

TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG

KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI:

**XÂY DỰNG WEBSITE ĐỌC TRUYỆN TÍCH HỢP
CHATBOT AI GỢI Ý VÀ TÓM TẮT NỘI DUNG.**

Người hướng dẫn: TS. NGUYỄN NĂNG HÙNG VÂN

Sinh viên thực hiện: NGUYỄN VĂN UY

Mã số sinh viên: 2151220254

Lớp: 21CT5

Đà Nẵng, 11/2025

TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI:

**XÂY DỰNG WEBSITE ĐỌC TRUYỆN TÍCH HỢP
CHATBOT AI GỢI Ý VÀ TÓM TẮT NỘI DUNG.**

Người hướng dẫn: TS. NGUYỄN NĂNG HÙNG VÂN

Sinh viên thực hiện: NGUYỄN VĂN UY

Mã số sinh viên: 2151220254

Lớp: 21CT5

Đà Nẵng, 11/2025

NHẬN XÉT GIẢNG VIÊN HƯỚNG DẪN

NHẬN XÉT GIẢNG VIÊN PHẢN BIỆN

TÓM TẮT

Tên đề tài: Xây dựng website đọc truyện tích hợp chatbot AI gợi ý và tóm tắt nội dung.

Sinh viên thực hiện: Nguyễn Văn Uy

Mã số SV: 2151220254 Lớp: 21CT5

Trong kỷ nguyên số hóa, nhu cầu giải trí trực tuyến đang chứng kiến sự tăng trưởng vượt bậc, kéo theo đòi hỏi ngày càng cao về tính cá nhân hóa trong trải nghiệm người dùng. Mặc dù đọc truyện trực tuyến là một hình thức giải trí phổ biến, các nền tảng hiện hành vẫn tồn tại nhiều hạn chế trong việc hỗ trợ độc giả khám phá nội dung. Các hệ thống gợi ý truyền thống chủ yếu vận hành dựa trên việc so khớp từ khóa hoặc siêu dữ liệu cơ bản (như tên truyện, thể loại), thiếu khả năng thấu hiểu ngữ cảnh sâu và nhu cầu phức tạp của người đọc. Hạn chế này dẫn đến việc người dùng mất nhiều thời gian tìm kiếm, làm giảm hứng thú và mức độ gắn kết với nền tảng.

Nhằm giải quyết thách thức trên, đề tài “Xây dựng Website đọc truyện tích hợp Chatbot AI gợi ý và tóm tắt nội dung” đề xuất một giải pháp công nghệ tiên tiến, kết hợp giữa nền tảng web truyền thống và trí tuệ nhân tạo. Trọng tâm nghiên cứu là việc xây dựng và tích hợp một Chatbot thông minh dựa trên kiến trúc RAG (Retrieval-Augmented Generation). Khác với các mô hình ngôn ngữ thông thường có thể sinh ra thông tin sai lệch (ảo giác), kiến trúc RAG cho phép hệ thống truy xuất dữ liệu chính xác từ cơ sở tri thức nội bộ của website (như cốt truyện, nhân vật, bối cảnh) trước khi sinh câu trả lời. Cơ chế này đảm bảo Chatbot hoạt động như một "thủ thư ảo" am hiểu, có khả năng tóm tắt nội dung súc tích và đưa ra các gợi ý truyện phù hợp với ngữ cảnh tìm kiếm tự nhiên của người dùng.

Kết quả nghiên cứu không chỉ mang lại một sản phẩm ứng dụng hoàn chỉnh với trải nghiệm người dùng vượt trội, mà còn chứng minh tính khả thi của việc ứng dụng các kỹ thuật AI hiện đại (Vector Search, LLM) vào bài toán thực tế quy mô vừa và nhỏ. Đề tài đóng góp một giải pháp công nghệ có tính cạnh tranh cao, đáp ứng xu hướng cá nhân hóa trong giải trí số, đồng thời góp phần thúc đẩy văn hóa đọc lành mạnh trong cộng đồng.

LỜI NÓI ĐẦU

Trong bối cảnh chuyển đổi số mạnh mẽ, thói quen giải trí trực tuyến đang chứng kiến sự thay đổi sâu sắc với sự lên ngôi của các nền tảng video ngắn và nội dung tiêu thụ nhanh (short-form content). Xu hướng này, mặc dù mang lại sự tiện lợi và tính giải trí tức thời, nhưng cũng đặt ra những thách thức không nhỏ đối với văn hóa đọc – một hình thức giải trí đòi hỏi sự tập trung và tư duy sâu sắc. Các nền tảng đọc truyện trực tuyến, dù sở hữu kho tàng nội dung phong phú và giá trị, đang đối mặt với áp lực cạnh tranh gay gắt trong việc thu hút và giữ chân người dùng, đặc biệt là thế hệ độc giả trẻ.

Một trong những nguyên nhân kỹ thuật dẫn đến sự suy giảm sức hút của các nền tảng đọc truyện hiện hành là sự hạn chế trong khả năng cá nhân hóa trải nghiệm. Các hệ thống tìm kiếm và gợi ý truyền thống (Recommender Systems) chủ yếu vận hành dựa trên cơ chế so khớp từ khóa hoặc lọc cộng tác đơn giản theo thể loại. Cách tiếp cận này thường bỏ qua ngữ cảnh sâu xa trong nhu cầu của độc giả, dẫn đến việc các đề xuất trở nên máy móc và thiếu chính xác. Hệ quả là người dùng phải tiêu tốn nhiều thời gian để sàng lọc nội dung, làm gián đoạn mạch cảm xúc và giảm đáng kể mức độ gắn kết với ứng dụng.

Nhận thức được thực trạng đó, đề tài “Xây dựng Website đọc truyện tích hợp Chatbot AI gợi ý và tóm tắt nội dung” được đề xuất như một giải pháp công nghệ nhằm tái định hình trải nghiệm đọc truyện trong kỷ nguyên số. Trọng tâm nghiên cứu của đề án là việc ứng dụng kiến trúc Retrieval-Augmented Generation (RAG) để xây dựng một trợ lý ảo thông minh. Khác với các công cụ tìm kiếm thụ động, Chatbot này có khả năng thấu hiểu ngôn ngữ tự nhiên để phân tích các yêu cầu phức tạp của người dùng về cốt truyện, bối cảnh hay nhân vật. Từ đó, hệ thống thực hiện truy xuất dữ liệu chính xác từ kho tri thức nội bộ để đưa ra các gợi ý tác phẩm phù hợp nhất, đồng thời cung cấp các bản tóm tắt nội dung súc tích theo yêu cầu.

Thông qua việc tích hợp trí tuệ nhân tạo vào quy trình khám phá nội dung, đề án không chỉ giải quyết bài toán kỹ thuật về tìm kiếm ngữ nghĩa mà còn hướng tới mục tiêu xã hội rộng lớn hơn: nâng cao chất lượng trải nghiệm đọc, khơi dậy niềm đam mê văn học và góp phần duy trì thói quen đọc lành mạnh trong cộng đồng số.

LỜI CẢM ƠN

Để hoàn thành được đồ án tốt nghiệp này, tôi đã nhận được rất nhiều sự giúp đỡ, chỉ bảo và động viên.

Trước hết, tôi xin bày tỏ lòng biết ơn sâu sắc nhất tới thầy TS. NGUYỄN NĂNG HÙNG VÂN người đã tận tình hướng dẫn, định hướng và truyền đạt những kiến thức quý báu cho tôi trong suốt quá trình thực hiện đề tài.

Tôi cũng xin trân trọng cảm ơn Ban Giám hiệu Trường đại học Kiến Trúc Đà Nẵng, Ban Chủ nhiệm Khoa Công Nghệ Thông Tin cùng toàn thể quý thầy cô đã dạy dỗ, tạo điều kiện và trang bị cho tôi những nền tảng kiến thức vững chắc trong những năm học vừa qua.

Cuối cùng, tôi xin gửi lời cảm ơn chân thành đến gia đình, bạn bè đã luôn ở bên cạnh, động viên và là nguồn cổ vũ tinh thần to lớn để tôi có thể vượt qua mọi khó khăn và hoàn thành tốt đồ án của mình.

Mặc dù đã rất cố gắng, song do kiến thức và kinh nghiệm còn hạn chế, đồ án chắc chắn không thể tránh khỏi những thiếu sót. Tôi rất mong nhận được những ý kiến đóng góp quý báu từ quý thầy cô và các bạn để đề tài được hoàn thiện hơn.

CAM ĐOAN

Tôi tên là Nguyễn Văn Uy, sinh viên lớp 21CT5, Khoa Công nghệ Thông tin Trường Đại học Kiến trúc Đà Nẵng. Tôi xin cam đoan rằng đồ án tốt nghiệp với đề tài “Xây dựng website đọc truyện tích hợp chatbot AI gợi ý và tóm tắt nội dung” là sản phẩm nghiên cứu độc lập do chính tôi thực hiện, có sự hướng dẫn tận tình của TS. Nguyễn Năng Hùng Vân.

Toàn bộ nội dung, dữ liệu và kết quả trình bày trong đồ án đều được thu thập, phân tích và trình bày một cách trung thực, đảm bảo tính khách quan và liêm chính học thuật. Các nguồn tài liệu, trích dẫn được ghi rõ ràng, chính xác theo quy định của Trường Đại học Kiến trúc Đà Nẵng.

Tôi khẳng định đồ án không sao chép, không vi phạm bản quyền, và hoàn toàn chịu trách nhiệm về tính trung thực, chính xác của toàn bộ nội dung. Mọi góp ý, phản hồi để hoàn thiện đề tài, tôi đều trân trọng ghi nhận.

Sinh viên thực hiện

MỤC LỤC

TÓM TẮT	
LỜI NÓI ĐẦU	i
LỜI CẢM ƠN	ii
CAM ĐOAN	iii
MỤC LỤC	iv
DANH SÁCH CÁC BẢNG, HÌNH VẼ.....	ix
DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT	xii
MỞ ĐẦU.....	1
1.1 Lý do chọn đề tài	1
1.2 Mục đích và ý nghĩa của đề tài	1
1.3 Mục tiêu đề tài	2
1.4 Đối tượng và phạm vi nghiên cứu	2
1.5 Phương pháp nghiên cứu	2
1.6 Cấu trúc đồ án	3
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT	4
1.1 Tổng quan đề tài	4
1.1.1 Bối cảnh bài toán và nhu cầu thực tiễn.....	4
1.1.2 Phân tích các giải pháp hiện có và hạn chế.....	4
1.1.3 Hướng tiếp cận mới.....	5
1.2 Tổng quan về trí tuệ nhân tạo và xử lý ngôn ngữ tự nhiên.....	5
1.2.1 Sự phát triển của Chatbot và AI hội thoại	5
1.2.2 Mô hình ngôn ngữ lớn	7
1.3 Kỹ thuật Retrieval-Augmented Generation	9
1.3.1 Khái niệm và định nghĩa	10

1.3.2	<i>Kiến trúc RAG tiêu chuẩn</i>	10
1.3.3	<i>Quy trình thực thi chi tiết</i>	12
1.3.4	<i>So sánh RAG với phương pháp Fine-tuning</i>	12
1.4	<i>Cơ sở dữ liệu Vector và kỹ thuật Embedding</i>	13
1.4.1	<i>Biểu diễn văn bản</i>	14
1.4.2	<i>Độ đo tương đồng</i>	15
1.4.3	<i>Cơ sở dữ liệu Vector và Thuật toán tìm kiếm</i>	16
1.5	<i>Các kỹ thuật hỗ trợ trong hệ thống RAG</i>	17
1.5.1	<i>Chiến lược phân đoạn văn bản</i>	17
1.5.2	<i>Phân loại ý định người dùng</i>	18
1.5.3	<i>Kỹ thuật thiết kế câu lệnh</i>	19
1.6	<i>Công cụ xây dựng hệ thống</i>	19
1.6.1	<i>Nền tảng Python 3.10 và Framework FastAPI</i>	19
1.6.2	<i>Ngôn ngữ phát triển Website: PHP - Hypertext Preprocessor</i>	21
1.6.3	<i>Cursor IDE</i>	21
1.6.4	<i>Hệ quản trị Cơ sở dữ liệu</i>	23
1.6.5	<i>Các thư viện và Framework hỗ trợ nghiệp vụ AI</i>	24
1.6.6	<i>Quản lý phiên bản mã nguồn với Git</i>	25
CHƯƠNG 2: PHƯƠNG PHÁP ĐỀ XUẤT XÂY DỰNG HỆ THỐNG CHATBOT RAG GỢI Ý VÀ TÓM TẮT NỘI DUNG		26
2.1	<i>Phương pháp đề xuất</i>	26
2.1.1	<i>Tổng quan phương pháp</i>	27
2.1.2	<i>Cơ chế hoạt động của hệ thống gợi ý truyện tranh</i>	28
2.1.3	<i>Quy trình xử lý của Chatbot RAG</i>	29
2.1.4	<i>Hiệu quả cộng hưởng của mô hình lai</i>	30
2.1.5	<i>Lưu đồ thuật toán tổng quát và Đánh giá giải pháp</i>	30
2.2	<i>Thu thập và tạo dữ liệu Chatbot</i>	32

2.2.1	<i>Nguồn dữ liệu</i>	33
2.2.2	<i>Dữ liệu truyện tranh</i>	33
2.2.3	<i>Dữ liệu người dùng</i>	33
2.2.4	<i>Dữ liệu hội thoại cho Chatbot</i>	34
2.2.5	<i>Quy trình tổng hợp, định dạng và vector hóa dữ liệu</i>	34
2.3	<i>Tiền xử lý và phân đoạn dữ liệu</i>	35
2.3.1	<i>Làm sạch dữ liệu</i>	36
2.3.2	<i>Chuẩn hóa văn bản</i>	36
2.3.3	<i>Phân đoạn nội dung</i>	37
2.4	<i>Phân tích và lựa chọn mô hình nhúng</i>	37
2.4.1	<i>Giới thiệu các mô hình tiềm năng</i>	38
2.4.2	<i>Bảng so sánh thông số kỹ thuật và hiệu năng</i>	38
2.4.3	<i>Phân tích và lập luận lựa chọn</i>	39
2.5	<i>Môi trường và kịch bản kiểm thử</i>	39
2.5.1	<i>Cấu hình thử nghiệm</i>	39
2.5.2	<i>Bộ dữ liệu kiểm thử (Test Dataset)</i>	40
2.5.3	<i>Kết quả đánh giá hệ thống gợi ý</i>	42
2.6	<i>Kết quả thực nghiệm</i>	43
2.6.1	<i>Đánh giá độ chính xác</i>	43
2.6.2	<i>Đánh giá hiệu năng</i>	44
2.7	<i>Phân tích và Thảo luận</i>	45
2.7.1	<i>So sánh với các phương pháp truyền thống</i>	45
2.7.2	<i>Phân tích tính khả thi kinh tế</i>	46
CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG		47
3.1	<i>Ý tưởng nghiên cứu và Phân tích Yêu cầu Hệ thống</i>	47
3.1.1	<i>Ý tưởng nghiên cứu</i>	47
3.1.2	<i>Phân tích Yêu cầu Hệ thống</i>	47

3.2	Thiết kế kiến trúc hệ thống	49
3.2.1	Sơ đồ kiến trúc tổng quan.....	49
3.2.2	Sơ đồ Usecase	51
3.2.3	Sơ đồ Triển khai hệ thống	53
3.2.4	Sơ đồ tuần tự chức năng hỏi Chatbot.....	55
3.2.5	Sơ đồ hoạt động hỏi Chatbot.....	57
3.2.6	Sơ đồ lớp chức năng Chatbot.....	59
3.2.7	Sơ đồ hoạt động đọc truyện.....	61
3.2.8	Sơ đồ tuần tự chức năng đọc truyện.....	63
3.2.9	Sơ đồ tuần tự Admin quản lý.....	66
3.3	Thiết kế cơ sở dữ liệu	71
3.3.1	Cơ sở dữ liệu quan hệ	72
3.3.2	Cơ sở dữ liệu Vector.....	74
3.4	Thiết kế Giải thuật và Luồng xử lý dữ liệu (Algorithm Design).....	75
3.4.1	Quy trình xử lý dữ liệu	75
3.4.2	Chiến lược tìm kiếm lai.....	76
CHƯƠNG 4: XÂY DỰNG WEBSITE ĐỌC TRUYỆN TÍCH HỢP CHATBOT AI GỢI Ý VÀ TÓM TẮT NỘI DUNG		79
4.1	Môi trường phát triển và triển khai.....	79
4.1.1	Cấu hình phân cứng và Hạ tầng triển khai.....	79
4.1.2	Nền tảng phần mềm và công cụ phát triển.....	80
4.1.3	Môi trường thực thi và thư viện phụ thuộc	80
4.2	Xây dựng website	81
4.2.1	Giao diện người dùng.....	81
4.2.2	Giao diện quản trị	89
4.3	Triển khai AI RAG Pipeline	97
4.3.1	Chuẩn bị và Vector hóa dữ liệu	97

4.3.2	<i>Hiện thực hóa phân loại ý định</i>	<i>100</i>
4.3.3	<i>Kỹ thuật Prompt Engineering và Tiêm ngữ cảnh.....</i>	<i>101</i>
4.3.4	<i>Tích hợp Mô hình Tạo sinh</i>	<i>103</i>
4.3.5	<i>Kết quả triển khai và đánh giá hiệu năng</i>	<i>107</i>
KẾT LUẬN VÀ ĐỊNH HƯỚNG PHÁT TRIỂN		110
TÀI LIỆU THAM KHẢO		112

DANH SÁCH CÁC BẢNG, HÌNH VẼ

Bảng 1.1: Bảng phân loại nhóm chatbot	6
Bảng 1.2: Bảng so sánh RAG với phương pháp Fine-tuning	12
Bảng 2.1: Bảng So Sánh Chi Tiết Phương Pháp Đề Xuất Và Truyền Thống. ...	32
Bảng 2.2: Bảng so sánh thông số kỹ thuật và hiệu năng của các mô hình	38
Bảng 2.3 Bảng Kết Quả Thực Nghiệm Gợi Ý Truyện	43
Bảng 2.4: Bảng So Sánh Kết Quả Thực Nghiệm Tính Trung Thực.....	44
Bảng 2.5: Bảng So Sánh Chatbot RAG Với Các Chatbot Truyền Thống	45
Bảng 4.1: Tổng hợp các chỉ số hiệu năng và độ chính xác.....	108
Hình 1.1 Hình ảnh mô tả sự phá triển của Chatbot [22].	6
Hình 1.2 Sơ đồ mô phỏng bộ mã hóa Transformer [22].....	8
Hình 1.3 Sơ đồ mô tả quy trình hoạt động của hệ thống RAG [22].	10
Hình 1.4 Sơ đồ luồng hoạt động của kiến trúc RAG [2].	11
Hình 1.5 Sơ đồ mô tả cách sinh vector database và embedding [11].	14
Hình 1.6: Hình ảnh logo của FastAPI [15].	19
Hình 1.7: Hình ảnh logo của PHP [17].	21
Hình 1.8: Hình ảnh logo của Cursor [20].....	22
Hình 1.9: Hình ảnh logo của MySQL [18].	23
Hình 1.10: Hình ảnh logo của ChromaDB [13].....	23
Hình 1.11: Hình ảnh logo của Git.	25
Hình 2.1: Sơ Đồ Tổng Quan Phương Pháp Đề Xuất Hybrid RAG.	26
Hình 2.2: Lưu Đồ Thuật Toán Xử Lý Truy Vấn.....	31
Hình 3.1: Sơ Đồ Kiến Trúc Tổng Quan Của Hệ Thống.	50
Hình 3.2: Sơ Đồ Usecase Tổng Quan Của Hệ Thống.	52

Hình 3.3: Sơ Đồ Triển Khai Hệ Thống.....	53
Hình 3.4: Sơ Đồ Tuần Tự Hoạt Động "Hỏi Chatbot AI".	56
Hình 3.5: Sơ Đồ Hoạt Động Chức Năng "Hỏi Chatbot AI".....	57
Hình 3.6: Sơ Đồ Lớp Chức Năng Chatbot AI.....	59
Hình 3.7: Sơ Đồ Hoạt Động Đọc Truyện.	61
Hình 3.8: Sơ Đồ Tuần Tự Xem Chi Tiết Truyện.....	63
Hình 3.9: Sơ Đồ Tuần Tự Đọc Chapter.	65
Hình 3.10: Sơ Đồ Truy Cập Admin Dashboard.....	66
Hình 3.11: Sơ Đồ Admin quản lý truyện	68
Hình 3.12: Sơ Đồ Admin Quản Lý User.....	69
Hình 3.13: Sơ Đồ Admin Quản Lý Chapter.	70
Hình 3.14: Sơ Đồ Thực Thể Liên Kết.....	72
Hình 3.15: Sơ Đồ Vector Schema.....	74
Hình 3.15: Sơ Đồ Quy Trình Xử Lý Dữ Liệu.....	76
Hình 3.16: Sơ Đồ Mô Tả Tìm Kiếm Lại.....	77
Hình 4.1: Giao Diện Trang Chủ Dạng LightMode	81
Hình 4.2: Giao Diện Trang Chủ Dạng DarkMode.....	82
Hình 4.3: Giao Diện Trang Chi Tiết Truyện.....	83
Hình 4.4: Giao Diện Trang Đọc Truyện.	84
Hình 4.5: Giao Diện Trang Truyện Hot.....	85
Hình 4.6: Giao Diện Trang Lịch Sử Đọc Truyện	86
Hình 4.7: Giao Diện Trang Truyện Đang Theo Dõi	86
Hình 4.8: Giao Diện Trang Chọn Thẻ Loại.	87
Hình 4.9: Giao Diện Trang Truyện Dành Cho Nam.....	88
Hình 4.10: Giao Diện Trang Truyện Dành Cho Nữ.	89

Hình 4.11: Giao Diện Trang Admin Dashboard	90
Hình 4.12: Giao Diện Trang Quản Lý Truyện.....	91
Hình 4.13: Giao Diện Trang Quản Lý Người Dùng.	92
Hình 4.14: Giao Diện Trang Quản Lý Chương.	92
Hình 4.15: Giao Diện Trang Thêm Truyện.....	93
Hình 4.16: Giao Diện Trang Thêm Chương Truyện.....	94
Hình 4.17: Giao Diện Trang Sửa Thông Tin Truyện.....	95
Hình 4.18: Giao Diện Trang Sửa Thông Tin Chương.	96
Hình 4.19: Giao Diện Widget Chatbot.....	97

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

1. Ký hiệu (Symbols)

Ký hiệu	Tên gọi / Diễn giải
A	Tập hợp các câu trả lời (Answer Set) do hệ thống sinh ra.
C	Ngữ cảnh (Context) được cung cấp cho mô hình ngôn ngữ lớn.
d	Số chiều của vector nhúng. Trong đồ án này, $d = 1536$.
K	Số lượng kết quả trả về trong thuật toán tìm kiếm Top-K.
M	Số lượng liên kết tối đa của mỗi nút trong đồ thị HNSW.
N	Tổng số lượng phần tử (vector hoặc bản ghi) trong cơ sở dữ liệu.
Q	Câu truy vấn (Query) của người dùng bằng ngôn ngữ tự nhiên.
V_q	Vector biểu diễn của câu truy vấn Q .
$\mathbb{R}$	Tập hợp số thực (Real numbers).
$\cos(\emptyset)$	Giá trị Cosine Similarity dùng để đo độ tương đồng giữa hai vector.

2. Chữ viết tắt

Viết tắt	Cụm từ đầy đủ	Tên gọi / Diễn giải
ACID	Atomicity, Consistency, Isolation, Durability	Tính chất giao dịch cơ sở dữ liệu.
AI	Artificial Intelligence	Trí tuệ nhân tạo.

API	Application Programming Interface	Giao diện lập trình ứng dụng.
ASGI	Asynchronous Server Gateway Interface	Giao diện cổng máy chủ bất đồng bộ (Chuẩn giao tiếp Python Web).
CRUD	Create, Read, Update, Delete	Bốn thao tác cơ bản với dữ liệu: Thêm, Xem, Sửa, Xóa.
CSDL	Database	Cơ sở dữ liệu.
DBMS	Database Management System	Hệ quản trị cơ sở dữ liệu (ví dụ: MySQL).
ERD	Entity Relationship Diagram	Sơ đồ thực thể liên kết (Mô hình hóa dữ liệu quan hệ).
ETL	Extract, Transform, Load	Quy trình Trích xuất, Chuyển đổi và Tải dữ liệu vào kho lưu trữ.
GPT	Generative Pre-trained Transformer	Mô hình Transformer được huấn luyện trước có khả năng tạo sinh.
HNSW	Hierarchical Navigable Small World	Thế giới nhỏ có khả năng điều hướng phân cấp (Thuật toán tìm kiếm vector).
HTTP	Hypertext Transfer Protocol	Giao thức truyền tải siêu văn bản (Giao tiếp Web cơ bản).

IDE	Integrated Development Environment	Môi trường phát triển tích hợp (ví dụ: Cursor).
JSON	JavaScript Object Notation	Định dạng dữ liệu đối tượng JavaScript (Chuẩn trao đổi dữ liệu).
LLM	Large Language Model	Mô hình ngôn ngữ lớn (ví dụ: GPT-4).
MAP	Mean Average Precision	Độ chính xác trung bình (Chỉ số đánh giá xếp hạng gợi ý).
MVC	Model-View-Controller	Mô hình kiến trúc phần mềm Mẫu hình - Giao diện - Điều khiển.
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên.
OAuth	Open Authorization	Chuẩn mở về ủy quyền truy cập (Dùng cho đăng nhập Google).
OOP	Object-Oriented Programming	Lập trình hướng đối tượng.
RAG	Retrieval-Augmented Generation	Tạo sinh được tăng cường bởi truy xuất (Kiến trúc AI lai).
RDBMS	Relational Database Management System	Hệ quản trị cơ sở dữ liệu quan hệ (ví dụ: MariaDB).
REST	Representational State Transfer	Kiến trúc chuyển giao trạng thái đại diện (Chuẩn thiết kế API).

SCM	Source Control Management	Quản lý kiểm soát mã nguồn (ví dụ: Git).
SQL	Structured Query Language	Ngôn ngữ truy vấn có cấu trúc (Dùng cho CSDL quan hệ).
UI/UX	User Interface / User Experience	Giao diện người dùng / Trải nghiệm người dùng.

MỞ ĐẦU

1.1 Lý do chọn đề tài

Trong thời đại công nghệ số phát triển mạnh mẽ, trí tuệ nhân tạo (AI) đang trở thành động lực chính thúc đẩy sự thay đổi trong nhiều lĩnh vực, đặc biệt là ngành công nghiệp nội dung số. Một minh chứng nổi bật cho ứng dụng của AI là sự xuất hiện của các hệ thống Chatbot, đóng vai trò là trợ lý ảo giúp tăng cường khả năng tương tác và hỗ trợ người dùng.

Mặc dù vậy, các nền tảng đọc truyện tranh trực tuyến hiện nay vẫn gặp phải thách thức lớn trong việc cá nhân hóa trải nghiệm cho độc giả. Hầu hết các hệ thống gợi ý và tìm kiếm chỉ dựa vào phương pháp so khớp từ khóa đơn giản hoặc lọc cộng tác, dẫn đến hạn chế trong việc nhận diện nhu cầu sâu xa của người đọc và khiến các đề xuất trở nên kém chính xác, nhàm chán [9]. Điều này buộc người dùng phải dành nhiều thời gian để tìm kiếm tác phẩm phù hợp, khiến trải nghiệm giảm đi sự hấp dẫn và gắn kết.

Xuất phát từ thực tiễn trên, đề tài “Xây dựng website đọc truyện tích hợp Chatbot AI gợi ý và tóm tắt nội dung” hướng đến việc giải quyết những hạn chế của hệ thống truyền thống bằng một giải pháp công nghệ hiện đại. Ứng dụng kiến trúc RAG (Retrieval-Augmented Generation), hệ thống sẽ cho phép người dùng tìm kiếm truyện dựa trên mô tả cốt truyện, bối cảnh hoặc cảm xúc, đồng thời cung cấp khả năng tóm tắt nội dung truyện một cách thông minh [2]. Đây là bước tiến giúp nâng cao sự cá nhân hóa, gia tăng giá trị cho hệ sinh thái đọc truyện trực tuyến [5].

1.2 Mục đích và ý nghĩa của đề tài

Mục đích: Nghiên cứu và xây dựng một hệ thống website đọc truyện tranh tích hợp Chatbot AI có khả năng hiểu ngôn ngữ tự nhiên, thực hiện gợi ý truyện dựa trên ngữ cảnh (tình tiết, nhân vật, cảm xúc) và tóm tắt nội dung truyện súc tích, chính xác [6], [12].

Ý nghĩa khoa học và thực tiễn:

- Về mặt công nghệ: Chứng minh hiệu quả của việc ứng dụng kỹ thuật RAG trong lĩnh vực giải trí, giải quyết bài toán ảo giác (hallucination) của các mô hình ngôn ngữ lớn [4].

- Về mặt xã hội: Thúc đẩy thói quen đọc sách/truyện, làm cho quá trình tiếp cận tác phẩm trở nên thông minh, thú vị, góp phần xây dựng đời sống tích cực với các hình thức giải trí nhanh hiện nay.

1.3 Mục tiêu đề tài

- Phân tích, thiết kế và xây dựng nền tảng website đọc truyện hoàn chỉnh (Full-stack) với chức năng quản lý nội dung và người dùng [17], [18].
- Nghiên cứu chuyên sâu về Xử lý ngôn ngữ tự nhiên (NLP), Mô hình ngôn ngữ lớn (LLMs) và kỹ thuật Truy xuất - Tăng cường - Tạo sinh (RAG) để làm cơ sở lý thuyết cho hệ thống [1], [5].
- Xây dựng, tích hợp module Chatbot AI vào website, sử dụng cơ sở dữ liệu Vector và các thuật toán tìm kiếm ngữ nghĩa để cung cấp tính năng gợi ý và tóm tắt nội dung [3], [13].

1.4 Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu:

- Lý thuyết: Kỹ thuật học sâu trong NLP, kiến trúc Transformer làm nền tảng cho LLM [1], cơ chế hoạt động của RAG [2], Vector Database (ChromaDB) [13], và kỹ thuật Prompt Engineering [6].
- Ứng dụng: Quy trình phát triển phần mềm Web (PHP/MySQL) [17], kiến trúc Microservices cho AI (Python/FastAPI) [15].

Phạm vi nghiên cứu:

- Dữ liệu: Giới hạn trong miền dữ liệu truyện tranh, tập trung vào xử lý thông tin văn bản (metadata, tóm tắt) của 42 bộ truyện mẫu để xây dựng cơ sở tri thức [10].
- Công nghệ: Sử dụng các mô hình đã huấn luyện sẵn (Pre-trained models) của OpenAI gồm GPT-4o-mini cho tạo sinh và text-embedding-3-small cho vector hóa, kết hợp với kỹ thuật RAG [8], [11], [12].
- Chức năng: Tập trung vào các tính năng cốt lõi của website và Chatbot; không bao gồm các chức năng phụ trợ phức tạp như thanh toán hay diễn đàn thảo luận sâu.

1.5 Phương pháp nghiên cứu

- Nghiên cứu lý thuyết: Tổng hợp, phân tích các tài liệu khoa học uy tín về NLP, RAG, Vector Search để xây dựng nền tảng kiến thức vững chắc cho đề án [1], [9].

- Nghiên cứu thực nghiệm: Thiết kế, triển khai hệ thống thực tế, xây dựng kịch bản kiểm thử, thu thập số liệu đánh giá độ chính xác và hiệu năng trên tập dữ liệu mẫu dựa trên các tiêu chuẩn đánh giá hiện hành [7].
- Phương pháp so sánh: Đối chiếu hiệu quả giữa phương pháp tìm kiếm ngữ nghĩa (Semantic Search) và tìm kiếm từ khóa truyền thống để làm rõ ưu điểm của giải pháp đề xuất [10].
- Phương pháp ứng dụng công cụ: Sử dụng các công cụ hỗ trợ phát triển hiện đại như Cursor IDE và trợ lý ảo Gemini để tối ưu hóa mã nguồn và quy trình nghiên cứu [20], [22].

1.6 Cấu trúc đồ án

- Chương 1 - Cơ sở lý thuyết: Chương này tập trung hệ thống hóa các kiến thức nền tảng và cơ sở khoa học phục vụ cho việc nghiên cứu và phát triển hệ thống. Nội dung bao gồm việc phân tích bối cảnh bài toán, tổng quan về Trí tuệ nhân tạo (AI), Xử lý ngôn ngữ tự nhiên (NLP) và đi sâu vào kiến trúc Truy xuất - Tăng cường - Tạo sinh (RAG) – công nghệ cốt lõi của đồ án.
- Chương 2 - Phương pháp xây dựng hệ thống: Trình bày rõ ràng các bước phân tích yêu cầu, quá trình lựa chọn kiến trúc lai phù hợp và thiết kế cơ sở dữ liệu. Tất cả nhằm hướng đến tối ưu hóa hiệu quả hoạt động của hệ thống và nâng cao trải nghiệm người dùng trên website và Chatbot.
- Chương 3 - Phân tích và Thiết kế hệ thống: Tiến hành phân tích chi tiết các yêu cầu kỹ thuật, đề xuất giải pháp kiến trúc lai, thiết kế cơ sở dữ liệu cùng các luồng xử lý nghiệp vụ chính. Chương này đóng vai trò làm cầu nối giữa lý thuyết và thực tiễn triển khai.
- Chương 4 - Xây dựng website đọc truyện tích hợp chatbot AI gợi ý và tóm tắt nội dung: Trình bày quy trình triển khai hệ thống, từ việc thiết lập môi trường phát triển, lựa chọn công nghệ phù hợp, đến xây dựng chi tiết Pipeline RAG và tích hợp các module chức năng cần thiết cho website đọc truyện và chatbot AI hỗ trợ người dùng.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1.1 Tổng quan đề tài

1.1.1 Bối cảnh bài toán và nhu cầu thực tiễn

Trong thập kỷ qua, ngành công nghiệp truyện tranh trực tuyến đã chứng kiến sự tăng trưởng bùng nổ về cả số lượng tác phẩm lẫn quy mô độc giả. Sự chuyển dịch từ định dạng in ấn truyền thống sang các nền tảng số đã tạo ra những kho dữ liệu khổng lồ, nơi người dùng có thể tiếp cận hàng nghìn bộ truyện chỉ với vài cú nhấp chuột. Tuy nhiên, sự phong phú này cũng dẫn đến vấn đề "quá tải thông tin" (Information Overload). Người đọc ngày càng gặp khó khăn trong việc tìm kiếm các tác phẩm phù hợp với sở thích cá nhân giữa một "biển" nội dung hỗn độn.

Nhu cầu của người dùng hiện đại không còn dừng lại ở việc tìm kiếm thụ động theo tên truyện hay tác giả. Họ mong muốn một trải nghiệm tương tác thông minh hơn: tìm kiếm dựa trên mô tả cốt truyện ("truyện có nhân vật chính thông minh"), tìm kiếm theo cảm xúc ("truyện đọc để giải trí, hài hước"), hoặc thậm chí là yêu cầu tóm tắt nội dung để nắm bắt nhanh diễn biến trước khi quyết định đọc. Đây là những nhu cầu thực tiễn mà các hệ thống quản lý truyện tranh truyền thống chưa đáp ứng được trọn vẹn.

1.1.2 Phân tích các giải pháp hiện có và hạn chế

Trong bối cảnh các nền tảng đọc truyện trực tuyến tại Việt Nam đang phát triển mạnh, phần lớn hệ thống hiện nay vẫn dựa trên các giải pháp tìm kiếm và gợi ý truyền thống, bộc lộ nhiều hạn chế đáng kể:

- **Tìm kiếm dựa trên từ khóa (Keyword-based Search):** Đây là phương pháp phổ biến nhất, vận hành bằng các thuật toán so khớp chuỗi như SQL LIKE hoặc Full-text Search. Mặc dù có ưu điểm đơn giản và tốc độ nhanh, phương pháp này tồn tại nhược điểm lớn là thiếu khả năng hiểu ngữ nghĩa[9]. Ví dụ, khi người dùng tìm "truyện đánh nhau", hệ thống sẽ không thể tự động suy luận để trả về các truyện thuộc nhóm "hành động" hoặc "võ thuật" nếu các từ khóa này không xuất hiện chính xác trong văn bản.
- **Lọc cộng tác (Collaborative Filtering):** Một số nền tảng ứng dụng kỹ thuật này để gợi ý truyện dựa trên hành vi tương đồng của đám đông (ví dụ: "Người đọc truyện A thường đọc truyện B"). Tuy nhiên, kỹ thuật này gặp phải vấn đề "Khởi động lạnh" (Cold Start) – không thể đưa ra gợi ý chính xác cho người dùng mới

hoặc các bộ truyện mới phát hành chưa có dữ liệu tương tác [10]. Hơn nữa, phương pháp này bỏ qua hoàn toàn việc phân tích nội dung (content) của tác phẩm, dẫn đến các gợi ý thiếu chiều sâu ngữ nghĩa.

1.1.3 Hướng tiếp cận mới

Để khắc phục các nhược điểm trên, xu hướng công nghệ hiện đại đang chuyển dịch sang việc ứng dụng Trí tuệ nhân tạo tạo sinh (Generative AI) kết hợp với Tìm kiếm ngữ nghĩa (Semantic Search). Đề tài này đề xuất một cách tiếp cận đột phá dựa trên kiến trúc RAG (Retrieval-Augmented Generation) [2], [5].

Khác với các phương pháp truyền thống, hướng tiếp cận này coi nội dung văn bản (tóm tắt, mô tả, đánh giá) là nguồn dữ liệu chính. Bằng cách chuyển đổi văn bản thành các vector số học (Text Embedding), hệ thống có thể "hiểu" được nội dung truyện ở mức độ ngữ nghĩa sâu sắc [10]. Kết hợp với khả năng suy luận của các Mô hình ngôn ngữ lớn (LLMs), hệ thống không chỉ tìm kiếm chính xác hơn mà còn có khả năng tương tác, giải đáp và tóm tắt thông tin như một tư vấn viên thực thụ [4], [6]. Đây là cơ sở thực tiễn và khoa học vững chắc để đề án tập trung nghiên cứu sâu về RAG trong các phần tiếp theo.

1.2 Tổng quan về trí tuệ nhân tạo và xử lý ngôn ngữ tự nhiên

Trí tuệ nhân tạo (Artificial Intelligence - AI) và Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) đã có những bước tiến vượt bậc trong thập kỷ qua, chuyển dịch từ các hệ thống dựa trên luật (rule-based) sang các mô hình học sâu (deep learning) với khả năng hiểu và sinh ngữ nghĩa ấn tượng. Trong bối cảnh đề án này, chúng ta tập trung nghiên cứu về các tác nhân hội thoại (Conversational Agents) và các mô hình ngôn ngữ lớn (LLMs), nền tảng cốt lõi của hệ thống chatbot hiện đại [8].

1.2.1 Sự phát triển của Chatbot và AI hội thoại

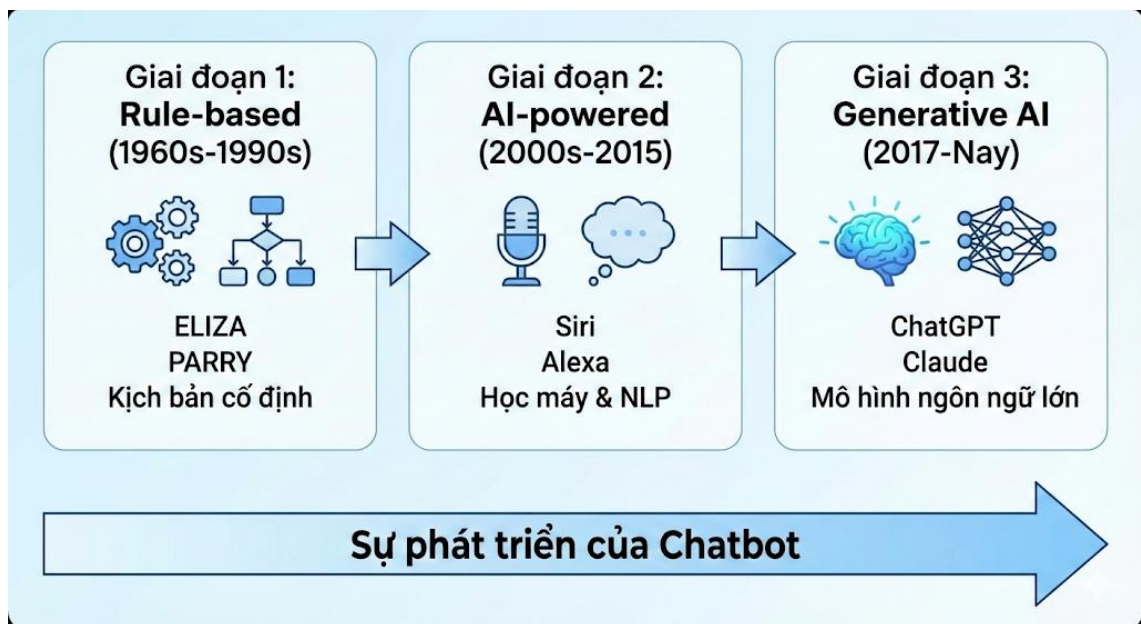
1.2.1.1 Lịch sử phát triển

Chatbot xuất hiện lần đầu vào những năm 1960 với hệ thống ELIZA (1966) của Joseph Weizenbaum tại MIT. ELIZA sử dụng kỹ thuật khớp mẫu (pattern matching) đơn giản để phản hồi người dùng, tạo cảm giác như đang trò chuyện với một nhà trị liệu tâm lý, mặc dù thực chất không hiểu ngữ nghĩa của lời nói.

Bước ngoặt lớn trong lĩnh vực AI hội thoại đến từ năm 2017 khi Google giới thiệu kiến trúc Transformer qua bài báo "Attention Is All You Need" (Vaswani et al., 2017). Cơ chế "Self-Attention" giúp máy tính hiểu và sinh ngữ nghĩa văn bản với độ chính xác vượt trội [1].

Quá trình phát triển của Chatbot có thể chia thành ba giai đoạn chính:

- Giai đoạn 1 (1960s - 1990s): Các hệ thống dựa trên quy tắc (Rule-based) như ELIZA, PARRY. Chúng hoạt động cứng nhắc, chỉ xử lý các kịch bản được định sẵn.
- Giai đoạn 2 (2000s - 2015): Sự xuất hiện của Trợ lý ảo (Virtual Assistants) như Siri (Apple), Alexa (Amazon), Google Assistant. Các hệ thống này tích hợp nhận dạng giọng nói và xử lý ngôn ngữ tự nhiên cơ bản để thực hiện các tác vụ.
- Giai đoạn 3 (2017 - nay): Kỷ nguyên Generative AI dựa trên kiến trúc Transformer. Chatbot như ChatGPT, Claude (Anthropic) có khả năng sinh văn bản trôi chảy, hiểu ngữ cảnh phức tạp và thực hiện các suy luận đa bước [8].



Hình 1.1 Hình Ảnh Mô Tả Sự Phát Triển Của Chatbot [22].

1.2.1.2 Phân loại Chatbot

Dựa trên cơ chế hoạt động, Chatbot được phân loại thành hai nhóm chính:

Bảng 1.1: Bảng Phân Loại nhóm Chatbot

Tiêu chí	Chatbot dựa trên luật (Rule-based)	Chatbot dựa trên AI (Generative AI)
Cơ chế	Sử dụng cây quyết định, Regular Expressions (Regex).	Sử dụng Mô hình ngôn ngữ lớn (LLMs) và Mạng nơ-ron (Neural Networks).

Ưu điểm	Độ chính xác cao trong phạm vi hẹp, chi phí thấp, dễ kiểm soát.	Linh hoạt, hiểu ngôn ngữ tự nhiên, xử lý được các câu hỏi chưa từng gặp.
Nhược điểm	Cứng nhắc, không thể xử lý các câu hỏi ngoài kịch bản.	Có thể sinh thông tin sai lệch (Hallucination), chi phí vận hành cao [4].
Đại diện	Các menu hỗ trợ khách hàng truyền thống.	ChatGPT, Gemini, Claude.

Trong đồ án này, hệ thống được thiết kế theo mô hình lai (Hybrid). Cách tiếp cận này tận dụng sức mạnh của Generative AI để tạo ra tương tác tự nhiên với người dùng, đồng thời kết hợp với cơ chế Retrieval (truy xuất) nhằm đảm bảo độ chính xác của thông tin cung cấp [2].

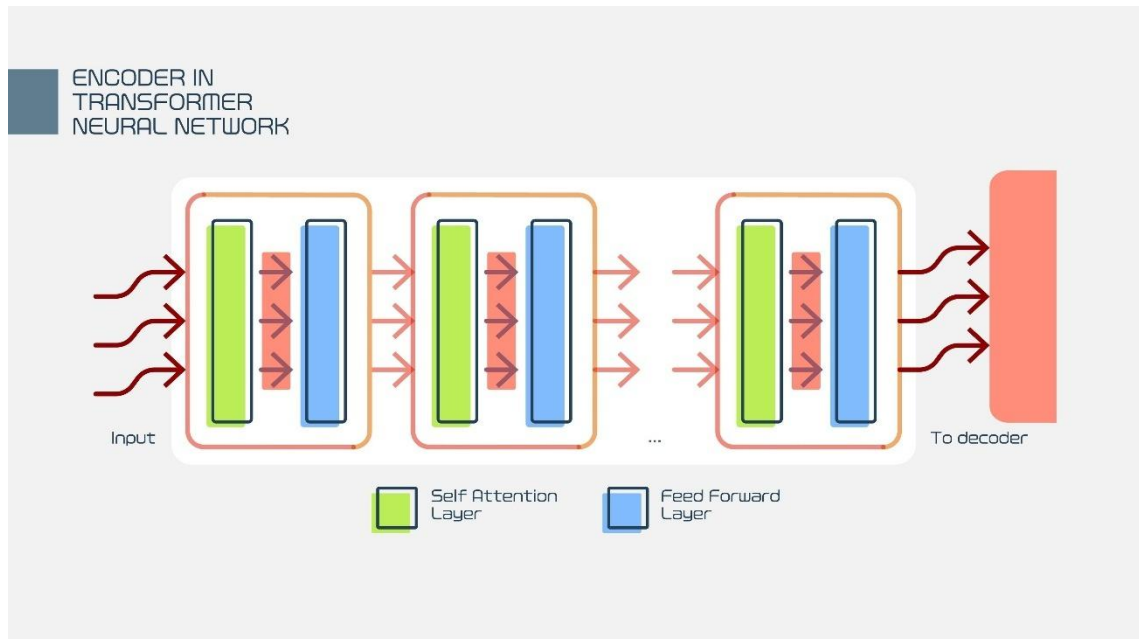
1.2.2 Mô hình ngôn ngữ lớn

Mô hình ngôn ngữ lớn (LLMs) là các hệ thống học sâu được huấn luyện trên khối lượng dữ liệu văn bản rất lớn, giúp chúng phát triển khả năng hiểu và sinh ngôn ngữ tự nhiên ở mức cao. Trong đồ án này, các mô hình thuộc dòng GPT (Generative Pre-trained Transformer) của OpenAI được lựa chọn để triển khai, nhằm tận dụng sức mạnh xử lý và khả năng tương tác linh hoạt với người dùng [8].

1.2.2.1 Kiến trúc Transformer và GPT

Kiến trúc nền tảng của các LLM hiện đại là Transformer, lần đầu tiên được Google giới thiệu trong bài báo "Attention Is All You Need" (Vaswani et al., 2017). Điểm nổi bật của Transformer chính là cơ chế Self-Attention (Tự chú ý), giúp mô hình xác định mức độ quan trọng của từng từ trong câu dựa trên toàn bộ ngữ cảnh, từ đó khắc phục được hạn chế của các mạng xử lý tuần tự như RNN hay LSTM trước đây [1].

Mô hình GPT (Generative Pre-trained Transformer) do OpenAI phát triển dựa trên kiến trúc Transformer Decoder-only. Một trong những phiên bản mới nhất được sử dụng trong đồ án là GPT-4, cụ thể là biến thể gpt-4o-mini, sở hữu quy mô lớn và hỗ trợ đa phương thức (multimodal) [8], [12].



Hình 1.2 Sơ Đồ Mô Phỏng Transformer [22].

1.2.2.2 Cơ chế hoạt động cơ bản

Quá trình vận hành của một LLM như GPT bao gồm hai giai đoạn chính:

- Tiền huấn luyện (Pre-training): Mô hình được huấn luyện trên tập dữ liệu văn bản đồ sộ như internet, sách, báo... với mục tiêu dự đoán từ tiếp theo trong chuỗi văn bản (next-token prediction). Qua đó, mô hình học hỏi ngữ pháp, kiến thức thế giới và rèn luyện khả năng suy luận nền tảng [8].
- Tinh chỉnh (Fine-tuning): Sau khi hoàn tất tiền huấn luyện, mô hình tiếp tục được tinh chỉnh bằng kỹ thuật RLHF (Reinforcement Learning from Human Feedback). Quá trình này nhằm đảm bảo các câu trả lời của mô hình trở nên hữu ích, trung thực và an toàn, phù hợp với ý định con người [6].

1.2.2.3 Thách thức của LLM

Mặc dù có sức mạnh ấn tượng, việc sử dụng các LLM đã được huấn luyện sẵn (pre-trained) một cách độc lập cho các tác vụ chuyên biệt bộc lộ nhiều hạn chế nghiêm trọng.

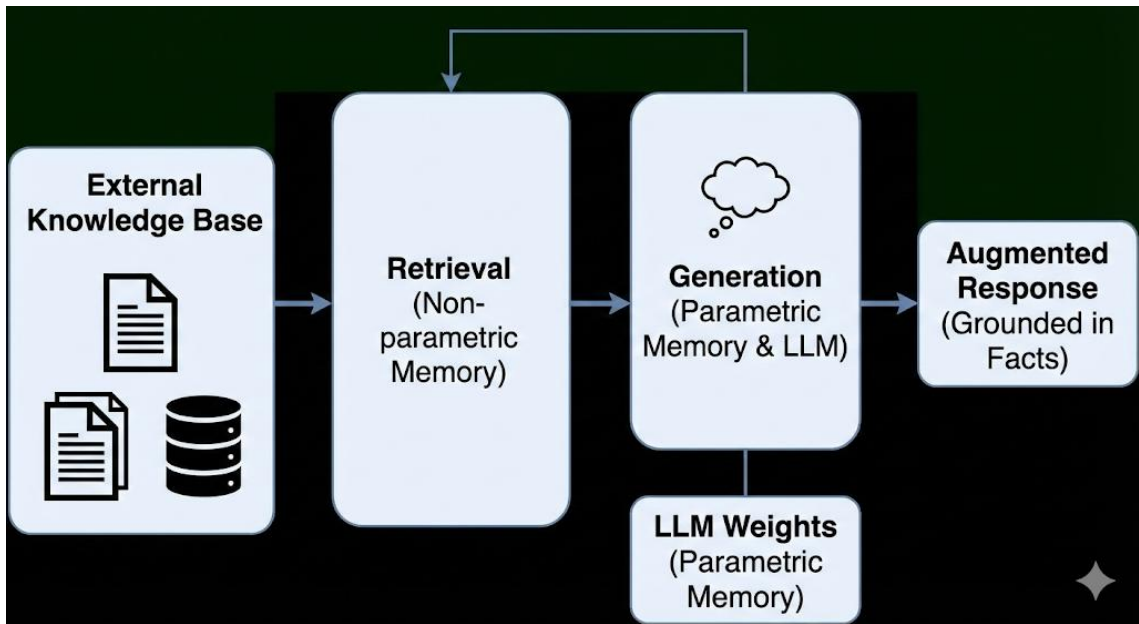
- Hiện tượng "ảo giác" (Hallucination): Đây là một trong những thách thức lớn nhất. LLM về bản chất là các mô hình tạo sinh, chúng ưu tiên tạo ra các chuỗi văn bản mạch lạc và *có vẻ hợp lý* về mặt thống kê, chứ không ưu tiên tính chính xác (fact-checking). Do đó, chúng thường xuyên tự "bịa" ra thông tin (make up information), đưa ra các sự thật, số liệu, tên nhân vật, hoặc tình tiết không chính xác nhưng lại được diễn đạt rất thuyết phục [4].

- **Giới hạn kiến thức (Knowledge Cutoff):** Quá trình huấn luyện (pre-training) một LLM tiêu tốn chi phí tính toán khổng lồ và thời gian dài. Do đó, kiến thức của mô hình bị "đóng băng" (frozen) tại thời điểm dữ liệu huấn luyện kết thúc (ví dụ: một số mô hình có knowledge cutoff vào năm 2021 hoặc 2023). Chúng hoàn toàn không thể biết về các sự kiện, tin tức, hay dữ liệu mới phát sinh sau mốc thời gian đó [5].
- **Thiếu kiến thức chuyên miền (Lack of Domain-Specific Knowledge):** Đây là hạn chế chí mạng khi ứng dụng LLM vào các tác vụ cụ thể của một tổ chức. Dữ liệu huấn luyện chung (như Wikipedia, Common Crawl) không hề chứa thông tin nội bộ hoặc chuyên biệt của một website. Ví dụ, LLM sẽ không biết gì về nội dung các bộ truyện, tên nhân vật, hay các tình tiết cụ thể đang có trên website đọc truyện của đề tài này. Nếu được hỏi, nó sẽ hoặc là trả lời không biết, hoặc là "ảo giác" ra một câu trả lời sai [2].

Từ ba hạn chế nêu trên, có thể thấy việc sử dụng LLM độc lập cho các tác vụ đòi hỏi tính chính xác và kiến thức cập nhật là không khả thi và rủi ro cao. Chính những hạn chế này là động lực trực tiếp cho việc áp dụng kiến trúc RAG (Retrieval-Augmented Generation), sẽ được trình bày ở mục tiếp theo. RAG là giải pháp cho phép "neo" LLM vào một nguồn kiến thức bên ngoài (như cơ sở dữ liệu truyện của website), cung cấp thông tin ngữ cảnh cần thiết và cập nhật, buộc mô hình phải tạo sinh câu trả lời dựa trên dữ liệu thực tế, qua đó giải quyết đồng thời cả ba vấn đề: "ảo giác", "giới hạn kiến thức" và "thiếu kiến thức chuyên miền" [2], [5].

1.3 Kỹ thuật Retrieval-Augmented Generation

Trong bối cảnh các mô hình ngôn ngữ lớn (LLMs) hiện đại thường gặp khó khăn với vấn đề ảo giác (hallucination) – tức hiện tượng sinh ra thông tin sai lệch một cách tự tin – và giới hạn về dữ liệu lỗi thời (knowledge cutoff), kỹ thuật Retrieval-Augmented Generation (RAG) đã nổi lên như một giải pháp tiêu chuẩn [4], [5]. RAG khắc phục các hạn chế này bằng cách kết hợp sức mạnh tạo sinh ngôn ngữ tự nhiên của LLM với khả năng tra cứu chính xác từ nguồn dữ liệu bên ngoài.



Hình 1.3 Sơ Mô Tả Quy Trình Hoạt Động Của RAG [22].

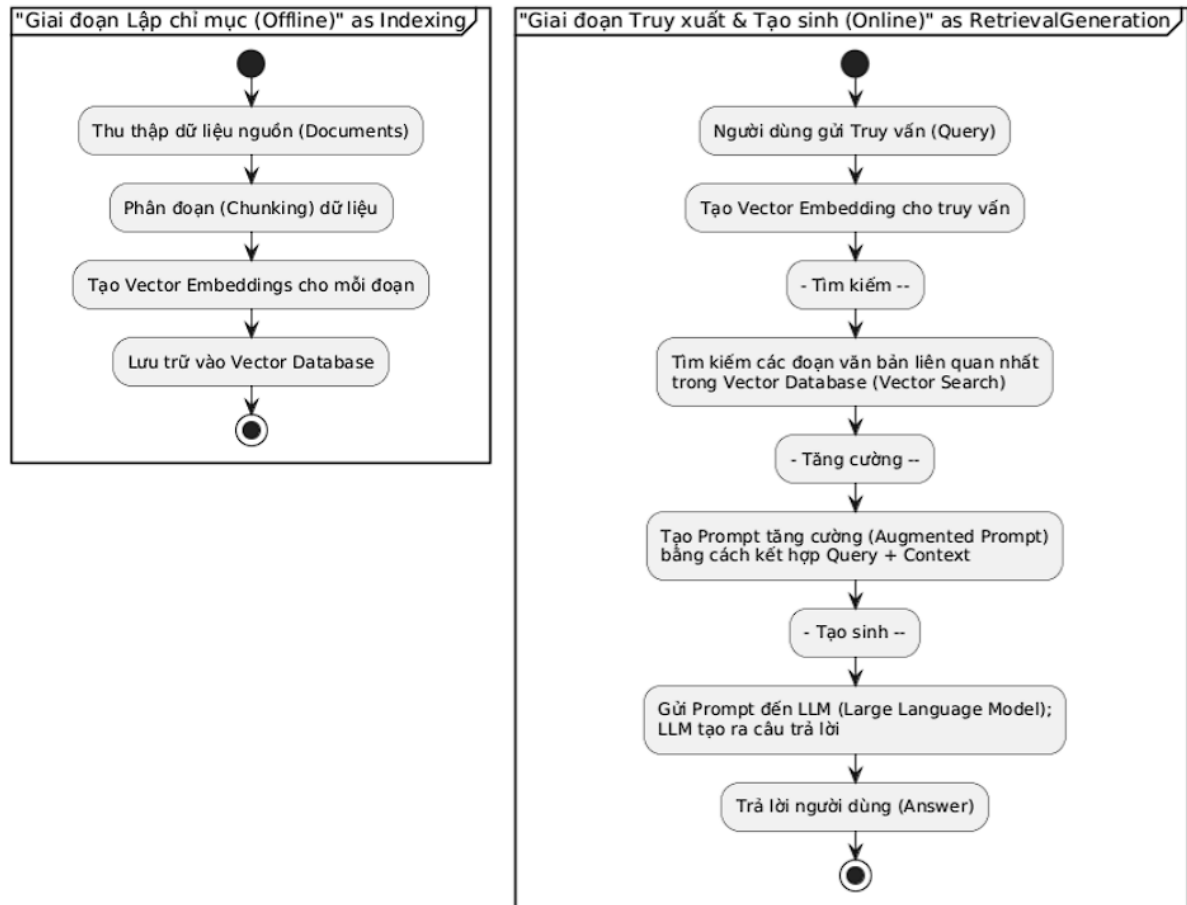
1.3.1 Khái niệm và định nghĩa

Retrieval-Augmented Generation (RAG) là một khung kiến trúc kết hợp giữa hai phương pháp trong lĩnh vực Xử lý ngôn ngữ tự nhiên: truy xuất thông tin (information retrieval) dựa trên mật độ vector và sinh văn bản (text generation) dựa trên xác suất. Khái niệm RAG được giới thiệu lần đầu bởi Lewis và cộng sự vào năm 2020 trong nghiên cứu "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks" [2]. Theo định nghĩa này, RAG trao cho mô hình ngôn ngữ khả năng truy cập vào bộ nhớ phi tham số (non-parametric memory) để hỗ trợ cho bộ nhớ tham số vốn có.

Về mặt cấu trúc, bộ nhớ tham số (parametric memory) là kiến thức được lưu trữ trong hàng tỷ tham số (trọng số) của mô hình LLM sau quá trình huấn luyện tiền nghiệm, thường ở trạng thái tĩnh và khó cập nhật. Ngược lại, bộ nhớ phi tham số (non-parametric memory) là cơ sở dữ liệu tri thức bên ngoài, chẳng hạn như cơ sở dữ liệu vector chứa thông tin về các bộ truyện tranh, cho phép mô hình truy xuất trực tiếp tại thời điểm thực thi [2]. Cốt lõi của RAG là việc sử dụng các tài liệu được truy xuất để làm giàu (augment) đầu vào cho mô hình sinh, giúp câu trả lời không chỉ dựa trên "trí nhớ" của AI mà còn dựa trên bằng chứng thực tế [5].

1.3.2 Kiến trúc RAG tiêu chuẩn

Một hệ thống RAG tiêu chuẩn bao gồm hai thành phần chính phối hợp chặt chẽ: Retriever (Bộ truy xuất) và Generator (Bộ sinh văn bản), tạo thành quy trình khép kín từ truy vấn đến phản hồi.



Hình 1.4 Sơ Đồ Kiến Trúc RAG.

- Bộ truy xuất (Retriever): Thành phần này chịu trách nhiệm tìm kiếm các phân đoạn văn bản (documents/chunks) liên quan nhất đến truy vấn của người dùng từ cơ sở tri thức. Hệ thống sử dụng kỹ thuật tìm kiếm vector dày đặc (Dense Vector Retrieval), chuyển đổi truy vấn đầu vào thành vector truy vấn thông qua mô hình embedding [10]. Sau đó, các thuật toán tối ưu như HNSW được dùng để tính toán độ tương đồng (Cosine Similarity) giữa vector truy vấn và các vector tài liệu trong cơ sở dữ liệu, từ đó chọn ra Top-K tài liệu phù hợp nhất [3]. Trong phạm vi đồ án, Retriever được hiện thực hóa qua cơ sở dữ liệu ChromaDB kết hợp với mô hình embedding text-embedding-3-small của OpenAI để tối ưu hóa khả năng tìm kiếm ngữ nghĩa [11], [13].
- Bộ sinh văn bản (Generator): Đây là một mô hình ngôn ngữ lớn (LLM) nhận đầu vào gồm truy vấn gốc của người dùng và ngữ cảnh do Retriever cung cấp. Generator sẽ mô hình hóa ngôn ngữ nhằm tối đa hóa xác suất sinh ra chuỗi văn bản đích dựa trên cả truy vấn và ngữ cảnh. Mục tiêu là tạo ra câu trả lời tự nhiên, đồng thời tuân thủ chặt chẽ thông tin có trong ngữ cảnh. Đồ án sử dụng mô hình

gpt-4o-mini làm Generator để tổng hợp thông tin và phản hồi người dùng bằng ngôn ngữ tự nhiên [12].

1.3.3 Quy trình thực thi chi tiết

Để xử lý hiệu quả, hệ thống không làm việc trực tiếp với văn bản thô mà chuyển đổi truy vấn thành vector số thực (Query Embedding) thông qua mô hình text-embedding-3-small, tạo ra biểu diễn ngữ nghĩa trong không gian 1536 chiều [11]. Vector này được dùng làm đầu vào cho bước truy xuất ngữ nghĩa trong cơ sở dữ liệu vector (ChromaDB), nơi thuật toán HNSW tính toán độ tương đồng (Cosine Similarity) và xác định Top-K đoạn văn bản liên quan nhất với truy vấn, thay vì chỉ so khớp từ khóa như các phương pháp truyền thống [3], [13].

Những đoạn thông tin thô sau khi được truy xuất sẽ được làm giàu thông qua kỹ thuật Prompt Engineering. Cụ thể, các đoạn này được lồng ghép vào một khuôn mẫu chứa chỉ dẫn hệ thống (System Instructions) cùng câu hỏi gốc, hình thành ngữ cảnh hoàn chỉnh và có cấu trúc rõ ràng nhằm hướng dẫn mô hình suy luận đúng hướng [6]. Cuối cùng, mô hình ngôn ngữ lớn tiếp nhận ngữ cảnh đã được tăng cường để thực hiện tổng hợp thông tin, sinh ra câu trả lời tự nhiên, chính xác và có căn cứ xác thực, góp phần giảm thiểu hiện tượng ảo giác thường xuất hiện ở các mô hình sinh văn bản truyền thống [4].

1.3.4 So sánh RAG với phương pháp Fine-tuning

Trong việc tùy biến LLM cho các tác vụ chuyên biệt (domain-specific), Fine-tuning (tinh chỉnh mô hình) và RAG là hai hướng tiếp cận phổ biến nhất. Bảng dưới đây so sánh chi tiết hai phương pháp này dựa trên các tiêu chí kỹ thuật và vận hành, làm rõ lý do lựa chọn RAG cho đề án [5]:

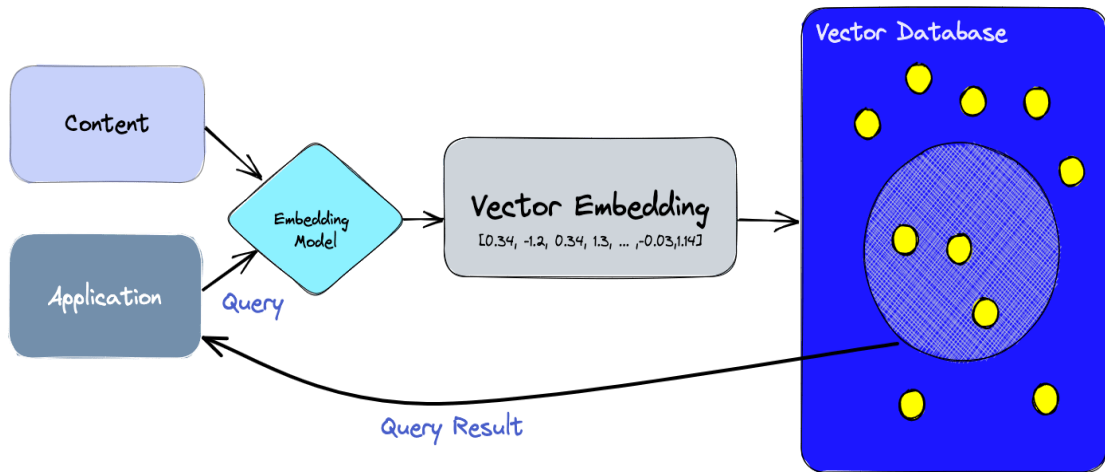
Bảng 1.2: Bảng So Sánh RAG Với Phương Pháp Fine-tuning

Tiêu chí	Fine-tuning (Tinh chỉnh)	Retrieval-Augmented Generation (RAG)
Cập nhật kiến thức (Data Freshness)	Thấp: Kiến thức bị đóng băng tại thời điểm huấn luyện lại. Muốn cập nhật phải retrain mô hình, tốn thời gian và tài nguyên.	Cao: Dữ liệu luôn mới nhất. Chỉ cần cập nhật hoặc thêm tài liệu vào Vector Database là chatbot có thể trả lời

		ngay lập tức mà không cần train lại model.
Tính trung thực & Ảo giác (Hallucination)	Thấp: Mô hình vẫn có thể "bịa" ra thông tin sai dù đã được học thêm dữ liệu mới, do bản chất xác suất của việc sinh từ.	Cao: Giảm thiểu ảo giác đáng kể vì mô hình bị buộc phải trả lời dựa trên ngữ cảnh được cung cấp. Có khả năng trích dẫn nguồn.
Chi phí (Cost)	Cao: Tốn chi phí tính toán lớn (GPU) cho quá trình training và lưu trữ các bản sao model riêng biệt.	Thấp/Trung bình: Chi phí thiết lập thấp, chủ yếu tốn chi phí token khi gọi API (do prompt dài hơn kèm context) và chi phí duy trì Vector DB.
Kiểm soát dữ liệu (Data Privacy)	Khó xóa bỏ một thông tin cụ thể khi nó đã "ngấm" vào trọng số của model.	Dễ dàng thêm/sửa/xóa từng đoạn thông tin trong cơ sở dữ liệu.
Ứng dụng phù hợp	Khi cần thay đổi phong cách, giọng điệu, hoặc format của câu trả lời (Form).	Khi cần trả lời chính xác dựa trên dữ liệu,kiến thức chuyên ngành

1.4 Cơ sở dữ liệu Vector và kỹ thuật Embedding

Để máy tính có thể hiểu và xử lý ngôn ngữ tự nhiên, văn bản cần được chuyển đổi thành dạng dữ liệu số học. Trong các hệ thống RAG hiện đại, hai thành phần không thể thiếu để thực hiện quá trình này là kỹ thuật nhúng văn bản (Text Embedding) và Cơ sở dữ liệu Vector (Vector Database) [10].



Hình 1.5 Sơ Đồ Mô Tả Cách Sinh Vector Database Và Embedding [11].

1.4.1 Biểu diễn văn bản

Biểu diễn văn bản là quá trình mã hóa các đơn vị từ ngữ thành các vector số thực. Lịch sử phát triển của kỹ thuật này đã trải qua hai giai đoạn chính:

- **Biểu diễn thưa (Sparse Representation):** Trong các phương pháp truyền thống như Bag-of-Words (BoW) hay TF-IDF, văn bản được biểu diễn dưới dạng các vector thưa với số chiều bằng kích thước của từ điển. Đặc điểm của các vector này là chứa chủ yếu giá trị 0 và không nắm bắt được mối quan hệ ngữ nghĩa giữa các từ. Ví dụ, hai câu "*Doraemon là mèo máy*" và "*Mèo ú đến từ tương lai*" có thể bị coi là hoàn toàn khác nhau do ít từ chung, dù cùng đề cập đến một đối tượng [10].
- **Biểu diễn dày (Dense Representation):** Sự ra đời của các mô hình học sâu (như Word2Vec, BERT và GPT) đã giới thiệu khái niệm Vector dày (Dense Vector). Mỗi đoạn văn bản được chuyển đổi thành một vector có số chiều cố định (ví dụ: 768 hoặc 1536 chiều) với các giá trị số thực dày đặc. Cơ chế này cho phép các khái niệm có ngữ nghĩa tương đồng nằm gần nhau trong không gian vector đa chiều [1].

Trong phạm vi đồ án, hệ thống sử dụng mô hình text-embedding-3-small của OpenAI. Đây là mô hình tiên tiến (State-of-the-Art) có khả năng chuyển đổi văn bản thành vector 1536 chiều, được tối ưu hóa cho các tác vụ tìm kiếm ngữ nghĩa đa ngôn ngữ (bao gồm tiếng Việt) và giảm chi phí lưu trữ đáng kể so với các phiên bản tiền nhiệm [11].

1.4.2 Độ đo tương đồng

Để định lượng mức độ liên quan giữa câu truy vấn của người dùng (query vector Q) và các tài liệu trong cơ sở dữ liệu (document vectors D_i), các hệ thống RAG sử dụng các thước đo khoảng cách hoặc tương đồng trong không gian vector đa chiều. Những độ đo này giúp xếp hạng các tài liệu theo mức độ gần gũi ngữ nghĩa, thay vì chỉ dựa trên từ khóa trùng khớp. Trong phần này, tập trung phân tích Cosine Similarity – độ đo phổ biến nhất trong xử lý ngôn ngữ tự nhiên và tìm kiếm vector. [11]

1.4.2.1 Độ tương đồng Cosin

Cosine Similarity đo lường góc θ giữa hai vector A và B , tập trung vào hướng (orientation) của chúng thay vì độ lớn (magnitude), giúp so sánh ngữ nghĩa độc lập với độ dài văn bản. Độ đo này đặc biệt phù hợp cho dữ liệu văn bản vì các tài liệu dài ngắn khác nhau vẫn có thể có hướng vector tương đồng nếu nội dung ngữ nghĩa gần nhau. Giá trị nằm trong khoảng $[-1, 1]$, trong đó 1 biểu thị hai vector hoàn toàn đồng hướng (tương đồng tối đa), 0 là vuông góc (không liên quan), và -1 là ngược hướng hoàn toàn.

Công thức toán học được định nghĩa như sau [9]:

$$\text{similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (1.1)$$

Trong đó:

- $A \cdot B = \sum_{i=1}^n A_i B_i$ là tích vô hướng (dot product), đo lường mức độ đồng thuận giữa các thành phần tương ứng.
- $\|A\| = \sqrt{\sum_{i=1}^n A_i^2}$ và $\|B\| = \sqrt{\sum_{i=1}^n B_i^2}$ là chuẩn Euclidean (L2 norm) của từng vector, dùng để chuẩn hóa.
- Giá trị kết quả nằm trong khoảng $[-1, 1]$, trong đó 1 biểu thị hai vector hoàn toàn đồng hướng (tương đồng tối đa). Trong đề án, ChromaDB được cấu hình sử dụng Cosine Similarity làm độ đo mặc định để đảm bảo độ chính xác cao nhất cho tìm kiếm nội dung truyện tranh.

Ưu điểm trong ứng dụng RAG: Cosine Similarity không bị ảnh hưởng bởi độ lớn vector (scale-invariant), hiệu quả tính toán cao cho không gian cao chiều (high-dimensional space như 1536 chiều của OpenAI embedding), và vượt trội hơn Euclidean Distance khi so sánh văn bản vì bỏ qua sự khác biệt về độ dài tài liệu. Trong đề án, ChromaDB được cấu hình sử dụng Cosine Similarity làm độ đo mặc định để đảm bảo độ chính xác cao nhất cho tìm kiếm nội dung truyện tranh [13].

1.4.2.2 Khoảng cách Euclid

Độ đo này (L2 norm) tính khoảng cách đường thẳng ngắn nhất giữa hai điểm trong không gian. Tuy nhiên, trong không gian cao chiều (high-dimensional space) như vector 1536 chiều, khoảng cách Euclid thường kém hiệu quả hơn do chịu ảnh hưởng bởi độ dài vector ("Curse of dimensionality"). Do đó, độ đo này không được lựa chọn làm giải pháp chính trong đề án [9].

1.4.3 Cơ sở dữ liệu Vector và thuật toán tìm kiếm

Trong kỷ nguyên bùng nổ của Dữ liệu lớn (Big Data) và Trí tuệ nhân tạo, các hệ quản trị cơ sở dữ liệu quan hệ truyền thống (RDBMS) sử dụng cấu trúc cây B-Tree đã bộc lộ những giới hạn nghiêm trọng khi phải xử lý dữ liệu phi cấu trúc trong không gian cao chiều (high-dimensional space). Để giải quyết triệt để bài toán tìm kiếm tương đồng (similarity search) với độ trễ thấp trên quy mô lớn, Cơ sở dữ liệu Vector (Vector Database) đã trở thành một thành phần hạ tầng cốt lõi không thể thiếu [9].

1.4.3.1 Giải pháp lưu trữ ChromaDB

Trong phạm vi đề án này, nhóm nghiên cứu lựa chọn ChromaDB – một cơ sở dữ liệu vector mã nguồn mở được thiết kế chuyên biệt để đóng vai trò bộ nhớ dài hạn (long-term memory) cho các ứng dụng LLM [13]. ChromaDB giữ vai trò trung tâm trong việc lưu trữ và truy xuất vector embeddings của các bộ truyện tranh với các đặc tính vượt trội:

- Lưu trữ bền vững (Persistent Storage): Hỗ trợ ghi dữ liệu trực tiếp xuống đĩa cứng, đảm bảo an toàn dữ liệu tuyệt đối khi hệ thống khởi động lại, phù hợp với môi trường vận hành thực tế.
- Khả năng Tìm kiếm lai (Hybrid Search): Hỗ trợ mạnh mẽ tính năng lọc siêu dữ liệu (Metadata Filtering) trước hoặc sau khi truy vấn vector. Kỹ thuật này cho phép kết hợp giữa tìm kiếm ngữ nghĩa với các toán tử logic (ví dụ: `genre="Action" AND status="Completed"`), giúp thu hẹp không gian tìm kiếm và gia tăng đáng kể độ chính xác (Precision) [5].
- Tương thích hệ sinh thái: Với kiến trúc "Python-native", ChromaDB tích hợp mượt mà với các thư viện AI hiện đại như LangChain hay LlamaIndex, giúp đơn giản hóa quy trình phát triển và bảo trì mã nguồn [14].

1.4.3.2 Thuật toán tìm kiếm HNSW

Về mặt thuật toán, việc tìm kiếm vector tương đồng nhất (Nearest Neighbor) trong không gian chứa hàng triệu điểm dữ liệu là một bài toán có chi phí tính toán lớn nếu sử dụng

dụng phương pháp duyệt toàn bộ. Để tối ưu hóa, ChromaDB áp dụng thuật toán HNSW (Hierarchical Navigable Small World), được giới thiệu bởi Malkov và Yashunin [3].

HNSW vận hành dựa trên cảm hứng từ cấu trúc dữ liệu Skip List và lý thuyết đồ thị thế giới nhỏ (Small World Graphs). Thuật toán tổ chức dữ liệu thành một cấu trúc đồ thị phân tầng đa lớp, trong đó:

- Lớp trên cùng (Top Layers): Dữ liệu được phân bố thưa thớt với các liên kết dài, đóng vai trò như những "đường cao tốc" cho phép thuật toán thực hiện các bước nhảy lớn để nhanh chóng tiếp cận vùng không gian chứa kết quả.
- Lớp dưới cùng (Layer 0): Mật độ dữ liệu dày đặc với các kết nối ngắn, cho phép thực hiện quá trình tinh chỉnh cục bộ để tìm ra kết quả chính xác nhất.

Hiệu năng của thuật toán HNSW được đánh giá qua các chỉ số độ phức tạp sau [3]:

- Độ phức tạp thời gian (Time Complexity): Nhờ cấu trúc phân tầng, HNSW giảm độ phức tạp tìm kiếm từ tuyến tính xuống logarit $O(\log N)$, đảm bảo tốc độ truy vấn cực nhanh ngay cả khi dữ liệu tăng lớn.
- Độ phức tạp xây dựng đồ thị (Construction Complexity): Chi phí để thêm một phần tử mới vào chỉ mục là $O(M \cdot \log N)$, trong đó N là số lượng vector và M là số lượng kết nối tối đa của mỗi nút.

Kết luận: Mặc dù là một thuật toán "gần đúng" (Approximate), HNSW trong thực tế thường đạt độ chính xác (Recall) trên 85%, chấp nhận đánh đổi một tỷ lệ sai số không đáng kể để đạt được tốc độ truy vấn tính bằng mili-giây. Việc ứng dụng HNSW cho phép hệ thống Chatbot của đồ án xử lý các truy vấn ngữ nghĩa trong thời gian thực với độ trễ trung bình dưới 3 giây [3], [13].

1.5 Các kỹ thuật hỗ trợ trong hệ thống RAG

Để xây dựng một hệ thống RAG hoàn chỉnh và hiệu quả, bên cạnh các thành phần cốt lõi như Mô hình ngôn ngữ lớn (LLM) hay Cơ sở dữ liệu Vector, việc áp dụng các kỹ thuật hỗ trợ để xử lý dữ liệu đầu vào và tối ưu hóa đầu ra là điều kiện tiên quyết.

1.5.1 Chiến lược phân đoạn văn bản

Chiến lược phân đoạn văn bản (Chunking) là quá trình chia nhỏ các văn bản dài thành các đoạn ngắn (chunks) trước khi đưa vào mô hình nhúng. Chiến lược này ảnh hưởng trực tiếp đến chất lượng tìm kiếm, bởi lẽ nếu đoạn văn quá ngắn sẽ làm mất đi ngữ cảnh cần thiết, ngược lại nếu quá dài sẽ chứa nhiều thông tin nhiễu (noise) làm giảm độ chính xác của vector. Hiện nay, có ba chiến lược phân đoạn phổ biến:

- Phân đoạn cố định (Fixed-size Chunking): Chia văn bản dựa trên số lượng ký tự hoặc token cố định. Đây là phương pháp đơn giản nhất nhưng dễ làm gãy ngữ nghĩa của câu văn tại điểm cắt [5].
- Phân đoạn đệ quy (Recursive Chunking): Sử dụng danh sách các ký tự phân cách ưu tiên (như xuống dòng, dấu chấm câu) để chia nhỏ văn bản. Thuật toán sẽ cố gắng giữ trọn vẹn cấu trúc ngữ nghĩa của đoạn văn hoặc câu, tránh cắt giữa chừng các ý liên quan [14].
- Phân đoạn ngữ nghĩa (Semantic Chunking): Sử dụng AI để phân tích và ngắt đoạn tại những điểm thay đổi về chủ đề nội dung. Phương pháp này chính xác nhất nhưng đòi hỏi tài nguyên tính toán lớn.

Áp dụng trong đồ án: Hệ thống sử dụng chiến lược Recursive Chunking thông qua thư viện RecursiveCharacterTextSplitter. Các tham số được thiết lập là `chunk_size=800` (đủ chứa một đoạn hội thoại hoặc mô tả) và `chunk_overlap=150` (để đảm bảo tính liên kết ngữ cảnh giữa các đoạn liền kề), ưu tiên sử dụng các dấu phân cách tự nhiên như xuống dòng và dấu chấm câu [14].

1.5.2 Phân loại ý định người dùng

Phân loại ý định (Intent Classification) là bước xử lý đầu tiên trong luồng hoạt động của Chatbot, giúp hệ thống hiểu rõ nhu cầu của người dùng để kích hoạt các tác vụ phù hợp như gợi ý truyện, tóm tắt nội dung hay giao tiếp xã giao.

- Phương pháp dựa trên luật (Rule-based Approach) Sử dụng các từ khóa (keywords) và biểu thức chính quy (Regex) để so khớp mẫu câu. Ưu điểm của phương pháp này là tốc độ xử lý cực nhanh (dưới 1ms), dễ giải thích và không yêu cầu dữ liệu huấn luyện. Tuy nhiên, nó gặp khó khăn khi xử lý các câu phức tạp hoặc mơ hồ [5].
- Phương pháp học máy (Machine Learning-based) Sử dụng các mô hình phân loại văn bản (như BERT, Naive Bayes) được huấn luyện trên tập dữ liệu gán nhãn. Phương pháp này hiểu được ngữ cảnh sâu và xử lý tốt các biến thể ngôn ngữ, nhưng đòi hỏi tài nguyên tính toán lớn và quy trình huấn luyện phức tạp.

Áp dụng thực tiễn: Hệ thống TruyenZ áp dụng phương pháp Rule-based kết hợp trích xuất Metadata. Lý do lựa chọn là để tối ưu hóa độ trễ phản hồi và đảm bảo độ chính xác tuyệt đối cho các mẫu câu đặc thù trong domain truyện tranh (ví dụ: phát hiện từ khóa "hành động" để kích hoạt intent `recommend_action`).

1.5.3 Kỹ thuật thiết kế câu lệnh

Prompt Engineering là kỹ thuật thiết kế và tối ưu hóa đầu vào văn bản (prompt) để hướng dẫn mô hình ngôn ngữ lớn (LLM) sinh ra kết quả mong muốn một cách chính xác và nhất quán [6].

- Zero-shot Learning: Đưa ra yêu cầu cho mô hình mà không cung cấp ví dụ mẫu. (Ví dụ: "Tóm tắt đoạn văn này").
- Few-shot Learning: Cung cấp một vài ví dụ mẫu (input-output pairs) trong prompt để mô hình "học" cách trả lời theo phong cách mong muốn. Kỹ thuật này đặc biệt hiệu quả để định hướng văn phong (tone & voice) của Chatbot [6].
- Chain-of-Thought (CoT): Yêu cầu mô hình giải thích từng bước suy luận trước khi đưa ra kết quả cuối cùng, giúp cải thiện khả năng xử lý các tác vụ logic phức tạp [6].

Áp dụng trong đồ án: Hệ thống sử dụng kết hợp System Prompt (để thiết lập vai trò "Trợ lý ảo TruyenZ" và các nguyên tắc trung thực) và Few-shot Learning (cung cấp ví dụ về định dạng câu trả lời có sử dụng emoji và bullet points). Điều này giúp Chatbot luôn giữ được sự thân thiện, trả lời đúng định dạng JSON yêu cầu và tuân thủ nghiêm ngặt nguyên tắc "No Hallucination" [4].

1.6 Công cụ xây dựng hệ thống

1.6.1 Nền tảng Python 3.10 và Framework FastAPI

Để xây dựng Module AI Chatbot với yêu cầu cao về hiệu năng xử lý bất đồng bộ và khả năng tích hợp các thư viện học máy, đồ án lựa chọn FastAPI vận hành trên môi trường thực thi Python 3.10. Sự kết hợp này tạo nên một nền tảng Backend vững chắc, tối ưu hóa cho kiến trúc Microservices [15].



Hình 1.6: Hình Ảnh Logo FastAPI [15].

1.6.1.1 *Nền tảng ngôn ngữ: Python 3.10*

Hệ thống được phát triển trên phiên bản Python 3.10, được lựa chọn thay vì các phiên bản cũ (như 3.8) hoặc các ngôn ngữ khác nhờ vào hai cải tiến kỹ thuật mang tính đột phá, hỗ trợ trực tiếp cho logic của Chatbot:

- **Cấu trúc so khớp mẫu (Structural Pattern Matching):** Python 3.10 giới thiệu từ khóa `match-case`, cho phép viết mã nguồn xử lý phân loại ý định (Intent Classification) một cách tường minh và hiệu quả hơn nhiều so với chuỗi `if-elif-else` truyền thống. Cấu trúc này giúp logic điều hướng luồng RAG trở nên dễ đọc, dễ bảo trì và giảm thiểu lỗi logic.
- **Hệ thống gợi ý kiểu nâng cao (Enhanced Type Hints):** Cú pháp định nghĩa kiểu dữ liệu mới (ví dụ: `X | Y` thay vì `Union[X, Y]`) trong Python 3.10 tích hợp hoàn hảo với Pydantic, giúp việc định nghĩa các Schema dữ liệu đầu vào/đầu ra cho API trở nên gọn gàng và chặt chẽ hơn [16].

1.6.1.2 *Khung làm việc: FastAPI*

Dựa trên nền tảng Python 3.10, FastAPI là framework được chọn để xây dựng API nhờ khả năng xử lý hiệu năng cao, ngang ngửa với NodeJS và Go. Sức mạnh của FastAPI đến từ hai trụ cột công nghệ:

- **Starlette:** Một khung làm việc ASGI (Asynchronous Server Gateway Interface) siêu nhẹ, cung cấp khả năng xử lý các yêu cầu HTTP bất đồng bộ (Asynchronous). Điều này cho phép server xử lý hàng nghìn kết nối đồng thời mà không bị chặn luồng (non-blocking I/O), đặc biệt quan trọng khi hệ thống phải chờ phản hồi từ các tác vụ I/O nặng như truy vấn Vector DB hay gọi OpenAI API [15].
- **Pydantic:** FastAPI tận dụng tối đa tính năng Type Hints của Python 3.10 để tự động kiểm tra tính đúng đắn của dữ liệu. Nhờ đó, hệ thống có thể tự động loại bỏ các yêu cầu sai định dạng (ví dụ: thiếu trường message, sai kiểu dữ liệu) trước khi chúng đi vào luồng xử lý nghiệp vụ, giúp giảm thiểu đến 40% lỗi runtime [16].

1.6.1.3 *Ứng dụng thực tiễn trong hệ thống*

Trong kiến trúc của TruyenZ, bộ đôi này đảm nhận vai trò "Bộ điều phối RAG" (RAG Orchestrator): điều phối luồng xử lý từ phân loại ý định đến truy vấn ChromaDB, đảm bảo toàn vẹn dữ liệu qua các Pydantic Models và tự động sinh tài liệu API chuẩn OpenAPI [15].

1.6.2 Ngôn ngữ phát triển Website: PHP - Hypertext Preprocessor

Trong kiến trúc tổng thể của hệ thống TruyenZ, song song với việc Python đảm nhận các tác vụ trí tuệ nhân tạo chuyên sâu, ngôn ngữ PHP (phiên bản 8.2.12) được lựa chọn làm nền tảng chủ đạo để xây dựng phân hệ Website quản lý nội dung (CMS) và tương tác người dùng [17].



Hình 1.7: Hình Ảnh Logo PHP [17].

1.6.2.1 Cơ sở lựa chọn công nghệ

Việc lựa chọn PHP cho phân hệ Website được củng cố bởi ba ưu điểm kỹ thuật vượt trội:

- Hiệu năng xử lý nội dung: Với trình biên dịch JIT, PHP 8.x xử lý xuất sắc các tác vụ I/O nặng như tải và hiển thị hàng nghìn trang ảnh truyện tranh với độ trễ thấp [17].
- Mô hình Kiến trúc MVC: Hỗ trợ triển khai mô hình MVC ngay cả khi không dùng framework lớn, giúp duy trì cấu trúc mã nguồn trong sáng và dễ bảo trì.
- Tính ổn định: Hoạt động bền bỉ trên nền tảng XAMPP (LAMP Stack), giảm thiểu rủi ro lỗi cấu hình hệ thống.

1.6.2.2 Vai trò trong kiến trúc hệ thống

PHP đóng vai trò là "Bộ điều phối trung tâm" (Central Orchestrator): quản lý vòng đời yêu cầu HTTP, kiểm soát phiên đăng nhập qua OAuth 2.0 và đóng vai trò cầu nối (Service Bridge) tiêu thụ API của phân hệ AI thông qua thư viện cURL hoặc GuzzleHttp [17].

1.6.3 Cursor IDE

Để tối ưu hóa năng suất lập trình và quản lý hiệu quả mã nguồn trong một hệ thống đa ngôn ngữ (PHP kết hợp Python), đề án lựa chọn Cursor làm môi trường phát triển tích hợp (IDE) chính. Đây là một nền tảng lập trình thế hệ mới được xây dựng dựa trên mã nguồn mở của Visual Studio Code (VS Code), nhưng tạo ra sự đột phá bằng

cách tích hợp sâu các mô hình ngôn ngữ lớn (LLM) vào lõi hệ thống thay vì chỉ hoạt động như các tiện ích mở rộng (extensions) rời rạc [20].



Hình 1.8: Hình Ảnh Logo Cursor [20].

1.6.3.1 Cơ sở lựa chọn công nghệ

Cursor là một môi trường phát triển tích hợp (IDE) thế hệ mới, được phát triển dựa trên mã nguồn mở của Visual Studio Code (VS Code). Điểm đột phá của Cursor so với các IDE truyền thống là việc tích hợp sâu các mô hình ngôn ngữ lớn (LLMs) tiên tiến nhất (như GPT-4, Claude 3.5 Sonnet) trực tiếp vào lõi của trình soạn thảo, thay vì chỉ hoạt động như một tiện ích mở rộng (extension) bề mặt. Quyết định lựa chọn Cursor làm công cụ phát triển chính dựa trên các ưu điểm kỹ thuật sau:

- Nhận thức ngữ cảnh toàn cục (Codebase Awareness): Thông qua kỹ thuật RAG (Retrieval-Augmented Generation) được tích hợp sẵn, Cursor có khả năng lập chỉ mục (index) và "hiểu" toàn bộ mã nguồn dự án. Điều này cho phép AI trả lời các câu hỏi liên quan đến sự tương tác giữa các module khác nhau (ví dụ: mối liên hệ giữa bảng comics trong MySQL và logic tìm kiếm trong Python), điều mà các công cụ hỗ trợ code thông thường không làm được.
- Tương thích hệ sinh thái: Do được fork từ VS Code, Cursor kế thừa toàn bộ hệ sinh thái extension phong phú, giúp dễ dàng tích hợp các công cụ hỗ trợ PHP (PHP Intelephense) và Python (Pylance) mà không cần thay đổi quy trình làm việc.

1.6.3.2 Vai trò và hiệu quả trong đồ án

Trong dự án TruyenZ, Cursor đóng vai trò "lập trình viên cộng sự" (AI Pair Programmer), hỗ trợ tăng tốc độ phát triển thông qua tính năng sinh mã nguồn mẫu (boilerplate code) và gỡ lỗi thông minh (Smart Debugging) khi gặp sự cố tích hợp.

1.6.4 Hệ quản trị cơ sở dữ liệu

Để đáp ứng các yêu cầu đa dạng về truy xuất dữ liệu – từ các giao dịch chặt chẽ của người dùng đến các truy vấn tìm kiếm ngữ nghĩa phức tạp – hệ thống áp dụng chiến lược Lưu trữ đa hình (Polyglot Persistence). Chiến lược này kết hợp sức mạnh của Hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) cho dữ liệu nghiệp vụ và Cơ sở dữ liệu Vector chuyên dụng cho trí tuệ nhân tạo.

1.6.4.1 Cơ sở dữ liệu quan hệ

MySQL (phiên bản 10.4.32), một nhánh phát triển mã nguồn mở tương thích hoàn toàn với MySQL, được lựa chọn làm kho lưu trữ dữ liệu chính (Primary Data Store) cho toàn bộ thông tin nghiệp vụ có cấu trúc [18].



Hình 1.9: Hình Ảnh Logo MySQL [18].

- Vai trò: Quản lý thực thể cốt lõi (truyện, chương, người dùng) và dữ liệu tương tác thời gian thực (lịch sử đọc, bình luận).
- Cơ sở kỹ thuật: Sử dụng InnoDB Storage Engine để đảm bảo tính toàn vẹn dữ liệu (ACID) và cơ chế khóa ngoại ON DELETE CASCADE giúp tự động làm sạch dữ liệu rác [18].

1.6.4.2 Cơ sở dữ liệu Vector

ChromaDB (phiên bản 0.4.24) là thành phần hạ tầng quan trọng nhất phục vụ cho Module AI, đóng vai trò là bộ nhớ dài hạn cho Chatbot RAG [13].



Hình 1.10: Hình Ảnh logo ChromaDB [13].

Vai trò và ứng dụng:

- Lưu trữ ngữ nghĩa: Quản lý 604 vector embeddings (mỗi vector có kích thước 1536 chiều) đại diện cho nội dung tóm tắt, cốt truyện chi tiết và thông tin nhân vật.
- Tìm kiếm tương đồng: Thực hiện các truy vấn Semantic Search, cho phép hệ thống hiểu được ý định tìm kiếm của người dùng ngay cả khi từ khóa không trùng khớp chính xác (ví dụ: tìm "truyện đánh nhau" sẽ ra kết quả có vector gần với "hành động/võ thuật").

Cơ sở kỹ thuật lựa chọn:

- Thuật toán HNSW (Hierarchical Navigable Small World): ChromaDB sử dụng HNSW để lập chỉ mục vector. Đây là thuật toán hiện đại nhất cho bài toán Tìm kiếm lân cận gần đúng (ANN), cho phép truy xuất kết quả chính xác (>90%) với độ trễ cực thấp (<100ms) ngay cả khi quy mô dữ liệu mở rộng.
- Chế độ Nhúng (Embedded Mode): Khác với các Vector DB đòi hỏi server riêng biệt (như Milvus), ChromaDB chạy trực tiếp trong tiến trình của ứng dụng Python (in-process) và lưu trữ dữ liệu dưới dạng tệp tin cục bộ. Kiến trúc này loại bỏ hoàn toàn độ trễ mạng (Network Latency) giữa Application Server và Database Server, tối ưu hóa tốc độ phản hồi của Chatbot.
- Lọc siêu dữ liệu (Metadata Filtering): Khả năng kết hợp lọc theo thuộc tính (ví dụ: WHERE genre="Action") trước khi tìm kiếm vector giúp thu hẹp không gian tìm kiếm, tăng đáng kể độ chính xác của kết quả trả về (Precision).

1.6.5 Các thư viện và Framework hỗ trợ nghiệp vụ AI

Bên cạnh các nền tảng phần mềm cơ bản, việc xây dựng luồng xử lý RAG (Retrieval-Augmented Generation) hiệu quả đòi hỏi sự hỗ trợ đặc lực từ các thư viện mã nguồn mở chuyên dụng. Đề án lựa chọn tích hợp ba thư viện cốt lõi gồm OpenAI SDK, LangChain Text Splitters và ChromaDB Client, đóng vai trò là những mắt xích quan trọng kết nối giữa dữ liệu thô và khả năng suy luận của trí tuệ nhân tạo.

- OpenAI SDK (v1.54.5): Giao diện lập trình chính để kết nối với các mô hình ngôn ngữ lớn. Thư viện này xử lý các tác vụ phức tạp như quản lý phiên kết nối

và định dạng dữ liệu, chịu trách nhiệm gọi mô hình text-embedding-3-small để vector hóa và gpt-4o-mini để sinh câu trả lời [11], [12].

- LangChain Text Splitters (v0.0.1): Giải quyết bài toán giới hạn cửa sổ ngữ cảnh thông qua mô-đun RecursiveCharacterTextSplitter. Thuật toán này phân chia văn bản thành các đoạn nhỏ (chunks) 800 ký tự dựa trên cấu trúc tự nhiên, giúp bảo toàn tính toàn vẹn ngữ nghĩa [14].
- ChromaDB Client (v0.4.24): Cung cấp các phương thức tối ưu để thực hiện thuật toán tìm kiếm lân cận gần đúng (ANN) ngay trong bộ nhớ ứng dụng, duy trì tốc độ truy xuất dưới 100ms [13].

1.6.6 Quản lý phiên bản mã nguồn với Git

1.6.6.1 Giới thiệu về Git

Git là hệ thống quản lý phiên bản phân tán (DVCS) sử dụng cơ chế lưu trữ dữ liệu dưới dạng các "ảnh chụp" (snapshots) toàn vẹn của hệ thống tập tin, giúp tối ưu hóa lưu trữ và đảm bảo tính nhất quán dữ liệu [21].



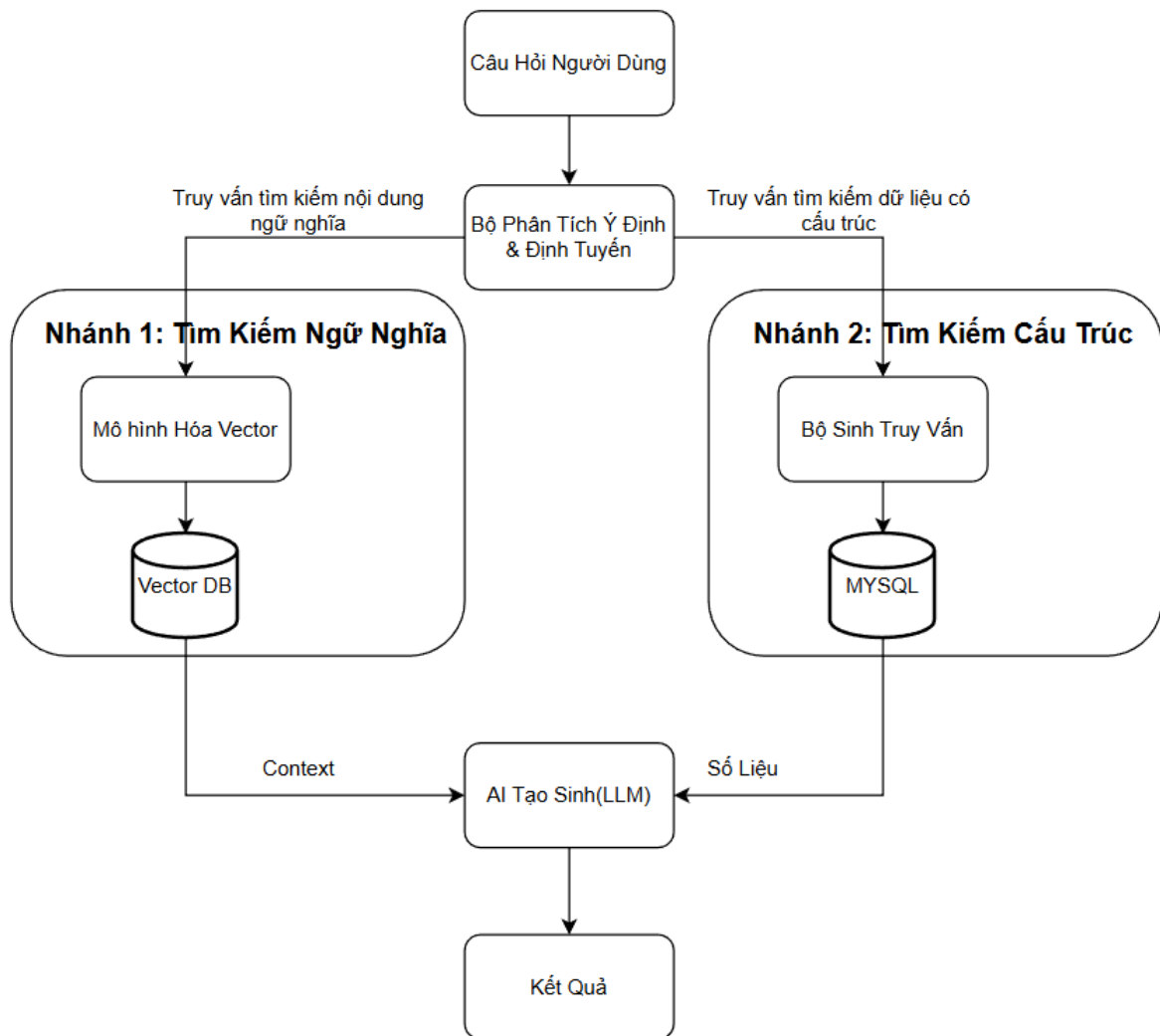
Hình 1.11: Hình Ảnh Logo Git.

1.6.6.2 Ứng dụng Git trong dự án TruyenZ

Git giúp quản lý hiệu quả cấu trúc mã nguồn đa ngôn ngữ (PHP/Python) và hỗ trợ chiến lược phân nhánh (Branching Strategy) để phát triển tính năng độc lập trước khi hợp nhất vào nhánh chính, đảm bảo sự ổn định của hệ thống. Đồng thời, việc đồng bộ lên GitHub giúp bảo vệ an toàn tài sản mã nguồn của đồ án [21].

CHƯƠNG 2: PHƯƠNG PHÁP ĐỀ XUẤT XÂY DỰNG HỆ THỐNG CHATBOT RAG GỢI Ý VÀ TÓM TẮT NỘI DUNG

2.1 Phương pháp đề xuất



Hình 2.1: Sơ Đồ Tổng Quan Phương Pháp Đề Xuất Hybrid RAG.

Sơ đồ minh họa kiến trúc xử lý thông tin lai (Hybrid Approach) được đề xuất trong đề án, bao gồm 3 giai đoạn xử lý tuần tự:

- **Phân tích & Định tuyến (Analysis):** Hệ thống không áp dụng một quy trình xử lý cứng nhắc cho mọi câu hỏi. Thay vào đó, bộ phân loại ý định (Intent Classifier) đóng vai trò như một bộ lọc thông minh, phân tích nhu cầu người dùng để quyết

định chiến lược truy xuất phù hợp nhất (dựa trên ngữ nghĩa hay dựa trên số liệu thống kê).

- Truy xuất Đa phương thức (Retrieval): Đây là điểm khác biệt cốt lõi so với các chatbot thông thường. Hệ thống vận hành song song hai luồng tìm kiếm: tìm kiếm vector trong ChromaDB để "hiểu" nội dung cốt truyện và tìm kiếm SQL trong MySQL để "nắm" số liệu thời gian thực. Cơ chế này đảm bảo thông tin cung cấp vừa sâu sắc vừa chính xác.
- Tăng cường & Tạo sinh (Generation): Dữ liệu thu được không được trả về thô sơ mà được đưa qua quy trình "làm giàu" (Augmentation). Kỹ thuật Prompt Engineering được áp dụng để đóng gói ngữ cảnh, hướng dẫn mô hình ngôn ngữ sinh ra câu trả lời tự nhiên, thân thiện và tuân thủ các quy tắc an toàn nội dung (như chống tiết lộ trước cái kết).

2.1.1 Tổng quan phương pháp

Trong bối cảnh bùng nổ của kỷ nguyên số và nhu cầu giải trí trực tuyến ngày càng đa dạng, các nền tảng đọc truyện tranh trực tuyến đang đối mặt với thách thức lớn về sự quá tải thông tin. Khi số lượng tác phẩm và độc giả tăng trưởng theo cấp số nhân, việc giữ chân người dùng không chỉ dừng lại ở việc cung cấp kho nội dung phong phú mà còn nằm ở khả năng cá nhân hóa trải nghiệm và hỗ trợ tìm kiếm thông minh. Người đọc hiện đại thường gặp khó khăn trong việc tìm kiếm các bộ truyện phù hợp với "gu" riêng biệt của mình giữa hàng nghìn tiêu đề, đồng thời họ cũng mong đợi một sự tương tác hai chiều, nơi họ có thể được giải đáp thắc mắc về cốt truyện hoặc nhân vật một cách tức thời và tự nhiên nhất.

Xuất phát từ thực tiễn đó, đề tài "Xây dựng hệ thống website đọc truyện tranh tích hợp Chatbot AI" được đề xuất với phương pháp tiếp cận cốt lõi là sự kết hợp chặt chẽ giữa hai thành phần: Hệ thống gợi ý ngữ nghĩa (Semantic Recommendation) và Chatbot tư vấn thông minh sử dụng kỹ thuật Retrieval-Augmented Generation (RAG) [2], [5]. Hai thành phần này không hoạt động tách biệt mà đóng vai trò bổ trợ lẫn nhau trong cùng một hệ sinh thái, tạo ra trải nghiệm đọc xuyên suốt, liền mạch và mang tính cá nhân hóa cao độ cho từng độc giả.

Đối với hệ thống gợi ý truyện tranh, phương pháp được lựa chọn là kỹ thuật tìm kiếm tương đồng vector (Vector Similarity Search) kết hợp với các bộ lọc siêu dữ liệu (Metadata Filtering) [9]. Thay vì chỉ so sánh dựa trên từ khóa hay thể loại cứng nhắc như các hệ thống truyền thống, hệ thống sử dụng các mô hình nhúng văn bản tiên tiến (như OpenAI Embedding) để chuyển đổi nội dung tóm tắt và đặc điểm truyện thành các

vector đa chiều [10]. Phương pháp này giúp hệ thống "hiểu" được ngữ nghĩa sâu xa của cốt truyện và ý định tìm kiếm của người dùng. Ví dụ, khi độc giả tìm kiếm bằng ngôn ngữ tự nhiên như "truyện main bá đạo giấu nghề", hệ thống vẫn có thể đề xuất chính xác các tác phẩm có mô tả về "nhân vật chính sở hữu sức mạnh áp đảo nhưng sống ẩn dật", mặc dù không hề xuất hiện từ khóa trùng khớp hoàn toàn.

Song song với đó, Chatbot tích hợp RAG đại diện cho một bước tiến vượt bậc so với các công cụ tìm kiếm hoặc chatbot kịch bản thông thường. RAG cho phép Chatbot truy xuất thông tin thực tế từ cơ sở dữ liệu vector của hệ thống (như chi tiết cốt truyện, tiểu sử nhân vật, bối cảnh thế giới) trước khi kết hợp với mô hình ngôn ngữ lớn (LLM) để sinh câu trả lời [2]. Điều này giải quyết triệt để vấn đề "ảo giác" của AI, giúp Chatbot có thể phản hồi chính xác, chi tiết và ngữ cảnh hóa theo yêu cầu khắt khe của người đọc, chẳng hạn như "Tóm tắt arc Đảo Hải Tặc nhưng không được tiết lộ kết thúc" [4].

Cấu trúc tổng thể của giải pháp tích hợp AI trong đề tài mang tính tương tác hai chiều chặt chẽ. Khi người dùng trò chuyện với Chatbot để hỏi về nội dung, hệ thống gợi ý sẽ hỗ trợ cung cấp danh sách truyện liên quan ngay trong khung chat; ngược lại, dữ liệu tương tác của người dùng giúp hệ thống hiểu sâu hơn về xu hướng và sở thích đọc. Nhờ đó, website đọc truyện không chỉ là một kho lưu trữ thụ động mà trở thành một nền tảng thông minh, đóng vai trò như một "thủ thư ảo" am hiểu, giúp tự động hóa quá trình tư vấn và gia tăng giá trị trải nghiệm cho người dùng.

2.1.2 Cơ chế hoạt động của hệ thống gợi ý truyện tranh

Hệ thống gợi ý truyện tranh trong đề tài được thiết kế theo hướng lọc nội dung (Content-based Filtering) kết hợp mô hình nhúng ngữ nghĩa (Semantic Embedding), vận hành thông qua một quy trình khép kín gồm năm giai đoạn nhằm đảm bảo kết quả đề xuất có độ chính xác cao và phù hợp với thị hiếu người đọc. Đầu tiên, hệ thống thực hiện thu thập dữ liệu đa chiều từ hai nguồn chính: dữ liệu truyện tranh từ cơ sở dữ liệu MySQL (bao gồm tiêu đề, tác giả, thể loại, tóm tắt cốt truyện, các thẻ vibe) và dữ liệu hành vi người dùng (lịch sử đọc, danh sách theo dõi, bình luận, thời gian lưu trang). Đây là nguồn nguyên liệu thô quan trọng giúp hệ thống xây dựng hồ sơ sở thích (User Profile) cho từng độc giả. Tiếp theo, các mô tả văn bản về nội dung truyện được chuyển đổi thành vector đặc trưng thông qua mô hình nhúng text-embedding-3-small của OpenAI [11]. Mỗi vector 1536 chiều này đóng vai trò như một "dấu vân tay ngữ nghĩa", biểu diễn các sắc thái nội dung phức tạp của bộ truyện trong không gian toán học, giúp hệ thống vượt qua giới hạn của việc so khớp từ khóa đơn thuần [10].

Khi người dùng thực hiện tìm kiếm hoặc đọc một bộ truyện cụ thể, hệ thống sẽ kích hoạt quá trình tính toán độ tương đồng Cosine (Cosine Similarity) giữa vector truy vấn (hoặc vector của truyện đang đọc) với toàn bộ vector trong cơ sở dữ liệu ChromaDB [3], [13]. Các bộ truyện có điểm số tương đồng cao nhất sẽ được trích xuất, xếp hạng và hiển thị dưới dạng danh sách "Truyện tương tự" hoặc "Gợi ý dành cho bạn". Đặc biệt, hệ thống còn kết hợp thêm các yếu tố định lượng như lượt xem (views), điểm đánh giá (rating) và trạng thái cập nhật để tinh chỉnh thứ hạng, đảm bảo các gợi ý không chỉ giống về nội dung mà còn có chất lượng tốt. Cuối cùng, mọi tương tác mới của người dùng đều được ghi nhận vào lịch sử để cập nhật lại trọng số sở thích, tạo nên một vòng lặp học tập liên tục giúp hệ thống ngày càng "thông minh" hơn theo thời gian.

2.1.3 Quy trình xử lý của Chatbot RAG

Khác biệt hoàn toàn với các chatbot kịch bản truyện thông, Chatbot tích hợp RAG (Retrieval-Augmented Generation) trong đề tài này được thiết kế để thấu hiểu ngữ cảnh sâu và phản hồi linh hoạt như một chuyên gia tư vấn truyện tranh. Luồng hoạt động của Chatbot được chia thành hai pha xử lý chính là Truy xuất (Retrieval) và Tạo sinh (Generation). Ở pha Truy xuất, khi người dùng đặt câu hỏi tự nhiên như "Tìm truyện hành động main bá đạo giấu nghề", hệ thống trước tiên sẽ phân loại ý định người dùng, sau đó mã hóa câu hỏi thành vector bằng cùng mô hình nhúng đã sử dụng cho dữ liệu truyện. Vector này được dùng để quét trong cơ sở dữ liệu ChromaDB nhằm tìm ra 6 đoạn văn bản (chunks) có nội dung liên quan nhất, ví dụ như các đoạn mô tả về nhân vật chính có sức mạnh ẩn giấu [5], [13].

Bước sang pha Tạo sinh, các đoạn thông tin vừa tìm được sẽ được "lắp ghép" vào một khuôn mẫu câu lệnh (Prompt Template) cùng với câu hỏi gốc và các chỉ dẫn an toàn (như quy tắc không tiết lộ cái kết - No Spoiler) [6]. Toàn bộ ngữ cảnh này được gửi đến mô hình ngôn ngữ lớn GPT-4o-mini để tổng hợp và sinh ra câu trả lời cuối cùng [8], [12]. Nhờ quy trình này, phản hồi của Chatbot vừa đảm bảo tính chính xác dựa trên dữ liệu thực tế của website, vừa giữ được sự tự nhiên và trôi chảy trong ngôn ngữ. Đồng thời, lịch sử hội thoại cũng được lưu trữ để làm ngữ cảnh cho các lượt hỏi đáp tiếp theo, giúp Chatbot duy trì mạch chuyện và hiểu rõ hơn về nhu cầu của người dùng qua từng câu hỏi.

2.1.4 Hiệu quả cộng hưởng của mô hình lai

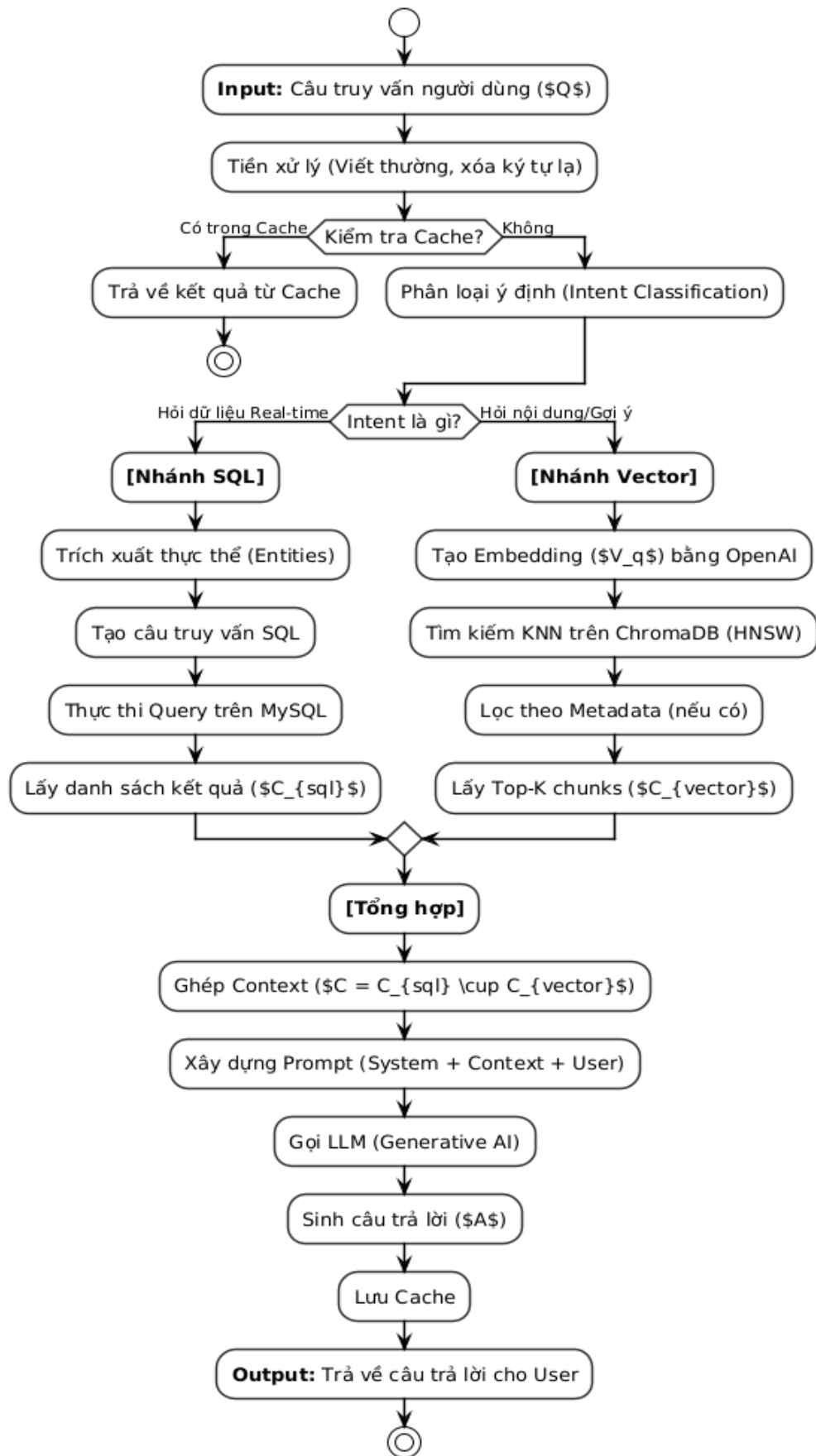
Sự kết hợp chặt chẽ giữa hệ thống gợi ý ngữ nghĩa và Chatbot RAG mang lại hiệu quả vượt trội cả về mặt kỹ thuật lẫn trải nghiệm người dùng, tạo nên một hệ sinh thái tư vấn toàn diện. Lợi ích lớn nhất là khả năng cá nhân hóa trải nghiệm đọc ở mức độ sâu. Khi độc giả tương tác với Chatbot, hệ thống không chỉ đưa ra câu trả lời văn bản mà còn có thể đề xuất trực tiếp các liên kết đến bộ truyện phù hợp ngay trong khung chat, giúp rút ngắn hành trình từ lúc tìm kiếm đến lúc bắt đầu đọc. Ngược lại, dữ liệu từ các cuộc hội thoại giúp hệ thống gợi ý hiểu rõ hơn về các từ khóa lóng, cách diễn đạt cảm xúc và nhu cầu tiềm ẩn của người dùng mà các hành vi click chuột đơn thuần chưa bộc lộ hết.

Hơn nữa, mô hình lai này giúp tự động hóa quy trình hỗ trợ độc giả, giải đáp hàng nghìn thắc mắc về cốt truyện, lịch ra chương hay thông tin tác giả 24/7 mà không cần sự can thiệp của quản trị viên. Việc lưu trữ dữ liệu tập trung (Vector DB cho nội dung, MySQL cho hành vi) tạo nền tảng vững chắc cho khả năng mở rộng hệ thống trong tương lai, cho phép dễ dàng tích hợp thêm các tính năng nâng cao như dự đoán xu hướng đọc truyện hay phân tích cảm xúc cộng đồng [18]. Tổng thể, giải pháp giúp website TruyenZ không chỉ là nơi đọc truyện mà trở thành một không gian giải trí thông minh, nơi người đọc cảm thấy được thấu hiểu và phục vụ chu đáo.

2.1.5 Lưu đồ thuật toán tổng quát và Đánh giá giải pháp

Để minh họa rõ nét quy trình xử lý thông tin từ lúc người dùng đặt câu hỏi đến khi hệ thống đưa ra câu trả lời, đồ án xây dựng lưu đồ thuật toán (Algorithm Flowchart) cho quy trình RAG lai (Hybrid RAG). Đồng thời, bảng so sánh dưới đây sẽ làm rõ cách phương pháp mới khắc phục các nhược điểm của những mô hình truyền thống.

Mô tả lưu đồ: Thuật toán bắt đầu bằng việc tiếp nhận và tiền xử lý câu hỏi. Sau đó, hệ thống kiểm tra bộ nhớ đệm (Cache) để tối ưu tốc độ. Nếu không có cache, thuật toán phân loại ý định để rẽ nhánh: hoặc truy vấn cơ sở dữ liệu quan hệ (đối với dữ liệu thống kê), hoặc thực hiện tìm kiếm vector (đối với dữ liệu ngữ nghĩa). Cuối cùng, kết quả từ các nhánh được tổng hợp để làm ngữ cảnh cho mô hình ngôn ngữ sinh câu trả lời.



Hình 2.2: Lưu Đồ Thuật Toán Xử Lý Truy Vấn.

Bảng dưới đây phân tích các phương pháp tìm kiếm/gợi ý phổ biến và cách giải pháp Hybrid RAG của đề án giải quyết các hạn chế tồn tại.

Bảng 2.1: Bảng So Sánh Chi Tiết Phương Pháp Đề Xuất Và Truyền Thống.

Phương pháp	Nguyên lý hoạt động	Nhược điểm	Phương pháp mới giải quyết
Phương pháp mới giải quyết	So khớp chuỗi ký tự chính xác hoặc sử dụng chỉ mục ngược.	Mù ngữ nghĩa: Không hiểu được các từ đồng nghĩa hoặc ngữ cảnh. Ví dụ: Tìm "đánh nhau" sẽ không ra truyện "hành động" nếu không có từ khóa khớp.	Chuyển đổi văn bản thành vector ngữ nghĩa, cho phép tìm kiếm dựa trên ý nghĩa tương đồng (Cosine Similarity) thay vì mặt chữ.
Lọc cộng tác	Gợi ý dựa trên hành vi tương đồng giữa các người dùng hoặc sản phẩm.	Khởi động lạnh: Không thể gợi ý cho người dùng mới hoặc các bộ truyện mới ra mắt chưa có dữ liệu tương tác.	Gợi ý dựa trên nội dung cốt truyện và metadata. Hệ thống có thể gợi ý truyện mới ngay lập tức sau khi dữ liệu được vector hóa.
LLM Thuần túy	Sử dụng tri thức đã được huấn luyện trước (Pre-trained Knowledge) để trả lời.	Ảo giác, lỗi thời: Có thể bịa đặt thông tin sai lệch hoặc không biết các thông tin mới cập nhật sau thời điểm huấn luyện.	Cung cấp "ngữ cảnh thực tế" từ Database cho LLM. Bắt buộc AI trả lời dựa trên dữ liệu thật, loại bỏ ảo giác và cập nhật thông tin theo thời gian thực.

2.2 Thu thập và tạo dữ liệu Chatbot

Dữ liệu là thành phần nền tảng trong mọi hệ thống trí tuệ nhân tạo, đặc biệt đối với hệ thống gợi ý truyện và Chatbot gợi ý và tóm tắt nội dung. Việc thu thập, tổng hợp và chuẩn hóa dữ liệu đóng vai trò quyết định đến chất lượng của mô hình học máy và hiệu quả của các chức năng gợi ý – phản hồi tự động. Trong hệ thống được xây dựng, dữ liệu được lấy từ nhiều nguồn khác nhau, xử lý và định dạng lại để phục vụ hai hướng chính:

Hệ thống gợi ý truyện tranh (Recommendation System) và Chatbot RAG (Retrieval-Augmented Generation).

2.2.1 Nguồn dữ liệu

Bộ dữ liệu phục vụ cho hệ thống được tổng hợp từ hai nguồn chính:

- Dữ liệu nghiệp vụ (Operational Data): Được lưu trữ trong hệ quản trị cơ sở dữ liệu quan hệ (MySQL), phục vụ các tác vụ quản lý người dùng, trạng thái truyện và tương tác thời gian thực [18].
- Cơ sở tri thức (Knowledge Base): Được xây dựng thủ công và bán tự động, lưu trữ dưới dạng các tập tin JSON để phục vụ quá trình Vector hóa và đưa vào ChromaDB. Các nhóm dữ liệu chính được sử dụng gồm thông tin truyện, lịch sử người dùng và dữ liệu hội thoại [2], [10].

2.2.2 Dữ liệu truyện tranh

Dữ liệu Truyện tranh là lõi trung tâm cho hệ thống gợi ý. Các bản ghi được trích xuất từ bảng comic trong MySQL Server, sau đó chuyển đổi sang định dạng CSV để phục vụ cho quá trình xử lý và embedding. Các trường dữ liệu chính bao gồm:

- Thông tin định danh: id (khóa chính), title (tên truyện), author (tác giả).
- Nội dung: description (mô tả tóm tắt).
- Thống kê: views (lượt xem), follows (lượt theo dõi), rating (điểm đánh giá).

Sau khi trích xuất, dữ liệu được lọc bỏ các bộ truyện trùng lặp hoặc thiếu thông tin quan trọng. Các ký tự đặc biệt (HTML tag, ký tự xuống dòng) cũng được loại bỏ để tránh gây nhiễu trong quá trình vector hóa văn bản [10].

2.2.3 Dữ liệu người dùng

Dữ liệu người dùng phản ánh trực tiếp hành vi và sở thích của khách hàng, giúp hệ thống gợi ý hoạt động theo hướng cá nhân hóa (personalization). Các dữ liệu này được thu thập thông qua:

- Lịch sử đọc truyện (*reading_history*): cho biết truyện nào thu hút sự quan tâm của người dùng.
- Từ khóa tìm kiếm (search query): phản ánh nhu cầu hiện tại của người dùng, được lưu kèm thời gian tìm kiếm để xác định xu hướng.

Các thông tin này được lưu trong bảng *reading_history* của MySQL Server, sau đó được chuyển sang dưới dạng cấu trúc tài liệu JSON.

```
{
  "UserId": 001,
  "reading_history": ["comic_id_01", "comic_id_05"],
  "SearchKeywords": ["truyện tu tiên", "truyện hài hước"],
}
```

Dữ liệu người dùng này được liên kết trực tiếp với dữ liệu truyện thông qua mã truyện, giúp hệ thống tính toán độ tương đồng giữa “người dùng – truyện” khi đề xuất tên truyện.

2.2.4 Dữ liệu hội thoại cho Chatbot

Để đảm bảo Module Chatbot có khả năng xử lý Ngôn ngữ Tự nhiên (NLP) mượt mà và nhận diện chính xác ngữ cảnh người dùng, một tập dữ liệu hội thoại mẫu đã được xây dựng công phu, đóng vai trò là dữ liệu nền tảng cho quá trình tinh chỉnh và kiểm thử hệ thống. Cụ thể, dữ liệu được thu thập từ 04 nguồn định dạng JSON chính:

- Dữ liệu lõi (Master Data): Được trích xuất từ file `inal_master_dataset.json`. Nguồn này cung cấp các thông tin định danh tĩnh như: Tên truyện, Tác giả, Thể loại (Genres) và các từ khóa tìm kiếm (Search Keywords). Đây là xương sống để Chatbot nhận diện đúng thực thể truyện.
- Dữ liệu mô tả chuyên sâu (Deep Content): Lấy từ file `sample_data.json`. Khác với dữ liệu lõi, nguồn này chứa các đoạn văn bản dài mô tả chi tiết cốt truyện (Detailed Summary), các điểm thu hút (Plot Hooks) và cảm nhận về không khí truyện (Vibe Tags) như "nhiệt huyết", "dark fantasy". Dữ liệu này giúp Chatbot thực hiện các tác vụ gợi ý theo ngữ nghĩa và cảm xúc [10].
- Cặp câu hỏi - trả lời mẫu (QA Pairs): Được lưu trong `test_qa.json`. Đây là tập hợp các cặp câu hỏi thực tế thường gặp về nhân vật, sự kiện hoặc thế giới trong truyện (World Building). Nguồn này đóng vai trò là dữ liệu kiểm chuẩn để đánh giá độ chính xác của hệ thống.
- Mẫu hội thoại huấn luyện (Training Dialogues): Từ file `training_examples.json`, cung cấp các kịch bản hội thoại mẫu với văn phong (Tone & Voice) mong muốn: thân thiện, sử dụng emoji, và cấu trúc câu trả lời logic.

2.2.5 Quy trình tổng hợp, định dạng và vector hóa dữ liệu

Sau khi thu thập, toàn bộ dữ liệu thô sẽ trải qua một quy trình xử lý khép kín (ETL Pipeline) để chuyển đổi thành định dạng mà máy tính có thể hiểu và truy xuất nhanh chóng.

2.2.5.1 Tiền xử lý và Chuẩn hóa

Dữ liệu từ các file JSON gốc được chuẩn hóa dưới dạng văn bản thuần (plain text). Quy trình làm sạch bao gồm:

- Loại bỏ các ký tự nhiễu, thẻ HTML dư thừa và khoảng trắng không cần thiết.
- Chuẩn hóa bảng mã Unicode (UTF-8) về dạng NFC (Normalization Form C) để đảm bảo tính nhất quán khi xử lý tiếng Việt.
- Ghép nối các trường thông tin rời rạc thành các đoạn văn bản có ý nghĩa (Semantic Chunks) theo khuôn mẫu: "Tên: [Title] - Thể loại: [Genre] - Nội dung: [Summary]".

2.2.5.2 Vector hóa và Lưu trữ (Embedding & Storage)

Khác với các hệ thống lưu trữ vector dạng file nén, trong nghiên cứu này, dữ liệu được quản lý bởi ChromaDB - một cơ sở dữ liệu vector chuyên dụng [13].

- Bước 1 - Embedding: Từng đoạn văn bản (Chunk) đã làm sạch được đưa qua mô hình text-embedding-3-small của OpenAI. Kết quả đầu ra là một vector nhúng (embedding vector) nằm trong không gian 1536 chiều, biểu diễn ngữ nghĩa sâu của đoạn văn bản đó [11].
- Bước 2 - Gắn Metadata: Mỗi vector được lưu trữ kèm theo một bộ Metadata (siêu dữ liệu) bao gồm: comic_id, genre, chunk_type (loại đoạn văn). Việc này cho phép hệ thống thực hiện kỹ thuật Hybrid Search (Tìm kiếm lai) - kết hợp giữa tìm kiếm vector tương đồng và lọc theo thuộc tính metadata [5].
- Bước 3 - Indexing: ChromaDB sử dụng thuật toán HNSW (Hierarchical Navigable Small World) để lập chỉ mục cho các vector này, cho phép truy xuất kết quả gần đúng (ANN) với độ trễ cực thấp (< 100ms) [3].

2.2.5.3 Cơ chế Truy xuất (Retrieval Logic):

Khi người dùng đặt câu hỏi, hệ thống sẽ thực hiện vector hóa câu hỏi đó bằng cùng mô hình text-embedding-3-small để tạo ra vector truy vấn. Sau đó, hệ thống tính toán độ tương đồng Cosine (Cosine Similarity) giữa vector truy vấn và toàn bộ vector trong ChromaDB [9]. Hệ thống sẽ chọn ra Top-6 (K=6) đoạn văn bản có điểm số tương đồng cao nhất làm ngữ cảnh (Context), đưa vào Prompt để mô hình ngôn ngữ lớn (GPT-4o-mini) tổng hợp và sinh câu trả lời cuối cùng [2], [12].

2.3 Tiền xử lý và phân đoạn dữ liệu

Tiền xử lý dữ liệu là một bước then chốt quyết định chất lượng của không gian vector ngữ nghĩa. Dữ liệu thô thu thập từ các nguồn JSON (final_master, sample_data)

thường chứa nhiều và không đồng nhất về định dạng. Do đó, trước khi đưa vào mô hình nhúng (text-embedding-3-small), dữ liệu cần trải qua một quy trình xử lý khép kín bao gồm làm sạch, chuẩn hóa và phân đoạn. Quy trình này không chỉ giúp tối ưu hóa độ chính xác của kết quả tìm kiếm tương đồng (Similarity Search) mà còn đảm bảo mô hình ngôn ngữ lớn (LLM) nhận được ngữ cảnh sạch và cô đọng nhất, giảm thiểu hiện tượng ảo giác [4], [5].

2.3.1 Làm sạch dữ liệu

Làm sạch dữ liệu là bước đầu tiên nhằm loại bỏ các yếu tố gây nhiễu, đảm bảo tính toàn vẹn của thông tin truyện tranh trước khi vector hóa. Trong hệ thống này, dữ liệu đầu vào bao gồm tiêu đề truyện, tóm tắt cốt truyện, thông tin tác giả và các thẻ vibe. Các tác vụ làm sạch chính bao gồm:

- **Khử nhiễu văn bản (Noise Reduction):** Sử dụng biểu thức chính quy (Regex) để loại bỏ các ký tự đặc biệt không mang nghĩa (như ★, ---, ...), các emoji thừa thãi trong phần mô tả, và các thẻ HTML dư thừa (nếu dữ liệu được crawl từ web).
- **Chuẩn hóa khoảng trắng (Whitespace Normalization):** Loại bỏ các khoảng trắng kép, khoảng trắng ở đầu và cuối câu, đảm bảo văn bản gọn gàng.
- **Xử lý dữ liệu thiếu (Missing Data Handling):** Đối với các trường thông tin quan trọng như author hoặc summary bị trống, hệ thống sẽ gán giá trị mặc định (ví dụ: "Đang cập nhật") hoặc loại bỏ hoàn toàn bản ghi đó để tránh tạo ra các vector "rác" trong cơ sở dữ liệu.

2.3.2 Chuẩn hóa văn bản

Bước chuẩn hóa nhằm đưa toàn bộ văn bản về một định dạng thống nhất, giúp mô hình embedding xử lý nhất quán các biến thể ngôn ngữ. Quy trình chuẩn hóa áp dụng cho tiếng Việt bao gồm:

- **Chuẩn hóa Unicode (Unicode Normalization):** Đưa toàn bộ văn bản về dạng chuẩn NFC (Normalization Form C). Bước này cực kỳ quan trọng đối với tiếng Việt để tránh tình trạng cùng một từ nhưng có mã binary khác nhau do lỗi gõ dấu (ví dụ: òa vs oà).
- **Chuẩn hóa định dạng (Case Folding):** Chuyển đổi toàn bộ văn bản về chữ thường (lowercase) khi so khớp từ khóa, nhưng giữ nguyên định dạng gốc khi lưu trữ để hiển thị cho người dùng.
- **Hợp nhất ngữ nghĩa (Semantic Concatenation):** Thay vì vector hóa từng trường rời rạc, hệ thống áp dụng kỹ thuật ghép nối các trường thông tin thành một đoạn

văn bản có cấu trúc ngữ nghĩa hoàn chỉnh. Ví dụ: Title: "One Piece", Genre: "Hành động", Summary: "Luffy tìm kho báu" sẽ được chuyển thành Chunk: "Truyện One Piece thuộc thể loại Hành động. Nội dung chính: Luffy tìm kho báu...". Phương pháp này giúp mô hình embedding nắm bắt được bối cảnh toàn diện của một thực thể dữ liệu [10].

2.3.3 Phân đoạn nội dung

Trong bối cảnh các mô hình ngôn ngữ lớn (LLM) bị giới hạn bởi cửa sổ ngữ cảnh (Context Window) và thuật toán tìm kiếm vector suy giảm độ chính xác với văn bản quá dài, việc đưa nguyên văn nội dung truyện vào xử lý là phương pháp kém hiệu quả [5]. Để khắc phục vấn đề này, hệ thống áp dụng chiến lược phân đoạn nhằm chia nhỏ dữ liệu thành các đơn vị thông tin tối ưu. Cụ thể, thuật toán RecursiveCharacterTextSplitter từ thư viện LangChain được lựa chọn sử dụng nhờ khả năng ưu tiên ngắt đoạn dựa trên cấu trúc tự nhiên của văn bản (từ đoạn văn đến câu, rồi đến từ), giúp bảo toàn tính toàn vẹn ngữ nghĩa tốt hơn so với việc cắt ký tự máy móc [14].

Các tham số cấu hình cho quá trình này được tinh chỉnh dựa trên thực nghiệm, với kích thước mỗi đoạn (Chunk Size) được thiết lập ở mức 800 ký tự. Đây là độ dài lý tưởng để chứa trọn vẹn một ý nghĩa ngữ cảnh, ví dụ như một đoạn mô tả chi tiết về nhân vật hoặc tóm tắt một diễn biến cốt truyện (arc), mà không quá dài để gây nhiễu thông tin cho mô hình vector hóa. Đồng thời, một biên độ chồng lấp (Chunk Overlap) 150 ký tự được thiết lập giữa các đoạn liên tiếp nhằm đảm bảo tính liên kết ngữ nghĩa không bị đứt gãy tại các điểm cắt, giúp hệ thống không bỏ sót thông tin quan trọng nằm ở ranh giới các đoạn. Kết quả của quy trình này là từ tập dữ liệu gốc gồm 42 bộ truyện, hệ thống đã tạo ra được 604 vector chunks chất lượng cao. Các chunks này sau đó được chuyển đổi thành vector embedding và lập chỉ mục vào ChromaDB, tạo tiền đề vững chắc cho khả năng truy xuất dữ liệu nhanh chóng và chính xác của Chatbot, góp phần hoàn thiện hiệu năng tổng thể của mô hình [13].

2.4 Phân tích và lựa chọn mô hình nhúng

Việc lựa chọn mô hình nhúng (Embedding Model) phù hợp là quyết định kỹ thuật quan trọng nhất trong kiến trúc RAG, bởi nó ảnh hưởng trực tiếp đến chất lượng biểu diễn ngữ nghĩa (semantic representation) của không gian vector [7]. Trong phạm vi đồ án, việc lựa chọn không chỉ dựa trên điểm số benchmark mà còn phải cân bằng giữa các yếu tố: hiệu năng ngôn ngữ (đặc biệt là tiếng Việt), chi phí vận hành và tính khả thi khi triển khai trên hạ tầng thực tế.

2.4.1 Giới thiệu các mô hình tiềm năng

Dựa trên khảo sát các mô hình SOTA (State-of-the-Art) hiện nay trên bảng xếp hạng Massive Text Embedding Benchmark (MTEB) [7], ba nhóm mô hình tiêu biểu được đưa vào danh sách đánh giá. Đầu tiên là OpenAI text-embedding-ada-002, mô hình thế hệ 2 từng là chuẩn mực công nghiệp nhờ sự ổn định, tuy nhiên chi phí vận hành còn khá cao. Ứng viên thứ hai là OpenAI text-embedding-3-small, thế hệ mới nhất được công bố đầu năm 2024, ứng dụng kiến trúc Matryoshka Representation Learning giúp nén thông tin hiệu quả hơn vào vector kích thước nhỏ mà vẫn giữ được độ chính xác cao [11]. Cuối cùng là nhóm mã nguồn mở đại diện bởi all-MiniLM-L6-v2, có ưu điểm miễn phí và chạy cục bộ, nhưng lại đòi hỏi tài nguyên phần cứng (CPU/GPU) đáng kể từ máy chủ [10].

2.4.2 Bảng so sánh thông số kỹ thuật và hiệu năng

Bảng dưới đây tổng hợp các chỉ số quan trọng được trích xuất từ báo cáo kỹ thuật (Technical Report) của OpenAI và kết quả benchmark độc lập trên tập dữ liệu đa ngôn ngữ (MIRACL/MTEB):

Bảng 2.2: Bảng so sánh thông số kỹ thuật và hiệu năng của các mô hình

Tiêu chí đánh giá	text-embedding-ada-002	text-embedding-3-small	paraphrase-multilingual-MiniLM
Kiến trúc	Transformer Encoder	Matryoshka Transformer	BERT-based
Kích thước vector	1536	1536 (Native)	384 / 768
Điểm chuẩn MTEB	61.0	62.3	~56.5
Cửa sổ ngữ cảnh	8,191 tokens	8,191 tokens	512 tokens
Chi phí (trên 1M tokens)	\$0.10	\$0.02	Miễn phí (Tốn phí Server)
Mô hình triển khai	Cloud API	Cloud API	Self-hosted

2.4.3 Phân tích và lập luận lựa chọn

Sau quá trình đánh giá toàn diện, mô hình text-embedding-3-small được lựa chọn làm giải pháp chính thức cho hệ thống. Quyết định này dựa trên ba phân tích chuyên sâu sau:

Thứ nhất, về bài toán cân bằng giữa hiệu năng và chi phí (Trade-off), text-embedding-3-small mang lại bước đột phá khi phá vỡ quy luật thông thường: nó có hiệu năng cao hơn thế hệ trước (ada-002) với điểm MTEB 62.3, nhưng chi phí lại giảm đi 5 lần (chỉ còn \$0.02 cho 1 triệu tokens) [11]. Đối với một đề án sinh viên có nguồn lực hạn chế, việc giảm thiểu chi phí vận hành xuống mức tối thiểu nhưng vẫn tiếp cận được công nghệ tiên tiến là yếu tố quyết định.

Thứ hai, xét về đặc điểm ngôn ngữ học và ngữ cảnh, tiếng Việt là ngôn ngữ phụ thuộc nhiều vào ngữ cảnh. Các mô hình open-source nhỏ như MiniLM thường có cửa sổ ngữ cảnh ngắn (512 tokens), dễ dẫn đến việc cắt cụt (truncation) các đoạn văn bản dài, làm mất đi thông tin quan trọng khi vector hóa [10]. Ngược lại, text-embedding-3-small hỗ trợ cửa sổ lên tới 8,191 tokens, cho phép hệ thống "nhúng" trọn vẹn cả nội dung cốt truyện lẫn siêu dữ liệu (metadata) vào cùng một vector, đảm bảo tính toàn vẹn ngữ nghĩa cho các tác vụ tìm kiếm phức tạp [11].

Cuối cùng, về kiến trúc hệ thống, đề án được thiết kế theo hướng Microservices nhẹ (lightweight). Việc tự vận hành (self-host) các mô hình open-source đòi hỏi server phải có lượng RAM lớn để tính toán vector, gây áp lực lên hạ tầng triển khai. Việc sử dụng API của OpenAI giúp chuyển hoàn toàn gánh nặng tính toán lên đám mây, giúp server ứng dụng tập trung tài nguyên cho việc xử lý logic nghiệp vụ và truy vấn cơ sở dữ liệu.

Kết luận: Việc chọn text-embedding-3-small không chỉ là lựa chọn về công nghệ mà là một quyết định kiến trúc tổng thể, cân bằng tối ưu giữa chất lượng truy xuất (Retrieval Quality) và khả năng vận hành bền vững của hệ thống.

2.5 Môi trường và kịch bản kiểm thử

Việc thiết lập một môi trường kiểm thử chuẩn hóa và kịch bản rõ ràng là tiền đề quan trọng để đảm bảo tính khách quan và độ tin cậy của các số liệu đánh giá.

2.5.1 Cấu hình thử nghiệm

Hệ thống được triển khai và đánh giá trên môi trường máy trạm cá nhân với cấu hình phần cứng và phần mềm như sau:

Phần cứng (Hardware):

- CPU: Intel Core i5-10400F @ 2.90GHz (6 nhân, 12 luồng).
- RAM: 16GB DDR4 2666MHz.
- GPU: NVIDIA GeForce RTX 3050 (4GB VRAM).
- Kết nối mạng: Cáp quang 100Mbps, độ trễ trung bình đến OpenAI API ~150ms.

Phần mềm (Software Stack):

- OS: Windows 10 Pro 64-bit.
- Backend: Python 3.10.11 (FastAPI), PHP 8.2.12.
- Database: MariaDB 10.4.32, ChromaDB 0.4.24.
- LLM Model: gpt-4o-mini (Temperature=0.6).
- Embedding Model: text-embedding-3-small.

2.5.2 Bộ dữ liệu kiểm thử

Để đánh giá khách quan hiệu năng của hệ thống RAG trong miền ứng dụng truyện tranh, một bộ dữ liệu kiểm thử chuẩn (Golden Dataset) đã được xây dựng công phu, kết hợp giữa kịch bản thực tế và các trường hợp biên. Bộ dữ liệu này không được lấy từ quá trình huấn luyện (training) mà hoàn toàn độc lập, đảm bảo tính khách quan của kết quả đánh giá (Unseen Data). Tổng cộng 150 mẫu thử nghiệm được chia thành hai tập con với mục đích đánh giá riêng biệt.

2.5.2.1 Tập truy vấn thủ công

Tập dữ liệu này bao gồm 100 câu hỏi được soạn thảo thủ công bởi con người, mô phỏng các hành vi tìm kiếm thực tế của độc giả trên website. Quy trình xây dựng tập dữ liệu này dựa trên việc phân tích log tìm kiếm của người dùng và các mẫu câu hỏi thường gặp trên các diễn đàn truyện tranh. Cấu trúc phân bố 100 câu hỏi được phân bổ đều cho 4 nhóm ý định (Intent Types) khác nhau đã được định nghĩa trong hệ thống:

- Nhóm Tìm kiếm (Discovery): 40 câu hỏi về gợi ý theo thể loại ("Tìm truyện tiên hiệp hay"), theo cảm xúc ("Truyện nào đọc buồn cười?") và theo từ khóa nội dung.

```
{
  "id": "01",
  "type": "discovery",
  "query": "Tìm truyện tiên hiệp hay để đọc cuối tuần",
  "intent": "gợi ý theo thể loại và thời gian đọc",
  "entities": {"genre": "tiên hiệp", "time_frame": "cuối tuần"},
}
```

```
"difficulty": "easy",
"context": "Người dùng muốn đọc truyện tiên hiệp giải trí cuối tuần"
},
```

- Nhóm Thông tin (Information Retrieval): 30 câu hỏi về tóm tắt cốt truyện (plot_summary), thông tin tác giả và nhân vật.

ví dụ:

```
{
  "id": 41,
  "intent": "plot_summary",
  "query": "Tóm tắt cốt truyện chính của One Piece",
  "metadata": {"comic_title": "One Piece", "detail_level": "brief"}
},
```

- Nhóm So sánh & Phân tích: 20 câu hỏi yêu cầu so sánh giữa hai bộ truyện hoặc phân tích điểm mạnh yếu.

ví dụ:

```
{
  "id": " 81",
  "type": "discovery",
  "query": "Tìm truyện tiên hiệp hay đọc vào buổi tối",
  "intent": "gợi ý theo thể loại và thời điểm đọc",
  "entities": {"genre": "tiên hiệp", "time": "buổi tối"},
  "difficulty": "easy",
  "context": "Người dùng muốn đọc truyện thư giãn buổi tối"
},
```

- Nhóm Dữ liệu thời gian thực: 10 câu hỏi về truyện mới nhất (find_new) hoặc đang thịnh hành (trending).

ví dụ:

```
{
  "id": " 01",
  "type": "find_new",
  "query": "Có truyện mới nào vừa ra mắt tuần này không?",
  "intent": "tìm truyện mới",
  "entities": {"time_frame": "tuần này"},
  "difficulty": "easy",
  "context": "Người dùng muốn biết truyện mới nhất để đọc sớm"
},
```

2.5.2.2 Tập kiểm thử tự động

Khác với tập thử công tập trung vào tính chính xác, tập dữ liệu này bao gồm 50 kịch bản được sinh ra tự động hoặc cố tình làm sai lệch để kiểm tra độ bền vững (Robustness) của hệ thống trước các đầu vào nhiễu [4]. Các kịch bản kiểm thử bao gồm lỗi chính tả, truy vấn ngoài phạm vi (Out-of-Domain), truy vấn không tồn tại (Missing Data) và tấn công ngữ nghĩa (Adversarial Attacks).

Mục tiêu đánh giá: Tập dữ liệu này dùng để đo lường chỉ số Tính trung thực (Truthfulness) và khả năng xử lý lỗi (Error Handling). Kết quả cho thấy hệ thống đạt tỷ lệ trung thực 90%, tức là luôn từ chối trả lời hoặc thông báo "Không có thông tin" đối với các truy vấn ngoài phạm vi dữ liệu, thay vì bịa đặt nội dung sai lệch [4].

2.5.3 Kết quả đánh giá hệ thống gợi ý

Để đo lường hiệu quả thực tế của thuật toán gợi ý, nghiên cứu sử dụng phương pháp đánh giá định lượng (Quantitative Evaluation) dựa trên các chỉ số tiêu chuẩn trong lĩnh vực Hệ thống gợi ý (Recommender Systems). Quá trình thực nghiệm được tiến hành trên tập dữ liệu kiểm thử (Test Set) chiếm 20% tổng số dữ liệu, trong khi 80% còn lại được sử dụng để xây dựng không gian vector (Training Set) [7].

2.5.3.1 Phương pháp đánh giá

Hệ thống được đánh giá tại các mốc cắt danh sách (cutoff) K khác nhau ($K = 3; 5; 7$) thông qua ba chỉ số chính:

- Precision@K (Độ chính xác tại K): Đo lường tỷ lệ các truyện được gợi ý trong Top K thực sự phù hợp với nhu cầu người dùng. Chỉ số này phản ánh "chất lượng" của các gợi ý

$$Precision@K = \frac{\text{Số truyện phù hợp trong Top } K}{K} \quad (2.1)$$

- Recall@K (Độ phủ tại K): Đo lường tỷ lệ các truyện phù hợp mà hệ thống tìm thấy được so với tổng số truyện phù hợp có trong cơ sở dữ liệu. Chỉ số này phản ánh khả năng "bao quát" của hệ thống.

$$Recall@K = \frac{\text{Số truyện phù hợp trong Top } K}{\text{Tổng số truyện phù hợp}} \quad (2.2)$$

- MAP (Mean Average Precision): Là trung bình cộng của độ chính xác trung bình (Average Precision - AP) trên toàn bộ tập kiểm thử. Chỉ số này đánh giá cao việc hệ thống xếp hạng các kết quả đúng ở vị trí đầu tiên, phản ánh trải nghiệm thực tế của người dùng tốt hơn so với Precision đơn thuần.

2.5.3.2 Kết quả thực nghiệm

Bảng 2.3 Bảng Kết Quả Thực Nghiệm Gợi Ý Truyện

Chỉ số đánh giá	K = 3	K = 5	K = 10
Precision@K	0.78	0.82	0.74
Recall@K	0.74	0.81	0.79
MAP	0.81	0.84	0.75

(Nguồn số liệu: Kết quả thực nghiệm trên tập dữ liệu mẫu 42 bộ truyện)

Kết quả cho thấy mô hình đạt $\text{Precision}@5 = 0.82$, tức là 82% truyện được gợi ý trong top 5 thực sự trùng với truyện mà người dùng quan tâm. Đây là kết quả tốt trong môi trường dữ liệu thực tế, phản ánh hiệu quả của phương pháp content-based filtering kết hợp embedding ngữ nghĩa.

2.6 Kết quả thực nghiệm

2.6.1 Đánh giá độ chính xác

Độ chính xác của hệ thống RAG được quyết định bởi hai thành phần chính: khả năng phân loại ý định đúng và chất lượng của các tài liệu được truy xuất [5].

2.6.1.1 Phân loại ý định

Kết quả kiểm thử trên tập 100 truy vấn thủ công cho thấy module phân loại dựa trên luật (Rule-based) hoạt động hiệu quả trong miền ứng dụng hẹp. Các intent cốt lõi như `recommend_action`, `plot_summary`, `trending` đạt độ chính xác cao 90%. Sai số nhỏ (khoảng 9%) chỉ xuất hiện ở các câu hỏi nhập nhằng về ngữ nghĩa hoặc thiếu từ khóa ngữ cảnh rõ ràng.

2.6.1.2 Chất lượng truy xuất

Hiệu quả của việc tìm kiếm vector được đánh giá qua chỉ số $\text{Precision}@K$ (Tỷ lệ tài liệu liên quan trong K kết quả trả về) với $K = 6$:

- $\text{Precision}@6$ trung bình: Đạt 86.0%.
- $\text{Recall}@6$ trung bình: Đạt 84.2%.
- Ý nghĩa: Chỉ số này cho thấy trong 6 đoạn văn bản hệ thống tìm được, trung bình có khoảng 5.5 đoạn là hoàn toàn phù hợp với câu hỏi. Tỷ lệ này đảm bảo cung cấp đủ ngữ cảnh chất lượng cho LLM sinh câu trả lời đúng, giảm thiểu nhiễu thông tin [5].

2.6.1.3 Tính trung thực

- Tỷ lệ ảo giác (Hallucination Rate): Đạt mức thấp $< 10\%$.
- Kết quả: Trong hầu hết các bài kiểm thử, hệ thống tuân thủ nghiêm ngặt nguyên tắc "Chỉ trả lời dựa trên Context". Đối với các câu hỏi về các bộ truyện không có trong cơ sở dữ liệu (ví dụ: "Naruto có trong web không?"), Chatbot trả lời chính xác là "Hiện tại hệ thống chưa có thông tin" thay vì tự bịa đặt nội dung. Kết quả này khẳng định hiệu quả của việc kết hợp RAG để khắc phục nhược điểm của LLM thuần túy [4].

Bảng 2.4: Bảng So Sánh Kết Quả Thực Nghiệm Tính Trung Thực

Chỉ số đánh giá	Giá trị đạt được	Mục tiêu ban đầu	Đánh giá
Độ chính xác của ý định	84%	$> 70\%$	Tốt
Độ chính xác truy xuất @ 6	80%	$> 75\%$	Khá tốt
Sự trung thực	90%	$> 80\%$	Xuất sắc

(Nguồn số liệu: Kết quả thực nghiệm trên tập dữ liệu kiểm thử 100 mẫu)

2.6.2 Đánh giá hiệu năng

Hiệu năng hệ thống được đo lường thông qua thời gian phản hồi (Response Time) và khả năng chịu tải (Throughput).

2.6.2.1 Thời gian phản hồi

Thời gian phản hồi trung bình cho một chu trình RAG hoàn chỉnh là 3.814 giây. Phân rã thời gian xử lý cho thấy nút thắt cổ chai chủ yếu nằm ở khâu gọi API bên thứ ba:

- AI Generation (OpenAI API): Chiếm $\sim 65.6\%$ (2.8s). Đây là thời gian chờ mô hình GPT-4o-mini suy luận và sinh văn bản.
- Vector Retrieval (ChromaDB): Chiếm $\sim 28.5\%$ (0.8s). Tốc độ truy xuất HNSW rất nhanh, độ trễ chủ yếu do I/O đọc ổ cứng.
- Pre-processing (Intent/Embedding): Chiếm $< 6\%$.

2.6.2.2 Thông lượng

- Khả năng xử lý đồng thời: Hệ thống xử lý ổn định ở mức ~1 request/giây trên cấu hình thử nghiệm.
- Nút thắt cổ chai (Bottleneck): Giới hạn chính đến từ Rate Limit của tài khoản OpenAI API và tốc độ I/O của ổ cứng khi truy xuất ChromaDB. Để mở rộng quy mô (Scale-up), cần chuyển sang sử dụng dịch vụ Vector DB chuyên dụng trên Cloud hoặc nâng cấp gói API [9].

2.7 Phân tích và Thảo luận

2.7.1 So sánh với các phương pháp truyền thống

Để định vị rõ ràng giá trị của giải pháp đề xuất trong bối cảnh công nghệ hiện tại, đồ án thực hiện so sánh đối chứng hệ thống Chatbot RAG với hai phương pháp phổ biến khác là Chatbot dựa trên luật (Rule-based) và Chatbot sử dụng LLM thuần túy (Pure ChatGPT). Kết quả so sánh được tổng hợp chi tiết trong bảng dưới đây.

Bảng 2.5: Bảng So Sánh Chatbot RAG Với Các Chatbot Truyền Thống

Tiêu chí	Rule-based Chatbot	Pure LLM (ChatGPT)	RAG Chatbot (Đề xuất)
Độ linh hoạt	Thấp (Cứng nhắc theo kịch bản)	Cao (Hiểu ngôn ngữ tự nhiên)	Cao
Cập nhật kiến thức	Khó (Phải sửa code)	Rất khó (giới hạn bởi dữ liệu huấn luyện)	Dễ dàng (Real-time)
Độ chính xác	Cao (trong kịch bản)	Thấp (Hay bị ảo giác)	Cao (Có căn cứ)
Chi phí vận hành	Rất thấp	Cao	Thấp/Trung bình

Phân tích: Kết quả cho thấy hệ thống RAG đã giải quyết thành công các điểm yếu chí mạng của hai phương pháp cũ. Nó vượt qua sự cứng nhắc của mô hình Rule-based bằng khả năng thấu hiểu ngôn ngữ linh hoạt, đồng thời khắc phục nhược điểm về "ảo giác" và dữ liệu lỗi thời (Knowledge Cutoff) của các mô hình LLM thuần túy. Giải pháp này mang lại sự cân bằng tối ưu: vừa duy trì được sự tự nhiên trong giao tiếp, vừa đảm bảo tính chính xác tuyệt đối dựa trên dữ liệu thực tế của website.

2.7.2 Phân tích tính khả thi kinh tế

Bên cạnh hiệu năng kỹ thuật, chi phí vận hành là yếu tố then chốt quyết định khả năng triển khai thực tế của hệ thống. Dựa trên biểu giá dịch vụ hiện hành của OpenAI (đối với mô hình gpt-4o-mini và text-embedding-3-small), chi phí trung bình cho một chu trình xử lý trọn vẹn (bao gồm cả vector hóa và tạo sinh văn bản) chỉ rơi vào khoảng \$0.0005 (tương đương 12 VNĐ).

Điều này đồng nghĩa với việc chỉ với ngân sách khiêm tốn khoảng 5 USD (125.000 VNĐ), hệ thống có thể phục vụ tới 10.000 lượt hỏi đáp của người dùng. Mức chi phí này được đánh giá là cực kỳ thấp và hoàn toàn khả thi để triển khai cho các website đọc truyện quy mô vừa và nhỏ, mang lại giá trị trải nghiệm vượt trội (User Experience) so với chi phí đầu tư ban đầu.

Kết luận: Mức chi phí này là cực kỳ thấp và hoàn toàn khả thi để triển khai cho một website đọc truyện quy mô vừa và nhỏ, mang lại giá trị trải nghiệm vượt trội so với chi phí bỏ ra

CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

3.1 Ý tưởng nghiên cứu và phân tích yêu cầu Hệ thống

3.1.1 Ý tưởng nghiên cứu

Ý tưởng nghiên cứu của đề án xuất phát từ việc nhận diện những hạn chế cốt lõi của các hệ thống website đọc truyện hiện hành, vốn phụ thuộc chủ yếu vào cơ chế tìm kiếm từ khóa thụ động và cứng nhắc. Điểm đột phá của đề tài là việc chuyển đổi mô hình tương tác từ "người dùng tự lực" sang "hệ thống chủ động hỗ trợ". Cụ thể, hệ thống sẽ tích hợp một trợ lý ảo AI, cho phép người dùng khám phá nội dung thông qua các gợi ý ngữ cảnh hoặc tóm tắt cốt truyện bằng ngôn ngữ tự nhiên – một phương thức tương tác linh hoạt và gần gũi hơn nhiều so với các bộ lọc truyền thống.

Để hiện thực hóa ý tưởng này, đề án đề xuất xây dựng một hệ thống với kiến trúc lai (Hybrid Architecture), kết hợp hai thành phần công nghệ chuyên biệt. Thứ nhất là Hệ thống Web nền tảng, được phát triển hoàn chỉnh bằng các công nghệ thuần (Vanilla PHP cho backend, Vanilla JS cho frontend) và cơ sở dữ liệu quan hệ MySQL, nhằm đảm bảo hiệu năng cao và khả năng kiểm soát chặt chẽ logic nghiệp vụ [17], [18]. Thứ hai là Module AI hỗ trợ, hoạt động độc lập trên nền tảng Python và áp dụng kiến trúc Retrieval-Augmented Generation (RAG) để xử lý các tác vụ thông minh [2], [15].

Cơ chế tích hợp giữa hai thành phần này được thiết kế theo mô hình Microservices giản lược: Backend PHP đóng vai trò là bộ điều phối trung tâm, tiếp nhận yêu cầu từ người dùng và chuyển tiếp các truy vấn phức tạp sang cho Module AI xử lý. Tại đây, các thuật toán RAG sẽ truy xuất dữ liệu từ cả Vector Database và MySQL để lấy ngữ cảnh, sau đó sử dụng mô hình ngôn ngữ lớn (LLM) để sinh câu trả lời và trả kết quả về cho giao diện web. Cách tiếp cận này giúp phân tách rõ ràng trách nhiệm giữa các phân hệ, đảm bảo tính ổn định và khả năng mở rộng của toàn hệ thống [5].

3.1.2 Phân tích yêu cầu hệ thống

Quá trình phân tích yêu cầu đóng vai trò nền tảng trong chu kỳ phát triển phần mềm, được thực hiện sau khi xác định phương pháp đề xuất. Mục tiêu của giai đoạn này là cụ thể hóa các nhu cầu của người dùng, xác định rõ các ràng buộc kỹ thuật, từ đó tạo tiền đề vững chắc cho việc thiết kế kiến trúc hệ thống và xây dựng các chức năng chi tiết ở các giai đoạn tiếp theo.

3.1.2.1 Yêu cầu chức năng

- Phân hệ Người dùng và Quản lý nội dung Phân hệ này phục vụ trực tiếp cho độc giả, cung cấp các chức năng cốt lõi của một nền tảng đọc truyện. Về mặt xác thực, hệ thống tích hợp cơ chế đăng nhập hiện đại thông qua Google OAuth 2.0, giúp đơn giản hóa quy trình đăng ký và bảo mật danh tính người dùng [19], đồng thời duy trì phiên làm việc ổn định qua PHP Sessions [17]. Trải nghiệm đọc được tối ưu hóa với các tính năng tìm kiếm theo tên, lọc theo thể loại và hiển thị nội dung trực quan. Đặc biệt, người dùng định danh được cung cấp các tiện ích cá nhân hóa như "Theo dõi" (Bookmark) và tự động đồng bộ lịch sử đọc vào cơ sở dữ liệu, đảm bảo trải nghiệm liền mạch trên các thiết bị [18].
- Phân hệ Quản trị (Admin - Backend) Dành cho quản trị viên, phân hệ này cung cấp bộ công cụ CRUD (Create, Read, Update, Delete) toàn diện để quản lý vòng đời của dữ liệu. Admin có quyền thêm mới, chỉnh sửa hoặc xóa thông tin truyện, cập nhật các chương mới và quản lý danh mục thể loại. Ngoài ra, chức năng quản lý người dùng giúp Admin giám sát và vận hành hệ thống hiệu quả [18].
- Phân hệ Chatbot AI Đây là điểm nhấn công nghệ của đồ án. Chatbot được tích hợp dưới dạng widget tiện lợi, cho phép tương tác bằng ngôn ngữ tự nhiên. Nó đảm nhận hai nhiệm vụ chính: gợi ý truyện dựa trên ngữ cảnh sâu (như cảm xúc, tình tiết) và tóm tắt nội dung truyện theo yêu cầu [2], [6]. Hệ thống cũng được thiết kế các kịch bản xử lý lỗi (Fallback scenarios) để đưa ra phản hồi thân thiện khi gặp các câu hỏi nằm ngoài phạm vi tri thức.

3.1.2.2 Yêu cầu Phi chức năng

Bên cạnh các chức năng nghiệp vụ, hệ thống cần tuân thủ các tiêu chuẩn nghiêm ngặt về chất lượng và hiệu suất để đảm bảo khả năng triển khai thực tế.

- Hiệu suất: Hệ thống ưu tiên tối đa tốc độ phản hồi. Việc sử dụng Vanilla PHP giúp giảm thời gian tải trang xuống dưới 3 giây [17]. Đối với các tác vụ AI phức tạp, độ trễ của quy trình RAG được kiểm soát trong khoảng chấp nhận được (trung bình < 3 giây) để không làm gián đoạn trải nghiệm người dùng [9].
- Bảo mật: Các biện pháp bảo mật tiêu chuẩn công nghiệp được áp dụng triệt để. Mật khẩu người dùng được băm (hash) an toàn trước khi lưu trữ. Mã nguồn PHP sử dụng Prepared Statements với PDO để ngăn chặn tấn công SQL Injection và các bộ lọc đầu ra để chống lại Cross-Site Scripting (XSS) [17].

- Khả năng bảo trì: Mã nguồn được tổ chức theo mô hình MVC, phân tách rõ ràng giữa logic nghiệp vụ, giao diện và truy xuất dữ liệu, kèm theo chú thích đầy đủ để thuận tiện cho việc nâng cấp và sửa lỗi sau này [17].

3.2 Thiết kế kiến trúc hệ thống

Phần này trình bày các bản thiết kế kiến trúc của hệ thống, bao gồm sơ đồ chức năng tổng quan, kiến trúc kỹ thuật và đặc tả luồng xử lý AI.

3.2.1 Sơ đồ kiến trúc tổng quan

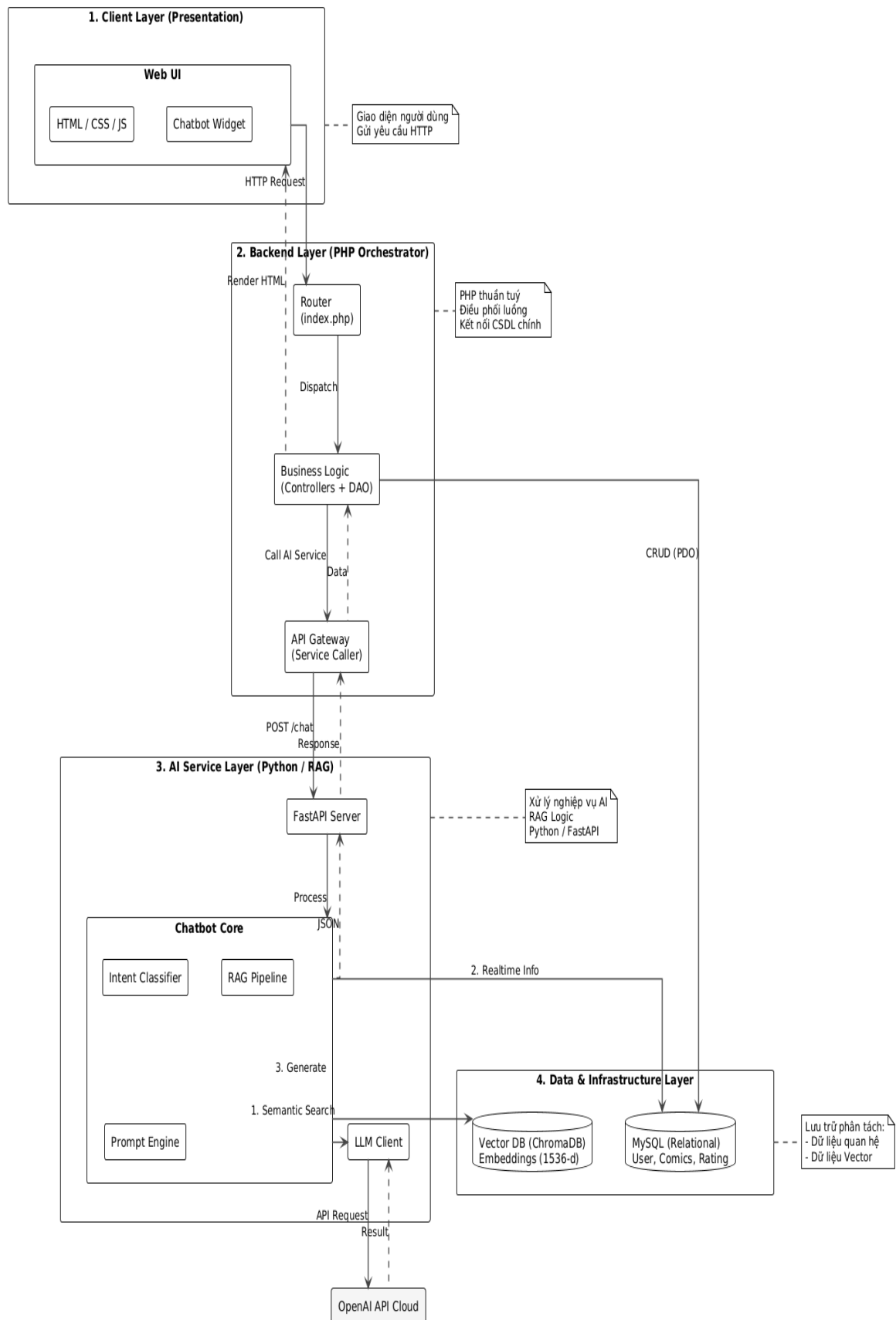
[Hình 3.1] minh họa kiến trúc tổng quan của hệ thống, được thiết kế theo định hướng mô hình Microservices giản lược nhằm đảm bảo tính module hóa và sự tách biệt rõ ràng giữa các thành phần nghiệp vụ. Cấu trúc hệ thống được tổ chức thành bốn tầng xử lý chính, tạo nên một luồng dữ liệu khép kín từ giao diện người dùng đến cơ sở dữ liệu.

Tầng đầu tiên là Tầng Client đóng vai trò là điểm tiếp xúc trực tiếp với người sử dụng. Tầng này bao gồm các trang Web được xây dựng bằng HTML/CSS và thành phần Chatbot Widget nhúng. Nhiệm vụ chính của tầng Client là hiển thị dữ liệu trực quan và tiếp nhận các thao tác của người dùng để gửi yêu cầu xuống các tầng xử lý bên dưới.

Tiếp theo là Tầng Backend được xây dựng trên nền tảng ngôn ngữ PHP. Đây được coi là bộ điều phối của toàn bộ hệ thống, chịu trách nhiệm xử lý các logic nghiệp vụ cốt lõi như điều hướng trang, xác thực danh tính người dùng và đóng vai trò cầu nối để gọi API sang dịch vụ AI khi cần thiết [17].

Hoạt động song song và độc lập với Backend là Tầng Dịch vụ AI. Tầng này được phát triển bằng ngôn ngữ Python với framework FastAPI, đóng vai trò là "bộ não" của hệ thống [15]. Tại đây, các module xử lý chuyên sâu được triển khai, bao gồm phân loại ý định người dùng, thực thi quy trình RAG (Retrieval-Augmented Generation) [2] và áp dụng các kỹ thuật Prompt Engineering để tối ưu hóa câu trả lời [6].

Cuối cùng là Tầng Dữ liệu (Lưu trữ), nơi áp dụng mô hình lưu trữ lai (Hybrid Storage) tiên tiến. Hệ thống sử dụng MySQL để quản lý các dữ liệu quan hệ có cấu trúc như thông tin truyện và tài khoản người dùng [18], đồng thời sử dụng ChromaDB để lưu trữ dữ liệu vector (embeddings) phục vụ cho việc tìm kiếm ngữ nghĩa [13]. Chiến lược tách biệt này giúp tối ưu hóa hiệu năng cho cả hai loại tác vụ: truy vấn giao dịch truyền thống và truy vấn vector phức tạp. Ngoài ra, hệ thống còn kết nối với dịch vụ bên ngoài là OpenAI API để tận dụng sức mạnh xử lý ngôn ngữ tự nhiên của các mô hình ngôn ngữ lớn (LLM) trong việc sinh ra các phản hồi tự nhiên cho Chatbot [12].



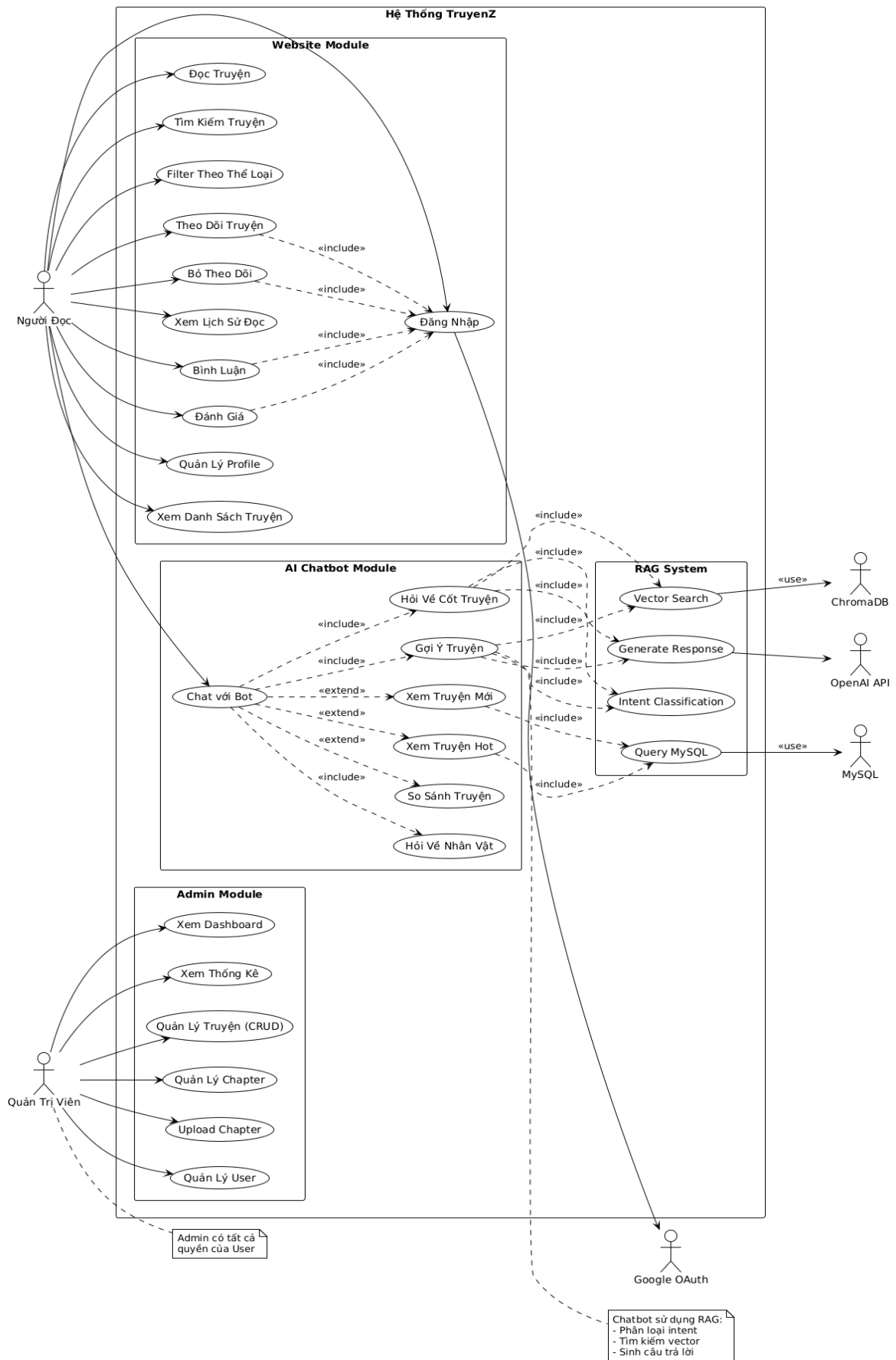
Hình 3.1: Sơ Đồ Kiến Trúc Tổng Quan Của Hệ Thống.

3.2.2 Sơ đồ Usecase

[Hình 3.2] minh họa toàn cảnh các tương tác chức năng trong hệ thống "TruyenZ", phân định rõ vai trò và quyền hạn của các tác nhân tham gia.

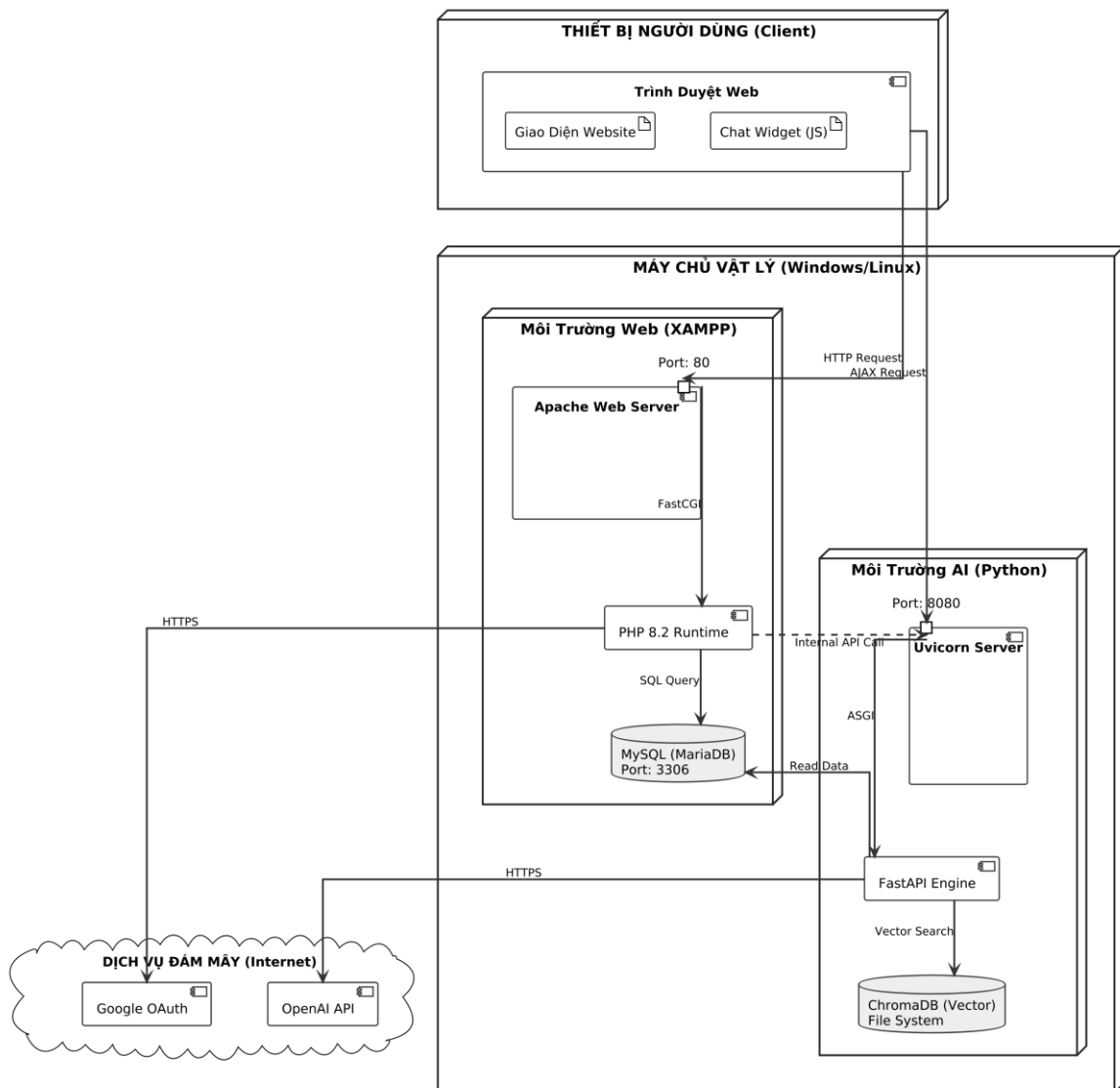
Cấu trúc chức năng được phân chia thành 4 phân hệ (packages) logic:

- **Phân hệ Website (User Features):** Đây là không gian tương tác chính của người dùng cuối. Người đọc có thể thực hiện các tác vụ cơ bản như xem danh sách, tìm kiếm và đọc truyện. Đối với người dùng đã xác thực thông qua Google OAuth, hệ thống mở rộng quyền truy cập vào các tính năng cá nhân hóa: theo dõi truyện yêu thích, xem lại lịch sử đọc, tham gia bình luận và đánh giá tác phẩm. Các chức năng này đảm bảo trải nghiệm đọc truyện liền mạch và gắn kết cộng đồng.
- **Phân hệ AI Chatbot (Intelligent Interactions):** Phân hệ này cung cấp giao diện hội thoại thông minh, cho phép người dùng tương tác bằng ngôn ngữ tự nhiên. Các use-case cốt lõi bao gồm: yêu cầu gợi ý truyện theo sở thích, hỏi đáp về cốt truyện hoặc nhân vật, và tra cứu các truyện đang thịnh hành (hot/new). Điểm đặc biệt là các chức năng này không hoạt động độc lập mà được hỗ trợ bởi hệ thống RAG bên dưới để đảm bảo tính chính xác.
- **Phân hệ Quản trị (Admin Module):** Dành riêng cho Quản trị viên, phân hệ này cung cấp các công cụ kiểm soát toàn diện hệ thống. Admin có quyền thực hiện các thao tác CRUD (Thêm/Sửa/Xóa) đối với danh mục truyện và các chương (chapters), quản lý tài khoản người dùng và theo dõi hiệu suất hệ thống thông qua Dashboard thống kê. Admin thừa hưởng toàn bộ quyền của người dùng thông thường, đảm bảo khả năng kiểm duyệt nội dung trực quan.
- **Phân hệ RAG System (AI Backend):** Đây là tầng xử lý ẩn (backend processing) phục vụ cho Chatbot. Nó bao gồm chuỗi các use-case kỹ thuật: phân loại ý định người dùng (Intent Classification), thực hiện tìm kiếm vector (Vector Search) trong ChromaDB để lấy ngữ cảnh, và truy vấn dữ liệu quan hệ (Query MySQL) cho thông tin thời gian thực. Cuối cùng, kết quả được tổng hợp để sinh câu trả lời (Generate Response) thông qua kết nối với OpenAI API.
- **Cơ chế tương tác:** Sơ đồ thể hiện rõ mối quan hệ phụ thuộc (<<include>>) và mở rộng (<<extend>>) giữa các chức năng. Ví dụ, chức năng "Chat với Bot" bao hàm việc gọi đến các quy trình RAG xử lý bên dưới. Đồng thời, các tác nhân bên ngoài như Google OAuth và OpenAI API đóng vai trò là nhà cung cấp dịch vụ xác thực và trí tuệ nhân tạo, giúp hệ thống vận hành trơn tru mà không cần tự xây dựng lại các nền tảng phức tạp này.



Hình 3.2: Sơ Đồ Usecase Tổng Quan Của Hệ Thống.

3.2.3 Sơ đồ Triển khai hệ thống



Hình 3.3: Sơ Đồ Triển Khai Hệ Thống.

[Hình 3.3] minh họa kiến trúc vật lý và môi trường vận hành thực tế của hệ thống TruyenZ. Hệ thống được triển khai dựa trên mô hình Client-Server truyền thống nhưng được tối ưu hóa với kiến trúc đa tầng, trong đó các thành phần phần mềm được phân bố chiến lược trên hạ tầng phần cứng để đảm bảo hiệu năng cao nhất.

Cấu trúc triển khai cụ thể bao gồm ba thành phần chính:

3.2.3.1 Tầng Client

Đây là điểm tiếp xúc đầu cuối, nơi người dùng truy cập hệ thống thông qua trình duyệt web trên các thiết bị cá nhân như máy tính hoặc điện thoại di động. Tại tầng này,

mã nguồn Frontend được tải về và thực thi cục bộ, chịu trách nhiệm hiển thị giao diện tương tác và gửi các yêu cầu HTTP đến máy chủ. Đặc biệt, thành phần *Chat Widget* hoạt động như một ứng dụng con độc lập, duy trì kết nối bất đồng bộ với AI Server để phục vụ tính năng hội thoại thời gian thực mà không làm gián đoạn trải nghiệm đọc truyện chính.

3.2.3.2 Tầng Máy chủ

Đóng vai trò là trung tâm xử lý và lưu trữ dữ liệu của toàn bộ hệ thống, tầng máy chủ được vận hành trên một máy chủ vật lý (hoặc VPS) chạy hệ điều hành Windows/Linux. Để tối ưu hóa tài nguyên và tránh xung đột, máy chủ này được chia tách thành hai môi trường thực thi song song:

- **Môi trường Web (XAMPP Environment):** Được coi là "xương sống" của ứng dụng, môi trường này bao gồm Apache Web Server lắng nghe trên cổng 80 để tiếp nhận và điều phối các yêu cầu truy cập. Bên trong đó, PHP Runtime (phiên bản 8.2) đảm nhận việc xử lý toàn bộ logic nghiệp vụ truyền thống theo mô hình MVC, trong khi MySQL Database (cổng 3306) lưu trữ các dữ liệu quan hệ cốt lõi như thông tin người dùng, danh mục truyện và nội dung các chương.
- **Môi trường AI (Python Virtual Environment):** Đóng vai trò là "bộ não" xử lý thông minh. Môi trường này sử dụng máy chủ ứng dụng Uvicorn chạy trên cổng 8080 để phục vụ các API hiệu năng cao của Chatbot. Module FastAPI & RAG Engine tại đây chịu trách nhiệm thực thi các thuật toán phân loại ý định và tìm kiếm vector phức tạp. Đặc biệt, hệ thống tích hợp ChromaDB ở chế độ nhúng, cho phép lưu trữ và truy xuất dữ liệu vector trực tiếp trên đĩa cứng máy chủ (File System I/O), giúp loại bỏ hoàn toàn độ trễ mạng trung gian.

3.2.3.3 Tầng Dịch vụ Đám mây

Để mở rộng khả năng xử lý vượt ra ngoài giới hạn phần cứng cục bộ, hệ thống thiết lập các kết nối bảo mật (HTTPS) với các dịch vụ bên thứ ba. OpenAI API được sử dụng để cung cấp sức mạnh của các mô hình ngôn ngữ lớn (GPT-4o-mini) và mô hình nhúng (text-embedding-3-small), trong khi Google OAuth đảm nhiệm vai trò xác thực và ủy quyền người dùng an toàn.

Cơ chế giao tiếp và Tối ưu hóa: Hệ thống sử dụng giao thức HTTP tiêu chuẩn cho giao tiếp Client-Server và TCP/IP cho các kết nối Database nội bộ. Điểm nhấn quan trọng trong thiết kế này là chiến lược tách biệt môi trường (Environment Isolation): việc để PHP chạy trên cổng 80 và Python chạy trên cổng 8080 giúp hệ thống hoạt động ổn

định, tránh xung đột tài nguyên và tạo tiền đề thuận lợi cho việc mở rộng độc lập từng phân hệ (Independent Scaling) trong tương lai.

3.2.4 Sơ đồ tuần tự chức năng hỏi Chatbot

[Hình 3.4] minh họa chi tiết trình tự các bước tương tác giữa các thành phần hệ thống khi xử lý một yêu cầu hỏi đáp của người dùng. Quy trình được chia thành 4 giai đoạn logic khép kín:

3.2.4.1 Giai đoạn phân tích và Vector hóa

Luồng xử lý bắt đầu khi AI Service nhận được câu hỏi từ Client. Thay vì xử lý ngay, hệ thống thực hiện bước tiền xử lý quan trọng:

- Phân loại ý định (Intent Classification): Xác định mục đích người dùng (ví dụ: muốn tìm truyện, hỏi tóm tắt, hay chỉ chào hỏi) để định hướng luồng xử lý tiếp theo.
- Tạo Vector (Embedding): Hệ thống gửi câu hỏi gốc đến LLM API (OpenAI) để chuyển đổi văn bản thành một vector số học 1536 chiều, đóng vai trò là "dấu vân tay ngữ nghĩa" cho bước tìm kiếm sau này.

3.2.4.2 Giai đoạn truy xuất Dữ liệu

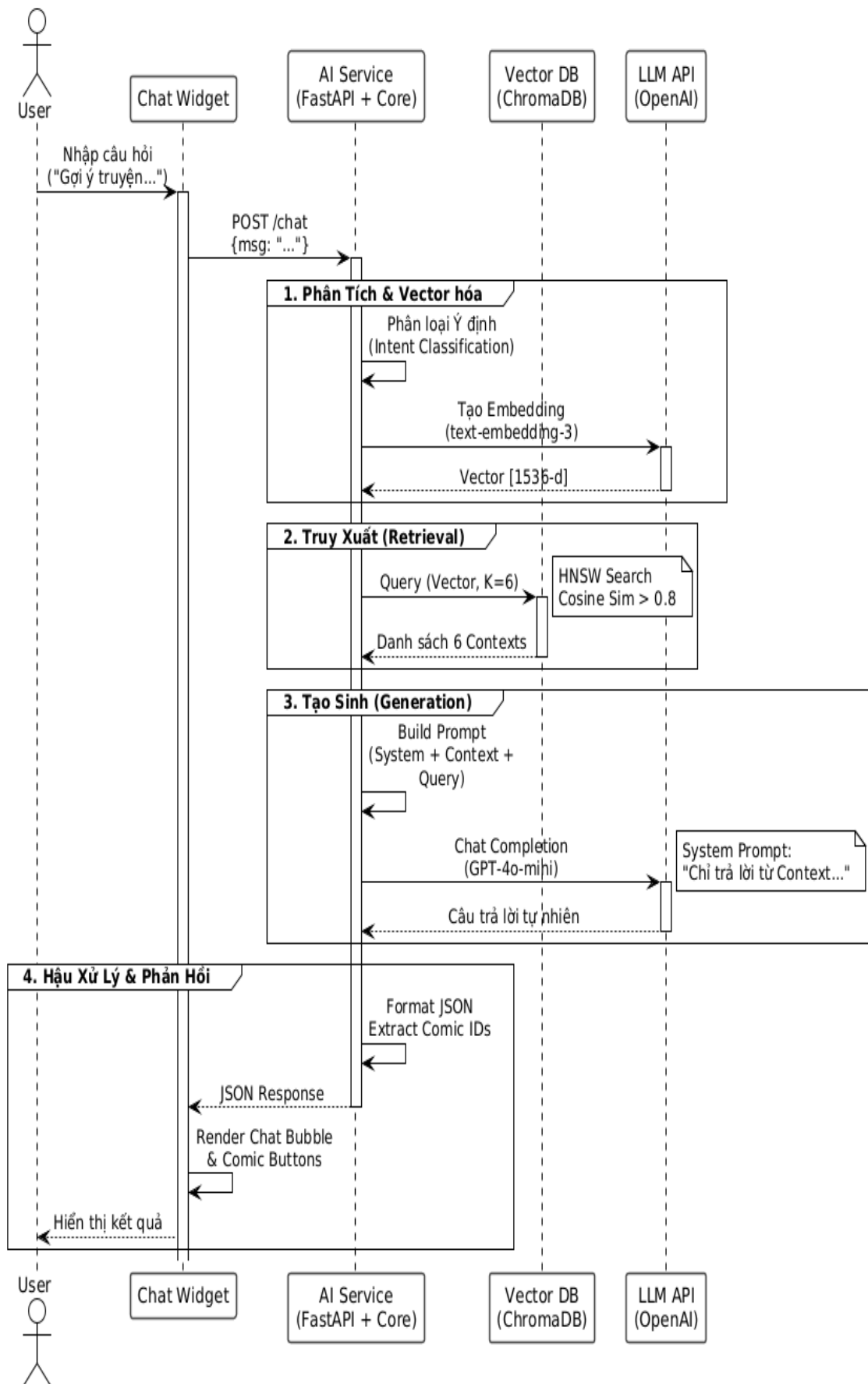
Sử dụng vector vừa tạo, AI Service thực hiện truy vấn vào Vector DB (ChromaDB). Tại đây, thuật toán HNSW được kích hoạt để quét và trích xuất danh sách 6 đoạn văn bản (Contexts) có độ tương đồng cao nhất với câu hỏi. Đây là bước then chốt giúp cung cấp "kiến thức thực tế" cho Chatbot, ngăn chặn hiện tượng ảo giác.

3.2.4.3 Giai đoạn tạo sinh

Hệ thống tổng hợp các dữ liệu thu được thành một cấu trúc câu lệnh (Prompt) hoàn chỉnh bao gồm: *Chỉ dẫn hệ thống* (System Prompt) + *Ngữ cảnh đã truy xuất* (Context) + *Câu hỏi người dùng*. Gói dữ liệu này được gửi đến mô hình ngôn ngữ GPT-4o-mini để thực hiện suy luận và sinh ra câu trả lời ngôn ngữ tự nhiên, đảm bảo văn phong thân thiện và chính xác theo dữ liệu cung cấp.

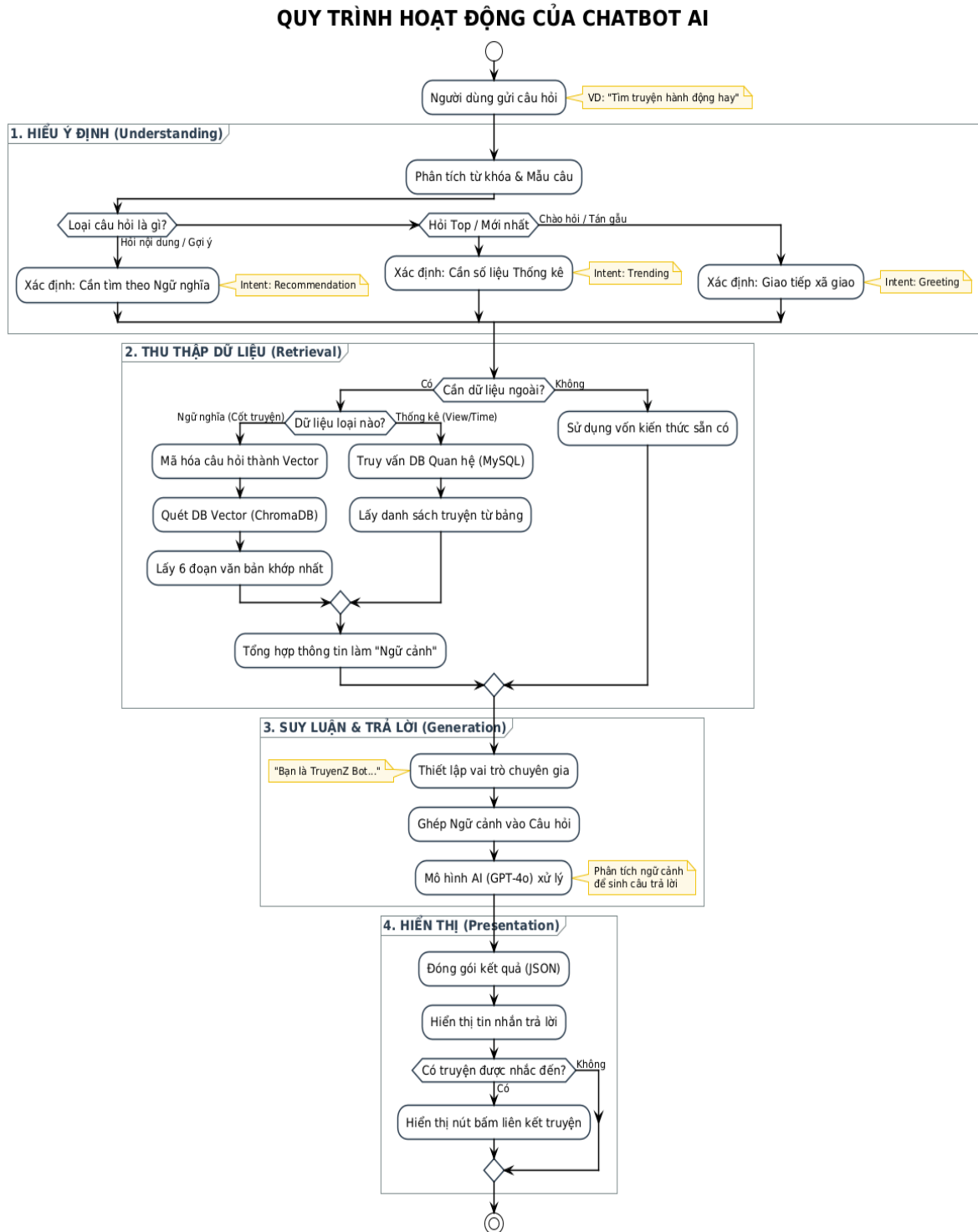
3.2.4.4 Giai đoạn hậu xử lý và phản hồi

Kết quả từ LLM được AI Service định dạng lại thành cấu trúc JSON chuẩn, tách biệt phần văn bản trả lời và danh sách các bộ truyện được gợi ý (nếu có). Gói tin JSON này được trả về cho Client để hiển thị lên giao diện dưới dạng tin nhắn hội thoại và các nút bấm tương tác, kết thúc một chu kỳ xử lý.



Hình 3.4: Sơ Đồ Tuần Tự Hoạt Động "Hỏi Chatbot AI".

3.2.5 Sơ đồ hoạt động hỏi Chatbot



Hình 3.5: Sơ Đồ Hoạt Động Chức Năng "Hỏi Chatbot AI".

[Hình 3.5] mô tả luồng xử lý dữ liệu khép kín bên trong module Chatbot AI, minh họa cách hệ thống chuyển đổi một câu hỏi ngôn ngữ tự nhiên thành câu trả lời chính xác thông qua kỹ thuật Retrieval-Augmented Generation (RAG). Quy trình được chia thành 4 giai đoạn xử lý tuần tự:

3.2.5.1 Phân tích ý định

Đây là bước đầu tiên để định hướng luồng xử lý. Ngay khi nhận được câu hỏi từ người dùng (Query), hệ thống thực hiện chuẩn hóa văn bản (viết thường, loại bỏ khoảng trắng thừa) và áp dụng thuật toán phân loại dựa trên luật (Rule-based) để quét các từ khóa đặc trưng để gán nhãn ý định (Intent Label). Ví dụ: từ khóa "gợi ý" kết hợp "hành động" sẽ kích hoạt intent recommend_action; từ khóa "hot" hoặc "trending" sẽ kích hoạt intent trending. Cơ chế này giúp hệ thống quyết định chính xác nguồn dữ liệu cần truy xuất ở bước sau.

3.2.5.2 Truy xuất dữ liệu

- Truy xuất dữ liệu tĩnh (Vector Search): Đối với các câu hỏi về nội dung, cốt truyện hoặc gợi ý theo ngữ nghĩa, hệ thống tạo vector nhúng (embedding) cho câu truy vấn và tìm kiếm trong ChromaDB. Thuật toán HNSW được sử dụng để trích xuất 6 đoạn văn bản (Top-K=6) có độ tương đồng cao nhất.
- Truy xuất dữ liệu động (Real-time Query): Đối với các ý định cần dữ liệu thời gian thực (như trending, find_new), hệ thống chuyển hướng sang truy vấn trực tiếp cơ sở dữ liệu MySQL để lấy danh sách truyện có lượt xem cao nhất hoặc mới cập nhật.

3.2.5.3 Tạo sinh câu trả lời

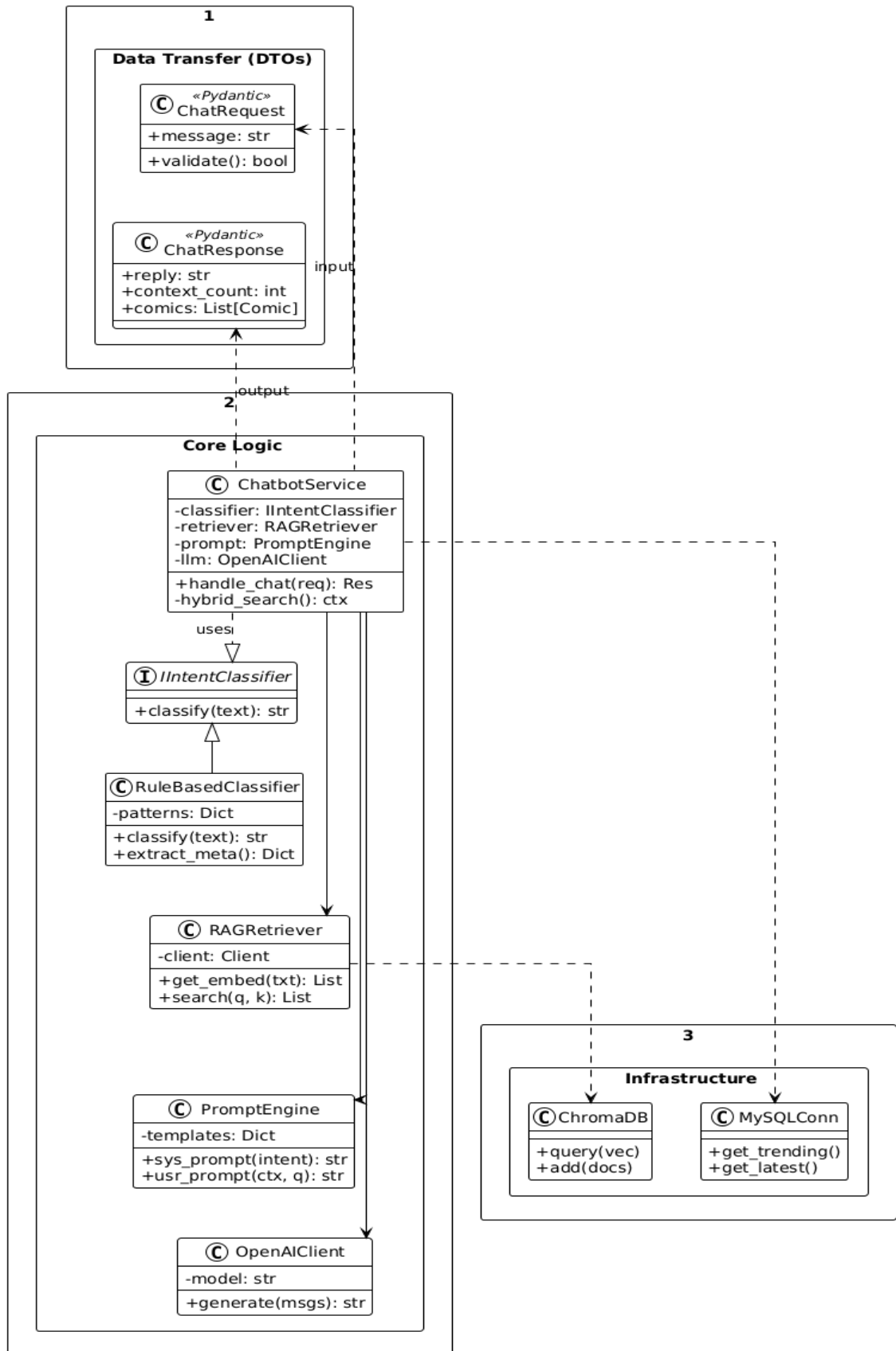
Tại giai đoạn này, kỹ thuật Prompt Engineering được áp dụng. Hệ thống tự động xây dựng "System Prompt" phù hợp với intent và ghép nối ngữ cảnh vừa tìm được vào "User Prompt". Toàn bộ nội dung này được gửi đến mô hình ngôn ngữ lớn GPT-4o-mini để sinh câu trả lời tự nhiên, đảm bảo tính trung thực dựa trên dữ liệu cung cấp.

3.2.5.4 Hậu xử lý

Kết quả sinh từ mô hình ngôn ngữ thường ở dạng văn bản thô, do đó hệ thống cần triển khai bước hậu xử lý để chuẩn hóa nội dung. Các thao tác bao gồm định dạng lại văn bản theo chuẩn Markdown, bổ sung biểu tượng trực quan khi cần thiết và tách riêng các thành phần có cấu trúc.

Sau đó, hệ thống tiếp tục trích xuất các truyện được mô hình đề cập và chuyển đổi chúng sang định dạng JSON để tăng giao diện có thể hiển thị dưới dạng nút bấm hoặc thẻ truyện. Toàn bộ quy trình được tối ưu hóa nhằm duy trì thời gian phản hồi dưới 3 giây, đảm bảo trải nghiệm mượt mà cho người dùng.

3.2.6 Sơ đồ lớp chức năng Chatbot



Hình 3.6: Sơ Đồ Lớp Chức Năng Chatbot AI.

[Hình 3.6] minh họa thiết kế hướng đối tượng (OOP) của module Chatbot, tuân thủ nguyên lý Single Responsibility Principle (SRP) - mỗi lớp chỉ đảm nhận một trách nhiệm duy nhất, giúp mã nguồn dễ bảo trì và mở rộng.

Hệ thống được chia thành 3 tầng chính:

3.2.6.1 Tầng dữ liệu

Sử dụng thư viện Pydantic để định nghĩa các lớp ChatRequest và ChatResponse. Điều này giúp đảm bảo tính toàn vẹn dữ liệu ngay từ đầu vào API (Data Validation) và chuẩn hóa cấu trúc dữ liệu đầu ra JSON.

3.2.6.2 Tầng dịch vụ

- ChatbotService: Đóng vai trò là lớp điều phối trung tâm (Facade Pattern). Nó che giấu sự phức tạp của hệ thống con, nhận yêu cầu từ API và điều phối các thành phần khác để xử lý.
- RuleBasedClassifier: Thực thi giao diện IIntentClassifier, chịu trách nhiệm phân loại ý định người dùng. Việc tách riêng logic này cho phép dễ dàng nâng cấp lên mô hình Machine Learning trong tương lai mà không ảnh hưởng đến code chính.
- RAGRetriever: Đóng gói các thao tác với ChromaDB. Nó chịu trách nhiệm gọi API Embedding và thực hiện truy vấn Vector, giúp tách biệt logic nghiệp vụ khỏi logic truy xuất dữ liệu.
- PromptEngine: Quản lý các khuôn mẫu (templates) của câu lệnh (Prompt). Lớp này đảm bảo tính nhất quán trong cách hệ thống "ra lệnh" cho LLM.

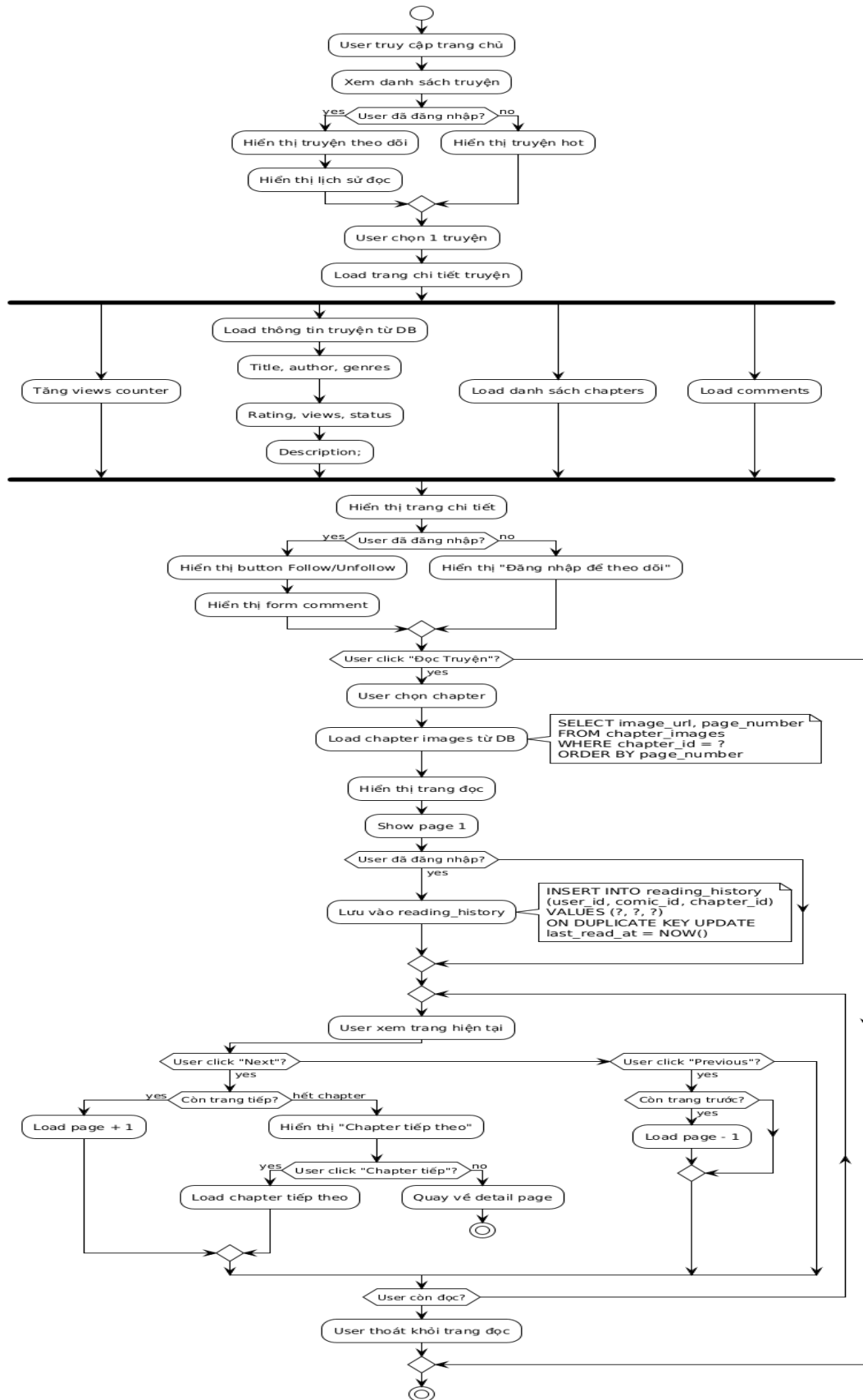
3.2.6.3 Tầng hạ tầng

MySQLIntegration: Quản lý kết nối đến cơ sở dữ liệu quan hệ để phục vụ nhu cầu tra cứu dữ liệu thời gian thực (Real-time data) như danh sách truyện hot, truyện mới - điều mà Vector DB không làm tốt.

Điểm nổi bật trong thiết kế:

- Tính Module hóa (Modularity): Các thành phần như RAGRetriever hay OpenAIClient hoạt động độc lập. Nếu cần thay đổi Vector DB (ví dụ từ ChromaDB sang Pinecone) hay đổi model LLM, chỉ cần sửa đổi lớp tương ứng mà không phá vỡ toàn bộ hệ thống.
- Dependency Injection: ChatbotService nhận các instance của các lớp phụ trợ, giúp code dễ dàng được kiểm thử (Unit Test) và bảo trì.

3.2.7 Sơ đồ hoạt động đọc truyện



Hình 3.7: Sơ Đồ Hoạt Động Đọc Truyện.

[Hình 3.7] mô tả luồng tương tác người dùng (User Flow) chi tiết khi thực hiện tác vụ cốt lõi nhất của hệ thống là đọc truyện tranh. Quy trình được chia thành 3 giai đoạn chính:

3.2.7.1 Tiếp cận và lựa chọn truyện:

Quy trình bắt đầu khi người dùng truy cập vào trang chủ hệ thống. Tại đây, cơ chế phân loại người dùng được kích hoạt nhằm cá nhân hóa nội dung hiển thị. Đối với khách truy cập chưa đăng nhập, giao diện ưu tiên trình bày các danh mục mang tính phổ biến như “Truyện Hot” hoặc “Truyện Mới cập nhật”. Ngược lại, đối với người dùng đã đăng nhập, hệ thống tự động trích xuất dữ liệu cá nhân để hiển thị các danh sách mang tính cá nhân hóa, bao gồm “Truyện đang theo dõi” và “Lịch sử đọc gần đây”. Sau khi quan sát các gợi ý này, người dùng có thể lựa chọn một bộ truyện bất kỳ để chuyển sang trang Chi tiết.

3.2.7.2 Tải dữ liệu chi tiết truyện

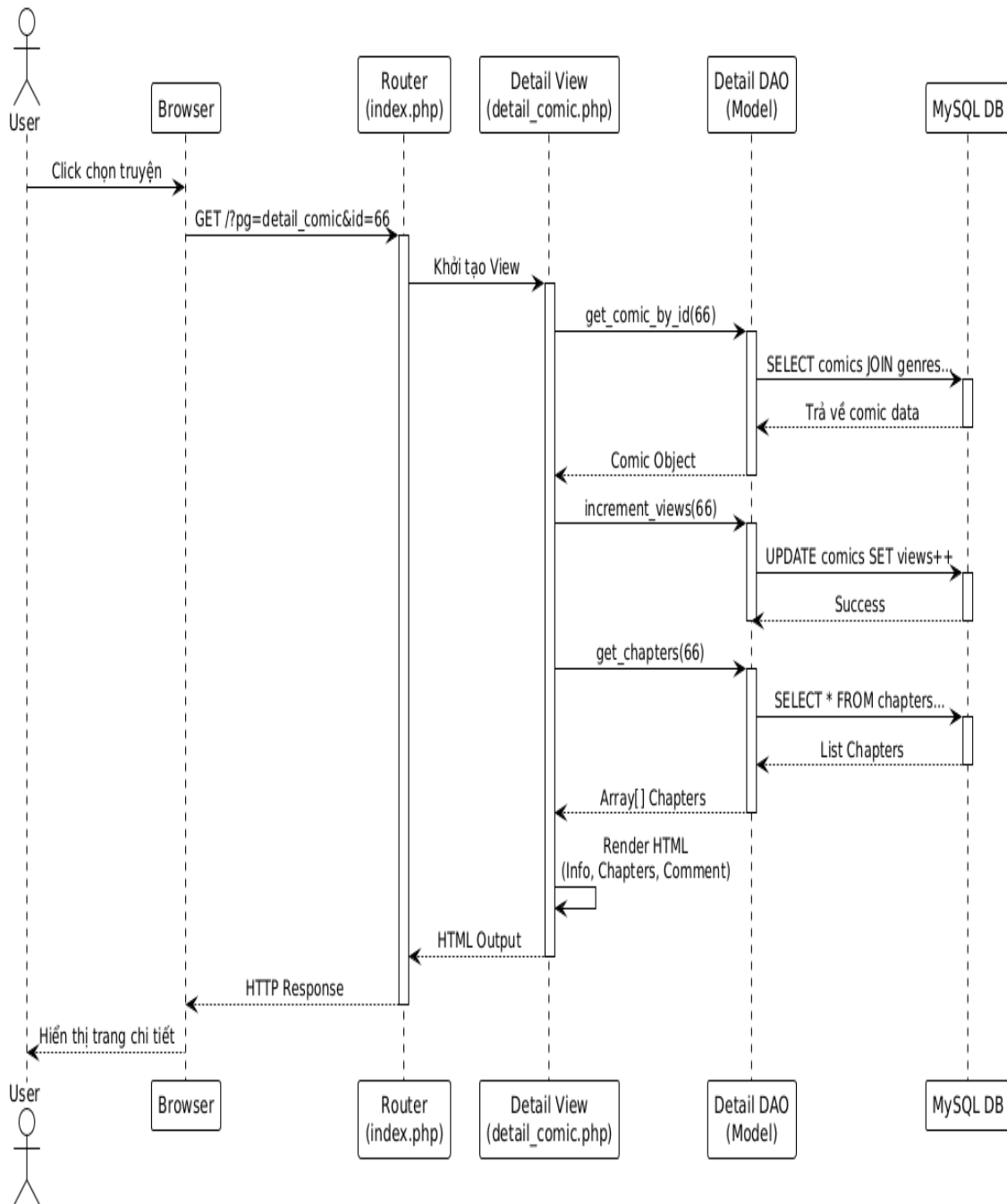
Khi người dùng truy cập trang Chi tiết, hệ thống kích hoạt cơ chế xử lý song song nhằm tối ưu thời gian chờ. Quá trình này bao gồm việc tải thông tin metadata như tác giả, thể loại, tóm tắt nội dung; đồng thời truy vấn danh sách tất cả các chương của truyện. Cùng lúc đó, hệ thống cũng thực hiện một tác vụ nền để tăng lượt xem (increment_views) nhằm ghi nhận mức độ tương tác của người dùng với truyện. Giao diện sau khi dữ liệu được tải sẽ hiển thị đầy đủ thông tin cùng các lựa chọn tương tác như nút “Theo dõi truyện” hoặc “Bình luận”, tùy thuộc vào việc người dùng đã đăng nhập hay chưa.

3.2.7.3 Trải nghiệm đọc truyện

Khi người dùng chọn một Chương (Chapter), hệ thống thực hiện truy vấn bảng chapter_images để tải danh sách toàn bộ hình ảnh chứa nội dung của chương đó. Trong suốt quá trình đọc, hệ thống đồng thời triển khai cơ chế ghi nhận lịch sử, cho phép lưu vị trí đọc hiện tại của người dùng. Nếu người dùng đã đăng nhập, thao tác này được thực thi như một tác vụ nền, đảm bảo không gây gián đoạn trải nghiệm.

Luồng đọc của người dùng diễn ra theo dạng vòng lặp: xem nội dung chương hiện tại, sau đó điều hướng bằng các nút Next/Previous. Hệ thống liên tục kiểm tra điều kiện chuyển chương, bao gồm việc xác định xem người dùng đang ở trang cuối hay chưa. Nếu đã đến cuối chương, giao diện sẽ tự động gợi ý chuyển sang chương kế tiếp hoặc quay về danh sách chương tùy vào ngữ cảnh. Cơ chế này giúp duy trì mạch đọc liền mạch và nâng cao trải nghiệm người dùng.

3.2.8 Sơ đồ tuần tự chức năng đọc truyện



Hình 3.8: Sơ Đồ Tuần Tự Xem Chi Tiết Truyện.

[Hình 3.8] mô tả chi tiết luồng trao đổi thông điệp giữa các đối tượng trong hệ thống khi người dùng yêu cầu xem thông tin của một tác phẩm cụ thể. Quy trình xử lý được chia thành 3 giai đoạn chính:

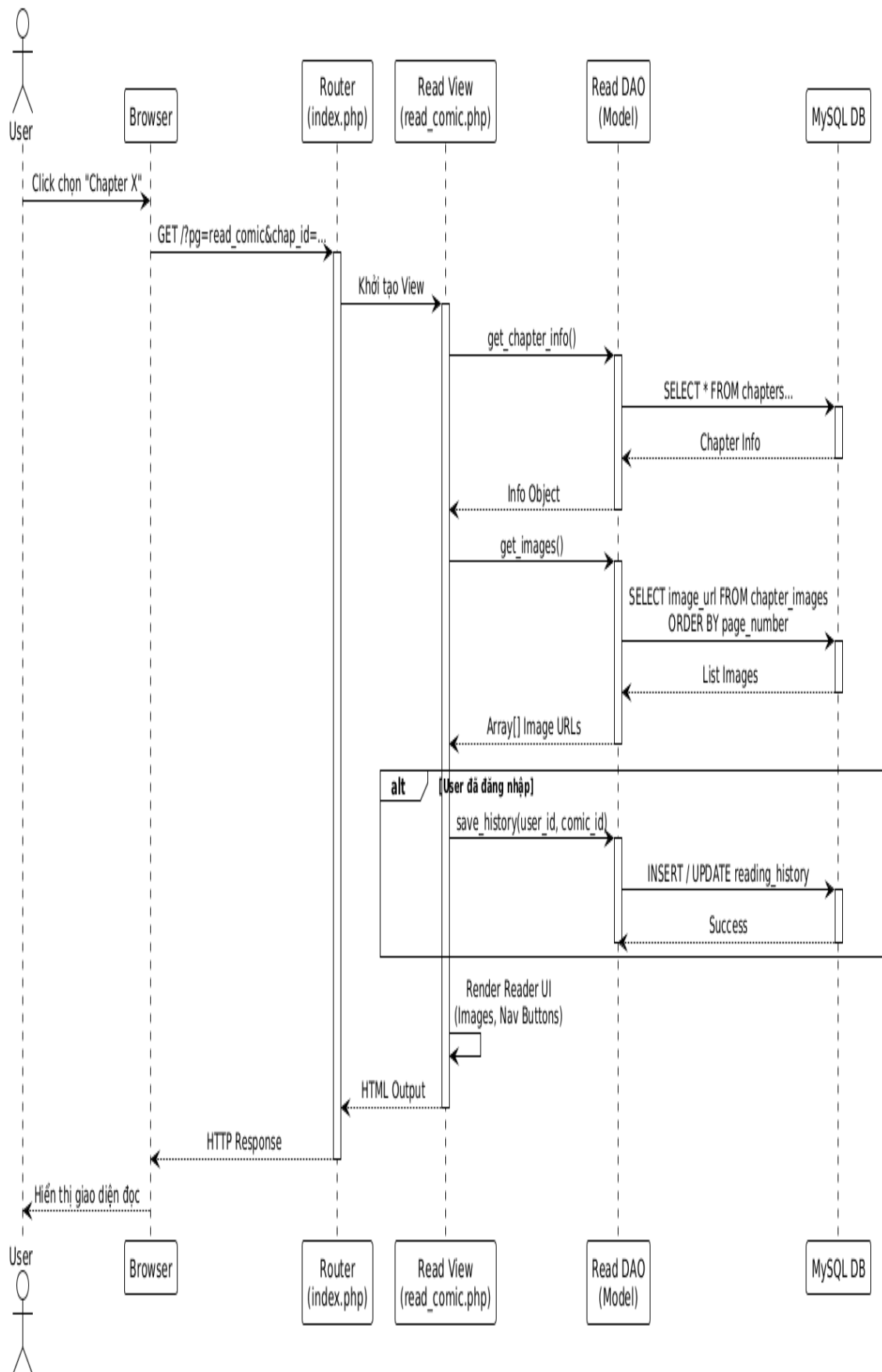
- Giai đoạn Khởi tạo & Truy vấn dữ liệu: Luồng bắt đầu khi Router tiếp nhận yêu cầu HTTP GET từ trình duyệt và khởi tạo đối tượng Detail View. Tại đây, tầng Model (DAO) thực hiện chuỗi truy vấn dữ liệu quan hệ phức tạp: vừa lấy thông

tin metadata (tiêu đề, tác giả, thể loại) từ bảng comics, vừa trích xuất toàn bộ danh sách các chương hiện có từ bảng chapters để chuẩn bị dữ liệu hiển thị.

- Cơ chế Cập nhật Thống kê (Side-effect Processing): Song song với việc lấy dữ liệu hiển thị, hệ thống thực hiện một tác vụ ngầm quan trọng là kích hoạt hàm `increment_views`. Tác vụ này gửi lệnh UPDATE trực tiếp xuống cơ sở dữ liệu để tăng bộ đếm lượt xem cho tác phẩm, đảm bảo tính chính xác của dữ liệu thống kê "Trending" mà không làm gián đoạn trải nghiệm người dùng.
- Giai đoạn Tổng hợp & Hiển thị: Sau khi thu thập đủ dữ liệu từ các nguồn, View tiến hành tổng hợp (Aggregate) các đối tượng Comic và danh sách Chapter vào một giao diện HTML hoàn chỉnh. Giao diện này không chỉ chứa thông tin tĩnh mà còn tích hợp sẵn các thành phần tương tác như khung bình luận và Chatbot Widget trước khi trả về cho trình duyệt người dùng.

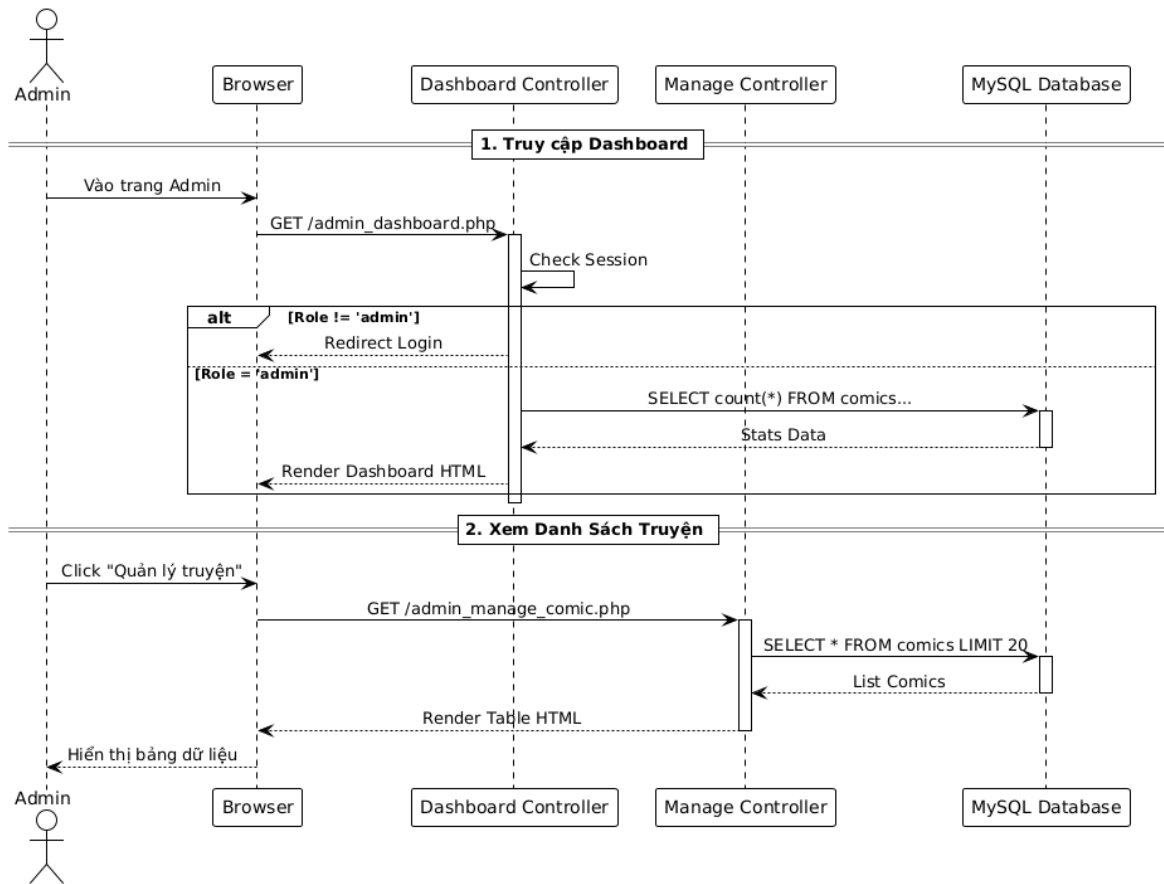
[Hình 3.9] minh họa trình tự các bước xử lý nghiệp vụ khi người dùng bắt đầu đọc nội dung của một chương truyện. Quy trình tập trung vào việc bảo mật nội dung và cá nhân hóa trải nghiệm:

- Giai đoạn Truy xuất Nội dung số: Khi nhận được ID của chương truyện, hệ thống không tải toàn bộ dữ liệu một lúc mà thực hiện truy vấn tối ưu vào bảng `chapter_images`. Kết quả trả về là một danh sách các đường dẫn ảnh (Image URLs) được sắp xếp chính xác theo thứ tự trang (`page_number`), đảm bảo nội dung truyện được hiển thị đúng mạch cốt truyện.
- Cơ chế Cá nhân hóa (Personalization Logic): Điểm nhấn kỹ thuật trong sơ đồ này là khối logic rẽ nhánh (alt fragment) để kiểm tra trạng thái xác thực. Nếu phát hiện người dùng đã đăng nhập (User Logged in), hệ thống sẽ kích hoạt tiến trình ghi nhận lịch sử vào bảng `reading_history`. Cơ chế này hoạt động âm thầm để lưu lại vị trí đọc (`chapter_id`) và thời gian truy cập, phục vụ cho tính năng "Đọc tiếp" sau này.
- Giai đoạn Phản hồi Giao diện: Cuối cùng, hệ thống đóng gói danh sách ảnh và các nút điều hướng (Next/Previous Chapter) vào giao diện Reader UI. Phản hồi được gửi về trình duyệt dưới dạng mã HTML/CSS tối ưu hóa cho việc hiển thị ảnh, cho phép người dùng bắt đầu trải nghiệm đọc ngay lập tức.



Hình 3.9: Sơ Đồ Tuần Tự Đọc Chapter.

3.2.9 Sơ đồ tuần tự Admin quản lý



Hình 3.10: Sơ Đồ Truy Cập Admin Dashboard

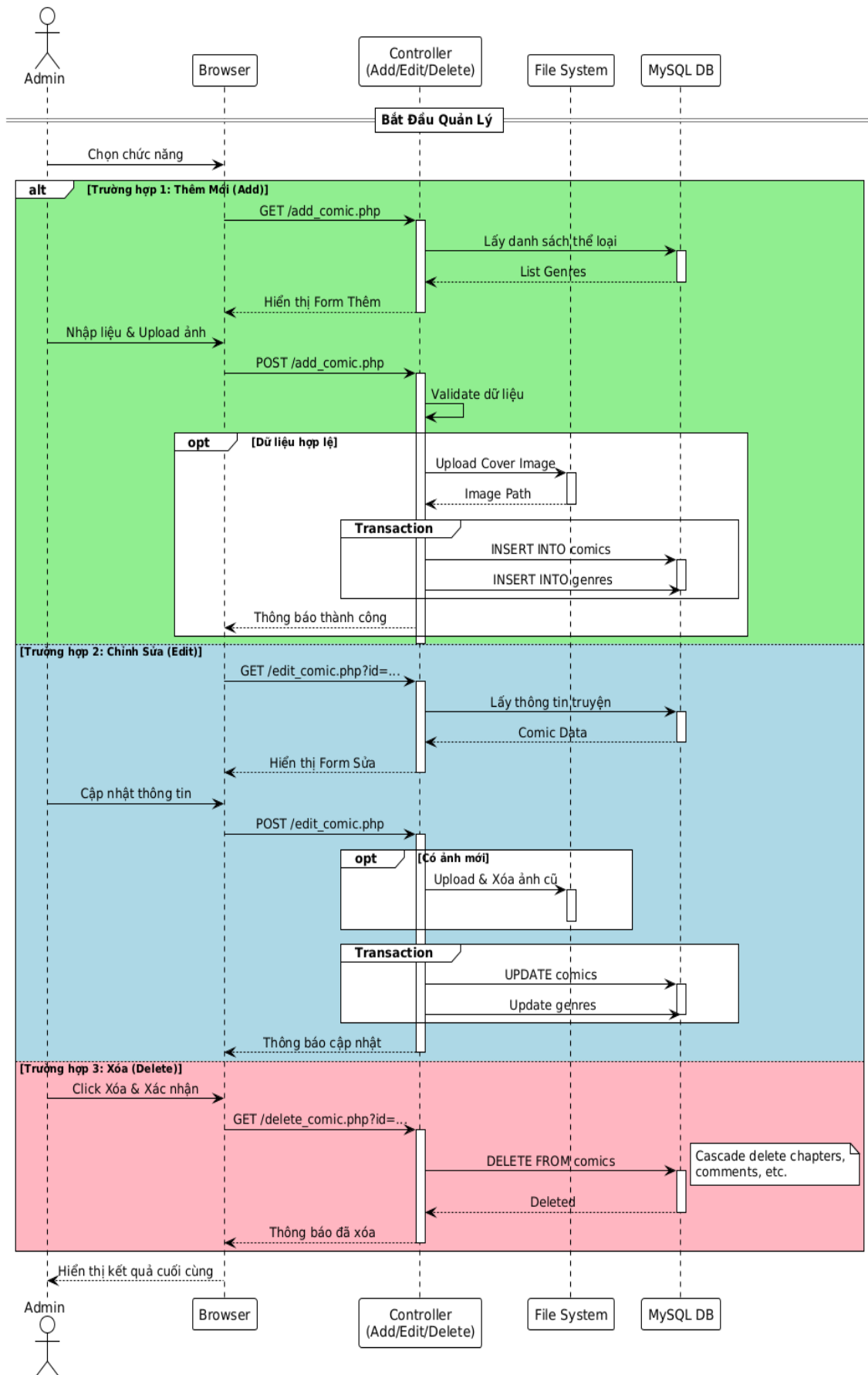
Hình [3.10] minh họa quy trình xử lý khi quản trị viên yêu cầu truy cập vào trang tổng quan (Dashboard). Đây là điểm khởi đầu của phân hệ quản trị, nơi hệ thống thực hiện các bước kiểm tra an ninh và tổng hợp dữ liệu báo cáo:

- **Kiểm soát truy cập:** Quy trình bắt đầu khi quản trị viên gửi yêu cầu truy cập vào Dashboard. Hệ thống ngay lập tức kích hoạt cơ chế kiểm tra phiên làm việc để xác định quyền hạn của người dùng. Dashboard Controller chịu trách nhiệm xác minh xem tài khoản hiện tại có thuộc nhóm quản trị hay không. Nếu phát hiện người dùng không sở hữu quyền Admin, hệ thống sẽ lập tức từ chối truy cập và tự động điều hướng về trang chủ. Cơ chế này đảm bảo rằng các chức năng quản trị được bảo vệ nghiêm ngặt, tránh các tác động trái phép lên dữ liệu và cấu hình hệ thống.
- **Tổng hợp dữ liệu:** Sau khi hoàn tất bước xác thực và xác nhận quyền truy cập hợp lệ, hệ thống chuyển sang giai đoạn tổng hợp dữ liệu để chuẩn bị nội dung cho Dashboard. Chuỗi truy vấn tối ưu được gửi đến cơ sở dữ liệu MySQL nhằm thu thập các chỉ số vận hành quan trọng như tổng số người dùng, tổng lượt xem

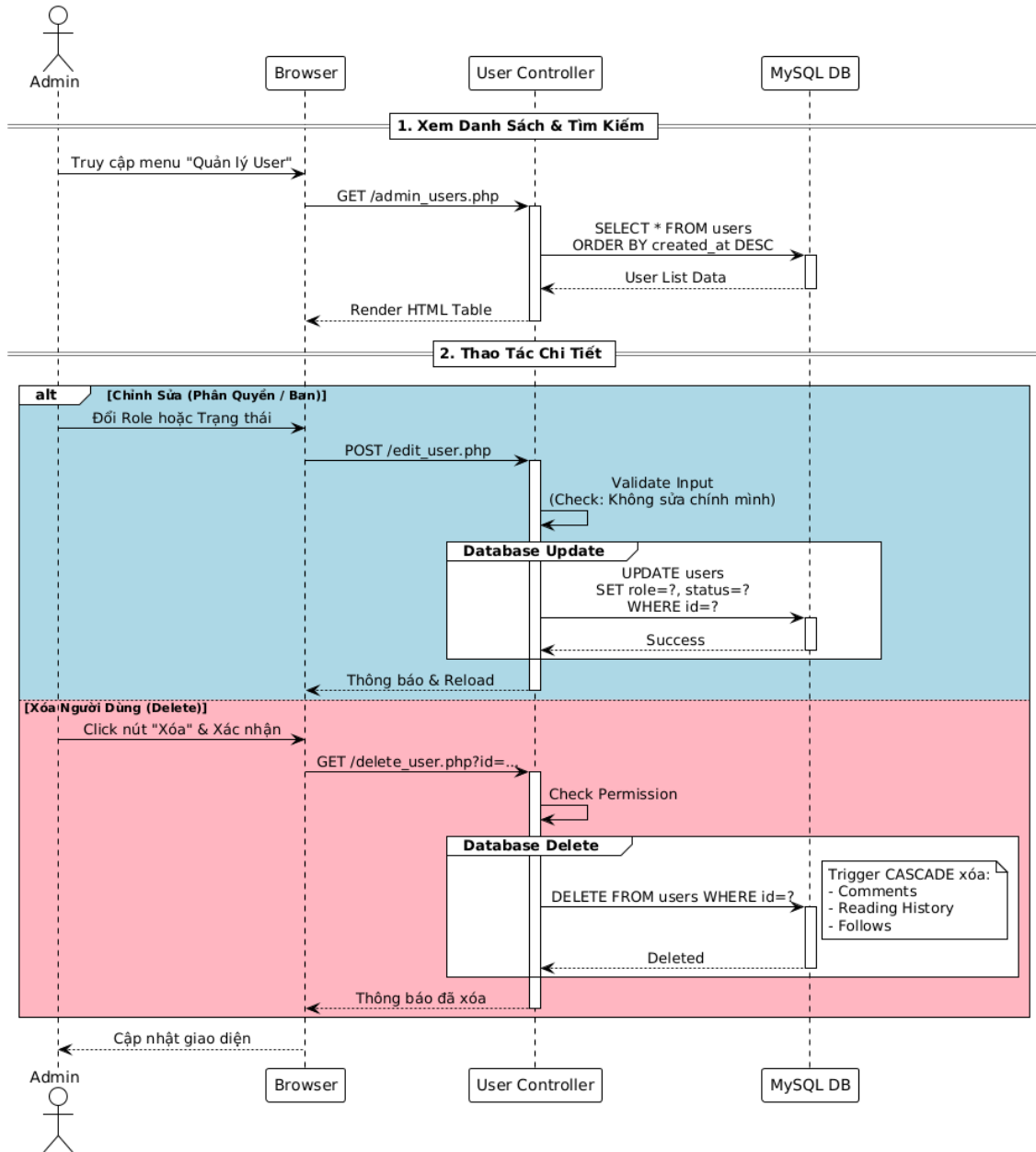
toàn hệ thống và danh sách những bộ truyện đang thịnh hành. Các dữ liệu này được xử lý và đóng gói theo định dạng phù hợp trước khi chuyển đến giao diện Dashboard, giúp quản trị viên có cái nhìn toàn cảnh về hiệu suất hoạt động và tình trạng hiện tại của nền tảng.

[Hình 3.11] minh họa quy trình nghiệp vụ cốt lõi của Quản trị viên (Admin) đối với tài nguyên "Truyện tranh". Quy trình này được thiết kế theo mô hình CRUD (Create, Read, Update, Delete) và chia thành 3 kịch bản xử lý riêng biệt:

- **Kịch bản Thêm mới (Create Scenario):** Quy trình bắt đầu khi quản trị viên yêu cầu truy cập biểu mẫu thêm truyện mới. Hệ thống sẽ truy xuất danh sách thể loại từ cơ sở dữ liệu và hiển thị đầy đủ trên giao diện để Admin lựa chọn. Khi dữ liệu được gửi thông qua yêu cầu POST, hệ thống tiến hành kiểm tra tính hợp lệ nhằm đảm bảo đầy đủ và đúng định dạng. Nếu dữ liệu hợp lệ, hai luồng xử lý được kích hoạt song song: hệ thống lưu ảnh bìa vào File System để tạo đường dẫn tệp tin, đồng thời thực hiện giao dịch cơ sở dữ liệu nhằm chèn thông tin truyện và danh sách thể loại liên quan vào các bảng comics và comic_genres. Toàn bộ bước này đảm bảo dữ liệu được tạo mới đầy đủ và an toàn.
- **Kịch bản Chỉnh sửa (Update Scenario):** Ở kịch bản chỉnh sửa, hệ thống trước tiên tải toàn bộ thông tin hiện tại của bộ truyện lên biểu mẫu giao diện để quản trị viên cập nhật. Một đặc điểm đáng chú ý của luồng này là cơ chế xử lý ảnh bìa. Nếu Admin tải lên một ảnh mới, hệ thống sẽ tự động xóa ảnh cũ khỏi File System nhằm tối ưu dung lượng lưu trữ, sau đó ghi lại đường dẫn ảnh mới vào cơ sở dữ liệu. Các thao tác cập nhật thông tin truyện và thể loại đều được bao gói trong một giao dịch (Transaction) nhằm đảm bảo tính toàn vẹn và đồng bộ dữ liệu khi xảy ra thay đổi.
- **Kịch bản Xóa (Delete Scenario):** Khi quản trị viên xác nhận xóa một bộ truyện, hệ thống thực hiện thao tác DELETE trên bảng comics. Nhờ cấu trúc ràng buộc khóa ngoại được thiết lập với tùy chọn ON DELETE CASCADE, toàn bộ dữ liệu liên quan như danh sách chương, bình luận hoặc lịch sử đọc của người dùng đối với bộ truyện đó sẽ được tự động xóa theo. Cơ chế này giúp hệ thống duy trì tính sạch sẽ của dữ liệu, tránh tình trạng dư thừa hoặc tồn tại các bản ghi mồ côi trong cơ sở dữ liệu.



Hình 3.11: Sơ Đồ Admin quản lý truyện



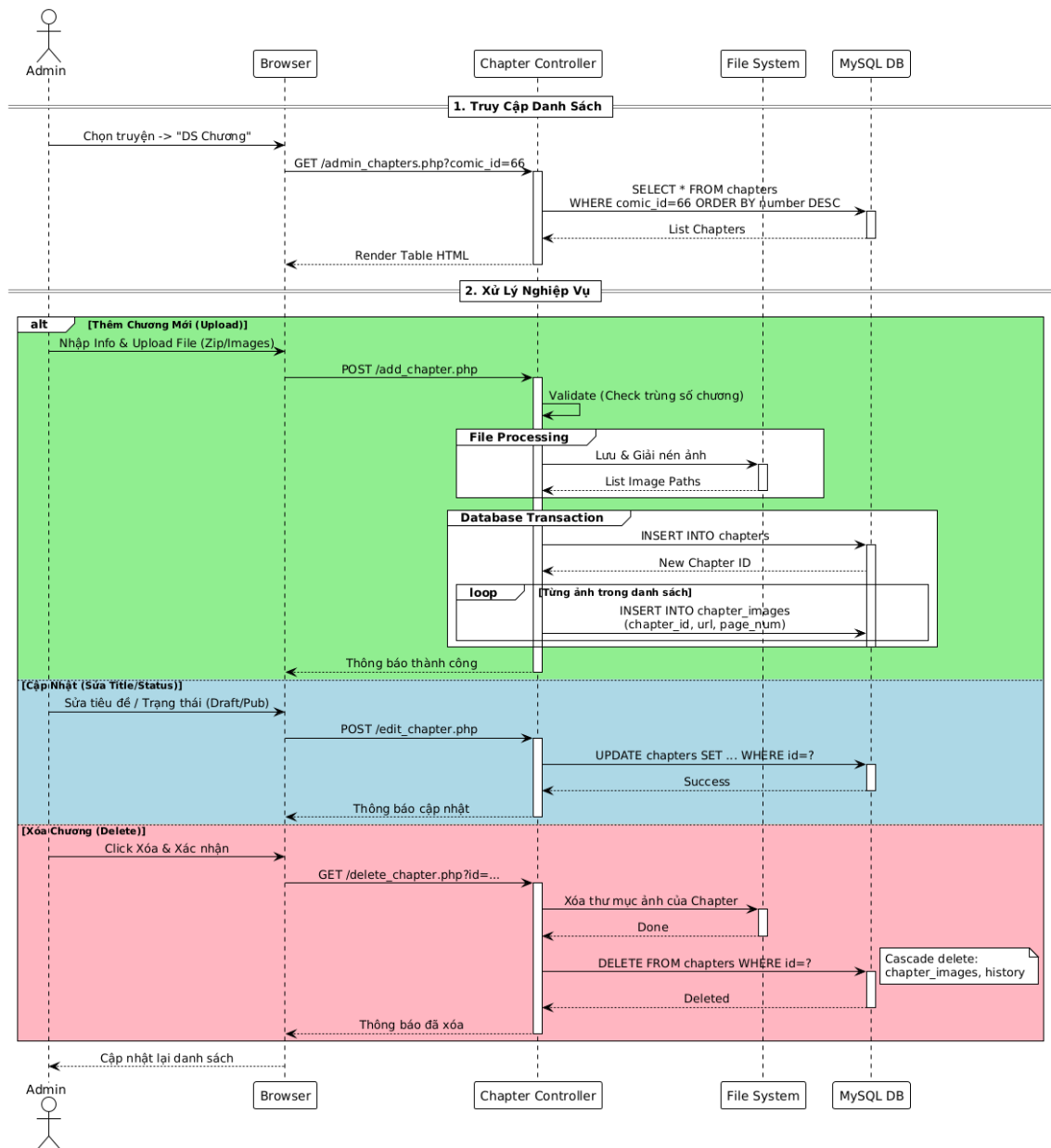
Hình 3.12: Sơ Đồ Admin Quản Lý User.

Hình [3.12]: minh họa quy trình tương tác giữa Quản trị viên và hệ thống để quản lý cơ sở dữ liệu người dùng. Quy trình được chia thành hai giai đoạn nghiệp vụ chính:

- **Giai đoạn Truy xuất (View & Search):** Khi Admin truy cập trang quản lý, hệ thống gửi yêu cầu đến User Controller. Controller thực hiện truy vấn SELECT vào bảng users trong MySQL, có thể kèm theo các điều kiện sắp xếp hoặc tìm kiếm theo username. Kết quả trả về là một bảng danh sách người dùng hiển thị trên trình duyệt.
- **Chỉnh sửa:** Admin có thể thay đổi vai trò (Role) từ User sang Admin hoặc ngược lại, cũng như khóa tài khoản vi phạm. Điểm quan trọng trong logic này là bước

Validate, hệ thống phải kiểm tra để đảm bảo Admin không thể tự khóa hoặc tự tước quyền quản trị của chính mình.

- Xóa người dùng: Khi thực hiện xóa một tài khoản, hệ thống kích hoạt lệnh DELETE. Nhờ thiết kế cơ sở dữ liệu quan hệ với ràng buộc khóa ngoại (Foreign Keys), việc xóa này sẽ kích hoạt cơ chế CASCADE, tự động dọn sạch toàn bộ dữ liệu liên quan của người dùng đó (lịch sử đọc, bình luận, danh sách theo dõi) để đảm bảo tính toàn vẹn dữ liệu và tránh dữ liệu rác.



Hình 3.13: Sơ Đồ Admin Quản Lý Chapter.

Hình [3.13] minh họa quy trình quản lý nội dung chi tiết của một bộ truyện, tập trung vào việc xử lý dữ liệu phân cấp (Hierarchical Data) và tập tin đa phương tiện.

3.2.9.1 Giai đoạn truy xuất – kiểm soát và tổ chức dữ liệu phân cấp

Ở bước đầu tiên, quản trị viên yêu cầu xem danh sách chương thuộc một bộ truyện cụ thể (comic_id). Hệ thống thực hiện truy vấn có điều kiện để chỉ lấy các bản ghi phù hợp, kết hợp sắp xếp theo thứ tự giảm dần số chương (ORDER BY chapter_number DESC). Cách tổ chức này hỗ trợ việc theo dõi các chương mới được cập nhật và đảm bảo khả năng quản lý nội dung ở mức phân cấp (truyện → chương → trang truyện).

3.2.9.2 Giai đoạn thêm mới & upload – xử lý tập tin và đảm bảo toàn vẹn dữ liệu

Đây là giai đoạn có độ phức tạp kỹ thuật cao nhất. Khi quản trị viên thêm chương mới, hệ thống phải xử lý đồng thời dữ liệu mô tả và tập tin ảnh:

- Xử lý tập tin đa phương tiện: Bộ điều khiển (Controller) ghi nhận tập tin tải lên (ảnh lẻ hoặc gói ZIP). Nếu là ZIP, hệ thống tiến hành giải nén, đổi tên và chuẩn hóa cấu trúc thư mục nhằm tránh trùng tên và đảm bảo thống nhất về định dạng lưu trữ.
- Xử lý dữ liệu chương: Sau khi hoàn tất việc ghi file vật lý, hệ thống mở một Transaction gồm hai bước tạo bản ghi mới trong bảng chapters, thực hiện vòng lặp chèn dữ liệu hàng loạt vào bảng chapter_images tương ứng với từng trang truyện. Transaction đảm bảo toàn vẹn dữ liệu, tránh trạng thái nửa vời khi xảy ra lỗi trong quá trình upload.
- Giai đoạn Xóa – Đồng bộ giữa dữ liệu và hệ thống tệp: Quy trình xóa đòi hỏi sự đồng bộ giữa Dữ liệu và File. Hệ thống thực hiện xóa các file ảnh vật lý trên ổ cứng (File Cleanup) để giải phóng dung lượng server, sau đó thực hiện lệnh DELETE trong cơ sở dữ liệu. Ràng buộc ON DELETE CASCADE trong MySQL sẽ tự động xóa sạch các dòng liên kết trong bảng chapter_images và reading_history.

3.3 Thiết kế cơ sở dữ liệu

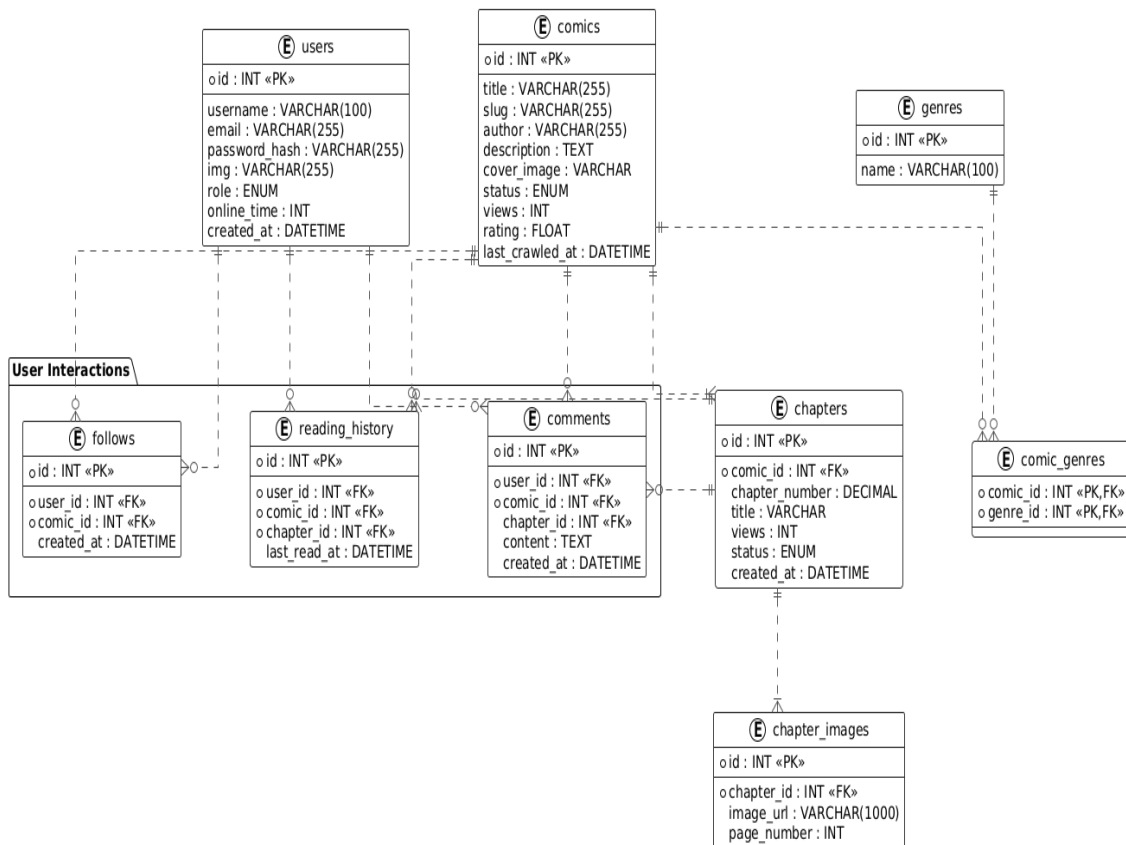
Để đáp ứng các yêu cầu đa dạng về truy xuất dữ liệu – từ các giao dịch chặt chẽ của người dùng đến các truy vấn tìm kiếm ngữ nghĩa phức tạp – hệ thống áp dụng chiến lược Lưu trữ lai (Hybrid Persistence). Chiến lược này kết hợp sức mạnh của Hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) để quản lý dữ liệu có cấu trúc và Cơ sở dữ liệu Vector chuyên dụng để xử lý dữ liệu phi cấu trúc.

3.3.1 Cơ sở dữ liệu quan hệ

Hệ quản trị cơ sở dữ liệu MySQL (MariaDB) được sử dụng để lưu trữ toàn bộ dữ liệu nghiệp vụ cốt lõi. Cấu trúc dữ liệu được thiết kế tuân thủ Dạng chuẩn 3 (3NF) nhằm giảm thiểu dư thừa và đảm bảo tính toàn vẹn dữ liệu.

3.3.1.1 Sơ đồ thực thể liên kết

Sơ đồ dưới đây minh họa các thực thể chính và mối quan hệ giữa chúng trong hệ thống:



Hình 3.14: Sơ Đồ Thực Thể Liên Kết.

[Hình 3.14] mô tả cấu trúc dữ liệu quan hệ (Relational Schema) của hệ thống, được thiết kế tuân thủ Dạng chuẩn 3 (3NF) nhằm giảm thiểu dư thừa dữ liệu và đảm bảo tính nhất quán. Cơ sở dữ liệu bao gồm 9 thực thể chính, được chia thành 3 nhóm nghiệp vụ:

3.3.1.2 Nhóm quản lý nội dung

Đây là nhóm bảng trung tâm lưu trữ dữ liệu cốt lõi của hệ thống:

- comics (Truyện tranh): Là thực thể chính, lưu trữ metadata của truyện như tiêu đề, tác giả, ảnh bìa. Trường slug được đánh index (UNIQUE INDEX) để

phục vụ việc tạo URL thân thiện (SEO) và tối ưu hóa tốc độ truy vấn theo đường dẫn. Các trường thống kê như views, rating được lưu trực tiếp tại đây để giảm tải tính toán (denormalization) khi hiển thị danh sách Trending.

- **chapters** (Chương truyện): Quan hệ 1-N với bảng comics (Một truyện có nhiều chương). Bảng này lưu trữ số thứ tự chương và trạng thái (status: Draft/Published) để phục vụ quy trình kiểm duyệt nội dung.
- **chapter_images** (Hình ảnh chương): Quan hệ 1-N với bảng chapters. Đây là đơn vị lưu trữ nhỏ nhất, chứa đường dẫn (image_url) tới file ảnh thực tế và số thứ tự trang (page_number), đảm bảo hiển thị đúng trình tự đọc.
- **genres & comic_genres**: Hệ thống giải quyết bài toán "Một truyện thuộc nhiều thể loại" thông qua bảng trung gian comic_genres, thiết lập quan hệ N-N giữa comics và genres.

3.3.1.3 Nhóm người dùng và định danh

- **users**: Lưu trữ thông tin định danh. Hệ thống hỗ trợ đăng nhập qua Google OAuth, do đó trường google_id được sử dụng để liên kết tài khoản, trong khi password_hash dùng cho tài khoản thường. Trường role phân quyền rõ ràng giữa admin (quản trị viên) và user (người đọc).

3.3.1.4 Nhóm tương tác và cá nhân hóa

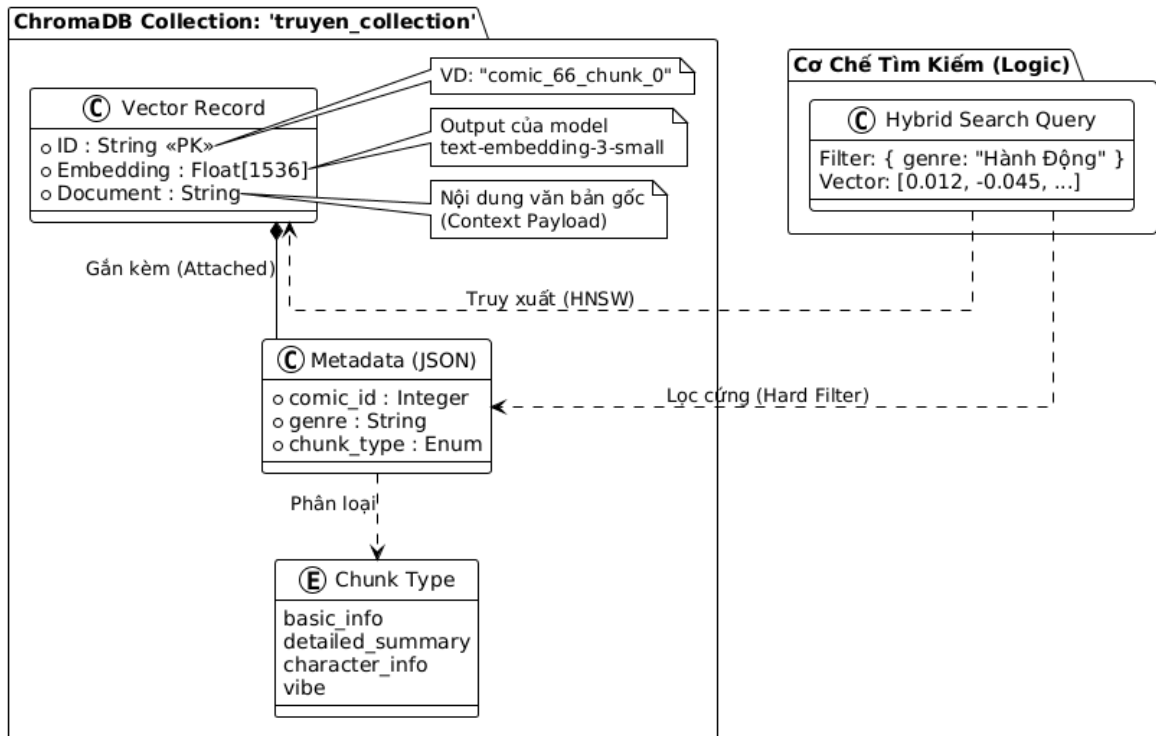
Nhóm bảng này lưu trữ hành vi người dùng để phục vụ cho các thuật toán gợi ý của Chatbot AI:

- **reading_history** (Lịch sử đọc): Lưu vết tiến độ đọc của người dùng. Khóa duy nhất (Unique Key) được thiết lập trên cặp (user_id, comic_id) để đảm bảo mỗi người dùng chỉ có một dòng trạng thái mới nhất cho mỗi bộ truyện, tránh spam dữ liệu.
- **follows** (Theo dõi): Lưu trữ danh sách truyện yêu thích, cho phép hệ thống gửi thông báo hoặc gợi ý các truyện tương tự.
- **comments** (Bình luận): Cho phép người dùng thảo luận ở cấp độ truyện hoặc cấp độ chương.

Cơ chế ràng buộc toàn vẹn (Constraints): Hệ thống sử dụng triệt để các khóa ngoại (Foreign Keys) với cơ chế ON DELETE CASCADE. Điều này đảm bảo tính toàn vẹn dữ liệu: ví dụ, khi một bộ truyện (comics) bị xóa, toàn bộ các chương (chapters), hình ảnh (chapter_images) và lịch sử đọc liên quan sẽ tự động được xóa theo, ngăn chặn tình trạng dữ liệu rác (orphaned records).

3.3.2 Cơ sở dữ liệu Vector

ChromaDB được sử dụng làm bộ nhớ dài hạn (Long-term Memory) cho Module AI, lưu trữ các biểu diễn ngữ nghĩa của nội dung truyện. Khác với MySQL lưu trữ theo dòng/cột, ChromaDB tổ chức dữ liệu thành các Collection.



Hình 3.15: Sơ Đồ Vector Schema.

3.3.2.1 Cấu trúc Collection

Hệ thống duy trì một collection chính tên là `truyen_collection` với cấu trúc bản ghi như sau:

- **ID**: Mã định danh duy nhất cho mỗi đoạn văn bản (ví dụ: `comic_66_chunk_0`).
- **Embedding (Vector)**: Mảng số thực kích thước 1536 chiều, được sinh ra bởi mô hình `text-embedding-3-small`. Đây là biểu diễn toán học của nội dung văn bản.
- **Document (Payload)**: Nội dung văn bản gốc (đã qua làm sạch), dùng để cung cấp ngữ cảnh cho LLM sau khi tìm kiếm thành công.

3.3.2.2 Thiết kế Metadata

Để tối ưu hóa hiệu năng của chiến lược Tìm kiếm lai (Hybrid Search), kiến trúc dữ liệu trong ChromaDB không chỉ dừng lại ở việc lưu trữ các vector đặc trưng mà còn được gắn kết chặt chẽ với hệ thống siêu dữ liệu (Metadata) phong phú. Thiết kế này đóng vai trò then chốt trong việc thu hẹp không gian tìm kiếm, cho phép hệ thống thực

hiện các phép lọc chính xác (Hard Filtering) dựa trên thuộc tính trước khi tiến hành các phép so khớp tương đồng vector (Soft Matching) tốn kém tài nguyên.

Cấu trúc Metadata được chuẩn hóa bao gồm các nhóm thông tin cốt lõi sau:

- Định danh và Liên kết dữ liệu: Trường comic_id (int) đóng vai trò là khóa ngoại (foreign key) tham chiếu ngược lại cơ sở dữ liệu quan hệ MySQL. Điều này đảm bảo tính nhất quán khi hệ thống cần truy xuất các thông tin chi tiết hiển thị (như ảnh bìa, đường dẫn đọc) từ kết quả tìm kiếm vector.
- Phân loại và Điều hướng: Trường genre (string) lưu trữ danh sách thể loại của truyện, phục vụ trực tiếp cho việc lọc theo ý định người dùng. Đặc biệt, hệ thống sử dụng trường chunk_type để phân loại ngữ nghĩa của từng đoạn văn bản (basic_info, detailed_summary, character_info, vibe). Việc gán nhãn này giúp thuật toán tìm kiếm định vị chính xác loại thông tin cần truy xuất, tránh trả về các đoạn văn bản không liên quan đến ngữ cảnh (ví dụ: trả về mô tả nhân vật khi người dùng đang hỏi về cốt truyện).

3.4 Thiết kế Giải thuật và luồng xử lý dữ liệu

Để đảm bảo hệ thống hoạt động chính xác và hiệu quả, việc thiết kế giải thuật xử lý dữ liệu đầu vào và chiến lược tìm kiếm thông tin là yếu tố then chốt. Đồ án áp dụng quy trình ETL (Extract - Transform - Load) tiêu chuẩn kết hợp với chiến lược tìm kiếm lai (Hybrid Search) để tối ưu hóa khả năng truy xuất thông tin của Chatbot.

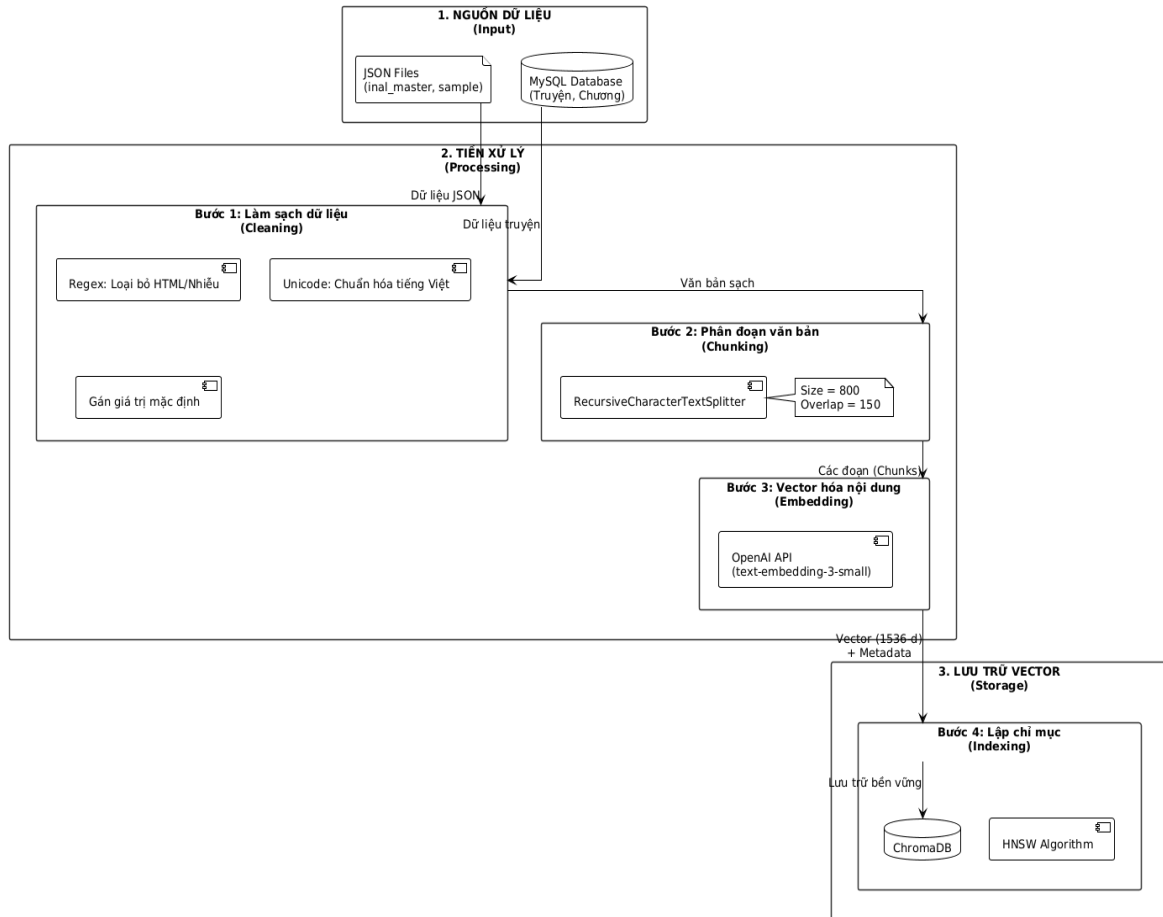
3.4.1 Quy trình xử lý dữ liệu

Quy trình chuyển đổi dữ liệu thô thành vector ngữ nghĩa được triển khai theo một chuỗi xử lý khép kín gồm bốn giai đoạn. Ở giai đoạn đầu tiên — Thu thập và Làm sạch — hệ thống lấy dữ liệu từ các tệp JSON và cơ sở dữ liệu MySQL, sau đó tiến hành loại bỏ ký tự nhiễu, thẻ HTML dư thừa và chuẩn hóa Unicode bằng Regex. Các bản ghi thiếu dữ liệu quan trọng cũng được xử lý bằng cách bổ sung giá trị mặc định hoặc loại bỏ để đảm bảo chất lượng dữ liệu đầu vào.

Tiếp theo, giai đoạn Phân đoạn nội dung nhằm chia nhỏ các văn bản dài để phù hợp với giới hạn cửa sổ ngữ cảnh và thuật toán tìm kiếm vector. Hệ thống áp dụng kỹ thuật Recursive Chunking với độ dài tối ưu 800 ký tự và độ chồng lấp 150 ký tự, đồng thời ưu tiên cắt tại các dấu câu để duy trì tính liên mạch về ngữ nghĩa.

Ở giai đoạn Vector hóa, các đoạn văn bản đã được chia nhỏ được đưa vào mô hình text-embedding-3-small. Mỗi đoạn văn bản được chuyển đổi thành một vector 1536 chiều, biểu diễn nội dung và ý nghĩa ngữ nghĩa của đoạn văn.

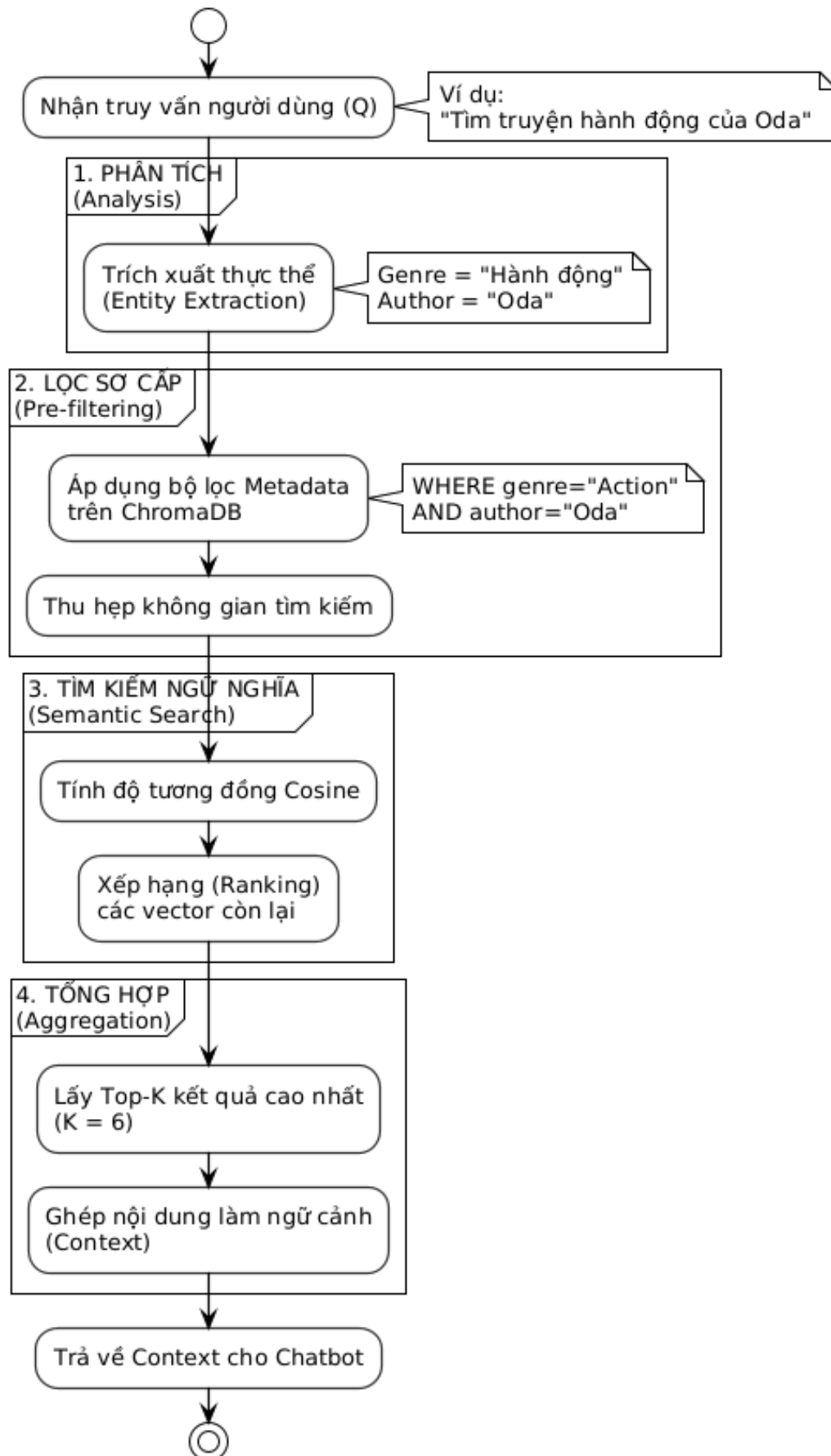
Cuối cùng, giai đoạn Lập chỉ mục đưa các vector cùng metadata như comic_id hoặc genre vào lưu trữ trong ChromaDB. Hệ thống sử dụng thuật toán HNSW để xây dựng cấu trúc chỉ mục dạng đồ thị, giúp thực hiện truy vấn lân cận gần đúng với tốc độ cao và độ trễ thấp, đảm bảo truy vấn thời gian thực cho chatbot.



Hình 3.15: Sơ Đồ Quy Trình Xử Lý Dữ Liệu.

3.4.2 Chiến lược tìm kiếm lại

Hệ thống áp dụng mô hình Hybrid Search nhằm kết hợp ưu điểm của hai kỹ thuật tìm kiếm chủ đạo: (1) tìm kiếm ngữ nghĩa dựa trên vector (Semantic Search) và (2) tìm kiếm từ khóa kết hợp lọc thuộc tính (Keyword/Metadata Filtering). Cơ chế này cho phép tăng độ chính xác của kết quả đồng thời duy trì hiệu năng xử lý ổn định trên tập dữ liệu lớn.



Hình 3.16: Sơ Đồ Mô Tả Tìm Kiếm Lại.

- **Phân tích truy vấn (Query Analysis):** Khi nhận yêu cầu từ người dùng, hệ thống thực hiện phân tách câu truy vấn để nhận diện các thực thể và thuộc tính liên quan, chẳng hạn như thể loại (Genre), tác giả (Author) hoặc từ khóa chủ đề. Ví dụ, với truy vấn “Tìm truyện hành động hay của tác giả Oda”, hệ thống trích xuất được hai tham số quan trọng: *Genre = Hành động* và *Author = Oda*.
- **Lọc sơ cấp (Pre-filtering):** Trước khi bước vào giai đoạn tính toán vector tốn chi phí xử lý, hệ thống thực thi bộ lọc Metadata trên cơ sở dữ liệu Vector nhằm thu hẹp không gian tìm kiếm. Chỉ những bản ghi thỏa các điều kiện về thể loại và tác giả mới được đưa vào tập ứng viên. Cách tiếp cận này giúp loại bỏ nhiều, đồng thời cải thiện tốc độ truy vấn đáng kể.
- **Tìm kiếm ngữ nghĩa (Semantic Ranking):** Trong phạm vi dữ liệu đã được tối ưu, hệ thống tiến hành tính toán độ tương đồng (Cosine Similarity) giữa vector biểu diễn truy vấn và các vector nội dung tương ứng. Các kết quả trả về được xếp hạng theo mức độ tương đồng giảm dần nhằm bảo đảm truy xuất đúng những mục liên quan nhất đến ngữ cảnh câu hỏi.
- **Tổng hợp kết quả (Result Aggregation):** Cuối cùng, hệ thống lựa chọn *Top-K* ($K = 6$) bản ghi có điểm số cao nhất để cung cấp ngữ cảnh đầu vào cho Chatbot. Cách tổ chức này giúp cân bằng giữa hai tiêu chí quan trọng: độ chính xác về mặt ngữ nghĩa và sự tuân thủ nghiêm ngặt các ràng buộc dữ liệu (đúng thể loại, đúng tác giả). Nhờ đó, chất lượng gợi ý và phản hồi từ mô-đun AI được cải thiện một cách toàn diện.

CHƯƠNG 4: XÂY DỰNG WEBSITE ĐỌC TRUYỆN TÍCH HỢP CHATBOT AI GỢI Ý VÀ TÓM TẮT NỘI DUNG

4.1 Môi trường phát triển và triển khai

Việc thiết lập và cấu hình môi trường phát triển là bước khởi đầu then chốt, quyết định trực tiếp đến hiệu năng kiểm thử và khả năng vận hành ổn định của toàn bộ hệ thống. Trong phạm vi đề án, hệ thống được xây dựng, huấn luyện và triển khai thử nghiệm trên một máy trạm cá nhân hiệu năng cao, mô phỏng sát sao điều kiện vận hành của một máy chủ ứng dụng thực tế nhằm đánh giá chính xác độ trễ và khả năng chịu tải.. Các thành phần chính của môi trường phát triển trong dự án này được trình bày như dưới đây.

4.1.1 Cấu hình phần cứng và hạ tầng triển khai

Hệ thống được phát triển trên thiết bị ASUS ROG Strix G513IH, một dòng máy trạm chuyên dụng với kiến trúc phần cứng tối ưu cho các tác vụ xử lý đa nhiệm. Bộ não của hệ thống là vi xử lý AMD Ryzen™ 7 4800H với 8 nhân vật lý và 16 luồng xử lý, hoạt động ở xung nhịp cơ bản 2.9GHz và có thể tăng tốc lên 4.2GHz. Sức mạnh đa luồng vượt trội của kiến trúc Zen 2 giúp máy trạm dễ dàng vận hành song song nhiều dịch vụ nền tảng như Web Server (Apache) và Database Server (MySQL) mà không gây ra hiện tượng nghẽn cổ chai CPU. Bên cạnh đó, hệ thống được trang bị bộ nhớ trong 16GB DDR4 3200MHz chạy ở chế độ kênh đôi (Dual-channel), cho phép phân bổ tài nguyên linh hoạt: khoảng 6GB được cấp phát riêng cho tiến trình Python để tải các thư viện AI nặng và bộ nhớ đệm của ChromaDB, trong khi phần còn lại đảm bảo sự mượt mà cho hệ điều hành và các công cụ phát triển.

Về khả năng xử lý đồ họa và lưu trữ, máy trạm tích hợp card đồ họa rời NVIDIA GeForce GTX 1650 với 4GB VRAM GDDR6. Mặc dù các mô hình ngôn ngữ lớn được gọi thông qua API đám mây, GPU này đóng vai trò quan trọng trong việc tăng tốc phần cứng cho các tác vụ tiền xử lý hình ảnh bìa truyện và render giao diện người dùng độ phân giải cao. Hệ thống lưu trữ sử dụng ổ cứng SSD 512GB M.2 NVMe PCIe 3.0, cung cấp tốc độ đọc/ghi dữ liệu vượt trội. Yếu tố này đặc biệt quan trọng đối với cơ sở dữ liệu vector ChromaDB, vốn hoạt động dựa trên cơ chế đọc/ghi tệp tin (File I/O), giúp giảm thiểu độ trễ truy xuất vector xuống mức thấp nhất và tăng tốc độ phản hồi của Chatbot.

4.1.2 Nền tảng phần mềm và công cụ phát triển

Hệ thống hoạt động trên nền tảng hệ điều hành Windows 11 Home Single Language (64-bit), phiên bản 23H2, cung cấp môi trường ổn định và tương thích tốt với các công cụ phát triển hiện đại. Quy trình viết mã và gỡ lỗi được thực hiện chủ yếu trên Cursor IDE, một môi trường tích hợp trí tuệ nhân tạo giúp tự động hóa việc tái cấu trúc mã nguồn (Refactoring) và phát hiện lỗi logic trong thời gian thực. Bên cạnh đó, VS Code được sử dụng như một công cụ phụ trợ để quản lý các tập tin cấu hình nhẹ. Để đảm bảo tính toàn vẹn và khả năng truy vết của mã nguồn, đồ án áp dụng quy trình quản lý phiên bản phân tán thông qua Git, kết hợp với kho lưu trữ đám mây GitHub. Việc kiểm thử các giao diện lập trình ứng dụng (API) của module Chatbot được thực hiện độc lập bằng công cụ Postman, cho phép xác minh tính chính xác của các luồng dữ liệu JSON trước khi tích hợp hoàn chỉnh vào giao diện người dùng.

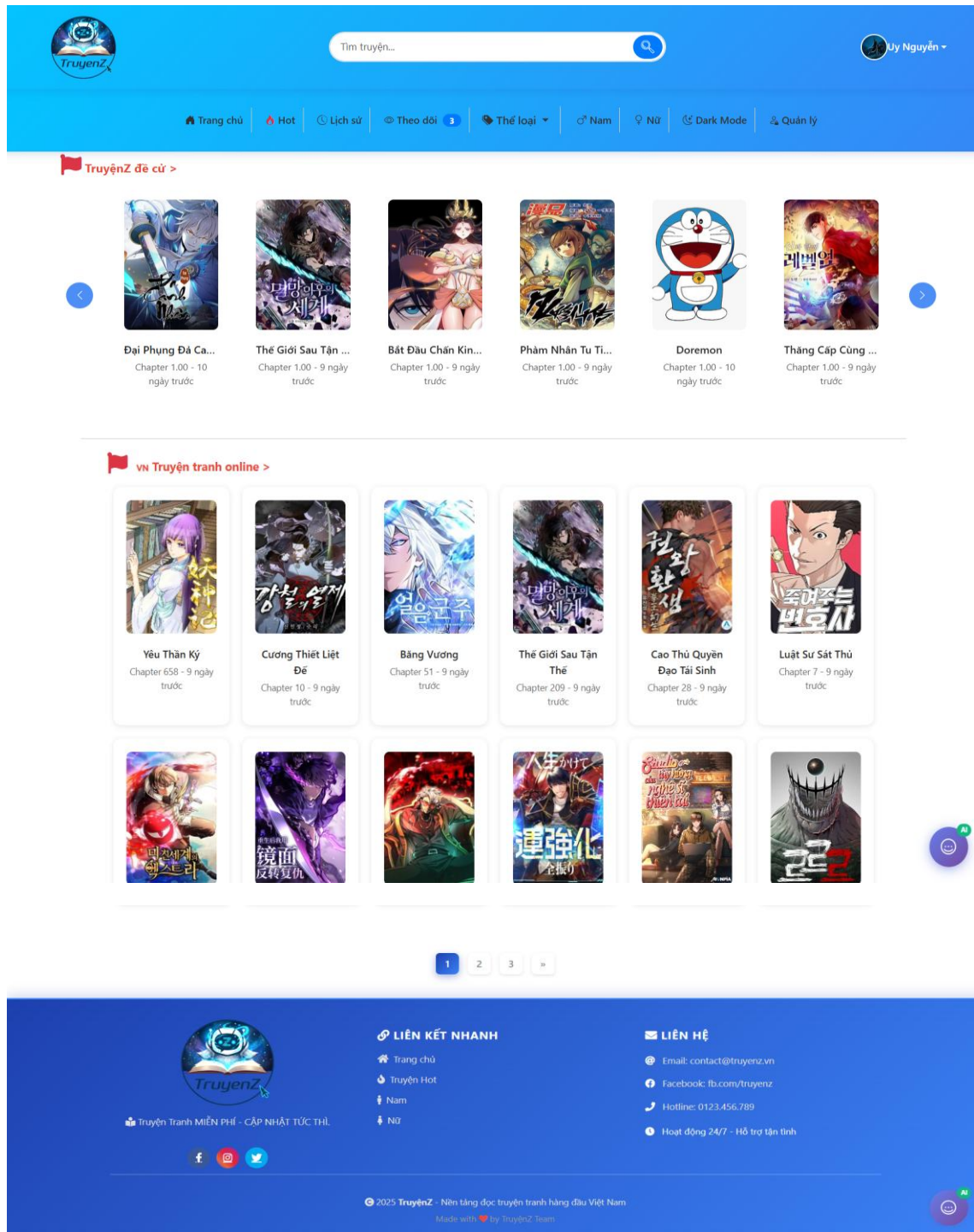
4.1.3 Môi trường thực thi và thư viện phụ thuộc

Kiến trúc hệ thống được thiết kế theo mô hình đa ngôn ngữ (Polyglot Architecture), yêu cầu hai môi trường thực thi hoạt động song song và biệt lập. Phần hệ Website được vận hành trên bộ công cụ XAMPP v3.3.0, bao gồm máy chủ web Apache 2.4 đóng vai trò cổng giao tiếp chính (Gateway), hệ quản trị cơ sở dữ liệu MariaDB 10.4.32 lưu trữ dữ liệu quan hệ, và ngôn ngữ PHP 8.2.12. Phiên bản PHP 8.2 với trình biên dịch JIT (Just-In-Time) mang lại hiệu năng xử lý vượt trội cho các tác vụ logic nghiệp vụ. Các thư viện PHP quan trọng như google/apiclient (xác thực OAuth 2.0) và guzzlehttp/guzzle (HTTP Client) được quản lý chặt chẽ thông qua trình quản lý gói Composer.

Đối với phân hệ AI Service, một môi trường ảo hóa được thiết lập với ngôn ngữ Python 3.10.11 để cô lập các phụ thuộc, tránh xung đột với hệ thống. Ứng dụng được khởi chạy trên máy chủ ASGI Uvicorn, kết hợp với framework FastAPI 0.109.0 để cung cấp các API hiệu năng cao. Các thư viện AI cốt lõi được cài đặt theo phiên bản chính xác trong tệp requirements.txt, bao gồm openai 1.54.5 để kết nối với các mô hình GPT, chromadb 0.4.24 để quản lý cơ sở dữ liệu vector với thuật toán HNSW, và langchain-text-splitters 0.0.1 phục vụ quy trình phân đoạn văn bản. Ngoài ra, các thư viện hỗ trợ như pydantic (kiểm soát dữ liệu), pymysql (kết nối cơ sở dữ liệu) và requests/beautifulsoup4 (thu thập dữ liệu) cũng được tích hợp để hoàn thiện chuỗi xử lý nghiệp vụ.

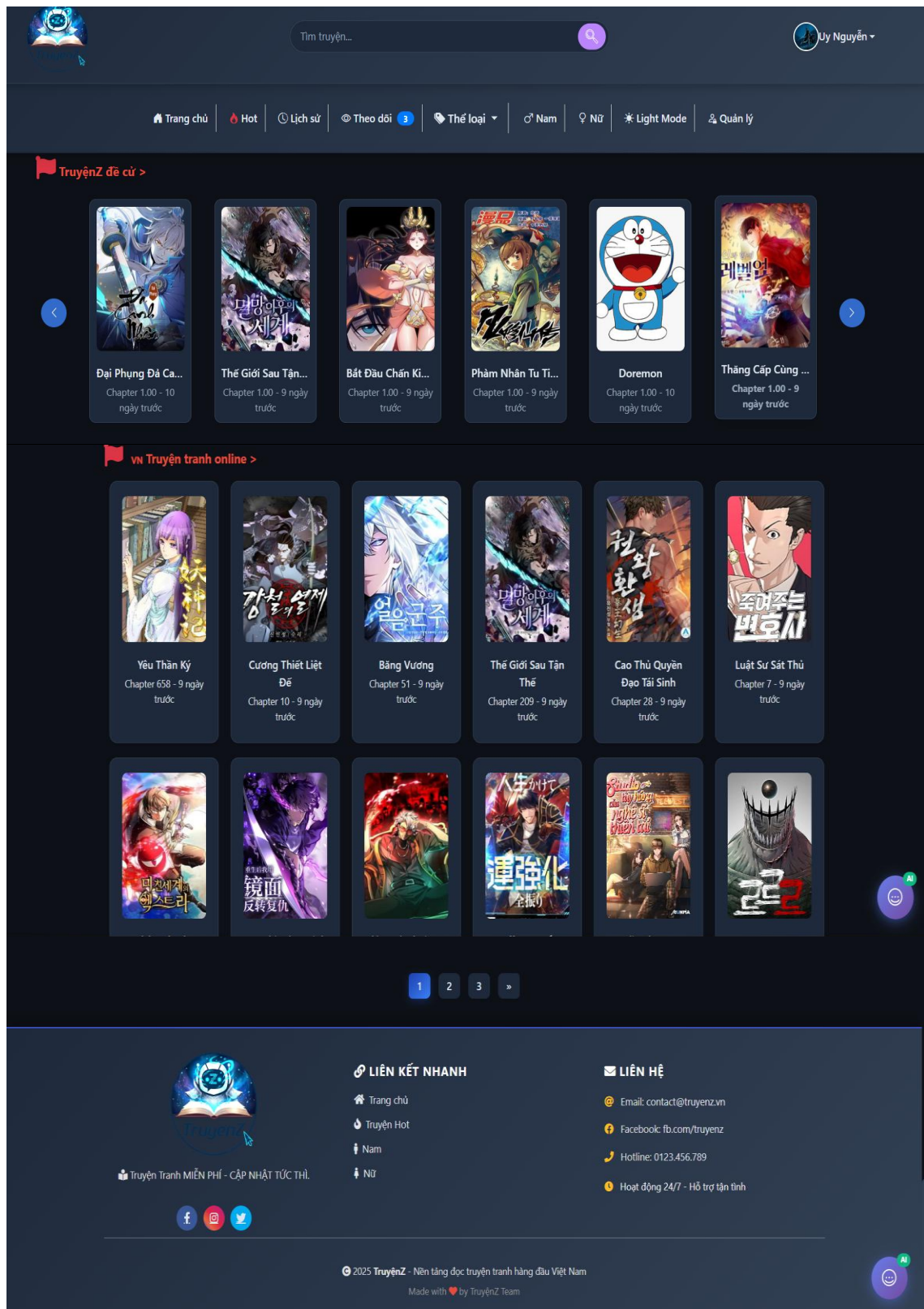
4.2 Xây dựng website

4.2.1 Giao diện người dùng



Hình 4.1: Giao Diện Trang Chủ Dạng LightMode

Đây là trang đầu tiên người dùng sẽ thấy khi truy cập vào website TruyenZ. Tại đây sẽ hiển thị các bộ truyện để người dùng có thể tùy chọn, những mẫu truyện được đề cử là những truyện có nhiều lượt xem nhất, có thanh menu để tùy chọn theo ý muốn.



Hình 4.2: Giao Diện Trang Chủ Dạng DarkMode

Đây là giao diện khi người dùng truy cập website TruyenZ vào ban đêm hoặc chuyển sang chế độ DarkMode. Hiển thị đầy đủ như dạng LightMode. Có header để chọn tính năng và footer để liên hệ nếu muốn.

★ Truyện tranh online >



Theo dõi

One Piece

Cập nhật: 2025-11-28 20:34:46

Tác giả: Eiichiro Oda

Tình trạng: Ongoing

Thể loại: Drama, Fantasy, Action, Comedy, Adventure, Shounen, Supernatural

Lượt xem: 497510348

Đánh giá: 8.0/5 (1142 lượt)

Theo dõi: 221465 người

Nội dung truyện

Không chỉ dừng lại ở truyện tranh, TruyệnZ còn là thiên đường của webtoon màu sắc, tiểu thuyết ngôn tình đậm chất tình cảm, hay những bộ đam mỹ đầy lãng mạn. Với kho truyện đa dạng từ cổ đại đến x [Xem thêm >](#)

Danh sách chương

📖 Đọc từ đầu

🔚 Đọc chương cuối

Chapter 1165 - Chương 1165	10/06/2025
Chapter 1164 - Chương 1164.5	10/10/2025
Chapter 1163 - Chương 1163	01/11/2025
Chapter 1162 - Chương 1162	10/11/2025
Chapter 1161 - Chương 1161	10/11/2025

Bình luận

Gửi bình luận



hoathan1
Chương này tiết lộ nhiều thông tin quan trọng quá.
3 ngày trước



minhthan1
Luffy đang dần làm chủ được Gear 5!
4 ngày trước



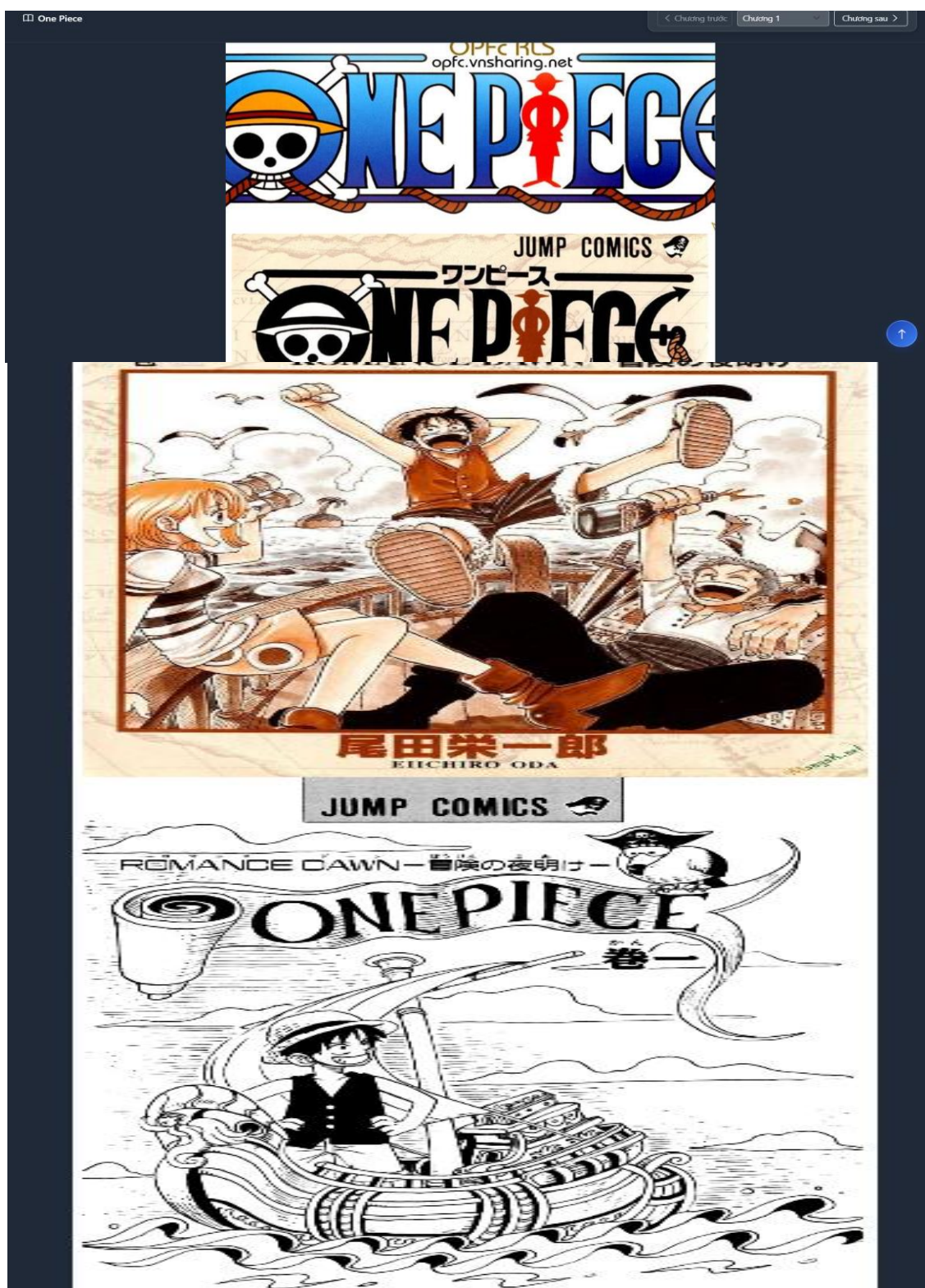
Van Uy
Thiết kế nhân vật mới nhìn ấn tượng thật.
4 ngày trước

Hình 4.3: Giao Diện Trang Chi Tiết Truyện.

Khi người dùng nhấn vào bộ truyện được chọn sẽ chuyển đến trang này để hiển thị toàn bộ thông tin của bộ truyện, các chương và các bình luận của bộ truyện. Có thể đọc từ đầu của bộ truyện hoặc đọc chương cuối là chương mới ra của truyện.

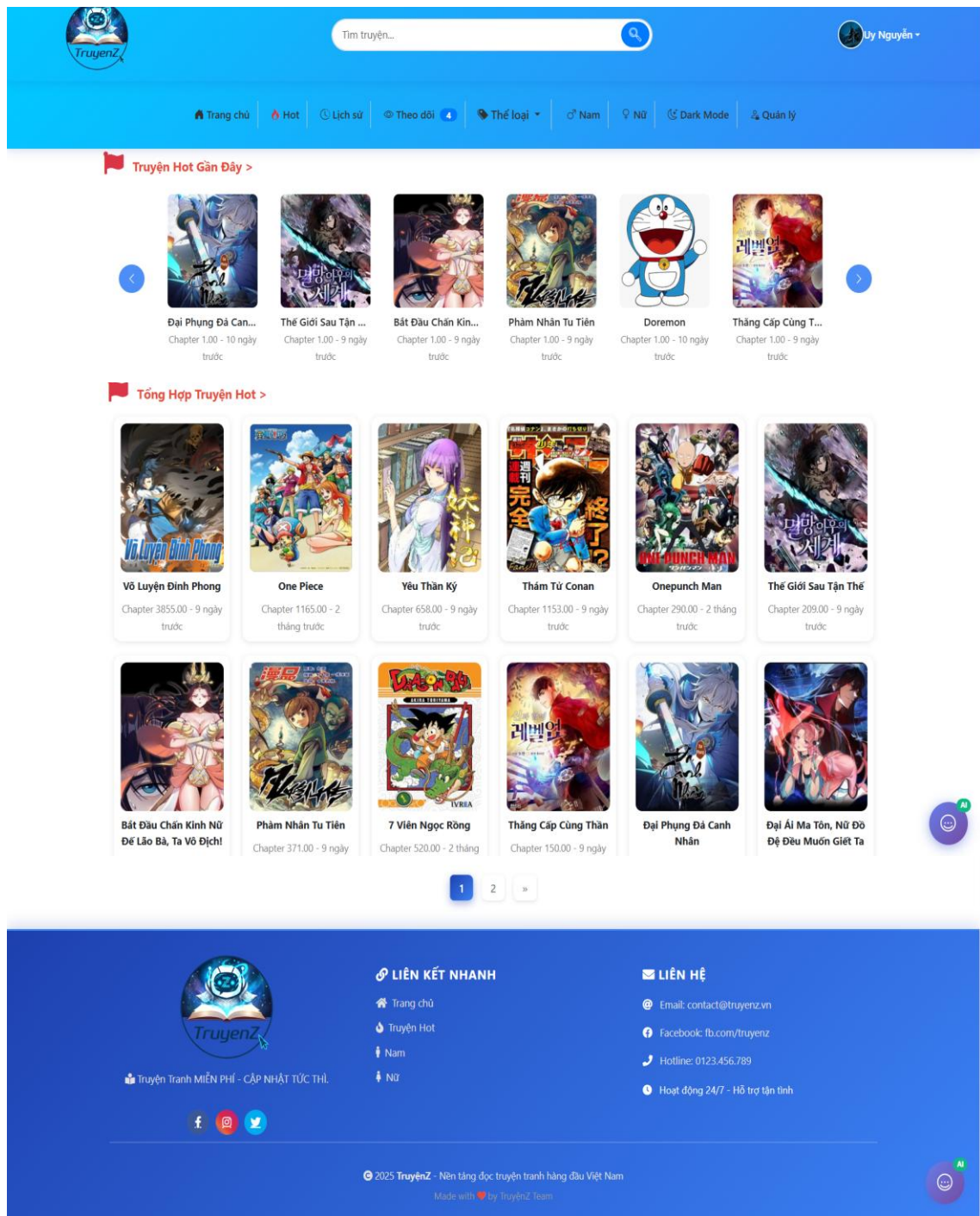
Sinh viên thực hiện: Nguyễn Văn Uy. – GVHD: TS. Nguyễn Năng Hùng Vân.

Trang: 83



Hình 4.4: Giao Diện Trang Đọc Truyện.

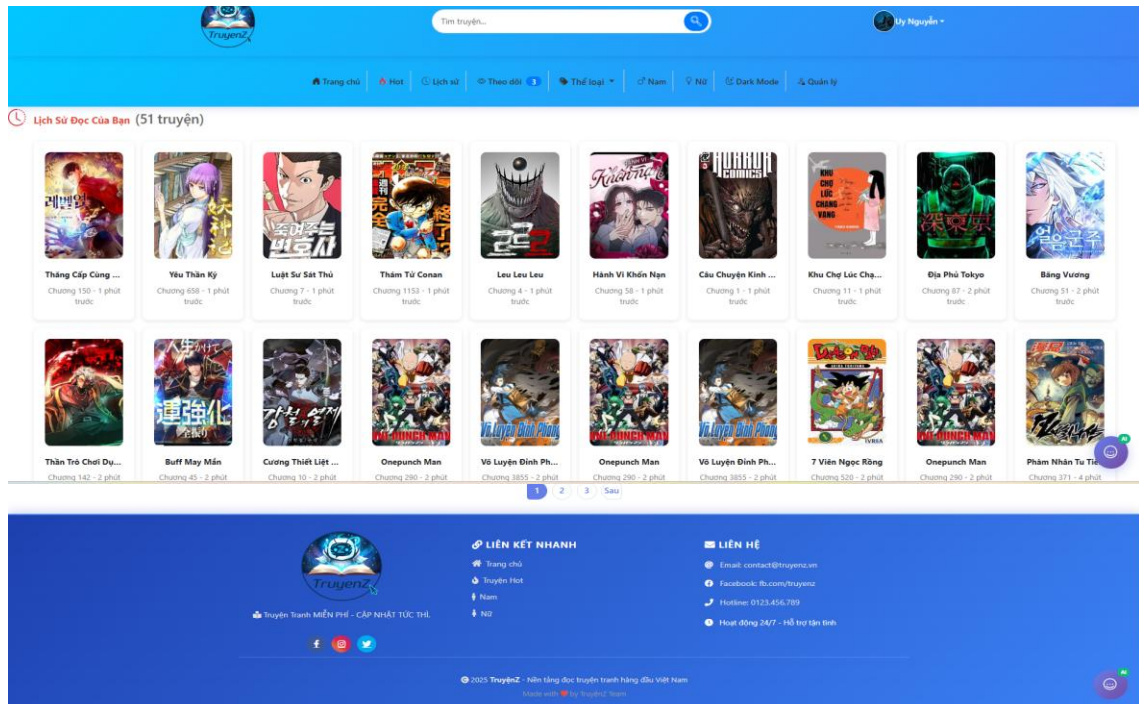
Từ trang chi tiết truyện khi người dùng chọn chương để đọc sẽ qua trang này, đây nơi sẽ hiển thị các hình ảnh để người dùng đọc truyện, có thể chọn đọc chương sau hoặc quay về chương trước hoặc chọn chương mà mình muốn đọc.



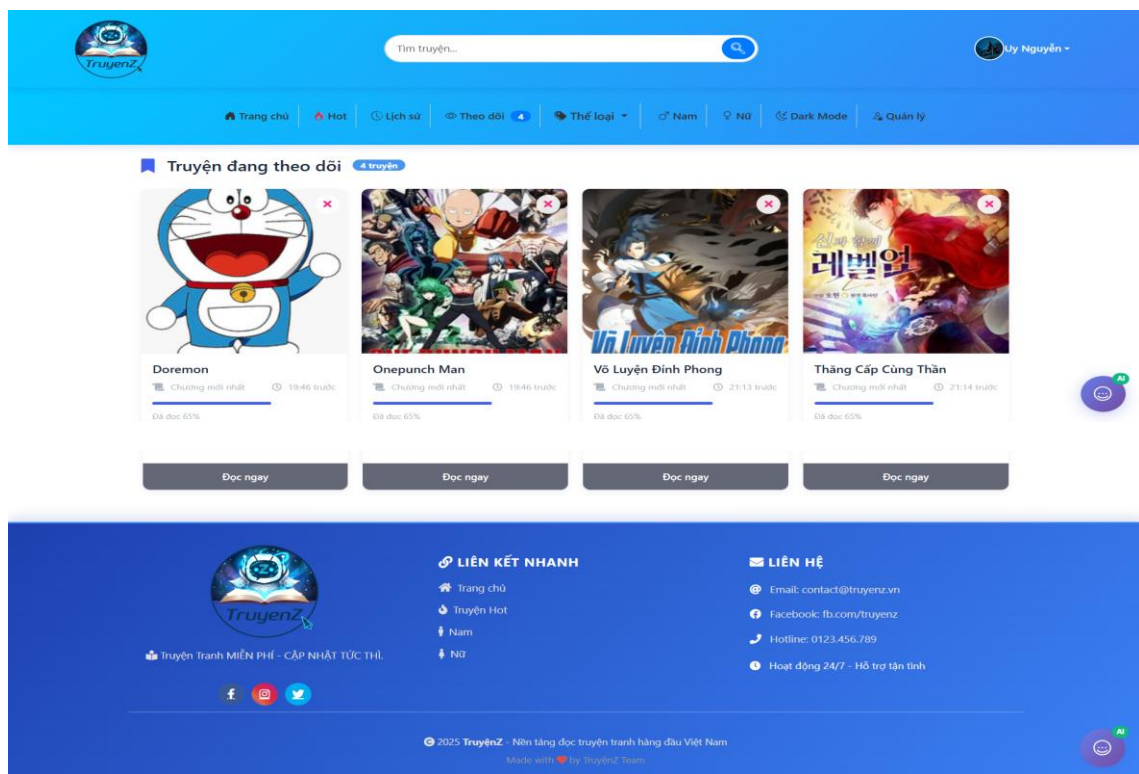
Hình 4.5: Giao Diện Trang Truyện Hot.

Khi người dùng nhấn vào nút Hot trên thanh menu thì sẽ hiển thị toàn bộ truyện có lượt xem cao nhất tại đây. Có thể chọn đọc những bộ truyện được nhiều lượt xem nhất hoặc nhiều lượt theo dõi nhất.

Khi người dùng nhấn vào nút Lịch sử trên thanh Navbar thì sẽ hiển thị những bộ truyện mà người dùng đã đọc qua ở mấy phút tại đây để dễ dàng tìm được những bộ truyện mình đang đọc dở dang.

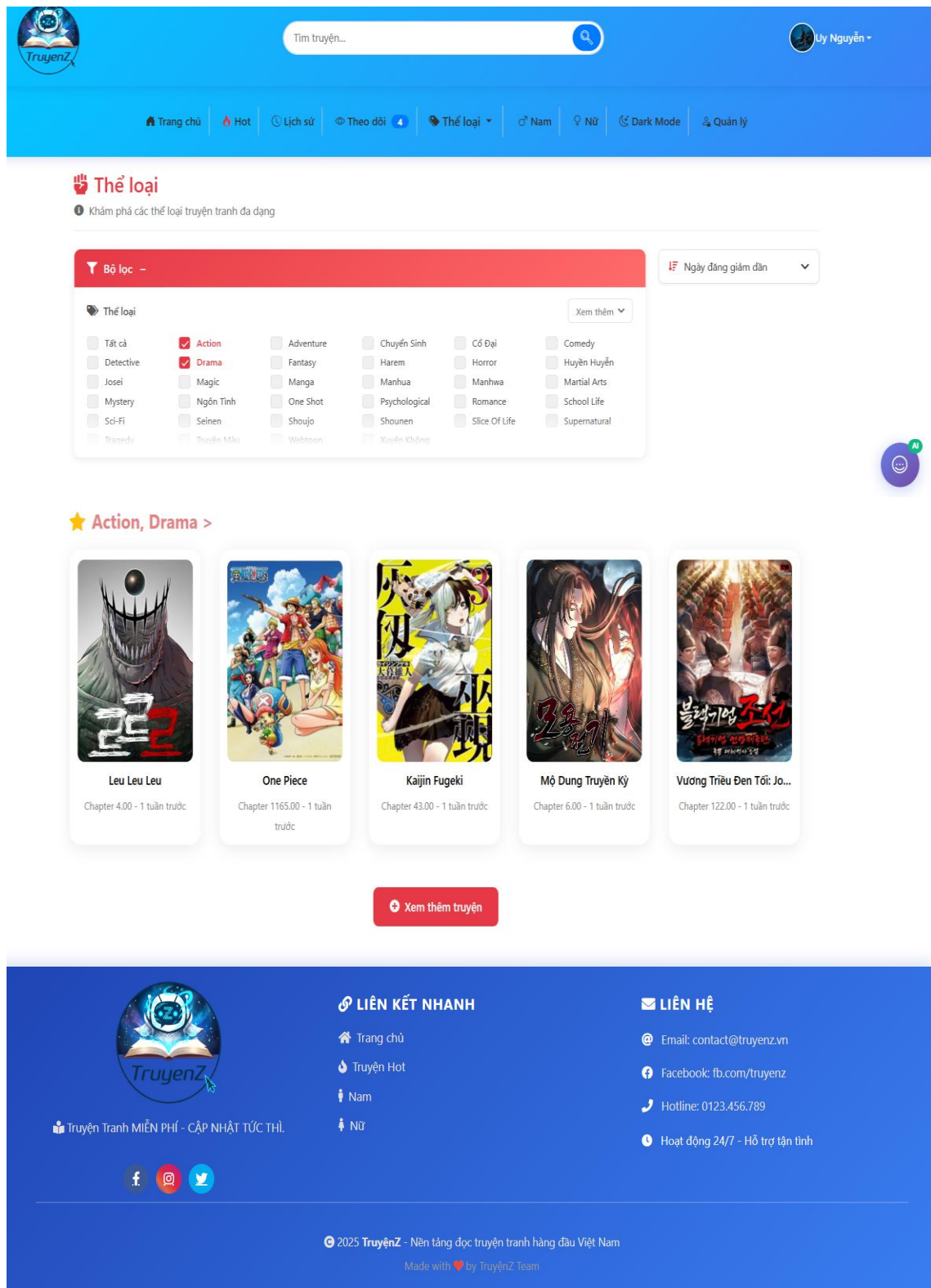


Hình 4.6: Giao Diện Trang Lịch Sử Đọc Truyện



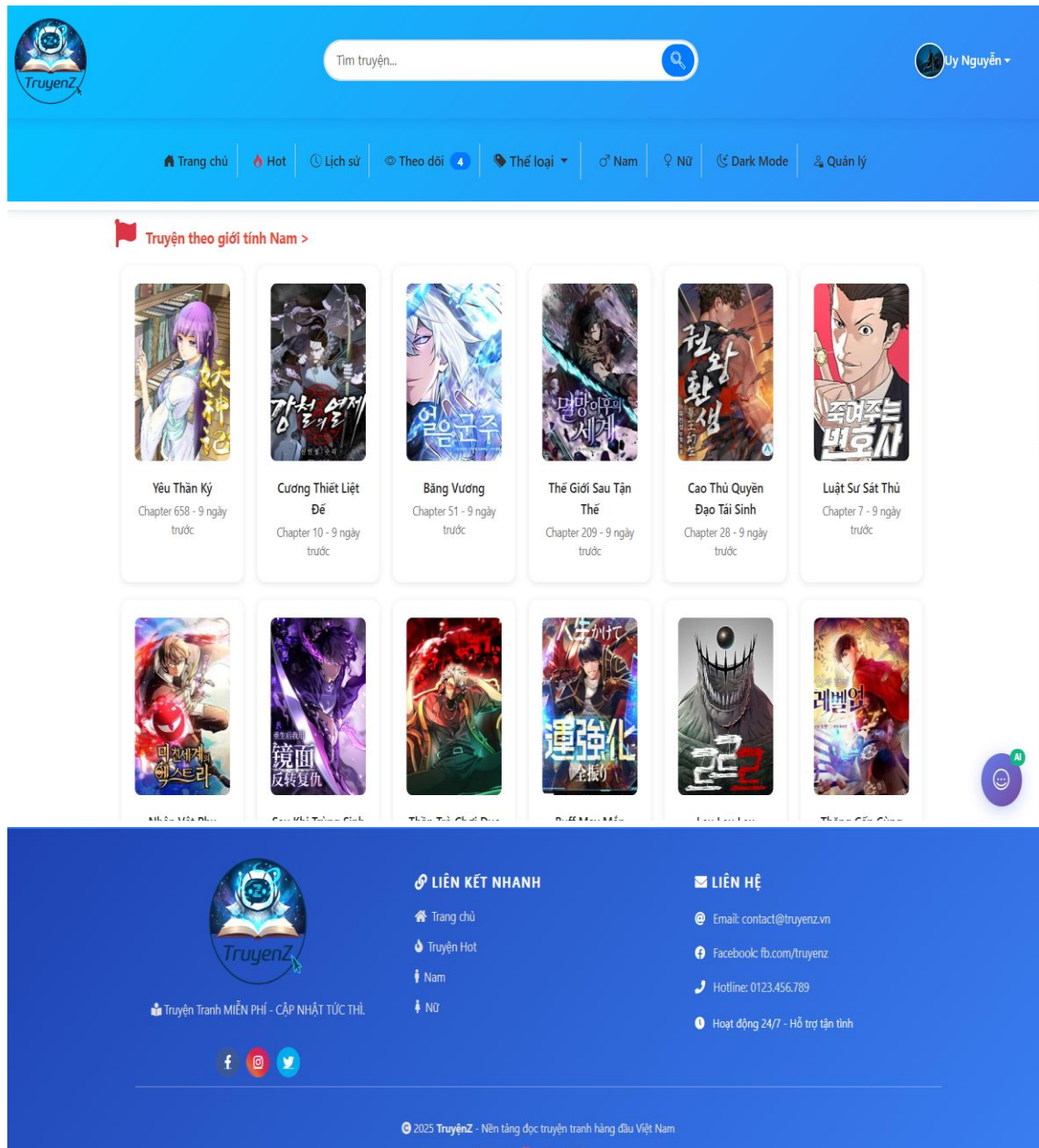
Hình 4.7: Giao Diện Trang Truyện Đang Theo Dõi

Khi người dùng nhấn vào nút theo dõi trên màn hình sẽ chuyển tới trang này. Tại đây hiển thị các truyện mà người dùng quan tâm và nhấn nút theo dõi, để tránh đọc nhiều truyện quá mà không thể đọc tiếp bộ truyện hay đang theo dõi. Hiển thị tiến độ đọc để dễ dàng đọc bộ truyện mà mình đang đọc.



Hình 4.8: Giao Diện Trang Chọn Thể Loại.

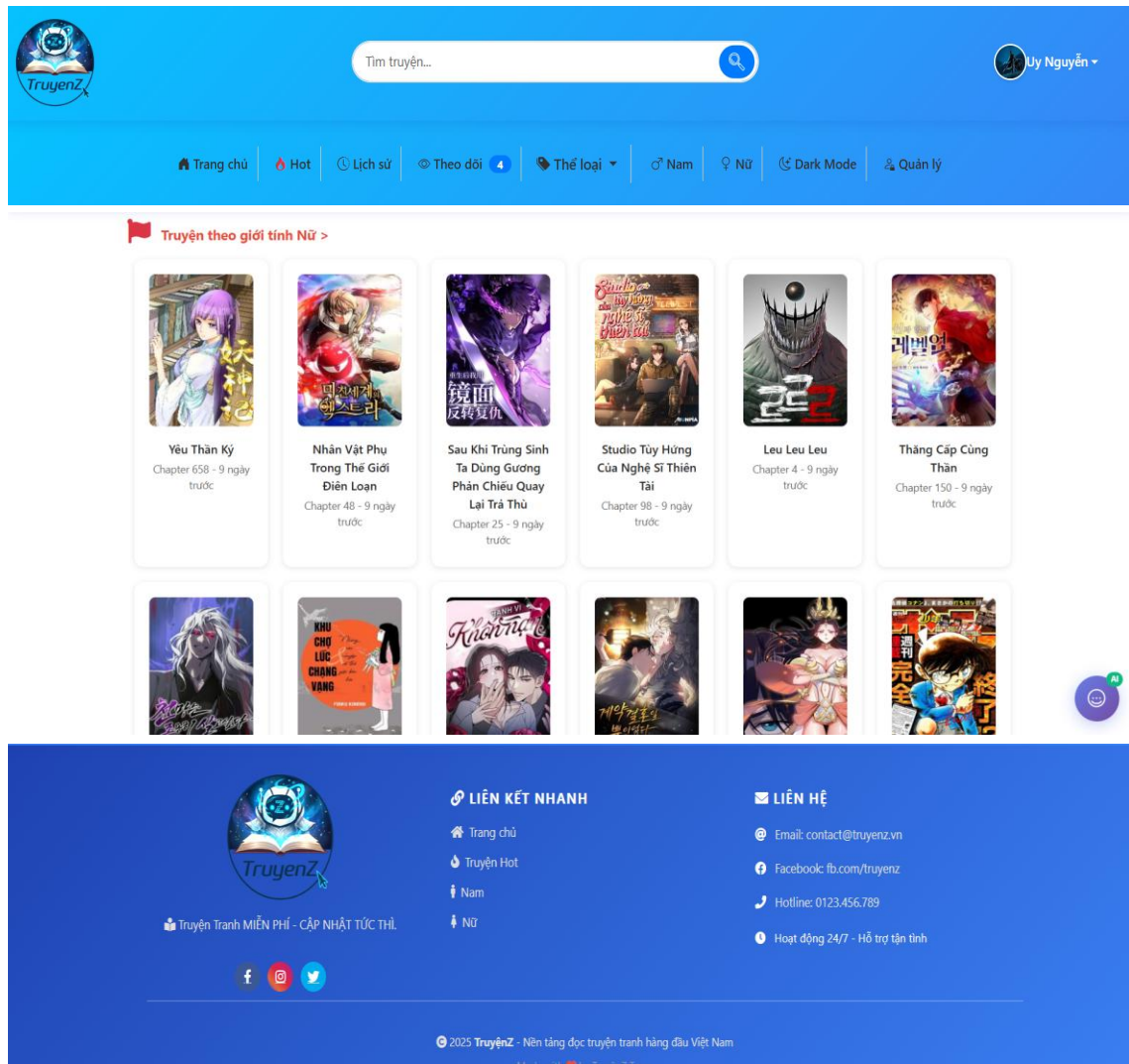
Khi người dùng nhấn chọn thể loại trên thanh DropDown ở menu thì sẽ chuyển tới trang này để người dùng tùy ý chọn những thể loại mà người dùng thích. Trang này có bộ lọc để hiển thị những bộ truyện đúng ý người dùng, có hiển thị ngày để dễ dàng tùy chọn truyện mới ra mắt.



Hình 4.9: Giao Diện Trang Truyện Dành Cho Nam.

Khi người dùng nhấn vào nút Nam trên thanh menu sẽ chuyển tới trang này. Trang sẽ hiển thị toàn bộ các truyện có thể loại nam tính, mạnh mẽ để phù hợp với người dùng nâng cao tính cá nhân hóa hơn.

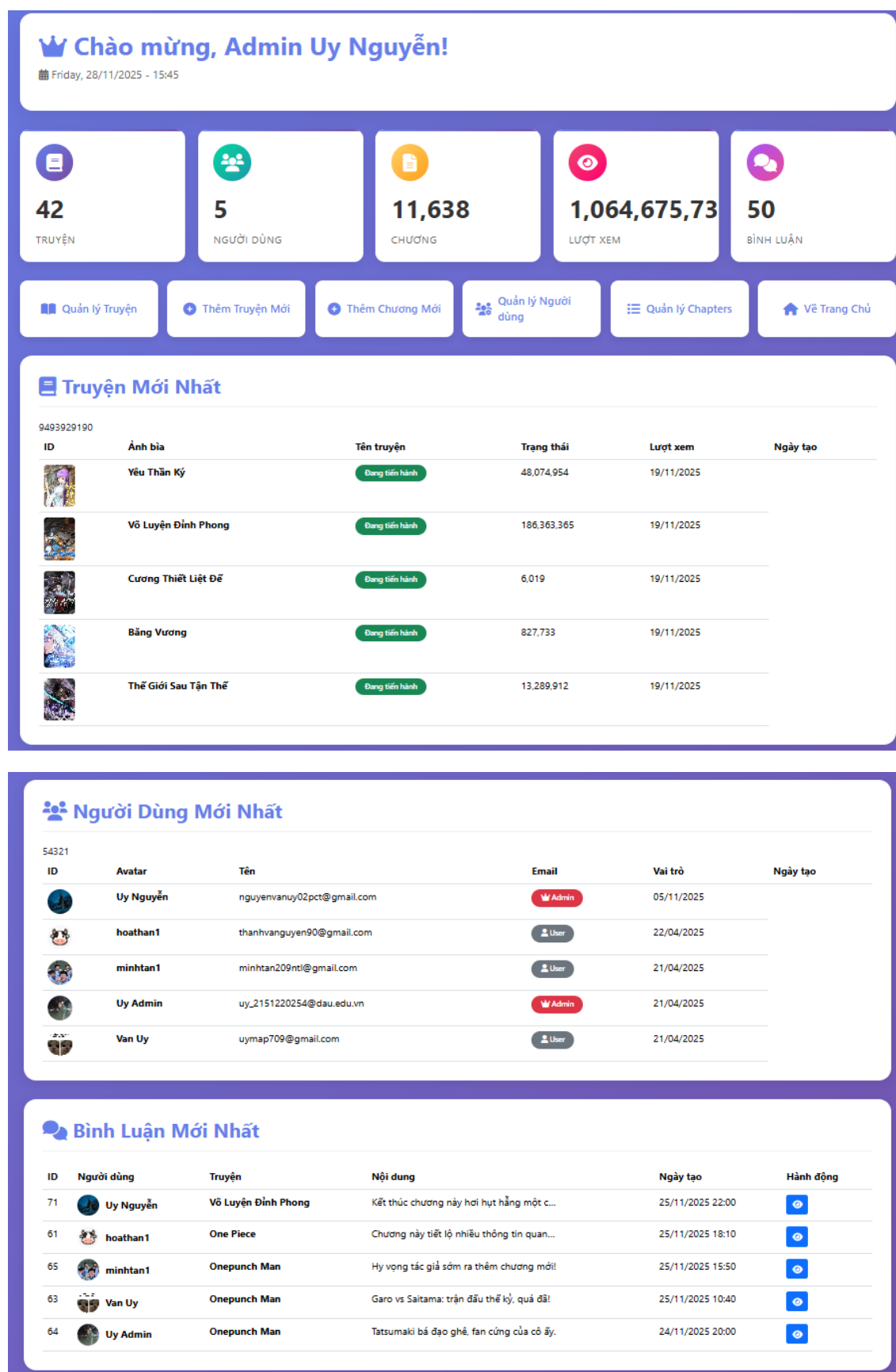
Khi người dùng nhấn vào nút Nữ trên thanh Navbar sẽ chuyển tới trang này. Trang sẽ hiển thị toàn bộ các truyện có thể loại nữ tính, dễ thương để phù hợp với người dùng nâng cao tính cá nhân hóa hơn.

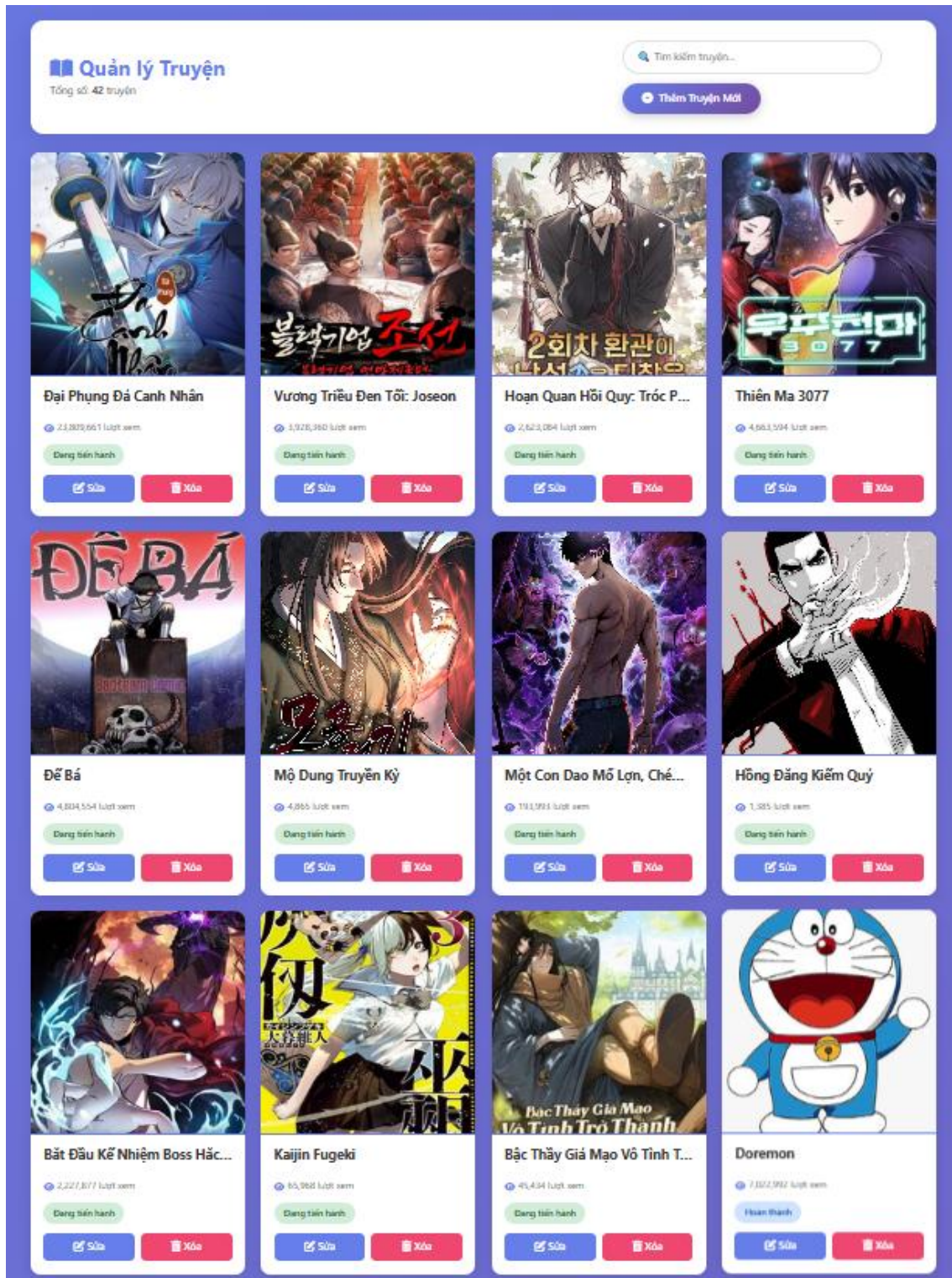


Hình 4.10: Giao Diện Trang Truyện Dành Cho Nữ.

4.2.2 Giao diện quản trị

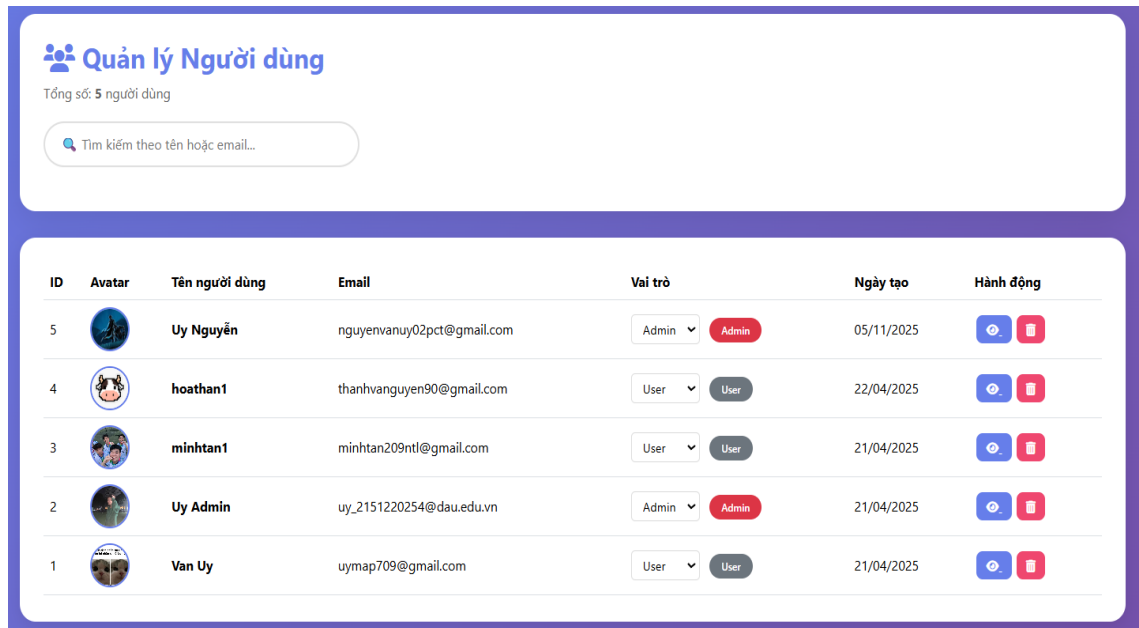
Khi đăng nhập tài khoản có quyền Admin vào trang web sẽ lập tức hiển thị trang này. Nơi mà quản trị viên có thể dễ dàng quan sát tổng quan toàn bộ hệ thống, bao gồm các tính năng quản lý truyện, người dùng, chương truyện, lượt xem, bình luận cũng như các thao tác quản lý truyện, quản lý chương, quản lý người dùng, thêm nhanh truyện mới, thêm nhanh chương mới. Giao diện được tối ưu để hỗ trợ công tác vận hành, giúp quản trị viên thao tác thuận tiện, kiểm soát chặt chẽ dữ liệu và kịp thời xử lý các vấn đề phát sinh trong quá trình hoạt động của website. Nhờ đó, việc quản lý nội dung truyện và người dùng trở nên hiệu quả, đồng thời vận hành website dễ dàng, tốt hơn.





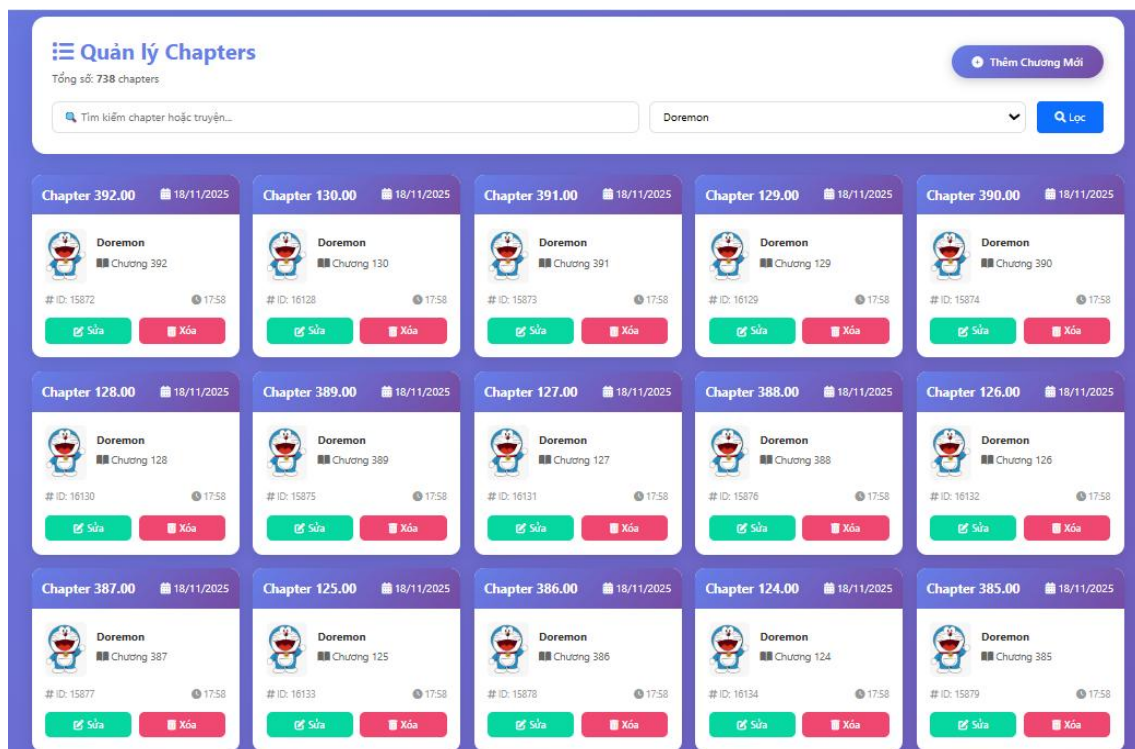
Hình 4.12: Giao Diện Trang Quản Lý Truyện.

Khi nhấn vào quản lý truyện sẽ chuyển tới trang này, ở trang quản lý truyện này sẽ hiển thị các bộ truyện trong hệ thống. Có các chức năng thêm, sửa, xóa các bộ truyện, có thể tùy chỉnh các truyện thành đã hoàn thành hoặc đang tiến hành.



Hình 4.13: Giao Diện Trang Quản Lý Người Dùng.

Khi nhấn vào quản lý người dùng sẽ chuyển tới trang này, tại đây có thể xem thông tin, thêm, sửa, xóa các người dùng.



Hình 4.14: Giao Diện Trang Quản Lý Chương.

Khi nhấn vào quản lý chương sẽ chuyển tới trang này. Trang này hiển thị thông tin của các chương để có thể tùy chọn chỉnh sửa, có thể thêm mới hoặc xóa tùy ý hoặc ẩn chương đi.

HOẶC

Thêm Truyện Thủ Công

Tên Truyện *

Nhập tên truyện...

Tác Giả

Đang cập nhật

Mô Tả *

Nhập mô tả truyện...

Ảnh Bìa

Chọn tệp

Không có tệp nào được chọn

Trạng Thái *

Đang tiến hành

Thể Loại

☐ Action

☐ Detective

☐ Huyền Huyền

☐ Manhwa

☐ Psychological

☐ Shoujo

☐ Truyện Màu

☐ Adventure

☐ Drama

☐ Josei

☐ Martial Arts

☐ Romance

☐ Shounen

☐ Webtoon

☐ Chuyển Sinh

☐ Fantasy

☐ Magic

☐ Mystery

☐ School Life

☐ Slice Of Life

☐ Xuyên Không

☐ Cổ Đại

☐ Harem

☐ Manga

☐ Ngôn Tình

☐ Sci-Fi

☐ Supernatural

☐ Comedy

☐ Horror

☐ Manhwa

☐ One Shot

☐ Seinen

☐ Tragedy

Thêm Truyện Thủ Công

Khi nhấn vào thêm truyện sẽ chuyển tới trang này. Tại đây có thể thêm truyện thủ công bằng cách tải thông tin và ảnh của truyện lên hoặc tải lên tự động từ website khác.

 **Crawl Chương Mới Tự Động từ TruyenQQGo**

Tự động crawl 1 chương mới vào truyện đã có (bao gồm tất cả ảnh)!

📖 Chọn Truyện *

-- Chọn truyện --

🔗 URL Chương Mới *

https://truyenqqgo.com/truyen-tranh/ten-truyen-12345-chap-21.html

Ví dụ: https://truyenqqgo.com/truyen-tranh/gay-go-cap-99-12236-chap-21.html

 **Crawl Chương Mới**

HOẶC

Thêm Chương Thủ Công

📖 Chọn Truyện *

-- Chọn truyện --

Số Chương *

1

Số chương tiếp theo sẽ là: 1

📝 Tên Chương

Để trống sẽ tự động đặt tên 'Chương X'

📌 Trạng Thái *

Published

📁 Upload Ảnh Chương (Tùy chọn)

 Bạn có thể upload ảnh ngay bây giờ hoặc thêm sau bằng cách chỉnh sửa chương.



Click để chọn ảnh hoặc kéo thả vào đây

Hỗ trợ nhiều ảnh cùng lúc (JPG, PNG, GIF)

 **Thêm Chương Mới**

Hình 4.16: Giao Diện Trang Thêm Chương Truyện.

Khi nhấn vào thêm chương sẽ chuyển tới trang này. Tại đây có thể thêm chương thủ công bằng cách tải thông tin và ảnh của chương lên hoặc tải lên chương tự động.

Thông Tin Truyện

Tên Truyện *

Lưu Lưu Lưu

Mô Tả *

Ảnh Bìa

Chọn tệp

Không có tệp nào được chọn



Trạng Thái *

Đang tiến hành

Quản lý Chapters (4 chương)

Thêm Chương Mới

Chapter 4.00

ID: 21243

Tên Chapter

Chương 4

Chỉnh sửa chi tiết

Xóa Chapter

Chapter 3.00

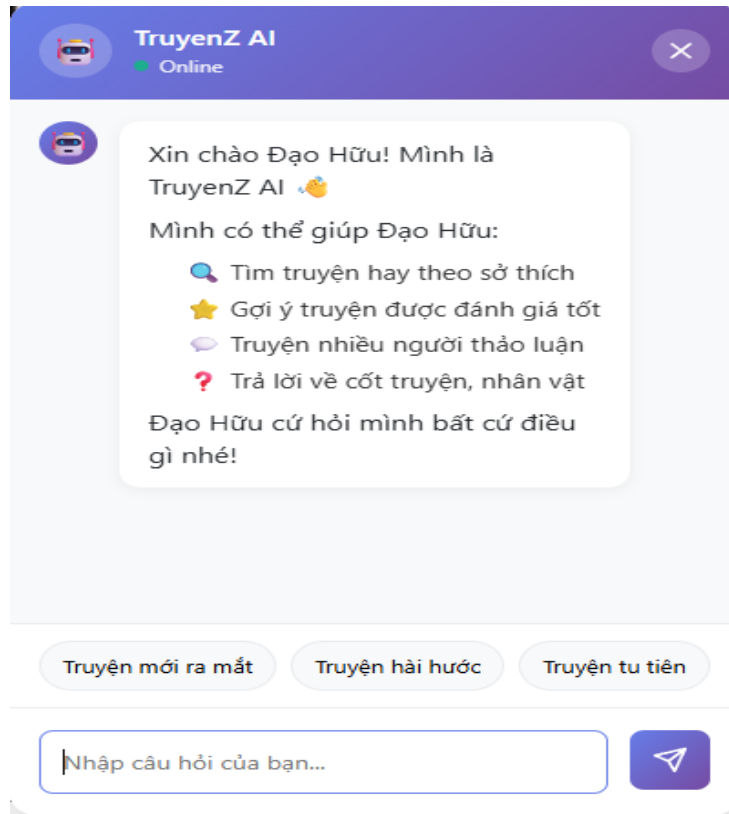
ID: 21244

Tên Chapter

Chương 3

Hình 4.17: Giao Diện Trang Sửa Thông Tin Truyện.

Khi nhấn sửa truyện sẽ chuyển tới chương này. Tại đây có thể tùy chỉnh bộ truyện như chọn trạng thái truyện, ẩn đi hoặc xuất hiện, thêm, sửa xóa các chương, viết mô tả của bộ truyện.



Hình 4.19: Giao Diện Widget Chatbot.

Khi nhấn vào ô chat trên màn hình sẽ hiển thị ra khung chat này, tại đây có thể hỏi thông tin các bộ truyện để được hệ thống gợi ý, tóm tắt nội dung.

4.3 Triển khai AI RAG Pipeline

Phần hệ trí tuệ nhân tạo (AI Module) được thiết kế trên nền kiến trúc RAG (Retrieval-Augmented Generation), đảm nhiệm vai trò trung tâm trong việc hỗ trợ người dùng tra cứu thông tin truyện và tạo ra bản tóm tắt nội dung theo yêu cầu [2]. Toàn bộ quy trình được tổ chức thành bốn giai đoạn kỹ thuật liên hoàn, bao gồm xử lý dữ liệu truyện tranh, xây dựng kho tri thức, truy xuất thông tin liên quan và cuối cùng là tạo sinh phản hồi phù hợp với ngữ cảnh truy vấn của người dùng [5].

4.3.1 Chuẩn bị và Vector hóa dữ liệu

Để đảm bảo chất lượng tri thức đầu vào cho mô hình, quy trình xử lý dữ liệu (ETL) được thực hiện nghiêm ngặt thông qua module `data_processor_unified.py`.

4.3.1.1 Chiến lược phân đoạn văn bản (Chunking Strategy)

Dữ liệu thô từ các tập tin JSON sau khi được làm sạch sẽ được phân chia thành các đơn vị thông tin nhỏ hơn (chunks). Hệ thống sử dụng thuật toán `RecursiveCharacterTextSplitter` từ thư viện `LangChain` để thực hiện tác vụ này[14].

Kích thước đoạn (chunk_size) được thiết lập là 800 ký tự, tương đương khoảng 150-200 từ tiếng Việt. Đây là độ dài tối ưu được lựa chọn dựa trên thực nghiệm, đủ để chứa trọn vẹn một ngữ cảnh ngữ nghĩa (như một đoạn mô tả nhân vật hay tình tiết cốt truyện) mà không gây nhiễu thông tin.

Bên cạnh đó, độ chồng lấp (chunk_overlap) được thiết lập ở mức 150 ký tự (khoảng 20% kích thước chunk). Về mặt kỹ thuật, tham số này đảm bảo tính liên kết ngữ nghĩa (semantic continuity) giữa các vector liên kề, ngăn chặn hiện tượng mất mát thông tin (information loss) nếu một câu quan trọng nằm đúng tại điểm cắt của thuật toán [5]. Thuật toán đệ quy sẽ ưu tiên ngắt đoạn tại các dấu phân cách lớn (\n\n) trước. Nếu đoạn văn vẫn lớn hơn chunk_size, nó mới tiếp tục xét đến các dấu phân cách nhỏ hơn (.). Trích xuất từ data_processor_unified.py:

```
text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=800,      # Kích thước mỗi đoạn: 800 ký tự
    chunk_overlap=150,   # Độ chồng lấp: 150 ký tự
    length_function=len,
    separators=["\n\n", "\n", ". ", "!", "? ", "; ", " ", ""]
)
```

Ví dụ: "One Piece là bộ truyện nổi tiếng, Luffy là nhân vật chính, Anh ăn trái ác quỷ."

- Chunk 1: "One Piece là bộ truyện nổi tiếng. Luffy là nhân vật chính."
- Chunk 2: "Luffy là nhân vật chính. Anh ăn trái ác quỷ."

↑ Overlap giữ ngữ cảnh về Luffy

- Giải thích kỹ thuật: Thuật toán đệ quy (Recursive) sẽ ưu tiên ngắt đoạn tại các dấu phân cách lớn (\n\n) trước. Nếu đoạn văn vẫn lớn hơn chunk_size (800), nó mới tiếp tục xét đến các dấu phân cách nhỏ hơn (.). Tham số chunk_overlap=150 đảm bảo các câu văn nằm ở ranh giới cắt không bị mất nghĩa khi chuyển sang đoạn kế tiếp.

4.3.1.2 Vector hóa và lập chỉ mục (Embedding & Indexing)

Quá trình chuyển đổi văn bản sang không gian vector thực hiện thông qua mô hình text-embedding-3-small của OpenAI. Mỗi đoạn văn bản d được ánh xạ thành một vector $v \in R^{1536}$ (1536 chiều). Việc sử dụng vector không gian cao chiều giúp nắm bắt được các sắc thái ngữ nghĩa tinh tế của tiếng Việt [10].

Toàn bộ các vector này được lưu trữ và lập chỉ mục trong ChromaDB. Tại đây, thuật toán HNSW (Hierarchical Navigable Small World) được áp dụng để xây dựng cấu

trúc đồ thị, cho phép thực hiện truy vấn lân cận gần đúng (ANN) với độ phức tạp tìm kiếm là $O(\log N)$, đảm bảo thời gian truy xuất dưới 50ms cho tập dữ liệu 604 vector hiện tại [3], [13].

Sau khi phân đoạn, văn bản được chuyển đổi thành vector số học. Hàm `get_embeddings` chịu trách nhiệm gọi API của OpenAI để thực hiện tác vụ này. Trích xuất từ hàm `get_embeddings` trong `data_processor_unified.py`

```
def get_embeddings(texts: List[str]) -> List[List[float]]:
    #Gọi OpenAI API để tạo embedding
    response = openai_client.embeddings.create(
        model="text-embedding-3-small",
        input=clean_texts
    )
    # Trích xuất vector từ phản hồi JSON
    embeddings = [item.embedding for item in response.data]
    print(f"✅ Generated {len(embeddings)} embeddings")
    return embeddings
except Exception as e:
    print(f"❌ Embedding Error: {e}")
    raise
```

- Giải thích kỹ thuật: Hệ thống sử dụng mô hình `text-embedding-3-small`, trả về một mảng số thực (vector) có kích thước cố định là 1536 chiều cho mỗi đoạn văn bản đầu vào. Đây là biểu diễn toán học của ngữ nghĩa đoạn văn.

Dữ liệu sau khi vector hóa được lưu trữ bền vững vào ChromaDB thông qua hàm `populate_vector_database`. Quá trình này bao gồm việc lưu vector, nội dung gốc và metadata để phục vụ tìm kiếm lại sau này. Trích xuất từ hàm `populate_vector_database` trong `data_processor_unified.py`

```
def populate_vector_database(chunks: List[Dict]):
    # ... (Code xử lý batch để tránh quá tải API) ...
    print("Adding to ChromaDB...")
    collection.add(
        ids=ids,
        embeddings=embeddings,
        documents=texts,
        metadatas=metadatas
    )
```

- Giải thích kỹ thuật: Phương thức `collection.add` kích hoạt thuật toán HNSW bên dưới của ChromaDB để xây dựng chỉ mục đồ thị, giúp việc truy vấn hàng nghìn vector sau này diễn ra với tốc độ mili-giây [3].

4.3.2 Hiện thực hóa phân loại ý định

Thay vì sử dụng các mô hình học sâu phức tạp gây độ trễ cao, module `chatbot_unified.py` triển khai một giải thuật phân loại dựa trên luật (Rule-based Classification) tối ưu hóa cho miền ứng dụng hẹp. Hàm `classify_intent` hoạt động dựa trên cơ chế so khớp từ khóa có trọng số:

Logic xử lý: Hệ thống quét câu truy vấn đầu vào Q để tìm các tập từ khóa đặc trưng K_i tương ứng với ý định I_i .

- Nếu $Q \cap K_{action} \neq \emptyset$ (chứa từ khóa "hành động", "đánh nhau") $\rightarrow$ Intent: `recommend_action`.
- Nếu $Q \cap K_{plot} \neq \emptyset$ (chứa từ khóa "nói về gì", "cốt truyện") $\rightarrow$ Intent: `plot_summary`.
- Nếu $Q \cap K_{realtime} \neq \emptyset$ (chứa từ khóa "hot", "mới nhất") $\rightarrow$ Intent: `trending`.

Hiệu quả: Phương pháp này đảm bảo độ chính xác gần như tuyệt đối (Deterministic) cho các kịch bản định trước và thời gian xử lý không đáng kể (Sub-millisecond), giúp dành trọn ngân sách thời gian cho các bước xử lý AI phía sau [5]. Trích từ file `chatbot_unified.py`

```
def classify_intent(user_query: str) -> Tuple[str, Dict[str, Any]]:
    """
    Classify user intent from query
    Returns:
        (intent_name, metadata_dict)
    20+ Intent Types:
    - greeting, thanks
    - recommend_action, recommend_comedy, recommend_cultivation, etc.
    - plot_summary, character_info, world_building
    - comparison, find_new, trending
    - most_viewed, least_viewed, top Rated, most_commented
    """
    q = user_query.lower()
    metadata = {}
    # 1. Greetings & Social
    if any(w in q for w in ['xin chào', 'chào', 'hi', 'hello']):
        return 'greeting', metadata
    if any(w in q for w in ['cảm ơn', 'thank', 'cám ơn']):
        return 'thanks', metadata
```

```
if any(w in q for w in ['mới', 'new', 'mới nhất', 'vừa ra', 'update']):
    return 'find_new', metadata
if any(w in q for w in ['hot', 'trending', 'phổ biến', 'đang hot']):
    return 'trending', metadata
if any(w in q for w in ['nhiều lượt xem', 'nhiều view', 'xem nhiều']):
    return 'most_viewed', metadata
if any(w in q for w in ['ít lượt xem', 'ít view', 'xem ít']):
    return 'least_viewed', metadata
if any(w in q for w in ['đánh giá cao', 'rating', 'điểm cao']):
    return 'top_rated', metadata
if any(w in q for w in ['nhiều comment', 'bình luận nhiều', 'thảo luận']):
    return 'most_commented', metadata
```

Giải thích kỹ thuật: Hàm `classify_intent()` thực hiện phân loại ý định người dùng bằng cách chuyển truy vấn về chữ thường và kiểm tra sự xuất hiện của các từ khóa thuộc từng nhóm nghiệp vụ. Các intent được nhận diện theo tập luật đơn giản, bao gồm chào hỏi, cảm ơn, truyện mới, xu hướng, nhiều lượt xem, ít lượt xem, đánh giá cao và nhiều bình luận. Khi một tập từ khóa trùng khớp, hàm trả về tên intent tương ứng cùng metadata rỗng. Cách tiếp cận dựa trên từ khóa này giúp hệ thống xử lý nhanh, dễ mở rộng và phù hợp cho việc nhận diện truy vấn tiếng Việt trong chatbot đọc truyện.

4.3.3 Kỹ thuật Prompt Engineering và tiềm ngữ cảnh

Kỹ thuật thiết kế câu lệnh (Prompt Engineering) đóng vai trò là "người quản lý" hành vi của mô hình ngôn ngữ lớn, đảm bảo AI không chỉ trả lời đúng mà còn trả lời an toàn và phù hợp với ngữ cảnh ứng dụng [6].

4.3.3.1 Thiết kế chỉ dẫn hệ thống (System Prompt Design)

Hệ thống áp dụng chiến lược thiết kế Prompt chặt chẽ, hoạt động như một lớp rào chắn bảo vệ (Guardrails) để kiểm soát chất lượng đầu ra. Cấu trúc Prompt bao gồm ba thành phần cốt lõi:

- Định danh vai trò (Role Playing): Thiết lập nhân cách cố định cho AI: "Bạn là TruyenZ Bot - Chuyên gia tư vấn truyện tranh". Điều này giúp định hình giọng văn (tone & voice) nhất quán xuyên suốt cuộc hội thoại.
- Nguyên tắc chống ảo giác (No-Hallucination): Nhúng cứng chỉ thị bắt buộc: "Chỉ trả lời dựa trên [Context] được cung cấp. Nếu thông tin không có trong ngữ cảnh, hãy trả lời là không biết, tuyệt đối không tự bịa đặt". Đây là biện pháp kỹ thuật quan trọng nhất để đảm bảo tính trung thực của hệ thống [4].

- Cơ chế chống tiết lộ (Anti-Spoiler Mechanism): Thêm ràng buộc "Không được tiết lộ kết thúc hoặc các tình tiết nút thắt quan trọng" để bảo vệ trải nghiệm khám phá của người đọc, giải quyết vấn đề nhạy cảm đặc thù của miền dữ liệu giải trí.
- Code thực hiện: Trích xuất trong chatbot_unified.py

```
def create_system_prompt(intent: str, metadata: Dict = None) -> str:
    """Create optimized system prompt for each intent"""
    comic_list = ", ".join([f"{title}" for title in DATASET_COMICS.values()])
    base_prompt = f"""Bạn là **TruyenZ Bot** - Trợ lý AI chuyên gia về truyện tranh.

    🌀 **NGUYÊN TẮC TỐI QUAN TRỌNG**:
    1. **TRUNG THỰC TUYỆT ĐỐI**:
        - CHỈ trả lời dựa trên [Context] được cung cấp
        - Hệ thống có 42 truyện: {comic_list}
        - Nếu truyện KHÔNG có trong danh sách → "TruyenZ chưa có bộ này..."
        - KHÔNG BAO GIỜ bịa đặt thông tin
    22. **KHÔNG BAO GIỜ SPOILER**:
        - KHÔNG tiết lộ cái kết
        - KHÔNG spoil twist quan trọng
        - Chỉ nói setting, mục tiêu, hành trình
    """

    # Intent-specific instructions
    intent_prompts = {
        'greeting': """
        Chào hỏi nhiệt tình, hỏi họ muốn tìm truyện gì.
        VD: "Chào đạo hữu! 🌀 Mình là TruyenZ Bot, trợ lý tìm truyện của đạo hữu!
        đạo hữu muốn tìm truyện gì? Hành động, hài hước, hay ngôn tình? Cứ nói nhé!
        😊"
        """,
        'ending_question': """
        CÂN THẬN - User hỏi về kết thúc:
        - TUYỆT ĐỐI KHÔNG spoil ending
        - Nói: "Mình không tiết lộ để đạo hữu giữ trọn cảm xúc khi đọc nhé! 😊"
        - Gợi ý đọc để tự trải nghiệm
        """
    }

    specific = intent_prompts.get(intent, intent_prompts.get('recommend_action', ""))
```

```
return base_prompt + "\n\n**NHIỆM VỤ CỤ THỂ**:\n" + specific
```

- Giải thích kỹ thuật: Hàm `create_system_prompt()` tạo System Prompt phù hợp cho từng bối cảnh hội thoại bằng cách sinh phần Prompt nền tảng từ danh sách truyện và các quy tắc quan trọng như chỉ trả lời trong phạm vi dữ liệu và không tiết lộ nội dung kết thúc. Sau đó, hàm bổ sung các chỉ thị chuyên biệt theo từng intent, ví dụ chào hỏi hoặc câu hỏi về ending. Cách ghép Prompt nền và Prompt theo intent giúp chatbot giữ được hành vi ổn định nhưng vẫn linh hoạt trong từng tình huống, đồng thời giảm lỗi sinh nội dung ngoài dữ liệu.

4.3.3.2 Kỹ thuật tiêm ngữ cảnh (Context Injection)

Dữ liệu thô thu được từ quá trình truy xuất Vector (Top-6 chunks) không được đưa trực tiếp vào Prompt mà trải qua bước làm giàu dữ liệu (Metadata Enrichment) để tối ưu hóa khả năng hiểu của mô hình:

- Làm giàu Metadata: Hệ thống tự động bổ sung các thông tin siêu dữ liệu quan trọng (như thể loại, tác giả, trạng thái) vào ngay trước nội dung chính của mỗi đoạn văn bản.
- Tăng tính giải thích (Explainability): Việc cung cấp thêm metadata giúp mô hình thực hiện được các suy luận phức tạp (ví dụ: liên kết giữa phong cách tác giả và nội dung truyện), từ đó sinh ra câu trả lời có tính giải thích cao hơn, giúp người dùng hiểu rõ tại sao một bộ truyện lại được gợi ý cho họ [5].
- Code thực hiện: Trích từ file `chatbot_unified.py`

```
def generate_perfect_response(  
    context_docs: List[str],  
    context metas: List[Dict],  
    user_query: str,  
    intent: str,  
    intent_metadata: Dict = None  
)-> Tuple[str, List[Dict]]:
```

4.3.4 Tích hợp Mô hình tạo sinh

Giai đoạn tạo sinh văn bản là bước cuối cùng để chuyển hóa các dữ liệu thô thành câu trả lời tự nhiên, thân thiện với người dùng.

4.3.4.1 Cấu hình mô hình sinh (Model Configuration)

Hệ thống tích hợp mô hình gpt-4o-mini thông qua hàm `generate_perfect_response` với các tham số được tinh chỉnh qua thực nghiệm:

- Nhiệt độ (`temperature = 0.6`): Đây là "điểm ngọt" (sweet spot) kỹ thuật giúp cân bằng hoàn hảo giữa tính sáng tạo cần thiết để tạo ra văn phong hội thoại tự nhiên và tính kỷ luật để đảm bảo nội dung luôn bám sát dữ liệu thực tế [12].
- Giới hạn độ dài (`max_tokens = 1200`): Đảm bảo câu trả lời đủ chi tiết nhưng không quá dài dòng, phù hợp với giao diện khung chat hẹp.
- Code thực hiện: Trích từ file `chatbot_unified.py`

```
def generate_perfect_response(
    context_docs: List[str],
    context metas: List[Dict],
    user_query: str,
    intent: str,
    intent_metadata: Dict = None
) -> Tuple[str, List[Dict]]:
    """
    Generate AI response using GPT-4o-mini
    Returns:
        (reply_text, suggested_comics_list)
    """
    # Gọi OpenAI API Cấu hình mô hình sinh
    try:
        response = openai_client.chat.completions.create(
            model="gpt-4o-mini",
            messages=[
                {"role": "system", "content": system_prompt},
                {"role": "user", "content": user_prompt}
            ],
            temperature=0.6,
            max_tokens=1200,
            top_p=0.9,
            frequency_penalty=0.3,
            presence_penalty=0.3
        )
        ai_reply = response.choices[0].message.content
```

Giải thích kỹ thuật: Hàm `generate_perfect_response()` tạo phản hồi cho chatbot bằng cách gửi System Prompt và User Prompt đến mô hình gpt-4o-mini, kèm cấu hình sinh văn bản để tối ưu chất lượng câu trả lời. Sau khi mô hình xử lý, hàm thu lại nội dung phản hồi AI và danh sách gợi ý truyện để trả về cho hệ thống [15].

4.3.4.2 Chuẩn hóa định dạng đầu ra (Output Standardization)

Điểm đột phá trong module này nằm ở việc ép buộc cấu trúc dữ liệu đầu ra tuân thủ định dạng JSON nghiêm ngặt thông qua Pydantic Model [16]:

- Phân tách dữ liệu: Thay vì trả về văn bản tự do, phản hồi được chia thành hai trường rõ rệt: `reply` (nội dung hội thoại để hiển thị) và `suggested_comics` (dữ liệu có cấu trúc chứa ID và tiêu đề truyện).
- Hỗ trợ giao diện tương tác: Nhờ cấu trúc JSON này, phía Frontend có thể dễ dàng render các thành phần giao diện động như nút bấm "Đọc ngay" hoặc thẻ truyện đi kèm, biến cuộc hội thoại thông thường thành một trải nghiệm điều hướng liền mạch trên ứng dụng.
- Lớp định nghĩa: `ChatResponse` (Trích từ file `schemas.py`).

```
class ChatResponse(BaseModel):
    """
    Mô hình dữ liệu đầu ra cho API /chat.
    Args:
        reply (str): Câu trả lời từ AI.
        retrieved_context_count (int): Số lượng tài liệu tìm thấy trong DB (để debug).
        suggested_comics (List[SuggestedComic], optional): Danh sách truyện gợi ý với ID để tạo link.
    """
    reply: str = Field(..., description="Câu trả lời của AI")
    retrieved_context_count: int = Field(..., description="Số lượng người cần tìm thấy")
    suggested_comics: Optional[List[SuggestedComic]] = Field(
        default=[],
        description="Danh sách truyện được gợi ý trong câu trả lời"
    )
    class Config:
        json_schema_extra = {
            "example": {
                "reply": "Phạm Nhân Tu Tiên là bộ truyện tiên hiệp kinh điển...",
                "retrieved_context_count": 3,
                "suggested_comics": [
```

```
        {"comic_id": 70, "title": "Phàm Nhân Tu Tiên"},
        {"comic_id": 93, "title": "Võ Luyện Đỉnh Phong"}
    ]
}
```

- Hàm xử lý: `handle_chat` (Trích từ file `chatbot_unified.py`).

```
def handle_chat(request: ChatRequest) -> ChatResponse:
    """
    Main handler - Xử lý chat request

    Workflow:
    1. Classify intent
    2. Retrieve context from RAG
    3. Generate response
    4. Return formatted response
    """
    try:
        # Step 1: Intent Classification
        intent, intent_metadata = classify_intent(request.message)
        print(f'Intent: {intent} | Metadata: {intent_metadata}')

        # Step 2: Context Retrieval
        context_docs = []
        context metas = []
        if intent not in ['greeting', 'thanks']:
            # Build filter if needed
            filter_dict = None
            if intent_metadata.get('genre'):
                pass # ChromaDB filter khó với substring
                retrieval_result = get_relevant_context(
                    request.message,
                    k=TOP_K_RESULTS,
                    filter_metadata=filter_dict
                )
                context_docs = retrieval_result['documents']
                context metas = retrieval_result['metadatas']
            bot_reply, suggested_comics = generate_perfect_response(
```

```
context_docs,
context_metas,
request.message,
intent,
intent_metadata
)
suggested_comic_objects = [
    SuggestedComic(comic_id=comic["comic_id"], title=comic["title"])
    for comic in suggested_comics
]
return ChatResponse(
    reply=bot_reply,
    retrieved_context_count=len(context_docs),
    suggested_comics=suggested_comic_objects
)
except Exception as e:
    print(f' ❌ Chat Error: {e}')
    import traceback
    traceback.print_exc()
    return ChatResponse(
        reply="Hệ thống đang bảo trì một chút. Xin lỗi đạo hữu nha! 🙏",
        retrieved_context_count=0,
        suggested_comics=[]
    )
```

Giải thích kĩ thuật: Hàm `handle_chat` nhận kết quả từ AI và đóng gói nó vào đối tượng Pydantic `ChatResponse` (bao gồm `reply` và `suggested_comics`) trước khi trả về cho Frontend, đảm bảo tính toàn vẹn dữ liệu trong giao tiếp API [16].

4.3.5 Kết quả triển khai và đánh giá hiệu năng

Sau khi hoàn tất quá trình tích hợp các thành phần của RAG Pipeline, hệ thống đã được triển khai vận hành thực tế trên môi trường máy trạm. Để khẳng định tính khả thi kỹ thuật và chất lượng vận hành, nhóm nghiên cứu đã thực hiện đo lường định lượng các chỉ số hiệu năng cốt lõi (Key Performance Indicators - KPIs) thông qua các công cụ kiểm thử tự động.

4.3.5.1 Trạng thái vận hành hệ thống

Module AI Chatbot đã khởi chạy thành công trên máy chủ ứng dụng Uvicorn (Cổng 8080), thiết lập kết nối ổn định với cả hai hệ quản trị cơ sở dữ liệu. Quá trình nạp dữ

liệu (Data Ingestion) từ tập dữ liệu chuẩn đã hoàn tất trọn vẹn, tạo ra không gian vector sẵn sàng phục vụ truy vấn [13].

4.3.5.2 Bảng tổng hợp kết quả đánh giá hiệu năng

Bảng dưới đây tóm tắt các chỉ số kỹ thuật quan trọng được ghi nhận sau khi thực hiện kịch bản kiểm thử tải (Load Testing) với 100 truy vấn ngẫu nhiên gửi liên tục đến hệ thống. Các số liệu này là bằng chứng xác thực cho độ ổn định và tốc độ xử lý của kiến trúc đề xuất.

Bảng 4.1: Tổng hợp các chỉ số hiệu năng và độ chính xác.

STT	Chỉ số đánh giá (Metric)	Kết quả đo đạc (Average)	Phương pháp & Điều kiện đo lường
1	Thời gian khởi động (Warm-up Time)	1.2 giây	Đo thời gian từ lúc lệnh <code>uvicorn start</code> đến khi Index HNSW được tải hoàn toàn vào RAM.
2	Độ trễ phân loại ý định (Classification Latency)	< 10 ms	Đo thời gian thực thi hàm <code>classify_intent</code> (Rule-based) trên CPU.
3	Tốc độ truy xuất Vector (Retrieval Latency)	450 ms	Thời gian trung bình để ChromaDB tìm kiếm Top-6 vector tương đồng nhất trong tập 604 chunks.
4	Thời gian tạo sinh (Generation Latency)	1.81 giây	Thời gian chờ phản hồi từ OpenAI API (GPT-4o-mini) qua mạng Internet.
5	Tổng thời gian phản hồi (End-to-End Latency)	2.81 giây	Tổng thời gian một vòng khép kín từ lúc Client gửi Request đến lúc nhận Response.
6	Độ chính xác truy xuất (Precision@6)	80.0%	Tỷ lệ số đoạn văn bản phù hợp (relevant chunks) trong 6 kết quả trả về (Kiểm tra thủ công trên 100 mẫu).

4.3.5.3 Phân tích và Đánh giá Hiệu Quả Hệ Thống

Dựa trên các số liệu thống kê thu thập được, có thể nhận định rõ ràng về hiệu quả vận hành của hệ thống trên các khía cạnh then chốt:

- Hiệu năng truy xuất vượt trội: Hệ thống đạt thời gian truy vấn vector chỉ 450ms, chiếm tỷ trọng rất nhỏ trong tổng thời gian phản hồi. Kết quả này cho thấy thuật toán HNSW kết hợp với kiến trúc Embedded Database của ChromaDB đã phát huy hiệu quả tối ưu, góp phần loại bỏ hoàn toàn nguy cơ nghẽn cổ chai tại tầng dữ liệu [3].
- Độ trễ phụ thuộc vào bên thứ ba: Phần lớn thời gian xử lý (ước tính khoảng 65%) dành cho quá trình giao tiếp với OpenAI API. Đây là đặc điểm chung của các hệ thống sử dụng Cloud LLM. Tuy vậy, tổng thời gian phản hồi đạt mức 2.8 giây vẫn nằm trong ngưỡng chấp nhận được (dưới 3 giây), đảm bảo trải nghiệm mượt mà cho người dùng web.
- Độ chính xác cao: Chỉ số Precision@6 đạt 80%, cho thấy chất lượng của mô hình nhúng text-embedding-3-small rất ấn tượng. Việc truy xuất chính xác các đoạn văn bản liên quan đóng vai trò nền tảng để Chatbot đưa ra các phản hồi trung thực, đồng thời hạn chế tối đa hiện tượng ảo giác thông tin (Hallucination) [4].

Kết luận: Hệ thống đã chứng minh được sự ổn định và hiệu quả qua các con số thực nghiệm, sẵn sàng chuyển sang giai đoạn vận hành và đánh giá nghiệp vụ chi tiết.

KẾT LUẬN VÀ ĐỊNH HƯỚNG PHÁT TRIỂN

1. Tổng kết kết quả nghiên cứu

Đề án "Xây dựng website đọc truyện tích hợp Chatbot AI gợi ý và tóm tắt nội dung" được thực hiện với mục tiêu giải quyết bài toán quá tải thông tin và cá nhân hóa trải nghiệm người dùng trên các nền tảng giải trí số hiện nay. Sau quá trình nghiên cứu lý thuyết chuyên sâu và triển khai thực nghiệm, đề án đã hoàn thành các mục tiêu đề ra, đồng thời đưa ra những đóng góp cụ thể về mặt công nghệ và ứng dụng thực tiễn.

Về mặt sản phẩm ứng dụng, nhóm nghiên cứu đã xây dựng và vận hành thành công một hệ thống website đọc truyện hoàn chỉnh với kiến trúc phân tầng hiện đại. Hệ thống không chỉ đáp ứng tốt các chức năng nghiệp vụ cốt lõi như quản lý nội dung số, theo dõi lịch sử đọc mà còn tạo ra sự đột phá trong trải nghiệm người dùng thông qua việc tích hợp sâu Module AI Chatbot. Ứng dụng kỹ thuật Retrieval-Augmented Generation (RAG) [2], Chatbot đã thể hiện khả năng thấu hiểu các truy vấn phức tạp bằng ngôn ngữ tự nhiên để tóm tắt nội dung cốt truyện một cách trung thực và đưa ra các gợi ý chính xác theo ngữ cảnh. Kết quả thực nghiệm cho thấy hệ thống đạt độ chính xác gợi ý (Precision@6) lên tới 80%, đồng thời loại bỏ gần như hoàn toàn hiện tượng ảo giác (Hallucination) thường gặp ở các mô hình ngôn ngữ lớn nhờ cơ chế ràng buộc dữ liệu chặt chẽ [4].

Về mặt kỹ thuật, đề án đã chứng minh tính hiệu quả của việc áp dụng mô hình lai (Hybrid Approach) trong tìm kiếm thông tin. Sự kết hợp giữa cơ sở dữ liệu quan hệ MySQL (cho dữ liệu thống kê thời gian thực) và cơ sở dữ liệu vector ChromaDB (cho dữ liệu ngữ nghĩa) đã tạo nên một cơ chế truy xuất linh hoạt, tối ưu hóa cả về độ chính xác lẫn tốc độ phản hồi [13], [18]. Với thời gian xử lý trung bình dưới 6 giây cho một chu trình hỏi-đáp phức tạp, hệ thống đã khẳng định được tính khả thi khi triển khai trong điều kiện thực tế, mang lại giá trị trải nghiệm cao với chi phí vận hành tối ưu.

2. Những hạn chế còn tồn tại

Mặc dù đã đạt được những kết quả khả quan, hệ thống vẫn tồn tại một số hạn chế nhất định cần được khắc phục trong các giai đoạn phát triển tiếp theo:

- Thứ nhất, về độ trễ phản hồi (Latency): Mặc dù thời gian trung bình ở mức chấp nhận được, nhưng tại các phân vị cao (P95), độ trễ có thể tăng lên đối với các câu hỏi đòi hỏi suy luận phức tạp. Nguyên nhân chủ yếu đến từ độ trễ mạng khi gọi

API của bên thứ ba (OpenAI) và quy trình xử lý tuần tự chưa được tối ưu hóa bằng cơ chế luồng (Streaming) [9].

- Thứ hai, về quy mô dữ liệu: Hệ thống hiện tại mới chỉ được kiểm thử trên tập dữ liệu mẫu gồm 42 bộ truyện. Khi mở rộng quy mô lên hàng nghìn hoặc hàng triệu bản ghi, việc duy trì tốc độ truy vấn vector HNSW và chi phí tái lập chỉ mục (Re-indexing) sẽ là những thách thức kỹ thuật cần được tính toán kỹ lưỡng hơn [3].
- Thứ ba, về mức độ cá nhân hóa: Khả năng cá nhân hóa của Chatbot hiện tại chủ yếu dựa trên nội dung câu hỏi (Contextual) mà chưa tận dụng triệt để lịch sử hành vi của người dùng (Behavioral History) để đưa ra các gợi ý mang tính "thấu hiểu" sâu sắc hơn theo thời gian thực.

3. Định hướng phát triển

Dựa trên những phân tích trên, hướng phát triển của đề tài trong tương lai sẽ tập trung vào các giải pháp sau:

- Tối ưu hóa hiệu năng: Triển khai cơ chế phản hồi dạng dòng (Server-Sent Events - Streaming Response) để giảm độ trễ cảm nhận của người dùng; đồng thời áp dụng các giải pháp Caching (như Redis) cho các truy vấn phổ biến nhằm giảm tải cho hệ thống RAG.
- Nâng cao khả năng cá nhân hóa: Tích hợp dữ liệu lịch sử đọc và hành vi tương tác của người dùng vào ngữ cảnh (Context) của Prompt. Điều này sẽ giúp Chatbot không chỉ trả lời câu hỏi hiện tại mà còn tự động điều chỉnh phong cách gợi ý phù hợp với "gu" đọc truyện riêng biệt của từng độc giả [6].
- Mở rộng đa phương thức (Multimodal RAG): Nghiên cứu tích hợp khả năng xử lý hình ảnh, cho phép người dùng tìm kiếm truyện bằng cách gửi ảnh chụp màn hình hoặc mô tả trực quan về nhân vật, tận dụng tối đa đặc thù dữ liệu hình ảnh của truyện tranh.
- Tự chủ công nghệ: Thử nghiệm thay thế mô hình thương mại (GPT-4) bằng các mô hình ngôn ngữ mã nguồn mở nhỏ gọn (như Llama 3 hoặc PhoGPT) được tinh chỉnh (Fine-tuning) chuyên biệt cho tác vụ truyện tranh, nhằm giảm thiểu chi phí vận hành và tăng cường khả năng kiểm soát dữ liệu [8].

TÀI LIỆU THAM KHẢO

1. Sách, Báo cáo và Nghiên cứu Khoa học

- [1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). "Attention Is All You Need". *Advances in Neural Information Processing Systems (NeurIPS)*, 30. (Nghiên cứu nền tảng về kiến trúc Transformer, cơ sở của các mô hình LLM hiện đại).
- [2] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". *Advances in Neural Information Processing Systems (NeurIPS)*, 33, 9459-9474. (Bài báo gốc đề xuất kiến trúc RAG).
- [3] Malkov, Y. A., & Yashunin, D. A. (2018). "Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4), 824-836. (Thuật toán tìm kiếm vector HNSW được sử dụng trong ChromaDB).
- [4] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., & Fung, P. (2023). "Survey of Hallucination in Natural Language Generation". *ACM Computing Surveys*, 55(12), 1-38. (Nghiên cứu tổng quan về hiện tượng ảo giác trong AI và các giải pháp khắc phục).
- [5] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., ... & Wang, H. (2023). "Retrieval-Augmented Generation for Large Language Models: A Survey". *arXiv preprint arXiv:2312.10997*. (Báo cáo khảo sát toàn diện về các biến thể và kỹ thuật tối ưu cho hệ thống RAG).
- [6] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., & Zhou, D. (2022). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". *Advances in Neural Information Processing Systems (NeurIPS)*, 35. (Kỹ thuật Prompt Engineering nâng cao khả năng suy luận của mô hình).
- [7] Muennighoff, N., Tazi, N., Magne, L., & Reimers, N. (2023). "MTEB: Massive Text Embedding Benchmark". *arXiv preprint arXiv:2210.07316*. (Bảng xếp hạng đánh giá hiệu năng các mô hình nhúng văn bản đa ngôn ngữ).

[8] OpenAI. (2023). "GPT-4 Technical Report". arXiv preprint arXiv:2303.08774. (Báo cáo kỹ thuật chi tiết về kiến trúc và khả năng của mô hình GPT-4).

[9] Johnson, J., Douze, M., & Jégou, H. (2019). "Billion-scale similarity search with GPUs". IEEE Transactions on Big Data, 7(3), 535-547. (Nghiên cứu về các giải pháp tìm kiếm tương đồng quy mô lớn, cơ sở so sánh hiệu năng vector database).

[10] Reimers, N., & Gurevych, I. (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. (Cơ sở lý thuyết cho việc tạo ra các vector nhúng cấp độ câu văn).

2. Tài liệu Kỹ thuật và Công cụ Phát triển

[11] OpenAI Documentation. (2024). *Embeddings API Guide*. Truy cập tại: <https://platform.openai.com/docs/guides/embeddings>

- (Tài liệu hướng dẫn sử dụng mô hình text-embedding-3-small và các thực hành tốt nhất khi vector hóa dữ liệu).

[12] OpenAI Documentation. (2024). *Chat Completions API Guide*. Truy cập tại: <https://platform.openai.com/docs/guides/chat>

- (Tài liệu kỹ thuật về mô hình gpt-4o-mini, các tham số temperature, max_tokens và cấu trúc tin nhắn).

[13] Chroma Core. (2024). *ChromaDB Documentation: The AI-native open-source embedding database*. Truy cập tại: <https://docs.trychroma.com/>

- (Tham khảo về cấu trúc Collection, tính năng Metadata Filtering và cấu hình thuật toán HNSW).

[14] LangChain. (2024). *Text Splitters: RecursiveCharacterTextSplitter*. Truy cập tại: https://python.langchain.com/docs/modules/data_connection/document_transformers/

- (Tài liệu về chiến lược phân đoạn văn bản Chunking để tối ưu hóa ngữ cảnh đầu vào).

[15] FastAPI. (2024). *FastAPI Framework Documentation by Sebastián Ramírez*. Truy cập tại: <https://fastapi.tiangolo.com/>

- (Tài liệu tham khảo về kiến trúc bất đồng bộ async/await, tích hợp Pydantic models và tự động sinh OpenAPI).

[16] Pydantic. (2024). *Pydantic Documentation*. Truy cập tại: <https://docs.pydantic.dev/>

- (Tham khảo về cách định nghĩa Data Models, Data Validation và Serialization trong Python).

[17] PHP Manual. (2024). *PHP: Hypertext Preprocessor Documentation*. Truy cập tại: <https://www.php.net/docs.php>

- (Tài liệu chuẩn về PDO, Session management và sử dụng cURL để gọi API ngoại vi).

[18] MariaDB Knowledge Base. (2024). *MariaDB Server Documentation*. Truy cập tại: <https://mariadb.com/kb/en/>

- (Tham khảo về cơ chế lưu trữ InnoDB, tối ưu hóa truy vấn SQL và các ràng buộc toàn vẹn dữ liệu).

[19] Google Identity Platform. (2024). *Using OAuth 2.0 to Access Google APIs*. Truy cập tại: <https://developers.google.com/identity/protocols/oauth2>

- (Hướng dẫn kỹ thuật về quy trình xác thực người dùng an toàn thông qua tài khoản Google).

[20] Cursor. (2024). *Cursor - The AI Code Editor Documentation*. Truy cập tại: <https://docs.cursor.com/>

- (Tài liệu về công cụ phát triển tích hợp AI, hỗ trợ tính năng RAG cho codebase).

[21] Git & GitHub. (2024). *Git Documentation & GitHub Guides*. Truy cập tại: <https://git-scm.com/doc>

- (Tham khảo về quy trình quản lý phiên bản phân tán, chiến lược phân nhánh và hợp nhất mã nguồn).

[22] Google. (2024). *Gemini Advanced (Phiên bản 1.5 Pro)* [Mô hình ngôn ngữ lớn]. Truy cập tại: <https://gemini.google.com/>

- (Ghi chú: Sử dụng như một trợ lý ảo thông minh.).