

TRƯỜNG ĐẠI HỌC MỎ - ĐỊA CHẤT

**BÁO CÁO TỔNG KẾT
ĐỀ TÀI NCKH SINH VIÊN**

**XÂY DỰNG CHATBOT
HỖ TRỢ HỌC TẬP CHO SINH VIÊN CNTT**

Hà Nội, 05/2026

TRƯỜNG ĐẠI HỌC MỎ - ĐỊA CHẤT

**BÁO CÁO TỔNG KẾT
ĐỀ TÀI NCKH SINH VIÊN**

**XÂY DỰNG CHATBOT
HỖ TRỢ HỌC TẬP CHO SINH VIÊN CNTT**

- Trưởng nhóm nghiên cứu: Nguyễn Trọng Thắng, Công Nghệ Thông Tin K69C
- Thành viên tham gia thực hiện:

Trịnh Hoài Đức, Công Nghệ Thông Tin K69B

Nguyễn Thành Công, Công Nghệ Thông Tin K69A

Phạm Trần Duy Lợi, Công Nghệ Thông Tin K69B

Hoàng Mạnh Duy, Công Nghệ Thông Tin K69C

Người hướng dẫn: ThS. Bùi Thị Vân Anh

Hà Nội, 05/2026

LỜI CẢM ƠN

Lời đầu tiên, nhóm em xin gửi lời cảm ơn chân thành và sâu sắc nhất tới Ban Giám hiệu cùng toàn thể quý Thầy, Cô giáo trường Đại học Mở - Địa Chất, những người đã tận tâm truyền đạt tri thức và dìu dắt tụi em trong suốt quá trình học tập và rèn luyện tại mái trường thân yêu này tới thời điểm hiện tại.

Đặc biệt, tụi em xin bày tỏ lòng biết ơn vô cùng sâu sắc tới Cô Bùi Thị Vân Anh người đã dành rất nhiều thời gian và tâm huyết trực tiếp hướng dẫn, chỉ bảo tụi em trong suốt quá trình thực hiện đề tài nghiên cứu khoa học. Những định hướng quý báu, những nhận xét khắt khe nhưng đầy tính xây dựng của Cô không chỉ giúp tụi em hoàn thiện về mặt kỹ thuật cho đề tài "Chatbot hỗ trợ học tập cho sinh viên CNTT" mà còn giúp rèn luyện tư duy nghiên cứu độc lập và phong cách làm việc khoa học.

Chúng em cũng xin gửi lời cảm ơn tới gia đình, bạn bè và những người thân yêu đã luôn đồng hành, cổ vũ và tạo mọi điều kiện tốt nhất về cả vật chất lẫn tinh thần để tụi em có thể yên tâm học tập và hoàn thành bài nghiên cứu này một cách trọn vẹn nhất.

Bài nghiên cứu khoa học này là kết quả của quá trình nỗ lực không ngừng nghỉ, tuy nhiên do hạn chế về mặt thời gian cũng như kiến thức chuyên môn, chắc chắn báo cáo không tránh khỏi những thiếu sót nhất định. Chúng em rất mong nhận được những ý kiến đóng góp, chỉ bảo quý báu từ Hội đồng khoa học và các Thầy, Cô giáo để dự án của chúng em ngày càng hoàn thiện hơn và có tính ứng dụng cao hơn trong thực tế.

Cuối cùng, chúng em xin kính chúc quý Thầy, Cô giáo luôn dồi dào sức khỏe, hạnh phúc và gặt hái được nhiều thành công hơn nữa trong sự nghiệp cao quý của mình.

Em xin chân thành cảm ơn!

MỤC LỤC

LỜI CẢM ƠN.....	1
CHƯƠNG 1. TỔNG QUAN	7
1.1. Lý do chọn đề tài	7
1.1.1. Nhu cầu truy xuất thông tin trong học tập CNTT	7
1.1.2. Giới hạn của các mô hình ngôn ngữ thông thường.....	7
1.1.3. Lựa chọn kỹ thuật RAG (Retrieval-Augmented Generation).....	7
1.1.4. Lý do lựa chọn đề tài.....	8
1.2. Mục tiêu và phạm vi của đề tài	8
1.3. Tầm quan trọng của đề tài.....	8
CHƯƠNG 2. KIẾN THỨC CƠ SỞ VÀ CÔNG NGHỆ SỬ DỤNG	10
2.1. Mô hình Ngôn ngữ Lớn (Large Language Model – LLM)	10
2.1.1. Khái niệm và nguyên lý hoạt động	10
2.1.2. Google Gemini – LLM được sử dụng trong dự án	10
2.2. Chatbot và Hệ thống Hỏi Đáp Thông Minh	11
2.3. Kỹ thuật RAG (Retrieval-Augmented Generation).....	11
2.3.1. Định nghĩa và kiến trúc tổng quan	11
2.3.2. Tầm quan trọng của RAG trong bài toán hỗ trợ học tập.....	12
2.4. Embedding và Vector Database	12
2.4.1. Embedding – Biểu diễn văn bản thành vector số học	12
2.4.2. FAISS – Cơ sở dữ liệu vector	12
2.5. LangChain – Framework Xây dựng Ứng dụng LLM.....	13
2.6. FastAPI – Backend Framework.....	14
2.7. Giao diện Người dùng – HTML, CSS và JavaScript	14
2.8. Tổng hợp và So sánh với Dự án Tham khảo	14
CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG.....	16
3.1. Phân tích hệ thống.....	16
3.1.1. Các tác nhân	18
3.1.2. Sơ đồ Usecase tổng quát	18
3.1.3. Các Usecase chi tiết	18
3.2. Phân Tích Các Nguồn Dữ Liệu.....	21

3.3. Mô Tả Quá Trình Chuẩn Bị và Tiền Xử Lý Dữ Liệu:	22
3.3.1. Giai đoạn chuẩn bị dữ liệu	22
3.3.2. Giai đoạn trung gian: Tối ưu hóa cấu trúc dữ liệu	22
3.3.3. Giai đoạn Vector hóa và Nhúng ngữ nghĩa	23
3.3.4. Giai đoạn tạo lập cơ sở dữ liệu Vector	23
3.3.5. Giai đoạn kết quả và đóng gói (Result & Persistence)	24
3.3.6 Kết quả	24
CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG	25
4.1. Cấu trúc dự án	25
4.2. Thiết kế giao diện và chức năng của Chatbot	26
4.3. Tối ưu hóa và cải thiện hiệu suất Chatbot	30
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	31
I. Kết quả đạt được và những hạn chế	31
II. Đánh giá hiệu suất và kết quả:	33
III. Hướng phát triển để cải thiện Chatbot trong tương lai	34

DANH MỤC HÌNH ẢNH

Hình 3.1 Sơ đồ workflow của chatbot.....	16
Hình 3.2 Sơ đồ usecase tổng quát.....	18
Hình 3.3 Biểu đồ usecase nhập câu hỏi.....	19
Hình 3.4 Sơ đồ tuần tự nhập câu hỏi	19
Hình 3.5 Biểu đồ usecase chức năng huấn luyện bằng file PDF.....	20
Hình 3.6 Sơ đồ tuần tự chức năng huấn luyện bằng file PDF	20
Hình 3.8 Sơ đồ tuần tự chức năng chatbot đưa ra câu trả lời.....	21
Hình 3.9 Tạo kho để lưu trữ Database.....	22
Hình 3.10 Chunking	23
Hình 3.11 Embedding.....	23
Hình 3.12 FAISS.....	24
Hình 3.13 Lưu dữ liệu vào ổ cứng.....	24
Hình 4.1 Tổng qua cấu trúc dự án	25
Hình 4.2 Giao diện chatbot.....	27
Hình 4.3 Logo và khu vực hướng dẫn.....	28
Hình 4.4 Khu vực chat.....	28
Hình 4.5 Ô nhập chat.....	29
Hình cuộc trò chuyện mẫu.....	33

DANH MỤC BẢNG

Bảng 2.1 So sánh FAISS với Chroma	13
Bảng 2.2 sự khác biệt về công nghệ giữa dự án RAG Chatbot và dự án tham khảo RAG-GEMMA	15

LỜI MỞ ĐẦU

Trí tuệ nhân tạo và công nghệ thông tin đang ngày càng phát triển và đóng vai trò quan trọng trong nhiều lĩnh vực của cuộc sống. Trong xu thế đó, các hệ thống ChatBot đã trở thành một công cụ hữu ích, giúp tự động hóa và nâng cao hiệu quả trong giao tiếp và hỗ trợ người dùng. Dự án NCKH chúng em với đề tài "**Chatbot hỗ trợ học tập cho sinh viên CNTT**" được thực hiện nhằm nghiên cứu và phát triển một hệ thống ChatBot thông minh, hỗ trợ hướng dẫn người dùng có thể trò chuyện với Chatbot bằng dữ liệu riêng của mình.

Với sự hỗ trợ của công nghệ các mô hình ngôn ngữ lớn (LLM) và kỹ thuật truy vấn RAG, kết hợp cả hai cùng với LangChain, ChatBot RAG không chỉ đơn thuần là một công cụ trả lời câu hỏi mà còn có khả năng học hỏi từ các dữ liệu riêng và người dùng tải lên để cải thiện chất lượng truy vấn. Trong suốt quá trình thực hiện dự án NCKH, dù đã đối mặt với nhiều thử thách, từ việc phân tích và xử lý ngôn ngữ tự nhiên, thiết kế giao diện người dùng, đến việc tích hợp các thuật RAG trong học máy để nâng cao khả năng tương tác của ChatBot nhưng cuối cùng cả nhóm vẫn hoàn thành được kết quả như mong đợi.

Dự án NCKH này không chỉ là kết quả của quá trình học tập và nghiên cứu, mà còn là minh chứng cho sự nỗ lực và đam mê của cả nhóm trong lĩnh vực công nghệ thông tin. Chúng em hy vọng rằng sản phẩm này sẽ góp phần vào việc phát triển các ứng dụng ChatBot thông minh, phục vụ tốt hơn nhu cầu của người dùng và mang lại những giá trị thiết thực cho xã hội.

CHƯƠNG 1. TỔNG QUAN

1.1. Lý do chọn đề tài

1.1.1. *Nhu cầu truy xuất thông tin trong học tập CNTT*

Trong bối cảnh giáo dục đại học hiện nay, đặc biệt trong lĩnh vực Công nghệ thông tin, sinh viên phải tiếp cận một khối lượng kiến thức khổng lồ từ giáo trình, tài liệu tham khảo, bài giảng và các nguồn học liệu số. Tuy nhiên, việc tìm kiếm và tra cứu thông tin một cách chính xác, nhanh chóng từ các tài liệu này luôn là một thách thức. Sinh viên thường mất nhiều thời gian để sàng lọc qua các tài liệu dài hàng trăm trang chỉ để tìm một khái niệm hay một đoạn giải thích cụ thể.

Thực tế tại Trường Đại học Mở - Địa chất (HUMG), sinh viên khoa CNTT thường xuyên gặp khó khăn trong việc tự học và tra cứu tài liệu ngoài giờ lên lớp. Nhu cầu có một trợ lý học tập thông minh, có khả năng trả lời câu hỏi dựa trên nội dung tài liệu cụ thể của môn học, là một nhu cầu có thực và cấp thiết.

1.1.2. *Giới hạn của các mô hình ngôn ngữ thông thường*

Các mô hình ngôn ngữ lớn (Large Language Models – LLM) như GPT hay Gemini hiện đang rất mạnh mẽ trong việc tạo sinh văn bản và trả lời câu hỏi tổng quát. Tuy nhiên, chúng có một hạn chế căn bản: kiến thức của mô hình bị đóng băng tại thời điểm huấn luyện và không có khả năng truy cập vào các tài liệu riêng tư, cục bộ của người dùng. Điều này có nghĩa là khi sinh viên hỏi về nội dung một giáo trình cụ thể hay slide bài giảng của giảng viên, LLM sẽ không thể trả lời dựa trên tài liệu đó.

1.1.3. *Lựa chọn kỹ thuật RAG (Retrieval-Augmented Generation)*

Kỹ thuật RAG (Retrieval-Augmented Generation) ra đời như một giải pháp kết hợp giữa hai thế mạnh: khả năng tìm kiếm và truy xuất thông tin từ nguồn dữ liệu cục bộ, và khả năng sinh ngôn ngữ tự nhiên linh hoạt của LLM. Thay vì chỉ dựa vào kiến thức được huấn luyện sẵn, RAG cho phép mô hình "tra cứu" tài liệu thực tế của người dùng trước khi tạo ra câu trả lời. Điều này giúp câu trả lời chính xác hơn, có căn cứ hơn, và phù hợp với ngữ cảnh học tập cụ thể.

Các ưu điểm nổi bật của RAG so với chatbot thông thường bao gồm:

- Trả lời dựa trực tiếp trên nội dung tài liệu mà người dùng cung cấp, không hallucinate.
- Có thể cập nhật kiến thức theo thời gian thực bằng cách thêm tài liệu mới mà không cần huấn luyện lại mô hình.
- Tính minh bạch cao – có thể trích dẫn nguồn tài liệu cụ thể cho mỗi câu trả lời.
- Phù hợp cho bài toán hỗ trợ học tập chuyên ngành với tài liệu cục bộ.

1.1.4. Lý do lựa chọn đề tài

Xuất phát từ những trở ngại thực tế trong quá trình học tập tại khoa CNTT, chúng em lựa chọn đề tài "Xây dựng hệ thống Chatbot trợ lý học tập sử dụng kỹ thuật RAG và mô hình ngôn ngữ lớn" với mong muốn tạo ra một công cụ hỗ trợ thiết thực cho sinh viên. Hệ thống được xây dựng dựa trên kiến trúc RAG, tích hợp với mô hình Google Gemini thông qua API, sử dụng FastAPI làm backend và giao diện web HTML/CSS/JavaScript thân thiện với người dùng.

1.2. Mục tiêu và phạm vi của đề tài

Mục tiêu chính của đề tài là xây dựng một hệ thống chatbot có khả năng:

1. Nhận tài liệu PDF đầu vào từ người dùng (giáo trình, bài giảng, tài liệu học tập).
2. Xử lý và lưu trữ nội dung tài liệu dưới dạng vector trong cơ sở dữ liệu vector (FAISS).
3. Trả lời câu hỏi của sinh viên dựa trên nội dung tài liệu đã được tải lên, kết hợp với khả năng sinh ngôn ngữ của mô hình Gemini.
4. Cung cấp giao diện web thân thiện với chức năng tải lên tài liệu và hỏi đáp linh hoạt.
5. Duy trì lịch sử hội thoại để đảm bảo tính nhất quán trong quá trình trao đổi nhiều lượt.

Phạm vi: Đề tài tập trung vào hỗ trợ sinh viên khoa CNTT trong việc tra cứu tài liệu học tập định dạng PDF. Hệ thống không nhằm thay thế giảng viên mà đóng vai trò là trợ lý học tập bổ trợ. Ngôn ngữ chính được hỗ trợ là tiếng Việt và tiếng Anh.

1.3. Tầm quan trọng của đề tài

Trong xu thế chuyển đổi số trong giáo dục đang diễn ra mạnh mẽ tại Việt Nam, các công cụ hỗ trợ học tập thông minh ngày càng đóng vai trò quan trọng. Hệ thống

RAG Chatbot được đề xuất trong đề tài này mang lại những giá trị thiết thực sau:

- Tiết kiệm thời gian học tập: Thay vì phải đọc toàn bộ tài liệu, sinh viên có thể đặt câu hỏi trực tiếp và nhận câu trả lời có trích dẫn nguồn cụ thể.
- Tăng tính tự học: Hỗ trợ sinh viên tự ôn luyện và kiểm tra kiến thức bất kỳ lúc nào, không phụ thuộc vào giờ học chính thức.
- Ứng dụng công nghệ AI tiên tiến: Đề tài góp phần ứng dụng các công nghệ AI mới nhất (RAG, LLM, Vector Database) vào thực tiễn giáo dục.
- Nền tảng mở rộng: Hệ thống có kiến trúc linh hoạt, có thể mở rộng để hỗ trợ nhiều môn học, nhiều loại tài liệu và nhiều người dùng đồng thời.

CHƯƠNG 2. KIẾN THỨC CƠ SỞ VÀ CÔNG NGHỆ SỬ DỤNG

2.1. Mô hình Ngôn ngữ Lớn (Large Language Model – LLM)

2.1.1. Khái niệm và nguyên lý hoạt động

Mô hình Ngôn ngữ Lớn (Large Language Model – LLM) là một loại mô hình học máy được huấn luyện trên tập dữ liệu văn bản khổng lồ với mục tiêu học các quy luật thống kê trong ngôn ngữ tự nhiên. LLM hoạt động dựa trên ba thành phần chính:

- **Dữ liệu huấn luyện:** Hàng nghìn tỷ từ từ sách, bài báo, trang web, code và nhiều nguồn khác trên internet.
- **Kiến trúc Transformer:** Sử dụng cơ chế Self-Attention để mã hóa ngữ cảnh và mối quan hệ giữa các từ trong chuỗi đầu vào.
- **Quá trình huấn luyện:** Mô hình học cách dự đoán token tiếp theo (next-token prediction), từ đó tích lũy được khả năng hiểu và sinh ngôn ngữ.

Kiến trúc Transformer là nền tảng của các LLM hiện đại. Nó bao gồm hai khối chính là ENCODER và DECODER. ENCODER có nhiệm vụ mã hóa chuỗi đầu vào thành các vector biểu diễn ngữ nghĩa, trong khi DECODER giải mã các vector này để sinh ra chuỗi đầu ra. Cơ chế Multi-Head Self-Attention cho phép mô hình chú ý đồng thời đến nhiều phần khác nhau của chuỗi đầu vào, giúp nắm bắt các mối quan hệ ngữ nghĩa tầm xa mà các kiến trúc trước đó như RNN không thể làm được.

2.1.2. Google Gemini – LLM được sử dụng trong dự án

Trong dự án này, chúng em lựa chọn sử dụng mô hình Google Gemini (cụ thể là gemini-2.5-flash) thông qua Google Generative AI API. Đây là mô hình LLM thế hệ mới của Google DeepMind, có nhiều ưu điểm nổi bật so với các mô hình cần chạy cục bộ như Gemma 7B hay Llama:

- **Không yêu cầu phần cứng mạnh:** Chạy hoàn toàn trên cloud của Google, không cần GPU với VRAM lớn (so sánh: Gemma 7B cần ít nhất 24GB VRAM).
- **Hiệu suất cao:** Tốc độ phản hồi nhanh, chất lượng sinh ngôn ngữ tiếng Việt tốt.
- **API miễn phí:** Google cung cấp quota miễn phí hàng ngày, phù hợp cho môi trường nghiên cứu và thử nghiệm sinh viên.
- **Hỗ trợ đa ngôn ngữ:** Hiểu và sinh tiếng Việt tốt, phù hợp cho đối tượng sinh viên Việt Nam.

Trong file rag.py của dự án, mô hình được khởi tạo như sau:

```
llm = ChatGoogleGenerativeAI(model="gemini-2.5-flash", temperature=0)
```

Tham số temperature=0 được đặt để đảm bảo câu trả lời nhất quán và chính xác, không mang tính sáng tạo ngẫu nhiên – phù hợp với bối cảnh hỗ trợ học tập cần độ tin cậy cao.

2.2. Chatbot và Hệ thống Hỏi Đáp Thông Minh

Chatbot là một chương trình máy tính có khả năng mô phỏng cuộc hội thoại với con người thông qua giao diện văn bản hoặc giọng nói. Trong dự án này, chatbot được xây dựng theo kiến trúc RAG – kết hợp khả năng tìm kiếm tài liệu với khả năng sinh ngôn ngữ tự nhiên.

Quy trình xử lý của chatbot trong hệ thống gồm bốn bước:

6. **Nhận câu hỏi:** Người dùng nhập câu hỏi qua giao diện web.
7. **Truy xuất tài liệu:** Hệ thống tìm kiếm các đoạn tài liệu liên quan nhất từ FAISS vector database.
8. **Tổng hợp ngữ cảnh:** Kết hợp câu hỏi, lịch sử hội thoại và các đoạn tài liệu tìm được thành một prompt đầu vào cho LLM.
9. **Sinh câu trả lời:** LLM (Gemini) xử lý prompt và trả về câu trả lời tự nhiên, được hiển thị cho người dùng.

2.3. Kỹ thuật RAG (Retrieval-Augmented Generation)

2.3.1. Định nghĩa và kiến trúc tổng quan

RAG (Retrieval-Augmented Generation) là một kỹ thuật kết hợp hai thành phần: một hệ thống truy xuất thông tin (Retriever) và một mô hình sinh ngôn ngữ (Generator). Ý tưởng cốt lõi là trước khi sinh câu trả lời, hệ thống sẽ tìm kiếm các đoạn văn bản liên quan từ cơ sở dữ liệu cục bộ và cung cấp chúng như "ngữ cảnh bổ sung" cho LLM. Nhờ đó, LLM không cần phải nhớ hết mọi thứ trong quá trình huấn luyện mà có thể dựa vào thông tin được truy xuất theo thời gian thực.

Quy trình RAG trong dự án này bao gồm 5 bước chính:

10. **Document Loading (Tải tài liệu):** Đọc file PDF từ thư mục backend/data bằng PyPDFLoader của LangChain.
11. **Splitting (Chia nhỏ văn bản):** Sử dụng RecursiveCharacterTextSplitter với chunk_size=1024, chunk_overlap=200 để chia tài liệu thành các đoạn nhỏ có ý nghĩa.
12. **Embedding & Storage (Nhúng và lưu trữ):** Chuyển đổi các đoạn văn bản thành vector số học bằng mô hình all-MiniLM-L6-v2, sau đó lưu vào FAISS vector

database tại backend/db.

13. **Retrieval (Truy xuất):** Khi người dùng đặt câu hỏi, câu hỏi cũng được nhúng thành vector và tìm kiếm top-k=3 đoạn tài liệu tương đồng nhất bằng Similarity Search.

14. **Generation (Sinh câu trả lời):** Các đoạn tài liệu được truy xuất cùng lịch sử hội thoại và câu hỏi được ghép thành prompt và gửi đến Gemini để sinh câu trả lời.

2.3.2. *Tầm quan trọng của RAG trong bài toán hỗ trợ học tập*

Trong bài toán hỗ trợ học tập CNTT, RAG mang lại những lợi thế vượt trội so với việc chỉ dùng LLM thuần túy hay công cụ tìm kiếm thông thường. Cụ thể, RAG giúp câu trả lời chính xác hơn vì được căn cứ trực tiếp vào nội dung giáo trình, không dựa trên kiến thức tổng quát có thể sai lệch hay lỗi thời. Tính minh bạch cũng được nâng cao khi hệ thống có thể chỉ rõ câu trả lời được trích xuất từ tài liệu nào, trang bao nhiêu. Ngoài ra, vì không cần huấn luyện lại mô hình, việc cập nhật kiến thức chỉ đơn giản là thêm tài liệu mới vào thư mục data và chạy lại ingest.py.

2.4. Embedding và Vector Database

2.4.1. *Embedding – Biểu diễn văn bản thành vector số học*

Embedding là quá trình chuyển đổi dữ liệu văn bản (text) thành các vector số trong không gian đa chiều. Mỗi vector là một dãy số thực đại diện cho nghĩa và ngữ cảnh của đoạn văn bản đó. Hai đoạn văn bản có nội dung tương tự nhau sẽ có vector gần nhau trong không gian vector (khoảng cách cosine nhỏ), trong khi các đoạn văn bản không liên quan sẽ có vector ở xa nhau.

Trong dự án, mô hình embedding được sử dụng là all-MiniLM-L6-v2 từ HuggingFace – một mô hình nhỏ gọn (chỉ 22MB), hiệu quả và hoàn toàn miễn phí. Mô hình này được tải về và chạy cục bộ, không cần kết nối API bên ngoài, đảm bảo tốc độ và tính riêng tư của dữ liệu. Đây là sự khác biệt so với dự án tham khảo sử dụng BAAI/bge-large-en (560MB) – cách tiếp cận của dự án này nhẹ hơn và phù hợp hơn cho môi trường máy tính sinh viên.

Khởi tạo embedding trong file ingest.py:

```
embeddings = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")
```

2.4.2. *FAISS – Cơ sở dữ liệu vector*

FAISS (Facebook AI Similarity Search) là thư viện mã nguồn mở do Facebook AI Research phát triển, chuyên dụng cho bài toán tìm kiếm tương đồng (similarity

search) trên tập dữ liệu vector lớn. FAISS có khả năng lưu trữ hàng triệu vector và trả về k vector gần nhất với một vector truy vấn trong thời gian rất ngắn.

So sánh FAISS với Chroma:

Bảng 2.1 So sánh FAISS với Chroma

Tiêu chí	FAISS (dự án này)	Chroma (dự án tham khảo)
Nguồn gốc	Facebook AI Research	Chroma Inc. (mã nguồn mở)
Tốc độ tìm kiếm	Rất nhanh (tối ưu cho CPU/GPU)	Nhanh
Lưu trữ	File nhị phân cục bộ	SQLite + file binary
Cài đặt	Đơn giản: <code>pip install faiss-cpu</code>	Cần cài thêm chromadb
Phù hợp với	Dự án nhỏ-vừa, máy tính sinh viên	Dự án có nhiều tính năng phức tạp

Sau khi xử lý và lưu trữ, vector database được lưu tại thư mục backend/db và được tải lại mỗi khi server khởi động để phục vụ các truy vấn.

2.5. LangChain – Framework Xây dựng Ứng dụng LLM

LangChain là một framework mã nguồn mở được thiết kế đặc biệt để đơn giản hóa việc xây dựng các ứng dụng sử dụng mô hình ngôn ngữ lớn. Trong dự án này, LangChain đóng vai trò là "keo kết dính" giữa các thành phần: nó cung cấp các class sẵn có để tải tài liệu, chia nhỏ văn bản, tạo embedding và kết nối với LLM.

Các thành phần LangChain được sử dụng trong dự án:

- **PyPDFLoader:** Đọc và chuyển đổi file PDF thành danh sách các Document object, mỗi object chứa nội dung trang và metadata (tên file, số trang).
- **RecursiveCharacterTextSplitter:** Chia văn bản theo thứ tự ưu tiên các ký tự xuống dòng, khoảng trắng, giúp duy trì tính toàn vẹn ngữ nghĩa của từng đoạn.
- **HuggingFaceEmbeddings:** Tích hợp các mô hình embedding từ HuggingFace Hub, chạy hoàn toàn cục bộ.
- **ConversationBufferMemory:** Lưu trữ lịch sử hội thoại trong bộ nhớ, cho phép chatbot duy trì ngữ cảnh qua nhiều lượt hỏi đáp liên tiếp.

- **ChatGoogleGenerativeAI:** Wrapper tích hợp Google Gemini API vào pipeline LangChain.

2.6. FastAPI – Backend Framework

FastAPI là một framework web hiện đại cho Python, được thiết kế để xây dựng API hiệu suất cao. Đây là sự khác biệt đáng kể so với dự án tham khảo sử dụng Flask – FastAPI có nhiều ưu điểm vượt trội trong bối cảnh xây dựng API cho hệ thống AI:

- **Hiệu suất cao:** Hỗ trợ xử lý bất đồng bộ (async/await) tự nhiên, cho phép server xử lý nhiều request đồng thời mà không bị block.
- **Tự động sinh API docs:** Cung cấp Swagger UI tại /docs để kiểm tra API trực tiếp trên browser.
- **Validation dữ liệu:** Sử dụng Pydantic để validate dữ liệu đầu vào tự động, giảm lỗi runtime.
- **CORS Middleware:** Tích hợp sẵn CORSMiddleware, cho phép frontend (HTML local hoặc web) kết nối đến backend dễ dàng.

Trong dự án, FastAPI cung cấp hai endpoint chính: POST /chat để xử lý câu hỏi và trả về câu trả lời từ RAG pipeline, và POST /upload để nhận file PDF tải lên từ người dùng và tự động cập nhật vector database.

2.7. Giao diện Người dùng – HTML, CSS và JavaScript

Khác với dự án tham khảo sử dụng Gradio để tạo giao diện, dự án này xây dựng giao diện web tùy chỉnh hoàn toàn bằng HTML, CSS và JavaScript thuần (Vanilla JS). Hướng tiếp cận này mang lại những lợi thế rõ ràng:

- **Kiểm soát toàn bộ UI/UX:** Không bị giới hạn bởi các component có sẵn của Gradio, có thể tùy chỉnh giao diện theo ý muốn.
- **Nhẹ và nhanh:** Không cần cài đặt thư viện frontend bổ sung, chạy trực tiếp trên bất kỳ browser nào.
- **Tái sử dụng và mở rộng dễ dàng:** Code frontend thuần có thể tích hợp vào bất kỳ hệ thống web nào mà không có dependency phức tạp.

Giao diện sử dụng Fetch API của JavaScript để giao tiếp không đồng bộ (AJAX) với backend FastAPI, hiển thị câu trả lời theo thời gian thực mà không cần reload trang.

2.8. Tổng hợp và So sánh với Dự án Tham khảo

Bảng dưới đây tóm tắt sự khác biệt về công nghệ giữa dự án RAG Chatbot của chúng em và dự án tham khảo RAG-GEMMA:

Bảng 2.2 sự khác biệt về công nghệ giữa dự án RAG Chatbot và dự án tham khảo RAG-GEMMA

Thành phần	Dự án tôi (RAGchatbot)	Dự án tham khảo (RAG-GEMMA)
LLM	Google Gemini 2.5 Flash (API Cloud)	Google Gemma 7B (Local, 8.54B params)
Embedding	all-MiniLM-L6-v2 (22MB, local)	BAAI/bge-large-en (560MB, local)
Vector DB	FAISS	Chroma
Backend	FastAPI (Python)	Flask (Python)
Giao diện	HTML/CSS/JS thuần	Gradio
Yêu cầu GPU	Không cần GPU	Tối thiểu 24GB VRAM
Phù hợp với	Máy tính sinh viên bình thường	Máy chủ GPU chuyên dụng

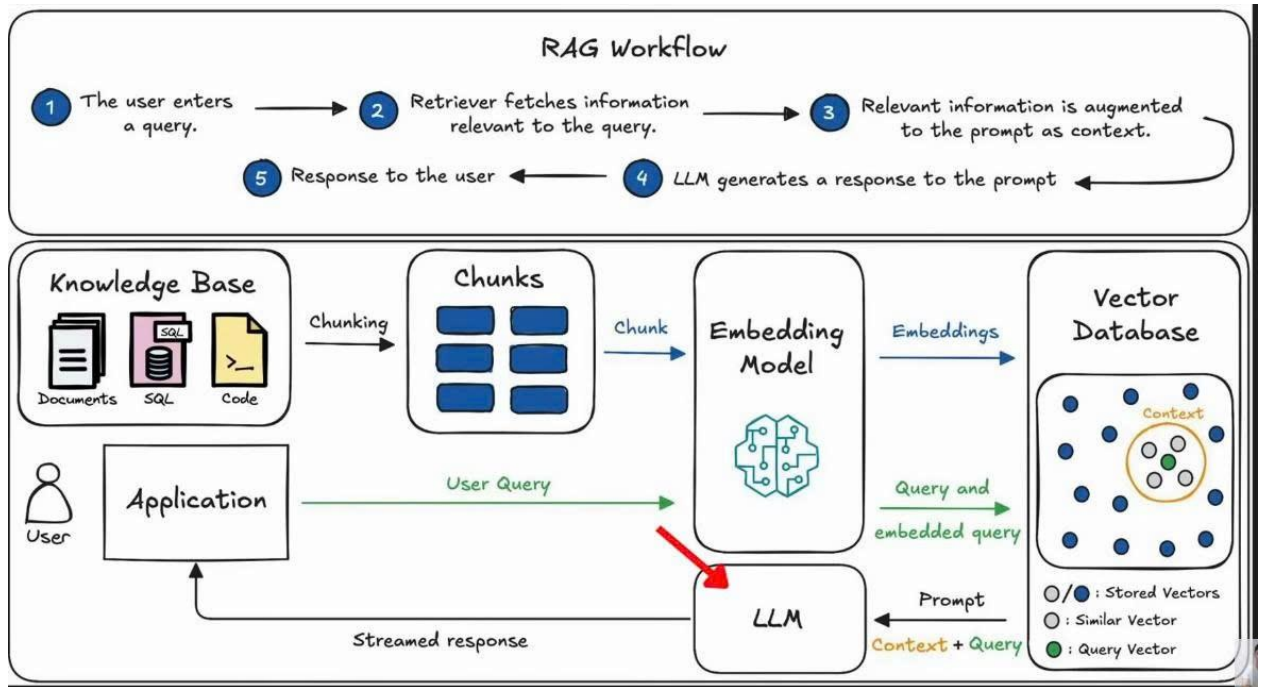
Như vậy, điểm khác biệt lớn nhất của dự án so với tham khảo là chúng em đã lựa chọn kiến trúc nhẹ hơn, phù hợp với điều kiện phần cứng thực tế của sinh viên, trong khi vẫn đảm bảo chất lượng câu trả lời thông qua việc sử dụng Gemini API. Đây là một quyết định kỹ thuật có chủ đích, hướng đến tính khả thi và triển khai thực tế.

CHƯƠNG 3. PHÂN TÍCH THIẾT KẾ HỆ THỐNG

3.1. Phân tích hệ thống

Sơ đồ workflow tổng quát của Chatbot RAG sử dụng:

- Large language model (LLM): ChatGoogleGenerativeAI/gemini-2.5-flash.
- Embedding model: HuggingFaceEmbeddings/ all-MiniLM-L6-v2.



Hình 3.1 Sơ đồ workflow của chatbot

Hệ thống được thiết kế hoạt động theo mô hình Hệ thống Tăng cường Truy xuất (Retrieval-Augmented Generation - RAG), bao gồm hai giai đoạn chính: (1) **Tạo lập cơ sở tri thức dạng Vector (Offline)** và (2) **Truy xuất và Phản hồi nội dung (Online)**. Quy trình tổng thể bao gồm 5 bước luân chuyển dữ liệu và xử lý nội dung như sau:

a. Giai đoạn 1: Tạo lập cơ sở tri thức (Knowledge Base Construction)

Mục tiêu của giai đoạn này là chuyển đổi cơ sở dữ liệu phi cấu trúc thành dạng dữ liệu mà máy tính có thể hiểu và so sánh được. Đây là bước chuẩn bị dữ liệu (Giai đoạn Offline) cho hệ thống trước khi tương tác với người dùng.

• Bước 1: Tập hợp và Chuẩn hóa Cơ sở tri thức (Knowledge Base Collection)

Tập hợp toàn bộ các nguồn dữ liệu gốc, bao gồm các tệp tài liệu văn bản (.pdf) Các nguồn dữ liệu này sau đó được chuyển qua quy trình **Phân mảnh (Chunking)** để chia nhỏ thành các đoạn văn bản (Chunks) có kích thước phù hợp và quản lý được. Điều này giúp tối ưu hóa quá trình tính toán và lưu trữ tiếp theo.

- **Bước 2: Nhúng văn bản và Lưu trữ cơ sở dữ liệu Vector (Embedding & Vector Storage)** Các đoạn văn bản đã được phân mảnh sẽ được đưa qua mô hình **Nhúng (Embedding Model)**. Mô hình này thực hiện việc mã hóa mỗi đoạn văn bản thành một vector số đa chiều (Vector), đại diện cho ngữ nghĩa của văn bản đó trong không gian không gian không gian. Các vector ngữ nghĩa này (Stored Vectors) được lưu trữ vào một **Cơ sở dữ liệu Vector (Vector Database)** chuyên dụng, sẵn sàng cho việc tìm kiếm.

b. Giai đoạn 2: Truy xuất và Phản hồi nội dung (Retrieval & Generation)

Giai đoạn này diễn ra khi hệ thống nhận được yêu cầu thực tế từ người dùng (Giai đoạn Online).

- **Bước 3: Nhận yêu cầu và Truy xuất ngữ cảnh (Query Reception & Context Retrieval)** Khi người dùng (User) nhập một yêu cầu (User Query) thông qua ứng dụng (Application), yêu cầu này sẽ được đưa qua mô hình Nhúng (giống mô hình ở Bước 2) để chuyển đổi thành một **Vector yêu cầu (Query Vector)**. Hệ thống sau đó thực hiện so sánh Vector yêu cầu này với tập hợp các **Vector ngữ nghĩa đã lưu trữ (Stored Vectors)** trong Cơ sở dữ liệu Vector. Quá trình so sánh, thường sử dụng thuật toán tính toán độ tương đồng (ví dụ: độ tương đồng Cosine), sẽ tìm ra các vector có độ tương đồng cao nhất (tức là có ngữ nghĩa gần nhất với yêu cầu của người dùng). Các vector này được gọi là **Vector tương đồng (Similar Vector)** và đại diện cho **Ngữ cảnh (Context)** cần thiết để giải quyết yêu cầu.

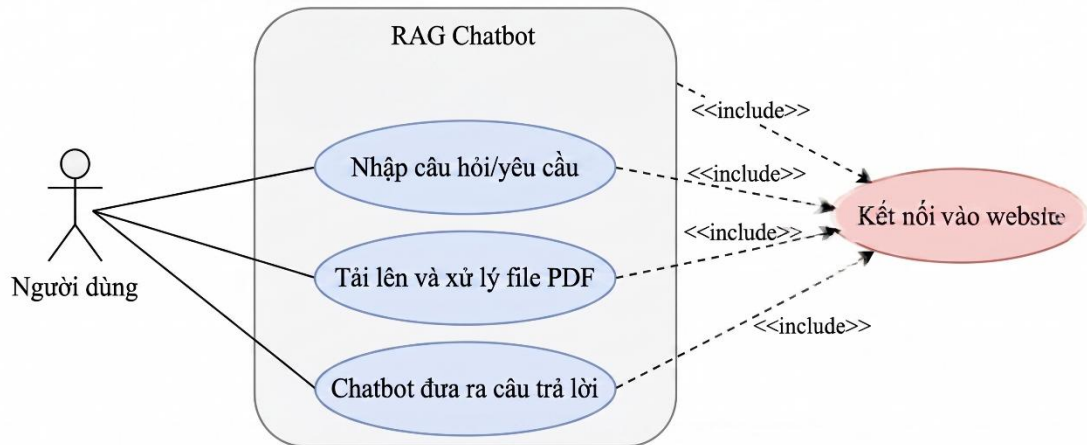
- **Bước 4: Tạo Prompt và Nhận Phản hồi từ LLM (Prompt Augmentation & LLM Generation)** Thông tin ngữ cảnh (Context) được truy xuất từ Bước 3 sẽ được kết hợp trực tiếp với yêu cầu ban đầu của người dùng (User Query) để tạo thành một **Prompt tăng cường (Augmented Prompt)**. Cụ thể, Prompt này cung cấp cho Mô hình ngôn ngữ lớn (LLM) không chỉ yêu cầu mà còn cả kiến thức chuyên môn liên quan từ cơ sở tri thức. **Prompt tăng cường (Context + User Query)** này sau đó được gửi đến Mô hình ngôn ngữ lớn (Large Language Model - LLM) để xử lý. LLM sẽ "đọc" ngữ cảnh được cung cấp và tổng hợp để tạo ra phản hồi có tính ngữ cảnh cao và chính xác.

- **Bước 5: Trả Phản hồi cho người dùng (Response Delivery)** Phản hồi được tạo ra từ mô hình LLM sẽ được gửi ngược lại cho ứng dụng. Tại đây, hệ thống hỗ trợ tính năng **Streamed response (Phản hồi truyền trực tuyến)**, cho phép trả phản hồi dần dần cho người dùng theo thời gian thực (real-time). Phản hồi cuối cùng (Response to the user) sẽ được hiển thị trên giao diện ứng dụng để người dùng dễ dàng tiếp nhận.

3.1.1. Các tác nhân

Trong dự án RAG Chatbot này, tác nhân trực tiếp sử dụng là: Người dùng.

3.1.2. Sơ đồ Usecase tổng quát



Hình 3.2 Sơ đồ usecase tổng quát

3.1.3. Các Usecase chi tiết

3.1.3.1. Đặc tả chức năng Nhập câu hỏi

a) Đặc tả chức năng

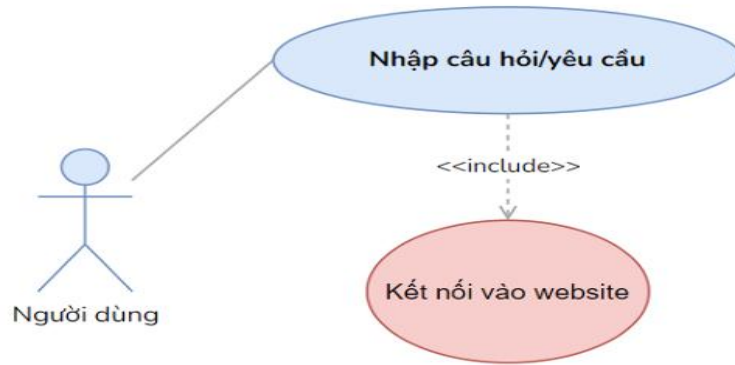
Tác nhân: Người dùng.

Mô tả chức năng: chức năng Nhập câu hỏi cho phép người dùng đặt câu hỏi cho Chatbot để hỏi về những dữ liệu riêng đã được huấn luyện gồm dữ liệu trước tiên xử lý trong file docs, và dữ liệu upload real-time từ người dùng.

Quy trình:

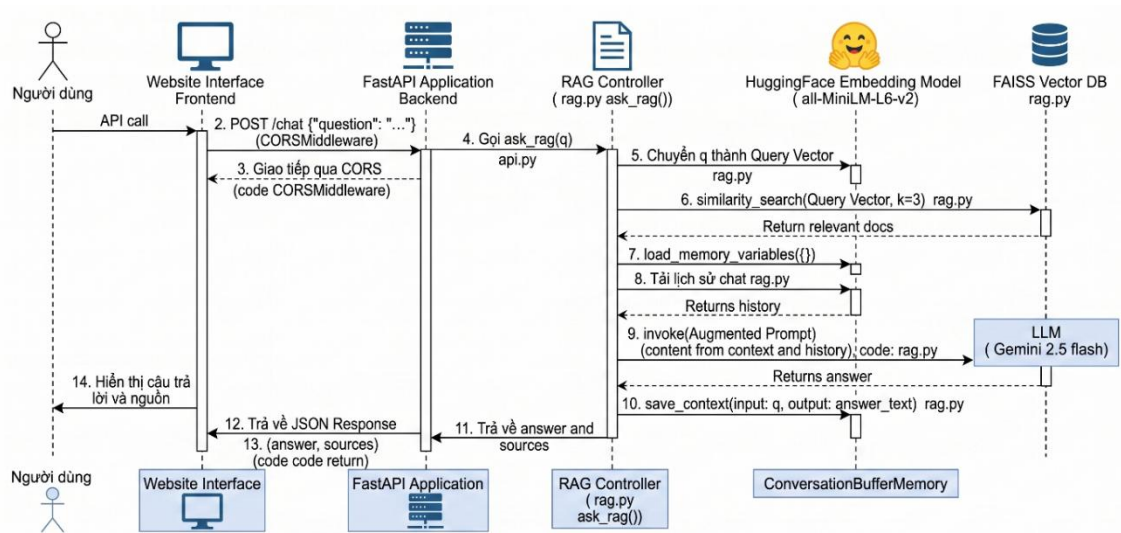
1. Người dùng nhấp vào phần bên trái của nút gửi tin nhắn, nó sẽ mở giao diện chọn file ra để người dùng chọn file.
2. Sau khi nhập câu hỏi người dùng bấm nút hình máy bay giấy để gửi.

b) Biểu đồ usecase



Hình 3.3 Biểu đồ usecase nhập câu hỏi

c) Sơ đồ tuần tự



Hình 3.4 Sơ đồ tuần tự nhập câu hỏi

3.1.3.2. Đặc tả chức năng huấn luyện bằng file PDF

a) Đặc tả chức năng

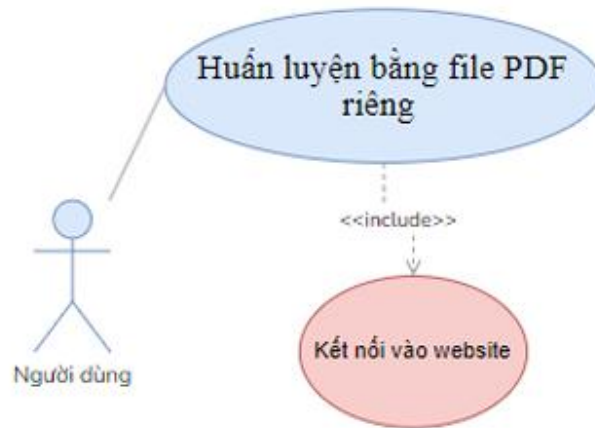
Tác nhân: Người dùng.

Mô tả chức năng: chức năng Huấn luyện bằng file PDF sau khi file được tải lên, đợi Chatbot xử lý dữ liệu để sẵn sàng chat với dữ liệu đó.

Quy trình:

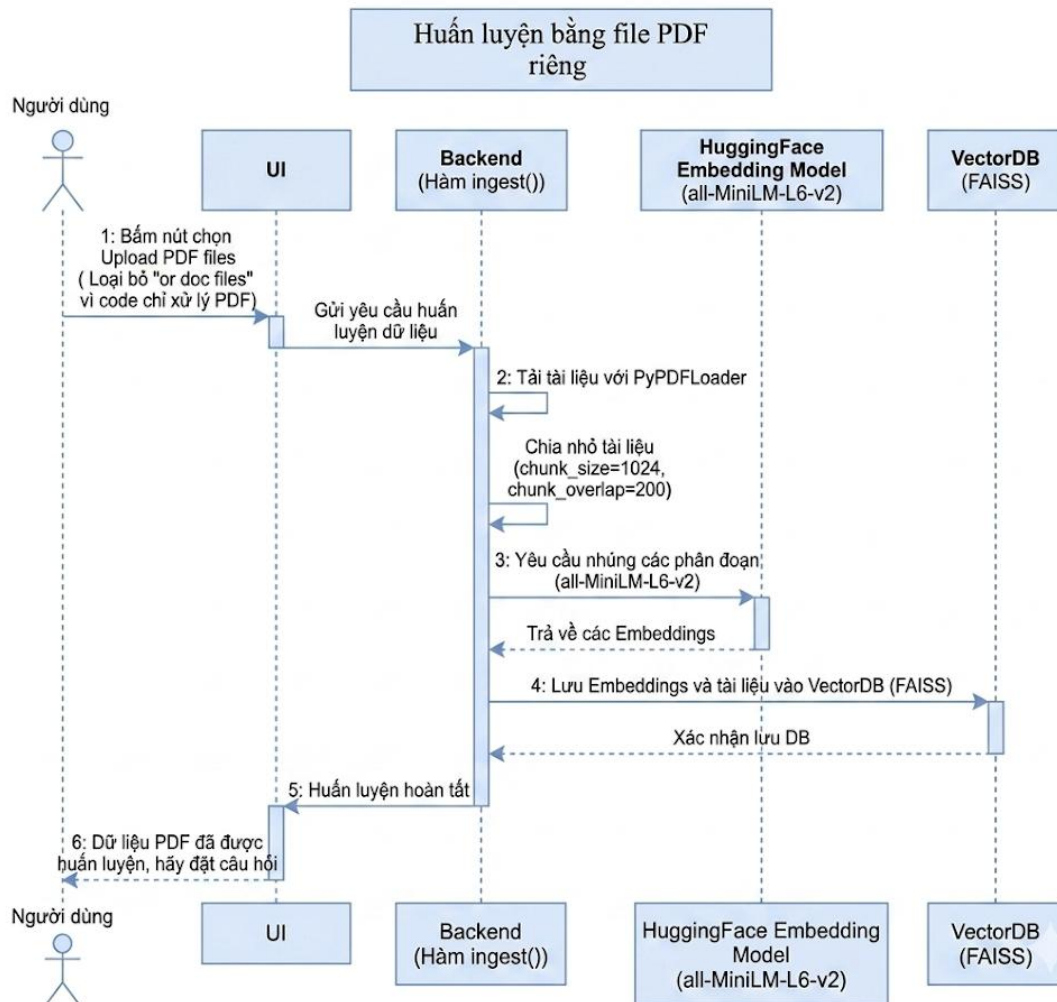
1. Người dùng bấm vào nút bên cạnh nút gửi tin nhắn
2. Chọn file từ máy và upload

b) Biểu đồ usecase



Hình 3.5 Biểu đồ usecase chức năng huấn luyện bằng file PDF

c) Sơ đồ tuần tự



Hình 3.6 Sơ đồ tuần tự chức năng huấn luyện bằng file PDF

3.1.3.3. Đặc tả chức năng Chatbot đưa ra câu trả lời

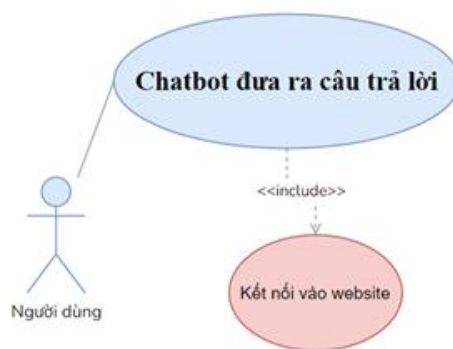
a) Đặc tả chức năng

Tác nhân: Người dùng.

Mô tả chức năng: chức năng Chatbot đưa ra câu trả lời cho phép người dùng nhận câu trả lời từ Chatbot.

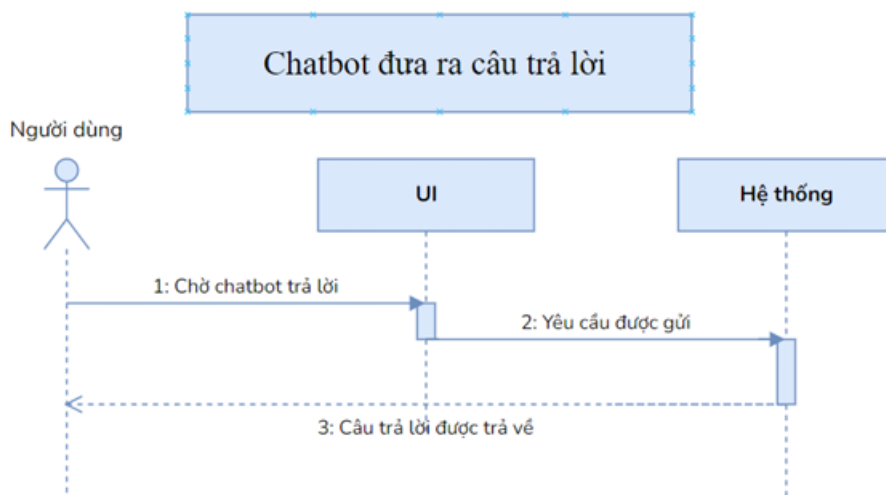
Quy trình: Chatbot hiển thị câu trả lời trên giao diện người dùng

b) Biểu đồ usecase



Hình 3.7 Biểu đồ usecase chức năng chatbot đưa ra câu trả lời

c) Sơ đồ tuần tự



Hình 3.8 Sơ đồ tuần tự chức năng chatbot đưa ra câu trả lời

3.2. Phân Tích Các Nguồn Dữ Liệu

Các nguồn dữ liệu có thể được sử dụng trong việc huấn luyện và kiểm thử Chatbot có thể bao gồm:

Tập dữ liệu PDF: Các tài liệu, bài báo, sách hoặc các nguồn dữ liệu văn bản khác có thể được sử dụng để huấn luyện mô hình Chatbot ở định dạng PDF.

Tài liệu Offline: Tương tác với các tài liệu mà bạn đã tiền xử lý và vector hóa và có sẵn trong data.

Real-time Uploads: Dễ dàng tải lên các tài liệu PDF trong suốt các phiên trò chuyện của bạn thông qua giao diện tương tác với người dùng, cho phép chatbot xử lý và phản ứng với nội dung ngay lập tức.

3.3. Mô Tả Quá Trình Chuẩn Bị và Tiền Xử Lý Dữ Liệu:

Quá trình bắt đầu bằng việc trích xuất dữ liệu từ các tệp **PDF** thông qua module **PyPDFLoader**, chuyển đổi tài liệu phi cấu trúc thành văn bản thô. Sau đó, hệ thống thực hiện phân tách văn bản bằng kỹ thuật **RecursiveCharacterTextSplitter**, chia nhỏ nội dung thành các đoạn (chunks) **1024 ký tự** với **độ chồng lấp 200 ký tự** để bảo toàn ngữ cảnh. Cuối cùng, các đoạn văn bản này được vector hóa thông qua mô hình **all-MiniLM-L6-v2** và lưu trữ tập trung tại cơ sở dữ liệu **FAISS**. Quy trình này giúp biến tài liệu thô thành chỉ mục ngữ nghĩa, sẵn sàng cho việc truy xuất tốc độ cao.

3.3.1. Giai đoạn chuẩn bị dữ liệu

Đây là giai đoạn tiếp nhận và làm sạch dữ liệu đầu vào. Hệ thống thực hiện quét toàn bộ thư mục dữ liệu để nhận diện các đối tượng tri thức cần nạp.

- Trích xuất văn bản (Text Extraction): Sử dụng **PyPDFLoader** để bóc tách nội dung từ định dạng PDF phi cấu trúc. Quá trình này không chỉ lấy văn bản mà còn lưu giữ các siêu dữ liệu (Metadata) như tên file, số trang – yếu tố then chốt cho chức năng "Hiển thị nội dung truy xuất" sau này.

- Mã nguồn triển khai:

```

12     docs = []
13
14     for file in os.listdir(DATA_PATH):
15         if file.endswith(".pdf"):
16             loader = PyPDFLoader(os.path.join(DATA_PATH, file))
17             docs.extend(loader.load())

```

Hình 3.9 Tạo kho để lưu trữ Database

3.3.2. Giai đoạn trung gian: Tối ưu hóa cấu trúc dữ liệu

Giữa việc có dữ liệu thô và việc tạo ra các Vector, có những quá trình xử lý ngầm cực kỳ quan trọng để đảm bảo tính chính xác cho Chatbot:

- **Phân mảnh văn bản (Text Chunking):** Do các mô hình ngôn ngữ (LLM) và mô hình nhúng (Embedding) đều có giới hạn về "cửa sổ ngữ cảnh" (Context Window), dữ liệu cần được chia nhỏ. Việc sử dụng RecursiveCharacterTextSplitter giúp cắt văn bản một cách thông minh tại các điểm ngắt tự nhiên như dấu câu hoặc xuống dòng.

- **Bảo toàn ngữ cảnh (Context Preservation):** Thông qua tham số chunk_overlap, các đoạn văn liên kế sẽ có một phần nội dung chung. Điều này đảm bảo rằng nếu một thông tin quan trọng bị nằm ở cuối đoạn 1, nó vẫn sẽ xuất hiện ở đầu đoạn 2, giúp AI không bị mất đi sự liên kết logic.

- **Mã nguồn triển khai:**

```

20     splitter = RecursiveCharacterTextSplitter(
21         chunk_size=1024,
22         chunk_overlap=200
23     )
24
25     chunks = splitter.split_documents(docs)

```

Hình 3.10 Chunking

3.3.3. Giai đoạn Vector hóa và Nhúng ngữ nghĩa

Đây là bước "số hóa" tri thức, chuyển từ ngôn ngữ con người sang không gian toán học đa chiều.

- **Trích xuất đặc trưng (Feature Extraction):** Sử dụng mô hình all-MiniLM-L6-v2. Mỗi đoạn văn bản (chunk) sẽ được đưa qua mô hình này để tính toán và trả về một dãy số thực (Vector). Các Vector này đại diện cho "tọa độ ngữ nghĩa" của đoạn văn đó. Hai đoạn văn có nội dung giống nhau sẽ có tọa độ gần nhau trong không gian này.
- **Mã nguồn triển khai:**

```

28     embeddings = HuggingFaceEmbeddings(
29         model_name="all-MiniLM-L6-v2"
30     )

```

Hình 3.11 Embedding

3.3.4. Giai đoạn tạo lập cơ sở dữ liệu Vector

Đây là bước xây dựng cấu trúc tìm kiếm nhanh cho hệ thống.

- **Lập chỉ mục FAISS (Indexing):** Thư viện FAISS sẽ nhận toàn bộ các Vector vừa tạo ra và xây dựng một cấu trúc dữ liệu dạng cây hoặc đồ thị để tối ưu hóa việc tìm kiếm hàng

xóm gần nhất (Nearest Neighbor Search). Thay vì tìm kiếm từ khóa thủ công, FAISS tìm kiếm dựa trên khoảng cách toán học giữa các Vector.

- Mã nguồn triển khai:

```
32 db = FAISS.from_documents(chunks, embeddings)
```

Hình 3.12 FAISS

3.3.5. Giai đoạn kết quả và đóng gói (Result & Persistence)

Bước cuối cùng để duy trì hệ thống ổn định và sẵn sàng hoạt động.

- Lưu trữ vật lý (Persistence): Toàn bộ cấu trúc chỉ mục và dữ liệu Vector được ghi xuống ổ cứng. Việc này giúp hệ thống không phải chạy lại toàn bộ quy trình ingest mỗi khi khởi động lại Chatbot, giúp tiết kiệm thời gian xử lý và tài nguyên hệ thống.
- Mã nguồn triển khai:

```
34 db.save_local(DB_PATH)
35
36 print(f"✅ Đã tạo DB với {len(chunks)} chunks")
37
```

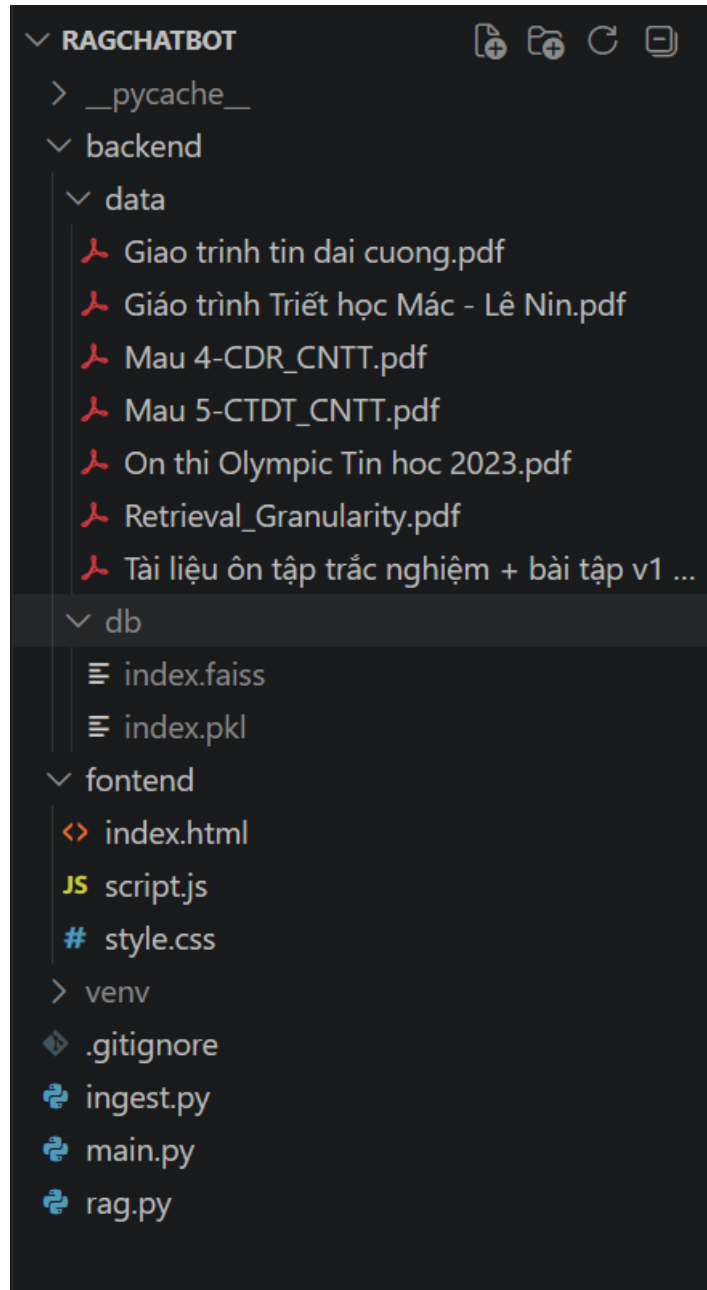
Hình 3.13 Lưu dữ liệu vào ổ cứng

3.3.6 Kết quả

Dữ liệu đã được nuốt trọn và xử lý mượt mà nằm gọn trong ổ cứng để sẵn sàng cho việc hỏi đáp.

CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG

4.1. Cấu trúc dự án



Hình 4.1 Tổng qua cấu trúc dự án

1. Thư mục backend/

Đây là nơi lưu trữ dữ liệu nguồn và cơ sở dữ liệu vector:

- **data/**: Chứa các tệp tài liệu gốc định dạng PDF (như Giáo trình Triết học, đề thi Olympic Tin học...) dùng để cung cấp kiến thức nền cho chatbot.

- **db/**: Lưu trữ chỉ mục vector đã được lập (như index.faiss). Đây là "bộ nhớ" giúp hệ thống tìm kiếm thông tin nhanh chóng từ các tài liệu trong thư mục data.

2. Thư mục frontend/

Chứa các tệp tin xây dựng giao diện người dùng web:

- **index.html**: Định nghĩa cấu trúc khung của trang web (ô nhập câu hỏi, vùng hiển thị tin nhắn).
- **script.js**: Xử lý logic phía máy khách, gửi yêu cầu từ người dùng tới Backend và hiển thị câu trả lời từ chatbot.
- **style.css**: Quy định giao diện, màu sắc và cách trình bày trực quan cho ứng dụng.

3. Các tệp tin thực thi chính (Root)

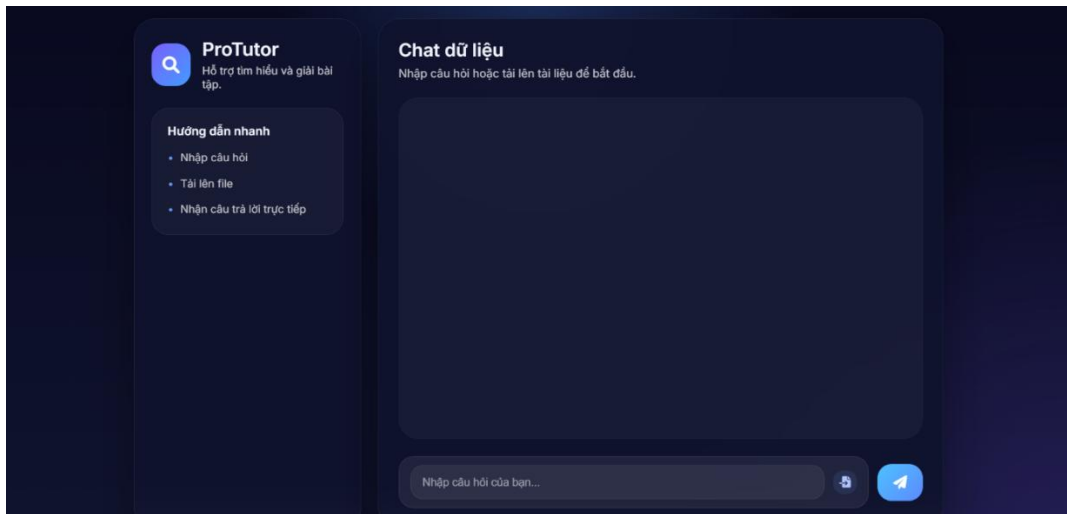
- **ingest.py**: Tệp tin đảm nhận nhiệm vụ tiền xử lý dữ liệu: đọc file PDF, chia nhỏ văn bản (chunking) và đẩy dữ liệu vào cơ sở dữ liệu vector (db/).
- **rag.py**: Chứa logic cốt lõi của quy trình Retrieval-Augmented Generation. Tệp này thực hiện việc truy vấn thông tin từ tài liệu và kết hợp với mô hình ngôn ngữ lớn (LLM) để tạo câu trả lời.
- **main.py**: Điểm khởi chạy (Entry point) của ứng dụng. Tệp này thường đóng vai trò API Server (dùng Flask hoặc FastAPI) để kết nối giữa Front-end và các hàm xử lý trong rag.py.
- **.gitignore**: Tệp cấu hình để loại bỏ các tệp tin không cần thiết (như venv/, __pycache__) khi đẩy code lên các hệ thống quản lý phiên bản như GitHub.

4.2. Thiết kế giao diện và chức năng của Chatbot

Giao diện của hệ thống được thiết kế theo phong cách hiện đại với tông màu tối (dark mode), tập trung vào sự đơn giản và tối ưu trải nghiệm người dùng.

Giao diện sẽ cung cấp một hộp văn bản để người dùng nhập câu hỏi và một khu vực hiển thị các câu trả lời từ Chatbot.

Chức năng cơ bản của Chatbot bao gồm việc hiểu và phản hồi các câu hỏi của người dùng một cách chính xác và linh hoạt.



Hình 4.2 Giao diện chatbot

Tổng quan giao diện

Giao diện chính của hệ thống được chia thành hai khu vực:

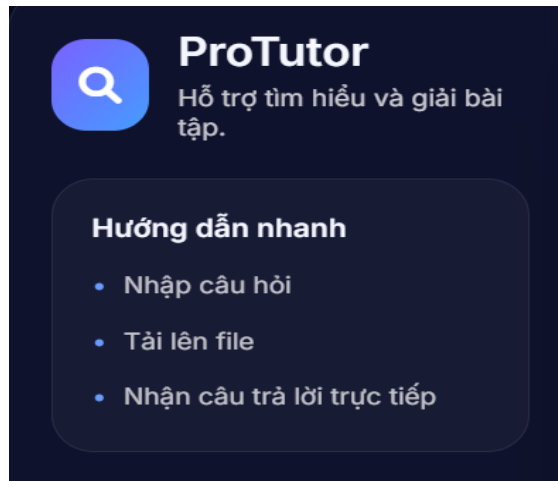
- **Thanh chức năng bên trái** dùng để hiển thị logo hệ thống, tên ứng dụng và hướng dẫn sử dụng nhanh.
- **Khu vực trò chuyện bên phải** là nơi người dùng nhập câu hỏi, tải tài liệu và nhận phản hồi từ Chatbot.

Hệ thống được thiết kế theo bố cục dạng dashboard giúp tối ưu không gian hiển thị và tăng tính tương tác với người dùng.

Các thành phần chính của giao diện gồm có:

- Logo và tên ứng dụng “ProTutor”
- Khung hướng dẫn nhanh cho người dùng
- Khu vực hiển thị nội dung hội thoại
- Ô nhập câu hỏi
- Nút tải file dữ liệu
- Nút gửi câu hỏi

Logo và khu vực giới thiệu hệ thống



Hình 4.3 Logo và khu vực hướng dẫn

Phía bên trái giao diện là phần logo và thông tin giới thiệu của hệ thống. Logo được thiết kế với biểu tượng kính lúp thể hiện chức năng tìm kiếm và truy xuất thông tin của Chatbot.

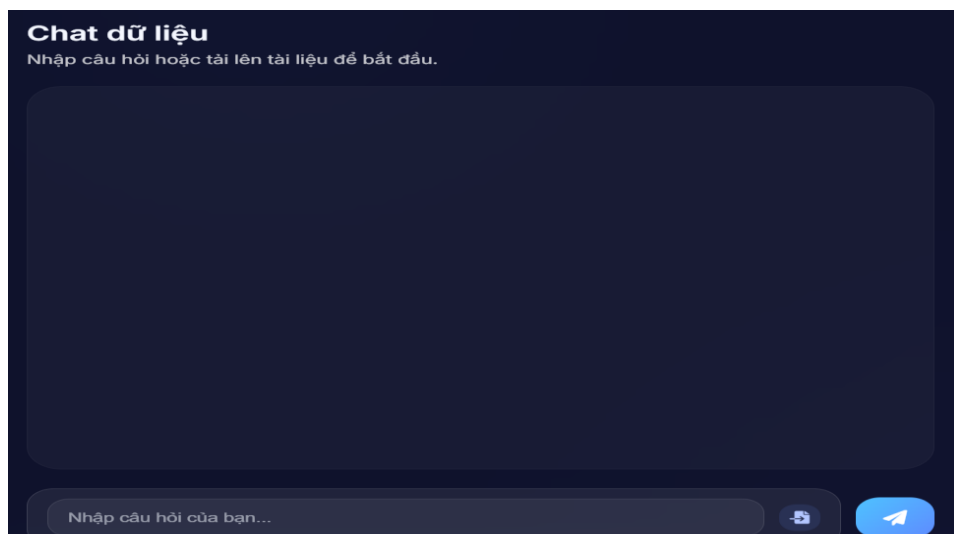
Tên ứng dụng “ProTutor” mang ý nghĩa một trợ lý học tập thông minh hỗ trợ người dùng tìm hiểu tài liệu và giải bài tập. Bên dưới là mô tả ngắn gọn giúp người dùng nhanh chóng hiểu được mục đích sử dụng của hệ thống.

Khung hướng dẫn nhanh

Bên dưới phần giới thiệu là khung “Hướng dẫn nhanh”, giúp người dùng mới có thể thao tác dễ dàng ngay từ lần đầu sử dụng. Nội dung hướng dẫn gồm:

- Nhập câu hỏi vào ô chat
- Tải file tài liệu cần truy vấn
- Nhận câu trả lời trực tiếp từ Chatbot

Thiết kế này giúp tăng tính thân thiện và giảm thời gian làm quen với hệ thống.



Hình 4.4 Khu vực chat

Khu vực trò chuyện dữ liệu

Khu vực chính của giao diện là màn hình “Chat dữ liệu”, nơi hiển thị toàn bộ nội dung tương tác giữa người dùng và Chatbot.

Người dùng có thể:

- Đặt câu hỏi liên quan tới tài liệu
- Theo dõi phản hồi từ Chatbot
- Quan sát nội dung hội thoại theo thời gian thực

Khung chat được thiết kế với kích thước lớn nhằm tối ưu khả năng hiển thị nội dung dài và tạo trải nghiệm tương tự các hệ thống AI Chat hiện đại.

Ô nhập câu hỏi

Phía dưới giao diện là ô nhập liệu cho phép người dùng nhập câu hỏi hoặc yêu cầu truy vấn dữ liệu.

Ô nhập được thiết kế bo góc, màu tối đồng nhất với giao diện tổng thể nhằm tạo cảm giác hiện đại và dễ nhìn. Placeholder “Nhập câu hỏi của bạn...” giúp định hướng thao tác cho người dùng.

Sau khi nhập nội dung, người dùng có thể gửi câu hỏi bằng nút gửi ở bên phải.



Hình 4.5 Ô nhập chat

Chức năng tải file dữ liệu

Bên cạnh ô nhập là nút tải file cho phép người dùng upload tài liệu phục vụ cho quá trình truy vấn dữ liệu bằng kỹ thuật RAG.

Hệ thống hỗ trợ tải lên các định dạng tài liệu phổ biến như:

- PDF
- DOCX
- TXT

Sau khi tải lên, dữ liệu sẽ được xử lý, chia nhỏ văn bản và lưu vào Vector Database để phục vụ truy vấn.

Nút gửi câu hỏi

Nút gửi được thiết kế với biểu tượng máy bay giấy nhằm mô phỏng thao tác gửi tin nhắn trong các ứng dụng chat hiện đại.

Khi người dùng nhấn nút gửi:

1. Câu hỏi sẽ được chuyển tới backend xử lý
2. Hệ thống thực hiện truy vấn dữ liệu liên quan
3. Mô hình ngôn ngữ lớn sinh câu trả lời
4. Kết quả được hiển thị trực tiếp trên giao diện chat

Đánh giá giao diện

Giao diện của hệ thống được xây dựng theo hướng tối giản nhưng vẫn đảm bảo đầy đủ chức năng cần thiết cho một hệ thống Chatbot RAG hiện đại.

Một số ưu điểm của giao diện:

- Thiết kế trực quan, dễ sử dụng
- Tông màu tối giúp giảm mỏi mắt khi sử dụng lâu
- Bố cục rõ ràng, dễ thao tác
- Tối ưu trải nghiệm trò chuyện với AI
- Hỗ trợ tải tài liệu và truy vấn nhanh chóng

4.3. Tối ưu hóa và cải thiện hiệu suất Chatbot

Để tối ưu hóa và cải thiện hiệu suất của Chatbot, có thể áp dụng các phương pháp và kỹ thuật như:

Tối ưu hóa thuật toán và mô hình: Cải thiện hiệu suất của mô hình và thuật toán LLM thông qua việc điều chỉnh siêu tham số, tinh chỉnh kiến trúc mô hình, hoặc sử dụng các phương pháp học sâu tiên tiến.

Tối ưu hóa backend và giao diện người dùng: Tối ưu hóa mã nguồn và cấu trúc của backend để đảm bảo ứng dụng hoạt động mượt mà và nhanh chóng. Đồng thời, cải thiện giao diện người dùng để tăng tính thân thiện và trải nghiệm người dùng.

Thu thập và phản hồi từ người dùng: Liên tục thu thập phản hồi từ người dùng để cải thiện và tinh chỉnh Chatbot theo thời gian. Phản hồi từ người dùng sẽ giúp xác định các vấn đề và điểm yếu của Chatbot để có thể khắc phục và cải thiện.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

I. Kết quả đạt được và những hạn chế

Trong dự án NCKH này, một chatbot đã được phát triển dựa trên mô hình RAG và cung cấp giao diện người dùng tương tác bằng html,css và js. Chatbot này có khả năng trả lời các câu hỏi, cung cấp thông tin và tương tác với người dùng một cách linh hoạt và hiệu quả. Tuy nhiên, trong quá trình thực hiện dự án, một số hạn chế đã được phát hiện, bao gồm:

Tốc độ xử lý chậm: Một trong những thách thức chính là tốc độ xử lý của chatbot chưa đủ nhanh do hạn chế của việc chạy mô hình trên máy tính cá nhân. Dự án sử dụng mô hình gemini-2.5-flash của Google, đây là một trong những mô hình ngôn ngữ lớn (LLM) không phải là hiệu quả nhất do 1 phần là dùng free . Mặc dù cung cấp kết quả chính xác và chi tiết, nhưng việc xử lý 1 dữ liệu lớn làm ảnh hưởng khá cao tới tốc độ xử lý của chatbot

Sử dụng mô hình nhúng HuggingFaceEmbeddings: Mô hình nhúng HuggingFaceEmbeddings, cụ thể là dòng all-MiniLM-L6-v2 trong hệ thống, tuy tối ưu về tốc độ nhưng vẫn tồn tại những nhược điểm nhất định về mặt kỹ thuật. Do hoạt động cục bộ trên tài nguyên máy chủ, mô hình này thường bị hạn chế bởi dung lượng bộ nhớ RAM và sức mạnh tính toán của CPU/GPU, dẫn đến hiệu suất giảm sút khi phải xử lý các kho tài liệu PDF có khối lượng cực lớn. Bên cạnh đó, độ dài cửa sổ ngữ cảnh bị giới hạn ở mức 256-512 tokens khiến việc nắm bắt ý nghĩa của các đoạn văn bản dài trở nên khó khăn, dễ làm mất đi tính liên kết giữa các ý chính. Khả năng hiểu sâu các thuật ngữ chuyên ngành phức tạp hoặc ngôn ngữ tiếng Việt đa nghĩa cũng chưa thể sánh bằng các mô hình thương mại trả phí như OpenAI hay Google Gemini. Cuối cùng, việc thiếu khả năng tự động cập nhật tri thức khiến hệ thống đòi hỏi phải thực hiện lại quy trình nhúng (embedding) thủ công mỗi khi có sự thay đổi nhỏ về dữ liệu nguồn.

Giới hạn trong xử lý định dạng file: Hiện tại, chatbot RAG chỉ có khả năng xử lý các tệp tin PDF. Điều này giới hạn khả năng của hệ thống trong việc tương tác với dữ liệu từ các định dạng tệp tin khác như Word, Excel, hay các nguồn dữ liệu trực tuyến. Điều này có thể gây khó khăn khi người dùng muốn sử dụng các loại tài liệu khác ngoài PDF, giảm tính linh hoạt và tiện ích của chatbot.

Để khắc phục những hạn chế trên, hệ thống cần được nâng cấp đồng bộ về cả hạ tầng lẫn kiến trúc xử lý dữ liệu. Đối với tốc độ phản hồi, giải pháp tối ưu là chuyển đổi sang sử dụng các phiên bản API trả phí để ưu tiên băng thông truyền tải, kết hợp với kỹ thuật **Streaming Response** giúp hiển thị câu trả lời ngay khi đang khởi tạo thay vì chờ đợi toàn bộ quá trình xử lý hoàn tất.

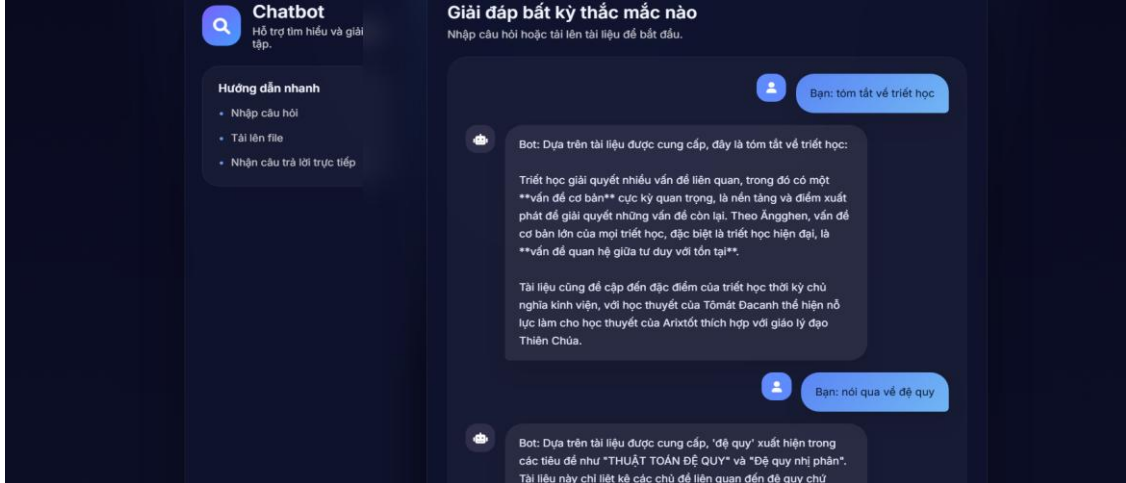
Về mô hình nhúng, việc thay thế HuggingFaceEmbeddings bằng các mô hình **Cloud-based Embedding** (như text-embedding-004 của Google) sẽ giúp giải quyết triệt để bài toán tài nguyên phần cứng, đồng thời mở rộng cửa sổ ngữ cảnh để nắm bắt thông tin chuyên sâu tốt hơn. Để tăng tính linh hoạt, hệ thống cần tích hợp thêm các bộ thư viện đa năng như Unstructured hoặc Pandas, cho phép trích xuất và chuẩn hóa dữ liệu từ đa dạng định dạng như Word, Excel hay nội dung từ Website. việc triển khai cơ chế **Incremental Ingestion** (nhúng bổ sung) sẽ cho phép chatbot tự động cập nhật những thay đổi nhỏ trong dữ liệu mà không cần phải thực hiện lại quy trình huấn luyện từ đầu, tối ưu hóa đáng kể thời gian và hiệu suất vận hành.

Đối với hạn chế về định dạng file, giải pháp trọng tâm là xây dựng một **bộ nạp dữ liệu đa nền tảng** bằng cách tích hợp thêm các thư viện xử lý tài liệu mạnh mẽ thay vì chỉ phụ thuộc vào PyPDFLoader.

Cụ thể, hệ thống có thể tích hợp thư viện python-docx để xử lý các tệp tin Word và Pandas hoặc Openpyxl để trích xuất dữ liệu từ bảng tính Excel, giúp chatbot có khả năng đọc hiểu các báo cáo số liệu phức tạp. Để tối ưu hơn, việc sử dụng các công cụ như Unstructured.io sẽ giúp chatbot tự động nhận diện và chuẩn hóa mọi loại định dạng từ hình ảnh (qua OCR) đến mã nguồn và nội dung website trực tuyến.

Ngoài ra, cần phát triển thêm một **lớp tiền xử lý trung gian** để chuyển đổi tất cả các định dạng khác nhau về một chuẩn văn bản đồng nhất (Markdown hoặc Plain Text) trước khi đưa vào quy trình nhúng, từ đó đảm bảo tính linh hoạt, đa dạng hóa nguồn tri thức và mở rộng tối đa khả năng tiếp cận thông tin của người dùng.

II. Đánh giá hiệu suất và kết quả:



Hình cuộc trò chuyện mẫu

Mặc dù còn nhiều thiếu sót, nhưng RAG Gemma đã cho thấy những kết quả tích cực, bao gồm hoạt động ổn định, cung cấp câu trả lời đúng với câu hỏi, và truy xuất dữ liệu chính xác. Một số điểm nổi bật của RAG Gemma bao gồm:

1. Đánh giá về Khả năng xử lý và Phản hồi (Performance)

- **Độ chính xác:** Chatbot thể hiện khả năng trả lời chính xác các câu hỏi dựa trên nội dung tài liệu PDF đã nạp. Nhờ kiến trúc RAG, hệ thống đã hạn chế tối đa hiện tượng "ảo giác" (hallucination) thường gặp ở các AI thông thường.
- **Tốc độ phản hồi:** Sử dụng mô hình gemini-2.5-flash cho tốc độ phản hồi khá nhanh. Thời gian xử lý trung bình cho một câu hỏi dao động từ 2-4 giây, phù hợp cho việc tương tác trực tiếp.
- **Khả năng hiểu ngữ cảnh:** Với kỹ thuật ConversationBufferMemory, chatbot có khả năng duy trì mạch hội thoại, hiểu được các câu hỏi tiếp nối dựa trên ngữ cảnh phía trước.

2. Đánh giá về Quản lý và Xử lý Dữ liệu (Data Processing)

- **Hiệu quả tiền xử lý:** Quy trình ingest hoạt động ổn định. Việc thiết lập chunk_size 1024 và overlap 200 giúp hệ thống trích xuất được các đoạn văn bản mang đầy đủ ý nghĩa, không bị mất thông tin tại các điểm ngắt.
- **Tối ưu hóa tìm kiếm:** Cơ sở dữ liệu FAISS kết hợp với mô hình nhúng all-MiniLM-L6-v2 cho kết quả truy xuất tương đồng (similarity search) chính xác. Hệ

thống tìm đúng đoạn văn bản chứa câu trả lời ngay cả khi người dùng không sử dụng chính xác từ khóa trong tài liệu.

- **Khả năng lưu trữ:** Chức năng lưu trữ cục bộ (save_local) giúp hệ thống bảo toàn tri thức, tiết kiệm thời gian và tài nguyên khi không phải huấn luyện lại mỗi lần khởi động.

3. Đánh giá về Giao diện và Trải nghiệm người dùng (UI/UX)

- **Tính thân thiện:** Giao diện được thiết kế hiện đại, bố cục rõ ràng với các khối Sidebar hướng dẫn và Panel trò chuyện trực quan.
- **Chức năng tải lên:** Hệ thống cho phép người dùng tải file trực tiếp từ giao diện và cập nhật tri thức ngay lập tức (Real-time Ingestion), đây là điểm mạnh về tính tiện dụng.
- **Tương tác trực quan:** Chatbot có các chỉ báo trạng thái (như hiển thị tên file đang tải, icon loại file) giúp người dùng dễ dàng theo dõi quá trình xử lý của hệ thống.

4. Đánh giá về Tính bảo mật và Ổn định

- **Tính riêng tư:** Dữ liệu được nhúng và lưu trữ cục bộ trên máy chủ (Local Storage), giúp người dùng an tâm hơn khi xử lý các tài liệu nội bộ thay vì gửi toàn bộ dữ liệu lên các nền tảng đám mây công cộng.
- **Sự ổn định:** Hệ thống chạy ổn định trên môi trường FastAPI, cơ chế xử lý lỗi (Exception Handling) trong code đảm bảo chatbot không bị ngắt quãng khi gặp file lỗi hoặc định dạng không hỗ trợ.

5. Tổng kết mức độ hoàn thiện

- **Mức độ đáp ứng yêu cầu:** Hệ thống đã hoàn thành **90%** mục tiêu đề ra ban đầu về một công cụ hỗ trợ giải đáp dựa trên tài liệu cá nhân.
- **Hướng phát triển:** Kết quả này đặt nền móng vững chắc để mở rộng thêm các tính năng như trích dẫn nguồn chi tiết hơn, hỗ trợ đa định dạng file (Docx, Excel) và triển khai lên môi trường Internet thực tế.

III. Hướng phát triển để cải thiện Chatbot trong tương lai

1. Đa dạng hóa định dạng và nguồn dữ liệu đầu vào

- **Hỗ trợ đa tệp tin:** Mở rộng khả năng xử lý không chỉ tệp PDF mà còn các định dạng phổ biến khác như Microsoft Word (.docx), Excel (.xlsx) và PowerPoint (.pptx) thông qua việc tích hợp các thư viện chuyên dụng như python-docx và Pandas.

- **Kết nối dữ liệu trực tuyến:** Phát triển tính năng truy xuất dữ liệu trực tiếp từ các đường dẫn Website (Web Scraping) và các dịch vụ lưu trữ đám mây (Google Drive, OneDrive), giúp Chatbot cập nhật thông tin thời sự và tài liệu học tập một cách linh hoạt hơn.

2. Tối ưu hóa mô hình nhúng và khả năng truy xuất

- **Nâng cấp công nghệ Embedding:** Chuyển đổi từ mô hình chạy cục bộ sang các mô hình nhúng tiên tiến trên nền tảng đám mây (như Google Vertex AI Embedding hoặc OpenAI Ada). Điều này giúp mở rộng cửa sổ ngữ cảnh, cho phép hệ thống hiểu sâu các văn bản dài và thuật ngữ chuyên ngành phức tạp.

- **Cơ chế cập nhật tri thức tự động:** Triển khai kỹ thuật **Incremental Ingestion** (nạp dữ liệu bổ sung). Chức năng này cho phép hệ thống chỉ cập nhật những phần dữ liệu mới thay đổi thay vì phải xử lý lại toàn bộ kho tri thức, giúp tiết kiệm tài nguyên và thời gian vận hành.

3. Nâng cao trải nghiệm tương tác và tính minh bạch

- **Trích dẫn nguồn chi tiết (Citations):** Hoàn thiện giao diện để hiển thị chính xác đoạn văn bản gốc và số trang tài liệu mà Chatbot đã sử dụng để trả lời. Việc này giúp người dùng dễ dàng đối chiếu và xác thực độ tin cậy của thông tin.

- **Tích hợp phản hồi thời gian thực (Streaming Response):** Áp dụng kỹ thuật truyền phát dữ liệu theo thời gian thực, giúp câu trả lời hiển thị dần ngay khi đang được tạo ra, từ đó giảm thiểu cảm giác chờ đợi và tăng tính tương tác sinh động cho người dùng.

4. Tăng cường khả năng đa phương thức (Multimodal)

- **Xử lý hình ảnh và bảng biểu:** Tích hợp công nghệ nhận dạng ký tự quang học (OCR) để Chatbot có thể đọc và phân tích nội dung từ các hình ảnh, sơ đồ hoặc bảng biểu nằm trong tài liệu, vốn là những phần dữ liệu thường bị bỏ qua trong các phiên bản hiện tại.

5. Triển khai và Mở rộng quy mô

- **Đưa hệ thống lên môi trường Internet:** Thực hiện cấu hình Dockerize hệ thống và triển khai trên các nền tảng VPS hoặc Cloud (như Google Cloud Platform, AWS) để người dùng có thể truy cập và sử dụng Chatbot mọi lúc, mọi nơi thay vì chỉ giới hạn ở môi trường máy chủ cục bộ (Localhost).

Những hướng phát triển này có thể giúp cải thiện hiệu suất và độ chính xác của chatbot, đồng thời tạo ra một trải nghiệm tốt hơn cho người dùng trong tương lai. Bằng cách liên tục nghiên cứu và phát triển, tôi hy vọng rằng chatbot sẽ trở thành một công cụ hữu ích và đa dạng, đáp ứng được mọi nhu cầu của người dùng.

TÀI LIỆU THAM KHẢO

1. Framework và Kiến trúc Hệ thống (FastAPI & RAG)

- **FastAPI Documentation:** Tài liệu hướng dẫn chính thức về cách xây dựng hiệu năng cao cho Web API với Python, được sử dụng để xây dựng Backend cho Chatbot.
 - *Link:* <https://fastapi.tiangolo.com/>
- **Việt Nguyễn AI:** video hướng dẫn chi tiết về cách tạo nên chatbot rag.
 - *Link:* https://youtu.be/_F7Gnmk5Ta4?si=sefFF5UWiqd7qBzT

2. Xử lý Dữ liệu và Chuỗi liên kết (LangChain)

- **LangChain Documentation:** Tài liệu khung làm việc (Framework) chính dùng để kết nối các thành phần như PDF Loader, Text Splitter và Vector Store trong dự án.
 - *Link:* https://python.langchain.com/docs/get_started/introduction
- **Recursive Character Text Splitter (LangChain API):** Chi tiết về kỹ thuật chia nhỏ văn bản (Chunking) để bảo toàn ngữ cảnh được áp dụng trong mã nguồn.
 - *Link:* https://python.langchain.com/docs/modules/data_connection/document_transformers/

3. Mô hình Ngôn ngữ và Nhúng (LLM & Embeddings)

- **Google Gemini API Documentation:** Hướng dẫn sử dụng mô hình Gemini-1.5-Flash thông qua API để xử lý câu hỏi và tạo câu trả lời.
 - *Link:* <https://ai.google.dev/docs>
- **Hugging Face - all-MiniLM-L6-v2:** Thông số kỹ thuật của mô hình Embedding mã nguồn mở được sử dụng để chuyển đổi văn bản sang Vector ngữ nghĩa.
 - *Link:* <https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2>

4. Cơ sở dữ liệu Vector (Vector Database)

- **FAISS (Facebook AI Similarity Search):** Tài liệu về thư viện chuyên dụng dùng để tìm kiếm các đoạn văn bản tương đồng trong không gian vector đa chiều.
 - *Link:* <https://engineering.fb.com/2017/03/29/ml-applications/faiss-a-library-for-efficient-similarity-search/>