# RAG-Based AI Chatbot - Developer Documentation

## Introduction

This documentation provides a step-by-step guide to understanding and working with the RAG-based AI chatbot. The chatbot retrieves relevant documents using ChromaDB and generates responses using the Mistral-7B model.

## 1. Project Overview

This chatbot follows the Retrieval-Augmented Generation (RAG) approach, which combines document retrieval and AI-generated responses.

### ****Core Features:****

* **Document Processing:** Extracts text from PDFs, Word documents, and Markdown files.
* **Vector Storage:** Stores document embeddings using ChromaDB.
* **Retrieval System:** Fetches the most relevant document chunks based on user queries.
* **AI Response Generation:** Uses Mistral-7B to generate context-aware responses.
* **FastAPI Backend:** Exposes an API for chatbot interactions.
* **Streamlit Frontend:** Provides a user-friendly chatbot interface.

## screenshot-1741427497380

## 2. Setup Instructions

### ****2.1 Requirements Installation****

Ensure you have Python installed (preferably Python 3.8+). Install dependencies using:

pip install -r requirements.txt

### ****2.2 Download the Embedding Model****

Run the script to download the Mistral-7B-Instruct-v0.1-GGUF.

!huggingface-cli download TheBloke/Mistral-7B-Instruct-v0.1-GGUF mistral-7b-instruct-v0.1.Q4\_K\_M.gguf --local-dir ./models --local-dir-use-symlinks False

### ****2.3 Download the Embedding Model****

Run the script to download the MiniLM model:

python download\_model\_minilm.py

This script fetches the **all-MiniLM-L6-v2** model and caches it locally.

### ****2.4 Run the Document\_processor:****

!python document\_processor.py

This will process the documents and store chunks and embeddings.

### ****2.4 Run the Retrieval:****

This retrieves the embeddings.

!python retrieval.py

#please enter "exit" in it.

This will start an API server at http://localhost:8000.

### ****2.4 Run to login into Hugging face:****

from huggingface\_hub import login

#write here your hugging face token

# you can use this link to get more information https://huggingface.co/docs/hub/security-tokens

# please create a " write " token

huggingface\_token = "hf\_xxxxxxxxxxxxxxxxxxxxxxx"

login(token=huggingface\_token)

print("successfully logged in to hugging face")

### ****2.4 Run to Generate responses from LLM:****

!python response\_generator.py

### ****2.4 Run the Backend****

Start the FastAPI backend by running:

uvicorn test\_backend:app --host 0.0.0.0 --port 8000

This will start an API server at http://localhost:8000.

### ****2.5 Run the Frontend****

Launch the Streamlit UI using:

streamlit run test\_frontend.py

## 3. Code Breakdown

### ****3.1 Document Processing (document\_processor.py)****

This module handles document ingestion, text extraction, and vector storage.

* extract\_text\_from\_pdf() - Reads text from a PDF file.
* extract\_text\_from\_word() - Reads text from a DOCX file.
* chunk\_text() - Splits text into smaller chunks.
* store\_in\_vector\_db() - Generates embeddings and stores them in ChromaDB.

### ****3.2 Retrieval Module (retrieval.py)****

* search\_documents(query, top\_k=3) - Searches for the top K relevant document chunks in ChromaDB.

### ****3.3 Response Generation (response\_generator.py)****

* generate\_response(prompt, context) - Generates an AI response using Mistral-7B.
* select\_best\_context(query, retrieved\_chunks) - Chooses the most relevant document chunk.

### ****3.4 FastAPI Backend (test\_backend.py)****

* /query/ API endpoint processes user queries and returns AI-generated responses.

### ****3.5 Streamlit Frontend (test\_frontend.py)****

* **Displays a chat interface.**
* **Calls the FastAPI backend** to retrieve responses.

## 4. How it Works

### ****Step 1: Document Processing****

* Documents are processed and stored in ChromaDB.
* Text is chunked, and embeddings are generated.

### ****Step 2: Query Handling****

* The user enters a query.
* The backend searches the most relevant document chunks.
* The best-matching context is selected.

### ****Step 3: Response Generation****

* Mistral-7B generates a response based on the retrieved context.
* The response is sent to the user.