TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

ĐỀ TÀI: PHÁT TRIỂN PHẦN MỀM SPOTIFY CLONE

Giảng viên hướng dẫn: Từ Lãng Phiêu

Sinh viên thực hiện: Nhóm 14

Nguyễn Văn A	xxxxxxxxxx
Nguyễn Trí Dũng	3121410112
Lê Công Thành	3119410387
Nguyễn Văn C	xxxxxxxxxx

Lớp: DCT1212 - Sáng thứ 4

TP. Hồ Chí Minh, ngày 10, tháng 5, năm 2025

Mục lục

I MÔ TẢ VỀ ĐỀ TÀI **2**
I.1 Khảo sát nghiệp vụ . . . 2
I.2 Thông tin về đề tài . . . 2
I.3 Phân công công việc . . . 2

II THIẾT KẾ CƠ SỞ DỮ LIỆU **4**
II.1 Thiết kế ERD . . . 4
II.2 Mô tả Use Case . . . 4

III THIẾT KẾ LƯỢC ĐỒ CSDL QUAN HỆ **10**
III.1 Lược đồ CSDL quan hệ . . . 10
III.2 Lược đồ toàn cục . . . 10

IV MÔ TẢ CSDL **11**
IV.1 Danh sách các bảng và thuộc tính . . . 11

V CÀI ĐẶT CSDL VÀ API TRÊN DJANGO **14**
V.1 Cài đặt môi trường . . . 14
V.2 Cài đặt API với DRF và WebSocket . . . 15
V.3 Cài đặt frontend với Vite và ReactJS . . . 16

VI THIẾT KẾ GIAO DIỆN NGƯỜI DÙNG **18**

VIIXÂY DỰNG CÁC TRUY VẤN VÀ API ĐÁP ỨNG YÊU CẦU CHỨC NĂNG **27**
VII.1 Hiển thị danh sách bài hát . . . 27
VII.2 Tìm kiếm bài hát theo tên, nghệ sĩ hoặc album . . . 27
VII.3 Thêm bài hát mới . . . 27
VII.4 Quản lý playlist (thêm bài hát vào playlist) . . . 28
VII.5 Hiển thị danh sách bài hát trong playlist . . . 28
VII.6 Gửi tin nhắn thời gian thực . . . 28
VII.7 Xóa bài hát khỏi playlist . . . 29
VII.8 Đăng ký người dùng mới . . . 29
VII.9 Đăng nhập người dùng . . . 29
VII.10 Hiển thị lịch sử trò chuyện . . . 29

VIIKẾT LUẬN **31**

IX TÀI LIỆU THAM KHẢO **32**

I MÔ TẢ VỀ ĐỀ TÀI

I.1 Khảo sát nghiệp vụ

Trong thời đại công nghệ số, các nền tảng nghe nhạc trực tuyến như Spotify đã trở thành công cụ phổ biến, đáp ứng nhu cầu giải trí và kết nối cộng đồng. Việc xây dựng một ứng dụng clone Spotify không chỉ giúp người dùng quản lý và thưởng thức âm nhạc mà còn cung cấp các tính năng như tạo playlist, tìm kiếm bài hát, và nhắn tin giữa các người dùng. Đề tài này tận dụng các công nghệ mã nguồn mở để phát triển một hệ thống hiệu quả, dễ mở rộng.

Nhóm đã quyết định chọn đề tài "Xây dựng ứng dụng clone Spotify" với mục tiêu:

- Phát triển một hệ thống web cho phép người dùng nghe nhạc, tạo playlist, và giao tiếp qua tin nhắn.
- Đảm bảo dữ liệu được lưu trữ logic, xử lý nhanh chóng, và đáp ứng các yêu cầu nghiệp vụ cơ bản của một nền tảng nghe nhạc.

Các chức năng chính của hệ thống bao gồm:

- Quản lý nghệ sĩ, album, bài hát: Hiển thị, thêm, sửa, xóa thông tin.
- Tìm kiếm bài hát: Tìm kiếm theo tiêu đề, nghệ sĩ, hoặc album.
- Quản lý playlist: Tạo, cập nhật, xóa playlist; thêm/xóa bài hát trong playlist.
- Quản lý tin nhắn: Gửi và nhận tin nhắn thời gian thực qua WebSocket.
- Xác thực người dùng: Đăng ký, đăng nhập, đăng xuất, và quản lý thông tin người dùng.
- Quản lý quyền admin: Cho phép superuser quản lý hệ thống.

I.2 Thông tin về đề tài

Đề tài sử dụng Django và Django REST Framework (DRF) cho backend, kết hợp Django Channels để hỗ trợ giao tiếp thời gian thực qua WebSocket. Cơ sở dữ liệu sử dụng MySQL thông qua XAMPP để đảm bảo hiệu suất tốt hơn so với SQLite mặc định. Frontend được xây dựng với ReactJS và Vite để tạo giao diện người dùng hiện đại, nhanh chóng, và responsive. Các công nghệ mã nguồn mở như Redis (cho Channels) và Bootstrap cũng được tích hợp để tăng cường tính năng và thẩm mỹ.

I.3 Phân công công việc

Bảng phân công công việc dưới đây mô tả các nhiệm vụ được phân bổ cho từng thành viên trong nhóm, cùng với tiến độ thực hiện theo tuần từ tuần 4 đến tuần 14:

STT	Thành viên	Tên công việc	Tuần										
			4	5	6	7	8	9	10	11	12	13	14
1	Nguyễn Trí Dũng	Thảo luận đề tài	x	x									
2	Nguyễn Trí Dũng	Khảo sát yêu cầu nghiệp vụ		x	x								

STT	Thành viên	Tên công việc	Tuần										
			4	5	6	7	8	9	10	11	12	13	14
3	Nguyễn Văn A	Thiết kế mô hình ERD			x	x							
4	Nguyễn Văn A	Thiết kế CSDL quan hệ				x	x						
5	Nguyễn Văn A	Mô tả CSDL và serializers					x	x					
6	Nguyễn Văn A	Xây dựng giao diện người dùng với ReactJS						x	x				
7	Lê Công Thành	Cài đặt backend và API với Django/-DRF							x	x			
8	Lê Công Thành	Cài đặt WebSocket với Django Channels								x	x		
9	Lê Công Thành	Tích hợp giao diện và backend									x	x	
10	Nguyễn Trí Dũng	Viết báo cáo và chuẩn bị thuyết trình										x	x

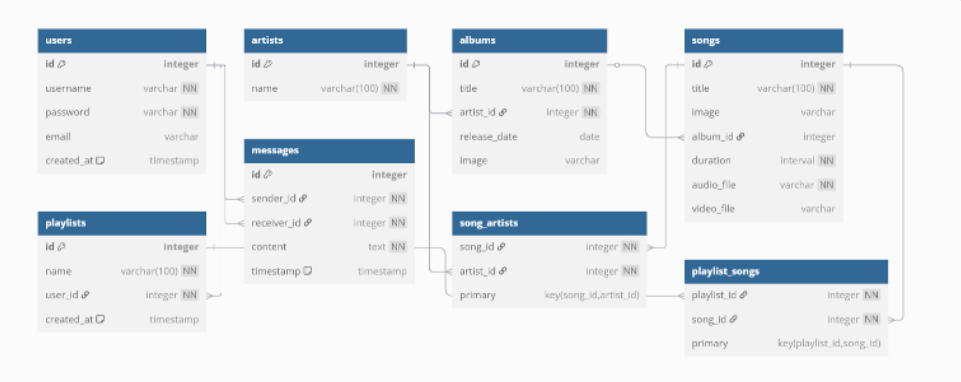
Bảng 1: Bảng phân công công việc của nhóm

II THIẾT KẾ CƠ SỞ DỮ LIỆU

II.1 Thiết kế ERD

Mô hình ERD (Entity-Relationship Diagram) thể hiện các thực thể chính: Artist, Album, Song, Playlist, User, và Message. Các mối quan hệ bao gồm:

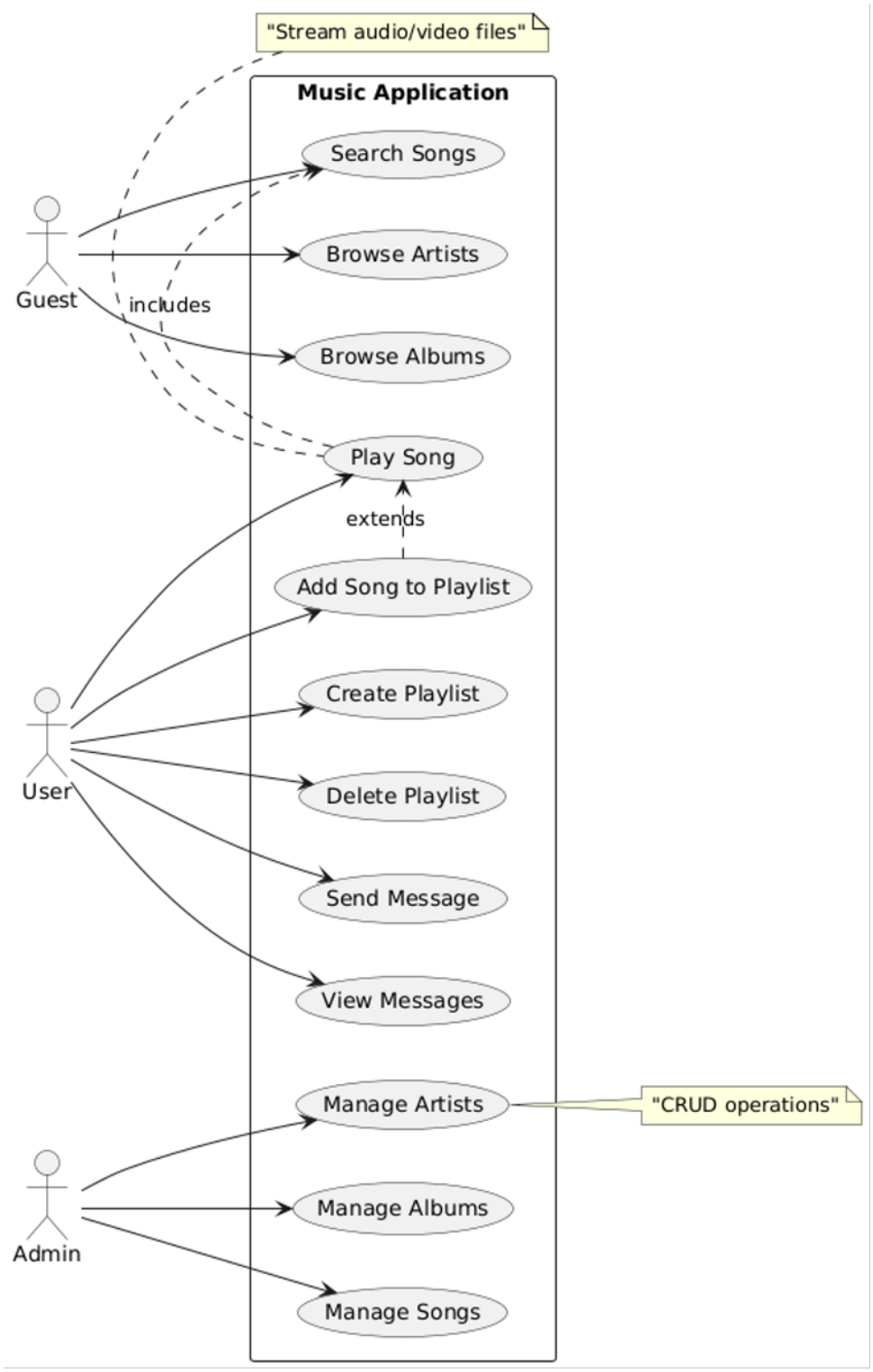
- Một Artist có nhiều Album (1:N).
- Một Album chứa nhiều Song (1:N), nhưng một Song có thể không thuộc Album nào.
- Một Song liên kết với nhiều Artist (N:N) thông qua bảng trung gian Song_Artist.
- Một User tạo nhiều Playlist (1:N), và một Playlist chứa nhiều Song (N:N) thông qua bảng trung gian Playlist_Song.
- Một User gửi và nhận nhiều Message (N:N).



Hình 1: Mô hình ERD của hệ thống

II.2 Mô tả Use Case

Sơ đồ Use Case minh họa các tương tác giữa Guest, User, Admin, và hệ thống:



Hình 2: Sơ đồ Use Case của ứng dụng

Các trường hợp sử dụng chính:

- **Xem danh sách nghệ sĩ**:
  - *Tác nhân*: Guest, User.
  - *Mô tả*: Guest hoặc User xem danh sách tất cả nghệ sĩ.

- *Điều kiện tiên quyết*: Có ít nhất một nghệ sĩ trong cơ sở dữ liệu.
- *Luồng chính*:
  1. Guest/User truy cập trang danh sách nghệ sĩ.
  2. Hệ thống truy xuất danh sách nghệ sĩ từ bảng `Artist`.
  3. Hệ thống hiển thị danh sách nghệ sĩ (tên, hình ảnh nếu có).
- *Luồng phụ*: Nếu không có nghệ sĩ, hiển thị thông báo "Không có nghệ sĩ".
- *Kết quả*: Guest/User thấy danh sách nghệ sĩ.

- **Xem danh sách album**:
  - *Tác nhân*: Guest, User.
  - *Mô tả*: Guest hoặc User xem danh sách album.
  - *Điều kiện tiên quyết*: Có ít nhất một album trong cơ sở dữ liệu.
  - *Luồng chính*:
    1. Guest/User truy cập trang danh sách album.
    2. Hệ thống truy xuất danh sách album từ bảng `Album`.
    3. Hệ thống hiển thị danh sách album (tiêu đề, nghệ sĩ, hình ảnh nếu có).
  - *Luồng phụ*: Nếu không có album, hiển thị thông báo "Không có album".
  - *Kết quả*: Guest/User thấy danh sách album.

- **Xem chi tiết bài hát**:
  - *Tác nhân*: Guest, User.
  - *Mô tả*: Guest hoặc User xem thông tin chi tiết bài hát (tên, nghệ sĩ, album, thời lượng, hình ảnh).
  - *Điều kiện tiên quyết*: Có ít nhất một bài hát trong cơ sở dữ liệu.
  - *Luồng chính*:
    1. Guest/User chọn bài hát từ danh sách.
    2. Hệ thống truy xuất thông tin từ bảng `Song`.
    3. Hệ thống hiển thị chi tiết bài hát.
  - *Luồng phụ*: Nếu bài hát không tồn tại, hiển thị thông báo lỗi.
  - *Kết quả*: Guest/User thấy thông tin chi tiết bài hát.

- **Đăng nhập**:
  - *Tác nhân*: User.
  - *Mô tả*: User đăng nhập để sử dụng các tính năng đã xác thực.
  - *Điều kiện tiên quyết*: User có tài khoản trong bảng `User`.
  - *Luồng chính*:
    1. User truy cập trang đăng nhập.
    2. User nhập tên đăng nhập và mật khẩu.
    3. Hệ thống xác thực thông tin với bảng `User`.

4. Hệ thống cấp phiên đăng nhập và chuyển hướng đến trang chính.

– *Luồng phụ*:

* Nếu thông tin sai, hiển thị thông báo lỗi.
* Nếu tài khoản bị khóa, hiển thị "Tài khoản bị khóa".

– *Kết quả*: User đăng nhập thành công.

- **Tạo playlist**:

– *Tác nhân*: User.

– *Mô tả*: User tạo playlist mới để lưu bài hát yêu thích.

– *Điều kiện tiên quyết*: User đã đăng nhập.

– *Luồng chính*:

1. User chọn "Tạo playlist" trên giao diện.
2. User nhập tên playlist.
3. Hệ thống lưu playlist vào bảng `Playlist` với thông tin user.
4. Hệ thống hiển thị thông báo xác nhận.

– *Luồng phụ*:

* Nếu tên playlist trùng, yêu cầu nhập tên khác.
* Nếu tên không hợp lệ (rỗng, quá dài), hiển thị thông báo lỗi.

– *Kết quả*: Playlist mới được tạo và liên kết với User.

- **Thêm bài hát vào playlist**:

– *Tác nhân*: User.

– *Mô tả*: User thêm bài hát vào playlist đã tạo.

– *Điều kiện tiên quyết*: User đã đăng nhập, có ít nhất một playlist và bài hát.

– *Luồng chính*:

1. User chọn playlist từ danh sách.
2. User chọn bài hát từ danh sách bài hát.
3. User nhấn "Thêm vào playlist".
4. Hệ thống cập nhật bảng `Playlist`$_{S}$*ong.Hthnghinththngboxcnhn*.

5. *Luồng phụ*: Nếu bài hát đã có trong playlist, hiển thị "Bài hát đã tồn tại".

– *Kết quả*: Bài hát được thêm vào playlist.

- **Nghe bài hát**:

– *Tác nhân*: User.

– *Mô tả*: User phát bài hát từ hệ thống.

– *Điều kiện tiên quyết*: User đã đăng nhập, bài hát có `audio`$_{f}$*ile.Luồng chính*:

– – User chọn bài hát từ danh sách hoặc playlist.

– User nhấn nút "Phát".

– Hệ thống truy xuất `Song.audio`$_{f}$*ile.Hthngphtbihttrngiaodin*.

- *Luồng phụ*: Nếu file âm thanh không tồn tại, hiển thị thông báo lỗi.
- *Kết quả*: User nghe được bài hát.

Gửi tin nhắn:

- *Tác nhân*: User.
- *Mô tả*: User gửi tin nhắn đến User khác.
- *Điều kiện tiên quyết*: User đã đăng nhập, người nhận hợp lệ.
- *Luồng chính*:
  1. User truy cập giao diện nhắn tin.
  2. User chọn người nhận từ danh sách.
  3. User nhập nội dung tin nhắn.
  4. User nhấn "Gửi".
  5. Hệ thống lưu tin nhắn vào bảng `Message`.
  6. Hệ thống thông báo gửi thành công.
- *Luồng phụ*:
  - Nếu nội dung rỗng, yêu cầu nhập nội dung.
  - Nếu người nhận không tồn tại, hiển thị thông báo lỗi.
- *Kết quả*: Tin nhắn được gửi và lưu.

Tìm kiếm:

- *Tác nhân*: Guest, User.
- *Mô tả*: Guest hoặc User tìm kiếm bài hát, album, nghệ sĩ theo từ khóa.
- *Điều kiện tiên quyết*: Hệ thống có dữ liệu để tìm kiếm.
- *Luồng chính*:
  1. Guest/User nhập từ khóa vào thanh tìm kiếm.
  2. Hệ thống truy vấn bảng `Artist`, `Album`, `Song`.
  3. Hệ thống hiển thị kết quả tìm kiếm.
- *Luồng phụ*: Nếu không có kết quả, hiển thị "Không tìm thấy kết quả".
- *Kết quả*: Guest/User thấy danh sách kết quả tìm kiếm.

Quản lý nghệ sĩ (Thêm/Sửa/Xóa):

- *Tác nhân*: Admin.
- *Mô tả*: Admin thêm, sửa, xóa thông tin nghệ sĩ.
- *Điều kiện tiên quyết*: Admin đã đăng nhập, có quyền quản trị.
- *Luồng chính (Thêm)*:
  1. Admin truy cập giao diện quản lý nghệ sĩ.
  2. Admin nhập tên nghệ sĩ mới.
  3. Hệ thống kiểm tra tính duy nhất của `Artist.name`.
  4. Hệ thống lưu nghệ sĩ vào bảng `Artist`.
  5. Hệ thống hiển thị thông báo xác nhận.

- *Luồng chính (Sửa)*:
  1. Admin chọn nghệ sĩ từ danh sách.
  2. Admin cập nhật tên nghệ sĩ.
  3. Hệ thống lưu thay đổi vào bảng `Artist`.
- *Luồng chính (Xóa)*:
  1. Admin chọn nghệ sĩ.
  2. Admin xác nhận xóa.
  3. Hệ thống xóa nghệ sĩ và album liên quan ($\texttt{on}_d elete = models.CASCADE$).
- *Luồng phụ*:
  - Nếu tên nghệ sĩ trùng, yêu cầu nhập tên khác.
  - Nếu nghệ sĩ không tồn tại, hiển thị thông báo lỗi.
- *Kết quả*: Thông tin nghệ sĩ được thêm, sửa, hoặc xóa thành công.

Quản lý album (Thêm/Sửa/Xóa):

- *Tác nhân*: Admin.
- *Mô tả*: Admin thêm, sửa, xóa thông tin album.
- *Điều kiện tiên quyết*: Admin đã đăng nhập, có ít nhất một nghệ sĩ.
- *Luồng chính (Thêm)*:
  1. Admin truy cập giao diện quản lý album.
  2. Admin nhập tiêu đề, chọn nghệ sĩ, ngày phát hành, tải ảnh (nếu có).
  3. Hệ thống lưu album vào bảng `Album`.
  4. Hệ thống hiển thị thông báo xác nhận.
- *Luồng chính (Sửa/Xóa)*: Tương tự quản lý nghệ sĩ.
- *Luồng phụ*: Nếu nghệ sĩ không tồn tại, yêu cầu chọn nghệ sĩ hợp lệ.
- *Kết quả*: Album được thêm, sửa, hoặc xóa thành công.

III THIẾT KẾ LƯỢC ĐỒ CSDL QUAN HỆ

III.1 Lược đồ CSDL quan hệ

Dựa trên mô hình ERD, lược đồ CSDL quan hệ bao gồm:

- `Artist`: (id, name)
- `Album`: (id, title, artist_id, release_date, image)
- `Song`: (id, title, image, album_id, duration, audio_file)
- `Song_Artist`: (song_id, artist_id)
- `Playlist`: (id, name, user_id, created_at)
- `Playlist_Song`: (playlist_id, song_id)
- `User`: (id, username, password, email)
- `Message`: (id, sender_id, receiver_id, content, timestamp)

III.2 Lược đồ toàn cục

- `Artist`: (id [PK], name [UNIQUE])
- `Album`: (id [PK], title, artist_id [FK → Artist], release_date, image)
- `Song`: (id [PK], title, image, album_id [FK → Album, NULLABLE], duration, audio_file)
- `Song_Artist`: (song_id [FK → Song], artist_id [FK → Artist], [PK: song_id, artist_id])
- `Playlist`: (id [PK], name, user_id [FK → User], created_at)
- `Playlist_Song`: (playlist_id [FK → Playlist], song_id [FK → Song], [PK: playlist_id, song_id])
- `User`: (id [PK], username, password, email)
- `Message`: (id [PK], sender_id [FK → User], receiver_id [FK → User], content, timestamp)

IV MÔ TẢ CSDL

IV.1 Danh sách các bảng và thuộc tính

Danh sách các bảng

STT	Bảng	Ý nghĩa	Ghi chú
1	Artist	Lưu thông tin nghệ sĩ (tên nghệ sĩ)	
2	Album	Lưu thông tin album (tên, nghệ sĩ, ngày phát hành, ảnh bìa)	
3	Song	Lưu thông tin bài hát (tên, album, thời lượng, file âm thanh, ảnh)	
4	Song_Artist	Lưu mối quan hệ nhiều-nhiều giữa bài hát và nghệ sĩ	
5	Playlist	Lưu thông tin playlist của người dùng (tên, người tạo, thời gian tạo)	
6	Playlist_Song	Lưu mối quan hệ nhiều-nhiều giữa playlist và bài hát	
7	User	Lưu thông tin người dùng (tên đăng nhập, email, mật khẩu)	
8	Message	Lưu thông tin tin nhắn giữa các người dùng	

Bảng 2: Danh sách các bảng trong cơ sở dữ liệu

Danh sách thuộc tính
tbl_Artist

STT	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	id	Integer	Primary Key	Mã nghệ sĩ
2	name	Varchar(100)	Unique	Tên nghệ sĩ

Bảng 3: Thuộc tính bảng Artist

tbl_Album

STT	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	id	Integer	Primary Key	Mã album
2	title	Varchar(100)		Tên album
3	artist_id	Integer	Foreign Key (Artist)	Mã nghệ sĩ
4	release_date	Date	Nullable	Ngày phát hành
5	image	ImageField	Nullable	Ảnh bìa album

Bảng 4: Thuộc tính bảng Album

tbl_Song

STT	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	id	Integer	Primary Key	Mã bài hát
2	title	Varchar(100)		Tên bài hát
3	image	ImageField	Nullable	Ảnh bài hát
4	album_id	Integer	Foreign Key (Album, Nullable)	Mã album
5	duration	DurationField	Nullable	Thời lượng bài hát
6	audio_file	FileField		File âm thanh

Bảng 5: Thuộc tính bảng Song

tbl_Song_Artist

STT	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	song_id	Integer	Foreign Key (Song)	Mã bài hát
2	artist_id	Integer	Foreign Key (Artist)	Mã nghệ sĩ

Bảng 6: Thuộc tính bảng Song_Artist

tbl_Playlist

STT	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	id	Integer	Primary Key	Mã playlist
2	name	Varchar(100)		Tên playlist
3	user_id	Integer	Foreign Key (User)	Mã người dùng
4	created_at	DateTime		Thời gian tạo

Bảng 7: Thuộc tính bảng Playlist

tbl_Playlist_Song

STT	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	playlist_id	Integer	Foreign Key (Playlist)	Mã playlist
2	song_id	Integer	Foreign Key (Song)	Mã bài hát

Bảng 8: Thuộc tính bảng Playlist_Song

tbl_User

STT	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	id	Integer	Primary Key	Mã người dùng
2	username	Varchar(150)	Unique	Tên đăng nhập
3	password	Varchar		Mật khẩu
4	email	Varchar(254)	Unique	Email

Bảng 9: Thuộc tính bảng User

tbl_Message

STT	Thuộc tính	Kiểu	Ràng buộc	Ghi chú
1	id	Integer	Primary Key	Mã tin nhắn
2	sender_id	Integer	Foreign Key (User)	Mã người gửi
3	receiver_id	Integer	Foreign Key (User)	Mã người nhận
4	content	Text		Nội dung tin nhắn
5	timestamp	DateTime		Thời gian gửi

Bảng 10: Thuộc tính bảng Message

V CÀI ĐẶT CSDL VÀ API TRÊN DJANGO

V.1 Cài đặt môi trường

- **Cài đặt XAMPP và MySQL**:
  - Cài đặt XAMPP để sử dụng MySQL làm cơ sở dữ liệu.
  - Khởi động MySQL trong XAMPP Control Panel và tạo database `spotify` qua phpMyAdmin hoặc MySQL CLI.
- **Cài đặt Python và các thư viện**:
  - Cài Python phiên bản mới nhất.
  - Cài các thư viện cần thiết:

```
pip install django djangorestframework django-cors-headers python-
    decouple pillow django-channels channels-redis mysqlclient
```

- **Tạo dự án Django**:
  - Chạy lệnh:

```
django-admin startproject backend
python manage.py startapp api
```

- **Cấu hình CSDL và DRF**:
  - Cấu hình MySQL trong `settings.py`:

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.mysql',
      'NAME': 'spotify',
      'USER': 'root',
      'PASSWORD': '',
      'HOST': 'localhost',
      'PORT': '3306',
   }
}
```

  - Cấu hình DRF và Channels:

```
INSTALLED_APPS = [
   'daphne',
   'django.contrib.admin',
   'django.contrib.auth',
   'django.contrib.contenttypes',
   'django.contrib.sessions',
   'django.contrib.messages',
   'django.contrib.staticfiles',
   'rest_framework',
   'rest_framework_simplejwt',
   'corsheaders',
   'channels',
   'api',
]
```

```
REST_FRAMEWORK = {
   'DEFAULT_AUTHENTICATION_CLASSES': [
      'rest_framework_simplejwt.authentication.JWTAuthentication',
   ]
}
ASGI_APPLICATION = 'backend.asgi.application'
CHANNEL_LAYERS = {
   'default': {
      'BACKEND': 'channels_redis.core.RedisChannelLayer',
      'CONFIG': {
         'hosts': [('127.0.0.1', 6379)],
      },
   },
}
CORS_ALLOWED_ORIGINS = [
   'http://127.0.0.1:3000',
   'http://localhost:3000',
   'http://localhost:5176',
   'http://localhost:5174',
   'http://localhost:5173',
   'http://localhost:8000',
]
```

  - Cấu hình media để lưu trữ file:

```
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'
```

- **Tạo mô hình CSDL**:
  - Định nghĩa các model trong `api/models.py`.
  - Chạy lệnh:

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

- **Cài đặt Redis**:
  - Cài Redis để hỗ trợ Channels cho WebSocket.
  - Chạy Redis server:

```
redis-server
```

- **Tạo superuser**:
  - Chạy lệnh:

```
python manage.py createsuperuser
```

V.2 Cài đặt API với DRF và WebSocket

- **Serializers**: Định nghĩa trong `api/serializers.py` để chuyển đổi dữ liệu model thành JSON và ngược lại. Ví dụ, SongSerializer xử lý định dạng thời lượng (duration) thành HH:MM:SS và xác thực dữ liệu đầu vào.

- **Views**: Định nghĩa trong `api/views.py` để xử lý yêu cầu HTTP. Sử dụng Generic Views của DRF cho CRUD và APIView cho các chức năng tùy chỉnh như tìm kiếm bài hát hoặc quản lý playlist.
- **WebSocket**: Sử dụng Django Channels trong `api/consumers.py` để hỗ trợ nhắn tin thời gian thực. ChatConsumer xác thực người dùng qua JWT và gửi tin nhắn tới các group dựa trên user_id.
- **URLs**: Cấu hình trong `api/urls.py` để ánh xạ các endpoint API:

```
from django.urls import path
from . import views
urlpatterns = [
   path('todos/', views.TodoList.as_view(), name='todo-list'),
   path('artists/', views.ArtistList.as_view(), name='artist-list'),
   path('artists/<int:pk>/', views.ArtistDetail.as_view(), name='
      artist-detail'),
   path('albums/', views.AlbumList.as_view(), name='album-list'),
   path('albums/<int:pk>/', views.AlbumDetail.as_view(), name='album')
      ,
   path('albums/<int:album_id>/add_songs/', views.AddSongToAlbumView.
      as_view(), name='add-songs-to-album'),
   path('albums/<int:album_id>/songs/', views.AlbumSongsView.as_view()
      , name='album-songs'),
   path('songs/', views.SongList.as_view(), name='song-list'),
   path('songs/<int:pk>/', views.SongDetail.as_view(), name='song'),
   path('search/', views.SongSearchAPI.as_view(), name='search'),
   path('playlists/', views.PlaylistAPI.as_view(), name='playlist-list-
      create'),
   path('playlists/<int:pk>/', views.PlaylistDetailAPI.as_view(), name
      ='playlist-detail'),
   path('playlists/<int:pk>/songs/', views.PlaylistSongAPI.as_view(),
      name='playlist-song'),
   path('auth/login/', views.LoginAPI.as_view(), name='login'),
   path('auth/logout/', views.LogoutAPI.as_view(), name='logout'),
   path('auth/user/', views.UserInfoAPI.as_view(), name='user-info'),
   path('auth/register/', views.RegisterAPI.as_view(), name='register')
      ,
   path('users/', views.UserListAPI.as_view(), name='user-list'),
   path('recent-chats/', views.RecentChatsAPI.as_view(), name='recent-
      chats'),
   path('messages/<str:receiver_username>/', views.MessageHistoryAPI.
      as_view(), name='message-history'),
]
```

V.3 Cài đặt frontend với Vite và ReactJS

- **Tạo dự án React với Vite**:
  - Chạy lệnh:

```
npm create vite@latest spotify-frontend --template react
npm install axios react-router-dom bootstrap
```

- **Cấu hình CORS**:

– Trong `settings.py`, cấu hình $\texttt{CORS}_{A}LLOWED_{O}RIGINSØchophpfrontend(localhost:5173hocccngkhc)truyc\l pAPI$.

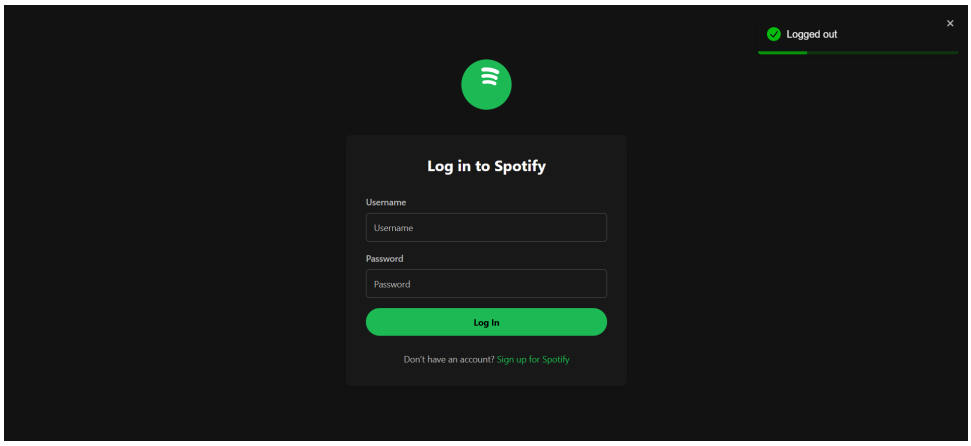
- **Tích hợp API**:
  - Sử dụng axios để gọi các endpoint API từ backend.
  - Xử lý xác thực JWT bằng cách lưu token trong localStorage và gửi trong header Authorization.
- **Tích hợp WebSocket**:
  - Sử dụng WebSocket API trong React để kết nối với ChatConsumer.

– Gửi token JWT trong query string: `ws://localhost:8000/ws/chat/?token=<`$\texttt{jwt}_{t}oken>$.

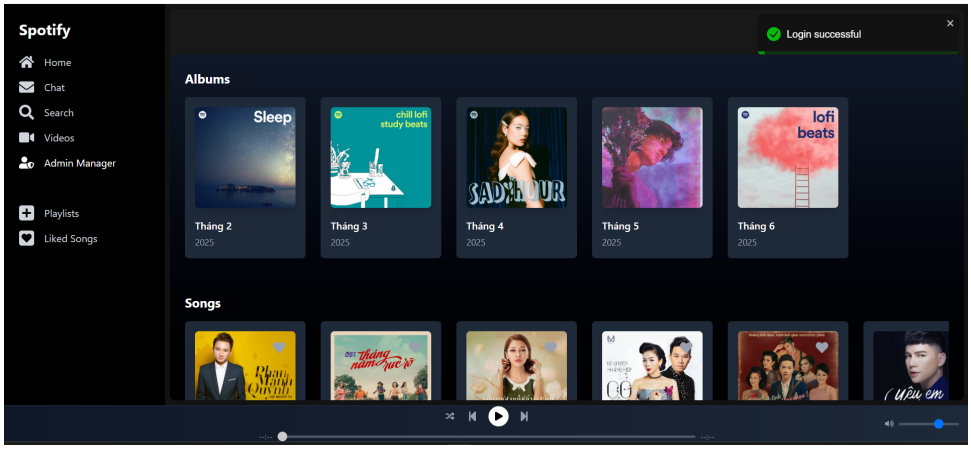
VI THIẾT KẾ GIAO DIỆN NGƯỜI DÙNG

Giao diện người dùng được xây dựng bằng ReactJS với Vite, tận dụng các component tái sử dụng và Bootstrap để đảm bảo tính responsive. Các màn hình chính bao gồm danh sách bài hát, tìm kiếm, quản lý playlist, nhắn tin, và trang đăng nhập/đăng ký. React Router được sử dụng để điều hướng giữa các trang.

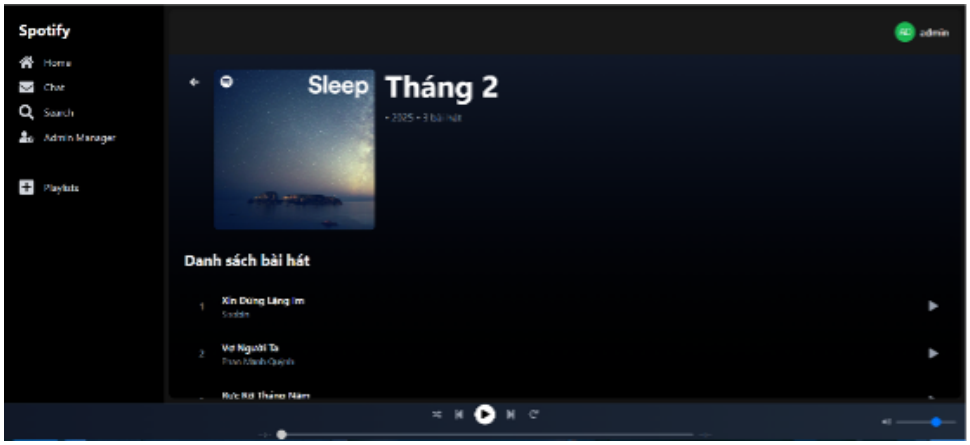
Các hình ảnh minh họa giao diện chính:



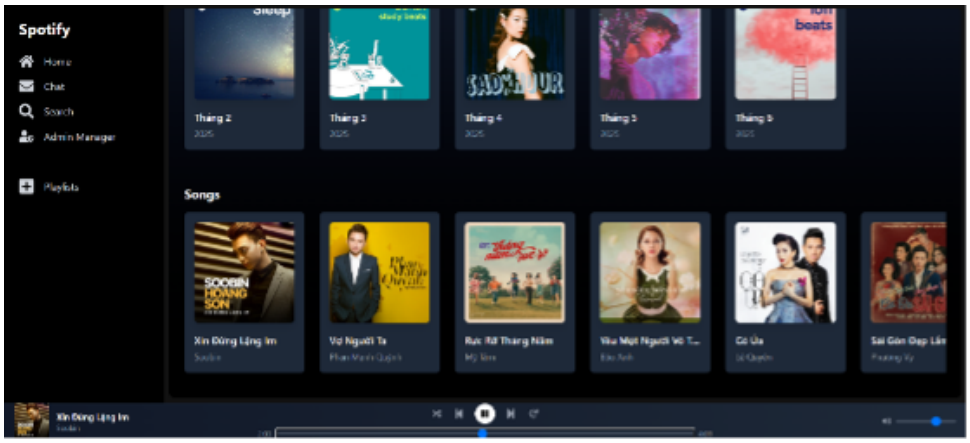
Hình 3: Giao diện trang đăng nhập



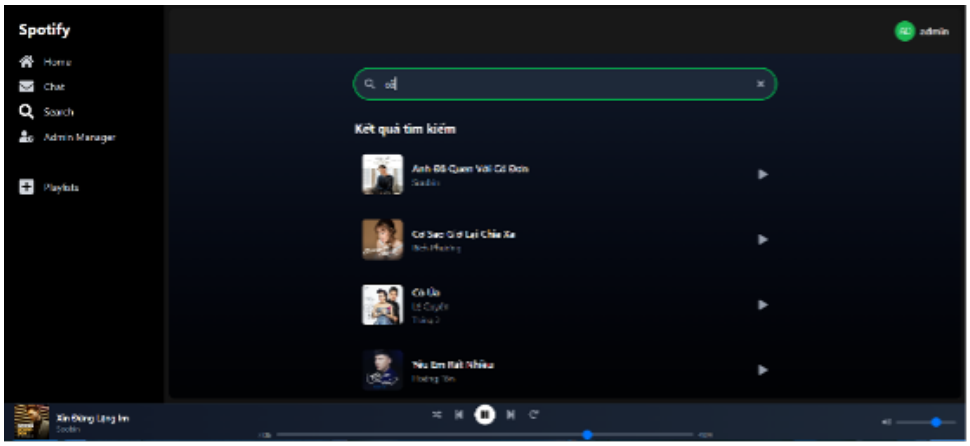
Hình 4: Giao diện trang chính



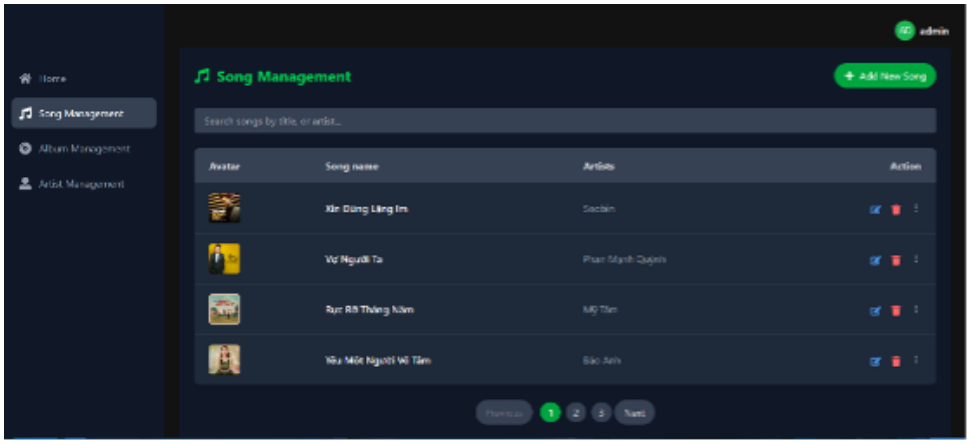
Hình 5: Giao diện trang album



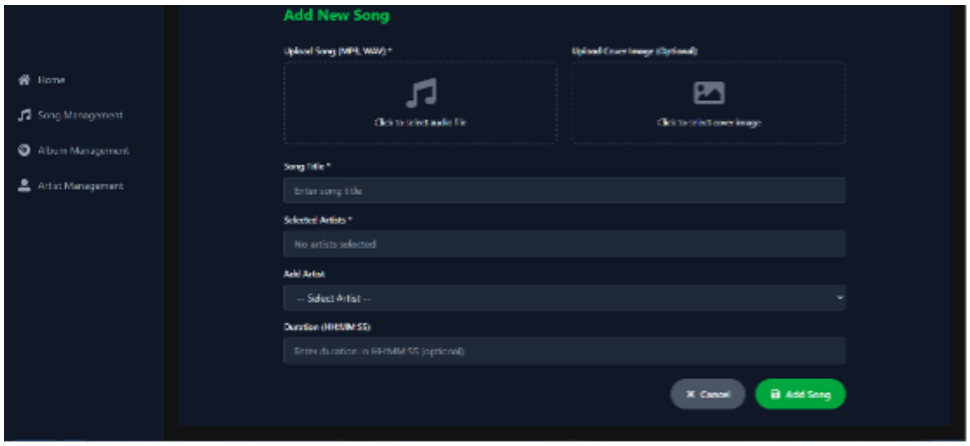
Hình 6: Chức năng phát nhạc, tua thời gian



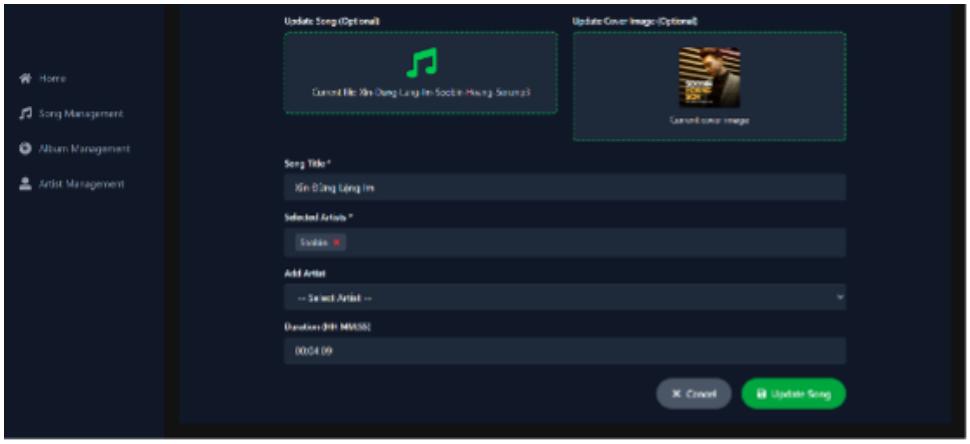
Hình 7: Giao diện tìm kiếm



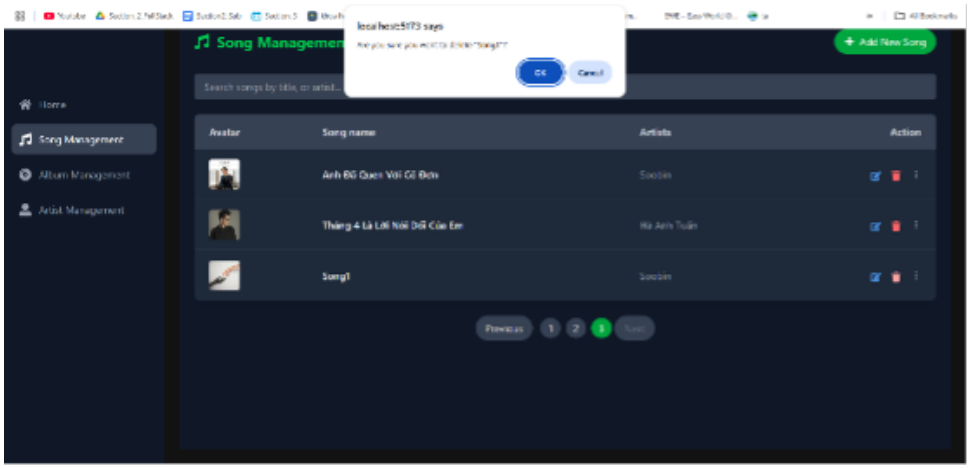
Hình 8: Giao diện quản lý bài hát



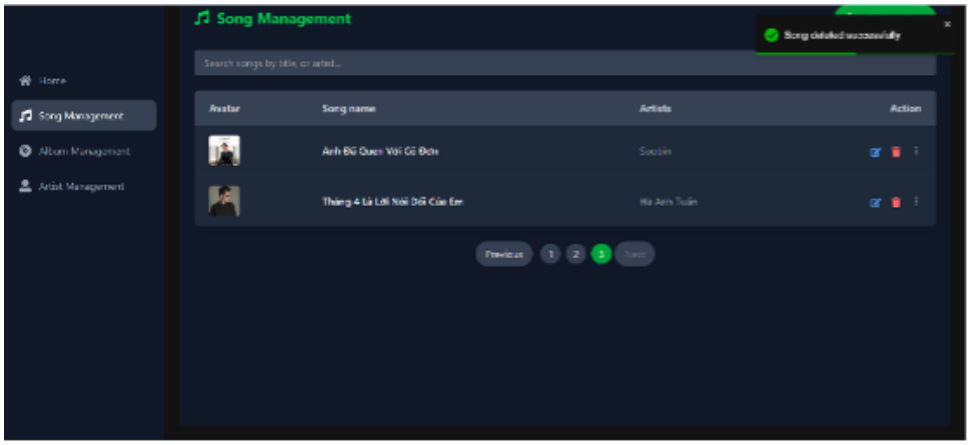
Hình 9: Giao diện thêm bài hát



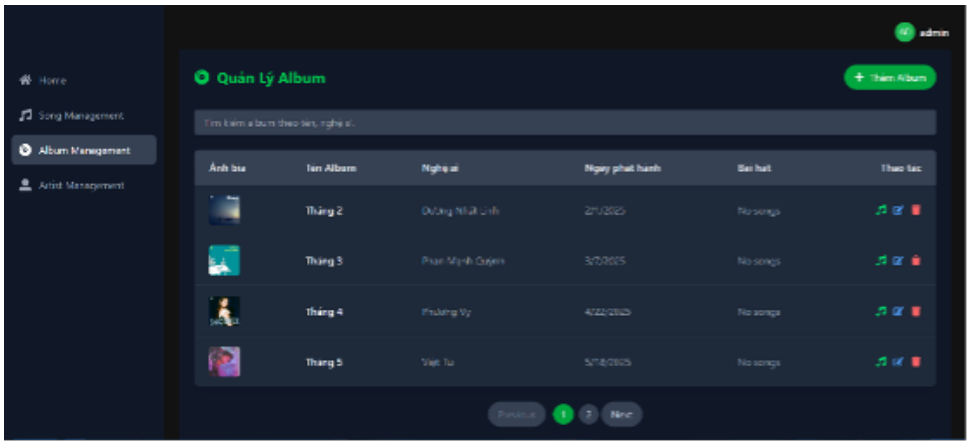
Hình 10: Giao diện chỉnh sửa bài hát



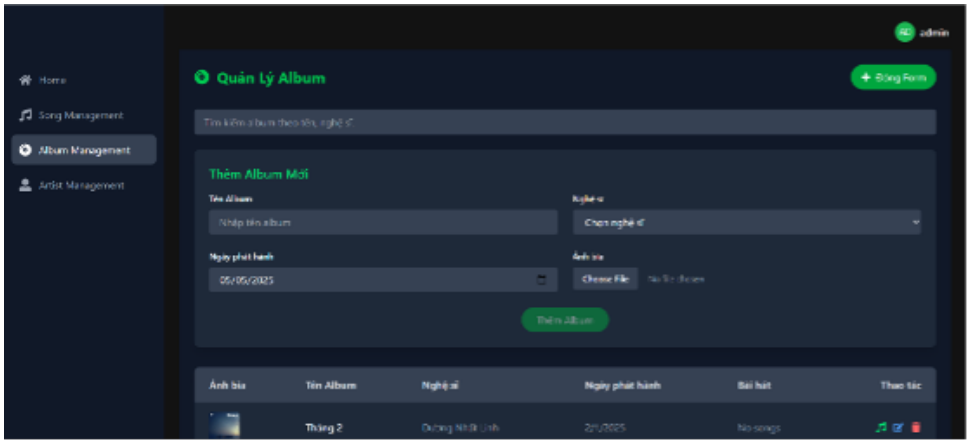
Hình 11: Giao diện xóa bài hát



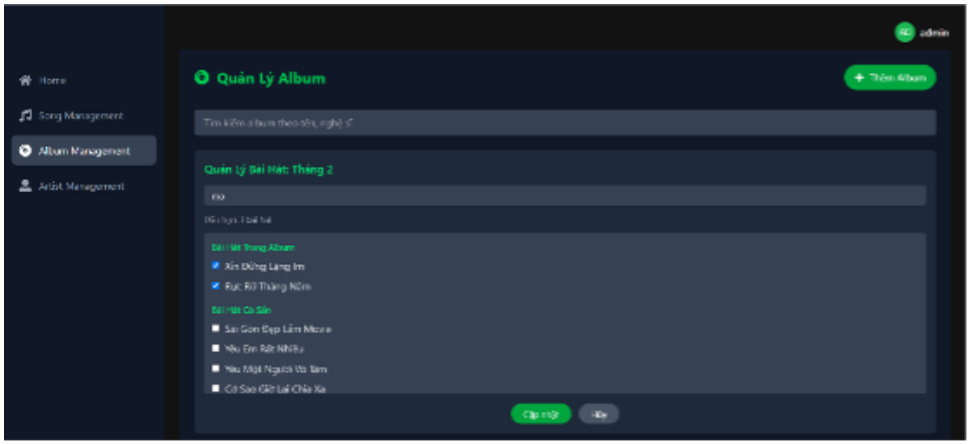
Hình 12: Kết quả chức năng xóa bài hát



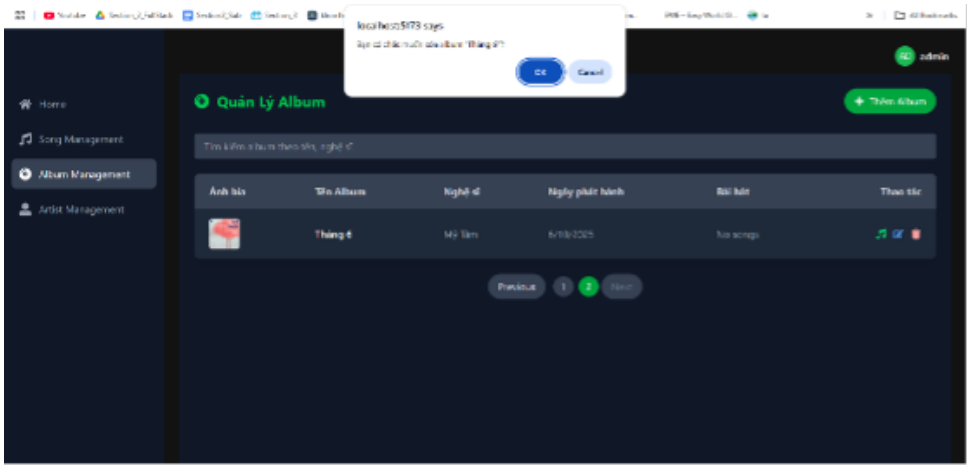
Hình 13: Giao diện quản lý album



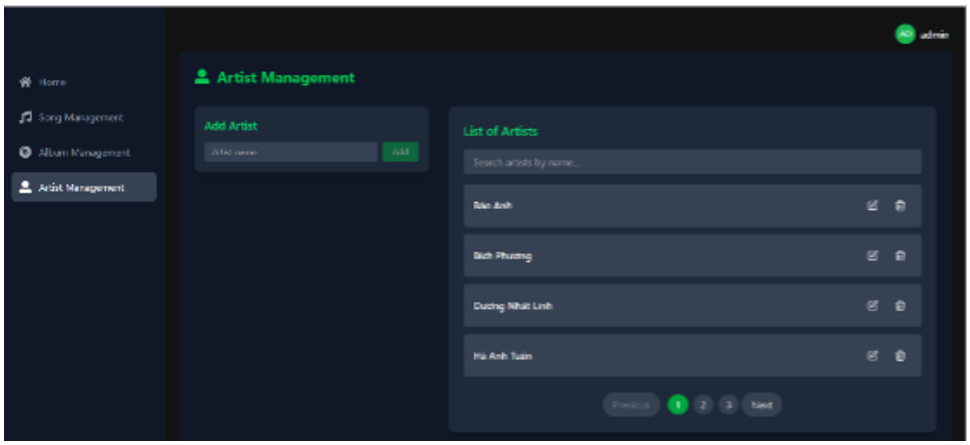
Hình 14: Giao diện thêm album



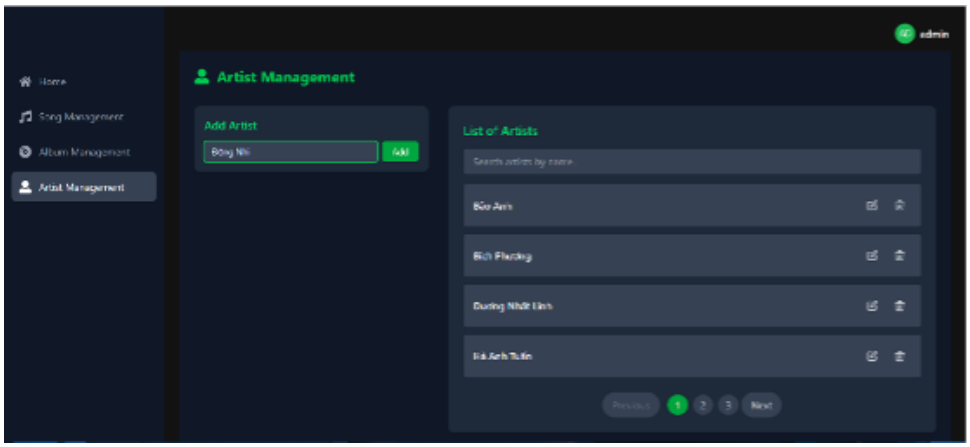
Hình 15: Giao diện thêm, tìm kiếm bài hát trong album



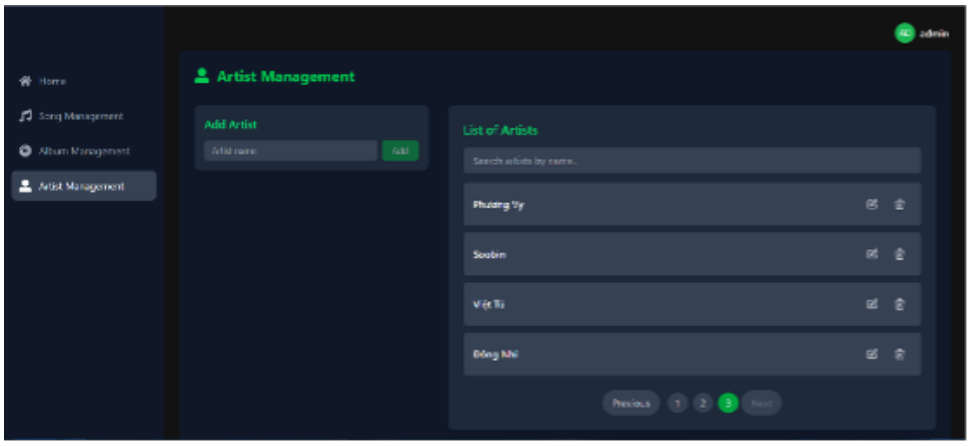
Hình 16: Giao diện xóa album



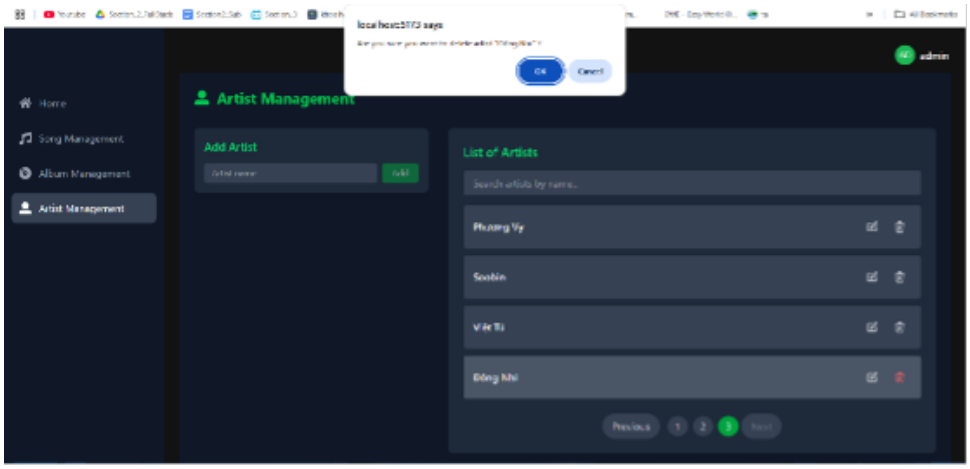
Hình 17: Giao diện quản lý nghệ sĩ



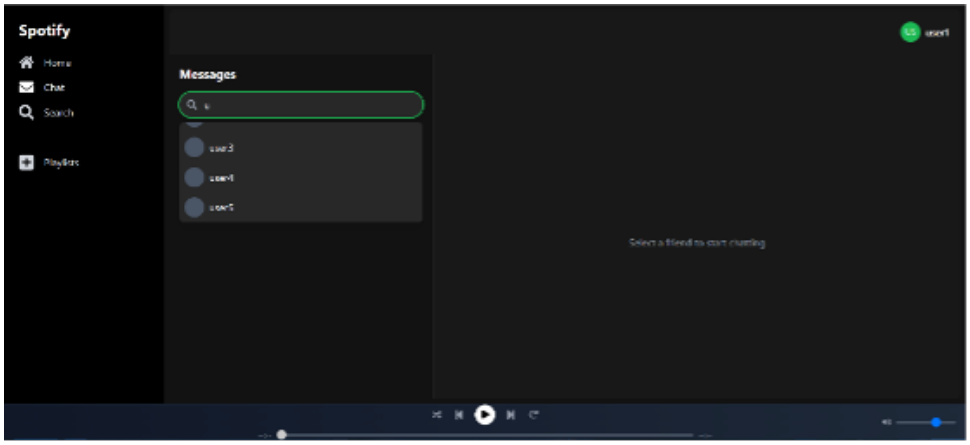
Hình 18: Chức năng thêm nghệ sĩ



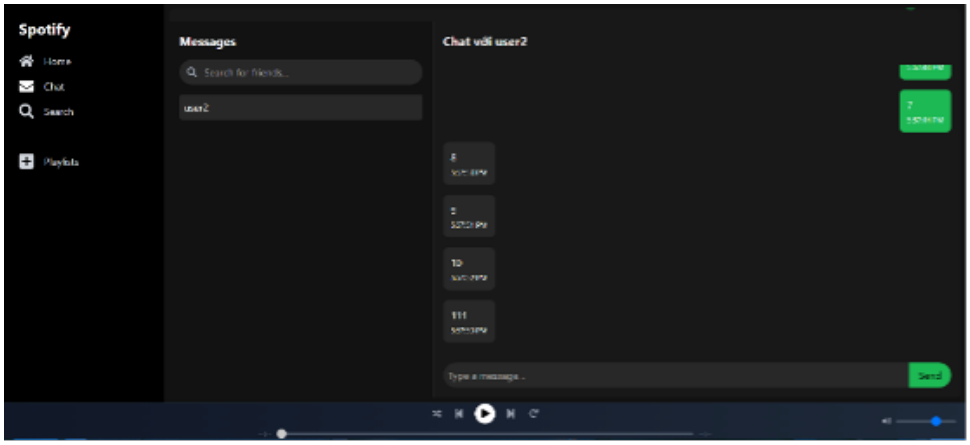
Hình 19: Kết quả sau khi thêm nghệ sĩ



Hình 20: Giao diện xóa nghệ sĩ



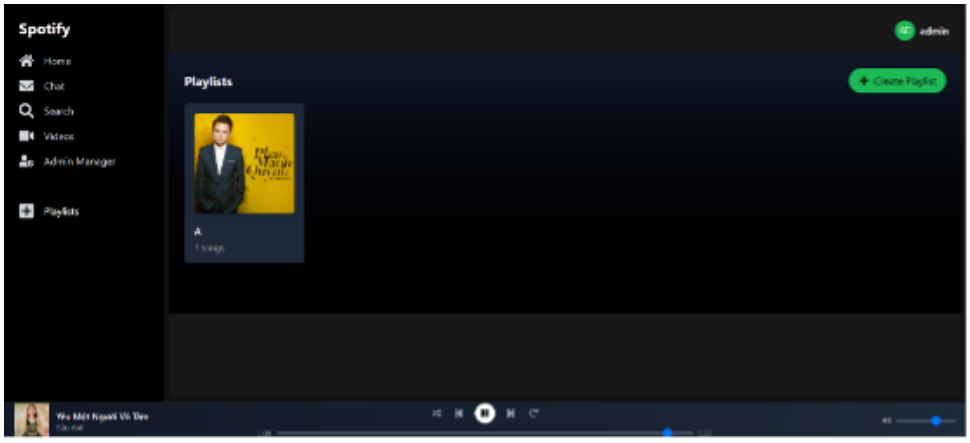
Hình 21: Giao diện tìm kiếm người dùng khác



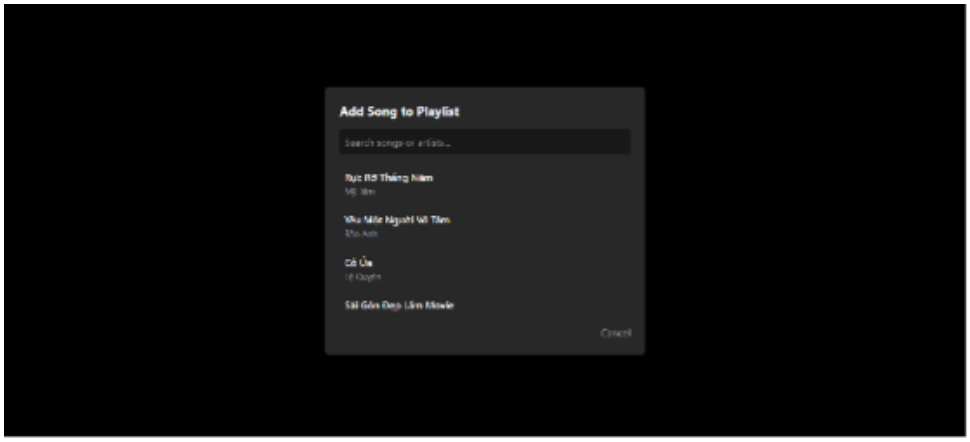
Hình 22: Giao diện chat với người dùng khác

Hình 23: Giao diện phát video âm nhạc

Hình 24: Giao diện tải video âm nhạc



Hình 25: Giao diện tạo playlist



Hình 26: Giao diện thêm bài hát vào playlist

VII XÂY DỰNG CÁC TRUY VẤN VÀ API ĐÁP ỨNG YÊU CẦU CHỨC NĂNG

VII.1 Hiển thị danh sách bài hát

API Endpoint: GET /api/songs/
Truy vấn Django ORM:

```
from api.models import Song
songs = Song.objects.all()
serializer = SongSerializer(songs, many=True)
```

Kết quả: Trả về danh sách tất cả bài hát dưới dạng JSON.

VII.2 Tìm kiếm bài hát theo tên, nghệ sĩ hoặc album

API Endpoint: GET /api/search/?q={query}
Truy vấn Django ORM:

```
from api.models import Song
from django.db.models import Q
search_query = request.GET.get('q', '').strip()
songs = Song.objects.filter(
   Q(title__icontains=search_query) |
   Q(artists__name__icontains=search_query) |
   Q(album__title__icontains=search_query)
).distinct()[:20]
serializer = SongSerializer(songs, many=True)
```

Kết quả: Trả về tối đa 20 bài hát phù hợp với từ khóa tìm kiếm.

VII.3 Thêm bài hát mới

API Endpoint: POST /api/songs/
Truy vấn Django ORM:

```
from api.models import Song, Album, Artist
from datetime import timedelta
data = request.data
album = Album.objects.get(id=data.get('album_id')) if data.get('album_id')
    else None
artist_ids = data.get('artist_ids', [])
duration_str = data.get('duration', None)
duration = None
if duration_str:
   hours, minutes, seconds = map(int, duration_str.split(':'))
   duration = timedelta(hours=hours, minutes=minutes, seconds=seconds)
song = Song.objects.create(
   title=data['title'],
   image=data.get('image'),
   album=album,
   duration=duration,
   audio_file=data.get('audio_file')
)
song.artists.set(Artist.objects.filter(id__in=artist_ids))
```

```
serializer = SongSerializer(song)
```

Kết quả: Tạo bài hát mới và trả về thông tin bài hát dưới dạng JSON.

VII.4 Quản lý playlist (thêm bài hát vào playlist)

API Endpoint: POST /api/playlists/{pk}/songs/
Truy vấn Django ORM:

```
from api.models import Playlist, Song
playlist = Playlist.objects.get(pk=pk, user=request.user)
song = Song.objects.get(id=request.data.get('song_id'))
playlist.songs.add(song)
serializer = PlaylistSerializer(playlist)
```

Kết quả: Thêm bài hát vào playlist và trả về thông tin playlist cập nhật.

VII.5 Hiển thị danh sách bài hát trong playlist

API Endpoint: GET /api/playlists/{pk}/songs/
Truy vấn Django ORM:

```
from api.models import Playlist
playlist = Playlist.objects.get(pk=pk, user=request.user)
songs = playlist.songs.all()
serializer = SongSerializer(songs, many=True)
```

Kết quả: Trả về danh sách bài hát trong playlist.

VII.6 Gửi tin nhắn thời gian thực

WebSocket: ws://localhost:8000/ws/chat/?token=<jwt_token>
Truy vấn Django ORM:

```
from api.models import Message
from django.contrib.auth.models import User
receiver = User.objects.get(username=text_data_json['receiver'])
msg = Message.objects.create(
   sender=self.user,
   receiver=receiver,
   content=text_data_json['message']
)
data = {
   'content': msg.content,
   'sender': self.user.username,
   'receiver': receiver.username,
   'timestamp': msg.timestamp.isoformat(),
}
await self.channel_layer.group_send(f'chat_{receiver.id}', {'type': '
    chat_message', **data})
await self.channel_layer.group_send(f'chat_{self.user.id}', {'type': '
    chat_message', **data})
```

Kết quả: Gửi tin nhắn thời gian thực tới người gửi và người nhận.

VII.7 Xóa bài hát khỏi playlist

API Endpoint: DELETE /api/playlists/{pk}/songs/

Truy vấn Django ORM:

```
from api.models import Playlist, Song
playlist = Playlist.objects.get(pk=pk, user=request.user)
song = Song.objects.get(id=request.data.get('song_id'))
playlist.songs.remove(song)
serializer = PlaylistSerializer(playlist)
```

Kết quả: Xóa bài hát khỏi playlist và trả về thông tin playlist cập nhật.

VII.8 Đăng ký người dùng mới

API Endpoint: POST /api/auth/register/

Truy vấn Django ORM:

```
from django.contrib.auth.models import User
from rest_framework_simplejwt.tokens import RefreshToken
username = request.data.get('username')
email = request.data.get('email')
password = request.data.get('password')
if User.objects.filter(username=username).exists():
   raise serializers.ValidationError('Tn ng nhp  tn  ti ')
if User.objects.filter(email=email).exists():
   raise serializers.ValidationError('Email ny  c s  dng ')
user = User.objects.create_user(
   username=username,
   email=email,
   password=password
)
refresh = RefreshToken.for_user(user)
serializer = UserSerializer(user)
```

Kết quả: Tạo người dùng mới và trả về token JWT cùng thông tin người dùng.

VII.9 Đăng nhập người dùng

API Endpoint: POST /api/auth/login/

Truy vấn Django ORM:

```
from django.contrib.auth import authenticate
from rest_framework_simplejwt.tokens import RefreshToken
username = request.data.get('username')
password = request.data.get('password')
user = authenticate(username=username, password=password)
if user is None:
   raise serializers.ValidationError('Thng tin ng nhp khng hp l')
refresh = RefreshToken.for_user(user)
```

Kết quả: Xác thực người dùng và trả về token JWT.

VII.10 Hiển thị lịch sử trò chuyện

API Endpoint: GET /api/messages/{receiver_username}/

Truy vấn Django ORM:

```
from api.models import Message
from django.contrib.auth.models import User
from django.db.models import Q
user = request.user
receiver = User.objects.get(username=receiver_username)
messages = Message.objects.filter(
   Q(sender=user, receiver=receiver) |
   Q(sender=receiver, receiver=user)
).order_by('timestamp')
serializer = MessageSerializer(messages, many=True)
```

Kết quả: Trả về lịch sử trò chuyện giữa người dùng hiện tại và người nhận.

VIII KẾT LUẬN

- **Ưu điểm của đồ án**:
  - Hệ thống quản lý nhạc trực tuyến với API RESTful mạnh mẽ và WebSocket cho nhắn tin thời gian thực.
  - Xác thực an toàn với JWT và hỗ trợ nhắn tin thời gian thực qua WebSocket.
  - Giao diện người dùng hiện đại, nhanh chóng với ReactJS và Vite.
  - Tích hợp MySQL qua XAMPP, đảm bảo hiệu suất tốt hơn SQLite.
  - Hỗ trợ tìm kiếm nhanh và quản lý playlist hiệu quả.
- **Nhược điểm của đồ án**:
  - Chưa tích hợp trình phát nhạc trực tuyến trong trình duyệt.
  - Bảo mật file âm thanh cần được cải thiện (mã hóa, kiểm soát truy cập).
  - Chưa hỗ trợ gợi ý bài hát dựa trên sở thích người dùng.
  - Giao diện frontend cần tối ưu hóa thêm về UX/UI.
- **Hướng phát triển trong tương lai**:
  - Thêm tính năng gợi ý bài hát thông minh sử dụng machine learning.
  - Cải thiện giao diện với các tính năng tương tác nâng cao trong ReactJS.
  - Triển khai hệ thống trên cloud (AWS, Heroku) để tăng khả năng mở rộng.
  - Tăng cường bảo mật với mã hóa file âm thanh và kiểm tra lỗ hổng XSS, CSRF.

IX TÀI LIỆU THAM KHẢO

1. Django Software Foundation. (n.d.). *Django documentation*. Retrieved May 5, 2025, from https://docs.djangoproject.com/
2. Encode. (n.d.). *Django REST framework documentation*. Retrieved May 5, 2025, from https://www.django-rest-framework.org/
3. Django Software Foundation. (n.d.). *Django Channels documentation*. Retrieved May 5, 2025, from https://channels.readthedocs.io/
4. Simple JWT. (n.d.). *Simple JWT documentation*. Retrieved May 5, 2025, from https://django-rest-framework-simple-jwt.readthedocs.io/
5. Meta. (n.d.). *React – A JavaScript library for building user interfaces*. Retrieved May 5, 2025, from https://reactjs.org/
6. You, E. (n.d.). *Vite documentation*. Retrieved May 5, 2025, from https://vitejs.dev/
7. Twitter, Inc. (n.d.). *Bootstrap documentation*. Retrieved May 5, 2025, from https://getbootstrap.com/
8. Oracle Corporation. (n.d.). *MySQL documentation*. Retrieved May 5, 2025, from https://dev.mysql.com/doc/
9. Redis Ltd. (n.d.). *Redis documentation*. Retrieved May 5, 2025, from https://redis.io/
10. Vincent, W. S. (2021). *Django for professionals* (3rd ed.). WelcomeToCode.