**RAG Project Walkthrough**

**👋 Introduction**

"Hi, I’d like to walk you through a **Retrieval-Augmented Generation (RAG) system** I implemented using **LangChain**, **Chroma**, **OpenAI embeddings**, and a **Gradio UI**. The goal was to build an intelligent Q&A assistant that answers user queries based on a custom text corpus—in this case, the *2024 State of the Union* address."

**📁 1. Data Ingestion and Preprocessing**

"First, I load a .txt file containing the source document. Since LLMs have input size limitations, I split the document using CharacterTextSplitter into overlapping chunks—each 1000 characters long with 200-character overlap."

**🔍 2. Embeddings and Vector Store Setup**

"Next, I initialize OpenAIEmbeddings using the text-embedding-3-small model. I use Chroma as the vector database to store and index those embeddings. Each text chunk is converted into a vector and stored under a named collection."

**🧠 3. Similarity Search for Retrieval**

"To test retrieval, I run a similarity search with the question *'Who invaded Ukraine?'* and retrieve the top 2 chunks based on cosine similarity. These are later passed as context into the language model."

**📜 4. Prompt Template Creation**

"I define a prompt template that guides the model to use the retrieved context to answer questions accurately and concisely. It also includes safety instructions like not hallucinating answers."

**🔗 5. LangChain RAG Pipeline**

"I construct a RAG chain using LangChain’s composable architecture. It flows like this:

* Retriever fetches relevant chunks.
* RunnablePassthrough passes the query forward.
* PromptTemplate integrates the query + context.
* LLM generates an answer.
* StrOutputParser formats the final result."

**🧪 6. LLM Choice: OpenAI & HuggingFace**

"By default, I use ChatOpenAI with gpt-4o-mini, but I also included HuggingFaceHub with zephyr-7b-beta for flexibility or open-source deployment scenarios."

**🖥️ 7. Gradio UI**

"Finally, I created a Gradio interface so users can type questions and get answers in real-time using the RAG chain behind the scenes. This makes it easy to demo or share the app."