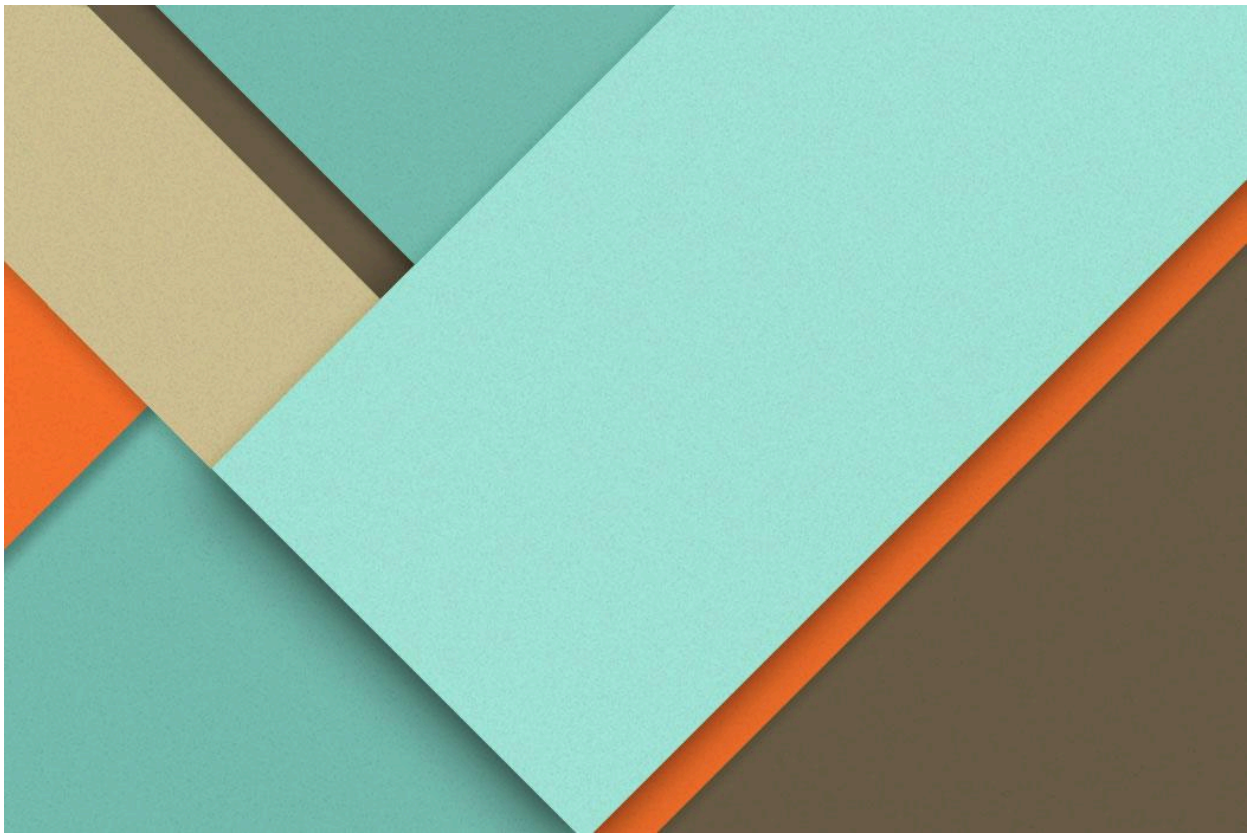

Prisma

(A Multifunctional AI Assistant)

Overview

Prisma is a multi-functional AI Assistant which utilizes multiple open-source large language models (LLMs) to provide answers to the user's query as per the requirement and personas (i.e. either in a technical or friendly way).

Block Diagram

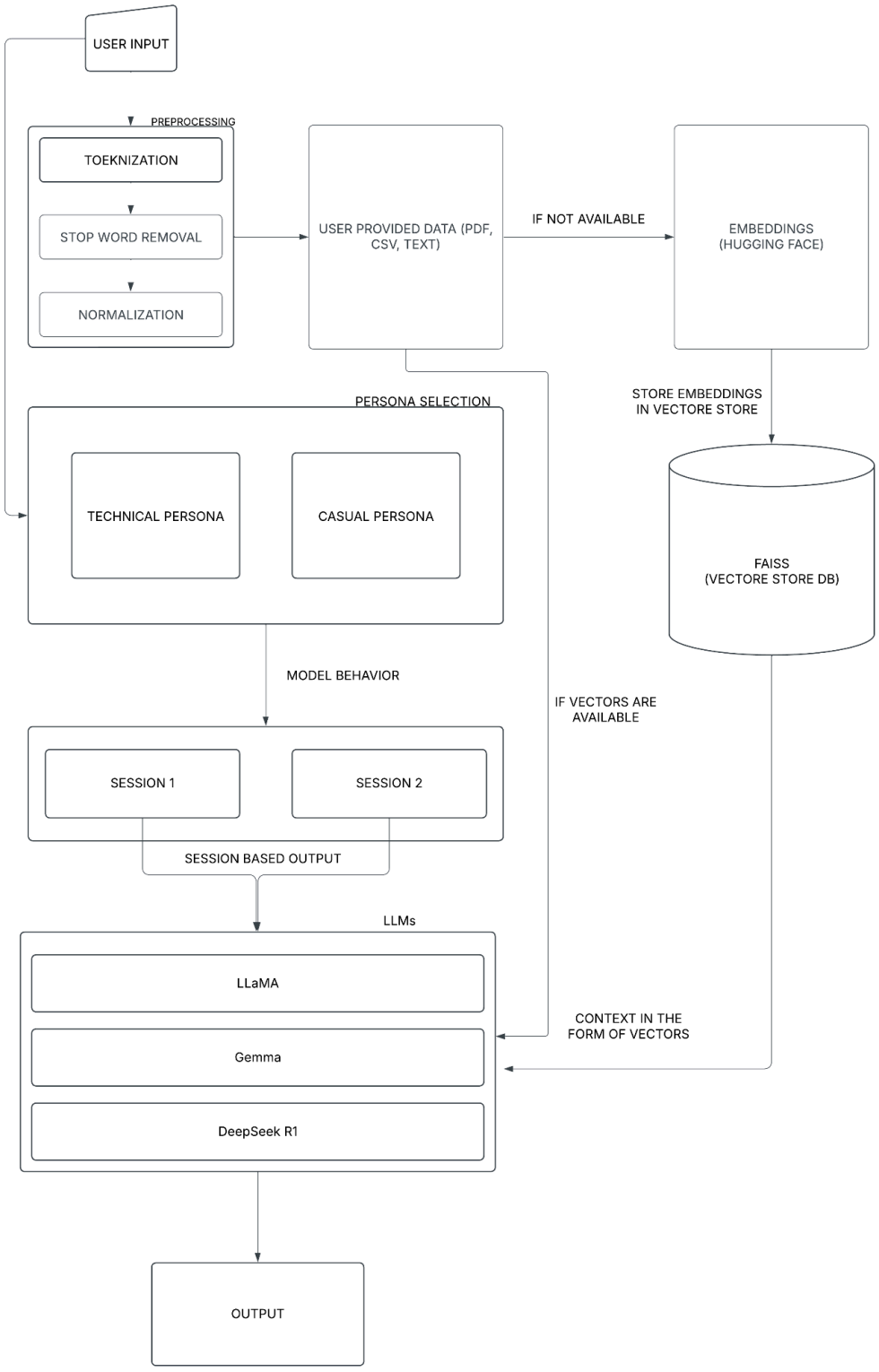

Figure 1: Block Diagram of the Prisma

The Figure 1 provides a general idea about the workflow which is being followed by the Prisma.

1. The user starts with a query and uploads documents, if any, the documents are then embedded and stored in the vector database which is FAISS (Facebook AI Similarity Search) in this case.
2. The user then needs to select a LLM model of his choice and select a persona (in which tone the user should get the response from the model), there are 3 models (LLaMA, Gemma and DeepSeek R1) and 2 Personas (Technical and Casual), here the LLaMA model and Technical Persona is selected by default.
3. Since the embedding and generation is done using API (Hugging Face and Groq) the user needs to enter their credentials. The user can also configure sessions, the sessions help models to retain the context of the past conversations.

Once done, the user can interact with the models.

Design Choices

1. The User Interface of the application is designed in Streamlit, a free and open-source framework which lets you design your LLM or any other DL, ML or Data Science related applications quickly.
2. I preferred Streamlit due to the speed of development (i.e the time taken to develop the UI), it is far less compared to other traditional ways.
3. Streamlit helps handling the user uploaded files, processing them without actually storing them on the machine.
4. The UI developed in Streamlit tends to follow the user-friendliness due to its components.
5. Prisma UI is divided into two sections
    a. On the left side, there is a sidebar where a user configures the behavior of the model.
    b. The same panel holds the component for helping users upload their own documents.
    c. Down in the panel, the user can add sessions.

Optimization

1. Since the documents uploaded by the users are embedded and the process of embedding takes time depending on the chunks of the documents provided, it is a good idea to store them in the session so when the user uploads the file again in some cases, the same documents are not embedded.

2. The requests to the API cannot be made before entering a query, preventing empty string requests.
3. Documents uploaded by the user are processed to ease the process of embedding.

Ethical Considerations

1. Bias and Harmful (Offensive) Comments are a threat to the response of the model. The default prompts for the model prevents these by telling the model not to respond to any such type of comments, but this is not the concrete solution.
2. The keys provided by the user are not stored internally, they are just referred during the API call

Potential Enhancements

1. Adding a robust mechanism to prevent offensive comments
    a. This can be done by implementing solutions like LLM Guard.
    b. The dual mechanism (Prompt and LLM Guard) will significantly reduce the problem.
2. The option can be given to use to manage their session, currently a user can add the session but cannot delete it, that option can be provided.
3. With switching of sessions, a new chat can be created, or the current chat can be cleaned.