

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
VIỆN KHOA HỌC KỸ THUẬT BƯU ĐIỆN



BÀI TẬP LỚN

Môn: Lập trình với ngôn ngữ Script
Đề tài: Website quản lý danh sách Anime cá nhân

GIẢNG VIÊN: Vũ Văn Thương

Lớp tín chỉ: RIPT1302-20242-04

THÔNG TIN SINH VIÊN:

HỌ VÀ TÊN

Trần Nhật Phúc

MÃ SINH VIÊN

B24DCCC229

Hà Nội, 07/2025

Mục Lục

LỜI MỞ ĐẦU.....	4
LỜI CẢM ƠN.....	5
MỞ ĐẦU.....	6
1. Giới thiệu đề tài.....	6
1.1. Bối cảnh và tầm quan trọng của quản lý thông tin cá nhân trên nền tảng web.....	6
1.2. Tổng quan về MyAnimeList và nhu cầu phát triển ứng dụng tương tự.....	7
1.3. Mục tiêu và phạm vi của dự án.....	7
2. Mục đích và ý nghĩa của đề tài.....	8
2.1. Mục đích.....	8
2.2. Ý nghĩa.....	9
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT.....	10
1. Ứng dụng website trong quản lý danh sách phim.....	10
1.1. Lợi ích của ứng dụng web trong quản lý dữ liệu cá nhân.....	10
1.2. So sánh với phương pháp truyền thống và các nền tảng hiện có.....	11
2. Giới thiệu về thể loại Anime và cộng đồng người xem.....	11
2.1. Lịch sử và sự phát triển của Anime.....	11
2.2. Đặc điểm cộng đồng người xem và nhu cầu của họ.....	12
3. Các chức năng phổ biến của hệ thống quản lý Anime.....	13
3.1. Phân tích các tính năng cốt lõi (Cơ sở dữ liệu, quản lý danh sách, đánh giá, tìm kiếm).....	13
3.2. Các API Anime công cộng và lựa chọn cho dự án.....	13
4. Công nghệ phát triển website.....	16
4.1. HTML/CSS/JavaScript (Frontend).....	16
4.1.1. Vai trò của HTML trong cấu trúc giao diện.....	17
4.1.2. Vai trò và ưu điểm của CSS trong thiết kế và responsive.....	17
4.1.3. Vai trò và chức năng của JavaScript trong tương tác người dùng và DOM Manipulation.....	18
4.1.4. Tích hợp API với Fetch/Axios và xử lý bất đồng bộ.....	19
4.2. Node.js / Express (Backend).....	20
4.2.1. Kiến trúc và ưu điểm của Node.js.....	20
4.2.2. Ưu điểm của Express trong xây dựng RESTful API.....	21
4.2.3. Các nguyên tắc thiết kế RESTful API.....	22
4.3. MySQL (Database).....	24
4.3.1. Khái niệm cơ sở dữ liệu quan hệ và MySQL.....	24
4.3.2. Vai trò của Primary Key và Foreign Key trong quan hệ dữ liệu.....	25
CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG.....	27
1. Người dùng và yêu cầu chức năng.....	27
1.1. Đối tượng người dùng và kịch bản sử dụng chi tiết.....	27
1.2. Yêu cầu chức năng (Functional Requirements).....	27

1.3. Yêu cầu phi chức năng (Non-functional Requirements: hiệu năng, bảo mật, khả năng mở rộng, khả năng sử dụng).....	28
2. Các tính năng chính của hệ thống.....	29
2.1. Đăng ký và đăng nhập.....	29
2.2. Thêm/bớt Anime vào Watchlist.....	32
2.3. Xem danh sách Anime đã lưu.....	32
2.4. Xóa toàn bộ Watchlist.....	33
2.5. Khám phá Anime.....	33
3. Thiết kế cơ sở dữ liệu.....	34
3.1. Chi tiết các bảng.....	34
3.2. UML.....	35
CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH.....	36
1. Công cụ phát triển.....	36
1.1. Visual Studio Code.....	36
1.2. MySQL Workbench.....	36
2. Xây dựng Backend.....	37
2.1. Thiết kế RESTful API.....	37
2.2. Kết nối cơ sở dữ liệu.....	37
3. Xây dựng Frontend.....	38
3.1. Thiết kế giao diện.....	38
3.2. Tích hợp API.....	45
4. Kiểm thử chức năng.....	45
CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	47
1. Kết quả đạt được.....	47
2. Hạn chế.....	47
3. Hướng phát triển.....	48

LỜI MỞ ĐẦU

Trong bối cảnh ngành công nghiệp Anime phát triển mạnh mẽ trên toàn cầu, nhu cầu quản lý và theo dõi danh sách phim cá nhân trở nên ngày càng cần thiết. Với số lượng lớn các bộ Anime mới được sản xuất hàng năm, người xem cần một công cụ thuận tiện để lưu trữ, sắp xếp và tra cứu những bộ phim yêu thích, cũng như lập kế hoạch theo dõi.

Đề tài “Website quản lý danh sách Anime cá nhân (MyAnimeList)” được thực hiện nhằm xây dựng một hệ thống website hỗ trợ người dùng tạo tài khoản, đăng nhập, thêm hoặc xóa Anime khỏi danh sách cá nhân (watchlist), và tra cứu thông tin phim. Hệ thống còn cung cấp giao diện trực quan lấy cảm hứng từ nền tảng Netflix, giúp trải nghiệm người dùng trở nên hấp dẫn và dễ sử dụng.

Trong quá trình triển khai, em đã nghiên cứu các công nghệ phát triển web hiện đại như HTML/CSS/JavaScript, Node.js/Express và MySQL. Đồng thời, em cũng áp dụng các nguyên tắc thiết kế hệ thống và cơ sở dữ liệu nhằm đảm bảo tính mở rộng, bảo mật và dễ bảo trì.

Hy vọng rằng sản phẩm và báo cáo này sẽ đóng góp một phần ý nghĩa trong việc học tập và thực hành phát triển phần mềm, đồng thời mang đến trải nghiệm hữu ích cho người dùng yêu thích Anime. Em rất mong nhận được những ý kiến đóng góp quý báu từ quý thầy cô và bạn bè để hoàn thiện đề tài tốt hơn trong tương lai.

LỜI CẢM ƠN

Em xin chân thành cảm ơn sự hướng dẫn tận tình của các thầy cô tại Học viện Công nghệ Bưu chính Viễn thông - Viện Khoa học Kỹ thuật Bưu điện đã hỗ trợ và chỉ dẫn trong suốt quá trình thực hiện đề tài. Đặc biệt, em xin gửi lời cảm ơn đến thầy Vũ Văn Thương, người đã đồng hành, giải đáp thắc mắc và đưa ra những gợi ý quý báu giúp em hoàn thiện sản phẩm và báo cáo.

Em cũng xin gửi lời cảm ơn đến các bạn bè đã luôn sẵn sàng hỗ trợ, chia sẻ kinh nghiệm và động viên trong quá trình phát triển dự án.

Cuối cùng, em xin cảm ơn tất cả những người đã quan tâm và đóng góp ý kiến để đề tài này trở nên hoàn thiện và thực tiễn hơn.

Trân trọng!

MỞ ĐẦU

1. Giới thiệu đề tài

1.1. Bối cảnh và tầm quan trọng của quản lý thông tin cá nhân trên nền tảng web

Trong kỷ nguyên số hóa hiện nay, việc quản lý thông tin cá nhân thông qua các nền tảng web đã trở thành một phần không thể thiếu trong đời sống hàng ngày. Các ứng dụng web cung cấp khả năng truy cập mọi lúc, mọi nơi thông qua trình duyệt, thay thế các phương pháp ghi chép truyền thống và mang lại sự tiện lợi vượt trội. Người dùng có thể dễ dàng lưu trữ, sửa đổi và chia sẻ thông tin một cách an toàn và thuận tiện. Một ưu điểm nổi bật khác là các ứng dụng web tự động nhận các bản cập nhật phần mềm và bảo mật, giúp giảm thiểu rủi ro xâm phạm an ninh dữ liệu mà không yêu cầu người dùng phải can thiệp. Việc này giúp người dùng không phải lo lắng về việc cài đặt thủ công hay tiêu tốn dung lượng ổ cứng trên thiết bị cá nhân.

Tuy nhiên, sự tiện lợi này đi kèm với những thách thức đáng kể về bảo mật. Việc quản lý thông tin cá nhân trên nền tảng web đòi hỏi một hệ thống bảo mật vững chắc để chống lại các mối đe dọa như SQL Injection và Cross-Site Scripting (XSS). Các lỗ hổng này có thể cho phép kẻ tấn công chèn mã độc vào website, thực hiện các thao tác như xóa, chèn, cập nhật thông tin độc hại vào cơ sở dữ liệu, hoặc đánh cắp dữ liệu nhạy cảm của người dùng như mật khẩu và thông tin cá nhân. Hậu quả của việc thiếu bảo mật có thể rất nghiêm trọng, bao gồm việc mất quyền quản trị trang web, ảnh hưởng đến thứ hạng tìm kiếm trên Google, và gây thiệt hại tài chính đáng kể cho doanh nghiệp, đồng thời làm mất lòng tin của người dùng.

Sự phổ biến ngày càng tăng của các ứng dụng web trong việc quản lý thông tin cá nhân đã làm tăng đáng kể khối lượng dữ liệu nhạy cảm được lưu trữ trực tuyến. Điều này đồng thời làm tăng nguy cơ bị tấn công nếu các biện pháp bảo mật không được đảm bảo. Một hệ quả tất yếu là khi càng nhiều người dùng tin tưởng và sử dụng một nền tảng, thì yêu cầu về bảo mật lại càng phải cao hơn để duy trì lòng tin đó và tránh các rủi ro pháp lý cũng như tài chính. Đối với một dự án cá nhân, việc chú trọng bảo mật ngay từ đầu không chỉ là một kỹ thuật tốt mà còn là yếu tố then chốt để xây dựng một sản phẩm đáng tin cậy và chuyên nghiệp, có thể tự tin đưa vào hồ sơ năng lực. Việc này thể hiện sự nhận thức sâu sắc của nhà phát triển về trách nhiệm đối với dữ liệu của người dùng.

1.2. Tổng quan về MyAnimeList và nhu cầu phát triển ứng dụng tương tự

MyAnimeList (MAL) là một trong những nền tảng hàng đầu dành cho cộng đồng yêu thích Anime trên toàn cầu. Nền tảng này cung cấp một cơ sở dữ liệu khổng lồ về Anime và Manga, cùng với các công cụ mạnh mẽ giúp người dùng lưu trữ, theo dõi tiến độ xem, đánh giá và chia sẻ danh sách các bộ Anime đã xem hoặc muốn xem. Thành công vang dội của MAL đã chứng minh một nhu cầu lớn và rõ ràng về một hệ thống quản lý danh sách Anime cá nhân hiệu quả và tiện lợi trong cộng đồng người hâm mộ.

Dự án "Website MyAnimeList" này được khởi xướng nhằm mục đích mô phỏng các chức năng cốt lõi của MyAnimeList, tạo ra một nền tảng cá nhân cho phép người dùng quản lý trải nghiệm xem Anime của riêng họ. Việc phát triển một ứng dụng tương tự không chỉ giúp đáp ứng một phần nhu cầu của cộng đồng mà còn là một cơ hội học hỏi tuyệt vời. Đây là dịp để nhà phát triển rèn luyện các kỹ năng chuyên môn và xây dựng một sản phẩm thực tế, có thể hoạt động như một minh chứng cho năng lực cá nhân.

1.3. Mục tiêu và phạm vi của dự án

Dự án "Website MyAnimeList" được định hình bởi các mục tiêu rõ ràng và phạm vi cụ thể nhằm đảm bảo tính khả thi và hiệu quả trong quá trình phát triển.

Mục tiêu:

- Mục tiêu chính:** Phát triển một website cá nhân hoàn chỉnh với các chức năng cốt lõi bao gồm đăng ký và đăng nhập người dùng, khả năng quản lý Watchlist (thêm, xóa, cập nhật trạng thái các bộ Anime), cũng như các tính năng tìm kiếm và khám phá Anime.
- Mục tiêu kỹ thuật:** Dự án nhằm mục đích rèn luyện và củng cố các kỹ năng phát triển full-stack. Điều này bao gồm việc thành thạo các công nghệ frontend như HTML, CSS, JavaScript; xây dựng các API backend sử dụng Node.js và Express.js; và quản lý cơ sở dữ liệu hiệu quả với MySQL. Việc thực hiện dự án từ đầu đến cuối sẽ giúp nhà phát triển nắm vững quy trình phát triển từ giao diện người dùng đến logic xử lý dữ liệu.
- Mục tiêu cá nhân:** Tạo ra một sản phẩm hoàn chỉnh và có tính ứng dụng cao, có thể đưa vào hồ sơ năng lực (portfolio) cá nhân. Sản phẩm này sẽ đóng vai trò là một minh chứng cụ thể về kỹ năng, kinh nghiệm và khả năng giải quyết vấn đề của cá nhân, giúp nhà tuyển dụng dễ dàng hình dung năng

lực thực tế của ứng viên trong các buổi phỏng vấn hoặc đánh giá hồ sơ.

Phạm vi:

Dự án tập trung vào việc triển khai các chức năng cốt lõi của một hệ thống quản lý Anime cá nhân, đảm bảo tính khả thi trong khuôn khổ một dự án cá nhân.

Phạm vi bao gồm:

- **Quản lý người dùng:** Triển khai các chức năng cơ bản như đăng ký tài khoản mới, đăng nhập vào hệ thống, và quản lý thông tin tài khoản đơn giản.
- **Quản lý Watchlist:** Cho phép người dùng thêm các bộ Anime vào danh sách cá nhân của họ, xóa bỏ các mục không mong muốn.
- **Khám phá Anime:** Cung cấp các công cụ tìm kiếm Anime theo tên, lọc theo các tiêu chí cơ bản như thể loại hoặc điểm số, và hiển thị các bộ Anime ngẫu nhiên để người dùng khám phá.
- **Tích hợp API ngoài:** Sử dụng một API Anime công cộng, cụ thể là Jikan API, để lấy dữ liệu về Anime như thông tin chi tiết, hình ảnh, và tóm tắt.
- **Giao diện người dùng:** Xây dựng một giao diện web thân thiện, dễ sử dụng và có khả năng responsive, đảm bảo hiển thị tốt trên nhiều loại thiết bị khác nhau, từ máy tính đến điện thoại di động.



Logo website

2. Mục đích và ý nghĩa của đề tài

2.1. Mục đích

Dự án "Website MyAnimeList" được thực hiện với hai mục đích chính, hướng đến cả việc phát triển kỹ năng cá nhân và tạo ra một sản phẩm có giá trị thực tiễn.

Thứ nhất, dự án này là một môi trường thực tế lý tưởng để **rèn luyện và củng cố các kỹ năng phát triển full-stack**. Phát triển full-stack đòi hỏi sự thành thạo ở cả hai mảng frontend và backend để xây dựng và duy trì một ứng dụng web

hoàn chỉnh. Cụ thể, dự án này sẽ giúp nhà phát triển áp dụng và nâng cao kiến thức về thiết kế giao diện người dùng (frontend) với HTML, CSS, JavaScript; xây dựng các API và logic nghiệp vụ phía máy chủ (backend) với Node.js và Express.js; và quản lý cơ sở dữ liệu hiệu quả với MySQL. Quá trình này bao gồm việc đưa ra các quyết định kiến trúc, tối ưu hóa hiệu suất và đảm bảo bảo mật cho toàn bộ hệ thống.

Thứ hai, dự án nhằm mục đích **tạo ra một sản phẩm cá nhân để giới thiệu trong hồ sơ năng lực (portfolio) hoặc dùng để demo cho nhà tuyển dụng**. Trong ngành công nghệ thông tin, một portfolio chuyên nghiệp với các dự án thực tế là yếu tố then chốt giúp ứng viên nổi bật giữa các ứng viên khác. Dự án này sẽ là một minh chứng cụ thể về kỹ năng lập trình, kinh nghiệm thực chiến và khả năng giải quyết vấn đề của cá nhân. Việc có một sản phẩm hoạt động được để trình bày sẽ giúp nhà tuyển dụng dễ dàng hình dung năng lực của ứng viên, thay vì chỉ dựa vào mô tả lý thuyết trong CV.

2.2. Ý nghĩa

Dự án "Website MyAnimeList" không chỉ mang lại lợi ích cho nhà phát triển mà còn có ý nghĩa nhất định đối với người dùng tiêm năng và cộng đồng Anime.

Trước hết, dự án **cung cấp một công cụ quản lý danh sách Anime cá nhân, hỗ trợ tổ chức và theo dõi thói quen xem Anime**. Đối với những người đam mê Anime, việc có một công cụ tương tự MyAnimeList giúp họ dễ dàng quản lý bộ sưu tập phim của mình, theo dõi tiến độ xem, ghi nhớ những bộ đã hoàn thành, đang xem hoặc muốn xem. Điều này giúp tổ chức thói quen giải trí một cách có hệ thống và tiện lợi hơn so với các phương pháp thủ công.

Thứ hai, quá trình thực hiện dự án giúp **nâng cao hiểu biết và phát triển tư duy thiết kế hệ thống web và quản lý dữ liệu**. Việc xây dựng một ứng dụng từ đầu đến cuối đòi hỏi phải đưa ra các quyết định quan trọng về kiến trúc hệ thống, lựa chọn công nghệ phù hợp, và giải quyết các vấn đề kỹ thuật thực tế phát sinh. Điều này bao gồm việc phân tích yêu cầu người dùng, thiết kế cơ sở dữ liệu tối ưu để đảm bảo tính toàn vẹn và hiệu suất, cũng như xây dựng các API hiệu quả để frontend và backend giao tiếp liền mạch. Thông qua quá trình này, nhà phát triển sẽ phát triển một cách tiếp cận toàn diện hơn trong việc xây dựng các ứng dụng web phức tạp.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1. Ứng dụng website trong quản lý danh sách phim

1.1. Lợi ích của ứng dụng web trong quản lý dữ liệu cá nhân

Ứng dụng web đóng vai trò quan trọng trong việc quản lý dữ liệu cá nhân nhờ vào những ưu điểm vượt trội so với các phương pháp truyền thống. Một trong những lợi ích lớn nhất là khả năng truy cập. Người dùng có thể truy cập thông tin từ bất kỳ trình duyệt web nào, trên nhiều thiết bị khác nhau như máy tính để bàn, máy tính bảng và điện thoại di động, đảm bảo tính linh hoạt và khả năng tiếp cận mọi lúc mọi nơi. Điều này loại bỏ sự phụ thuộc vào một thiết bị cụ thể hoặc phần mềm cài đặt cục bộ.

Bên cạnh đó, ứng dụng web mang lại sự đơn giản đáng kể cho người dùng.

Người dùng không cần phải tải xuống hoặc cài đặt bất kỳ phần mềm nào, giúp họ dễ dàng truy cập và sử dụng ngay lập tức. Hơn nữa, người dùng cuối không cần phải lo lắng về việc bảo trì hay dung lượng ổ cứng, vì các ứng dụng web tự động nhận các bản cập nhật phần mềm và bảo mật. Điều này đảm bảo rằng ứng dụng luôn được cập nhật với các tính năng mới nhất và các biện pháp bảo mật được tăng cường, giảm thiểu rủi ro về xâm phạm an ninh.

Một lợi thế khác của ứng dụng web là khả năng mở rộng. Các doanh nghiệp (hoặc cá nhân phát triển dự án) có thể thêm người dùng khi cần mà không phải bổ sung cơ sở hạ tầng hoặc phần cứng tốn kém. Phần lớn dữ liệu của ứng dụng web được lưu trữ trên đám mây, điều này có nghĩa là không cần phải đầu tư thêm dung lượng lưu trữ cục bộ để chạy ứng dụng. Khả năng mở rộng này đặc biệt hữu ích cho các hệ thống quản lý danh sách như MyAnimeList, nơi người dùng có thể thêm hàng trăm, thậm chí hàng nghìn mục vào danh sách của họ theo thời gian.

Các lợi ích như khả năng truy cập rộng rãi, sự đơn giản cho người dùng và khả năng điều chỉnh quy mô trực tiếp cải thiện trải nghiệm người dùng. Khi người dùng không phải lo lắng về việc cài đặt, cập nhật phần mềm hay quản lý dung lượng lưu trữ, họ sẽ có xu hướng sử dụng ứng dụng thường xuyên hơn. Sự đơn giản trong việc sử dụng ở giao diện người dùng (frontend) được hỗ trợ bởi sự phức tạp được quản lý hiệu quả ở phía máy chủ (backend và lưu trữ đám mây). Điều này cho phép ứng dụng web cung cấp các chức năng phức tạp mà không yêu cầu cài đặt hay cấu hình phần mềm từ phía người dùng. Mối quan hệ này

giữa kiến trúc hệ thống và sự hài lòng của người dùng là một yếu tố quan trọng. Việc thiết kế ứng dụng web cần ưu tiên sự tiện lợi cho người dùng cuối, đồng thời đảm bảo rằng các quyết định kiến trúc backend hỗ trợ tối đa các lợi ích này, đặc biệt là khả năng mở rộng để đáp ứng sự tăng trưởng dữ liệu và số lượng người dùng trong tương lai.

1.2. So sánh với phương pháp truyền thống và các nền tảng hiện có

Trước khi có các ứng dụng web chuyên biệt, việc quản lý danh sách phim hoặc Anime thường được thực hiện bằng các phương pháp thủ công hoặc công cụ cục bộ. Các phương pháp này bao gồm ghi chép trên giấy, sử dụng bảng tính Excel, hoặc các phần mềm quản lý dữ liệu được cài đặt trực tiếp trên máy tính cá nhân. Những cách tiếp cận này có nhiều hạn chế đáng kể. Chúng thường gặp khó khăn trong việc chia sẻ thông tin với người khác, không thể truy cập từ xa khi người dùng không ở gần thiết bị lưu trữ, thiếu các tính năng tìm kiếm và lọc mạnh mẽ, và đặc biệt dễ bị mất dữ liệu do sự cố phần cứng hoặc lỗi người dùng.

Các nền tảng web hiện có như MyAnimeList đã khắc phục những nhược điểm này bằng cách cung cấp một giải pháp tập trung, có thể truy cập dễ dàng qua bất kỳ trình duyệt nào. Các nền tảng này mang lại sự tiện lợi, khả năng đồng bộ hóa dữ liệu và một cộng đồng người dùng rộng lớn. Tuy nhiên, mỗi nền tảng lại có những đặc điểm riêng về giao diện, bộ tính năng, và cộng đồng người dùng. Dự án MyAnimeList này không nhằm mục đích cạnh tranh trực tiếp với các nền tảng lớn đã có mặt trên thị trường. Thay vào đó, nó là một sản phẩm cá nhân, cho phép nhà phát triển kiểm soát hoàn toàn các tính năng, thiết kế, và quan trọng hơn là học hỏi sâu sắc về cách xây dựng một hệ thống quản lý dữ liệu phức tạp từ đầu đến cuối. Đây là một cơ hội để khám phá các khía cạnh kỹ thuật và thiết kế mà có thể không được tiếp cận khi chỉ sử dụng các nền tảng có sẵn.

2. Giới thiệu về thể loại Anime và cộng đồng người xem

2.1. Lịch sử và sự phát triển của Anime

Anime là thuật ngữ chỉ các bộ phim hoạt hình có nguồn gốc từ Nhật Bản, nổi tiếng với phong cách đồ họa đặc trưng, cốt truyện đa dạng và chiều sâu cảm xúc. Lịch sử của Anime bắt đầu từ những năm đầu thế kỷ 20, với những bộ phim ngắn thử nghiệm mang ảnh hưởng từ hoạt hình phương Tây. Trải qua nhiều thập kỷ, Anime đã phát triển mạnh mẽ, từ những series truyền hình dài tập như "Astro Boy" (1963) đến các phim điện ảnh bom tấn đạt giải thưởng

quốc tế như "Spirited Away" (2001) hay "Your Name" (2016).



Anime

Hiện nay, có hàng ngàn bộ Anime thuộc đủ mọi thể loại, từ hành động, lãng mạn, giả tưởng, khoa học viễn tưởng, đến đời thường và tâm lý xã hội. Sự đa dạng về nội dung và phong cách nghệ thuật đã giúp Anime trở thành một hiện tượng văn hóa toàn cầu, thu hút hàng triệu người hâm mộ trên khắp thế giới và tạo ra một ngành công nghiệp giải trí không lồ.

2.2. Đặc điểm cộng đồng người xem và nhu cầu của họ

Cộng đồng người xem Anime trên toàn cầu rất lớn và đa dạng về độ tuổi, sở thích và quốc tịch, đặc biệt phổ biến ở châu Á, Bắc Mỹ và châu Âu. Những người hâm mộ Anime thường có nhu cầu cao trong việc quản lý và tương tác với nội dung mà họ yêu thích.

Các nhu cầu chính của cộng đồng người xem bao gồm:

- **Theo dõi tiến độ xem:** Người hâm mộ thường xem nhiều bộ Anime cùng lúc hoặc theo dõi các series dài tập. Do đó, họ cần một công cụ để ghi nhớ những bộ đã xem, đang xem, hoặc muốn xem trong tương lai.
- **Khám phá Anime mới:** Với hàng ngàn bộ Anime được phát hành mỗi năm, người dùng luôn tìm kiếm các bộ mới theo thể loại yêu thích, dựa trên điểm số đánh giá cao, hoặc theo đề xuất từ cộng đồng.
- **Lưu trữ thông tin:** Có một nơi tập trung để lưu trữ thông tin về các bộ Anime yêu thích, bao gồm tóm tắt, hình ảnh, thể loại, và các ghi chú cá nhân.
- **Tương tác cộng đồng (tùy chọn):** Nhiều người dùng mong muốn có khả năng đánh giá Anime, viết bình luận, và chia sẻ cảm nghĩ của mình với những người hâm mộ khác, tạo nên một môi trường trao đổi sôi nổi.

Dự án này tập trung vào việc đáp ứng các nhu cầu cốt lõi về quản lý và khám phá Anime, tạo ra một công cụ hữu ích và cá nhân hóa cho người dùng.

3. Các chức năng phổ biến của hệ thống quản lý Anime

3.1. Phân tích các tính năng cốt lõi (Cơ sở dữ liệu, quản lý danh sách, đánh giá, tìm kiếm)

Một hệ thống quản lý Anime hiệu quả và toàn diện thường bao gồm một tập hợp các tính năng cốt lõi được thiết kế để phục vụ tối đa nhu cầu của người dùng. Các tính năng này là nền tảng cho trải nghiệm người dùng mượt mà và thông tin đầy đủ.

- **Cơ sở dữ liệu Anime phong phú:** Đây là trái tim của mọi hệ thống quản lý Anime. Nó phải chứa thông tin chi tiết về hàng ngàn bộ Anime, bao gồm tiêu đề tiếng Anh và tiếng Nhật, thể loại (ví dụ: hành động, lãng mạn, giả tưởng, khoa học viễn tưởng), tóm tắt cốt truyện, hình ảnh đại diện (poster, ảnh bìa), điểm số đánh giá trung bình, trạng thái phát sóng (đang chiếu, đã hoàn thành), số tập, và thông tin về nhà sản xuất hoặc studio.
- **Quản lý danh sách cá nhân:** Tính năng này cho phép người dùng tạo và quản lý các danh sách tùy chỉnh để theo dõi tiến độ xem của họ. Các danh sách phổ biến bao gồm "Watchlist" (đang xem hoặc muốn xem), "Completed" (đã hoàn thành), "Dropped" (đã bỏ dở), "On-Hold" (tạm dừng), và "Favorites" (yêu thích). Người dùng có thể dễ dàng thêm, xóa hoặc cập nhật trạng thái của một bộ Anime trong các danh sách này.
- **Đánh giá, bình luận và chia sẻ:** Để khuyến khích tương tác cộng đồng, hệ thống thường cung cấp công cụ để người dùng đánh giá Anime (ví dụ: theo thang điểm 1-10), viết bình luận hoặc nhận xét về các bộ phim, và chia sẻ cảm nghĩ của mình với những người khác trong cộng đồng. Tính năng này giúp người dùng khám phá các bộ Anime mới dựa trên ý kiến của người khác và tạo ra một môi trường trao đổi sôi nổi.
- **Tìm kiếm, lọc và gợi ý:** Để giúp người dùng dễ dàng tìm thấy Anime mong muốn, hệ thống cần có các chức năng tìm kiếm mạnh mẽ (theo tên, từ khóa). Ngoài ra, khả năng lọc theo nhiều tiêu chí khác nhau như thể loại, năm phát hành, điểm số, hoặc trạng thái phát sóng là cần thiết. Một số hệ thống còn cung cấp tính năng gợi ý ngẫu nhiên hoặc gợi ý cá nhân hóa dựa trên lịch sử xem của người dùng.

3.2. Các API Anime công cộng và lựa chọn cho dự án

Để có được dữ liệu Anime phong phú mà không cần tự xây dựng một cơ sở dữ liệu khổng lồ từ đầu, việc tích hợp với các API Anime công cộng là một giải pháp hiệu quả và tiết kiệm thời gian. Có một số API phổ biến trong cộng đồng phát triển Anime, mỗi API có những đặc điểm riêng:

- **Jikan API:** Đây là một API REST không chính thức cho MyAnimeList.net. Jikan hoạt động bằng cách scraping dữ liệu từ website MyAnimeList để cung cấp chức năng API mà MAL còn thiếu. Một điểm quan trọng cần lưu ý là Jikan chỉ hỗ trợ các yêu cầu không xác thực (non-authenticated requests), nghĩa là không thể dùng để cập nhật danh sách của người dùng trên MyAnimeList. Tuy nhiên, nó cung cấp các endpoint mạnh mẽ để truy xuất dữ liệu công cộng như tìm kiếm Anime, lọc theo thể loại, và lấy Anime ngẫu nhiên. Jikan có giới hạn tốc độ là 30 yêu cầu mỗi phút và 2 yêu cầu mỗi giây.
- **AniList API:** Cung cấp quyền truy cập mạnh mẽ vào hơn 0 nghìn mục Anime và Manga, bao gồm dữ liệu nhân vật, nhân viên và lịch phát sóng trực tiếp. AniList sử dụng GraphQL, một công nghệ cho phép người dùng truy vấn và thao tác dữ liệu một cách linh hoạt và hiệu quả hơn so với RESTful API truyền thống. AniList và AniChart đều được xây dựng trên cùng một API, đảm bảo tính tương thích và độ tin cậy cao.
- **Kitsu API:** Đây là một nền tảng khám phá Anime khác, cũng cung cấp API. Kitsu API cung cấp các endpoint cho xác thực người dùng (đăng nhập, đăng ký, thay đổi mật khẩu) và quản lý tài sản (assets)

Lựa chọn cho dự án:

Dự án này lựa chọn Jikan API để lấy dữ liệu Anime công cộng. Lý do chính là Jikan cung cấp một nguồn dữ liệu phong phú và cập nhật từ MyAnimeList, đồng thời có sẵn các wrapper cho nhiều ngôn ngữ lập trình, giúp việc tích hợp trở nên dễ dàng.



Logo Jikan API

Việc lựa chọn Jikan API có tác động đáng kể đến thiết kế hệ thống. Jikan API không hỗ trợ các yêu cầu xác thực để cập nhật danh sách người dùng. Điều này tạo ra một rào cản lớn nếu dự án muốn quản lý watchlist của người dùng trực tiếp thông qua Jikan. Tuy nhiên, dự án này lại cần chức năng quản lý watchlist cá nhân. Do đó, một hệ quả quan trọng là dự án buộc phải lưu trữ dữ liệu watchlist của người dùng *nội bộ* trong cơ sở dữ liệu MySQL của riêng mình. Jikan sẽ chỉ được sử dụng để *truy xuất* thông tin Anime công cộng (tìm kiếm, chi tiết, ngẫu nhiên), không phải để quản lý danh sách cá nhân. Điều này có nghĩa là bảng Animes trong cơ sở dữ liệu của dự án sẽ cần lưu trữ thông tin cơ bản của Anime (được lấy từ Jikan) để hiển thị, và bảng Watchlist sẽ liên kết user_id với anime_id (từ bảng Animes nội bộ). Việc hiểu rõ giới hạn của API bên ngoài là cực kỳ quan trọng trong thiết kế hệ thống, vì nó định hình kiến trúc dữ liệu và logic nghiệp vụ. Đối với Jikan, việc có giới hạn tốc độ (30 yêu cầu/phút, 2 yêu cầu/giây) cũng ngụ ý rằng backend cần có cơ chế caching để tránh gọi API quá thường xuyên, nâng cao hiệu suất và tuân thủ chính sách của Jikan.

Bảng 1: So sánh các API Anime chính

Tiêu chí	Jikan API (v4)	AniList API (GraphQL)	Kitsu API
Nguồn dữ liệu	MyAnimeList.net (không chính thức, scraping)	Cơ sở dữ liệu riêng (hơn 500k mục)	Nền tảng khám phá Anime
Loại API	RESTful API	GraphQL	RESTful API
Yêu cầu xác thực	Không hỗ trợ cho việc cập nhật danh sách người dùng	Hỗ trợ (GraphQL mutations cho quản lý danh sách)	Hỗ trợ (endpoints auth/login, auth/register)
Giới hạn tốc độ	30 yêu cầu/phút, 2 yêu cầu/giây	Có giới hạn nhưng thường cao hơn hoặc linh hoạt hơn	Có giới hạn (cần kiểm tra tài liệu)

		(cần kiểm tra tài liệu)	
Tính năng chính	Truy xuất dữ liệu công cộng (tìm kiếm, lọc, ngẫu nhiên)	Truy vấn linh hoạt, quản lý danh sách, cộng đồng	Xác thực, quản lý tài sản, có thể có quản lý danh sách (cần kiểm tra thêm)
Giấy phép	MIT License	Không rõ trong tài liệu	Không rõ trong tài liệu

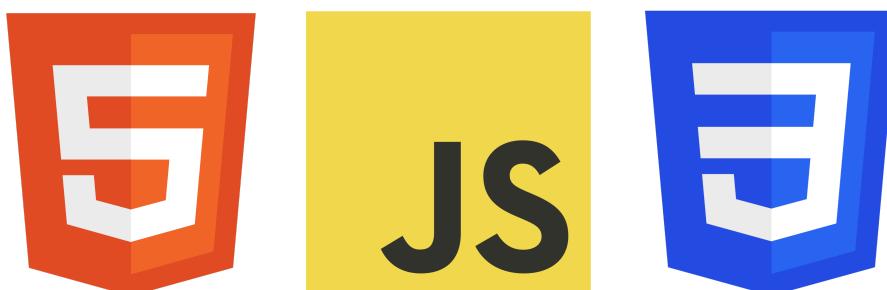
Bảng này cung cấp một cái nhìn tổng quan so sánh các API Anime phổ biến, giúp người đọc hiểu rõ hơn về lý do lựa chọn Jikan API cho dự án này (chủ yếu cho dữ liệu công cộng) và những hạn chế của nó, đặc biệt là việc không hỗ trợ cập nhật danh sách người dùng. Điều này cũng cố luận điểm về việc cần xây dựng cơ sở dữ liệu nội bộ cho watchlist.

4. Công nghệ phát triển website

Dự án phát triển website MyAnimeList sử dụng một bộ công nghệ full-stack phổ biến và mạnh mẽ, bao gồm HTML, CSS, JavaScript cho frontend; Node.js và Express.js cho backend; và MySQL cho cơ sở dữ liệu. Sự lựa chọn này nhằm mục đích tận dụng lợi thế của JavaScript trên toàn bộ stack, từ đó tối ưu hóa quá trình phát triển và bảo trì.

4.1. HTML/CSS/JavaScript (Frontend)

Frontend là phần giao diện mà người dùng tương tác trực tiếp. Để xây dựng một giao diện web hiện đại, ba công nghệ nền tảng không thể thiếu là HTML, CSS và JavaScript.



Logo HTML JS CSS

4.1.1. Vai trò của HTML trong cấu trúc giao diện

HTML (Hypertext Markup Language) là ngôn ngữ đánh dấu tiêu chuẩn để tạo cấu trúc và nội dung cho các trang web. Nó định nghĩa các thành phần cơ bản như tiêu đề, đoạn văn, hình ảnh, liên kết, bảng biểu, và các form nhập liệu.

HTML không phải là một ngôn ngữ lập trình, do đó không có khả năng tạo ra các chức năng "động" mà chỉ tập trung vào việc bố cục và định dạng nội dung. Trong dự án này, HTML được sử dụng để xây dựng khung sườn của các trang như trang chủ hiển thị danh sách Anime, trang đăng nhập/đăng ký, trang Watchlist cá nhân, và trang chi tiết Anime. Việc sử dụng HTML5 mang lại nhiều ưu điểm như hỗ trợ tốt hơn cho các ứng dụng web, tính năng lưu dữ liệu tạm thời, và tích hợp các element mới cho đa phương tiện và cấu trúc ngữ nghĩa, giúp trang web hiện đại và dễ quản lý hơn.

4.1.2. Vai trò và ưu điểm của CSS trong thiết kế và responsive

CSS (Cascading Style Sheets) là ngôn ngữ được sử dụng để định dạng và tạo kiểu cho các tài liệu HTML. Nó kiểm soát cách các phần tử HTML được hiển thị trên màn hình, bao gồm màu sắc, phông chữ, bố cục, khoảng cách, và hiệu ứng. Mỗi quan hệ giữa HTML và CSS là vô cùng mật thiết; không có CSS, website sẽ chỉ là một trang trắng chứa đầy văn bản nhảm chán và thiếu tính thẩm mỹ.

Các ưu điểm chính của CSS bao gồm:

- **Tiết kiệm thời gian và dễ bảo trì:** Thay vì định dạng từng phần tử HTML riêng lẻ, CSS cho phép định nghĩa kiểu dáng một lần và áp dụng cho nhiều trang hoặc nhiều phần tử. Điều này giúp tiết kiệm thời gian phát triển và dễ dàng thay đổi kiểu dáng trên toàn bộ website chỉ bằng cách sửa đổi một file CSS duy nhất.
- **Tốc độ tải trang nhanh chóng:** Việc sử dụng CSS hiệu quả giúp giảm lượng mã cần thiết trong HTML, từ đó tăng tốc độ tải trang, cải thiện trải nghiệm người dùng.
- **Khả năng tương thích đa thiết bị (Responsive Design):** CSS có khả năng tương thích với nhiều thiết bị khác nhau, cho phép thiết kế website hiển thị tốt trên cả desktop, tablet và mobile thông qua các kỹ thuật như Media Queries. Điều này đảm bảo rằng giao diện luôn thân thiện và dễ sử dụng bất kể kích thước màn hình.

4.1.3. Vai trò và chức năng của JavaScript trong tương tác người dùng và DOM Manipulation

JavaScript là ngôn ngữ lập trình phổ biến nhất thế giới và là một trong ba ngôn ngữ cốt lõi của lập trình web, cùng với HTML và CSS. Nó mang lại tính động và tương tác cho website, cho phép kiểm soát hành vi của trang web tốt hơn so với việc chỉ dùng HTML. Các chức năng chính của JavaScript trong dự án này bao gồm:

- **Xử lý tương tác người dùng:** JavaScript phản ứng với các sự kiện của người dùng như nhấp chuột vào nút, di chuyển chuột qua các phần tử, hoặc nhập liệu vào các trường form. Điều này cho phép tạo ra các trải nghiệm động như hiển thị/ẩn nội dung, xác thực form ngay lập tức, và các hiệu ứng chuyển động mượt mà trên giao diện người dùng.
- **Thao tác DOM (Document Object Model):** DOM là một giao diện lập trình cho các tài liệu web, biểu diễn cấu trúc HTML dưới dạng các nút và đối tượng. JavaScript sử dụng DOM API để truy cập và thay đổi cấu trúc, kiểu dáng và nội dung của tài liệu một cách linh hoạt. Ví dụ, các phương thức như `document.querySelector()` được dùng để chọn các phần tử HTML, `addEventListener()` để lắng nghe các sự kiện, `textContent` để thay đổi nội dung văn bản, và `createElement()` cùng `appendChild()/removeChild()` để thêm hoặc xóa các phần tử động.
- **Xác thực đầu vào:** JavaScript có thể được sử dụng để kiểm tra tính hợp lệ của dữ liệu nhập vào từ người dùng ngay trên trình duyệt (client-side validation). Điều này giúp giảm thiểu việc gửi các yêu cầu không hợp lệ lên máy chủ, cải thiện hiệu suất và giảm tải cho backend.

Sự kết hợp của HTML, CSS và JavaScript cho phép xây dựng một giao diện người dùng phong phú và thân thiện. HTML cung cấp cấu trúc, CSS cung cấp kiểu dáng, nhưng JavaScript mới là yếu tố mang lại sự sống động và tương tác cho trang web. Khả năng thao tác DOM của JavaScript là chìa khóa để tạo ra giao diện người dùng động, phản hồi nhanh chóng với các hành động của người dùng. Thông qua DOM, JavaScript biến một trang web tĩnh thành một ứng dụng web tương tác, nơi người dùng có thể thêm/xóa anime, tìm kiếm động mà không cần tải lại trang. Điều này trực tiếp cải thiện "tính đơn giản cho người dùng" và "khả năng truy cập" đã đề cập ở mục 1.1. Để đạt được trải nghiệm người dùng tối ưu, nhà phát triển cần có sự hiểu biết sâu sắc về cách ba công nghệ này tương tác và hỗ trợ cho nhau. Việc tối ưu hóa DOM manipulation và xử lý sự kiện là cần thiết để tránh tình trạng giao diện bị giật lag hoặc phản hồi chậm.

4.1.4. Tích hợp API với Fetch/Axios và xử lý bất đồng bộ

Để giao tiếp với backend và các API bên ngoài (như Jikan API), frontend cần các cơ chế để thực hiện các yêu cầu HTTP.

- **Fetch API:** Là một giao diện hiện đại trong JavaScript cho phép thực hiện các yêu cầu HTTP một cách bất đồng bộ. Nó thay thế phương thức XMLHttpRequest cũ hơn và cung cấp một cách tiếp cận dựa trên Promise, giúp xử lý dữ liệu bất đồng bộ dễ dàng hơn. Phương thức fetch() trả về một Promise mà khi thành công sẽ phân giải thành một đối tượng Response. Sau đó, cần gọi các phương thức như .json() (hoặc .text(), .blob(), v.v.) trên đối tượng Response để phân tích cú pháp nội dung phản hồi thành định dạng mong muốn.
- **Axios:** Là một thư viện HTTP client phổ biến dựa trên Promise cho cả trình duyệt và môi trường Node.js. Axios thường được ưa chuộng hơn Fetch API trong một số trường hợp vì nó cung cấp một API dễ sử dụng hơn, bao gồm việc tự động chuyển đổi JSON, xử lý lỗi tốt hơn (tự động từ chối Promise khi có lỗi HTTP 4xx/5xx thay vì chỉ lỗi mạng), và khả năng hủy yêu cầu.
- **Xử lý bất đồng bộ (Async/Await):** Cả Fetch và Axios đều hoạt động với Promise, cho phép sử dụng cú pháp async/await để viết code bất đồng bộ trông giống như code đồng bộ, giúp dễ đọc và dễ bảo trì hơn. Cú pháp này cho phép tạm dừng việc thực thi hàm cho đến khi một Promise được giải quyết, giúp quản lý luồng dữ liệu phức tạp dễ dàng hơn.

Một điểm quan trọng cần lưu ý là Fetch API chỉ từ chối Promise khi có lỗi mạng (ví dụ: URL không tồn tại, người dùng offline), chứ không phải lỗi HTTP như 404 (Không tìm thấy) hay 500 (Lỗi máy chủ nội bộ). Điều này đòi hỏi nhà phát triển phải chủ động kiểm tra thuộc tính response.ok của đối tượng Response. Nếu response.ok là false, điều đó có nghĩa là yêu cầu không thành công về mặt HTTP, và cần phải xử lý lỗi một cách tường minh, ví dụ bằng cách ném một Error để được bắt bởi khối .catch(). Nếu không kiểm tra response.ok, ứng dụng frontend có thể cố gắng xử lý dữ liệu từ một phản hồi lỗi, dẫn đến lỗi ứng dụng hoặc trải nghiệm người dùng kém. Việc sử dụng async/await giúp cấu trúc code rõ ràng hơn cho việc kiểm tra này, dẫn đến code frontend mạnh mẽ và đáng tin cậy hơn. Kỹ năng xử lý bất đồng bộ và quản lý trạng thái tải/lỗi là tối quan trọng cho các ứng dụng web hiện đại, đặc biệt khi tương tác với API. Việc cung cấp phản hồi trực quan cho người dùng (ví dụ: loading spinners, thông báo lỗi rõ ràng) trong quá trình này là một thực tiễn tốt về thiết kế UI/UX.

4.2. Node.js / Express (Backend)

Backend là "bộ não" của ứng dụng, chịu trách nhiệm xử lý logic nghiệp vụ, tương tác với cơ sở dữ liệu và cung cấp các API cho frontend.

4.2.1. Kiến trúc và ưu điểm của Node.js

Node.js là một môi trường runtime JavaScript mã nguồn mở và đa nền tảng, cho phép lập trình viên chạy mã JavaScript bên ngoài trình duyệt web. Nó được xây dựng trên V8 JavaScript Engine của Google Chrome, giúp biên dịch JavaScript thành mã máy rất nhanh chóng, từ đó đạt được hiệu suất cao.



Logo NodeJS

Kiến trúc độc đáo của Node.js dựa trên mô hình **đơn luồng (single-threaded)** kết hợp với cơ chế **I/O không chặn (non-blocking I/O)** và **vòng lặp sự kiện (event loop)**. Khi một tác vụ I/O (ví dụ: đọc file, truy vấn cơ sở dữ liệu) được gửi đi, Node.js không chờ kết quả trả về mà ngay lập tức chuyển sang xử lý các yêu cầu khác. Khi tác vụ I/O hoàn thành, một callback sẽ được đẩy vào hàng đợi và được thực thi khi luồng chính sẵn sàng. Cơ chế này cho phép Node.js xử lý hàng nghìn yêu cầu đồng thời mà không gặp vấn đề nghẽn tài nguyên, mặc dù chỉ sử dụng một luồng chính.

Các ưu điểm nổi bật của Node.js bao gồm:

- **Hiệu suất cao và khả năng mở rộng tốt:** Nhờ mô hình non-blocking I/O và event loop, Node.js xử lý nhiều kết nối đồng thời hiệu quả. Điều này làm cho nó rất phù hợp cho các kiến trúc microservices và dễ dàng mở rộng hệ thống theo chiều ngang (horizontal scaling) để đáp ứng lượng người dùng tăng lên.
- **Sử dụng JavaScript cho toàn bộ dự án (Full-stack JavaScript):** Một lợi thế lớn là nhà phát triển có thể dùng cùng một ngôn ngữ (JavaScript) cho cả frontend và backend. Điều này giúp giảm độ phức tạp của dự án, dễ dàng

chia sẻ thư viện và kiến thức giữa các phần, từ đó tăng năng suất phát triển.

- **Hệ sinh thái npm phong phú:** Node Package Manager (npm) cung cấp một kho lưu trữ khổng lồ với hàng triệu gói và module miễn phí, đáp ứng hầu hết các nhu cầu phát triển. Điều này giúp xây dựng ứng dụng nhanh chóng với nhiều tính năng mà không cần phải tự viết lại từ đầu.
- **Khả năng xử lý real-time tốt:** Node.js rất phù hợp cho các ứng dụng cần cập nhật dữ liệu liên tục như ứng dụng chat, game online, nhờ hỗ trợ WebSocket.

Mô hình đơn luồng của Node.js có thể bị nhầm lẫn là một hạn chế về hiệu suất. Tuy nhiên, các phân tích chỉ ra rằng nhờ I/O không chặn và vòng lặp sự kiện, Node.js lại vượt trội trong việc xử lý đồng thời nhiều kết nối, đặc biệt là các tác vụ I/O-bound (như truy vấn cơ sở dữ liệu, gọi API bên ngoài). Việc sử dụng Node.js cho backend của MyAnimeList sẽ giúp ứng dụng xử lý nhanh chóng các yêu cầu từ nhiều người dùng đồng thời khi họ thêm/xóa anime vào watchlist hoặc tìm kiếm dữ liệu. Điều này trực tiếp cải thiện hiệu suất và khả năng mở rộng của ứng dụng. Lựa chọn Node.js là một quyết định chiến lược cho các ứng dụng web hiện đại cần khả năng phản hồi nhanh và xử lý lượng lớn kết nối, phù hợp với một ứng dụng có nhiều tương tác người dùng như dự án này. Tuy nhiên, cần lưu ý rằng mô hình đơn luồng có thể trở nên kém hiệu quả đối với các tác vụ CPU-intensive (như xử lý hình ảnh phức tạp hoặc tính toán nặng). Đối với những trường hợp này, cần kết hợp với các kỹ thuật như worker threads, mặc dù điều này ít liên quan đến một ứng dụng CRUD cơ bản như dự án này.

4.2.2. Ưu điểm của Express trong xây dựng RESTful API

Express.js là một framework web tối giản và linh hoạt cho Node.js, cung cấp một bộ tính năng mạnh mẽ để xây dựng các ứng dụng web và API. Nó là một trong những framework phổ biến nhất trong hệ sinh thái Node.js.



Logo ExpressJS

Các ưu điểm chính của Express.js bao gồm:

- **Đơn giản, nhẹ và dễ cấu hình:** Express được thiết kế để tối giản, giúp nhà phát triển nhanh chóng bắt đầu xây dựng ứng dụng mà không gặp nhiều rào

cần hay cần cấu hình phức tạp.

- **Hỗ trợ middleware mạnh mẽ:** Middleware là các hàm có quyền truy cập vào đối tượng yêu cầu (request), đối tượng phản hồi (response) và hàm middleware tiếp theo trong chu trình yêu cầu-phản hồi của ứng dụng. Điều này cho phép code được module hóa và tái sử dụng cho các tác vụ như xác thực, ghi log, phân tích cú pháp dữ liệu, và xử lý lỗi.
- **Phù hợp cho RESTful API:** Express.js đặc biệt thích hợp để xây dựng các RESTful API nhờ khả năng định tuyến rõ ràng và hỗ trợ middleware. Tính chất nhẹ của nó cũng là lựa chọn tuyệt vời cho các ứng dụng web quy mô nhỏ đến trung bình, nơi hiệu suất và tốc độ phát triển là yếu tố then chốt.
- **Ứng dụng rộng rãi:** Express.js được sử dụng rộng rãi để xây dựng backend API, ứng dụng web real-time và microservices, tận dụng hiệu quả và tính linh hoạt của Node.js.

4.2.3. Các nguyên tắc thiết kế RESTful API

REST (Representational State Transfer) là một tập hợp các nguyên tắc kiến trúc để thiết kế các giao diện lập trình ứng dụng (API) web hiệu quả và có khả năng mở rộng. Việc tuân thủ các nguyên tắc này giúp API dễ hiểu, dễ sử dụng và dễ bảo trì, đồng thời tối ưu hóa hiệu suất và khả năng mở rộng của hệ thống.



Các nguyên tắc cốt lõi của thiết kế RESTful API bao gồm:

- **Kiến trúc Client-Server (Client-Server Architecture):** Trong mô hình này, client và server hoạt động độc lập với nhau. Client (ví dụ: trình duyệt web, ứng dụng di động) chịu trách nhiệm về giao diện người dùng và khởi tạo các yêu cầu, trong khi server xử lý dữ liệu và gửi phản hồi. Sự tách biệt này giúp tăng tính linh hoạt trong phát triển và khả năng bảo trì lâu dài cho cả hai phía.
- **Không trạng thái (Stateless Communication):** Mỗi yêu cầu từ client đến server phải chứa tất cả thông tin cần thiết để server xử lý yêu cầu đó, mà không cần server lưu trữ bất kỳ dữ liệu phiên cụ thể nào của client giữa các

yêu cầu. Điều này đơn giản hóa việc mở rộng ngang của server API, vì mỗi yêu cầu là độc lập và có thể được xử lý bởi bất kỳ server nào, đồng thời giảm chi phí bộ nhớ. Ví dụ, mỗi yêu cầu API cần bao gồm một token xác thực (như JSON Web Token) thay vì dựa vào một session ID được lưu trên server.

- **Giao diện đồng nhất (Uniform Interface):** Nguyên tắc này quy định một tập hợp các quy tắc chuẩn để tương tác với API. Điều này bao gồm việc sử dụng nhất quán các phương thức HTTP (GET, POST, PUT, DELETE) cho các thao tác CRUD (Create, Read, Update, Delete), các mẫu URL chuẩn, và định dạng phản hồi dễ đoán. Một giao diện đồng nhất giúp API dễ sử dụng hơn cho nhà phát triển, giảm thiểu thời gian học hỏi và khả năng xảy ra lỗi.
- **Có thể lưu vào bộ nhớ đệm (Cacheable Responses):** Các phản hồi từ API RESTful có thể được lưu vào bộ nhớ đệm (cache), cho phép client hoặc các trung gian lưu trữ chúng trong một khoảng thời gian nhất định. Điều này giảm thiểu nhu cầu gửi các yêu cầu lặp lại cho cùng một dữ liệu, cải thiện độ trễ và giảm tải cho server, từ đó nâng cao hiệu suất và trải nghiệm người dùng.
- **Hệ thống phân lớp (Layered System):** Nguyên tắc này cho phép API hoạt động qua nhiều lớp (ví dụ: các máy chủ trung gian, bộ cân bằng tải, cổng bảo mật), mà client không cần biết về sự tồn tại của các lớp bên dưới. Cách tiếp cận này nâng cao bảo mật và khả năng mở rộng, cho phép thay đổi hoặc thêm các lớp mà không ảnh hưởng đến client.

Việc tuân thủ các nguyên tắc RESTful API không chỉ là một lý thuyết mà là một thực hành quan trọng để xây dựng API mạnh mẽ và bền vững. Ví dụ, tính "không trạng thái" (statelessness) là nền tảng cho việc sử dụng JSON Web Tokens (JWT) để quản lý phiên, vì JWT tự chứa thông tin cần thiết mà không yêu cầu server phải lưu trữ trạng thái phiên. Các nguyên tắc RESTful API, đặc biệt là tính không trạng thái và giao diện đồng nhất, trực tiếp hỗ trợ khả năng mở rộng và dễ bảo trì của backend. Khi API được thiết kế nhất quán, việc thêm các tính năng mới hoặc mở rộng hệ thống sẽ dễ dàng hơn, giảm thiểu "technical debt" (nợ kỹ thuật) trong tương lai. Một API được thiết kế tốt theo nguyên tắc RESTful là dấu hiệu của một kiến trúc sư phần mềm có kinh nghiệm, điều này làm tăng giá trị của dự án trong hồ sơ năng lực. Nó cho thấy không chỉ khả năng viết mã mà còn là khả năng thiết kế hệ thống có tầm nhìn dài hạn.

4.3. MySQL (Database)

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở (RDBMS) hoạt động theo mô hình client-server. Nó là một lựa chọn phổ biến cho các ứng dụng web nhờ tốc độ cao, độ bảo mật tốt và khả năng tương thích với nhiều ngôn ngữ lập trình như Node.js, PHP, Perl.



Logo MySQL

4.3.1. Khái niệm cơ sở dữ liệu quan hệ và MySQL

Trong một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS), dữ liệu được tổ chức thành các bảng (còn gọi là quan hệ), với mỗi bảng chứa các hàng (bản ghi) và cột (trường dữ liệu). Các bảng này được liên kết với nhau thông qua các khóa chính (Primary Key) và khóa ngoại (Foreign Key), tạo nên các mối quan hệ giữa các tập dữ liệu. Cơ chế hoạt động của MySQL là tạo bảng để lưu trữ dữ liệu và định nghĩa mối quan hệ giữa chúng. Sau đó, client sẽ gửi yêu cầu SQL bằng một lệnh đặc biệt, và server sẽ phản hồi thông tin và trả về kết quả cho máy client.

MySQL có nhiều ưu điểm nổi bật, làm cho nó trở thành lựa chọn phù hợp cho dự án này:

- **Nhanh chóng và hiệu quả:** MySQL hoạt động hiệu quả và tiết kiệm chi phí, giúp tăng tốc độ thực thi các truy vấn dữ liệu.
- **Mạnh mẽ và khả năng mở rộng:** MySQL có thể xử lý lượng lớn dữ liệu và dễ dàng mở rộng khi nhu cầu tăng lên.
- **Đa tính năng:** Hỗ trợ nhiều chức năng SQL mong đợi từ một hệ quản trị cơ sở dữ liệu quan hệ, cung cấp một hệ thống lớn các hàm tiện ích mạnh mẽ.
- **Độ bảo mật cao:** Thích hợp cho các ứng dụng truy cập cơ sở dữ liệu qua internet nhờ sở hữu nhiều tính năng bảo mật cấp cao.
- **Dễ dàng sử dụng và ổn định:** MySQL là một cơ sở dữ liệu dễ sử dụng, ổn định, tốc độ cao và hoạt động trên nhiều hệ điều hành.
- **Miễn phí:** Phiên bản MySQL Community Server là miễn phí, rất phù hợp cho các dự án cá nhân hoặc các doanh nghiệp nhỏ có ngân sách hạn chế.

Tuy nhiên, MySQL cũng có một số hạn chế cần lưu ý cho các ứng dụng quy mô rất lớn: dung lượng có thể bị hạn chế khi số lượng bản ghi quá lớn, đòi hỏi các biện pháp như tạo cache MySQL hoặc chia tải database ra nhiều server. Ngoài ra, độ tin cậy của MySQL trong cách thức xử lý các chức năng cụ thể như kiểm toán hay giao dịch phức tạp có thể kém hơn một số RDBMS khác. Đối với quy mô của một dự án cá nhân MyAnimeList, những hạn chế này không phải là vấn đề lớn.

4.3.2. Vai trò của Primary Key và Foreign Key trong quan hệ dữ liệu

Trong MySQL và các hệ quản trị cơ sở dữ liệu quan hệ khác, Primary Key (PK) và Foreign Key (FK) là hai khái niệm cốt lõi để thiết lập và duy trì mối quan hệ giữa các bảng, đảm bảo tính toàn vẹn dữ liệu và hiệu quả truy vấn.

- **Khóa chính (Primary Key - PK):** Là một cột hoặc tập hợp các cột có giá trị duy nhất cho mỗi bản ghi trong bảng, dùng để nhận diện duy nhất một hàng dữ liệu. Một bảng chỉ có thể có một khóa chính. Khóa chính đảm bảo tính duy nhất và không được phép chứa giá trị NULL, là yếu tố nền tảng cho việc truy xuất và quản lý dữ liệu hiệu quả. Trong bảng Users, user_id là khóa chính; trong bảng Animes, anime_id là khóa chính.
- **Khóa ngoại (Foreign Key - FK):** Là một cột hoặc tập hợp các cột trong một bảng (gọi là bảng con) tham chiếu đến khóa chính của một bảng khác (gọi là bảng cha). Khóa ngoại thiết lập mối quan hệ giữa hai bảng và đảm bảo tính toàn vẹn tham chiếu (referential integrity), nghĩa là không thể có một bản ghi trong bảng con mà không có bản ghi tương ứng trong bảng cha.

Lợi ích của Foreign Key bao gồm:

- **Đảm bảo toàn vẹn dữ liệu:** Khóa ngoại ngăn chặn việc chèn các bản ghi không hợp lệ hoặc "mồ côi" (orphaned records) vào bảng con. Ví dụ, không thể thêm một Anime vào Watchlist nếu user_id hoặc anime_id đó không tồn tại trong bảng Users hoặc Animes. Điều này giúp duy trì tính nhất quán và chính xác của dữ liệu.
- **Nâng cao chất lượng dữ liệu:** Bằng cách liên kết các bảng, khóa ngoại đảm bảo dữ liệu không chỉ chính xác mà còn liên quan đến các thực thể khác trong cơ sở dữ liệu. Điều này giúp tránh các lỗi logic và đảm bảo rằng mọi dữ liệu đều có ý nghĩa trong ngữ cảnh của ứng dụng.
- **Đơn giản hóa truy xuất dữ liệu:** Khóa ngoại giúp dễ dàng truy vấn dữ liệu liên quan giữa các bảng thông qua các phép nối (JOIN operations), thay vì phải thực hiện nhiều truy vấn riêng lẻ và ghép nối dữ liệu thủ công. Điều

này cải thiện hiệu suất truy vấn và đơn giản hóa logic ứng dụng.

Trong bảng Watchlist, liên kết Users và Animes thông qua user_id và anime_id. Để đảm bảo một người dùng không thể thêm cùng một bộ Anime vào watchlist của họ nhiều lần, cần một ràng buộc duy nhất trên cặp cột (user_id, anime_id). Đây là một trường hợp điển hình của **khóa duy nhất tổng hợp (composite unique key)**. Việc áp dụng ràng buộc

UNIQUE (user_id, anime_id) trực tiếp trong lược đồ cơ sở dữ liệu là một biện pháp bảo vệ mạnh mẽ hơn so với việc chỉ kiểm tra ở tầng ứng dụng. Nó đảm bảo tính toàn vẹn dữ liệu ngay cả khi có lỗi logic ở tầng ứng dụng hoặc truy cập trực tiếp vào cơ sở dữ liệu. Thiết kế cơ sở dữ liệu chi tiết và áp dụng các ràng buộc phù hợp là nền tảng cho một ứng dụng ổn định và đáng tin cậy. Điều này thể hiện sự cẩn trọng và kiến thức sâu rộng về quản lý dữ liệu.

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

1. Người dùng và yêu cầu chức năng

1.1. Đối tượng người dùng và kịch bản sử dụng chi tiết

Đối tượng người dùng chính của hệ thống là những người hâm mộ Anime ở mọi lứa tuổi, những người mong muốn một công cụ hiệu quả để quản lý và khám phá các bộ Anime. Các kịch bản sử dụng chi tiết được phân tích để đảm bảo hệ thống đáp ứng đầy đủ nhu cầu của họ:

- **Người dùng mới:**

- Truy cập trang web lần đầu tiên, được chào đón bởi giao diện thân thiện.
- Chọn tùy chọn "Đăng ký" và điền thông tin cần thiết (tên người dùng, mật khẩu) vào biểu mẫu.
- Nhận thông báo xác nhận đăng ký thành công và được chuyển hướng đến trang đăng nhập.
- Đăng nhập vào hệ thống bằng tài khoản vừa tạo.

- **Người dùng hiện tại:**

- Truy cập trang web và chọn "Đăng nhập".
- Nhập tên người dùng và mật khẩu để truy cập vào tài khoản cá nhân.
- Sau khi đăng nhập, có thể tìm kiếm Anime theo tên, thể loại, hoặc các tiêu chí khác thông qua thanh tìm kiếm hoặc bộ lọc.
- Chọn một bộ Anime từ kết quả tìm kiếm hoặc trang chủ để xem thông tin chi tiết (tóm tắt, điểm số, hình ảnh, thể loại).
- Trên trang chi tiết Anime, có thể thêm bộ Anime đó vào Watchlist cá nhân bằng cách nhấp vào nút "Thêm vào Watchlist".
- Truy cập trang Watchlist để xem danh sách các bộ Anime đã lưu.
- Trên trang Watchlist, có thể xóa một bộ Anime khỏi danh sách.
- Sử dụng chức năng "Random Anime" để khám phá một bộ Anime ngẫu nhiên khi không biết nên xem gì.
- Thực hiện "Xóa toàn bộ Watchlist" nếu muốn bắt đầu lại danh sách của mình.
- Đăng xuất khỏi hệ thống để kết thúc phiên làm việc.

1.2. Yêu cầu chức năng (Functional Requirements)

Các yêu cầu chức năng mô tả các hành vi cụ thể mà hệ thống phải thực hiện để

đáp ứng nhu cầu của người dùng.

- **FR1: Quản lý tài khoản người dùng:**

- FR1.1: Hệ thống phải cho phép người dùng đăng ký tài khoản mới với tên người dùng và mật khẩu duy nhất.
- FR1.2: Hệ thống phải cho phép người dùng đăng nhập vào tài khoản đã đăng ký.
- FR1.3: Hệ thống phải xác thực thông tin đăng nhập và quản lý phiên người dùng một cách an toàn.
- FR1.4: Hệ thống phải cho phép người dùng đăng xuất khỏi tài khoản của mình.

- **FR2: Quản lý Watchlist cá nhân:**

- FR2.1: Người dùng đã đăng nhập phải có khả năng thêm một bộ Anime vào Watchlist của mình.
- FR2.2: Người dùng phải có khả năng xóa một bộ Anime khỏi Watchlist.
- FR2.3: Hệ thống phải hiển thị danh sách các bộ Anime trong Watchlist của người dùng với thông tin cơ bản.
- FR2.4: Hệ thống phải cho phép người dùng xóa toàn bộ Watchlist của mình.

- **FR3: Khám phá Anime:**

- FR3.1: Hệ thống phải cho phép người dùng tìm kiếm Anime theo tên hoặc ID từ cơ sở dữ liệu.
- FR3.2: Hệ thống phải cho phép người dùng lọc Anime theo thể loại, điểm số, và trạng thái phát sóng.
- FR3.3: Hệ thống phải có chức năng hiển thị một bộ Anime ngẫu nhiên từ cơ sở dữ liệu.
- FR3.4: Hệ thống phải hiển thị trang chi tiết cho mỗi bộ Anime, bao gồm tóm tắt, điểm số, hình ảnh, thể loại, và các thông tin liên quan khác.

1.3. Yêu cầu phi chức năng (Non-functional Requirements: hiệu năng, bảo mật, khả năng mở rộng, khả năng sử dụng)

Các yêu cầu phi chức năng mô tả chất lượng của hệ thống và cách nó hoạt động, chứ không phải chức năng cụ thể.

- **NFR1: Hiệu năng (Performance):**

- NFR1.1: Thời gian phản hồi cho các thao tác CRUD (Create, Read, Update, Delete) trên Watchlist không quá 2 giây.
- NFR1.2: Thời gian tải trang hiển thị danh sách Anime (bao gồm tìm kiếm và lọc) không quá 3 giây.

- NFR1.3: Hệ thống phải có khả năng xử lý đồng thời ít nhất 50 người dùng mà không giảm đáng kể hiệu suất.
- **NFR2: Bảo mật (Security):**
 - NFR2.1: Mật khẩu người dùng phải được băm (hash) và thêm muối (salting) an toàn bằng thuật toán Bcrypt trước khi lưu trữ trong cơ sở dữ liệu.
 - NFR2.2: Giao tiếp giữa client và server phải được mã hóa sử dụng HTTPS để bảo vệ dữ liệu truyền tải.
 - NFR2.3: Hệ thống phải sử dụng JSON Web Tokens (JWT) để xác thực và ủy quyền, đảm bảo tính không trạng thái và bảo mật phiên.
 - NFR2.4: Dữ liệu nhạy cảm của người dùng, đặc biệt là mật khẩu, không được lưu trữ dưới dạng plaintext trong bất kỳ hệ thống nào.
 - NFR2.5: Hệ thống phải có khả năng chống lại các cuộc tấn công phổ biến như SQL Injection và Cross-Site Scripting (XSS) thông qua việc xác thực đầu vào nghiêm ngặt và sử dụng parameterized queries.
- **NFR3: Khả năng mở rộng (Scalability):**
 - NFR3.1: Kiến trúc backend phải hỗ trợ mở rộng theo chiều ngang để đáp ứng số lượng người dùng tăng lên trong tương lai.
 - NFR3.2: Cơ sở dữ liệu phải được thiết kế để dễ dàng mở rộng và tối ưu cho việc truy vấn lớn, đảm bảo hiệu suất ổn định khi dữ liệu phát triển.
- **NFR4: Khả năng sử dụng (Usability):**
 - NFR4.1: Giao diện người dùng phải trực quan, dễ hiểu và dễ điều hướng, giúp người dùng mới nhanh chóng làm quen và sử dụng hiệu quả.
 - NFR4.2: Hệ thống phải cung cấp phản hồi rõ ràng và kịp thời cho các hành động của người dùng (ví dụ: thông báo thêm/xóa thành công, lỗi đăng nhập).
 - NFR4.3: Giao diện phải responsive, hiển thị tốt trên các kích thước màn hình khác nhau (desktop, tablet, mobile) để đảm bảo trải nghiệm nhất quán trên mọi thiết bị.

2. Các tính năng chính của hệ thống

2.1. Đăng ký và đăng nhập

Quy trình xác thực người dùng trong dự án này được thiết kế theo cách đơn giản và dễ triển khai cho ứng dụng cá nhân, sử dụng fetch API và localStorage của trình duyệt để lưu trạng thái đăng nhập.

Đăng ký

Người dùng mới có thể tạo tài khoản thông qua một form đăng ký trên giao diện web. Quy trình cụ thể:

- Khi người dùng nhập tên người dùng (username), mật khẩu (password) và xác nhận mật khẩu (confirmPassword), sự kiện submit của form sẽ bị chặn (e.preventDefault()), đảm bảo không reload trang.
- Ứng dụng frontend kiểm tra xem mật khẩu và xác nhận mật khẩu có khớp không. Nếu không, hiển thị thông báo lỗi ngay trên trình duyệt.
- Nếu hợp lệ, một yêu cầu HTTP POST được gửi đến API /createUser trên server.
- Backend (Node.js/Express) nhận dữ liệu JSON, kiểm tra tính duy nhất của username trong CSDL MySQL và lưu mật khẩu (ở server trong ví dụ gốc chưa hash, minh họa cực kỳ đơn giản).
- Server phản hồi về client với thông báo JSON, ví dụ: { message: 'Đăng ký tài khoản thành công!' }.
- Frontend hiển thị thông báo thành công và chuyển hướng sang trang đăng nhập.

```
frontend > register > js script.js > ...
1  document.addEventListener('DOMContentLoaded', function() {
2    const registerForm = document.querySelector('#registerForm');
3    const usernameInput = document.querySelector('#username');
4    const passwordInput = document.querySelector('#password');
5    const confirmPasswordInput = document.querySelector('#confirmPassword');
6
7    registerForm.addEventListener('submit', function(e) {
8      e.preventDefault(); // Ngăn form submit mặc định
9
10   // Kiểm tra mật khẩu khớp nhau
11   if (passwordInput.value !== confirmPasswordInput.value) {
12     alert('Mật khẩu xác nhận không khớp!');
13     return;
14   }
15
16   fetch('http://127.0.0.1:3000/createUser', {
17     method: 'POST',
18     headers: {
19       'Content-Type': 'application/json'
20     },
21     body: JSON.stringify({ username: usernameInput.value, password: passwordInput.value })
22   })
23     .then(response => response.json())
24     .then(data => {
25       alert(data.message);
26       window.location.href = '../login/index.html';
27     })
28     .catch(err => {
29       alert('Lỗi: ' + 'Tài khoản đã tồn tại!');
30     });
31   });
32});
```

Script trang đăng ký

Đăng nhập

Quy trình đăng nhập sử dụng phương thức GET để kiểm tra thông tin tài khoản:

- Người dùng nhập username và password trong form đăng nhập.
- Khi form submit, sự kiện bị chặn lại để xử lý bằng JavaScript.
- Frontend gọi API.
- Server tìm trong CSDL thông tin người dùng theo username và trả về JSON gồm username và password (password plaintext trong ví dụ).
- Ở client, password do người dùng nhập được so sánh trực tiếp với password trả về từ server.
- Nếu khớp, thông tin username và trạng thái đăng nhập (loggedIn=true) được lưu vào localStorage:
- Sau đó điều hướng người dùng tới trang chính (hoặc loading).

```
frontend > login > js script.js > ...
1  document.addEventListener('DOMContentLoaded', function () {
2      const loginForm = document.querySelector('#loginForm');
3      const usernameInput = document.querySelector('#username');
4      const passwordInput = document.querySelector('#password');
5
6      loginForm.addEventListener('submit', function (e) {
7          e.preventDefault();
8
9          fetch('http://localhost:3000/getUser?username=' + encodeURIComponent(usernameInput.value))
10         .then(response => {
11             if (!response.ok) throw new Error('Tài khoản không tồn tại!');
12             return response.json();
13         })
14         .then(data => {
15             if (data.password === passwordInput.value) {
16                 localStorage.setItem('username', usernameInput.value);
17                 localStorage.setItem('loggedIn', 'true');
18                 window.location.href = '../loading/index.html';
19             } else {
20                 alert('Sai mật khẩu!');
21             }
22         })
23         .catch(err => {
24             alert('Lỗi: ' + err.message);
25         });
26     });
27 });


```

Script trang đăng nhập

Lưu trạng thái phiên (Session)

Hệ thống không sử dụng session phía server hay JWT. Thay vào đó, trạng thái đăng nhập được lưu trữ đơn giản ở client-side:

- localStorage của trình duyệt lưu giá trị username và loggedIn.

- Các trang cần bảo vệ có thể kiểm tra localStorage.

Ưu điểm:

- Rất dễ triển khai cho dự án nhỏ.
- Không cần server lưu session.

Hạn chế:

- Không bảo mật (password trả về nguyên văn từ server, localStorage có thể bị xem từ trình duyệt).
- Không chống được giả mạo request.

2.2. Thêm/bớt Anime vào Watchlist

Tính năng thêm/bớt Anime vào Watchlist là một trong những chức năng cốt lõi và được sử dụng thường xuyên nhất của ứng dụng. Để đảm bảo trải nghiệm người dùng mượt mà và trực quan, mỗi card Anime trên giao diện sẽ được trang bị một nút "bookmark" hoặc biểu tượng tương tự.

Khi người dùng nhấp vào nút "Thêm vào Watchlist" trên một card Anime, một yêu cầu sẽ được gửi đến backend. Backend sẽ xử lý yêu cầu này bằng cách thêm một bản ghi mới vào bảng Watchlist trong cơ sở dữ liệu, liên kết user_id của người dùng hiện tại với anime_id của bộ Anime được chọn. Để ngăn chặn việc một bộ Anime bị thêm vào Watchlist nhiều lần bởi cùng một người dùng, bảng Watchlist sẽ có một ràng buộc duy nhất tổng hợp (composite unique key) trên cặp cột (user_id, anime_id). Điều này đảm bảo tính toàn vẹn dữ liệu ở cấp độ cơ sở dữ liệu, ngay cả khi có lỗi ở tầng ứng dụng.

Ngược lại, khi người dùng muốn xóa một bộ Anime khỏi Watchlist của mình, họ sẽ nhấp vào nút "Xóa" hoặc biểu tượng tương ứng trên card Anime đó (thường là trên trang Watchlist). Một yêu cầu DELETE sẽ được gửi đến backend, và backend sẽ xóa bản ghi tương ứng khỏi bảng Watchlist. Hệ thống sẽ cung cấp phản hồi trực quan cho người dùng, ví dụ như một thông báo xác nhận thành công hoặc lỗi.

2.3. Xem danh sách Anime đã lưu

Trang Watchlist là trung tâm quản lý danh sách Anime cá nhân của người dùng. Khi người dùng truy cập trang này, hệ thống sẽ gửi một yêu cầu đến backend để lấy tất cả các bộ Anime đã được lưu trong Watchlist của người dùng đó.

Backend sẽ truy vấn cơ sở dữ liệu, kết hợp thông tin từ bảng Watchlist với bảng Animes để lấy các chi tiết cần thiết về mỗi bộ Anime.

Giao diện trang Watchlist sẽ hiển thị các card Anime với thông tin cơ bản như tiêu đề, hình ảnh, thể loại, và có thể là điểm số hoặc trạng thái xem. Các card này được thiết kế để dễ nhìn và dễ thao tác, cho phép người dùng nhanh chóng nắm bắt thông tin và thực hiện các hành động như cập nhật trạng thái hoặc xóa Anime khỏi danh sách. Việc hiển thị danh sách một cách rõ ràng và có tổ chức là yếu tố quan trọng để nâng cao khả năng sử dụng của ứng dụng.

2.4. Xóa toàn bộ Watchlist

Để cung cấp sự linh hoạt tối đa cho người dùng, hệ thống sẽ có một nút "Clear" hoặc "Xóa toàn bộ Watchlist" trên trang Watchlist. Khi người dùng nhấp vào nút này và xác nhận hành động, một yêu cầu sẽ được gửi đến backend để xóa tất cả các bản ghi liên quan đến user_id của người dùng đó trong bảng Watchlist.

Chức năng này cho phép người dùng dễ dàng làm mới danh sách của mình mà không cần phải xóa từng mục một. Tuy nhiên, để tránh việc xóa nhầm dữ liệu, hệ thống sẽ yêu cầu người dùng xác nhận lại hành động này trước khi thực hiện.

2.5. Khám phá Anime

Tính năng khám phá Anime là một phần quan trọng giúp người dùng tìm kiếm và tìm thấy các bộ Anime mới.

- **Random Anime:** Hệ thống sẽ cung cấp một nút hoặc chức năng "Random Anime". Khi được kích hoạt, một yêu cầu sẽ được gửi đến Jikan API để lấy ngẫu nhiên một bộ Anime. Thông tin về bộ Anime đó sẽ được hiển thị trên giao diện, cho phép người dùng khám phá những bộ phim mà họ có thể chưa từng biết đến.
- **Tìm kiếm:** Người dùng có thể sử dụng thanh tìm kiếm để tìm Anime theo tên hoặc ID. Yêu cầu tìm kiếm sẽ được gửi đến backend, sau đó backend sẽ sử dụng Jikan API để truy vấn dữ liệu. Kết quả tìm kiếm sẽ được hiển thị dưới dạng danh sách các card Anime trên giao diện.
- **Lọc:** Để thu hẹp kết quả tìm kiếm, người dùng có thể lọc Anime theo các tiêu chí như thể loại, điểm số, hoặc trạng thái phát sóng. Các tùy chọn lọc này sẽ được gửi kèm trong yêu cầu đến backend, và backend sẽ chuyển tiếp các tham số lọc này đến Jikan API.
- **Chi tiết Anime:** Khi người dùng nhấp vào một card Anime từ trang chủ,

kết quả tìm kiếm, hoặc trang Watchlist, hệ thống sẽ chuyển hướng đến một trang chi tiết Anime. Trang này sẽ hiển thị đầy đủ thông tin về bộ Anime đó, bao gồm tóm tắt cốt truyện (synopsis), điểm số đánh giá, hình ảnh lớn, danh sách thẻ loại, và các thông tin bổ sung khác được lấy từ Jikan API.

3. Thiết kế cơ sở dữ liệu

Thiết kế cơ sở dữ liệu là một giai đoạn cực kỳ quan trọng, quyết định sự thành công của ứng dụng về khả năng mở rộng, hiệu suất, bảo mật và toàn vẹn dữ liệu. Một thiết kế tốt đảm bảo dữ liệu được lưu trữ một cách có tổ chức, hiệu quả và an toàn.

3.1. Chi tiết các bảng

Cơ sở dữ liệu MySQL cho dự án này sẽ bao gồm hai bảng chính để lưu trữ thông tin người dùng, watchlist.

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	username	varchar(50)	NO	UNI	NULL	
	password	varchar(255)	NO		NULL	

Bảng users

	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	username	varchar(255)	NO		NULL	
	mal_id	int	NO		NULL	
	title	varchar(255)	NO		NULL	
	image_url	varchar(500)	YES		NULL	
	score	float	YES		NULL	

Bảng watchlist

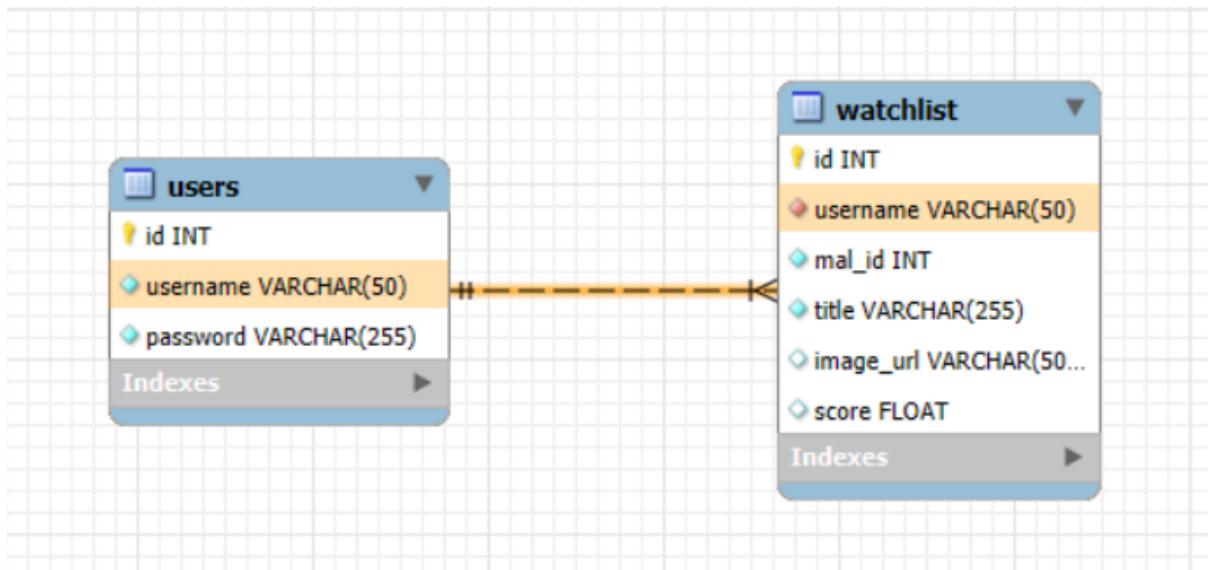
Bảng Users lưu trữ thông tin cơ bản của người dùng. Trường **username** đóng vai trò khóa chính và là giá trị duy nhất, đảm bảo mỗi người dùng chỉ có một tài khoản. Mật khẩu được lưu ở cột **password** để tăng cường bảo mật, giúp hạn chế rủi ro nếu dữ liệu bị rò rỉ.

Bảng Watchlist lưu danh sách Anime mà người dùng muốn theo dõi. Mỗi bản ghi trong Watchlist chứa **username** (khóa ngoại tham chiếu Users) và các thông

tin cơ bản về Anime như **mal_id**, **title**, **image_url**, và **score**. Thiết kế này gộp thông tin Anime trực tiếp vào bảng Watchlist thay vì tách riêng bảng Animes, giúp giảm phức tạp trong mối quan hệ và dễ triển khai trong dự án cá nhân nhỏ.

Ràng buộc **UNIQUE(username, mal_id)** được sử dụng để đảm bảo rằng một người dùng không thể thêm cùng một Anime nhiều lần vào Watchlist của họ, duy trì tính nhất quán và tránh dữ liệu dư thừa.

3.2. UML



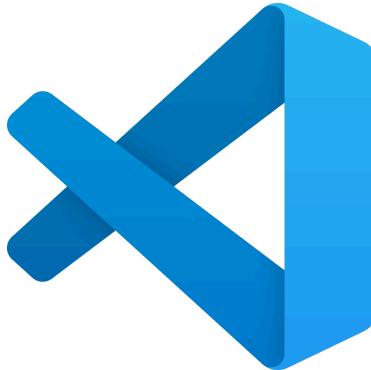
CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH

1. Công cụ phát triển

Quá trình xây dựng chương trình được thực hiện với sự hỗ trợ của các công cụ phát triển chuyên nghiệp, giúp tối ưu hóa quy trình làm việc và tăng cường hiệu quả.

1.1. Visual Studio Code

Visual Studio Code (VS Code) là môi trường phát triển tích hợp (IDE) chính được sử dụng trong dự án này. VS Code là một trình soạn thảo mã nguồn nhẹ nhưng mạnh mẽ, hỗ trợ đa ngôn ngữ lập trình, bao gồm JavaScript, HTML, CSS, và SQL. Nó cung cấp một loạt các tính năng hữu ích như IntelliSense (tự động hoàn thành mã), gỡ lỗi tích hợp, kiểm soát phiên bản Git, và một hệ sinh thái plugin phong phú. Các plugin này giúp tăng cường năng suất, hỗ trợ định dạng mã, kiểm tra lỗi cú pháp, và tích hợp trực tiếp với các công cụ khác, làm cho quá trình phát triển trở nên nhanh chóng và thuận tiện.



Visual Studio Code

1.2. MySQL Workbench

MySQL Workbench là một công cụ quản lý cơ sở dữ liệu đồ họa được sử dụng để thiết kế, phát triển và quản lý cơ sở dữ liệu MySQL. Trong dự án này, MySQL Workbench được sử dụng để:

- **Thiết kế ERD (Entity-Relationship Diagram):** Cho phép trực quan hóa cấu trúc cơ sở dữ liệu, định nghĩa các bảng, cột, khóa chính, khóa ngoại và mối quan hệ giữa chúng một cách dễ dàng.
- **Quản lý cơ sở dữ liệu:** Cung cấp giao diện để tạo, sửa đổi, xóa bảng, chèn

dữ liệu mẫu, và thực hiện các truy vấn SQL trực tiếp. Điều này giúp kiểm tra và gỡ lỗi cơ sở dữ liệu trong quá trình phát triển backend.



MySQL Workbench

2. Xây dựng Backend

Backend được xây dựng bằng Node.js và framework Express.js, chịu trách nhiệm xử lý logic nghiệp vụ, quản lý dữ liệu và cung cấp API cho frontend.

2.1. Thiết kế RESTful API

Các API được thiết kế theo nguyên tắc RESTful, đảm bảo tính không trạng thái, giao diện đồng nhất và khả năng mở rộng. Các route chính được định nghĩa để phục vụ các yêu cầu từ frontend.

2.2. Kết nối cơ sở dữ liệu

Để tương tác với cơ sở dữ liệu MySQL, dự án sử dụng thư viện mysql2 cho Node.js. mysql2 cung cấp một API Promise, hoạt động rất tốt với cú pháp `async/await` của ES7, giúp việc viết code bất đồng bộ trở nên dễ đọc và dễ bảo trì hơn.

Ví dụ về cách tạo kết nối và thực hiện truy vấn:

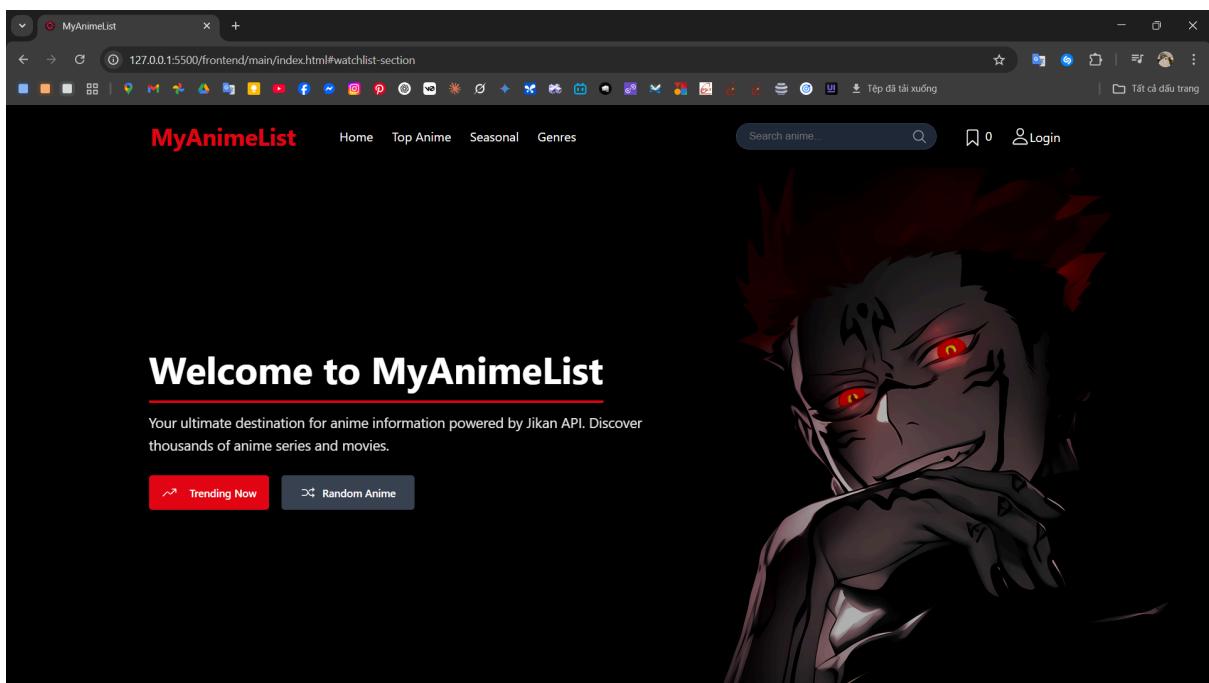
```
backend > js db.js > ...
1 import mysql from 'mysql2';
2
3 const db = mysql.createPool({
4   host: '127.0.0.1',
5   user: 'root',
6   password: '1234',
7   database: 'anime_db'
8 }).promise();
9
10 export const getUser = async (username) => {
11   const [rows] = await db.query('SELECT * FROM users WHERE username = ?', [username]);
12   return rows[0];
13 };
```

Việc sử dụng connection pool giúp giảm thời gian kết nối đến server MySQL bằng cách tái sử dụng các kết nối đã có, thay vì đóng và mở lại liên tục, từ đó cải thiện độ trễ của các truy vấn. Ngoài ra, có thể sử dụng một ORM (Object-Relational Mapping) như Sequelize để trừu tượng hóa các thao tác cơ sở dữ liệu, giúp code sạch hơn và dễ quản lý hơn, mặc dù mysql2 trực tiếp cũng đủ mạnh cho quy mô dự án này.

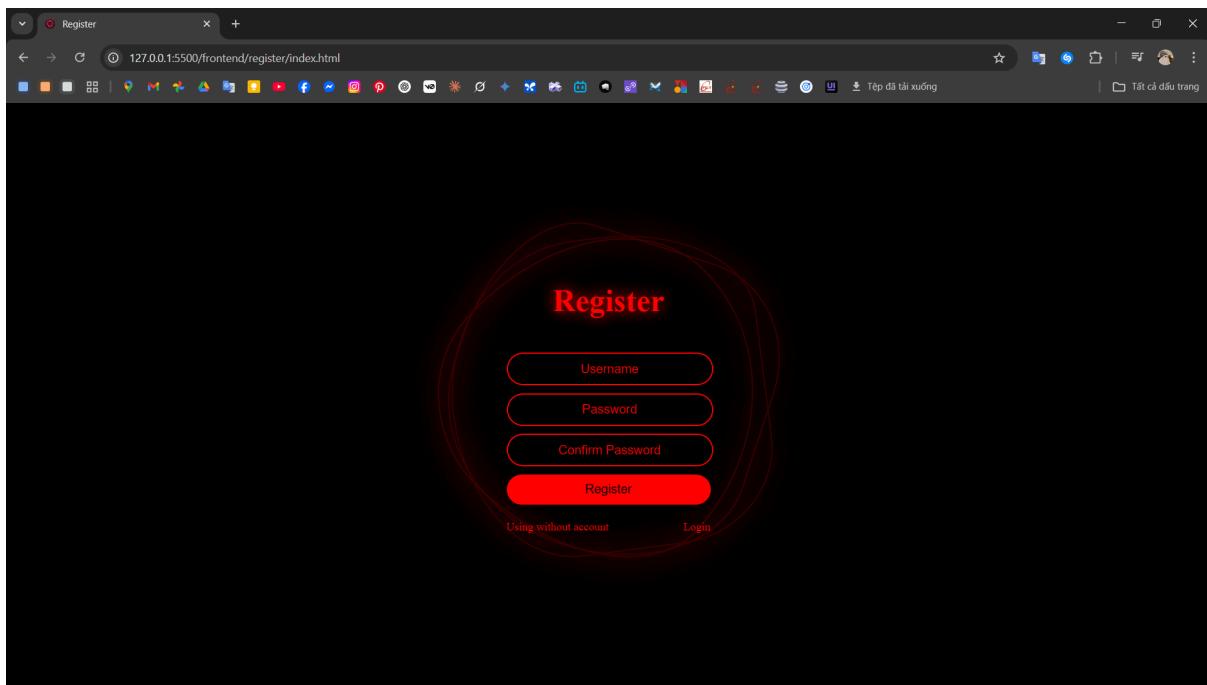
3. Xây dựng Frontend

Frontend được xây dựng để cung cấp một giao diện người dùng trực quan và tương tác, sử dụng HTML cho cấu trúc, CSS cho kiểu dáng và JavaScript cho logic tương tác.

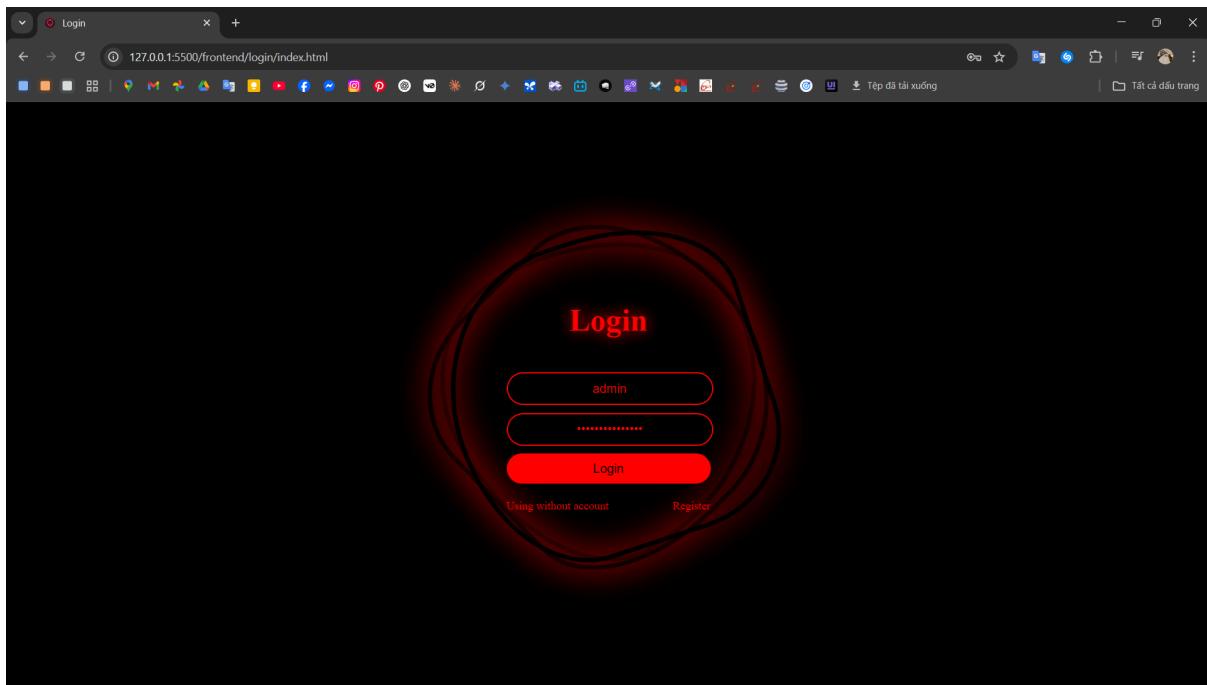
3.1. Thiết kế giao diện



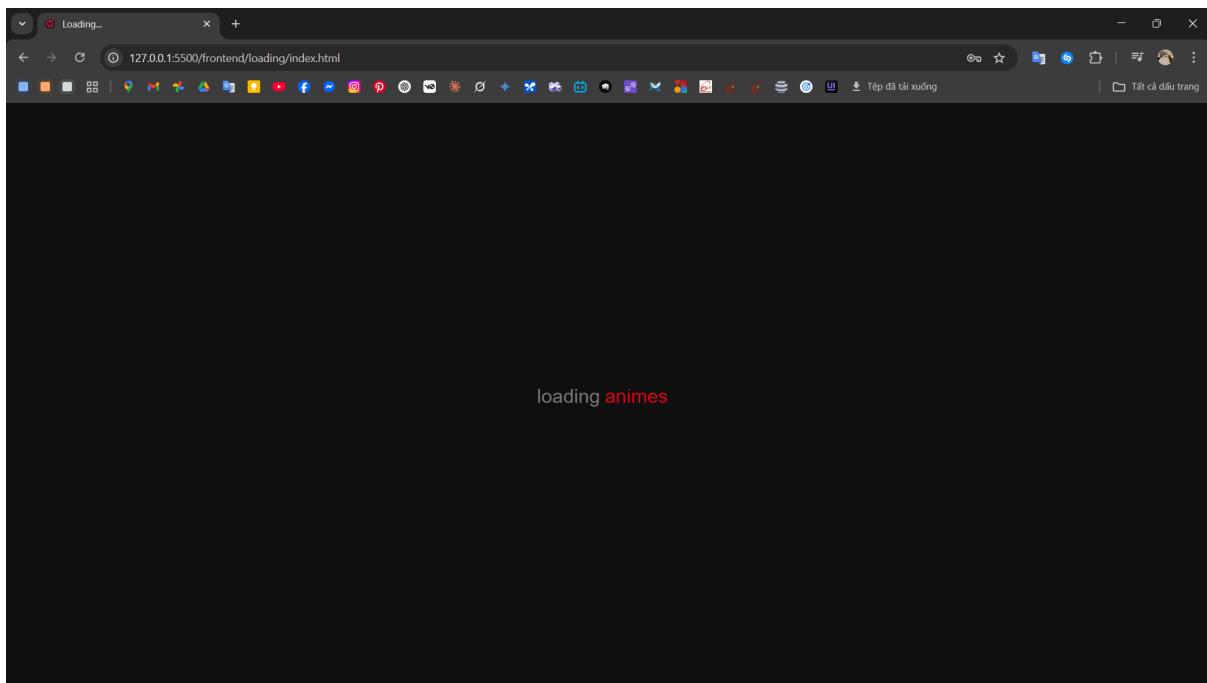
Hình 3.1.1 Giao diện trang chủ.



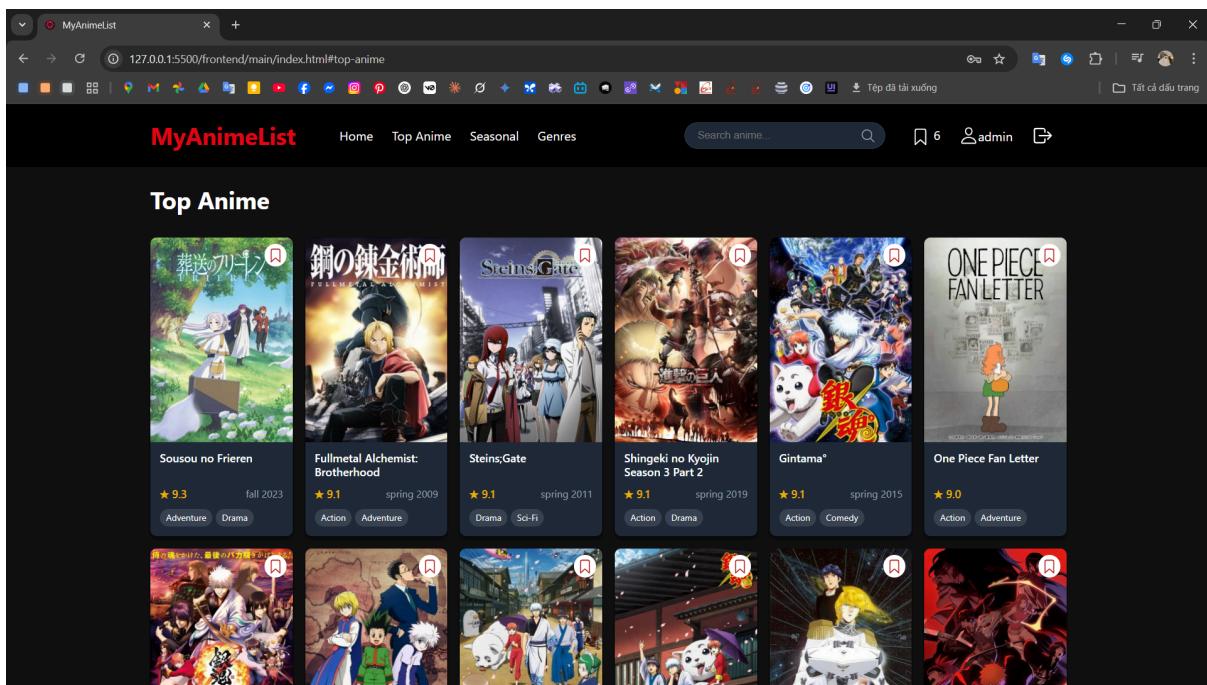
Hình 3.1.2 Giao diện đăng ký tài khoản.



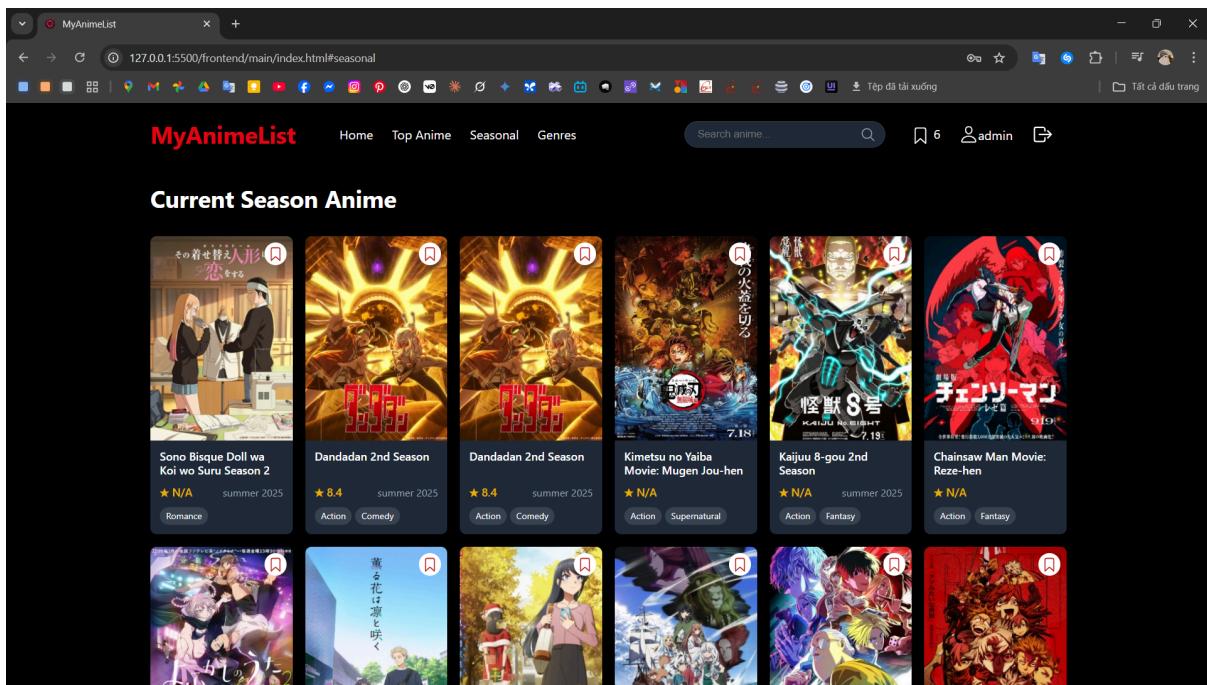
Hình 3.1.3 Giao diện đăng nhập.



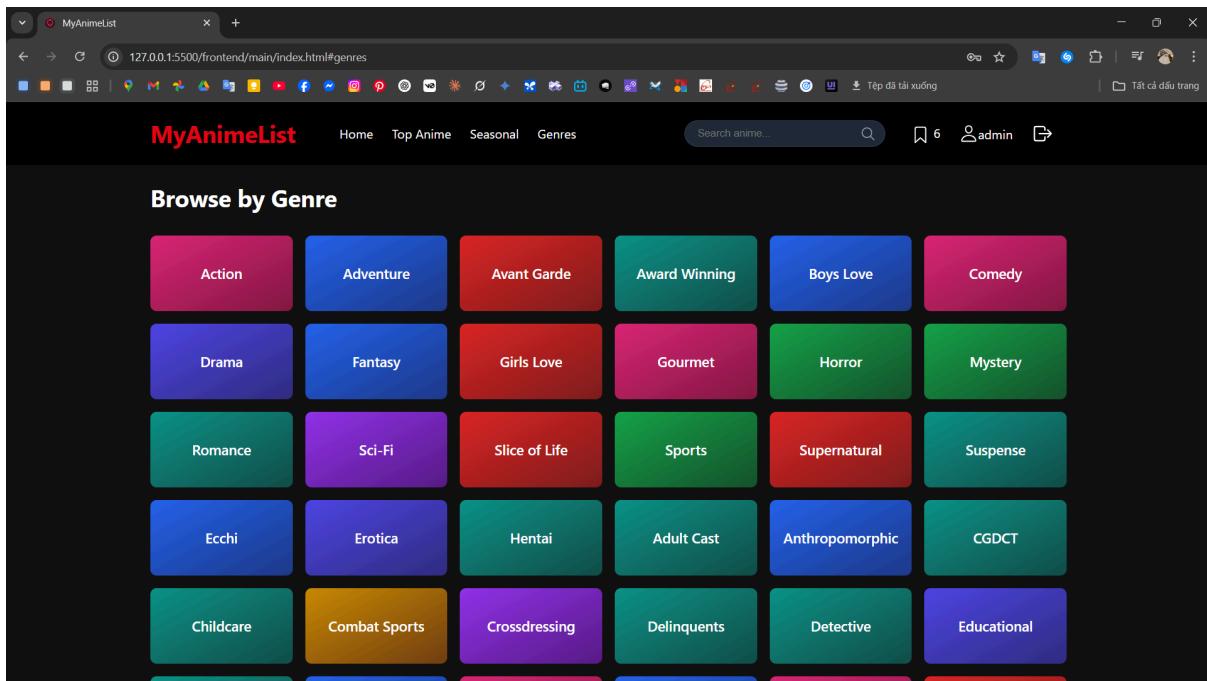
Hình 3.1.4 Màn hình loading.



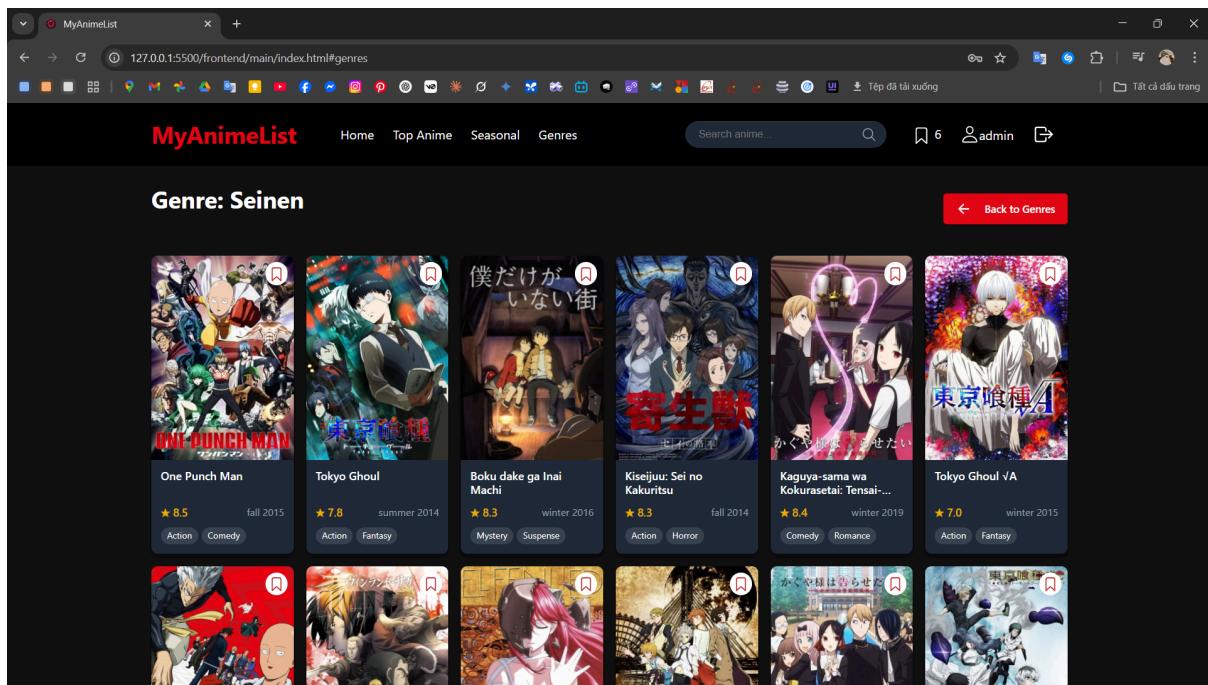
Hình 3.1.5 Top Anime.



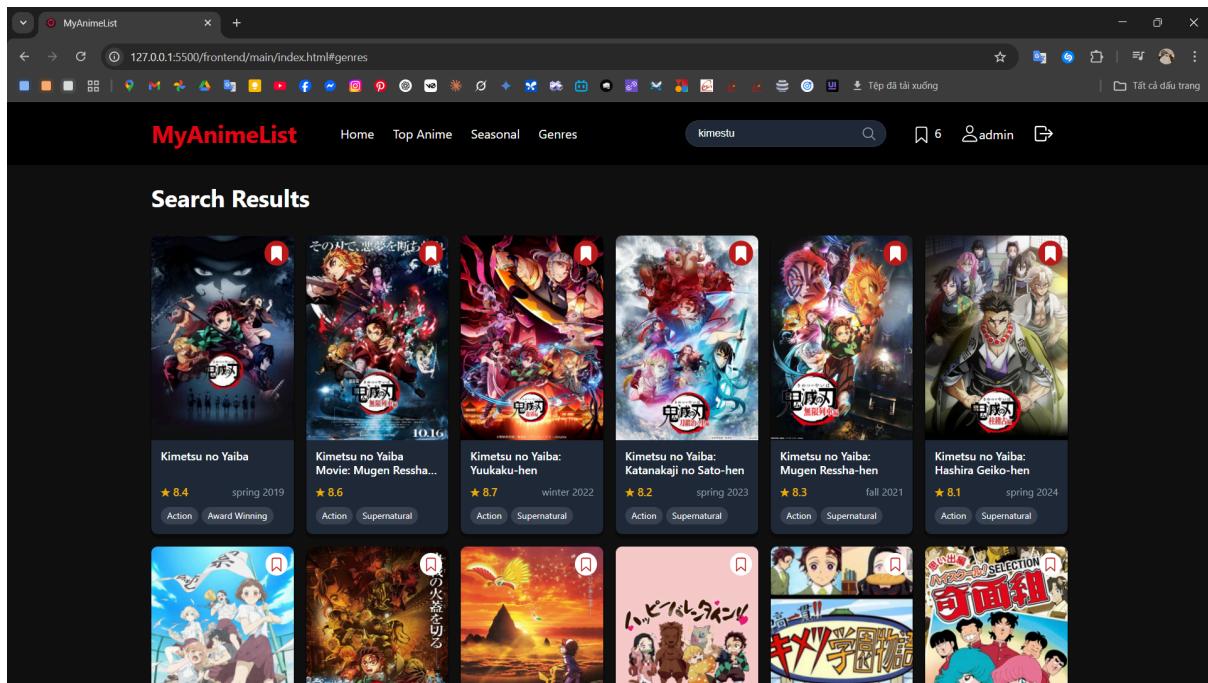
Hình 3.1.6 Current Season Anime.



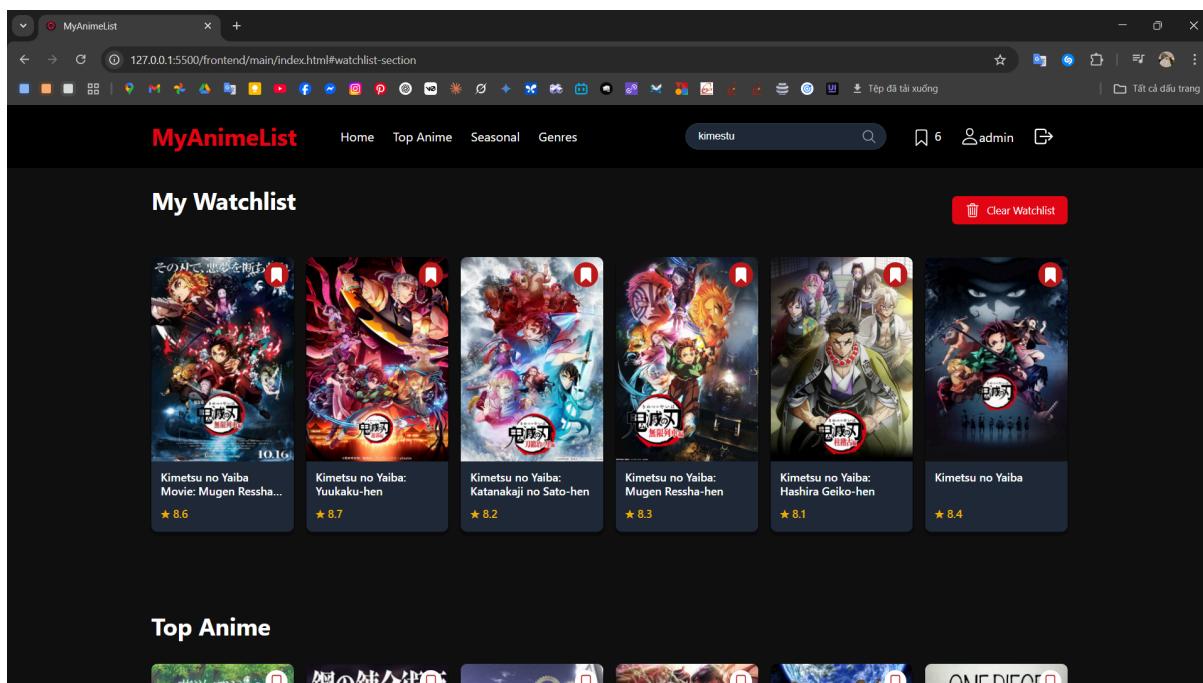
Hình 3.1.7 Giao diện chọn thể loại.



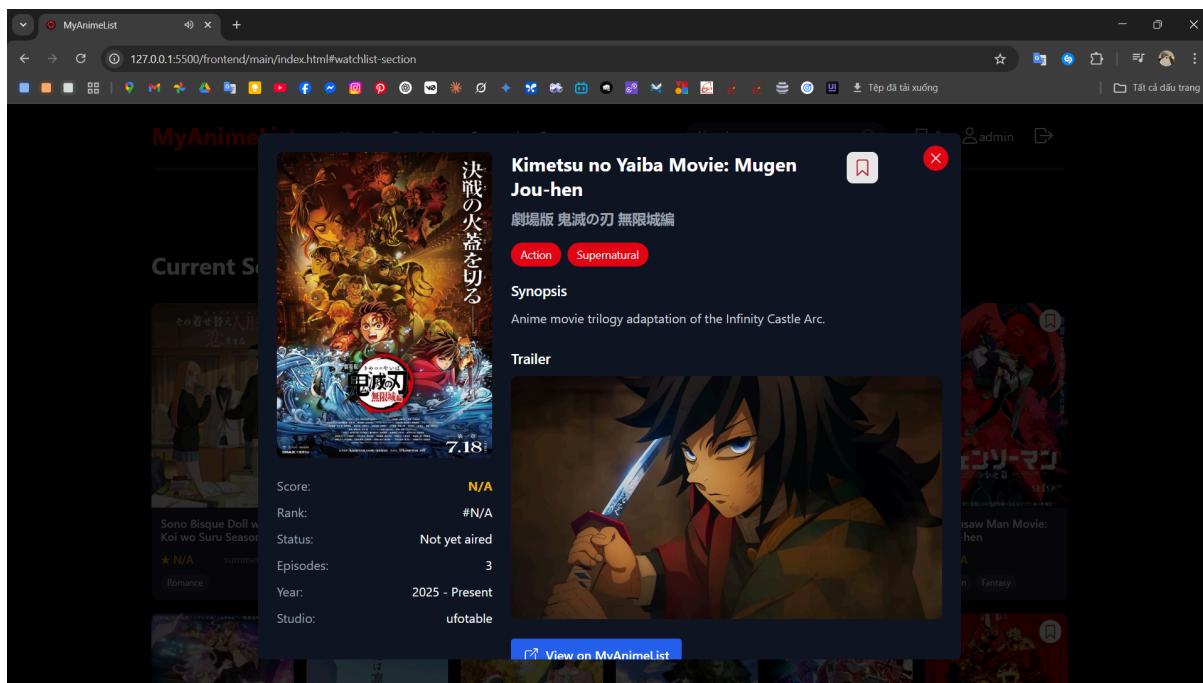
Hình 3.1.8 Chức năng chọn thể loại.



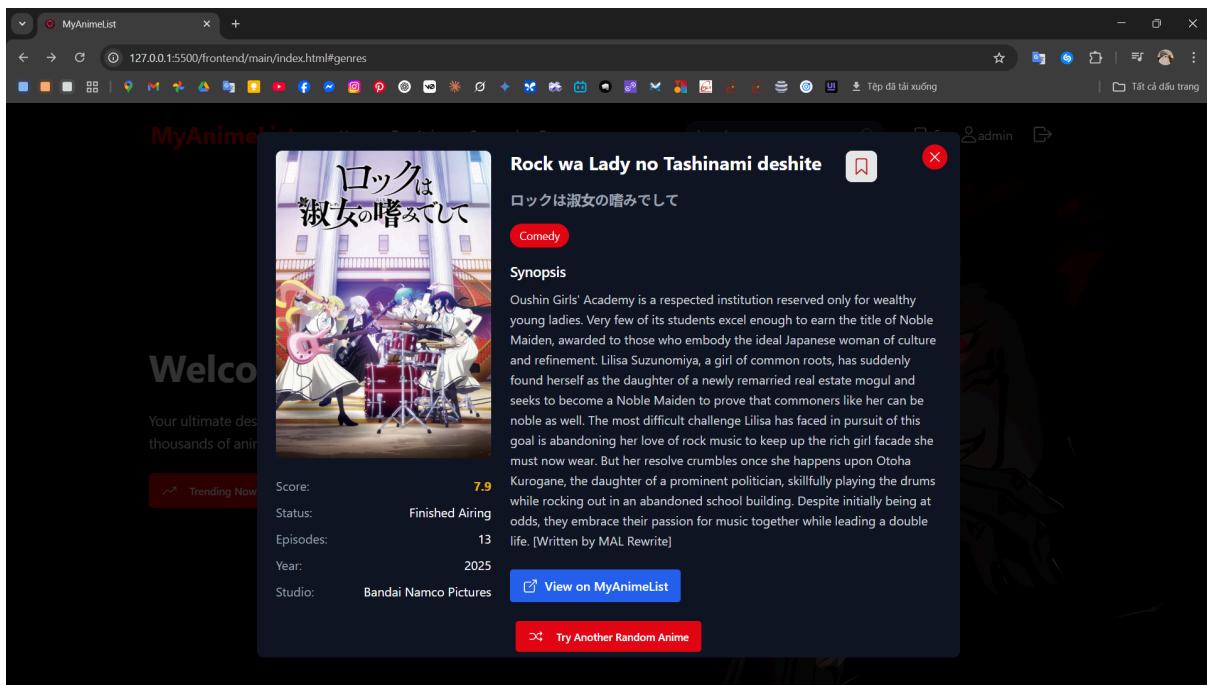
Hình 3.1.9 Chức năng tìm kiếm.



Hình 3.1.10 Chức năng watchlist.



Hình 3.1.11 Thủ chi tiết Anime.



Hình 3.1.12 Chức năng Random Anime.

Nhận xét:

Giao diện website được thiết kế lấy cảm hứng từ Netflix, với mục tiêu tạo trải nghiệm người dùng trực quan, hiện đại và dễ sử dụng. Màu nền tối (dark theme) được sử dụng làm màu chủ đạo giúp làm nổi bật hình ảnh Anime và tạo cảm giác điện ảnh, đồng thời giảm mỏi mắt khi sử dụng lâu. Các thành phần giao diện được bố trí dạng lưới (grid), với poster phim có kích thước đồng đều, mang lại cảm giác gọn gàng và dễ duyệt qua danh sách.

Thanh điều hướng được thiết kế tối giản nhưng trực quan, cho phép người dùng truy cập nhanh các mục quan trọng như Trang chủ, Tìm kiếm, và Watchlist cá nhân. Các nút thêm/bớt Anime khỏi Watchlist được hiển thị trực tiếp trên poster, tương tự trải nghiệm thêm vào My List trên Netflix, giúp thao tác nhanh chóng và không cần rời khỏi trang đang xem.

Màu nhấn (accent color) được sử dụng hợp lý để thu hút chú ý đến các nút quan trọng mà không gây rối mắt. Font chữ hiện đại, dễ đọc, cùng bố cục responsive hỗ trợ tốt trên cả máy tính và thiết bị di động.

Tổng thể, giao diện mang lại cảm giác quen thuộc và chuyên nghiệp cho người dùng – lấy cảm hứng rõ rệt từ Netflix nhưng được tinh chỉnh cho phù hợp với đặc thù quản lý danh sách Anime cá nhân.

3.2. Tích hợp API

Frontend giao tiếp với backend thông qua các API RESTful đã thiết kế. Việc tích hợp API được thực hiện bằng cách sử dụng fetch hoặc axios để gửi các yêu cầu HTTP (GET, POST, DELETE) đến các endpoint của backend.

- **Gửi yêu cầu:** Khi người dùng thực hiện một hành động (ví dụ: đăng nhập, thêm Anime vào Watchlist, tìm kiếm), JavaScript sẽ tạo một yêu cầu HTTP tương ứng. Ví dụ, để thêm Anime vào Watchlist:

```
await fetch("http://localhost:3000/watchlist", {
  method: "POST",
  headers: { "Content-Type": "application/json" },
  body: JSON.stringify({
    username,
    mal_id: anime.mal_id,
    title: anime.title,
    image_url: anime.images?.jpg?.image_url || anime.image_url,
    score: anime.score
  })
});
watchlist.push({
  mal_id: anime.mal_id,
  title: anime.title,
  image_url: anime.images?.jpg?.image_url || anime.image_url,
  score: anime.score
});
showNotification(`Đã thêm "${anime.title}" vào danh sách theo dõi`);
```

- **Xử lý kết quả và render DOM:** Sau khi nhận được phản hồi từ backend, JavaScript sẽ xử lý kết quả (ví dụ: phân tích cú pháp JSON) và cập nhật giao diện người dùng bằng cách thao tác DOM. Ví dụ, sau khi thêm Anime thành công, giao diện Watchlist có thể được cập nhật để hiển thị mục mới. Đối với các yêu cầu tìm kiếm, dữ liệu Anime nhận được sẽ được dùng để tạo động các card Anime và hiển thị trên trang. Các lỗi từ backend cũng sẽ được xử lý và hiển thị thông báo phù hợp cho người dùng.

4. Kiểm thử chức năng

Kiểm thử là một giai đoạn quan trọng trong quy trình phát triển phần mềm để đảm bảo chất lượng, độ tin cậy và hiệu suất của ứng dụng. Trong dự án này, các phương pháp kiểm thử khác nhau được áp dụng.

- **Kiểm thử UI với các kịch bản người dùng:** Sau khi frontend và backend được tích hợp, hệ thống được kiểm thử toàn diện từ góc độ người dùng. Các

kịch bản kiểm thử bao gồm:

- **Đăng ký và Đăng nhập:** Kiểm tra quy trình đăng ký tài khoản mới, đăng nhập thành công và xử lý các trường hợp lỗi (tên người dùng đã tồn tại, mật khẩu sai).
 - **Quản lý Watchlist:** Kiểm thử chức năng thêm, xóa Anime vào/khỏi Watchlist. Đảm bảo rằng một bộ Anime không thể được thêm hai lần vào Watchlist của cùng một người dùng. Khả năng xóa toàn bộ Watchlist.
 - **Khám phá Anime:** Kiểm thử chức năng tìm kiếm (theo tên, ID), lọc theo thể loại và hiển thị Anime ngẫu nhiên. Xác minh rằng các trang chi tiết Anime hiển thị đúng thông tin.
 - **Kiểm thử giao diện:** Đảm bảo giao diện hiển thị đúng trên các kích thước màn hình khác nhau (responsive design) và các yếu tố tương tác hoạt động mượt mà.
-
- **Kiểm thử API với Postman:** Trước khi tích hợp frontend, các API của backend được kiểm thử độc lập bằng Postman. Postman là một công cụ phổ biến cho phép gửi các yêu cầu HTTP (GET, POST, PUT, DELETE) đến các endpoint API và xem phản hồi. Điều này giúp xác minh rằng các API hoạt động đúng như mong đợi, xử lý dữ liệu chính xác và trả về các mã trạng thái HTTP phù hợp. Postman cũng hỗ trợ quản lý môi trường và biến, giúp dễ dàng kiểm thử các kịch bản khác nhau (ví dụ: đăng nhập với các tài khoản khác nhau).



Logo Postman

Các loại kiểm thử này đảm bảo rằng ứng dụng web hoạt động bình thường và chính xác, từ các liên kết đến các biểu mẫu và kết nối cơ sở dữ liệu.

CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết quả đạt được

Dự án "Website MyAnimeList" đã đạt được các mục tiêu chính đề ra, tạo ra một ứng dụng web cơ bản nhưng đầy đủ chức năng, thể hiện khả năng phát triển full-stack.

- Hoàn thiện hệ thống CRUD Watchlist:** Dự án đã thành công trong việc triển khai các chức năng cốt lõi cho phép người dùng đăng ký, đăng nhập, và quản lý danh sách Anime cá nhân của họ. Các thao tác thêm, xóa, cập nhật trạng thái Anime trong Watchlist đã được xây dựng và kiểm thử thành công, đảm bảo tính toàn vẹn dữ liệu thông qua các ràng buộc cơ sở dữ liệu.
- Tích hợp thành công Jikan API:** Hệ thống đã tích hợp và sử dụng hiệu quả Jikan API để truy xuất dữ liệu Anime công cộng, bao gồm thông tin chi tiết, hình ảnh, và khả năng tìm kiếm, lọc, cũng như hiển thị Anime ngẫu nhiên. Điều này cho phép ứng dụng có một cơ sở dữ liệu Anime phong phú mà không cần tự xây dựng từ đầu.
- Giao diện cơ bản, responsive trên desktop và mobile:** Giao diện người dùng được thiết kế đơn giản, trực quan và có khả năng responsive, đảm bảo ứng dụng hiển thị tốt và dễ sử dụng trên cả máy tính để bàn và các thiết bị di động.
- Rèn luyện kỹ năng full-stack:** Thông qua quá trình thực hiện dự án, nhà phát triển đã củng cố và nâng cao đáng kể các kỹ năng về frontend (HTML, CSS, JavaScript), backend (Node.js, Express.js) và quản lý cơ sở dữ liệu (MySQL), cùng với việc áp dụng các nguyên tắc thiết kế RESTful API và bảo mật cơ bản.

2. Hạn chế

Mặc dù đã đạt được các mục tiêu quan trọng, dự án vẫn còn một số hạn chế cần được cải thiện trong các phiên bản phát triển tiếp theo:

- Chưa có feature comments/rating:** Hiện tại, hệ thống chưa hỗ trợ các tính năng tương tác cộng đồng như bình luận hoặc đánh giá Anime trực tiếp từ người dùng. Các đánh giá hiển thị chỉ là điểm số trung bình từ Jikan API.
- Bảo mật JWT chưa nâng cao:** Chưa sử dụng JWT để xác thực, cơ chế quản lý token (ví dụ: refresh tokens, token revocation) chưa được triển khai

để tối ưu hóa bảo mật và trải nghiệm người dùng trong các phiên dài. Việc lưu trữ token ở localStorage cũng có thể tiềm ẩn rủi ro XSS nếu không được xử lý cẩn thận.

- **Xử lý lỗi và thông báo cho người dùng:** Mặc dù có xử lý lỗi cơ bản, hệ thống có thể cải thiện việc hiển thị thông báo lỗi chi tiết và thân thiện hơn cho người dùng cuối.

3. Hướng phát triển

Dựa trên kết quả đạt được và các hạn chế hiện có, dự án có thể được phát triển theo các hướng sau để nâng cao trải nghiệm người dùng và mở rộng tính năng:

- **Xây dựng gợi ý cá nhân:** Triển khai một hệ thống gợi ý Anime dựa trên lịch sử xem, thể loại yêu thích, hoặc đánh giá của người dùng. Điều này có thể sử dụng các thuật toán đơn giản hoặc tích hợp các thư viện học máy.
- **Phát triển ứng dụng di động:** Mở rộng ứng dụng sang nền tảng di động để người dùng có thể quản lý Watchlist của họ mọi lúc mọi nơi. Các framework đa nền tảng như React Native hoặc Flutter là những lựa chọn tiềm năng. React Native cho phép xây dựng ứng dụng di động bằng JavaScript với một codebase duy nhất cho cả Android và iOS, giúp tiết kiệm thời gian và nguồn lực. Flutter, được phát triển bởi Google, cũng cho phép tạo ứng dụng gốc cho di động, web và máy tính từ một codebase duy nhất, sử dụng ngôn ngữ Dart.
- **Cải thiện UI/UX:** Tiếp tục tối ưu hóa giao diện người dùng và trải nghiệm người dùng dựa trên các nguyên tắc thiết kế UI/UX hiện đại (ví dụ: tính nhất quán, đơn giản, giảm tải nhận thức). Điều này bao gồm việc tinh chỉnh bố cục, cải thiện hiệu ứng chuyển động, và đảm bảo phản hồi trực quan tốt hơn.
- **Hỗ trợ đa ngôn ngữ:** Triển khai tính năng đa ngôn ngữ để ứng dụng có thể phục vụ người dùng từ các quốc gia khác nhau, hiển thị nội dung bằng ngôn ngữ phù hợp (ví dụ: Tiếng Việt, Tiếng Anh, Tiếng Nhật).
- **Nâng cao bảo mật:** Cải thiện cơ chế bảo mật JWT bằng cách triển khai refresh tokens và cơ chế thu hồi token để tăng cường an toàn cho phiên người dùng. Ngoài ra, áp dụng các biện pháp bảo mật backend nâng cao hơn như sử dụng middleware Helmet để thiết lập các HTTP headers bảo mật.
- **Tích hợp tính năng theo dõi tập đã xem:** Cho phép người dùng ghi lại số tập đã xem của mỗi bộ Anime, thay vì chỉ trạng thái chung.