

TRƯỜNG ĐẠI HỌC SÀI GÒN
KHOA CÔNG NGHỆ THÔNG TIN



Phát Triển Phần Mềm Mã Nguồn Mở

Spotify Clone

GVHD: Từ Lãng Phiêu
SV: Mai Phúc Lâm - 3122410207
Nguyễn Hữu Lộc - 3122410219
Nguyễn Đức Duy Lâm - 3122410208
Văn Tuấn Kiệt - 3122410202
Nguyễn Đình Thông - 3122410400
Hồ Hưng Lộc - 3122410219

TP. HỒ CHÍ MINH, THÁNG 2/2024

Mục lục

Lời cảm ơn	6
1 GIỚI THIỆU	7
1.1 Giới thiệu sơ lược về đề tài	7
1.2 Lý do chọn đề tài	8
1.3 Mục tiêu	8
1.4 Phân công công việc cho các thành viên	9
2 ĐẶC TẢ YÊU CẦU PHẦN MỀM	10
2.1 Yêu cầu chức năng	10
2.1.1 Người dùng	10
2.1.2 Quản lý	11
2.2 Yêu cầu phi chức năng	11
3 KIẾN TRÚC PHẦN MỀM	13
3.1 Mô hình ứng dụng	13
3.2 Công cụ	16
3.3 Kiến trúc phần mềm	21
4 SƠ ĐỒ ERD	23
4.1 Mô hình Dữ liệu	23
4.1.1 Mô hình Người Dùng (NguoiDung)	23
4.1.2 Mô hình Nghệ Sĩ (NgheSi)	24
4.1.3 Mô hình Album	25
4.1.4 Mô hình Bài Hát (BaiHat)	25
4.1.5 Mô hình Danh Sách Phát (DanhSachPhat)	26
4.1.6 Mô hình Thanh Toán (ThanhToan)	26
5 Xây dựng giao diện người dùng	28
5.1 SideBarTop	28
5.2 LibraryCard	32

5.3	Search	35
5.4	DynamicGrid	37
5.5	ItemCard	39
5.6	PlaylistDetail	41
5.7	TrackDetail	44
5.8	VideoPlayer	44
5.9	TrackDisplayer	46
5.10	ButtonGroup	48
5.11	ControllerSlider	51
5.12	VolumeController	54
6	BÁO CÁO KẾT QUẢ	56
6.1	Chức năng phát nhạc	56
6.2	Chức năng phát video âm nhạc	56
6.3	Chức năng tải video âm nhạc	57
6.4	Chức năng tạo album và đánh dấu bài hát yêu thích	58
6.4.1	Chức năng tạo album	58
6.4.2	Chức năng đánh dấu bài hát yêu thích	61
6.5	Chức năng tạo playlist theo cảm xúc	61
6.6	Quản trị	62
6.6.1	Quản lý người dùng	62
6.6.2	Quản lý bài hát	66
6.6.3	Quản lý album	70
6.6.4	Quản lý nghệ sĩ	74
6.6.5	Quản lý loại bài hát	79
6.6.6	Thống kê	81
6.7	Tính năng chat tích hợp	82
6.8	Chức năng tìm kiếm	83
6.8.1	Tìm kiếm theo từ khóa	83
6.8.2	Tìm kiếm bài hát dựa trên tệp ghi âm	85
6.9	Tính năng chat tích hợp	85
6.10	Premium	86
6.10.1	Mua Premium thanh toán bằng PayPal	87
6.10.2	Mua Premium thanh toán bằng Zalo pay	88
6.11	Bảng xếp hạng và album hot	89
7	HƯỚNG PHÁT TRIỂN	90
8	MÔI TRƯỜNG CHẠY ỨNG DỤNG VÀ CÁCH CÀI ĐẶT	91
8.1	Môi trường chạy ứng dụng	91
8.2	Cách cài đặt	91

Danh sách hình vẽ

5.1	Side Bar Top	29
5.2	Tạo Playlist	29
5.3	Tạo Album	30
5.4	Tạo Playlist theo cảm xúc	31
5.5	Card Library	33
5.6	Trang tìm kiếm	36
5.7	DynamicGrid	38
5.8	ItemCard	40
5.9	Trang Playlist	42
5.10	Trang Track	44
5.11	Trang xem video	45
5.12	Bài hát đang phát	47
5.13	Các nút điều khiển	50
5.14	Giao diện thanh tua tiến trình bài hát	53
5.15	Thanh điều chỉnh âm lượng và biểu tượng loa	54
6.1	Giao diện chức năng phát nhạc	56
6.2	Giao diện chức năng phát video âm nhạc	57
6.3	Chức năng tải video âm nhạc	57
6.4	Giao diện hiện biểu mẫu tạo album	58
6.5	Giao diện điền thông tin tạo album	59
6.6	Giao diện tạo album thành công	59
6.7	Giao diện thêm bài hát vào album	60
6.8	Giao diện album đã bài hát thành công	61
6.9	Giao diện hiện biểu mẫu tạo playlist theo cảm xúc	61
6.10	Giao diện chọn cảm xúc	62
6.11	Giao diện tạo playlist theo cảm xúc thành công	62
6.12	Hiển thị danh sách người dùng	63
6.13	Xem chi tiết thông tin người dùng	63
6.14	Khóa tài khoản người dùng	64
6.15	Khóa tài khoản người dùng thành công	64
6.16	Mở khóa tài khoản người dùng	64

6.17	Mở khóa tài khoản người dùng thành công	65
6.18	Cấp quyền quản trị cho tài khoản	65
6.19	Cấp quyền quản trị cho tài khoản thành công	66
6.20	Danh sách bài hát	66
6.21	Thêm bài hát mới	67
6.22	Thêm bài hát mới thành công	67
6.23	Sửa bài hát	68
6.24	Sửa bài hát thành công	68
6.25	Khóa bài hát	69
6.26	Khóa bài hát thành công	69
6.27	Mở khóa bài hát	69
6.28	Mở khóa bài hát thành công	70
6.29	Tìm kiếm bài hát	70
6.30	Danh sách album	71
6.31	Sửa album	71
6.32	Sửa album thành công	72
6.33	Khóa album	72
6.34	Khóa album thành công	73
6.35	Mở khóa album	73
6.36	Mở khóa album thành công	74
6.37	Tìm kiếm album	74
6.38	Danh sách nghệ sĩ	75
6.39	Thêm nghệ sĩ mới	75
6.40	Thêm nghệ sĩ mới thành công	76
6.41	Sửa nghệ sĩ	76
6.42	Sửa nghệ sĩ thành công	77
6.43	Khóa nghệ sĩ	77
6.44	Khóa nghệ sĩ thành công	78
6.45	Mở khóa nghệ sĩ	78
6.46	Mở khóa nghệ sĩ thành công	79
6.47	Tìm kiếm nghệ sĩ	79
6.48	Danh sách loại bài hát	80
6.49	Sửa loại bài hát	80
6.50	Sửa loại bài hát	81
6.51	Xóa loại bài hát	81
6.52	Thông kê: Số lượng người dùng, bài hát, danh sách phát, nghệ sĩ, bài hát phát hành trong 1 năm	82
6.53	Thông kê: Tỷ lệ người dùng đăng ký Premium theo biểu đồ tròn và biểu đồ đường	82
6.54	Giao diện tính năng chat	83
6.55	Giao diện tìm kiếm theo từ khóa	83

6.56 Giao diện tìm kiếm nghệ sĩ	84
6.57 Giao diện tìm kiếm bài hát	84
6.58 Giao diện tìm kiếm bài hát	84
6.59 Giao diện tìm kiếm bài hát dựa trên tệp ghi âm	85
6.60 Giao diện tính năng chat	86
6.61 Giao diện mua premium	86
6.62 Mua Premium thanh toán bằng PayPal	87
6.63 Mua Premium thanh toán bằng PayPal	87
6.64 Mua Premium thanh toán bằng PayPal thành công	88
6.65 Mua Premium thanh toán bằng Zalo pay	88
6.66 Mua Premium thanh toán bằng Zalo pay thành công	89
6.67 Giao diện bảng xếp hạng và album hot	89

Lời cảm ơn

Trong suốt quá trình học tập môn Phần mềm mã nguồn mở và thực hiện phần mềm “Spotify”, chúng em xin gửi lời cảm ơn chân thành nhất đến tất cả những người đã hỗ trợ và đồng hành cùng chúng em trong suốt chặng đường này.

Trước tiên, chúng em xin bày tỏ lòng biết ơn sâu sắc đến giảng viên Từ Lãng Phiêu, người đã tận tình hướng dẫn chúng em từ những bước đầu tiên cho đến khi phần mềm được hoàn thành. Thầy không chỉ cung cấp những kiến thức quý báu mà còn tạo điều kiện thuận lợi để chúng em nghiên cứu, học hỏi và áp dụng vào thực tế. Sự hướng dẫn tận tâm, những góp ý sắc sảo cùng kinh nghiệm thực tế mà thầy chia sẻ đã giúp chúng em củng cố nền tảng chuyên môn vững chắc, đồng thời vượt qua những khó khăn trong quá trình phát triển phần mềm.

Bên cạnh đó, chúng em cũng xin gửi lời cảm ơn chân thành đến các thành viên trong nhóm, những người đã luôn hỗ trợ, đồng hành và chia sẻ kinh nghiệm trong suốt quá trình thực hiện dự án. Sự hợp tác chặt chẽ, tinh thần làm việc nhóm hiệu quả cùng những trải nghiệm thực tế từ các bạn đã giúp chúng em làm quen với quy trình làm việc chuyên nghiệp, tạo nền tảng quan trọng cho con đường phát triển sau này.

Mặc dù đã nỗ lực hết mình để hoàn thành phần mềm, nhưng do thời gian và kiến thức còn hạn chế, chắc chắn vẫn còn những thiếu sót. Chúng em rất mong nhận được sự góp ý từ quý thầy cô và các bạn để có thể tiếp tục cải thiện và hoàn thiện sản phẩm hơn nữa. Những ý kiến đóng góp của thầy cô sẽ là nguồn động lực to lớn, giúp chúng em rút ra bài học kinh nghiệm quý báu và không ngừng phát triển trong tương lai.

Chúng em xin chân thành cảm ơn!

Chương 1

GIỚI THIỆU

1.1 Giới thiệu sơ lược về đề tài

Spotify là một nền tảng phát nhạc trực tuyến hàng đầu thế giới, cung cấp cho người dùng quyền truy cập vào hàng triệu bài hát, podcast và video từ các nghệ sĩ trên toàn cầu. Ra mắt vào năm 2008, Spotify đã thay đổi cách mọi người tiếp cận và thưởng thức âm nhạc, chuyển từ việc sở hữu bản ghi vật lý sang mô hình phát trực tuyến tiện lợi và hợp pháp.

Spotify hoạt động trên nhiều thiết bị, bao gồm máy tính, điện thoại di động, máy tính bảng, loa thông minh, TV và ô tô, cho phép người dùng nghe nhạc mọi lúc, mọi nơi. Nền tảng này cung cấp cả phiên bản miễn phí với quảng cáo và phiên bản trả phí (Spotify Premium) với nhiều tính năng nâng cao như nghe nhạc offline, chất lượng âm thanh cao hơn và không có quảng cáo. Spotify cũng nổi tiếng với khả năng cá nhân hóa trải nghiệm người dùng thông qua các playlist được tạo tự động dựa trên thói quen nghe nhạc, như "Discover Weekly" và "Daily Mix".



1.2 Lý do chọn đề tài

Trước khi Spotify xuất hiện, ngành công nghiệp âm nhạc đối mặt với nhiều thách thức, đặc biệt là vấn đề vi phạm bản quyền do việc chia sẻ nhạc trái phép trên các nền tảng như Napster. Daniel Ek và Martin Lorentzon, những người sáng lập Spotify, nhận thấy cần thiết phải tạo ra một dịch vụ âm nhạc trực tuyến hợp pháp, cung cấp trải nghiệm nghe nhạc chất lượng cao và thuận tiện, đồng thời đảm bảo quyền lợi cho các nghệ sĩ và nhà sản xuất. Mục tiêu của họ là cung cấp một giải pháp thay thế hấp dẫn hơn so với việc tải nhạc bất hợp pháp, bằng cách mang đến cho người dùng quyền truy cập tức thì vào một thư viện âm nhạc khổng lồ với chất lượng cao.

1.3 Mục tiêu

Mục tiêu chính của Spotify là "democratize audio"—dân chủ hóa việc tiếp cận âm thanh. Điều này có nghĩa là cung cấp cho người dùng trên toàn thế giới quyền truy cập dễ dàng và hợp pháp vào kho nội dung âm thanh phong phú, đồng thời tạo cơ hội cho các nghệ sĩ, dù lớn hay nhỏ, tiếp cận với khán giả toàn cầu mà không cần thông qua các kênh phân phối truyền thống. Spotify cũng đặt mục tiêu không ngừng cải thiện trải nghiệm người dùng thông qua việc ứng dụng công nghệ tiên tiến như trí tuệ nhân tạo để cá nhân hóa nội dung và đề xuất âm nhạc phù hợp với sở thích của từng cá nhân.

Ngoài ra, Spotify còn hướng đến việc mở rộng hệ sinh thái âm thanh của

mình bằng cách tích hợp podcast và audiobook, biến nền tảng này thành điểm đến toàn diện cho mọi nhu cầu nghe của người dùng. Điều này không chỉ tăng cường giá trị cho người dùng mà còn tạo thêm nguồn thu nhập cho các nhà sáng tạo nội dung.

Tóm lại, Spotify được phát triển với mục tiêu cung cấp một giải pháp nghe nhạc trực tuyến hợp pháp, chất lượng cao và thuận tiện, đồng thời hỗ trợ các nghệ sĩ tiếp cận khán giả rộng rãi hơn. Nền tảng này không ngừng đổi mới và mở rộng để đáp ứng nhu cầu ngày càng cao của người dùng và thị trường âm nhạc toàn cầu.

1.4 Phân công công việc cho các thành viên

Thành viên	Công việc đã làm
Văn Tuấn Kiệt	Thiết kế yêu cầu chức năng, API bảng xếp hạng, tích hợp thanh toán ZaloPay/Visa, quản lý bài hát/thẻ loại, chức năng Premium, sửa chức năng phát nhạc, tổng hợp đóng góp, mô tả ERD.
Nguyễn Đức Duy Lâm	Tạo source Frontend, API danh sách phát và bài hát, tích hợp danh sách phát, chức năng phát nhạc, phát video, fix lỗi, viết README.
Hồ Hưng Lộc	Viết báo cáo (bìa, mục lục, lời cảm ơn, giới thiệu đề tài, yêu cầu chức năng/phi chức năng, công nghệ), vẽ ERD, API bài hát và tìm kiếm, giao diện admin, tạo album/danh sách phát theo cảm xúc.
Nguyễn Hữu Lộc	Thiết kế database, API nghệ sĩ, API thanh toán, tích hợp Frontend nghệ sĩ/album/thanh toán, chức năng thống kê, cải tiến quản lý bài hát.
Mai Phúc Lâm	Tạo source Backend, code model, API đăng nhập/đăng ký/đăng xuất/quên mật khẩu, tích hợp đăng nhập/tìm kiếm vào Frontend, chat real-time bằng WebSocket.
Nguyễn Đình Thông	Quản lý thể loại bài hát (BE), tích hợp FE và BE Danh sách phát, quản lý album (BE), quản lý người dùng (Xem danh sách người dùng, khóa/mở khóa tài khoản, thêm quyền admin), giao diện đăng ký premium, tìm bài hát từ tệp ghi âm (BE - FE).

Chương 2

ĐẶC TẢ YÊU CẦU PHÂN MỀM

2.1 Yêu cầu chức năng

2.1.1 Người dùng

- **Đăng ký và đăng nhập:** Người dùng có thể tạo tài khoản và đăng nhập vào hệ thống. Người dùng có thể đặt lại mật khẩu nếu quên mật khẩu.
- **Tìm kiếm bài hát:** Người dùng có thể tìm kiếm bài hát bằng tên bài hát, tên nghệ sĩ và có thể tìm kiếm album bằng tên bài hát, tên nghệ sĩ.
- **Phát nhạc:** Người dùng có thể phát bài hát, album, và danh sách phát yêu thích.
- **Danh sách phát:** Người dùng có thể thêm bài hát vào danh sách phát cá nhân của mình hay tải xuống.
- **Quản lý thư viện cá nhân:** Người dùng có thể xem và chỉnh sửa danh sách bài hát, bài hát yêu thích.
- **Quản lý thông tin cá nhân:** Người dùng có thể xem thông tin cá nhân của mình.
- **Premium:** Nếu người dùng chưa đăng ký tài khoản premium thì sẽ phải nghe các quảng cáo sau mỗi 2 bài hát, đồng thời không được nghe các bài hát vip.

- **Thanh toán:** Người dùng có thể thanh toán cho các gói dịch vụ cao cấp qua các phương thức thanh toán trực tuyến bao gồm ZaloPay, Paypal.
- **Tìm bài hát dựa trên giai điệu:** Người dùng có thể tìm bài hát thông qua việc ghi âm một đoạn nhạc nào đó, hệ thống sẽ hiển thị bài hát tương ứng nếu tìm ra.

2.1.2 Quản lý

- **Quản lý người dùng:** Quản lý có thể xem danh sách người dùng, thêm, cập nhật, và tìm kiếm người dùng theo tên, email, hoặc số điện thoại.
- **Quản lý danh sách phát chung:** Quản lý có thể tạo và duy trì các danh sách phát chung cho người dùng.
- **Thông kê:** Quản lý có thể thống kê số lượng người dùng, lượt nghe, và các bài hát phổ biến theo thời gian.
- **Quản lý album:** Quản lý có thể xem và quản lý thông tin các album.
- **Quản lý nghệ sĩ:** Quản lý có thể xem và quản lý thông tin các nghệ sĩ.

2.2 Yêu cầu phi chức năng

Hiệu suất

- **Tìm kiếm nhanh chóng:** Các thao tác tìm kiếm phải có thời gian phản hồi dưới 5 giây.
- **Xử lý giao dịch đồng thời:** Hệ thống phải có khả năng xử lý 500 giao dịch đồng thời mà không gặp phải độ trễ.

Tính bảo mật

- **Mã hóa SSL cho thanh toán:** Đảm bảo các giao dịch tài chính được bảo vệ thông qua SSL/TLS.
- **Xác thực hai yếu tố (2FA):** Sử dụng xác thực OTP hoặc sinh trắc học để tăng cường bảo mật khi người dùng thực hiện giao dịch nhạy cảm.

Tính sẵn có

- **Hoạt động 24/7:** Hệ thống cần phải luôn sẵn sàng với mức độ uptime 99.9

Khả năng sử dụng

- **Giao diện thân thiện:** Giao diện người dùng phải đơn giản và dễ sử dụng, với các tính năng quan trọng hiển thị rõ ràng và dễ tìm.
- **Hỗ trợ đa ngôn ngữ:** Hệ thống cần cung cấp hỗ trợ cho nhiều ngôn ngữ và đảm bảo khả năng truy cập cho người dùng có nhu cầu đặc biệt.

Tính tương thích

- **Tương thích với các trình duyệt:** Ứng dụng cần phải tương thích với các trình duyệt web phổ biến như Chrome, Cốc cốc, và Edge.
- **Tương thích với thiết bị di động:** Giao diện cần phải responsive và tương thích với các hệ điều hành di động (iOS, Android).

Tính bảo trì

- **Dễ bảo trì và mở rộng tính năng:** Cần áp dụng kiến trúc modular để dễ dàng thêm hoặc thay đổi chức năng mà không ảnh hưởng đến toàn bộ hệ thống.
- **CI/CD:** Sử dụng quy trình CI/CD để tự động triển khai và cập nhật hệ thống mà không cần downtime.

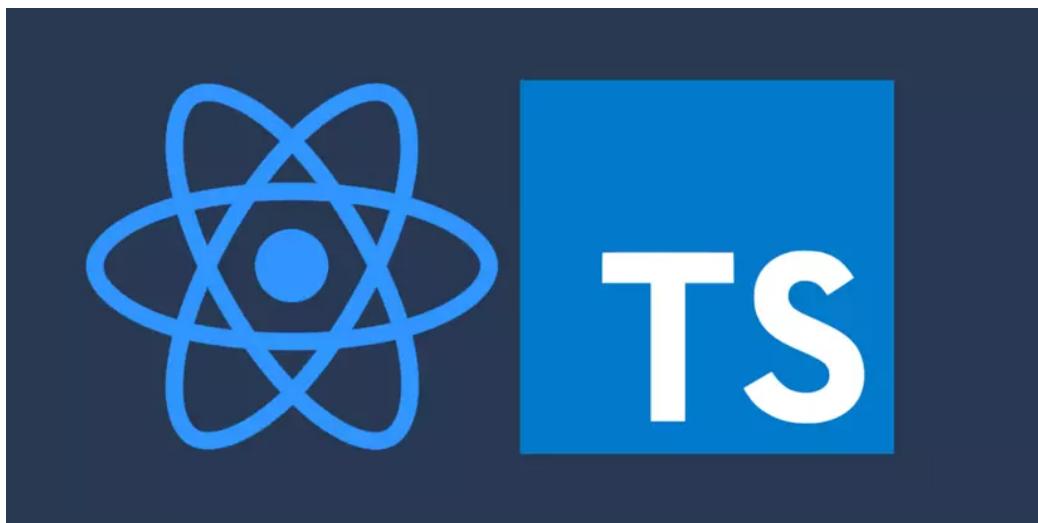
Chương 3

KIẾN TRÚC PHẦN MỀM

3.1 Mô hình ứng dụng

Frontend:

- **ReactJS:** React là một thư viện JavaScript mã nguồn mở, được phát triển bởi Facebook vào năm 2013, nhằm xây dựng giao diện người dùng cho các ứng dụng web. React cho phép các lập trình viên phát triển các thành phần giao diện (components) một cách hiệu quả, dễ bảo trì và tái sử dụng. Với React, người dùng có thể trải nghiệm giao diện mượt mà và dễ dàng tương tác với ứng dụng.



Backend:

- **Python - Django:** Django là một framework Python mạnh mẽ giúp phát triển các ứng dụng web nhanh chóng và dễ dàng hơn. Django cung

cấp các tính năng như ORM (Object Relational Mapping) để quản lý cơ sở dữ liệu, giúp việc phát triển các API RESTful trở nên đơn giản và thuận tiện. Nó cũng hỗ trợ bảo mật và quản lý người dùng, giúp dễ dàng xử lý các yêu cầu API của người dùng.



Hệ quản trị cơ sở dữ liệu:

- **MySQL:** là một hệ thống quản trị cơ sở dữ liệu mã nguồn mở (Relational Database Management System, viết tắt là RDBMS), thuộc quyền sở hữu của Oracle, được sử dụng để quản lý và lưu trữ dữ liệu. Nó sử dụng SQL (Structured Query Language) làm ngôn ngữ chính để truy vấn và thao tác với cơ sở dữ liệu. MySQL phổ biến trong các ứng dụng web, đặc biệt là các ứng dụng sử dụng kiến trúc LAMP (Linux, Apache, MySQL, PHP/Python/Perl). Các ứng dụng web lớn nhất như Facebook, Twitter, YouTube, Google, và Yahoo! đều dùng MySQL cho mục đích lưu trữ dữ liệu. Nó đã tương thích với nhiều hệ tầng máy tính quan trọng như Linux, macOS, Microsoft Windows, và Ubuntu.



Giao thức truyền thông:

- **WebSocket:** là một giao thức truyền thông giúp cho việc thiết lập kênh truyền thông hai chiều giữa máy chủ và máy khách. WebSocket hoạt động bằng cách thiết lập kết nối HTTP liên tục với máy chủ và sau đó nâng cấp nó lên kết nối websocket hai chiều bằng cách gửi Upgrade header. WebSocket được hỗ trợ trong hầu hết các trình duyệt web hiện đại và cho các trình duyệt không hỗ trợ, chúng tôi có các thư viện cung cấp dự phòng cho các kỹ thuật khác như Comet và HTTP Long Polling.



3.2 Công cụ

PyCharm:

- PyCharm là một môi trường phát triển tích hợp (Integrated Development Environment - IDE) được thiết kế chuyên biệt cho lập trình Python, do JetBrains phát triển và ra mắt lần đầu tiên vào tháng 2 năm 2010. PyCharm cung cấp các tính năng mạnh mẽ như gợi ý mã thông minh, kiểm tra lỗi thời gian thực, tích hợp công cụ quản lý phiên bản (Git, SVN), và hỗ trợ phát triển web với các framework như Django và Flask. IDE này có hai phiên bản: Community (miễn phí, mã nguồn mở) và Professional (trả phí, với nhiều tính năng nâng cao). PyCharm được sử dụng rộng rãi trong các dự án phát triển phần mềm, đặc biệt là các ứng dụng Python, và tương thích với nhiều nền tảng như Windows, macOS, và Linux.



Visual Studio Code (VS Code):

- Visual Studio Code là một trình soạn thảo mã nguồn nhẹ nhưng mạnh mẽ, hỗ trợ đa nền tảng như Windows, macOS và Linux. Với các tính năng như hoàn thành mã thông minh, tích hợp Git, và hỗ trợ nhiều tiện ích mở rộng, VS Code được sử dụng để phát triển frontend của dự án Instagram clone với ReactJS. Công cụ này giúp lập trình viên dễ dàng viết mã, gõ lỗi, và quản lý các thư viện như Ant Design, Redux, và TailwindCSS một cách hiệu quả.

Postman:

- Postman là một công cụ phổ biến được sử dụng để kiểm thử và làm việc với các API, đặc biệt là API kiểu REST. API đóng vai trò quan trọng trong việc kết nối các thành phần của ứng dụng, và Postman giúp đơn giản hóa quá trình gọi và kiểm tra API mà không cần viết mã. Công cụ này hỗ trợ tất cả các phương thức HTTP như GET, POST, PUT, PATCH, DELETE, và cho phép lưu lại lịch sử các yêu cầu (request) để sử dụng lại khi cần. Trong dự án Instagram clone, Postman được sử dụng để kiểm tra các API như đăng bài post (POST /api/post), lấy danh sách bài post (GET /api/post), và gửi tin nhắn.



MySQL Workbench:

- MySQL Workbench là một công cụ quản lý cơ sở dữ liệu MySQL, cung cấp giao diện đồ họa để phát triển, thiết kế và quản lý cơ sở dữ liệu một cách trực quan. Công cụ này cho phép dễ dàng tạo, chỉnh sửa cơ sở dữ liệu, thực hiện các thao tác như đảo ngược (reverse engineering) để tạo mô hình từ cơ sở dữ liệu hiện có, và chuyển tiếp (forward engineering) để triển khai mô hình thành cơ sở dữ liệu. Trong dự án Instagram clone, MySQL Workbench được sử dụng để thiết kế và quản lý cơ sở dữ liệu MySQL, bao gồm các bảng như users, posts, comments, và messages.

GitHub:

- GitHub là một nền tảng quản lý mã nguồn phổ biến, cho phép các lập trình viên chia sẻ, cộng tác và quản lý phiên bản mã nguồn một cách hiệu quả. Sự phát triển của GitHub bắt đầu vào ngày 19 tháng 10 năm 2007, và trang web chính thức được ra mắt vào tháng 4 năm 2008

bởi **Tom Preston-Werner**, **Chris Wanstrath**, và **PJ Hyett**. Microsoft đã mua lại GitHub vào tháng 6 năm 2018, giúp nền tảng này có thêm nhiều nguồn lực và tích hợp sâu hơn với các sản phẩm của Microsoft, như Azure và Visual Studio.

- GitHub cung cấp không chỉ một kho lưu trữ mã nguồn mà còn là một công cụ mạnh mẽ để cộng tác và chia sẻ mã nguồn giữa các lập trình viên trên toàn cầu. Git, hệ thống quản lý phiên bản mà GitHub sử dụng, cho phép các lập trình viên theo dõi và kiểm soát sự thay đổi trong mã nguồn theo thời gian, giúp họ có thể quay lại các phiên bản trước nếu cần thiết và làm việc đồng thời mà không gặp phải vấn đề xung đột dữ liệu.
- GitHub cung cấp các tính năng như **branches** (nhánh) để làm việc song song trên các tính năng hoặc sửa lỗi mà không làm ảnh hưởng đến mã nguồn chính. Người dùng có thể **fork** (tạo bản sao) các dự án từ người khác để làm việc trên đó và sau đó tạo **pull request** (yêu cầu hợp nhất) để đóng góp các thay đổi của mình vào dự án gốc.
- GitHub cũng cung cấp các công cụ hỗ trợ như **issue tracking** (theo dõi vấn đề), **project boards** (bảng dự án), **actions** (tự động hóa quy trình phát triển), và **wikis** để tài liệu hóa các dự án, giúp các nhóm lập trình viên dễ dàng hợp tác, theo dõi tiến độ, và quản lý các vấn đề phát sinh trong suốt quá trình phát triển phần mềm.
- Ngoài ra, GitHub còn có tính năng **GitHub Pages**, cho phép người dùng lưu trữ các trang web tĩnh trực tiếp từ kho GitHub của mình. GitHub còn hỗ trợ **private repositories** (kho lưu trữ riêng tư), giúp các lập trình viên hoặc tổ chức bảo vệ mã nguồn của họ khỏi việc chia sẻ công khai.
- Nói chung, GitHub là một công cụ cực kỳ quan trọng trong cộng đồng lập trình viên, giúp tăng cường sự hợp tác, bảo vệ mã nguồn và thúc đẩy quy trình phát triển phần mềm nhanh chóng và hiệu quả.

Giấy phép mã nguồn mở (Open Source License):

- Giấy phép Mã nguồn mở (Open Source License) là các giấy phép cho phép người dùng tự do truy cập, sử dụng, sửa đổi và phân phối phần mềm, miễn là họ tuân theo các điều kiện của giấy phép. Các giấy phép này giúp phần mềm được phát triển và chia sẻ rộng rãi, thúc đẩy cộng đồng đóng góp vào việc phát triển phần mềm. Một số giấy phép mã nguồn mở phổ biến bao gồm: MIT License, GPL (General Public

License), và Apache License 2.0. MIT License là một trong những giấy phép mã nguồn mở đơn giản nhất, cho phép người dùng sử dụng, sao chép, sửa đổi và phân phối phần mềm miễn phí, nhưng yêu cầu phải bao gồm bản quyền và thông báo giấy phép. GPL yêu cầu phần mềm phát hành dưới mã nguồn mở và bất kỳ phần mềm phát triển từ phần mềm này cũng phải tiếp tục là mã nguồn mở. Apache License 2.0 tương đối thoải mái nhưng yêu cầu bảo vệ các quyền sở hữu trí tuệ liên quan đến phần mềm. Các giấy phép mã nguồn mở thúc đẩy sự phát triển của phần mềm thông qua cộng đồng và giúp bảo vệ quyền lợi của các tác giả phần mềm.

Windows Subsystem for Linux (WSL):

- Windows Subsystem for Linux (WSL) là một tính năng trên Windows 10 và Windows Server 2019, cho phép người dùng chạy môi trường Linux trực tiếp trên Windows mà không cần phải cài đặt một máy ảo hoặc hệ điều hành dual-boot. WSL giúp các nhà phát triển sử dụng các công cụ và phần mềm Linux, ngay cả khi họ đang làm việc trên hệ điều hành Windows. WSL cung cấp một trải nghiệm người dùng rất tiện lợi vì nó cho phép sử dụng các công cụ dòng lệnh của Linux, như Bash, grep, sed, awk, và nhiều phần mềm khác mà thường chỉ có sẵn trên Linux. Điều này rất hữu ích đối với các nhà phát triển web và phần mềm, những người cần phải phát triển trên môi trường Linux mà không muốn chuyển sang hệ điều hành khác. WSL có hai phiên bản: WSL 1 và WSL 2. WSL 1 tạo ra một lớp tương thích giữa Windows và Linux, cho phép chạy các lệnh Linux trực tiếp trên Windows mà không cần máy ảo. WSL 2 cải thiện hiệu suất với một nhân Linux thực sự, giúp hỗ trợ đầy đủ các hệ thống file của Linux và tăng tốc độ đáng kể, đặc biệt là đối với các ứng dụng yêu cầu hệ thống file hoặc nhân Linux đầy đủ.

Amazon Web Services (AWS):

- Amazon Web Services (AWS) là nền tảng dịch vụ đám mây của Amazon, cung cấp các dịch vụ như tính toán, lưu trữ, cơ sở dữ liệu, phân tích dữ liệu và các công cụ phát triển khác. AWS giúp các công ty và lập trình viên xây dựng và vận hành các ứng dụng mà không cần phải quản lý cơ sở hạ tầng vật lý. AWS cung cấp các dịch vụ đám mây quy mô lớn, đáng tin cậy, bảo mật và chi phí hiệu quả, phục vụ nhiều loại ứng dụng từ ứng dụng web đến ứng dụng di động, từ cơ sở hạ tầng IT cho đến AI và học máy. Một số dịch vụ chính của AWS bao gồm Amazon EC2 (Elastic Compute Cloud), Amazon S3 (Simple Storage

Service), AWS Lambda, và Amazon RDS (Relational Database Service). Amazon EC2 cung cấp các máy ảo để bạn chạy ứng dụng, server và dịch vụ của mình. Amazon S3 là dịch vụ lưu trữ dữ liệu đám mây có khả năng mở rộng cao, giúp bạn lưu trữ và truy xuất dữ liệu một cách dễ dàng. AWS Lambda là dịch vụ tính toán không máy chủ, cho phép bạn chạy mã mà không cần phải quản lý máy chủ. Amazon RDS cung cấp các dịch vụ cơ sở dữ liệu như MySQL, PostgreSQL, và Oracle trên đám mây, giúp việc quản lý cơ sở dữ liệu trở nên dễ dàng và tiết kiệm chi phí. AWS có mô hình thanh toán theo mức sử dụng, giúp các doanh nghiệp chỉ trả tiền cho những dịch vụ mà họ sử dụng, từ đó tối ưu hóa chi phí và tài nguyên cho các công ty và tổ chức.



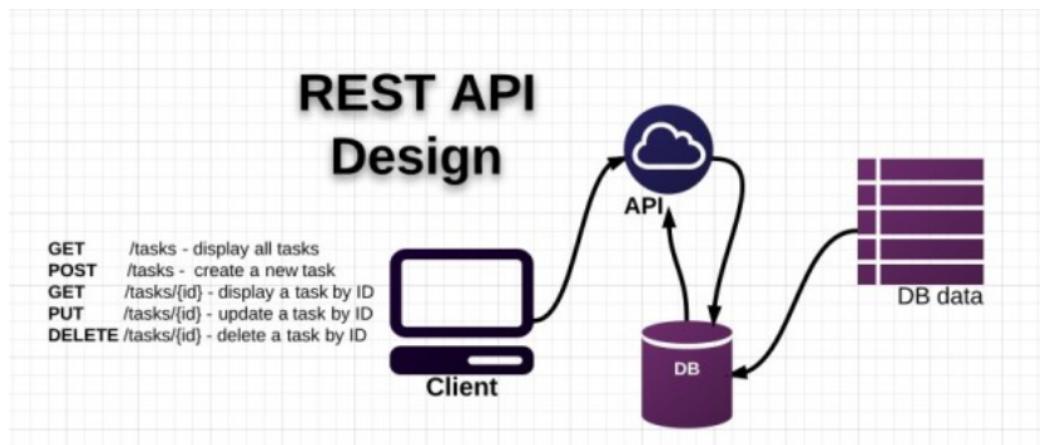
Deploy: Amazon EC2 và VPS:

- **Amazon EC2:** Amazon EC2 là dịch vụ đám mây của Amazon Web Services (AWS), cho phép người dùng thuê máy chủ ảo (instances) để triển khai và chạy ứng dụng. EC2 mang lại sự linh hoạt, khả năng mở rộng và tính sẵn sàng cao cho các ứng dụng web. Với Clone Spotify, EC2 sẽ được sử dụng để triển khai phần backend và giúp ứng dụng có thể mở rộng khi lượng người dùng tăng.
 - **VPS:** VPS là một máy chủ ảo được phân chia từ một máy chủ vật lý duy nhất. Mỗi VPS có hệ điều hành riêng biệt, giúp triển khai ứng dụng và quản lý tài nguyên như một máy chủ độc lập. Với Clone Spotify, VPS sẽ được sử dụng cho việc triển khai các dịch vụ khác ngoài backend, như các ứng dụng phụ trợ hoặc lưu trữ dữ liệu không dung đến hệ thống chính.

3.3 Kiến trúc phần mềm

RESTful API:

- **Khái niệm:** RESTful API là một tiêu chuẩn được sử dụng trong việc thiết kế API cho các ứng dụng web (thiết kế Web services) nhằm mục đích quản lý các tài nguyên (resource) một cách hiệu quả. Nó tập trung vào các tài nguyên hệ thống như tệp văn bản, hình ảnh, âm thanh, video, hoặc dữ liệu động, bao gồm các trạng thái tài nguyên được định dạng và truyền tải thông qua giao thức HTTP. RESTful API cho phép các ứng dụng giao tiếp với nhau thông qua các phương thức HTTP chuẩn, giúp đơn giản hóa việc truy cập và xử lý dữ liệu trên các nền tảng khác nhau.



Điều giải các thành phần

- **API (Application Programming Interface):** API là một tập hợp các quy tắc và cơ chế cho phép một ứng dụng hoặc thành phần tương tác với một ứng dụng hoặc thành phần khác. API đóng vai trò như một cầu nối, giúp các ứng dụng trao đổi dữ liệu với nhau một cách dễ dàng. Dữ liệu trả về từ API thường được định dạng ở các kiểu phổ biến như JSON hoặc XML, phù hợp để tích hợp vào các ứng dụng web hoặc di động.
- **REST (REpresentational State Transfer):** REST là một kiểu kiến trúc (architectural style) được sử dụng để thiết kế API, dựa trên việc chuyển đổi trạng thái biểu diễn của tài nguyên. REST tận dụng các phương thức HTTP đơn giản như GET, POST, PUT, DELETE để thực hiện các thao tác trên tài nguyên. Thay vì sử dụng một URL để xử lý thông tin phức tạp, REST gửi các yêu cầu HTTP đến một URL

cụ thể nhằm truy xuất hoặc chỉnh sửa dữ liệu, giúp giao tiếp giữa các hệ thống trở nên trực quan và hiệu quả.

- **RESTful API:** RESTful API là một tiêu chuẩn thiết kế API cho các ứng dụng web, tập trung vào việc quản lý các tài nguyên (resource) và cho phép các ứng dụng khác nhau (web, mobile, v.v.) giao tiếp với nhau. Chức năng quan trọng nhất của RESTful API là quy định cách sử dụng các phương thức HTTP (như GET để lấy dữ liệu, POST để tạo mới, PUT để cập nhật, DELETE để xóa) và cách định dạng URL để truy cập các tài nguyên. RESTful API không quy định logic mã nguồn của ứng dụng và không bị giới hạn bởi ngôn ngữ lập trình, do đó bất kỳ ngôn ngữ hoặc framework nào (như Java với Spring Boot, JavaScript với Node.js) cũng có thể được sử dụng để thiết kế một RESTful API.

Chương 4

SƠ ĐỒ ERD

4.1 Mô hình Dữ liệu

Dưới đây là các mô hình dữ liệu chính trong hệ thống:

4.1.1 Mô hình Người Dùng (NguoiDung)

Mô hình NguoiDung lưu trữ thông tin về người dùng của hệ thống. Các trường chính của entity này bao gồm:

- **nguo_i_dung_id**: Khóa chính, định danh duy nhất cho mỗi người dùng.
 - **email**: Địa chỉ email của người dùng, được sử dụng để đăng nhập.
 - **so_dien_thoai**: Số điện thoại của người dùng.

- **ten_hien_thi**: Tên hiển thị của người dùng trên hệ thống.
- **gioi_tinh**: Giới tính của người dùng, có thể là "Nam" hoặc "Nữ".
- **avatar_url**: URL của ảnh đại diện người dùng.
- **ngay_sinh**: Ngày sinh của người dùng.
- **quoc_gia**: Quốc gia của người dùng.
- **la_premium**: Trường boolean xác định xem người dùng có phải là người dùng Premium không.
- **google_id** và **facebook_id**: Các trường chứa ID của người dùng từ Google hoặc Facebook (nếu đăng nhập qua các nền tảng này).
- **ngay_tao** và **ngay_cap_nhat**: Thời gian tạo và cập nhật thông tin người dùng.
- **is_active** và **is_staff**: Các trường quản lý trạng thái hoạt động và quyền hạn của người dùng (staff).

Chức năng của mô hình này là quản lý thông tin người dùng, cung cấp cơ sở dữ liệu cho việc đăng nhập, tạo và cập nhật thông tin người dùng, cũng như phân quyền cho người dùng.

4.1.2 Mô hình Nghệ Sĩ (NgheSi)

Mô hình NgheSi lưu trữ thông tin về nghệ sĩ. Các trường chính bao gồm:

- **nghe_si_id**: Khóa chính, định danh nghệ sĩ.
- **ten_nghe_si**: Tên nghệ sĩ.
- **tieu_su**: Thông tin mô tả về nghệ sĩ.
- **anh_dai_dien**: URL của ảnh đại diện nghệ sĩ.
- **ngay_sinh**: Ngày sinh của nghệ sĩ.
- **quoc_gia**: Quốc gia của nghệ sĩ.
- **is_active**: Trạng thái hoạt động của nghệ sĩ.
- **created_at** và **updated_at**: Thời gian tạo và cập nhật thông tin nghệ sĩ.

Chức năng của mô hình này là lưu trữ và quản lý thông tin của các nghệ sĩ, giúp người dùng có thể tìm kiếm và theo dõi các nghệ sĩ yêu thích.

4.1.3 Mô hình Album

Mô hình **Album** lưu trữ thông tin về các album nhạc. Các trường chính bao gồm:

- **album_id**: Khóa chính, định danh album.
- **ten_album**: Tên album.
- **nghe_si**: Khóa ngoại liên kết đến mô hình **NgheSi**, chỉ ra nghệ sĩ sở hữu album.
- **anh_bia**: URL của ảnh bìa album.
- **ngay_phat_hanh**: Ngày phát hành album.
- **the_loai**: Thể loại của album.
- **is_active**: Trạng thái hoạt động của album.
- **created_at** và **updated_at**: Thời gian tạo và cập nhật album.

Chức năng của mô hình này là quản lý thông tin về các album mà nghệ sĩ phát hành, giúp người dùng dễ dàng tìm kiếm và nghe nhạc theo album.

4.1.4 Mô hình Bài Hát (BaiHat)

Mô hình **BaiHat** lưu trữ thông tin về các bài hát trong hệ thống. Các trường chính bao gồm:

- **bai_hat_id**: Khóa chính, định danh bài hát.
- **ten_bai_hat**: Tên bài hát.
- **nghe_si**: Khóa ngoại liên kết đến mô hình **NgheSi**, chỉ ra nghệ sĩ thể hiện bài hát.
- **album**: Khóa ngoại liên kết đến mô hình **Album**, chỉ ra album mà bài hát thuộc về.
- **the_loai**: Thể loại của bài hát.
- **file_bai_hat**: Đường dẫn đến tệp bài hát (mp3, mp4, v.v.).
- **duong_dan**: URL của bài hát, giúp người dùng phát bài hát từ hệ thống.

- **loi_bai_hat**: Lời bài hát.
- **thoi_luong**: Thời gian bài hát (tính bằng giây).
- **ngay_phat_hanh**: Ngày phát hành bài hát.

Chức năng của mô hình này là lưu trữ thông tin bài hát, bao gồm các tệp âm thanh và lời bài hát, giúp người dùng nghe và tìm kiếm bài hát.

4.1.5 Mô hình Danh Sách Phát (DanhSachPhat)

Mô hình DanhSachPhat lưu trữ thông tin về các danh sách phát nhạc của người dùng. Các trường chính bao gồm:

- **danh_sach_phat_id**: Khóa chính, định danh danh sách phát.
- **nguo_dung_id**: Khóa ngoại liên kết đến mô hình NguoiDung, chỉ ra người dùng sở hữu danh sách phát.
- **ten_danh_sach**: Tên danh sách phát.
- **mo_ta**: Mô tả về danh sách phát.
- **la Cong_khai**: Trạng thái công khai của danh sách phát.
- **ngay_tao**: Thời gian tạo danh sách phát.
- **tong_thoi_luong**: Tổng thời gian của tất cả bài hát trong danh sách phát (tính bằng giây).
- **so_thu_tu**: Thứ tự hiển thị của danh sách phát.
- **anh_danh_sach**: URL của ảnh đại diện cho danh sách phát.
- **so_nguo_theo_doi**: Số người theo dõi danh sách phát.

Chức năng của mô hình này là quản lý các danh sách phát của người dùng, giúp người dùng tạo và chia sẻ các danh sách nhạc.

4.1.6 Mô hình Thanh Toán (ThanhToan)

Mô hình ThanhToan lưu trữ thông tin về các giao dịch thanh toán của người dùng đối với các gói Premium. Các trường chính bao gồm:

- **thanh_toan_id**: Khóa chính, định danh giao dịch thanh toán.

- **nguoit dung id**: Khóa ngoại liên kết đến mô hình NguoiDung, chỉ ra người dùng thực hiện giao dịch.
- **so_tien**: Số tiền đã thanh toán.
- **loai_thanh_toan**: Loại giao dịch (ví dụ: thẻ tín dụng, PayPal).
- **ngay_thanh_toan**: Ngày thanh toán.
- **trang_thai_thanh_toan**: Trạng thái của giao dịch (thành công, thất bại).

Chức năng của mô hình này là quản lý thông tin giao dịch thanh toán của người dùng để cấp quyền Premium cho người dùng.

Chương 5

Xây dựng giao diện người dùng

5.1 SideBarTop

Mục đích:

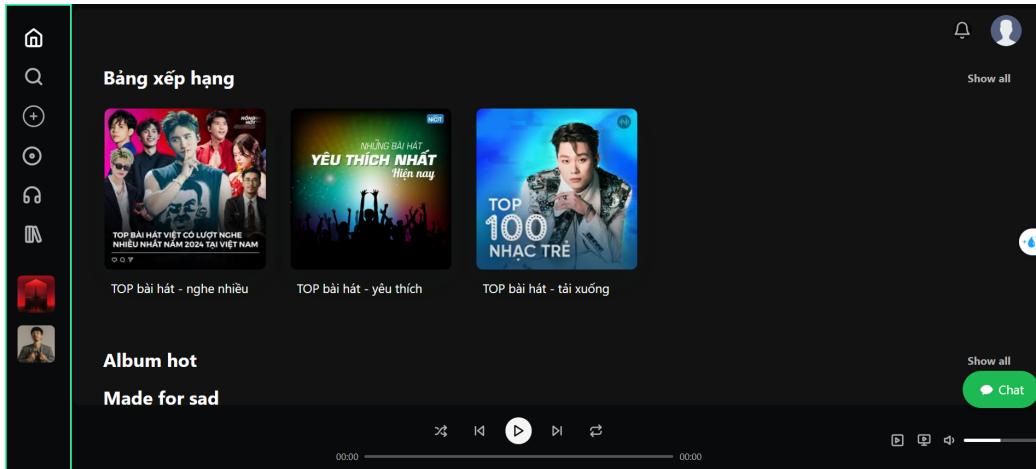
SidebarTop là thanh điều hướng chính của ứng dụng, được đặt bên trái màn hình. Thành phần này cho phép:

- Điều hướng tới các trang chính như **Home**, **Search**, **Library**.
- Tạo **danh sách phát** (playlist) mới.
- Tạo **album**.
- Tạo **playlist theo cảm xúc** như: vui, buồn, thư giãn, sôi động.

Luồng hoạt động:

- A. **Điều hướng:**

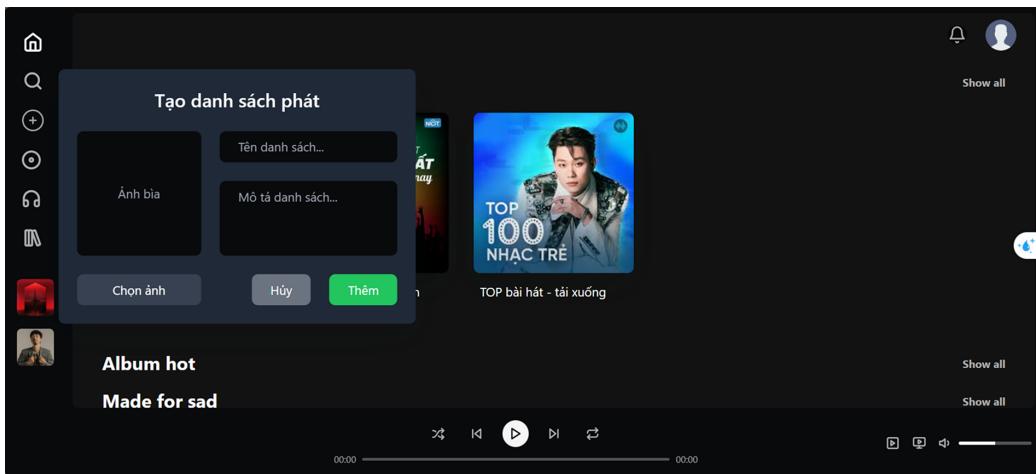
- Sử dụng các biểu tượng như *HomeIcon*, *SearchIcon*, *LibraryBig*.
 - Khi người dùng click, sử dụng component *IconLink* để điều hướng đến trang tương ứng.



Hình 5.1: Side Bar Top

- **B. Tạo danh sách phát (playlist):**

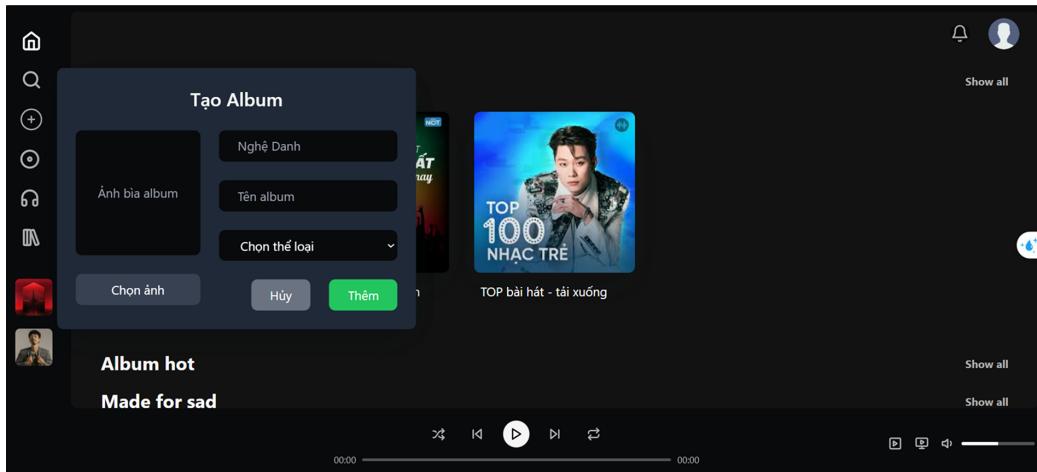
1. Nhấn vào biểu tượng + để mở biểu mẫu nhập thông tin playlist.
2. Nhập các trường: tên, mô tả, ảnh.
3. Gửi dữ liệu FormData qua API: POST /danh sach phat/them/.
4. Nếu thành công:
 - Reset form, hiển thị thông báo.
 - Gọi hàm refresh() để cập nhật giao diện.



Hình 5.2: Tạo Playlist

- **C. Tạo album:**

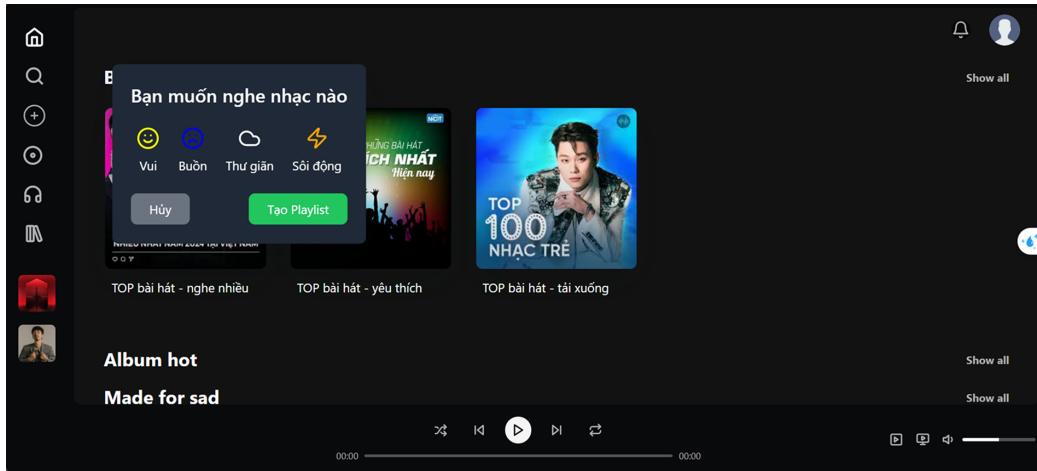
1. Nhấn vào biểu tượng **Disc** để mở form tạo album.
2. Nhập các trường: tên album, nghệ sĩ, thể loại, ảnh.
3. Gọi API lấy danh sách thể loại từ `/loaibaihat/`.
4. Gửi dữ liệu `FormData` tới API: `POST /album/create/`.



Hình 5.3: Tạo Album

- D. Tạo playlist theo cảm xúc:

1. Nhấn vào biểu tượng **Headphone** để vào biểu tượng **biểu tus**.
2. Chọn 1 trong 4 cảm xúc: vui, buồn, thư giãn, sôi động.
3. Hệ thống chọn ngẫu nhiên một ảnh phù hợp với cảm xúc.
4. Gửi API: `POST /danh sach phat/them_theo_cam_xuc/`.



Hình 5.4: Tạo Playlist theo cảm xúc

Code minh họa các chức năng trong SidebarTop

- A. Điều hướng tới các trang chính (Home, Search, Library):

```
<IconLink Icon={HomeIcon} title="Home" to="/" />
<IconLink Icon={SearchIcon} title="Search" to="/search" />
<TooltipWrapper tooltipContent="Expand Your Library">
  <LibraryBig />
</TooltipWrapper>
```

- B. Tạo danh sách phát mới (Playlist):

* Hiển thị form nhập tên, mô tả, ảnh:

```
{showInput && (
  <form onSubmit={handleSubmit}>
    <input value={playlistName} onChange={...} />
    <textarea value={playlistDescription} />
    <input type="file" onChange={handleImageChange} />
    <button onClick={handleSubmit}>Thêm</button>
  </form>
)}
```

* Gửi API POST /danh sach phat/them/:

```
const formData = new FormData();
formData.append("ten_danh_sach", playlistName);
```

```
        formData.append("mo_ta", playlistDescription);
        formData.append("anh_danh_sach", playlistImage);
        await axios.post("http://localhost:8000/danhsachphat/them/", formData);
```

– C. Tạo album:

- * Hiển thị form nhập tên, nghệ sĩ, thể loại, ảnh:

```
{showAlbumInput && (
    <form onSubmit={handleAlbumSubmit}>
        <input value={albumName} onChange={...} />
        <input value={artistName} onChange={...} />
        <select value={albumGenre} onChange={...}>
            <option value="Pop">Pop</option>
        </select>
        <input type="file" onChange={handleAlbumImageChange} />
        <button onClick={handleAlbumSubmit}>Thêm</button>
    </form>
)}
```

- * Gửi API POST /album/create/:

```
const formData = new FormData();
formData.append("ten_album", albumName);
formData.append("the_loai", albumGenre);
...
await axios.post("http://127.0.0.1:8000/album/create/", formData);
```

– D. Tạo playlist theo cảm xúc:

- * Chọn cảm xúc:

```
<Smile onClick={() => handleMoodSelect(1)} />
<Frown onClick={() => handleMoodSelect(2)} />
```

- * Gửi cảm xúc và ảnh lên API:

```
const formData = new FormData();
formData.append("cam_xuc", moodString);
formData.append("anh_danh_sach", file);
await axios.post("http://127.0.0.1:8000/danhsachphat/them_theo_cam_xuc", formData);
```

5.2 LibraryCard

Mục đích:

Component *LibraryCard* dùng để hiển thị một danh sách phát (playlist) trong thư viện của người dùng. Người dùng có thể:

- Nhấp chuột để điều hướng đến trang chi tiết playlist.

- Nhấp chuột phải để hiển thị menu xác nhận xoá playlist.

Luồng hoạt động:

1. Render card:

- Nếu `isCollapsed = true`: hiển thị thu gọn chỉ gồm ảnh và tooltip.
- Nếu `isCollapsed = false`: hiển thị đầy đủ tên và mô tả danh sách.

2. Click chuột trái:

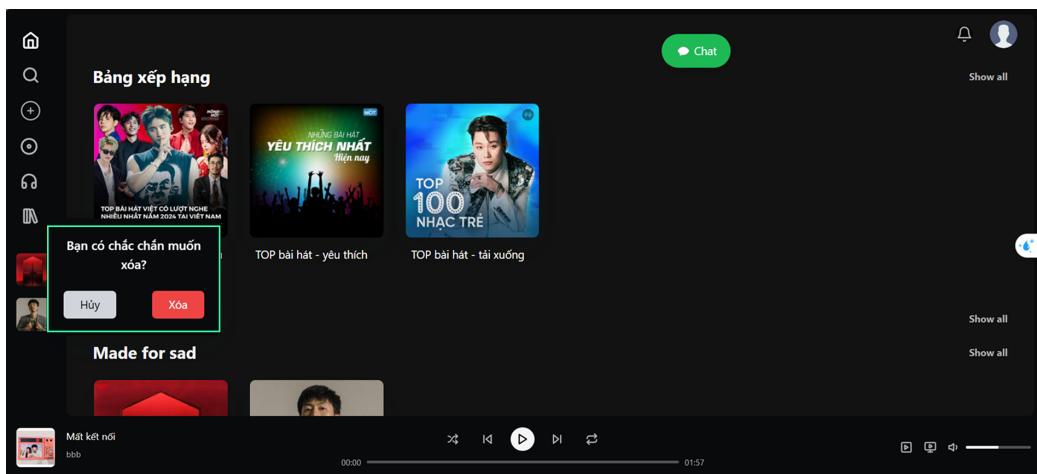
- Sử dụng `navigate()` để điều hướng đến `/playlist/?danh sach phat id=....`

3. Click chuột phải:

- Hiển thị modal xác nhận xoá playlist.
- Gán `danh_sach_phat_id` vào state `selectedId`.

4. Nhấn nút "Xoá":

- Gửi API: `DELETE /danh sach phat/xoa/{id}/`.
- Nếu thành công:
 - Đóng modal.
 - Gọi hàm `refresh()` để cập nhật danh sách playlist.



Hình 5.5: Card Library

Code minh họa:

```

// A. Điều hướng tới chi tiết playlist
<div onClick={() => navigate('/playlist/?danh_sach_phat_id=${danh_sach_phat_id}')}>

// B. Hiển thị dạng thu gọn (isCollapsed = true)
{isCollapsed && (
  <TooltipWrapper tooltipContent={ten_danh_sach}>
    <div className="p-2 hover:bg-gray-700">
      <LibraryCardImage image={anh_danh_sach} />
    </div>
  </TooltipWrapper>
)};

// C. Hiển thị dạng đầy đủ (isCollapsed = false)
 {!isCollapsed && (
  <div className="p-2 hover:bg-gray-700">
    <LibraryCardContent name={ten_danh_sach} songCount={mo_ta} />
  </div>
)};

// D. Click chuột phải để mở modal xoá
<div onContextMenu={handleRightClick}>...</div>

// E. Modal xác nhận xoá
{isModalOpen && (
  <div className="modal">
    <p>Bạn có chắc chắn muốn xoá?</p>
    <button onClick={handleDelete}>Xoá</button>
    <button onClick={() => setIsModalOpen(false)}>Huỷ</button>
  </div>
)};

// F. Gửi API xoá
const handleDelete = async () => {
  await axios.delete('http://127.0.0.1:8000/danh_sach_phat/xoa/${selectedId}');
  refresh(); // cập nhật danh sách
}

```

5.3 Search

Mục đích:

Trang *Search* cho phép người dùng tìm kiếm các nội dung như:

- Bài hát.
- Nghệ sĩ.
- Album.
- Thêm bài hát vào playlist cá nhân.

Luồng hoạt động:

1. Tìm kiếm:

- Người dùng nhập từ khóa vào ô tìm kiếm.
- Khi nhấn "Tìm kiếm", gửi request tới API: GET /common/api/search?q={query}.

2. Xử lý kết quả:

- Kết quả trả về gồm 3 danh sách:
 - **bai_hat**: danh sách bài hát.
 - **nghe_si**: danh sách nghệ sĩ.
 - **albums**: danh sách album.
- Hiển thị theo bộ lọc: Tất cả, Bài hát, Nghệ sĩ, Album.

3. Phát nhạc:

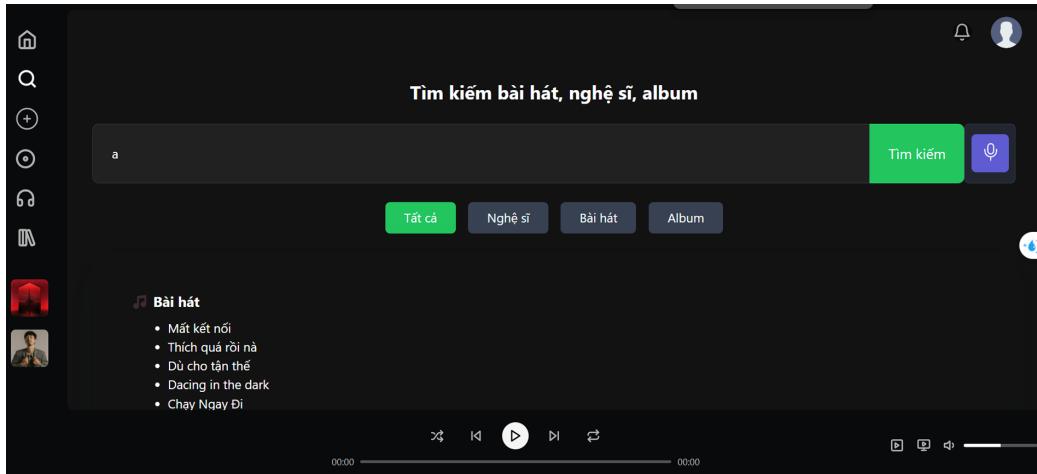
- Khi click vào tên bài hát, phát nhạc bằng trình phát audio mặc định.

4. Chuột phải vào bài hát:

- Hiển thị menu thêm vào playlist.
- Chọn playlist → gửi API: POST /danh sach phat/them-baihat.

5. Giao diện ghi âm (AudioRecorder):

- Cho phép ghi âm để tìm kiếm bài hát qua giọng nói (chức năng giả định trong phiên bản hiện tại).



Hình 5.6: Trang tìm kiếm

Code minh họa:

- A. Người dùng nhập từ khóa và nhấn "Tìm kiếm"

```
<input value={query} onChange={(e) => setQuery(e.target.value)} />
<button onClick={handleSearch}>Tìm kiếm</button>
```

- B. Gửi API tìm kiếm GET /common/api/search?q=

```
const handleSearch = async () => {
  const response = await fetch('http://127.0.0.1:8000/common/api/search');
  const data = await response.json();
  setResults(data);
};
```

- C. Hiển thị danh sách kết quả theo bộ lọc

```
{filter === "bai_hat" && results?.bai_hat.map((song) => (
  <li onClick={() => handleClickSong(song.duong_dan)}>
    {song.ten_bai_hat}
  </li>
))}
```

- D. Phát bài hát khi click tên

```
const handleClickSong = (duongDan) => {
  const audio = document.querySelector('#audio-player');
  audio.src = duongDan;
  audio.play();
};
```

- E. Chuột phải mở menu "Thêm vào playlist"

```
<li onContextMenu={(e) => handleRightClick(e, song.bai_hat_id)}>  
  {song.ten_bai_hat}  
</li>
```

- F. Hiển thị modal chọn playlist

```
{isModalOpen && (  
  <div style={{ top: modalPosition.y, left: modalPosition.x }} className="...">  
    <p>Thêm vào playlist:</p>  
    {playlists.map((p) => (  
      <div onClick={() => handleAddSongToPlaylist(selectedId, p.danh_sach_id)}>  
        {p.ten_danh_sach}  
      </div>  
    ))}  
  </div>  
)}
```

- G. Gửi API POST /dansachphat/them-baihat/

```
const handleAddSongToPlaylist = async (bai_hat_id, danh_sach_phat_id) =>  
  await fetch("http://127.0.0.1:8000/thembaihatvaodansachphat/dansachphat",  
    method: "POST",  
    headers: { "Content-Type": "application/json" },  
    body: JSON.stringify({ bai_hat_id, danh_sach_phat_id }),  
  );  
};
```

5.4 DynamicGrid

Mục đích:

Component *DynamicGrid* hiển thị danh sách các phần tử như *ItemCard* theo dạng lưới (grid), với số lượng cột tự động theo chiều rộng.

Luồng hoạt động:

1. Nhận props:

- *title, description*: tiêu đề và mô tả.
- *items*: danh sách các đối tượng có *order, danh_sach_phat_id*.
- *to*: URL đích khi click tiêu đề.
- *Component*: thành phần con để render trong grid.

2. Tính số cột:

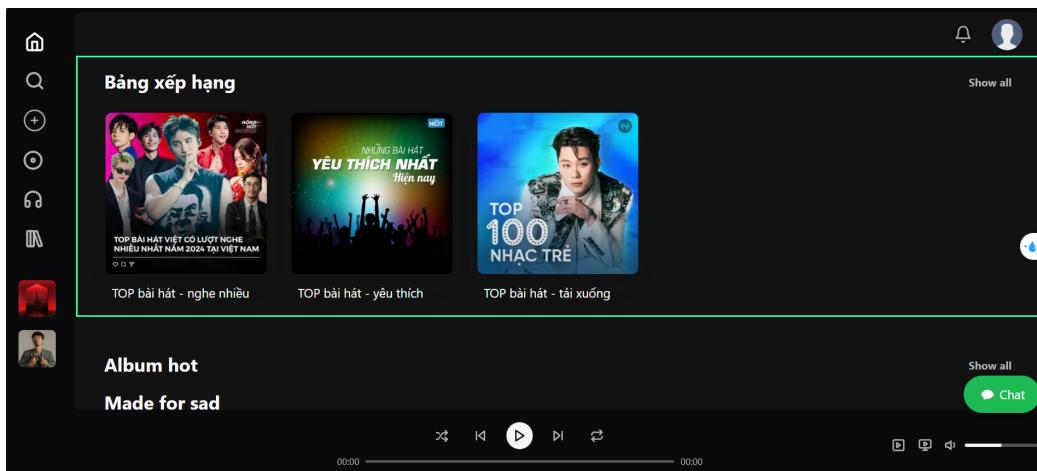
- Dựa vào chiều rộng cửa sổ (`mainWidth`) chia cho `cardSizePx = 220` để tính `columnCount`.

3. Render:

- Hiển thị tiêu đề + nút “Show all”.
- Render các phần tử đầu tiên (theo số cột), sắp xếp theo `order`.

4. Click “Show all”:

- Điều hướng tới `/showall` và truyền `items` qua state.



Hình 5.7: DynamicGrid

Code minh họa

```
// A. Nhận props: title, description, items, to, Component
function DynamicGrid({ title, description, items, to, Component }) {
  const width = useAppControllerStore((state) => state.mainWidth);
  const columnCount = Math.floor(width / 220); // B. Tính số cột

  return (
    <div>
      // C. Header hiển thị tiêu đề và nút Show all
      <div className="flex justify-between items-center px-3.5 py-2">
        <Link to={to || '/'}>
```

```

        <h2 className="text-white text-2xl font-bold hover:underline">{tit
      </Link>
      <Link to="/showall" state={{ items }}>
        <span className="text-sm text-gray-400 hover:underline">Show all<
          </Link>
        </div>

      // D. Hiển thị mô tả nếu có
      {description && <p className="px-3.5 text-sm text-gray-400">{descript

      // E. Hiển thị grid các item
      <div
        style={{ display: 'grid', gridTemplateColumns: 'repeat(${columnCount
        className="gap-4 px-3.5 mt-4"
      >
      {items
        .slice(0, columnCount)
        .sort((a, b) => a.order - b.order)
        .map((item) => (
          <Component key={item.danh_sach_phat_id} {...item} />
        )))
      </div>
    </div>
  );
}

```

5.5 ItemCard

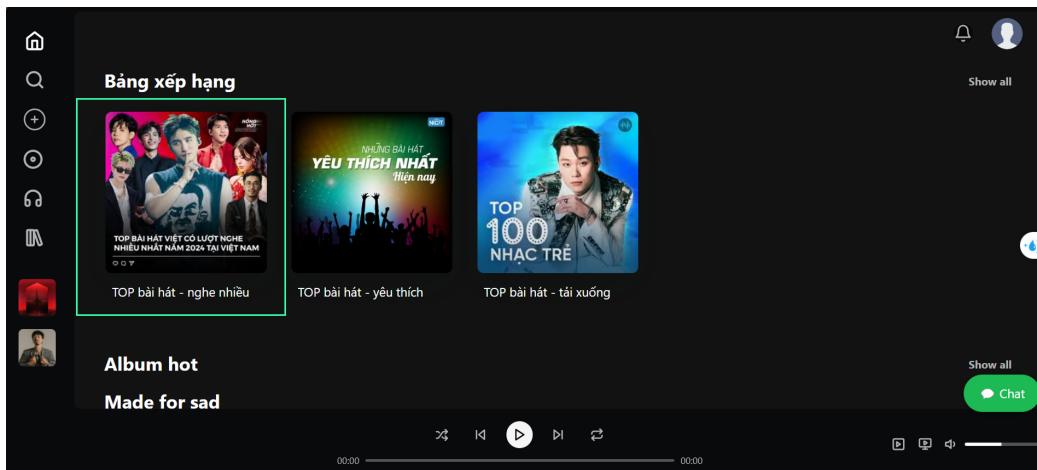
Mục đích:

Hiển thị một thẻ danh sách nhạc (playlist, album, hoặc bảng xếp hạng), cho phép điều hướng đến chi tiết tương ứng.

Luồng hoạt động:

1. Nhận props: `danh_sach_phat_id`, `album_id`, `bkh_id`, ảnh, tên, mô tả.
2. Render:
 - Hiển thị ảnh, tên danh sách (rút gọn nếu dài).
 - Hiển thị mô tả nếu có.
3. Khi click vào thẻ:

- Dùng `useNavigate()` điều hướng đến `/playlist/?...` tùy loại dữ liệu.



Hình 5.8: ItemCard

Code minh họa:

```
// A. Nhận props từ cha: ID, ảnh, tên, mô tả
function ItemCard({ danh_sach_phat_id, album_id, bxh_id, anh_danh_sach, ten_danh_sach, ... })
  const navigate = useNavigate();

  // B. Xác định đường dẫn cần điều hướng
  const openSong = () => {
    if (danh_sach_phat_id) navigate('/playlist/?danh sach phat id=${danh_sach_phat_id}');
    else if (album_id) navigate('/playlist/?album id=${album_id}');
    else if (bxh_id) navigate('/playlist/?bxh=${bxh_id}');
  };

  return (
    // C. Click vào thẻ để điều hướng
    <div onClick={openSong} className="group relative flex flex-col p-3 hover:scale-105">

      /* D. Hiển thị ảnh */
      <img src={anh_danh_sach} className="w-full aspect-square object-cover" alt="Thumbnail image of the playlist/album/bxh" />

      /* E. Tên danh sách (giới hạn chiều dài) */
      <p className="truncate text-white pt-3">{ten_danh_sach}</p>
    </div>
  );
}
```

```

    /* F. Mô tả nếu có, nếu không để trống */
    {description ? (
        <p className="text-sm text-gray-400 line-clamp-2">{description}</p>
    ) : (
        <div className="h-5"></div>
    )
);
}

```

5.6 PlaylistDetail

Mục đích:

Trang chi tiết cho playlist, album hoặc BXH, hỗ trợ phát nhạc, thêm bài hát, và hiển thị thông tin.

Luồng xử lý:

1. Xác định loại dữ liệu:

- Dựa vào URL: ?danh sach phat id, ?album id, ?bxh.
- Gọi API tương ứng: /danh sach phat/, /album/, /bxh/.

2. Lấy danh sách bài hát:

- Gọi getSongById(), lấy nghệ sĩ và album.
- Nếu không Premium:
 - Chèn quảng cáo mỗi 2 bài.
 - Bài Premium sẽ phát audio thay thế.

3. Phát nhạc:

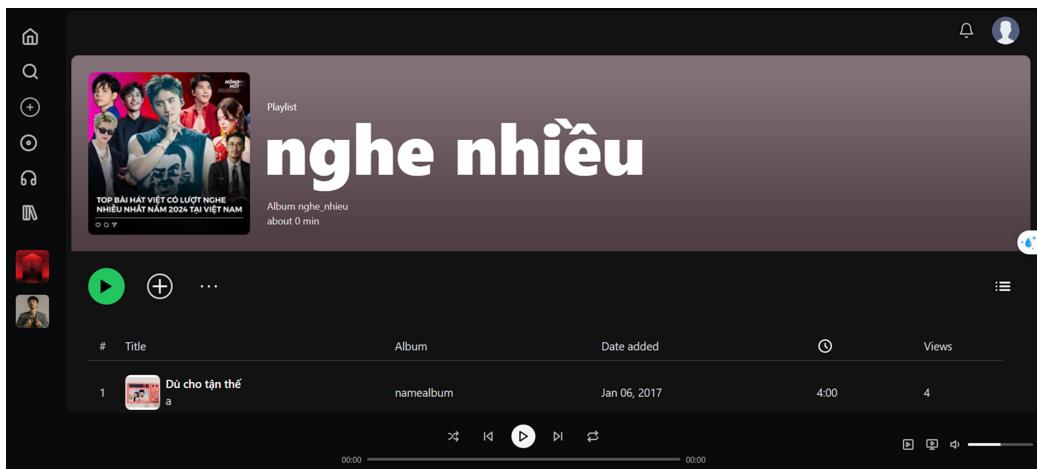
- Click 1 lần chọn bài, click 2 lần phát.
- Nếu không Premium và đã nghe quá 2 bài, phát quảng cáo 30s.

4. Giao diện bài hát:

- Hiển thị STT | Tên | Album | Ngày phát hành | Thời lượng | Lượt nghe.
- Cho phép thêm vào playlist khác bằng chuột phải.

5. Thêm bài hát (nếu album chờ duyệt):

- Hiển thị modal nhập thông tin + upload file nhạc.
- Gửi lên API dạng FormData qua addBaiHat.



Hình 5.9: Trang Playlist

Code minh họa luồng xử lý phát Playlist

- A. Xác định loại playlist từ URL:

```
const searchParams = new URLSearchParams(location.search);
const danhSachPhatId = searchParams.get('dansachsachphatid');
const albumId = searchParams.get('albumid');
const bxh = searchParams.get('bxh');
const playlistId = danhSachPhatId || albumId || bxh;
```

- B. Gọi API lấy danh sách bài hát:

```
useEffect(() => {
  async function fetchSongs() {
    let rawSongs;
    if (danhSachPhatId) rawSongs = await getSongFromPlayList(playlistId);
    else if (albumId) rawSongs = await getSongAlbum(playlistId);
    else rawSongs = await getSongBXH(playlistId);

    const details = await Promise.all(rawSongs.map(async ({ bai_hat }) =>
      const song = await getSongById(bai_hat);
```

```

        return {
          ...song,
          nghe_si: (await axios.get('/api/nghesi/${song.nghe_si}')).data,
          ten_album: (await axios.get('/album/${song.album}')).data.ten_a
        };
      }));
    }

    const isPremium = (await getUserInfo("")).la_premium;
    const finalSongs = [];
    details.forEach((s, i) => {
      if (!isPremium && s.is_premium) s.duong_dan = '/ad.mp3';
      finalSongs.push(s);
      if ((i + 1) % 2 === 0 && !isPremium)
        finalSongs.push({ the_loai: 'Ad', ten_bai_hat: 'Quảng cáo', duong_dan });
    });
    setSongs(finalSongs);
  }
  fetchSongs();
}, [playlistId]);

```

- C. Phát bài hát khi người dùng click 2 lần:

```

const handleClick = (song) => {
  if (!isPremium && songsPlayed >= 2) return playAdThen(song);
  dispatch(setCurrentSong(song));
  playAudio(song.duong_dan);
};

```

- D. Hiển thị danh sách bài hát:

```

songs.map((s, i) => (
  <div key={i} onClick={() => handleClick(s)}>
    {i+1}. {s.ten_bai_hat} - {s.nghe_si} ({s.ten_album})
  </div>
));

```

- E. Nếu là nghệ sĩ và album đang pending, cho phép thêm bài hát:

```

if (albumPending) {
  <Button onClick={() => setShowModal(true)}>Thêm bài hát</Button>
}

```

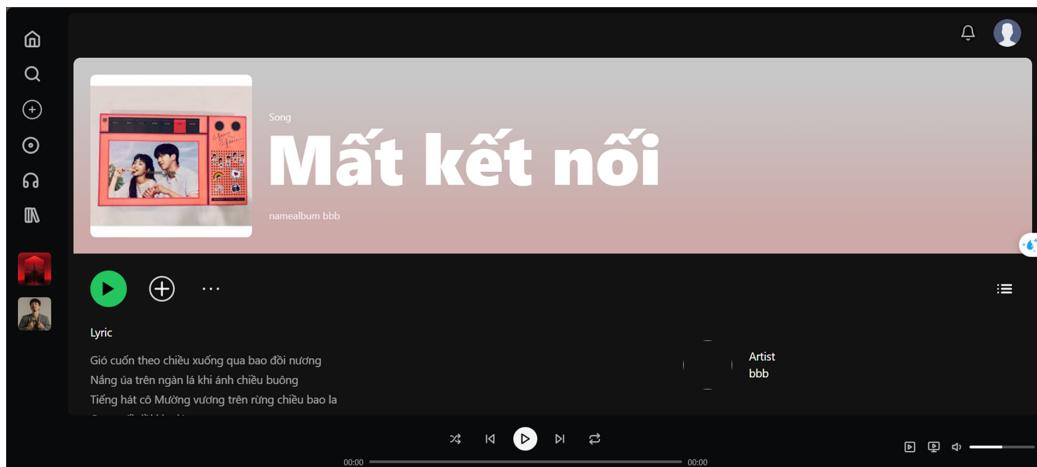
5.7 TrackDetail

Mục đích:

Hiển thị chi tiết một bài hát, gồm phát nhạc, lời bài hát (đồng bộ thời gian), và thông tin liên quan.

Luồng xử lý:

1. Lấy id từ URL /track/:id và gọi `getSongById()`.
2. Gọi API lấy album và nghệ sĩ tương ứng.
3. Phát nhạc: set `audioPlayer.src`, gọi `play()`, dispatch `setCurrentSong(song)`.
4. Lời bài hát:
 - Phân tích `loi_bai_hat` theo timestamp [hh:mm:ss].
 - Đồng bộ với `audio.currentTime`, làm nổi bật dòng đang phát.
5. Phân tích ảnh bìa tạo màu nền gradient tương ứng.



Hình 5.10: Trang Track

5.8 VideoPlayer

Mục đích:

Hiển thị và đồng bộ video bài hát đang phát, hỗ trợ tải về nếu có video.

Luồng xử lý:

1. Lấy bài hát hiện tại từ Redux: `state.player.currentSong`.

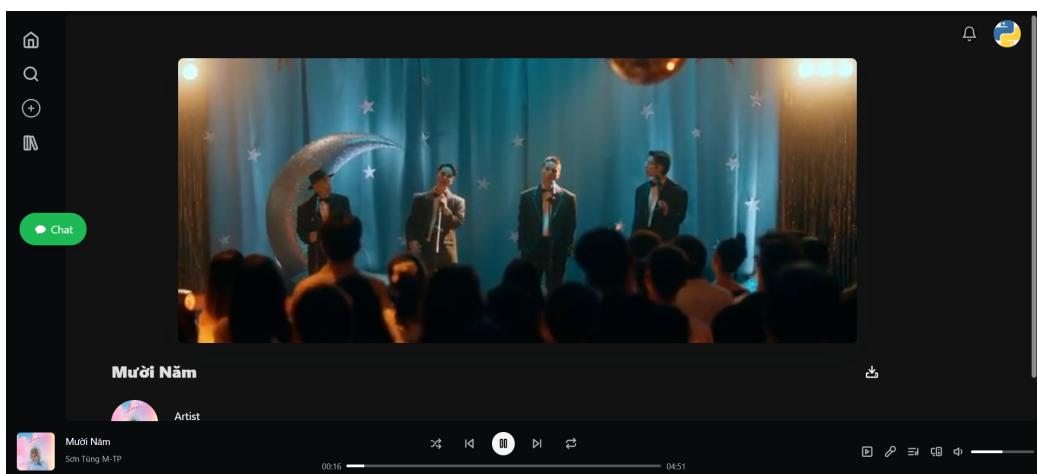
2. Phân biệt video hay mp3:

- Nếu .mp3 thì không có video.
- Nếu là video (.mp4, ...) thì hiển thị thẻ <video>.

3. Đồng bộ thời gian: mỗi 500ms cập nhật video.currentTime = audio.currentTime.

4. Khi isPaused thay đổi: pause/play video.

5. Tải video/audio: fetch videoUrl, gọi saveAs() với tên bài hát.



Hình 5.11: Trang xem video

Code minh họa luồng xử lý phát Video/Audio

- A. Lấy bài hát hiện tại và kiểm tra có video không:

```
const currentSong = useSelector((state) => state.player.currentSong);
const isVideo = currentSong?.duong_dan?.endsWith('.mp4') ||
                currentSong?.duong_dan?.endsWith('.webm');
```

- B. Đồng bộ thời gian video = audio mỗi 500ms:

```
useEffect(() => {
    const audio = document.querySelector('#audio-player');
    const sync = () => videoRef.current && audio &&
        (videoRef.current.currentTime = audio.currentTime);
    const interval = setInterval(sync, 500);
    return () => clearInterval(interval);
}, []);
```

- C. Khi tab bị ẩn rồi hiện lại, video phát từ vị trí cũ:

```
useEffect(() => {
  const resume = () => !document.hidden && videoRef.current?.play();
  document.addEventListener('visibilitychange', resume);
  return () => document.removeEventListener('visibilitychange', resume)
}, []);
```

- D. Đồng bộ trạng thái Play/Pause với video:

```
useEffect(() => {
  if (videoRef.current) {
    isPaused ? videoRef.current.pause() : videoRef.current.play();
  }
}, [isPaused]);
```

- E. Nút tải xuống video/audio khi người dùng bấm:

```
const download = async () => {
  const res = await fetch(currentSong.duong_dan);
  const blob = await res.blob();
  saveAs(blob, `${currentSong.ten_bai_hat}.mp4`);
};
```

- F. Hiển thị giao diện:

```
return (
  <div>
    {isVideo ? (
      <video ref={videoRef} src={currentSong.duong_dan} autoPlay muted />
    ) : (
      <p>Bài hát này không có video</p>
    )}
    {isVideo && <button onClick={download}>Tải xuống</button>}
  </div>
);
```

5.9 TrackDisplayer

Mục đích: TrackDisplayer là component nhỏ hiển thị thông tin bài hát đang phát tại thanh điều khiển, gồm:

- Ảnh bìa album.

- Tên bài hát.

- Tên nghệ sĩ.

Luồng xử lý:

1. Nhận các props đầu vào:

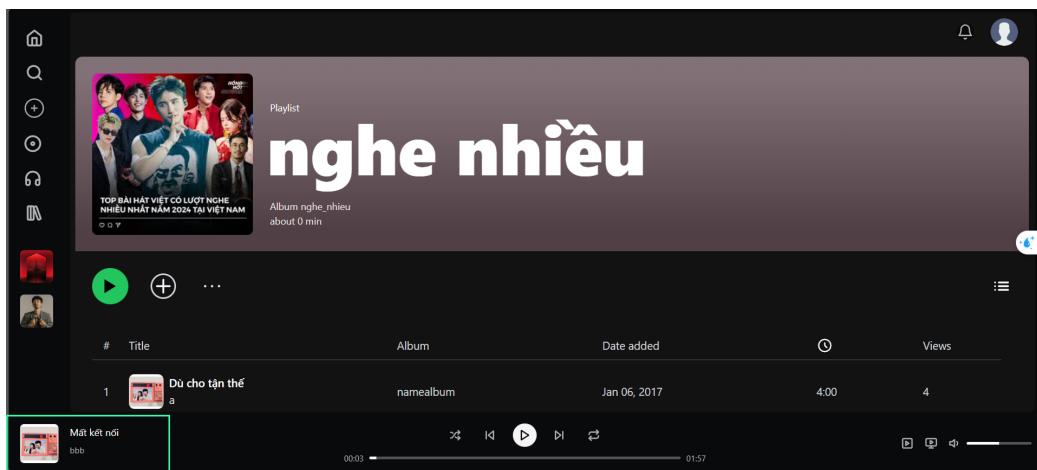
- song: thông tin bài hát hiện tại.
- artist: đối tượng nghệ sĩ (nếu cần chi tiết).
- artistName: tên nghệ sĩ dạng chuỗi.

2. Gọi API lấy thông tin album:

- Dùng `song.album_id` để gửi GET tới `/album/:id`.
- Lưu kết quả vào state `album`.

3. Render giao diện:

- Nếu có dữ liệu: hiển thị ảnh bìa, tên bài hát và tên nghệ sĩ.



Hình 5.12: Bài hát đang phát

Code minh họa hiển thị thông tin bài hát hiện tại

A. Nhận props đầu vào:

```

type Props = {
  song: { album_id: number; ten_bai_hat: string } | null;
  artistName: string;
};

```

B. Gọi API lấy thông tin album:

```

useEffect(() => {
  if (!song?.album_id) return;
  axios.get('http://127.0.0.1:8000/album/${song.album_id}/')
    .then(res => setAlbum(res.data))
    .catch(err => console.error("Lỗi lấy album:", err));
}, [song]);

```

C. Render thông tin bài hát:

```

if (!song || !album) return null;

return (
  <div className="flex items-center gap-3">
    <img src={album.anh_bia} alt="cover" className="w-14 h-14 rounded-md" />
    <div>
      <div className="text-white text-sm">{song.ten_bai_hat}</div>
      <div className="text-gray-400 text-xs">{artistName}</div>
    </div>
  </div>
);

```

5.10 ButtonGroup

Mục đích: ButtonGroup là nhóm các nút điều khiển nhạc trung tâm, bao gồm:

- Play / Pause
- Next / Previous
- Shuffle (ngẫu nhiên)

- Repeat (lặp một bài / toàn danh sách)

Luồng xử lý:

1. Play/Pause:

- Kiểm tra trạng thái `isPaused` từ Redux.
- Nếu đang pause → gọi `audio.play()` và cập nhật Redux.
- Nếu đang phát → gọi `audio.pause()` và cập nhật Redux.

2. Next / Previous:

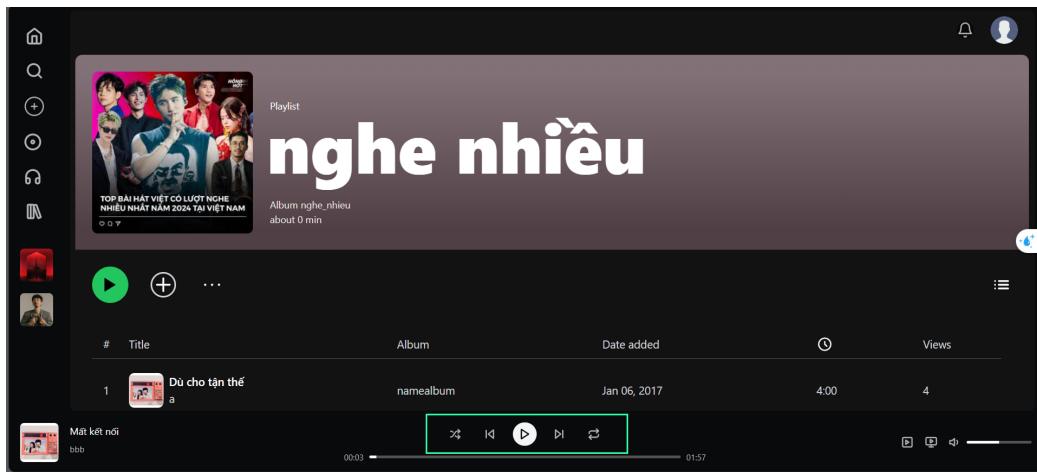
- Nếu bài hiện tại là quảng cáo (`the_loai = "Advertisement"`) → chặn next/prev.
- Nếu bật Shuffle:
 - Phát bài tiếp theo trong danh sách đã được xáo trộn (`shuffledList`).
- Nếu không Shuffle:
 - Xác định bài tiếp theo dựa vào chỉ số hiện tại `currentIndex`.

3. Repeat:

- Có 3 chế độ: `none`, `one`, `all`.
- Tùy vào trạng thái, sự kiện `audio.onended` sẽ có hành vi:
 - `one`: phát lại bài hiện tại.
 - `all`: phát bài tiếp theo, nếu hết thì quay lại đầu danh sách.
 - `none`: dừng lại khi kết thúc danh sách.

4. Shuffle:

- Khi bật:
 - Tạo danh sách mới đã được xáo trộn bằng `shuffleArray`.
 - Gán `idShuffledList` cho mỗi bài để theo dõi.
- Khi tắt:
 - Trở lại danh sách gốc `listAudio`.



Hình 5.13: Các nút điều khiển

Code minh họa ButtonGroup - Điều khiển nhạc

- A. Play/Pause:

```
const onPlay = () => {
  isPaused ? audio.play() : audio.pause();
  dispatch(togglePlayPause());
};
```

- B. Next / Previous:

```
const play = (i) => {
  const song = listAudio[i];
  if (!song || song.the_loai === "Advertisement") return;
  audio.src = song.duong_dan;
  audio.load(); audio.play();
  dispatch(setCurrentSong(song));
};
```

- C. Repeat / Shuffle:

```
<button onClick={toggleRepeat}>
  {isRepeat === 'one' ? <Repeat1Icon /> : <RepeatIcon />}
</button>
<button onClick={toggleShuffle}><ShuffleIcon /></button>
```

- D. Giao diện điều khiển:

```
<div className="flex gap-3">
  <button onClick={toggleShuffle}><ShuffleIcon /></button>
  <button onClick={() => play(currentIndex - 1)}><SkipBackIcon /></button>
  <button onClick={onPlay}>
    {isPaused ? <PlayIcon /> : <PauseIcon />}
  </button>
  <button onClick={() => play(currentIndex + 1)}><SkipForwardIcon /></button>
  <button onClick={toggleRepeat}>
    {isRepeat === 'one' ? <Repeat1Icon /> : <RepeatIcon />}
  </button>
</div>
```

5.11 ControllerSlider

Mục đích: *ControllerSlider* là thanh tiến trình bài hát, cho phép người dùng:

- Theo dõi thời gian phát hiện tại.
- Tua bài hát bằng cách kéo thanh slider.
- Vô hiệu hóa thanh trượt nếu bài hát đang phát là quảng cáo.

Luồng hoạt động:

1. Khởi tạo thanh trượt:

- Giá trị mặc định là [0].
- Giá trị max được lấy từ thời lượng bài hát `audioPlayer.duration`.

2. Cập nhật theo thời gian thực:

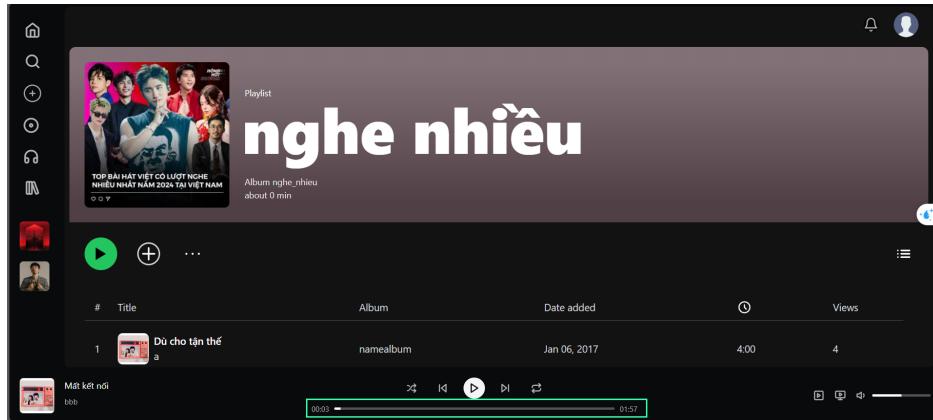
- Nếu người dùng không kéo slider (`!isDragging`) → cập nhật `currentTime` theo `audioPlayer.currentTime`.

3. Tương tác kéo thanh trượt:

- Khi bắt đầu kéo → gán `isDragging = true`.
- Khi nhả chuột (`onPointerUp`) → gán `audioPlayer.currentTime = value`.

4. Trạng thái vô hiệu hóa:

- Nếu bài hát có `the_loai = "Advertisement"` → slider sẽ bị vô hiệu hóa.



Hình 5.14: Giao diện thanh tua tiến trình bài hát

Code minh họa: Luồng xử lý thanh tiến trình ControllerSlider

- A. Nhận props và khởi tạo state:

```
const [isDragging, setIsDragging] = useState(false);
const [currentTime, setCurrentTime] = useState([0]);
const sliderRef = useRef<HTMLSpanElement>(null);
const audio = document.querySelector('#audio-player');
```

- B. Lấy dữ liệu bài hát hiện tại và thời lượng:

```
const { currentSong } = useSelector((state: RootState) => state.player);
const duration = audio?.duration ?? 0;
```

- C. Hàm xử lý tua khi người dùng dùng thả chuột:

```
const onSliderCommit = useCallback((value) => {
  if (audio && value[0] != null) {
    audio.currentTime = value[0];
    setCurrentTime(value);
  }
}, [audio]);
```

- D. Đồng bộ thời gian với audio mỗi 500ms:

```
useEffect(() => {
  if (!audio || isDragging) return;
  const interval = setInterval(() => {
    setCurrentTime([audio.currentTime]);
  }, 500);
  return () => clearInterval(interval);
}, [isDragging, audio]);
```

- E. Giao diện thanh trượt:

5.12 VolumeController

Mục đích: *VolumeController* là thành phần điều khiển âm lượng, cho phép người dùng:

- Tăng/giảm âm lượng nhạc đang phát.
- Bật hoặc tắt tiếng (mute).
- Thay đổi biểu tượng loa theo mức âm lượng.

Luồng hoạt động:

1. Khởi tạo:

- Mặc định `volume = 0.5`.
- Gán `audioPlayer.volume = 0.5`.

2. Kéo thanh âm lượng (Slider):

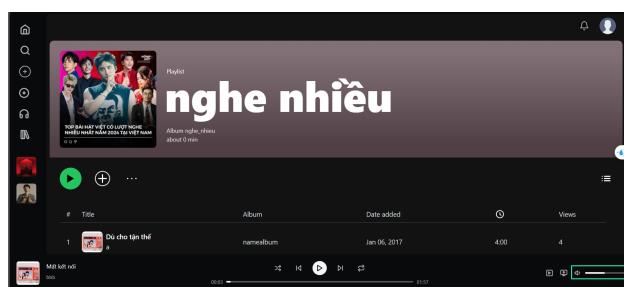
- Cập nhật giá trị `volume = value[0] / 100`.
- Nếu `volume = 0` → tự động chuyển biểu tượng sang mute.

3. Click biểu tượng loa (onMuteClick):

- Toggle trạng thái `isMuted`.
- Gán lại `audioPlayer.muted = true/false`.

4. Tự động chọn biểu tượng phù hợp:

- `VolumeIcon` nếu đang mute hoặc `volume = 0`.
- `VolumeIcon` nếu `volume < 0.3`.
- `Volume1Icon` nếu `volume < 0.6`.
- `Volume2Icon` nếu `volume ≥ 0.6`.



Hình 5.15: Thanh điều chỉnh âm lượng và biểu tượng loa

Code minh họa luồng xử lý điều chỉnh âm lượng

- A. Khởi tạo:

```
const [volume, setVolume] = useState(0.5)
const [isMuted, setIsMuted] = useState(false)
const audio = document.querySelector('#audio-player')
```

- B. Khi kéo thanh âm lượng (Slider):

```
const onChange = (value) => {
  const vol = value[0] / 100
  setVolume(vol)
  audio.volume = vol
  setIsMuted(vol === 0)
}
```

- C. Khi nhấn icon loa (mute/unmute):

```
const onMuteClick = () => {
  setIsMuted(!isMuted)
  audio.muted = !isMuted
}
```

- D. Tự chọn icon phù hợp:

```
const Icon = isMuted || volume === 0
  ? VolumeXIcon
  : volume < 0.3 ? VolumeIcon
  : volume < 0.6 ? Volume1Icon
  : Volume2Icon
```

- E. Giao diện:

```
<ControlButton Icon={Icon} onClick={onMuteClick} />
<Slider value={[volume * 100]} onChange={onChange} />
```

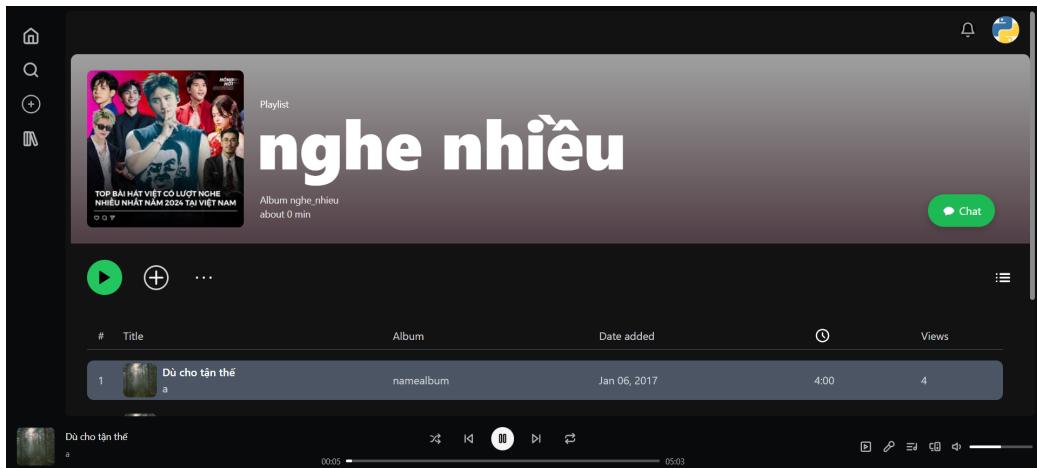
Chương 6

BÁO CÁO KẾT QUẢ

Trong quá trình phát triển phần mềm **Spotify Clone**, nhóm đã hiện thực các tính năng chính như sau:

6.1 Chức năng phát nhạc

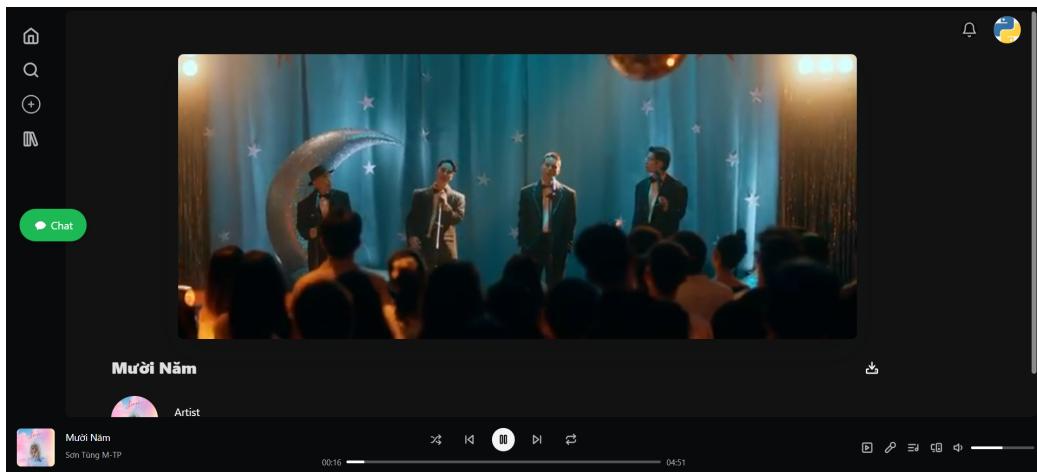
Người dùng có thể chọn và phát các bài hát từ danh sách bài hát có sẵn. Tính năng này hỗ trợ phát nhạc liên tục, phát ngẫu nhiên và lặp lại bài hát.



Hình 6.1: Giao diện chức năng phát nhạc

6.2 Chức năng phát video âm nhạc

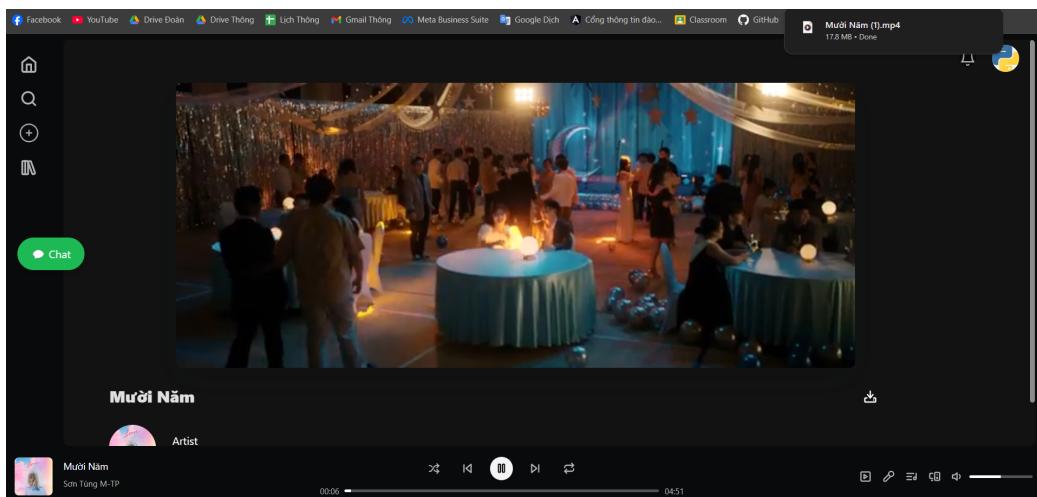
Người dùng có thể xem các video âm nhạc và có điều khiển phát lại.



Hình 6.2: Giao diện chức năng phát video âm nhạc

6.3 Chức năng tải video âm nhạc

Hệ thống cho phép người dùng tải video âm nhạc về thiết bị cá nhân thông qua nút tải xuống trong giao diện phát video.

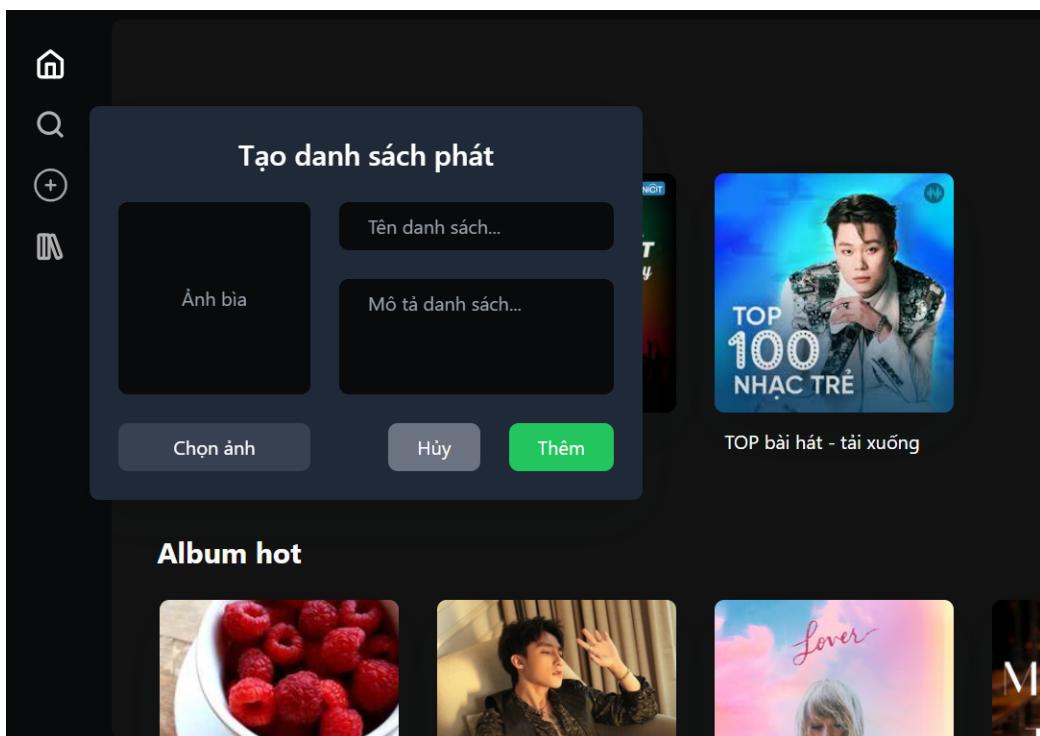


Hình 6.3: Chức năng tải video âm nhạc

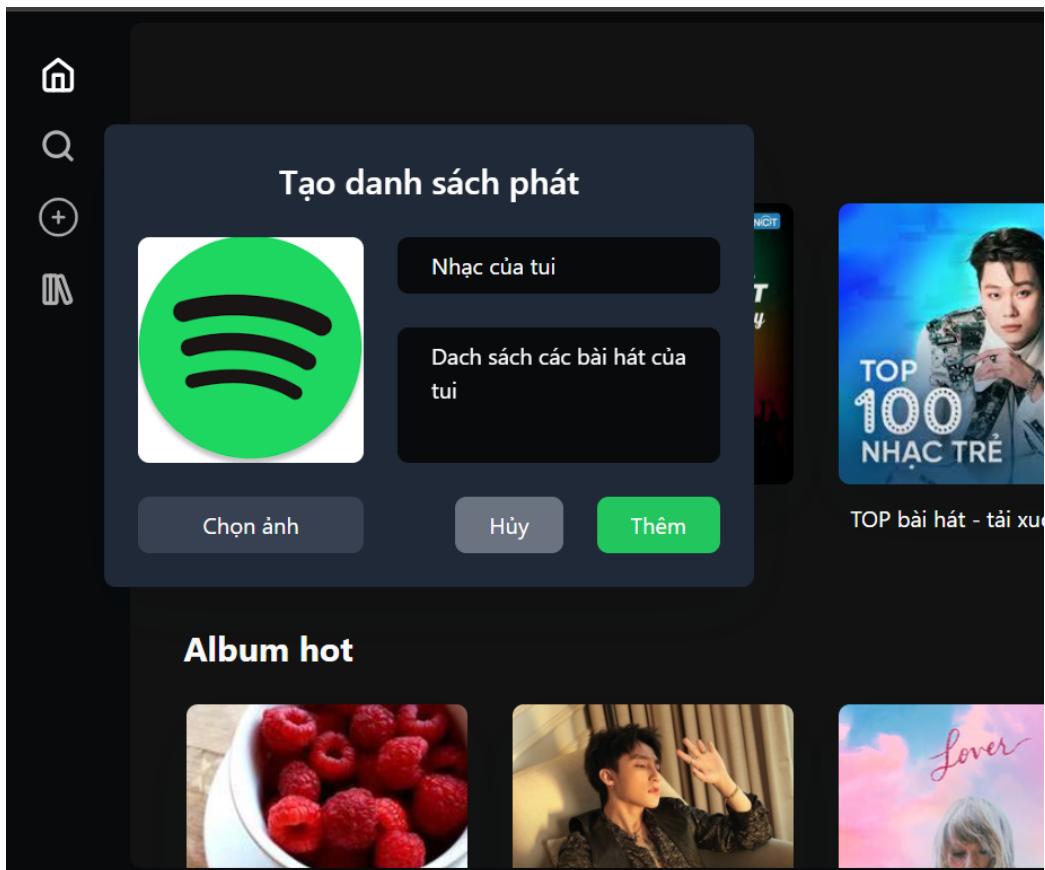
6.4 Chức năng tạo album và đánh dấu bài hát yêu thích

Người dùng có thể tạo album riêng để lưu trữ các bài hát yêu thích và dễ dàng truy cập lại. Ngoài ra, có thể đánh dấu các bài hát là “Yêu thích” để sắp xếp và lọc nhanh.

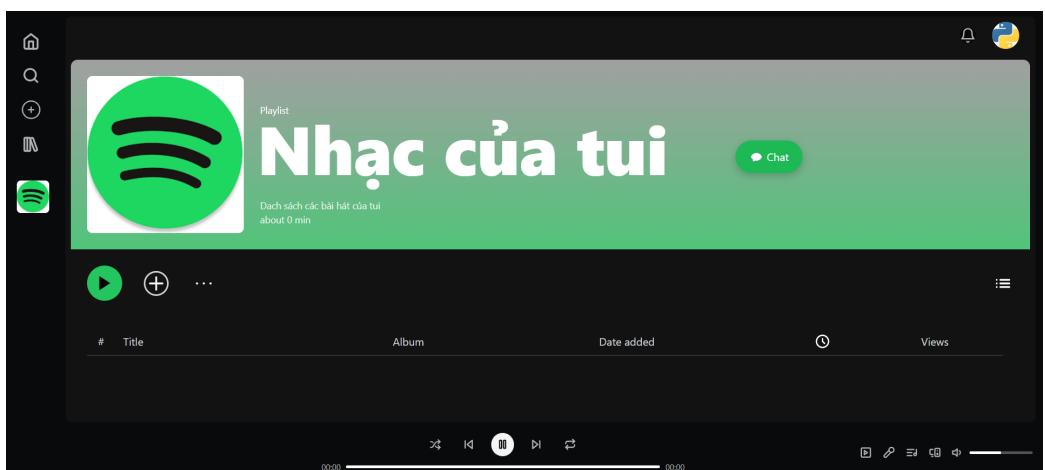
6.4.1 Chức năng tạo album



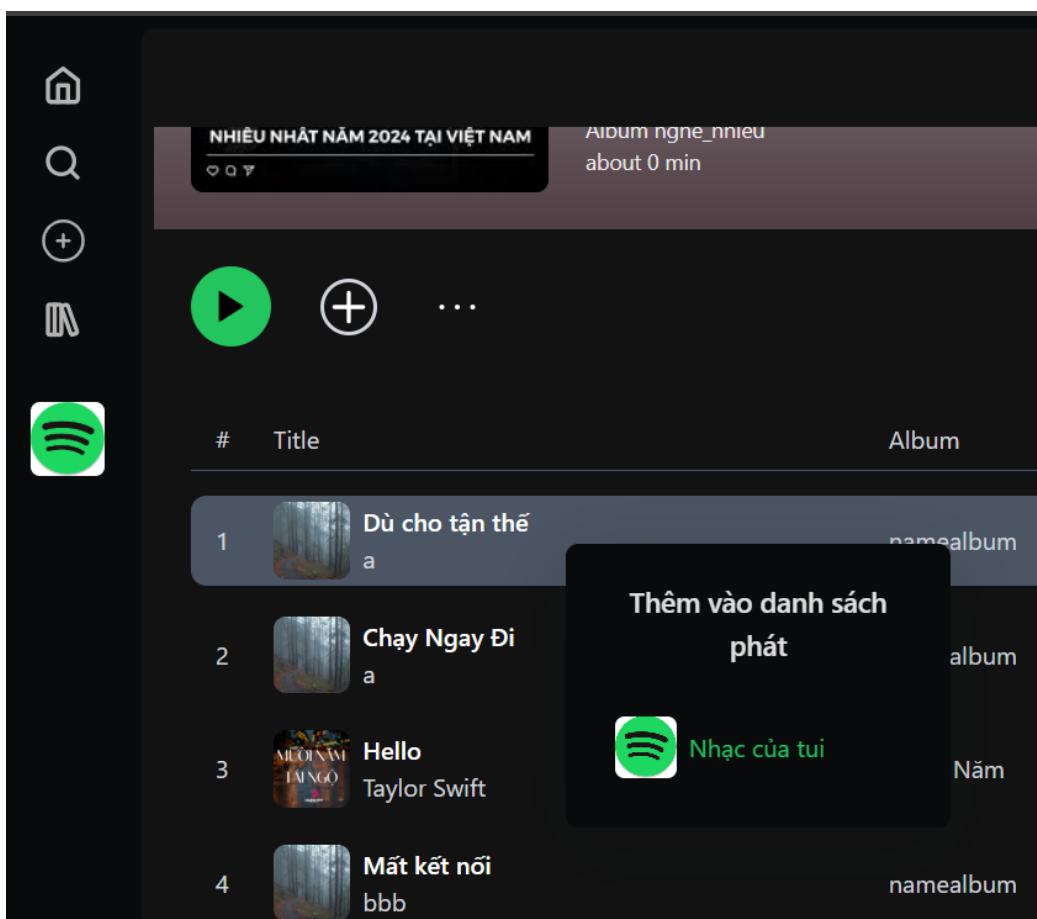
Hình 6.4: Giao diện hiện biểu mẫu tạo album



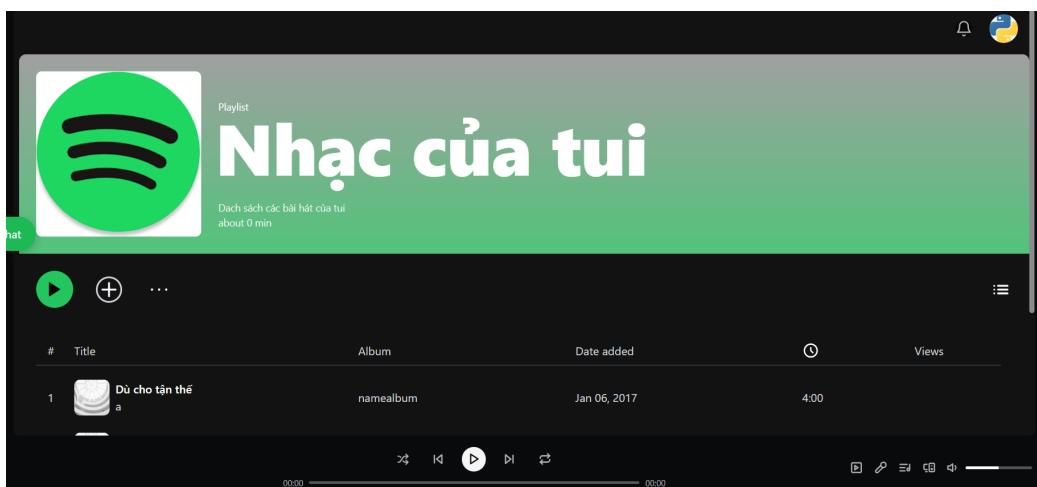
Hình 6.5: Giao diện điền thông tin tạo album



Hình 6.6: Giao diện tạo album thành công



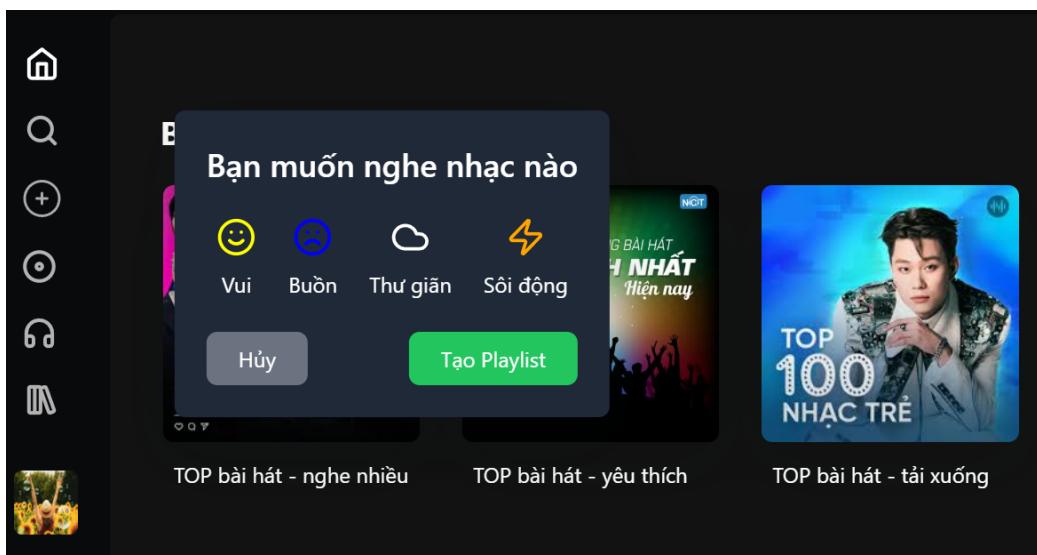
Hình 6.7: Giao diện thêm bài hát vào album



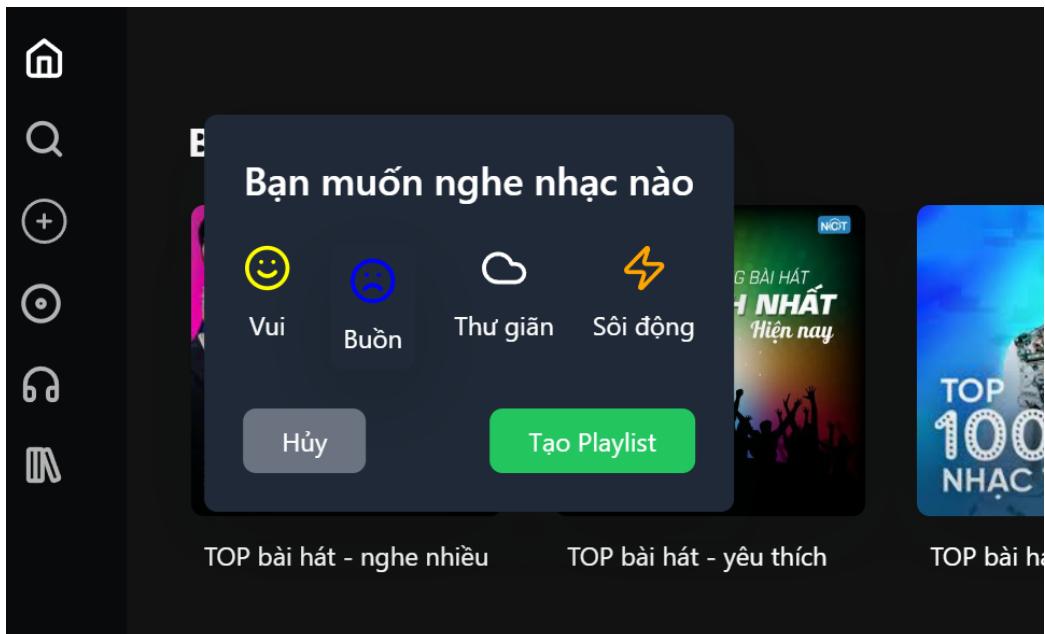
Hình 6.8: Giao diện album đã bài hát thành công

6.4.2 Chức năng đánh dấu bài hát yêu thích

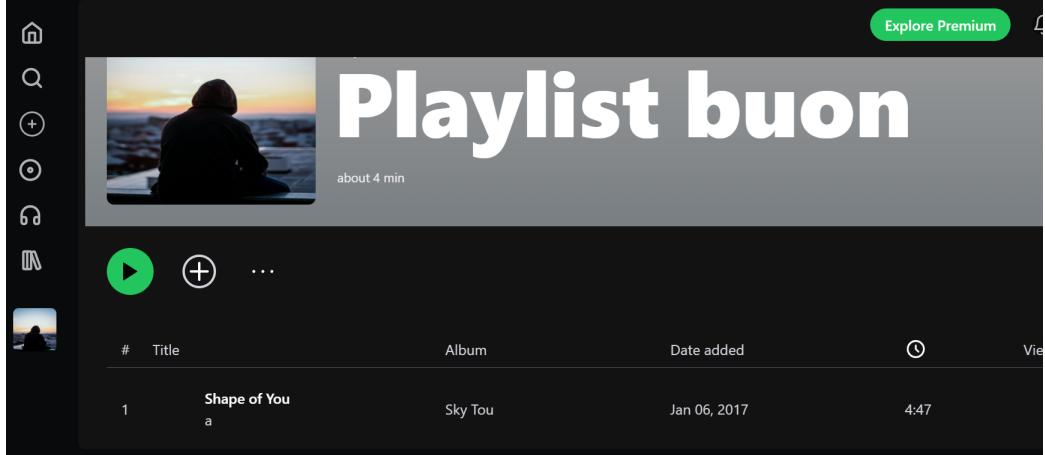
6.5 Chức năng tạo playlist theo cảm xúc



Hình 6.9: Giao diện hiện biểu mẫu tạo playlist theo cảm xúc



Hình 6.10: Giao diện chọn cảm xúc



Hình 6.11: Giao diện tạo playlist theo cảm xúc thành công

6.6 Quản trị

6.6.1 Quản lý người dùng

- Hiển thị danh sách người dùng

STT	Ảnh đại diện	Tên hiển thị	Email	Số điện thoại	Trạng thái	Vai trò	Hành động
12		Lý Thị K	user10@example.com	0901234570	<button>Hoạt động</button>	<button>Người dùng</button>	<button>Xem chi tiết</button> <button>Khóa</button> <button>Cấp quyền Admin</button>
13		Nguyễn Đình Thông	nguyendinhthongd@gmail.com	0865824502	<button>Hoạt động</button>	<button>Quản trị viên</button>	<button>Xem chi tiết</button> <button>Khóa</button>
14		NguyenDinhThong	nguyenthong2446666@gmail.com		<button>Hoạt động</button>	<button>Người dùng</button>	<button>Xem chi tiết</button> <button>Khóa</button> <button>Cấp quyền Admin</button>

Hình 6.12: Hiển thị danh sách người dùng

- Xem chi tiết thông tin người dùng

Ảnh đại diện	
ID người dùng	14
Tên hiển thị	Nguyễn Đình Thông
Email	nguyendinhthongd@gmail.com
Số điện thoại	0865824502
Trạng thái	<button>Hoạt động</button>
Vai trò	<button>Quản trị viên</button>
Ngày tạo tài khoản	8/4/2025

Hình 6.13: Xem chi tiết thông tin người dùng

- Khóa tài khoản người dùng

Quản lý người dùng							
STT	Ảnh đại diện	Tên hiển thị	Email	Số điện thoại	Trạng thái	Hành động	
11		Vũ Văn I	user9@example.com	0901234569	Hoạt động	Người dùng	Khóa Cấp quyền Admin
12		Lý Thị K	user10@example.com	0901234570	Hoạt động	Người dùng	Xem chi tiết Khóa Cấp quyền Admin

Hình 6.14: Khóa tài khoản người dùng

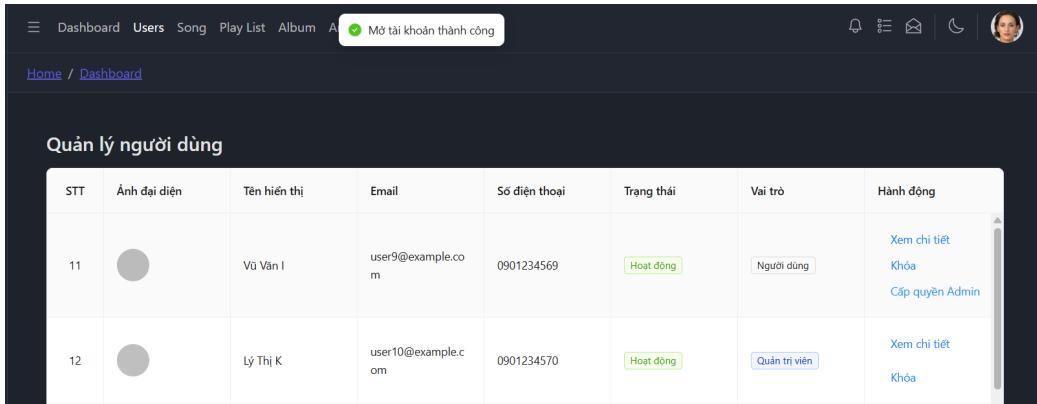
Quản lý người dùng							
STT	Ảnh đại diện	Tên hiển thị	Email	Số điện thoại	Trạng thái	Vai trò	Hành động
11		Vũ Văn I	user9@example.com	0901234569	Đã khóa	Người dùng	Xem chi tiết Mở khóa Cấp quyền Admin
12		Lý Thị K	user10@example.com	0901234570	Hoạt động	Người dùng	Xem chi tiết Khóa Cấp quyền Admin

Hình 6.15: Khóa tài khoản người dùng thành công

- Mở khóa tài khoản người dùng

Quản lý người dùng							
STT	Ảnh đại diện	Tên hiển thị	Email	Số điện thoại	Trạng thái	Hành động	
11		Vũ Văn I	user9@example.com	0901234569	Đã khóa	Người dùng	Mở khóa Cấp quyền Admin
12		Lý Thị K	user10@example.com	0901234570	Hoạt động	Quản trị viên	Xem chi tiết Khóa

Hình 6.16: Mở khóa tài khoản người dùng

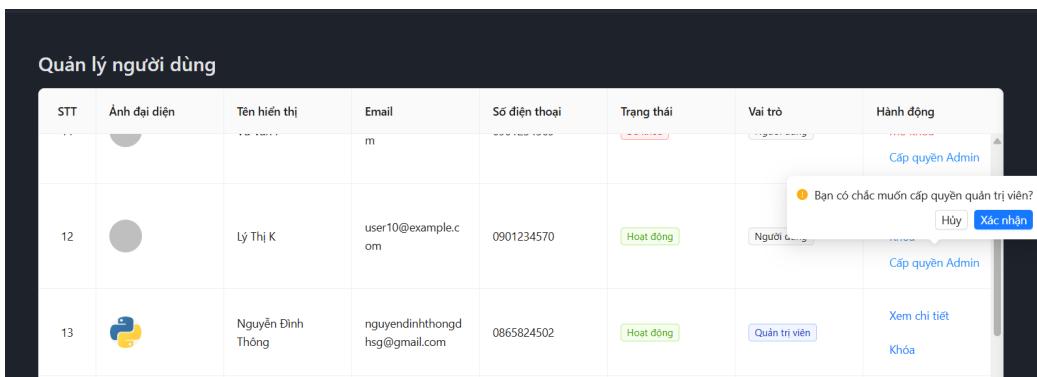


The screenshot shows a user management interface with a dark header bar. The header includes navigation links like Dashboard, Users, Song, Play List, Album, and a search bar with the message "Mở tài khoản thành công" (Account opened successfully). Below the header is a breadcrumb trail: Home / Dashboard. The main section is titled "Quản lý người dùng" (User Management) and contains a table with columns: STT, Ánh đại diện (Avatar), Tên hiển thị (Display Name), Email, Số điện thoại (Phone Number), Trạng thái (Status), Vai trò (Role), and Hành động (Actions). Two rows of data are shown:

STT	Ánh đại diện	Tên hiển thị	Email	Số điện thoại	Trạng thái	Vai trò	Hành động
11		Vũ Văn I	user9@example.com	0901234569	Hoạt động	Người dùng	Xem chi tiết Khóa Cấp quyền Admin
12		Lý Thị K	user10@example.com	0901234570	Hoạt động	Quản trị viên	Xem chi tiết Khóa

Hình 6.17: Mở khóa tài khoản người dùng thành công

- Cấp quyền quản trị cho tài khoản



This screenshot shows the continuation of the user management interface from Figure 6.17. In the "Hành động" column for user ID 13, the "Cấp quyền Admin" button is being clicked. A confirmation dialog box appears asking "Bạn có chắc muốn cấp quyền quản trị viên?" (Are you sure you want to grant administrator rights?). The dialog has two buttons: "Hủy" (Cancel) and "Xác nhận" (Confirm). The user's cursor is hovering over the "Xác nhận" button.

STT	Ánh đại diện	Tên hiển thị	Email	Số điện thoại	Trạng thái	Vai trò	Hành động
11		Vũ Văn I	user9@example.com	0901234569	Hoạt động	Người dùng	Cấp quyền Admin
12		Lý Thị K	user10@example.com	0901234570	Hoạt động	Người dùng	Cấp quyền Admin
13		Nguyễn Đinh Thông	nguyendinhthongd@gmail.com	0865824502	Hoạt động	Quản trị viên	Xem chi tiết Khóa

Hình 6.18: Cấp quyền quản trị cho tài khoản

The screenshot shows a table with columns: STT, Ánh đại diện, Tên hiển thị, Email, Số điện thoại, Trạng thái, Vai trò, and Hành động. Two users are listed:

STT	Ánh đại diện	Tên hiển thị	Email	Số điện thoại	Trạng thái	Vai trò	Hành động
12		Lý Thị K	user10@example.com	0901234570	Hoạt động	Quản trị viên	Xem chi tiết Khóa
13		Nguyễn Đình Thông	nguyendinhthongd.hsg@gmail.com	0865824502	Hoạt động	Quản trị viên	Xem chi tiết Khóa

Hình 6.19: Cấp quyền quản trị cho tài khoản thành công

6.6.2 Quản lý bài hát

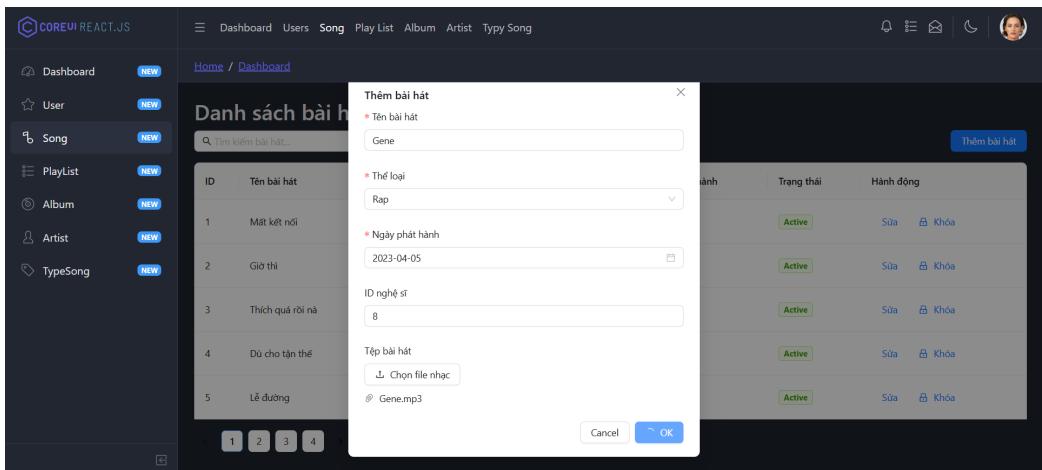
- Danh sách bài hát

The screenshot shows a table with columns: ID, Tên bài hát, Thể loại, Thời lượng, ID nghệ sĩ, Ngày phát hành, Trạng thái, and Hành động. Five songs are listed:

ID	Tên bài hát	Thể loại	Thời lượng	ID nghệ sĩ	Ngày phát hành	Trạng thái	Hành động
1	Mắt két nỗi	Pop	4 phút	2	6/1/2017	Active	Sửa Khóa
2	Giờ thi	Pop	4 phút	2	6/1/2017	Active	Sửa Khóa
3	Thích quá rồi nà	Pop	4 phút	1	6/1/2017	Active	Sửa Khóa
4	Dù cho tận thế	Pop	4 phút	1	6/1/2017	Active	Sửa Khóa
5	Lẽ đường	Pop	4 phút	1	6/1/2017	Active	Sửa Khóa

Hình 6.20: Danh sách bài hát

- Thêm bài hát mới

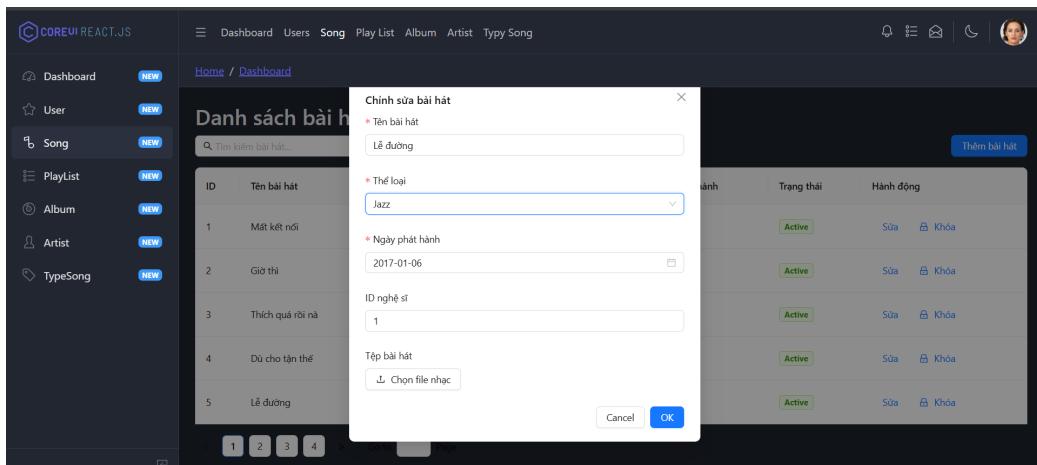


Hình 6.21: Thêm bài hát mới

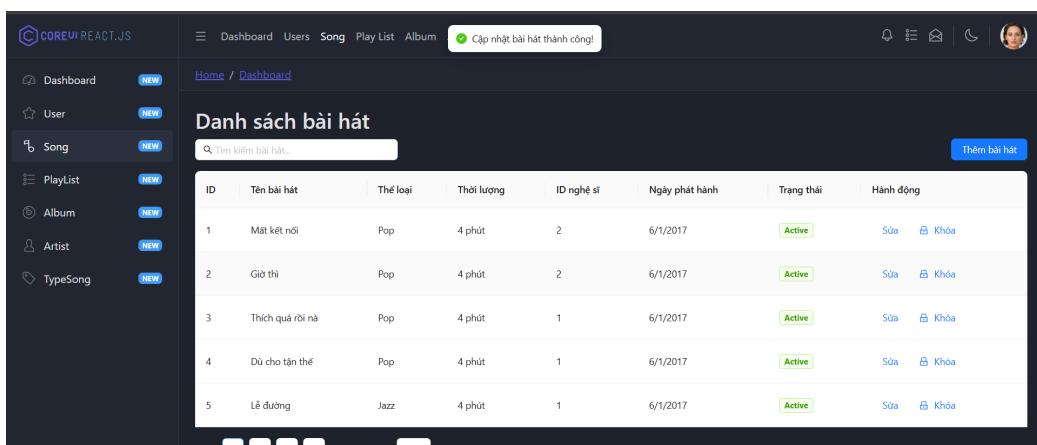
ID	Tên bài hát	Thể loại	Thời lượng	ID nghệ sĩ	Ngày phát hành	Trạng thái	Hành động
16	Uptown Funk	Funk-Pop	4 phút 30 giây	10	10/11/2014	Active	Sửa Khóa
17	Gene	Rap	3 phút 48 giây	8	5/4/2023	Active	Sửa Khóa

Hình 6.22: Thêm bài hát mới thành công

- Sửa bài hát



Hình 6.23: Sửa bài hát



Hình 6.24: Sửa bài hát thành công

- Khóa bài hát

Danh sách bài hát

ID	Tên bài hát	Thể loại	Thời lượng	ID nghệ sĩ	Ngày phát hành	Trạng thái	Hành động
1	Mắt kết nối	Pop	4 phút	2	6/1/2017	Active	Sửa Khóa
2	Giờ thì	Pop	4 phút	2	6/1/2017	Active	Sửa Khóa
3	Thích quá rồi nà	Pop	4 phút	1	6/1/2017	Active	Sửa Khóa

Bạn có chắc muốn khóa bài hát này?

Hình 6.25: Khóa bài hát

COREUI REACT.JS

Danh sách bài hát

ID	Tên bài hát	Thể loại	Thời lượng	ID nghệ sĩ	Ngày phát hành	Trạng thái	Hành động
1	Mắt kết nối	Pop	4 phút	2	6/1/2017	Inactive	Sửa Mở khóa
2	Giờ thì	Pop	4 phút	2	6/1/2017	Active	Sửa Khóa
3	Thích quá rồi nà	Pop	4 phút	1	6/1/2017	Active	Sửa Khóa

Hình 6.26: Khóa bài hát thành công

- Mở khóa bài hát

Danh sách bài hát

ID	Tên bài hát	Thể loại	Thời lượng	ID nghệ sĩ	Ngày phát hành	Trạng thái	Hành động
1	Mắt kết nối	Pop	4 phút	2	6/1/2017	Active	Sửa Mở khóa
2	Giờ thì	Pop	4 phút	2	6/1/2017	Active	Sửa Khóa
3	Thích quá rồi nà	Pop	4 phút	1	6/1/2017	Active	Sửa Khóa
4	Dù cho tận thế	Pop	4 phút	1	6/1/2017	Active	Sửa Khóa
5	Lẽ đường	Jazz	4 phút	1	6/1/2017	Active	Sửa Khóa

Hình 6.27: Mở khóa bài hát

The screenshot shows a dark-themed application interface. At the top, there's a navigation bar with links like Dashboard, Users, Song, Play List, Album, Artist, Type Song, and a button for updating status. A success message 'Cập nhật trạng thái thành công!' (Status updated successfully!) is displayed. Below the navigation is a breadcrumb trail 'Home / Dashboard'. The main title is 'Danh sách bài hát' (Song list). There's a search bar with placeholder 'Tim kiếm bài hát...' and a blue 'Thêm bài hát' (Add song) button. The table below has columns: ID, Tên bài hát, Thể loại, Thời lượng, ID nghệ sĩ, Ngày phát hành, Trạng thái, and Hành động. The data rows are:

ID	Tên bài hát	Thể loại	Thời lượng	ID nghệ sĩ	Ngày phát hành	Trạng thái	Hành động
1	Mất kết nối	Pop	4 phút	2	6/1/2017	Active	Sửa Khóa
2	Giờ thì	Pop	4 phút	2	6/1/2017	Active	Sửa Khóa
3	Thích quá rồi nà	Pop	4 phút	1	6/1/2017	Active	Sửa Khóa
4	Dù cho tận thế	Pop	4 phút	1	6/1/2017	Active	Sửa Khóa
5	Lẽ đường	Jazz	4 phút	1	6/1/2017	Active	Sửa Khóa

Hình 6.28: Mở khóa bài hát thành công

- Tìm kiếm bài hát

This screenshot shows the same application interface as the previous one, but with a search query 'Gi' entered into the search bar. The search results table shows one row for the song 'Giờ thì'.

ID	Tên bài hát	Thể loại	Thời lượng	ID nghệ sĩ	Ngày phát hành	Trạng thái	Hành động
2	Giờ thì	Pop	4 phút	2	6/1/2017	Active	Sửa Khóa

Hình 6.29: Tìm kiếm bài hát

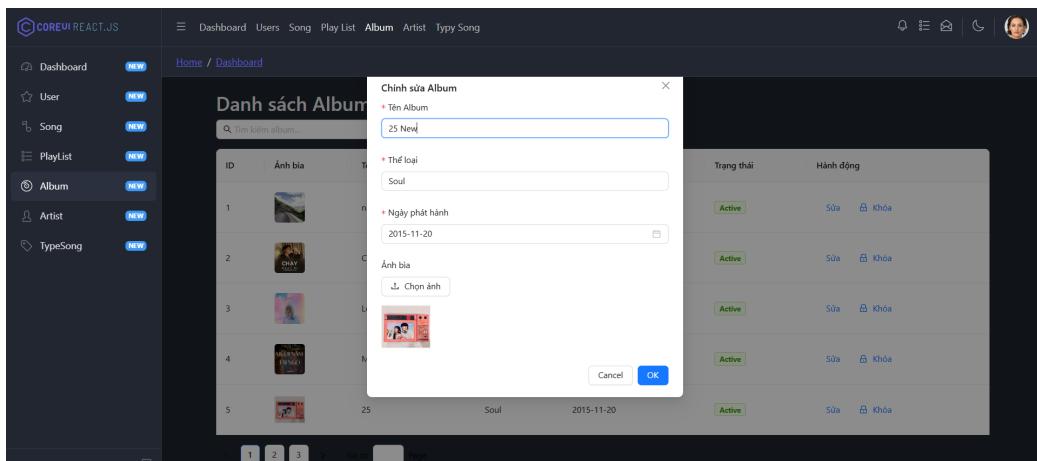
6.6.3 Quản lý album

- Danh sách album

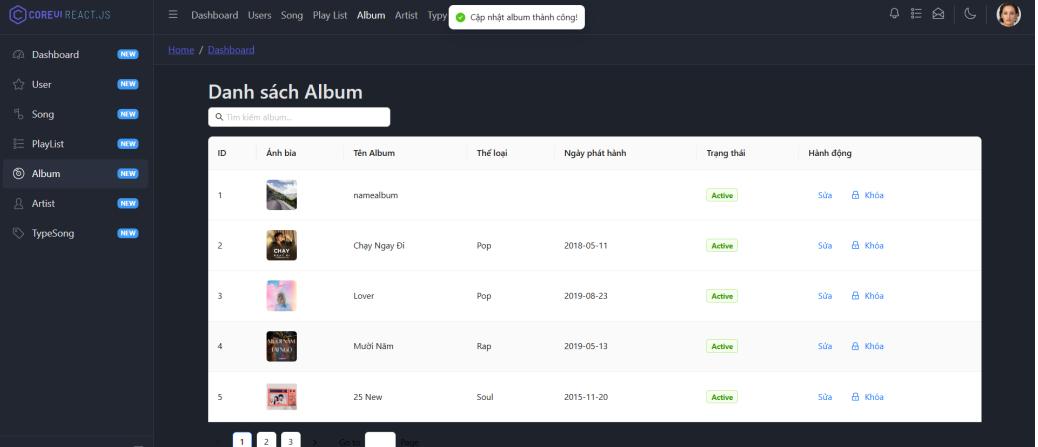
ID	Ánh bìa	Tên Album	Thể loại	Ngày phát hành	Trạng thái	Hành động
1		namealbum			Active	Sửa Khóa
2		Chạy Ngay Đi	Pop	2018-05-11	Active	Sửa Khóa
3		Lover	Pop	2019-08-23	Active	Sửa Khóa
4		Mùa Dưới Nắng	Rap	2019-11-13	Active	Sửa Khóa
5		25	Soul	2015-11-20	Active	Sửa Khóa

Hình 6.30: Danh sách album

- Sửa album



Hình 6.31: Sửa album

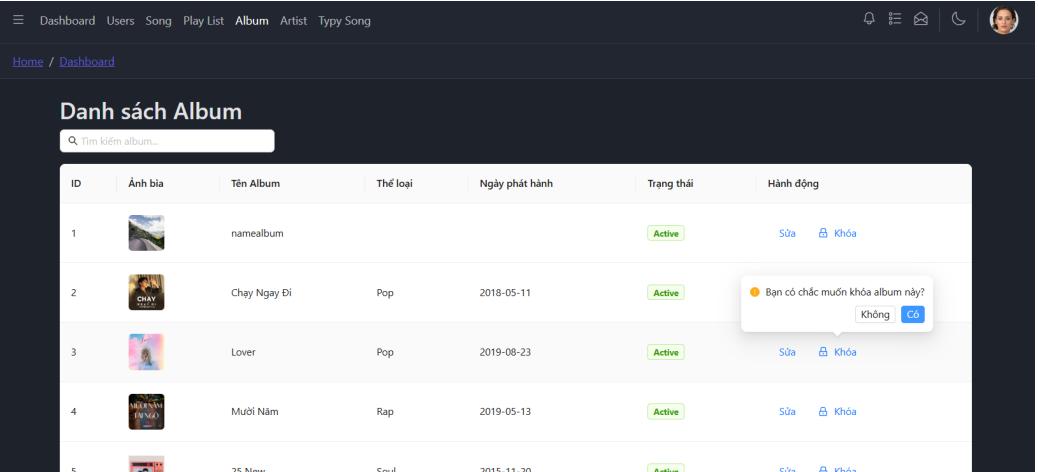


The screenshot shows a dark-themed dashboard with a sidebar containing links like Dashboard, User, Song, Playlist, Album, Artist, and Type Song. The main area displays a table titled "Danh sách Album" (Album List) with columns: ID, Ánh bìa (Cover), Tên Album (Album Name), Thể loại (Genre), Ngày phát hành (Release Date), Trạng thái (Status), and Hành động (Actions). Five albums are listed, all marked as "Active". A success message at the top right says "Cập nhật album thành công!" (Album updated successfully!).

ID	Ánh bìa	Tên Album	Thể loại	Ngày phát hành	Trạng thái	Hành động
1		namealbum			Active	Sửa <input type="button" value="Khóa"/>
2		Chạy Ngay Đi	Pop	2018-05-11	Active	Sửa <input type="button" value="Khóa"/>
3		Lover	Pop	2019-08-23	Active	Sửa <input type="button" value="Khóa"/>
4		Mười Năm	Rap	2019-05-13	Active	Sửa <input type="button" value="Khóa"/>
5		25 New	Soul	2015-11-20	Active	Sửa <input type="button" value="Khóa"/>

Hình 6.32: Sửa album thành công

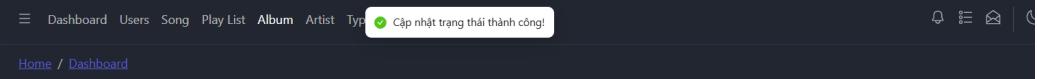
- Khóa album



This screenshot is similar to the previous one, showing the "Danh sách Album" table. In the row for album ID 2 (Chạy Ngay Đi), the "Khóa" button is highlighted with a blue border. A modal dialog box appears asking "Bạn có chắc muốn khóa album này?" (Are you sure you want to lock this album?). The user has selected "Có" (Yes).

ID	Ánh bìa	Tên Album	Thể loại	Ngày phát hành	Trạng thái	Hành động
1		namealbum			Active	Sửa <input type="button" value="Khóa"/>
2		Chạy Ngay Đi	Pop	2018-05-11	Active	Bạn có chắc muốn khóa album này? Không <input checked="" type="button" value="Có"/>
3		Lover	Pop	2019-08-23	Active	Sửa <input type="button" value="Khóa"/>
4		Mười Năm	Rap	2019-05-13	Active	Sửa <input type="button" value="Khóa"/>
5		25 New	Soul	2015-11-20	Active	Sửa <input type="button" value="Khóa"/>

Hình 6.33: Khóa album

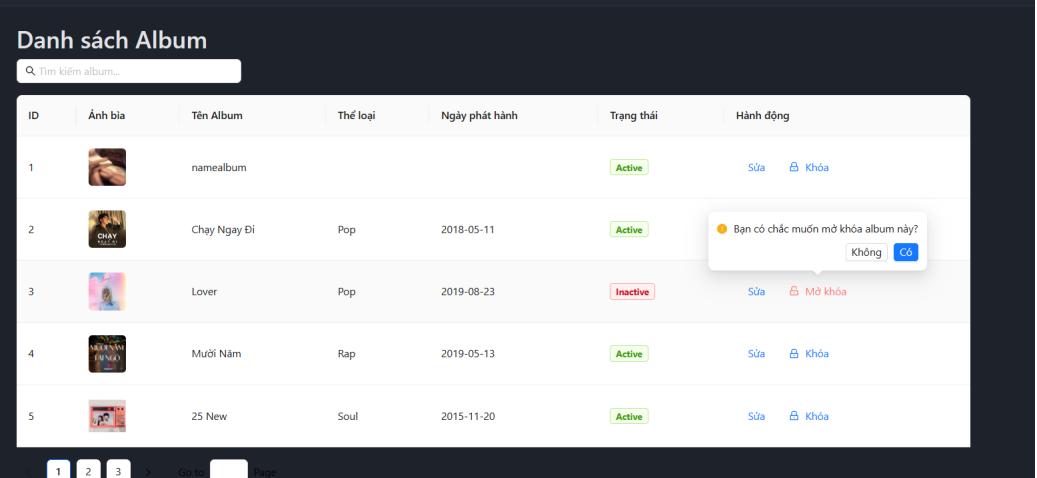


The screenshot shows a dark-themed dashboard with a navigation bar at the top containing links for Dashboard, Users, Song, Play List, Album, Artist, Type, and a success message: "Cập nhật trạng thái thành công!" (Status updated successfully!). Below the navigation is a breadcrumb trail: Home / Dashboard. The main title is "Danh sách Album". A search bar is present above the table. The table has columns: ID, Ánh bìa, Tên Album, Thể loại, Ngày phát hành, Trạng thái, and Hành động. The data rows are:

ID	Ánh bìa	Tên Album	Thể loại	Ngày phát hành	Trạng thái	Hành động
1		namealbum		2018-05-11	Active	Sửa Khóa
2		Chạy Ngay Đi	Pop	2018-05-11	Active	Sửa Khóa
3		Lover	Pop	2019-08-23	Inactive	Sửa Mở khóa
4		Mười Năm	Rap	2019-05-13	Active	Sửa Khóa

Hình 6.34: Khóa album thành công

- Mở khóa album



The screenshot shows the same dashboard as Figure 6.34. The table data is identical. A modal window is overlaid on the page, asking "Bạn có chắc muốn mở khóa album này?" (Are you sure you want to unlock this album?). It has two buttons: "Không" (No) and "Có" (Yes). The "Có" button is highlighted.

ID	Ánh bìa	Tên Album	Thể loại	Ngày phát hành	Trạng thái	Hành động
1		namealbum		2018-05-11	Active	Sửa Khóa
2		Chạy Ngay Đi	Pop	2018-05-11	Active	Sửa Khóa
3		Lover	Pop	2019-08-23	Inactive	Sửa Mở khóa
4		Mười Năm	Rap	2019-05-13	Active	Sửa Khóa
5		25 New	Soul	2015-11-20	Active	Sửa Khóa

Hình 6.35: Mở khóa album

The screenshot shows a dark-themed dashboard with a navigation bar at the top. A success message 'Cập nhật trạng thái thành công!' (Status updated successfully!) is displayed. Below it, a breadcrumb navigation shows 'Home / Dashboard'. The main content area is titled 'Danh sách Album' (Album List) and contains a search bar 'Tim kiếm album...'. A table lists five albums with columns: ID, Ảnh bìa (Cover), Tên Album (Album Name), Thể loại (Genre), Ngày phát hành (Release Date), Trạng thái (Status), and Hành động (Actions). The albums are:

ID	Ảnh bìa	Tên Album	Thể loại	Ngày phát hành	Trạng thái	Hành động
1		namealbum			Active	Sửa Khóa
2		Chạy Ngay Đi	Pop	2018-05-11	Active	Sửa Khóa
3		Lover	Pop	2019-08-23	Active	Sửa Khóa
4		Mười Năm	Rap	2019-05-13	Active	Sửa Khóa
5		25 New	Soul	2015-11-20	Active	Sửa Khóa

Pagination controls at the bottom allow for navigating through pages.

Hình 6.36: Mở khóa album thành công

- Tìm kiếm album

This screenshot shows the same application interface as Figure 6.36, but with a search query 'Ne' entered into the search bar. The search results table shows one item: '25 New' by Soul, released on 2015-11-20, with an 'Active' status and 'Sửa Khóa' actions.

Hình 6.37: Tìm kiếm album

6.6.4 Quản lý nghệ sĩ

- Danh sách nghệ sĩ

The screenshot shows a dark-themed web application interface for managing artists. On the left, there's a sidebar with navigation links: Dashboard, User, Song, Playlist, Album, Artist (which is currently selected and highlighted in blue), and TypeSong. The main content area has a title 'Danh sách Nghệ Sĩ' and a search bar. Below it is a table listing 10 artists:

ID	Ánh	Tên nghệ sĩ	Quốc gia	Ngày sinh	Trạng thái	Hành động
6		Adele	Anh	1988-05-05	Active	Sửa Khóa
7		Bích Phương	Việt Nam	1989-09-30	Active	Sửa Khóa
8		Ed Sheeran	Anh	1991-02-17	Active	Sửa Khóa
9		Hà Anh Tuấn	Việt Nam	1984-12-17	Active	Sửa Khóa
10		Billie Eilish	Mỹ	2001-12-18	Active	Sửa Khóa

Pagination at the bottom shows pages 1, 2, 3, and Go to [page].

Hình 6.38: Danh sách nghệ sĩ

- Thêm nghệ sĩ mới

The screenshot shows a modal dialog titled 'Thêm Nghệ Sĩ' (Add Artist) over a dark-themed artist list. The modal has the following fields:

- Tên nghệ sĩ: Binz
- Ngày sinh: 1988-04-16
- Quốc gia: Việt Nam
- Tiểu sử: Description
- Ảnh đại diện: Chọn ảnh (Select photo)

At the bottom of the modal are 'Cancel' and 'OK' buttons.

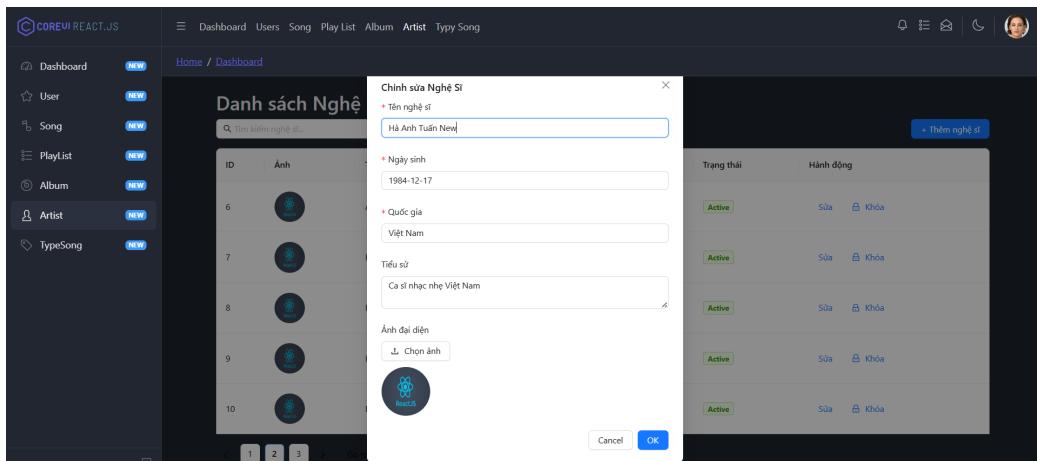
Hình 6.39: Thêm nghệ sĩ mới

ID	Ảnh	Tên nghệ sĩ	Quốc gia	Ngày sinh	Trạng thái	Hành động
11		Trịnh Công Sơn	Việt Nam	1939-02-28	Active	Sửa Khóa
12		Bruno Mars	Mỹ	1985-10-08	Active	Sửa Khóa
13		Binz	Việt Nam	1988-04-16	Active	Sửa Khóa

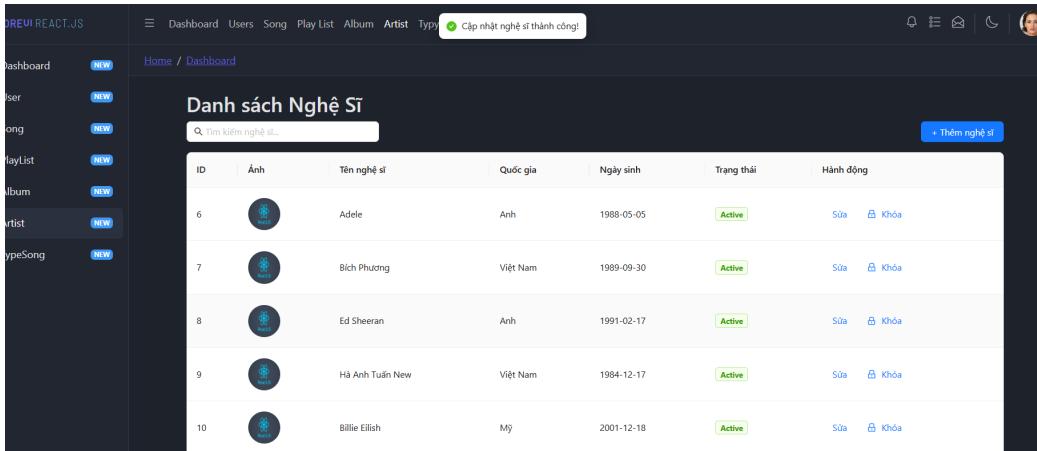
Go to Page < 1 2 3 >

Hình 6.40: Thêm nghệ sĩ mới thành công

- Sửa nghệ sĩ



Hình 6.41: Sửa nghệ sĩ

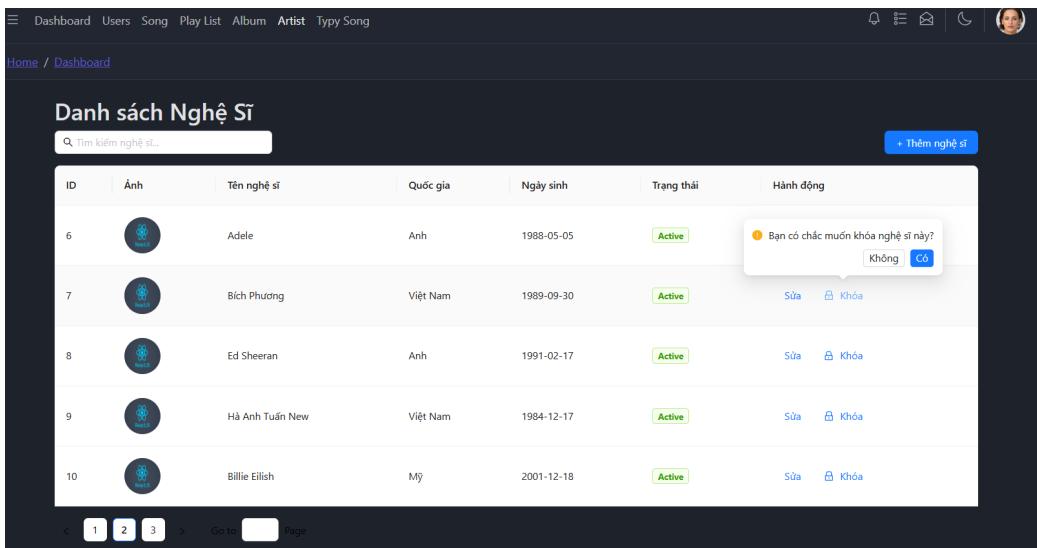


The screenshot shows a dark-themed dashboard with a sidebar containing links like 'Dashboard', 'User', 'Song', 'Play List', 'Album', 'Artist', 'Typy', and 'TypeSong'. The main area displays a table titled 'Danh sách Nghệ Sĩ' (Artist List) with columns: ID, Ánh (Avatar), Tên nghệ sĩ (Artist Name), Quốc gia (Country), Ngày sinh (Birth Date), Trạng thái (Status), and Hành động (Actions). The table contains 5 rows of data. A success message 'Cập nhật nghệ sĩ thành công!' (Artist updated successfully!) is displayed at the top right.

ID	Ánh	Tên nghệ sĩ	Quốc gia	Ngày sinh	Trạng thái	Hành động
6		Adele	Anh	1988-05-05	Active	Sửa Khóa
7		Bích Phương	Việt Nam	1989-09-30	Active	Sửa Khóa
8		Ed Sheeran	Anh	1991-02-17	Active	Sửa Khóa
9		Hà Anh Tuấn New	Việt Nam	1984-12-17	Active	Sửa Khóa
10		Billie Eilish	Mỹ	2001-12-18	Active	Sửa Khóa

Hình 6.42: Sửa nghệ sĩ thành công

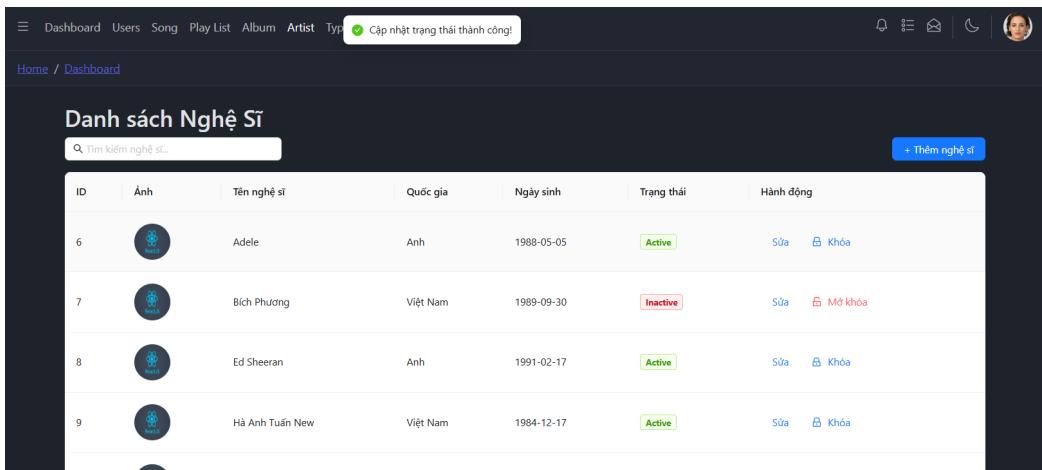
- Khóa nghệ sĩ



The screenshot shows the same artist list interface. When the lock icon is clicked for the first artist (Adele), a confirmation dialog appears asking 'Bạn có chắc muốn khóa nghệ sĩ này?' (Are you sure you want to lock this artist?). The user has selected 'Có' (Yes). The dialog also includes 'Không' (No) and a close button.

ID	Ánh	Tên nghệ sĩ	Quốc gia	Ngày sinh	Trạng thái	Hành động
6		Adele	Anh	1988-05-05	Active	Bạn có chắc muốn khóa nghệ sĩ này? Không Có
7		Bích Phương	Việt Nam	1989-09-30	Active	
8		Ed Sheeran	Anh	1991-02-17	Active	Sửa Khóa
9		Hà Anh Tuấn New	Việt Nam	1984-12-17	Active	Sửa Khóa
10		Billie Eilish	Mỹ	2001-12-18	Active	Sửa Khóa

Hình 6.43: Khóa nghệ sĩ

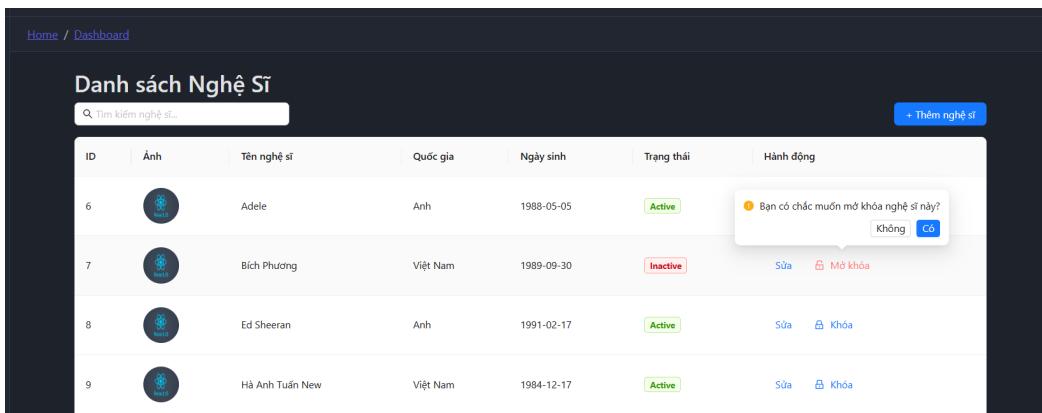


The screenshot shows a dark-themed web application interface. At the top, there is a navigation bar with links: Dashboard, Users, Song, Play List, Album, Artist, Type, and a button labeled 'Cập nhật trạng thái thành công!' (Status updated successfully!). Below the navigation bar is a breadcrumb trail: Home / Dashboard. The main title is 'Danh sách Nghệ Sĩ' (Artist List). A search bar with placeholder text 'Tìm kiếm nghệ sĩ...' is located above the table. A blue button '+Thêm nghệ sĩ' (Add artist) is on the right. The table has columns: ID, Ảnh (Avatar), Tên nghệ sĩ (Artist Name), Quốc gia (Country), Ngày sinh (Birth Date), Trạng thái (Status), and Hành động (Actions). The data rows are:

ID	Ảnh	Tên nghệ sĩ	Quốc gia	Ngày sinh	Trạng thái	Hành động
6		Adele	Anh	1988-05-05	Active	Sửa Khóa
7		Bích Phương	Việt Nam	1989-09-30	Inactive	Sửa Mở khóa
8		Ed Sheeran	Anh	1991-02-17	Active	Sửa Khóa
9		Hà Anh Tuấn New	Việt Nam	1984-12-17	Active	Sửa Khóa

Hình 6.44: Khóa nghệ sĩ thành công

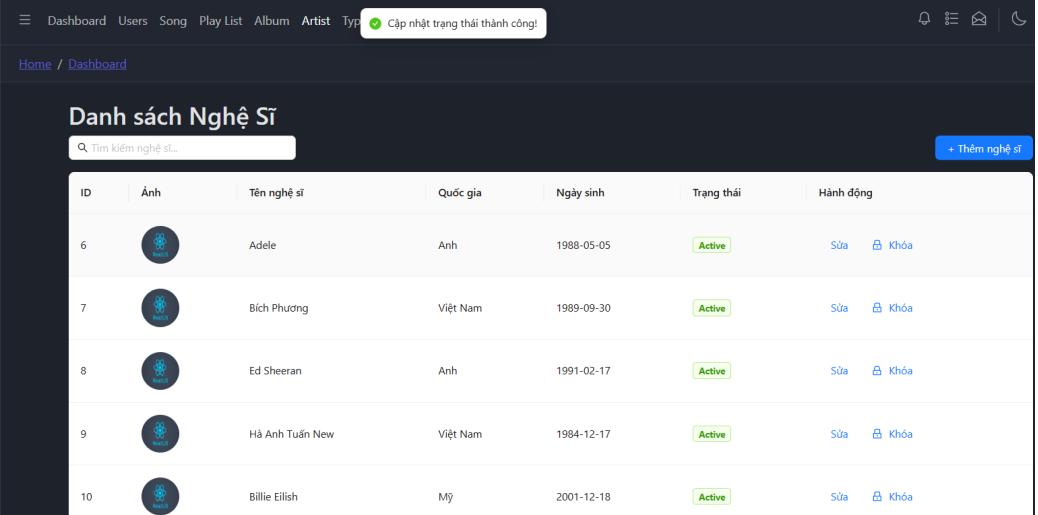
- Mở khóa nghệ sĩ



This screenshot is similar to the previous one but shows a modal dialog box asking if the user wants to unlock an artist. The dialog contains the text 'Bạn có chắc muốn mở khóa nghệ sĩ này?' (Are you sure you want to unlock this artist?) with two buttons: 'Không' (No) and 'Có' (Yes). The rest of the interface and data table are identical to Figure 6.44.

ID	Ảnh	Tên nghệ sĩ	Quốc gia	Ngày sinh	Trạng thái	Hành động
6		Adele	Anh	1988-05-05	Active	● Bạn có chắc muốn mở khóa nghệ sĩ này? Không Có
7		Bích Phương	Việt Nam	1989-09-30	Inactive	Sửa Mở khóa
8		Ed Sheeran	Anh	1991-02-17	Active	Sửa Khóa
9		Hà Anh Tuấn New	Việt Nam	1984-12-17	Active	Sửa Khóa

Hình 6.45: Mở khóa nghệ sĩ

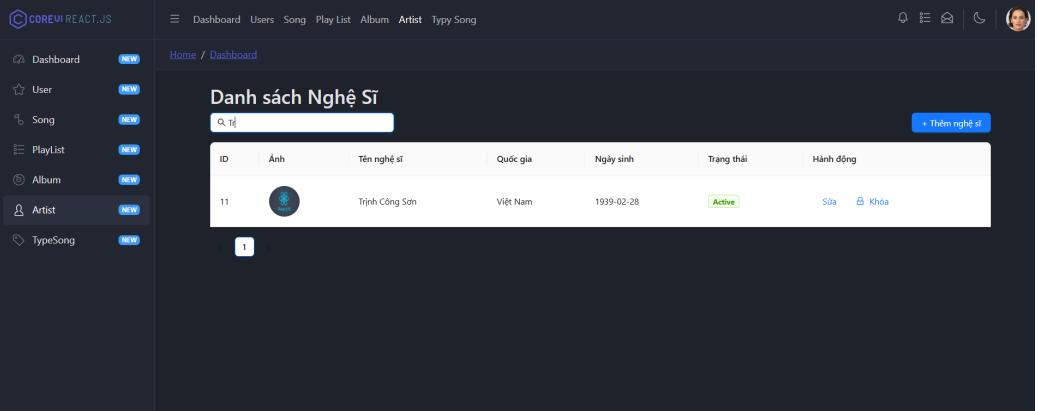


The screenshot shows a dark-themed application interface titled "Danh sách Nghệ Sĩ". At the top, there is a navigation bar with links for Dashboard, Users, Song, Play List, Album, Artist, Type, and a button labeled "Cập nhật trạng thái thành công!" (Update status successfully!). Below the navigation is a breadcrumb trail: Home / Dashboard. The main content area is titled "Danh sách Nghệ Sĩ" and contains a search bar with placeholder text "Tìm kiếm nghệ sĩ...". A blue button labeled "+ Thêm nghệ sĩ" is located in the top right corner of the table header. The table has columns for ID, Ảnh (Avatar), Tên nghệ sĩ (Artist Name), Quốc gia (Country), Ngày sinh (Birth Date), Trạng thái (Status), and Hành động (Actions). The data rows are as follows:

ID	Ảnh	Tên nghệ sĩ	Quốc gia	Ngày sinh	Trạng thái	Hành động
6		Adele	Anh	1988-05-05	Active	Sửa Khóa
7		Bích Phương	Việt Nam	1989-09-30	Active	Sửa Khóa
8		Ed Sheeran	Anh	1991-02-17	Active	Sửa Khóa
9		Hà Anh Tuấn New	Việt Nam	1984-12-17	Active	Sửa Khóa
10		Billie Eilish	Mỹ	2001-12-18	Active	Sửa Khóa

Hình 6.46: Mở khóa nghệ sĩ thành công

- Tìm kiếm nghệ sĩ



The screenshot shows a dark-themed application interface titled "Danh sách Nghệ Sĩ". On the left, there is a sidebar with a tree view of modules: Dashboard (NEW), User (NEW), Song (NEW), Playlist (NEW), Album (NEW), Artist (NEW), and TypeSong (NEW). The "Artist" node is expanded. The main content area is titled "Danh sách Nghệ Sĩ" and contains a search bar with placeholder text "Tìm kiếm nghệ sĩ...". A blue button labeled "+ Thêm nghệ sĩ" is located in the top right corner of the table header. The table has columns for ID, Ảnh (Avatar), Tên nghệ sĩ (Artist Name), Quốc gia (Country), Ngày sinh (Birth Date), Trạng thái (Status), and Hành động (Actions). There is one data row:

ID	Ảnh	Tên nghệ sĩ	Quốc gia	Ngày sinh	Trạng thái	Hành động
11		Trịnh Công Sơn	Việt Nam	1939-02-28	Active	Sửa Khóa

Hình 6.47: Tìm kiếm nghệ sĩ

6.6.5 Quản lý loại bài hát

- Danh sách loại bài hát

ID	Tên loại bài hát	Mô tả	Ngày tạo	Hành động
	Cải lương	Đây là cải lương nè	2025-03-21T09:52:31Z	<button>Sửa</button> <button>Xóa</button>
	Pop	Nhạc pop phổ biến	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>
	Rock	Nhạc rock mạnh mẽ	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>
	Rap	Nhạc rap năng động	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>
	Ballad	Nhạc ballad nhẹ nhàng	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>
	Soul	Nhạc soul sâu lắng	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>
	Alternative	Nhạc thay thế độc đáo	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>
	Nhạc Trịnh	Nhạc Trịnh Công Sơn	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>

Hình 6.48: Danh sách loại bài hát

- Sửa loại bài hát

ID	Tên loại bài hát	Mô tả	Ngày tạo	Hành động
	Cải lương	Đây là cải lương	09:52:31Z	<button>Sửa</button> <button>Xóa</button>
	Pop	Nhạc pop phổ biến	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>
	Rock	Nhạc rock mạnh mẽ	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>
	Rap	Nhạc rap năng động	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>
	Ballad	Nhạc ballad nhẹ nhàng	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>

Hình 6.49: Sửa loại bài hát

ID	Tên loại bài hát	Mô tả	Ngày tạo	Hành động
Cái lương New	Đây là cái lương	2025-03-21T09:52:31Z	<button>Sửa</button> <button>Xóa</button>	
Pop	Nhạc pop phổ biến	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	
Rock	Nhạc rock mạnh mẽ	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	
Rap	Nhạc rap năng động	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	
Ballad	Nhạc ballad nhẹ nhàng	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	
Soul	Nhạc soul sâu lắng	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	
Alternative	Nhạc thay thế độc đáo	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	

Hình 6.50: Sửa loại bài hát

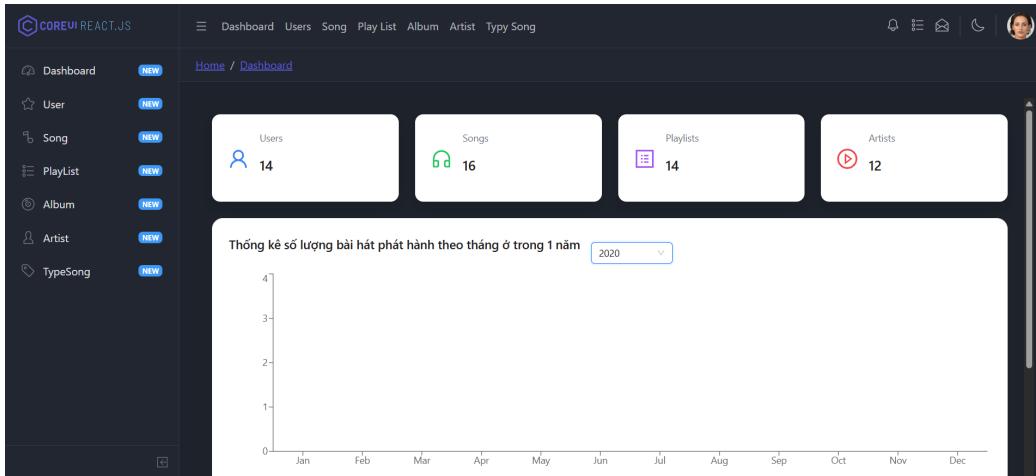
- Xóa loại bài hát

ID	Tên loại bài hát	Mô tả	Ngày tạo	Hành động
Pop	Nhạc pop phổ biến	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	
Rock	Nhạc rock mạnh mẽ	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	
Rap	Nhạc rap năng động	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	
Ballad	Nhạc ballad nhẹ nhàng	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	
Soul	Nhạc soul sâu lắng	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	
Alternative	Nhạc thay thế độc đáo	2025-04-08T09:42:48Z	<button>Sửa</button> <button>Xóa</button>	

Hình 6.51: Xóa loại bài hát

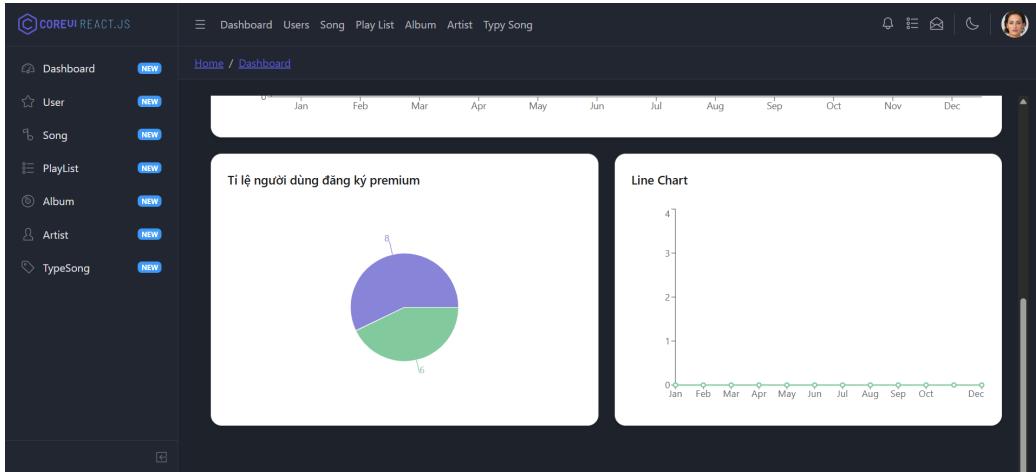
6.6.6 Thông kê

- Thông kê: Số lượng người dùng, bài hát, danh sách phát, nghệ sĩ, bài hát phát hành trong 1 năm



Hình 6.52: Thông kê: Số lượng người dùng, bài hát, danh sách phát, nghệ sĩ, bài hát phát hành trong 1 năm

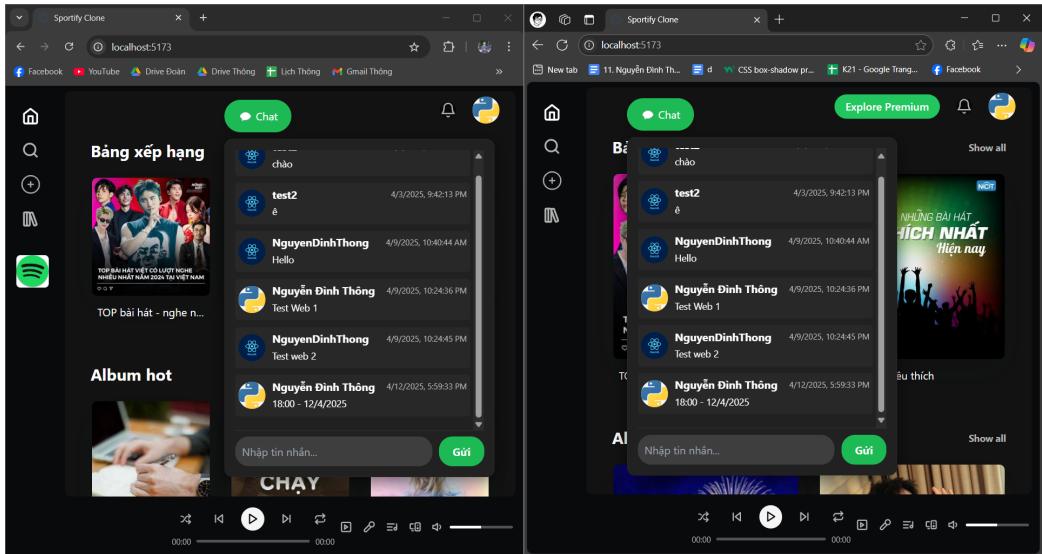
- Thông kê: Tỷ lệ người dùng đăng ký Premium theo biểu đồ tròn và biểu đồ đường



Hình 6.53: Thông kê: Tỷ lệ người dùng đăng ký Premium theo biểu đồ tròn và biểu đồ đường

6.7 Tính năng chat tích hợp

Ứng dụng hỗ trợ người dùng gửi tin nhắn trực tiếp trong giao diện web, giúp tăng cường tương tác cộng đồng. Tính năng có thể được tích hợp bằng WebSocket.

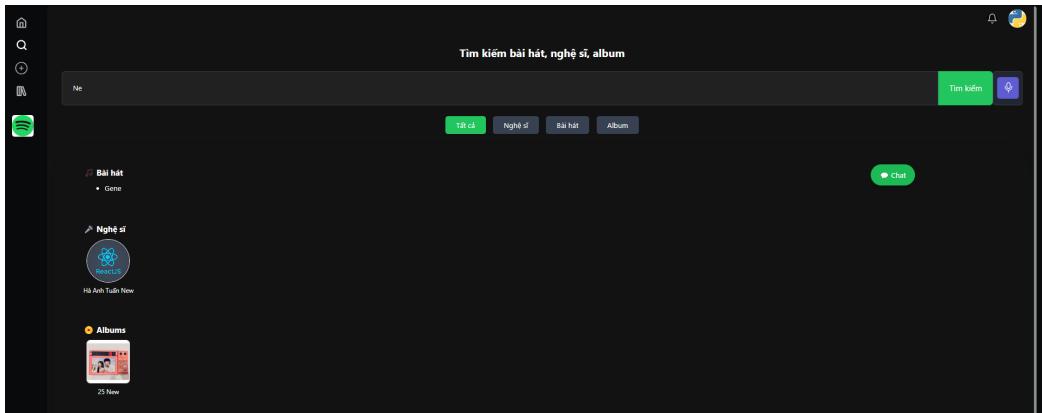


Hình 6.54: Giao diện tính năng chat

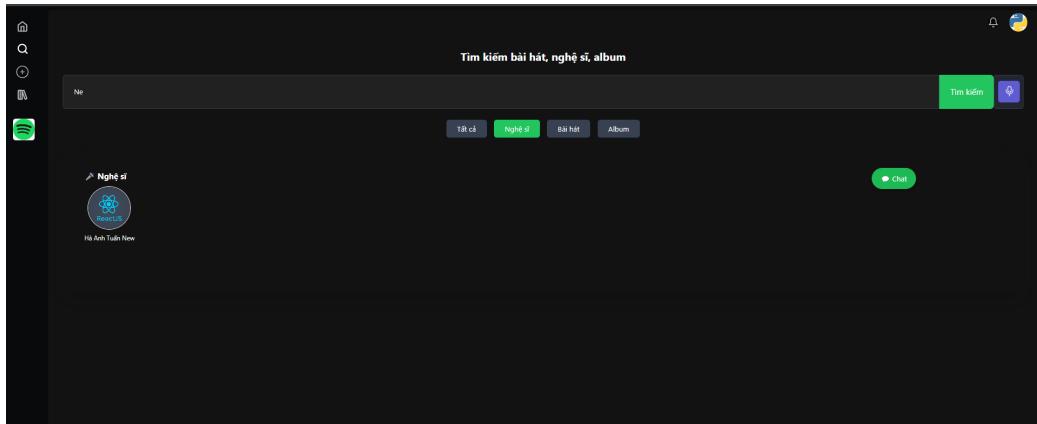
6.8 Chức năng tìm kiếm

6.8.1 Tìm kiếm theo từ khóa

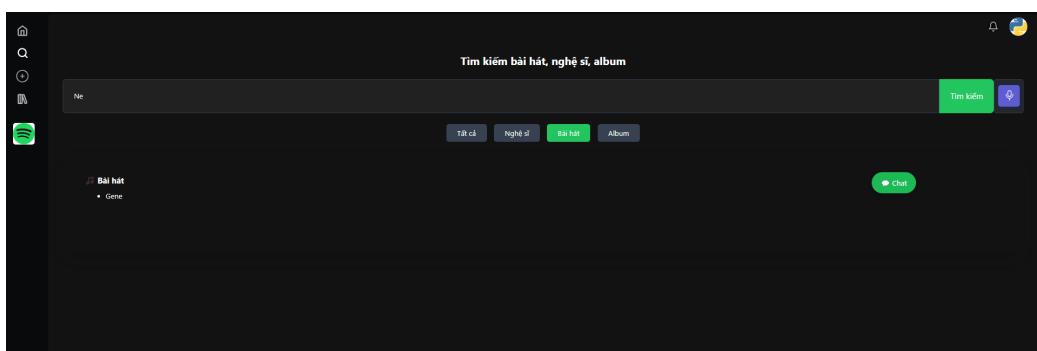
Chức năng cho phép người dùng nhập từ khóa (tên bài hát, ca sĩ, thể loại, v.v.) vào thanh tìm kiếm để nhận kết quả gợi ý phù hợp. Kết quả sẽ hiển thị danh sách các bài hát, ca sĩ, album tương ứng và cho phép lọc.



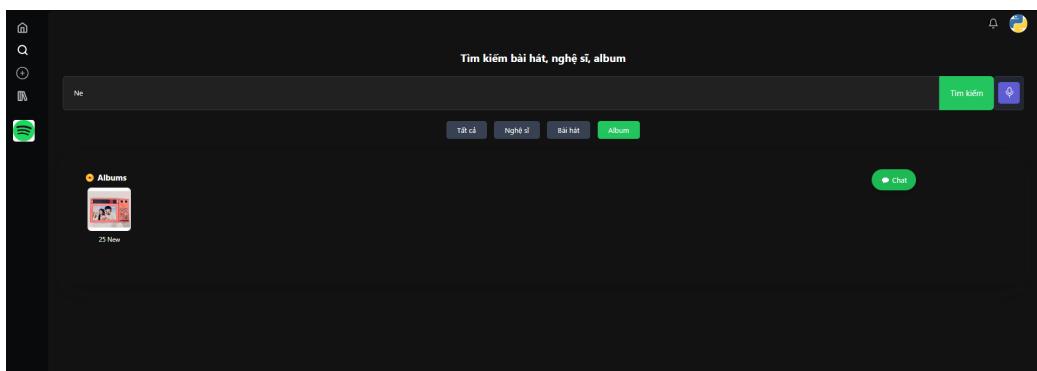
Hình 6.55: Giao diện tìm kiếm theo từ khóa



Hình 6.56: Giao diện tìm kiếm nghệ sĩ



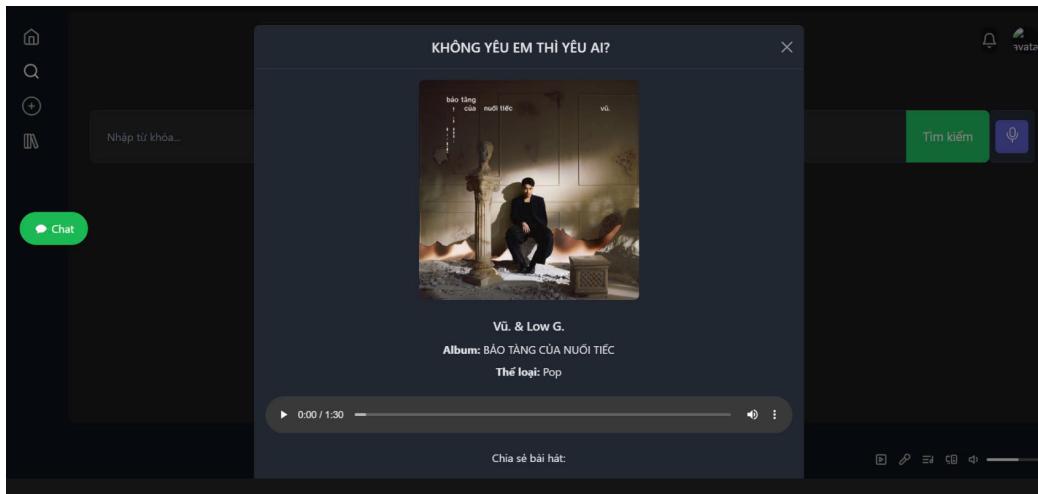
Hình 6.57: Giao diện tìm kiếm bài hát



Hình 6.58: Giao diện tìm kiếm bài hát

6.8.2 Tìm kiếm bài hát dựa trên tệp ghi âm

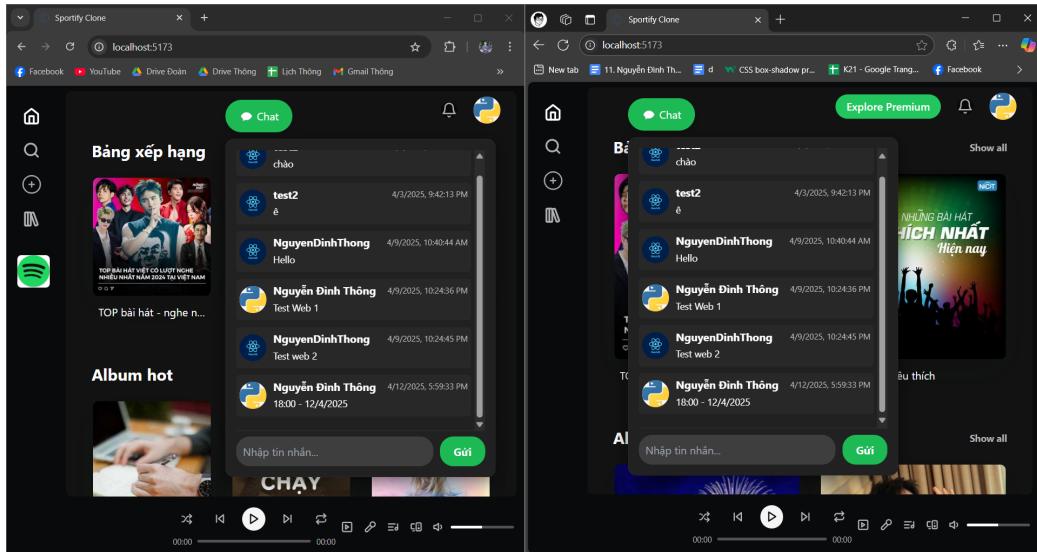
Người dùng có thể ghi âm đoạn bài hát. Hệ thống sử dụng mô hình mã nguồn mở Shazamio để phân tích và trả về dữ liệu của bài hát giống nhất.



Hình 6.59: Giao diện tìm kiếm bài hát dựa trên tệp ghi âm

6.9 Tính năng chat tích hợp

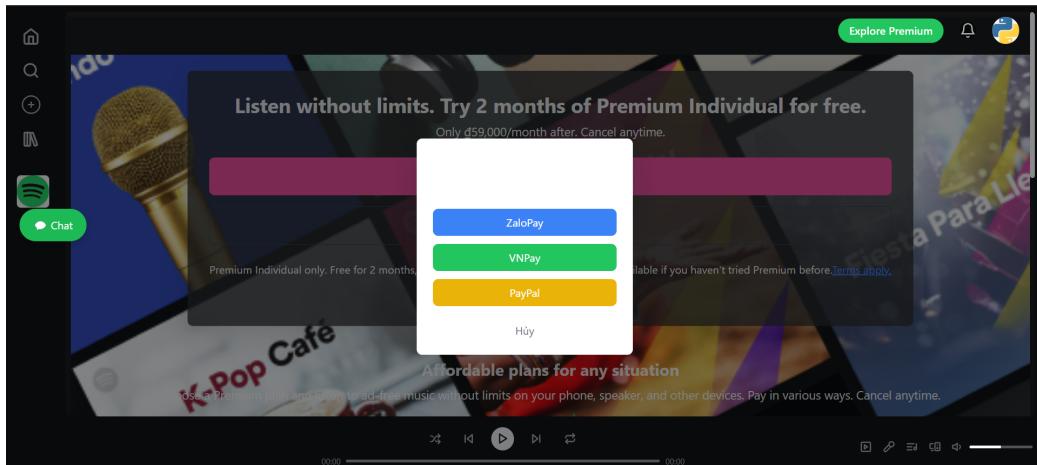
Ứng dụng hỗ trợ người dùng gửi tin nhắn trực tiếp trong giao diện web, giúp tăng cường tương tác cộng đồng. Tính năng có thể được tích hợp bằng WebSocket.



Hình 6.60: Giao diện tính năng chat

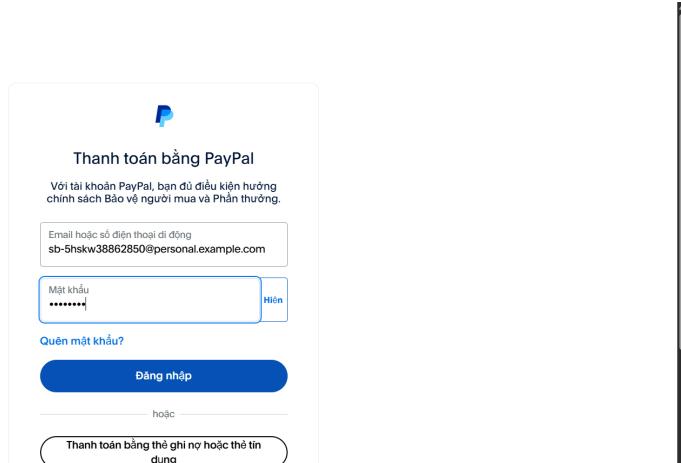
6.10 Premium

Khi người dùng mua Premium thì sẽ nghe nhạc mà không bị chèn quảng cáo giữa các bài.

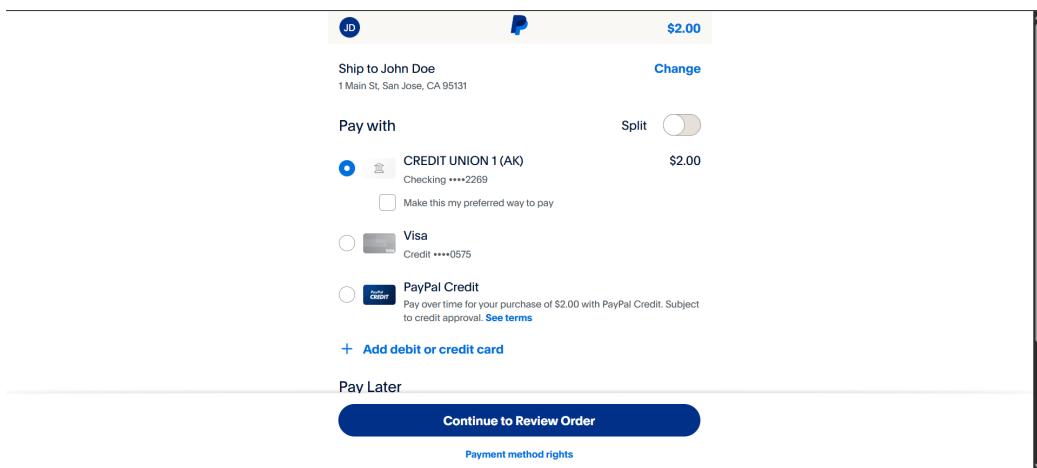


Hình 6.61: Giao diện mua premium

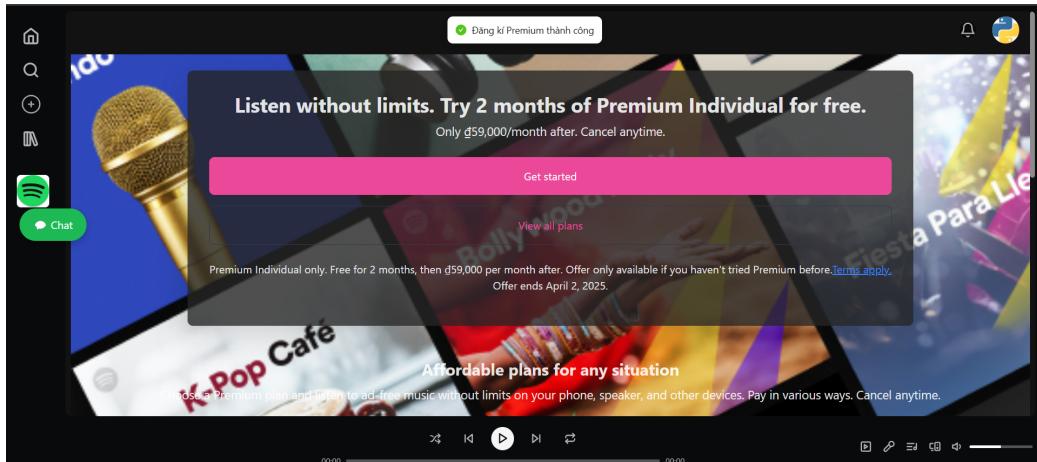
6.10.1 Mua Premium thanh toán bằng PayPal



Hình 6.62: Mua Premium thanh toán bằng PayPal

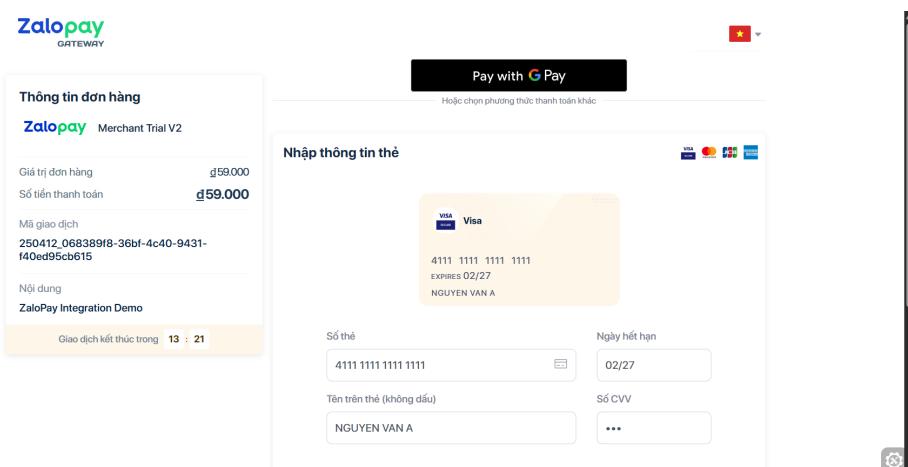


Hình 6.63: Mua Premium thanh toán bằng PayPal

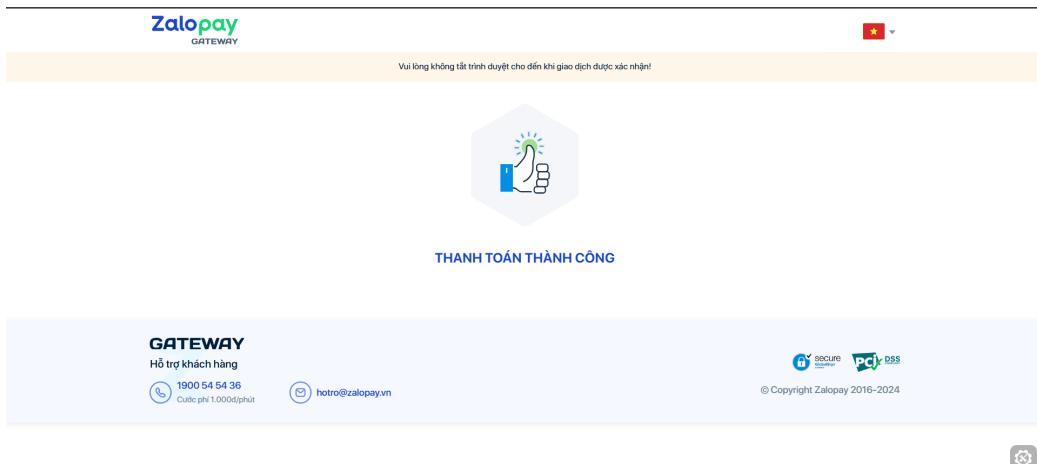


Hình 6.64: Mua Premium thanh toán bằng PayPal thành công

6.10.2 Mua Premium thanh toán bằng Zalo pay



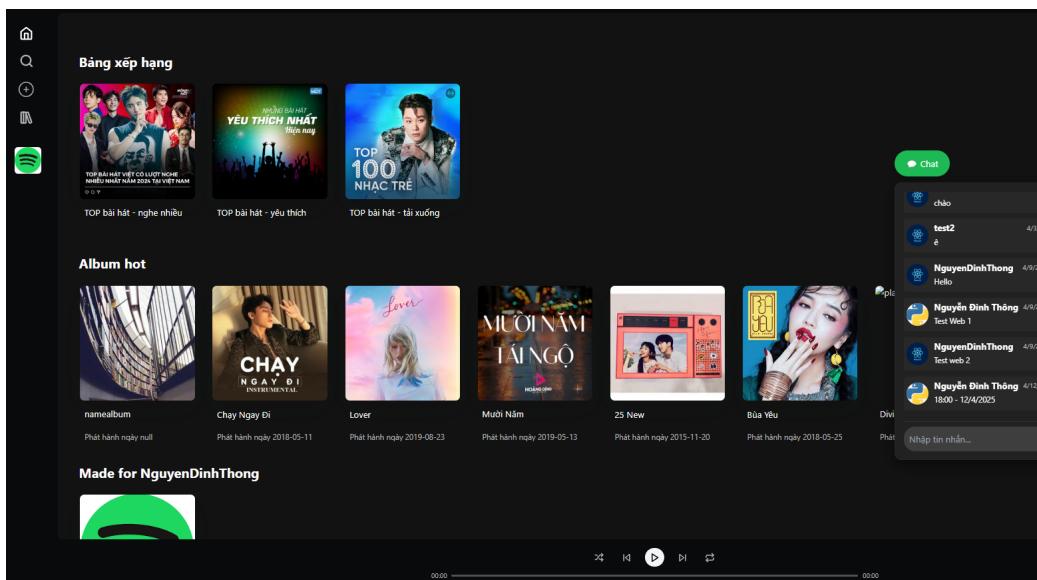
Hình 6.65: Mua Premium thanh toán bằng Zalo pay



Hình 6.66: Mua Premium thanh toán bằng Zalo pay thành công

6.11 Bảng xếp hạng và album hot

Cung cấp hệ thống bảng xếp hạng âm nhạc thịnh hành dựa vào lượt nghe.



Hình 6.67: Giao diện bảng xếp hạng và album hot

Chương 7

HƯỚNG PHÁT TRIỂN²

- **Tích hợp các dịch vụ phát nhạc trực tuyến:** Tích hợp các dịch vụ phát nhạc trực tuyến để cung cấp nguồn nhạc phong phú và chất lượng cao.
- **Cải thiện trải nghiệm người dùng (UX/UI):** Cải thiện trải nghiệm người dùng (UX/UI): Thiết kế giao diện hấp dẫn, thân thiện và dễ sử dụng hơn, đồng thời tối ưu hóa hiệu suất để người dùng có trải nghiệm mượt mà.
- **Phát triển các tính năng xã hội:** Phát triển các tính năng xã hội: Cho phép người dùng chia sẻ playlist, theo dõi bạn bè và xem hoạt động nghe nhạc của họ để tạo sự kết nối trong cộng đồng người dùng.
- **Hỗ trợ đa nền tảng:** Hỗ trợ đa nền tảng: Đảm bảo ứng dụng hoạt động trên nhiều thiết bị như iPad, di động và máy tính để bàn, đồng thời đồng bộ hóa dữ liệu người dùng giữa các thiết bị.
- **Tối ưu hóa cho di động:** Tối ưu hóa cho di động: Đảm bảo ứng dụng di động có hiệu suất tốt, giao diện thân thiện và tiết kiệm pin, mang lại trải nghiệm người dùng tối ưu trên các thiết bị di động.
- **Phân tích và đề xuất nhạc thông minh:** Phân tích và đề xuất nhạc thông minh: Sử dụng trí tuệ nhân tạo để phân tích thói quen nghe nhạc của người dùng và đề xuất các bài hát, playlist phù hợp, tạo sự cá nhân hóa trong trải nghiệm người dùng.
- **Đảm bảo tuân thủ bản quyền:** Đảm bảo tuân thủ bản quyền: Thiết lập các thỏa thuận với các nhà cung cấp nội dung và tuân thủ quy định về bản quyền âm nhạc để tránh các vấn đề pháp lý liên quan.

Chương 8

MÔI TRƯỜNG CHẠY ỨNG DỤNG VÀ CÁCH CÀI ĐẶT

8.1 Môi trường chạy ứng dụng

- **Backend - Django:** Sử dụng ngôn ngữ Python phiên bản 3.12.7, framework Django 5.1.6. Các gói phụ thuộc được liệt kê trong file `requirements.txt`.
- **Frontend - React:** Dựa trên ngôn ngữ JavaScript (ES6+), React phiên bản 18.3.1. Môi trường chạy gồm Node.js v20.16.0 và npm 10.8.1. Các gói phụ thuộc được khai báo trong file `package.json`.
- **Database - MySQL:** Sử dụng MySQL 8.0.30 và MySQL Workbench 8.0.30 để quản lý cơ sở dữ liệu.

8.2 Cách cài đặt

- Cài đặt Backend - Django:

```
git clone https://github.com/lamkbvn/BACKEND_SPOTIFY.git
cd BACKEND_SPOTIFY/my_project
pip install -r requirements.txt
python manage.py migrate
python manage.py runserver
```

- **Cài đặt Frontend - React:**

```
git clone https://github.com/duylam15/react-clone-spotify.git  
cd react-clone-spotify  
npm install  
npm run dev
```

- **Cài đặt Database - MySQL:** Nhập dữ liệu từ file `spotify.sql` vào cơ sở dữ liệu: