

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

ĐỀ TÀI: PHÁT TRIỂN PHẦN MỀM SPOTIFY CLONE

Giảng viên hướng dẫn : **THS TỪ LÃNG PHIÊU**

Sinh viên thực hiện : **TRẦN NHẬT TIẾN 3122560077**
PHAN VIỆT TOÀN 3122560079
CAO THÁI BẢO 3122410014
CAI QUANG TÙNG 3122410455
LÊ MINH THẮNG 3121410461

Thành phố Hồ Chí Minh, tháng 3 năm 2024

LỜI CẢM ƠN

Trước tiên, nhóm chúng em xin gửi lời cảm ơn chân thành đến thầy **Từ Lăng Phiêu**, giảng viên môn Phát triển phần mềm Mã nguồn mở, người đã tận tình hướng dẫn, định hướng và đưa ra những gợi ý quý báu để nhóm hoàn thành đồ án giữa kỳ này. Sự hỗ trợ và những kiến thức mà thầy truyền đạt không chỉ giúp chúng em hiểu rõ hơn về các thuật toán tối ưu hóa mà còn tạo động lực để nhóm nỗ lực thực hiện báo cáo một cách tốt nhất.

Chúng em cũng xin cảm ơn các bạn cùng lớp đã đóng góp ý kiến, chia sẻ tài liệu và tham gia thảo luận, giúp nhóm có thêm góc nhìn đa dạng để hoàn thiện nội dung. Sự động viên và hợp tác từ các bạn là nguồn cảm hứng lớn trong suốt quá trình thực hiện đồ án.

Cuối cùng, nhóm xin gửi lời tri ân đến gia đình và bạn bè, những người luôn ủng hộ và tạo điều kiện để chúng em có thời gian, tâm sức tập trung vào dự án này. Mặc dù báo cáo không tránh khỏi những thiếu sót, chúng em hy vọng rằng sản phẩm này sẽ nhận được sự góp ý từ thầy và các bạn để ngày càng hoàn thiện hơn.

NHẬN XÉT

(Của giảng viên hướng dẫn)

Mục Lục

| | |
|--|-----------|
| Chương 1: Giới thiệu về dự án..... | 5 |
| Chương 2: Phân Tích Dự Án | 6 |
| 2.1. User Stories của ứng dụng Spotify | 6 |
| 2.2. Usecase tổng quát..... | 8 |
| 2.3.1: Chức Năng Phát Nhạc | 9 |
| 2.3.2: Chức Năng Tạo Playlist cá nhân | 10 |
| 2.3.4: Quản Lý Bài Hát | 12 |
| 2.3.5: Quản Lý Album..... | 13 |
| 2.3.6: Xem Thống Kê Hệ Thống..... | 14 |
| 2.3.7: Chức Năng Chat Tích Hợp | 16 |
| 2.4. Flowchart tổng quan..... | 17 |
| 2.5. Cơ sở dữ liệu..... | 19 |
| CHƯƠNG 3: TRIỂN KHAI DỰ ÁN..... | 26 |
| 3.1. Tổng Quan Công Nghệ..... | 26 |
| 3.1.1. Frontend - ReactJS..... | 26 |
| 3.1.2. Backend - Django | 27 |
| 3.1.3. Cơ Sở Dữ Liệu - PostgreSQL | 27 |
| 3.1.4. Tích Hợp AI - Gemini/A2A | 28 |
| 3.1.5. Môi Trường Phát Triển | 28 |
| 3.2. Kỹ Thuật Polling Trong Ứng Dụng Web | 29 |
| 3.2.1. Khái Niệm | 29 |
| 3.2.2. Ứng Dụng Trong Spotify | 29 |
| 3.2.3. Ưu Điểm..... | 30 |
| 3.2.4. Nhược Điểm | 30 |
| 3.2.5. Giải Pháp Thay Thế | 30 |
| 3.3. A2A Protocol và Gemini Trong Ứng Dụng AI | 30 |
| 3.3.1. Giới Thiệu | 30 |
| 3.3.2. Ứng Dụng Trong Spotify | 30 |
| 3.3.3. Ưu Điểm..... | 30 |
| 3.3.4. Nhược Điểm | 30 |

| | |
|--|-----------|
| 3.4. Cấu Trúc Mã Nguồn..... | 31 |
| 3.4.1. Tổng Quan..... | 31 |
| 3.4.2. Mối Liên Kết Giữa Các Thành Phần Trong Dự Án Spotify | 31 |
| Chương 4: Sản phẩm Spotify..... | 35 |
| 4.1: Màn hình user..... | 35 |
| * Màn hình chính..... | 35 |
| * Màn hình đăng nhập, đăng kí. | 35 |
| * Màn hình hồ sơ nghệ sĩ. | 37 |
| * Màn hình Playlist..... | 38 |
| * Màn hình tìm kiếm và xem MV | 39 |
| 4.2: Giao diện gửi tin nhắn..... | 40 |
| 4.3: Màn hình admin. | 45 |
| Chương 5. Kết luận | 50 |
| 5.1. Nhận Xét Chung | 50 |
| 5.2. Hướng Phát Triển Trong Tương Lai..... | 50 |

Chương 1: Giới thiệu về dự án.

Trong bối cảnh công nghệ số phát triển, các nền tảng nghe nhạc trực tuyến như Spotify đã trở thành công cụ giải trí thiết yếu, đáp ứng nhu cầu thưởng thức âm nhạc và kết nối cộng đồng của người dùng. Xuất phát từ mục tiêu học tập và nghiên cứu, nhóm chúng em đã thực hiện đề án **Spotify** – một ứng dụng mô phỏng nền tảng Spotify, tích hợp các công nghệ hiện đại như Django, ReactJS, PostgreSQL, giao thức A2A (Application-to-Application), và trí tuệ nhân tạo Gemini. Dự án nhằm xây dựng một hệ thống web hiệu quả, cung cấp trải nghiệm nghe nhạc cá nhân hóa và các tính năng tương tác thời gian thực.

Spotify được thiết kế dựa trên nguyên tắc lấy người dùng làm trung tâm, tập trung vào các chức năng cốt lõi sau:

1. **Phát nhạc trực tuyến:** Sử dụng React Player để cung cấp trải nghiệm phát nhạc mượt mà, hỗ trợ điều khiển phát/tạm dừng, tua bài, và chuyển bài tự động.
2. **Phát video âm nhạc:** Tích hợp phát video ca nhạc, mở rộng trải nghiệm đa phương tiện.
3. **Quản lý playlist và bài hát yêu thích:** Cho phép người dùng tạo, chỉnh sửa, và lưu trữ playlist cá nhân trong thư viện riêng.
4. **Trang quản trị viên:** Cung cấp giao diện quản lý người dùng, bài hát, album, và thống kê hệ thống, đảm bảo kiểm soát nội dung hiệu quả.
5. **Nhắn tin thời gian thực:** Tích hợp khung chat với hỗ trợ AI từ Gemini, cho phép người dùng chia sẻ bài hát và kết nối cộng đồng.

Dự án không chỉ nhằm mục đích mô phỏng các tính năng của Spotify mà còn mang lại giá trị học thuật thông qua việc ứng dụng các công nghệ tiên tiến, từ phát triển API REST với Django, quản lý cơ sở dữ liệu PostgreSQL, đến tích hợp AI trong nhắn tin. Đồng thời, đề án giúp nhóm rèn luyện các kỹ năng thiết yếu như thiết kế giao diện người dùng (UI/UX), tối ưu hiệu năng, và làm việc nhóm hiệu quả.

Chương 2: Phân Tích Dự Án

2.1. User Stories của ứng dụng Spotify

Dưới đây là các user stories chính, mô tả nhu cầu của người dùng và quản trị viên đối với hệ thống **Spotify**:

2.1.1. Phát Nhạc Trực Tuyến

- Với vai trò là người dùng, tôi muốn phát bài hát chỉ với một cú nhấp chuột để thưởng thức âm nhạc nhanh chóng.
 - Yêu cầu:
 - Giao diện có các nút Phát/Dừng, Tăng/Giảm âm lượng.
 - Hỗ trợ phát nhạc liên tục qua React Player, không bị gián đoạn.
 - Tự động chuyển bài trong playlist khi bài hiện tại kết thúc.
 - Xử lý lỗi khi bài hát không tải được (hiển thị thông báo).

2.1.2. Phát Video Âm Nhạc

- Với vai trò là người dùng, tôi muốn xem video âm nhạc của bài hát để có trải nghiệm đa phương tiện phong phú.
 - Yêu cầu:
 - Tích hợp trình phát video trong giao diện, hỗ trợ phát/tạm dừng.
 - Đảm bảo chất lượng video ổn định, phù hợp với tốc độ mạng.

2.1.3. Quản Lý Playlist và Bài Hát Yêu Thích

- Với vai trò là người dùng đã đăng ký, tôi muốn tạo và quản lý playlist để cá nhân hóa trải nghiệm âm nhạc.
 - Yêu cầu:
 - Cho phép tạo, sửa, xóa playlist thông qua giao diện trực quan.
 - Hỗ trợ thêm/xóa bài hát vào playlist.
 - Hiển thị playlist và bài hát yêu thích trong mục “Thư viện cá nhân”.
 - Lưu trữ dữ liệu playlist trong cơ sở dữ liệu PostgreSQL.

2.1.4. Quản Trị Hệ Thống

- Với vai trò là quản trị viên, tôi muốn quản lý nội dung và người dùng để đảm bảo hệ thống hoạt động ổn định.

- Yêu cầu:

- Giao diện admin riêng biệt.
- Hỗ trợ thêm, sửa, xóa người dùng, bài hát, và album.
- Cung cấp thống kê về lượt phát, số lượng người dùng hoạt động, và tổng số bài hát.

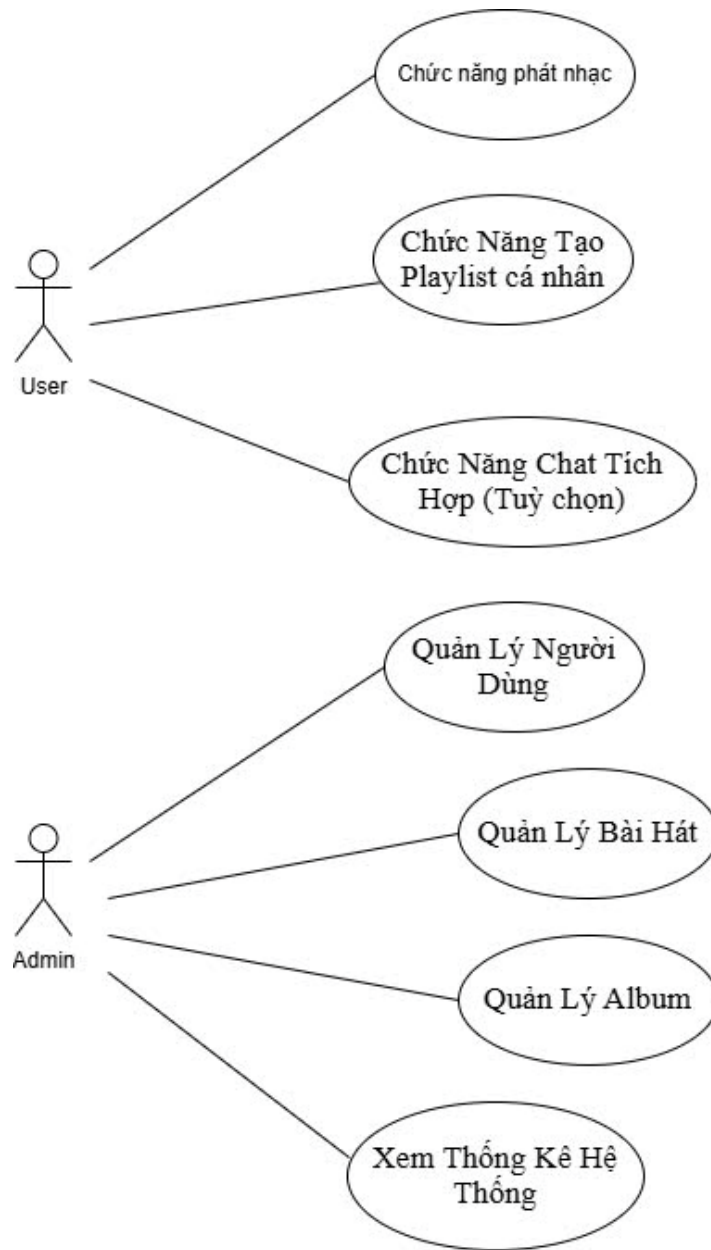
2.1.5. Nhắn Tin Thời Gian Thực

- tôi muốn trò chuyện với người dùng khác để chia sẻ cảm xúc và kết nối cộng đồng.

- Yêu cầu:

- Tích hợp khung chat thời gian thực sử dụng kỹ thuật polling.
- Cho phép gửi tin nhắn văn bản, emoji.
- Hỗ trợ thông báo tin nhắn mới và xử lý lỗi khi kết nối gián đoạn.
- Tích hợp AI Gemini để gợi ý trả lời hoặc gợi ý bài hát.

2.2. Usecase tổng quát



2.3. Usecase của các chức năng.

2.3.1: Chức Năng Phát Nhạc

Use Case: Người dùng có thể chọn bài hát và phát nhạc thông qua giao diện của ứng dụng.

| | |
|------------------------------|--|
| Tên Use Case | Phát nhạc |
| Mô tả | Cho phép người dùng chọn và phát bài hát trực tuyến thông qua giao diện ứng dụng Spotify. |
| Tác nhân | Người dùng đã đăng nhập. |
| Tiền điều kiện | Người dùng đã đăng nhập thành công Dữ liệu bài hát đã được lưu trữ trong cơ sở dữ liệu PostgreSQL Kết nối mạng ổn định |
| Luồng chính | Người dùng truy cập danh sách bài hát hoặc playlist trong giao diện ReactJS. Người dùng nhấp vào bài hát để phát. Frontend gửi yêu cầu GET tới endpoint API <code>/api/tracks/<id></code> (Django REST Framework). Backend truy xuất dữ liệu bài hát từ bảng tracks trong PostgreSQL. Dữ liệu bài hát (URL file MP3) được trả về và phát thông qua React Player. Giao diện hiển thị thanh điều khiển (phát/tạm dừng, âm lượng). |
| Luồng phụ (xử lý lỗi) | Nếu bài hát không tồn tại: Hiển thị thông báo “Bài hát không khả dụng”. Nếu kết nối mạng bị gián đoạn: Tạm dừng phát và hiển thị thông báo “Vui lòng kiểm tra kết nối”. |
| Kết quả | Bài hát được phát thành công. |

2.3.2: Chức Năng Tạo Playlist cá nhân

Use Case: Người dùng có thể tạo playlist cá nhân và thêm bài hát vào danh sách yêu thích.

| | |
|------------------------------|---|
| Tên Use Case | Tạo playlist cá nhân |
| Mô tả | Cho phép người dùng tạo album cá nhân (playlist) và đánh dấu bài hát yêu thích để cá nhân hóa trải nghiệm âm nhạc. |
| Tác nhân | Người dùng đã đăng nhập. |
| Tiền điều kiện | Người dùng đã đăng nhập Dữ liệu bài hát đã có trong cơ sở dữ liệu PostgreSQL. |
| Luồng chính | <p>Người dùng truy cập mục “Thư viện cá nhân” trên giao diện ReactJS.</p> <p>Người dùng nhấp vào nút “Tạo playlist mới” và nhập tên playlist.</p> <p>Frontend gửi yêu cầu POST tới endpoint <code>/api/playlists/</code> với dữ liệu playlist (Django REST Framework).</p> <p>Backend lưu playlist vào bảng <code>playlists</code> trong PostgreSQL.</p> <p>Người dùng chọn bài hát từ danh sách và nhấp “Thêm vào playlist” hoặc “Yêu thích”.</p> <p>Frontend gửi yêu cầu POST tới <code>/api/playlist_tracks/</code> lưu bài hát.</p> <p>Backend cập nhật bảng <code>playlist_tracks</code> hoặc <code>likes</code> trong PostgreSQL.</p> <p>Giao diện hiển thị thông báo “Đã thêm thành công”.</p> |
| Luồng phụ (xử lý lỗi) | <p>Nếu tên playlist trùng: Hiển thị thông báo “Tên playlist đã tồn tại”.</p> <p>Nếu bài hát không tồn tại: Hiển thị thông báo “Bài hát không khả dụng”.</p> |
| Kết quả | Playlist được tạo và được cập nhật. |

2.3.3: Quản Lý Người Dùng

Use Case này cho phép Admin thực hiện các thao tác quản lý người dùng như: xem danh sách, thêm mới, chỉnh sửa hoặc xóa người dùng.

| Trường | Nội dung |
|-----------------------|--|
| Chức năng | Quản Lý Người Dùng |
| Mô tả | Cho phép Admin xem danh sách, thêm mới, chỉnh sửa và xóa thông tin người dùng nhằm duy trì dữ liệu chính xác và cập nhật. |
| Tác nhân | Quản trị viên (Admin) |
| Tiền điều kiện | <ul style="list-style-type: none">- Quản trị viên đã đăng nhập với vai trò admin- Hệ thống đã có dữ liệu người dùng trong bảng users của PostgreSQL |
| Luồng chính | <p>Quản trị viên truy cập trang “Quản Lý Người Dùng” trên giao diện admin (ReactJS).</p> <p>Frontend gửi yêu cầu GET tới endpoint /api/users/ để lấy danh sách người dùng.</p> <p>Backend truy xuất dữ liệu từ bảng users và trả về thông tin (tên, email, vai trò, trạng thái).</p> <p>Quản trị viên thực hiện một trong các thao tác:</p> <ul style="list-style-type: none">○ Xem chi tiết: Nhấp vào người dùng để xem thông tin (ngày đăng ký, hoạt động gần đây) qua API /api/users/<id>.○ Thêm mới: Điền form (tên, email, mật khẩu) và gửi POST tới /api/users/. Backend lưu dữ liệu vào users.○ Chỉnh sửa: Cập nhật thông tin qua form và gửi PUT tới /api/users/<id>. Backend cập nhật users.○ Xóa: Chọn “Xóa”, xác nhận thao tác, gửi DELETE tới /api/users/<id>. Backend xóa bản ghi. <p>Giao diện hiển thị thông báo xác nhận thao tác thành công.</p> |
| Luồng phụ (xử lý lỗi) | Nếu người dùng không tồn tại: Hiển thị thông báo “Người dùng không tìm thấy”. |

| | |
|----------------|--|
| | <p>Nếu email trùng khi thêm mới: Hiển thị thông báo “Email đã được sử dụng”.</p> <p>Nếu thiếu quyền: Hiển thị thông báo “Bạn không có quyền thực hiện thao tác này”.</p> |
| Kết quả | Dữ liệu người dùng được cập nhật một cách chính xác, đảm bảo hệ thống có thông tin đầy đủ và đúng đắn. |

2.3.4: Quản Lý Bài Hát

Use Case này giúp Admin thực hiện việc quản lý bài hát trong hệ thống bao gồm thêm, sửa, xoá và xem chi tiết.

| Trường | Nội dung |
|-----------------------|--|
| Chức năng | Quản Lý Bài Hát |
| Mô tả | Cho phép quản trị viên xem, thêm, chỉnh sửa, và xóa bài hát, bao gồm file âm thanh và metadata (tên, nghệ sĩ, album, thời lượng). |
| Tác nhân | Quản trị viên (Admin) |
| Tiền điều kiện | <p>Quản trị viên đã đăng nhập với vai trò admin.</p> <p>Dữ liệu bài hát và file âm thanh đã được lưu trong bảng tracks và hệ thống lưu trữ file</p> |
| Luồng chính | <p>Quản trị viên truy cập trang “Quản Lý Bài Hát” trên giao diện admin (ReactJS).</p> <p>Frontend gửi yêu cầu GET tới endpoint <code>/api/tracks/</code> để lấy danh sách bài hát (tên, nghệ sĩ, album, thời lượng, lượt phát).</p> <p>Backend truy xuất dữ liệu từ bảng tracks trong PostgreSQL và trả về kết quả.</p> <p>Quản trị viên thực hiện một trong các thao tác:</p> <ul style="list-style-type: none"> Xem chi tiết: Nhấp vào bài hát để xem metadata qua API <code>/api/tracks/<id></code>. |

| | |
|-------------------------------|--|
| | <ul style="list-style-type: none"> ○ Thêm mới: Điền form (tên, nghệ sĩ, album, file MP3) và gửi POST tới /api/tracks/. Backend lưu file vào hệ thống lưu trữ và cập nhật tracks. ○ Chính sửa: Cập nhật thông tin qua form và gửi PUT tới /api/tracks/<id>. Backend cập nhật tracks. ○ Xóa: Chọn “Xóa”, xác nhận thao tác, gửi DELETE tới /api/tracks/<id>. Backend xóa bản ghi và file liên quan. <p>Giao diện hiển thị thông báo xác nhận thao tác thành công.</p> |
| Luồng phụ (xử lý lỗi): | <p>Nếu bài hát không tồn tại: Hiển thị thông báo “Bài hát không tìm thấy”.</p> <p>Nếu file âm thanh không hợp lệ: Hiển thị thông báo “File không đúng định dạng”.</p> |
| Kết quả | Danh sách bài hát được cập nhật chính xác, đảm bảo tính nhất quán của dữ liệu âm thanh và metadata liên quan. |

2.3.5: Quản Lý Album

Use Case này cho phép Admin quản lý album, bao gồm các thao tác thêm mới, chỉnh sửa, xóa hoặc liên kết các bài hát.

| Trường | Nội dung |
|-----------------------|--|
| Chức năng | Quản Lý Album |
| Mô tả | Cho phép Admin xem danh sách album, thêm mới, chỉnh sửa và xóa album, cùng với việc liên kết danh sách bài hát tương ứng. |
| Tác nhân | Quản trị viên (Admin) |
| Tiền điều kiện | <p>Quản trị viên đã đăng nhập với vai trò admin</p> <p>Dữ liệu album và bài hát đã được lưu trong bảng albums và tracks của PostgreSQL</p> |
| Luồng chính | Quản trị viên truy cập trang “Quản Lý Album” trên giao diện admin (ReactJS). |

| | |
|------------------------------|--|
| | <p>Frontend gửi yêu cầu GET tới endpoint <code>/api/albums/</code> để lấy danh sách album (tên, nghệ sĩ, ngày phát hành, danh sách bài hát).</p> <p>Backend truy xuất dữ liệu từ bảng <code>albums</code> và <code>tracks</code> trong PostgreSQL, trả về kết quả.</p> <p>Quản trị viên thực hiện một trong các thao tác:</p> <ul style="list-style-type: none"> Xem chi tiết: Nhấp vào album để xem thông tin và danh sách bài hát qua API <code>/api/albums/<id></code>. Thêm mới: Điền form (tên, nghệ sĩ, năm phát hành, hình ảnh bìa, danh sách bài hát) và gửi POST tới <code>/api/albums/</code>. Backend lưu dữ liệu vào <code>albums</code> và liên kết với <code>tracks</code>. Chỉnh sửa: Cập nhật thông tin qua form và gửi PUT tới <code>/api/albums/<id></code>. Backend cập nhật <code>albums</code>. Xóa: Chọn “Xóa”, xác nhận thao tác, gửi DELETE tới <code>/api/albums/<id></code>. Backend xóa bản ghi album (không xóa bài hát liên kết). <p>Giao diện hiển thị thông báo xác nhận thao tác thành công.</p> |
| Luồng phụ (xử lý lỗi) | <p>Nếu album không tồn tại: Hiển thị thông báo “Album không tìm thấy”.</p> <p>Nếu tên album trùng: Hiển thị thông báo “Tên album đã tồn tại”.</p> <p>Nếu bài hát liên kết không hợp lệ: Hiển thị thông báo “Bài hát không tồn tại”.</p> |
| Kết quả | Danh sách album được cập nhật, liên kết hợp lệ với bài hát tương ứng, đảm bảo quản lý nội dung âm nhạc an toàn. |

2.3.6: Xem Thống Kê Hệ Thống

Use Case giúp Admin truy cập các số liệu thống kê về hệ thống để theo dõi tình trạng hoạt động và đưa ra quyết định phù hợp.

| Trường | Nội dung |
|------------------|-----------------------|
| Chức năng | Xem Thống Kê Hệ Thống |

| | |
|------------------------------|---|
| Mô tả | Cung cấp cho Admin các số liệu thống kê chi tiết về lượt phát, số lượng người dùng hoạt động và tổng số bài hát hiện có. |
| Tác nhân | Quản trị viên (Admin) |
| Tiền điều kiện | <p>Quản trị viên đã đăng nhập với vai trò admin</p> <p>Dữ liệu thống kê (lượt phát, người dùng, bài hát) đã được thu thập và lưu trong PostgreSQL.</p> <p>Quyền admin được cấp qua Django REST Framework.</p> |
| Luồng chính | <p>Quản trị viên truy cập trang “Thống Kê Hệ Thống” trên giao diện admin (ReactJS).</p> <p>Frontend gửi yêu cầu GET tới endpoint <code>/api/statistics/</code> để lấy dữ liệu thống kê.</p> <p>Backend truy xuất dữ liệu từ các bảng tracks (lượt phát), users (người dùng hoạt động), và tổng số bản ghi trong tracks.</p> <p>Giao diện hiển thị các biểu đồ và chỉ số:</p> <ul style="list-style-type: none"> ○ Lượt phát theo thời gian (ngày, tuần, tháng). ○ Số lượng người dùng hoạt động (đăng nhập trong khoảng thời gian). ○ Tổng số bài hát hiện có trong hệ thống. <p>Quản trị viên chọn khoảng thời gian hoặc kiểu biểu đồ (cột, đường) để lọc dữ liệu qua API <code>/api/statistics/?range=<time></code>.</p> <p>Giao diện cập nhật biểu đồ theo yêu cầu.</p> |
| Luồng phụ (xử lý lỗi) | <p>Nếu dữ liệu trống: Hiển thị thông báo “Chưa có dữ liệu thống kê”.</p> <p>Nếu truy xuất thất bại: Hiển thị thông báo “Lỗi tải dữ liệu, vui lòng thử lại”.</p> |
| Kết quả | Quản trị viên nhận được số liệu thống kê chính xác, hỗ trợ phân tích và ra quyết định quản trị. |

2.3.7: Chức Năng Chat Tích Hợp

Use Case: Người dùng có thể nhắn tin với nhau thông qua tính năng chat tích hợp trên giao diện web.

| | |
|------------------------------|---|
| Tên Use Case | Chat Thời Gian Thực |
| Mô tả | Cho phép người dùng giao tiếp thời gian thực thông qua khung chat tích hợp, hỗ trợ gửi tin nhắn văn bản, emoji, với tích hợp AI Gemini để gợi ý bài hát. |
| Tác nhân | Người dùng đã đăng nhập |
| Tiền điều kiện | Người dùng đã đăng nhập Dữ liệu người dùng và tin nhắn được lưu trong bảng users và messages của PostgreSQL. |
| Luồng chính | <p>Người dùng truy cập mục “Chat” trên giao diện ReactJS.</p> <p>Người dùng tìm kiếm người nhận bằng username qua ô tìm kiếm.</p> <p>Frontend gửi yêu cầu GET tới endpoint <code>/api/users/?username=<query></code> để lấy danh sách người dùng phù hợp.</p> <p>Người dùng chọn người nhận và nhập tin nhắn (văn bản, emoji).</p> <p>Frontend gửi yêu cầu POST tới <code>/api/messages/</code> với nội dung tin nhắn (Django REST Framework).</p> <p>Backend lưu tin nhắn vào bảng messages trong PostgreSQL.</p> <p>Frontend sử dụng polling (gửi GET định kỳ tới <code>/api/messages/?user=<id></code>) để nhận tin nhắn mới.</p> <p>Nếu kích hoạt AI, frontend gửi nội dung tin nhắn tới API Gemini qua giao thức A2A để nhận gợi ý trả lời.</p> <ol style="list-style-type: none">1. Giao diện hiển thị tin nhắn và gợi ý (nếu có) trong khung chat. |
| Luồng phụ (xử lý lỗi) | <p>Nếu người nhận không tồn tại: Hiển thị thông báo “Người dùng không tìm thấy”.</p> <p>Nếu kết nối mạng gián đoạn: Hiển thị thông báo “Không thể gửi tin nhắn, vui lòng kiểm tra kết nối”.</p> |

| | |
|----------------|--|
| | Nếu API Gemini lỗi: Hiển thị thông báo “Không thể tạo gợi ý, vui lòng thử lại”. |
| Kết quả | Tin nhắn được gửi và hiển thị thành công, người dùng có thể giao tiếp thời gian thực với trải nghiệm được nâng cao nhờ AI. |

2.3. Các Tính Năng Được Xây Dựng

- Đăng ký và Đăng nhập: Xác thực người dùng và lưu thông tin trong bảng users của PostgreSQL.
- Tạo và Quản Lý Playlist Cá nhân: Cho phép người dùng tạo, chỉnh sửa, và xóa playlist, lưu dữ liệu trong bảng playlists và playlist_tracks.
- Phát Nhạc Trực Tuyến: Sử dụng React Player để phát nhạc từ file MP3, truy xuất qua API /api/tracks/ từ bảng tracks.
- Tìm Kiếm Nội Dung: Hỗ trợ tìm kiếm bài hát, album, và nghệ sĩ qua API /api/search/?q=<query>, trả về kết quả từ bảng tracks, albums, artists.
- Quản Lý Bài Hát Yêu Thích: Cho phép đánh dấu và quản lý bài hát yêu thích, lưu vào bảng likes.
- Trang Quản Trị Viên: Giao diện admin để quản lý người dùng (users), nghệ sĩ (artists), album (albums), và playlist (playlists)
- Chat Thời Gian Thực (Tùy chọn): Tích hợp khung chat dựa trên polling, lưu tin nhắn trong bảng messages, và sử dụng Gemini qua giao thức A2A để gợi ý trả lời thông minh.

Mỗi tính năng được thiết kế để tối ưu hóa trải nghiệm người dùng, đảm bảo hiệu năng hệ thống, và thể hiện khả năng tích hợp các công nghệ tiên tiến như Django, ReactJS, PostgreSQL, và AI.

2.4. Flowchart tổng quan

Luồng Dữ Liệu Chính

1. Truy cập Website:

- Người dùng mở trình duyệt và truy cập giao diện Spotify (ReactJS).

- Frontend tải các thành phần giao diện (components) và gửi yêu cầu khởi tạo tới backend.

2. Đăng nhập:

- Người dùng nhập thông tin đăng nhập (email, mật khẩu).
- Frontend gửi yêu cầu POST tới endpoint `/api/auth/login/`.
- Backend xác thực thông tin từ bảng users trong PostgreSQL và trả về token.
- Frontend lưu token trong local storage để sử dụng cho các yêu cầu sau.

3. Tìm Kiếm Bài Hát, Album, Nghệ Sĩ:

- Người dùng nhập từ khóa tìm kiếm trong thanh tìm kiếm.
- Frontend gửi yêu cầu GET tới `/api/search/?q=<query>`.
- Backend truy xuất dữ liệu từ bảng tracks, albums, và artists, trả về danh sách kết quả.
- Frontend hiển thị kết quả trên giao diện.

4. Phát Nhạc:

- Người dùng chọn bài hát từ danh sách hoặc playlist.
- Frontend gửi yêu cầu GET tới `/api/tracks/<id>` để lấy URL file MP3.
- Backend truy xuất dữ liệu từ bảng tracks và trả về URL.
- Frontend sử dụng React Player để phát nhạc, hiển thị thanh điều khiển (phát/tạm dừng, tua bài, âm lượng).

5. Tạo và Quản Lý Playlist:

- Người dùng truy cập mục “Thư viện cá nhân” và chọn “Tạo playlist mới”.
- Frontend gửi yêu cầu POST tới `/api/playlists/` với thông tin playlist (tên, mô tả).
- Backend lưu dữ liệu vào bảng playlists trong PostgreSQL.
- Người dùng thêm bài hát vào playlist, frontend gửi POST tới `/api/playlist_tracks/`.
- Backend lưu liên kết vào bảng playlist_tracks.

6. Quản Lý Admin:

- Quản trị viên truy cập trang admin, frontend gửi yêu cầu GET tới /api/users/, /api/tracks/, /api/albums/, /api/artists/.
- Backend truy xuất dữ liệu từ các bảng tương ứng (users, tracks, albums, artists) và trả về.
- Quản trị viên thực hiện thêm/sửa/xóa dữ liệu:
 - Thêm: Gửi POST tới endpoint tương ứng (ví dụ: /api/tracks/).
 - Sửa: Gửi PUT tới /api/tracks/<id>.
 - Xóa: Gửi DELETE tới /api/tracks/<id>.
- Backend cập nhật cơ sở dữ liệu và trả về thông báo.

7. Chat Thời Gian Thực:

- Người dùng truy cập khung chat, frontend gửi GET tới /api/users/?username=<query> để tìm người nhận.
- Người dùng gửi tin nhắn, frontend gửi POST tới /api/messages/ với nội dung (văn bản, emoji, liên kết bài hát).
- Backend lưu tin nhắn vào bảng messages.
- Frontend sử dụng kỹ thuật polling (GET định kỳ tới /api/messages/?user=<id>) để cập nhật tin nhắn mới.
- Frontend gửi nội dung tin nhắn tới API Gemini qua giao thức A2A để nhận gợi ý trả lời.
- Giao diện hiển thị tin nhắn và gợi ý AI

2.5. Cơ sở dữ liệu

2.5.1. Mô Tả Các Bảng Chính

Dưới đây là danh sách các bảng chính trong cơ sở dữ liệu, cùng với mô tả và vai trò của chúng:

1. users:

- **Mô tả:** Lưu trữ thông tin người dùng (bao gồm cả người dùng thường và quản trị viên).
- **Cột chính:**

- id (SERIAL, PRIMARY KEY): Mã định danh duy nhất.
- email (VARCHAR): Địa chỉ email để đăng nhập.
- password (VARCHAR): Mật khẩu đã mã hóa.
- username (VARCHAR): Tên người dùng hiển thị.
- role (VARCHAR): Vai trò (user/admin).
- created_at (TIMESTAMP): Thời gian đăng ký.
- **Vai trò:** quản lý người dùng, và chat (tìm kiếm username).

2. profiles:

- **Mô tả:** Lưu trữ thông tin hồ sơ cá nhân của người dùng.
- **Cột chính:**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - user_id (INTEGER, FOREIGN KEY → users.id): Liên kết với người dùng.
 - full_name (VARCHAR): Họ tên.
 - bio (TEXT): Tiểu sử.
 - avatar_url (VARCHAR): URL ảnh đại diện.
- **Vai trò:** Cá nhân hóa trải nghiệm người dùng, hiển thị thông tin trong giao diện.

3. artists:

- **Mô tả:** Lưu trữ thông tin về nghệ sĩ.
- **Cột chính:**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - name (VARCHAR): Tên nghệ sĩ.
 - bio (TEXT): Tiểu sử nghệ sĩ.
 - image_url (VARCHAR): URL ảnh nghệ sĩ.
- **Vai trò:** Hỗ trợ tìm kiếm và quản lý nội dung âm nhạc.

4. albums:

- **Mô tả:** Lưu trữ thông tin về album nhạc.
- **Cột chính:**
 - id (SERIAL, PRIMARY KEY): Mã định danh.

- title (VARCHAR): Tên album.
- artist_id (INTEGER, FOREIGN KEY → artists.id): Liên kết với nghệ sĩ.
- release_date (DATE): Ngày phát hành.
- cover_url (VARCHAR): URL ảnh bìa album.
- **Vai trò:** Quản lý album, liên kết với bài hát, và hiển thị trong tìm kiếm.

5. tracks:

- **Mô tả:** Lưu trữ thông tin về bài hát.
- **Cột chính:**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - title (VARCHAR): Tên bài hát.
 - artist_id (INTEGER, FOREIGN KEY → artists.id): Liên kết với nghệ sĩ.
 - album_id (INTEGER, FOREIGN KEY → albums.id): Liên kết với album.
 - duration (INTEGER): Thời lượng (giây).
 - file_url (VARCHAR): URL file MP3.
 - play_count (INTEGER): Số lượt phát.
- **Vai trò:** Hỗ trợ phát nhạc (React Player), tìm kiếm, và thống kê lượt phát.

6. playlists:

- **Mô tả:** Lưu trữ thông tin về playlist do người dùng tạo.
- **Cột chính:**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - user_id (INTEGER, FOREIGN KEY → users.id): Liên kết với người dùng.
 - title (VARCHAR): Tên playlist.
 - created_at (TIMESTAMP): Thời gian tạo.
- **Vai trò:** Hỗ trợ tạo và quản lý playlist cá nhân.

7. **playlist_tracks:**

- **Mô tả:** Lưu trữ liên kết giữa playlist và bài hát.
- **Cột chính:**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - playlist_id (INTEGER, FOREIGN KEY → playlists.id): Liên kết với playlist.
 - track_id (INTEGER, FOREIGN KEY → tracks.id): Liên kết với bài hát.
 - added_at (TIMESTAMP): Thời gian thêm.
- **Vai trò:** Quản lý danh sách bài hát trong playlist.

8. **messages:**

- **Mô tả:** Lưu trữ tin nhắn trong chức năng chat thời gian thực.
- **Cột chính:**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - sender_id (INTEGER, FOREIGN KEY → users.id): Người gửi.
 - receiver_id (INTEGER, FOREIGN KEY → users.id): Người nhận.
 - content (TEXT): Nội dung tin nhắn.
 - sent_at (TIMESTAMP): Thời gian gửi.
- **Vai trò:** Hỗ trợ chat thời gian thực (polling) và tích hợp Gemini/A2A.

9. **likes:**

- **Mô tả:** Lưu trữ bài hát yêu thích của người dùng.
- **Cột chính:**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - user_id (INTEGER, FOREIGN KEY → users.id): Liên kết với người dùng.
 - track_id (INTEGER, FOREIGN KEY → tracks.id): Liên kết với bài hát.
 - liked_at (TIMESTAMP): Thời gian thích.

- **Vai trò:** Quản lý danh sách bài hát yêu thích.

10. **followers:**

- **Mô tả:** Lưu trữ mối quan hệ theo dõi giữa các người dùng.
- **Cột chính:**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - follower_id (INTEGER, FOREIGN KEY → users.id): Người theo dõi.
 - followed_id (INTEGER, FOREIGN KEY → users.id): Người được theo dõi.
 - followed_at (TIMESTAMP): Thời gian theo dõi.
- **Vai trò:** Hỗ trợ tính năng kết nối cộng đồng.

11. **music_videos:**

- **Mô tả:** Lưu trữ thông tin về video âm nhạc (nếu có tính năng phát video).
- **Cột chính:**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - title (VARCHAR): Tên video.
 - artist_id (INTEGER, FOREIGN KEY → artists.id): Liên kết với nghệ sĩ.
 - video_url (VARCHAR): URL file video.
 - duration (INTEGER): Thời lượng (giây).
- **Vai trò:** Hỗ trợ phát video âm nhạc (tùy chọn).

12. **user_albums** và **user_album_tracks:**

- **Mô tả:** Lưu trữ album và bài hát do người dùng tạo (nếu có tính năng này).
- **Cột chính (user_albums):**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - user_id (INTEGER, FOREIGN KEY → users.id): Liên kết với người dùng.
 - title (VARCHAR): Tên album.

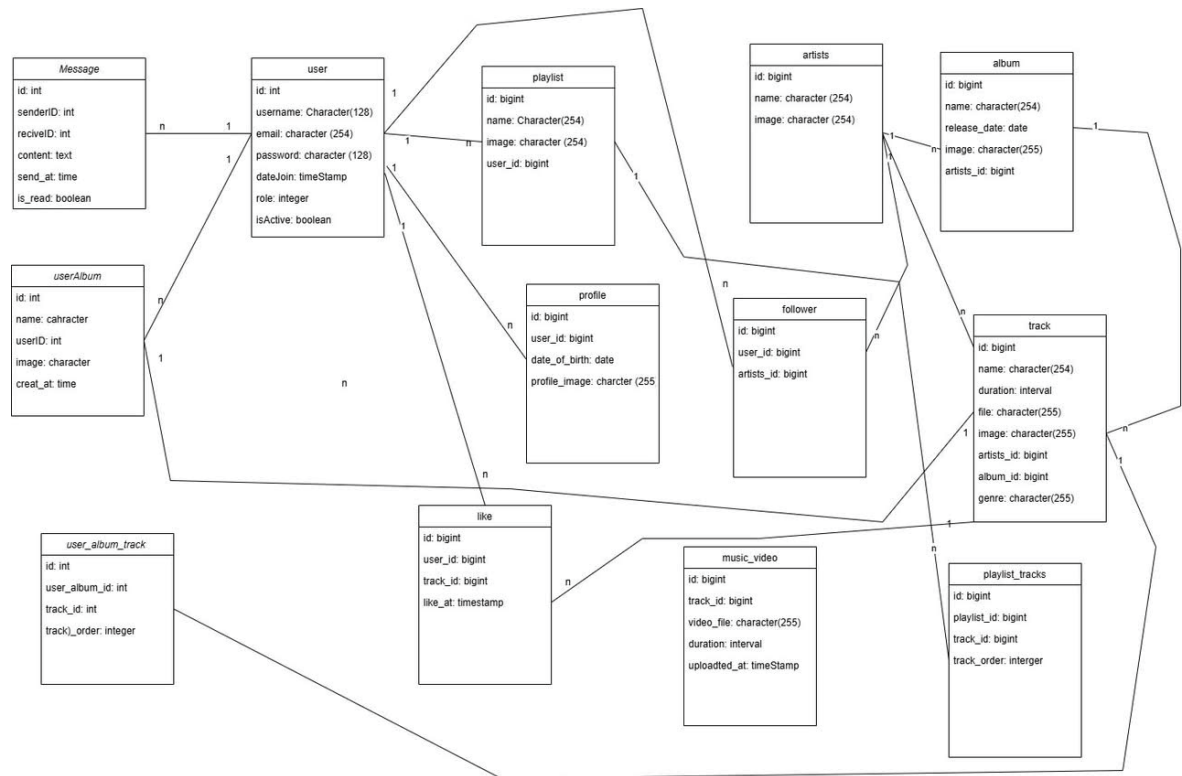
- created_at (TIMESTAMP): Thời gian tạo.
- **Cột chính (user_album_tracks):**
 - id (SERIAL, PRIMARY KEY): Mã định danh.
 - user_album_id (INTEGER, FOREIGN KEY → user_albums.id): Liên kết với album.
 - track_id (INTEGER, FOREIGN KEY → tracks.id): Liên kết với bài hát.
- **Vai trò:** Hỗ trợ tính năng tạo album cá nhân.

2.5.2. Mối Quan Hệ Giữa Các Bảng

Các bảng được liên kết thông qua khóa ngoại để đảm bảo tính toàn vẹn dữ liệu:

- **users ↔ profiles:** Quan hệ 1:1 (user_id trong profiles liên kết với id trong users).
- **users ↔ playlists:** Quan hệ 1:N (một người dùng tạo nhiều playlist).
- **playlists ↔ playlist_tracks:** Quan hệ 1:N (một playlist chứa nhiều bài hát).
- **tracks ↔ playlist_tracks:** Quan hệ N:N thông qua bảng trung gian playlist_tracks.
- **artists ↔ albums** và **artists ↔ tracks:** Quan hệ 1:N (một nghệ sĩ có nhiều album/bài hát).
- **albums ↔ tracks:** Quan hệ 1:N (một album chứa nhiều bài hát).
- **users ↔ messages:** Quan hệ 1:N (một người dùng gửi/nhận nhiều tin nhắn).
- **users ↔ likes:** Quan hệ 1:N (một người dùng thích nhiều bài hát).
- **users ↔ followers:** Quan hệ N:N (một người dùng theo dõi/được theo dõi bởi nhiều người).
- **users ↔ user_albums** và **user_albums ↔ user_album_tracks:** Quan hệ 1:N và N:N.

2.5.3. Sơ Đồ ERD



CHƯƠNG 3: TRIỂN KHAI DỰ ÁN.

3.1. Tổng Quan Công Nghệ

Hệ thống Spotify hoạt động theo mô hình client-server:

- Frontend (ReactJS): Xây dựng giao diện người dùng động, gửi yêu cầu API, và phát nhạc qua React Player.
- Backend (Django): Xử lý logic nghiệp vụ, cung cấp API RESTful, và quản lý dữ liệu từ PostgreSQL.
- Cơ sở dữ liệu (PostgreSQL): Lưu trữ thông tin người dùng, bài hát, playlist, và tin nhắn.
- AI (Gemini/A2A): Hỗ trợ gợi ý trả lời thông minh trong chat.
- Môi trường phát triển: Sử dụng môi trường ảo Python và Node.js để quản lý thư viện, đảm bảo tính đồng nhất.

3.1.1. Frontend - ReactJS

ReactJS là thư viện JavaScript dùng để xây dựng giao diện người dùng tương tác.

Các tính năng chính bao gồm:

- Giao diện: Sử dụng component (trong src/components/) để hiển thị danh sách bài hát, playlist, và khung chat.
- Phát nhạc: Tích hợp React Player để phát file MP3 từ URL lấy qua API /api/tracks/<id>.
- Gọi API: Sử dụng axios để gửi yêu cầu HTTP (GET, POST, PUT, DELETE) tới backend.
- Quản lý trạng thái: Sử dụng React Hooks (useState, useEffect) để cập nhật giao diện và xử lý dữ liệu.
- Thư viện chính:
 - react, react-dom: Xây dựng component giao diện.
 - axios: Gửi yêu cầu HTTP.
 - react-player: Phát nhạc và video (nếu có).
 - react-router-dom: Quản lý định tuyến trang.

Vai trò: ReactJS đảm bảo giao diện trực quan, phản hồi nhanh, và hỗ trợ các tính năng như tìm kiếm, phát nhạc, và chat.

3.1.2. Backend - Django

Django là framework Python dùng để xây dựng backend theo mô hình MTV (Model-Template-View). Các thành phần chính:

- API RESTful: Sử dụng Django REST Framework để cung cấp các endpoint như `/api/tracks/`, `/api/playlists/`, `/api/messages/`, `/api/inbox/<user_id>/`.
- Quản lý dữ liệu: Kết nối với PostgreSQL qua psycopg2 để truy xuất và cập nhật các bảng (users, tracks, messages).
- Logic nghiệp vụ: Xử lý tìm kiếm, lưu playlist, quản lý tin nhắn, và xác thực người dùng.
- Thư viện chính:
 - Django==5.2: Framework chính.
 - djangorestframework==3.16.0: Xây dựng API.
 - django-cors-headers==4.7.0: Xử lý CORS.
 - psycopg2==2.9.10: Kết nối PostgreSQL.
 - asgiref==3.8.1: Hỗ trợ bất đồng bộ.
 - sqlparse==0.5.3: Phân tích câu lệnh SQL.

Vai trò: Django cung cấp backend ổn định, API hiệu quả, và quản lý dữ liệu an toàn.

3.1.3. Cơ Sở Dữ Liệu - PostgreSQL

PostgreSQL là hệ quản trị cơ sở dữ liệu quan hệ, lưu trữ dữ liệu cho Spotify. Các bảng chính:

- users: Thông tin đăng nhập (email, password, username, role).
- tracks: Dữ liệu bài hát (title, artist_id, file_url, play_count).
- playlists, playlist_tracks: Quản lý playlist cá nhân.
- messages: Lưu tin nhắn chat (sender_id, receiver_id, content, sent_at).
- Tối ưu hóa:
 - Sử dụng sequence (SELECT setval('<table>_id_seq', ...);) để quản lý ID.
 - Tạo index trên users.email, tracks.title để tăng tốc tìm kiếm.

- Khóa ngoại đảm bảo toàn vẹn dữ liệu.

Vai trò: PostgreSQL hỗ trợ lưu trữ và truy xuất dữ liệu nhanh chóng cho các tính năng như phát nhạc, tìm kiếm, và chat.

3.1.4. Tích Hợp AI - Gemini/A2A

Chức năng chat được nâng cao nhờ AI:

- Gemini: Mô hình AI của Google, xử lý ngôn ngữ tự nhiên và gợi ý trả lời thông minh.
- A2A Protocol: Giao thức mở, cho phép frontend gửi tin nhắn tới Gemini qua API.
- Triển khai:
 - Frontend gửi POST tới /api/messages/ để lưu tin nhắn.
 - Tin nhắn được chuyển tới Gemini qua A2A (dùng requests==2.31.0).
 - Gemini trả về gợi ý, hiển thị trong khung chat.
- Thư viện hỗ trợ:
 - requests==2.31.0: Gửi yêu cầu HTTP.
 - urllib3, certifi: Đảm bảo kết nối an toàn.

Vai trò: Tăng cường trải nghiệm chat với gợi ý AI, cá nhân hóa giao tiếp.

3.1.5. Môi Trường Phát Triển

- Môi trường ảo Python:
 - Sử dụng virtualenv để tạo môi trường ảo trên Ubuntu, cô lập các thư viện backend như Django, psycpg2, djangoestframework.
 - Lệnh thiết lập:
 - sudo apt update
 - sudo apt install python3 python3-pip python3-venv
 - python3 -m venv venv
 - source venv/bin/activate
 - pip install -r requirements.txt
 - File requirements.txt liệt kê các thư viện (Django==5.2, psycpg2==2.9.10, v.v.).
- Node.js:

- Cài đặt Node.js và npm trên Ubuntu để quản lý thư viện frontend (react, axios, react-player).
- Lệnh cài đặt:
- `sudo apt install nodejs npm`
`npm install`
- File package.json liệt kê các thư viện Node.js.
- Công cụ hỗ trợ:
 - PyYAML==6.0.1: Xử lý file cấu hình YAML (như cấu hình Django).
 - rich, colorama: Hiển thị log màu trong terminal, hỗ trợ debug trên Ubuntu.
 - pipx, setuptools, wheel: Quản lý và cài đặt thư viện Python.
- Quy trình triển khai:
 - Cài đặt PostgreSQL trên Ubuntu:
 - `sudo apt install postgresql postgresql-contrib`
`sudo -u postgres psql -c "CREATE DATABASE Spotify;"`
 - Chạy backend: `python manage.py runserver`.
 - Chạy frontend: `npm start`.

Vai trò: Môi trường ảo trên Ubuntu đảm bảo dự án chạy ổn định, cô lập thư viện, và dễ dàng triển khai trên các máy phát triển hoặc server.

3.2. Kỹ Thuật Polling Trong Ứng Dụng Web

3.2.1. Khái Niệm

Polling là kỹ thuật gửi yêu cầu HTTP định kỳ từ client tới server để kiểm tra dữ liệu mới, thường dùng cho các ứng dụng cần cập nhật gần thời gian thực như chat.

3.2.2. Ứng Dụng Trong Spotify

Trong Spotify, polling được triển khai trong:

- Messages.jsx: Cập nhật danh sách cuộc trò chuyện bằng API GET `/api/inbox/<user_id>/` mỗi 1 giây.
- MessageDetail.jsx: Cập nhật tin nhắn giữa hai người dùng qua API GET `/api/messages/<sender_id>/<receiver_id>/` mỗi 1 giây.
- Triển khai: Sử dụng React Hooks (`useEffect`, `setInterval`) để gửi yêu cầu và cập nhật giao diện.

3.2.3. Ưu Điểm

- Đơn giản, dễ tích hợp với API RESTful.
- Tương thích mọi trình duyệt.
- Tùy chỉnh tần suất (1 giây trong Spotify).

3.2.4. Nhược Điểm

- Tăng tải server do yêu cầu định kỳ.
- Độ trễ nhẹ (cập nhật sau 1 giây).
- Tiêu tốn băng thông khi không có dữ liệu mới.

3.2.5. Giải Pháp Thay Thế

- WebSocket: Kết nối hai chiều, server đẩy dữ liệu mới.
- Server-Sent Events (SSE): Server gửi dữ liệu một chiều, giảm tải.
- Dự án: Polling được chọn vì tính đơn giản, nhưng SSE có thể thay thế trong tương lai.

3.3. A2A Protocol và Gemini Trong Ứng Dụng AI

3.3.1. Giới Thiệu

A2A Protocol là giao thức mở của Google, cho phép các tác nhân AI giao tiếp xuyên nền tảng. Gemini là mô hình AI xử lý ngôn ngữ tự nhiên, hỗ trợ gợi ý thông minh.

3.3.2. Ứng Dụng Trong Spotify

- Chức năng chat:
 - Frontend gửi tin nhắn qua API /api/messages/ và chuyển tới Gemini qua A2A.
 - Gemini phân tích và trả về gợi ý trả lời (ví dụ: gợi ý bài hát theo ngữ cảnh).
- Cải tiến: Thay polling (1 giây) bằng Server-Sent Events (SSE) để bot đẩy tin nhắn mới, giảm tải server.

3.3.3. Ưu Điểm

- A2A: Chuẩn hóa giao tiếp AI, bảo mật, hỗ trợ SSE.
- Gemini: Hiểu ngữ cảnh, tạo phản hồi thông minh.

3.3.4. Nhược Điểm

- A2A: Thiết lập ban đầu phức tạp.

- Gemini: Chi phí API cao, giới hạn tần suất.

3.4. Cấu Trúc Mã Nguồn

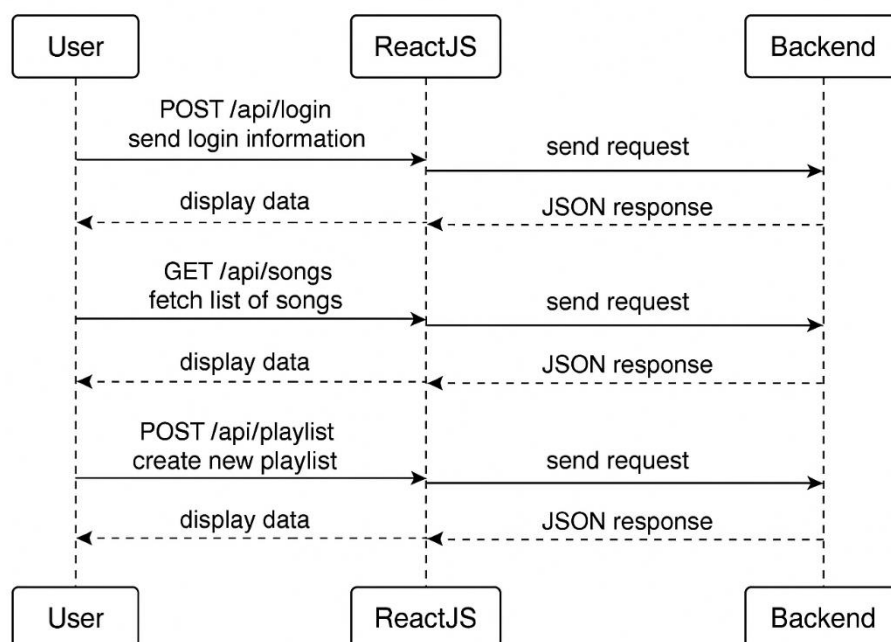
3.4.1. Tổng Quan

Dự án Spotify được tổ chức theo mô hình phân lớp, tách biệt frontend (ReactJS), backend (Django), và cơ sở dữ liệu (PostgreSQL). Cấu trúc mã nguồn rõ ràng, dễ bảo trì, và hỗ trợ làm việc nhóm.

3.4.2. Môi Liên Kết Giữa Các Thành Phần Trong Dự Án Spotify

a: Lớp Giao Diện Người Dùng (Frontend – ReactJS)

- Công nghệ: ReactJS
- Vai trò:
 - + Hiển thị giao diện người dùng.
 - + Gửi request đến backend qua API.
 - + Hiển thị dữ liệu phản hồi từ backend (JSON).
- Ví dụ các request:
 - + POST /api/login → gửi thông tin đăng nhập



- + GET /api/songs → lấy danh sách bài hát
- + POST /api/playlist → tạo playlist mới

b: Lớp Xử Lý Logic (Backend – Django REST Framework)

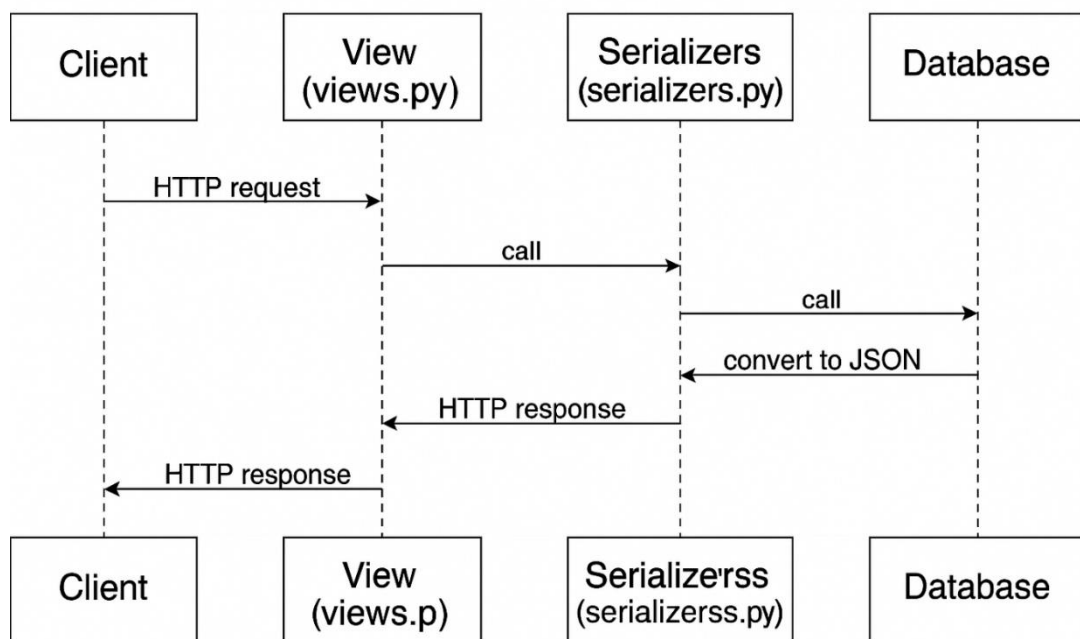
- Công nghệ: Django REST Framework

- Vai trò:

- + Xử lý các HTTP request từ frontend.
- + Truy vấn và cập nhật dữ liệu từ cơ sở dữ liệu.
- + Trả dữ liệu JSON về cho frontend.

- Các lớp quan trọng:

- + views.py – xử lý logic và HTTP response.
- + serializers.py – chuyển đổi model \leftrightarrow JSON.
- + models.py – định nghĩa cấu trúc dữ liệu.
- + urls.py – định tuyến API endpoint.



c: Lớp Dữ Liệu (Database – PostgreSQL)

- Công nghệ: PostgreSQL

- Vai trò:

- + Lưu trữ toàn bộ thông tin người dùng, bài hát, album, playlist,...
- + Django ORM truy cập cơ sở dữ liệu một cách trừu tượng hóa.

- Ví dụ bảng dữ liệu:

+ User – thông tin người dùng

+ Song – bài hát

+ Playlist – danh sách phát

+ Artist, Album – các bảng liên kết

d: Luồng Dữ Liệu Tổng Thể

Ví dụ khi người dùng muốn nghe một bài hát:

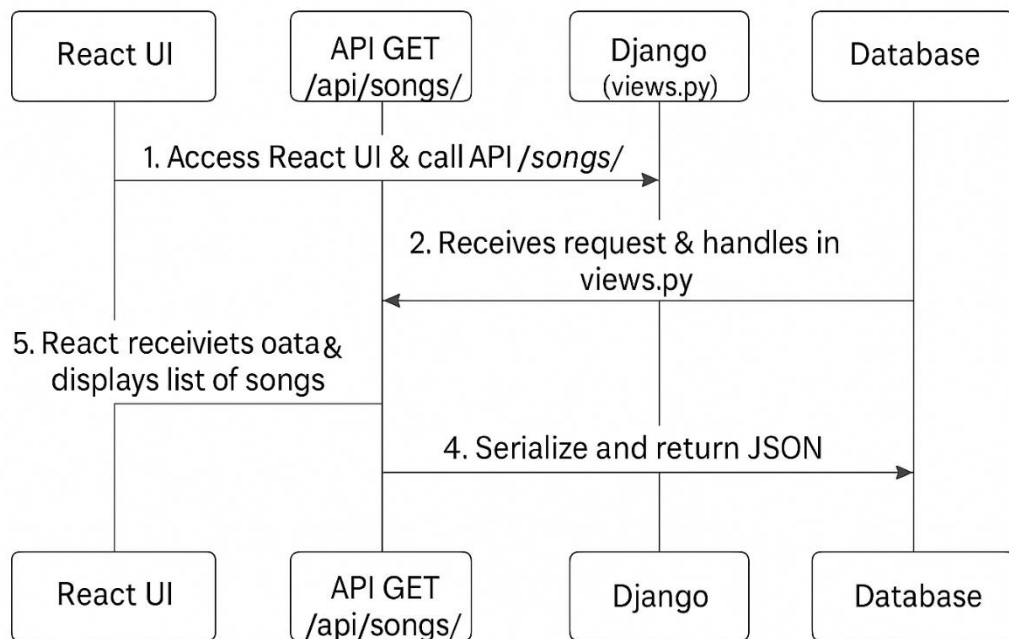
1. Truy cập React UI → gọi API GET /api/songs/

2. Django nhận request và xử lý tại views.py

3. Truy vấn dữ liệu từ model Song → Database

4. Serialize và trả về JSON

5. React nhận dữ liệu và hiển thị danh sách bài hát



e: Sơ Đồ Kết Nối Các Thành Phần

Người dùng → Frontend (ReactJS)

Frontend → Backend (Django REST API)

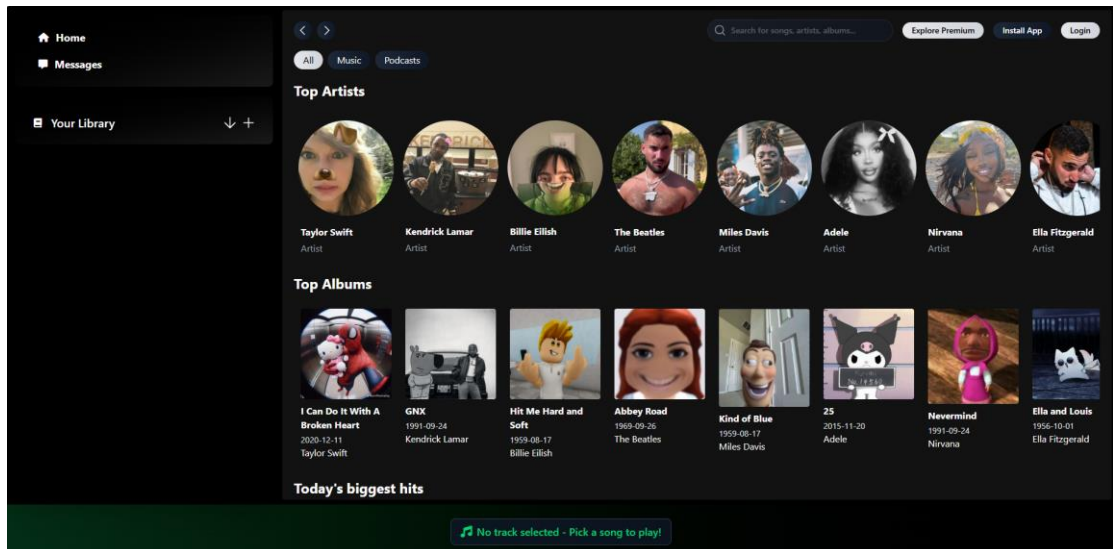
Backend → Database (PostgreSQL)

Database → Backend → Frontend → Người dùng

Chương 4: Sản phẩm Spotify.

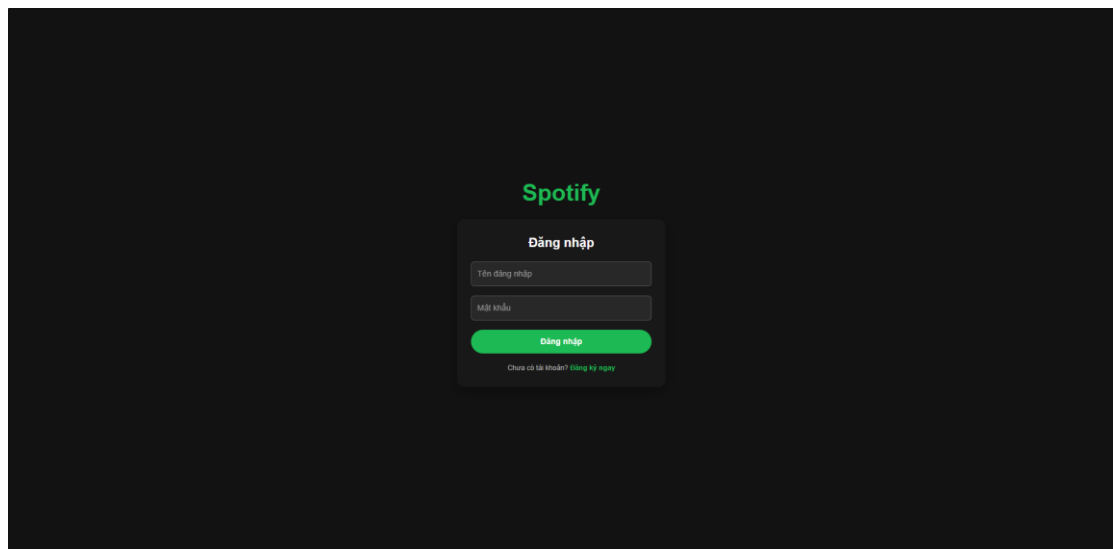
4.1: Màn hình user.

* Màn hình chính

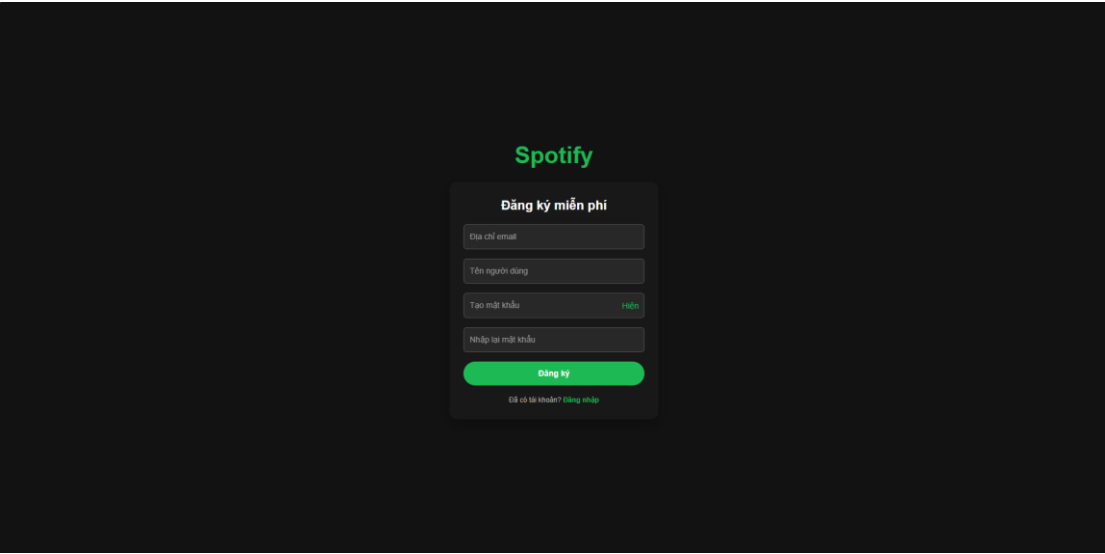


Màn hình chính (chưa login)

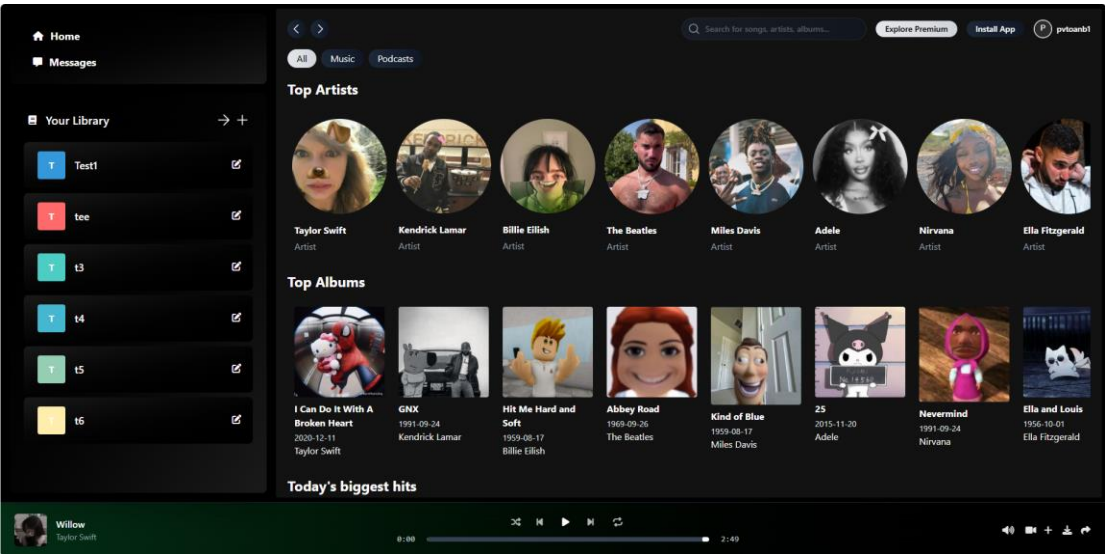
* Màn hình đăng nhập, đăng kí.



Màn hình đăng nhập

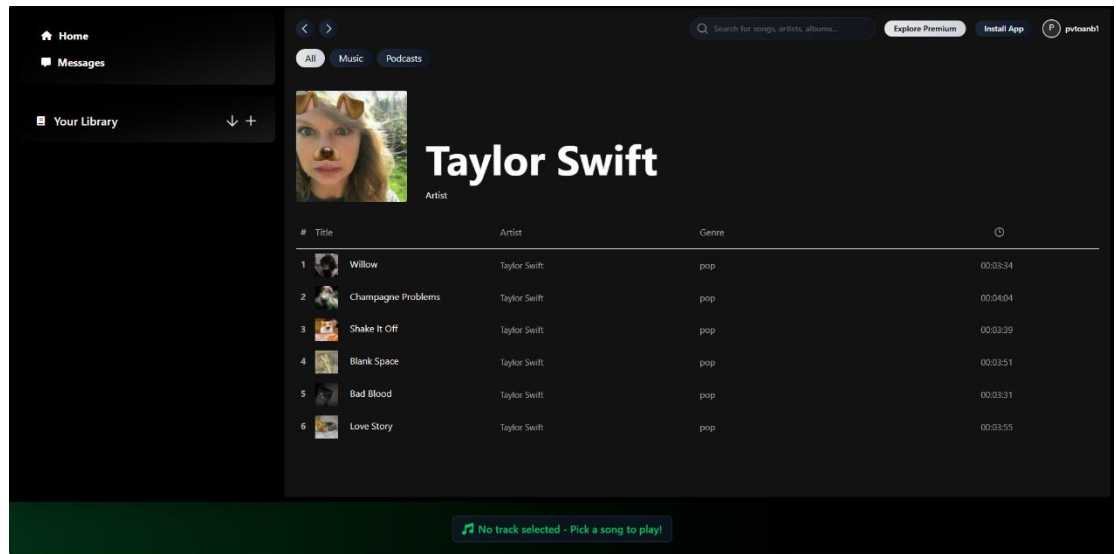


Màn hình đăng kí

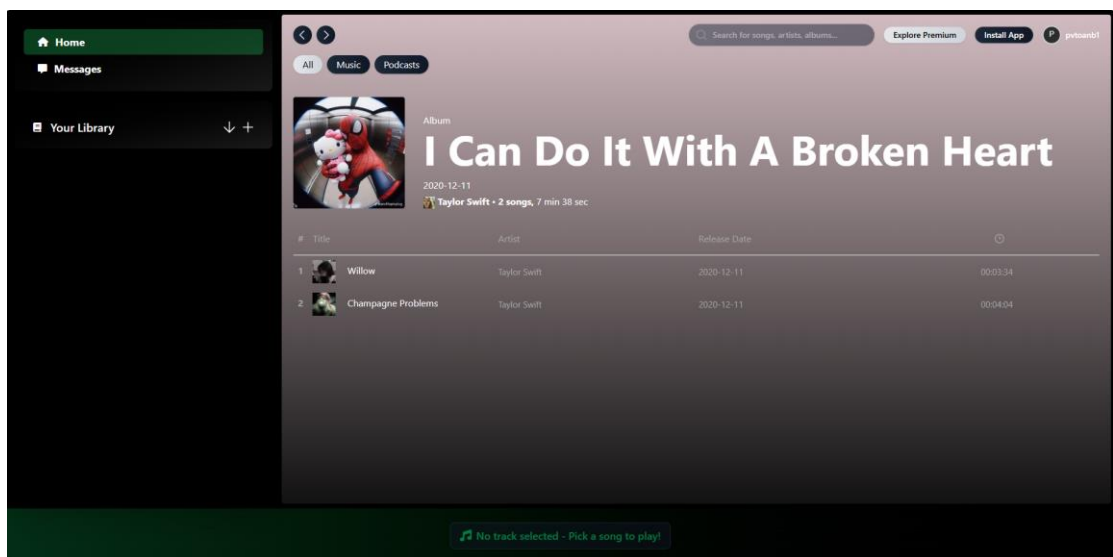


Đã đăng nhập

* Màn hình hồ sơ nghệ sĩ.

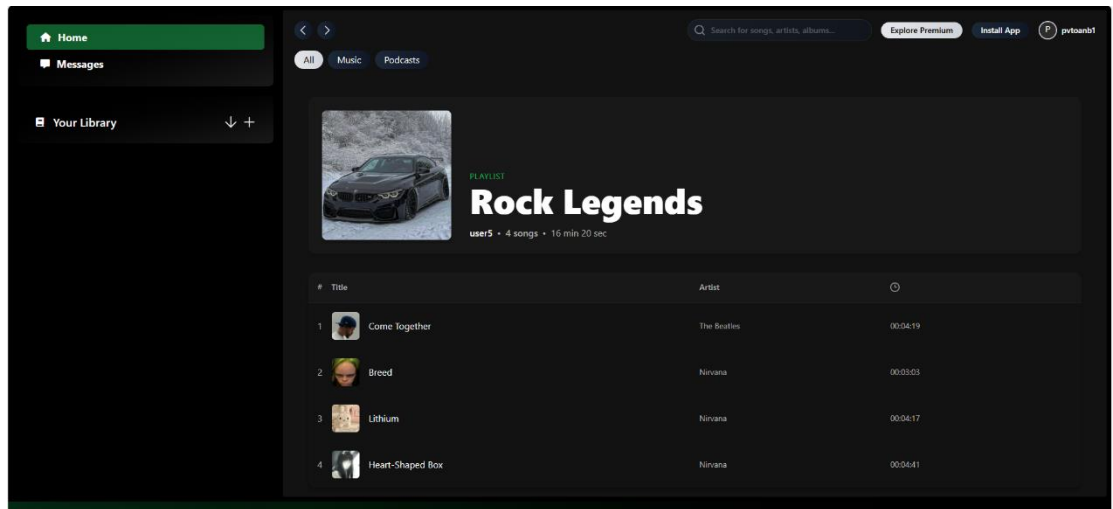


Hồ sơ artist



Màn hình album

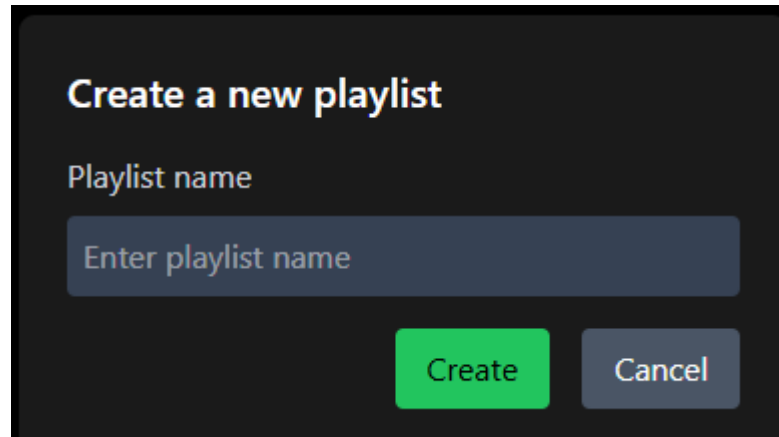
* Màn hình Playlist



Playlist

The 'Edit playlist' form has a title 'Edit playlist'. It contains a 'Playlist name' label and a text input field with the value 'Test1'. Below this is a 'Cover image (optional)' label and a file selection button that says 'Choose File No file chosen'. At the bottom right are two buttons: 'Save' (green) and 'Cancel' (grey).

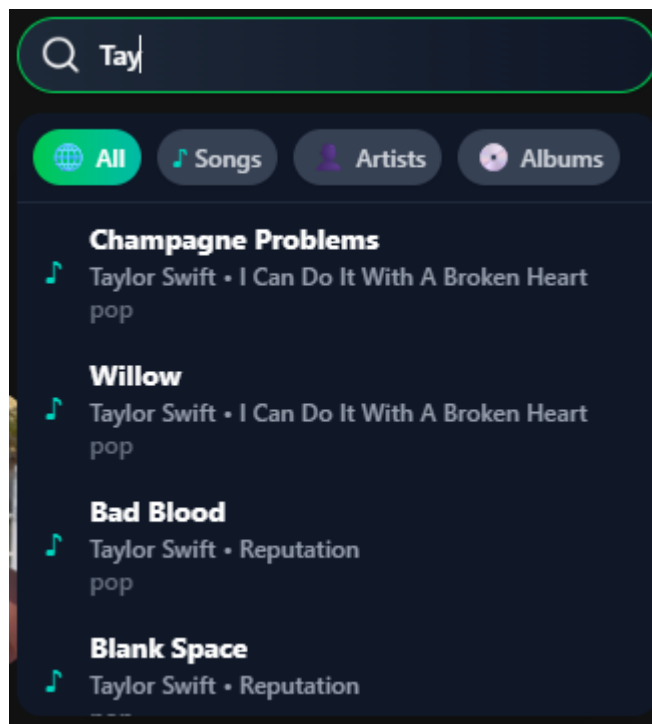
Form edit playlist



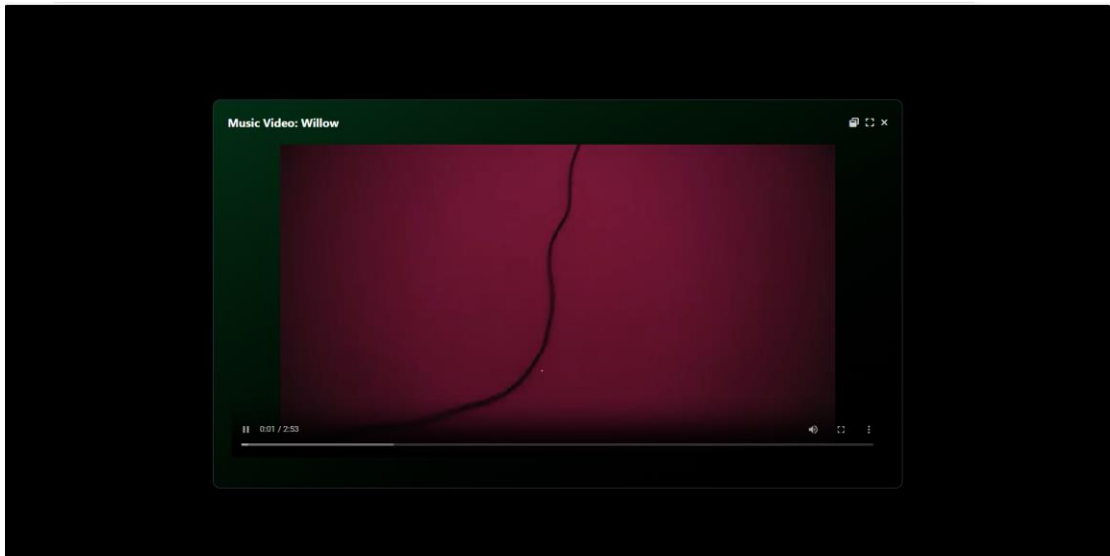
The screenshot shows a dark-themed modal window titled "Create a new playlist". Below the title is a label "Playlist name" followed by a text input field containing the placeholder text "Enter playlist name". At the bottom right of the modal are two buttons: a green "Create" button and a grey "Cancel" button.

Form add playlist

* Màn hình tìm kiếm và xem MV

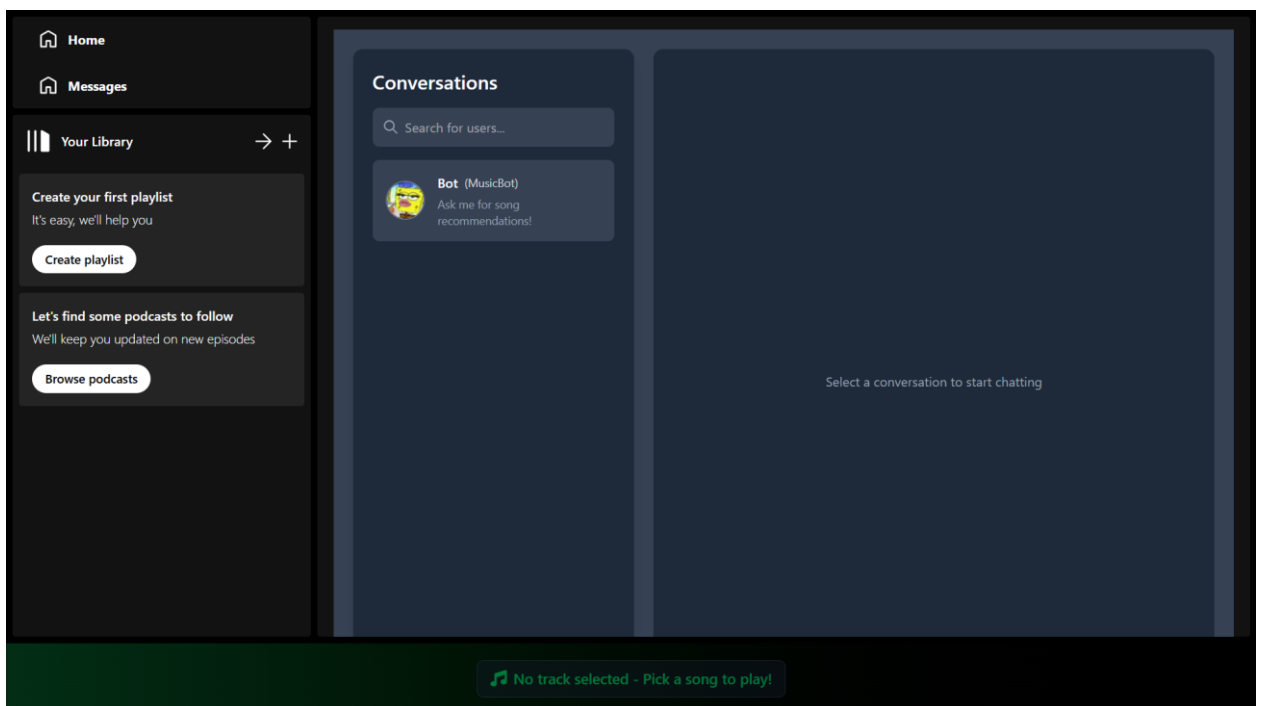


Searchbar

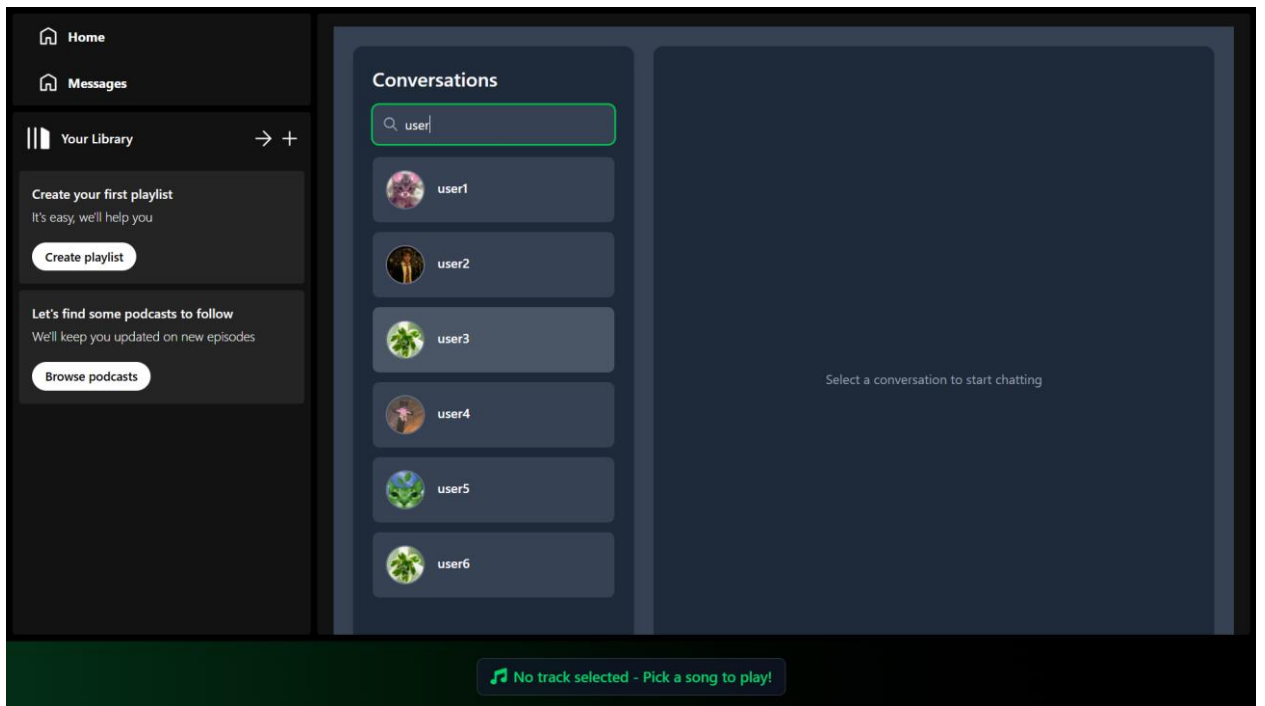


Xem MV

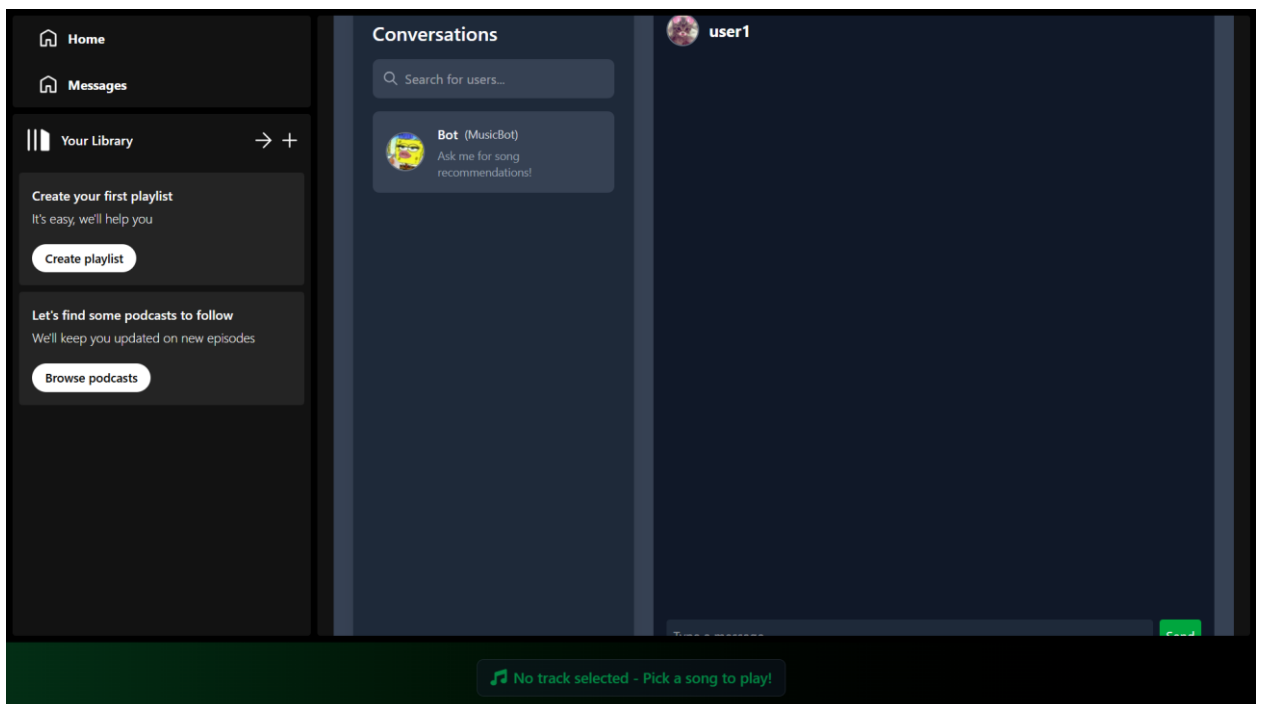
4.2: Giao diện gợi tin nhắn.



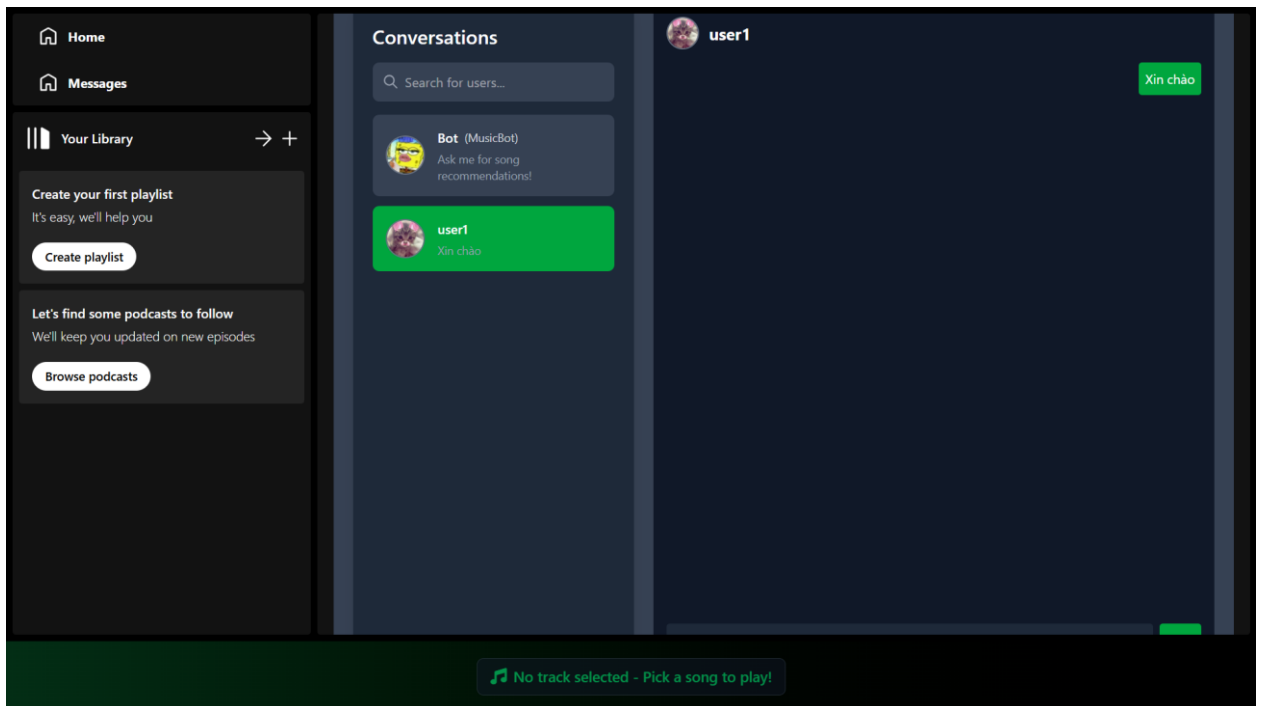
Màn hình tìm kiếm user để bắt đầu cuộc trò chuyện



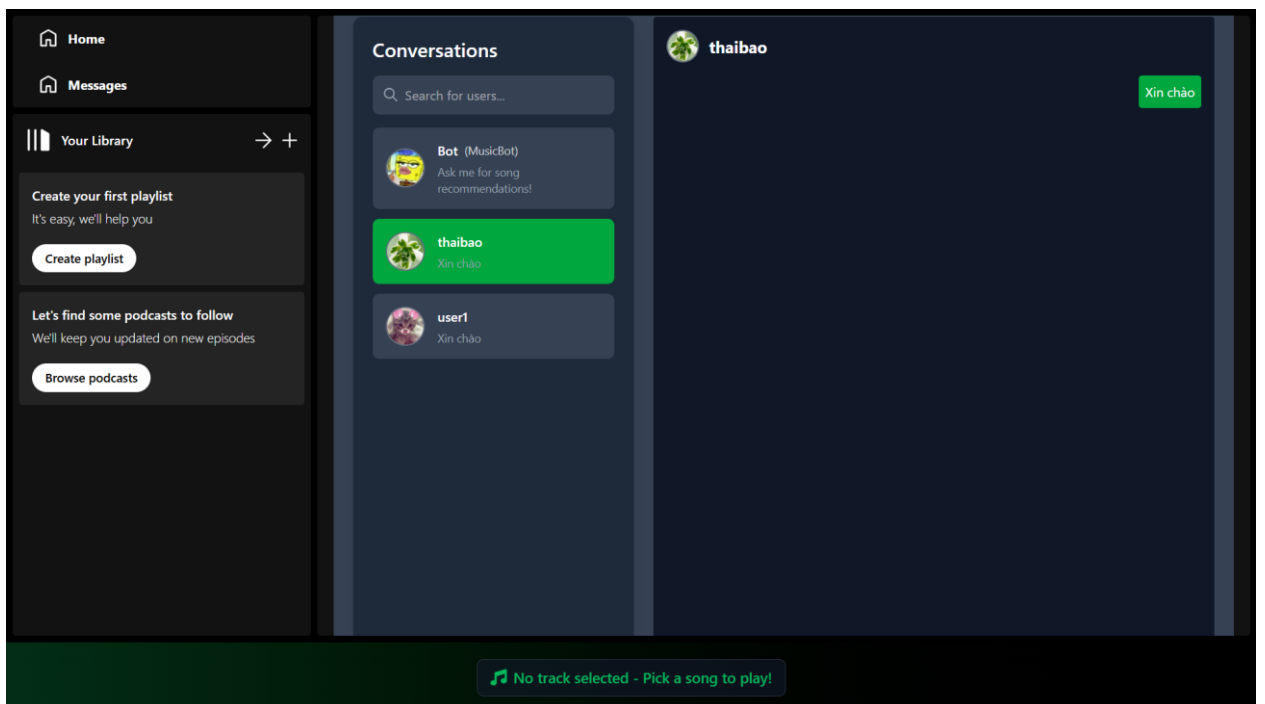
Chọn user để bắt đầu cuộc trò chuyện



Màn hình nhắn tin



Chuyển đổi qua lại màn hình nhắn tin của các user với nhau



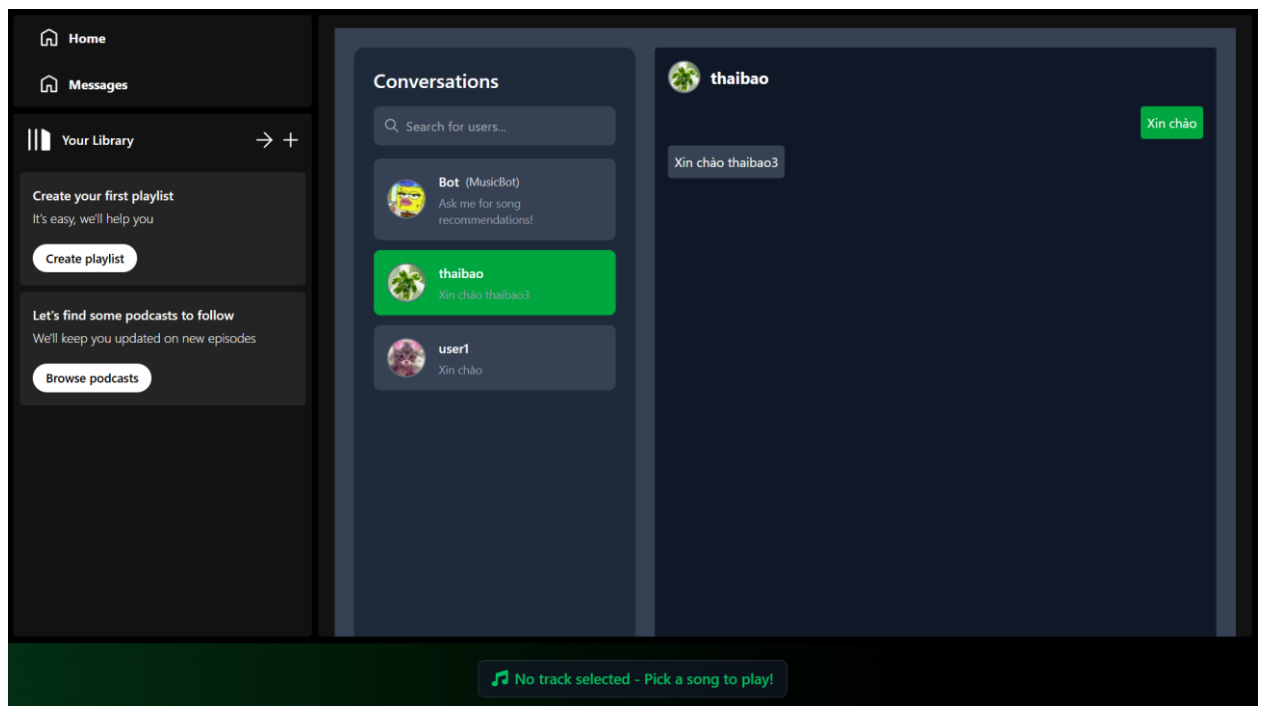
Màn hình tin nhắn cuộc trò chuyện



thaibao3

Xin chào

Màn hình tin nhắn cuộc trò chuyện



Bắt đầu cuộc trò chuyện

4.3: Màn hình admin.

Spotify Admin

Người dùng

Nghệ sĩ

Album

Bài hát

Playlist

Theo dõi nghệ sĩ

Lượt thích bài hát

Thống kê

Đăng xuất

Danh sách người dùng

Thêm người dùng

| ID | Tên người dùng | Email | Vai trò | Ngày tham gia | Hành động |
|----|----------------|-------------------|---------------|---------------------|--------------|
| 1 | user11 | user1@example.com | Người dùng | 17:00:00 01/03/2025 | Xem chi tiết |
| 2 | user2 | user2@example.com | Quản trị viên | 22:30:00 02/03/2025 | Xem chi tiết |
| 3 | user3 | user3@example.com | Người dùng | 16:15:00 03/03/2025 | Xem chi tiết |
| 4 | user4 | user4@example.com | Người dùng | 21:00:00 05/03/2025 | Xem chi tiết |
| 5 | user5 | user5@example.com | Người dùng | 16:30:00 06/03/2025 | Xem chi tiết |
| 6 | user6 | user6@example.com | Người dùng | 18:15:00 07/03/2025 | Xem chi tiết |
| 7 | john_doe | john@example.com | Người dùng | 15:00:00 08/03/2025 | Xem chi tiết |
| 8 | jane_smith | jane@example.com | Người dùng | 16:00:00 08/03/2025 | Xem chi tiết |
| 9 | mike_jones | mike@example.com | Người dùng | 17:00:00 08/03/2025 | Xem chi tiết |
| 10 | emma_watson | emma@example.com | Người dùng | 18:00:00 08/03/2025 | Xem chi tiết |

<

1

>

Trang quản lý người dùng

Spotify Admin

Người dùng

Nghệ sĩ

Album

Bài hát

Playlist

Theo dõi nghệ sĩ

Lượt thích bài hát

Thống kê

Đăng xuất

Danh sách người dùng

Thêm người dùng

| ID | Tên người dùng | Email | Ngày tham gia | Hành động |
|----|----------------|-------------------|---------------------|--------------|
| 1 | user11 | user1@example.com | 17:00:00 01/03/2025 | Xem chi tiết |
| 2 | user2 | user2@example.com | 22:30:00 02/03/2025 | Xem chi tiết |
| 3 | user3 | user3@example.com | 16:15:00 03/03/2025 | Xem chi tiết |
| 4 | user4 | user4@example.com | 21:00:00 05/03/2025 | Xem chi tiết |
| 5 | user5 | user5@example.com | 16:30:00 06/03/2025 | Xem chi tiết |
| 6 | user6 | user6@example.com | 18:15:00 07/03/2025 | Xem chi tiết |
| 7 | john_doe | john@example.com | 15:00:00 08/03/2025 | Xem chi tiết |
| 8 | jane_smith | jane@example.com | 16:00:00 08/03/2025 | Xem chi tiết |
| 9 | mike_jones | mike@example.com | 17:00:00 08/03/2025 | Xem chi tiết |
| 10 | emma_watson | emma@example.com | 18:00:00 08/03/2025 | Xem chi tiết |

<

1

>

Thông tin người dùng

Tên người dùng
john_doe

Email
john@example.com

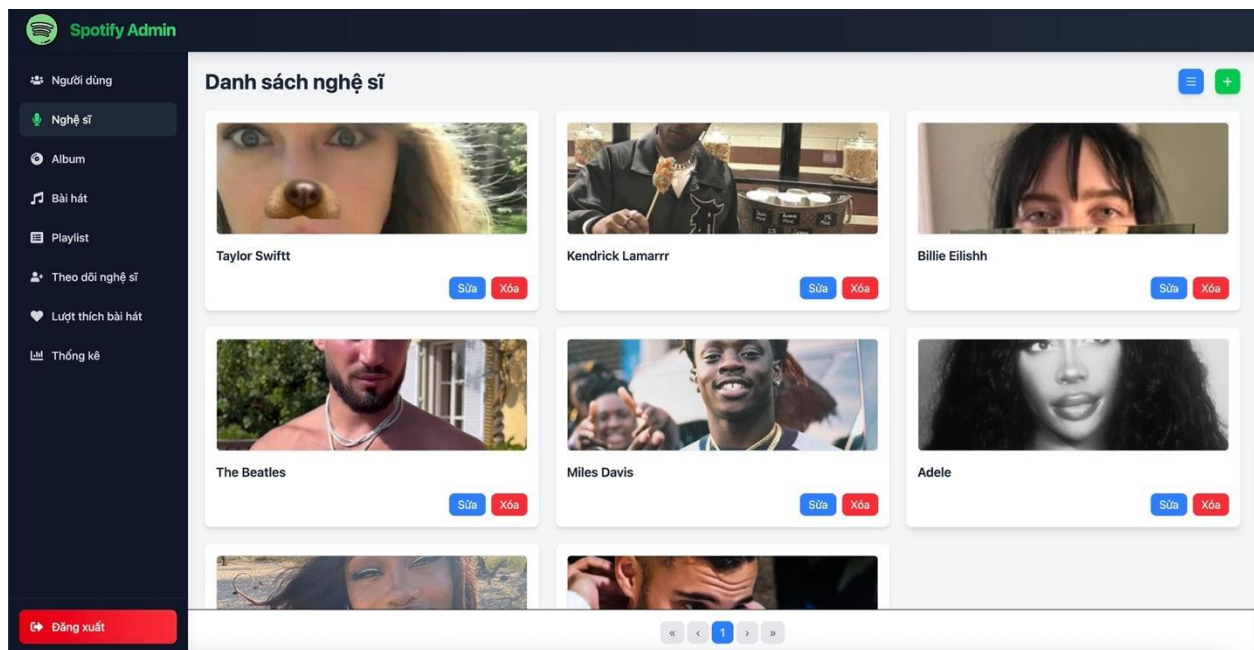
Ngày tham gia
15:00:00 08/03/2025

Ngày sinh
07:00:00 15/01/1992

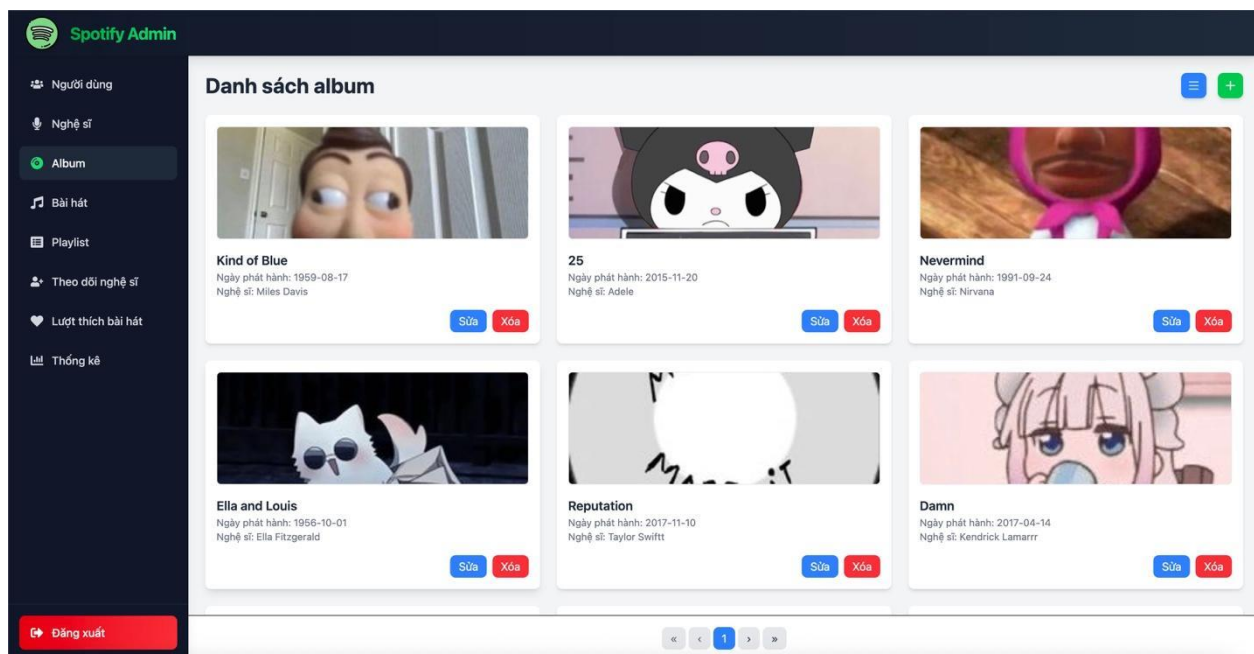
Vai trò
Người dùng

SửaXóa

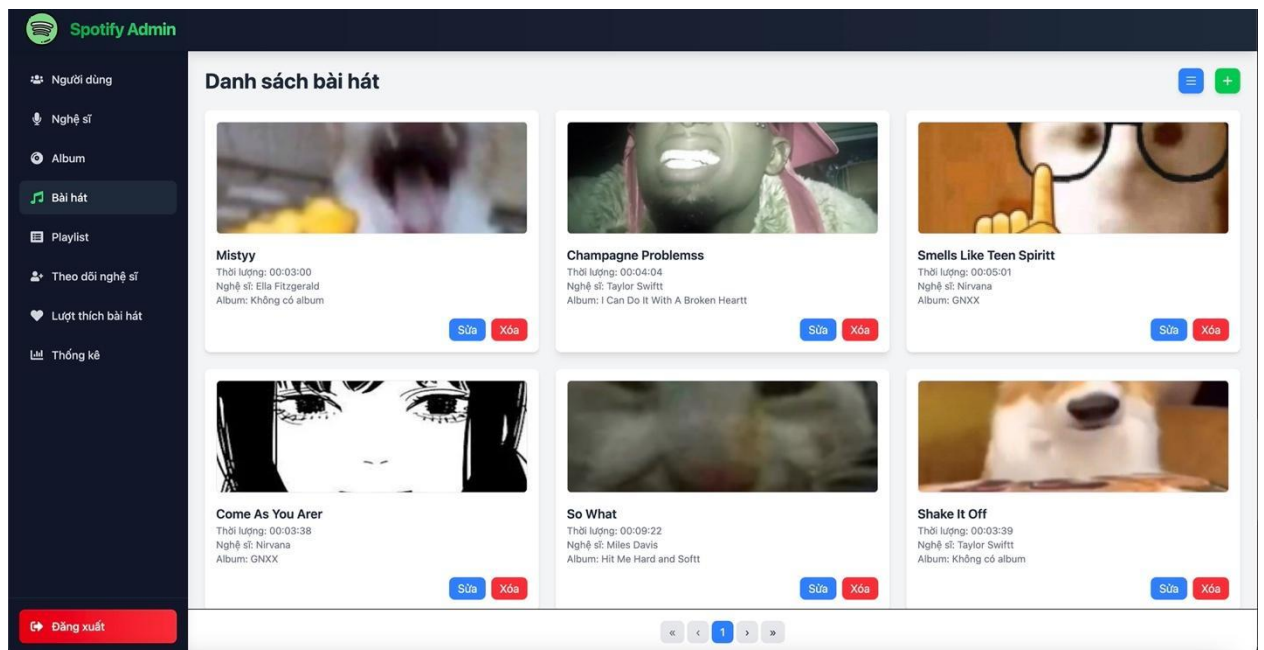
Xem chi tiết người dùng



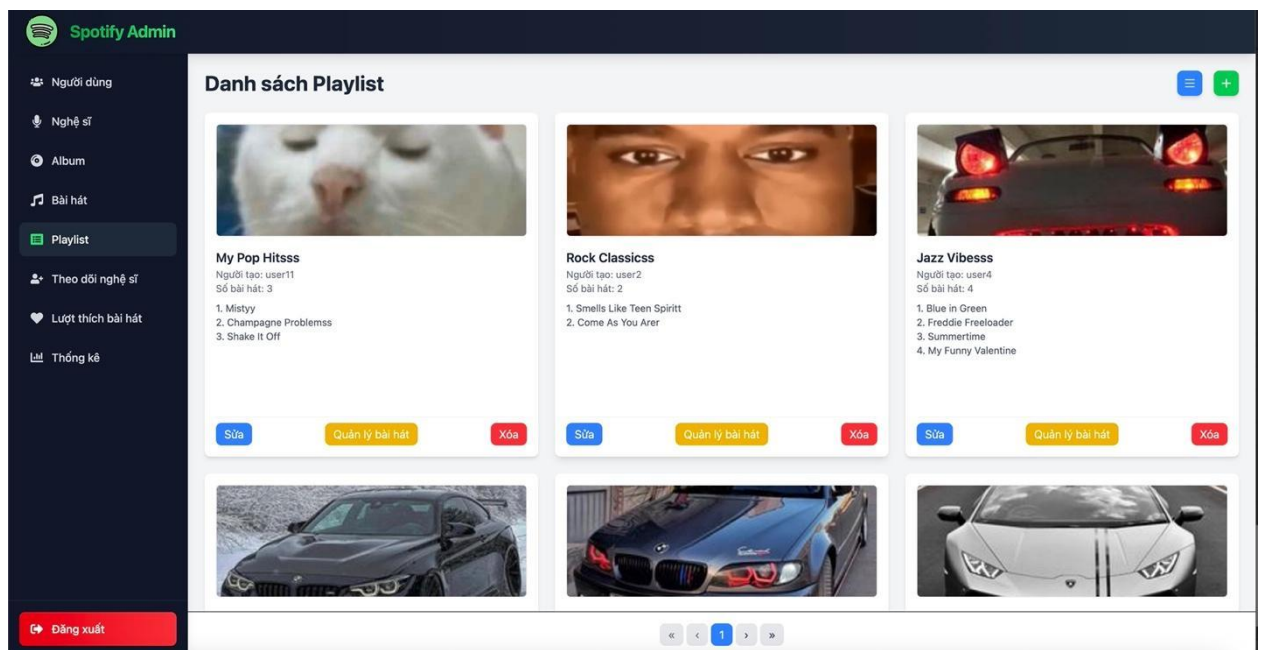
Trang quản lý nghệ sĩ



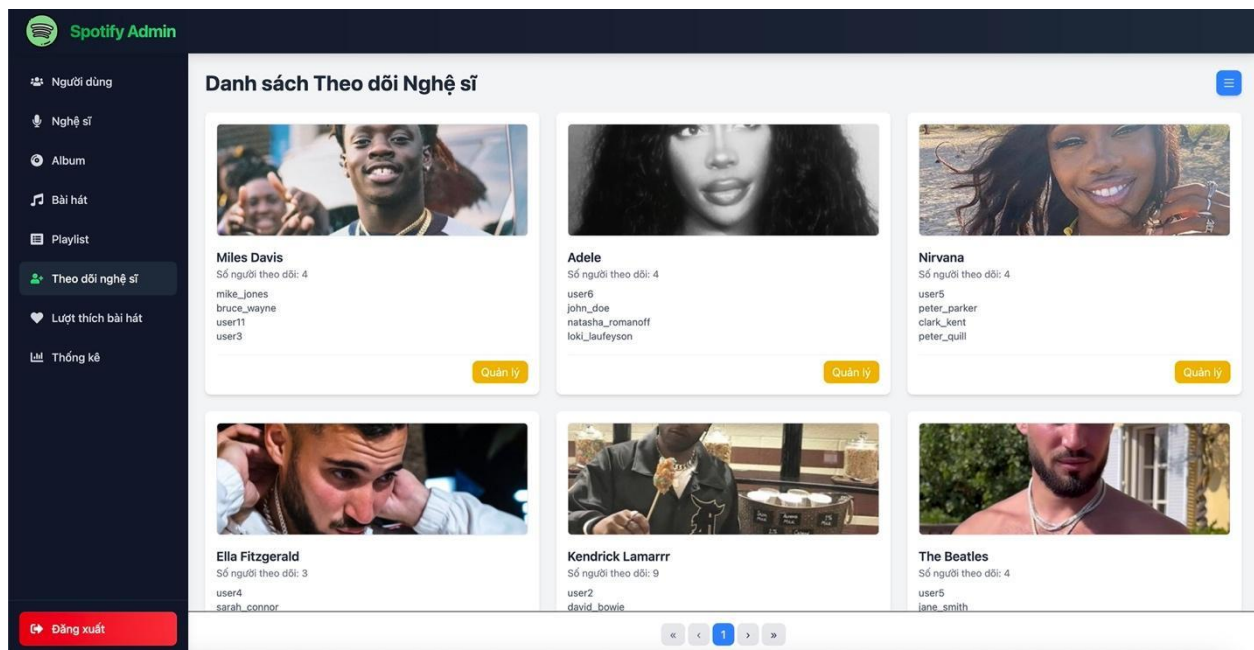
Trang quản lý album



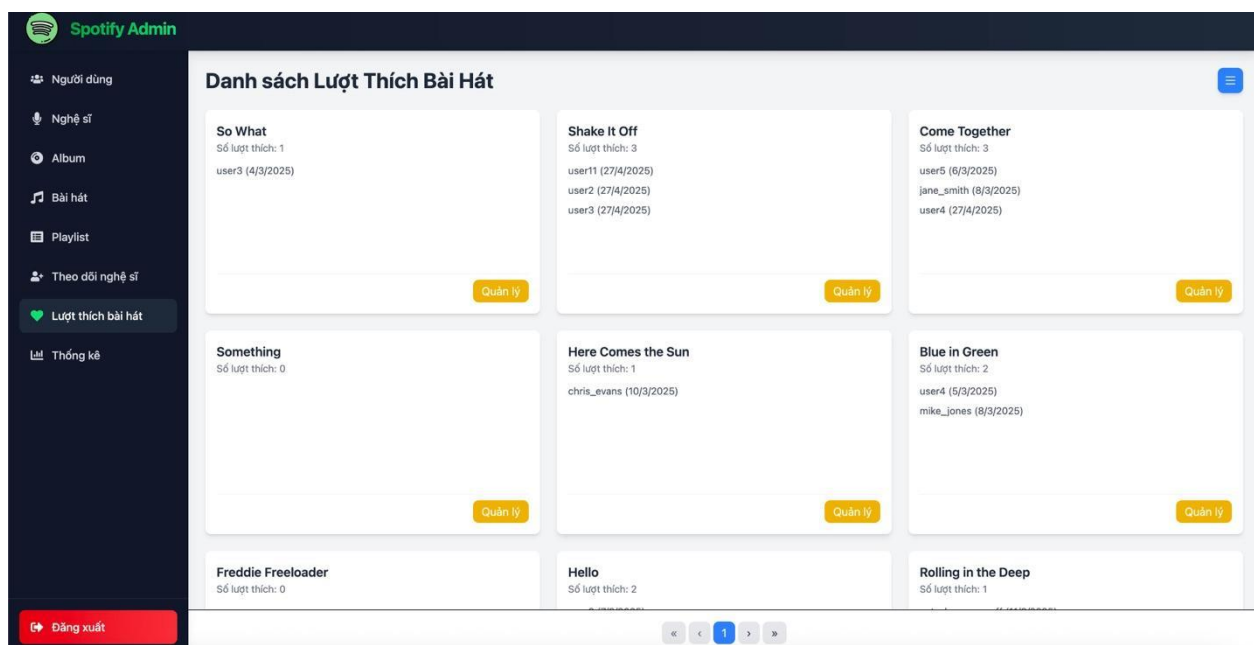
Trang quản lý bài hát



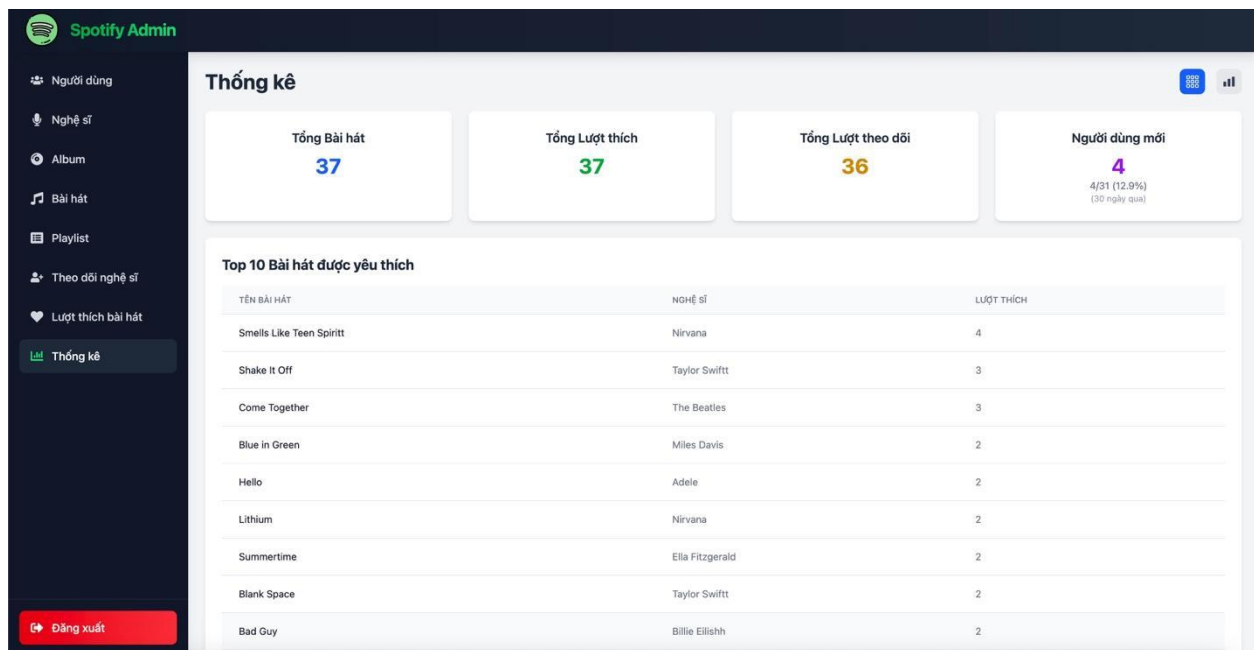
Trang quản lý playlist



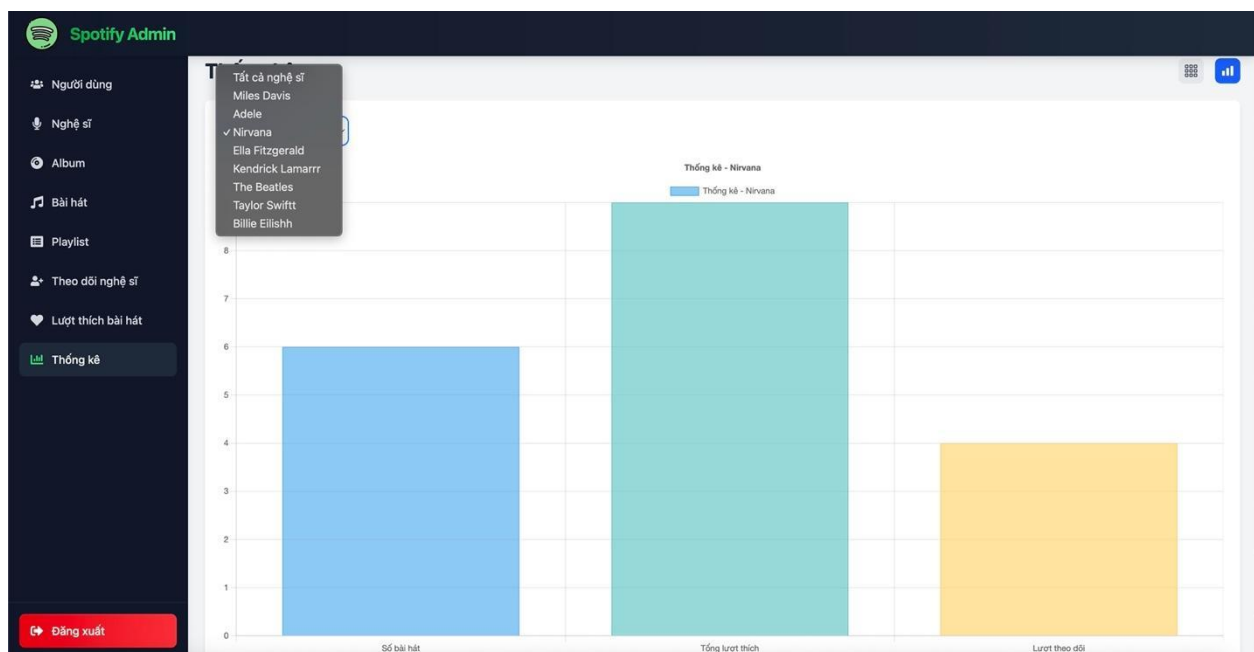
Trang quản lý lượt theo dõi nghệ sĩ



Trang quản lý lượt thích bài hát



Trang thống kê tổng



Biểu đồ thống kê theo từng nghệ sĩ

Chương 5. Kết luận

5.1. Nhận Xét Chung

Dự án Spotify đã thành công trong việc xây dựng một ứng dụng web phát nhạc trực tuyến, mô phỏng các tính năng cốt lõi của Spotify, bao gồm phát nhạc, quản lý playlist, tìm kiếm bài hát, quản trị nội dung, và chat thời gian thực. Hệ thống được triển khai dựa trên mô hình Client-Server, với sự tách biệt rõ ràng giữa frontend (ReactJS), backend (Django REST Framework), cơ sở dữ liệu (PostgreSQL), và server AI (A2A Protocol).

Các công nghệ được lựa chọn đã phát huy hiệu quả:

- ReactJS cung cấp giao diện người dùng tương tác, hỗ trợ phát nhạc qua React Player và hiển thị dữ liệu như danh sách bài hát, playlist, và khung chat.
- Django REST Framework đảm bảo xử lý logic nghiệp vụ, cung cấp API RESTful ổn định (ví dụ: /api/tracks/, /api/messages/)
- PostgreSQL lưu trữ dữ liệu hiệu quả với các bảng như users, tracks, messages, được tối ưu hóa bằng index và sequence (setval).
- A2A Protocol và Gemini nâng cao trải nghiệm chat bằng gợi ý trả lời thông minh, giảm tải server thông qua Server-Sent Events (SSE).
- Môi trường phát triển trên Ubuntu đảm bảo tính đồng nhất và dễ triển khai.

Cấu trúc mã nguồn được tổ chức khoa học, với các thư mục frontend/, backend/, và a2a_server/ tách biệt, dễ bảo trì và mở rộng. Kỹ thuật polling, dù đơn giản, đã đáp ứng yêu cầu cập nhật tin nhắn gần thời gian thực, trong khi tích hợp A2A/Gemini mang lại tiềm năng cá nhân hóa trải nghiệm người dùng. Dự án đáp ứng tốt mục tiêu đề ra, thể hiện khả năng kết hợp các công nghệ hiện đại để xây dựng một hệ thống phát nhạc trực tuyến mạnh mẽ, an toàn, và thân thiện với người dùng.

5.2. Hướng Phát Triển Trong Tương Lai

Để nâng cao chất lượng và mở rộng quy mô của Spotify, một số hướng phát triển có thể được xem xét:

- Tối ưu hóa chức năng chat: Thay thế kỹ thuật polling bằng WebSocket hoặc Server-Sent Events (SSE) để cập nhật tin nhắn thời gian thực, giảm tải server và cải thiện độ trễ.

- Tích hợp dịch vụ bên ngoài: Kết nối với các API của Spotify, YouTube, hoặc SoundCloud để mở rộng thư viện bài hát và cung cấp nội dung phong phú hơn.
- Nâng cấp AI: Tăng cường khả năng của Gemini trong việc gợi ý bài hát dựa trên sở thích người dùng, phân tích ngữ cảnh chat, hoặc xử lý đa phương thức (hình ảnh, âm thanh).
- Mở rộng tính năng:
 - Thêm chức năng phát nhạc offline bằng cách lưu trữ tạm thời trên thiết bị.
 - Phát triển ứng dụng di động (React Native) để hỗ trợ đa nền tảng.
 - Tích hợp hệ thống đề xuất cá nhân hóa dựa trên lịch sử nghe và playlist.
- Tăng cường hiệu suất và bảo mật:
 - Sử dụng caching (Redis) để tăng tốc truy vấn API.
 - Áp dụng mã hóa end-to-end cho tin nhắn chat.
 - Triển khai trên cloud (AWS, Google Cloud) để hỗ trợ nhiều người dùng đồng thời.
- Cải thiện giao diện người dùng: Tối ưu hóa trải nghiệm trên thiết bị di động và thêm giao diện tối (dark mode).

Những cải tiến này sẽ giúp Spotify không chỉ duy trì tính cạnh tranh mà còn mở rộng khả năng phục vụ người dùng, hướng tới một nền tảng phát nhạc trực tuyến toàn diện hơn.