TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

HỌC PHẦN: PHẦN MỀM MÃ NGUỒN MỞ

Phần mềm Spotify

Nhóm sinh viên thực hiện:

Họ và tên	**MSSV**
Văn Tuấn Kiệt	3122410202
Mai Phúc Lâm	3122410207
Nguyễn Đức Duy Lâm	3122410208
Nguyễn Hữu Lộc	3122410213
Hồ Hưng Lộc	3122410219
Nguyễn Đình Thông	3122410400

Giáo viên hướng dẫn: ThS. Từ Lãng Phiêu

TP.HCM, 2025

Mục lục

Lời cảm ơn **2**

1 GIỚI THIỆU **3**
1.1 Giới thiệu sơ lược về đề tài 3
1.2 Lý do chọn đề tài 4
1.3 Mục tiêu 4

2 ĐẶC TẢ YÊU CẦU PHẦN MỀM **6**
2.1 Yêu cầu chức năng 6
2.1.1 Người dùng 6
2.1.2 Quản lý 7
2.2 Yêu cầu phi chức năng 7

3 KIẾN TRÚC PHẦN MỀM **9**
3.1 Mô hình ứng dụng 9
3.2 Công cụ 12
3.3 Kiến trúc phần mềm 17

4 SƠ ĐỒ ERD **19**
4.1 Mô hình Dữ liệu 19
4.1.1 Mô hình Người Dùng (NguoiDung) 19
4.1.2 Mô hình Nghệ Sĩ (NgheSi) 20
4.1.3 Mô hình Album 21
4.1.4 Mô hình Bài Hát (BaiHat) 21
4.1.5 Mô hình Danh Sách Phát (DanhSachPhat) 22
4.1.6 Mô hình Thanh Toán (ThanhToan) 22

5 BÁO CÁO KẾT QUẢ **24**

6 HƯỚNG PHÁT TRIỂN **25**

Lời cảm ơn

Trong suốt quá trình học tập môn Phần mềm mã nguồn mở và thực hiện phần mềm "Spotify", chúng em xin gửi lời cảm ơn chân thành nhất đến tất cả những người đã hỗ trợ và đồng hành cùng chúng em trong suốt chặng đường này.

Trước tiên, chúng em xin bày tỏ lòng biết ơn sâu sắc đến giảng viên Từ Lãng Phiêu, người đã tận tình hướng dẫn chúng em từ những bước đầu tiên cho đến khi phần mềm được hoàn thành. Thầy không chỉ cung cấp những kiến thức quý báu mà còn tạo điều kiện thuận lợi để chúng em nghiên cứu, học hỏi và áp dụng vào thực tế. Sự hướng dẫn tận tâm, những góp ý sắc sảo cùng kinh nghiệm thực tế mà thầy chia sẻ đã giúp chúng em củng cố nền tảng chuyên môn vững chắc, đồng thời vượt qua những khó khăn trong quá trình phát triển phần mềm.

Bên cạnh đó, chúng em cũng xin gửi lời cảm ơn chân thành đến các thành viên trong nhóm, những người đã luôn hỗ trợ, đồng hành và chia sẻ kinh nghiệm trong suốt quá trình thực hiện dự án. Sự hợp tác chặt chẽ, tinh thần làm việc nhóm hiệu quả cùng những trải nghiệm thực tế từ các bạn đã giúp chúng em làm quen với quy trình làm việc chuyên nghiệp, tạo nền tảng quan trọng cho con đường phát triển sau này.

Mặc dù đã nỗ lực hết mình để hoàn thành phần mềm, nhưng do thời gian và kiến thức còn hạn chế, chắc chắn vẫn còn những thiếu sót. Chúng em rất mong nhận được sự góp ý từ quý thầy cô và các bạn để có thể tiếp tục cải thiện và hoàn thiện sản phẩm hơn nữa. Những ý kiến đóng góp của thầy cô sẽ là nguồn động lực to lớn, giúp chúng em rút ra bài học kinh nghiệm quý báu và không ngừng phát triển trong tương lai.

Chúng em xin chân thành cảm ơn!

Chương 1

GIỚI THIỆU

1.1 Giới thiệu sơ lược về đề tài

Spotify là một nền tảng phát nhạc trực tuyến hàng đầu thế giới, cung cấp cho người dùng quyền truy cập vào hàng triệu bài hát, podcast và video từ các nghệ sĩ trên toàn cầu. Ra mắt vào năm 2008, Spotify đã thay đổi cách mọi người tiếp cận và thưởng thức âm nhạc, chuyển từ việc sở hữu bản ghi vật lý sang mô hình phát trực tuyến tiện lợi và hợp pháp.

Spotify hoạt động trên nhiều thiết bị, bao gồm máy tính, điện thoại di động, máy tính bảng, loa thông minh, TV và ô tô, cho phép người dùng nghe nhạc mọi lúc, mọi nơi. Nền tảng này cung cấp cả phiên bản miễn phí với quảng cáo và phiên bản trả phí (Spotify Premium) với nhiều tính năng nâng cao như nghe nhạc offline, chất lượng âm thanh cao hơn và không có quảng cáo. Spotify cũng nổi tiếng với khả năng cá nhân hóa trải nghiệm người dùng thông qua các playlist được tạo tự động dựa trên thói quen nghe nhạc, như "Discover Weekly"và "Daily Mix".

1.2 Lý do chọn đề tài

Trước khi Spotify xuất hiện, ngành công nghiệp âm nhạc đối mặt với nhiều thách thức, đặc biệt là vấn đề vi phạm bản quyền do việc chia sẻ nhạc trái phép trên các nền tảng như Napster. Daniel Ek và Martin Lorentzon, những người sáng lập Spotify, nhận thấy cần thiết phải tạo ra một dịch vụ âm nhạc trực tuyến hợp pháp, cung cấp trải nghiệm nghe nhạc chất lượng cao và thuận tiện, đồng thời đảm bảo quyền lợi cho các nghệ sĩ và nhà sản xuất. Mục tiêu của họ là cung cấp một giải pháp thay thế hấp dẫn hơn so với việc tải nhạc bất hợp pháp, bằng cách mang đến cho người dùng quyền truy cập tức thì vào một thư viện âm nhạc khổng lồ với chất lượng cao.

1.3 Mục tiêu

Mục tiêu chính của Spotify là "democratize audio"– dân chủ hóa việc tiếp cận âm thanh. Điều này có nghĩa là cung cấp cho người dùng trên toàn thế giới quyền truy cập dễ dàng và hợp pháp vào kho nội dung âm thanh phong phú, đồng thời tạo cơ hội cho các nghệ sĩ, dù lớn hay nhỏ, tiếp cận với khán giả toàn cầu mà không cần thông qua các kênh phân phối truyền thống. Spotify cũng đặt mục tiêu không ngừng cải thiện trải nghiệm người dùng thông qua việc ứng dụng công nghệ tiên tiến như trí tuệ nhân tạo để cá nhân hóa nội dung và đề xuất âm nhạc phù hợp với sở thích của từng cá nhân.

Ngoài ra, Spotify còn hướng đến việc mở rộng hệ sinh thái âm thanh của

mình bằng cách tích hợp podcast và audiobook, biến nền tảng này thành điểm đến toàn diện cho mọi nhu cầu nghe của người dùng. Điều này không chỉ tăng cường giá trị cho người dùng mà còn tạo thêm nguồn thu nhập cho các nhà sáng tạo nội dung.

Tóm lại, Spotify được phát triển với mục tiêu cung cấp một giải pháp nghe nhạc trực tuyến hợp pháp, chất lượng cao và thuận tiện, đồng thời hỗ trợ các nghệ sĩ tiếp cận khán giả rộng rãi hơn. Nền tảng này không ngừng đổi mới và mở rộng để đáp ứng nhu cầu ngày càng cao của người dùng và thị trường âm nhạc toàn cầu.

Chương 2

ĐẶC TẢ YÊU CẦU PHẦN MỀM

2.1 Yêu cầu chức năng

2.1.1 Người dùng

- **Đăng ký và đăng nhập:** Người dùng có thể tạo tài khoản và đăng nhập vào hệ thống. Người dùng có thể đặt lại mật khẩu nếu quên mật khẩu.
- **Tìm kiếm bài hát:** Người dùng có thể tìm kiếm bài hát bằng tên bài hát, tên nghệ sĩ và có thể tìm kiếm album bằng tên bài hát, tên nghệ sĩ.
- **Phát nhạc:** Người dùng có thể phát bài hát, album, và danh sách phát yêu thích.
- **Danh sách phát:** Người dùng có thể thêm bài hát vào danh sách phát cá nhân của mình hay tải xuống.
- **Quản lý thư viện cá nhân:** Người dùng có thể xem và chỉnh sửa danh sách bài hát, bài hát yêu thích.
- **Quản lý thông tin cá nhân:** Người dùng có thể xem thông tin cá nhân của mình.
- **Premium:** Nếu người dùng chưa đăng ký tài khoản premium thì sẽ phải nghe các quảng cáo sau mỗi 2 bài hát, đồng thời không được nghe các bài hát vip.

- **Thanh toán:** Người dùng có thể thanh toán cho các gói dịch vụ cao cấp qua các phương thức thanh toán trực tuyến bao gồm ZaloPay, Paypal.
- **Tìm bài hát dựa trên giai điệu:** Người dùng có thể tìm bài hát thông qua việc ghi âm một đoạn nhạc nào đó, hệ thống sẽ hiển thị bài hát tương ứng nếu tìm ra.

2.1.2 Quản lý

- **Quản lý người dùng:** Quản lý có thể xem danh sách người dùng, thêm, cập nhật, và tìm kiếm người dùng theo tên, email, hoặc số điện thoại.
- **Quản lý danh sách phát chung:** Quản lý có thể tạo và duy trì các danh sách phát chung cho người dùng.
- **Thống kê:** Quản lý có thể thống kê số lượng người dùng, lượt nghe, và các bài hát phổ biến theo thời gian.
- **Quản lý album:** Quản lý có thể xem và quản lý thông tin các album.
- **Quản lý nghệ sĩ:** Quản lý có thể xem và quản lý thông tin các nghệ sĩ.

2.2 Yêu cầu phi chức năng

Hiệu suất

- **Tìm kiếm nhanh chóng:** Các thao tác tìm kiếm phải có thời gian phản hồi dưới 5 giây.
- **Xử lý giao dịch đồng thời:** Hệ thống phải có khả năng xử lý 500 giao dịch đồng thời mà không gặp phải độ trễ.

Tính bảo mật

- **Mã hóa SSL cho thanh toán:** Đảm bảo các giao dịch tài chính được bảo vệ thông qua SSL/TLS.
- **Xác thực hai yếu tố (2FA):** Sử dụng xác thực OTP hoặc sinh trắc học để tăng cường bảo mật khi người dùng thực hiện giao dịch nhạy cảm.

Tính sẵn có

- **Hoạt động 24/7:** Hệ thống cần phải luôn sẵn sàng với mức độ uptime 99.9

Khả năng sử dụng

- **Giao diện thân thiện:** Giao diện người dùng phải đơn giản và dễ sử dụng, với các tính năng quan trọng hiển thị rõ ràng và dễ tìm.
- **Hỗ trợ đa ngôn ngữ:** Hệ thống cần cung cấp hỗ trợ cho nhiều ngôn ngữ và đảm bảo khả năng truy cập cho người dùng có nhu cầu đặc biệt.

Tính tương thích

- **Tương thích với các trình duyệt:** Ứng dụng cần phải tương thích với các trình duyệt web phổ biến như Chrome, Cốc cốc, và Edge.
- **Tương thích với thiết bị di động:** Giao diện cần phải responsive và tương thích với các hệ điều hành di động (iOS, Android).

Tính bảo trì

- **Dễ bảo trì và mở rộng tính năng:** Cần áp dụng kiến trúc modular để dễ dàng thêm hoặc thay đổi chức năng mà không ảnh hưởng đến toàn bộ hệ thống.
- **CI/CD:** Sử dụng quy trình CI/CD để tự động triển khai và cập nhật hệ thống mà không cần downtime.

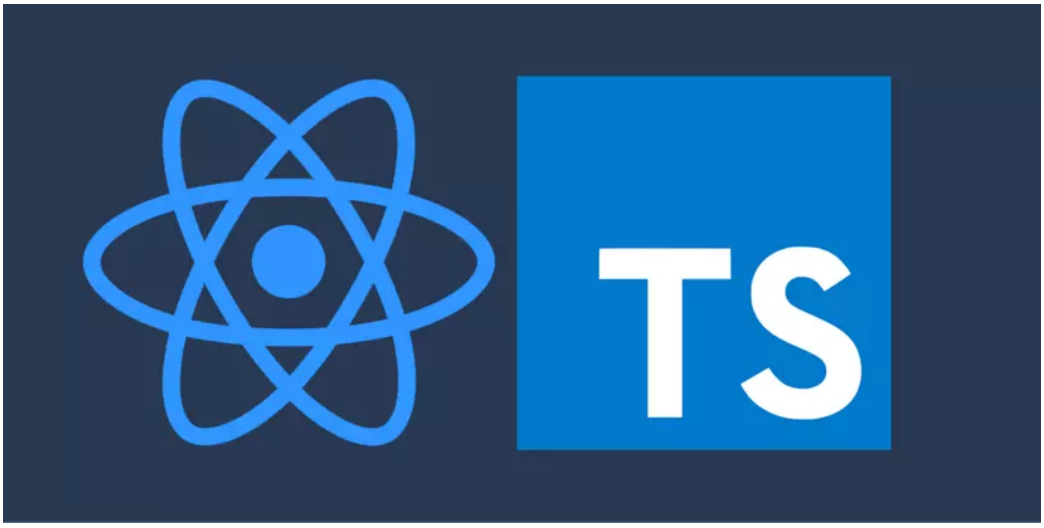
Chương 3

KIẾN TRÚC PHẦN MỀM

3.1 Mô hình ứng dụng

Frontend:

- **ReactJS:** React là một thư viện JavaScript mã nguồn mở, được phát triển bởi Facebook vào năm 2013, nhằm xây dựng giao diện người dùng cho các ứng dụng web. React cho phép các lập trình viên phát triển các thành phần giao diện (components) một cách hiệu quả, dễ bảo trì và tái sử dụng. Với React, người dùng có thể trải nghiệm giao diện mượt mà và dễ dàng tương tác với ứng dụng.



Backend:

- **Python - Django:** Django là một framework Python mạnh mẽ giúp phát triển các ứng dụng web nhanh chóng và dễ dàng hơn. Django cung

cấp các tính năng như ORM (Object Relational Mapping) để quản lý cơ sở dữ liệu, giúp việc phát triển các API RESTful trở nên đơn giản và thuận tiện. Nó cũng hỗ trợ bảo mật và quản lý người dùng, giúp dễ dàng xử lý các yêu cầu API của người dùng.

Hệ quản trị cơ sở dữ liệu:

- **MySQL:** là một hệ thống quản trị cơ sở dữ liệu mã nguồn mở (Relational Database Management System, viết tắt là RDBMS), thuộc quyền sở hữu của Oracle, được sử dụng để quản lý và lưu trữ dữ liệu. Nó sử dụng SQL (Structured Query Language) làm ngôn ngữ chính để truy vấn và thao tác với cơ sở dữ liệu. MySQL phổ biến trong các ứng dụng web, đặc biệt là các ứng dụng sử dụng kiến trúc LAMP (Linux, Apache, MySQL, PHP/Python/Perl).Các ứng dụng web lớn nhất như Facebook, Twitter, YouTube, Google, và Yahoo! đều dùng MySQL cho mục đích lưu trữ dữ liệu. Nó đã tương thích với nhiều hạ tầng máy tính quan trọng như Linux, macOS, Microsoft Windows, và Ubuntu.

Giao thức truyền thông:

- **WebSocket:** là một giao thức truyền thông giúp cho việc thiết lập kênh truyền thông hai chiều giữa máy chủ và máy khách. WebSocket hoạt động bằng cách thiết lập kết nối HTTP liên tục với máy chủ và sau đó nâng cấp nó lên kết nối websocket hai chiều bằng cách gửi Upgrade header. WebSocket được hỗ trợ trong hầu hết các trình duyệt web hiện đại và cho các trình duyệt không hỗ trợ, chúng tôi có các thư viện cung cấp dự phòng cho các kỹ thuật khác như Comet và HTTP Long Polling.

3.2 Công cụ

PyCharm:

- PyCharm là một môi trường phát triển tích hợp (Integrated Development Environment - IDE) được thiết kế chuyên biệt cho lập trình Python, do JetBrains phát triển và ra mắt lần đầu tiên vào tháng 2 năm 2010. PyCharm cung cấp các tính năng mạnh mẽ như gợi ý mã thông minh, kiểm tra lỗi thời gian thực, tích hợp công cụ quản lý phiên bản (Git, SVN), và hỗ trợ phát triển web với các framework như Django và Flask. IDE này có hai phiên bản: Community (miễn phí, mã nguồn mở) và Professional (trả phí, với nhiều tính năng nâng cao). PyCharm được sử dụng rộng rãi trong các dự án phát triển phần mềm, đặc biệt là các ứng dụng Python, và tương thích với nhiều nền tảng như Windows, macOS, và Linux.

Visual Studio Code (VS Code):

- Visual Studio Code là một trình soạn thảo mã nguồn nhẹ nhưng mạnh mẽ, hỗ trợ đa nền tảng như Windows, macOS và Linux. Với các tính năng như hoàn thành mã thông minh, tích hợp Git, và hỗ trợ nhiều tiện ích mở rộng, VS Code được sử dụng để phát triển frontend của dự án Instagram clone với ReactJS. Công cụ này giúp lập trình viên dễ dàng viết mã, gỡ lỗi, và quản lý các thư viện như Ant Design, Redux, và TailwindCSS một cách hiệu quả.

Postman:

- Postman là một công cụ phổ biến được sử dụng để kiểm thử và làm việc với các API, đặc biệt là API kiểu REST. API đóng vai trò quan trọng trong việc kết nối các thành phần của ứng dụng, và Postman giúp đơn giản hóa quá trình gọi và kiểm tra API mà không cần viết mã. Công cụ này hỗ trợ tất cả các phương thức HTTP như GET, POST, PUT, PATCH, DELETE, và cho phép lưu lại lịch sử các yêu cầu (request) để sử dụng lại khi cần. Trong dự án Instagram clone, Postman được sử dụng để kiểm tra các API như đăng bài post (POST /api/post), lấy danh sách bài post (GET /api/post), và gửi tin nhắn.

MySQL Workbench:

- MySQL Workbench là một công cụ quản lý cơ sở dữ liệu MySQL, cung cấp giao diện đồ họa để phát triển, thiết kế và quản lý cơ sở dữ liệu một cách trực quan. Công cụ này cho phép dễ dàng tạo, chỉnh sửa cơ sở dữ liệu, thực hiện các thao tác như đảo ngược (reverse engineering) để tạo mô hình từ cơ sở dữ liệu hiện có, và chuyển tiếp (forward engineering) để triển khai mô hình thành cơ sở dữ liệu. Trong dự án Instagram clone, MySQL Workbench được sử dụng để thiết kế và quản lý cơ sở dữ liệu MySQL, bao gồm các bảng như users, posts, comments, và messages.

GitHub:

- GitHub là một nền tảng quản lý mã nguồn phổ biến, cho phép các lập trình viên chia sẻ, cộng tác và quản lý phiên bản mã nguồn một cách hiệu quả. Sự phát triển của GitHub bắt đầu vào ngày 19 tháng 10 năm 2007, và trang web chính thức được ra mắt vào tháng 4 năm 2008

bởi **Tom Preston-Werner**, **Chris Wanstrath**, và **PJ Hyett**. Microsoft đã mua lại GitHub vào tháng 6 năm 2018, giúp nền tảng này có thêm nhiều nguồn lực và tích hợp sâu hơn với các sản phẩm của Microsoft, như Azure và Visual Studio.

- GitHub cung cấp không chỉ một kho lưu trữ mã nguồn mà còn là một công cụ mạnh mẽ để cộng tác và chia sẻ mã nguồn giữa các lập trình viên trên toàn cầu. Git, hệ thống quản lý phiên bản mà GitHub sử dụng, cho phép các lập trình viên theo dõi và kiểm soát sự thay đổi trong mã nguồn theo thời gian, giúp họ có thể quay lại các phiên bản trước nếu cần thiết và làm việc đồng thời mà không gặp phải vấn đề xung đột dữ liệu.

- GitHub cung cấp các tính năng như **branches** (nhánh) để làm việc song song trên các tính năng hoặc sửa lỗi mà không làm ảnh hưởng đến mã nguồn chính. Người dùng có thể **fork** (tạo bản sao) các dự án từ người khác để làm việc trên đó và sau đó tạo **pull request** (yêu cầu hợp nhất) để đóng góp các thay đổi của mình vào dự án gốc.

- GitHub cũng cung cấp các công cụ hỗ trợ như **issue tracking** (theo dõi vấn đề), **project boards** (bảng dự án), **actions** (tự động hóa quy trình phát triển), và **wikis** để tài liệu hóa các dự án, giúp các nhóm lập trình viên dễ dàng hợp tác, theo dõi tiến độ, và quản lý các vấn đề phát sinh trong suốt quá trình phát triển phần mềm.

- Ngoài ra, GitHub còn có tính năng **GitHub Pages**, cho phép người dùng lưu trữ các trang web tĩnh trực tiếp từ kho GitHub của mình. GitHub còn hỗ trợ **private repositories** (kho lưu trữ riêng tư), giúp các lập trình viên hoặc tổ chức bảo vệ mã nguồn của họ khỏi việc chia sẻ công khai.

- Nói chung, GitHub là một công cụ cực kỳ quan trọng trong cộng đồng lập trình viên, giúp tăng cường sự hợp tác, bảo vệ mã nguồn và thúc đẩy quy trình phát triển phần mềm nhanh chóng và hiệu quả.

Giấy phép mã nguồn mở (Open Source License):

- Giấy phép Mã nguồn mở (Open Source License) là các giấy phép cho phép người dùng tự do truy cập, sử dụng, sửa đổi và phân phối phần mềm, miễn là họ tuân theo các điều kiện của giấy phép. Các giấy phép này giúp phần mềm được phát triển và chia sẻ rộng rãi, thúc đẩy cộng đồng đóng góp vào việc phát triển phần mềm. Một số giấy phép mã nguồn mở phổ biến bao gồm: MIT License, GPL (General Public

License), và Apache License 2.0. MIT License là một trong những giấy phép mã nguồn mở đơn giản nhất, cho phép người dùng sử dụng, sao chép, sửa đổi và phân phối phần mềm miễn phí, nhưng yêu cầu phải bao gồm bản quyền và thông báo giấy phép. GPL yêu cầu phần mềm phát hành dưới mã nguồn mở và bất kỳ phần mềm phát triển từ phần mềm này cũng phải tiếp tục là mã nguồn mở. Apache License 2.0 tương đối thoải mái nhưng yêu cầu bảo vệ các quyền sở hữu trí tuệ liên quan đến phần mềm. Các giấy phép mã nguồn mở thúc đẩy sự phát triển của phần mềm thông qua cộng đồng và giúp bảo vệ quyền lợi của các tác giả phần mềm.

Windows Subsystem for Linux (WSL):

- Windows Subsystem for Linux (WSL) là một tính năng trên Windows 10 và Windows Server 2019, cho phép người dùng chạy môi trường Linux trực tiếp trên Windows mà không cần phải cài đặt một máy ảo hoặc hệ điều hành dual-boot. WSL giúp các nhà phát triển sử dụng các công cụ và phần mềm Linux, ngay cả khi họ đang làm việc trên hệ điều hành Windows. WSL cung cấp một trải nghiệm người dùng rất tiện lợi vì nó cho phép sử dụng các công cụ dòng lệnh của Linux, như Bash, grep, sed, awk, và nhiều phần mềm khác mà thường chỉ có sẵn trên Linux. Điều này rất hữu ích đối với các nhà phát triển web và phần mềm, những người cần phải phát triển trên môi trường Linux mà không muốn chuyển sang hệ điều hành khác. WSL có hai phiên bản: WSL 1 và WSL 2. WSL 1 tạo ra một lớp tương thích giữa Windows và Linux, cho phép chạy các lệnh Linux trực tiếp trên Windows mà không cần máy ảo. WSL 2 cải thiện hiệu suất với một nhân Linux thực sự, giúp hỗ trợ đầy đủ các hệ thống file của Linux và tăng tốc độ đáng kể, đặc biệt là đối với các ứng dụng yêu cầu hệ thống file hoặc nhân Linux đầy đủ.

Amazon Web Services (AWS):

- Amazon Web Services (AWS) là nền tảng dịch vụ đám mây của Amazon, cung cấp các dịch vụ như tính toán, lưu trữ, cơ sở dữ liệu, phân tích dữ liệu và các công cụ phát triển khác. AWS giúp các công ty và lập trình viên xây dựng và vận hành các ứng dụng mà không cần phải quản lý cơ sở hạ tầng vật lý. AWS cung cấp các dịch vụ đám mây quy mô lớn, đáng tin cậy, bảo mật và chi phí hiệu quả, phục vụ nhiều loại ứng dụng từ ứng dụng web đến ứng dụng di động, từ cơ sở hạ tầng IT cho đến AI và học máy. Một số dịch vụ chính của AWS bao gồm Amazon EC2 (Elastic Compute Cloud), Amazon S3 (Simple Storage

Service), AWS Lambda, và Amazon RDS (Relational Database Service). Amazon EC2 cung cấp các máy ảo để bạn chạy ứng dụng, server và dịch vụ của mình. Amazon S3 là dịch vụ lưu trữ dữ liệu đám mây có khả năng mở rộng cao, giúp bạn lưu trữ và truy xuất dữ liệu một cách dễ dàng. AWS Lambda là dịch vụ tính toán không máy chủ, cho phép bạn chạy mã mà không cần phải quản lý máy chủ. Amazon RDS cung cấp các dịch vụ cơ sở dữ liệu như MySQL, PostgreSQL, và Oracle trên đám mây, giúp việc quản lý cơ sở dữ liệu trở nên dễ dàng và tiết kiệm chi phí. AWS có mô hình thanh toán theo mức sử dụng, giúp các doanh nghiệp chỉ trả tiền cho những dịch vụ mà họ sử dụng, từ đó tối ưu hóa chi phí và tài nguyên cho các công ty và tổ chức.

Deploy: Amazon EC2 và VPS:

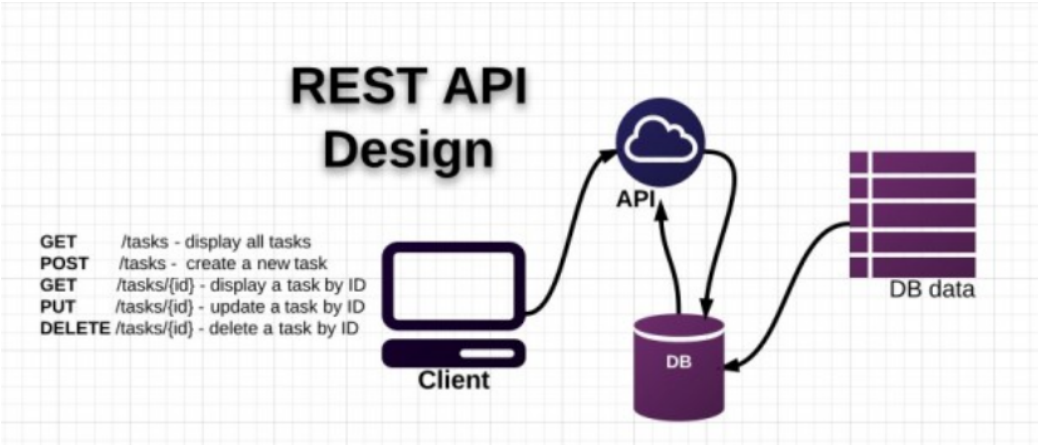
- **Amazon EC2:** Amazon EC2 là dịch vụ đám mây của Amazon Web Services (AWS), cho phép người dùng thuê máy chủ ảo (instances) để triển khai và chạy ứng dụng. EC2 mang lại sự linh hoạt, khả năng mở rộng và tính sẵn sàng cao cho các ứng dụng web. Với Clone Spotify, EC2 sẽ được sử dụng để triển khai phần backend và giúp ứng dụng có thể mở rộng khi lượng người dùng tăng.

- **VPS:** VPS là một máy chủ ảo được phân chia từ một máy chủ vật lý duy nhất. Mỗi VPS có hệ điều hành riêng biệt, giúp triển khai ứng dụng và quản lý tài nguyên như một máy chủ độc lập. Với Clone Spotify, VPS sẽ được sử dụng cho việc triển khai các dịch vụ khác ngoài backend, như các ứng dụng phụ trợ hoặc lưu trữ dữ liệu không đụng đến hệ thống chính.

3.3 Kiến trúc phần mềm

RESTful API:

- **Khái niệm:** RESTful API là một tiêu chuẩn được sử dụng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) nhằm mục đích quản lý các tài nguyên (resource) một cách hiệu quả. Nó tập trung vào các tài nguyên hệ thống như tệp văn bản, hình ảnh, âm thanh, video, hoặc dữ liệu động, bao gồm các trạng thái tài nguyên được định dạng và truyền tải thông qua giao thức HTTP. RESTful API cho phép các ứng dụng giao tiếp với nhau thông qua các phương thức HTTP chuẩn, giúp đơn giản hóa việc truy cập và xử lý dữ liệu trên các nền tảng khác nhau.



Diễn giải các thành phần

- **API (Application Programming Interface):** API là một tập hợp các quy tắc và cơ chế cho phép một ứng dụng hoặc thành phần tương tác với một ứng dụng hoặc thành phần khác. API đóng vai trò như một cầu nối, giúp các ứng dụng trao đổi dữ liệu với nhau một cách dễ dàng. Dữ liệu trả về từ API thường được định dạng ở các kiểu phổ biến như JSON hoặc XML, phù hợp để tích hợp vào các ứng dụng web hoặc di động.

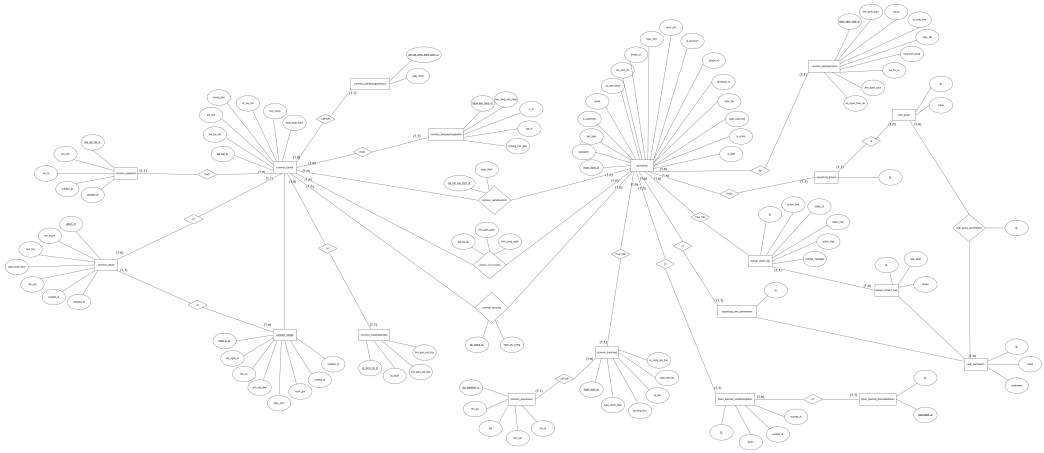
- **REST (REpresentational State Transfer):** REST là một kiểu kiến trúc (architectural style) được sử dụng để thiết kế API, dựa trên việc chuyển đổi trạng thái biểu diễn của tài nguyên. REST tận dụng các phương thức HTTP đơn giản như GET, POST, PUT, DELETE để thực hiện các thao tác trên tài nguyên. Thay vì sử dụng một URL để xử lý thông tin phức tạp, REST gửi các yêu cầu HTTP đến một URL

cụ thể nhằm truy xuất hoặc chỉnh sửa dữ liệu, giúp giao tiếp giữa các hệ thống trở nên trực quan và hiệu quả.

- **RESTful API:** RESTful API là một tiêu chuẩn thiết kế API cho các ứng dụng web, tập trung vào việc quản lý các tài nguyên (resource) và cho phép các ứng dụng khác nhau (web, mobile, v.v.) giao tiếp với nhau. Chức năng quan trọng nhất của RESTful API là quy định cách sử dụng các phương thức HTTP (như GET để lấy dữ liệu, POST để tạo mới, PUT để cập nhật, DELETE để xóa) và cách định dạng URL để truy cập các tài nguyên. RESTful API không quy định logic mã nguồn của ứng dụng và không bị giới hạn bởi ngôn ngữ lập trình, do đó bất kỳ ngôn ngữ hoặc framework nào (như Java với Spring Boot, JavaScript với Node.js) cũng có thể được sử dụng để thiết kế một RESTful API.

Chương 4

SƠ ĐỒ ERD



4.1 Mô hình Dữ liệu

Dưới đây là các mô hình dữ liệu chính trong hệ thống:

4.1.1 Mô hình Người Dùng (NguoiDung)

Mô hình `NguoiDung` lưu trữ thông tin về người dùng của hệ thống. Các trường chính của entity này bao gồm:

- **nguoi_dung_id**: Khóa chính, định danh duy nhất cho mỗi người dùng.
- **email**: Địa chỉ email của người dùng, được sử dụng để đăng nhập.
- **so_dien_thoai**: Số điện thoại của người dùng.

- **ten_hien_thi**: Tên hiển thị của người dùng trên hệ thống.
- **gioi_tinh**: Giới tính của người dùng, có thể là "Nam"hoặc "Nữ".
- **avatar_url**: URL của ảnh đại diện người dùng.
- **ngay_sinh**: Ngày sinh của người dùng.
- **quoc_gia**: Quốc gia của người dùng.
- **la_premium**: Trường boolean xác định xem người dùng có phải là người dùng Premium không.
- **google_id** và **facebook_id**: Các trường chứa ID của người dùng từ Google hoặc Facebook (nếu đăng nhập qua các nền tảng này).
- **ngay_tao** và **ngay_cap_nhat**: Thời gian tạo và cập nhật thông tin người dùng.
- **is_active** và **is_staff**: Các trường quản lý trạng thái hoạt động và quyền hạn của người dùng (staff).

Chức năng của mô hình này là quản lý thông tin người dùng, cung cấp cơ sở dữ liệu cho việc đăng nhập, tạo và cập nhật thông tin người dùng, cũng như phân quyền cho người dùng.

4.1.2 Mô hình Nghệ Sĩ (NgheSi)

Mô hình `NgheSi` lưu trữ thông tin về nghệ sĩ. Các trường chính bao gồm:

- **nghe_si_id**: Khóa chính, định danh nghệ sĩ.
- **ten_nghe_si**: Tên nghệ sĩ.
- **tieu_su**: Thông tin mô tả về nghệ sĩ.
- **anh_dai_dien**: URL của ảnh đại diện nghệ sĩ.
- **ngay_sinh**: Ngày sinh của nghệ sĩ.
- **quoc_gia**: Quốc gia của nghệ sĩ.
- **is_active**: Trạng thái hoạt động của nghệ sĩ.
- **created_at** và **updated_at**: Thời gian tạo và cập nhật thông tin nghệ sĩ.

Chức năng của mô hình này là lưu trữ và quản lý thông tin của các nghệ sĩ, giúp người dùng có thể tìm kiếm và theo dõi các nghệ sĩ yêu thích.

4.1.3 Mô hình Album

Mô hình `Album` lưu trữ thông tin về các album nhạc. Các trường chính bao gồm:

- **album_id**: Khóa chính, định danh album.
- **ten_album**: Tên album.
- **nghe_si**: Khóa ngoại liên kết đến mô hình `NgheSi`, chỉ ra nghệ sĩ sở hữu album.
- **anh_bia**: URL của ảnh bìa album.
- **ngay_phat_hanh**: Ngày phát hành album.
- **the_loai**: Thể loại của album.
- **is_active**: Trạng thái hoạt động của album.
- **created_at** và **updated_at**: Thời gian tạo và cập nhật album.

Chức năng của mô hình này là quản lý thông tin về các album mà nghệ sĩ phát hành, giúp người dùng dễ dàng tìm kiếm và nghe nhạc theo album.

4.1.4 Mô hình Bài Hát (BaiHat)

Mô hình `BaiHat` lưu trữ thông tin về các bài hát trong hệ thống. Các trường chính bao gồm:

- **bai_hat_id**: Khóa chính, định danh bài hát.
- **ten_bai_hat**: Tên bài hát.
- **nghe_si**: Khóa ngoại liên kết đến mô hình `NgheSi`, chỉ ra nghệ sĩ thể hiện bài hát.
- **album**: Khóa ngoại liên kết đến mô hình `Album`, chỉ ra album mà bài hát thuộc về.
- **the_loai**: Thể loại của bài hát.
- **file_bai_hat**: Đường dẫn đến tệp bài hát (mp3, mp4, v.v.).
- **duong_dan**: URL của bài hát, giúp người dùng phát bài hát từ hệ thống.

- **loi_bai_hat**: Lời bài hát.
- **thoi_luong**: Thời gian bài hát (tính bằng giây).
- **ngay_phat_hanh**: Ngày phát hành bài hát.

Chức năng của mô hình này là lưu trữ thông tin bài hát, bao gồm các tệp âm thanh và lời bài hát, giúp người dùng nghe và tìm kiếm bài hát.

4.1.5 Mô hình Danh Sách Phát (DanhSachPhat)

Mô hình `DanhSachPhat` lưu trữ thông tin về các danh sách phát nhạc của người dùng. Các trường chính bao gồm:

- **danh_sach_phat_id**: Khóa chính, định danh danh sách phát.
- **nguoi_dung_id**: Khóa ngoại liên kết đến mô hình `NguoiDung`, chỉ ra người dùng sở hữu danh sách phát.
- **ten_danh_sach**: Tên danh sách phát.
- **mo_ta**: Mô tả về danh sách phát.
- **la_cong_khai**: Trạng thái công khai của danh sách phát.
- **ngay_tao**: Thời gian tạo danh sách phát.
- **tong_thoi_luong**: Tổng thời gian của tất cả bài hát trong danh sách phát (tính bằng giây).
- **so_thu_tu**: Thứ tự hiển thị của danh sách phát.
- **anh_danh_sach**: URL của ảnh đại diện cho danh sách phát.
- **so_nguoi_theo_doi**: Số người theo dõi danh sách phát.

Chức năng của mô hình này là quản lý các danh sách phát của người dùng, giúp người dùng tạo và chia sẻ các danh sách phát nhạc.

4.1.6 Mô hình Thanh Toán (ThanhToan)

Mô hình `ThanhToan` lưu trữ thông tin về các giao dịch thanh toán của người dùng đối với các gói Premium. Các trường chính bao gồm:

- **thanh_toan_id**: Khóa chính, định danh giao dịch thanh toán.

- **nguoi_dung_id**: Khóa ngoại liên kết đến mô hình `NguoiDung`, chỉ ra người dùng thực hiện giao dịch.
- **so_tien**: Số tiền đã thanh toán.
- **loai_thanh_toan**: Loại giao dịch (ví dụ: thẻ tín dụng, PayPal).
- **ngay_thanh_toan**: Ngày thanh toán.
- **trang_thai_thanh_toan**: Trạng thái của giao dịch (thành công, thất bại).

Chức năng của mô hình này là quản lý thông tin giao dịch thanh toán của người dùng để cấp quyền Premium cho người dùng.

Chương 5

BÁO CÁO KẾT QUẢ

Chương 6

HƯỚNG PHÁT TRIỂN

- **Tích hợp các dịch vụ phát nhạc trực tuyến**: Tích hợp các dịch vụ phát nhạc trực tuyến để cung cấp nguồn nhạc phong phú và chất lượng cao.
- **Cải thiện trải nghiệm người dùng (UX/UI)**: Cải thiện trải nghiệm người dùng (UX/UI): Thiết kế giao diện hấp dẫn, thân thiện và dễ sử dụng hơn, đồng thời tối ưu hóa hiệu suất để người dùng có trải nghiệm mượt mà.
- **Phát triển các tính năng xã hội**: Phát triển các tính năng xã hội: Cho phép người dùng chia sẻ playlist, theo dõi bạn bè và xem hoạt động nghe nhạc của họ để tạo sự kết nối trong cộng đồng người dùng.
- **Hỗ trợ đa nền tảng**: Hỗ trợ đa nền tảng: Đảm bảo ứng dụng hoạt động trên nhiều thiết bị như iPad, di động và máy tính để bàn, đồng thời đồng bộ hóa dữ liệu người dùng giữa các thiết bị.
- **Tối ưu hóa cho di động**: Tối ưu hóa cho di động: Đảm bảo ứng dụng di động có hiệu suất tốt, giao diện thân thiện và tiết kiệm pin, mang lại trải nghiệm người dùng tối ưu trên các thiết bị di động.
- **Phân tích và đề xuất nhạc thông minh**: Phân tích và đề xuất nhạc thông minh: Sử dụng trí tuệ nhân tạo để phân tích thói quen nghe nhạc của người dùng và đề xuất các bài hát, playlist phù hợp, tạo sự cá nhân hóa trong trải nghiệm người dùng.
- **Đảm bảo tuân thủ bản quyền**: Đảm bảo tuân thủ bản quyền: Thiết lập các thỏa thuận với các nhà cung cấp nội dung và tuân thủ quy định về bản quyền âm nhạc để tránh các vấn đề pháp lý liên quan.