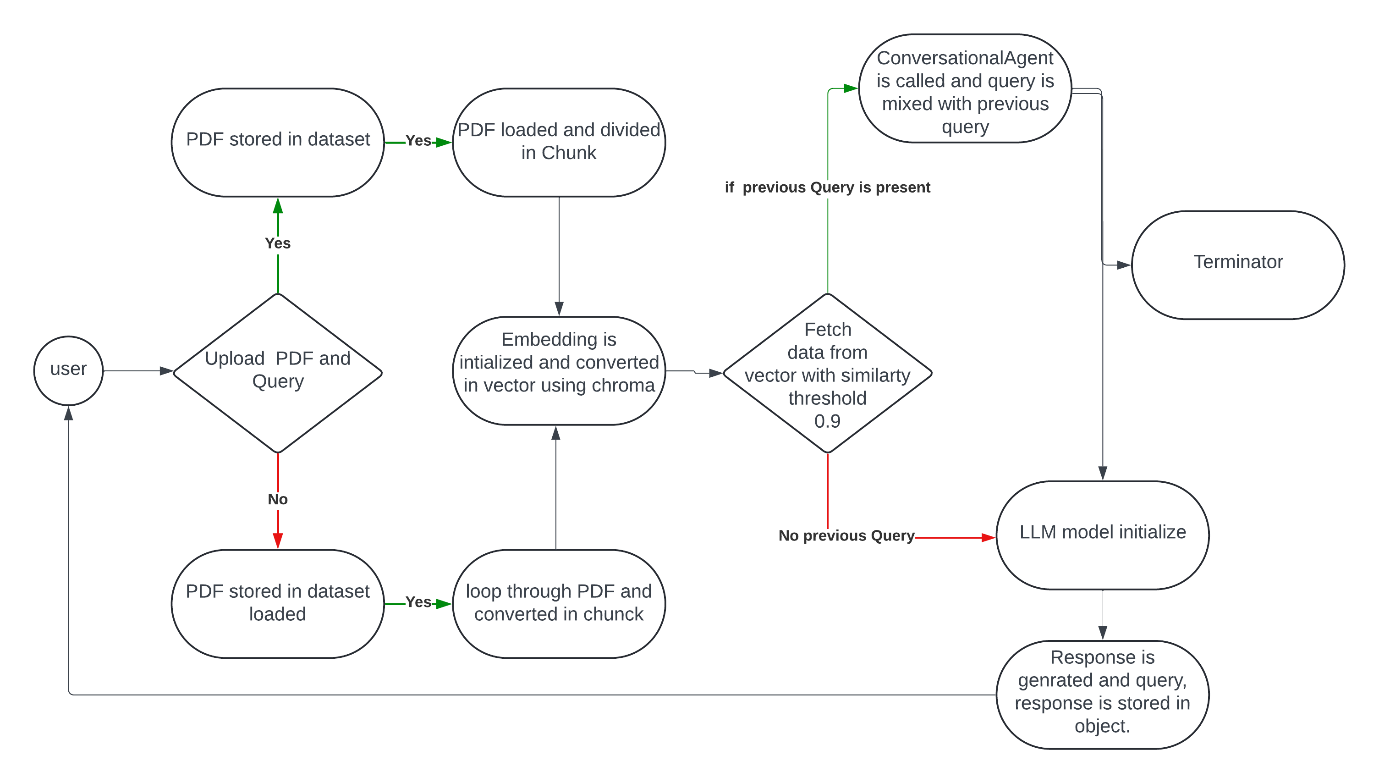
PROCESS PLANNING OF CHATBOT

**OVERVIEW OF THE PROCESS IN DIAGRAM:**

****

**Step-by-Step Process**

**1. Requirements:**

* Data collection: gone through many articles to finalizing a best article and converted to pdf using AI tool.
* API selection: search for various model like GPT-3 by openAI ,transformer from hugging face and langchain. Finalized langchain for its modular and flexible framework for building advanced NLP applications with a focus on chaining operations and managing conversational context.
* Model selection: Together offers many model in which meta-llama/Llama-3-70b-chat-hf is collection of pretrained and instruction tuned generative text models in 70B sizes from hugging face. The cost is also average ($0.9) per 1M token.
* Finalized LangChain for document loading, text splitting, and embeddings.

**2. Data Preparation**

Data preparation involves collected PDF, loading, and processing the documents to make them suitable for query-answering tasks.

**Loading PDF Documents:**

* Use PyPDFLoader from langchain\_community.document\_loaders to load documents from both files and folders.

**3. Text Splitting**

**Splitting Documents:**

* Use RecursiveCharacterTextSplitter to split documents into manageable chunks (2242) for processing.

**4. Embeddings and Vector Store**

**Generating Embeddings:**

* Use SentenceTransformerEmbeddings to convert document chunks into embeddings.
* An embedding function converts text data into numerical vectors (embeddings) that can capture the semantic meaning of the text. So that it make easy to integrate with vector stores.

**Vector Store:**

* Use Chroma for storing and retrieving document embeddings.

**5. Language Model Integration**

**Using Together API:**

* Set up the Together LLM for generating responses based on queries.

**6. Building the Query Agent**

**Prompt Template:**

* Define a prompt template for the LLM to generate responses.
* We can define the template so that it show

**RetrievalQA Chain:**

* Use the RetrievalQA chain from LangChain to create a query-answering pipeline.

**Conversational Agent:**

* Implement a class to handle the conversational context and history.

#### 7. Streamlit User Interface

**Building the UI:**

* Use Streamlit to create a simple user interface for uploading files and entering queries so that easy testing/query input can be done.

### Future Enhancements

* Can add a list for **a topic related search**. So that only relevant topics are used.
* **Optimize** the **data preparation process** so that a faster query can be generated.
* Can add **logs so that if its crash**, it should be easy to debug.