**RAG System - Simplified Documentation**

**1. Project Overview**

The Retrieval-Augmented Generation (RAG) system allows users to upload PDFs, extract their content, and ask questions about them. It uses LlamaIndex to process the documents, ChromaDB to store text embeddings, and OpenAI’s GPT-4o-mini to generate answers.

**2. How It Works**

1. **User Input**: Users upload PDFs and type queries.
2. **PDF Processing**: LlamaIndex extracts text from the PDF.
3. **Embedding Generation**: The extracted text is converted into vector embeddings using OpenAI’s embedding model.
4. **Storage**: These embeddings are stored in ChromaDB for fast retrieval.
5. **Query Handling**: When a user asks a question, the system converts the query into embeddings, finds the most relevant document chunks, and retrieves them.
6. **Generating an Answer**: The retrieved text and the user’s question are sent to GPT-4o-mini, which generates a concise response.
7. **Exit Option**: Users can type 'exit' at any time to end the session.

**3. Tech Stack**

* **Language**: Python (easy to use and widely supported in AI development)
* **Libraries**:
  + *LlamaIndex*: Connects language models with external data, enabling document indexing and retrieval.
  + *ChromaDB*: A vector database for fast and efficient storage of text embeddings.
  + *OpenAI*: Provides GPT-4o-mini for text generation and embedding creation.
  + *python-dotenv*: Manages API keys securely by loading them from a .env file.
* **Models Used**:
  + GPT-4o-mini
  + text-embedding-3-small
* **Vector Database**: ChromaDB

**4. Setup Instructions**

**Prerequisites**

* Install a python supported IDE
* Get an **OpenAI API Key**

**Steps to Set Up the Environment**

1. **Create a Virtual Environment**:

python -m venv venv

1. **Activate the Environment**:

Python scripts\Activate

1. **Install Required Libraries**:

pip install llama-index chromadb openai python-dotenv

1. **Add OpenAI API Key**:
   * Create a .env file in the project folder.
   * Add the following line:

OPEN\_AI\_KEY=your\_openai\_api\_key

1. **Run the Application**:

py app.py

**5. Code Overview**

* **Loading PDFs**: LlamaIndex reads and extracts text from PDFs.
* **Generating Embeddings**: OpenAIEmbedding from LlamaIndex converts text into embeddings.
* **Storing Data**: ChromaDB saves embeddings in a collection named rag\_collection.
* **Handling Queries**: When a user asks a question, the system finds relevant text chunks from ChromaDB and sends them to GPT-4o-mini to generate a response.

This document provides a simple overview of the RAG system, how it works, the technology behind it, and how to set it up efficiently.