ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Hệ thống Đa Tác Tử Hỗ trợ Phát triển Giao Diện Web

LÊ NGỌC VĂN
vanln.240823e@sis.hust.edu.vn

Chương trình đào tạo: Kỹ sư chuyên sâu Trí tuệ nhân tạo tạo sinh

Giảng viên hướng dẫn: TS. Nguyễn Đức Anh _____

Chữ kí GVHD

Khoa: Khoa học máy tính

Trường: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 01/2026

ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Hệ thống Đa Tác Tử Hỗ trợ Phát triển Giao Diện Web

LÊ NGỌC VĂN
vanln.240823e@sis.hust.edu.vn

Chương trình đào tạo: Kỹ sư chuyên sâu Trí tuệ nhân tạo tạo sinh

Giảng viên hướng dẫn: TS. Nguyễn Đức Anh _____

Chữ kí GVHD

Khoa: Khoa học máy tính

Trường: Công nghệ Thông tin và Truyền thông

HÀ NỘI, 01/2026

LỜI CẢM ƠN

Trước hết em xin bày tỏ lòng biết ơn sâu sắc nhất đến giảng viên hướng dẫn Tiến sĩ Nguyễn Đức Anh đã tận tình giúp đỡ, hướng dẫn em rất nhiều trong suốt quá trình học tập và hoàn thành đồ án tốt nghiệp.

Em xin chân thành cảm ơn các thầy cô Đại học Bách Khoa Hà Nội, Trường Công nghệ thông tin và Truyền thông đã giảng dạy, trang bị cho em những kiến thức cần thiết để em có thể hoàn thành tốt đồ án tốt nghiệp này. Xin gửi lời cảm ơn đến gia đình và bạn bè, những người luôn bên em động viên và tạo điều kiện thuận lợi, tận tình giúp đỡ em trong quá trình hoàn thiện đồ án tốt nghiệp.

Do thời gian có hạn, trình độ hiểu biết của bản thân còn nhiều hạn chế nên trong đồ án không tránh khỏi những thiếu sót, em rất mong nhận được sự đóng góp ý kiến của các thầy cô giáo để em có thể cải thiện những điểm còn thiếu sót, trang bị thêm kiến thức để áp dụng vào công việc thực tế trong tương lai.

Em xin chân thành cảm ơn!

TÓM TẮT NỘI DUNG ĐỒ ÁN

Trong những năm gần đây, nhu cầu phát triển giao diện web ngày càng gia tăng cùng với sự đa dạng về thiết bị, người dùng và yêu cầu trải nghiệm. Tuy nhiên, quá trình phát triển giao diện web hiện nay vẫn còn tồn tại nhiều vấn đề như tốn nhiều thời gian, phụ thuộc lớn vào kinh nghiệm cá nhân của lập trình viên, khó đảm bảo tính nhất quán giữa thiết kế và triển khai, cũng như gặp nhiều hạn chế trong việc tối ưu trải nghiệm người dùng. Các hướng tiếp cận phổ biến hiện nay bao gồm sử dụng framework giao diện, thư viện UI/UX, công cụ thiết kế kéo-thả hoặc áp dụng trí tuệ nhân tạo để sinh mã tự động. Mặc dù các hướng này đã phần nào hỗ trợ quá trình phát triển, chúng vẫn còn hạn chế ở khả năng thích nghi linh hoạt, thiếu sự phối hợp thông minh giữa các khâu phân tích yêu cầu, thiết kế và hiện thực hóa giao diện.

Xuất phát từ thực tế đó, đồ án này lựa chọn hướng tiếp cận xây dựng hệ thống đa tác tử (multi-agent system) nhằm hỗ trợ phát triển giao diện web một cách thông minh và tự động hơn. Hướng tiếp cận này được lựa chọn vì khả năng phân tách nhiệm vụ, phối hợp linh hoạt giữa các tác tử chuyên biệt, cũng như khả năng mở rộng và thích nghi tốt với các yêu cầu thay đổi trong quá trình phát triển phần mềm.

Theo hướng tiếp cận đã chọn, đồ án đề xuất một hệ thống gồm nhiều tác tử đảm nhiệm các vai trò khác nhau như phân tích yêu cầu người dùng, đề xuất cấu trúc giao diện, sinh mã giao diện và đánh giá mức độ phù hợp về trải nghiệm người dùng. Các tác tử này tương tác với nhau thông qua cơ chế trao đổi thông tin để tạo ra giao diện web hoàn chỉnh, nhất quán và tối ưu.

Đóng góp chính của đồ án là xây dựng được một mô hình hệ thống đa tác tử áp dụng vào lĩnh vực phát triển giao diện web, đồng thời triển khai thành công một hệ thống thử nghiệm cho thấy khả năng hỗ trợ hiệu quả quá trình thiết kế và xây dựng giao diện. Kết quả đạt được cho thấy hệ thống giúp giảm thời gian phát triển, nâng cao tính nhất quán và cải thiện chất lượng giao diện web so với các phương pháp truyền thống.

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

ABSTRACT

In recent years, the demand for web interface development has increased significantly along with the diversity of devices, users, and user experience requirements. However, the current process of web interface development still faces several challenges, such as being time-consuming, heavily dependent on individual developer experience, difficult to ensure consistency between design and implementation, and encountering limitations in optimizing user experience. Common existing approaches include the use of UI frameworks, UI/UX libraries, drag-and-drop design tools, or the application of artificial intelligence for automatic code generation. Although these approaches have partially supported the development process, they still suffer from limitations in adaptability and lack intelligent coordination among requirement analysis, interface design, and implementation stages.

From this reality, this thesis chooses a multi-agent system approach to support web interface development in a more intelligent and automated manner. This approach is selected due to its ability to decompose tasks, enable flexible collaboration among specialized agents, and provide strong scalability and adaptability to changing requirements during the software development process.

Based on the selected approach, this thesis proposes a system consisting of multiple agents responsible for different roles, such as analyzing user requirements, recommending interface structures, generating interface code, and evaluating user experience suitability. These agents interact with each other through information exchange mechanisms to produce a complete, consistent, and optimized web interface.

The main contribution of this thesis is the construction of a multi-agent system model applied to the field of web interface development, along with the successful implementation of a prototype system that demonstrates effective support for the interface design and development process. The final results show that the proposed system helps reduce development time, enhance consistency, and improve the overall quality of web interfaces compared to traditional methods.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

Viết tắt	Tên tiếng Anh	Tên tiếng Việt
API	Application Programming Interface	Giao diện lập trình ứng dụng
EUD	End-User Development	Phát triển ứng dụng cho người dùng cuối
HTML	HyperText Markup Language	Ngôn ngữ đánh dấu siêu văn bản
UI	User Interface	Giao diện người dùng
UX	User Experience	Trải nghiệm người dùng
UI Sketch	User Interface Sketch	Phác thảo giao diện người dùng
Agent	Software Agent	Tác tử phần mềm, thực thể độc lập thực hiện một nhiệm vụ cụ thể
Multi-Agent	Multi-Agent System	Hệ thống đa tác tử
BA Agent	Business Analyst Agent	Tác tử phân tích nghiệp vụ và yêu cầu người dùng
Prompt	Prompt Instruction	Câu lệnh đầu vào dùng để định hướng quá trình sinh nội dung
Context	Execution Context	Ngữ cảnh thực thi, lưu trữ thông tin xuyên suốt quy trình xử lý
Deploy	Application Deployment	Quá trình triển khai ứng dụng để đưa vào sử dụng
CNTT	Information Technology	Công nghệ thông tin
ĐATN	Graduation Thesis	Đồ án tốt nghiệp
SV	Student	Sinh viên

Thuật ngữ	Ý nghĩa
Browser	Trình duyệt web, môi trường chạy và hiển thị giao diện người dùng
Cache Memory	Bộ nhớ đệm giúp lưu trữ tạm thời dữ liệu nhằm tăng tốc độ tải và phản hồi giao diện
Frontend	Phần giao diện phía người dùng, bao gồm bố cục, màu sắc và các thành phần tương tác
Backend	Phần xử lý logic và dữ liệu phía máy chủ, hỗ trợ hoạt động của hệ thống
Interpreter	Trình thông dịch, thực thi mã nguồn theo từng dòng trong quá trình chạy
Compiler	Trình biên dịch, chuyển đổi mã nguồn sang mã máy hoặc mã trung gian trước khi thực thi
Responsive Design	Thiết kế giao diện có khả năng thích ứng với nhiều kích thước màn hình khác nhau
Web Component	Thành phần giao diện độc lập, có thể tái sử dụng trong nhiều màn hình khác nhau

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Trong bối cảnh chuyển đổi số diễn ra mạnh mẽ, các ứng dụng và hệ thống dựa trên nền tảng web ngày càng đóng vai trò quan trọng trong hầu hết các lĩnh vực như thương mại điện tử, giáo dục, tài chính, y tế và dịch vụ công. Giao diện web không chỉ là kênh tương tác trực tiếp giữa hệ thống và người dùng mà còn ảnh hưởng lớn đến hiệu quả sử dụng, mức độ hài lòng và khả năng tiếp cận của người dùng. Do đó, yêu cầu đối với việc phát triển giao diện web ngày càng trở nên khắt khe, đòi hỏi vừa đảm bảo tính thẩm mỹ, tính nhất quán, vừa đáp ứng tốt các tiêu chí về trải nghiệm người dùng và khả năng mở rộng.

Tuy nhiên, thực tế cho thấy quá trình phát triển giao diện web hiện nay vẫn gặp nhiều khó khăn và thách thức. Việc xây dựng giao diện thường đòi hỏi sự phối hợp chặt chẽ giữa nhiều vai trò khác nhau như phân tích yêu cầu, thiết kế UI/UX và lập trình frontend. Sự thiếu đồng bộ hoặc sai lệch giữa các khâu này có thể dẫn đến việc giao diện không đáp ứng đúng nhu cầu người dùng, mất nhiều thời gian chỉnh sửa và làm gia tăng chi phí phát triển. Ngoài ra, với sự đa dạng về thiết bị, độ phân giải và hành vi người dùng, việc đảm bảo giao diện hoạt động hiệu quả và nhất quán trên nhiều nền tảng càng làm cho bài toán trở nên phức tạp hơn.

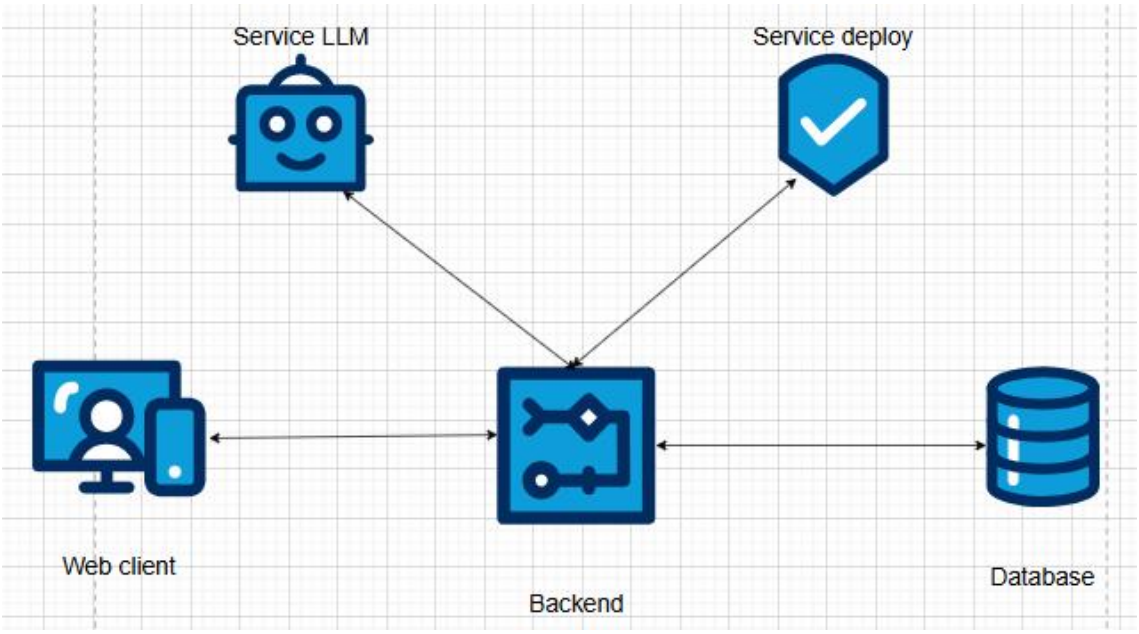
Nếu bài toán hỗ trợ hiệu quả quá trình phát triển giao diện web được giải quyết, sẽ mang lại nhiều lợi ích thiết thực cho các nhóm đối tượng khác nhau, bao gồm lập trình viên, nhà thiết kế, doanh nghiệp và người dùng cuối. Việc giảm thiểu thời gian phát triển, nâng cao chất lượng và tính nhất quán của giao diện sẽ góp phần cải thiện hiệu suất làm việc, tối ưu chi phí và nâng cao trải nghiệm người dùng. Không chỉ dừng lại ở lĩnh vực phát triển web, các vấn đề liên quan đến phối hợp tác vụ, tự động hóa và hỗ trợ ra quyết định trong quá trình thiết kế và triển khai giao diện còn có tiềm năng được mở rộng và áp dụng sang các lĩnh vực khác như phát triển phần mềm nói chung, thiết kế hệ thống tương tác người–máy và các ứng dụng số thông minh.

1.2 Mục tiêu và phạm vi đề tài

Hiện nay, đã có nhiều công cụ, framework và nền tảng hỗ trợ phát triển giao diện web được sử dụng rộng rãi trong thực tế. Các công cụ này giúp lập trình viên và nhà thiết kế xây dựng giao diện nhanh hơn, đảm bảo tính thẩm mỹ và khả năng tương thích trên nhiều thiết bị. Tuy nhiên, các sản phẩm hiện tại vẫn tồn tại nhiều hạn chế. Phần lớn các công cụ chỉ tập trung hỗ trợ từng khâu riêng lẻ như thiết kế giao diện, xây dựng bố cục hoặc sinh mã giao diện, trong khi thiếu sự phối hợp tổng thể giữa các bước phân tích yêu cầu, thiết kế và triển khai. Ngoài ra, việc phát triển giao diện web vẫn phụ thuộc nhiều vào thao tác thủ công và kinh nghiệm cá nhân, khiến quá trình phát triển tốn thời gian, khó đảm bảo tính nhất quán và gặp khó khăn khi yêu cầu thay đổi.

Nhận thấy những hạn chế nêu trên, em hướng tới việc nghiên cứu và xây dựng một hệ thống hỗ trợ phát triển giao diện web theo hướng tự động hóa và phối hợp thông minh hơn. Mục tiêu chính của đề tài bao gồm: hỗ trợ phân tích yêu cầu giao diện từ người dùng hoặc tài liệu đầu vào, hỗ trợ đề xuất cấu trúc và thành phần giao diện phù hợp với yêu cầu, và hỗ trợ quá trình hiện thực hóa giao diện web nhằm giảm bớt khối lượng công việc thủ công cho người phát triển. Thông qua đó, hệ thống hướng đến việc nâng cao hiệu quả phát triển, đảm bảo tính nhất quán và cải thiện chất lượng giao diện web.

Để đáp ứng các mục tiêu này, đề tài tập trung nghiên cứu và xây dựng một hệ thống hỗ trợ phát triển giao diện web dựa trên mô hình đa tác tử. Hệ thống được thiết kế nhằm cho phép các thành phần chức năng hoạt động tương đối độc lập nhưng vẫn có khả năng phối hợp với nhau trong quá trình hỗ trợ phát triển giao diện. Cách tiếp cận này giúp hệ thống có khả năng thích nghi tốt hơn với các yêu cầu khác nhau, đồng thời tạo tiền đề cho việc mở rộng và tích hợp với các công cụ phát triển web hiện có. Bên cạnh đó, hệ thống hướng tới việc cung cấp một giao diện thân thiện, giúp người dùng dễ dàng tương tác và theo dõi quá trình hỗ trợ phát triển giao diện.



Hình 1.1: Sơ đồ tổng quan hệ thống

Mục tiêu của đề tài là xây dựng một hệ thống hỗ trợ phát triển giao diện web MAS4UI dựa trên mô hình đa tác tử, nhằm hỗ trợ các bước chính trong quá trình phát triển giao diện, góp phần giảm thời gian phát triển, nâng cao tính nhất quán và cải thiện chất lượng giao diện web trong các ứng dụng thực tế.

1.3 Định hướng giải pháp

Từ việc xác định rõ nhiệm vụ cần giải quyết ở phần 1.2, đồ án lựa chọn giải quyết bài toán hỗ trợ phát triển giao diện web theo định hướng xây dựng hệ thống đa tác tử (Multi-Agent System) kết hợp với các mô hình ngôn ngữ lớn (LLM) và các công nghệ phát triển web hiện đại. Cụ thể, hệ thống tận dụng API của các mô hình trí tuệ nhân tạo như GPT, Claude và Gemini để hỗ trợ phân tích yêu cầu, đề xuất giao diện và sinh mã giao diện. Các công nghệ web như Next.js cho frontend và Node.js cho backend được sử dụng nhằm xây dựng một nền tảng web linh hoạt, dễ mở rộng và thuận tiện cho người dùng.

Theo định hướng trên, giải pháp của đồ án là xây dựng một hệ thống gồm nhiều tác tử với vai trò chuyên biệt, phối hợp với nhau thông qua một tác tử điều phối trung tâm. Tác tử Orchestrator đóng vai trò trung gian, chịu trách nhiệm điều phối luồng xử lý và giao tiếp giữa các tác tử khác. Tác tử Business Analyst (BA) thực hiện việc thu thập và làm rõ yêu cầu từ khách hàng thông qua tương tác trực tiếp, đảm bảo thông tin đầu vào đầy đủ và nhất quán. Sau đó, các yêu cầu này được chuyển cho tác tử SketchUI, có nhiệm vụ tạo ra bản phác thảo tổng quan giao diện người dùng. Bản phác thảo này được gửi lại cho khách hàng để xem xét và chỉnh sửa nếu cần. Khi giao diện tổng quan đã được thống nhất, tác tử GenCode tiến hành sinh mã giao diện web dựa trên thiết kế đã được phê duyệt. Cuối cùng, tác tử Deploy đảm nhiệm việc triển khai ứng dụng lên môi trường chạy thử hoặc môi trường trực tuyến, đồng thời cung cấp đường dẫn truy cập cho khách hàng.

Đóng góp chính của đồ án là đề xuất và hiện thực một mô hình hệ thống đa tác tử áp dụng cho bài toán hỗ trợ phát triển giao diện web, trong đó các tác tử được tổ chức theo một quy trình rõ ràng từ thu thập yêu cầu đến triển khai sản phẩm. Kết quả đạt được là một hệ thống thử nghiệm cho phép người dùng nhận được giao diện web hoàn chỉnh thông qua một quy trình bán tự động, góp phần giảm thời gian phát triển, tăng mức độ nhất quán và nâng cao hiệu quả phối hợp giữa các bước trong quá trình phát triển giao diện web.

1.4 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Chương 2 tập trung vào việc khảo sát và phân tích yêu cầu của hệ thống. Nội dung chương bao gồm

khảo sát hiện trạng các công cụ và giải pháp hỗ trợ phát triển giao diện web hiện nay, tổng quan các chức năng của hệ thống thông qua các biểu đồ use case, quy trình nghiệp vụ và đặc tả chi tiết các chức năng chính. Bên cạnh đó, chương này cũng xác định các yêu cầu phi chức năng nhằm làm rõ các tiêu chí về hiệu năng, khả năng mở rộng và tính sử dụng của hệ thống. Các kết quả phân tích trong chương này là cơ sở cho việc thiết kế và triển khai hệ thống ở các chương sau.

Chương 3 trình bày các công nghệ được sử dụng trong đồ án. Chương này giới thiệu tổng quan các công nghệ, nền tảng và công cụ chính được lựa chọn, bao gồm các mô hình ngôn ngữ lớn, công nghệ phát triển web và các công nghệ hỗ trợ triển khai hệ thống. Nội dung chương nhằm cung cấp nền tảng kỹ thuật phục vụ cho việc thiết kế và hiện thực hệ thống đa tác tử.

Chương 4 tập trung vào thiết kế, triển khai và đánh giá hệ thống. Trong chương này, đồ án trình bày thiết kế kiến trúc tổng thể của hệ thống, lựa chọn kiến trúc phần mềm, thiết kế tổng quan và thiết kế chi tiết các thành phần. Tiếp theo, chương mô tả quá trình xây dựng ứng dụng, các thư viện và công cụ sử dụng, kết quả đạt được và minh họa các chức năng chính của hệ thống. Cuối cùng, các nội dung về kiểm thử và triển khai hệ thống cũng được trình bày nhằm đánh giá mức độ đáp ứng yêu cầu của hệ thống.

Chương 5 trình bày các giải pháp và những đóng góp nổi bật của đồ án. Chương này tổng hợp và phân tích các điểm mới, giá trị và hiệu quả mà hệ thống mang lại so với các giải pháp hiện có, đồng thời làm rõ những cải tiến đạt được trong quá trình nghiên cứu và triển khai.

Chương 6 đưa ra kết luận và hướng phát triển trong tương lai. Chương này tổng kết các kết quả chính của đồ án, đánh giá mức độ hoàn thành mục tiêu đề ra, đồng thời đề xuất các hướng phát triển và mở rộng hệ thống nhằm nâng cao tính ứng dụng và hiệu quả trong thực tế.

Sau các chương nội dung chính, phần tài liệu tham khảo và phụ lục được trình bày nhằm cung cấp các nguồn tham chiếu và các tài liệu bổ trợ phục vụ cho việc nghiên cứu và hoàn thiện đồ án.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

Chương 2 của ĐATN sẽ tập trung phân tích khảo sát hiện trạng qua việc khảo sát người dùng, các hệ thống đã có, ứng dụng tương tự qua đó phân tích và xác định được các yêu cầu phần mềm cần thiết cho hệ thống hiện tại, xác định tổng quan chức năng của hệ thống bằng việc phân tích biểu đồ usecase tổng quát và phân rã biểu đồ usecase từng tác nhân, và đặc tả chức năng từng usecase quan trọng như: đăng nhập, đăng ký, đăng xuất, tạo yêu cầu mới, xem lịch sử chat, chỉnh sửa UI sketch... Đồng thời chương này cũng sẽ phân tích những yêu cầu phi chức năng của hệ thống cần đạt được.

2.1 Khảo sát hiện trạng

Để phân tích và xác định các yêu cầu phần mềm cho hệ thống web hỗ trợ phát triển ứng dụng tự động theo mô hình đa tác tử, quá trình khảo sát hiện trạng được thực hiện dựa trên ba nguồn thông tin chính, bao gồm: (i) người dùng/khách hàng, (ii) các hệ thống và công cụ phát triển phần mềm đã có, và (iii) các ứng dụng, nền tảng tương tự đang được sử dụng phổ biến trên thị trường. Việc khảo sát từ nhiều nguồn khác nhau giúp đảm bảo tính khách quan, thực tiễn và toàn diện trong việc xác định yêu cầu, đồng thời làm cơ sở cho quá trình thiết kế và triển khai hệ thống ở các chương tiếp theo.

Trước hết, khảo sát từ phía người dùng/khách hàng cho thấy nhu cầu xây dựng các ứng dụng web ngày càng gia tăng, không chỉ giới hạn trong nhóm lập trình viên chuyên nghiệp mà còn mở rộng đến các cá nhân, nhóm khởi nghiệp và doanh nghiệp nhỏ. Qua trao đổi và tìm hiểu thực tế, có thể nhận thấy quá trình phát triển phần mềm hiện nay vẫn tiêu tốn nhiều thời gian và nguồn lực, đặc biệt ở các giai đoạn thu thập yêu cầu, phân tích nghiệp vụ, thiết kế giao diện và viết mã nguồn. Đối với những người không có nền tảng kỹ thuật chuyên sâu, việc chuyển đổi ý tưởng thành một sản phẩm phần mềm hoàn chỉnh là một thách thức lớn. Do đó, người dùng mong muốn có một hệ thống cho phép mô tả yêu cầu bằng ngôn ngữ tự nhiên, được hỗ trợ phân tích nghiệp vụ, đề xuất giao diện và tự động sinh mã nguồn, từ đó rút ngắn thời gian phát triển và giảm sự phụ thuộc vào đội ngũ lập trình lớn.

Bên cạnh khảo sát người dùng, việc nghiên cứu các hệ thống và công cụ phát triển phần mềm hiện có cho thấy sự tồn tại của nhiều nền tảng hỗ trợ phát triển ứng dụng nhanh. Một số nền tảng tiêu biểu có thể kể đến như các công cụ tạo website kéo-thả, các nền tảng *low-code/no-code* và các trợ lý lập trình thông minh. Ví dụ, các nền tảng tạo website cho phép người dùng xây dựng giao diện nhanh chóng thông qua các thành phần có sẵn, với chi phí sử dụng dao động từ miễn phí đến vài chục đô la Mỹ mỗi tháng tùy theo gói dịch vụ. Các nền tảng *low-code/no-code* cung cấp khả năng xây dựng ứng dụng với mức độ lập trình tối thiểu, hỗ trợ các chức năng cơ bản như quản lý dữ liệu, xác thực người dùng và triển khai ứng dụng, tuy nhiên thường bị giới hạn về khả năng tùy biến và mở rộng hệ thống.

Ngoài ra, trên thị trường hiện nay cũng xuất hiện các trợ lý lập trình thông minh dựa trên trí tuệ nhân tạo, có khả năng hỗ trợ sinh mã nguồn, gợi ý thuật toán và giải thích logic chương trình. Các công cụ này thường hoạt động theo mô hình đăng ký thuê bao, với chi phí từ vài chục đến hàng trăm đô la Mỹ mỗi tháng, tùy thuộc vào mức độ sử dụng và tính năng đi kèm. Ưu điểm của các hệ thống này là giúp lập trình viên tăng năng suất và giảm lỗi trong quá trình viết mã. Tuy nhiên, phần lớn các công cụ chỉ hỗ trợ một giai đoạn cụ thể trong vòng đời phát triển phần mềm, chủ yếu tập trung vào sinh mã nguồn, mà chưa bao phủ đầy đủ các bước như phân tích yêu cầu, thiết kế giao diện và triển khai ứng dụng một cách thống nhất.

Việc khảo sát các ứng dụng và nền tảng tương tự cho thấy mặc dù nhiều hệ thống đã ứng dụng trí tuệ nhân tạo để tự động hóa một số công đoạn, nhưng vẫn tồn tại nhiều hạn chế. Cụ thể, các hệ thống hiện tại thường thiếu khả năng phối hợp linh hoạt giữa các chức năng, chưa phân tách rõ ràng vai trò của từng thành phần xử lý, dẫn đến khó khăn trong việc mở rộng và bảo trì. Ngoài ra, mức độ tương tác với người dùng trong quá trình làm rõ yêu cầu còn hạn chế, khiến kết quả cuối cùng đôi khi chưa đáp ứng đầy đủ mong muốn ban đầu. Về mặt chi phí, nhiều nền tảng yêu cầu mức phí tương đối cao khi sử dụng ở quy mô lớn, gây khó khăn cho các cá nhân hoặc nhóm phát triển nhỏ.

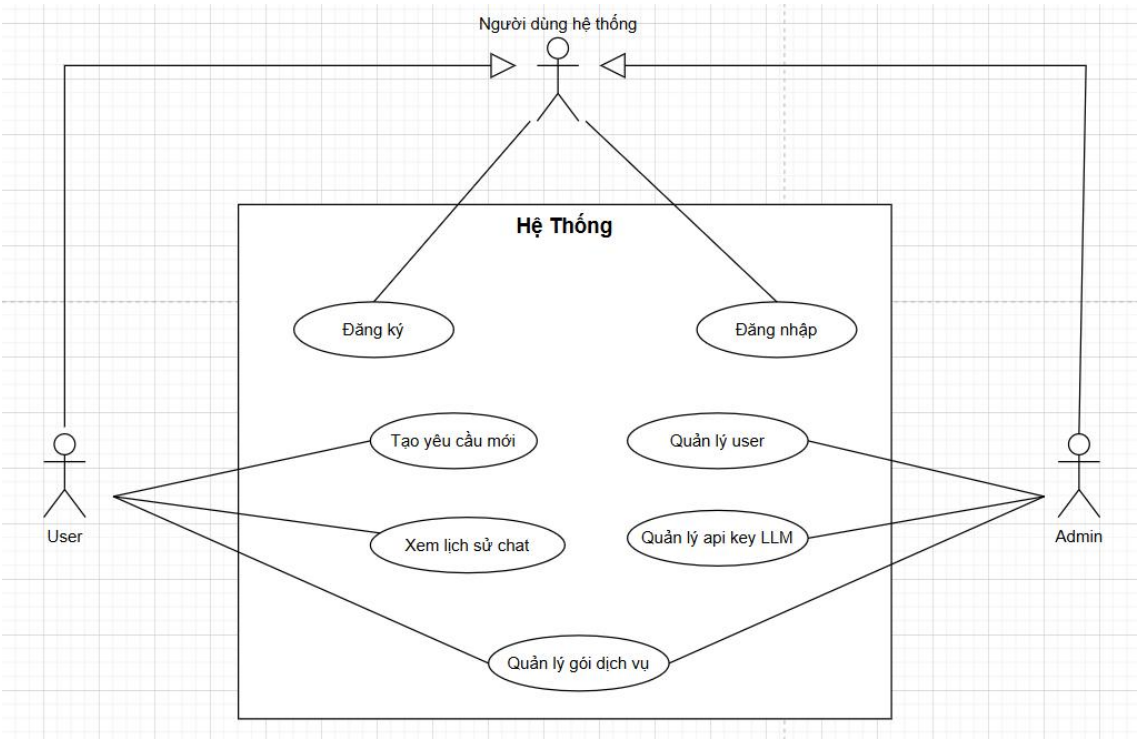
Từ quá trình khảo sát hiện trạng, có thể nhận thấy nhu cầu cấp thiết về một hệ thống hỗ trợ phát triển ứng dụng web theo hướng tự động hóa toàn diện và có khả năng phối hợp nhiều chức năng trong một quy trình thống nhất. Hệ thống cần cho phép người dùng tham gia trực tiếp vào quá trình xác nhận và điều chỉnh ở từng giai đoạn, từ phân tích yêu cầu, thiết kế giao diện, sinh mã nguồn cho đến triển khai ứng dụng. Trên cơ sở đó, các tính năng quan trọng cần phát triển bao gồm: thu thập và phân tích yêu cầu bằng ngôn ngữ tự nhiên, đề xuất giao diện người dùng tổng quan, sinh mã nguồn tự động và triển khai ứng dụng nhanh chóng. Những nội dung này sẽ được cụ thể hóa và phân tích chi tiết trong các mục tiếp theo của chương.

2.2 Tổng quan chức năng

Hệ thống sẽ có một số chức năng chính như sau: Tạo yêu cầu sinh giao diện mới, trong quá trình sinh có thể tham gia chỉnh sửa vào các giai đoạn như UI sketch, text promt,..Ngoài ra người dùng có thể xem lại những yêu cầu cũ và tiếp tục chỉnh sửa. Những UI sinh ra sẽ được deploy lên vercel và trả về liên kết cho người dùng có thể xem trước và download source code.

2.2.1 Biểu đồ use case tổng quát

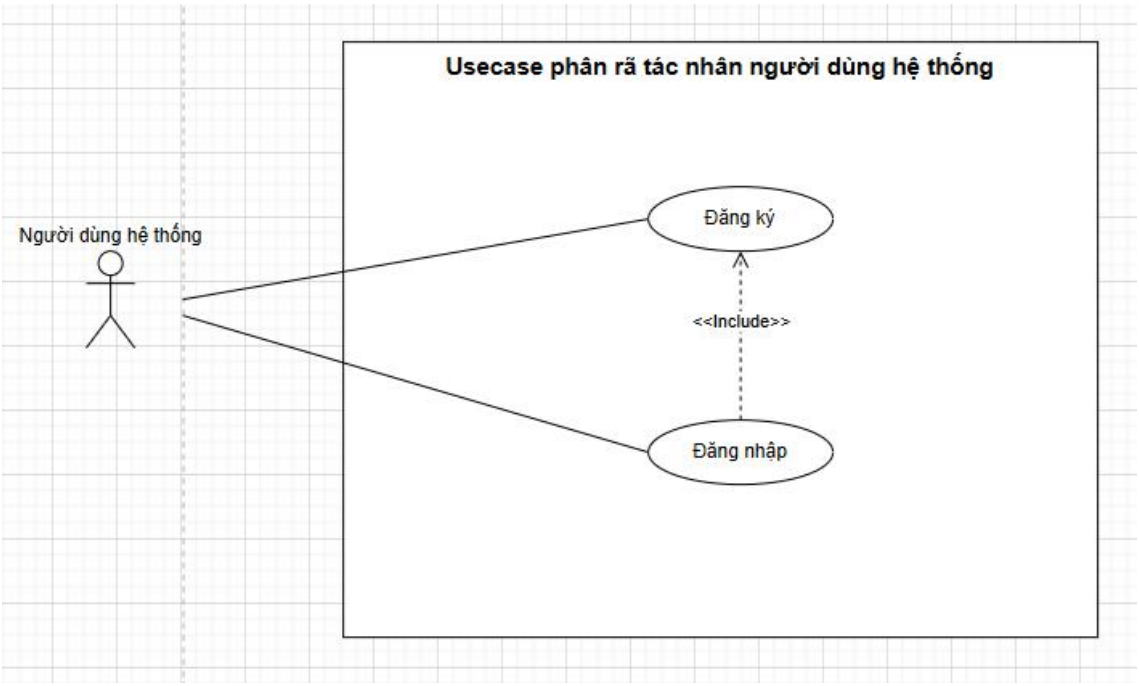
Hệ thống gồm có hai tác nhân chính như sau: Người dùng có thể đăng ký/hủy gói dịch vụ, tạo đoạn chat mới, xem lại lịch sử chat. Admin có thể quản lý hệ thống với các chức năng như cấu hình gói dịch vụ, quản lý user và quản lý key cho các dịch vụ LLM.



Hình 2.1: Biểu đồ use case tổng quan

2.2.2 Biểu đồ use case phân rã tác nhân người dùng hệ thống

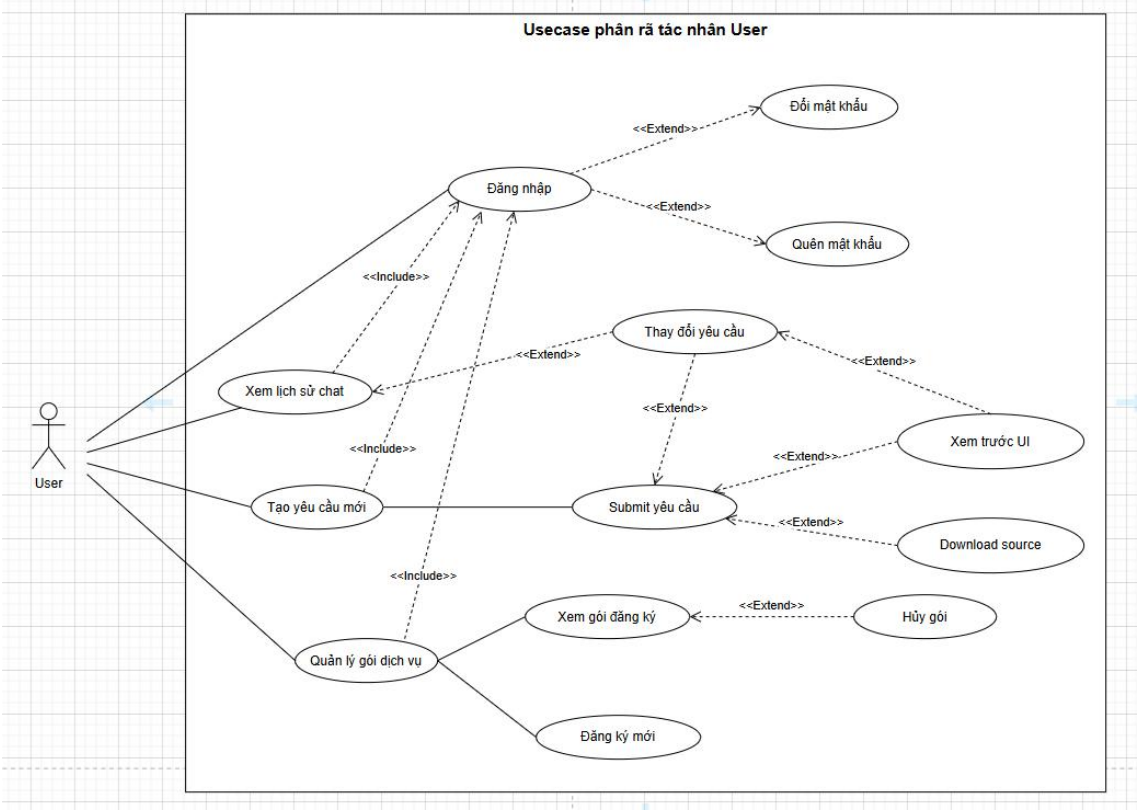
Use case này cho phép người dùng hệ thống thực hiện đăng ký tài khoản mới và đăng nhập vào hệ thống. Trong trường hợp người dùng chưa có tài khoản, người dùng tiến hành đăng ký, sau đó thực hiện đăng nhập để truy cập và sử dụng các chức năng của hệ thống. Việc đăng nhập là điều kiện cần để người dùng có thể thực hiện các nghiệp vụ chính trong hệ thống.



Hình 2.2: Biểu đồ use case phân rã tác nhân người dùng hệ thống

2.2.3 Biểu đồ use case phân rã tác nhân user

User có thể tạo yêu cầu chat mới hoặc sử dụng những đoạn chat cũ để chỉnh sửa yêu cầu và tạo UI. Sau đó có thể xem trước UI đã được thệ thống deploy và download source code. Ngoài ra user còn có thể tùy chỉnh hủy/ đăng ký mới gói dịch vụ tương ứng cho từng loại model được sử dụng để sinh code

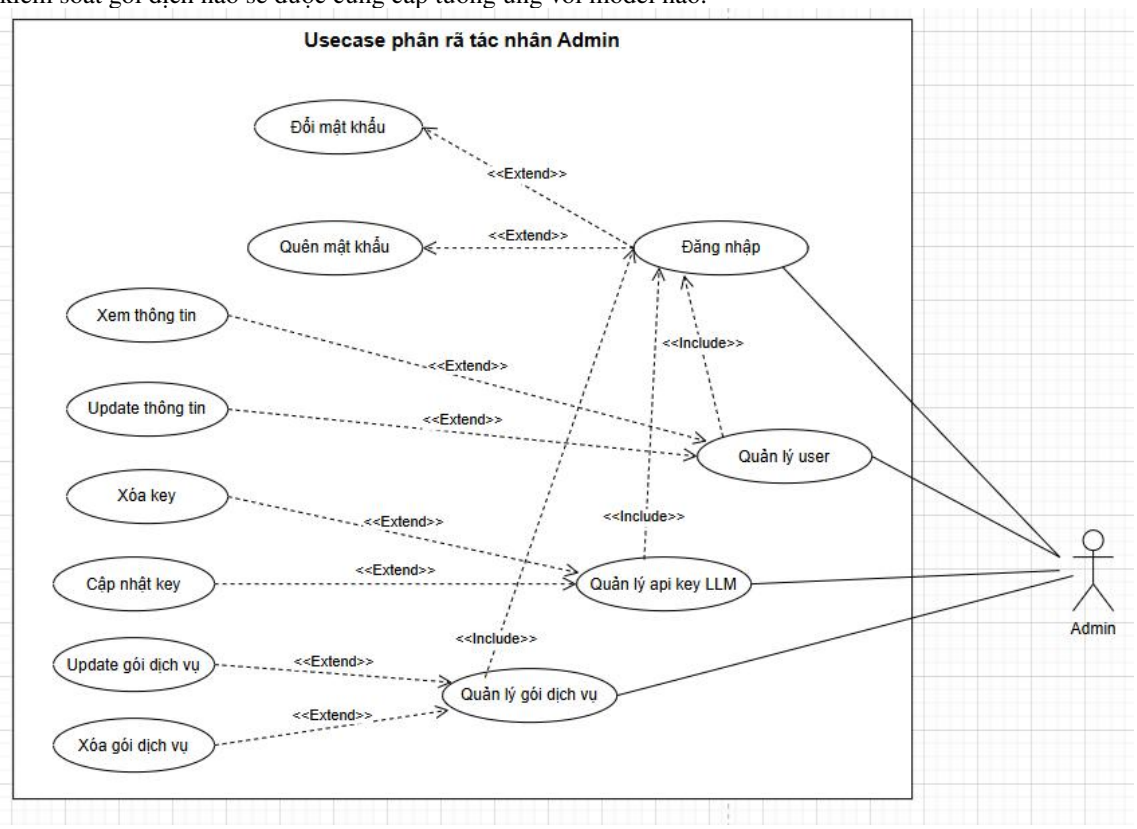


Hình 2.3: Biểu đồ use case phân rã tác nhân user

2.2.4 Biểu đồ use case phân rã tác nhân admin

Admin có thể quản lý các user trong hệ thống thông qua việc xem hoặc update thông tin người dùng. Ngoài ra admin cũng có thể quản lý hệ thống api key LLM. Đây là chìa khóa để kết nối đến các dịch vụ Gemini, Claude, Gpt. Admin cần quản lý, update một cách chặt chẽ để đảm bảo hoạt động của hệ thống.

Cuối cùng admin cần cần quản lý các gói dịch vụ mà hệ thống cung cấp cho user. Chức năng cần cập nhật, kiểm soát gói dịch nào sẽ được cung cấp tương ứng với model nào.



Hình 2.4: Biểu đồ use case phân rã tác nhân admin

2.3 Đặc tả chức năng

2.3.1 Đặc tả use case tạo yêu cầu mới

Mã Use case	A01	Tên usecase	Tạo yêu cầu mới
Mục đích	Giúp người dùng thực hiện yêu cầu gen UI mới		
Tác nhân	User		
Sự kiện kích hoạt	User nhập yêu cầu vào ô chat và gửi		
Điều kiện tiên quyết	User đã có tài khoản trong hệ thống		
Luồng sự kiện chính (Thành công)	STT	Thực hiện bởi	Hành động
	1	User	Chọn đoạn chat mới và nhập yêu cầu
	2	Hệ thống	Bóc tách yêu cầu để lấy thông tin về dự án
	3	User	Khi đã đủ thông tin sẽ yêu cầu bắt đầu
	4	Hệ thống	Sinh ảnh UI sketch
	5	User	Xem trước UI sketch
	6	User	Chỉnh sửa và submit UI sketch
	7	Hệ thống	Gen code dựa vào UI sketch và thông tin đã thu thập
	8	Hệ thống	Deploy UI, upload source code và trả về liên kết
	9	User	Xem trước UI và download source code
Luồng sự kiện thay thế	STT	Thực hiện bởi	Hành động
	2a	Hệ thống	Yêu cầu thông tin cần thiết còn thiếu từ user
Hậu điều kiện	Hiển thị giao diện đã được sinh cho user		

Bảng 2.1: Tạo yêu cầu mới

2.3.2 Đặc tả use case xem lịch sử chat

Mã Use case	A02	Tên usecase	Xem lịch sử chat
Mục đích	Giúp người dùng xem lại hoặc tiếp tục triển khai yêu cầu cũ		
Tác nhân	User		
Sự kiện kích hoạt	User chọn đoạn chat trong danh sách lịch sử		
Điều kiện tiên quyết	User đã có đoạn chat cũ trước đó		
Luồng sự kiện chính (Thành công)	STT	Thực hiện bởi	Hành động
	1	User	Chọn đoạn chat trong lịch sử chat
	2	Hệ thống	Hiển thị thông tin đoạn chat cho user
	3	User	Xem lại lịch sử chat
Luồng sự kiện thay thế	STT	Thực hiện bởi	Hành động
	3a	User	Cập nhật yêu cầu cũ
Hậu điều kiện	Hiển thị giao diện đã được sinh cho user		

Bảng 2.2: Xem lịch sử chat

2.3.3 Đặc tả use case quản lý gói dịch vụ

Mã Use case	A01	Tên usecase	Quản lý gói dịch vụ
Mục đích	Giúp admin quản lý, cấu hình các gói dịch vụ của hệ thống		
Tác nhân	Admin		
Sự kiện kích hoạt	Admin truy cập mục quản lý gói dịch vụ		
Điều kiện tiên quyết	Sử dụng tài khoản admin		
Luồng sự kiện chính (Thành công)	STT	Thực hiện bởi	Hành động
	1	Admin	Chọn mục quản lý gói dịch vụ
	2	Hệ thống	Hiển thị danh sách gọi dịch vụ hiện có của hệ thống
	3	Admin	Chọn gói dịch vụ và tiến hành chỉnh sửa
	4	Hệ thống	Cập nhật gói dịch vụ
Luồng sự kiện thay thế	STT	Thực hiện bởi	Hành động
	3a	Admin	Chọn gói dịch vụ và xóa.
Hậu điều kiện	Danh sách gói dịch vụ được update.		

Bảng 2.3: Quản lý gói dịch vụ

2.3.4 Đặc tả use case quản lý api key LLM

Mã Use case	A01	Tên usecase	Quản lý api key LLM
Mục đích	Giúp admin quản lý danh sách api key LLM của hệ thống		
Tác nhân	Admin		
Sự kiện kích hoạt	Admin truy cập mục quản lý api key LLM		
Điều kiện tiên quyết	Sử dụng tài khoản admin		
Luồng sự kiện chính (Thành công)	STT	Thực hiện bởi	Hành động
	1	Admin	Chọn mục quản lý api key LLM
	2	Hệ thống	Hiển thị danh sách api key LLM hiện có của hệ thống
	3	Admin	Chọn dịch vụ LLM và tiến hành update api key
	4	Hệ thống	Cập nhật api key cho dịch vụ LLM
Luồng sự kiện thay thế	STT	Thực hiện bởi	Hành động
	3a	Admin	Xóa api key của dịch vụ LLM khỏi hệ thống.
Hậu điều kiện	Danh sách api key LLM được update.		

Bảng 2.4: Quản lý api key LLM

2.4 Yêu cầu phi chức năng

Bên cạnh các yêu cầu chức năng đã được xác định, hệ thống web hỗ trợ phát triển ứng dụng tự động theo mô hình đa tác tử cần đáp ứng một số yêu cầu phi chức năng quan trọng nhằm đảm bảo tính ổn định, hiệu quả và khả năng ứng dụng thực tế trong quá trình vận hành. Các yêu cầu phi chức năng của hệ thống được xem xét trên các khía cạnh chính bao gồm: hiệu năng, độ tin cậy, tính dễ sử dụng, tính dễ bảo trì và các yêu cầu về mặt kỹ thuật.

Hiệu năng hệ thống

Hiệu năng là một trong những yêu cầu phi chức năng quan trọng đối với hệ thống, đặc biệt trong bối cảnh hệ thống cần xử lý các yêu cầu phức tạp liên quan đến phân tích ngôn ngữ tự nhiên, sinh giao diện, sinh mã nguồn và triển khai ứng dụng. Hệ thống cần đảm bảo thời gian phản hồi hợp lý đối với các thao tác chính của người dùng như tạo yêu cầu mới, chỉnh sửa giao diện, sinh mã nguồn và xem kết quả triển khai. Các tác vụ xử lý dài được thiết kế theo cơ chế bất đồng bộ nhằm tránh gây gián đoạn trải nghiệm người dùng. Đồng thời, hệ thống cần có khả năng phục vụ nhiều người dùng truy cập và sử dụng đồng thời mà không làm suy giảm đáng kể hiệu suất tổng thể.

Độ tin cậy và ổn định

Hệ thống cần đảm bảo hoạt động ổn định và đáng tin cậy trong suốt quá trình sử dụng. Các thành phần chính như quản lý phiên làm việc, lưu trữ lịch sử yêu cầu và kết quả sinh mã cần được thiết kế sao cho hạn chế tối đa lỗi phát sinh. Trong trường hợp xảy ra sự cố, hệ thống cần có khả năng ghi nhận lỗi và phục hồi ở mức độ phù hợp, đảm bảo không làm mất dữ liệu quan trọng của người dùng. Việc duy trì trạng thái nhất quán giữa các tác tử trong kiến trúc đa tác tử cũng là một yêu cầu quan trọng nhằm đảm bảo tính chính xác của kết quả sinh ra.

Tính dễ sử dụng

Tính dễ sử dụng là yêu cầu quan trọng đối với hệ thống, đặc biệt khi đối tượng người dùng không chỉ bao gồm các lập trình viên chuyên nghiệp mà còn có thể là những người không có nền tảng kỹ thuật sâu. Giao diện người dùng cần được thiết kế trực quan, rõ ràng và dễ thao tác, cho phép người dùng dễ dàng mô tả yêu cầu bằng ngôn ngữ tự nhiên và tham gia chỉnh sửa ở từng giai đoạn như phân tích yêu cầu, phác thảo giao diện và sinh mã nguồn. Các chức năng chính cần được bố trí hợp lý, hạn chế các thao tác phức tạp và

cung cấp phản hồi rõ ràng trong quá trình sử dụng. Ngoài ra, hệ thống cần có hướng dẫn sử dụng cơ bản nhằm hỗ trợ người dùng nhanh chóng làm quen và khai thác hiệu quả các chức năng.

Tính dễ bảo trì và mở rộng

Hệ thống được thiết kế theo kiến trúc đa tác tử nhằm tăng tính dễ bảo trì và khả năng mở rộng trong tương lai. Mỗi tác tử đảm nhiệm một vai trò riêng biệt, cho phép dễ dàng nâng cấp, chỉnh sửa hoặc thay thế mà không ảnh hưởng lớn đến toàn bộ hệ thống. Việc ghi log và theo dõi trạng thái hoạt động của các tác tử giúp hỗ trợ quá trình phát hiện và xử lý lỗi. Ngoài ra, hệ thống cần cho phép bổ sung thêm các tác tử mới hoặc mở rộng chức năng hiện có khi nhu cầu sử dụng tăng lên, đáp ứng tốt yêu cầu phát triển lâu dài.

Yêu cầu về mặt kỹ thuật

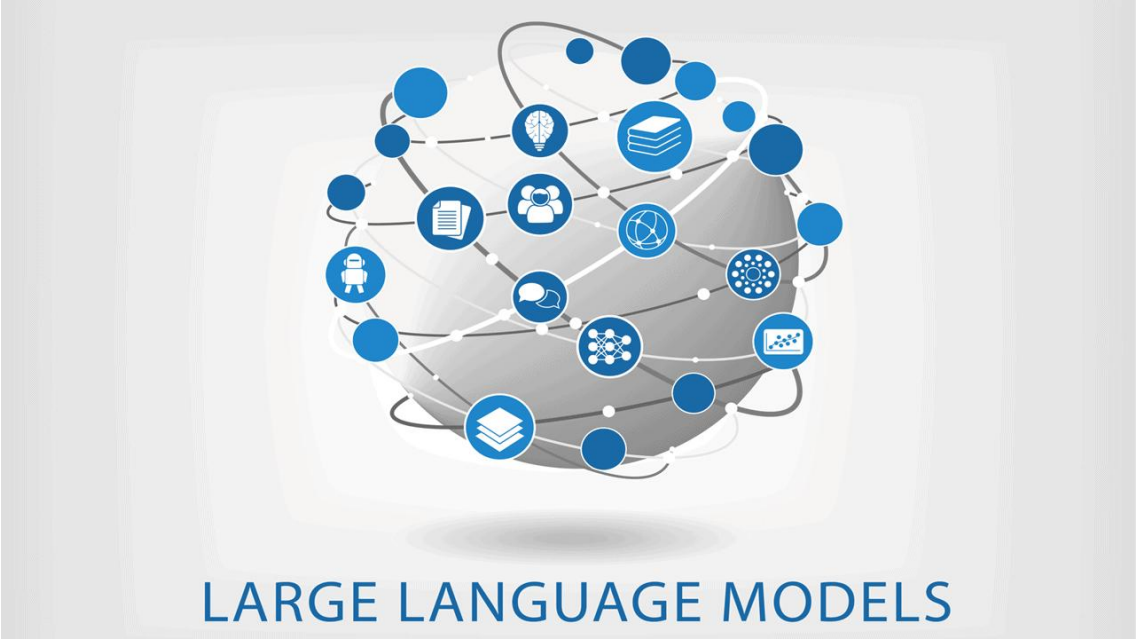
Về mặt kỹ thuật, hệ thống cần sử dụng các công nghệ phát triển web hiện đại nhằm đảm bảo khả năng triển khai nhanh chóng và khả năng mở rộng linh hoạt. Hệ thống cần hỗ trợ lưu trữ dữ liệu hiệu quả cho các thông tin như tài khoản người dùng, lịch sử yêu cầu, kết quả sinh giao diện và mã nguồn. Việc giao tiếp giữa các thành phần trong hệ thống cần đảm bảo an toàn và nhất quán thông qua các giao thức mạng phù hợp. Ngoài ra, hệ thống cần có khả năng triển khai và cập nhật ứng dụng một cách thuận tiện, cho phép người dùng nhanh chóng truy cập sản phẩm sau khi hoàn tất quá trình sinh và triển khai.

Nhìn chung, các yêu cầu phi chức năng nêu trên đóng vai trò quan trọng trong việc đảm bảo hệ thống hoạt động hiệu quả, ổn định và đáp ứng tốt nhu cầu của người dùng. Đây là cơ sở quan trọng cho quá trình thiết kế kiến trúc và triển khai hệ thống trong các chương tiếp theo của đồ án.

CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

Chương này trình bày các công nghệ và nền tảng được sử dụng trong quá trình xây dựng hệ thống web hỗ trợ phát triển ứng dụng tự động theo mô hình đa tác tử. Các công nghệ được lựa chọn dựa trên mức độ phổ biến, khả năng mở rộng, hiệu năng xử lý cũng như sự phù hợp với các yêu cầu chức năng và phi chức năng đã phân tích ở Chương 2.

3.1 Mô hình ngôn ngữ lớn (Large Language Model – LLM)



Hình 3.1: Large Language Model

Mô hình ngôn ngữ lớn (Large Language Model – LLM) là thành phần cốt lõi của hệ thống, đóng vai trò trung tâm trong việc xử lý ngôn ngữ tự nhiên, phân tích yêu cầu người dùng và sinh nội dung tự động. LLM cho phép hệ thống tiếp nhận đầu vào là các mô tả nghiệp vụ, yêu cầu chức năng và mong muốn của người dùng dưới dạng ngôn ngữ tự nhiên, từ đó chuyển đổi chúng thành các biểu diễn có cấu trúc phục vụ cho các bước thiết kế và triển khai ứng dụng. Điều này đáp ứng trực tiếp yêu cầu về tính thân thiện và khả năng sử dụng của hệ thống đã được phân tích trong Chương 2.

Trong đồ án này, hệ thống tích hợp các mô hình ngôn ngữ lớn thông qua API, bao gồm GPT, Claude và Gemini. Việc sử dụng nhiều mô hình khác nhau giúp tận dụng thế mạnh riêng của từng LLM trong các giai đoạn của quy trình phát triển phần mềm. Cụ thể, các mô hình LLM được sử dụng để hỗ trợ thu thập và làm rõ yêu cầu người dùng, phân tích nghiệp vụ, đề xuất kiến trúc tổng quan, phác thảo giao diện người dùng, sinh mã nguồn ở mức khung và hỗ trợ giải thích, chỉnh sửa kết quả theo phản hồi của người dùng.

Ưu điểm nổi bật của LLM là khả năng hiểu và sinh ngôn ngữ tự nhiên với độ linh hoạt và chính xác cao. Nhờ đó, người dùng không cần có kiến thức lập trình chuyên sâu vẫn có thể mô tả yêu cầu hệ thống một cách trực quan. LLM có khả năng tổng hợp thông tin từ nhiều nguồn, nhận diện các thực thể và mối quan hệ trong nghiệp vụ, từ đó hỗ trợ xây dựng các đặc tả chức năng và quy trình nghiệp vụ ở mức tổng quan. Bên cạnh đó, LLM còn có khả năng sinh nội dung kỹ thuật như mô tả API, cấu trúc dữ liệu và mã nguồn mẫu, góp phần rút ngắn đáng kể thời gian phát triển phần mềm so với phương pháp truyền thống.

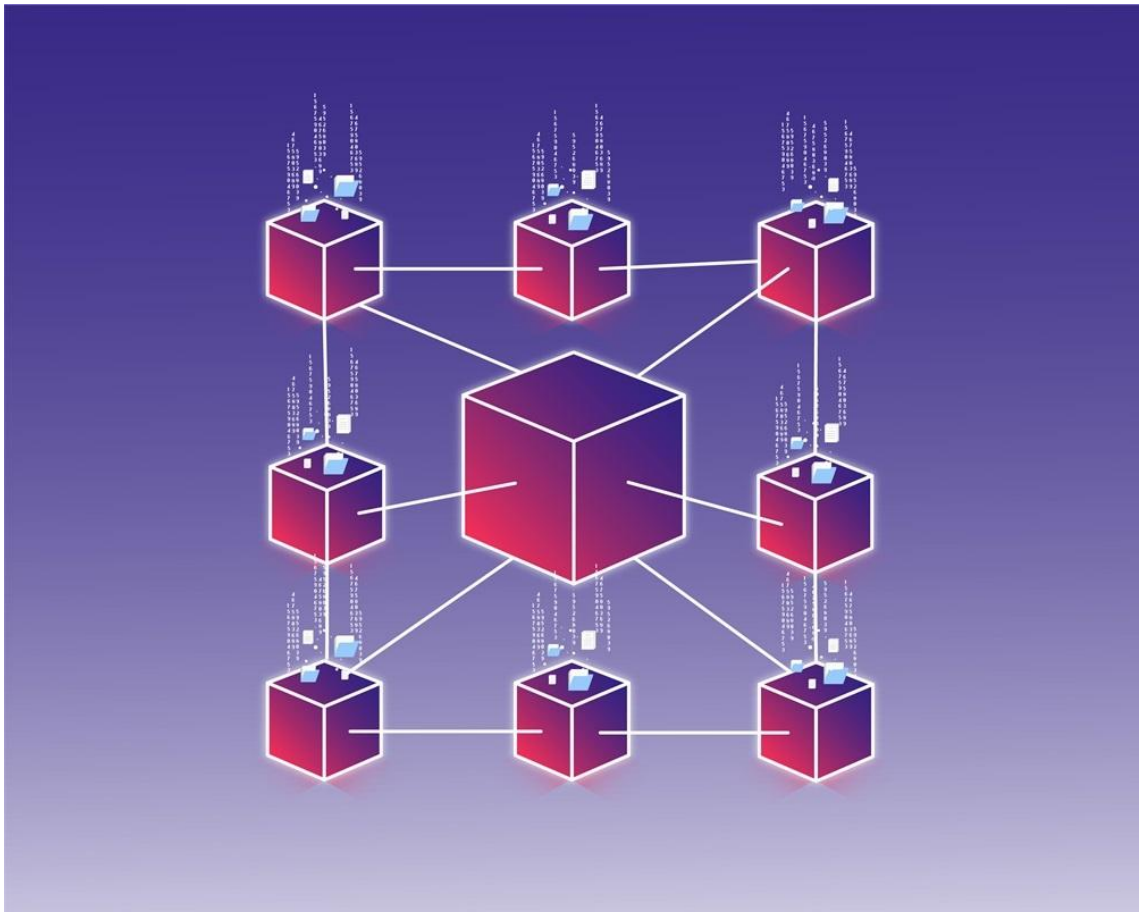
Ngoài các ưu điểm kể trên, việc sử dụng LLM còn giúp tăng tính linh hoạt cho hệ thống. Các mô hình có thể được thay thế hoặc nâng cấp thông qua API mà không cần thay đổi kiến trúc tổng thể. Điều này giúp hệ thống dễ dàng thích nghi với sự phát triển nhanh chóng của các công nghệ trí tuệ nhân tạo và các mô

hình ngôn ngữ mới trong tương lai.

Tuy nhiên, LLM cũng tồn tại một số hạn chế nhất định. Kết quả sinh ra phụ thuộc nhiều vào ngữ cảnh đầu vào và cách diễn đạt yêu cầu của người dùng, do đó có thể xuất hiện các nội dung chưa chính xác, chưa đầy đủ hoặc chưa phù hợp hoàn toàn với yêu cầu thực tế. Ngoài ra, việc sử dụng LLM thông qua API chịu ảnh hưởng bởi các yếu tố như chi phí sử dụng, độ trễ phản hồi và chính sách giới hạn của nhà cung cấp dịch vụ. Bên cạnh đó, LLM không đảm bảo tuyệt đối tính đúng đắn của mã nguồn sinh ra và có thể tạo ra các giải pháp chưa tối ưu về mặt kỹ thuật.

Để khắc phục các hạn chế trên, trong hệ thống, LLM được kết hợp với cơ chế kiểm soát theo từng bước và xác nhận từ phía người dùng. Các kết quả trung gian như phân tích yêu cầu, phác thảo giao diện và mã nguồn sinh ra đều được trình bày để người dùng xem xét, chỉnh sửa và xác nhận trước khi chuyển sang giai đoạn tiếp theo. Cách tiếp cận này giúp nâng cao độ tin cậy của hệ thống, đồng thời đảm bảo rằng sản phẩm cuối cùng phù hợp với nhu cầu thực tế của người dùng.

3.2 Kiến trúc đa tác tử (Multi-Agent Architecture)



Hình 3.2: Multi-Agent Architecture

Hệ thống trong đồ án được xây dựng dựa trên kiến trúc đa tác tử (Multi-Agent Architecture), trong đó toàn bộ quy trình phát triển ứng dụng web được chia nhỏ thành các tác vụ độc lập và được đảm nhiệm bởi các tác tử chuyên biệt. Mỗi tác tử đại diện cho một vai trò cụ thể trong vòng đời phát triển phần mềm, từ giai đoạn thu thập yêu cầu, thiết kế giao diện, sinh mã nguồn cho đến triển khai và cung cấp sản phẩm cuối cùng cho người dùng. Kiến trúc này phù hợp với yêu cầu tự động hóa quy trình phát triển ứng dụng đã được phân tích trong Chương 2, đồng thời giúp hệ thống vận hành một cách linh hoạt và có khả năng mở rộng cao.

Trong hệ thống, các tác tử chính được xây dựng và phân công nhiệm vụ như sau:

Orchestration Agent (Orches) là tác tử điều phối trung tâm của toàn bộ hệ thống. Tác tử này chịu trách nhiệm quản lý luồng xử lý tổng thể, phân phối nhiệm vụ cho các tác tử khác, thu thập kết quả trung gian và đảm bảo quá trình trao đổi thông tin giữa các tác tử diễn ra nhất quán. Orches đóng vai trò quan trọng trong việc kiểm soát trạng thái hệ thống và đảm bảo các bước trong quy trình phát triển được thực hiện đúng thứ tự.

Business Analyst Agent (BA) đảm nhiệm vai trò thu thập, phân tích và làm rõ yêu cầu từ phía người dùng. Tác tử này tương tác trực tiếp với người dùng thông qua ngôn ngữ tự nhiên, hỗ trợ chuyển đổi các mô tả nghiệp vụ chưa rõ ràng thành các yêu cầu có cấu trúc. BA giúp giảm thiểu sai lệch trong quá trình hiểu yêu cầu, từ đó nâng cao chất lượng của các bước thiết kế và triển khai về sau.

SketchUI Agent có nhiệm vụ đề xuất và sinh giao diện người dùng ở mức tổng quan dựa trên các yêu cầu đã được phân tích và xác nhận. Tác tử này tập trung vào việc xây dựng bố cục giao diện, cấu trúc các thành phần chính và luồng tương tác cơ bản, giúp người dùng hình dung được giao diện của ứng dụng trước khi tiến hành sinh mã nguồn chi tiết. Kết quả của SketchUI Agent có thể được chỉnh sửa và xác nhận lại bởi người dùng trước khi chuyển sang giai đoạn tiếp theo.

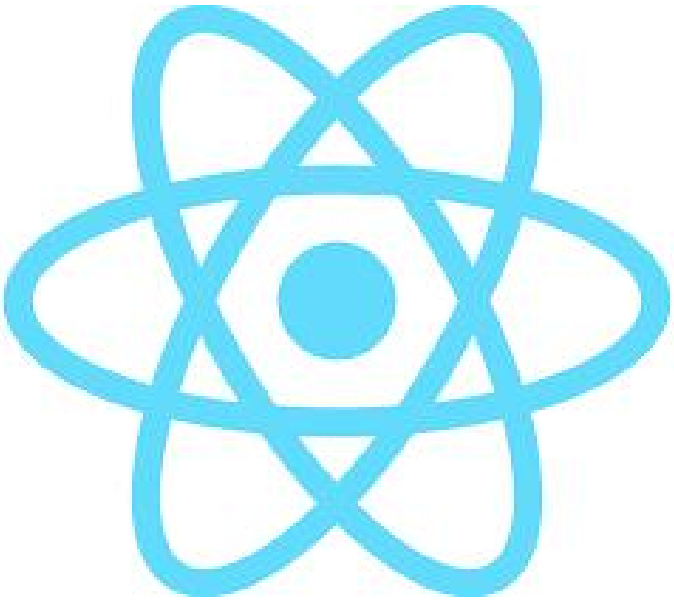
Generate Code Agent chịu trách nhiệm sinh mã nguồn ứng dụng dựa trên giao diện và nghiệp vụ đã được xác nhận. Tác tử này tạo ra các đoạn mã ở mức khung, bao gồm cấu trúc dự án, các thành phần giao diện và các chức năng chính của hệ thống. Việc tách riêng tác tử sinh mã nguồn giúp hệ thống dễ dàng mở rộng sang nhiều ngôn ngữ lập trình hoặc framework khác nhau trong tương lai.

Deploy Agent đảm nhiệm giai đoạn triển khai ứng dụng. Tác tử này thực hiện đóng gói, triển khai ứng dụng lên môi trường chạy thực tế và cung cấp đường dẫn truy cập cho người dùng. Nhờ đó, toàn bộ quy trình từ yêu cầu ban đầu đến sản phẩm cuối cùng được tự động hóa ở mức cao, giảm thiểu các thao tác thủ công.

Ưu điểm nổi bật của kiến trúc đa tác tử là khả năng phân tách rõ ràng chức năng giữa các thành phần trong hệ thống. Mỗi tác tử có thể được phát triển, nâng cấp hoặc thay thế độc lập mà không ảnh hưởng đến toàn bộ hệ thống, từ đó giúp tăng tính linh hoạt và khả năng bảo trì. Bên cạnh đó, kiến trúc này cho phép mở rộng hệ thống bằng cách bổ sung thêm các tác tử mới để hỗ trợ các chức năng nâng cao trong tương lai.

Tuy nhiên, kiến trúc đa tác tử cũng đặt ra một số thách thức. Việc phối hợp giữa nhiều tác tử làm gia tăng độ phức tạp trong quản lý luồng xử lý, đồng bộ dữ liệu và trạng thái hệ thống. Nếu cơ chế điều phối không được thiết kế chặt chẽ, hệ thống có thể gặp phải các vấn đề như xung đột thông tin, sai lệch dữ liệu hoặc lặp lại xử lý. Do đó, vai trò của Orchestration Agent là đặc biệt quan trọng trong việc đảm bảo tính nhất quán và ổn định của toàn bộ hệ thống.

3.3 ReactJS:



Hình 3.3: ReactJS:

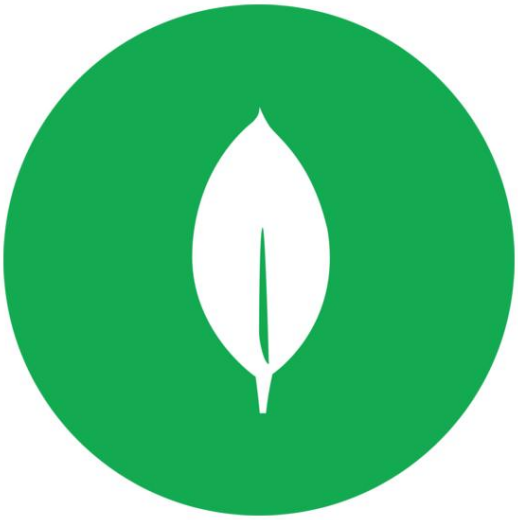
ReactJS là một thư viện JavaScript được tạo ra để xây dựng giao diện người dùng có khả năng tương tác tốt và nhanh chóng cho các ứng dụng web và di động. Nó là một thư viện mã nguồn mở, xây dựng dựa trên các component, giao diện người dùng chỉ chịu trách nhiệm cho tầng view của ứng dụng. ReactJS mang lại nhiều lợi thế cho các lập trình viên, khiến nó trở thành một lựa chọn tốt hơn so với các Framework khác như Angular. Các tính năng nổi bật của React bao gồm: (i) JSX được sử dụng với ReactJS để mô tả giao diện người dùng trông như thế nào. Bằng cách sử dụng JSX, ta có thể viết các cấu trúc HTML trong cùng một tệp chứa mã JavaScript. Điều này làm cho mã dễ hiểu và dễ gỡ lỗi hơn, vì nó tránh việc sử dụng các cấu trúc DOM JavaScript phức tạp,(ii) Virtual DOM là một định dạng dữ liệu của JavaScript, với khối lượng nhẹ và được dùng để thể hiện nội dung của DOM tại một thời điểm nhất định nào đó. Khi trạng thái của một đối tượng thay đổi, VDOM chỉ thay đổi đối tượng đó trong DOM thực thay vì cập nhật tất cả các đối tượng. React sử dụng sử dụng VDOM, giúp các ứng dụng web chạy nhanh hơn nhiều so với những ứng dụng được phát triển với các Framework front-end khác. ReactJS chia giao diện người dùng phức tạp thành các thành phần riêng lẻ, cho phép nhiều người dùng làm việc trên từng thành phần đồng thời, do đó đẩy nhanh thời gian phát triển cũng như cải thiện hiệu năng của ứng dụng, (iii) Redux giúp quản lý trạng thái ứng dụng một cách hiệu quả và có tổ chức, đặc biệt là trong các dự án lớn và phức tạp, (iv) One-way Data Binding ám chỉ việc truyền dữ liệu chỉ theo một hướng, từ component cha đến component con, làm cho mã nguồn trở nên rõ ràng và dễ hiểu. Điều này giúp tăng tính dễ kiểm thử và dễ debug, tránh hiệu ứng lặp vô hạn và cung cấp hiệu suất tốt hơn. One-way data binding cũng tạo điều kiện thuận lợi cho quản lý re-render và giữ cho ứng dụng linh hoạt và hiệu quả trong quá trình phát triển. Tuy nhiên, React tồn tại 1 số nhược điểm, như là việc học React đôi khi có thể đầy thách thức với người mới bắt đầu, đặc biệt là khi phải làm quen với các khái niệm như JSX, Components, State và Props. Một vấn đề khác là sự đa dạng lớn của cộng đồng React, với nhiều lựa chọn về thư viện và công nghệ bổ sung. Sự phong phú này có thể tạo ra khó khăn trong quá trình lựa chọn và kết hợp các công cụ phù hợp với dự án cụ thể. Hay việc đối mặt với thách thức SEO, React đôi khi gặp vấn đề khi triển khai các ứng dụng SPA, đòi hỏi sự tối ưu hóa đặc biệt để đảm bảo khả năng tương tác tốt với các công cụ tìm kiếm. Việc quản lý trạng thái của ứng dụng trong React cũng có thể trở nên phức tạp khi dự án phát triển lớn, đòi hỏi sự quan tâm đặc biệt đối với việc sử dụng các giải pháp như Redux hoặc MobX. Cuối cùng, mặc dù React cải thiện hiệu suất thông qua Virtual DOM, nhưng vấn đề về hiệu suất vẫn có thể nảy sinh nếu không quản lý tốt trạng thái và render components. Những nhược điểm này, mặc dù không làm giảm giá trị của React, nhưng đều là những thách thức cần xem xét và giải quyết khi xây dựng ứng dụng.

3.4 NodeJS:

Hình 3.4: NodeJs

Node.js là một mã nguồn mở, một môi trường cho các máy chủ và ứng dụng mạng, sử dụng Google V8 JavaScript engine để thực thi mã, và một tỷ lệ lớn các module cơ bản được viết bằng JavaScript. Các ứng dụng node.js thì được viết bằng JavaScript. Node.js chứa một thư viện built-in cho phép các ứng dụng hoạt động như một Webserver mà không cần phần mềm như Nginx, Apache HTTP Server hoặc IIS. Node.js cung cấp kiến trúc hướng sự kiện (event-driven) và non-blocking I/O API, tối ưu hóa thông lượng của ứng dụng và có khả năng mở rộng cao. Mọi hàm trong Node.js là không đồng bộ (asynchronous). Do đó, các tác vụ đều được xử lý và thực thi ở chế độ nền (background processing). Node.js có rất nhiều ưu điểm nhiều ưu điểm khi sử dụng trong các tình huống cụ thể. Với khả năng xử lý JSON APIs, Node.js trở thành lựa chọn ưu việt cho việc xây dựng RESTful APIs, nhờ vào cơ chế event-driven và non-blocking I/O. Đối với ứng dụng trên một trang, Node.js thích hợp với việc xử lý nhiều request đồng thời, giúp ứng dụng có thời gian phản hồi nhanh và giữ cho trang web hoạt động mượt mà, lý tưởng cho các ứng dụng như Gmail. Trên Unix, Node.js có khả năng tận dụng tối đa để xử lý hàng nghìn process và tạo ra môi trường hoạt động hiệu quả, đặc biệt là trong các công cụ shell Unix. Node.js cũng thích hợp cho việc xử lý streaming data, nơi nó có thể xây dựng các proxy để phân vùng và quản lý hiệu suất cho các luồng dữ liệu lớn. Đặc biệt, trong việc xây dựng ứng dụng web thực (real-time), như các ứng dụng chat hoặc feed của Facebook, Twitter, Node.js làm việc hiệu quả và linh hoạt, giúp tạo ra trải nghiệm người dùng mượt mà và thực tế. Mặc dù được đánh giá cao với mô hình non-blocking I/O và khả năng xử lý hàng nghìn kết nối đồng thời, nhưng vẫn mang theo một số nhược điểm đáng chú ý. Điều đầu tiên là tính chất single-threaded của nó, tuy đã giúp giảm thiểu overhead trong xử lý đồng thời, nhưng cũng tạo ra thách thức khi đối mặt với các tác vụ nặng nề và đòi hỏi sự can thiệp của nhiều CPU. Một vấn đề khác là hiện tượng "callback hell" hoặc "pyramid of doom," xuất phát từ việc sử dụng nhiều callback, làm cho mã nguồn trở nên khó đọc và khó bảo trì. Trong khi đó, khả năng xử lý dữ liệu lớn và phức tạp của Node.js không luôn hiệu quả, đặc biệt so với một số nền tảng như Java hoặc Python. Node.js cũng gặp khó khăn trong việc đối mặt với các ứng dụng CPU-intensive, nơi nó thường không phải là lựa chọn tốt nhất, đặc biệt là khi đối mặt với xử lý đồ họa hoặc tính toán phức tạp. Cuối cùng, quản lý trạng thái trong các ứng dụng lớn đôi khi trở nên phức tạp, vì Node.js không cung cấp một framework chuẩn để quản lý trạng thái như một số ngôn ngữ và nền tảng khác.

3.5 MongoDB:



Hình 3.5: MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu phi quan hệ (NoSQL) mã nguồn mở, được thiết kế để lưu trữ và quản lý dữ liệu theo mô hình tài liệu (document-oriented). Nó sử dụng một định dạng dữ liệu linh hoạt, thường là BSON, để biểu diễn dữ liệu trong các bản ghi, còn được gọi là "tài liệu". MongoDB đem lại nhiều ưu điểm đáng kể trong quá trình phát triển ứng dụng. MongoDB sử dụng mô hình dữ liệu NoSQL, cho phép lưu trữ dữ liệu dưới dạng BSON giúp linh hoạt và dễ mở rộng khi cần thay đổi cấu trúc dữ liệu. Một trong những điểm mạnh nổi bật của MongoDB là khả năng mở rộng ngang (horizontal scaling), cho phép dễ dàng mở rộng khả năng chịu tải của hệ thống bằng cách thêm các node vào cụm MongoDB. Điều này giúp ứng dụng xử lý được lượng dữ liệu lớn và số lượng truy vấn đồng thời mà không làm giảm hiệu suất. MongoDB còn hỗ trợ các truy vấn phức tạp thông qua một ngôn ngữ truy vấn mạnh mẽ, cùng với khả năng indexing hiệu quả, giúp tối ưu hóa hiệu suất truy vấn và tìm kiếm. Điều này làm cho việc truy xuất dữ liệu nhanh chóng và hiệu quả. Khả năng lưu trữ dữ liệu dạng tài liệu (document-oriented) của MongoDB cũng mang lại lợi ích khi cần lưu trữ các đối tượng phức tạp, giảm độ phức tạp trong quá trình phát triển và bảo trì mã nguồn. Hơn nữa, MongoDB hỗ trợ tính năng tự động sharding, giúp chia nhỏ dữ liệu và phân phối nó trên nhiều máy chủ, tăng cường khả năng chịu tải và đồng thời cung cấp khả năng mở rộng dữ liệu mà không làm giảm hiệu suất. Một trong những nhược điểm quan trọng của MongoDB là khả năng phân đoạn trong quá trình thêm/xóa dữ liệu, khiến cho dữ liệu không được lưu trữ một cách liên tục trên đĩa và gây ra vấn đề về kích thước thực tế của cơ sở dữ liệu cùng với tốc độ truy xuất dữ liệu. Bên cạnh đó, MongoDB đòi hỏi một lượng bộ nhớ đáng kể để hoạt động hiệu quả, và vấn đề này có thể trở thành một thách thức đặc biệt khi dữ liệu ngày càng lớn. Trong một số trường hợp, việc truy vấn dữ liệu phức tạp và nâng cao cũng có thể trở nên phức tạp hơn so với các hệ quản trị cơ sở dữ liệu quan hệ truyền thống. Mặc dù MongoDB hỗ trợ các truy vấn linh hoạt, nhưng nó không hỗ trợ giao dịch trên nhiều tài liệu, điều này có thể tạo khó khăn khi cần thực hiện các thao tác liên quan đến nhiều tài liệu cùng một lúc. Đối với các ứng dụng đòi hỏi tính toàn vẹn cao, như hệ thống ngân hàng hoặc tài chính, MongoDB có thể không phải là lựa chọn lý tưởng vì khả năng khó khăn trong việc duy trì tính nhất quán dữ liệu.

3.6 Vercel

Hình 3.6: Vercel

Vercel là nền tảng triển khai (deployment platform) ứng dụng web được sử dụng trong hệ thống nhằm tự động hóa quá trình đưa sản phẩm từ môi trường phát triển lên môi trường vận hành. Trong đồ án này, Vercel được lựa chọn để triển khai các ứng dụng web được sinh ra từ hệ thống, đặc biệt là các ứng dụng sử dụng framework Next.js. Nền tảng này cung cấp cơ chế build và deploy tự động, giúp rút ngắn đáng kể thời gian đưa sản phẩm vào sử dụng so với phương pháp triển khai thủ công truyền thống.

Một trong những ưu điểm nổi bật của Vercel là khả năng tích hợp chặt chẽ với các hệ thống quản lý mã nguồn như GitHub. Mỗi khi mã nguồn được cập nhật, Vercel có thể tự động thực hiện quá trình build, kiểm tra và triển khai phiên bản mới của ứng dụng. Điều này phù hợp với mục tiêu của hệ thống là tự động hóa tối đa quy trình phát triển phần mềm, từ khâu thu thập yêu cầu cho đến khâu triển khai và cung cấp sản phẩm hoàn chỉnh cho người dùng cuối.

Bên cạnh đó, Vercel hỗ trợ cơ chế mở rộng linh hoạt, cho phép ứng dụng hoạt động ổn định khi số lượng người truy cập tăng lên. Việc cung cấp đường dẫn truy cập ngay sau khi triển khai giúp người dùng có thể nhanh chóng kiểm tra, đánh giá và sử dụng sản phẩm mà không cần thực hiện các cấu hình phức tạp về hạ tầng. Điều này đặc biệt phù hợp với các ứng dụng web được sinh tự động, nơi tốc độ phản hồi và trải nghiệm người dùng là yếu tố quan trọng.

Tuy nhiên, việc sử dụng Vercel cũng tồn tại một số hạn chế. Hệ thống phụ thuộc vào nền tảng triển khai bên thứ ba, dẫn đến rủi ro liên quan đến chính sách sử dụng, giới hạn tài nguyên và khả năng tùy biến hạ tầng. Trong gói sử dụng miễn phí, Vercel áp đặt các giới hạn nhất định về tài nguyên và băng thông, có thể ảnh hưởng đến hiệu năng khi hệ thống mở rộng quy mô. Khi số lượng ứng dụng được triển khai và lượng người dùng tăng lên, chi phí vận hành trên Vercel có thể tăng đáng kể. Do đó, trong tương lai, hệ thống có thể cân nhắc các phương án triển khai thay thế hoặc kết hợp nhằm tối ưu chi phí và hiệu năng.

3.7 GitHub

Hình 3.7: Github

GitHub là nền tảng quản lý mã nguồn và cộng tác phần mềm được sử dụng trong đồ án nhằm hỗ trợ quá trình phát triển, lưu trữ và kiểm soát phiên bản của hệ thống. Trong quá trình xây dựng hệ thống web hỗ trợ phát triển ứng dụng tự động, GitHub đóng vai trò là kho lưu trữ trung tâm, nơi toàn bộ mã nguồn của các thành phần trong hệ thống được quản lý một cách nhất quán và có hệ thống.

GitHub cho phép theo dõi lịch sử thay đổi của mã nguồn thông qua cơ chế kiểm soát phiên bản (version control), giúp nhóm phát triển dễ dàng xem lại các lần chỉnh sửa, so sánh sự khác biệt giữa các phiên bản và khôi phục mã nguồn khi cần thiết. Bên cạnh đó, cơ chế quản lý nhánh (branching) của GitHub hỗ trợ việc phát triển song song nhiều tính năng, giúp giảm xung đột trong quá trình làm việc nhóm và nâng cao hiệu quả phát triển phần mềm.

Một ưu điểm quan trọng khác của GitHub là khả năng tích hợp tốt với các công cụ và nền tảng phát triển hiện đại. Trong hệ thống này, GitHub được kết nối trực tiếp với nền tảng triển khai Vercel, cho phép tự động hóa quy trình triển khai mỗi khi mã nguồn được cập nhật. Điều này góp phần hiện thực hóa quy trình phát triển liên tục và triển khai liên tục (CI/CD), phù hợp với mục tiêu tự động hóa toàn bộ vòng đời phát triển ứng dụng của đồ án.

Ngoài ra, GitHub còn cung cấp các công cụ hỗ trợ quản lý dự án như Issues, Pull Requests và Discussions, giúp theo dõi tiến độ công việc, trao đổi kỹ thuật và kiểm soát chất lượng mã nguồn. Các công cụ này đặc biệt hữu ích trong quá trình phát triển hệ thống phức tạp, nơi nhiều thành phần và chức năng cần được phối hợp chặt chẽ.

Tuy nhiên, việc sử dụng GitHub cũng đặt ra một số thách thức nhất định. Do là nền tảng trực tuyến, hệ thống phụ thuộc vào kết nối mạng và chính sách vận hành của bên thứ ba. Bên cạnh đó, việc quản lý quyền truy cập và bảo mật mã nguồn cần được chú trọng, đặc biệt đối với các dự án có yêu cầu cao về bảo mật và quyền riêng tư. Trong trường hợp không được cấu hình và quản lý chặt chẽ, nguy cơ rò rỉ thông tin hoặc truy cập trái phép có thể xảy ra.

Chương này đã trình bày các công nghệ chính được sử dụng trong đồ án, bao gồm mô hình ngôn ngữ lớn, kiến trúc đa tác tử, các nền tảng phát triển web, cơ sở dữ liệu và công nghệ triển khai. Việc lựa chọn các công nghệ này nhằm đảm bảo hệ thống đáp ứng tốt các yêu cầu chức năng đã đề ra, đồng thời tạo nền tảng cho việc thiết kế và triển khai hệ thống ở các chương tiếp theo.

CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

Chương 4 sẽ đi phân tích các kết quả thực hiện, bao gồm phân tích thiết kế kiến trúc, thiết kế tổng quan, thiết kế chi tiết gói, thiết kế giao diện, thiết kế gói, thiết kế cơ sở dữ liệu và minh họa, triển khai của sản phẩm. Từ đó có cái nhìn rõ ràng, tổng quan về hệ thống đa tác tử hỗ trợ phát triển giao diện web.

4.1 Thiết kế kiến trúc

4.1.1 Lựa chọn kiến trúc phần mềm

Dựa trên mục tiêu xây dựng hệ thống web hỗ trợ phát triển giao diện và ứng dụng theo hướng tự động hóa, thực hiện theo quy trình khép kín từ thu thập yêu cầu đến sinh mã và triển khai sản phẩm, đồ án lựa chọn **kiến trúc đa tác tử (Multi-Agent Architecture)** làm kiến trúc phần mềm cốt lõi cho toàn bộ hệ thống.

Khác với các kiến trúc truyền thống tập trung vào phân lớp giao diện – nghiệp vụ – dữ liệu, kiến trúc đa tác tử tập trung mô hình hóa hệ thống như một tập hợp các tác tử thông minh, mỗi tác tử đảm nhiệm một vai trò cụ thể trong quy trình phát triển phần mềm. Cách tiếp cận này phù hợp với bài toán của đồ án, trong đó quy trình phát triển giao diện web được chia thành nhiều giai đoạn độc lập nhưng có mối quan hệ chặt chẽ và cần sự phối hợp liên tục giữa các thành phần xử lý.

Kiến trúc đa tác tử

Kiến trúc đa tác tử là mô hình kiến trúc trong đó hệ thống được cấu thành từ nhiều tác tử (agent) hoạt động tương đối độc lập. Mỗi tác tử có mục tiêu, tri thức và khả năng xử lý riêng, đồng thời có thể tương tác, trao đổi thông tin và phối hợp với các tác tử khác để hoàn thành mục tiêu chung của hệ thống.

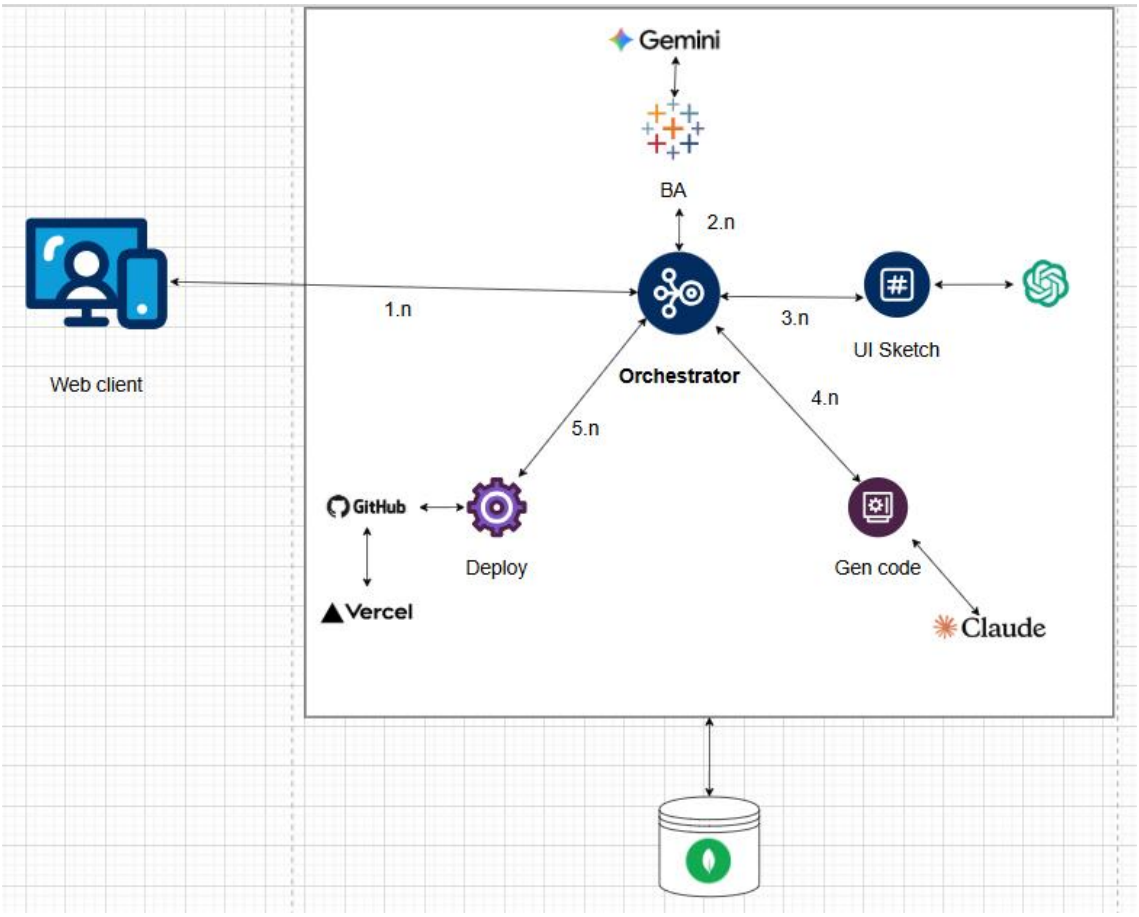
Trong đồ án này, kiến trúc đa tác tử được lựa chọn làm kiến trúc trung tâm vì quy trình phát triển ứng dụng tự động bao gồm nhiều bước có tính chất khác nhau, từ phân tích yêu cầu, thiết kế giao diện, sinh mã nguồn cho đến triển khai sản phẩm. Việc phân tách các bước này thành các tác tử chuyên biệt giúp hệ thống:

- Mô hình hóa rõ ràng quy trình phát triển phần mềm dưới dạng các vai trò độc lập.
- Tăng tính linh hoạt khi mở rộng hoặc thay đổi một giai đoạn xử lý.
- Cho phép các tác tử hoạt động bán song song và phối hợp theo ngữ cảnh.

Đánh giá lựa chọn kiến trúc

Việc lựa chọn kiến trúc đa tác tử làm kiến trúc phần mềm cốt lõi giúp hệ thống phản ánh đúng bản chất của bài toán phát triển ứng dụng tự động, trong đó tri thức và xử lý được phân tán theo vai trò. Kiến trúc này không chỉ đáp ứng tốt yêu cầu hiện tại của đồ án mà còn tạo nền tảng thuận lợi cho việc mở rộng thêm các tác tử mới trong tương lai, chẳng hạn như tác tử kiểm thử, tối ưu giao diện hoặc đánh giá trải nghiệm người dùng.

4.1.2 Thiết kế tổng quan



Hình 4.1: Sơ đồ tổng quan hệ thống

Dựa trên kiến trúc đa tác tử đã được lựa chọn, hệ thống được thiết kế theo hướng phân tách thành các tác tử độc lập, mỗi tác tử đảm nhiệm một vai trò cụ thể trong quy trình hỗ trợ phát triển giao diện web tự động. Thiết kế tổng quan của hệ thống tập trung làm rõ cách các tác tử được tổ chức, cách chúng tương tác với nhau và vai trò điều phối trung tâm của Orchestration Agent trong việc quản lý toàn bộ quy trình phát triển.

Các tác tử chính trong hệ thống bao gồm:

- **Orchestration Agent (Orches)**: tác tử điều phối trung tâm, chịu trách nhiệm quản lý toàn bộ luồng xử lý, phân phối nhiệm vụ cho các tác tử chuyên biệt và tổng hợp kết quả giữa các giai đoạn.
- **Business Analyst Agent (BA)**: tác tử thu thập, phân tích và làm rõ yêu cầu người dùng thông qua hội thoại, chuyển đổi yêu cầu tự nhiên thành đặc tả có cấu trúc.
- **SketchUI Agent**: tác tử đề xuất và phác thảo thiết kế giao diện người dùng ở mức tổng quan dựa trên yêu cầu đã được phân tích.
- **Generate Code Agent**: tác tử sinh mã nguồn giao diện và logic ứng dụng dựa trên thiết kế giao diện và đặc tả nghiệp vụ đã được xác nhận.
- **Deploy Agent**: tác tử chịu trách nhiệm đóng gói và triển khai ứng dụng, đồng thời cung cấp kết quả triển khai cho người dùng.

Về mặt tổng quan, hệ thống được tổ chức theo mô hình điều phối tập trung, trong đó Orchestration Agent đóng vai trò là điểm trung gian duy nhất kết nối các tác tử. Người dùng tương tác với hệ thống thông qua giao diện web, các yêu cầu từ người dùng được chuyển đến Orchestration Agent để xử lý. Dựa trên trạng

thái hiện tại của quy trình và ngữ cảnh phiên làm việc, Orchestration Agent lần lượt kích hoạt các tác tử phù hợp.

Khi người dùng khởi tạo một yêu cầu mới, Orchestration Agent trước tiên kích hoạt Business Analyst Agent để thu thập và phân tích yêu cầu. Trong giai đoạn này, Business Analyst Agent có thể lặp lại nhiều lần quá trình trao đổi với người dùng nhằm làm rõ, bổ sung hoặc điều chỉnh yêu cầu khi thông tin ban đầu chưa đầy đủ hoặc còn mơ hồ. Quá trình này chỉ kết thúc khi yêu cầu được chuẩn hóa thành một đặc tả có cấu trúc và đủ điều kiện để chuyển sang các bước tiếp theo.

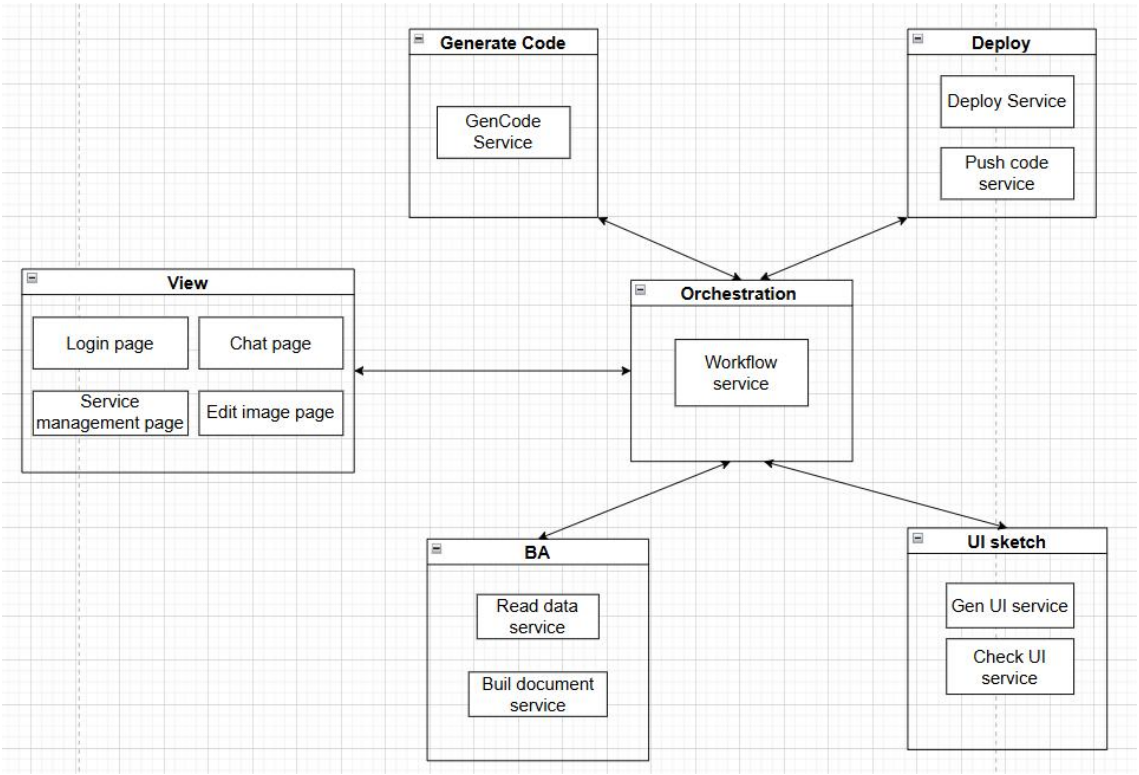
Sau khi đặc tả yêu cầu được hoàn thiện, Orchestration Agent kích hoạt SketchUI Agent để đề xuất và phác thảo thiết kế giao diện tổng quan. Kết quả phác thảo giao diện không được đưa trực tiếp cho người dùng mà trước hết được chuyển lại cho Business Analyst Agent để kiểm tra mức độ phù hợp với yêu cầu ban đầu. Trong trường hợp giao diện đề xuất chưa đáp ứng đầy đủ đặc tả nghiệp vụ, Business Analyst Agent sẽ yêu cầu SketchUI Agent hiệu chỉnh hoặc sinh lại giao diện. Quá trình này có thể lặp lại nhiều lần cho đến khi giao diện đạt yêu cầu về mặt nghiệp vụ và logic.

Chỉ khi giao diện đã được Business Analyst Agent xác nhận, Orchestration Agent mới chuyển giao kết quả này cho người dùng để xem xét và phản hồi. Sau khi người dùng chấp thuận giao diện, Orchestration Agent tiếp tục kích hoạt Generate Code Agent để sinh mã nguồn giao diện và logic ứng dụng tương ứng. Cuối cùng, Deploy Agent được gọi để thực hiện đóng gói và triển khai ứng dụng, đồng thời cung cấp đường dẫn truy cập cho người dùng.

Trong thiết kế này, các tác tử không giao tiếp trực tiếp với nhau mà chỉ trao đổi thông tin thông qua Orchestration Agent. Cách tiếp cận này giúp giảm sự phụ thuộc chặt chẽ giữa các tác tử, hạn chế xung đột thông tin và đảm bảo tính nhất quán của ngữ cảnh xuyên suốt toàn bộ quy trình. Mỗi tác tử có thể được phát triển, nâng cấp hoặc thay thế độc lập mà không làm ảnh hưởng đến các tác tử khác cũng như cấu trúc tổng thể của hệ thống.

Thiết kế tổng quan theo kiến trúc đa tác tử với cơ chế điều phối tập trung và các vòng lặp kiểm soát chất lượng ở từng giai đoạn giúp hệ thống mô hình hóa tốt quy trình phát triển giao diện web tự động. Đồng thời, thiết kế này tạo nền tảng thuận lợi cho việc mở rộng hệ thống trong tương lai, chẳng hạn như bổ sung thêm các tác tử mới hoặc cải tiến chiến lược phối hợp giữa các tác tử mà không cần thay đổi toàn bộ kiến trúc.

4.1.3 Thiết kế chi tiết gói



Hình 4.2: Thiết kế gói

Hình trên minh họa thiết kế tổng quan hệ thống theo kiến trúc đa tác tử với mô hình điều phối tập trung. Hệ thống được tổ chức thành các package chính, tương ứng với các tác tử và các thành phần giao diện, trong đó Orchestration đóng vai trò trung tâm điều phối toàn bộ luồng xử lý.

Package View chịu trách nhiệm cung cấp giao diện tương tác với người dùng, bao gồm các màn hình như đăng nhập, quản lý dịch vụ, trò chuyện và chỉnh sửa giao diện. Các yêu cầu từ người dùng được gửi đến Workflow Service trong package Orchestration, nơi quản lý trạng thái quy trình và kích hoạt các tác tử tương ứng.

Các package BA, UI Sketch, Generate Code và Deploy đại diện cho các tác tử chuyên biệt trong hệ thống. Mỗi tác tử đảm nhiệm một giai đoạn cụ thể của quy trình phát triển giao diện web tự động, từ phân tích yêu cầu, phác thảo giao diện, sinh mã nguồn cho đến triển khai ứng dụng. Các tác tử không giao tiếp trực tiếp với nhau mà chỉ trao đổi thông tin thông qua Orchestration, giúp giảm sự phụ thuộc chặt chẽ và đảm bảo tính nhất quán của luồng xử lý.

Thiết kế tổng quan này phản ánh rõ vai trò điều phối trung tâm của Orchestration Agent cũng như sự phân tách trách nhiệm giữa các tác tử, tạo nền tảng thuận lợi cho việc mở rộng, bảo trì và nâng cấp hệ thống trong tương lai.

4.2 Thiết kế chi tiết

4.2.1 Thiết kế giao diện

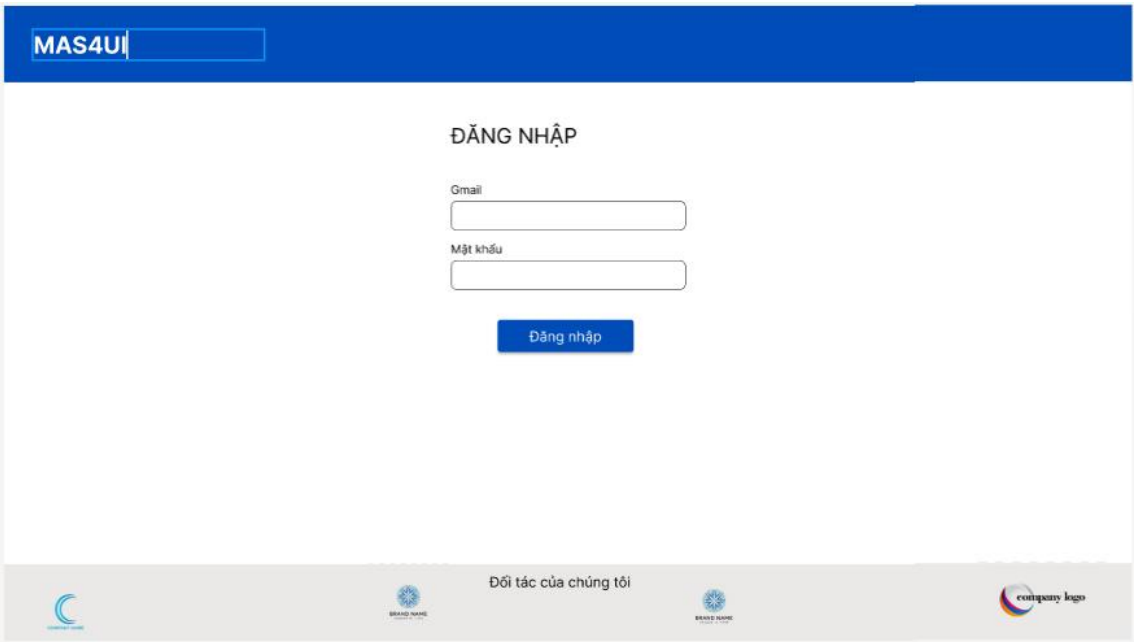
Giao diện của hệ thống được thiết kế theo hướng phân tách thành nhiều thành phần (component) độc lập, nhằm tăng khả năng tái sử dụng, mở rộng và bảo trì trong quá trình phát triển. Cách tiếp cận này giúp các thành phần giao diện có thể được sử dụng lại ở nhiều chức năng khác nhau, đồng thời giảm sự phụ thuộc chặt chẽ giữa các phần của hệ thống.

Ứng dụng web được thiết kế ưu tiên cho môi trường máy tính để bàn, với chiều rộng hiển thị dao động trong khoảng từ 1200 đến 1600 pixels, nhằm đảm bảo nội dung được trình bày đầy đủ và rõ ràng trên hầu

hết các màn hình phổ biến hiện nay. Độ phân giải mục tiêu của giao diện là 1920×1080 pixels (Full HD), giúp tối ưu trải nghiệm người dùng trong quá trình tương tác với hệ thống.

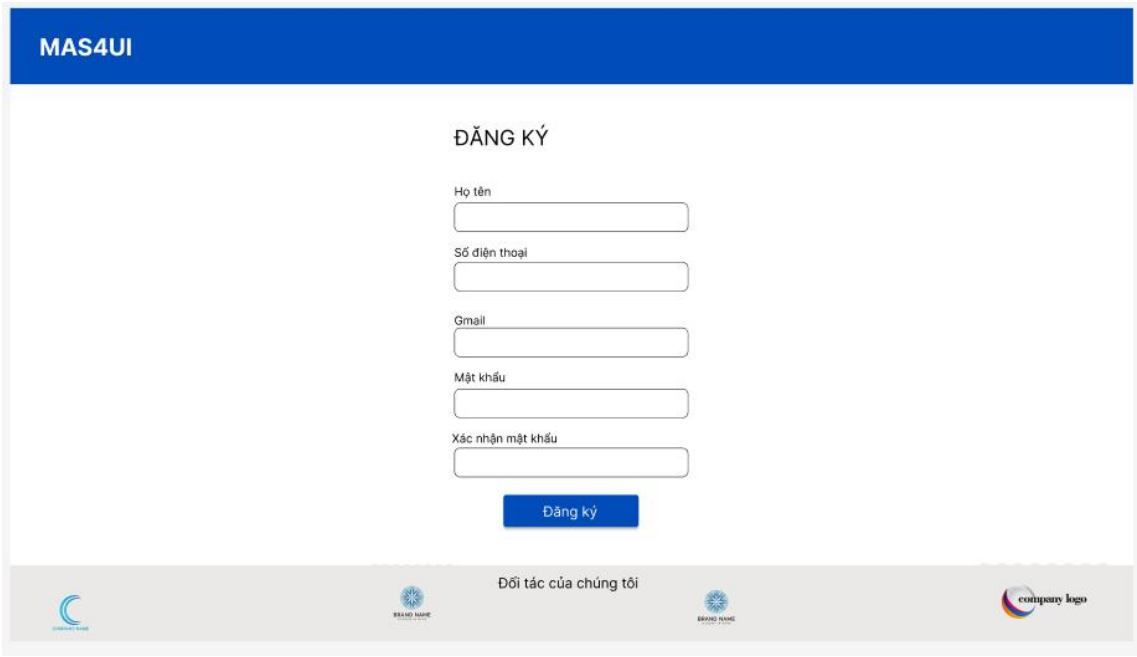
Về mặt thẩm mỹ, giao diện sử dụng tông màu chủ đạo là sự kết hợp giữa màu xanh và trắng, tạo cảm giác thân thiện, hiện đại và dễ quan sát. Cách phối màu này đồng thời giúp làm nổi bật các thành phần tương tác quan trọng, hỗ trợ người dùng dễ dàng theo dõi và sử dụng các chức năng của hệ thống.

Bố cục tổng thể của giao diện được chia thành ba phần chính, bao gồm phần đầu trang (Header), phần chân trang (Footer) và khu vực nội dung chính. Trong đó, khu vực nội dung chính bao gồm thanh điều hướng và vùng hiển thị nội dung tương ứng với từng chức năng. Cách tổ chức này giúp giao diện rõ ràng, logic và thuận tiện cho người dùng trong quá trình sử dụng. Các thiết kế giao diện chi tiết cho từng chức năng quan trọng của hệ thống sẽ được trình bày trong các mục tiếp theo.



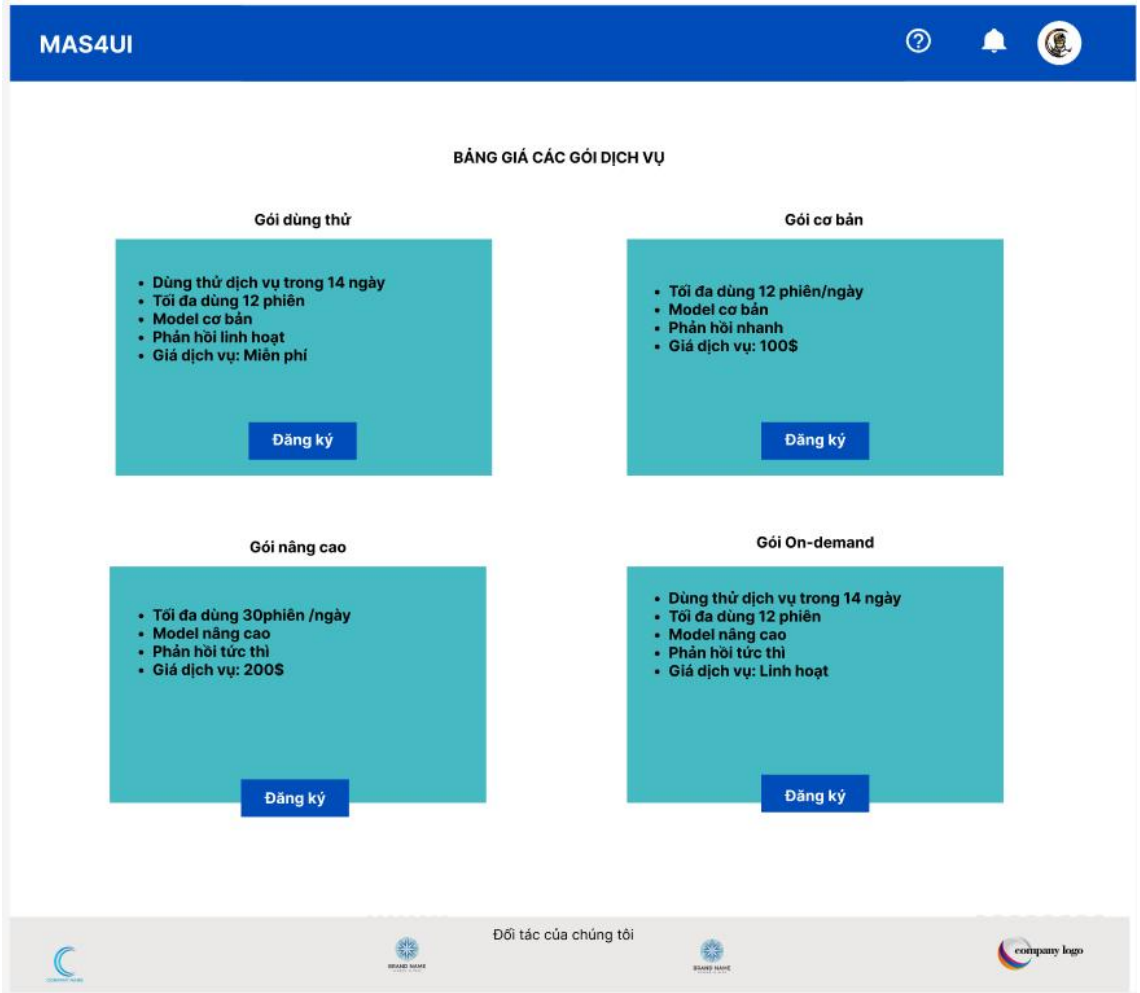
Hình 4.3: Thiết kế giao diện login

Giao diện đăng nhập gồm hai trường chính để xác thực thông tin là username và password. Người dùng cần nhập thông tin tương ứng để có thể sử dụng website



Hình 4.4: Thiết kế giao diện đăng ký

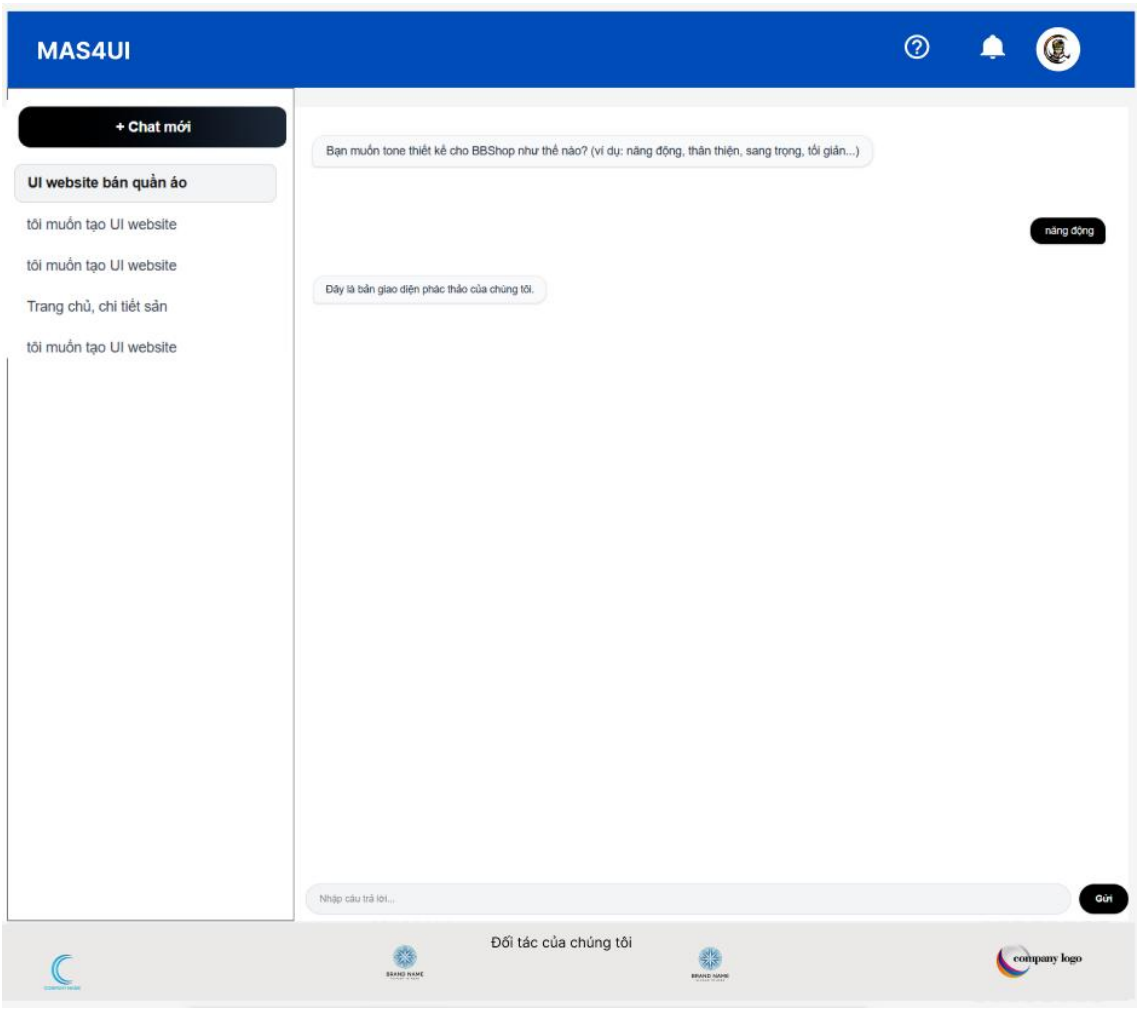
Giao diện đăng ký gồm năm trường chính để đăng ký tài khoản. Người dùng cần nhập thông tin họ tên, số điện thoại, gmail, mật khẩu để đăng ký.



Hình 4.5: Thiết kế giao diện danh sách gói

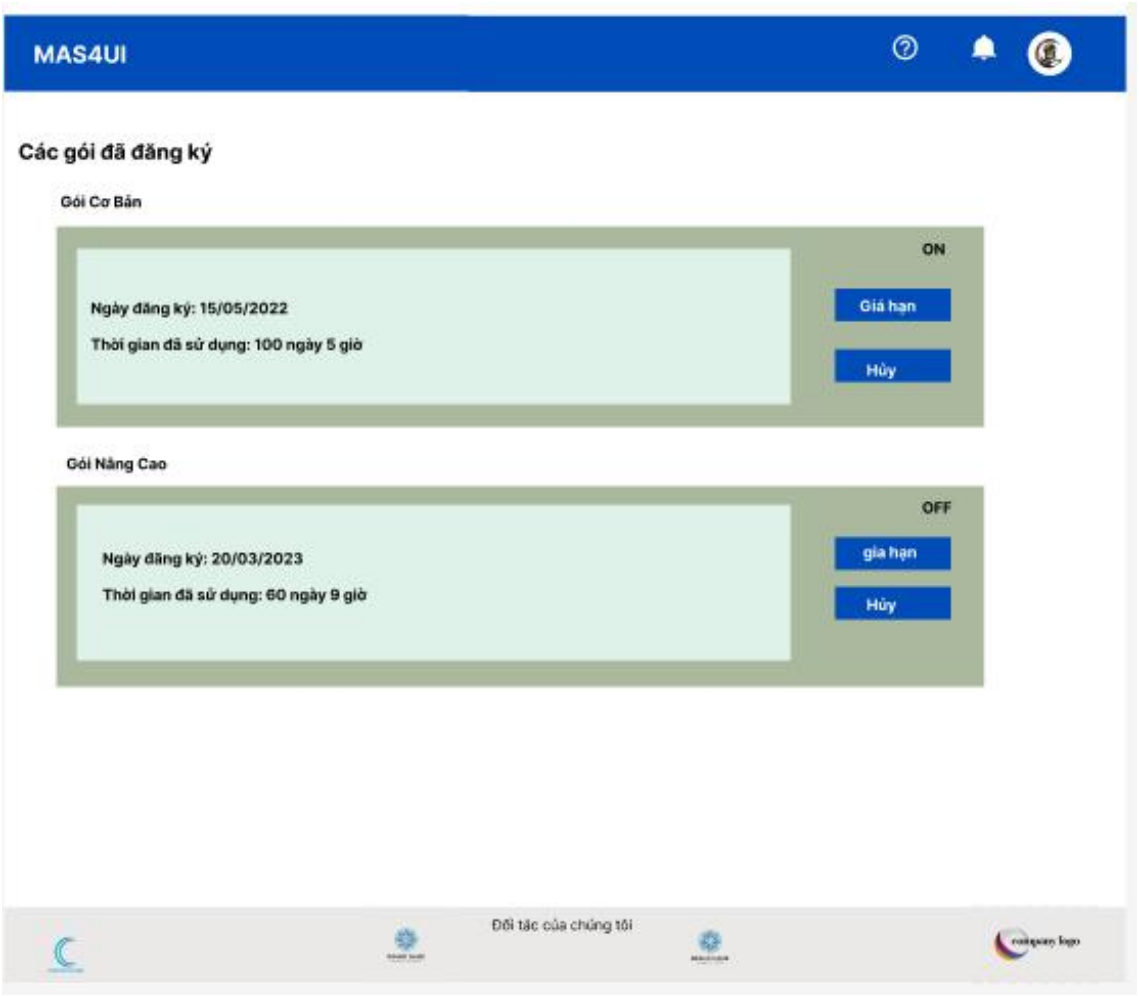
Màn hình danh sách gói hiển thị danh sách các gói dịch vụ hiện có trong hệ thống để user có thể lựa

chọn và đăng ký.



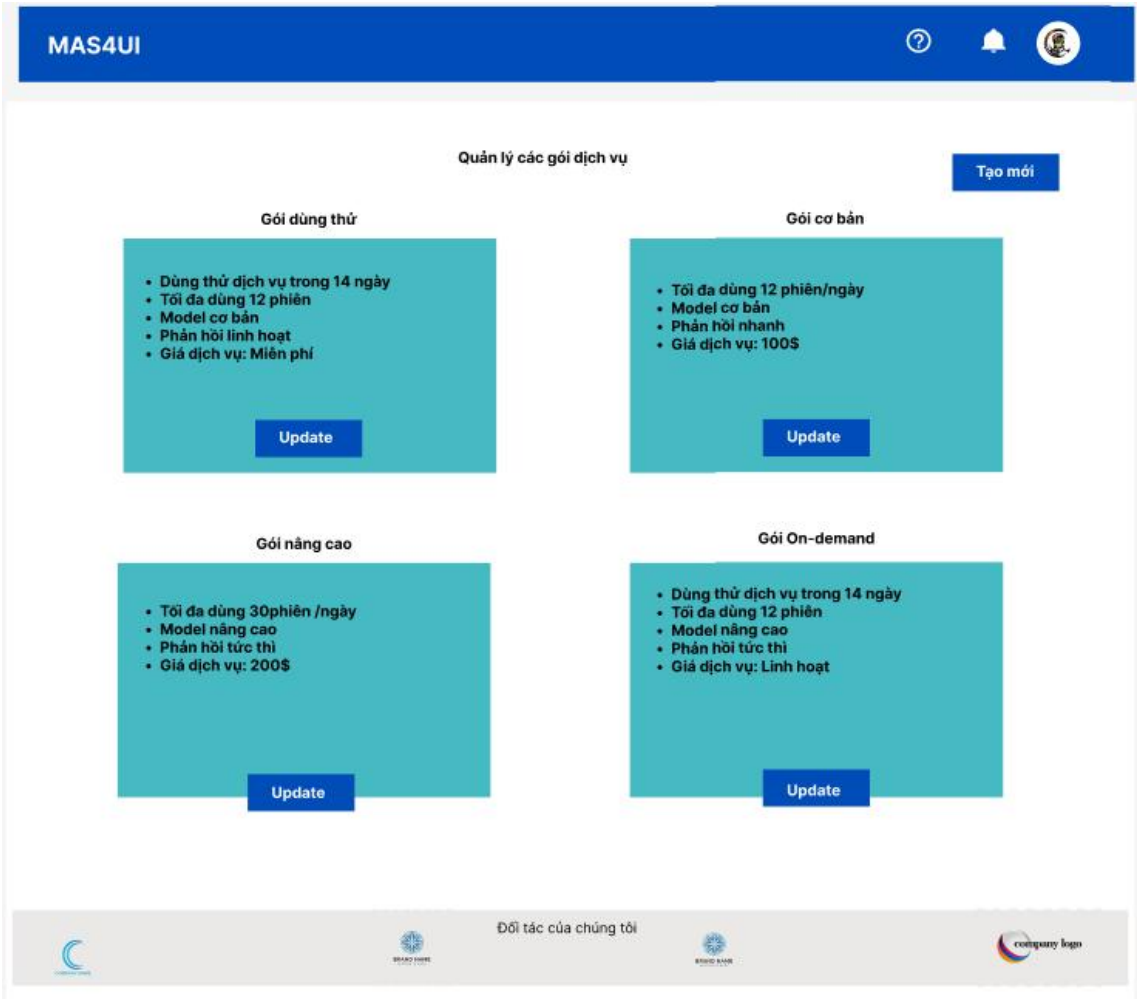
Hình 4.6: Thiết kế giao diện chat

Màn hình giao diện chat cho phép người dùng trò chuyện với chat bot để tạo yêu cầu mới để gen UI, đồng thời có thể xem lại danh sách đoạn chat cũ để xem chi tiết.



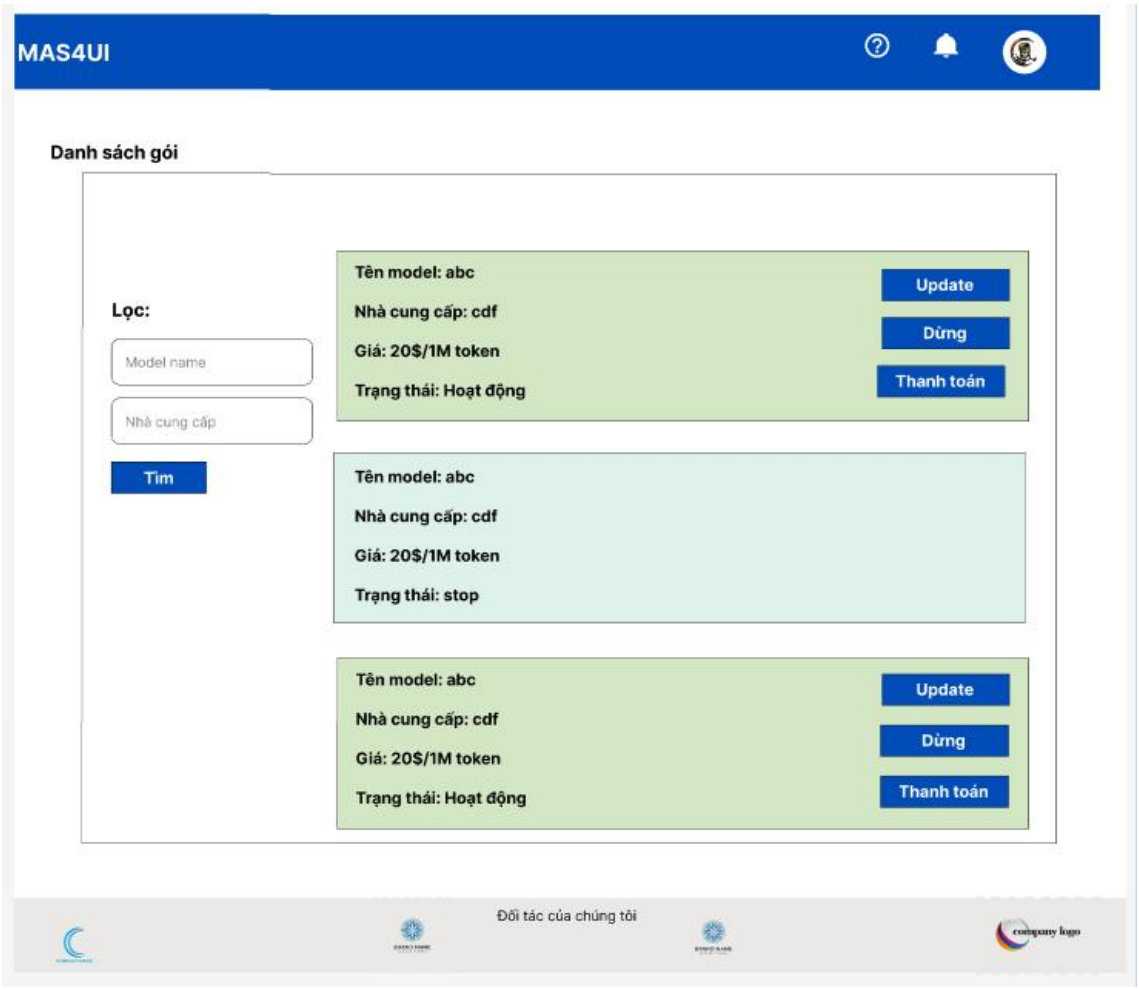
Hình 4.7: Thiết kế giao diện quản lý gói dịch vụ user

User có thể thông qua màn hinh quản lý gói dịch vụ user để xem danh sách bản thân đã đăng ký, đang sử dụng hoặc đã hủy. Đồng thời cũng có thể thao tác tương ứng trên các gói dịch vụ đang có.



Hình 4.8: Thiết kế giao diện quản lý gói dịch vụ admin

Admin có thể thông qua màn hình quản lý gói dịch vụ để xem, cập nhật những gói dịch vụ mà hệ thống đang cung cấp. Admin có thể config từng gói dịch vụ như giá, model sử dụng, giới hạn trong ngày...



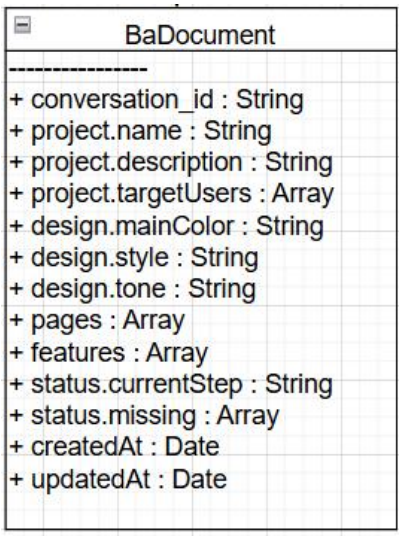
Hình 4.9: Thiết kế giao diện quản lý api key LLM

Api LLM hoạt động dựa vào hệ thống key. Tại màn hình quản lý api key LLM, admin có thể update api key cho các dịch vụ LLM hệ thống đang sử dụng.

4.2.2 Thiết kế lớp

Trong phần này em xin trình một số các thiết kế lớp chủ đạo của hệ thống: BaDocument, Conversation, Message.

Lớp BaDocument dùng để lưu trữ thông tin phân tích nghiệp vụ (Business Analysis) được xây dựng dựa trên nội dung hội thoại. Lớp này quản lý các thông tin về dự án, thiết kế giao diện, danh sách trang, chức năng và trạng thái hoàn thiện của tài liệu BA.

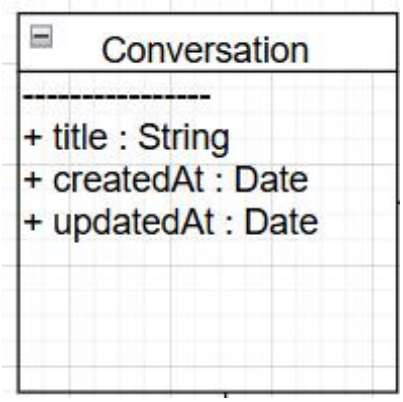


Hình 4.10: Class BaDocument

Tên trường	Kiểu dữ liệu	Ý nghĩa
conversation_id	String	Định danh cuộc hội thoại liên quan đến tài liệu BA
project.name	String	Tên dự án
project.description	String	Mô tả ngắn gọn về dự án
project.targetUsers	Array<String>	Danh sách đối tượng người dùng mục tiêu của dự án
design.mainColor	String	Màu sắc chủ đạo của giao diện hệ thống
design.style	String	Phong cách thiết kế giao diện
design.tone	String	Tông màu hoặc cảm xúc tổng thể của giao diện
pages	Array<String>	Danh sách các trang chức năng của hệ thống
features	Array<String>	Danh sách các chức năng chính của hệ thống
status.currentStep	String	Bước hiện tại trong quá trình thu thập thông tin BA
status.missing	Array<String>	Danh sách các thông tin còn thiếu cần bổ sung
createdAt	Date	Thời điểm tạo tài liệu BA
updatedAt	Date	Thời điểm cập nhật gần nhất tài liệu BA

Bảng 4.1: Bảng mô tả các thuộc tính của lớp BaDocument

Lớp Conversation dùng để quản lý thông tin tổng quát của một cuộc hội thoại trong hệ thống. Lớp này lưu trữ tiêu đề và thời gian tạo, cập nhật cuộc hội thoại, đồng thời đóng vai trò liên kết với các tin nhắn và tài liệu BA liên quan.

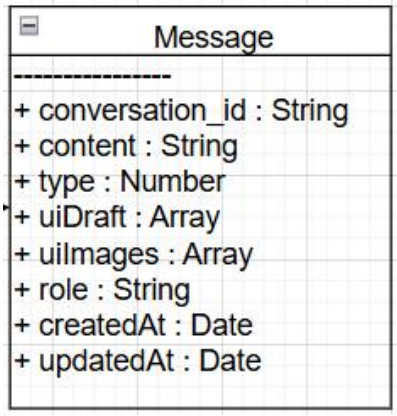


Hình 4.11: Class Conversation

Tên trường	Kiểu dữ liệu	Ý nghĩa
title	String	Tiêu đề hoặc tên của cuộc hội thoại
createdAt	Date	Thời điểm tạo cuộc hội thoại
updatedAt	Date	Thời điểm cập nhật gần nhất cuộc hội thoại

Bảng 4.2: Bảng mô tả các thuộc tính của lớp Conversation

Lớp Message đại diện cho từng tin nhắn phát sinh trong một cuộc hội thoại. Lớp này lưu trữ nội dung tin nhắn, vai trò người gửi, loại tin nhắn và các dữ liệu giao diện được sinh ra trong quá trình trao đổi.

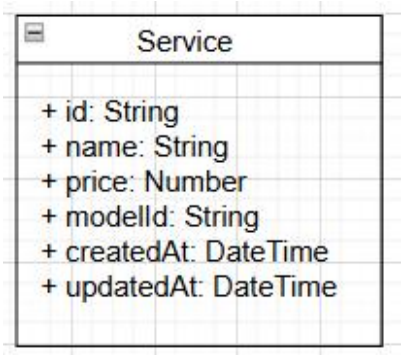


Hình 4.12: Class message

Tên trường	Kiểu dữ liệu	Ý nghĩa
conversation_id	String	Định danh cuộc hội thoại mà tin nhắn thuộc về
content	String	Nội dung tin nhắn
type	Number	Loại tin nhắn (ví dụ: văn bản, hệ thống, phản hồi AI, ...)
uiDraft	Array	Dữ liệu bản nháp giao diện được tạo trong quá trình hội thoại
uiImages	Array	Danh sách hình ảnh giao diện liên quan đến tin nhắn
role	String	Vai trò gửi tin nhắn (user, system, assistant, ...)
createdAt	Date	Thời điểm tạo tin nhắn
updatedAt	Date	Thời điểm cập nhật gần nhất tin nhắn

Bảng 4.3: Bảng mô tả các thuộc tính của lớp Message

Lớp Service đại diện cho các dịch vụ hoặc gói chức năng mà hệ thống cung cấp cho người dùng. Mỗi Service lưu trữ thông tin về tên dịch vụ, chi phí sử dụng và mô hình AI tương ứng được áp dụng. Lớp này đóng vai trò trung gian giúp hệ thống quản lý và phân loại các dịch vụ khác nhau một cách rõ ràng và linh hoạt

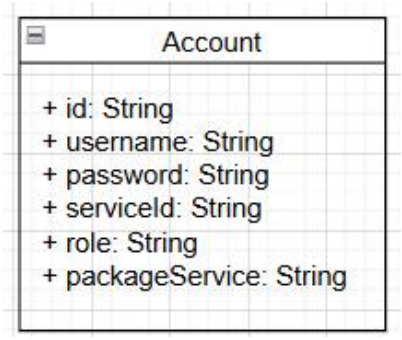


Hình 4.13: Class service

Tên trường	Kiểu dữ liệu	Ý nghĩa
id	String	Định danh duy nhất của dịch vụ trong hệ thống
name	String	Tên dịch vụ được cung cấp cho người dùng
price	Number	Giá tiền của dịch vụ hoặc gói dịch vụ
modelId	String	Định danh mô hình AI được sử dụng cho dịch vụ
createdAt	DateTime	Thời điểm tạo dịch vụ
updatedAt	DateTime	Thời điểm cập nhật thông tin dịch vụ gần nhất

Bảng 4.4: Bảng mô tả các thuộc tính của lớp Service

Lớp Account dùng để quản lý thông tin tài khoản người dùng trong hệ thống. Lớp này bao gồm các thuộc tính phục vụ cho việc xác thực, phân quyền và liên kết với dịch vụ mà người dùng đăng ký. Thông qua Account, hệ thống có thể kiểm soát quyền truy cập, gói dịch vụ đang sử dụng và đảm bảo an toàn trong quá trình vận hành.

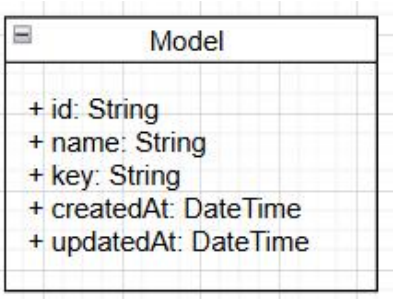


Hình 4.14: Class account

Tên trường	Kiểu dữ liệu	Ý nghĩa
id	String	Định danh duy nhất của tài khoản người dùng
username	String	Tên đăng nhập của người dùng trong hệ thống
password	String	Mật khẩu được mã hóa để xác thực người dùng
serviceId	String	Định danh dịch vụ mà tài khoản đang sử dụng
role	String	Vai trò của tài khoản (user, admin, . . .)
packageService	String	Gói dịch vụ mà tài khoản đã đăng ký

Bảng 4.5: Bảng mô tả các thuộc tính của lớp Account

Lớp Model biểu diễn các mô hình AI được tích hợp trong hệ thống. Mỗi Model lưu trữ thông tin định danh, tên mô hình và khóa truy cập cần thiết để kết nối đến dịch vụ AI tương ứng. Lớp này cho phép hệ thống quản lý nhiều mô hình khác nhau, tạo điều kiện thuận lợi cho việc mở rộng, thay thế hoặc nâng cấp mô hình trong tương lai.

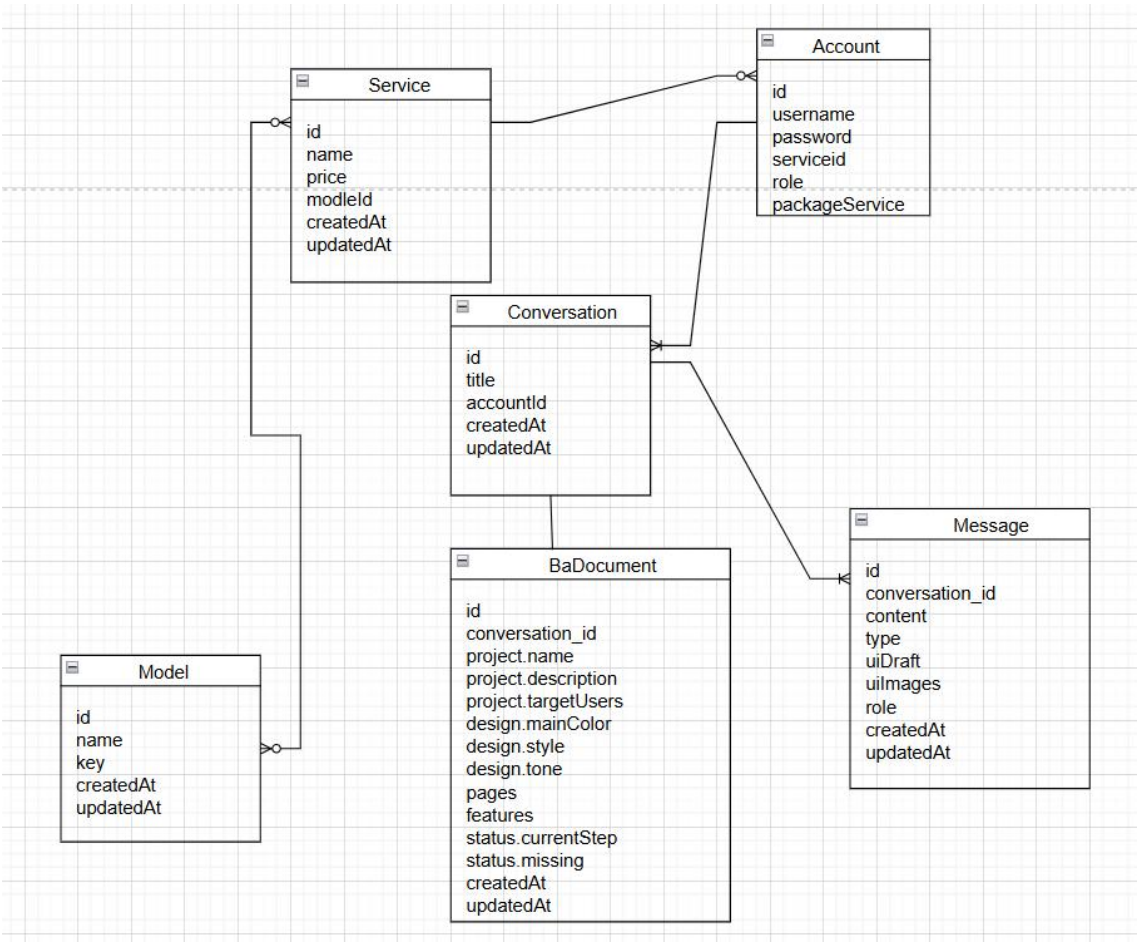


Hình 4.15: Class model

Tên trường	Kiểu dữ liệu	Ý nghĩa
id	String	Định danh duy nhất của mô hình AI
name	String	Tên mô hình AI được sử dụng trong hệ thống
key	String	Khóa truy cập (API key) của mô hình AI
createdAt	DateTime	Thời điểm tạo thông tin mô hình
updatedAt	DateTime	Thời điểm cập nhật thông tin mô hình gần nhất

Bảng 4.6: Bảng mô tả các thuộc tính của lớp Model

4.2.3 Thiết kế cơ sở dữ liệu



Hình 4.16: Thiết kế cơ sở dữ liệu

Tên trường	Kiểu dữ liệu	Ý nghĩa
id	Integer	Định danh tài khoản
username	String	Tên đăng nhập hệ thống
password	String	Mật khẩu tài khoản
serviceId	Integer	Định danh dịch vụ sử dụng
role	String	Vai trò của tài khoản trong hệ thống
packageService	String	Gói dịch vụ mà tài khoản đăng ký

Bảng 4.7: Thiết kế bảng Account

Tên trường	Kiểu dữ liệu	Ý nghĩa
id	Integer	Định danh dịch vụ
name	String	Tên dịch vụ
price	Integer	Giá của dịch vụ
modelId	Integer	Định danh mô hình AI sử dụng
createdAt	Date	Thời điểm tạo dịch vụ
updatedAt	Date	Thời điểm cập nhật dịch vụ

Bảng 4.8: Thiết kế bảng Service

Tên trường	**Kiểu dữ liệu**	**Ý nghĩa**
id	Integer	Định danh mô hình AI
name	String	Tên mô hình AI
key	String	Khóa truy cập mô hình
createdAt	Date	Thời điểm tạo mô hình
updatedAt	Date	Thời điểm cập nhật mô hình

Bảng 4.9: Thiết kế bảng Model

Tên trường	Kiểu dữ liệu	Ý nghĩa
id	Integer	Định danh cuộc hội thoại
title	String	Tiêu đề cuộc hội thoại
accountId	Integer	Định danh tài khoản sở hữu hội thoại
createdAt	Date	Thời điểm tạo hội thoại
updatedAt	Date	Thời điểm cập nhật hội thoại

Bảng 4.10: Thiết kế bảng Conversation

Tên trường	Kiểu dữ liệu	Ý nghĩa
id	Integer	Định danh tin nhắn
conversation_id	Integer	Định danh cuộc hội thoại
content	String	Nội dung tin nhắn
type	Integer	Loại tin nhắn
uiDraft	Array	Dữ liệu bản nháp giao diện
uiImages	Array	Danh sách hình ảnh giao diện
role	String	Vai trò gửi tin nhắn
createdAt	Date	Thời điểm tạo tin nhắn
updatedAt	Date	Thời điểm cập nhật tin nhắn

Bảng 4.11: Thiết kế bảng Message

Tên trường	Kiểu dữ liệu	Ý nghĩa
id	Integer	Định danh tài liệu BA
conversation_id	Integer	Định danh cuộc hội thoại liên quan
project.name	String	Tên dự án
project.description	String	Mô tả dự án
project.targetUsers	Array<String>	Đối tượng người dùng mục tiêu
design.mainColor	String	Màu sắc chủ đạo giao diện
design.style	String	Phong cách thiết kế
design.tone	String	Tông màu giao diện
pages	Array<String>	Danh sách các trang chức năng
features	Array<String>	Danh sách chức năng hệ thống
status.currentStep	String	Bước hiện tại của tài liệu BA
status.missing	Array<String>	Thông tin còn thiếu
createdAt	Date	Thời điểm tạo tài liệu BA
updatedAt	Date	Thời điểm cập nhật tài liệu BA

Bảng 4.12: Thiết kế bảng BaDocument

4.3 Xây dựng ứng dụng

4.3.1 Thư viện và công cụ sử dụng

Mục đích	Công cụ	Phiên bản	Địa chỉ URL
IDE lập trình	Visual Studio Code	1.85.1	https://code.visualstudio.com/
Ngôn ngữ lập trình	JavaScript	ES6	https://developer.mozilla.org/en-US/docs/Web/JavaScript
Cơ sở dữ liệu	MongoDB	6.0	https://www.mongodb.com/
Thiết kế giao diện ứng dụng	Figma	Web v2024	https://www.figma.com/
Quản lý mã nguồn	Git	2.44.0	https://git-scm.com/
Triển khai ứng dụng	Vercel	v2	https://vercel.com/
Soạn thảo báo cáo	Overleaf	2024	https://www.overleaf.com/

Bảng 4.13: Bảng mô tả thư viện và công nghệ sử dụng

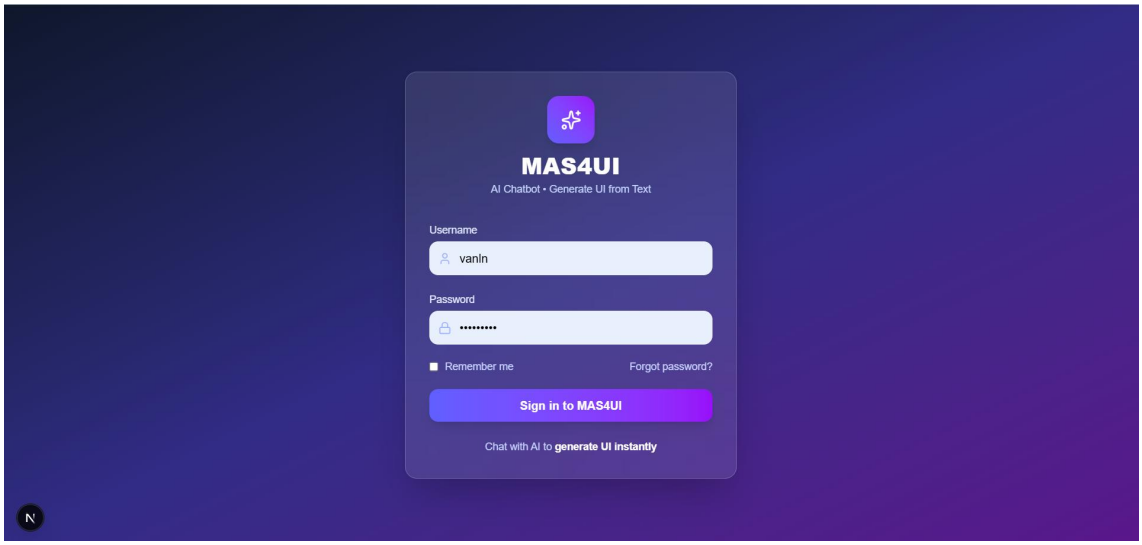
4.3.2 Kết quả đạt được

Trong quá trình thực hiện đồ án, sinh viên đã xây dựng và triển khai thành công Hệ thống Đa Tác Tử Hỗ trợ Phát triển Giao Diện Web. Hệ thống được thiết kế nhằm hỗ trợ người dùng trong việc phát triển giao diện web theo hướng tự động hóa, thông qua việc mô tả yêu cầu bằng ngôn ngữ tự nhiên và tương tác từng bước trong suốt quy trình phát triển.

Hệ thống bao gồm một giao diện web cho phép người dùng tạo và quản lý các yêu cầu phát triển giao diện, kết hợp với kiến trúc đa tác tử ở phía xử lý. Mỗi tác tử đảm nhiệm một vai trò cụ thể như phân tích yêu cầu, phác thảo giao diện, sinh mã nguồn và triển khai ứng dụng. Các tác tử này được điều phối tập trung thông qua Orchestration Agent, đảm bảo luồng xử lý thống nhất, linh hoạt và có khả năng lặp lại khi yêu cầu chưa được làm rõ hoặc chưa đạt mức mong muốn.

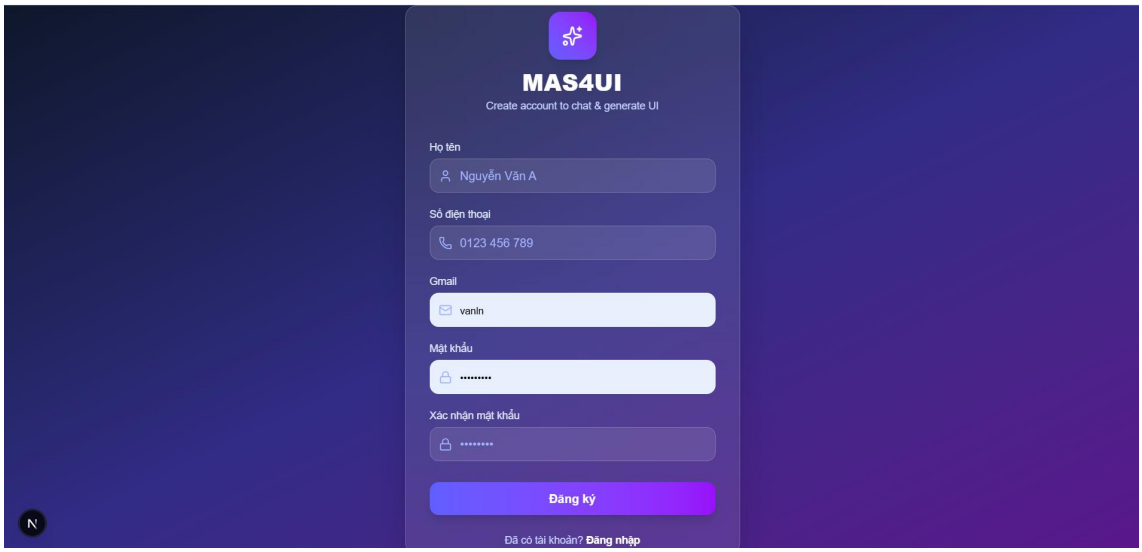
Bên cạnh việc thiết kế và hiện thực kiến trúc hệ thống, sinh viên cũng đã tiến hành thử nghiệm hệ thống với nhiều kịch bản khác nhau. Kết quả cho thấy hệ thống hoạt động ổn định, các tác tử phối hợp hiệu quả trong việc hỗ trợ phát triển giao diện web và giúp giảm đáng kể thời gian cũng như công sức so với quy trình phát triển truyền thống. Những kết quả đạt được bước đầu khẳng định tính khả thi và tiềm năng ứng dụng của hệ thống trong thực tế.

4.3.3 Minh họa các chức năng chính



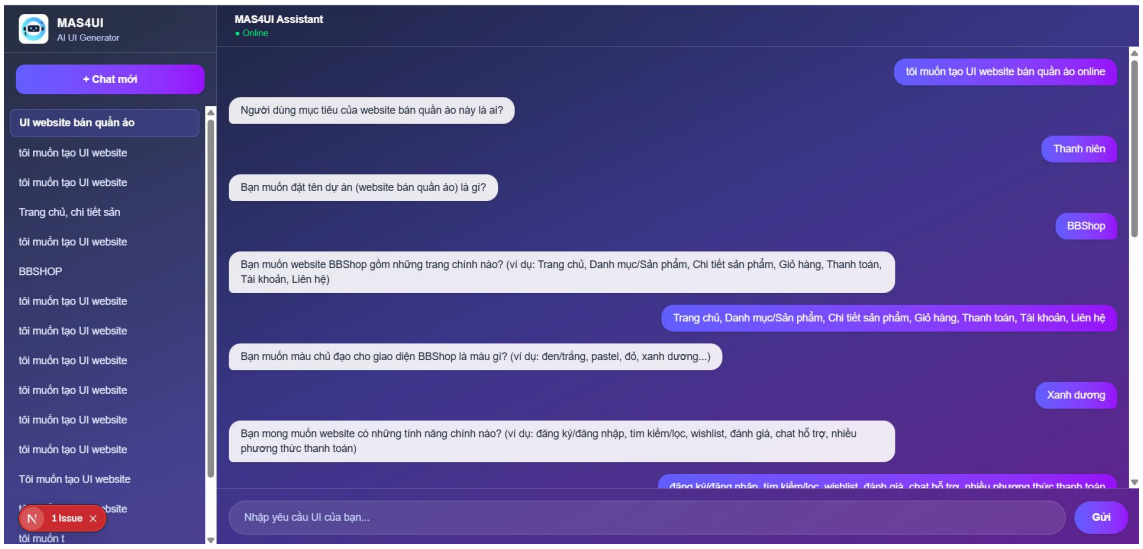
Hình 4.17: Màn hình đăng nhập

Trên đây là màn hình đăng nhập chung cho cả người dùng và admin. Sau đó tùy vào quyền của từ đối tượng sẽ được đưa đến các trang khác nhau.



Hình 4.18: Màn hình đăng ký

Màn hình đăng ký cho phép người dùng tạo tài khoản miễn phí để vào hệ thống.

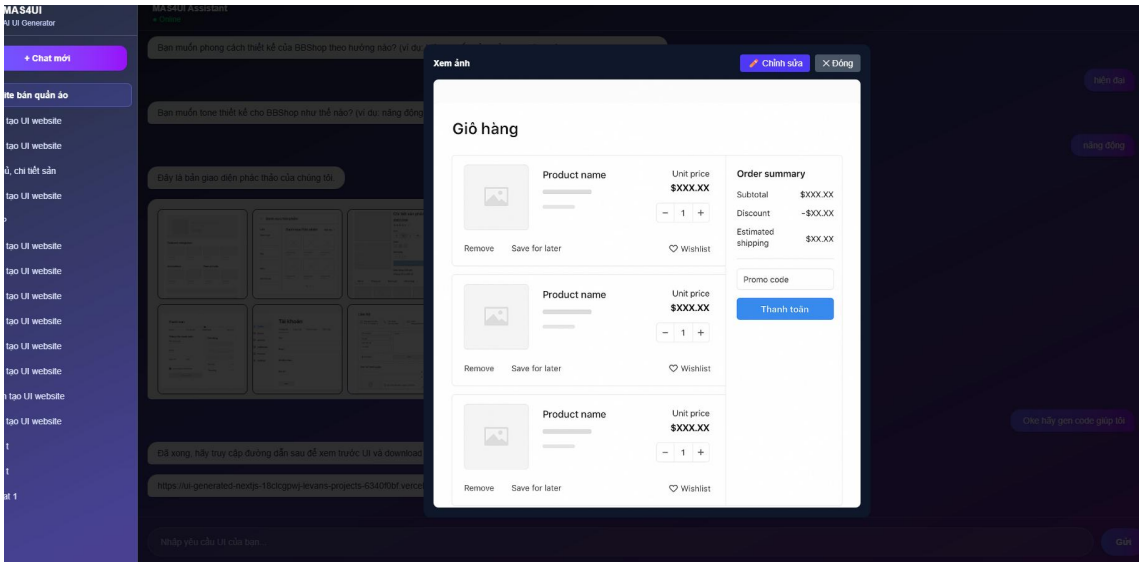


Hình 4.19: Màn hình chat

Hình minh họa thể hiện giao diện chính của Hệ thống Đa Tác Tử hỗ trợ phát triển giao diện web. Giao diện được thiết kế dưới dạng ứng dụng web, gồm thanh điều hướng bên trái và khu vực nội dung chính ở trung tâm, giúp người dùng dễ dàng theo dõi và tương tác với hệ thống.

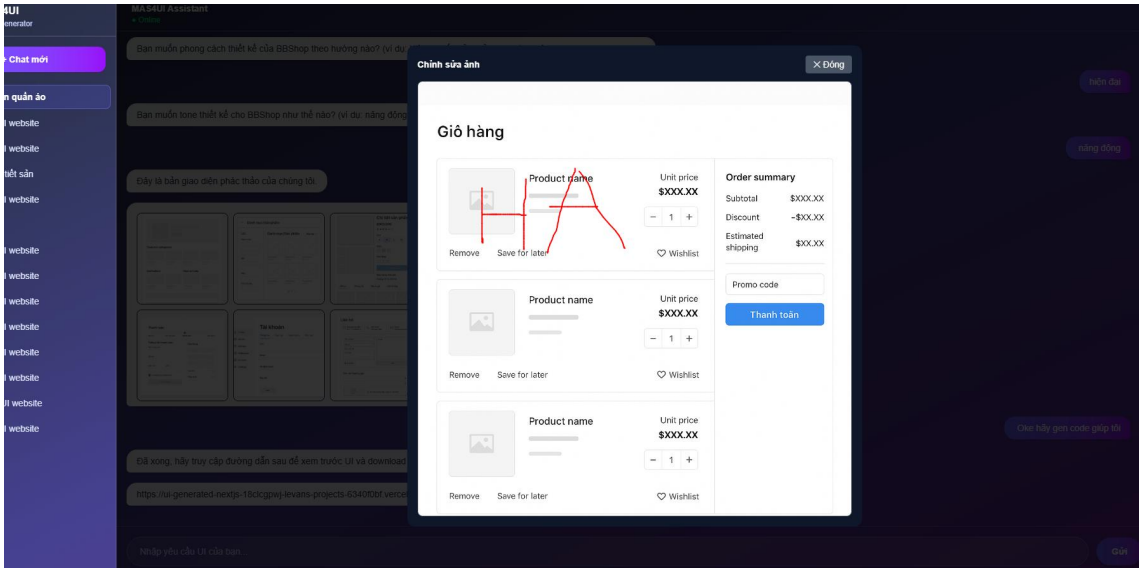
Thanh điều hướng bên trái cho phép người dùng quản lý các phiên làm việc, tạo yêu cầu mới và truy cập lại lịch sử trao đổi trước đó. Khu vực trung tâm hiển thị nội dung hội thoại giữa người dùng và hệ thống, đồng thời trình bày kết quả xử lý của các tác tử. Trong ví dụ, hệ thống sinh ra các bản phác thảo giao diện (UI sketch) cho một website, giúp người dùng hình dung tổng thể bố cục và các màn hình chức năng chính.

Sau khi hoàn tất bước sinh giao diện, hệ thống cung cấp đường dẫn triển khai để người dùng xem trước giao diện web đã được tạo. Khu vực nhập liệu phía dưới cho phép người dùng tiếp tục chỉnh sửa yêu cầu hoặc chuyển sang các bước tiếp theo như sinh mã nguồn và triển khai, phản ánh rõ quy trình phối hợp giữa các tác tử trong hệ thống.



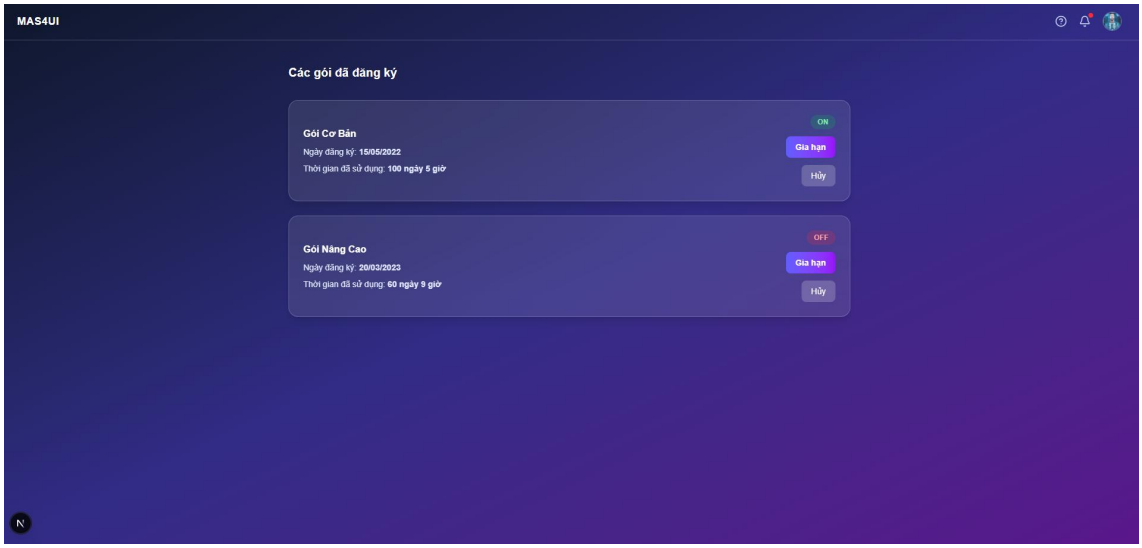
Hình 4.20: Xem chi tiết UI sketch

Tại màn hình chat user có thể click UI sketch được sinh ra để phóng to và xem chi tiết.



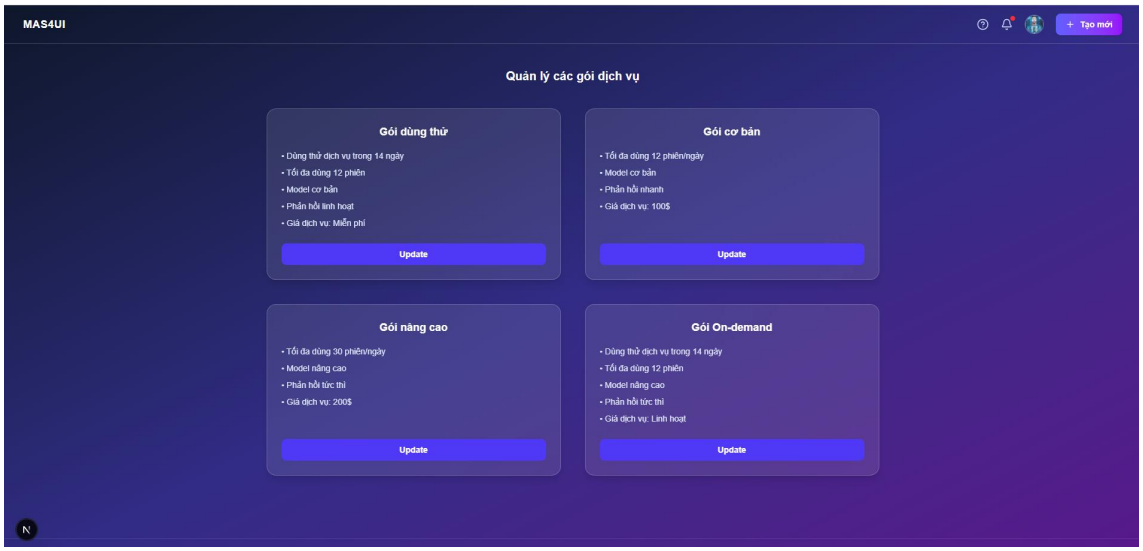
Hình 4.21: Màn hình chỉnh sửa UI sketch

Trong màn hình xem chi tiết UI sketch, user có thể chọn chỉnh sửa và update UI sketch để sử dụng cho giai đoạn tiếp theo.



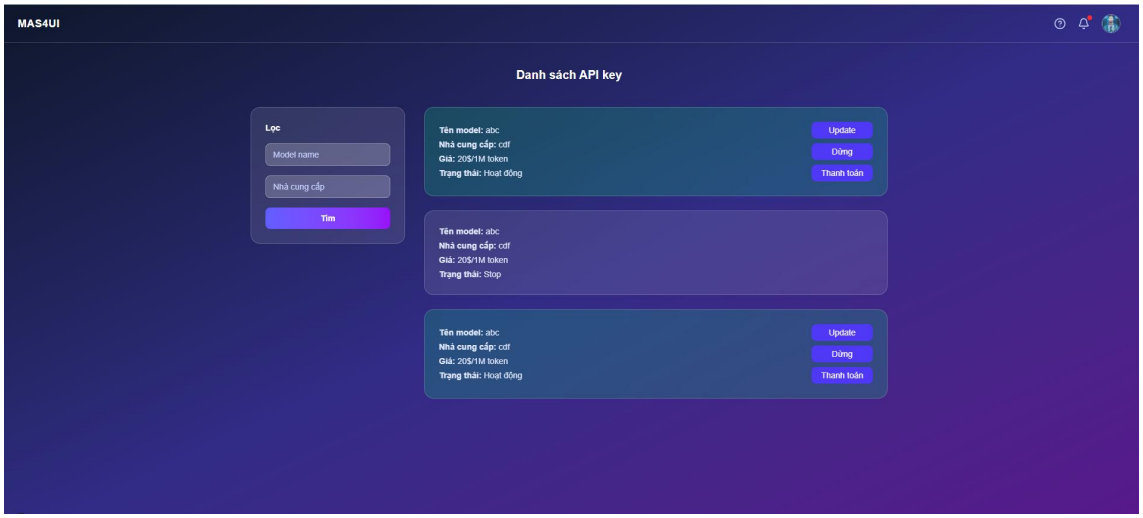
Hình 4.22: Màn hình quản lý gói dịch vụ cho user

Màn hình quản lý gói dịch vụ trên cho phép user có thể quản lý gói dịch vụ mình đang sử dụng. User có nhiều lựa chọn như hủy đăng ký, gia hạn...



Hình 4.23: Màn hình quản lý gói cho admin

Khác với màn hình quản lý gói của user, đối với admin màn hình cho phép cập nhật cấu hình của các gói dịch vụ. admin có thể quy định model nào được sử dụng cho gói, giá của gói...

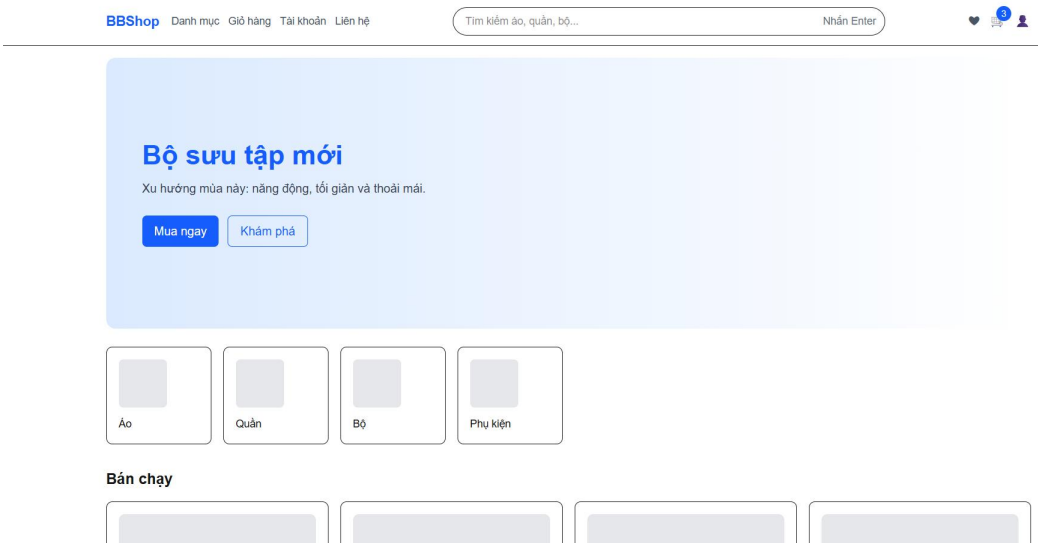


Hình 4.24: Màn hình quản lý api key LLM

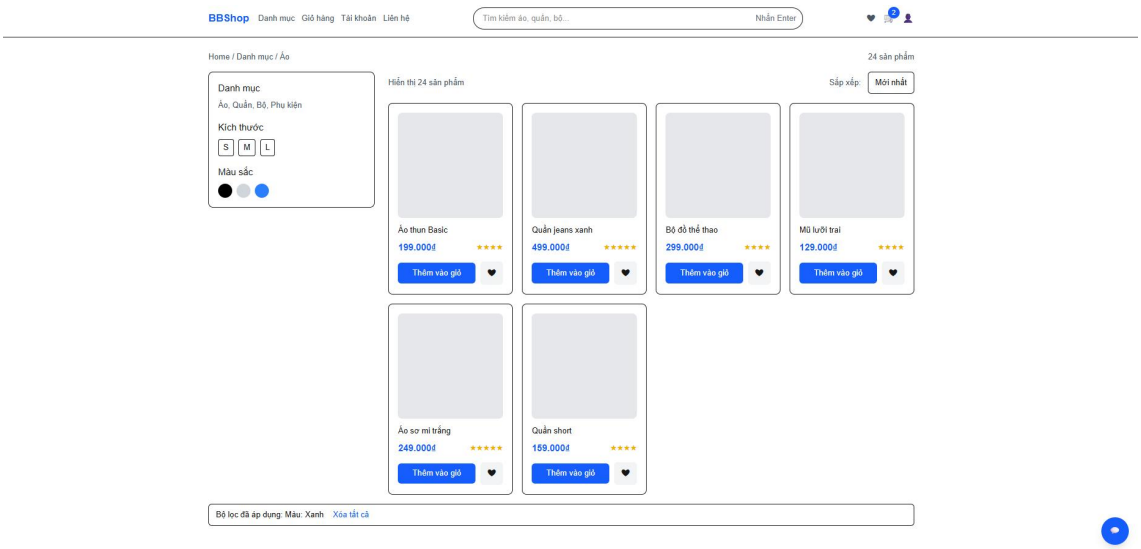
Cuối cùng, hệ thống có sử dụng dịch vụ LLM của các nhà cung cấp, vì thế cần cung cấp màn hình này cho admin để có thể quản lý key, quản lý quá trình sử dụng dịch vụ của các nhà cung cấp kia.

4.3.4 Minh họa một sản phẩm của hệ thống

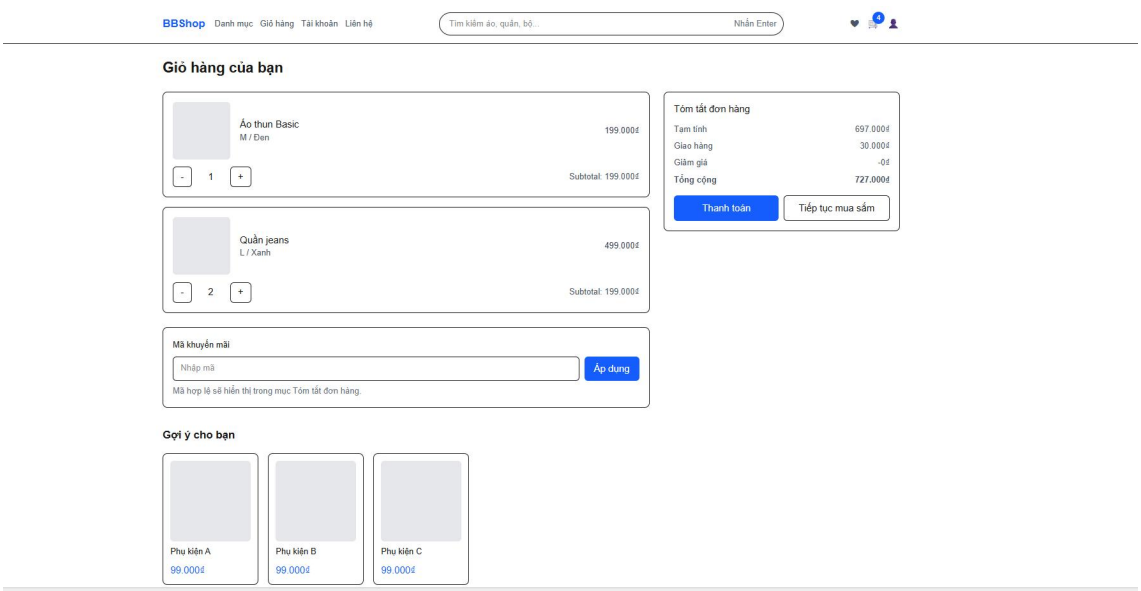
Sản phẩm được sinh ra từ hệ thống và đã được deploy tại đường dẫn: https://ui-generated-nextjs-18clcgpwj-levans-projects-6340f0bf.vercel.app



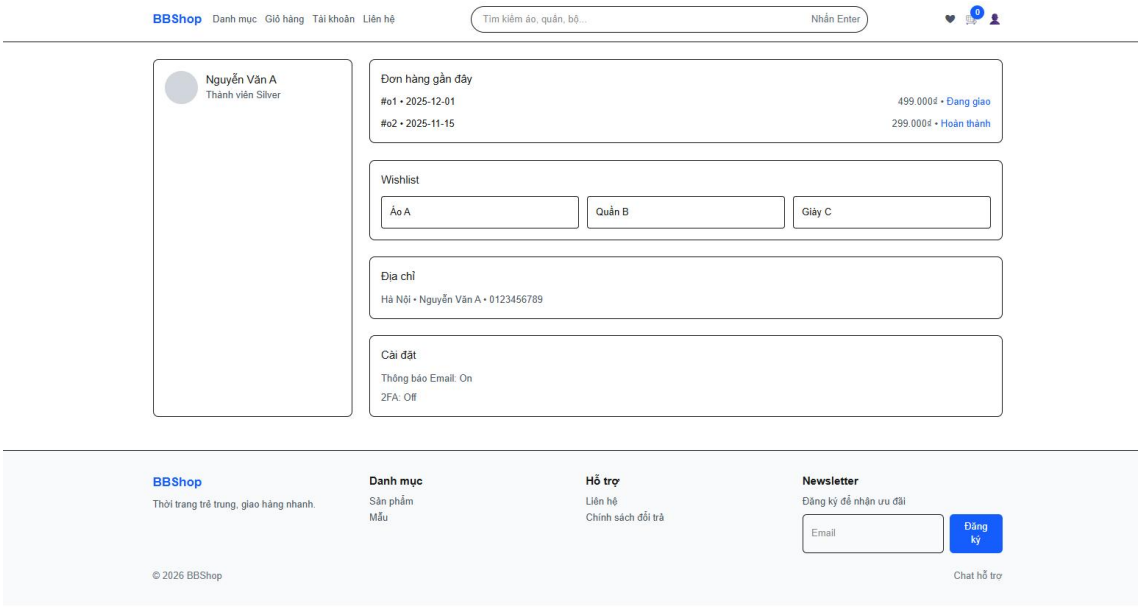
Hình 4.25: Màn hình trang chủ



Hình 4.26: Màn hình danh sách sản phẩm



Hình 4.27: Màn hình giỏ hàng



Hình 4.28: Màn hình thông tin người dùng

4.4 Kiểm thử

Nhóm chức năng	STT	Tên test case	Kết quả mong muốn	Trạng thái
Đăng nhập	1	Kiểm tra hiển thị giao diện	Hiển thị giao diện	Đạt
	2	Bỏ trống một trong các trường của form đăng nhập	Hiển thị thông báo lỗi	Đạt
	3	Đăng nhập với tài khoản và mật khẩu đúng	Đăng nhập thành công	Đạt
	4	Đăng nhập với tài khoản hoặc mật khẩu sai	Hiển thị thông báo lỗi	Đạt
Tạo mới yêu cầu	1	Yêu cầu thiếu thông tin	Hệ thống yêu cầu bổ sung	Đạt
	2	Yêu cầu đủ thông tin	Xác nhận sinh UI sketch	Đạt
Sinh UI sketch	1	Sinh UI từ yêu cầu	UI đúng mô tả	Đạt
	2	Sinh lại UI khi chỉnh sửa	UI cập nhật đúng	Đạt
Sinh code	1	Sinh mã từ UI	Mã đúng cấu trúc	Đạt
	2	Kiểm tra nhất quán UI–code	Mã phản ánh đúng UI	Đạt
	3	Sinh lại mã khi UI đổi	Mã cập nhật đúng	Đạt
Xem trước/đánh giá	1	Xem trước giao diện	Hiển thị đúng	Đạt
	2	Đánh giá tốc độ phản hồi	Đáp ứng tốt	Đạt
	3	Đánh giá độ ổn định	Hoạt động ổn định	Đạt

Bảng 4.14: Bảng kiểm thử các chức năng chính của hệ thống

4.5 Triển khai

Em đã triển khai thử nghiệm hệ thống trong môi trường phát triển nội bộ và nhờ sự hỗ trợ của một nhóm người dùng để đánh giá hiệu quả trong quá trình thiết kế giao diện web. Các người dùng tham gia thử nghiệm bằng cách nhập yêu cầu mô tả giao diện và tương tác trực tiếp với hệ thống trong nhiều kịch bản khác nhau.

Về mức độ chính xác, hệ thống đa tác tử hoạt động ổn định trong việc phân tích yêu cầu người dùng và sinh ra các bản phác thảo giao diện phù hợp. Các giao diện được tạo ra phản ánh đúng bố cục, chức năng và luồng tương tác theo mô tả ban đầu, không xảy ra tình trạng thiếu thành phần giao diện hoặc sinh sai cấu trúc màn hình.

Về tốc độ xử lý, hệ thống cho kết quả khá tốt khi thời gian từ lúc người dùng gửi yêu cầu đến khi nhận được bản phác thảo giao diện và mã nguồn mẫu ở mức ngắn, đáp ứng được nhu cầu thiết kế nhanh trong giai đoạn đầu của quá trình phát triển. Trong suốt quá trình thử nghiệm, hệ thống duy trì được tính ổn định và khả năng phản hồi liên tục.

Tuy nhiên, do phạm vi thử nghiệm còn giới hạn và số lượng người dùng đồng thời chưa lớn, khả năng mở rộng và chịu tải của hệ thống chưa được đánh giá đầy đủ. Trong tương lai, hệ thống cần được triển khai với quy mô lớn hơn và tích hợp thêm các tác tử chuyên sâu nhằm nâng cao hiệu quả phối hợp và khả năng ứng dụng thực tế.

CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

5.1 Đặt vấn đề và động cơ nghiên cứu

Trong những năm gần đây, nhu cầu phát triển giao diện web ngày càng gia tăng, đặc biệt trong bối cảnh các doanh nghiệp và cá nhân mong muốn nhanh chóng xây dựng sản phẩm với chi phí và thời gian tối ưu. Tuy nhiên, quá trình phát triển giao diện web truyền thống vẫn đòi hỏi nhiều công đoạn thủ công, từ thu thập yêu cầu, phân tích nghiệp vụ, thiết kế giao diện cho đến hiện thực mã nguồn và triển khai. Các công cụ hỗ trợ hiện nay chủ yếu tập trung vào một hoặc một vài khía cạnh riêng lẻ, thiếu sự phối hợp tổng thể và chưa cho phép người dùng kiểm soát linh hoạt toàn bộ quy trình.

Xuất phát từ thực tế đó, đề tài tập trung nghiên cứu và đề xuất một hệ thống hỗ trợ phát triển giao diện web dựa trên kiến trúc đa tác tử. Mục tiêu chính không chỉ là tự động hóa quá trình sinh giao diện, mà còn đảm bảo tính linh hoạt, khả năng mở rộng và đặc biệt là cho phép người dùng can thiệp, điều chỉnh ở từng giai đoạn của quy trình sinh. Đây chính là động cơ cốt lõi dẫn đến việc xây dựng các giải pháp và đóng góp được trình bày trong chương này.

5.2 Giải pháp kiến trúc đa tác tử cho bài toán phát triển giao diện web

5.2.1 Bài toán đặt ra

Bài toán phát triển giao diện web tự động có tính phức tạp cao do liên quan đến nhiều giai đoạn khác nhau, mỗi giai đoạn yêu cầu kiến thức và kỹ năng riêng biệt. Nếu tiếp cận theo hướng một tác nhân đơn lẻ hoặc một mô hình xử lý tuyến tính, hệ thống sẽ khó mở rộng, khó bảo trì và thiếu khả năng kiểm soát chi tiết.

5.2.2 Giải pháp đề xuất

Đề tài đề xuất một kiến trúc đa tác tử, trong đó hệ thống được chia thành nhiều tác tử độc lập, mỗi tác tử đảm nhiệm một vai trò và nhiệm vụ cụ thể trong quy trình phát triển giao diện web. Các tác tử chính bao gồm tác tử phân tích yêu cầu, tác tử thiết kế giao diện, tác tử sinh mã và tác tử triển khai. Mỗi tác tử được thiết kế như một đơn vị xử lý chuyên biệt, tập trung giải quyết một lớp bài toán rõ ràng.

5.2.3 Kết quả đạt được

Giải pháp này giúp hệ thống trở nên linh hoạt hơn, dễ mở rộng và dễ bảo trì. Việc phân tách nhiệm vụ rõ ràng cho từng tác tử giúp giảm độ phức tạp tổng thể, đồng thời cho phép nâng cấp hoặc thay thế từng tác tử mà không ảnh hưởng đến toàn bộ hệ thống.

5.3 Giải pháp điều phối tập trung và kiểm soát luồng sinh

5.3.1 Bài toán đặt ra

Trong hệ thống đa tác tử, nếu các tác tử giao tiếp trực tiếp với nhau, hệ thống sẽ dễ rơi vào tình trạng phụ thuộc chặt chẽ, khó kiểm soát luồng xử lý và dễ phát sinh xung đột ngữ cảnh.

5.3.2 Giải pháp đề xuất

Đề tài lựa chọn mô hình điều phối tập trung thông qua Orchestration Agent. Tác tử này đóng vai trò trung gian duy nhất, chịu trách nhiệm quản lý trạng thái quy trình, phân phối nhiệm vụ cho các tác tử chuyên biệt và tổng hợp kết quả giữa các giai đoạn. Mọi trao đổi thông tin đều được thực hiện thông qua tác tử điều phối, thay vì giao tiếp trực tiếp giữa các tác tử.

5.3.3 Kết quả đạt được

Giải pháp điều phối tập trung giúp đảm bảo tính nhất quán của ngữ cảnh xuyên suốt toàn bộ quy trình sinh giao diện. Đồng thời, hệ thống tránh được các xung đột thông tin, giảm sự phụ thuộc giữa các tác tử và tăng khả năng kiểm soát luồng xử lý.

5.4 Giải pháp cho phép người dùng can thiệp từng bước trong quá trình sinh

5.4.1 Bài toán đặt ra

Một hạn chế lớn của nhiều hệ thống sinh tự động hiện nay là người dùng chỉ có thể nhận kết quả cuối cùng, trong khi không có khả năng điều chỉnh hoặc phản hồi ở các bước trung gian. Điều này làm giảm tính thực tiễn của hệ thống trong các bài toán yêu cầu độ chính xác và tùy biến cao.

5.4.2 Giải pháp đề xuất

Hệ thống trong đề tài được thiết kế theo hướng cho phép người dùng can thiệp vào từng giai đoạn của quy trình sinh. Cụ thể, sau khi tác tử phân tích yêu cầu hoàn thành, người dùng có thể kiểm tra và bổ sung thông tin nếu cần. Tương tự, kết quả phác thảo giao diện từ SketchUI Agent sẽ được đưa cho người dùng hoặc tác tử phân tích nghiệp vụ xác nhận trước khi chuyển sang giai đoạn sinh mã.

5.4.3 Kết quả đạt được

Giải pháp này giúp nâng cao độ chính xác của kết quả cuối cùng, đồng thời tạo cảm giác kiểm soát và tin cậy cho người dùng. Hệ thống vừa đảm bảo tính tự động hóa, vừa giữ được sự linh hoạt cần thiết trong các dự án thực tế.

5.5 Đóng góp về bộ Prompt và chuẩn hóa tương tác tác tử

5.5.1 Bài toán đặt ra

Trong các hệ thống dựa trên tác tử thông minh, chất lượng tương tác giữa tác tử và mô hình xử lý phụ thuộc rất lớn vào cách xây dựng prompt. Prompt thiếu cấu trúc hoặc không nhất quán sẽ dẫn đến kết quả sinh không ổn định.

5.5.2 Giải pháp đề xuất

Một đóng góp quan trọng của đề tài là xây dựng bộ prompt chuẩn hóa cho từng tác tử, tương ứng với vai trò và nhiệm vụ cụ thể. Các prompt được thiết kế có cấu trúc rõ ràng, bao gồm mục tiêu, ngữ cảnh đầu vào, ràng buộc và định dạng đầu ra. Đồng thời, prompt cũng được xây dựng theo hướng dễ điều chỉnh để phù hợp với các yêu cầu khác nhau của người dùng.

5.5.3 Kết quả đạt được

Bộ prompt chuẩn hóa giúp nâng cao tính ổn định và nhất quán của kết quả sinh, đồng thời giảm đáng kể sai lệch giữa các lần xử lý. Đây là một đóng góp có thể tái sử dụng và mở rộng cho các hệ thống đa tác tử tương tự trong tương lai.

5.6 Tổng kết các đóng góp chính

Tổng hợp lại, chương này đã trình bày các đóng góp quan trọng của đề tài, bao gồm việc đề xuất kiến trúc đa tác tử cho bài toán phát triển giao diện web, thiết kế cơ chế điều phối tập trung, xây dựng quy trình cho phép người dùng can thiệp từng bước và phát triển bộ prompt chuẩn hóa cho các tác tử. Những đóng góp này không chỉ giải quyết hiệu quả bài toán đặt ra mà còn tạo nền tảng cho việc mở rộng và phát triển các hệ thống hỗ trợ phát triển phần mềm tự động trong tương lai.

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Trong quá trình thực hiện Đồ Án Tốt Nghiệp (ĐATN), đề tài *"Hệ thống Đa Tác Tử Hỗ trợ Phát triển Giao Diện Web"* đã đạt được những kết quả quan trọng cả về mặt nghiên cứu và triển khai thực nghiệm. So với các công cụ hỗ trợ phát triển giao diện web truyền thống, hệ thống được xây dựng trong đề tài đã thể hiện sự khác biệt rõ rệt thông qua việc áp dụng kiến trúc đa tác tử, cho phép tự động hóa nhiều giai đoạn trong quy trình phát triển giao diện từ phân tích yêu cầu, thiết kế giao diện đến sinh mã và triển khai ứng dụng.

Trong suốt quá trình thực hiện, em đã hoàn thành các công việc chính bao gồm: phân tích bài toán và yêu cầu của hệ thống hỗ trợ phát triển giao diện web, thiết kế kiến trúc hệ thống đa tác tử với tác tử điều phối trung tâm và các tác tử chuyên biệt, xây dựng quy trình xử lý từ tiếp nhận yêu cầu người dùng đến sinh giao diện và mã nguồn, đồng thời triển khai thử nghiệm hệ thống trên môi trường thực tế. Một trong những đóng góp nổi bật của đề tài là việc đề xuất và hiện thực hóa mô hình phối hợp giữa các tác tử, giúp hệ thống có khả năng mở rộng, linh hoạt và giảm đáng kể sự can thiệp thủ công trong quá trình phát triển giao diện web.

Bên cạnh những kết quả đạt được, hệ thống vẫn còn tồn tại một số hạn chế như khả năng xử lý các yêu cầu phức tạp ở mức độ chi tiết cao, cũng như chưa đánh giá đầy đủ hiệu năng khi triển khai ở quy mô lớn. Tuy nhiên, thông qua quá trình thực hiện đề tài, em đã tích lũy được nhiều kinh nghiệm thực tiễn trong việc thiết kế hệ thống đa tác tử, xây dựng ứng dụng web hiện đại và tích hợp các công nghệ AI vào quy trình phát triển phần mềm.

6.2 Hướng phát triển

Trong thời gian tới, để hoàn thiện hơn hệ thống đã xây dựng, hướng phát triển đầu tiên là tiếp tục cải tiến và mở rộng các chức năng của từng tác tử. Cụ thể, tác tử phân tích yêu cầu có thể được nâng cao khả năng hiểu ngữ cảnh và xử lý các yêu cầu phức tạp hơn, trong khi tác tử thiết kế giao diện và sinh mã có thể được cải tiến để tạo ra các giao diện có tính thẩm mỹ và tùy biến cao hơn.

Bên cạnh đó, hệ thống có thể được mở rộng theo hướng hỗ trợ nhiều framework và công nghệ giao diện web khác nhau, không chỉ giới hạn ở một nền tảng triển khai cụ thể. Việc tích hợp thêm các cơ chế đánh giá chất lượng giao diện, phản hồi từ người dùng và học hỏi từ các lần sử dụng trước sẽ giúp hệ thống ngày càng thông minh và phù hợp hơn với nhu cầu thực tế.

Ngoài ra, một hướng phát triển tiềm năng khác là nghiên cứu áp dụng hệ thống đa tác tử này vào các lĩnh vực phát triển phần mềm rộng hơn, không chỉ dừng lại ở giao diện web mà còn mở rộng sang thiết kế trải nghiệm người dùng (UX), xây dựng ứng dụng đa nền tảng hoặc hỗ trợ phát triển phần mềm tự động ở mức độ cao hơn. Những hướng phát triển này hứa hẹn sẽ giúp hệ thống trở thành một công cụ hỗ trợ hiệu quả trong quy trình phát triển phần mềm hiện đại.

TÀI LIỆU THAM KHẢO

[1] M. Wooldridge, *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2009.

[2] S. J. Russell and P. Norvig, "Intelligent agents: Theory and practice," *AI Magazine*, vol. 16, no. 1, pp. 23–38, 1995.

[3] M. Wooldridge and N. R. Jennings, "Agent-based software engineering," *IEE Proceedings-Software*, vol. 144, no. 1, pp. 26–37, 1997.

[4] N. R. Jennings, "On agent-based software engineering," *Artificial Intelligence*, vol. 117, no. 2, pp. 277–296, 2000.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[6] J. Niederst Robbins, *Learning Web Design*. O'Reilly Media, 2012.

[7] R. Hartson and P. Pyla, "The ux design lifecycle," *Morgan Kaufmann*, 2003.

[8] MDN Web Docs, *Web technologies for front-end development*. Accessed: Jan. 10, 2025. [Online]. Available: `https://developer.mozilla.org`.

[9] OpenAI, *Multi-agent systems and ai-assisted development*. Accessed: Jan. 10, 2025. [Online]. Available: `https://platform.openai.com/docs`.

[10] N. V. A, *Phát triển ứng dụng Web hiện đại*. Nhà xuất bản Khoa học và Kỹ thuật, 2021.