

TRƯỜNG ĐẠI HỌC SÀI GÒN  
KHOA CÔNG NGHỆ THÔNG TIN



## PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

---

Đề Tài

# Phát triển phần mềm Spotify Clone

---

GVHD: Từ Lăng Phiêu

SV: Quách Gia Quy - 3121410411 - quyquach22092003@gmail.com  
Nguyễn Quang Hà - 3121410170 - haquangnguyen30@gmail.com  
Nguyễn Công Đức - 3121410161 - nguyencongduc18THD@gmail.com  
Lê Thiên Phúc - 3121410392 - lethienphuc31103@gmail.com

TP. HỒ CHÍ MINH, THÁNG 5/2025

# Mục lục

<b>1</b>	<b>Phần giới thiệu</b>	<b>3</b>
1.1	Mục tiêu đề tài . . . . .	3
1.2	Phạm vi ứng dụng . . . . .	3
1.3	Đối tượng hướng tới . . . . .	3
<b>2</b>	<b>Cơ sở lý thuyết</b>	<b>4</b>
2.1	Các công nghệ và thư viện sử dụng . . . . .	4
2.1.1	Frontend . . . . .	4
2.1.2	Backend . . . . .	4
2.1.3	Database . . . . .	4
2.1.4	Các công nghệ hỗ trợ và Deployment . . . . .	5
2.2	Kiến trúc hệ thống . . . . .	5
2.2.1	Tổng quan kiến trúc . . . . .	5
2.2.2	Các thành phần chính . . . . .	5
2.2.3	Luồng dữ liệu . . . . .	6
2.2.4	Kiến trúc Deployment . . . . .	7
2.3	Phân tích và thiết kế hệ thống . . . . .	9
2.3.1	Mô hình MVC và RESTful API . . . . .	9
2.3.2	Quy trình xác thực người dùng . . . . .	9
2.3.3	Quy trình phát nhạc . . . . .	10
2.3.4	Quy trình phát video . . . . .	11
2.4	Mô hình cơ sở dữ liệu . . . . .	12
2.4.1	Các bảng dữ liệu chính . . . . .	12
2.4.2	Mối quan hệ giữa các bảng . . . . .	13
2.4.3	Thiết kế lưu trữ file media . . . . .	14
2.5	Các tính năng chính . . . . .	14
2.5.1	Phát nhạc trực tuyến . . . . .	14
2.5.2	Phát video âm nhạc . . . . .	15
2.5.3	Tạo và quản lý playlist . . . . .	15
2.5.4	Quản lý thư viện cá nhân . . . . .	15
2.5.5	Tải nhạc và video . . . . .	15
2.5.6	Tìm kiếm và khám phá . . . . .	15
2.5.7	Tương tác xã hội . . . . .	16
2.5.8	Giao diện quản trị . . . . .	16
<b>3</b>	<b>HIỆN THỰC</b>	<b>17</b>
3.1	Giao diện người dùng (Frontend) . . . . .	17
3.2	Giao diện quản trị viên (Admin Frontend) . . . . .	21
3.3	Mã nguồn (Backend và Frontend) . . . . .	24
3.3.1	Cấu trúc mã nguồn Frontend . . . . .	24
3.3.2	Cấu trúc mã nguồn Admin Frontend . . . . .	25
3.3.3	Cấu trúc mã nguồn Backend (Django) . . . . .	26
3.3.4	Các chức năng đã triển khai . . . . .	28



<b>4</b>	<b>Cách Thức Cài Đặt Và Chạy Ứng Dụng</b>	<b>29</b>
4.1	Môi trường chạy ứng dụng . . . . .	29
4.2	Hướng dẫn cài đặt . . . . .	29
4.2.1	Clone source code từ GitHub . . . . .	29
4.2.2	Cài đặt Backend . . . . .	29
4.2.3	Cài đặt Frontend . . . . .	30
4.2.4	Cài đặt Admin Frontend . . . . .	30
4.3	Hướng dẫn chạy ứng dụng . . . . .	30
4.3.1	Khởi chạy Backend . . . . .	30
4.3.2	Khởi chạy Frontend . . . . .	30
4.3.3	Khởi chạy Admin Frontend . . . . .	30
4.3.4	Truy cập ứng dụng . . . . .	31
<b>5</b>	<b>Phân công nhiệm vụ nhóm</b>	<b>32</b>
5.1	Vai trò và nhiệm vụ các thành viên . . . . .	32



# 1 Phần giới thiệu

## 1.1 Mục tiêu đề tài

Đề án này tập trung vào mục tiêu thiết kế giao diện web và xây dựng các chức năng cơ bản của một ứng dụng nghe nhạc và xem video âm nhạc tương tự như Spotify. Ứng dụng được phát triển để có thể triển khai và sử dụng trên hệ điều hành Linux mã nguồn mở. Các mục tiêu cụ thể bao gồm:

- Xây dựng chức năng phát nhạc trực tuyến với giao diện trực quan, cho phép người dùng tìm kiếm, lựa chọn và điều khiển việc phát các bài hát.
- Phát triển chức năng phát video âm nhạc, hỗ trợ hiển thị video và điều khiển các tùy chọn phát.
- Hiện thực hóa chức năng cho phép người dùng tải xuống các video âm nhạc để nghe hoặc xem offline.
- Cung cấp khả năng cho người dùng đã đăng ký tạo và quản lý các playlist nhạc cá nhân, cũng như lưu trữ các bài hát yêu thích.
- Xây dựng một trang quản trị (Admin) cho phép người quản lý hệ thống thực hiện các tác vụ như quản lý người dùng, bài hát, album và các nội dung khác.

## 1.2 Phạm vi ứng dụng

Phần mềm Spotify Clone này được xây dựng trong phạm vi một đề án học tập, do đó, phạm vi ứng dụng chính tập trung vào việc trình bày các kiến thức và kỹ năng đã học trong môn học. Ứng dụng hướng đến khả năng chạy trên các hệ thống sử dụng hệ điều hành Linux mã nguồn mở, thể hiện khả năng phát triển ứng dụng đa nền tảng (trong giới hạn của các công nghệ được sử dụng). Mặc dù mô phỏng các chức năng của Spotify, đề án này không đặt mục tiêu thay thế hoặc cạnh tranh với các nền tảng thương mại hiện có. Thay vào đó, nó tập trung vào việc hiện thực hóa các tính năng cốt lõi để minh họa khả năng phát triển phần mềm web của nhóm sinh viên.

## 1.3 Đối tượng hướng tới

Đối tượng hướng tới chính của phần mềm Spotify Clone này là:

- **Người dùng cuối:** Những người sử dụng ứng dụng để nghe nhạc, xem video âm nhạc, tạo playlist cá nhân và lưu trữ bài hát yêu thích. Giao diện người dùng được thiết kế để thân thiện và dễ sử dụng trên các trình duyệt web trên hệ điều hành Linux.
- **Quản trị viên hệ thống:** Người có quyền truy cập vào trang Admin để quản lý nội dung, người dùng và các khía cạnh khác của ứng dụng. Trang Admin được thiết kế để cung cấp các công cụ quản lý hiệu quả.



## 2 Cơ sở lý thuyết

### 2.1 Các công nghệ và thư viện sử dụng

Nhóm chúng tôi đã sử dụng các công nghệ và thư viện hiện đại để phát triển ứng dụng Spotify Clone với kiến trúc chia thành 3 lớp chính: Frontend, Backend và Database. Dưới đây là các công nghệ chính được sử dụng:

#### 2.1.1 Frontend

- **ReactJS**: Thư viện JavaScript mã nguồn mở được phát triển bởi Facebook (nay là Meta), cho phép xây dựng giao diện người dùng linh hoạt với cơ chế component-based và virtual DOM giúp tối ưu hiệu suất.
- **Material UI (MUI)**: Thư viện UI components cho React tuân theo nguyên tắc thiết kế Material Design của Google, cung cấp các components đẹp mắt, dễ tùy biến và responsive.
- **React Router**: Thư viện quản lý routing trong ứng dụng single-page, cho phép điều hướng giữa các trang mà không cần tải lại toàn bộ trang.
- **Axios**: Thư viện dựa trên Promise giúp thực hiện các HTTP requests từ frontend tới backend API một cách đơn giản và hiệu quả.
- **Tailwind CSS**: Framework CSS utility-first giúp thiết kế giao diện responsive nhanh chóng thông qua các lớp định sẵn, hỗ trợ tùy biến cao và tối ưu hiệu suất.

#### 2.1.2 Backend

- **Django**: Framework web Python mạnh mẽ theo mô hình MTV (Model-Template-View), cung cấp cấu trúc project rõ ràng và nhiều tính năng bảo mật.
- **Django REST Framework**: Bộ công cụ mạnh mẽ để xây dựng Web API trên nền tảng Django, hỗ trợ các tính năng như authentication, serialization, viewsets và permissions.
- **Django CORS Headers**: Middleware cho phép cross-origin requests từ frontend đến backend API.
- **SimpleJWT**: Thư viện cung cấp JSON Web Token authentication cho Django REST Framework, giúp xác thực người dùng an toàn.
- **Pillow**: Thư viện xử lý hình ảnh trong Python, được sử dụng để xử lý ảnh album, avatar người dùng.

#### 2.1.3 Database

- **MySQL**: Hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở phổ biến, được sử dụng để lưu trữ thông tin người dùng, bài hát, album và playlist.
- **Django ORM**: Công cụ ánh xạ đối tượng-quan hệ (Object-Relational Mapping) tích hợp sẵn trong Django, giúp tương tác với database thông qua các models Python mà không cần viết SQL trực tiếp.

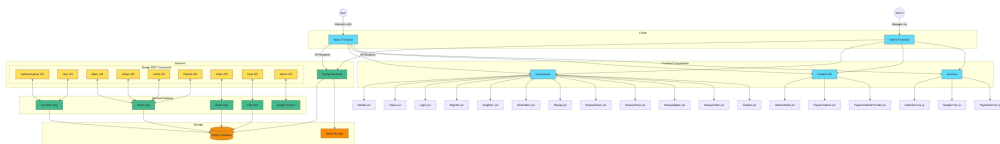
#### 2.1.4 Các công nghệ hỗ trợ và Deployment

- **Git**: Hệ thống quản lý phiên bản phân tán, giúp quản lý mã nguồn và hỗ trợ làm việc nhóm hiệu quả.
- **AWS (Amazon Web Services)**: Nền tảng điện toán đám mây được sử dụng để triển khai và vận hành ứng dụng Spotify Clone, cụ thể:
  - **EC2 (Elastic Compute Cloud)**: Dịch vụ cung cấp máy chủ ảo đàn hồi để chạy ứng dụng backend với khả năng mở rộng linh hoạt theo nhu cầu.
  - **EBS (Elastic Block Store)**: Dịch vụ lưu trữ block-level hiệu năng cao, thường được sử dụng làm ổ đĩa gắn liền với các instance EC2.
  - **VPC (Virtual Private Cloud)**: Dịch vụ mạng ảo cho phép tạo môi trường mạng riêng biệt, bảo mật và kiểm soát hoàn toàn các tài nguyên đám mây.
  - **Data Transfer**: Dịch vụ quản lý lưu lượng truyền dữ liệu vào/ra giữa các dịch vụ AWS và internet, bao gồm:
    - \* Regional Data Transfer: Truyền dữ liệu giữa các dịch vụ trong cùng khu vực AWS
    - \* Data Transfer Out: Truyền dữ liệu từ AWS ra internet

## 2.2 Kiến trúc hệ thống

Hệ thống Spotify Clone được thiết kế theo mô hình kiến trúc 3 tầng (3-tier architecture), với sự phân tách rõ ràng giữa giao diện người dùng, xử lý logic nghiệp vụ và lưu trữ dữ liệu. Kiến trúc này cho phép ứng dụng dễ dàng mở rộng, bảo trì và cải tiến từng thành phần một cách độc lập.

### 2.2.1 Tổng quan kiến trúc



Hình 1: Flowchart tổng quan kiến trúc hệ thống Spotify Clone

### 2.2.2 Các thành phần chính

- **Client Layer (Frontend)**:



- Giao diện người dùng được xây dựng bằng ReactJS và Material UI (MUI)
- Quản lý trạng thái ứng dụng thông qua Redux
- Xử lý routing với React Router
- Tương tác với API thông qua Axios
- Phát nhạc và video với HTML5 Audio/Video API và các thư viện hỗ trợ

- **API Layer (Backend):**

- Django REST Framework xử lý các request và response
- Authentication sử dụng JWT (JSON Web Tokens)
- Business Logic xử lý các yêu cầu nghiệp vụ
- Media Processing xử lý file âm thanh và video
- Serializers chuyển đổi dữ liệu giữa JSON và Python objects

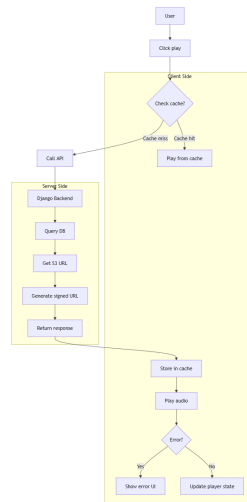
- **Data Layer:**

- MySQL Database lưu trữ dữ liệu có cấu trúc
- AWS S3 lưu trữ file media (audio, video, hình ảnh)
- Django ORM kết nối và thao tác với database

### 2.2.3 Luồng dữ liệu

Khi người dùng tương tác với ứng dụng, luồng dữ liệu sẽ diễn ra như sau:

1. Người dùng tương tác với giao diện React
2. Frontend gửi HTTP request đến Backend API
3. API xác thực người dùng thông qua JWT
4. Backend xử lý logic nghiệp vụ và tương tác với Database
5. Kết quả được trả về Frontend dưới dạng JSON
6. Frontend cập nhật UI dựa trên dữ liệu nhận được
7. Trong trường hợp phát nhạc/video, file media được stream trực tiếp từ AWS S3

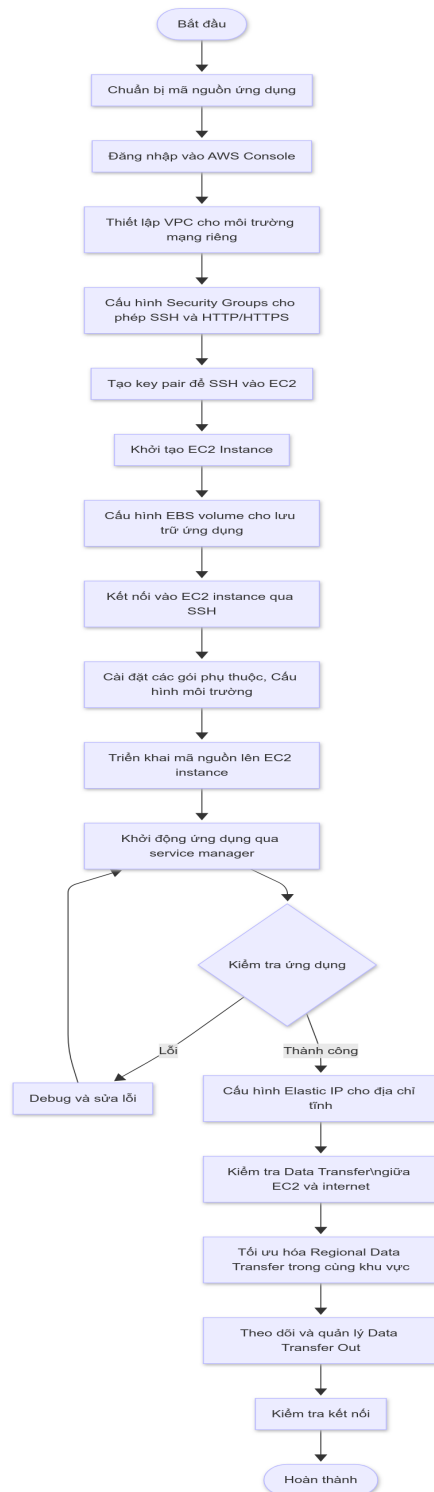


Hình 2: Flowchart luồng dữ liệu trong hệ thống Spotify Clone

#### 2.2.4 Kiến trúc Deployment

Như minh họa trong Hình 3, luồng dữ liệu trong hệ thống AWS EC2 triển khai theo quy trình tối ưu bao gồm: chuẩn bị mã nguồn, thiết lập VPC và Security Groups đảm bảo bảo mật, khởi tạo EC2 kèm EBS cho lưu trữ, và cấu hình Data Transfer hiệu quả. Hệ thống quản lý dữ liệu qua Regional Data Transfer và Data Transfer Out, cùng với Elastic IP cho kết nối ổn định, tạo nên trải nghiệm người dùng mượt mà và hiệu suất cao.





Hình 3: Kiến trúc triển khai trên AWS

## 2.3 Phân tích và thiết kế hệ thống

### 2.3.1 Mô hình MVC và RESTful API

Ứng dụng Spotify Clone được thiết kế dựa trên mô hình MVC (Model-View-Controller) hiện đại, kết hợp với kiến trúc RESTful API để tạo sự tách biệt rõ ràng giữa các thành phần:

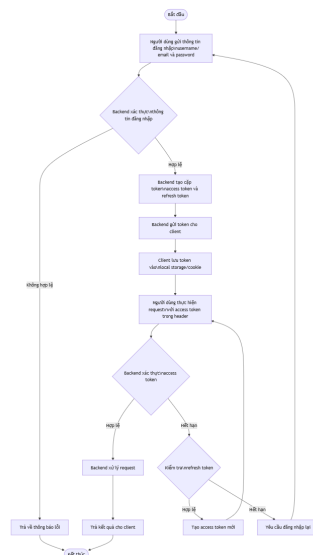
- **Model:** Các models trong Django đại diện cho cấu trúc dữ liệu và logic nghiệp vụ. Mỗi model tương ứng với một bảng trong cơ sở dữ liệu và định nghĩa các trường dữ liệu, quan hệ và ràng buộc.
- **View:** Trong kiến trúc của chúng tôi, giao diện người dùng được xây dựng bằng ReactJS đảm nhận vai trò View, hiển thị dữ liệu và tương tác với người dùng.
- **Controller:** Django REST Framework đóng vai trò Controller, xử lý các request từ client, tương tác với Model và trả về response phù hợp.

RESTful API được thiết kế với các nguyên tắc:

- **Stateless:** Mỗi request chứa đầy đủ thông tin cần thiết, không phụ thuộc vào session.
- **Resource-oriented:** Mỗi tài nguyên (user, song, album, playlist) có các endpoint riêng.
- **Standard HTTP methods:** Sử dụng GET, POST, PUT, DELETE cho các thao tác CRUD.
- **JSON format:** Dữ liệu được truyền tải dưới dạng JSON để dễ dàng xử lý ở cả client và server.

### 2.3.2 Quy trình xác thực người dùng

Hệ thống xác thực người dùng dựa trên JWT (JSON Web Token) đảm bảo tính bảo mật và hiệu suất cao. Quy trình đăng nhập và xác thực diễn ra như sau:



Hình 4: Quy trình xác thực người dùng

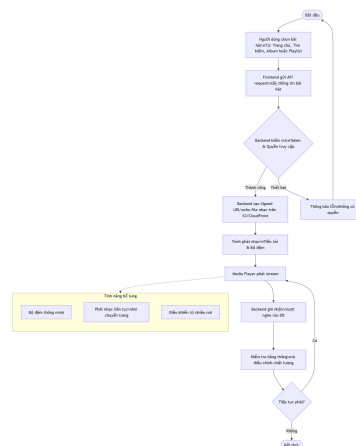
Quy trình xác thực JWT bao gồm:

- **Đăng nhập:** Người dùng gửi thông tin đăng nhập (username/email và password)
- **Xác thực:** Backend xác thực thông tin đăng nhập với database
- **Tạo token:** Nếu thông tin chính xác, backend tạo cặp token (access token và refresh token)
- **Gửi token:** Backend gửi token cho client và lưu trữ ở local storage hoặc cookie
- **Sử dụng token:** Client gửi access token trong header của mỗi request
- **Xác thực token:** Backend xác thực token trước khi xử lý request
- **Làm mới token:** Khi access token hết hạn, client sử dụng refresh token để lấy access token mới

Access token có thời hạn ngắn (15-30 phút) để đảm bảo an toàn, trong khi refresh token có thời hạn dài hơn (7-30 ngày).

### 2.3.3 Quy trình phát nhạc

Quy trình phát nhạc được thiết kế để đảm bảo trải nghiệm nghe nhạc liền mạch và chất lượng cao cho người dùng:



Hình 5: Quy trình phát nhạc từ lựa chọn đến phát

Quy trình này bao gồm các bước chính:

- **Lựa chọn bài hát:** Người dùng chọn bài hát từ giao diện (trang chủ, tìm kiếm, album, playlist)
- **Yêu cầu thông tin bài hát:** Frontend gửi request API để lấy thông tin chi tiết và URL của file âm thanh
- **Xác thực và phân quyền:** Backend kiểm tra token và quyền truy cập
- **Tạo URL truy cập:** Backend tạo URL có thời hạn (Signed URL) để truy cập file từ AWS S3/CloudFront

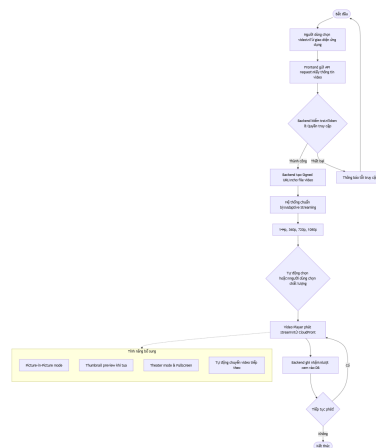
- **Tiền tải và bộ nhớ đệm:** Trình phát nhạc tải trước một phần bài hát để giảm độ trễ
- **Phát nhạc:** Media Player của trình duyệt phát file âm thanh dưới dạng stream
- **Cập nhật lượt nghe:** Backend ghi nhận lượt nghe vào database
- **Streaming thích ứng:** Điều chỉnh chất lượng stream dựa trên băng thông của người dùng (adaptive streaming)

Ngoài ra, hệ thống còn hỗ trợ:

- Bộ nhớ đệm thông minh để giảm lượng dữ liệu tải xuống
- Tiếp tục phát nhạc khi chuyển trang (persistent playback)
- Điều khiển phát nhạc từ nhiều nơi trong ứng dụng

### 2.3.4 Quy trình phát video

Tương tự như phát nhạc, quy trình phát video được tối ưu hóa cho trải nghiệm người dùng, với các tính năng bổ sung đặc biệt cho nội dung video:



Hình 6: Quy trình phát video

Quy trình phát video bao gồm:

- **Chọn video:** Người dùng chọn video âm nhạc từ giao diện
- **Yêu cầu thông tin video:** Frontend gửi request API để lấy thông tin và URL video
- **Xác thực và phân quyền:** Backend kiểm tra token và quyền truy cập
- **Tạo URL truy cập:** Backend tạo URL có thời hạn để truy cập file video
- **Adaptive Streaming:** Hệ thống cung cấp nhiều phiên bản video với chất lượng khác nhau (144p, 360p, 720p, 1080p)
- **Chọn chất lượng video:** Hệ thống tự động chọn hoặc người dùng có thể tự chọn chất lượng video phù hợp



- **Phát video:** Video Player phát video dưới dạng streaming từ AWS CloudFront
- **Cập nhật lượt xem:** Backend ghi nhận lượt xem vào database

Các tính năng bổ sung cho phát video:

- Picture-in-Picture mode cho phép người dùng xem video trong khi duyệt các phần khác của ứng dụng
- Video thumbnail previews khi tua video
- Chế độ Theater mode và Fullscreen
- Tự động chuyển sang video tiếp theo trong playlist

## 2.4 Mô hình cơ sở dữ liệu

Ứng dụng Spotify Clone sử dụng mô hình cơ sở dữ liệu quan hệ được triển khai trên MySQL. Mô hình này được thiết kế để tối ưu cho việc lưu trữ và truy xuất các dữ liệu liên quan đến người dùng, nghệ sĩ, bài hát, album và các tương tác.

### 2.4.1 Các bảng dữ liệu chính

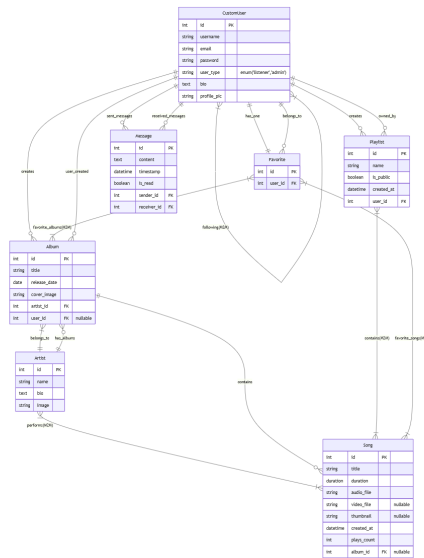
- **CustomUser:** Mở rộng từ AbstractUser của Django, lưu thông tin người dùng
  - `id`, `username`, `email`, `password`: Thông tin tài khoản cơ bản
  - `user_type`: Phân loại người dùng ('listener' hoặc 'admin')
  - `bio`, `profile_pic`: Thông tin cá nhân
  - `following`: Quan hệ many-to-many với chính nó, theo dõi người dùng khác
- **Artist:** Lưu thông tin các nghệ sĩ
  - `id`, `name`: Định danh và tên nghệ sĩ
  - `bio`: Thông tin tiểu sử
  - `image`: Ảnh đại diện nghệ sĩ
- **Album:** Lưu thông tin các album âm nhạc
  - `id`, `title`: Định danh và tên album
  - `artist_id`: Khóa ngoại liên kết với nghệ sĩ
  - `release_date`: Ngày phát hành
  - `cover_image`: Ảnh bìa album
  - `user_id`: Khóa ngoại liên kết với người dùng (đối với album do người dùng tạo)
- **Song:** Thông tin các bài hát
  - `id`, `title`: Định danh và tên bài hát
  - `album_id`: Khóa ngoại liên kết với album (có thể null)
  - `artists`: Quan hệ many-to-many với Artist (một bài hát có thể do nhiều nghệ sĩ thể hiện)



- **duration**: Thời lượng bài hát
- **audio\_file**: Đường dẫn đến file âm thanh
- **video\_file**: Đường dẫn đến file video (nếu có)
- **thumbnail**: Ảnh thu nhỏ của bài hát
- **created\_at**: Thời gian tạo bài hát
- **plays\_count**: Số lượt phát
- **Playlist**: Danh sách phát do người dùng tạo
  - **id, name**: Định danh và tên playlist
  - **user\_id**: Khóa ngoại liên kết với người dùng sở hữu
  - **songs**: Quan hệ many-to-many với Song
  - **is\_public**: Trạng thái công khai của playlist
  - **created\_at**: Thời gian tạo playlist
- **Favorite**: Lưu trữ danh sách yêu thích của người dùng
  - **id, user\_id**: Định danh và liên kết với người dùng
  - **songs**: Quan hệ many-to-many với Song (bài hát yêu thích)
  - **albums**: Quan hệ many-to-many với Album (album yêu thích)

#### 2.4.2 Mối quan hệ giữa các bảng

- Một **User** có thể:
  - Theo dõi nhiều **User** khác (quan hệ self-referential many-to-many)
  - Tạo nhiều **Album**
  - Tạo nhiều **Playlist**
  - Có một **Favorite** duy nhất
- Một **Artist** có thể:
  - Tạo nhiều **Album**
  - Thể hiện nhiều **Song**
- Một **Album**:
  - Thuộc về một **Artist**
  - Có thể thuộc về một **User** (trong trường hợp album do người dùng tạo)
  - Chứa nhiều **Song**
  - Có thể được yêu thích bởi nhiều **Favorite**
- Một **Song**:
  - Có thể thuộc về một **Album** (hoặc không thuộc album nào)
  - Có thể được thể hiện bởi nhiều **Artist**
  - Có thể thuộc về nhiều **Playlist**
  - Có thể được yêu thích bởi nhiều **Favorite**



Hình 7: Sơ đồ mô hình cơ sở dữ liệu

### 2.4.3 Thiết kế lưu trữ file media

Trong mô hình dữ liệu, các file media như nhạc, video và hình ảnh được lưu trữ theo cách sau:

- **Ảnh đại diện người dùng**: Được lưu trữ trong thư mục `profile_pics/`
- **Ảnh nghệ sĩ**: Được lưu trữ trong thư mục `artists/`
- **Ảnh bìa album**: Được lưu trữ trong thư mục `albums/`
- **File âm thanh**: Được lưu trữ trong thư mục `songs/audio/`
- **File video**: Được lưu trữ trong thư mục `songs/video/`
- **Ảnh thu nhỏ bài hát**: Được lưu trữ trong thư mục `songs/thumbnails/`

Trong môi trường sản xuất, các file này được lưu trữ trên AWS S3 và được phân phối thông qua CloudFront CDN để tối ưu hiệu suất.

## 2.5 Các tính năng chính

Ứng dụng Spotify Clone được thiết kế với nhiều tính năng chính để mang lại trải nghiệm tốt nhất cho người dùng:

### 2.5.1 Phát nhạc trực tuyến

- Phát nhạc với chất lượng cao
- Điều khiển phát nhạc: play, pause, skip, repeat, shuffle
- Hiển thị thông tin bài hát đang phát



- Điều chỉnh âm lượng và equalizer
- Tiếp tục phát nhạc khi chuyển trang

### 2.5.2 Phát video âm nhạc

- Xem video âm nhạc với nhiều độ phân giải
- Chế độ Theater và Fullscreen
- Picture-in-Picture mode
- Tự động chuyển sang video tiếp theo

### 2.5.3 Tạo và quản lý playlist

- Tạo playlist cá nhân
- Thêm/xóa bài hát vào playlist
- Sắp xếp thứ tự bài hát
- Chia sẻ playlist với người dùng khác
- Playlist công khai và riêng tư

### 2.5.4 Quản lý thư viện cá nhân

- Yêu thích bài hát và album
- Theo dõi nghệ sĩ
- Xem lịch sử nghe nhạc
- Đề xuất nội dung dựa trên thói quen nghe

### 2.5.5 Tải nhạc và video

- Tải nhạc để nghe offline
- Tải video để xem offline
- Quản lý không gian lưu trữ
- Đồng bộ hóa nội dung đã tải xuống

### 2.5.6 Tìm kiếm và khám phá

- Tìm kiếm bài hát, album, nghệ sĩ, playlist
- Tìm kiếm theo lời bài hát
- Khám phá nhạc mới phát hành
- Đề xuất nhạc theo thể loại, tâm trạng, hoạt động





#### **2.5.7 Tương tác xã hội**

- Theo dõi bạn bè
- Xem hoạt động nghe nhạc của bạn bè
- Chia sẻ bài hát, album, playlist
- Tích hợp với mạng xã hội

#### **2.5.8 Giao diện quản trị**

- Quản lý người dùng
- Quản lý nội dung (bài hát, album, nghệ sĩ)
- Xem thống kê và báo cáo
- Quản lý quyền và vai trò

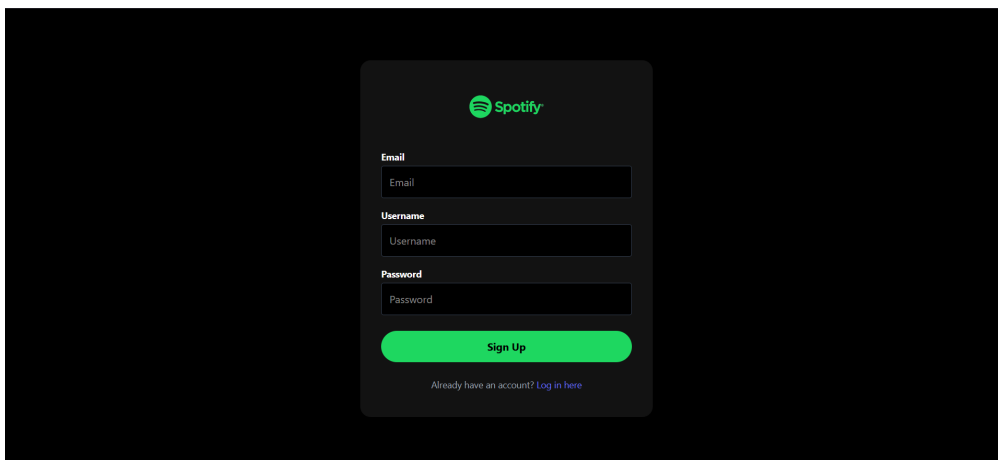
## 3 HIỆN THỰC

### 3.1 Giao diện người dùng (Frontend)

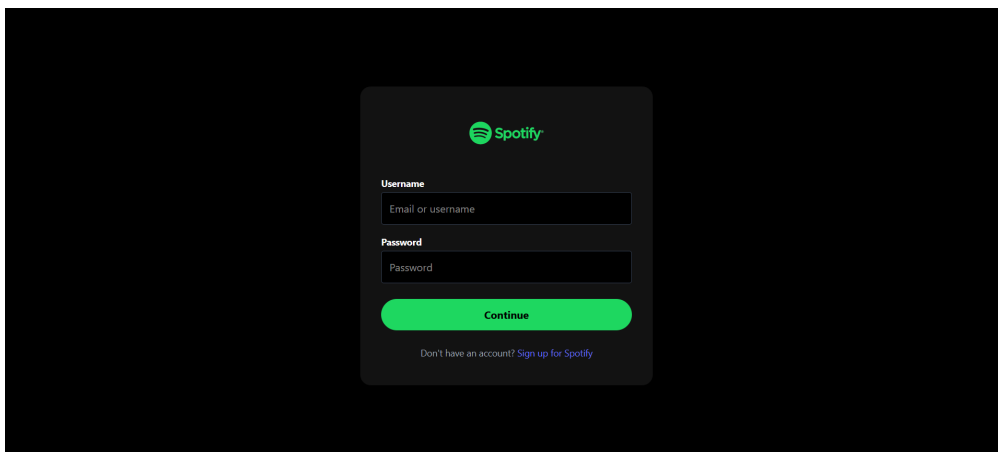
Giao diện người dùng của ứng dụng được phát triển bằng ReactJS kết hợp với Ant Design và Tailwind CSS, đảm bảo trải nghiệm người dùng mượt mà, hiện đại và tương thích trên nhiều thiết bị.

Các trang giao diện chính bao gồm:

- **Trang đăng nhập / đăng ký:** Cho phép người dùng tạo tài khoản và đăng nhập bằng tài khoản đã có.

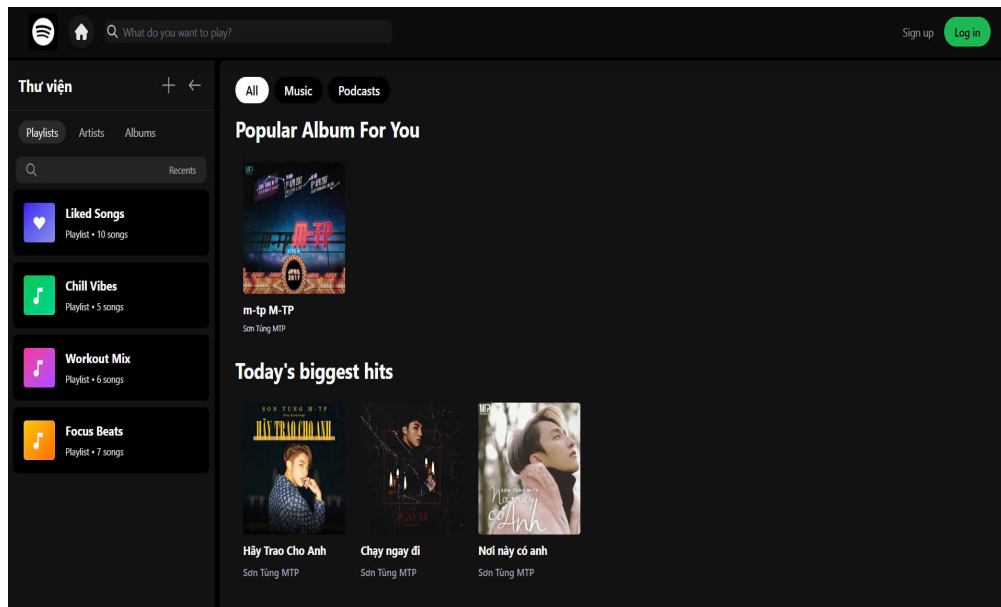


Hình 8: Giao diện đăng ký



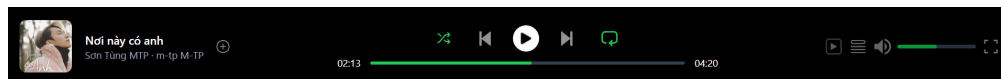
Hình 9: Giao diện đăng nhập

- **Trang chính (home):** Hiển thị các bài hát, album nổi bật, danh sách gợi ý cho người dùng.

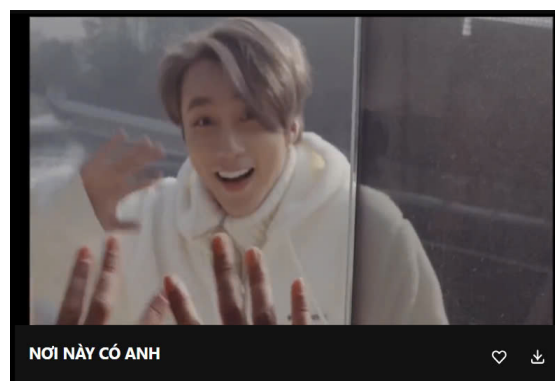


Hình 10: Trang chính hiển thị danh sách bài hát

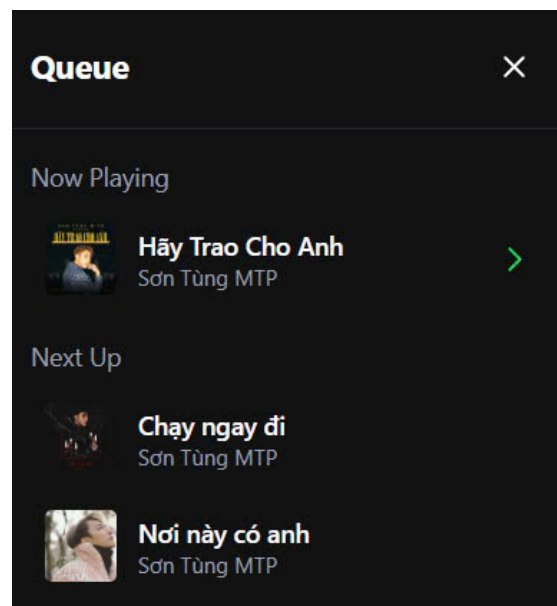
- **Thanh điều khiển phát nhạc:** Giao diện phát nhạc với các nút điều khiển, hiển thị thông tin bài hát đang phát, xem video và danh sách chờ.



Hình 11: Giao diện thanh phát nhạc

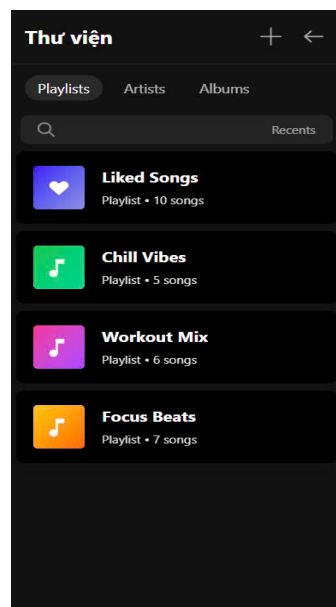


Hình 12: Giao diện phát video

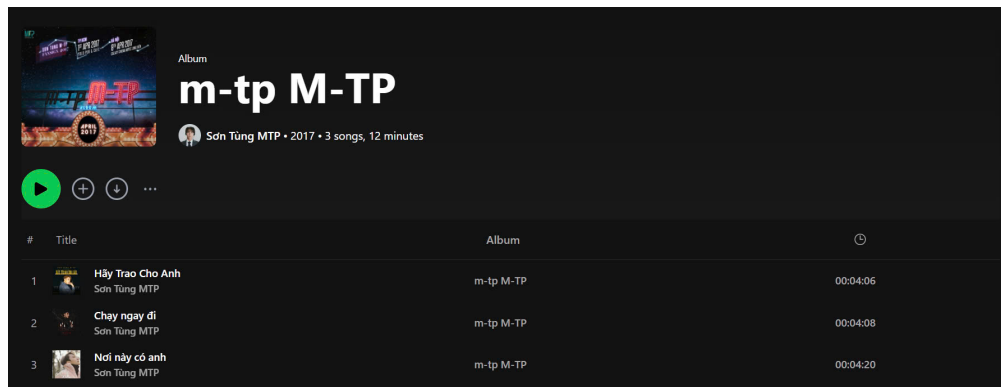


Hình 13: Giao diện danh sách chờ

- **Trang quản lý playlist và album:** Cho phép người dùng tạo, chỉnh sửa danh sách phát cá nhân.



Hình 14: Giao diện quản lý playlist



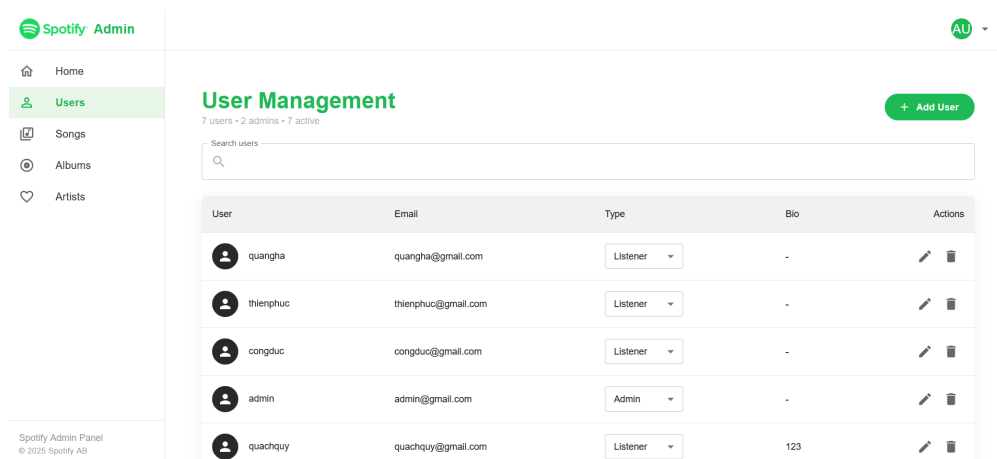
Hình 15: Giao diện quản lý album

### 3.2 Giao diện quản trị viên (Admin Frontend)

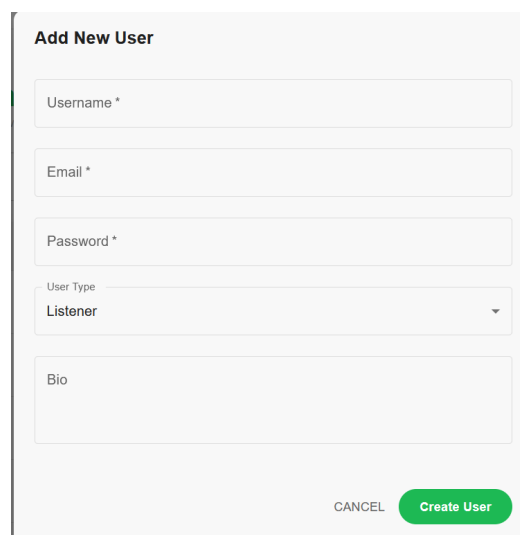
Giao diện quản trị viên được xây dựng bằng ReactJS kết hợp với MUI Components, cung cấp các công cụ quản lý hệ thống mạnh mẽ với giao diện chuyên nghiệp và dễ sử dụng.

Các trang giao diện chính bao gồm:

- **Trang quản lý người dùng:** Hiển thị danh sách người dùng, thông tin tài khoản và các chức năng quản lý.



Hình 16: Giao diện quản lý người dùng



**Add New User**

Listener

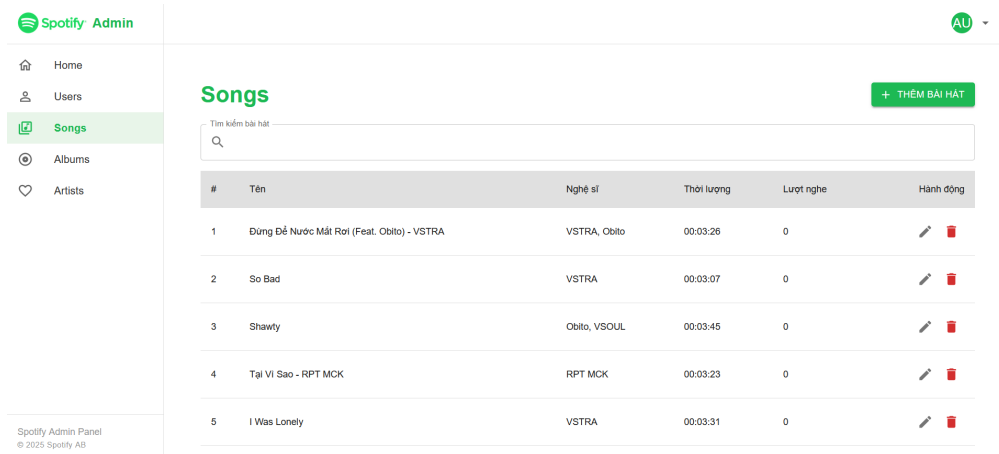
CANCEL

Create User

Hình 17: Giao diện thêm mới người dùng



- **Trang quản lý bài hát:** Cho phép thêm mới, chỉnh sửa và xóa các bài hát trong hệ thống.



Hình 18: Giao diện quản lý bài hát

Thêm bài hát

Tên bài hát

Nghệ sĩ

Album

Thời lượng (HH:MM:SS)

Lượt nghe  
0

Tập tin:

FILE ÂM THANH

FILE VIDEO (TỰY CHỌN)

HÌNH ĐẠI DIỆN (TỰY CHỌN)

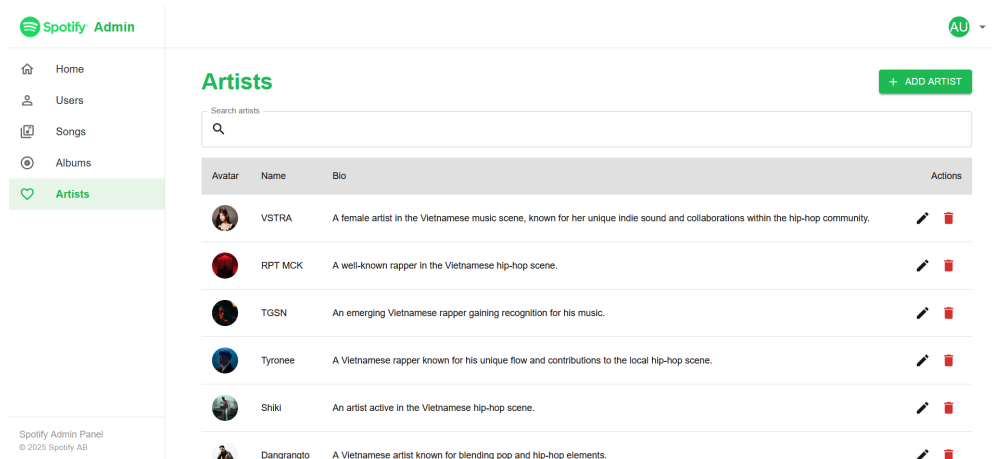
HỦY

THÊM

Hình 19: Giao diện thêm mới bài hát



- **Trang quản lý nghệ sĩ:** Cung cấp công cụ quản lý thông tin nghệ sĩ, album và các tác phẩm liên quan.



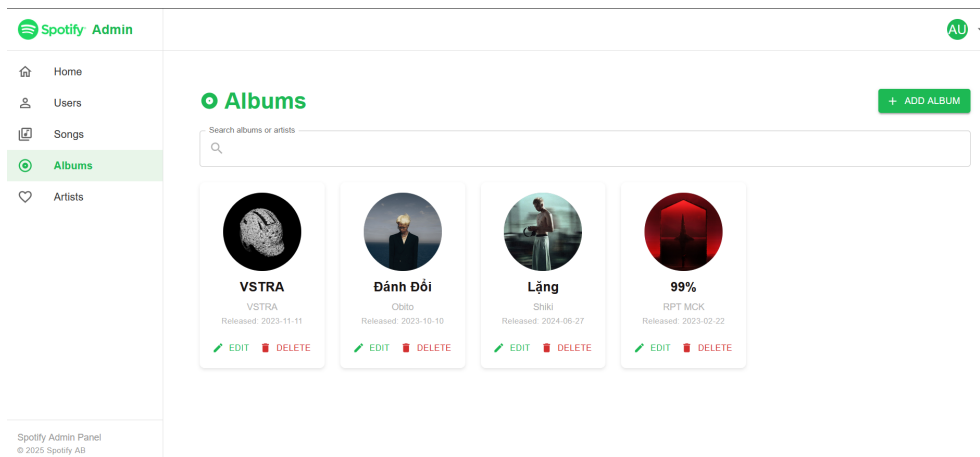
Hình 20: Giao diện quản lý nghệ sĩ

### Add Artist

Hình 21: Giao diện quản lý nghệ sĩ



- **Trang quản lý playlist:** Theo dõi và quản lý các playlist do người dùng tạo.



Hình 22: Giao diện quản lý playlist hệ thống

Giao diện admin được thiết kế với các thành phần chuyên biệt:

- Sidebar điều hướng với các module quản trị
- Data tables với chức năng phân trang, sắp xếp và lọc
- Form nhập liệu với validation
- Hệ thống thông báo và logs hoạt động
- Dashboard tổng quan với các chỉ số quan trọng

### 3.3 Mã nguồn (Backend và Frontend)

#### 3.3.1 Cấu trúc mã nguồn Frontend

- **/src:** Thư mục chứa mã nguồn chính của ứng dụng frontend.
  - **/api:** Module giao tiếp với backend API.
    - \* **auth.js:** Xử lý các API liên quan đến xác thực (login, getCurrentUser).
    - \* **axiosInstance.js:** Cấu hình cơ bản cho Axios (baseUrl, headers).
  - **/assets:** Lưu trữ tài nguyên tĩnh.
  - **/components:** Chứa các thành phần giao diện người dùng tái sử dụng.
    - \* **AlbumItem.jsx:** Component hiển thị thông tin và giao diện tương tác cho một album nhạc.
    - \* **Display.jsx:** Component bao bọc chính cho các view hiển thị.
    - \* **DisplayAlbum.jsx:** Giao diện hiển thị danh sách album.
    - \* **DisplayVideo.jsx:** Component chuyên biệt cho phát video âm nhạc.
    - \* **Login.jsx:** Form đăng nhập với các trường nhập liệu và xác thực.
    - \* **Navbar.jsx:** Thanh điều hướng chính của ứng dụng.

- \* **Player.jsx**: Thanh điều khiển phát nhạc với các chức năng play/pause, next/previous, volume control.
  - \* **Register.jsx**: Form đăng ký tài khoản người dùng mới.
  - \* **Sidebar.jsx**: Menu điều hướng phụ chứa các playlist của người dùng.
  - \* **SongItem.jsx**: Component hiển thị thông tin và tương tác cho một bài hát đơn lẻ.
  - **/context**: Quản lý trạng thái ứng dụng thông qua React Context API.
    - \* **AuthContext.jsx**: Quản lý trạng thái đăng nhập và thông tin người dùng.
    - \* **Login.jsx**: Kiểm soát trạng thái xác thực (đăng nhập/đăng xuất) toàn cục.
    - \* **PlayerContext.jsx**: Quản lý trạng thái phát nhạc toàn cục.
    - \* **PlayerContextProvider.jsx**: Cung cấp PlayerContext cho các component con.
  - **/service**: Các tiện ích hỗ trợ.
    - \* **AlbumService.js**: Xử lý API liên quan đến album (lấy danh sách và chi tiết album).
    - \* **SongService.js**: Xử lý API bài hát (lấy danh sách và chi tiết bài hát).
  - **App.jsx**: Component gốc của ứng dụng.
  - **index.css**: Các style CSS toàn cục.
  - **main.jsx**: Điểm vào chính của ứng dụng React.
- **.env**: Chứa các biến môi trường.
  - **.gitignore**: Chỉ định các file và thư mục Git nên bỏ qua.
  - **eslint.config.js**: Cấu hình ESLint cho việc linting mã JavaScript.
  - **index.html**: File HTML gốc của ứng dụng.
  - **package-lock.json**: Ghi lại phiên bản chính xác của các dependencies.
  - **package.json**: Chứa thông tin về dự án và các dependencies.
  - **README.md**: File Markdown chứa thông tin giới thiệu về dự án.
  - **vite.config.js**: Cấu hình Vite build tool.

### 3.3.2 Cấu trúc mã nguồn Admin Frontend

- **/src**: Thư mục chứa mã nguồn chính của ứng dụng admin frontend.
  - **/assets**: Lưu trữ tài nguyên tĩnh.
  - **/components**: Chứa các thành phần giao diện người dùng tái sử dụng.
    - \* **/Admin**: Các component dành riêng cho trang quản trị.
    - \* **/Shared**: Các component dùng chung trong ứng dụng.
  - **/context**: Quản lý trạng thái ứng dụng.
    - \* **AuthContext.jsx**: Quản lý trạng thái xác thực và thông tin người dùng.
  - **/Layout**: Chứa các layout (bố cục) của trang.
    - \* **AdminLayout.jsx**: Layout cho trang quản trị.

- /pages: Chứa các view (trang) của ứng dụng.
    - \* /Albums: Các trang liên quan đến album.
    - \* /Artists: Các trang liên quan đến nghệ sĩ.
    - \* /Songs: Các trang liên quan đến bài hát.
    - \* /Users: Các trang liên quan đến người dùng.
    - \* AuthHandler.jsx: Xử lý logic xác thực.
    - \* Dashboard.jsx: Trang dashboard (bảng điều khiển).
    - \* Login.jsx: Trang đăng nhập.
  - /services: Các module giao tiếp với backend API.
    - \* AlbumService.js: Xử lý API liên quan đến album.
    - \* api.js: Cấu hình cơ bản cho việc gọi API.
    - \* ArtistService.js: Xử lý API liên quan đến nghệ sĩ.
    - \* AuthService.js: Xử lý các API liên quan đến xác thực.
    - \* SongService.js: Xử lý API liên quan đến bài hát.
    - \* UserService.js: Xử lý API liên quan đến người dùng.
  - App.jsx: Component gốc của ứng dụng.
  - index.css: Các style CSS toàn cục.
  - main.jsx: Điểm vào chính của ứng dụng React.
- .env: Chứa các biến môi trường.
  - .gitignore: Chỉ định các file và thư mục Git nên bỏ qua.
  - eslint.config.js: Cấu hình ESLint cho việc linting mã JavaScript.
  - index.html: File HTML gốc của ứng dụng.
  - package-lock.json: Ghi lại phiên bản chính xác của các dependencies.
  - package.json: Chứa thông tin về dự án và các dependencies.
  - README.md: File Markdown chứa thông tin giới thiệu về dự án.
  - tailwind.config.js: Cấu hình Tailwind CSS.
  - vite.config.js: Cấu hình Vite build tool.

### 3.3.3 Cấu trúc mã nguồn Backend (Django)

- /accounts: Quản lý xác thực người dùng
  - admin.py: Cấu hình giao diện quản trị cho các models trong accounts.
  - apps.py: Cấu hình ứng dụng accounts.
  - models.py: Định nghĩa các models liên quan đến người dùng (ví dụ: User, Profile).
  - permissions.py: Định nghĩa các quyền tùy chỉnh cho ứng dụng accounts.
  - serializers.py: Chuyển đổi dữ liệu giữa các models accounts và các định dạng dữ liệu khác (ví dụ: JSON).
  - tests.py: Viết các unit tests cho ứng dụng accounts.

- `urls.py`: Định nghĩa các URL patterns cho ứng dụng accounts.
- `views.py`: Xử lý logic nghiệp vụ liên quan đến xác thực người dùng (ví dụ: đăng nhập, đăng ký).
- `/media`: Quản lý file media (Django media root)
  - Thư mục này chứa các file media được người dùng tải lên, được Django quản lý theo cấu hình trong `settings.py`. Các thư mục con có thể bao gồm:
    - \* `/albums`: Lưu trữ ảnh bìa album.
    - \* `/artists`: Lưu trữ ảnh đại diện nghệ sĩ.
    - \* `/songs`: Lưu trữ các file âm thanh.
      - `/audio`: Chứa các file audio thực tế.
      - `/thumbnails`: Chứa ảnh thu nhỏ của bài hát (nếu có).
- `/music`: Core ứng dụng nghe nhạc
  - `admin.py`: Cấu hình giao diện quản trị cho các models trong music.
  - `apps.py`: Cấu hình ứng dụng music.
  - `models.py`: Định nghĩa các models cốt lõi của ứng dụng nghe nhạc (ví dụ: Song, Album, Artist, Playlist).
  - `serializers.py`: Chuyển đổi dữ liệu giữa các models music và các định dạng dữ liệu khác.
  - `tests.py`: Viết các unit tests cho ứng dụng music.
  - `urls.py`: Định nghĩa các URL patterns cho ứng dụng music (các API liên quan đến phát nhạc, playlist).
  - `views.py`: Xử lý logic nghiệp vụ và định nghĩa các API endpoints cho ứng dụng nghe nhạc.
- `/spotify_clone`: Cấu hình project chính
  - `__init__.py`: Đánh dấu thư mục là một package Python.
  - `asgi.py`: Cấu hình ASGI cho phép Django chạy trên các web server asynchronous.
  - `settings.py`: Chứa tất cả các cấu hình của project Django (database, middleware, apps, v.v.).
  - `urls.py`: Định nghĩa các URL patterns toàn cục cho project, bao gồm cả việc include urls từ các ứng dụng khác.
  - `wsgi.py`: Cấu hình WSGI cho phép Django chạy trên các web server truyền thống.
- `/venv`: Môi trường ảo Python
  - Thư mục này chứa một môi trường Python ảo độc lập, nơi các dependencies của project được cài đặt, tránh xung đột với các cài đặt Python toàn hệ thống.
- `manage.py`: Một script dòng lệnh để tương tác với project Django (ví dụ: chạy server phát triển, tạo migrations, chạy tests).
- `requirements.txt`: Liệt kê các Python packages và phiên bản cần thiết cho project.



#### 3.3.4 Các chức năng đã triển khai

- Đăng ký / đăng nhập người dùng với JWT
- Xem danh sách bài hát, album, nghệ sĩ
- Phát nhạc trực tuyến từ backend
- Xem và phát video âm nhạc
- Tạo playlist cá nhân
- Giao diện và chức năng quản trị nội dung
- (Tùy chọn) Giao diện trò chuyện (đang phát triển)



## 4 Cách Thức Cài Đặt Và Chạy Ứng Dụng

### 4.1 Môi trường chạy ứng dụng

Ứng dụng web nghe nhạc trực tuyến được phát triển với các thành phần chính sau:

- **Frontend:** React.js (phiên bản 18.x trở lên)
- **Backend:** Django (phiên bản 4.x) với Django REST framework
- **Cơ sở dữ liệu:** PostgreSQL (khuyến nghị) hoặc SQLite (cho mục đích phát triển)
- **Node.js:** Phiên bản 16.x trở lên (yêu cầu cho frontend)
- **Python:** Phiên bản 3.8 trở lên (yêu cầu cho backend)

### 4.2 Hướng dẫn cài đặt

#### 4.2.1 Clone source code từ GitHub

```
https://github.com/QgQ220903/spotify-clone.git  
cd [spotify-clone]
```

#### 4.2.2 Cài đặt Backend

1. Vào thư mục backend:

```
cd backend
```

2. Tạo và kích hoạt môi trường ảo Python:

```
python -m venv venv  
source venv/bin/activate # Trên Linux/MacOS  
venv\Scripts\activate    # Trên Windows
```

3. Cài đặt các dependencies:

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

4. Thiết lập cơ sở dữ liệu:

```
python manage.py migrate
```

5. Tạo super user (tùy chọn):

```
python manage.py createsuperuser
```



#### 4.2.3 Cài đặt Frontend

1. Mở terminal mới và vào thư mục frontend:

```
cd ../frontend
```

2. Cài đặt các dependencies:

```
npm install
```

#### 4.2.4 Cài đặt Admin Frontend

1. Mở terminal mới và vào thư mục admin-frontent:

```
cd ../admin-frontent
```

2. Cài đặt các dependencies:

```
npm install
```

### 4.3 Hướng dẫn chạy ứng dụng

#### 4.3.1 Khởi chạy Backend

```
cd backend  
python manage.py runserver
```

Backend sẽ chạy tại <http://localhost:8000>

#### 4.3.2 Khởi chạy Frontend

```
cd frontend  
npm run dev
```

Frontend sẽ tự động mở tại <http://localhost:5173>

#### 4.3.3 Khởi chạy Admin Frontend

```
cd admin-frontent  
npm run dev
```

Frontend sẽ tự động mở tại <http://localhost:5174>



#### 4.3.4 Truy cập ứng dụng

- Mở trình duyệt và truy cập `http://localhost:5173`
- Để truy cập admin Django (nếu cần): `http://localhost:8000/admin`

**Lưu ý:** Đảm bảo cả backend và frontend đều đang chạy đồng thời để ứng dụng hoạt động đầy đủ chức năng.



## 5 Phân công nhiệm vụ nhóm

### 5.1 Vai trò và nhiệm vụ các thành viên

- **Thành viên 1: Quách Gia Quy**

- **Nhiệm vụ chính:**

- \* Phân công công việc và quản lý tiến độ dự án
    - \* Thiết kế và phát triển hệ thống backend với Django
    - \* Xây dựng RESTful API cho ứng dụng
    - \* Thiết kế cơ sở dữ liệu và models
    - \* Phát triển giao diện người dùng trang admin
    - \* Xử lý các tương tác CRUD cho ca sĩ, nhạc, user

- **Thành viên 2: Nguyễn Quang Hà**

- **Nhiệm vụ chính:**

- \* Thiết kế cơ sở dữ liệu và models albums, songs
    - \* Xử lý logic nghiệp vụ phía server
    - \* Phát triển giao diện người dùng nghe nhạc, album
    - \* Xử lý các tương tác người dùng nghe nhạc album, tải video
    - \* Kết nối với backend API

- **Thành viên 3: Nguyễn Công Đức**

- **Nhiệm vụ chính:**

- \* Phát triển giao diện người dùng tạo playlist nghe nhạc
    - \* Xử lý các tương tác người dùng tạo playlist nghe nhạc
    - \* Kết nối với backend API
    - \* Phát triển giao diện người dùng trang admin
    - \* Xử lý các tương tác CRUD cho album, playlist

- **Thành viên 4: Lê Thiên Phúc**

- **Nhiệm vụ chính:**

- \* Phát triển giao diện người dùng nghe nhạc, xem video, tải video
    - \* Xử lý các tương tác người dùng nghe nhạc, xem video
    - \* Kết nối với backend API