"The Shakespearean Scholar" - A Containerized RAG System

General Instructions

- There must be **ONLY ONE** submission made per group but all the group members should contribute towards completing the assignment.
- Institute policies will apply in cases of **plagiarism**.
- Create separate .ipynb or .py files for each part. The file name should follow the format: “A2_.ipynb/.py”
- Create a single .ipynb file to generate the final outputs that are required for submittables. It should be named as “A2__infer.ipynb”. Clearly indicate which cell corresponds to the output of which task/subtask. Outputs will be checked from this inference file only by TAs.
- Carefully read the deliverables for all tasks. Along with the code files, submit all the other files mentioned for each task, strictly following the naming convention instructed.
- **Only one** person has to submit the .zip file containing all the mentioned files and the report PDF. It will be named “A2_.zip”. The person with the alphabetically smallest name should submit it.
- You are required to submit your trained models. You must also retain all your checkpoints and load and run them during the **demo**.
- Your report must include the details of each group member’s contribution.

1. Assignment Overview

Objective: To design, implement, and evaluate a full-stack, containerized Retrieval-Augmented Generation (RAG) system. This system will function as an expert AI tutor on William Shakespeare's *The Tragedy of Julius Caesar*.
The Dataset: You will use the provided Folger Edition PDF of *Julius Caesar* (julius-caesar.pdf) as the sole knowledge base. This dataset has been intentionally chosen. It is a non-trivial parsing challenge due to its formatting (headers, footers, line numbers, and table-based dialogue), which will require a robust data engineering pipeline before any AI work can begin.
Target Persona: The final system must adopt the persona of an "Expert Shakespearean

Scholar." Its target audience is an ICSE Class 10 student, so answers must be insightful, academically rigorous, clear, and *always* cite textual evidence.
System Architecture: This is a systems-design assignment. You will not just build a notebook. You will build a containerized application with a clear separation of concerns, exposing its logic via an API.

2. Core Assignment Phases & Requirements

Phase 1: Data ETL (Extract, Transform, Load) & Chunking Strategy

This is the most critical engineering phase. A high-quality RAG system is built on high-quality data.

- **Extraction:** Write a robust script to extract all text from the PDF.
  - *Recommended Tools:* pdfplumber (good for table handling), PyMuPDF (fast), or unstructured.io.
- **Transformation & Cleaning:** This is a significant parsing task. Your script must:
  - Remove all "Folger edition artifacts": page headers, footers (e.g., "ACT 1. SC. 2"), page numbers, and "FTLN xxxx" line numbers.
  - Intelligently reconstruct the play's structure. Handle fragmented text and table-based dialogue (e.g., page 10) to create clean, structured data (e.g., a JSONL file).
- **Chunking Strategy:**
  - You must design and justify a logical chunking strategy.
  - **Naive fixed-size chunking is not acceptable** as it will break the semantic context of soliloquies and dialogue.
  - **Required:** Implement a logical chunking strategy (e.g., chunk by scene, by speech/soliloquy) and *enrich each chunk with metadata* (e.g., {"act": 2, "scene": 1, "speaker": "BRUTUS", "is_soliloquy": true}).

Phase 2: Indexing & Vector Store

- **Embedding Model:** Choose an appropriate open-source embedding model (e.g., bge-base-en-v1.5, nomic-embed-text-v1.5, or a baseline like all-MiniLM-L6-v2). Justify your choice.
- **Vector Store:** Index your prepared data chunks into a vector database.
  - *Recommended Tools:* ChromaDB (easy to persist and containerize) or FAISS.

Phase 3: RAG API Backend (Mandatory)

You must build an API-first application to serve your RAG pipeline. This is how we will evaluate your system.

- **Framework:** Use **FastAPI** (recommended) or Flask.
- **API Endpoint:** You must implement (at minimum) the following endpoint:
  - POST /query
  - **Request Body:** {"query": "Your question here..."}
  - **Response Body:** {"answer": "The scholar's answer...", "sources": [{"chunk": "...", "metadata": {...}}, ...]}
- **RAG Logic:** This API will orchestrate the full pipeline:

1. Take the user's query.
2. Embed the query.
3. Retrieve the top-k relevant chunks from the vector store.
4. Call the Generation model (see Phase 4).
5. Return the structured JSON response.

Phase 4: Prompt Engineering & Generation

- **Generation Model:** You have two choices:
  1. **Gemini API:** Use the Google Gemini API (e.g., gemini-2.5-flash-preview-09-2025 via langchain-google-genai). This is the simplest way to get high-quality generation.
  2. **Open Source (via Ollama):** For a fully containerized stack, use a model like Llama-3-8B or Mistral-7B served via Ollama.
- **System Prompt:** This is critical. You must design a detailed **system prompt** that instructs the LLM on its "Shakespearean Scholar" persona, its target audience, and its constraints (e.g., "You must *only* use the provided context," "You must cite your sources," "Your tone is academic and insightful").

Phase 5: Containerization (Mandatory)

The entire application must be runnable via a single docker-compose up command.

- **Dockerfile:** Create a Dockerfile for your Python backend API.
- **docker-compose.yml:** Create a docker-compose.yml file that orchestrates all necessary services. This will likely include:
  1. Your **FastAPI backend** service.
  2. A **Vector Database** service (e.g., chromadb/chroma).
  3. (If using) An **Ollama** service (ollama/ollama) to serve the local LLM.
  4. (If doing optional UI) A **Streamlit** service.

Phase 6: Evaluation Testbed (Mandatory)

You must evaluate your system's performance using a "golden set" of questions.

- **Baseline Testbed (evaluation.json):** We are providing a baseline testbed of 25 factual-retrieval questions. You must create an evaluation.json file in your repository, copy the content below into it, and use it as your starting point.

```
[
  {
    "question": "How does Caesar first enter the play?",
    "ideal_answer": "In a triumphal procession; he has defeated
the sons of his deceased rival, Pompey"
  },
  {
    "question": "What does the Soothsayer say to Caesar?",
    "ideal_answer": "“Beware the Ides of March”"
  },
  {
    "question": "What does Cassius first ask Brutus?",
    "ideal_answer": "Why he has been so distant and contemplative
```

```
lately"
  },
  {
    "question": "What does Brutus admit to Cassius?",
    "ideal_answer": "That he fears the people want Caesar to be
king"
  },
  {
    "question": "What does Antony offer Caesar in the
marketplace?",
    "ideal_answer": "The crown"
  },
  {
    "question": "That night, which of the following omens are
seen?",
    "ideal_answer": "All of the above (Dead men walking, Lions
strolling in the marketplace, Lightning)"
  },
  {
    "question": "What finally convinces Brutus to join the
conspirators?",
    "ideal_answer": "Forged letters planted by Cassius"
  },
  {
    "question": "Why does Calpurnia urge Caesar to stay home
rather than appear at the Senate?",
    "ideal_answer": "She has had nightmares about his death"
  },
  {
    "question": "Why does Caesar ignore Calpurnia's warnings?",
    "ideal_answer": "Decius convinces him that Calpurnia has
interpreted the dream and the omens incorrectly"
  },
  {
    "question": "What does Artemidorus offer Caesar in the
street?",
    "ideal_answer": "A letter warning him about the conspiracy"
  },
  {
    "question": "What do the conspirators do at the Senate?",
    "ideal_answer": "All of the above (Kneel around Caesar, Stab
him to death, Proclaim “Tyranny is dead!”)"
  },
  {
    "question": "What does Antony do when he arrives at Caesar's
body?",
    "ideal_answer": "All of the above (He weeps over Caesar's
body, He shakes hands with the conspirators, and he swears
```

```
allegiance to Brutus for the moment)"
  },
  {
    "question": "After the assassination of Caesar, which of the
conspirators addresses the plebeians first?",
    "ideal_answer": "Brutus"
  },
  {
    "question": "What is Brutus's explanation for killing
Caesar?",
    "ideal_answer": "Caesar was ambitious"
  },
  {
    "question": "What does Antony tell the crowd?",
    "ideal_answer": "All of the above (That Brutus is an honorable
man, That Caesar brought riches to Rome and turned down the crown,
That Caesar bequeathed all of the citizens a large sum of money)"
  },
  {
    "question": "What is the crowd's response to Antony's
speech?",
    "ideal_answer": "Rage; they chase the conspirators from the
city"
  },
  {
    "question": "Who is Octavius?",
    "ideal_answer": "Caesar's adopted son and appointed heir"
  },
  {
    "question": "Octavius and Antony join together with whom?",
    "ideal_answer": "Lepidus"
  },
  {
    "question": "Why do Brutus and Cassius argue?",
    "ideal_answer": "Brutus asked for money and Cassius withheld
it"
  },
  {
    "question": "What news do Brutus and Cassius receive from
Rome?",
    "ideal_answer": "All of the above (Portia is dead, Many
senators are dead, The armies of Antony and Octavius are marching
toward Philippi)"
  },
  {
    "question": "What appears at Brutus's bedside in camp?",
    "ideal_answer": "Caesar's ghost"
  },
```

```
  {
    "question": "What does Cassius think has happened to his and
Brutus's armies?",
    "ideal_answer": "He believes that they have been defeated by
Antony and Octavius"
  },
  {
    "question": "What is Cassius's response to this situation?",
    "ideal_answer": "He has his servant stab him"
  },
  {
    "question": "What does Brutus do when he sees the battle is
lost?",
    "ideal_answer": "He kills himself"
  },
  {
    "question": "What does Antony call Brutus at the end?",
    "ideal_answer": "The noblest Roman of them all"
  }
]
```

- **Your Task (Extension):** Your task is to **extend this evaluation.json file** by adding at least **10 new, more complex questions** of your own.
    - These new questions must be more analytical and thematic.
    - **Character Analysis (Add Your Own):** e.g., "What are Brutus's internal conflicts as shown in his soliloquy in Act 2, Scene 1?"
    - **Thematic/Complex (Add Your Own):** e.g., "Compare and contrast the role of fate vs. free will as seen through the actions of Caesar and Cassius."
- **Evaluation Report:** Write an EVALUATION.md report that includes:
    - Your *full* testbed (the 35+ questions).
    - The answers generated by your system.
        - **Quantitative Metrics:** Use a framework like RAGAs to score your pipeline on (at minimum) **Faithfulness** and **Answer Relevancy**.
        - **Qualitative Analysis:** Discuss *why* your system failed or succeeded on certain questions. (e.g., "Our scene-based chunking struggled with questions comparing two different scenes...").

Phase 7: Frontend UI (Optional, Good-to-Have)

- To visualize your project, create a simple UI.
- **Streamlit:** Create a frontend.py using Streamlit that calls your FastAPI backend. Add this as another service in your docker-compose.yml.
- **OpenWebUI:** If using Ollama, you can configure OpenWebUI as a service in your docker-compose.yml to provide a chat interface.

3. Submission Guidelines

1. **GitHub Repository:** All your code must be in a single, private GitHub repository. Add

your instructor as a collaborator.

2. **README.md:** The root of your repo must have a detailed README.md with:
    - Project overview.
    - Architecture diagram.
    - Justification for your design choices (chunking strategy, embedding model, LLM).
    - **Clear instructions on how to run your project using docker-compose up**.
3. **docker-compose.yml:** A single, working docker-compose.yml file.
4. **API Documentation:** Your FastAPI backend will auto-generate /docs. Ensure this is clean and usable.
5. **Evaluation Report:** The EVALUATION.md file containing your testbed and performance analysis.

4. Evaluation Criteria

You will be graded on the following:

- **System Functionality (15%):** Does the docker-compose up command successfully launch all services? Does the /query API work?
- **Data Pipeline Quality (30%):** How effectively did you parse the complex PDF? Is the data clean? Is your chunking strategy logical and well-justified?
- **RAG Quality (35%):** How good are the answers? (Based on your Evaluation Report, RAGAs metrics, and our own spot-checking). This includes the quality of your prompt engineering.
- **Evaluation & Analysis (20%):** The depth and thoughtfulness of your EVALUATION.md report. Did you create a strong testbed? Did you correctly identify your system's strengths and weaknesses?

5. Resources

Extra Datasets to improve quality

- link

Evaluation

- **Note on Evaluation Testbed:** The link below is a resource that may be helpful for generating *ideas* for the 10+ new analytical questions you are required to add. You must still use the 25 provided questions as your baseline testbed.
- link

Other Useful Resources

- https://medium.com/data-science-collective/run-an-llm-on-your-computer-using-ollama-docker-and-open-webui-b960f2df1d53
- https://docs.langchain.com/oss/python/langchain/rag
- https://huggingface.co/sentence-transformers
- https://youtube.com/playlist?list=PLZoTAELRMXVM8Pf4U67L4UuDRgV4TNX9D&si=0I

GgoCBWAEiMSqxw
- https://docs.docker.com/compose/

Submission Deadline

15/11/2025, EoD to be submitted in LMS.

Note: Late submissions will not be accepted. If you face any issues, contact TAs before the deadline.