Amlgo Labs – Junior AI Engineer Assignment

Project Title: Document-based RAG Chatbot

Candidate Name: Ravishankar Singh

Submission Date: 05-07-2025

1. Project Overview

Objective:

To develop an AI-powered chatbot capable of answering user questions based on document content using the Retrieval-Augmented Generation (RAG) framework.

Problem Statement:

Manual navigation of lengthy PDFs, such as Terms and Conditions or Privacy Policies, is time-consuming. This project builds an interactive chatbot that provides fast, accurate, and document-grounded answers, improving accessibility and understanding.

2. Technical Architecture

Pipeline: User Query → Retriever → Vector DB (Chroma) → Prompt Template → LLM (Phi via Ollama) → Response (Streamed to User)

Steps:

- • PDF document is cleaned (headers, footers, HTML removed).
- • Text is chunked into 100–300 word segments using sentence-aware splitting.
- • Embeddings generated using all-MiniLM-L6-v2 (HuggingFace).
- • Chunks are stored in Chroma vector database.
- • Semantic search retrieves top-k relevant chunks for a query.
- • Retrieved chunks and user query are injected into a prompt template.

· • LLM (Phi) generates factual response.

· • Response is streamed in real-time (token-by-token).

· • Relevant source text is displayed alongside the answer.

3. Chunking & Embedding

Chunking Logic:

Used RecursiveCharacterTextSplitter with sentence-aware logic.
Each chunk size: 100–300 words.
Overlap of ~30 tokens was added to preserve context continuity.
Ensured natural breaks at sentence boundaries to improve embedding relevance.

Embedding Model:

Model: all-MiniLM-L6-v2 (via HuggingFace)
Reason: Fast, lightweight, well-suited for semantic retrieval.
Vector DB: Chroma — efficient local storage and semantic search.

4. Prompt Template

You are a helpful AI assistant. Use the following context from the document to answer the question factually.

Context:
{retrieved_chunks}

Question:
{user_query}

Answer:

The prompt ensures the model answers only using the given context.
Reduces hallucination and improves factual grounding.

5. Sample Queries & Answers

· Q1: What is the return policy?

The return policy states that customers can return items within 30 days of purchase as long as the product is unused and in original condition.

· Q2: Who can access my data?

Your data may be accessed by authorized employees, partners, and service providers, as outlined in the privacy section of the document.

· Q3: Is cancellation allowed?

Yes, cancellation is allowed within 24 hours of placing the order, provided the item has not been shipped.

· Q4: Can I exchange used products?

The model responded: "Exchange policy is not clearly defined." (Indicates limitation due to missing information in the document.)

· Q5: Are there any age restrictions?

Yes, users must be at least 18 years of age to create an account or use the service.

6. Observations & Limitations

Positives:

· • Real-time streaming enhances interactivity and user engagement.

· • High accuracy due to context-grounded generation from vector store.

· • Lightweight: Can run locally on CPU (no GPU or cloud needed).

Limitations:

· • Responses may be short or vague for edge-case queries.

· • Current version supports only single-document QA.

· • Chunking is static; dynamic chunking based on headings can improve future accuracy.

7. Tools & Libraries Used

• **LangChain** – Used for building the RAG pipeline, including the retriever and prompt generation.
• **HuggingFace** – Used to load the embedding model (`all-MiniLM-L6-v2`) for semantic similarity.
• **ChromaDB** – Serves as the vector database to store and retrieve document chunks efficiently.
• **Streamlit** – Powers the user interface of the chatbot with support for real-time answer streaming.
• **Ollama + Phi** – Lightweight local LLM used to generate the final responses based on context.
• **Python** – Used for scripting, integration, and overall orchestration of the application.

8. Conclusion

This project demonstrates how open-source tools can be combined to build a robust and efficient document-based chatbot using the RAG approach. The real-time response experience and document-grounded outputs show promise for customer support, legal tech, and document summarization applications.

Note: Demo video and GitHub repo link are included in README.md (as per submission instructions).