KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO KẾT THÚC MÔN CÔNG NGHỆ PHẦN MỀM

MÃ MÔN HỌC: 220055

THIẾT KẾ WEBSITE SÁNG TÁC TRUYỆN CHAT TÍCH HỢP AI

Giảng viên hướng dẫn:
ThS. Nguyễn Bảo Ân

Sinh viên thực hiện:
Huỳnh Phước Thọ
MSSV: 110122169
Nguyễn Huỳnh Phú Vinh
MSSV: 110122203
Nguyễn Phú Vinh
MSSV: 110122204

Trà Vinh, tháng 7 năm 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

....
....

Trà Vinh, ngày ..... tháng …… năm ……

Giáo viên hướng dẫn

(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....

Trà Vinh, ngày ..... tháng …… năm ……

Thành viên hội đồng

(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Chúng tôi xin gửi lời cảm ơn chân thành đến ThS. Nguyễn Bảo Ân, giảng viên hướng dẫn, người đã tận tình hướng dẫn, hỗ trợ và đóng góp những ý kiến quý báu trong suốt quá trình thực hiện đồ án. Sự định hướng và những góp ý chuyên môn của thầy đã giúp chúng tôi hoàn thiện dự án một cách khoa học và hiệu quả.

Chúng tôi cũng xin bày tỏ lòng biết ơn đến Khoa Kỹ thuật và Công nghệ, Bộ môn Công nghệ Thông tin, Trường Đại học Trà Vinh, đã tạo điều kiện thuận lợi về cơ sở vật chất, tài liệu học tập và môi trường nghiên cứu để chúng tôi có thể triển khai đồ án này.

Cuối cùng, chúng tôi xin gửi lời cảm ơn đến tất cả những ai đã trực tiếp hoặc gián tiếp góp phần vào sự thành công của đồ án này. Mặc dù đã cố gắng hết sức, dự án vẫn không thể tránh khỏi những thiếu sót. Chúng tôi rất mong nhận được những ý kiến đóng góp để hoàn thiện hơn trong tương lai.

Chúng tôi xin chân thành cảm ơn!

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH MỤC BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Giải thích
1	MCP	Model Context Protocol
2	AI	Artificial Intelligence
3	API	Application Programming Interface
4	CI/CD	Continuous Integration/Continuous Deployment
5	CRUD	Create, Read, Update, Delete
6	CSRF	Cross-Site Request Forgery
7	DoS/DDoS	Denial-of-service/Distributed Denial-of-service
8	DVCS	Distributed Version Control System
9	ISR	Incremental Static Regeneration
10	JWT	JSON Web Tokens
11	LLM	Large Language Model
12	PBI	Product Backlog Item
13	SEO	Search Engine Optimization
14	SSG	Static Site Generation

CHƯƠNG 1 GIỚI THIỆU

1.1 Mô tả bài toán

Trong thời đại số hóa, nhu cầu sáng tạo và tiêu thụ nội dung, đặc biệt là các tác phẩm truyện, ngày càng gia tăng. Tuy nhiên, quá trình sáng tác văn học luôn là một thử thách, đòi hỏi không chỉ tài năng mà còn cả sự kiên trì và nguồn cảm hứng dồi dào. Các tác giả, từ nghiệp dư đến chuyên nghiệp, thường xuyên đối mặt với những khó khăn như: thiếu thốn ý tưởng, khó khăn trong việc quản lý các tuyến nhân vật và cốt truyện phức tạp, cũng như thiếu các công cụ hỗ trợ thông minh để phát triển ý tưởng.

Các phần mềm soạn thảo văn bản truyền thống chỉ cung cấp các chức năng cơ bản, trong khi các nền tảng AI tạo sinh tuy mạnh mẽ nhưng lại thường hoạt động độc lập, không tích hợp sâu vào quy trình quản lý và xây dựng một câu chuyện hoàn chỉnh. Điều này tạo ra một khoảng trống lớn trên thị trường: một nền tảng hợp nhất, nơi tác giả có thể vừa quản lý tác phẩm một cách có hệ thống, vừa tận dụng được sức mạnh của trí tuệ nhân tạo để khơi nguồn sáng tạo và tự động hóa các công đoạn lặp lại.

Để giải quyết vấn đề này, dự án "Thiết Kế Website Sáng Tác Truyện Chat Tích Hợp AI" được đề xuất. Mục tiêu của dự án là phát triển một nền tảng web full-stack, sử dụng Next.js, cơ sở dữ liệu MySQL và tích hợp sâu với mô hình ngôn ngữ lớn Gemini, nhằm cung cấp một trợ lý ảo thông minh cho các nhà văn. Hệ thống không chỉ giúp người dùng quản lý truyện, chương, nhân vật một cách khoa học mà còn cho phép họ tương tác trực tiếp với AI để tạo ra ý tưởng, phát triển hội thoại, và xây dựng cốt truyện một cách liền mạch và hiệu quả.

1.2 Mô tả ứng dụng

Đây là một ứng dụng web được thiết kế như một không gian sáng tác thông minh, cung cấp bộ công cụ toàn diện cho việc viết và quản lý truyện với sự hỗ trợ đắc lực từ trí tuệ nhân tạo. Hệ thống được xây dựng cho các đối tượng chính sau:

Người dùng (Tác giả): Là người dùng cuối của hệ thống. Sau khi đăng ký tài khoản (qua email/mật khẩu hoặc Google), họ có thể:

Tạo và quản lý truyện: Khởi tạo các dự án truyện mới, thiết lập các thông tin cơ bản như tiêu đề, mô tả, thể loại và ảnh bìa.

Xây dựng thế giới truyện: Quản lý chi tiết các chương, xây dựng hồ sơ nhân vật với các thuộc tính đa dạng như ngoại hình, tính cách, tiểu sử.

Sáng tác nội dung: Soạn thảo nội dung cho từng chương, bao gồm các đoạn kể chuyện và hội thoại giữa các nhân vật.

Tương tác với AI: Sử dụng cửa sổ chat tích hợp để trò chuyện với AI, yêu cầu gợi ý về cốt truyện, phát triển nhân vật, hoặc tự động tạo ra các đoạn hội thoại dựa trên bối cảnh đã có.

Xuất bản và chia sẻ: Chuyển trạng thái truyện để chia sẻ với cộng đồng.

Hệ thống AI Phụ trợ: Đây là một tác nhân bên ngoài, tương tác với ứng dụng web thông qua một lớp giao thức chuẩn hóa gọi là Model Context Protocol (MCP).

Hệ thống cung cấp một MCP Server hoạt động như một cầu nối, cho phép các trợ lý AI bên ngoài có thể truy cập và thay đổi dữ liệu truyện của họ.

Thông qua MCP, một trợ lý AI có thể thực thi các công cụ tương ứng với các chức năng của ứng dụng web. Điều này mở ra khả năng quản lý và sáng tác truyện thông qua giao diện trò chuyện tự nhiên với các AI tiên tiến.

Tóm lại, ứng dụng web này không chỉ là một trình soạn thảo văn bản, mà là một nền tảng quản lý và sáng tác toàn diện, biến AI thành một người đồng hành sáng tạo thực thụ của các nhà văn.

1.3 Đặc tả các chức năng hệ thống

1.3.1 Chức năng Quản lý Người dùng và Xác thực

Đăng ký: Cho phép người dùng mới tạo tài khoản bằng email và mật khẩu. Mật khẩu được mã hóa an toàn.

Đăng nhập/Đăng xuất: Hỗ trợ đăng nhập bằng tài khoản nội bộ và qua Google . Hệ thống quản lý phiên làm việc an toàn.

Quản lý tài khoản: Người dùng có thể xem và chỉnh sửa thông tin cá nhân như tên hiển thị, ảnh đại diện.

Quản lý API Key: Người dùng có thể tạo và quản lý các khóa API để sử dụng cho các tích hợp bên ngoài.

1.3.2 Chức năng Quản lý Truyện và Nội dung

Quản lý Truyện:

Tạo truyện mới với tiêu đề, mô tả, thể loại chính, ảnh bìa.

Xem danh sách tất cả các truyện của người dùng.

Xem chi tiết một truyện, bao gồm danh sách chương và nhân vật.

Cập nhật thông tin và trạng thái của truyện.

Xóa truyện.

Quản lý Chương:

Thêm chương mới vào một truyện với tiêu đề, tóm tắt và số thứ tự.

Sắp xếp, chỉnh sửa và xóa các chương trong truyện.

Quản lý Nhân vật:

Thêm nhân vật mới vào truyện với tên, mô tả, giới tính, ngoại hình, tính cách, và vai trò.

Chỉnh sửa và xóa các nhân vật.

Quản lý Nội dung:

Thêm/sửa/xóa các đoạn nội dung trong một chương, phân biệt giữa lời kể và lời thoại của nhân vật cụ thể.

1.3.3 Chức năng Tương tác với Trí tuệ Nhân tạo (Gemini)

Chat tương tác: Cung cấp giao diện chat, nơi người dùng có thể trò chuyện với AI để nhận gợi ý, phân tích hoặc gợi ý ý tưởng.

Tạo sinh nội dung theo ngữ cảnh: AI có khả năng đọc và hiểu bối cảnh của truyện, chương, hoặc nhân vật để tạo ra các nội dung phù hợp.

Tạo sinh hội thoại: Hệ thống lưu trữ các đoạn hội thoại do AI tạo ra ở một khu vực chờ duyệt, cho phép người dùng xem xét, chỉnh sửa và quyết định thêm vào nội dung chính thức của chương.

Phản hồi liên tục: Phản hồi từ AI được trả về liên tục để tăng trải nghiệm thời gian thực cho người dùng.

1.3.4 Chức năng Tích hợp Giao thức MCP

Cung cấp MCP Server: Một máy chủ riêng biệt lắng nghe các yêu cầu từ các client AI tương thích.

Đăng ký công cụ: Đăng ký các chức năng của hệ thống thành các công cụ mà AI có thể gọi.

Xác thực: Chuyển tiếp các yêu cầu từ AI client đến API nội bộ của ứng dụng web một cách an toàn, sử dụng thông tin xác thực của người dùng.

1.3.5 Chức năng Cao cấp và Thanh toán

Phân quyền người dùng cao cấp: Hệ thống có cơ chế đánh dấu người dùng trả phí.

Tích hợp cổng thanh toán: Tích hợp với VNPay để xử lý các giao dịch khi người dùng mua các gói tính năng cao cấp.

CHƯƠNG 2 NGHIÊN CỨU LÝ THUYẾT

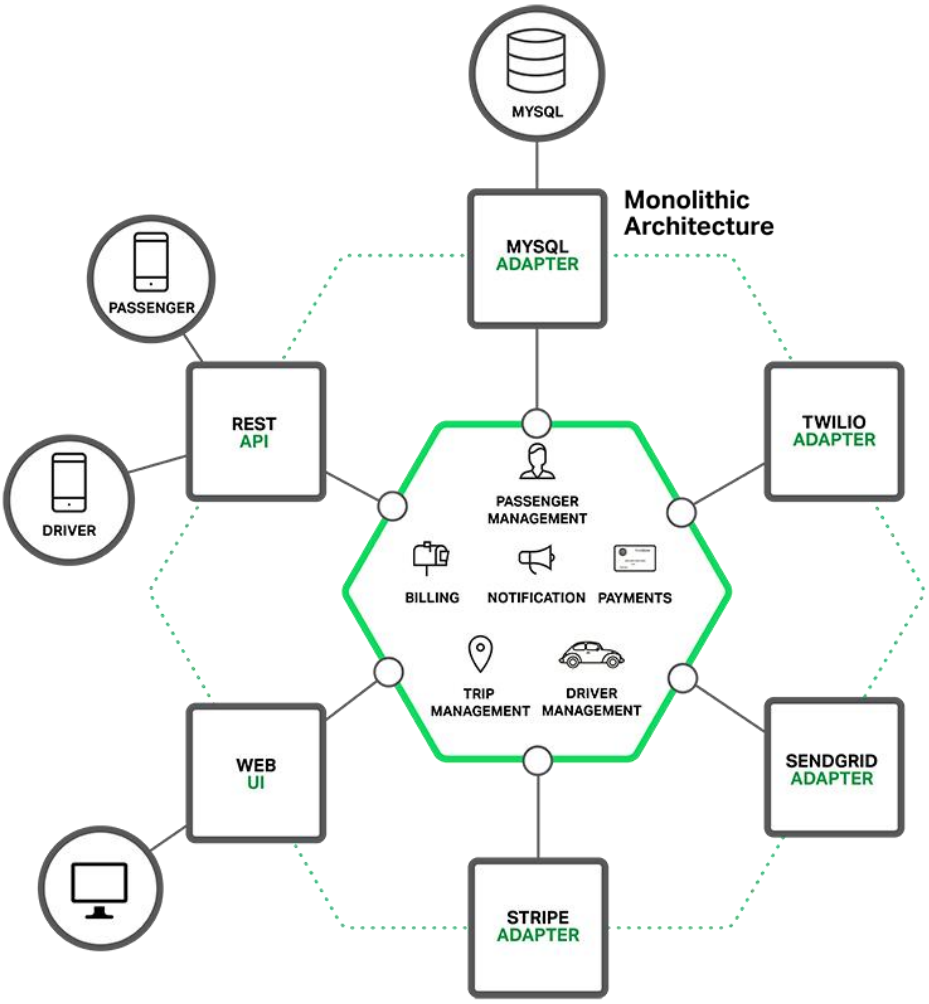
2.1 . Kiến trúc phần mềm

Kiến trúc phần mềm là quá trình định nghĩa một giải pháp có cấu trúc nhằm đáp ứng các yêu cầu về kỹ thuật và vận hành, đồng thời tối ưu hóa các thuộc tính chung về chất lượng như hiệu năng, bảo mật và khả năng quản lý. Nó đóng vai trò như một bản thiết kế chi tiết cho hệ thống, xác định các thành phần phần mềm cấp cao, mối quan hệ giữa chúng và các nguyên tắc, quy tắc chi phối việc thiết kế và phát triển theo thời gian. Việc lựa chọn một mẫu kiến trúc phù hợp là một trong những quyết định quan trọng nhất, ảnh hưởng sâu sắc đến toàn bộ vòng đời của sản phẩm.

2.1.1 Kiến trúc Monolithic

Kiến trúc Monolithic là một mô hình thiết kế phần mềm truyền thống trong đó một ứng dụng được xây dựng như một đơn vị duy nhất, thống nhất và không thể phân chia. Tất cả các thành phần chức năng của ứng dụng, chẳng hạn như lớp giao diện người dùng, lớp xử lý logic nghiệp vụ, và lớp truy cập dữ liệu, được kết nối chặt chẽ với nhau và được triển khai dưới dạng một tiến trình duy nhất.

Sơ đồ kiến trúc Monolithic



Hình 2.1: Kiến trúc của Monolithic

Ưu điểm và Nhược điểm

Ưu điểm:

Phát triển đơn giản: Quá trình phát triển, gỡ lỗi và kiểm thử ban đầu rất nhanh chóng vì toàn bộ mã nguồn nằm trong một dự án duy nhất.

Triển khai dễ dàng: Quy trình triển khai đơn giản, chỉ cần sao chép và chạy một đơn vị ứng dụng duy nhất lên máy chủ.

Hiệu năng cao: Giao tiếp giữa các thành phần diễn ra thông qua các lệnh gọi hàm nội bộ, có độ trễ rất thấp so với giao tiếp qua mạng.

Nhược điểm:

Khó mở rộng: Không thể mở rộng từng chức năng một cách độc lập. Để xử lý tải tăng cao, phải nhân bản toàn bộ ứng dụng.

Ràng buộc công nghệ: Rất khó để áp dụng các công nghệ mới. Mọi thay đổi về framework hoặc ngôn ngữ đều ảnh hưởng đến toàn bộ ứng dụng.

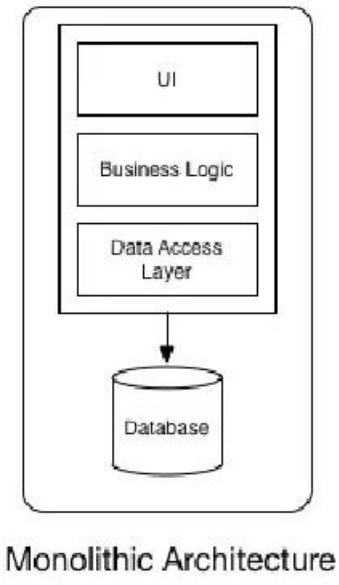
Độ tin cậy thấp: Một lỗi ở một thành phần nhỏ có thể làm ngưng hoạt động của toàn bộ hệ thống.

Bảo trì phức tạp: Khi ứng dụng phát triển lớn, mã nguồn trở nên cồng kềnh, khó hiểu và khó bảo trì, làm chậm chu kỳ phát triển.

2.1.2 Kiến trúc Microservices

Kiến trúc Microservices là một phương pháp tiếp cận kiến trúc hiện đại, trong đó một ứng dụng lớn được cấu thành từ một tập hợp các dịch vụ nhỏ, độc lập và có khả năng triển khai riêng lẻ. Mỗi dịch vụ được xây dựng xung quanh một chức năng nghiệp vụ cụ thể, có mã nguồn riêng, quản lý dữ liệu riêng và giao tiếp với nhau thông qua các giao thức nhẹ và được định nghĩa rõ ràng, phổ biến nhất là API qua giao thức HTTP/REST.

Sơ đồ kiến trúc Microservices



Hình 2.2: Sơ đồ kiến trúc Microservices

Ưu điểm và nhược điểm kiến trúc Microservices

Ưu điểm:

Khả năng mở rộng linh hoạt: Có thể mở rộng từng dịch vụ một cách độc lập dựa trên nhu cầu thực tế, giúp tối ưu hóa việc sử dụng tài nguyên.

Linh hoạt về công nghệ: Mỗi dịch vụ có thể được xây dựng bằng một ngôn ngữ lập trình hoặc framework khác nhau, phù hợp nhất với chức năng của nó.

Tính kháng lỗi cao: Lỗi ở một dịch vụ thường không làm ảnh hưởng đến toàn bộ hệ thống. Các dịch vụ khác vẫn có thể tiếp tục hoạt động bình thường.

Bảo trì và phát triển dễ dàng: Mỗi dịch vụ có một mã nguồn nhỏ, tập trung, giúp các nhóm nhỏ có thể phát triển, triển khai và bảo trì một cách độc lập và nhanh chóng.

Nhược điểm:

Phức tạp trong vận hành: Yêu cầu các hệ thống tự động hóa mạnh mẽ để quản lý việc triển khai và giám sát hàng chục hoặc hàng trăm dịch vụ.

Thách thức của hệ thống phân tán: Các vấn đề như độ trễ mạng, nhất quán dữ liệu, và xử lý lỗi trong giao tiếp giữa các dịch vụ trở nên phức tạp hơn.

Khó kiểm thử toàn diện: Việc kiểm thử các kịch bản tương tác giữa nhiều dịch vụ khác nhau là một thách thức lớn.

Chi phí ban đầu cao: Đòi hỏi đầu tư ban đầu lớn hơn về thời gian và công sức để thiết lập hạ tầng và quy trình làm việc.

2.2 Phương pháp luận phát triển - Agile và Scrum

Việc lựa chọn một phương pháp luận phát triển phần mềm phù hợp đóng vai trò quyết định đến sự thành công của dự án, ảnh hưởng đến cách đội ngũ cộng tác, phản ứng với thay đổi và bàn giao sản phẩm. Trong bối cảnh các yêu cầu của dự án có thể thay đổi và cần sự linh hoạt cao, các phương pháp luận linh hoạt đã chứng tỏ được ưu thế vượt trội so với các mô hình truyền thống như thác nước.

2.2.1 Triết lý phát triển phần mềm linh hoạt (Agile)

Giới thiệu

Agile không phải là một quy trình hay một công cụ cụ thể, mà là một triết lý, một tư duy về phát triển phần mềm. Ra đời từ bản **Tuyên ngôn Agile** vào năm 2001, Agile nhấn mạnh vào việc phát triển lặp đi lặp lại và tăng trưởng, khuyến khích sự hợp tác chặt chẽ giữa các thành viên và khách hàng, đồng thời đề cao khả năng phản ứng nhanh với các thay đổi.

Bốn giá trị cốt lõi của Tuyên ngôn Agile

Cá nhân và sự tương tác hơn là quy trình và công cụ.

Phần mềm chạy tốt hơn là tài liệu đầy đủ.

Hợp tác với khách hàng hơn là đàm phán hợp đồng.

Phản hồi với sự thay đổi hơn là bám sát kế hoạch.

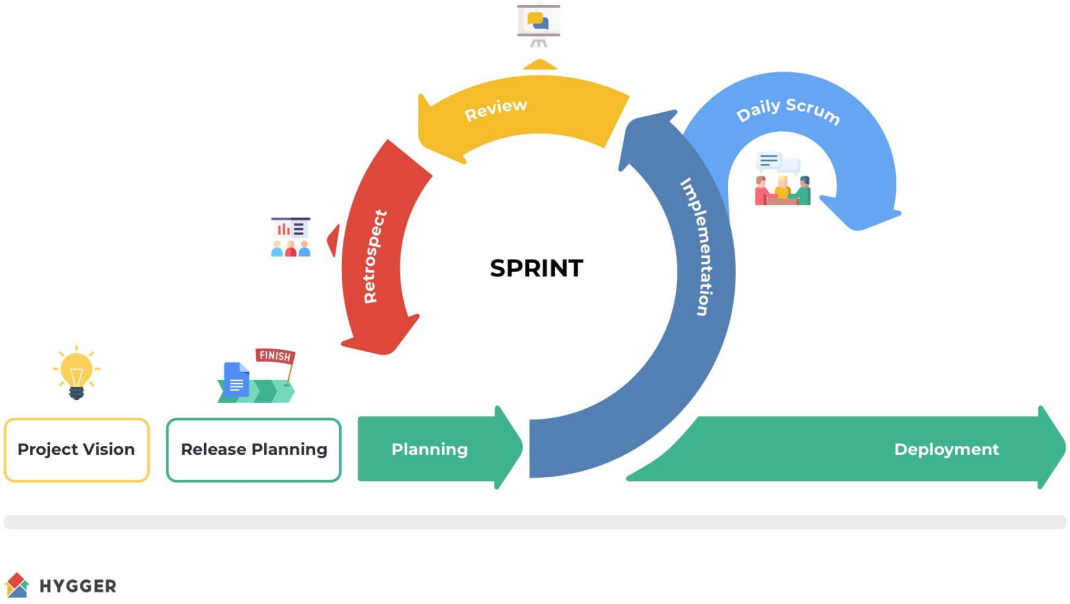
Điều này không có nghĩa là các vế bên phải không quan trọng, nhưng Agile ưu tiên và đề cao các giá trị ở vế bên trái hơn.

2.2.2 Khung làm việc Scrum

Định nghĩa

Scrum là một khung làm việc gọn nhẹ, giúp con người, đội nhóm và tổ chức tạo ra giá trị thông qua các giải pháp thích ứng cho những vấn đề phức tạp. Nó không phải là một phương pháp luận hoàn chỉnh mà cung cấp một bộ các quy tắc, vai trò, sự kiện và tạo tác để các đội ngũ có thể xây dựng và cải tiến quy trình làm việc của riêng mình. Scrum dựa trên chủ nghĩa kinh nghiệm, khẳng định rằng kiến thức đến từ kinh nghiệm và việc ra quyết định dựa trên những gì đã biết. Ba trụ cột của Scrum là: Minh bạch, Thanh tra, và Thích ứng.

Sơ đồ quy trình Scrum



Hình 2.3: Sơ đồ quy trình Scrum

Các thành phần của Scrum

Scrum bao gồm ba thành phần chính: Các vai trò, các tạo tác, và các sự kiện.

1. Các vai trò trong Scrum:

Product Owner: Là người chịu trách nhiệm tối đa hóa giá trị của sản phẩm được tạo ra bởi nhóm phát triển. Product Owner là người duy nhất quản lý Product Backlog, bao gồm việc xác định, sắp xếp thứ tự ưu tiên và đảm bảo các hạng mục trong đó được trình bày một cách rõ ràng.

Development Team: Bao gồm các chuyên gia cùng nhau làm việc để tạo ra một phần tăng trưởng có khả năng phát hành vào cuối mỗi Sprint. Nhóm có tính tự quản và đa chức năng, sở hữu tất cả các kỹ năng cần thiết để hoàn thành công việc.

Scrum Master: Là một nhà lãnh đạo phục vụ cho nhóm Scrum. Scrum Master chịu trách nhiệm đảm bảo nhóm hiểu và tuân thủ các lý thuyết, quy tắc và giá trị của Scrum. Họ giúp loại bỏ các trở ngại và tạo điều kiện cho các sự kiện Scrum diễn ra một cách hiệu quả.

2. Các tạo tác của Scrum:

Product Backlog: Là một danh sách được sắp xếp thứ tự ưu tiên của tất cả mọi thứ có thể cần đến trong sản phẩm. Đây là nguồn yêu cầu duy nhất cho bất kỳ thay đổi nào được thực hiện đối với sản phẩm.

Sprint Backlog: Bao gồm các hạng mục từ Product Backlog đã được chọn cho Sprint, cộng với một kế hoạch để hiện thực hóa các hạng mục đó và đạt được Mục tiêu Sprint.

Increment: Là tổng hợp của tất cả các hạng mục trong Product Backlog đã được hoàn thành trong một Sprint và giá trị của các phần tăng trưởng từ các Sprint trước đó. Vào cuối Sprint, Increment mới phải ở trạng thái "có thể sử dụng được".

3. Các sự kiện trong Scrum:

Tất cả các sự kiện trong Scrum đều được giới hạn thời gian.

Sprint: Là "trái tim" của Scrum, một khung thời gian cố định từ một đến bốn tuần, trong đó một phần tăng trưởng "hoàn thành", có thể sử dụng và có khả năng phát hành được tạo ra.

Sprint Planning: Được tổ chức vào đầu mỗi Sprint, nơi toàn bộ nhóm Scrum cùng nhau lập kế hoạch cho công việc sẽ được thực hiện trong Sprint.

Daily Scrum: Là một cuộc họp ngắn kéo dài 15 phút dành cho Nhóm phát triển để đồng bộ hóa hoạt động và tạo kế hoạch cho 24 giờ tiếp theo.

Sprint Review: Diễn ra vào cuối Sprint để thanh tra Increment và điều chỉnh Product Backlog nếu cần. Đây là một buổi họp không chính thức, không phải là một buổi báo cáo tình hình, mà là nơi để thu thập phản hồi và thúc đẩy sự hợp tác.

Sprint Retrospective: Diễn ra sau Sprint Review và trước buổi Sprint Planning tiếp theo. Đây là cơ hội để nhóm Scrum tự thanh tra và tạo ra một kế hoạch cải tiến cho Sprint sắp tới.

2.3 Hệ thống Quản lý Phiên bản và Tích hợp liên tục

Trong mọi dự án phát triển phần mềm, đặc biệt là các dự án có nhiều thành viên tham gia, việc quản lý mã nguồn và tự động hóa các quy trình làm việc là yếu

tố sống còn để đảm bảo chất lượng và hiệu suất. Các công cụ quản lý phiên bản và tích hợp liên tục đóng vai trò trung tâm trong việc giải quyết các thách thức này.

2.3.1 Git - Hệ thống Quản lý Phiên bản Phân tán

Định nghĩa

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System - DVCS) mã nguồn mở, được tạo ra bởi Linus Torvalds vào năm 2005. "Phân tán" có nghĩa là mỗi lập trình viên không chỉ có một bản sao làm việc của mã nguồn, mà còn có một bản sao đầy đủ của toàn bộ kho lưu trữ , bao gồm cả lịch sử thay đổi. Điều này cho phép các nhà phát triển làm việc độc lập, hợp nhất thay đổi và theo dõi lịch sử một cách hiệu quả mà không cần kết nối liên tục đến một máy chủ trung tâm.

Các khái niệm cốt lõi

- **Repository:** Là một thư mục chứa toàn bộ mã nguồn của dự án cùng với lịch sử thay đổi của nó, được lưu trữ trong một thư mục ẩn có tên là .git.
- **Commit:** Là một "ảnh chụp nhanh" của trạng thái dự án tại một thời điểm nhất định. Mỗi commit có một mã định danh duy nhất và một thông điệp mô tả những thay đổi đã được thực hiện.
- **Branch:** Là một dòng phát triển độc lập. Các nhà phát triển thường tạo ra các nhánh mới để phát triển các tính năng riêng biệt hoặc sửa lỗi mà không làm ảnh hưởng đến nhánh chính.
- **Merge:** Là hành động gộp các thay đổi từ một nhánh này vào một nhánh khác.
- **Remote:** Là một phiên bản của kho lưu trữ được lưu trữ trên một máy chủ từ xa, cho phép nhiều người cùng hợp tác trên một dự án.

2.3.2 GitHub - Nền tảng Lưu trữ và Hợp tác

Định nghĩa

GitHub là một nền tảng dịch vụ dựa trên web, cung cấp dịch vụ lưu trữ kho lưu trữ Git cho việc kiểm soát phiên bản và cộng tác. Nó không chỉ đơn thuần là nơi lưu trữ mã nguồn mà còn cung cấp một hệ sinh thái các công cụ mạnh mẽ để hỗ trợ toàn bộ vòng đời phát triển phần mềm, bao gồm quản lý dự án, theo dõi lỗi, đánh giá mã nguồn và tự động hóa quy trình.

Các tính năng nổi bật

- **Pull Request:** Đây là tính năng cốt lõi của GitHub, cho phép một nhà phát triển đề xuất các thay đổi của mình từ một nhánh này sang một nhánh khác. Pull Request tạo ra một không gian để thảo luận, đánh giá mã nguồn và chạy các kiểm thử tự động trước khi các thay đổi được chính thức hợp nhất.
- **Issue Tracking:** Cung cấp một hệ thống để theo dõi các công việc cần làm, các lỗi cần sửa, và các yêu cầu tính năng mới.
- **Project Boards:** Cung cấp các bảng công việc theo kiểu Kanban hoặc Scrum để quản lý và trực quan hóa tiến độ của dự án.
- **GitHub Actions:** Một công cụ tự động hóa mạnh mẽ được tích hợp sẵn.

2.3.3 GitHub Actions - Tự động hóa Quy trình Làm việc (CI/CD)

Định nghĩa

GitHub Actions là một nền tảng tích hợp và phân phối liên tục (CI/CD) được tích hợp trực tiếp vào GitHub. Nó cho phép tự động hóa các quy trình làm việc của bạn ngay từ bên trong kho lưu trữ. Bạn có thể xây dựng, kiểm thử và triển khai mã nguồn của mình trực tiếp từ GitHub.

Cấu trúc và Khái niệm chính

- **Workflow:** Là một quy trình tự động hóa có thể định cấu hình, bao gồm một hoặc nhiều công việc. Workflows được định nghĩa trong các file YAML và được lưu trữ trong thư mục .github/workflows của kho lưu trữ.
- **Event:** Là một hoạt động cụ thể trong kho lưu trữ sẽ kích hoạt một workflow. Ví dụ: một push lên một nhánh, một pull_request được tạo, hoặc theo một lịch trình định sẵn.

- **Job:** Là một tập hợp các bước được thực thi trên cùng một máy ảo. Các jobs có thể chạy song song hoặc tuần tự.
- **Step:** Là một tác vụ riêng lẻ có thể chạy các lệnh hoặc một **action**.
- **Action:** Là một ứng dụng tùy chỉnh cho nền tảng GitHub Actions, đóng gói các tác vụ phức tạp và có thể tái sử dụng. Có hàng ngàn actions có sẵn trên GitHub Marketplace, ví dụ như actions/checkout để lấy mã nguồn, hoặc actions/setup-node để thiết lập môi trường Node.js.

Lợi ích và Ứng dụng

Việc sử dụng GitHub Actions mang lại nhiều lợi ích thiết thực cho dự án:

- **Tự động hóa Kiểm thử:** Tự động chạy các bộ unit test và integration test mỗi khi có một Pull Request mới, đảm bảo rằng các thay đổi không phá vỡ các chức năng hiện có.
- **Đảm bảo Chất lượng Mã nguồn:** Tự động chạy các công cụ kiểm tra định dạng mã như ESLint để đảm bảo mã nguồn tuân thủ các tiêu chuẩn đã đề ra.
- **Tự động hóa Triển khai:** Tự động xây dựng Docker image và triển khai ứng dụng lên các môi trường như staging hoặc production sau khi các thay đổi đã được hợp nhất vào nhánh chính.

Bằng cách tích hợp GitHub Actions, các đội ngũ có thể xây dựng một quy trình CI/CD hoàn chỉnh, giúp giảm thiểu sai sót do con người, tăng tốc độ bàn giao sản phẩm và nâng cao chất lượng phần mềm một cách đáng kể.

2.4 Các công nghệ và Framework chính

Việc lựa chọn một ngăn xếp công nghệ phù hợp là yếu tố quyết định đến sự thành công của một dự án phần mềm. Ngăn xếp công nghệ không chỉ ảnh hưởng đến hiệu năng và khả năng mở rộng của sản phẩm mà còn tác động trực tiếp đến hiệu suất làm việc của đội ngũ phát triển.

2.4.1 **Next.js - Framework Full-stack**

Giới thiệu

Next.js là một framework mã nguồn mở được xây dựng trên nền tảng React, do Vercel phát triển. Nó cung cấp một bộ công cụ và quy ước mạnh mẽ để xây dựng các ứng dụng web hiện đại, từ các trang web tĩnh đến các ứng dụng động, phức tạp. Điểm đặc biệt của Next.js là khả năng hỗ trợ phát triển full-stack, cho phép xây dựng cả giao diện người dùng (frontend) và logic máy chủ (backend) trong cùng một dự án.

Các tính năng nổi bật

+ Rendering linh hoạt: Next.js hỗ trợ nhiều chiến lược render khác nhau như Server-Side Rendering (SSR), Static Site Generation (SSG), và Incremental Static Regeneration (ISR), giúp tối ưu hóa hiệu năng và SEO.
+ Hệ thống Routing dựa trên file: Tự động tạo các tuyến đường (routes) dựa trên cấu trúc thư mục trong app/ hoặc pages/, giúp việc điều hướng trở nên trực quan và dễ quản lý.
+ API Routes: Đây là một trong những tính năng then chốt, cho phép các nhà phát triển tạo ra các endpoint API ngay bên trong ứng dụng Next.js. Các file được đặt trong thư mục app/api/ sẽ được ánh xạ thành các API endpoint, giúp xây dựng backend một cách liền mạch mà không cần một máy chủ riêng biệt.

2.4.2 . TypeScript - Ngôn ngữ lập trình

Giới thiệu

TypeScript là một ngôn ngữ lập trình mã nguồn mở được phát triển bởi Microsoft. Về bản chất, nó là một tập hợp cha của JavaScript, bổ sung thêm hệ thống kiểu tĩnh và các tính năng hướng đối tượng kinh điển. Toàn bộ mã nguồn TypeScript sau đó sẽ được biên dịch thành mã JavaScript thuần để có thể chạy trên trình duyệt hoặc môi trường Node.js.

Các tính năng nổi bật

+ Phát hiện lỗi sớm: Hệ thống kiểu tĩnh giúp phát hiện các lỗi liên quan đến kiểu dữ liệu ngay trong quá trình phát triển thay vì ở giai đoạn chạy ứng dụng, giúp giảm thiểu lỗi và tăng độ tin cậy của mã nguồn.
+ Nâng cao trải nghiệm lập trình: Các trình soạn thảo mã như VS Code có thể tận dụng thông tin về kiểu để cung cấp các tính năng gợi ý mã, điều hướng và tái cấu trúc mã nguồn một cách thông minh và chính xác.
+ Tăng tính dễ đọc và bảo trì: Việc khai báo kiểu rõ ràng giúp mã nguồn trở nên dễ hiểu hơn, đặc biệt trong các dự án lớn có nhiều thành viên hợp tác.

2.4.3 Cơ sở dữ liệu quan hệ và MySQL

Giới thiệu Mô hình dữ liệu quan hệ

Mô hình dữ liệu quan hệ là một phương pháp tổ chức dữ liệu thành các bảng gồm các hàng và cột. Mỗi bảng đại diện cho một loại thực thể và mỗi hàng trong bảng là một bản ghi cụ thể của thực thể đó. Mối quan hệ giữa các bảng được thiết lập thông qua các khóa ngoại, giúp đảm bảo tính toàn vẹn, nhất quán và giảm thiểu sự dư thừa dữ liệu.

Hệ quản trị cơ sở dữ liệu MySQL

MySQL là một trong những hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở phổ biến và mạnh mẽ nhất thế giới. Nó được biết đến với hiệu năng ổn định, độ tin cậy cao và cộng đồng hỗ trợ lớn.

2.4.4 . RESTful API

Định nghĩa

REST (Representational State Transfer) là một kiểu kiến trúc phần mềm để thiết kế các ứng dụng mạng. Một API được thiết kế theo các nguyên tắc của REST được gọi là RESTful API. Thay vì là một tiêu chuẩn cứng nhắc, REST là một tập hợp các ràng buộc và quy ước thiết kế, giúp tạo ra các hệ thống có khả năng mở rộng, linh hoạt và dễ bảo trì.

Các tính năng nổi bật

+ Client-Server: Tách biệt rõ ràng giữa client và server.

+ Stateless: Mỗi yêu cầu từ client đến server phải chứa tất cả thông tin cần thiết để server hiểu và xử lý nó. Server không lưu trữ bất kỳ trạng thái nào của client giữa các yêu cầu.
+ Resource-based: Mọi thứ trong hệ thống đều được coi là một tài nguyên, và mỗi tài nguyên được định danh duy nhất bằng một URI.
+ Thao tác qua các phương thức HTTP: Client tương tác với tài nguyên thông qua các phương thức HTTP chuẩn như GET, POST, PUT/PATCH , và DELETE.

2.4.5 NextAuth.js - Thư viện xác thực

Giới thiệu

NextAuth.js là một giải pháp xác thực mã nguồn mở, hoàn chỉnh và linh hoạt, được thiết kế đặc biệt cho các ứng dụng Next.js. Nó cung cấp một cách đơn giản để thêm chức năng đăng nhập/đăng xuất vào ứng dụng, hỗ trợ nhiều nhà cung cấp xác thực và chiến lược khác nhau.

Các tính năng chính

+ **Hỗ trợ đa nhà cung cấp:** Dễ dàng tích hợp với các nhà cung cấp phổ biến như Google, Facebook, GitHub, cũng như hỗ trợ xác thực qua email/mật khẩu truyền thống.
+ **Quản lý phiên làm việc tự động:** Tự động xử lý việc tạo và quản lý session, hỗ trợ cả chiến lược dùng cơ sở dữ liệu và JWT (JSON Web Tokens) với các cookie an toàn.
+ **Bảo mật tích hợp:** Tích hợp sẵn các biện pháp bảo vệ chống lại các tấn công phổ biến như CSRF.

2.5 Công nghệ Container hóa và Triển khai

Trong phát triển phần mềm hiện đại, việc đảm bảo ứng dụng hoạt động nhất quán trên nhiều môi trường khác nhau, từ máy tính của lập trình viên đến máy chủ sản phẩm, là một thách thức lớn. Công nghệ container hóa ra đời để giải quyết vấn đề này, cung cấp một phương pháp đóng gói và triển khai ứng dụng một cách hiệu quả và đáng tin cậy.

2.5.1 **Docker**

Định nghĩa

Docker là một nền tảng mã nguồn mở hàng đầu cho phép tự động hóa việc triển khai, mở rộng và quản lý ứng dụng bằng cách sử dụng công nghệ container. Container là một đơn vị phần mềm nhẹ, độc lập và có thể thực thi, bao gồm mọi thứ cần thiết để chạy một ứng dụng: mã nguồn, thời gian chạy, các công cụ hệ thống, và thư viện. Về cơ bản, Docker cho phép đóng gói một ứng dụng và các phụ thuộc của nó vào một container ảo có thể chạy trên bất kỳ máy chủ Linux, Windows, hay macOS nào.

Các khái niệm cốt lõi

- **Dockerfile:** Là một file văn bản chứa một tập hợp các chỉ thị và lệnh để Docker tự động xây dựng một Image. Nó hoạt động như một bản thiết kế hay công thức để tạo ra môi trường cho ứng dụng.
- **Image:** Là một gói thực thi, tĩnh và không thay đổi, chứa mã nguồn ứng dụng cùng với tất cả các thư viện và phụ thuộc cần thiết. Image được tạo ra từ một Dockerfile.
- **Container:** Là một thực thể chạy của một Image. Có thể có nhiều Container chạy từ cùng một Image, mỗi container hoạt động trong một môi trường được cô lập hoàn toàn với các container khác và với máy chủ.

Lợi ích của Docker

+ **Tính nhất quán:** Docker giải quyết triệt để vấn đề bằng cách đóng gói ứng dụng trong một container, nó đảm bảo ứng dụng sẽ chạy theo cùng một cách, bất kể môi trường triển khai cuối cùng là gì.
+ **Tính di động:** Container có thể dễ dàng được di chuyển và chạy trên bất kỳ hệ thống nào có cài đặt Docker, từ máy tính xách tay cá nhân đến các nhà cung cấp dịch vụ đám mây.
+ **Hiệu quả tài nguyên:** Container chia sẻ nhân của hệ điều hành máy chủ, giúp chúng nhẹ hơn và khởi động nhanh hơn đáng kể so với các máy ảo truyền thống.

2.5.2 Docker Compose

Định nghĩa

Trong khi Docker rất hiệu quả để quản lý từng container riêng lẻ, hầu hết các ứng dụng thực tế đều bao gồm nhiều thành phần khác. Docker Compose là một công cụ được thiết kế để định nghĩa và chạy các ứng dụng đa container. Bằng cách sử dụng một file cấu hình YAML, người dùng có thể định nghĩa tất cả các dịch vụ, mạng, và không gian lưu trữ cần thiết cho ứng dụng của mình.

Các thành phần chính trong file docker-compose.yml

+ **Services:** Mỗi service tương ứng với một container sẽ được chạy, định nghĩa các thuộc tính như image nào sẽ được sử dụng, các cổng mạng nào sẽ được mở, và các biến môi trường nào sẽ được truyền vào.
+ **Networks:** Cho phép tạo ra các mạng ảo riêng biệt để các container trong cùng một ứng dụng có thể giao tiếp với nhau một cách an toàn, sử dụng tên của service thay vì địa chỉ IP.
+ **Volumes:** Dùng để lưu trữ dữ liệu một cách bền vững. Dữ liệu trong volumes sẽ không bị mất ngay cả khi container bị xóa và tạo lại, điều này cực kỳ quan trọng đối với các dịch vụ như cơ sở dữ liệu.

2.5.3 Tích hợp và Triển khai liên tục (CI/CD)

Định nghĩa

CI/CD là một phương pháp trong thực hành DevOps, kết hợp giữa Tích hợp liên tục và Triển khai/Phân phối liên tục, nhằm tự động hóa quy trình xây dựng, kiểm thử và triển khai phần mềm.

Continuous Integration (CI): Là thực hành tự động hóa việc hợp nhất các thay đổi về mã nguồn từ nhiều người đóng góp vào một kho lưu trữ chung. Mỗi lần hợp nhất, một quy trình xây dựng và kiểm thử tự động sẽ được kích hoạt để phát hiện các lỗi tích hợp sớm nhất có thể.

Continuous Deployment (CD): Là bước tiếp theo của CI, tự động hóa việc triển khai mọi thay đổi đã vượt qua giai đoạn kiểm thử lên môi trường sản.

Quy trình làm việc (Pipeline)

Một quy trình CI/CD điển hình thường bao gồm các bước sau:

- **Commit & Push:** Lập trình viên đẩy mã nguồn mới lên kho lưu trữ.
- **Build:** Hệ thống CI tự động xây dựng ứng dụng, ví dụ như tạo một Docker Image mới.
- **Test:** Chạy các bộ kiểm thử tự động để đảm bảo chất lượng.
- **Deploy:** Nếu tất cả các bước trên thành công, ứng dụng sẽ được tự động triển khai lên môi trường đích.

Công cụ và Lợi ích

Các công cụ như **GitHub Actions**, Jenkins, hay GitLab CI/CD cho phép định nghĩa các quy trình này dưới dạng mã. Việc áp dụng CI/CD giúp giảm thiểu các công việc thủ công, tăng tốc độ phát hành sản phẩm, cải thiện chất lượng mã nguồn và cho phép các đội ngũ phát triển phản ứng nhanh hơn với các yêu cầu thay đổi.

2.6 Tích hợp Trí tuệ Nhân tạo và Giao thức mở rộng

Sự phát triển vượt bậc của trí tuệ nhân tạo (AI), đặc biệt là các mô hình ngôn ngữ lớn, đã mở ra những phương thức tương tác và tạo sinh nội dung hoàn toàn mới. Việc tích hợp các công nghệ này không chỉ là một tính năng mà đã trở thành một phần cốt lõi của nhiều ứng dụng hiện đại.

2.6.1 Mô hình Ngôn ngữ lớn và Gemini

Định nghĩa Mô hình Ngôn ngữ lớn

Mô hình Ngôn ngữ lớn (LLM) là một loại mô hình học sâu được huấn luyện trên một khối lượng dữ liệu văn bản khổng lồ. Dựa trên kiến trúc Transformer, các LLM có khả năng hiểu, phân tích, tóm tắt, dịch thuật và quan trọng nhất là tạo ra văn bản mới có tính mạch lạc, tự nhiên và phù hợp với ngữ cảnh. Chúng học được các mẫu cú pháp, ngữ nghĩa, và các kiến thức tiềm ẩn trong dữ liệu huấn luyện, cho phép chúng thực hiện các tác vụ ngôn ngữ phức tạp mà không cần được lập trình một cách tường minh cho từng tác vụ.

Gemini

Gemini là một trong những họ mô hình ngôn ngữ lớn tiên tiến nhất do Google phát triển. Điểm nổi bật của Gemini là kiến trúc **đa phương thức** tự nhiên, cho phép nó xử lý và suy luận đồng thời trên nhiều loại dữ liệu khác nhau, bao gồm văn bản, hình ảnh, âm thanh và video. Họ mô hình Gemini bao gồm nhiều phiên bản được tối ưu hóa cho các mục đích sử dụng khác nhau:

- **Gemini Ultra:** Phiên bản mạnh mẽ nhất, dành cho các tác vụ đòi hỏi sự phức tạp và suy luận cao.
- **Gemini Pro:** Phiên bản cân bằng giữa hiệu năng và chi phí, phù hợp với đa số ứng dụng.
- **Gemini Flash/Nano:** Các phiên bản được tối ưu hóa về tốc độ và hiệu quả, lý tưởng cho các ứng dụng đòi hỏi độ trễ thấp như chat tương tác hoặc các tác vụ trên thiết bị di động.

Kỹ nghệ câu lệnh (Prompt Engineering)

Việc giao tiếp hiệu quả với một LLM không chỉ đơn giản là đặt câu hỏi. Kỹ nghệ câu lệnh là một lĩnh vực nghiên cứu và thực hành về việc thiết kế và tối ưu hóa các câu lệnh đầu vào để điều khiển và dẫn dắt mô hình tạo ra kết quả đầu ra mong muốn. Một prompt hiệu quả thường bao gồm các yếu tố sau:

+ **Bối cảnh:** Cung cấp thông tin nền cần thiết để mô hình hiểu được tình huống.
+ **Chỉ thị:** Nêu rõ nhiệm vụ mà mô hình cần thực hiện.
+ **Ví dụ:** Đưa ra một hoặc vài cặp ví dụ đầu vào-đầu ra để mô hình học theo định dạng mong muốn.
+ **Định dạng đầu ra:** Yêu cầu mô hình trả về kết quả theo một cấu trúc cụ thể.

2.6.2 **Giao thức cho Trợ lý AI - Model Context Protocol (MCP)**

Bài toán và sự cần thiết

Khi các trợ lý AI như các chatbot trên máy tính để bàn hoặc di động ngày càng trở nên thông minh, một thách thức lớn xuất hiện: làm thế nào để các trợ lý này có thể tương tác và thực hiện hành động trên các ứng dụng của bên thứ ba một cách an toàn và chuẩn hóa. Mỗi ứng dụng có một bộ API riêng, và việc dạy cho một AI cách sử dụng hàng ngàn bộ API khác nhau là không khả thi. Giao thức MCP ra đời để giải quyết vấn đề này.

Định nghĩa và Cơ chế hoạt động

Model Context Protocol (MCP) là một giao thức mã nguồn mở được thiết kế để tạo ra một cầu nối chuẩn hóa giữa các trợ lý AI (client) và các ứng dụng bên ngoài (server). Nó hoạt động tương tự như một "API dành cho AI", cho phép một trợ lý AI có thể khám phá và sử dụng các chức năng mà một ứng dụng cung cấp.

Quy trình hoạt động cơ bản như sau:

+ **Cung cấp Công cụ:** Ứng dụng triển khai một MCP Server và định nghĩa các chức năng của mình dưới dạng một bộ công cụ.
+ **Thực thi Công cụ:** Khi người dùng ra lệnh cho trợ lý AI, trợ lý AI sẽ nhận diện được yêu cầu này tương ứng với công cụ.
+ **Gửi yêu cầu MCP:** Trợ lý AI sẽ gửi một yêu cầu có cấu trúc theo chuẩn MCP đến MCP Server, chứa tên công cụ cần gọi và các tham số cần thiết.
+ **Xử lý và Phản hồi:** MCP Server nhận yêu cầu, xác thực, và chuyển nó thành một lệnh gọi API nội bộ đến logic của ứng dụng. Sau khi nhận được kết quả, MCP Server sẽ định dạng lại kết quả theo chuẩn MCP và gửi ngược lại cho trợ lý AI.

Lợi ích

Việc áp dụng MCP mang lại những lợi ích chiến lược:

+ **Tính tương hợp:** Tạo ra một tiêu chuẩn chung, cho phép bất kỳ trợ lý AI nào tương thích với MCP đều có thể kết nối và tương tác với ứng dụng.

+ **Tính mở rộng:** Mở rộng khả năng truy cập và điều khiển ứng dụng ra ngoài giao diện người dùng truyền thống, cho phép người dùng tương tác thông qua ngôn ngữ tự nhiên một cách mạnh mẽ.
+ **Tách biệt logic:** Tách biệt logic cốt lõi của ứng dụng khỏi logic giao tiếp phức tạp với các trợ lý AI, giúp hệ thống trở nên gọn gàng và dễ bảo trì hơn.

CHƯƠNG 3 PHÂN TÍCH YÊU CẦU

3.1 Phân tích Đối tượng Người dùng

Hệ thống được xây dựng để phục vụ cho các nhóm người dùng khác nhau, mỗi nhóm có những nhu cầu và mục tiêu riêng.

Các đối tượng trong dự án bao gồm:

Tác giả (Nhà văn) Bao gồm các nhà văn chuyên nghiệp, bán chuyên và những người mới bắt đầu sáng tác. Những người trực tiếp sử dụng các công cụ của hệ thống để xây dựng và quản lý tác phẩm của mình.

Cần một giao diện soạn thảo mạnh mẽ, hệ thống quản lý khoa học, và một trợ lý AI tích hợp sâu, hiểu ngữ cảnh.

Độc giả: Là những người truy cập vào website để tìm kiếm và đọc các truyện đã được xuất bản.

Cần một giao diện đọc thân thiện, dễ điều hướng và có các tính năng tìm kiếm, lọc truyện hiệu quả.

Trợ lý AI bên ngoài (thông qua MCP) Là các hệ thống AI của bên thứ ba có khả năng tương tác với ứng dụng thông qua giao thức Model Context Protocol (MCP).

Cần một API chuẩn hóa, an toàn và được tài liệu hóa rõ ràng để có thể "giao tiếp" và điều khiển các chức năng của hệ thống.

3.2 3 Yêu cầu Chức năng

Yêu cầu chức năng được tổ chức thành Product Backlog, bao gồm các hạng mục công việc (Product Backlog Items - PBIs) được ưu tiên dựa trên giá trị mang lại và tính khả thi kỹ thuật.

3.2.1 Product Backlog

Product Backlog là một danh sách được sắp xếp thứ tự ưu tiên, chứa tất cả các yêu cầu, tính năng, cải tiến và sửa lỗi cần thiết cho sản phẩm. Dưới đây là danh sách các Hạng mục Product Backlog chính của dự án.

PBI Nhóm 1: Hệ thống Xác thực và Quản lý Người dùng

PBI-1: Xây dựng hệ thống đăng ký tài khoản mới bằng email và mật khẩu, với cơ chế mã hóa an toàn.

PBI-2: Triển khai chức năng đăng nhập bằng tài khoản đã đăng ký.

PBI-3: Tích hợp phương thức đăng nhập nhanh thông qua tài khoản Google.

PBI-4: Xây dựng chức năng Quên mật khẩu và quy trình đặt lại mật khẩu an toàn qua email.

PBI-5: Phát triển trang quản lý tài khoản, cho phép người dùng xem và cập nhật thông tin cá nhân.

PBI-6: Tích hợp Google Drive API để người dùng có thể tải lên và thay đổi ảnh đại diện.

PBI Nhóm 2: Quản lý Nội dung Truyện

PBI-7: Xây dựng chức năng tạo, đọc, cập nhật, xóa cho đối tượng Truyện, bao gồm các thông tin cơ bản như tiêu đề, mô tả, và ảnh bìa.

PBI-8: Xây dựng chức năng cho đối tượng Nhân vật, liên kết với một truyện cụ thể.

PBI-9: Xây dựng chức năng CRUD cho đối tượng Chương, bao gồm khả năng sắp xếp thứ tự các chương trong một truyện.

PBI-10: Phát triển trang soạn thảo nội dung chương, cho phép quản lý và sắp xếp các đoạn hội thoại và lời kể.

PBI-11: Triển khai hệ thống phân loại truyện bằng Thẻ thay cho danh mục cố định, cho phép gắn nhiều thẻ cho một truyện.

PBI-12: Xây dựng trang "Thư viện truyện" để hiển thị danh sách các truyện đã xuất bản.

PBI Nhóm 3: Tích hợp Trí tuệ Nhân tạo

PBI-13: Tích hợp Gemini AI để cung cấp chức năng gợi ý ý tưởng khi người dùng tạo truyện mới.

PBI-14: Tích hợp mô hình tạo ảnh để gợi ý và tạo ảnh bìa cho truyện.

PBI-15: Nâng cao chức năng tạo nhân vật bằng cách sử dụng AI để gợi ý các chi tiết về ngoại hình, tính cách, tiểu sử.

PBI-16: Xây dựng component Chatbot Trợ lý AI, cho phép người dùng tương tác trực tiếp để phát triển truyện.

PBI-17: Triển khai cơ chế phản hồi liên tục cho Chatbot để cải thiện trải nghiệm tương tác thời gian thực.

PBI-18: Cho phép người dùng tạo nhanh một truyện hoặc một nhân vật hoàn chỉnh thông qua vài câu lệnh với AI.

PBI-19: Nâng cao chức năng tạo hội thoại, cho phép AI tạo ra nhiều đoạn hội thoại cùng lúc dựa trên ngữ cảnh.

PBI-20: Tích hợp AI vào chức năng tìm kiếm, cho phép tìm kiếm ngữ nghĩa và trả về kết quả dựa trên mức độ liên quan.

PBI Nhóm 4: Trải nghiệm người dùng và Giao diện

PBI-21: Thiết kế và triển khai giao diện Đăng nhập/Đăng ký thân thiện, sử dụng thư viện shadcn/ui.

PBI-22: Xây dựng hệ thống điều hướng đáp ứng, hoạt động tốt trên cả máy tính và thiết bị di động.

PBI-23: Triển khai chức năng Yêu thích truyện.

PBI-24: Triển khai chức năng Đánh dấu vị trí đang đọc trong một chương.

PBI-25: Xây dựng hệ thống theo dõi tiến độ đọc của người dùng qua từng chương.

PBI-26: Phát triển chức năng tìm kiếm nâng cao với các bộ lọc chi tiết.

PBI-27: Cải thiện trải nghiệm tải trang bằng cách thêm màn hình loading và nhà cung cấp trạng thái tải.

PBI-28: Triển khai các hiệu ứng giao diện tinh tế như con trỏ tùy chỉnh, cuộn vô hạn, và thanh trượt truyện nổi bật.

PBI Nhóm 5: Thanh toán và Cơ sở hạ tầng

PBI-29: Tích hợp cổng thanh toán VNPay để xử lý giao dịch cho các tính năng cao cấp.

PBI-30: Xây dựng hệ thống "Huy hiệu người ủng hộ" để ghi nhận những người dùng đã thanh toán.

PBI-31: Thiết lập và cấu hình dự án Next.js ban đầu với các thiết lập cơ bản.

PBI-32: Tích hợp Model Context Protocol (MCP) để cho phép các trợ lý AI bên ngoài tương tác với hệ thống.

PBI-33: Xây dựng các kịch bản kiểm thử cho API để đảm bảo chất lượng và sự ổn định.

PBI-34: Hoàn thiện tài liệu dự án một cách toàn diện.

3.3 Yêu cầu Phi chức năng

Yêu cầu phi chức năng xác định các thuộc tính chất lượng của hệ thống, đảm bảo hiệu năng, bảo mật, và khả năng sử dụng:

Hiệu năng: Hệ thống phải xử lý ít nhất 1.000 người dùng đồng thời với thời gian phản hồi trung bình dưới 2 giây cho các yêu cầu API.

Khả năng mở rộng: Hỗ trợ mở rộng ngang thông qua container hóa với Docker, cho phép triển khai thêm instance khi lưu lượng tăng.

Bảo mật:

- Mật khẩu được mã hóa bằng bcrypt.
- API sử dụng HTTPS và xác thực bằng JWT.
- Bảo vệ chống tấn công CSRF và XSS thông qua NextAuth.js.

Tính khả dụng: Đạt uptime 99.9% trong môi trường sản phẩm.

Khả năng tương thích: Hoạt động trên các trình duyệt phổ biến (Chrome, Firefox, Safari) và thiết bị di động (iOS, Android).

Khả năng bảo trì: Mã nguồn được tổ chức mô-đun, sử dụng TypeScript để tăng tính dễ đọc và bảo trì.

Khả năng sử dụng: Giao diện trực quan, thời gian học cách sử dụng không quá 10 phút cho người dùng mới.

3.4 Ràng buộc Kỹ thuật

Dự án chịu các ràng buộc kỹ thuật sau:

Ngăn xếp công nghệ: Sử dụng Next.js (phiên bản 14 trở lên) với TypeScript, cơ sở dữ liệu MySQL với mô hình quan hệ.

Tích hợp AI: Tích hợp Gemini AI qua API chính thức, yêu cầu cấu hình API Key và quản lý giới hạn tần suất gọi.

Container hóa: Ứng dụng được triển khai trong container Docker, sử dụng Docker Compose để quản lý đa container (web, database, MCP Server)

Hạn chế tài nguyên: Đội phát triển gồm 2-3 thành viên, yêu cầu ưu tiên các tính năng cốt lõi và tối ưu hóa hiệu suất.

Khả năng tương thích: Hệ thống phải hoạt động trên các trình duyệt phổ biến và thiết bị di động, sử dụng shadcn/ui và TailwindCSS.

CHƯƠNG 4 THIẾT KẾ HỆ THỐNG

4.1 Kiến trúc tổng thể

Để đáp ứng yêu cầu về tính linh hoạt, tốc độ phát triển và khả năng triển khai nhất quán, dự án được xây dựng theo kiến trúc Full-stack Monolithic sử dụng Next.js. Kiến trúc này cho phép cả logic Frontend và Backend cùng tồn tại trong một codebase duy nhất, giúp đơn giản hóa quy trình phát triển và quản lý.

Mô hình Client-Server: Hệ thống tuân thủ mô hình client-server cơ bản, nơi giao diện người dùng (client) chạy trên trình duyệt của người dùng và giao tiếp với logic nghiệp vụ (server) thông qua các API endpoint. Điều này giúp tách biệt rõ ràng giữa giao diện và xử lý dữ liệu.

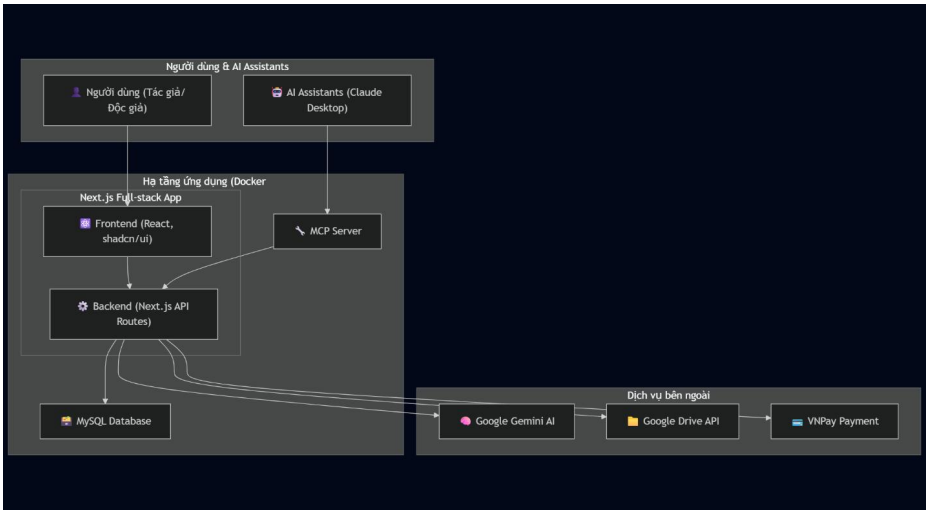
Next.js Full-stack: Điểm đặc biệt của kiến trúc này là cả Frontend và Backend đều được xây dựng trên cùng một nền tảng Next.js, tận dụng tối đa các tính năng của framework:

- **Frontend:** Được xây dựng bằng React và các component từ thư viện shadcn/ui, đảm bảo giao diện người dùng hiện đại, đáp ứng (responsive) và có trải nghiệm tốt nhất. Next.js cung cấp các cơ chế render linh hoạt như Server-Side Rendering (SSR) và Static Site Generation (SSG) để tối ưu hóa hiệu năng và SEO.
- **Backend:** Được triển khai thông qua API Routes của Next.js. Các tệp trong thư mục src/app/api/ sẽ tự động được ánh xạ thành các endpoint API. Lớp backend này chịu trách nhiệm xử lý logic nghiệp vụ, xác thực người dùng, tương tác với cơ sở dữ liệu và các dịch vụ bên ngoài.

Cơ sở dữ liệu: Hệ thống sử dụng MySQL 8.0, một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mạnh mẽ và phổ biến. Việc lựa chọn MySQL đảm bảo tính toàn vẹn dữ liệu (ACID), khả năng truy vấn phức tạp và hiệu năng ổn định cho một ứng dụng có cấu trúc dữ liệu rõ ràng như quản lý truyện.

Containerization: Toàn bộ ứng dụng (web app, database, MCP server) được đóng gói (containerized) bằng Docker và quản lý bởi Docker Compose. Điều này đảm bảo môi trường phát triển và triển khai nhất quán trên mọi máy tính, loại bỏ

các vấn đề liên quan đến sự khác biệt môi trường, đồng thời giúp việc cài đặt và vận hành trở nên dễ dàng và di động.



Hình 3.1: Sơ đồ kiến trúc tổng thể của hệ thống

4.2 Thiết kế Cơ sở dữ liệu (Database Design)

Hệ thống sử dụng mô hình cơ sở dữ liệu quan hệ để đảm bảo tính toàn vẹn, nhất quán và dễ dàng truy vấn dữ liệu phức tạp liên quan đến truyện, chương và nhân vật.

Hệ quản trị CSDL: MySQL 8.0.

Mô hình Quan hệ Thực thể (ERD): Sơ đồ ERD được thiết kế để mô tả chi tiết các thực thể và mối quan hệ giữa chúng.

Các thực thể chính:

- users: Lưu trữ thông tin người dùng, bao gồm cả thông tin xác thực (mật khẩu đã mã hóa) và hồ sơ cá nhân (tên, ảnh đại diện).
- stories: Lưu trữ thông tin tổng quan về mỗi câu chuyện (tiêu đề, mô tả, ảnh bìa, trạng thái).
- story_chapters: Lưu trữ thông tin về các chương trong một câu chuyện.
- story_characters: Lưu trữ hồ sơ chi tiết của các nhân vật.
- chapter_dialogues: Lưu trữ các đoạn hội thoại hoặc mô tả trong một chương.

- story_outlines: Lưu trữ các bản phác thảo, đại cương cho truyện, giúp tác giả lên kế hoạch.
- main_categories và story_tags: Quản lý hệ thống phân loại truyện, cho phép tìm kiếm và lọc hiệu quả.
- api_keys: Quản lý các khóa API cho phép AI bên ngoài tương tác một cách an toàn.

Mối quan hệ: Các bảng được liên kết với nhau thông qua các khóa ngoại (Foreign Keys) để thể hiện các mối quan hệ logic, ví dụ:

- Một users có thể tạo nhiều stories (quan hệ 1-nhiều).
- Một stories có thể có nhiều story_chapters (quan hệ 1-nhiều).
- Một stories và một story_tags có quan hệ nhiều-nhiều, được thể hiện qua bảng trung gian story_tag_relations.

4.3 Thiết kế API (API Design)

API của hệ thống được thiết kế theo tiêu chuẩn RESTful, đảm bảo tính nhất quán, dễ hiểu và dễ tích hợp cho cả client nội bộ và các dịch vụ bên ngoài.

Định dạng dữ liệu: JSON được sử dụng làm định dạng chuẩn cho cả request và response.

Cấu trúc Endpoint: Các endpoint được tổ chức xoay quanh các tài nguyên chính của hệ thống, giúp API trở nên logic và dễ đoán.

- /api/stories: Quản lý các tài nguyên liên quan đến truyện (lấy danh sách, tạo mới).
- /api/stories/{id}/chapters: Quản lý các chương của một truyện cụ thể.
- /api/user: Quản lý thông tin tài khoản người dùng (cập nhật hồ sơ, đổi mật khẩu).
- /api/auth: Xử lý các nghiệp vụ xác thực (đăng nhập, đăng ký, quên mật khẩu).

Phương thức HTTP: Sử dụng đúng ngữ nghĩa của các phương thức HTTP để thực hiện các thao tác CRUD (Create, Read, Update, Delete):

- GET: Lấy thông tin tài nguyên (ví dụ: lấy danh sách truyện, chi tiết một chương).
- POST: Tạo mới một tài nguyên (ví dụ: tạo truyện mới, thêm nhân vật).
- PUT/PATCH: Cập nhật một tài nguyên đã có (ví dụ: sửa đổi thông tin truyện).
- DELETE: Xóa một tài nguyên (ví dụ: xóa một chương).

Tài liệu hóa API: Hệ thống tự động tạo tài liệu API theo chuẩn OpenAPI 3.0 (Swagger) từ các chú thích (JSDoc comments) trong mã nguồn. Điều này giúp các lập trình viên (bao gồm cả việc tích hợp AI) có thể dễ dàng hiểu và tương tác với API mà không cần xem mã nguồn chi tiết.

4.4 Thiết kế Giao diện (UI/UX Design)

Giao diện người dùng (UI) và trải nghiệm người dùng (UX) được thiết kế với mục tiêu mang lại sự thân thiện, trực quan và tập trung vào quá trình sáng tạo của tác giả.

Công cụ thiết kế: Toàn bộ giao diện được thiết kế và tạo mẫu (prototype) trên Figma trước khi tiến hành lập trình. Điều này đảm bảo tính nhất quán và cho phép thu thập phản hồi sớm.

Nguyên tắc thiết kế:

- **Tối giản và Hiện đại:** Giao diện được thiết kế gọn gàng, loại bỏ các yếu tố gây nhiễu để người dùng tập trung vào việc viết. Sử dụng không gian trắng và hệ thống phân cấp trực quan rõ ràng.
- **Đáp ứng (Responsive):** Hệ thống đảm bảo hoạt động tốt và hiển thị đẹp trên nhiều kích thước màn hình khác nhau, từ máy tính để bàn đến máy tính bảng và điện thoại di động.
- **Nhất quán:** Sử dụng một hệ thống thiết kế (Design System) nhất quán về màu sắc, font chữ, và các thành phần giao diện, tạo ra trải nghiệm liền mạch trên toàn bộ ứng dụng.

Thư viện Component:

- **shadcn/ui:** Sử dụng bộ thư viện component được xây dựng trên Radix UI và Tailwind CSS. shadcn/ui cung cấp các thành phần UI chất lượng cao, dễ tùy biến và có tính přístupність (accessibility) tốt, giúp đẩy nhanh quá trình phát triển Frontend.
- **Tailwind CSS:** Sử dụng framework CSS utility-first để xây dựng giao diện một cách nhanh chóng và hiệu quả. Tailwind CSS cho phép tạo ra các thiết kế tùy chỉnh cao mà không cần viết CSS tùy chỉnh.

CHƯƠNG 5 TRIỂN KHAI VÀ CÔNG NGHỆ SỬ DỤNG

5.1 Ngăn xếp Công nghệ (Technology Stack)

Việc lựa chọn một ngăn xếp công nghệ hiện đại và phù hợp là yếu tố then chốt quyết định sự thành công của dự án. Sau quá trình nghiên cứu và cân nhắc kỹ lưỡng, dự án "Thiết kế Website Sáng tác truyện Chat tích hợp AI" đã được xây dựng trên các công nghệ và framework sau:

Bảng 5.1: Các công nghệ sử dụng trong dự án

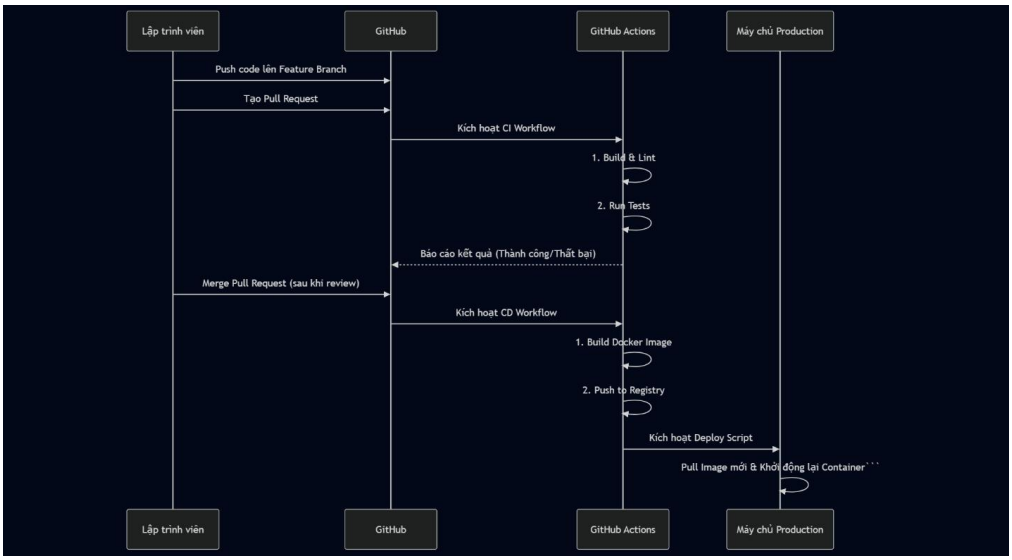
Hạng mục	Công nghệ	Lý do lựa chọn và Vai trò trong dự án
Framework Full-stack	Next.js 14+	Nền tảng chính cho toàn bộ ứng dụng. Được chọn làm giải pháp full-stack để thống nhất cả Frontend và Backend trong cùng một codebase, giúp đơn giản hóa quá trình phát triển và triển khai. Các tính năng nổi bật như Server-Side Rendering (SSR), App Router và API Routes tích hợp giúp xây dựng một ứng dụng web có hiệu năng cao, tối ưu SEO và cấu trúc mã nguồn rõ ràng.
Ngôn ngữ lập trình	TypeScript	Ngôn ngữ lập trình chính của dự án. TypeScript, với hệ thống kiểu tĩnh (static typing), mang lại lợi ích vượt trội so với JavaScript thuần túy. Nó giúp phát hiện lỗi sớm ngay trong quá trình viết code, tăng cường tính dễ đọc, dễ bảo trì và nâng cao trải nghiệm lập trình (Developer Experience) cho toàn đội ngũ, đặc biệt quan trọng trong các dự án phức tạp.
Cơ sở dữ liệu	MySQL	Hệ quản trị cơ sở dữ liệu quan hệ được chọn để lưu trữ toàn bộ dữ liệu của ứng dụng. MySQL nổi tiếng về tính ổn định, hiệu năng cao và cộng đồng hỗ trợ lớn. Mô hình quan hệ đảm bảo tính toàn vẹn dữ liệu cho các thực thể phức tạp như người dùng, truyện, nhân vật, và các mối quan hệ giữa chúng.

Hạng mục	Công nghệ	Lý do lựa chọn và Vai trò trong dự án
Giao diện người dùng	TailwindCSS & shadcn/ui	Bộ công cụ xây dựng giao diện chính. TailwindCSS cung cấp một phương pháp xây dựng giao diện nhanh chóng và linh hoạt theo triết lý utility-first, cho phép tùy biến giao diện ở mức độ chi tiết mà không cần rời khỏi file HTML/JSX. shadcn/ui được xây dựng trên TailwindCSS, cung cấp một bộ sưu tập các component UI đẹp mắt, có khả năng tùy biến cao và tuân thủ các tiêu chuẩn về khả năng truy cập (accessibility).
Xác thực	NextAuth.js	Giải pháp xác thực toàn diện cho ứng dụng. NextAuth.js là một thư viện xác thực hoàn chỉnh và linh hoạt, được thiết kế đặc biệt cho Next.js. Nó hỗ trợ nhiều phương thức xác thực như email/mật khẩu (credentials) và đăng nhập qua các nhà cung cấp bên thứ ba (OAuth) như Google, giúp quản lý phiên làm việc một cách an toàn và hiệu quả.
Trí tuệ Nhân tạo	Google Gemini API	Trái tim của các tính năng thông minh. Gemini API được tích hợp sâu vào hệ thống để cung cấp các tính năng sáng tạo như gợi ý ý tưởng, tạo sinh nội dung (cốt truyện, hội thoại, nhân vật), và tạo ảnh bìa cho truyện. Việc lựa chọn Gemini dựa trên khả năng xử lý đa phương thức và hiệu năng cao của mô hình.
Containerization	Docker & Docker Compose	Công cụ đóng gói và quản lý môi trường ứng dụng. Toàn bộ ứng dụng (web, database, MCP server) được đóng gói vào các container riêng biệt. Docker đảm bảo tính nhất quán giữa môi trường phát triển và sản phẩm. Docker Compose được sử dụng để định nghĩa và quản lý môi trường đa container, giúp khởi chạy toàn bộ hệ thống chỉ bằng một lệnh.
CI/CD	GitHub Actions	Nền tảng tự động hóa quy trình làm việc. Được tích hợp trực tiếp vào kho mã nguồn GitHub, GitHub Actions giúp tự động hóa các công việc CI/CD như build, kiểm thử và triển khai mỗi khi có sự thay đổi trên mã nguồn, đảm bảo chất lượng và tăng tốc

Hạng mục	Công nghệ	Lý do lựa chọn và Vai trò trong dự án
		độ phát hành sản phẩm.
Thanh toán	VNPay	Cổng thanh toán tích hợp. VNPay được chọn để xử lý các giao dịch cho các gói tính năng cao cấp, cung cấp phương thức thanh toán an toàn, quen thuộc và tiện lợi cho người dùng tại thị trường Việt Nam.
Giao thức Mở rộng	Model Context Protocol (MCP)	Cầu nối cho các trợ lý AI bên ngoài. Dự án triển khai một máy chủ MCP riêng biệt để tạo ra một giao thức chuẩn hóa. Điều này cho phép các trợ lý AI của bên thứ ba (ví dụ: Claude Desktop) có thể tương tác và điều khiển các chức năng của ứng dụng một cách an toàn, mở ra tiềm năng tích hợp và mở rộng hệ sinh thái trong tương lai.

5.2 Quy trình Tích hợp và Triển khai liên tục (CI/CD)

Để đảm bảo chất lượng mã nguồn và tăng tốc độ phát hành sản phẩm, dự án đã áp dụng một quy trình CI/CD hoàn chỉnh, được tự động hóa hoàn toàn bằng GitHub Actions. Quy trình này giúp giảm thiểu sai sót do con người, tự động hóa các công việc lặp lại và đảm bảo mọi thay đổi trên mã nguồn đều được kiểm tra kỹ lưỡng trước khi triển khai.



Hình 5.1: Sơ đồ quy trình tích hợp và triển khai liên tục

Các bước chi tiết trong quy trình:

Commit & Push: Lập trình viên hoàn thành một tính năng hoặc sửa lỗi trên một nhánh (branch) riêng và đẩy mã nguồn lên kho lưu trữ GitHub.

Tạo Pull Request (PR): Một PR được tạo để đề xuất hợp nhất các thay đổi từ nhánh tính năng vào nhánh chính (thường là main hoặc develop).

Kích hoạt CI Pipeline: Ngay khi PR được tạo, GitHub Actions sẽ tự động kích hoạt một workflow CI, bao gồm các công việc (jobs) sau:

- Build Application: Hệ thống tự động cài đặt các dependencies và xây dựng (build) ứng dụng Next.js. Bước này giúp xác minh rằng mã nguồn không có lỗi cú pháp và có thể biên dịch thành công.
- Linting & Type Checking: Chạy các công cụ như ESLint và TypeScript Compiler (tsc) để đảm bảo mã nguồn tuân thủ các tiêu chuẩn về định dạng (coding conventions) và an toàn về kiểu dữ liệu.
- Automated Testing: Chạy các bộ kiểm thử tự động (unit test, integration test) để xác minh rằng các thay đổi mới không phá vỡ các chức năng hiện có của hệ thống.

Code Review & Merge: Nếu tất cả các bước trong CI pipeline đều thành công (hiển thị dấu tick xanh trên PR), các thành viên khác trong nhóm sẽ tiến hành đánh giá mã nguồn (code review). Sau khi được duyệt, PR sẽ được hợp nhất vào nhánh chính.

Kích hoạt CD Pipeline (Triển khai): Việc hợp nhất vào nhánh chính sẽ kích hoạt một workflow triển khai khác, thực hiện các bước sau một cách tự động:

- **Build Docker Image:** GitHub Actions xây dựng một Docker image mới cho ứng dụng, chứa phiên bản mã nguồn mới nhất và tất cả các dependencies cần thiết.
- **Push to Registry:** Image mới được đẩy lên một Docker Registry (ví dụ: Docker Hub, GitHub Container Registry) để lưu trữ và quản lý phiên bản.
- **Deploy to Server:** Một script trên máy chủ sản phẩm (production server) sẽ được kích hoạt (thông qua SSH hoặc webhook). Script này sẽ kéo

(pull) Docker image mới nhất về và khởi động lại container của ứng dụng, hoàn tất quá trình triển khai mà không gây gián đoạn dịch vụ.

5.3 Cấu hình Docker và Containerization

Toàn bộ hệ thống được container hóa bằng Docker để đảm bảo tính nhất quán, di động và dễ dàng quản lý trên các môi trường khác nhau, từ máy tính cá nhân của lập trình viên đến máy chủ sản phẩm.

Dockerfile:

Một file Dockerfile được đặt ở thư mục gốc của dự án để định nghĩa các bước xây dựng image cho ứng dụng Next.js. File này sử dụng phương pháp multi-stage build để tối ưu hóa kích thước image, chỉ bao gồm những gì cần thiết để chạy ứng dụng ở môi trường production.

Stage 1 (builder): Cài đặt đầy đủ dependencies (bao gồm cả devDependencies), build ứng dụng để tạo ra các file tĩnh và mã nguồn đã được biên dịch.

Stage 2 (runner): Sử dụng một base image gọn nhẹ (như node:20-alpine), chỉ sao chép các file đã build từ stage builder và các dependencies cần thiết cho production. Điều này giúp giảm đáng kể kích thước image cuối cùng, tăng cường bảo mật và cải thiện tốc độ triển khai.

docker-compose.yml:

File này định nghĩa và quản lý các dịch vụ (services) đa container của ứng dụng, cho phép khởi chạy toàn bộ hệ thống bằng một lệnh duy nhất (docker-compose up). Các dịch vụ chính bao gồm:

web: Dịch vụ chính chạy ứng dụng Next.js, được build từ Dockerfile. Dịch vụ này được expose ra ngoài để người dùng có thể truy cập.

db: Dịch vụ chạy cơ sở dữ liệu MySQL, sử dụng image chính thức từ Docker Hub. Dữ liệu được lưu trữ bền vững thông qua Docker volumes, đảm bảo không bị mất ngay cả khi container bị xóa và tạo lại.

Docker Compose sẽ tự động tạo một mạng ảo (virtual network) để các container có thể giao tiếp với nhau một cách an toàn thông qua tên dịch vụ (ví dụ: dịch vụ web có thể kết nối đến db qua hostname db).

CHƯƠNG 6 QUẢN LÝ DỰ ÁN

6.1 Tổng quan Kế hoạch Scrum

Dự án "Thiết Kế Website Sáng Tác Truyện Chat Tích Hợp AI" áp dụng khung làm việc Scrum, một phương pháp phát triển phần mềm linh hoạt hiệu quả. Kế hoạch Scrum của dự án được xây dựng chi tiết với 7 Sprint với mỗi Sprint giúp tập trung vào mục đích cụ thể giúp hoàn thành web theo kế hoạch đề ra. mỗi Sprint kéo dài từ 1 đến 2 tuần, các Sprint được lập kế hoạch dựa trên Product Backlog mục đích tập trung vào các hạng mục công việc cụ thể từ Product Backlog, nhằm đạt được các mục tiêu đã đề ra và bao gồm phân công nhiệm vụ, thời gian thực hiện và điểm Story (Story Points) để đánh giá độ phức tạp của các nhiệm vụ. Mỗi Sprint bao gồm các sự kiện Scrum như Sprint Planning, Daily Scrum, Sprint Review và Sprint Retrospective để đảm bảo tính minh bạch, thanh tra và thích ứng.

Sprint Backlog là một tập hợp các hạng mục từ Product Backlog đã được chọn để thực hiện trong một Sprint, cùng với kế hoạch để đạt được Mục tiêu Sprint. Dựa trên kế hoạch chi tiết, mục tiêu của từng Sprint trong dự án "Thiết Kế Website Sáng Tác Truyện Chat Tích Hợp AI" được tóm tắt như sau:

Mục tiêu Sprint 1: Xây dựng nền tảng ban đầu của ứng dụng, tập trung hoàn thiện luồng xác thực người dùng cơ bản (đăng ký, đăng nhập, quên mật khẩu) và xây dựng bộ khung giao diện chính.

Mục tiêu Sprint 2: Mở rộng hệ thống quản lý người dùng với các tính năng nâng cao như đăng nhập qua Google, quản lý tài khoản, và tích hợp nền tảng thanh toán VNPay.

Mục tiêu Sprint 3: Tập trung phát triển các chức năng nghiệp vụ cốt lõi, cho phép người dùng có thể quản lý toàn diện vòng đời của một tác phẩm, bao gồm quản lý truyện, chương, nhân vật, và nội dung hội thoại.

Mục tiêu Sprint 4: Tích hợp sâu rộng trí tuệ nhân tạo vào các khía cạnh khác nhau của quá trình sáng tác, từ việc tạo ý tưởng, tạo hình ảnh, phát triển nhân vật cho đến chatbot trợ lý.

Mục tiêu Sprint 5: Nâng cao trải nghiệm đọc và khám phá của người dùng thông qua việc triển khai các tính năng như yêu thích, đánh dấu, theo dõi tiến độ và hệ thống tìm kiếm mạnh mẽ.

Mục tiêu Sprint 6: Tập trung vào việc phát triển trợ lý chat AI thành một công cụ sáng tác chuyên sâu, hỗ trợ tạo và chỉnh sửa các thành phần của truyện thông qua giao diện trò chuyện.

Mục tiêu Sprint 7: Hoàn thiện và tối ưu hóa sản phẩm, bao gồm cải thiện giao diện trên thiết bị di động, tích hợp giao thức MCP, và hoàn tất tài liệu kỹ thuật của dự án.

6.2 Phân công Nhiệm vụ

Sprint 1

Mục tiêu Sprint 1: Xây dựng nền tảng ban đầu của ứng dụng, tập trung hoàn thiện luồng xác thực người dùng cơ bản (đăng ký, đăng nhập, quên mật khẩu) và xây dựng bộ khung giao diện chính.

Bảng 6.1: Phân công nhiệm vụ Sprint 1

ID	Nhiệm vụ	Người thực hiện	Điểm Story	Ngày bắt đầu	Ngày kết thúc
1	Khởi tạo dự án Next.js và cấu hình cơ bản	Huỳnh Phước Thọ	3	22/02/2025	22/02/2025
2	Xây dựng hệ thống xác thực với next-auth	Huỳnh Phước Thọ	8	22/02/2025	28/02/2025
3	Tạo giao diện đăng nhập/đăng ký với shadcn/ui	Nguyễn Phú Vinh	5	22/02/2025	23/02/2025
4	Phát triển hệ thống điều hướng responsive	Nguyễn Phú Vinh	5	23/02/2025	25/02/2025
5	Xây dựng menu mobile với chế độ tối	Huỳnh Phước Thọ	3	25/02/2025	26/02/2025
6	Thêm tính năng quên mật khẩu	Nguyễn Phú Vinh	3	23/02/2025	26/02/2025
7	Tạo component footer và cập nhật giao diện	Nguyễn Phú Vinh	3	24/02/2025	26/02/2025
8	Thêm component tính năng cho trang chủ	Nguyễn Phú Vinh	3	26/02/2025	27/02/2025

Sprint 2

Mục tiêu Sprint 2: Mở rộng hệ thống quản lý người dùng với các tính năng nâng cao như đăng nhập qua Google, quản lý tài khoản, và tích hợp nền tảng thanh toán VNPay.

Bảng 6.2: Phân công nhiệm vụ Sprint 2

ID	Nhiệm vụ	Người thực hiện	Điểm Story	Ngày bắt đầu	Ngày kết thúc
9	Triển khai đăng ký người dùng với MySQL và bcrypt	Huỳnh Phước Thọ	5	27/02/2025	01/03/2025
10	Xây dựng trang cài đặt và cải thiện điều hướng	Nguyễn Phú Vinh	5	01/03/2025	02/03/2025
11	Thêm tính năng quản lý tài khoản người dùng	Huỳnh Phước Thọ	8	03/03/2025	05/03/2025
12	Tích hợp Google Drive cho lưu trữ avatar	Huỳnh Phước Thọ	5	04/03/2025	05/03/2025
13	Thêm hỗ trợ đăng nhập Google OAuth	Huỳnh Phước Thọ	5	05/03/2025	06/03/2025
14	Xây dựng hệ thống đặt lại mật khẩu qua email	Huỳnh Phước Thọ	8	05/03/2025	07/03/2025
15	Tích hợp cổng thanh toán VNPay	Nguyễn Phú Vinh	8	08/03/2025	09/03/2025
16	Thêm tính năng huy hiệu người ủng hộ	Nguyễn Phú Vinh	3	09/03/2025	10/03/2025

Sprint 3

Mục tiêu Sprint 3: Tập trung phát triển các chức năng nghiệp vụ cốt lõi, cho phép người dùng có thể quản lý toàn diện vòng đời của một tác phẩm, bao gồm quản lý truyện, chương, nhân vật, và nội dung hội thoại.

Bảng 6.3: Phân công nhiệm vụ Sprint 3

ID	Nhiệm vụ	Người thực hiện	Điểm Story	Ngày bắt đầu	Ngày kết thúc
17	Triển khai hệ thống quản lý truyện với CRUD	Huỳnh Phước Thọ	8	10/03/2025	12/03/2025
18	Xây dựng quản lý nhân vật và chi tiết truyện	Huỳnh Phước Thọ	8	11/03/2025	13/03/2025
19	Thêm chức năng quản lý chương	Huỳnh Phước Thọ	8	12/03/2025	15/03/2025
20	Tạo trang viết chương với quản lý đối thoại	Huỳnh Phước Thọ	8	12/03/2025	16/03/2025
21	Triển khai xuất bản chương và hệ thống sắp xếp	Huỳnh Phước Thọ	5	13/03/2025	15/03/2025
22	Thay thế hệ thống danh mục bằng thẻ	Nguyễn Phú Vinh	5	14/03/2025	17/03/2025
23	Phát triển thư viện truyện với điều hướng	Nguyễn Phú Vinh	8	16/03/2025	18/03/2025
24	Thêm chức năng di chuyển đối thoại	Nguyễn Phú Vinh	3	16/03/2025	17/03/2025

Sprint 4

Mục tiêu Sprint 4: Tích hợp sâu rộng trí tuệ nhân tạo vào các khía cạnh khác nhau của quá trình sáng tác, từ việc tạo ý tưởng, tạo hình ảnh, phát triển nhân vật cho đến chatbot trợ lý.

Bảng 6.4: Phân công nhiệm vụ Sprint 4

ID	Nhiệm vụ	Người thực hiện	Điểm Story	Ngày bắt đầu	Ngày kết thúc
25	Tích hợp Gemini AI cho tạo ý tưởng truyện	Nguyễn Huỳnh Phú Vinh	8	20/03/2025	22/03/2025
26	Thêm gợi ý ảnh bìa được tạo bởi AI	Nguyễn Huỳnh Phú Vinh	5	22/03/2025	23/03/2025
27	Triển khai tạo ảnh Together AI	Nguyễn Huỳnh Phú Vinh	8	22/03/2025	24/03/2025
28	Cải thiện tạo nhân vật với gợi ý AI	Nguyễn Huỳnh Phú Vinh	8	21/03/2025	25/03/2025
29	Thêm component chatbot trợ lý truyện	Nguyễn Huỳnh Phú Vinh	8	24/03/2025	26/03/2025
30	Triển khai phản hồi streaming cho chat	Nguyễn Huỳnh Phú Vinh	5	25/03/2025	26/03/2025
31	Thêm tạo truyện và nhân vật bằng AI	Nguyễn Phú Vinh	8	26/03/2025	28/03/2025

ID	Nhiệm vụ	Người thực hiện	Điểm Story	Ngày bắt đầu	Ngày kết thúc
32	Cải thiện tạo đối thoại với đa đối thoại	Nguyễn Huỳnh Phú Vinh	8	27/03/2025	29/03/2025

Sprint 5

Mục tiêu Sprint 5: Nâng cao trải nghiệm đọc và khám phá của người dùng thông qua việc triển khai các tính năng như yêu thích, đánh dấu, theo dõi tiến độ và hệ thống tìm kiếm mạnh mẽ.

Bảng 6.5: Phân công nhiệm vụ Sprint 5

ID	Nhiệm vụ	Người thực hiện	Điểm Story	Ngày bắt đầu	Ngày kết thúc
33	Triển khai chức năng yêu thích	Nguyễn Phú Vinh	5	15/04/2025	16/04/2025
34	Thêm chức năng đánh dấu và cải thiện giao diện	Huỳnh Phước Thọ	5	14/04/2025	16/04/2025
35	Thêm theo dõi đọc chương	Huỳnh Phước Thọ	5	13/04/2025	15/04/2025
36	Triển khai chức năng tìm kiếm nâng cao	Nguyễn Phú Vinh	8	17/04/2025	18/04/2025
37	Thêm tìm kiếm bằng AI với điểm liên quan	Nguyễn Huỳnh Phú Vinh	8	18/04/2025	19/04/2025
38	Thêm nhà cung cấp loading và cải thiện điều hướng	Nguyễn Phú Vinh	5	20/04/2025	21/04/2025
39	Thêm component con trỏ tùy chỉnh với hiệu ứng hover	Nguyễn Phú Vinh	3	22/04/2025	23/04/2025
40	Thêm màn hình loading ban đầu	Nguyễn Phú Vinh	3	26/04/2025	27/04/2025

Sprint 6

Mục tiêu Sprint 6: Tập trung vào việc phát triển trợ lý chat AI thành một công cụ sáng tác chuyên sâu, hỗ trợ tạo và chỉnh sửa các thành phần của truyện thông qua giao diện trò chuyện.

Bảng 6.6: Phân công nhiệm vụ Sprint 6

ID	Nhiệm vụ	Người thực hiện	Điểm Story	Ngày bắt đầu	Ngày kết thúc
41	Thêm cuộn vô hạn và thanh trượt truyện nổi bật	Nguyễn Huỳnh Phú Vinh	5	02/05/2025	03/05/2025
42	Triển khai quy trình tạo truyện với xử lý lệnh	Nguyễn Huỳnh Phú Vinh	8	11/05/2025	14/05/2025
43	Thêm lịch sử chat với hỗ trợ hình ảnh	Nguyễn Huỳnh Phú Vinh	8	10/05/2025	12/05/2025
44	Triển khai trợ lý chat AI cho phát triển truyện	Nguyễn Huỳnh Phú Vinh	8	07/05/2025	08/05/2025
45	Thêm hỗ trợ tạo chương và outline	Nguyễn Huỳnh Phú Vinh	8	15/05/2025	17/05/2025
46	Cải thiện cấu trúc dữ liệu truyện	Nguyễn Huỳnh Phú Vinh	5	15/05/2025	16/05/2025
47	Thêm chức năng tạo nhân vật vào AI chat	Nguyễn Huỳnh Phú Vinh	8	15/05/2025	18/05/2025
48	Thêm chức năng chỉnh sửa truyện	Huỳnh Phước Thọ	5	18/05/2025	19/05/2025

Sprint 7

Mục tiêu Sprint 7: Hoàn thiện và tối ưu hóa sản phẩm, bao gồm cải thiện giao diện trên thiết bị di động, tích hợp giao thức MCP, và hoàn tất tài liệu kỹ thuật của dự án.

Bảng 6.7: Phân công nhiệm vụ Sprint 7

ID	Nhiệm vụ	Người thực hiện	Điểm Story	Ngày bắt đầu	Ngày kết thúc
49	Triển khai trợ lý chat AI và tái cấu trúc components	Nguyễn Phú Vinh	8	22/05/2025	24/05/2025
50	Cải thiện khả năng phản hồi mobile và cập nhật nội dung	Nguyễn Phú Vinh	5	24/05/2025	25/05/2025
51	Thêm tích hợp MCP và cải thiện cơ sở hạ tầng	Nguyễn Huỳnh Phú Vinh	8	25/05/2025	26/05/2025
52	Thêm script kiểm tra API toàn diện	Nguyễn Huỳnh Phú Vinh	5	19/07/2025	19/07/2025
53	Cập nhật README với tài liệu toàn diện	Nguyễn Huỳnh Phú Vinh	3	19/07/2025	19/07/2025
54	Cập nhật tham chiếu repository sang vị trí mới	Nguyễn Huỳnh Phú Vinh	2	19/07/2025	19/07/2025

6.3 Quy trình Thực hiện Scrum

Mỗi Sprint được thực hiện theo quy trình Scrum tiêu chuẩn, bao gồm các sự kiện sau:

Sprint Planning: Được tổ chức vào đầu mỗi Sprint để xác định Mục tiêu Sprint và chọn các hạng mục từ Product Backlog để đưa vào Sprint Backlog. Nhóm chúng tôi thảo luận để phân công nhiệm vụ và ước lượng điểm Story.

Daily Scrum: Các cuộc họp ngắn hàng ngày (15 phút) để đồng bộ hóa tiến độ, thảo luận các vấn đề phát sinh và lập kế hoạch cho 24 giờ tiếp theo.

Sprint Review: Cuối mỗi Sprint, nhóm trình bày Increment hoàn thành để nhận phản hồi từ Product Owner và các bên liên quan, đồng thời điều chỉnh Product Backlog nếu cần.

Sprint Retrospective: Cuối Sprint, nhóm thảo luận về những gì đã làm tốt, những gì cần cải thiện và lập kế hoạch hành động cho Sprint tiếp theo.

6.4 Quản lý Rủi ro

Trong quá trình lập kế hoạch, nhóm chúng tôiđã xác định một số rủi ro tiềm ẩn và đề xuất biện pháp giảm thiểu:

Rủi ro về kỹ thuật: Tích hợp Gemini AI hoặc VNPay có thể gặp lỗi do tài liệu không đầy đủ hoặc thay đổi API. Giải pháp: Thực hiện kiểm thử tích hợp sớm và duy trì liên lạc với nhà cung cấp API.

Rủi ro về tiến độ: Một số nhiệm vụ phức tạp có thể mất nhiều thời gian hơn dự kiến. Giải pháp: Ước lượng điểm Story cẩn thận và dành thời gian đệm cho các nhiệm vụ có độ phức tạp cao.

Rủi ro về nhân sự: Thành viên nhóm có thể gặp vấn đề cá nhân hoặc lịch trình học tập. Giải pháp: Phân công nhiệm vụ linh hoạt và đảm bảo giao tiếp thường xuyên qua Daily Scrum.

Kế hoạch Scrum này đã cung cấp một lộ trình rõ ràng, giúp nhóm thực hiện dự án một cách có tổ chức, minh bạch và hiệu quả, đảm bảo hoàn thành các mục tiêu đã đề ra trong thời gian quy định.

CHƯƠNG 7 KIỂM THỬ

7.1 Chiến lược kiểm thử tổng thể

Chiến lược kiểm thử của dự án được xây dựng theo phương pháp đa tầng, nhằm đảm bảo chất lượng ở mọi cấp độ của ứng dụng, từ các đơn vị mã nguồn nhỏ nhất đến toàn bộ hệ thống khi tích hợp. Cách tiếp cận này giúp phát hiện lỗi sớm, giảm thiểu chi phí sửa chữa và đảm bảo sản phẩm cuối cùng hoạt động ổn định.

Chiến lược được chia thành ba mảng chính:

Kiểm thử API (Backend) với Postman: Tập trung vào việc xác thực logic nghiệp vụ, tính toàn vẹn dữ liệu và bảo mật ở cấp độ backend. Đây là lớp kiểm thử nền tảng, đảm bảo "xương sống" của ứng dụng hoạt động chính xác.

Kiểm thử Tích hợp Liên tục (CI) với GitHub Actions: Tự động hóa quy trình kiểm thử mỗi khi có sự thay đổi về mã nguồn. Lớp kiểm thử này đảm bảo rằng các thay đổi mới không phá vỡ các chức năng hiện có, duy trì sự ổn định của toàn bộ hệ thống.

Kiểm thử Đơn vị (Unit Test): Kiểm tra các hàm (function) và thành phần (component) riêng lẻ để đảm bảo chúng hoạt động đúng như thiết kế. Đây là lớp kiểm thử chi tiết nhất, giúp xác thực từng phần nhỏ của ứng dụng.

7.2 Kiểm thử API (Backend) với Postman

Mục tiêu: Đảm bảo tất cả các API endpoint của backend hoạt động chính xác, xử lý đúng logic nghiệp vụ, trả về đúng định dạng dữ liệu và xử lý tốt các trường hợp lỗi. Mục tiêu là xác thực tính đúng đắn của các thao tác CRUD (Create, Read, Update, Delete) và các quy trình phức tạp hơn.

Công cụ và Quy trình:

Công cụ: Postman được sử dụng làm công cụ chính để thiết kế, gửi các yêu cầu HTTP (GET, POST, PUT, DELETE) và xác thực phản hồi từ server.

Quy trình: Nhóm đã tạo một bộ sưu tập (Collection) các yêu cầu API trên Postman. Bộ sưu tập này bao gồm các kịch bản kiểm thử cho toàn bộ các chức năng cốt lõi. Mỗi yêu cầu đều đi kèm với các bài kiểm thử tự động (viết bằng JavaScript trong tab "Tests" của Postman) để xác thực các yếu tố như:

Mã trạng thái HTTP (ví dụ: 200 OK, 201 Created, 400 Bad Request, 401 Unauthorized).

Cấu trúc và kiểu dữ liệu của phản hồi JSON.

Giá trị cụ thể trong dữ liệu trả về.

Kết luận kiểm thử API: Tất cả các API endpoint chính đã được kiểm thử và cho kết quả hoạt động ổn định, đúng với logic đã thiết kế. Hệ thống xử lý tốt các trường hợp dữ liệu đầu vào hợp lệ và cả các trường hợp lỗi, đảm bảo tính toàn vẹn và bảo mật cho dữ liệu người dùng.

7.3 Kiểm thử Tích hợp Liên tục (CI) với GitHub Actions

Mục tiêu: Tự động hóa việc kiểm thử mỗi khi có sự thay đổi về mã nguồn, đảm bảo rằng các thay đổi mới không phá vỡ các chức năng hiện có của hệ thống. Việc này giúp phát hiện lỗi sớm và duy trì chất lượng mã nguồn ở mức cao.

Quy trình: Một quy trình làm việc (workflow) CI đã được cấu hình trong file .github/workflows/ci.yml để tự động kích hoạt mỗi khi một Pull Request được tạo hoặc cập nhật. Workflow này sẽ thực hiện các bước sau:

Test & Build:

Biên dịch mã nguồn TypeScript để kiểm tra lỗi cú pháp và đảm bảo mã nguồn hợp lệ.

Chạy ESLint để kiểm tra và đảm bảo mã nguồn tuân thủ các quy tắc về code style đã định sẵn.

Docker Build Test: Xây dựng một Docker image từ mã nguồn mới nhất để đảm bảo ứng dụng có thể được đóng gói và triển khai một cách chính xác.

Security Scan: Tự động quét các dependencies để phát hiện các lỗ hổng bảo mật đã biết, giúp ngăn chặn các rủi ro tiềm ẩn.

Performance Check: Chạy Lighthouse để đánh giá hiệu suất của ứng dụng, đảm bảo các thay đổi không làm giảm tốc độ tải trang và trải nghiệm người dùng.

Báo cáo kết quả: Chỉ những Pull Request vượt qua tất cả các bài kiểm thử trên mới được phép hợp nhất (merge) vào nhánh chính. Điều này đảm bảo rằng mã nguồn trên nhánh chính luôn ở trạng thái ổn định và sẵn sàng để triển khai.

Kết quả: Quy trình CI trên GitHub Actions đã được thiết lập và hoạt động hiệu quả. Mỗi Pull Request đều được tự động kiểm tra, giúp nhóm phát hiện sớm các lỗi tiềm ẩn ngay từ giai đoạn phát triển. Việc tích hợp này đã góp phần đáng kể vào việc duy trì chất lượng mã nguồn và sự ổn định của ứng dụng trong suốt quá trình phát triển theo nhóm.

7.4 Kiểm thử Đơn vị (Unit Test)

Mục tiêu: Đảm bảo các hàm (function) và thành phần (component) riêng lẻ hoạt động đúng như thiết kế, đặc biệt là các hàm xử lý logic phức tạp ở cả backend và frontend. Việc này giúp cô lập và xác thực từng phần nhỏ của ứng dụng một cách độc lập.

Phạm vi và Công cụ:

Framework (Đề xuất): Sử dụng các thư viện phổ biến như Jest hoặc Vitest để viết và quản lý các bài kiểm thử.

Phạm vi: Tập trung vào các hàm và thành phần quan trọng, có logic phức tạp, ví dụ như:

Các hàm tiện ích (/src/lib/utils.ts).

Các service xử lý logic nghiệp vụ (/src/services/).

Các hàm liên quan đến xác thực và mã hóa mật khẩu.

Các hàm tính toán và xử lý dữ liệu phức tạp.

Kết quả: Các bài kiểm thử đơn vị được tích hợp vào quy trình CI/CD. Mỗi khi có thay đổi, hệ thống sẽ tự động chạy lại các bài test này, đảm bảo rằng các hàm và thành phần cốt lõi luôn hoạt động chính xác và không bị ảnh hưởng bởi các thay đổi mới.

7.5 Tổng kết

Chiến lược kiểm thử đa tầng đã chứng tỏ được hiệu quả trong việc đảm bảo chất lượng cho dự án "Thiết Kế Website Sáng Tác Truyện Chat Tích Hợp AI". Việc kết hợp giữa kiểm thử API thủ công, kiểm thử tích hợp tự động và kiểm thử đơn vị đã giúp nhóm phát hiện và khắc phục lỗi một cách nhanh chóng, từ đó xây dựng một sản phẩm cuối cùng ổn định, an toàn và đáp ứng tốt các yêu cầu đã đề ra.

CHƯƠNG 8 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

8.1 Quá trình thực hiện

Dự án được thực hiện theo phương pháp luận phát triển phần mềm linh hoạt Scrum, với quy trình làm việc được lập kế hoạch chi tiết qua 7 Sprint. Việc áp dụng Scrum đã giúp đội ngũ làm việc một cách có hệ thống, minh bạch và hiệu quả. Kế hoạch phát triển chi tiết được trình bày trong Chương 4 đã đóng vai trò kim chỉ nam, giúp nhóm theo dõi tiến độ, phân công công việc rõ ràng giữa các thành viên với các chuyên môn khác nhau (Backend, Frontend, AI) và linh hoạt thích ứng với các thay đổi trong quá trình phát triển.

Việc sử dụng các công cụ hiện đại như Next.js đã cho phép xây dựng một ứng dụng full-stack một cách đồng nhất, trong khi Docker và Docker Compose đảm bảo môi trường phát triển và triển khai nhất quán cho tất cả các thành viên. Kho lưu trữ mã nguồn trên GitHub đóng vai trò trung tâm cho việc quản lý phiên bản và hợp tác. Nhìn chung, quá trình thực hiện dự án đã diễn ra thành công, bám sát kế hoạch và đạt được các mục tiêu đề ra cho từng Sprint.

8.2 Kết quả đạt được

Sau 7 Sprint phát triển, dự án đã xây dựng thành công Website Sáng Tác Truyện Chat Tích Hợp AI với một bộ tính năng toàn diện, đáp ứng đầy đủ các yêu cầu cốt lõi đã đặt ra. Các kết quả chính bao gồm:

Hệ thống Quản lý Người dùng và Xác thực Hoàn chỉnh:

+ Triển khai thành công chức năng đăng ký, đăng nhập bằng email/mật khẩu và xác thực qua Google.
+ Hoàn thiện quy trình đặt lại mật khẩu an toàn qua email và trang quản lý thông tin tài khoản cá nhân.
+ Tích hợp thành công cổng thanh toán VNPay và hệ thống ghi nhận người dùng cao cấp.

Nền tảng Quản lý Nội dung Truyện Chuyên sâu:

+ Xây dựng đầy đủ các chức năng tạo, đọc, cập nhật, xóa cho các thực thể cốt lõi: Truyện, Chương, và Nhân vật.

+ Hệ thống cho phép quản lý nội dung chi tiết đến từng đoạn hội thoại và cung cấp một giao diện soạn thảo trực quan.

Tích hợp Trí tuệ Nhân tạo Sâu rộng và Đa dạng:

+ Tích hợp thành công **Gemini** để hỗ trợ người dùng trong nhiều khâu sáng tác: gợi ý ý tưởng truyện, phát triển chi tiết nhân vật, và tạo sinh hội thoại theo ngữ cảnh.
+ Triển khai trợ lý Chatbot AI với cơ chế phản hồi streaming, mang lại trải nghiệm tương tác mượt mà và tự nhiên.
+ Tích hợp mô hình tạo ảnh để tạo ảnh bìa, trực quan hóa ý tưởng cho tác giả.
+ Cho phép thực thi các lệnh tạo truyện, tạo nhân vật trực tiếp từ giao diện chat AI.

Trải nghiệm Người dùng Phong phú và Thân thiện:

+ Xây dựng giao diện người dùng hiện đại, đáp ứng trên nhiều thiết bị, sử dụng thư viện Radix UI và TailwindCSS.
+ Triển khai các tính năng nâng cao trải nghiệm đọc như Yêu thích, Đánh dấu, Theo dõi tiến độ đọc và Tìm kiếm nâng cao.

Cơ sở hạ tầng Hiện đại và Khả năng Mở rộng:

+ Toàn bộ hệ thống được container hóa bằng Docker, đảm bảo tính di động và nhất quán.
+ Triển khai thành công **Model Context Protocol (MCP) Server**, mở ra một kênh tương tác hoàn toàn mới cho các trợ lý AI bên ngoài, đặt nền móng cho một hệ sinh thái mở trong tương lai.

8.3 Hạn chế của dự án

Bên cạnh những kết quả đã đạt được, dự án vẫn còn một số hạn chế cần được cải thiện trong các phiên bản tiếp theo:

+ **Kiểm thử tự động:** Mặc dù đã có các kịch bản kiểm thử API cơ bản, hệ thống vẫn chưa có một bộ kiểm thử toàn diện, đặc biệt là kiểm thử đầu cuối cho các luồng người dùng phức tạp.
+ **Bảo mật nâng cao:** Các biện pháp bảo mật nâng cao như giới hạn tần suất yêu cầu để chống tấn công DoS/DDoS, và các cơ chế giám sát bảo mật chuyên sâu vẫn chưa được triển khai.
+ **Hiệu năng và Giám sát:** Hệ thống chưa được trang bị các lớp đệm cho các truy vấn dữ liệu thường xuyên và thiếu các công cụ giám sát hiệu năng ứng dụng để theo dõi và tối ưu hóa ở môi trường sản phẩm.
+ **Tài liệu hướng dẫn người dùng:** Dự án đã có tài liệu kỹ thuật chi tiết, nhưng vẫn còn thiếu một bộ tài liệu hướng dẫn sử dụng hoàn chỉnh dành cho người dùng cuối.

8.4 Hướng phát triển trong tương lai

Để tiếp tục phát triển thành một nền tảng hàng đầu cho các nhà văn, nhóm đề xuất các hướng phát triển tiềm năng sau:

Tái cấu trúc sang Kiến trúc Microservices:

Khi hệ thống phát triển và lượng người dùng tăng lên, có thể xem xét tách các thành phần phức tạp và đòi hỏi tài nguyên cao như **dịch vụ xử lý AI** hoặc **dịch vụ thanh toán** thành các microservice độc lập. Điều này giúp tăng khả năng mở rộng, linh hoạt trong việc lựa chọn công nghệ và cải thiện tính kháng lỗi của hệ thống.

Nâng cao và Đa dạng hóa Hệ thống AI:

+ **Kiểm tra tính nhất quán:** Phát triển tính năng AI có khả năng đọc toàn bộ tác phẩm để phát hiện các mâu thuẫn trong cốt truyện hoặc sự thiếu nhất quán trong tính cách nhân vật.
+ **Phân tích và Gợi ý văn phong:** Tích hợp AI để phân tích văn phong của người viết và đưa ra các gợi ý cải thiện, hoặc thậm chí giúp "biến đổi" một đoạn văn theo phong cách của các tác giả nổi tiếng.

+ **Hỗ trợ đa ngôn ngữ:** Sử dụng khả năng dịch thuật của AI để hỗ trợ các nhà văn sáng tác và tiếp cận độc giả trên toàn thế giới.

Xây dựng Tính năng Cộng đồng và Hợp tác:

+ **Bình luận và Đánh giá:** Cho phép độc giả bình luận, xếp hạng và thảo luận về các tác phẩm được xuất bản.
+ **Chế độ hợp tác:** Phát triển tính năng cho phép nhiều tác giả cùng làm việc trên một câu chuyện theo thời gian thực, tương tự như Google Docs.
+ **Phát triển Ứng dụng Di động:**
+ Xây dựng các ứng dụng gốc cho iOS và Android để mang lại trải nghiệm tốt nhất cho người dùng trên thiết bị di động, tận dụng bộ RESTful API đã được thiết kế sẵn.

Mở rộng Hệ sinh thái MCP:

Công bố và cung cấp tài liệu chi tiết về các công cụ MCP, khuyến khích các nhà phát triển trợ lý AI bên thứ ba tích hợp với nền tảng, tạo ra một hệ sinh thái sáng tạo phong phú.

DANH MỤC TÀI LIỆU THAM KHẢO

[1]Edward Thien Hoang, "Kiến trúc một khối (Monolithic)",https://topdev.vn/blog /kien-truc-mot-khoi-monolithic/
*[2]*Do Van Nam**, "***
Đôi nét về Microservice architecture và Monolithic architecture"https://viblo.asia /p/doi-net-ve-microservice-architecture-va-monolithic-architecture-XL6lAAvrlek
[3]Nguyen Thi Hong C, "Mô hình Agile – Quy trình Scrum", https://viblo.asia/p/mo-hinh-agile-quy-trinh-scrum-LzD5dwezljY
[4]Vercel, "Next.js Documentation," Next.js by Vercel. [Online]. Available: https://nextjs.org/docs.
[5]Oracle, "MySQL 8.0 Reference Manual," MySQL Developer Zone. [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/.
[6]Docker Inc., "Docker guides," Docker. [Online]. Available: https://docs.docker.com/guides/
[7]Google, "Gemini API Documentation," Google AI for Developers, 2024. [Online] . : https://ai.google.dev/docs/gemini_api_overview.
[8]K. Schwaber và J. Sutherland, "The Scrum Guide," Scrum.org, 2020. [Online]. https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Vietnamese.pdf.