FAB - RAG Chatbot

1. Project Overview

FAB-RAG is a lightweight and efficient Retrieval-Augmented Generation (RAG) system specifically engineered for interactive question answering over a custom dataset consisting of textual content about the history of Ancient Greece. Leveraging the capabilities of LangChain for orchestration, FAISS for fast similarity search, and OpenAI's `gpt-4o-mini` for natural language generation, FAB-RAG delivers an intuitive and responsive user experience. One of its key strengths lies in maintaining conversation history through session memory, allowing it to answer follow-up questions contextually, thereby simulating human-like dialogue.

2. Objective

The primary goal of FAB-RAG is to build a robust, conversationally aware QA system with the following objectives:

- **Answer Accuracy:** Extract and deliver precise answers from curated historical text files.
- **Conversational Context Retention:** Maintain session history to handle follow-up or related queries effectively.
- **Faithful Contextual Responses:** Ensure answers are grounded strictly in the retrieved documents to avoid hallucinations.

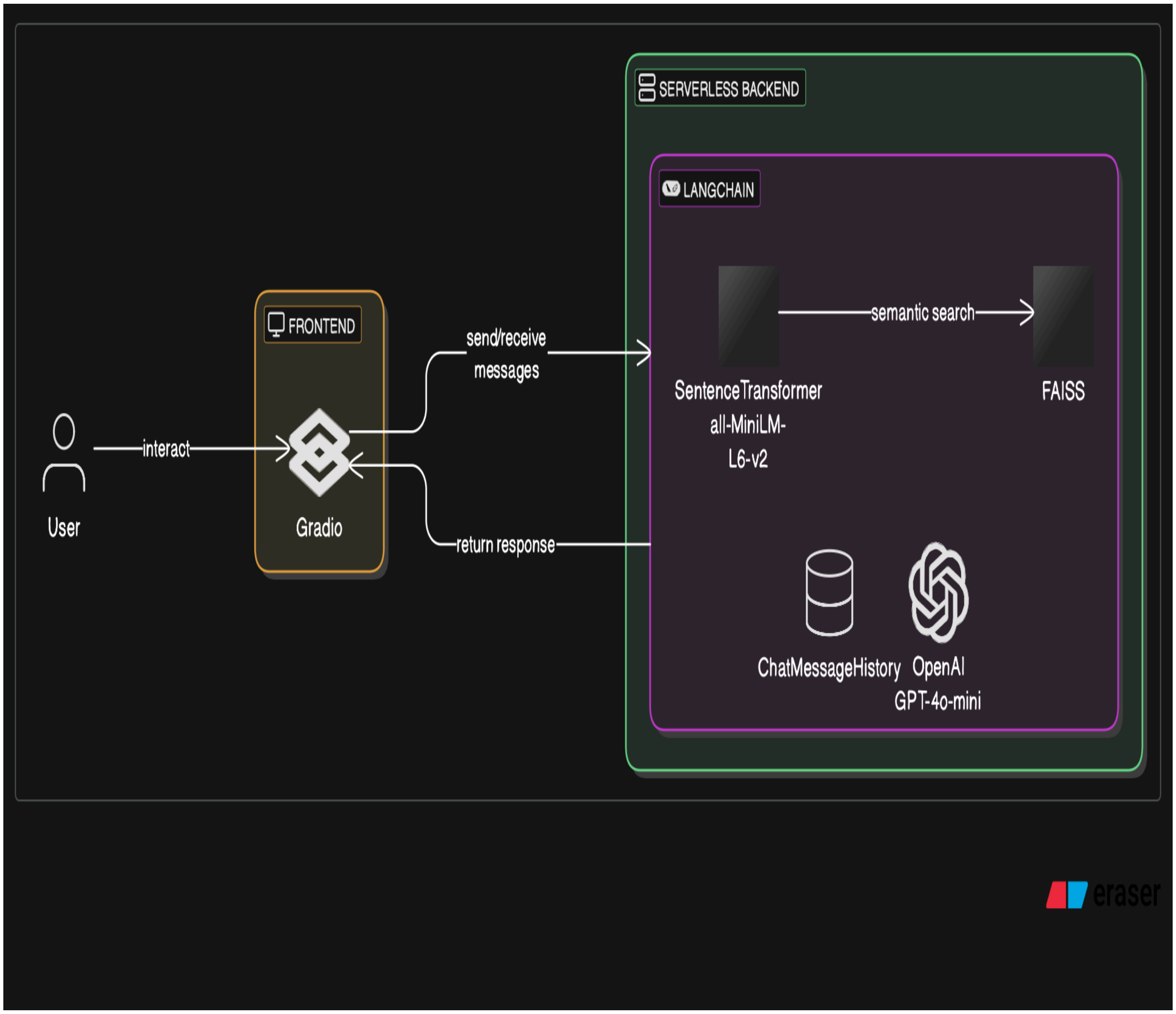
3. Technology Stack

The solution is developed using a highly modular and flexible stack:

- **Framework:** Python 3.12.9 and LangChain
    - Python provides the language runtime, while LangChain offers essential abstractions to create LLM-based applications with chaining, retrieval, and memory components.
- **LLM (Language Model):** OpenAI **`gpt-4o-mini`**
    - This is a lightweight variant of OpenAI's GPT-4 model, optimized for low latency and cost-effective usage without compromising significantly on reasoning quality.
- **Embeddings:** SentenceTransformer **`all-MiniLM-L6-v2`**
    - A 384-dimensional model from Hugging Face that offers high-quality sentence-level embeddings ideal for semantic similarity tasks. This model strikes a balance between accuracy, speed, and compute efficiency.
- **Vector Store:** FAISS (Facebook AI Similarity Search)

    - FAISS enables rapid approximate nearest-neighbor search on dense vectors, which is crucial for scalable and performant document retrieval.
- **Frontend:** Gradio
    - Provides an easy-to-use and responsive web interface for users to ask questions and view responses in real-time.
- **Session Store:** LangChain ChatMessageHistory
    - Manages individual user sessions, preserving conversational history that enhances context understanding in subsequent interactions.

4. Architecture



5. Key Modules

5.1 Document Loader

- Loads `.txt` files from the **`ancient_greece`** folder.
- Cleans and parses files into LangChain Document objects.
- **Note:** Text splitter is not used currently. Each `.txt` file is treated as a single document, which may affect retrieval granularity.
- **Ideal Strategy (Future Plan):** Integrate a recursive text splitter with chunking and overlap to improve retrieval precision and manage long documents efficiently.

5.2 FAISS Indexing

- Builds or loads FAISS index.
- Uses cosine similarity for vector retrieval.

5.3 Embedding Model: `all-MiniLM-L6-v2`

- 384-dimensional transformer model.
- Chosen for its balance of performance, speed, and dimensionality.
- Trained on NLI, STS, and paraphrasing tasks — ideal for semantic search.

5.4 LLM Model: `gpt-4o-mini`

- Offers concise, fluent generation.
- Handles historical context, reasoning, and rewriting of follow-up queries.

5.5 Contextual Retriever

- Reformulates follow-up questions using a history-aware chain.
- Ensures RAG retrieval context is aligned with the actual intent.

5.6 Conversational Memory

- Maintains separate sessions using unique session IDs.
- Built using **`RunnableWithMessageHistory`**.

5.7 Evaluation

- Metrics computed: Cosine Similarity, BLEU, ROUGE-1/2/L, METEOR, BERTScore.
- LLM-evaluated: Faithfulness, Relevance, Context-Faithfulness, Truthfulness.

5.8 Customer Satisfaction Evaluation

To assess real-world effectiveness, customer satisfaction should be evaluated through user feedback collected during A/B Testing & and afterwards for further improvement. Users can be asked to rate their experience based on the following metrics:

- **Accuracy of Answers:** Did the chatbot provide correct and helpful responses?
- **Ease of Use:** Was the interface intuitive and easy to interact with?
- **Context Awareness:** Did the chatbot understand follow-up questions and maintain conversation flow?
- **Overall Satisfaction:** General impression and likelihood to recommend the system to others.

6. Evaluation Snapshot

Sample QA :

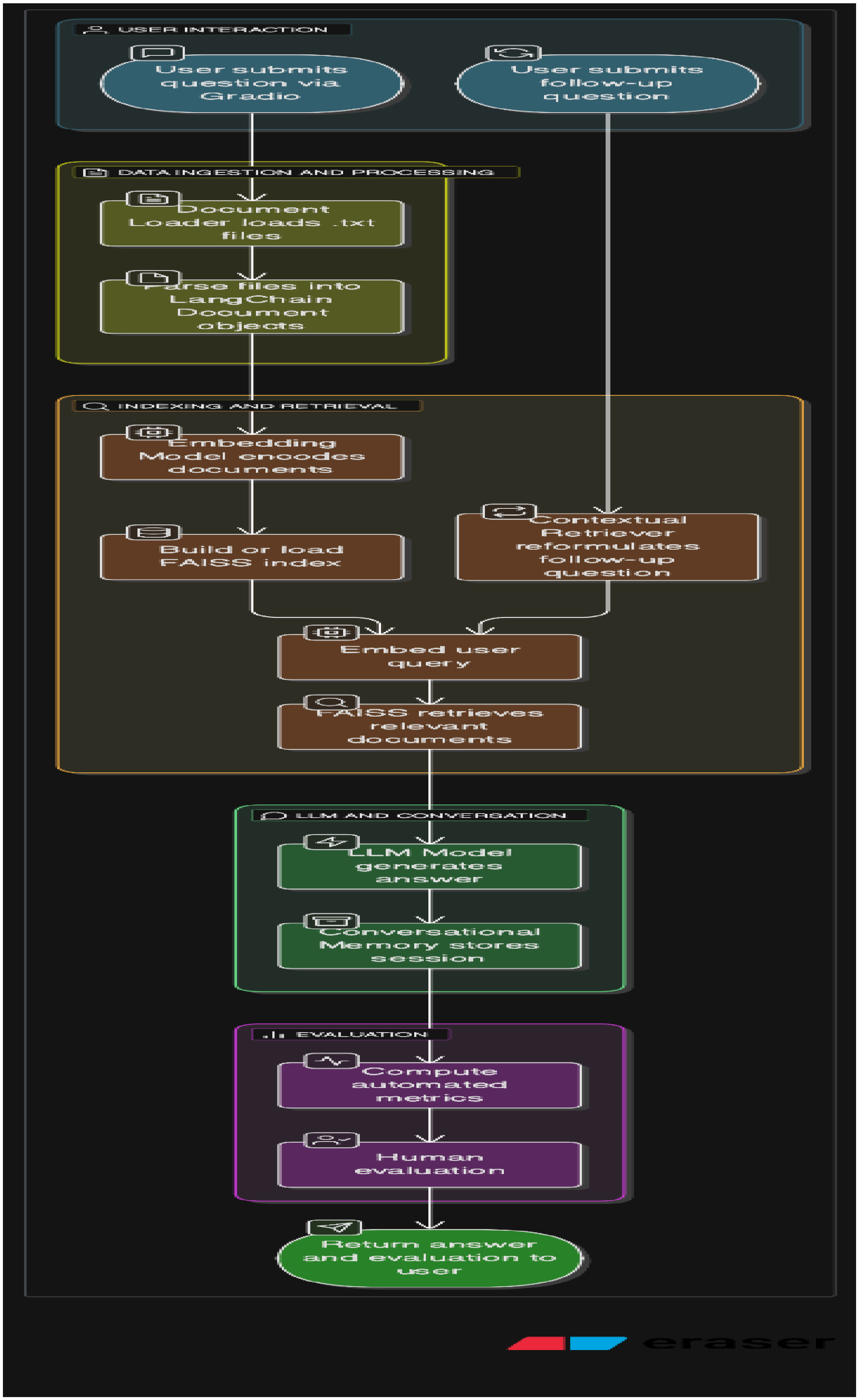
```
[
        {
           "question": "When did the story of Ancient Greece begin?",
             "expected_answer": "The story begins around 3000 BCE when a group of seafaring people known as the Minoans settled on the island of Crete.",
            "ground_truth_chunk": "The story begins around 3000 BCE when a group of seafaring people known as the Minoans settled on the island of Crete."
         }

]
```

Sample Result :

question,expected_answer,ground_truth_chunk,rag_answer,cosine_similarity,bleu_score,rouge-1,rouge-2,rouge-l,meteor,bertscore_f1,faithfulness,context_faithfulness,relevance,truthfulness

How does the internal Pipeline will look like

Below the the internal architecture of pipelines


USER INTERACTION
User submits question via Gradio
User submits follow-up question
DATA INGESTION AND PROCESSING
Document Loader loads .txt files
Parse files into LangChain Document objects
INDEXING AND RETRIEVAL
Embedding Model encodes documents
Build or load FAISS index
Contextual Retriever reformulates follow-up question
Embed user query
FAISS retrieves relevant documents
LLM AND CONVERSATION
LLM Model generates answer
Conversational Memory stores session
EVALUATION
Compute automated metrics
Human evaluation
Return answer and evaluation to user
eraser

7. Reasoning for Model Choices

Embedding Model: `all-MiniLM-L6-v2`

- **Dimensionality:** 384
- **Reasoning:**
  - Lightweight and fast inference, suitable for CPU and local deployment.
  - Strong semantic representation trained on natural language inference (NLI), semantic textual similarity (STS), and paraphrasing tasks, making it ideal for semantic search and retrieval.
  - Provides a balanced embedding dimension (384) that offers good accuracy without excessive computational cost.
  - Well-integrated with FAISS and LangChain, ensuring seamless vector store creation and retrieval.

Discussion on Embedding Alternatives

Why Not `text-ada-002` Embeddings?

- **`text-ada-002`** embeddings, offered by OpenAI, have higher dimensionality and tend to be more computationally expensive and costly.
- While powerful, they are often overkill for smaller or domain-specific datasets like Ancient Greece historical texts.
- **`all-MiniLM-L6-v2`** provides sufficiently rich semantic information at lower latency and cost.
- Additionally, SentenceTransformer models like **`all-MiniLM-L6-v2`** can be used offline, offering flexibility not available with API-based embeddings like **`text-ada-002`**.

Why Not BERTopic Embeddings?

- BERTopic embeddings are primarily designed for topic modeling and clustering rather than fine-grained semantic similarity search.
- The clustering mechanism in **BERTopic** can reduce retrieval precision in tasks that require detailed passage-level matching, such as question answering.
- The added computational complexity of BERTopic is unnecessary for a RAG system focused on accurate document retrieval.
- SentenceTransformer embeddings, such as **`all-MiniLM-L6-v2`**, better align with FAISS vector stores for efficient and accurate similarity-based retrieval.

LLM: `gpt-4o-mini`

- Offers cost-effective, fluent, and context-aware generation.
- Handles conversational history and complex historical context effectively.
- Balances inference speed with generation quality for an interactive QA system.

8. Deployment & Usage

Run locally using **`gradio`** or integrate with web frontends.

Supports:

- Dynamic question input
- Session-based memory
- Document-based QA with RAG

Want to check out Code and Deployment, Check it out below:

- HuggingFace : https://huggingface.co/spaces/Nishthaaa/langchain_openai_rag_chatbot
- Github : https://github.com/kukretinishtha/fab-rag

9. Future Work

- Add PDF and image OCR ingestion
- Integrate multilingual support
- Expand domain to other historical datasets
- Add text splitting for more granular retrieval using recursive chunking with overlap
- Extensive evaluation and report need to be generated as this project demonstrates how the chatbot should be in a production ready environment.

10. Contributors

- @kukretinishtha