AI Boot Camp FInal Project

Retrieval-Augmented Generation (RAG) LLM

Team Members:

Jeff Flachman

Kerek Spinney

Ashwini Kumar

Exec Summary

Purpose:

Local Large Language Model (LLM) capable of reviewing, summarizing, and leveraging proprietary documents to facilitate a local knowledge base.

Key Features:

- Utilize RAG (Retrieval-Augmented Generation) Workflow to create and query the datastore of documents
  - Loads documents (e.g. pdf, word, etc) into a vector database
  - Use LLM answer query based on the content from the data files
  - Provides a summary of the content based on a query
- Quick knowledge base retrieval
- Can be deployed on a laptop or url
- Scaleable

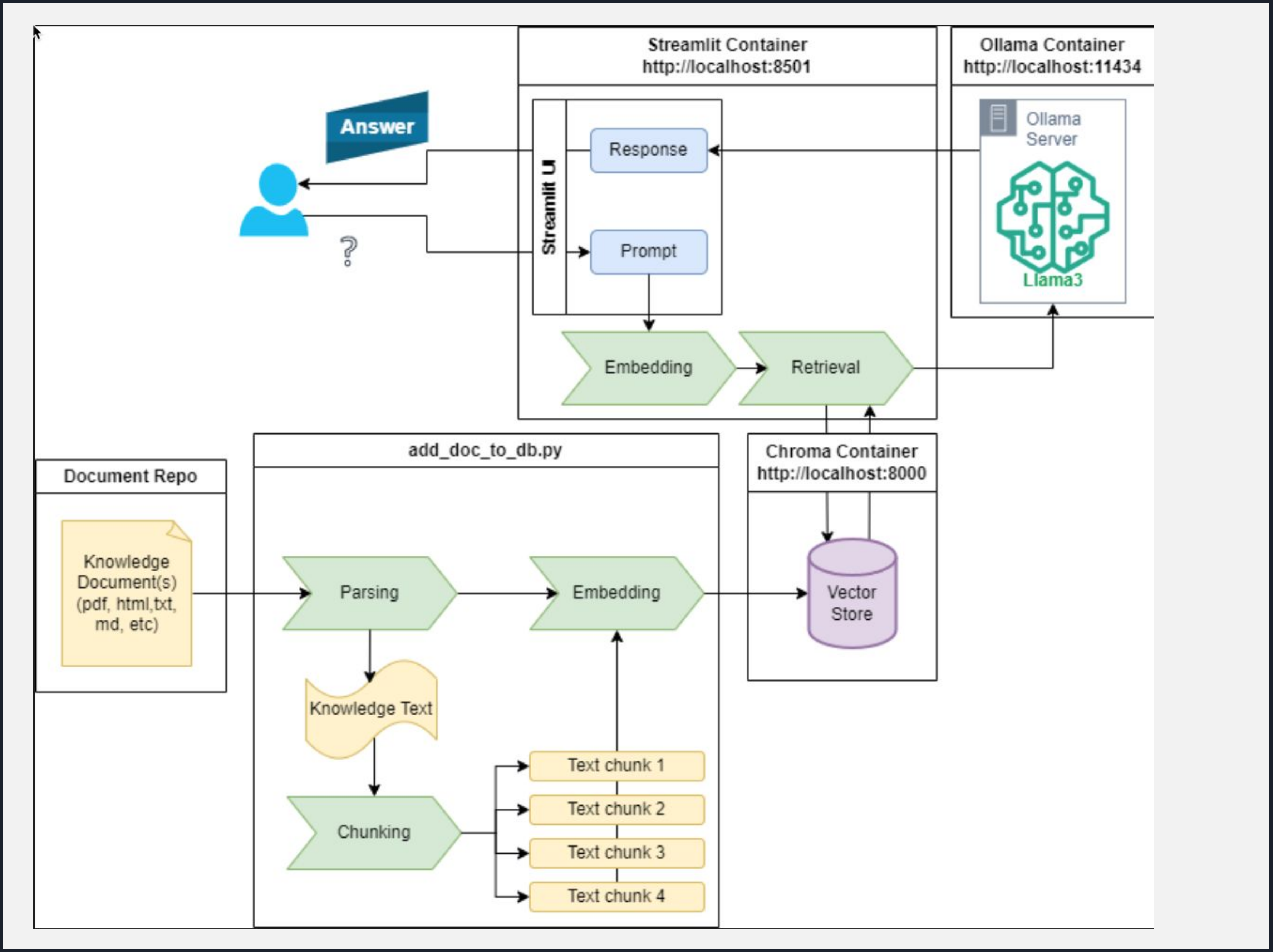
Goals

1. Build Prototype version using RAG and LLM
2. Load data related to “GenAI”
3. Give the user the ability to ask a knowledge base questions and return with accurate answers

Data Collection & Cleaning

1. **Researched and Gathered information related to GenAI.**
2. **Saved all content as PDF and loaded into Data folder**
3. **PDFs are a graphic method for printing a document.**
   a. They do not contain attributes like header, footer, footnotes, etc.
   b. These artifacts end up in the middle of paragraph the spans two pages.
   c. This greatly impacts the effectiveness of a RAG solution**.**
4. **Text files and Word Documents are better formats** to import with fewer document artifacts impacting the effectiveness of the solution

Technical Approach



Model Optimization

Importing documents:

- **Chunk/Splitting the text can be done in many different ways.**
  - **Word Tokens:** Or random groups of works is too Simple
  - **Implemented:** Split into sentences as chunks
  - **Future Goal:** Semantic splitting groups sentences that talk about the same type of information using embeddings of each sentence group

LLM Selection:

- **Trade off between model size and model robustness**
- **Used Huggingface Leaderboard:** to select the best performing model for a model size.
- **Running on a local laptop limits the size of the models that can be used.**
  - More performance models are ~0.5 Billion parameters model vs 7b or 70b.
  - Selected 0.5b model to provide perfant response on local system.

Demo

DEMO

Challenges Encountered

- **Setting up Docker.**
- **Finding materials online to assist in our project.**
    - Sort through many different approaches that did not necessarily meet our deployment methods.
- **Combining all portions of the project and have it run.**
- **Taking on a large project.**
- **Using PDFs.**

Future Considerations

- Test other LLM model options.
- Test other chunking/splitting functions.
- Test additional data and file types
- Work with Streamlit to add a user interface.
- Secure access

Streamlit

Questions?

Technical Approach

- **Python file** to read, convert, chunk and load documents into the Vector Database
- **Vector Database:** used ChromaDB in a docker container to provide easy configuration, deployment and use of the vector database
- **User Interface python file**
  - **Uses Streamlet** to allow users to prompt questions that are answered based on the documents loaded into the vector databse
  - **Uses the RAG workflow**
    - Take a prompt, and get closest matching document chunks from ChromaDB
    - Submit the Chunks as contact along with a system context and the users prompt to the LLM
    - Displays the Response based on the document chunks back to the user.
- **LLM Server** using llama-cpp-server in a docker container that loads any number of LLM models for use in the solution.