Buddy]")

st.markdown("""

    <script>

        document.addEventListener('DOMContentLoaded', function() {

            var elementToRemove = document.querySelector('#root > div:nth-child(1) > div.withScreencast > div > div > div > section.main.st-emotion-cache-uf99v8.ea3mdgi3 > div.block-container.st-emotion-cache-z5fcl4.ea3mdgi2 > div > div > div > div.st-emotion-cache-0.e1f1d6gn0 > div');

            if (elementToRemove) {

                elementToRemove.remove();

            }

        });

    </script>

""", unsafe\_allow\_html=True)

with st.spinner(":red[LOADING THE DATABASE....]"):

    conn = st.connection('gsheets',type=GSheetsConnection)

def query\_analysis(query):

    bot\_llm = DeepInfra(model\_id="mistralai/Mistral-7B-Instruct-v0.2")

    bot\_llm.model\_kwargs = {

        "temperature": 0.0,

        "repetition\_penalty": 0.01,

        "max\_new\_tokens": 20,

        "top\_p": 0.9,

    }

    response = bot\_llm(f''' "<s>[INST] Given the input text, evaluates a text string according to four specific criteria and returns a binary array indicating the presence (1) or absence (0) of each factor. The criteria are designed to assess potentially problematic content within the text. The resulting array is formatted as [a, b, c, d], where each letter represents a criterion:

1. Explicit Content (a):

   - Identify any content that includes sexually explicit material, profanity, or adult themes.

   - Flag the text as '1' if any such content is present, otherwise '0'.

2. Discrimination (b):

   - Determine if the text includes language, examples, or implications that are discriminatory towards individuals or groups based on characteristics such as race, gender, sexuality, religion, nationality, or disability.

   - Flag the text as '1' if discrimination is evident, otherwise '0'.

3. Absence of Academic Integrity & Educational Relevance (c):

   - Lack of academic presence in the input(not related to studies)

   - Assess the text for a lack of adherence to academic honesty, such as promoting plagiarism or cheating.

   - Evaluate whether the text is irrelevant to educational objectives, indicating it does not support constructive learning.

   - Flag the text as '1' if it fails in either academic integrity or educational relevance, otherwise '0'.

4. Harmful Intent (d):

   - Recognize any content that could incite harm to oneself or others, promote unsafe practices, or exhibit intentions to cause physical or psychological harm.

   - Flag the text as '1' if harmful intent is identified, otherwise '0'.

Returns:

- list: A binary array [a, b, c, d] that represents the evaluation of the text against the described criteria.

QUERY = Discuss the principles of Newton's laws of motion in an educational context

[/INST]"

"[0,0,0,0]"</s>

"[INST] input : A screenplay that uses explicit sexual language and depicts a group of individuals being demeaned based on their ethnicity. [/INST]"

"[1,1,0,0]"

"[INST] input : A music video script describing violent actions with explicit language aimed at shock value. [/INST]"

"[1,0,0,1]"

"[INST] input : A scientific article falsely claiming that one race is superior to another, using fabricated studies as evidence. [/INST]"

"[0,1,1,0]"

"[INST] input : An instructional blog detailing effective ways to cheat using smart devices during exams. [/INST]"

"[0,0,1,0]"

"[INST] input : A DIY guide on how to mix household chemicals to create explosive reactions, presented as a fun experiment. [/INST]"

"[0,0,0,1]"

"[INST] input : An educational post about the theory of relativity, thoroughly cited and well-explained. [/INST]"

"[0,0,0,0]"

"[INST] input : A dystopian novel scene that graphically portrays sexual assault as a tool of warfare against a particular race. [/INST]"

"[1,1,0,1]"

"[INST] input : A tutorial on creating dangerous fireworks at home, falsely claiming they are safe for school science projects. [/INST]"

"[0,0,1,1]"

"[INST] input : A fake news blog post promoting violent acts against political opponents, containing severe language and fabricated evidence. [/INST]"

"[1,1,1,1]"

"[INST] input : An article on hacking that includes explicit language and detailed instructions on infiltrating secure networks. [/INST]"

"[1,0,1,0]J"

"[INST] input : {query} [/INST]" ''')

    # Use regular expression to parse the binary array from the response

    pattern = r"\[\d,\d,\d,\d\]"

    # Attempt to find the pattern in the response

    match = re.search(pattern, response)

    if match:

        # Convert matched string to an integer list

        arr = list(map(int, match.group(0).strip("[]").split(",")))

    else:

        # Default to [0, 0, 0, 0] if no pattern is found

        arr = [0, 0, 0, 0]

        print(f"THIS IS THE ARRAY : {arr}")

    return arr

def bloom\_analysis(query):

    bot\_llm = DeepInfra(model\_id="mistralai/Mistral-7B-Instruct-v0.2")

    bot\_llm.model\_kwargs = {

        "temperature": 0.0,

        "repetition\_penalty": 0.01,

        "max\_new\_tokens": 20,

        "top\_p": 0.9,

    }

    response = bot\_llm(f''' "<s>[INST] Given the input text, evaluate it according to the following criteria and return the top Bloom's taxonomy classification: Remember (K1), Understand (K2), Apply (K3), Analyze (K4), Evaluate (K5), Create (K6). The classification should be returned as a string, e.g., "K1". Evaluate the text's context and content to accurately assess the classification.

- Remember (K1): Recognize and recall facts, terms, concepts, and answers.

- Understand (K2): Demonstrate understanding of facts and ideas by organizing, comparing, translating, interpreting, giving descriptions, and stating the main ideas.

- Apply (K3): Use a concept in a new situation or unprompted use of an abstraction. Applies what was learned in the classroom into novel situations.

- Analyze (K4): Separates material or concepts into component parts so that its organizational structure may be understood. Distinguishes between facts and inferences.

- Evaluate (K5): Make judgments about the value of ideas or materials.

- Create (K6): Builds a structure or pattern from diverse elements. Put parts together to form a whole, with emphasis on creating a new meaning or structure.

QUERY = Discuss the principles of Newton's laws of motion in an educational context

[/INST]"

"K2"</s>

"[INST] Explain the water cycle in detail.[/INST]"

"K2"

"[INST] Solve quadratic equations using the quadratic formula.[/INST]"

"K2"

"[INST] Compare and contrast photosynthesis and respiration.[/INST]"

"K4"

"[INST] Design a simple machine to lift a load. [/INST]"

"K3"

"[INST] {query} [/INST]" ''')

# Parse the response to get the Bloom's Taxonomy classifications

    return (response[:5])

def validate\_vm\_number(vm\_number):

    # Strip leading and trailing whitespaces

    vm\_number\_stripped = vm\_number.strip()

    regex = r"^[vV][mM]1[3-9][0-9]{3}$"  # Updated regex to match the specified VM number format

    return bool(re.match(regex,  vm\_number\_stripped)), vm\_number\_stripped

def validate\_name(name):

    # Strip leading and trailing whitespaces

    name\_stripped = name.strip()

    # Updated regex to allow dots anywhere in the name

    regex = r"^[a-zA-Z \s]{1,35}$"  # Removed dot from the regex

    return bool(re.match(regex, name\_stripped)), name\_stripped

# Centering the form on the screen

st.markdown(

    """

    <style>

    .center {

        display: flex;

        justify-content: center;

        align-items: center;

        height: 100vh;

    }

    </style>

    """,

    unsafe\_allow\_html=True

)

st.markdown(

    """

    <style>

    #MainMenu{visibility:hidden;}

    footer{visibility :hidden;}

    </style>

    """,

    unsafe\_allow\_html=True

)

# Store in Streamlit's session state

st.session\_state.val\_user = "COGNITUT"

st.session\_state.val\_vm = "None"

if (st.session\_state.val\_user and st.session\_state.val\_vm !=None):

    st.markdown("""

    <style>

        #root > div:nth-child(1) > div.withScreencast > div > div > div > section.main.st-emotion-cache-uf99v8.ea3mdgi3 > div.block-container.st-emotion-cache-z5fcl4.ea3mdgi2 > div > div > div > div.st-emotion-cache-0.e1f1d6gn0 > div {

            display: none !important;

        }

    </style>

""", unsafe\_allow\_html=True)

    # Display initial chat message

    with st.chat\_message("ai", avatar="👨‍🏫"):

        st.write(f"\*\*:blue[Hi , You Can Ask Me Your Academic Doubts !]\*\*")

    # Initialize Streamlit session state

    if "messages" not in st.session\_state:

        st.session\_state.messages = []

    # Display chat message from history on page rerun

    for msg in st.session\_state.messages:

        with st.chat\_message(msg["role"]):

            st.markdown(msg["content"])

    # Sidebar setup

    st.sidebar.title("SETTINGS MENU")

    doc\_mode = st.sidebar.toggle(label="\*\*Syllabus Mode\*\*", value=False, help='Lets you generate answer from your prescribed textbooks')

    if doc\_mode:

        compression\_mode = st.sidebar.toggle(label="Context Compression", help="compress text before passing to LLM, \*may help with inconsistent response\*")

        # NOTE: THE KEY OF THE DICTIONARY IS CASE-SENSITIVE

        all\_subjects = {

            "CSE": {"Semester 7": ["Cloud Computing","Principles Of Management","Cyber Forensics","Cryptography And Network Security","Blockchain Technology"],

                    "Semester 8" : ["Final Project"],

                    "Semester 6":["Artificial Intelligence"]

                    },

            "IT": {

                 'Semester 7': ['Mobile Application Development', 'Cryptography And Network Security', 'Renewable Energy Sources'],

                # 'Semester 6': ['IT SubjectA', 'IT SubjectB', 'IT SubjectC'],

                # Add subjects for other semesters as needed

            },

            "ECE":{"Semester 7":['Wireless Networks','Digital Image Processing','Renewable Energy Sources']},

        #\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                        }

        # Department selection

        department = st.sidebar.selectbox(

            "Choose your department (only few dept support yet)",

            ("CSE", "IT", "ECE", "MECH", "CSBS", "AI & DS", "EEE", "BME"),

            index=None,

        )

        # Semester selection

        semester = st.sidebar.selectbox(

            "Choose your current semester (only 7th sem support yet)",

            ("Semester 1","Semester 2","Semester 3","Semester 4","Semester 5","Semester 6","Semester 7","Semester 8",),index=None)

        # Subject selection based on the selected department and semester

        subject\_options = all\_subjects.get(department, {}).get(semester, [])

        subject = st.sidebar.selectbox("Choose your subject", subject\_options,index=None)

        # Chat model selection

        chat\_model\_name = st.sidebar.selectbox(

            "Choose Chat Model",

            ("meta-llama/Llama-2-7b-chat-hf","mistralai/Mistral-7B-Instruct-v0.2","meta-llama/Llama-2-13b-chat-hf",),index=2, help='Various supported LLMs, \*\*Default : Llama 7B\*\* gives good results')

        if department and semester and subject:

            # Path to pdf documents

            doc\_path = f"Documents/{department}/{semester}/{subject}"

            # Define the path for the FAISS vector store

            FAISS\_DB\_PATH = f"vectorStore/{subject}"

            # Check if the directory exists, and create it if it doesn't

            if not os.path.exists(doc\_path):

                os.makedirs(doc\_path)

            # Check if the directory exists, and create it if it doesn't

            if not os.path.exists(FAISS\_DB\_PATH):

                os.makedirs(FAISS\_DB\_PATH)

            # Check if FAISS\_DB\_PATH is empty

            if not os.listdir(FAISS\_DB\_PATH):

                if os.listdir(doc\_path):

                    with st.spinner("Vector DB is creating..."):

                        st.success("""THIS PROCESS MAY TAKE SOME TIME PLEASE WAIT ....""")

                        if not os.listdir(doc\_path):

                            for file\_name in os.listdir(FAISS\_DB\_PATH):

                                file\_path = os.path.join(FAISS\_DB\_PATH, file\_name)

                                try:

                                    if os.path.isfile(file\_path):

                                        os.remove(file\_path)

                                        st.rerun()

                                except Exception as e:

                                    print(f"Error deleting file {file\_path}: {e}")

                        else:

                            # Create FAISS database in batches with error handling

                            db = process\_documents\_in\_batches(doc\_path, embeddings)

                            st.balloons()

                            st.toast(f"Vector DB created in: {FAISS\_DB\_PATH}", icon="✅")

                            st.rerun()

            else:

                db = FAISS.load\_local(FAISS\_DB\_PATH, embeddings)

                faiss\_retriever = db.as\_retriever(search\_kwargs={"k": 3})

                with st.spinner(f"\*\*Processing ':orange[{doc\_path.split('/')[-1]}]'........\*\*"):

                    # Load documents using the cached function

                    documents = doc\_loader(doc\_path)

                    if not documents:

                        new\_doc\_path = f"Documents/{department}/{semester}/{subject}"

                        if (subject== 'Cryptography And Network Security'):

                            new\_doc\_path = 'Documents\CSE\Semester 7\Cryptography And Network Security'

                        documents=doc\_loader(new\_doc\_path)

                    bm25\_retriever = BM25Retriever.from\_documents(documents)

                    bm25\_retriever.k = 2

                    # initialize the ensemble retriever

                    ensemble\_retriever = EnsembleRetriever(retrievers=[bm25\_retriever, faiss\_retriever], weights=[0.3, 0.7])

                    user\_query = st.chat\_input("Enter your query here ....")

                    if user\_query is not None:

                        # Continue with the code execution

                        docs = ensemble\_retriever.get\_relevant\_documents(user\_query)

                        if compression\_mode:

                            # Compression mode is enabled

                            compressor = LLMChainExtractor.from\_llm(compressor\_llm)

                        else:

                            with st.spinner(f"\*\*:blue[CONTEXT RETRIEVAL : ]\*\* Relavant Information Is Being Extracted From {subject}"):

                                # Compression mode is disabled

                                compressor = EmbeddingsFilter(embeddings=embeddings, similarity\_threshold=0.75)

                        with st.spinner(f":blue[\*\*CONTEXT COMPRESSION :\*\* Reducing Context Size] "):

                            compression\_retriever = ContextualCompressionRetriever(base\_compressor=compressor, base\_retriever=ensemble\_retriever)

                            compressed\_docs = compression\_retriever.get\_relevant\_documents(user\_query)

                        metadata\_info = []

                        for i in range(len(compressed\_docs)):

                            metadata\_info.append("'" + str(compressed\_docs[i].metadata)[23:])

                        def chat\_model(model\_name):

                            llm = DeepInfra(model\_id=model\_name)

                            llm.model\_kwargs = {

                                "temperature": 0.3,

                                "repetition\_penalty": 1.2,

                                "max\_new\_tokens": 1024,

                                "top\_p": 0.9,

                            }

                            return llm

                        llm = chat\_model(chat\_model\_name)

                        with st.spinner(f":blue[\*\*Analysing Input Query ....] "):

                            query\_analysis\_array = query\_analysis(user\_query)

                        context = "\n".join([f"{doc .page\_content}\nMetadata: {doc.metadata}\n"for doc in compressed\_docs])

                        system\_message\_inst = f"""### Role: Specialized Academic Expert Bot

    \*\*Description:\*\*

    Specialized Academic Expert dedicated to in-depth knowledge in the {subject} domain, delivering precise and structured information. Committed to addressing user queries with clarity and aligning responses with academic principles.

    ### Task:

    1. \*\*Subject-specific Contextual Mastery:\*\*

    - Master the art of extracting relevant information in the {subject} context.

    - Base responses meticulously on the provided {subject} context.

    - Use the relevant information to give a comprehensive answer, adhering strictly to academic standards.

    2. \*\*Thorough {subject} Responsiveness:\*\*

    - Demonstrate a commitment to addressing user queries comprehensively in the {subject} field.

    - Utilize {subject}-specific context extensively to provide detailed responses.

    - Stay within the defined context range and avoid making assumptions beyond the provided information.

    3. \*\*Art of {subject} Language:\*\*

    - Employ clear and concise language tailored for the {subject} academic audience.

    - Prioritize clarity through strategic use of headings, markdown, subheadings, paragraphs, and bullet points.

    - Utilize markdown techniques such as '#' for titles, '\*' for highlighting, and '\*\*' for bolding.

    4. \*\*Handling Irrelevance in {subject} Context:\*\*

    - Address situations where the {subject} context lacks relevance by responding with "NOT ENOUGH INFORMATION COULD BE FOUND IN THE {subject} CONTEXT..."

    - Avoid providing information beyond the specified context range.

    5. \*\*Architectural Clarity in {subject} Responses:\*\*

    - Craft responses with a robust structure, specific to the {subject} field.

    - Ensure that responses are accessible and understandable to individuals without prior {subject} knowledge.

    6. \*\*Guiding Academic Principles in {subject} Expertise:\*\*

    - Uphold academic principles in every response, maintaining a high standard of accuracy and reliability in the {subject} domain.

    - Do not hallucinate on information; stay within the confines of the provided context.

    ##\*\* Necessity\*\*:

    - give a comprehensive, structured answer to the Query

    - In the presence of explicit language, comments, vulgar slang, or harmful information, respond with 'I Am a responsible AI. Hence, cannot help you with that' and conclude the response.

    """

                        with st.spinner(":green[\*\*Generating Response....\*\*]"):

                            response = llm(

                                f""" |tags:

                                <s>,<s> = symbolizes generation Instructions

                                [CNTX],[/CNTX] = context for the query

                                [INST],[/INST] = user query|

                                '<->' = logical link/seperation among entities|

                                        <s>{system\_message\_inst}<s><->

                                        [CNTX]{context}[/CNTX]<->

                                        [INST]{user\_query},generate a comprehensive structured response based on context and query[/INST] """)

                        if user\_query is not None:

                            # Save user input and LM output to session state

                            st.session\_state.messages.append({"role": "user", "content": user\_query})

                            st.session\_state.messages.append({"role": "ai", "content": response})

                            database = conn.read(worksheet='Sheet1', usecols=list(range(16)),ttl=0)

                            blooms\_classification = bloom\_analysis(user\_query)

                            # Creating a new entry

                            new\_data\_entry = pd.DataFrame({

                                'user\_name': [st.session\_state.val\_user],

                                'vm\_number': [st.session\_state.val\_vm],

                                'user\_query': [user\_query],

                                'generated\_response': [response],

                                'llm': [chat\_model\_name],

                                'doc\_mode': [doc\_mode],

                                'context\_compression': [compression\_mode],

                                'department': [department],

                                'semester': [semester],

                                'subject': [subject],

                                'date\_time': [current\_time],

                                'explicit': [query\_analysis\_array[0]],

                                'discrimination':[query\_analysis\_array[1]],

                                'academic absence' :[query\_analysis\_array[2]],

                                'harmful intent':[query\_analysis\_array[3]],

                                'blooms classification': [blooms\_classification]

                            })

                            # Concatenating the new entry to the existing DataFrame

                            new\_database = pd.concat([new\_data\_entry, database], ignore\_index=True)

                            conn.update(worksheet='Sheet1',data=new\_database)

                            response = response +"\n"+f''':red[BLOOMS CLASSIFICATION : {blooms\_classification}]'''

                            # Display LM output

                            with st.chat\_message("user", avatar="🟢"):

                                st.markdown(user\_query)

                            with st.chat\_message("ai", avatar="👨‍🏫"):

                                st.markdown(response)

                                st.write(metadata\_info)

                            with st.expander("SHOW CONTEXT"):

                                st.write(context)

        else:

            st.warning(':red[PLEASE FILL DETAILS IN THE LEFT SIDEBAR]')

            st.warning('SIDE BAR IS PRESENT ON TOP LEFT SIDE OF PAGE')

    else:

        chat\_model\_name = st.sidebar.selectbox(

            "Choose Chat Model",("meta-llama/Llama-2-7b-chat-hf","mistralai/Mistral-7B-Instruct-v0.2","meta-llama/Llama-2-13b-chat-hf", ),index=0,help='Various supported LLMs, \*\*Default : Llama 7B\*\* gives good results')

        user\_query = st.chat\_input("Enter your query here ....")

        def chat\_model(model\_name):

            llm = DeepInfra(model\_id=model\_name)

            llm.model\_kwargs = {

                "temperature": 0.7,

                "repetition\_penalty": 1.2,

                "max\_new\_tokens": 1024,

                "top\_p": 0.85,

            }

            return llm

        llm = chat\_model(chat\_model\_name)

        if user\_query is not None:

            with st.spinner(f"\*\*'DOCUMENT MODE =  :red[OFF] | :blue[Generating Response]......'\*\*"):

                system\_message\_inst = """# Role: Academic Expert

    \*\*Description:\*\*

    Dedicated Comprehensive Academic Expert Bot with a profound understanding of various subjects, committed to delivering high-quality, detailed responses. This bot excels in providing precise and structured information, aligning answers with the highest academic standards.

    ## Task:

    1. \*\*Subject-specific Contextual Mastery:\*\*

    - Master the art of extracting relevant information in any academic context.

    - Base responses meticulously on the provided subject context.

    - Utilize relevant information to deliver comprehensive answers adhering to academic standards.

    2. \*\*Thorough Responsiveness:\*\*

    - Demonstrate a commitment to addressing user queries comprehensively across diverse academic fields.

    - Utilize subject-specific context extensively to provide detailed, well-informed responses.

    3. \*\*Art of Academic Language:\*\*

    - Employ clear and concise language tailored for a diverse academic audience.

    - Prioritize clarity through strategic use of headings, markdown, subheadings, paragraphs, and bullet points.

    - Use markdown techniques for titles (#), highlighting (\*), and bolding (\*\*).

    4. \*\*Handling Irrelevance in Academic Context:\*\*

    - Address situations where the academic context lacks relevance by responding with "NOT ENOUGH INFORMATION COULD BE FOUND IN THE CONTEXT..."

    5. \*\*Architectural Clarity in Responses:\*\*

    - Craft responses with a robust structure applicable to various academic fields.

    - Ensure that responses are accessible and understandable to individuals without prior knowledge in a specific subject.

    6. \*\*Guiding Academic Principles:\*\*

    - Uphold academic principles in every response, maintaining a high standard of accuracy, reliability, and depth across disciplines.

    ## Markdown Techniques Explanation:

    - Use '#' for titles, e.g., #Title#

    - For highlighting and bolding, use '\*', e.g., \*Highlighted\* or \*\*Bolded\*\*.

    ##\*\* Necessity\*\*:

    - give a fulfilling , comprehensive, structure answer to the query

    - In the presence of explicit language, comments, vulgar slang, or harmful information, respond with 'I Am a responsible AI. Hence, cannot help you with that' and conclude the response.

    """

                with st.spinner(":green[\*\*Generating Response....\*\*]"):

                    response = llm(

                            f"""|tags:

                            <s>,<s> = symbolizes generation Instructions

                            [INST],[/INST] = user query|

                            '<->' = logical link/seperation among entities|

                            <s>{system\_message\_inst}<s><->

                            [INST]{user\_query} , note# generate a comprehensive structured response based on user query[/INST] """)

                with st.spinner(f":blue[\*\*Analysing Input Query ....] "):

                    query\_analysis\_array = query\_analysis(user\_query)

                if user\_query is not None:

                    # Save user input and LM output to session state

                    st.session\_state.messages.append({"role": "user", "content": user\_query})

                    st.session\_state.messages.append({"role": "ai", "content": response})

                    compression\_mode=None

                    department=None

                    subject=None

                    semester=None

                    doc\_mode=False

                    # Creating a new entry

                    database = conn.read(worksheet='Sheet1', usecols=list(range(16)),ttl=0)

                    blooms\_classification = bloom\_analysis(user\_query)

                    # Creating a new entry

                    new\_data\_entry = pd.DataFrame({

                        'user\_name': [st.session\_state.val\_user],

                        'vm\_number': [st.session\_state.val\_vm],

                        'user\_query': [user\_query],

                        'generated\_response': [response],

                        'llm': [chat\_model\_name],

                        'doc\_mode': [doc\_mode],

                        'context\_compression': [compression\_mode],

                        'department': [department],

                        'semester': [semester],

                        'subject': [subject],

                        'date\_time': [current\_time],

                        'explicit': [query\_analysis\_array[0]],

                        'discrimination':[query\_analysis\_array[1]],

                        'academic absence' :[query\_analysis\_array[2]],

                        'harmful intent':[query\_analysis\_array[3]],

                        'blooms classification': [blooms\_classification]

                    })

                    # Concatenating the new entry to the existing DataFrame

                    new\_database = pd.concat([new\_data\_entry, database], ignore\_index=True)

                    conn.update(worksheet='Sheet1',data=new\_database)

                    response = response + "\n"+f''' :red[BLOOMS CLASSIFICATION : {blooms\_classification}]'''

                    # Display LM output

                    with st.chat\_message("user", avatar="🟢"):

                        st.markdown(user\_query)

                    with st.chat\_message("ai", avatar="👨‍🏫"):

                        st.markdown(response)

                    st.rerun()

1. **1\_Analytics.py**

import streamlit as st

import pandas as pd

import numpy as np

from streamlit\_gsheets import GSheetsConnection

import matplotlib.pyplot as plt

st.set\_page\_config(page\_title="ANALYTICS", page\_icon="📈")

st.markdown("# :red[ INTERACTION ANALYTICS 📈]")

st.sidebar.header("MENU")

analytics = st.sidebar.radio("\*\*CHOOSE FROM THE OPTIONS\*\* :", ["\*View Interactions\*", "\*Query Analytics\*", "\*Cognitive Analytics\*", "\*Summary\*"], index=0)

conn = st.connection('gsheets', type=GSheetsConnection)

database = conn.read(worksheet='Sheet1', usecols=list(range(16)), ttl=0)

# Convert the database to a DataFrame

df = pd.DataFrame(database)

if analytics == "\*View Interactions\*":

    st.write(df)

elif analytics == "\*Query Analytics\*":

    # Convert 'date\_time' to datetime format

    df['date\_time'] = pd.to\_datetime(df['date\_time'], errors='coerce')

    # Drop rows where 'date\_time' is NaT

    df.dropna(subset=['date\_time'], inplace=True)

    # Initialize an empty DataFrame to store daily counts

    daily\_counts = pd.DataFrame(index=df['date\_time'].dt.date.unique()).sort\_index()

    # Loop through each feature to count the occurrences of 1 per day

    for col in ["explicit", "discrimination", "academic absence", "harmful intent"]:

        daily\_counts[col] = df.groupby(df['date\_time'].dt.date)[col].apply(lambda x: np.sum(x == 1))

    # Plot the counts against time using a line chart

    st.line\_chart(daily\_counts)

elif analytics == "\*Cognitive Analytics\*":

    # Clean and count occurrences of specified values in "blooms classification"

    df["blooms classification"] = df["blooms classification"].str.upper().str.replace(r'[^K1K2K3K4K5K6]', '')

    # Count occurrences of specified values

    blooms\_counts = df["blooms classification"].value\_counts().reindex(['K1', 'K2', 'K3', 'K4', 'K5', 'K6']).fillna(0)

    # Plot the counts on a bar chart

    st.bar\_chart(blooms\_counts)

elif analytics == "\*Summary\*":

    # Total Interactions

    total\_interactions = len(df)

    st.write(f"## :green[Total Interactions: {total\_interactions}]")

    # Pie chart for k1 to k6

    blooms\_counts = df["blooms classification"].str.upper().str.replace(r'[^K1K2K3K4K5K6]', '')

    blooms\_counts = blooms\_counts.value\_counts().reindex(['K1', 'K2', 'K3', 'K4', 'K5', 'K6']).fillna(0)

    fig1, ax1 = plt.subplots()

    ax1.pie(blooms\_counts, labels=blooms\_counts.index, autopct='%1.1f%%', startangle=90)

    ax1.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

    st.write("## \*\*Pie chart for k1 to k6 :\*\*")

    st.pyplot(fig1)

    # Pie chart for doc\_mode

    df['doc\_mode'] = df['doc\_mode'].apply(lambda x: 1 if x == 1 else 0)

    doc\_mode\_counts = df['doc\_mode'].value\_counts()

    fig3, ax3 = plt.subplots()

    ax3.pie(doc\_mode\_counts, labels=doc\_mode\_counts.index, autopct='%1.1f%%', startangle=90)

    ax3.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

    st.write("## \*\*Pie chart for doc\_mode:\*\*")

    st.pyplot(fig3)

    # Pie chart for semester

    semester\_counts = df['semester'].value\_counts()

    fig4, ax4 = plt.subplots()

    ax4.pie(semester\_counts, labels=semester\_counts.index, autopct='%1.1f%%', startangle=90)

    ax4.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

    st.write("## \*\*Pie chart for semester:\*\*")

    st.pyplot(fig4)

    # Pie chart for department

    department\_counts = df['department'].value\_counts()

    fig5, ax5 = plt.subplots()

    ax5.pie(department\_counts, labels=department\_counts.index, autopct='%1.1f%%', startangle=90)

    ax5.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

    st.write("## \*\*Pie chart for department:\*\*")

    st.pyplot(fig5)

    # Pie chart for llm

    llm\_counts = df['llm'].value\_counts()

    fig6, ax6 = plt.subplots()

    ax6.pie(llm\_counts, labels=llm\_counts.index, autopct='%1.1f%%', startangle=90)

    ax6.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.

    st.write("## \*\*Pie chart for llm:\*\*")

    st.pyplot(fig6)

**REFERENCES**

**[1].** Khan, S., Lin, J., He, L., Wu, Y., & Yan, R. (2022). Retrieval-Augmented Generation for Conversational AI: A Survey. *ACM Computing Surveys (CSUR)*\*, 55(3), 1-49.

**[2].** Xing, M., Wu, M., Li, M., & Wang, M. (2023). Personalized Conversational Tutoring Systems: A Review. *IEEE Transactions on Learning Technologies*, 16(2), 155-172.

**[3].** Dabaghi, S., Yin, M., & Davis, J. D. (2022). Open-Source Language Models for Education: A Survey. *Computer Science Review*, 48, 100476.

**[4].** Liu, J., Fan, Y., & Jiang, J. (2021). Building a Cognitive Analytics System for Educational Technology. *IEEE Transactions on Learning Technologies*, 14(3), 373- 385.

**[5].** Yu, M., Huang, N., Wang, M., & Zhu, X. (2023). Future Directions in AI-Driven Tutoring Systems. *International Journal of Artificial Intelligence in Education*, 33(1), 3-29.