TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

ĐỀ TÀI: PHÁT TRIỂN PHẦN MỀM SPOTIFY CLONE

NHÓM 20

Sinh viên thực hiện: 3121410168 - Phạm Trà Trường Giang

Giảng viên hướng dẫn: ThS. Từ Lãng Phiêu

Email liên hệ: giangphamtratuong@gmail.com

TPHCM, ngày 5 tháng 5 năm 2025

LỜI CẢM ƠN

Đầu tiên, em xin chân thành gửi lời cảm ơn sâu sắc đến Thầy Từ Lãng Phiêu đã luôn tận tình hướng dẫn và chỉ bảo trong suốt quá trình thực hiện báo cáo này.

Những lời khuyên có giá trị và sự hướng dẫn từ Thầy không chỉ giúp em vượt qua những thách thức mà còn thúc đẩy sự sáng tạo và khả năng tư duy phân tích.

Tuy nhiên, trong quá trình thực hiện, em nhận thức rằng vẫn còn những thiếu sót và sai sót mà em chưa thể khắc phục hết. Em rất mong nhận được những phản hồi và góp ý từ Thầy để có thể hoàn thiện sản phẩm hơn nữa.

Cuối cùng, em xin kính chúc Thầy luôn giữ vững niềm đam mê và nhiệt huyết trong sứ mệnh giảng dạy và nghiên cứu, và hy vọng được tiếp tục học hỏi những kiến thức từ thầy trong tương lai.

Trân trọng,

Nhóm 20

MỤC LỤC

DANH MỤC HÌNH ẢNH

PHẦN 1: MỞ ĐẦU

1. Giới thiệu đề tài

Trong thời đại công nghệ số phát triển mạnh mẽ, các nền tảng âm nhạc trực tuyến như Spotify đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày, mang đến trải nghiệm nghe nhạc cá nhân hóa và tiện lợi cho hàng triệu người dùng trên toàn cầu. Tuy nhiên, việc tiếp cận và tùy chỉnh các tính năng của những nền tảng thương mại này thường bị giới hạn bởi chi phí hoặc các chính sách bản quyền. Điều này đã khơi dậy nhu cầu xây dựng các ứng dụng mã nguồn mở hoặc dự án cá nhân để học hỏi và thực hành các công nghệ hiện đại trong phát triển phần mềm. Dự án "Spotify Clone" ra đời với mục tiêu tái hiện các chức năng cơ bản của một nền tảng âm nhạc, từ quản lý người dùng, tạo album, đến hiển thị danh sách bài hát, như một cơ hội để áp dụng kiến thức về lập trình backend và frontend, đồng thời cung cấp một sản phẩm thực tế để nghiên cứu và mở rộng.

Lý do chọn đề tài còn xuất phát từ sự phổ biến của các công nghệ như Django, React, và REST API trong ngành công nghiệp phần mềm. Việc kết hợp các công nghệ này không chỉ giúp người học nắm bắt các kỹ thuật phát triển ứng dụng web hiện đại mà còn tạo nền tảng để phát triển các dự án lớn hơn trong tương lai. Hơn nữa, dự án này cho phép em phát triển khám phá các khía cạnh thực tế như quản lý cơ sở dữ liệu, xử lý lỗi, và thiết kế giao diện người dùng, điều mà lý thuyết đơn thuần không thể cung cấp

2. Mục tiêu

Mục tiêu chính của dự án Spotify Clone là xây dựng một ứng dụng web mô phỏng các tính năng cơ bản của một nền tảng âm nhạc, bao gồm:

- Xây dựng hệ thống đăng nhập và đăng ký người dùng với xác thực an toàn.
- Tích hợp chức năng tạo và quản lý album cá nhân, cho phép người dùng thêm bài hát vào album của mình.
- Thiết kế giao diện thân thiện để xem chi tiết album và danh sách bài hát.
- Phát triển cơ sở hạ tầng backend và frontend sử dụng các công nghệ hiện đại như Django REST Framework và React.

Ngoài ra, dự án nhằm mục đích:

- Rèn luyện kỹ năng lập trình nhóm, quản lý mã nguồn, và giải quyết vấn đề thực tế.
- Tạo ra một sản phẩm có thể mở rộng để tích hợp các tính năng nâng cao trong tương lai, như phát nhạc trực tuyến hoặc gợi ý bài hát.

3. Phạm vi

Phạm vi của dự án tập trung vào các chức năng cốt lõi của một nền tảng âm nhạc cơ bản, bao gồm:

- Quản lý người dùng: Đăng nhập, đăng ký, cập nhật thông tin cá nhân.
- Quản lý nội dung âm nhạc: Tạo và xem album, hiển thị danh sách bài hát.
- Giao diện người dùng: Thiết kế các trang chính như Library, Album Detail sử dụng React + Vite và Tailwind CSS.

Dự án không bao gồm các tính năng phức tạp như phát nhạc trực tiếp (streaming), xử lý bản quyền âm nhạc, hoặc tích hợp trí tuệ nhân tạo (AI) để gợi ý bài hát do giới hạn thời gian và nguồn lực. Tuy nhiên, các chức năng này có thể được xem xét để mở rộng trong các giai đoạn sau.

4. Các chức năng của dự án

4.1. Quản lý người dùng

- **Đăng nhập và Đăng ký**:
    - Người dùng có thể đăng nhập vào hệ thống bằng username và mật khẩu thông qua giao diện frontend được xây dựng bằng React.
    - Tính năng đăng ký cho phép tạo tài khoản mới với thông tin cơ bản như username, email, và mật khẩu.
    - Xác thực người dùng được thực hiện bằng JWT (JSON Web Token) thông qua Django REST Framework, đảm bảo an toàn và bảo mật thông tin.

    - Sau khi đăng nhập thành công, thông tin người dùng (username, email) được lưu trữ trong localStorage và hiển thị trên giao diện.
- **Cập nhật thông tin cá nhân**:
    - Người dùng đã đăng nhập có thể cập nhật thông tin cá nhân, bao gồm username, email.
    - Chức năng này được thực hiện thông qua API POST/PUT, với dữ liệu được gửi từ frontend và xử lý bởi backend Django.

4.2. Quản lý nội dung âm nhạc

- **Tạo và Quản lý Album**:
    - Người dùng có thể tạo album mới bằng cách nhập tiêu đề, chọn nghệ sĩ, và thêm danh sách bài hát từ giao diện Library.
    - Dữ liệu album (title, artist, song_ids) được gửi qua yêu cầu POST đến endpoint /api/music/albums/ với xác thực JWT.
    - Backend tự động gán người dùng hiện tại (request.user) vào trường user của album, đảm bảo quyền sở hữu.
    - Album được lưu trữ trong cơ sở dữ liệu SQLite với trường song_ids dưới dạng JSONField để lưu danh sách ID bài hát.
- **Xem chi tiết Album**:
    - Người dùng có thể truy cập chi tiết một album thông qua URL /album/:id, nơi :id là ID của album.
    - Chức năng này sử dụng API GET /api/music/albums/:id/ để lấy thông tin album, bao gồm tiêu đề, ảnh bìa, ngày tạo, và danh sách bài hát tương ứng với song_ids.
    - Giao diện hiển thị thông tin album và danh sách bài hát với bố cục thân thiện, sử dụng Tailwind CSS để tối ưu hóa trải nghiệm người dùng.

4.3. Giao diện người dùng

- **Trang Library**:
    - Trang này liệt kê tất cả các album thuộc về người dùng hiện tại, được lấy từ API /api/music/albums/.

    - Mỗi album được hiển thị dưới dạng thẻ (card) với hình ảnh bìa và tiêu đề, kèm theo liên kết đến trang chi tiết.
    - Nút "Tạo Album Mới" mở form để người dùng nhập thông tin và chọn bài hát, với danh sách bài hát được tải từ API /api/music/songs/.
- **Trang Album Detail**:
    - Trang chi tiết hiển thị thông tin cụ thể của album, bao gồm ảnh bìa, tên album, ngày tạo, và danh sách bài hát.
    - Dữ liệu được xử lý động từ API, với xử lý lỗi chi tiết (401, 403, 404) để thông báo người dùng khi gặp vấn đề.
- **Xử lý định tuyến**:
    - Sử dụng React Router để quản lý các tuyến đường như /, /library, /album/:id, đảm bảo chuyển đổi mượt mà giữa các trang.

4.4. Tính năng bổ trợ

- **Xử lý lỗi và Debug**:
    - Ứng dụng tích hợp cơ chế làm mới token tự động khi gặp lỗi 401 Unauthorized, sử dụng refresh token thông qua API /api/user/refresh/.
    - Giao diện frontend hiển thị thông báo lỗi cụ thể (ví dụ: "Phiên đăng nhập hết hạn" hoặc "Album không tồn tại") để hỗ trợ người dùng.
- **Quản lý phiên làm việc**:
    - Người dùng có thể đăng xuất, xóa thông tin phiên làm việc khỏi localStorage, và quay lại trang đăng nhập.
    - Tính năng này được tích hợp với API /api/user/logout/ để đảm bảo an toàn.

PHẦN 2: CƠ SỞ LÝ THUYẾT

1. Lý thuyết chung

Dự án Spotify Clone được xây dựng dựa trên sự kết hợp của các công nghệ và thư viện hiện đại, đảm bảo hiệu suất và khả năng mở rộng. Dưới đây là các công nghệ chính được sử dụng trong dự án:

- **Django**: Một framework Python mạnh mẽ để phát triển backend. Django cung cấp các tính năng như ORM (Object-Relational Mapping) để quản lý cơ sở dữ liệu, hệ thống xác thực người dùng, và cấu trúc MVC (Model-View-Controller) để tổ chức mã nguồn. Trong dự án này, Django được sử dụng để xây dựng các API và xử lý logic backend.
- **Django REST Framework (DRF)**: Thư viện mở rộng của Django để phát triển API RESTful. DRF hỗ trợ serialization (chuyển đổi dữ liệu thành định dạng JSON), xác thực JWT (JSON Web Token), và các tính năng như ViewSet để quản lý API một cách hiệu quả. Dự án sử dụng DRF để tạo các endpoint như /api/music/albums/ và /api/user/login/.
- **Vite + React**: Vite là công cụ build hiện đại giúp tăng tốc quá trình phát triển frontend. Kết hợp với React – thư viện JavaScript để xây dựng giao diện người dùng – Vite cho phép phát triển các thành phần giao diện tái sử dụng, sử dụng JSX để kết hợp HTML và JavaScript, và quản lý trạng thái (state) hiệu quả. Trong dự án, Vite được dùng để thiết lập môi trường phát triển nhanh chóng, còn React được dùng để xây dựng các trang như **Library**, **AlbumDetail**, và các thành phần như **Sidebar**, **Player**.
- **Axios**: Thư viện HTTP client để thực hiện các yêu cầu API từ frontend đến backend. Axios hỗ trợ xử lý các yêu cầu HTTP (GET, POST, PUT, DELETE) và quản lý lỗi dễ dàng. Dự án sử dụng Axios để gọi các API như lấy danh sách album hoặc tạo album mới.
- **React Router**: Thư viện quản lý định tuyến trong ứng dụng React. React Router cho phép chuyển đổi giữa các trang (như /library, /album/:id) mà không cần tải lại toàn bộ trang, mang lại trải nghiệm người dùng mượt mà.

- **SQLite**: Cơ sở dữ liệu mặc định của Django, được sử dụng trong môi trường phát triển. SQLite là một cơ sở dữ liệu nhẹ, không cần máy chủ riêng, phù hợp cho các dự án nhỏ và vừa. Dự án lưu trữ thông tin người dùng, bài hát, nghệ sĩ, và album trong SQLite.
- **Pillow**: Thư viện Python để xử lý hình ảnh, được sử dụng để hỗ trợ các trường ImageField trong Django (như ảnh bìa album, ảnh nghệ sĩ).
- **django-cors-headers**: Thư viện Django để xử lý vấn đề CORS (Cross-Origin Resource Sharing), cho phép frontend (React) và backend (Django) giao tiếp với nhau dù chạy trên các domain khác (localhost:3000 và localhost:8000).
- **Tailwind CSS**: Framework CSS để thiết kế giao diện người dùng. Tailwind cung cấp các lớp tiện ích (utility classes) để tạo giao diện responsive và đẹp mắt mà không cần viết CSS thủ công. Dự án sử dụng Tailwind để định dạng các trang và thành phần giao diện.

2. Cấu trúc mã nguồn

Cấu trúc mã nguồn của dự án được tổ chức khá rõ ràng để đảm bảo tính dễ đọc và khả năng bảo trì. Cấu trúc chính bao gồm:

- **backend/**: Thư mục chứa mã nguồn backend (Django).
    - server/: Chứa các tệp cấu hình dự án.
        - settings.py: Cấu hình các ứng dụng, middleware, CORS, và cơ sở dữ liệu.
        - urls.py: Định nghĩa các tuyến API (như /api/music/albums/, /api/user/login/).
    - music/: Ứng dụng quản lý nội dung âm nhạc.
        - models.py: Định nghĩa các mô hình Song, Artist, Album.
        - views.py: Xử lý logic API với ViewSet (SongViewSet, ArtistViewSet, AlbumViewSet).
        - serializers.py: Định nghĩa các serializer để chuyển đổi dữ liệu (SongSerializer, AlbumSerializer).
- **frontend/**: Thư mục chứa mã nguồn frontend (React).

- src/pages/: Các trang giao diện.
  - HomePage.jsx: Trang chính.
  - AlbumPage.jsx: Trang quản lý album.
  - UserAlbumPage.jsx: Trang chi tiết album.
- src/components/: Các thành phần tái sử dụng.
  - Sidebar.js: Thanh điều hướng.
  - Player.js: Trình phát nhạc (chưa hoàn thiện chức năng phát nhạc).
- App.js: Tệp chính để định nghĩa định tuyến và quản lý trạng thái toàn cục.

3. Mô hình ứng dụng

Ứng dụng được thiết kế theo mô hình MVC (Model-View-Controller):

- **Model**: Đại diện cho dữ liệu và logic cơ sở dữ liệu, được định nghĩa trong Django (CustomUser, Song, Artist, Album). Các mô hình này ánh xạ trực tiếp vào bảng cơ sở dữ liệu thông qua Django ORM.
- **View**: Bao gồm hai phần:
  - Backend: API views trong Django REST Framework (ViewSet) để xử lý yêu cầu và trả về dữ liệu JSON.
  - Frontend: Giao diện React để hiển thị dữ liệu cho người dùng (Library, AlbumDetail).
- **Controller**: Quản lý luồng dữ liệu giữa Model và View, được thực hiện thông qua các API endpoint và logic trong React (state, useEffect).

4. Các tính năng được xây dựng

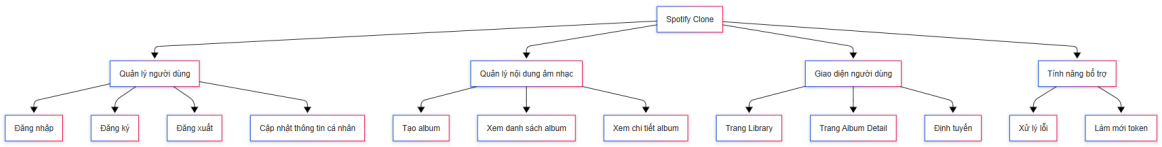
Dự án đã triển khai các tính năng chính sau:

- **Đăng nhập và Đăng ký**: Người dùng có thể tạo tài khoản và đăng nhập với xác thực JWT.
- **Tạo và Quản lý Album**: Người dùng có thể tạo album mới, chọn bài hát, và xem danh sách album của mình trong trang Library.

- **Xem chi tiết Album**: Hiển thị thông tin album (tiêu đề, ảnh bìa, ngày tạo) và danh sách bài hát trong trang AlbumDetail.
- **Xử lý lỗi**: Tích hợp cơ chế làm mới token tự động và hiển thị thông báo lỗi chi tiết (401, 403, 404).

PHẦN 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

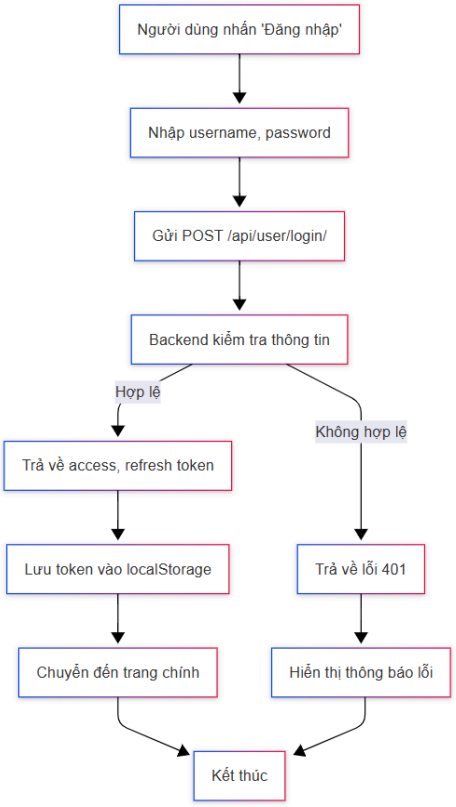
1. Sơ đồ phân rã chức năng



Hình 1: Sơ đồ phân rã chức năng

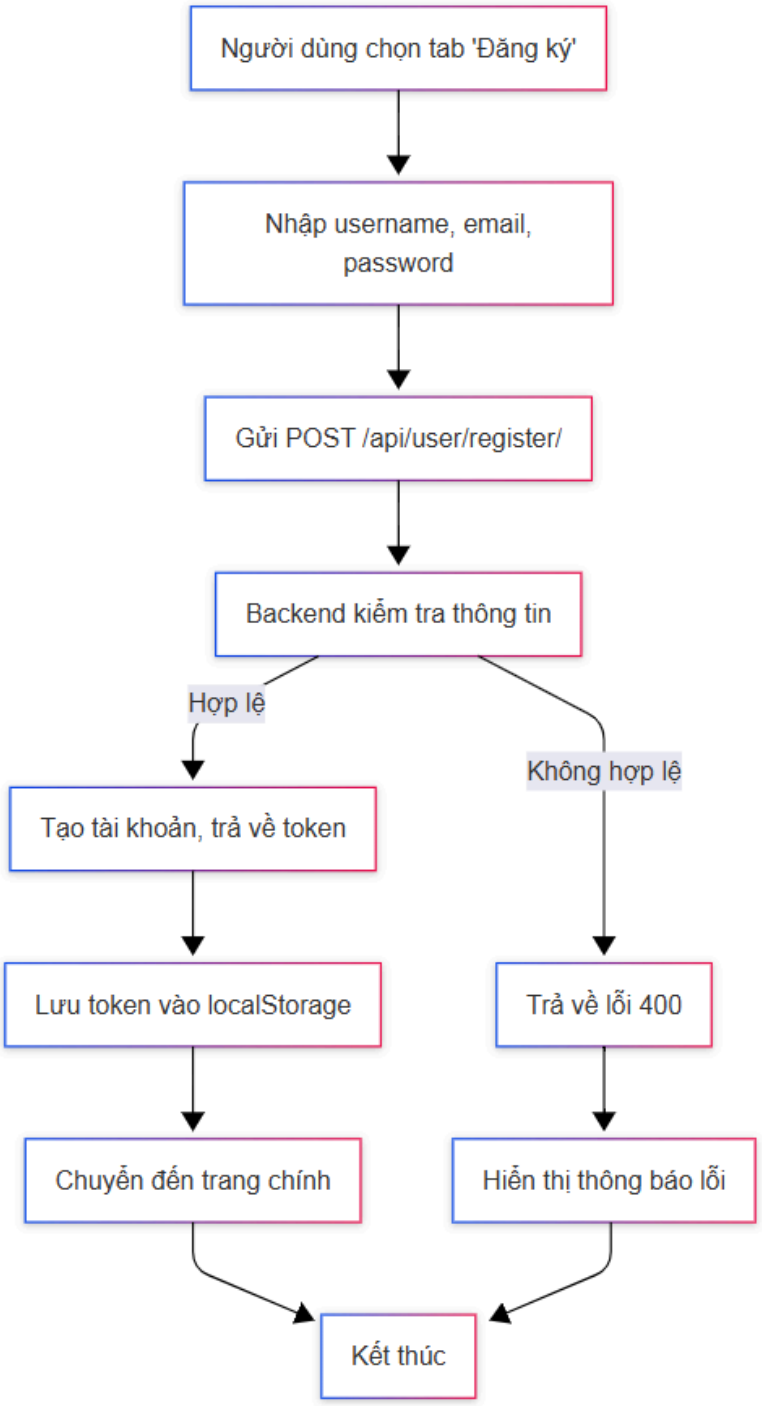
2. Sơ đồ luồng

2.1 Chức năng đăng nhập



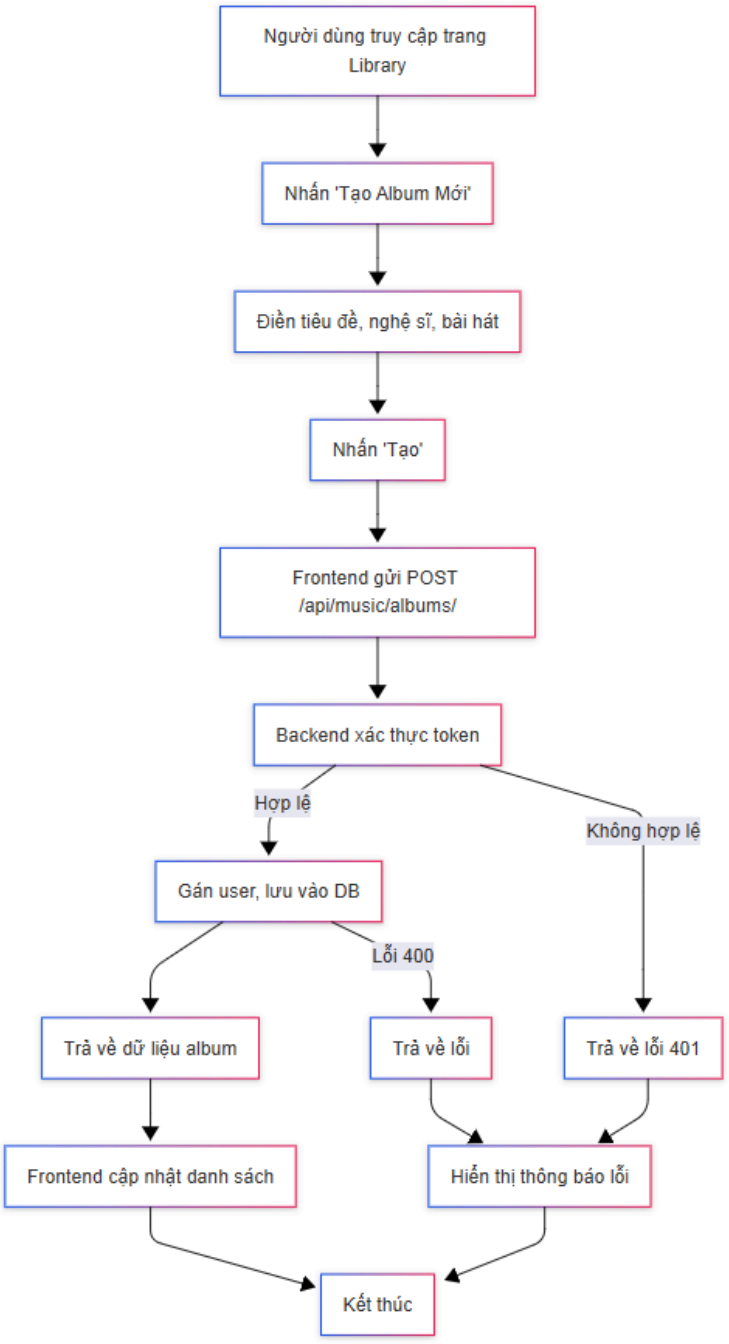
Hình 2: Sơ đồ luồng đăng nhập

2.2 Chức năng đăng ký



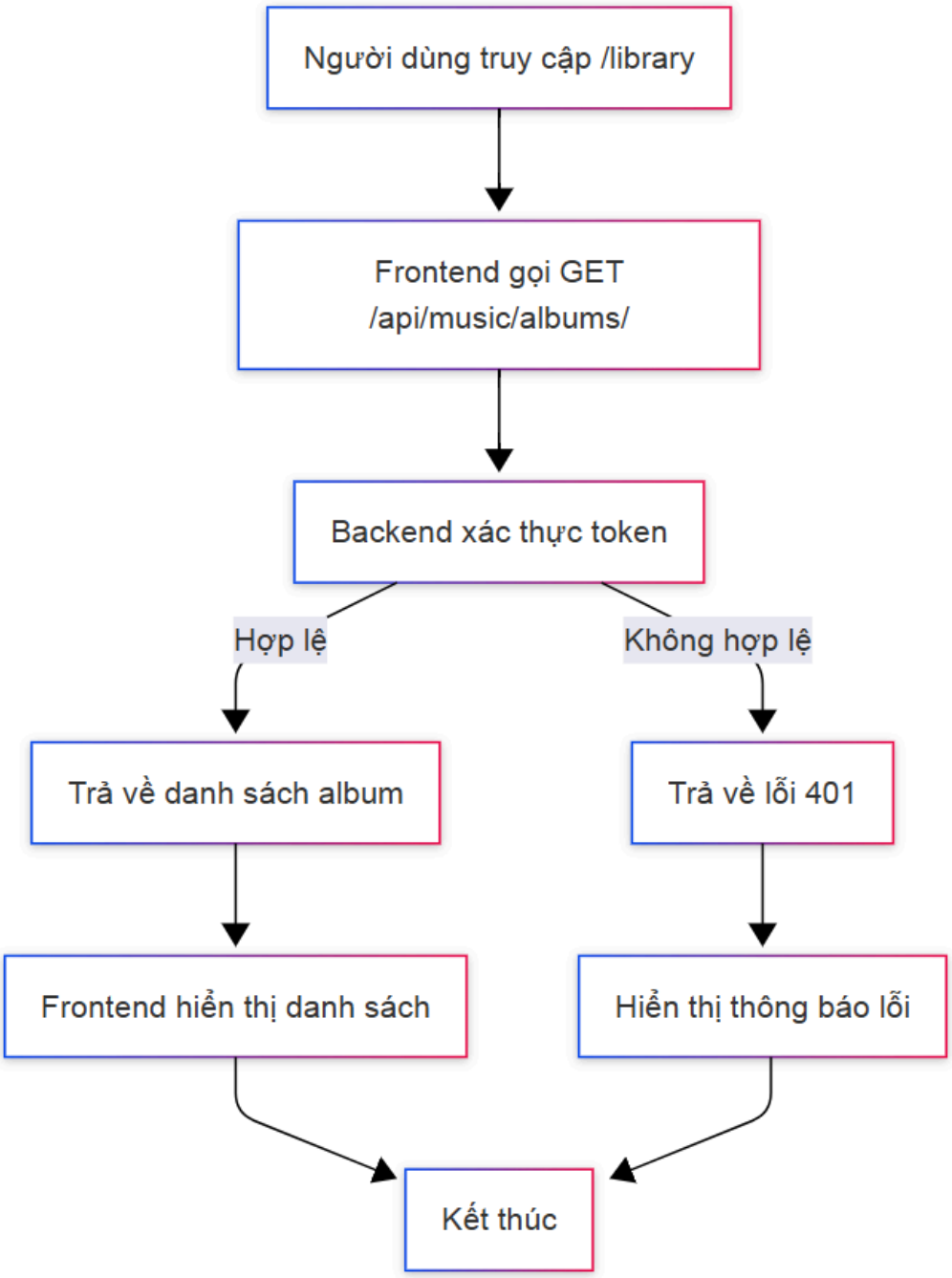
Hình 3: Sơ đồ luồng đăng ký

2.3 Chức năng tạo album



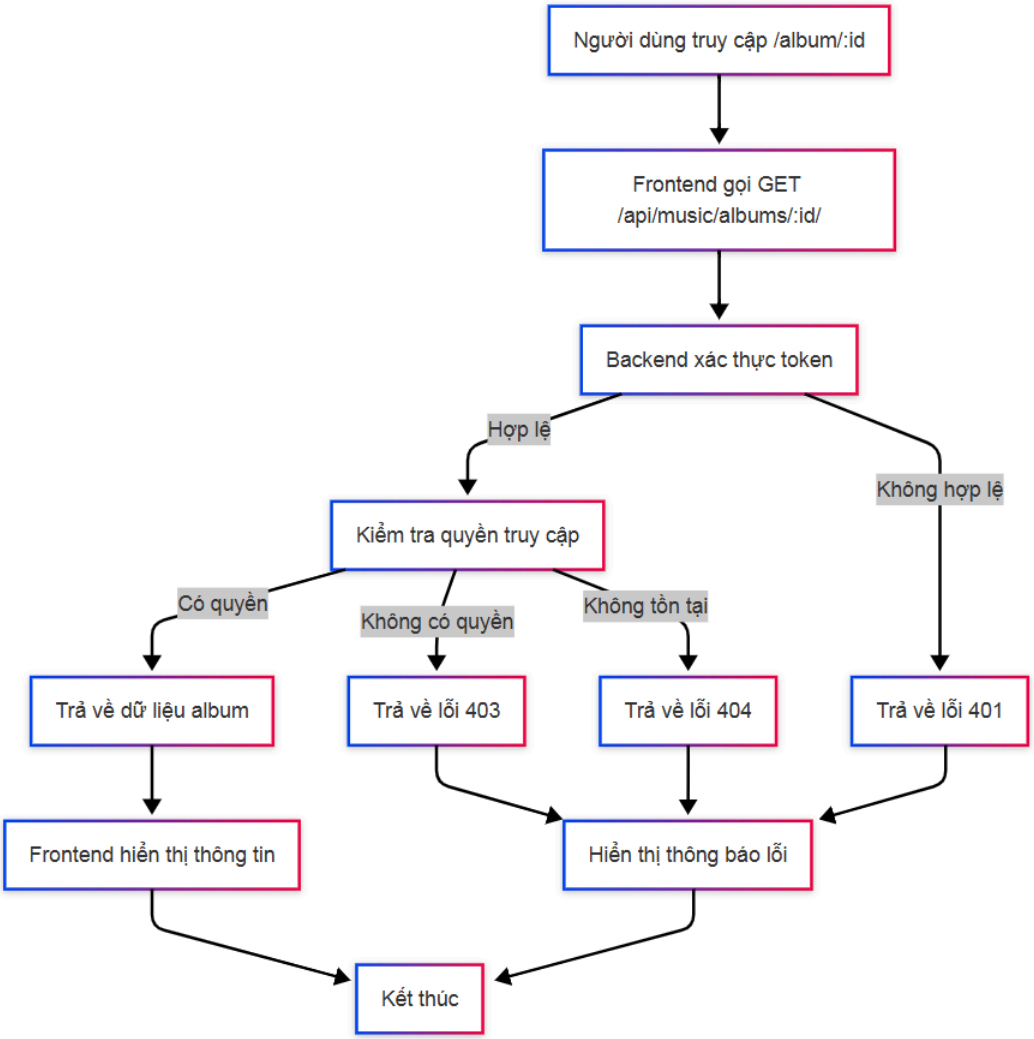
Hình 3: Sơ đồ luồng tạo album

2.4 Lấy danh sách album đã tạo



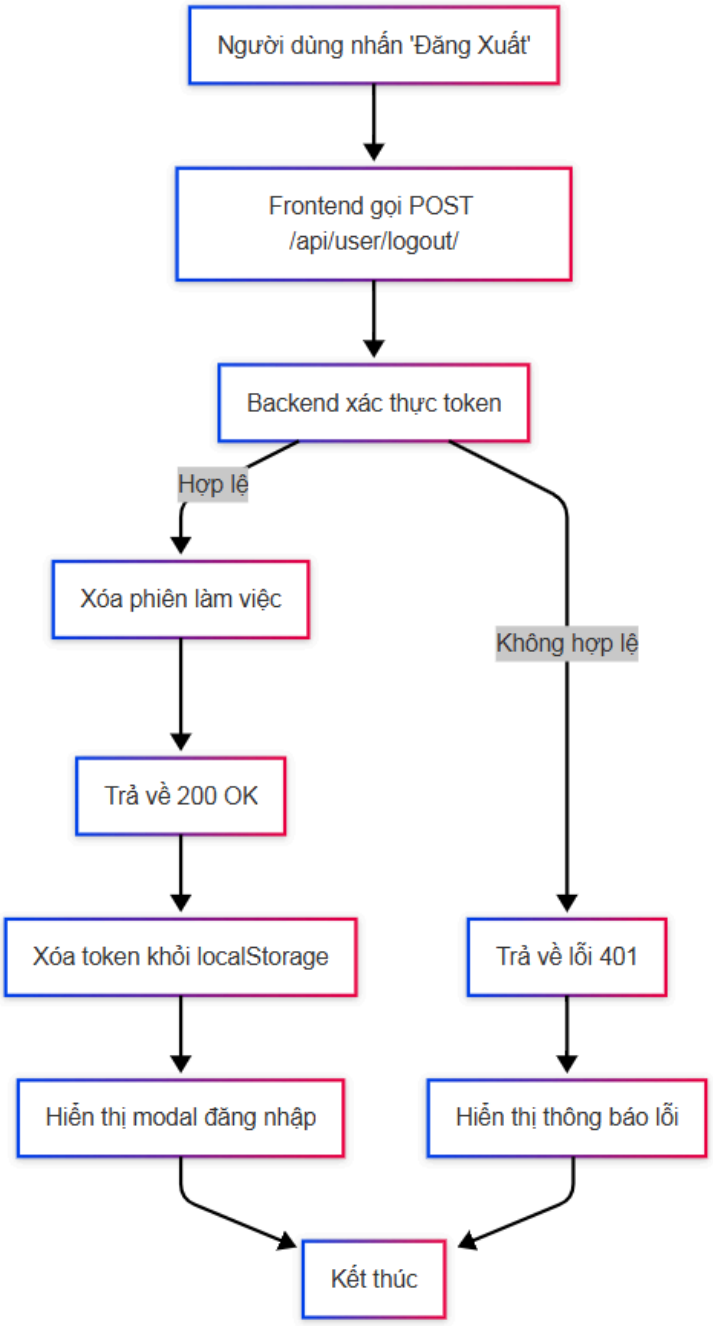
Hình 4: Sơ đồ luồng lấy danh sách album

2.5 Luồng xem chi tiết album



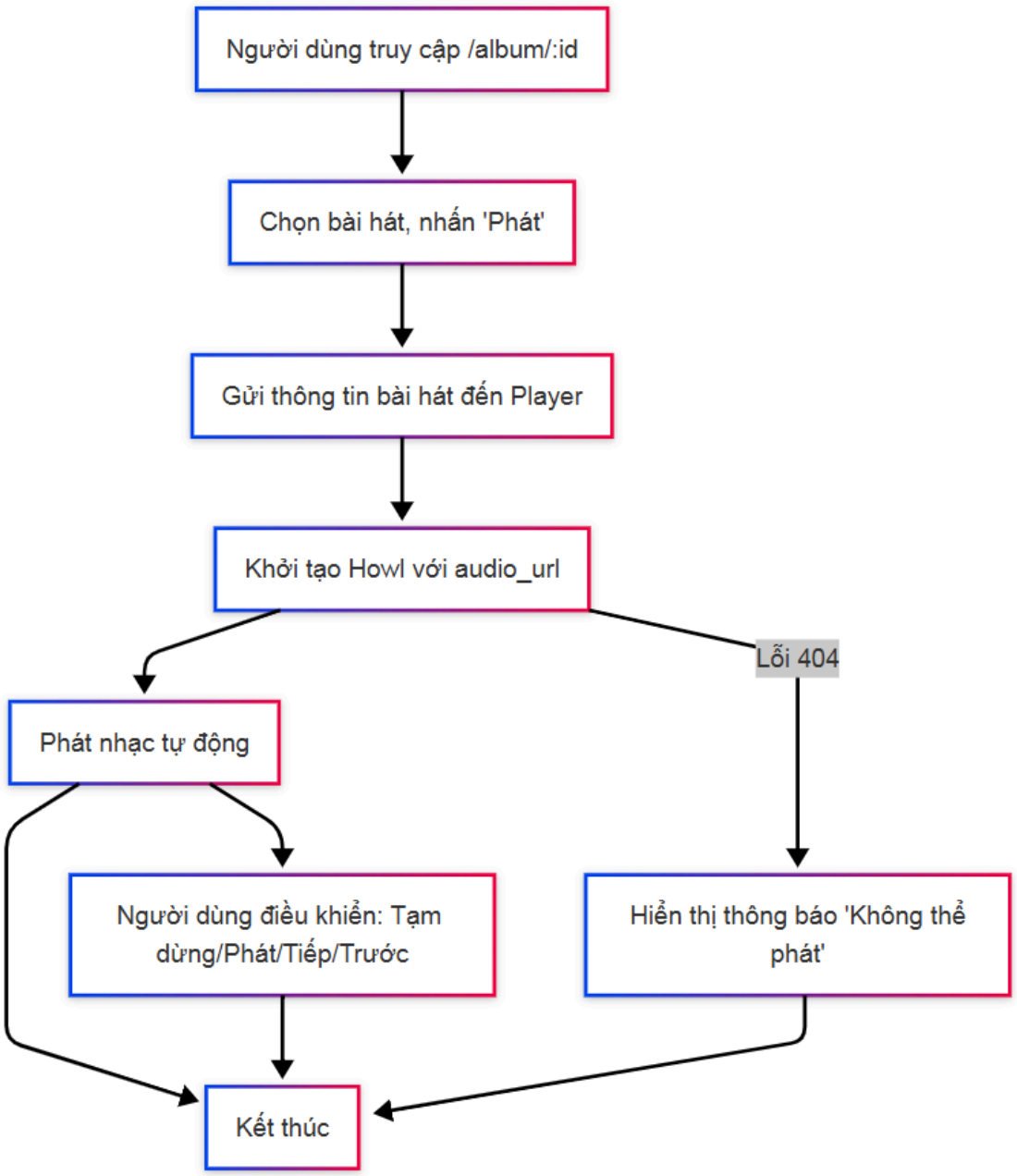
Hình 5: Sơ đồ luồng xem chi tiết album

2.6 Luồng đăng xuất



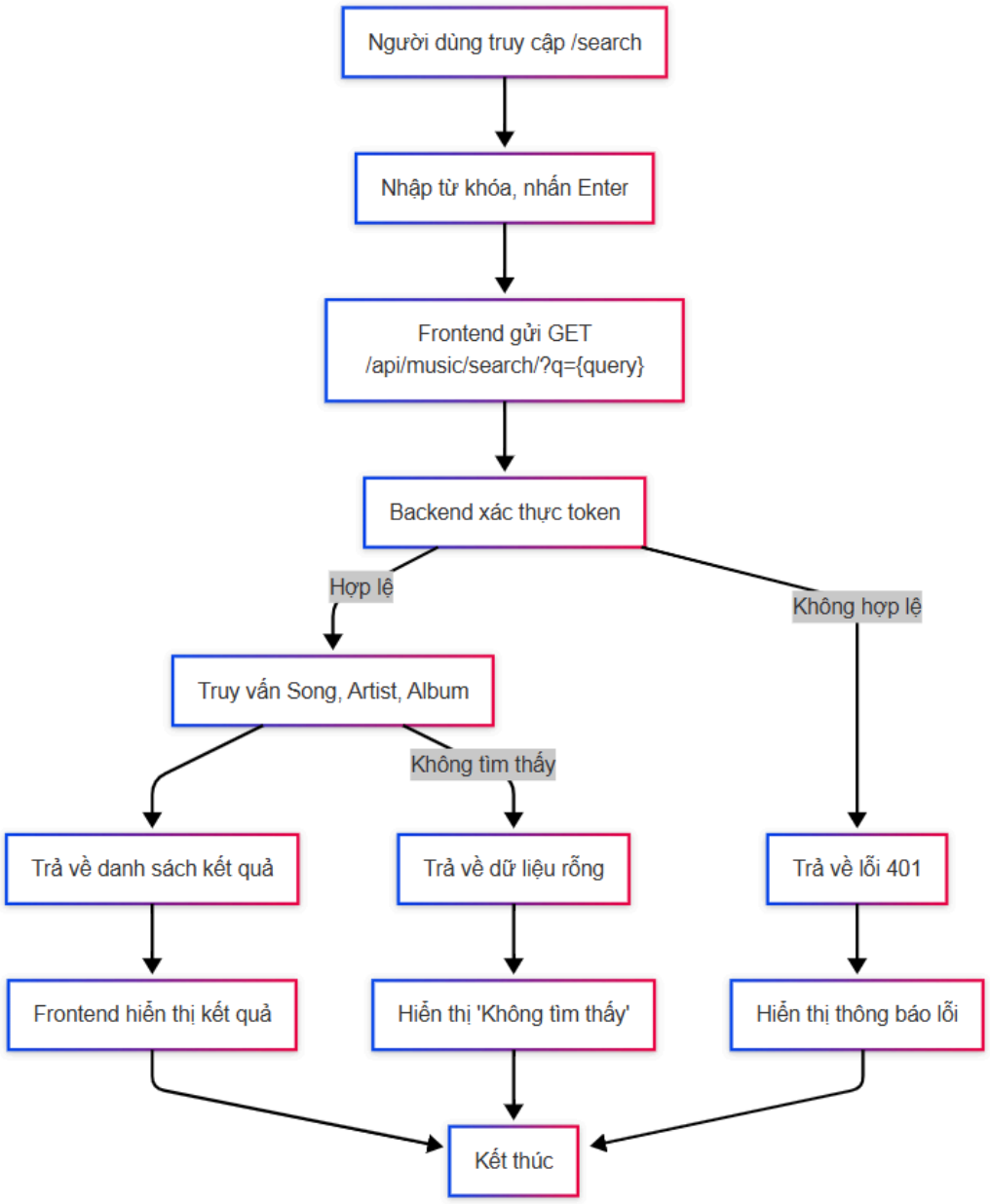
Hình 6: Sơ đồ luồng đăng xuất

2.7 Luồng phát nhạc



Hình 7: Sơ đồ luồng phát nhạc

2.8 Luồng tìm kiếm



Hình 8: Sơ đồ luồng chức năng tìm kiếm

3. Cơ sở dữ liệu

1. Tổng quan về cơ sở dữ liệu

Cơ sở dữ liệu của dự án Spotify Clone được thiết kế để lưu trữ thông tin liên quan đến người dùng, bài hát, nghệ sĩ, và album. Hiện tại, dự án sử dụng **SQLite** làm hệ quản trị cơ sở dữ liệu (HTQLCS) mặc định của Django trong môi trường phát triển, nhờ tính đơn giản, nhẹ, và không yêu cầu cấu hình máy chủ riêng. Tuy nhiên, trong môi trường sản xuất, có thể thay thế bằng các HTQLCS mạnh hơn như PostgreSQL hoặc MySQL để đảm bảo hiệu suất và khả năng mở rộng.

Mục tiêu của thiết kế cơ sở dữ liệu là:

- Lưu trữ dữ liệu một cách có tổ chức để hỗ trợ các chức năng như quản lý người dùng, tạo album, và tìm kiếm.
- Đảm bảo tính toàn vẹn dữ liệu thông qua các mối quan hệ giữa các bảng.
- Hỗ trợ truy vấn nhanh chóng và hiệu quả thông qua các chỉ mục (indexes) và quan hệ khóa ngoại (foreign keys).

2. Thiết kế cơ sở dữ liệu

2.1. Các bảng chính

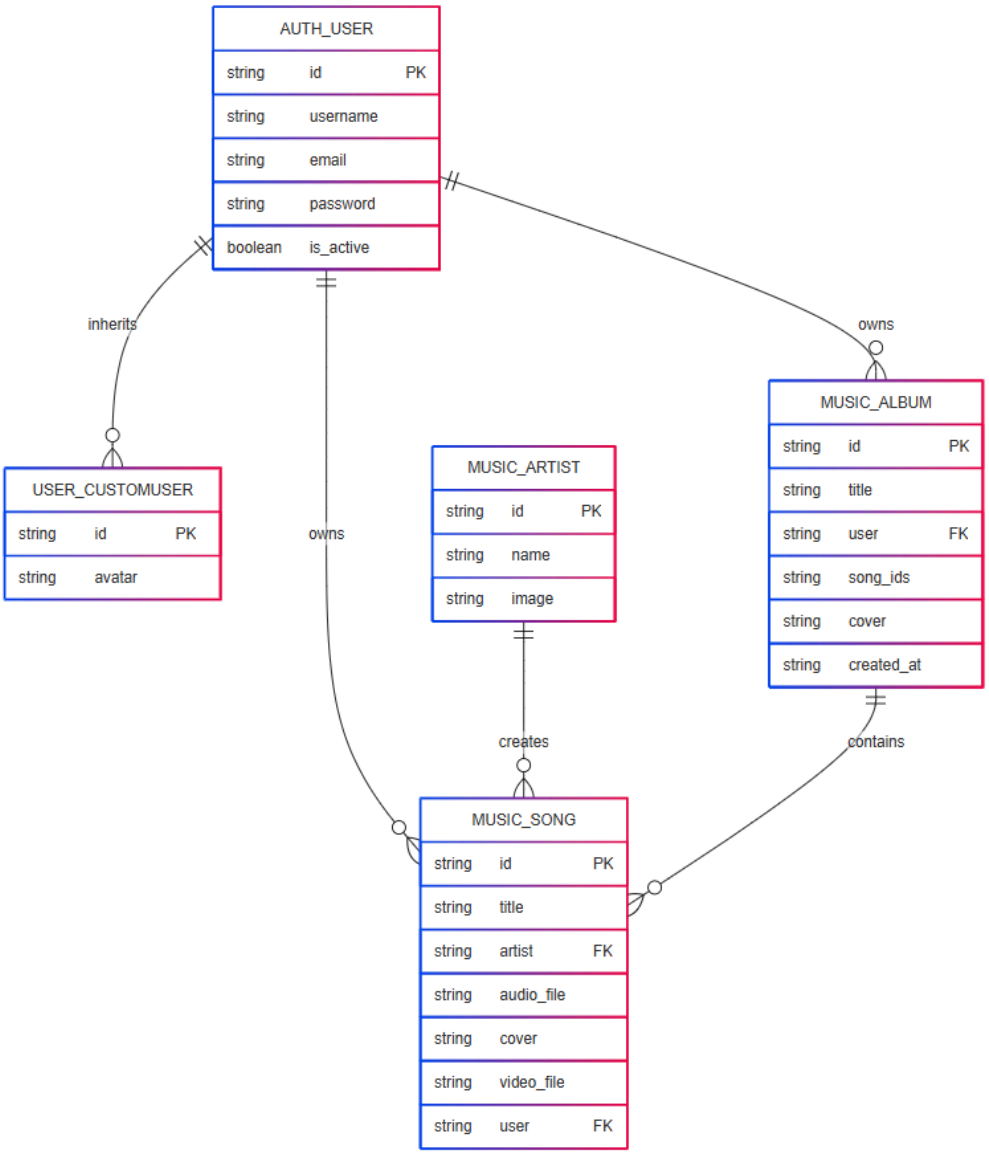
Cơ sở dữ liệu bao gồm các bảng sau, được ánh xạ từ các mô hình Django:

- **auth_user**:
    - Bảng mặc định của Django để lưu thông tin người dùng.
    - Các trường chính:
        - id (Integer, Primary Key): Khóa chính, định danh người dùng.
        - username (CharField): Tên người dùng, duy nhất.
        - email (EmailField): Địa chỉ email, duy nhất.
        - password (CharField): Mật khẩu đã mã hóa.
        - is_active (Boolean): Trạng thái kích hoạt tài khoản.
- **user_customuser**:
    - Bảng mở rộng từ auth_user để thêm thông tin tùy chỉnh cho người dùng.

    - Các trường chính:
        - id (Integer, Primary Key): Khóa chính, kế thừa từ auth_user.
        - avatar (ImageField): Hình ảnh đại diện của người dùng (tùy chọn, upload_to='avatars/').
    - Quan hệ: Kế thừa từ auth_user thông qua cơ chế OneToOneField.
- **music_song**:
    - Bảng lưu thông tin bài hát.
    - Các trường chính:
        - id (Integer, Primary Key): Khóa chính, định danh bài hát.
        - title (CharField, max_length=200): Tên bài hát.
        - artist (ForeignKey): Khóa ngoại liên kết với bảng music_artist, chỉ định nghệ sĩ sáng tác.
        - audio_file (FileField, upload_to='songs/'): Tệp âm thanh của bài hát.
        - cover (ImageField, upload_to='covers/', blank=True, null=True): Ảnh bìa bài hát (tùy chọn).
        - video_file (FileField, upload_to='videos/', blank=True, null=True): Tệp video liên quan (tùy chọn).
        - user (ForeignKey): Khóa ngoại liên kết với auth_user, chỉ định chủ sở hữu bài hát.
    - Quan hệ: Liên kết với music_artist (1-n) và auth_user (1-n).
- **music_artist**:
    - Bảng lưu thông tin nghệ sĩ.
    - Các trường chính:
        - id (Integer, Primary Key): Khóa chính, định danh nghệ sĩ.
        - name (CharField, max_length=100): Tên nghệ sĩ.
        - image (ImageField, upload_to='artists/', blank=True, null=True): Ảnh đại diện nghệ sĩ (tùy chọn).
    - Quan hệ: Liên kết với music_song (1-n).
- **music_album**:
    - Bảng lưu thông tin album.

- Các trường chính:
  - id (Integer, Primary Key): Khóa chính, định danh album.
  - title (CharField, max_length=200): Tên album.
  - user (ForeignKey): Khóa ngoại liên kết với auth_user, chỉ định chủ sở hữu album.
  - song_ids (JSONField, default=list): Danh sách ID bài hát trong album (lưu dưới dạng JSON).
  - cover (ImageField, upload_to='albums/', blank=True, null=True): Ảnh bìa album (tùy chọn).
  - created_at (DateTimeField, auto_now_add=True): Ngày tạo album.
- Quan hệ: Liên kết với auth_user (1-n).

2.2 Mô hình ERD



Hình 9: Mô hình ERD

PHẦN 4: HƯỚNG DẪN CÀI ĐẶT DỰ ÁN

1. Công nghệ sử dụng

Backend: Django, Django REST Framework

Frontend: React+Vite, Tailwind CSS

Cơ sở dữ liệu: SQLite (mặc định), hoặc có thể dùng PostgreSQL/MySQL nếu cần

Xác thực: Django Authentication

Chạy server frontend/backend riêng biệt

2. Yêu cầu hệ thống

Trên Ubuntu, cần cài đặt các công cụ sau:

- Python ≥ 3.8
- Node.js ≥ 14.x
- npm ≥ 6.x
- Git
- SQLite (có sẵn trên hầu hết các bản Ubuntu)
- Visual Studio Code (khuyến nghị)

3. Hướng dẫn cài đặt

3.1 Clone mã nguồn từ github

Tải mã nguồn

```
git clone
https://github.com/giang-de-newbie/spotify-clone-basic.git
cd spotify-clone-basic
```

3.2 Cài đặt Backend (Django)

3.2.1 Tạo và kích hoạt môi trường ảo Python

```
cd backend
python3 -m venv venv
source venv/bin/activate
```

3.2.2 Cài đặt các thư viện phụ thuộc

```
pip install -r requirements.txt
```

3.2.3 Khởi tạo cơ sở dữ liệu và tài khoản quản trị

```
python manage.py migrate
python manage.py createsuperuser
```

Nhập tên người dùng, email và mật khẩu theo hướng dẫn để tạo tài khoản admin.

3.2.4 Chạy server backend

```
python manage.py runserver
```

Mặc định server sẽ chạy tại http://localhost:8000/

3.3 Cài đặt Frontend (React+Vite)

3.3.1 Chuyển sang thư mục frontend và cài đặt

```
cd ../frontend
npm install
```

3.3.2 Khởi động frontend

```
npm run dev
```

Giao diện người dùng sẽ chạy tại http://localhost:5173/

4. Sử dụng ứng dụng

Đăng ký / Đăng nhập: Truy cập trang đăng nhập hoặc đăng ký tại /login hoặc /register

Trang quản trị: Truy cập /admin bằng tài khoản superuser để quản lý bài hát và nghệ sĩ

Thư viện người dùng: Quản lý album và bài hát cá nhân tại /my-albums

Trình phát nhạc: Phát bài hát và chuyển playlist thông qua trình phát ở dưới cùng màn hình

PHẦN 5: TỔNG KẾT

Dự án Spotify Clone là một bài học thực tiễn trong việc xây dựng ứng dụng web đa chức năng với mô hình client-server. Mục tiêu ban đầu là tái hiện một nền tảng nghe nhạc trực tuyến có giao diện thân thiện, hỗ trợ phát nhạc, MV, tạo playlist, và quản lý nội dung người dùng.

Thông qua việc sử dụng React+Vite ở frontend và Django Rest Framework ở backend, em đã có cơ hội làm việc với REST API, định tuyến, quản lý trạng thái giao diện và xử lý dữ liệu người dùng. Đây là cơ hội tốt để hiểu rõ hơn về cách các phần trong một hệ thống web hiện đại phối hợp với nhau.

Tuy nhiên, trong quá trình triển khai, em cũng nhận ra một số hạn chế và lỗi còn tồn tại, bao gồm:

- Giao diện chưa tối ưu hoàn toàn cho thiết bị di động.
- Quản lý trạng thái và cập nhật UI còn chưa mượt trong một số thao tác (như tạo playlist).
- Chưa xử lý tốt các trường hợp ngoại lệ (như lỗi mạng, dữ liệu rỗng).
- Cấu trúc backend có thể được tách gọn và tối ưu hoá hơn cho quy mô mở rộng sau này.

Những vấn đề trên giúp em nhận ra tầm quan trọng của việc kiểm thử kỹ lưỡng, xử lý lỗi hợp lý, và thiết kế kiến trúc linh hoạt ngay từ đầu.

Dù còn nhiều điểm chưa hoàn thiện, song dự án đã giúp em rèn luyện tư duy phân tích, kỹ năng giải quyết vấn đề và tính kiên nhẫn khi làm việc độc lập. Đây cũng là tiền đề để em tiếp tục cải tiến sản phẩm và phát triển các dự án phức tạp hơn trong tương lai.