**Lab 11 Report – Large Language Models (LLMs)**

**Platform:** Gemini Flash 2.0 + Vercel + Hugging Face Space + MongoDB  
**Frontend:** <https://medical-chatbot-henna.vercel.app/>  
**Backend:** [https://huggingface.co/spaces/BinKhoaLe1812/Medical-Chatbot](https://huggingface.co/spaces/BinKhoaLe1812/Medical-Chatbot" \t "_new)

**GitHub:** <https://github.com/Lelekhoa1812/AutoGen-RAG-Medical-Chatbot/>

**1. Prompt Engineering**

**Techniques Applied:**

1. **Persona Prompting**

Explicitly define the chatbot as a **medical assistant**. This is done by embedding a role-based identity into the prompt, for example:

prompt = f"""

You are a medical chatbot, designed to answer medical questions.

Please format your answer using markdown.

\*\*Bold for titles\*\*, \*italic for emphasis\*, and clear headings.

\*\*Medical knowledge (trained with 256,916 data entries):\*\*

{knowledge\_base}

\*\*Question:\*\* {user\_query}

\*\*Language Required:\*\* {lang}

"""

completion = gemini\_flash\_completion(prompt, model=self.model\_name,temperature=0.7)

This persona prompt helps the LLM maintain a consistent and medically-aligned tone throughout all answers.

**b) RAG-based Context Injection**  
The user query is **augmented with top-k retrieved passages** from a custom FAISS index of medical data stored in MongoDB. This enhances factual accuracy and prevents hallucination.

**c) Language Selector in Prompt**  
The prompt also includes a lang parameter, allowing for multilingual support.

**2. Advanced Prompt Engineering**

**Techniques Applied:**

**a) K-shot / Semantic K-shot**

While traditional k-shot was not used in the classic format, semantic equivalents were applied by feeding **similar semantic responses** via FAISS retrieval — emulating a semantic few-shot paradigm from relevant training QA pairs.

**b) Chain of Thought Prompting**

By using Gemini’s multi-line markdown prompts and letting the model “explain answers” rather than give one-line replies, we emulate chain-of-thought logic to produce **stepwise reasoning** in medical answers.

**c) Chaining**

Prompt chaining was accomplished via the combination of:

* Retrieval of relevant answers (retrieve\_medical\_info)
* Constructed markdown-formatted prompt
* LLM generation using Gemini Flash 2.0

This modular chaining design is implemented in the chatbot class:

class RAGMedicalChatbot:

def \_\_init\_\_(self, model\_name, retrieve\_function):

self.model\_name = model\_name

self.retrieve = retrieve\_function

def chat(self, user\_query, lang="EN"):

retrieved\_info = self.retrieve(user\_query)

knowledge\_base = "\n".join(retrieved\_info)

prompt = ...

completion = gemini\_flash\_completion(prompt, model=self.model\_name, temperature=0.7)

return completion.strip()

**d) Retrieval-Augmented Generation (RAG)**

* Semantic search over a custom **FAISS index**
* Embedding via **quantized SentenceTransformer**
* Combined into Gemini prompt for final generation

This hybrid pipeline bridges traditional IR and generative AI, ensuring the chatbot outputs are grounded in known facts.

**3. Architectures**

**Custom-Built LLM QA System**

Developed a **complete medical chatbot pipeline** using:

* **Gemini Flash 2.0 API** (Google's LLM)
* **FAISS + SentenceTransformer** for retrieval
* **MongoDB** for storage and GridFS for FAISS model
* **FastAPI** backend with a deployed Docker container
* **Frontend via Vercel** for web-based chat interaction

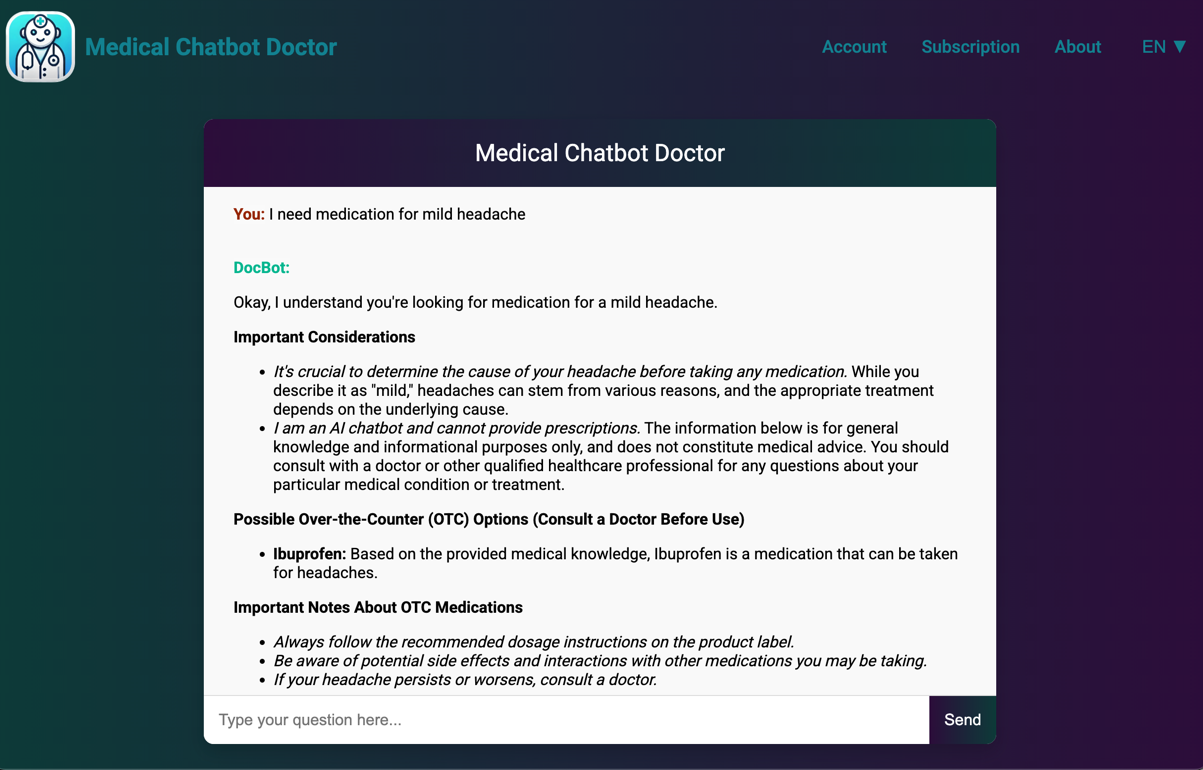
**Key Components:**

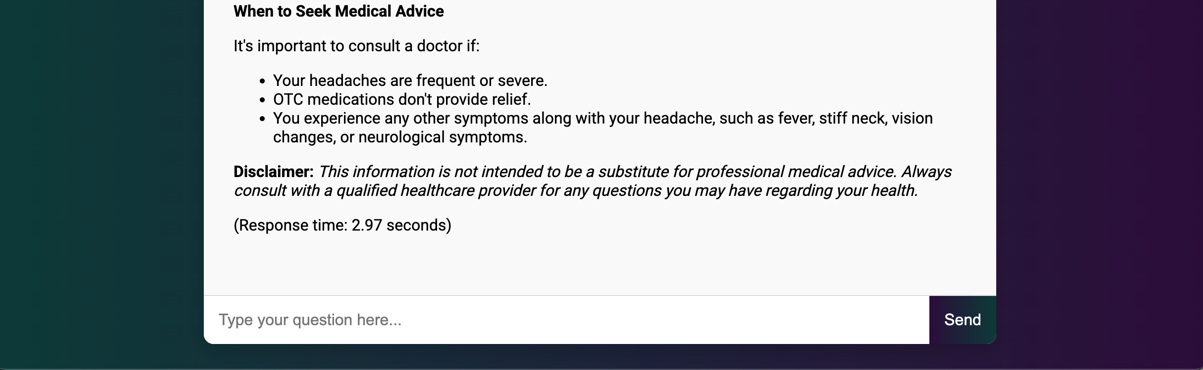
* app.py: Core server that loads embeddings, handles user prompts, and serves responses via /chat endpoint.
* Dockerfile: Enables reproducible cloud deployment using Python 3.11 base and preloading the embedding model (For HF Space Docker image build).
* retrieve\_medical\_info(): RAG logic using FAISS.
* gemini\_flash\_completion(): Calls Gemini LLM with retrieved context.

**Summary**

|  |  |
| --- | --- |
| **Component** | **Description** |
| Prompt Engineering | Persona, markdown formatting, role-based logic |
| Advanced Prompting | Chaining, RAG, semantic retrieval, markdown chain-of-thought |
| LLM Used | Gemini Flash 2.0 (free-tier) via google.genai |
| Backend Stack | FastAPI, MongoDB, FAISS, SentenceTransformer |
| Deployment | Hugging Face Space + Dockerfile + Vercel frontend |

**QA Screenshots**





We can try with Chinese language too:



