ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Đặng Quang Huy

PHÁT TRIỂN PHẦN MỀM CHATBOT DỰA VÀO MÔ HÌNH NGÔN NGỮ LỚN ĐỂ TRUY XUẤT THÔNG TIN TỪ CƠ SỞ DỮ LIỆU QUAN HỆ CHO NGHIỆP VỤ ĐÀO TẠO ĐẠI HỌC

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Khoa học máy tính

HÀ NỘI – 2025

ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Đặng Quang Huy

PHÁT TRIỂN PHẦN MỀM CHATBOT DỰA VÀO MÔ HÌNH NGÔN NGỮ LỚN ĐỂ TRUY XUẤT THÔNG TIN TỪ CƠ SỞ DỮ LIỆU QUAN HỆ CHO NGHIỆP VỤ ĐÀO TẠO ĐẠI HỌC

KHÓA LUẬN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

Ngành: Khoa học máy tính

Cán bộ hướng dẫn: PGS.TS Đặng Đức Hạnh

HÀ NỘI – 2025

TÓM TẮT

Tóm tắt:

Trong bối cảnh giáo dục đại học đòi hỏi các giải pháp tự động hóa hiệu quả, chatbot dựa trên mô hình ngôn ngữ lớn (LLM) là một công cụ tiềm năng để hỗ trợ truy xuất thông tin từ cơ sở dữ liệu quan hệ, phục vụ nghiệp vụ đào tạo. Khóa luận này đề xuất một hệ thống chatbot với các đặc điểm nổi bật: (1) giao diện người dùng được xây dựng bằng React (JavaScript), (2) phía backend sử dụng Golang để đảm bảo hiệu suất cao, (3) tích hợp lời gọi hàm với các mã nguồn dịch vụ cụ thể và hàm generic để truy xuất dữ liệu từ cơ sở dữ liệu quan hệ thông qua truy vấn SQL động, (4) cơ chế bảo mật và phân quyền chặt chẽ, cho phép học sinh và giáo viên chỉ truy xuất dữ liệu trong phạm vi quyền hạn.

Hệ thống đề xuất một phương pháp phân quyền SQL, trong đó LLM sinh ra các câu lệnh SQL, được phân tích thành cây cú pháp bằng postgres-parser, sau đó kiểm tra quyền truy cập dựa trên các node. Giải pháp được triển khai và kiểm chứng trên một hệ thống cơ sở dữ liệu mô phỏng, đạt hiệu quả cao trong việc trả lời các câu hỏi về lịch học, điểm số và thông tin sinh viên.

Khóa luận cung cấp một giải pháp thực tiễn cho quản lý đào tạo đại học, đồng thời mở ra hướng nghiên cứu về ứng dụng LLM, RAG và phân quyền SQL trong các hệ thống thông tin giáo dục.

Từ khóa: *Chatbot, Mô hình ngôn ngữ lớn, Cơ sở dữ liệu quan hệ, Phân quyền SQL, gọi hàm*

LỜI CAM ĐOAN

Tôi xin cam đoan rằng: Báo cáo khóa luận tốt nghiệp với đề tài **"Phát triển phần mềm chatbot dựa vào mô hình ngôn ngữ lớn để truy xuất thông tin từ cơ sở dữ liệu quan hệ cho nghiệp vụ đào tạo đại học"** là kết quả nghiên cứu và thực hiện của chính tôi dưới sự hướng dẫn của **PGS.TS. Đặng Đức Hạnh**.

Các nội dung, số liệu, hình ảnh và kết quả trình bày trong báo cáo là trung thực và chưa từng được công bố trong bất kỳ công trình nào khác. Các thông tin thứ cấp được sử dụng trong khóa luận là có nguồn gốc và được trích dẫn rõ ràng. Trong trường hợp phát hiện có sự sao chép hoặc gian lận trong báo cáo, tôi xin hoàn toàn chịu trách nhiệm trước hội đồng chấm khóa luận và quy định của Nhà trường.

Hà Nội, tháng 5 năm 2025

Sinh viên thực hiện

Đặng Quang Huy

LỜI CẢM ƠN

Trước tiên, em xin bày tỏ lòng biết ơn chân thành và sâu sắc đến **PGS.TS. Đặng Đức Hạnh**, người đã tận tình hướng dẫn, hỗ trợ và định hướng em trong suốt quá trình thực hiện khóa luận tốt nghiệp. Sự chỉ dẫn tận tâm của Thầy là nguồn động lực lớn giúp em hoàn thành tốt đề tài này.

Em xin gửi lời cảm ơn đến **Khoa Công Nghệ Thông Tin**, **Trường Đại Học Công Nghệ - Đại Học Quốc Gia Hà Nội**, cùng toàn thể quý Thầy, Cô đã trang bị cho em những kiến thức quý báu trong suốt quá trình học tập tại trường, là nền tảng để em thực hiện đề tài khóa luận này.

Bên cạnh đó, em cũng xin cảm ơn các anh/chị và bạn bè đã nhiệt tình giúp đỡ, chia sẻ tài liệu và đóng góp ý kiến trong quá trình nghiên cứu và triển khai giải pháp.

Cuối cùng, em xin gửi lời cảm ơn sâu sắc đến gia đình – những người luôn ở bên cạnh, động viên và tạo điều kiện tốt nhất cho em trong suốt thời gian học tập và nghiên cứu.

Mục lục

Danh sách hình vẽ

Danh sách bảng

Danh sách mã nguồn

Mở đầu

Giới thiệu

Cơ sở dữ liệu quan hệ xưa nay vẫn chứng minh được tính bền vững và ổn định của mình và vẫn đang chiếm được thị phần lớn trong giới công nghệ. Cùng với đó, truy xuất dữ liệu từ cơ sở dữ liệu quan hệ vẫn luôn là chủ đề nóng hổi trong khoa học máy tính và công nghệ. Tuy nhiên, với người dùng phổ thông, không phải ai cũng có kiến thức chuyên ngành về ngôn ngữ truy vấn SQL, khiến cho việc truy vấn dữ liệu gặp nhiều khó khăn và bị giới hạn bởi những logic định sẵn. Vì vậy, nhu cầu truy xuất dữ liệu bằng ngôn ngữ tự nhiên vẫn luôn hiện hữu và đang ngày càng trở nên khả dĩ với sự phát triển của trí tuệ nhân tạo AI và các mô hình ngôn ngữ lớn gần đây. Sự ra đời của một giải pháp cho phép truy xuất dữ liệu bằng ngôn ngữ tự nhiên nhưng vẫn đảm bảo tính bảo mật và phân quyền rõ ràng là tất yếu.

Xét bài toán trong bối cảnh cụ thể là trường đại học, nhu cầu truy xuất thông tin nhanh chóng, tiện lợi và thân thiện cũng ngày càng trở nên cấp thiết. Sinh viên thường xuyên có nhiều thắc mắc về thời khóa biểu, điểm số, hoặc các quy trình học tập, nhưng không phải lúc nào cũng tiện đến phòng đào tạo để được giải đáp. Thông tin bị phân tán qua nhiều kênh như email, website đào tạo, hoặc thông báo riêng lẻ, gây khó khăn trong việc tra cứu. Mặc dù các cổng thông tin một cửa đã được triển khai để tập trung dữ liệu, sự tiện lợi của LLM khuyến khích sinh viên sử dụng ngôn ngữ tự nhiên để đặt câu hỏi, mang lại cảm giác gần gũi và linh hoạt hơn. Giảng viên cần một công cụ hỗ trợ để quản lý lớp học, tra cứu thông tin về sinh viên hoặc lịch giảng dạy một cách nhanh chóng, tương tự như một trợ lý đắc lực có thể trả lời hầu hết các câu hỏi. Việc truy xuất thông tin từ cơ sở dữ liệu quan hệ thường đòi hỏi kỹ năng kỹ thuật, gây trở ngại cho những giảng viên không chuyên về công nghệ.

Những khó khăn trên cho thấy sự cần thiết của một giải pháp tích hợp, tận dụng sức mạnh của LLM để xử lý ngôn ngữ tự nhiên và truy xuất dữ liệu từ cơ sở dữ liệu quan hệ. Khóa luận này hướng đến phát triển một hệ thống chatbot thông minh, tập trung giải quyết các vấn đề trong nghiệp vụ đào tạo đại học, hỗ trợ sinh viên, giảng viên một cách hiệu quả, đồng thời đặt nền tảng cho các ứng dụng tương tự trong các lĩnh vực khác.

Mục tiêu đề tài

Mục tiêu của khóa luận là xây dựng một hệ thống chatbot tích hợp mô hình ngôn ngữ lớn (LLM) với cơ chế phân quyền và bảo mật, đảm bảo sinh viên và giảng viên chỉ truy cập được dữ liệu trong phạm vi quyền hạn của mình. Hệ thống sử dụng LLM để sinh câu lệnh SQL từ các yêu cầu ngôn ngữ tự nhiên, kết hợp với phân tích cây cú pháp bằng postgres-parser để kiểm tra và thực thi phân quyền truy cập. Giải pháp nhằm cung cấp một công cụ an toàn, thân thiện, hỗ trợ truy xuất thông tin từ cơ sở dữ liệu quan hệ cho nghiệp vụ đào tạo đại học, đồng thời đóng góp vào nghiên cứu về ứng dụng LLM trong quản lý dữ liệu.

Phạm vi đề tài

Nghiên cứu tập trung vào phát triển hệ thống chatbot hỗ trợ nghiệp vụ đào tạo đại học, với các phạm vi cụ thể:

- Hỗ trợ sinh viên và giảng viên truy xuất thông tin về thời khóa biểu, điểm số, hồ sơ sinh viên và môn học, thông tin ngành, khoa trong môi trường đại học.
- Áp dụng cơ chế phân quyền để đảm bảo mỗi nhóm người dùng chỉ truy cập dữ liệu được phép, tăng cường bảo mật và quyền riêng tư.
- Triển khai và đánh giá hệ thống trên cơ sở dữ liệu quan hệ mô phỏng để kiểm chứng tính khả thi và hiệu quả trong bối cảnh giáo dục.

Ý nghĩa của đề tài

Đề tài mang lại giá trị quan trọng về cả mặt thực tiễn và học thuật trong bối cảnh nghiệp vụ đào tạo đại học:

- **Tăng cường khả năng tiếp cận**: Chatbot cho phép sinh viên và giảng viên truy xuất thông tin quan trọng (như thời khóa biểu, điểm số, hồ sơ sinh viên và môn học) thông qua câu hỏi ngôn ngữ tự nhiên, giảm phụ thuộc vào kỹ năng kỹ thuật và các nguồn thông tin phân tán, từ đó cải thiện trải nghiệm người dùng và hiệu quả vận hành.
- **Nâng cao bảo mật**: Hệ thống áp dụng cơ chế phân quyền và kiểm tra truy vấn SQL an toàn bằng LLM và phân tích cây cú pháp, đảm bảo quyền riêng tư dữ liệu và bảo vệ thông tin nhạy cảm trong giáo dục.

– **Đóng góp học thuật**: Đề tài đề xuất một phương pháp phân quyền khi truy xuất dữ liệu quan hệ sử dụng LLM, đảm bảo tính bảo mật thông tin.

– **Đổi mới mở rộng**: Việc chứng minh tính khả thi của chatbot dựa trên LLM trong môi trường đại học mô phỏng đặt nền tảng cho các ứng dụng tương lai trong các lĩnh vực sử dụng nhiều dữ liệu như y tế hoặc quản lý doanh nghiệp.

Đề tài không chỉ giải quyết các thách thức tức thời trong đào tạo đại học mà còn khởi đầu cho ứng dụng công nghệ AI trong các hệ thống truy xuất dữ liệu an toàn, thân thiện với người dùng trong những lĩnh vực khác.Phần còn lại của khóa luận được cấu trúc như dưới đây.

Cấu trúc khóa luận

– **Chương 1: Cơ sở lý thuyết** — Trình bày các kiến thức nền tảng về mô hình ngôn ngữ lớn (LLM), cơ sở dữ liệu quan hệ, bộ phân tích cú pháp SQL và giao thức Server-Sent Events (SSE), làm cơ sở cho việc phát triển hệ thống.

– **Chương 2: Phát biểu bài toán và đặc tả yêu cầu** — Phân tích nghiệp vụ đào tạo đại học, phát biểu bài toán, đề xuất giải pháp phần mềm và đặc tả các yêu cầu chức năng, phi chức năng của hệ thống chatbot.

– **Chương 3: Phân tích thiết kế hệ thống** — Trình bày quá trình phân tích, thiết kế kiến trúc hệ thống, các ca sử dụng, thiết kế cơ sở dữ liệu và quy tắc phân quyền truy vấn SQL.

– **Chương 4: Cài đặt và thực nghiệm** — Mô tả quá trình cài đặt các thành phần chính của hệ thống, tích hợp mô hình ngôn ngữ lớn, kiểm thử các chức năng tiêu biểu và đánh giá kết quả thực nghiệm.

Chương 1.

Cơ sở lý thuyết

1.1. Mô hình ngôn ngữ lớn (LLM)

Trước khi đi vào khái niệm của mô hình ngôn ngữ lớn, ta tìm hiểu khái niệm trí tuệ nhân tạo - AI. Trí tuệ nhân tạo là một công nghệ có khả năng giải quyết vấn đề như con người. Cách thức hoạt động của AI dường như mô phỏng trí tuệ của con người – nó có thể nhận dạng hình ảnh, làm thơ và đưa ra dự đoán dựa trên dữ liệu.

Các tổ chức hiện đại thu thập khối lượng dữ liệu cực lớn từ nhiều nguồn khác nhau như cảm biến thông minh, nội dung do con người tạo ra, công cụ giám sát và bản ghi hệ thống. Công nghệ trí tuệ nhân tạo phân tích và sử dụng dữ liệu để hỗ trợ hoạt động kinh doanh một cách hiệu quả. Ví dụ: công nghệ AI có thể phản hồi các cuộc trò chuyện của con người trong lĩnh vực hỗ trợ khách hàng, tạo hình ảnh và văn bản gốc để tiếp thị và đưa ra các đề xuất thông minh để phân tích.

Cuối cùng, mục tiêu của trí tuệ nhân tạo là làm cho phần mềm thông minh hơn để đáp ứng tương tác người dùng tùy chỉnh và giải quyết vấn đề phức tạp

1.1.1. Large Language Models (LLM) là gì?

Mô hình ngôn ngữ lớn (LLM) là các hệ thống trí tuệ nhân tạo được thiết kế để hiểu và tạo ra ngôn ngữ tự nhiên, dựa trên việc huấn luyện với khối lượng dữ liệu văn bản khổng lồ. Chúng có khả năng thực hiện các tác vụ như trả lời câu hỏi, tạo văn bản, tóm tắt nội dung, và hỗ trợ lập trình, trở thành nền tảng quan trọng cho các ứng dụng tương tác với người dùng. LLM nổi bật nhờ khả năng linh hoạt, cho phép xử lý nhiều loại yêu cầu ngôn ngữ mà không cần huấn luyện lại hoàn toàn, từ chatbot đến phân tích văn bản.

Trong bối cảnh luận văn này, LLM đóng vai trò cốt lõi trong việc sinh ra các câu truy vấn và hiểu ý định người dùng, tạo nền tảng cho chatbot thông minh tích hợp với cơ

sở dữ liệu. Sự dễ tiếp cận của LLM thông qua các giao diện lập trình đã mở ra cơ hội để áp dụng chúng vào các giải pháp thực tế.

1.1.2. Giới thiệu về API của LLM

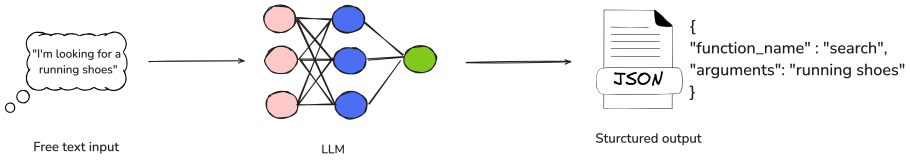
API (Application Programming Interface) của mô hình ngôn ngữ lớn là cầu nối giữa các mô hình phức tạp và ứng dụng thực tế, cho phép tích hợp LLM vào hệ thống như chatbot mà không cần xây dựng mô hình từ đầu. Các API này cho phép gửi đầu vào văn bản (như câu hỏi của người dùng) và nhận phản hồi được tạo ra bởi LLM, chẳng hạn như câu trả lời tự nhiên hoặc truy vấn SQL. Điều này đơn giản hóa quá trình phát triển, giúp chatbot của chúng ta tận dụng sức mạnh của LLM một cách hiệu quả.

Việc sử dụng API cho LLM không chỉ tăng tốc độ phát triển mà còn cung cấp khả năng tùy chỉnh và mở rộng, dẫn đến việc khám phá các API cụ thể như OpenAI API và Google AI API, phù hợp với mục tiêu của chatbot trong luận văn này.

1.1.3. Gọi hàm (Function calling)

Gọi hàm (function calling) như được định nghĩa bởi Martin Fowler [1] là một khả năng cho phép các mô hình ngôn ngữ lớn (LLM) vượt qua việc chỉ tạo văn bản đơn giản bằng cách tương tác với các công cụ bên ngoài và các ứng dụng thực tế. Với tính năng này, một LLM có thể phân tích đầu vào ngôn ngữ tự nhiên, rút ra ý định của người dùng, và tạo ra đầu ra có cấu trúc chứa tên hàm và các tham số cần thiết để gọi hàm đó.

Điều quan trọng cần nhấn mạnh là khi sử dụng gọi hàm, chính LLM không thực thi hàm. Thay vào đó, nó xác định hàm phù hợp, thu thập tất cả các tham số cần thiết, và cung cấp thông tin dưới dạng JSON có cấu trúc. Đầu ra JSON này có thể dễ dàng được chuyển đổi ngược (deserialized) thành một cuộc gọi hàm trong Python (hoặc bất kỳ ngôn ngữ lập trình nào khác) và thực thi trong môi trường chạy chương trình. Hình 1.1 minh họa cách hoạt động của gọi hàm:



Hình 1.1. Yêu cầu ngôn ngữ tự nhiên đến đầu ra có cấu trúc

Ví dụ, nếu người dùng nói ”Thời tiết hôm nay thế nào” , LLM sẽ trả về lời gọi hàm `getWeather` thay vì trả lời bằng ngôn ngữ tự nhiện

1.2. Cơ sở dữ liệu quan hệ và bộ phân tích cú pháp SQL

1.2.1. Tổng quan cơ sở dữ liệu quan hệ

Cơ sở dữ liệu quan hệ là một loại cơ sở dữ liệu tổ chức dữ liệu thành các hàng và cột, cùng nhau tạo thành một bảng nơi các điểm dữ liệu có mối quan hệ với nhau. Một trong những cơ sở dữ liệu quan hệ được sử dụng phổ biến nhất là Postgres [2]

Dữ liệu thường được cấu trúc trên nhiều bảng, có thể được liên kết với nhau thông qua khóa chính hoặc khóa ngoại. Những định danh duy nhất này thể hiện các mối quan hệ khác nhau tồn tại giữa các bảng, và các mối quan hệ này thường được minh họa thông qua các loại mô hình dữ liệu khác nhau. Các nhà phân tích sử dụng các truy vấn SQL để kết hợp các điểm dữ liệu khác nhau và tóm tắt hiệu suất kinh doanh, giúp các tổ chức có được cái nhìn sâu sắc, tối ưu hóa quy trình làm việc và xác định các cơ hội mới.

Ví dụ, hãy tưởng tượng công ty của bạn duy trì một bảng cơ sở dữ liệu chứa thông tin khách hàng, bao gồm dữ liệu công ty ở cấp tài khoản. Cũng có thể có một bảng khác mô tả tất cả các giao dịch riêng lẻ liên quan đến tài khoản đó. Cùng nhau, các bảng này có thể cung cấp thông tin về các ngành công nghiệp khác nhau mua một sản phẩm phần mềm cụ thể.

Các cột (hoặc trường) cho bảng khách hàng có thể bao gồm ID Khách hàng, Tên Công ty, Địa chỉ Công ty, Ngành công nghiệp, v.v.; các cột cho bảng giao dịch có thể là Ngày Giao dịch, ID Khách hàng, Số tiền Giao dịch, Phương thức Thanh toán, v.v. Các bảng này có thể được liên kết với nhau thông qua trường ID Khách hàng chung. Do đó, bạn có thể truy vấn bảng để tạo ra các báo cáo giá trị, chẳng hạn như báo cáo doanh số theo ngành hoặc công ty, giúp định hướng thông điệp đến các khách hàng tiềm năng.

Cơ sở dữ liệu quan hệ cũng thường được liên kết với các cơ sở dữ liệu giao dịch, thực hiện các lệnh, hoặc giao dịch, một cách tổng hợp. Một ví dụ phổ biến được sử dụng để minh họa điều này là chuyển khoản ngân hàng. Một số tiền nhất định được rút từ một tài khoản, sau đó được gửi vào một tài khoản khác. Tổng số tiền được rút và gửi, và giao

dịch này không thể xảy ra ở trạng thái một phần. Các giao dịch có các thuộc tính cụ thể. Được đại diện bởi từ viết tắt ACID, các thuộc tính ACID được định nghĩa như sau:

- **Tính nguyên tử (Atomicity):** Tất cả các thay đổi đối với dữ liệu được thực hiện như thể chúng là một thao tác duy nhất. Nghĩa là, tất cả các thay đổi được thực hiện, hoặc không có thay đổi nào được thực hiện.
- **Tính nhất quán (Consistency):** Dữ liệu duy trì trạng thái nhất quán từ đầu đến cuối, củng cố tính toàn vẹn dữ liệu.
- **Tính cô lập (Isolation):** Trạng thái trung gian của một giao dịch không hiển thị với các giao dịch khác, và do đó, các giao dịch chạy đồng thời dường như được tuần tự hóa.
- **Tính bền vững (Durability):** Sau khi hoàn thành thành công một giao dịch, các thay đổi đối với dữ liệu được lưu trữ vĩnh viễn và không bị hoàn tác, ngay cả trong trường hợp hệ thống gặp sự cố.

  Những thuộc tính này cho phép xử lý giao dịch đáng tin cậy.

1.2.2. Bộ phân tích cú pháp SQL

Bộ phân tích cú pháp SQL là một thành phần cốt lõi trong các hệ quản trị cơ sở dữ liệu quan hệ (RDBMS), chịu trách nhiệm phân tích các truy vấn SQL để đảm bảo chúng đúng về mặt cú pháp và có thể được thực thi hiệu quả. Khóa luận chọn Postgres làm cơ sở dữ liệu quan hệ, vì vậy ta sẽ đi vào tìm hiểu cụ thể bộ phân tích cú pháp SQL trong Postgres hoạt động như thế nào.

Bộ phân tích cú pháp SQL là một mô-đun phần mềm phân tích cú pháp các câu lệnh SQL, chuyển đổi chúng từ dạng văn bản thành các cấu trúc dữ liệu có thể xử lý, chẳng hạn như Cây cú pháp trừu tượng (Abstract Syntax Tree - AST). Quá trình này đảm bảo rằng truy vấn tuân thủ các quy tắc ngữ pháp của ngôn ngữ SQL và phù hợp với lược đồ cơ sở dữ liệu. [3]

Trong PostgreSQL, bộ phân tích cú pháp là một phần của quy trình xử lý truy vấn, hoạt động như bước đầu tiên sau khi nhận truy vấn từ ứng dụng khách. Nó phân tích cú pháp truy vấn để tạo ra một cây truy vấn (query tree), là biểu diễn nội bộ của truy vấn, trước khi chuyển sang các giai đoạn tối ưu hóa và thực thi [4].

Cơ chế hoạt động của bộ phân tích cú pháp SQL trong PostgreSQL bao gồm các bước kỹ thuật sau:

1. **Phân tích từ vựng (Lexical Analysis):** Bộ phân tích cú pháp sử dụng một công cụ phân tích từ vựng (lexer), được triển khai bằng công cụ Flex, để chia nhỏ truy vấn SQL thành các token. Các token này là các đơn vị cơ bản như từ khóa (SELECT, FROM), định danh (tên bảng, cột), toán tử (>, =), hoặc giá trị chữ (literals). Trong PostgreSQL, tệp `scan.l` định nghĩa các quy tắc để nhận diện token, chẳng hạn như nhận diện từ khóa `SELECT` hoặc chuỗi ký tự biểu thị tên bảng [5].
2. **Phân tích cú pháp (Syntactic Analysis):** Sau khi token hóa, bộ phân tích cú pháp sử dụng một trình phân tích cú pháp (parser), được xây dựng bằng công cụ Bison, để kiểm tra cấu trúc cú pháp của truy vấn dựa trên ngữ pháp SQL. Trong PostgreSQL, ngữ pháp được định nghĩa trong tệp `gram.y`, tuân theo chuẩn SQL (như SQL:2011) với các mở rộng dành riêng cho PostgreSQL. Trình phân tích cú pháp xây dựng một Cây cú pháp trừu tượng (AST), biểu diễn cấu trúc phân cấp của truy vấn, bao gồm các thành phần như mệnh đề SELECT, điều kiện WHERE, hoặc phép nối JOIN. Quá trình này sử dụng thuật toán LALR(1), một dạng phân tích LR, để đảm bảo hiệu quả và tránh quay lui [5].
3. **Phân tích ngữ nghĩa (Semantic Analysis):** Sau khi tạo AST, PostgreSQL thực hiện phân tích ngữ nghĩa để đảm bảo truy vấn có ý nghĩa trong bối cảnh lược đồ cơ sở dữ liệu. Điều này bao gồm kiểm tra sự tồn tại của các bảng và cột được tham chiếu, xác minh tính hợp lệ của các kiểu dữ liệu, và đảm bảo các phép nối hoặc điều kiện hợp lý. Trong PostgreSQL, giai đoạn này chuyển đổi AST thành cây truy vấn (query tree), bổ sung thông tin về lược đồ và chuẩn bị cho giai đoạn lập kế hoạch truy vấn [5].

Việc triển khai bộ phân tích cú pháp trong PostgreSQL phải đối mặt với một số thách thức kỹ thuật. Thứ nhất, ngữ pháp SQL của PostgreSQL rất phức tạp do hỗ trợ các chuẩn SQL quốc tế cùng với các tiện ích mở rộng độc quyền, dẫn đến tệp `gram.y` chứa hàng nghìn quy tắc. Thứ hai, hiệu suất là yếu tố quan trọng, vì bộ phân tích cú pháp phải xử lý nhanh các truy vấn trong môi trường thời gian thực. PostgreSQL tối ưu hóa hiệu suất bằng cách sử dụng các công cụ như Flex và Bison, vốn tạo ra mã C hiệu quả, và bằng cách giảm thiểu xung đột ngữ pháp trong quá trình phân tích.

Tóm lại, bộ phân tích cú pháp SQL trong PostgreSQL là một hệ thống tinh vi, chuyển đổi các truy vấn SQL thành các cấu trúc nội bộ để thực thi. Bằng cách kết hợp phân tích từ vựng, cú pháp, và ngữ nghĩa, nó đảm bảo tính chính xác và hiệu quả của các truy vấn, đồng thời hỗ trợ các tính năng SQL phức tạp. Hiểu rõ cơ chế hoạt động của bộ phân tích cú pháp SQL giúp ta ứng dụng vào các trường hợp hữu ích khi cần thiết, ví dụ như khóa luận đã ứng dụng bộ phân tích cú pháp vào phân quyền câu truy vấn SQL

1.3. Server-Sent Events (SSE)

Server-Sent Events (SSE) [6], hay còn gọi là Sự kiện gửi từ máy chủ, là một giao thức cho phép máy khách nhận các cập nhật tự động từ máy chủ thông qua kết nối HTTP. Giao thức này mô tả cách máy chủ khởi tạo việc truyền dữ liệu đến máy khách sau khi máy khách thiết lập kết nối ban đầu. SSE được Ian Hickson đề cập lần đầu trong WHATWG Web Applications 1.0 vào năm 2004, và đến tháng 9 năm 2006, trình duyệt Opera triển khai tính năng thực nghiệm với tên gọi Server-Sent Events. Với SSE, máy khách thiết lập một kết nối HTTP với máy chủ, sử dụng trường tiêu đề `Accept: text/event-stream` (tức phản hồi phải có kiểu MIME `event-stream`), và giữ kết nối mở để máy chủ liên tục gửi các thông điệp. Mỗi thông điệp bao gồm một hoặc nhiều dòng, mỗi dòng kết thúc bằng ký tự xuống dòng, và thông điệp được kết thúc bằng một dòng trống. Dòng thông điệp có dạng ”tên trường: giá trị”; các dòng bắt đầu bằng dấu hai chấm (:) hoặc không có dấu hai chấm sẽ bị bỏ qua. Giao thức định nghĩa ba trường chính: `event`, `id`, và `data`, trong đó `data` là bắt buộc, còn các trường khác là tùy chọn. Nếu thông điệp có nhiều dòng `data`, giá trị của trường `data` sẽ là sự kết hợp của tất cả các giá trị trên các dòng đó, với tối đa một dòng `event` và `id` cho mỗi thông điệp.

Trong chatbot của chúng ta, SSE được áp dụng để truyền tin nhắn từ hệ thống tới người dùng, đảm bảo người dùng có thể nhận được tin nhắn sớm nhất có thể và có trải nghiệm tốt nhất.

1.4. Tổng kết chương

Chương 1 đã trình bày các kiến thức nền tảng quan trọng làm cơ sở cho việc phát triển hệ thống chatbot hỗ trợ nghiệp vụ đào tạo đại học. Nội dung chương bao gồm tổng quan về trí tuệ nhân tạo, mô hình ngôn ngữ lớn (LLM), cơ sở dữ liệu quan hệ, bộ phân tích cú pháp SQL và giao thức Server-Sent Events (SSE). Những kiến thức này không chỉ giúp hiểu rõ các công nghệ và nguyên lý được sử dụng trong hệ thống mà còn làm rõ vai trò của từng thành phần trong việc giải quyết bài toán truy xuất thông tin tự động, an toàn và hiệu quả. Các khái niệm và công nghệ được trình bày trong chương này sẽ là nền tảng cho các chương tiếp theo về phân tích, thiết kế và triển khai hệ thống.

Chương 2.

Đặc tả yêu cầu

Trong phần này, chúng tôi giới thiệu mục tiêu của chương, tập trung vào nhu cầu phát triển một hệ thống chatbot hỗ trợ nghiệp vụ đào tạo đại học thông qua truy xuất thông tin từ cơ sở dữ liệu quan hệ. Chương sẽ trình bày bài toán, mô tả nghiệp vụ, đề xuất giải pháp phần mềm, và đặc tả các yêu cầu của hệ thống. Cấu trúc chương bao gồm: phát biểu bài toán, mô tả nghiệp vụ đào tạo, giải pháp phần mềm, đặc tả yêu cầu, và tổng kết.

2.1. Mô tả nghiệp vụ tổ chức đào tạo đại học

Phần này phân tích các nghiệp vụ liên quan đến đào tạo đại học, tập trung vào các hoạt động truy vấn thông tin của sinh viên và giảng viên, từ đó xác định các khó khăn trong việc truy xuất dữ liệu từ cơ sở dữ liệu quan hệ và đề xuất giải pháp phần mềm. Nội dung bao gồm phân tích nghiệp vụ, phát biểu bài toán, và mô tả giải pháp chatbot thông minh hỗ trợ truy vấn.

2.1.1. Phân tích nghiệp vụ của sinh viên và giảng viên

Trong môi trường đào tạo đại học, sinh viên và giảng viên thực hiện các nghiệp vụ liên quan đến truy vấn thông tin từ cơ sở dữ liệu quan hệ, chẳng hạn như tra cứu thời khóa biểu, điểm số, hoặc danh sách sinh viên. Tuy nhiên, các hoạt động này gặp nhiều thách thức về tính thân thiện, hiệu quả, và truy cập dữ liệu, đặc biệt khi người dùng không có kỹ năng kỹ thuật.

2.1.1.1. Từ khía cạnh sinh viên

Sinh viên thường xuyên thực hiện các hoạt động truy vấn để hỗ trợ học tập và quản lý thông tin cá nhân, bao gồm:

– *Tra cứu thời khóa biểu*: Kiểm tra lịch học, phòng học, và thời gian của các môn học trong tuần hoặc học kỳ. Thông tin này có thể thay đổi đột xuất, ví dụ như đổi phòng học hoặc điều chỉnh giờ học.

– *Kiểm tra điểm số và tiến độ học tập*: Tra cứu điểm thi, hoặc số tín chỉ đã hoàn thành để lập kế hoạch học tập hoặc xác nhận điều kiện tốt nghiệp.

– *Tra cứu thông tin môn học đã đăng ký*: Xem danh sách môn học đã đăng ký, điều kiện tiên quyết, hoặc lịch trình liên quan để đảm bảo không trùng lịch.

– *Tra cứu học bổng và tài chính*: Kiểm tra thông tin về học bổng, mức học phí, lịch thanh toán, hoặc các hỗ trợ tài chính.

Các hoạt động này gặp phải những khó khăn sau:

– *Phân tán thông tin*: Thời khóa biểu, điểm số, và thông báo được lưu trữ trên nhiều kênh như cổng thông tin đào tạo, email, hoặc bảng tin khoa. Ví dụ, sinh viên có thể phải kiểm tra cả email và cổng thông tin để xác nhận lịch thi, gây mất thời gian và nhầm lẫn.

– *Giao diện không thân thiện*: Cổng thông tin yêu cầu nhập mã môn học, mã sinh viên, hoặc sử dụng các bộ lọc phức tạp, gây khó khăn cho sinh viên mới hoặc không quen công nghệ.

– *Thiếu hỗ trợ tức thời*: Khi có thắc mắc, ví dụ "Môn học hôm nay có đổi phòng không?", sinh viên phải liên hệ phòng đào tạo hoặc chờ thông báo, dẫn đến chậm trễ.

– *Hạn chế truy vấn linh hoạt*: Hệ thống hiện tại không hỗ trợ câu hỏi tự do bằng ngôn ngữ tự nhiên, như "Tuần này tôi học môn của các thầy nào?", buộc sinh viên sử dụng các mẫu truy vấn cố định.

2.1.1.2. Từ khía cạnh giảng viên

Giảng viên thực hiện các hoạt động truy vấn để quản lý lớp học và hỗ trợ sinh viên, bao gồm:

– *Tra cứu lịch giảng dạy*: Kiểm tra lịch dạy, phòng học, và thời gian để chuẩn bị bài giảng hoặc điều chỉnh kế hoạch.

– *Tra cứu thông tin sinh viên*: Xem danh sách sinh viên, điểm số, và các thông tin liên quan

– *Kiểm tra thông tin môn học*: Xác nhận thông tin về môn học, số lượng sinh viên để phối hợp với phòng đào tạo.

– *Tư vấn học tập*: Tra cứu dữ liệu sinh viên, như số tín chỉ hoặc môn học đã hoàn thành, để đưa ra lời khuyên về kế hoạch học tập hoặc định hướng nghề nghiệp.

– *Tra cứu thông báo và thay đổi*: Kiểm tra các cập nhật về lịch dạy, phòng học từ phòng đào tạo.

Các hoạt động này gặp phải những khó khăn sau:

– *Nhiều hệ thống riêng lẻ*: Giảng viên phải sử dụng nhiều công cụ, như cổng thông tin, email, hoặc phần mềm quản lý lớp, dẫn đến thiếu đồng bộ và mất thời gian.

– *Truy cập dữ liệu chậm*: Tra cứu danh sách sinh viên, lịch sử điểm số, hoặc lịch dạy yêu cầu nhiều bước, như đăng nhập, chọn lớp, nhập mã môn học, làm giảm hiệu quả, đặc biệt trong các tình huống cần thông tin nhanh.

– *Khối lượng câu hỏi lặp lại*: Giảng viên nhận nhiều câu hỏi trùng lặp từ sinh viên, như "Lịch thi khi nào?" hoặc "Bài tập tuần này là gì?", làm tăng khối lượng công việc không cần thiết.

2.1.2. Phát biểu bài toán

Dựa trên phân tích nghiệp vụ, việc truy vấn thông tin từ cơ sở dữ liệu quan hệ trong đào tạo đại học đối mặt với nhiều thách thức, đặc biệt về tính thân thiện, hiệu quả, và bảo mật. Các hệ thống hiện tại không đáp ứng được nhu cầu của sinh viên và giảng viên, đặc biệt là những người không có kỹ năng kỹ thuật, đồng thời thiếu khả năng xử lý linh hoạt và đảm bảo an toàn dữ liệu.

2.1.2.1. Một số thách thức đặt ra

– *Thiếu tính linh hoạt trong truy vấn dữ liệu*: Các hệ thống hiện tại chỉ hỗ trợ truy vấn dựa trên các mẫu logic được lập trình sẵn, như chọn mã môn học hoặc mã sinh viên từ danh sách. Người dùng không thể đặt câu hỏi tự do, như "Lịch học tuần này của

tôi là gì?" hoặc "Sinh viên nào chưa nộp bài tập?", vì hệ thống không được thiết kế để xử lý các truy vấn ngoài logic cố định. Tuy nhiên, nếu cho phép người dùng chạy trực tiếp truy vấn SQL SELECT, hệ thống sẽ đối mặt với rủi ro bảo mật, như tấn công tiêm nhiễm SQL hoặc truy cập trái phép.

– *Phân tán và thiếu đồng bộ thông tin*: Dữ liệu được lưu trữ trong cơ sở dữ liệu quan hệ nhưng phân phối qua nhiều kênh, như cổng thông tin, email, hoặc bảng tin, gây không nhất quán. Ví dụ, sinh viên có thể thấy lịch thi trên email khác với cổng thông tin do chưa cập nhật.

– *Hiệu quả thấp trong truy vấn*: Tra cứu thông tin đòi hỏi nhiều bước thủ công hoặc xác thực, làm giảm năng suất. Ví dụ, giảng viên cần nhiều thời gian để tra cứu danh sách sinh viên hoặc kiểm tra lịch dạy từ các nguồn khác nhau.

– *Thiếu hỗ trợ ngôn ngữ tự nhiên*: Hệ thống hiện tại không cho phép đặt câu hỏi tự do bằng tiếng Việt, như "Lịch học tuần này của tôi là gì?" hoặc "Sinh viên nào có điểm thấp không đủ điều kiện thi cuối kỳ?", buộc người dùng sử dụng các mẫu truy vấn cố định, làm giảm tính linh hoạt.

2.1.2.2. Nhu cầu cấp thiết

Cần một giải pháp tích hợp, cho phép sinh viên và giảng viên truy vấn thông tin từ cơ sở dữ liệu quan hệ một cách nhanh chóng, thân thiện, và an toàn. Giải pháp phải vượt qua rào cản kỹ thuật, hỗ trợ truy vấn bằng ngôn ngữ tự nhiên, đảm bảo phân quyền rõ ràng, và tối ưu hóa hiệu suất để đáp ứng nhiều người dùng đồng thời. Những yêu cầu này đặt nền tảng cho hệ thống chatbot thông minh, được trình bày chi tiết trong phần tiếp theo.

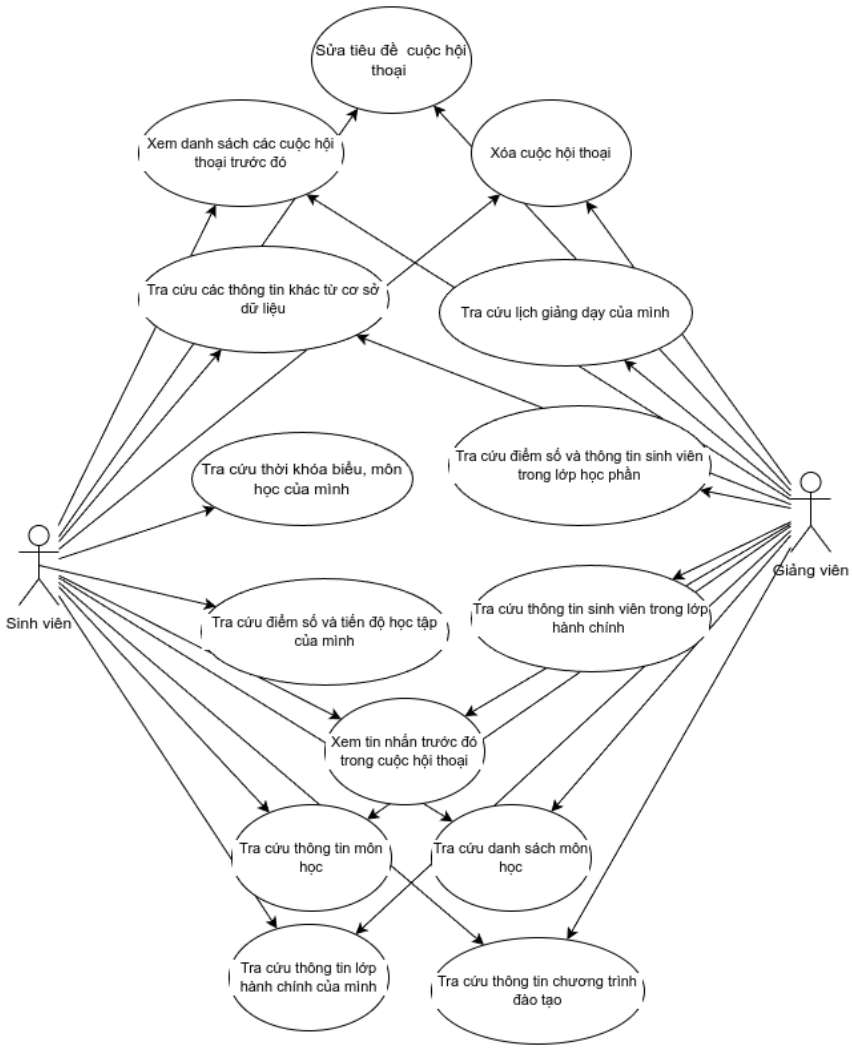
2.1.3. Giải pháp phần mềm

Để giải quyết các thách thức về tính linh hoạt, hiệu quả, và bảo mật trong việc truy vấn thông tin từ cơ sở dữ liệu quan hệ trong đào tạo đại học, khóa luận đề xuất phát triển một hệ thống chatbot thông minh, tận dụng sức mạnh của mô hình ngôn ngữ lớn (LLM) để hỗ trợ sinh viên và giảng viên tra cứu dữ liệu một cách tự nhiên và an toàn. Hệ thống được thiết kế như một ứng dụng web, cho phép người dùng truy cập thông qua bất kỳ

thiết bị có kết nối internet và trình duyệt web, chẳng hạn như máy tính hoặc điện thoại thông minh. Với giao diện được xây dựng bằng React, hệ thống mang lại trải nghiệm thân thiện, hỗ trợ người dùng đặt câu hỏi bằng ngôn ngữ tự nhiên tiếng Việt, ví dụ như "Lịch học tuần này của tôi là gì?" hoặc "Danh sách sinh viên không đủ điểm qua môn tại môn Cấu trúc dữ liệu và giải thuật?". Người dùng có thể hỏi bất cứ câu hỏi nào mà dữ liệu có trong cơ sở dữ liệu và quyền hạn của họ cho phép. Chatbot sẽ phân tích câu hỏi và tự động chuyển đổi thành các truy vấn SQL hoặc gọi các hàm được định nghĩa trước (function calling) để lấy ra dữ liệu cần thiết và trả về cho người dùng dưới dạng ngôn ngữ tự nhiên thân thiện và dễ hiểu. Việc sử dụng cả 2 cơ chế này giúp hệ thống vừa giữ được sự linh hoạt với mọi loại câu hỏi khác nhau lại vừa đảm bảo chính xác trong những nghiệp vụ quan trọng. Phần backend, được phát triển bằng Golang, đảm bảo hiệu suất cao, gọn nhẹ có thể phục vụ được nhiều yêu cầu cùng một lúc. Ngoài ra Golang cũng là một ngôn ngữ thân thiện với môi trường điện toán đám mây, điều này giúp chúng ta có thể dễ dàng triển khai hệ thống máy chủ lên đám mây nếu cần thiết trong tương lai, đảm bảo khả năng mở rộng dễ dàng nhưng vẫn tiết kiếm tri phí. Để duy trì bảo mật, hệ thống áp dụng xác thực và phân quyền bằng JWT token đã được chứng minh là 1 phương pháp bền bỉ và được sử dụng rộng rãi trong các doanh nghiệp. Ngoài ra với sự xuất hiện bởi SQL sinh ra bởi LLM, hệ thống cũng sử dụng postgres parser để có thể phân tích câu truy vấn và phân quyền phù hợp. Hệ thống sẽ tập trung hộ trỡ các câu truy vấn SELECT và truy xuất dữ liệu thay vì tạo mới và sửa đổi dữ liệu vì việc thay đổi dữ liệu nên được thực hiện một cách cẩn thận với nhiều bước xác thực khác nhau để đảm bảo tính toàn vẹn và ổn định của dữ liệu. Việc thay đổi hay tạo mới dữ liệu dường như không phù hợp với ngôn ngữ tự nhiên dựa trên các khía cạnh về bảo mật và tính bền vững của dữ liệu. Bên cạnh đó, chatbot cung cấp phản hồi tức thì cho các thắc mắc thường gặp, như thay đổi lịch học, phòng học, các môn mở kỳ này,... vào bất cứ thời điểm nào trong ngày mà không cần đến gặp trực tiếp phòng đào tạo, từ đó giảm áp lực cho phòng đào tạo và nâng cao hiệu quả vận hành. Với khả năng hỗ trợ đa dạng các nghiệp vụ truy vấn, từ tra cứu thủ tục học tập đến quản lý thông tin lớp học, hệ thống không chỉ cải thiện trải nghiệm người dùng mà còn mở ra tiềm năng ứng dụng trong các lĩnh vực khác, như quản lý hành chính hoặc dịch vụ khách hàng, nhờ tính linh hoạt và khả năng mở rộng của nó.

2.2. Đặc tả yêu cầu

Phần này trình bày các yêu cầu chức năng và phi chức năng của hệ thống chatbot, bao gồm mô hình ca sử dụng và các yêu cầu bổ sung.



Hình 2.1. Mô hình ca sử dụng

2.2.1. Mô hình ca sử dụng

Để hiểu rõ cách hệ thống chatbot hỗ trợ các chức năng chính cho người dùng, phần này sẽ trình bày các yêu cầu chức năng chính đối với các người dùng (actor), bỏ qua các chức năng cơ bản như đăng nhập, đăng xuất. Trước hết, ta cần xem xét mô hình ca sử dụng tổng quát được minh họa trong Hình 2.1, nơi các ca sử dụng được thiết kế để phản

ánh nhu cầu thực tế của sinh viên và giảng viên. Hình 2.1 cung cấp cái nhìn tổng quan về các ca sử dụng riêng biệt cho từng vai trò, cùng với một số ca sử dụng chung, đặt nền tảng cho việc phân tích chi tiết từng ca sử dụng trong các bảng bên dưới.

2.2.2. Ca sử dụng của sinh viên

Để bắt đầu, phần này tập trung vào các ca sử dụng dành riêng cho sinh viên, nơi hệ thống chatbot cung cấp các dịch vụ quan trọng đáp ứng nhu cầu học tập và quản lý cá nhân của họ. Dưới đây là các ca sử dụng chi tiết, được trình bày thông qua các bảng mô tả quy trình cụ thể, bắt đầu với việc tra cứu thông tin học tập cá nhân.

2.2.2.1. Tra cứu thời khóa biểu, môn học cá nhân

Với mục tiêu hỗ trợ sinh viên trong việc quản lý lịch học và môn học, ca sử dụng này mô tả cách chatbot xử lý yêu cầu tra cứu thông tin cá nhân. Chi tiết được trình bày trong bảng dưới đây.

Bảng 2.1. Ca sử dụng tra cứu thời khóa biểu, môn học cá nhân

Ca sử dụng	Tra cứu thời khóa biểu, môn học cá nhân
Mô tả	Sinh viên tra cứu lịch học cá nhân (môn học, thời gian, địa điểm) và thông tin về các môn học đã đăng ký (tên môn, số tín chỉ, giảng viên) cho một kỳ học hoặc khoảng thời gian cụ thể.
Tác nhân chính	Sinh viên
Điều kiện trước	Sinh viên đã đăng nhập vào hệ thống chatbot qua giao diện web.
Điều kiện sau	Hệ thống trả về thông tin thời khóa biểu hoặc môn học nếu thành công, hoặc thông báo lỗi nếu không tìm thấy thông tin.
Luồng cơ bản	1. Sinh viên nhập câu hỏi, ví dụ: "Lịch học tuần này của tôi là gì?" hoặc "Môn học kỳ này của tôi là gì?". 2. Hệ thống gọi LLM để phân tích câu hỏi và xác định cần truy vấn SQL hay sử dụng hàm có sẵn. 3. Hệ thống thực thi hàm hoặc truy vấn SQL tương ứng để lấy dữ liệu. 4. Hệ thống chuyển kết quả cho LLM để tạo câu trả lời tự nhiên. 5. Hệ thống truyền câu trả lời về giao diện để hiển thị cho sinh viên.
Luồng thay thế	Nếu câu hỏi không rõ ràng, không hợp lệ, hoặc ngoài quyền truy cập, hệ thống trả về thông báo lỗi do LLM tạo ra.
Yêu cầu đặc biệt	Chỉ hiển thị thời khóa biểu và môn học của chính sinh viên.
Mở rộng	Không

2.2.2.2. Tra cứu điểm số và tiến độ học tập

Tiếp nối với các nhu cầu học tập quan trọng khác, ca sử dụng này tập trung vào việc hỗ trợ sinh viên tra cứu điểm số và tiến độ học tập cá nhân, nhằm cung cấp thông tin chi tiết về thành tích học tập của họ. Các bước cụ thể và yêu cầu được trình bày trong bảng Bảng**??**.

Bảng 2.2. Ca sử dụng tra cứu điểm số và tiến độ học tập

Ca sử dụng	Tra cứu điểm số và tiến độ học tập
Mô tả	Sinh viên tra cứu điểm số cá nhân (điểm môn học, điểm trung bình tích lũy) và tiến độ học tập (số tín chỉ đã hoàn thành, yêu cầu tốt nghiệp) cho một kỳ học hoặc toàn bộ chương trình học.
Tác nhân chính	Sinh viên
Điều kiện trước	Sinh viên đã đăng nhập vào hệ thống chatbot qua giao diện web.
Điều kiện sau	Hệ thống trả về thông tin điểm số và tiến độ học tập nếu thành công, hoặc thông báo lỗi nếu không tìm thấy thông tin.
Luồng cơ bản	1. Sinh viên nhập câu hỏi, ví dụ: "Điểm môn Cấu trúc dữ liệu và giải thuật của tôi là bao nhiêu?" hoặc "Tôi đã hoàn thành bao nhiêu tín chỉ?". 2. Hệ thống gọi LLM để phân tích câu hỏi và xác định truy vấn SQL hoặc hàm có sẵn. 3. Hệ thống thực thi hàm hoặc truy vấn SQL để lấy dữ liệu điểm số và tiến độ. 4. Hệ thống chuyển kết quả cho LLM để tạo câu trả lời tự nhiên. 5. Hệ thống hiển thị câu trả lời trên giao diện web cho sinh viên.
Luồng thay thế	Nếu câu hỏi không rõ ràng, không hợp lệ, hoặc ngoài quyền truy cập, hệ thống trả về thông báo lỗi do LLM tạo ra.
Yêu cầu đặc biệt	Chỉ hiển thị điểm số và tiến độ học tập của chính sinh viên.
Mở rộng	Không

2.2.3. Ca sử dụng của giảng viên

Tiếp theo, phần này tập trung vào các ca sử dụng dành riêng cho giảng viên, nơi hệ thống chatbot hỗ trợ các chức năng cần thiết để quản lý công việc giảng dạy và tương tác với sinh viên. Dưới đây là các ca sử dụng chi tiết, được trình bày thông qua các bảng mô tả quy trình cụ thể, bắt đầu với việc tra cứu lịch giảng dạy cá nhân.

2.2.3.1. Tra cứu lịch giảng dạy

Nhằm hỗ trợ giảng viên trong việc quản lý lịch trình công việc, ca sử dụng này mô tả cách chatbot xử lý yêu cầu tra cứu lịch giảng dạy cá nhân. Chi tiết được trình bày trong Bảng 2.3.

Bảng 2.3. Ca sử dụng tra cứu lịch giảng dạy

Ca sử dụng	Tra cứu lịch giảng dạy
Mô tả	Giảng viên tra cứu lịch giảng dạy cá nhân, bao gồm thông tin về môn học, thời gian, địa điểm, và loại tiết (lý thuyết hoặc thực hành) cho một khoảng thời gian cụ thể (ngày, tuần, hoặc kỳ).
Tác nhân chính	Giảng viên
Điều kiện trước	Giảng viên đã đăng nhập vào hệ thống chatbot qua giao diện web.
Điều kiện sau	Hệ thống trả về lịch giảng dạy chi tiết nếu thành công, hoặc thông báo lỗi nếu không tìm thấy thông tin.
Luồng cơ bản	1. Giảng viên nhập câu hỏi, ví dụ: "Lịch giảng dạy tuần này của tôi là gì?". 2. Hệ thống gọi LLM để phân tích câu hỏi và xác định cần truy vấn SQL hay sử dụng hàm có sẵn. 3. Hệ thống thực thi hàm hoặc truy vấn SQL tương ứng để lấy dữ liệu. 4. Hệ thống chuyển kết quả cho LLM để tạo câu trả lời tự nhiên. 5. Hệ thống truyền câu trả lời về giao diện để hiển thị cho giảng viên. 6. Hệ thống sử dụng Server-Sent Events (SSE) để gửi thông báo cập nhật nếu có thay đổi trong lịch giảng dạy.
Luồng thay thế	Nếu câu hỏi không rõ ràng, không hợp lệ, hoặc ngoài quyền truy cập, hệ thống trả về thông báo lỗi do LLM tạo ra.
Yêu cầu đặc biệt	Chỉ hiển thị lịch giảng dạy của chính giảng viên.
Mở rộng	Không

2.2.3.2. Tra cứu điểm số và thông tin sinh viên lớp học phần

Sau khi hỗ trợ quản lý lịch giảng dạy, ca sử dụng này mở rộng chức năng của chatbot để hỗ trợ giảng viên trong việc tra cứu thông tin chi tiết về sinh viên và điểm số của các lớp học phần mà họ phụ trách. Các bước thực hiện và yêu cầu cụ thể được trình bày trong Bảng 2.4.

Bảng 2.4. Ca sử dụng tra cứu điểm số và thông tin sinh viên lớp học phần

Ca sử dụng	Tra cứu điểm số và thông tin sinh viên lớp học phần
Mô tả	Giảng viên tra cứu thông tin về sinh viên trong các lớp học phần mà họ giảng dạy, bao gồm tên, mã số sinh viên, và điểm số, để quản lý lớp học và đánh giá học tập.
Tác nhân chính	Giảng viên
Điều kiện trước	Giảng viên đã đăng nhập vào hệ thống chatbot qua giao diện web.
Điều kiện sau	Hệ thống trả về thông tin và điểm số của sinh viên nếu thành công, hoặc thông báo lỗi nếu không tìm thấy thông tin.
Luồng cơ bản	1. Giảng viên nhập câu hỏi, ví dụ: "Ai là sinh viên trong lớp INT1032 của tôi?" hoặc "Điểm số của sinh viên trong lớp INT1032 là gì?". 2. Hệ thống gọi LLM để phân tích câu hỏi và xác định cần truy vấn SQL hay sử dụng hàm có sẵn. 3. Hệ thống thực thi hàm hoặc truy vấn SQL tương ứng để lấy dữ liệu. 4. Hệ thống chuyển kết quả cho LLM để tạo câu trả lời tự nhiên. 5. Hệ thống truyền câu trả lời về giao diện để hiển thị cho giảng viên.
Luồng thay thế	Nếu câu hỏi không rõ ràng, không hợp lệ, hoặc ngoài quyền truy cập, hệ thống trả về thông báo lỗi do LLM tạo ra.
Yêu cầu đặc biệt	Chỉ hiển thị thông tin và điểm số của sinh viên trong các lớp học mà giảng viên giảng dạy.
Mở rộng	Không

2.2.3.3. Tra cứu thông tin sinh viên trong lớp hành chính

Tiếp tục mở rộng vai trò hỗ trợ giảng viên, ca sử dụng này tập trung vào việc hỗ trợ giảng viên cố vấn trong việc tra cứu thông tin chi tiết về sinh viên trong lớp hành chính mà họ phụ trách, bao gồm hầu hết các thông tin của sinh viên để có thể đưa ra lời khuyên cụ thể và kịp thời. Chi tiết được trình bày trong Bảng 2.5.

Bảng 2.5. Ca sử dụng tra cứu thông tin sinh viên trong lớp hành chính

Ca sử dụng	Tra cứu thông tin sinh viên trong lớp hành chính
Mô tả	Giảng viên cố vấn tra cứu thông tin chi tiết về sinh viên trong lớp hành chính mà họ phụ trách, bao gồm tên, mã số sinh viên, email, giới tính, ngày sinh, điểm số các môn học, học bổng, và học phí, để hỗ trợ quản lý và tư vấn học tập toàn diện.
Tác nhân chính	Giảng viên
Điều kiện trước	Giảng viên đã đăng nhập vào hệ thống chatbot qua giao diện web và là cố vấn của lớp hành chính.
Điều kiện sau	Hệ thống trả về thông tin chi tiết của sinh viên nếu thành công, hoặc thông báo lỗi nếu không tìm thấy thông tin.
Luồng cơ bản	1. Giảng viên nhập câu hỏi, ví dụ: "Sinh viên trong lớp hành chính của tôi là ai?", "Điểm số của sinh viên X là gì?". 2. Hệ thống gọi LLM để phân tích câu hỏi và xác định cần truy vấn SQL hay sử dụng hàm có sẵn. 3. Hệ thống thực thi hàm hoặc truy vấn SQL tương ứng để lấy dữ liệu. 4. Hệ thống chuyển kết quả cho LLM để tạo câu trả lời tự nhiên. 5. Hệ thống truyền câu trả lời về giao diện để hiển thị cho giảng viên.
Luồng thay thế	Nếu câu hỏi không rõ ràng, không hợp lệ, hoặc ngoài quyền truy cập, hệ thống trả về thông báo lỗi do LLM tạo ra.
Yêu cầu đặc biệt	Chỉ hiển thị thông tin chi tiết của sinh viên trong lớp hành chính mà giảng viên là cố vấn, bao gồm điểm số, học bổng, và học phí.
Mở rộng	Không

2.2.4. Ca sử dụng chung cho giảng viên và sinh viên

Phần này trình bày các ca sử dụng được chia sẻ giữa giảng viên và sinh viên, nơi hệ thống chatbot cung cấp các chức năng cơ bản và nâng cao để hỗ trợ tương tác và truy cập thông tin. Các ca sử dụng được tổ chức theo luồng từ quản lý hội thoại cá nhân, đến quản lý nội dung hội thoại, và cuối cùng là tra cứu thông tin học tập, được chi tiết hóa qua các bảng dưới đây. Bắt đầu với các chức năng liên quan đến việc xem danh sách hội thoại cá nhân.

2.2.4.1. Xem danh sách hội thoại cá nhân

Nhằm hỗ trợ người dùng trong việc theo dõi các tương tác trước đó, ca sử dụng này mô tả cách chatbot hiển thị danh sách hội thoại cá nhân. Chi tiết được trình bày trong Bảng 2.6.

Bảng 2.6. Ca sử dụng xem danh sách hội thoại cá nhân

Ca sử dụng	Xem danh sách hội thoại cá nhân
Mô tả	Người dùng xem danh sách các hội thoại đã tạo trong hệ thống chatbot.
Tác nhân chính	Sinh viên, Giảng viên
Điều kiện trước	Người dùng đã đăng nhập vào hệ thống chatbot.
Điều kiện sau	Hệ thống hiển thị danh sách hội thoại của người dùng.
Luồng cơ bản	1. Người dùng chọn tính năng xem danh sách hội thoại. 2. Hệ thống truy vấn cơ sở dữ liệu để lấy danh sách hội thoại của người dùng. 3. Hệ thống hiển thị danh sách hội thoại trên giao diện.
Luồng thay thế	Nếu không có hội thoại, hệ thống hiển thị thông báo không có hội thoại.
Yêu cầu đặc biệt	Chỉ hiển thị hội thoại của người dùng hiện tại.
Mở rộng	Không

2.2.4.2. Xem lịch sử tin nhắn

Tiếp nối việc quản lý hội thoại, ca sử dụng này cho phép người dùng xem lại lịch sử tin nhắn của một hội thoại cụ thể, hỗ trợ việc theo dõi các cuộc trao đổi trước đây. Chi tiết được trình bày trong Bảng 2.7.

Bảng 2.7. Ca sử dụng xem lịch sử tin nhắn

Ca sử dụng	Xem lịch sử tin nhắn
Mô tả	Người dùng xem lại lịch sử tin nhắn của một hội thoại cụ thể trong hệ thống chatbot.
Tác nhân chính	Sinh viên, Giảng viên
Điều kiện trước	Người dùng đã đăng nhập và có ít nhất một hội thoại.
Điều kiện sau	Hệ thống hiển thị lịch sử tin nhắn của hội thoại được chọn.
Luồng cơ bản	1. Người dùng chọn hội thoại cần xem lịch sử. 2. Hệ thống truy vấn cơ sở dữ liệu để lấy lịch sử tin nhắn. 3. Hệ thống hiển thị lịch sử tin nhắn trên giao diện.
Luồng thay thế	Nếu hội thoại không có tin nhắn, hệ thống thông báo không có dữ liệu.
Yêu cầu đặc biệt	Chỉ hiển thị tin nhắn của hội thoại thuộc về người dùng hiện tại.
Mở rộng	Không

2.2.4.3. Sửa tiêu đề hội thoại

Chuyển sang quản lý nội dung hội thoại, ca sử dụng này hỗ trợ người dùng chỉnh sửa tiêu đề của một hội thoại hiện có để tổ chức tốt hơn các tương tác. Chi tiết được trình bày trong Bảng 2.8.

Bảng 2.8. Ca sử dụng sửa tiêu đề hội thoại

Ca sử dụng	Sửa tiêu đề hội thoại
Mô tả	Người dùng chỉnh sửa tiêu đề của một hội thoại hiện có trong hệ thống chatbot.
Tác nhân chính	Sinh viên, Giảng viên
Điều kiện trước	Người dùng đã đăng nhập và có ít nhất một hội thoại.
Điều kiện sau	Tiêu đề hội thoại được cập nhật thành công hoặc thông báo lỗi nếu thất bại.
Luồng cơ bản	1. Người dùng nhấp vào hội thoại để sửa tiêu đề. 2. Người dùng xác nhận chỉnh sửa tiêu đề. 3. Hệ thống cập nhật tiêu đề trong cơ sở dữ liệu. 4. Hệ thống thông báo cập nhật thành công.
Luồng thay thế	Nếu người dùng hủy xác nhận, hệ thống giữ nguyên tiêu đề hiện tại.
Yêu cầu đặc biệt	Chỉ cho phép chỉnh sửa hội thoại của người dùng hiện tại.
Mở rộng	Không

2.2.4.4. Xóa hội thoại

Tiếp tục với quản lý nội dung, ca sử dụng này cho phép người dùng xóa một hội thoại không còn cần thiết để tối ưu hóa không gian lưu trữ. Chi tiết được trình bày trong Bảng 2.9.

Bảng 2.9. Ca sử dụng xóa hội thoại

Ca sử dụng	Xóa hội thoại
Mô tả	Người dùng xóa một hội thoại hiện có trong hệ thống chatbot.
Tác nhân chính	Sinh viên, Giảng viên
Điều kiện trước	Người dùng đã đăng nhập và có ít nhất một hội thoại.
Điều kiện sau	Hội thoại được xóa khỏi hệ thống hoặc thông báo lỗi nếu thất bại.
Luồng cơ bản	1. Người dùng nhấp vào hội thoại để xóa. 2. Người dùng xác nhận xóa. 3. Hệ thống xóa hội thoại khỏi cơ sở dữ liệu. 4. Hệ thống thông báo xóa thành công.
Luồng thay thế	Nếu người dùng hủy xác nhận, hệ thống giữ nguyên hội thoại.
Yêu cầu đặc biệt	Chỉ cho phép xóa hội thoại của người dùng hiện tại.
Mở rộng	Không

2.2.4.5. Tra cứu thông tin môn học cụ thể

Chuyển sang các chức năng tra cứu thông tin, ca sử dụng này hỗ trợ người dùng tìm kiếm chi tiết về các môn học cụ thể, phục vụ nhu cầu học tập và giảng dạy. Chi tiết được trình bày trong Bảng 2.10.

Bảng 2.10. Ca sử dụng tra cứu thông tin môn học cụ thể

Ca sử dụng	Tra cứu thông tin môn học cụ thể
Mô tả	Người dùng (giảng viên hoặc sinh viên) tra cứu thông tin về các môn học cụ thể, bao gồm thời gian, địa điểm, và giảng viên giảng dạy cho các lớp học hoặc kỳ học.
Tác nhân chính	Giảng viên, Sinh viên
Điều kiện trước	Người dùng đã đăng nhập vào hệ thống chatbot qua giao diện web.
Điều kiện sau	Hệ thống trả về thông tin môn học nếu thành công, hoặc thông báo lỗi nếu không tìm thấy thông tin.
Luồng cơ bản	1. Người dùng nhập câu hỏi, ví dụ: "Lớp INT1032 1 học khi nào và ở đâu?" hoặc "Ai dạy môn INT1032 1 kỳ này?". 2. Hệ thống gọi LLM để phân tích câu hỏi và xác định cần truy vấn SQL hay sử dụng hàm có sẵn. 3. Hệ thống thực thi hàm hoặc truy vấn SQL tương ứng để lấy dữ liệu. 4. Hệ thống chuyển kết quả cho LLM để tạo câu trả lời tự nhiên. 5. Hệ thống truyền câu trả lời về giao diện để hiển thị cho người dùng.
Luồng thay thế	Nếu câu hỏi không rõ ràng, không hợp lệ, hoặc ngoài quyền truy cập, hệ thống trả về thông báo lỗi do LLM tạo ra.
Yêu cầu đặc biệt	Không có yêu cầu đặc biệt.
Mở rộng	Không

2.2.4.6. Tra cứu danh sách môn học mở

Tiếp theo, ca sử dụng này hỗ trợ người dùng tra cứu danh sách các môn học được mở, phục vụ việc lập kế hoạch học tập hoặc tư vấn. Chi tiết được trình bày trong Bảng 2.11.

Bảng 2.11. Ca sử dụng tra cứu danh sách môn học mở

Ca sử dụng	Tra cứu danh sách môn học mở
Mô tả	Người dùng (giảng viên hoặc sinh viên) tra cứu danh sách các môn học được mở trong một kỳ học cụ thể, bao gồm mã môn học, tên môn học, và số tín chỉ, để hỗ trợ lập kế hoạch đăng ký môn học hoặc tư vấn học tập.
Tác nhân chính	Giảng viên, Sinh viên
Điều kiện trước	Người dùng đã đăng nhập vào hệ thống chatbot qua giao diện web.
Điều kiện sau	Hệ thống trả về danh sách môn học nếu thành công, hoặc thông báo lỗi nếu không tìm thấy thông tin.
Luồng cơ bản	1. Người dùng nhập câu hỏi, ví dụ: "Kỳ này mở những môn học nào?" hoặc "Môn học nào có thể đăng ký trong kỳ 2 2025?". 2. Hệ thống gọi LLM để phân tích câu hỏi và xác định cần truy vấn SQL hay sử dụng hàm có sẵn. 3. Hệ thống thực thi hàm hoặc truy vấn SQL tương ứng để lấy dữ liệu. 4. Hệ thống chuyển kết quả cho LLM để tạo câu trả lời tự nhiên. 5. Hệ thống truyền câu trả lời về giao diện để hiển thị cho người dùng.
Luồng thay thế	Nếu câu hỏi không rõ ràng, không hợp lệ, hoặc ngoài quyền truy cập, hệ thống trả về thông báo lỗi do LLM tạo ra.
Yêu cầu đặc biệt	Không có yêu cầu đặc biệt.
Mở rộng	Không

2.2.4.7. Tra cứu thông tin chương trình đào tạo

Chuyển sang thông tin chi tiết hơn, ca sử dụng này hỗ trợ người dùng tra cứu thông tin về các chương trình đào tạo, phục vụ nhu cầu lập kế hoạch dài hạn. Chi tiết được trình bày trong Bảng 2.12.

Bảng 2.12. Ca sử dụng tra cứu thông tin chương trình đào tạo

Ca sử dụng	Tra cứu thông tin chương trình đào tạo
Mô tả	Người dùng (giảng viên hoặc sinh viên) tra cứu thông tin về các chương trình đào tạo, bao gồm tên chương trình, loại bằng, thời gian đào tạo, và các môn học liên quan.
Tác nhân chính	Giảng viên, Sinh viên
Điều kiện trước	Người dùng đã đăng nhập vào hệ thống chatbot qua giao diện web.
Điều kiện sau	Hệ thống trả về thông tin chương trình đào tạo nếu thành công, hoặc thông báo lỗi nếu không tìm thấy thông tin.
Luồng cơ bản	1. Người dùng nhập câu hỏi, ví dụ: "Chương trình CNTT gồm những môn học nào?" hoặc "Thời gian đào tạo của ngành Kỹ sư là bao lâu?". 2. Hệ thống gọi LLM để phân tích câu hỏi và xác định cần truy vấn SQL hay sử dụng hàm có sẵn. 3. Hệ thống thực thi hàm hoặc truy vấn SQL tương ứng để lấy dữ liệu. 4. Hệ thống chuyển kết quả cho LLM để tạo câu trả lời tự nhiên. 5. Hệ thống truyền câu trả lời về giao diện để hiển thị cho người dùng.
Luồng thay thế	Nếu câu hỏi không rõ ràng, không hợp lệ, hoặc ngoài quyền truy cập, hệ thống trả về thông báo lỗi do LLM tạo ra.
Yêu cầu đặc biệt	Không có yêu cầu đặc biệt.
Mở rộng	Không

2.2.4.8. Tra cứu thông tin khác trong cơ sở dữ liệu

Cuối cùng, ca sử dụng này cung cấp khả năng tra cứu các thông tin khác trong cơ sở dữ liệu, đảm bảo an toàn và quyền truy cập phù hợp. Chi tiết được trình bày trong Bảng 2.13.

Bảng 2.13. Ca sử dụng tra cứu thông tin khác trong cơ sở dữ liệu

Ca sử dụng	Tra cứu thông tin khác trong cơ sở dữ liệu
Mô tả	Người dùng (sinh viên hoặc giảng viên) tra cứu các thông tin khác trong cơ sở dữ liệu, đảm bảo chỉ truy cập dữ liệu trong quyền của mình. Ví dụ như ”môn Cấu trúc dữ liệu và giải thuật kỳ này có những thầy cô nào giảng dạy?”
Tác nhân chính	Sinh viên, Giảng viên
Điều kiện trước	Người dùng đã đăng nhập vào hệ thống chatbot qua giao diện web.
Điều kiện sau	Hệ thống trả về thông tin nếu thành công, hoặc thông báo lỗi nếu không tìm thấy thông tin hoặc không có quyền truy cập.
Luồng cơ bản	1. Người dùng nhập câu hỏi, ví dụ: . 2. Hệ thống gọi LLM để phân tích câu hỏi và xác định cần truy vấn SQL hay sử dụng hàm có sẵn. 3. Hệ thống xác nhận quyền truy cập của người dùng cho truy vấn trước khi thực thi. 4. Hệ thống thực thi hàm hoặc truy vấn SQL tương ứng để lấy dữ liệu. 5. Hệ thống chuyển kết quả cho LLM để tạo câu trả lời tự nhiên. 6. Hệ thống truyền câu trả lời về giao diện để hiển thị cho người dùng.
Luồng thay thế	Nếu câu hỏi không rõ ràng, không hợp lệ, hoặc ngoài quyền truy cập, hệ thống trả về thông báo lỗi do LLM tạo ra.
Yêu cầu đặc biệt	Đảm bảo chỉ truy cập dữ liệu trong quyền của người dùng.
Mở rộng	Không

2.2.5. Đặc tả yêu cầu bổ sung

Phần này trình bày các yêu cầu bổ sung liên quan đến chất lượng của hệ thống chatbot thông minh, nhằm đảm bảo hệ thống hoạt động hiệu quả, dễ sử dụng, an toàn, và đáng tin cậy cho giảng viên và sinh viên. Các yêu cầu này tập trung vào hiệu suất, tính dễ sử dụng, bảo mật, độ tin cậy, khả năng mở rộng, và khả năng tiếp cận, hỗ trợ việc tra cứu thông tin học tập một cách thuận tiện và nhanh chóng.

Thời gian phản hồi nhanh Hệ thống chatbot phải trả lời các câu hỏi của người dùng, như tra cứu lịch học hoặc thông tin môn học, trong vòng 5 giây đối với hầu hết các yêu cầu. Điều này đảm bảo giảng viên và sinh viên nhận được thông tin nhanh chóng.

Hiểu câu hỏi chính xác Hệ thống phải hiểu chính xác các câu hỏi bằng tiếng Việt, kể cả khi người dùng sử dụng các cách diễn đạt khác nhau, chẳng hạn "Lịch học tuần này" hoặc "Thời khóa biểu tuần này". Hệ thống cần đạt độ chính xác cao lớn hơn 70% với mọi câu hỏi và lớn hơn 90% với các câu phổ biến, để đảm bảo người dùng nhận được câu trả lời đúng với ý định của họ.

Giao diện thân thiện với người dùng Giao diện web của chatbot phải dễ sử dụng, với thiết kế rõ ràng và hướng dẫn đơn giản, phù hợp cho cả sinh viên năm nhất và giảng viên lớn tuổi có ít kinh nghiệm kỹ thuật. Người dùng cần dễ dàng nhập câu hỏi và nhận câu trả lời mà không gặp khó khăn trong việc điều hướng.

Bảo mật thông tin Hệ thống phải bảo vệ thông tin nhạy cảm, như điểm số của sinh viên hoặc thông tin cá nhân, khỏi truy cập trái phép. Chỉ những người dùng được xác minh, như giảng viên hoặc sinh viên, mới có thể truy cập dữ liệu của chính họ hoặc dữ liệu mà họ được phép xem, đảm bảo an toàn và riêng tư.

Hệ thống sẵn sàng cao Hệ thống chatbot phải hoạt động liên tục, sẵn sàng sử dụng hầu hết thời gian, ví dụ đạt 99,9% thời gian hoạt động. Điều này đảm bảo giảng viên và sinh viên có thể tra cứu thông tin bất kỳ lúc nào, kể cả trong các giai đoạn cao điểm như trước kỳ thi hoặc kỳ đăng ký học.

Hỗ trợ nhiều người dùng Hệ thống phải hỗ trợ một lượng lớn người dùng cùng lúc, ví dụ lên đến 10.000 người dùng trong giờ cao điểm, mà không làm giảm tốc độ hoặc chất lượng phản hồi.

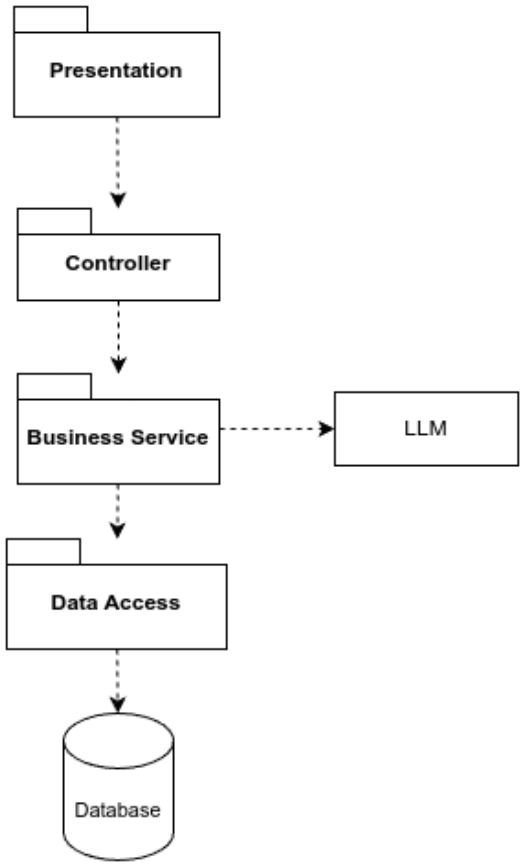
2.2.6. Tổng kết chương

Trong chương vừa rồi, khóa luận đã làm rõ bài toán tra cứu thông tin học tập trong môi trường đại học. Hệ thống chatbot thông minh được đề xuất giúp giảng viên và sinh viên truy vấn thông tin bằng ngôn ngữ tự nhiên. Các yêu cầu chức năng bao gồm các ca sử dụng cho giảng viên (tra cứu lịch giảng dạy, thông tin sinh viên), sinh viên (tra cứu thời khóa biểu, học phí), và ca sử dụng chung (thông tin chương trình đào tạo, môn học mở). Các yêu cầu phi chức năng đảm bảo hệ thống trả lời nhanh, dễ sử dụng, bảo mật thông tin, và hoạt động ổn định. Từ việc đặc tả yêu cầu này, khóa luận sẽ tiến hành phân tích và thiết kế hệ thống. Trong Chương 4 tiếp theo, khóa luận sẽ trình bày chi tiết thiết kế hệ thống dựa trên các yêu cầu từ Chương 3.

Chương 3.

Phân tích thiết kế hệ thống

Phần này tập trung vào việc phân tích và thiết kế hệ thống chatbot, nhằm xác định các yêu cầu chức năng và phi chức năng, đồng thời xây dựng một kiến trúc phần mềm hiệu quả. Quá trình này sẽ áp dụng các phương pháp phân tích hiện đại, bao gồm việc sử dụng các sơ đồ UML (như Use Case Diagram và Class Diagram) để mô hình hóa yêu cầu, cùng với các nguyên tắc thiết kế hướng đối tượng và mô hình kiến trúc phân lớp (layered architecture). Ngoài ra chương trình bày cụ thể thuật toán phân quyền của chương trình. Ngay sau đây, chúng ta sẽ đi sâu vào phân tích kiến trúc của hệ thống để đảm bảo tính mở rộng, bảo mật, và hiệu suất trong quá trình triển khai.



Hình 3.1. Kiến trúc tổng quan

3.1. Phân tích kiến trúc

Để xây dựng một hệ thống chatbot vững chắc, việc phân tích kiến trúc là bước quan trọng nhằm xác định các thành phần chính và mối quan hệ giữa chúng. Quá trình này sẽ sử dụng mô hình kiến trúc phân lớp (layered architecture). Dưới đây là mô hình tổng quan kiến trúc, cung cấp cái nhìn chi tiết về cách hệ thống được tổ chức và vận hành.

3.1.1. Mô hình tổng quan kiến trúc

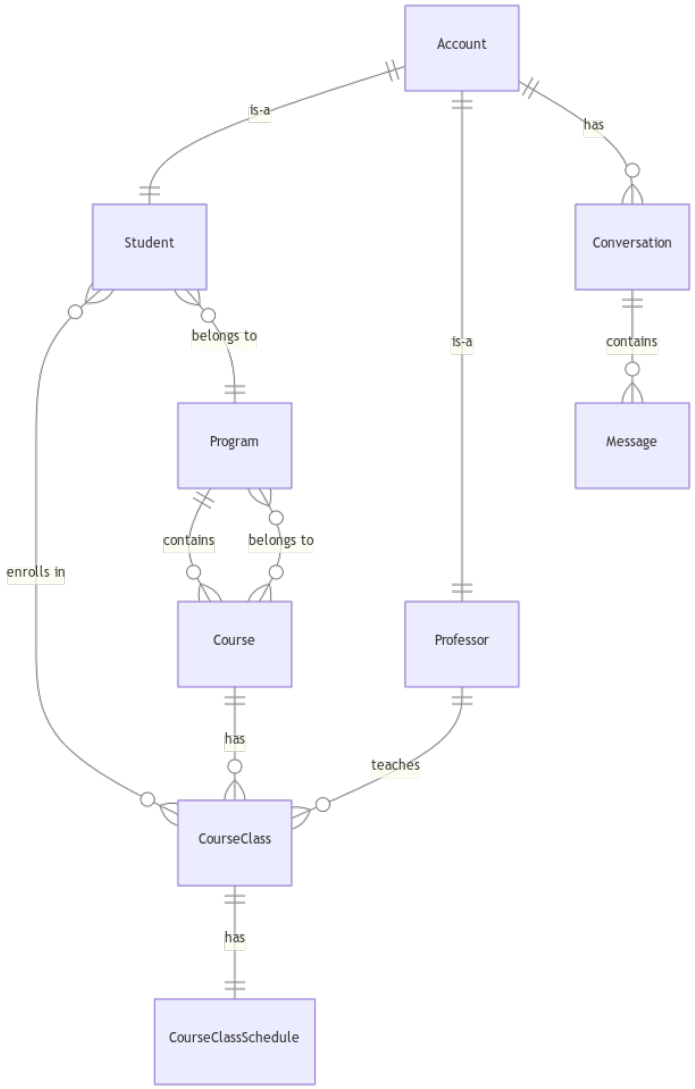
Để triển khai hiệu quả hệ thống chatbot, mô hình tổng quan kiến trúc được minh họa trong Hình 3.1, thể hiện cách các lớp tương tác để xử lý yêu cầu từ người dùng. Hình 3.1 cung cấp cái nhìn tổng quan về cấu trúc phần mềm, bao gồm bốn lớp chính tham gia trong việc xử lý và truyền tải dữ liệu, được mô tả như sau:

- **Lớp biểu diễn (Presentation)** được xây dựng bằng `React`, chứa các thành phần giao diện web mà người dùng (sinh viên và giảng viên) tương tác trực tiếp, đảm bảo trải nghiệm người dùng mượt mà và phản hồi nhanh chóng với các cập nhật thời gian thực thông qua giao thức SSE.
- **Lớp điều khiển (Controller)** được triển khai bằng framework `gin` trên nền `Golang`, nhận yêu cầu từ lớp biểu diễn và điều phối đến các dịch vụ phù hợp phía dưới. Lớp này cũng sử dụng Server-Sent Events (SSE) để trả về gói tin phản hồi cho lớp biểu diễn khi phản hồi tin nhắn.
- **Lớp dịch vụ (Business Service)** được viết bằng `Golang`, chứa logic nghiệp vụ chính của hệ thống. Lớp này nhận yêu cầu từ lớp điều khiển, thực hiện tính toán và gọi đến OpenAI LLM khi cần thiết thông qua thư viện `go-openai` để phân tích các câu hỏi tự nhiên (natural language queries), đồng thời tương tác với lớp truy cập dữ liệu để lấy thông tin cần thiết.
- **Lớp truy cập dữ liệu (Data Access)** sử dụng `sqlx` trên nền `Golang`, nhận yêu cầu truy xuất dữ liệu từ lớp dịch vụ và chuyển đổi thành các câu truy vấn SQL để tương tác với cơ sở dữ liệu, đảm bảo hiệu suất truy vấn tối ưu và tính bảo mật của dữ liệu.

Sau khi nắm được cấu trúc phân lớp, phần tiếp theo sẽ trình bày các trừu tượng chính giúp phân tích các ca sử dụng trước khi chuyển sang giai đoạn thiết kế chi tiết.

3.1.2. Các trừu tượng chính

Các trừu tượng chính được xác định như những thực thể cốt lõi để bắt đầu phân tích các ca sử dụng, cung cấp nền tảng cho việc hiểu rõ các yêu cầu chức năng của hệ thống chatbot. Hình 3.2 minh họa mối quan hệ giữa các trừu tượng này, giúp làm rõ vai trò và cách chúng tương tác trong quá trình phân tích. Các trừu tượng này sẽ được sử dụng để xây dựng mô hình Use Case Diagram và xác định các mối quan hệ giữa người dùng, hệ thống, và dữ liệu.



Hình 3.2. Các trừu tượng chính

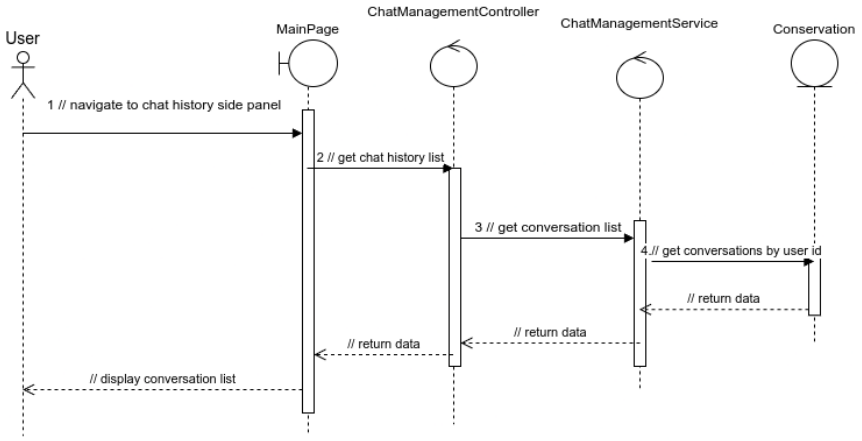
Danh sách các trừu tượng chính bao gồm như sau:

- **Tài khoản (User Account)** Một hồ sơ tài khoản của thầy cô, sinh viên chứa các thông tên, tên tài khoản, mật khẩu, Mỗi một tài khoản có 1 ID duy nhất được sử dụng để xác định người dùng, cấp quyền truy cập cho từng loại tài khoản.
- **Cuộc hội thoại (Conservation)** chỉ một đoạn hội thoại gồm nhiều tin nhắn của người dùng với hệ thống chatbot
- **Tin nhắn (Message)** chỉ một tin nhắn giữa người dùng và hệ thống chatbot, có thể là tin nhắn của người dùng gửi đến hệ thống hoặc tin nhắn trả lời của hệ thống.
- **Học sinh (Student)** Đối tượng sinh viên với các thông tin về mã sinh viên, tên, ngày sinh, giới tính, email, lớp hành chính
- **Giảng viên (Professor)** với các thông tin về cấp bậc, khoa và cũng sẽ cơ sở để tạo liên kết tới khi ta xét đến các môn dạy, các lịch dạy tiếp theo
- **Học phần (Course)** biểu diễn thông tin về 1 học phần như tên, số tín chỉ, học phần nào tiên quyết,...
- **Lớp học phần (Course Class)** biểu thị một lớp mở ra trong một học kỳ. Một học phần có thể được mở nhiều lớp khác nhau trong cùng một kỳ học. Lớp học phần thường cũng sẽ được dạy bởi một hoặc một số giảng viên (giảng viên lý thuyết và thực hành)
- **Lịch học phần (Course Class Schedule)** biểu thị một buổi dạy học của một lớp học phần. Lịch học phần thường có 2 loại chính là lịch học lý thuyết và thực hành. Một lớp học phần có thể được dạy một số buổi trong tuần, có thể chỉ có nguyên lý thuyết hoặc cả lý thuyết và thực hành. Mỗi lịch học thường đi kèm với 1 giảng viên đứng lớp, địa điểm phòng học và ngày trong tuần.
- **Lớp hành chính (Administrative Class)** là lớp nhóm học sinh được tổ chức trong trường đại học dưới sự hướng dẫn của một thầy cố vấn.

3.1.3. Phân tích ca sử dụng

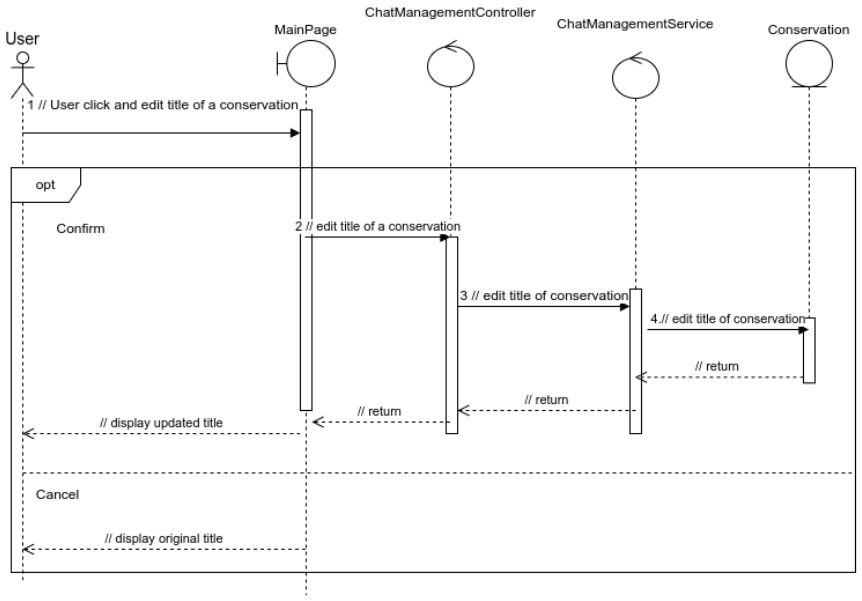
Sau khi xác định các trừu tượng chính, bước tiếp theo là phân tích chi tiết từng ca sử dụng để hiểu rõ cách các thành phần hệ thống tương tác với nhau trong quá trình xử lý yêu cầu từ người dùng. Quá trình này sử dụng các biểu đồ tuần tự (sequence diagrams)

để minh họa trình tự các thông điệp và hành động giữa các thực thể, từ giao diện người dùng đến các lớp logic và dữ liệu, cung cấp cơ sở vững chắc cho giai đoạn thiết kế sau này. Các biểu đồ tuần tự dưới đây trình bày cách hệ thống xử lý từng ca sử dụng, bắt đầu từ các chức năng về quản lý hội thoại, và sau đó là các chức năng tra cứu thông tin học tập



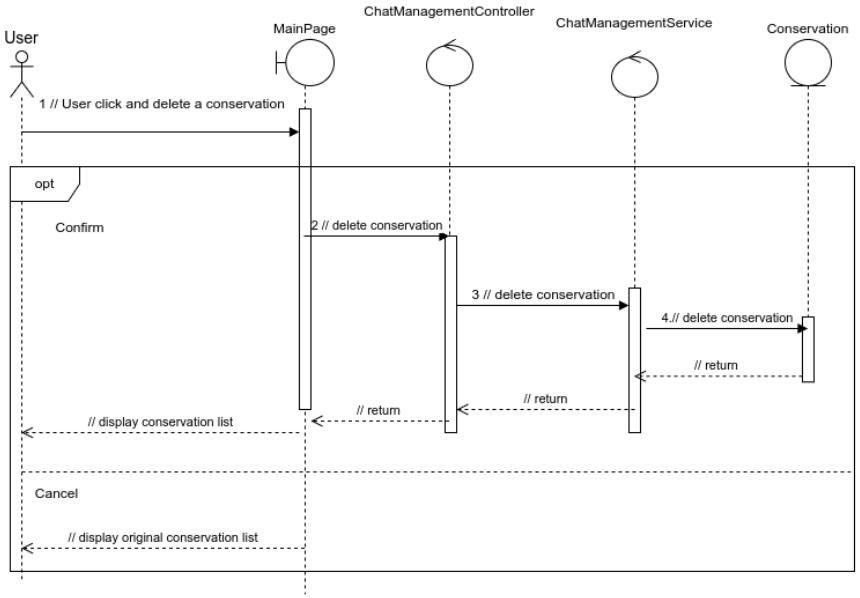
Hình 3.3. Biểu đồ tuần tự ca sử dụng xem danh sách hội thoại cá nhân

Hình 3.3 mô tả trình tự thực hiện cho ca sử dụng xem danh sách hội thoại cá nhân



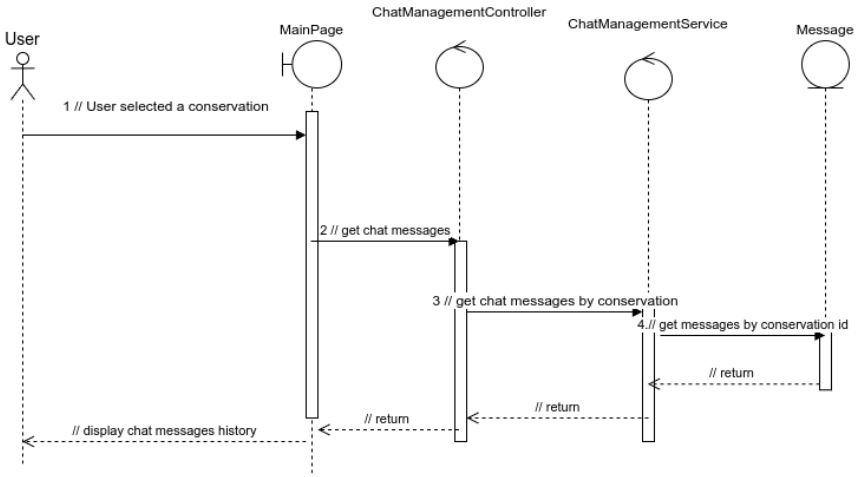
Hình 3.4. Biểu đồ tuần tự ca sử dụng sửa tiêu đề hội thoại

Hình 3.4 mô tả trình tự thực hiện cho ca sử dụng sửa tiêu đề hội thoại



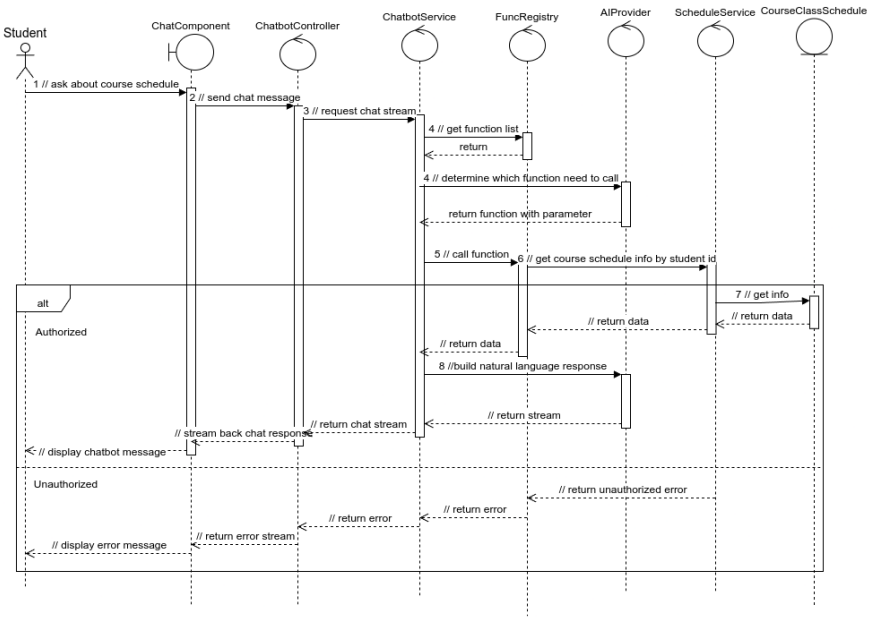
Hình 3.5. Biểu đồ tuần tự ca sử dụng xóa hội thoại

Hình 3.5 mô tả trình tự thực hiện cho ca sử dụng xóa hội thoại



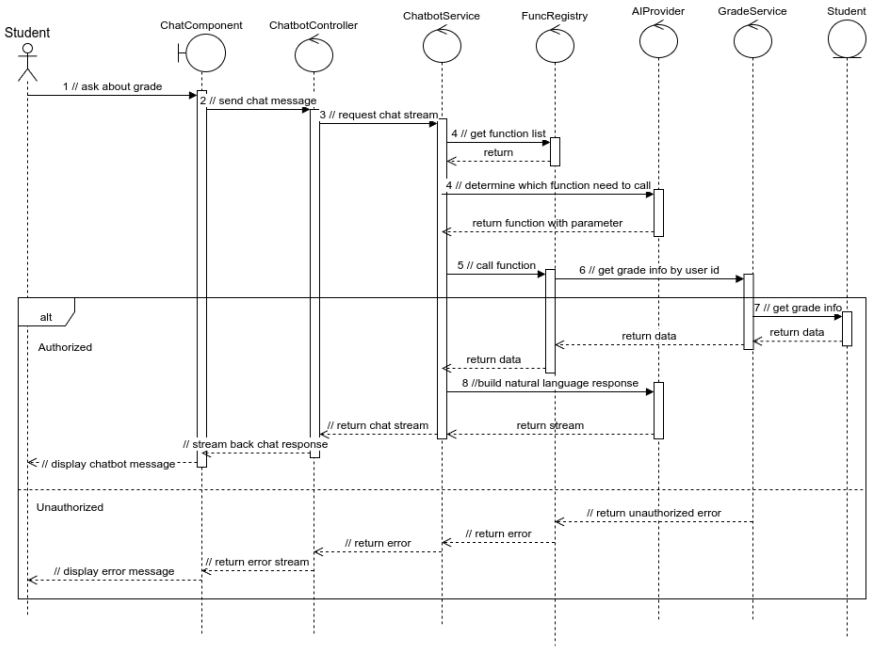
Hình 3.6. Biểu đồ tuần tự ca sử dụng xem lịch sử tin nhắn

Hình 3.6 mô tả trình tự thực hiện cho ca sử dụng xem lịch sử tin nhắn



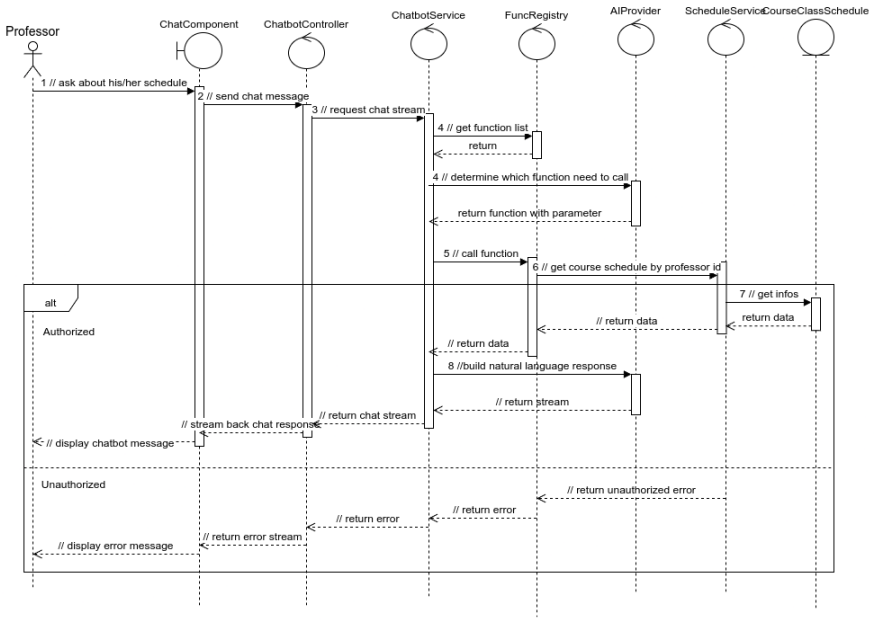
Hình 3.7. Biểu đồ tuần tự ca sử dụng tra cứu thời khóa biểu, môn học cá nhân

Hình 3.7 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin môn học và thời khóa biểu cá nhân.



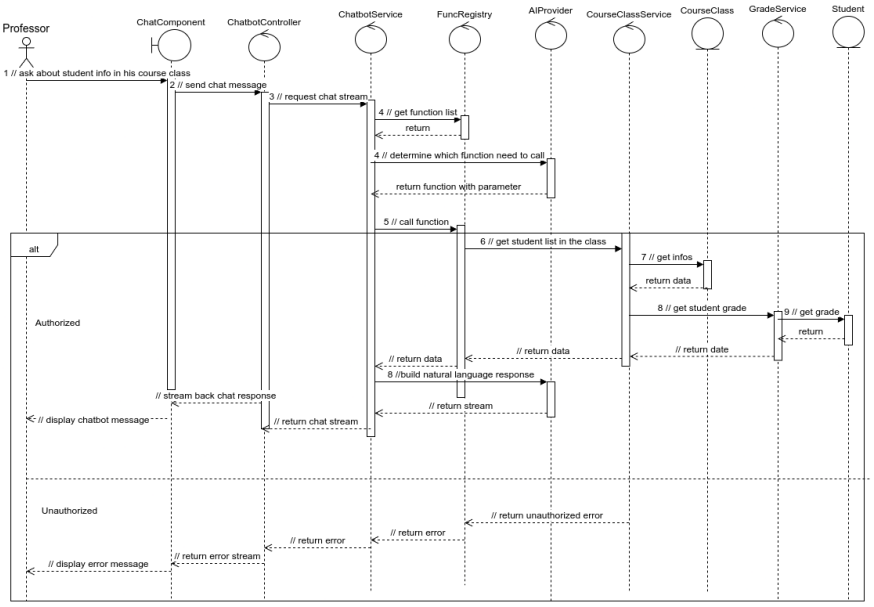
Hình 3.8. Biểu đồ tuần tự ca sử dụng tra cứu điểm số và tiến độ học tập

Hình 3.8 mô tả trình tự thực hiện cho ca sử dụng tra cứu điểm số và tiến độ học tập.



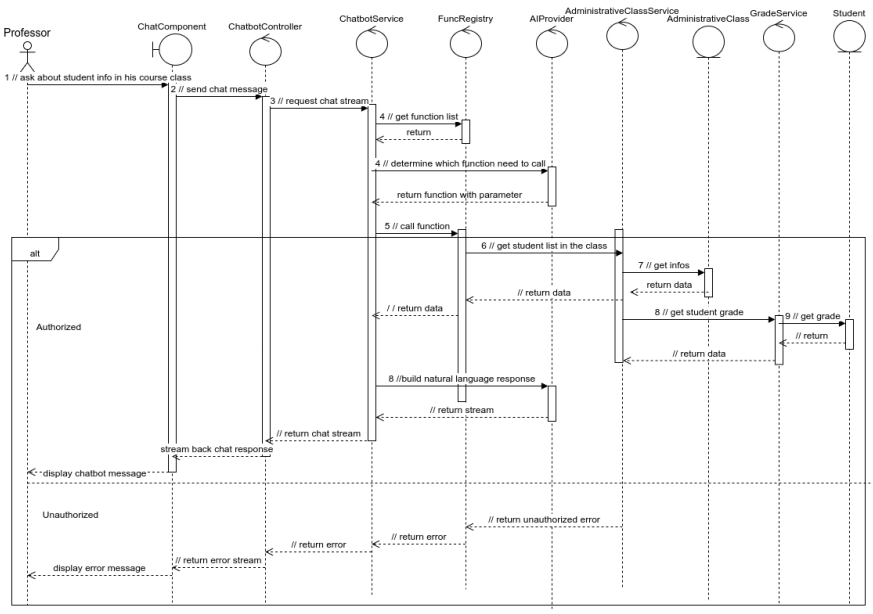
Hình 3.9. Biểu đồ tuần tự ca sử dụng tra cứu lịch giảng dạy

Hình 3.9 mô tả trình tự thực hiện cho ca sử dụng tra cứu lịch giảng dạy.



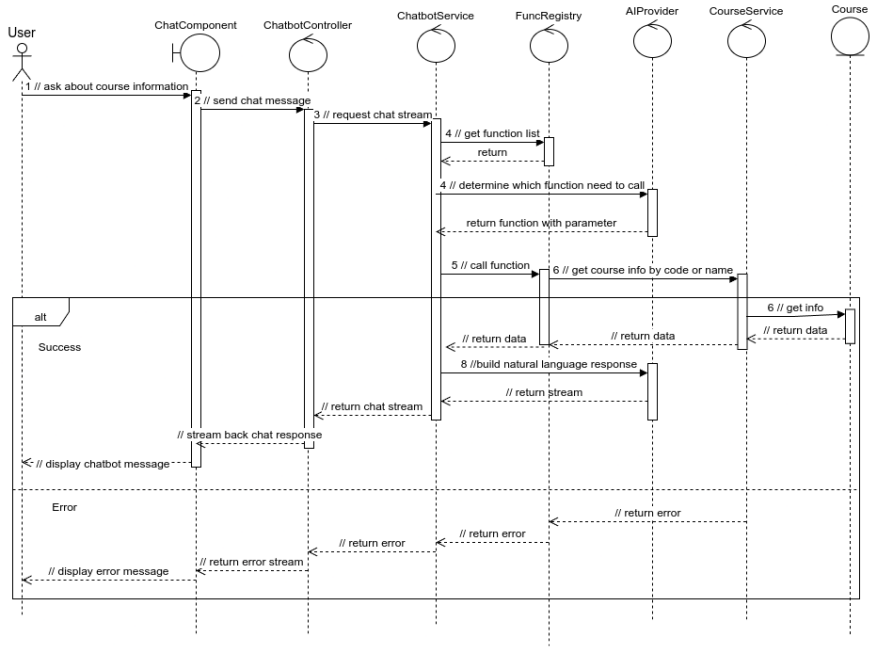
Hình 3.10. Biểu đồ tuần tự ca sử dụng tra cứu điểm số và thông tin sinh viên lớp học phần

Hình 3.10 mô tả trình tự thực hiện cho ca sử dụng tra cứu điểm số và thông tin sinh viên lớp học phần.



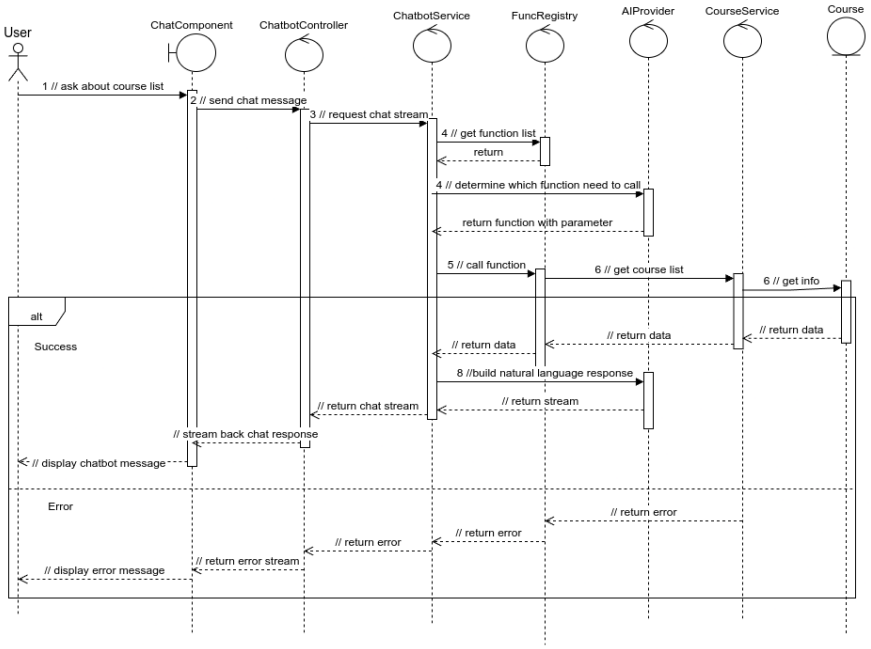
Hình 3.11. Biểu đồ tuần tự ca sử dụng tra cứu thông tin sinh viên trong lớp hành chính

Hình 3.11 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin sinh viên trong lớp hành chính.



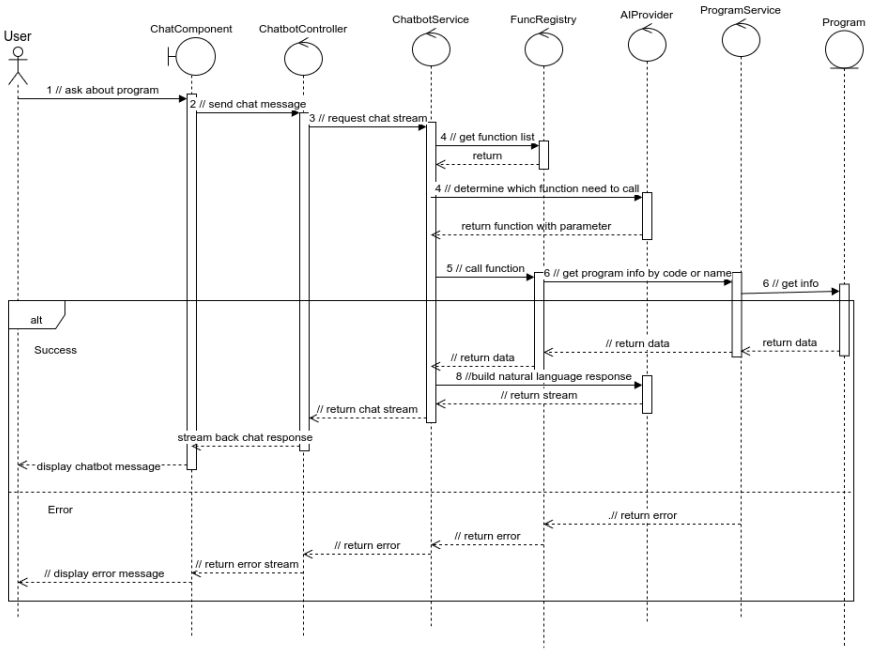
Hình 3.12. Biểu đồ tuần tự ca sử dụng tra cứu thông tin môn học cụ thể

Hình 3.12 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin môn học cụ thể.



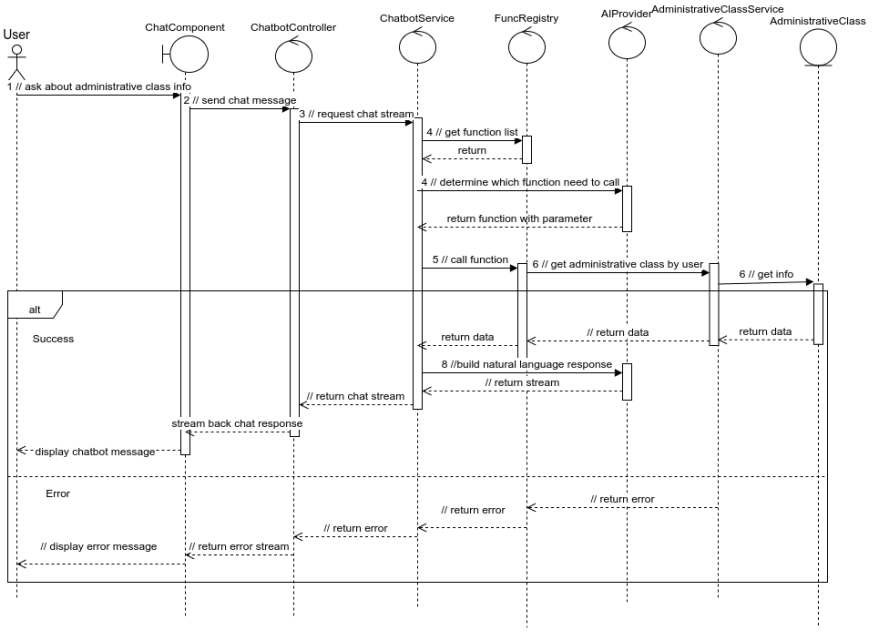
Hình 3.13. Biểu đồ tuần tự ca sử dụng tra cứu danh sách môn học mở

Hình 3.13 mô tả trình tự thực hiện cho ca sử dụng tra cứu danh sách môn học mở.



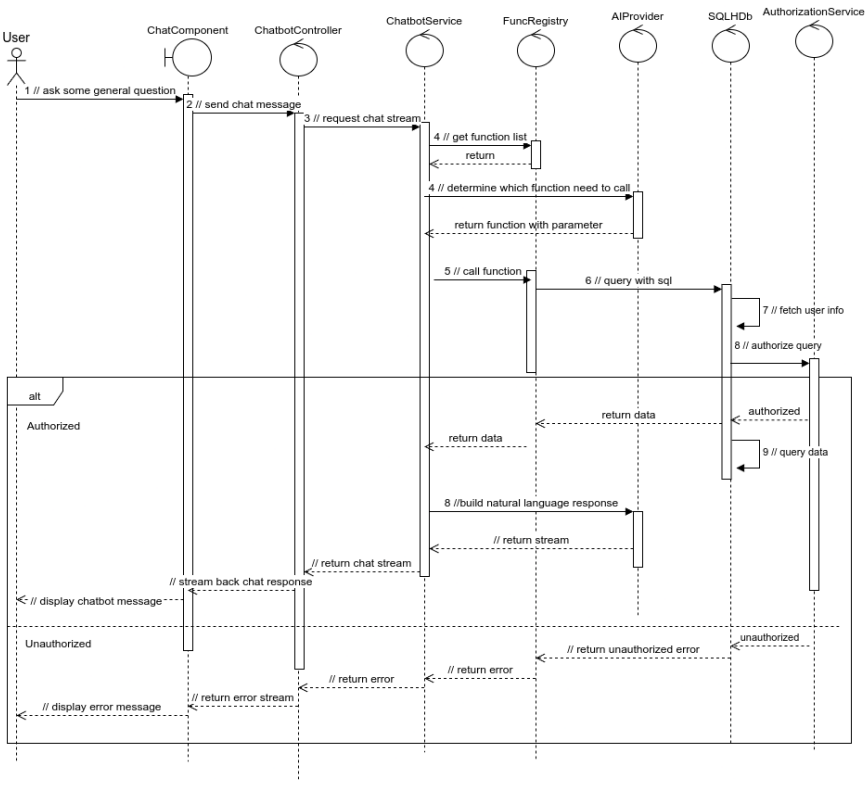
Hình 3.14. Biểu đồ tuần tự ca sử dụng tra cứu thông tin chương trình đào tạo

Hình 3.14 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin chương trình đào tạo.



Hình 3.15. Biểu đồ tuần tự ca sử dụng tra cứu thông tin lớp hành chính cá nhân

Hình 3.15 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tinh lớp hành chính cá nhân

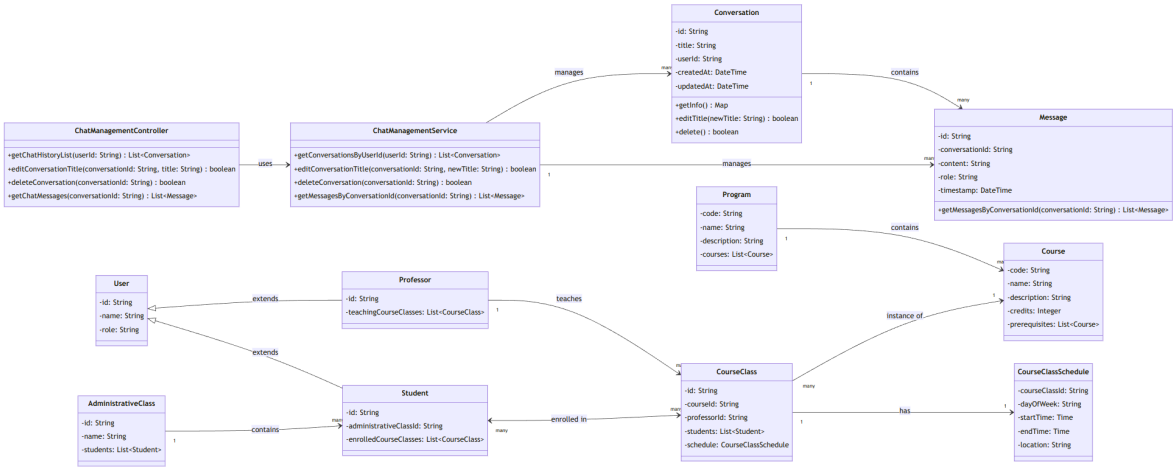


Hình 3.16. Biểu đồ tuần tự ca sử dụng tra cứu thông tin khác trong cơ sở dữ liệu

Hình 3.16 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin khác trong cơ sở dữ liệu, đảm bảo chỉ tra cứu trong quyền truy cập.

3.1.4. Biểu đồ lớp phân tích

Dựa trên các biểu đồ tuần tự đã phân tích, bước tiếp theo là trích xuất biểu đồ lớp phân tích (analysis class diagram) để xác định các lớp chính và mối quan hệ giữa chúng, theo phương pháp IBM. Biểu đồ này chuyển đổi các tương tác động thành mô hình tĩnh, làm rõ cấu trúc hệ thống chatbot trước khi thiết kế.



Hình 3.17. Biểu đồ lớp phân tích

Do biểu đồ rất lớn nên khóa luận chỉ minh họa một phần chính của biểu đồ lớp phân tích cùng các lớp thực thể phân tích chính như trong Hình 3.17.

Phần tiếp theo sẽ chuyển sang giai đoạn thiết kế chi tiết, nơi các lớp này sẽ được triển khai thành các thành phần cụ thể.

3.2. Thiết kế hệ thống

Sau khi hoàn thành giai đoạn phân tích với các biểu đồ tuần tự và biểu đồ lớp phân tích, chúng ta tiến vào giai đoạn thiết kế hệ thống, nơi các khái niệm trừu tượng được chuyển đổi thành các thành phần cụ thể và thực thi được. Mục tiêu của phần này là xây dựng một thiết kế chi tiết, đảm bảo hệ thống chatbot hoạt động hiệu quả, tích hợp các công nghệ đã chọn như React, Golang, và OpenAI LLM, đồng thời đáp ứng các yêu cầu chức năng và phi chức năng đã xác định trước đó. Quá trình thiết kế bắt đầu bằng việc xác định các lớp thiết kế dựa trên các lớp phân tích, sau đó mở rộng thành các mô hình chi tiết hơn như cơ sở dữ liệu, và thuật taons. Dưới đây, chúng ta sẽ tập trung vào bước đầu tiên là xác định các lớp thiết kế, kết nối trực tiếp với các lớp phân tích đã xây dựng.

3.2.1. Xác định các lớp

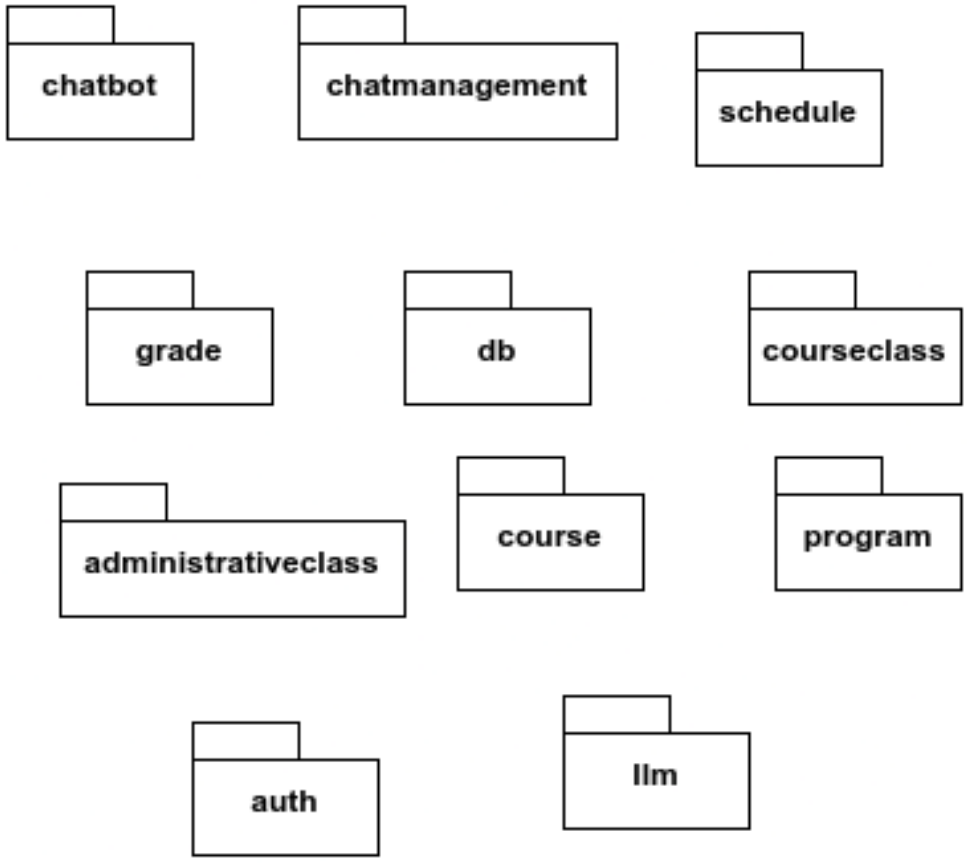
Để chuyển từ giai đoạn phân tích sang thiết kế, chúng ta cần ánh xạ các lớp phân tích thành các lớp thiết kế cụ thể, phù hợp với kiến trúc phân lớp và các công nghệ được sử dụng. Quá trình này bao gồm việc giữ nguyên một số lớp cơ bản, bổ sung các lớp bổ trợ để hỗ trợ triển khai, và điều chỉnh để phù hợp với logic thực thi của hệ thống. Bảng 3.1 tóm tắt quá trình chuyển đổi từ các lớp phân tích sang các lớp thiết kế, làm rõ mối quan hệ và sự mở rộng cần thiết để triển khai hệ thống chatbot một cách hiệu quả.

Bảng 3.1. Bảng chuyển đổi từ lớp phân tích sang lớp thiết kế

Lớp phân tích	**Lớp thiết kế**
Conversation	Conversation
Message	Message
CourseClassSchedule	CourseClassSchedule, Faculty, Professor
Student	Student, CourseClassEnrollment
CourseClass	CourseClass
AdministrativeClass	AdministrativeClass
Course	Course
Program	Program
UserAccount	UserAccount
LoginForm	LoginForm
MainPage	MainPage
ChatComponent	ChatComponent
UserController	UserController
UserService	UserService
JwtService	JwtService
ChatManagementController	ChatManagementController
ChatManagementService	ChatManagementService
ChatbotController	ChatbotController
ChatbotService	ChatbotService
ScheduleService	ScheduleService
FuncRegistry	FuncRegistry
AIProvider	AIProvider
GradeService	GradeService
CourseClassService	CourseClassService
AdministrativeClassService	AdministrativeClassService
CourseService	CourseService
ProgramService	ProgramService
SQLHdb	SQLHdb
AuthorizationService	AuthorizationService

Phần tiếp theo, chúng ta sẽ chia các lớp trên vào các gói phù hợp

3.2.2. Xác định các gói



Hình 3.18. Biểu đồ gói

Các lớp sẽ được chia vào các gói như trong Hình 3.18.

– **Gói chatbot**: Tập trung vào việc xử lý các tương tác chính của chatbot với người dùng, bao gồm quản lý logic trò chuyện giữa người dùng và hệ thống

– **Gói chatmanagement**: Chịu trách nhiệm quản lý hội thoại, hỗ trợ các chức năng như xem, chỉnh sửa và xóa các hội thoại giữa người dùng và chatbot.

– **Gói schedule**: Đảm nhiệm việc quản lý và tra cứu lịch trình, bao gồm thời khóa biểu cá nhân và lịch giảng dạy, giúp người dùng dễ dàng truy cập thông tin lịch học.

– **Gói grade**: Tập trung vào việc quản lý và tra cứu điểm số, cũng như theo dõi tiến độ học tập của người dùng, hỗ trợ các yêu cầu liên quan đến kết quả học tập.

– **Gói db**: Cung cấp giao diện để truy cập và quản lý dữ liệu từ cơ sở dữ liệu, đảm bảo hệ thống có thể lưu trữ và truy xuất thông tin một cách hiệu quả.

– **Gói courseclass**: Quản lý thông tin liên quan đến các lớp học phần, bao gồm việc tổ chức và tra cứu dữ liệu về các lớp học cụ thể trong từng học kỳ.

– **Gói administrativeclass**: Xử lý thông tin về các lớp hành chính, hỗ trợ tra cứu và quản lý dữ liệu liên quan đến các lớp hành chính của sinh viên.

– **Gói course**: Đảm nhiệm việc quản lý thông tin về các môn học, bao gồm tra cứu danh sách môn học và các chi tiết liên quan.

– **Gói program**: Tập trung vào việc quản lý thông tin về các chương trình đào tạo, hỗ trợ tra cứu và cung cấp chi tiết về chương trình học.

– **Gói auth**: Chịu trách nhiệm xác thực và ủy quyền, bao gồm quản lý đăng nhập, đăng xuất và kiểm soát quyền truy cập của người dùng.

– **Gói llm**: Hỗ trợ tích hợp và cấu hình các mô hình ngôn ngữ lớn, cung cấp các chức năng liên quan đến việc sử dụng công nghệ AI trong hệ thống.

3.2.3. Cơ chế thiết kế

Phần này trình bày chi tiết các cơ chế phân tích được sử dụng trong khóa luận, cách chúng được chuyển đổi thành các cơ chế thiết kế và cuối cùng là cách chúng được triển khai trong hệ thống. Các cơ chế này đóng vai trò quan trọng trong việc đảm bảo tính nhất quán, bảo mật và khả năng mở rộng của hệ thống. Bảng 3.2 dưới đây mô tả cách các cơ chế phân tích được ánh xạ sang cơ chế thiết kế và cơ chế cài đặt cụ thể.

Bảng 3.2. Mô tả cơ chế thiết kế

Cơ chế phân tích	Cơ chế thiết kế	Cơ chế cài đặt
Persistency	RDBMS	Postgres, sqlx
Security	JWT tokens	Golang-JWT
Distribution	REST API, SSE	Framework Gin

Cơ chế **Persistency** được triển khai thông qua việc sử dụng hệ quản trị cơ sở dữ liệu quan hệ (RDBMS). Hệ thống sử dụng PostgreSQL làm cơ sở dữ liệu chính, kết hợp với thư viện sqlx trong ngôn ngữ Go để thực hiện các thao tác truy vấn và quản lý dữ liệu. Việc này đảm bảo tính toàn vẹn dữ liệu, hiệu suất cao và khả năng mở rộng khi khối lượng dữ liệu tăng lên.

Đối với cơ chế **Security**, khóa luận sử dụng JSON Web Tokens (JWT) để xác thực và phân quyền người dùng. Thư viện Golang-JWT được tích hợp để tạo, xác minh và quản lý các token, đảm bảo rằng chỉ những người dùng hợp lệ mới có thể truy cập vào các tài nguyên bảo mật của hệ thống. Cơ chế này giúp bảo vệ hệ thống khỏi các cuộc tấn công như truy cập trái phép hoặc giả mạo danh tính.

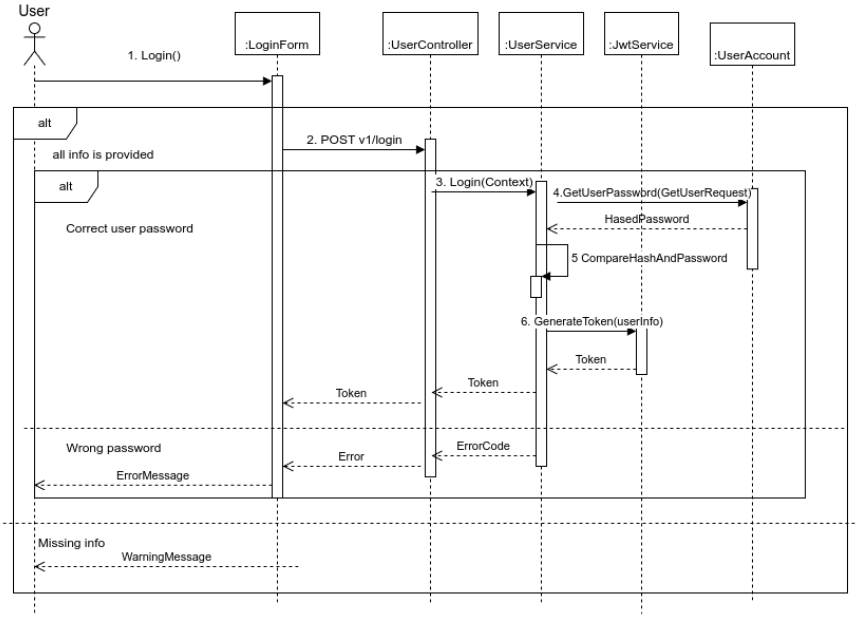
Cơ chế **Distribution** được thiết kế dựa trên kiến trúc REST API kết hợp với Server-Sent Events (SSE) để hỗ trợ giao tiếp thời gian thực và truyền tải dữ liệu hiệu quả. Framework Gin, một framework web nhẹ và mạnh mẽ trong Go, được sử dụng để xây dựng các endpoint API và quản lý các kết nối SSE. Điều này cho phép hệ thống xử lý các yêu cầu đồng thời một cách nhanh chóng và đáng tin cậy, đồng thời hỗ trợ các ứng dụng yêu cầu cập nhật dữ liệu theo thời gian thực.

Việc áp dụng các cơ chế thiết kế và cài đặt này không chỉ đáp ứng các yêu cầu phân tích ban đầu mà còn tạo nền tảng cho một hệ thống mạnh mẽ, dễ bảo trì và có khả năng mở rộng trong tương lai. Các công nghệ được lựa chọn đều là những công nghệ phổ biến và đã được chứng minh trong các ứng dụng thực tế, đảm bảo tính khả thi và hiệu quả của hệ thống.

3.2.4. Thiết kế ca sử dụng

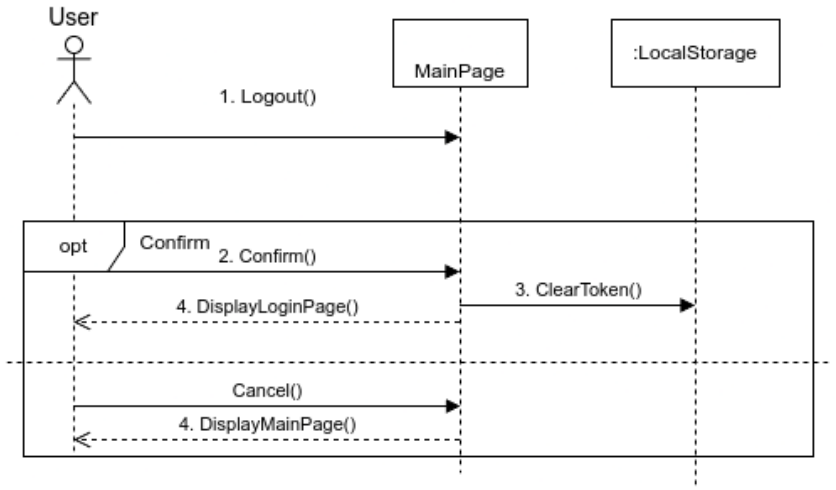
Phần này mô tả chi tiết các ca sử dụng chính của hệ thống, được minh họa thông qua các biểu đồ trình tự (sequence diagrams). Các biểu đồ này thể hiện rõ ràng luồng tương tác giữa các thành phần trong hệ thống, từ người dùng đến các lớp thiết kế, nhằm

đáp ứng các yêu cầu chức năng đã xác định trong giai đoạn phân tích. Mỗi ca sử dụng được trình bày kèm theo biểu đồ tương ứng, giúp làm rõ cách các cơ chế thiết kế được áp dụng để thực hiện chức năng cụ thể. Các ca sử dụng được thiết kế dựa trên hướng dẫn của phương pháp phân tích thiết kế hướng đối tượng, đảm bảo tính nhất quán và khả năng truy vết từ yêu cầu đến triển khai.



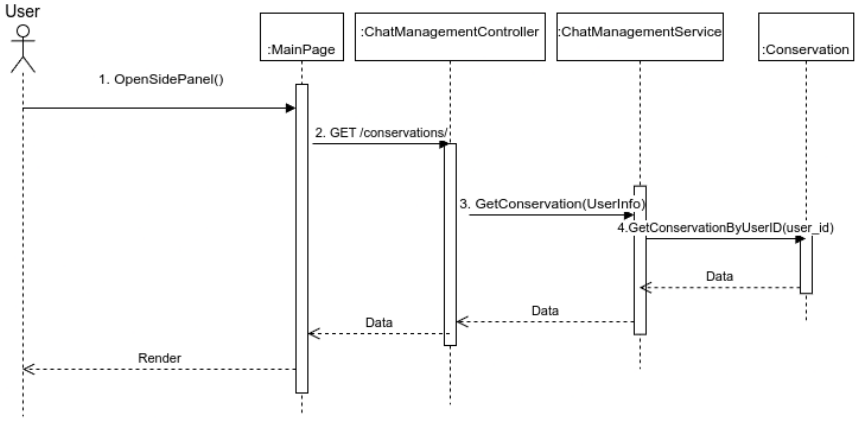
Hình 3.19. Thiết kế ca sử dụng đăng nhập

Hình 3.19 mô tả trình tự thực hiện cho ca sử dụng đăng nhập



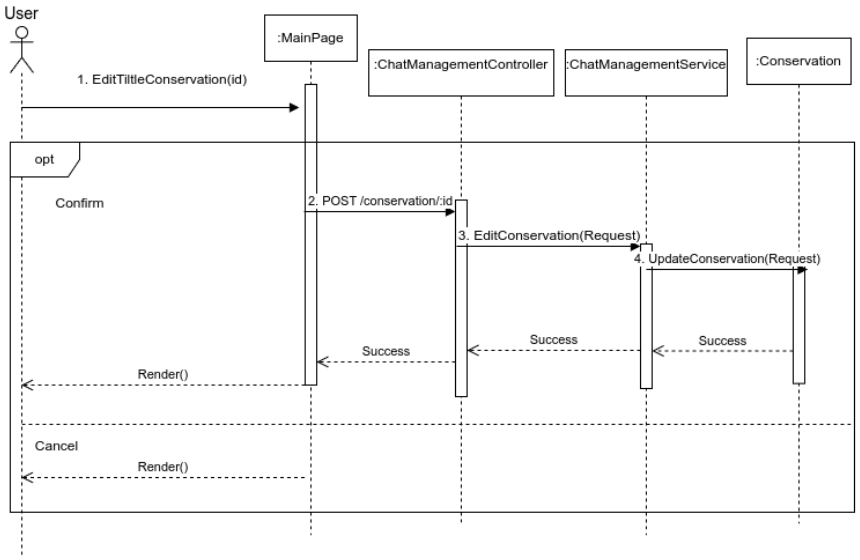
Hình 3.20. Thiết kế ca sử dụng đăng xuất

Hình 3.20 mô tả trình tự thực hiện cho ca sử dụng đăng xuất



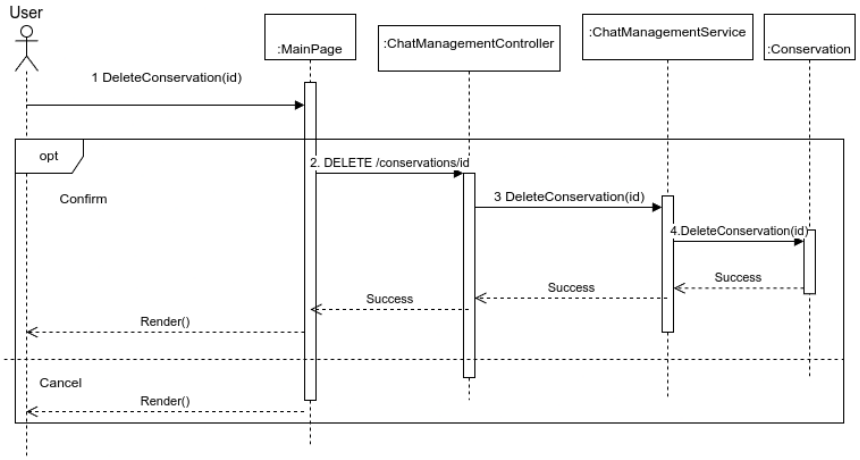
Hình 3.21. Thiết kế ca sử dụng xem danh sách hội thoại cá nhân

Hình 3.21 mô tả trình tự thực hiện cho ca sử dụng xem danh sách hội thoại cá nhân



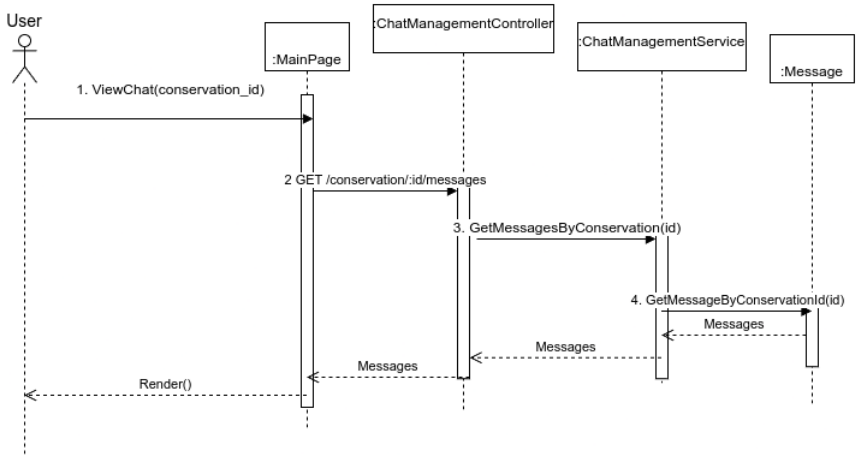
Hình 3.22. Thiết kế ca sử dụng sửa tiêu đề hội thoại

Hình 3.22 mô tả trình tự thực hiện cho ca sử dụng sửa tiêu đề hội thoại



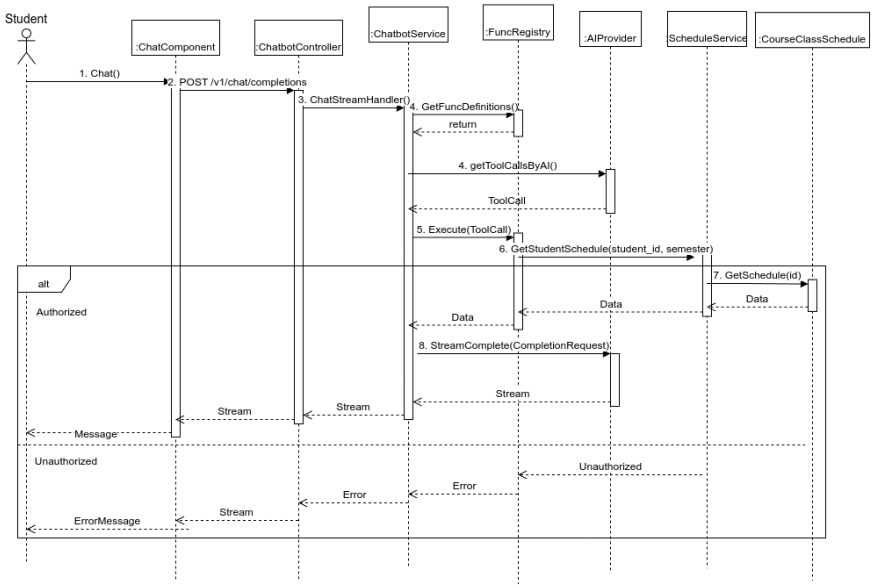
Hình 3.23. Thiết kế ca sử dụng xóa hội thoại

Hình 3.23 mô tả trình tự thực hiện cho ca sử dụng xóa hội thoại



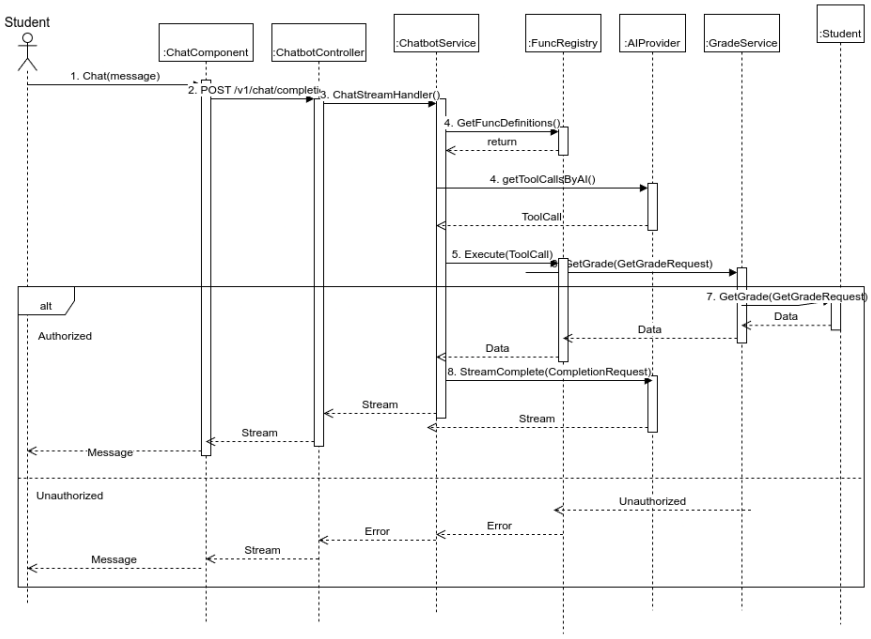
Hình 3.24. Thiết kế ca sử dụng xem lịch sử tin nhắn

Hình 3.24 mô tả trình tự thực hiện cho ca sử dụng xem lịch sử tin nhắn



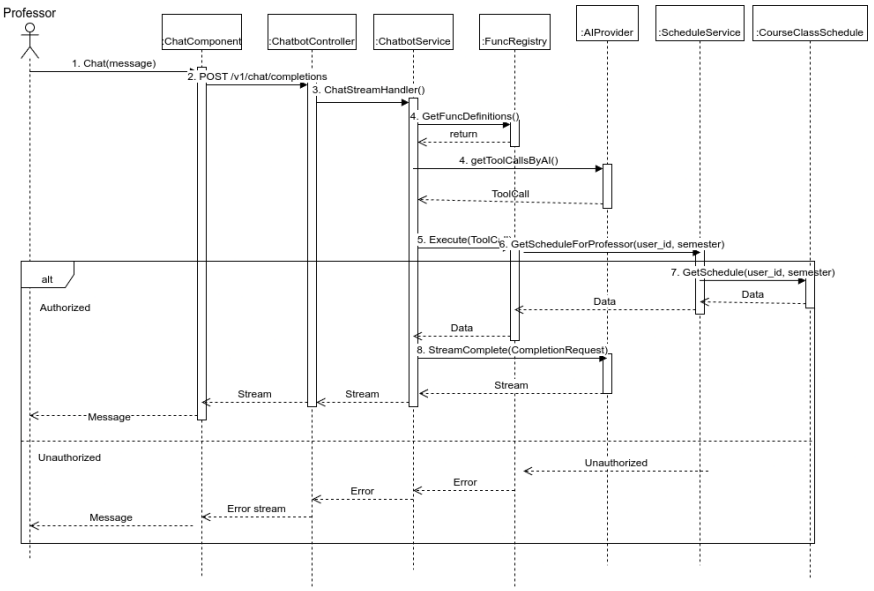
Hình 3.25. Thiết kế ca sử dụng tra cứu thời khóa biểu, môn học cá nhân

Hình 3.25 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin môn học và thời khóa biểu cá nhân.



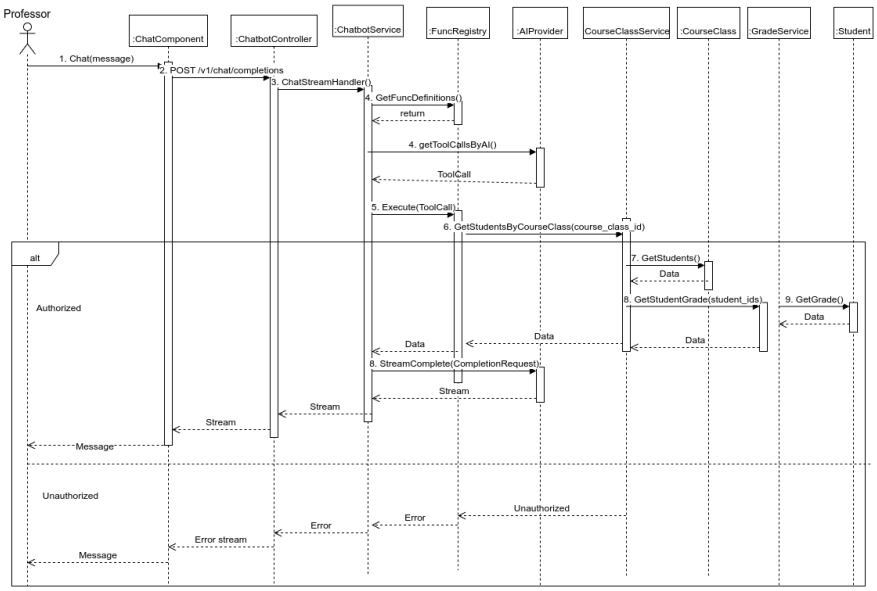
Hình 3.26. Thiết kế ca sử dụng tra cứu điểm số và tiến độ học tập

Hình 3.26 mô tả trình tự thực hiện cho ca sử dụng tra cứu điểm số và tiến độ học tập.



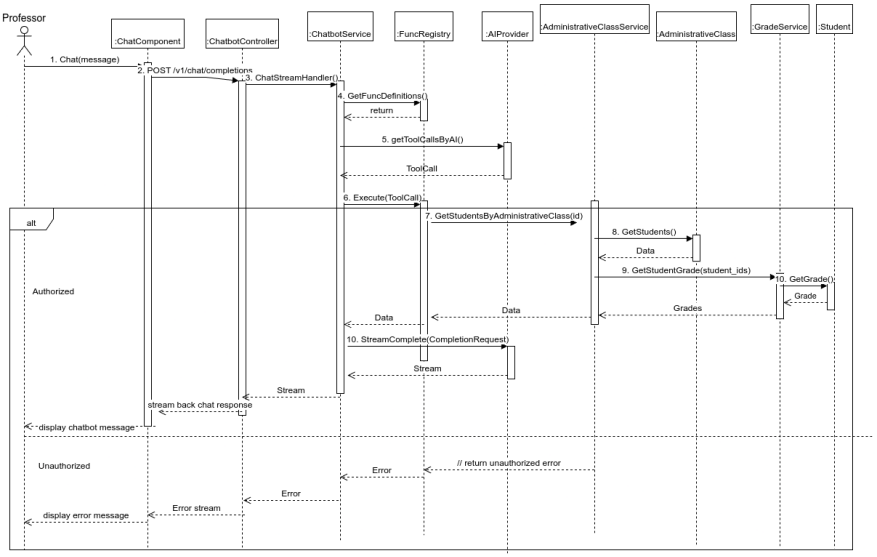
Hình 3.27. Thiết kế ca sử dụng tra cứu lịch giảng dạy

Hình 3.27 mô tả trình tự thực hiện cho ca sử dụng tra cứu lịch giảng dạy.



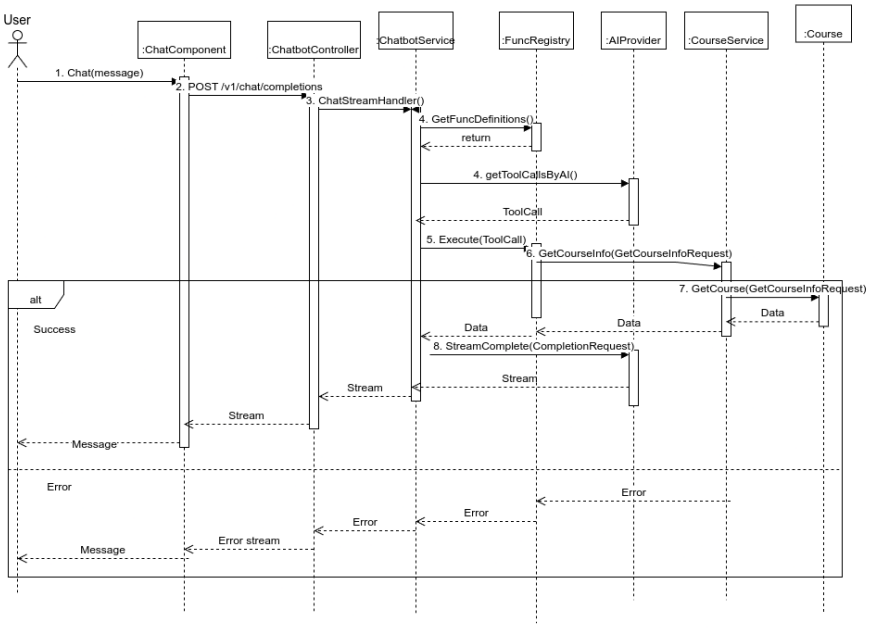
Hình 3.28. Thiết kế ca sử dụng tra cứu điểm số và thông tin sinh viên lớp học phần

Hình 3.28 mô tả trình tự thực hiện cho ca sử dụng tra cứu điểm số và thông tin sinh viên lớp học phần.



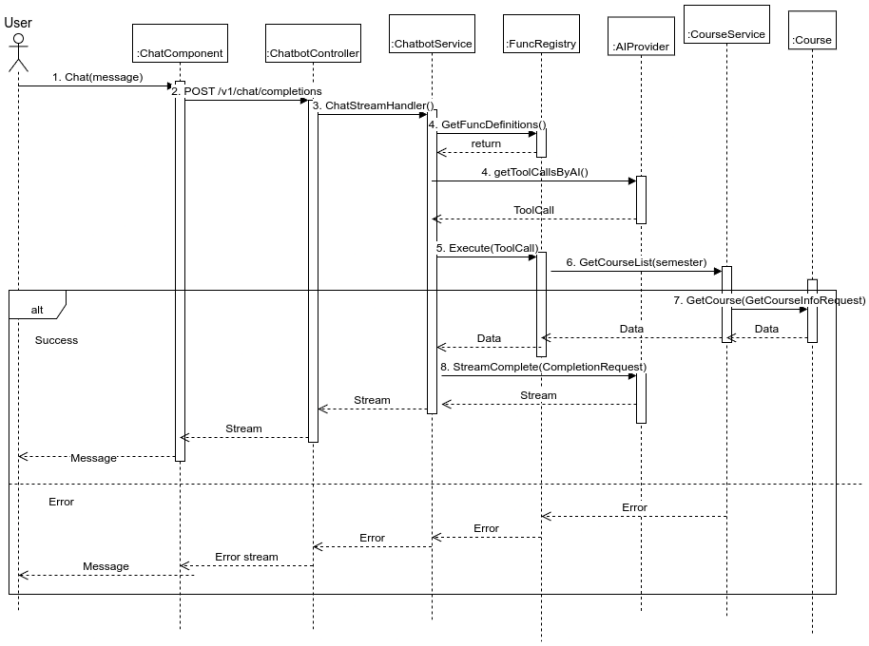
Hình 3.29. Thiết kế ca sử dụng tra cứu thông tin sinh viên trong lớp hành chính

Hình 3.29 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin sinh viên trong lớp hành chính.



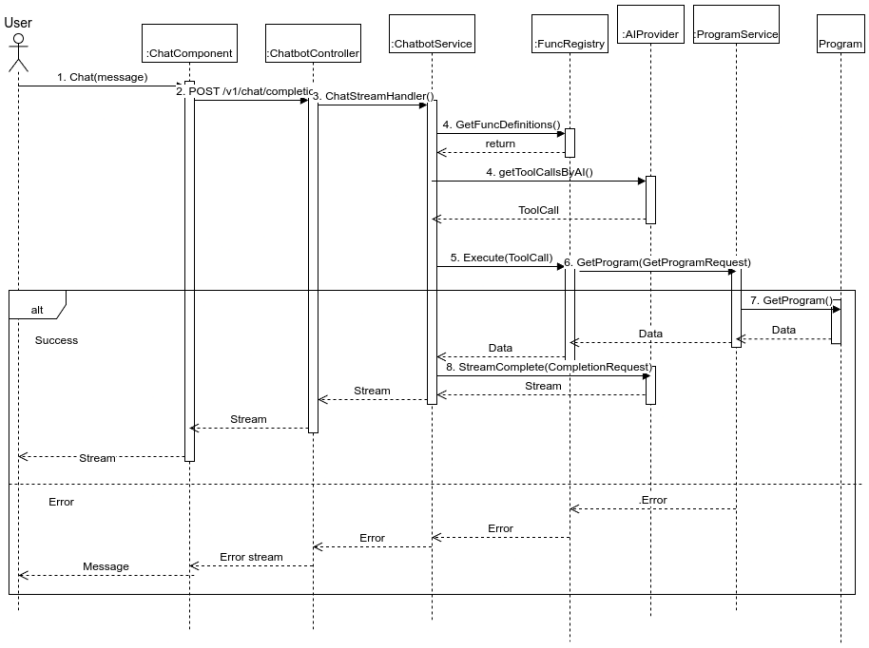
Hình 3.30. Thiết kế ca sử dụng tra cứu thông tin môn học cụ thể

Hình 3.30 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin môn học cụ thể.



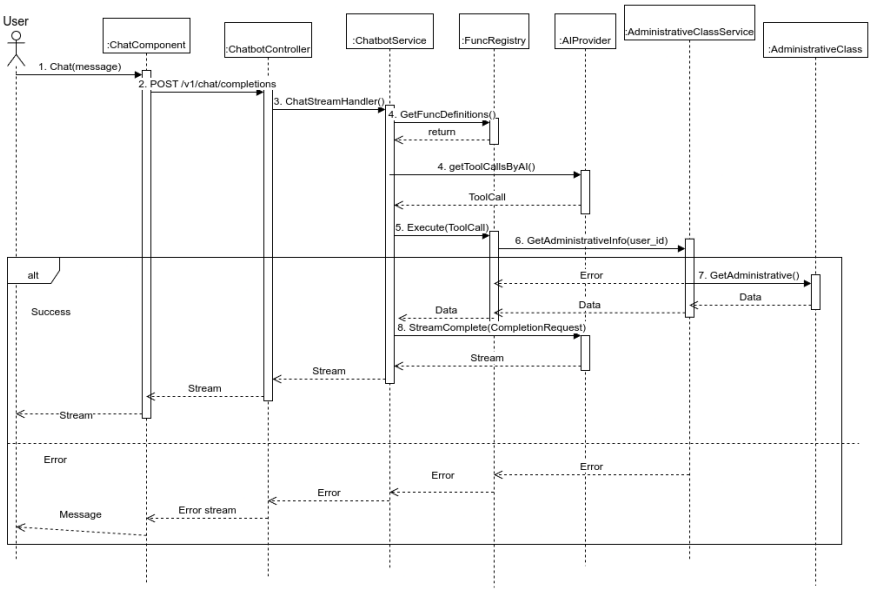
Hình 3.31. Thiết kế ca sử dụng tra cứu danh sách môn học mở

Hình 3.31 mô tả trình tự thực hiện cho ca sử dụng tra cứu danh sách môn học mở.



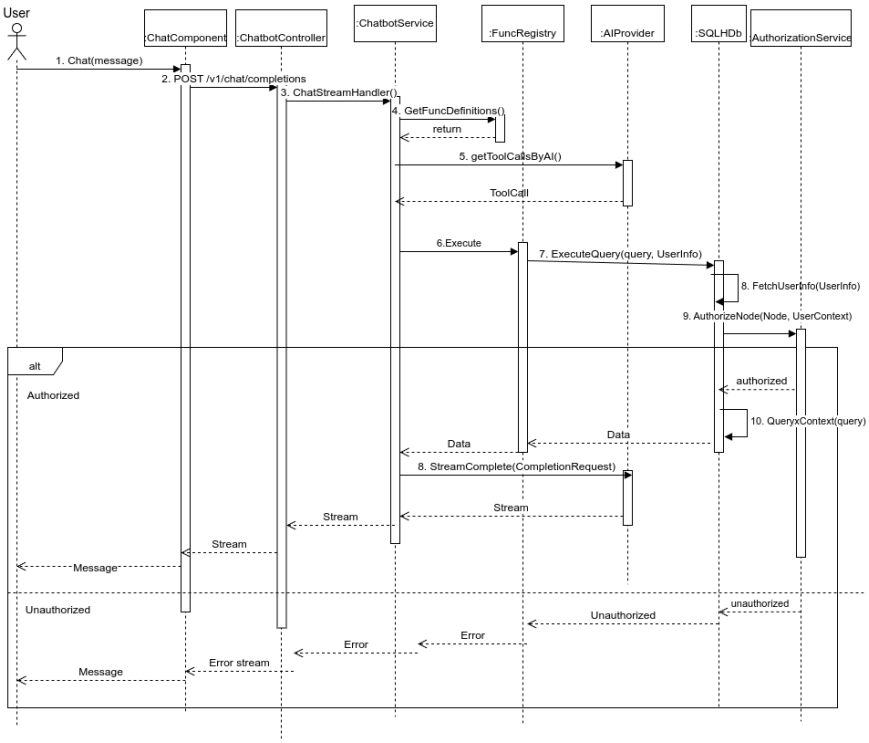
Hình 3.32. Thiết kế ca sử dụng tra cứu thông tin chương trình đào tạo

Hình 3.32 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin chương trình đào tạo.



Hình 3.33. Thiết kế ca sử dụng tra cứu thông tin lớp hành chính cá nhân

Hình 3.33 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin lớp hành chính cá nhân.

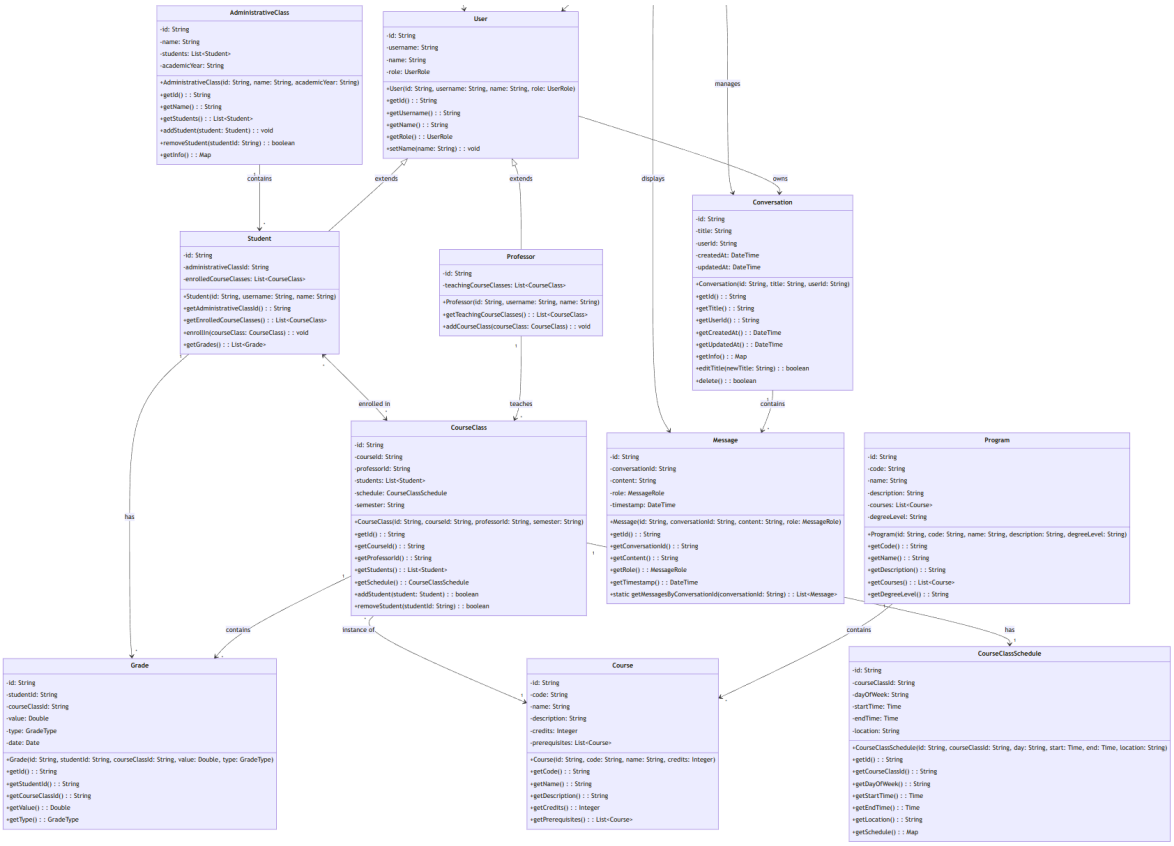


Hình 3.34. Thiết kế ca sử dụng tra cứu thông tin khác trong cơ sở dữ liệu

Hình 3.34 mô tả trình tự thực hiện cho ca sử dụng tra cứu thông tin khác trong cơ sở dữ liệu, đảm bảo chỉ tra cứu trong quyền truy cập.

3.2.5. Biểu đồ lớp thiết kế

Do quy mô lớn của biểu đồ lớp bao gồm tất cả các lớp trong hệ thống, phần này chỉ tập trung trình bày các lớp chính. Các lớp này đóng vai trò cốt lõi trong việc thực hiện các chức năng của hệ thống và được thiết kế dựa trên các nguyên tắc của OOAD, đảm bảo tính mô-đun, khả năng tái sử dụng và dễ dàng mở rộng. Biểu đồ lớp chính được thể hiện trong Hình 3.36.

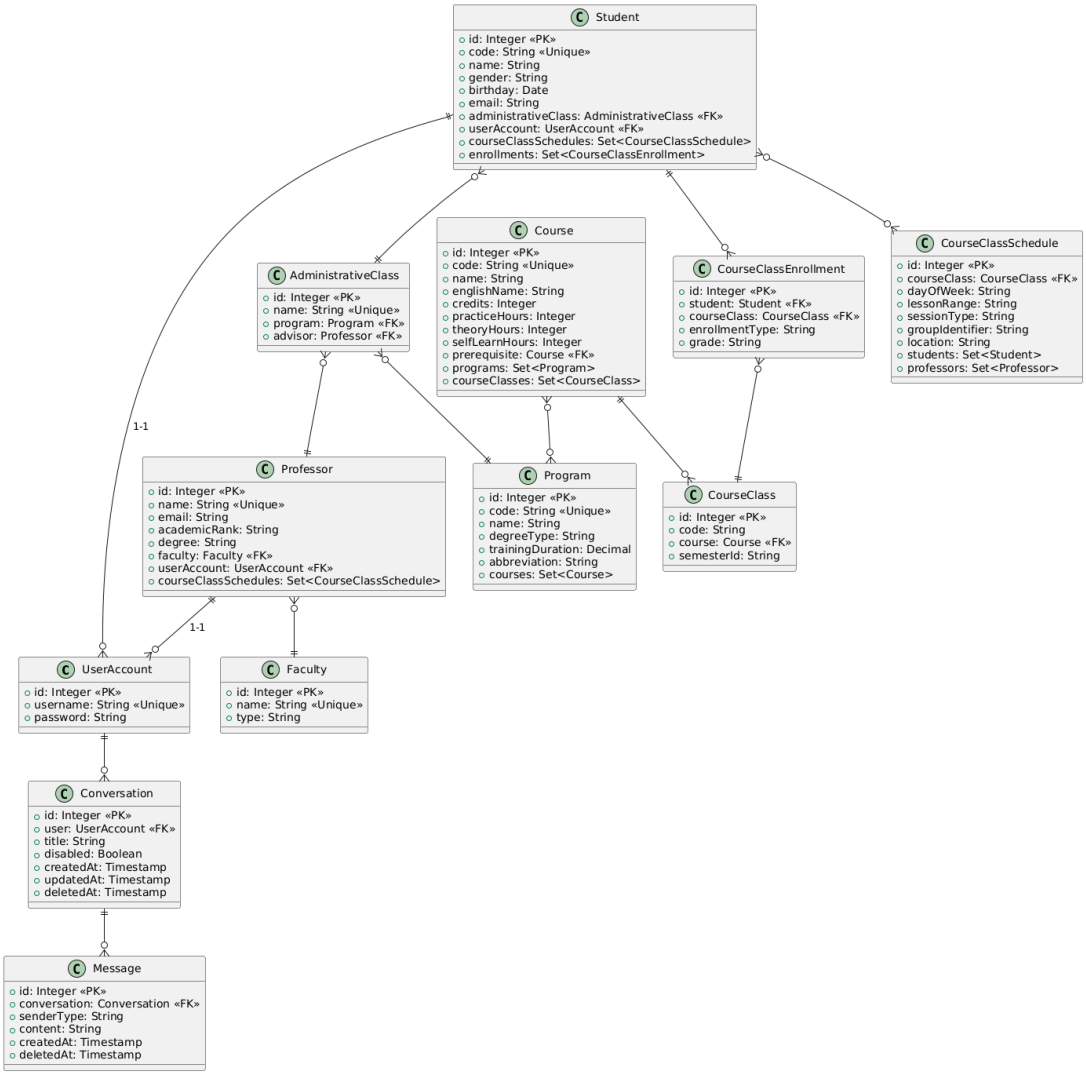


Hình 3.35. Biểu đồ lớp chính trong hệ thống

3.2.6. Thiết kế dữ liệu

Phần này mô tả chi tiết quá trình thiết kế dữ liệu của hệ thống, từ việc xác định các thực thể chính đến xây dựng lược đồ cơ sở dữ liệu quan hệ. Các thực thể chính, đại diện cho các đối tượng cốt lõi trong hệ thống, được minh họa rõ ràng thông qua biểu đồ lớp. Dựa trên các thực thể này, lược đồ quan hệ được phát triển để đảm bảo tính nhất quán, toàn vẹn dữ liệu và hiệu quả trong truy xuất. Cuối cùng, thiết kế cơ sở dữ liệu quan hệ được hoàn thiện, cung cấp nền tảng vững chắc cho việc triển khai hệ thống.

Biểu đồ lớp thể hiện các thực thể chính của hệ thống được trình bày trong Hình 3.36.

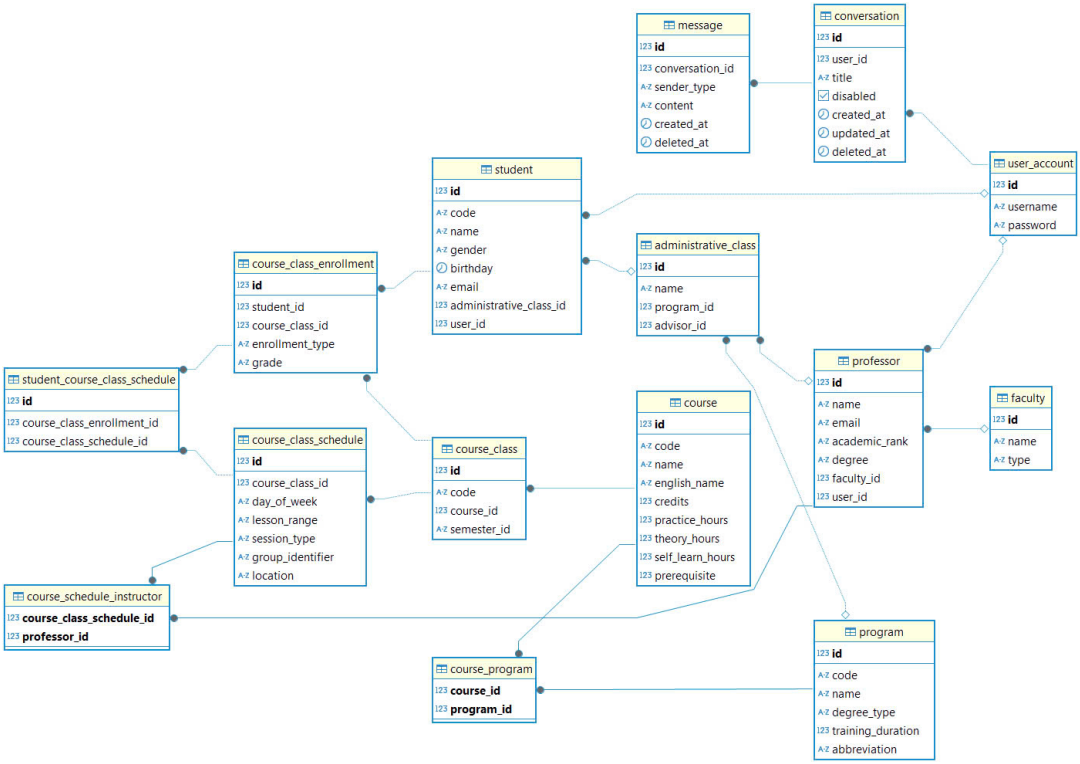


Hình 3.36. Biểu đồ thực thể chính trong hệ thống

Dựa trên các thực thể được xác định trong Hình 3.36, lược đồ quan hệ được thiết kế để ánh xạ các thực thể và mối quan hệ giữa chúng vào mô hình dữ liệu quan hệ. Lược đồ quan hệ được thể hiện trong Hình 3.37.

Hình 3.37. Lược đồ quan hệ

Từ lược đồ quan hệ, cơ sở dữ liệu quan hệ được thiết kế chi tiết, đảm bảo đáp ứng các yêu cầu về lưu trữ, truy vấn và quản lý dữ liệu của hệ thống. Thiết kế cơ sở dữ liệu quan hệ được minh họa trong Hình 3.39.



Hình 3.38. Thiết kế lược đồ cơ sở dữ liệu

3.3. Thiết kế tương tác với mô hình ngôn ngữ lớn

Hệ thống được thiết kế để tận dụng Mô hình Ngôn ngữ Lớn (LLM), cụ thể là thông qua API là `https://api.openai.com/v1/chat/completions` của OpenAI, nhằm mục đích hiểu các yêu cầu truy vấn bằng ngôn ngữ tự nhiên từ người dùng và hỗ trợ truy xuất thông tin một cách hiệu quả. Quá trình này bao gồm nhiều bước tương tác giữa các thành phần của hệ thống và LLM.

Luồng tương tác với LLM

1. **Tiếp nhận yêu cầu người dùng:** Người dùng (sinh viên hoặc giảng viên) nhập câu hỏi bằng ngôn ngữ tự nhiên thông qua giao diện người dùng (Lớp Biểu diễn).
2. **Chuyển tiếp yêu cầu:** Yêu cầu từ giao diện được chuyển đến `ChatbotController` trong Lớp Điều khiển, sau đó được xử lý bởi `ChatbotService` trong Lớp Dịch vụ.

3. **Tương tác với LLM qua AIProvider:** Yêu cầu này bao gồm nội dung câu hỏi của người dùng, lịch sử của cuộc hội thoại hiện tại, và một danh sách các ”công cụ” (tools) hoặc ”hàm” (functions) mà LLM có thể yêu cầu hệ thống thực thi. Các hàm này được quản lý và cung cấp thông tin mô tả (schema) bởi `FuncRegistry`.
4. **Xử lý yêu cầu bởi LLM:**
   - LLM của OpenAI phân tích ngữ nghĩa của câu hỏi và ngữ cảnh hội thoại.
   - Dựa trên phân tích, LLM quyết định hành động tiếp theo:
     + **Yêu cầu gọi hàm (Function Calling):** Nếu câu hỏi của người dùng có thể được giải quyết bằng một trong các hàm đã đăng ký (ví dụ: tra cứu lịch học, lấy thông tin điểm số), LLM sẽ trả về một cấu trúc dữ liệu (thường là JSON) chỉ định tên hàm cần gọi và các tham số đầu vào cho hàm đó.
     + **Yêu cầu sinh truy vấn SQL:** Đối với các yêu cầu truy xuất dữ liệu phức tạp hơn mà không có hàm cụ thể nào đáp ứng trực tiếp, LLM có thể được thiết kế để sinh ra một câu lệnh SQL tương ứng.
5. **Thực thi yêu cầu và thu thập dữ liệu:**
   - **Trường hợp gọi hàm:** Nếu LLM yêu cầu gọi hàm, `AIProvider` nhận phản hồi này. `ChatbotService` sử dụng `FuncRegistry` để xác định và thực thi hàm nghiệp vụ tương ứng (ví dụ: gọi phương thức trong `ScheduleService`, `GradeService`). Kết quả trả về từ hàm này (dữ liệu nghiệp vụ) được thu thập.
   - **Trường hợp sinh SQL:** Nếu LLM sinh ra câu lệnh SQL, `AIProvider` nhận câu lệnh này. `ChatbotService` chuyển câu lệnh SQL này đến `AuthorizationService` để kiểm tra và áp dụng các quy tắc phân quyền (như đã mô tả trong thiết kế phân quyền). Nếu hợp lệ, câu lệnh SQL được thực thi trên cơ sở dữ liệu (thông qua `SQLHdb`). Kết quả từ cơ sở dữ liệu được thu thập.
6. **LLM tạo phản hồi cuối cùng:**
   - Dữ liệu thu thập được (từ việc gọi hàm hoặc thực thi SQL) được gửi trở lại cho LLM (thông qua `AIProvider`) làm ngữ cảnh bổ sung.

– LLM sử dụng dữ liệu này để tổng hợp và tạo ra một câu trả lời hoàn chỉnh, mạch lạc bằng ngôn ngữ tự nhiên, phù hợp với câu hỏi ban đầu của người dùng.

7. **Hiển thị phản hồi cho người dùng:** `AIProvider` trả về câu trả lời cuối cùng từ LLM (thường dưới dạng một luồng dữ liệu - stream) cho `ChatbotService`. Dịch vụ này sau đó chuyển phản hồi đến `ChatbotController` để hiển thị trên giao diện người dùng, thường là từng phần một để cải thiện trải nghiệm.

3.4. Thiết kế phân quyền

3.4.1. Quy tắc phân quyền

Với việc dựa vào kết quả sinh từ LLM, đặc biệt là cho phép truy vấn bằng SQL sinh ra bởi mô hình ngôn ngữ lớn, đòi hỏi chúng ta cần có một cơ chế phân quyền hợp lý và hiệu quả. Vì vậy, khóa luận đề xuất một hướng tiếp cận phân quyền trong bài toán hiện tại như sau. Chúng ta sẽ phân quyền theo hai vai trò chính của người dùng là giảng viên và học sinh, đảm bảo áp dụng RBAC (Role-based access control)—phương pháp phân quyền theo vai trò. Do hệ thống của đề tài thao tác trực tiếp với cơ sở dữ liệu thường xuyên, đề tài lựa chọn phân quyền dựa trên các bảng (thay vì miền nghiệp vụ). Cụ thể, ta thực hiện phân quyền các bảng theo RLS (Row-Level Security) như sau:

– **Các bảng công khai** Các bảng sau được công khai với cả hai vai trò (giảng viên và học sinh): `falculty`, `program`, `course`, `course_program`, `course_class_schedule`, `course_schedule_instructor`.

– **Bảng student** Mỗi sinh viên chỉ được truy cập dữ liệu thông tin cá nhân của chính mình. Giảng viên có thể truy cập dữ liệu thông tin cá nhân của học sinh trong lớp mình giảng dạy hoặc lớp mình cố vấn.

– **Bảng professor** Mỗi sinh viên chỉ có thể truy cập dữ liệu thông tin cá nhân của giảng viên giảng dạy mình hoặc cố vấn mình. Giảng viên chỉ có thể truy cập dữ liệu thông tin cá nhân của chính mình.

– **Bảng administrative_class** Sinh viên và giảng viên chỉ có thể truy cập thông tin lớp hành chính của mình (dưới vai trò tham gia hoặc cố vấn).

– **Bảng course_class_enrollment** Sinh viên chỉ có thể xem thông tin đăng ký và điểm các môn của mình. Giảng viên có thể xem thông tin của học sinh mình cố vấn hoặc trong lớp mình giảng dạy, nhưng không được xem thông tin học sinh tham gia của lớp mình không đứng lớp.

– **Bảng student_course_class_schedule** Sinh viên chỉ có thể xem lịch học của chính mình. Giảng viên có thể xem lịch học và sinh viên của các môn mình đứng lớp hoặc của sinh viên mình cố vấn.

– **Các bảng còn lại** Các bảng `user_account`, `message`, `conversation` không thuộc nghiệp vụ của trường đại học, nên mỗi người dùng chỉ có thể truy cập vào thông tin của chính mình.

Để minh họa rõ hơn, bảng sau tóm tắt các chính sách phân quyền theo RLS:

Bảng 3.3. Tóm tắt chính sách phân quyền theo RLS

Bảng	**Quyền truy cập của sinh viên**	**Quyền truy cập của giảng viên**
falculty,program, course,course_program, course_class_schedule, course_schedule_instructor	Công khai	Công khai
student	Chỉ truy cập thông tin cá nhân của chính mình	Truy cập thông tin học sinh trong lớp giảng dạy hoặc cố vấn
professor	Chỉ truy cập thông tin giảng viên giảng dạy mình hoặc cố vấn mình	Chỉ truy cập thông tin cá nhân của chính mình
administrative_class	Chỉ truy cập thông tin lớp hành chính của mình	Chỉ truy cập thông tin lớp hành chính mình cố vấn
course_class_enrollment	Chỉ xem thông tin đăng ký và điểm của mình	Xem thông tin học sinh mình cố vấn hoặc trong lớp giảng dạy
student_course_class _schedule	Chỉ xem lịch học của mình	Xem lịch học và sinh viên của lớp mình đứng lớp hoặc sinh viên mình cố vấn
user_account,message, conversation	Chỉ truy cập thông tin của chính mình	Chỉ truy cập thông tin của chính mình

3.4.2. Triển khai quy tắc phân quyền

Ta sẽ phân quyền sau khi nhận lời gọi hàm từ LLM, vì vậy ta sẽ ra làm 2 phần: hàm nghiệp vụ, câu truy vấn. Hàm nghiệp vụ là các hàm logic của lớp dịch vụ đã được đăng ký với LLM như ở phần trên. Câu truy vấn là một hàm đặc biệt được đăng ký giúp cho LLM có khả năng sinh ra và gọi câu truy vấn SQL. Mã nguồn 3.1 là định nghĩa của hàm truy vấn đó.

```
func (db *SQLHDb) ExecuteQuery(ctx context.Context, req
   QueryRequest) (*QueryResult, error)
```

Mã nguồn 3.1. ExecuteQuery

Các hàm nghiệp vụ định sẵn có định nghĩa khá rõ ràng với yêu cầu đầu vào chặt chẽ nên việc phân quyền không gặp nhiều khó khăn. Ta có thể trích xuất id và vai trò của người dùng, sau đó dựa vào thông tin đó để phân quyền cho người dùng.

Với câu truy vấn, mọi thứ trở nên phức tạp hơn khi cấu trúc của câu truy vấn rất đa dạng. Khóa luận đã triển khai thử các hướng tiếp cận sau đây.

3.4.2.1. Hướng tiếp cận ngây thơ

Cách đơn giản nhất ta có thể xử lý chuỗi (string) để kiểm tra xem câu SQL đầu vào có chứa điều kiện phân quyền đúng không. Ví dụ một sinh viên với id = 1, khi truy vấn thông tin cá nhân cần có điều kiện id=1, hoặc student_id=1.

```
SELECT * FROM student WHERE id = 1
```

Mã nguồn 3.2. Câu truy vấn đơn giản hợp lệ

Câu truy vấn như trong mã nguồn 3.2 sẽ hợp lệ. Tuy nhiên, điều này có thể dễ dàng bị vượt qua bằng cách sử dụng câu truy vấn như Mã nguồn 3.3.

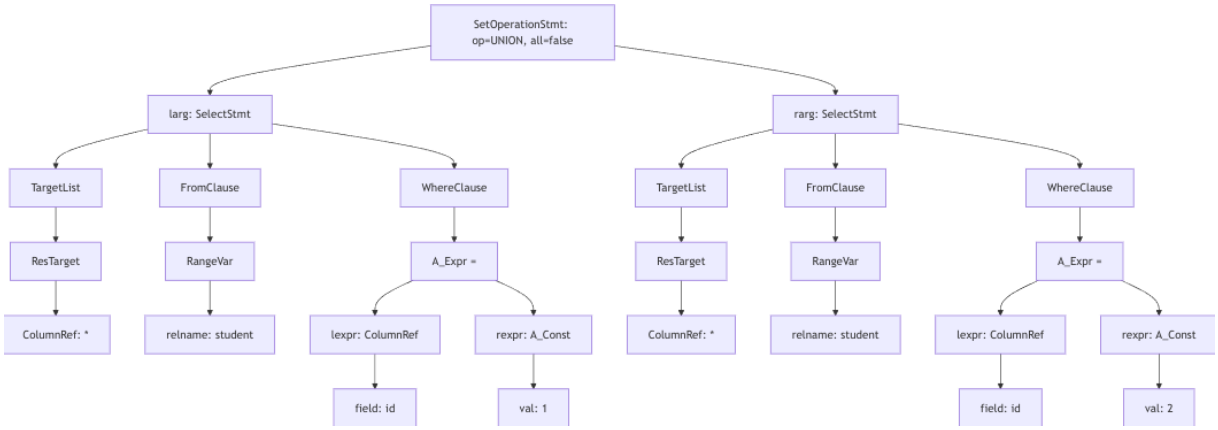
```
SELECT * FROM student WHERE id = 1
UNION
SELECT * FROM student WHERE id = 2
```

Mã nguồn 3.3. Câu truy vấn vượt qua phân quyền

Câu truy vấn trong Mã nguồn **??** cho thấy cách tiếp cận này là không khả thi

3.4.2.2. Hướng tiếp cận phân tích câu truy vấn

Để có thể phân quyền được câu truy vấn, hệ thống của ta cần phải hiểu được câu truy vấn. Vậy làm sao để hiểu được câu truy vấn? Lấy ý tưởng từ chương trình dịch (compiler) có thể hiểu được ngôn ngữ lập trình và thực thi các câu lệnh tương ứng và SQL có thể coi là một ngôn ngữ lập trình truy vấn. Từ đó, khóa luận đề xuất ra hướng tiếp cận này. Giống như các ngôn ngữ lập trình khác, SQL cũng có 1 bộ phân tích (parser) giúp phân giải chuỗi ngôn ngữ đầu vào thành một cây cú pháp (Abstract Syntax Tree). Ví dụ câu truy vấn với UNION ở phần trước có thể phân tích như sau



Hình 3.39. Câu truy vấn SELECT sau khi phân giải

Từ đây sau khi có cây cú pháp, ta có thể thực hiện phân quyền dựa trên đó. Ví dụ với câu truy vấn UNION trên, sau khi phân giải ta biết đây là một câu select với UNION ta sẽ phân quyền bằng cách bắt cả 2 vế đều phải thỏa mãn điều kiện phân quyền thì cả câu mới thỏa mãn.

Bắt đầu với ý tưởng đó, khóa luận sử dụng pg_query một thư viện Golang đóng gói lại parser chính thức của cơ sở dữ liệu Postgres để phân tích câu truy vấn. Vì có rất nhiều loại nốt khác nhau trong cây cú pháp (trong Postgres có hơn 100 loại nốt trải dài ở các pha khác nhau, pha phân giải, phân tích và thực thi), khóa luận đã sàng lọc và đưa ra là ở pha phân giải ta chỉ cần quan tâm đến một số nốt chính sau:

– **Node_SelectStmt**: Nốt thể hiện 1 câu truy vấn SELECT

- **Node_WithClause**: Nốt thể hiện câu lệnh WITH (CTE)
- **Node_RangeVar**: Nốt thể hiện một bảng trong cơ sở dữ liệu, thường nằm trong câu lệnh FROM
- **Node_FromExpr**: Nốt thể hiện 1 câu FROM
- **Node_SubLink**: Nốt thể hiện 1 câu truy vấn con (subselect) nhưng nằm trong câu điều kiện WHERE hoặc ON
- **Node_RangeSubselect**: Nốt thể hiện một câu truy vấn con nhưng nằm trong FROM
- **Node_JoinExpr**: Nốt thể hiện 1 câu lệnh JOIN
- **Node_BoolExpr**: Nốt thể hiện một biểu thức điều kiện AND, OR, NOT
- **Node_AExpr**: Nốt thể hiện một phép so sánh cơ bản như a = 1, b > c, ...
- **Node_ColumnRef**: Nốt thể hiện một cột của bảng
- **Alias**: Dùng để tạo một biệt hiệu, thường dùng cho câu truy vấn con (SubSelect) hoặc điều kiện Join và CTE

Ngoài ra, với các nốt khác chúng ta có thể mặc định là nó không đóng góp vào việc phân quyền câu truy vấn. Ta cũng có 1 vài nốt khác cần lưu ý là nốt con của các nốt trên nhưng sẽ nói ở phần sau.

Để thực hiện duyệt qua cây truy vấn và xác thực phân quyền, ta cần kiểm tra rằng trong tất cả các bảng lấy ra đã thỏa mãn chính sách phân quyền chưa. Ta dùng một hàm đệ quy sau như trong Mã nguồn 3.4.

Khi duyệt cây, ta có thể bắt gặp rất nhiều trường hợp khác nhau, ví dụ như một câu select khi đang trong BoolExpr của điều kiện WHERE. Vì vậy ta tạo hàm AuthorizeNode như ở trên để có thể gọi đệ quy trong các trường hợp đó. Và trách nhiệm phân giải xem nốt đó là loại gì để gọi tới cài đặt cụ thể sẽ là của hàm AuthorizeNode.

Ta thấy có 2 loại nốt chính: nốt cung cấp dữ liệu, nốt lọc dữ liệu. Nốt cung cấp dữ liệu thường đến từ câu lệnh FROM, WITH, JOIN. Nốt lọc dữ liệu là các nốt trong câu điều kiện WHERE, ON. Do đó để có thể xác thực câu truy vấn có hợp lệ không, ta cần truyền thông tin từ các nốt cung cấp (các nốt nào đã được chọn ra) tới các nốt lọc dữ liệu. Để làm được điều đó, ta sẽ biểu diễn bảng trong bộ nhớ dưới dạng một struct và truyền nó đi như 1 tham số (tables trong hàm AuthorizeNode ở trên).

```
func (s *AuthorizationService) AuthorizeNode(node
   *pgquery.Node, tables *om.OrderedMap[string,
   *TableInfoV2], neg bool, userInfo UserContext)
   (*AuthorizationResult, error) {
    if node.GetNode() == nil {
        return nil, errors.New("node is nil")
    }
    switch node.GetNode().(type) {
    case *pgquery.Node_SelectStmt:
        return
           s.authorizeSelectStmt(node.GetSelectStmt(),
           tables, neg, userInfo)
    case *pgquery.Node_BoolExpr:
        return s.authorizeBoolExpr(node.GetBoolExpr(),
           tables, neg, userInfo)
    case *pgquery.Node_AExpr:
        return s.authorizeAExpr(node.GetAExpr(), tables,
           neg, userInfo)
    case *pgquery.Node_SubLink:
        return s.authorizeSubLink(node.GetSubLink(),
           tables, neg, userInfo)
    case *pgquery.Node_JoinExpr:
        return s.authorizeJoinExpr(node.GetJoinExpr(),
           tables, neg, userInfo)
    case *pgquery.Node_FromExpr:
        return s.authorizeFromExpr(node.GetFromExpr(),
           tables, neg, userInfo)
    case *pgquery.Node_RangeVar:
        return s.authorizeRangeVar(node.GetRangeVar(),
           tables, userInfo)
    case *pgquery.Node_RangeSubselect:
        return
           s.authorizeRangeSubselect(node.GetRangeSubselect(),
           tables, neg, userInfo)
    case *pgquery.Node_WithClause:
        return
           s.authorizeWithClause(node.GetWithClause(),
           tables, neg, userInfo)
    }
    return &AuthorizationResult{
        Authorized: false,
        Tables:     tables,
    }, nil
}
```


Mã nguồn 3.4. Hàm đệ quy kiểm tra phân quyền

Với hướng tiếp cận đầu tiên, khóa luận sử dụng cấu trúc như trong Mã nguồn 3.5 để biểu diễn một bảng trong cơ sở dữ liệu.

```
type TableInfo struct {
    Name          string
    Alias         string              // Empty if no alias
    ColumnsAlias  map[string]string // Map of alias to
       column name
    HasStar       bool
}
```

Mã nguồn 3.5. Cấu trúc TableInfo

Tuy nhiên cách tiếp cận này nhanh chóng gặp phải hạn chế, khi cần phải xử lý với các biệt hiệu (Alias). Ví dụ như câu truy vấn SQL trong Mã nguồn 3.6:

```
select * from (student s
     left join administrative_class ac on
        s.administrative_class_id = ac.id) mytable(my_id)
where my_id = 1;
```

Mã nguồn 3.6. Câu truy vấn SQL với biệt hiệu

Với lưu ý rằng mytable(my_id) là nốt Alias với định dạng như trong Mã nguồn 3.7:

```
type Alias struct {
    Aliasname string
    Colnames  []*Node
}
```

Mã nguồn 3.7. Cấu trúc Alias

Câu SELECT trong Mã nguồn 3.6 hoàn toàn là một câu lệnh hợp lệ, tuy nhiên việc xử lý ánh xạ giữa biểu thức bên trong và câu lệnh điều kiện gặp nhiều khó khăn. Chúng ta cần ánh xạ đúng các cột của biểu thức ở trong với alias nhưng việc các cột nào được chọn lại nằm trong TargetList của câu Select. Sau nhiều nỗ lực, hướng tiếp cận khác nhau, khóa luận đi đến hướng tiếp cận sử dụng cấu trúc như trong Mã nguồn 3.8:

```
type TableInfoV2 struct {
    Name string
    // Columns is a map of alias to ColumnInfo
    Columns *om.OrderedMap[string, *ColumnInfo]
    Alias string
    // UnAuthorizedTables is a map of alias to array of
       TableInfo
    // UnAuthorizedTables is a map of alias to TableInfo
    UnAuthorizedTables *om.OrderedMap[string,
       *TableInfoV2] // Consider use ordered map if needed
    Authorized         bool
    IsDatabase         bool
}
```

Mã nguồn 3.8. Cấu trúc TableInfoV2

Trong đó TableInfoV2 thay vì biểu diễn một bảng thật trong cơ sở dữ liệu, sẽ biểu diễn một bảng ảo (virtual table) trong bộ nhớ trong phạm vi hiện tại. Ý tưởng này diễn ra hết sức tự nhiên, vì khi ta chọn 1 hoặc nhiều bảng thì cơ sở dữ liệu luôn gộp nó thành 1 bảng tổng thể để trả kết quả ra ngoài. Do đó, qua mỗi lần được ánh xạ bởi biệt hiệu, ta sẽ tạo ra một bảng ảo mới là hợp của những bảng ở trong nó.

TableInfoV2 ngoài trường Name, Alias ra thì sẽ có một trường không thể thiếu là Columns. Trường này thể hiện các cột hiện tại của bảng ảo (thường được ánh xạ bởi TargetList của câu SELECT và Colnames của Alias). Các cột này được thể hiện qua struct như trong Mã nguồn 3.9:

```
type ColumnInfo struct {
    Name        string
    SourceTable *TableInfoV2
}
```

Mã nguồn 3.9. Cấu trúc ColumnInfo

SourceTable ám chỉ việc cột này được ánh xạ từ bảng nào ở trong. Trường này rất quan trọng vì nó giúp ta có thể phân giải ra được nguồn gốc sâu xa dữ liệu của cột hiện tại.

Ngoài ra, ta cũng có Authorized để thể hiện là bảng này đã được phân quyền hay chưa .Và ta cũng lưu UnAuthorizedTables, để có thể truy cập nhanh vào tất cả những bảng (bảng thật trong cơ sở dữ liệu) nào vẫn chưa được phân quyền khi duyệt cây trên câu truy vấn này.

Tiếp theo đây, khóa luận sẽ trình bày cụ thể thuật toán với từng nốt

3.4.2.3. Thuật toán phân quyền câu truy vấn SELECT

Sau khi đã có hàm `AuthorizeNode` để duyệt qua cây cú pháp một cách đệ quy, hàm `authorizeSelectStmt` sẽ chịu trách nhiệm xử lý cụ thể các câu truy vấn SELECT – loại câu truy vấn phổ biến và phức tạp nhất trong SQL. Hàm này sẽ kiểm tra từng thành phần của câu SELECT, từ mệnh đề WITH, các toán tử SET (UNION, INTERSECT, EXCEPT), đến mệnh đề FROM, WHERE, HAVING, và đảm bảo rằng tất cả các bảng liên quan đều thỏa mãn chính sách phân quyền.

Hàm `authorizeSelectStmt` nhận vào một đối tượng `SelectStmt` (đại diện cho câu truy vấn SELECT), một danh sách các bảng ảo (`tables`), một biến cờ `neg` (để xử lý các trường hợp phủ định trong điều kiện), và thông tin người dùng (`userInfo`). Kết quả trả về là một `AuthorizationResult`, chứa thông tin về việc câu truy vấn có được phân quyền hay không và danh sách các bảng ảo đã được cập nhật.

Dưới đây là các bước chính trong thuật toán:

1. **Khởi tạo kết quả phân quyền**
   Ban đầu, giả định rằng câu truy vấn được phân quyền (`Authorized: true`) và tạo một danh sách bảng ảo rỗng để lưu kết quả.
2. **Kiểm tra mệnh đề INTO**
   Nếu câu SELECT có mệnh đề INTO (thường dùng trong SELECT INTO), hệ thống không áp dụng phân quyền vì câu lệnh này không trả về dữ liệu trực tiếp cho người dùng, mà chỉ lưu vào một bảng tạm.
3. **Xử lý mệnh đề WITH (CTE)**
   Nếu câu SELECT có mệnh đề WITH, hệ thống sẽ duyệt qua từng CTE (Common Table Expression) và gọi đệ quy `AuthorizeNode` để phân quyền. Kết quả phân

quyền của từng CTE được hợp nhất vào danh sách bảng ảo tổng thể. Câu truy vấn chỉ được coi là hợp lệ nếu tất cả CTE đều được phân quyền.

4. **Xử lý toán tử SET**
   Các toán tử SET như UNION, INTERSECT, EXCEPT được xử lý riêng biệt:
   - **UNION**: Cả hai vế (trái và phải) của UNION đều phải được phân quyền. Danh sách các bảng chưa được phân quyền là hợp của các bảng chưa được phân quyền từ hai vế.
   - **INTERSECT**: Câu truy vấn được phép nếu ít nhất một vế được phân quyền. Danh sách bảng chưa được phân quyền là giao của hai vế.
   - **EXCEPT**: Chỉ xem xét vế bên trái, bỏ qua vế bên phải để đơn giản hóa, vì việc loại trừ dữ liệu không ảnh hưởng đến phân quyền của vế trái.
5. **Xử lý mệnh đề FROM**
   Hệ thống duyệt qua từng mục trong mệnh đề FROM (có thể là bảng, câu truy vấn con, hoặc JOIN) và gọi `AuthorizeNode` để phân quyền. Kết quả từ các mục này được hợp nhất vào danh sách bảng ảo tổng thể. Nếu bất kỳ mục nào không được phân quyền, toàn bộ câu truy vấn sẽ bị coi là không hợp lệ.
6. **Xử lý mệnh đề WHERE và HAVING**
   Nếu câu truy vấn chưa được phân quyền sau bước FROM, hệ thống sẽ kiểm tra các điều kiện trong WHERE và HAVING:
   - Gọi `AuthorizeNode` để phân quyền các biểu thức điều kiện.
   - Các điều kiện này có thể lọc bỏ dữ liệu từ các bảng chưa được phân quyền, từ đó cập nhật trạng thái phân quyền của bảng ảo.
   - Kết quả phân quyền cuối cùng được cập nhật dựa trên WHERE và HAVING.
7. **Lưu TargetList**
   Nếu câu SELECT có danh sách cột được chọn (`TargetList`), hệ thống lưu thông tin này để sử dụng trong việc ánh xạ biệt hiệu hoặc xử lý câu truy vấn con sau này.

3.4.2.4. Xử lý phân quyền cho câu lệnh JOIN

Hàm `authorizeJoinExpr` thực hiện các bước sau để phân quyền:

1. **Phân quyền cho các vế của JOIN**: Hàm gọi đệ quy `AuthorizeNode` để phân quyền cho cả vế trái và vế phải của JOIN. Kết quả phân quyền của hai vế được hợp nhất vào danh sách bảng ảo `curTables`.
2. **Xử lý trường hợp cả hai vế đều được phân quyền**: Nếu cả hai vế đều thỏa mãn chính sách phân quyền, hàm kiểm tra xem có biệt hiệu (`Alias`) được đặt cho JOIN hay không. Nếu có, một bảng ảo mới sẽ được tạo bằng cách gọi `createVirtualTableForAlias` để ánh xạ biệt hiệu này, và kết quả phân quyền được trả về với trạng thái `Authorized: true`.
3. **Xử lý JOIN với `USING` hoặc `NATURAL`**: Nếu JOIN sử dụng mệnh đề `USING` hoặc là `NATURAL JOIN`, Trạng thái phân quyền sẽ là `false`.
4. **Xử lý theo loại JOIN**: Tùy thuộc vào loại JOIN (`INNER`, `LEFT`, `RIGHT`, `FULL`), hệ thống áp dụng các quy tắc khác nhau để cập nhật danh sách các bảng chưa được phân quyền:
   - `JOIN_INNER`: Chỉ giữ lại các bảng chưa được phân quyền có mặt ở cả hai vế (giao của hai danh sách).
   - `JOIN_LEFT`: Bỏ qua các bảng chưa được phân quyền ở vế phải.
   - `JOIN_RIGHT`: Bỏ qua các bảng chưa được phân quyền ở vế trái.
   - `JOIN_FULL`: Giữ nguyên tất cả các bảng chưa được phân quyền từ cả hai vế.
   - Các loại JOIN không hỗ trợ sẽ trả về lỗi.
5. **Xử lý điều kiện JOIN (`Quals`)**: Nếu có điều kiện JOIN (ví dụ: `ON` clause), hàm gọi `AuthorizeNode` để phân quyền cho điều kiện này, sử dụng danh sách bảng ảo `curTables`. Kết quả phân quyền từ điều kiện sẽ được cập nhật vào `qualResult`.
6. **Tạo bảng ảo cho biệt hiệu (nếu có)**: Nếu JOIN có biệt hiệu, một bảng ảo mới được tạo dựa trên biệt hiệu và được gán vào danh sách bảng ảo của kết quả.
7. **Trả về kết quả**: Hàm trả về `qualResult`, chứa trạng thái phân quyền cuối cùng và danh sách bảng ảo đã được cập nhật.

3.4.2.5. Xử lý phân quyền cho câu truy vấn con (SubLink)

Hàm `authorizeSubLink` thực hiện một cách đơn giản nhưng hiệu quả do đặc điểm của nốt `SubLink`:

- **Gọi đệ quy**: Hàm gọi `AuthorizeNode` để phân quyền cho câu truy vấn con (`Subselect`) bên trong `SubLink`.
- **Kết quả**: Kết quả phân quyền từ `AuthorizeNode` được trả về trực tiếp, bao gồm trạng thái phân quyền và danh sách bảng ảo đã được cập nhật.

3.4.2.6. Xử lý phân quyền cho câu truy vấn con trong FROM (RangeSubselect)

Hàm `authorizeRangeSubselect` thực hiện các bước sau để phân quyền:

- **Phân quyền cho subquery**: Hàm gọi đệ quy tới `AuthorizeNode` để phân quyền cho câu truy vấn con (`Subquery`) bên trong `RangeSubselect`. Tham số `neg` được đặt thành `false`, vì subquery này không liên quan đến phủ định tại cấp độ này.
- **Xử lý biệt hiệu (`Alias`)**: Nếu subquery có biệt hiệu (được định nghĩa qua `Alias`), hàm tạo một bảng ảo mới bằng cách gọi `createVirtualTableForAlias`. Bảng ảo này được xây dựng dựa trên danh sách cột mục tiêu (`TargetList`) và danh sách bảng ảo hiện tại. Sau khi tạo, `TargetList` được đặt lại thành `nil` để tránh trùng lặp hoặc xung đột trong quá trình xử lý.
- **Trả về kết quả**: Hàm trả về kết quả phân quyền từ `AuthorizeNode`, với danh sách bảng ảo đã được cập nhật nếu có biệt hiệu.

3.4.2.7. Xử lý phân quyền cho tham chiếu bảng (RangeVar)

Nốt RangeVar là nốt cơ sở nhất trong các nốt cung cấp dữ liệu. Hàm authorizeRangeVar sẽ xây dựng bảng này trong cây của TableInfoV2 và xác định xem bảng này có cần được phân quyền không. Hàm `authorizeRangeVar` thực hiện các bước sau để phân quyền:

1. **Lấy danh sách cột từ lược đồ**: Hàm sử dụng `schemaService.GetColumns` để lấy tất cả các cột của bảng và thêm chúng vào danh sách `Columns` của `TableInfoV2`. Mỗi cột được biểu diễn bằng một `ColumnInfo`, với thông tin về tên cột và nguồn gốc từ chính bảng hiện tại.
2. **Xử lý biệt hiệu (`Alias`)**: Nếu `RangeVar` có biệt hiệu được định nghĩa, giá trị `Alias` của `TableInfoV2` được cập nhật thành tên biệt hiệu này, thay vì sử dụng tên bảng gốc. Bảng ảo `TableInfoV2` được thêm vào danh sách `tables` bằng cách sử dụng `MergeMaps`, với khóa là tên biệt hiệu của bảng.
3. **Trả về kết quả**: Hàm trả về một `AuthorizationResult` với trạng thái phân quyền là true nếu bảng là bảng công khai.

3.4.2.8. Xử lý phân quyền cho mệnh đề WITH (WithClause)

Hàm `authorizeWithClause` thực hiện các bước sau để phân quyền:

1. **Khởi tạo kết quả phân quyền**: Hàm tạo một đối tượng `AuthorizationResult` với trạng thái ban đầu là `Authorized: true` và một danh sách bảng ảo rỗng để lưu kết quả.
2. **Duyệt qua các CTE**: Hàm lặp qua tất cả các CTE được định nghĩa trong `WithClause`. Với mỗi CTE, hàm gọi đệ quy `AuthorizeNode` để phân quyền, truyền vào danh sách bảng ảo hiện tại, cờ `neg`, và thông tin người dùng.
3. **Hợp nhất kết quả phân quyền**: Kết quả phân quyền từ mỗi CTE được hợp nhất vào danh sách bảng ảo tổng thể bằng cách sử dụng `MergeMaps`. Trạng thái phân quyền tổng thể được cập nhật bằng cách sử dụng toán tử logic `AND`, nghĩa là toàn bộ `WithClause` chỉ được coi là hợp lệ nếu tất cả các CTE đều được phân quyền.
4. **Trả về kết quả**: Sau khi xử lý tất cả các CTE, hàm trả về `AuthorizationResult` với trạng thái phân quyền và danh sách bảng ảo đã được cập nhật.

3.4.2.9. Xử lý phân quyền cho biểu thức điều kiện (BoolExpr)

Điều kiện NOT của biểu thức `BoolExpr` là một điều kiện thú vị. Khi nó yêu cầu biểu thức trong đó chọn tất cả dữ liệu không được phân quyền. Chính vì lẽ đó, ta cần

phải thêm vào một tham số `neg` cho hàm `AuthorizeNode` để đảo ngược điều kiện cần được phân quyền, ví dụ từ `student_id = 1` thành `student_id != 1`.

Hàm `authorizeBoolExpr` thực hiện các bước sau để phân quyền:

1. **Xử lý theo loại biểu thức logic**: Hàm kiểm tra loại biểu thức logic (`Boolop`) và xử lý tương ứng:
   - `OR_EXPR`: Với biểu thức `OR`, hàm duyệt qua từng biểu thức con và gọi `AuthorizeNode` để phân quyền. Danh sách các bảng chưa được phân quyền từ các biểu thức con được hợp nhất bằng cách lấy hợp của chúng.
   - `AND_EXPR`: Với biểu thức `AND`, hàm cũng duyệt qua từng biểu thức con và gọi `AuthorizeNode`. Tuy nhiên, danh sách các bảng chưa được phân quyền được cập nhật bằng cách lấy giao của chúng, nghĩa là chỉ giữ lại các bảng chưa được phân quyền xuất hiện ở tất cả các biểu thức con.
   - `NOT_EXPR`: Với biểu thức `NOT`, hàm yêu cầu đúng một biểu thức con. Hàm gọi `AuthorizeNode` với cờ `neg` được đảo ngược, và trạng thái phân quyền cùng danh sách bảng được cập nhật từ kết quả của biểu thức con.
2. **Xác định trạng thái phân quyền cuối cùng**: Sau khi xử lý các biểu thức con, hàm duyệt qua danh sách bảng ảo để xác định trạng thái phân quyền tổng thể. Đối với các bảng thật trong cơ sở dữ liệu, trạng thái `Authorized` của bảng được kiểm tra. Đối với các bảng ảo, hàm kiểm tra xem danh sách bảng chưa được phân quyền của chúng có rỗng hay không.
3. **Trả về kết quả**: Hàm trả về `AuthorizationResult` với trạng thái phân quyền và danh sách bảng ảo đã được cập nhật.

3.4.2.10. Xử lý phân quyền cho biểu thức so sánh (AExpr)

Hàm `authorizeAExpr` sử dụng các chính sách RLS để xác định xem một biểu thức so sánh có hợp lệ hay không. Các điều kiện RLS được hỗ trợ bao gồm:

- **Các bảng công khai**: Các bảng như `program`, `semester`, `course`, `course_program`, `course_class`, `course_class_schedule`,

course_schedule_instructor, và faculty được coi là công khai và có thể truy cập bởi tất cả các vai trò mà không cần điều kiện lọc.

– **Phân quyền cho vai trò student**:

  + student: Chỉ cho phép truy cập các hàng có id trùng với studentInfo.ID của sinh viên.

  + administrative_class: Chỉ cho phép truy cập các hàng có id trùng với studentInfo.AdministrativeClassID.

  + course_class_enrollment_id: Lọc theo student_id trùng với studentInfo.ID.

  + student_course_class_schedule: Lọc theo student_id trùng với studentInfo.ID.

  + student_scholarship: Lọc theo student_id trùng với studentInfo.ID.

– **Phân quyền cho vai trò professor**:

  + professor: Chỉ cho phép truy cập các hàng có id trùng với professorInfo.ID của giảng viên.

  + administrative_class: Lọc theo id thuộc danh sách professorInfo.AdvisedClassIDs.

  + course_class_enrollment: Lọc theo course_class_id thuộc professorInfo.TaughtCourseClassIDs hoặc student_id thuộc professorInfo.AdvisedStudentIDs.

  + student_course_class_schedule: Lọc theo id thuộc professorInfo.TaughtScheduleIDs.

  + student: Lọc theo id thuộc professorInfo.AdvisedStudentIDs.

Hàm authorizeAExpr thực hiện các bước sau để phân quyền:

– **Xử lý theo loại biểu thức so sánh**: Hàm kiểm tra loại biểu thức (Kind) và xử lý tương ứng:

  + AEXPR_OP: Đây là trường hợp so sánh cơ bản như a op b. Hàm chỉ cho phép các biểu thức dạng cột so sánh với một giá trị hằng (ví dụ: user_id

= id). Nếu điều kiện này được đáp ứng và biểu thức thỏa mãn chính sách phân quyền, các bảng liên quan sẽ được đánh dấu là đã phân quyền. Hàm cũng hỗ trợ trường hợp RowExpr = SubLink, ví dụ (t.user_id, t.col1) = (SELECT ...), bằng cách kiểm tra từng cột trong RowExpr và phân quyền cho subquery ở vế phải.

+ AEXPR_IN: Đây là trường hợp a IN (b, c, d). Hàm chỉ cho phép dạng authorizeColumn IN (id), với danh sách giá trị bên phải chỉ chứa một phần tử. Nếu điều kiện này được đáp ứng và biểu thức thỏa mãn chính sách phân quyền, các bảng liên quan sẽ được cập nhật trạng thái phân quyền.

– **Xác định trạng thái phân quyền cuối cùng**: Sau khi xử lý biểu thức, hàm duyệt qua danh sách bảng ảo để xác định trạng thái phân quyền tổng thể. Đối với bảng thật, trạng thái Authorized của bảng được kiểm tra. Đối với bảng ảo, hàm kiểm tra xem danh sách bảng chưa được phân quyền có rỗng hay không.

– **Trả về kết quả**: Hàm trả về AuthorizationResult với trạng thái phân quyền và danh sách bảng ảo đã được cập nhật.

3.5. Tổng kết chương

Chương vừa rồi khóa luận đã làm rõ việc phân tích, thiết kế ca sử dụng cho hệ thống từ việc đặc tả yêu cầu. Tiếp đó là thiết kế cơ sở dữ liệu và quy tắc phân quyền. Từ việc thiết kế trên khóa luận tiến hành cài đặt thực nghiệm và cụ thể thuật toán mà phần này sẽ được trình bày ở chương tiếp theo.

Chương 4.

Cài đặt và thực nghiệm

Chương này, khóa luận sẽ đưa ra cụ thể cách cài đặt từ các bước thiết kế trên . Cụ thể, chương này sẽ đi nhanh qua việc cài đặt các nghiệp vụ thông dụng và đi sâu vào cách hệ thống tương tác với mô hình ngôn ngữ lớn. Sau đó là thực nghiệm với một vài tính năng.

4.1. Cài đặt các nghiệp vụ cơ bản

Với các nghiệp vụ cơ bản, ta sẽ dựa theo thiết kế trên và triển khai cài đặt các lớp tương ứng. Với mỗi lớp dịch vụ (service), ta cũng triển khai cùng 1 interface và mẫu thiết kế Dependency injection giúp việc unit testing và integration testing được dễ dàng hơn. Vì một lớp dịch vụ có thể gọi tới các lớp dịch vụ khác. Khi triển khai interface, ta có thể dễ dàng cài đặt thêm mock service cho các lớp dịch vụ phụ thuộc, giúp ta có thể kiểm tra lớp dịch vụ hiện tại mà không cần phải khởi tạo toàn bộ các lớp dịch vụ phụ thuộc (các lớp dịch vụ này thậm chí có thể phụ thuộc vào các lớp dịch vụ khác nữa). Ví dụ, lớp dịch vụ CourseClassService phụ thuộc vào CourseService như trong Mã nguồn 4.1:

```
package courseclass

type ServiceImpl struct {
  courseService course.Service
  repo          Repository
}

func NewServiceImpl(repo Repository, courseService
   course.Service) *ServiceImpl {
  return &ServiceImpl{
    repo:          repo,
    courseService: courseService,
  }
}
```

Mã nguồn 4.1. Cấu trúc ServiceImpl

(Lưu ý theo triết lý của Golang, chúng ta không nên đặt tên lớp bắt đầu giống với tên gói, vì vậy ở phần code trên ta lược bỏ tiền đề CourseClass và để tên là ServiceImpl) Với ví dụ này, nếu ta không có interface cho courseService thì khi viết unit test cho courseClassService ta sẽ phải khởi tạo cả courseService thật. Và nếu có lỗi logic trong courseService, thì unit test trong courseClassService sẽ không hoạt động như mong muốn. Vì vậy ta sẽ cài đặt interface cho courseService như trong Mã nguồn 4.2:

```
package course

type GetCourseRequest struct {
  Code *string `json:"code,omitempty"`
  Name *string `json:"name,omitempty"`
}

type Service interface {
  GetCourse(ctx context.Context, req GetCourseRequest)
     (*GetCourseResponse, error)
}
```

Mã nguồn 4.2. Interface Service của Course

Và khi test, ta có thể cài đặt mock service như trong Mã nguồn 4.3:

```go
package courseclass

type MockCourseService struct {
}

func NewMockCourseService() *MockCourseService {
  return &MockCourseService{}
}

func (m *MockCourseService) GetCourse(ctx
   context.Context, req course.GetCourseRequest)
   (*course.GetCourseResponse, error) {
  return &course.GetCourseResponse{
    Code:        "CS101",
    Name:        "Computer Science 101",
    EnglishName: "Computer Science 101",
  }, nil
}
```

Mã nguồn 4.3. MockCourseService để kiểm thử

Khi đó, ta có thể viết test cho courseClassService mà không bị ảnh hưởng bởi logic của courseService và dữ liệu trong cơ sở dữ liệu. Ngoài ra để thực hiện điều đó, đoạn mã khởi tạo courseClassService trên cũng áp dụng Dependency Injection. Nghĩa là lớp dịch vụ sẽ không tự mình khởi tạo các lớp phụ thuộc mà nhận vào từ bên ngoài qua hàm khởi tạo (như ví dụ phía trước) hoặc hàm setter. Đây là ví dụ, khi không áp dụng Dependency Injection như trong Mã nguồn 4.4:

```go
package courseclass

func NewServiceImpl() *ServiceImpl {
  return &ServiceImpl{
    repo:          NewRepository(),
    courseService: course.NewServiceImpl(),
  }
}
```

Mã nguồn 4.4. Khởi tạo ServiceImpl không dùng Dependency Injection

Trong trường hợp này, chúng ta không có cách nào khởi tạo courseClassService sử dụng MockCourseService numar để test. Nếu dùng dependency injection, ta có thể khởi tạo như trong Mã nguồn 4.5:

```
    courseClassService :=
      NewServiceImpl(NewMockCourseService(),
      NewMockCourseClassRepository())
```

Mã nguồn 4.5. Khởi tạo với Dependency Injection

Ngoài ra, sử dụng dependency injection cũng giúp mã nguồn dễ thay đổi và mở rộng hơn khi muốn thêm các lớp dịch vụ mới,... Áp dụng các nguyên lý này, ta cài đặt các lớp nghiệp vụ khác theo biểu đồ tuần tự đã phân tích.

4.2. Tích hợp mô hình ngôn ngữ lớn

Ta sẽ tận dụng mô hình ngôn ngữ lớp của các tập đoàn công nghệ lớn hiện nay như OpenAI. Để đảm bảo tính linh hoạt và có thể thêm các mô hình khác trong tương lai hoặc tự triển khai một mô hình ngôn ngữ lớn, Khóa luận đề xuất cài đặt interface như trong Mã nguồn 4.6 với hai hàm chính.

```
package llm

type AIProvider interface {
  Complete(ctx context.Context, req CompletionRequest)
     (Message, error)
  StreamComplete(ctx context.Context, req
     CompletionRequest) (<-chan StreamChunk, error)
}
```

Mã nguồn 4.6. Interface AIProvider

Hàm Complete gọi tới bên cung cấp mô hình ngôn ngữ lớn một cách đồng bộ, tức là ta sẽ đợi mô hình sinh ra toàn bộ câu trả lời rồi mới trả về kết quả qua Message. Hàm này quan trọng, vì trong một số trường hợp, ta cần biết toàn bộ câu trả lời mới có thể tiếp tục luồng xử lý. Ví dụ như chờ mô hình sinh ra câu truy vấn SQL, ta cần toàn bộ câu SQL

mới có thể tiến hành phân quyền và truy cập dữ liệu. Trong khi đó, hàm StreamComplete dựa vào giao thức SSE (Server Sent Event) để tận dụng khả năng sinh ra từng token một của mô hình. Ta sẽ thiết lập một kênh (channel) dữ liệu với mô hình là StreamChunk. Khi mô hình sinh ra từ mới, mô hình sẽ truyền ngay dữ liệu đó vào kênh StreamChunk. Việc này thực sự hiệu quả khi truyền dữ liệu tới người dùng, và thực tế hệ thống chatbot đề xuất của đề tài cũng sử dụng kênh dữ liệu này để trả dữ liệu tới trình duyệt của người dùng cuối. Với việc có thể nhận được dữ liệu ngay khi có và liên tục sẽ làm tăng tính phản hồi (responsive), từ đó tăng trải nghiệm người dùng.

Vì chúng ta sẽ tận dụng tính năng function calling của các mô hình ngôn ngữ lớn, nên các lớp truyền dữ liệu (Data Transfer Object - DTO) cần được thiết kế để hỗ trợ đặc điểm này. Lớp truyền dữ liệu gửi lên LLM sẽ có dạng như trong Mã nguồn 4.7:

```
type CompletionRequest struct {
  Messages            []Message
  Model               string
  ResponseFormat      *ResponseFormat
  Tools               []Tool
  FunctionCallingMode FunctionCallingMode
}
type ResponseFormat struct {
  Type   ResponseFormatType `json:"type"`
  Schema *jsonschema.Schema `json:"schema,omitempty"`
  Name   string             `json:"name"`
}
```

Mã nguồn 4.7. Cấu trúc CompletionRequest và ResponseFormat

Trong đó, ResponseFormat sẽ định nghĩa cấu trúc trả về của tin nhắn. Điều này là tối quan trọng, vì dữ liệu trả về từ LLM thường không ổn định, do đó ta cần yêu cầu trả về với một định dạng định trước. Từ đó, ta có thể phân tích thông điệp trả về một cách chính xác và an toàn hơn. ResponseFormat sẽ tuân theo chuẩn của jsonschema.

Sau đó, ta sẽ tiến hành cài đặt AIProvider cho từng mô hình ngôn ngữ lớn cụ thể. Ví dụ như trong Mã nguồn 4.8:

```go
package llm

type OpenAIProvider struct {
  client *openai.Client
}

func NewOpenAIProvider(client *openai.Client)
   *OpenAIProvider {
  return &OpenAIProvider{client: client}
}
```

Mã nguồn 4.8. Cài đặt OpenAIProvider

Ngoài ra, để có thể sử dụng function calling hiệu quả, ta cần một cơ chế để LLM có thể nhận biết các service của hệ thống. Khóa luận đề xuất cài đặt lớp FuncRegistry như trong Mã nguồn 4.9:

```go
package llm

type FuncRegistry interface {
  Register(fn FuncDefinition)
  Execute(ctx context.Context, toolCall ToolCall)
     (string, error)
  GetFuncDefinitions() []FuncDefinition
}
```

Mã nguồn 4.9. Interface FuncRegistry

Với ba hàm chính. Hàm Register giúp các lớp dịch vụ có thể đăng ký hàm của mình với FuncRegistry - nơi hoạt động như một quản lý tập trung của các hàm trong hệ thống. Hàm GetFuncDefinitions giúp lấy ra các hàm đã được đăng ký trong hệ thống để có thể gửi lên LLM. Hiện tại, ta sẽ hỗ trợ một quỹ hàm duy nhất, tuy nhiên khi cần trong tương lai ta có thể mở rộng ra thành nhiều quỹ nếu muốn. Hàm Execute phụ trách việc gọi hàm sau khi nhận lời gọi từ LLM. Hàm này cần thiết vì để có thể đăng ký và gửi lên LLM, các hàm phải được định dạng lại dưới một định dạng chung với bản đăng ký như trong Mã nguồn 4.10.

```go
// FuncDefinition represents a function definition for a
   tool
type FuncDefinition struct {
  Name        string              `json:"name"`
  Description string
     `json:"description,omitempty"`
  Parameters  *jsonschema.Schema `json:"parameters"`
  Handler     FuncHandler
}
```

Mã nguồn 4.10. Cấu trúc FuncDefinition

Trong đó mỗi hàm sẽ được đóng gói lại bởi signature của hàm đó, đi kèm với tên và mô tả để LLM có thể hiểu. Cụ thể, mỗi khi cần đóng gói một hàm thành định dạng chung trên để có thể đăng ký, ta sẽ dùng hàm như trong Mã nguồn 4.11:

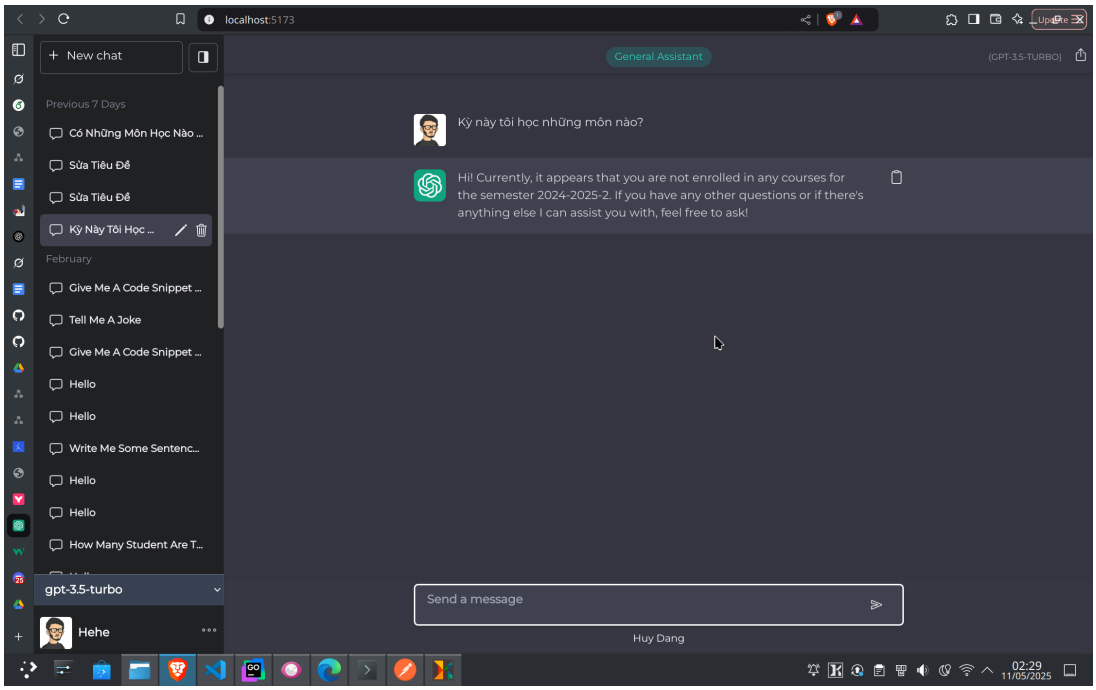
```go
    func FuncWrapper[T any, R any](name string,
       description string, fn func(ctx context.Context,
       args T) (R, error)) FuncDefinition
```

Mã nguồn 4.11. Hàm FuncWrapper

Sau đó, khi nhận lời gọi hàm từ LLM, hệ thống cần dùng Execute để phân giải ra hàm cụ thể (của service nào) sẽ được gọi và gọi hàm đó để lấy kết quả.

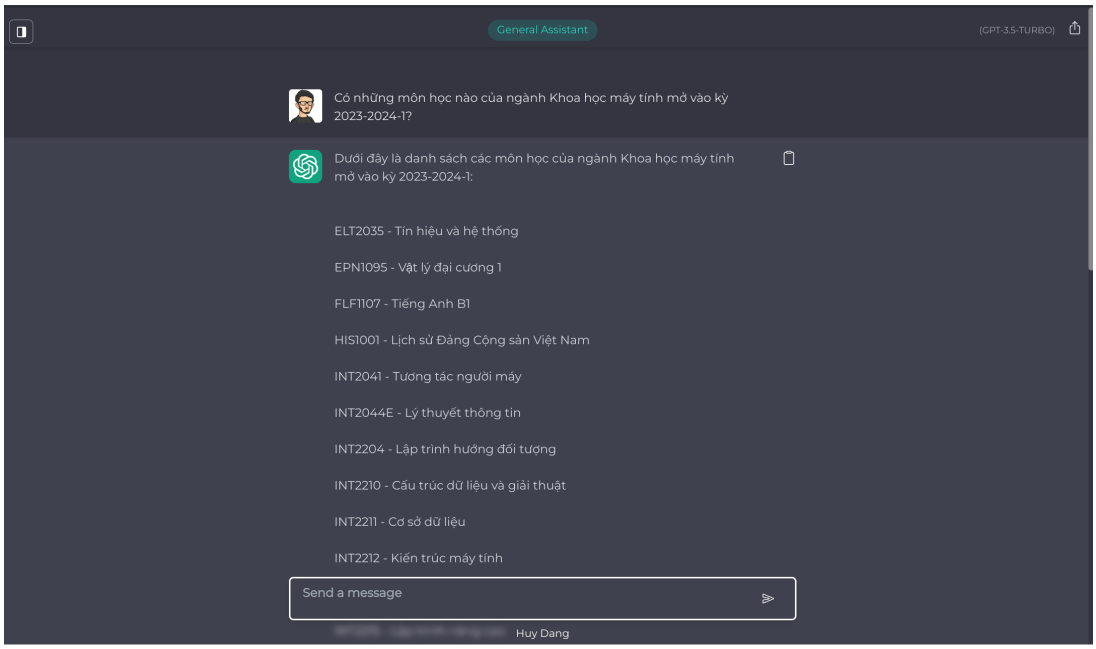
4.3. Thực nghiệm

Ta thực kiểm tra các tính năng của đề tài như dưới đây. Hình 4.1 là tính năng xem danh sách các cuộc hội thoại và lịch sử cuộc trò chuyện.



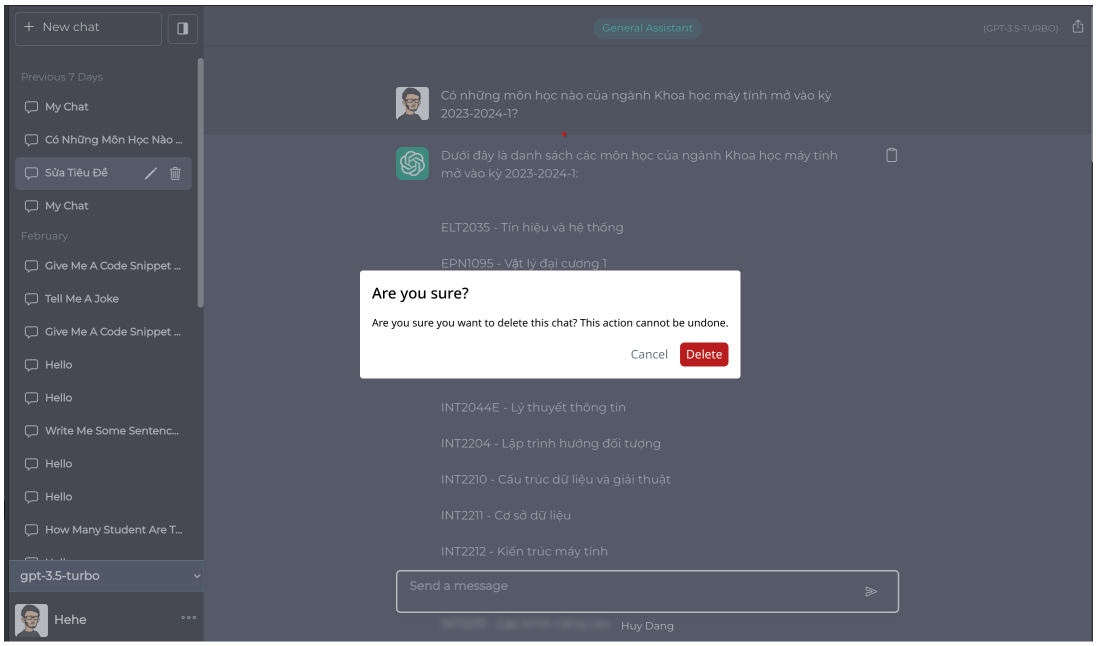
Hình 4.1. Xem danh sách các cuộc hội thoại và lịch sử cuộc trò chuyện

Tiếp đó, Hình 4.2 là tính năng sửa tiêu đề đoạn hội thoại



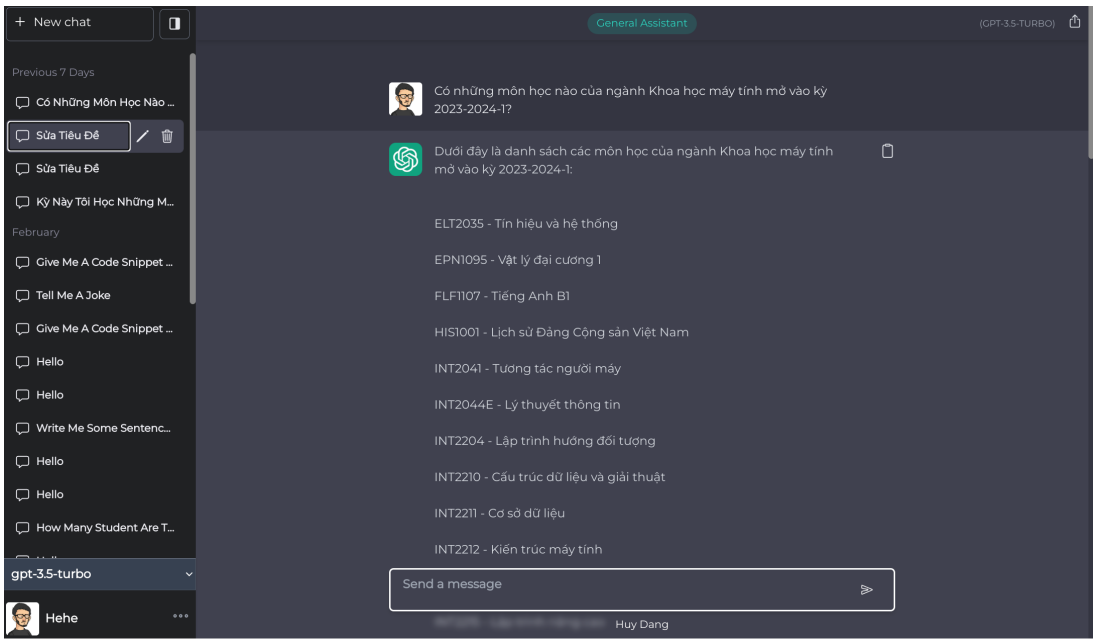
Hình 4.2. Sửa tiêu đề cuộc hội thoại

Hình 4.3 thể hiện tính nắng xóa đoạn hội thoại



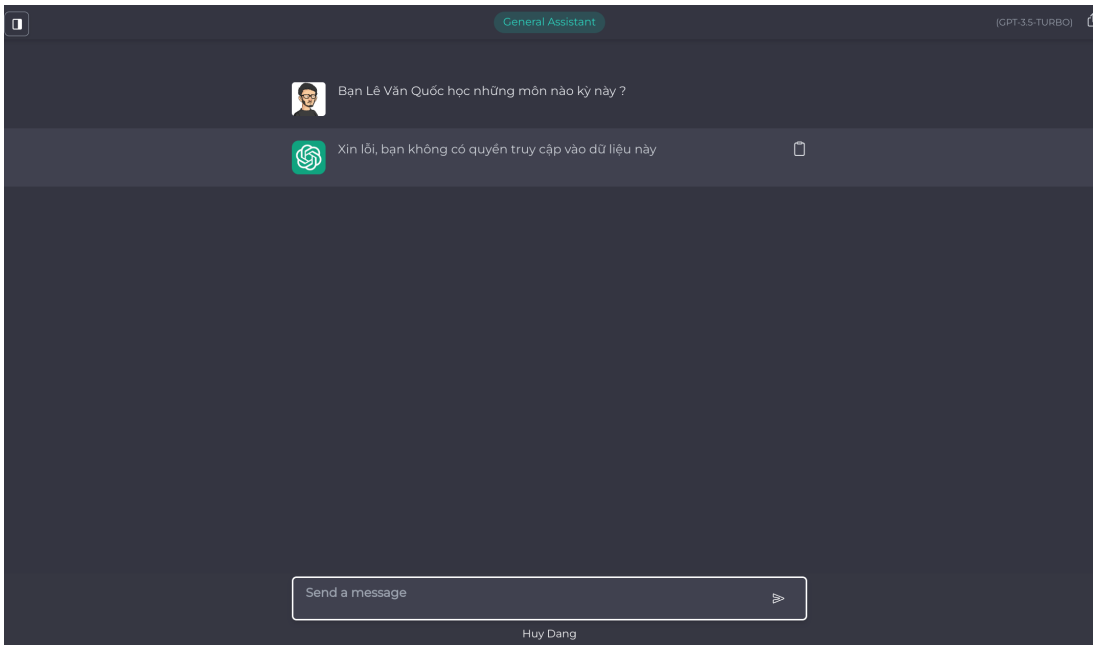
Hình 4.3. Xóa đoạn hội thoại

Hình 4.4 thể hiện tính nắng tra cứu thông tin



Hình 4.4. Tra cứu thông tin

Hình 4.5 thể hiện khi người dùng tra cứu thông tin không được phép



Hình 4.5. Tra cứu thông tin không được phép

4.4. Tổng kết chương

Chương này đã trình bày chi tiết quá trình cài đặt các thành phần chính của hệ thống chatbot, từ việc xây dựng các lớp dịch vụ theo nguyên lý Dependency Injection, tích hợp mô hình ngôn ngữ lớn (LLM) với cơ chế gọi hàm, cho đến việc triển khai các tính năng nghiệp vụ và kiểm tra thực nghiệm các chức năng tiêu biểu. Các ví dụ minh họa về giao diện và kết quả thực nghiệm đã cho thấy hệ thống đáp ứng tốt các yêu cầu về truy xuất thông tin, phân quyền và trải nghiệm người dùng. Những kết quả này là cơ sở quan trọng để đánh giá hiệu quả của giải pháp và đề xuất các hướng phát triển tiếp theo cho hệ thống trong môi trường thực tế.

Kết luận

Khóa luận đã trình bày quá trình nghiên cứu, thiết kế và xây dựng một hệ thống chatbot hỗ trợ nghiệp vụ đào tạo đại học dựa trên mô hình ngôn ngữ lớn (LLM) và cơ sở dữ liệu quan hệ. Hệ thống được phát triển với kiến trúc phân lớp rõ ràng, tận dụng các công nghệ hiện đại như React cho giao diện người dùng, Golang cho backend, và tích hợp OpenAI GPT để xử lý ngôn ngữ tự nhiên. Đặc biệt, giải pháp phân quyền dựa trên phân tích cú pháp SQL giúp đảm bảo an toàn dữ liệu và ngăn chặn truy cập trái phép.

Trong quá trình thực hiện, khóa luận đã đề xuất các phương pháp tiếp cận mới cho việc phân quyền truy vấn SQL, xây dựng cơ chế đóng gói và đăng ký hàm cho LLM, cũng như thiết kế các lớp dịch vụ theo nguyên lý Dependency Injection để tăng khả năng kiểm thử và mở rộng hệ thống. Việc tích hợp LLM giúp chatbot có thể hiểu và xử lý đa dạng các câu hỏi tự nhiên, đồng thời cơ chế phân quyền đảm bảo chỉ những người dùng hợp lệ mới có thể truy cập thông tin phù hợp với vai trò của mình.

Tuy nhiên, hệ thống vẫn còn một số hạn chế như khả năng xử lý các truy vấn phức tạp hoặc diễn đạt không rõ ràng còn phụ thuộc vào chất lượng của mô hình ngôn ngữ lớn và dữ liệu huấn luyện. Ngoài ra, việc mở rộng hệ thống để tích hợp thêm các nguồn dữ liệu hoặc chức năng mới cũng cần được nghiên cứu thêm để đảm bảo tính ổn định và bảo mật.

Trong tương lai, một hướng phát triển quan trọng của đề tài là tận dụng dữ liệu lịch sử hội thoại để tinh chỉnh (fine-tune) mô hình ngôn ngữ lớn. Việc thu thập và phân tích các đoạn hội thoại thực tế giữa người dùng và chatbot sẽ giúp nhận diện các dạng câu hỏi phổ biến, các lỗi hiểu sai hoặc các trường hợp hệ thống trả lời chưa tối ưu. Từ đó, dữ liệu này có thể được sử dụng để huấn luyện bổ sung cho mô hình, giúp chatbot hiểu tốt hơn ngữ cảnh nghiệp vụ, thích nghi với cách diễn đạt tự nhiên của sinh viên và giảng viên, cũng như cải thiện độ chính xác khi sinh truy vấn SQL hoặc trả lời các câu hỏi đặc thù. Hướng tiếp cận này không chỉ nâng cao chất lượng phản hồi mà còn giúp hệ thống ngày càng hoàn thiện, đáp ứng tốt hơn nhu cầu thực tế trong môi trường giáo dục đại học.

Tóm lại, khóa luận đã đóng góp một giải pháp thực tiễn và có ý nghĩa cho bài toán truy xuất thông tin trong đào tạo đại học, đồng thời mở ra tiềm năng ứng dụng các công nghệ AI hiện đại vào các hệ thống thông tin giáo dục tại Việt Nam.

Tài liệu tham khảo

[1] Martin Fowler, “Function Calling in LLMs,” https://martinfowler.com/articles/function-call-LLM.html.

[2] PostgreSQL Global Development Group, “PostgreSQL Documentation,” https://www.postgresql.org/docs/17/index.html.

[3] Red Gate, “Understanding SQL Query Parsing: Part 1,” https://www.red-gate.com/simple-talk/databases/oracle-databases/understanding-sql-query-parsing-part-1/.

[4] Sonra, “SQL Parser Use Cases,” https://sonra.io/sql-parser-use-cases/.

[5] PostgreSQL Source Code, “Parser Implementation (scan.l, gram.y),” https://github.com/postgres/postgres/tree/master/src/backend/parser.

[6] MDN Web Docs, “Server-sent events,” https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events.