Trường Đại Học Sài Gòn

BÁO CÁO ĐỒ ÁN

Đề tài

Trang web nghe nhạc Spotify

Nhóm sinh viên thực hiện

Tăng Hoàng Lương - 3121560052
Lương Gia Tuấn - 3121560005
Phan Huỳnh Thanh Phong - 3121560003
Nguyễn Thị Mai Trinh - 3122410428

Email nhóm: `luonghaong2468@gmail.com`

Ngày 22 tháng 5 năm 2025

Mục lục

1 Phân công công việc

Tên	**Nhiệm vụ**
Tăng Hoàng Lương	Setup frontend, làm các control của frontend, Nối API frontend với backend
Lương Gia Tuấn	Setup Bucket, Viết API stream song, video, setup aws, apache2, SSL để deploy
Phan Huỳnh Thanh Phong	Viết API account, Playlist, Follow, Yêu thích, Setup JWT
Nguyễn Thị Mai Trinh	Viết báo cáo, làm giao diện trang web

2 Giới thiệu

Trong bối cảnh công nghệ số ngày càng phát triển mạnh mẽ, nhu cầu giải trí trực tuyến, đặc biệt là âm nhạc và video, đã trở thành một phần không thể thiếu trong đời sống hàng ngày của người dùng. Các nền tảng phát nhạc và video theo yêu cầu như Spotify, YouTube Music hay Apple Music đã và đang định hình lại cách con người tiếp cận và tận hưởng các nội dung đa phương tiện. Điều này đặt ra yêu cầu cho việc phát triển những ứng dụng ngày càng thông minh, linh hoạt và cá nhân hóa hơn để đáp ứng thị hiếu ngày càng đa dạng của người dùng.

Xuất phát từ thực tiễn đó, đề tài lựa chọn hướng tới việc xây dựng một ứng dụng web hiện đại, cho phép người dùng nghe nhạc và xem video âm nhạc trực tuyến theo yêu cầu, đồng thời cung cấp trải nghiệm cá nhân hóa, tiện lợi và thân thiện. Không chỉ là công cụ phát nội dung, hệ thống còn đóng vai trò như một nền tảng giải trí số, nơi người dùng có thể tạo album riêng, lưu trữ danh sách yêu thích, theo dõi nghệ sĩ yêu thích và tải nội dung về thiết bị.

Mục tiêu chính của đề tài là phát triển một nền tảng web tích hợp đầy đủ các chức năng cốt lõi và hiện đại, cụ thể như:

- **Truy cập đa nền tảng**: Hệ thống có khả năng hoạt động trên nhiều loại thiết bị (máy tính, điện thoại, máy tính bảng) thông qua trình duyệt web, giúp người dùng thưởng thức âm nhạc mọi lúc, mọi nơi.
- **Cá nhân hóa trải nghiệm**: Hỗ trợ người dùng tạo và quản lý album cá nhân, lưu danh sách bài hát yêu thích, theo dõi nghệ sĩ, và nhận gợi ý nội dung dựa trên sở thích.
- **Tải xuống tiện lợi**: Cho phép người dùng tải video âm nhạc để xem ngoại tuyến trong trường hợp không có kết nối internet.
- **Giao diện hiện đại**: Xây dựng giao diện thân thiện, dễ sử dụng với thiết kế trực quan, đẹp mắt, và có khả năng tùy chỉnh theo sở thích cá nhân.
- **Hiệu suất và mở rộng**: Hệ thống được thiết kế để xử lý hiệu quả với lượng người dùng lớn và dễ dàng mở rộng, tích hợp thêm các tính năng nâng cao như trí tuệ nhân tạo hoặc mạng xã hội trong tương lai.

Ngoài những tính năng phục vụ trải nghiệm người dùng, đề tài cũng đề cao yếu tố kỹ thuật và bảo mật. Hệ thống sẽ ứng dụng các công nghệ tiên tiến như RESTful API, lưu trữ đám mây (AWS S3), và mã hóa SSL nhằm đảm bảo hiệu suất cao và an toàn dữ liệu.

Với định hướng đó, đề tài kỳ vọng tạo ra một sản phẩm không chỉ đáp ứng được nhu cầu giải trí số hiện tại, mà còn đủ tiềm năng phát triển thành một nền tảng toàn diện, cạnh tranh và có giá trị lâu dài trong lĩnh vực nội dung số.

3 Cơ sở lý thuyết

3.1 Phần mềm mã nguồn mở

Phần mềm mã nguồn mở (open source software) là loại phần mềm mà mã nguồn được công khai, cho phép người dùng tự do truy cập, sử dụng, chỉnh sửa và phân phối mã nguồn theo các điều khoản của giấy phép mã nguồn mở. Đặc điểm chính của phần mềm mã nguồn mở bao gồm:

- **Tự do truy cập**: Mã nguồn được cung cấp công khai, cho phép người dùng nghiên cứu và sử dụng mà không bị hạn chế.
- **Cộng đồng phát triển**: Người dùng và lập trình viên có thể đóng góp vào việc phát triển, sửa lỗi hoặc bổ sung tính năng mới.
- **Miễn phí hoặc chi phí thấp**: Phần mềm mã nguồn mở thường được cung cấp miễn phí, nhưng có thể có chi phí liên quan đến các dịch vụ hỗ trợ hoặc tùy chỉnh.
- **Tính minh bạch**: Mã nguồn công khai giúp tăng cường tính an toàn và cho phép cộng đồng kiểm tra, phát hiện lỗi hoặc lỗ hổng bảo mật.

Một số ví dụ phổ biến về phần mềm mã nguồn mở bao gồm hệ điều hành Linux, trình duyệt web Mozilla Firefox, trình phát đa phương tiện VLC Media Player và bộ ứng dụng văn phòng LibreOffice.

3.2 Các loại giấy phép mã nguồn mở

Giấy phép mã nguồn mở là các thỏa thuận pháp lý quy định cách phần mềm mã nguồn mở có thể được sử dụng, chỉnh sửa và phân phối. Các giấy phép này đảm bảo quyền tự do của người dùng đồng thời đặt ra một số điều kiện nhất định. Dưới đây là một số loại giấy phép mã nguồn mở phổ biến:

- **GNU General Public License (GPL)**: Đây là giấy phép phổ biến nhất, yêu cầu mọi phần mềm phái sinh (dựa trên mã nguồn GPL) cũng phải được phát hành dưới giấy phép GPL. Điều này đảm bảo mã nguồn luôn được công khai và tự do.
- **MIT License**: Một giấy phép đơn giản và linh hoạt, cho phép người dùng sử dụng, chỉnh sửa và phân phối phần mềm với ít hạn chế, miễn là giữ nguyên thông báo bản quyền.
- **Apache License**: Cung cấp sự linh hoạt tương tự như MIT, nhưng bao gồm các điều khoản rõ ràng hơn về bản quyền và bảo vệ pháp lý cho người đóng góp.
- **BSD License**: Tương tự MIT, giấy phép BSD cho phép sử dụng mã nguồn mà không yêu cầu phần mềm phái sinh phải công khai mã nguồn, nhưng vẫn yêu cầu giữ thông báo bản quyền.
- **Mozilla Public License (MPL)**: Kết hợp giữa GPL và các giấy phép tự do hơn, MPL yêu cầu mã nguồn của các tệp được chỉnh sửa phải công khai, nhưng không bắt buộc toàn bộ dự án phái sinh phải tuân theo MPL.

Mỗi loại giấy phép có các đặc điểm riêng, phù hợp với các mục đích khác nhau, từ việc bảo vệ quyền tự do của mã nguồn (như GPL) đến việc cho phép tích hợp vào các dự án thương mại (như MIT hoặc Apache). Việc lựa chọn giấy phép phụ thuộc vào mục tiêu của nhà phát triển và cộng đồng sử dụng phần mềm.

3.3 Công nghệ sử dụng

Dự án này sử dụng hai công nghệ chính: **frontend** sử dụng ReactJS kết hợp với Next.js, và **backend** sử dụng Django kết hợp với cơ sở dữ liệu MongoDB.

3.3.1 Frontend

- Next.js: Framework mạnh mẽ xây dựng trên nền tảng React, hỗ trợ server-side rendering (SSR), static site generation (SSG), routing tự động, và tối ưu SEO.
- React.js: Thư viện JavaScript dùng để xây dựng giao diện người dùng theo hướng component.
- TypeScript: Ngôn ngữ lập trình mở rộng từ JavaScript, hỗ trợ kiểm tra kiểu tĩnh trong quá trình biên dịch, giúp mã nguồn rõ ràng và dễ bảo trì.
- Tailwind CSS: CSS framework theo hướng utility-first, giúp xây dựng giao diện nhanh chóng và linh hoạt.

3.3.2 Backend

- Django: Framework web mạnh mẽ của Python, giúp xây dựng backend nhanh chóng với kiến trúc MTV (Model-Template-View).
- Django REST Framework (DRF): Thư viện mở rộng của Django, hỗ trợ xây dựng các API RESTful một cách dễ dàng và chuẩn hóa.
- S3 Bucket (AWS S3): Dịch vụ lưu trữ đối tượng của Amazon Web Services, được sử dụng để lưu trữ và quản lý các tệp tĩnh như hình ảnh, video, tài liệu.
- JWT (JSON Web Token): Cơ chế xác thực người dùng bằng token, giúp đảm bảo tính bảo mật và không cần lưu trạng thái phiên đăng nhập trên server.

3.3.3 Cơ sở dữ liệu

- MongoDB: Cơ sở dữ liệu NoSQL dạng document, lưu trữ dữ liệu dưới dạng JSON (BSON), phù hợp với các ứng dụng web linh hoạt, mở rộng dễ dàng và truy vấn nhanh.

3.4 Cấu trúc mã nguồn

Dự án được chia thành hai phần chính: **frontend** sử dụng ReactJS và Next.js (một framework dựa trên ReactJS) và **backend** sử dụng Django. Cấu trúc mã nguồn giúp tổ chức tốt hơn và dễ dàng triển khai.

Frontend – ReactJS và Next.js (MuseArchive)

```
MuseArchive/
+ app/                          // Chứa các thành phần của ứng dụng
|   +-- AppWrapper.tsx          // Sử dụng để bao bọc toàn bộ ứng dụng
|   +-- album                   // Quản lý album
|   +-- api                     // API của ứng dụng
|   +-- component               // Các component tái sử dụng
|   +-- context                 // quản lý và chia sẻ trạng thái toàn cục
|   +-- librarypage             // Trang thư viện
|   +-- likedsongpage           // Trang các bài hát yêu thích
|   +-- loginpage               // Trang đăng nhập tương ứng với route login
|   +-- page.tsx                // Tệp giao diện trang chính
|   +-- musician                // Trang nhạc sĩ
|   +-- playlist                // Trang danh sách phát
|   +-- profile                 // Trang cá nhân
|   +-- settingpage             // Trang cài đặt
|   +-- signuppage              // Trang đăng ký
|   +-- track                   // Trang bài hát
+ eslint.config.mjs             // Cấu hình ESLint cho dự án
+ next-env.d.ts                 // Tệp khai báo các loại Next.js
+ next.config.ts                // Cấu hình Next.js
+ node_modules                  // Thư mục chứa các thư viện phụ thuộc
+ package-lock.json             // Quản lý phiên bản phụ thuộc
+ package.json                  // Thông tin và các phụ thuộc của dự án
+ postcss.config.mjs            // Cấu hình PostCSS
+ public                        // Chứa tài nguyên tĩnh như ảnh, favicon...
+ tsconfig.json                 // Cấu hình TypeScript
```

Frontend của dự án sử dụng **React** kết hợp với **Next.js** phiên bản 13+, sử dụng cấu trúc thư mục mới `app/` để tổ chức route và layout.

Trong đó:

- Mỗi route là một thư mục con bên trong `app/`, và chứa một file `page.tsx` đại diện cho nội dung hiển thị của trang.
- Ví dụ: `app/login/page.tsx` là trang đăng nhập, `app/signup/page.tsx` là trang đăng ký.
- Thành phần `AppWrapper` được sử dụng để bao bọc toàn bộ ứng dụng, giúp chia sẻ các context chung (như PlayerProvider, AuthProvider) cho tất cả các trang.

Thư mục component chứa các component được tái sử dụng ở nhiều nơi như Header, Sidebar, Navbar,...

Thư mục context được dùng để quản lý và chia sẻ trạng thái toàn cục (global state) hoặc logic dùng chung cho nhiều component trong ứng dụng. Ví dụ có PlayContext thể hiện trạng thái bài hát đang phát, thời gian, play/pause/next/prev

Backend – Django (django-spotify)

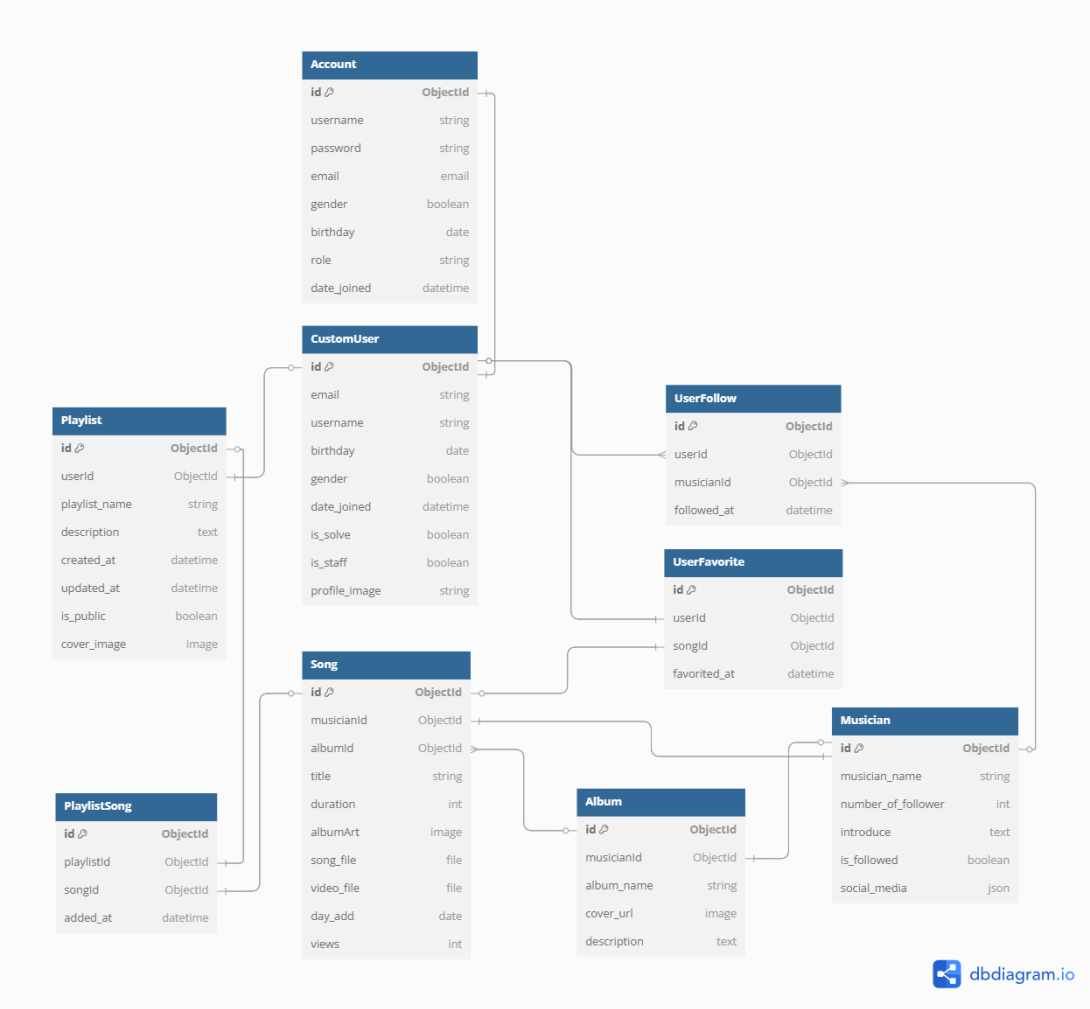
```
django-spotify/
```

```
+-- spotify/                    // Thư mục cấu hình chính (settings, wsgi, urls...)
+-- spotify_app/
|   +-- migrations/             // Các file migration của cơ sở dữ liệu
|   +-- admin.py                // Cấu hình trang admin của Django
|   +-- apps.py                 // Cấu hình app
|   +-- models.py               // Định nghĩa các model (bảng dữ liệu)
|   +-- views.py                // Xử lý các request và trả về response
|   +-- urls.py                 // Định nghĩa các route API
|   +-- serializers.py          // Chuyển đổi dữ liệu giữa model và JSON
+-- spotify_users/              // Xử lý người dùng
+-- manage.py                   // Tệp chính để chạy các lệnh Django
+-- requirements.txt            // Danh sách các thư viện Python cần thiết
+-- db.sqlite3                  // Cơ sở dữ liệu
+-- README.md                   // Tài liệu hướng dẫn cài đặt và sử dụng
```

Mã nguồn backend sử dụng **Django** và được tổ chức với các module chính như `models`, `views`, `serializers`, và `urls` để xử lý logic, API và cơ sở dữ liệu.

3.5 Cơ sở dữ liệu

3.5.1 Sơ đồ ERD:



3.5.2 Mô tả các bảng:

Bảng `CustomUser`

Tên trường	Kiểu dữ liệu	Ghi chú
id	ObjectId	Khóa chính (PK)
email	string	
username	string	
birthday	date	
gender	boolean	
date_joined	datetime	
is_active	boolean	
is_staff	boolean	
profile_image	string	

Bảng `Account`

Tên trường	Kiểu dữ liệu	Ghi chú
id	ObjectId	Khóa chính (PK)
username	string	
email	email	
gender	boolean	
birthday	date	
role	string	
date_joined	datetime	

Bảng `UserFollow`

Tên trường	Kiểu dữ liệu	Ghi chú
id	ObjectId	Khóa chính (PK)
followed_at	datetime	

Bảng `Musician`

Tên trường	Kiểu dữ liệu	Ghi chú
id	ObjectId	Khóa chính (PK)
musician_name	string	
number_of_follower	int	
introduce	text	
is_followed	boolean	
social_media	json	

Bảng `Album`

Tên trường	Kiểu dữ liệu	Ghi chú
id	ObjectId	Khóa chính (PK)
album_name	string	
coverurl	image	
description	text	

Bảng `Song`

Tên trường	Kiểu dữ liệu	Ghi chú
id	ObjectId	Khóa chính (PK)
title	string	
duration	int	
albumArt	image	
song_file	file	
video_file	file	
day_add	date	
views	int	

Bảng `Playlist`

Tên trường	Kiểu dữ liệu	Ghi chú

id	ObjectId	Khóa chính (PK)
playlist_name	string	
description	text	
created_at	datetime	
updated_at	datetime	
is_public	boolean	
cover_image	image	

Bảng `UserFavorite`

Tên trường	**Kiểu dữ liệu**	**Ghi chú**
id	ObjectId	Khóa chính (PK)
favorited_at	datetime	

Mối quan hệ giữa các bảng:

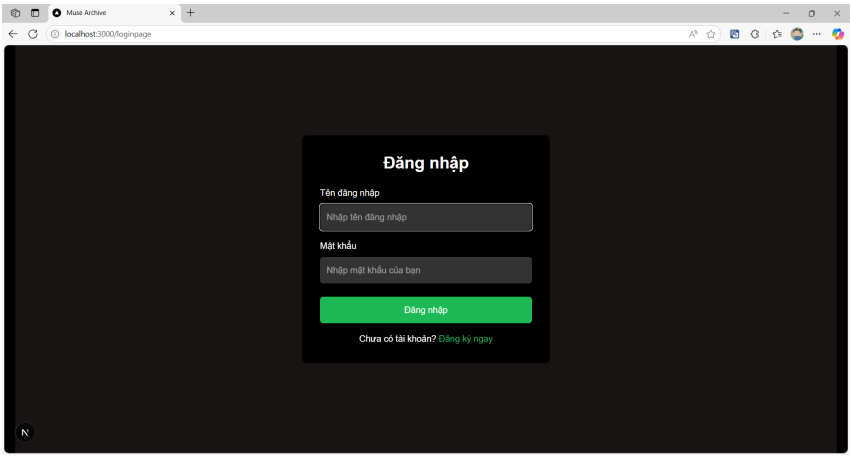
- `CustomUser` có thể tạo `Playlist`, theo dõi `Musician`, và yêu thích `Song`.
- `Musician` có thể tạo `Song`, biểu diễn trong `Album`.
- `Album` chứa nhiều `Song`.
- `Song` có thể thuộc nhiều `Playlist`, và được nhiều `UserFavorite` đánh dấu.

4 Hiện thực hệ thống

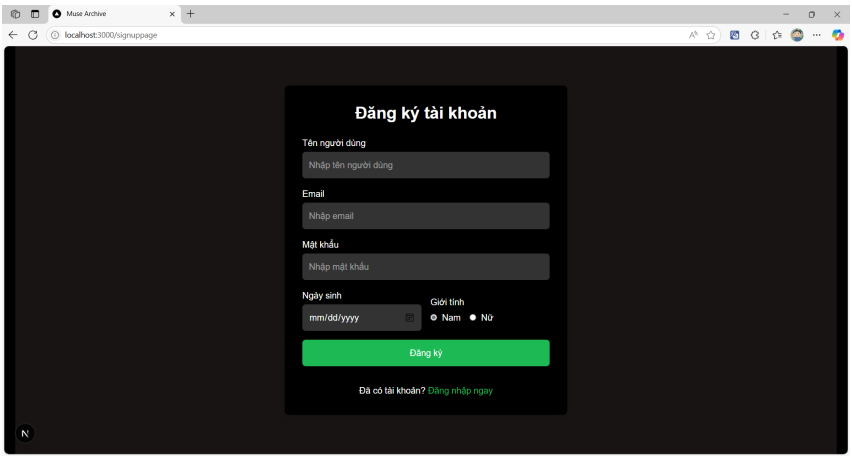
4.1 Tính năng đã xây dựng

- Đăng ký / đăng nhập
- Chức Năng Phát Nhạc: phát các bài hát khi click, có các chức năng như: dừng-chạy bài, trộn bài, lặp bài, bài tiếp theo hoặc bài trước đó trong playlist/album, điều chỉnh âm lượng
- Trang Chủ: hiển thị các playlist, bài hát, nhạc sĩ hiện có
- Trang Chi Tiết Nhạc Sĩ: hiển thị ảnh avatar, ảnh nền, chi tiết của nhạc sĩ, danh sách các bài hát được nghe nhiều nhát của nhạc sĩ, các album của nhạc sĩ đó.
- Trang Thông Tin Cá Nhân: hiển thị thông tin cá nhân của người dùng đang đăng nhập
- Trang Chi Tiết Nhạc: hiển thị thông tin bài hát, video âm nhạc của bài hát đó, cho tải về video âm nhạc.
- Sidebar: hiển thị các playlist user đã tạo, hiển thị playlist bài hát yêu thích (khi nhấn tim ở dòng bài hát thì sẽ cho bài hát đó vào đây), có nút tạo playlist mới
- Trang bài hát yêu thích: để danh sách các bài hát đã tim vào đó
- Trang Playlist: hiển thị danh sách các bài hát (có 2 loại, playlist của user và playlist thường)
- Trang Album: hiển thị danh sách các bài hát trong album (chỉ có nhạc sĩ mới có album

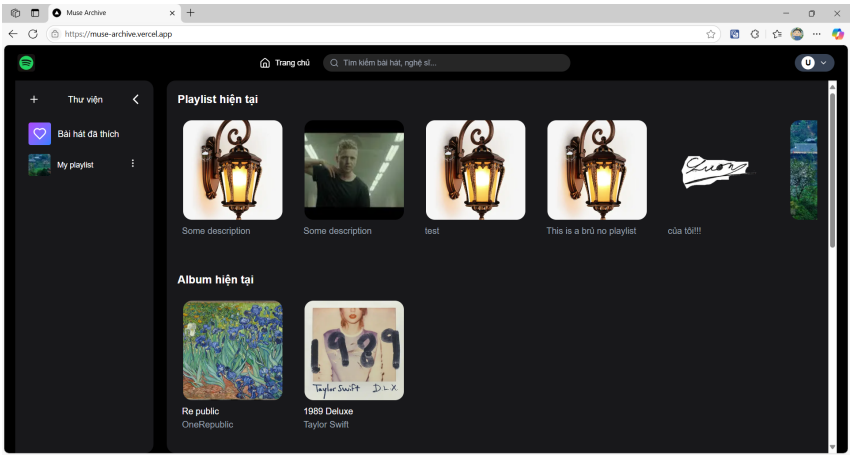
4.2 Giao diện ứng dụng



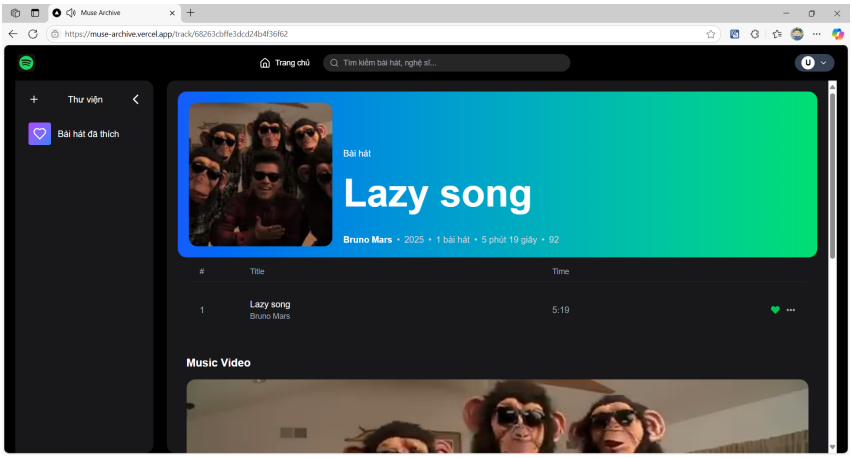
Hình 1: Giao diện trang đăng nhập



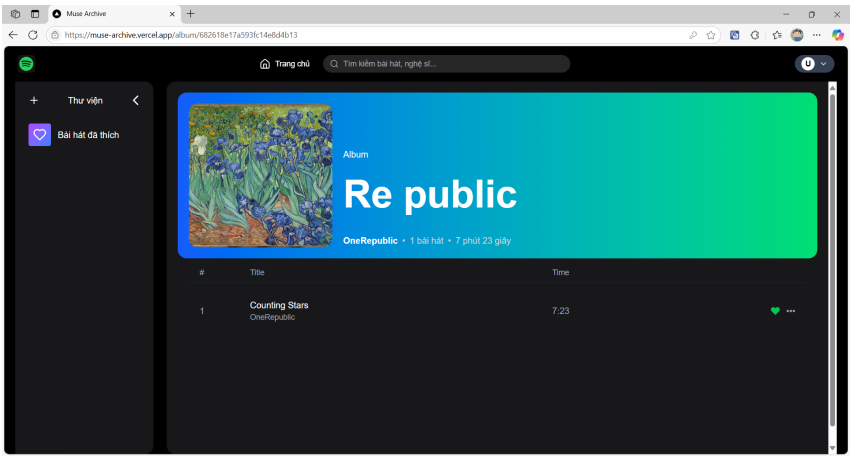
Hình 2: Giao diện trang đăng ký



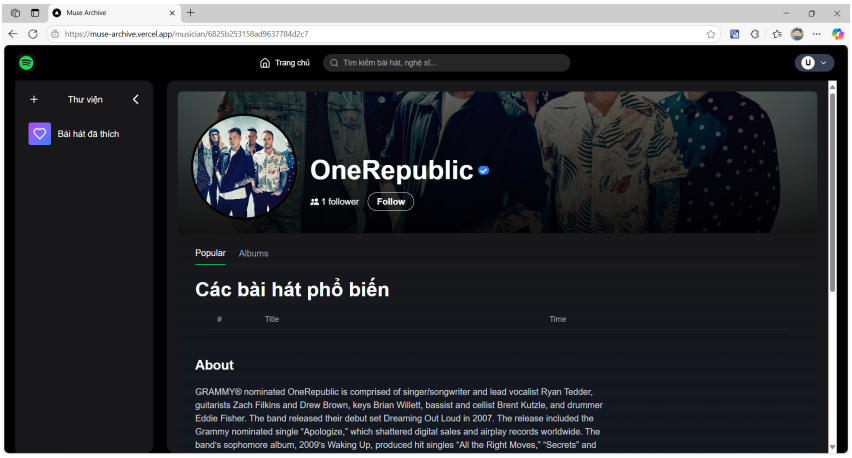
Hình 3: Giao diện trang chủ



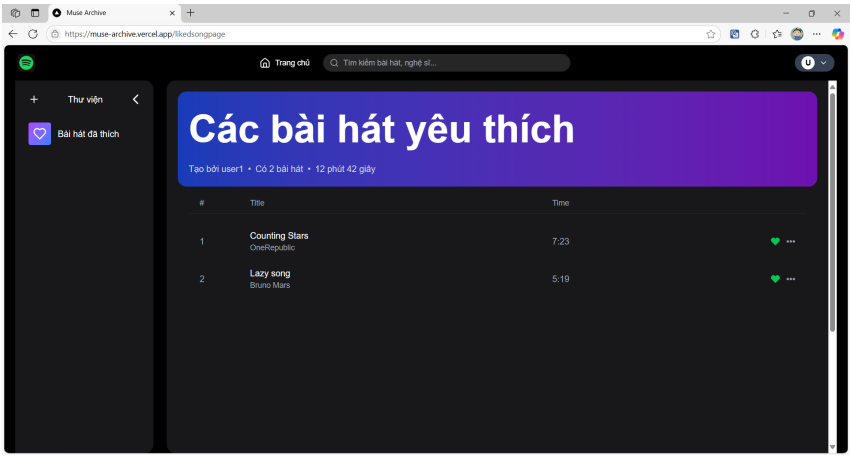
Hình 4: Giao diện trang bài hát



Hình 5: Giao diện trang Album



Hình 6: Giao diện trang nghệ sĩ



Hình 7: Giao diện trang bài hát yêu thích

5 Cài đặt và triển khai hệ thống

5.1 Môi trường yêu cầu

- Node.js v20+
- Python 3

5.2 Hướng dẫn chạy ứng dụng

5.2.1 Chạy Frontend

1. Clone project frontend từ GitHub
2. Cài đặt package frontend: `npm install`
3. Chạy frontend: `npm run dev`

5.2.2 Chạy Backend

1. Clone project backend từ GitHub: git clone https://github.com/zdnghost/django-spotify.git
2. Tạo môi trường ảo:

   ```
   python –m venv env
   ```

3. Kích hoạt môi trường ảo:

   ```
   source env/bin/activate
   ```

4. Cài đặt các package cần thiết:

   ```
   pip install –r requirements.txt
   ```

5. Chạy backend:

   ```
   python manage.py runserver
   ```

6 Kết luận

Dự án "Trang web nghe nhạc Spotify" đã hoàn thành các mục tiêu cơ bản về xây dựng một nền tảng nghe nhạc trực tuyến với giao diện thân thiện, tính năng đa dạng và hiệu năng ổn định. Việc sử dụng các công nghệ hiện đại như ReactJS, Next.js cho frontend và Django cùng MongoDB cho backend đã giúp tạo ra một hệ thống có khả năng mở rộng, dễ bảo trì và vận hành hiệu quả.

6.1 Ưu điểm của dự án:

Giao diện người dùng thân thiện, trực quan, giúp người dùng dễ dàng thao tác và tận hưởng trải nghiệm nghe nhạc, xem video.

Khả năng cá nhân hóa cao, cho phép người dùng tạo album, lưu danh sách yêu thích, tải video về thiết bị.

Bảo mật tốt với cơ chế xác thực JWT, bảo vệ dữ liệu người dùng và thông tin đăng nhập.

Hiệu suất ổn định nhờ tối ưu API và sử dụng S3 Bucket để quản lý tài nguyên đa phương tiện.

Kiến trúc rõ ràng, tách biệt front-end và back-end, thuận tiện cho việc phát triển và mở rộng trong tương lai.

6.2 Nhược điểm và hạn chế:

Chưa hoàn thiện các tính năng nâng cao như gợi ý bài hát dựa trên AI, chưa hỗ trợ đa nền tảng (ứng dụng di động).

Chưa tối ưu hoàn toàn cho đa luồng phát nhạc đồng thời, gây giới hạn trong trải nghiệm người dùng khi nghe nhiều bài cùng lúc.

Chưa có các tính năng xã hội hoặc tương tác người dùng, hạn chế khả năng kết nối cộng đồng người nghe.

Một số phần giao diện và trải nghiệm người dùng cần tinh chỉnh thêm để tăng tính hấp dẫn và tiện ích.

Trong tương lai, dự án có thể tiếp tục phát triển theo hướng mở rộng tính năng, tích hợp trí tuệ nhân tạo để cá nhân hóa hơn nữa, xây dựng ứng dụng di động và cải thiện khả năng tương tác người dùng. Đồng thời, việc hoàn thiện thêm về hiệu năng và bảo mật sẽ giúp hệ thống đáp ứng tốt hơn nhu cầu ngày càng đa dạng của người dùng.

Tóm lại, dự án không chỉ đạt được các mục tiêu kỹ thuật đề ra mà còn là bước khởi đầu vững chắc giúp nhóm phát triển kỹ năng, kinh nghiệm thực tiễn trong lĩnh vực phát triển ứng dụng web đa phương tiện, tạo nền tảng để tiến tới các dự án phức tạp và quy mô hơn trong tương lai.

7 Tài liệu tham khảo

[1] ReactJS documentation: `https://reactjs.org`

[2] Django documentation: `https://www.djangoproject.com/`