Project Title:

AI Powered Log Parsing Tool

Objective:

To develop a robust AI-powered solution that utilizes LLMs to parse device logs, identify patterns and anomalies, and provide actionable insights to diagnose network issues.

Team 04

Team

Team Members

Dhanushree B	4VV21IS033
Karthik R	4VV21CS073
Khushi Kashinath	4VV21CS078
Rohan Vijay	4VV21CI044
Yashas J Kumar	4VV21CI059

HPE Mentor:

Mr. Sribharath Doopati
sribharath.doopati@hpe.com

College Mentor:

Neeti Shukla
Assistant Professor, Dept of ISE, VVCE

Background Works

- Collaborated with **APIs of Chat-GPT, Gemini Pro**, and other models, providing logs for inference.
- Explored diverse implementation methods such as **NLP, RLHF**, etc.
- Utilized the **Hugging Face platform** to experiment with various **embedding** and **generative models.**
- Studied the impact of **generic** and **domain-specific models.**
- Compared the efficiency of different LLMs, including **Llama 2, Llama 3, Mistral v0.2 and 0.3, Google Phi**, etc.
- Learned and implemented various fine-tuning techniques like **LoRA, QLoRA, and PEFT**.
- Implemented **Retrieval-Augmented Generation (RAG)** in multiple ways using different models and parameters.
- Used diverse environments such as **Google Colab, Kaggle, and Azure VM** to train and run the models.
- Ensured scalability and reliability of the solution for deployment in **large-scale network environments**.
- Explored Different types of **Retrievers, Pipelines and Chains** using in the RAG Application.
- Explored Available Cloud Options to Deploy the project.

Technical Approach

Technical Approach

Retrieval-Augmented Generation (RAG)

- **Retrieval-Augmented Generation (RAG)** is an advanced technique that combines information retrieval and generative modeling to enhance the performance of AI systems.

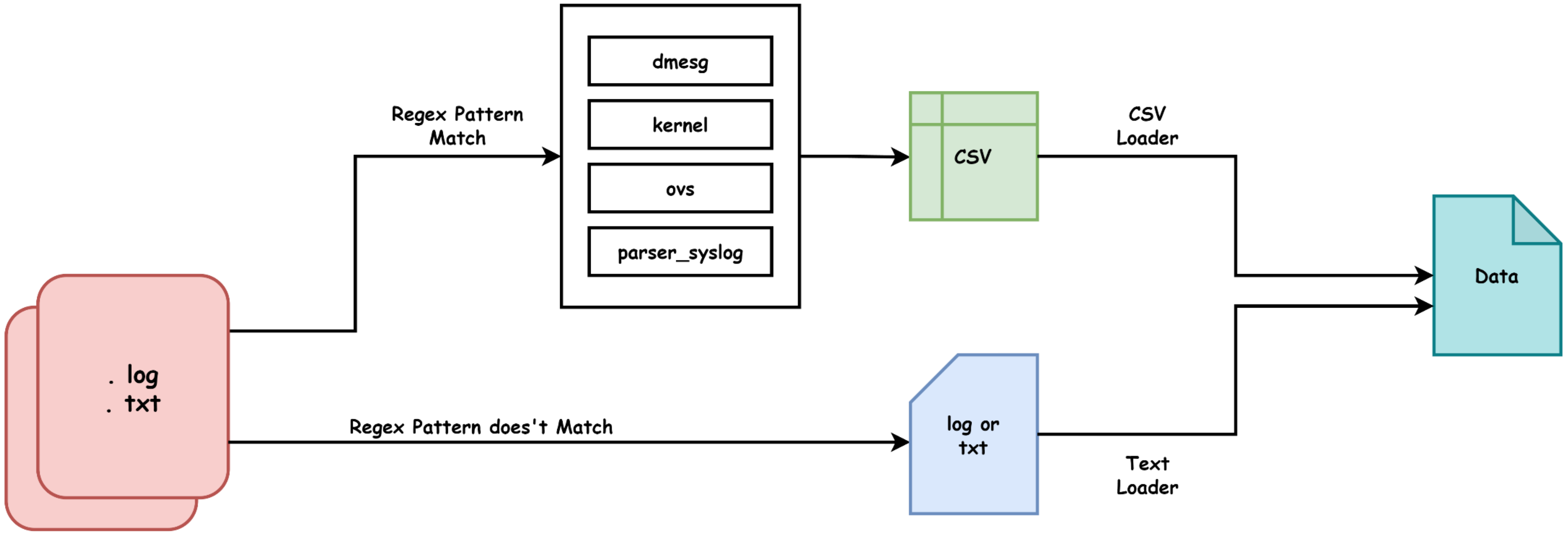
- RAG leverages the strengths of both approaches: it retrieves relevant documents or data points from a large corpus and then uses a generative model to process and produce contextually appropriate and coherent outputs based on the retrieved information.

- RAG is particularly effective in complex tasks such as question answering, content creation, and problem-solving scenarios where access to diverse and structured information is crucial.

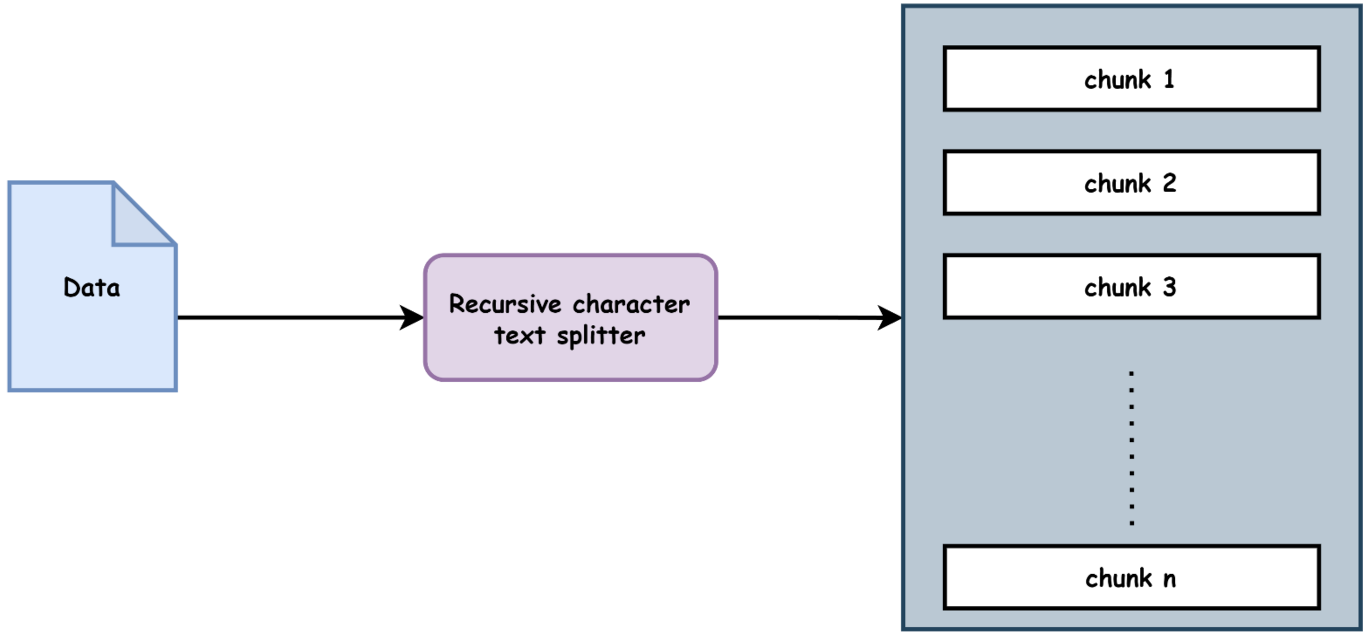
Implementation of the RAG

1. **Load the log or text file.**
2. **Split the data into a number of chunks.**
3. **Use Embedding-001, an embedding model, to convert the text chunks into vectors.**
4. **Store the vectors inside a FAISS vector database.**
5. **Configure a Llama3-70B-8192, a Llama 3 model, as the generative model.**
6. **Define the prompt template suitable for the corresponding model.**
7. **Define a document chain using the LLM and the prompt.**
8. **Configure and use a retriever to retrieve the top k (2) semantically similar chunks as context.**
9. **Configure a retrieval chain using the retriever and document chain to answer any query related to the provided logs using a Streamlit UI.**

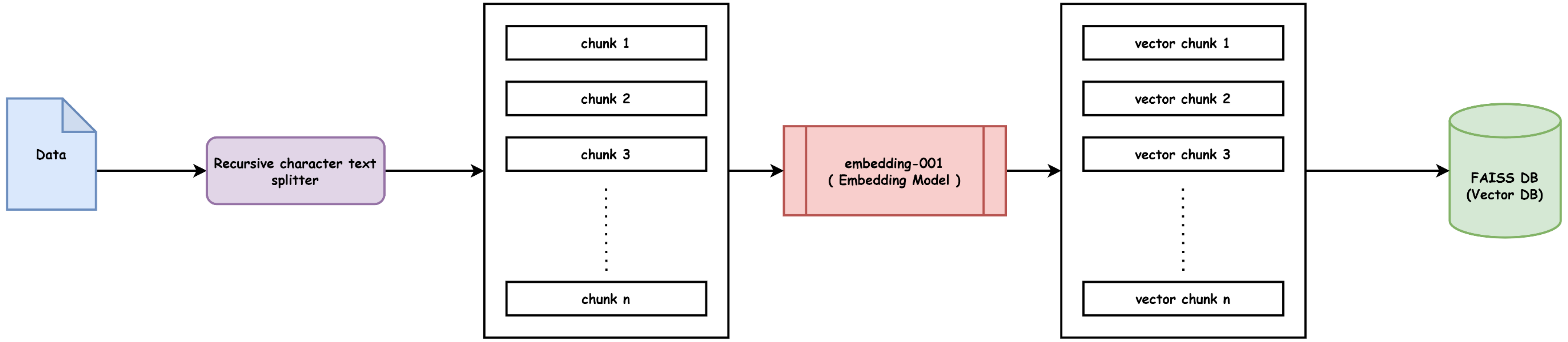
1. Load the log or text file.



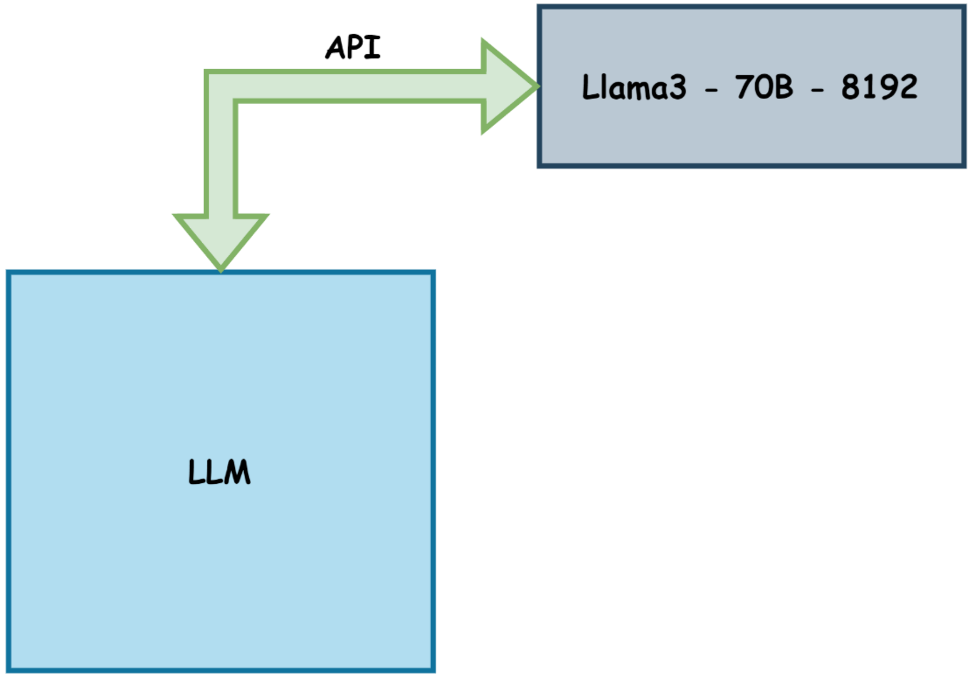
2. Split the data into a number of chunks.



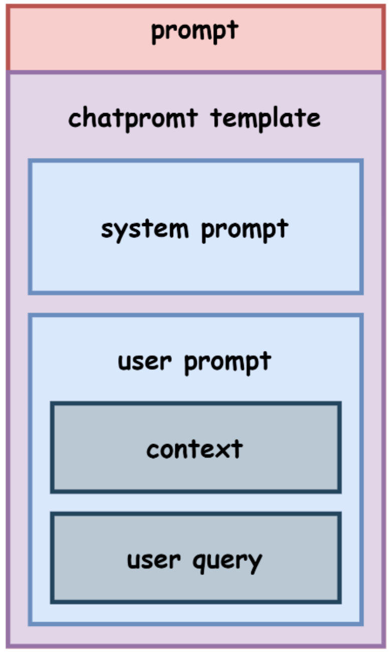
3. **Use Embedding-001, an embedding model, to convert the text chunks into vectors.**
4. **Store the vectors inside a FAISS vector database.**



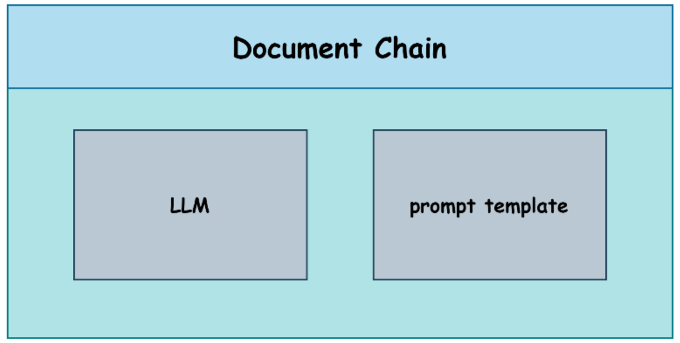
5. Configure a Llama3-70B-8192, a Llama 3 model, as the generative model.



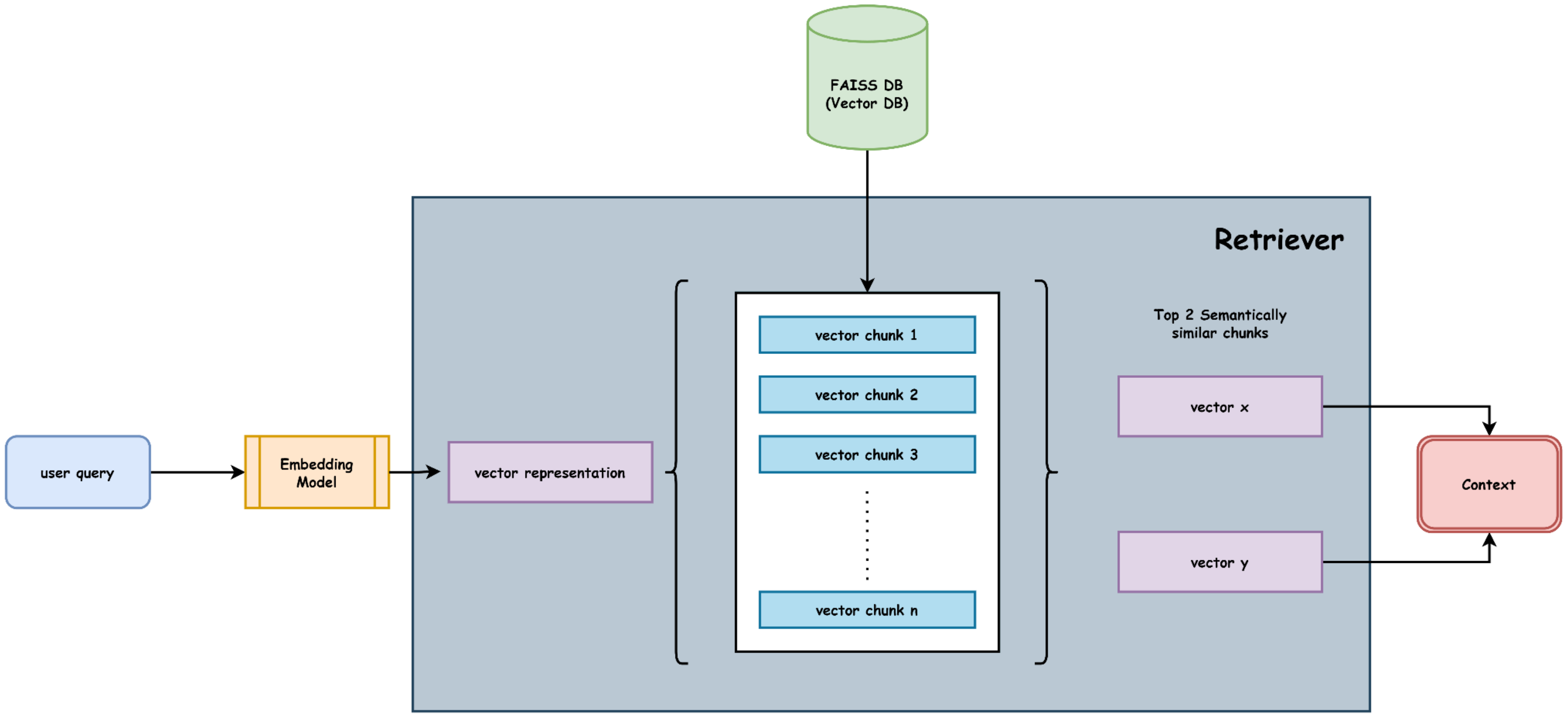
6. **Define the prompt template suitable for the corresponding model.**



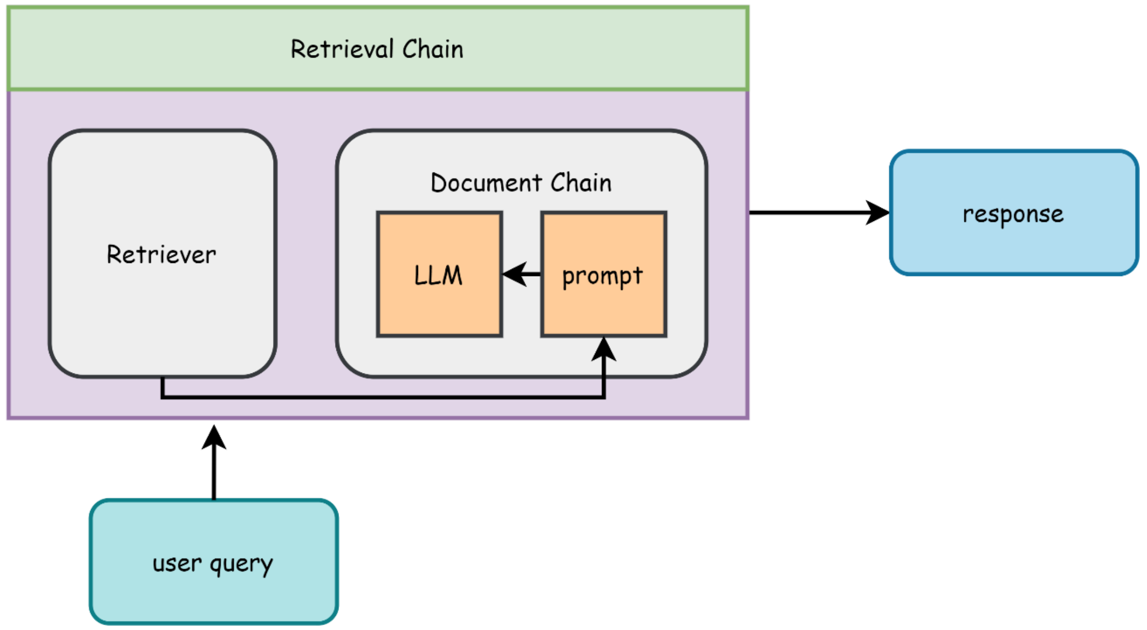
7. **Define a document chain using the LLM and the prompt.**

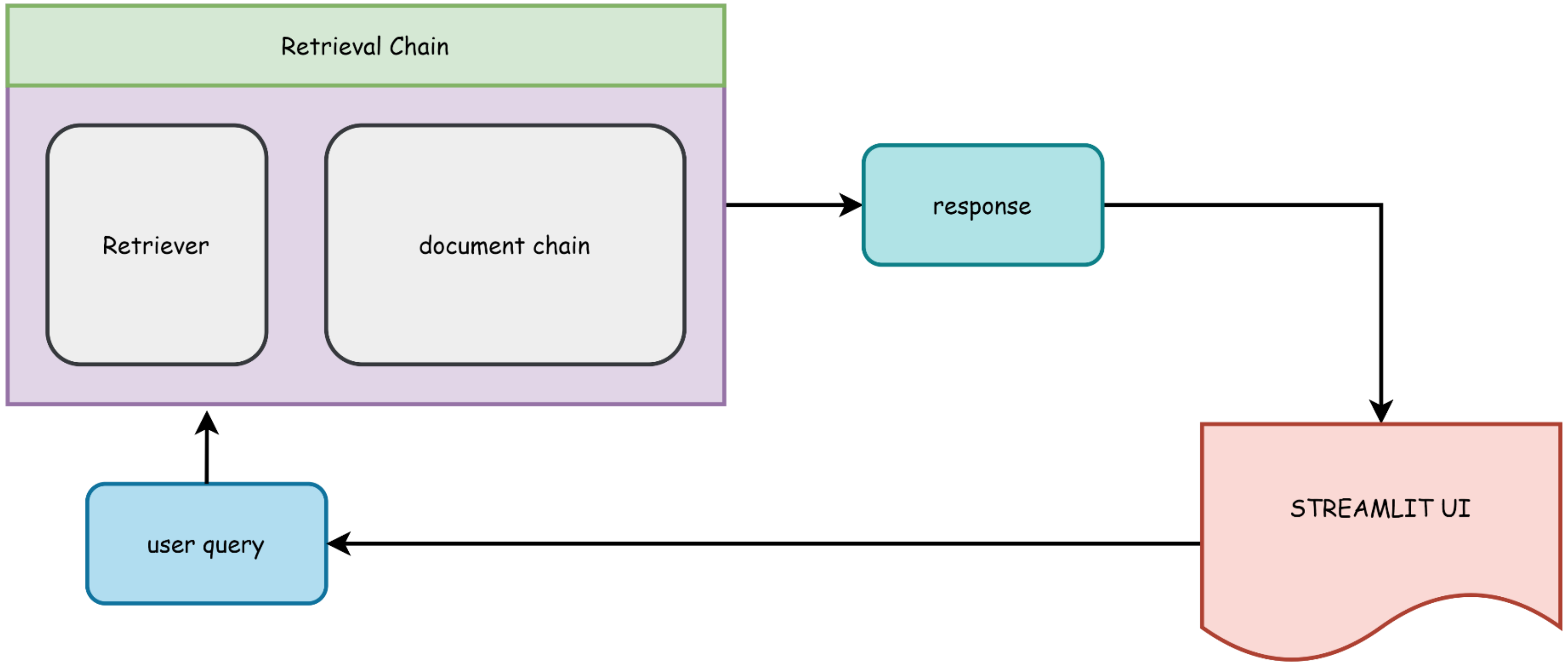


8. Configure and use a retriever to retrieve the top k (2) semantically similar chunks as context.



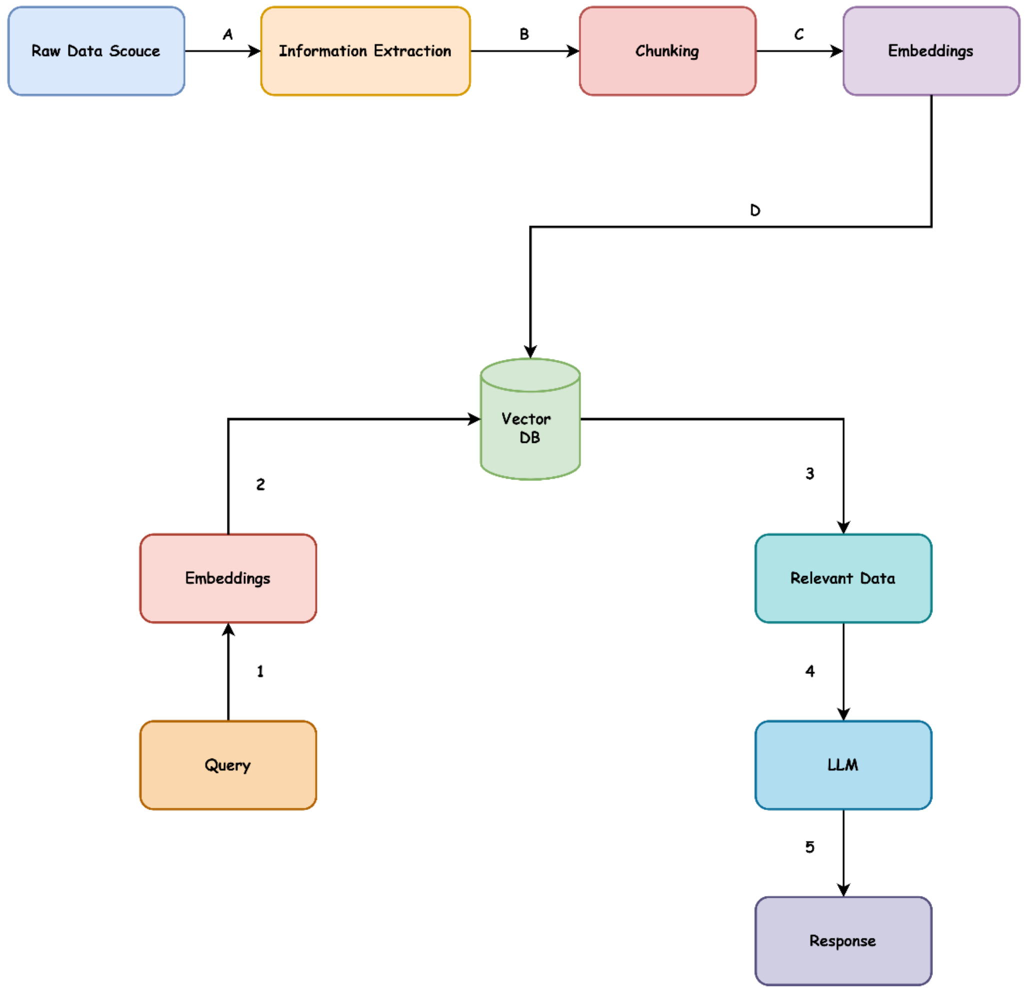
9. Configure a retrieval chain using the retriever and document chain to answer any query related to the provided logs using a Streamlit UI.




Retrieval Chain
Retriever
document chain
response
STREAMLIT UI
user query

Abstract Architecture

Abstract Architecture



Tech Stack

Tech Stack

Programming Language	Python
Framework	Langchain
Embedding Model	Embeddings-001
Generative Model	Llama3-70B-8192
Deployment	Streamlit

Limitations

Limitations

- **Scalability and Performance:** Embedding models and FAISS for vector storage and retrieval can be resource-intensive as data grows, leading to increased latency. Deploying the Llama3-70B-8192 model on GROQ Cloud may face computational resource and throughput constraints, affecting performance during peak usage.

- **Model and API Dependency:** Relying on the Llama3-70B-8192 model and GROQ Cloud API introduces vulnerabilities due to external dependencies. Service disruptions, API access policy changes, or cloud infrastructure limitations could impact functionality and reliability, posing challenges to maintaining consistent performance and long-term project sustainability.

- **Contextual Answer Limitation:** The Retrieval-Augmented Generation (RAG) approach depends on retrieving context from semantically similar chunks. If relevant information is spread across multiple chunks, the model's efficiency decreases, potentially leading to incomplete or less precise answers. Ensuring comprehensive context coverage is challenging, especially with fragmented information in large documents.

Conclusion

Conclusion

- In conclusion, this project leverages advanced natural language processing techniques and modern machine learning models to efficiently analyze and process log data. By converting logs into structured formats and utilizing embedding models for vectorization, the system facilitates effective retrieval and contextual understanding of log information.

- The integration of a powerful LLM, coupled with a responsive Streamlit UI, enables interactive querying and insightful analysis. However, the project faces challenges related to scalability and external dependencies, which must be addressed to ensure robust and sustainable performance.

- Overall, this project represents a significant step forward in automating and enhancing log analysis through cutting-edge AI technologies.

Thank you