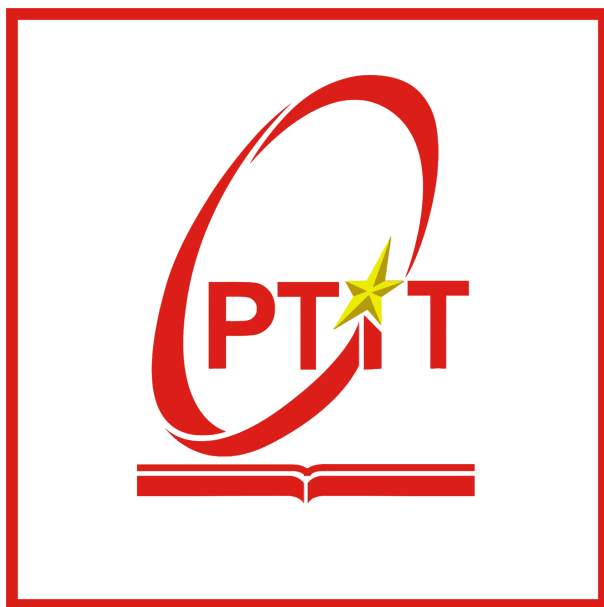


**BỘ THÔNG TIN VÀ TRUYỀN THÔNG
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**



BÁO CÁO CUỐI KỲ THỰC TẬP CƠ SỞ

XÂY DỰNG WEBSITE PHÁT NHẠC TRỰC TUYẾN SỬ DỤNG MERN STACK

Giảng viên hướng dẫn: TS. Kim Ngọc Bách

Sinh viên thực hiện:

- Nguyễn Quang Trung - B22DCDT321

MỤC LỤC

1. TỔNG QUAN VỀ ĐỀ TÀI.....	3
1.1. Giới thiệu chung.....	3
1.2. Mục tiêu đồ án.....	3
1.3. Phạm vi đồ án.....	4
2. PHÂN TÍCH YÊU CẦU.....	5
2.1. Yêu cầu chức năng.....	5
2.2. Yêu cầu phi chức năng.....	7
3. CÔNG NGHỆ VÀ CÔNG CỤ SỬ DỤNG.....	8
3.1. MERN Stack - Kiến trúc chính.....	8
3.2. Công nghệ mở rộng MERN Stack.....	10
3.3. Công cụ phát triển và testing.....	11
4. THIẾT KẾ GIAO DIỆN VÀ HỆ THỐNG.....	12
4.1. Kiến trúc tổng thể.....	12
4.2. Thiết kế cơ sở dữ liệu.....	12
4.3. API Design.....	17
4.4. Frontend Architecture.....	18
4.5. Backend Architecture.....	20
4.6. Thiết kế giao diện người dùng (UX/UI).....	21
4.7. Lưu đồ thuật giải.....	29
5. CÀI ĐẶT VÀ TRIỂN KHAI.....	35
5.1. Cài đặt môi trường phát triển.....	35
5.2. Chạy ứng dụng.....	36
6. KẾT LUẬN.....	37
6.1. Tóm tắt đồ án.....	37
6.2. Mục tiêu đạt được.....	37
6.3. Ý nghĩa của đồ án.....	37
7. TÀI LIỆU THAM KHẢO.....	38

1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu chung

Hiện nay, âm nhạc trực tuyến đã trở thành một phần không thể thiếu trong đời sống số của con người, giúp kết nối cảm xúc, giải trí và thậm chí hỗ trợ công việc, học tập. Với sự phát triển không ngừng của công nghệ, nhu cầu nghe nhạc trực tuyến ngày càng gia tăng, kéo theo sự xuất hiện của nhiều nền tảng phát nhạc trực tuyến phổ biến như Spotify, Apple Music, SoundCloud hay Nhaccuatui.

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ và nhu cầu giải trí số ngày càng tăng cao, việc xây dựng một nền tảng streaming nhạc trực tuyến trở thành một chủ đề hấp dẫn và có tính ứng dụng cao. Đề án "Website Streaming Nhạc" được thực hiện nhằm tạo ra một ứng dụng web hoàn chỉnh cho phép người dùng nghe nhạc trực tuyến với giao diện thân thiện và hiệu suất cao.

1.2. Mục tiêu đề án

→ **Mục tiêu chính:**

- Cung cấp nền tảng phát nhạc trực tuyến tiện lợi
 - Xây dựng một trang web nghe nhạc trực tuyến với giao diện thân thiện, dễ sử dụng, cho phép người dùng truy cập và thưởng thức âm nhạc một cách nhanh chóng mà không cần tải về thiết bị.
- Kho nhạc phong phú và đa dạng
 - Hỗ trợ nhiều thể loại nhạc khác nhau, từ nhạc Việt Nam, quốc tế đến các bản remix, indie, lo-fi, giúp đáp ứng nhu cầu của nhiều đối tượng người dùng.
 - Hỗ trợ tải lên và chia sẻ nhạc từ người dùng (nếu có cơ chế quản lý nội dung phù hợp).
- Tìm kiếm và phân loại nhạc hiệu quả
 - Hỗ trợ tìm kiếm theo tên bài hát, nghệ sĩ, album, thể loại.
 - Phân loại nhạc theo xu hướng, bảng xếp hạng, chủ đề để dễ dàng tiếp cận nội dung hot.

→ **Mục tiêu cụ thể:**

- Nghiên cứu và ứng dụng công nghệ TypeScript, Node.js, Express.js,...
- Sử dụng Zustand cho quản lý state và Vite cho build tool
- Thiết kế kiến trúc hệ thống phân tách frontend-backend rõ ràng
- Triển khai các tính năng streaming âm thanh

1.3. Phạm vi đề án

Đề án tập trung vào việc phát triển một ứng dụng web streaming nhạc với các chức năng cốt lõi bao gồm phát nhạc, quản lý playlist, tìm kiếm bài hát và giao diện người dùng hiện đại. Hệ thống được xây dựng theo mô hình Client-Server với frontend sử dụng React/TypeScript và backend sử dụng Node.js/Express.

2. PHÂN TÍCH YÊU CẦU

2.1. Yêu cầu chức năng

1. Các chức năng đối với người dùng

→ Đăng ký, đăng nhập và quản lý tài khoản

- Đăng ký tài khoản.
- Đăng nhập và cập nhật thông tin cá nhân (ảnh đại diện, tên hiển thị, mật khẩu, v.v.).
- Xem lịch sử nghe nhạc và quản lý danh sách bài hát yêu thích.

→ Nghe nhạc trực tuyến

- Nghe nhạc trực tuyến.
- Hỗ trợ các tính năng: tua nhanh, lặp lại, xáo trộn danh sách phát.

→ Tạo và quản lý danh sách phát

- Tạo danh sách phát cá nhân, chỉnh sửa hoặc xóa danh sách.
- Đặt danh sách phát ở chế độ công khai hoặc riêng tư.
- Gợi ý bài hát dựa trên danh sách phát của người dùng.

→ Tìm kiếm và khám phá nhạc

- Tìm kiếm bài hát theo tên, nghệ sĩ, album, thể loại.
- Xem bảng xếp hạng nhạc phổ biến theo tuần, tháng, năm.
- Nghe nhạc theo thể loại hoặc chủ đề (lofi, EDM, ballad, rap, v.v.).

→ Bình luận và tương tác

- Bình luận và đánh giá bài hát, album, danh sách phát.
- Thả tim bài hát yêu thích, chia sẻ bài hát lên mạng xã hội.
- Theo dõi nghệ sĩ yêu thích và nhận thông báo khi có nhạc mới.

2. Các chức năng đối với nghệ sĩ

→ **Đăng ký tài khoản nghệ sĩ**

- Xác minh tài khoản nghệ sĩ để đăng tải và quản lý nhạc của riêng mình.
- Cập nhật hồ sơ nghệ sĩ (tên, ảnh đại diện, tiểu sử, liên kết MXH).

→ **Tải lên và quản lý bài hát**

- Tải lên bài hát, album mới.
- Cập nhật thông tin bài hát: tên, thể loại, nghệ sĩ hợp tác, ảnh bìa, lời bài hát.
- Quản lý danh sách bài hát đã tải lên (chỉnh sửa, xóa, ẩn/hiện bài hát).

→ **Theo dõi thống kê bài hát**

- Xem số lượt nghe, lượt yêu thích, lượt chia sẻ.
- Thống kê từ người dùng.
- Báo cáo phân tích về đối tượng người nghe (độ tuổi, khu vực, xu hướng nghe nhạc).

→ **Tương tác với người hâm mộ**

- Trả lời bình luận của người nghe dưới bài hát.
- Tạo bài đăng cập nhật tin tức, sự kiện, thông báo nhạc mới.

3. Các chức năng đối với quản trị viên

→ **Quản lý người dùng**

- Xem danh sách người dùng, tìm kiếm tài khoản theo tên hoặc email.
- Khóa/mở khóa tài khoản vi phạm hoặc có hành vi không phù hợp.
- Xóa hoặc chỉnh sửa thông tin tài khoản.

→ **Quản lý bài hát và nội dung**

- Kiểm duyệt bài hát do nghệ sĩ hoặc người dùng tải lên trước khi công khai.
- Xóa bài hát có nội dung không phù hợp.

- Chỉnh sửa, cập nhật thông tin bài hát, album nếu cần.
- Quản lý nội dung lời bài hát, hình ảnh album để đảm bảo phù hợp với chính sách.

→ **Quản lý danh sách phát công khai**

- Tạo, chỉnh sửa, xóa danh sách phát chính thức trên hệ thống.
- Kiểm duyệt danh sách phát do người dùng tạo nếu có nội dung không phù hợp.
- Đề xuất danh sách phát theo xu hướng, thể loại hoặc nghệ sĩ nổi bật.

→ **Quản lý báo cáo vi phạm**

- Xử lý báo cáo vi phạm từ người dùng liên quan đến bài hát, bình luận, tài khoản.
- Kiểm tra và phản hồi khiếu nại về bản quyền từ nghệ sĩ hoặc tổ chức sở hữu nội dung.

2.2. Yêu cầu phi chức năng

→ **Khả năng sử dụng:**

- Giao diện trực quan, dễ sử dụng
- Tương thích trên các trình duyệt phổ biến
- Responsive design cho mobile và desktop

→ **Bảo mật:**

- Xác thực người dùng an toàn
- Bảo vệ dữ liệu cá nhân

3. CÔNG NGHỆ VÀ CÔNG CỤ SỬ DỤNG

3.1. MERN Stack - Kiến trúc chính

Dự án sử dụng MERN Stack như kiến trúc chính với các công nghệ hiện đại được mở rộng. MERN Stack bao gồm MongoDB, Express.js, React và Node.js - một stack công nghệ JavaScript full-stack mạnh mẽ cho phép phát triển ứng dụng web hiện đại với hiệu suất cao.

Cấu trúc của MERN Stack được tích hợp seamlessly, cho phép tích hợp liền mạch giữa các thành phần Frontend và Backend. Đây là cách cấu trúc của MERN hoạt động trong dự án:

M - MongoDB (Database Layer): MongoDB được sử dụng làm cơ sở dữ liệu NoSQL chính với các ưu điểm:

- **Document-oriented storage:** Lưu trữ dữ liệu dưới dạng JSON-like documents, phù hợp với cấu trúc dữ liệu phức tạp của ứng dụng streaming
- **Flexible schema:** Cho phép thay đổi cấu trúc dữ liệu dễ dàng khi requirements thay đổi
- **High performance:** Tối ưu cho read operations và complex queries
- **Scalability:** Hỗ trợ horizontal scaling và sharding
- **Indexing:** Powerful indexing capabilities cho tìm kiếm nhanh bài hát, nghệ sĩ
- **Aggregation pipeline:** Xử lý data analytics và reporting

Trong dự án, MongoDB lưu trữ:

- User profiles và authentication data
- Song metadata (title, artist, album, genre, duration)
- Playlist information và user preferences
- Listening history và analytics data
- Audio file references và storage paths

E - Express.js (Backend Framework): Express.js đóng vai trò là web framework cho Node.js với các tính năng:

- **Minimal và flexible:** Lightweight framework không enforce specific structure

- **Middleware system:** Powerful middleware pattern cho authentication, logging, error handling
- **RESTful API:** Dễ dàng tạo RESTful endpoints cho frontend consumption
- **Static file serving:** Serve audio files và static assets
- **Template engines support:** Tích hợp với various view engines
- **Third-party middleware:** Huge ecosystem của middleware packages

Trong dự án, Express.js xử lý:

- RESTful API endpoints (/api/songs, /api/users, /api/playlists)
- Authentication middleware với JWT tokens
- File upload handling cho audio files
- Audio streaming với proper headers và buffering
- CORS configuration cho frontend-backend communication
- Error handling và response formatting

R - React (Frontend Library): React được sử dụng để xây dựng user interface với architecture hiện đại:

- **Component-based architecture:** Tái sử dụng components và maintainable code
- **Virtual DOM:** Efficient rendering và performance optimization
- **Hooks system:** Modern state management với useState, useEffect, custom hooks
- **JSX syntax:** Declarative UI development
- **Ecosystem:** Huge library ecosystem cho UI components, routing, state management
- **Developer tools:** Excellent debugging tools và hot reloading

Trong dự án, React components bao gồm:

- AudioPlayer component với playback controls
- SongList và PlaylistManager components
- Search và Filter components
- UserProfile và Authentication forms
- ResponsiveLayout cho mobile/desktop
- Real-time audio visualization components

N - Node.js (Runtime Environment): Node.js đóng vai trò là môi trường thời gian chạy mã phía máy chủ với:

- **Event-driven architecture:** Perfect cho real-time applications như streaming
- **Non-blocking I/O:** High performance cho concurrent requests
- **JavaScript full-stack:** Same language cho frontend và backend
- **NPM ecosystem:** Largest package manager với millions of packages
- **Cross-platform:** Runs on various operating systems
- **Streaming capabilities:** Built-in streaming support cho audio delivery

Trong dự án, Node.js xử lý:

- Server-side JavaScript execution
- Real-time audio streaming processing
- File system operations cho audio files
- Concurrent user request handling
- Background tasks và scheduling
- Integration với third-party services

3.2. Công nghệ mở rộng MERN Stack

TypeScript (Language Enhancement): TypeScript được tích hợp vào toàn bộ MERN stack để:

- **Static type checking:** Catch errors at compile-time thay vì runtime
- **Better IDE support:** IntelliSense, auto-completion, refactoring tools
- **Code documentation:** Types serve as living documentation
- **Large-scale application:** Better maintainability cho complex projects
- **Interface definitions:** Clear contracts between frontend-backend

Zustand (State Management): Zustand thay thế Redux trong MERN stack với:

- **Simplified API:** Less boilerplate code compared to Redux
- **TypeScript support:** First-class TypeScript integration
- **Performance:** Optimized re-renders và subscription model
- **Developer experience:** Easy debugging và devtools integration
- **Small bundle size:** Minimal impact on application size

Vite (Build Tool và Development Server): Vite thay thế Create React App trong MERN stack với:

- **Lightning fast HMR:** Hot Module Replacement in milliseconds
- **ES modules:** Native ES modules support cho faster builds
- **Build optimization:** Tree-shaking và code splitting out of the box
- **Plugin ecosystem:** Rich plugin system cho various integrations
- **TypeScript support:** Built-in TypeScript compilation
- **Development server:** Optimized dev server với caching

3.3. Công cụ phát triển và testing

Postman (API Testing): Kiểm thử API backend trước khi kết nối với frontend:

- **API endpoint testing:** Test các route Express.js
- **Authentication testing:** JWT token validation
- **File upload testing:** Audio file upload endpoints
- **Response validation:** Verify API responses và error handling
- **Environment management:** Different environments (dev, staging, prod)
- **Collection sharing:** Team collaboration cho API documentation

Test cases bao gồm:

- User authentication (register, login, logout)
- Song management (upload, get, search, delete)
- Playlist operations (create, update, add songs, remove songs)
- Audio streaming endpoints với proper headers

Version Control:

- Git cho version control
- GitHub cho code repository và collaboration

Development Tools:

- ESLint cho code linting
- Prettier cho code formatting

4. THIẾT KẾ GIAO DIỆN VÀ HỆ THỐNG

4.1. Kiến trúc tổng thể

Hệ thống được thiết kế theo mô hình Client-Server với kiến trúc phân tách rõ ràng giữa frontend và backend. Kiến trúc này mang lại các lợi ích:

Frontend (Client):

- Single Page Application (SPA) sử dụng React
- State management với Zustand
- Audio player component với Web Audio API
- Responsive UI components

Backend (Server):

- RESTful API với Express.js
- Authentication và authorization middleware
- Audio streaming endpoints
- Database integration layer

Communication:

- HTTP/HTTPS cho API calls
- WebSocket cho real-time features (optional)
- Stream protocol cho audio delivery

4.2. Thiết kế cơ sở dữ liệu

- User model

```
import mongoose from "mongoose";

const userSchema = new mongoose.Schema(
  {
    fullName: {
      type: String,
      required: true,
    },
    imageUrl: {
      type: String,
      required: true,
    },
  },
  {
    timestamps: true,
  }
);
```

```

    clerkId: {
      type: String,
      required: true,
      unique: true,
    },
    likedSongs: [{ type: mongoose.Schema.Types.ObjectId, ref: "Song" }],
    role: {
      type: String,
      enum: ["user", "artist", "admin"],
      default: "user"
    },
    // Thông tin thêm cho artist
    artistInfo: {
      bio: { type: String, default: "" },
      genres: [{ type: String }],
      monthlyListeners: { type: Number, default: 0 },
      verified: { type: Boolean, default: false }
    }
  },
  { timestamps: true } // createdAt, updatedAt
);

export const User = mongoose.model("User", userSchema);

```

- Song model

```

import mongoose from "mongoose";

const songSchema = new mongoose.Schema(
  {
    title: {
      type: String,
      required: true,
    },
    artist: {
      type: String,

```

```
        required: true,
    },
    imageUrl: {
        type: String,
        required: true,
    },
    audioUrl: {
        type: String,
        required: true,
    },
    duration: {
        type: Number,
        required: true,
    },
    plays: {
        type: Number,
        default: 0,
    },
    artistId: {
        type: String,
        required: true,
    },
    albumId: {
        type: mongoose.Schema.Types.ObjectId,
        ref: "Album",
    },
}
```

```

    },
    { timestamps: true }
  );

export const Song = mongoose.model("Song", songSchema);

```

- Playlist model

```

import mongoose from "mongoose";

const playlistSchema = new mongoose.Schema(
  {
    name: { type: String, required: true },
    userId: { type: mongoose.Schema.Types.ObjectId, ref: "User", required:
true },
    songs: [{ type: mongoose.Schema.Types.ObjectId, ref: "Song" }],
  },
  { timestamps: true }
);

export const Playlist = mongoose.model("Playlist", playlistSchema);

```

- Message model

```

import mongoose from "mongoose";

const messageSchema = new mongoose.Schema(
  {
    senderId: { type: String, required: true }, // Clerk user ID
    receiverId: { type: String, required: true }, // Clerk user ID
  }
);

```

```

        content: { type: String, required: true },
      },
      { timestamps: true }
    );

export const Message = mongoose.model("Message", messageSchema);

```

- Comment model

```

import mongoose from "mongoose";

const commentSchema = new mongoose.Schema({
  songId: { type: String, required: true },
  userId: { type: String, required: true },
  content: { type: String, required: true },
  parentId: { type: String, default: null },
  likes: { type: [String], default: [] },
  createdAt: { type: Date, default: Date.now },
});

const Comment = mongoose.model("Comment", commentSchema);

export default Comment;

```

- Album model

```

import mongoose from "mongoose";

const albumSchema = new mongoose.Schema(

```



```

{
  title: { type: String, required: true },
  artist: { type: String, required: true },
  artistId: { type: String, required: true },
  imageUrl: { type: String, required: true },
  releaseYear: { type: Number, required: true },
  songs: [{ type: mongoose.Schema.Types.ObjectId, ref: "Song" }],
},
{ timestamps: true }
); // createdAt, updatedAt

export const Album = mongoose.model("Album", albumSchema);

```

4.3. API Design

User Endpoints:

- GET / - Lấy danh sách users
- POST /:userId/like/:songId - Like bài hát
- GET /:userId/liked-songs - Lấy danh sách bài hát đã like

Authentication Endpoints:

- POST /api/auth/register - Đăng ký tài khoản
- POST /api/auth/login - Đăng nhập
- POST /api/auth/logout - Đăng xuất
- GET /api/auth/profile - Lấy thông tin profile

Song Endpoints:

- GET / - Lấy tất cả bài hát (admin)
- GET /featured - Lấy bài hát nổi bật
- GET /:userId/liked-songs - Lấy bài hát đã like của user

Playlist Endpoints:

- GET /user/:userId - Lấy playlist của user
- POST / - Tạo playlist mới
- GET /:id - Lấy thông tin playlist
- DELETE /:id - Xóa playlist
- POST /:id/songs - Thêm bài hát vào playlist
- DELETE /:id/songs/:songId - Xóa bài hát khỏi playlist

Comment Endpoints:

- GET /song/:songId - Lấy comment của bài hát
- POST / - Tạo comment mới
- DELETE /:id - Xóa comment
- POST /:id/like - Like comment

Stats API

- GET / - Lấy thống kê (admin)

Admin API

- POST /songs - Tạo bài hát mới
- POST /albums - Tạo album mới
- GET /users - Lấy danh sách users
- PUT /users/:id/role - Cập nhật role của user
- DELETE /users/:id - Xóa user

Artist API

- GET /:id - Lấy thông tin nghệ sĩ
- GET /featured - Lấy danh sách nghệ sĩ nổi bật

Album API

- GET / - Lấy tất cả album
- GET /:id - Lấy thông tin album theo ID

4.4. Frontend Architecture

Component Structure:

```

frontend/
├── src/
│   ├── pages/                # Các trang chính của ứng dụng
│   │   ├── admin/           # Trang quản trị
│   │   ├── album/           # Trang album
│   │   ├── artist/          # Trang nghệ sĩ
│   │   ├── auth-callback/    # Xử lý callback authentication
│   │   ├── chat/            # Trang chat
│   │   ├── home/            # Trang chủ
│   │   ├── liked-songs/      # Trang bài hát yêu thích
│   │   ├── playlist/        # Trang playlist
│   │   ├── Search/          # Trang tìm kiếm
│   │   └── song/            # Trang bài hát
│   ├── components/          # Các component tái sử dụng
│   │   ├── ui/              # UI components
│   │   ├── comments/        # Components liên quan đến comment
│   │   ├── playlists/       # Components liên quan đến playlist
│   │   ├── skeletons/       # Loading skeleton components
│   │   ├── AlbumItem.tsx    # Component hiển thị album
│   │   ├── SongItem.tsx     # Component hiển thị bài hát
│   │   ├── Topbar.tsx       # Component thanh điều hướng trên cùng
│   │   └── SignInOAuthButtons.tsx # Component nút đăng nhập OAuth
│   ├── layout/              # Layout components
│   ├── lib/                 # Thư viện và utilities
│   ├── providers/           # Context providers
│   ├── stores/              # State management stores
│   ├── types/               # TypeScript type definitions
│   ├── App.tsx              # Component gốc của ứng dụng
│   ├── main.tsx             # Entry point của ứng dụng
│   └── index.css             # Global CSS styles
├── public/                  # Static files
├── package.json             # Dependencies và scripts
└── tsconfig.json            # TypeScript configuration

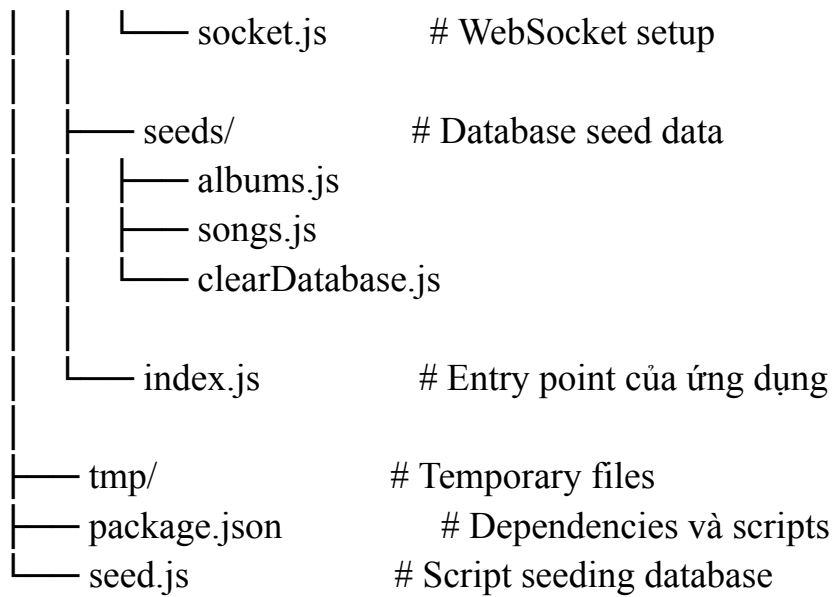
```

```
|— vite.config.ts      # Vite configuration
|— tailwind.config.js  # Tailwind CSS configuration
|— postcss.config.js   # PostCSS configuration
```

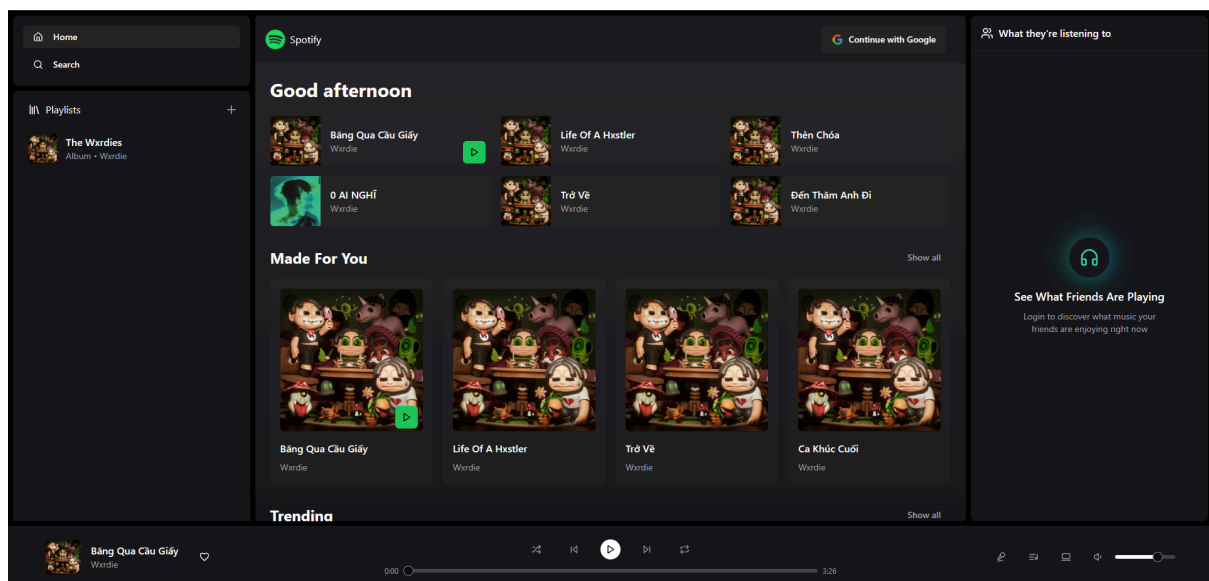
4.5. Backend Architecture

Component Structure:

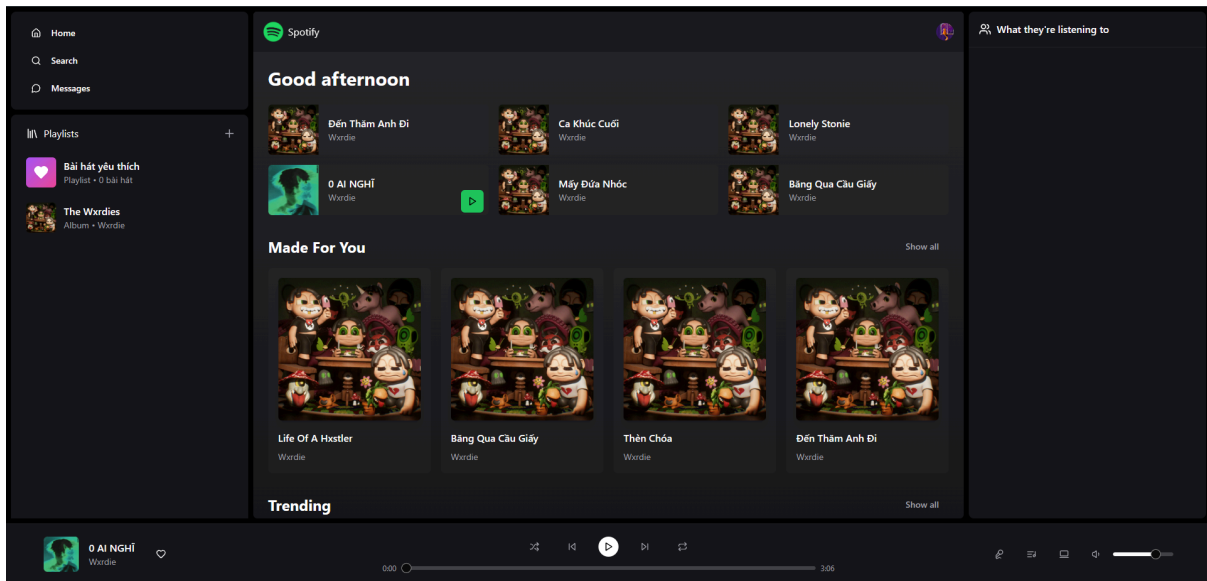
```
backend/
|— src/
|   |— controllers/      # Xử lý logic nghiệp vụ
|   |   |— artist.controller.js
|   |   |— comment.controller.js
|   |
|   |— routes/          # Định nghĩa các API endpoints
|   |   |— admin.route.js    # API quản trị
|   |   |— album.route.js    # API album
|   |   |— artist.route.js   # API nghệ sĩ
|   |   |— auth.route.js     # API xác thực
|   |   |— comment.route.js  # API bình luận
|   |   |— playlist.route.js # API playlist
|   |   |— song.route.js     # API bài hát
|   |   |— stat.route.js     # API thống kê
|   |   |— user.route.js     # API người dùng
|   |
|   |— models/          # Định nghĩa database models
|   |   |— user.model.js
|   |   |— song.model.js
|   |   |— album.model.js
|   |   |— playlist.model.js
|   |   |— comment.model.js
|   |   |— message.model.js
|   |
|   |— middleware/      # Middleware functions
|   |   |— auth.middleware.js # Xác thực và phân quyền
|   |
|   |— lib/             # Thư viện và utilities
|   |   |— db.js          # Database connection
```



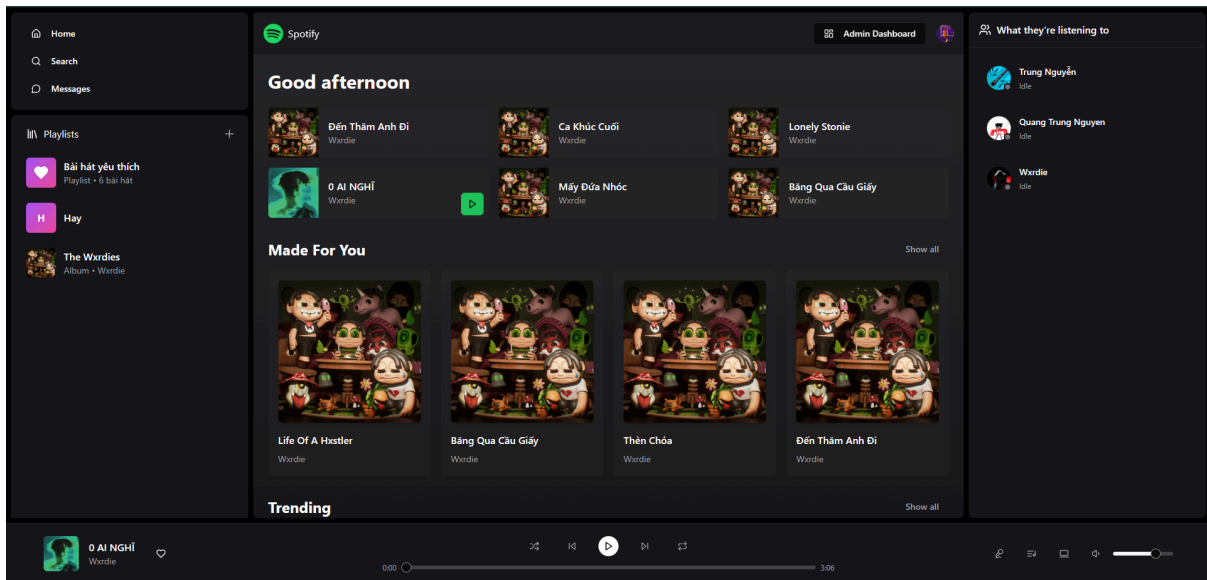
4.6. Thiết kế giao diện người dùng (UX/UI)



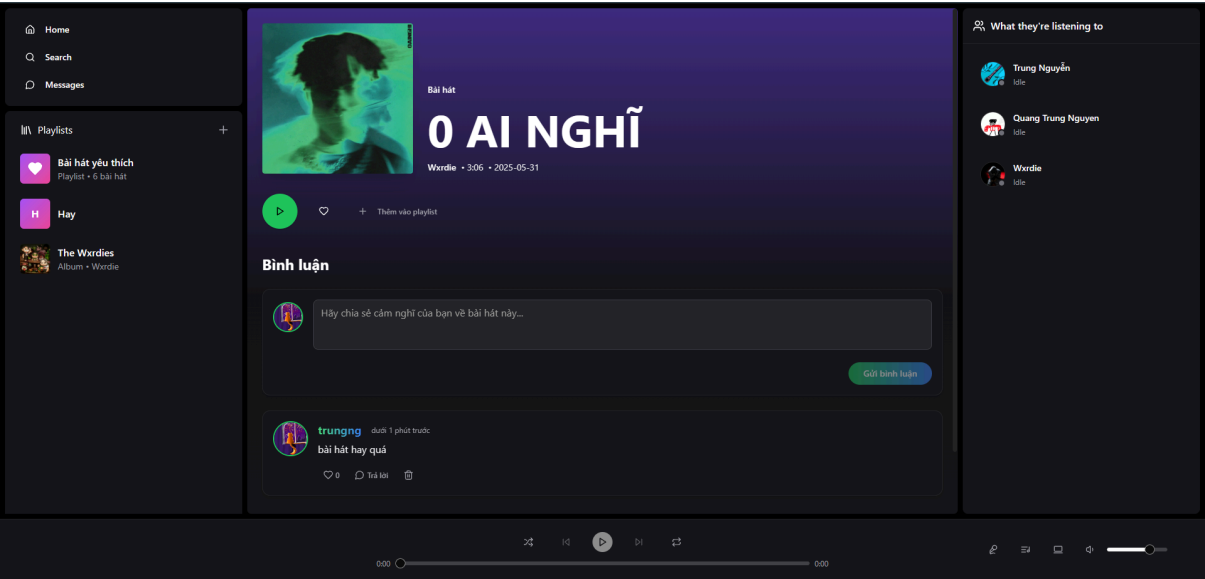
Main Layout - Trang chủ - Chưa đăng ký/đăng nhập



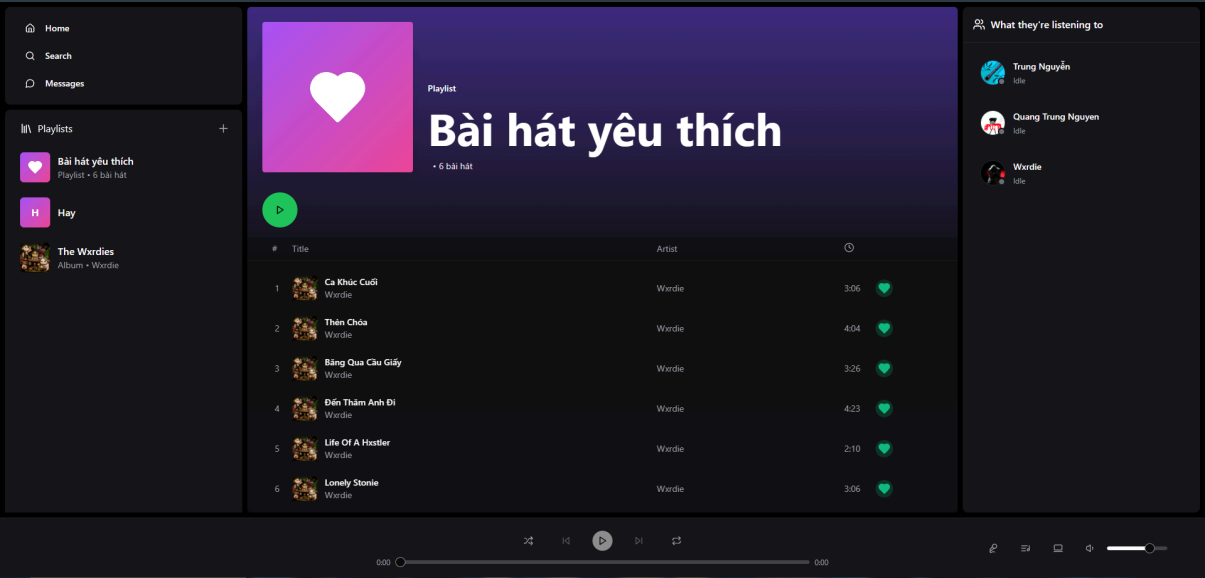
Main Layout - Trang chủ - Đã đăng ký/đã đăng nhập - người dùng



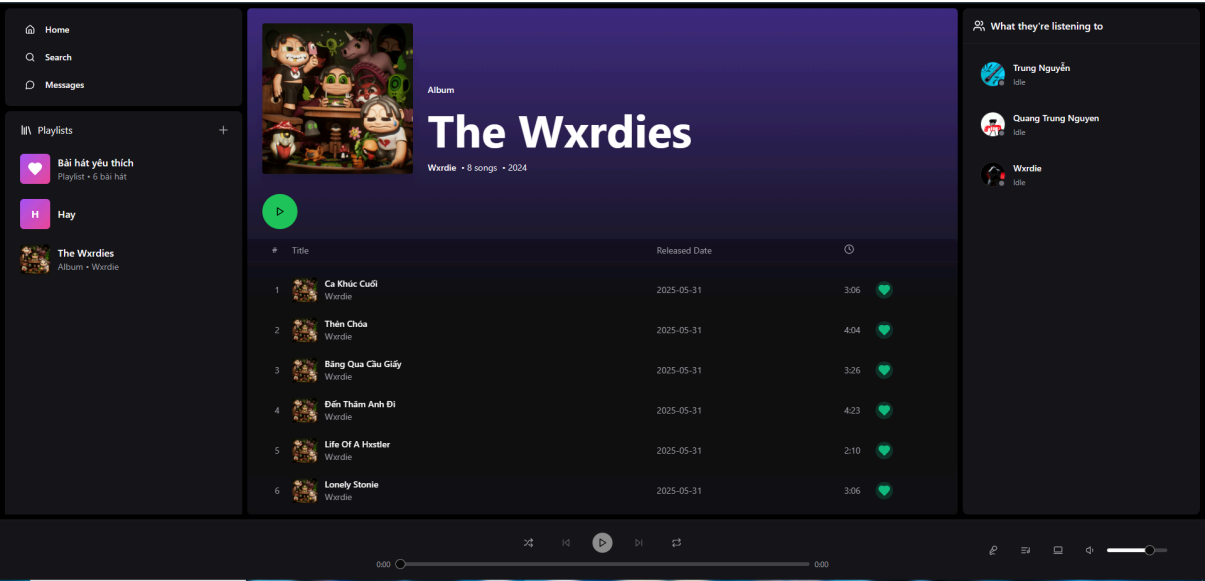
Main Layout - Trang chủ - Đã đăng ký/đã đăng nhập - admin/artist



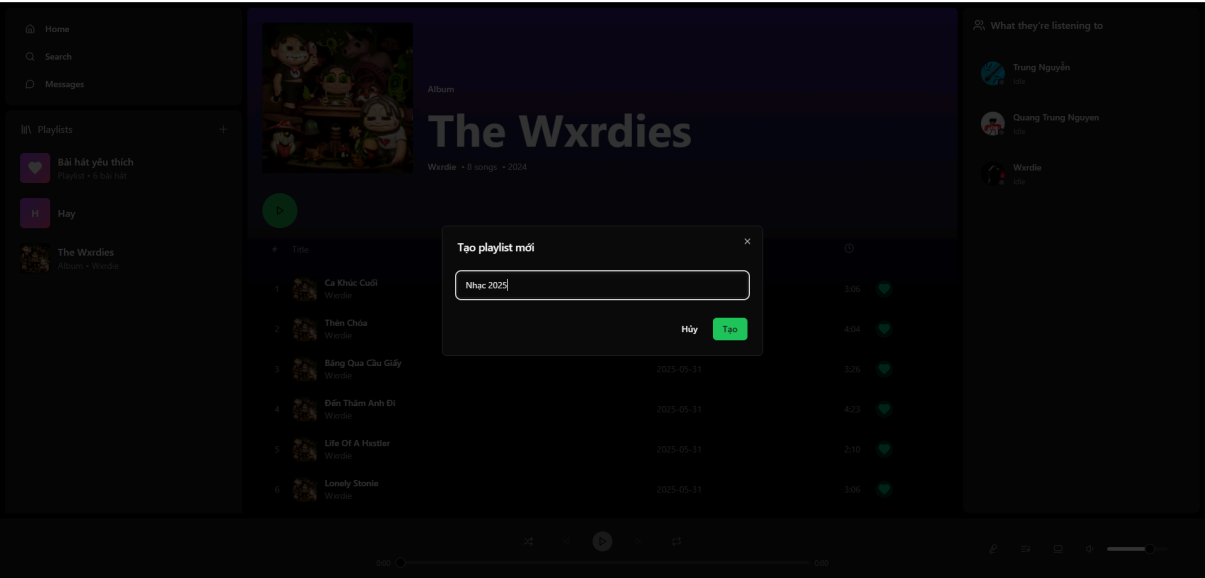
Song Page



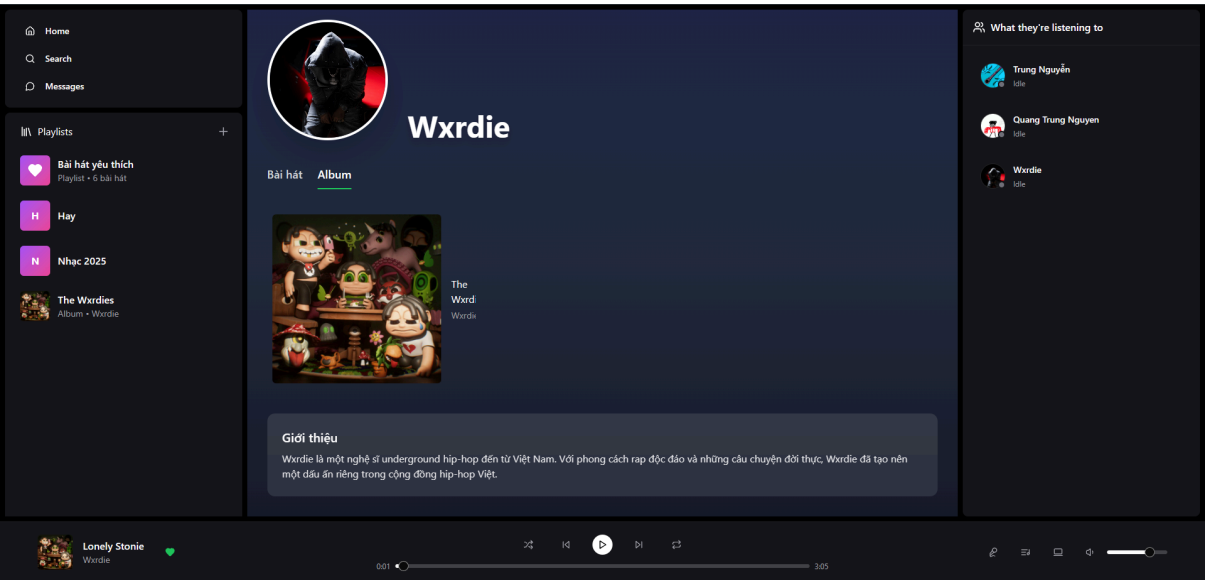
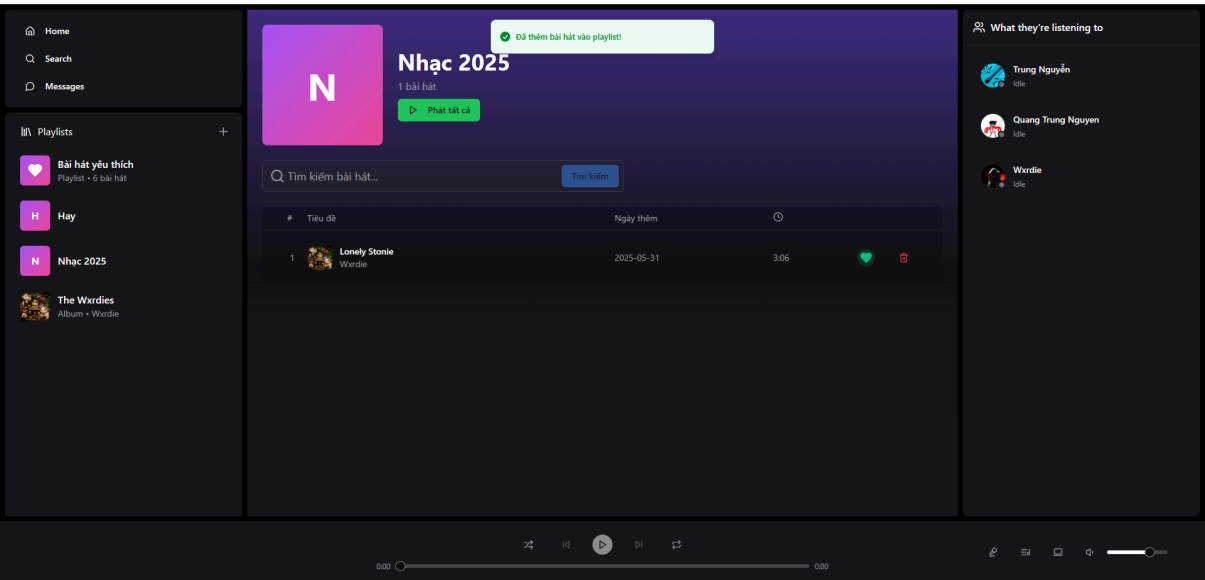
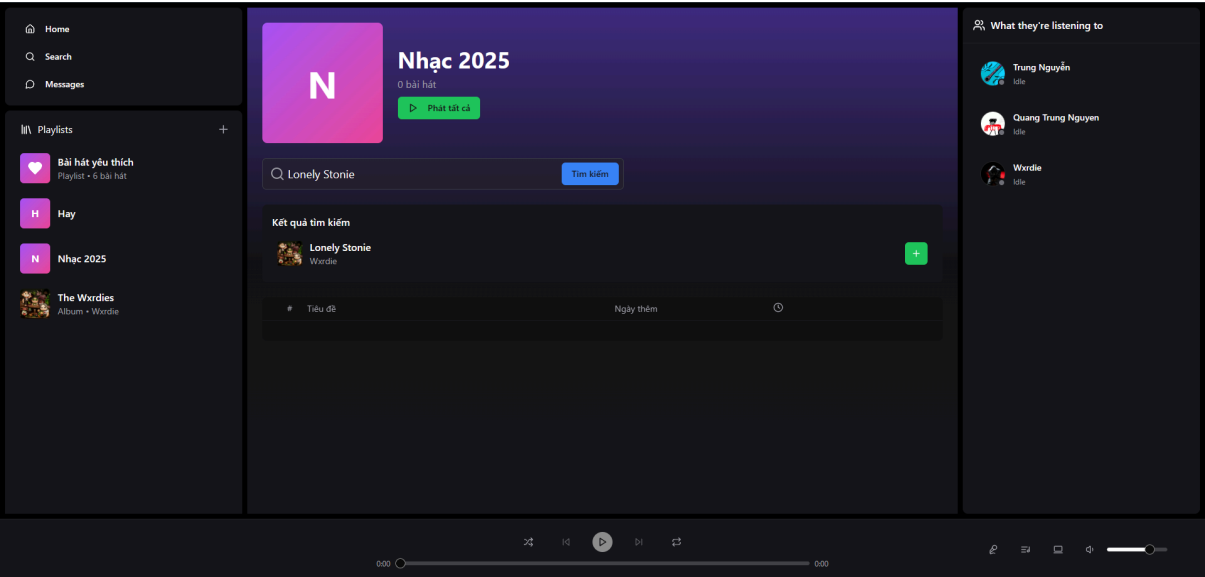
Trang bài hát yêu thích



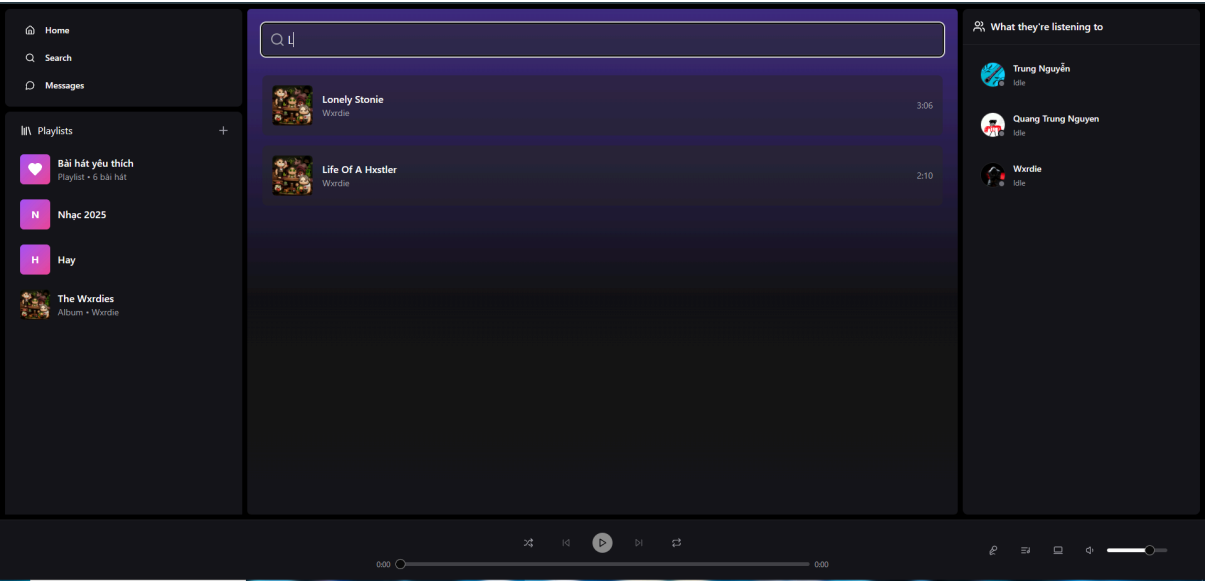
Trang album



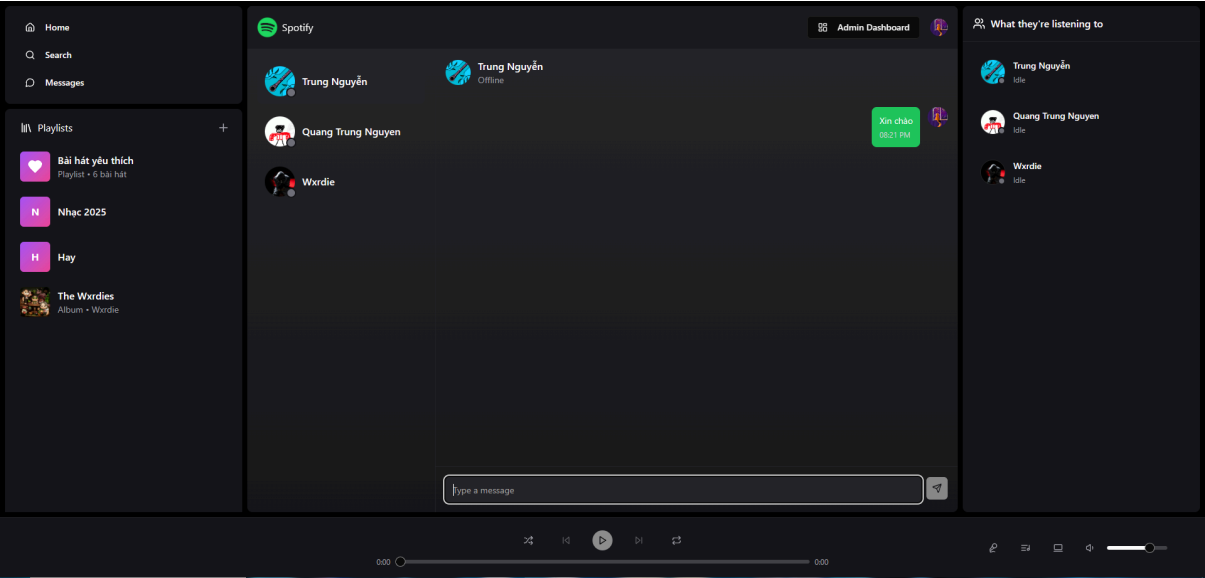
Tạo playlist mới và thêm bài hát vào playlist




Trang nghệ sĩ




Trang tìm kiếm




Trang tin nhắn




Music Manager
Manage your music catalog




Total Songs
9



Total Albums
1



Total Artists
1






















Total Users
4


Songs

Albums


Users




Title	Artist	Duration	Actions
 O AI NGHĨ	Wordie	🕒 3:06	
 Trở Về	Wordie	🕒 3:14	
 Máy Dừa Nhóc	Wordie	🕒 4:23	
 Lonely Stonie	Wordie	🕒 3:06	
 Life Of A Hustler	Wordie	🕒 2:10	
 Đến Thăm Anh Đi	Wordie	🕒 4:23	
 Bàng Qua Cầu Giấy	Wordie	🕒 3:26	
 Thần Chúa	Wordie	🕒 4:04	
 Ca Khúc Cuối	Wordie	🕒 3:06	




Music Manager
Manage your music catalog




Total Songs
9



Total Albums
1



Total Artists
1






Total Users
4


Songs

Albums


Users




Title	Artist	Release Year	Songs	Actions
 The Wordies	Wordie	📅 2024	🕒 8 songs	




Music Manager
Manage your music catalog




Total Songs
9



Total Albums
1



Total Artists
1















Total Users
4

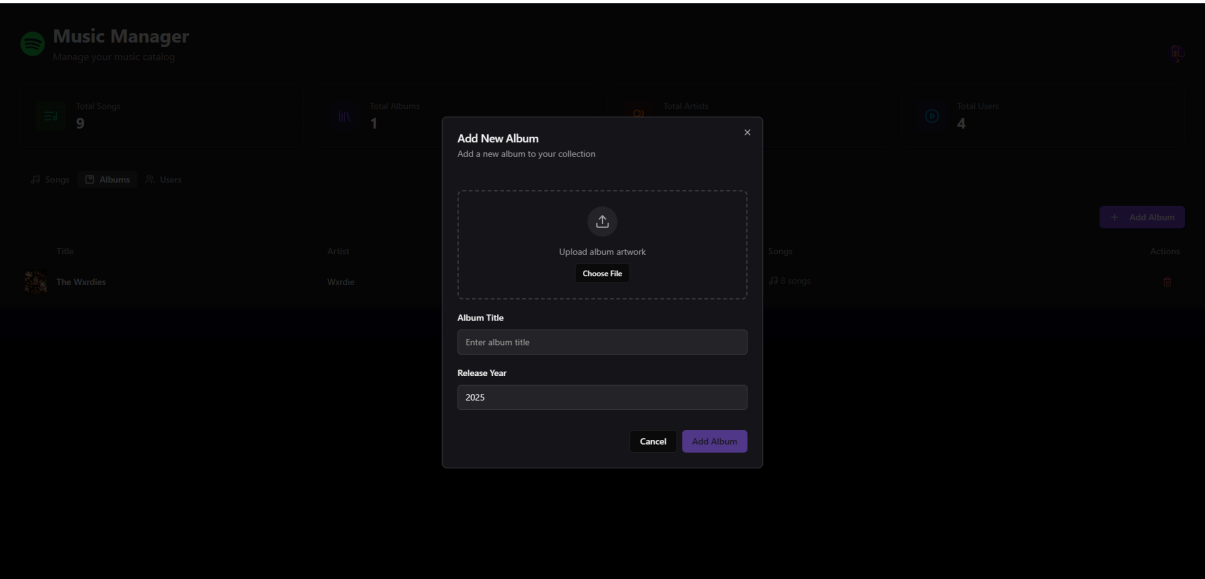
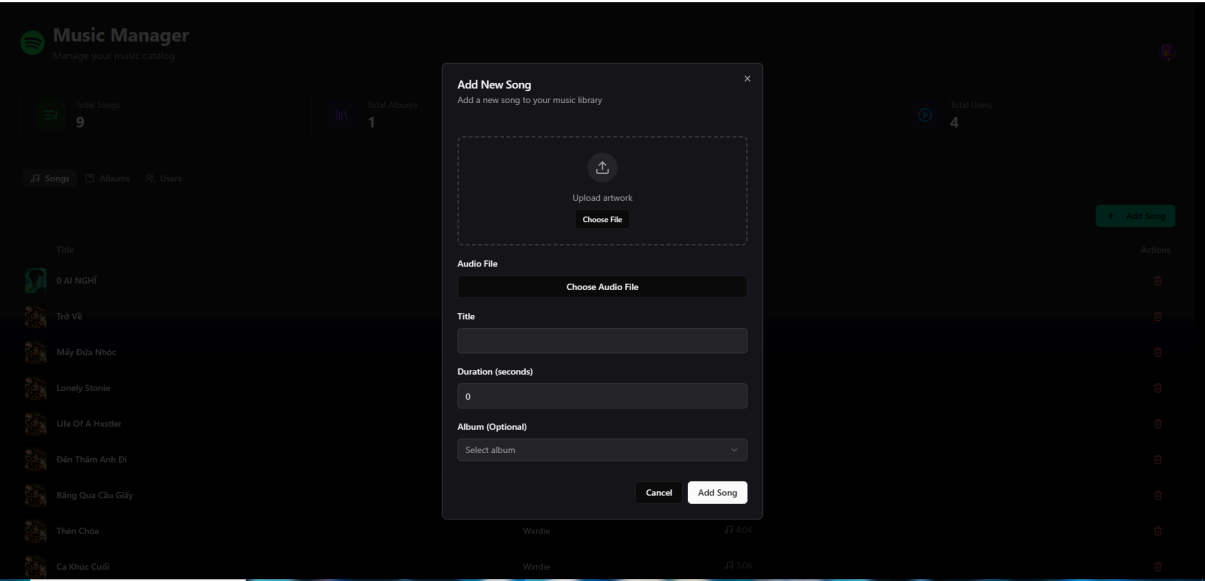
Songs

Albums

Users

User	Role	Actions
 Quang Trung Nguyen user_2wS1JE0bOMoUnZA61wNMnqtMgb3	User 	
 Wordie wordie_artist	Artist 	
 trungng user_2wER14D8s1qTauhQIEqLMug1g	Admin 	
 Trung Nguyễn user_2wMheKUFbIFokNTFeDqpVAB1H0V	Admin 	

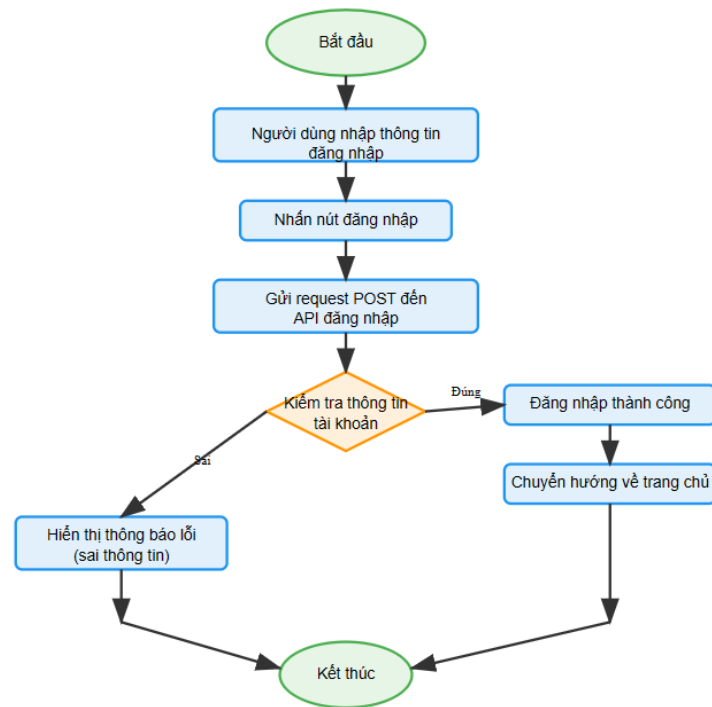
Trang quản trị



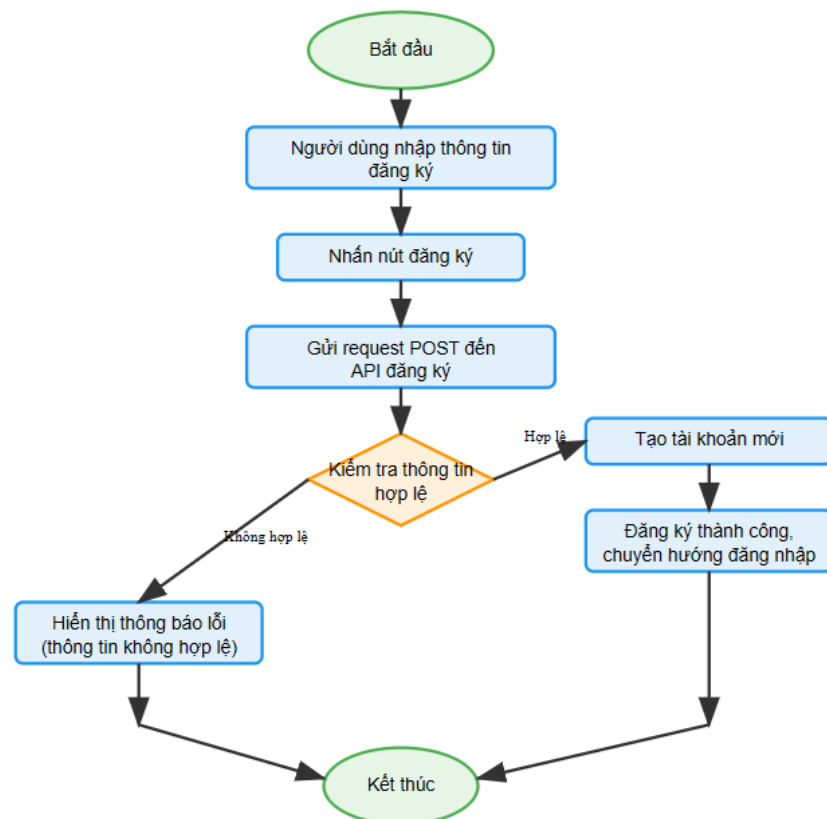
Thêm album/bài hát

4.7. Lưu đồ thuật giải

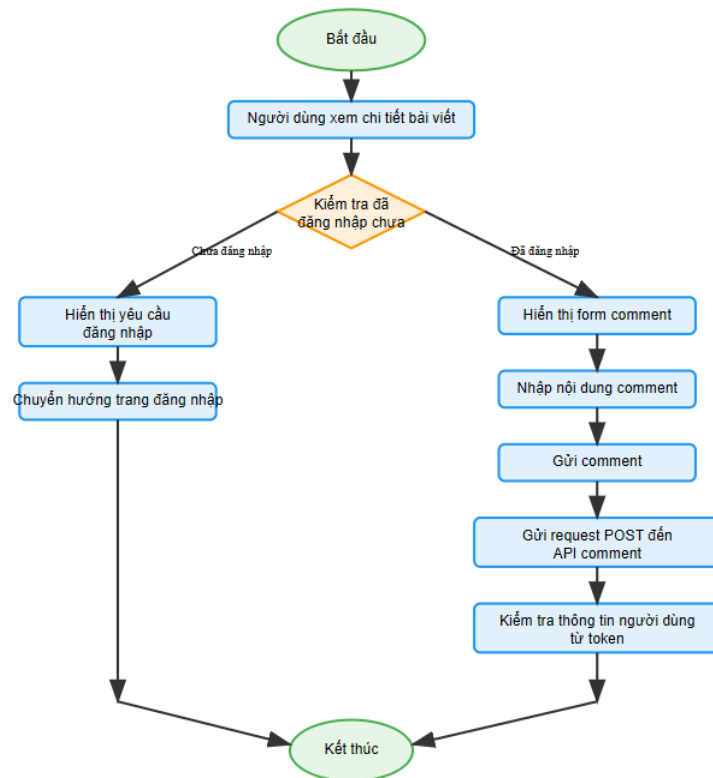
- Chức năng đăng nhập



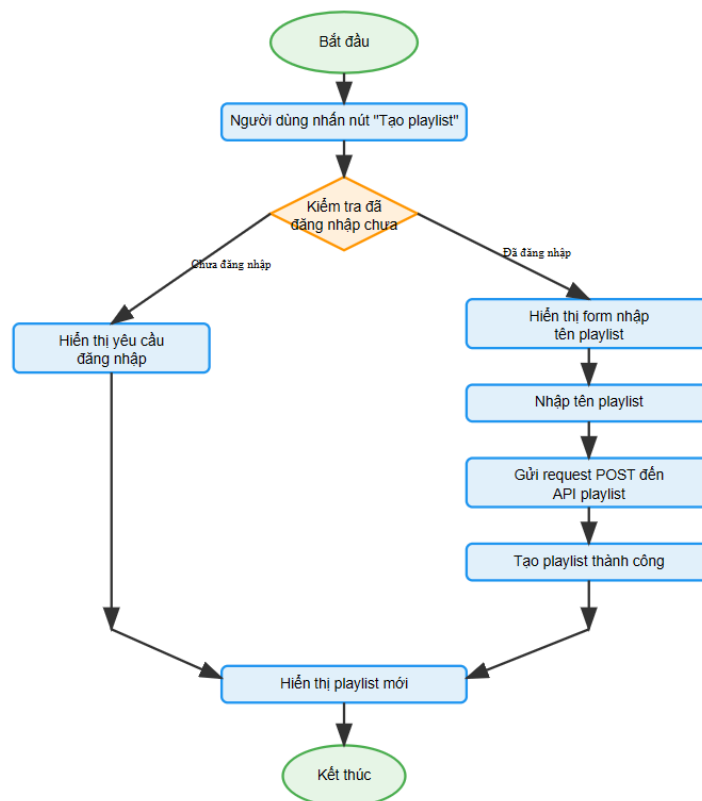
- Chức năng đăng ký



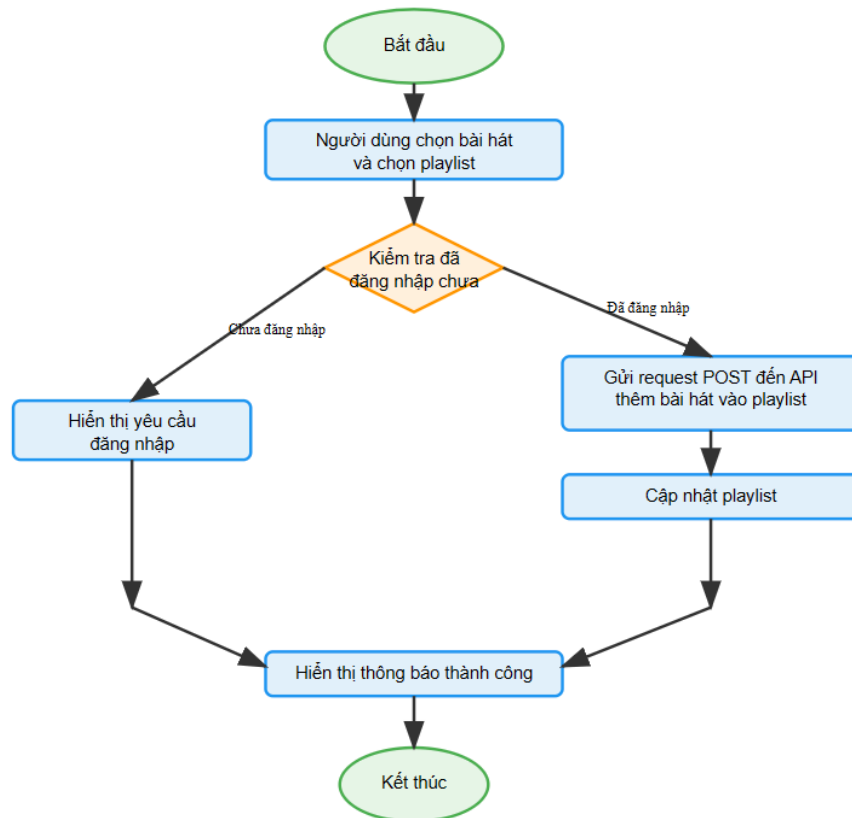
- Chức năng comment



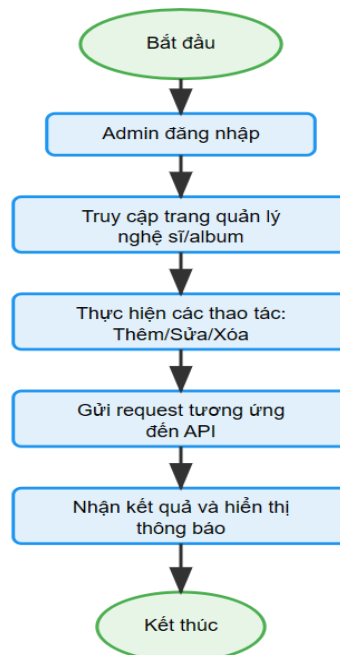
- Chức năng tạo playlist



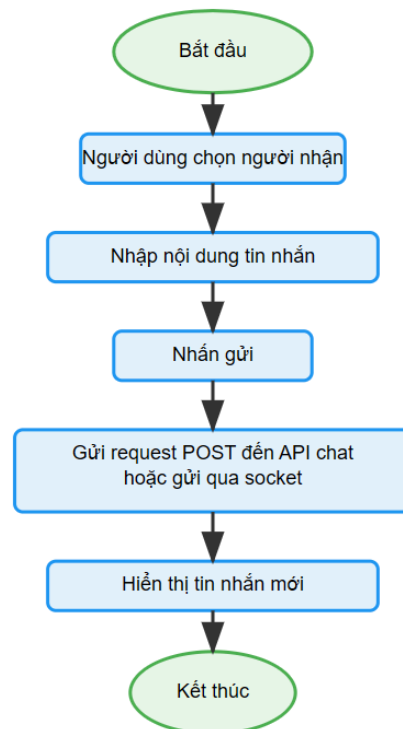
- Chức năng thêm bài hát vào playlist



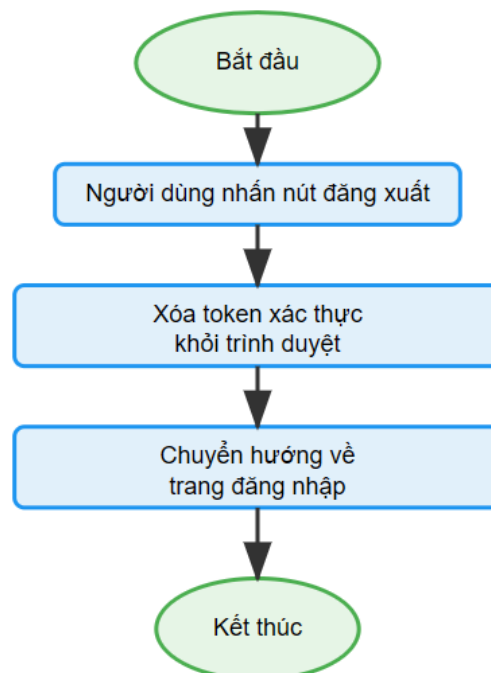
- Chức năng quản lý của admin (mặc định đã đăng nhập tài khoản có role admin)



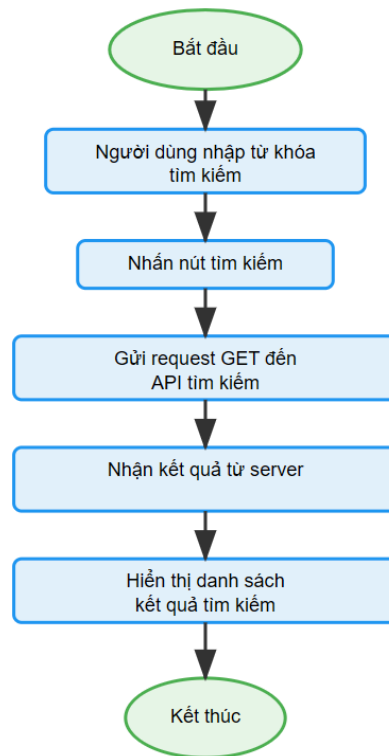
- Chức năng nhắn tin (mặc định đã đăng nhập tài khoản)



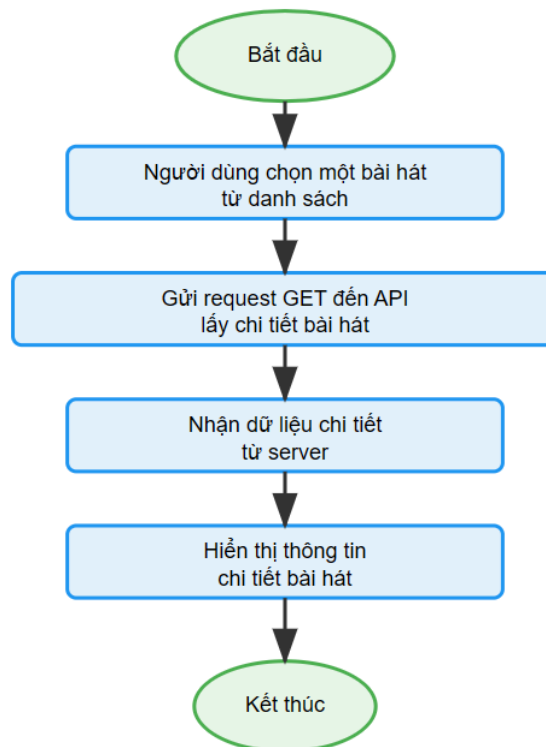
- Chức năng đăng xuất



- Chức năng tìm kiếm bài hát



- Chức năng xem chi tiết (lấy trang) bài hát, album, nghệ sĩ



5. CÀI ĐẶT VÀ TRIỂN KHAI

5.1. Cài đặt môi trường phát triển

Yêu cầu hệ thống:

- Node.js version 16.x hoặc cao hơn
- NPM hoặc Yarn package manager
- Git version control
- Code editor (VS Code khuyến nghị)

Các bước cài đặt:

1. Clone repository:

```
git clone
https://github.com/trungng29/tcs_music_streaming_platform.git
cd tcs_music_streaming_platform
```

2. Cài đặt dependencies cho backend:

```
cd backend
npm install
```

3. Cài đặt dependencies cho frontend:

```
cd ../frontend
npm install
```

4. Cấu hình environment variables:

```
# Backend .env
PORT=3000
DB_CONNECTION_STRING=mongodb://localhost:27017/music
_streaming
JWT_SECRET=your_secret_key
AUDIO_STORAGE_PATH=./uploads/audio
```

```
# Frontend .env  
VITE_API_BASE_URL=http://localhost:3000/api
```

5.2. Chạy ứng dụng

Development mode:

Terminal 1 - Backend:

```
cd backend  
npm run dev
```

Terminal 2 - Frontend:

```
cd frontend  
npm run dev
```

Production build:

```
# Build frontend  
cd frontend  
npm run build
```

```
# Start backend in production  
cd ../backend  
npm start
```

6. KẾT LUẬN

6.1. Tóm tắt đồ án

Đồ án "Website Streaming Nhạc" đã được thực hiện thành công với việc ứng dụng các công nghệ JavaScript hiện đại để xây dựng một ứng dụng web streaming nhạc hoàn chỉnh. Hệ thống được phát triển với kiến trúc phân tách rõ ràng giữa frontend và backend, sử dụng TypeScript cho type safety, Zustand cho state management, và Vite cho build optimization.

6.2. Mục tiêu đạt được

Đồ án đã đạt được các mục tiêu đề ra:

- Xây dựng thành công website streaming nhạc với đầy đủ tính năng cơ bản
- Áp dụng hiệu quả các công nghệ hiện đại
- Tạo ra giao diện người dùng trực quan và responsive

6.3. Ý nghĩa của đồ án

Về mặt học thuật:

- Nắm vững các công nghệ frontend và backend hiện đại
- Hiểu sâu về kiến trúc ứng dụng web full-stack
- Áp dụng các best practices trong software development
- Phát triển kỹ năng problem-solving và debugging

Về mặt thực tiễn:

- Tạo ra sản phẩm có thể ứng dụng thực tế
- Cung cấp nền tảng cho việc phát triển thêm các tính năng nâng cao
- Demonstrate technical skills cho career development
- Contribute to open-source community

7. TÀI LIỆU THAM KHẢO

- **TypeScript Documentation** - <https://www.typescriptlang.org/docs/>
- **React Documentation** - <https://react.dev/>
- **Node.js Documentation** - <https://nodejs.org/en/docs/>
- **Express.js Guide** - <https://expressjs.com/>
- **Zustand Documentation** - <https://zustand-demo.pmnd.rs/>
- **Vite Guide** - <https://vitejs.dev/guide/>
- **Tailwind Documentation** - <https://v2.tailwindcss.com/docs>
- **Clerk Documentation** - <https://clerk.com/docs>
-  Full Stack Spotify Clone: Next 13.4, React, Stripe, Supabase,...
-  Learn MongoDB in 1 Hour 
-  Advanced Spotify Clone: Build & Deploy a MERN Stack Sp...
-  JWT Authentication Tutorial - Node.js
-  What Is JWT and Why Should You Use JWT
-  What Is JWT and Why Should You Use JWT
-  Create Spotify Clone Using React JS & Tailwind CSS | Build...
-  Tailwind in 100 Seconds
-  Tailwind CSS v4 Full Course 2025 | Master Tailwind in One ...
-  React Full Course for free  (2024)
-  Every React Concept Explained in 12 Minutes
-  React JS 19 Full Course 2025 | Build an App and Master Rea...

