

TRƯỜNG ĐẠI HỌC SÀI GÒN  
KHOA CÔNG NGHỆ THÔNG TIN



## PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

---

### Xây dựng WebApp sử dụng Django Spotify Clone

**GVHD:**Từ Lãng Phiêu

**SV:**

Koong Chấn Phong - koongchanphong0712@gmail.com

Lê Gia Cường - legiacuong789@gmail.com

Phạm Minh Trung - stsupermarik@gmail.com

Võ Đình Văn - Dinhvanvo510@gmail.com

Hoàng Sỹ Khiêm - dkyytbytb5@gmail.com

TP. HỒ CHÍ MINH, THÁNG 05/2025

# Mục lục

<b>1 Phần Mở Đầu</b>	<b>3</b>
1.1 Bối cảnh và động lực thực hiện đề tài . . . . .	3
1.2 Mục tiêu của đề tài . . . . .	3
1.3 Đối tượng và phạm vi của ứng dụng . . . . .	4
1.3.1 Đối tượng người dùng mục tiêu . . . . .	4
1.3.2 Phạm vi chức năng chính . . . . .	4
1.4 Phương pháp thực hiện dự kiến . . . . .	5
<b>2 Tổng Quan Đề Tài và Công Nghệ Sử Dụng</b>	<b>6</b>
2.1 Lý do chọn đề tài và ý nghĩa . . . . .	6
2.2 Đối tượng người dùng ứng dụng hướng tới . . . . .	7
2.2.1 Người dùng cá nhân (End-Users) . . . . .	7
2.2.2 Người quản trị hệ thống (Administrators) . . . . .	7
2.3 Các công nghệ, thư viện và công cụ nền tảng được sử dụng . . . . .	8
2.3.1 Phát triển Backend (Server-side Development) . . . . .	8
2.3.2 Phát triển Frontend (Client-side Development) . . . . .	9
2.3.3 Cơ sở dữ liệu (Database System) . . . . .	10
2.3.4 Lưu trữ và Phân phối File Media (Âm thanh, Hình ảnh) . .	11
2.3.5 Công cụ Phát triển, Quản lý Mã nguồn và Kiểm thử . . .	11
2.4 Cấu trúc mã nguồn . . . . .	12
2.5 Mô hình ứng dụng . . . . .	13
2.6 Các tính năng được xây dựng . . . . .	14
2.6.1 Use Case Diagram . . . . .	15
2.7 Xây dựng database . . . . .	15
<b>3 Chức năng chung của User</b>	<b>16</b>
3.1 Đăng ký và đăng nhập . . . . .	16
3.1.1 Đăng ký . . . . .	16
3.1.2 Đăng nhập . . . . .	18
3.1.2.1 Xử lý phía server . . . . .	18
3.2 Quản lý hồ sơ cá nhân . . . . .	19
3.3 Tìm kiếm Users, Playlists, Songs . . . . .	20
3.3.0.1 Xử lý phía server . . . . .	21
<b>4 Nhắn tin</b>	<b>22</b>
4.1 Các cuộc hội thoại (Conversation) . . . . .	22
4.1.0.1 Code xử lý hiển thị tin nhắn . . . . .	22

4.1.0.2	Giải thích code . . . . .	22
4.2	Chi tiết cuộc trò chuyện (Conservation detail) . . . . .	23
4.2.0.1	Giao diện gồm 3 phần như hình, mỗi phần lần lượt là: . . . . .	23
4.2.0.2	Đoạn code thiết lập một kết nối WebSocket để giao tiếp hai chiều thời gian thực, cho phép người dùng gửi và nhận tin nhắn một cách nhanh chóng và hiệu quả: . . . . .	24
<b>5</b>	<b>Songs và Albums</b>	<b>27</b>
5.1	Xem và tương tác với albums của người dùng khác (play, follow) . . . . .	27
5.2	Tạo Songs . . . . .	28
5.3	Playlists . . . . .	31
<b>6</b>	<b>Admin</b>	<b>33</b>
6.1	Song Managements . . . . .	33
6.2	User Managements . . . . .	34
6.3	Genres Managements . . . . .	35
6.4	Thống Kê . . . . .	36
<b>7</b>	<b>Kết Quả Đạt Được và Đánh Giá</b>	<b>37</b>
7.1	Kết quả đạt được . . . . .	37
7.2	Đánh giá sản phẩm . . . . .	38
7.2.1	Ưu điểm của hệ thống . . . . .	39
7.2.2	Nhược điểm và hạn chế . . . . .	39
7.3	Bài học kinh nghiệm . . . . .	40
<b>Tài liệu tham khảo</b>		<b>41</b>

# 1 Phần Mở Đầu

Trong kỷ nguyên số hóa hiện nay, nhu cầu giải trí trực tuyến, đặc biệt là thưởng thức âm nhạc, ngày càng trở nên phổ biến và thiết yếu trong đời sống hàng ngày. Các nền tảng nghe nhạc trực tuyến hàng đầu như Spotify đã minh chứng sức hút mạnh mẽ thông qua việc cung cấp trải nghiệm cá nhân hóa, giao diện trực quan, dễ sử dụng và khả năng truy cập đa nền tảng. Việc xây dựng một ứng dụng mô phỏng các tính năng cốt lõi của Spotify (thường được gọi là "Spotify Clone") không chỉ đơn thuần là một bài tập kỹ thuật, mà còn là cơ hội để người học áp dụng và làm sâu sắc thêm kiến thức về các công nghệ hiện đại. Quá trình này bao gồm việc xử lý các vấn đề như phát nhạc trực tuyến (streaming), quản lý dữ liệu người dùng, và có thể mở rộng đến các tính năng đề xuất thông minh. Do đó, việc phát triển một hệ thống Spotify Clone vừa mang đậm tính học thuật, giúp nâng cao kỹ năng lập trình, tư duy thiết kế hệ thống, vừa có giá trị thực tiễn cao, phản ánh khả năng tích hợp và ứng dụng công nghệ trong môi trường phát triển phần mềm hiện đại.

## 1.1 Bối cảnh và động lực thực hiện đề tài

Sự phát triển không ngừng của công nghệ Internet và các thiết bị thông minh đã thúc đẩy mạnh mẽ thị trường âm nhạc trực tuyến. Người dùng ngày càng có xu hướng chuyển dịch từ các hình thức nghe nhạc truyền thống sang các nền tảng streaming nhờ sự tiện lợi, kho nội dung đa dạng và khả năng tùy chỉnh cao. Điều này tạo ra một lĩnh vực ứng dụng phong phú và đầy thách thức cho các nhà phát triển phần mềm. Đối với sinh viên ngành Công nghệ Thông tin, việc tham gia vào các dự án thực tế, mô phỏng những sản phẩm thành công là một phương pháp học tập hiệu quả. Đề tài xây dựng ứng dụng Spotify Clone được lựa chọn không chỉ vì tính phổ biến của sản phẩm gốc mà còn vì nó bao hàm nhiều khía cạnh công nghệ quan trọng, từ frontend, backend, cơ sở dữ liệu đến các dịch vụ phụ trợ. Đây chính là động lực để nhóm thực hiện đề tài này, nhằm mục tiêu củng cố kiến thức và rèn luyện kỹ năng chuyên môn.

## 1.2 Mục tiêu của đề tài

Đề tài này hướng tới các mục tiêu chính sau:

- Xây dựng ứng dụng web nghe nhạc trực tuyến:** Phát triển một sản phẩm phần mềm có các chức năng cơ bản tương tự như Spotify, cho phép người dùng tìm kiếm, nghe nhạc và quản lý thư viện cá nhân.

- **Áp dụng các công nghệ mục tiêu:** Triển khai ứng dụng sử dụng Django Framework cho phần backend, thư viện ReactJS cho phần frontend, và hệ quản trị cơ sở dữ liệu MongoDB để lưu trữ dữ liệu.
- **Nâng cao kiến thức và kỹ năng:** Củng cố kiến thức về quy trình phát triển phần mềm full-stack, rèn luyện kỹ năng làm việc với API, quản lý trạng thái ứng dụng, và tương tác với cơ sở dữ liệu NoSQL.
- **Hoàn thành sản phẩm và báo cáo:** Giao nộp một sản phẩm phần mềm có thể hoạt động được (ở mức độ nhất định) và một tài liệu báo cáo chi tiết, đáp ứng yêu cầu của môn học.

## 1.3 Đối tượng và phạm vi của ứng dụng

### 1.3.1 Đối tượng người dùng mục tiêu

Ứng dụng được phát triển chủ yếu hướng đến:

- Người dùng có nhu cầu nghe nhạc trực tuyến, khám phá bài hát mới và quản lý các playlist cá nhân.
- Sinh viên và những người quan tâm đến việc tìm hiểu cách một ứng dụng nghe nhạc được xây dựng.

### 1.3.2 Phạm vi chức năng chính

Do giới hạn về thời gian và nguồn lực, ứng dụng sẽ tập trung vào các chức năng cốt lõi sau:

- Đăng ký và đăng nhập tài khoản người dùng.
- Tìm kiếm bài hát, album, nghệ sĩ.
- Phát nhạc trực tuyến với các điều khiển cơ bản.
- Tạo, xem và quản lý các playlist cá nhân.
- Xem thông tin chi tiết của bài hát và album.
- Chức năng "Thích" (Like) bài hát/album.
- (Tùy chọn nếu có) Giao diện quản trị cơ bản để quản lý nội dung.

Các tính năng nâng cao như hệ thống gợi ý phức tạp, nghe nhạc offline, podcast, video, và các tính năng xã hội sâu sẽ không thuộc phạm vi của đề tài này.

## 1.4 Phương pháp thực hiện dự kiến

Nhóm dự kiến thực hiện đề tài theo các bước chính sau:

- Nghiên cứu và Phân tích Yêu cầu:** Tìm hiểu kỹ các chức năng của Spotify và xác định các yêu cầu cần thiết cho phiên bản clone.
- Thiết kế Hệ thống:** Bao gồm thiết kế kiến trúc ứng dụng, thiết kế cơ sở dữ liệu (MongoDB collections), và thiết kế API backend.
- Triển khai Backend:** Xây dựng các API sử dụng Django và Django REST Framework.
- Triển khai Frontend:** Phát triển giao diện người dùng bằng ReactJS, tương tác với backend thông qua API.
- Tích hợp và Kiểm thử:** Kết nối các thành phần, kiểm thử các chức năng để đảm bảo hoạt động ổn định.
- Viết Báo cáo và Hoàn thiện Sản phẩm:** Tài liệu hóa quá trình thực hiện và hoàn thiện các chi tiết cuối cùng của ứng dụng.

Quá trình phát triển sẽ áp dụng các nguyên tắc của phát triển phần mềm linh hoạt, với việc chia nhỏ công việc và kiểm tra tiến độ thường xuyên.

## 2 Tổng Quan Đề Tài và Công Nghệ Sử Dụng

Chương này nhằm mục đích cung cấp cái nhìn tổng quan về đề tài "Xây dựng ứng dụng Spotify Clone", bao gồm lý do và động lực thực hiện, cũng như phân tích các đối tượng người dùng mà ứng dụng hướng đến. Phần quan trọng nhất của chương sẽ tập trung giới thiệu và làm rõ vai trò của các công nghệ, thư viện và công cụ nền tảng đã được lựa chọn để phát triển sản phẩm. Điều này không chỉ đặt nền móng kỹ thuật cho các quyết định thiết kế và triển khai được trình bày ở các chương sau, mà còn thể hiện sự tìm hiểu và cân nhắc trong việc lựa chọn technology stack phù hợp cho dự án.

### 2.1 Lý do chọn đề tài và ý nghĩa

Âm nhạc từ lâu đã trở thành một phần không thể thiếu trong đời sống tinh thần của con người, đóng vai trò quan trọng trong việc giải trí, biểu đạt cảm xúc, giảm căng thẳng và nâng cao chất lượng cuộc sống. Sự bùng nổ của công nghệ thông tin và Internet đã làm thay đổi sâu sắc cách chúng ta tiếp cận và tiêu thụ âm nhạc, với xu hướng nghe nhạc trực tuyến (streaming) ngày càng trở nên phổ biến và chiếm ưu thế. Các nền tảng hàng đầu như Spotify, Apple Music, hay Zing MP3 tại Việt Nam, đã minh chứng cho tiềm năng và sức hút của mô hình này.

Việc lựa chọn đề tài xây dựng một ứng dụng nghe nhạc trực tuyến tương tự Spotify (Spotify Clone) xuất phát từ nhiều lý do và mang lại những ý nghĩa thiết thực sau:

- Đáp ứng nhu cầu thực tế và xu hướng thị trường:** Nhu cầu giải trí bằng âm nhạc trực tuyến là rất lớn và không ngừng tăng. Việc xây dựng một ứng dụng có các chức năng cốt lõi như phát nhạc, quản lý playlist, tìm kiếm bài hát, nghệ sĩ, album sẽ đáp ứng được nhu cầu cơ bản của người dùng hiện đại.
- Cơ hội học hỏi và áp dụng công nghệ chuyên sâu:** Đây là một đề tài phức hợp, đòi hỏi sự kết hợp của nhiều công nghệ và kiến thức từ phát triển full-stack. Sinh viên có cơ hội thực hành với các công nghệ backend (ví dụ: Django), frontend (ví dụ: ReactJS), cơ sở dữ liệu (ví dụ: MongoDB), xử lý API, quản lý trạng thái ứng dụng, và có thể cả các vấn đề liên quan đến streaming media và lưu trữ file dung lượng lớn.
- Hiểu biết về kiến trúc hệ thống lớn:** Mô phỏng một ứng dụng có quy mô như Spotify giúp sinh viên hình dung và tiếp cận với cách thiết kế, tổ chức và vận hành của các hệ thống phần mềm phức tạp trong thực tế, từ đó nâng cao tư duy hệ thống.

- **Nâng cao kỹ năng phát triển phần mềm toàn diện:** Quá trình thực hiện đề tài bao gồm đầy đủ các bước của một chu trình phát triển phần mềm: từ phân tích yêu cầu, thiết kế, lập trình, kiểm thử, đến triển khai (ở mức độ mô phỏng) và viết tài liệu. Điều này giúp rèn luyện một cách toàn diện các kỹ năng cần thiết.
- **Giá trị cho hồ sơ năng lực và tiềm năng phát triển:** Một sản phẩm "clone" được xây dựng tốt không chỉ là một bài tập lớn hoàn thành mà còn là một dự án giá trị trong portfolio cá nhân, thể hiện năng lực kỹ thuật và sự chủ động học hỏi. Hơn nữa, nó có thể là nền tảng để phát triển thêm các ý tưởng sáng tạo và tính năng độc đáo trong tương lai.

Tóm lại, đề tài này không chỉ là một thử thách kỹ thuật mà còn là một cơ hội quý báu để củng cố kiến thức, rèn luyện kỹ năng và tạo ra một sản phẩm có ý nghĩa.

## 2.2 Đối tượng người dùng ứng dụng hướng tới

Ứng dụng Spotify Clone được phát triển với mục tiêu phục vụ các nhóm đối tượng người dùng sau:

### 2.2.1 Người dùng cá nhân (End-Users)

Đây là nhóm đối tượng trọng tâm, bao gồm:

- **Người yêu âm nhạc nói chung:** Những người có thói quen nghe nhạc thường xuyên cho các mục đích giải trí, thư giãn, học tập, hoặc làm việc. Họ mong muốn một ứng dụng dễ sử dụng, có kho nhạc (mô phỏng) đa dạng, cho phép tạo và quản lý playlist cá nhân, khám phá nhạc mới và theo dõi nghệ sĩ.
- **Người dùng trẻ, năng động:** Nhóm này thường ưu tiên các giao diện hiện đại, trực quan, tốc độ phản hồi nhanh và các tính năng tương tác cơ bản như "Thích" bài hát.

### 2.2.2 Người quản trị hệ thống (Administrators)

Mặc dù không phải là người dùng cuối trực tiếp thưởng thức âm nhạc, nhưng vai trò quản trị viên rất quan trọng để duy trì và quản lý nội dung của ứng dụng (trong phạm vi mô phỏng của đề tài):

- **Quản lý nội dung âm nhạc:** Có khả năng thêm, sửa, xóa thông tin bài hát, album, nghệ sĩ.

- **Quản lý người dùng:** Xem danh sách người dùng, có thể thực hiện các tác vụ quản trị cơ bản như khóa tài khoản (nếu cần thiết).

## 2.3 Các công nghệ, thư viện và công cụ nền tảng được sử dụng

Để hiện thực hóa các mục tiêu đề ra, việc lựa chọn một bộ công cụ (technology stack) phù hợp, hiện đại và có tính hỗ trợ cao là vô cùng quan trọng. Nhóm đã quyết định sử dụng các công nghệ, thư viện và công cụ chính sau đây:

### 2.3.1 Phát triển Backend (Server-side Development)

- **Python (Ngôn ngữ lập trình):** Python được lựa chọn làm ngôn ngữ chính cho việc phát triển backend nhờ vào cú pháp sáng sủa, dễ đọc, cùng với một hệ sinh thái thư viện vô cùng phong phú và cộng đồng hỗ trợ đông đảo. Khả năng phát triển nhanh (Rapid Application Development - RAD) và sự phù hợp với các tác vụ xử lý dữ liệu và phát triển web là những ưu điểm nổi bật.
- **Django Framework:** Django là một framework web Python bậc cao (high-level), được thiết kế để thúc đẩy việc phát triển các ứng dụng web an toàn, có khả năng mở rộng một cách nhanh chóng và hiệu quả. Django tuân theo kiến trúc MVT (Model-View-Template) và cung cấp sẵn nhiều tính năng mạnh mẽ:
  - Hệ thống ORM (Object-Relational Mapper) cho phép tương tác với cơ sở dữ liệu bằng cú pháp Python tự nhiên. Tuy nhiên, do dự án sử dụng MongoDB (một CSDL NoSQL), nhóm sẽ cần sử dụng các thư viện kết nối chuyên biệt như PyMongo, hoặc có thể xem xét Djongo nếu muốn có trải nghiệm tương tự ORM.
  - Trang quản trị (Admin Panel) được tự động sinh, giúp tiết kiệm thời gian và công sức trong việc xây dựng các giao diện quản lý dữ liệu cơ bản.
  - Hệ thống định tuyến URL (URL dispatching), middleware, quản lý template và xử lý form một cách hiệu quả và có cấu trúc.
  - Các cơ chế bảo mật tích hợp sẵn giúp phòng chống các lỗ hổng web phổ biến.
- **Django REST Framework (DRF):** Là một bộ công cụ (toolkit) mạnh mẽ và linh hoạt, được xây dựng trên nền tảng Django, chuyên dùng để phát triển các Web API theo chuẩn RESTful. DRF giúp đơn giản hóa và chuẩn hóa quá trình:

- **Serialization và Deserialization:** Chuyển đổi dữ liệu giữa các đối tượng Python (ví dụ: model instances, querysets) và các định dạng dữ liệu phổ biến cho API như JSON một cách dễ dàng.
- **Xây dựng API Endpoints:** Cung cấp các lớp view (APIView, Generic Views, ViewSets) và routers giúp định nghĩa các điểm cuối API với các phương thức HTTP (GET, POST, PUT, DELETE) một cách nhanh chóng và có tổ chức.
- **Xác thực (Authentication) và Phân quyền (Permissions):** Hỗ trợ nhiều cơ chế xác thực (ví dụ: Token Authentication, Session Authentication) và phân quyền để kiểm soát truy cập vào API.

### 2.3.2 Phát triển Frontend (Client-side Development)

- **ReactJS (Thư viện JavaScript):** React là một thư viện JavaScript mã nguồn mở, được phát triển bởi Meta, chuyên dùng để xây dựng giao diện người dùng (UI) động, tương tác cao và có khả năng tái sử dụng cao. Các lý do chính lựa chọn React bao gồm:
  - **Kiến trúc dựa trên Component (Component-Based Architecture):** Giao diện người dùng được chia thành các thành phần (components) độc lập, mỗi component quản lý trạng thái (state) và logic hiển thị riêng. Điều này giúp mã nguồn dễ quản lý, bảo trì, tái sử dụng và phát triển theo nhóm hiệu quả hơn.
  - **Virtual DOM (DOM ảo):** React sử dụng một bản sao của DOM trong bộ nhớ (Virtual DOM). Khi trạng thái của một component thay đổi, React sẽ tính toán sự khác biệt (diffing) và chỉ cập nhật những phần thực sự cần thiết trên DOM thật, giúp tối ưu hóa hiệu năng và mang lại trải nghiệm người dùng mượt mà.
  - **JSX (JavaScript XML):** Là một phần mở rộng cú pháp cho JavaScript, cho phép viết mã giống HTML (hoặc XML) trực tiếp trong các file JavaScript. JSX giúp việc mô tả cấu trúc UI trở nên trực quan, dễ đọc và dễ viết hơn.
- **Quản lý Trạng thái (State Management):**
  - **React Context API:** Đối với các trạng thái toàn cục không quá phức tạp (ví dụ: thông tin người dùng đăng nhập, trạng thái của trình phát nhạc), Context API là một giải pháp tích hợp sẵn trong React, cho phép chia sẻ dữ liệu giữa các component mà không cần truyền props qua nhiều cấp (prop drilling).

- (Tùy chọn) **Thư viện quản lý trạng thái chuyên dụng (ví dụ: Redux, Zustand)**: Nếu ứng dụng phát triển với nhiều trạng thái phức tạp và cần một luồng dữ liệu rõ ràng, dễ gõ lỗi hơn, các thư viện như Redux (cùng Redux Toolkit) hoặc các giải pháp nhẹ hơn như Zustand có thể được cân nhắc.
  - **React Router DOM**: Là thư viện tiêu chuẩn để triển khai việc định tuyến (routing) phía client trong các ứng dụng đơn trang (Single Page Applications - SPA) xây dựng bằng React. Nó cho phép người dùng điều hướng giữa các "trang" hoặc "view" khác nhau của ứng dụng mà không cần tải lại toàn bộ trang web từ server.
  - **Axios**: Một thư viện HTTP client dựa trên Promise, được sử dụng để thực hiện các yêu cầu API (ví dụ: GET, POST, PUT, DELETE) từ ứng dụng React đến backend Django một cách thuận tiện, hỗ trợ nhiều tính năng như interceptors, error handling.
- ### 2.3.3 Cơ sở dữ liệu (Database System)
- **MongoDB**: Là một hệ quản trị cơ sở dữ liệu NoSQL mã nguồn mở, thuộc loại document-oriented database (cơ sở dữ liệu hướng tài liệu). MongoDB lưu trữ dữ liệu dưới dạng các tài liệu BSON (Binary JSON), cung cấp sự linh hoạt cao trong cấu trúc dữ liệu. Các ưu điểm chính khi chọn MongoDB cho dự án này:
    - **Schema linh hoạt (Flexible Schema)**: Cho phép các tài liệu trong cùng một collection có các trường khác nhau, rất hữu ích khi metadata của bài hát hoặc thông tin người dùng có thể có nhiều thuộc tính tùy chọn hoặc thay đổi theo thời gian.
    - **Khả năng mở rộng (Scalability)**: MongoDB được thiết kế để dễ dàng mở rộng theo chiều ngang (horizontal scaling) thông qua sharding, phù hợp cho các ứng dụng có tiềm năng tăng trưởng dữ liệu lớn.
    - **Hiệu năng cho các truy vấn nhất định**: Có thể cho hiệu năng tốt với các truy vấn trên dữ liệu phi cấu trúc hoặc bán cấu trúc, và khi làm việc với các tài liệu lớn hoặc dữ liệu nhúng.
  - **PyMongo**: Là thư viện driver Python chính thức để ứng dụng Django tương tác với MongoDB, cung cấp các API để thực hiện kết nối, truy vấn và các thao tác CRUD.

### 2.3.4 Lưu trữ và Phân phối File Media (Âm thanh, Hình ảnh)

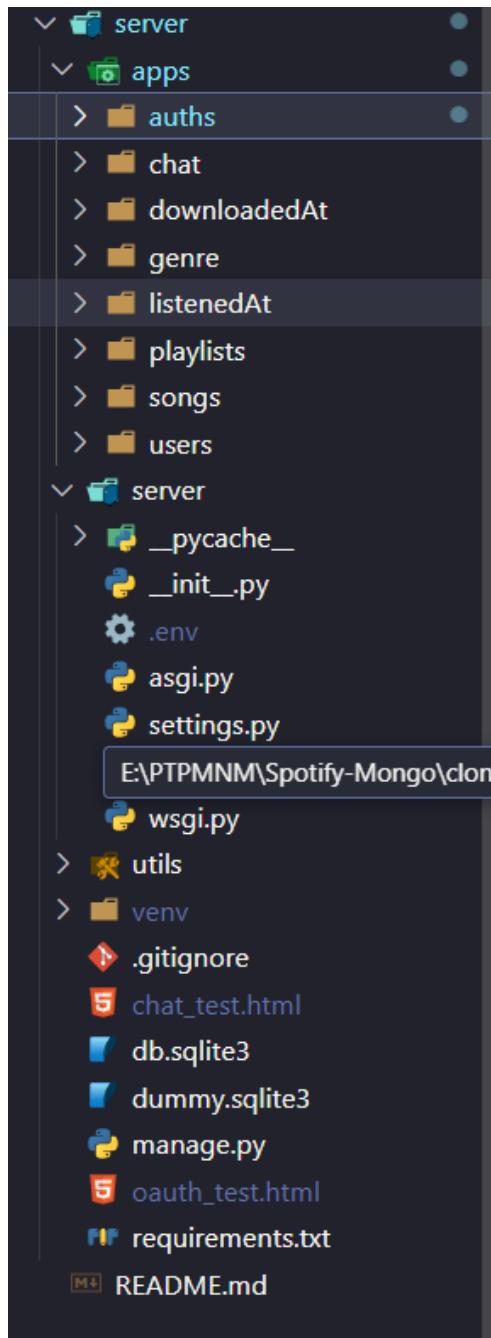
- **Cloudinary (hoặc Amazon S3 - AWS S3):** Các dịch vụ lưu trữ đám mây (cloud storage) sẽ được ưu tiên sử dụng để lưu trữ các file âm thanh và hình ảnh (ảnh bìa album, ảnh đại diện). Lý do:
  - Giảm tải cho máy chủ ứng dụng chính.
  - Đảm bảo tính sẵn sàng cao và tốc độ truy cập nhanh cho file media.
  - Dễ dàng quản lý và mở rộng dung lượng lưu trữ.
  - Cloudinary cung cấp nhiều API tiện ích cho việc xử lý và tối ưu hóa hình ảnh, video trực tiếp trên cloud.
- **Thư viện SDK tương ứng:** Sử dụng các thư viện SDK chính thức do nhà cung cấp dịch vụ cloud cung cấp (ví dụ: Cloudinary Python SDK, Boto3 cho AWS S3) để tích hợp vào backend Django.

### 2.3.5 Công cụ Phát triển, Quản lý Mã nguồn và Kiểm thử

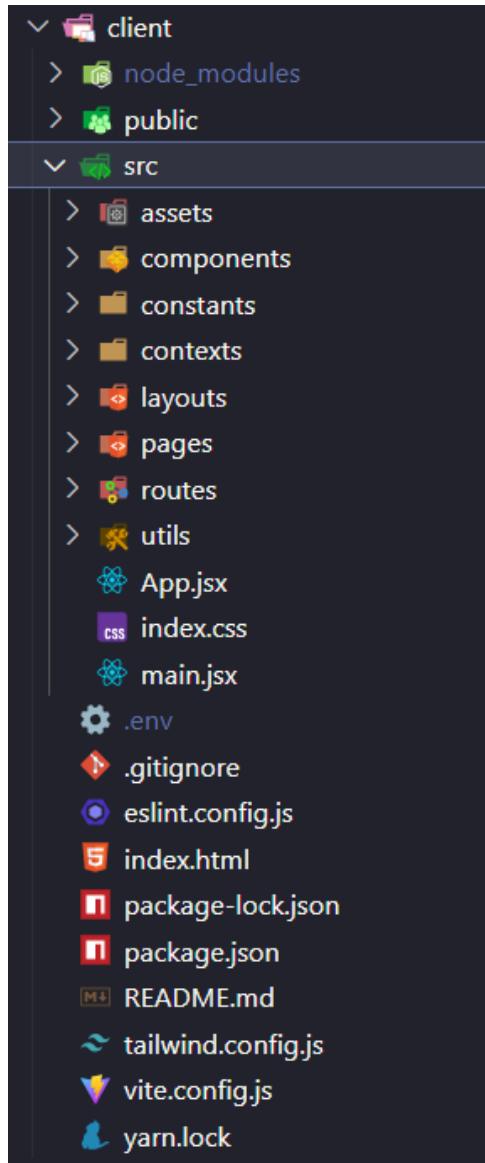
- **Git và GitHub (hoặc GitLab/Bitbucket):** Git là hệ thống quản lý phiên bản phân tán tiêu chuẩn. GitHub (hoặc các nền tảng tương tự) được sử dụng làm nơi lưu trữ mã nguồn từ xa, quản lý dự án và hỗ trợ làm việc nhóm.
- **Môi trường phát triển tích hợp (IDE) / Trình soạn thảo mã nguồn:** Visual Studio Code (VS Code) được ưu tiên lựa chọn nhờ tính linh hoạt, nhẹ nhàng, cộng đồng lớn và nhiều tiện ích mở rộng hỗ trợ phát triển Python, Django, JavaScript, React.
- **Postman (hoặc Insomnia):** Công cụ đồ họa để thiết kế, gửi yêu cầu và kiểm thử các API RESTful một cách hiệu quả.
- **Trình quản lý gói (Package Managers):** Pip cho các gói Python và npm (hoặc yarn) cho các gói JavaScript/React.

Việc lựa chọn và kết hợp các công nghệ này được kỳ vọng sẽ tạo ra một nền tảng vững chắc để xây dựng ứng dụng Spotify Clone một cách hiệu quả, đồng thời cung cấp môi trường tốt để nhóm học hỏi và phát triển kỹ năng.

## 2.4 Cấu trúc mã nguồn



Hình 1: Cấu trúc thư mục phía server



Hình 2: Cấu trúc thư mục phía client

## 2.5 Mô hình ứng dụng

Ứng dụng Spotify Clone được xây dựng theo kiến trúc **client-server**, trong đó:

- **Client (giao diện người dùng)**: được phát triển bằng **React JS**, đảm nhận việc hiển thị giao diện và xử lý các tương tác từ người dùng. Client gửi các yêu cầu HTTP (REST API) đến server để lấy dữ liệu như danh sách bài hát, playlist, thông tin người dùng, v.v.
- **Server (xử lý nghiệp vụ)**: được phát triển bằng **Django**, có nhiệm vụ xử lý các yêu cầu từ client, tương tác với cơ sở dữ liệu, xác thực người dùng, và trả về dữ liệu dưới dạng JSON. Django cũng quản lý logic nghiệp vụ như phát nhạc, tạo/sửa playlist, và quản lý tài khoản người dùng.

- **Cơ sở dữ liệu:** Django sử dụng hệ quản trị cơ sở dữ liệu (ví dụ: PostgreSQL hoặc SQLite) để lưu trữ thông tin người dùng, bài hát, playlist, lịch sử phát nhạc, v.v.

Mô hình hoạt động tổng quan như sau:

1. Người dùng tương tác với giao diện React JS (client).
2. Client gửi yêu cầu HTTP tới server Django thông qua các API.
3. Django xử lý yêu cầu, truy xuất dữ liệu từ cơ sở dữ liệu, sau đó phản hồi về client dưới dạng JSON.
4. Client nhận dữ liệu và hiển thị kết quả cho người dùng.

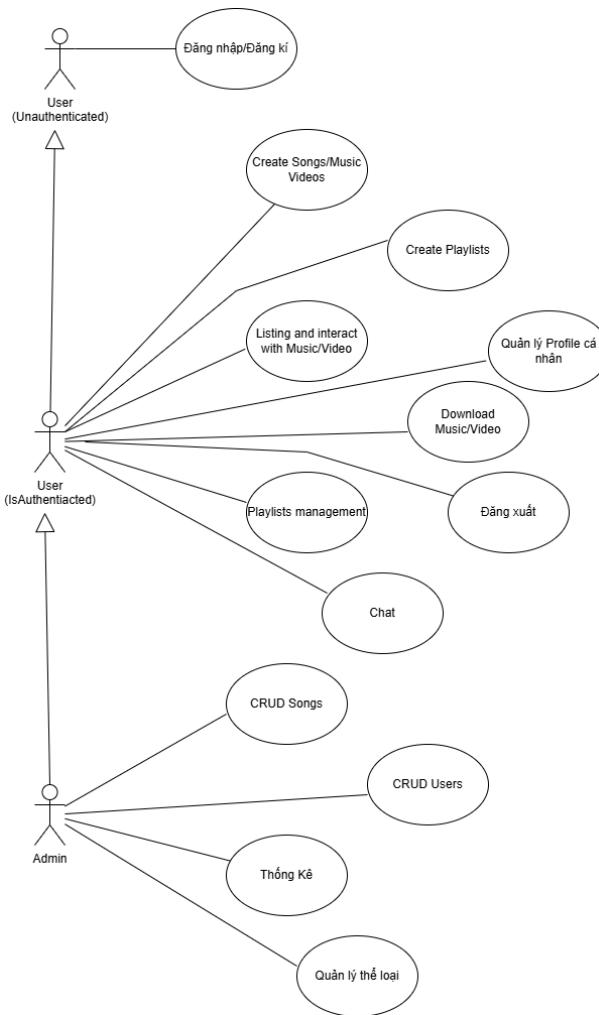
Mô hình kiến trúc này giúp tách biệt rõ ràng giữa phần giao diện và phần xử lý nghiệp vụ, giúp dễ bảo trì, mở rộng và tái sử dụng trong tương lai.

## 2.6 Các tính năng được xây dựng

Dựa trên yêu cầu đề bài, các tính năng chính được xây dựng trong hệ thống Spotify Clone bao gồm:

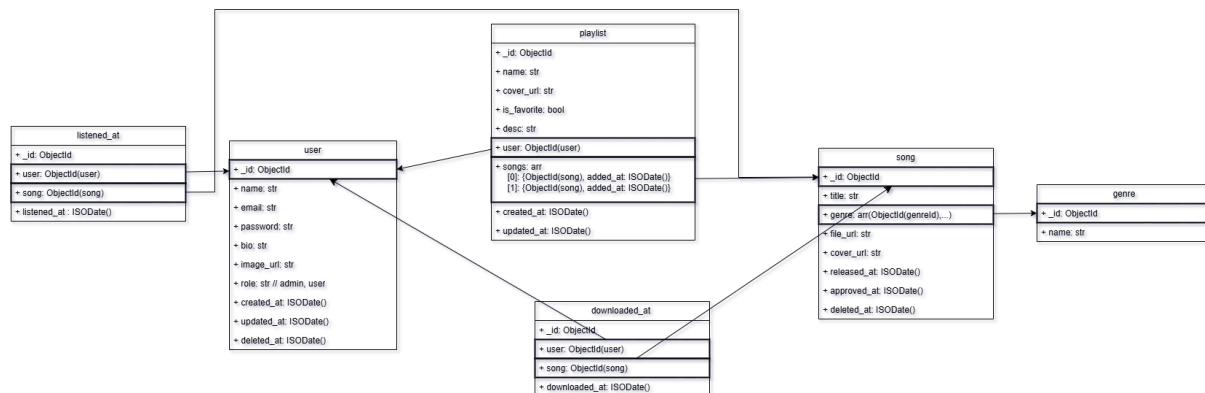
- Chức năng phát nhạc.
- Chức năng phát video âm nhạc.
- Chức năng tải video âm nhạc.
- Chức năng tạo album và thêm bài hát yêu thích cho người dùng.
- Giao diện và chức năng quản trị (Admin): Quản lý Songs, Quản lý Users, Quản lý playlists và Thống Kê.
- Tính năng chat tích hợp trong giao diện web (tùy chọn).

### 2.6.1 Use Case Diagram



Hình 3: Usecase tổng quan hệ thống

### 2.7 Xây dựng database



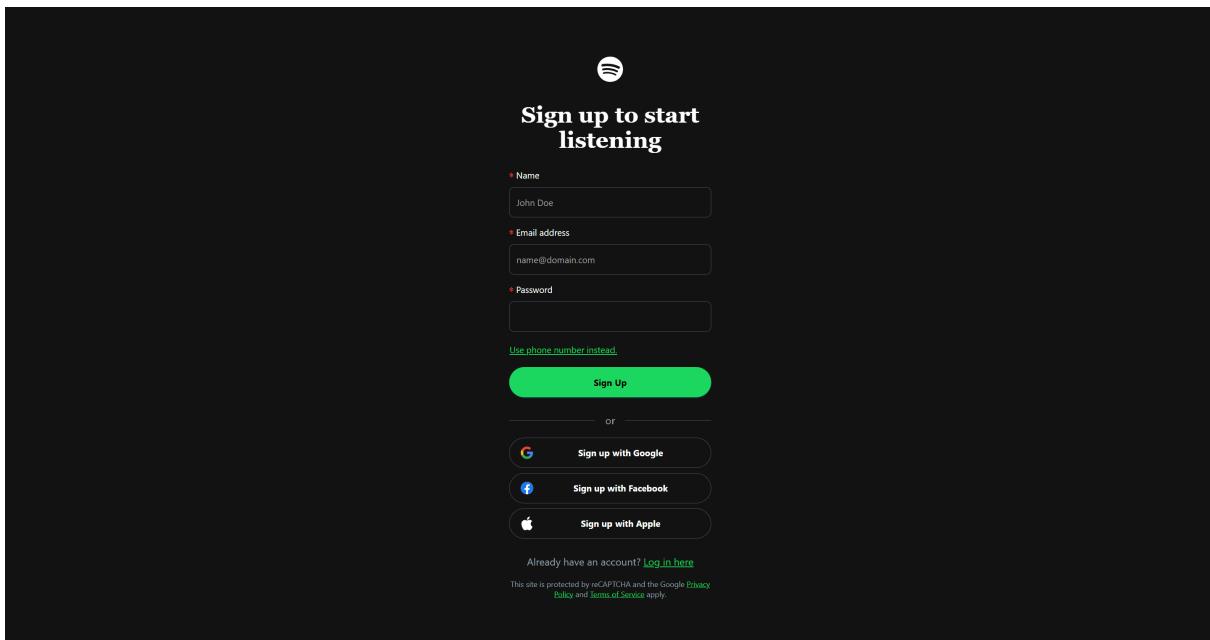
Hình 4: Mô hình class diagram của hệ thống

### 3 Chức năng chung của User

#### 3.1 Đăng ký và đăng nhập

##### 3.1.1 Đăng ký

Để sử dụng ứng dụng, người dùng cần tạo tài khoản cá nhân. Nhóm đã xây dựng một biểu mẫu đăng ký (signup pop-up) với giao diện thân thiện và chức năng xử lý phía máy chủ nhằm hỗ trợ người dùng tạo tài khoản một cách dễ dàng và an toàn. Các bước triển khai chức năng đăng ký như sau:



Hình 5: Giao diện trang đăng ký

Khi người dùng gửi thông tin qua biểu mẫu, hệ thống sẽ xử lý như sau:

- **Thu thập dữ liệu:** Biểu mẫu bao gồm các trường thông tin cơ bản như: tên người dùng (username), email và mật khẩu.
- **Giao diện người dùng:** Mỗi trường được thiết kế kèm thông báo lỗi tùy chỉnh để nâng cao trải nghiệm sử dụng và đảm bảo tính nhất quán về mặt giao diện.
- **Xử lý phía máy chủ:** Một view trong Django sẽ tiếp nhận dữ liệu từ biểu mẫu. Nếu thông tin hợp lệ (ví dụ: mật khẩu trùng khớp, email chưa được đăng ký...), hệ thống sẽ tạo một tài khoản mới và lưu vào cơ sở dữ liệu.
- **Tự động chuyển hướng:** Sau khi đăng ký thành công, người dùng sẽ được chuyển hướng đến trang chính của ứng dụng.

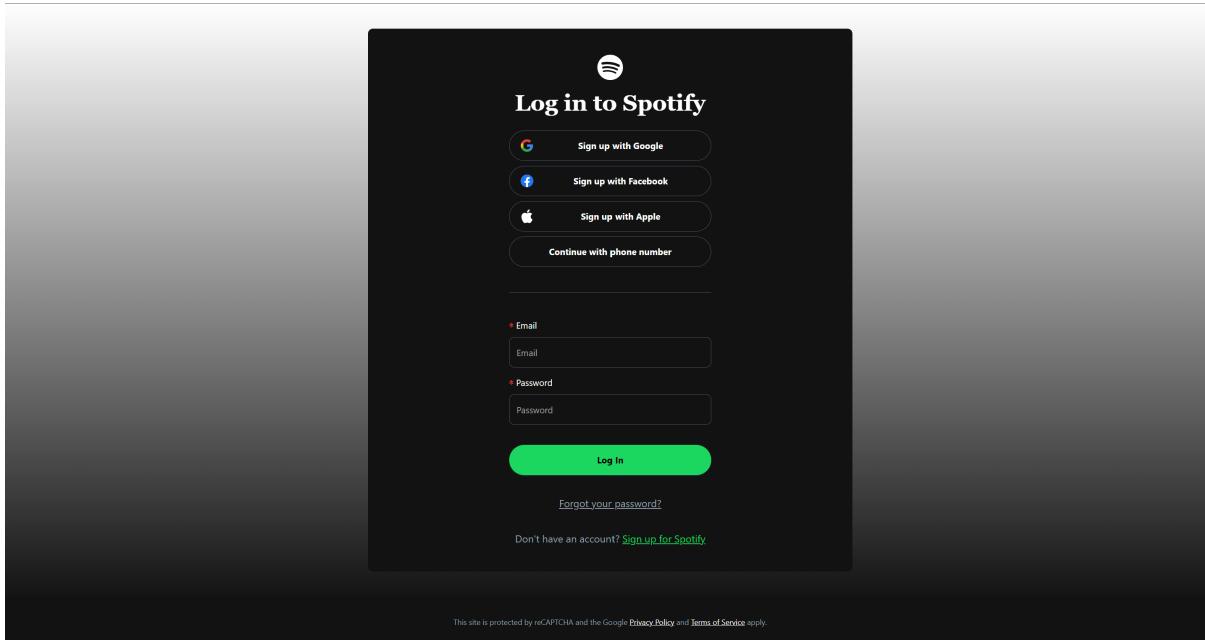


- **Tùy chỉnh biểu mẫu:** Biểu mẫu đăng ký được xây dựng dựa trên UserCreationForm của Django, sau đó được mở rộng để thêm các trường cần thiết và tùy chỉnh giao diện hiển thị.

Việc thiết kế và triển khai chức năng đăng ký một cách trực quan và bảo mật giúp người dùng dễ dàng tiếp cận và bắt đầu sử dụng ứng dụng ngay từ bước đầu tiên.

### 3.1.2 Đăng nhập

Sau khi người dùng đã có tài khoản, họ cần đăng nhập để truy cập vào ứng dụng Spotify. Chúng em đã tạo một trang chứa một form đăng nhập (login form) để người dùng nhập email và mật khẩu. Sau đó sẽ gửi yêu cầu đăng nhập (login request) từ form đến server.



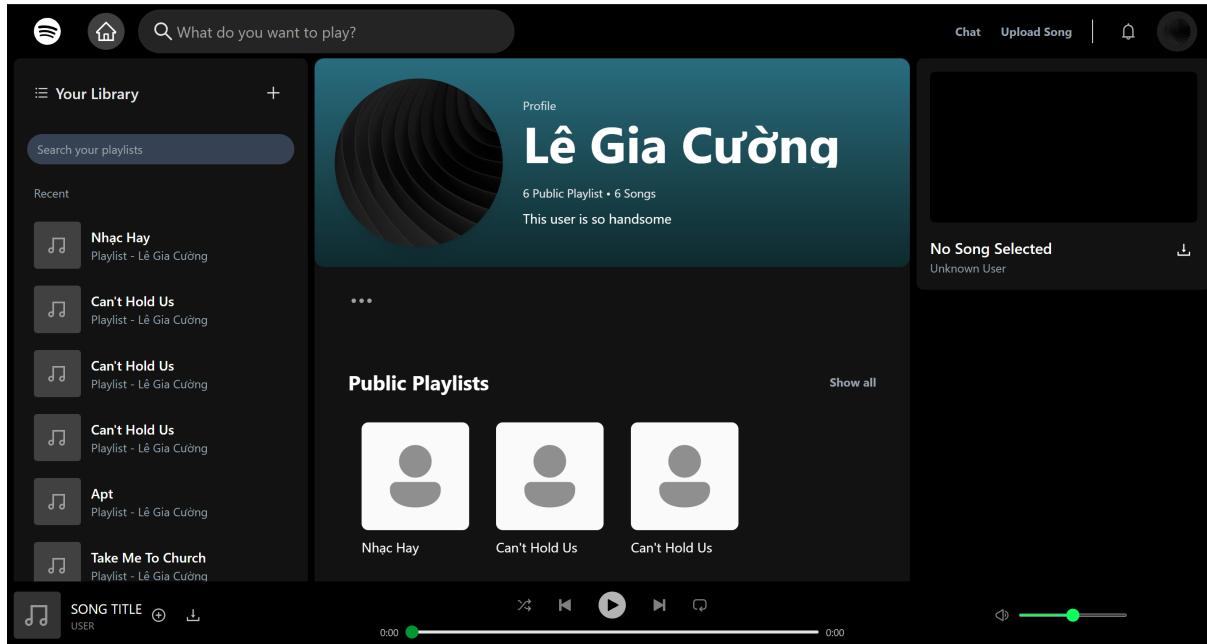
Hình 6: Giao diện đăng nhập

#### 3.1.2.1 Xử lý phía server

- Trên server, tạo một view để xử lý yêu cầu login.
- Trong view này, kiểm tra thông tin đăng nhập với thông tin trong cơ sở dữ liệu. Nếu thông tin chính xác, đánh dấu người dùng là đã đăng nhập và chuyển hướng họ đến trang chính của mạng xã hội. Nếu không, thông báo lỗi và yêu cầu nhập lại.

### 3.2 Quản lý hồ sơ cá nhân

Ví dụ: Người dùng có thể cập nhật thông tin cá nhân như ảnh đại diện, tên hiển thị, mật khẩu, v.v.



Hình 7: Giao diện profile cá nhân của user

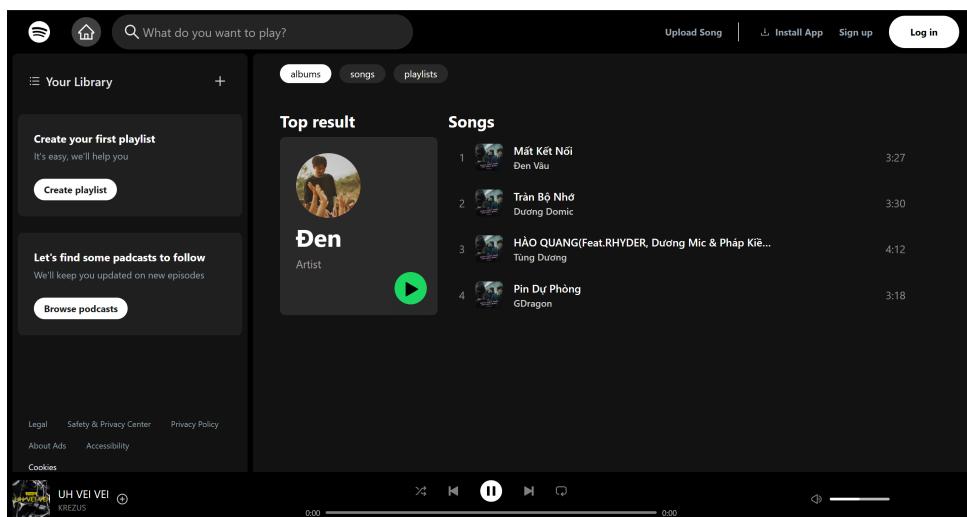
### 3.3 Tìm kiếm Users, Playlists, Songs

Chức năng tìm kiếm là một phần không thể thiếu trong các ứng dụng âm nhạc hiện đại, giúp người dùng nhanh chóng truy cập các nội dung mong muốn như người dùng khác, danh sách phát (playlist), hoặc bài hát. Trong hệ thống của chúng em, thanh tìm kiếm được bố trí cố định trên thanh header nhằm tạo sự thuận tiện và dễ tiếp cận cho người dùng.

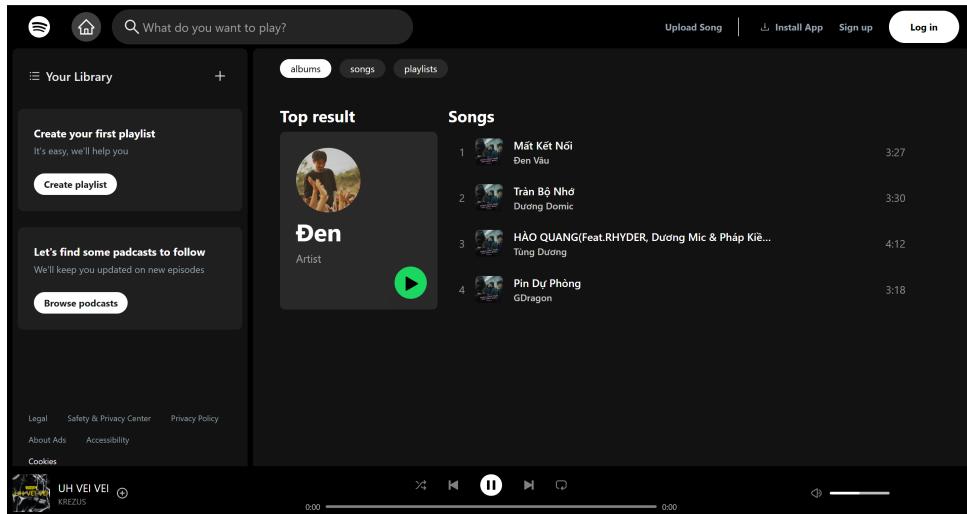
Khi người dùng nhập từ khóa vào ô tìm kiếm, hệ thống sẽ gợi ý và hiển thị các kết quả phù hợp một cách linh hoạt và tức thì. Các bước xử lý chức năng tìm kiếm như sau:

- Khi người dùng nhập dữ liệu, sự kiện `input` được kích hoạt trên ô tìm kiếm.
- Một đoạn mã JavaScript sẽ lấy giá trị nhập vào và gửi yêu cầu AJAX đến URL `/search/`, với tham số `q` là chuỗi tìm kiếm.
- Máy chủ xử lý truy vấn, lọc danh sách người dùng, playlist hoặc bài hát phù hợp và trả về kết quả dưới dạng JSON.
- Kết quả tìm kiếm sau đó được hiển thị trực tiếp bên dưới ô tìm kiếm, dưới dạng danh sách các phần tử `<li>`, mỗi phần tử chứa ảnh đại diện (nếu có), tên người dùng hoặc tên bài hát/danh sách phát.
- Nếu ô tìm kiếm rỗng, kết quả hiển thị sẽ bị xóa (sử dụng phương thức `empty()` để làm trống danh sách).

Chức năng này mang lại trải nghiệm người dùng tốt hơn, giúp thao tác nhanh chóng và tương tác hiệu quả trong hệ thống.



Hình 8: Giao diện thanh tìm kiếm



Hình 9: Code Python xử lý tìm kiếm phía server

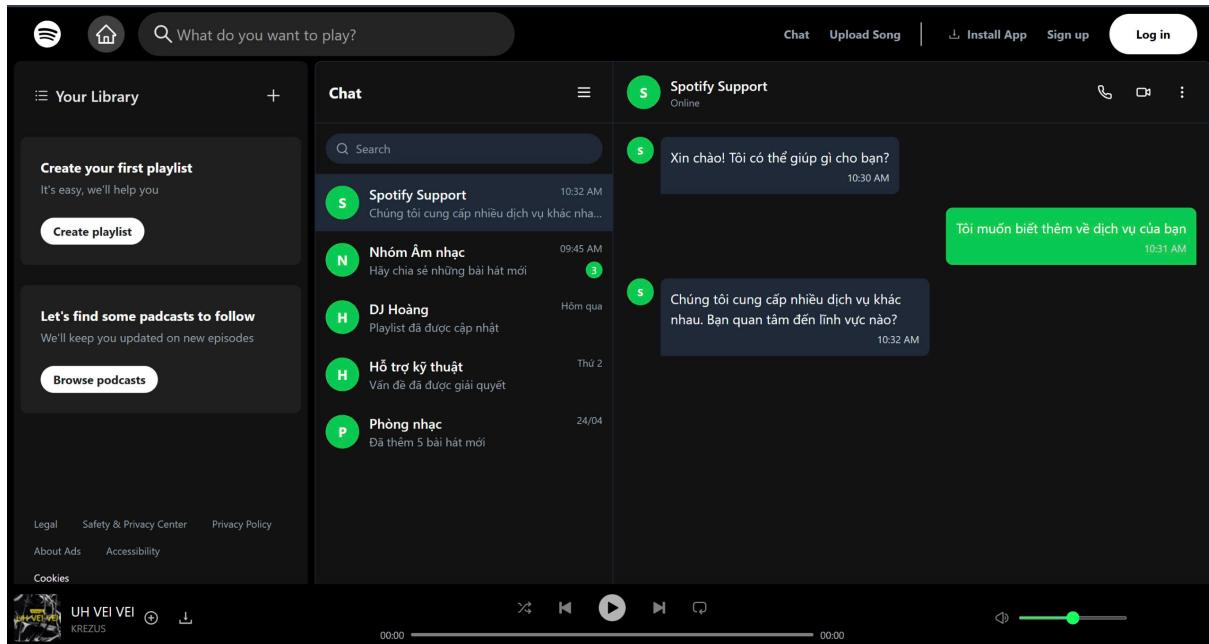
### 3.3.0.1 Xử lý phía server

- Nhận tham số request, đại diện cho yêu cầu từ client.
- Trích xuất từ khóa tìm kiếm từ yêu cầu.
- Tìm kiếm trong cơ sở dữ liệu các người dùng có thông tin liên quan đến từ khóa tìm kiếm.
- Thu thập thông tin của các người dùng phù hợp và đóng gói chúng thành một danh sách kết quả.
- Trả về kết quả dưới dạng JSON, chứa thông tin về các người dùng phù hợp hoặc một danh sách rỗng nếu không tìm thấy kết quả nào.

## 4 Nhắn tin

### 4.1 Các cuộc hội thoại (Conversation)

Chức năng này hiển thị danh sách các cuộc trò chuyện mà người dùng tham gia. Mỗi cuộc trò chuyện được hiển thị với thông tin cơ bản như tên người gửi, nội dung tin nhắn cuối cùng và thời gian gửi. Khi người dùng chọn một cuộc trò chuyện, họ sẽ được chuyển đến trang chi tiết của cuộc trò chuyện đó.



Hình 10: Giao diện danh sách các tin nhắn

#### 4.1.0.1 Code xử lý hiển thị tin nhắn

#### 4.1.0.2 Giải thích code

- `@login_required`: Đây là một decorator của Django, được sử dụng để đảm bảo rằng chỉ những người dùng đã đăng nhập mới có thể truy cập hàm này. Nếu người dùng chưa đăng nhập, họ sẽ được chuyển hướng đến trang đăng nhập.
- `user_id = request.user`: Lấy thông tin người dùng hiện tại từ đối tượng `request` và gán nó vào biến `user_id`.
- `chat_message = ChatMessage.objects.filter(...)`: Tạo một truy vấn để lấy tất cả các tin nhắn (`ChatMessage`) liên quan đến người dùng hiện tại. Cụ thể:

- Subquery và OuterRef được sử dụng để lấy ID của tin nhắn cuối cùng trong mỗi cuộc trò chuyện giữa người dùng hiện tại và các người dùng khác.
  - `Q(sender__reciever=user_id) | Q(reciever__sender=user_id)`: Sử dụng `Q objects` để tạo điều kiện lọc phức tạp, lấy tin nhắn mà người dùng hiện tại là người gửi hoặc người nhận.
  - `.distinct().annotate(last_msg=Subquery(...))`: Loại bỏ các bản ghi trùng lặp và gắn thêm thông tin về tin nhắn cuối cùng.
  - `.values_list('last_msg', flat=True).order_by("-id")`: Lấy danh sách ID của các tin nhắn cuối cùng và sắp xếp chúng theo thứ tự giảm dần.
- `context = {'chat_message': chat_message}`: Tạo một từ điển `context` chứa danh sách các tin nhắn để truyền đến template.
  - `return render(request, 'chat/inbox.html', context)`: Sử dụng hàm `render` để tạo `HttpResponse` dựa trên template `chat/inbox.html` và từ điển `context` đã tạo. `HttpResponse` này sẽ hiển thị danh sách các tin nhắn trong hộp thư đến của người dùng.

## 4.2 Chi tiết cuộc trò chuyện (Conservation detail)

Trang chi tiết cuộc trò chuyện hiển thị toàn bộ lịch sử tin nhắn trong cuộc trò chuyện đó. Tin nhắn được sắp xếp theo thứ tự thời gian, với tin nhắn mới nhất ở cuối. Người dùng có thể xem tin nhắn đã gửi và nhận, và cũng có thể gửi tin nhắn mới từ trang này. Tin nhắn mới gửi đi sẽ được hiển thị ngay lập tức trong cuộc trò chuyện.

### 4.2.0.1 Giao diện gồm 3 phần như hình, mỗi phần lần lượt là:

- Phần 1: Thanh header chứa các label như Avatar người gửi, Tên người gửi, Trạng thái Online/Offline, Kèm với các button Call, Video call và thông tin chi tiết của đoạn chat.
- Phần 2: Content của đoạn chat sẽ nằm ở giữa và hiển thị toàn bộ lịch sử tin nhắn của người dùng sắp xếp theo thứ tự thời gian từ quá khứ đến hiện tại.
- Phần 3: Giao diện cuối cùng là thanh nhập nội dung cần gửi với các button như gửi File, thả cảm xúc, gửi giọng nói, và nút send khi người dùng nhập text nội dung muốn gửi.

Hàm được sử dụng để hiển thị chi tiết hộp thư đến của người dùng dựa trên username được cung cấp:

- `@login_required`: Decorator này đảm bảo rằng chỉ người dùng đã đăng nhập mới có thể truy cập hàm này.
- `user_id = request.user`: Lấy thông tin người dùng hiện tại từ yêu cầu.
- `message_list`: Lấy danh sách các tin nhắn cuối cùng trong mỗi cuộc trò chuyện giữa người dùng hiện tại và các người dùng khác.
- `sender` và `receiver`: Đặt người gửi và người nhận dựa trên thông tin người dùng hiện tại và username.
- `messages_detail`: Lọc các tin nhắn giữa người gửi và người nhận và sắp xếp chúng theo ngày.
- `messages_detail.update(is_read=True)`: Cập nhật trạng thái của các tin nhắn thành đã đọc.
- `context`: Tạo một từ điển chứa thông tin chi tiết về tin nhắn, người gửi, người nhận và danh sách tin nhắn.

**4.2.0.2 Đoạn code thiết lập một kết nối WebSocket để giao tiếp hai chiều thời gian thực, cho phép người dùng gửi và nhận tin nhắn một cách nhanh chóng và hiệu quả:**

- Biến `receiver` được khởi tạo với giá trị `null` và sau đó được gán giá trị dựa trên username của người nhận và người dùng đã đăng nhập.
- Đoạn code tạo URL cho kết nối WebSocket bằng cách sử dụng host hiện tại của trang web và username của người nhận.
- Các hàm xử lý sự kiện `onopen`, `onmessage`, và `onclose` được định nghĩa để xử lý các trạng thái khác nhau của kết nối WebSocket, bao gồm việc thiết lập, nhận tin nhắn, và đóng kết nối.
- Hàm `onmessage` hiển thị đoạn html cuộc trò chuyện giữa người nhận và người gửi tin nhắn.

```
 1 import { Send, Paperclip, Mic, Smile } from "lucide-react";
 2 import { useChat } from "../../contexts/Chat";
 3
 4 const MessageInput = () => {
 5   const { inputMessage, setInputMessage, sendMessage, fetchConversations } =
 6     useChat();
 7
 8   const handleSetInputMessage = (e) => {
 9     setInputMessage(e.target.value);
10   };
11
12   const handleKeyPress = async (e) => {
13     if (e.key === "Enter" && !e.shiftKey) {
14       e.preventDefault();
15       sendMessage(e);
16       await fetchConversations();
17     }
18   };
19
20   const handleSendClick = async (e) => {
21     e.preventDefault();
22     sendMessage(e);
23     await fetchConversations();
24   };
25
26   return (
27     <div className='p-4 border-t border-gray-800'>
28       <div className='flex items-center bg-gray-800 rounded-full p-2 ps-4'>
29         <button className='text-gray-400 mr-2'>
30           <Paperclip size={20} className='cursor-pointer' />
31         </button>
32         <input
33           type='text'
34           value={inputMessage}
35           onChange={handleSetInputMessage}
36           onKeyDown={handleKeyPress}
37           placeholder='Nhập tin nhắn...'
38           className='flex-1 bg-transparent outline-none text-white'
39         />
40         <button className='text-gray-400 mx-2'>
41           <Smile size={20} className='cursor-pointer' />
42         </button>
43         <button className='text-gray-400 mr-2'>
44           <Mic size={20} className='cursor-pointer' />
45         </button>
46         <button
47           onClick={handleSendClick}
48           className='bg-green-500 w-fit h-8 px-2 rounded-full flex items-center justify-center gap-2 cursor-pointer'>
49           <Send size={16} />
50           <p className='text-sm'>Send</p>
51         </button>
52       </div>
53     </div>
54   );
55 };
56
57 export default MessageInput;
58
```

Hình 11: Code xử lý khi nhấn nút gửi

```
1 const sendMessage = useCallback(
2     (event) => {
3         event?.preventDefault();
4
5         if (!inputMessage.trim() || !socket || !activeConversation || !user?.id) {
6             return;
7         }
8
9         try {
10             const messageData = {
11                 message: inputMessage,
12                 senderId: user?.id,
13                 receiverId: activeConversation,
14             };
15
16             socket.send(JSON.stringify(messageData));
17             setInputMessage("");
18         } catch (error) {
19             console.error("Error sending message:", error);
20             setError("Failed to send message");
21         }
22     },
23     [inputMessage, socket, activeConversation, user?.id, user?.name]
24 );
```

Hình 12: Code xử lý hàm Send()

- **Lấy Dữ Liệu Đầu Vào:** Khi ấn vào Gửi tin nhắn, code lấy giá trị của tin nhắn từ ô và lưu vào biến `message`. Sau đó lấy người dùng hiện tại lưu vào `sender`.
- Nếu tin nhắn có ở dạng file hoặc ảnh thì lưu src của nó vào.
- Các trường dữ liệu được lưu chung vào biến `data`, sau đó dùng `socket.send()` để gửi dữ liệu đến máy chủ thông qua kết nối WebSocket.

## 5 Songs và Albums

### 5.1 Xem và tương tác với albums của người dùng khác (play, follow)

Để người dùng có thể tận hưởng và khám phá nội dung âm nhạc từ những người dùng khác, hệ thống cho phép hiển thị chi tiết các album được chia sẻ công khai. Tại trang chi tiết album, người dùng có thể nghe từng bài hát trong album, theo dõi (follow) người tạo album hoặc thêm album vào danh sách yêu thích của riêng mình. Giao diện được thiết kế trực quan, với các nút điều khiển phát nhạc, hiển thị ảnh bìa, tên album, danh sách bài hát, cũng như các thông tin về người tạo album, giúp tăng trải nghiệm tương tác và kết nối giữa các người dùng.



Hình 13: Giao diện Albums Detail

#### Xử lý "like" album/track (Ví dụ)

Để xử lý việc người dùng "like" một album hoặc track:

- Trong yêu cầu Ajax, truyền một tham số id đại diện cho ID của album/track mà người dùng muốn "like".
- Trong view Django (ví dụ `like_item_view`), đọc giá trị của tham số id từ `request.GET['id']` và sử dụng nó để truy vấn đối tượng (Album/Track) từ cơ sở dữ liệu.
- Kiểm tra xem người dùng hiện tại đã "like" hay chưa. Nếu đã "like", gỡ bỏ họ khỏi danh sách "likes". Ngược lại, thêm họ vào danh sách "likes".

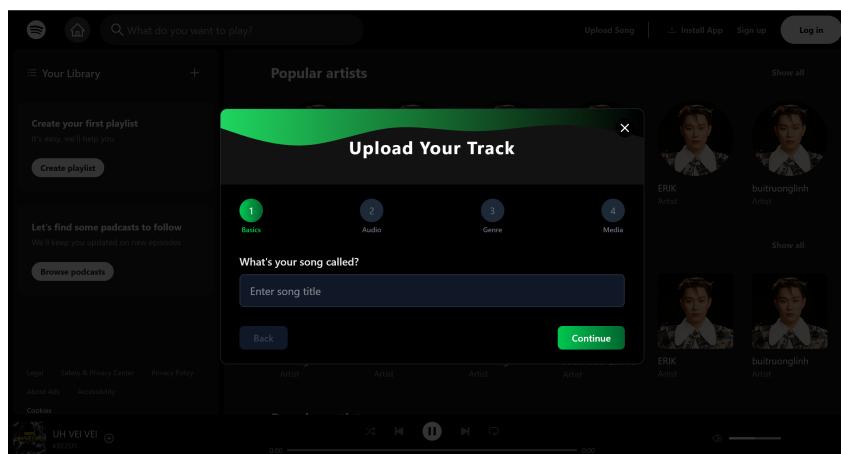
- Nếu người dùng "like" một item của người khác, một thông báo có thể được gửi đến chủ sở hữu của item đó (ví dụ: sử dụng hàm `send_notification`).
- Cuối cùng, dữ liệu trả về bao gồm một JSON chứa trạng thái thành công và tổng số lượt "like" mới.

```
<AiFillHeart>
  className='text-5xl ■text-white hover:scale-[1.1] cursor-pointer'
  onClick={async() => {
    setIsFavorited(!isFavorited);
    await removeSongFromPlaylist(song?.id)
  }}
/>
(
<AiOutlineHeart>
  className='text-5xl ■text-white hover:scale-[1.1] cursor-pointer'
  onClick={async() => {
    setIsFavorited(!isFavorited);
    await addSongToPlaylist({
      playlist_id: favoritePlaylist.id,
      song_id: song.id,
    });
  }}
/>
```

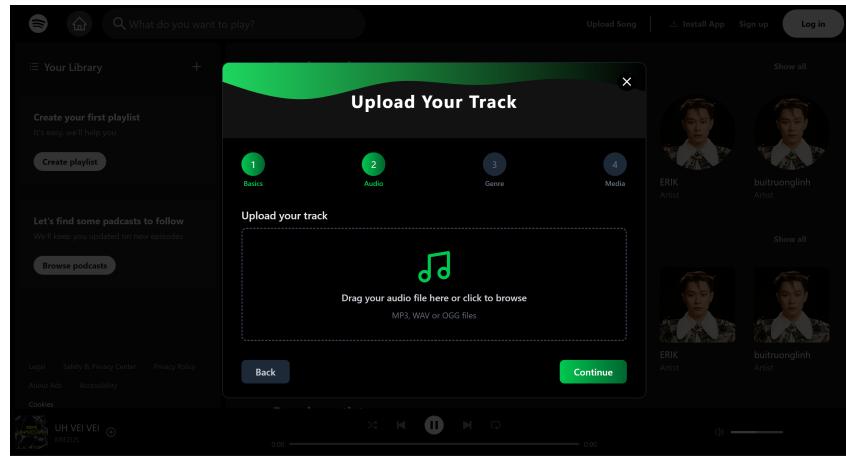
Hình 14: Code xử lý "like" phía server (Ví dụ)

## 5.2 Tạo Songs

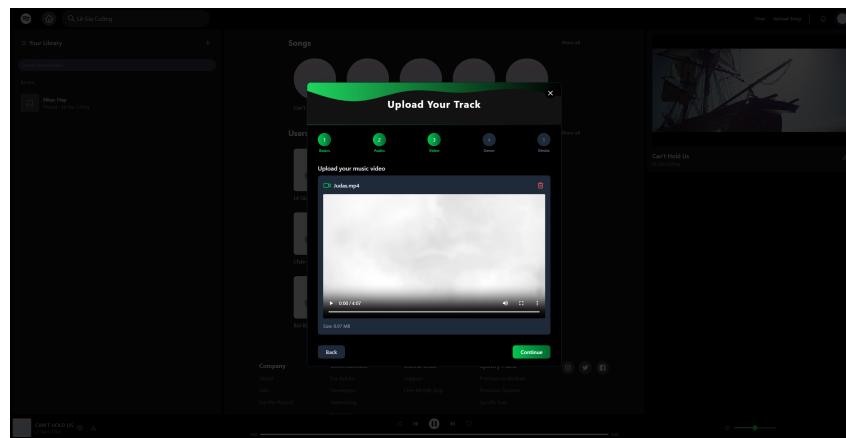
Chức năng tạo Tracks cho phép người dùng đăng tải các bài hát (track) của riêng mình để hiển thị trên trang cá nhân, từ đó chia sẻ với cộng đồng người dùng khác. Quá trình tạo track được thực hiện thông qua các bước đơn giản và trực quan.



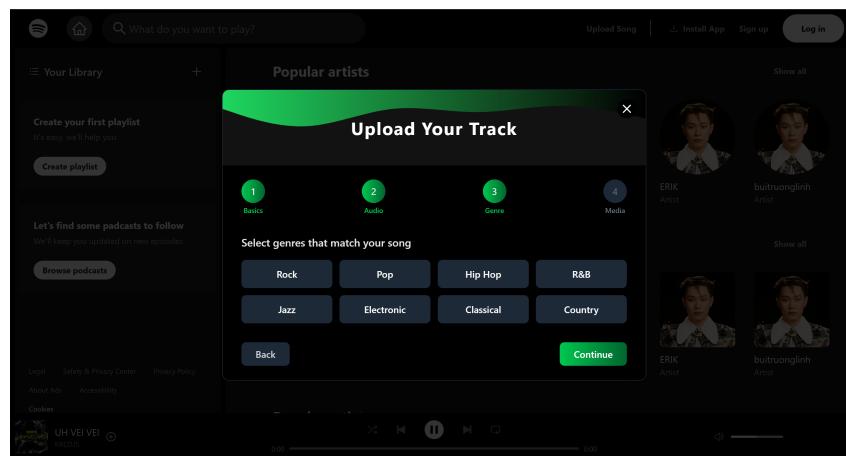
Hình 15: Bước 1: Giao diện tạo bài hát - Nhập tên bài hát



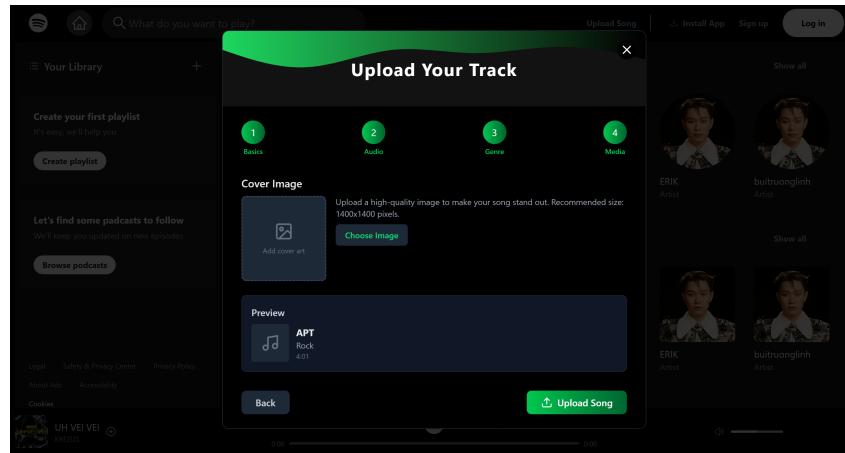
Hình 16: Bước 2: Giao diện tạo bài hát - Upload File nhạc



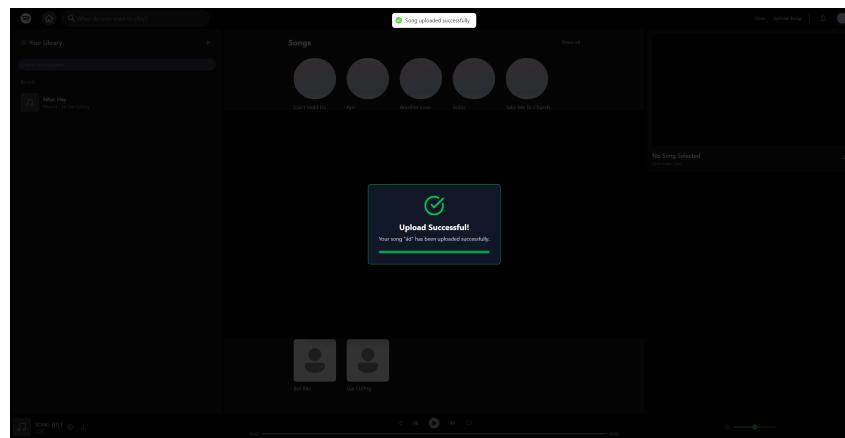
Hình 17: Bước 3: Giao diện tạo bài hát - Upload File MV ca nhạc (nếu có)



Hình 18: Bước 4: Giao diện tạo bài hát - Upload thể loại nhạc



**Hình 19:** Bước 5: Giao diện tạo bài hát - Upload cover Image và hoàn tất



**Hình 20:** Bước 6: Giao diện tạo bài hát - Sau khi Upload thành công

## Các bước xử lý tạo track

- **Bước 1:** Giao diện upload yêu cầu người dùng nhập tiêu đề tên bài hát.
- **Bước 2:** Người dùng upload file nhạc cho track.
- **Bước 3:** Người dùng upload file video nhạc cho track.
- **Bước 4:** Thiết lập các thông tin như thể loại cho bài hát.
- **Bước 5:** Upload cover image avatar cho bài hát và xác nhận lại các thông tin vừa chọn dựa trên phần preview và xác nhận tạo bài hát.
- **Bước 6:** Gửi yêu cầu lên server – hệ thống xử lý dữ liệu bằng cách:
  - Kiểm tra phương thức POST và các trường dữ liệu hợp lệ (tiêu đề, file nhạc, ảnh bìa...).
  - Sinh một mã định danh duy nhất cho track bằng thư viện `shortuuid`.
  - Nếu hợp lệ, tiến hành lưu track mới vào cơ sở dữ liệu và trả về kết quả dưới dạng JSON.
  - Nếu có lỗi, phản hồi JSON sẽ kèm theo thông báo cụ thể.

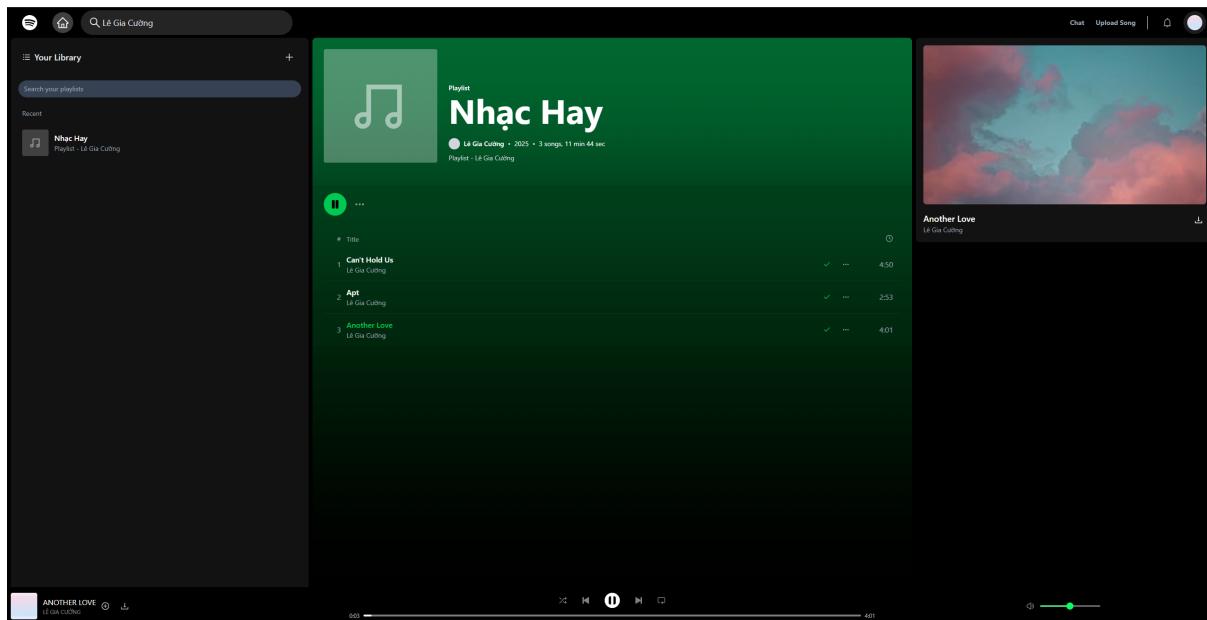
## 5.3 Playlists

### Chức năng Playlist dành cho người dùng

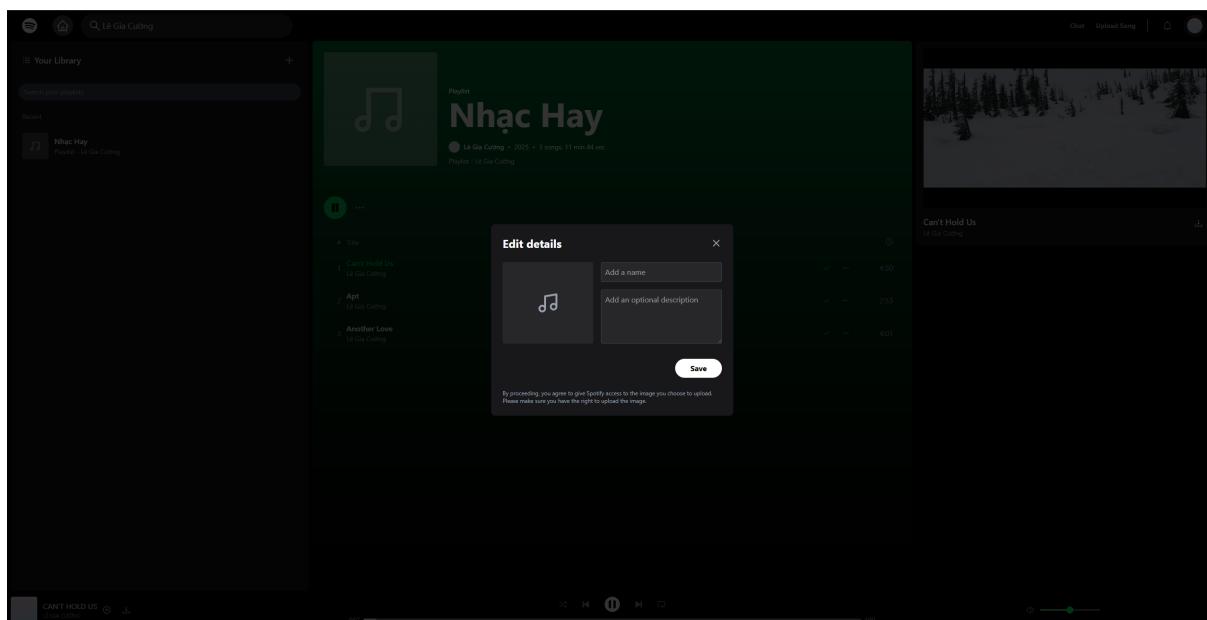
Ứng dụng cho phép người dùng cá nhân hóa trải nghiệm nghe nhạc bằng cách tạo và quản lý Playlist riêng, đồng thời có thể truy cập các Album được phát hành bởi nghệ sĩ.

- Người dùng có thể tạo Playlist mới, đặt tên và thêm mô tả tùy chọn.
- Cho phép thêm hoặc xoá các bài hát yêu thích vào Playlist.
- Mỗi Playlist có thể cập nhật ảnh bìa và thông tin mô tả.
- Cho phép xem chi tiết nội dung một Playlist hoặc Album: danh sách bài hát, ảnh bìa, người tạo, ngày phát hành.
- Người dùng có thể chia sẻ Playlist qua liên kết công khai nếu muốn.
- Có thể đánh dấu Playlist là riêng tư (chỉ mình xem) hoặc công khai (người khác có thể xem).

- Hỗ trợ phát ngẫu nhiên toàn bộ bài hát trong một Playlist.
- Cho phép nghe thử từng bài hát trong Playlist trực tiếp từ giao diện người dùng.



Hình 21: Giao diện Playlist dành cho người dùng



Hình 22: Giao diện khi tạo bài hát

## 6 Admin

### 6.1 Song Managements

#### Chức năng quản lý bài hát dành cho Admin

Hệ thống cung cấp cho quản trị viên (Admin) một giao diện riêng để quản lý tất cả các bài hát (tracks) được đăng tải bởi người dùng trong hệ thống. Chức năng này nhằm đảm bảo nội dung được kiểm soát chặt chẽ, tránh vi phạm chính sách và cải thiện trải nghiệm người dùng.

- Hiển thị danh sách toàn bộ bài hát từ tất cả người dùng, bao gồm: tên bài hát, người đăng, thể loại, trạng thái (hiển thị / bị ẩn), ngày tạo.
- Cho phép tìm kiếm bài hát theo tên, ID, người đăng hoặc thể loại.
- Cho phép lọc các bài hát theo tình trạng (ví dụ: đang hiển thị, bị báo cáo, đang chờ kiểm duyệt...).
- Có thể chỉnh sửa thông tin bài hát, cập nhật trạng thái hoặc xóa bài hát khỏi hệ thống.
- Tích hợp chức năng xem chi tiết một bài hát bao gồm: ảnh bìa, thời lượng, thông tin người đăng và số lượt nghe.
- Có thể khóa một người dùng nếu bài hát đăng tải vi phạm nghiêm trọng.

The screenshot shows the Spotify Clone Project Admin interface. On the left, there's a sidebar with navigation links: Dashboard, Song Management (which is currently selected and highlighted in blue), User Management, Analytics, Settings, and Sign out. The main content area is titled "Song Management". It features a search bar at the top right with placeholder text "Search by title, artist, or genre". Below the search bar is a "Filter" button. The main area is a table with the following columns: No, Song, Genre, Duration, Released, and Actions. There are 6 rows of data:

No	Song	Genre	Duration	Released	Actions
1	APT User2	Pop	2:53	N/A	
2	Another Love User2	Rock	4:01	N/A	
3	Can't Hold Us User2	Rock	4:50	N/A	
4	Take Me To Church User2	Jazz Classical	4:00	N/A	
5	Judas User2	Rock Classical	4:09	N/A	
6	APT User2	Rock	4:01	N/A	

Total 6 songs

Hình 23: Giao diện quản lý bài hát của Admin

## 6.2 User Managements

### Chức năng quản lý người dùng dành cho Admin

Hệ thống cung cấp cho quản trị viên (Admin) một giao diện riêng để quản lý thông tin tài khoản người dùng, giúp kiểm soát hoạt động và xử lý các trường hợp vi phạm chính sách sử dụng nền tảng.

- Hiển thị danh sách toàn bộ người dùng, bao gồm: tên người dùng, email, vai trò (user/admin), trạng thái tài khoản (đang hoạt động / bị khóa), ngày đăng ký.
- Cho phép tìm kiếm người dùng theo tên, email hoặc ID.
- Cho phép lọc người dùng theo trạng thái tài khoản (đang hoạt động, bị khóa, đang bị báo cáo...).
- Có thể cập nhật thông tin người dùng (ví dụ: tên, vai trò), khóa hoặc mở khóa tài khoản.
- Tích hợp chức năng xem chi tiết một người dùng bao gồm: danh sách bài hát đã đăng, số lượt nghe, danh sách album, và hoạt động gần đây.
- Có thể xóa tài khoản người dùng vĩnh viễn trong trường hợp vi phạm nghiêm trọng.

ID	Name	Email	Actions
681act09e522e4da1ad34ba5	admin	admin@gmail.com	Stats Delete
681acc0af522e4da1ad348a6	Lê Gia Cường	gc@gmail.com	Stats Delete
681acc0ef522e4da1ad348a7	Cẩm Nhựng	cn@gmail.com	Stats Delete
681acc0be522e4da1ad34ba8	Minh Cảnh	mc@gmail.com	Stats Delete
681acc0fe522e4da1ad34a9	Kim Bạc	kb@gmail.com	Stats Delete
681b0b80a0da1da5c1f03385	Như Quỳnh	nq@gmail.com	Stats Delete
681b0b82a0da1da5c1f03389	Chân Phong	cp@gmail.com	Stats Delete
681b0b8a0e5fda15c21f03387	Sỹ Khâm	sk@gmail.com	Stats Delete
681b0babef5fda15c21f03388	Minh Trung	mt@gmail.com	Stats Delete
681b0babef5fda15c21f03389	Dinh Văn	dv@gmail.com	Stats Delete

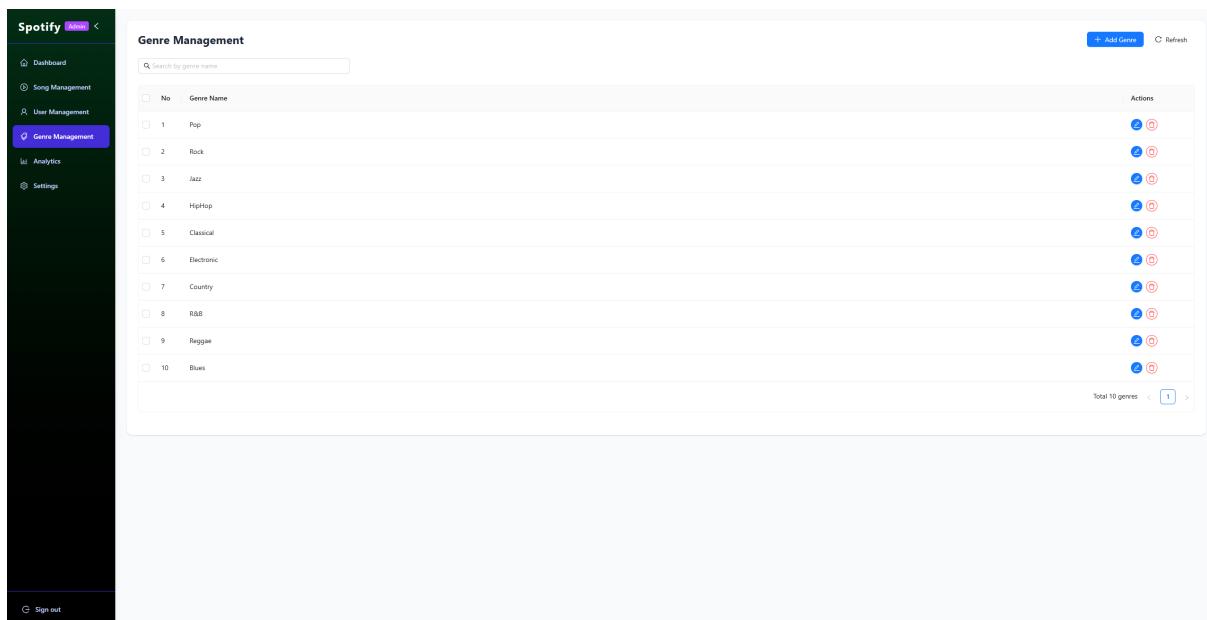
**Hình 24:** Giao diện quản lý người dùng của Admin

## 6.3 Genres Managements

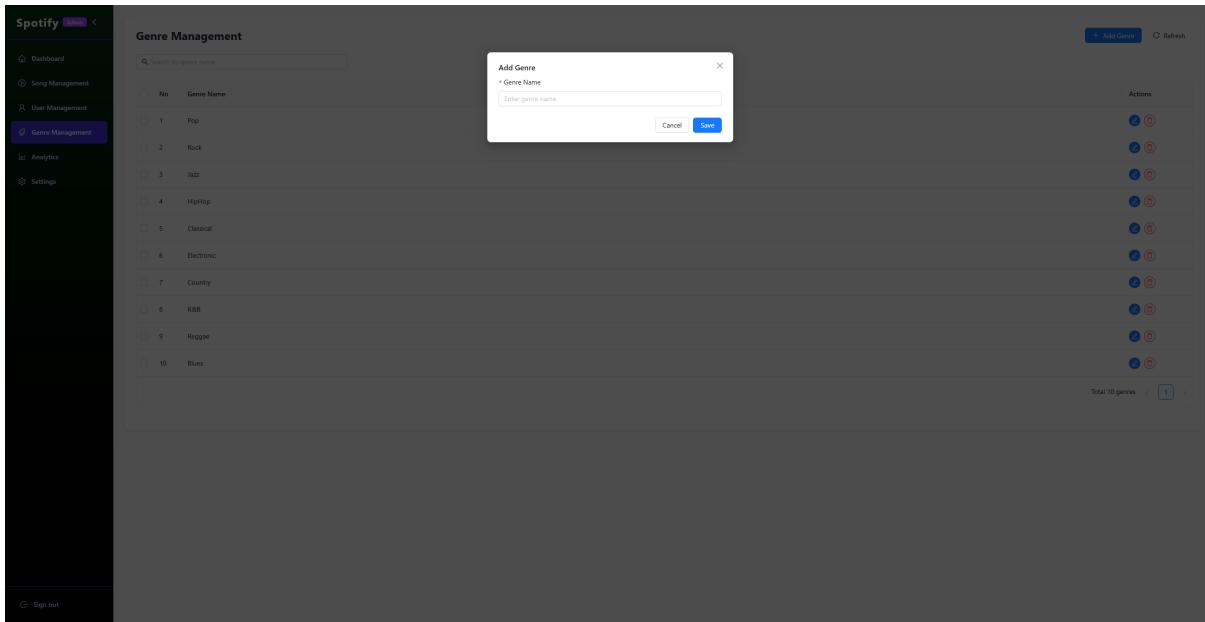
### Chức năng quản lý thể loại nhạc dành cho Admin

Chức năng quản lý thể loại (genres) giúp quản trị viên dễ dàng thiết lập, cập nhật và kiểm soát danh sách các thể loại nhạc hiện có trên hệ thống. Việc tổ chức thể loại rõ ràng giúp nâng cao trải nghiệm người dùng khi tìm kiếm và phân loại nội dung âm nhạc.

- Hiển thị danh sách tất cả thể loại đang được sử dụng trong hệ thống, bao gồm: tên thể loại, mô tả (nếu có), số lượng bài hát thuộc thể loại đó.
- Cho phép tìm kiếm thể loại theo tên hoặc ID.
- Cho phép thêm mới thể loại nhạc để người dùng có thể lựa chọn khi đăng bài hát.
- Có thể chỉnh sửa tên hoặc mô tả của thể loại hiện có.
- Có thể xóa thể loại khỏi hệ thống. Khi xóa, hệ thống sẽ yêu cầu xác nhận và có thể chuyển các bài hát thuộc thể loại đó sang một thể loại mặc định.
- Đảm bảo rằng tên thể loại là duy nhất và không được trùng lặp.



**Hình 25:** Giao diện quản lý thể loại nhạc của Admin



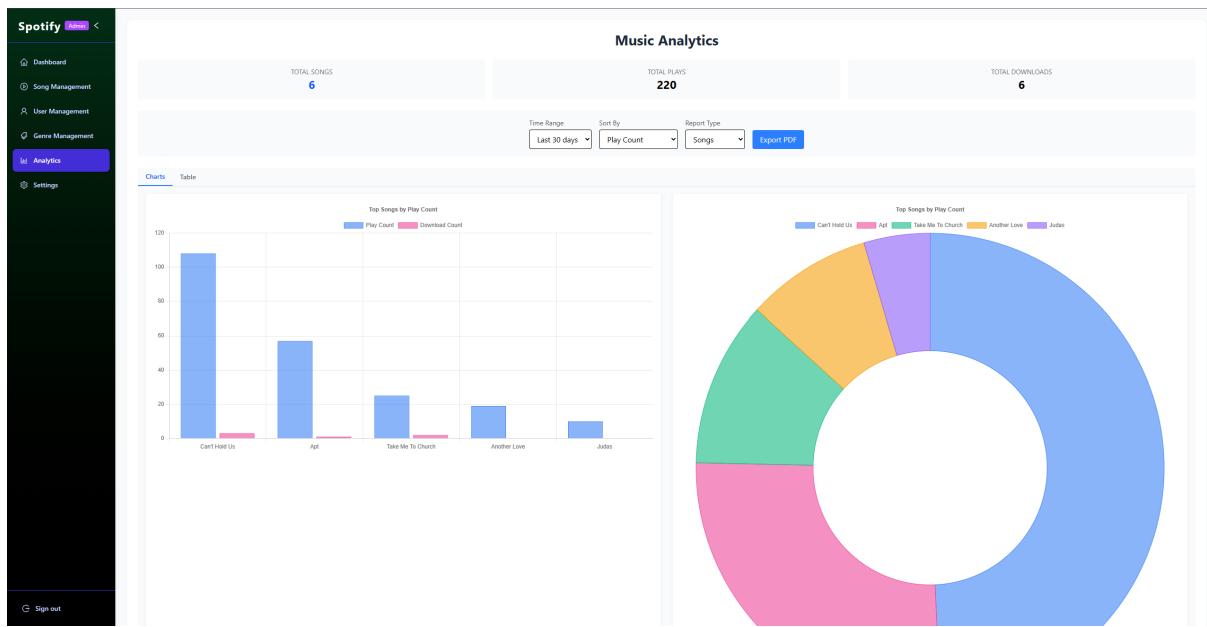
**Hình 26:** Giao diện tạo mới một thể loại nhạc của Admin

## 6.4 Thông Kê

### Chức năng thống kê dành cho Admin

Chức năng thống kê giúp quản trị viên (Admin) nắm bắt được toàn bộ hoạt động trên hệ thống thông qua các số liệu trực quan, từ đó đưa ra quyết định phù hợp nhằm cải thiện hiệu suất và chất lượng dịch vụ.

- Thống kê số lượng người dùng đăng ký theo ngày, tuần, tháng.
- Thống kê tổng số bài hát và video được tải lên hệ thống.
- Thống kê số lượt nghe theo thời gian, theo bài hát, hoặc theo người dùng.
- Thống kê top bài hát và video được nghe/stream nhiều nhất.
- Thống kê số lượng báo cáo vi phạm, người dùng bị khóa và bài hát bị ẩn.
- Biểu đồ thể hiện tỷ lệ phân bố các thể loại nhạc phổ biến.
- Hỗ trợ xuất báo cáo thống kê dưới dạng PDF hoặc Excel để lưu trữ hoặc chia sẻ.



Hình 27: Giao diện thống kê hoạt động trên hệ thống

## 7 Kết Quả Đạt Được và Đánh Giá

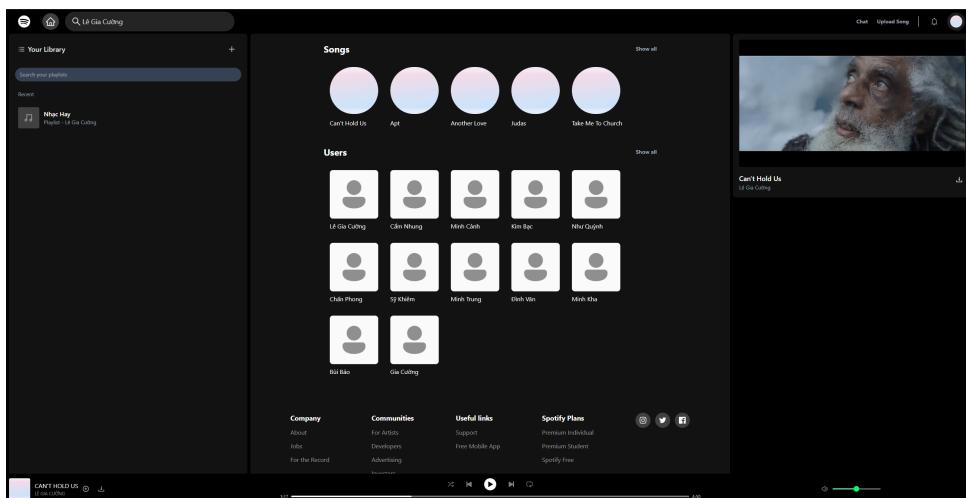
Chương này tập trung trình bày các kết quả chính mà dự án đã đạt được sau quá trình triển khai, đồng thời đưa ra những đánh giá khách quan về sản phẩm Spotify Clone đã xây dựng, bao gồm các chức năng đã hoàn thành, những ưu điểm, nhược điểm còn tồn tại và những bài học kinh nghiệm quan trọng mà nhóm đã rút ra.

### 7.1 Kết quả đạt được

Sau thời gian nỗ lực thực hiện, dự án đã xây dựng thành công ứng dụng web Spotify Clone với các chức năng cốt lõi sau, đáp ứng các mục tiêu chính đã đề ra:

- Xác thực người dùng:** Cho phép người dùng đăng ký tài khoản mới, đăng nhập vào hệ thống và đăng xuất.
- Quản lý hồ sơ cá nhân:** Người dùng có thể xem và cập nhật các thông tin cơ bản của mình (ví dụ: tên hiển thị, ảnh đại diện).
- Tìm kiếm nội dung:** Cung cấp khả năng tìm kiếm bài hát, album, nghệ sĩ dựa trên từ khóa nhập vào.
- Phát nhạc trực tuyến:** Xây dựng thành công trình phát nhạc với các chức năng điều khiển cơ bản: play, pause, next, previous, điều chỉnh âm lượng, thanh tiến trình.

- **Quản lý Playlist cá nhân:** Người dùng có thể tạo mới playlist, đổi tên, xóa playlist, thêm bài hát vào playlist và xóa bài hát khỏi playlist.
- **Tương tác với nội dung:** Cho phép người dùng "Thích" (Like) các bài hát, album yêu thích và xem lại danh sách đã thích.
- **Hiển thị thông tin chi tiết:** Giao diện hiển thị thông tin chi tiết cho từng bài hát và album (bao gồm danh sách bài hát).
- **(Nếu có) Tải lên bài hát:** Chức năng cho phép người dùng (Admin/Nghệ sĩ) tải lên file âm thanh và thông tin metadata.
- **(Nếu có) Nhắn tin cơ bản:** Chức năng chat 1-1 giữa các người dùng.
- **Giao diện quản trị (Admin):** Cung cấp các chức năng cơ bản để quản trị viên quản lý người dùng và nội dung âm nhạc.



Hình 28: Giao diện chính của ứng dụng Spotify Clone.



Hình 29: Giao diện trình phát nhạc.

## 7.2 Đánh giá sản phẩm

Dựa trên các kết quả đạt được và quá trình phát triển, nhóm xin đưa ra những đánh giá tổng quan về sản phẩm Spotify Clone:

### 7.2.1 Ưu điểm của hệ thống

- **Hoàn thành các chức năng cốt lõi:** Ứng dụng đã mô phỏng thành công những tính năng quan trọng nhất của một nền tảng nghe nhạc trực tuyến, tạo ra một sản phẩm có thể sử dụng được cho các nhu cầu cơ bản.
- **Ứng dụng thành công công nghệ mục tiêu:** Nhóm đã áp dụng hiệu quả bộ ba công nghệ Django - React - MongoDB, thể hiện khả năng học hỏi và tích hợp các công nghệ web hiện đại.
- **Kiến trúc Client-Server rõ ràng:** Việc tách biệt giữa backend API và frontend React giúp mã nguồn có cấu trúc tốt, dễ quản lý và tiềm năng cho việc bảo trì, mở rộng sau này.
- **Giao diện người dùng tương đối thân thiện:** Giao diện được thiết kế hướng tới sự đơn giản, cố gắng mang lại trải nghiệm tương tự Spotify, giúp người dùng dễ dàng làm quen và thao tác. (Tự đánh giá mức độ thành công)
- **Sử dụng CSDL NoSQL linh hoạt:** MongoDB đã chứng tỏ sự phù hợp trong việc lưu trữ metadata âm nhạc và thông tin người dùng một cách linh hoạt.

### 7.2.2 Nhược điểm và hạn chế

Bên cạnh những thành tựu, sản phẩm vẫn còn một số điểm hạn chế cần được nhìn nhận thẳng thắn:

- **Chức năng còn đơn giản:** So với Spotify thực tế, ứng dụng còn thiếu rất nhiều tính năng nâng cao như gợi ý nhạc, radio, podcast, nghe offline, các tính năng xã hội, v.v.
- **Trải nghiệm người dùng (UX) cần hoàn thiện:** Giao diện có thể chưa thực sự mượt mà trên mọi trình duyệt hoặc thiết bị, tốc độ phản hồi của một số chức năng có thể cần tối ưu thêm, các hiệu ứng chuyển động còn đơn giản.
- **Hiệu năng và khả năng chịu tải chưa được kiểm chứng:** Do chưa được kiểm thử với lượng lớn người dùng và dữ liệu, hiệu năng thực tế của hệ thống dưới tải nặng vẫn là một dấu hỏi.
- **Xử lý lỗi và trường hợp biên:** Có thể vẫn còn tồn tại các trường hợp lỗi chưa được lường trước hoặc xử lý chưa triệt để, đặc biệt với các dữ liệu đầu vào không mong muốn.

- **Kiểm thử còn hạn chế:** Việc chủ yếu dựa vào kiểm thử thủ công có thể bỏ sót nhiều lỗi tiềm ẩn.
- **Bảo mật:** Mặc dù đã cố gắng áp dụng các nguyên tắc bảo mật cơ bản, nhưng ứng dụng có thể vẫn còn các điểm yếu cần được rà soát kỹ hơn.
- **Vấn đề bản quyền:** Hoàn toàn chưa có cơ chế xử lý bản quyền âm nhạc.

### 7.3 Bài học kinh nghiệm

Quá trình thực hiện dự án đã mang lại cho nhóm nhiều bài học quý giá:

- **Tầm quan trọng của Phân tích và Thiết kế:** Một bản thiết kế tốt (dù là ở mức cơ bản) cho kiến trúc, CSDL và API giúp định hướng rõ ràng và giảm thiểu đáng kể việc phải sửa đổi lớn trong quá trình triển khai.
- **Sự cần thiết của Quản lý mã nguồn:** Việc sử dụng Git/GitHub một cách nhất quán, chia nhánh hợp lý và ghi chú commit rõ ràng là cực kỳ quan trọng để quản lý dự án và hợp tác hiệu quả.
- **Thách thức khi tích hợp công nghệ:** Việc kết hợp nhiều công nghệ khác nhau (Python/Django, JavaScript/React, NoSQL/MongoDB) đòi hỏi sự tìm hiểu kỹ lưỡng về cách chúng tương tác và các vấn đề tương thích có thể xảy ra.
- **Kỹ năng Gỡ lỗi (Debugging):** Khả năng đọc hiểu thông báo lỗi, sử dụng các công cụ gỡ lỗi (debugger, browser dev tools, logging) và phương pháp khoanh vùng lỗi là kỹ năng sống còn của lập trình viên.
- **Tầm quan trọng của việc chia nhỏ công việc:** Phân chia các chức năng lớn thành các nhiệm vụ nhỏ hơn, dễ quản lý hơn giúp theo dõi tiến độ và tạo động lực hoàn thành.
- **Kỹ năng tự học và tìm kiếm thông tin:** Không phải mọi vấn đề đều có sẵn lời giải, khả năng tìm kiếm tài liệu, đọc hiểu documentation và học hỏi từ cộng đồng là rất cần thiết.
- **Kiên trì và không ngại khó:** Phát triển phần mềm luôn tiềm ẩn những vấn đề không lường trước. Sự kiên trì, bình tĩnh đối mặt và tìm cách giải quyết là yếu tố quan trọng để đi đến đích.

## Tài liệu

- [1] Tài liệu chính thức Django Framework. <https://docs.djangoproject.com/en/stable/> (Truy cập lần cuối: 09/05/2025).
- [2] Tài liệu chính thức Django REST Framework. <https://www.django-rest-framework.org/> (Truy cập lần cuối: 09/05/2025).
- [3] Tài liệu chính thức ReactJS. <https://react.dev/> (Truy cập lần cuối: 09/05/2025).
- [4] Tài liệu chính thức MongoDB. <https://docs.mongodb.com/manual/> (Truy cập lần cuối: 09/05/2025).
- [5] Tài liệu chính thức Python 3. <https://docs.python.org/3/> (Truy cập lần cuối: 09/05/2025).
- [6] MDN Web Docs - Hướng dẫn JavaScript. <https://developer.mozilla.org/en-US/docs/Web/JavaScript> (Truy cập lần cuối: 09/05/2025).
- [7] Tài liệu React Router (Thư viện định tuyến cho React). <https://reactrouter.com/en/main> (Truy cập lần cuối: 09/05/2025).
- [8] Tài liệu Axios (Thư viện gọi API cho JavaScript). <https://axios-http.com/docs/intro> (Truy cập lần cuối: 09/05/2025).
- [9] Tài liệu chính thức Git SCM. <https://git-scm.com/doc> (Truy cập lần cuối: 09/05/2025).
- [10] Stack Overflow - Nền tảng hỏi đáp dành cho lập trình viên. <https://stackoverflow.com/> (Tham khảo thường xuyên trong quá trình phát triển).
- [11] W3Schools Online Web Tutorials. <https://www.w3schools.com/> (Truy cập lần cuối: 09/05/2025).