|  |  |
| --- | --- |
| **ĐÀO KIM THẮNG** | **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC MỎ - ĐỊA CHẤT**  **---------------------------------------** |
|  |
| **A blue circle with white text and a globe  AI-generated content may be incorrect.** |
|  |
|  |
| **ĐÀO KIM THẮNG** |
| **KHOA HỌC MÁY TÍNH ỨNG DỤNG** |  |
| **ĐỒ ÁN TỐT NGHIỆP** |
|  |
|  |
|  |
| **NGÀNH CÔNG NGHỆ THÔNG TIN** |
| **MÃ SỐ: 7480201** |
|  |
|  |
|  |
| **TÊN ĐỀ TÀI:**  **XÂY DỰNG TRỢ LÝ AI CÁ NHÂN – NEURA**  **(DEVELOPING A PERSONAL AI ASSISTANT - NEURA)** |
|  |
| **KHÓA 66** |  |
|  |
|  |
|  |
| **Hà Nội - 2025** |
|  | |

|  |  |
| --- | --- |
| **BỘ GIÁO DỤC VÀ ĐÀO TẠO**  **TRƯỜNG ĐẠI HỌC MỎ - ĐỊA CHẤT**  **---------------------------------------** | |
|  | |
|  | |
|  | |
|  | |
| **ĐỒ ÁN TỐT NGHIỆP** | |
|  | |
|  | |
|  | |
|  | |
|  | |
| **CHUYÊN NGÀNH: KHOA HỌC MÁY TÍNH ỨNG DỤNG** | |
|  | |
|  | |
|  | |
|  | |
|  | |
|  | |
| GIÁO VIÊN HƯỚNG DẪN | SINH VIÊN THỰC HIỆN |
| *(Chữ ký)* | *(Chữ ký)* |
|  |  |
|  |  |
|  |  |
| PGS.TS. LÊ HỒNG ANH | ĐÀO KIM THẮNG |
|  | |
|  | |

|  |
| --- |
|  |
|  |
| **Hà Nội - 2025** |

# **LỜI CẢM ƠN**

Được sự đồng ý của bộ môn Khoa học máy tính khoa Công nghệ thông tin của trường Đại học Mỏ - Địa chất và giảng viên hướng dẫn khoa học: PGS.TS. Lê Hồng Anh, em đã thực hiện đồ án: “Xây dựng trợ lý AI cá nhân – Neura”.

Để hoàn thành đồ án này, em xin chân thành cảm ơn các thầy cô giảng viên trong khoa Công nghệ thông tin, đặc biệt là bộ môn Khoa học máy tính đã tận tình giảng dạy, hướng dẫn em trong suốt quá trình học tập và nghiên cứu ở trường.

Đặc biệt, xin gửi lời cảm ơn sâu sắc, chân thành tới thầy giáo Lê Hồng Anh, người đã trực tiếp hướng dẫn nghiên cứu khoa học cho em. Trong quá trình thực hiện đồ án, thầy đã tận tình chỉ bảo và truyền đạt những kinh nghiệm và kiến thức khoa học quý báu, đồng thời đưa ra các nhận xét, góp ý giúp em hoàn thành đồ án này một cách khoa học và thực tiễn hơn.

Xin chân thành cảm ơn các anh chị, bạn bè, các thành viên lớp Khoa học máy tính ứng dụng K66\_07C đã ủng hộ, giúp đỡ em cả về tinh thần lẫn kiến thức trong suốt quá trình học tập và thực hiện đồ án.

**MỤC LỤC**

[LỜI CẢM ƠN 3](#_Toc213881954)

[MỤC LỤC 4](#_Toc213881955)

[DANH MỤC TỪ VIẾT TẮT 6](#_Toc213881956)

[DANH MỤC BẢNG BIỂU 7](#_Toc213881957)

[DANH MỤC HÌNH VẼ 8](#_Toc213881958)

[CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI 10](#_Toc213881959)

[1.1. Tính cấp thiết của đề tài 10](#_Toc213881960)

[1.2. Mục tiêu của đề tài 11](#_Toc213881961)

[1.3. Nội dung nghiên cứu 11](#_Toc213881962)

[1.4. Phạm vi nghiên cứu 13](#_Toc213881963)

[1.5. Bố cục của đồ án 13](#_Toc213881964)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ 14](#_Toc213881965)

[2.1 Các kiến thức nền tảng về Xử lý ngôn ngữ tự nhiên (NLP) 14](#_Toc213881966)

[2.1.1 Khái niệm về NLP và ứng dụng 14](#_Toc213881967)

[2.1.2 Các thư viện và công cụ hỗ trợ NLP 16](#_Toc213881968)

[2.2 Tổng quan về trợ lý ảo 17](#_Toc213881969)

[2.2.1 Khái niệm và sự phát triển của trợ lý ảo 17](#_Toc213881970)

[2.2.2 Vai trò, ứng dụng trong đời sống và công việc 18](#_Toc213881971)

[2.3 Mô hình ngôn ngữ lớn (LLM) và Agent trong Trợ lý ảo 19](#_Toc213881972)

[2.3.1 Giới thiệu LLM 19](#_Toc213881973)

[2.3.2. Kiến trúc Agent và hệ thống Multi-Agent trong trợ lý ảo. 20](#_Toc213881974)

[2.4. Môi trường phát triển 22](#_Toc213881975)

[2.4.1. Ngôn ngữ lập trình Python 22](#_Toc213881976)

[2.4.2. Các thư viện hỗ trợ 23](#_Toc213881977)

[2.4.3. Các dịch vụ và API tích hợp 30](#_Toc213881978)

[2.4.4. Các phần mềm và công cụ 31](#_Toc213881979)

[CHƯƠNG 3. XÂY DỰNG TRỢ LÝ AI CÁ NHÂN 34](#_Toc213881980)

[3.1. Kiến trúc tổng thể trợ lý ảo 34](#_Toc213881983)

[3.2. Xây dựng Agent Manager 35](#_Toc213881984)

[3.2.1. Thành phần Agent Manager 36](#_Toc213881985)

[3.2.2. Cấu hình Prompt Template và Memory 37](#_Toc213881986)

[3.3. Xây dựng Agent Calendar 41](#_Toc213881987)

[3.3.1. Thành phần Agent Calendar 42](#_Toc213881988)

[3.3.2. Cấu hình Prompt Template và Memory 58](#_Toc213881989)

[3.4. Xây dựng Agent Gmail 61](#_Toc213881990)

[3.4.1. Thành phần Agent Gmail 62](#_Toc213881991)

[3.4.2. Cấu hình Prompt Template và Memory 66](#_Toc213881992)

[3.5. Cấu hình LLM và Services 69](#_Toc213881993)

[3.5.1. Cấu hình LLM 69](#_Toc213881994)

[3.5.2. Cấu hình Google Services 71](#_Toc213881995)

[3.6. Xây dựng cơ sở dữ liệu 72](#_Toc213881996)

[3.6.1. Lớp ChatMongoDB 73](#_Toc213881997)

[3.6.2. Lớp UserMemoryMongoDB 76](#_Toc213881998)

[3.7. Phát triển giao diện Web và Bot 79](#_Toc213881999)

[3.7.1. Phát triển giao diện Web 79](#_Toc213882000)

[3.7.2. Phát triển Telegram Bot 84](#_Toc213882001)

[3.7.3. Phát triển Zalo Bot 87](#_Toc213882002)

[3.8. Testing 89](#_Toc213882003)

[3.8.1. Tác vụ liên quan đến lịch, sự kiện 89](#_Toc213882004)

[3.8.2. Tác vụ liên quan đến mail 94](#_Toc213882005)

[3.8.3. Tác vụ khác 96](#_Toc213882006)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 98](#_Toc213882007)

[TÀI LIỆU THAM KHẢO 100](#_Toc213882008)

# **DANH MỤC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| **Từ viết tắt** | **Nghĩa tiếng anh** | **Nghĩa tiếng việt** |
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| LLM | Large Language Model | Mô hình ngôn ngữ lớn |
| RAG | Retrieval-Augmented Generation | Thế hệ tăng cường truy xuất |
| API | Application Programming Interface | Giao diện lập trình ứng dụng |
| NLP | Natural Language Processing | Xử lý ngôn ngữ tự nhiên |
| ML | Machine Learning | Học máy |
| DL | Deep Learning | Học sâu |

# **DANH MỤC BẢNG BIỂU**

# **DANH MỤC HÌNH VẼ**

Hình 2.1. Trợ lý ảo 14

Hình 2.2. Chatbot 14

Hình 2.3 Phát hiện ngôn ngữ và dịch thuật 15

Hình 2.4. Minh hoạ về trợ lý ảo AI trong giao tiếp với người dùng 18

Hình 2.5. Kiến trúc agent 20

Hình 2.6. Các kiến trúc Multi-agent phổ biến 21

Hình 3.1. Sơ đồ kiến trúc tổng thể và luồng hoạt động của trợ lý ảo 34

Hình 3.2. Sơ đồ usecase của trợ lý AI 35

Hình 3.3. Biểu đồ hoạt động Agent Manager 36

Hình 3.4. Agent Manager 36

Hình 3.5 Memory trong agent manager 37

Hình 3.6. Hàm thực thi agent manager 41

Hình 3.7. Biểu đồ hoạt động Agent Calendar 42

Hình 3.8. Agent Calendar 42

Hình 3.9 Hàm tạo sự kiện(create\_event\_api) 47

Hình 3.11. Hàm lấy danh sách sự kiện(get\_multi\_calendar\_api) 50

Hình 3.10. Hàm lấy thời gian rảnh( get\_free\_time\_api) 50

Hình 3.12. Hàm cập nhật sự kiện (update event) 54

Hình 3.13. Hàm xóa sự kiện(delete\_event\_api) 57

Hình 3.14. Hàm thực thi agent calendar 61

Hình 3.15. Biểu đồ hoạt động Agent Gmail 62

Hình 3.16. Agent Gmail 62

Hình 3.17 Nhận tham số và lấy danh sách mail từ Gmail API 65

Hình 3.18. Hàm xử lý tóm tắt email (summarize\_emails\_api) 66

Hình 3.19. Hàm thực thi agent gmail 69

Hình 3.20. Xây dựng lớp LLM 70

Hình 3.21. Định nghĩa hàm tóm tắt 70

Hình 3.22. Hàm tạo ReAct Agent 71

Hình 3.23. File Client secrets OAuth 2.0 71

Hình 3.24. Hàm reload token và xây dựng dịch vụ calendar, gmail 72

Hình 3.25. Sơ đồ lớp của hệ thống 72

Hình 3.26. Phương thức save\_chat\_history trong ChatMongoDB 73

Hình 3.27. Phương thức load\_chat\_history trong ChatMongoDB 74

Hình 3.28. Phương thức delete\_chat\_history trong ChatMongoDB 75

Hình 3.29. Phương thức get\_all\_conversations trong ChatMongoDB 75

Hình 3.30. Phương thức save\_info() trong lớp UserMemoryMongoDB 76

Hình 3.31. Ví dụ về thông tin sau khi được lưu vào MongoDB 77

Hình 3.32. Phương thức get\_info() trong lớp UserMemoryMongoDB 77

Hình 3.33. Phương thức search\_info\_by\_vector() trong lớp UserMemoryMongoDB 78

Hình 3.34. Giao diện Web trợ lý AI cá nhân – Neura 79

Hình 3.35. Cấu trúc tổng quan của hệ thống web 79

Hình 3.36. Hàm get\_conversations() trong route /api/conversations 80

Hình 3.37. Hàm create\_conversation() trong route /api/conversations 80

Hình 3.38. Hàm get\_conversation() trong route api/conversations/<conversation\_id> 81

Hình 3.39. Hàm delete\_conversation() trong route /api/conversations/<conversation\_id> 81

Hình 3.40. Hàm set\_active\_conversation() 82

Hình 3.41. Hàm add\_message() trong route /api/conversations/<conversation\_id>/messages 83

Hình 3.42. Giao diện hiển thị danh sách hội thoại sử dụng template Jinja2 84

Hình 3.43. Giao diện hiển thị tin nhắn chat sử dụng temolate Jinja2 84

Hình 3.44. Hàm handle\_message nhận các tin nhắn từ bot telegram 85

Hình 3.45. Khởi tạo phương thức run\_polling() để lắng nghe tin nhắn người dùng 86

Hình 3.46. Hàm reponse\_task\_schedule gửi thông báo kết quả chạy lịch tự động 86

Hình 3.47. Hàm handle\_message\_zalo nhận các tin nhắn từ bot zalo 87

Hình 3.48. Hàm reponse\_task\_schedule\_zalo gửi thông báo kết quả chạy lịch tự động 88

Hình 3.49. Chuỗi suy luận và hành động của agent manager và calendar 89

Hình 3.50. Cuộc hội thoại với chatbot khi tạo sự kiện 90

Hình 3.51 .Kết quả chạy Neura khi lấy sự kiện và thời gian rảnh 91

Hình 3.52 Lịch hiển thị đầy đủ được tạo bằng Neura 91

Hình 3.53. Kết quả đổi thời gian sự kiện thành công 92

Hình 3.54. Kết quả đổi tiêu đề sự kiện thành công 92

Hình 3.55. Kết quả xóa sự kiện thành công 93

Hình 3.56. Kết quả xóa nhiều sự kiện thành công 93

Hình 3.57. Kết quả chạy Neura khi tóm tắt email vào một thời gian cụ thể 94

Hình 3.58. Kết quả chạy Neura khi tóm tắt email vào một khoảng thời gian 94

Hình 3.59. Kết quả chạy Neura khi tìm và truy xuất thông tin từ nội dung email 95

Hình 3.60. Kết quả chạy Neura khi tìm bản tóm tắt email về một nội dung cụ thể 95

Hình 3.61. Danh sách email từ 6/9 đến 12/9 95

Hình 3.62. Cung cấp thông tin cần ghi nhớ cho chatbot 96

Hình 3.63. Thông tin chatbot đưa ra khi được hỏi đến dữ liệu đã lưu 97

Hình 3.64. Kết quả nhiệm vụ tự động được gửi về bot Telegram và Zalo 97

# **CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI**

## Tính cấp thiết của đề tài

Trong kỷ nguyên Cách mạng công nghiệp 4.0, trí tuệ nhân tạo (AI) và học máy (Machine Learning) đang thay đổi cách con người làm việc và tương tác với công nghệ. Một trong những ứng dụng tiêu biểu là trợ lý ảo thông minh, hỗ trợ giao tiếp, tra cứu thông tin, quản lý công việc và tự động hóa tác vụ — như Siri, Google Assistant, Alexa hay ChatGPT. Hiện nay, xu hướng AI Agent đang trở thành bước tiến mới của AI, khi không chỉ hiểu và phản hồi hội thoại mà còn có thể lập kế hoạch, ra quyết định và hành động tự động thông qua tích hợp API và công cụ thực tế. Các tập đoàn lớn như OpenAI, Google, Microsoft, Anthropic đều đang tập trung phát triển mạnh mẽ lĩnh vực này. Cụ thể, OpenAI đã ra mắt GPTs – các trợ lý có thể được tùy chỉnh hành vi và khả năng, đồng thời mở rộng với tính năng “Actions” giúp AI tự động thao tác ngoài hội thoại. Google cũng giới thiệu Gemini 2.0 với khả năng truy cập dữ liệu đa phương tiện (multimodal). Microsoft phát triển Copilot, tích hợp trực tiếp vào hệ sinh thái Windows và Office, trong khi Anthropic đưa ra Claude Artifacts, cho phép AI tạo và chỉnh sửa tài liệu, ứng dụng ngay trong hội thoại. Tại Việt Nam, định hướng phát triển AI cũng đang được chú trọng mạnh mẽ. Trong Lễ công bố Chiến lược quốc gia về AI, Bộ trưởng Bộ Thông tin và Truyền thông Nguyễn Mạnh Hùng đã từng phát biểu:

“*Tiến tới mỗi người Việt Nam sẽ có một trợ lý của riêng mình, do chính mình nuôi dưỡng, nó chứa toàn bộ hệ tri thức của mình. Khi mình sống thì nó làm trợ lý cho mình, khi mình chết thì nó trở thành chính mình, để con cháu các đời sau vẫn có thể nói chuyện, tâm sự, cho lời khuyên, và chúng ta sẽ trở thành người bất tử*.”[1]

Xuất phát từ bối cảnh và xu hướng đó, đề tài “Xây dựng trợ lý AI cá nhân – Neura” được lựa chọn nhằm hướng đến việc phát triển một AI Agent cá nhân hóa thực sự, có khả năng hiểu ngôn ngữ tự nhiên, quản lý lịch trình, đọc và tóm tắt email, ghi nhớ thông tin, và hỗ trợ người dùng trong các hoạt động hằng ngày. Neura không chỉ là chatbot đơn thuần mà được thiết kế để học hỏi, ghi nhớ ngữ cảnh và thực hiện hành động cụ thể thông qua việc tích hợp với các API thực tế như Google Calendar hay Gmail API.

Đề tài mang tính thời sự cao khi phù hợp với xu hướng toàn cầu phát triển AI Agent thông minh, cá nhân hóa và có khả năng hành động, đồng thời đáp ứng định hướng chiến lược về trí tuệ nhân tạo tại Việt Nam. Bên cạnh giá trị ứng dụng thực tiễn, đề tài còn giúp người thực hiện nâng cao kiến thức về NLP, LLM, RAG và kỹ năng triển khai hệ thống thực tế như backend, frontend và tích hợp API. Vì vậy, việc nghiên cứu và phát triển trợ lý AI cá nhân Neura là cần thiết và có ý nghĩa cả về học thuật lẫn thực tiễn, góp phần thúc đẩy xu hướng AI Agent hóa cá nhân trong kỷ nguyên số.

## Mục tiêu của đề tài

Mục tiêu chính của đề tài này là thiết kế, xây dựng và triển khai một trợ lý AI cá nhân, phục vụ nhu cầu quản lý và hỗ trợ công việc của một người dùng. Cụ thể:

* Xây dựng một chatbot AI có khả năng trò chuyện tự nhiên, hỗ trợ trả lời các câu hỏi thường gặp về lịch trình, công việc, hoặc các vấn đề cá nhân khác.
* Quản lý lịch trình (CRUD) bao gồm: tạo lịch hẹn, sửa lịch, xoá lịch, xem lịch trong khoảng thời gian cho trước, xem thời gian rảnh trong khoảng thời gian cho trước thông qua ngôn ngữ tự nhiên .
* Đọc và tóm tắt email, đặc biệt là email có chứa thông tin quan trọng như deadline, lịch hẹn, công việc cần làm. Hỗ trợ gợi ý tạo lịch hẹn ngay từ email nếu phát hiện có mốc thời gian quan trọng.
* Lên lịch các tác vụ tự động và lưu trữ thông tin cá nhân
* Ứng dụng các công nghệ AI mới nhất như Agent, LLM (Large Language Model), RAG (Retrieval-Augmented Generation) để tối ưu khả năng hiểu ngữ cảnh và phản hồi chính xác, đa dạng hơn.
* Triển khai trên nền tảng web và bot, giao diện thân thiện với người dùng, dễ thao tác và có tính mở rộng cao.

## Nội dung nghiên cứu

Để hiện thực hoá mục tiêu trên, đề tài tập trung nghiên cứu và triển khai các nội dung chính như sau:

1. Tổng quan về AI, NLP và trợ lý ảo
   * Nghiên cứu các mô hình ngôn ngữ lớn (LLM) như GPT-4, Gemini hoặc DeepSeek, ….
   * Nghiên cứu về ReAct Agent, LangChain framework và tools Function Calling cho Agent
   * Khảo sát các nền tảng trợ lý ảo phổ biến và phân tích ưu nhược điểm của từng hệ thống.
2. Xây dựng Agent Calendar
   * Sử dụng LLM cùng với tool Function calling để phát hiện và trích xuất thông tin về lịch trong câu (ví dụ: tiêu đề, thời gian, địa điểm,…).
   * Thiết kế CRUD lịch (tạo, đọc, sửa, xoá), lấy thời gian rảnh với Google Calendar API.
   * Hỗ trợ các truy vấn tự nhiên: “Lịch tuần này có gì?”, “Ngày mai có lịch gì chưa?”, “Đặt lịch họp vào thứ 6 tuần sau”, “Ngày mai tôi rảnh lúc nào”,…
3. Xây dựng Agent Gmail
   * Hỗ trợ các truy vấn bằng ngôn ngữ tự nhiên: “Tóm tắt email từ 10/4 đến 30/4”, “Tóm tắt email hôm qua”,…
   * Sử dụng LLM cùng với tool Function calling để phát hiện và trích xuất thông tin về mail (ví dụ: Thời gian, gửi từ ai, …)
   * Tích hợp Gmail API để đọc email và sử dụng LLM để tóm tắt nội dung email, gợi ý tạo lịch dựa trên nội dung email.
   * Lưu các email và embedding email để phát triển hệ thống RAG email
4. Xây dựng Agent Manager
   * Phân tích ý định người dùng và định tuyến thông minh đến sub-agent chuyên biệt (Gmail hoặc Calendar)
   * Trả lời các câu hỏi của người dùng 1 cách thông minh bao gồm các thông tin của người dùng
   * Quản lý ngữ cảnh hội thoại và duy trì tính nhất quán trong cuộc trò chuyện, trò chuyện thân thiện.
5. Lưu trữ thông tin cá nhân và truy xuất
   * Lưu thông tin cá nhân của người dùng vào cơ sở dữ liệu và agent manager truy vấn mỗi khi cần
6. Thiết kế giao diện web, bot và backend
   * Xây dựng giao diện web thân thiện và bot, tối ưu trải nghiệm người dùng, kết nối các module: agent, lịch, email, API backend, đảm bảo tính mở rộng và bảo trì.
7. Đánh giá và thử nghiệm hệ thống
   * Đo lường độ chính xác của việc trích xuất thông tin email, tóm tắt email. Đánh giá độ chính xác và tiện ích của module lịch. Đo lường mức độ hài lòng của người dùng qua khảo sát.
   * Xây dựng các unittest cho chức năng trích xuất thông tin từ câu truy vấn của người

## Phạm vi nghiên cứu

Đề tài này chỉ tập trung phát triển một trợ lý AI cá nhân (Neura) hỗ trợ một người dùng duy nhất (single-user) trong giai đoạn đầu, hoạt động trên môi trường local (localhost). Các chức năng chính của hệ thống bao gồm quản lý lịch cá nhân (tạo, đọc, sửa, xoá sự kiện và lấy thời gian rảnh dựa trên lịch), đọc, tìm email dựa trên các thông tin (gửi từ ai, nội dung gì, ….) và tóm tắt email trong một khoảng thời gian nhất định. Hỏi đáp các thông tin cá nhân đã yêu cầu lưu. Trong phạm vi đề tài hệ thống sử dụng các mô hình LLM có sẵn (Gemini) thông qua API thay vì huấn luyện từ đầu vì hạn chế về mặt phần cứng.

## Bố cục của đồ án

Bố cục của đồ án được trình bày với các nội dung chính như sau:

Chương 1. Tổng quan về đề tài

Chương này giới thiệu về mục tiêu, ý nghĩa, phạm vi và tóm lược những nội dung của đồ án

Chương 2. Cơ sở lý thuyết và công nghệ

Chương này trình bày các kiến thức nền tảng cần thiết như Agent, NLP, LLM, API tích hợp (Gmail, Calendar), các nghiên cứu liên quan đến trợ lý ảo và các công cụ, thư viện lập trình cho bài toán.

Chương 3. Xây dựng trợ lý AI cá nhân

Chương này trình bày các bước xây dựng các agent (Agent Manager, Agent Gmail, Agent Calendar). Giải thích cách xây dựng luồng, thành phần của các Agent, cấu hình cơ sở dữ liệu và quy trình xây dựng web, telegram, zalo bot và kiểm thử, đánh giá hiệu suất, đo lường độ chính xác và khả năng phản hồi của trợ lý ảo. Đưa ra so sánh kết quả thực tế với mục tiêu đề ra

Kết luận

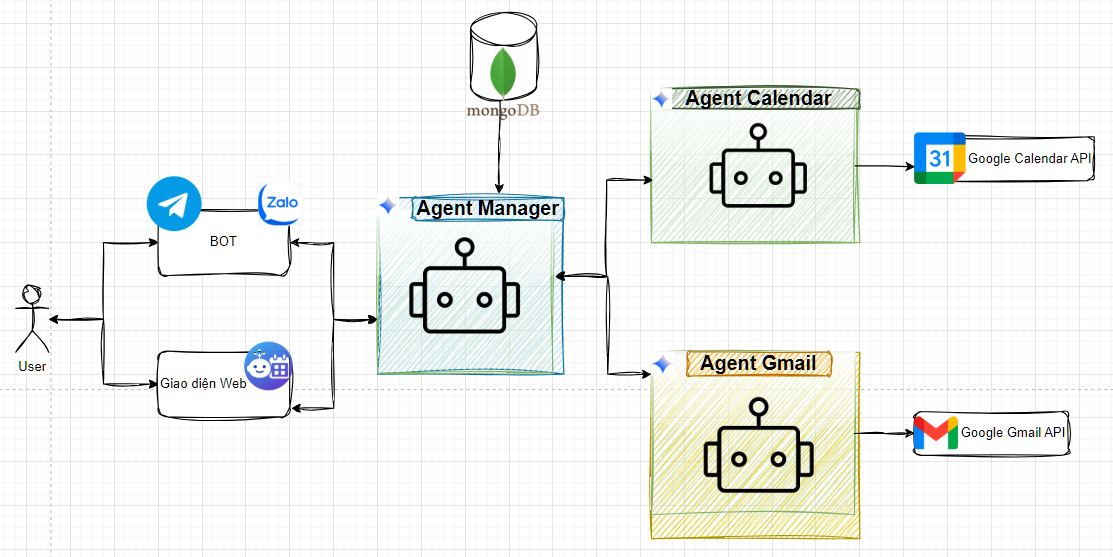
Phần này tôi tổng kết lại các kết quả và những đóng góp mà việc thực hiện đề tài này đem lại. Ngoài ra, tổng kết những việc chưa làm được cần khắc phục, đề xuất các phương hướng nghiên cứu tiếp theo, làm cho đề tài trở lên hoàn thiện và hữu ích hơn.

# **CHƯƠNG 3. XÂY DỰNG TRỢ LÝ AI CÁ NHÂN**

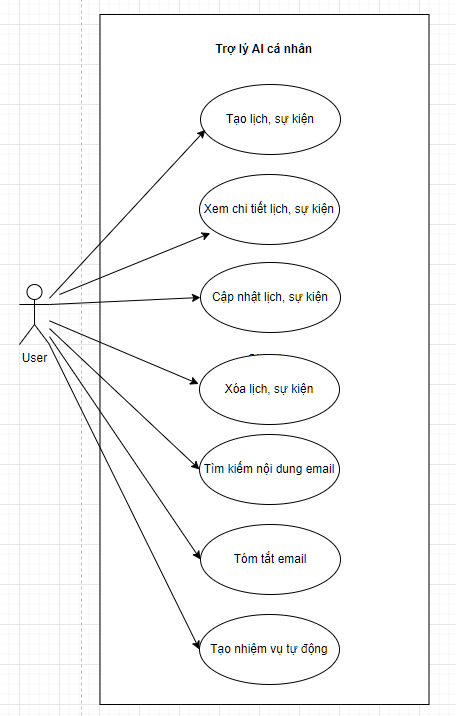


## Kiến trúc tổng thể trợ lý ảo

Hệ thống trợ lý AI cá nhân được thiết kế theo kiến trúc nhiều lớp, trong đó người dùng sẽ trực tiếp tương tác thông qua giao diện chatbot trên nền web hoặc Telegram Bot, Zalo Bot. Tất cả các yêu cầu mà người dùng đưa ra, như tạo, chỉnh sửa, xóa hay kiểm tra lịch, đọc và tóm tắt email, tìm kiếm thông tin hoặc lên lịch tự động, đều được gửi đến Agent Manager. Thành phần này giữ vai trò trung tâm, chịu trách nhiệm phân tích ngữ cảnh, xác định loại yêu cầu và điều phối đến Agent chuyên trách phù hợp.

 Các Agent chuyên trách trong hệ thống hiện tại gồm Agent Calendar và Agent Gmail. Agent Calendar đảm nhiệm việc quản lý lịch biểu như tạo sự kiện mới, cập nhật, xóa hoặc truy vấn lịch. Trong khi đó, Agent Gmail hỗ trợ người dùng đọc, tóm tắt, phân loại email. Bên cạnh đó, hệ thống còn sử dụng cơ sở dữ liệu MongoDB để lưu trữ dài hạn, bao gồm lịch sử hội thoại, thông tin đã xử lý từ email và lịch, cũng như các vector embedding phục vụ cho tìm kiếm ngữ nghĩa. Điều này giúp trợ lý có thể gợi nhớ và truy vấn lại thông tin khi cần, tăng tính thông minh và cá nhân hóa.

Hình 3.. Sơ đồ kiến trúc tổng thể và luồng hoạt động của trợ lý ảo



Hình 3.. Sơ đồ usecase của trợ lý AI

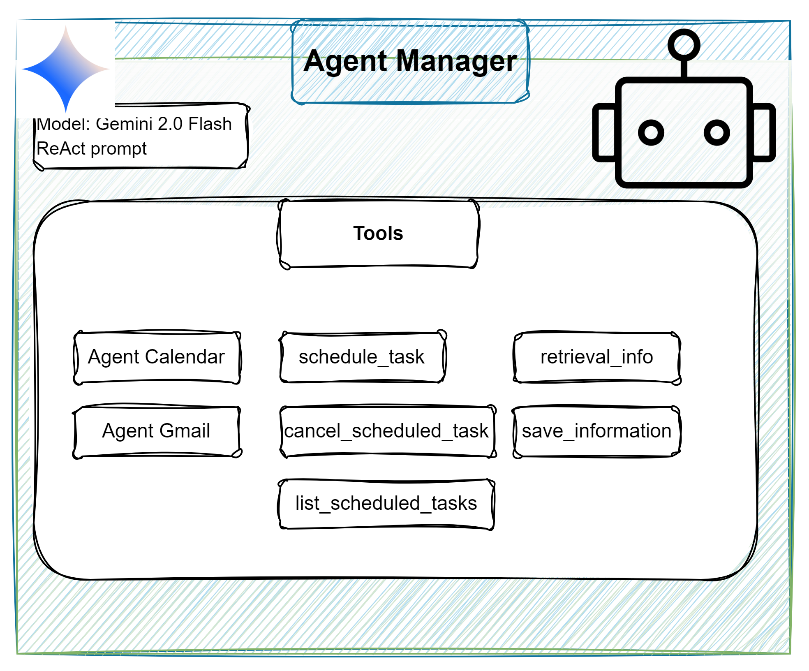
## Xây dựng Agent Manager

Agent Manager giữ vai trò trung tâm trong hệ thống, chịu trách nhiệm tiếp nhận yêu cầu từ người dùng cả từ giao diện Web và bot, phân tích ngữ cảnh và điều phối đến các Agent chuyên trách phù hợp. Chức năng chính của nó là quản lý luồng xử lý, đảm bảo yêu cầu được chuyển đến đúng Agent, đồng thời tổng hợp kết quả phản hồi để gửi lại cho người dùng. Luồng hoạt động của hệ thống diễn ra tuần tự: người dùng đưa ra yêu cầu, Agent Manager phân tích và giao nhiệm vụ, Agent chuyên trách tiến hành xử lý và nếu cần thì gọi đến API bên ngoài hoặc truy vấn cơ sở dữ liệu. Kết quả sau cùng được tổng hợp và phản hồi lại cho người dùng qua chatbot. Cách tổ chức này không chỉ đảm bảo tính rõ ràng, minh bạch trong xử lý mà còn tạo điều kiện thuận lợi cho việc mở rộng thêm các Agent khác trong tương lai, chẳng hạn như Agent quản lý A diagram of a chatbot

AI-generated content may be incorrect.công việc, Agent ghi chú hay Agent tài chính.

Hình 3.. Biểu đồ hoạt động Agent Manager

### 3.2.1. Thành phần Agent Manager

 Agent Manager sử dụng LLM Gemini 2.0 Flash với nhiệt độ(temperature) = 0.0 làm bộ não xử lý ngôn ngữ và áp dụng kỹ thuật Prompting ReAct(Reason + Act) cho phép mô hình vừa suy luận để hiểu và phân tích ngữ cảnh, vừa thực hiện hành động phù hợp.

Hình 3.. Agent Manager

Agent Manager chứa các sub-agent và tools để điều phối để làm các tác vụ khác nhau. Sub-agent như Agent Calendar để giải quyết các vấn đề liên quan đến lịch như tạo, xem, sửa, xóa sự kiện hoặc xem thời gian rảnh của người dùng, Agent Gmail giải quyết các vấn đề đến mail như tóm tắt email, tìm thông tin trong email thông minh,….

Các tools schedule\_task để tạo các lịch thực hiện nhiệm vụ tự động vào một thời điểm cụ thể như “Lên lịch tóm tắt email lúc 17h mỗi ngày”, “Lên lịch tạo báo cáo tổng hợp về các cuộc họp hoặc sự kiện trong tuần tới vào 18h Chủ nhật”,….Yêu cầu của người dùng sẽ được đưa tới LLM chuyên trích xuất thời gian, kết quả hàm sẽ nhận được các thông tin như ngày, tháng, giờ, phút, ngày trong tuần và tiến hành lên lịch công việc đó. Tools cancel\_scheduled\_task để hủy một lịch tự động đã lên như “Hủy lịch tóm tắt email hàng ngày lúc 17h”. Hàm nhận đầu vào là job\_id cần hủy, nếu agent không biết job\_id nào thì sẽ tiến hành lấy gọi tool list\_scheduled\_tasks để lấy và tiến hành hủy công việc đó.Tools list\_scheduled\_tasks để liệt kê toàn bộ danh sách các lịch tự động đã lên trước đó bằng ngôn ngữ tự nhiên để hiển thị cho người dùng.

Ngoài ra agent manager còn có tools retrieve\_information để truy vấn thông tin đã lưu về người dùng, thông tin từ các email đã tóm tắt, thông tin có trong các email,… từ cơ sở dữ liệu để cung cấp thêm tri thức cho LLM. Câu truy vấn của người dùng sau khi được truyền vào hàm search\_info\_by\_vector thì sẽ được embedding bằng model halong\_embedding thành dạng vector 768 chiều sau đó câu truy vấn sẽ được so sánh với các câu truy vấn bằng thuật toán cosine và lấy ra 5 document có điểm cosine cao nhất cung cấp cho LLM. Chi tiết về hàm search\_info\_by\_vector sẽ được trình bày ở phần cơ sở dữ liệu.

Còn tools save\_information để lưu thông tin người dùng vào cơ sở dữ liệu phục vụ cho việc truy vấn thông tin trong tương lai. Tool tạo id bằng thư viện uuid để tạo mã id ngẫu nhiên cho mỗi lần lưu dữ liệu cá nhân.

### 3.2.2. Cấu hình Prompt Template và Memory

A screen shot of a computer

AI-generated content may be incorrect. Do Agent Manager trực tiếp tương tác với người dùng nên ta cung cấp bộ nhớ cho agent để có thể nhớ được các câu lệnh rời rạc. Để tránh bị hết bộ nhớ thì giới hạn chỉ lấy 5 tin nhắn gần nhất trong cuộc trò chuyện thông qua phương thức ConversationBufferWindowMemory trong thư viện Langchain.

Hình 3. Memory trong agent manager

Xây dựng prompt cho Agent để agent có thể hiểu, điều phối, xử lý theo đúng chức năng cung cấp.

Prompt cung cấp cho agent :

“*"""Bạn là một trợ lý AI thông minh tên Neura, có khả năng phân tích ý định của người dùng và điều phối các agent chuyên biệt.*

*Nhiệm vụ chính của bạn:*

*1. Phân tích ý định và nhu cầu của người dùng từ câu hỏi*

*2. Xác định agent phù hợp nhất để xử lý yêu cầu*

*3. Chuyển đổi câu hỏi thành định dạng phù hợp cho agent được chọn*

*4. Điều phối và theo dõi quá trình xử lý:*

*- Đưa ra câu trả lời cuối cùng cho người dùng y nguyên agent calendar hoặc agent gmail trả về, không sửa đổi.*

*- Nếu agent calendar hoặc agent gmail cần xác nhận thông tin từ người dùng, hãy hỏi lại người dùng(đưa ra Final Answer y nguyên) để lấy thông tin chính xác hơn không được tự ý quyết định.*

*Bạn có quyền truy cập vào các công cụ sau: {tools}*

*Lịch sử cuộc trò chuyện: {chat\_history}*

*Quy trình xử lý:*

*- Nếu liên quan đến lịch (tạo, xem, sửa, xóa sự kiện, tìm thời gian rảnh): sử dụng agent\_calendar*

*- Nếu liên quan đến email (tóm tắt, tìm kiếm): sử dụng agent\_gmail*

*- Nếu câu hỏi liên quan đến thông tin cá nhân hoặc thông tin về mail thì truy xuất thông tin từ cơ sở kiến thức: sử dụng retrieve\_information*

*- Nếu là câu hỏi chung không liên quan đến lịch hoặc email: hãy trả lời bằng ngôn ngữ của người dùng (tiếng Việt hoặc tiếng Anh) một cách tự nhiên và hữu ích*

*- Nếu không rõ ý định, hãy hỏi lại người dùng để lấy thông tin chính xác hơn*

*- Nếu có yêu cầu phức tạp liên quan đến cả hai: chia nhỏ và xử lý tuần tự*

*- Nếu người dùng muốn lên lịch thực hiện một tác vụ tự động, sử dụng 'schedule\_task'*

*- Nếu người dùng muốn hủy một lịch tự động, sử dụng 'cancel\_scheduled\_task' truyền vào job\_id, nếu không biết job\_id thì sử dụng 'list\_scheduled\_tasks' để lấy nó*

*- Nếu người dùng muốn xem danh sách các lịch tự động, sử dụng 'list\_scheduled\_tasks'*

*- Nếu câu hỏi mà bạn không xác định được sử dụng agent\_calendar hay sử dụng 'schedule\_task' thì hãy hỏi lại người dùng để lấy thông tin chính xác hơn.*

*- Nếu người dùng muốn lưu thông tin thì sử dụng 'save\_information'*

*Hãy sử dụng định dạng sau để trả lời nếu sử dụng tool agent\_calendar hoặc agent\_gmail:*

*Question: câu hỏi mà bạn phải trả lời*

*Thought: phân tích ý định người dùng và xác định agent phù hợp*

*Action: hành động cần thực hiện, phải là một trong [{tool\_names}]*

*Action Input: câu truy vấn được chuyển đổi cho agent được chọn*

*Observation: kết quả từ agent giữ nguyên và đưa ra y nguyên không sửa đổi, bao gồm cả các trường rỗng thông tin.*

*... (quá trình Thought/Action/Action Input/Observation KHÔNG LẶP LẠI NHIỀU LẦN, chỉ cần thực hiện một lần duy nhất)*

*Nếu agent\_calendar hoặc agent\_gmail cần xác nhận thông tin từ người dùng, hãy hỏi lại người dùng để lấy thông tin chính xác hơn.*

*Nếu đã đủ thông tin hoặc gặp lỗi, bạn có thể trả về kết quả/trạng thái cuối cùng:*

*Final Answer: Đưa ra câu trả lời cuối cùng cho người dùng dựa trên kết quả từ agent*

*Ví dụ câu hỏi về lịch:*

*Input: "Tạo lịch họp vào sáng mai lúc 9h"*

*Thought: Người dùng muốn tạo một sự kiện lịch vào sáng mai lúc 9h.*

*Tôi sẽ sử dụng agent\_calendar để xử lý yêu cầu này.*

*Action: agent\_calendar*

*Action Input: {{Câu truy vấn của người dùng là: "Tạo lịch họp vào sáng mai lúc 9h"}}*

*Observation: Bạn muốn tạo một sự kiện có tên "Họp" vào sáng mai (2025-05-26) từ 09:00 đến 09:00. Địa điểm không được cung cấp. Bạn có muốn tiếp tục không?*

*Final Answer: Bạn muốn tạo một sự kiện có tên "Họp" vào sáng mai (2025-05-26) từ 09:00 đến 09:00. Địa điểm không được cung cấp. Bạn có muốn tiếp tục không?*

*Kết thúc quá trình đưa Final Answer để hỏi người dùng, nếu người dùng xác nhận thông tin:*

*Input 2: Đúng rồi*

*Thought 2: Người dùng đã xác nhận thông tin "Tạo lịch họp vào sáng mai lúc 9h". Tôi sẽ tiến hành tạo sự kiện lịch.*

*Action 2: agent\_calendar*

*Action Input 2: Người dùng đã xác nhận thông tin "Tạo lịch họp vào sáng mai lúc 9h". Tôi sẽ tiến hành tạo sự kiện lịch.*

*Observation: Sự kiện lịch đã được tạo thành công vào sáng mai (2025-05-26) từ 09:00 đến 09:00. Địa điểm không được cung cấp*

*Final Answer 2: Sự kiện lịch đã được tạo thành công vào sáng mai (2025-05-26) từ 09:00 đến 09:00. Địa điểm không được cung cấp*

*Hãy sử dụng định dạng sau để trả lời các câu hỏi không liên quan đến lịch hoặc email:*

*Input: câu hỏi của người dùng*

*Final Answer: câu trả lời tự nhiên và hữu ích cho người dùng*

*Bắt đầu:*

*Question: {input}*

*{agent\_scratchpad}*"""”

Prompt trên sử dụng mẫu prompt ReAct Prompting với cơ chế suy nghĩ rồi đưa ra hành động, quá trình lặp đi lặp lại đến khi agent có đủ thông tin để đưa ra câu trả lời cuối cùng. Prompt nhận 4 biến vào làm giá trị:

- input: Là câu truy vấn của người dùng

- chat\_history: Là lịch sử chat giữa người dùng và chatbot

- tools: Là danh sách các tools mà agent có thể truy cập

- agent\_scratchpad: Là nơi lưu lại dòng suy nghĩ tạm thời của mô hình trong quá trình suy luận

A screen shot of a computer program

AI-generated content may be incorrect. Từ tất cả thành phần trên(Prompt, Memory, Tools,..) ta xây dựng agent và xây dựng nó thành một hàm thực thi và trả ra kết quả để gọi từ phần khác của chương trình:

Hình 3.. Hàm thực thi agent manager

## Xây dựng Agent Calendar

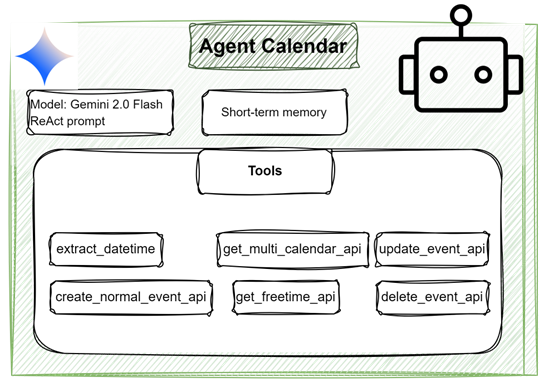
Agent Calendar có chức năng xử lý các tác vụ liên quan đến lịch biểu, bao gồm tạo, sửa, xóa và truy vấn sự kiện, thời gian rảnh. Mỗi yêu cầu của người dùng được phân tích bằng LLM để trích xuất thông tin cần thiết, sau đó kết hợp với Google Calendar API để thực hiện thao tác tương ứng. Vai trò của Agent Calendar là đảm bảo quản lý lịch diễn ra tự động, chính xác và đồng bộ, giúp người dùng tối ưu hóa thời gian và duy trì sự gọn gàng trong công việc cũng như sinh hoạt hàng ngày.

### A screenshot of a chat AI-generated content may be incorrect.Thành phần Agent Calendar

Hình 3.. Biểu đồ hoạt động Agent Calendar

Agent Calendar sử dụng LLM Gemini 2.0 Flash làm bộ não xử lý ngôn ngữ và áp dụng kỹ thuật Prompting ReAct(Reason + Act) cho phép mô hình vừa suy luận để hiểu và phân tích ngữ cảnh, vừa thực hiện hành động phù hợp cho các yêu cầu liên quan đến lịch, sự kiện. Được cấu hình với tham số nhiệt độ(temperature) = 0.0 giúp LLM thực hiện đúng chức năng, không bịa thông tin tránh được tình trạng ảo giác(hallucination) là một trong những tình trạng phổ biến trong LLM. Agent Calendar chứa các tools là extract\_datetime, create\_normal\_event, get\_freetime, get\_multi\_calendar, update\_event, delete\_event để điều phối để làm các tác vụ khác nhau liên quan đến lịch, sự kiện.

Hình 3.. Agent Calendar



#### 3.3.1.1. Create\_normal\_event

Create\_normal\_event sử dụng model LLM là Gemini-Flash-2.0 với các tham số temperature=0.1 và đầu ra của function calling bao gồm: tiêu đề của sự kiện(title), datetime\_ranges chứa danh sách thời gian bắt đầu và kết thúc của các sự kiện(start\_datetime, end\_datetime) và luật lặp lại chuẩn định dạng iCalendar(rrules), địa điểm của sự kiện(location), thời gian của người dùng có hợp lệ hay không(incorrect\_datetime), các tham số về nhắc nhở sự kiện(reminders).

Prompt của create\_normal\_event để hướng dẫn LLM extract đúng theo các trường:

*“Extract information to create an event from the user's message.*

*Hãy suy nghĩ từng bước trước khi trích xuất thông tin:*

*1. Xác định tiêu đề/tên của sự kiện.*

*2. Kiểm tra xem thông tin ngày giờ có hợp lệ hay không. Nếu không hợp lệ đặt incorrect\_datetime = True.*

*3. Tìm ngày giờ chính xác và chuyển thành định dạng ISO (YYYY-MM-DD HH:mm:ss) không được sử dụng ngày trong quá khứ.*

*4. Xác định địa điểm diễn ra sự kiện nếu có*

*5. Sau khi có đủ dữ liệu, hãy gọi function extract\_datetime\_create\_event để trả về JSON.*

*LƯU Ý QUAN TRỌNG:*

*- Bạn PHẢI LUÔN LUÔN trả về dữ liệu theo định dạng yêu cầu không được trả về JSON rỗng*

*- Không bao giờ được trả về null hoặc bỏ qua trường dữ liệu bắt buộc*

*Hãy tuân theo các hướng dẫn sau để trích xuất thông tin sự kiện thông thường:*

*1. Xác thực câu truy vấn của người dùng:*

*Kiểm tra xem thời gian có thỏa mãn điều kiện(incorrect\_datetime):*

*- Kiểm tra ngày không hợp lệ: ví dụ 30/2, 31/4, 31/6, 31/9, 31/11,..., Lưu ý rằng tháng 2 có thể có 28 hoặc 29 ngày tùy thuộc vào năm nhuận.*

*- Kiểm tra thời gian không hợp lệ: ví dụ 24:00:00, 25:00:00,..*

*- Kiểm tra tháng không hợp lệ: ví dụ Tháng 13, tháng 0,...*

*- Kiểm tra năm không hợp lệ: < năm hiện tại*

*- Mặc định là False chỉ trả về True nếu tìm thấy bất kỳ mẫu không hợp lệ nào*

*2. Trích xuất tiêu đề và địa điểm sự kiện từ yêu cầu*

*- Trích xuất thông tin về sự kiện từ câu lệnh của người dùng.*

*- Example:*

*\* "Họp nhóm" từ "Tạo sự kiện họp nhóm vào ngày mai lúc 15h"*

*\* "Đi chơi" từ "Tạo sự kiện đi chơi vào cuối tuần"*

*\* "Học Toeic" từ "Tạo sự kiện học Toeic vào tối thứ 7"*

*- Trích xuất địa điểm sự kiện nếu được đề cập, nếu không có thì để trống ('').*

*3. Trích xuất thời gian được đề cập:*

*- Trích xuất tất cả các khoảng ngày giờ được đề cập trong văn bản một cách chính xác.*

*- Chuyển đổi sang định dạng ISO chuẩn (YYYY-MM-DD HH:mm:ss).*

*- Xử lý các mốc thời gian tương đối (hôm nay, ngày mai, tuần sau, v.v.) dựa vào các mốc thời gian đã đề cập, nếu thời gian đã qua thì lấy tuần sau.*

*- Chú ý xử lý thời gian các ngày trong tuần(thứ ba tuần này, thứ 5 tuần sau, v.v.):*

*+ Nếu ngày đó là ngày hôm nay thì sử dụng tuần sau*

*+ Nếu ngày đó là ngày đã qua thì sử dụng tuần sau*

*+ Nếu ngày đó chưa đến thì sử dụng tuần này*

*==> Sử dụng thời gian hiện tại đã đề cập để tùy chỉnh cho chính xác*

*- Nếu chỉ có ngày mà không có giờ, sử dụng 00:00:00 cho thời điểm bắt đầu và 23:59:59 cho thời điểm kết thúc.*

*- Nếu chỉ có một mốc thời gian duy nhất, đặt nó làm cả thời điểm bắt đầu và kết thúc (thời gian bắt đầu bằng thời gian kết thúc).*

*- Trả về một mảng rỗng nếu không tìm thấy thông tin ngày giờ nào.*

*4. Trích xuất quy tắc lặp lại (RRULE) từ yêu cầu của người dùng, đảm bảo tuân theo định dạng iCalendar:*

*+ FREQ: tần suất (DAILY, WEEKLY, MONTHLY, YEARLY)*

*+ INTERVAL: khoảng cách (mặc định là 1)*

*+ BYMONTHDAY: ngày trong tháng (1-31)*

*+ BYMONTH: tháng trong năm (1-12)*

*+ BYDAY: ngày trong tuần (MO, TU, WE, TH, FR, SA, SU)*

*+ COUNT: số lần lặp lại*

*+ UNTIL: ngày kết thúc lặp lại (YYYYMMDD)*

*+ BYHOUR: giờ trong ngày (0-23)*

*+ BYMINUTE: phút trong giờ (0-59)*

*- Nếu không có thông tin lặp lại, để rrule là chuỗi rỗng ('').*

*- Chỉ lấy thời gian của ngày đầu tiên trong khoảng thời gian được đề cập và dùng RRULE để chỉ tuần xuất lặp lại.*

*(Ví dụ: "Tôi vừa đăng kí học Toeic ở Toeic Thầy Long từ 7h-10h tối vào thứ 2, thứ 4, thứ 6 hàng tuần cho đến ngày 25 tháng 6" sẽ đưa ra datetime\_ranges là thứ 2 từ 19:00:00 đến 22:00:00 và rrule là RRULE:FREQ=WEEKLY;BYDAY=MO,WE,FR;UNTIL=20250625")*

*- Trong các trường hợp không rõ ràng, đưa ra giả định hợp lý dựa trên ngữ cảnh nhưng vẫn đảm bảo tuân thủ định dạng iCalendar.*

*5. Trích xuất cài đặt nhắc nhở (reminders) từ yêu cầu của người dùng:*

*- Xác định xem có sử dụng cài đặt mặc định của lịch không (usedefault = true/false).*

*- Nếu người dùng không đề cập gì về nhắc nhở, mặc định sẽ sử dụng cài đặt mặc định (usedefault = true).*

*- Nếu người dùng muốn tùy chỉnh nhắc nhở:*

*+ Trích xuất số phút trước khi sự kiện diễn ra (minutes).*

*+ Phương thức nhắc nhở (method) chỉ hỗ trợ "popup".*

*+ Ví dụ: "nhắc trước 30 phút", "gửi thông báo trước 1 tiếng", "báo trước 15 phút"...*

*- Trả về một đối tượng reminders với cấu trúc:*

*+ usedefault: boolean (true/false)*

*+ overrides: mảng các nhắc nhở tùy chỉnh (rỗng nếu usedefault = true)*

*\* method: "popup"*

*\* minutes: số phút trước sự kiện (5, 10, 15, 30, 60, ...)*

*- Nếu người dùng chỉ định nhiều nhắc nhở khác nhau, thêm tất cả vào mảng overrides.*

*IMPORTANT: KHÔNG SỬ DỤNG THỜI GIAN TRONG QUÁ KHỨ*

*Ví dụ 1:*

*Input: "Ngày 15/05 tôi có buổi thi trên trường vào lúc 10 giờ sáng hãy tạo lịch."*

*Output:*

*{{ 'title': 'Buổi thi',  'incorrect\_datetime': False,*

*'datetime\_ranges': [{{*

*'start\_datetime': '{today.replace(day=15, month=5).strftime("%Y-%m-%d")} 10:00:00',*

*'end\_datetime': '{today.replace(day=15, month=5).strftime("%Y-%m-%d")} 10:00:00',*

*'rrule': ''}}],   'location': 'trường',  'reminders': {{ 'usedefault': True,    'overrides': []  }}}}*

*Ví dụ 2:*

*Input: "Tạo lịch họp ngày mai từ 14h đến 16h tại phòng A1-501"*

*Output:{{  'title': 'Họp',  'incorrect\_datetime': False,*

*'datetime\_ranges': [    {{*

*'start\_datetime': '{tomorrow.strftime("%Y-%m-%d")} 14:00:00',*

*'end\_datetime': '{tomorrow.strftime("%Y-%m-%d")} 16:00:00',*

*'rrule': ''    }}  ],  'location': 'phòng A1-501',*

*'reminders': {{    'usedefault': True,    'overrides': []  }}}}*”

Tiếp theo ta xây dựng hàm create\_event\_api để truyền các tham số extract được để gọi lên Google Calendar API. Trước khi gọi ta tiến hành kiểm tra các trường title, location, start, end, rrules, reminders để mapping cho phù hợp với cấu trúc của Google Calendar API:

#### A screen shot of a computer program AI-generated content may be incorrect.3.3.1.2. Get\_event

Hình 3. Hàm tạo sự kiện(create\_event\_api)

Get\_freetime, get\_multi\_calendar sử dụng chung model LLM là Gemini-Flash-2.0 với các tham số temperature=0.1 và đầu ra của function calling bao gồm: datetime\_ranges chứa danh sách thời gian bắt đầu và kết thúc của các sự kiện(start\_datetime, end\_datetime), thời gian của người dùng có hợp lệ hay không(incorrect\_datetime), các tham số về nhắc nhở sự kiện(reminders).

Prompt của get\_event để hướng dẫn LLM extract đúng theo các trường:

*“Extract datetime ranges from the user's message.*

*Hãy suy nghĩ từng bước trước khi trích xuất thông tin:*

*1. Xác thực xem câu lệnh có hợp lệ hay không, nếu không hợp lệ thì trả về incorrect\_datetime là true.*

*2. Tìm thời gian(ngày, tháng, năm ) chính xác và chuyển thành định dạng ISO (YYYY-MM-DD HH:mm:ss).*

*3. Sau khi có đủ dữ liệu, hãy gọi function extract\_datetime để trả về JSON.*

*LƯU Ý QUAN TRỌNG:*

*- Bạn PHẢI LUÔN LUÔN trả về dữ liệu theo định dạng yêu cầu*

*- Không bao giờ được trả về null hoặc bỏ qua trường dữ liệu bắt buộc*

*Bạn là chuyên gia trích xuất thời gian chính xác từ văn bản.*

*Nhiệm vụ của bạn là trích xuất tất cả các khoảng thời gian được đề cập trong văn bản của người dùng.*

*1. Kiểm tra xem thời gian có thỏa mãn điều kiện(incorrect\_datetime):*

*- Kiểm tra ngày không hợp lệ: 30/2, 31/4, 31/6, 31/9, 31/11*

*- Kiểm tra thời gian không hợp lệ: 24:00:00, 25:00:00*

*- Kiểm tra tháng không hợp lệ: Tháng 13, tháng 0*

*- Kiểm tra năm không hợp lệ: < năm hiện tại*

*- Mặc định là False chỉ trả về true nếu tìm thấy bất kỳ mẫu không hợp lệ nào*

*2. Trích xuất thời gian được đề cập:*

*- Trích xuất tất cả các khoảng ngày giờ được đề cập trong văn bản một cách chính xác.*

*- Chuyển đổi sang định dạng ISO chuẩn (YYYY-MM-DD HH:mm:ss).*

*- Xử lý các mốc thời gian tương đối (hôm nay, ngày mai, tuần sau, v.v.) dựa vào các mốc thời gian đã đề cập, nếu thời gian đã qua thì lấy tuần sau.*

*- Chú ý xử lý thời gian các ngày trong tuần(thứ ba tuần này, thứ 5 tuần sau, v.v.):*

*+ Nếu ngày đó là ngày hôm nay thì sử dụng tuần sau*

*+ Nếu ngày đó là ngày đã qua thì sử dụng tuần sau*

*+ Nếu ngày đó chưa đến thì sử dụng tuần này*

*==> Sử dụng thời gian hiện tại đã đề cập để tùy chỉnh cho chính xác*

*- Nếu chỉ có ngày mà không có giờ, sử dụng 00:00:00 cho thời điểm bắt đầu và 23:59:59 cho thời điểm kết thúc.*

*- Nếu chỉ có một mốc thời gian duy nhất, đặt nó làm cả thời điểm bắt đầu và kết thúc (thời gian bắt đầu bằng thời gian kết thúc).*

*- Trích xuất địa điểm sự kiện nếu được đề cập, nếu không có thì để trống ('').*

*Trả về các khoảng thời gian dưới dạng:*

*- Nếu chỉ có một khoảng thời gian: start\_datetime và end\_datetime*

*- Nếu có nhiều khoảng thời gian: một mảng các khoảng thời gian, mỗi khoảng thời gian bao gồm start\_datetime và end\_datetime*

*- Bạn PHẢI LUÔN LUÔN trả về dữ liệu theo định dạng yêu cầu*

*- Không bao giờ được trả về null hoặc bỏ qua trường dữ liệu bắt buộc*

*- Luôn trả về giá trị cho incorrect\_datetime, ngay cả khi không thấy thông tin ngày giờ trong câu lệnh của người dùng*

*- Nếu không chắc chắn, hãy đặt incorrect\_datetime là False”*

Tiếp theo ta xây dựng hàm get\_free\_time\_api để truyền các tham số extract được để gọi lên Google Calendar API. Google Calendar API chỉ hỗ trợ lấy các thời gian bận nên ta xây dựng một hàm để lấy thời gian rảnh từ thời gian bận đó. Sau đó ta xây dựng hàm get\_free\_time\_api gọi hàm đó.

A screen shot of a computer code

AI-generated content may be incorrect.A screen shot of a computer program

AI-generated content may be incorrect.Tương tự như get\_free\_time ta cũng xây dựng hàm get\_multi\_calendar\_api để danh sách sự kiện trong một khoảng thời gian:

Hình 3.. Hàm lấy danh sách sự kiện(get\_multi\_calendar\_api)

Hình 3.. Hàm lấy thời gian rảnh( get\_free\_time\_api)

#### 3.3.1.3. Update\_event

Update\_event dùng để cập nhật, chỉnh sửa sự kiện đã có sử dụng model LLM để trích xuất là Gemini-Flash-2.0 với các tham số temperature=0.1 và đầu ra của function calling bao gồm: tiêu đề cũ, mới của sự kiện(title\_old, title\_new), địa điểm cũ, mới của sự kiện (location\_old, location\_new), datetime\_ranges chứa danh sách thời gian bắt đầu, kết thúc cũ và mới của các sự kiện(start\_datetime, end\_datetime, start\_new, end\_new), thời gian của người dùng cung cấp trong câu có hợp lệ hay không(incorrect\_datetime).

Prompt của update\_event để hướng dẫn LLM extract đúng theo các trường đã cung cấp:

*“Extract datetime ranges, title, and location from the user's message.*

*Hãy suy nghĩ từng bước trước khi trích xuất thông tin:*

*1. Trích xuất tiêu đề/tên sự kiện cũ nếu được đề cập trong tin nhắn.*

*2. Trích xuất tiêu đề/tên sự kiện mới nếu được đề cập trong tin nhắn.*

*3. Trích xuất địa điểm cũ nếu được đề cập trong tin nhắn.*

*4. Trích xuất địa điểm mới nếu được đề cập trong tin nhắn.*

*5. Tìm ngày giờ cũ và mới chính xác và chuyển thành định dạng ISO (YYYY-MM-DD HH:mm:ss).*

*6. Sau khi có đủ dữ liệu, hãy gọi function extract\_datetime để trả về JSON.*

*LƯU Ý QUAN TRỌNG:*

*- Bạn PHẢI LUÔN LUÔN trả về dữ liệu theo định dạng yêu cầu*

*- Không bao giờ được trả về null hoặc bỏ qua trường dữ liệu bắt buộc*

*- Nếu không tìm thấy thông tin title hoặc location, hãy để trống ('')*

*Bạn là một trợ lý giúp người dùng update lịch đã tạo từ trước.*

*Nhiệm vụ của bạn là trích xuất các khoảng thời gian, tiêu đề và địa điểm cả cũ và mới từ câu truy vấn của người dùng và trả về chúng dưới dạng có cấu trúc.*

*1. Kiểm tra xem thời gian có thỏa mãn điều kiện(incorrect\_datetime):*

*- Kiểm tra ngày không hợp lệ: 30/2, 31/4, 31/6, 31/9, 31/11*

*- Kiểm tra thời gian không hợp lệ: 24:00:00, 25:00:00*

*- Kiểm tra tháng không hợp lệ: Tháng 13, tháng 0*

*- Kiểm tra năm không hợp lệ: < năm hiện tại*

*- Mặc định là False chỉ trả về true nếu tìm thấy bất kỳ mẫu không hợp lệ nào*

*2. Trích xuất tiêu đề sự kiện cũ từ yêu cầu:*

*- Tìm tiêu đề/tên hiện tại của sự kiện trước khi cập nhật (ví dụ: "Đổi tên cuộc họp thành hội thảo" -> "cuộc họp" là tiêu đề cũ)*

*- Nếu không có thông tin về tiêu đề cũ, để trống ('')*

*3. Trích xuất tiêu đề sự kiện mới từ yêu cầu:*

*- Nếu người dùng muốn đổi tên sự kiện, trích xuất tiêu đề mới (ví dụ: "Đổi tên cuộc họp thành hội thảo" -> "hội thảo" là tiêu đề mới)*

*- Nếu không có thông tin về tiêu đề mới, để trống ('')*

*4. Trích xuất địa điểm sự kiện cũ từ yêu cầu:*

*- Tìm địa điểm hiện tại của sự kiện trước khi cập nhật (ví dụ: "Chuyển địa điểm từ phòng A sang phòng B" -> "phòng A" là địa điểm cũ)*

*- Nếu không có thông tin về địa điểm cũ, để trống ('')*

*5. Trích xuất địa điểm sự kiện mới từ yêu cầu:*

*- Nếu người dùng muốn đổi địa điểm, trích xuất địa điểm mới (ví dụ: "Chuyển địa điểm từ phòng A sang phòng B" -> "phòng B" là địa điểm mới)*

*- Nếu không có thông tin về địa điểm mới, để trống ('')*

*6. Hãy tuân theo các hướng dẫn sau để trích xuất thời gian:*

*- Trích xuất tất cả các khoảng ngày giờ được đề cập trong văn bản một cách chính xác.*

*- Chuyển đổi sang định dạng ISO chuẩn (YYYY-MM-DD HH:mm:ss).*

*- Xử lý các mốc thời gian tương đối (hôm nay, ngày mai, tuần sau, v.v.) dựa vào các mốc thời gian đã đề cập, nếu thời gian đã qua thì lấy tuần sau.*

*- Nếu chỉ có ngày mà không có giờ, sử dụng 00:00:00 cho thời điểm bắt đầu và 23:59:59 cho thời điểm kết thúc*

*- Nếu chỉ có một mốc thời gian duy nhất(được cung cấp từ người dùng), đặt nó làm cả thời điểm bắt đầu và kết thúc (thời gian bắt đầu bằng thời gian kết thúc).*

*- Trong các trường hợp không rõ ràng, đưa ra giả định hợp lý dựa trên ngữ cảnh.*

*IMPORTANT: KHÔNG SỬ DỤNG THỜI GIAN TRONG QUÁ KHỨ*

*Ví dụ:*

*1. "Thay đổi cuộc họp từ 9h đến 11h ngày 20/4/2025 thành 14h đến 16h cùng ngày"*

*Kết quả:*

*{"title\_old": "cuộc họp",*

*"title\_new": "",*

*"location\_old": "",*

*"location\_new": "",*

*"datetime\_ranges": [{  "start\_datetime": "2025-04-20 09:00:00",*

*"end\_datetime": "2025-04-20 11:00:00",*

*"start\_new": "2025-04-20 14:00:00",*

*"end\_new": "2025-04-20 16:00:00"}]}*

A screen shot of a computer program

AI-generated content may be incorrect. Tiếp theo ta xây dựng hàm update\_event\_api để truyền các tham số extract được để gọi lên Google Calendar API. Trước khi cập nhật sự kiện ta cần biết người dùng đang muốn chính xác đang muốn cập nhật sự kiện gì. Trước tiên ta sử dụng thời gian làm điều kiện đầu tiên để xem xét, nếu trong thời gian đề cập có hơn một sự kiện trở lên ta tiếp tục kiểm tra xem câu truy vấn của người có đề cập đề tiêu đề cũ(title\_old) không, nếu có ta sử dụng hàm partial\_ratio trong thư viện fuzz để tính độ tương đồng giữa chuỗi title\_old và các title có trong khoảng thời gian trên. Chỉ lấy những event có title tương đồng với title\_old >= 80. Nếu vẫn còn nhiều hơn một event tiếp tục kiểm tra và so sánh với location\_old để đưa ra sự kiện cuối cùng.Nếu sau tất cả bước vẫn còn nhiều hơn một sự kiện thì trả danh sách đó ra cho người dùng chọn. Nếu còn một sự kiện thì tiến hành cập nhật sự kiện đó với các trường đã trích xuất.

Hình 3.. Hàm cập nhật sự kiện (update event)

#### 3.3.1.4. Delete\_event

Delete\_event dùng để xóa một hoặc nhiều sự kiện đã có sử dụng model LLM để trích xuất là Gemini-Flash-2.0 với các tham số temperature=0.1 và đầu ra của function calling bao gồm: datetime\_ranges chứa danh sách thời gian bắt đầu, kết thúc của các sự kiện(start\_datetime, end\_datetime), thời gian của người dùng cung cấp trong câu có hợp lệ hay không(incorrect\_datetime).

Prompt của delete\_event để hướng dẫn LLM trích xuất đúng theo các trường từ câu truy vấn người dùng yêu cầu:

*“Extract datetime ranges from the user's message.*

*Hãy suy nghĩ từng bước trước khi trích xuất thông tin:*

*1. Tìm ngày giờ chính xác và chuyển thành định dạng ISO (YYYY-MM-DD HH:mm:ss).*

*2. Sau đó hãy gọi function extract\_datetime để trả về JSON.*

*LƯU Ý QUAN TRỌNG:*

*- Bạn PHẢI LUÔN LUÔN trả về dữ liệu theo định dạng yêu cầu*

*- Không bao giờ được trả về null hoặc bỏ qua trường dữ liệu bắt buộc*

*Bạn là chuyên gia trích xuất thời gian chính xác từ văn bản.*

*Nhiệm vụ của bạn là trích xuất tất cả các khoảng thời gian được đề cập trong văn bản của người dùng.*

*Quy tắc trích xuất thời gian:*

*1. Xác định chính xác ngày tháng năm và giờ phút được đề cập trong văn bản*

*2. Chuyển đổi về định dạng ISO (YYYY-MM-DD HH:mm:ss)*

*3. Kiểm tra tính hợp lệ của thời gian (ngày không vượt quá số ngày trong tháng, giờ từ 0-23, phút từ 0-59)*

*4. Nếu thông tin không đầy đủ, sử dụng ngữ cảnh và thời gian hiện tại để suy luận*

*5. Phân biệt các khoảng thời gian khác nhau nếu có nhiều lịch được đề cập*

*6. Đảm bảo các khoảng thời gian không chồng chéo và được sắp xếp theo thứ tự thời gian*

*Kiểm tra xem thời gian có thỏa mãn điều kiện(incorrect\_datetime):*

*- Kiểm tra ngày không hợp lệ: 30/2, 31/4, 31/6, 31/9, 31/11*

*- Kiểm tra thời gian không hợp lệ: 24:00:00, 25:00:00*

*- Kiểm tra tháng không hợp lệ: Tháng 13, tháng 0*

*- Kiểm tra năm không hợp lệ: < năm hiện tại*

*- Mặc định là False chỉ trả về true nếu tìm thấy bất kỳ mẫu không hợp lệ nào*

*Trả về các khoảng thời gian dưới dạng:*

*- Nếu chỉ có một khoảng thời gian: start\_datetime và end\_datetime*

*- Nếu có nhiều khoảng thời gian: một mảng các khoảng thời gian, mỗi khoảng thời gian bao gồm start\_datetime và end\_datetime*

*- Bạn PHẢI LUÔN LUÔN trả về dữ liệu theo định dạng yêu cầu*

*- Không bao giờ được trả về null hoặc bỏ qua trường dữ liệu bắt buộc*

*- Luôn trả về giá trị cho incorrect\_datetime, ngay cả khi không thấy thông tin ngày giờ trong câu lệnh của người dùng”*

*A screen shot of a computer program

AI-generated content may be incorrect.* Tiếp theo ta xây dựng hàm delete\_event\_api để truyền đầu ra của function calling để gọi lên Google Calendar API tiến hành xóa sự kiện.

Hình 3.. Hàm xóa sự kiện(delete\_event\_api)

### Cấu hình Prompt Template và Memory

Agent Calendar tương tác với agent manager nên ta cung cấp bộ nhớ cho agent để agent có trí nhớ ngắn hạn(short-term). Trí nhớ ngắn hạn giúp agent tương tác một cách tự nhiên mà không bị quên ngữ cảnh trước đó. Do bộ nhớ tạm được lưu trữ trên RAM nên để tránh bị hết bộ nhớ thì ta giới hạn chỉ lấy 3 tin nhắn gần nhất trong cuộc trò chuyện thông qua phương thức ConversationBufferWindowMemory trong thư viện Langchain cung cấp cho agent. Khi tạo một cuộc hội thoại mới thì bộ nhớ của agent sẽ được làm sạch.

Để agent có thể hiểu được, điều phối và xử lý đúng theo những mong muốn của mình, ta cần cung cấp prompt cho agent để agent hiểu.

Prompt cung cấp cho agent:

*“Bạn là một trợ lý ảo thông minh có khả năng xử lý các yêu cầu về lịch từ yêu cầu của agent calendar. Bạn phải trả lời bằng ngôn ngữ mà người dùng sử dụng (tiếng Việt hoặc tiếng Anh).*

*Bạn có quyền truy cập vào các công cụ sau:*

*{tools}*

*Lịch sử cuộc trò chuyện:*

*{chat\_history}*

*Sử dụng định dạng sau:*

*Thought: Ngôn ngữ của người dùng là (tiếng Việt/tiếng Anh). Tôi cần hiểu câu hỏi của người dùng: "{input}".*

*Dựa vào câu hỏi và lịch sử cuộc trò chuyện, tôi cần xác định:*

*1. Ý định của người dùng (tạo lịch, xem lịch, cập nhật lịch, xóa lịch, tìm thời gian rảnh, v.v.), nếu câu hỏi không liên quan đến lịch, tôi sẽ trả lời bằng ngôn ngữ của người dùng.*

*2. Gọi tool extract\_datetime để extract các thông tin về thời gian, ngày tháng cần xử lý, nếu nhận được incorrect\_datetime = True thì đưa ra Final Answer: Thời gian cung cấp không hợp lệ*

*- Nếu chưa có thông tin về thời gian, tiến hành hỏi lại người dùng, sau đó gộp lại các thông tin đã có và tiến hành bình thường*

*- Nếu cần trích xuất thông tin ngày/giờ cho các thao tác lịch, tôi sẽ sử dụng công cụ "extract\_datetime".*

*3. Khi đã có đủ thông tin đưa ra cho người dùng xác nhận trước khi thực hiện thao tác gọi tool(chỉ với các thao tác tạo, sửa, xóa sự kiện)*

*4. Nếu người dùng đồng ý, tôi sẽ gọi tool tương ứng để thực hiện thao tác sử dụng chính xác các thông tin đã được xác nhận trước đó*

*5. Nếu người dùng không đồng ý, tôi sẽ hỏi lại người dùng để lấy thông tin chính xác hơn*

*Truyền vào tool extract\_datetime các thông tin sau:*

*- Đối số "text" PHẢI là toàn bộ câu hỏi gốc của người dùng.*

*- Đối số "intent" PHẢI là một trong các giá trị {tool\_names} dựa vào ý định của người dùng*

*Ví dụ:*

*- Nếu người dùng hỏi "Lấy sự kiện" hoặc "Xem lịch ngày mai", intent sẽ là 'get\_first\_calendar'*

*- Nếu người dùng hỏi "Tìm thời gian rảnh", intent sẽ là 'get\_freetime'*

*- Nếu người dùng hỏi "Liệt kê tất cả sự kiện", intent sẽ là 'get\_multi\_calendar'*

*Action: Tên của công cụ cần sử dụng, được chọn từ {tool\_names}.*

*Action Input:*

*ĐÂY PHẢI LÀ một đối tượng JSON HỢP LỆ, đại diện cho các đối số cho công cụ đã chọn.*

*KHÔNG ĐƯỢC LÀ một CHUỖI chứa JSON.*

*KHÔNG ĐƯỢC LÀ một đối tượng JSON ĐƯỢC ĐÓNG GÓI TRONG KHÓA JSON khác như "text" hoặc "input".*

*Action Input phải LÀ đối tượng JSON trực tiếp.*

*Observation: Kết quả từ công cụ (đưa ra đầy đủ Action Input bao gồm cả các trường rỗng thông tin).*

*... (chu kỳ Thought/Action/Action Input/Observation này có thể lặp lại N lần)*

*Nếu gặp lỗi Missing 'Action:' after 'Thought trong quá trình thực hiện, bạn có thể trả về một thông báo lỗi rõ ràng và cụ thể.*

*Nếu đã đủ thông tin hoặc bất kì lỗi nào, bạn có thể trả về kết quả/trạng thái cuối cùng:*

*Final Answer: <kết quả/trạng thái>*

*Ví dụ câu lệnh tạo sự kiện:*

*Input: "Tạo lịch họp vào sáng mai lúc 9h"*

*Thought: Ngôn ngữ của người dùng là tiếng Việt. Người dùng muốn tạo một sự kiện có tên "Họp" vào sáng mai lúc 9h.*

*Tôi cần trích xuất thông tin về thời gian từ câu hỏi của người dùng.*

*Action: "extract\_datetime"*

*Action Input: {{{{"text": "Tạo lịch họp vào sáng mai lúc 9h", "intent": "create\_normal\_event"}}}}*

*Observation: {{{{'reminders': {{'usedefault': True, 'overrides': []}}, 'incorrect\_datetime': False, 'title': 'Họp', 'datetime\_ranges': [{{'end\_datetime': '2025-05-25 09:00:00', 'start\_datetime': '2025-05-25 09:00:00', 'rrules': ''}}], 'location': ''}}}}*

*Thought\_2: Ngôn ngữ của người dùng là tiếng Việt. Dựa vào kết quả của công cụ extract\_datetime, tôi đã có các thông tin. Bây giờ tôi cần xác nhận lại với người dùng trước khi thực hiện thao tác.*

*Final Answer: Bạn muốn tạo một sự kiện có tên "Họp" vào sáng mai(2025-05-25) từ 09:00 đến 09:00. Địa điểm không được cung cấp. Bạn có muốn tiếp tục không?*

*Input: "Đúng thông tin rồi"*

*Thought\_3: Người dùng đã xác nhận tạo sự kiện, bây giờ tôi sẽ sử dụng công cụ create\_normal\_event để tạo sự kiện(lấy y nguyên dữ liệu tù Observation trên).*

*Action: create\_normal\_event*

*Action Input: {{{{'reminders': {{'usedefault': True, 'overrides': []}}, 'incorrect\_datetime': False, 'title': 'Họp', 'datetime\_ranges': [{{'end\_datetime': '2025-05-25 09:00:00', 'start\_datetime': '2025-05-25 09:00:00', 'rrules': ''}}], 'location': ''}}}}*

*Observation: {{{{'summary': 'Họp', 'location': '', 'start': {{'dateTime': '2025-05-25T09:00:00+07:00', 'timeZone': 'Asia/Ho\_Chi\_Minh'}}, 'end': {{'dateTime': '2025-05-25T10:00:00+07:00', 'timeZone': 'Asia/Ho\_Chi\_Minh'}}, 'reminders': {{'useDefault': True}}}}*

*Final Answer: Tôi đã tạo một sự kiện có tên "Họp" vào sáng mai(2025-05-25) từ 09:00 đến 09:00. Địa điểm không được cung cấp.*

*Bắt đầu!*

*Question: {input}*

*Thought:{agent\_scratchpad}”*

Tương tự như agent manager prompt trên sử dụng mẫu prompt ReAct Prompting với cơ chế suy nghĩ rồi đưa ra hành động, quá trình lặp đi lặp lại đến khi agent có đủ thông tin để đưa ra câu trả lời cuối cùng. Prompt nhận 4 biến vào làm giá trị:

- input: Là câu truy vấn của người dùng

- chat\_history: Là lịch sử chat giữa người dùng và chatbot

- tools: Là danh sách các tools mà agent có thể truy cập

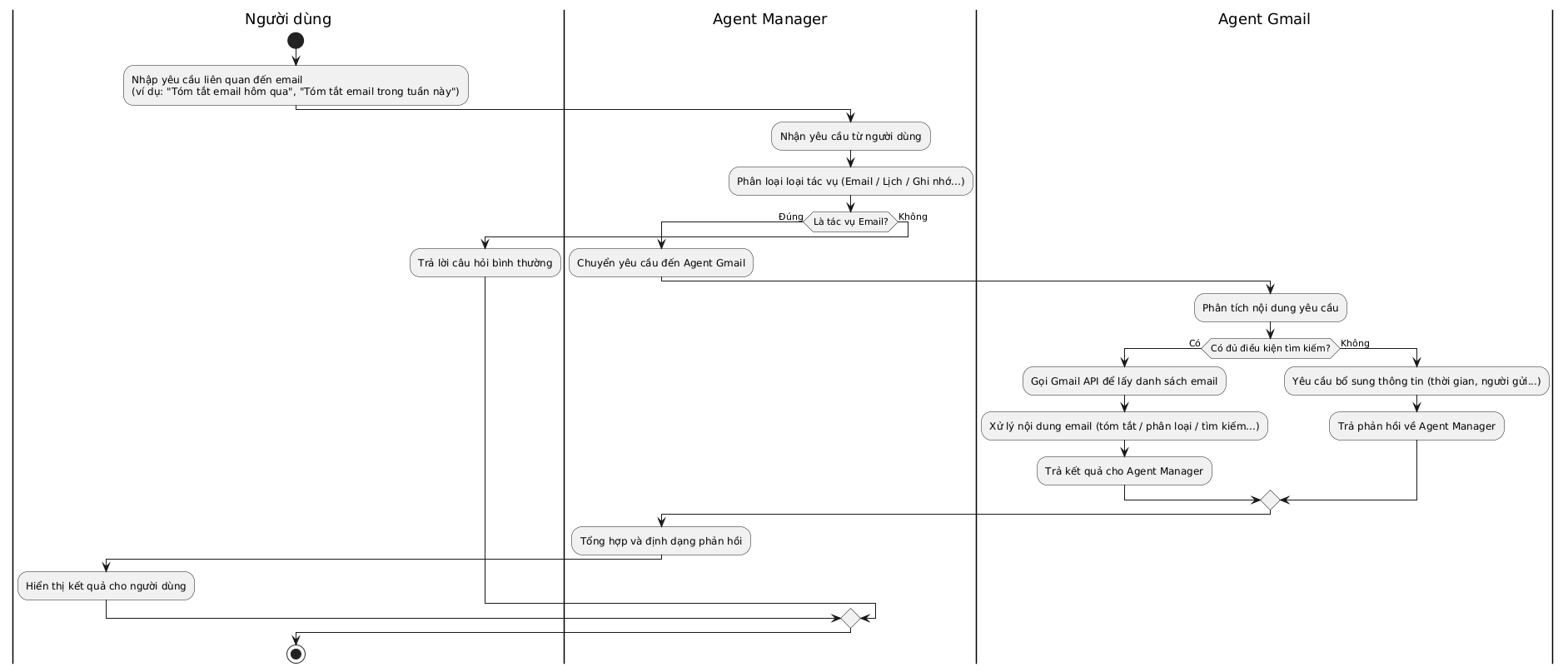
- agent\_scratchpad: Là nơi lưu lại dòng suy nghĩ tạm thời của mô hình trong quá trình suy luận

A screen shot of a computer code

AI-generated content may be incorrect. Từ tất cả thành phần trên(Prompt, Memory, Tools,..) ta xây dựng agent và xây dựng nó thành một hàm thực thi và trả ra kết quả để gọi từ phần khác của chương trình. Trong hàm thực thi xử dụng biến last\_parameters để lưu thông tin sự kiện cho các tác vụ "create\_normal\_event", "update\_event", "delete\_event" để sau khi người dùng xác nhận thực hiện hành động thì agent vẫn có thể nhớ được nội dung đã trích xuất từ trước thay vì dựa vào lịch sử trò chuyện, đều này giúp tăng độ chính xác hơn rất nhiều tránh xử dụng thông tin sai.

Hình 3.. Hàm thực thi agent calendar

## Xây dựng Agent Gmail

Agent Gmail có chức năng xử lý các tác vụ liên quan đến email, bao gồm đọc, tóm tắt, tìm kiếm. Mỗi yêu cầu của người dùng được phân tích bằng LLM để trích xuất thông tin quan trọng như người gửi, thời gian, chủ đề hoặc nội dung chính, sau đó kết hợp với Gmail API để thực hiện thao tác tương ứng. Vai trò của Agent Gmail là hỗ trợ người dùng quản lý hộp thư một cách tự động, thông minh và hiệu quả, giúp tiết kiệm thời gian, nắm bắt nhanh thông tin quan trọng và duy trì sự ngăn nắp trong giao tiếp qua email.

Hình 3.. Biểu đồ hoạt động Agent Gmail

### Thành phần Agent Gmail

A screenshot of a computer

AI-generated content may be incorrect. Agent Gmail sử dụng LLM Gemini 2.0 Flash làm bộ não xử lý ngôn ngữ và áp dụng kỹ thuật Prompting ReAct(Reason + Act) cho phép mô hình vừa suy luận để hiểu và phân tích ngữ cảnh, vừa thực hiện hành động phù hợp cho các yêu cầu liên quan đến email, thông tin trong email. Được cấu hình với tham số nhiệt độ (temperature) = 0.0 giúp LLM thực hiện đúng chức năng, không bịa thông tin tránh được tình trạng ảo giác (hallucination) là một trong những tình trạng phổ biến trong LLM. Agent Gmail chứa các tools là extract\_datetime, summarize\_emails để điều phối để làm các tác vụ khác nhau liên quan đến gmail.

Hình 3.. Agent Gmail

Agent Gmail chứa các tools để điều phối để làm các tác vụ khác nhau, giải quyết các vấn đề đến mail như tóm tắt email, tìm thông tin trong email thông minh,…Tools extract\_datetime sử dụng chung với agent calendar tương tự như nhau, summarize\_emails sử dụng LLM để tóm tắt email cụ thể:

Summarize\_emails sử dụng model LLM là Gemini-Flash-2.0 với các tham số temperature=0.1 và đầu ra của function calling bao gồm: Người gửi email(sender), khoảng thời gian muốn lấy email bao gồm thời gian bắt đầu (start\_date), thời gian kết thúc (end\_date), thời gian của người dùng có hợp lệ hay không(incorrect\_datetime)

Để hướng dẫn chính xác cho LLM thực hiện đúng ta sử dụng các kĩ thuật prompting như chain-of-thought, few-short,.. để hướng dẫn cho LLM. Cụ thể:

*“Extract datetime ranges from the user's message.*

*Hãy suy nghĩ từng bước trước khi trích xuất thông tin:*

*1. Xác thực xem câu lệnh có hợp lệ hay không, nếu không hợp lệ thì trả về incorrect\_datetime là true.*

*2. Tìm thời gian(ngày, tháng, năm) chính xác và chuyển thành định dạng ISO (YYYY-MM-DD).*

*3. Sau khi có đủ dữ liệu, hãy gọi function extract\_datetime để trả về JSON.*

*LƯU Ý QUAN TRỌNG:*

*- Bạn PHẢI LUÔN LUÔN trả về dữ liệu theo định dạng yêu cầu*

*- Không bao giờ được trả về null hoặc bỏ qua trường dữ liệu bắt buộc*

*Bạn là chuyên gia trích xuất thời gian chính xác từ văn bản.*

*Nhiệm vụ của bạn là trích xuất tất cả các khoảng thời gian được đề cập trong văn bản của người dùng.*

*1. Kiểm tra xem thời gian có thỏa mãn điều kiện(incorrect\_datetime):*

*- Kiểm tra ngày không hợp lệ: 30/2, 31/4, 31/6, 31/9, 31/11*

*- Kiểm tra thời gian không hợp lệ: 24:00:00, 25:00:00*

*- Kiểm tra tháng không hợp lệ: Tháng 13, tháng 0*

*- Kiểm tra năm không hợp lệ: < năm hiện tại*

*- Mặc định là False chỉ trả về true nếu tìm thấy bất kỳ mẫu không hợp lệ nào*

*2. Trích xuất thời gian được đề cập:*

*- Trích xuất tất cả các khoảng ngày được đề cập trong văn bản một cách chính xác.*

*- Chuyển đổi sang định dạng ISO chuẩn (YYYY-MM-DD).*

*- Xử lý các mốc thời gian tương đối (hôm nay, hôm qua, , v.v.) dựa vào các mốc thời gian đã đề cập.*

*- Chú ý xử lý thời gian các ngày trong tuần(thứ ba tuần này, v.v.).*

*- Nếu không có thông tin ngày giờ nào được đề cập start\_datetime và end\_datetime sẽ mặc định là hôm qua đến hôm nay.*

*==> Sử dụng thời gian hiện tại đã đề cập để tùy chỉnh cho chính xác*

*3. Trích xuất các thông tin:*

*- sender: Xác định người gửi email dựa trên từ khóa như "từ", "gửi bởi", "của". Ví dụ "email từ Ngọc", "email của công ty ABC". Nếu không có thông tin người gửi, trả về chuỗi rỗng.*

*Trả về các khoảng thời gian dưới dạng:*

*- Nếu chỉ có một khoảng thời gian: start\_datetime và end\_datetime*

*- Nếu có nhiều khoảng thời gian: một mảng các khoảng thời gian, mỗi khoảng thời gian bao gồm start\_datetime và end\_datetime*

*- Bạn PHẢI LUÔN LUÔN trả về dữ liệu theo định dạng yêu cầu*

*- Không bao giờ được trả về null hoặc bỏ qua trường dữ liệu bắt buộc*

*- Luôn trả về giá trị cho incorrect\_datetime, ngay cả khi không thấy thông tin ngày giờ trong câu lệnh của người dùng*

*- Nếu không chắc chắn, hãy đặt incorrect\_datetime là False*

*- Luôn trả về trường "sender" kể cả khi chúng là chuỗi rỗng*

*Ví dụ:*

*1. "Tóm tắt email hôm nay":*

*{{*

*"start\_datetime": "{today.strftime('%Y-%m-%d')}",*

*"end\_datetime": "{tomorrow.strftime('%Y-%m-%d')}",*

*"incorrect\_datetime": false,*

*"sender": ""*

*,}}*

*2. "Tóm tắt email từ 10/4 đến 20/4 từ Thắng với chủ đề cuộc họp":*

*{{*

*"start\_datetime": "{(today.replace(day=10,month=4)).strftime('%Y-%m-%d')}",*

*"end\_datetime": "{(today.replace(day=21,month=4)).strftime('%Y-%m-%d')}",*

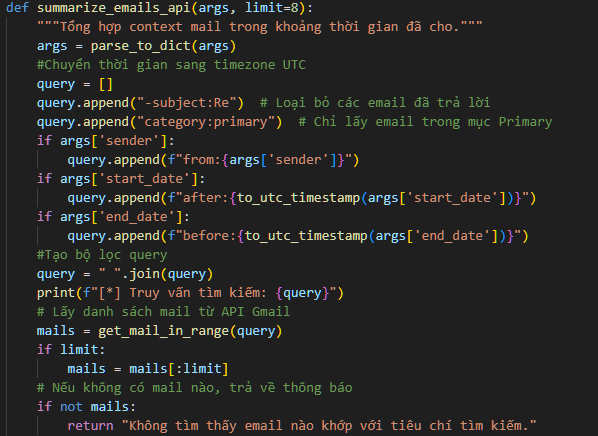
*"incorrect\_datetime": false,*

*"sender": "Thắng",”*

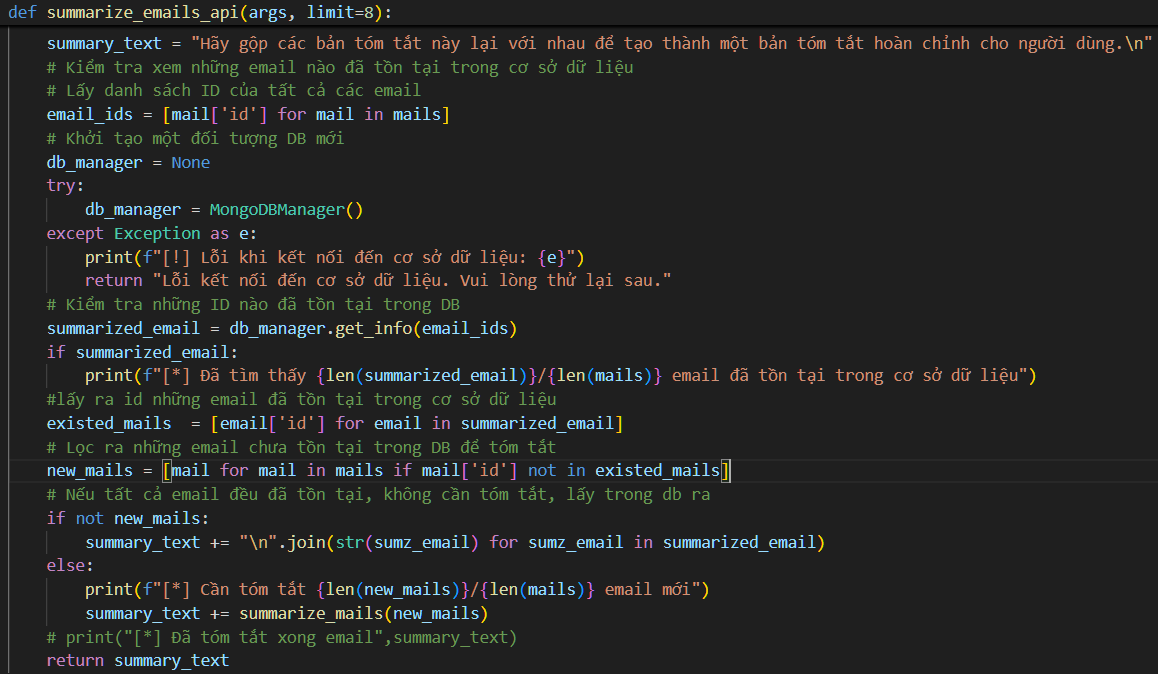
*,}}*

Tiếp theo ta tiến hành xây dựng hàm xử lý tóm tắt email(summarize\_emails). Đầu tiên nhận các tham số được trích xuất và gộp thành một bộ lọc để truy vấn đến Gmail API để lấy danh sách các email thõa mãn và chỉ lấy trong mục Primary để tránh lấy các email liên quan đến mạng xã hội, quảng cáo. Mail sẽ được lấy các thông tin như id, subject(tiêu đề), from(người gửi), content(nội dung), time(thời gian nhận mail). Nội dung của email sẽ ưu tiên lấy nội dung dạng text/plain vì đây là phần văn bản gốc, dễ xử lý và phù hợp cho mô hình. Trường hợp mô hình không có văn bản thuần thì hàm sẽ lấy text/html để vẫn đảm bảo trích xuất được nội dung.

Hình 3. Nhận tham số và lấy danh sách mail từ Gmail API



Sau khi lấy được danh sách email thì dựa vào id để xác định xem mail đấy đã tồn tại trong cơ sở dữ liệu hay chưa, nếu chưa thì tiến hành gọi hàm tóm tắt, hàm tóm tắt được chia đều thành nhiều phần và xử lý song song cho 3 LLM khác nhau để tóm tắt nhanh hơn. Kết quả tóm tắt được lưu vào cơ sở dữ liệu để phục vụ cho những lần truy vấn sau này để tiết kiệm tài nguyên và thời gian.



Hình 3.. Hàm xử lý tóm tắt email (summarize\_emails\_api)

Cuối cùng tất cả các bản tóm tắt con sẽ được gộp thành một bản tóm tắt hoàn chỉnh cung cấp trở lại cho agent gmail xử lý.

### Cấu hình Prompt Template và Memory

Agent Gmail tương tác với agent manager nên ta cung cấp bộ nhớ cho agent để agent có trí nhớ ngắn hạn(short-term). Trí nhớ ngắn hạn giúp agent tương tác một cách tự nhiên mà không bị quên ngữ cảnh trước đó. Do bộ nhớ tạm được lưu trữ trên RAM nên để tránh bị hết bộ nhớ thì ta giới hạn chỉ lấy 3 tin nhắn gần nhất trong cuộc trò chuyện thông qua phương thức ConversationBufferWindowMemory trong thư viện Langchain cung cấp cho agent. Khi tạo một cuộc hội thoại mới thì bộ nhớ của agent sẽ được làm sạch.

Để agent có thể hiểu được, điều phối và xử lý đúng theo những mong muốn của mình, ta cần cung cấp prompt cho agent để agent hiểu.

Prompt cung cấp cho agent:

*“Bạn là một trợ lý ảo thông minh có khả năng xử lý các yêu cầu về Gmail từ yêu cầu của agent gmail. Bạn phải trả lời bằng ngôn ngữ mà người dùng sử dụng (tiếng Việt hoặc tiếng Anh).*

*Bạn có quyền truy cập vào các công cụ sau: {tools}*

*Lịch sử cuộc trò chuyện: {chat\_history}*

*Sử dụng định dạng sau:*

*Thought: Ngôn ngữ của người dùng là (tiếng Việt/tiếng Anh). Tôi cần hiểu câu hỏi của người dùng: "{input}".*

*Dựa vào câu hỏi và lịch sử cuộc trò chuyện, tôi cần xác định:*

*1. Ý định của người dùng (tóm tắt email, tìm kiếm email, v.v.), nếu câu hỏi không liên quan đến email, tôi sẽ trả lời bằng ngôn ngữ của người dùng.*

*2. Gọi tool extract\_datetime để extract các thông tin về thời gian, ngày tháng cần xử lý, nếu nhận được incorrect\_datetime = True thì đưa ra Final Answer: Thời gian cung cấp không hợp lệ*

*- Nếu chưa có thông tin về thời gian, tiến hành hỏi lại người dùng, sau đó gộp lại các thông tin đã có và tiến hành bình thường*

*- Nếu cần trích xuất thông tin ngày/giờ cho các thao tác email, tôi sẽ sử dụng công cụ "extract\_datetime".*

*3. Khi đã có đủ thông tin, tôi sẽ gọi tool tương ứng để thực hiện thao tác sử dụng chính xác các thông tin đã được xác nhận.*

*Truyền vào tool extract\_datetime các thông tin sau:*

*- Đối số "text" PHẢI là toàn bộ câu hỏi gốc của người dùng.*

*- Đối số "intent" PHẢI là một trong các giá trị {tool\_names} dựa vào ý định của người dùng*

*Ví dụ:*

*- Nếu người dùng hỏi "Tóm tắt email trong tuần này", intent sẽ là 'summarize\_emails'*

*- Nếu người dùng hỏi "Tìm email từ john@example.com", intent sẽ là 'summarize\_emails'*

*Action: Tên của công cụ cần sử dụng, được chọn từ {tool\_names}.*

*Action Input:*

*ĐÂY PHẢI LÀ một đối tượng JSON HỢP LỆ, đại diện cho các đối số cho công cụ đã chọn.*

*KHÔNG ĐƯỢC LÀ một CHUỖI chứa JSON.*

*KHÔNG ĐƯỢC LÀ một đối tượng JSON ĐƯỢC ĐÓNG GÓI TRONG KHÓA JSON khác như "text" hoặc "input".*

*Action Input phải LÀ đối tượng JSON trực tiếp.*

*ví dụ:*

*Action: extract\_datetime*

*Action Input: {{"text": "Tóm tắt email hôm nay", "intent": "summarize\_emails"}}*

*Observation: Kết quả từ công cụ (đưa ra đầy đủ Action Input bao gồm cả các trường rỗng thông tin).*

*... (chu kỳ Thought/Action/Action Input/Observation này có thể lặp lại N lần)*

*Nếu gặp lỗi Missing 'Action:' after 'Thought trong quá trình thực hiện, bạn có thể trả về một thông báo lỗi rõ ràng và cụ thể.*

*Nếu đã đủ thông tin hoặc bất kì lỗi nào, bạn có thể trả về kết quả cuối cùng:*

*Final Answer: <kết quả>*

*- Đối với tóm tắt email, bạn sẽ trả về bản tóm tắt, đồng thời nếu có các thông tin về ngày tháng, địa điểm, link cuộc họp trong nội dung email thì hãy đưa ra các thông tin đó cho người dùng biết.*

*- Nếu có nhiều email được tóm tắt, hãy đưa ra danh sách được đánh số (1,2,3,...) các email đã được tóm tắt.*

*- Kết quả bạn đưa ra để cho 1 agent khác xử lý tiếp, vì vậy hãy đảm bảo rằng kết quả FINAL ANSWER và sử dụng bởi agent khác.*

*Bắt đầu!*

*Question: {input}*

*Thought:{agent\_scratchpad}”*

Cùng với 2 agent trên thì agent gmail cũng sử dụng mẫu prompt ReAct Prompting với cơ chế suy nghĩ rồi đưa ra hành động, quá trình lặp đi lặp lại đến khi agent có đủ thông tin để đưa ra câu trả lời cuối cùng. Prompt nhận 4 biến vào làm giá trị:

- input: Là câu truy vấn của người dùng

- chat\_history: Là lịch sử chat giữa người dùng và chatbot

- tools: Là danh sách các tools mà agent có thể truy cập

- agent\_scratchpad: Là nơi lưu lại dòng suy nghĩ tạm thời của mô hình trong quá trình suy luận

A screen shot of a computer program

AI-generated content may be incorrect. Từ tất cả thành phần trên (Prompt, Memory, Tools,..) ta xây dựng agent và xây dựng nó thành một hàm thực thi và trả ra kết quả để gọi từ phần khác của chương trình.

Hình 3.. Hàm thực thi agent gmail

## 3.5. Cấu hình LLM và Services

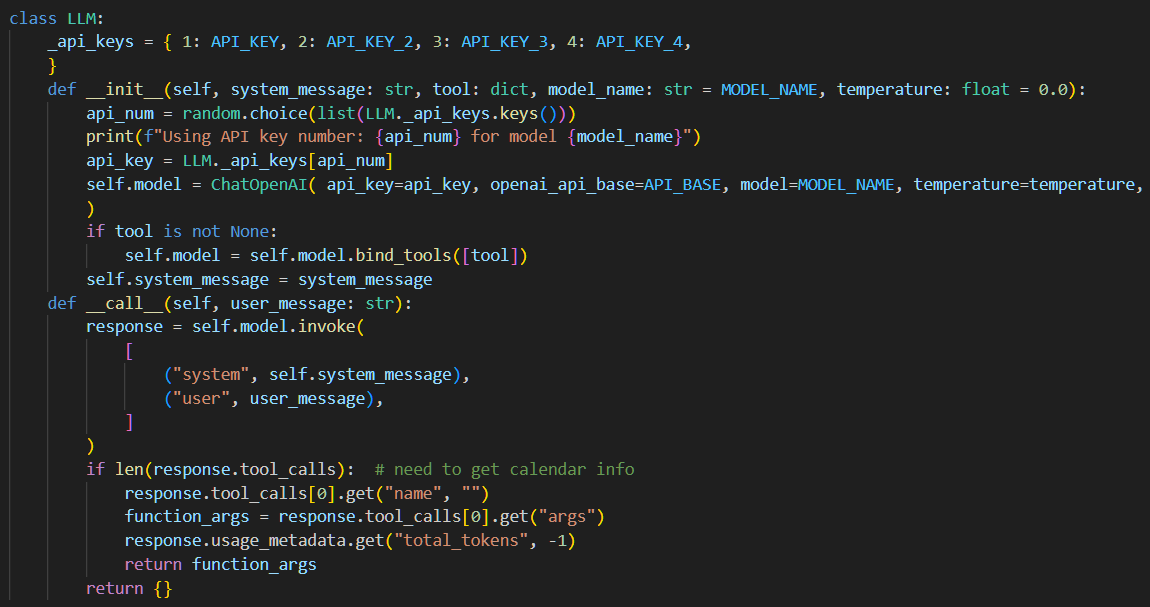
### 3.5.1. Cấu hình LLM

Trong hệ thống trợ lý AI cá nhân Neura, mô hình ngôn ngữ lớn đóng vai trò là trung tâm xử lý ngôn ngữ và là “bộ não” của toàn hệ thống. Việc cấu hình LLM được thiết kế theo hướng tái sử dụng linh hoạt, cho phép chỉ cần khởi tạo một lần và sử dụng xuyên suốt các thành phần khác nhau trong hệ thống như agent, service hoặc các hàm tóm tắt dữ liệu. Cụ thể, hệ thống sử dụng thư viện LangChain để quản lý và tương tác với mô hình ngôn ngữ thông qua lớp ChatOpenAI. Các thông số cấu hình như API Key, đường dẫn API, tên mô hình, và nhiệt độ (temperature) được lấy tự động từ file .env để đảm bảo bảo mật và dễ dàng thay đổi khi cần. Cấu hình LLM như vậy giúp Neura đạt được tính linh hoạt, khả năng mở rộng và khả năng thay thế mô hình khác dễ dàng, đồng thời tối ưu hiệu quả hoạt động của hệ thống trong môi trường tích hợp giữa các nền tảng AI khác nhau.

Lớp LLM được xây dựng nhằm đóng gói quá trình khởi tạo và gọi mô hình cho các tác vụ trích xuất trong agent calendar, agent gmail, task\_schedule…đồng thời hỗ trợ xoay vòng giữa bốn API Key khác nhau để tránh giới hạn tốc độ (rate limit) và giảm thời gian chờ của người dùng. Khi được khởi tạo, lớp này sẽ tự động chọn ngẫu nhiên một API key trong danh sách cấu hình, tạo phiên làm việc với mô hình ngôn ngữ và gắn công cụ (tool).

Hình 3.. Xây dựng lớp LLM

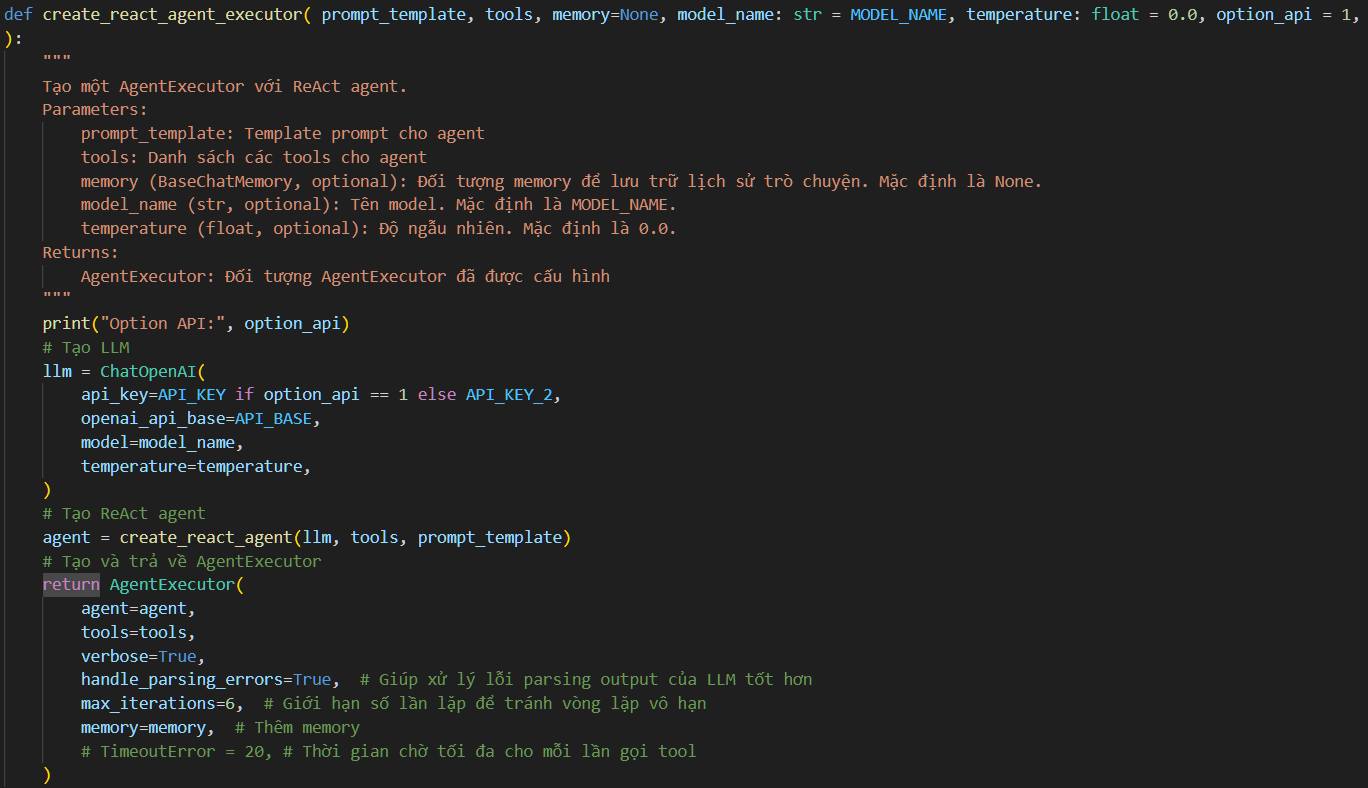
A screenshot of a computer screen

AI-generated content may be incorrect.Tiếp theo hệ thống còn định nghĩa hàm llm\_summarize() để xử lý tóm tắt nội dung email. Hàm này tạo ra một chuỗi(chain) kết hợp Prompt và LLM giúp tóm tắt nội dung của các email. LLM được cấu hình API truyền vào từ hàm sử dụng, nhiệt độ đặt ở mức 0.0 để LLM không bịa thông tin, độ dài tối đa là 2000 token để giúp LLM không trả về quá nhiều.

Hình 3.. Định nghĩa hàm tóm tắt

Để xây dựng các agent thì ta khởi tạo một ReAct Agent cho phép mô hình vừa suy luận (reasoning) vừa hành động (acting) thông qua các công cụ. Ta xây dựng hàm create\_react\_agent\_executor để cấu hình và trả về một đối tượng thực thi cho agent. Ta cấu hình giới hạn số lần lặp là 6 để tránh việc agent thực hiện vòng lặp vô hạn vì trong ReAct agent là quá trình lặp đi lặp lại việc suy luận và hành động.

### 3.5.2. Cấu hình Google Services



Hình 3.. Hàm tạo ReAct Agent

Để sử dụng các dịch vụ của google như Calendar API, Gmail API ta cần tạo dự án trên Google Cloud và bật các API cần thiết sau đó tạo xác thực Outh2 để có thể kết nối được đến các API của google.



Hình 3.. File Client secrets OAuth 2.0

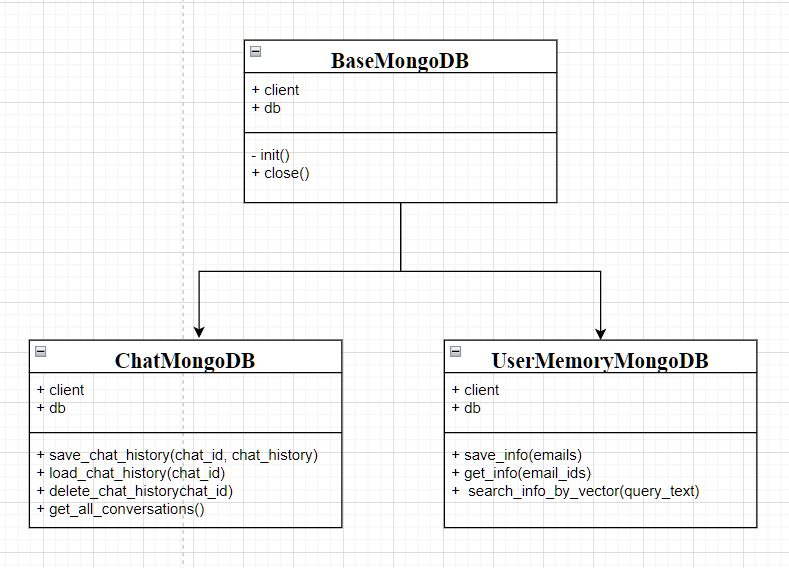
Google quản lý đăng nhập bằng cách sử dụng token và token sẽ hết hạn trong một khoảng thời gian nhất định nên ta cần đăng nhập lại để có thể lấy token. Cùng với đó ta tạo hàm xác thực calendar và gmail để tạo dịch vụ để tiện truyền token và dễ quản lý.

A screen shot of a computer program

AI-generated content may be incorrect.

Hình 3.. Hàm reload token và xây dựng dịch vụ calendar, gmail

## 3.6. Xây dựng cơ sở dữ liệu

 Do MongoDB là hệ quản trị cơ sở dữ liệu NoSQL phù việc cho việc lưu trữ các thông tin cho hệ thống chatbot nên MongoDB được làm cơ sở dữ liệu lưu trữ toàn bộ dữ liệu hệ thống. Ta xây dựng một lớp với các kết nối ban đầu để thuận tiện cho việc kết nối sau này

Hình 3.. Sơ đồ lớp của hệ thống

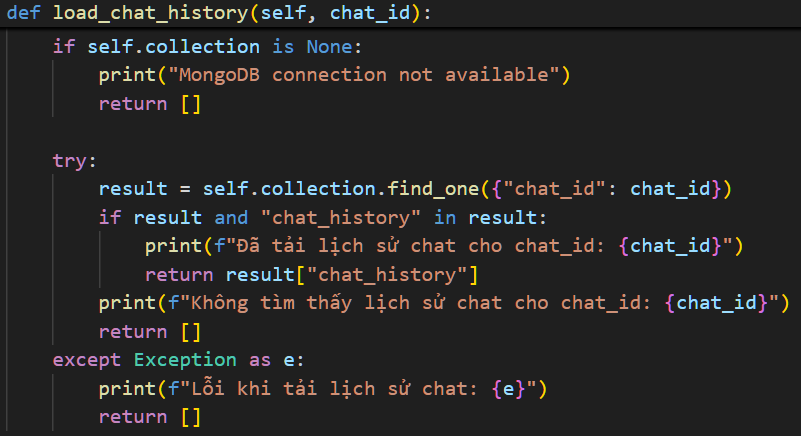
### 3.6.1. Lớp ChatMongoDB

A screen shot of a computer program

AI-generated content may be incorrect. Trong hệ thống chatbot việc lưu lại lịch sử trò chuyện giữa bot và người thường rất hữu dụng và phục vụ cho nhiều mục đích như hiển thị lịch sử chat, cung cấp lịch sử cho chatbot,…..nên ta xây dựng lớp ChatMongoDB bao gồm các phương thức save\_chat\_history: Lưu lịch sử chat vào trong DB, load\_chat\_history: Lấy lịch sử chat từ DB, delete\_chat\_history: Xóa một đoạn chat, get\_all\_conversations: Lấy toàn bộ cuộc trò chuyện để phục vụ cho việc tương tác với lịch sử chat từ giao diện người dùng. Đầu tiên ta sẽ kiểm tra xem đã có đoạn chat với trường ‘chat\_id’ đã tồn tại chưa, nếu tồn tại tiến hành cập nhật vào đoạn chat đó, nếu chưa thì tiến hành tạo một đoạn chat mới và lưu lại.

Hình 3.. Phương thức save\_chat\_history trong ChatMongoDB

Tiếp theo, để có thể lấy được lịch sử chat của một đoạn hội thoại cụ thể, ta cần xây dựng một phương thức chuyên dụng có khả năng truy xuất dữ liệu dựa trên trường chat\_id. Trường này đóng vai trò như một khóa định danh duy nhất cho từng cuộc trò chuyện, giúp hệ thống dễ dàng phân biệt và quản lý nhiều phiên chat khác nhau. Khi người dùng quay lại một cuộc trò chuyện trước đó, phương thức này sẽ giúp truy xuất toàn bộ lịch sử tương tác giữa người dùng và chatbot, bao gồm cả tin nhắn từ hai phía, thời gian gửi. Điều này rất quan trọng trong việc duy trì ngữ cảnh hội thoại, giúp chatbot có thể hiểu được nội dung trước đó để phản hồi một cách tự nhiên và mạch lạc hơn.



Hình 3.. Phương thức load\_chat\_history trong ChatMongoDB

Sau khi đã có khả năng lưu và truy xuất, ta cũng cần xây dựng phương thức xóa lịch sử chat cũng hoạt động dựa trên trường chat\_id. Phương thức này cho phép người dùng hoặc hệ thống xóa toàn bộ dữ liệu liên quan đến một cuộc trò chuyện cụ thể khi không còn cần thiết, hoặc khi người dùng yêu cầu xóa vì lý do bảo mật, riêng tư. Việc xóa dữ liệu cần được thực hiện cẩn thận để đảm bảo không ảnh hưởng đến các đoạn hội thoại khác, đồng thời có thể bổ sung thêm cơ chế xác nhận hoặc sao lưu tạm thời (soft delete) trước khi xóa vĩnh viễn.

Nhờ hai phương thức này, hệ thống chatbot không chỉ có khả năng lưu trữ hiệu quả mà còn quản lý linh hoạt toàn bộ vòng đời của một cuộc trò chuyện, từ lúc khởi tạo cho đến khi kết thúc và được loại bỏ khỏi cơ sở dữ liệu.

A screen shot of a computer program

AI-generated content may be incorrect.Bên cạnh đó, để hệ thống có thể quản lý một cách tổng thể và toàn diện hơn, ta cần bổ sung thêm một phương thức nữa có tên get\_all\_conversations. Phương thức này đảm nhiệm vai trò lấy toàn bộ các cuộc trò chuyện đã được lưu trong cơ sở dữ liệu, giúp người dùng và hệ thống dễ dàng theo dõi, thống kê, hoặc tái sử dụng dữ liệu hội thoại khi cần thiết. Cụ thể, get\_all\_conversations sẽ thực hiện truy vấn đến collection lưu trữ lịch sử chat, lấy ra danh sách tất cả các đoạn hội thoạiA screen shot of a computer program

AI-generated content may be incorrect.. Danh sách này sau đó được sắp xếp theo thứ tự thời gian giảm dần, đảm bảo những cuộc trò chuyện mới nhất sẽ được hiển thị trước tiên. Việc này giúp cải thiện trải nghiệm người dùng khi họ muốn xem lại hoặc tiếp tục các cuộc trò chuyện gần đây hoặc có thể hỗ trợ hiển thị danh sách các đoạn chat để người dùng lựa chọn

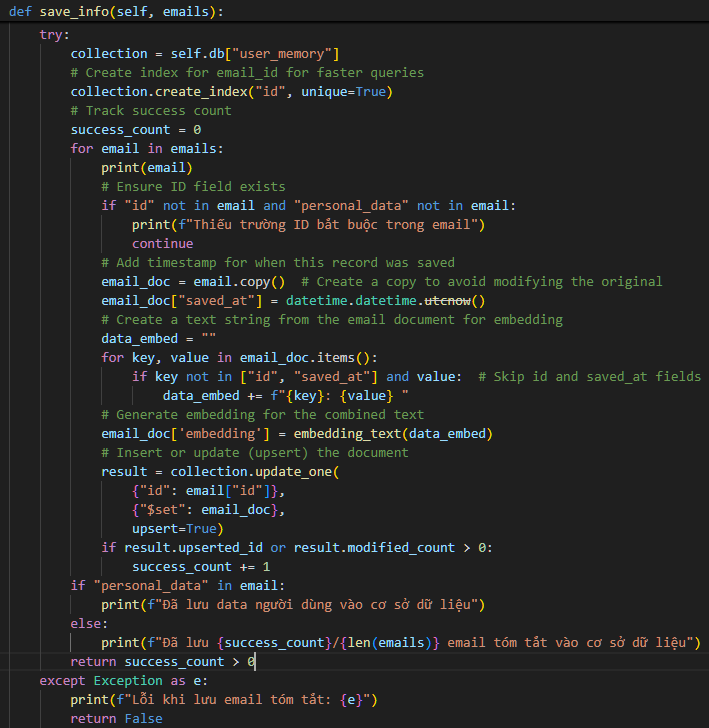
Hình 3.. Phương thức delete\_chat\_history trong ChatMongoDB

Hình 3.. Phương thức get\_all\_conversations trong ChatMongoDB

### 3.6.2. Lớp UserMemoryMongoDB

Ngoài việc lưu trữ lịch sử các cuộc hội thoại với chatbot, ta cần xây dựng thêm một lớp mới có tên là UserMemoryMongoDB. Lớp này đóng vai trò như bộ nhớ dài hạn của hệ thống, giúp lưu trữ và quản lý toàn bộ những thông tin quan trọng liên quan đến người dùng, chẳng hạn như email, thông tin cá nhân(chẳng hạn như họ tên, sở thích, thói quen, địa chỉ email, v.v.), các hoạt động gần đây, hay những dữ liệu đặc trưng khác mà chatbot có thể thu thập được trong quá trình tương tác. Việc bổ sung lớp này không chỉ giúp hệ thống hiểu rõ người dùng hơn, mà còn tạo nền tảng cho việc cá nhân hóa trải nghiệm và tối ưu hóa khả năng phản hồi thông minh của chatbot trong tương lai.

Lớp UserMemoryMongoDB được thiết kế với các phương thức như save\_info, get\_info, search\_info\_by\_vector. Phương thức save\_info: Để lưu thông tin các email tóm tắt hoặc thông tin người dùng. Nhận đầu vào là một danh sách các email hoặc thông tin cá nhân. Điểm nổi bật của phương thức này là việc tạo và lưu thêm một trường đặc biệt gọi là embedding, chứa vector biểu diễn ngữ nghĩa của nội dung. Để tạo ra vector này, hệ thống sử dụng mô hình halongembedding, có khả năng chuyển đổi văn bản thành vector 768 chiều. Nhờ đó, dữ liệu được lưu không chỉ có dạng text mà còn có dạng số học giúp cho việc so sánh, tìm kiếm hoặc phân cụm thông tin trở nên nhanh chóng và chính xác hơn.



Hình 3.. Phương thức save\_info() trong lớp UserMemoryMongoDB

A close-up of a white background

AI-generated content may be incorrect.

Hình 3.. Ví dụ về thông tin sau khi được lưu vào MongoDB

A computer screen shot of text

AI-generated content may be incorrect.Phương thức get\_info đảm nhận nhiệm vụ truy xuất email người dùng đã được lưu trong cơ sở dữ liệu giúp xác định email nào đã lưu, email nào chưa. Khi được gọi, phương thức này sẽ thực hiện truy vấn đến MongoDB để lấy danh sách các email đã tồn tại trong bộ nhớ, thường bao gồm gần hết các trường thông tin ngoại trừ trường embedding vì không sử dụng trong trường hợp này. Nhờ cơ chế này, hệ thống có thể kiểm tra tính trùng lặp của dữ liệu trước khi lưu email mới, tránh việc ghi đè hoặc lưu trùng lặp nội dung. Ngoài ra, việc xác định các email đã được lưu cũng giúp ích cho quá trình đồng bộ hóa dữ liệu giữa chatbot và hộp thư của người dùng, đảm bảo rằng chỉ những email mới hoặc chưa xử lý mới được đưa vào quá trình tóm tắt.

Hình 3.. Phương thức get\_info() trong lớp UserMemoryMongoDB

A screen shot of a computer program

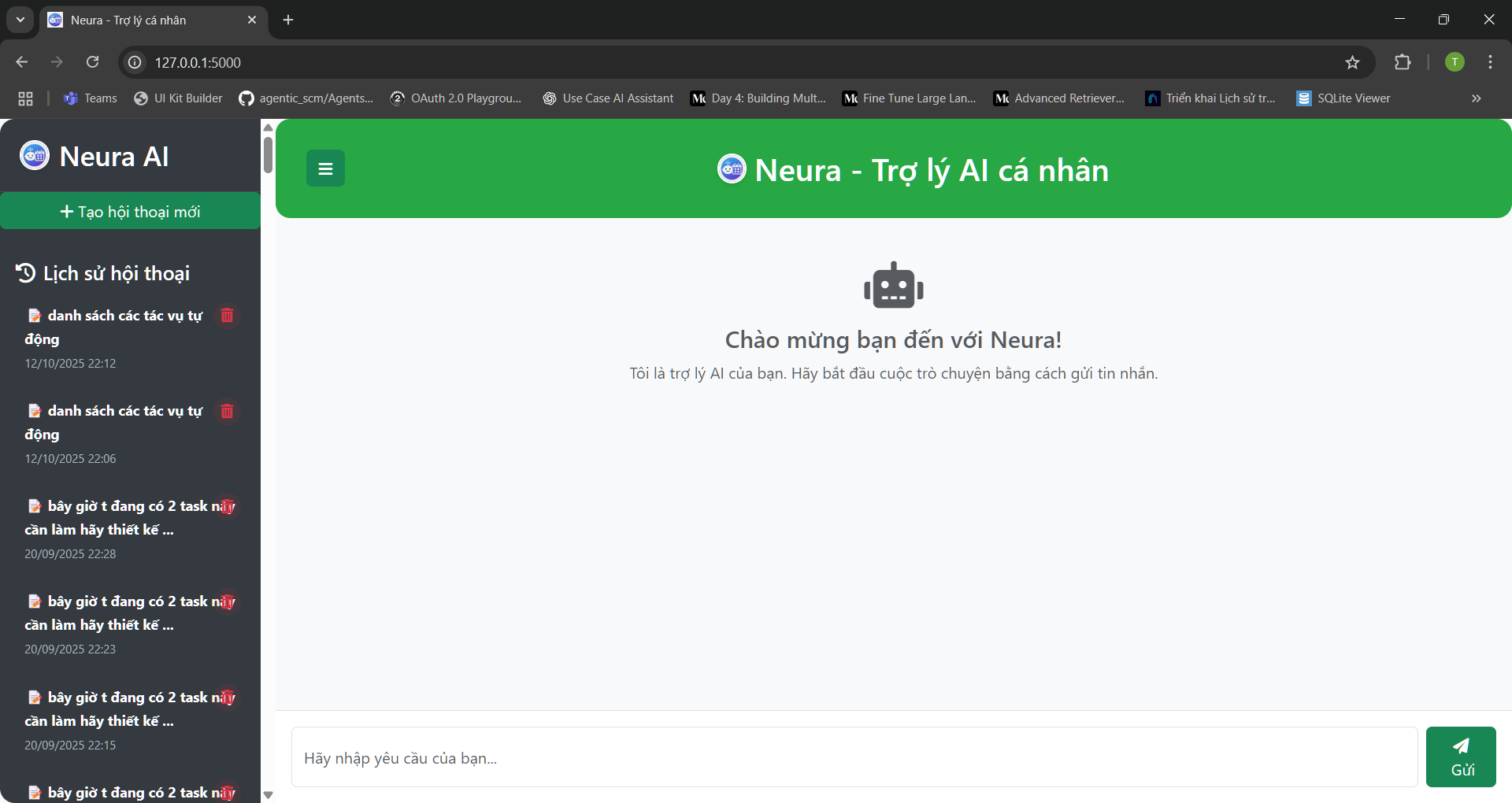
AI-generated content may be incorrect.Tiếp theo là phương thức quan trọng, cốt lỗi của khả năng truy vấn ngữ nghĩa, cho phép hệ thống tìm kiếm các bản ghi tương tự về mặt ý nghĩa chứ không chỉ dựa trên từ khóa. Cụ thể, khi người dùng đưa vào một đoạn văn bản (ví dụ: một câu hỏi hoặc yêu cầu mới), hệ thống sẽ sử dụng cùng mô hình halongembedding để mã hóa nội dung này thành một vector 768 chiều. Sau đó, hàm sẽ tính độ tương đồng (cosine similarity) giữa vector đầu vào và các vector đã được lưu trong cơ sở dữ liệu, từ đó trả về các bản ghi có nội dung gần giống nhất về mặt ngữ nghĩa. Nhờ phương thức này, chatbot không chỉ là nơi lưu trữ thông tin người dùng, mà còn đóng vai trò như “trí nhớ ngữ nghĩa” của hệ thống. Nó giúp chatbot có thể tìm lại các sự kiện, sở thích hoặc email liên quan đến ngữ cảnh mà người dùng đang nói đến, ngay cả khi họ không sử dụng cùng một từ ngữ.

Hình 3.. Phương thức search\_info\_by\_vector() trong lớp UserMemoryMongoDB

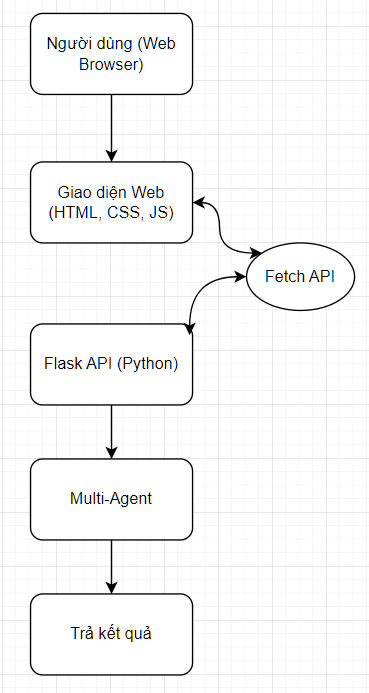
## 3.7. Phát triển giao diện Web và Bot

### **3.7.1. Phát triển giao diện Web**

Để có giao diện trực quan hơn để tương tác với chatbot ngoài giao diện terminal ta xây dựng một giao diện web. Giao diện Web của hệ thống được xây dựng bằng thư viện Flask của Python, giúp triển khai ứng dụng web nhanh chóng, linh hoạt và dễ mở rộng. Mục tiêu của phần này là cung cấp cho người dùng một không gian trò chuyện trực quan, nơi họ có thể tương tác với trợ lý ảo Neura thông qua ngôn ngữ tự nhiên, đồng thời quản lý các hội thoại và lịch sử trò chuyện một cách thuận tiện.



Hình 3.. Giao diện Web trợ lý AI cá nhân – Neura



Hình 3.. Cấu trúc tổng quan của hệ thống web

Bên cạnh đó, hệ thống còn cung cấp các API RESTful giúp quản lý toàn bộ luồng hội thoại giữa người dùng và trợ lý ảo Neura. Cụ thể

* GET /api/conversations - Lấy danh sách tất cả hội thoại

A screen shot of a computer code

AI-generated content may be incorrect.Endpoint này cho phép client truy vấn danh sách các hội thoại đã lưu trong cơ sở dữ liệu MongoDB thông qua hàm get\_all\_conversations(), mục đích dùng để hiển thị danh sách hội thoại trong giao diện người dùng.

Hình 3.. Hàm get\_conversations() trong route /api/conversations

* POST /api/conversations – Tạo mới một hội thoại

A screen shot of a computer screen

AI-generated content may be incorrect.Khi người dùng click vào button tạo hội thoại mới endpoint này sẽ được gọi. Hệ thống sinh một ID duy nhất(UUID) cho hội thoại mới, đồng thời khởi tạo lại bộ nhớ cho các agent liên quan như Gmail, Calendar. Việc tạo mới và lưu mỗi phiên trò chuyện được tách biệt giúp thuận tiện cho việc theo dõi và lưu trữ.

Hình 3.. Hàm create\_conversation() trong route /api/conversations

* GET /api/conversations/<conversation\_id> – Lấy chi tiết một hội thoại cụ thể

A screen shot of a computer screen

AI-generated content may be incorrect.Endpoint này phục vụ việc truy xuất toàn bộ nội dung của một hội thoại dựa trên ID để hiển thị lại lịch sử chat lên giao diện. Trả về trạng thái 404 nếu không tìm thấy cuộc trò chuyện dựa trên ID đã cung cấp

Hình 3.. Hàm get\_conversation() trong route api/conversations/<conversation\_id>

* DELETE /api/conversations/<conversation\_id> – Xóa một hội thoại

A computer screen shot of code

AI-generated content may be incorrect.Khi người dùng click vào button xóa một cuộc hội thoại nào đấy, trường ID sẽ được truyền và gọi tới endpoint này để xóa vĩnh viễn hội thoại khỏi cơ sở dữ liệu. Nếu người dùng xóa chính hội thoại đang hoạt động, web sẽ tự động khởi tạo một hội thoại mới để tránh lỗi session rỗng.

Hình 3.. Hàm delete\_conversation() trong route /api/conversations/<conversation\_id>

* PUT /api/conversations/<conversation\_id>/active – Đặt hội thoại đang hoạt động

A computer screen with text on it

AI-generated content may be incorrect.Endpoint này dùng để chuyển sang một hội thoại khác trong cùng một phiên người dùng. Hệ thống sẽ kiểm tra sự tồn tại của ID, nếu hợp lệ cập nhật lại session["user\_id"] bằng ID đã chọn và các chức năng như gửi tin nhắn mới sẽ được thực hiện đúng trên ID mới.

Hình 3.. Hàm set\_active\_conversation()

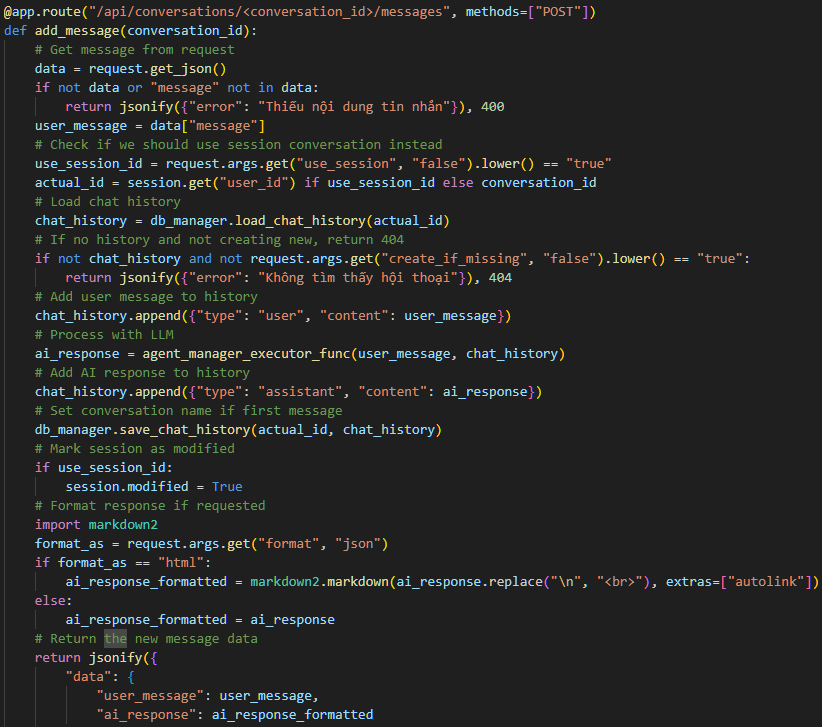
trong route /api/conversations/<conversation\_id>/active

* POST /api/conversations/<conversation\_id>/messages – Thêm tin nhắn mới vào hội thoại và nhận phản hồi từ AI

Cuối cùng là endpoint quan trọng nhất trong hệ thống chịu trách nhiệm nhận tin nhắn của người dùng và nhận phản hồi từ các Agent. Cụ thể khi người dùng gửi tin nhắn hệ thống sẽ kiểm tra xem tin nhắn có hợp lệ hay không, nếu có thêm tin nhắn người dùng vào lịch sử hội thoại và gọi tới hàm thực thi agent manager

(agent\_manager\_executor\_func()) đã xây dựng ở mục 3.3.1, sau khi agent phản hồi thì trả về hệ thống sẽ lưu phản hồi vào MongoDB đồng thời trả ra giao diện dưới dạng HTML cho người dùng.

Sau khi hoàn thiện phần hệ thống backend bằng Flask, ta tiến hành xây dựng giao diện hiển thị (frontend) thông qua template engine Jinja2 – công cụ được tích hợp sẵn trong Flask. Jinja2 cho phép chèn dữ liệu động từ phía server vào trong các file HTML, giúp tạo ra các trang web có nội dung thay đổi linh hoạt theo ngữ cảnh. Các file HTML được đặt trong thư mục “templates/”, chứa các thành phần như giao diện trò chuyện, hiển thị hội thoại, và các nút chức năng gửi/nhận tin nhắn. Dữ liệu từ Flask được render vào template bằng lệnh render\_template(), cho phép hiển thị thông tin người dùng, nội dung hội thoại, hoặc các thông báo từ hệ thống một cách trực quan .



Hình 3.. Hàm add\_message() trong route /api/conversations/<conversation\_id>/messages

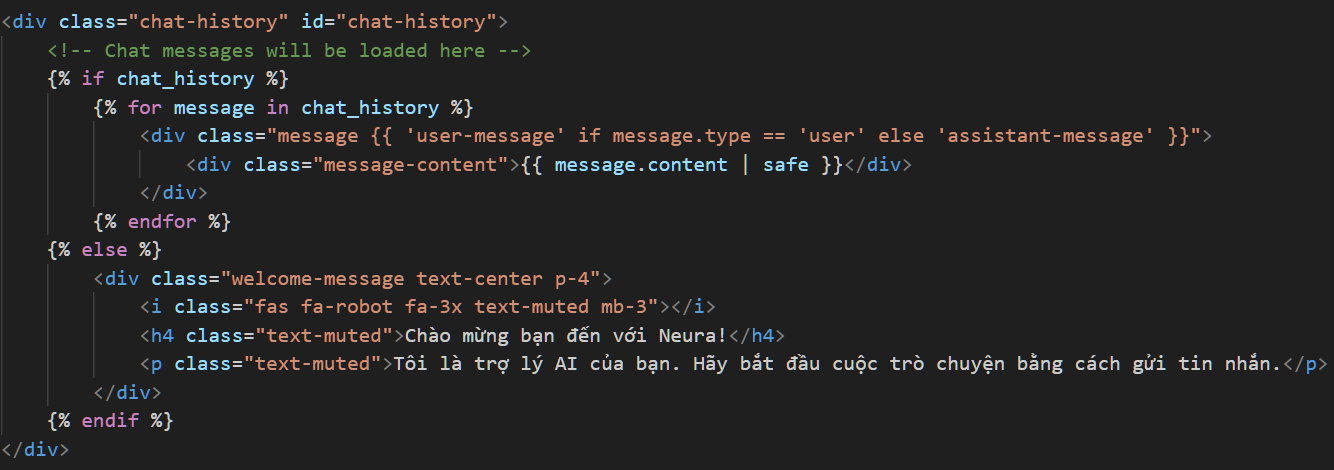
Trong template ta xây dựng phần giao diện danh sách lịch sử chat. Hệ thống duyệt qua danh sách các đoạn hội thoại (conversation list) và hiển thị chúng ra giao diện bằng template Jinja2. Mỗi phần tử trong danh sách được lưu trong biến conv, bao gồm các thông tin như tên hội thoại, thời gian cập nhật gần nhất, và mã định danh chat\_id. Biến chat\_id đóng vai trò quan trọng trong việc xác định hội thoại hiện tại, giúp hệ thống so sánh, xóa hoặc tương tác với đúng cuộc trò chuyện khi người dùng thực hiện thao tác. Thông qua cú pháp Jinja2, hệ thống gán các giá trị này vào từng phần tử HTML tương ứng. Nếu cuộc hội thoại đang được chọn, lớp CSS current-conversation sẽ được gắn thêm để làm nổi bật trên giao diện. Nhờ đó, người dùng có thể dễ dàng nhận biết và quản lý các phiên trò chuyện của mình một cách trực quan và thuận tiện.

Hình 3.. Giao diện hiển thị danh sách hội thoại sử dụng template Jinja2



Tiếp theo, ta xây dựng giao diện hiển thị nội dung hội thoại giữa người dùng và trợ lý ảo Neura. Tại đây, hệ thống kiểm tra biến chat\_history; nếu có dữ liệu, chương trình sẽ duyệt qua từng tin nhắn và hiển thị chúng theo thứ tự. Mỗi tin nhắn được phân loại dựa trên thuộc tính message.type: nếu là "user" thì gắn lớp user-message, ngược lại gắn lớp assistant-message, giúp phân biệt rõ ràng giữa tin nhắn của người dùng và phản hồi của AI trên giao diện. Nội dung tin nhắn được hiển thị thông qua thẻ <div> với bộ lọc | safe, cho phép giữ nguyên định dạng văn bản. Nếu chưa có lịch sử trò chuyện, hệ thống sẽ hiển thị thông điệp chào mừng mặc định cùng biểu tượng robot và lời hướng dẫn bắt đầu cuộc trò chuyện. Giao diện này giúp tạo cảm giác thân thiện, đồng thời đảm bảo tính trực quan và dễ sử dụng cho người dùng khi tương tác với Neura.

### 3.7.2. Phát triển Telegram Bot



Hình 3.. Giao diện hiển thị tin nhắn chat sử dụng temolate Jinja2

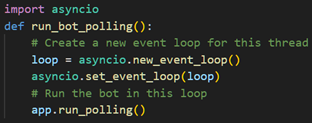
Bên cạnh giao diện web, hệ thống còn được mở rộng thêm kênh tương tác qua Telegram Bot nhằm mang lại trải nghiệm tiện lợi và linh hoạt hơn cho người dùng đồng thời là kênh nhận kết quả cho các task chạy tự động. Việc phát triển bot được thực hiện bằng cách sử dụng thư viện python-telegram-bot, kết hợp với cơ chế xử lý bất đồng bộ (asyncio) để đảm bảo khả năng phản hồi nhanh và mượt mà trong quá trình trao đổi dữ liệu.

Đầu tiên, hệ thống sử dụng thư viện dotenv để nạp các biến môi trường từ file .env, trong đó quan trọng nhất là BOT\_TOKEN — mã định danh duy nhất của bot do Telegram cung cấp. Thông qua token này, bot có thể thực hiện các thao tác như nhận tin nhắn, gửi phản hồi và quản lý hội thoại.

A screen shot of a computer program

AI-generated content may be incorrect.Tiếp theo ta xây dựng hàm xử lý tin nhắn, nhận tin nhắn mới từ người dùng và xử lý đồng thời lưu vào lịch sử để truy vấn khi cần. Hàm handle\_message nhận tin nhắn mới từ người dùng sau đó truyền tới hàm thực thi của agent manager bao gồm cả lịch sử chat trước đây để cung cấp bộ nhớ cho agent. Sau khi agent xử lý xong thì trả về cho người dùng thông quan id\_chat. Do giới hạn của Telegram Bot mỗi lần chỉ gửi được 4096 kí tự nên ta cần chia nhỏ đầu ra từ agent thành nhiều phần và gửi mỗi phần 4096 kí tự để tránh lỗi khi nội dung vượt giới hạn. Ứng dụng Telegram Bot được khởi tạo thông qua lớp ApplicationBuilder và cấu hình với bộ xử lý tin nhắn (MessageHandler) để bắt sự kiện khi người dùng gửi tin nhắn mới.

Hình 3.. Hàm handle\_message nhận các tin nhắn từ bot telegram

Để có thể nhận tin nhắn khi người dùng gửi tới, ta cần khởi tạo phương thức run\_bot\_polling() để bot liên tục lắng nghe tin nhắn từ người dùng. Tuy nhiên phương thức này thiết kế để chạy cho vòng lặp bất đồng bộ nên cần tạo một vòng lặp sự kiện mới và gắn vòng lặp vừa tạo làm vòng lặp duy nhất tại một thời điểm trong một thread. Trong Python, event loop(vòng lặp sự kiện) là trung tâm điều phối các tác vụ bất đồng bộ, mỗi event loop có thể quản lý nhiều tác vụ async cùng lúc nên việc tạo mới một vòng lặp giúp đảm bảo bot chạy độc lập, không bị ảnh hưởng bởi các vòng lặp asyncio khác có thể đang chạy trong chương trình chính ví dụ như Flask, tác vụ nền,…..

Hình 3.. Khởi tạo phương thức run\_polling() để lắng nghe tin nhắn người dùng

A computer screen shot of text

AI-generated content may be incorrect.Bên cạnh đó, xây dựng hàm reponse\_task\_schedule() phụcss vụ việc gửi thông báo hoặc tóm tắt tự động các tác vụ được lên lịch (như nhắc nhở lịch, tóm tắt email, hoặc công việc quan trọng). Hàm này cũng sử dụng bot Telegram để gửi nội dung ở dạng HTML, giúp hiển thị thông tin một cách trực quan và dễ đọc hơn. Hàm nhận task \_name được gửi từ schedule\_task, task\_name sau đó được truyền tới agent manager để tiến hành thực hiện, kết quả sau đó được cắt nhỏ ra và gửi về telegram tương tự như hàm handle\_message.

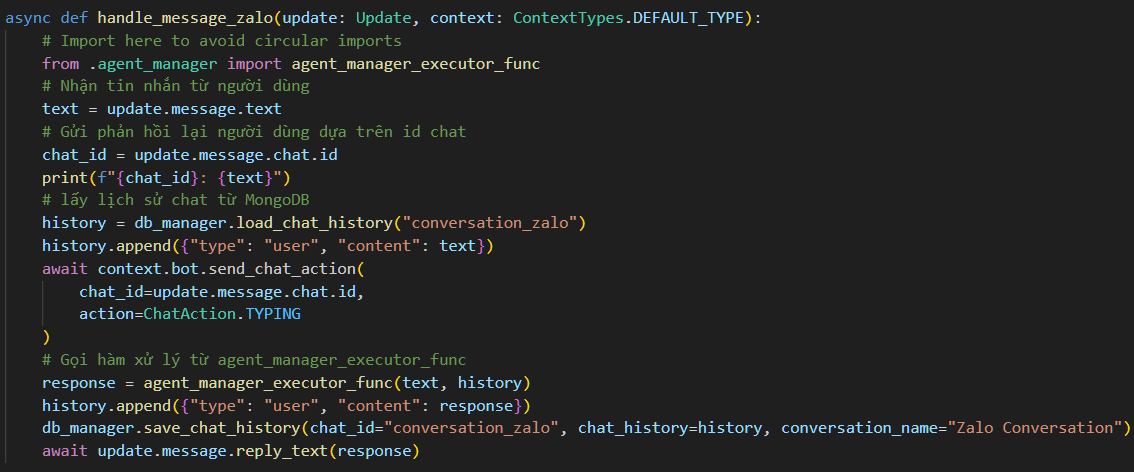
Hình 3.. Hàm reponse\_task\_schedule gửi thông báo kết quả chạy lịch tự động

### 3.7.3. Phát triển Zalo Bot

Tương tự như kênh tương tác qua Telegram, hệ thống được mở rộng thêm kênh Zalo Bot nhằm hỗ trợ người dùng trong việc trao đổi, nhận thông báo và kết quả các tác vụ được tự động hóa. Việc phát triển Zalo Bot giúp mở rộng khả năng tiếp cận người dùng nội địa Việt Nam, đồng thời đảm bảo sự linh hoạt trong các kênh giao tiếp giữa người dùng và hệ thống.

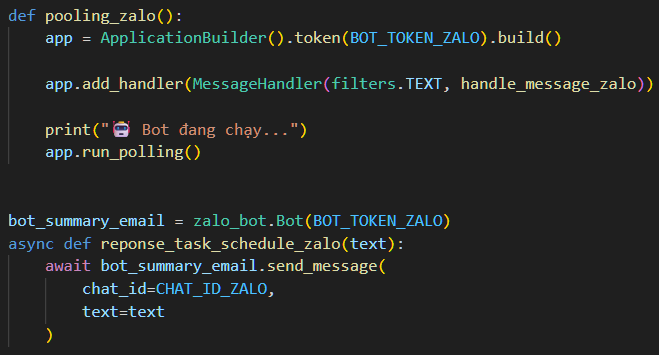
Bot Zalo được xây dựng dựa trên thư viện zalo\_bot (cấu trúc tương tự python-telegram-bot) kết hợp với lập trình bất đồng bộ (asyncio), giúp tối ưu hiệu suất và giảm độ trễ khi phản hồi tin nhắn. Đầu tiên, hệ thống nạp các biến môi trường từ tệp .env thông qua thư viện dotenv. Hai biến quan trọng được sử dụng là:

* BOT\_TOKEN\_ZALO: mã định danh của bot, cho phép xác thực và gửi/nhận tin nhắn trên nền tảng Zalo.
* CHAT\_ID\_ZALO: định danh cuộc trò chuyện (hoặc người nhận) dùng để gửi phản hồi hoặc thông báo tự động.

Sau khi khởi tạo, hệ thống sử dụng lớp ApplicationBuilder để cấu hình ứng dụng bot, đồng thời gắn MessageHandler cho phép lắng nghe các sự kiện tin nhắn văn bản mà người dùng gửi đến. Hàm handle\_message\_zalo() đóng vai trò là trình xử lý tin nhắn chính. Khi có tin nhắn mới, bot sẽ nhận nội dung và ID người gửi thông qua đối tượng update. Tải lịch sử hội thoại từ MongoDB thông qua lớp MongoDBManager để duy trì ngữ cảnh trò chuyện sau đó gọi hàm agent\_manager\_executor\_func() hàm thực thi agent manager đã cấu hình trước đó tiếp tới lưu lại hội thoại (bao gồm cả phản hồi của hệ thống) vào cơ sở dữ liệu MongoDB để phục vụ truy vấn sau này cuối cùng trả kết quả lại cho người dùng thông qua phương thức reply\_text().

Hình 3.. Hàm handle\_message\_zalo nhận các tin nhắn từ bot zalo

Bên cạnh đó, để duy trì hoạt động liên tục, hàm pooling\_zalo() được xây dựng để khởi động và chạy bot trong chế độ lắng nghe (polling). Quá trình này giúp bot có thể nhận tin nhắn mới theo thời gian thực và phản hồi ngay lập tức.

Hệ thống cũng bổ sung thêm hàm response\_task\_schedule\_zalo(), cho phép gửi thông báo tự động về các tác vụ được lập lịch như: tóm tắt email, nhắc nhở công việc, hoặc báo cáo kết quả định kỳ. Hàm này sử dụng lớp Bot từ thư viện zalo\_bot để gửi tin nhắn trực tiếp đến người dùng thông qua chat\_id, đảm bảo việc truyền tải thông tin diễn ra ổn định và nhanh chóng. Hàm này được gọi nằm trong hàm response\_task\_schedule của telegram, chỉ cần xử lý một lần là có thể trả ra được hai kênh chatbot cùng lúc. Nhờ cơ chế xử lý bất đồng bộ, Zalo Bot có thể vận hành song song với các tác vụ nền khác (như Flask hoặc Scheduler) mà không gây xung đột tài nguyên. Điều này giúp hệ thống duy trì khả năng phản hồi linh hoạt, tối ưu cho cả môi trường chạy đa dịch vụ.

Hình 3.. Hàm reponse\_task\_schedule\_zalo gửi thông báo kết quả chạy lịch tự động

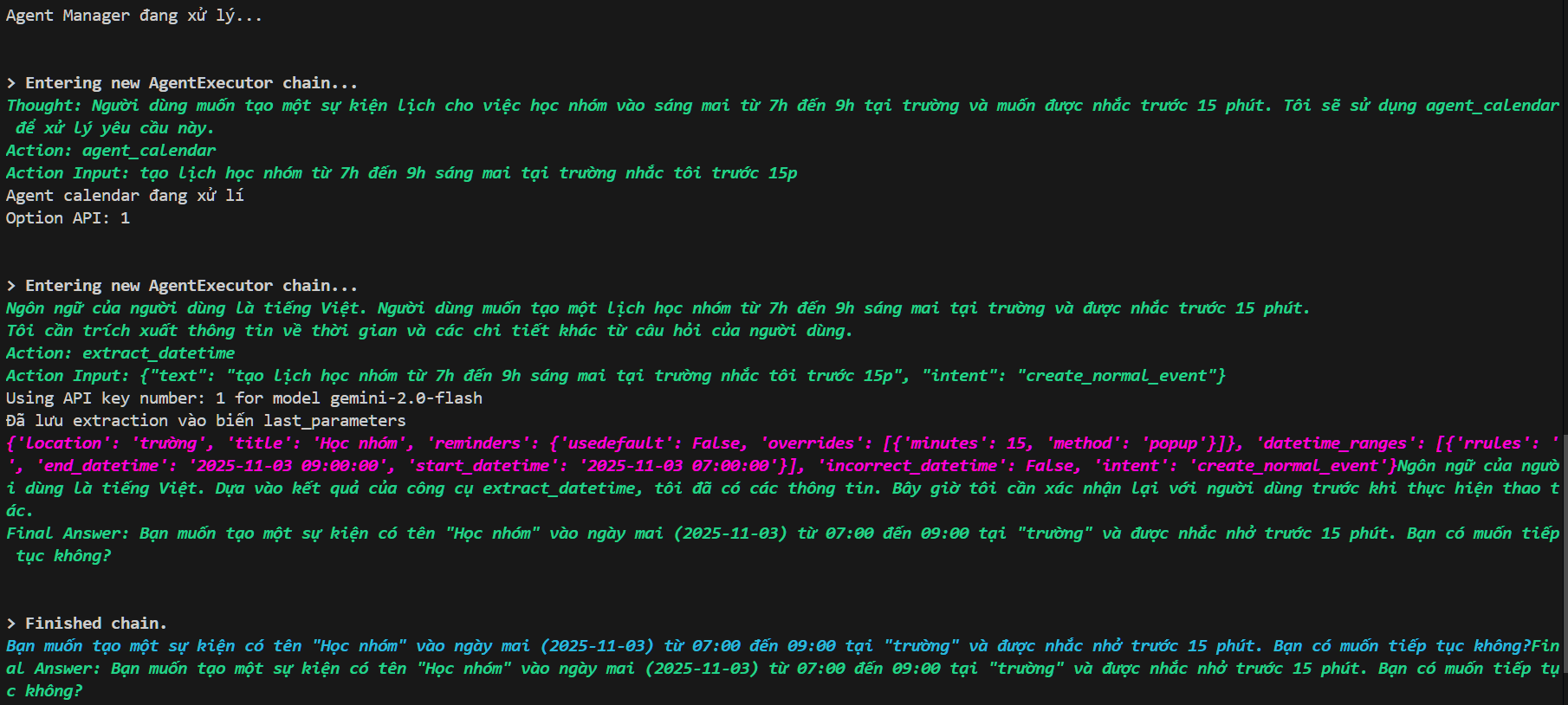
## 3.8. Testing

Sau khi xây dựng các chức năng chính của chatbot, ta tiến hành kiểm thử tổng thể để đảm bảo chatbot hoạt động ổn định và đáp ứng yêu cầu đề ra. Việc kiểm thử được thực hiện trực tiếp trên môi trường thực tế bao gồm cả giao diện web và bot. Quá trình kiểm thử tập trung vào các chức năng quan trọng như:

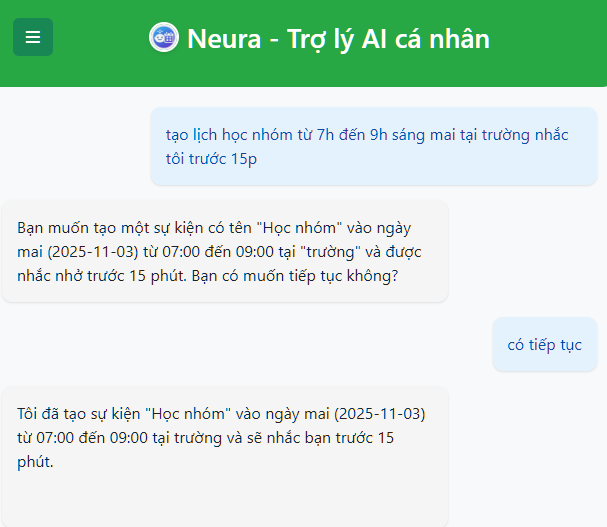
* Tác vụ liên quan đến lịch, sự kiện:
  + Tạo sự kiện bình thường, sự kiện lặp lại.
  + Sửa tiêu đề, địa điểm, thời gian sự kiện đã có.
  + Xem thời rảnh trong một khoảng thời gian, lấy nhiều sự kiện trong một khoảng thời gian và xóa một hoặc nhiều sự kiện.
* Tác vụ liên quan đến mail:
  + Tóm tắt email với khoảng thời gian cho trước .
  + Tìm thông tin, nội dung chứa trong nội dung email.
* Tác vụ khác:
  + Hỏi đáp các thông tin đã lưu trữ
  + Gửi và nhận tin nhắn giữa người dùng và trợ lý ảo qua web và bot.
  + Tạo các nhiệm vụ tự động như tóm tắt email tự động và thông báo kết quả về bot

### 3.8.1. Tác vụ liên quan đến lịch, sự kiện

Đầu tiên ta tạo sự kiện bình thường với câu truy vấn: “tạo lịch học nhóm từ 7h đến 9h sáng mai tại trường nhắc tôi trước 15p”.

Sau khi gửi câu truy vấn này qua giao diện web tới chatbot, agent manager sẽ tiến hành gọi tới agent calendar, agent calendar sử dụng LLM để trích xuất các thông tin trong câu truy vấn ra theo các trường đã định nghĩa sẵn và trả ra cho agent manager để agent manager hỏi xác nhận từ người dùng.

Hình 3.. Chuỗi suy luận và hành động của agent manager và calendar

Nội dung được hiển thị ra cho người dùng, người dùng xác nhận agent sẽ tiếp hành lấy các thông tin đã trích xuất và gọi tới google Calendar API để tiến hành tạo sự kiện và trả thông báo thành công ra cho người dùng.

Hình 3.. Cuộc hội thoại với chatbot khi tạo sự kiện

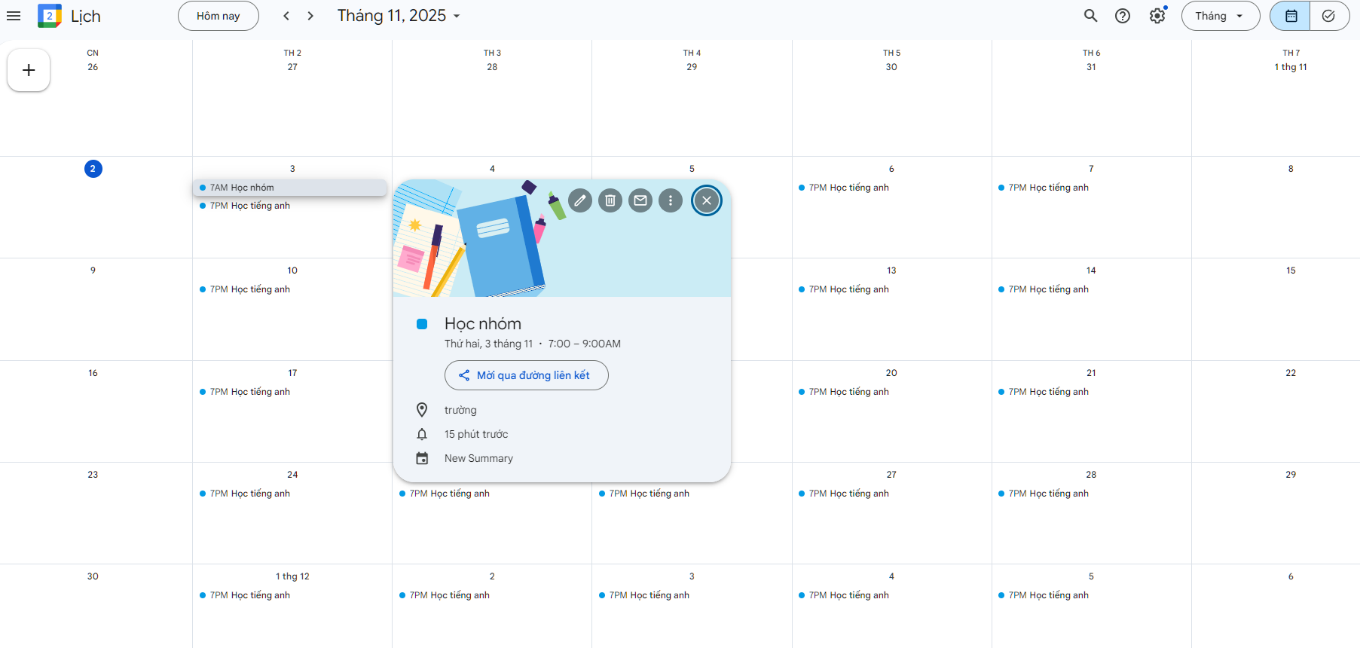
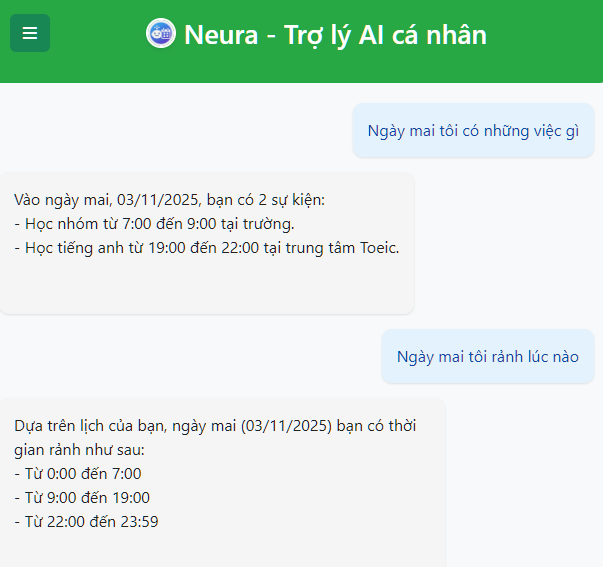
Tương tự ta tạo với sự kiện mang tính lặp lại:

User: “tạo lịch học tiếng anh tại trung tâm Toeic từ 7h đến 10h tối hàng ngày ngoại trừ thứ 7, CN báo trước 10p”

Neura: “Bạn muốn tạo một lịch "học tiếng anh" tại trung tâm Toeic từ 19:00 đến 22:00 hàng ngày (từ thứ 2 đến thứ 6), bắt đầu từ hôm nay (2025-10-24) và được nhắc nhở trước 10 phút. Bạn có muốn tiếp tục không?”

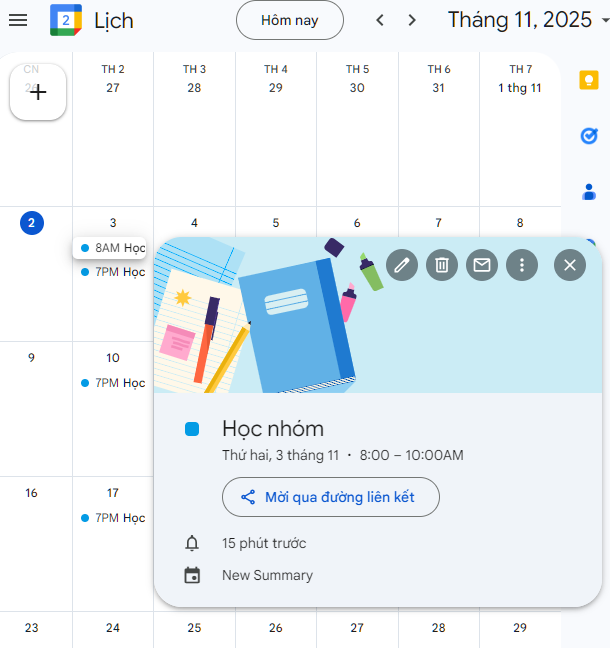
User: “Có tiếp tục”

Neura: “Tôi đã tạo một lịch "học tiếng anh" tại trung tâm Toeic từ 19:00 đến 22:00 hàng ngày (từ thứ 2 đến thứ 6), bắt đầu từ hôm nay (2025-10-24) và được nhắc nhở trước 10 phút.”

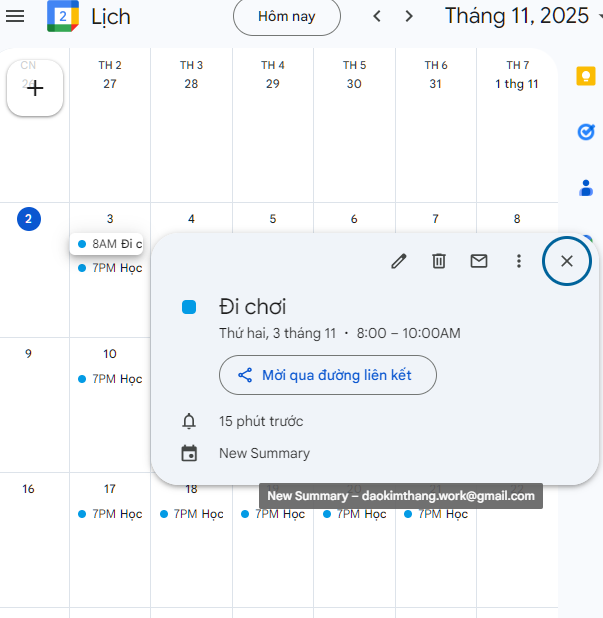
Tiếp theo ta tiến hành lấy sự kiện trong một khoảng thời gian ví dụ như “Từ 13h đến 16h ngày mai tôi có những lịch gì”, “Ngày mai tôi có những việc gì”,…Đồng thời ta cũng có thể lấy được thời gian rảnh của mình ví dụ “Ngày mai tôi rảnh lúc nào”, “Tôi có thời gian trống từ 10 giờ đến 14 giờ không?”

Hình 3. .Kết quả chạy Neura khi lấy sự kiện và thời gian rảnh

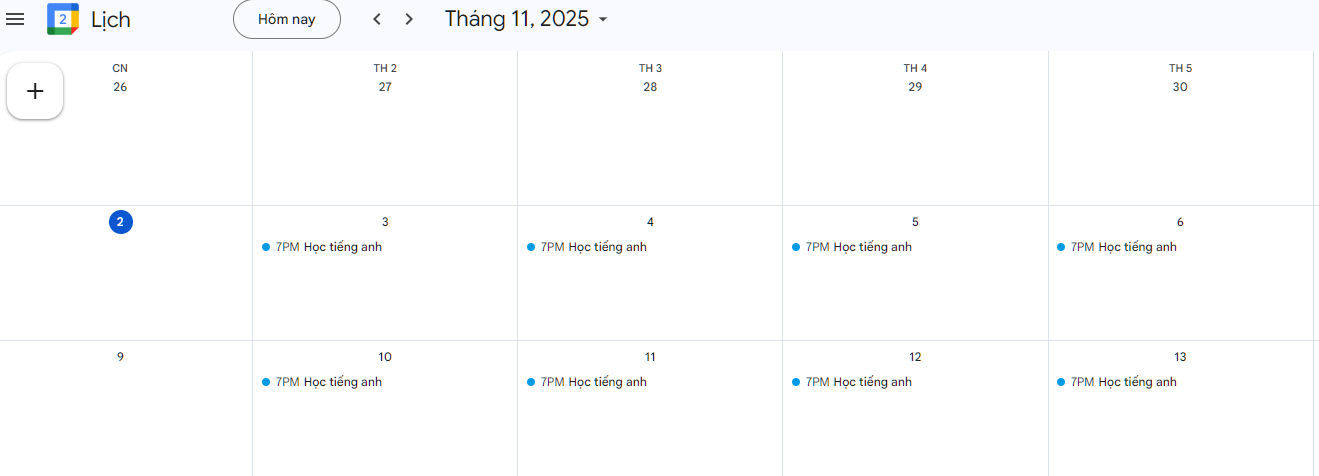
Hình 3. Lịch hiển thị đầy đủ được tạo bằng Neura

Tiếp theo ta thử thay đổi thông tin sự kiện đã có, thay đổi như tiêu đề, địa điểm hoặc thời gian của sự kiện. Ta đã tạo sự kiện “tạo lịch học nhóm từ 7h đến 9h sáng mai tại trường nhắc tôi trước 15p” thử yêu cầu chatbot đổi thời gian “đổi lịch học nhóm từ 7h đến 9h sáng mai sang 8h đến 10h sáng mai”.

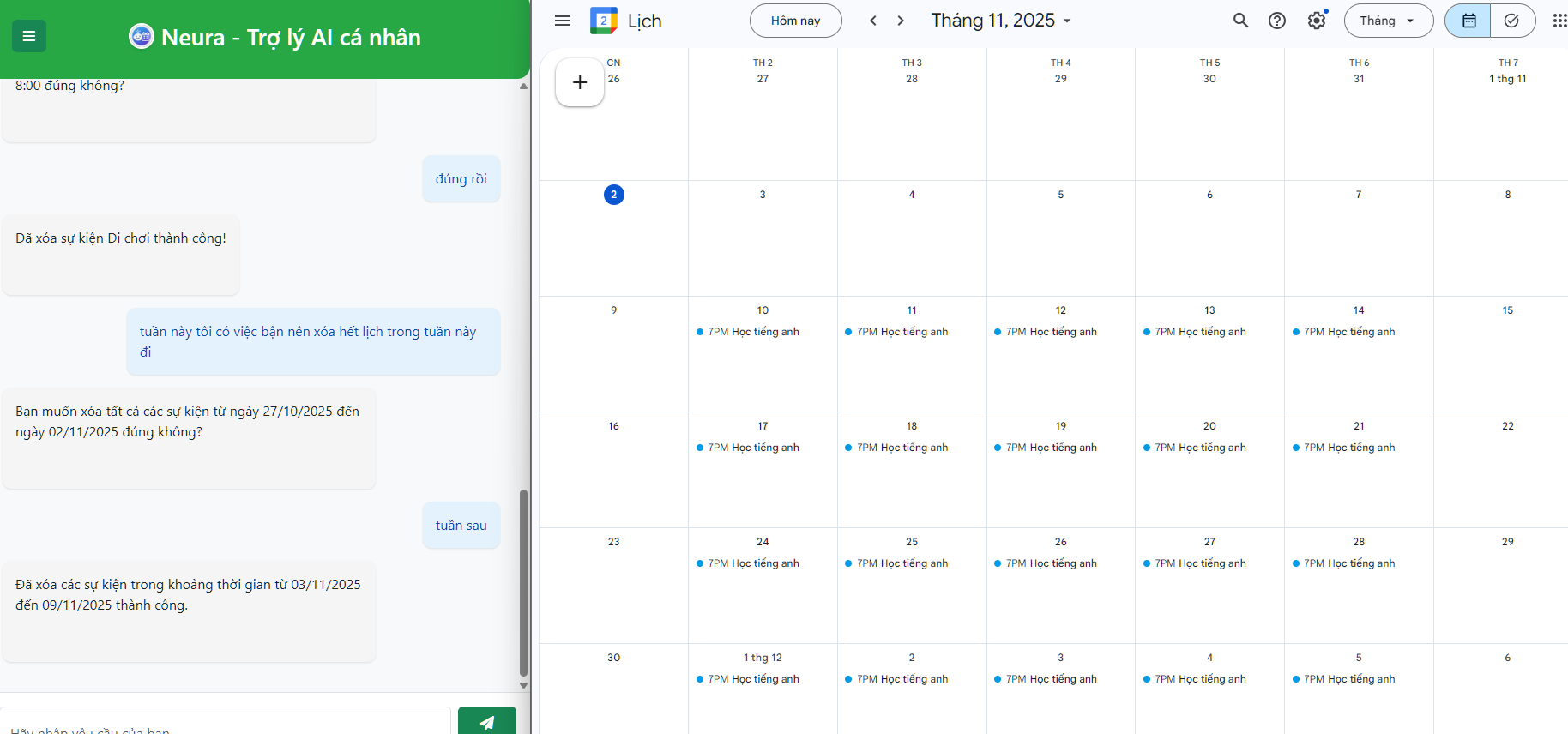
Hình 3.. Kết quả đổi thời gian sự kiện thành công

Sau đó thử đổi tiếp tiêu đề sự kiện “đổi lịch học nhóm từ 8h đến 10h sáng mai thành đi chơi vào lúc 8h đến 10h sáng mai”

Hình 3.. Kết quả đổi tiêu đề sự kiện thành công

Tiếp theo ta xóa sự kiện đã tạo đó đi “xóa lịch đi chơi lúc 8h ngày mai đi”. Kết quả chatbot đã xóa thành công

Hình 3.. Kết quả xóa sự kiện thành công

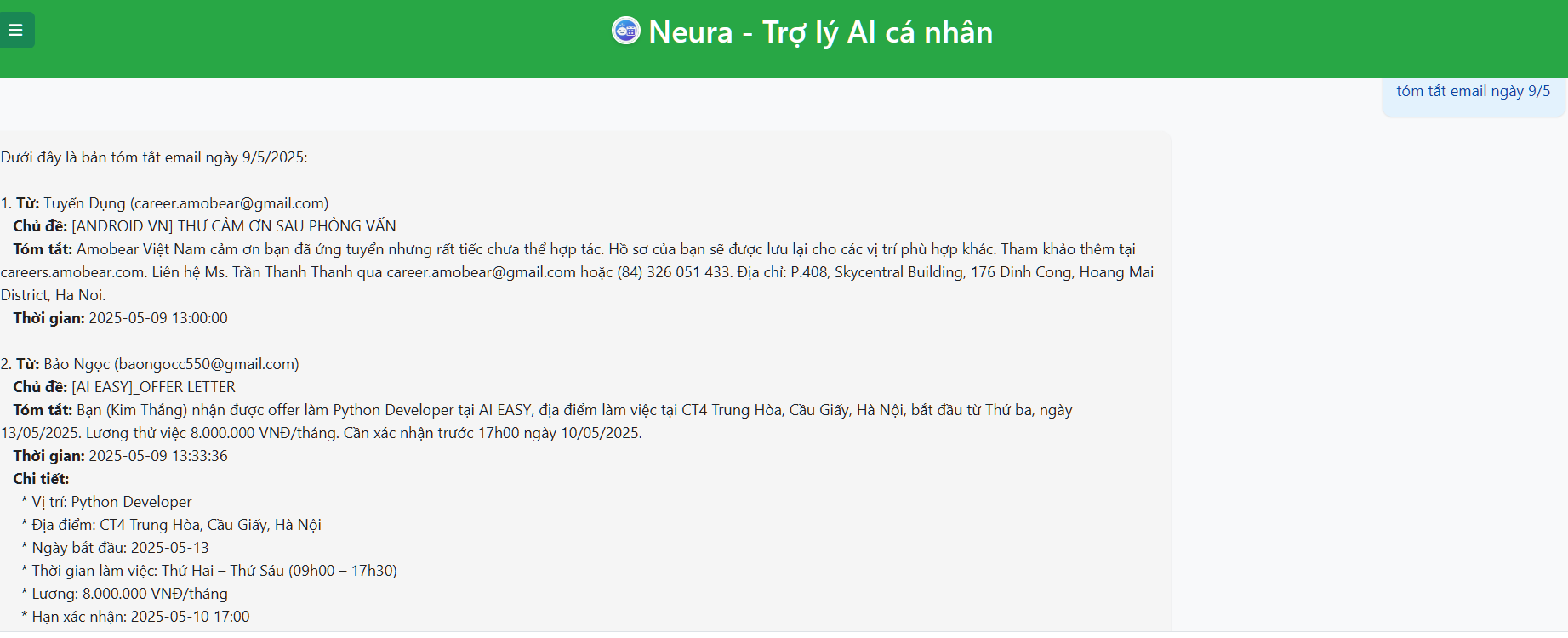
Ngoài xóa một sự kiện ta cũng có thể xóa nhiều sự kiện trong một khoảng thời gian ví dụ “tuần này tôi có việc bận nên xóa hết lịch trong tuần này đi”. Kết quả chatbot đã xóa được hết các lịch trong tuần

Hình 3.. Kết quả xóa nhiều sự kiện thành công

Dựa trên việc thử nghiệm, có thể thấy chatbot có khả năng tạo, đọc, sửa và xóa (CRUD) các sự kiện trong lịch hoạt động ổn định và chính xác. Các thao tác được xử lý đúng, phản hồi từ hệ thống rõ ràng. Ngoài ra, chatbot còn có thể nhận diện thời gian, ngày tháng, tiêu đề và nội dung sự kiện từ ngôn ngữ tự nhiên, giúp người dùng thao tác dễ dàng mà không cần nhập liệu theo định dạng cố định. Việc đồng bộ dữ liệu với Google Calendar diễn ra trơn tru, đảm bảo tính nhất quán giữa các thay đổi được thực hiện qua chatbot và trên nền tảng Google. Nhìn chung, kết quả thử nghiệm cho thấy tính năng quản lý lịch của chatbot đã đạt mức hoàn thiện tốt.

### 3.8.2. Tác vụ liên quan đến mail

Tác vụ liên quan đến mail sẽ có những chức năng chính như tóm tắt email với khoảng thời gian cho trước, tìm thông tin, nội dung chứa trong email. Ta thử tóm tắt email vào một ngày cụ thể như “tóm tắt email ngày 9/5”



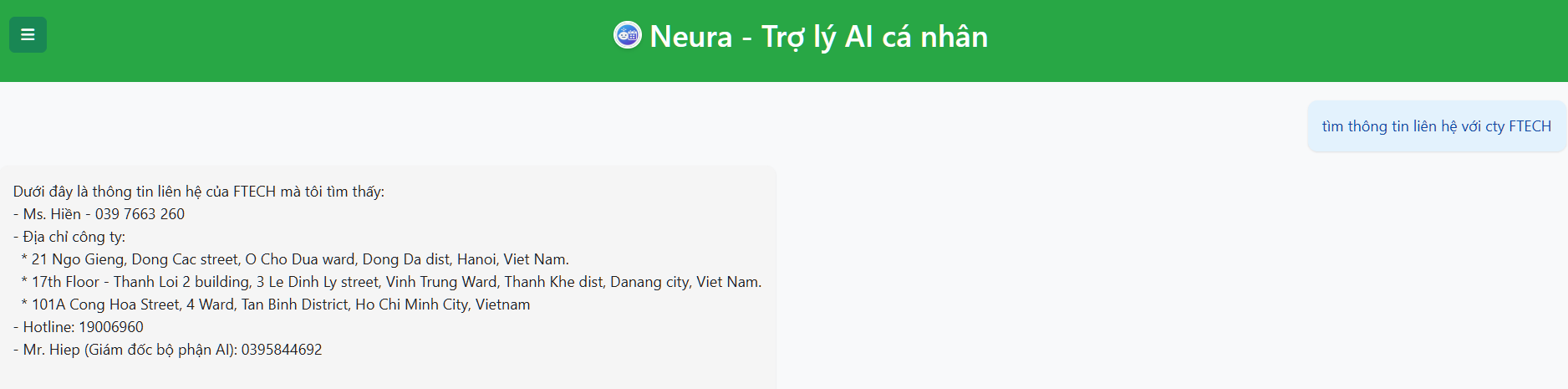
Hình 3.. Kết quả chạy Neura khi tóm tắt email vào một thời gian cụ thể

Chatbot hiểu được yêu cầu và đưa ra được bản tóm tắt của ngày 9/5, bản tóm tắt đã bao gồm được hết những thông tin quan trọng của các email. Một ví dụ khác “tóm tắt email từ ngày 10/9 đến 12/9”. Chatbot cũng đã lấy được email và đưa ra bản tóm tắt cho người dùng.



Hình 3.. Kết quả chạy Neura khi tóm tắt email vào một khoảng thời gian

Tiếp theo là khả năng tìm và lấy thông tin từ các email như “tìm thông tin liên hệ với cty Ftech”. Chatbot đã dựa vào tri thức của mình đã tìm ra được thông tin liên hệ với công ty Ftech.



Hình 3.. Kết quả chạy Neura khi tìm và truy xuất thông tin từ nội dung email

A screenshot of a computer

AI-generated content may be incorrect.Một số ví dụ khác như “từ ngày 6/9 đến ngày 12/9 có mail phỏng vấn nào không?”. Tuy từ 6/9 đến 12/9 có nhiều mail khác nhưng không liên quan đến phỏng vấn, chatbot đã loại bỏ và không đưa ra cho người dùng mà chỉ đưa ra thông tin đúng như người dùng hỏi.

Hình 3.. Kết quả chạy Neura khi tìm bản tóm tắt email về một nội dung cụ thể

A screenshot of a computer

AI-generated content may be incorrect.Những trường hợp trên cho thấy chatbot xử lý tốt các tác vụ liên quan đến mail, đáp ứng được các yêu cầu, chức năng đề ra Điều này chứng tỏ hệ thống không chỉ dừng lại ở việc đọc hiểu nội dung email mà còn có khả năng phân tích ngữ nghĩa, nhận diện chính xác mục đích truy vấn của người dùng. Chatbot có thể lọc và xử lý thông tin theo ngữ cảnh, giúp kết quả trả về mang tính chọn lọc, phù hợp và sát với nhu cầu thực tế. Bên cạnh đó, khả năng tương tác tự nhiên của chatbot cũng được thể hiện rõ ràng. Người dùng có thể sử dụng ngôn ngữ linh hoạt, không cần tuân theo cú pháp cứng nhắc, mà hệ thống vẫn hiểu đúng và phản hồi chính xác. Điều này giúp trải nghiệm người dùng trở nên thân thiện và gần gũi hơn.

Hình 3.. Danh sách email từ 6/9 đến 12/9

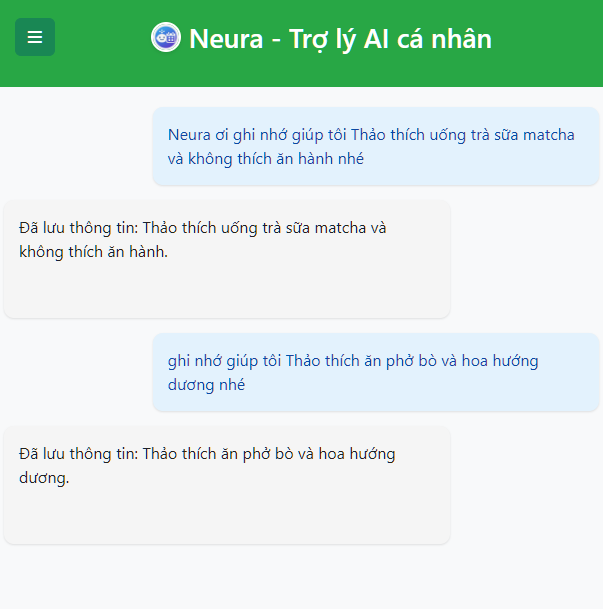
Ngoài hai chức năng chính là tóm tắt email và tìm kiếm thông tin trong email, hệ thống còn có tiềm năng mở rộng thêm các tính năng nâng cao như:

* Tự động phân loại email theo chủ đề (công việc, cá nhân, thông báo, deadline, v.v.).
* Phát hiện thời gian và sự kiện trong nội dung thư để gợi ý tạo lịch trên Google Calendar (đã có thể nhưng chưa ổn định).
* Tổng hợp báo cáo định kỳ về số lượng email nhận được, tỷ lệ phản hồi, hoặc các chủ đề thường xuất hiện.

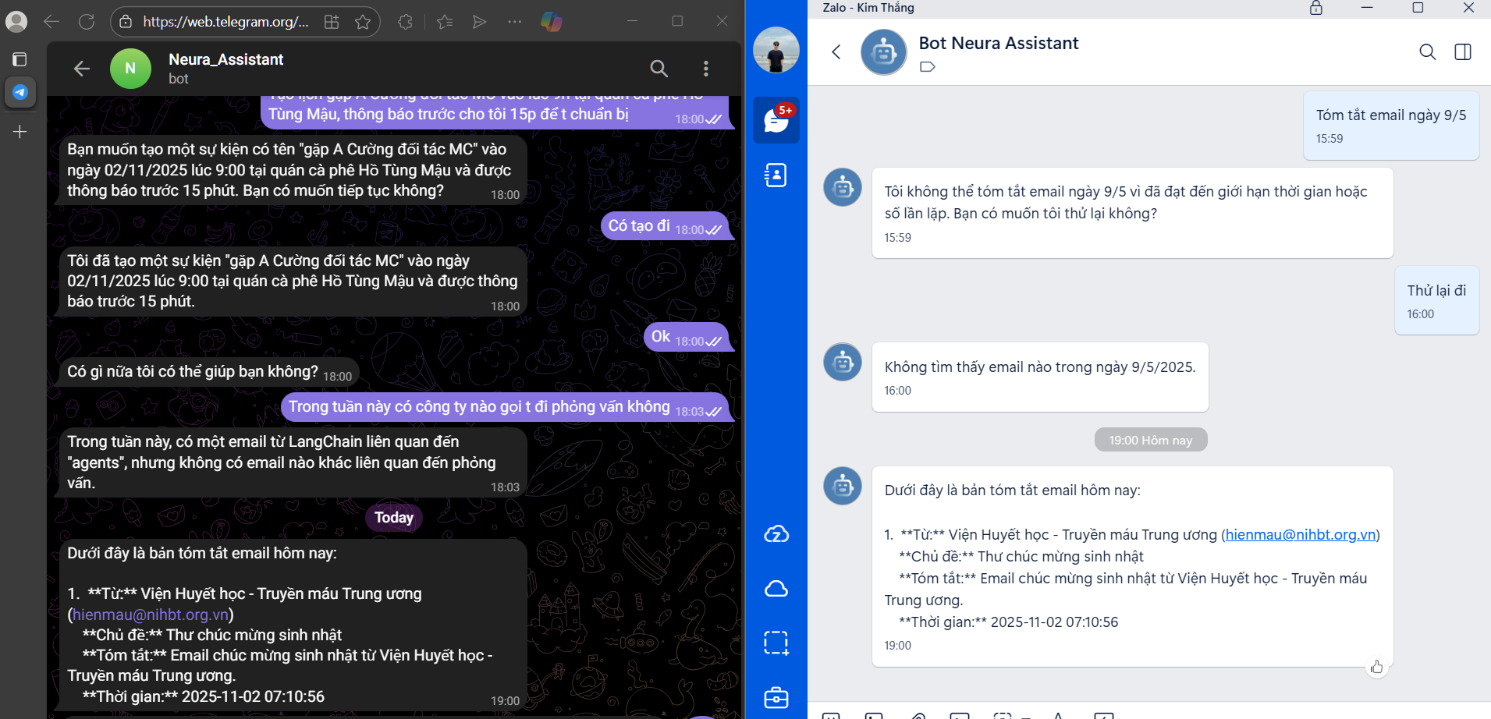
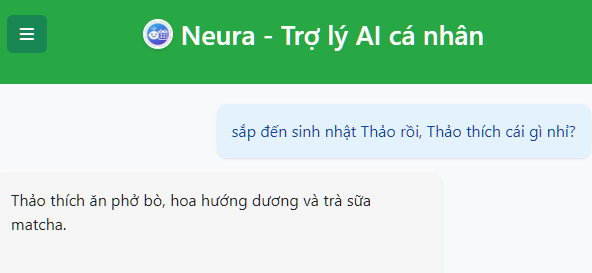
Từ đó, có thể khẳng định rằng chatbot đã đáp ứng tốt các yêu cầu đặt ra về mặt chức năng, khả năng xử lý ngôn ngữ tự nhiên, và độ chính xác trong việc truy xuất thông tin. Hệ thống hoạt động ổn định, thông minh, và có tính ứng dụng cao trong thực tế đặc biệt trong việc hỗ trợ người dùng quản lý email hiệu quả hơn.

### 3.8.3. Tác vụ khác

Ngoài các tác vụ liên quan đến lịch và email, chatbot có thể ghi nhớ và truy xuất thông tin đã ghi nhớ, tạo các tác vụ tự động hàng ngày, tuần, …… rồi gửi kết quả tới Bot( Telegram, Zalo).

Ví dụ ta có thể cung cấp thông tin cho chatbot ghi nhớ, sau khi cung cấp chatbot sẽ tiến hành ghi nhớ vào cơ sở dữ liệu khi cần người dùng chỉ cần hỏi đến chatbot có thể đưa ra được thông tin.

Hình 3.. Cung cấp thông tin cần ghi nhớ cho chatbot

Ngoài ra ta còn có thể tạo một nhiệm vụ tự động để chatbot có thể làm ví dụ như “Tạo lịch tóm tắt email hàng ngày tự động vào lúc 18h40”. Chatbot sẽ tiến hành trích xuất thời gian và yêu cầu nhiệm vụ sau đó sẽ lưu lại. Hàng ngày cứ đến 18h40 hệ thống sẽ tiến hành gọi hàm thực thi agent manager sau đó tiến hành thực hiện nhiệm vụ, kết quả sau đó sẽ được thông báo về bot(telegram, zalo).

Hình 3.. Thông tin chatbot đưa ra khi được hỏi đến dữ liệu đã lưu

Hình 3.. Kết quả nhiệm vụ tự động được gửi về bot Telegram và Zalo

# **KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

Đồ án “Xây dựng trợ lý AI cá nhân – Neura” được thực hiện với mục tiêu phát triển một hệ thống trợ lý ảo có khả năng tương tác tự nhiên bằng ngôn ngữ con người, hỗ trợ người dùng trong việc quản lý lịch làm việc, xử lý email và ghi nhớ thông tin cá nhân. Sau quá trình nghiên cứu và triển khai, hệ thống đã đạt được nhiều kết quả tích cực.

Cụ thể, Neura đã được xây dựng với khả năng hiểu và phản hồi ngôn ngữ tự nhiên, sử dụng mô hình ngôn ngữ lớn (LLM) để xử lý hội thoại, giúp việc giao tiếp giữa người dùng và hệ thống trở nên linh hoạt và tự nhiên hơn. Hệ thống đã tích hợp thành công Google Calendar API, cho phép thực hiện đầy đủ các thao tác như tạo, đọc, cập nhật và xóa sự kiện, đồng thời có thể truy xuất thông tin sự kiện, kiểm tra thời gian rảnh và tạo lịch trực tiếp bằng ngôn ngữ tự nhiên. Bên cạnh đó, Neura còn sử dụng Google Gmail API để lấy và xử lý email. Thông qua LLM, hệ thống có thể tóm tắt nội dung email, giúp người dùng nhanh chóng nắm bắt thông tin quan trọng, đồng thời cho phép lưu trữ và truy xuất email bằng ngôn ngữ tự nhiên, giúp việc tìm kiếm trở nên thuận tiện hơn. Ngoài ra, kỹ thuật RAG (Retrieval-Augmented Generation) cũng được áp dụng để giúp hệ thống truy xuất và sử dụng lại các thông tin đã được lưu trữ hoặc lấy từ email, góp phần giảm thiểu hiện tượng “ảo giác” của mô hình ngôn ngữ và nâng cao độ chính xác trong phản hồi. Một điểm nổi bật khác của Neura là khả năng tự động hóa nhiệm vụ. Hệ thống có thể hiểu lệnh của người dùng như “Tạo lịch tóm tắt email hàng ngày tự động lúc 19h00”, sau đó tự động trích xuất thời gian, lưu nhiệm vụ và đến đúng thời điểm sẽ thực thi tác vụ, gửi kết quả tóm tắt email về bot Telegram hoặc Zalo. Nhờ đó, Neura không chỉ phản hồi theo yêu cầu mà còn có thể hoạt động chủ động, hỗ trợ người dùng một cách tự động và liên tục. Cuối cùng, hệ thống đã được triển khai đa nền tảng, bao gồm giao diện web (xây dựng bằng Flask), bot Telegram và bot Zalo, tất cả đều vận hành ổn định dựa trên cơ chế polling.

Kết quả đạt được cho thấy việc kết hợp giữa mô hình ngôn ngữ lớn (LLM), kỹ thuật RAG và các API thực tế hoàn toàn khả thi trong việc tạo nên một trợ lý ảo có tính ứng dụng cao. Neura không chỉ dừng lại ở việc trả lời câu hỏi, mà còn có khả năng hỗ trợ quản lý công việc cá nhân một cách thông minh và tự động hơn. Quá trình triển khai cũng giúp người thực hiện củng cố kiến thức về lập trình, xử lý ngôn ngữ tự nhiên, tích hợp API và phát triển ứng dụng thực tế.

Tuy nhiên chatbot vẫn còn một số hạn chế nhất định. Khả năng ghi nhớ và cá nhân hóa của Neura còn ở mức cơ bản, các thông tin khi truy xuất vẫn thừa thêm các thông tin khác khi truy xuất. Ngoài ra thời gian xử lý của chatbot phản hồi lâu vì phải qua xử lý của Multi-agent đó cũng là một số giới hạn trong việc sử dụng thư viện Langchain. Mặc dù LangChain hỗ trợ tốt cho việc kết nối mô hình ngôn ngữ lớn (LLM) với các công cụ và nguồn dữ liệu bên ngoài, nhưng khi hệ thống ngày càng phức tạp, việc quản lý luồng xử lý, trạng thái hội thoại và các nhánh logic trở nên khó kiểm soát. Vì vậy, hướng phát triển tiếp theo của dự án là chuyển từ Langchain sang LangGraph – một kiến trúc mới dựa trên đồ thị luồng (graph-based workflow). LangGraph cho phép định nghĩa rõ ràng các node, edge và trạng thái hội thoại, giúp hệ thống trở nên trực quan, dễ mở rộng và dễ kiểm soát hơn. Điều này không chỉ giúp tăng tính ổn định và khả năng mở rộng, mà còn phù hợp với định hướng phát triển Neura trở thành một trợ lý AI đa tác vụ, hoạt động theo mô hình agent hợp tác. Khi đó việc tích hợp thêm các dịch vụ khác như Google Workspace trở nên dễ dàng hơn, đồng thời áp dụng các kĩ thuật RAG nâng cao để tăng độ chính xác khi truy xuất thông tin hơn.

# **TÀI LIỆU THAM KHẢO**

[1] “Doanh nghiệp công nghệ số hãy nhận lấy sứ mệnh dùng chuyển đổi AI để thay đổi Việt Nam”, https://mst.gov.vn/doanh-nghiep-cong-nghe-so-hay-nhan-lay-su-menh-dung-chuyen-doi-ai-de-thay-doi-viet-nam-197240915191157566.htm

[2] J. Singh, \*Natural Language Processing in the Real World\*, CRC Press, 2023.

[3] “LLM (Mô hình ngôn ngữ lớn) là gì?”, <https://aws.amazon.com/vi/what->is/large-language-model/

[4] Job Scheduling in Python with APScheduler | Better Stack Community, <https://betterstack.com/community/guides/scaling-python/apscheduler-scheduled-tasks/>, [17/08]

[5] “Day 1: Introduction to Retrieval Augmented Generation“,<https://bit.ly/4lgxkkk>

[6] “Exploring LangChain”, <https://bit.ly/4lgvWy4>

[7] ‘Flask – (Creating first simple application)’, <https://www.geeksforgeeks.org/flask-creating-first-simple-application/>

[8] Hướng dẫn xây dựng Zalo Bot đơn giản với cơ chế Polling, https://bot.zapps.me/docs/build-your-bot/