Gen AI Training Assignment - 2: Building a Retrieval-Augmented Generation (RAG) Application

Objective

Learners will build a complete RAG pipeline from data ingestion through retrieval and generation using open-source tools (LangChain, Hugging Face models, Chroma/FAISS, Llama 2/Gemini). They will demonstrate each step with code and evaluate performance and quality.

1. Dataset Selection

Choose a Kaggle text dataset (≥10,000 entries). Examples:

- SMS Spam Collection
- Amazon Product Reviews
- Enron Email Dataset

Task: Download via Kaggle API or web UI and list files.

2. Data Ingestion & Parsing

Goal: Read raw files into a cleaned DataFrame.

Steps:

1. Load into Pandas.
2. Clean text (remove HTML, normalize whitespace, drop nulls).

Deliverable: data_ingest.py script that outputs a DataFrame with id, text, and metadata.

3. Chunking Documents

Goal: Split long texts into ~500-token passages.

Steps:

1. Implement chunk_text(text, size=500).
2. Track doc_id and chunk_id.

Deliverable: chunker.py with a function returning (doc_id, chunk_id, chunk_text) triples.

4. Embedding Chunks

Goal: Embed each chunk into vector space.

Choose two Hugging Face models:

instructor-large

RedHatAI/bge-large-en-v1.5-quant

Steps:

1. Load models via Transformers.

2. Generate embeddings batch-wise.

Deliverable: embedder.py; compare embedding dims and speed.

5. Vector Database Indexing

Options:

Chroma: Local, metadata support.

FAISS: Scalable, multiple index types.

Steps:

1. Connect to chosen DB.
2. Insert (chunk_id, embedding, metadata).
3. Test k-NN retrieval.

Deliverable: vector_store.py with indexing & simple search.

6. LangChain Retrieval Pipeline

Components:

1. Document loader from chunks.
2. HF embedding wrapper.
3. Chroma/FAISS vector store integration.
4. Similarity retriever (top-k).

Deliverable: app.py defining query(text) → answer using RetrievalQA.

7. LLM Integration for Generation

Choose at least one:

Llama 2 (7B or 13B) via Transformers

Google Gemini API free tier

Steps:

1. Configure LangChain LLM to call your model.
2. Measure answer latency and quality.

8. Evaluation & Reporting

- Assess Retrieval Quality: Inspect top-k contexts.
- Assess Generation Quality: Relevance, factuality.
- Performance Metrics: Indexing time, query latency.

Deliverable: report.md summarizing results, challenges, and recommendations.

Submission Checklist

data_ingest.py

chunker.py

embedder.py

vector_store.py

app.py

report.md