**Index**

1. Technology stack
2. Components
3. Working WRT Flow chart
4. Deployment
5. Scalability, Performance and Security
6. **Technology stack**

a. Coding language : Python

b. Environment : Google colab with T4 GPU for faster execution

1. **Components**

a. Framework : Langchain

b. Transformer : Hugging Face

c. Model : meta-llama/Llama-2-7b-chat-hf. Llama 2 is trained on 2 trilion tokens and therefore is a stable model and has enough context for our use case.

d. Vector DB : Facebook AI Similarity Search (FAISS)

1. **Working WRT Flow chart**

a. Document processing flow :

* + 1. Document is loaded and broken into chunks by using Langchain framework's text splitter with specified chunk size and sufficient overlap so as not to miss the context while chunking.
    2. After chunking, the chunks are then embedded using Hugging face embeddings which is also part of langchain framework. With embedding process the tokens in the chunks are converted into mathematical repesentation which is easier to store and query. For storing we need a special type of Database called vector database.
    3. Embeddings are stored in FAISS database.

b. User query answering flow :

* + 1. User's query along with his chained queries are together sent to the Llama model via Chain models provided by Langchain.
    2. A standalone question is generated based on current query, hugging face transformer and the context provided.
    3. For the generated query, vector databased is queried for matching results and alongwith Langchain QA chain it is then sent to Llama 2 model for contextual and grammatical corrections based on the configurations like temparature settings, repetition penalty etc.

1. **Deployment**

a. Machine requirements :

1. For deployment we need GPU since the transformers and LLM model needs large computing power.
2. I have used Google colab with T4 GPU for faster execution of the queries.

b. Code deployment :

i. The code saved in 'Langchain QA Bot with Llama 2' file is a Google colab file which can be reused in Google colab environment as is.

1. **Scalability, Performance and Security** 
   1. Scalability :
      1. We can scalability of models, we can use distributed computing.
      2. We can also separate the current application hosting and new model updates on separate machines and switch the application to newer model machine when it is ready.
      3. Distributed computing on different machines with powerful machines using CUDA like architecture would be performant for large models.
   2. Performance :
      1. We have used T4- GPU with CUDA and loaded the model on GPU for faster execution.
      2. Normal CPU would take much time to process the model as compared to GPU.
      3. Sometimes the CPU crashed while loading the model.
   3. Security :
      1. For security, we can use Azure key vault for storing the API keys, tokens and accessing them via keys.
      2. We can also use Environment variables to store the API keys and tokens, but it wont be as secure as using any Cloud based key-value storing service.