Task: Implement a Dockerized RESTful RAG System

Objective: Create a Retrieval-Augmented Generation (RAG) system that implements a REST service and responds to user queries by retrieving relevant documents. You do not need to complete the entire task within the time limit—just focus on key components. *Optional*: Also generate a response using an LLM (Language Model) based on the retrieved documents.

You have 1 hour to complete this task. Your implementation will subsequently be discussed in a 30-minute, follow-up interview.

Requirements:

- REST API: Implement a public GET endpoint /query. The service should take a query parameter and return a JSON response. The expected response format is:

```
{
    "documents": ["<list of relevant documents found>"]
}
```

- *Optional*: Add a response field to the JSON response, where the LLM generates a response based on the relevant documents:

```
{
    "documents": ["<list of relevant documents found>"],
    "response": "<LLM response to query>"
}
```

Implementation:

- data: The RAG system should load documents from a folder called 'data' located in the project root directory. The files in the data folder can be assumed to be plain text files (.md, .txt, etc).
- container: The entire system must be Dockerized. Include a Dockerfile for easy setup and deployment.
- libraries & tools: The choice of language, libraries, and tools is completely up to the candidate, with the exception of:
    - **OpenAI API** must be used for embedding. (and the optional LLM response)
    - **Docker** must be used for containerization.
    - **GitHub** must be used for version control.

Deliverables:

- A public GitHub repository containing the solution.
- The repository should include clear instructions on how to build and run the project using Docker.

Please ensure that your repository is public and share the link by the end of the task.