Capstone Project 2: RAG-Based Internal Knowledge Assistant using LLAMA + FAISS

1. Problem Statement

Organizations often store their internal knowledge—policies, technical documents, SOPs, guidelines, product manuals, compliance rules, department handbooks—across PDFs, Word files, and text documents.
Employees spend a large portion of their time **searching**, **scrolling**, and **manually reading** these documents.

The goal of this capstone is to build a **Retrieval-Augmented Generation (RAG) based AI Assistant** using **LLAMA**, **LangChain**, and **FAISS** that can instantly answer user queries from internal documents using natural language.

Instead of fine-tuning a model, the system will:

1. **Retrieve** relevant chunks from documents
2. **Augment** the query with retrieved context
3. **Generate** an accurate answer using the LLAMA model

This creates a high-accuracy, domain-aware question-answering assistant.

2. Objective

- Build a complete RAG pipeline using LLAMA.
- Load multiple documents and prepare them for retrieval.
- Generate embeddings and store them using FAISS.
- Build query → retrieval → LLAMA generation pipeline.
- Provide an end-to-end Python-only implementation (no UI).
- Make the solution reusable for enterprise document search.

3. RAG Workflow Overview

Your system must implement the following flow:

1. Document Loading

Load PDFs/text documents from a folder.

2. Text Chunking

Split documents into chunks of 300–800 tokens with overlap.

3. Embedding Generation

Generate vector embeddings using LLAMA-compatible embeddings model.

4. FAISS Vector Index Creation

Store embeddings in FAISS index
→ Enables high-speed similarity search.

5. Retrieval Layer

Perform semantic search to retrieve top-k most relevant chunks.

6. Augmented LLM Response using LLAMA

Send **query + retrieved context** to LLAMA to generate final answer.

4. Dataset / Document Inputs

You can use any of the following as sample inputs:

- Company policy PDF
- Safety guidelines

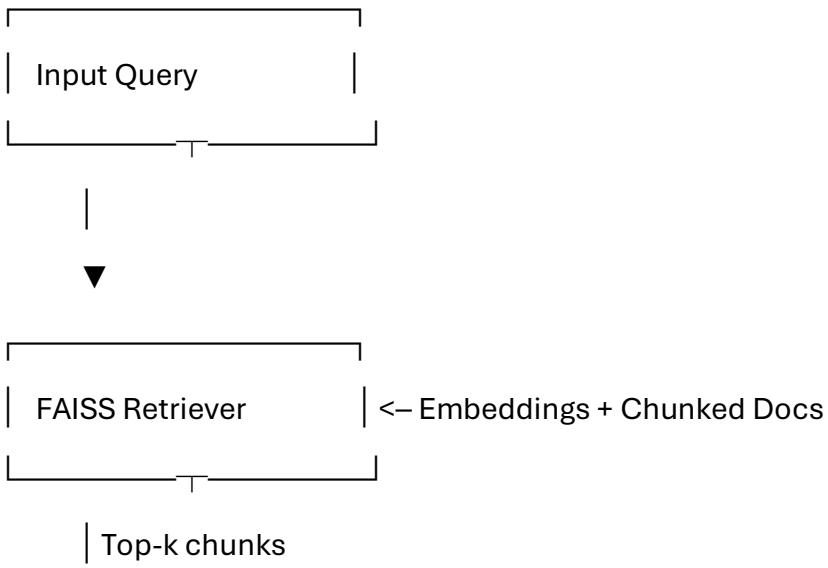
- HR handbook
- Product technical documentation
- Compliance documents
- Research reports
- Custom text files

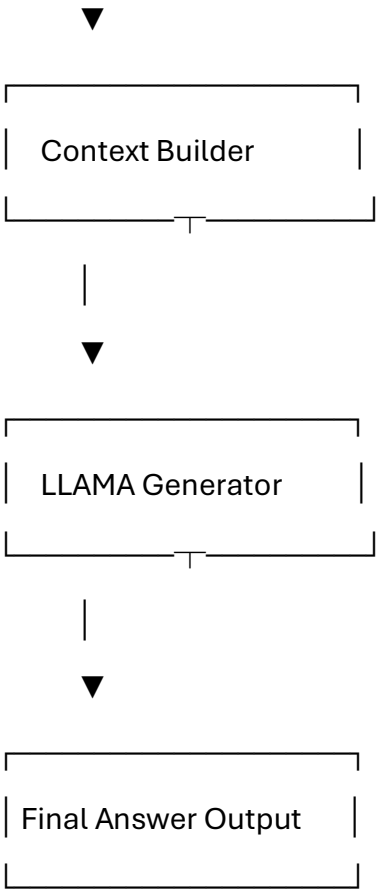
(Instructor may provide sample PDFs)

5. Tools & Frameworks

- **LLAMA model** on IBM Watson Studio
- **LangChain** (for RAG orchestration)
- **FAISS** (vector store)
- **Hugging Face Transformers**
- **PyPDF2 / pdfplumber / unstructured** for PDF reading
- **Python** (full coding-based implementation)

6. System Architecture





7. Project Tasks (What must build)

A. Document Loader & Preprocessor

- Load PDFs / text files
- Extract text
- Clean and normalize text
- Chunk text with overlap

B. Embedding Creation

- Use LLAMA-compatible embedding model

- Generate vector embeddings for each chunk

C. FAISS Vector Store

- Create FAISS index
- Insert embeddings + metadata
- Save & load FAISS index

D. Retrieval Pipeline

- Semantic search over FAISS
- Return top-k relevant chunks
- Format results for context injection

E. LLAMA Generation Pipeline

- Construct final prompt:

```
Context:

<insert retrieved chunks here>


Question:

<user query>


Answer:
```

- Use LLAMA model on IBM Watson Studio for inference
- Generate accurate and grounded output

F. End-to-End RAG Script

Single Python script performing:

- Load docs → chunk → embed → store → retrieve → answer

Example:

python rag_pipeline.py --query "What is our leave policy?"

8. Evaluation Metrics

- **Context relevance** (qualitative)
- **Answer accuracy** by manual comparison
- **Latency & retrieval speed**
- **Coverage**: Can it answer most document-related questions?

9. Expected Outcome

A fully functional **RAG-based Internal Knowledge Assistant**, capable of:

- Answering internal document questions instantly
- Reducing manual reading time
- Supporting HR, IT, Quality, and Compliance teams
- Running fully on code (no UI required)
- Deployable as a backend service