

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

Xây dựng ứng dụng

Nghe nhạc, Quản lý & Chat Django

GVHD: Từ Lăng Phiêu
SV: Mai Thành Đạt - 3121410143

TP. HỒ CHÍ MINH, THÁNG 5/2025

Mục lục

1	Mở đầu	2
1.1	Giới thiệu đề tài	2
1.2	Lý do chọn đề tài	2
1.3	Mục tiêu	2
1.4	Phạm vi	3
1.5	Tính ứng dụng	3
2	Phân tích yêu cầu	4
2.1	Yêu cầu chức năng	4
2.2	Yêu cầu phi chức năng	4
2.3	Yêu cầu kĩ thuật	4
3	Công nghệ áp dụng	5
3.1	Django	5
3.2	Django REST Framework (DRF)	5
3.3	Django Channels	6
3.4	ReactJS	7
3.5	Cloudinary	7
4	Phân tích và thiết kế	9
4.1	Cơ sở dữ liệu	9
4.1.1	ERD	9
4.1.2	Vật lý	9
4.2	Phân chia hệ thống	13
4.3	Mô tả chi tiết các ứng dụng con	13
4.3.1	Authentication	13
4.3.2	Relationship	14
4.3.3	Profiles	14
4.3.4	Chat	14
4.4	Mô hình hoạt động của ứng dụng	14
5	Thực nghiệm và phân tích kết quả	16
5.1	Đăng nhập vào hệ thống	16
5.2	Đăng ký tài khoản	16
5.3	Chức năng quên mật khẩu	17
5.4	Chức năng đổi mật khẩu mới	18
5.5	Giao diện màn hình chính	18
5.6	Chức năng phát nhạc	19
5.7	Chức năng thêm danh sách yêu thích (song, album, artist)	19
5.8	Chức năng xem video	19
5.9	Chức năng bạn bè	19
5.10	Chức năng chat	21
6	Cách thức cài đặt, môi trường cài ứng dụng	24
7	Kết luận và hướng phát triển	26
7.1	Kết luận	26
7.2	Hướng phát triển	26

1 Mở đầu

1.1 Giới thiệu đề tài

Với sự phát triển không ngừng của công nghệ, nhu cầu giải trí, quản lý nội dung cá nhân và giao tiếp trực tuyến của người dùng ngày càng cao. Tuy nhiên, nhiều ứng dụng hiện nay vẫn chưa tích hợp đầy đủ các tính năng cần thiết, hoặc chưa mang lại trải nghiệm người dùng tối ưu về mặt giao diện, khả năng tùy chỉnh và bảo mật thông tin. Nhận thấy được điều đó, đề tài xây dựng ứng dụng nghe nhạc, quản lý và trò chuyện được triển khai nhằm mục đích tạo ra một nền tảng tiện ích, nơi người dùng không chỉ có thể thưởng thức âm nhạc yêu thích mà còn dễ dàng quản lý danh sách bài hát, tạo playlist cá nhân và trò chuyện, chia sẻ cảm xúc với bạn bè một cách nhanh chóng và an toàn. Đề tài hướng đến việc phát triển một hệ sinh thái giải trí – giao tiếp toàn diện, giúp người dùng kết nối, thư giãn và quản lý nội dung của mình mọi lúc, mọi nơi.

1.2 Lý do chọn đề tài

Trong thời đại số hiện nay, nhu cầu giải trí và kết nối cộng đồng thông qua các nền tảng trực tuyến ngày càng trở nên phổ biến. Tuy nhiên, người dùng thường phải sử dụng nhiều ứng dụng riêng biệt để phục vụ cho từng nhu cầu như nghe nhạc, quản lý nội dung cá nhân hay trò chuyện với bạn bè. Điều này gây bất tiện và thiếu tính đồng bộ trong trải nghiệm. Xuất phát từ thực tế đó, nhóm chúng tôi lựa chọn đề tài xây dựng một ứng dụng tích hợp các chức năng nghe nhạc, quản lý cá nhân và trò chuyện trực tuyến nhằm mang đến một giải pháp toàn diện, đáp ứng đầy đủ nhu cầu giải trí và kết nối trong cùng một nền tảng. Đây không chỉ là một xu hướng thiết yếu mà còn là bước đi thực tế để nâng cao trải nghiệm người dùng và bắt kịp với sự phát triển của công nghệ hiện đại.

1.3 Mục tiêu

- Tạo ra một nền tảng tích hợp giữa giải trí (nghe nhạc), quản lý cá nhân (playlist, bài hát yêu thích) và giao tiếp trực tuyến (chat, chia sẻ).
- Cung cấp chức năng nghe nhạc trực tuyến chất lượng cao, hỗ trợ tìm kiếm và gợi ý bài hát theo sở thích người dùng.
- Hỗ trợ người dùng tạo và quản lý danh sách phát cá nhân, lịch sử nghe và các bài hát đã lưu.
- Cho phép người dùng trò chuyện trực tuyến, nhắn tin nhóm, chia sẻ bài hát, hình ảnh và tệp tin với bạn bè.
- Thiết kế giao diện hiện đại, trực quan, thân thiện với mọi đối tượng người dùng.
- Đảm bảo bảo mật thông tin cá nhân và dữ liệu người dùng, đồng thời tối ưu hiệu năng hoạt động trên cả web và thiết bị di động.
- Ứng dụng các công nghệ mới để nâng cao tốc độ xử lý, khả năng mở rộng và tính ổn định của hệ thống.



1.4 Phạm vi

Website đáp ứng các nhu cầu cơ bản và thiết yếu của một nền tảng giải trí và giao tiếp trực tuyến, bao gồm: nghe nhạc, quản lý nội dung cá nhân và trò chuyện với người dùng khác. Đối tượng sử dụng hướng đến là mọi cá nhân yêu thích âm nhạc và mong muốn có một nền tảng tích hợp để vừa thưởng thức âm nhạc, vừa có thể chia sẻ và tương tác với bạn bè, người thân hoặc đồng nghiệp. Ngoài ra, website cũng phù hợp với các nhóm nhỏ, tổ chức cần một công cụ giao tiếp nội bộ kết hợp với giải trí nhẹ nhàng.

1.5 Tính ứng dụng

Website tích hợp nghe nhạc, quản lý và trò chuyện mang lại nhiều tính ứng dụng thiết thực, phục vụ đa dạng nhu cầu trong học tập, làm việc và giải trí hằng ngày.

- Giải trí cá nhân: Người dùng có thể nghe nhạc trực tuyến mọi lúc, mọi nơi với chất lượng cao; tạo và quản lý danh sách phát theo sở thích cá nhân; chia sẻ bài hát yêu thích đến bạn bè.
- Kết nối và tương tác xã hội: Trò chuyện, chia sẻ nội dung âm nhạc, hình ảnh, trạng thái và tệp tin với bạn bè, người thân; phù hợp với việc giao lưu, tâm sự, học tập hay làm việc nhóm từ xa.
- Giao tiếp nội bộ trong tổ chức: Có thể được sử dụng làm công cụ liên lạc nội bộ cho các nhóm làm việc, tổ chức nhỏ với chức năng chat, chia sẻ tài liệu, liên kết qua âm nhạc để tạo không khí làm việc thoải mái, sáng tạo.

2 Phân tích yêu cầu

2.1 Yêu cầu chức năng

- Phát nhạc trực tuyến: Người dùng có thể nghe nhạc trực tiếp từ nền tảng với chất lượng âm thanh cao, hỗ trợ các thao tác điều khiển như phát/tạm dừng, chuyển bài, lặp lại, phát ngẫu nhiên,...
- Phát video âm nhạc: Ứng dụng hỗ trợ trình phát video nhạc, cho phép người dùng xem các MV ca nhạc, biểu diễn trực tiếp hoặc video liên quan đến âm nhạc ngay trong giao diện web.
- Tải video âm nhạc: Người dùng có thể tải về các video âm nhạc được cho phép để nghe/xem ngoại tuyến, phục vụ nhu cầu giải trí không cần kết nối Internet.
- Tạo album và danh sách yêu thích: Người dùng có thể tạo album nhạc cá nhân, thêm bài hát vào danh sách yêu thích hoặc danh sách phát tùy chỉnh nhằm cá nhân hóa trải nghiệm nghe nhạc.
- Quản trị hệ thống (Admin): Trang quản trị dành cho admin với các chức năng như quản lý người dùng, bài hát, video, báo cáo vi phạm, duyệt nội dung và kiểm soát hoạt động hệ thống.
- (Tùy chọn) Tính năng chat tích hợp: Hệ thống có thể tích hợp thêm chức năng nhắn tin văn bản giữa người dùng, giúp tăng khả năng tương tác và chia sẻ trải nghiệm âm nhạc trực tiếp trên nền tảng.

2.2 Yêu cầu phi chức năng

- Bảo mật: Đảm bảo bảo mật cao cho dữ liệu người dùng, bao gồm mã hóa thông tin cá nhân, bảo vệ quyền riêng tư và sử dụng các cơ chế xác thực mạnh mẽ để ngăn chặn truy cập trái phép.
- Hiệu suất: Ứng dụng cần có hiệu suất ổn định, cho phép phát nhạc và video mượt mà, không gián đoạn, đồng thời đảm bảo các chức năng chat và quản lý nội dung hoạt động nhanh chóng và hiệu quả.
- Khả năng mở rộng: Hệ thống phải được thiết kế để dễ dàng mở rộng, hỗ trợ tăng trưởng người dùng và tính năng trong tương lai mà không ảnh hưởng đến hiệu suất hay chất lượng dịch vụ.
- Tương thích đa nền tảng: Ứng dụng phải hoạt động tốt trên các nền tảng web, Android, iOS và các hệ điều hành máy tính như Windows và macOS, nhằm tiếp cận và phục vụ người dùng trên nhiều thiết bị khác nhau.

2.3 Yêu cầu kỹ thuật

- Công cụ sử dụng: Visual Studio Code, Laragon, Postman, Git
- Frontend: ReactJs
- Backend: Django, Django Rest Framework, Django Channel.
- Database: MySQL.

3 Công nghệ áp dụng

3.1 Django

Django là một framework phát triển web mạnh mẽ được xây dựng bằng ngôn ngữ Python. Nó được thiết kế để giúp các nhà phát triển xây dựng các ứng dụng web phức tạp một cách nhanh chóng và dễ dàng, bằng cách cung cấp các công cụ và thư viện mạnh mẽ.

- ORM(Object-Relational Mapping): Django cung cấp một ORM mạnh mẽ, cho phép nhà phát triển tương tác với cơ sở dữ liệu thông qua các đối tượng Python, thay vì việc viết các truy vấn SQL trực tiếp. Điều này giúp giảm sự phụ thuộc vào cơ sở dữ liệu cụ thể và làm cho việc thay đổi cơ sở dữ liệu trở nên dễ dàng hơn.
- URL Routing: Django sử dụng một hệ thống routing mạnh mẽ để ánh xạ các URL vào các view xử lý tương ứng. Điều này giúp tổ chức code của bạn và cho phép bạn dễ dàng mở rộng ứng dụng của mình.
- Admin interface tự động: Django cung cấp một giao diện quản trị tự động được tạo dựa trên mô hình dữ liệu được xác định trong ứng dụng của bạn. Điều này giúp bạn quản lý và thao tác với dữ liệu một cách dễ dàng mà không cần phải viết code cho phần quản trị.
- Bảo mật: Django đi kèm với nhiều tính năng bảo mật tích hợp sẵn như bảo vệ chống CSRF (Cross-Site Request Forgery), bảo vệ chống XSS (Cross-Site Scripting), và quản lý phiên.
- Cộng đồng và tài liệu: Django có một cộng đồng lớn và năng động, với nhiều tài liệu, hướng dẫn và các thư viện bổ sung giúp bạn dễ dàng học và sử dụng.

3.2 Django REST Framework (DRF)

Django REST Framework (DRF) là một bộ công cụ mở rộng Django, được thiết kế để giúp việc xây dựng các API RESTful trở nên dễ dàng và mạnh mẽ hơn. Được xây dựng trên cơ sở của Django, DRF cung cấp các tính năng và công cụ cho việc xử lý dữ liệu JSON, quản lý xác thực, kiểm tra quyền truy cập, và nhiều hơn nữa.

Dưới đây là một số tính năng chính của Django REST Framework:

- Serializers: DRF cung cấp một cơ chế mạnh mẽ để chuyển đổi các đối tượng Python sang JSON và ngược lại thông qua serializers. Serializers giúp xác định các trường dữ liệu cần được bao gồm hoặc loại bỏ khi trả về dữ liệu qua API, cũng như xử lý validation của dữ liệu đầu vào.
- Viewsets và Routers: DRF giúp tổ chức code của bạn thông qua việc sử dụng viewsets và routers. Viewsets cho phép bạn xác định tất cả các phương thức API (như GET, POST, PUT, DELETE) cho một tập hợp các đối tượng, trong khi routers tự động xác định các URL cho các viewsets của bạn.
- Authentication và Permissions: DRF cung cấp các lớp cho xác thực và kiểm tra quyền truy cập, cho phép bạn xác định ai có quyền truy cập vào các API của bạn và những gì họ có thể làm.

- **Throttling:** Để ngăn chặn các cuộc tấn công DDOS (Distributed Denial of Service) hoặc giới hạn số lượng yêu cầu mà một người dùng có thể thực hiện trong một khoảng thời gian nhất định, DRF cung cấp các cơ chế throttling.
- **Pagination:** DRF cho phép bạn dễ dàng thực hiện phân trang cho kết quả trả về từ các API, giúp quản lý tải trang và tối ưu hóa hiệu suất.
- **Tích hợp tốt với Django:** DRF được thiết kế để tích hợp tốt với Django, cho phép bạn sử dụng các tính năng của Django như ORM, authentication backend, và middleware một cách dễ dàng.
- **Tài liệu và hỗ trợ:** Django REST Framework có tài liệu phong phú và cộng đồng hỗ trợ lớn, bao gồm các ví dụ, hướng dẫn và diễn đàn để giúp bạn giải quyết các vấn đề phát triển.

3.3 Django Channels

Django Channels là một thư viện mở rộng cho Django, cho phép bạn xây dựng ứng dụng web theo mô hình real-time và các ứng dụng có khả năng xử lý các sự kiện đồng thời (concurrent events). Trong Django truyền thống, các request được xử lý tuần tự, điều này có nghĩa là mỗi request phải chờ đợi cho đến khi request trước đó hoàn thành trước khi được xử lý. Tuy nhiên, với Django Channels, bạn có thể xử lý các sự kiện đồng thời mà không cần phải chờ đợi.

Dưới đây là một số tính năng chính của Django Channels:

- **Real-time:** Django Channels cho phép bạn xây dựng các ứng dụng real-time như chat, trò chơi trực tuyến, thông báo tức thời và nhiều ứng dụng khác mà yêu cầu giao tiếp đồng thời giữa máy chủ và trình duyệt.
- **Websockets:** Thư viện này hỗ trợ giao thức Websockets, cho phép máy chủ gửi dữ liệu tới trình duyệt mà không cần yêu cầu mới từ phía client. Điều này giúp giảm độ trễ và tăng tính tương tác trong ứng dụng.
- **Protocol support:** Ngoài Websockets, Django Channels còn hỗ trợ một loạt các giao thức như HTTP long-polling, HTTP2, và MQTT (Message Queuing Telemetry Transport).
- **Routing và Consumers:** Channels sử dụng các "consumers" để xử lý các message được gửi tới từ client, và các "routers" để xác định cách xử lý các message dựa trên loại và nội dung của chúng.
- **Asynchronous I/O:** Channels hỗ trợ việc sử dụng các coroutine và async/await trong Python để xử lý các tác vụ I/O mà không cần chờ đợi, giúp tối ưu hiệu suất của ứng dụng trong các trường hợp có nhiều kết nối đồng thời.
- **Integrations:** Django Channels tích hợp tốt với Django, cho phép bạn sử dụng các tính năng của Django như middleware, authentication, và ORM trong các ứng dụng Channels.

Tóm lại, Django Channels là một công cụ mạnh mẽ cho việc phát triển các ứng dụng real-time và có khả năng xử lý sự kiện đồng thời trong Django. Nó mở ra khả năng xây dựng các ứng dụng web tương tác cao và cung cấp trải nghiệm người dùng tốt hơn.

3.4 ReactJS

ReactJS là một thư viện JavaScript phổ biến được phát triển bởi Facebook, được sử dụng để xây dựng giao diện người dùng (UI) cho các ứng dụng web hiệu quả và linh hoạt. Đặc điểm chính của ReactJS là việc sử dụng một cách tiếp cận gọi là "component-based", trong đó UI được chia thành các thành phần độc lập, tái sử dụng và dễ quản lý.

Dưới đây là một số điểm nổi bật của ReactJS:

- **Component-Based:** ReactJS cho phép bạn xây dựng UI bằng cách tạo ra các thành phần (components) độc lập. Mỗi thành phần có thể chứa mã HTML, CSS và JavaScript của riêng nó, giúp tăng tính tái sử dụng và dễ dàng quản lý mã nguồn.
- **Virtual DOM:** ReactJS sử dụng một cơ chế gọi là Virtual DOM để tối ưu hóa việc cập nhật UI. Thay vì cập nhật trực tiếp DOM mỗi khi có thay đổi, React tạo ra một bản sao của DOM (Virtual DOM) và so sánh nó với DOM thực tế. Sau đó, React chỉ cập nhật những phần của DOM mà thực sự thay đổi, giúp tăng hiệu suất và đáp ứng nhanh hơn.
- **One-Way Data Binding:** ReactJS sử dụng mô hình one-way data binding, có nghĩa là dữ liệu chỉ di chuyển theo một hướng từ component cha đến các component con. Điều này giúp giảm sự phức tạp và dễ dàng theo dõi dòng dữ liệu trong ứng dụng.
- **JSX:** JSX là một phần mở rộng của JavaScript, cho phép bạn viết mã HTML trong JavaScript một cách dễ dàng và hiệu quả. JSX giúp tạo ra mã nguồn dễ đọc hơn và cho phép bạn kết hợp logic JavaScript và UI một cách mạch lạc.
- **Tích hợp tốt với các thư viện và các framework khác:** ReactJS có thể tích hợp dễ dàng với các thư viện và frameworks khác như Redux (quản lý trạng thái ứng dụng), React Router (định tuyến trong ứng dụng đơn trang), và các thư viện UI khác như Material-UI hoặc Ant Design.
- **Cộng đồng lớn và hỗ trợ mạnh mẽ:** ReactJS có một cộng đồng lớn và năng động, với nhiều tài liệu, hướng dẫn và ví dụ mã nguồn mở được chia sẻ trên Internet. Điều này làm cho việc học và phát triển ứng dụng React trở nên dễ dàng hơn.

Tóm lại, ReactJS là một thư viện JavaScript mạnh mẽ và linh hoạt, được sử dụng rộng rãi để xây dựng các ứng dụng web hiệu quả, linh hoạt và dễ bảo trì. Nó là lựa chọn phổ biến cho các nhà phát triển web khi muốn xây dựng các ứng dụng có giao diện người dùng phức tạp và tương tác.

Tóm lại, Ant Design là một thư viện giao diện người dùng (UI) mạnh mẽ và linh hoạt cho React, giúp bạn xây dựng các ứng dụng web có giao diện người dùng chuyên nghiệp và hiệu quả một cách nhanh chóng và dễ dàng.

3.5 Cloundinary

Cloundinary là một dịch vụ quản lý hình ảnh và video trên đám mây, cung cấp các công cụ và dịch vụ để tải lên, lưu trữ, xử lý và quản lý tài nguyên đa phương tiện của bạn trực tuyến. Được phát triển bởi công ty Cloundinary Ltd., nó được sử dụng rộng rãi trong việc xây dựng các ứng dụng web và di động để quản lý và tối ưu hóa hình ảnh và video.

Dưới đây là một số tính năng chính của Cloundinary:

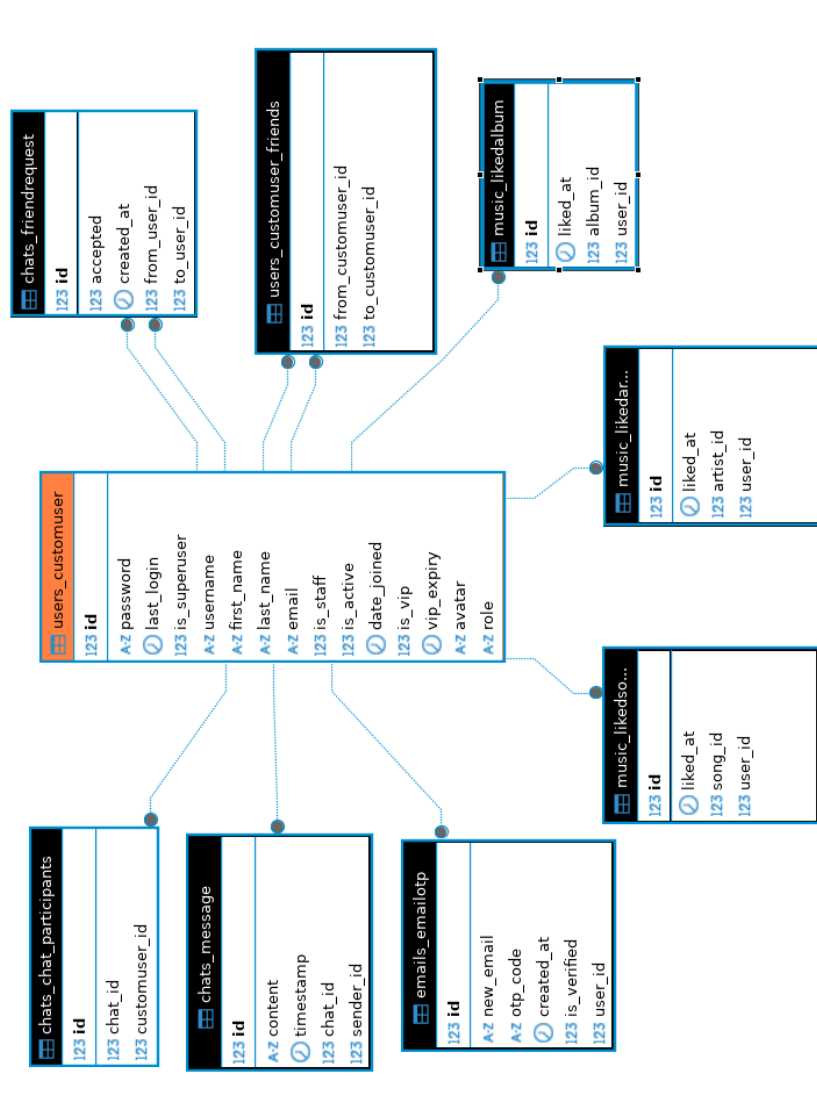
- **Tải lên và Lưu trữ:** Cloudinary cho phép bạn tải lên hình ảnh và video từ nhiều nguồn khác nhau, bao gồm cả máy tính cá nhân, thiết bị di động, và các dịch vụ lưu trữ đám mây khác như Amazon S3 và Google Cloud Storage. Hình ảnh và video được lưu trữ trên đám mây của Cloudinary, giúp giảm tải cho máy chủ của bạn.
- **Xử lý và Chỉnh sửa:** Cloudinary cung cấp một loạt các công cụ và API cho việc xử lý và chỉnh sửa hình ảnh và video. Bạn có thể thực hiện các thao tác như cắt, thay đổi kích thước, xoay, lật, mờ hoặc làm sáng hình ảnh, cũng như chuyển đổi định dạng hoặc tạo ảnh thumbnail một cách dễ dàng.
- **Tối ưu hóa và Giao diện tốc độ:** Cloudinary tự động tối ưu hóa hình ảnh và video của bạn để đảm bảo tải nhanh và hiệu suất cao trên các thiết bị và mạng khác nhau. Điều này giúp cải thiện trải nghiệm người dùng và tăng tỷ lệ chuyển đổi trên ứng dụng web và di động của bạn.
- **Quản lý Tài nguyên:** Cloudinary cung cấp các công cụ và giao diện quản lý để giúp bạn tổ chức và quản lý tài nguyên đa phương tiện của mình một cách hiệu quả. Bạn có thể tìm kiếm, lọc, sắp xếp và xóa tài nguyên một cách dễ dàng.
- **Phân phối và Giao diện tốc độ:** Cloudinary cung cấp các tính năng phân phối nội dung (CDN) để giúp tối ưu hóa tải nội dung đa phương tiện trên toàn cầu. Điều này giúp giảm độ trễ và tăng tốc độ tải trang cho người dùng trên toàn thế giới.
- **Tích hợp và Hỗ trợ:** Cloudinary có các thư viện và SDK để tích hợp dễ dàng vào các ứng dụng web và di động của bạn, bao gồm cả các thư viện JavaScript, PHP, Python, Ruby và nhiều ngôn ngữ khác. Hỗ trợ khách hàng đa kênh cũng được cung cấp để giúp bạn giải quyết các vấn đề kỹ thuật và triển khai.

Tóm lại, Cloudinary là một dịch vụ quản lý hình ảnh và video trên đám mây mạnh mẽ và linh hoạt, cung cấp các công cụ và dịch vụ để tải lên, lưu trữ, xử lý và quản lý tài nguyên đa phương tiện của bạn một cách dễ dàng và hiệu quả. Điều này giúp cải thiện trải nghiệm người dùng và tối ưu hóa hiệu suất của ứng dụng web và di động của bạn.

4 Phân tích và thiết kế

4.1 Cơ sở dữ liệu

4.1.1 ERD



Hình 1: Erd

4.1.2 Vật lý



STT	Tên bảng	Mô tả
1	user	Bảng thông tin người dùng
2	chat_participants	Lưu thông tin user nào trong phòng chat
3	chat_messege	Lưu tin nhắn, cũng như thông tin người gửi
4	chat_friendrequest	Lưu thông tin lời mời kết bạn
5	user_customuser_friend	Lưu thông tin quan hệ bạn bè giữa user
6	music_likealbum	Lưu thông tin album và user người thích album
7	music_likeartist	Lưu thông tin nghệ sĩ và user người thích nghệ sĩ
8	music_likesong	Lưu thông tin bài hát và user người thích bài hát
9	email_emailotp	Lưu thông tin về người yêu cầu otp

Bảng 1: Danh sách bảng trong cơ sở dữ liệu

Danh sách trên là một số bảng chính, ngoài ra còn có một số bảng cấu hình của framework Django. Sau đây là cấu trúc chi tiết từng bảng

Column Name	#	Data Type	Not Null	Auto Increment	Key
123 id	1	bigint	[v]	[v]	PRI
A-Z password	2	varchar(128)	[v]	[]	
🕒 last_login	3	datetime(6)	[]	[]	
123 is_superuser	4	tinyint(1)	[v]	[]	
A-Z username	5	varchar(150)	[v]	[]	UNI
A-Z first_name	6	varchar(150)	[v]	[]	
A-Z last_name	7	varchar(150)	[v]	[]	
A-Z email	8	varchar(254)	[v]	[]	
123 is_staff	9	tinyint(1)	[v]	[]	
123 is_active	10	tinyint(1)	[v]	[]	
🕒 date_joined	11	datetime(6)	[v]	[]	
123 is_vip	12	tinyint(1)	[v]	[]	
🕒 vip_expiry	13	datetime(6)	[]	[]	
A-Z avatar	14	varchar(100)	[]	[]	
A-Z role	15	varchar(10)	[v]	[]	

Hình 2: Bảng user



Column Name	#	Data Type	Not Null	Auto Increment	Key
123 id	1	bigint	[v]	[v]	PRI
123 customuser_i	3	bigint	[v]	[]	MUL
123 chat_id	2	bigint	[v]	[]	MUL

Hình 3: Bảng chat-participants

Column Name	#	Data Type	Not Null	Auto Increment	Key
123 id	1	bigint	[v]	[v]	PRI
A-Z content	2	longtext	[v]	[]	
🕒 timestamp	3	datetime(6)	[v]	[]	
123 chat_id	4	bigint	[v]	[]	MUL
123 sender_id	5	bigint	[v]	[]	MUL

Hình 4: Bảng chat-messege

Column Name	#	Data Type	Not Null	Auto Increment	Key
123 id	1	bigint	[v]	[v]	PRI
123 accepted	2	tinyint(1)	[v]	[]	
🕒 created_at	3	datetime(6)	[v]	[]	
123 from_user_id	4	bigint	[v]	[]	MUL
123 to_user_id	5	bigint	[v]	[]	MUL

Hình 5: Bảng chat-friendrequest

Column Name	#	Data Type	Not Null	Auto Increment	Key
123 id	1	bigint	[v]	[v]	PRI
123 from_customuser_id	2	bigint	[v]	[]	MUL
123 to_customuser_id	3	bigint	[v]	[]	MUL

Hình 6: Bảng user-customuser-friend

Column Name	#	Data Type	Not Null	Auto Increment	Key
123 id	1	bigint	[v]	[v]	PRI
🕒 liked_at	2	datetime(6)	[v]	[]	
123 album_id	3	bigint	[v]	[]	MUL
123 user_id	4	bigint	[v]	[]	MUL

Hình 7: Bảng music-likealbum



Column Name	#	Data Type	Not Null	Auto Increment	Key
123 id	1	bigint	[v]	[v]	PRI
🕒 liked_at	2	datetime(6)	[v]	[]	
123 artist_id	3	bigint	[v]	[]	MUL
123 user_id	4	bigint	[v]	[]	MUL

Hình 8: Bảng music-likeartist

Column Name	#	Data Type	Not Null	Auto Increment	Key
123 id	1	bigint	[v]	[v]	PRI
🕒 liked_at	2	datetime(6)	[v]	[]	
123 song_id	3	bigint	[v]	[]	MUL
123 user_id	4	bigint	[v]	[]	MUL

Hình 9: Bảng music-likesong

Column Name	#	Data Type	Not Null	Auto Increment	Key
123 id	1	bigint	[v]	[v]	PRI
A-Z new_email	2	varchar(254)	[v]	[]	
A-Z otp_code	3	varchar(6)	[v]	[]	
🕒 created_at	4	datetime(6)	[v]	[]	
123 is_verified	5	tinyint(1)	[v]	[]	
123 user_id	6	bigint	[v]	[]	MUL

Hình 10: Bảng email-emailotp

4.2 Phân chia hệ thống

Việc chia nhỏ hệ thống thành các ứng dụng con giúp cho việc quản lý dễ dàng hơn, đồng thời giúp nâng cao khả năng mở rộng và bảo trì. Sau khi phân tích, hệ thống sẽ được chia thành 5 ứng dụng con:

- **Authentication:** Xử lý các chức năng đăng ký, đăng nhập, đăng xuất và xác thực người dùng.
- **Relationship:** Quản lý mối quan hệ giữa người dùng, bao gồm chức năng gửi, chấp nhận và từ chối lời mời kết bạn.
- **Profiles:** Quản lý thông tin hồ sơ người dùng như ảnh đại diện, tên hiển thị, ngày tham gia, trạng thái VIP,...
- **Chức năng phát nhạc:** Người dùng có thể nghe các bài hát trực tuyến với giao diện điều khiển phát nhạc hiện đại.
- **Chức năng phát video âm nhạc:** Cho phép người dùng xem các video âm nhạc có sẵn trên hệ thống với trải nghiệm mượt mà.
- **Chức năng tải video âm nhạc:** Người dùng có thể tải các video âm nhạc về thiết bị cá nhân để xem ngoại tuyến.
- **Tạo album, danh sách yêu thích:** Người dùng có thể tự tạo album nhạc riêng, thêm bài hát vào danh sách yêu thích để tiện theo dõi.
- **Trang Admin:** Quản trị viên có thể quản lý người dùng, bài hát, video,...
- **Tính năng chat tích hợp (tuỳ chọn):** Giao diện giống Spotify nhưng được tích hợp thêm hệ thống chat realtime để người dùng vừa nghe nhạc vừa trò chuyện.

4.3 Mô tả chi tiết các ứng dụng con

4.3.1 Authentication

Ứng dụng sẽ xử lý các chức năng sau:

- Đăng ký tài khoản.
- Đăng nhập.
 - Đăng nhập bằng tài khoản người dùng.
 - Đăng nhập bằng tài khoản Google.
- Quên mật khẩu.
- Đổi mật khẩu.



4.3.2 Relationship

Ứng dụng sẽ xử lý các chức năng sau:

- Lấy danh sách bạn bè.
- Lấy danh sách lời mời kết bạn.
- Gửi lời mời kết bạn.
- Xóa / Hủy lời mời kết bạn.
- Chấp nhận lời mời kết bạn.
- Tìm kiếm bạn bè.

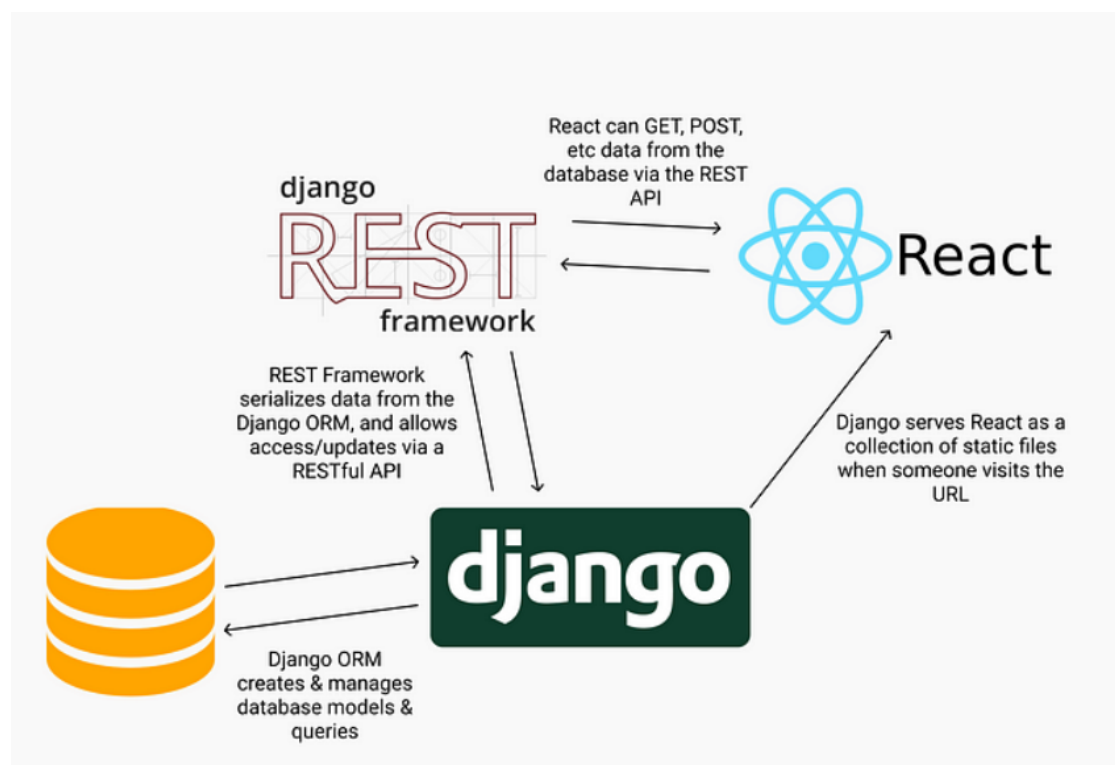
4.3.3 Profiles

- Lấy thông tin của một người dùng.
- Cập nhật thông tin người dùng.
- Cập nhật mật khẩu

4.3.4 Chat

- Tạo cuộc trò chuyện
- Lấy danh sách cuộc trò chuyện
- Lấy danh sách tin nhắn của cuộc trò chuyện
- Nhắn tin

4.4 Mô hình hoạt động của ứng dụng

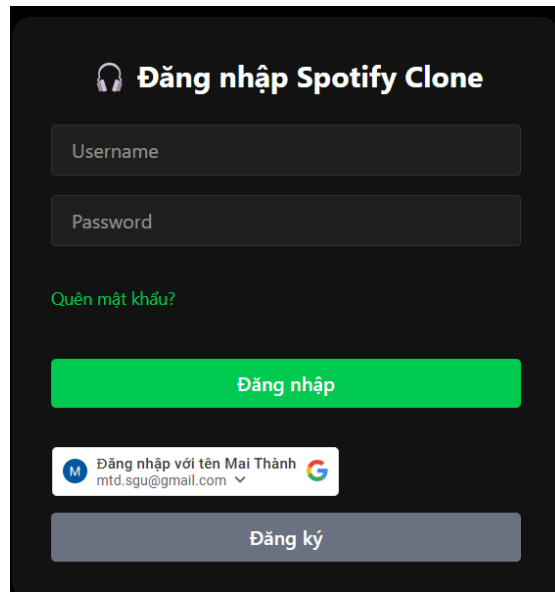


Hình 11: Mô hình hoạt động

5 Thực nghiệm và phân tích kết quả

5.1 Đăng nhập vào hệ thống

Người dùng thực hiện đăng ký tài khoản trước khi đăng nhập. Khi vào trang đăng nhập sẽ hiển thị màn hình đăng nhập như sau:



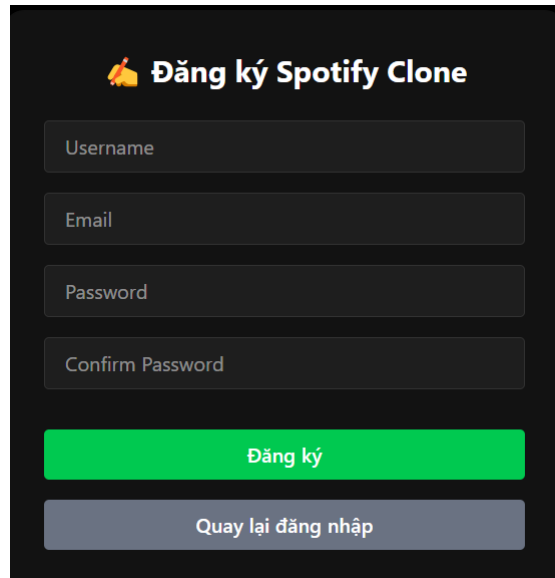
Hình 12: Giao diện đăng nhập

Người dùng có thể đăng nhập theo 2 cách:

- Cách 1: Sử dụng liên kết Google hoặc Github.
- Cách 2: Sử dụng email và password mà người dùng đã đăng ký trước với hệ thống đó.

5.2 Đăng ký tài khoản

Người dùng cần đăng ký tài khoản mới có thể đăng nhập vào hệ thống. Khi vào trang đăng ký sẽ hiển thị màn hình đăng ký như sau:



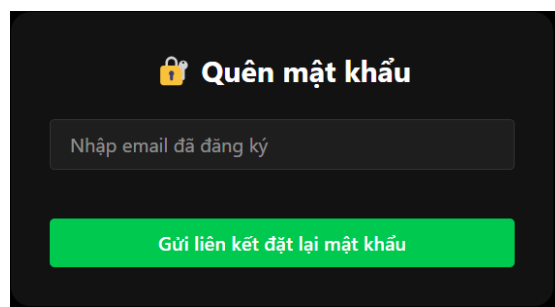
Hình 13: Giao diện đăng ký

Người dùng có thể đăng ký theo 2 cách:

- Cách 1: Sử dụng email và password để đăng ký tài khoản.

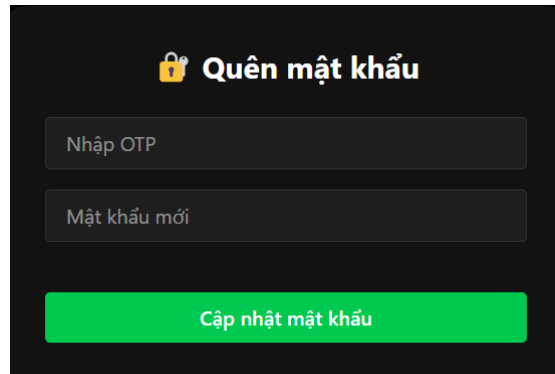
5.3 Chức năng quên mật khẩu

Khi quên mật khẩu, người dùng nhấn vào nút Quên mật khẩu ở trang đăng nhập, giao diện quên mật khẩu sẽ hiện ra như sau:



Hình 14: Giao diện quên mật khẩu

Người dùng tiến hành nhập email mà người dùng đã đăng ký tài khoản, sau đó nhấn nút Submit, hệ thống sẽ gửi email về cho người dùng một mã OTP. Người dùng tiến hành kiểm tra email và nhận OTP vào ô nhập OTP và mật khẩu mới để tiến hành đổi mật khẩu mới.

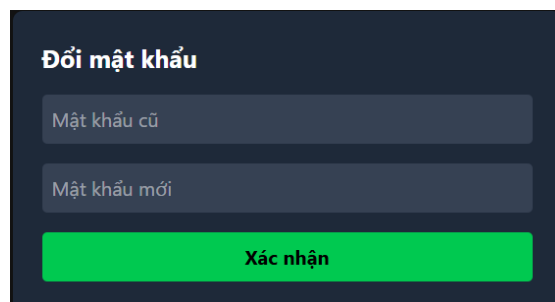


Hình 15: Giao diện nhập otp

Người dùng nhập mật khẩu mới và xác nhận mật khẩu mới sau đó nhấn nút Submit để cập nhật lại mật khẩu và được chuyển hướng đến trang Đăng nhập để đăng nhập.

5.4 Chức năng đổi mật khẩu mới

Khi muốn đổi mật khẩu mới, người dùng nhấn vào nút Settings, màn hình đổi mật khẩu sẽ hiển thị như sau:



Hình 16: Giao diện đổi mật khẩu

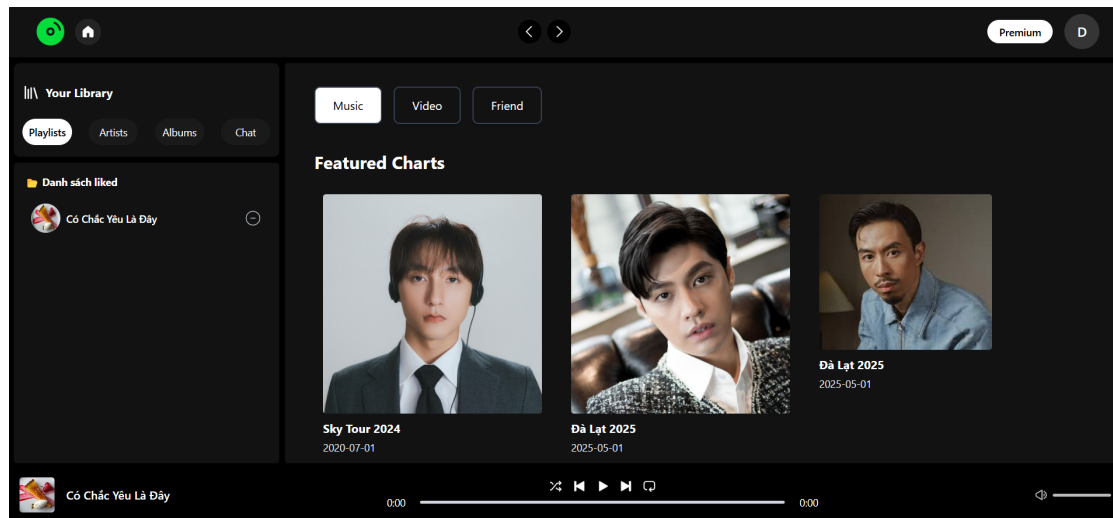
Người dùng tiến hành nhập mật khẩu hiện tại, mật khẩu mới và xác nhận mật khẩu mới để cập nhật mật khẩu.

5.5 Giao diện màn hình chính

Sau khi đã đăng nhập thành công, người dùng sẽ được chuyển đến giao diện màn hình chính, màn hình giao diện chính như sau:

Giao diện màn hình chính được bố trí với bố cục như sau:

- Bên trái cùng là thanh menu dọc cho phép người dùng chuyển giữa các trang khác nhau như: trang playlist, chat,...
- Ở giữa gồm có các nút music, video, friend, các album, song, artist.
- Ở dưới sẽ có thanh phát nhạc.



Hình 17: Giao diện màn hình chính

5.6 Chức năng phát nhạc

Chúng ta sẽ chọn 1 bài hát để phát nhạc:



Hình 18: Giao diện phát nhạc

5.7 Chức năng thêm danh sách yêu thích (song, album, artist)

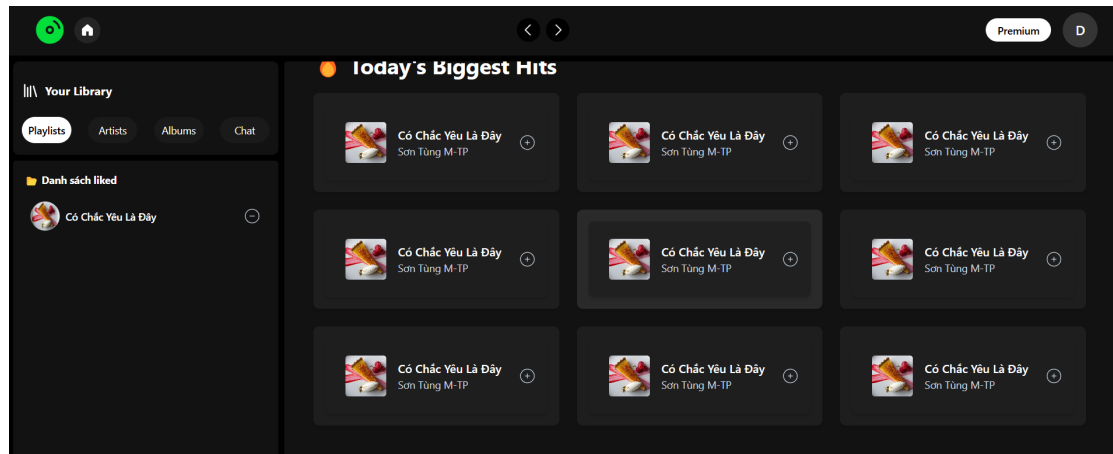
Để thêm playlist, song, artist, album ta chỉ cần chọn vào "+" để thêm vào danh sách yêu thích tương ứng.

5.8 Chức năng xem video

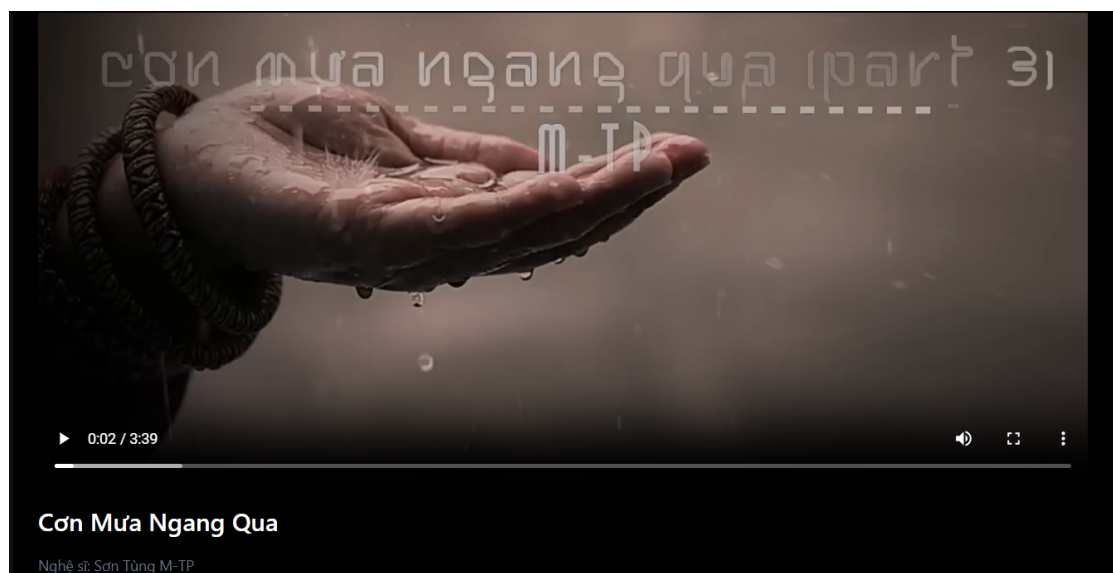
Ngoài việc nghe nhạc, người dùng có thể thưởng thức các video ca nhạc, MV hoặc video liên quan đến nghệ sĩ yêu thích. Tại giao diện video, người dùng có thể lựa chọn các danh mục như “Đề xuất”, “Thịnh hành” hoặc “Mới phát hành” để tìm kiếm nội dung phù hợp với sở thích cá nhân.

5.9 Chức năng bạn bè

Ứng dụng cho phép người dùng kết nối với bạn bè để chia sẻ bài hát, playlist hoặc các hoạt động âm nhạc khác. Tại giao diện bạn bè, người dùng có thể tìm kiếm, gửi lời mời kết bạn, chấp nhận hoặc từ chối kết bạn. Ngoài ra, người dùng còn có thể xem trạng thái hoạt động và sở thích âm nhạc của bạn bè.



Hình 19: Giao diện like

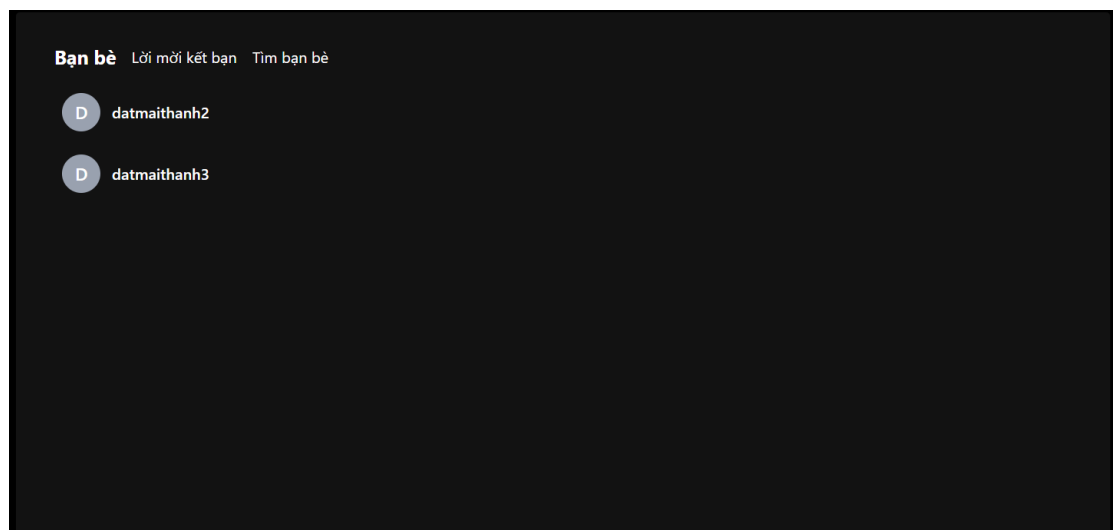


Hình 20: Giao diện xem video

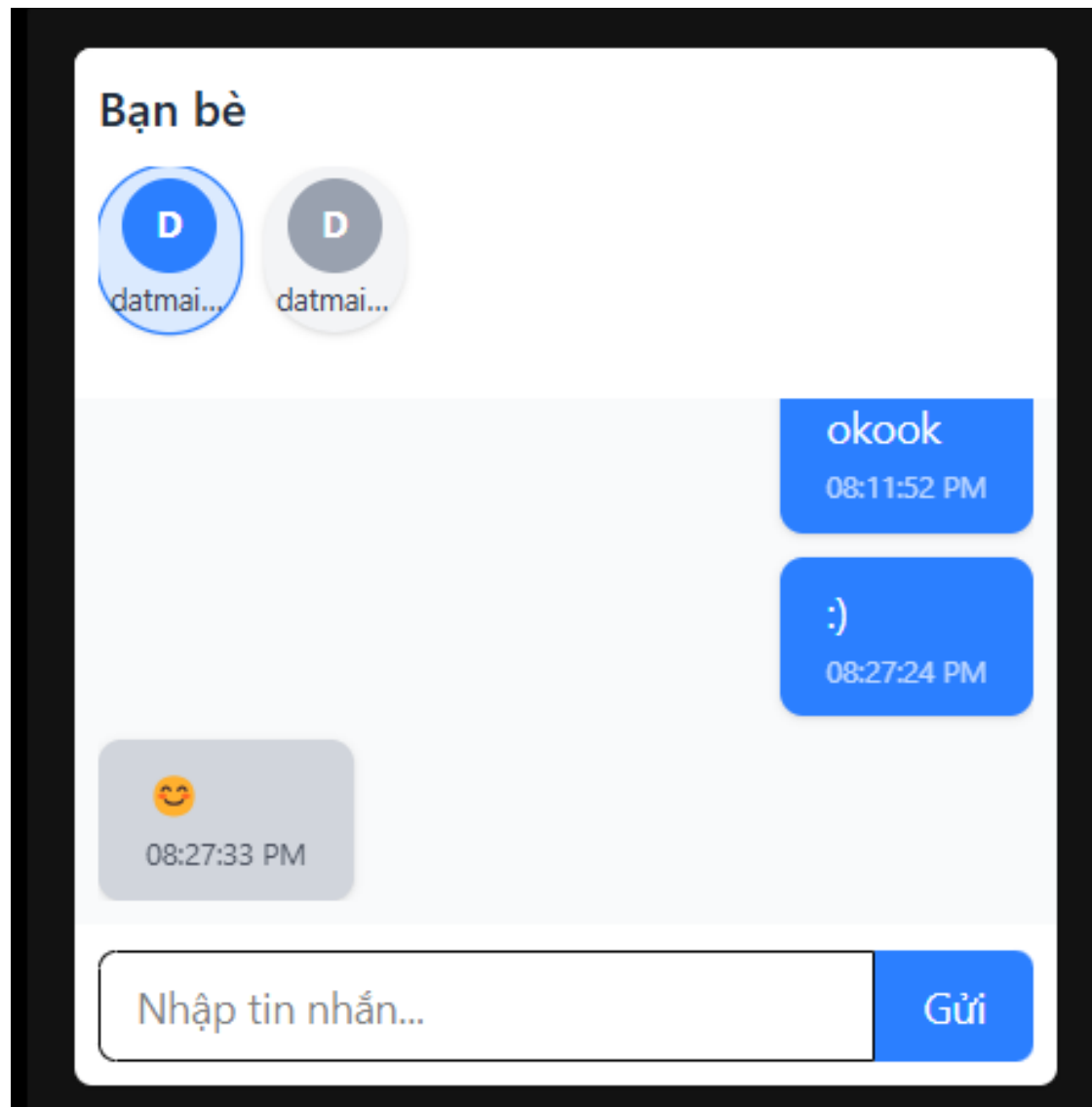


5.10 Chức năng chat

Chức năng chat giúp người dùng giao tiếp trực tiếp với bạn bè trong ứng dụng. Người dùng có thể gửi tin nhắn văn bản, chia sẻ bài hát, album hoặc thảo luận về các chủ đề âm nhạc. Giao diện chat đơn giản, dễ sử dụng và hỗ trợ hiển thị theo thời gian thực.



Hình 21: Giao diện danh sách bạn bè



Hình 22: Giao diện trò chuyện



6 Cách thức cài đặt, môi trường cài ứng dụng

Download Project

1. Change into the project directory

```
1 git clone https://github.com/datmaithanh/Spotify-Linux.git
```

2. Navigate to the project directory

```
1 cd Spotify
```

Start Client

Requirement

1. Change into the client directory

```
1 cd frontend
```

2. Install npm dependencies

```
1 npm install
```

3. Start the client interface

```
1 npm run dev
```

Start Server

Django Setup - Use Python version 3.12

1. Change into the project directory:

```
1 cd backend
```

2. Install pipenv:

```
1 pip install -r requirement.txt
```

3. Update the database:

```
1 python manage.py migrate
```

4. Start the server:



```
1 daphne config.asgi:application
```

7 Kết luận và hướng phát triển

7.1 Kết luận

Ứng dụng nghe nhạc kết hợp nhắn tin đã hoàn thiện một số chức năng cơ bản phục vụ người dùng như: nghe nhạc, xem video, thêm vào danh sách yêu thích, kết bạn, và nhắn tin trò chuyện. Giao diện người dùng trực quan, dễ sử dụng. Tuy nhiên, ứng dụng vẫn còn một số điểm chưa hoàn chỉnh và cần được nâng cấp để đáp ứng tốt hơn nhu cầu của người sử dụng.

7.2 Hướng phát triển

Sản phẩm hiện tại đã đáp ứng được những yêu cầu cơ bản, tuy nhiên vẫn cần tiếp tục cải tiến trong các phiên bản tương lai. Một số hướng phát triển cụ thể như sau:

- Khắc phục các lỗi còn tồn tại trong phiên bản hiện tại.
- Hoàn thiện các chức năng còn thiếu, một số chức năng quan trọng cần được bổ sung:
 - Hỗ trợ đa nền tảng (Web, Android, iOS).
 - Đồng bộ hóa dữ liệu người dùng và lịch sử nghe nhạc giữa các thiết bị.
 - Tìm kiếm nâng cao cho bài hát, nghệ sĩ, album, playlist.
- Cải thiện giao diện người dùng theo xu hướng hiện đại, tối ưu trải nghiệm sử dụng.
- Tăng cường bảo mật tài khoản người dùng và bảo vệ dữ liệu cá nhân.
- Tối ưu hiệu suất phát nhạc và tốc độ tải nội dung.

Ngoài ra, ứng dụng có thể tích hợp thêm các chức năng mở rộng để thu hút và giữ chân người dùng, chẳng hạn như:

- Cho phép người dùng đăng bài viết, chia sẻ cảm nhận về bài hát hoặc nghệ sĩ yêu thích.
- Hỗ trợ các loại tin nhắn đa phương tiện (ảnh động, sticker, biểu tượng cảm xúc).
- Tạo các phòng nghe nhạc chung (room), nơi nhiều người có thể cùng nghe và bình luận theo thời gian thực.
- Tích hợp thêm dịch vụ livestream, radio, hoặc podcast.



Tài liệu

- [1] <https://docs.djangoproject.com/en/5.0/>, lần truy cập cuối: 01/05/2024.
- [2] <https://www.django-rest-framework.org/topics/documenting-your-api/>, lần truy cập cuối: 01/05/2025.
- [3] <https://channels.readthedocs.io/en/latest/>, lần truy cập cuối: 01/05/2025.
- [4] <https://django-rest-framework-simplejwt.readthedocs.io/en/latest/>, lần truy cập cuối: 01/05/2025.
- [5] <https://github.com/jameshenry2020/complete-authentication-with-JWT-and-Social-Auth-in-django-rest-framework-and-react>, lần truy cập cuối: 01/05/2025.
- [6] https://github.com/kmrifat/django_chat, lần truy cập cuối: 01/05/2025.
- [7] <https://gemini.google.com/app>, lần truy cập cuối: 01/05/2025.
- [8] <https://chatgpt.com/>, lần truy cập cuối: 01/05/2025.