

ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN

— * —



PHÁT TRIỂN HỆ ĐIỀU HÀNH MÃ NGUỒN MỞ

PHÁT TRIỂN PHẦN MỀM SPOTIFY CLONE

GVHD: Từ Lăng Phiêu

Sinh viên:

3121410497 - Huỳnh Quốc Tiến

quoctien01062003@gmail.com

TP. HỒ CHÍ MINH, 04/2025

Mục lục

1	Mở đầu	1
1.1	Giới thiệu đề tài	1
1.2	Mục tiêu đề tài	1
1.3	Phạm vi của dự án	2
1.4	Công nghệ sử dụng	2
2	Nội dung báo cáo	3
2.1	Cơ sở lý thuyết	3
2.2	Front-end	5
2.3	Back-end	5
2.4	Databse	5
3	Hiện thực và Thiết kế giao diện	15
3.1	Thiết kế giao diện	15
3.2	Các chức năng	20
3.2.1	Chức năng phát nhạc	20
3.2.2	Chức năng phát video âm nhạc	21
3.2.3	Chức năng tải video âm nhạc	22
3.2.4	Chức năng User tạo album, bài hát yêu thích	23
3.2.5	Trang Admin	23
3.2.6	Tùy chọn) Chức năng Chat AI	24
3.3	Cấu trúc mã nguồn	24
4	Cài đặt và Hướng dẫn sử dụng	26
4.1	Môi trường chạy ứng dụng	26
4.2	Hướng dẫn cài đặt	26
4.3	Hướng dẫn sử dụng	27
4.4	Các chú ý khi sử dụng phần mềm	28
4.5	Kết luận và Hướng phát triển	28

Chương 1

Mở đầu

1.1 Giới thiệu đề tài

Spotify là một trong những nền tảng phát nhạc trực tuyến phổ biến nhất hiện nay, cung cấp kho nhạc khổng lồ cùng nhiều tính năng hấp dẫn như phát nhạc theo yêu cầu, tạo playlist, đề xuất bài hát dựa trên sở thích người dùng. Đề tài "Phát triển phần mềm Spotify Clone" yêu cầu xây dựng một ứng dụng tương tự với các chức năng cơ bản của Spotify, sử dụng công nghệ web hiện đại và triển khai trên môi trường mã nguồn mở Linux.

1.2 Mục tiêu đề tài

Mục tiêu chính:

- Thiết kế giao diện web cho phần mềm Spotify Clone một cách trực quan và thân thiện với người dùng
- Lập trình các chức năng cơ bản của ứng dụng Spotify Clone bằng ngôn ngữ Python (sử dụng framework Django cho back-end và React/Angular cho front-end).
- Xây dựng một hệ thống cơ sở dữ liệu hiệu quả để quản lý dữ liệu của ứng dụng (người dùng, bài hát, video, album, danh sách yêu thích, v.v.).
- Triển khai ứng dụng trên môi trường mã nguồn mở Linux.

Mục tiêu học tập:

- Nâng cao kỹ năng lập trình Python và các framework liên quan (Django, React/Angular)
- Hiểu rõ hơn về quy trình phát triển ứng dụng web full-stack.
- Rèn luyện kỹ năng thiết kế giao diện người dùng (UI/UX).
- Tìm hiểu và ứng dụng các công nghệ mới trong lĩnh vực phát triển web

1.3 Phạm vi của dự án

Các tính năng sẽ được triển khai (theo yêu cầu đề bài)

- Phát nhạc, video, tải video âm nhạc.
- Người dùng tạo album, người dùng tạo danh sách bài hát yêu thích, Chat tích hợp.

Phương pháp tiếp cận

Hiểu kỹ về cách thức hoạt động, giao diện và các tính năng của Spotify. Phân tích yêu cầu của đề bài và xác định các bước cần thực hiện và lên kế hoạch thiết kế giao diện người dùng (wireframe, mock-up) trước khi bắt đầu lập trình. Thiết kế mô hình cơ sở dữ liệu. Chia dự án thành các giai đoạn nhỏ, tập trung hoàn thành từng chức năng một cách tuần tự

1.4 Công nghệ sử dụng

- **Ngôn ngữ lập trình:** Python 3.
- **Front-end framework:** Reactjs
- **Back-end framework:** Django
- **Database:** SQLite3
- **Công cụ sử dụng:** Pycharm, Visual Code
 - **Ubuntu Server 22.04 LTS:** Hệ điều hành chính thức cho EC2 instances
 - **Cấu hình tối thiểu:** t2.micro (1 vCPU, 1GB RAM) - Miễn phí tier
- **Dịch vụ AWS: EC2:** Máy chủ ảo chạy ứng dụng

Chương 2

Nội dung báo cáo

2.1 Cơ sở lý thuyết

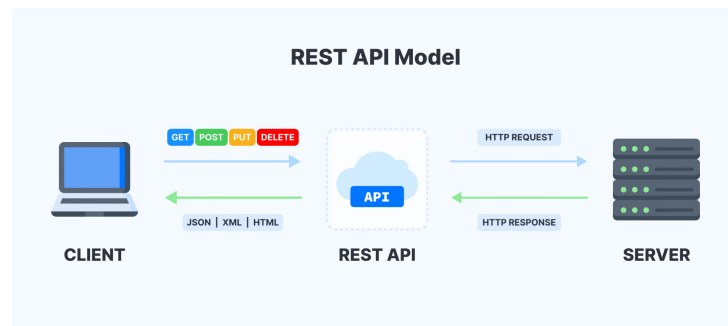
Tổng quan về kiến trúc ứng dụng web (Client-Server)

Kiến trúc Client-Server là mô hình phổ biến trong phát triển ứng dụng web, trong đó công việc được phân chia giữa hai thành phần chính. Về khái niệm cơ bản, một hoặc nhiều client (thường là ứng dụng web trên trình duyệt của người dùng) gửi yêu cầu (request) đến server (máy chủ chứa ứng dụng và dữ liệu). Server sau đó xử lý yêu cầu này và trả về phản hồi (response) cho client. Mô hình này cho phép phân tán công việc xử lý, tối ưu hóa hiệu suất và tăng cường bảo mật hệ thống.

Client, thường là trình duyệt web, đóng vai trò quan trọng trong việc hiển thị giao diện người dùng (UI) dựa trên HTML, CSS và JavaScript. Client xử lý các tương tác của người dùng như nhấp chuột, nhập liệu, kéo thả và gửi yêu cầu đến server thông qua các phương thức HTTP như GET, POST, PUT, DELETE. Ngoài ra, client có thể thực hiện một số xử lý dữ liệu đơn giản phía người dùng để giảm tải cho server và sử dụng cookie, localStorage để lưu trữ thông tin phiên làm việc tạm thời.

Server đảm nhận các nhiệm vụ quan trọng như xử lý logic nghiệp vụ phức tạp và thực hiện các tính toán chuyên sâu. Server quản lý dữ liệu bằng cách kết nối với cơ sở dữ liệu để lưu trữ, truy xuất và cập nhật thông tin. Nó cũng xử lý xác thực và phân quyền để kiểm soát quyền truy cập vào tài nguyên, cung cấp API để client có thể tương tác với hệ thống, xử lý nhiều yêu cầu đồng thời từ các client khác nhau, và đảm bảo bảo mật cho toàn bộ hệ thống. Mô hình này cho phép dễ dàng mở rộng bằng cách thêm nhiều client hoặc server, tạo nên kiến trúc linh hoạt và mạnh mẽ cho các ứng dụng web hiện đại.

Luồng hoạt động cơ bản



Hình 2.1: Luồng hoạt động của web Client-Server

Khi người dùng thực hiện hành động, trình duyệt chuyển đổi hành động đó thành yêu cầu HTTP (HTTP Request) và gửi đến địa chỉ server tương ứng. Yêu cầu này bao gồm phương thức HTTP (như GET, POST, PUT, DELETE), URL đích, các headers chứa thông tin về trình duyệt, cookie, định dạng dữ liệu, và body chứa dữ liệu đi kèm nếu có

Khi server nhận được yêu cầu HTTP, nó tiến hành phân tích nội dung và thực hiện các hoạt động cần thiết như xác thực người dùng (nếu cần), truy xuất hoặc cập nhật dữ liệu trong cơ sở dữ liệu, thực thi logic nghiệp vụ, và chuẩn bị dữ liệu phản hồi

Trình duyệt nhận phản hồi từ server, xử lý và hiển thị nội dung cho người dùng bằng cách phân tích và vẽ trang HTML, thực thi mã JavaScript, áp dụng CSS, và hiển thị trang web hoặc thông báo lỗi

Ứng dụng trong dự án

Trong dự án Spotify Clone, kiến trúc Client-Server được áp dụng rõ ràng giữa Front-end và Back-end. Front-end, được xây dựng bằng ReactJS (hoặc Angular), đóng vai trò là client, chịu trách nhiệm hiển thị giao diện người dùng, xử lý các tương tác như tìm kiếm bài hát, điều khiển phát nhạc, thêm bài hát vào danh sách yêu thích và gửi các yêu cầu tới server. Khi người dùng thực hiện các thao tác trên ứng dụng, Front-end sẽ gửi các HTTP request (API request) tới Back-end.

Back-end được xây dựng bằng Django, đóng vai trò là server, chịu trách nhiệm tiếp nhận và xử lý các yêu cầu từ Front-end. Server sẽ thực hiện các tác vụ như xác thực người dùng, truy xuất hoặc cập nhật cơ sở dữ liệu, và trả về dữ liệu dạng JSON hoặc file media cho Front-end. Cơ sở dữ liệu sẽ lưu trữ các thông tin như người dùng, bài hát, album, playlist, lịch sử phát nhạc,...

Luồng hoạt động tổng thể của hệ thống là: người dùng thao tác trên giao diện → Front-end gửi yêu cầu API → Back-end xử lý yêu cầu và tương tác với cơ sở dữ liệu → dữ liệu được trả về Front-end để cập nhật giao diện. Kiến trúc Client-Server này giúp tách biệt rõ ràng giữa giao diện người dùng và logic xử lý phía server, đồng thời tăng tính mở rộng, dễ bảo trì cho toàn bộ hệ thống.

2.2 Front-end

React là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook, dùng để xây dựng giao diện người dùng (UI) hoặc các thành phần UI cho các ứng dụng web. React nổi bật với mô hình component-based (dựa trên các thành phần độc lập) và cách tiếp cận declarative (khai báo) giúp việc xây dựng giao diện trở nên trực quan, dễ quản lý và dễ mở rộng. Sử dụng React cho phần Front-end của dự án vì React có các ưu điểm phù hợp với yêu cầu của dự án, như khả năng phát triển giao diện nhanh chóng, quản lý UI phức tạp dễ dàng thông qua component-based architecture, hiệu suất cao với Virtual DOM, và tính linh hoạt trong việc xây dựng ứng dụng web đơn trang (SPA)

2.3 Back-end

Django là một framework web cấp cao được viết bằng ngôn ngữ Python, miễn phí và mã nguồn mở. Django tuân theo kiến trúc MTV (Model-Template-View), giúp tổ chức mã nguồn rõ ràng và dễ mở rộng. Framework này nổi bật với triết lý "batteries included", nghĩa là Django cung cấp sẵn rất nhiều tính năng cần thiết cho việc phát triển web như hệ thống xác thực người dùng, quản lý cơ sở dữ liệu ORM, quản lý admin, bảo mật, định tuyến URL, và nhiều công cụ hỗ trợ khác. Nhờ vậy, lập trình viên có thể xây dựng các ứng dụng web mạnh mẽ và bảo mật một cách nhanh chóng mà không cần phải cài đặt hoặc viết lại quá nhiều thứ từ đầu

Django REST Framework (DRF) là một bộ công cụ mạnh mẽ và linh hoạt dành cho Django, được sử dụng để xây dựng các Web API dễ dàng và hiệu quả. DRF mở rộng sức mạnh của Django, giúp việc phát triển các dịch vụ API trở nên nhanh chóng, có cấu trúc rõ ràng và dễ bảo trì. Đây là lựa chọn phổ biến cho các dự án cần giao tiếp giữa Front-end và Back-end thông qua giao thức HTTP

Các khái niệm liên quan

- **Models** : Định nghĩa các trường, kiểu dữ liệu, khóa chính, khóa ngoại và các ràng buộc
- **Views** : Mô tả các loại Views trong Django (function-based views và class-based views) và cách chúng xử lý request và trả về response
- **URLs** : Mô tả cách định nghĩa các URL patterns trong urls.py để ánh xạ các đường dẫn đến các Views tương ứng
- **Migrations** : Sử dụng để quản lý các thay đổi trong models và áp dụng chúng vào cơ sở dữ liệu
- **Serializers (DRF)** : Trong Django REST Framework, Serializers có vai trò chuyển đổi dữ liệu giữa object (ví dụ model instances) và các định dạng như JSON, và ngược lại. Serializers hoạt động tương tự như Django Forms nhưng dùng cho API.

2.4 Database

SQLite3 là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) nhẹ, mã nguồn mở, tuân thủ tiêu chuẩn SQL và được tích hợp sẵn trong Python. Khác với các hệ quản trị cơ sở dữ liệu truyền thống như PostgreSQL hay MySQL, SQLite không yêu cầu server riêng biệt để vận hành

Lí do lựa chọn

Dễ dàng triển khai không cần cài đặt hoặc cấu hình máy chủ cơ sở dữ liệu riêng, phù hợp cho giai đoạn phát triển và thử nghiệm. Tích hợp sẵn Python hỗ trợ SQLite3 mặc định, giúp tiết kiệm thời gian thiết lập môi trường và đáp ứng tốt cho dự án quy mô nhỏ: Với một ứng dụng như Spotify Clone ở mức prototype, SQLite3 có thể đáp ứng tốt về hiệu suất và khả năng lưu trữ. Phù hợp khi triển khai ban đầu trên môi trường Linux đơn giản

Bảng	Thông tin
Genre	id (PK), name
Artist	id (PK), name, bio, image
ArtistGenres	artist_id (FK), genre_id (FK) (ManyToMany)
Album	id (PK), title, artist_id (FK), release_date, cover_image, genre_id (FK), created_by_id (FK)
Song	id (PK), title, album_id (FK), duration, audio_file, image, song_number, plays, lyrics, created_at
SongArtists	song_id (FK), artist_id (FK) (ManyToMany)
Playlist	id (PK), title, user_id (FK), is_public, cover_image, description, created_at
PlaylistSongs	playlist_id (FK), song_id (FK) (ManyToMany)
UserProfile	id (PK), user_id (FK - unique), profile_picture
FavoriteSongs	userprofile_id (FK), song_id (FK) (ManyToMany)
FavoriteAlbums	userprofile_id (FK), album_id (FK) (ManyToMany)
ChatHistory	id (PK), user_id (FK), message, response, timestamp, read
PasswordResetOTP	id (PK), user_id (FK), otp, created_at, expires_at, is_used
Video	id (PK), title, description, video_file, thumbnail, duration, created_at, views, genre_id (FK), album_id (FK)
VideoArtists	video_id (FK), artist_id (FK) (ManyToMany)
UserAlbum	id (PK), user_id (FK), title, description, cover_image, created_at
UserAlbumSongs	useralbum_id (FK), song_id (FK) (ManyToMany)

Bảng 2.1: Thiết kế bảng dữ liệu hệ thống

Frontend (ReactJS + Vite)

- **Build tool:** Vite
- **UI Framework:** Ant Design (antd) + Icon libraries:
 - @ant-design/icons
 - @fortawesome/fontawesome-free
 - @heroicons/react
 - lucide-react
 - react-icons
- **State management:** Redux Toolkit (@reduxjs/toolkit, react-redux)
- **Routing:** react-router-dom
- **HTTP Client:** axios

-
- **Real-time communication:** `socket.io-client`
 - **Utilities:**
 - `file-saver`, `jszip` (lưu file, tạo file nén)
 - `prop-types` (kiểm tra kiểu dữ liệu props)
 - **Dev tools:**
 - `eslint`, `eslint-plugin-react-hooks`, `eslint-plugin-react-refresh`
 - `@vitejs/plugin-react`
 - `@types/react`, `@types/react-dom`

Backend (Django + Channels + MongoDB)

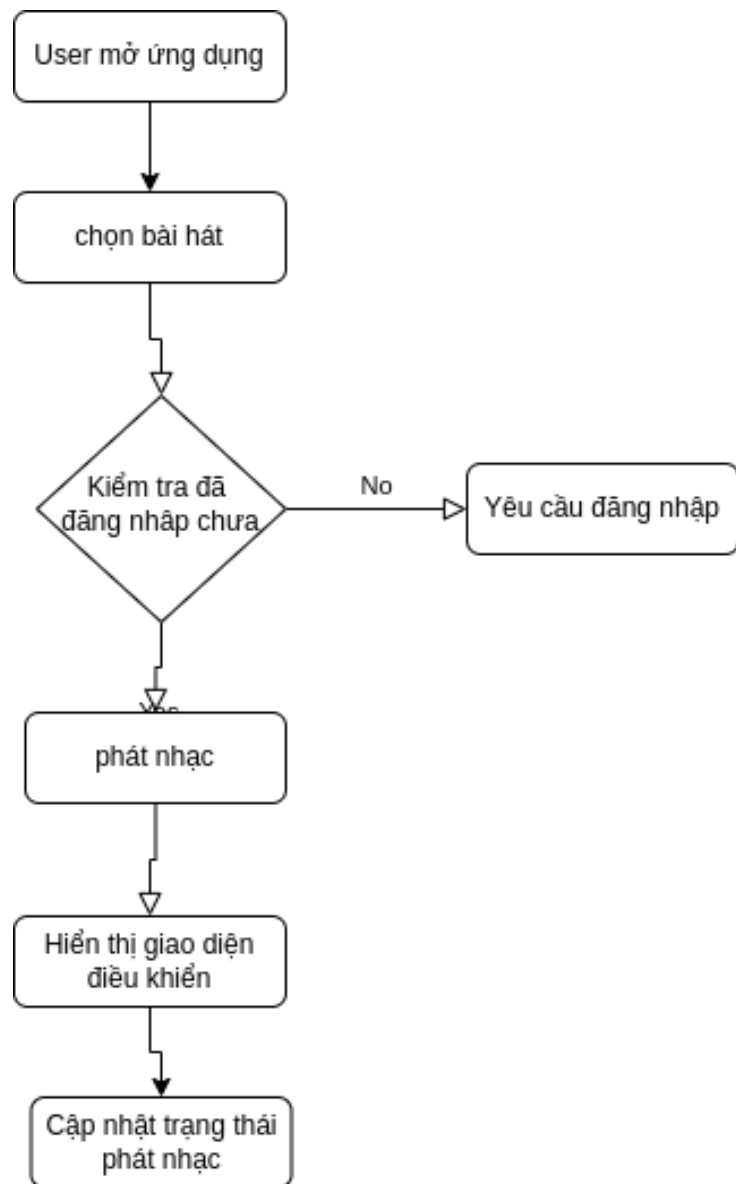
- **Web framework:** Django 5.2
- **API framework:** Django REST Framework (`djangorestframework`)
- **Authentication:** JWT (`djangorestframework-simplejwt`)
- **Real-time:** Channels (WebSocket support)
- **MongoDB connector:**
 - `django`
 - `django-mongodb-backend`
 - `pymongo`
- **Socket communication:**
 - `python-socketio`
 - `python-engineio`
- **ASGI server:** Uvicorn
- **Testing:** `pytest`, `pytest-cov`
- **Environment management:** `python-dotenv`
- **Utilities:**
 - HTTP clients: `requests`, `httpx`
 - Metadata xử lý audio: `mutagen`
 - AI/NLP: `transformers`, `huggingface-hub`, `tokenizers`, `safetensors`
 - Xử lý ảnh: `Pillow`
 - SQL parsing: `sqlparse`
 - Giao tiếp microservices: `protobuf`, `grpcio`
 - I/O acceleration và hot-reload: `uvloop`, `watchfiles`

Cấu trúc hệ thống

- **Frontend:** ReactJS kết nối WebSocket, quản lý trạng thái bằng Redux.
- **Backend:** Django + Channels, cơ sở dữ liệu MongoDB (document database).
- **Authentication:** JWT Tokens.
- **Real-time:** WebSocket + Socket.IO.
- **Extra features:** Tích hợp AI/NLP, gợi ý thông minh.
- **ASGI server:** Uvicorn.

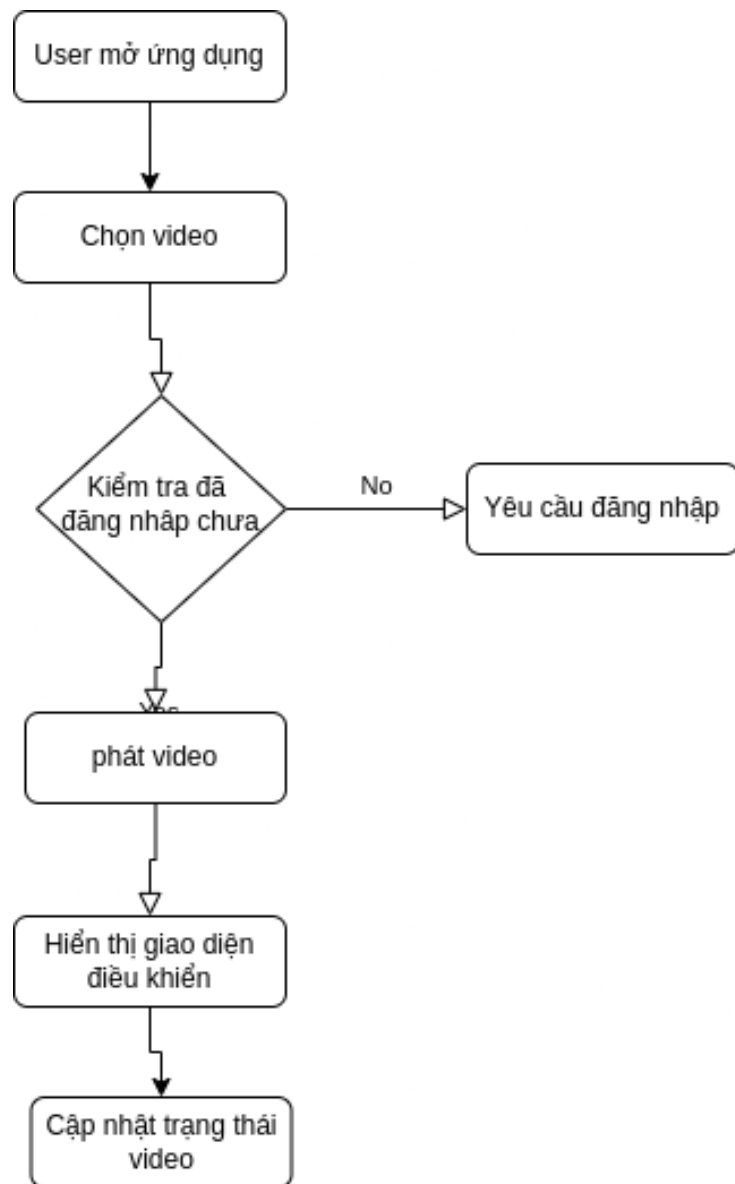
Các Flowchart cho các chức năng

Chức năng phát nhạc



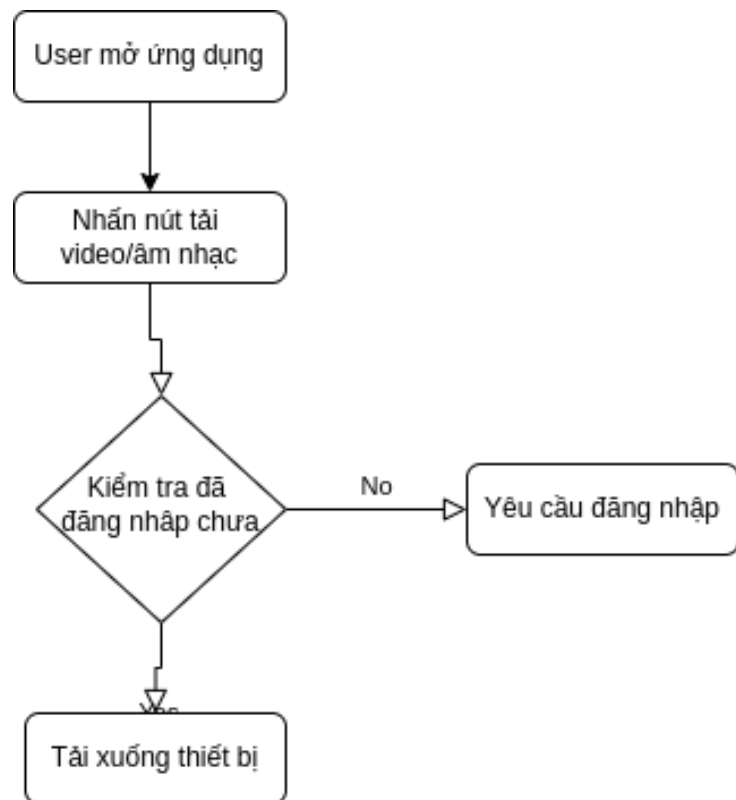
Hình 2.2: Flowchart chức năng phát nhạc

Chức năng phát video



Hình 2.3: Flowchart chức năng phát video

Chức năng tải xuống video/âm nhạc



Hình 2.4: Flowchart chức năng tải xuống video/âm nhạc

Chức năng thêm bài nhạc yêu thích



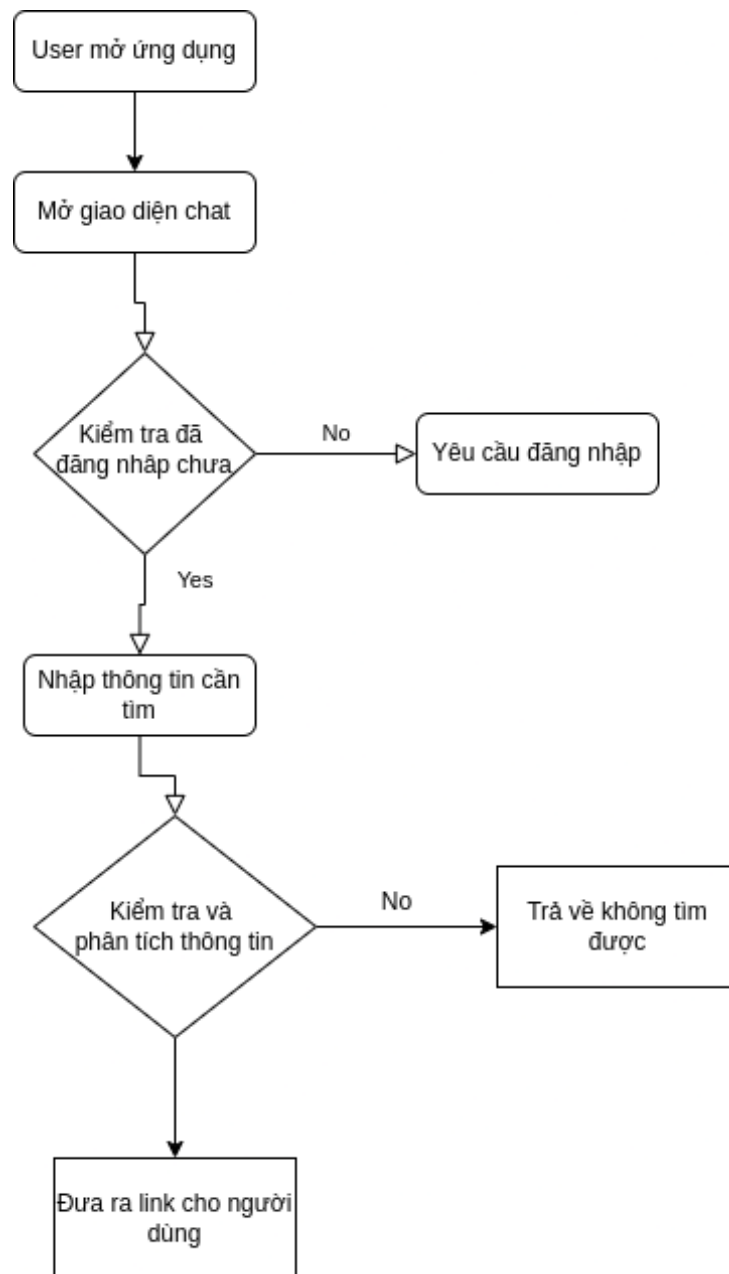
Hình 2.5: Flowchart chức năng thêm bài nhạc yêu thích

Chức năng quản lí người dùng



Hình 2.6: Flowchart chức năng quản lí người dùng

Chức năng tìm kiếm nhạc bằng AI



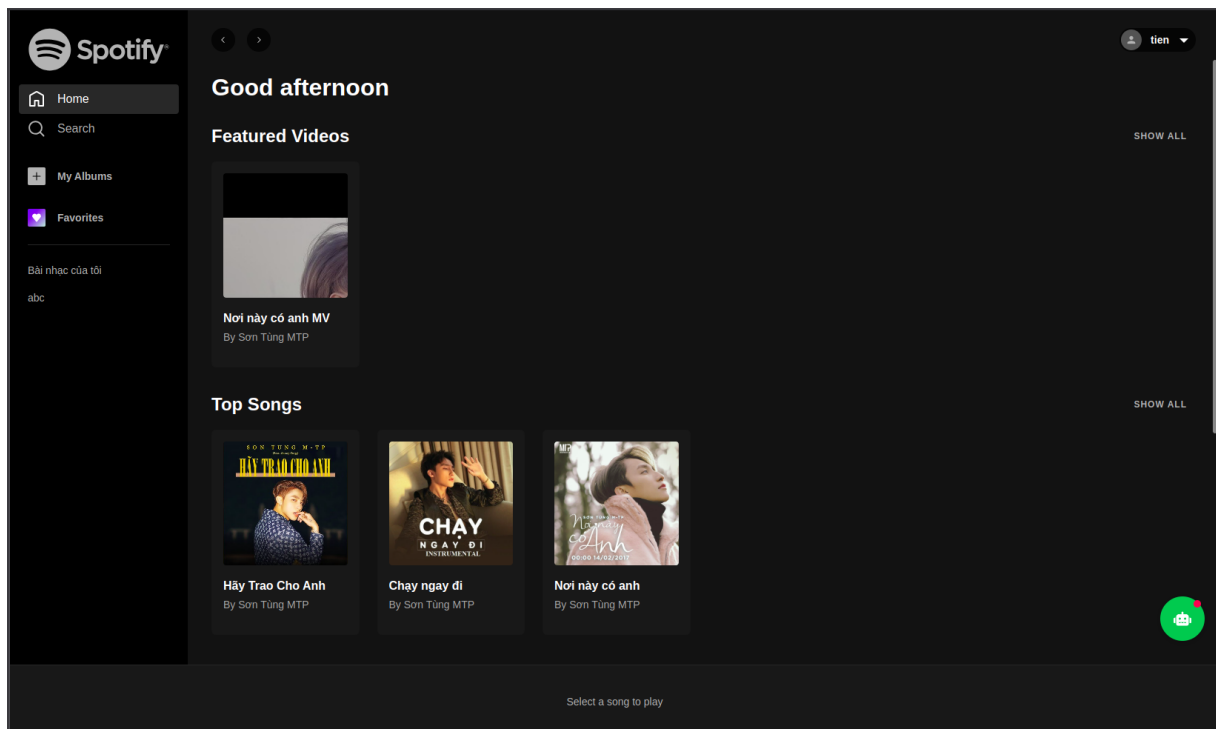
Hình 2.7: Flowchart chức năng tìm kiếm nhạc bằng AI

Chương 3

Hiện thực và Thiết kế giao diện

3.1 Thiết kế giao diện

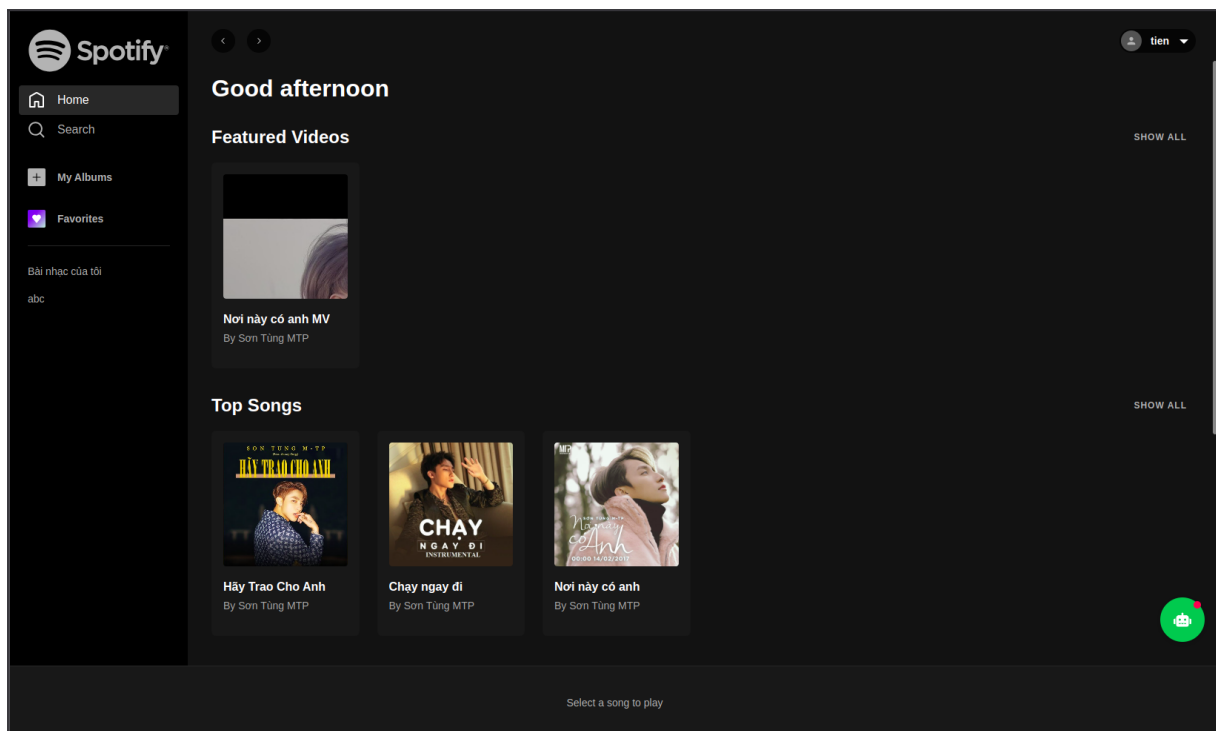
Trang chủ



Hình 3.1: Giao diện trang chủ

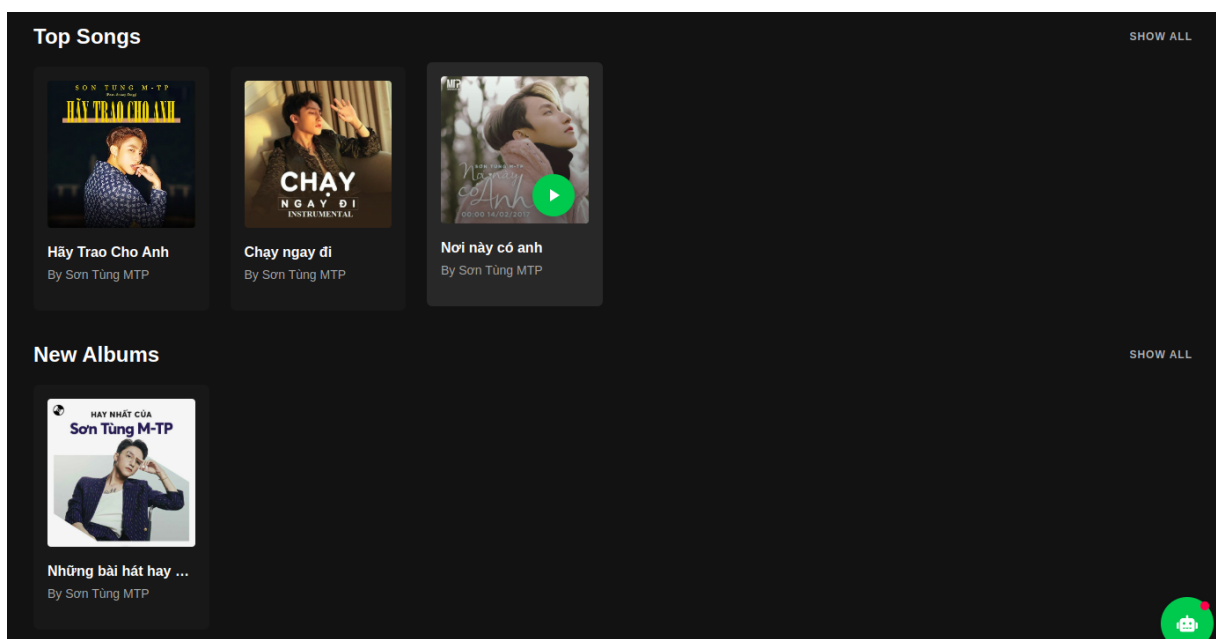
Giao diện sẽ hiện thị các section nhạc, video, album và tác giả. Bên header bao gồm Logo của ứng dụng và Nút button Đăng nhập nếu chưa đăng nhập và thông tin user nếu đã đăng nhập rồi. Bên trái là sidebar bao gồm Trang chủ, search, tạo album, danh sách nhạc yêu thích của tôi và danh sách album yêu thích của tôi (Nếu chưa đăng nhập thì không hiển thị danh sách album yêu thích và ngược lại thì đã đăng nhập rồi thì sẽ hiện thị danh sách đó ra).

Trang danh sách bài hát/album



Hình 3.2: Giao diện trang chủ

Trang phát nhạc/video

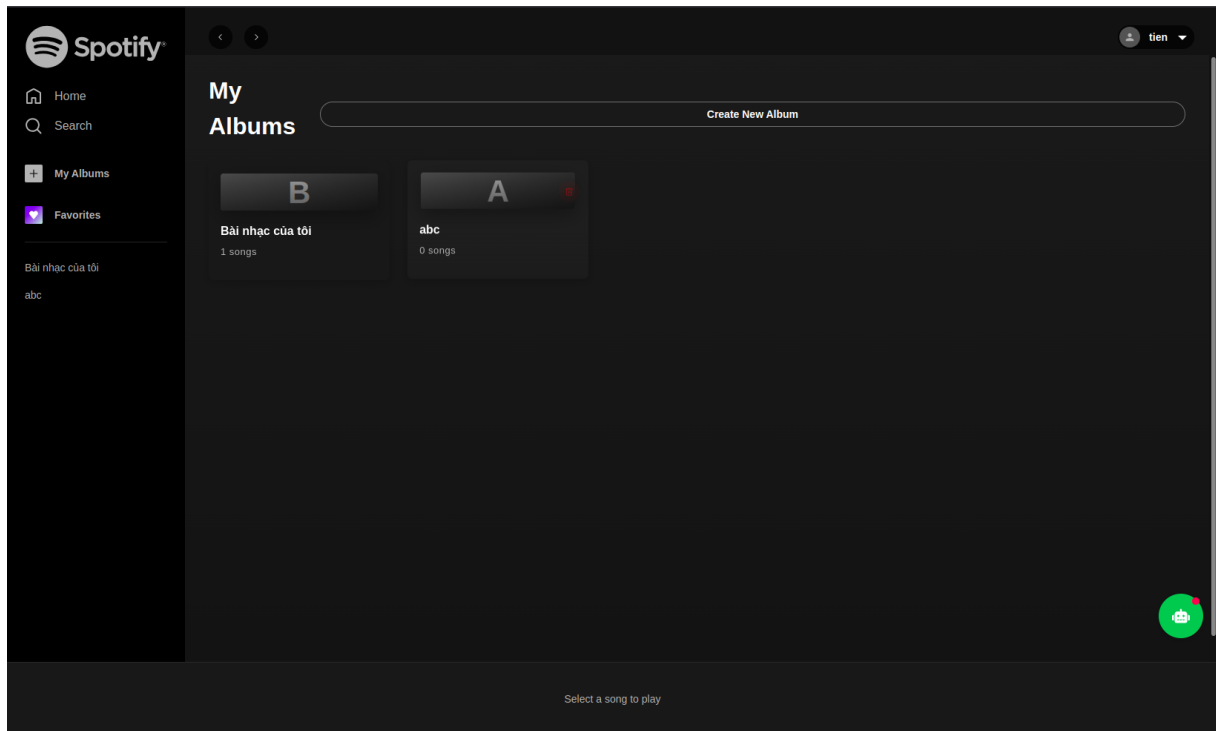


Hình 3.3: Giao diện phát nhạc/video

Gồm những danh sách nhạc, video, album, nhạc sĩ của tất cả các bài nhạc liên quan. Những bài này hiện thị dưới dạng hình chữ nhật (section). Trong mỗi section có hình ảnh của từng

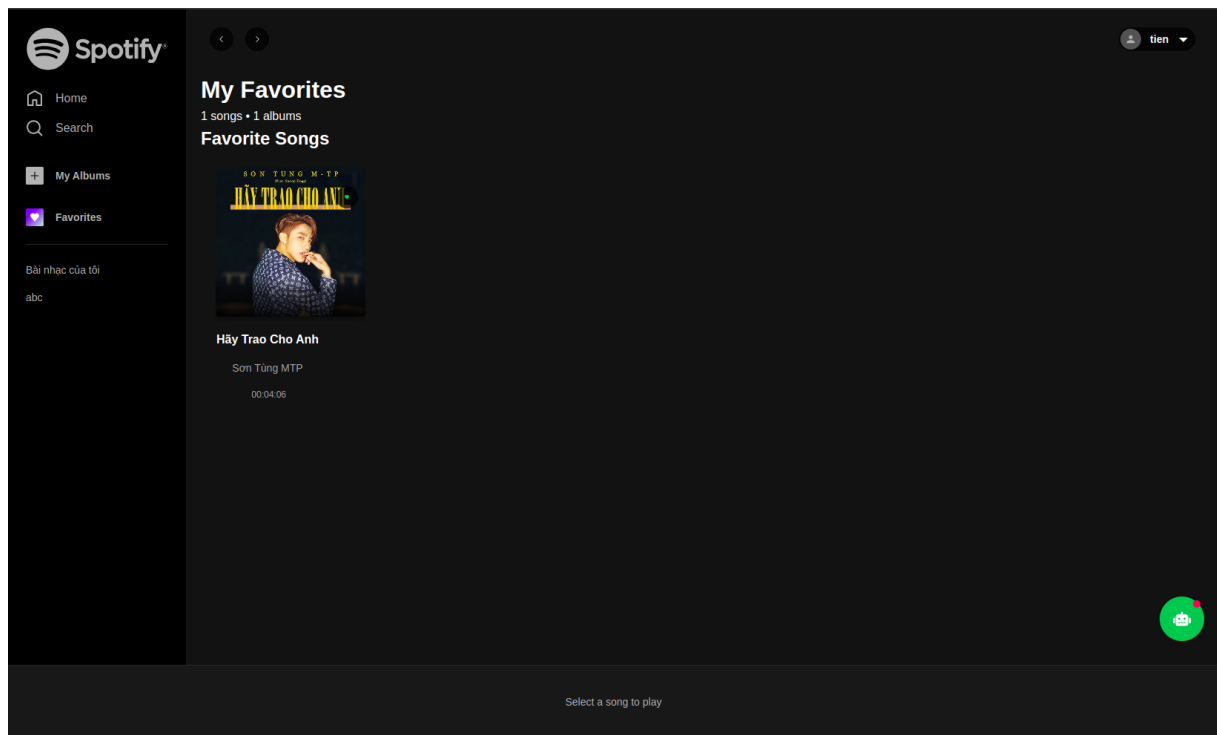
section (Album thì hiển thị hình ảnh album, nhạc thì hiển thị hình ảnh nhạc) và thông tin của section đó (Tên bài nhạc và nhạc sĩ hoặc tên album và nhạc sĩ của album đó). Khi hover section được chọn thì sẽ hiện lên nút play nhạc. Khi click thì sẽ phát nhạc được chọn

Trang tạo album/thêm bài hát yêu thích



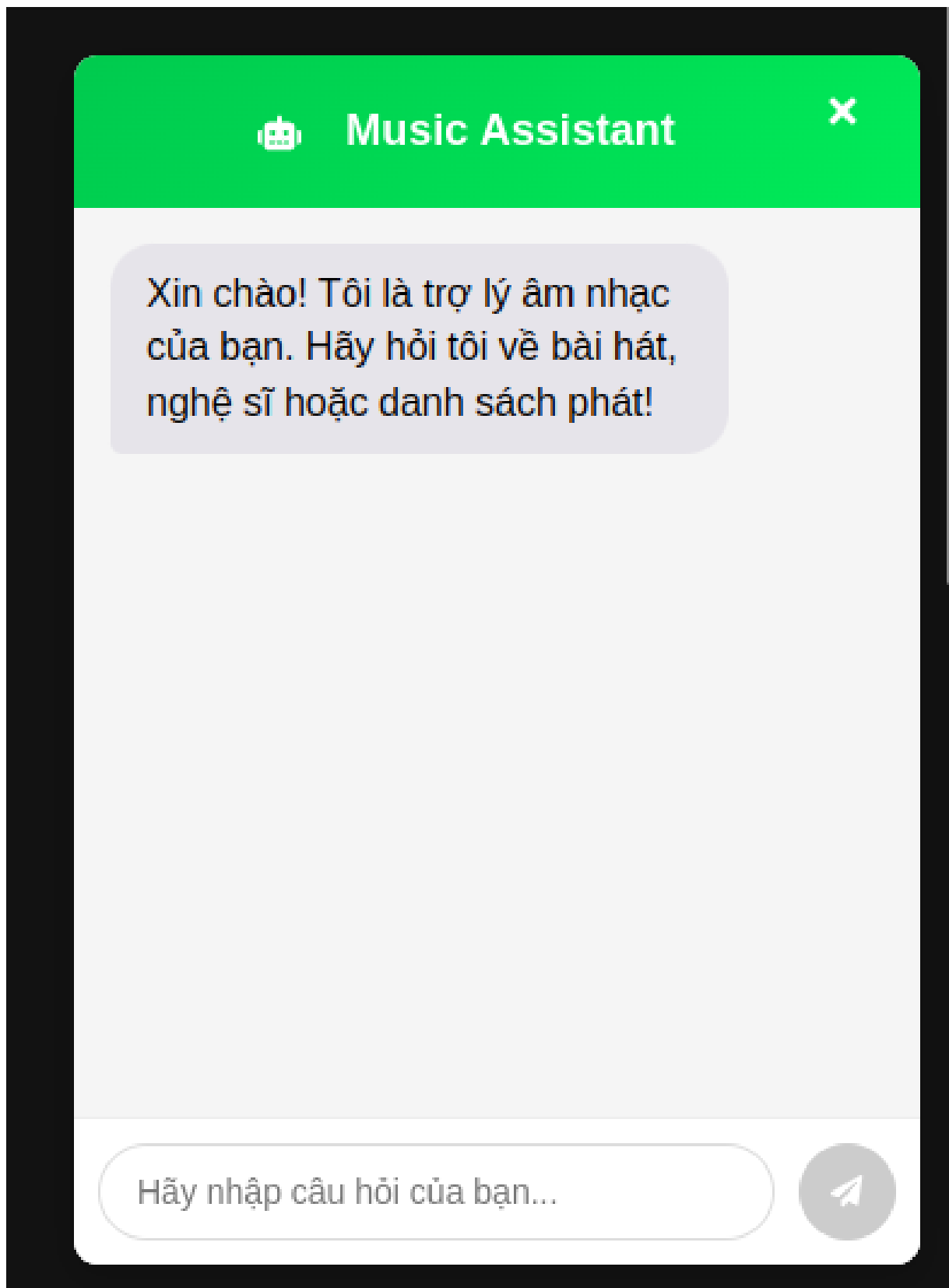
Hình 3.4: Giao diện trang tạo album

"My Albums" (Album của tôi) đang được chọn và các tùy chọn khác bị mờ đi ở phía dưới. Phía trên bên phải có nút "Create New Album" (Tạo Album Mới). Ở phần trung tâm, có hai album được hiển thị dưới dạng các thẻ hình chữ nhật bo tròn góc. Album bên trái có nhãn "B" lớn màu trắng trên nền xám, bên dưới là tiêu đề "Bài nhạc của tôi" và thông tin "1 songs" (1 bài hát). Album bên phải có nhãn "A" lớn màu trắng trên nền xám, bên dưới là tiêu đề "abc" và thông tin "0 songs" (0 bài hát). Phần lớn không gian còn lại của màn hình là màu đen, cho thấy không có nhiều album được tạo. Ở góc dưới bên phải có một biểu tượng màu xanh lá cây hình tròn có dấu cộng màu trắng bên trong. Phía dưới cùng có một dòng chữ nhỏ màu xám "Select a song to play" (Chọn một bài hát để phát)



Hình 3.5: Giao diện trang bài hát yêu thích

Tiêu đề "My Favorites" được hiển thị lớn ở phía trên, bên dưới là thông tin "1 songs. 1 albums" (1 bài hát, 1 album) và tiêu đề phụ "Favorite Songs" (Bài hát yêu thích).

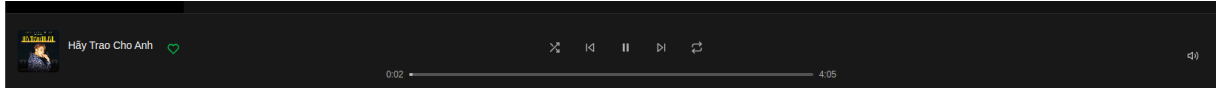


Hình 3.6: Giao diện chat box AI

Hỏi AI về nhạc thì sẽ được trả lời lại bằng link nhạc hoặc album hoặc Nhạc sĩ

3.2 Các chức năng

3.2.1 Chức năng phát nhạc



Hình 3.7: Chức năng phát nhạc

Front-end:

- Sử dụng `Audio` API để phát nhạc.
- Dùng `React Context` để quản lý trạng thái phát nhạc như:
 - Bài hát đang phát.
 - Danh sách phát (queue).
 - Trạng thái phát/tạm dừng.

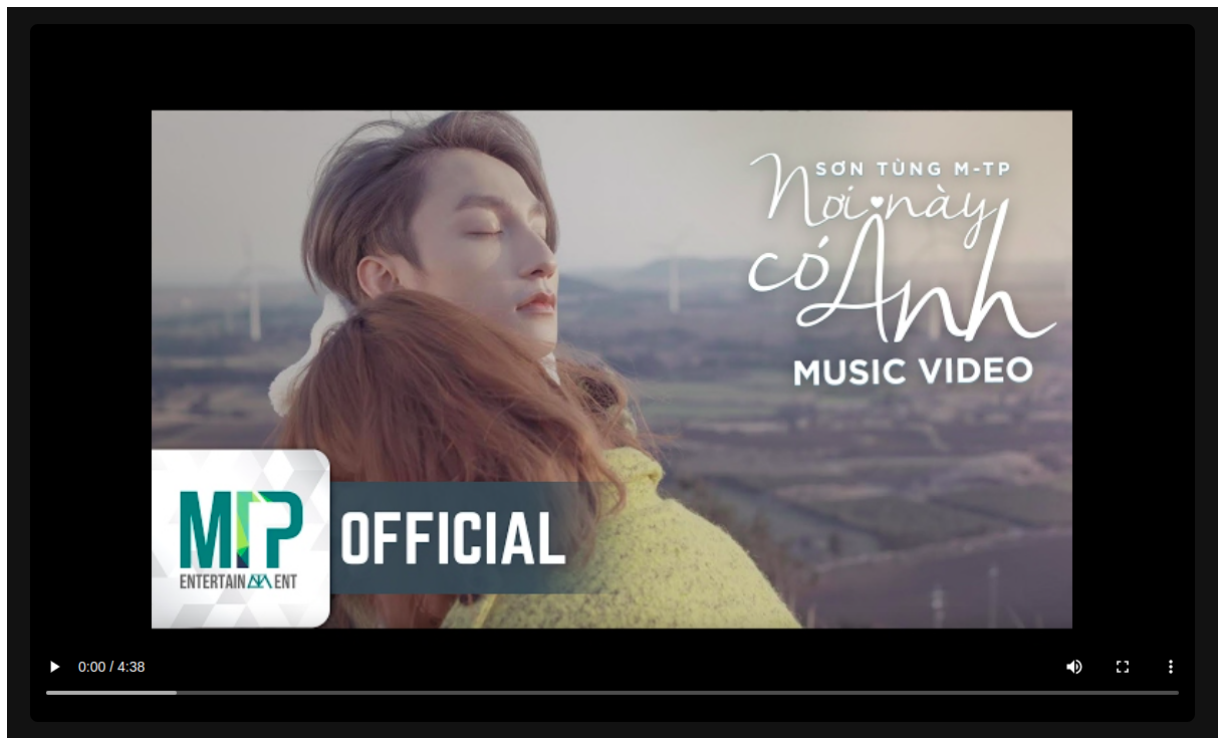
Tương tác backend:

- Gửi request API dạng `GET` đến `/api/songs/<id>/` để lấy file `.mp3`.

Điều khiển:

- **Play/Pause:** sử dụng `audioRef.current.play()` và `audioRef.current.pause()`.
- **Seek bar:** điều chỉnh vị trí phát bằng `audioRef.current.currentTime`.
- **Âm lượng:** điều chỉnh âm lượng bằng `audioRef.current.volume`.

3.2.2 Chức năng phát video âm nhạc



Hình 3.8: Chức năng phát video

Front-end:

- Sử dụng thẻ HTML `<video>`.

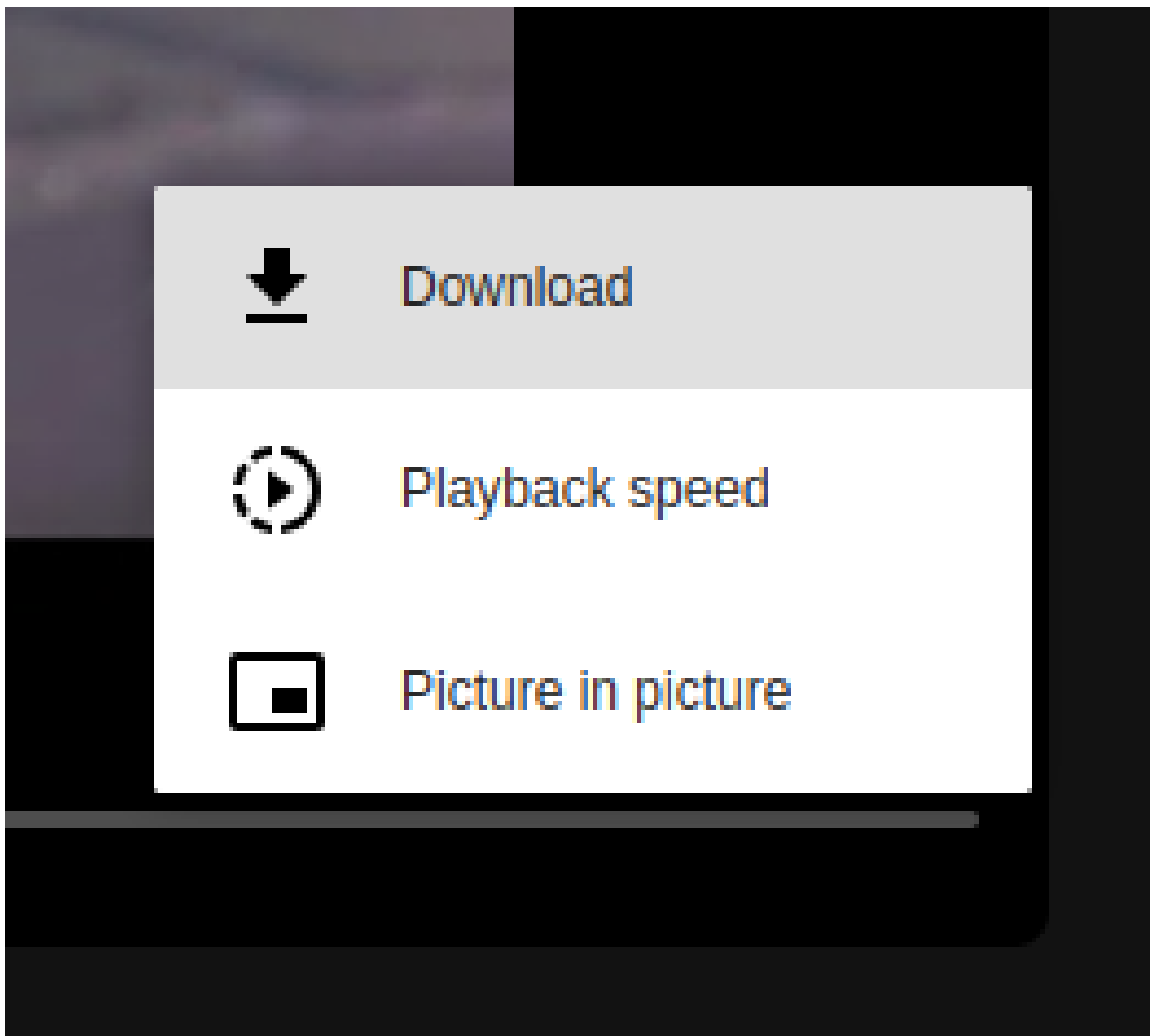
Tương tác backend:

- Gửi API GET đến `/api/videos/<id>/` để stream file `.mp4`.

Điều khiển:

- Các chức năng tạm dừng, tua, điều chỉnh âm lượng được điều khiển thông qua props của `react-player`.

3.2.3 Chức năng tải video âm nhạc



Hình 3.9: Chức năng download video âm nhạc

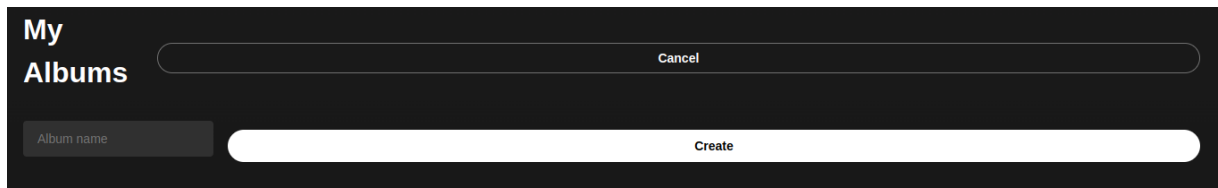
Quy trình:

1. Client gửi request đến API `/api/videos/<id>/`.
2. Server kiểm tra quyền truy cập và trả về file video với header `Content-Disposition: attachment`.

Lưu trữ:

- File video được lưu tại thư mục `/media/videos/` trên server.

3.2.4 Chức năng User tạo album, bài hát yêu thích



Hình 3.10: Chức năng tạo album

Giao diện:

- Gồm một form tạo album với các trường: tên album, ảnh bìa, mô tả.
- Hiển thị danh sách bài hát có thể chọn để thêm vào album.

Lưu trữ:

- Album: Bảng Album(id, name, user_id, cover).
- Yêu thích: Bảng FavoriteSong(user_id, song_id).

Hiển thị:

- Sử dụng các API GET như /api/user/albums/ và /api/user/favorites/ để hiển thị dữ liệu.

3.2.5 Trang Admin

Dashboard:

- Hiển thị tổng số người dùng, số bài hát, lượt phát,...

Quản lý người dùng:

- Thực hiện CRUD qua API /api/admin/users/.

Quản lý bài hát:

- Upload file nhạc, thêm/sửa thông tin bài hát.

Quản lý video:

- Tương tự quản lý bài hát nhưng xử lý file .mp4.

Quản lý album:

- Cho phép sửa thông tin album, thêm bài hát vào album.

3.2.6 Tuỳ chọn) Chức năng Chat AI

Cơ chế hoạt động:

- Tích hợp chatbot AI giúp người dùng tìm kiếm bài hát bằng ngôn ngữ tự nhiên.
- Ví dụ câu lệnh: “Phát bài hát mới nhất của Sơn Tùng”, “Gợi ý nhạc buồn piano”.
- Chatbot xử lý và ánh xạ sang API tìm kiếm bài hát phù hợp.

Giao diện:

- Khung chat hiển thị lịch sử hội thoại.
- Gợi ý nhạc dưới dạng danh sách phát nhỏ.
- Người dùng có thể nhấn vào bài hát được gợi ý để phát trực tiếp.

Xử lý phía backend:

1. Nhận input từ người dùng gửi lên server.
2. Gọi API AI handler để phân tích ý định và nội dung câu lệnh.
3. Gửi truy vấn đến API tìm kiếm bài hát: `/api/songs/search/?q=...`
4. Trả về kết quả nhạc phù hợp.

Lưu trữ (nếu có):

- Lưu lịch sử hội thoại vào bảng ChatLog(user_id, message, response, timestamp).

3.3 Cấu trúc mã nguồn

Phân chia dự án: Dự án được chia thành hai phần chính: server/ (backend dùng Django) và client/ (frontend dùng ReactJS với Vite).

Backend (server/):

- music/: App chính chứa các thành phần cốt lõi của hệ thống nhạc:
 - models.py: Định nghĩa các mô hình dữ liệu (bài hát, album, video...).
 - serializers.py: Chuyển đổi dữ liệu giữa Python object và JSON.
 - views.py: Xử lý logic của các API liên quan đến âm nhạc.
 - urls.py: Cấu hình đường dẫn cho các API trong app.
 - consumers.py: Xử lý giao tiếp WebSocket cho chức năng chat real-time.
 - admin.py: Đăng ký model cho trang quản trị Django.
- media/: Chứa các tệp media đã upload như songs/, videos/, albums/, artists/.
- manage.py: Tập tin điều khiển quản lý project Django.
- db.sqlite3: Cơ sở dữ liệu SQLite của hệ thống.

Frontend (client/):

- `src/components/`: Các component chính của giao diện:
 - `MusicPlayer.jsx`, `Player.jsx`: Giao diện phát nhạc/video.
 - `ChatBot.jsx`: Giao diện chat với AI.
 - `Sidebar.jsx`, `AppHeader.jsx`: Thanh điều hướng và tiêu đề.
- `contexts/`: Sử dụng React Context để quản lý trạng thái như: phát nhạc (`MusicPlayerContext.jsx`), người dùng (`AuthContext.jsx`).
- `services/api.js`: Định nghĩa các hàm gọi API đến server.
- `pages/`: Các trang chính (trang chủ, danh sách bài hát, album, v.v.).
- `styles/`, `assets/`: Quản lý CSS và tài nguyên tĩnh (ảnh, icon...).
- `App.jsx`, `main.jsx`: Cấu trúc khởi tạo ứng dụng React.
- `vite.config.js`: Cấu hình Vite cho frontend.

Chương 4

Cài đặt và Hướng dẫn sử dụng

4.1 Môi trường chạy ứng dụng

- **Hệ điều hành:** Linux (Ubuntu 20.04 trở lên khuyến nghị).
- **Phần mềm yêu cầu:**
 - Python 3.10+
 - Node.js 18+ và npm hoặc yarn
 - Django 4.x
 - Django REST Framework
 - Vite (dùng cho frontend React)
 - SQLite3 hoặc PostgreSQL/MySQL

4.2 Hướng dẫn cài đặt

Cài đặt Backend (Django)

1. Di chuyển đến thư mục `server/`
2. Tạo môi trường ảo:

```
python -m venv venv
source venv/bin/activate
```

3. Cài dependencies:

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

4. Tạo database và migrate:

```
python manage.py makemigrations
python manage.py migrate
```

5. Khởi động server:

```
python manage.py runserver
```

Cài đặt Frontend (React)

1. Di chuyển đến thư mục client/

2. Cài dependencies:

```
npm install
```

3. Chạy frontend development server:

```
npm run dev
```

4.3 Hướng dẫn sử dụng

Người dùng cuối

- Đăng ký/Đăng nhập tài khoản.
- Nghe nhạc: Chọn bài hát từ danh sách để phát.
- Xem video: Truy cập phần video để phát nhạc dạng video.
- Tạo album: Truy cập mục Album, nhấn “Tạo album” và chọn bài hát.
- Thêm bài hát yêu thích: Nhấn biểu tượng “trái tim” ở bài hát.

Quản trị viên (Admin)

- Truy cập trang của django
- Quản lý người dùng: Thêm, sửa, xóa.
- Quản lý bài hát: Thêm mới, upload nhạc, chỉnh sửa metadata.
- Quản lý video: Tương tự bài hát, có thêm upload video.
- Quản lý album: Tạo, chỉnh sửa, thêm bài hát vào album.

4.4 Các chú ý khi sử dụng phần mềm

- Luôn đảm bảo backend và frontend đang chạy đồng thời.
- Media files (ảnh, nhạc, video) cần được đặt đúng vào thư mục `media/`.
- Kiểm tra file `.env` chứa đầy đủ thông tin cấu hình (nếu sử dụng PostgreSQL hoặc môi trường production).
- Không upload file quá lớn (trên 100MB) nếu không có CDN hoặc dịch vụ lưu trữ ngoài.
- Khi gặp lỗi kết nối API, kiểm tra lại URL backend được cấu hình trong frontend (`api.js`).

4.5 Kết luận và Hướng phát triển

Kết luận

Dự án Spotify Clone được xây dựng với mục tiêu mô phỏng một nền tảng phát nhạc trực tuyến với các chức năng cơ bản và nâng cao như: phát nhạc, phát video âm nhạc, tạo album cá nhân, thêm bài hát vào danh sách yêu thích, quản lý dữ liệu qua trang Admin và tích hợp chatbot AI.

Hệ thống sử dụng kiến trúc tách biệt frontend (ReactJS + Vite) và backend (Django REST Framework), đảm bảo rõ ràng về mặt xử lý logic, giao diện và bảo trì hệ thống. Các chức năng chính đã được hiện thực đầy đủ, đồng thời hỗ trợ lưu trữ dữ liệu media hiệu quả, đảm bảo trải nghiệm người dùng thân thiện.

Hướng phát triển

Dự án có tiềm năng mở rộng với nhiều tính năng mới trong tương lai:

- **Gợi ý âm nhạc thông minh:** Sử dụng thuật toán học máy để gợi ý bài hát theo lịch sử nghe.
- **Tích hợp thanh toán:** Cho phép người dùng nâng cấp tài khoản (premium) để tải nhạc không giới hạn hoặc nghe nhạc không quảng cáo.
- **Tạo playlist công khai và chia sẻ:** Cho phép người dùng chia sẻ playlist với cộng đồng.
- **Phát trực tiếp (livestream nhạc):** Cho phép nghệ sĩ phát trực tiếp tới người hâm mộ.
- **Ứng dụng di động:** Xây dựng phiên bản mobile sử dụng React Native hoặc Flutter.
- **Tối ưu hiệu năng streaming:** Tích hợp các CDN hoặc dịch vụ như AWS S3 để cải thiện tốc độ truyền tải.

Với nền tảng kỹ thuật hiện tại, dự án hoàn toàn có khả năng phát triển thành một hệ thống hoàn chỉnh, phục vụ nhu cầu giải trí âm nhạc cho người dùng trên nhiều nền tảng.

Tài liệu tham khảo

- [1] Spotify Developer. Spotify web api. <https://developer.spotify.com/documentation/web-api>, 2025. Accessed on: 1 May 2025.
- [2] Adrian Holovaty and Jacob Kaplan-Moss. *The Definitive Guide to Django: Web Development Done Right*. Apress, 2nd edition, 2023.
- [3] Mark Winterbottom. *Designing RESTful APIs with Django*. Leanpub, 2021. <https://leanpub.com/djangoapi>.
- [4] React Team. React – a javascript library for building user interfaces. <https://react.dev>, 2025. Accessed on: 1 May 2025.
- [5] GoldFire Studios. Howler.js – javascript audio library. <https://howlerjs.com>, 2024. Accessed on: 1 May 2025.
- [6] Evan You. Vite – next generation frontend tooling. <https://vitejs.dev>, 2025. Accessed on: 1 May 2025.