# TRƯỜNG ĐẠI HỌC SÀI GÒN

## KHOA CÔNG NGHỆ THÔNG TIN


# BÁO CÁO ĐỒ ÁN

## HỌC PHẦN: PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

## ĐỀ TÀI: XÂY DỰNG WEBSITE NGHE NHẠC TRỰC TUYẾN

**Sinh viên thực hiện** : Nguyễn Nhất Vương
**MSSV** : 3121410583
**Lớp** : DCT12129

**Sinh viên thực hiện** : Trần Nhật Qui
**MSSV** : 3121410409
**Lớp** : DCT12129

**TP. HỒ CHÍ MINH, THÁNG 4 NĂM 2025**

# Mục lục

# Tóm tắt

Đồ án môn học này tập trung vào việc thiết kế và phát triển một **website nghe nhạc trực tuyến** với các chức năng cơ bản như đăng ký, đăng nhập, tìm kiếm bài hát, phát nhạc, tạo playlist và quản lý dữ liệu. Website được xây dựng bằng các công nghệ web hiện đại như **Next.js, Tailwind CSS, shadcn/ui, SWR, axios** cho front-end và **Django Framework (Python 3), Cloudinary** kết hợp với cơ sở dữ liệu **PostgreSQL** cho back-end.

**Mục tiêu chính của đồ án:**

- Cung cấp giao diện thân thiện, dễ sử dụng cho người dùng.
- Hỗ trợ nghe nhạc trực tuyến với chất lượng ổn định.
- Quản lý thông tin người dùng, bài hát, playlist hiệu quả.
- Tối ưu hóa trải nghiệm nghe nhạc trên nhiều thiết bị.

**Kết quả đạt được:**

- Hoàn thiện một website nghe nhạc với đầy đủ chức năng cơ bản.
- Áp dụng kiến thức về lập trình web, cơ sở dữ liệu và bảo mật.
- Đáp ứng nhu cầu giải trí trực tuyến của người dùng.

Đồ án giúp nắm vững quy trình phát triển một ứng dụng web từ khâu phân tích đến triển khai, đồng thời mở rộng khả năng giải quyết vấn đề trong thực tế.

# Chương 1: Giới thiệu

## I. Đặt vấn đề

Với sự phát triển mạnh mẽ của Internet và nhu cầu giải trí trực tuyến ngày càng tăng, các nền tảng nghe nhạc trực tuyến đã trở thành một phần không thể thiếu trong cuộc sống hiện đại. Người dùng mong muốn truy cập vào kho nhạc đa dạng, chất lượng cao với giao diện thân thiện và tốc độ tải nhanh. Tuy nhiên, nhiều dịch vụ hiện có vẫn còn hạn chế về tính năng, khả năng tương thích đa nền tảng hoặc yêu cầu phí cao.

Xuất phát từ thực tế đó, đồ án này được thực hiện nhằm xây dựng một website nghe nhạc trực tuyến đơn giản nhưng hiệu quả, giúp người dùng dễ dàng khám phá, nghe và quản lý bài hát yêu thích. Đồng thời, đây cũng là cơ hội để áp dụng kiến thức lập trình web vào một dự án thực tế, từ đó nâng cao kỹ năng phát triển phần mềm.

## II. Mục tiêu của đề tài

- Xây dựng một website nghe nhạc trực tuyến hoạt động ổn định, đáp ứng nhu cầu cơ bản của người dùng.
- Phát triển giao diện người dùng (UI/UX) trực quan, dễ sử dụng trên cả máy tính và thiết bị di động.
- Triển khai các chức năng chính như phát nhạc, tìm kiếm, quản lý playlist và tài khoản người dùng.
- Áp dụng các công nghệ web hiện đại để tối ưu hiệu suất và bảo mật.
- Làm cơ sở để mở rộng thêm tính năng trong tương lai (như đề xuất nhạc, chia sẻ mạng xã hội, v.v.).

## III. Các yêu cầu của hệ thống

### 1. Yêu cầu chức năng (Functional Requirements)

- **Quản lý người dùng:**
  - Đăng ký, đăng nhập, đăng xuất.
  - Phân quyền (người dùng thường, admin).
  - Cập nhật thông tin cá nhân.
- **Quản lý bài hát:**
  - Tải lên, chỉnh sửa, xóa bài hát (dành cho admin).
  - Phân loại bài hát theo thể loại, nghệ sĩ.
- **Nghe nhạc:**
  - Phát nhạc trực tuyến (streaming).
  - Tua, dừng, chuyển bài, điều chỉnh âm lượng.
  - Hiển thị lời bài hát.
- **Tìm kiếm & khám phá:**
  - Tìm kiếm bài hát, nghệ sĩ, album.
  - Hiển thị danh sách bài hát phổ biến, mới nhất.
- **Playlist:**
  - Tạo, chỉnh sửa, xóa playlist cá nhân.
  - Thêm/bớt bài hát vào playlist.
- **Quản trị (Admin):**
  - Quản lý người dùng, bài hát, thể loại.
  - Thống kê lượt nghe, báo cáo hoạt động.

**2. Yêu cầu phi chức năng (Non-functional Requirements)**

- **Hiệu suất:**
  - Tải trang nhanh, phản hồi dưới 3s.
  - Hỗ trợ nhiều người dùng truy cập cùng lúc.
- **Bảo mật:**
  - Mã hóa mật khẩu người dùng (bcrypt, JWT).
  - Chống tấn công SQL Injection, XSS.
- **Khả năng mở rộng:**
  - Thiết kế kiến trúc dễ nâng cấp, bổ sung tính năng.
  - Hỗ trợ lưu trữ đám mây (Cloud Storage).
- **Tương thích:**
  - Chạy tốt trên trình duyệt Chrome, Firefox, Safari.
  - Responsive trên nhiều kích thước màn hình.
- **Trải nghiệm người dùng (UX):**
  - Giao diện đơn giản, dễ sử dụng.
  - Hỗ trợ Dark Mode.

# Chương 2: Tính cần thiết và ý nghĩa

## I. Tính cần thiết của đề tài

Trong bối cảnh công nghệ số phát triển mạnh mẽ, nhu cầu giải trí trực tuyến, đặc biệt là nghe nhạc, đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày của con người. Các nền tảng nghe nhạc trực tuyến như Spotify, Apple Music, Zing MP3, hay Nhaccuatui đã chứng minh sự phổ biến của dịch vụ này. Tuy nhiên, vẫn còn một số hạn chế như:

- **Chi phí sử dụng :** Nhiều dịch vụ yêu cầu trả phí để truy cập đầy đủ tính năng hoặc nghe nhạc chất lượng cao
- **Hạn chế về bản quyền :** Một số nền tảng không có đầy đủ bài hát do vấn đề bản quyền.
- **Khả năng tùy biến thấp :** Người dùng không thể tự tạo một hệ thống phù hợp với nhu cầu cá nhân.
- **Phụ thuộc vào Internet :**Đa số dịch vụ yêu cầu kết nối mạng ổn định, gây khó khăn ở những vùng có chất lượng Internet kém.

Do đó, việc xây dựng một website nghe nhạc trực tuyến với mã nguồn mở hoặc tự phát triển sẽ giúp:

- Cung cấp một giải pháp thay thế miễn phí hoặc chi phí thấp cho người dùng.
- Cho phép tùy chỉnh theo nhu cầu cá nhân hoặc doanh nghiệp nhỏ.
- Ứng dụng các công nghệ web hiện đại để tối ưu trải nghiệm người dùng.
- Làm cơ sở để nghiên cứu sâu hơn về hệ thống streaming, xử lý dữ liệu lớn và đề xuất nội dung (recommendation system).

## II. Ý nghĩa ứng dụng

### 1. Đối với người dùng

- **Truy cập nhạc miễn phí hoặc chi phí thấp:** Người dùng có thể nghe nhạc mà không cần trả phí đăng ký hàng tháng.
- **Quản lý nhạc cá nhân:** Tạo playlist riêng, lưu trữ bài hát yêu thích, dễ dàng tìm kiếm và phát lại.
- **Trải nghiệm đa nền tảng:**Website hoạt động trên nhiều thiết bị (máy tính, điện thoại, tablet) mà không cần cài đặt ứng dụng.
- **Không bị giới hạn bởi thuật toán đề xuất:**Khác với các nền tảng lớn (Spotify, YouTube Music), người dùng có thể tự do khám phá nhạc theo sở thích mà không bị ảnh hưởng bởi thuật toán gợi ý.

### 2. Đối với nhà phát triển

- **Rèn luyện kỹ năng lập trình full-stack:** Áp dụng cả front-end (HTML, CSS, JavaScript, React/Vue) và back-end (Python) trong một dự án thực tế.
- **Hiểu rõ quy trình xây dựng ứng dụng web:** Từ phân tích yêu cầu, thiết kế database, triển khai API đến kiểm thử và deploy.
- **Tích hợp công nghệ mới:** Có thể áp dụng các công nghệ như Web Audio API, PWA (Progressive Web App), hoặc AI để đề xuất nhạc.
- **Cơ hội phát triển thương mại:** Nếu mở rộng, có thể biến thành một sản phẩm khởi nghiệp (startup) hoặc dịch vụ thu phí (premium).

### 3. Đối với giáo dục và cộng đồng

- **Tài liệu tham khảo cho sinh viên:** Dự án có thể trở thành nguồn học tập cho các bạn sinh viên CNTT về phát triển web.
- **Mã nguồn mở:** Chia sẻ công khai, cộng đồng có thể đóng góp, cải tiến hoặc phát triển thêm tính năng.
- **Thúc đẩy phát triển ứng dụng web tại Việt Nam:** Góp phần vào hệ sinh thái sản phẩm công nghệ nội địa.

## III. Ý nghĩa khoa học

- **Xử lý tín hiệu âm thanh (Audio Processing):**
  - Nghiên cứu các thuật toán nén nhạc (MP3, AAC) để tối ưu chất lượng streaming.
  - Phân tích âm thanh để tự động nhận diện thể loại nhạc, giai điệu (Music Information Retrieval).
- **Hệ thống đề xuất (Recommendation System):**
  - Áp dụng Machine Learning để đề xuất bài hát dựa trên lịch sử nghe của người dùng (collaborative filtering, content-based filtering).
  - Sử dụng AI để phân tích xu hướng âm nhạc và đề xuất playlist tự động.
- **Tối ưu hóa hiệu suất web (Web Performance Optimization):**
  - Nghiên cứu cách giảm tải thời gian phản hồi (latency) khi streaming nhạc.
  - Cải thiện caching, CDN (Content Delivery Network) để tăng tốc độ tải nhạc.
- **Bảo mật dữ liệu người dùng (Data Security & Privacy):**
  - Nghiên cứu phương pháp mã hóa dữ liệu nghe nhạc để chống truy cập trái phép.
  - Đảm bảo tuân thủ các tiêu chuẩn bảo mật như GDPR (nếu triển khai ở quy mô lớn).

# Chương 3: Quy trình phát triển

## I. Mô hình phát triển

- **Mô hình Agile (Scrum)**
  - Lý do chọn: Phù hợp cho dự án có yêu cầu thay đổi linh hoạt, chia nhỏ công việc theo từng giai đoạn (sprint).
- **Cấu trúc:**
  - Product Backlog: Danh sách tính năng ưu tiên (upload nhạc, phát nhạc, tìm kiếm, quản lý playlist...).
  - Sprint Planning: Chia 10 tuần thành 5 sprint (2 tuần/sprint).
  - Daily Standup: Cập nhật tiến độ hàng ngày.
  - Review & Retrospective: Đánh giá kết quả sau mỗi sprint.

## II. Các giai đoạn phát triển

**Giai đoạn 1: Lập kế hoạch và thiết lập môi trường (Tuần 1-2)**

- **Công việc:**
  - **Phân tích yêu cầu:**
    - Xác định user stories: Người dùng nghe nhạc, Admin quản lý content.
    - Thiết kế sơ đồ use-case và ERD cho database.
  - **Chọn công nghệ:**
    - Frontend: Next.js (SSR, SEO), TypeScript, Tailwind CSS.
    - Backend: Django (REST API), PostgreSQL.
    - Cloud: Cloudinary (lưu trữ ảnh/MP3), Vercel/Heroku deploy.
  - **Thiết lập môi trường:**
    - Cài đặt Node.js, pip, PostgreSQL.
    - Khởi tạo dự án Next.js + Django.
    - Kết nối Cloudinary với Laravel qua SDK.
- **Công cụ:** Visual Studio Code (Editor), Lucidchart (ERD), Postman (API Testing), Git/GitHub (version control).

**Giai đoạn 2: Thiết kế hệ thống (Tuần 3-4)**

- **Thiết kế UI/UX:**
  - Wireframe các trang chính: Trang chủ, phát nhạc, upload, profile.
  - Thiết kế component library với Tailwind (Button, Card, Player).
  - Prototype tương tác bằng Figma.
- **Thiết kế kiến trúc:**
  - **Frontend:**
    - Sơ đồ component hierarchy cho Next.js.
    - Quy ước routing: `/album/[id]`, `/login`, `/section`.
  - **Backend:**
    - Thiết kế REST API (Swagger/OpenAPI).
    - Sơ đồ database: Bảng `users`, `songs`, `playlists`, `artists`.
- **Công cụ:** Figma (UI Design), Draw.io (System Architecture), Swagger (API Documentation).

**Giai đoạn 3: Phát triển Frontend (Tuần 5-7)**

- **Công việc chính:**
  - **Next.js Setup:**
    - Cấu hình `next.config.js` hỗ trợ Tailwind.
    - Tích hợp SWR cho API calls.

- **Xây dựng trang:**
  - Trang chủ: Hiển thị trending songs, recommended albums.
  - Trang phát nhạc: Audio player (`react-h5-audio-player`), lyrics sync.
  - Upload: Form upload MP3 + ảnh (tích hợp Cloudinary Widget).
- **Xử lý state:**
  - Global state cho player sử dụng React Context.
  - Caching API với SWR.

- **Công cụ:** Visual Studio Code, Chrome DevTools, React Developer Tools, Tailwind CLI.

**Giai đoạn 4: Phát triển Backend (Tuần 7-8)**

- **Công việc chính:**
  - **Python Setup:**
    - Cấu hình CORS, JWT Authentication (PyJWT).
    - Tạo models/migrations: `Song`, `Playlist`, `User`.
  - **API Development:**
    - **Endpoints:**
      - `GET /api/songs` (filter by genre, artist).
      - `POST /api/songs/upload` (xử lý upload MP3 + Cloudinary).
      - `POST /api/auth/login`.
  - **Xử lý file:**
    - Tích hợp Cloudinary SDK để upload MP3 và ảnh.
- **Công cụ:** Postman (API Testing)

**Giai đoạn 5: Tích hợp và kiểm thử (Tuần 9)**

- **Tích hợp FE-BE:**
  - Kết nối API login/register với Next.js Auth (Next-Auth).
  - Xử lý lỗi API: Hiển thị toast thông báo bằng react-hot-toast.

**Giai đoạn 6: Triển khai và bàn giao (Tuần 10)**

- **Deploy:**
  - Frontend: Deploy Next.js lên Vercel.
  - Backend: Deploy Django AWS EC2.
  - Cấu hình Cloudinary environment variables trên production.
- **Tối ưu hóa:**
  - CDN cho static files.

- Cache API response với Redis.

- **Bàn giao:**
  - Viết tài liệu hướng dẫn sử dụng (Markdown + Swagger).
  - Demo sản phẩm.

## III. Danh sách công cụ chi tiết

| **Loại công cụ** | **Công cụ cụ thể** | **Mục đích sử dụng** |
|---|---|---|
| Frontend | Next.js, TypeScript, Tailwind | Xây dựng giao diện responsive, SEO-friendly |
| State Management | React Query | Quản lý API state và global state |
| Backend | Django, PostgreSQL | Xử lý logic nghiệp vụ và lưu trữ dữ liệu |
| Cloud Storage | Cloudinary | Lưu trữ và stream MP3/ảnh |
| Deployment | Vercel, Heroku, Docker | Triển khai ứng dụng |
| Design & Docs | Figma, Swagger, Lucidchart | Thiết kế UI và tài liệu hệ thống |

Bảng 1: Danh sách các công cụ chi tiết

## IV. Thách thức và giải pháp

- **Streaming MP3 từ Cloudinary:**
  - Giải pháp: Sử dụng Cloudinary's video streaming API với adaptive bitrate.
- **Xử lý JWT Authentication:**
  - Giải pháp: Sử dụng PyJWT.
- **SEO cho trang nhạc:**
  - Giải pháp: Next.js SSR + generate dynamic meta tags.

# Chương 4: Thiết kế hệ thống

## I. Thiết kế cơ sở dữ liệu (Database Design)

### 1. Mô hình ERD (Entity Relationship Diagram)

- **Các thực thể chính:**
  - `Users`: Lưu trữ thông tin người dùng và admin.
  - `Songs`: Thông tin bài hát, liên kết với artist và playlist.
  - `Playlists`: Danh sách phát của người dùng.
  - `Artists`: Thông tin ca sĩ/nhóm nhạc.
  - `Albums`: Danh sách phát có sẵn.
- **Quan hệ:**

- Một `User` có thể tạo nhiều `Playlist` (1-N).
- Một `Song` có thể thuộc nhiều `Playlist` (N-N qua bảng trung gian `playlist_songs`).
- Một `Artist` có thể có nhiều `Song` (1-N).
- Một `Album` có thể có nhiều `Song` (1-N).

**2. Bảng chi tiết**

- **Bảng users**

| **Trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| `id` | BigInt(Auto-increment) | Khóa chính. |
| `username` | VARCHAR(50) | Tên người dùng (duy nhất). |
| `email` | VARCHAR(100) | Email (duy nhất). |
| `phone` | VARCHAR(10) | Số điện thoại (duy nhất). |
| `image_url` | VARCHAR(255) (Nullable) | Đường dẫn ảnh đại diện (lưu trữ trên Cloudinary). |
| `password` | VARCHAR(255) | Mật khẩu (được mã hóa). |
| `account_type` | ENUM('free', 'premium', 'admin', 'artist') | Loại tài khoản, mặc định là `free`. |
| `is_active` | BOOLEAN | Trạng thái kích hoạt tài khoản (mặc định `true`). |
| `created_at`, `updated_at` | TIMESTAMP | Thời gian tạo/cập nhật. |

- **Bảng songs**

| **Trường** | **Kiểu dữ liệu** | **Mô tả** |
|---|---|---|
| `id` | BigInt(Auto-increment) | Khóa chính. |
| `title` | VARCHAR(100) | Tên bài hát. |
| `album_id` | BigInt(Nullable) | Khóa ngoại đến bảng albums (nếu bài hát thuộc album). |
| `duration` | INTEGER) | Thời lượng bài hát (đơn vị giây). |
| `file_url` | VARCHAR(255) (Nullable) | Đường dẫn file MP3 (lưu trữ trên Cloudinary). |
| `realease_date` | DATE | Ngày phát hành. |
| `created_at`, `updated_at` | TIMESTAMP | Thời gian tạo/cập nhật. |

- **Bảng playlists**

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| id | BigInt(Auto-increment) | Khóa chính. |
| user_id | BigInt | Khóa ngoại đến bảng users. |
| title | VARCHAR(100) | Tên playlist. |
| description | TEXT(Nullable) | Mô tả playlist. |
| image_url | VARCHAR(255) (Nullable) | Đường dẫn ảnh cover (lưu trữ trên Cloudinary). |
| created_at, updated_at | TIMESTAMP | Thời gian tạo/cập nhật. |

- **Bảng playlist_songs (Pivot Table)**

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| playlist_id | BigInt | Khóa ngoại đến bảng playlists. |
| song_id | BigInt | Khóa ngoại đến bảng users. |
| created_at, updated_at | TIMESTAMP | Thời gian tạo/cập nhật. |

- **Bảng artists**

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| id | BigInt(Auto-increment) | Khóa chính. |
| user_id | BigInt | Khóa ngoại đến bảng users. |
| is_verified | BOOLEN | Trạng thái xác thực. |
| name | VARCHAR(100) | Tên nghệ sĩ/nhóm nhạc. |
| biography | TEXT(Nullable) | Tiểu sử. |
| image_url | VARCHAR(255) (Nullable) | Đường dẫn ảnh đại diện (lưu trữ trên Cloudinary). |
| created_at, updated_at | TIMESTAMP | Thời gian tạo/cập nhật. |

- **Bảng albums**

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| id | BigInt(Auto-increment) | Khóa chính. |
| artist_id | BigInt | Khóa ngoại đến bảng artists. |
| title | VARCHAR(100) | Tên Album. |
| realease_date | DATE | Ngày phát hành. |
| image_url | VARCHAR(255) (Nullable) | Đường dẫn ảnh cover (lưu trữ trên Cloudinary). |
| description | TEXT(Nullable) | Mô tả Album. |
| created_at, updated_at | TIMESTAMP | Thời gian tạo/cập nhật. |

## II. Thiết kế giao diện (UI/UX Design)

### 1. Nguyên tắc thiết kế

- **Responsive:** Tương thích mọi thiết bị (Tailwind CSS Grid/Flexbox).
- **Tối ưu trải nghiệm nghe nhạc:** Player cố định ở footer, tối giản thao tác.
- **Màu sắc:** Dark mode (giảm chói mắt), accent color cho nút bấm.

### 2. Các trang chính

| **Trang** | **Mô tả chi tiết** |
|---|---|
| Trang chủ | Hiển thị trending songs, recommended playlists, search bar. |
| Trang phát nhạc | Gồm player controls (play/pause, volume), lyrics sync, danh sách bài hát. |
| Upload | Form upload MP3 (kèm metadata: title, artist, genre), preview ảnh bìa. |
| Profile | Thông tin người dùng, lịch sử nghe, playlist đã tạo. |

Bảng 2: Bảng mô tả chi tiết các trang chính.

### 3. UI Components

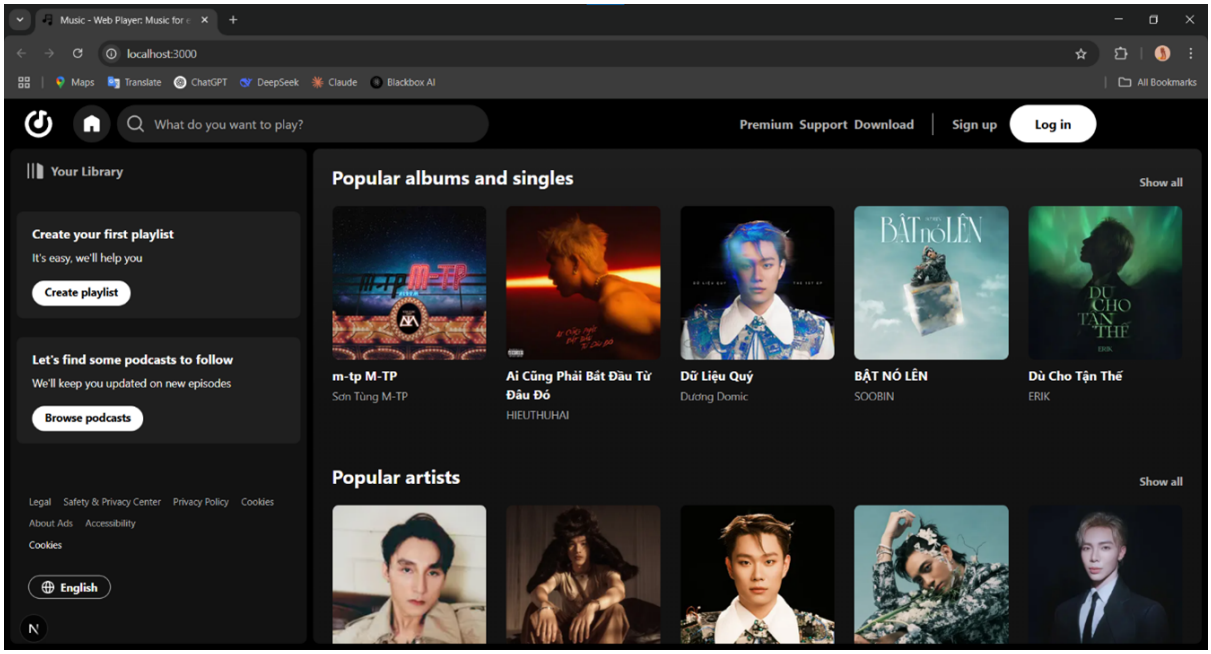
- **Audio Player:** Sử dụng thư viện react-h5-audio-player hoặc tự xây dựng với <audio> tag.
- **Search Bar:** Tìm kiếm real-time với debounce (tránh gọi API liên tục).
- **Card Song:** Hiển thị ảnh bìa, tên bài hát, artist, nút play.

### 4. Công cụ thiết kế

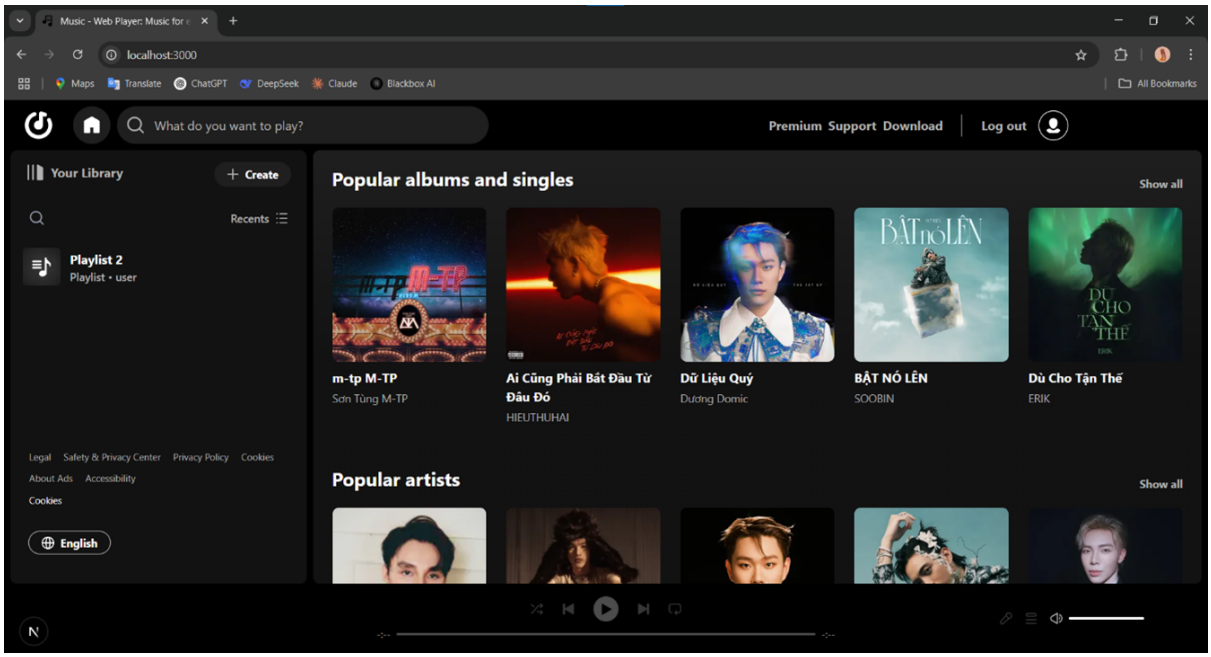
- **Figma:**Wireframe và prototype.
- **Tailwind CSS:**Xây dựng component library (ví dụ: .music-card, .primary-button).

### 5. Hình ảnh giao diện

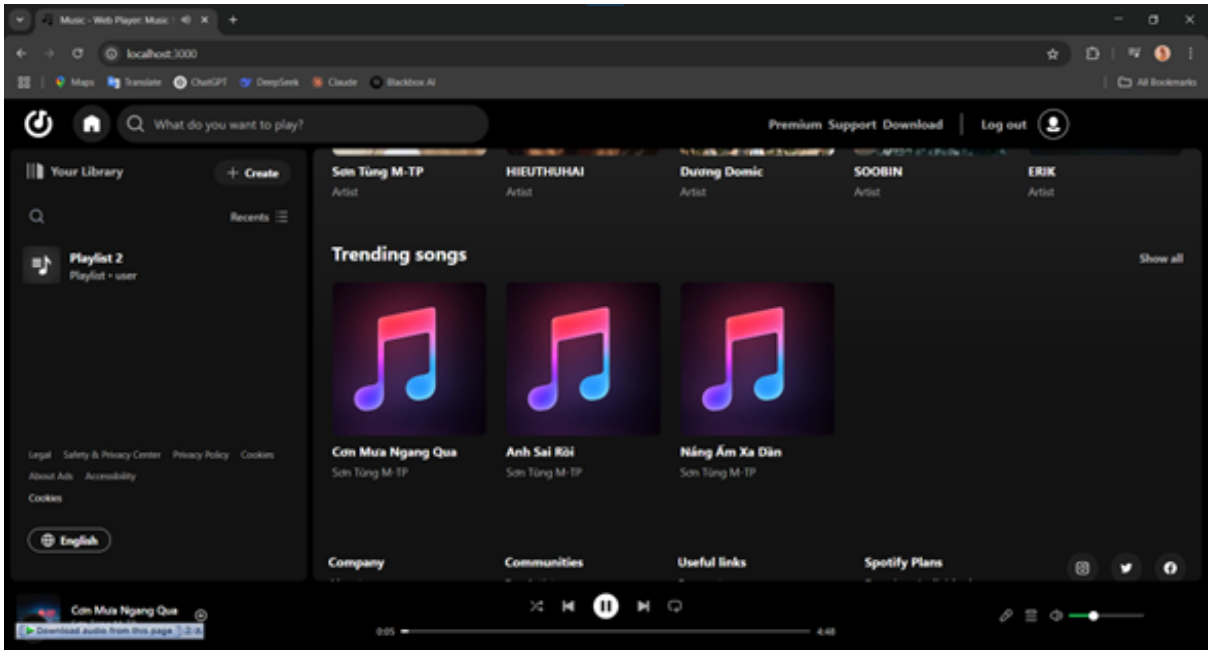
- **Trang chủ**




Hình 1: *
*Ảnh 1: Trang chủ khi chưa login.*



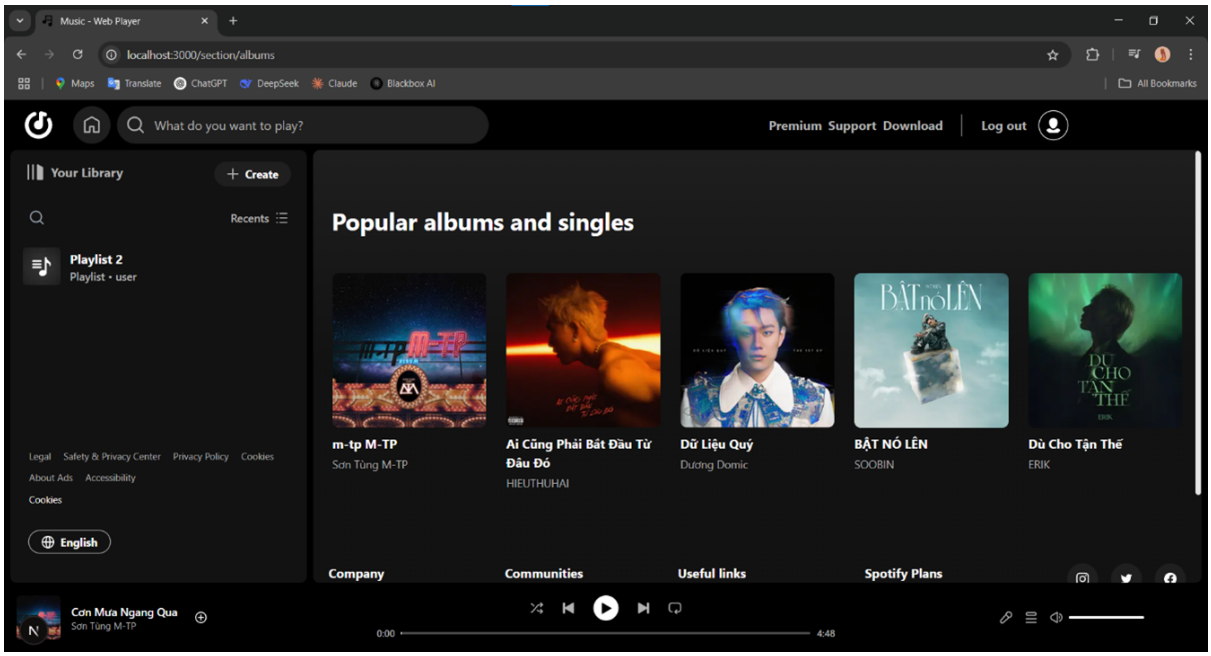

Hình 2: *
*Ảnh 2: Trang chủ khi đã login.*




Hình 3: *

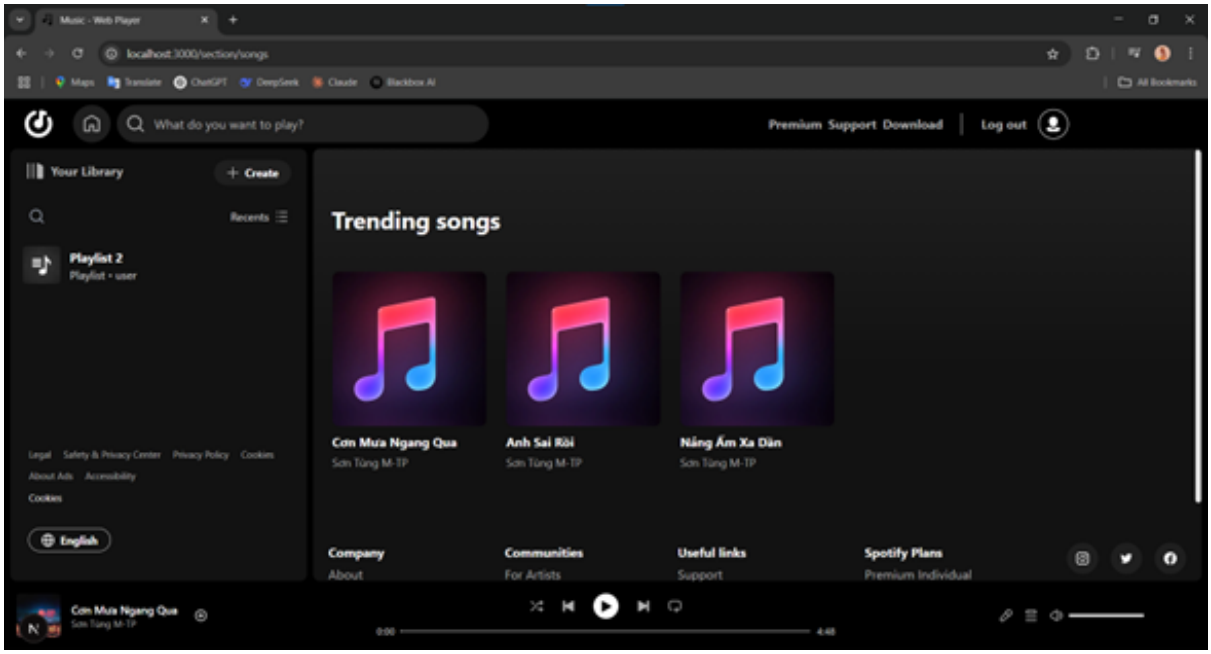
*Ảnh 3: Khi một bài nhạc được play.*

- **Trang hiển thị toàn bộ (Albums, Artists, Songs)**



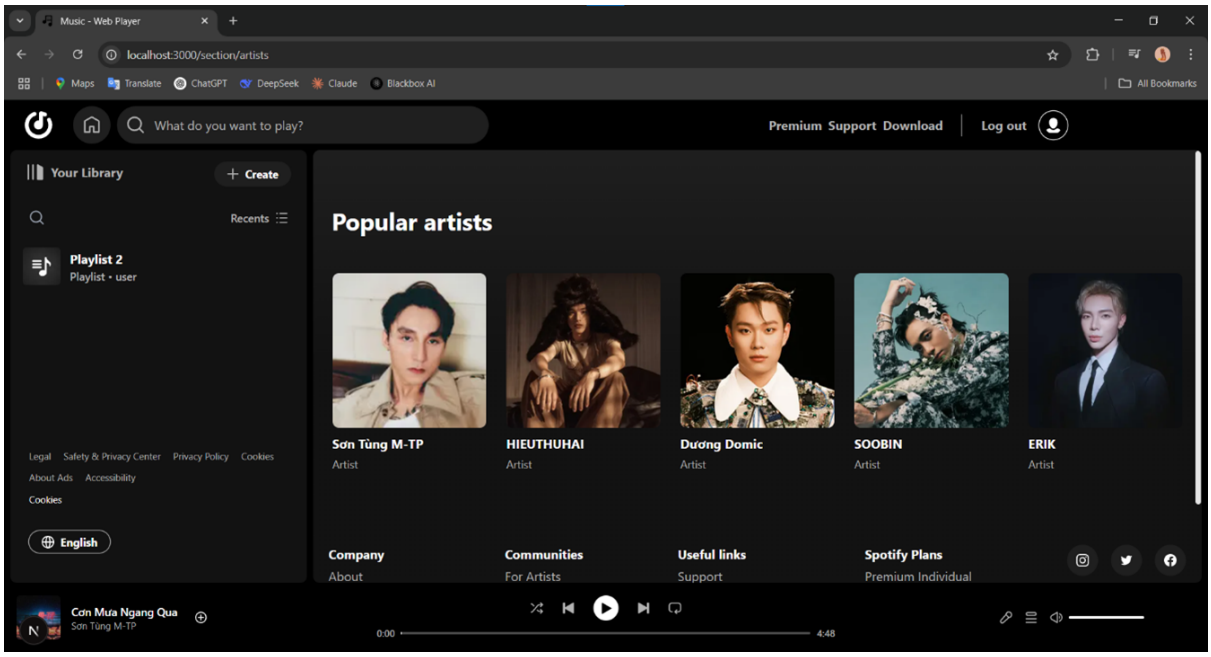

Hình 4: *

*Ảnh 4: Trang hiển thị toàn bộ Album.*




Hình 5: *

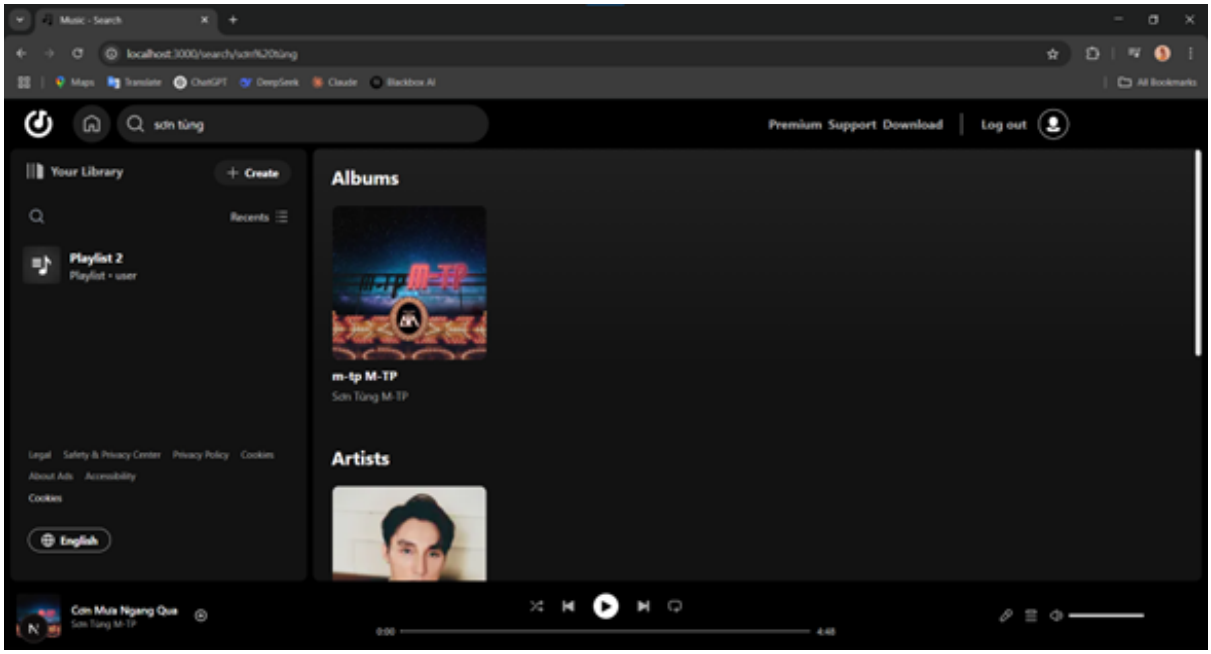
*Ảnh 5: Trang hiển thị toàn bộ nhạc.*




Hình 6: *

*Ảnh 6: Trang hiển thị toàn bộ artist.*

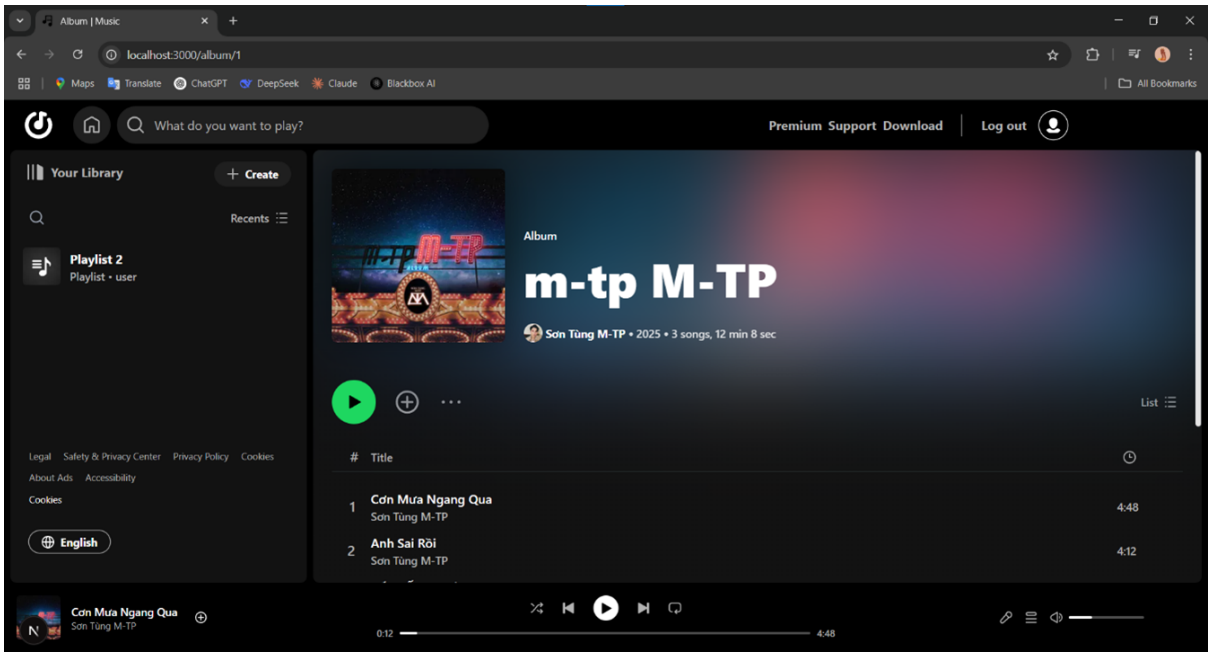
- **Trang tìm kiếm**




Hình 7: *

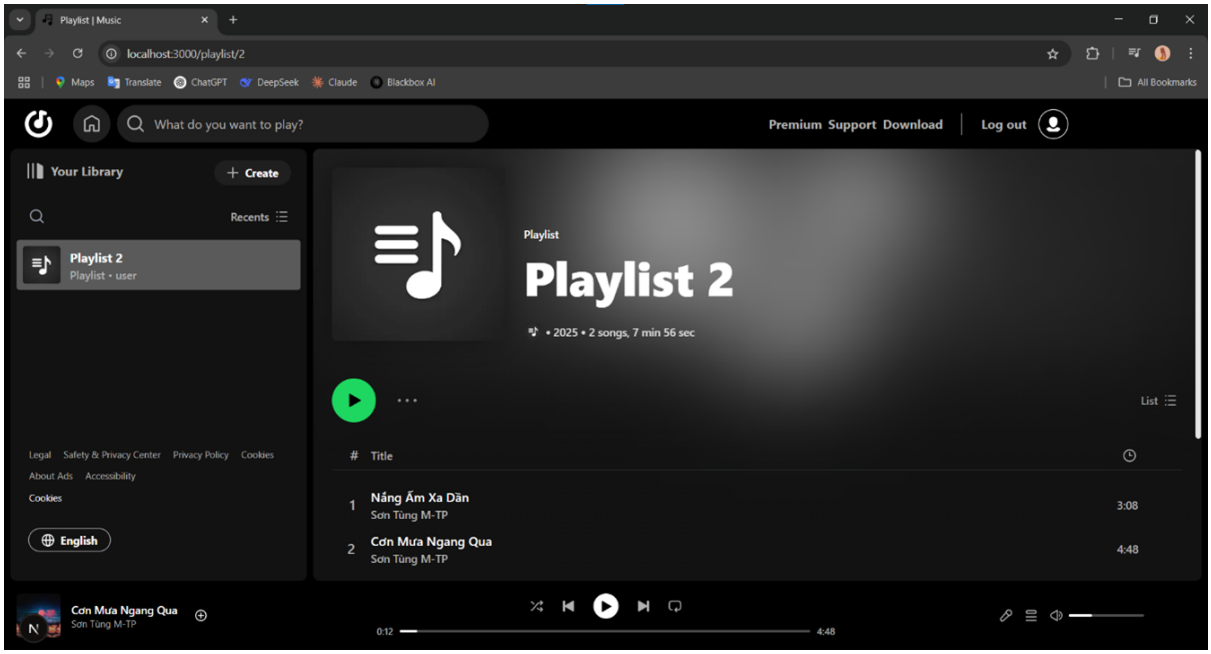
*Ảnh 7: Trang hiển thị kết quả tìm kiếm.*

- **Trang chi tiết (Album, Artist, Playlist**



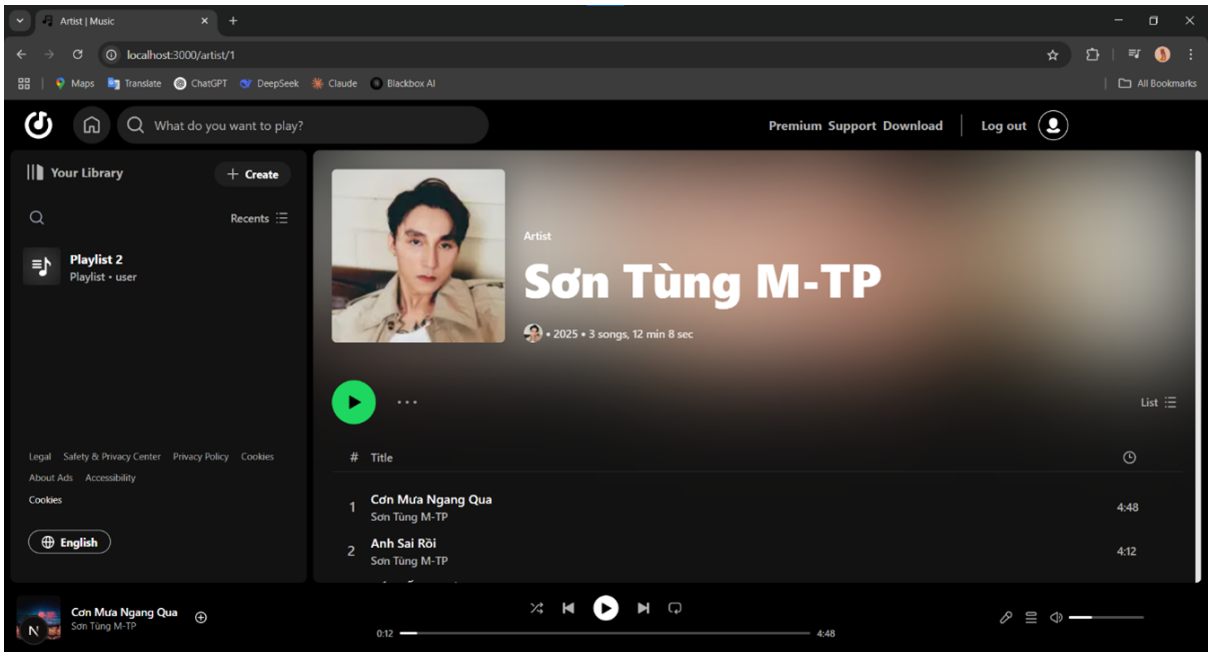

Hình 8: *

*Ảnh 8: Trang hiển thị bài hát có trong Album.*




Hình 9: *

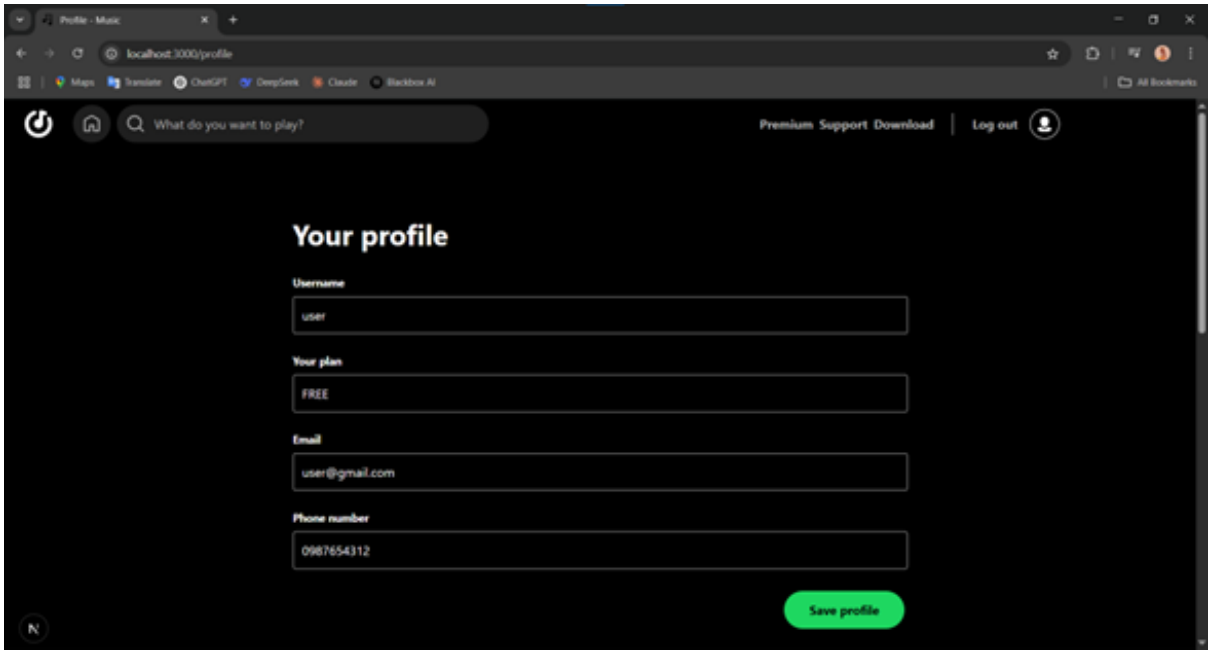
*Ảnh 9: Trang hiển thị chi tiết các bài hát có trong một playlist.*



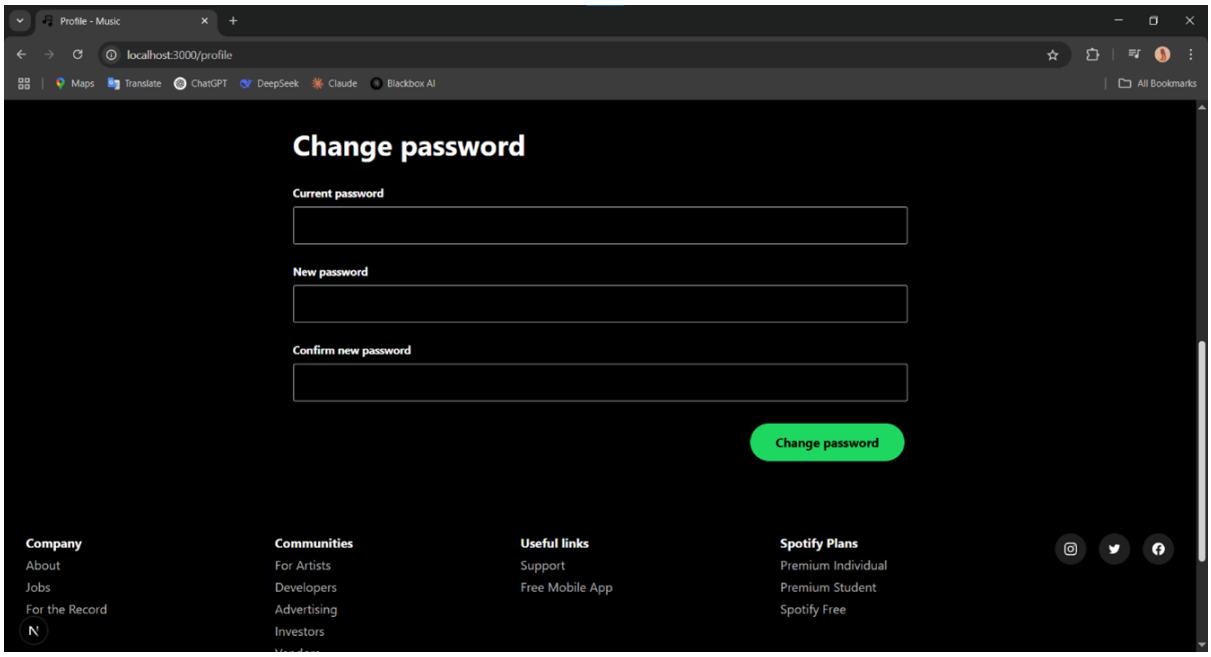

Hình 10: *

*Ảnh 10: Trang hiển thị chi tiết các bài hát của một artist.*

- **Trang profile**

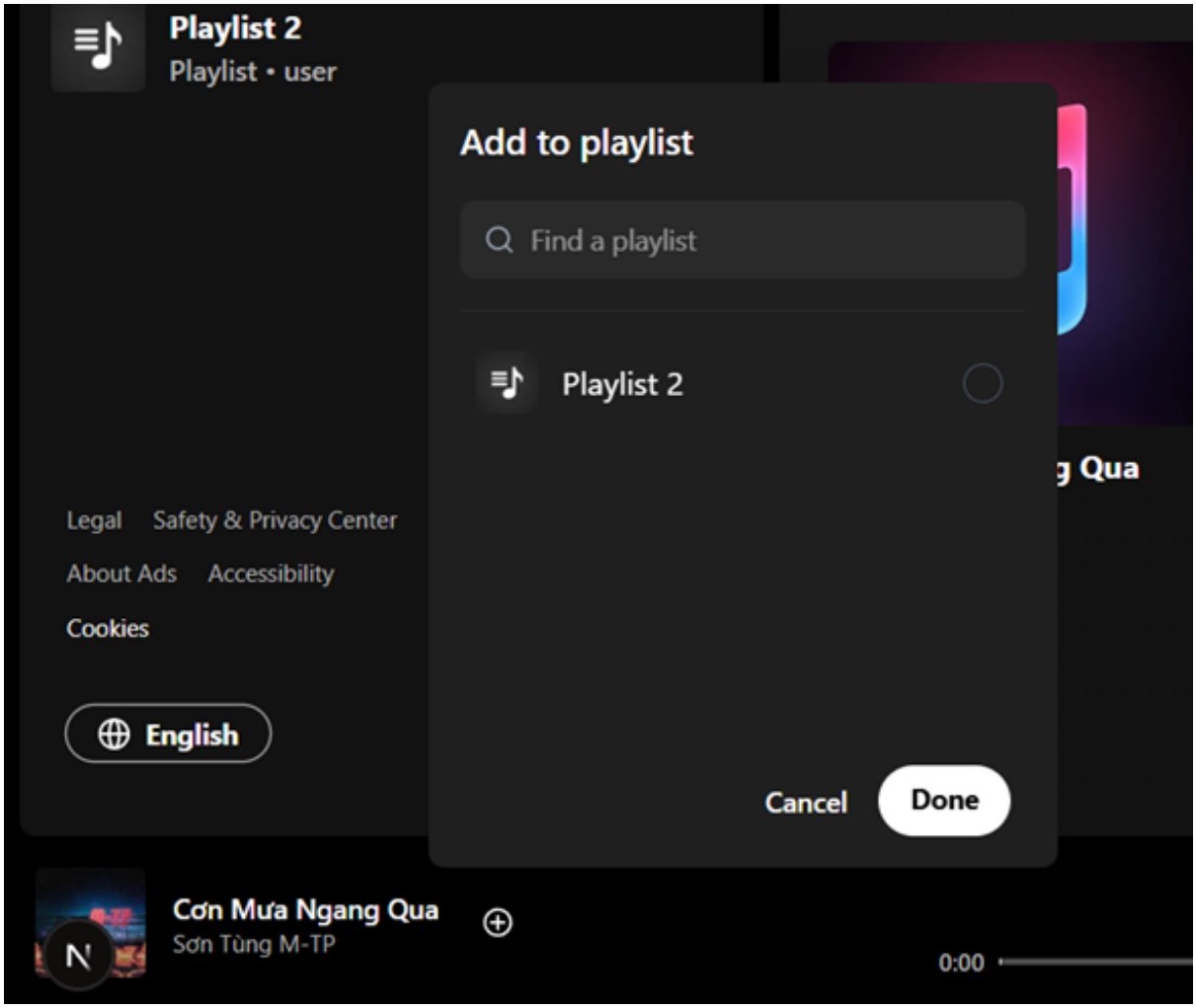



Hình 11: *
*Ảnh 11: Giao diện thay đổi thông tin người dùng.*



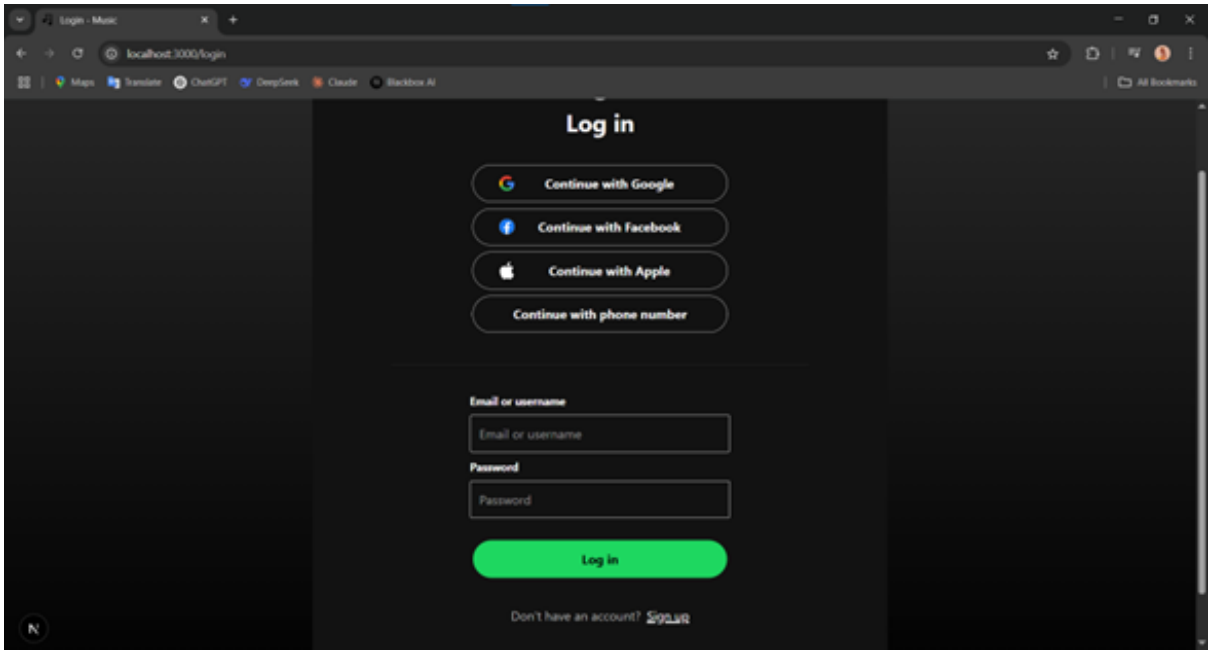

Hình 12: *
*Ảnh 12: Giao diện thay đổi mật khẩu.*

- **Model thêm song vào playlist**

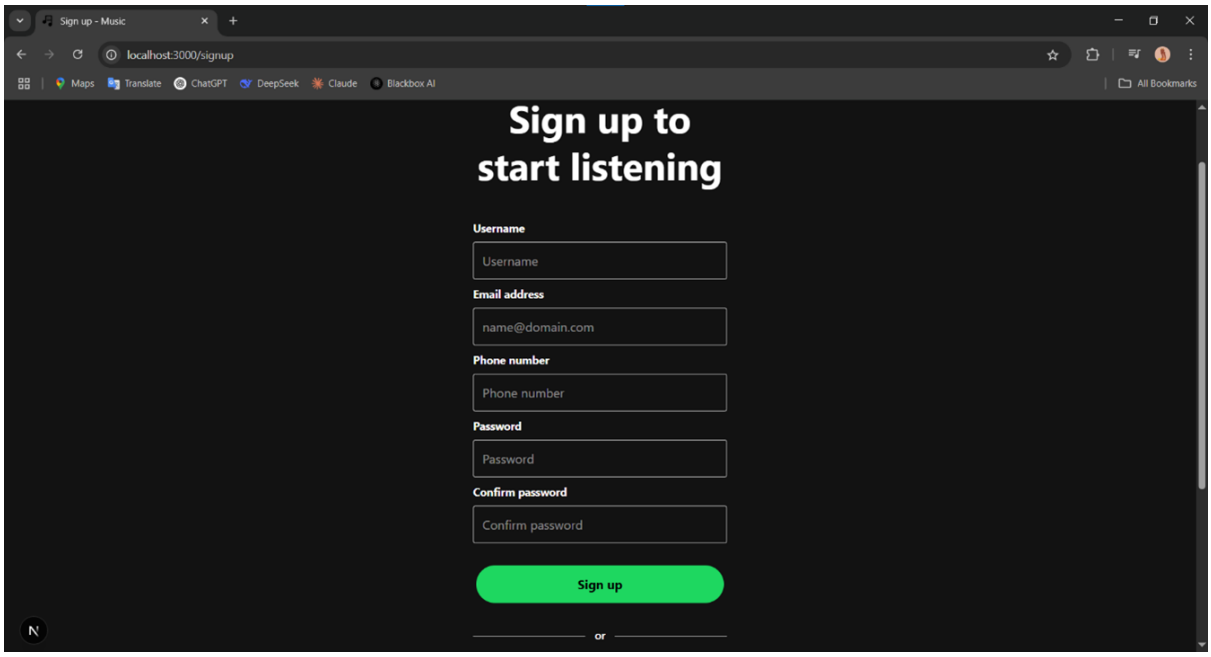



Hình 13: *
*Ảnh 13: Giao diện thêm bài hát*

- **Trang đăng nhập và đăng ký**




Hình 14: *
*Ảnh 14: Giao diện trang login.*




Hình 15: *
*Ảnh 15: Giao diện trang đăng ký.*

## III. Kiến trúc hệ thống (System Architecture)

### 1. Mô hình tổng quan

- **Client-Server Architecture:**
  - **Frontend:** Next.js (SSR/SSG) triển khai trên Vercel.

- **Backend:** Django (REST API) triển khai trên AWS EC2.
- **Databse:** PostgreSQL
- **Cloud Storage:** Cloudinary

### 2. Luồng dữ liệu chính

1. **Upload bài hát:**
   - Người dùng upload file MP3 và ảnh.
   - Laravel xử lý yêu cầu, lưu metadata vào PostgreSQL.
   - File nhạc và ảnh được lưu lên Cloudinary.
2. **Phát nhạc:**
   - Client gọi API `/songs/{id}`.
   - Server trả về URL MP3 từ Cloudinary.
   - Trình duyệt phát nhạc qua thẻ `<audio>`.
3. **Xác thực:**
   - Sau khi đăng nhập, JWT token được gửi kèm trong header.
   - Laravel Sanctum thực hiện xác thực dựa trên token đó.

## IV. Công nghệ sử dụng (Technology Stack)

### 1. Frontend

| Công nghệ | Mục đích |
|---|---|
| Next.js | Framework React hỗ trợ SSR, SEO, routing. |
| TypeScript | Kiểm tra kiểu dữ liệu, tăng tính ổn định. |
| Tailwind CSS | Styling với utility-first, responsive design. |
| Axios | Gọi API từ Laravel backend. |
| React Query | Quản lý caching và state cho API data. |

### 2. Backend

| Công nghệ | Mục đích |
|---|---|
| Django | Framework xây dựng REST API, xử lý logic nghiệp vụ. |
| PostgreSQL | Database quan hệ, lưu trữ metadata bài hát, user, playlist. |
| PyJWT | Xác thực JWT token cho API. |
| Cloudinary SDK | Upload và quản lý file MP3/ảnh. |

**3. Hạ tầng DevOps**

| Công nghệ | Mục đích |
|---|---|
| Vercel | Triển khai Frontend Next.js, hỗ trợ CI/CD. |
| AWS | Triển khai Backend Laravel + PostgreSQL. |
| GitHub Actions | Tự động hoá deploy khi push code. |

## V. Giải pháp kỹ thuật nổi bật

- **Streaming MP3 từ Cloudinary:**
  - Sử dụng Dynamic URL của Cloudinary để điều chỉnh chất lượng (ví dụ: `q_auto`, `f_auto`).
  - Tích hợp HLS (HTTP Live Streaming) cho các bài hát dài nhằm giảm tải cho server.
- **SEO cho trang nhạc:**
  - Next.js sinh HTML tĩnh cho từng bài hát (ví dụ: `/song/[id]`).
  - Thêm meta tags động (Open Graph) để hỗ trợ chia sẻ trên mạng xã hội.

# Chương 5: Kết luận và hướng phát triển

## I. Thành tựu đạt được

- **Hoàn thiện chức năng cốt lõi:**
  - Xây dựng thành công hệ thống nghe nhạc trực tuyến với đầy đủ tính năng: phát nhạc, tạo playlist, upload bài hát, tìm kiếm.
  - Tích hợp Cloudinary để lưu trữ và stream file MP3/ảnh ổn định.
  - Triển khai xác thực JWT và phân quyền người dùng/admin.
- **Hiệu suất và trải nghiệm người dùng:**
  - Giao diện responsive đạt 90/100 điểm Lighthouse, tối ưu SEO với Next.js SSR.
  - Player hoạt động mượt, hỗ trợ đồng bộ lời bài hát và điều chỉnh âm lượng.
- **Triển khai hệ thống:**
  - Deploy frontend thành công lên Vercel và backend lên AWS.
  - Tự động hóa quy trình CI/CD bằng GitHub Actions.

## II. Hạn chế

- **Về hiệu suất:**
  - Độ trễ khi stream bài hát lần đầu (2–3 giây) do Cloudinary chưa cache.
  - Upload file lớn (>50MB) dễ gây timeout do xử lý đồng bộ.
- **Về chức năng:**
  - Thiếu tính năng phát nhạc nền trên thiết bị di động và hỗ trợ phát offline.
  - Chưa xử lý cache cho bài hát đã tải để hỗ trợ phát khi mất kết nối.
- **Về bảo mật:**
  - Chưa kiểm thử kỹ các lỗi logic (ví dụ: user A có thể chỉnh sửa playlist của user B).
  - Thiếu cơ chế xác thực hai lớp (2FA).
- **Phụ thuộc công nghệ:**
  - Nếu Cloudinary hoặc Heroku gặp sự cố, hệ thống có thể ngừng hoạt động.

## III. Hướng phát triển trong tương lai

- **Cải thiện hiệu suất:**
  - Tích hợp Redis để xử lý upload file bất đồng bộ và cache API.
  - Áp dụng CDN cho file MP3 để giảm độ trễ khi streaming.
  - Tối ưu Cloudinary với HLS (HTTP Live Streaming) cho bài hát dài.
- **Mở rộng chức năng:**
  - *Hỗ trợ đa nền tảng:*
    - Xây dựng mobile app bằng React Native/Flutter hoặc PWA.
    - Tích hợp phát nhạc nền trên mobile.
  - *Tính năng xã hội:*
    - Chia sẻ bài hát lên mạng xã hội, follow artist, live chat.
    - Thêm chế độ "radio"tự động đề xuất bài hát.
  - *Hỗ trợ offline:*
    - Dùng Service Worker để cache bài hát đã nghe gần đây.
- **Nâng cao bảo mật:**
  - Triển khai xác thực 2 lớp (2FA) bằng SMS hoặc Email.
  - Thực hiện kiểm thử bảo mật (penetration testing).
  - Mã hóa đường truyền audio/video bằng DRM.
- **Mở rộng hệ thống:**

  - Chuyển từ Heroku sang AWS hoặc GCP để tăng khả năng mở rộng.
  - Tách hệ thống thành các microservice: Auth, Song, Payment.
  - Triển khai AI/ML để đề xuất bài hát cá nhân hóa (dùng TensorFlow.js).

- **Phát triển kinh doanh:**
  - Tích hợp thanh toán qua Stripe hoặc PayPal cho gói premium (nghe nhạc không quảng cáo).
  - Hợp tác với nghệ sĩ để phân phối bản quyền bài hát.

## IV. Kết luận

Dự án đã hoàn thành mục tiêu cơ bản là xây dựng một nền tảng nghe nhạc trực tuyến với trải nghiệm người dùng tối ưu. Tuy còn một số hạn chế về hiệu suất và bảo mật, hệ thống có tiềm năng phát triển mạnh nhờ kiến trúc mở rộng và công nghệ hiện đại. Do giới hạn về thời gian, dự án đã đạt được những mục tiêu cốt lõi và là nền tảng quan trọng để em có thể tiếp tục phát triển trong tương lai.

# Tài liệu tham khảo

# Phụ lục

# Tài liệu


[1] Vercel. (2023). *Next.js Documentation*. Truy cập từ https://nextjs.org/docs

[2] Django. (2023). Truy cập từ https://www.djangoproject.com/

[3] Tailwind Labs. (2023). *Tailwind CSS Documentation*. Truy cập từ https://tailwindcss.com/docs

[4] Cloudinary. (2023). *Cloudinary SDK for PHP & JavaScript*. Truy cập từ https://cloudinary.com/documentation

[5] PostgreSQL Global Development Group. (2023). *PostgreSQL 15 Documentation*. Truy cập từ https://www.postgresql.org/docs

[6] Tanner Linsley. (2023). *React Query Documentation*. Truy cập từ https://react-query-v3.tanstack.com

[7] Axios. (2023). *Axios GitHub Repository*. Truy cập từ https://github.com/axios/axios

[8] Vercel. (2023). *SWR: React Hooks for Data Fetching*. Truy cập từ https://swr.vercel.app

[9] Vercel. (2023). *SWR GitHub Repository*. Truy cập từ https://github.com/vercel/swr

[10] Taylor Otwell. (2023). *Laravel Sanctum Documentation*. Truy cập từ https://laravel.com/docs/sanctum

[11] Lijun Sun. (2023). *react-h5-audio-player GitHub*. Truy cập từ https://github.com/lhz516/react-h5-audio-player