Problem statement : Smart Lightweight Medical Query System (SLiMQ)

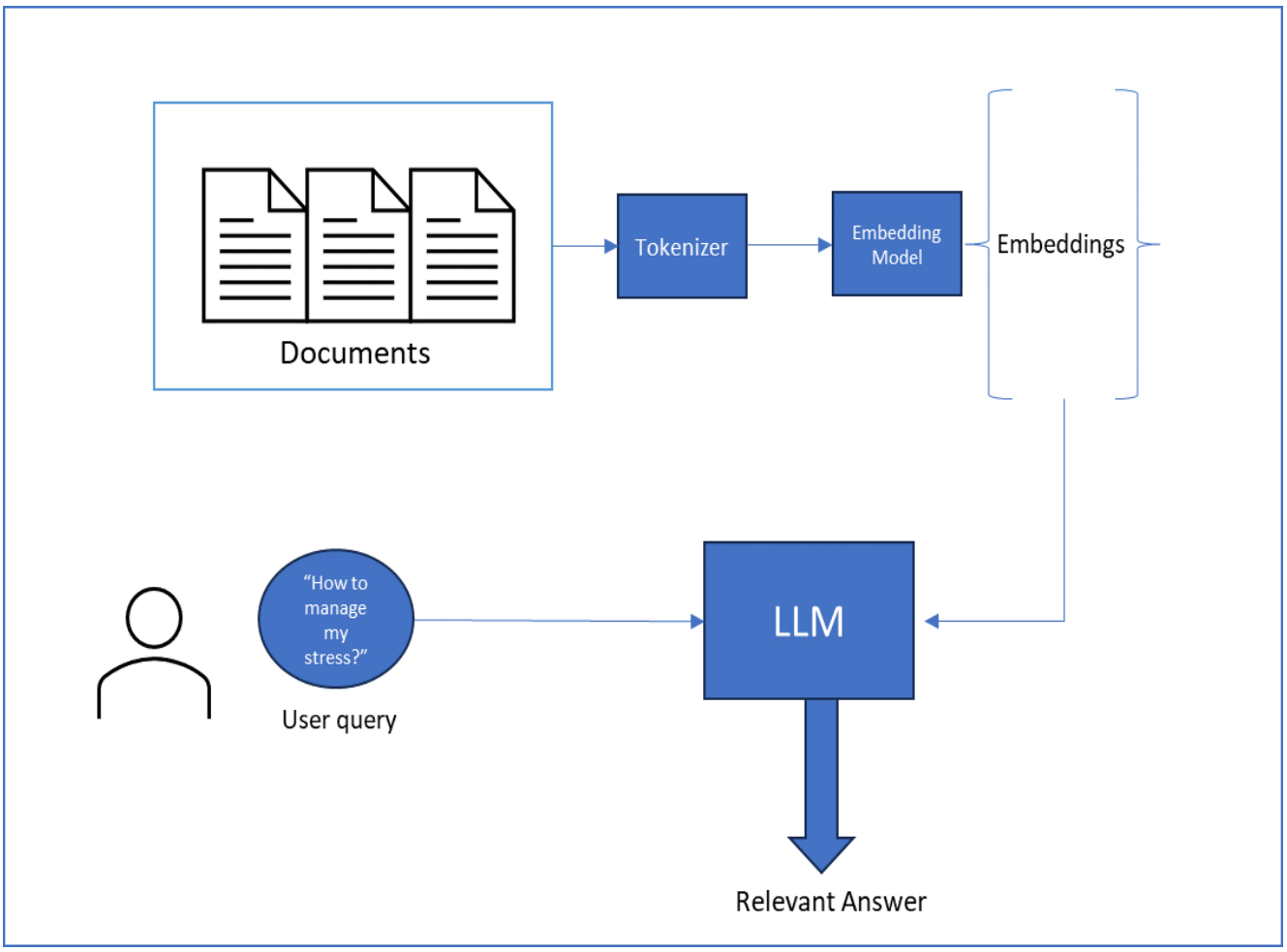
We are going to design a smart medical response system to aid doctors, the solution should be lightweight enough to run on off the shelf machines/edge devices.
A sample interaction may look like this:

User : What are tips for managing my bipolar disorder?
System : Along with seeing your doctor and therapist and taking your medicines, simple daily habits can make a difference. Start with these strategies.

- Pay attention to your sleep. This is especially important for people with bipolar disorder... (178 words truncated)
- Eat well. There's no specific diet... (29 words truncated)
- Focus on the basics: Favor fruits, vegetables, lean protein, and whole grains. And cut down on fat, salt, and sugar.
- Tame stress. (81 words truncated) You can also listen to music or spend time with positive people who are good company. (73 words truncated)
- Limit caffeine. It can keep you up at night and possibly affect your mood. (47 words truncated)
- Avoid alcohol and drugs. They can affect how your medications work.

Astute observer might ask, how is this system different from say ChatGPT?
Here, the key idea is that the query corresponds to unseen data. As LLMs have knowledge cut-off or not trained for specific use cases or on private data, the direct answer to query most likely won't be the right answer.

This requires using the external data source to generate the prompt which on adding to the query would help to generate the more correct response.



Deliverables: What we expect?

- A jupyter notebook/Standalone app containing the solution.
- This will be run with evaluation queries (kept private) to judge the robustness of the pipeline.

Dependencies: What you might need?

- A python/Jupyter based environment (totally up to you, you can create a pure C++ application if you want)
- Embedding model (Use standard embeddings model from huggingface or GitHub . eg SentencePiece, LlamaIndex)
- Vector database (Even numpy is fine)
- A pretrained LLMs for answer completion (Pretrained models are available from hugging face, we won't provide the model weights)
    - Llama-7b
    - Flan-T5
- A document pipeline eg: Langchain

- A dataset for testing/fine-tuning (will be provided during problem annoucement)

Challenges for students

- C-tier : How can you use and store the external data source? (Solution: Vector database)
- B- tier : Context window is limited, how they use the external data and use that information along with the query within the context window (Solution: external packages like LangChain)
- B-tier : Solution can make use of external endpoints, but locally running solutions will receive bonus points.
- A-tier : Use of quantized models: allows running of solution on non-GPU and resource quantized machines (look into llama.cpp or mlc-chat)
- S-tier : Run the entire thing from an Android application!

This problem statement may not require knowing the internal details on transformers or complexities in training them. Assuming most of the students would have an experience using these LLMs like ChatGPT, they can relate to the problem.