Übungsaufgabe: RAG-Systemvariationen

In dieser Übung sollt Ihr ein vorhandenes Retrieval-Augmented Generation (RAG) System **anpassen, erweitern und reflektieren**. Ihr arbeitet auf Basis eines vorbereiteten **Jupyter Notebooks**, das ein einfaches RAG-System mit **LangChain**, **OpenAI**, **Chroma** und Standard-Prompting verwendet.

Ziel ist es, dass Ihr die Struktur eines RAG-Systems versteht und konkret erlebt, welche Komponenten **austauschbar** sind und wie sich diese Veränderungen auf Funktionalität und Effizienz auswirken.

- Das Notebook `genai_rag.ipynb` enthält:
  - eine Textquelle (z. B. Interview)
  - ein ChromaRetriever
  - Chunking mit einem `CharacterTextSplitter`
  - ein PromptTemplate
  - das Modell `ChatOpenAI` (GPT-4o-mini) über LangChain
- Ihr könnt den OpenAI API Key des Dozenten verwenden (mit den genannten Einschränkungen bzgl. Modelle und RPM/TPM).

✅ Eure Aufgabe (alle Gruppen machen alles!)

Bearbeitet das Notebook so, dass ihr die folgenden **7 Aspekte bewusst verändert, verbessert oder ersetzt**. Kommentiert eure Änderungen im Code und dokumentiert eure Beobachtungen in Markdown-Zellen.

Aspekt	Was ihr verändern sollt	Beispiel-Alternative
1. Modellzugang	LangChain-Zugriff (`ChatOpenAI`)	OpenAI SDK
2. LLM-Modelltyp	OpenAI	Open-Source-Modelle: HuggingFace (Mistral, Llama 3), Ollama, LM Studio etc.
3. Retriever	ChromaRetriever	Anderer Vektorstore: FAISS, Weaviate, Pinecone, scikit-learn (manuell)
4. Chunking	aktuelle Chunking-Strategie	andere Länge, Overlap, semantisches Chunking (NLP)
5. Prompt-Template	Prompt umschreiben, Varianten testen mit klaren Rollen oder Struktur	few-Shot, strukturierte Antworten, „you are a helpful bot“
6. RAG-Logik modularisieren	Retrieval, Prompting und Inferenz in eigene Funktionen trennen	retrieve(), format_prompt(), call_llm() usw.