**Project Description – Tapu Rares Gabriel 15222616**

The goal of this project was to explore the capabilities of Retrieval-Augmented Generation (RAG) techniques to create a scalable document retrieval system without relying on extensive computational resources. This project also included building and optimizing an audio processing pipeline for a voice agent, integrating components such as speech-to-text (STT), text-to-speech (TTS), and language model processing. The research question guiding this project was: **Can we retrieve documents at scale without using extensive compute power?**

The initial phase of the project involved creating a basic voice agent. Using a comprehensive tutorial, I implemented the back-end of the voice agent, which included capturing audio, sending it to Microsoft Cognitive Services (MCS) for transcription, processing the transcript with OpenAI, and playing the generated response. This step provided a foundational understanding of the entire audio data pipeline, including capturing, processing, and generating responses. However, the initial implementation revealed significant latency issues, with a total processing time exceeding 25 seconds.

To address these latency issues, I began isolating and timestamping each part of the pipeline to monitor and optimize each component individually. By separating the STT and TTS processes into different files and rewriting the STT code, I achieved a substantial reduction in latency, bringing it down to around 5 seconds. This optimization emphasized the importance of asynchronous processing and thorough error handling, which were crucial for reducing latency and improving system reliability.

The next phase involved exploring Langchain, a framework for building language models. I delved into its core modules: langchain, langchain\_community, and langchain\_core. Using Langsmith, the LLM-devops platform, I traced API calls and implemented a basic chat application to understand concepts like prompt templates, memory, and chains. This exploration provided a comprehensive understanding of Langchain’s architecture and functionalities, setting the stage for implementing Retrieval-Augmented Generation (RAG) techniques.

A significant portion of the project focused on implementing RAG techniques to enhance the performance of language models by retrieving relevant documents before generating responses. The RAG process involved several stages: query translation and construction, indexing, retrieving with k-nearest neighbors (k-NN), and generating responses. Query translation and construction helped the language model understand the task effectively. Indexing involved splitting the data into smaller chunks, embedding them, and creating a vector database for efficient searching and retrieval. Using the k-NN algorithm, the system searched for the nearest neighbors in the vector embedding space, ensuring relevant document retrieval. Finally, the model generated responses based on the retrieved documents.

Implementing the entire RAG pipeline provided valuable insights into each component’s role in document retrieval. I learned the importance of embedding, indexing, and k-NN in efficient document retrieval. Additionally, I explored the impact of chunk size and overlapping on retrieval performance, optimizing these parameters for better results.

These implementations demonstrated that documents could be retrieved at scale without extensive computational resources.

The final phase of the project involved integrating the conversational RAG agent into the existing pipeline. This required redesigning the RAG to be asynchronous and ensuring seamless integration with the asynchronous pipeline. One of the main challenges was handling responses involving multiple API calls and extracting the necessary information. Successfully integrating the RAG agent enabled scalable document retrieval with minimal computational resources and enhanced the system’s efficiency.

In conclusion, this project successfully demonstrated that documents could be retrieved at scale without extensive compute power using RAG techniques. By implementing and optimizing various components of the RAG pipeline, I gained critical insights into embedding, indexing, and retrieving documents using k-nearest neighbors. Additionally, optimizing the audio processing pipeline for a voice agent provided valuable lessons in asynchronous processing and real-time communication. This project not only answered the research question affirmatively but also provided a deep understanding of the techniques required for scalable document retrieval and efficient audio processing systems.