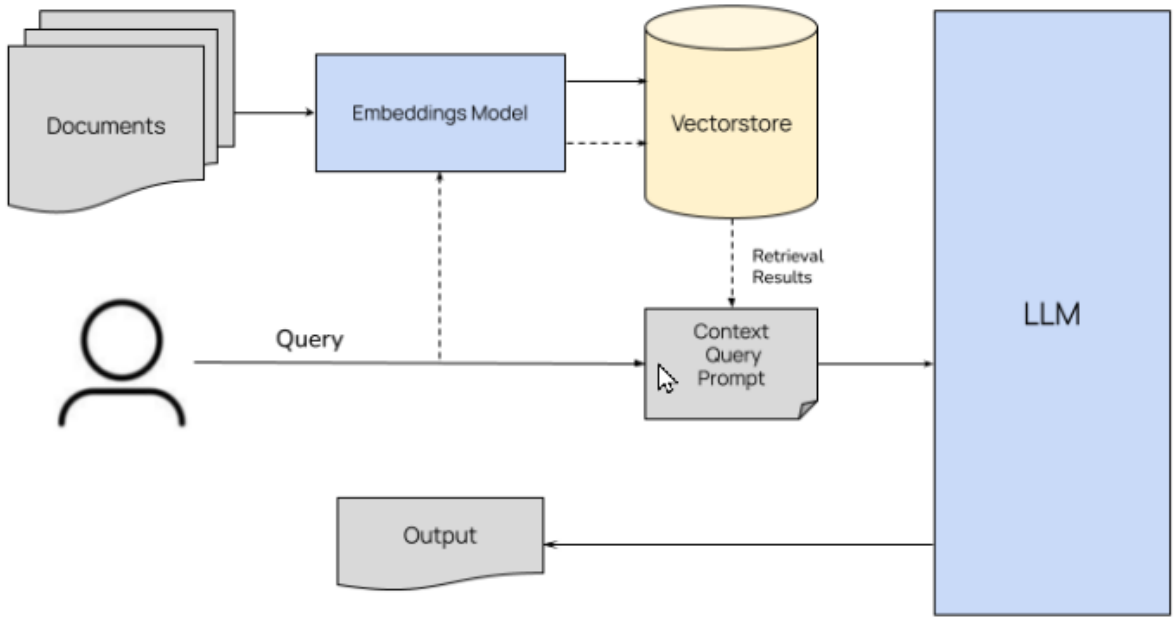
Project Title: RAG Document Q&A Chatbot Development

1. Introduction:

The purpose of this document is to outline the requirements for developing a RAG (Retrieval-Augmented Generation) Q&A chatbot with the capability to load and process data from various file formats, store embeddings in a local vector database, and provide a frontend UI for user interaction.

2. Project Overview:

The project involves the development of a **RAG Q&A chatbot** leveraging state-of-the-art techniques in natural language processing (NLP), machine learning and Generative AI. The Q&A chatbot will be capable of understanding user queries, retrieving relevant information from pre-loaded data files, and presenting the information in a user-friendly manner through a **Streamlit-based frontend UI.**



3. Functional Requirements:

<u>Data Loading (one time):</u>

The chatbot should be able to load data from various file formats including PDF, WORD, JSON, CSV, Key-value pair, EXCEL, etc.
Upon loading, the data should be processed to extract relevant text information for further processing.

<u>Embedding Generation:</u>

Utilize a pre-trained language model (e.g., BERT) to generate embeddings for the text data.
Store the generated embeddings in a local vector database for efficient retrieval.

Query Processing:

Accept user queries in natural language.
Process the queries to understand user intent and retrieve relevant embeddings from the vector database.

Response Generation:

Generate responses to user queries by leveraging the RAG framework.
Use the retrieved embeddings to augment the response generation process for improved relevance.

Frontend UI:

Develop a Streamlit-based frontend UI for the chatbot.
The UI should allow users to input queries and visualize the retrieved information in a structured format.

4. Non-Functional Requirements:

Performance:

The chatbot should respond to user queries within a reasonable timeframe.
Ensure efficient storage and retrieval of embeddings to minimize response latency.

Scalability:

Design the system to handle large volumes of data and user queries.
Implement mechanisms for horizontal scaling if needed to accommodate future growth.

Usability:

The frontend UI should be intuitive and easy to use for non-technical users.
Provide clear instructions and guidance on how to interact with the chatbot.

5. Technical Stack:

Language: Python

NLP Framework: llamaindex

Vector Database: FAISS

Frontend Framework: Streamlit

6. Milestones:

Show full working chatbot on streamlit

7. Deliverables:

Source code with **full documentation**