

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



Phát triển phần mềm mã nguồn mở

Phát triển phần mềm Spotify Clone

GVHD: Từ Lăng Phiêu
SV: Nguyễn Đức Huy - 3121410230
Trần Thái Hoàng - 3121410214
Trần Tấn Phát - 3121410380
Phan Thanh Hữu - 3121410246

Email liên hệ nhóm: th09062003@gmail.com

TP. HỒ CHÍ MINH, THÁNG 5/2025

Mục lục

1	GIỚI THIỆU ĐỀ TÀI	5
1.1	Giới thiệu Spotify Clone	5
1.2	Lý do chọn đề tài	5
1.3	Mục tiêu	5
1.4	Đối tượng và phạm vi sử dụng	5
2	CÔNG NGHỆ VÀ PHẦN MỀM MÃ NGUỒN MỞ SỬ DỤNG	5
2.1	Angular	5
2.2	Git + GitHub	6
2.3	WebSocket	6
2.4	Django	6
3	PHÂN TÍCH VÀ THIẾT KẾ SPOTIFY CLONE	7
3.1	Các chức năng chính	7
3.2	Mô hình ứng dụng	7
4	CÀI ĐẶT MÔI TRƯỜNG PHÁT TRIỂN	8
4.1	Cài đặt và cấu hình môi trường phát triển cho Angular	8
4.2	Cấu trúc thư mục của Project	13
4.3	Routing và phân chia các module	15
5	NHIỆM VỤ VÀ VAI TRÒ CỦA THÀNH VIÊN	16
6	KẾT QUẢ ĐẠT ĐƯỢC	17
6.1	Màn hình login	17
6.2	Màn hình Sign Up	17
6.3	Màn hình HomePage	18
6.4	Màn hình Artist Management	18
6.5	Màn hình User Management	18
6.6	Màn hình Role Management	19
6.7	Màn hình Track Management	19
6.8	Màn hình Album Management	19
6.9	Màn hình Search User	20
6.10	Màn hình Chat	20
7	ĐÁNH GIÁ VÀ ĐỊNH HƯỚNG PHÁT TRIỂN	20
7.1	Ưu điểm và nhược điểm	20
7.2	Hướng phát triển	21
8	KẾT LUẬN	22
9	TÀI LIỆU THAM KHẢO	22



LỜI CẢM ƠN

Trước hết, nhóm chúng em xin gửi lời cảm ơn chân thành đến Trường Đại học Sài Gòn, Khoa Công nghệ Thông tin đã tạo điều kiện để chúng em thực hiện tiểu luận này.

Chúng em xin bày tỏ lòng biết ơn sâu sắc đến Thầy Từ Lăng Phiêu, người đã tận tình hỗ trợ, định hướng và góp ý để chúng em có thể hoàn thiện đề tài này.

Bên cạnh đó, chúng em cũng xin gửi lời cảm ơn đến mọi người đã chia sẻ thông tin, kinh nghiệm thực tế, giúp chúng em có cái nhìn rõ hơn về việc phát triển phần mềm mã nguồn mở.

Cuối cùng, chúng em xin cảm ơn gia đình, bạn bè đã luôn động viên, hỗ trợ trong suốt quá trình thực hiện đề tài này.

Nhóm chúng em rất mong nhận được ý kiến đóng góp để hoàn thiện hơn nữa nghiên cứu này.



LỜI MỞ ĐẦU

Trong thời đại công nghệ số phát triển mạnh mẽ, các ứng dụng phát nhạc trực tuyến đã trở thành một phần không thể thiếu trong đời sống giải trí của con người. Spotify – một trong những nền tảng phát nhạc phổ biến nhất thế giới – không chỉ nổi bật bởi kho nhạc phong phú mà còn bởi giao diện hiện đại và trải nghiệm người dùng thân thiện. Điều này đã truyền cảm hứng cho nhóm chúng em thực hiện đề tài Spotify Clone như một cách học hỏi và tiếp cận với các công nghệ phát triển phần mềm hiện đại.

Thông qua đề tài này, nhóm không chỉ tìm hiểu sâu hơn về các công nghệ mã nguồn mở như Angular, Django, GitHub, mà còn thực hành xây dựng ứng dụng thực tế với mô hình SPA, kết hợp giữa frontend – backend – WebSocket. Việc tích hợp tính năng chat thời gian thực, cùng với khả năng phát nhạc, tìm kiếm và quản lý danh sách phát, đã giúp nhóm hình dung rõ hơn về quá trình xây dựng một hệ thống web hoàn chỉnh và có khả năng mở rộng.

Đồ án không chỉ là cơ hội áp dụng kiến thức đã học mà còn là dịp để nhóm rèn luyện kỹ năng làm việc nhóm, quản lý dự án, nghiên cứu công nghệ mới và giải quyết vấn đề thực tế trong quá trình phát triển phần mềm.

Nhóm xin chân thành cảm ơn thầy đã tạo điều kiện để chúng em thực hiện đề tài này. Mặc dù đã cố gắng hoàn thiện tốt nhất trong khả năng, đồ án chắc chắn vẫn còn những thiếu sót, nhóm rất mong nhận được sự góp ý để cải thiện trong tương lai.



NHẬN XÉT CỦA GIẢNG VIÊN



1 GIỚI THIỆU ĐỀ TÀI

1.1 Giới thiệu Spotify Clone

Spotify Clone là một ứng dụng nghe nhạc trực tuyến được xây dựng với mục tiêu mô phỏng lại giao diện và chức năng cơ bản của Spotify – nền tảng phát nhạc trực tuyến nổi tiếng toàn cầu. Ứng dụng cho phép người dùng nghe nhạc, xem danh sách bài hát, tìm kiếm bài hát và phát bài hát theo thời gian thực với trải nghiệm thân thiện. Dự án này được phát triển theo mô hình SPA (Single Page Application) sử dụng Angular ở phía frontend, Django ở phía backend và WebSocket cho việc xử lý dữ liệu thời gian thực.

1.2 Lý do chọn đề tài

Spotify là một trong những nền tảng nghe nhạc phổ biến và có giao diện người dùng hiện đại, logic xử lý phức tạp, đặc biệt là với tính năng phát nhạc liên tục và cập nhật trạng thái thời gian thực. Việc chọn xây dựng một Spotify Clone giúp nhóm:

- Áp dụng kiến thức về lập trình frontend/backend kết hợp.
- Trải nghiệm thực tế trong việc sử dụng công nghệ mã nguồn mở hiện đại.
- Hiểu sâu hơn về cách hoạt động của một ứng dụng truyền phát dữ liệu đa phương tiện.

1.3 Mục tiêu

- Xây dựng một phiên bản Spotify đơn giản với các chức năng: phát nhạc, tìm kiếm bài hát, hiển thị danh sách phát, giao diện thân thiện.
- Sử dụng Angular để xây dựng giao diện.
- Sử dụng Django làm backend REST API.
- Tích hợp WebSocket để xây dựng tính năng chat.
- Quản lý mã nguồn và làm việc nhóm hiệu quả thông qua Git + GitHub.

1.4 Đối tượng và phạm vi sử dụng

- Đối tượng sử dụng: người dùng cá nhân, sinh viên, lập trình viên học tập và trải nghiệm giao diện nghe nhạc hiện đại.
- Phạm vi: ứng dụng chủ yếu tập trung vào frontend với một số chức năng backend mô phỏng như phát nhạc, danh sách bài hát và xử lý trạng thái phát. Chưa tích hợp thanh toán, tải bài hát, đề xuất AI hoặc tính năng bảo mật cao.

2 CÔNG NGHỆ VÀ PHẦN MỀM MÃ NGUỒN MỞ SỬ DỤNG

2.1 Angular

Angular là framework mã nguồn mở do Google phát triển, được sử dụng để xây dựng ứng dụng web dưới dạng SPA. Angular hỗ trợ mô hình component-based, binding dữ liệu, quản lý router,



HTTP client và nhiều tiện ích khác giúp phát triển frontend nhanh chóng, dễ mở rộng. Trong dự án này, Angular được dùng để:

- Xây dựng giao diện người dùng tương tác.
- Quản lý trạng thái bài hát đang phát.
- Kết nối API từ Django để lấy danh sách nhạc.
- Tương tác với WebSocket để xây dựng tính năng chat.

2.2 Git + GitHub

Git là hệ thống quản lý phiên bản phân tán, còn GitHub là nền tảng lưu trữ mã nguồn online. Hai là công cụ mã nguồn mở phổ biến trong quản lý dự án phần mềm. Nhóm sử dụng Git để:

- Theo dõi tiến trình làm việc.
- Quản lý các nhánh (branch) và hợp nhất (merge) tính năng.
- Thực hiện pull request, ghi chú commit, và giải quyết xung đột mã nguồn khi làm việc nhóm.

2.3 WebSocket

WebSocket là một giao thức mạng giúp tạo kết nối hai chiều liên tục giữa client và server, cho phép trao đổi dữ liệu theo thời gian thực mà không cần gửi nhiều request HTTP riêng lẻ. Đây là công nghệ rất phù hợp để xây dựng các tính năng tương tác như trò chuyện (chat) hoặc điều khiển phát nhạc đồng bộ. Trong dự án Spotify Clone, WebSocket được sử dụng để triển khai tính năng chat thời gian thực, cho phép người dùng có thể:

- Gửi và nhận tin nhắn ngay lập tức trong giao diện nghe nhạc.
- Tham gia vào các phòng chat theo playlist hoặc bài hát, tạo ra trải nghiệm tương tác cộng đồng.
- Trao đổi ý kiến, chia sẻ cảm xúc khi nghe nhạc giống như đang ở cùng một không gian thực tế.

2.4 Django

Django là framework backend mã nguồn mở dùng để xây dựng các ứng dụng web nhanh chóng, bảo mật và có thể mở rộng. Nhóm sử dụng Django để:

- Tạo REST API cung cấp dữ liệu bài hát, playlist.
- Quản lý thông tin bài hát trong database (SQLite/PostgreSQL).
- Triển khai WebSocket qua Django Channels (nếu có).
- Phục vụ yêu cầu phía client từ Angular



3 PHÂN TÍCH VÀ THIẾT KẾ SPOTIFY CLONE

3.1 Các chức năng chính

1. **Đăng nhập:** Cho phép người dùng đăng nhập vào spotify clone
2. **Đăng ký:** Cho phép người dùng đăng ký tài khoản spotify clone
3. **Đăng xuất:** Cho phép người dùng thoát ra khỏi tài khoản spotify clone
4. **Chức năng phát nhạc:** cho phép người dùng phát các bài hát theo thời gian thực, điều khiển phát/tạm dừng, tua bài hát.
5. **Chức năng phát video âm nhạc:** người dùng có thể xem video ca nhạc tích hợp trong giao diện ứng dụng.
6. **Chức năng tải video âm nhạc:** hỗ trợ người dùng tải video nhạc về thiết bị (nếu được cấp phép).
7. **Chức năng người dùng tạo album, bài hát yêu thích:** người dùng có thể tạo danh sách album cá nhân, thêm bài hát vào danh sách yêu thích.
8. **Trang Admin:** quản trị viên có thể thêm/sửa/xóa bài hát, quản lý người dùng và kiểm soát nội dung.
9. **Tính năng tùy chọn - Chat thời gian thực:** tích hợp khung chat trong giao diện Spotify Clone, cho phép người dùng tương tác, trao đổi khi nghe nhạc.

3.2 Mô hình ứng dụng

Kiến trúc tổng thể

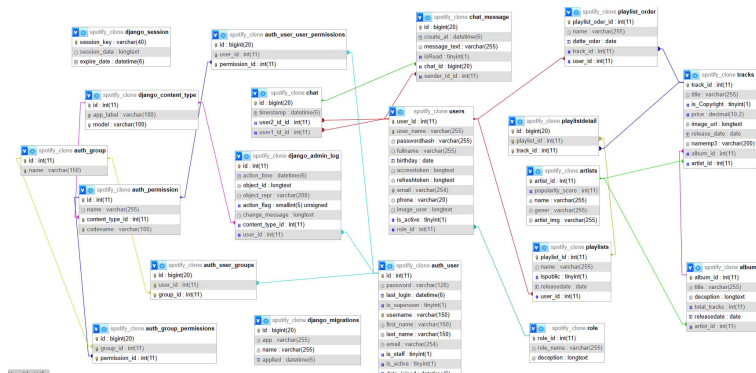
Frontend (Client):

- Sử dụng **Angular** để xây dựng giao diện **SPA (Single Page Application)**.
- Gửi yêu cầu HTTP đến Django để lấy dữ liệu bài hát, danh sách phát, người dùng.
- Kết nối đến server **WebSocket** để gửi/nhận tin nhắn chat.
- Quản lý trạng thái phát nhạc, chuyển bài, hiển thị lời bài hát.

Backend (Server):

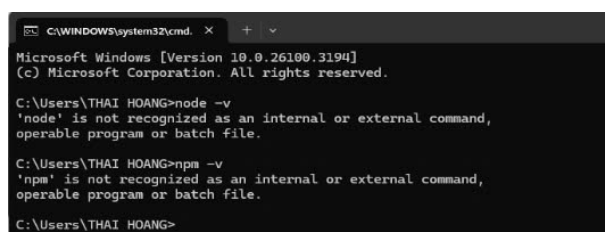
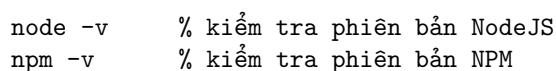
- Sử dụng **Django REST Framework** để tạo API cung cấp dữ liệu nhạc, danh sách phát, người dùng.
- Sử dụng **Django Channels** để xử lý kết nối WebSocket (tin nhắn chat thời gian thực).
- Quản lý cơ sở dữ liệu (bằng **SQLite** hoặc **PostgreSQL**).
- Đảm nhận các chức năng xác thực người dùng và xử lý nghiệp vụ.

Cơ sở dữ liệu (Database):



4.1 Cài đặt và cấu hình môi trường phát triển cho Angular

- Nhấn tổ hợp phím **Windows + R**, gõ cmd và nhấn Enter.
- Gõ các lệnh sau để kiểm tra:



Có thể thấy rằng chúng ta chưa cài đặt môi trường NodeJS. Bây giờ chúng ta tiến hành cài đặt NodeJS.

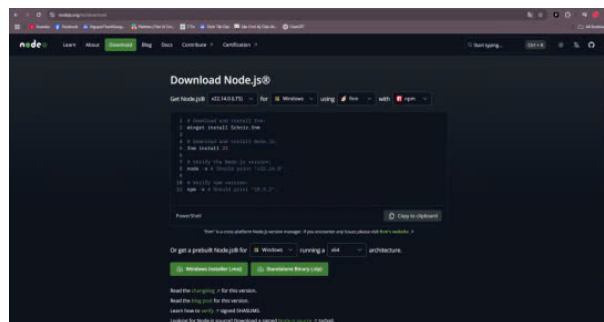
Tải NodeJS

Truy cập vào trang chính thức để tải NodeJS: <https://nodejs.org/en/download/>

Tại đây bạn sẽ thấy nhiều phiên bản NodeJS dành cho:

- Windows 32-bit và 64-bit,
- Linux,
- macOS.

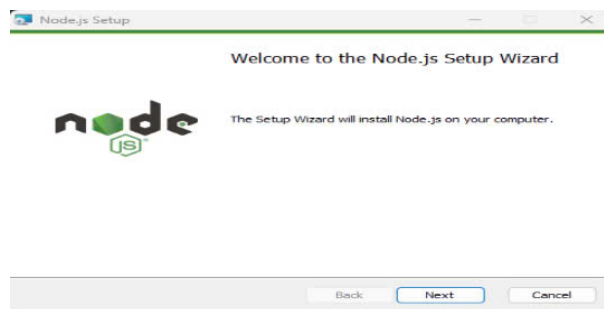
Ở đây, chúng tôi sử dụng Windows 64-bit, vì vậy chọn tải bản Windows Installer (.msi) 64-bit.



Cài đặt NodeJS

Sau khi tải xong file cài đặt, bạn tiến hành các bước sau:

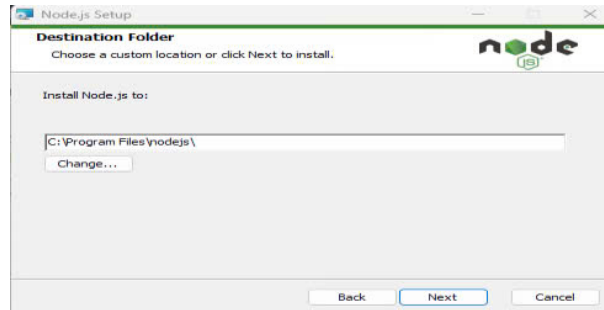
- Mở file .msi vừa tải.
- Nhấn **Next** để tiếp tục.



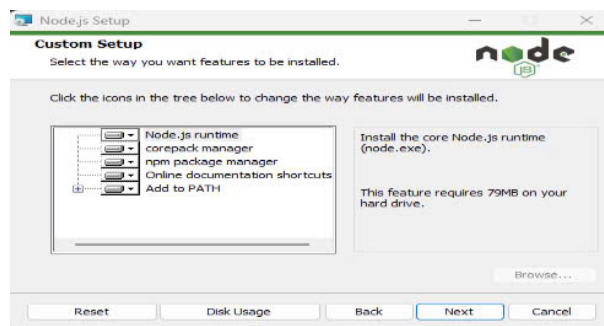
- Tích chọn “*I accept the terms in the License Agreement*” → Nhấn **Next**.



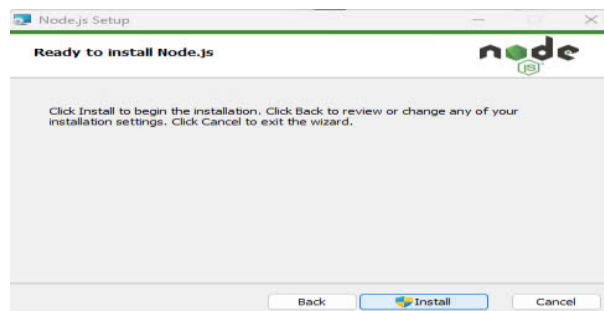
- Chọn vị trí lưu trữ hoặc để mặc định → Nhấn **Next**.



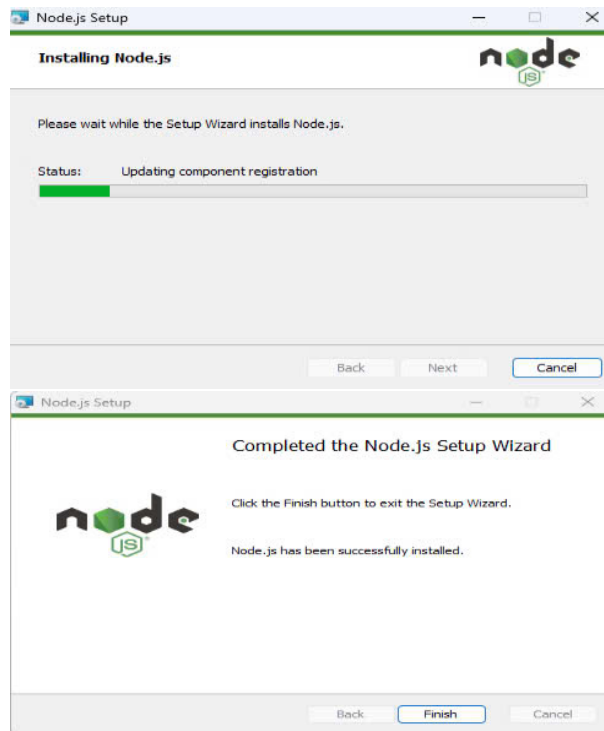
- Chọn các thành phần cài đặt, có thể để mặc định → Nhấn **Next**.



- Nhấn **Install** để bắt đầu quá trình cài đặt.



- Sau khi cài đặt thành công, nhấn **Finish** để hoàn tất.



Để cài đặt Angular CLI, bạn mở cửa sổ terminal (CMD) và chạy lệnh sau:

```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

C:\Users\THAI HOANG>node -v
v22.14.0

C:\Users\THAI HOANG>npm -v
10.9.2

C:\Users\THAI HOANG>
```

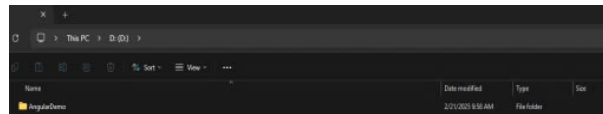
`npm install -g @angular/cli@17`

```
Microsoft Windows [Version 10.0.26100.3194]
(c) Microsoft Corporation. All rights reserved.

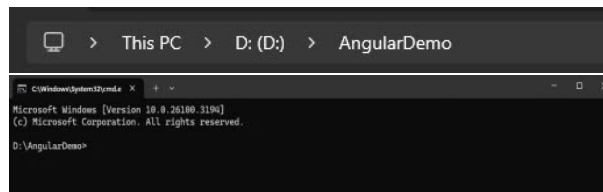
C:\Users\THAI HOANG>npm install -g @angular/cli@17
npm warn deprecated read-package-json@7.0.1: This package is no longer supported. Please use @pnpmcli/package-json instead.
added 244 packages in 29s
43 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New major version of npm available! 10.9.2 -> 11.1.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.1.0
npm notice To update run: npm install -g npm@11.1.0
npm notice
```

Tạo Project Angular mới

- Tạo một thư mục ở ổ đĩa bất kỳ, ví dụ ổ D với tên là AngularDemo.

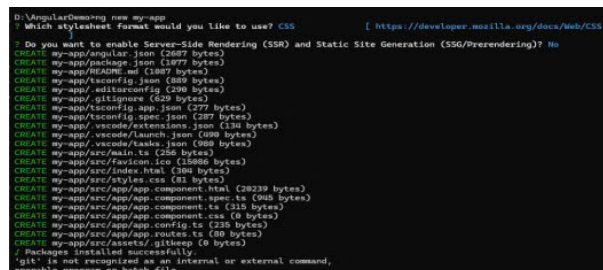


- Truy cập vào thư mục đó, nhấn vào thanh địa chỉ và gõ cmd rồi nhấn Enter.

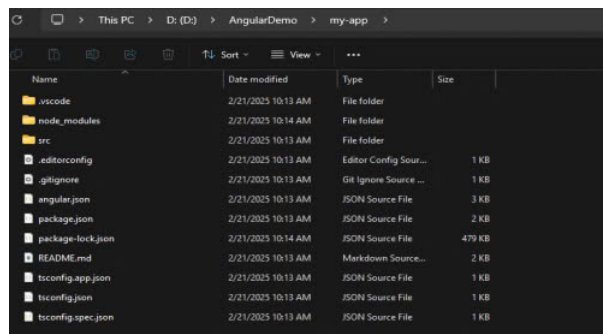


- Tạo project mới với tên là my-app bằng lệnh:

ng new my-app



Sau khi quá trình kết thúc, bạn đã có một project Angular hoàn chỉnh.



Chạy Project Angular

Để chạy ứng dụng, bạn cần thực hiện các bước sau:

```
cd my-app  
ng serve --open
```

Lệnh 'ng serve' sẽ khởi chạy một máy chủ phát triển cục bộ, theo dõi thay đổi và build lại ứng dụng khi có cập nhật. Tùy chọn **-open** hoặc **-o** sẽ tự động mở trình duyệt tại địa chỉ: <http://localhost:4200>

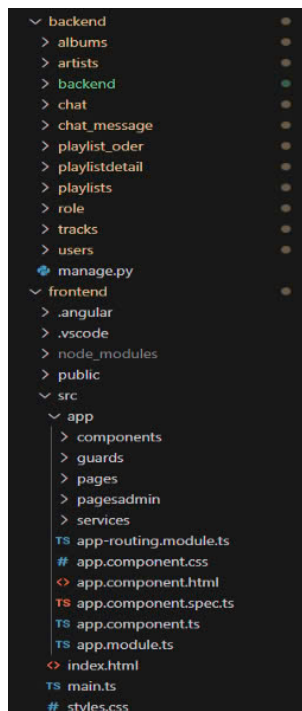
```
Microsoft Windows [Version 10.0.26100.3190]
(c) Microsoft Corporation. All rights reserved.

D:\AngularDemo\my-app>ng serve --open
Initial chunk files      Names      Raw size
polyfills.js            polyfills  88.09 kB
main.js                 main      22.17 kB
styles.css              styles    95 bytes
| Initial total | 110.35 kB

Application bundle generation complete. [2.928 seconds]

Watch mode enabled. Watching for file changes...
-> Local: http://localhost:4200/
-> press h + enter to show help
```

4.2 Cấu trúc thư mục của Project



Thư mục Backend

Đây là phần phía máy chủ (server), sử dụng **Django** – một framework Python mạnh mẽ để xây dựng API, xử lý dữ liệu, xác thực người dùng và giao tiếp với cơ sở dữ liệu. Dựa vào cấu trúc, có thể thấy mỗi thư mục con bên trong là một *Django App*, đại diện cho từng chức năng riêng biệt:



Tên thư mục	Mô tả chức năng
albums/	Quản lý thông tin các album âm nhạc.
artists/	Quản lý thông tin nghệ sĩ.
tracks/	Quản lý các bài hát (track).
users/	Quản lý tài khoản, thông tin người dùng.
role/	Quản lý vai trò, phân quyền người dùng.
chat/, chat_message/	Xử lý tính năng trò chuyện.
playlists/, playlistdetail/, playlist_order/	Quản lý danh sách phát nhạc.
backend/	Chứa cài đặt chính của Django như <code>settings.py</code> , <code>urls.py</code> .
manage.py	Tập tin chính để khởi chạy và quản lý dự án Django.

Mỗi app sẽ có các tệp chuẩn như:

- `models.py`: Định nghĩa bảng dữ liệu (model).
- `views.py`: Viết các xử lý logic.
- `urls.py`: Định nghĩa các API endpoint riêng cho app.
- `serializers.py`: Chuyển đổi dữ liệu từ model sang JSON (nếu dùng Django REST Framework).

Thư mục Frontend

Đây là phần giao diện người dùng (client-side), được xây dựng bằng Angular – một framework hiện đại dùng để tạo các SPA (Single Page Application). Cấu trúc dự án mô tả cho từng chức năng riêng biệt:

Tên thư mục	Mô tả chức năng
components/	Chứa các thành phần giao diện dùng lại nhiều lần như header, navbar...
pages/	Chứa các trang hiển thị chính cho người dùng (user)
pagesadmin/	Các trang dành riêng cho quản trị viên (admin)
guards/	Dùng để bảo vệ route (AuthGuard, AdminGuard)
services/	Gọi API từ backend, chia sẻ logic giữa các component
app-routing.module.ts	Định nghĩa các tuyến đường (routing) chính của toàn bộ ứng dụng Angular
app.module.ts	Khai báo các module con và các thành phần cần thiết cho Angular hoạt động
index.html, main.ts	Điểm khởi tạo ứng dụng Angular
styles.css	Định nghĩa CSS tổng cho toàn bộ ứng dụng

Cấu trúc `src/app` được chia module hợp lý, mỗi phần như `pages`, `pagesadmin` được xây dựng thành module riêng và **lazy loading** qua file `app-routing.module.ts`, giúp tối ưu hiệu suất bằng cách chỉ tải những phần cần thiết khi người dùng truy cập.

Việc sử dụng thư mục **services/** để tập trung toàn bộ các logic giao tiếp với API (backend) giúp mã nguồn rõ ràng, dễ kiểm thử và tuân thủ nguyên tắc **Separation of Concerns (SoC)**.

Ngoài ra, các component tái sử dụng nằm trong thư mục **components/**, chẳng hạn như **header**, **navbar**, **footer**,... giúp đảm bảo tính đồng nhất giao diện và tiết kiệm công sức khi phát triển hoặc thay đổi layout sau này.

4.3 Routing và phân chia các module

Dự án được tổ chức theo kiến trúc module hóa, tức là chia nhỏ hệ thống thành các phần riêng biệt tương ứng với từng chức năng cụ thể. Việc phân chia này được áp dụng ở cả hai phía: **Backend (Django)** và **Frontend (Angular)**, nhằm tăng tính mở rộng, dễ bảo trì và giúp các thành viên trong nhóm có thể phát triển song song mà không ảnh hưởng đến nhau.

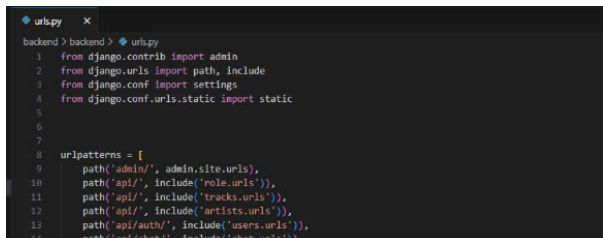
Backend (Django)

Phía backend được xây dựng bằng framework Django, mỗi chức năng được tổ chức dưới dạng một app riêng biệt. Ví dụ: *albums*, *artists*, *tracks*, *users*, *chat*,... Mỗi app đóng vai trò như một module độc lập, đảm nhiệm một phần nghiệp vụ của hệ thống như quản lý bài hát, người dùng, hoặc trò chuyện.

Mỗi app sẽ có các file riêng như:

- **models.py**: định nghĩa bảng dữ liệu
- **views.py**: xử lý logic nghiệp vụ
- **urls.py**: định tuyến API cho từng app

Tất cả các tuyến đường (route) của các app sẽ được gom lại trong file **backend/urls.py** chính bằng cách sử dụng hàm **include()**, giúp dự án dễ tổ chức và mở rộng.



```
1 from django.contrib import admin
2 from django.urls import path, include
3 from django.conf import settings
4 from django.conf.urls.static import static
5
6
7
8 urlpatterns = [
9     path('admin/', admin.site.urls),
10    path('api/', include('role.urls')),
11    path('api/', include('tracks.urls')),
12    path('api/', include('artists.urls')),
13    path('api/auth/', include('users.urls')),
14    path('api/chat/', include('chat.urls')),
15]
```

Nhờ đó, khi frontend gọi các API như **GET /api/users/**, Django sẽ biết được phải định tuyến đến app **users**.

Frontend (Angular)

Phía frontend được xây dựng bằng Angular, tuân thủ mô hình component-based và chia module rõ ràng theo từng nhóm chức năng.

Cấu trúc được phân chia thành các module chính như:

- **pages/**: chứa các trang dành cho người dùng thông thường
- **pagesadmin/**: chứa các trang dành riêng cho quản trị viên
- **components/**: chứa các thành phần giao diện có thể tái sử dụng (header, navbar,...)

- **services/**: xử lý logic và gọi API từ backend
- **guards/**: kiểm soát quyền truy cập router (ví dụ như kiểm tra đã đăng nhập, hoặc có phải admin không)

Tuyến đường được định nghĩa trong file `app-routing.module.ts`, sử dụng cơ chế **lazy loading** để tối ưu hiệu năng. Chỉ khi người dùng truy cập một module nào đó thì Angular mới tải module đó vào bộ nhớ.

```
22 const routes: Routes = [  
23   { path: '', redirectTo: 'home', pathMatch: 'full' },  
24   { path: 'home', component: HomeComponent },  
25   { path: 'admin', component: AdminComponent,  
26     children: [  
27       { path: 'users', component: UsersComponent },  
28       { path: 'user-group', component: UsergroupComponent },  
29       { path: 'artist', component: ArtistComponent },  
30       { path: 'category', component: CategoryComponent },  
31       { path: 'album', component: AlbumComponent },  
32       { path: 'song', component: SongsListComponent },  
33     ] },  
34   { path: 'login', component: LoginComponent },  
35   // { path: 'song/:song_id', component: SongComponent },  
36   { path: 'signup', component: SignupComponent },  
37   { path: 'playlist', component: PlaylistComponent },  
38   { path: 'register', component: SignupComponent },  
39   { path: 'chat/:chatId', component: ChatComponent },  
40   { path: 'search', component: SearchUsersComponent },  
41 ]
```

Một số lợi ích:

- Tăng tính tái sử dụng mã nguồn
- Dễ dàng bảo trì và mở rộng
- Phát triển song song hiệu quả giữa các thành viên
- Tối ưu hiệu năng (nhờ lazy loading trong Angular)
- Quản lý quyền truy cập linh hoạt hơn (dùng guard)

5 NHIỆM VỤ VÀ VAI TRÒ CỦA THÀNH VIÊN

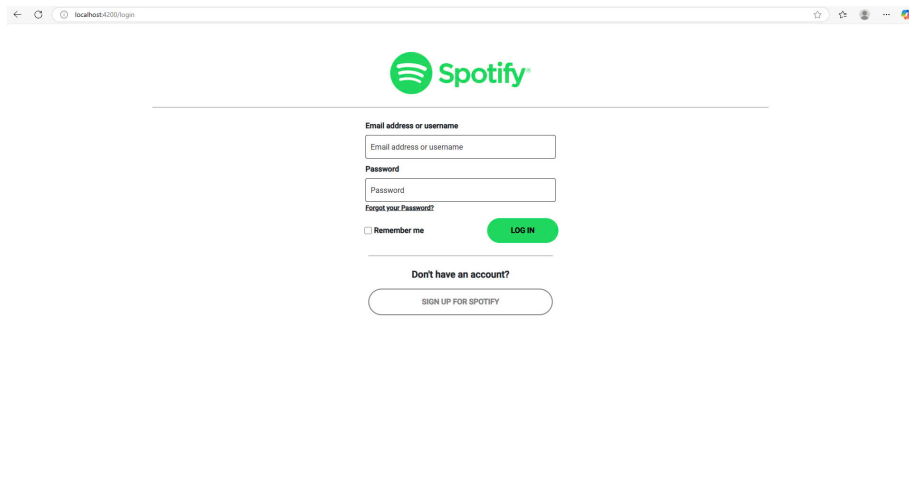
Họ và tên	MSSV	Nhiệm vụ
Nguyễn Đức Huy	3121410230	Frontend+Backend(user : hiển thị ds nhạc, playlist, thêm nhạc vào playlist, login logout, chat)
Trần Thái Hoàng	3121410214	UI,Frontend,Backend(admin)
Trần Tấn Phát	3121410380	Backend(albums, playlist, playlist detail, playlist oder)
Phan Thanh Hữu	3121410246	Frontend,Backend(Search,Chat)

Bảng 1: Phân công nhiệm vụ các thành viên



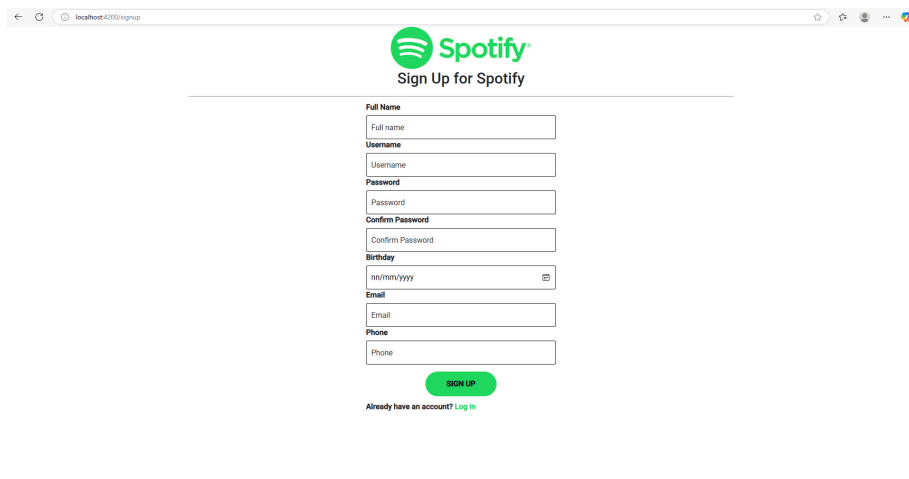
6 KẾT QUẢ ĐẠT ĐƯỢC

6.1 Màn hình login



The screenshot shows the Spotify login page in a web browser. The browser's address bar displays 'localhost:4200/login'. The Spotify logo is centered at the top. Below it, the login form includes a text input for 'Email address or username', another for 'Password', and a 'Forgot your Password?' link. There is a 'Remember me' checkbox and a green 'LOG IN' button. At the bottom, a link for 'SIGN UP FOR SPOTIFY' is provided for users who do not have an account.

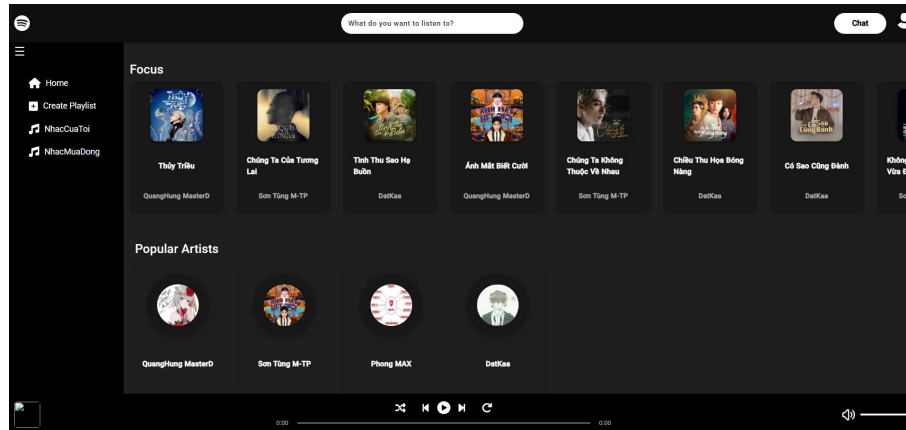
6.2 Màn hình Sign Up



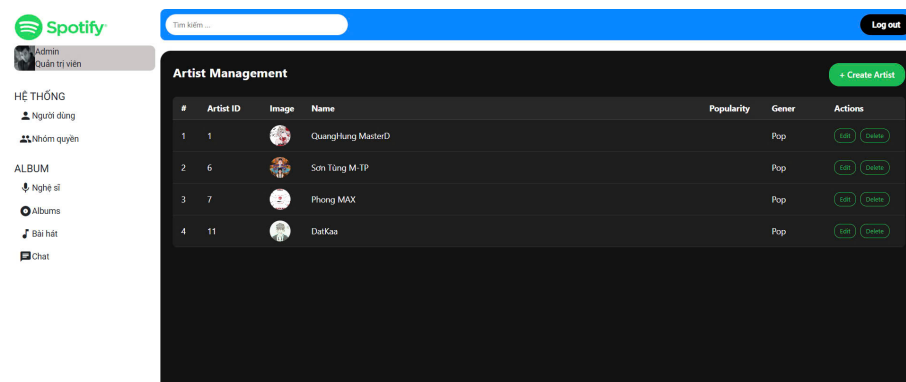
The screenshot shows the Spotify sign-up page in a web browser. The browser's address bar displays 'localhost:4200/signup'. The Spotify logo is centered at the top, with the text 'Sign Up for Spotify' below it. The sign-up form contains several fields: 'Full Name' (with a sub-field for 'Full name'), 'Username', 'Password', 'Confirm Password', 'Birthday' (with a date picker), 'Email', and 'Phone'. A green 'SIGN UP' button is located at the bottom of the form. Below the button, a link for 'Already have an account? Log in' is visible.



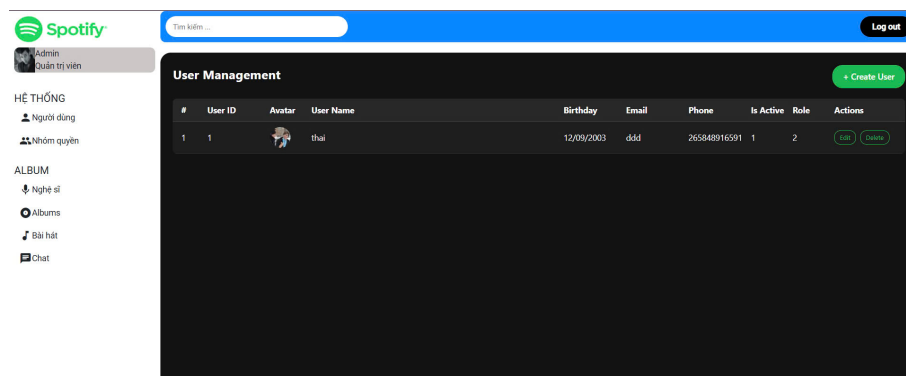
6.3 Màn hình HomePage



6.4 Màn hình Artist Management

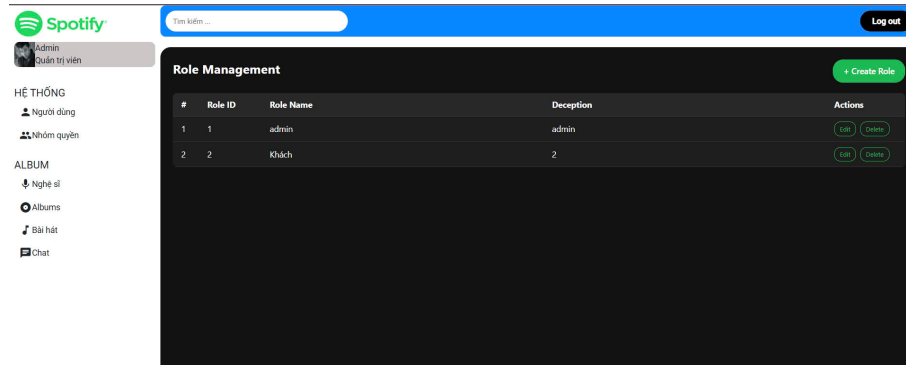


6.5 Màn hình User Management

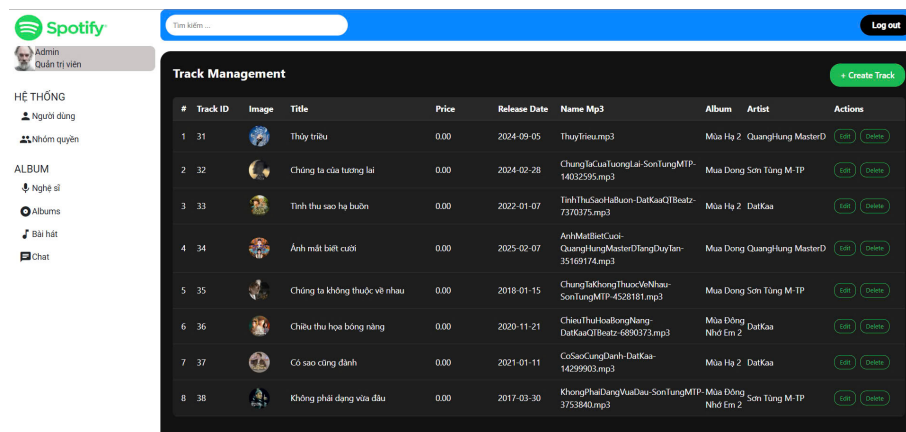




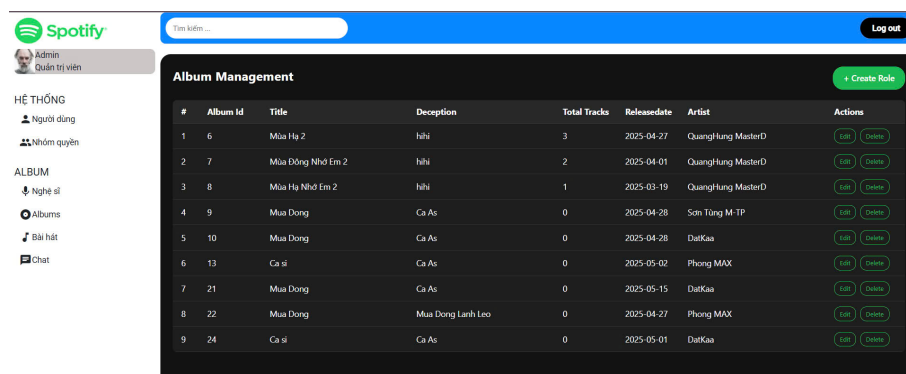
6.6 Màn hình Role Management



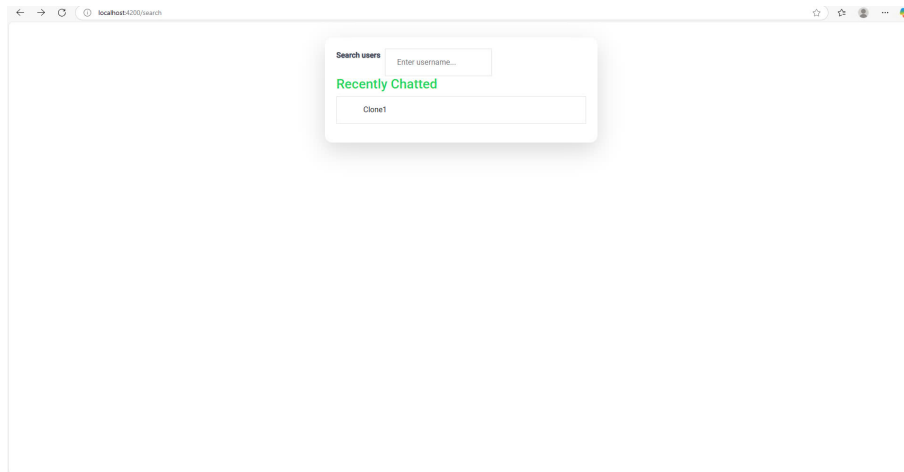
6.7 Màn hình Track Management



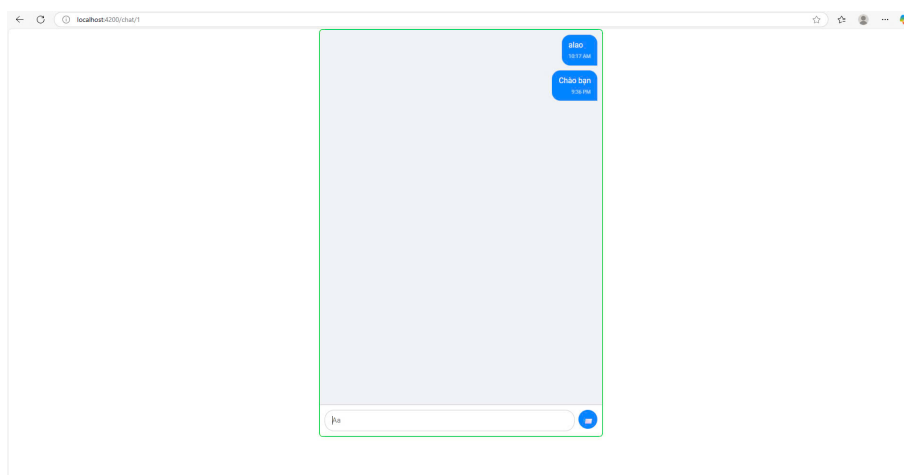
6.8 Màn hình Album Management



6.9 Màn hình Search User



6.10 Màn hình Chat



7 ĐÁNH GIÁ VÀ ĐỊNH HƯỚNG PHÁT TRIỂN

7.1 Ưu điểm và nhược điểm

Ưu điểm:

- Kiến trúc rõ ràng, dễ mở rộng: Hệ thống được xây dựng theo mô hình client-server với frontend và backend tách biệt, tạo điều kiện cho việc nâng cấp, sửa đổi và bảo trì từng phần mà không ảnh hưởng đến toàn bộ ứng dụng. Nhờ cấu trúc module hóa ở cả hai phía, các chức năng như quản lý người dùng, album, chat,... đều được chia nhỏ theo từng nhóm nghiệp vụ cụ thể.
- Công nghệ hiện đại, phổ biến: Ứng dụng sử dụng Angular cho frontend – một framework mạnh mẽ với khả năng hỗ trợ routing, lazy loading, component-based và reactive program-

mình giúp cải thiện hiệu năng và trải nghiệm người dùng. Phía backend sử dụng Django Rest Framework (DRF) giúp dễ dàng tạo và quản lý API RESTful theo chuẩn hiện đại, hỗ trợ xác thực, phân quyền và xử lý dữ liệu mạnh mẽ.

- Tổ chức mã nguồn tốt: Các thư mục và module được phân chia rõ ràng, từ giao diện người dùng (pages/), quản trị (pagesadmin/), cho đến các thành phần dùng chung (components/), giúp quá trình phát triển và gỡ lỗi dễ dàng hơn. Backend cũng chia app rõ ràng theo chức năng (users, albums, chat,...).
- Khả năng tái sử dụng cao: Với cách tổ chức thành phần theo dạng component và service, phần frontend có thể dễ dàng mở rộng tính năng hoặc tái sử dụng ở các phần khác trong hệ thống.

Nhược điểm:

- Giao diện chưa thực sự thân thiện: Mặc dù chức năng đầy đủ nhưng thiết kế giao diện còn đơn giản, màu sắc và bố cục chưa trực quan, dễ nhìn đối với người dùng phổ thông, đặc biệt là trên thiết bị di động.
- Thiếu kiểm thử tự động: Ứng dụng hiện tại chủ yếu được kiểm tra thủ công, chưa có hệ thống kiểm thử tự động (unit test, e2e test), nên khó đảm bảo chất lượng khi cập nhật hoặc thêm chức năng mới.
- Khả năng mở rộng còn giới hạn: Chưa có hệ thống xử lý tải cao hoặc cơ chế cache để tối ưu truy xuất dữ liệu khi có nhiều người dùng đồng thời. Bên cạnh đó, phần backend vẫn chạy dạng đơn khối (monolithic), khó chia tách nếu hệ thống phát triển quy mô lớn hơn.
- Chưa tích hợp lưu trữ thực tế: File ảnh, nhạc hoặc dữ liệu đa phương tiện vẫn lưu tạm thời, chưa được kết nối với dịch vụ lưu trữ đám mây (cloud storage), dẫn đến khó triển khai thực tế ở môi trường production.

7.2 Hướng phát triển

Trong tương lai, hệ thống có thể được nâng cấp và mở rộng theo các hướng sau:

- Nâng cấp giao diện người dùng: Thiết kế lại UI theo hướng hiện đại, chuẩn responsive để tương thích tốt trên cả máy tính và thiết bị di động. Có thể sử dụng thư viện UI mạnh như Angular Material hoặc Tailwind CSS kết hợp.
- Mở rộng chức năng người dùng: Cho phép người dùng đăng tải bài hát, tạo playlist cá nhân, chia sẻ danh sách phát,... từ đó tăng trải nghiệm và mức độ tương tác của người dùng với hệ thống.
- Phân quyền nâng cao: Bổ sung nhiều cấp độ quyền (admin, editor, moderator, viewer) với giao diện quản trị tương ứng, tăng khả năng kiểm soát và bảo mật nội dung.
- Tích hợp streaming media: Cho phép người dùng nghe nhạc trực tuyến thay vì chỉ tải file, đồng thời sử dụng kỹ thuật nén và buffer để tối ưu việc truyền tải âm thanh.
- Xây dựng hệ thống kiểm thử: Bổ sung unit test và integration test cho cả frontend (Jasmine, Karma) và backend (pytest, Django TestCase) để đảm bảo hệ thống hoạt động ổn định và phát hiện lỗi sớm hơn.



- Triển khai thực tế: Đưa hệ thống lên các nền tảng cloud như AWS, Firebase, hoặc Vercel (cho frontend), sử dụng Docker để đóng gói và triển khai backend nhanh chóng. Kết hợp CI/CD để tự động hóa quá trình cập nhật.
- Phát triển mobile app: Sử dụng Flutter hoặc React Native để xây dựng ứng dụng di động có thể đồng bộ dữ liệu với backend hiện tại, nhằm mở rộng phạm vi sử dụng và tiếp cận người dùng tốt hơn.
- Tối ưu backend: Nếu hệ thống phát triển lớn hơn, có thể chuyển backend sang kiến trúc microservices và tích hợp công nghệ như Redis (cache), Celery (nền tảng xử lý nền), Elasticsearch (tìm kiếm nâng cao)

8 KẾT LUẬN

Sau quá trình tìm hiểu và thực hiện đề tài, nhóm đã hoàn thành hệ thống ứng dụng quản lý và chia sẻ album âm nhạc với giao diện hiện đại, thân thiện người dùng, cùng với hệ thống backend mạnh mẽ, hỗ trợ đầy đủ các chức năng cần thiết như quản lý người dùng, album, và trò chuyện. Việc áp dụng mô hình tách biệt frontend-backend sử dụng **Angular** và **Django REST Framework** không chỉ giúp hệ thống dễ phát triển mà còn tạo tiền đề mở rộng cho tương lai.

Trong quá trình làm việc, nhóm đã học hỏi được nhiều kỹ năng quan trọng như:

- Cách xây dựng và tổ chức dự án phần mềm một cách logic, rõ ràng.
- Áp dụng kiến thức lập trình web, thiết kế RESTful API, giao tiếp giữa frontend và backend.
- Quản lý mã nguồn và phối hợp làm việc nhóm hiệu quả thông qua **Git** và **GitHub**.
- Rèn luyện kỹ năng tự học, tự nghiên cứu công nghệ mới để giải quyết vấn đề thực tế.

Tuy vẫn còn một số điểm hạn chế như giao diện chưa hoàn toàn tối ưu hoặc chưa có kiểm thử tự động, nhưng đây cũng là những cơ sở quan trọng để nhóm nhìn nhận và phát triển hệ thống tốt hơn trong tương lai.

Qua quá trình thực hiện đề tài này, nhóm nhận thấy rằng việc kết hợp giữa lý thuyết học thuật và thực tiễn triển khai đóng vai trò rất quan trọng, giúp sinh viên không chỉ hiểu sâu về công nghệ mà còn biết cách vận dụng vào thực tế để tạo ra sản phẩm có giá trị.

9 TÀI LIỆU THAM KHẢO

<https://devwork.vn/tai-lieu-mien-phi/189/tai-lieu-huong-dan-lap-trinh-django-danh-cho-lap-trinh-vien>

<https://playzone.edu.vn/huong-dan-angular/>