NSK AI

Phase 1 Final Project – Naive RAG Chatbot

Objective

Build and deploy your own Naive Retrieval-Augmented Generation (RAG) system that can answer user questions based on custom documents of your choice.

The goal is to demonstrate your ability to apply all concepts learned in Phase 1, from document ingestion to retrieval and generation.

Project Scope

You have complete freedom to decide:
- Document source: YouTube transcript, PDF, website scraping, plain text files, etc.
- Embedding model: OpenAI, HuggingFace, etc.
- Vector store: FAISS, Chroma, Pinecone, Weaviate, etc.
- Reranker: Optional — you can integrate Sentence transformers CrossEncoders, Cohere Rerank, BM25 + dense retrieval, or others.
- Framework: Langchain in Python is the default, but if you have prior knowledge, you can use FastAPI, JavaScript/TypeScript (Langchain.js), or other supported languages.

The final application should support interactive Q&A over your chosen documents.

Minimum Requirements

- **Document Ingestion: (5 Points)**

Load data using one or more Langchain document loaders. Apply an appropriate chunking strategy (size and overlap).

- **Indexing and Storage: (5 Points)**

Generate embeddings using a chosen embedding model. Store vectors in your chosen vector database.

- **Retrieval: (5 Points)**

Implement at least one retrieval method (dense, sparse, or hybrid). Return relevant chunks for a user query.

- **Generation: (5 Points)**

Use Langchain's Prompt Templates to structure LLM input. Implement a Runnable or chain that takes the user query, retrieves relevant context, and generates a grounded answer.

- **User Interaction: (5 Points)**

Support one of the following: CLI/terminal interaction, Jupyter Notebook/Google Colab, Web app (Streamlit, Gradio, Next.js, etc.), or API endpoint (e.g., via FastAPI).

You are expected to make your submissions individually. However, you are expected to work with your team members for the project.
You cannot submit the same project with your teammates

Stretch Goals (Optional) (3 points each)

- Implement a reranker to improve retrieval quality.
- Allow switching between multiple document sets.
- Add a summarization mode in addition to Q&A.
- Deploy your app to the cloud (e.g., Vercel, Streamlit Cloud, Render, HuggingFace Spaces).

TOTAL SCORE (30/30)

Deliverables

Each participant must submit via the provided form:

1. GitHub Repository URL containing:
   - Complete, working source code
   - README.md with:
     - Description of the project and chosen tech stack
     - Setup instructions
     - Example queries and answers
     - Any limitations or known issues
2. (Optional but encouraged): Live deployment link.

Evaluation Criteria

- Core functionality – All required components implemented correctly.
- Code clarity – Clean, well-structured, and readable code.
- Understanding of concepts – Correct usage of Langchain constructs, retrieval methods, and prompt templates.
- Creativity – Choice of data source and extra features.
- Deployment – Bonus points for working hosted apps.

Learning Outcomes

By completing this project, you will:

- Gain practical experience building an end-to-end RAG system.
- Apply indexing, retrieval, and generation concepts from Weeks 1–3.
- Understand the impact of embeddings, retrieval methods, and prompt design on output quality.
- Build a portfolio-ready project that demonstrates your Phase 1 skills.

Submit here: [Task Submit Form]

Deadline for project: **Wednesday, August 20 2025. 10PM WAT (Extended)**

All bootcamp resources on our youtube: https://www.youtube.com/@nskaicommunity

And Github: https://github.com/nsk-ai/RAG-Bootcamp-2025

You are expected to make your submissions individually. However, you are expected to work with your team members for the project.

You cannot submit the same project with your teammates