

**BỘ GIÁO DỤC ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM**



ĐỒ ÁN TỐT NGHIỆP

XÂY DỰNG ỨNG DỤNG ĐỌC TRUYỆN TRANH TRÊN NỀN TẢNG ASP.NET CORE WEB API VÀ VUEJS

SINH VIÊN THỰC HIỆN : PHẠM QUANG THANH

MÃ SINH VIÊN

1451020209

KHOA

: CÔNG NGHỆ THÔNG TIN

HÀ NỘI - 2024

**BỘ GIÁO DỤC ĐÀO TẠO
TRƯỜNG ĐẠI HỌC ĐẠI NAM**



PHẠM QUANG THANH

**XÂY DỰNG ỨNG DỤNG ĐỌC TRUYỆN
TRANH TRÊN NỀN TẢNG ASP.NET CORE
WEB API VÀ VUEJS**

CHUYÊN NGÀNH : CÔNG NGHỆ THÔNG TIN

MÃ SỐ : 74.80.201

NGƯỜI HƯỚNG DẪN : TS. TRẦN QUÝ NAM

HÀ NỘI - 2024

LỜI CAM ĐOAN

Em xin cam đoan rằng toàn bộ nội dung của báo cáo thực tập tốt nghiệp này là kết quả của công sức nghiên cứu và làm việc của chính em dưới sự hướng dẫn của thầy Trần Quý Nam. Mọi thông tin, dữ liệu, và kết quả được trình bày trong báo cáo đều là trung thực và chưa từng được công bố trước đây ở bất kỳ nơi nào khác. Em không sao chép hoặc tham khảo ý kiến của bất kỳ công trình nào khác mà không được ghi rõ nguồn gốc.

Thứ Hai, ngày 17 tháng 06 năm 2024

Sinh viên

LỜI CẢM ƠN

Em xin bày tỏ lòng biết ơn sâu sắc đến tất cả những cá nhân và tổ chức đã hỗ trợ và đồng hành cùng em trong suốt quãng thời gian này. Em muốn gửi lời cảm ơn đặc biệt đến thầy Trần Quý Nam, người đã dành thời gian và công sức để hướng dẫn, hỗ trợ em trong việc hoàn thành báo cáo thực tập với sự kiên nhẫn và sự chỉ bảo không ngừng. Cũng xin gửi lời tri ân đến quý thầy cô trong Khoa Công Nghệ Thông Tin - Đại Học Đại Nam, người đã luôn dành sự quan tâm và sự hướng dẫn tận tình cho em.

Dù em nhận thức được rằng bản thân còn nhiều hạn chế về kiến thức chuyên môn và thời gian, và có thể gặp phải những sai sót trong quá trình thực hiện, nhưng em hy vọng nhận được sự góp ý và phản hồi tích cực từ tất cả để bài báo cáo của em được hoàn thiện và cải thiện hơn.

Em xin chân thành cảm ơn!

LỜI NÓI ĐẦU

Trong thời đại số hóa hiện nay, việc phát triển ứng dụng đọc truyện tranh trên nền tảng ASP.NET CORE Web API và VueJS không chỉ là một dự án công nghệ mà còn là một bước tiến tiến về trải nghiệm người dùng và quản lý nội dung truyện tranh.

Ứng dụng này không chỉ giúp người dùng dễ dàng truy cập và đọc các tác phẩm truyện tranh một cách thuận tiện, mà còn cung cấp một giao diện tương tác linh hoạt và thân thiện. Nó là một không gian sáng tạo, kết nối giữa thế giới truyện tranh và công nghệ, giúp người dùng thưởng thức và khám phá thế giới truyện tranh một cách tiện lợi và thú vị.

Dự án này không chỉ tập trung vào việc xây dựng một ứng dụng đọc truyện tranh thông thường, mà còn đặt trọng điểm vào việc quản lý nội dung, tối ưu hóa trải nghiệm người dùng và tương tác linh hoạt với cộng đồng yêu thích truyện tranh. Hãy cùng nhau khám phá và trải nghiệm hành trình của dự án này, nơi mà sự kết hợp giữa công nghệ và truyện tranh tạo nên một không gian độc đáo và thú vị cho mọi người.

[illegible]

DANH MỤC VIẾT TẮT

STT	Kí hiệu viết tắt	Chữ viết đầy đủ
1	AI	Artificial Intelligence
2	BA	User Interface/User Experience
3	IDE	Integrated Development Environment
4	UI/UX	Business Analyst
5

MỤC LỤC

CHƯƠNG 1: KHÁI QUÁT ĐỀ TÀI.....	1
1.1. Lý do chọn đề tài	1
1.2. Mục tiêu nghiên cứu	2
1.3. Phạm vi nghiên cứu	3
1.3.1. Công nghệ.....	3
1.3.2. Đối tượng người dùng	4
1.3.3. Thời gian thực hiện.....	4
1.4. Phương pháp nghiên cứu	4
1.4.1. Nghiên cứu tài liệu.....	4
1.4.2. Phân tích yêu cầu	5
1.4.3. Thiết kế hệ thống.....	5
1.4.4. Phát triển và kiểm thử	5
1.4.5. Triển khai và bảo trì	6
1.5. Các thành phần quan trọng trong xây dựng đề tài.....	6
1.6. Bố cục đồ án	7
CHƯƠNG 2: TỔNG QUAN VỀ ASP.NET CORE WEB API VÀ VUE.JS	12
2.1. Tổng quan về ASP.NET CORE WEB API.....	12
2.1.1. Khái niệm.....	12
2.1.2. Thành phần của ASP.NET CORE WEB API	12
2.1.3. Các loại ASP.NET CORE WEB API phổ biến	14
2.1.4. Hỗ trợ công nghệ và tích hợp CSDL	15
2.1.5. Ưu điểm và thách thức khi sử dụng ASP.NET CORE WEB API.....	16
2.1.6. Lý do chọn ASP.NET Core Web API cho bài toán.....	16

2.2.	Tổng quan về Vue.js.....	17
2.2.1.	Khái niệm.....	17
2.2.2.	Cấu trúc	17
2.2.3.	Tích hợp	18
2.2.4.	Ứng dụng của vuejs	18
2.2.5.	Ưu và nhược điểm của Vue.js.....	20
2.2.6.	Lý do chọn Vue.js cho bài toán	22
CHƯƠNG 3: ĐẶC TẢ YÊU CẦU VÀ THIẾT KẾ HỆ THỐNG.....		24
3.1.	Đặc tả yêu cầu	24
3.1.1.	Mô tả bài toán.....	24
3.1.2.	Yêu cầu chức năng.....	24
3.1.3.	Các yêu cầu phi chức năng	26
3.2.	Thiết kế hệ thống	26
3.2.1.	Thiết kế CSDL.....	27
3.2.2.	Đặc tả yêu cầu phần mềm	31
3.2.3.	Biểu đồ phân tích ca sử dụng	43
3.2.4.	Biểu đồ trạng thái.....	43
3.2.5.	Biểu đồ tuần tự	43
3.2.6.	Biểu đồ hoạt động của use case/ hệ thống/ phương thức	44
3.2.7.	Biểu đồ phân cấp chức năng (Functional Decomposition Diagram - FDD).....	45
CHƯƠNG 4: XÂY DỰNG CHƯƠNG TRÌNH VÀ ĐÁNH GIÁ		47
4.1.	Xây dựng BackEnd.....	47
4.1.1.	Khởi tạo Project	47

4.1.2.	<i>Khởi tạo Folder Models</i>	48
4.1.3.	<i>Khởi tạo class ShopDbContext, add-migration</i>	49
4.1.4.	<i>Khởi tạo Folder Repositories, Services</i>	49
4.2.	<i>Xây dựng FrontEnd</i>	52
4.2.1.	<i>Trang đăng nhập, đăng ký và Profile</i>	52
4.2.2.	<i>Trang Quản lý</i>	55
4.2.3.	<i>Trang xem comics</i>	56
4.3.	<i>Đánh giá</i>	58
4.3.1.	<i>Kết quả đạt được</i>	58
4.3.2.	<i>Ưu điểm</i>	58
4.4.3.	<i>Nhược điểm</i>	59
4.4.4.	<i>Tính Năng Mở Rộng</i>	59
KẾT LUẬN		60
TÀI LIỆU THAM KHẢO		61

Danh Mục Bảng

STT	Mô Tả
Bảng 3.1.2	Yêu cầu chức năng
Bảng 3.1.3	Yêu cầu phi chức năng
Bảng 3.2.1	Bảng Images
Bảng 3.2.1	Bảng Comments
Bảng 3.2.1	Bảng Chapter
Bảng 3.2.1	Bảng Genre
Bảng 3.2.1	Bảng GenreComic
Bảng 3.2.1	Bảng Account
Bảng 3.2.1	Bảng Profile
Bảng 3.2.1	Bảng Like
Bảng 3.2.1	Bảng Role
Bảng 3.2.1	Bảng Comic

Danh Mục Hình Ảnh

STT	Mô Tả
Hình 3.2.1.1	Biểu đồ tổng quát trang quản trị
Hình 3.2.1.1	Biểu đồ phân rã của use case Xem comics
Bảng 3.2.1	Biểu đồ phân rã của use case Quản lý likes comics
Bảng 3.2.1	Biểu đồ phân rã của use case Quản lý comics
Bảng 3.2.1	Biểu đồ phân rã của use case Quản lý chapters.
Bảng 3.2.1	Biểu đồ phân rã của use case Quản lý lượt xem
Bảng 3.2.1	Biểu đồ phân rã của use case Quản lý images.
Hình 3.2.1.2	biểu đồ lớp thực thể
Hình 3.2.3	biểu đồ phân tích
Hình 3.2.4	biểu đồ trạng thái
Hình 3.2.5	biểu đồ tuần tự use case đăng nhập.
Hình 3.2.5	biểu đồ tuần tự use case đăng xuất.
Hình 3.2.6	Biểu đồ hoạt động của use case đăng nhập.
Hình 3.2.6	Biểu đồ hoạt động của use case thêm comics
Hình 3.2.7	Biểu đồ phân cấp
Hình 4.1.1	Khởi tạo project
Hình 4.1.1	Sau khi tạo project
Hình 4.1.2	Khởi tạo Folder Models
Hình 4.1.2	Domain
Hình 4.1.2	DTO
Hình 4.1.3	Context
Hình 4.1.3	Migrations
Hình 4.1.4	Khởi tạo Responsitory
Hình 4.1.4	Service
Hình 4.1.4	Mapping
Hình 4.1.4	Controlllers
Hình 4.2.1	Trang login
Hình 4.2.1	Trang sign up
Hình 4.2.1	Trang Profile
Hình 4.2.2	Trang quản lý
Hình 4.2.3	Trang quản lý CategoryComic
Hình 4.2.4	Trang chủ
Hình 4.2.4	Chi tiết Comic
Hình 4.2.4	Chi tiết Image

CHƯƠNG 1: KHÁI QUÁT ĐỀ TÀI

1.1. Lý do chọn đề tài

Trong xã hội hiện đại ngày nay thì những vấn đề, nhu cầu về mặt giải trí đối với con người là một điều tất yếu mà ai cũng có để giải tỏa, xả stress cho bản thân sau những giờ làm việc mệt mỏi, sau những lúc học hành căng thẳng. Có nhiều hình thức cho mọi người có thể lựa chọn để có thể giải quyết vấn đề đó, sự lựa chọn của mỗi người là khác nhau dựa trên sở thích cũng như nhu cầu của từng người. Nhưng kết quả là đều đều mang đến sự thoải mái trong tâm trí và thư giãn đầu óc.

Với việc internet được phủ sóng toàn cầu thì việc sử dụng mạng xã hội để tìm kiếm những niềm vui là điều rất dễ dàng và tiện lợi. Với đề án lần này, tôi tập trung vào cộng đồng người đam mê đọc truyện tranh nói chung cũng như bản thân tôi nói riêng. Sự quan tâm này của bản thân đã thúc đẩy tôi đến việc nghiên cứu về đề tài “Xây dựng ứng dụng đọc truyện tranh trên nền tảng ASP.NET CORE Web API và VueJS” và tìm hiểu những cách giải quyết về những thách thức mà nó mang lại cho bản thân.

Truyện tranh là cái tên không còn xa lạ gì đối với đại đa số mọi người cả ở trong nước Việt Nam hay ngoài nước. Truyện tranh bắt đầu phổ biến từ cuối thế kỷ 19 và tiếp tục phát triển lan rộng ra toàn cầu vào thế kỷ 20 và 21, đem lại ảnh hưởng lớn đối với nền văn hóa giải trí. Không chỉ qua sách giấy, mà người đọc cũng có thể truy cập vào nhiều nguồn truyện tranh trực tuyến thông qua internet. Các trang web truyện tranh và ứng dụng di động cung cấp hàng ngàn tiêu đề truyện tranh từ nhiều thể loại khác nhau, từ hành động, phiêu lưu, tình cảm đến kinh dị, khoa học viễn tưởng và thần thoại.

Khả năng tương tác của internet cũng đã thúc đẩy sự phát triển của cộng đồng truyện tranh trực tuyến. Người đọc có thể thảo luận về truyện tranh yêu thích của họ, chia sẻ ý kiến và tạo ra các nội dung liên quan thông qua các diễn đàn, trang mạng xã hội và các nền tảng truyện tranh khác.

Bên cạnh đó, internet cũng đã tạo điều kiện thuận lợi cho các tác giả và họa sĩ truyện tranh mới nổi bật. Nhờ vào sự phổ biến của các trang web tự xuất bản và các dịch vụ chia sẻ nội dung, họ có thể chia sẻ tác phẩm của mình với cộng đồng một cách dễ dàng và nhanh chóng hơn. Với vai trò ngày càng quan trọng trong văn hóa giải trí, truyện tranh tiếp tục là một phần không thể thiếu của cuộc sống hiện đại, đem lại niềm vui và giải trí cho mọi người.

1.2. Mục tiêu nghiên cứu

Mục tiêu chính của dự án là phát triển một ứng dụng đọc truyện tranh trên nền tảng ASP.NET CORE Web API và VueJS nhằm cung cấp cho người dùng một trải nghiệm đọc truyện trực tuyến tốt nhất. Chúng tôi đặt ra các mục tiêu cụ thể như sau:

Xây dựng giao diện người dùng thân thiện và dễ sử dụng: Mục tiêu này nhằm tạo ra một trang web trực quan, có khả năng tương tác cao, giúp người dùng dễ dàng tìm kiếm và đọc truyện tranh một cách thuận lợi.

Phát triển các chức năng tìm kiếm và lọc thông tin phong phú: Chúng tôi muốn người dùng có thể tìm kiếm và lựa chọn truyện theo nhiều tiêu chí khác nhau như thể loại, tác giả, rating, v.v., từ đó tạo ra trải nghiệm đọc truyện cá nhân hóa và linh hoạt.

Hiển thị thông tin chi tiết và hình ảnh đầy đủ về từng truyện tranh: Mục tiêu này giúp cung cấp cho người dùng cái nhìn toàn diện và rõ ràng về từng truyện, giúp họ có thể đưa ra quyết định mua hàng thông tin và chi tiết.

Tích hợp tính năng đánh giá và nhận xét: Chúng tôi muốn tạo ra một cộng đồng đọc truyện tranh tích cực và đa dạng bằng cách cho phép người dùng chia sẻ ý kiến và đánh giá về các truyện tranh.

Thêm tính năng yêu thích: Chúng tôi sẽ phát triển tính năng cho phép người dùng đánh dấu truyện tranh yêu thích của họ. Điều này giúp họ dễ dàng quản lý và truy cập lại các truyện mà họ quan tâm mà không cần phải tìm kiếm lại từ đầu.

Lưu trữ lịch sử đọc: Chúng tôi cũng sẽ tích hợp tính năng lưu trữ lịch sử đọc của người dùng. Điều này cho phép họ tiếp tục đọc từ nơi họ đã dừng lại trước đó, cũng như theo dõi và quản lý các truyện họ đã đọc trong quá khứ. Điều này tăng cường trải nghiệm đọc truyện và giúp người dùng tiết kiệm thời gian trong việc tìm kiếm lại các truyện đã đọc trước đó.

Xây dựng hệ thống quản lý linh hoạt và dễ mở rộng: Mục tiêu này nhằm đảm bảo tích hợp các tính năng mới và cập nhật thông tin một cách hiệu quả, đồng thời đảm bảo ổn định và linh hoạt cho ứng dụng trong tương lai.

1.3. Phạm vi nghiên cứu

1.3.1. Công nghệ

ASP.NET Core: Sử dụng ASP.NET Core để phát triển backend, bao gồm các dịch vụ Web API để quản lý dữ liệu truyện tranh, người dùng, và các chức năng liên quan.

Vue.js: Sử dụng Vue.js để phát triển frontend, tạo ra giao diện người dùng thân thiện, tương tác mượt mà và hiệu suất cao.

Cơ sở dữ liệu: Sử dụng SQL Server để lưu trữ dữ liệu truyện tranh, thông tin người dùng và các dữ liệu liên quan khác. Entity Framework sẽ được sử dụng để thao tác với cơ sở dữ liệu.

Các công nghệ hỗ trợ khác:

Bootstrap: Sử dụng Bootstrap để thiết kế giao diện người dùng, đảm bảo tính thẩm mỹ và khả năng tương thích với nhiều loại thiết bị khác nhau.

JWT (JSON Web Token): Sử dụng JWT để xác thực và phân quyền người dùng, đảm bảo an toàn cho các giao dịch giữa client và server.

Azure hoặc AWS: Sử dụng các dịch vụ đám mây như Azure hoặc AWS để triển khai ứng dụng, đảm bảo khả năng mở rộng và độ tin cậy.

1.3.2. Đối tượng người dùng

Ứng dụng hướng tới mọi đối tượng người dùng yêu thích truyện tranh, từ trẻ em đến người lớn. Đặc biệt chú trọng vào việc tạo ra một giao diện thân thiện và dễ sử dụng để thu hút người dùng ở mọi lứa tuổi.

1.3.3. Thời gian thực hiện

Đề tài dự kiến được thực hiện trong vòng 2 tháng, với các giai đoạn chính sau:

Giai đoạn 1: Nghiên cứu và phân tích yêu cầu

Giai đoạn 2: Thiết kế hệ thống

Giai đoạn 3: Phát triển ứng dụng

Giai đoạn 4: Kiểm thử và tinh chỉnh

Giai đoạn 5: Triển khai và bảo trì

1.4. Phương pháp nghiên cứu

1.4.1. Nghiên cứu tài liệu

Tài liệu về ASP.NET Core và Web API: Nghiên cứu các tài liệu, sách vở, và hướng dẫn trực tuyến về ASP.NET Core và cách xây dựng Web API để có cái nhìn tổng quan và nắm vững các kỹ thuật cần thiết.

Tài liệu về Vue.js: Tìm hiểu các khái niệm cơ bản và nâng cao của Vue.js, thông qua các sách, tài liệu hướng dẫn và khóa học trực tuyến.

Tài liệu về cơ sở dữ liệu và Entity Framework: Nghiên cứu cách thiết kế và quản lý cơ sở dữ liệu, cũng như cách sử dụng Entity Framework để tương tác với cơ sở dữ liệu.

1.4.2. Phân tích yêu cầu

Khảo sát người dùng: Tiến hành khảo sát người dùng tiềm năng để hiểu rõ nhu cầu, mong muốn và các vấn đề họ gặp phải khi sử dụng các ứng dụng đọc truyện tranh hiện có.

Phân tích chức năng: Xác định và phân tích các chức năng cần thiết của ứng dụng, đảm bảo đáp ứng đúng nhu cầu của người dùng.

1.4.3. Thiết kế hệ thống

Thiết kế kiến trúc hệ thống: Xác định kiến trúc tổng thể của hệ thống, bao gồm việc phân chia các thành phần backend và frontend, các giao thức giao tiếp giữa các thành phần.

Thiết kế cơ sở dữ liệu: Xác định các bảng, mối quan hệ giữa các bảng và các ràng buộc cần thiết để đảm bảo tính nhất quán và toàn vẹn của dữ liệu.

Thiết kế giao diện người dùng: Sử dụng các công cụ thiết kế giao diện để tạo ra các mockup, wireframe, và prototype, đảm bảo giao diện người dùng trực quan và dễ sử dụng.

1.4.4. Phát triển và kiểm thử

Phát triển từng phần: Thực hiện phát triển ứng dụng theo từng phần nhỏ, bắt đầu từ các chức năng cơ bản đến các chức năng nâng cao. Sử dụng các phương pháp lập trình Agile để quản lý quá trình phát triển.

Kiểm thử đơn vị: Tiến hành kiểm thử đơn vị (unit testing) để đảm bảo từng thành phần nhỏ của ứng dụng hoạt động đúng như mong đợi.

Kiểm thử tích hợp: Kiểm thử tích hợp (integration testing) để đảm bảo các thành phần khác nhau của hệ thống hoạt động cùng nhau một cách mượt mà.

Kiểm thử chức năng: Tiến hành kiểm thử chức năng (functional testing) để đảm bảo ứng dụng đáp ứng đúng các yêu cầu chức năng đã đề ra.

Kiểm thử hiệu năng: Kiểm thử hiệu năng (performance testing) để đảm bảo ứng dụng hoạt động ổn định và có thể xử lý được lượng lớn người dùng đồng thời.

1.4.5. Triển khai và bảo trì

Triển khai ứng dụng: Triển khai ứng dụng lên môi trường thực tế, sử dụng các dịch vụ đám mây như Azure hoặc AWS để đảm bảo tính khả dụng và khả năng mở rộng.

Bảo trì và cập nhật: Tiến hành bảo trì, cập nhật và cải tiến ứng dụng dựa trên phản hồi của người dùng và các vấn đề phát sinh trong quá trình sử dụng thực tế.

1.5. Các thành phần quan trọng trong xây dựng đề tài

Trong việc xây dựng đề tài về ứng dụng đọc truyện tranh trên nền tảng ASP.NET Core Web API và VueJS, có một số thành phần quan trọng cần được chú ý:

- ❖ ASP.NET Core Web API: Đây là phần backend của ứng dụng, được sử dụng để xử lý các yêu cầu từ phía client, quản lý dữ liệu và tương tác với cơ sở dữ liệu.
- ❖ VueJS: Đây là một framework JavaScript cho phép xây dựng giao diện người dùng tương tác. VueJS sẽ được sử dụng trong phần frontend của ứng dụng để hiển thị giao diện người dùng và tương tác với backend thông qua các API đã được cung cấp.
- ❖ Cơ sở dữ liệu: Là nơi lưu trữ thông tin về người dùng, truyện tranh, chương và các dữ liệu khác liên quan đến ứng dụng. Cơ sở dữ liệu có thể được thiết kế dựa trên nhu cầu cụ thể của ứng dụng, ví dụ như sử dụng SQL Server, MySQL, hoặc MongoDB.
- ❖ Xác thực và Phân quyền: Các tính năng xác thực và phân quyền đảm bảo rằng chỉ người dùng được ủy quyền mới có thể truy cập vào các chức năng như thêm, sửa, xóa truyện tranh hoặc quản lý tài khoản cá nhân. Điều này thường

được thực hiện thông qua việc sử dụng JWT (JSON Web Tokens) hoặc các phương thức xác thực khác.

- ❖ **Giao diện người dùng (UI/UX):** Phần giao diện người dùng cần được thiết kế sao cho dễ sử dụng và hấp dẫn để thu hút người dùng. Sự tương tác mượt mà và tính thẩm mỹ của giao diện cũng là yếu tố quan trọng trong việc xây dựng một ứng dụng thành công.
- ❖ **Quản lý trạng thái (State Management):** Đặc biệt quan trọng trong các ứng dụng phức tạp, quản lý trạng thái giúp theo dõi và điều khiển dữ liệu trên toàn bộ ứng dụng. Trong VueJS, Vuex có thể được sử dụng để quản lý trạng thái của ứng dụng.
- ❖ **Tối ưu hóa hiệu suất (Performance Optimization):** Đảm bảo ứng dụng hoạt động mượt mà và nhanh chóng là một yếu tố quan trọng. Tối ưu hóa cả backend và frontend để giảm thiểu thời gian tải và tăng trải nghiệm người dùng.
- ❖ **Kiểm thử (Testing):** Thực hiện các bài kiểm thử đơn vị, kiểm thử tích hợp và kiểm thử chấp nhận người dùng để đảm bảo tính ổn định và chất lượng của ứng dụng.

1.6. Bố cục đồ án

Chương 1: khái quát đề tài

1.1. Lý do chọn đề tài

1.2. Mục tiêu nghiên cứu

1.3 Phạm vi nghiên cứu

1.3.1 Công nghệ

1.3.2 Tính năng

1.3.3 Đối tượng người dùng

1.3.4 Thời gian thực hiện

1.4 Phương pháp nghiên cứu

1.4.1 Nghiên cứu tài liệu

1.4.2 Phân tích yêu cầu

1.4.3 Thiết kế hệ thống

1.4.4 Phát triển và kiểm thử

1.4.5 Triển khai và bảo trì

1.5. Các thành phần quan trọng trong xây dựng đề tài

1.6. Bố cục đồ án

Chương 2: tổng quan về asp.net core web api và vue.js

2.1. Tổng quan về ASP.NET CORE WEB API

2.1.1. Khái niệm

2.1.2. Thành phần của ASP.NET CORE WEB API

2.1.3. Các loại ASP.NET CORE WEB API phổ biến

2.1.4. Ưu điểm và thách thức khi sử dụng ASP.NET CORE WEB API

2.2. Tổng quan về Vue.js

2.2.1. Khái niệm

2.2.2. Ưu và nhược điểm của Vue.js

Chương 3: đặc tả yêu cầu và thiết kế hệ thống

3.1. Đặc tả yêu cầu

3.1.1. Mô tả bài toán

3.1.2. Yêu cầu chức năng

3.1.3. Các yêu cầu phi chức năng

3.2. Thiết kế hệ thống

3.2.1. Thiết kế CSDL

3.1.1. Bảng Images

3.1.2. Bảng Comments

3.1.3. Bảng Chapters

3.1.4. Bảng Genres

3.1.5. Bảng GenreComics

3.1.6. Bảng Account

3.1.7. Bảng Profile

3.1.8. Bảng Likes

3.1.9. Bảng Role

3.1.10. Bảng Comics

3.2.2. Đặc tả yêu cầu phần mềm

3.2.1.1. Các yêu cầu chức năng

3.2.2.2. Các đặc tả ca sử dụng

3.2.2.3. Biểu đồ lớp thực thể

3.2.3. Biểu đồ phân tích ca sử dụng

3.2.4. Biểu đồ trạng thái

3.2.5. Biểu đồ tuần tự

3.2.6. Biểu đồ hoạt động của use case/ hệ thống/ phương thức

Chương 4: xây dựng chương trình và đánh giá

4.1. Xây dựng BackEnd

4.1.1. Khởi tạo Project

4.1.2. Khởi tạo Folder Models

4.1.3. Khởi tạo class ShopDbContext, add-migration

4.1.4. Khởi tạo Folder Repositories, Services

4.2. Xây dựng FrontEnd

4.2.2. Trang Quản lý

4.2.4. Trang xem Comics

4.2.5. Trang chủ, chi tiết

4.3. Đánh giá

4.3.1. Kết quả đạt được

4.3.2. Ưu điểm

4.3.3. Nhược điểm

Kết luận

Tài liệu tham khảo

Tiểu kết:

Chương 1 đã trình bày tổng quan về đề tài "Xây dựng ứng dụng đọc truyện tranh trên nền tảng ASP.NET Core Web API và Vue.js". Nội dung chương bao gồm việc giới thiệu lý do chọn đề tài, mục tiêu nghiên cứu, phạm vi nghiên cứu và phương pháp nghiên cứu.

Đề tài tập trung vào việc phát triển một ứng dụng đọc truyện tranh trực tuyến, đáp ứng nhu cầu giải trí ngày càng cao của người dùng. Ứng dụng này sẽ sử dụng công nghệ ASP.NET Core để xây dựng backend và Vue.js để phát triển frontend, nhằm tạo ra một hệ thống hiện đại, hiệu suất cao và bảo mật.

Mục tiêu của đề tài là cung cấp các chức năng cơ bản như đăng ký, đăng nhập, đọc truyện, tìm kiếm truyện, quản lý danh sách truyện yêu thích và lưu trữ lịch sử đọc truyện. Phạm vi nghiên cứu bao gồm việc áp dụng các công nghệ tiên

tiền, đảm bảo tính năng đáp ứng yêu cầu của người dùng, và thời gian thực hiện trong vòng 6 tháng.

Phương pháp nghiên cứu bao gồm việc nghiên cứu tài liệu, phân tích yêu cầu người dùng, thiết kế hệ thống, phát triển và kiểm thử ứng dụng, cũng như triển khai và bảo trì sau khi hoàn thiện.

Chương này đặt nền tảng vững chắc cho các chương tiếp theo, nơi sẽ đi sâu vào các khía cạnh cụ thể của công nghệ, thiết kế và phát triển hệ thống, đảm bảo ứng dụng được xây dựng theo đúng yêu cầu và mục tiêu đã đề ra.

CHƯƠNG 2: TỔNG QUAN VỀ ASP.NET CORE WEB API VÀ VUE.JS

2.1. Tổng quan về ASP.NET CORE WEB API

2.1.1. Khái niệm

ASP.NET Core Web API là một framework phát triển ứng dụng web dựa trên .NET Core, được thiết kế để cung cấp các công cụ và tính năng cho việc xây dựng và phục vụ các API web RESTful hoặc các dịch vụ web khác trên nền tảng .NET Core.

API, viết tắt của Application Programming Interface, đóng vai trò là giao diện cho việc giao tiếp giữa các ứng dụng. Trong ngữ cảnh của ASP.NET Core Web API, các API này thường được sử dụng để truy cập và thao tác dữ liệu từ một ứng dụng hoặc dịch vụ khác.

Kiến trúc RESTful (Representational State Transfer) được sử dụng rộng rãi trong ASP.NET Core Web API. Đây là một kiến trúc cho phép các hệ thống phân tán giao tiếp với nhau qua giao thức HTTP. ASP.NET Core Web API cho phép các nhà phát triển xây dựng các API RESTful, cho phép các ứng dụng gửi và nhận dữ liệu thông qua các phương thức HTTP như GET, POST, PUT và DELETE.

.NET Core, một framework mã nguồn mở của Microsoft, là nền tảng chính để xây dựng ASP.NET Core Web API. .NET Core cho phép các nhà phát triển xây dựng và triển khai các dịch vụ web trên nhiều nền tảng, bao gồm Windows, Linux và macOS.

ASP.NET Core Web API cung cấp một cách tiện lợi và mạnh mẽ để xây dựng các dịch vụ web linh hoạt và hiệu quả trên nền tảng .NET Core.

2.1.2. Thành phần của ASP.NET CORE WEB API

ASP.NET Core Web API bao gồm các thành phần cốt lõi quan trọng giúp nhà phát triển xây dựng và quản lý các dịch vụ web một cách hiệu quả. Các thành phần chính của ASP.NET Core Web API bao gồm:

- **Controllers (Bộ điều khiển):** Controllers là các lớp trong ASP.NET Core Web API được sử dụng để xử lý các yêu cầu HTTP từ client và phản hồi lại với dữ liệu tương ứng. Mỗi controller có thể chứa nhiều phương thức hành động (action methods) để xử lý các yêu cầu HTTP khác nhau.
- **Routing (Định tuyến):** Routing là quá trình xác định xem yêu cầu HTTP nào sẽ được chuyển đến controller và action method tương ứng. ASP.NET Core Web API cung cấp các cơ chế định tuyến linh hoạt cho phép nhà phát triển xác định các mẫu địa chỉ URL và ánh xạ chúng với các controller và action methods tương ứng.
- **Model Binding (Ràng buộc mô hình):** Model binding là quá trình ánh xạ dữ liệu từ yêu cầu HTTP đến các tham số của action method trong controller. ASP.NET Core Web API cung cấp các cơ chế ràng buộc mô hình tự động để giúp nhà phát triển dễ dàng trích xuất dữ liệu từ yêu cầu và sử dụng chúng trong mã logic.
- **Middleware (Phần mềm trung gian):** Middleware là các thành phần phần mềm được thêm vào pipeline xử lý của ứng dụng ASP.NET Core để thực hiện các chức năng như xác thực, ghi nhật ký, nén dữ liệu và nhiều hơn nữa. Middleware làm cho việc xây dựng các tính năng phức tạp trong ASP.NET Core Web API trở nên linh hoạt và dễ dàng.
- **Filters (Bộ lọc):** Filters là các thành phần được sử dụng để thực hiện các chức năng quản lý tác vụ trước, trong và sau khi action method trong controller được thực thi. Các loại filter bao gồm Authorization Filters, Action Filters, Result Filters và Exception Filters, cho phép nhà phát triển kiểm soát và tùy chỉnh hành vi của các action method.
- **Tóm lại,** các thành phần của ASP.NET Core Web API là các phần tử cốt lõi định nghĩa cách ứng dụng xử lý và phản hồi lại các yêu cầu HTTP từ client. Bằng cách sử dụng các thành phần này một cách linh hoạt và hiệu

quả, nhà phát triển có thể xây dựng các dịch vụ web mạnh mẽ và linh hoạt trên nền tảng .NET Core.

2.1.3. Các loại ASP.NET CORE WEB API phổ biến

ASP.NET Core Web API là một nền tảng phát triển mạnh mẽ cho việc xây dựng các loại API đa dạng, phù hợp với các nhu cầu và yêu cầu khác nhau của ứng dụng web. Dưới đây là một số loại ASP.NET Core Web API phổ biến mà nhà phát triển thường sử dụng:

- RESTful API là một loại API phổ biến, tuân thủ theo các nguyên tắc của kiến trúc Representational State Transfer (REST). Với RESTful API, các phương thức HTTP như GET, POST, PUT và DELETE được sử dụng để thực hiện các hoạt động CRUD trên các tài nguyên.
- GraphQL API là một lựa chọn mạnh mẽ cho việc truy vấn dữ liệu từ Backend. Với GraphQL, client có thể chỉ định chính xác dữ liệu mà họ cần thông qua các truy vấn linh hoạt.
- SOAP API, mặc dù không còn phổ biến như trước, vẫn được sử dụng trong một số dự án. SOAP là một giao thức dùng để truyền dữ liệu giữa các ứng dụng và cung cấp các tiêu chuẩn mạnh mẽ cho việc truyền tin an toàn.
- OData API là một giao thức mở cho phép truy cập vào và sửa đổi dữ liệu thông qua HTTP. Với OData API, client có thể thực hiện các truy vấn phức tạp và linh hoạt đến dữ liệu.
- JSON-RPC API là một giao thức cho phép gọi các hàm hoặc phương thức từ xa thông qua HTTP và truyền tham số dưới dạng JSON. Dựa trên JSON, JSON-RPC API cung cấp cách tiện lợi để giao tiếp giữa Backend và Frontend.trợ.

2.1.4. Hỗ trợ công nghệ và tích hợp CSDL

ASP.NET Core Web API cung cấp sự linh hoạt và tích hợp mạnh mẽ với các công nghệ và cơ sở dữ liệu (CSDL) khác nhau, giúp nhà phát triển dễ dàng xây dựng và quản lý các dịch vụ web. Dưới đây là một số công nghệ và CSDL mà ASP.NET Core Web API hỗ trợ và tích hợp:

Entity Framework Core (EF Core): Entity Framework Core là một ORM (Object-Relational Mapping) framework cho phép nhà phát triển tương tác với cơ sở dữ liệu một cách dễ dàng thông qua các đối tượng .NET. ASP.NET Core Web API tích hợp chặt chẽ với EF Core, cung cấp một cách tiện lợi để thực hiện các thao tác CRUD (Create, Read, Update, Delete) với cơ sở dữ liệu.

ADO.NET: ASP.NET Core Web API cũng hỗ trợ việc sử dụng ADO.NET để tương tác trực tiếp với cơ sở dữ liệu thông qua các câu truy vấn SQL. Dù không được sử dụng phổ biến như EF Core trong các ứng dụng ASP.NET Core mới, nhưng ADO.NET vẫn là một lựa chọn cho các tác vụ tương tác với cơ sở dữ liệu tùy chỉnh hoặc phức tạp hơn.

SQL Server, MySQL, PostgreSQL, SQLite và các loại CSDL khác: ASP.NET Core Web API hỗ trợ tích hợp với nhiều loại CSDL phổ biến như SQL Server, MySQL, PostgreSQL, và SQLite thông qua các provider tương ứng. Điều này cho phép nhà phát triển lựa chọn CSDL phù hợp với yêu cầu cụ thể của dự án.

NoSQL Databases: Ngoài các CSDL quan hệ, ASP.NET Core Web API cũng có khả năng tích hợp với các loại CSDL NoSQL như MongoDB, Redis và Cassandra thông qua các thư viện và công cụ hỗ trợ.

Dependency Injection (DI): ASP.NET Core Web API tích hợp sâu với Dependency Injection (DI), cho phép nhà phát triển dễ dàng quản lý và tiêm các dịch vụ CSDL vào các controller và các thành phần khác của ứng dụng.

2.1.5. Ưu điểm và thách thức khi sử dụng ASP.NET CORE WEB API

ASP.NET Core Web API là một công nghệ mạnh mẽ cho việc phát triển các dịch vụ web, mang lại nhiều ưu điểm và thách thức đặc biệt.

Trong số những ưu điểm nổi bật, ASP.NET Core Web API hỗ trợ đa nền tảng, cho phép triển khai trên nhiều hệ điều hành khác nhau, từ Windows đến Linux và macOS. Điều này tạo ra tính linh hoạt và di động cho quá trình phát triển ứng dụng.

Đồng thời, hiệu suất của ASP.NET Core Web API được tối ưu hóa để đảm bảo ứng dụng hoạt động mượt mà và đáp ứng tốt trong các tải cao. Sự tích hợp sâu với các công nghệ khác của .NET Core như Entity Framework Core, Identity và Dependency Injection cũng giúp việc phát triển ứng dụng trở nên thuận tiện và hiệu quả.

Tuy nhiên, việc sử dụng ASP.NET Core Web API cũng đặt ra một số thách thức. Đầu tiên là sự phức tạp ban đầu khi học và tiếp cận công nghệ này, đặc biệt đối với những người mới bắt đầu hoặc chuyển đổi từ các nền tảng khác. Quản lý tài nguyên cũng là một thách thức, đặc biệt là khi xử lý lượng truy cập lớn hoặc phát triển các ứng dụng phức tạp.

Ngoài ra, việc cập nhật và đảm bảo tính tương thích với các phiên bản mới của .NET Core và ASP.NET Core cũng đòi hỏi sự chú ý và công sức từ phía nhà phát triển.

2.1.6. Lý do chọn ASP.NET Core Web API cho bài toán

- Hiệu suất và độ tin cậy cao: ASP.NET Core Web API cung cấp hiệu suất cao và khả năng xử lý lượng lớn yêu cầu đồng thời, phù hợp với ứng dụng đọc-truyện tranh có thể có lượng người dùng lớn.
- Bảo mật mạnh mẽ: Với các tính năng bảo mật tích hợp sẵn, ASP.NET Core giúp bảo vệ dữ liệu người dùng và thông tin truyện tranh khỏi các nguy cơ bảo mật.

- Hỗ trợ đa nền tảng: Khả năng chạy trên nhiều hệ điều hành giúp dễ dàng triển khai và quản lý ứng dụng trên các môi trường khác nhau.
- Hỗ trợ tốt cho RESTful services: ASP.NET Core Web API hỗ trợ tốt cho việc xây dựng các dịch vụ RESTful, cho phép tương tác dễ dàng giữa client và server.

2.2. Tổng quan về Vue.js

2.2.1. Khái niệm

VueJS là một framework mã nguồn mở của JavaScript được sử dụng để phát triển các giao diện web tương tác. Nó là một trong những framework nổi tiếng được sử dụng để đơn giản hóa việc phát triển web. VueJS tập trung vào view layer. Nó có thể dễ dàng tích hợp vào các dự án lớn để phát triển front-end mà không gặp bất kỳ sự cố nào.

2.2.2. Cấu trúc

Vue.js được xây dựng dựa trên một mô hình thành phần, một trong những điểm mạnh và đặc trưng nhất của nó. Trong mô hình này, giao diện người dùng được chia thành các thành phần độc lập, mỗi thành phần đại diện cho một phần cụ thể của giao diện người dùng. Mỗi thành phần có thể chứa HTML, CSS và JavaScript, cho phép tái sử dụng và tổ chức code một cách hiệu quả. Điều này giúp tăng tính module và dễ bảo trì của ứng dụng. Bằng cách này có thể tập trung vào phát triển từng phần nhỏ một cách độc lập, điều này làm giảm sự phức tạp và tăng tính linh hoạt của ứng dụng.

Vue.js cung cấp các directives để tương tác với DOM và dữ liệu của ứng dụng. Điều này giúp dễ dàng thêm các chức năng động vào giao diện người dùng, như hiển thị điều kiện, lặp qua danh sách và xử lý sự kiện. Bằng cách này có thể kiểm soát và tùy chỉnh giao diện người dùng một cách linh hoạt và hiệu quả.

Vue.js cũng đi kèm với các công cụ hỗ trợ như Vuex và Vue Router để quản lý trạng thái và routing trong ứng dụng. Vuex cung cấp một cách tiếp cận đơn giản và mạnh mẽ để quản lý trạng thái của ứng dụng, trong khi Vue Router cho phép quản lý các route và navigation trong ứng dụng một cách dễ dàng.

2.2.3. Tích hợp

Vue.js tích hợp dễ dàng với các backend frameworks như ASP.NET Core Web API thông qua việc sử dụng Axios. Axios là một thư viện HTTP client mạnh mẽ cho JavaScript, cho phép thực hiện các yêu cầu HTTP từ frontend Vue.js đến các endpoint của ASP.NET Core Web API. Điều này giúp truy cập dữ liệu từ server và hiển thị nó trên giao diện người dùng một cách linh hoạt và hiệu quả.

Việc tích hợp Vue.js với ASP.NET Core Web API mở ra nhiều cơ hội cho việc xây dựng các ứng dụng web động và phong phú. Bằng cách này có thể tận dụng sức mạnh của cả hai framework để tạo ra các ứng dụng mạnh mẽ và linh hoạt.

2.2.4. Ứng dụng của vuejs

2.2.4.1. Phát triển ứng dụng

- Thiết kế cấu trúc:

Xác định yêu cầu của ứng dụng và thiết kế cấu trúc của nó, bao gồm việc xác định các thành phần cần thiết, quản lý trạng thái, routing, và các tính năng khác.

- Lập trình và triển khai thành phần:

Phát triển các thành phần của ứng dụng, bao gồm HTML, CSS, và JavaScript, và triển khai chúng trong ứng dụng Vue.js. Đảm bảo rằng các thành phần hoạt động như mong muốn và đáp ứng được yêu cầu của ứng dụng.

- Quản lý trạng thái:

Sử dụng Vuex hoặc các cách tiếp cận khác để quản lý trạng thái của ứng dụng một cách hiệu quả, đảm bảo rằng dữ liệu được đồng bộ hóa và tự động cập nhật khi cần thiết.

- Routing:

Sử dụng Vue Router để quản lý các route trong ứng dụng, đảm bảo rằng các route được xác định đúng và người dùng có thể điều hướng qua các trang một cách dễ dàng.

2.2.4.2. Triển khai ứng dụng

- Tối ưu hóa mã nguồn:

Kiểm tra và tối ưu hóa mã nguồn của ứng dụng trước khi triển khai, bao gồm việc loại bỏ mã không cần thiết, nén và gộp các tập tin, và tối ưu hóa hình ảnh.

- Triển khai ứng dụng:

Triển khai ứng dụng Vue.js lên môi trường sản phẩm, bao gồm việc sao chép các tập tin cần thiết lên máy chủ và cấu hình các môi trường như production.

- Kiểm tra lại:

Kiểm tra lại ứng dụng trên môi trường triển khai để đảm bảo rằng tất cả các tính năng hoạt động như mong đợi và không có lỗi nào xảy ra.

2.2.4.3. Kiểm thử ứng dụng

- Kiểm thử đơn vị:

Sử dụng các framework kiểm thử đơn vị như Jest hoặc Mocha để kiểm thử các phần riêng lẻ của ứng dụng, bao gồm các thành phần, hàm, và modules.

- Kiểm thử tích hợp:

Kiểm thử tích hợp các thành phần của ứng dụng và đảm bảo rằng chúng hoạt động đúng khi được kết hợp với nhau.

- Kiểm thử giao diện người dùng:

Kiểm thử giao diện người dùng của ứng dụng trên các trình duyệt và thiết bị khác nhau để đảm bảo rằng nó hoạt động một cách đáng tin cậy trên mọi nền tảng.

2.2.4.4. Mở rộng ứng dụng

- Tối ưu hóa hiệu suất:

Tối ưu hóa hiệu suất của ứng dụng bằng cách tối ưu hóa mã nguồn, sử dụng caching và lazy loading, và tối ưu hóa các yêu cầu HTTP.

- Thêm tính năng mới:

Mở rộng ứng dụng bằng cách thêm tính năng mới để cung cấp giá trị cho người dùng, bao gồm việc tương tác với dữ liệu mới và cung cấp trải nghiệm người dùng tốt hơn.

- Tăng cường bảo mật:

Tăng cường bảo mật cho ứng dụng bằng cách thêm các lớp bảo mật mới, kiểm tra và cập nhật các phần mềm bảo mật, và thực hiện các biện pháp bảo mật khác để bảo vệ dữ liệu và người dùng.

2.2.5. Ưu và nhược điểm của Vue.js

Ưu điểm:

- Kích thước nhỏ: Tập zip được tải xuống của framework này chỉ nặng 18 KB. Điều này khiến nó không chỉ cài đặt nhanh mà còn tác động tích cực đến SEO và UX.
- Kết xuất và hiệu suất DOM ảo: Mô hình đối tượng tài liệu (DOM) là thứ có thể gặp phải khi kết xuất các trang web. DOM đại diện cho một trang HTML với các kiểu, thành phần và nội dung dưới dạng cấu trúc cây của các đối tượng (nút). Các đối tượng cây DOM lưu trữ dưới dạng cây và được tạo bởi trình duyệt khi tải trang. Khi người dùng tương tác với trang, các đối tượng sẽ thay đổi trạng thái của chúng, do đó trình duyệt sẽ phải cập nhật thông tin và hiển thị trên màn hình. Tuy nhiên, việc cập nhật toàn bộ DOM rất phức tạp. Ưu tiên tốc độ load, VueJS sử dụng DOM ảo. Hãy coi đây là một bản sao của DOM gốc giúp tìm ra những phần tử cần cập nhật mà không cần kết xuất lại toàn bộ cây nút. Cách tiếp cận này giúp hiển thị trang khá nhanh và cải thiện hiệu suất ứng dụng.
- Hệ thống phản ứng và các tùy chọn ràng buộc dữ liệu: Liên kết dữ liệu là kết nối giữa mô hình dữ liệu (nguồn dữ liệu) và mẫu DOM hoặc HTML của chế độ xem. Liên kết dữ liệu một chiều cho phép thông tin truyền theo một hướng, từ mô hình sang chế độ xem hoặc ngược lại. Trong trường hợp đầu tiên, các thay đổi đối với nguồn sẽ tự động cập nhật DOM, nhưng nó không hoạt động ngược lại vì DOM có quyền truy cập chỉ đọc vào mô hình.
- Liên kết dữ liệu hai chiều cho phép trao đổi dữ liệu giữa mô hình và chế độ xem theo cả hai hướng. Nói cách khác, mô hình cũng lắng nghe các sự kiện trên DOM và bất kỳ cập nhật nào ở một bên sẽ phản ánh ngay lập tức ở bên kia. Cách tiếp cận này loại bỏ mã soạn sẵn và đơn giản hóa việc phát triển ứng dụng. Tuy nhiên, việc khó Debug và dễ xảy ra lỗi khiến luồng hai chiều không phù hợp cho các dự án lớn.

Nhược điểm:

- Thiếu hỗ trợ cho các dự án quy mô lớn: Quy mô nhóm phát triển và cộng đồng của VueJS vẫn không thể so sánh với Angular hay React. Framework này cũng không được hỗ trợ tài chính từ các doanh nghiệp lớn. Để được áp dụng trong các dự án quy mô lớn, công nghệ phải ổn định và được hỗ trợ mạnh mẽ để các vấn đề có thể được giải quyết nhanh chóng. Mặc dù VueJS không gặp nhiều vấn đề và thậm chí còn có nhu cầu đến từ các doanh nghiệp như IBM và Adobe, nhưng nó chủ yếu được sử dụng trong các dự án tương đối nhỏ.
- Nguy cơ đến từ việc quá linh hoạt: Tính linh hoạt là một đặc tính gây tranh cãi của một dự án lớn. Cung cấp cho nhóm phát triển quá nhiều tùy chọn có thể dẫn đến các cách tiếp cận lập trình khác nhau trong một nhóm. Và kết quả là, nó trở thành một công cụ vô hiệu hóa cuối cùng thay vì một phần mềm hoạt động.
- Nguồn tài nguyên giới hạn: Mặc dù hệ sinh thái khá rộng và có tất cả các công cụ cần thiết để bắt đầu phát triển với VueJS, nhưng framework này không lớn bằng React hay Angular. Nói chính xác hơn, chỉ cần so sánh số lượng plugin có sẵn cho React và Vue.js, sự khác biệt là ở hàng trăm đơn vị. Các plugin hiện có có thể được sử dụng với các framework khác cũng thường không được hỗ trợ.

2.2.6. Lý do chọn Vue.js cho bài toán

Độ linh hoạt và dễ dàng tích hợp:

Vue.js cho phép tích hợp dễ dàng với ASP.NET Core Web API thông qua việc sử dụng Axios, giúp truy cập và quản lý dữ liệu từ server một cách hiệu quả.

Hiệu suất cao và khả năng tùy chỉnh:

Vue.js có hiệu suất cao và cho phép tùy chỉnh linh hoạt theo nhu cầu của ứng dụng, giúp tối ưu hóa trải nghiệm người dùng.

Cộng đồng lớn và hỗ trợ mạnh mẽ:

Vue.js có một cộng đồng lớn và hỗ trợ mạnh mẽ từ cộng đồng, điều này giúp giải quyết các vấn đề và tìm kiếm giải pháp một cách nhanh chóng.

Dễ học và sử dụng:

Vue.js có một cú pháp rõ ràng và dễ hiểu, giúp nhà phát triển mới bắt đầu có thể nhanh chóng làm quen và phát triển ứng dụng một cách hiệu quả.

Tiểu kết:

Trong chương này, giới thiệu về hai công nghệ quan trọng: ASP.NET Core Web API và Vue.js. ASP.NET Core Web API là một framework mạnh mẽ được thiết kế để xây dựng các dịch vụ web RESTful trên nền tảng .NET Core. Đã tìm hiểu về khái niệm, các tính năng cơ bản, cách hoạt động, các thành phần chính, cũng như ưu điểm và thách thức khi sử dụng ASP.NET Core Web API. Đã thấy lý do chọn ASP.NET Core Web API cho dự án của mình, như khả năng hiệu suất cao, tính bảo mật mạnh mẽ, khả năng hỗ trợ đa nền tảng và khả năng tương thích tốt với các dịch vụ RESTful.

Tiếp theo, khám phá Vue.js - một framework JavaScript phổ biến cho việc xây dựng giao diện người dùng tương tác trên web. Tìm hiểu về khái niệm, ưu và nhược điểm của Vue.js. Vue.js là một công cụ dễ học và sử dụng, có tính linh hoạt cao và hiệu suất tốt, tuy nhiên, nó cũng đối mặt với một số thách thức như hệ sinh thái nhỏ hơn so với các framework khác.

CHƯƠNG 3: ĐẶC TẢ YÊU CẦU VÀ THIẾT KẾ HỆ THỐNG

3.1. Đặc tả yêu cầu

3.1.1. Mô tả bài toán

Đề tài này tập trung vào việc phát triển 1 ứng dụng web đọc truyện tranh nhằm hướng đến cộng đồng yêu thích đọc truyện tranh, truyện chữ hay cả anime và các bộ phim hoạt hình khác được chuyển thể từ truyện tranh. Ứng dụng web được thiết kế cho cả bên trang người dùng lẫn bên quản trị viên, đáp ứng đầy đủ yêu cầu các vai trò.

Mục tiêu chính của đề tài là xây dựng lên 1 ứng dụng web tiện ích, giao diện ưa nhìn cho người dùng. Trong đó sẽ có các phần có kiếm doanh thu cho quản trị viên để duy trì web và lợi nhuận về cho bản thân. Với các tính năng quản lý toàn diện từ truyện, các chương, ảnh, hồ sơ, bình luận, lượt thích và quản lý người dùng. Vai trò, thao tác được tích hợp vào web để đảm bảo quá trình quản lý được thực hiện 1 cách hiệu quả và chính xác.

3.1.2. Yêu cầu chức năng

Bảng 3.1.2. Yêu cầu chức năng

Loại yêu cầu	Mô tả
Quản lý tài khoản	* Tạo tài khoản mới * Đăng nhập và đăng xuất * Thay đổi thông tin tài khoản * Xóa tài khoản
Quản lý truyện tranh	* Thêm truyện tranh mới * Cập nhật thông tin truyện tranh * Xóa truyện tranh * Danh sách truyện tranh * Chi tiết truyện tranh * Tìm kiếm truyện tranh
Quản lý chương	* Thêm chương mới * Cập nhật thông tin chương * Xóa chương * Danh sách chương * Chi tiết chương

Quản lý hình ảnh	* Thêm hình ảnh mới * Cập nhật thông tin hình ảnh * Xóa hình ảnh
Quản lý bình luận	* Thêm bình luận mới * Cập nhật thông tin bình luận * Xóa bình luận * Danh sách bình luận
Quản lý thể loại	* Thêm thể loại mới * Cập nhật thông tin thể loại * Xóa thể loại * Danh sách thể loại
Quản lý lượt thích	* Thêm lượt thích * Xóa lượt thích * Danh sách lượt thích
Quản lý vai trò	* Thêm vai trò mới * Cập nhật thông tin vai trò * Xóa vai trò * Danh sách vai trò
Quản lý quyền hạn	* Gán vai trò cho người dùng * Cập nhật quyền hạn vai trò
Quản lý hồ sơ	* Cập nhật thông tin hồ sơ
Quản lý tài khoản	* Tạo tài khoản mới * Đăng nhập và đăng xuất * Thay đổi thông tin tài khoản * Xóa tài khoản
Quản lý truyện tranh	* Thêm truyện tranh mới * Cập nhật thông tin truyện tranh * Xóa truyện tranh * Danh sách truyện tranh * Chi tiết truyện tranh * Tìm kiếm truyện tranh

3.1.3. Các yêu cầu phi chức năng

Bảng 3.1.3. Yêu cầu phi chức năng

Loại yêu cầu	Mô tả
Hiệu suất	* Hệ thống phải có thể xử lý đồng thời nhiều truy cập từ người dùng. * Thời gian phản hồi của hệ thống phải nhanh chóng.
Bảo mật	* Dữ liệu người dùng phải được bảo mật an toàn. * Hệ thống phải có các biện pháp chống tấn công và truy cập trái phép.
Khả dụng	* Hệ thống phải hoạt động liên tục 24/7. * Thời gian ngừng hoạt động của hệ thống phải được giảm thiểu.
Khả năng bảo trì	* Hệ thống phải dễ dàng bảo trì và nâng cấp. * Mã nguồn của hệ thống phải dễ hiểu và dễ sửa đổi.
Khả năng mở rộng	* Hệ thống phải có thể mở rộng để đáp ứng nhu cầu sử dụng ngày càng tăng. * Hệ thống phải có thể tích hợp với các hệ thống khác.
Khả năng sử dụng	* Giao diện người dùng phải đơn giản và dễ sử dụng. * Hệ thống phải dễ dàng học hỏi và sử dụng.
Tính tương thích	* Hệ thống phải tương thích với nhiều trình duyệt web và thiết bị khác nhau.

3.2. Thiết kế hệ thống

3.2.1. Thiết kế CSDL

Bảng 3.2.1. Bảng Images

Cột	Kiểu dữ liệu	Mô tả
Imageid	INT	Khóa chính, Mã định danh duy nhất cho mỗi hình ảnh
Chapterid	INT	Khóa ngoại, Mã định danh cho chương mà hình ảnh thuộc về
Name	VARCHAR(255)	Tên của hình ảnh
Path	VARCHAR(255)	Đường dẫn đến hình ảnh
Rawword	TEXT	Văn bản thô của hình ảnh (được sử dụng cho mục đích tìm kiếm)
CommentedAt	DATETIME	Thời gian hình ảnh được bình luận

Bảng 3.2.1. Bảng Comments

Cột	Kiểu dữ liệu	Mô tả
Commentid	INT	Khóa chính, Mã định danh duy nhất cho mỗi bình luận
Imageid	INT	Khóa ngoại, Mã định danh cho hình ảnh mà bình luận thuộc về

Comicid	INT	Khóa ngoại, Mã định danh cho truyện tranh mà hình ảnh thuộc về
Accountid	INT	Mã định danh cho tài khoản đã đăng bình luận
CommentText	TEXT	Văn bản của bình luận
CommentedAt	DATETIME	Thời gian bình luận được đăng

Bảng 3.2.1. Bảng Chapter

Cột	Kiểu dữ liệu	Mô tả
Chapterid	INT	Mã định danh duy nhất cho mỗi chương
Comicid	INT	Khóa ngoại, Mã định danh cho truyện tranh mà chương thuộc về
ChapterlNumber	INT	Số chương
ChapterTibe	VARCHAR(255)	Tiêu đề chương
Created Date	DATETIME	Ngày tạo chương

Bảng 3.2.1. Bảng Genre

Cột	Kiểu dữ liệu	Mô tả
Genreid	INT	Mã định danh duy nhất cho mỗi thể loại

Name	VARCHAR(255)	Tên thể loại
------	--------------	--------------

Bảng 3.2.1. Bảng GenreComic

Cột	Kiểu dữ liệu	Mô tả
Genreid	INT	Mã định danh cho thể loại
Comicid	INT	Mã định danh cho truyện tranh

Bảng 3.2.1. Bảng Account

Cột	Kiểu dữ liệu	Mô tả
accountid	INT	Khóa chính, mã số duy nhất cho mỗi tài khoản
username	VARCHAR(255)	Tên người dùng
email	VARCHAR(255)	Địa chỉ email
password	VARCHAR(255)	Mật khẩu

Bảng 3.2.1. Bảng Profile

Cột	Kiểu dữ liệu	Mô tả
Profield	INT	Khóa chính, Mã định danh duy nhất cho mỗi hồ sơ
Accountid	INT	Khóa ngoại, Mã định danh cho tài khoản mà hồ sơ thuộc về
avatar	VARCHAR(255)	Ảnh đại diện

FirstName	VARCHAR(255)	Tên
LastName	VARCHAR(255)	Họ
Birthday	DATE	Ngày sinh
Gender	VARCHAR(255)	Giới tính
Address	VARCHAR(255)	Địa chỉ

Bảng 3.2.1. Bảng Like

Cột	Kiểu dữ liệu	Mô tả
Likeld	INT	Khóa chính, Mã định danh duy nhất cho mỗi lượt thích
Comicid	INT	Khóa ngoại, Mã định danh cho truyện tranh được thích
Accountid	INT	Khóa ngoại, Mã định danh cho tài khoản đã thích truyện tranh

Bảng 3.2.1. Bảng Role

Cột	Dữ liệu	Mô tả
roleid	INT	Khóa chính, mã số duy nhất cho mỗi vai trò
Accountid	INT	Khóa ngoại, tham chiếu đến bảng Roles.
rolename	VARCHAR(255)	Tên vai trò

Bảng 3.2.1. Bảng Comic

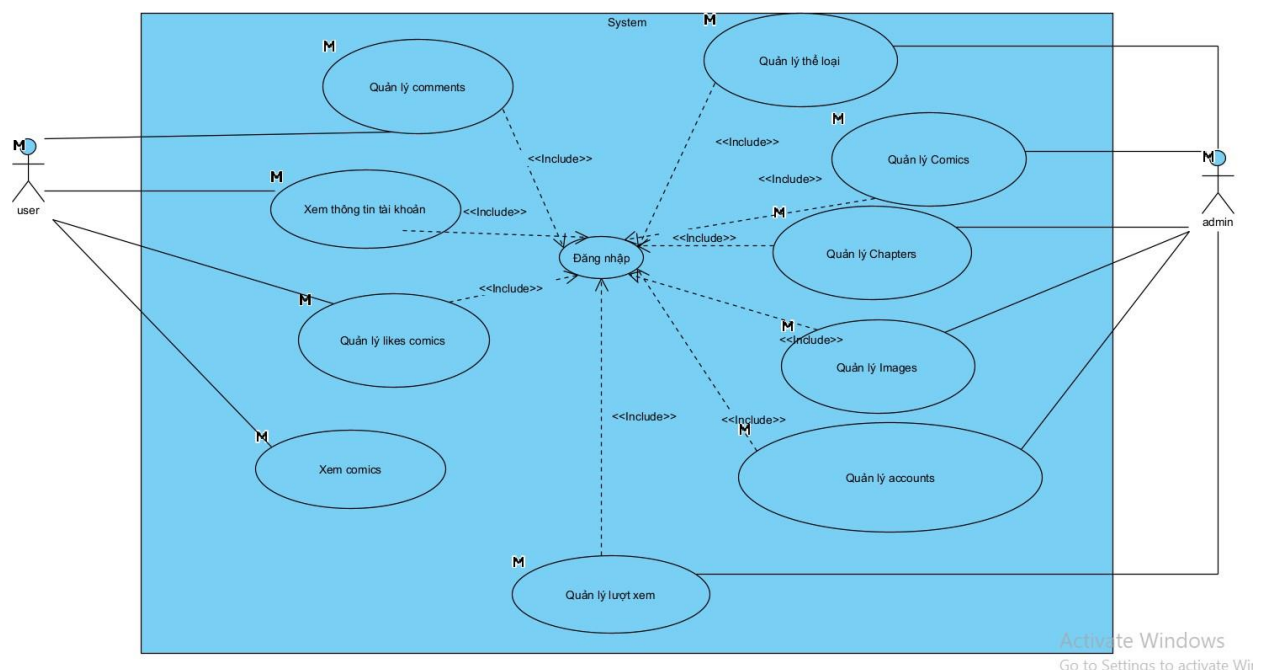
Cột	Dữ liệu	Mô tả
comiciid	INT	Khóa chính, mã số duy nhất cho mỗi truyện tranh
comiciist	VARCHAR(255)	Tên tác giả
description	TEXT	Mô tả truyện tranh
coverimage	VARCHAR(255)	Ảnh bìa truyện tranh
country	VARCHAR(255)	Quốc gia

3.2.2. Đặc tả yêu cầu phần mềm

3.2.1.1. Các yêu cầu chức năng

Các yêu cầu chức năng Trang quản trị.

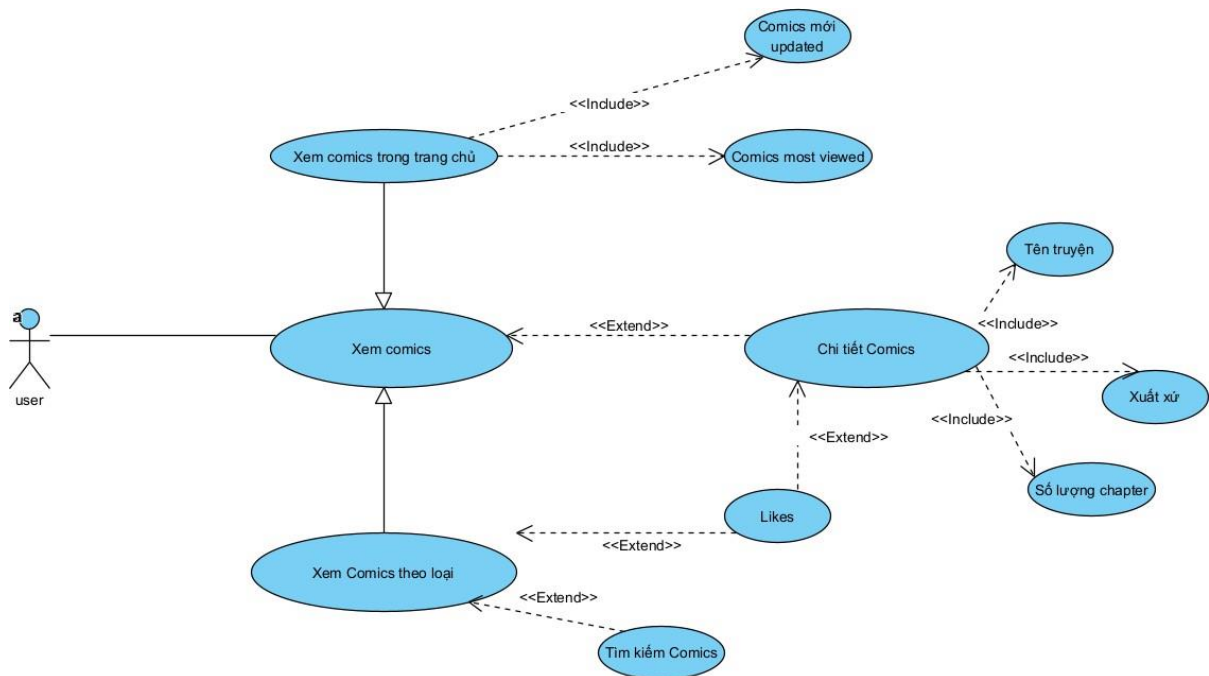
- Biểu đồ tổng quát.



Hình 3.2.1.1: Biểu đồ tổng quát trang quản trị

Biểu đồ tổng quát trang quản trị gồm các chức năng chính như sau: Các chức năng ở phía người quản lý gồm Quản lý thể loại, Quản lý comics, Quản lý chapters, quản lý images, quản lý accounts và quản lý lượt xem.. Các chức năng phía người dùng như quản lý comics đã likes, xem thông tin tài khoản, xem comics, comments. Tất cả các chức năng này trừ xem comics đều yêu cầu phải đăng nhập.

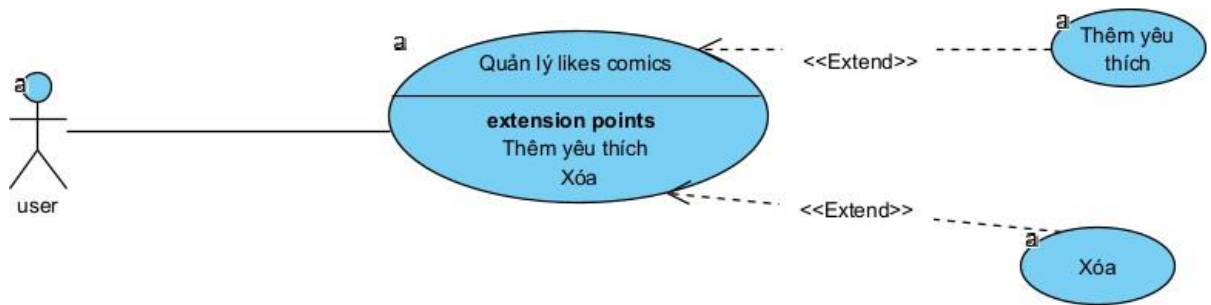
- Biểu đồ phân rã của use case Xem comics.



Hình 3.2.1.1: Biểu đồ phân rã của use case Xem comics

Biểu đồ 2 thể hiện biểu đồ phân rã của use case xem comics, gồm các use case phân rã là: Xem tại trang chủ, xem theo loại, xem chi tiết,...

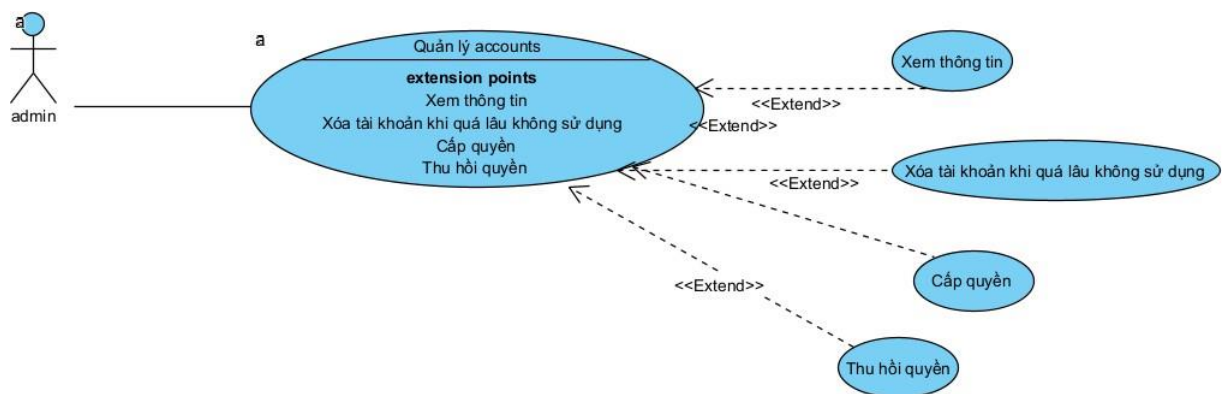
- Biểu đồ phân rã của use case likes comics.



Hình 3.2.1.1: Biểu đồ phân rã của use case Quản lý likes comics

Biểu đồ 3 thể hiện biểu đồ phân rã của use case likes comics, gồm các use case phân rã là: thêm yêu thích, xóa.

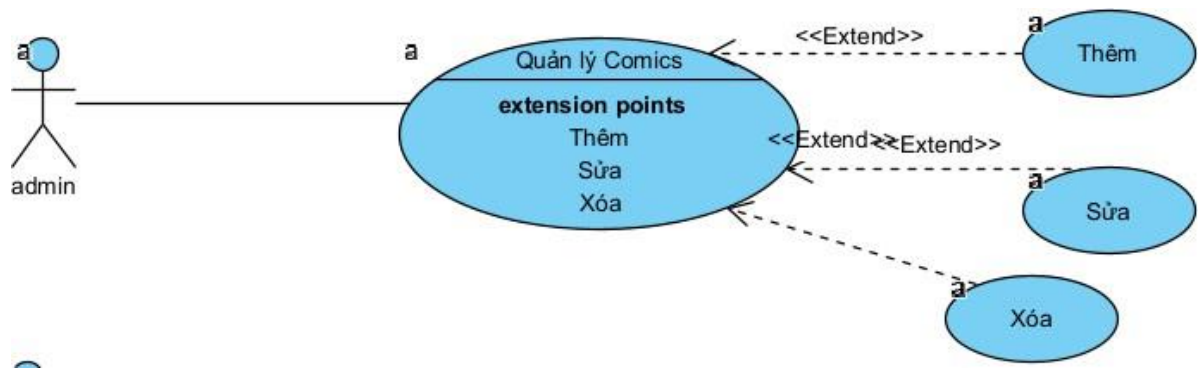
- Biểu đồ phân rã của use case Quản lý accounts.



Hình 3.2.1.1: Biểu đồ phân rã của use case Quản lý đơn hàng

Biểu đồ 4 thể hiện biểu đồ phân rã của use case Quản lý accounts, gồm các use case phân rã là: xem thông tin, xóa tài khoản khi lâu không sử dụng, cấp quyền, thu hồi quyền.

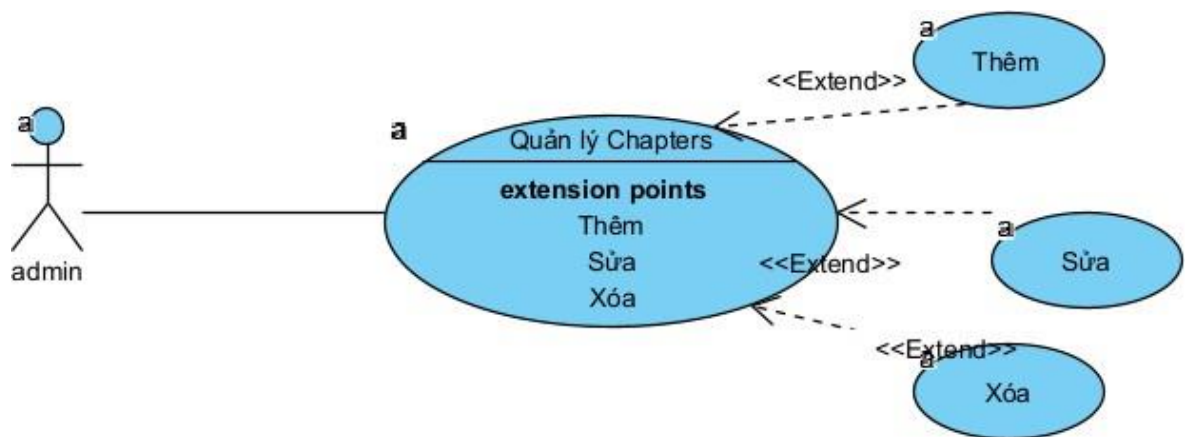
- Biểu đồ phân rã của use case Quản lý comics.



Hình 3.2.1.1: Biểu đồ phân rã của use case Quản lý comics

Biểu đồ 5 thể hiện biểu đồ phân rã của use case Quản lý comics gồm các use case phân rã là: Nhập thông tin comics, Sửa thông tin comics, Xóa thông tin comics.

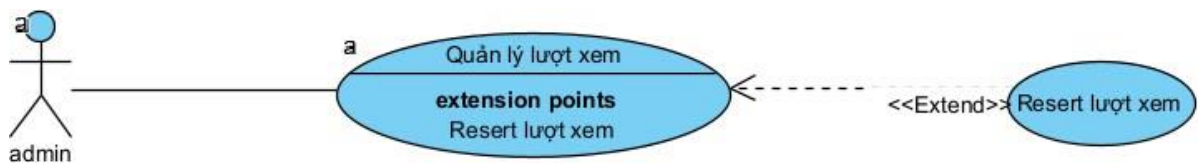
- Biểu đồ phân rã của use case Quản lý chapters.



Hình 3.2.1.1: Biểu đồ phân rã của use case Quản lý chapters.

Biểu đồ 6 thể hiện biểu đồ phân rã của use case Quản lý chapters gồm các use case phân rã là: Nhập thông tin chapters với ngày được up lên tự động, Sửa thông tin chapters, Xóa thông tin chapters.

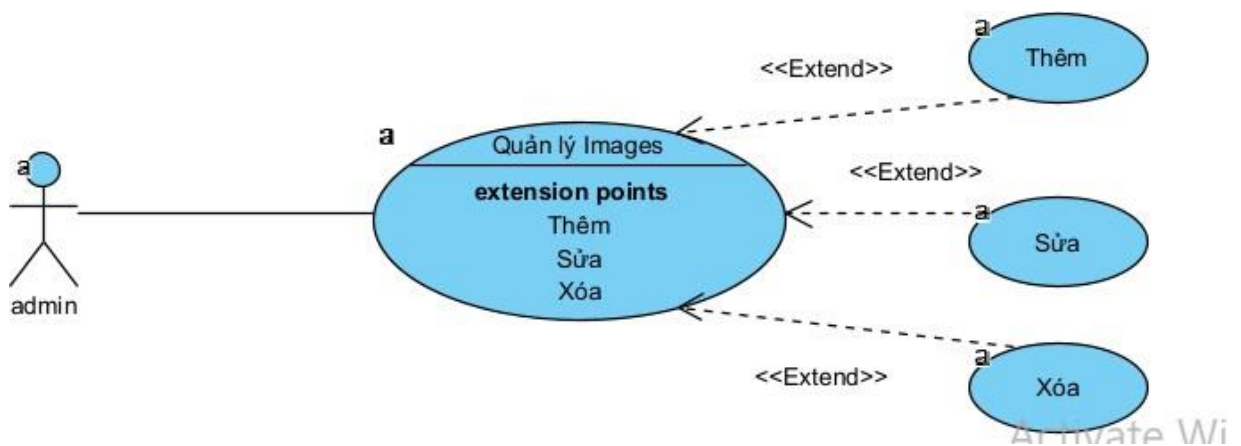
- Biểu đồ phân rã của use case Quản lý lượt xem(views).



Hình 3.2.1.1: Biểu đồ phân rã của use case Quản lý lượt xem.

Biểu đồ 7 thể hiện biểu đồ phân rã của use case Quản lý lượt xem gồm các use case phân rã là: resert lượt xem.

- Biểu đồ phân rã của use case Quản lý images.



Hình 3.2.1.1: Biểu đồ phân rã của use case Quản lý images.

Biểu đồ 8 thể hiện biểu đồ phân rã của use case Quản lý images gồm các use case phân rã là: up các ảnh của chapters, sửa ảnh, xóa ảnh.

3.2.1.2. Các đặc tả ca sử dụng

- Use case: Create Account

Mô tả: Người dùng hoặc quản trị viên tạo tài khoản mới.

Actor: User, Admin

Điều kiện xảy ra: Người dùng hoặc quản trị viên muốn tạo tài khoản mới.

Tiền điều kiện (Pre-Condition): Không có.

Hậu điều kiện (Post-Condition): Tài khoản mới được tạo và lưu vào hệ thống.

Luồng chính (Basic-Flow):

1. Người dùng hoặc quản trị viên truy cập vào trang đăng ký.
2. Người dùng hoặc quản trị viên nhập thông tin tài khoản mới (email, mật khẩu, thông tin cá nhân, v.v.).
3. Hệ thống kiểm tra tính hợp lệ của thông tin.
4. Nếu thông tin hợp lệ, hệ thống tạo tài khoản mới và lưu vào cơ sở dữ liệu.
5. Hệ thống thông báo tài khoản đã được tạo thành công và chuyển hướng đến trang đăng nhập.

Luồng phụ (Alternative-Flow):

Nếu thông tin không hợp lệ, hệ thống hiển thị lỗi và yêu cầu người dùng hoặc quản trị viên sửa lại thông tin.

Luồng sự kiện rẽ nhánh (Exception-Flow):

E1: Email đã tồn tại trong hệ thống

Hệ thống thông báo lỗi và yêu cầu người dùng hoặc quản trị viên nhập lại email khác.

- Use case: Login

Mô tả: Đăng nhập vào hệ thống bằng tài khoản người dùng hoặc quản trị viên.

Actor: User, Admin

Điều kiện xảy ra: Người dùng hoặc quản trị viên muốn đăng nhập vào hệ thống.

Tiền điều kiện (Pre-Condition):

Người dùng hoặc quản trị viên đã có tài khoản hợp lệ trong hệ thống.

Người dùng hoặc quản trị viên biết thông tin đăng nhập của mình (email và mật khẩu).

Hậu điều kiện (Post-Condition):

Người dùng hoặc quản trị viên được xác thực và truy cập vào hệ thống.

Hệ thống ghi nhận phiên đăng nhập (session).

Luồng chính (Basic-Flow):

1. Người dùng hoặc quản trị viên truy cập vào trang đăng nhập.
2. Người dùng hoặc quản trị viên nhập email và mật khẩu.
3. Hệ thống kiểm tra tính hợp lệ của email và mật khẩu.
4. Nếu thông tin hợp lệ, hệ thống xác thực người dùng hoặc quản trị viên.
5. Hệ thống tạo phiên đăng nhập cho người dùng hoặc quản trị viên.
6. Hệ thống thông báo đăng nhập thành công và chuyển hướng người dùng hoặc quản trị viên đến trang chủ hoặc bảng điều khiển quản trị.

Luồng phụ (Alternative-Flow):

Nếu thông tin không hợp lệ, hệ thống hiển thị thông báo lỗi và yêu cầu người dùng hoặc quản trị viên nhập lại thông tin.

Luồng sự kiện rẽ nhánh (Exception-Flow):

E1: Email không tồn tại

Hệ thống thông báo lỗi và yêu cầu người dùng hoặc quản trị viên nhập lại email.

E2: Mật khẩu không chính xác

Hệ thống thông báo lỗi và yêu cầu người dùng hoặc quản trị viên nhập lại mật khẩu.

E3: Nhập sai mật khẩu quá nhiều lần

Hệ thống khóa tài khoản tạm thời và thông báo cho người dùng hoặc quản trị viên.

- Use case: Like Comic

Mô tả: Thích một truyện tranh.

Actor: User

Điều kiện xảy ra: Người dùng muốn thích một truyện tranh.

Tiền điều kiện (Pre-Condition
)

: Người dùng đã đăng nhập và truyện tranh đã tồn tại.

Hậu điều kiện (Post-Condition): Một lượt thích mới được tạo và lưu vào bảng Likes.

Luồng chính (Basic-Flow)::

1. Người dùng chọn truyện tranh muốn thích.
2. Người dùng nhấp vào nút "Thích".
3. Hệ thống kiểm tra xem người dùng đã thích truyện tranh này chưa.
4. Nếu chưa, hệ thống lưu thông tin vào bảng Likes.
5. Hệ thống thông báo truyện tranh đã được thích thành công.

Luồng phụ (Basic-Flow):

Nếu người dùng đã thích truyện tranh, hệ thống thông báo lỗi và không lưu thông tin.

Luồng sự kiện rẽ nhánh (Exception-Flow):

Nếu comiciid không tồn tại, hệ thống thông báo lỗi và yêu cầu người dùng chọn lại truyện tranh.

- Use case: Comment on Comic

Mô tả: Bình luận về một truyện tranh.

Actor: User

Điều kiện xảy ra: Người dùng muốn bình luận về một truyện tranh.

Tiền điều kiện: Người dùng đã đăng nhập và truyện tranh đã tồn tại.

Hậu điều kiện: Một bình luận mới được tạo và lưu vào bảng Comments.

Luồng chính:

1. Người dùng chọn truyện tranh muốn bình luận.
2. Người dùng nhập nội dung bình luận (commenttext, commentedas).
3. Hệ thống kiểm tra tính hợp lệ của bình luận.
4. Hệ thống lưu bình luận vào bảng Comments.
5. Hệ thống thông báo bình luận đã được đăng thành công.

Luồng phụ:

Nếu bình luận không hợp lệ, hệ thống hiển thị lỗi và yêu cầu người dùng nhập lại bình luận.

Luồng sự kiện rẽ nhánh:

Nếu comicid không tồn tại, hệ thống thông báo lỗi và yêu cầu người dùng chọn lại truyện tranh.

- Use case: Xem comic

Mô tả: Người dùng có thể xem chi tiết và nội dung của một truyện tranh.

Actor: Người dùng (User)

Điều kiện xảy ra: Người dùng đã đăng nhập vào hệ thống và đã truy cập vào trang web hoặc ứng dụng di động.

Tiền điều kiện (Pre-Condition):

Người dùng đã tìm kiếm và chọn một truyện tranh cụ thể để xem.

Hậu điều kiện (Post-Condition):

Người dùng đã xem thành công nội dung của truyện tranh.

Luồng chính (Basic Flow):

1. Người dùng truy cập vào trang chi tiết của truyện tranh.
2. Hệ thống hiển thị thông tin chi tiết về truyện tranh bao gồm tiêu đề, tác giả, mô tả, ảnh bìa, và danh sách các chương.
3. Người dùng chọn một chương để xem.
4. Hệ thống hiển thị nội dung của chương được chọn.
5. Người dùng có thể cuộn xuống để đọc toàn bộ nội dung của chương.
6. Người dùng có thể thực hiện các thao tác như cuộn trang, phóng to, thu nhỏ hoặc bình luận nếu có.

Luồng phụ (Alternative Flow):

Nếu không có chương nào có sẵn để xem:

Hệ thống thông báo cho người dùng biết rằng không có nội dung để hiển thị và hướng dẫn họ quay lại trang trước.

Luồng sự kiện rẽ nhánh (Exception-Flow):

Nếu hệ thống gặp lỗi khi tải nội dung của chương:

Hệ thống thông báo lỗi cho người dùng và yêu cầu họ thử lại sau.

- Use Case: Quản lý Comics

Mô tả:

Admin có thể quản lý danh sách truyện tranh, bao gồm thêm, sửa, xóa và xem chi tiết truyện tranh.

Actor:

Admin

Điều kiện xảy ra:

Admin đã đăng nhập vào hệ thống và có quyền truy cập vào chức năng quản lý Comics.

Tiền điều kiện (Pre-Condition):

Hệ thống đã hiển thị giao diện quản lý Comics cho Admin.

Hậu điều kiện (Post-Condition):

Thay đổi trong danh sách truyện tranh được lưu lại trong hệ thống.

Luồng chính (Basic Flow):

1. Admin truy cập vào chức năng quản lý Comics.
2. Hệ thống hiển thị danh sách các truyện tranh hiện có và các tùy chọn quản lý.
3. Admin có thể thêm mới, sửa đổi hoặc xóa truyện tranh từ danh sách.
4. Admin có thể xem chi tiết của một truyện tranh bằng cách nhấp vào tên của nó trong danh sách.

Luồng phụ (Alternative Flow):

Nếu không có truyện tranh nào tồn tại trong danh sách:

Hệ thống hiển thị thông báo "Không có truyện tranh nào tồn tại" và không hiển thị bất kỳ tùy chọn nào khác.

Luồng sự kiện rẽ nhánh (Exception Flow):

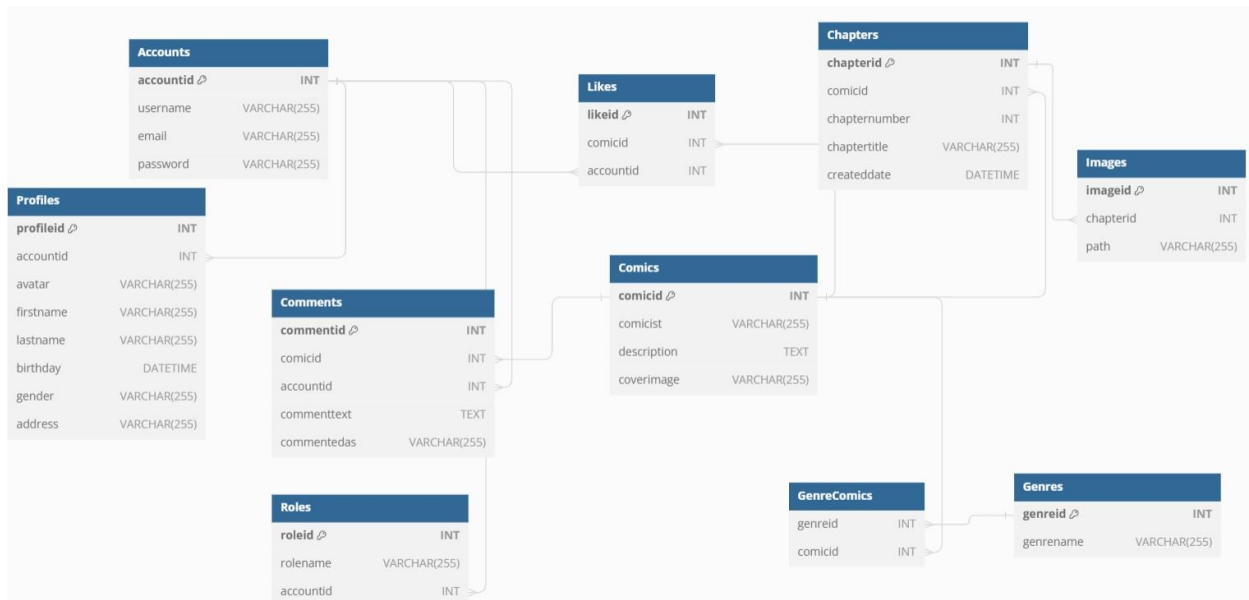
Nếu có lỗi xảy ra trong quá trình thêm, sửa hoặc xóa truyện tranh:

Hệ thống hiển thị thông báo lỗi cụ thể cho Admin và không thực hiện thay đổi nào trong danh sách.

Mở rộng:

Admin có thể thực hiện tìm kiếm và sắp xếp danh sách truyện tranh theo tiêu chí như tên, tác giả hoặc thể loại.

3.2.2.3. Biểu đồ lớp thực thể



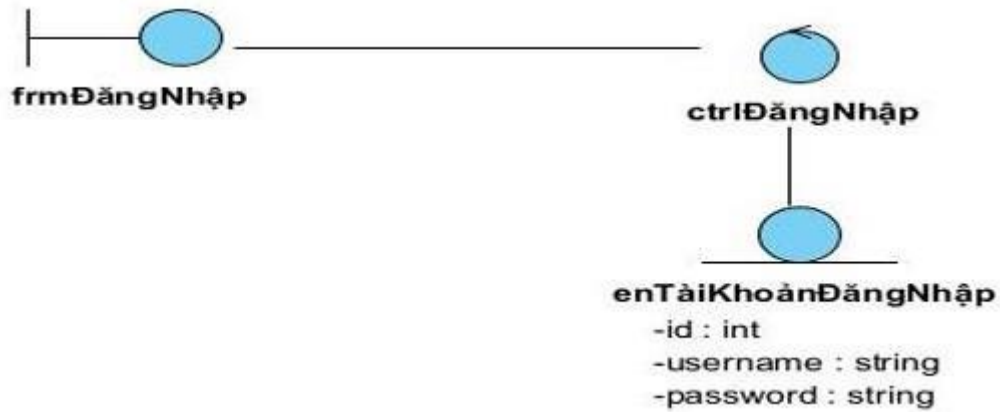
Hình 3.2.1.2: biểu đồ lớp thực thể

Biểu đồ thực thể có các lớp liên quan đến nhau:

- Lớp accounts có quan hệ với lớp profiles, comments, roles, likes
- Lớp comics có quan hệ với lớp comments, chapters, genres
- Lớp chapters có quan hệ với lớp images

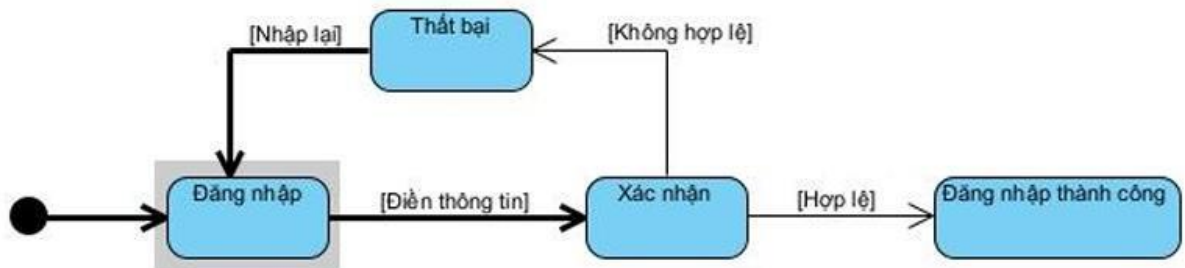
3.2.3. Biểu đồ phân tích ca sử dụng

Biểu đồ phân tích của use case Đăng nhập



Hình 3.2.3: biểu đồ phân tích

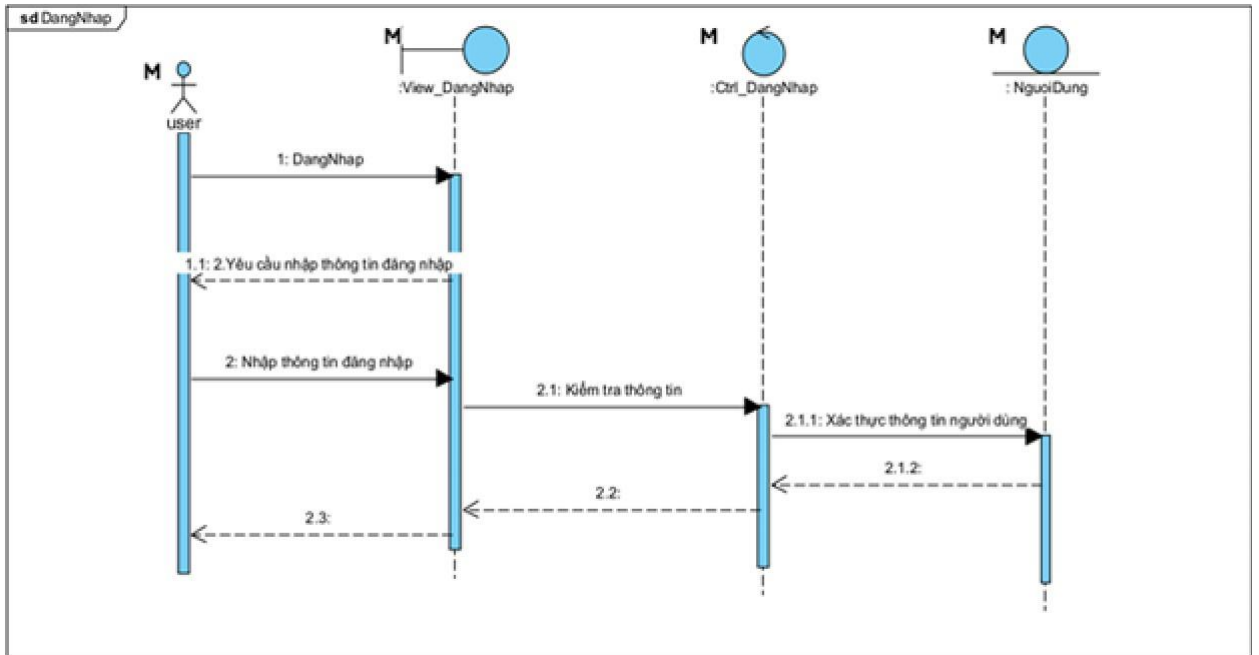
3.2.4. Biểu đồ trạng thái



Hình 3.2.4: biểu đồ trạng thái

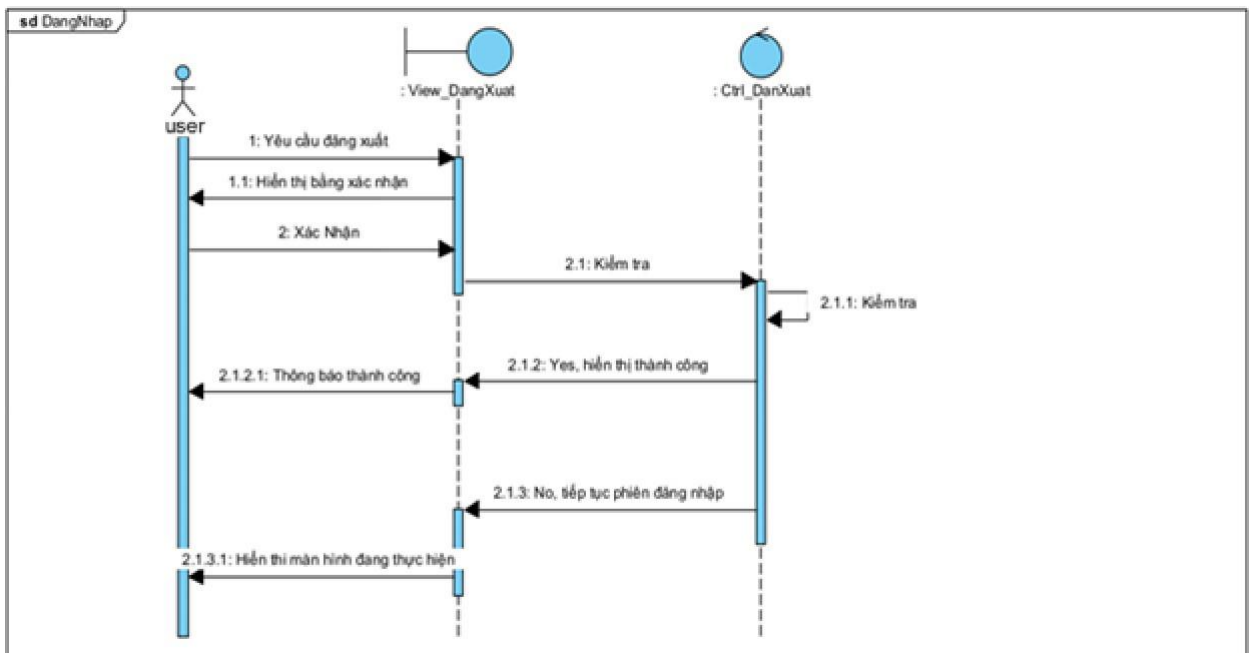
3.2.5. Biểu đồ tuần tự

Biểu đồ tuần tự của use case đăng nhập.



Hình 3.2.5: biểu đồ tuần tự use case đăng nhập.

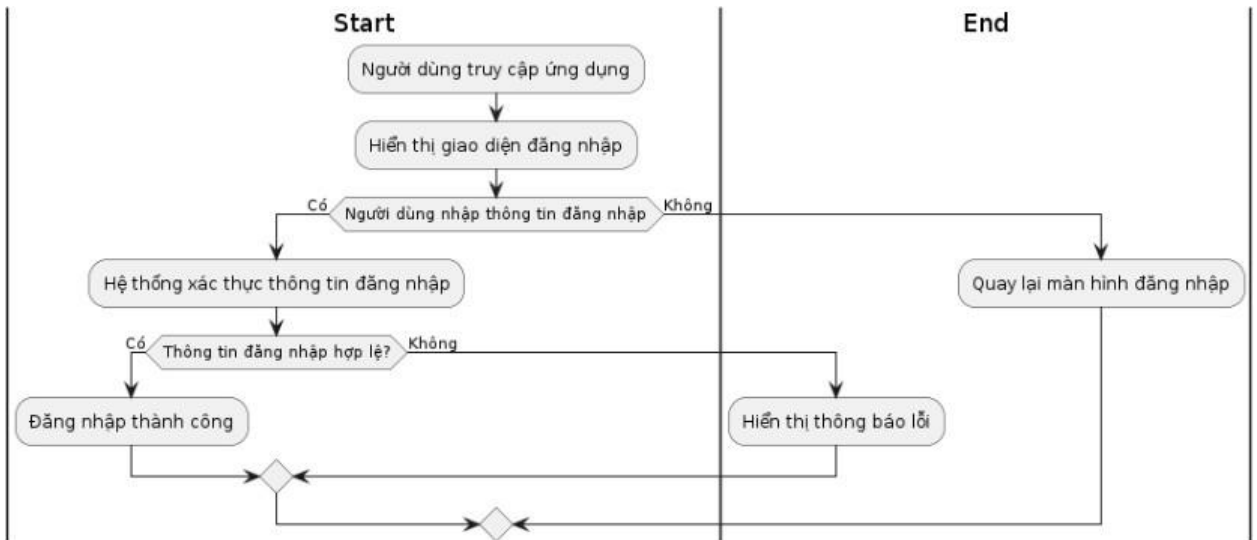
Biểu đồ tuần tự của use case đăng xuất.



Hình 3.2.5: biểu đồ tuần tự use case đăng xuất.

3.2.6. Biểu đồ hoạt động của use case/ hệ thống/ phương thức

Biểu đồ hoạt động của use case Đăng Nhập



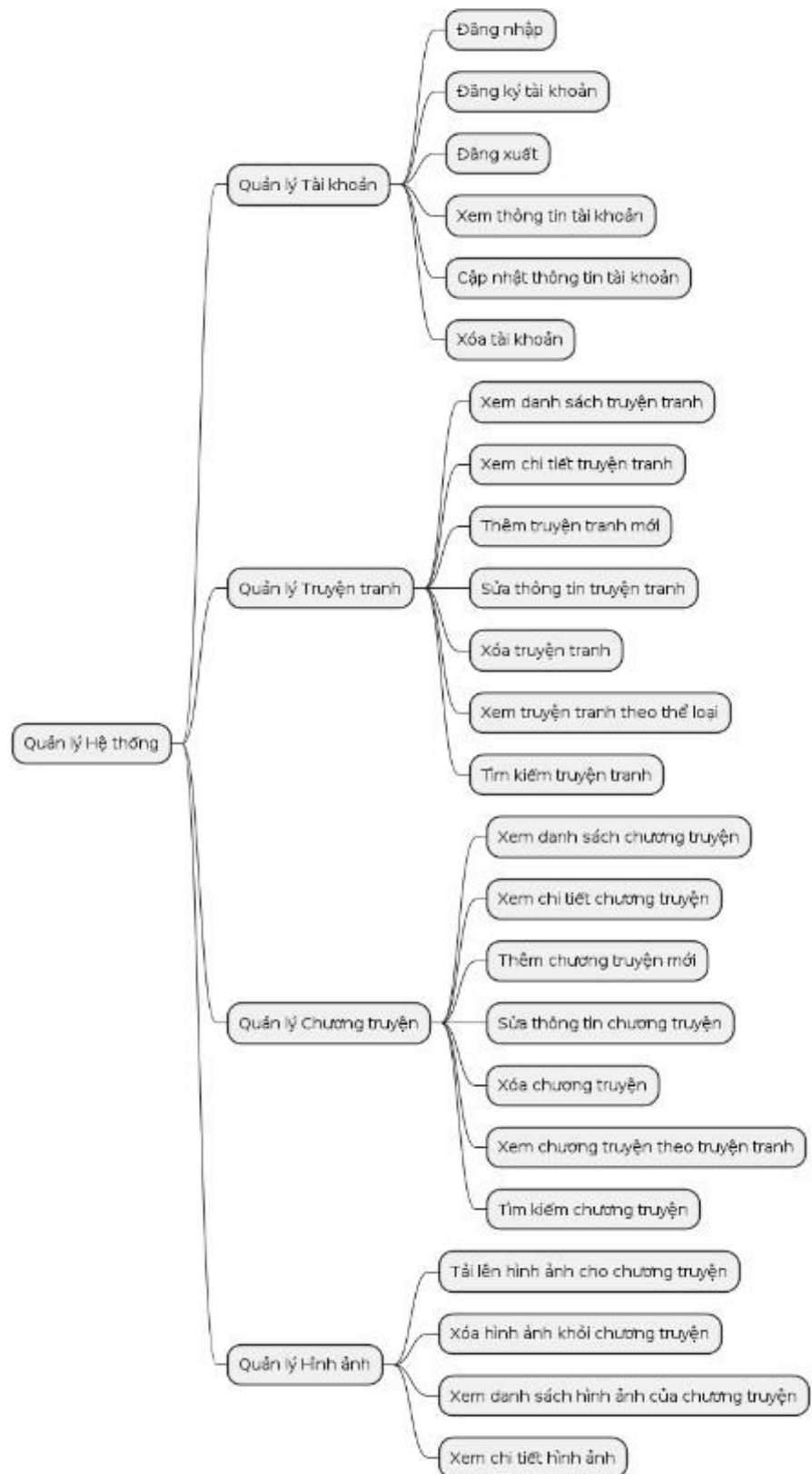
Hình 3.2.6: Biểu đồ hoạt động của use case đăng nhập.

Biểu đồ hoạt động của use case thêm comics



Hình 3.2.6: Biểu đồ hoạt động của use case thêm comics

3.2.7. Biểu đồ phân cấp chức năng (Functional Decomposition Diagram - FDD)

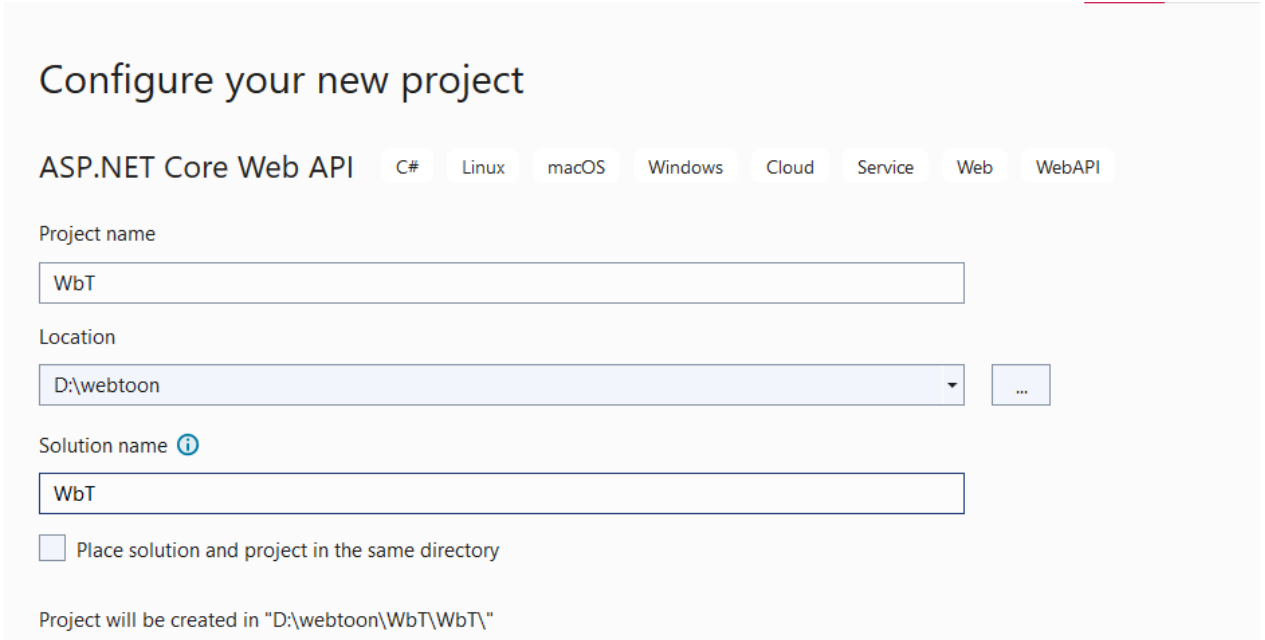


Hình 3.2.7: Biểu đồ phân cấp

CHƯƠNG 4: XÂY DỰNG CHƯƠNG TRÌNH VÀ ĐÁNH GIÁ

4.1. Xây dựng BackEnd

4.1.1. Khởi tạo Project



Configure your new project

ASP.NET Core Web API C# Linux macOS Windows Cloud Service Web WebAPI

Project name
WbT

Location
D:\webtoon

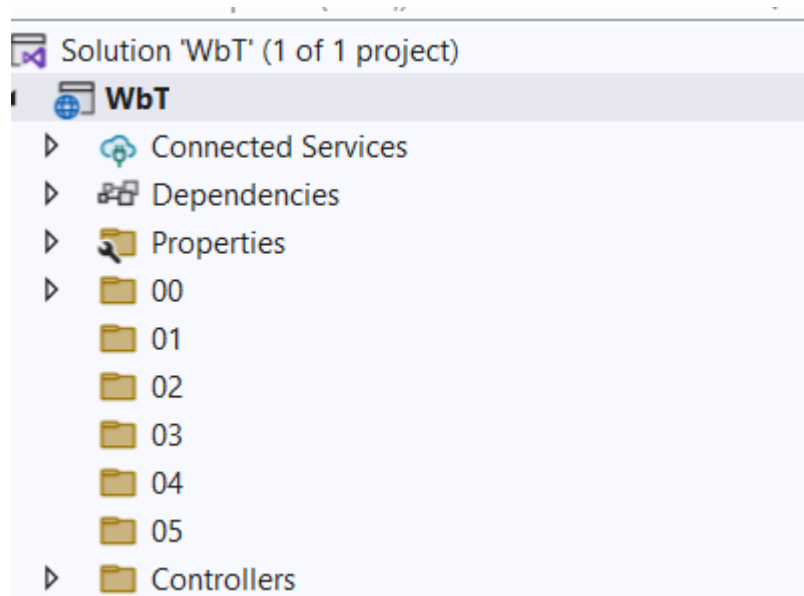
Solution name ⓘ
WbT

☐ Place solution and project in the same directory

Project will be created in "D:\webtoon\WbT\WbT\'"

Hình 4.1.1: Khởi tạo project

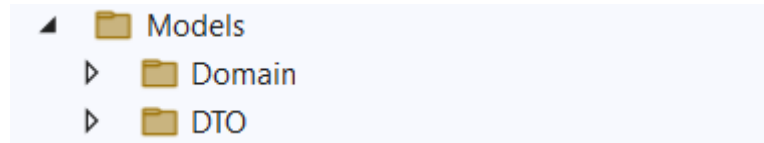
- Tạo một project mới sử dụng ASP.NET Core Web API



Hình 4.1.1: Sau khi tạo project

Sau khi tạo project ta được như hình trên

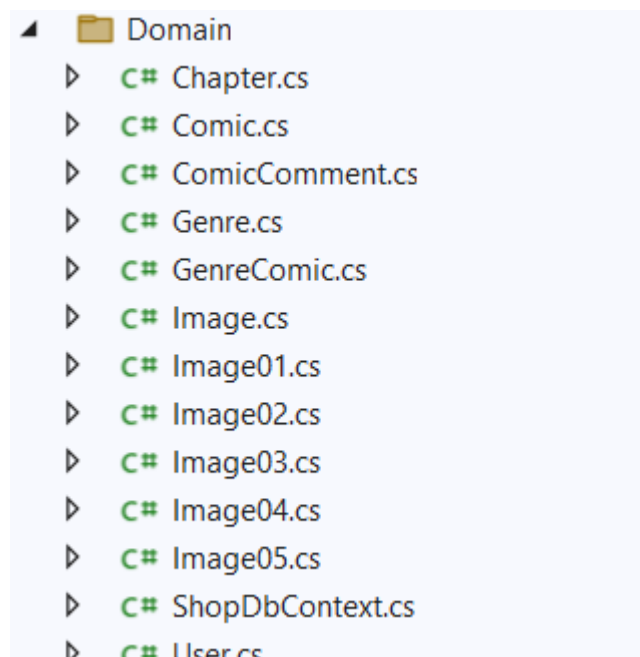
4.1.2. Khởi tạo Folder Models



Hình 4.1.2: Khởi tạo Folder Models

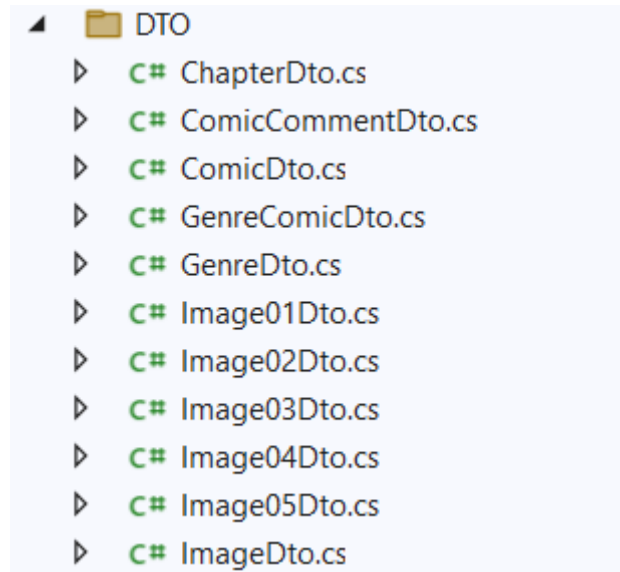
- Sau khi tạo folder.Models. Ta tiếp tục tạo folder là Domain, DTO.

a) Domain



Hình 4.1.2: Domain

b) DTO



Hình 4.1.2: DTO

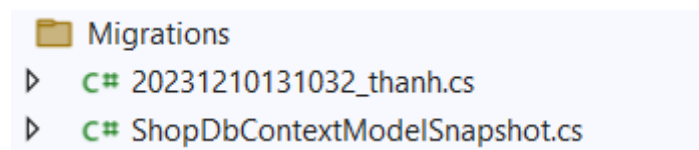
4.1.3. Khởi tạo class *ShopDbContext*, *add-migration*

a) ShopDbContext



Hình 4.1.3: Context

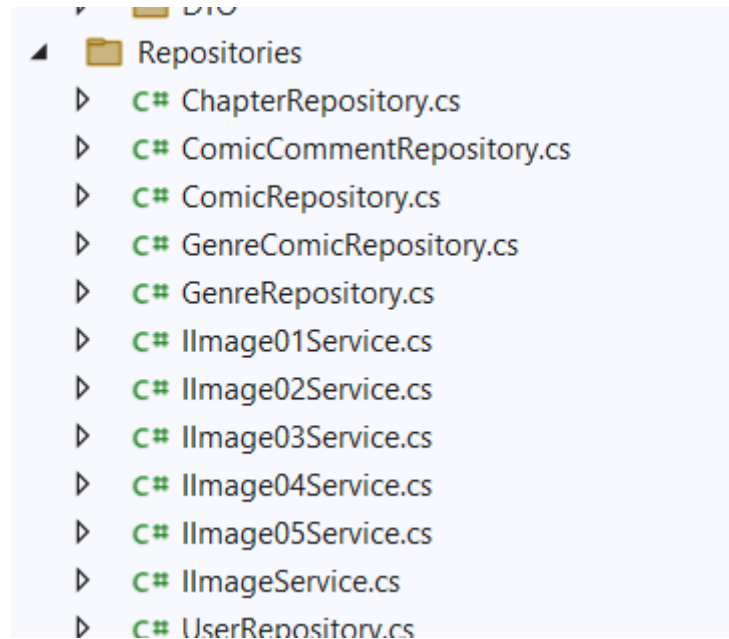
b) Migrations



Hình 4.1.3: Migrations

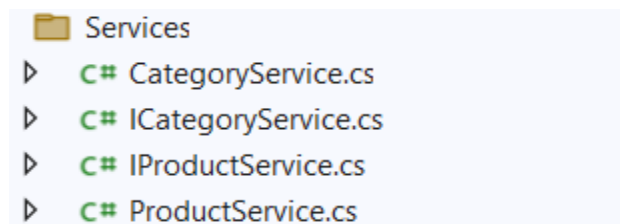
4.1.4. Khởi tạo Folder *Repositories*, *Services*

a) Repositories



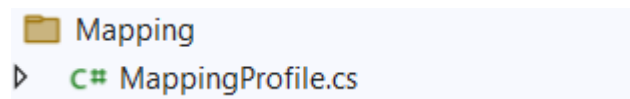
Hình 4.1.4: Khởi tạo Repositories

b) Services



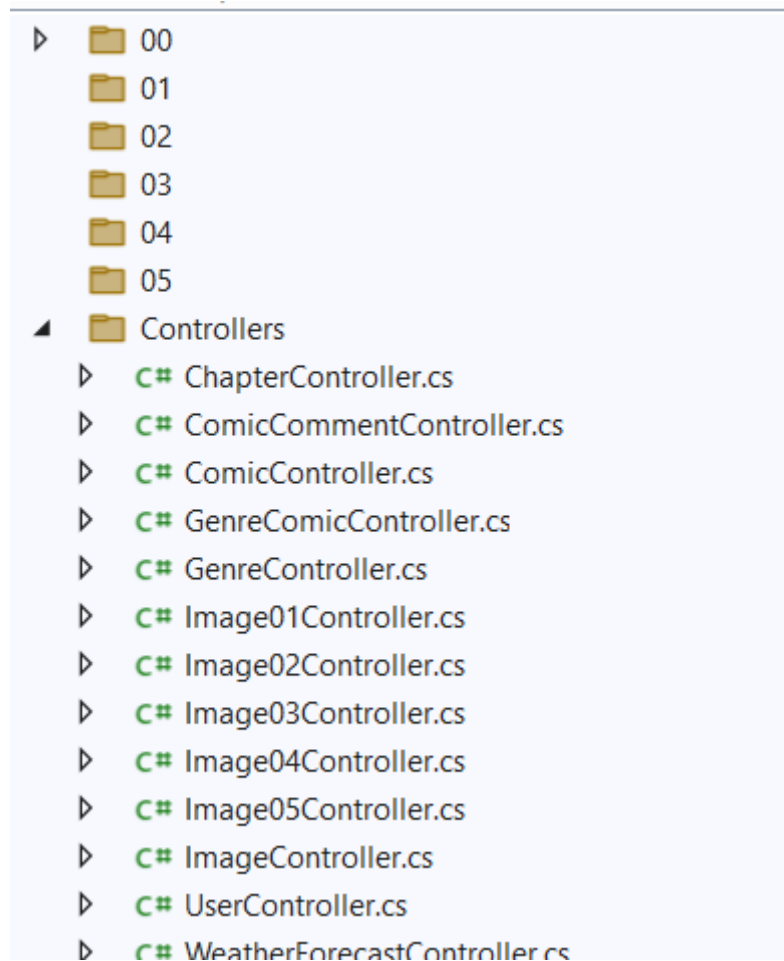
Hình 4.1.4: Services

d) Mapping



Hình 4.1.4: Mapping

- Sau khi đã hoàn thành khởi tạo và liên kết các Folder ta tiếp tục tạo các Controller trong project WbT.



Hình 4.1.4: Controllers

4.2. Xây dựng FrontEnd

4.2.1. Trang đăng nhập, đăng ký và Profile

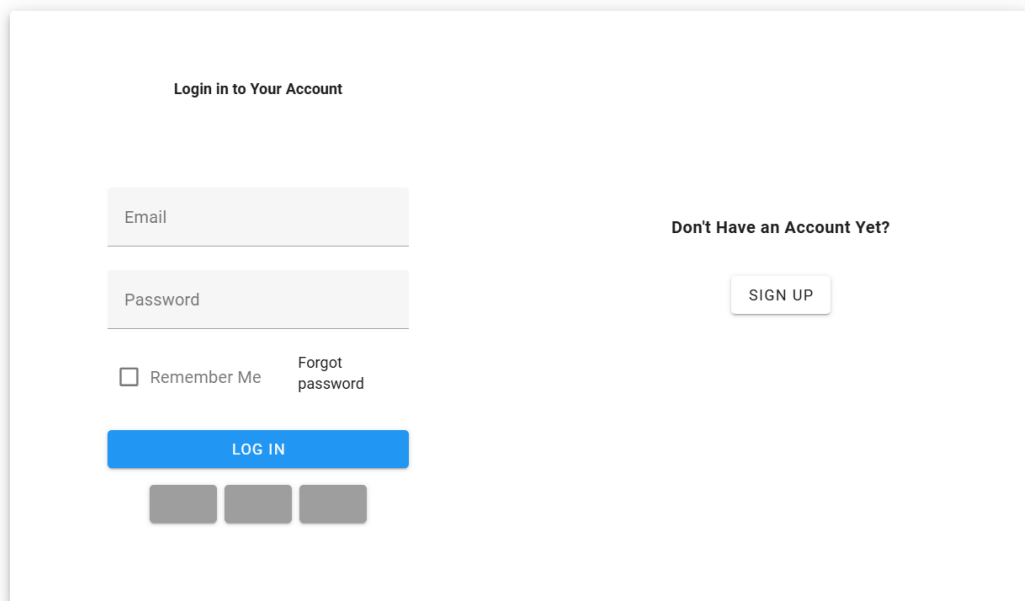
Trang đăng nhập và đăng ký không chỉ là nơi để người dùng nhập thông tin tài khoản, mà còn là nơi tạo và quản lý token cho việc xác thực. Dưới đây là mô tả về cách token được sử dụng trong hai trang này:

- Đăng nhập

Xác thực thông qua JWT (JSON Web Token): Sau khi người dùng nhập thông tin đăng nhập và gửi yêu cầu, máy chủ sẽ xác thực thông tin và tạo một JWT chứa thông tin xác thực như ID người dùng và thời gian hết hạn.

Lưu trữ token: Token JWT được lưu trữ trên phía máy khách, thường được lưu trong cookie hoặc local storage để sử dụng cho các yêu cầu tiếp theo mà không cần người dùng nhập lại thông tin đăng nhập.

Xác thực và phân quyền: Mỗi lần người dùng gửi yêu cầu đến máy chủ, token sẽ được gửi kèm theo để xác thực. Máy chủ sẽ kiểm tra tính hợp lệ của token và dựa vào đó phân quyền truy cập cho người dùng.

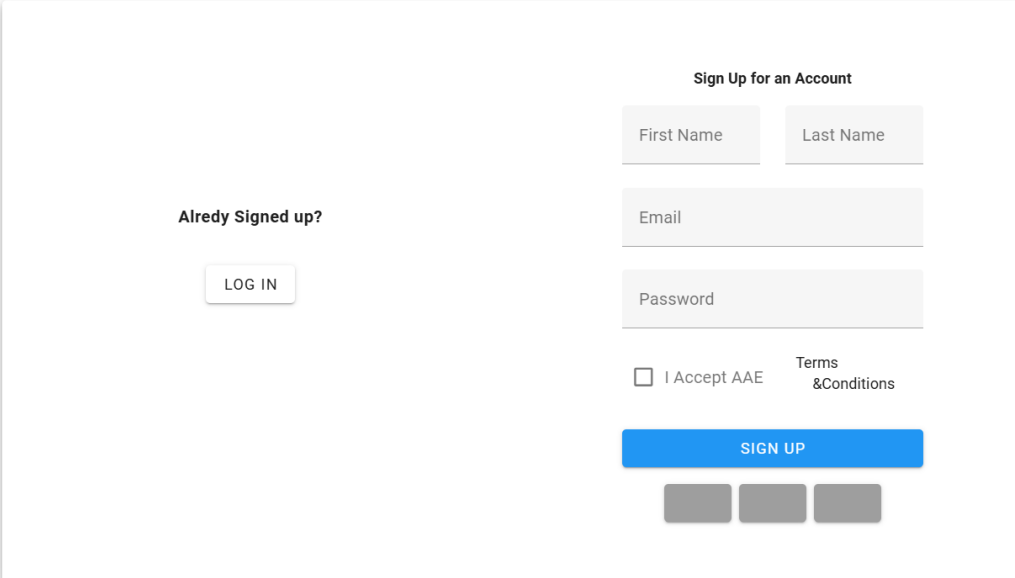
A login form titled "Login in to Your Account". It features two input fields: "Email" and "Password". Below the "Password" field, there is a checkbox labeled "Remember Me" and a link "Forgot password". A blue "LOG IN" button is positioned below these elements. To the right of the form, there is a link "Don't Have an Account Yet?" and a "SIGN UP" button. At the bottom of the form, there are three small, empty square boxes.

Hình 4.2.1: Trang login

- Đăng ký

Tạo token cho tài khoản mới: Sau khi người dùng nhập thông tin và gửi yêu cầu đăng ký, máy chủ sẽ tạo một token mới và gửi lại cho người dùng. Token này có thể được sử dụng ngay lập tức để đăng nhập mà không cần thực hiện lại quá trình đăng nhập.

Xác thực và gửi token: Máy chủ sẽ gửi token cho người dùng qua email hoặc trả về trong phản hồi sau khi đăng ký thành công. Người dùng sẽ sử dụng token này để đăng nhập lần đầu tiên và kích hoạt tài khoản.



The image shows a web form titled "Sign Up for an Account". On the left, there is a link "Already Signed up?" with a "LOG IN" button below it. The main form area contains input fields for "First Name", "Last Name", "Email", and "Password". Below these fields is a checkbox labeled "I Accept AAE" followed by a link "Terms & Conditions". At the bottom of the form is a prominent blue "SIGN UP" button. Below the button are three small, grey rectangular placeholders.

Hình 4.2.1: Trang sign up

- Profile :

Là nơi mà người dùng có thể tạo và quản lý thông tin cá nhân của mình sau khi đăng ký tài khoản. Dưới đây là một số chức năng và lợi ích mà profile mang lại cho người dùng:

Hiển thị thông tin cá nhân: Profile cho phép người dùng hiển thị thông tin cá nhân như tên, họ, ngày sinh, giới tính, địa chỉ và ảnh đại diện. Điều này giúp tạo ra một trang cá nhân đầy đủ và thú vị.

Tùy chỉnh hồ sơ: Người dùng có thể tùy chỉnh thông tin cá nhân của họ bất cứ lúc nào. Họ có thể cập nhật thông tin mới như đổi ảnh đại diện, thay đổi địa chỉ, hoặc cập nhật thông tin về bản thân.

Up ảnh đại diện: Chức năng up ảnh đại diện cho phép người dùng tải lên hình ảnh của họ để làm ảnh đại diện trên trang cá nhân và trong các hoạt động liên quan đến tài khoản của họ. Điều này tạo ra sự cá nhân hóa và nhận dạng dễ dàng hơn cho người dùng trên nền tảng.

Quản lý thông tin cá nhân: Profile cung cấp cho người dùng một nơi để quản lý toàn bộ thông tin cá nhân của họ.

Tạo liên kết với tài khoản: Profile liên kết trực tiếp với tài khoản của người dùng, giúp họ dễ dàng quản lý thông tin cá nhân và theo dõi hoạt động trên nền tảng.

profile



First Name

Last Name

Address

Contact Number

Gender

Cancel

Save

Hình 4.2.1: Trang Profile

4.2.2. Trang Quản lý

Trang quản lý là nơi quản lý các chức năng quản trị của ứng dụng. Các chức năng có thể bao gồm:

Quản lý tài khoản: Thêm, sửa, xóa người dùng, cũng như quản lý vai trò và quyền hạn của họ.

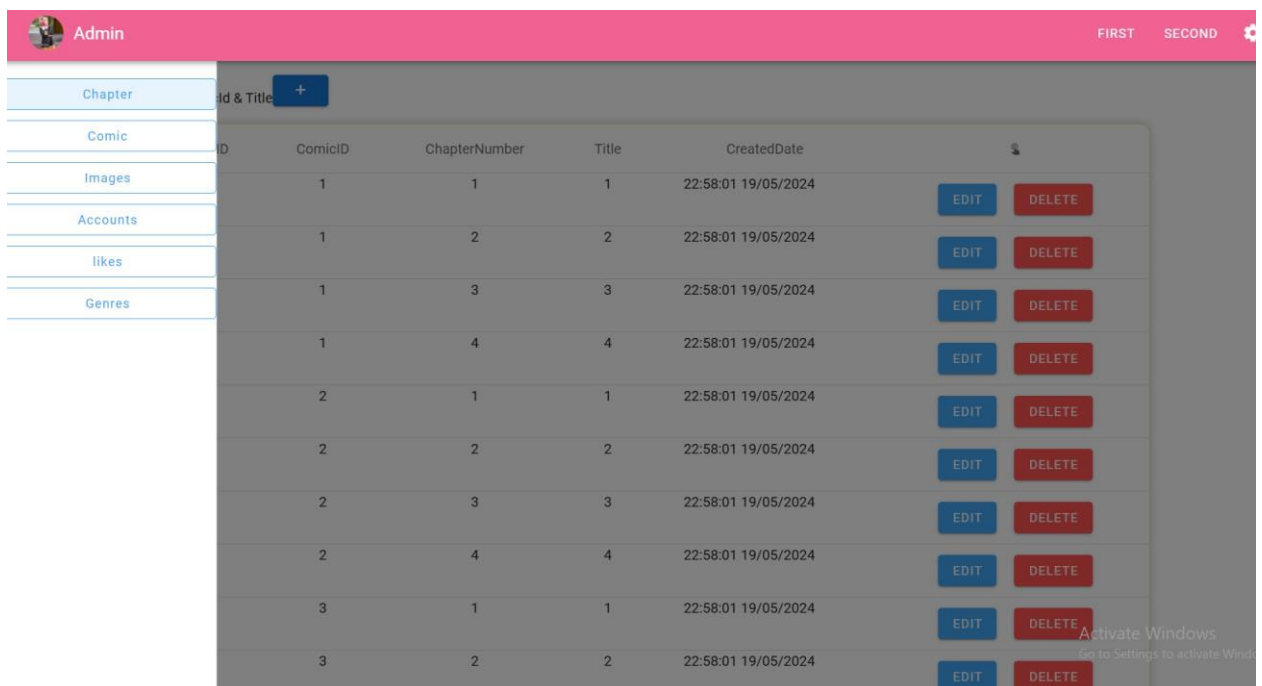
Quản lý truyện tranh: Hiển thị danh sách truyện tranh, cho phép quản lý và thao tác các truyện tranh như thêm, sửa, xóa.

Quản lý thể loại: Hiển thị danh sách các thể loại truyện tranh, cho phép thêm, sửa, xóa thể loại.

Quản lý chapters: Xem, Thêm, sửa, xóa các chapters .

Quản lý Images: Xem, Thêm, sửa, xóa các Images .

Quản lý Likes: Xem, Xóa các like từ quản lý.

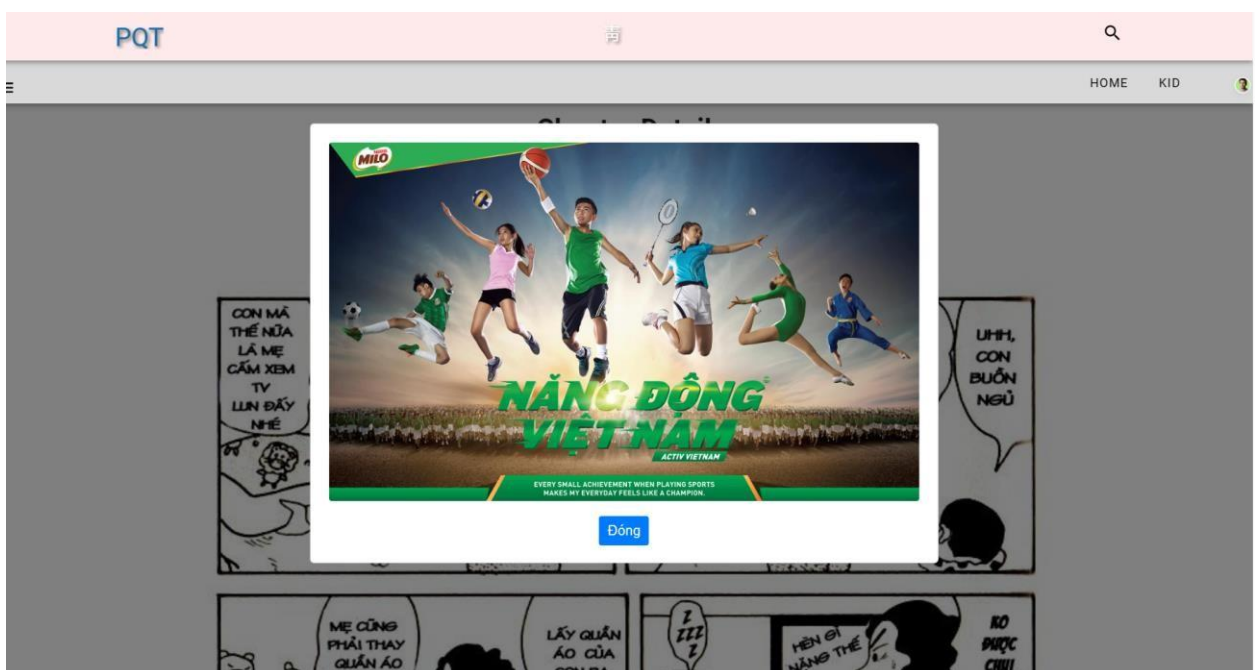


Hình 4.2.2: Trang quản lý

4.2.3. Trang xem comics

Trang xem truyện tranh là nơi người dùng có thể thưởng thức các truyện tranh trực tuyến. Các chức năng của trang này có thể bao gồm:

- Cho phép người dùng đọc truyện theo từng chapter hoặc theo trang.
- Hiện thị quảng cáo tạo kinh phí duy trì web.
- Giúp người dùng tương tác, trò chuyện bằng cách bình luận dưới những comic mà mình đọc.
- Thêm những comic yêu thích vào kho riêng tư.



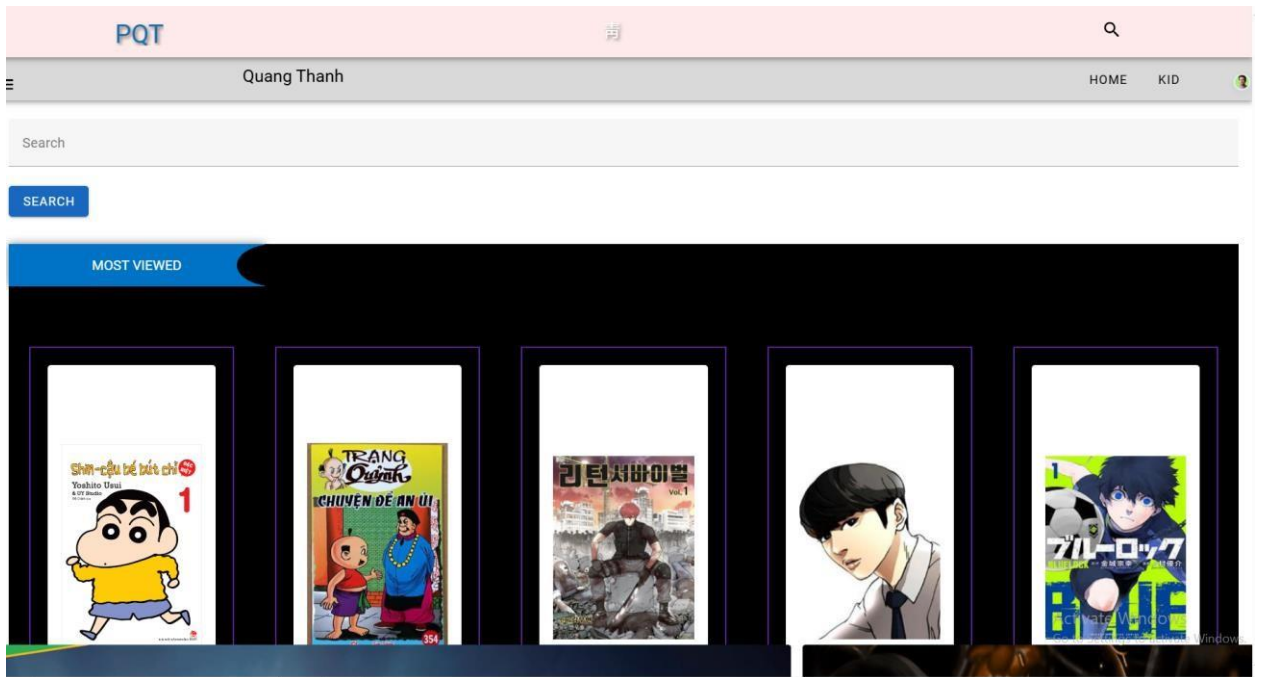
Hình 4.2.3: Trang quản lý CategoryComic

4.2.4. Trang chủ, chi tiết

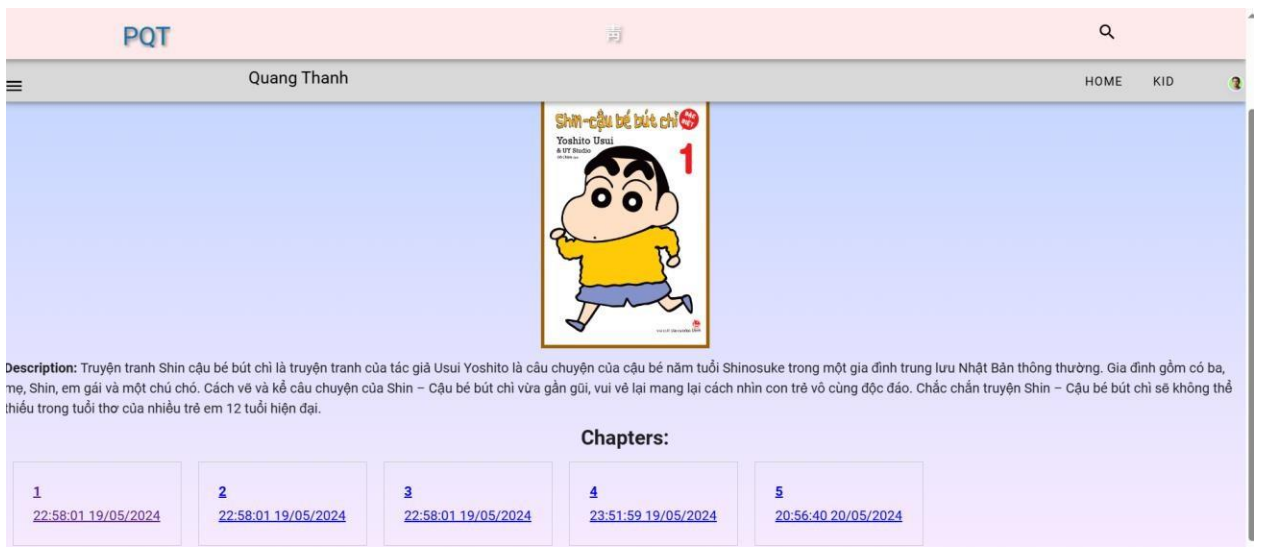
Trang chủ là trang mặc định khi người dùng truy cập vào ứng dụng. Các chức năng và nội dung của trang chủ có thể bao gồm:

- Cung cấp chức năng tìm kiếm để người dùng có thể tìm kiếm truyện theo tiêu đề hoặc tác giả.
- Hiện thị các truyện tranh nổi bật hoặc mới nhất.
- Cung cấp liên kết đến các thể loại truyện tranh khác nhau.
- Hiện thị danh sách các truyện tranh theo thể loại.

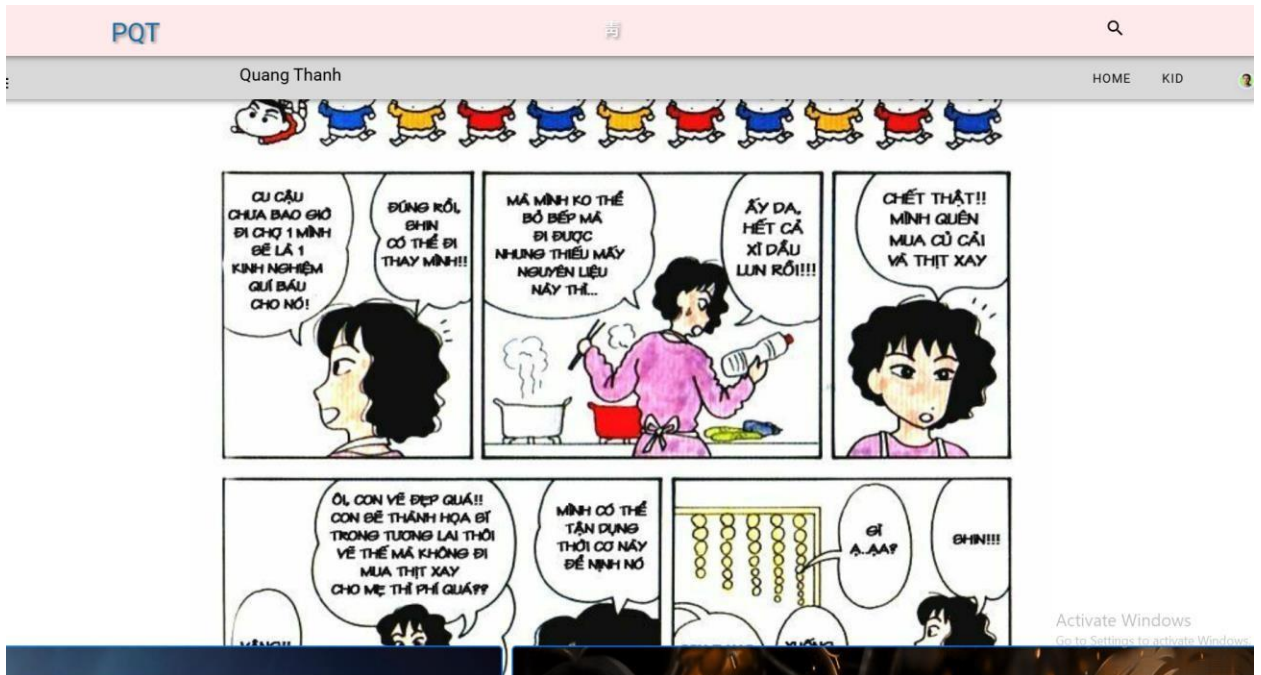
- Hiện thị chi tiết truyện tranh khi người dùng chọn một truyện cụ thể.
- Thông tin về ứng dụng và nhà phát triển.
- Các chức năng xã hội như tương tác qua thanh công cụ comments.
- Hiện thị thời gian upload chapters.



Hình 4.2.4: Trang chủ



Hình 4.2.4: Chi tiết Comic



Hình 4.2.4: Chi tiết Image

4.3. Đánh giá

4.3.1. Kết quả đạt được

- Website hoàn chỉnh: Đề tài đã thực hiện xây dựng một website đọc truyện tranh, cho phép người dùng đọc truyện, tìm kiếm

theo các tiêu chí khác nhau và bình luận với ý kiến cá nhân.

- Thiết kế giao diện thân thiện: Giao diện của website được thiết kế đẹp mắt, dễ sử dụng, tạo ấn tượng tốt đối với người dùng. - Tính năng tương tác: Website cung cấp các tính năng tương tác như bình luận truyện tranh, chia sẻ truyện lên mạng xã hội, giúp tạo cơ hội tương tác giữa người dùng.

- Tính thẩm mỹ và chất lượng hình ảnh: Các bộ truyện được trình bày đẹp mắt, với hình ảnh chất lượng cao, giúp thu hút người xem.

4.3.2. Ưu điểm

- Tạo ra cơ hội cho truyện tranh: Website giúp tạo ra cơ hội

tiếp cận và quảng cáo cho các bộ truyện mới, giúp nâng cao giá trị và nhận thức của công đồng về các thể loại truyện hiện nay trên khắp các nước. - Tính tương tác cao: Các tính năng tương tác giúp tạo ra môi trường trải nghiệm đa chiều, thu hút người dùng quay lại trang web.

4.4.3. Nhược điểm

- Cạnh tranh: Cũng có nhiều website làm về ứng dụng đọc truyện tranh đẹp mắt, đòi hỏi website phải có chiến lược tiếp thị và quảng cáo mạnh mẽ để nổi bật cũng như giao diện dễ tiếp cận.
- Chức năng website: Các chức năng trên website còn chưa nhiều, mới có những chức năng cơ bản.
- Quản lý và cập nhật: Hệ thống quản lý còn chưa hoàn thiện.
- Bảo mật thông tin: Cần đảm bảo an ninh thông tin để bảo vệ thông tin cá nhân và giao dịch của người dùng trên website.

4.4.4. Tính Năng Mở Rộng

- Cải Thiện Hệ Thống Quản Lý: Phát triển hệ thống quản lý để có khả năng quản lý và cập nhật dữ liệu một cách linh hoạt và hiệu quả.
- Mở Rộng Chức Năng: Bổ sung các chức năng mới như quản lý truyện tranh, thể loại, chapters, images, likes để cung cấp trải nghiệm đa dạng cho người dùng.
- Tăng Cường Bảo Mật: Đảm bảo bảo mật thông tin cá nhân và giao dịch thông qua việc triển khai các biện pháp bảo mật tiên tiến và chuẩn mực.

Các bổ sung và cải tiến này sẽ giúp nâng cao chất lượng và trải nghiệm của website đọc truyện tranh, từ đó thu hút và giữ chân được người dùng, đồng thời củng cố vị thế trong thị trường cạnh tranh.

KẾT LUẬN

Trong quá trình nghiên cứu và phát triển ứng dụng đọc truyện tranh trên nền tảng ASP.NET Core Web API và VueJS, tôi đã chứng kiến sức hấp dẫn của thị trường đọc truyện tranh và nhu cầu ngày càng tăng cao của độc giả. Việc tạo ra một nền tảng trực tuyến chuyên nghiệp và thuận tiện không chỉ giúp kết nối giữa người đọc và người sáng tác mà còn mang lại nhiều lợi ích cho cả hai bên.

Thông qua việc cung cấp thông tin chi tiết và chất lượng về truyện tranh, chăm sóc khách hàng chuyên nghiệp, và cung cấp các tiện ích trực tuyến để tăng trải nghiệm người dùng, ứng dụng có thể thu hút được nhiều độc giả và tạo ra một cộng đồng đam mê truyện tranh sôi động.

Tôi tin rằng, thông qua việc xây dựng ứng dụng đọc truyện tranh trên nền tảng ASP.NET Core Web API và VueJS, không chỉ giúp độc giả tiếp cận dễ dàng với các tác phẩm truyện tranh mới nhất mà còn thúc đẩy sự phát triển của ngành công nghiệp truyện tranh. Bằng cách sử dụng công nghệ và kỹ thuật tiên tiến, ứng dụng có thể tối ưu hóa trải nghiệm đọc truyện, tăng cường quảng bá các tác phẩm và tạo ra một môi trường đọc truyện trực tuyến an toàn, tiện lợi và thú vị cho độc giả.

TÀI LIỆU THAM KHẢO

Tiếng Việt:

- [1]. Giáo Trình Lý Thuyết Và Thiết Kế Cơ Sở Dữ Liệu, Đại Học Đại Nam, 2023.
- [2]. Giáo Trình Thiết Kế Web 1 Của Thầy Phạm Văn Tiệp.
- [3]. Thiết Kế Giao Diện Người Dùng (UI) của Luận Nguyễn.

Tiếng Anh:

- [1]. Tokenify: Simplifying Token Authentication in Vue.js and .NET Core Web API.
- [2]. Vue JS And .NET Core Web API Full Stack Web Development Master Course.
- [3]. FileDrop: Simplifying File Uploads in Vue.js and .NET Core Web API.

Danh mục website tham khảo:

- [1]. <https://vuetifyjs.com/en/>
- [2]. <https://www.w3schools.com/>
- [3]. <https://stackoverflow.com/questions/tagged/asp.net-core>