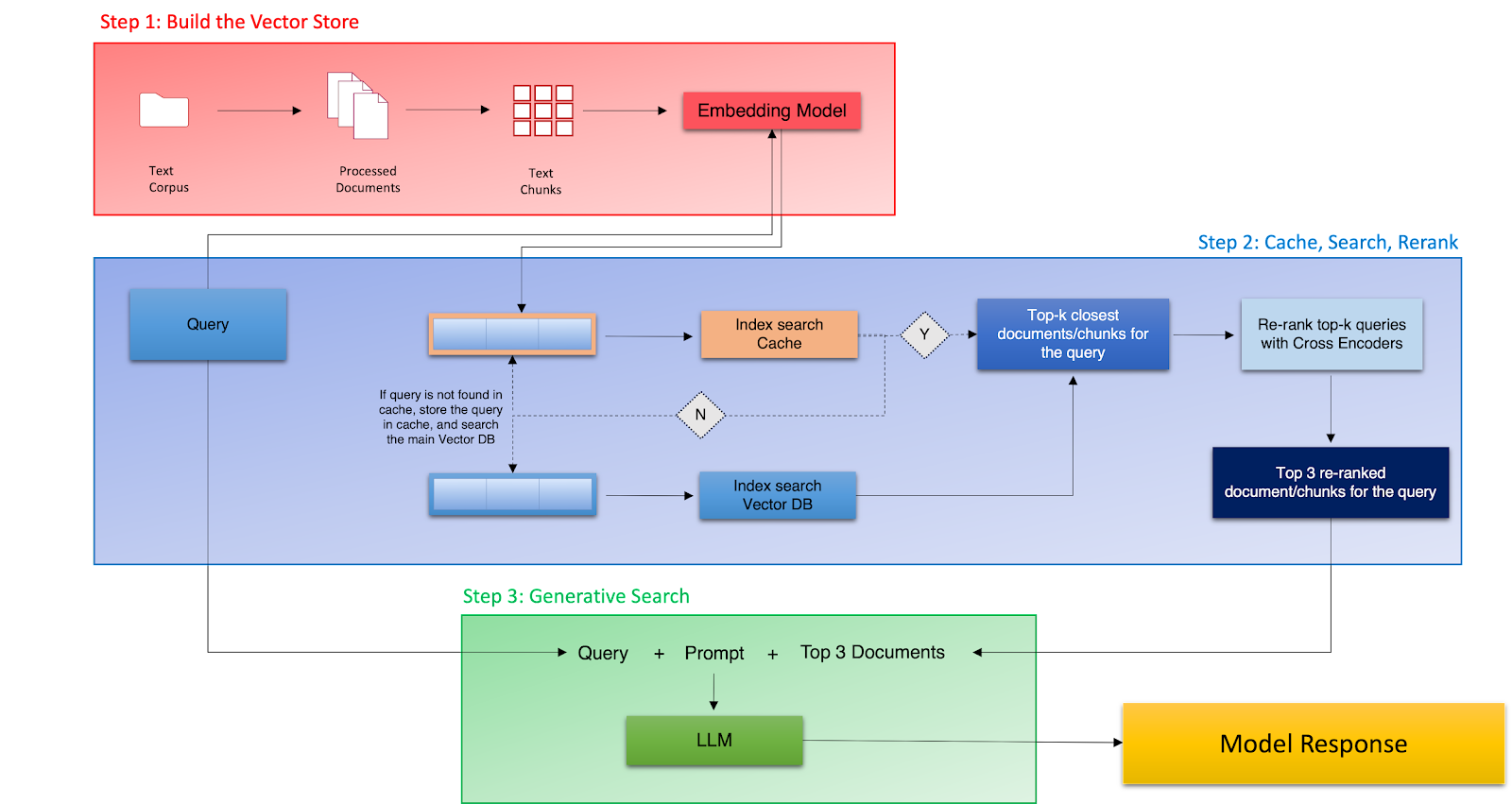
**Overview:**

"Mr Helpmate AI " is designed to perform generative search by utilizing OpenAI’s language model and ChromaDB, a database designed to handle embeddings and vector-based search.



Below is an overview of its structure and components:

### 1. **Setup and Import Dependencies**

* **Dependencies**: The first section involves installing essential libraries, including pdfplumber, tiktoken, openai, chromaDB, and sentence-transformers, which are needed to work with PDF files, process text, interact with OpenAI's API, and perform vector searches.
* **Imports**: Necessary Python packages are imported, such as pdfplumber for reading PDFs, openai for API access, and chromaDB for database management.

### 2. **PDF Preprocessing**

* This part of the notebook focuses on reading and processing PDF files using pdfplumber.
* **Single PDF Processing**: A sample PDF file path is specified, and its contents are extracted (including tables, if any).
* **Multiple PDF Processing**: Plans for processing multiple PDFs by iterating through them, extracting text, and saving it in a structured format such as a DataFrame. This setup is aimed at handling a batch of PDF documents.

### 3. **Embedding Generation and Storage**

* **Text Processing and Tokenization**: Once the text is extracted from PDFs, tokenization and encoding are done using tiktoken to optimize and prepare the text for embedding.
* **Embedding**: OpenAI’s API is used to convert the preprocessed text into embeddings, which are vector representations of text useful in semantic search.
* **Storage in ChromaDB**: These embeddings are then stored in ChromaDB, allowing for efficient search and retrieval of similar or relevant information based on vector-based queries.

### 4. **Search and Retrieval**

* **Generative Search Query**: Using a text-based input, the notebook demonstrates how to query the stored embeddings and retrieve the closest matches. This is likely intended to support natural language searches, where users input a query, and the system responds with relevant document snippets.
* **Summarization and Response Generation**: The retrieved information could be further processed or summarized, potentially with OpenAI’s language model, to provide a coherent answer to the user's query.

### Summary

Overall, the notebook combines document preprocessing, embedding generation, and vector-based search for creating a generative AI-based search system. It is structured to handle large document collections and facilitate semantic search by leveraging advanced language models and a robust vector database. The architecture appears scalable, designed for efficient handling, storage, and retrieval of embeddings that facilitate natural language query responses.