Customer Support Chatbot

The main Intention behind this Project is to develop a chatbot that supports customer tickets. The system uses retrieval augmented generation (RAG) to fetch solutions.

The chatbot is deployed in a fictitious e-commerce firm called Voltathena, which sells electrical, electronics, and other tech products online.

The Project is done using the following steps:

1. ***Create synthetic data for the company:*** Synthetic data is generated in pdf format to include it in the database. The data involves ***a short overview of the company, employees of the customer relationship departmen***t, ***manuals and FAQs of the product sold on the platform, Refund Policy, and Product Catalog.***

2. ***Indexing***: In indexing, we need to perform the following steps.
    i. 1. Document Loading
    ii. 2. Document splitting
    iii. 3. Document Embedding
    iv. 4. Document Storing

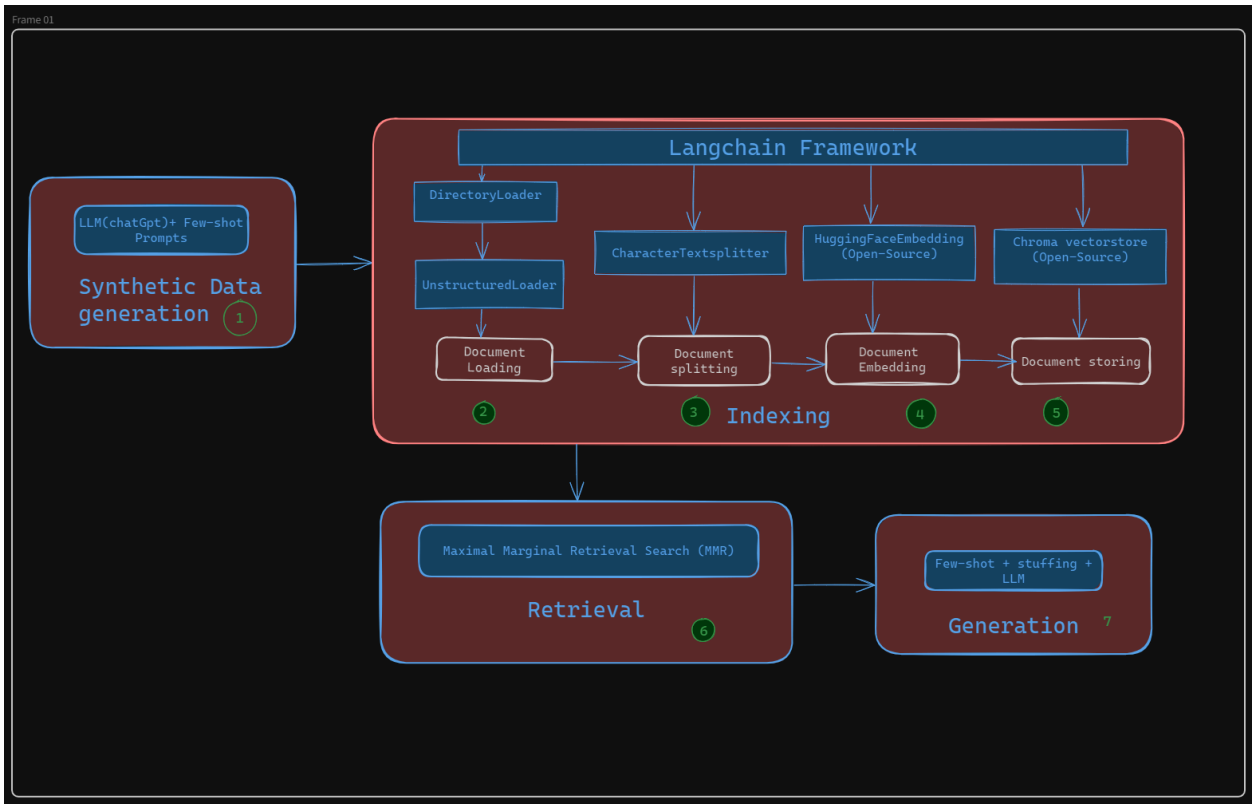
    Document loading was done using - Directory Loader and `UnstructuredLoader` from Langchain. The `UnstructuredLoader` has the advantage of ***loading multiple file formats and extracting text from them.*** The `characterTextSplitter` with `separator` as period, is used to split the documents into chunks of size 500 and overlapping of 20. This ensures the split is done at period ***capturing the whole sentence in the chun***k, which in turn helps in preserving the semantic meaning in the chunk.

3. ***HuggingFace Embedding*** is used to embed the data. It is an ***open-source embedding*** model that provides room for experimentation and tinkering, especially when we start out from scratch. This is economical because we ***don't need to worry about the wastage of tokens*** which is the case when working with paid proprietary embedding models like `OpenAI Embedding`. Then Embeddings are stored in the `chroma` Vector database which is also open-source. We can embed all the documents at once. This approach saves time and ensures consistency across embeddings.

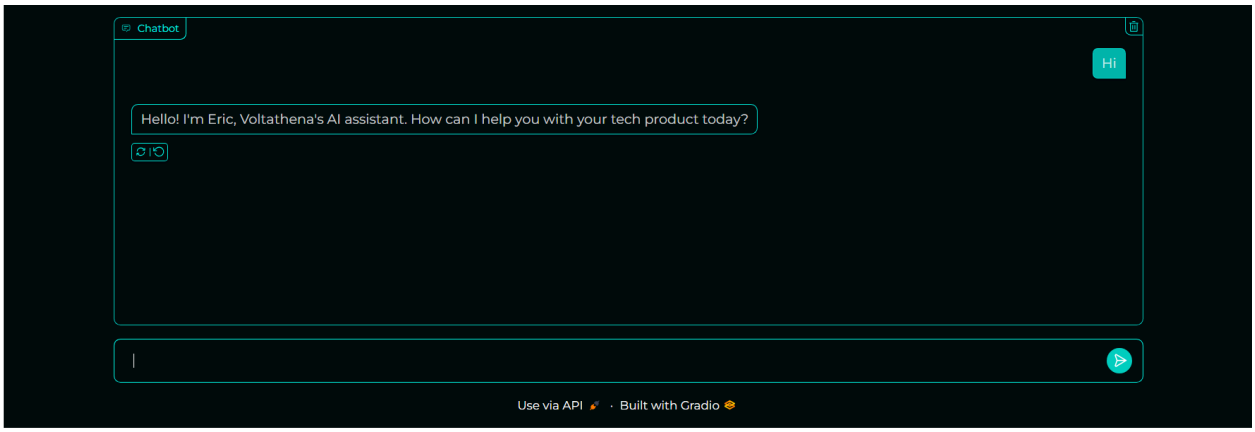
4. ***Retrieval***: The next stage is retrieval. Maximal Marginal Relevance search is used for retrieval(MMR). MMR balances ***relevance and diversity.*** This ensures that retrieved documents are not only relevant but also diverse enough to ***avoid redundancy***. This ***eliminates the need to deal with duplicate documents***, which we might have encountered if we had used a simple `similarity search`. Through hyperparameter `lambda` we can control the relevance or diversity of the document.

5. ***Generation:*** In the generation step, we stuff the document into the user query as the context and feed it to the LLM. The Model used is ***Zephyr-7b-beta***, which is also an ***open-source*** LLM Model available for free.

6. The ***FewshotPromptTemplate*** in Hugging Face Provides an easy way to include Few-shot Prompts in the LLM. This enables us to Learn from the examples and respond efficiently.



7. Finally, A proto-type was deployed in Gradio for the demo.



Further Enhancements and Improvements

1. **Prompt Engineering:** The prompts could be further improved by explicitly instructing the model not to guess, which helps in reducing hallucinations.

2. **Confidence Scoring**: A simulated confidence score mechanism can be implemented to assess whether the generated response meets a predefined threshold.

3. **Post-processing**: After generating a response, to enhance factual accuracy, a mechanism could be implemented to validate the logic).

4. **User Feedback Loop**: Users flagging the system to flag incorrect answers, can allow the chatbot to gather data for future improvements.