

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



ĐỒ ÁN TỐT NGHIỆP

XÂY DỰNG ỨNG DỤNG WEBSITE NGHE NHẠC SỬ DỤNG NUXTJS VÀ LARAVEL

SINH VIÊN THỰC HIỆN : ĐINH CÔNG HOÀNG

MÃ SINH VIÊN: 1451020096

KHOA: CÔNG NGHỆ THÔNG TIN

HÀ NỘI - 2024

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



ĐINH CÔNG HOÀNG

XÂY DỰNG ỨNG DỤNG WEBSITE NGHE NHẠC SỬ DỤNG NUXTJS VÀ LARAVEL

CHUYÊN NGÀNH : CÔNG NGHỆ THÔNG TIN
MÃ SỐ : 74.80.201

NGƯỜI HƯỚNG DẪN: Th.S ĐẶNG KHÁNH TRUNG

HÀ NỘI - 2024

LỜI CAM ĐOAN

Em xin cam đoan rằng đồ án này là kết quả nghiên cứu và làm việc của cá nhân em, được thực hiện dưới sự hướng dẫn của thầy Th.S Đặng Khánh Trung và tập thể giảng viên Khoa Công nghệ thông tin trường Đại học Đại Nam.

Toàn bộ nội dung trong đồ án, bao gồm ý tưởng, thiết kế, mã nguồn và các phần liên quan khác đều là sản phẩm của sự nỗ lực và sáng tạo cá nhân. Mọi thông tin tham khảo từ các nguồn tài liệu, trang web hay các nghiên cứu khác đã được trích dẫn đầy đủ và rõ ràng theo quy định.

Em xin chịu hoàn toàn trách nhiệm về tính trung thực và chính xác của nội dung đồ án này. Trong trường hợp có bất kỳ khiếu nại hay tranh chấp nào về quyền sở hữu trí tuệ, em xin chịu hoàn toàn trách nhiệm trước nhà trường và pháp luật.

Hà Nội, ngày 24 tháng 4 năm 2024

Trân trọng

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành đến thầy Th.S Đặng Khánh Trung, người đã dành nhiều thời gian, công sức và tâm huyết để hướng dẫn, động viên và giúp đỡ tôi hoàn thành đồ án này.

Em xin bày tỏ lòng biết ơn đến tất cả các thầy cô trong Khoa Công nghệ thông tin trường Đại học Đại Nam đã truyền đạt cho em những kiến thức quý báu và hỗ trợ em trong suốt quá trình học tập tại trường. Những bài giảng và kinh nghiệm của các thầy cô là nền tảng vững chắc giúp em hoàn thiện bản thân và thực hiện thành công đồ án này.

Em cũng xin cảm ơn sự ủng hộ và động viên từ gia đình, bạn bè và các đồng nghiệp. Gia đình luôn là chỗ dựa vững chắc, tạo điều kiện tốt nhất để tôi tập trung học tập và nghiên cứu. Bạn bè và đồng nghiệp đã cùng chia sẻ, góp ý và hỗ trợ em trong quá trình thực hiện đồ án.

Em xin chân thành cảm ơn!

LỜI NÓI ĐẦU

Trong bối cảnh phát triển mạnh mẽ của công nghệ thông tin, việc ứng dụng các công nghệ mới vào các lĩnh vực khác nhau đang trở thành xu hướng tất yếu. Một trong những lĩnh vực được hưởng lợi nhiều nhất từ sự phát triển này chính là lĩnh vực giải trí trực tuyến, đặc biệt là âm nhạc. Với mong muốn tạo ra một nền tảng nghe nhạc trực tuyến thân thiện, tiện ích và dễ sử dụng, em đã lựa chọn đề tài ***"Xây dựng ứng dụng website nghe nhạc sử dụng NuxtJS và Laravel"*** cho đồ án tốt nghiệp của mình.

Vì đồ án được thực hiện trong phạm vi thời gian hạn hẹp và hạn chế về mặt kiến thức chuyên môn, do đó bài báo cáo của em không thể tránh khỏi những sai sót nhất định. Đồng thời bản thân báo cáo là kết quả của một quá trình tổng kết, thu thập kết quả từ việc khảo sát thực tế, những bài học đúc rút từ trong quá trình học tập và làm đồ án của em. Em rất mong có được những ý kiến đóng góp của thầy, cô để bài báo cáo và bản thân em hoàn thiện hơn.

[illegible]

DANH MỤC HÌNH ẢNH

Hình 1: Trang chủ NuxtJS	4
Hình 2: Nuxt Lifecycle Hooks	5
Hình 3: Trang chủ Tailwind.....	7
Hình 4: Trang chủ Laravel.....	10
Hình 5: Ưu điểm của Laravel	12
Hình 6: Mô hình MVC	16
Hình 7: Cách thức hoạt động.....	17
Hình 8: Giao diện web Pusher.....	17
Hình 9: Giao diện quản lý channels pusher.....	18
Hình 10: Biểu đồ tổng quát trang người dùng.....	25
Hình 11: Biểu đồ tổng quát trang admin	26
Hình 12: Biểu đồ phân rã của use case nghe nhạc	27
Hình 13: Biểu đồ phân rã của use case Quản lý Album	27
Hình 14: Biểu đồ phân rã của use case Quản lý bình luận.....	28
Hình 15: Biểu đồ phân rã của use case quản lý tài khoản	29
Hình 16: Biểu đồ phân rã của use case Quản lý tương tác.....	29
Hình 17: Biểu đồ phân rã của use case quản lý bài hát	30
Hình 18: Biểu đồ phân rã của use case Xem bài hát	30
Hình 19: Biểu đồ phân rã của use case Quản lý thể loại.....	31
Hình 20: Biểu đồ phân rã của use case Quản lý người dùng	31
Hình 21: Biểu đồ phân rã của use case thống kê.....	32
Hình 22: Biểu đồ lớp thực thể	36
Hình 23: Biểu đồ lớp phân tích của use case quản lý thể loại.....	37

Hình 24: Biểu đồ lớp phân tích của use case quản lý album.....	37
Hình 25: Biểu đồ lớp phân tích của use case quản lý comment.....	38
Hình 26: Biểu đồ lớp phân tích của use case quản lý tương tác.....	38
Hình 27: Biểu đồ lớp phân tích của use case quản lý playlist.....	39
Hình 28: Biểu đồ lớp phân tích của use case quản lý bài hát.....	39
Hình 29: Biểu đồ lớp phân tích của use case thêm bài hát.....	40
Hình 30: Biểu đồ lớp phân tích của use case thêm bài hát vào playlist	40
Hình 31: Biểu đồ lớp phân tích của use case lấy bài hát theo loại	41
Hình 32: Biểu đồ tuần tự chức năng quản lý thể loại	42
Hình 33: Biểu đồ tuần tự chức năng xem bài hát theo thể loại	42
Hình 34: Biểu đồ tuần tự chức năng quản lý comment	43
Hình 35: Biểu đồ tuần tự chức năng quản lý playlist	43
Hình 36: Biểu đồ tuần tự chức năng quản lý bài hát	44
Hình 37: Biểu đồ tuần tự chức năng quản lý album.....	44
Hình 38: Biểu đồ hoạt động chức năng quản lý thể loại	45
Hình 39: Biểu đồ hoạt động chức năng quản lý album	46
Hình 40: Biểu đồ hoạt động chức năng quản lý playlist	46
Hình 41: Biểu đồ hoạt động chức năng quản lý bài hát	47
Hình 42: Biểu đồ hoạt động chức năng xem sản phẩm theo loại	47
Hình 43: Biểu đồ hoạt động chức năng bình luận.....	48
Hình 44: Trang đăng nhập	57
Hình 45: Trang đăng ký.....	57
Hình 46: Trang chủ.....	58
Hình 47: Trang xem bài hát theo thể loại	58

Hình 48: Trang tìm kiếm	59
Hình 49: Trang chi tiết bài hát.....	59
Hình 50: Bình luận	60
Hình 51: Trang chi tiết nghệ sĩ	60
Hình 52: Trang cá nhân người dùng.....	61
Hình 53: Trang sửa thông tin người dùng	61
Hình 54: Trang chi tiết playlist.....	62
Hình 55: Giao diện thêm bài hát.....	62
Hình 56: Trang thống kê.....	63

DANH MỤC BẢNG

Bảng 1: Bảng Genre	48
Bảng 2: Bảng Album	49
Bảng 3: Bảng playlist	49
Bảng 4: Bảng Song	50
Bảng 5: Bảng User	51
Bảng 6: Bảng Comment	52
Bảng 7: Bảng GenreSong	52
Bảng 8: Bảng PlaylistSong	53
Bảng 9: Bảng Interaction	53

MỤC LỤC

Chương 1. Tổng quan về đề tài	1
1.1. Lý do chọn đề tài.....	1
1.2. Mục đích nghiên cứu.....	1
1.3. Phạm vi nghiên cứu.....	1
1.4. Phương pháp nghiên cứu.....	2
1.5. Bố cục đồ án.....	3
Chương 2. Cơ sở lý thuyết.....	4
2.1. Giới thiệu về NuxtJS	4
2.1.1. Khái niệm.....	4
2.1.2. Các tính năng của NuxtJS.....	5
2.2. Giới thiệu về TailWind	6
2.2.1. Khái niệm.....	6
2.2.2. Đặc điểm chính của Tailwind CSS	7
2.2.3. Lợi ích khi sử dụng Tailwind CSS.....	8
2.3. Giới thiệu về Laravel.....	8
2.3.1. Khái niệm Laravel	8
2.3.2. Các tính năng chính	10
2.3.3. Ưu điểm và nhược điểm của Laravel.....	11
2.3.4. Mô hình MVC.....	13
2.4. Giới thiệu về Pusher	16
2.4.1. Khái niệm pusher	16
2.4.2. Ưu và nhược điểm của Pusher	18
2.4.3. Sử dụng với Laravel.....	19

Chương 3: Phân tích thiết kế hệ thống	21
3.1. Khảo sát thực tế.....	21
3.2. Đặc tả yêu cầu phần mềm	22
3.2.1. Yêu cầu chức năng.....	22
3.2.2. Yêu cầu phi chức năng.....	23
3.2.3. Các yêu cầu hệ thống khác	23
3.3. Phân tích thiết kế hệ thống sử dụng UML	24
3.3.1. Biểu đồ ca sử dụng.....	24
3.3.2. Kịch bản các ca sử dụng	32
3.3.3. Biểu đồ lớp thực thể.....	36
3.3.4. Biểu đồ lớp phân tích ca sử dụng	36
3.3.5. Biểu đồ tuần tự.....	41
3.3.6. Biểu đồ hoạt động.....	45
3.4. Thiết kế cơ sở dữ liệu.....	48
Chương 4. Triển khai hệ thống và đánh giá kết quả đạt được.....	55
4.1. Cài đặt môi trường	55
4.2. Cấu Trúc Tổ Chức Dữ Liệu.....	55
4.2.1. Cấu trúc thư mục và file cho NuxtJS.....	55
4.2.2. Cấu Trúc Tổ Chức Dữ Liệu của Laravel (Backend).....	56
4.3. Phát triển ứng dụng	56
4.3.1. Phát triển FrontEnd.....	56
4.3.2. Phát triển API.....	63
4.4. Đánh giá kết quả và hướng phát triển	65
KẾT LUẬN	67

TÀI LIỆU THAM KHẢO	68
--------------------------	----

Chương 1. Tổng quan về đề tài

1.1. Lý do chọn đề tài

Trong thời đại số hóa ngày nay, người dùng ngày càng dựa vào internet để tìm kiếm và thưởng thức âm nhạc, đây là một hình thức giải trí quan trọng, thu hút đông đảo người dùng ở mọi lứa tuổi. Bên cạnh đó là sự phát triển của ngành âm nhạc hiện đại cùng với đông đảo số lượng nghệ sĩ muốn đưa âm nhạc của mình đến với người nghe. Do đó, việc phát triển một trang web nghe nhạc không chỉ đáp ứng được nhu cầu giải trí mà còn góp phần vào sự phát triển của công nghệ web và ứng dụng số.

Xây dựng một trang web nghe nhạc không chỉ là quá trình cung cấp một dịch vụ giải trí, mà còn là cách để áp dụng và thực hành các công nghệ web tiên tiến. Người phát triển có thể học hỏi và sử dụng các công nghệ như HTML5, CSS3, và JavaScript, NuxtJS, Tailwind cũng như khám phá các framework và thư viện phía server như Laravel, MySQL. Việc này không những giúp nâng cao kỹ năng lập trình và phát triển web mà còn giúp họ hiểu sâu hơn về cách thức hoạt động của các ứng dụng web động.

Chính vì những lý do này em đã chọn đề tài **Xây dựng ứng dụng Website nghe nhạc sử dụng NuxtJS và Laravel.**

1.2. Mục đích nghiên cứu

Mục đích nghiên cứu của đề tài xây dựng trang web nghe nhạc nhằm vào việc phát triển một nền tảng thân thiện với người dùng, cung cấp trải nghiệm nghe nhạc trực tuyến nhanh chóng và hiệu quả. Để đạt được điều này, dự án sẽ tập trung vào việc áp dụng các công nghệ web hiện đại như NuxtJs và Laravel, cùng với việc khám phá ứng dụng của các framework phát triển web phổ biến. Ngoài ra, đề tài cũng nhằm mục đích thu thập và phân tích dữ liệu người dùng để hiểu rõ hơn về hành vi và sở thích của họ, từ đó cá nhân hóa trải nghiệm người dùng và nâng cao hiệu quả của trang web. Qua đó, dự án không chỉ mang lại giá trị giải trí cho người dùng mà còn góp phần nâng cao kỹ năng, kiến thức chuyên môn.

1.3. Phạm vi nghiên cứu

Đầu tiên, việc phát triển giao diện người dùng sẽ tập trung vào việc sử dụng NuxtJS, một framework dựa trên VueJS. Mục tiêu là tạo ra một trang web có khả năng phản hồi tốt,

nhanh chóng, và thân thiện với người dùng. NuxtJS cung cấp các tính năng như Server-Side Rendering (SSR) hoặc Static Site Generation (SSG), giúp cải thiện tốc độ tải trang và tối ưu hóa cho công cụ tìm kiếm (SEO). Các yếu tố thiết kế UX/UI sẽ được chú trọng để đảm bảo trải nghiệm người dùng mượt mà và dễ chịu khi truy cập và sử dụng trang web.

Mảng backend sẽ được xây dựng dựa trên Laravel, một framework PHP mạnh mẽ cho phát triển web và API. Công việc chính bao gồm thiết kế và triển khai một RESTful API để xử lý các yêu cầu từ phía frontend. API sẽ quản lý các chức năng chính như đăng nhập/người dùng, streaming nhạc, quản lý playlist và thu thập cũng như phân tích dữ liệu người dùng. Laravel cũng sẽ giúp tối ưu hóa quá trình quản lý cơ sở dữ liệu, đảm bảo truy xuất dữ liệu nhanh chóng và an toàn.

Việc tích hợp giữa NuxtJS và Laravel API phải đảm bảo suôn sẻ, với các kết nối API được xử lý hiệu quả để duy trì tính bảo mật và hiệu suất cao. Xác thực người dùng và quản lý phiên làm việc sẽ được thực hiện thông qua các công nghệ như OAuth hoặc JWT (JSON Web Tokens). Việc này không chỉ giúp duy trì tính liên tục và an toàn của phiên người dùng mà còn cải thiện độ tin cậy và tính chuyên nghiệp của trang web.

Một yếu tố quan trọng khác trong phạm vi nghiên cứu là đảm bảo trang web tuân thủ các quy định về bản quyền âm nhạc và bảo vệ dữ liệu cá nhân của người dùng. Các biện pháp bảo mật và tuân thủ pháp lý sẽ được triển khai để bảo vệ nền tảng khỏi các rủi ro pháp lý và tăng cường an ninh thông tin. Các chính sách bảo mật, mã hóa dữ liệu và quản lý quyền truy cập sẽ được thiết lập một cách nghiêm ngặt để đảm bảo trang web hoạt động trong khuôn khổ pháp luật và đạo đức.

1.4. Phương pháp nghiên cứu

Bước đầu tiên trong dự án là tiến hành nghiên cứu sâu rộng về nhu cầu của người dùng và yêu cầu kỹ thuật cho ứng dụng web nghe nhạc. Điều này bao gồm việc phân tích thị trường hiện tại và thu thập ý kiến phản hồi từ người dùng tiềm năng để hiểu rõ họ mong đợi những tính năng nào và các vấn đề cần giải quyết qua trang web mới. Dữ liệu thu thập được sẽ giúp định hình các tính năng chính cần phát triển, như tạo và quản lý playlist, tìm kiếm bài hát, nghe nhạc và tính năng bình luận.

Sau khi đã rõ ràng về yêu cầu, lập kế hoạch chi tiết và thiết kế cho ứng dụng. Quá trình này bao gồm việc thiết kế giao diện người dùng sử dụng các nguyên tắc thiết kế UX/UI hiện đại để tạo ra trải nghiệm người dùng mượt mà và thân thiện.

Với kế hoạch và thiết kế đã được xác nhận, bước tiếp theo là phát triển ứng dụng. NuxtJS sẽ được sử dụng để phát triển phần frontend, trong khi Laravel API sẽ xử lý backend. Hệ thống sẽ được tích hợp với Pusher để thực hiện các chức năng như bình luận thời gian thực, mang đến sự tương tác và gắn kết cao cho người dùng.

Giai đoạn cuối cùng của dự án là phân tích kết quả và đánh giá hiệu quả của việc sử dụng NuxtJS và Laravel trong việc phát triển trang web nghe nhạc. Điều này bao gồm việc kiểm tra kỹ lưỡng hiệu suất ứng dụng, tính năng, và độ tin cậy thông qua các bài kiểm thử và thu thập phản hồi từ người dùng cuối. Dựa trên những phân tích này, nhóm phát triển có thể xác định các điểm mạnh cũng như các lĩnh vực cần cải thiện, từ đó đề xuất các giải pháp để nâng cao chất lượng sản phẩm và trải nghiệm người dùng.

1.5. Bố cục đề án

Bố cục của đề án này gồm 4 chương:

Chương 1: Giới thiệu tổng quan về lý do và tính cấp thiết của đề tài này. Trong chương này cũng trình bày về các vấn đề cần giải quyết, mục đích và phạm vi nghiên cứu của đề tài.

Chương 2: Trình bày về cơ sở lý thuyết được sử dụng trong đề tài bao gồm các nội dung về các ngôn ngữ lập trình.

- Đầu tiên sẽ giới thiệu về NuxtJS và các tính năng của nó.
- Phần tiếp theo sẽ là phần Tailwind, các đặc điểm và lợi ích chính khi sử dụng.
- Thứ 3 sẽ giới thiệu về Laravel, các tính năng chính của nó và mô hình MVC.
- Phần cuối cùng sẽ là về Pusher và cách thức hoạt động của Pusher.

Chương 3: Sau khi tìm hiểu về các công nghệ sẽ sử dụng trong đề án, tiếp theo sẽ là phần khảo sát, phân tích thiết kế hệ thống, trình bày về các chức năng chính, luồng hoạt động của website.

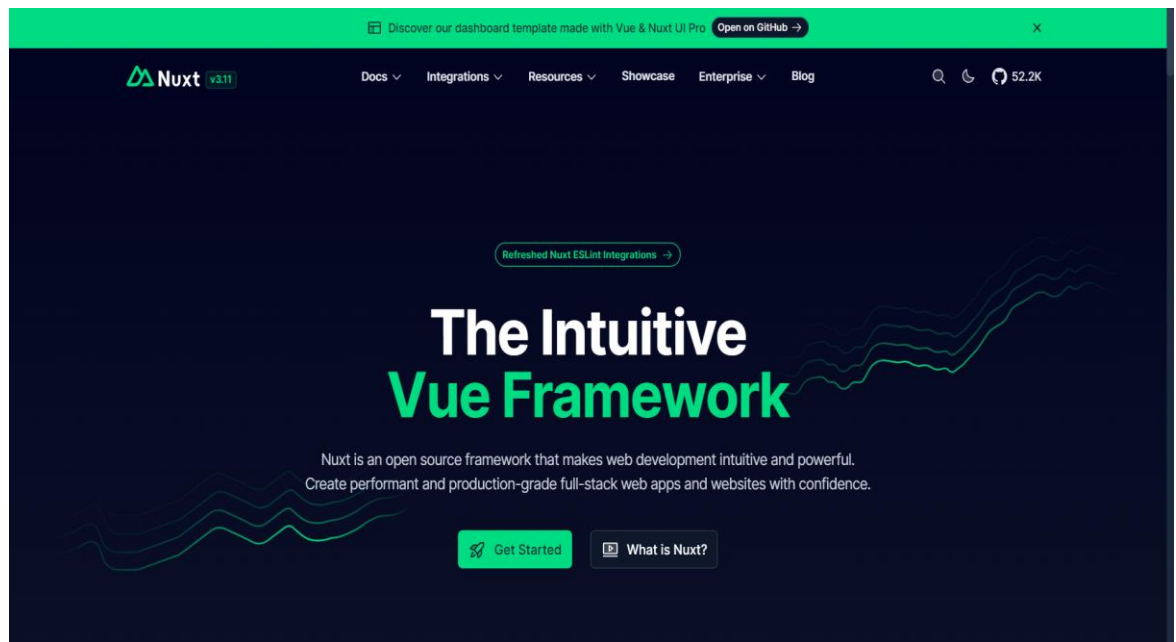
Chương 4: Cuối cùng sẽ là phần xây dựng website, đánh giá kết quả đạt được.

Chương 2. Cơ sở lý thuyết

2.1. Giới thiệu về NuxtJS

2.1.1. Khái niệm

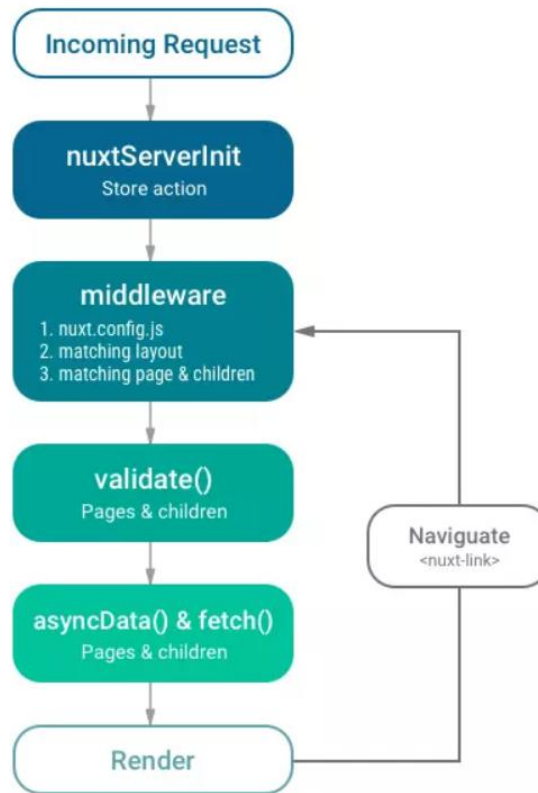
NuxtJS là một framework dựa trên VueJS, được thiết kế để tạo điều kiện thuận lợi và tối ưu hóa quá trình phát triển các ứng dụng web phức tạp dựa trên Vue. NuxtJS cung cấp một cấu trúc dự án rõ ràng và các tính năng mạnh mẽ để giúp các nhà phát triển dễ dàng xây dựng các giao diện người dùng tương tác và hiện đại.



Hình 1: Trang chủ NuxtJS

Nuxt Lifecycle Hooks là các phương thức đặc biệt mà NuxtJS cung cấp để giúp bạn can thiệp vào các giai đoạn khác nhau của vòng đời ứng dụng hoặc trang. Chúng giúp bạn thực hiện các tác vụ cụ thể vào những thời điểm nhất định trong quá trình xử lý và render. `nuxtServerInit`: Được gọi một lần duy nhất trên server khi khởi tạo ứng dụng.

- `middleware`: Các tệp middleware có thể chạy trước khi mỗi route được render.
- `validate`: Được sử dụng để kiểm tra tính hợp lệ của route parameters.
- `layout`: Xác định layout nào sẽ được sử dụng cho trang hiện tại.
- `transition`: Định nghĩa hiệu ứng chuyển tiếp giữa các trang.



Hình 2: Nuxt Lifecycle Hooks

2.1.2. Các tính năng của NuxtJS

- **Server-Side Rendering (SSR):** Server side là kịch bản thường được sử dụng bởi máy chủ để người sử dụng tùy chỉnh website đang truy cập. Server side là một kỹ thuật được dùng để phát triển website. Bên cạnh chức năng cung cấp việc hiển thị tập tin website, nó còn phản hồi cho server biết các tùy chỉnh của người truy cập đối với website. NuxtJS hỗ trợ SSR giúp các ứng dụng của bạn được render trên phía server trước khi gửi tới trình duyệt.
- **Automatic Code Splitting:** NuxtJS tự động tách code cho mỗi route để chỉ các tài nguyên cần thiết cho trang hiện tại mới được tải, giúp tăng tốc độ tải trang.
- **Vuex:** NuxtJS hỗ trợ tích hợp sẵn Vuex để quản lý trạng thái ứng dụng, làm cho việc quản lý trạng thái trở nên đơn giản và dễ dàng tích hợp vào các ứng dụng lớn.
- **Powerful Plugin Architecture:** Hệ thống plugin của NuxtJS cho phép bạn dễ dàng tích hợp và mở rộng chức năng của ứng dụng VueJS, chẳng hạn như axios hoặc vuex.

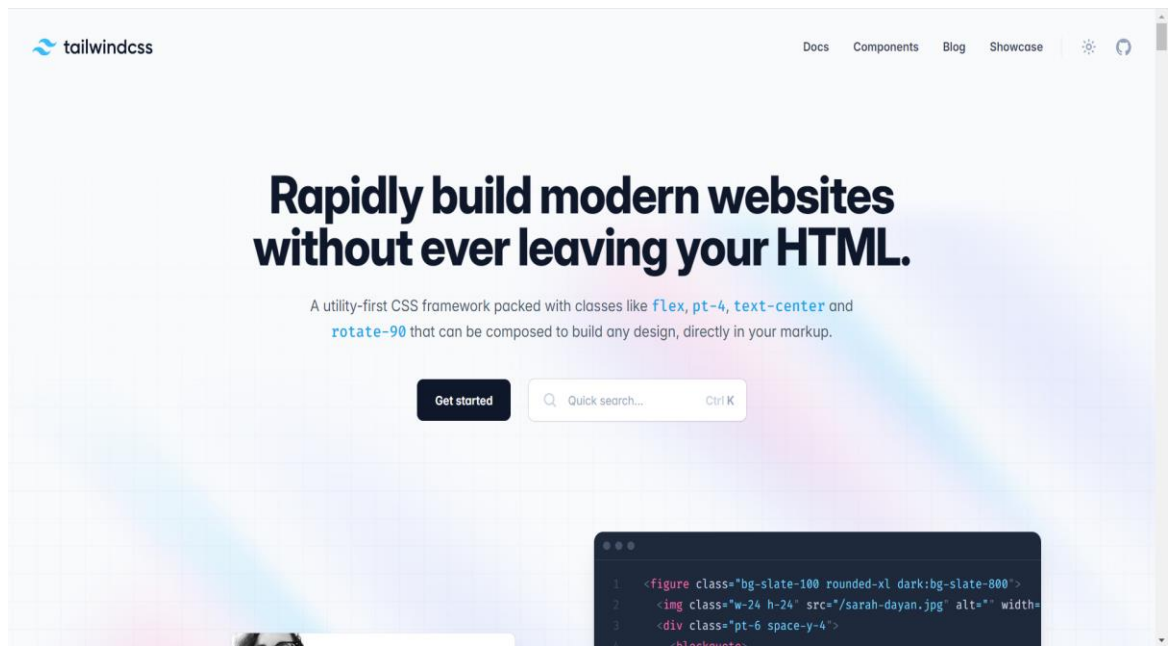
- **File-based Routing:** Cấu trúc thư mục pages của NuxtJS tự động tạo các routes dựa trên các file Vue trong thư mục đó. Điều này giúp quản lý routing trở nên dễ dàng và ít rắc rối hơn.
- **Asynchronous Data Handling:** NuxtJS cung cấp tính năng `asyncData` và `fetch` để xử lý dữ liệu bất đồng bộ, cho phép bạn lấy dữ liệu trước khi component được render, rất hữu ích cho SSR.
- **Layouts and Custom Error Pages:** NuxtJS cho phép tạo các layout khác nhau và trang lỗi tùy chỉnh, giúp tạo nên một ứng dụng đồng nhất và dễ dàng quản lý.
- **Middleware:** NuxtJS cho phép bạn định nghĩa các middleware, có thể chạy trước khi trang được render hoặc trước khi route được navigate tới, giúp bạn có thể xử lý xác thực, kiểm tra quyền truy cập, và các tác vụ khác.
- **SEO Optimization:** NuxtJS cung cấp các cấu hình dễ dàng cho các thẻ meta và quản lý SEO, giúp cải thiện khả năng hiển thị của trang web trên các công cụ tìm kiếm.

Những tính năng này làm cho NuxtJS trở thành một lựa chọn hấp dẫn cho các nhà phát triển muốn xây dựng các ứng dụng VueJS nhanh chóng, hiệu quả, với khả năng tối ưu hóa SEO và hiệu suất cao.

2.2. Giới thiệu về TailWind

2.2.1. Khái niệm

Tailwind CSS là một framework CSS tiện ích trước tiên (utility-first) được thiết kế để giúp các nhà phát triển xây dựng các giao diện người dùng tùy chỉnh nhanh chóng và hiệu quả mà không cần phải viết nhiều mã CSS từ đầu. Khác với các framework CSS truyền thống như Bootstrap hay Foundation, Tailwind cung cấp một tập hợp rộng lớn các lớp tiện ích (utility classes) mà bạn có thể áp dụng trực tiếp vào các phần tử HTML để kiểm soát gần như mọi khía cạnh của thiết kế trang web.



Hình 3: Trang chủ Tailwind

2.2.2. Đặc điểm chính của Tailwind CSS

- **Utility-First:** Tailwind cho phép xây dựng thiết kế bằng cách sử dụng các lớp tiện ích với mục đích duy nhất. Mỗi lớp tiện ích áp dụng một thuộc tính CSS duy nhất hoặc một tập hợp các thuộc tính tương quan, giúp bạn tạo kiểu nhanh chóng và dễ dàng chỉnh sửa.
- **Highly Customizable:** Một trong những lợi thế lớn của Tailwind là tính cấu hình cao. Nó cho phép bạn tùy chỉnh cấu hình để phù hợp với thiết kế và hệ thống thiết kế của bạn, từ màu sắc, font, khoảng cách, kích thước, biên và nhiều hơn nữa.
- **Responsive Design:** Tailwind hỗ trợ thiết kế đáp ứng ngay lập tức với các lớp tiện ích được tiền tố bởi các điểm ngắt (breakpoints). Điều này cho phép các nhà phát triển dễ dàng áp dụng các kiểu khác nhau cho các thiết bị ở các kích thước màn hình khác nhau.
- **Component-Friendly:** Mặc dù Tailwind là một framework utility-first, nó cũng hỗ trợ việc tạo các thành phần có thể tái sử dụng. Bạn có thể xây dựng các thành phần bằng cách sử dụng các lớp tiện ích và tái sử dụng chúng trong toàn bộ dự án của mình.
- **No Built-In Components:** Khác với Bootstrap, Tailwind không đi kèm với các thành phần giao diện người dùng được xây dựng sẵn. Điều này giúp giảm bớt kích thước của tập tin cuối cùng và tránh tải những mã không cần thiết vào trang web của bạn.

- **PurgeCSS Integration:** Tailwind CSS tích hợp sẵn với PurgeCSS, công cụ này giúp loại bỏ các CSS không được sử dụng trước khi trang web được triển khai, tối ưu hóa tốc độ tải trang bằng cách giảm kích thước tập tin CSS.

2.2.3. Lợi ích khi sử dụng Tailwind CSS

- **Tốc độ phát triển nhanh:** Việc sử dụng các lớp tiện ích giúp bạn nhanh chóng xây dựng giao diện mà không cần phải viết lại mã CSS.
- **Dễ dàng bảo trì:** Cập nhật kiểu dáng của một trang web hoặc ứng dụng là dễ dàng hơn vì thay đổi chỉ cần thực hiện ở mức lớp tiện ích.
- **Sự nhất quán:** Khi được cấu hình đúng, Tailwind có thể giúp duy trì sự nhất quán trong thiết kế trên toàn bộ ứng dụng.

Tailwind CSS phù hợp với việc phát triển website nghe nhạc với việc tích hợp tốt với NuxtJS. Với hệ thống utility-first của Tailwind CSS, có thể nhanh chóng áp dụng các lớp CSS trực tiếp trong các component của Nuxt, giúp tăng tốc quá trình phát triển.

2.3. Giới thiệu về Laravel

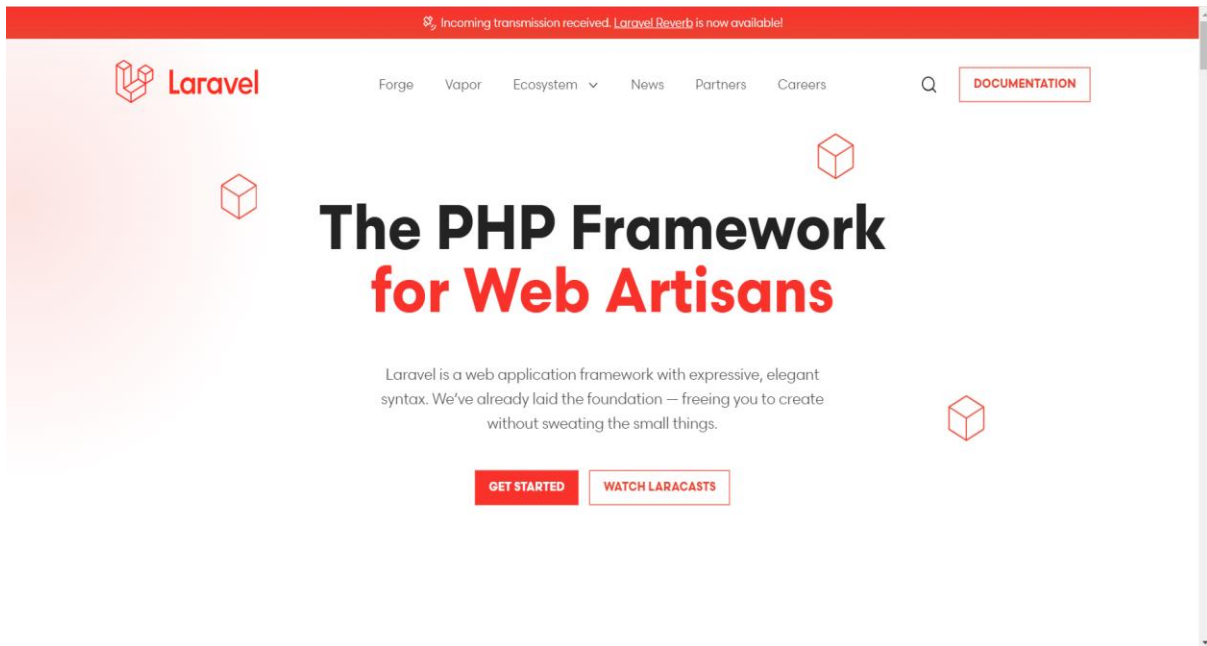
2.3.1. Khái niệm Laravel

PHP framework là thư viện làm cho sự phát triển của những ứng dụng web viết bằng ngôn ngữ PHP trở nên trôi chảy hơn. Bằng cách cung cấp 1 cấu trúc cơ bản để xây dựng những ứng dụng đó. Hay nói cách khác, PHP framework giúp bạn thúc đẩy nhanh chóng quá trình phát triển ứng dụng. Giúp bạn tiết kiệm được thời gian, tăng sự ổn định cho ứng dụng. Giảm thiểu số lần phải viết lại code cho lập trình viên.

Laravel là một framework phát triển ứng dụng web mã nguồn mở được viết bằng ngôn ngữ PHP. Nó cung cấp các công cụ và thư viện để giúp các nhà phát triển xây dựng các ứng dụng web một cách nhanh chóng và hiệu quả. Laravel được thiết kế để giảm bớt công việc lặp lại thông qua việc cung cấp các tính năng như routing, session management, authentication và một ORM (Object-Relational Mapping) mạnh mẽ để tương tác với cơ sở dữ liệu. Nó cũng có hệ thống template mạnh mẽ, hỗ trợ các công cụ tốt cho kiểm tra và triển khai ứng dụng. Laravel được xem là một trong những framework PHP phổ biến nhất hiện nay.

Khi xây dựng một ứng dụng web bằng Laravel, các bước hoạt động chính sẽ là như sau:

- Định tuyến (Routing): Laravel cung cấp tính năng định tuyến mạnh mẽ, cho phép định tuyến các URL đến các hành động (Action) cụ thể trong ứng dụng web.
- Xử lý yêu cầu (Request Handling): Khi một yêu cầu được gửi đến ứng dụng web, Laravel sẽ xử lý yêu cầu và đưa ra phản hồi (Response) tương ứng.
- Xử lý dữ liệu (Data Handling): Laravel cung cấp các tính năng để xử lý dữ liệu trong ứng dụng web, bao gồm truy vấn CSDL, tạo và thao tác với các model (mô hình) dữ liệu, v.v.
- Xử lý logic (Logic Handling): Với kiến trúc MVC, Laravel cho phép phân tách logic ứng dụng web thành các phần riêng biệt, bao gồm Model (mô hình), View (giao diện) và Controller (bộ điều khiển).
- Tích hợp các thư viện và gói phần mềm (Package Integration): Laravel cho phép tích hợp các thư viện và gói phần mềm (package) từ cộng đồng Laravel và thư viện PHP khác để mở rộng tính năng của ứng dụng web.
- Kiểm thử (Testing): Laravel cung cấp các công cụ kiểm thử để đảm bảo tính đúng đắn và ổn định của ứng dụng web.
- Triển khai (Deployment): Sau khi hoàn thành phát triển ứng dụng web, Laravel cung cấp các tính năng để triển khai ứng dụng web trên các môi trường như máy chủ web (web server) hoặc các nền tảng đám mây (cloud platform).



Hình 4: Trang chủ Laravel

2.3.2. Các tính năng chính

Dưới đây là một số tính năng chính của Laravel:

- **Routing:** Laravel cung cấp một hệ thống routing mạnh mẽ và linh hoạt, cho phép các nhà phát triển định nghĩa các tuyến đường URL và xử lý các yêu cầu HTTP dễ dàng.
- **Middleware:** Laravel hỗ trợ middleware, cho phép các nhà phát triển xử lý các yêu cầu HTTP trước khi chúng được xử lý bởi ứng dụng.
- **Blade Template Engine:** Blade là một engine mẫu dựa trên PHP cung cấp các tính năng như kế thừa mẫu, phân mảnh mẫu và cú pháp ngắn gọn, giúp các nhà phát triển tạo ra các giao diện người dùng đẹp và dễ bảo trì.
- **Eloquent ORM:** Eloquent là một ORM (Object Relational Mapping) mạnh mẽ, giúp các nhà phát triển tương tác với cơ sở dữ liệu dễ dàng hơn bằng cách sử dụng các đối tượng PHP thay vì viết câu truy vấn SQL.
- **Migration:** Laravel cung cấp một hệ thống migration cho phép các nhà phát triển quản lý cơ sở dữ liệu của ứng dụng dễ dàng và an toàn.
- **Artisan CLI:** Artisan là một công cụ dòng lệnh mạnh mẽ cho phép các nhà phát triển tạo và quản lý các tệp mẫu, thực hiện các tác vụ cần thiết cho ứng dụng, và nhiều tính năng khác.

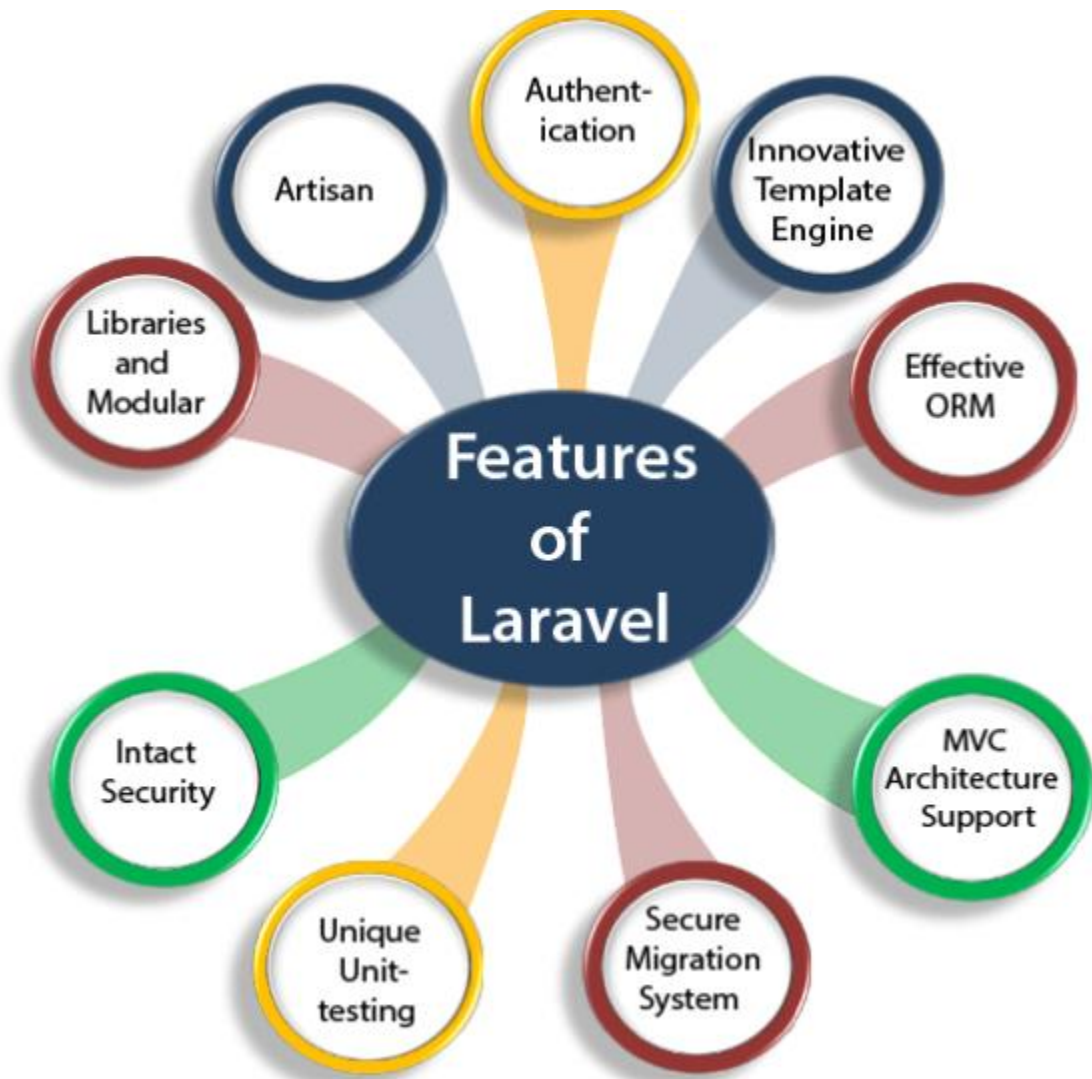
- **Authentication:** Laravel cung cấp các tính năng xác thực và phân quyền mạnh mẽ, giúp các nhà phát triển xây dựng các hệ thống đăng nhập và phân quyền dễ dàng và an toàn.
- **Queueing:** Laravel hỗ trợ queueing, cho phép các nhà phát triển xử lý các tác vụ nền một cách hiệu quả, giúp tăng tốc độ và hiệu suất của ứng dụng.

2.3.3. Ưu điểm và nhược điểm của Laravel

Laravel, như mọi framework, có những ưu và nhược điểm riêng. Dưới đây là một số ưu và nhược điểm của Laravel.

Ưu điểm:

- **Tiện ích và tính năng mạnh mẽ:** Laravel cung cấp một loạt các tính năng và công cụ mạnh mẽ như routing, ORM, authentication, caching, và nhiều tính năng khác giúp giảm thời gian phát triển và tăng hiệu suất.
- **Cộng đồng lớn và hỗ trợ tích cực:** Laravel có một cộng đồng lớn, nhiệt tình và tích cực, với nhiều tài liệu, hướng dẫn và gói mở rộng (packages) sẵn có.
- **Mô hình MVC mạnh mẽ:** Laravel áp dụng mô hình MVC, giúp tách biệt logic ứng dụng và giao diện người dùng, dễ dàng bảo trì và mở rộng mã nguồn.
- **Laravel Eloquent ORM:** Eloquent là một ORM mạnh mẽ và linh hoạt, giúp tương tác với cơ sở dữ liệu dễ dàng hơn và giảm thiểu việc viết mã SQL.
- **Hệ thống Template Blade:** Blade là một hệ thống template mạnh mẽ, linh hoạt và dễ sử dụng, giúp xây dựng các giao diện người dùng đẹp mắt và hiệu quả.



Hình 5: Ưu điểm của Laravel

Nhược điểm:

- Độ phức tạp: Đôi khi, việc học Laravel có thể mất một khoảng thời gian khá lớn, đặc biệt đối với những người mới bắt đầu với PHP hoặc framework PHP.
- Hiệu suất: Mặc dù Laravel cung cấp nhiều tính năng và tiện ích, nhưng đôi khi hiệu suất của nó có thể không tốt bằng một số framework khác do overhead của các tính năng này.
- Phiên bản và tương thích: Khi Laravel ra phiên bản mới, việc nâng cấp từ phiên bản cũ có thể gây ra một số vấn đề tương thích với các ứng dụng đã tồn tại.

Mặc dù có nhược điểm nhất định, Laravel vẫn là một trong những framework PHP phổ biến nhất và được nhiều nhà phát triển ưa chuộng với tính linh hoạt và mạnh mẽ của nó.

Việc sử dụng laravel để thiết backend cho trang web nghe nhạc mang lại nhiều lợi ích lớn và phù hợp để kết hợp cùng NuxtJS.

2.3.4. Mô hình MVC

Mô hình MVC (Model-View-Controller) là một kiến trúc phần mềm phổ biến được sử dụng trong việc phát triển ứng dụng web và các ứng dụng khác. Nó phân chia ứng dụng thành ba thành phần chính:

- **Model (Mô hình):** Đại diện cho dữ liệu và logic liên quan đến dữ liệu trong ứng dụng. Mô hình chịu trách nhiệm cho việc truy cập và xử lý dữ liệu từ cơ sở dữ liệu hoặc bất kỳ nguồn dữ liệu nào khác cần thiết. Nó không biết gì về cách dữ liệu được hiển thị hoặc tương tác với người dùng.
- **View (Cảnh):** Là phần giao diện người dùng của ứng dụng, chịu trách nhiệm cho việc hiển thị dữ liệu cho người dùng và tương tác với họ. View không chứa logic kinh doanh hoặc dữ liệu, nó chỉ hiển thị dữ liệu được cung cấp từ mô hình.
- **Controller (Bộ điều khiển):** Là trung gian giữa mô hình và cảnh, chịu trách nhiệm xử lý các yêu cầu từ người dùng, tương tác với mô hình để lấy dữ liệu, và sau đó chuyển dữ liệu này cho cảnh để hiển thị. Nó là nơi chứa logic kinh doanh và xử lý yêu cầu từ người dùng.

Mô hình MVC (Model-View-Controller) có một lịch sử phát triển và đã trở thành một trong những mô hình phát triển phần mềm phổ biến nhất.

Lịch sử:

- Mô hình MVC được phát triển lần đầu tiên vào những năm 1970 bởi Trygve Reenskaug, một kỹ sư người Na Uy, trong một dự án nghiên cứu tại Xerox PARC.
- Nó đã được sử dụng trong việc phát triển các ứng dụng desktop trên các hệ thống ngôn ngữ lập trình như Smalltalk.
- Sau đó, MVC đã được áp dụng trong các ngôn ngữ lập trình khác như Java, C++, và cuối cùng là web development với các framework như Ruby on Rails và Laravel.

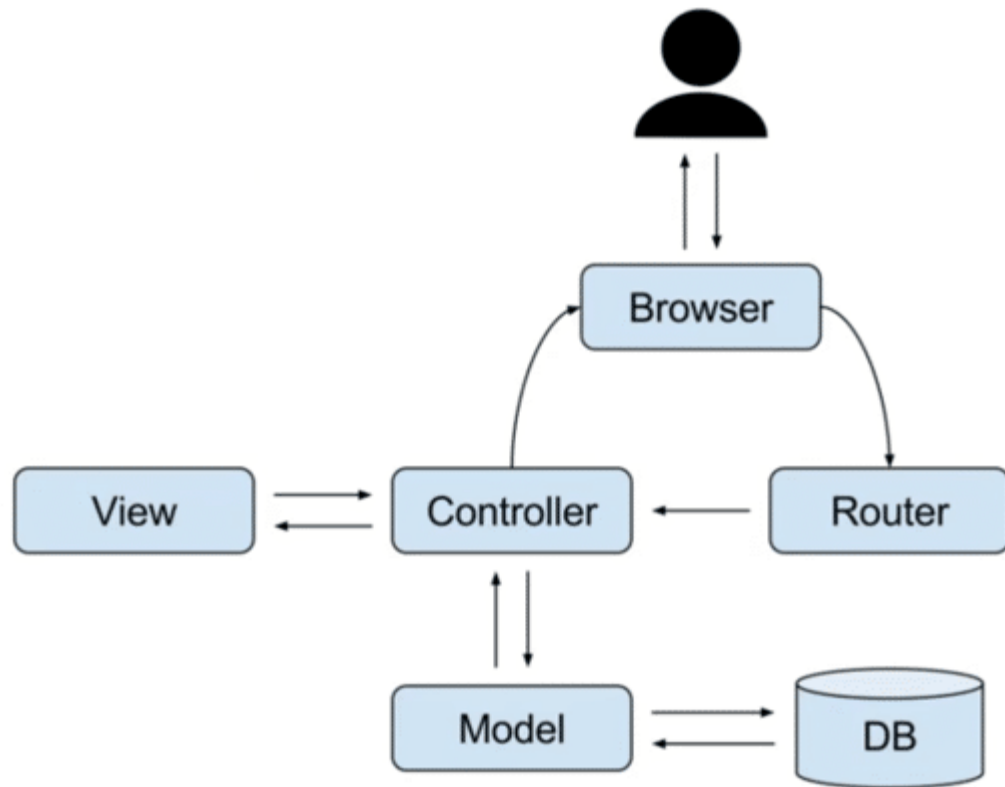
Ưu điểm:

- Tách biệt logic: Mô hình MVC giúp tách biệt logic dữ liệu (Model), giao diện người dùng (View) và điều khiển luồng logic (Controller), giúp cho mã nguồn dễ bảo trì và tái sử dụng.
- Tích hợp: Nó giúp nhiều người lập trình làm việc song song trên cùng một dự án mà không ảnh hưởng lẫn nhau, do mỗi thành phần có trách nhiệm rõ ràng.
- Phát triển đồng thời: Vì giao diện người dùng và logic dữ liệu được tách biệt, nhiều nhóm có thể làm việc đồng thời trên các phần khác nhau của dự án mà không gặp xung đột.
- Kiểm thử dễ dàng: Việc tách biệt các phần giúp việc kiểm thử trở nên dễ dàng hơn, vì bạn có thể kiểm tra logic mô hình và giao diện người dùng một cách riêng biệt.
- Nhược điểm:
- Độ phức tạp: Mô hình MVC có thể phức tạp đối với các dự án nhỏ, đặc biệt là đối với các dự án không yêu cầu sự phân tách rõ ràng giữa logic và giao diện người dùng.
- Khó khăn trong việc học và triển khai ban đầu: Người mới bắt đầu có thể cảm thấy khó khăn khi học cách sử dụng mô hình MVC và triển khai nó vào các dự án của họ.
- Tăng cường độ phức tạp khi quản lý dự án lớn: Trong các dự án lớn, việc quản lý các phần tử của mô hình MVC và các tương tác giữa chúng có thể trở nên phức tạp.

Các bước hoạt động của MVC

- Người dùng tương tác với View:
 - Người dùng thực hiện một hành động nào đó trên giao diện người dùng (UI) như nhấp chuột vào nút, nhập dữ liệu vào biểu mẫu, v.v.
 - Ví dụ: Người dùng nhấp vào nút "Tải bài hát" sau khi nhập nội dung bài hát vào một form.
- View gửi yêu cầu đến Controller:
 - View gửi yêu cầu (request) đến Controller để xử lý hành động của người dùng.
 - Ví dụ: Form "Lưu bài hát" gửi dữ liệu bài hát đến Controller.
- Controller xử lý yêu cầu:

- Controller nhận yêu cầu từ View và quyết định hành động cần thực hiện. Nó có thể tương tác với Model để truy xuất hoặc lưu trữ dữ liệu.
- Ví dụ: Controller nhận dữ liệu bài hát từ form và gọi phương thức create() của Model Song để lưu bài hát vào cơ sở dữ liệu.
- Controller cập nhật Model:
 - Nếu cần thiết, Controller sẽ gọi các phương thức của Model để cập nhật dữ liệu hoặc lấy dữ liệu từ cơ sở dữ liệu.
- Model thực hiện thao tác dữ liệu:
 - Model thực hiện các thao tác liên quan đến dữ liệu, chẳng hạn như lưu trữ, truy vấn, cập nhật hoặc xóa dữ liệu.
 - Ví dụ: Model Song lưu bài hát mới vào cơ sở dữ liệu.
- Model thông báo cho Controller:
 - Sau khi thực hiện thao tác, Model có thể thông báo cho Controller.
 - Ví dụ: Model Song có thể trả về bài hát mới sau khi lưu bài hát mới thành công.
- Controller cập nhật View:
 - Controller nhận phản hồi từ Model và quyết định cách cập nhật View để phản ánh trạng thái mới của dữ liệu.
 - Ví dụ: Controller lấy danh sách các bài hát mới và cập nhật View để hiển thị bài hát mới nhất cho người dùng.
- View hiển thị thông tin cho người dùng:
 - View nhận dữ liệu từ Controller và cập nhật giao diện người dùng để phản ánh thông tin mới nhất.
 - Ví dụ: View cập nhật danh sách bài hát trên trang để hiển thị bài hát mới vừa được lưu.



Hình 6: Mô hình MVC

2.4. Giới thiệu về Pusher

2.4.1. Khái niệm pusher

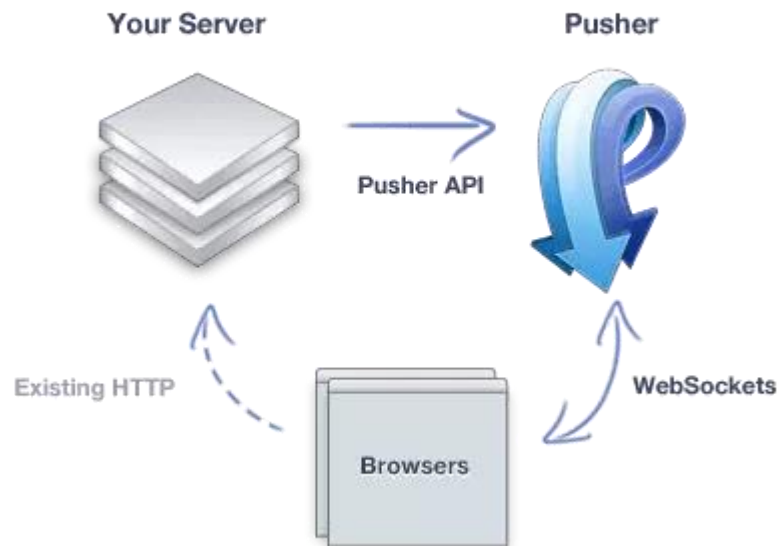
Pusher là một dịch vụ WebSocket được sử dụng để tạo và quản lý kết nối realtime giữa máy chủ và ứng dụng web hoặc di động. WebSocket là một giao thức cho phép truyền dữ liệu hai chiều giữa máy khách và máy chủ qua một kết nối TCP duy trì liên tục, điều này cho phép truyền dữ liệu mà không cần phải làm mới trang hoặc gửi các yêu cầu HTTP mới.

Pusher cung cấp các thư viện và API dễ sử dụng để tích hợp chức năng realtime vào ứng dụng của bạn một cách dễ dàng. Bằng cách sử dụng Pusher, bạn có thể gửi và nhận thông điệp realtime giữa máy khách và máy chủ, như thông báo, cập nhật dữ liệu realtime, trò chơi đa người chơi và nhiều ứng dụng khác.

Pusher cung cấp một giao diện quản lý trực tuyến để theo dõi và quản lý các kết nối realtime, thông điệp, và các sự kiện xảy ra trong ứng dụng của bạn.

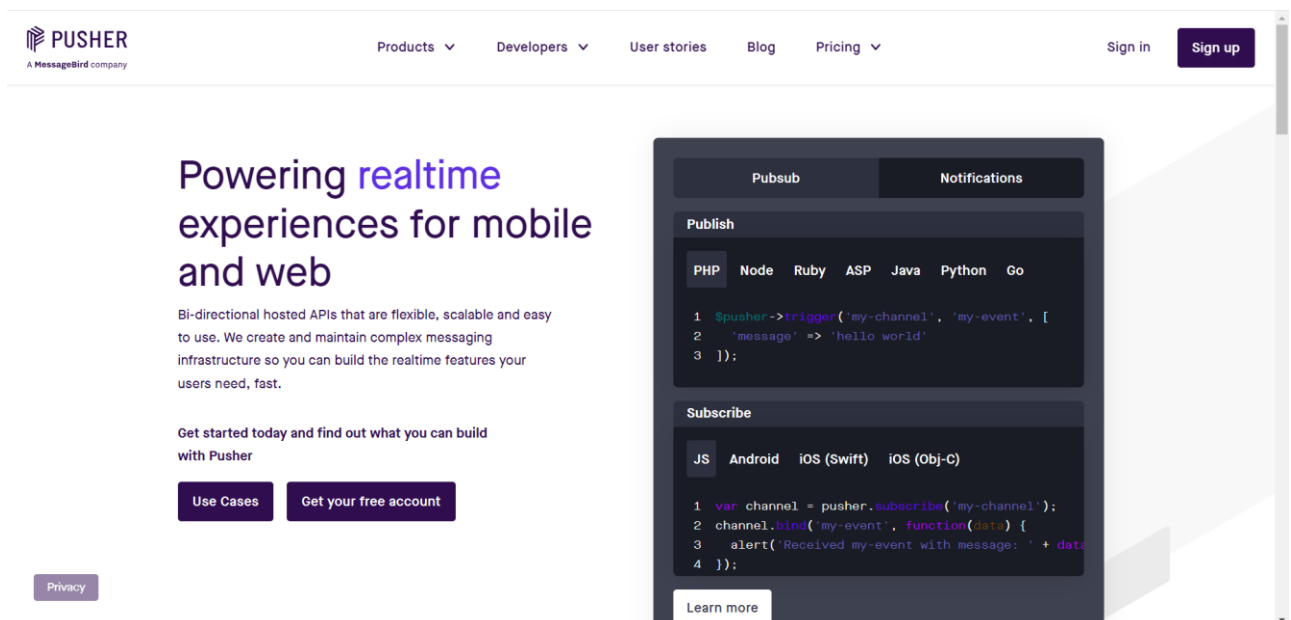
Cách thức hoạt động:

Understanding Pusher

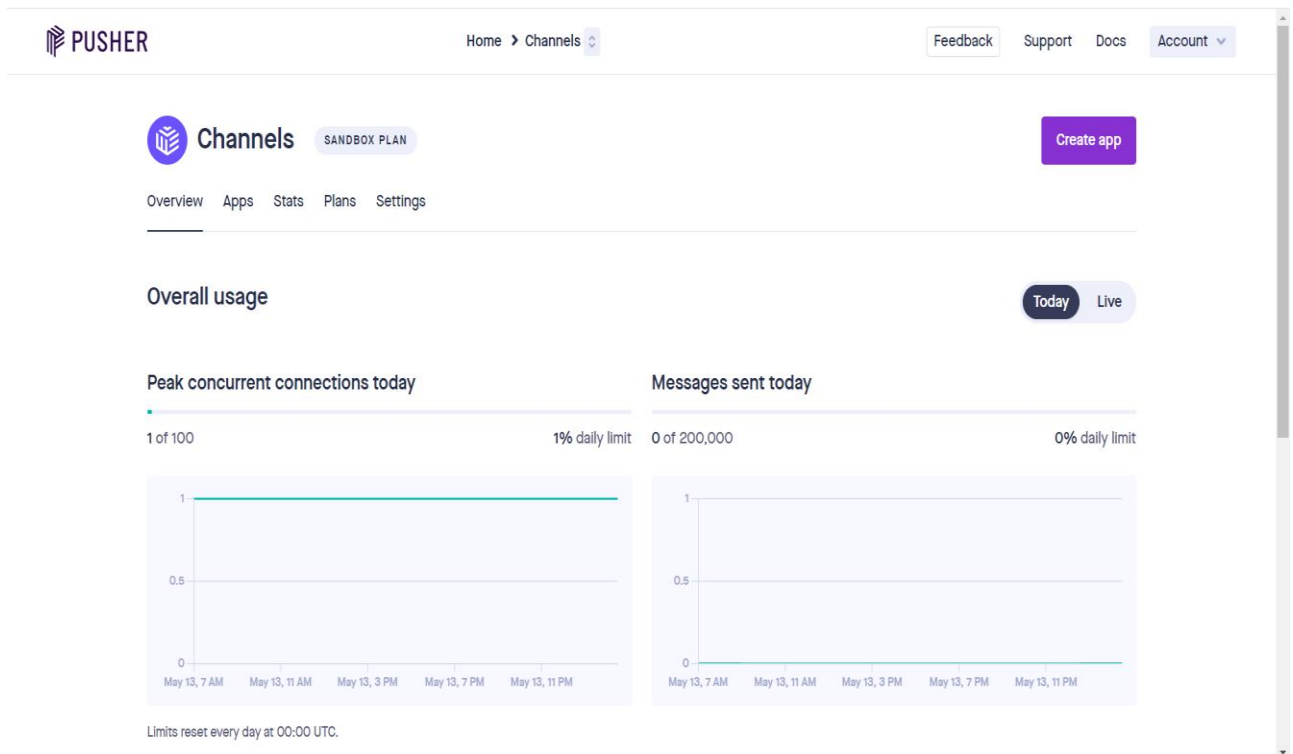


Hình 7: Cách thức hoạt động

Client duyệt web -> dữ liệu sẽ được chuyển đến server-> server chuyển tiếp đến Pusher thông qua pusher API -> Pusher trả kết lại cho client .



Hình 8: Giao diện web Pusher



Hình 9: Giao diện quản lý channels pusher

2.4.2. Ưu và nhược điểm của Pusher

Ưu điểm:

- Dễ tích hợp: Pusher cung cấp các thư viện và API dễ sử dụng cho nhiều ngôn ngữ lập trình và nền tảng, giúp tích hợp chức năng realtime vào ứng dụng một cách nhanh chóng và dễ dàng.
- Realtime: Pusher sử dụng giao thức WebSocket để tạo kết nối realtime giữa máy chủ và máy khách, cho phép truyền dữ liệu một cách nhanh chóng và hiệu quả mà không cần phải làm mới trang hoặc gửi các yêu cầu HTTP mới.
- Tích hợp dịch vụ đám mây: Pusher là một dịch vụ đám mây, do đó bạn không cần phải lo lắng về việc triển khai và duy trì cơ sở hạ tầng máy chủ của riêng mình.
- Quản lý kết nối và sự kiện: Pusher cung cấp một giao diện quản lý trực tuyến để theo dõi và quản lý các kết nối realtime, thông điệp, và sự kiện xảy ra trong ứng dụng của bạn.

Nhược điểm:

- Phụ thuộc vào dịch vụ bên ngoài: Việc sử dụng Pusher đồng nghĩa với việc phụ thuộc vào một dịch vụ bên ngoài, do đó bạn cần đảm bảo rằng dịch vụ này hoạt động ổn định và tin cậy.
- Chi phí: Pusher cung cấp các gói dịch vụ có phí, với mức phí tăng theo nhu cầu sử dụng và tính năng được sử dụng. Do đó, việc sử dụng Pusher có thể tạo ra chi phí đối với các ứng dụng có lượng truy cập lớn.
- Giới hạn về quyền kiểm soát: Khi sử dụng dịch vụ đám mây như Pusher, bạn có thể gặp phải giới hạn về quyền kiểm soát và tùy chỉnh so với việc triển khai và duy trì máy chủ của riêng mình.

2.4.3. Sử dụng với Laravel

Khi kết hợp sử dụng Laravel với Pusher, ta có thể tận dụng những lợi ích sau:

- Realtime và hiệu suất: Laravel cung cấp một cấu trúc mạnh mẽ để phát triển các ứng dụng web, trong khi Pusher giúp bạn tích hợp tính năng realtime một cách dễ dàng. Kết hợp cả hai giúp ứng dụng của bạn có thể gửi và nhận dữ liệu realtime một cách nhanh chóng và hiệu quả.
- Giảm độ trễ: Bằng cách sử dụng Pusher, bạn có thể giảm thiểu độ trễ trong việc cập nhật dữ liệu giữa máy chủ và máy khách. Điều này đặc biệt hữu ích trong các trường hợp cần cập nhật dữ liệu ngay lập tức như chat realtime, bảng xếp hạng trò chơi, hoặc hệ thống thông báo.
- Thiết kế dễ bảo trì: Sử dụng Laravel giúp bạn áp dụng mô hình MVC để tổ chức và quản lý mã nguồn. Kết hợp với Pusher, bạn có thể tách biệt logic ứng dụng và các tính năng realtime, giúp mã nguồn dễ bảo trì hơn.
- Phát triển nhanh chóng: Laravel cung cấp nhiều tính năng và công cụ giúp phát triển ứng dụng web một cách nhanh chóng và hiệu quả. Kết hợp với Pusher, bạn có thể thêm tính năng realtime vào ứng dụng mà không cần phải xây dựng từ đầu, giảm thời gian và công sức phát triển.

- Cộng đồng hỗ trợ lớn: Laravel và Pusher đều có cộng đồng lớn và tích cực, với nhiều tài liệu, hướng dẫn và gói mở rộng (packages) sẵn có. Điều này giúp bạn dễ dàng tìm kiếm giải pháp cho các vấn đề cụ thể trong quá trình phát triển.

Như vậy, ta đã đi qua 1 số lý thuyết cơ bản về các công nghệ chính sẽ được sử dụng để thiết kế trang web nghe nhạc.

Chương 3: Phân tích thiết kế hệ thống

3.1. Khảo sát thực tế

Mục tiêu của khảo sát thực tế là thu thập thông tin từ người dùng tiềm năng để hiểu rõ nhu cầu, thói quen và mong muốn của họ đối với một trang web nghe nhạc. Từ đó, có thể đưa ra các chức năng cần thiết nhằm tạo ra một sản phẩm đáp ứng tốt nhất kỳ vọng của người dùng.

Sử dụng hai phương pháp chính để thu thập dữ liệu:

- Khảo sát trực tuyến: Sử dụng các trang mạng xã hội và các ứng dụng nghe nhạc nổi tiếng như zing mp3, spotify; theo dõi các bình luận, nhận xét của người dùng về trang web từ đó đưa ra những ưu, nhược điểm của các website này để thiết kế các chức năng cần thiết cho website.
- Phỏng vấn trực tiếp: tiến hành hỏi trực tiếp với một số người dùng để thu thập thông tin chi tiết và sâu hơn về thói quen và nhu cầu của họ.

Kết quả khảo sát:

- Thói Quen Nghe Nhạc
 - Thời gian nghe nhạc: Phần lớn người dùng nghe nhạc từ 1-2 giờ mỗi ngày.
 - Thiết bị sử dụng: Điện thoại di động là thiết bị phổ biến nhất để nghe nhạc, tiếp theo là máy tính và máy tính bảng.
 - Thời gian nghe nhạc: Người dùng thường nghe nhạc vào buổi tối và đêm khuya.
- Sử Dụng Các Nền Tảng Nghe Nhạc
 - Nền tảng phổ biến: Spotify và Apple Music là hai nền tảng được sử dụng nhiều nhất, tiếp theo là Zing MP3 và Nhaccuatui.
 - Điểm thích: Người dùng đánh giá cao tính năng đề xuất nhạc theo sở thích và danh sách phát cá nhân của các nền tảng này.
 - Điểm không thích: Quảng cáo nhiều và giao diện phức tạp là hai yếu tố khiến người dùng không hài lòng.
- Tính Năng Quan Trọng

- Tính năng ưu tiên: Dựa trên kết quả khảo sát, các tính năng được người dùng đánh giá là quan trọng nhất bao gồm: phát nhạc liên tục, đề xuất nhạc theo trending, danh sách phát cá nhân
- Giao diện người dùng: Người dùng thích giao diện đơn giản, dễ dùng và hiện đại.

Kết Luận và Đề Xuất Chức Năng

- Phát nhạc liên tục: Đảm bảo trải nghiệm nghe nhạc liền mạch, không bị gián đoạn.
- Đề xuất nhạc theo trending.
- Danh sách phát cá nhân: Cho phép người dùng tạo và quản lý danh sách phát riêng của họ.
- Tìm kiếm nâng cao: Cải thiện tính năng tìm kiếm để người dùng có thể tìm thấy bài hát.
- Album cá nhân: Cho phép người dùng tạo và quản lý album của họ.
- Tương tác cá nhân: Cho phép người dùng like, bình luận các bài hát.

3.2. Đặc tả yêu cầu phần mềm

3.2.1. Yêu cầu chức năng

Quản lý người dùng

- Đăng ký tài khoản:
 - Người dùng có thể đăng ký tài khoản mới bằng email và mật khẩu.
- Đăng nhập và Đăng xuất:
 - Người dùng có thể đăng nhập bằng email/mật khẩu hoặc tài khoản mạng xã hội.
 - Người dùng có thể đăng xuất khỏi hệ thống.
- Quên mật khẩu:
 - Người dùng có thể yêu cầu đặt lại mật khẩu qua email.

Tìm kiếm và Khám phá nhạc

- Tìm kiếm: Người dùng có thể tìm kiếm bài hát.
- Khám phá: Hiển thị các bài hát, nghệ sĩ, album nổi bật và mới nhất.

Nghe nhạc

- Phát nhạc:
 - Người dùng có thể phát nhạc trực tuyến với chất lượng cao.
 - Hỗ trợ các chế độ phát lại (lặp lại, phát ngẫu nhiên).
- Danh sách phát:
 - Người dùng có thể tạo, chỉnh sửa, và xóa danh sách phát.
 - Người dùng có thể thêm bài hát vào danh sách phát cá nhân.

Tương tác xã hội

- Người dùng có thể bình luận, đánh giá và yêu thích bài hát.

3.2.2. Yêu cầu phi chức năng

- Hiệu suất
 - Trang web phải tải nhanh và có thời gian phản hồi dưới 2 giây cho các yêu cầu chính.
- Bảo mật
 - Bảo vệ thông tin cá nhân của người dùng bằng cách sử dụng mã hóa.
- Khả năng mở rộng
 - Hệ thống phải dễ dàng mở rộng để hỗ trợ số lượng lớn người dùng và dữ liệu.
- Tính tương thích
 - Trang web phải tương thích với các trình duyệt phổ biến (Chrome, Firefox, Safari, Edge).
 - Hỗ trợ cả phiên bản desktop và mobile.

3.2.3. Các yêu cầu hệ thống khác

- Yêu cầu về phần cứng
 - Máy chủ phải có cấu hình đủ mạnh để đáp ứng yêu cầu xử lý và lưu trữ.
- Yêu cầu về phần mềm
 - Sử dụng các công nghệ web hiện đại như HTML5, CSS3, JavaScript, và các framework như NuxtJS.
 - Backend sử dụng Laravel API.

- Cơ sở dữ liệu có thể sử dụng MySQL.

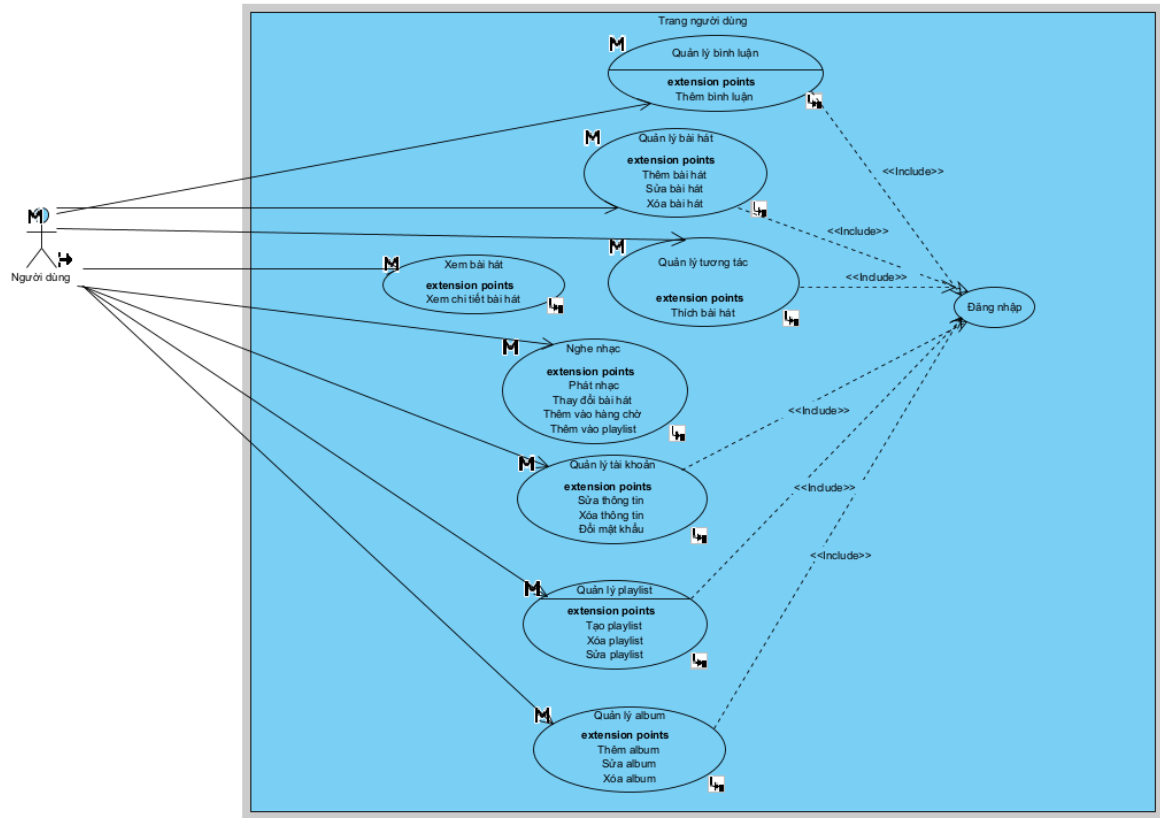
3.3. Phân tích thiết kế hệ thống sử dụng UML

3.3.1. Biểu đồ ca sử dụng

Một trang dành cho người mua hàng và một trang dành cho người quản trị. Tác nhân tham gia vào trang người dùng là khách hàng khi thực hiện vào website xem và nghe nhạc. Tác nhân tham gia vào trang quản lý thông tin của website và thống kê gồm 1 tác nhân là quản trị viên.

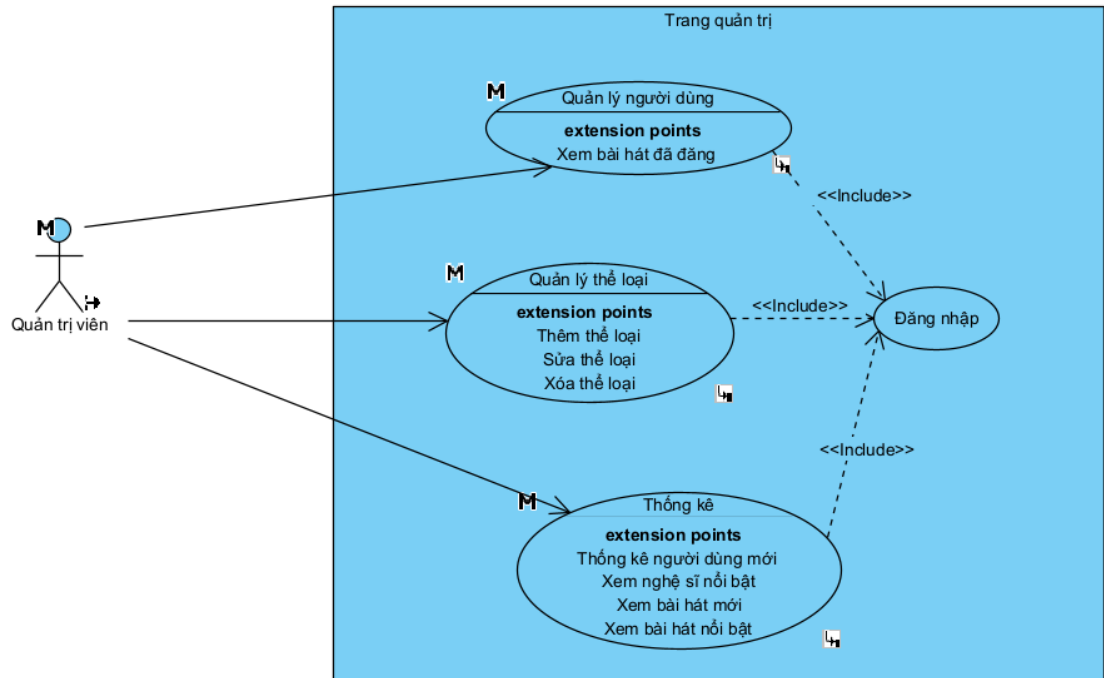
Trong biểu đồ tổng quát người dùng, để thực hiện các chức năng như thích, bình luận, tạo về thêm nhạc vào playlist, người dùng đều phải thực hiện đăng nhập. Vì vậy các ca sử dụng này có mối quan hệ <<include>> với ca sử dụng *Đăng nhập*.

Trong trang quản trị, để thực hiện tất cả các use case, quản trị viên đều phải thực hiện đăng nhập trước khi sử dụng. Vì vậy hoạt động đăng nhập được tách riêng thành một ca sử dụng và các ca sử dụng khác trong hệ thống có mối quan hệ <<include>> với ca sử dụng *Đăng nhập*.



Hình 10: Biểu đồ tổng quát trang người dùng

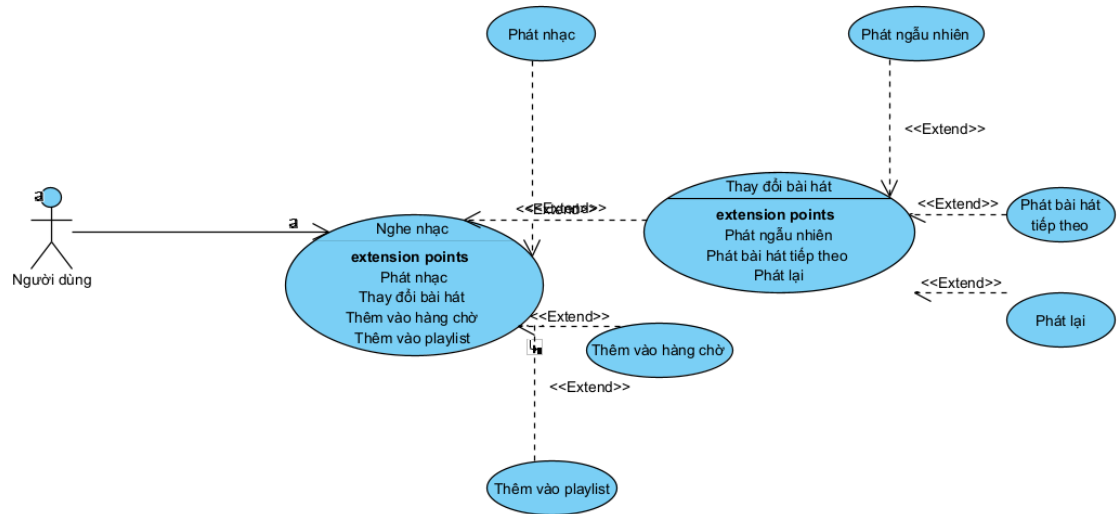
Biểu đồ tổng quát trang người dùng gồm các chức năng chính như sau: quản lý loại bình luận, quản lý bài hát, quản lý playlist, quản lý tài khoản, quản lý album, tương tác, bình luận.



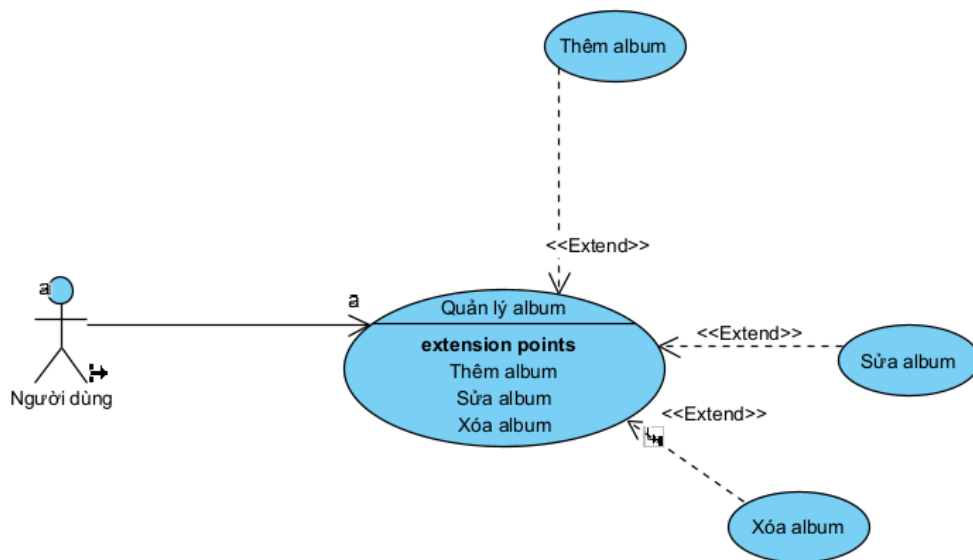
Hình 11: Biểu đồ tổng quát trang admin

Biểu đồ tổng quát trang quản trị gồm các chức năng chính như sau: quản lý người dùng, quản lý thể loại, thống kê. Tất cả các chức năng này đều yêu cầu đăng nhập.

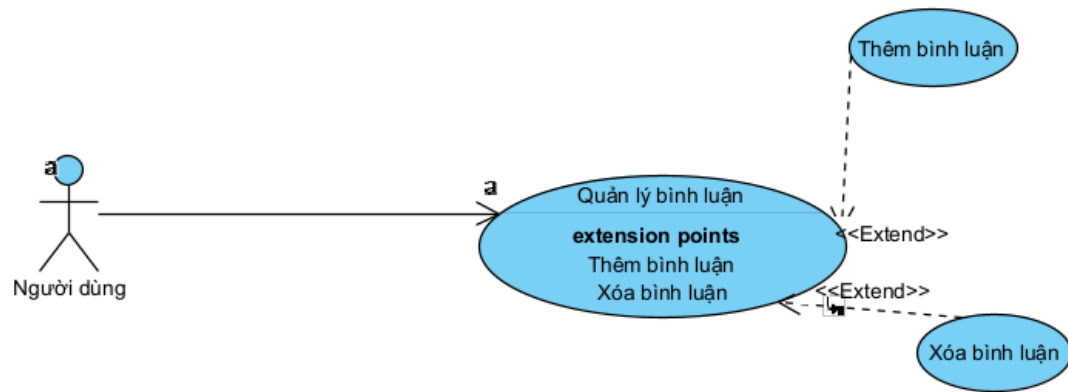
Sau khi xây dựng biểu đồ ca sử dụng mức tổng quát, tiếp tục xây dựng các biểu đồ ca sử dụng phân rã.



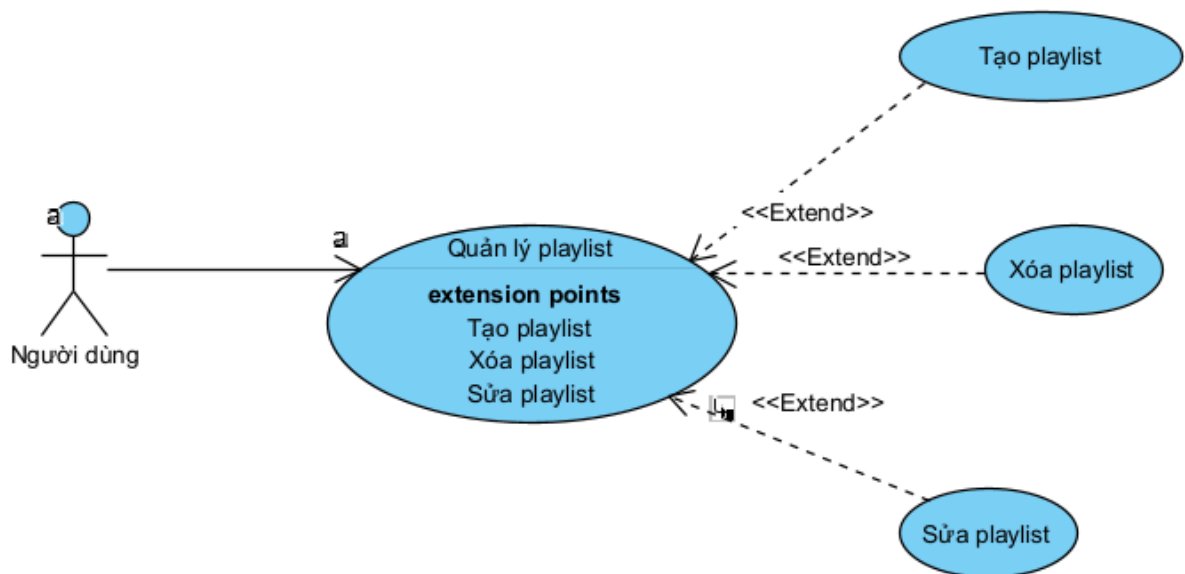
Hình 12: Biểu đồ phân rã của use case nghe nhạc



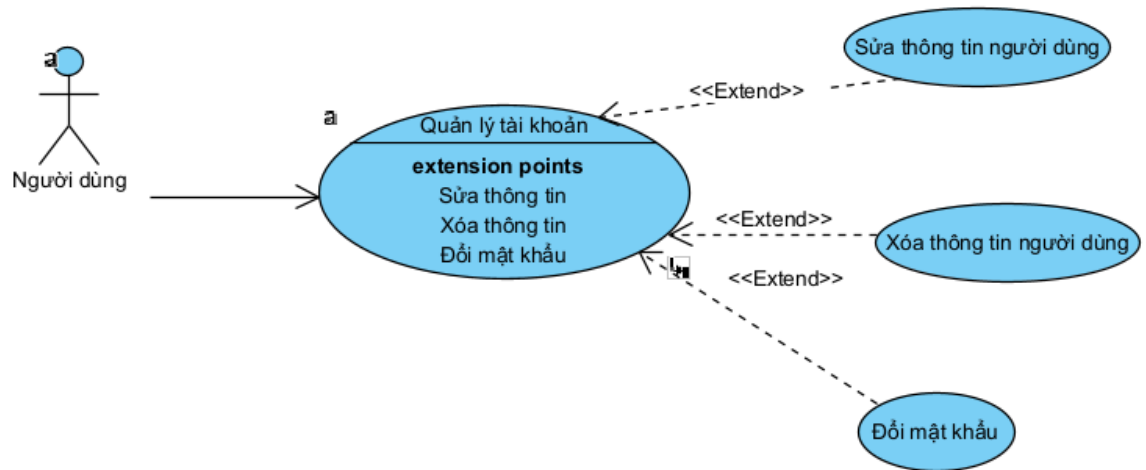
Hình 13: Biểu đồ phân rã của use case Quản lý Album



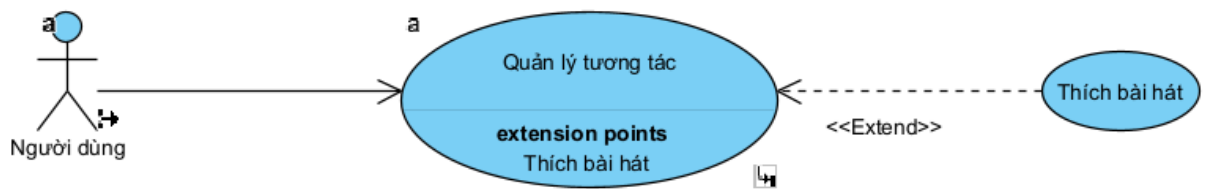
Hình 14: Biểu đồ phân rã của use case Quản lý bình luận



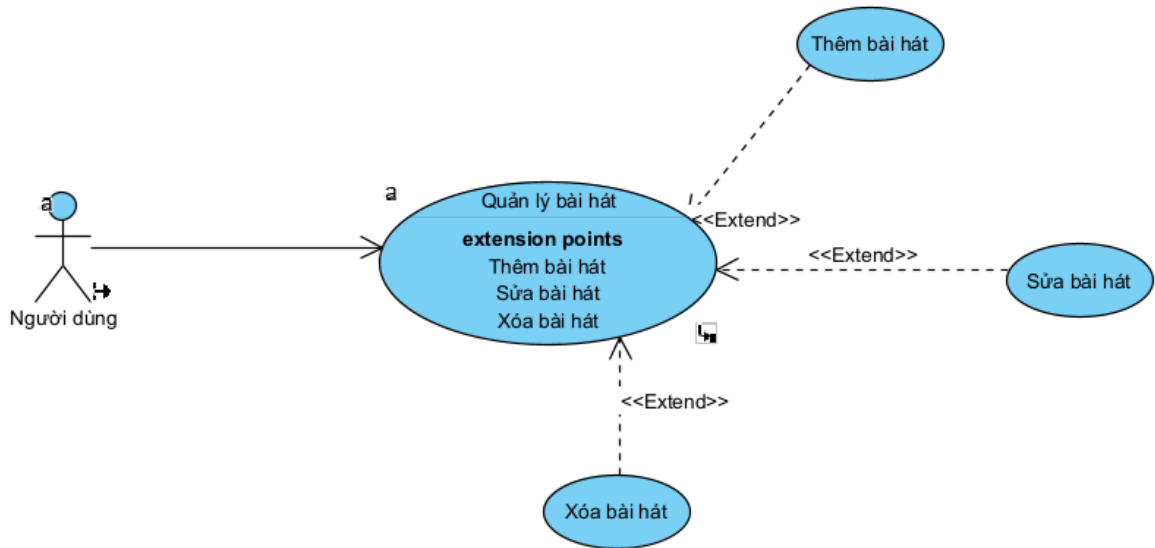
Hình 15: Biểu đồ phân rã của use case Quản lý Playlist



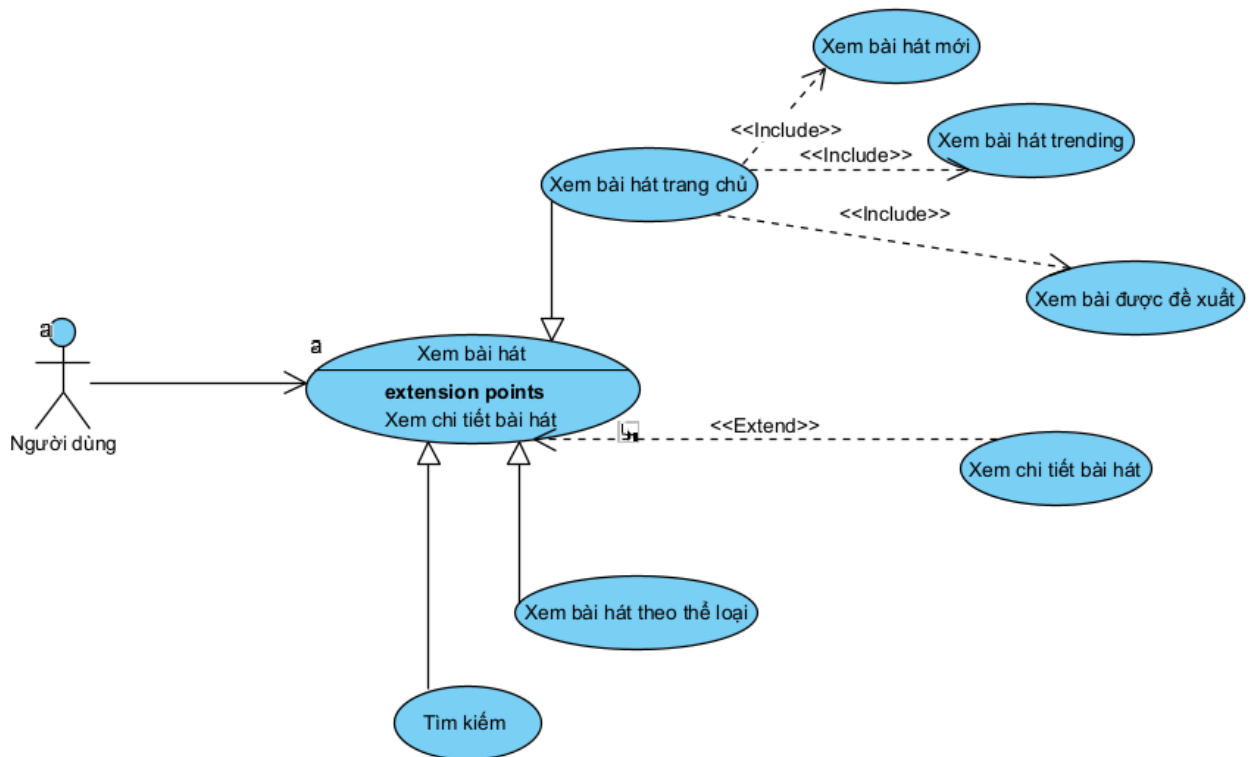
Hình 15: Biểu đồ phân rã của use case quản lý tài khoản



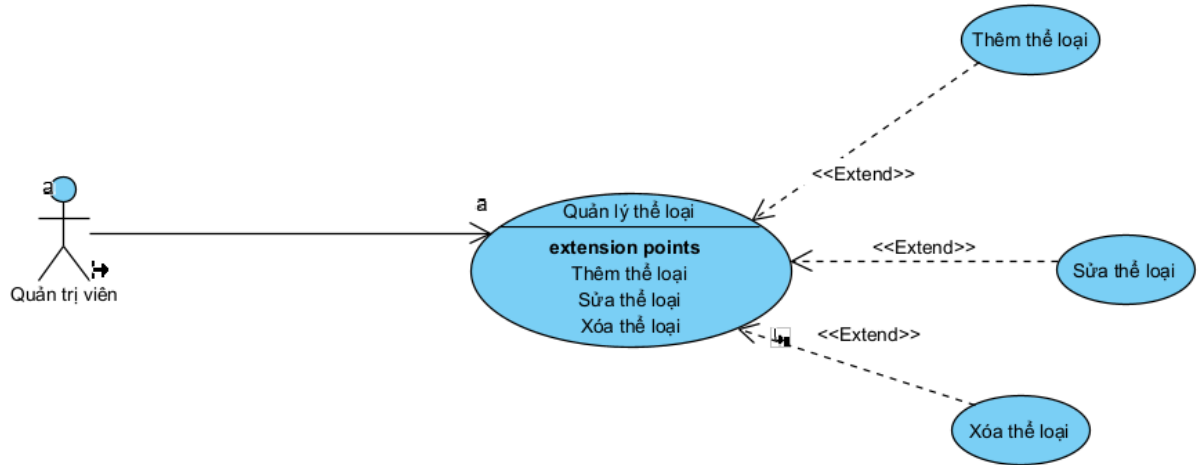
Hình 16: Biểu đồ phân rã của use case Quản lý tương tác



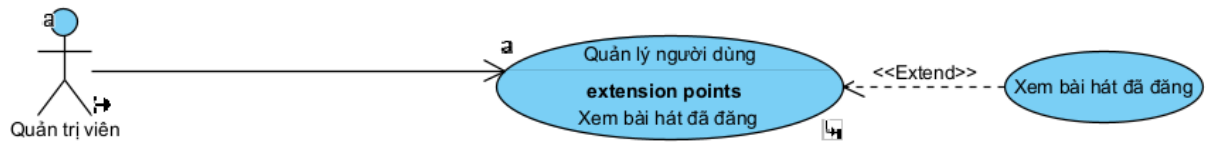
Hình 17: Biểu đồ phân rã của use case quản lý bài hát



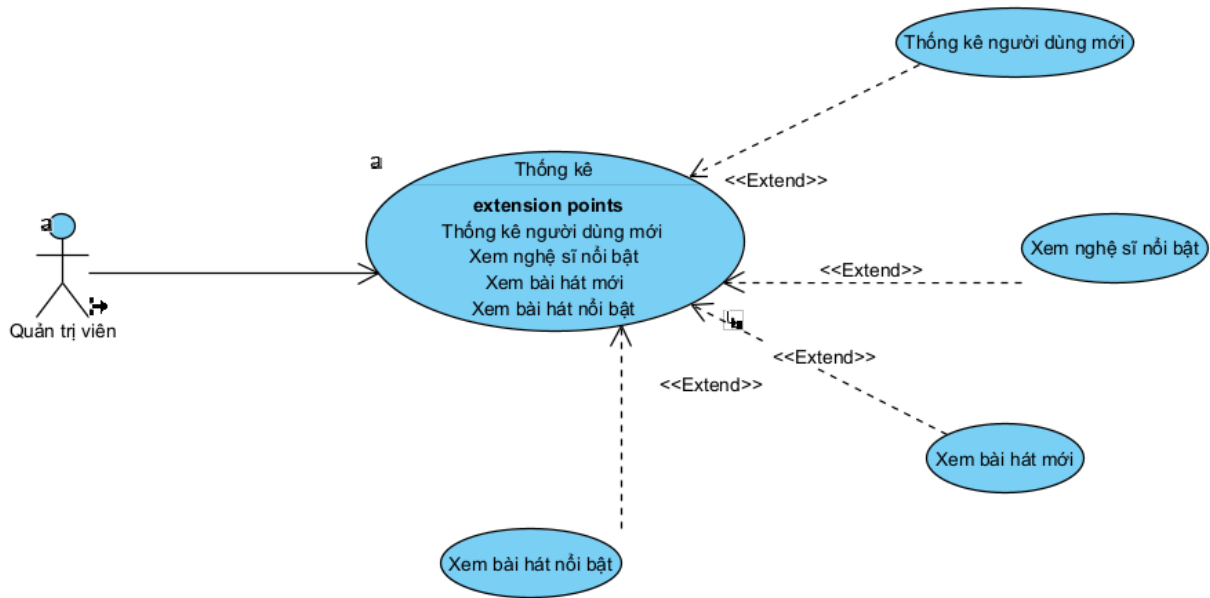
Hình 18: Biểu đồ phân rã của use case Xem bài hát



Hình 19: Biểu đồ phân rã của use case Quản lý thể loại



Hình 20: Biểu đồ phân rã của use case Quản lý người dùng



Hình 21: Biểu đồ phân rã của use case thống kê

3.3.2. Kịch bản các ca sử dụng

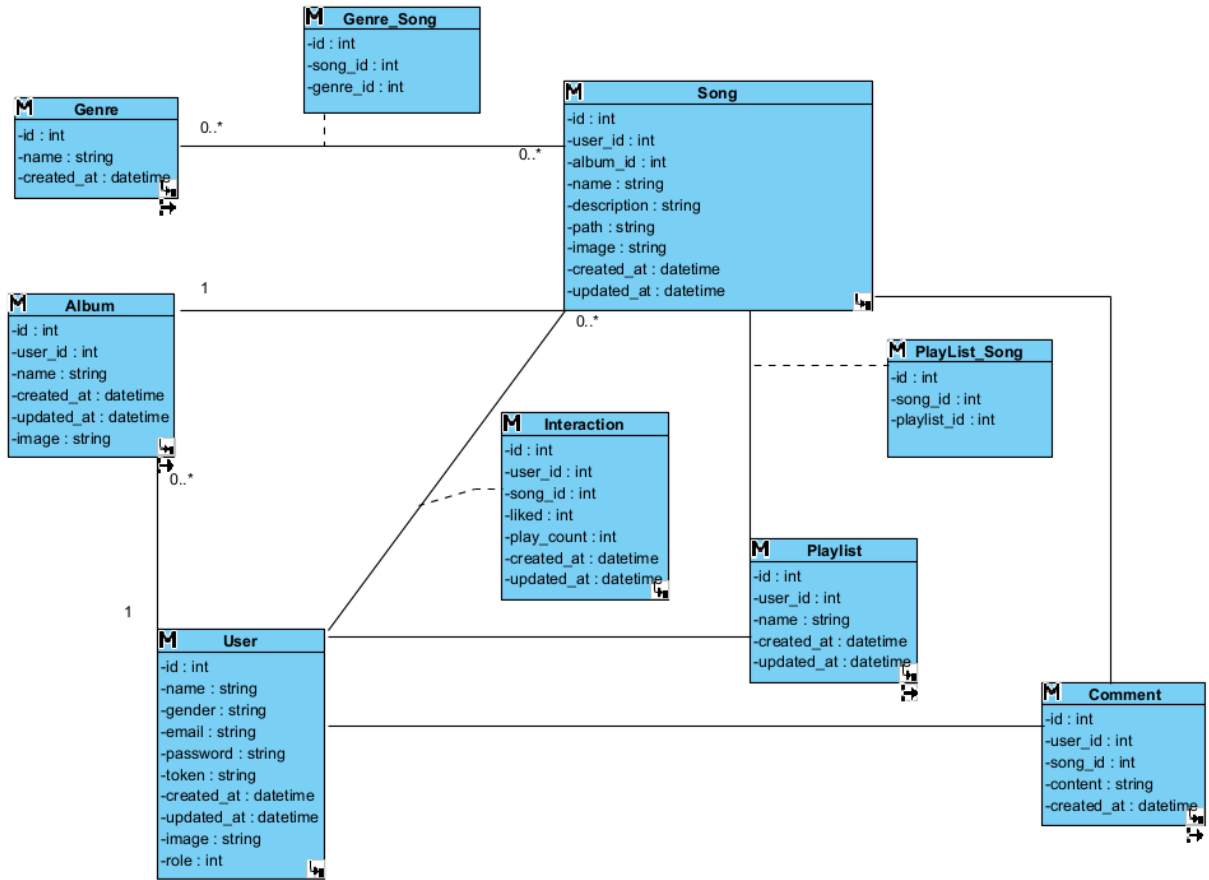
- Chức năng đăng nhập
 - Người dùng yêu cầu chức năng đăng nhập.
 - Hệ thống hiển thị giao diện đăng nhập.
 - Người dùng nhập tài khoản, mật khẩu rồi nhấn đăng nhập.
 - Hệ thống kiểm tra tài khoản và mật khẩu, nếu không đúng thì hiển thị sai thông tin và yêu cầu đăng nhập lại. Nếu đúng thì chuyển về trang sử dụng của người dùng.
- Chức năng quản lý thể loại nhạc
 - Quản trị viên yêu cầu chức năng quản lý loại nhạc.
 - Hệ thống chuyển sang trang quản lý, hiển thị các loại nhạc lên giao diện.
 - Tại đây quản trị viên có thể lựa chọn các ca sử dụng như thêm, sửa, xóa.
- Chức năng thêm loại sản phẩm
 - Quản trị viên yêu cầu chức năng thêm loại nhạc.
 - Hệ thống hiển thị giao diện thêm loại nhạc.
 - Quản trị viên nhập thông tin loại sản phẩm và yêu cầu thêm mới loại sản phẩm.

- Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu thông tin sai thì thông báo lỗi và nhập lại. Nếu hợp lệ thì thêm loại nhạc và quay về trang quản lý.
- Chức năng sửa thông tin loại nhạc
 - Admin yêu cầu chức năng sửa thông tin loại nhạc.
 - Hệ thống hiển thị giao diện sửa và hiển thị thông tin loại nhạc cần sửa.
 - Quản trị viên nhập thông tin muốn sửa loại sản phẩm và yêu cầu sửa.
 - Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu thông tin sai thì thông báo lỗi và nhập lại. Nếu hợp lệ thì lưu thông tin mới của loại nhạc và quay về trang quản lý.
- Chức năng quản lý người dùng
 - Quản trị viên yêu cầu chức năng quản lý người dùng.
 - Hệ thống hiển thị danh sách người dùng.
 - Tại đây quản trị viên có thể lựa chọn các ca sử dụng xem danh sách bài hát của người dùng.
- Chức năng quản lý bài hát
 - Người dùng yêu cầu chức năng quản lý bài hát.
 - Hệ thống hiển thị thông tin các bài hát.
 - Tại đây người dùng có thể lựa chọn các ca sử dụng như thêm, sửa, xóa
- Chức năng thêm bài hát
 - Người dùng yêu cầu chức năng thêm bài hát.
 - Hệ thống hiển thị giao diện thêm.
 - Người dùng nhập thông tin thêm và yêu cầu thêm.
 - Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu thông tin sai thì thông báo lỗi và nhập lại. Nếu hợp lệ thì lưu thông tin mới của bài hát và quay về trang quản lý.
- Chức năng sửa thông tin bài hát
 - Người dùng yêu cầu chức năng sửa thông tin bài hát.
 - Hệ thống hiển thị giao diện sửa và hiển thị thông tin bài hát cần sửa.
 - Người dùng nhập thông tin thêm và yêu cầu sửa.

- Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu thông tin sai thì thông báo lỗi và nhập lại. Nếu hợp lệ thì lưu thông tin mới của bài hát và quay về trang quản lý.
- Chức năng thêm playlist
 - Người dùng yêu cầu chức năng thêm playlist.
 - Hệ thống hiển thị giao diện thêm.
 - Người dùng nhập thông tin thêm và yêu cầu thêm.
 - Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu thông tin sai thì thông báo lỗi và nhập lại. Nếu hợp lệ thì lưu thông tin mới của playlist và quay về trang quản lý.
- Chức năng sửa thông tin playlist
 - Người dùng yêu cầu chức năng sửa thông tin playlist.
 - Hệ thống hiển thị giao diện sửa và hiển thị thông tin playlist cần sửa.
 - Người dùng nhập thông tin thêm và yêu cầu sửa.
 - Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu thông tin sai thì thông báo lỗi và nhập lại. Nếu hợp lệ thì lưu thông tin mới của playlist và quay về trang quản lý.
- Chức năng thêm album
 - Người dùng yêu cầu chức năng thêm album.
 - Hệ thống hiển thị giao diện thêm.
 - Người dùng nhập thông tin thêm và yêu cầu thêm.
 - Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu thông tin sai thì thông báo lỗi và nhập lại. Nếu hợp lệ thì lưu thông tin mới của album và quay về trang quản lý.
- Chức năng sửa thông tin album
 - Người dùng yêu cầu chức năng sửa thông tin album.
 - Hệ thống hiển thị giao diện sửa và hiển thị thông tin album cần sửa.
 - Người dùng nhập thông tin thêm và yêu cầu sửa.

- Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu thông tin sai thì thông báo lỗi và nhập lại. Nếu hợp lệ thì lưu thông tin mới của album và quay về trang quản lý.
- Chức năng bình luận
 - Người dùng chọn bài hát muốn bình luận và nhập thông tin bình luận vào form.
 - Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu hợp lệ thì lưu thông tin bình luận của người dùng vào hệ thống và quay về trang chi tiết bài hát.
- Chức năng thích bài hát
 - Người dùng chọn bài hát yêu thích.
 - Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu hợp lệ thì lưu thông tin đánh giá của người dùng vào hệ thống và quay về trang đánh giá.
- Chức năng quản lý tài khoản
 - Người dùng yêu cầu thực hiện chức năng quản lý thông tin tài khoản.
 - Hệ thống hiển thị giao diện thông tin tài khoản và người dùng có thể thực hiện chức năng cập nhật thông tin người dùng.
 - Hệ thống kiểm tra tính hợp lệ của thông tin. Nếu hợp lệ thì lưu thông tin tài khoản của người dùng vào hệ thống và quay về trang đánh giá.

3.3.3. Biểu đồ lớp thực thể

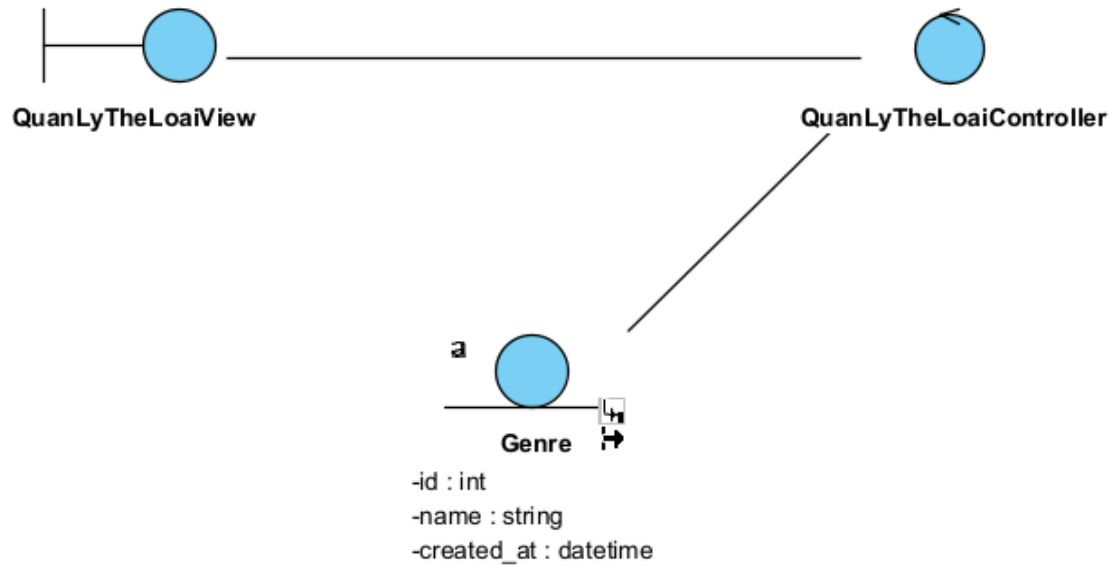


Hình 22: Biểu đồ lớp thực thể

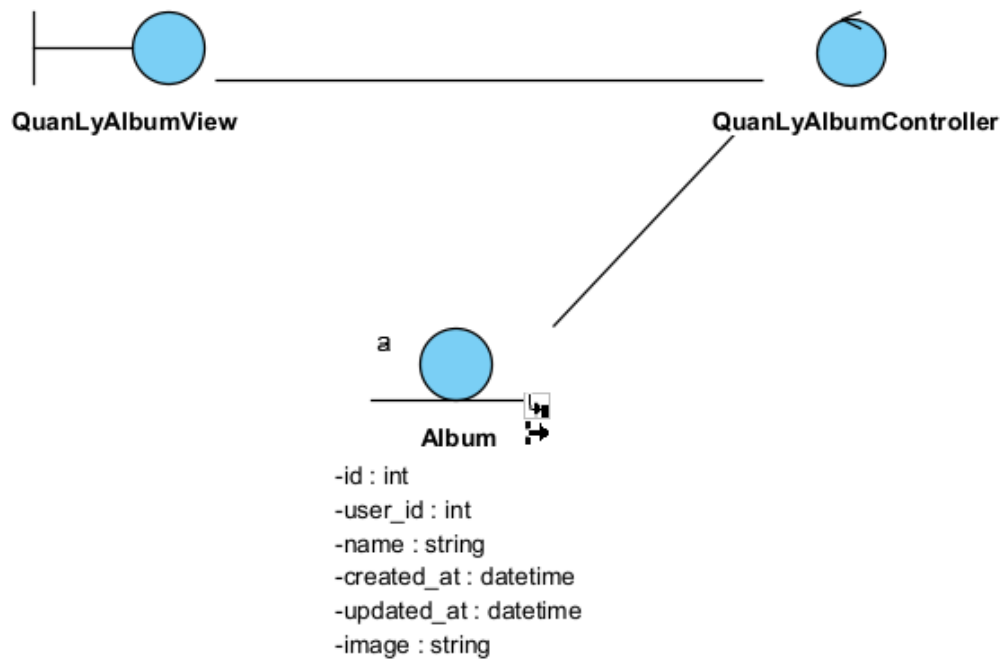
3.3.4. Biểu đồ lớp phân tích ca sử dụng

Từ kịch bản các luồng sự kiện trong mô tả chi tiết từng ca sử dụng, tiến hành phân tích để xác định các lớp tham gia thực hiện ca sử dụng và mô hình hóa bằng biểu đồ lớp VOPC cho từng ca sử dụng.

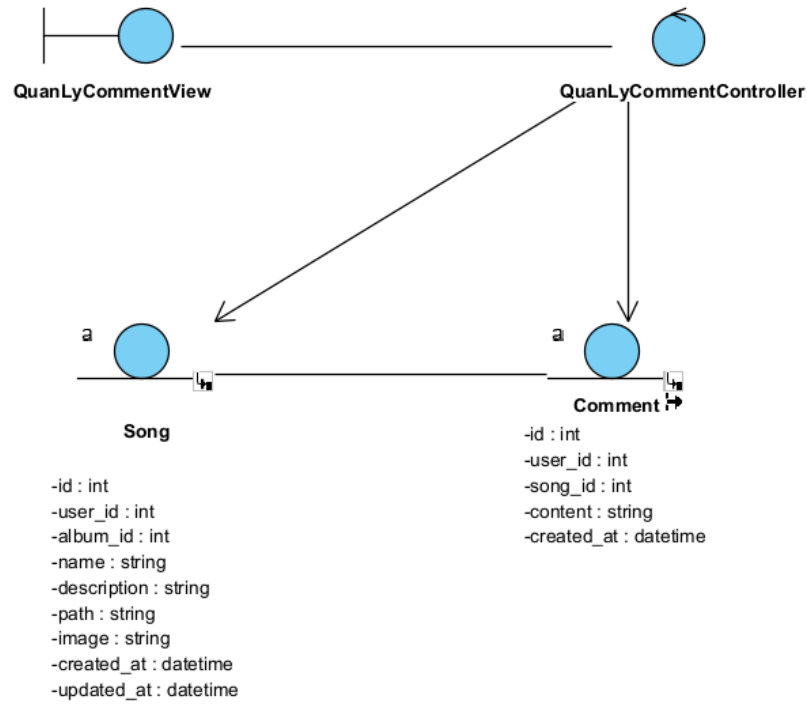
Dưới đây là các biểu đồ lớp phân tích của các ca sử dụng:



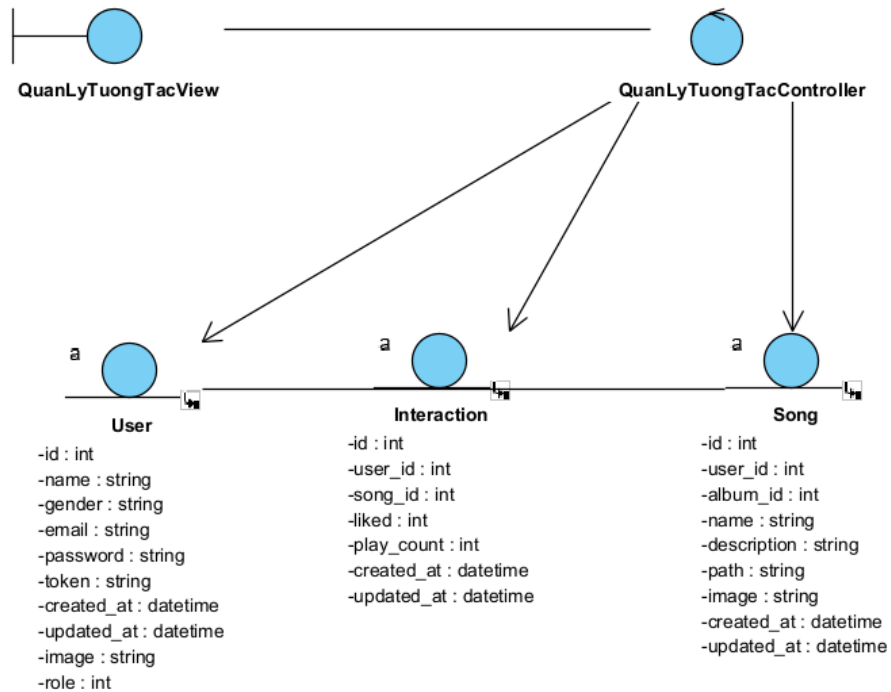
Hình 23: Biểu đồ lớp phân tích của use case quản lý thể loại



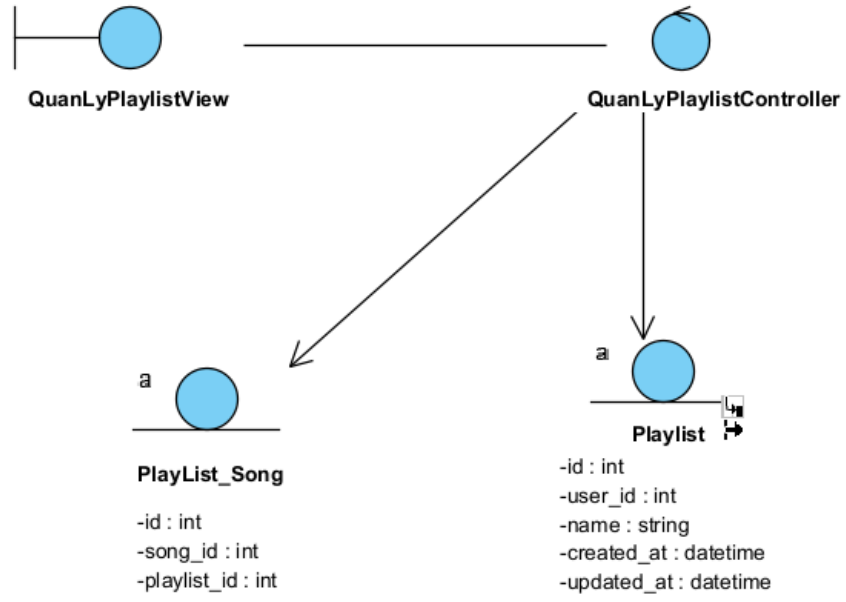
Hình 24: Biểu đồ lớp phân tích của use case quản lý album



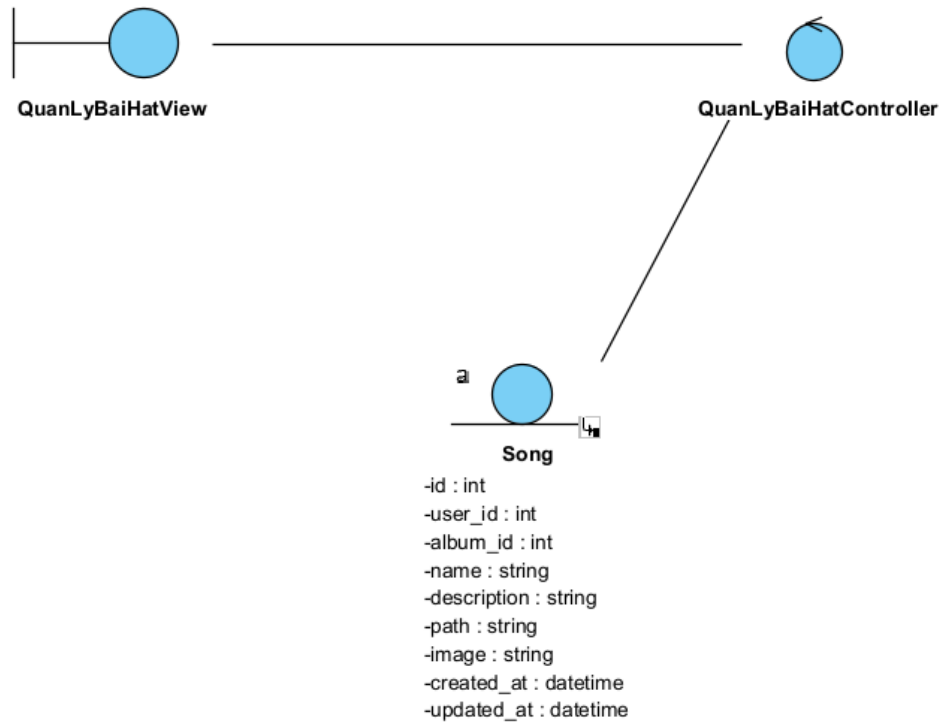
Hình 25: Biểu đồ lớp phân tích của use case quản lý comment



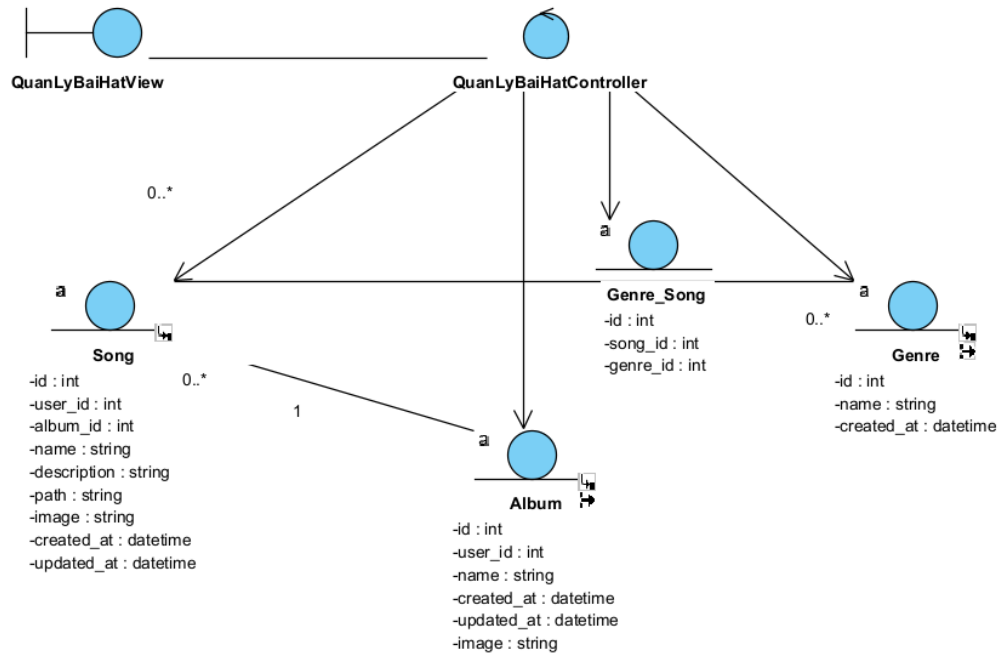
Hình 26: Biểu đồ lớp phân tích của use case quản lý tương tác



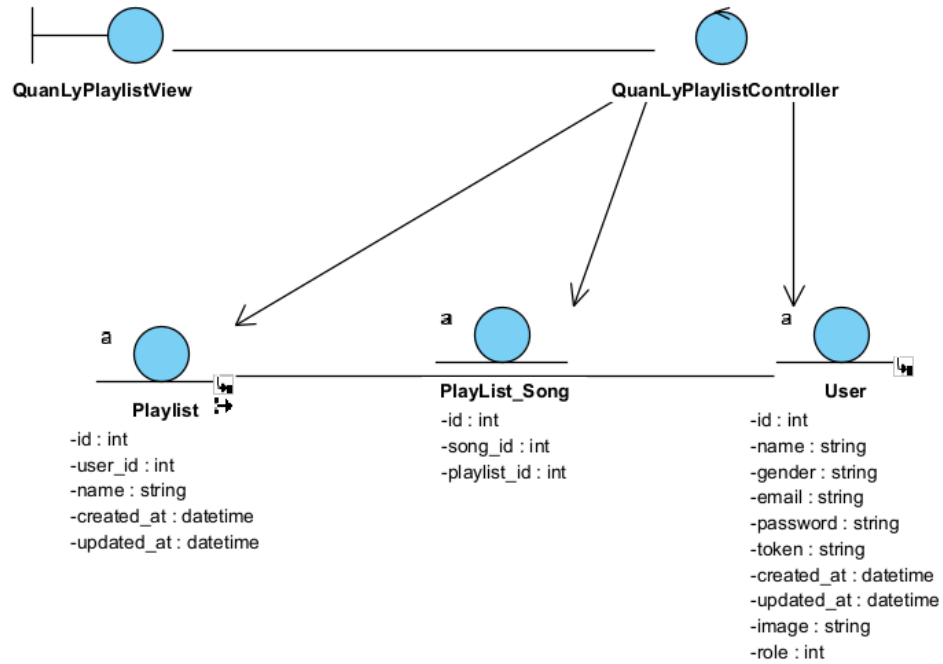
Hình 27: Biểu đồ lớp phân tích của use case quản lý playlist



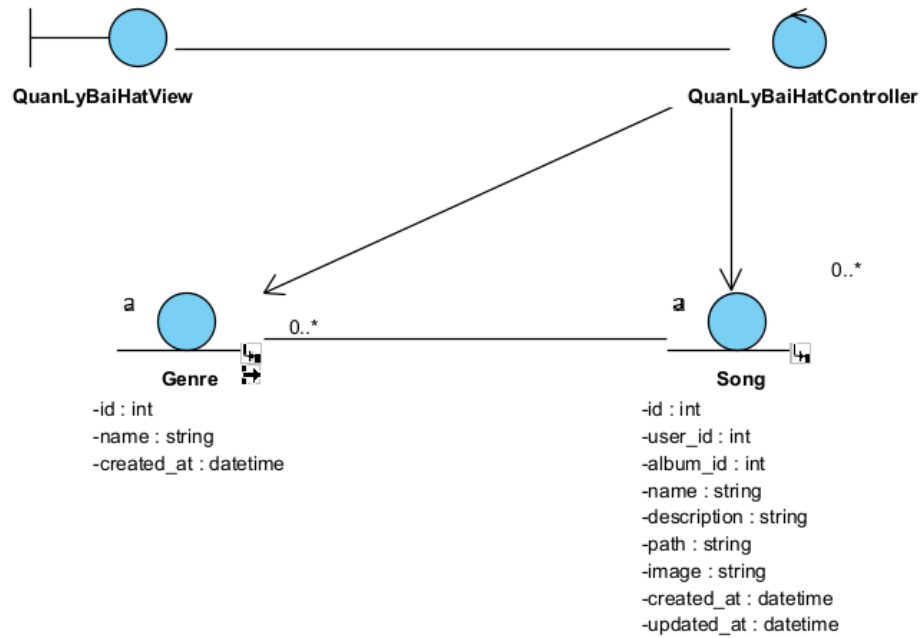
Hình 28: Biểu đồ lớp phân tích của use case quản lý bài hát



Hình 29: Biểu đồ lớp phân tích của use case thêm bài hát



Hình 30: Biểu đồ lớp phân tích của use case thêm bài hát vào playlist

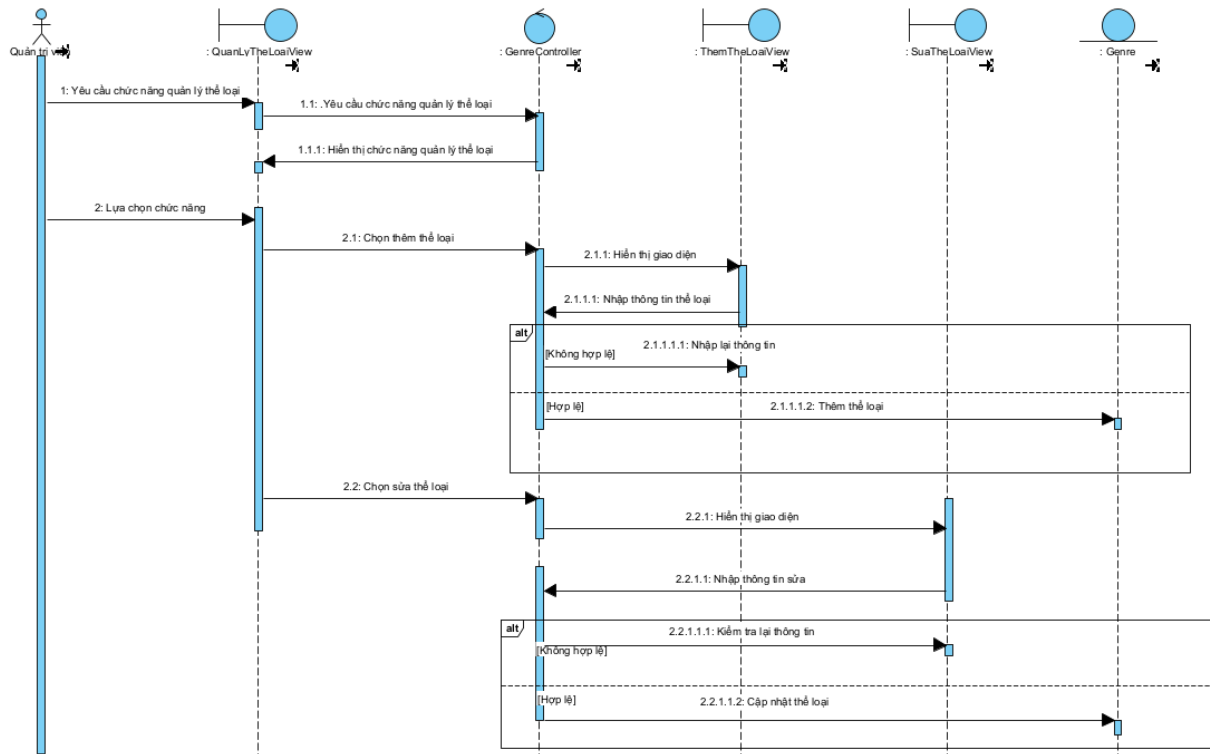


Hình 31: Biểu đồ lớp phân tích của use case lấy bài hát theo loại

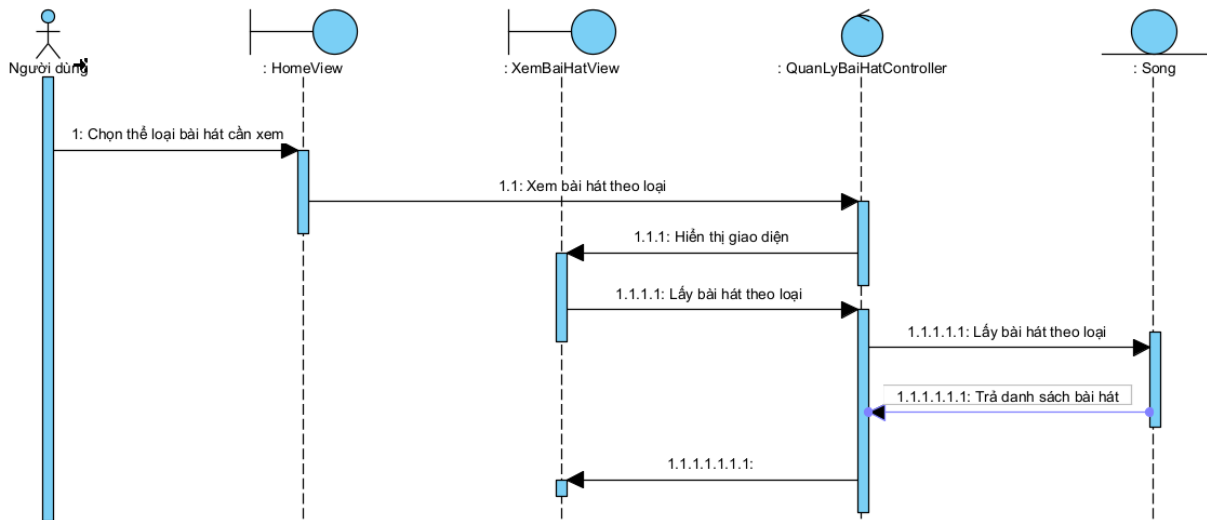
3.3.5. Biểu đồ tuần tự

Biểu đồ tuần tự sẽ được sử dụng để mô hình hóa các chuỗi tương tác giữa các đối tượng là thể hiện của các lớp tham gia thực hiện ca sử dụng.

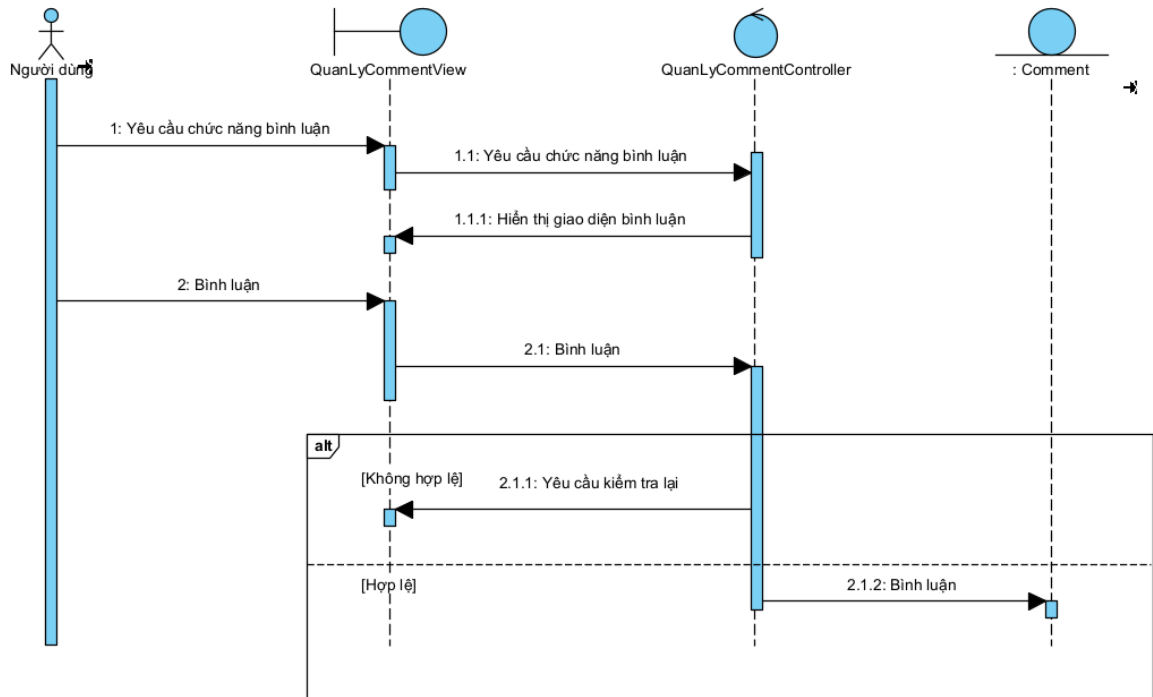
Dưới đây là các biểu đồ tuần tự:



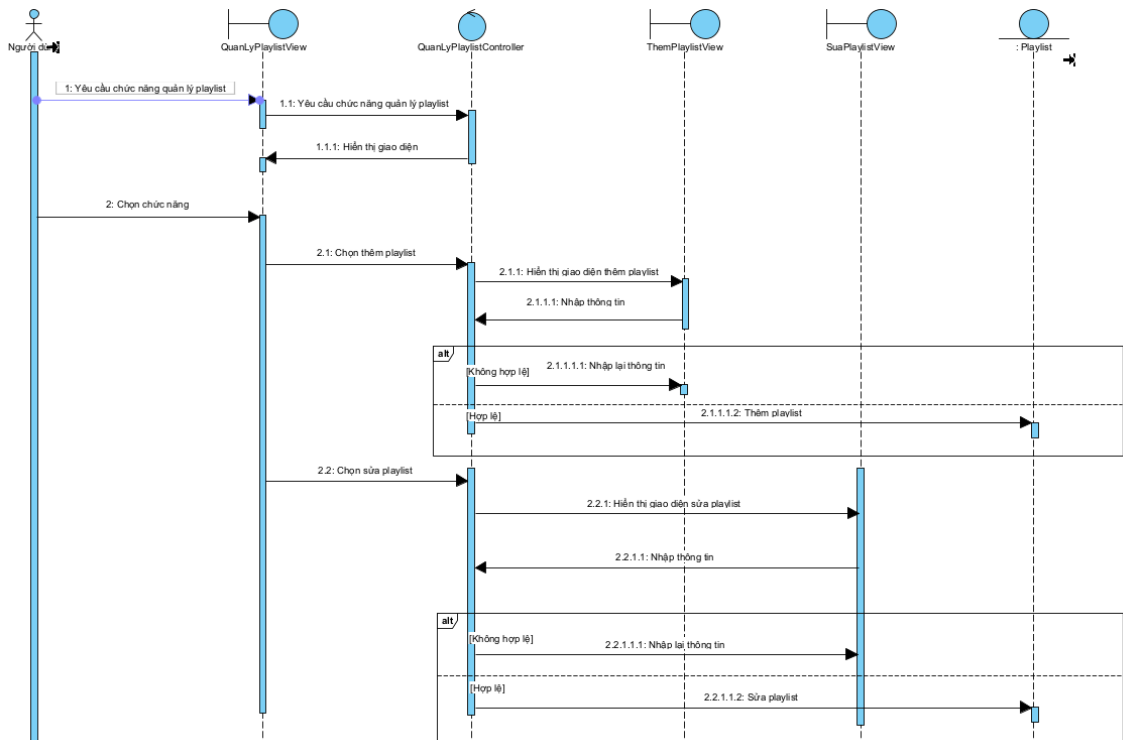
Hình 32: Biểu đồ tuần tự chức năng quản lý thể loại



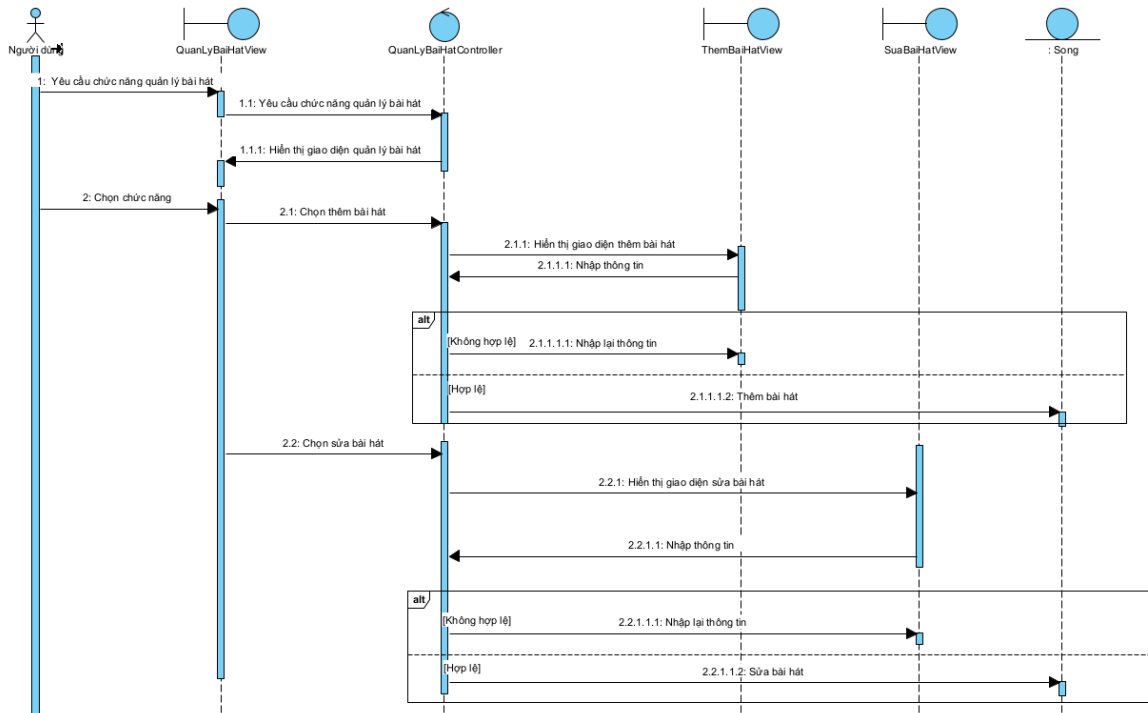
Hình 33: Biểu đồ tuần tự chức năng xem bài hát theo thể loại



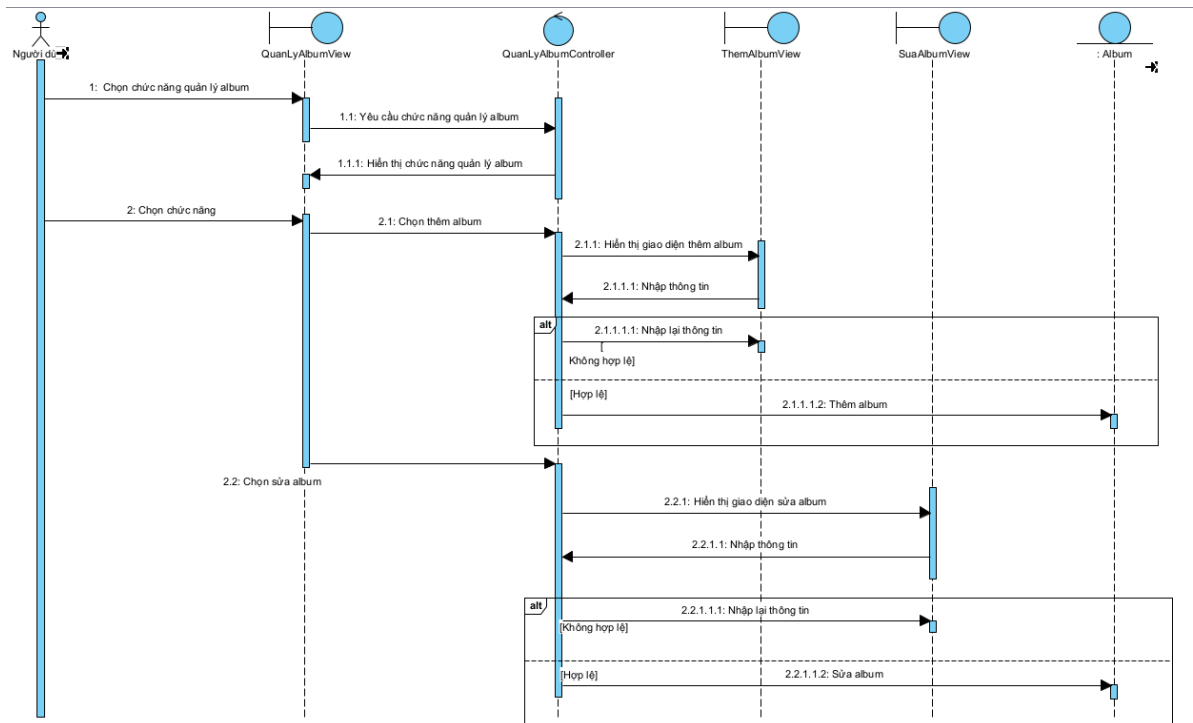
Hình 34: Biểu đồ tuần tự chức năng quản lý comment



Hình 35: Biểu đồ tuần tự chức năng quản lý playlist



Hình 36: Biểu đồ tuần tự chức năng quản lý bài hát

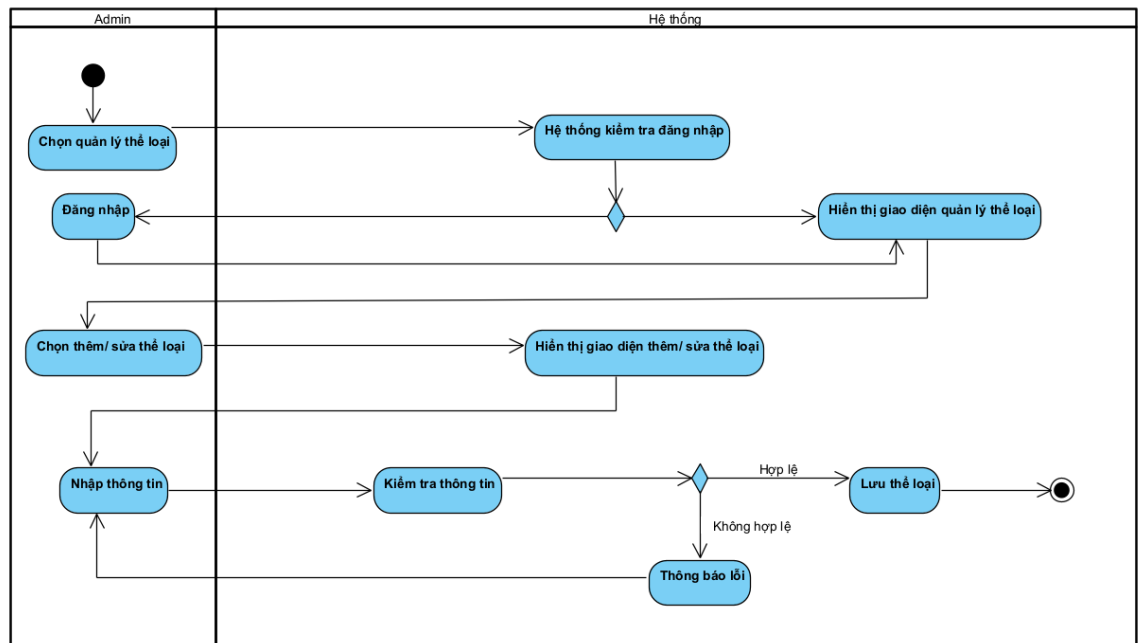


Hình 37: Biểu đồ tuần tự chức năng quản lý album

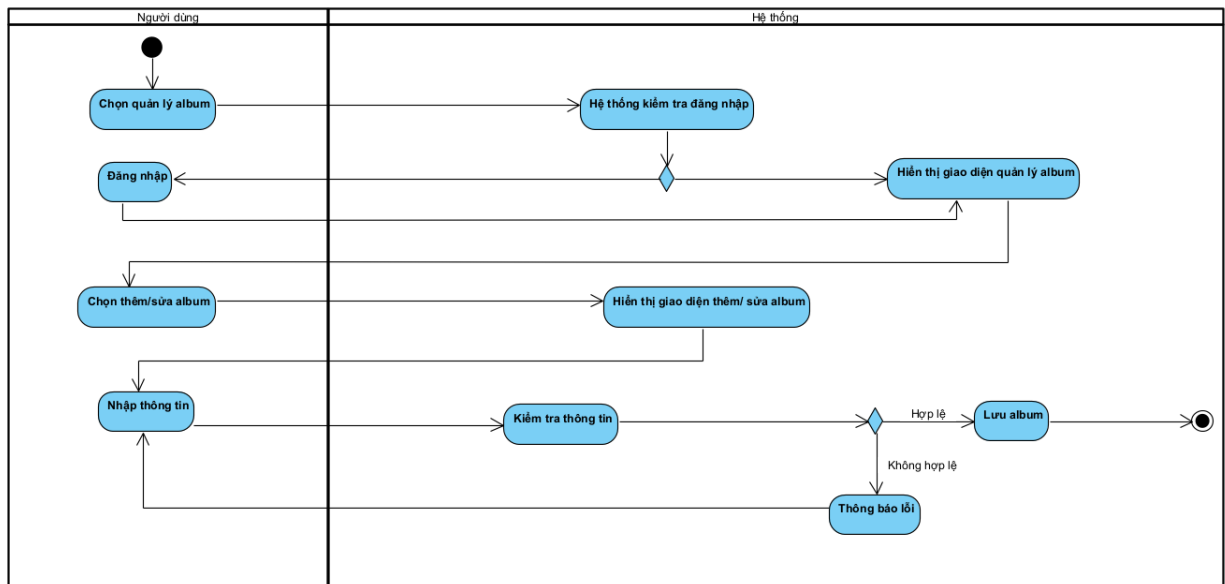
3.3.6. Biểu đồ hoạt động

Biểu đồ hoạt động là biểu đồ mô tả các bước thực hiện, các hành động, các nút quyết định và điều kiện rẽ nhánh để điều khiển luồng thực hiện của hệ thống. Đối với những luồng thực thi có nhiều tiến trình chạy song song thì biểu đồ hoạt động là sự lựa chọn tối ưu cho việc thể hiện.

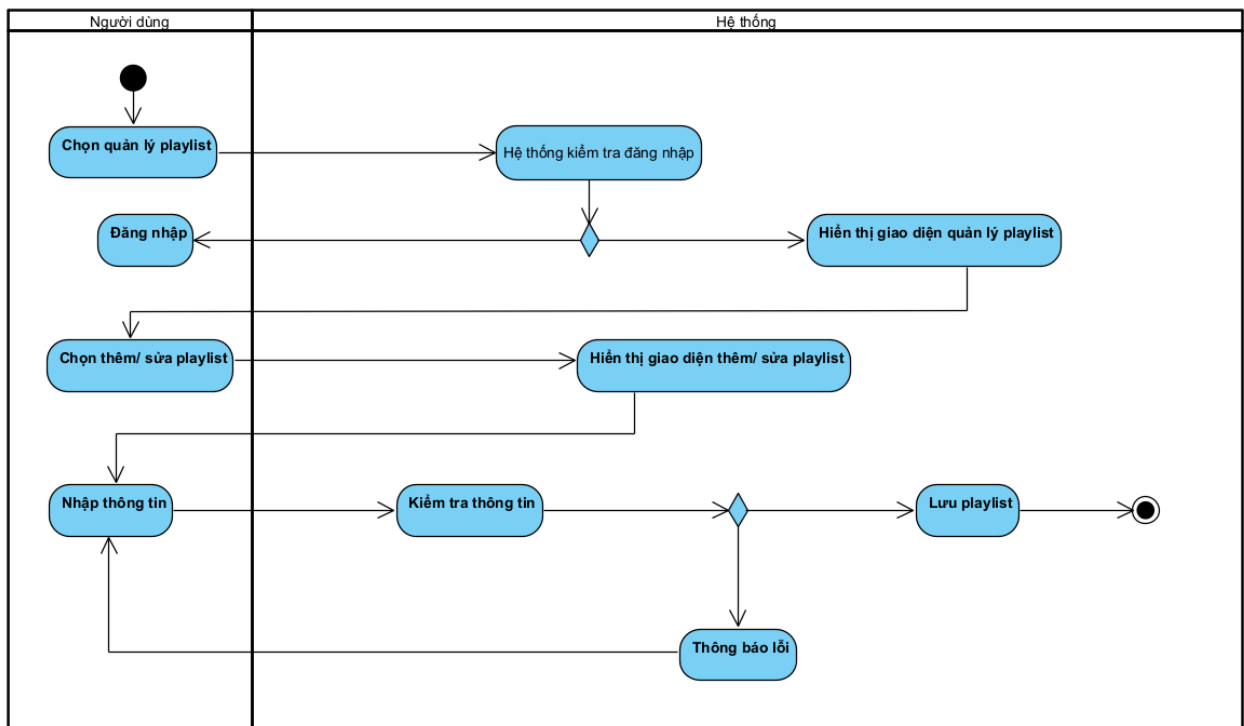
Dưới đây là các biểu đồ hoạt động của hệ thống trong quá trình phân tích và thiết kế:



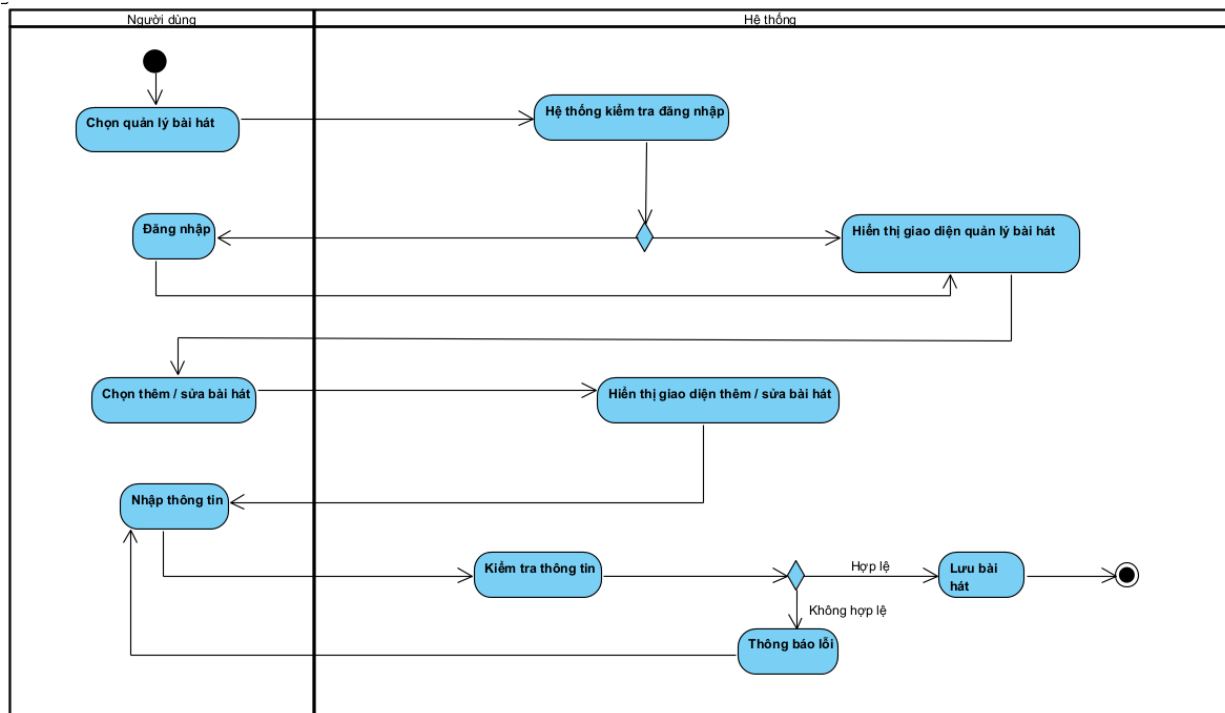
Hình 38: Biểu đồ hoạt động chức năng quản lý thẻ loại



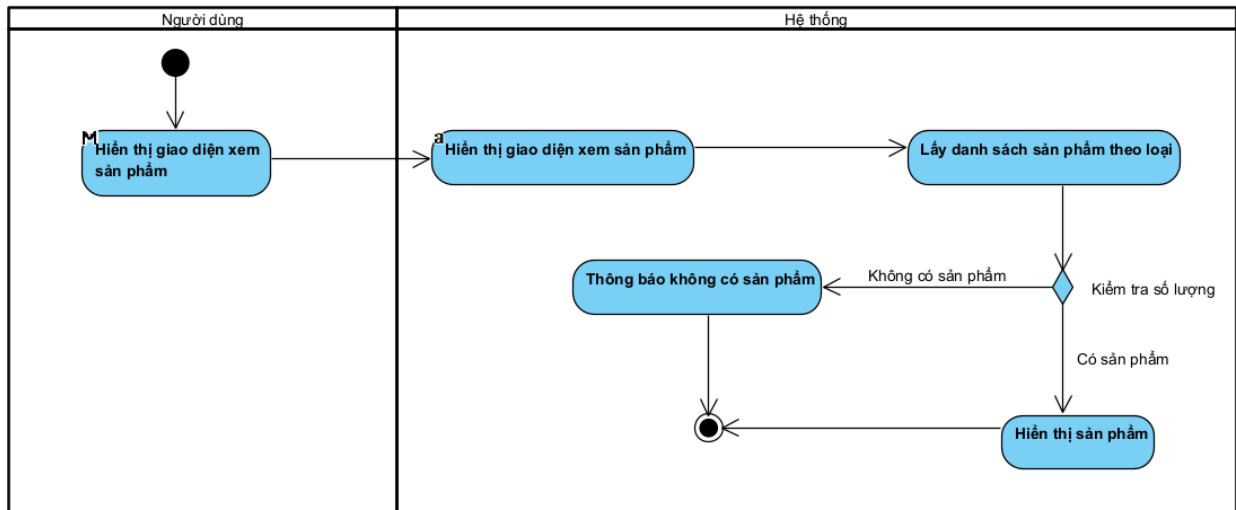
Hình 39: Biểu đồ hoạt động chức năng quản lý album



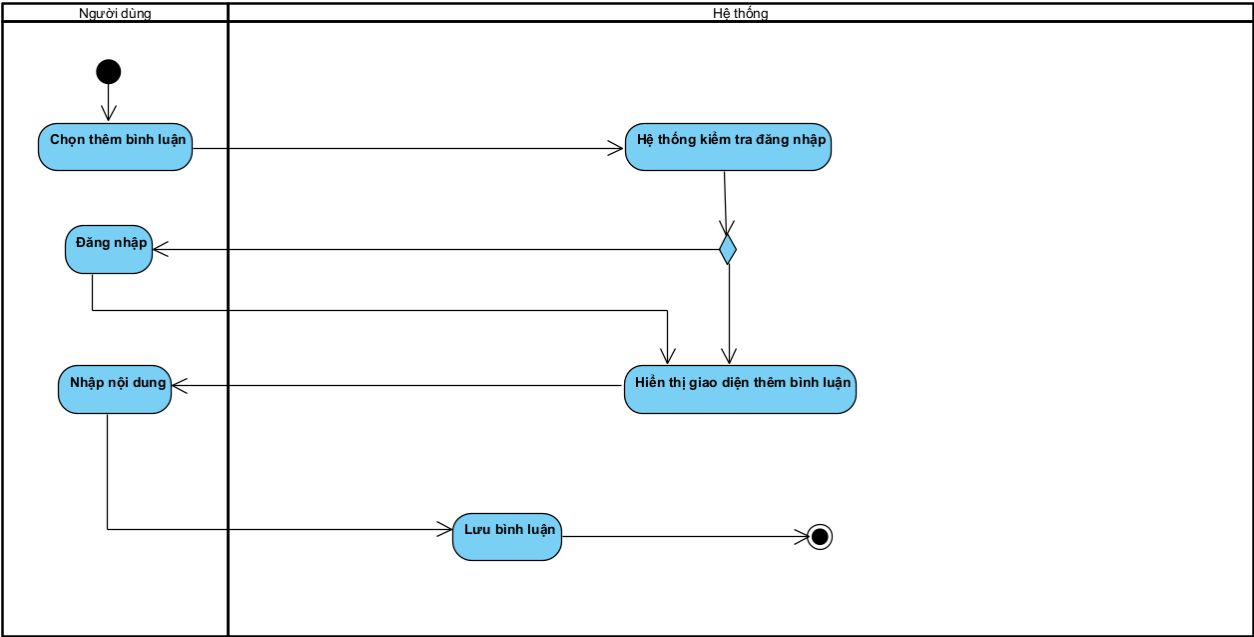
Hình 40: Biểu đồ hoạt động chức năng quản lý playlist



Hình 41: Biểu đồ hoạt động chức năng quản lý bài hát



Hình 42: Biểu đồ hoạt động chức năng xem sản phẩm theo loại



Hình 43: Biểu đồ hoạt động chức năng bình luận

3.4. Thiết kế cơ sở dữ liệu

Sau khi tiến hành phân tích và thiết kế các chức năng cần thiết cho trang web, sau đó sẽ tiến hành thiết kế các bảng cơ sở dữ liệu.

Bảng Genre: Lưu trữ thông tin thể loại

Tên thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
id	int	Mã thể loại	Primary-key
name	Nvarchar(255)	Tên thể loại	
created_at	datetime	Ngày tạo	
updated_at	datetime	Ngày cập nhật	

Bảng 1: Bảng Genre

Bảng Album: Lưu trữ thông tin album

Tên thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
id	int	Mã album	Primary-key
name	nvarchar(255)	Tên album	
image	nvarchar(255)	Hình ảnh	
created_at	datetime	Ngày tạo	
updated_at	datetime	Ngày cập nhật	

Bảng 2: Bảng Album

Bảng Playlist: Lưu trữ thông tin danh sách phát

Tên thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
id	int	Mã playlist	Primary-key
user_id	Int	Mã người dùng	Foreign-key
name	nvarchar(255)	Tên album	
created_at	datetime	Ngày tạo	
updated_at	datetime	Ngày cập nhật	

Bảng 3: Bảng playlist

Bảng Song: Lưu trữ thông tin bài hát

Tên thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
id	int	Mã bài hát	Primary-key
user_id	int	Mã người dùng	Foreign-key
album_id	int	Mã album	Foreign-key
name	nvarchar(255)	Tên bài hát	
description	nvarchar(255)	Mô tả bài hát	
path	nvarchar(255)	Đường dẫn nhạc	
image	nvarchar(255)	Hình ảnh	
created_at	datetime	Ngày tạo	
updated_at	datetime	Ngày cập nhật	

Bảng 4: Bảng Song

Bảng User: Lưu trữ thông tin người dùng

Tên thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
id	int	Mã người dùng	Primary-key
role	int	Quyền	
name	nvarchar(255)	Tên người dùng	
gender	int	Giới tính	
description	nvarchar(255)	Mô tả người dùng	
image	nvarchar(255)	Hình ảnh	
address	nvarchar(255)	Địa chỉ	
social_network	nvarchar(255)	Mạng xã hội	
email	nvarchar(255)	Email	
password	nvarchar(255)	Password	
created_at	datetime	Ngày tạo	
updated_at	datetime	Ngày cập nhật	

Bảng 5: Bảng User

Bảng Comment: Lưu trữ thông tin bình luận

Tên thuộc tính	Kiểu dữ liệu	Mô tả	Ghi chú
id	int	Mã bình luận	Primary-key
user_id	int	Mã người dùng	Foreign_key
parent_id	int	Mã bình luận cha	
song_id	int	Mã bài hát	Foreign_key
content	nvarchar(255)	Nội dung	
created_at	datetime	Ngày tạo	
updated_at	datetime	Ngày cập nhật	

Bảng 6: Bảng Comment

Bảng GenreSong: Lưu trữ thông tin bài hát thuộc thể loại nào

Tên thuộc tính	Kiểu dữ liệu	Mô tả	
id	int	Mã	Primary-key
genre_id	int	Mã thể loại	Foreign_key
song_id	int	Mã bài hát	Foreign_key
created_at	datetime	Ngày tạo	
updated_at	datetime	Ngày cập nhật	

Bảng 7: Bảng GenreSong

Bảng PlaylistSong: Lưu trữ thông tin bài hát đang thuộc danh sách phát nào

Tên thuộc tính	Kiểu dữ liệu	Mô tả	
id	int	Mã	Primary-key
playlist_id	int	Mã playlist	Foreign_key
song_id	int	Mã bài hát	Foreign_key
created_at	datetime	Ngày tạo	
updated_at	datetime	Ngày cập nhật	

Bảng 8: Bảng PlaylistSong

Bảng Interaction: Lưu trữ thông tin tương tác

Tên thuộc tính	Kiểu dữ liệu	Mô tả	
id	int	Mã	Primary-key
user_id	int	Mã người dùng	Foreign_key
song_id	int	Mã bài hát	Foreign_key
liked	int	Thích	
play_count	int	Số lượt nghe	
created_at	datetime	Ngày tạo	
updated_at	datetime	Ngày cập nhật	

Bảng 9: Bảng Interaction

Qua chương này, ta đã hoàn thành phân tích thiết kế hệ thống và đưa ra các chức năng cần thiết của đề tài này, bên cạnh đó cũng đã hoàn thành việc xây dựng cơ sở dữ liệu của hệ thống.

Chương 4. Triển khai hệ thống và đánh giá kết quả đạt được

4.1. Cài đặt môi trường

Đầu tiên, bắt đầu với việc cài đặt Laravel. Để làm được điều này, cần cài đặt Composer, một trình quản lý phụ thuộc cho PHP. Sau khi Composer được cài đặt, tạo một dự án Laravel mới. Tiếp theo, cần cấu hình tệp môi trường để thiết lập các thông tin kết nối cơ sở dữ liệu, như loại cơ sở dữ liệu, địa chỉ máy chủ, tên cơ sở dữ liệu, tên người dùng và mật khẩu. Sau đó, tạo cơ sở dữ liệu MySQL phù hợp và chạy các lệnh migration của Laravel để tạo các bảng trong cơ sở dữ liệu. Để đảm bảo rằng các yêu cầu từ front-end có thể truy cập API của Laravel, cần cài đặt và cấu hình middleware CORS, giúp quản lý các yêu cầu từ các nguồn khác nhau.

Tiếp theo, thiết lập môi trường cho NuxtJS, một framework dựa trên VueJS, được sử dụng cho phần front-end của trang web.

- Trước tiên, cần cài đặt Node.js và npm, trình quản lý gói cho JavaScript. Sau khi cài đặt Node.js và npm, chúng ta tạo một dự án NuxtJS mới.
- Để front-end có thể giao tiếp với back-end Laravel, cài đặt module axios và cấu hình nó để sử dụng địa chỉ API của Laravel.
- Tiếp theo, tạo các component và trang cần thiết trong NuxtJS để hiển thị danh sách bài hát, đồng thời cấu hình để gọi API và lấy dữ liệu từ back-end.

Để áp dụng phong cách cho trang web, sẽ sử dụng Tailwind CSS, một framework CSS tiện dụng và tùy biến cao. Để tích hợp Tailwind CSS vào NuxtJS, cài đặt các gói cần thiết và cấu hình chúng trong dự án NuxtJS.

Cuối cùng sẽ là cài đặt pusher ở cả laravel và nuxtJS, điều này sẽ cho phép phát triển tính năng bình luận thời gian thực.

4.2. Cấu Trúc Tổ Chức Dữ Liệu

4.2.1. Cấu trúc thư mục và file cho NuxtJS

Cấu trúc thư mục và file trong dự án NuxtJS như sau:

- assets/: Thư mục chứa tài nguyên tĩnh như hình ảnh, CSS.
 - images/: Chứa các hình ảnh sử dụng trong ứng dụng.

- styles/: Chứa các file CSS hoặc SCSS để định kiểu cho ứng dụng.
- components/: Thư mục chứa các thành phần giao diện có thể tái sử dụng.
- layouts/: Thư mục chứa các bố cục trang.
- pages/: Thư mục chứa các trang chính của ứng dụng.
- plugins/: Thư mục chứa các plugin của Vue.js hoặc các thư viện cần thiết.
- stores/: Thư mục chứa các module Vuex để quản lý trạng thái ứng dụng.
- nuxt.config.js: File cấu hình NuxtJS.
- package.json: File thông tin và phụ thuộc của dự án.

4.2.2. Cấu Trúc Tổ Chức Dữ Liệu của Laravel (Backend)

- app
 - Http
 - Controllers: Thư mục chứa các controller
 - Middleware: Thư mục chứa các middleware.
 - Requests: Thư mục chứa các request form validation.
 - Models: Thư mục chứa các model.
 - Services: Thư mục chứa các service.
 - Providers: Thư mục chứa các service provider.
- database
 - migrations: Thư mục chứa các migration.
 - seeders: Thư mục chứa các seeder.
- routes
 - api.php: Định nghĩa các route API.
- config: Thư mục chứa các file cấu hình.
- .env: File cấu hình môi trường.

4.3. Phát triển ứng dụng

4.3.1. Phát triển FrontEnd

Dưới đây là một số hình ảnh về trang web sau quá trình phát triển:

- Đăng nhập: Tại trang đăng nhập người dùng có thể thực hiện đăng nhập vào hệ thống, nếu chưa có tài khoản có thể chọn đăng ký.

Welcome to the music web

Email

Password

Đăng nhập

Chưa có tài khoản? [Đăng ký](#)



Hình 44: Trang đăng nhập

- Trang đăng ký: Tại trang này người dùng nhập các thông tin như tên, giới tính, email, password, sau đó xác nhận đăng ký tài khoản.

Welcome to the music web

Tên của bạn

☐ Nam ☐ Nữ

Email

Password

Nhập lại password

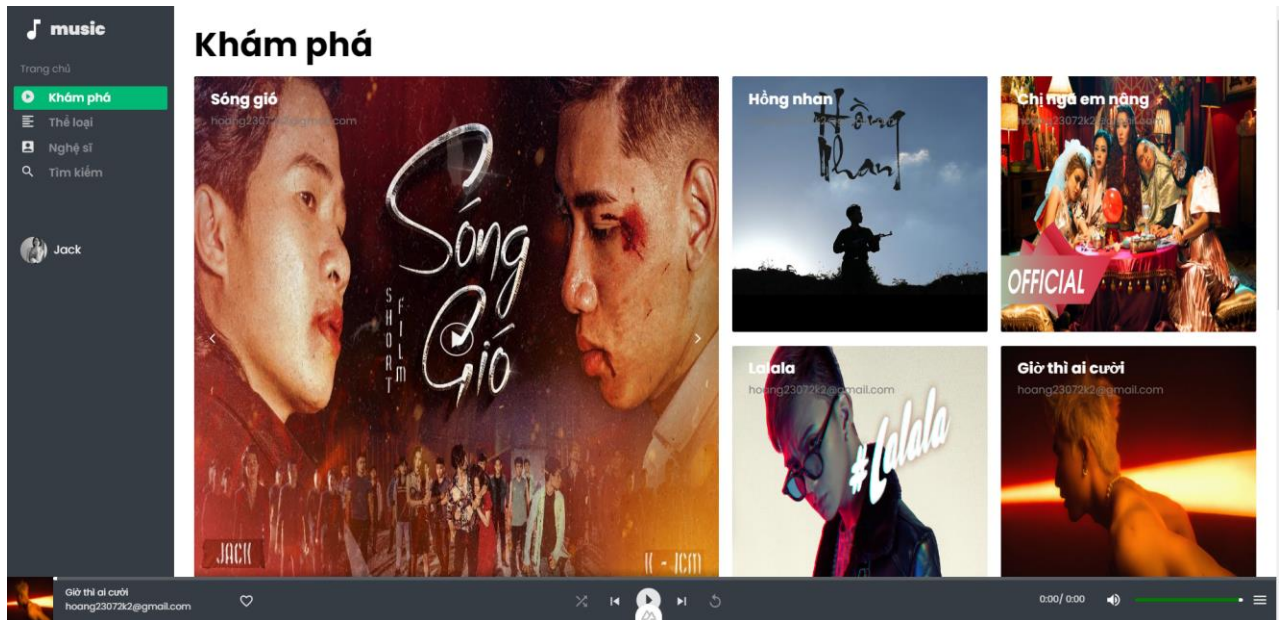
Đăng ký

Đã có tài khoản? [Đăng nhập](#)



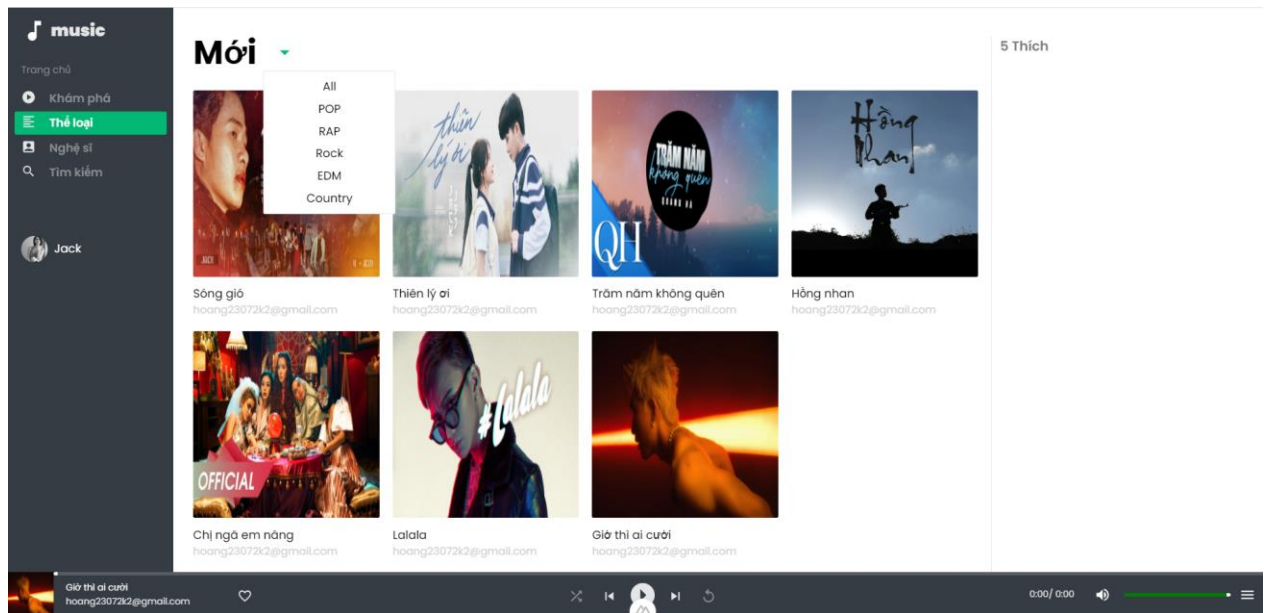
Hình 45: Trang đăng ký

- Trang chủ: Tại trang chủ người dùng có thể theo dõi về các bài hát mới, các bài hát trending. Giao diện người dùng thân thiện cho phép người dùng điều chỉnh âm lượng, chọn chế độ phát nhạc ngẫu nhiên hoặc lặp lại.



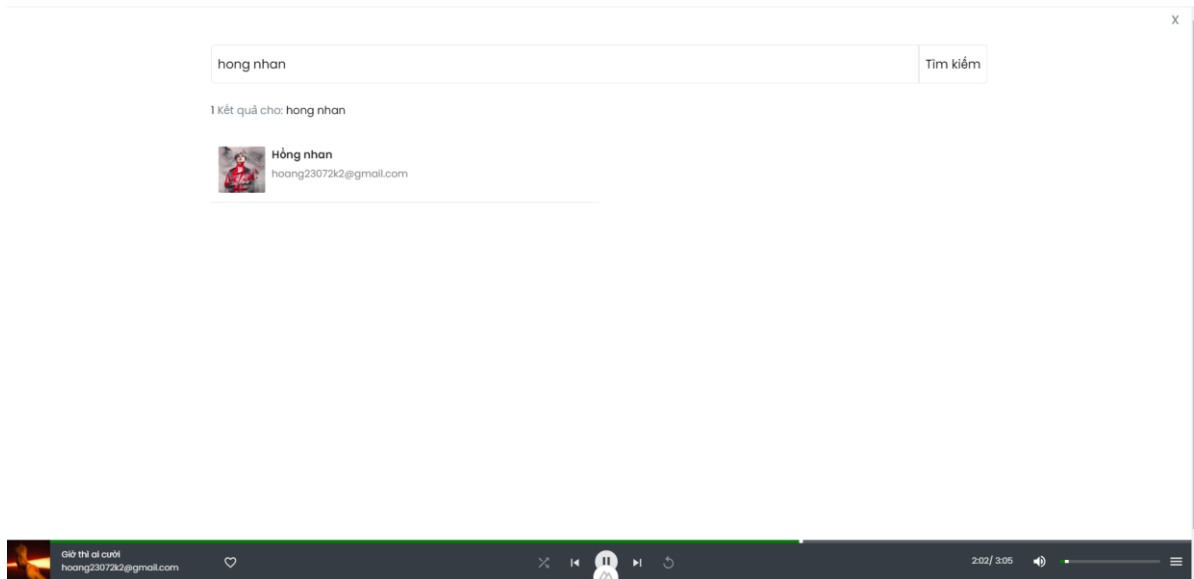
Hình 46: Trang chủ

- Trang xem bài hát theo thể loại: Tại đây người dùng có thể thực hiện chọn thể loại nhạc muốn tìm kiếm, sau đó hệ thống sẽ hiện thị danh sách bài hát lên.



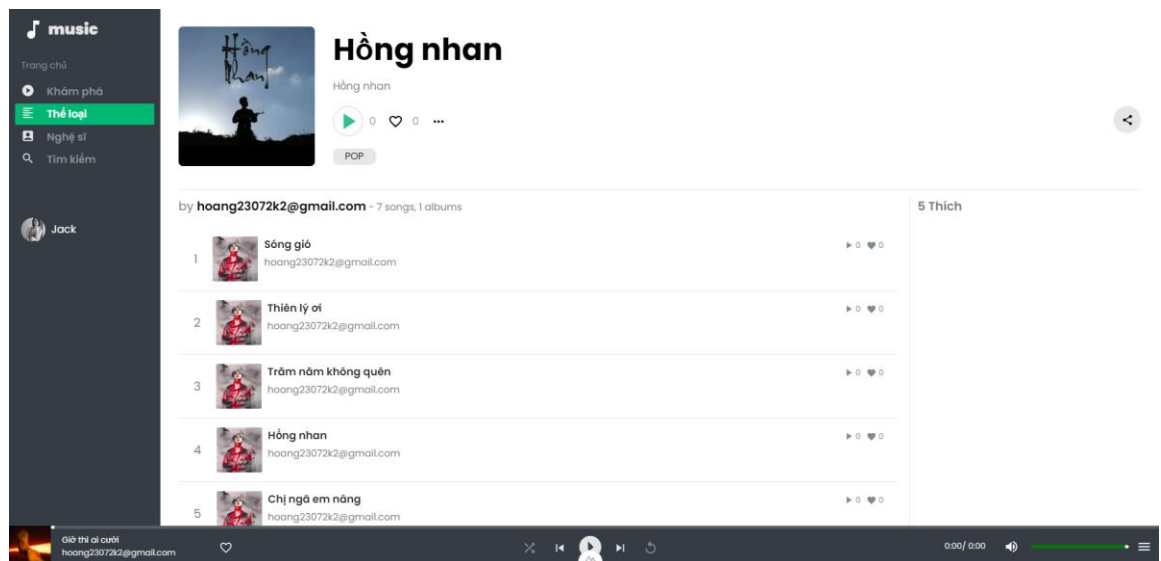
Hình 47: Trang xem bài hát theo thể loại

- Trang tìm kiếm: Tại đây người dùng thực hiện nhập từ khóa và ấn nút tìm kiếm, hệ thống sẽ tìm kiếm và hiển thị kết quả.



Hình 48: Trang tìm kiếm

- Trang chi tiết bài hát: Tại trang chi tiết bài hát, người dùng có thể xem thông tin chi tiết như mô tả, tác giả, các bài hát khác của tác giả đó.



Hình 49: Trang chi tiết bài hát

- **Phản bình luận:** Tại trang chi tiết bài hát, người dùng có thể xem các bình luận về bài hát, có thể đăng tải bình luận mới hoặc trả lời lại bình luận.

Bình luận



Jack 2024-05-26T17:54:45.000000Z
Hay

↩ Reply



hoang23072k2@gmail.com 2024-05-26T17:56:25.000000Z
Đồng ý!!!!

Để lại bình luận

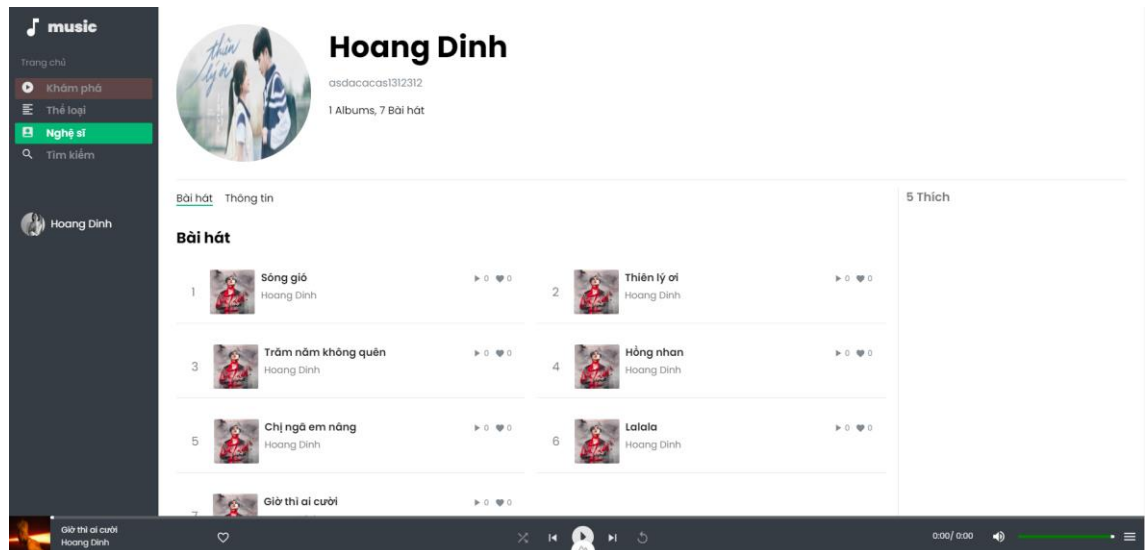
Bình luận

Type your comment

Đăng tải

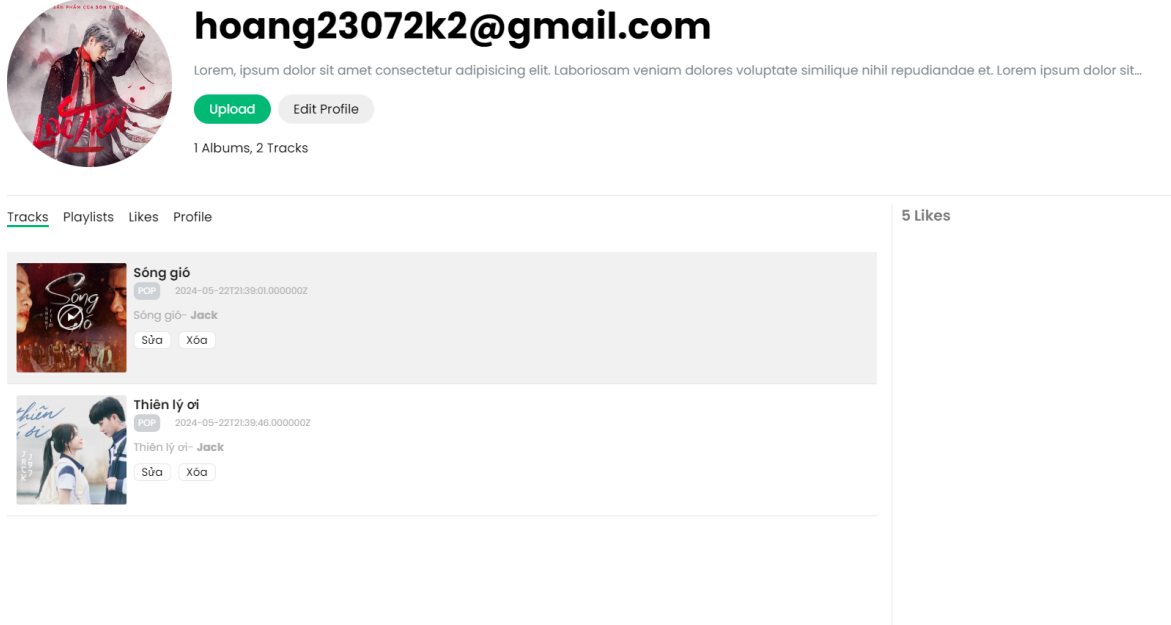
Hình 50: Bình luận

- **Trang chi tiết nghệ sĩ:** Tại trang này, người dùng có thể xem chi tiết về 1 nghệ sĩ, bao gồm thông tin, các bài hát, album mà nghệ sĩ này đã đăng tải.



Hình 51: Trang chi tiết nghệ sĩ

- **Trang cá nhân:** Tại trang cá nhân, người dùng có thể thực hiện xem các thông tin như danh sách phát cá nhân, các bài hát đã tải, danh sách nhạc yêu thích.



hoang23072k2@gmail.com
 Lorem, ipsum dolor sit amet consectetur adipisicing elit. Laboriosam veniam dolores voluptate similique nihil repudiandae et. Lorem ipsum dolor sit...

[Upload](#) [Edit Profile](#)

1 Albums, 2 Tracks

[Tracks](#) [Playlists](#) [Likes](#) [Profile](#)

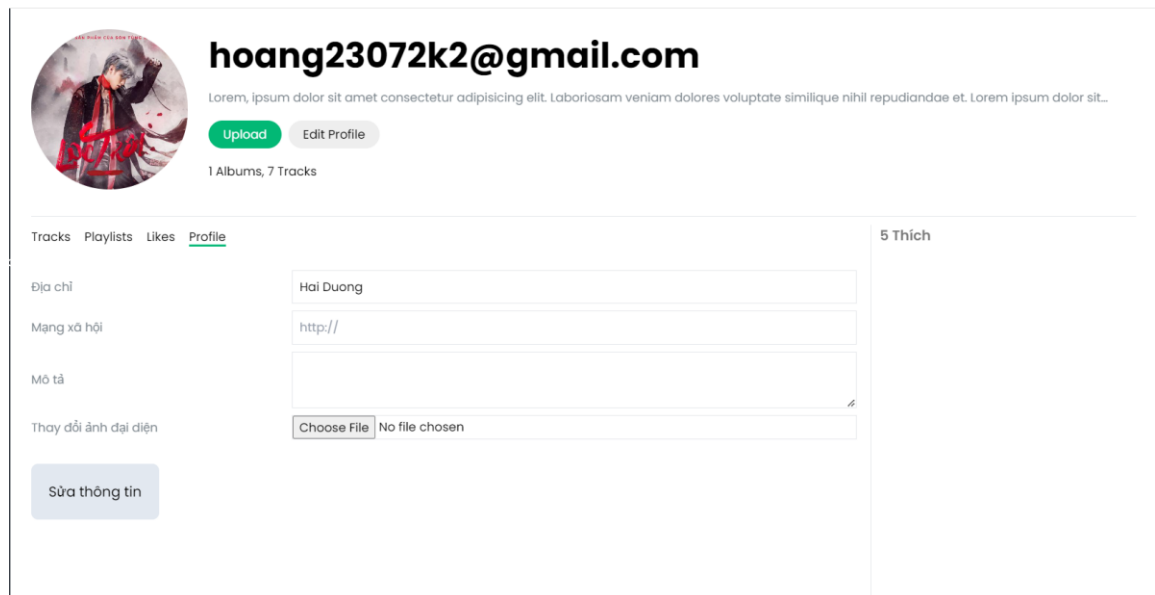
5 Likes

Sóng gió
 POP 2024-05-22T21:39:01.000000Z
 Sóng gió- Jack
[Sửa](#) [Xóa](#)

Thiên lý ơi
 POP 2024-05-22T21:39:46.000000Z
 Thiên lý ơi- Jack
[Sửa](#) [Xóa](#)

Hình 52: Trang cá nhân người dùng

- Trang sửa thông tin người dùng: Tại trang này người dùng có thể thay đổi thông tin của bản thân như địa chỉ, trang mạng xã hội, ảnh đại diện...



hoang23072k2@gmail.com
 Lorem, ipsum dolor sit amet consectetur adipisicing elit. Laboriosam veniam dolores voluptate similique nihil repudiandae et. Lorem ipsum dolor sit...

[Upload](#) [Edit Profile](#)

1 Albums, 7 Tracks

[Tracks](#) [Playlists](#) [Likes](#) [Profile](#)

5 Thích

Địa chỉ: Hai Duong

Mạng xã hội: http://

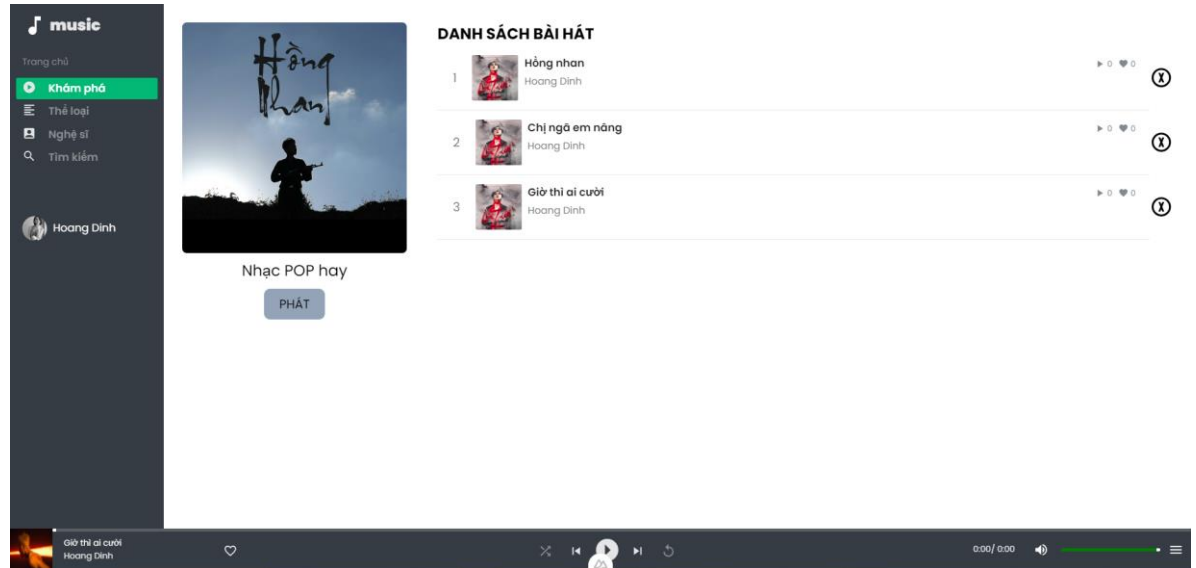
Mô tả:

Thay đổi ảnh đại diện: [Choose File](#) No file chosen

[Sửa thông tin](#)

Hình 53: Trang sửa thông tin người dùng

- Trang chi tiết playlist: Tại trang này người dùng có thể xem chi tiết về 1 playlist của mình, bao gồm danh sách các bài hát và các hành động như phát playlist, xóa bài hát khỏi playlist.



Hình 54: Trang chi tiết playlist

- Giao diện thêm bài hát: Người dùng thực hiện nhập thông tin và thêm bài hát vào hệ thống.

Tên bài hát

Sóng gió

Chọn thể loại

POP

RAP

Rock

EDM

Country

POP

RAP

Mô tả bài hát

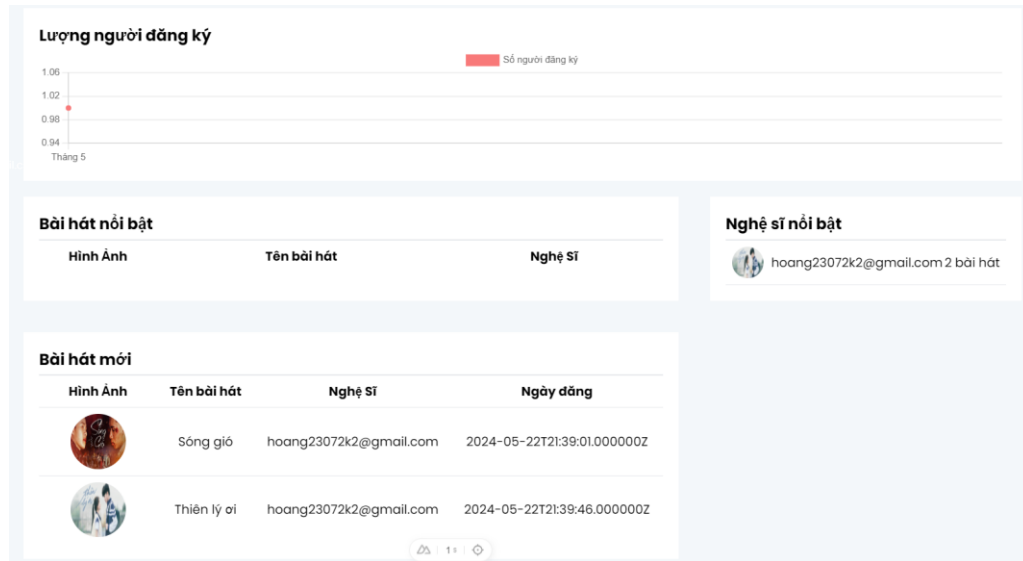
B I @ := != “

Sóng gió- Jack

Chọn ảnh

Hình 55: Giao diện thêm bài hát

- Trang thống kê: Quản trị viên có thể theo dõi các thông tin quan trọng của web như lượng người dùng mới theo từng tháng của năm, các bài hát trending hiện này, các bài hát mới, qua đó có thể đưa ra các hướng phát triển tiếp theo của hệ thống.



Hình 56: Trang thống kê

Kết luận: Ứng dụng nghe nhạc của em đã đạt được mục tiêu là mang lại trải nghiệm tuyệt vời cho người dùng, từ việc nghe nhạc cá nhân đến tương tác xã hội và quản lý playlist. Chúng tôi tin rằng sản phẩm này sẽ góp phần nâng cao chất lượng cuộc sống âm nhạc của người dùng và mở ra nhiều cơ hội phát triển trong tương lai.

4.3.2. Phát triển API

Phần này em sẽ mô tả chi tiết về thiết kế Restful API của hệ thống. Restful API cho phép các thao tác CRUD (Create, Read, Update, Delete) và các chức năng khác như xác thực, quản lý playlist và tương tác xã hội thông qua các phương thức HTTP tiêu chuẩn.

Hệ thống đã thiết kế được gồm 32 API để phục vụ hết cho các nhu cầu của hệ thống.

1 số API đã phát triển như:

- API lấy ra các bài hát trending: để phục vụ cho chức năng gợi ý bài hát đến người dùng.
 - URL: <http://127.0.0.1:8000/api/v1/dashboard/trending-songs>
 - Phương thức HTTP: GET

- Output: các bài hát có số lượng phát và like cao.
- API lấy ra thông tin chi tiết của bài hát: để cho người dùng có thể xem thông tin của 1 bài hát.
 - URL: <http://127.0.0.1:8000/api/v1/songs/{id}>
 - Phương thức HTTP: GET
 - Input: id của bài hát
 - Output: Thông tin chi tiết của bài hát đã chọn.
- API lấy ra thông tin chi tiết của nghệ sĩ: cho chức năng xem chi tiết nghệ sĩ.
 - URL: <http://127.0.0.1:8000/api/v1/users/detail/{id}>
 - Phương thức HTTP: GET
 - Input: id nghệ sĩ
 - Output: Thông tin chi tiết của nghệ sĩ đã chọn.
- API thêm bài hát: cho phép người dùng tải bài hát của mình lên hệ thống.
 - URL: <http://127.0.0.1:8000/api/v1/songs/add>
 - Phương thức HTTP: POST
 - Input: tên bài hát, id album, mô tả, hình ảnh, file nhạc, thể loại, người đăng.
 - Output: Hệ thống trả về thông tin của bài hát vừa đăng tải.
- API cập nhật bài hát: cho phép người dùng cập nhật bài hát của mình.
 - URL: <http://127.0.0.1:8000/api/v1/songs/update/{id}>
 - Phương thức HTTP: PUT
 - Input: tên, mô tả, ảnh, file nhạc, thể loại.
 - Output: Hệ thống trả về thông tin mới bài hát vừa cập nhật.
- API đăng nhập: đăng nhập vào tài khoản người dùng.
 - URL: <http://127.0.0.1:8000/api/v1/users/login>
 - Phương thức HTTP: POST
 - Input: email, mật khẩu.
 - Output: Hệ thống trả về token.
- API đăng ký: cho phép người dùng đăng ký tài khoản cá nhân.
 - URL: <http://127.0.0.1:8000/api/v1/users/register>
 - Phương thức HTTP: POST

- Input: tên, giới tính, email, mật khẩu, xác nhận mật khẩu.
- Output: Hệ thống trả về thông tin tài khoản mới.
- API đăng tải bình luận: cho phép người dùng đăng tải bình luận.
 - URL: <http://127.0.0.1:8000/api/v1/comments/post>
 - Phương thức HTTP: POST
 - Input: mã người dùng, mã bài hát, nội dung, hoặc có thể là mã bình luận gốc.
 - Output: Hệ thống trả về thông tin bình luận.

Kết luận: Với các API này sau khi kết hợp với phần frontend đã tạo nên 1 trang web nghe nhạc đáp ứng được các nhu cầu nghe nhạc của người dùng hiện nay.

4.4. Đánh giá kết quả và hướng phát triển

Sau khi hoàn thành phát triển hệ thống, em sẽ đưa ra 1 số đánh giá về kết quả đạt được và đưa ra hướng phát triển trong tương lai.

Đánh giá kết quả đạt được

- Hoàn thành các chức năng chính
 - Trang web nghe nhạc được phát triển bằng NuxtJS và Laravel API đã hoàn thành các chức năng chính như:
 1. Phát nhạc trực tuyến: Người dùng có thể tìm kiếm và phát nhạc từ thư viện nhạc.
 2. Quản lý tài khoản người dùng: Đăng ký, đăng nhập, và quản lý thông tin cá nhân.
 3. Quản lý danh sách phát: Người dùng có thể tạo, chỉnh sửa và xóa danh sách phát cá nhân.
 4. Giao diện người dùng thân thiện: Sử dụng NuxtJS và Tailwind đã tạo ra giao diện người dùng mượt mà và tối ưu cho trải nghiệm người dùng.
- Hiệu năng và bảo mật
 - Tối ưu hóa hiệu năng: Sử dụng NuxtJS với khả năng render phía server (SSR) giúp cải thiện tốc độ tải trang và trải nghiệm người dùng.
 - Bảo mật API: Laravel API được bảo vệ bởi các phương thức xác thực và phân quyền chặt chẽ, đảm bảo an toàn cho dữ liệu người dùng.

Hướng phát triển:

- Nâng cao trải nghiệm người dùng
 - Tối ưu hóa hiệu năng hơn nữa: Tiếp tục tối ưu hóa mã nguồn và cơ sở dữ liệu để giảm thời gian phản hồi và tải trang, đặc biệt là với những người dùng có kết nối mạng yếu.
 - Cải thiện giao diện người dùng: Lắng nghe phản hồi từ người dùng để điều chỉnh giao diện và chức năng theo nhu cầu thực tế.
- Phát triển tính năng mới
 - Tính năng gợi ý thông minh: Sử dụng machine learning để đề xuất các bài hát và danh sách phát dựa trên thói quen nghe nhạc của người dùng.
 - Chế độ nghe offline: Phát triển tính năng cho phép người dùng tải về và nghe nhạc offline khi không có kết nối internet.
- Phát triển ứng dụng di động: Xây dựng ứng dụng di động cho cả Android và iOS để người dùng có thể trải nghiệm dịch vụ trên nhiều nền tảng.

KẾT LUẬN

Sau một quá trình nghiên cứu và triển khai, đồ án "Xây dựng ứng dụng website nghe nhạc sử dụng NuxtJS và Laravel" đã hoàn thành với những kết quả đáng khích lệ. Việc áp dụng các công nghệ hiện đại như NuxtJS và Laravel đã giúp em xây dựng một trang web nghe nhạc trực tuyến có giao diện thân thiện, hiệu suất cao và đáp ứng được các yêu cầu đề ra. Trang web đã tích hợp đầy đủ các chức năng cơ bản như phát nhạc trực tuyến, quản lý tài khoản người dùng, quản lý danh sách phát và tìm kiếm nhạc. Mặc dù đạt được nhiều kết quả tích cực, cũng có những nhược điểm cần cải thiện trong tương lai, hiệu năng của trang web vẫn có thể được nâng cao hơn, và một số lỗi nhỏ cần được khắc phục dựa trên phản hồi của người dùng.

Để tiếp tục phát triển dự án này trong tương lai, em định hướng nâng cao trải nghiệm người dùng thông qua tối ưu hóa hiệu năng và cải thiện giao diện dựa trên phản hồi thực tế. Bổ sung các tính năng mới như gợi ý thông minh, chế độ nghe offline và tích hợp với các dịch vụ nhạc số khác sẽ là ưu tiên hàng đầu. Bên cạnh đó, việc phát triển ứng dụng di động và mở rộng thư viện nhạc thông qua hợp tác với các nền tảng khác cũng sẽ được chú trọng.

Cuối cùng, đồ án này không chỉ giúp em nâng cao kiến thức và kỹ năng lập trình mà còn cung cấp một cái nhìn toàn diện về quy trình phát triển một sản phẩm công nghệ từ giai đoạn ý tưởng đến khi triển khai thực tế.

TÀI LIỆU THAM KHẢO

Tiếng Việt

[1] TS. Chu Thị Minh Huệ, Giáo trình phân tích và thiết kế hướng đối tượng với UML.

Danh mục các Website tham khảo

[1] Nguyen Quang Phu. (1/5/2020). Giới thiệu về NuxtJs. <https://viblo.asia/p/gioi-thieu-ve-nuxtjs-Qbq5Q0qGID8>

[2] NuxtLabs. (15/3/2024). NuxtJS. <https://nuxt.com/>

[2] Evan You. (10/3/2024). VueJS. <https://vuejs.org/>

[3] Taylor Otwell. (20/4/2024). Laravel. <https://laravel.com/>