**Process Book – Tapu Rares Gabriel 15222616**

**Day 1: 20.05.2024**

**What I’ve worked on:**

* Created the first instance of the voice agent.

**What problems I encountered:**

* High latency in the audio data processing pipeline, over 25 seconds.
* Inefficient audio data processing, despite the OpenAI call taking only one second.

**What I learned:**

* Understood the entire audio data pipeline: capturing audio, processing with Microsoft Cognitive Services (MCS), generating a transcript, interacting with OpenAI for a response, and playing the audio response.

**Which resources did I use:**

* Followed a one-hour tutorial to create the back-end of the voice agent.
* Conducted additional research on improving the app.

<https://www.youtube.com/watch?v=dNVsFbo9hF8>

**Additional Notes:**

* First GitHub commit made.
* Plan to isolate and timestamp each part of the pipeline for better monitoring and split the audio processing file for easier testing.

**Day 2: 21.05.2024**

**What I’ve worked on:**

* Implemented timestamps to monitor processing times in the pipeline.
* Separated STT (Speech-to-Text) and TTS (Text-to-Speech) processing into different files.

**What problems I encountered:**

* Identified STT as the biggest bottleneck with an 18-second processing time.
* Sequential processing in STT instead of asynchronous.
* Limited error handling in the tutorial code.

**What I learned:**

* Rewriting the STT code improved processing time from 18 seconds to 1-2 seconds.
* Understanding the importance of asynchronous processing and error handling.

**Which resources did I use:**

* Continued using the tutorial for structure and additional research on optimizing STT.

https://learn.microsoft.com/en-us/rest/api/cognitiveservices/ (MCS API)

**Additional Notes:**

* Latency reduced to around 5 seconds.
* Second GitHub commit made.
* Future improvements planned: chunk-based processing and Websockets for real-time communication.

**Day 3: 22.05.2024**

**What I’ve worked on:**

* Reviewed and tested the code, recording detailed timestamps for each pipeline stage.

**What problems I encountered:**

* Need to further optimize each part of the pipeline for reduced latency.

**What I learned:**

* Current processing times: 1.04s for transcript processing, 1.51s for OpenAI response, and 0.56s for speaking the response.

**Which resources did I use:**

* Continued following the tutorial and additional documentation on asynchronous processing.

<https://levelup.gitconnected.com/async-processing-in-python-make-data-pipelines-scream-ceea74a537ad>

<https://github.com/theboxahaan/pipelines>

**Additional Notes:**

* Planned improvements for STT, including switching to asynchronous processing and implementing Websockets.

**Day 4: 23.05.2024**

**What I’ve worked on:**

* Made asynchronous improvements in both STT and TTS pipelines.

**What problems I encountered:**

* Ensuring both files process the code fully asynchronously. Didn’t manage to implement Websockets

**What I learned:**

* Reduced STT processing time from 1.04s to 0.87s and TTS from 0.56s to 0.47s.

**Which resources did I use:**

* Documentation and tutorials on asynchronous programming and Websockets.

<https://websockets.readthedocs.io/en/stable/>

**Additional Notes:**

* Planning to optimize the OpenAI call next.

**Day 5: 24.05.2024**

**What I’ve worked on:**

* Explored Langchain, focusing on its core modules: langchain, langchain\_community, and langchain\_core.

**What problems I encountered:**

* None encountered; focused on learning new techniques.

**What I learned:**

* Understood concepts like prompt templates, memory, chains, and started using Langsmith for tracing API calls.

**Which resources did I use:**

* Langchain documentation and tutorials.

<https://python.langchain.com/v0.2/docs/introduction/>

<https://docs.smith.langchain.com>

**Additional Notes:**

* Implemented a basic chat app to test Langchain concepts.

**Day 6: 25.05.2024**

**What I’ve worked on:**

* Learned about Retrieval-Augmented Generation (RAG) techniques.

**What problems I encountered:**

* None encountered; focused on understanding the RAG pipeline.

**What I learned:**

* Followed a tutorial to understand query translation, indexing, embedding, retrieval, and generating responses.

**Which resources did I use:**

* Tutorial by a Langchain engineer.

<https://www.youtube.com/watch?v=sVcwVQRHIc8&t=3872s>

**Additional Notes:**

* Understood the complete RAG pipeline from query to response generation.

**Day 7: 26.05.2024**

**What I’ve worked on:**

* Started implementing a basic conversational RAG using Langchain documentation.

**What problems I encountered:**

* Challenges in splitting documents and setting up the retriever and chat agent.

**What I learned:**

* Implemented document loading, chunk splitting, retriever creation, chain design, and memory addition.

**Which resources did I use:**

* Langchain documentation and Langraph framework.

<https://python.langchain.com/v0.1/docs/langgraph/>

**Additional Notes:**

* Initialized a chat agent and monitored traces with Langsmith.

**Day 8: 27.05.2024**

**What I’ve worked on:**

* Integrated the conversational RAG agent into the current pipeline.

**What problems I encountered:**

* Redesigning the conversational RAG to be async and integrate seamlessly with the existing async pipeline.
* Handling the response content involving multiple API calls and extracting the necessary information.

**What I learned:**

* Overcame difficulties in making the RAG agent async and extracting content from nested dictionaries.

**Which resources did I use:**

* Langchain and Langraph documentation and previous project code.

**Additional Notes:**

* Successfully integrated the RAG agent, preparing to further optimize and expand the functionality.