

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM



ĐỒ ÁN 2

**PHÁT TRIỂN HỆ THỐNG GỌI Ý NHẠC THÔNG MINH
DỰA TRÊN NHẬN DIỆN CẢM XÚC TỪ CAMERA**

Giảng viên hướng dẫn : TS. Huỳnh Minh Đức

Sinh viên thực hiện 1 : Phạm Hoàng Duy

Mã sinh viên : 22520339

Sinh viên thực hiện 2 : Hà Phú Thịnh

Mã sinh viên 2 : 22521405

TP. Hồ Chí Minh, 2025

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Thầy Huỳnh Minh Đức – Giảng Viên Khoa Công nghệ Phần mềm, đã là người đồng hành cùng chúng em trong suốt quá trình thực hiện đồ án và giúp đỡ chúng em hoàn thiện đồ án này cách tốt nhất.

Chúng em cũng xin cảm ơn Trường Đại học Công nghệ Thông tin và Khoa Công nghệ Phần mềm đã tạo cơ hội cho chúng em được thực hành môn Đồ Án 2, một môn học mang tính ứng dụng và chuyên môn cao để phần nào giúp chúng em trang bị những kiến thức hữu dụng cho công việc tương lai sau này.

Chúng con cũng xin cảm ơn ba mẹ và người thân vì đã luôn tạo điều kiện, quan tâm, giúp đỡ và động viên để chúng con có thể hoàn thiện đồ án môn học này.

Cuối cùng, chúng em xin cảm ơn những người bạn tốt, những người cộng sự luôn kè vai sát cánh đưa ra những lời khuyên và giải pháp kịp thời để đồ án của chúng em được hoàn thành một cách trọn vẹn.

Tuy nhiên, với điều kiện thời gian và kinh nghiệm còn có hạn đối với sinh viên như chúng em, chúng em nhận thấy còn nhiều thiếu sót và chỉnh sửa đối với đồ án. Chúng em mong nhận được các ý kiến, đóng góp từ quý thầy cô để giúp chúng em có thể cải thiện, học hỏi và nâng cao kiến thức của bản thân.

TP. Hồ Chí Minh, tháng 06 năm 2025

Nhóm sinh viên thực hiện

Phạm Hoàng Duy – Hà Phú Thịnh

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....	1
1.1. Lý do chọn đề tài.....	1
1.2. Mục tiêu	2
1.3. Phạm vi	2
1.4. Đối tượng sử dụng	3
1.5. Phương pháp thực hiện	4
1.6. Công nghệ sử dụng	4
1.7. Kết quả mong đợi	5
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT.....	6
2.1. Nền tảng lý thuyết.....	6
2.1.1. Nhận diện cảm xúc.....	6
2.1.2. Gợi ý âm nhạc	13
2.1.3. Restricted Boltzmann Machine (RBM)	16
2.1.4. KNN	19
2.1.5. Hybrid method	20
2.2. Công nghệ áp dụng.....	22
2.2.1. Typescript.....	22
2.2.2. ReactJS	23
2.2.3. NodeJS	24
2.2.4. Python (cho Recommender Service)	25
2.2.4. PostgreSQL	25
2.2.5. Mô hình máy khách – máy chủ (client – server)	27
2.2.6. Python	28
2.2.7. Redis.....	29
2.2.8. Hybrid Algorithm (RBM + k-NN).....	30
CHƯƠNG 3. TRIỂN KHAI THUẬT TOÁN	31
3.1. Ý tưởng thuật toán	31
3.2. Triển khai phát hiện cảm xúc.....	31
3.2.1. Mô tả luồng xử lý	31
3.2.2. Các Loại Sự kiện Tương tác (Interaction Events)	33
3.2.3. Scoring Model.....	34
3.2.4. Cơ chế Cập nhật Điểm Tích lũy.....	37

3.3. Triển khai phân tích cảm xúc bài hát.....	38
3.4. Triển khai recommend dựa trên cảm xúc	41
3.5. Pipeline End-to-End	42
CHƯƠNG 4. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG.....	43
4.1. Xác định yêu cầu.....	43
4.1.1. Yêu cầu chức năng	43
4.1.2. Yêu cầu nghiệp vụ.....	43
4.2. Mô hình hóa yêu cầu	44
4.2.1. Lược đồ Use Case	44
4.2.2. Danh sách chức năng	45
4.2.3. Danh sách tác nhân (Actor).....	48
4.2.4. Danh sách trường hợp sử dụng (Use Case).....	49
4.2.5. Đặc tả Use Case	53
4.3. Thiết kế hệ thống	72
4.3.1. Kiến trúc hệ thống.....	72
4.3.2. Bảng mô tả thành phần.....	73
4.4. Thiết kế dữ liệu	77
4.4.1. Sơ đồ cơ sở dữ liệu (Database Diagram)	77
4.4.2. Mô tả các Table trong cơ sở dữ liệu.....	77
4.5. Thiết kế giao diện.....	86
4.5.1. Danh sách các màn hình.....	86
4.6. Thiết kế luồng Recommendations	88
4.6.1. Gán nhãn music emotion.....	88
4.6.2. Gửi user interact emotion data	89
4.6.3. Thu thập và xử lý user interact emotion data.....	90
4.6.4. Tạo và lưu recommendations	90
4.6.5. Lấy recommendations cho user	91
CHƯƠNG 5. TRIỂN KHAI ỨNG DỤNG	91
5.1. Triển khai Frontend	91
5.1.1. Quy trình triển khai Frontend	91
5.1.2. Lợi ích khi triển khai với Cloudflare:	94
5.1.3. Lợi ích khi triển khai với Docker.....	95
5.2. Triển khai Backend	95
5.2.1. Quy trình triển khai Backend	96

5.2.2. Xây dựng blue green deployment	96
5.3. Triển khai recommender services.....	96
CHƯƠNG 6. HIỆN THỰC HÓA VÀ KẾT QUẢ	99
6.1. Nhận diện cảm xúc bài hát	99
6.1.1. Huấn luyện mô hình.....	100
6.1.2. Chạy job gán nhãn.....	102
6.2. Nhận diện cảm xúc người dùng.....	107
6.3. Thu thập và gửi sự kiện	110
6.4. Xử lý sự kiện.....	112
6.4.1. Xử lý sự kiện cron_collect	114
6.4.2. Xử lý sự kiện like	114
6.4.3. Xử lý sự kiện listen_through.....	115
6.4.4. Xử lý sự kiện start_playing	115
6.4.5. Xử lý sự kiện skip	115
6.5. Tạo Recommendation.....	116
6.5.1. Retrieve dữ liệu	116
6.5.2. Xử lý theo từng cảm xúc	117
6.5.3. Áp dụng Hybrid Algorithm.....	118
6.5.4. Tạo và lưu trữ recommendations vào Redis	119
6.6. Truy xuất Recommendation	119
CHƯƠNG 7. KẾT LUẬN.....	123
7.1. Kết quả đồ án	123
7.2. Ưu điểm và hạn chế	124
7.2.1. Ưu điểm.....	124
7.2.2. Hạn chế	124
7.3. Hướng phát triển	125
TÀI LIỆU THAM KHẢO	126

DANH SÁCH HÌNH ẢNH

<i>Hình 2.1. Cấu trúc mạng RBM.....</i>	18
<i>Hình 2.2. ReLU activation function</i>	18
<i>Hình 2.3. RBM backward pass.....</i>	19
<i>Hình 2.4. RBM forward pass.....</i>	19
<i>Hình 2.5. Logo Typescript.....</i>	23
<i>Hình 2.6. Logo ReactJS.....</i>	24
<i>Hình 2.7. Logo NodeJS</i>	25
<i>Hình 2.8. Logo PostgreSQL</i>	26
<i>Hình 2.9. Mô hình Client-Server.....</i>	28
<i>Hình 2.10. Điểm lỗi duy nhất</i>	28
<i>Hình 3.1. Lược đồ use case</i>	45
<i>Hình 3.2. Sơ đồ kiến trúc hệ thống.....</i>	72
<i>Hình 3.3. Sơ đồ cơ sở dữ liệu</i>	77

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1. Lý do chọn đề tài

Trong những năm gần đây, cùng với sự phát triển nhanh chóng của trí tuệ nhân tạo và học máy, các ứng dụng nhận diện cảm xúc ngày càng được nghiên cứu và ứng dụng rộng rãi trong nhiều lĩnh vực như chăm sóc sức khỏe, giáo dục, thương mại điện tử và đặc biệt là lĩnh vực giải trí. Âm nhạc, với vai trò là một phương tiện truyền tải và điều tiết cảm xúc con người, đã chứng minh được sự gắn kết mật thiết với trạng thái tâm lý của người nghe. Tuy nhiên, hầu hết các nền tảng nghe nhạc hiện nay chỉ dừng lại ở việc gợi ý bài hát dựa trên lịch sử nghe hoặc sở thích chủ quan được thiết lập sẵn, chưa thực sự khai thác được yếu tố cảm xúc thời điểm thực của người dùng. Từ thực tế đó, nhóm chúng em nhận thấy rằng việc xây dựng một hệ thống gợi ý âm nhạc dựa trên cảm nhận diện từ khuôn mặt thông qua webcam là một hướng tiếp cận mới mẻ, mang tính ứng dụng cao, có thể góp phần cá nhân hóa trải nghiệm âm nhạc một cách tự nhiên và thông minh hơn.

Bên cạnh đó, với sự phổ biến của các thiết bị có tích hợp camera như laptop, điện thoại thông minh hay máy tính bảng, việc sử dụng webcam để thu nhận hình ảnh khuôn mặt người dùng và phân tích cảm xúc trở nên dễ dàng, thuận tiện và không gây cản trở cho trải nghiệm của người sử dụng. Hệ thống này không chỉ phù hợp với xu hướng phát triển công nghệ hiện nay mà còn có khả năng mở rộng, tích hợp vào nhiều nền tảng khác nhau như ứng dụng di động, trình phát nhạc trực tuyến hoặc các thiết bị IoT. Ngoài yếu tố công nghệ, đề tài này còn mang tính nhân văn cao khi âm nhạc được sử dụng như một công cụ hỗ trợ điều tiết cảm xúc, giúp người dùng thư giãn, giải tỏa căng thẳng hoặc tìm kiếm sự đồng cảm trong những thời điểm tâm lý nhất định.

Xuất phát từ những lý do trên, nhóm em quyết định chọn đề tài “Xây dựng hệ thống gợi ý âm nhạc thông minh dựa trên nhận diện cảm xúc từ camera” nhằm tìm hiểu, nghiên cứu và hiện thực hóa ý tưởng kết hợp giữa công nghệ nhận diện cảm xúc và hệ thống đề xuất âm nhạc. Đây không chỉ là một thử thách kỹ thuật thú vị, mà còn là cơ hội để nhóm vận dụng kiến thức đã học, đồng thời tiếp cận với các công nghệ mới trong lĩnh vực xử lý ảnh, học sâu và xây dựng hệ thống gợi ý thông minh, từ đó nâng cao kỹ năng và kinh nghiệm thực tiễn cho các thành viên trong nhóm.

1.2. Mục tiêu

- Xây dựng module nhận diện cảm xúc người dùng qua hình ảnh khuôn mặt thu từ camera theo thời gian thực.
- Áp dụng mô hình học sâu để phân loại các trạng thái cảm xúc (ví dụ: vui, buồn, tức giận, ngạc nhiên, v.v.).
- Thiết kế cơ sở dữ liệu âm nhạc được gắn nhãn theo cảm xúc để phục vụ quá trình gợi ý bài hát tương ứng.
- Triển khai cơ chế kết nối giữa kết quả phân loại cảm xúc và việc chọn lọc/gợi ý nhạc, đảm bảo độ chính xác và tốc độ phản hồi.
- Thiết kế giao diện đơn giản, trực quan, giúp người dùng dễ dàng tương tác và nhận được gợi ý âm nhạc phù hợp ngay lập tức.

1.3. Phạm vi

- **Phạm vi nhận diện cảm xúc:** Ứng dụng chỉ tập trung nhận diện cảm xúc của người dùng thông qua hình ảnh khuôn mặt được thu từ webcam, không xét đến giọng nói hay ngôn ngữ cơ thể. Các cảm xúc được phân loại thuộc nhóm 7 cảm xúc cơ bản: vui (happy), buồn (sad), tức giận (angry), ngạc nhiên (surprised), bình thường (neutral), sợ hãi (feared), ghê tởm (disgusted).
- **Phạm vi gợi ý âm nhạc:** Danh sách bài hát được sử dụng trong hệ thống là tập dữ liệu cố định do nhóm thực hiện thu thập và phân loại thủ công theo từng loại

cảm xúc. Hệ thống không tích hợp với các nền tảng phát nhạc trực tuyến như Spotify hay YouTube. Để gán nhãn cảm xúc cho kho nhạc, để tài ứng dụng các mô hình học máy. Các mô hình này được huấn luyện để tự động phân loại cảm xúc của một bài hát dựa trên các đặc trưng âm học được trích xuất từ file âm thanh.

- **Phạm vi người dùng:** Ứng dụng hướng đến người dùng phổ thông có thiết bị hỗ trợ webcam, sử dụng với mục đích giải trí cá nhân. Triển khai các chức năng nâng cao như lưu lịch sử cảm xúc hay cá nhân hóa theo thói quen nghe nhạc dài hạn bằng cách áp dụng mô hình Hybrid Recommender System kết hợp giữa Content Based Filtering và Collaborative Filtering. Bằng cách này hệ thống có thể học được xu hướng liên quan giữa cảm xúc - bài hát muốn nghe của người dùng dựa trên lịch sử cảm xúc khi nghe nhạc của họ, từ đó cá nhân hóa cách thức gợi ý nhạc từ cảm xúc theo từng người dùng khác nhau, không cần áp đặt quy luật gợi ý theo một quy chuẩn chung cụ thể cho toàn bộ người dùng.
- **Phạm vi kỹ thuật:** Ứng dụng được phát triển trên nền web, sử dụng các công nghệ như face-api.js cho xử lý ảnh và mô hình AI, cùng với ReactJS cho phần front-end. Hệ thống chạy trên máy tính cá nhân, chưa triển khai lên môi trường server thực tế hoặc thiết bị di động.

1.4. Đối tượng sử dụng

Đối tượng sử dụng của ứng dụng là người dùng phổ thông có nhu cầu nghe nhạc phù hợp với tâm trạng trong thời gian thực, đặc biệt là những người yêu thích công nghệ và mong muốn trải nghiệm các tiện ích thông minh, cá nhân hóa. Ứng dụng cũng có thể hỗ trợ tốt cho các cá nhân thường xuyên làm việc trong môi trường căng thẳng, cần thư giãn tinh thần thông qua âm nhạc. Bên cạnh đó, sản phẩm còn phù hợp với các nhà phát triển, nhà nghiên cứu hoặc sinh viên trong lĩnh vực trí tuệ

nhân tạo, xử lý ảnh và hệ thống gợi ý – như một công cụ tham khảo hoặc thử nghiệm các mô hình nhận diện cảm xúc và đề xuất nội dung theo ngữ cảnh.

1.5. Phương pháp thực hiện

- Tìm hiểu công nghệ: ReactJS, MongoDB, ExpressJS, ...
- Quản lý source code: Github
- Trải nghiệm và phân tích các ứng dụng tương tự trên thị trường
- Phân tích và xác định yêu cầu
- Thiết kế
 - Thiết kế đối tượng
 - Thiết kế dữ liệu
 - Thiết kế giao diện
- Cài đặt
- Kiểm thử
- Hoàn thiện sản phẩm
- Báo cáo

1.6. Công nghệ sử dụng

- Công cụ thiết kế UI/UX: Figma
- Thư viện hỗ trợ: face-api.js
- Front – end:
 - + Ngôn ngữ sử dụng: TypeScript
 - + Thư viện sử dụng: React JS, TailwindCSS, ShadCN
- Back – end:
 - + Ngôn ngữ sử dụng: TypeScript, Python
 - + Thư viện/Framework sử dụng: Express JS và Node JS

Service chính: Được phát triển bằng Node.js và framework Express.js. Service này chịu trách nhiệm xử lý các API request từ front-end, quản lý logic nghiệp vụ cơ bản và tương tác với cơ sở dữ liệu.

Recommender Service: Được xây dựng như một service riêng biệt bằng ngôn ngữ Python, tận dụng các thư viện khoa học dữ liệu mạnh mẽ như Pandas, scikit-learn, và Surprise, Tensorflow để huấn luyện mô hình và tính toán danh sách gợi ý..

- Cơ sở dữ liệu:

- + PostgreSQL: Được sử dụng làm cơ sở dữ liệu quan hệ chính, lưu trữ các dữ liệu có cấu trúc và cần tính toán cao như thông tin người dùng, thông tin bài hát, và dữ liệu tương tác cảm xúc-âm nhạc.
- + Redis: Được sử dụng làm hệ thống cache (bộ nhớ đệm). Các danh sách bài hát gợi ý sau khi được Recommender Service tính toán sẽ được lưu trữ tại đây để giảm độ trễ và tăng tốc độ phản hồi cho người dùng.

- Quản lý mã nguồn: Github

1.7. Kết quả mong đợi

- Xây dựng được một hệ thống có khả năng nhận diện cảm xúc của người dùng thông qua hình ảnh khuôn mặt thu từ webcam.
- Gợi ý được các bài hát phù hợp với từng trạng thái cảm xúc nhận diện được, đảm bảo tính hợp lý và phản hồi nhanh.
- Học được tương quan giữa tính cách từng người dùng cụ thể đối với loại nhạc mà họ muốn nghe khi có cảm xúc đó. Ví dụ người A thường sẽ nghe nhạc vui khi buồn, còn người B lại muốn nghe nhạc buồn khi vui,... Từ đó cá nhân hóa được cách thức gợi ý cho từng người dùng riêng biệt.
- Hệ thống hoạt động ổn định, giao diện trực quan, dễ sử dụng đối với người dùng phổ thông.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1. Nền tảng lý thuyết

2.1.1. Nhận diện cảm xúc

2.1.1.1. Khái niệm về nhận diện cảm xúc qua khuôn mặt

Nhận diện cảm xúc qua khuôn mặt (Facial Emotion Recognition – FER) là một lĩnh vực trong trí tuệ nhân tạo và thị giác máy tính, tập trung vào việc tự động xác định trạng thái cảm xúc của con người thông qua các biểu hiện khuôn mặt. Hệ thống FER thường sử dụng hình ảnh hoặc video làm đầu vào để phát hiện và phân tích các đặc trưng khuôn mặt, từ đó suy luận cảm xúc tương ứng.

Cảm xúc là một thành phần quan trọng trong giao tiếp của con người, không chỉ được thể hiện qua lời nói mà còn thông qua nét mặt, ánh mắt, cử chỉ và giọng điệu. Trong đó, biểu hiện khuôn mặt được xem là kênh trực quan và phổ biến nhất để phản ánh trạng thái cảm xúc. Nghiên cứu của nhà tâm lý học Paul Ekman đã chỉ ra rằng có một tập hợp các cảm xúc cơ bản mang tính phổ quát, bao gồm: vui vẻ (happy), buồn bã (sad), tức giận (angry), sợ hãi (fear), ngạc nhiên (surprised), ghê tởm (disgust) và trung tính (neutral). Các cảm xúc này có thể được nhận biết thông qua sự thay đổi ở các vùng cơ mặt như lông mày, mắt, miệng, v.v.

Việc xây dựng hệ thống nhận diện cảm xúc qua khuôn mặt đòi hỏi sự kết hợp giữa nhiều lĩnh vực, bao gồm xử lý ảnh, học máy, học sâu và mô hình hóa hành vi con người. Các ứng dụng thực tiễn của FER rất đa dạng, có thể kể đến như:

- **Hỗ trợ giảng dạy thông minh:** theo dõi cảm xúc học sinh để điều chỉnh nội dung giảng dạy phù hợp.
- **Dịch vụ khách hàng:** đánh giá cảm xúc người dùng trong quá trình tương tác với hệ thống.
- **An ninh và giám sát:** phát hiện trạng thái nghi ngờ hoặc căng thẳng.

- **Giải trí và cá nhân hóa nội dung:** điều chỉnh để xuất nội dung (phim, nhạc, quảng cáo) theo cảm xúc người dùng.

Trong khuôn khổ đồ án này, nhận diện cảm xúc giữ vai trò trung tâm, là tiền đề để đưa ra các gợi ý bài hát phù hợp với tâm trạng hiện tại của người dùng, nhằm tạo ra trải nghiệm âm nhạc mang tính cá nhân hóa và tương tác cao hơn.

2.1.1.2. Nguyên lý hoạt động của hệ thống nhận diện cảm xúc

Hệ thống nhận diện cảm xúc từ khuôn mặt người hoạt động thông qua một chuỗi các bước xử lý ảnh, trích xuất đặc trưng và phân loại cảm xúc dựa trên biểu hiện khuôn mặt thu được từ camera. Quy trình này bao gồm năm giai đoạn chính: thu nhận hình ảnh đầu vào, phát hiện khuôn mặt, trích xuất đặc trưng khuôn mặt, phân loại cảm xúc và xử lý kết quả đầu ra.

1. Thu nhận hình ảnh đầu vào

Hệ thống sử dụng webcam tích hợp trên thiết bị của người dùng để ghi nhận hình ảnh khuôn mặt theo thời gian thực. Dữ liệu đầu vào là các khung hình (frames) được trích xuất liên tục từ luồng video nhằm đảm bảo phản hồi tức thời với sự thay đổi trong biểu cảm của người dùng. Mỗi khung hình sẽ được đưa qua các bước xử lý tiếp theo để phân tích cảm xúc tại thời điểm đó.

2. Phát hiện khuôn mặt (Face Detection)

Trong mỗi khung hình, hệ thống cần xác định xem khuôn mặt người dùng nằm ở vị trí nào. Đây là một bước quan trọng, bởi các bước tiếp theo chỉ được thực hiện trên vùng chứa khuôn mặt. Kỹ thuật phát hiện khuôn mặt được triển khai thông qua các mô hình học sâu được huấn luyện trước, có khả năng định vị chính xác khuôn mặt trong điều kiện ánh sáng và góc nhìn khác nhau. Trong đồ án này, mô hình **TinyFaceDetector** được tích hợp trong thư viện **face-api.js** được sử dụng để phát hiện khuôn mặt với tốc độ

cao và độ chính xác đủ dùng, phù hợp cho ứng dụng chạy trên trình duyệt web.

3. Trích xuất đặc trưng khuôn mặt (Facial Feature Extraction)

Sau khi định vị được khuôn mặt, hệ thống tiếp tục xác định các điểm đặc trưng quan trọng trên khuôn mặt (facial landmarks), chẳng hạn như vị trí của mắt, mũi, miệng, lông mày và cằm. Các đặc trưng này phản ánh các chuyển động cơ mặt đặc trưng của từng loại cảm xúc. Trong đồ án, mô hình **FaceLandmark68Net** từ **face-api.js** được sử dụng để trích xuất 68 điểm mốc trên khuôn mặt. Thông tin này là nền tảng để đánh giá sự biến đổi của khuôn mặt và diễn giải trạng thái cảm xúc.

4. Phân loại cảm xúc (Emotion Classification)

Với các đặc trưng hình học của khuôn mặt đã được xác định, hệ thống sẽ thực hiện phân loại cảm xúc thông qua một mô hình học sâu được huấn luyện trước. Mô hình **FaceExpressionNet** trong thư viện **face-api.js** là mô hình được sử dụng trong đồ án, có khả năng nhận diện bảy loại cảm xúc cơ bản:

- **Vui vẻ (happy)**
- **Buồn bã (sad)**
- **Tức giận (angry)**
- **Ngạc nhiên (surprised)**
- **Sợ hãi (fearful)**
- **Ghê tởm (disgusted)**
- **Trung tính (neutral).**

Mô hình này trả về một vector chứa xác suất ứng với từng loại cảm xúc. Hệ thống sẽ chọn cảm xúc có xác suất cao nhất làm đầu ra cuối cùng.

5. Xử lý kết quả và tích hợp với hệ thống gợi ý âm nhạc

Kết quả phân loại cảm xúc sau khi được xác định sẽ được gửi đến hệ thống gợi ý âm nhạc. Đây là một mô-đun riêng biệt (triển khai bằng FastAPI) có chức năng ánh xạ các trạng thái cảm xúc sang danh sách các bài hát phù hợp. Ví dụ, nếu cảm xúc được phát hiện là "sad", hệ thống sẽ đề xuất các bài nhạc nhẹ nhàng, mang tính xoa dịu. Nếu cảm xúc là "happy", hệ thống sẽ đề xuất các bài nhạc sôi động, tích cực.

Toàn bộ quá trình nhận diện cảm xúc được thực hiện trực tiếp trên trình duyệt phía người dùng (client-side) nhằm đảm bảo độ trễ thấp, phản hồi tức thì và tránh rủi ro về bảo mật và quyền riêng tư. Điều này cho phép ứng dụng vận hành hiệu quả mà không cần gửi hình ảnh khuôn mặt về máy chủ, đồng thời vẫn đảm bảo được chất lượng phân tích cảm xúc trong thời gian thực.

2.1.1.3. Giới thiệu thư viện face-api.js

face-api.js là một thư viện mã nguồn mở được phát triển dựa trên nền tảng TensorFlow.js, cho phép triển khai các mô hình học sâu (deep learning) xử lý khuôn mặt trực tiếp trong trình duyệt web. Đây là một giải pháp toàn diện cho các tác vụ liên quan đến nhận diện khuôn mặt như phát hiện khuôn mặt, nhận diện danh tính, phân tích biểu cảm cảm xúc, trích xuất điểm đặc trưng (facial landmarks), và ước lượng hướng nhìn.

Thư viện được viết hoàn toàn bằng JavaScript, hoạt động trên nền tảng WebGL thông qua TensorFlow.js, cho phép khai thác GPU của máy client để tăng tốc xử lý mô hình. Nhờ đó, face-api.js có thể thực hiện nhận diện khuôn mặt theo thời gian thực ngay trên trình duyệt mà không cần đến hệ thống xử lý phía máy chủ.

Trong đồ án này, nhóm sử dụng face-api.js để triển khai tính năng nhận diện cảm xúc từ webcam. Quá trình này bao gồm việc phát hiện khuôn mặt từ

luồng video, trích xuất đặc trưng khuôn mặt, và cuối cùng phân tích biểu cảm để suy luận cảm xúc của người dùng.

face-api.js tích hợp sẵn nhiều mô hình được huấn luyện trước (pretrained models), có thể được tải động từ các tệp **.bin**. Trong ứng dụng của đồ án, ba mô hình chính được sử dụng gồm:

- **Tiny Face Detector (tinyFaceDetector):** Đây là mô hình dùng để phát hiện khuôn mặt trong ảnh hoặc video. Nó được thiết kế tối ưu về tốc độ và kích thước mô hình, phù hợp để sử dụng trên các thiết bị có tài nguyên hạn chế. Mô hình nhận đầu vào là hình ảnh/video và trả về tọa độ hộp giới hạn (bounding box) của khuôn mặt.
- **Face Landmark Detection (faceLandmark68Net):** Mô hình này có khả năng xác định 68 điểm đặc trưng trên khuôn mặt (bao gồm các vị trí của mắt, mũi, miệng, cằm...). Các điểm này được sử dụng để hiểu cấu trúc khuôn mặt, phục vụ các tác vụ nâng cao như ước lượng chuyển động khuôn mặt hoặc phân tích cảm xúc.
- **Facial Expression Recognition (faceExpressionNet):** Mô hình phân loại cảm xúc này được huấn luyện để phát hiện bảy trạng thái cảm xúc cơ bản gồm: **happy, sad, angry, fearful, disgusted, surprised, và neutral**. Đầu ra của mô hình là một vector xác suất, biểu thị mức độ tương ứng với từng cảm xúc. Cảm xúc có xác suất cao nhất sẽ được chọn làm kết quả đầu ra.

Việc sử dụng face-api.js cho phép thực hiện toàn bộ quá trình nhận diện cảm xúc ngay trên trình duyệt, không cần truyền hình ảnh qua mạng. Mô hình được tải từ backend hoặc từ thư mục tĩnh trên server frontend, và được lưu tạm trên bộ nhớ máy client để phục vụ nhiều lần gọi. Khung hình từ webcam

được xử lý liên tục mỗi 100–300ms nhằm đảm bảo khả năng phản hồi thời gian thực.

2.1.1.4. Ưu nhược điểm của face-api.js

Trong quá trình lựa chọn công cụ cho tác vụ nhận diện cảm xúc trong thời gian thực từ webcam, nhóm đã tiến hành khảo sát và thử nghiệm một số giải pháp như OpenCV, MediaPipe và face-api.js. Sau khi so sánh, nhóm quyết định sử dụng face-api.js nhờ sự phù hợp về mặt công nghệ, dễ tích hợp với React, cũng như khả năng xử lý trực tiếp trên trình duyệt. Dưới đây là những đánh giá chi tiết về ưu và nhược điểm của thư viện này:

Ưu điểm:

- **Chạy trực tiếp trên trình duyệt:** face-api.js hoạt động hoàn toàn phía client nhờ vào TensorFlow.js, không cần backend xử lý hình ảnh, giúp giảm tải cho máy chủ và bảo vệ quyền riêng tư hình ảnh của người dùng.
- **Tích hợp đơn giản với React và các framework hiện đại:** thư viện được viết bằng JavaScript, dễ dàng tích hợp vào ứng dụng React (hoặc bất kỳ frontend framework nào khác). Ngoài ra, các API rõ ràng, hỗ trợ Promise và async/await giúp đồng bộ dễ dàng với luồng xử lý của ứng dụng.
- **Không cần huấn luyện mô hình:** các mô hình đã được huấn luyện sẵn và có thể tải về sử dụng ngay. Điều này phù hợp với các đồ án có thời gian hạn chế, giúp nhóm có thể tập trung vào tích hợp và xây dựng tính năng gợi ý nhạc.
- **Hỗ trợ nhiều tác vụ khuôn mặt trong một thư viện duy nhất:** face-api.js không chỉ hỗ trợ nhận diện cảm xúc mà còn có thể nhận diện danh tính, phát hiện khuôn mặt, ước lượng tuổi và giới

tính, trích xuất điểm đặc trưng, v.v. Nhờ đó, hệ thống có thể mở rộng tính năng trong tương lai một cách linh hoạt.

Nhược điểm:

- **Phụ thuộc nhiều vào phần cứng client:** do mọi xử lý diễn ra trên trình duyệt người dùng, hiệu năng sẽ bị ảnh hưởng nếu thiết bị yếu, dẫn đến tốc độ nhận diện chậm hoặc giật hình.
- **Chưa hỗ trợ tốt trên thiết bị di động:** mặc dù có thể hoạt động trên mobile, nhưng tốc độ xử lý thường thấp hơn đáng kể so với desktop. Điều này ảnh hưởng đến tính khả dụng nếu triển khai ứng dụng dưới dạng Progressive Web App hoặc web mobile.
- **Mô hình nhận diện cảm xúc đơn giản:** mô hình nhận diện cảm xúc trong face-api.js chỉ phân loại 7 cảm xúc cơ bản, chưa đủ tinh vi để nhận biết những trạng thái cảm xúc phức tạp hơn như lǎng, phấn khích, bối rối, mệt mỏi,... Điều này có thể làm giảm chất lượng gợi ý âm nhạc trong một số trường hợp.
- **Không hỗ trợ mô hình tùy chỉnh:** face-api.js hiện không cung cấp cơ chế dễ dàng để thay thế mô hình nhận diện cảm xúc bằng mô hình tự huấn luyện (custom-trained). Việc tùy biến đòi hỏi phải chỉnh sửa thư viện hoặc chuyển sang sử dụng TensorFlow.js trực tiếp.

2.1.1.5. Kết luận

Nhận diện cảm xúc qua khuôn mặt là một trong những lĩnh vực quan trọng và đang được ứng dụng rộng rãi trong các hệ thống tương tác người–máy hiện đại. Việc tìm hiểu khái niệm, nguyên lý hoạt động cũng như lựa chọn công cụ phù hợp là nền tảng thiết yếu để hiện thực hóa chức năng này trong đồ án.

Sau quá trình khảo sát và thử nghiệm, nhóm quyết định lựa chọn **face-api.js** làm thư viện chính để triển khai tính năng nhận diện cảm xúc. Lý do chọn thư viện này là vì khả năng xử lý trực tiếp trên trình duyệt thông qua TensorFlow.js, giúp giảm tải cho backend, đồng thời đảm bảo quyền riêng tư cho người dùng khi không cần truyền dữ liệu hình ảnh lên máy chủ. Bên cạnh đó, face-api.js có thể tích hợp dễ dàng với frontend sử dụng React và TypeScript, có sẵn các mô hình huấn luyện, hỗ trợ nhận diện bảy cảm xúc cơ bản, phù hợp với phạm vi và mục tiêu của đồ án.

Mặc dù còn tồn tại một số hạn chế như phụ thuộc vào hiệu năng thiết bị client và độ phân giải cảm xúc ở mức cơ bản, face-api.js vẫn là giải pháp hợp lý, cân bằng giữa tính khả dụng, hiệu năng và thời gian phát triển. Việc lựa chọn công cụ này giúp nhóm tập trung nguồn lực vào việc tích hợp tính năng gợi ý nhạc theo cảm xúc – mục tiêu chính của hệ thống.

2.1.2. Gợi ý âm nhạc

Hệ thống gợi ý âm nhạc của ứng dụng được thiết kế dựa trên một kiến trúc lai (Hybrid), kết hợp giữa các quy trình xử lý Offline để chuẩn bị dữ liệu và huấn luyện mô hình, và các quy trình xử lý Real-time để tương tác với người dùng và phục vụ gợi ý. Nguyên lý hoạt động của hệ thống được chia thành các giai đoạn chính như sau:

2.1.2.1. Giai đoạn 1: Phân loại và Gán nhãn Cảm xúc cho Âm nhạc (Xử lý Offline)

Đây là bước tiền xử lý dữ liệu cốt lõi, hoạt động một cách tự động và định kỳ:

Mục đích: Làm giàu dữ liệu cho kho nhạc, biến mỗi bài hát thành một đối tượng có thể được gợi ý dựa trên thuộc tính cảm xúc.

Cơ chế hoạt động: Một Cron Job được lập lịch chạy định kỳ bằng ngôn ngữ Python. Tác vụ này sẽ quét qua cơ sở dữ liệu (database) các bài hát, sử dụng một mô hình học máy đã được huấn luyện trước (ví dụ: Random Forest) để trích xuất các đặc trưng âm học và tự động gán nhãn cảm xúc (Vui, Buồn, Năng lượng, v.v.) cho những bài hát chưa được phân loại. Kết quả được lưu trực tiếp vào cơ sở dữ liệu chính (main_db).

2.1.2.2. Giai đoạn 2: Thu thập Cảm xúc và Tương tác Người dùng (Xử lý Real-time)

Giai đoạn này diễn ra khi người dùng tương tác trực tiếp với ứng dụng:

- + Thu thập cảm xúc: Giao diện Front-end sử dụng camera và thư viện face-api.js để liên tục nhận diện cảm xúc hiện tại của người dùng.
- + Gửi dữ liệu: Khi một cảm xúc được xác định, Front-end sẽ gửi một gói dữ liệu đến Back-end (emotion_collect_service). Gói dữ liệu này bao gồm: userId (định danh người dùng), emotion (cảm xúc hiện tại), và musicId (nếu người dùng đang nghe một bài hát cụ thể), và cả duration_percentage (tức phần trăm hoàn thành bài hát ở thời điểm gửi request, ví dụ 70 tức 70% bài hát đã được phát, mục đích là để tính toán recommend score dựa trên % hoàn thành bài hát trước khi thực hiện skip bài, next bài,...)
- + Lưu trữ trạng thái: Back-end ngay lập tức lưu cặp (userId, emotion) vào Redis. Việc này giúp hệ thống luôn nắm bắt được trạng thái cảm xúc mới nhất của người dùng để phục vụ cho các yêu cầu gợi ý tức thì.
- + Tính toán điểm tương tác: Đồng thời, Back-end áp dụng một công thức tính điểm (scoring formula) để lượng hóa mức độ yêu thích của người dùng đối với bài hát đang nghe (musicId) dưới trạng thái cảm xúc (emotion) đó. Điểm số này

(score) cùng với userId, musicId, emotion được ghi vào bảng emotion_recommender trong cơ sở dữ liệu emotion_collect_db. Bảng dữ liệu này chính là để huấn luyện mô hình gợi ý ở giai đoạn sau.

2.1.2.3. Giai đoạn 3: Huấn luyện Mô hình và Tạo Gợi ý (Xử lý Offline theo đợt)

Đây là trái tim của hệ thống, nơi trí tuệ nhân tạo được áp dụng để tạo ra các gợi ý cá nhân hóa:

- + **Cơ chế hoạt động:** Một service riêng biệt (recommender_service) được viết bằng Python sẽ được kích hoạt định kỳ (ví dụ: mỗi 5 phút).
- + **Thu thập dữ liệu:** Service này kết nối đến cả hai cơ sở dữ liệu: đọc dữ liệu tương tác (userId, musicId, emotion, score) từ emotion_collect_db và đọc thông tin chi tiết của bài hát (thể loại, nghệ sĩ,...) từ main_db.
- + **Xây dựng mô hình Hybrid:** Dữ liệu được phân tách theo từng loại cảm xúc. Với mỗi cảm xúc, một mô hình gợi ý Hybrid riêng biệt được huấn luyện, kết hợp giữa:
 - Lọc cộng tác (Collaborative Filtering): Sử dụng thuật toán RBM (Restricted Boltzmann Machine) và k-NN (k-Nearest Neighbors) trên dữ liệu (userId, musicId, score) để tìm ra những người dùng có cùng gu âm nhạc khi ở cùng một trạng thái cảm xúc.
 - Lọc dựa trên nội dung (Content-Based Filtering): Sử dụng các thuộc tính của bài hát (thể loại, nghệ sĩ, cảm xúc đã gán nhãn ở Giai đoạn 1) để tìm các bài hát tương tự.

Lưu trữ kết quả: Sau khi huấn luyện, service sẽ tạo ra một danh sách các bài hát được gợi ý cho mỗi cặp (userId, emotion). Kết quả này được lưu trữ vào Redis dưới dạng key-value, với key là userId:emotion và value là một danh sách các

musicId. Việc tính toán trước và lưu vào cache này giúp tối ưu hóa tốc độ phản hồi ở giai đoạn cuối.

2.1.2.4. Giai đoạn 4: Phục vụ Gợi ý và Cơ chế Dự phòng (Xử lý Real-time)

Đây là giai đoạn cuối cùng, khi người dùng yêu cầu một danh sách phát mới:

- Người dùng yêu cầu gợi ý nhạc. Back-end (main_service) lấy userId và trạng thái cảm xúc hiện tại của họ từ Redis (đã được lưu ở Giai đoạn 2).
- Hệ thống thực hiện một truy vấn cực nhanh vào Redis với key = userId:emotion để lấy danh sách các musicId đã được cá nhân hóa và tính toán trước ở Giai đoạn 3.
- Cơ chế dự phòng (Fallback Strategy): Trong trường hợp không tìm thấy gợi ý trong Redis (ví dụ: người dùng mới, hoặc chưa có tương tác cho cảm xúc đó - vấn đề "Cold Start"), hệ thống sẽ kích hoạt cơ chế dự phòng theo các cấp độ ưu tiên:
 - + Cấp 1 - Gợi ý theo quy tắc (Rule-based): Áp dụng một bộ quy tắc ánh xạ cứng. Ví dụ, nếu người dùng đang buồn, hệ thống sẽ gợi ý các bài hát vui hoặc thư giãn (CHILL).
 - + Cấp 2 - Gợi ý theo độ phổ biến (Popularity-based): Nếu gợi ý theo quy tắc vẫn không đủ số lượng bài hát yêu cầu (top N), hệ thống sẽ lấp đầy danh sách bằng những bài hát được nghe nhiều nhất hoặc có điểm số cao nhất trong toàn bộ hệ thống.

2.1.3. Restricted Boltzmann Machine (RBM)

RBM ban đầu được đề xuất với tên gọi Harmonium bởi Paul Smolensky vào năm 1986, và trở nên nổi bật sau khi Geoffrey Hinton cùng các cộng sự áp dụng các thuật toán học nhanh cho chúng vào giữa những năm 2000. RBM đã được ứng dụng trong nhiều lĩnh vực như giảm chiều dữ liệu, phân loại, bộ lọc cộng tác, học đặc trưng, mô hình hóa chủ đề, miễn dịch học, và thậm chí cả cơ học lượng tử nhiều vật thể.

Trong lĩnh vực recommender system, nó đã được sử dụng từ những năm 2007, khá lâu trước khi AI trở nên phổ biến như ngày nay. Tuy vậy, RBM vẫn là kỹ thuật được sử dụng trong hệ thống khuyến nghị ngày nay.

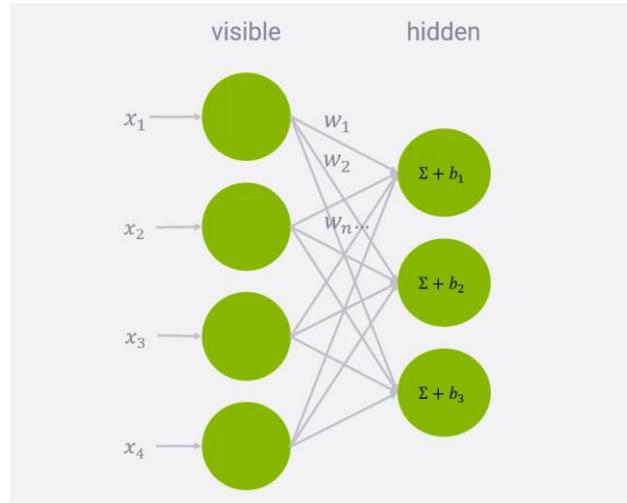
Tại Netflix Prize, sau khi cuộc thi diễn ra, Netflix nhận ra rằng Matrix Factorization (cụ thể là SVD) cùng với RBM đã cho kết quả tốt nhất ở cuộc thi này ở điểm số RMSE. Họ nhận ra rằng, việc kết hợp giữa Matrix Factorization và RBM đã cho ra kết quả tốt hơn đáng kể, từ 8.9 xuống còn 8.8 điểm RMSE (RMSE càng thấp tức là càng tốt). Vài năm về trước, Netflix xác nhận rằng họ vẫn đang sử dụng RBM như một phần trong hệ thống khuyến nghị mà họ đang triển khai ở môi trường production.

• Ý tưởng

Restricted Boltzmann Machine là một unsupervised deep learning model (mô hình học sâu không giám sát), trong đó mỗi nút được kết nối với tất cả các nút khác. Restricted Boltzmann Machine - RBM là mô hình đồ thị xác suất không hướng, bao gồm một lớp các biến quan sát và một lớp các biến ẩn. Không có nút nào trong cùng một lớp được kết nối với nhau, vì lý do này mà chúng được gọi là Restricted Boltzmann Machine. Chúng ta cũng có thể nghĩ về RBM như việc chuyển đổi dữ liệu đầu vào từ dimension sang các dimensions khác. Chúng chủ yếu được sử dụng như là các khối xây dựng cho Deep Network.

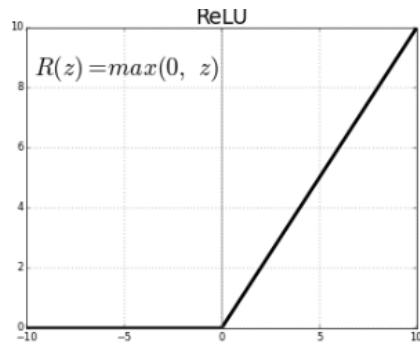
RBM chủ yếu được sử dụng như một mô hình sinh (generative model), nhưng chúng cũng có thể được sử dụng như một mô hình phân biệt (discriminative model). Để sử dụng RBM như một mô hình phân biệt, chúng ta có thể sử dụng các đặc trưng được học từ lớp ẩn làm đầu vào cho một mô hình phân biệt chuẩn.

Cấu trúc mạng RBM:



Hình 2.1. Cấu trúc mạng RBM

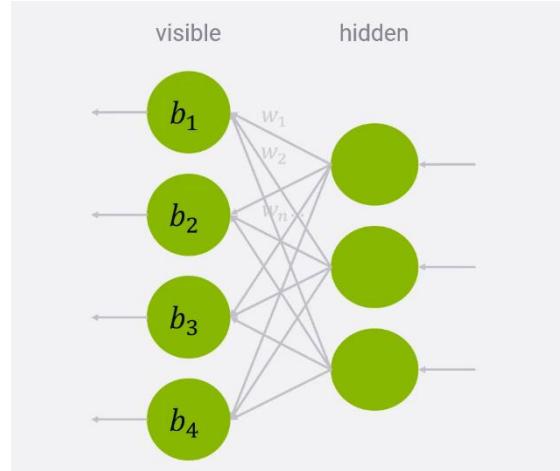
RBM gồm có 2 lớp: visible layer và hidden layer. Mô hình sẽ được train bằng cách đưa dữ liệu training vào visible layer, sau đó thực hiện train và tính toán weight, bias giữa chúng. Một activation function như ReLU sẽ được dùng để tạo ra output cho mỗi neuron ở hidden layer:



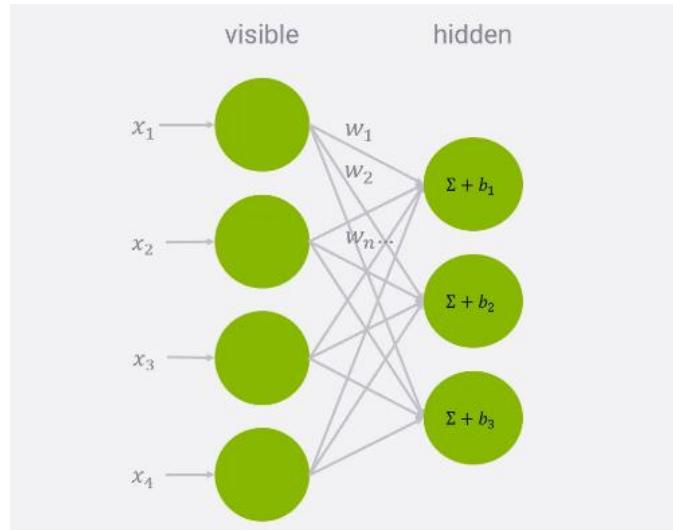
Hình 2.2. ReLU activation function

Từ cấu trúc mạng, ta có thể nói rằng mô hình này được gọi là *Restricted Boltzmann Machine* bởi vì giao tiếp giữa các neuron trong cùng một layer bị *Restricted* (hạn chế) bởi vì neuron trong layer này chỉ có thể nối tới các neuron của tầng khác mà thôi và không thể nối các neuron trong cùng một layer.

Quá trình train một RBM model sẽ diễn ra tương tự các mạng neural network thông thường: thực hiện forward pass và backward pass lặp đi lặp lại ở mỗi epoch để tìm ra weight và bias hợp lý, với điều kiện sao cho evaluate result là thấp nhất có thể.



Hình 2.3. RBM backward pass



Hình 2.4. RBM forward pass

2.1.4. KNN

KNN (K-Nearest Neighbors): là một thuật toán học máy không tham số (*non-parametric*), có thể được sử dụng để tìm ra các mục tương tự nhất (nearest neighbors) dựa trên độ tương đồng đã được tính toán, từ đó gợi ý các mục mà người dùng có thể quan tâm.

Bước 1: Chọn tham số K, tham số này có thể được thử nghiệm nhiều giá trị để chọn ra số K có kết quả gợi ý tốt nhất.

Bước 2: Sau khi đã có K, điểm dự đoán có thể được tính dựa trên trung bình trọng số của các điểm số từ K mục lân cận với công thức:

$$\text{Predicted Rating}(i) = \frac{\sum_{j \in N} (\text{Similarity}(i, j) \cdot \text{Rating}(j))}{\sum_{j \in N} |\text{Similarity}(i, j)|}$$

trong đó:

i : mục cần dự đoán

j : các mục lân cận K gần nhất

N : tập hợp K mục lân cận

$\text{Similarity}(i, j)$: Độ tương đồng giữa i và j

$\text{Rating}(j)$: Điểm đánh giá của người dùng cho mục j

Bước 3: Sau khi tính toán xong có thể xếp hạng các mục dựa trên điểm số dự đoán và đề xuất các mục có điểm dự đoán cao nhất cho người dùng.

2.1.5. Hybrid method

2.1.5.1. Giới thiệu

Hybrid method trong recommender system là sự kết hợp của nhiều kỹ thuật hoặc phương pháp khác nhau nhằm tận dụng ưu điểm của từng phương pháp và giảm thiểu các hạn chế của chúng. Các phương pháp kết hợp này thường kết hợp nhiều phương pháp lại với nhau như collaborative filtering và content-based filtering, hoặc kết hợp nhiều kỹ thuật khác như phân tích matrix factorization, và thậm chí các mô hình học sâu (deep learning).

Bằng cách tích hợp nhiều phương pháp, các hệ thống kết hợp (hybrid) có thể giảm thiểu những điểm yếu của từng phương pháp riêng biệt, chẳng hạn như cold-start problem. Các sự kết hợp phổ biến bao gồm collaborative filtering và content-based filtering. Có nhiều phương pháp được sử dụng để kết hợp các kỹ thuật này, chẳng hạn như phương pháp trọng số, chuyển đổi, kết hợp, hoặc phương pháp cấp độ meta.

Các hệ thống kết hợp thường cho ra các khuyến nghị chính xác hơn so với các hệ thống sử dụng một phương pháp duy nhất. Tuy nhiên, việc triển khai hiệu quả yêu cầu phải được tối ưu cẩn thận để cân bằng các thành phần và đảm bảo hiệu suất.

Đặc biệt, hybrid method cũng có thể hữu ích trong việc giải quyết cold start problem. Ý tưởng là trong giai đoạn đầu khi còn ít dữ liệu, việc sử dụng các mô hình Collaborative filtering sẽ rất khó khăn, do đó chúng ta có thể sẽ kết hợp Collaborative filtering với Content based filtering để sử dụng các đặc điểm của item từ ban đầu và gợi ý được cho user.

Ví dụ, nếu ứng dụng của chúng ta vừa triển khai và chưa có một lượng user đủ lớn, việc áp dụng Collaborative filtering gần như là không thể vì không có đủ dữ liệu đầu vào. Do đó ta có thể áp dụng Hybrid method để kết hợp Collaborative filtering với Content based, nhờ có content based mà ta có thể gợi ý cho user dựa theo đặc điểm sẵn

có của item mà không cần phải đợi đủ dữ liệu user. Sau này khi hệ thống có nhiều người dùng hơn, ta có thể điều chỉnh lại trọng số của Collaborative filtering để Hybrid method nghiên cứu về phương pháp này nhiều hơn.

Chúng tôi quyết định áp dụng Hybrid Method kết hợp giữa RBM và KNN để tối ưu hóa việc Recommend nhạc cho một người dùng dựa trên từng loại cảm xúc của người đó. Lý do lựa chọn phương pháp này đó là nhằm tận dụng được điểm mạnh của cả 2 loại recommend:

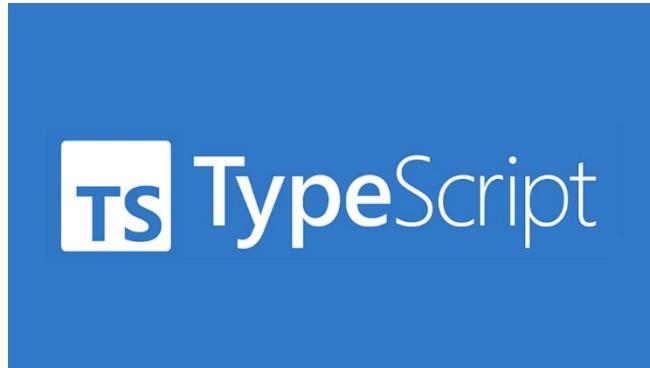
- + KNN: là một loại content based filtering, nhờ đó có thể gợi ý được các bài nhạc có sự tương đồng với bài nhạc mà người A đã nghe khi họ có cảm xúc B. Rất tốt để giải quyết cold start problem.
- + RBM: là một loại collaborative filtering, dùng để tận dụng lượng data từ chính những người dùng khác trên hệ thống, ví dụ nếu người A nghe bài hát B khi có cảm xúc C, hệ thống sẽ có thể gợi ý bài hát B cho người dùng D nếu họ có cùng cảm xúc C.

2.2.Công nghệ áp dụng

2.2.1. Typescript

2.2.1.1. Giới thiệu

TypeScript là một ngôn ngữ lập trình mã nguồn mở được phát triển bởi Microsoft. Nó là một siêu ngôn ngữ (superset) của JavaScript, nghĩa là tất cả mã JavaScript hợp lệ đều là mã TypeScript hợp lệ. TypeScript thêm vào các tính năng như kiểu tĩnh (static typing), Interface, và các tính năng hướng đối tượng khác, giúp tăng cường kiểm tra lỗi và bảo trì mã nguồn.



Hình 2.5. Logo Typescript

2.2.1.2. Ưu điểm

- Kiểm tra lỗi tĩnh: TypeScript cung cấp tính năng kiểm tra lỗi trước khi chạy chương trình, giúp phát hiện lỗi sớm và giảm thiểu các lỗi khi thực thi.
- Hỗ trợ IDE và công cụ tốt hơn: TypeScript cung cấp tính năng tự động hoàn thiện mã, hỗ trợ gỡ lỗi và các công cụ IDE mạnh mẽ hơn, giúp tăng năng suất lập trình viên.
- Cải thiện khả năng bảo trì: TypeScript hỗ trợ các khái niệm hướng đối tượng như class, interface, và module, giúp mã nguồn dễ hiểu và dễ bảo trì hơn trong các dự án lớn.

2.2.1.3. Nhược điểm

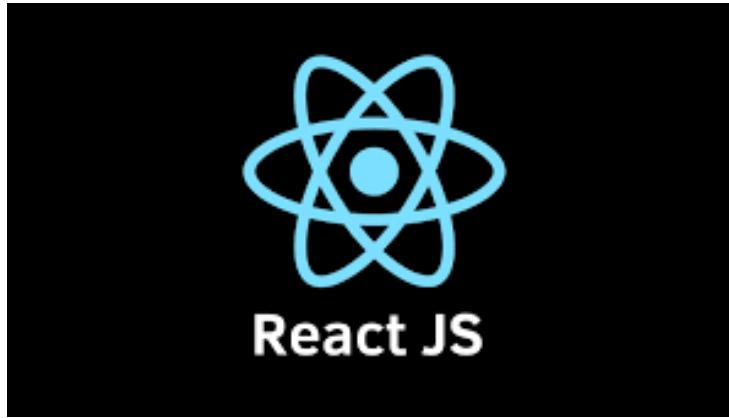
- Tăng thời gian học tập: TypeScript có thể khó tiếp cận đối với các lập trình viên mới vì nó yêu cầu kiến thức về JavaScript và kiểu dữ liệu tĩnh.
- Yêu cầu biên dịch: TypeScript cần được biên dịch thành JavaScript trước khi chạy, điều này có thể làm tăng thêm thời gian xây dựng và triển khai.

2.2.2. ReactJS

2.2.2.1. Giới thiệu

ReactJS là một thư viện JavaScript mã nguồn mở dùng để xây dựng giao diện người dùng, đặc biệt là các ứng dụng trang đơn (SPA). React.js được phát triển bởi

Facebook và được thiết kế để tăng tốc quá trình phát triển giao diện web thông qua việc sử dụng các component.



Hình 2.6. Logo ReactJS

2.2.2.2. Ưu điểm

- Hiệu suất cao: ReactJS sử dụng Virtual DOM, giúp tăng hiệu suất cập nhật giao diện và cải thiện trải nghiệm người dùng.
- Tái sử dụng component: ReactJS cho phép tái sử dụng các thành phần giao diện, giúp giảm thiểu thời gian phát triển và bảo trì.
- Cộng đồng phát triển mạnh mẽ: ReactJS có một cộng đồng lớn và đang phát triển nhanh chóng, cung cấp nhiều thư viện, plugin và tài liệu học tập.

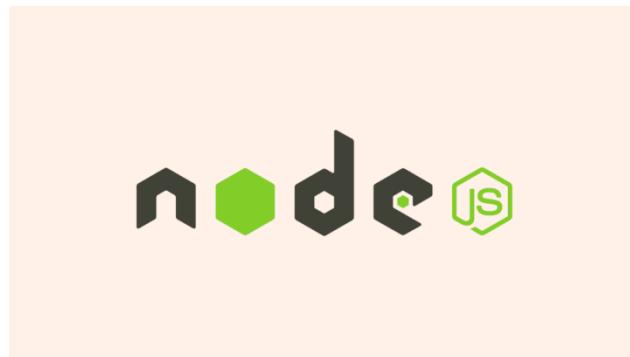
2.2.2.3. Nhược điểm

- Phức tạp cho người mới bắt đầu: Việc học và sử dụng ReactJS đòi hỏi kiến thức về JavaScript hiện đại (ES6+), JSX, và một số khái niệm phức tạp như state management, hooks.
- Cập nhật thường xuyên: ReactJS thay đổi và cập nhật thường xuyên, khiến việc theo kịp công nghệ và duy trì ứng dụng có thể tốn thời gian.

2.2.3. NodeJS

2.2.3.1. Giới thiệu

NodeJS là một mã nguồn mở, môi trường cho các máy chủ và ứng dụng mạng. NodeJS sử dụng Google V8 JavaScript engine để thực thi mã, và một tỷ lệ lớn các mô-đun cơ bản được viết bằng JavaScript. NodeJS cung cấp kiến trúc hướng sự kiện (event-driven) và non-blocking I/O API, tối ưu hóa thông lượng của ứng dụng và có khả năng mở rộng cao. Mọi hàm trong NodeJS là không đồng bộ (asynchronous). Do đó, các tác vụ đều được xử lý và thực thi ở chế độ nền (background processing).



Hình 2.7. Logo NodeJS

2.2.3.2. Ưu điểm

- Bởi cơ chế không đồng bộ, NodeJS nhận và xử lý nhiều kết nối chỉ với một single-thread. Điều này giúp hệ thống tốn ít RAM nhất và chạy nhanh nhất khi không phải tạo thread mới cho mỗi truy vấn.
- Sử dụng NPM (Node Package Manager) và các module Node để quản lý và chia sẻ code dễ dàng, đồng thời các package được tạo ra để rút gọn quá trình xây dựng

2.2.3.3. Nhược điểm

- Không có khả năng mở rộng, vì vậy không thể tận dụng lợi thế mô hình đa lõi trong các phần cứng cáp server hiện nay.
- Khó thao tác với cơ sở dữ liệu quan hệ.

2.2.4. Python (cho Recommender Service)

2.2.4. PostgreSQL

2.2.4.1. Giới thiệu

PostgreSQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở thuộc loại RDBMS (Relational Database Management System). PostgreSQL lưu trữ dữ liệu theo mô hình quan hệ, tức là các bảng và các quan hệ giữa chúng được quản lý thông qua các ràng buộc (constraints).



Hình 2.8. Logo PostgreSQL

2.2.4.2. Ưu điểm

- PostgreSQL hỗ trợ các ràng buộc (constraints) giữa các bảng, giúp đảm bảo tính toàn vẹn dữ liệu.

- Dễ dàng thực hiện các thao tác join giữa các bảng để truy vấn dữ liệu phức tạp.
- Được sử dụng rộng rãi trong các hệ thống yêu cầu tính bảo mật cao và độ tin cậy.

2.2.4.3. Nhược điểm

- Với một số tình huống, việc thực hiện các thao tác phức tạp có thể yêu cầu nhiều thời gian hơn so với các cơ sở dữ liệu NoSQL.
- Cấu hình và quản lý PostgreSQL có thể phức tạp hơn so với các hệ quản trị cơ sở dữ liệu NoSQL.

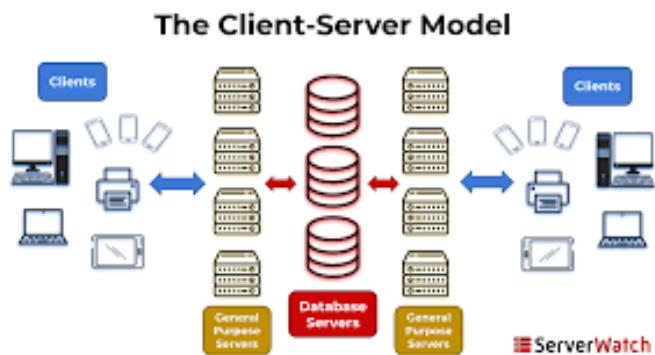
2.2.5. Mô hình máy khách – máy chủ (client – server)

2.2.5.1. Giới thiệu

Kiểu Client-Server (Khách-Chủ) cho hệ thống phân tán với hai thành phần riêng biệt là Khách (Client) và Chủ (Server). Trong mô hình này, client và server tương tác với nhau thông qua mạng hoặc Internet.

Trong mô hình client-server, server đóng vai trò như là một trung tâm điều khiển, cung cấp các dịch vụ và tài nguyên cho các client. Các client được cấp quyền truy cập vào các tài nguyên và dịch vụ của server thông qua các giao thức và phương thức truy cập như HTTP, FTP, SSH, Telnet, và nhiều hơn nữa.

Các ứng dụng client-server rất phổ biến trong các môi trường mạng và Internet, bao gồm các ứng dụng web, email, trò chơi trực tuyến, hệ thống quản lý cơ sở dữ liệu, và nhiều hơn nữa.



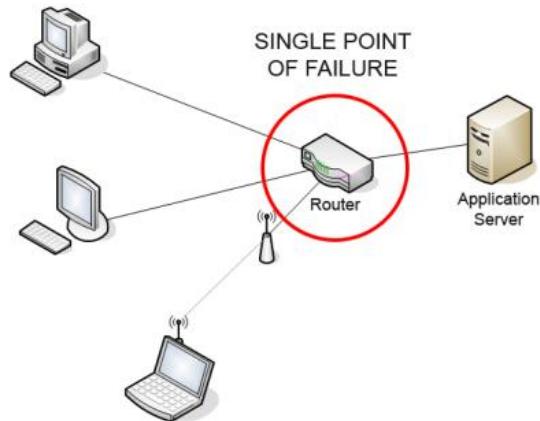
Hình 2.9. Mô hình Client-Server

2.2.5.2. Ưu điểm

- Chia sẻ tài nguyên.
- Thuận tiện cho việc bảo trì.
- An ninh: dữ liệu được giữ ở server nên được bảo vệ tốt hơn.

2.2.5.3. Nhược điểm

- Nghệp vụ và dữ liệu thường được để chung phía server.



Hình 2.10. Điểm lỗi duy nhất

2.2.6. Python

2.2.6.1 Giới thiệu

Python là ngôn ngữ lập trình đa năng, được sử dụng riêng cho Recommender Service nhờ thế mạnh vượt trội trong lĩnh vực khoa học dữ liệu và học máy.

2.2.6.2. Ưu điểm

- Hệ sinh thái AI/ML hàng đầu: Cung cấp các thư viện mạnh mẽ và toàn diện như Pandas, scikit-learn, Surprise, TensorFlow, PyTorch.
- Cú pháp đơn giản: Dễ đọc và dễ viết, giúp tăng tốc độ phát triển và đơn giản hóa việc hiện thực hóa các thuật toán phức tạp.
- Cộng đồng lớn: Được hỗ trợ bởi một cộng đồng khoa học dữ liệu và học máy khổng lồ.

2.2.6.3. Nhược điểm

- **Tốc độ thực thi:** Chậm hơn so với các ngôn ngữ biên dịch. Global Interpreter Lock (GIL) cũng là một hạn chế đối với các tác vụ đa luồng thực sự.

2.2.7. Redis

2.2.7.1 Giới thiệu

Redis là một kho lưu trữ dữ liệu key-value trong bộ nhớ, được sử dụng làm hệ thống cache để tăng tốc độ truy xuất.

2.2.7.2. Ưu điểm

- Tốc độ cực nhanh: Do hoạt động trên RAM, Redis cho phép đọc/ghi dữ liệu với độ trễ cực thấp, lý tưởng để lưu danh sách gợi ý và trạng thái người dùng.
- Linh hoạt: Hỗ trợ nhiều cấu trúc dữ liệu đa dạng như String, List, Hash, Set.

- Giảm tải cho CSDL chính: Giúp giảm đáng kể số lượng truy vấn đọc đến PostgreSQL.

2.2.7.3. Nhược điểm

- Giới hạn bởi RAM: Dung lượng lưu trữ bị giới hạn bởi bộ nhớ RAM vật lý.
- Dữ liệu dễ mất: Mặc định, dữ liệu sẽ mất khi khởi động lại nếu không cấu hình cơ chế lưu trữ bền vững (persistence).

2.2.8. Hybrid Algorithm (RBM + k-NN)

2.2.8.1 Giới thiệu

Đây là phương pháp gợi ý cốt lõi, kết hợp hai thuật toán để tận dụng ưu điểm của cả hai.

2.2.8.2. Ưu điểm

- Tăng độ chính xác: Khắc phục các điểm yếu của từng mô hình riêng lẻ (ví dụ: RBM giỏi học các đặc trưng ngầm, k-NN hiệu quả trong việc tìm kiếm tương đồng).
- Giải quyết vấn đề "Cold Start": Có khả năng xử lý tốt hơn đối với người dùng mới hoặc các bài hát mới.

2.2.8.3. Nhược điểm

- Độ phức tạp cao: Khó triển khai và tinh chỉnh hơn so với việc sử dụng một thuật toán duy nhất.
- Tốn tài nguyên: Yêu cầu nhiều tài nguyên tính toán và thời gian hơn để huấn luyện.

CHƯƠNG 3. TRIỂN KHAI THUẬT TOÁN

3.1. Ý tưởng thuật toán

Ý tưởng trung tâm của đề tài là xây dựng một hệ thống gợi ý âm nhạc có khả năng "thấu cảm", vượt qua giới hạn của các phương pháp truyền thống chỉ dựa trên lịch sử nghe nhạc. Thay vì gợi ý những gì người dùng đã thích trong quá khứ, hệ thống của chúng tôi tập trung vào việc trả lời câu hỏi: "Người dùng muốn nghe gì ngay tại thời điểm này, với tâm trạng này?"

Để hiện thực hóa ý tưởng này, thuật toán được xây dựng dựa trên hai luồng dữ liệu chính:

- + **Dữ liệu cảm xúc thời gian thực của người dùng:** trong phạm vi đề tài này, chúng tôi sử dụng phương pháp trích xuất cảm xúc từ dữ liệu webcam, sau này có thể thay thế thành các phương pháp trích xuất cảm xúc khác, ví dụ từ dữ liệu nhịp tim,...
- + **Dữ liệu thuộc tính cảm xúc của âm nhạc:** Được phân tích và gán nhãn tự động trước cho từng bài hát trong kho nhạc.

Hai luồng dữ liệu này được đưa vào một **mô hình gợi ý Hybrid**, kết hợp giữa sức mạnh của Lọc cộng tác (Collaborative Filtering) để tìm ra sở thích tiềm ẩn và Lọc dựa trên nội dung (Content-Based Filtering) để gợi ý các bài hát có thuộc tính cảm xúc tương đồng. Toàn bộ quy trình được thiết kế để vừa có khả năng cá nhân hóa sâu sắc, vừa đảm bảo tốc độ phản hồi nhanh chóng cho người dùng.

3.2. Triển khai phát hiện cảm xúc

3.2.1. Mô tả luồng xử lý

Chức năng này được triển khai hoàn toàn ở phía client (trình duyệt) để đảm bảo tốc độ và tính riêng tư.

- **Công nghệ sử dụng:** ReactJS, TypeScript, và thư viện **face-api.js** (xây dựng trên nền tảng TensorFlow.js).
- **Quy trình triển khai:**
 1. **Tải mô hình:** Khi ứng dụng khởi chạy, các mô hình AI đã được huấn luyện sẵn của **face-api.js** (**tinyFaceDetector** cho việc phát hiện khuôn mặt và **faceExpressionNet** cho việc nhận diện biểu cảm) được tải về từ thư mục public của ứng dụng.
 2. **Truy cập Webcam:** Hệ thống sử dụng API **navigator.mediaDevices.getUserMedia** của trình duyệt để yêu cầu quyền truy cập và nhận luồng video (video stream) từ webcam của người dùng.
 3. **Vòng lặp nhận diện:** Một vòng lặp **setInterval** được thiết lập để chạy mỗi 500 mili giây. Trong mỗi lần lặp:
 - Một khung hình (frame) từ video stream sẽ được chụp lại.
 - Hàm **faceapi.detectSingleFace(...).withFaceExpressions()** được gọi để phân tích khung hình đó.
 - Hàm này trả về một đối tượng chứa xác suất của 7 loại cảm xúc cơ bản. Cảm xúc có xác suất cao nhất được xác định là cảm xúc hiện tại (**currentEmotion**).
 4. **Thu thập dữ liệu:** Cảm xúc vừa được nhận diện sẽ được đẩy vào một mảng tạm thời (**emotionHistory**). Sau mỗi 10 giây, một "job" ở client sẽ phân tích mảng này để tìm ra cảm xúc chủ đạo và gửi lên máy chủ.

Để mô hình gợi ý có thể "học" được sở thích của người dùng, việc thu thập và lượng hóa các hành vi tương tác là một bước tối quan trọng. Hệ thống không chỉ ghi nhận hành động mà còn phiên dịch chúng thành một điểm số (**recommender score**) có ý nghĩa, phản ánh mức độ yêu thích của người dùng đối với một bài hát trong một trạng thái cảm xúc cụ thể. Quá trình này được thực hiện bởi **Emotion Collector Service**.

3.2.2. Các Loại Sự kiện Tương tác (Interaction Events)

Hệ thống được thiết kế để ghi nhận 5 loại sự kiện chính, bao gồm cả các tín hiệu tường minh và ngầm định từ người dùng:

1. **like**: Là tín hiệu tích cực tường minh và mạnh mẽ nhất, thể hiện người dùng chủ động bày tỏ sự yêu thích đối với bài hát.
2. **start_playing**: Một tín hiệu ngầm định cho thấy sự quan tâm ban đầu. Hành động này cho thấy người dùng tò mò và quyết định nghe thử bài hát.
3. **listen_through**: Tín hiệu tích cực mạnh, cho thấy người dùng đã kiên nhẫn nghe trọn vẹn một bài hát, một dấu hiệu rõ ràng của sự hài lòng.
4. **skip**: Tín hiệu ngầm định có thể mang sắc thái tiêu cực hoặc trung tính. Ý nghĩa của nó phụ thuộc nhiều vào thời điểm người dùng bỏ qua bài hát.
5. **cron_collect**: Đây là một sự kiện kỹ thuật, không phản ánh tương tác với bài hát. Nó chỉ dùng để cập nhật trạng thái cảm xúc mới nhất của người dùng vào Redis để phục vụ cho việc làm mới gợi ý.

3.2.3. Scoring Model

Mỗi sự kiện tương tác được ánh xạ thành một điểm số cụ thể. Điểm số này sau đó sẽ được cộng dồn vào bản ghi (`userId`, `musicId`, `emotion`) tương ứng trong cơ sở dữ liệu.

a. Phản hồi Tường minh (Explicit Feedback)

Đây là các hành động có chủ đích của người dùng và được gán trọng số cao nhất.

- **Sự kiện like:** Được gán một điểm số dương cố định và cao nhất: **+5.0 điểm.**

b. Phản hồi Ngầm (Implicit Feedback)

Đây là các hành động được suy ra từ hành vi nghe nhạc của người dùng.

- **Sự kiện listen_through:** Được gán một điểm số dương cao, thể hiện sự hài lòng: **+2.0 điểm.**
- **Sự kiện start_playing:** Được gán một điểm số dương nhỏ để ghi nhận sự quan tâm ban đầu: **+0.5 điểm.**
- **Sự kiện skip:** Đây là trường hợp phức tạp nhất. Điểm số được tính toán dựa trên phần trăm thời lượng bài hát đã nghe (p) thông qua một công thức phi tuyến sử dụng hàm `tanh` để phản ánh đúng tâm lý người nghe:
 - Nghe càng ít, điểm càng âm nặng (thể hiện sự không thích rõ rệt).
 - Nghe khoảng 50-60%, điểm số gần bằng 0 (vùng không chắc chắn).
 - Nghe gần hết mới bỏ qua, điểm số sẽ là số dương nhỏ.

- Công thức được áp dụng như sau:

$$\text{score} = \text{MAX_SCORE} \cdot \tanh\left(\frac{p - \text{NEUTRAL_POINT}}{\text{SENSITIVITY}}\right)$$

Với các tham số được cấu hình trong hệ thống:

- MAX_SCORE = 2.0
- NEUTRAL_POINT = 55 (Nghe 55% được 0 điểm)
- SENSITIVITY = 25

Công thức này được áp dụng nhằm:

+ **Bước 1: (p - NEUTRAL_POINT) - Dịch chuyển điểm trung tâm**

Mục đích: Hàm tanh(x) gốc có điểm trung tâm (nơi giá trị bằng 0) tại x=0. Nhưng chúng ta không muốn điểm số bằng 0 khi người dùng nghe 0% bài hát. Chúng ta muốn điểm số bằng 0 tại một điểm hợp lý hơn, ví dụ như 55% hoặc 60%. Phép trừ này chính là để "dịch chuyển" điểm 0 đó.

Ví dụ: Nếu ta chọn NEUTRAL_POINT = 55:

Khi người dùng nghe p = 55%, biểu thức này sẽ là 55 - 55 = 0.

Khi người dùng nghe p < 55%, biểu thức này sẽ là số âm.

Khi người dùng nghe p > 55%, biểu thức này sẽ là số dương.

Điều này đảm bảo điểm số sẽ âm khi nghe ít hơn 55% và dương khi nghe nhiều hơn 55%.

+ **Bước 2: SENSITIVITY - Điều chỉnh độ nhạy (độ dốc)**

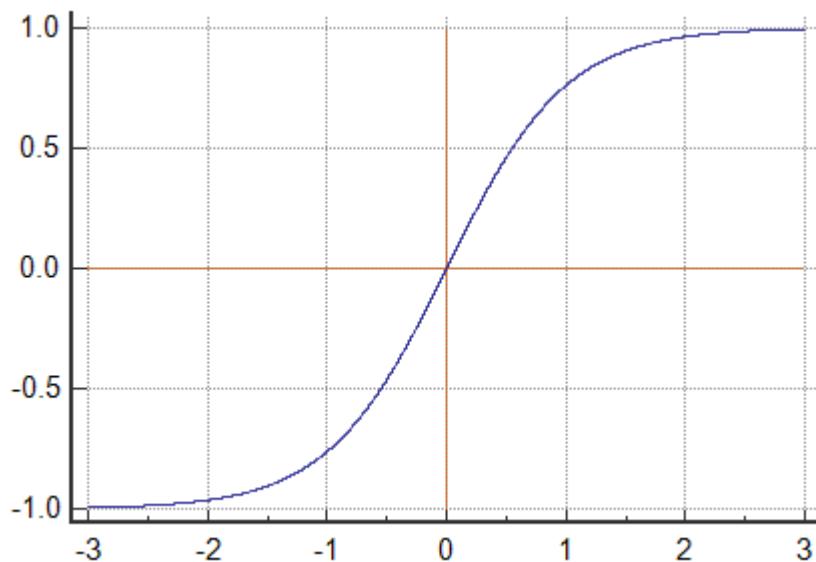
Mục đích: Tham số này quyết định xem đường cong "dốc" hay "thoải". Nó kiểm soát mức độ nhanh chóng mà điểm số chuyển từ âm sang dương.

Ví dụ:

SENSITIVITY nhỏ (ví dụ: 15): Đường cong sẽ rất dốc. Điều này có nghĩa là chỉ cần nghe lệch khỏi NEUTRAL_POINT một chút, điểm số sẽ ngay lập tức tăng vọt lên gần MAX_SCORE hoặc tụt xuống gần -MAX_SCORE. Hệ thống trở nên rất "khó tính" và phân định rạch ròi thích/ghét.

SENSITIVITY lớn (ví dụ: 40): Đường cong sẽ rất thoải. Điểm số sẽ thay đổi từ từ. Vùng "trung tính" sẽ rất rộng. Hệ thống trở nên "dễ tính" hơn và cho phép người dùng nghe một khoảng dài trước khi đưa ra kết luận.

+ Bước 3: tanh(...) - Áp dụng đường cong



Sau khi có được giá trị $x = (p - \text{NEUTRAL_POINT}) / \text{SENSITIVITY}$, chúng ta đưa nó vào hàm tanh.

Kết quả của $\tanh(x)$ sẽ là một con số mượt mà trong khoảng $[-1, 1]$, mang hình dạng chữ S mà chúng ta mong muốn.

+ **Bước 4: * MAX_SCORE - Mở rộng thang điểm**

Mục đích: Khoảng [-1, 1] có thể quá nhỏ để thể hiện sự khác biệt. Chúng ta cần nhân nó với một hệ số để có được thang điểm phù hợp với hệ thống..

Ví dụ: Nếu muốn điểm số dao động trong khoảng [-2, +2], ta đặt MAX_SCORE = 2. Nếu muốn là [-5, +5], ta đặt MAX_SCORE = 5.

3.2.4. Cơ chế Cập nhật Điểm Tích lũy

Thay vì ghi đè, hệ thống **cộng dồn điểm số mới vào điểm số hiện có** của một bản ghi (`userId, musicId, emotion`).

- **Logic:**

1. Khi một sự kiện xảy ra, hệ thống tính toán điểm `score` tương ứng.
2. Hệ thống truy vấn vào `Emotion collector database` để tìm bản ghi với (`userId, musicId, emotion`) hiện tại.
3. **Nếu tồn tại:** Cập nhật bản ghi bằng cách `existing_score + new_score`.
4. **Nếu không tồn tại:** Tạo một bản ghi mới với điểm số vừa tính được.

=> Cách tiếp cận này cho phép mô hình ghi nhận sự thay đổi trong sở thích của người dùng theo thời gian. Một bài hát ban đầu có thể chỉ được "start_playing" (+0.5 điểm), nhưng sau nhiều lần nghe và được "like" (+5 điểm), điểm số tích lũy cao sẽ phản ánh chính xác nó đã trở thành một bài hát yêu thích của người dùng trong một tâm trạng nhất định.

3.3. Triển khai phân tích cảm xúc bài hát

Đây là một tác vụ xử lý ngoại tuyến (offline) được thực hiện ở phía back-end để làm giàu dữ liệu.

- **Công nghệ sử dụng:** Python, **scikit-learn**, và các thư viện xử lý âm thanh như **Librosa**.
- **Quy trình triển khai:**
 1. **Lập lịch tác vụ:** Một Cron Job được thiết lập để tự động kích hoạt service Python này theo định kỳ (ví dụ: mỗi giờ một lần).
 2. **Truy xuất dữ liệu:** Service truy vấn vào **Main Database (PostgreSQL)** để lấy danh sách các bài hát chưa có nhãn cảm xúc.
 3. **Trích xuất đặc trưng:** Với mỗi bài hát, service truy cập vào file âm thanh được lưu trữ trên **Object Storage (MinIO)**. Thư viện Librosa được sử dụng để phân tích và trích xuất một vector đặc trưng âm học từ file nhạc, bao gồm các thông số như MFCCs, Spectral Centroid, Tempo, Chroma, v.v.

Đặc trưng Âm học	Định nghĩa	Tương quan Cảm xúc điển hình
Tempo (Nhịp độ)	Tốc độ của bản nhạc (BPM).	Cao: Năng lượng cao, vui vẻ, phấn khích. Thấp: Năng lượng thấp, buồn, thư thái.
Mode (Điệu thức)	Thang âm hoặc khuôn khổ (ví dụ: trưởng/thứ).	Trưởng: Tích cực, vui vẻ. Thứ: Tiêu cực, buồn bã.
MFCCs	Biểu diễn phổ công suất ngắn hạn trên thang Mel, phản ánh	Phức tạp, liên quan đến nhiều khía cạnh cảm xúc, thường

	âm sắc.	dùng kết hợp với các đặc trưng khác.
ZCR	Tốc độ tín hiệu vượt qua mức zero, liên quan đến độ nhiễu/tính bộ gõ.	Cao có thể chỉ sự kích động hoặc âm thanh sắc nét.
Spectral Centroid	"Trọng tâm" của phổ, liên quan đến "độ sáng" của âm thanh.	Cao: Âm thanh sáng, có thể liên quan đến sự vui vẻ, năng động. Thấp: Âm thanh tối, có thể liên quan đến sự trầm lắng, buồn.
Harmony (Hòa âm)	Sự kết hợp đồng thời của các nốt nhạc.	Thuận: Dễ chịu, bình yên. Nghịch: Căng thẳng, khó chịu.
Pitch (Cao độ)	Vị trí của âm thanh trên thang tần số.	Cao: Kích thích, căng thẳng. Thấp: Bình tĩnh, nặng nề.
Rhythm (Tiết tấu)	Kiểu mẫu của các nhịp và cách chúng được nhóm lại.	Đều đặn: Ổn định. Phức tạp/Bất thường: Kích thích, bất ổn.

4. **Dự đoán và Gán nhãn:** Vector đặc trưng này được đưa vào một mô hình **Random Forest** đã được huấn luyện trước. Mô hình sẽ dự đoán và trả về nhãn cảm xúc phù hợp nhất cho bài hát. Mô hình này được chúng tôi huấn luyện trên tập DEAM.

DEAM là một tập dữ liệu công khai, được sử dụng rộng rãi trong cộng đồng nghiên cứu về phân tích cảm xúc âm nhạc. Đặc điểm chính của DEAM bao gồm:

- **Nội dung:** Chứa 1,802 đoạn trích và bài hát hoàn chỉnh.

- **Gán nhãn:** Mỗi bài hát được gán nhãn cảm xúc theo mô hình không gian 2 chiều **Valence-Arousal**.
 - **Valence:** Biểu thị mức độ tích cực hoặc tiêu cực của cảm xúc (từ buồn bã đến vui vẻ).
 - **Arousal:** Biểu thị mức năng lượng hoặc cường độ của cảm xúc (từ bình lặng, thư giãn đến sôi động, mãnh liệt).

Mô hình của chúng ta cần dự đoán các nhãn cảm xúc rời rạc (như **happy, sad**) thay vì các giá trị (**Valence, Arousal**) liên tục. Do đó, một bước tiền xử lý quan trọng là ánh xạ các tọa độ V-A từ tập DEAM sang các nhãn cảm xúc mà hệ thống sử dụng.

Bảng mô tả các nhãn và quy tắc tiền xử lý:

Nhãn	Ý nghĩa	Chỉ số valence	Chỉ số arousal
ENERGY	Bài hát sôi động, tràn đầy năng lượng	> 5	> 7
HAPPY	Bài hát vui vẻ	> 5	> 5
ROMANTIC	Bài hát lãng mạn	> 5	> 3 và < 7
CHILL	Bài hát thư giãn, thoái mái	> 3 và < 7	< 3
SAD	Bài hát buồn	< 5	< 5
OTHER	Khác	Không thuộc các điều	Không thuộc các điều

		kiện phía trên	kiện phía trên
--	--	----------------	----------------

5. **Cập nhật cơ sở dữ liệu:** Nhãn cảm xúc vừa được dự đoán sẽ được cập nhật lại vào thông tin của bài hát trong **Main Database**.

3.4. Triển khai recommend dựa trên cảm xúc

Đây là tác vụ tính toán nặng và cốt lõi nhất, được thực hiện ngoại tuyến để chuẩn bị sẵn các gợi ý.

- **Công nghệ sử dụng:** Python, Pandas, và thư viện hệ thống gợi ý **Surprise**.
- **Quy trình triển khai:**
 1. **Lập lịch tác vụ:** Một Cron Job khác được lập lịch để chạy service này định kỳ (ví dụ: mỗi đêm).
 2. **Tổng hợp dữ liệu:** Service kết nối đến cả hai nguồn CSDL:
 - **Emotion collector database:** Để lấy toàn bộ dữ liệu tương tác của người dùng (**userId**, **musicId**, **emotion**, **score**).
 - **Main database:** Để lấy thông tin chi tiết về các bài hát.
 3. **Phân tách dữ liệu theo cảm xúc:** Toàn bộ dữ liệu tương tác được chia thành 7 bộ dữ liệu con, tương ứng với 7 loại cảm xúc.
 4. **Huấn luyện mô hình Hybrid:** Với mỗi bộ dữ liệu cảm xúc:
 - Một mô hình Hybrid duy nhất kết hợp **RBM** và **k-NN** được huấn luyện. Mô hình này học các tương tác của người dùng *chỉ trong trạng thái cảm xúc đó*.

- Việc này giúp hệ thống hiểu được rằng "sở thích của người dùng A khi vui" có thể hoàn toàn khác với "sở thích của người dùng A khi buồn".
5. **Tạo và Lưu trữ gợi ý:** Sau khi huấn luyện, mô hình sẽ tạo ra một danh sách top-N bài hát gợi ý cho mỗi người dùng ứng với từng cảm xúc. Danh sách này sau đó được lưu vào **Redis** dưới dạng **key-value** (ví dụ: `key = "user:1:emotion:2", value = "[10, 25, 30]"`).

3.5. Pipeline End-to-End

Toàn bộ các thuật toán trên được tích hợp thành một pipeline hoàn chỉnh, kết hợp giữa xử lý offline và real-time.

1. **Bước 0 (Offline - Chuẩn bị):** Các Cron Job chạy nền để phân tích cảm xúc bài hát và tạo sẵn các danh sách gợi ý cá nhân hóa trong Redis.
2. **Bước 1 (Realtime - Khởi tạo):** Người dùng truy cập ứng dụng. Front-end bắt đầu nhận diện cảm xúc và gọi API để lấy danh sách gợi ý ban đầu (dựa trên cảm xúc "neutral" hoặc top phổ biến).
3. **Bước 2 (Realtime - Tương tác):**
 - **Gửi cảm xúc:** Cứ mỗi 10 giây, cảm xúc chủ đạo của người dùng được tổng hợp và gửi lên **Emotion Collector Service**.
 - **Cập nhật gợi ý:** Ngay sau khi gửi thành công, Front-end tự động gọi lại API để làm mới danh sách gợi ý. **Main Service** ở back-end sẽ truy vấn vào Redis với **userId** và **emotionId** mới nhất để lấy ra danh sách phù hợp.

4. **Bước 3 (Thời gian thực - Phản hồi):** Người dùng tương tác với các bài hát (nghe, bỏ qua, thích). Các sự kiện này được gửi đến **Emotion Collector Service** và lưu lại.
5. **Bước 4 (Chu kỳ lặp lại):** Dữ liệu tương tác mới ở Bước 3 sẽ được **Recommender Service** sử dụng trong lần chạy Cron Job tiếp theo, tạo thành một vòng lặp cải tiến liên tục, giúp các gợi ý ngày càng chính xác hơn.

CHƯƠNG 4. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

4.1. Xác định yêu cầu

4.1.1. Yêu cầu chức năng

Đảm bảo đầu ra cho các chức năng nêu ở mục **1.3. Phạm vi chức năng**.

4.1.2. Yêu cầu nghiệp vụ

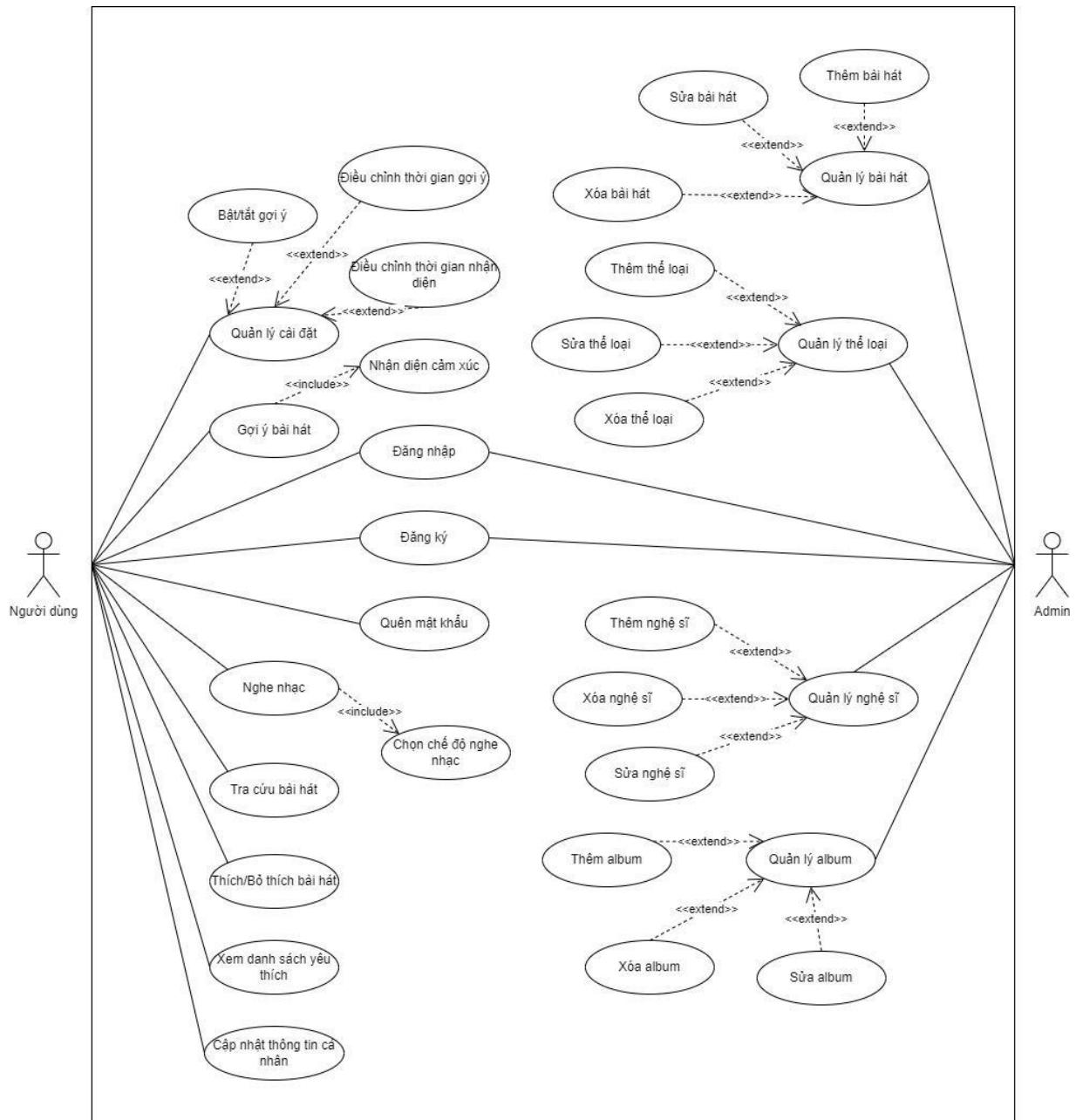
Bảng 3.1. Bảng yêu cầu nghiệp vụ

STT	Mã yêu cầu	Mô tả
1	BR01	Email được sử dụng là email hợp lệ.
2	BR02	Mật khẩu của tài khoản phải có ít nhất 10 ký tự và bao gồm ít nhất một chữ số, một chữ cái viết hoa và một ký tự đặc biệt.
3	BR03	Mail xác thực thông tin tồn tại trong 1 ngày.
4	BR04	Tài khoản đã được xác thực qua gmail mới được phép đăng nhập.
5	BR05	Phải nhập vào gmail và xác thực thông tin qua gmail mới được phép đổi mật khẩu.
6	BR06	Gợi ý tối đa 10 bài mỗi lần.
7	BR07	Nhận diện chỉ hoạt động khi người dùng đồng ý.
8	BR08	Thời gian phân tích không quá 5 giây.
9	BR09	Tìm kiếm không phân biệt hoa thường.
10	BR10	Mỗi bài hát chỉ có một trạng thái (thích hoặc không thích) tại một thời điểm.
11	BR11	Thời gian gợi ý từ 10 giây đến 5 phút.
12	BR12	Thời gian nhận diện từ 500 miligiây đến 5 giây.

13	BR13	Yêu cầu xác nhận trước khi xóa.
----	------	---------------------------------

4.2. Mô hình hóa yêu cầu

4.2.1. Lược đồ Use Case



Hình 3.1. Lược đồ use case

4.2.2. Danh sách chức năng

Bảng 3.2. Bảng danh sách chức năng

STT	Chức năng	Bao gồm	Mô tả
1	Xác thực tài khoản	<ul style="list-style-type: none"> ● Đăng ký ● Đăng nhập ● Quên mật khẩu 	Cho phép người dùng sử dụng tài khoản để truy cập vào hệ thống, thực hiện các chức năng cao hơn.
2	Nghe nhạc	<ul style="list-style-type: none"> ● Chọn chế độ nhạc 	Cho phép người nghe thưởng thức âm nhạc một cách linh hoạt với các chế độ nghe khác nhau như cá nhân (Personal) và học tập (Study) nhằm cung cấp những bài nhạc gợi ý đa dạng hơn
3	Gợi ý bài hát	<ul style="list-style-type: none"> ● Nhận diện cảm xúc 	Đề xuất bài hát dựa trên sở thích, lịch sử nghe nhạc hoặc trạng thái cảm xúc của người nghe, sử dụng công nghệ nhận diện để phân tích và cung cấp gợi ý phù hợp, giúp nâng cao trải nghiệm khám phá âm nhạc.
4	Tra cứu bài hát	N/A	Hỗ trợ người nghe tìm kiếm và xem thông tin chi tiết về bài hát, bao gồm lời bài hát, thông tin nghệ sĩ, album liên quan, và các chi tiết bổ sung như năm phát hành, giúp người dùng dễ dàng khám phá nội dung yêu thích.
5	Thích/Bỏ thích bài hát	N/A	Cho phép người nghe đánh dấu hoặc bỏ đánh dấu bài hát yêu thích một cách nhanh chóng, hỗ trợ cá nhân hóa danh sách nhạc và dễ dàng quản lý các bài hát

			quan trọng trong thư viện cá nhân.
6	Xem danh sách yêu thích	N/A	Hiển thị danh sách bài hát mà người nghe đã yêu thích một cách trực quan và dễ tiếp cận, cho phép sắp xếp hoặc lọc theo tiêu chí như ngày thêm hoặc thể loại, giúp người dùng quản lý sở thích âm nhạc hiệu quả..
6	Cập nhật thông tin cá nhân	N/A	Cho phép người nghe chỉnh sửa thông tin cá nhân như tên, email, số điện thoại hoặc sở thích âm nhạc, đảm bảo dữ liệu luôn chính xác và phản ánh đúng nhu cầu cá nhân hóa của từng người dùng.
7	Quản lý cài đặt	<ul style="list-style-type: none"> ● Bật/Tắt gợi ý ● Điều chỉnh thời gian gợi ý ● Điều chỉnh thời gian nhận diện 	Hỗ trợ người nghe tùy chỉnh các thiết lập liên quan đến gợi ý và nhận diện, bao gồm bật/tắt tính năng, điều chỉnh tần suất gợi ý hoặc thời gian xử lý nhận diện cảm xúc, giúp tối ưu hóa trải nghiệm sử dụng theo ý muốn.
8	Quản lý bài hát	<ul style="list-style-type: none"> ● Thêm bài hát ● Xóa bài viết ● Sửa bài hát 	Cho phép admin thực hiện các thao tác quản lý bài hát trong hệ thống một cách toàn diện, bao gồm thêm mới bài hát, chỉnh sửa thông tin chi tiết, và xóa bài hát không còn phù hợp, đảm bảo thư viện âm nhạc luôn được cập nhật và chất lượng.

9	Quản lý thể loại	<ul style="list-style-type: none"> ● Thêm thể loại ● Sửa thể loại ● Xóa thể loại 	Hỗ trợ admin quản lý danh mục thể loại âm nhạc một cách hiệu quả, bao gồm thêm thể loại mới, chỉnh sửa thông tin hiện có, và xóa thể loại không còn sử dụng, giúp tổ chức nội dung một cách khoa học và phù hợp với người dùng.
10	Quản lý album	<ul style="list-style-type: none"> ● Thêm album ● Sửa album ● Xóa album 	Cho phép admin quản lý danh sách album trong hệ thống một cách chi tiết, bao gồm thêm album mới, chỉnh sửa thông tin như tên hoặc nghệ sĩ, và xóa album không còn phù hợp, đảm bảo dữ liệu luôn chính xác và đầy đủ.
11	Quản lý nghệ sĩ	<ul style="list-style-type: none"> ● Thêm nghệ sĩ ● Sửa nghệ sĩ ● Xóa nghệ sĩ 	Hỗ trợ admin quản lý thông tin nghệ sĩ trong hệ thống một cách toàn diện, bao gồm thêm thông tin mới, chỉnh sửa dữ liệu hiện có như tiểu sử hoặc ảnh đại diện, và xóa nghệ sĩ không còn hoạt động, giữ cho hệ thống luôn cập nhật.

4.2.3. Danh sách tác nhân (Actor)

Bảng 3.3. Bảng tác nhân

STT	Mã Actor	Tên Actor	Mô tả
2	LS	Người nghe nhạc (Listener)	Người dùng có tài khoản có thể đăng nhập vào hệ thống và truy cập toàn quyền các chức năng dành riêng cho người dùng.

3	AD	Quản trị viên (Admin)	Quản trị viên có thể truy cập toàn quyền tất cả chức năng và trực tiếp quản lý danh sách bài hát, album, nghệ sĩ và thể loại nhạc.
---	----	-----------------------	--

4.2.4. Danh sách trường hợp sử dụng (Use Case)

Bảng 3.4. Bảng các use case

Mã Use Case	Tên Use Case	Mã Actor	Mô tả
UC01	Đăng ký	LS, AD	Hỗ trợ người dùng và admin tạo tài khoản mới một cách dễ dàng, cung cấp các bước đăng ký rõ ràng để nhập thông tin cá nhân và xác nhận, giúp mở rộng cộng đồng người sử dụng và đảm bảo quản lý quyền truy cập hiệu quả.
UC02	Đăng nhập	LS, AD	Người dùng có thể đăng nhập vào hệ thống thông qua tài khoản đã đăng ký.
UC03	Quên mật khẩu	LS	Người dùng có thể lấy lại tài khoản thông qua xác thực email trong trường hợp người dùng đã quên.
UC04	Nghe nhạc	LS	Cho phép người dùng thưởng thức âm nhạc một cách linh hoạt với các chế độ nghe khác nhau như phát ngẫu nhiên, phát lặp lại hoặc phát theo danh sách, mang lại trải nghiệm cá nhân hóa và thoải mái khi sử dụng hệ thống.
UC05	Chọn chế độ nghe nhạc	LS	Hỗ trợ người dùng lựa chọn các chế độ nghe nhạc khác nhau như cá nhân (Personal) và Học tập (Study) giúp tối

			ưu hóa trải nghiệm nghe nhạc theo sở thích cá nhân.
UC06	Gợi ý bài hát	LS	Đề xuất bài hát dựa trên sở thích, lịch sử nghe nhạc hoặc trạng thái cảm xúc của người dùng, sử dụng công nghệ nhận diện để phân tích và cung cấp gợi ý phù hợp, giúp nâng cao trải nghiệm khám phá âm nhạc.
UC07	Nhận diện cảm xúc	LS	Sử dụng công nghệ để phân tích trạng thái cảm xúc của người dùng dựa trên dữ liệu đầu vào, cung cấp thông tin phục vụ cho việc đề xuất bài hát phù hợp với tâm trạng hiện tại.
UC08	Tra cứu bài hát	LS	Hỗ trợ người dùng tìm kiếm và xem thông tin chi tiết về bài hát, bao gồm lời bài hát, thông tin nghệ sĩ, album liên quan, và các chi tiết bổ sung như năm phát hành, giúp người dùng dễ dàng khám phá nội dung yêu thích.
UC09	Thích/Bỏ thích bài hát	LS	Cho phép người dùng đánh dấu hoặc bỏ đánh dấu bài hát yêu thích một cách nhanh chóng, hỗ trợ cá nhân hóa danh sách nhạc và dễ dàng quản lý các bài hát quan trọng trong thư viện cá nhân.
UC10	Xem danh sách yêu thích	LS	Hiển thị danh sách bài hát mà người dùng đã yêu thích một cách trực quan và dễ tiếp cận, cho phép sắp xếp hoặc lọc theo tiêu chí như ngày thêm hoặc thể loại, giúp người dùng quản lý sở thích âm nhạc hiệu quả.

UC11	Cập nhật thông tin cá nhân	LS	Cho phép người dùng chỉnh sửa thông tin cá nhân như tên, email, số điện thoại hoặc sở thích âm nhạc, đảm bảo dữ liệu luôn chính xác và phản ánh đúng nhu cầu cá nhân hóa của từng người dùng.
UC12	Bật/Tắt gợi ý	LS	Cho phép người dùng kích hoạt hoặc vô hiệu hóa tính năng gợi ý bài hát, giúp kiểm soát trải nghiệm cá nhân hóa theo nhu cầu sử dụng.
UC13	Điều chỉnh thời gian gợi ý	LS	Hỗ trợ người dùng thiết lập tần suất xuất hiện của gợi ý bài hát, đảm bảo phù hợp với thói quen nghe nhạc và không làm gián đoạn trải nghiệm.
UC14	Điều chỉnh thời gian nhận diện	LS	Cho phép người dùng tùy chỉnh thời gian xử lý nhận diện cảm xúc, tối ưu hóa hiệu suất và độ chính xác của các gợi ý dựa trên trạng thái cá nhân.
UC15	Thêm bài hát	AD	Cho phép admin thêm mới bài hát vào hệ thống, bao gồm nhập thông tin chi tiết như tiêu đề, nghệ sĩ, và thể loại, đảm bảo thư viện âm nhạc được mở rộng và cập nhật.
UC16	Sửa bài hát	AD	Cho phép admin chỉnh sửa thông tin chi tiết của bài hát như tiêu đề, nghệ sĩ, hoặc thể loại, đảm bảo dữ liệu luôn chính xác và phản ánh đúng thực tế.
UC17	Xóa bài hát	AD	Hỗ trợ admin xóa bài hát không còn phù hợp hoặc không cần thiết khỏi hệ

			thống, đảm bảo chất lượng và tính nhất quán của thư viện âm nhạc.
UC18	Thêm thể loại	AD	Hỗ trợ admin thêm thể loại mới vào danh mục, giúp mở rộng và tổ chức nội dung âm nhạc một cách khoa học theo nhu cầu người dùng.
UC19	Sửa thể loại	AD	Hỗ trợ admin chỉnh sửa thông tin thể loại hiện có, đảm bảo danh mục thể loại luôn chính xác và phản ánh đúng nội dung âm nhạc.
UC20	Xóa thể loại	AD	Cho phép admin xóa thể loại không còn sử dụng khỏi hệ thống, đảm bảo danh mục luôn gọn gàng và phù hợp với thực tế.
UC21	Thêm album	AD	Cho phép admin thêm album mới vào hệ thống, bao gồm nhập thông tin như tên album, nghệ sĩ, và danh sách bài hát, hỗ trợ mở rộng thư viện âm nhạc.
UC22	Sửa album	AD	Hỗ trợ admin chỉnh sửa thông tin album như tên, nghệ sĩ, hoặc danh sách bài hát, đảm bảo dữ liệu luôn chính xác và cập nhật.
UC23	Xóa album	AD	Cho phép admin xóa album không còn phù hợp khỏi hệ thống, duy trì chất lượng và tính nhất quán của thư viện âm nhạc.
UC24	Thêm nghệ sĩ	AD	Hỗ trợ admin thêm thông tin nghệ sĩ mới vào hệ thống, bao gồm tiểu sử, ảnh đại diện, và danh sách bài hát,

			đảm bảo dữ liệu nghệ sĩ đầy đủ và chính xác.
UC25	Sửa nghệ sĩ	AD	Cho phép admin chỉnh sửa thông tin nghệ sĩ như tiêu sử hoặc ảnh đại diện, đảm bảo dữ liệu luôn phản ánh đúng thực tế và cập nhật.
UC26	Xóa nghệ sĩ	AD	Hỗ trợ admin xóa thông tin nghệ sĩ không còn hoạt động khỏi hệ thống, giữ cho danh sách nghệ sĩ luôn sạch sẽ và phù hợp..

4.2.5. ĐẶC TẢ USE CASE

Bảng 3.7. Đặc tả use case đăng ký

Use Case ID	UC01
Use Case Name	Đăng ký
Description	Cho phép người dùng chưa có tài khoản đăng ký để đăng nhập vào hệ thống.
Actor(s)	LS, AD
Priority	Cao
Trigger	Người dùng chọn nút “Đăng ký”.
Pre-condition(s)	<ul style="list-style-type: none"> • Người dùng ở trang xác thực tài khoản
Post-condition(s)	<ul style="list-style-type: none"> • Người dùng đăng ký tài khoản thành công. • Hệ thống ghi nhận hoạt động người dùng vào Activity Log.
Basic Flow	<ol style="list-style-type: none"> 1. Người dùng chọn nút “Đăng ký” được hiển thị vào màn hình. 2. Người dùng nhập vào các trường trong biểu mẫu. 3. Hệ thống gửi mail xác nhận cho người dùng. 4. Người dùng xác nhận thông tin qua email.
Alternate Flow	2a. Người dùng chọn đăng ký bằng Gmail.

	2a1. Hệ thống hiển thị đăng ký bằng Gmail và chọn tài khoản để đăng ký. Use case tiếp tục đến bước 4.
Exception Flow	4b. Người dùng không nhận thông tin qua Email. 4b1. Hệ thống không thay đổi. Use case dừng lại.
Business Rules	BR01, BR02, BR03

Bảng 3.8. Đặc tả use case đăng nhập

Use Case ID	UC02
Use Case Name	Đăng nhập
Description	Người dùng đăng nhập vào hệ thống với tài khoản đã đăng ký trước đó.
Actor(s)	LS, AD
Priority	Cao
Trigger	Người dùng chọn nút “Đăng nhập”.
Pre-condition(s)	<ul style="list-style-type: none"> • Tài khoản đã được xác thực và phân quyền • Người dùng ở trang xác thực tài khoản
Post-condition(s)	<ul style="list-style-type: none"> • Người dùng đăng nhập vào hệ thống thành công.
Basic Flow	<ol style="list-style-type: none"> 1. Người dùng nhập vào tài khoản và chọn nút “Đăng nhập”. 2. Hệ thống xác thực thông tin đăng nhập và cho phép người dùng truy cập vào hệ thống.
Alternate Flow	<ol style="list-style-type: none"> 1a. Người dùng chọn đăng nhập bằng tài khoản Gmail. 1a1. Hệ thống chuyển đến trang đăng nhập của Google. 2a. Người dùng chọn tài khoản Gmail và đăng nhập. <p>Use case tiếp tục đến bước 3.</p>
Exception Flow	<ol style="list-style-type: none"> 2b. Hệ thống xác thực thông tin đăng nhập không thành công và hiển thị thông báo. <p>Use case dừng lại.</p>
Business Rules	BR01, BR04

Bảng 3.9. Đặc tả use case quên mật khẩu

Use Case ID	UC03
Use Case Name	Quên mật khẩu.
Description	Người dùng quên mật khẩu và cần lấy lại mật khẩu.
Actor(s)	LS
Priority	Cao
Trigger	Người dùng chọn nút “Quên mật khẩu”.
Pre-condition(s)	<ul style="list-style-type: none"> • Tài khoản đã được xác thực và phân quyền. • Người dùng ở trang xác thực tài khoản
Post-condition(s)	Người dùng tạo được mật khẩu mới và đăng nhập vào hệ thống.
Basic Flow	<ol style="list-style-type: none"> 1. Người dùng nhập vào mail đã đăng ký trước đó. 2. Hệ thống gửi mail xác nhận cho người dùng. 3. Người dùng xác nhận thông tin qua email. 4. Hệ thống chuyển đến trang tạo mật khẩu mới. 5. Người dùng tạo mật khẩu mới.
Alternate Flow	N/A
Exception Flow	<p>3a. Người dùng không nhận thông tin qua Email. 3a1. Hệ thống không thay đổi. Use case dừng lại.</p> <p>5b. Người dùng không nhập vào mật khẩu mới. 5b1. Hệ thống không thay đổi. Use case dừng lại.</p>
Business Rules	BR05

Bảng 3.10. Đặc tả use case nghe nhạc

Use Case ID	UC04
Use Case Name	Nghe nhạc
Description	Cho phép người dùng nghe nhạc với các chế độ như ngẫu nhiên, lặp lại hoặc theo danh sách, mang lại trải nghiệm cá nhân hóa.
Actor(s)	LS

Priority	Cao
Trigger	Người dùng chọn bài hát
Pre-condition(s)	<ul style="list-style-type: none"> Người dùng đã đăng nhập. Hệ thống có kết nối internet hoặc dữ liệu đã tải.
Post-condition(s)	<ul style="list-style-type: none"> Bài hát hoặc danh sách phát được phát thành công.
Basic Flow	<ol style="list-style-type: none"> Hệ thống hiển thị danh sách bài hát/danh sách phát. Người dùng chọn bài hoặc danh sách. Hệ thống tải và phát nhạc. Kết thúc khi người dùng dừng hoặc chuyển bài.
Alternate Flow	N/A
Exception Flow	<ol style="list-style-type: none"> Nếu tải thất bại, hiển thị lỗi và kiểm tra kết nối.
Business Rules	N/A

Bảng 3.11. Đặc tả use case chọn chế độ nghe nhạc

Use Case ID	UC05
Use Case Name	Chọn chế độ nghe nhạc
Description	Hỗ trợ người dùng chọn chế độ nghe nhạc (cá nhân, học tập) để tối ưu hóa trải nghiệm.
Actor(s)	LS
Priority	Cao
Trigger	Chọn ở Modal lúc vừa đăng nhập hoặc Chọn ở nút “Change mode”
Pre-condition(s)	<ul style="list-style-type: none"> Người dùng chưa chọn chế độ nghe nhạc
Post-condition(s)	<ul style="list-style-type: none"> Người dùng đã chọn chế độ nghe nhạc Bài hát sẽ được đề xuất dựa trên chế độ nghe nhạc
Basic Flow	<ol style="list-style-type: none"> Người dùng chọn một chế độ nghe nhạc Hệ thống sẽ bắt đầu đề xuất bài hát dựa trên chế độ nghe nhạc người dùng đã chọn.
Alternate Flow	N/A

Exception Flow	N/A
Business Rules	N/A

Bảng 3.12. Đặc tả use case gọi ý bài hát

Use Case ID	UC06
Use Case Name	Gọi ý bài hát
Description	Đề xuất bài hát dựa trên sở thích hoặc trạng thái cảm xúc, sử dụng nhận diện để cung cấp gợi ý phù hợp.
Actor(s)	LS
Priority	Cao
Trigger	Người dùng bật cho phép gợi ý
Pre-condition(s)	<ul style="list-style-type: none"> • Người dùng đã đăng nhập. • Đã nhận diện được cảm xúc từ webcam. • Cho phép gợi ý đã được bật
Post-condition(s)	Bài hát gợi ý sẽ được thêm vào danh sách phát hiện tại
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống gửi dữ liệu cảm xúc về server. 2. Hệ thống xử lý và chọn bài hát phù hợp. 3. Hệ thống hiển thị bài hát gợi ý.
Alternate Flow	N/A
Exception Flow	N/A
Business Rules	BR06, BR07, BR12

Bảng 3.13. Đặc tả use case nhận diện cảm xúc

Use Case ID	UC07
Use Case Name	Nhận diện cảm xúc
Description	Phân tích trạng thái cảm xúc của người dùng để hỗ trợ gợi ý bài hát phù hợp.
Actor(s)	LS
Priority	Cao
Trigger	Người dùng bật webcam.

Pre-condition(s)	<ul style="list-style-type: none"> Người dùng đã đăng nhập. Webcam đã được bật.
Post-condition(s)	Trả về các cảm xúc được nhận diện từ webcam.
Basic Flow	<ol style="list-style-type: none"> Hệ thống thu thập dữ liệu từ webcam. Hệ thống phân tích và xác định trạng thái cảm xúc. Kết quả được lưu lại, tính toán và gửi về server sau một khoảng thời gian.
Alternate Flow	N/A
Exception Flow	N/A
Business Rules	BR08, BR11

Bảng 3.14. Đặc tả use case tra cứu bài hát

Use Case ID	UC08
Use Case Name	Tra cứu bài hát
Description	Hỗ trợ người dùng tìm kiếm và xem thông tin chi tiết về bài hát như lời bài, nghệ sĩ, và album.
Actor(s)	LS
Priority	Trung bình
Trigger	Người dùng qua tab “Explore”, nhập thông tin vào thanh tìm kiếm và chọn tìm kiếm
Pre-condition(s)	<ul style="list-style-type: none"> Người dùng đã đăng nhập. Hệ thống có dữ liệu bài hát.
Post-condition(s)	Thông tin bài hát được hiển thị cho người dùng.
Basic Flow	<ol style="list-style-type: none"> Hệ thống hiển thị thanh tìm kiếm. Người dùng nhập từ khóa và nhấn tìm kiếm. Hệ thống truy vấn cơ sở dữ liệu và trả về kết quả.
Alternate Flow	N/A
Exception Flow	Nếu không tìm thấy, hiển thị thông báo "No song found!".

Business Rules	BR09
----------------	------

Bảng 3.15. Đặc tả use case thích/bỏ thích bài hát

Use Case ID	UC09
Use Case Name	Thích/BỎ thích bài hát
Description	Cho phép người dùng đánh dấu hoặc bỏ đánh dấu bài hát yêu thích để cá nhân hóa danh sách.
Actor(s)	LS
Priority	Thấp
Trigger	Người dùng chọn thêm vào danh sách yêu thích hoặc loại khỏi danh sách yêu thích
Pre-condition(s)	<ul style="list-style-type: none"> • Người dùng đã đăng nhập. • Bài hát đang hiển thị trên giao diện.
Post-condition(s)	<ul style="list-style-type: none"> • Trạng thái yêu thích được cập nhật trong danh sách cá nhân.
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị nút "Thích" hoặc "BỎ thích" bên cạnh bài hát. 2. Người dùng nhấn nút tương ứng. 3. Hệ thống cập nhật trạng thái vào cơ sở dữ liệu. 4. Kết thúc khi cập nhật thành công.
Alternate Flow	N/A
Exception Flow	N/A
Business Rules	BR10

Bảng 3.16. Đặc tả use case xem danh sách yêu thích

Use Case ID	UC10
Use Case Name	Xem danh sách yêu thích
Description	Hiển thị danh sách bài hát yêu thích của người dùng để quản lý dễ dàng.

Actor(s)	LS
Priority	Trung bình
Trigger	Mở tab “My Favorite”
Pre-condition(s)	<ul style="list-style-type: none"> Người dùng đã đăng nhập. Người dùng có ít nhất một bài hát yêu thích.
Post-condition(s)	<ul style="list-style-type: none"> Danh sách yêu thích được hiển thị trên giao diện.
Basic Flow	<ol style="list-style-type: none"> Người dùng mở tab "My Favorite". Hệ thống tải và hiển thị danh sách từ cơ sở dữ liệu.
Alternate Flow	N/A
Exception Flow	Nếu không có bài hát yêu thích thì sẽ trả về “No favourite song found!”
Business Rules	N/A

Bảng 3.17. Đặc tả use case cập nhật thông tin cá nhân

Use Case ID	UC11
Use Case Name	Cập nhật thông tin cá nhân
Description	Người dùng thay đổi các thông tin đến tài khoản như tên, email, mật khẩu, ...
Actor(s)	LS
Priority	Trung bình
Trigger	Người dùng nhấn vào nút “Save” ở trang “Profile”
Pre-condition(s)	<ul style="list-style-type: none"> Tài khoản đã được xác thực và phân quyền. Người dùng ở trang quản lý cá nhân. Thông tin của người dùng đã được lưu trên hệ thống.
Post-condition(s)	Thông tin của người dùng thay đổi thành công.
Basic Flow	<ol style="list-style-type: none"> Người dùng thay đổi các trường thông tin. Người dùng nhấn vào nút “Cập nhật”. Hệ thống cập nhật các thông tin được thay đổi.
Alternate Flow	N/A

Exception Flow	<p>1a. Người dùng không thay đổi các trường thông tin nào.</p> <p>1a1. Hệ thống không thay đổi.</p> <p>Use case dừng lại.</p> <p>2a. Người dùng không nhấn vào nút “Cập nhật”.</p> <p>2a1. Hệ thống không thay đổi.</p> <p>Use case dừng lại.</p>
Business Rules	N/A

Bảng 3.18. Đặc tả use case bật/tắt gợi ý

Use Case ID	UC12
Use Case Name	Bật/Tắt gợi ý
Description	Cho phép người dùng kích hoạt hoặc vô hiệu hóa tính năng gợi ý bài hát theo nhu cầu.
Actor(s)	LS
Priority	Thấp
Trigger	Người dùng chọn bật/tắt gợi ý
Pre-condition(s)	Người dùng đang ở giao diện cài đặt.
Post-condition(s)	Trạng thái gợi ý được cập nhật và áp dụng ngay.
Basic Flow	<ol style="list-style-type: none"> Hệ thống hiển thị công tắc "Bật/Tắt gợi ý". Người dùng thay đổi trạng thái công tắc. Hệ thống lưu và áp dụng thay đổi. Kết thúc khi cập nhật thành công.
Alternate Flow	N/A
Exception Flow	N/A
Business Rules	N/A

Bảng 3.19. Đặc tả use case điều chỉnh thời gian gợi ý

Use Case ID	UC13
Use Case Name	Điều chỉnh thời gian gợi ý

Description	Hỗ trợ người dùng thiết lập tần suất gọi ý bài hát phù hợp với thói quen nghe nhạc.
Actor(s)	LS
Priority	Thấp
Trigger	Người dùng thay đổi giá trị thời gian trong cài đặt.
Pre-condition(s)	Người dùng đang ở giao diện cài đặt.
Post-condition(s)	Thời gian gọi ý được cập nhật và áp dụng.
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị trường nhập thời gian gọi ý. 2. Người dùng điều chỉnh giá trị thời gian. 3. Hệ thống lưu và áp dụng thay đổi. 4. Kết thúc khi cập nhật thành công.
Alternate Flow	N/A
Exception Flow	Nếu giá trị không hợp lệ, hiển thị lỗi và giữ giá trị cũ. Nếu lỗi lưu, hiển thị lỗi và thử lại.
Business Rules	N/A

Bảng 3.20. Đặc tả use case điều chỉnh thời gian nhận diện

Use Case ID	UC14
Use Case Name	Điều chỉnh thời gian nhận diện
Description	Cho phép người dùng tùy chỉnh thời gian xử lý nhận diện cảm xúc để tối ưu hóa hiệu suất.
Actor(s)	LS
Priority	Thấp
Trigger	Người dùng thay đổi giá trị thời gian trong cài đặt.
Pre-condition(s)	Người dùng đang ở giao diện cài đặt.
Post-condition(s)	Thời gian nhận diện được cập nhật và áp dụng.
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị trường nhập thời gian nhận diện. 2. Người dùng điều chỉnh giá trị thời gian.

	<p>3. Hệ thống lưu và áp dụng thay đổi.</p> <p>4. Kết thúc khi cập nhật thành công.</p>
Alternate Flow	N/A
Exception Flow	Nếu giá trị không hợp lệ, hiển thị lỗi và giữ giá trị cũ. Nếu lỗi lưu, hiển thị lỗi và thử lại.
Business Rules	N/A

Bảng 3.21. Đặc tả use case thêm bài hát

Use Case ID	UC15
Use Case Name	Thêm bài hát
Description	Cho phép admin thêm bài hát mới vào hệ thống với thông tin chi tiết.
Actor(s)	AD
Priority	Cao
Trigger	Admin nhấn nút "Add song" trên giao diện quản lý.
Pre-condition(s)	Admin đã đăng nhập.
Post-condition(s)	Bài hát mới được thêm vào cơ sở dữ liệu.
Basic Flow	<p>1. Hệ thống hiển thị form nhập thông tin bài hát.</p> <p>2. Admin nhập thông tin và nhấn "Save".</p> <p>3. Hệ thống kiểm tra và lưu vào cơ sở dữ liệu.</p> <p>4. Kết thúc khi lưu thành công.</p>
Alternate Flow	N/A
Exception Flow	Nếu thông tin trùng lặp, hiển thị lỗi và yêu cầu chỉnh sửa. Nếu lỗi lưu, hiển thị lỗi và thử lại.
Business Rules	N/A

Bảng 3.22. Đặc tả use case sửa bài hát

Use Case ID	UC16
-------------	------

Use Case Name	Sửa bài hát
Description	Cho phép admin chỉnh sửa thông tin bài hát để cập nhật dữ liệu.
Actor(s)	AD
Priority	Trung bình
Trigger	Admin chọn bài hát và nhấn "Edit" trên giao diện quản lý.
Pre-condition(s)	<ul style="list-style-type: none"> • Admin đã đăng nhập. • Bài hát tồn tại trong cơ sở dữ liệu.
Post-condition(s)	Thông tin bài hát được cập nhật trong hệ thống.
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị form chỉnh sửa thông tin bài hát. 2. Admin chỉnh sửa thông tin. 3. Admin nhấn "Save" để cập nhật. 4. Kết thúc khi cập nhật thành công.
Alternate Flow	N/A
Exception Flow	Nếu thông tin không hợp lệ, hiển thị lỗi và yêu cầu chỉnh sửa. Nếu lỗi cập nhật, hiển thị lỗi và thử lại.
Business Rules	N/A

Bảng 3.23. ĐẶC TẢ USE CASE XÓA BÀI HÁT

Use Case ID	UC17
Use Case Name	Xóa bài hát
Description	Cho phép người dùng tùy chỉnh thời gian xử lý nhận diện cảm xúc để tối ưu hóa hiệu suất.
Actor(s)	AD
Priority	Trung bình
Trigger	Admin chọn bài hát và nhấn "Delete" trên giao diện quản lý.
Pre-condition(s)	<ul style="list-style-type: none"> • Admin đã đăng nhập. • Bài hát tồn tại trong cơ sở dữ liệu.
Post-condition(s)	Bài hát được xóa khỏi hệ thống.

Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị danh sách bài hát. 2. Admin chọn bài hát và nhấn "Delete". 3. Hệ thống xác nhận và xóa khỏi cơ sở dữ liệu. 4. Kết thúc khi xóa thành công.
Alternate Flow	N/A
Exception Flow	Nếu bài hát đang được sử dụng, hiển thị cảnh báo và hủy xóa. Nếu lỗi xóa, hiển thị lỗi và thử lại.
Business Rules	N/A

Bảng 3.24. Đặc tả use case thêm thể loại

Use Case ID	UC18
Use Case Name	Thêm thể loại
Description	Cho phép admin thêm thể loại mới vào danh mục để mở rộng nội dung âm nhạc.
Actor(s)	AD
Priority	Trung bình
Trigger	Admin nhấn nút "Add genre" trên giao diện quản lý.
Pre-condition(s)	Admin đã đăng nhập.
Post-condition(s)	Thể loại mới được thêm vào cơ sở dữ liệu.
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị form nhập thông tin thể loại. 2. Admin nhập thông tin và nhấn "Save". 3. Hệ thống kiểm tra và lưu vào cơ sở dữ liệu. 4. Kết thúc khi lưu thành công.
Alternate Flow	N/A
Exception Flow	Nếu giá trị không hợp lệ, hiển thị lỗi và yêu cầu chỉnh sửa. Nếu lỗi lưu, hiển thị lỗi và thử lại.
Business Rules	N/A

Bảng 3.25. Đặc tả use case sửa thẻ loại

Use Case ID	UC19
Use Case Name	Sửa thẻ loại
Description	Cho phép admin chỉnh sửa thông tin thẻ loại để cập nhật dữ liệu.
Actor(s)	AD
Priority	Trung bình
Trigger	Admin chọn thẻ loại và nhấn "Edit" trên giao diện quản lý.
Pre-condition(s)	<ul style="list-style-type: none"> ● Admin đã đăng nhập. ● Thẻ loại tồn tại trong cơ sở dữ liệu.
Post-condition(s)	Thông tin thẻ loại được cập nhật trong hệ thống.
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị form chỉnh sửa thông tin thẻ loại. 2. Admin chỉnh sửa thông tin. 3. Admin nhấn "Save" để cập nhật. 4. Kết thúc khi cập nhật thành công.
Alternate Flow	N/A
Exception Flow	Nếu giá trị không hợp lệ, hiển thị lỗi và yêu cầu chỉnh sửa. Nếu lỗi lưu, hiển thị lỗi và thử lại.
Business Rules	N/A

Bảng 3.26. Đặc tả use case xóa thẻ loại

Use Case ID	UC20
Use Case Name	Xóa thẻ loại
Description	Hỗ trợ admin xóa thẻ loại không còn sử dụng khỏi danh mục.
Actor(s)	AD
Priority	Thấp
Trigger	Admin chọn thẻ loại và nhấn "Delete" trên giao diện quản lý.
Pre-condition(s)	<ul style="list-style-type: none"> ● Admin đã đăng nhập. ● Thẻ loại tồn tại và không chứa bài hát.

Post-condition(s)	Thể loại được xóa khỏi hệ thống.
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị danh sách thể loại. 2. Admin chọn thể loại và nhấn "Delete". 3. Hệ thống xác nhận và xóa khỏi cơ sở dữ liệu. 4. Kết thúc khi xóa thành công.
Alternate Flow	N/A
Exception Flow	Nếu thể loại chưa bài hát, hiển thị cảnh báo và hủy xóa. Nếu lỗi xóa, hiển thị lỗi và thử lại.
Business Rules	N/A

Bảng 3.27. Đặc tả use case thêm album

Use Case ID	UC21
Use Case Name	Thêm album
Description	Cho phép admin thêm album mới vào hệ thống với thông tin chi tiết.
Actor(s)	AD
Priority	Cao
Trigger	Admin nhấn nút "Add album" trên giao diện quản lý.
Pre-condition(s)	Admin đã đăng nhập.
Post-condition(s)	Album mới được thêm vào cơ sở dữ liệu.
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị form nhập thông tin album. 2. Admin nhập thông tin và nhấn "Save". 3. Hệ thống kiểm tra và lưu vào cơ sở dữ liệu. 4. Kết thúc khi lưu thành công.
Alternate Flow	N/A
Exception Flow	Nếu tên album trùng lặp, hiển thị lỗi và yêu cầu chỉnh sửa. Nếu lỗi lưu, hiển thị lỗi và thử lại.
Business Rules	N/A

Bảng 3.28. Đặc tả use case sửa album

Use Case ID	UC22
Use Case Name	Sửa album
Description	Cho phép admin chỉnh sửa thông tin album để cập nhật dữ liệu.
Actor(s)	AD
Priority	Trung bình
Trigger	Admin chọn album và nhấn "Edit" trên giao diện quản lý.
Pre-condition(s)	<ul style="list-style-type: none"> • Admin đã đăng nhập. • Album tồn tại trong cơ sở dữ liệu.
Post-condition(s)	Thông tin album được cập nhật trong hệ thống.
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị form chỉnh sửa thông tin album. 2. Admin chỉnh sửa thông tin. 3. Admin nhấn "Save" để cập nhật. 4. Kết thúc khi cập nhật thành công.
Alternate Flow	N/A
Exception Flow	Nếu thông tin không hợp lệ, hiển thị lỗi và yêu cầu chỉnh sửa. Nếu lỗi cập nhật, hiển thị lỗi và thử lại.
Business Rules	N/A

Bảng 3.29. Đặc tả use case xóa album

Use Case ID	UC23
Use Case Name	Xóa album
Description	Hỗ trợ admin xóa album không còn phù hợp khỏi hệ thống.
Actor(s)	AD
Priority	Thấp
Trigger	Admin chọn album và nhấn "Delete" trên giao diện quản lý.
Pre-condition(s)	<ul style="list-style-type: none"> • Admin đã đăng nhập.

	<ul style="list-style-type: none"> Album tồn tại và không chứa bài hát đang sử dụng.
Post-condition(s)	Album được xóa khỏi hệ thống.
Basic Flow	<ol style="list-style-type: none"> Hệ thống hiển thị danh sách album. Admin chọn album và nhấn "Delete". Hệ thống xác nhận và xóa khỏi cơ sở dữ liệu. Kết thúc khi xóa thành công.
Alternate Flow	N/A
Exception Flow	<p>Nếu album chứa bài hát đang sử dụng, hiển thị cảnh báo và hủy xóa.</p> <p>Nếu lỗi xóa, hiển thị lỗi và thử lại.</p>
Business Rules	N/A

Bảng 3.30. Đặc tả use case thêm nghệ sĩ

Use Case ID	UC24
Use Case Name	Thêm nghệ sĩ
Description	Cho phép admin thêm nghệ sĩ mới vào hệ thống với thông tin chi tiết.
Actor(s)	AD
Priority	Trung bình
Trigger	Admin nhấn nút "Add artist" trên giao diện quản lý.
Pre-condition(s)	Admin đã đăng nhập.
Post-condition(s)	Nghệ sĩ mới được thêm vào cơ sở dữ liệu.
Basic Flow	<ol style="list-style-type: none"> Hệ thống hiển thị form nhập thông tin nghệ sĩ. Admin nhập thông tin và nhấn "Save". Hệ thống kiểm tra và lưu vào cơ sở dữ liệu. Kết thúc khi lưu thành công.
Alternate Flow	N/A
Exception Flow	<p>Nếu tên nghệ sĩ trùng lặp, hiển thị lỗi và yêu cầu chỉnh sửa.</p> <p>Nếu lỗi lưu, hiển thị lỗi và thử lại.</p>

Business Rules	N/A
----------------	-----

Bảng 3.31. Đặc tả use case sửa nghệ sĩ

Use Case ID	UC25
Use Case Name	Sửa nghệ sĩ
Description	Cho phép admin chỉnh sửa thông tin nghệ sĩ để cập nhật dữ liệu.
Actor(s)	AD
Priority	Thấp
Trigger	Admin chọn nghệ sĩ và nhấn "Edit" trên giao diện quản lý.
Pre-condition(s)	<ul style="list-style-type: none"> • Admin đã đăng nhập. • Nghệ sĩ tồn tại trong cơ sở dữ liệu.
Post-condition(s)	Thông tin nghệ sĩ được cập nhật trong hệ thống.
Basic Flow	<ol style="list-style-type: none"> 1. Hệ thống hiển thị form chỉnh sửa thông tin nghệ sĩ. 2. Admin chỉnh sửa thông tin. 3. Admin nhấn "Save" để cập nhật. 4. Kết thúc khi cập nhật thành công.
Alternate Flow	N/A
Exception Flow	Nếu thông tin không hợp lệ, hiển thị lỗi và yêu cầu chỉnh sửa. Nếu lỗi cập nhật, hiển thị lỗi và thử lại.
Business Rules	N/A

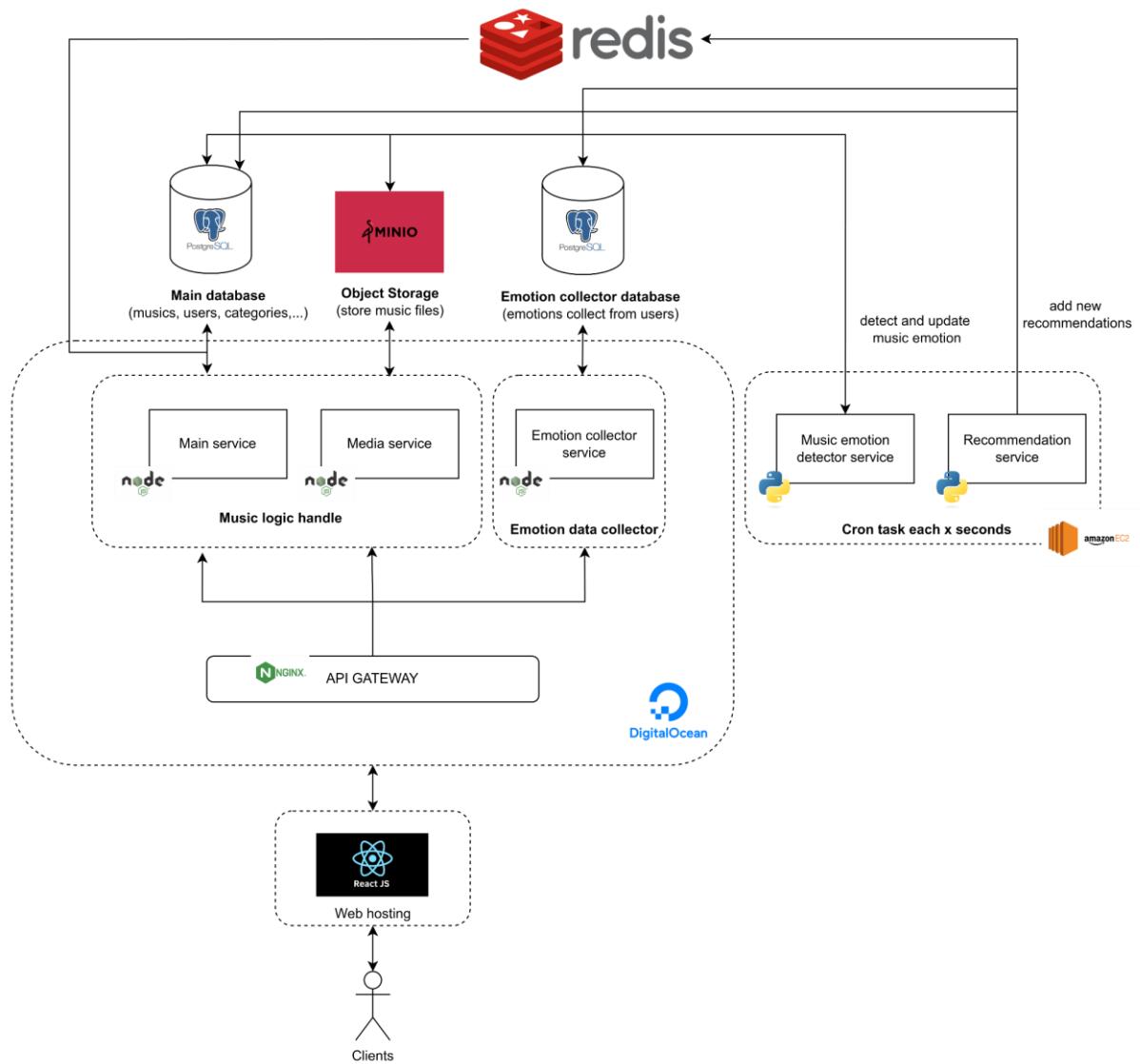
Bảng 3.32. Đặc tả use case xóa nghệ sĩ

Use Case ID	UC26
Use Case Name	Xóa nghệ sĩ
Description	Hỗ trợ admin xóa nghệ sĩ không còn phù hợp khỏi hệ thống.
Actor(s)	AD
Priority	Thấp
Trigger	Admin chọn nghệ sĩ và nhấn "Delete" trên giao diện quản lý.

Pre-condition(s)	<ul style="list-style-type: none"> Admin đã đăng nhập. Nghệ sĩ tồn tại và không có album nào đang sử dụng.
Post-condition(s)	Nghệ sĩ được xóa khỏi hệ thống.
Basic Flow	<ol style="list-style-type: none"> Hệ thống hiển thị danh sách nghệ sĩ. Admin chọn nghệ sĩ và nhấn "Delete". Hệ thống xác nhận và xóa khỏi cơ sở dữ liệu. Kết thúc khi xóa thành công.
Alternate Flow	N/A
Exception Flow	<p>Nếu nghệ sĩ có album đang sử dụng, hiển thị cảnh báo và hủy xóa.</p> <p>Nếu lỗi xóa, hiển thị lỗi và thử lại.</p>
Business Rules	N/A

4.3. Thiết kế hệ thống

4.3.1. Kiến trúc hệ thống



Hình 3.2. Sơ đồ kiến trúc hệ thống

4.3.2. Bảng mô tả thành phần

Bảng 3.33. Bảng mô tả các thành phần trong hệ thống

STT	Thành phần	Mô tả
1	Client	Đối tượng sử dụng cuối cùng, tương tác với hệ thống thông qua trình duyệt web trên thiết bị cá nhân có webcam.
2	Web Hosting (React.js)	Giao diện người dùng (Front-end) dạng Single Page Application, chịu trách nhiệm hiển thị, quản lý tương tác và nhận diện cảm xúc.
3	API Gateway (NGINX)	Cổng giao tiếp duy nhất, điều hướng các yêu cầu từ client đến các service phù hợp, quản lý rate-limit và bảo mật.
4	Main service (Node.js)	Xử lý các nghiệp vụ chính của ứng dụng nghe nhạc như quản lý người dùng, bài hát, danh mục và tìm kiếm.
5	Media service (Node.js)	Chuyên xử lý các tác vụ liên quan đến media, bao gồm tải lên, tải xuống và streaming file nhạc.
6	Emotion collector service (Node.js)	Thu thập và ghi nhận dữ liệu cảm xúc và các sự kiện tương tác do người dùng gửi lên.
7	Music emotion detector service (Python)	Tác vụ nền (cron job) chạy định kỳ để phân tích và tự động gán nhãn cảm xúc cho các bài hát trong hệ thống.
8	Recommendation service (Python)	Tác vụ nền (cron job) sử dụng thuật toán Hybrid để tạo danh sách gợi ý cá nhân hóa cho từng người dùng dựa trên cảm xúc.
9	Main database (PostgreSQL)	Cơ sở dữ liệu chính, lưu trữ dữ liệu có cấu trúc và lâu dài như thông tin bài hát, người dùng, thể loại.
10	Emotion collector database (PostgreSQL)	Cơ sở dữ liệu chuyên dụng để lưu trữ lịch sử tương tác và dữ liệu cảm xúc của người dùng.
11	Object Storage (MinIO)	Hệ thống lưu trữ đối tượng, chuyên dùng để lưu trữ các file media (nhạc, ảnh bìa) có dung lượng lớn.

12	Redis	Hệ thống cache trong bộ nhớ, dùng để lưu trữ dữ liệu cần truy xuất nhanh như danh sách gợi ý đã tính toán và trạng thái cảm xúc.
13	Amazon EC2	Nền tảng hạ tầng đám mây nơi các thành phần của hệ thống được triển khai.
14	Digital Ocean	Nền tảng hạ tầng đám mây nơi các thành phần của hệ thống được triển khai.

4.3.2.1. Music Logic Handle

Đây là khối nghiệp vụ cốt lõi, chịu trách nhiệm cho các chức năng cơ bản của một ứng dụng nghe nhạc. Khối này được xây dựng theo kiến trúc microservices với các service con đảm nhiệm các vai trò riêng biệt.

- + Main service: Được phát triển bằng Node.js, service này là trung tâm xử lý logic chính. Nó cung cấp các API để quản lý xác thực và phân quyền người dùng (đăng nhập, đăng ký), thực hiện các thao tác CRUD (Tạo, Đọc, Cập nhật, Xóa) cho các tài nguyên như bài hát, nghệ sĩ, thể loại, và xử lý các chức năng tìm kiếm. Service này giao tiếp trực tiếp với Main Database (PostgreSQL) để đảm bảo tính toàn vẹn và nhất quán của dữ liệu. Ngoài ra, nó cũng truy xuất danh sách gợi ý đã được tính toán trước từ Redis để trả về cho người dùng.
- + Media service: Service này cũng được xây dựng bằng Node.js, được thiết kế chuyên biệt để xử lý các tác vụ liên quan đến file media. Chức năng chính bao gồm xử lý yêu cầu tải lên (upload) bài hát mới và quan trọng nhất là cung cấp khả năng phát nhạc trực tuyến (streaming) một cách hiệu quả. Bằng cách giao tiếp với Object Storage (MinIO), Media service giúp trùm tượng hóa việc lưu trữ file, tối ưu hóa băng thông và giảm tải cho các service nghiệp vụ khác.

4.3.2.2. Emotion Data Collector

Đây là khói đóng vai trò cầu nối giữa tương tác thời gian thực của người dùng và hệ thống thu thập dữ liệu cho các mô hình AI.

- + Emotion collector service: Service này là một điểm cuối (endpoint) chuyên dụng, làm nhiệm vụ nhận và thu thập dữ liệu cảm xúc do phía Front-end gửi lên. Mỗi khi người dùng có một tương tác (bắt đầu nghe, bỏ qua, thích,...) hoặc khi hệ thống tổng hợp được cảm xúc chủ đạo, service này sẽ tiếp nhận, chuẩn hóa và lưu trữ các bản ghi này vào Emotion collector database (PostgreSQL). Dữ liệu này là nguồn đầu vào cực kỳ quan trọng để Recommender service huấn luyện và cá nhân hóa gợi ý.

4.3.2.3. Tác vụ Nền Định kỳ (Cron task each x seconds)

Đây là bộ não xử lý ngoại tuyến (offline) của hệ thống, bao gồm các tác vụ nặng về tính toán và được lập lịch chạy định kỳ để không ảnh hưởng đến trải nghiệm thời gian thực của người dùng.

- + Music emotion detector service: Service này được viết bằng Python, chạy định kỳ để làm giàu dữ liệu cho kho nhạc. Nó truy vấn Main database để tìm các bài hát chưa được gán nhãn cảm xúc, sau đó truy cập vào file nhạc tương ứng trong Object Storage (MinIO). Dựa trên việc phân tích các đặc trưng âm học, một mô hình học máy sẽ dự đoán và gán nhãn cảm xúc cho bài hát đó.
- + Recommendation service: Đây là trái tim của hệ thống gợi ý. Service Python này tổng hợp dữ liệu từ hai nguồn: thông tin bài hát từ Main database và dữ liệu tương tác cảm xúc từ Emotion collector database. Dựa trên bộ dữ liệu này, nó áp dụng thuật toán Hybrid để xây dựng các mô hình gợi ý riêng cho từng loại cảm

xúc. Sau khi tính toán xong, danh sách gợi ý được cá nhân hóa cho từng người dùng sẽ được đẩy vào Redis để chờ được phục vụ.

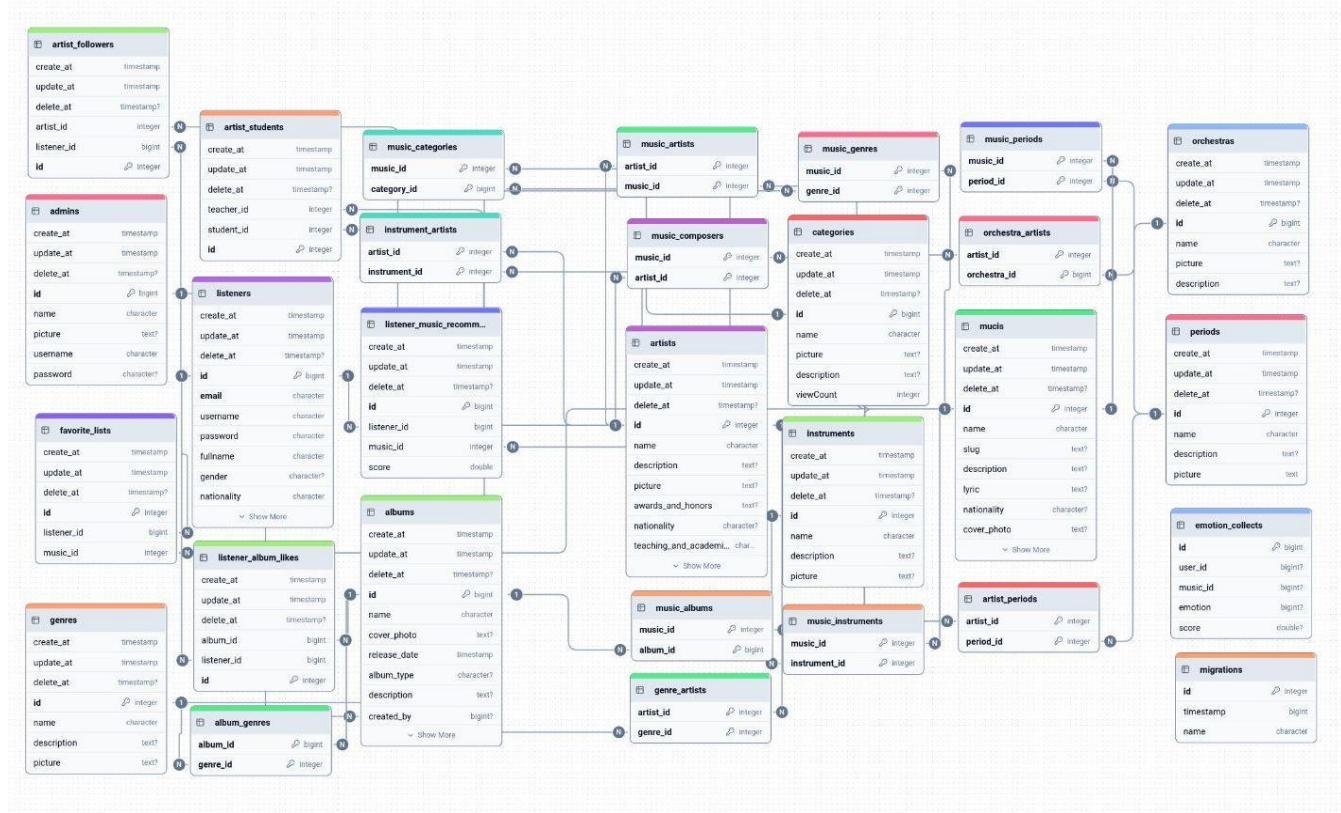
4.3.2.4. Hạ tầng Kỹ thuật và Lưu trữ (Infrastructure)

Đây là nền tảng vật lý và phần mềm hỗ trợ toàn bộ hệ thống hoạt động một cách ổn định và hiệu quả.

- + API Gateway (NGINX): Đóng vai trò là lớp cổng giao tiếp duy nhất giữa client và hệ thống back-end. NGINX chịu trách nhiệm điều hướng (routing) các yêu cầu đến đúng microservice, cân bằng tải (load balancing), giới hạn tần suất yêu cầu (rate limiting) để chống tấn công, và xử lý mã hóa SSL.
- + Redis: Một hệ thống cache hiệu năng cao hoạt động trên bộ nhớ RAM. Trong kiến trúc này, Redis có hai vai trò chính: lưu trữ các danh sách gợi ý đã được tính toán trước để giảm độ trễ phản hồi, và lưu trữ trạng thái cảm xúc hiện tại của người dùng để truy vấn nhanh.
- + Main database (PostgreSQL): Lưu trữ các dữ liệu lõi, có cấu trúc và yêu cầu tính toàn vẹn cao của ứng dụng.
- + Emotion collector database (PostgreSQL): Tách biệt việc lưu trữ dữ liệu tương tác ra khỏi CSDL chính giúp tối ưu hóa cho các tác vụ ghi (write-intensive) và tránh ảnh hưởng đến hiệu năng của các nghiệp vụ chính.
- + Object Storage (MinIO): Là giải pháp lưu trữ các file media dưới dạng đối tượng. Việc sử dụng MinIO giúp dễ dàng mở rộng dung lượng lưu trữ, quản lý file hiệu quả và tối ưu hóa chi phí so với việc lưu file trực tiếp trong CSDL.

4.4. Thiết kế dữ liệu

4.4.1. Sơ đồ cơ sở dữ liệu (Database Diagram)



Hình 3.3. Sơ đồ cơ sở dữ liệu

4.4.2. Mô tả các Table trong cơ sở dữ liệu

Bảng 3.34. Các trường trong collection "migrations"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	serial	Khóa chính, tự động tăng
2	email	bigint	Thời gian thực hiện migration
3	name	varchar	Tên của migration

Bảng 3.35. Các trường trong collection "admins"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	bigserial	Khóa chính, tự động tăng
2	name	varchar(30)	Tên admin
3	picture	text	Ảnh đại diện
4	username	varchar(70)	Tên đăng nhập
5	password	varchar(100)	Mật khẩu
6	update_at	timestamp	Thời gian cập nhật
7	delete_at	timestamp	Thời gian xóa (mềm)
8	created_at	timestamp	Thời gian tạo

Bảng 3.36. Các trường trong collection "categories"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	bigserial	Khóa chính, tự động tăng
2	name	varchar(50)	Tên danh mục
3	picture	text	Ảnh danh mục
4	description	text	Mô tả danh mục
5	viewCount	integer	Số lượt xem
6	create_at	timestamp	Thời gian tạo
7	update_at	timestamp	Thời gian cập nhật
8	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.37. Các trường trong collection "periods"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	serial	Khóa chính, tự động tăng
2	name	varchar(100)	Tên giai đoạn
3	description	text	Mô tả giai đoạn
4	picture	text	Ảnh giai đoạn
5	create_at	timestamp	Thời gian tạo
6	update_at	timestamp	Thời gian cập nhật
7	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.38. Các trường trong trong “music”

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	serial	Khóa chính, tự động tăng
2	name	varchar(100)	Tên bài nhạc
3	slug	text	Đường dẫn tĩnh
4	description	text	Mô tả bài nhạc
5	lyric	text	Lời bài hát
6	nationality	varchar(50)	Quốc tịch
7	cover_photo	text	Ảnh bìa
8	resource_link	varchar	Liên kết tài nguyên
9	releaseYear	integer	Năm phát hành
10	listenCount	integer	Số lượt nghe
11	favoriteCount	integer	Số lượt yêu thích
12	create_at	timestamp	Thời gian tạo
13	update_at	timestamp	Thời gian cập nhật
14	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.39. Các trường trong collection "instruments"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	String	ID của người dùng
2	name	varchar(50)	Tên nhạc cụ
3	description	text	Mô tả nhạc cụ
4	picture	text	Ảnh nhạc cụ
5	create_at	timestamp	Thời gian tạo
6	update_at	timestamp	Thời gian cập nhật
7	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.40. Các trường trong collection "orchestras"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	String	ID của người dùng
2	name	varchar(50)	Tên dàn nhạc

3	description	text	Mô tả dàn nhạc
4	picture	text	Ảnh dàn nhạc
5	create_at	timestamp	Thời gian tạo
6	update_at	timestamp	Thời gian cập nhật
7	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.41. Các trường trong collection "artists"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	serial	Khóa chính, tự động tăng
2	name	varchar(30)	Tên nghệ sĩ
3	description	text	Mô tả nghệ sĩ
4	picture	text	Ảnh nghệ sĩ
5	awards_and_honors	text	Giải thưởng và danh hiệu
6	nationality	varchar(50)	Quốc tịch
7	teaching_and_academic_contributions	varchar(150)	Đóng góp giảng dạy/học thuật
8	significant_performance	text	Buổi biểu diễn nổi bật
9	roles	text[]	Vai trò của nghệ sĩ
10	date_of_birth	date	Ngày sinh
11	date_of_death	date	Ngày mất
12	viewCount	integer	Số lượt xem
13	followers	integer	Số người theo dõi
14	create_at	timestamp	Thời gian tạo
15	update_at	timestamp	Thời gian cập nhật
16	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.42. Các trường trong collection "artist_students"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	teacher_id	integer	ID giáo viên (nghệ sĩ)
2	student_id	integer	ID học sinh (nghệ sĩ)
3	id	serial	Khóa chính, tự động tăng
4	create_at	timestamp	Thời gian tạo

5	update_at	timestamp	Thời gian cập nhật
6	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.43. Các trường trong collection "genres"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	serial	Khóa chính, tự động tăng
2	name	varchar(100)	Tên thể loại
3	description	text	Mô tả thể loại
4	picture	text	Ảnh thể loại
5	create_at	timestamp	Thời gian tạo
6	update_at	timestamp	Thời gian cập nhật
7	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.44. Các trường trong collection "albums"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	bigserial	Khóa chính, tự động tăng
2	name	varchar(50)	Tên album
3	cover_photo	text	Ảnh bìa album
4	release_date	timestamp	Ngày phát hành
5	album_type	varchar(50)	Loại album
6	description	text	Mô tả album
7	viewCount	integer	Số lượt xem
8	likeCount	integer	Số lượt thích
9	created_by	bigint	ID admin tạo album
10	create_at	timestamp	Thời gian tạo
11	update_at	timestamp	Thời gian cập nhật
12	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.45. Các trường trong collection "listeners"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả

1	id	bigserial	Khóa chính, tự động tăng
2	email	varchar(100)	Email người nghe (duy nhất)
3	username	varchar(70)	Tên đăng nhập
4	password	varchar(100)	Mật khẩu
5	fullname	varchar(50)	Họ tên đầy đủ
6	gender	varchar(10)	Giới tính
7	nationality	varchar(50)	Quốc tịch
8	birthdate	timestamp	Ngày sinh
9	create_at	timestamp	Thời gian tạo
10	update_at	timestamp	Thời gian cập nhật
11	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.46. Các trường trong collection "favorite_lists"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	serial	Khóa chính, tự động tăng
2	listener_id	bigint	ID người nghe
3	music_id	integer	ID bài nhạc
4	create_at	timestamp	Thời gian tạo
5	update_at	timestamp	Thời gian cập nhật
6	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.47. Các trường trong collection "listener_album_likes"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	serial	Khóa chính, tự động tăng
2	listener_id	bigint	ID người nghe
3	album_id	bigint	ID album
4	create_at	timestamp	Thời gian tạo
5	update_at	timestamp	Thời gian cập nhật
6	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.48. Các trường trong collection "artist_followers"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	serial	Khóa chính, tự động tăng
2	listener_id	bigint	ID người nghe
3	artist_id	integer	ID nghệ sĩ
4	create_at	timestamp	Thời gian tạo
5	update_at	timestamp	Thời gian cập nhật
6	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.49. Các trường trong collection "listener_music_recommend_score"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	id	bigserial	Khóa chính, tự động tăng
2	listener_id	bigint	ID người nghe
3	music_id	integer	ID bài nhạc
4	score	double precision	Điểm gợi ý
5	create_at	timestamp	Thời gian tạo
6	update_at	timestamp	Thời gian cập nhật
7	delete_at	timestamp	Thời gian xóa (mềm)

Bảng 3.50. Các trường trong collection "music_albums"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	music_id	integer	ID bài nhạc
2	album_id	bigint	ID album

Bảng 3.51. Các trường trong collection "music_genres"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	music_id	integer	ID bài nhạc
2	genre_id	integer	ID thể loại

Bảng 3.52. Các trường trong collection "music_periods"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	music_id	integer	ID bài nhạc
2	period_id	integer	ID giai đoạn

Bảng 3.53. Các trường trong collection "music_categories"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	music_id	integer	ID bài nhạc
2	category_id	bigint	ID danh mục

Bảng 3.54. Các trường trong collection "music_composers"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	music_id	integer	ID bài nhạc
2	artist_id	integer	ID nhạc sĩ

Bảng 3.55. Các trường trong collection "music_artists"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	artist_id	integer	ID nghệ sĩ
2	music_id	integer	ID bài nhạc

Bảng 3.56. Các trường trong collection "artist_periods"

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	artist_id	integer	ID nghệ sĩ

2	period_id	integer	ID giai đoạn
---	-----------	---------	--------------

Bảng 3.57. Các trường trong “orchestra_artists”

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	artist_id	integer	ID nghệ sĩ
2	orchestra_id	bigint	ID dàn nhạc

Bảng 3.58. Các trường trong “genre_artists”

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	artist_id	integer	ID nghệ sĩ
2	genre_id	integer	ID thể loại

Bảng 3.59. Các trường trong “instrument_artists”

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	artist_id	integer	ID nghệ sĩ
2	instrument_id	integer	ID nhạc cụ

Bảng 3.60. Các trường trong “album_genres”

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả
1	album_id	bigint	ID album
2	genre_id	integer	ID thể loại

Bảng 3.61. Các trường trong “emotion_collects”

STT	Tên thuộc tính	Kiểu dữ liệu	Mô tả

1	id	bigserial	Khóa chính, tự động tăng
2	user_id	bigint	ID người dùng
3	music_id	bigint	ID bài nhạc
4	emotion	integer	Loại cảm xúc
5	score	double precision	Điểm cảm xúc

4.5. Thiết kế giao diện

4.5.1. Danh sách các màn hình

Bảng 3.62. Bảng danh sách màn hình

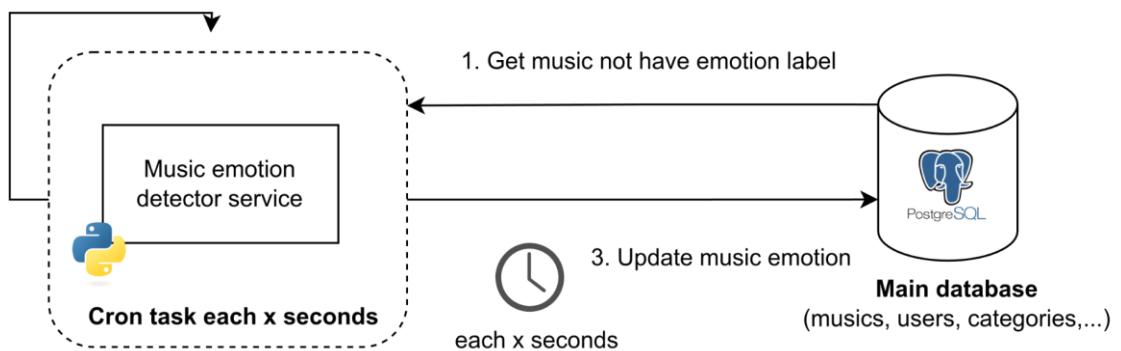
STT	Tên màn hình	Mô tả	Giao diện
1	Home	Hiển thị thông tin về ứng dụng và trình bày các chức năng được tích hợp bên trong.	home
2	Log in	Cho phép người dùng đăng nhập thông qua tài khoản Email hoặc Google.	login
3	Sign up	Người dùng có thể đăng ký thông qua Email là tên đăng nhập hoặc đăng ký thông qua tài khoản Google.	signup
4	Forgot Password	Cho phép người dùng lấy lại mật khẩu bằng việc nhập vào Email đã đăng ký trước đó.	forgot-password
5	Reset Password	Sau khi quên mật khẩu thông qua Email, người dùng được điều hướng tới trang cập nhật mật khẩu mới và bắt đầu đăng nhập lại bằng mật khẩu này.	reset-password
6	Choose mode	Hiển thị các chế độ nghe nhạc cho người dùng có thể chọn lựa nhằm tối ưu hóa trải nghiệm người dùng	choose-mode
7	Current queue	Hiển thị danh sách nhạc hiện tại của người dùng. Các bài hát được hệ thống gợi ý sẽ được tự động thêm vào danh sách này. Người dùng có thể loại bỏ những bài hát mình muốn khỏi danh sách	current-queue
8	My favourite	Hiển thị danh sách những bài hát yêu thích của người dùng. Người dùng có thể thêm chúng vào danh sách nghe hiện tại.	my-favorite

9	Explore	Trang khám phá cho phép người dùng tìm kiếm bài hát cũng như khám phá các nghệ sĩ, album, thể loại nhạc phổ biến.	explore
10	Settings	Người dùng có thể điều chỉnh các thông số của ứng dụng như khoảng thời gian gợi ý bài hát, khoảng thời gian giữa những lần nhận diện cảm xúc, ...	settings
11	View musics	Admin có thể xem được danh sách các bài nhạc hiện tại đang có trong hệ thống.	view-musics
12	Add music	Admin có thể thêm một bài nhạc mới vào hệ thống.	add-music
13	Edit music	Admin có thể cập nhật thông tin chi tiết của một bài hát đang có trong hệ thống.	edit-music
14	View albums	Admin có thể xem được danh sách các album hiện tại đang có trong hệ thống.	view-albums
15	Add album	Admin có thể thêm một album mới vào hệ thống.	add-album
16	Edit album	Admin có thể cập nhật thông tin chi tiết của một album đang có trong hệ thống.	edit-album
17	View artists	Admin có thể xem được danh sách các nghệ sĩ hiện tại đang có trong hệ thống.	view-artists
18	Add artist	Admin có thể thêm một nghệ sĩ mới vào hệ thống.	add-artist
19	Edit artist	Admin có thể cập nhật thông tin chi tiết của một nghệ sĩ đang có trong hệ thống.	edit-artist
20	View genres	Admin có thể xem được danh sách các thể loại hiện tại đang có trong hệ thống.	view-genres
21	Add genre	Admin có thể thêm một thể loại mới vào hệ thống.	add-genre
22	Edit genre	Admin có thể cập nhật thông tin chi tiết của một thể loại đang có trong hệ thống.	edit-genre

4.6. Thiết kế luồng Recommendations

4.6.1. Gán nhãn music emotion

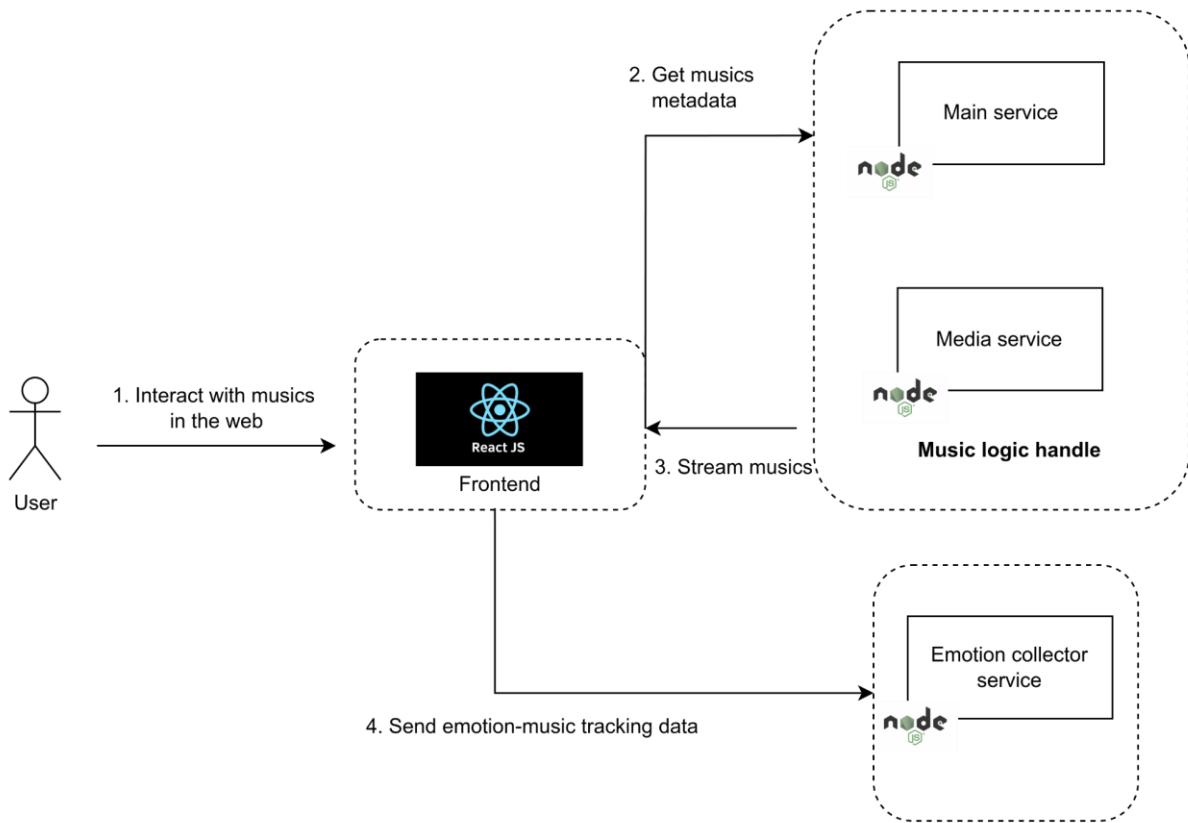
2. Processing and detecting
music's emotion of muscs in database



Mô tả:

1. Music emotion detector service là một cron job, sẽ chạy mỗi x giây, thực hiện lấy dữ liệu các bài hát trong database mà chưa được dán nhãn emotion.
2. Thực hiện xử lý và detect ra loại emotion của các bài hát
3. Cập nhật emotion phát hiện được cho các bài hát.

4.6.2. Gửi user interact emotion data



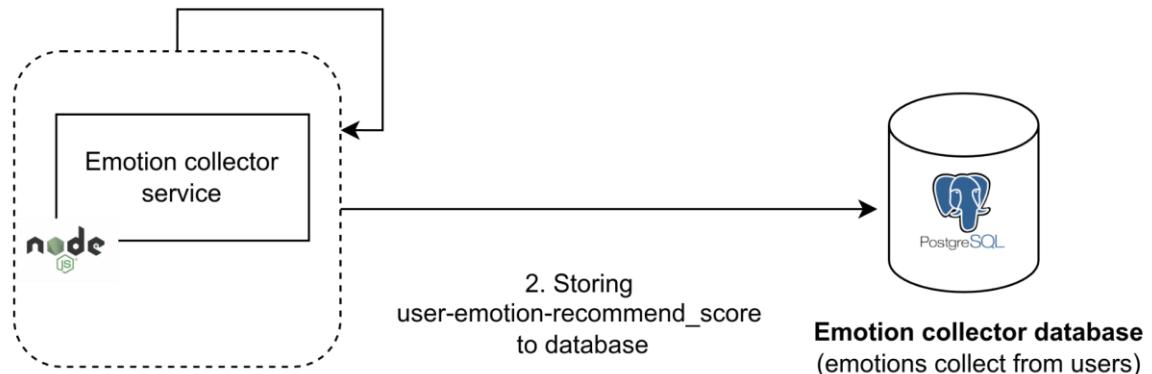
Tại bước này, ta có một luồng xử lý khi một user - người dùng cuối tương tác với web.

Mô tả:

1. Người dùng tương tác với web
2. Frontend lấy ra dữ liệu các bài hát để trả cho người dùng
3. Frontend gọi tới Media Service để stream nhạc cho người dùng
4. Đồng thời, với mỗi hành động, frontend sẽ tracking các hành động như: chọn bài, chuyển bài, like bài,... kết hợp với dữ liệu emotion của người dùng tại thời điểm đó, sau đó gửi thông tin cho Emotion collector service.

4.6.3. Thu thập và xử lý user interact emotion data

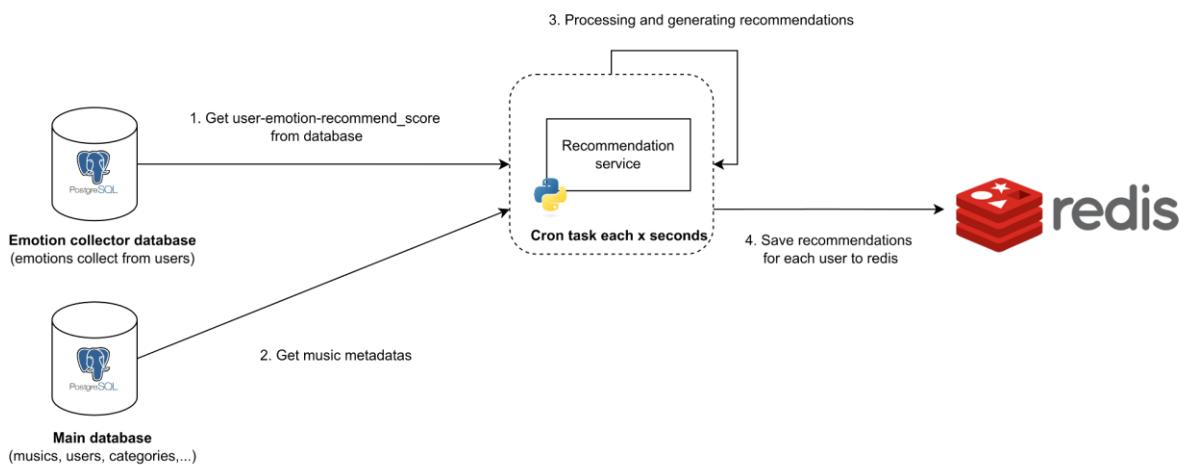
1. Processing and calculate recommend score



Mô tả:

1. Emotion collector service sau khi nhận được dữ liệu emotion tracking từ frontend, sẽ tiến hành tiền xử lý, tính toán recommend score (điểm số khuyến nghị của một người dùng đối với một bài hát, ứng với một cảm xúc cụ thể).
2. Lưu dữ liệu user-emotion-recommend_score vào database.

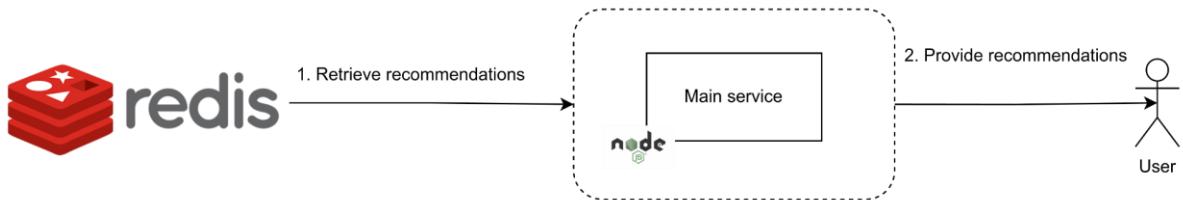
4.6.4. Tạo và lưu recommendations



Mô tả:

1. Lấy ra dữ liệu user-emotion-recommend_score từ database
2. Lấy ra dữ liệu nhạc từ database
3. Hai loại dữ liệu trên sẽ được đưa vào tiền xử lý, sau đó dùng để chạy thuật toán Hybrid, nhằm tạo recommendations cho từng user trong hệ thống.
4. Lưu recommendations đã tạo được vào Redis.

4.6.5. Lấy recommendations cho user



Mô tả:

1. Main service sẽ tiến hành lấy ra recommendations data từ Redis
2. Main service sẽ cung cấp những bài hát được gợi ý từ dữ liệu recommendations cho user.

CHƯƠNG 5. TRIỂN KHAI ỨNG DỤNG

5.1. Triển khai Frontend

5.1.1. Quy trình triển khai Frontend

1. Đảm bảo mã nguồn đã được đẩy lên repository trên GitHub.
2. Đảm bảo đã khởi tạo Docker Hub Repository
3. Đảm bảo đã cấu hình biến môi trường, các secret repository để Github Actions có thể sử dụng để triển khai luồng CI/CD.

4. Đảm bảo đã config Nginx để routing request đến đúng port của frontend.
5. Đảm bảo đã cấu hình Cloudflare để triển khai DNS mapping, DDOS protection, Bot detection,...
6. Đảm bảo đã viết Dockerfile để Dockerize ứng dụng ReactJS để có thể deploy trên môi trường Docker.
7. Xây dựng Github Actions Workflows Pipeline để tạo luồng CI/CD tự động deploy Frontend lên server mỗi khi có pull request.

```
1 name: Frontend CI/CD Pipeline
2
3 env:
4   IMAGE_NAME: haphuthinh/emobeat-fe:latest
5
6 on:
7   push:
8     branches:
9       - release
10
11 jobs:
12   build_and_publish:
13     runs-on: ubuntu-latest
14     steps:
15       - name: Checkout code
16         uses: actions/checkout@v3
17
18       - name: Log in to Docker Hub
19         run: echo "${{ secrets.DOCKER_HUB_PASSWORD }} " | docker login -u "${{ secrets.
20           DOCKER_HUB_USERNAME }} " --password-stdin {{}}
21
22       - name: Build Docker image
23         run: docker build -t $IMAGE_NAME .
24
25       - name: Push Docker image to Docker Hub
26         run: docker push $IMAGE_NAME
27
28 deploy:
29   runs-on: ubuntu-latest
30   needs: build_and_publish
31
32   steps:
33     - name: Checkout code
34       uses: actions/checkout@v3
35
36     - name: Copy source code to server
37       uses: appleboy/scp-action@v0.1.1
38       with:
39         host: ${{ secrets.SERVER_HOST }}
40         username: ${{ secrets.SERVER_USER }}
41         password: ${{ secrets.SERVER_PASSWORD }}
42         source: "./**"
43         target: "/root/emobeat/frontend"
44
45     - name: Deploy to server
46       uses: appleboy/ssh-action@v0.1.4
47       with:
48         host: ${{ secrets.SERVER_HOST }}
49         username: ${{ secrets.SERVER_USER }}
50         password: ${{ secrets.SERVER_PASSWORD }}
51         port: 22
52         script: |
53           cd /root/emobeat/frontend/
54           sudo docker compose down
55           sudo docker compose pull
56           sudo docker compose up -d
```

Ảnh: Github Action Workflows của dự án, pipeline này sẽ thực hiện: mỗi khi có pull request vào nhánh release, sẽ lấy code mới về, build docker image sau đó push lên docker hub. Sau đó sẽ thực hiện pull Docker Image mới về server, thực hiện chạy frontend docker mới và tắt frontend docker cũ.

5.1.2. Lợi ích khi triển khai với Cloudflare:

Mạng lưới phân phối nội dung (CDN) toàn cầu: Cloudflare sở hữu một trong những mạng lưới CDN lớn và nhanh nhất thế giới. Nội dung tĩnh của trang web (như HTML, CSS, JavaScript, và hình ảnh) sẽ được lưu trữ (cached) tại các máy chủ gần với người dùng nhất. Điều này giúp giảm thiểu độ trễ mạng (latency) và tăng tốc độ tải trang một cách đáng kể, dù người dùng ở bất kỳ đâu.

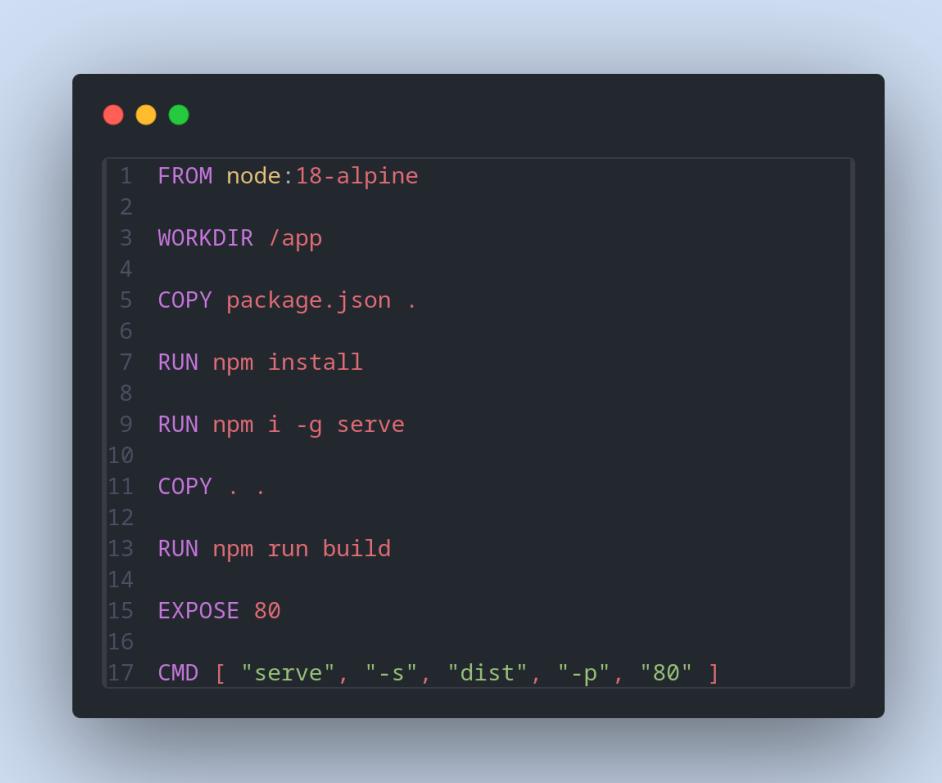
Tối ưu hóa nội dung tự động: Cloudflare tự động nén các tệp tin (ví dụ: Brotli, GZIP), tối ưu hóa hình ảnh (Polish), và rút gọn mã (Auto Minify) HTML, CSS, JavaScript. Những kỹ thuật này giúp giảm kích thước tệp tin phải tải về, giúp trang web nhẹ hơn và tải nhanh hơn.

Chống tấn công DDoS: Cloudflare nổi tiếng với khả năng chống lại các cuộc tấn công từ chối dịch vụ (DDoS) cực lớn. Hệ thống sẽ tự động phát hiện và chặn lưu lượng truy cập độc hại trước khi chúng kịp đến máy chủ.

Tường lửa ứng dụng web (WAF): WAF của Cloudflare giúp bảo vệ trang web khỏi các lỗ hổng bảo mật phổ biến như SQL injection, Cross-Site Scripting (XSS), và nhiều loại tấn công khác.

SSL/TLS miễn phí: Cloudflare cung cấp chứng chỉ SSL/TLS miễn phí, giúp mã hóa kết nối giữa người dùng và trang web (HTTPS). Điều này không chỉ bảo vệ dữ liệu người dùng mà còn là một yếu tố quan trọng trong việc xếp hạng SEO.

5.1.3. Lợi ích khi triển khai với Docker.



```
1 FROM node:18-alpine
2
3 WORKDIR /app
4
5 COPY package.json .
6
7 RUN npm install
8
9 RUN npm i -g serve
10
11 COPY . .
12
13 RUN npm run build
14
15 EXPOSE 80
16
17 CMD [ "serve", "-s", "dist", "-p", "80" ]
```

Ảnh: Dockerfile của Frontend.

- Môi trường nhất quán tuyệt đối (Dev-Prod Parity)
- Đơn giản hóa việc cài đặt và thiết lập
- Tính di động và khả năng triển khai linh hoạt

5.2. Triển khai Backend

5.2.1. Quy trình triển khai Backend

1. Đảm bảo mã nguồn đã được đẩy lên repository trên GitHub.

2. Đảm bảo đã khởi tạo Docker Hub Repository
3. Đảm bảo đã cấu hình biến môi trường, các secret repository để Github Actions có thể sử dụng để triển khai luồng CI/CD.
4. Đảm bảo đã config Nginx để routing request đến đúng port của frontend.
5. Đảm bảo đã cấu hình Cloudflare để triển khai DNS mapping, DDOS protection, Bot detection,...
6. Đảm bảo đã viết Dockerfile để Dockerize ứng dụng backend để có thể deploy trên môi trường Docker.
7. Xây dựng Github Actions Workflows Pipeline để tạo luồng CI/CD tự động deploy backend service lên server mỗi khi có pull request.

5.2.2. Xây dựng blue green deployment

Áp dụng kĩ thuật blue-green deployment để triển khai backend service, bằng cách sử dụng Nginx làm load balancer, khi một service cũ đang hoạt động, ta tiến hành deploy service mới chạy song song với service cũ, sau đó khi service mới đã hoạt động ổn định, ta sẽ dần dần chuyển hướng request và tắt service cũ đi.

5.3. Triển khai recommender services

Các recommender services sẽ được triển khai bằng cách sử dụng conda export environment.xml, sau đó sẽ tạo được một môi trường conda bên trong Docker Container:



A screenshot of a terminal window showing a Dockerfile. The terminal has a dark theme with red, yellow, and green window control buttons at the top. The Dockerfile content is as follows:

```
1 # Giai đoạn 1: Cài đặt Conda và môi trường
2 FROM continuumio/miniconda3 AS builder
3 WORKDIR /env
4 COPY environment.yml .
5 RUN conda env create -f environment.yml && conda clean -afy
6
7 # Giai đoạn 2: Image cuối cùng
8 FROM openjdk:11-jre-slim
9 WORKDIR /app
10 COPY --from=builder /opt/conda /opt/conda
11 ENV PATH=/opt/conda/bin:$PATH
12 COPY . .
13 CMD ["conda", "run", "--no-capture-output", "-n", "EmobeatDetect", "python", "main.py"]
14
```

Các recommender service cũng được xây dựng pipeline tự động build docker image mỗi khi có pull request vào nhánh develop:



```
1 name: "[BUILD DOCKER] Emobeat recommendation"
2
3 env:
4   IMAGE_NAME: haphuthinh/emobeat-recommend:${{ github.ref_name }}-${{ github.sha }}}
5
6 on:
7   push:
8     branches:
9       - release
10
11 jobs:
12   build_and_publish:
13     runs-on: ubuntu-latest
14
15   steps:
16     # Checkout code from the repository
17     - name: Checkout code
18       uses: actions/checkout@v3
19
20     # Log in to Docker Hub
21     - name: Log in to Docker Hub
22       run: echo "${{ secrets.DOCKER_HUB_PASSWORD }} " | docker login -u "${{ secrets.DOCKER_HUB_USERNAME }}"
23       " --password-stdin
24
25     # Build Docker image and push to Docker Hub
26     - name: Build Docker image
27       run: docker build -t $IMAGE_NAME ./recommendation.service
28
29     # Push Docker image to Docker Hub
30     - name: Push Docker image to Docker Hub
31       run: docker push $IMAGE_NAME
```

Sau đó, trên các server AWS EC2 Ubuntu, ta sẽ thực hiện tạo cron task chạy các docker image này, sử dụng docker compose:

```
*/3 * * * * cd /path/to/your/project && docker-compose up -d
```

Crontask trên sẽ chạy mỗi 3 phút.

CHƯƠNG 6. HIỆN THỰC HÓA VÀ KẾT QUẢ

6.1. Nhận diện cảm xúc bài hát

Phần này sẽ hoạt động ở Music Emotion Detector Service, đây là một cron task sẽ chạy sau mỗi x giây (x do người vận hành có thể tùy chỉnh). Nhiệm vụ của nó là lấy danh sách các bài hát mới, chưa được gán nhãn cảm xúc và tiến hành gán nhãn cảm xúc cho các bài hát đó.

Quy trình này đảm bảo rằng mọi bài hát trong hệ thống đều được làm giàu với metadata về cảm xúc, phục vụ trực tiếp cho recommender system.

6.1.1. Huấn luyện mô hình

6.1.1.1. Tiết xử lý dữ liệu



```
1 annotations = pd.read_csv(annotation_path)
2 annotations.columns = annotations.columns.str.strip()
3
4 X =
5 y_valence =
6 y_arousal = []
7 []
8 for index, row in annotations.iterrows():
9     if (index + 1) % 10 == 0:
10         print(f"Đang xử lý bài hát {index + 1}/{len(annotations)}: {row['song_id']}")"
11
12     song_id = row['song_id']
13     mp3_path = os.path.join(audio_dir, f"{int(song_id)}.mp3")
14     if os.path.exists(mp3_path):
15         features = extract_features_from_mp3(mp3_path)
16         if features is not None:
17             X.append(features)
18             y_valence.append(row['valence_mean'])
19             y_arousal.append(row['arousal_mean'])
20         else:
21             print(f"File not found: {mp3_path}")
22
23     # Chuyển thành numpy array
24     X = np.array(X)
25     y_valence = np.array(y_valence)
26     y_arousal = np.array(y_arousal)
27
28     # Lưu đặc trưng
29     print("Lưu đặc trưng đã trích xuất...")
30     np.save(features_file, X)
31     np.save(valence_file, y_valence)
32     np.save(arousal_file, y_arousal)
33
34     print(f"Đã lưu: {X.shape[0]} mẫu")
35 return X, y_valence, y_arousal
```

- + Đọc file `annotations.csv` chứa `song_id` cùng với các giá trị `valence_mean` và `arousal_mean` tương ứng cho mỗi bài hát.
- + Duyệt qua từng bài hát trong file annotation. Với mỗi bài, tìm file MP3 tương ứng, sau đó gọi hàm `extract_features_from_mp3` (đã được mô tả ở phần trước) để trích xuất ra một vector 35 đặc trưng âm thanh.

- + Các vector đặc trưng được lưu vào mảng `X`, trong khi các giá trị valence và arousal được lưu vào `y_valence` và `y_arousal`.

Nếu các file đặc trưng (`features.npy`, `valence.npy`, `arousal.npy`) đã tồn tại từ lần chạy trước, ta sẽ tải trực tiếp chúng lên thay vì phải tốn thời gian trích xuất lại từ đầu.

6.1.1.2. Huấn luyện mô hình



```
# GridSearchCV cho valence
grid_search_valence = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=5, # 5-fold cross-validation
    scoring='neg_mean_squared_error',
    n_jobs=-1,
    verbose=1
)

# GridSearchCV cho arousal
grid_search_arousal = GridSearchCV(
    estimator=rf,
    param_grid=param_grid,
    cv=5,
    scoring='neg_mean_squared_error',
    n_jobs=-1,
    verbose=1
)
```

Ta sử dụng GridSearchCV để tối ưu hóa các siêu tham số.

Sau khi `GridSearchCV` hoàn tất và tìm ra bộ tham số tốt nhất (`best_params_`), script sẽ sử dụng chính bộ tham số này để huấn luyện mô hình cuối cùng.



```
# Huấn luyện mô hình cuối cùng với tham số tốt nhất
model_valence = RandomForestRegressor(**best_params_valence, random_state=42)
model_valence.fit(X_train, y_valence_train)

model_arousal = RandomForestRegressor(**best_params_arousal, random_state=42)
model_arousal.fit(X_train, y_arousal_train)
```

6.1.2. Chạy job gán nhãn

6.1.2.1. Truy vấn các bài hát cần phân tích



```
# 1. Lấy danh sách bài hát chưa có nhãn cảm xúc
# media_id được giả định là object_name trong MinIO
query_fetch = "SELECT id, resource_link FROM mucis WHERE emotion IS NULL AND
resource_link IS NOT NULL"
cursor.execute(query_fetch)
songs_to_process = cursor.fetchall()

if not songs_to_process:
    print("Không có bài hát mới nào cần phân tích cảm xúc.")
    return
```

Đầu tiên, script kết nối vào cơ sở dữ liệu **PostgreSQL** và tìm kiếm tất cả các bài hát chưa có nhãn cảm xúc (`emotion IS NULL`) nhưng đã có đường dẫn tài nguyên (`resource_link IS NOT NULL`). Đây là điểm khởi đầu của toàn bộ quy trình.

6.1.2.2. Tải File Music từ MinIO



```
# 2. Lặp qua từng bài hát để xử lý
for song_id, resource_link in songs_to_process:
    object_name = None
    try:
        parsed_url = urlparse(resource_link)
        query_params = parse_qs(parsed_url.query)
        media_category = query_params.get('mediaCategory', [None])[0]
        file_name = query_params.get('fileName', [None])[0]
        if not media_category or not file_name:
            raise ValueError(
                "URL không chứa 'mediaCategory' hoặc 'fileName'")
        object_name = f"{media_category}/128/{file_name}"
        # 128 tức 128kbps bitrate
        print(f" -> Trích xuất Object Name: '{object_name}'")
    except (ValueError, KeyError, IndexError) as e:
        print(f" -> LỖI: Không thể phân tích object name từ resource_link: '{resource_link}'.
        Lỗi: {e}. Bỏ qua bài hát này.")
        continue # Bỏ qua và xử lý bài hát tiếp theo
    print(f"\nĐang xử lý bài hát ID: {song_id}, Object Name: {object_name}")
    temp_file_path = None
    try:
        # 3. Tải file nhạc từ MinIO
        print(f" -> Đang tải '{object_name}' từ bucket '{MINIO_BUCKET}'...")
        response = minio_client.get_object(MINIO_BUCKET, object_name)
        # Tạo file tạm thời để lưu nội dung nhạc
        with tempfile.NamedTemporaryFile(delete=False, suffix=".mp3") as temp_file:
            temp_file.write(response.read())
            temp_file_path = temp_file.name
    except Exception as e:
        print(f" -> LỖI: Không thể tải '{object_name}' từ MinIO.
        Lỗi: {e}. Bỏ qua bài hát này.")
```

Với mỗi bài hát, ta không làm việc trực tiếp với `resource_link` (một URL cho client). Thay vào đó, ta phân tích URL này để trích xuất ra **tên đối tượng (object name)** trong kho lưu trữ **MinIO**. Sau đó, ta tải file âm thanh (MP3) từ MinIO về một file tạm trên máy chủ để chuẩn bị cho việc phân tích.

6.1.2.3. Trích xuất Đặc trưng Âm thanh

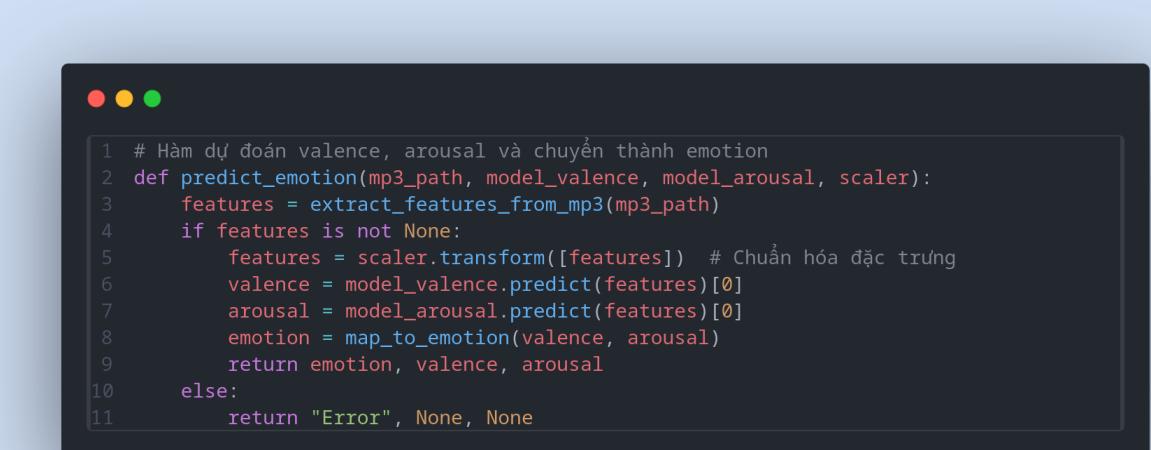


```
1 # Hàm trích xuất đặc trưng từ file MP3
2 def extract_features_from_mp3(mp3_path):
3     try:
4         y, sr = librosa.load(mp3_path, sr=None)
5         mfcc = np.mean(librosa.feature.mfcc(y=y, sr=sr, n_mfcc=13), axis=1) # 13 đặc trưng
6         chroma = np.mean(librosa.feature.chroma_stft(y=y, sr=sr), axis=1) # 12 đặc trưng
7         spectral_contrast = np.mean(librosa.feature.spectral_contrast(y=y, sr=sr), axis=1) # 7 đặc trưng
8         tempo = librosa.beat.tempo(y=y, sr=sr)[0] # 1 đặc trưng
9         rms = np.mean(librosa.feature.rms(y=y)) # 1 đặc trưng
10        zcr = np.mean(librosa.feature.zero_crossing_rate(y=y)) # 1 đặc trưng
11        features = np.concatenate((mfcc, chroma, spectral_contrast, [tempo, rms, zcr]))
12        return features # Tổng cộng 35 đặc trưng
13    except Exception as e:
14        print(f"Error processing {mp3_path}: {e}")
15    return None
```

Đây là cốt lõi của việc phân tích âm nhạc. Ta sử dụng thư viện **librosa** để phân tích file MP3 tạm và trích xuất ra một vector gồm **35 đặc trưng âm thanh** quan trọng, bao gồm:

- **MFCCs (Mel-Frequency Cepstral Coefficients):** Đặc trưng về âm sắc.
- **Chroma:** Đặc trưng về cao độ, hợp âm.
- **Spectral Contrast:** Đặc trưng về sự tương phản trong phổ tần.
- **Tempo, RMS (Root Mean Square), Zero-Crossing Rate:** Đặc trưng về nhịp điệu và năng lượng.

6.1.2.4. Dự đoán cảm xúc bằng Machine Learning

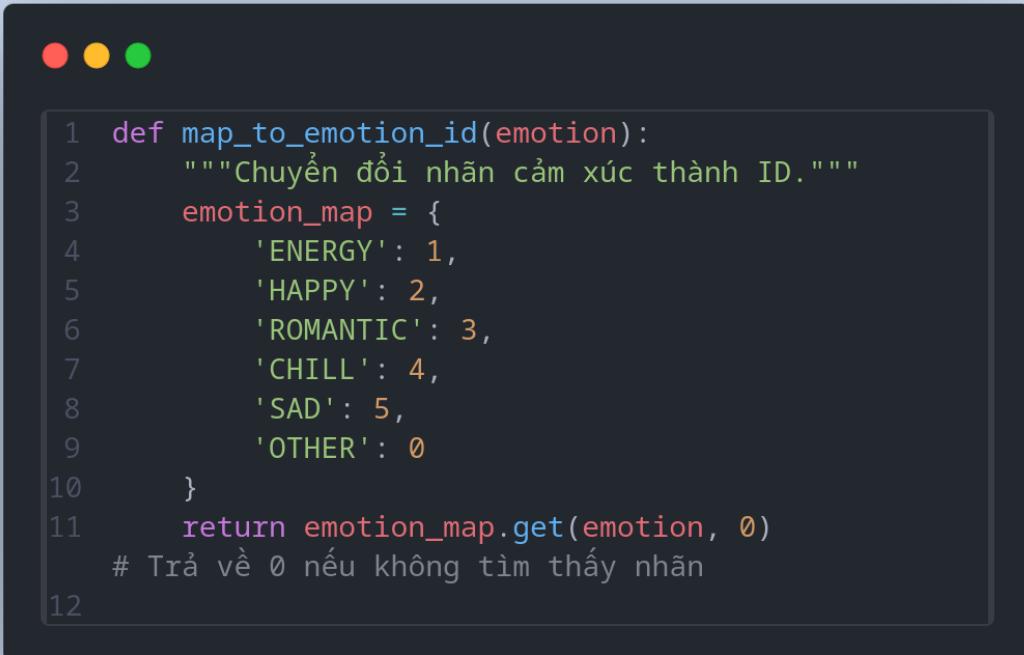


```
1 # Hàm dự đoán valence, arousal và chuyển thành emotion
2 def predict_emotion(mp3_path, model_valence, model_arousal, scaler):
3     features = extract_features_from_mp3(mp3_path)
4     if features is not None:
5         features = scaler.transform([features]) # Chuẩn hóa đặc trưng
6         valence = model_valence.predict(features)[0]
7         arousal = model_arousal.predict(features)[0]
8         emotion = map_to_emotion(valence, arousal)
9         return emotion, valence, arousal
10    else:
11        return "Error", None, None
```

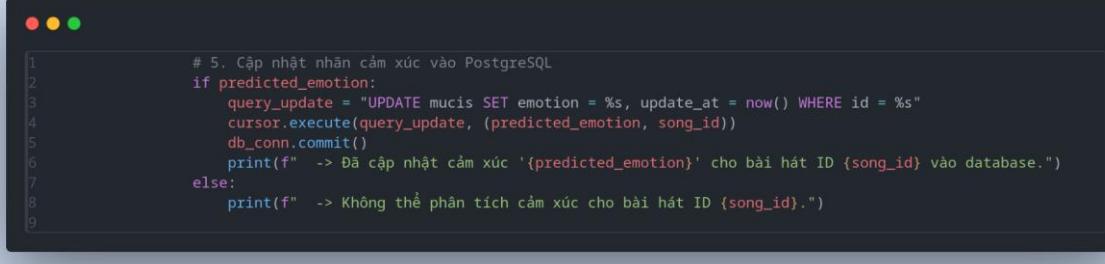
Vector đặc trưng sau đó được đưa qua một **scaler** để chuẩn hóa, rồi được cung cấp cho hai mô hình **Random Forest Regressor** đã được huấn luyện trước (.pk1 files):

1. **model_valence**: Dự đoán mức độ **Valence** (từ tiêu cực đến tích cực).
2. **model_arousal**: Dự đoán mức độ **Arousal** (từ bình tĩnh đến sôi động).

6.1.2.5. Ánh xạ sang nhãn cảm xúc và cập nhật Database



```
1 def map_to_emotion_id(emotion):
2     """Chuyển đổi nhãn cảm xúc thành ID."""
3     emotion_map = {
4         'ENERGY': 1,
5         'HAPPY': 2,
6         'ROMANTIC': 3,
7         'CHILL': 4,
8         'SAD': 5,
9         'OTHER': 0
10    }
11    return emotion_map.get(emotion, 0)
# Trả về 0 nếu không tìm thấy nhãn
12
```



```
1
2
3
4
5     # 5. Cập nhật nhãn cảm xúc vào PostgreSQL
6     if predicted_emotion:
7         query_update = "UPDATE mucis SET emotion = %s, update_at = now() WHERE id = %s"
8         cursor.execute(query_update, (predicted_emotion, song_id))
9         db_conn.commit()
10        print(f" -> Đã cập nhật cảm xúc '{predicted_emotion}' cho bài hát ID {song_id} vào database.")
11    else:
12        print(f" -> Không thể phân tích cảm xúc cho bài hát ID {song_id}.")
```

Cập giá trị (Valence, Arousal) được ánh xạ sang một nhãn cảm xúc rời rạc và dễ hiểu (ví dụ: **HAPPY**, **SAD**, **ENERGY**, **CHILL**). Nhãn này sau đó được chuyển thành một ID số. Cuối cùng, ta thực hiện một câu lệnh **UPDATE** để ghi lại ID cảm xúc này vào đúng hàng của bài hát trong bảng **mucis** của PostgreSQL.

6.2. Nhận diện cảm xúc người dùng

Chúng ta sử dụng thư viện **face-api.js** để thực hiện việc nhận diện ngay trên trình duyệt của người dùng, ngoài ra đảm bảo giảm tải cho backend, đảm bảo quyền riêng tư cho người dùng (dữ liệu video từ webcam không bị gửi lên server).



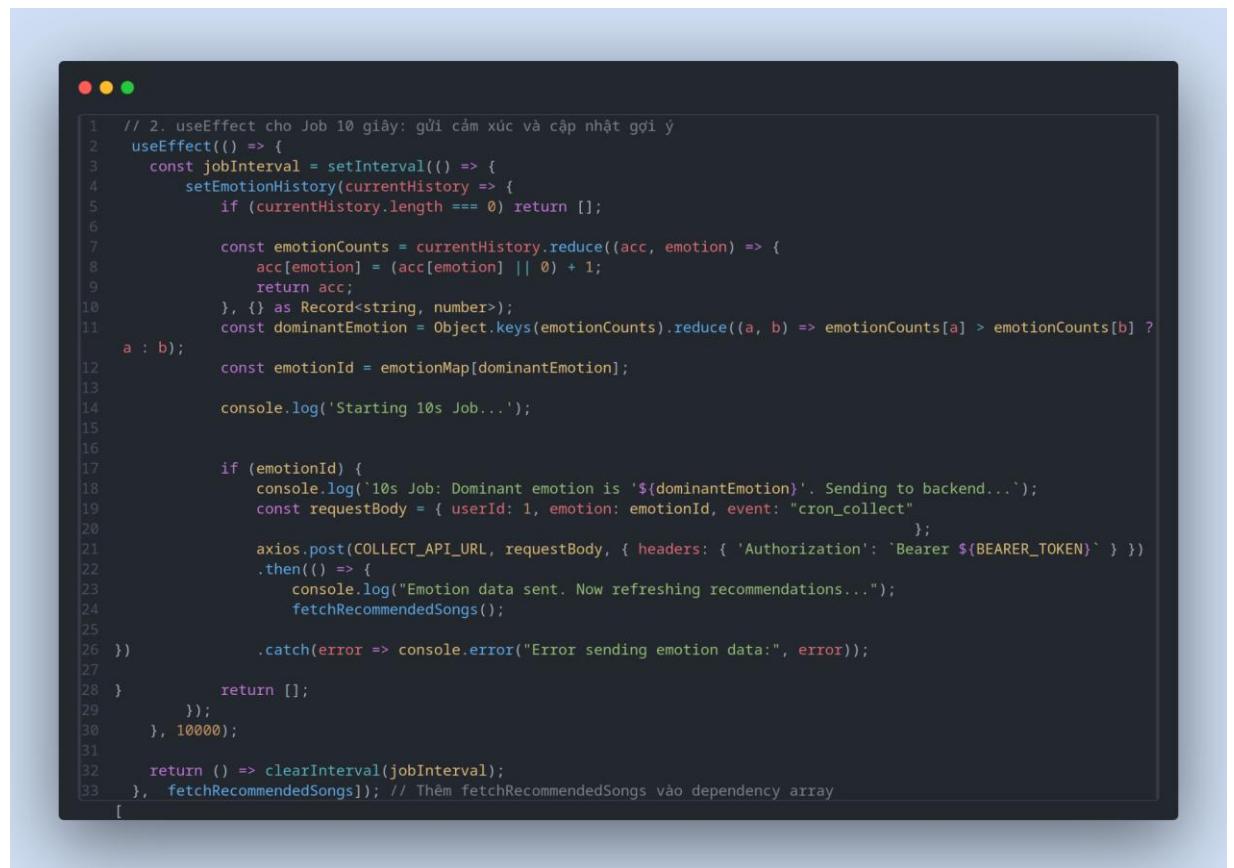
```
1  const handlePlay = () => {
2      detectionIntervalRef.current = setInterval(async () => {
3          if (!videoRef.current || videoRef.current.paused ||
4              videoRef.current.ended) return;
5          const detections = await faceapi.detectSingleFace(
6              videoRef.current, new faceapi.TinyFaceDetectorOptions().
7                  withFaceExpressions());
8          if (detections) {
9              const topEmotion = Object.entries(detections.
10                  expressions).reduce((prev, current) => current[1] > prev[1] ?
11                  current : prev)[0];
12              setCurrentEmotion(topEmotion);
13              setEmotionHistory(prev => [...prev, topEmotion]);
14          } else {
15              setCurrentEmotion("Can't detect face");
16          }
17      }, 500);
18  };
19 }
```

Luồng hoạt động:

1. **Khởi tạo:** Khi component được tải, một **useEffect** hook sẽ được kích hoạt để tải các mô hình machine learning cần thiết và yêu cầu quyền truy cập webcam.

2. **Phát hiện liên tục:** Sau khi người dùng cho phép, một `setInterval` được thiết lập để chạy mỗi 500ms. Trong mỗi chu kỳ, nó sẽ chụp một khung hình từ video, phát hiện khuôn mặt và phân tích các biểu cảm.
3. **Lưu trữ trạng thái:** Cảm xúc có điểm số cao nhất được xác định và cập nhật vào state `currentEmotion` (để hiển thị trên UI) và `emotionHistory` (để phân tích sâu hơn).

Thay vì chỉ dựa vào cảm xúc tại một thời điểm, hệ thống phân tích tâm trạng **chủ đạo** của người dùng trong một khoảng thời gian (10 giây) để có được cái nhìn ổn định hơn.



```

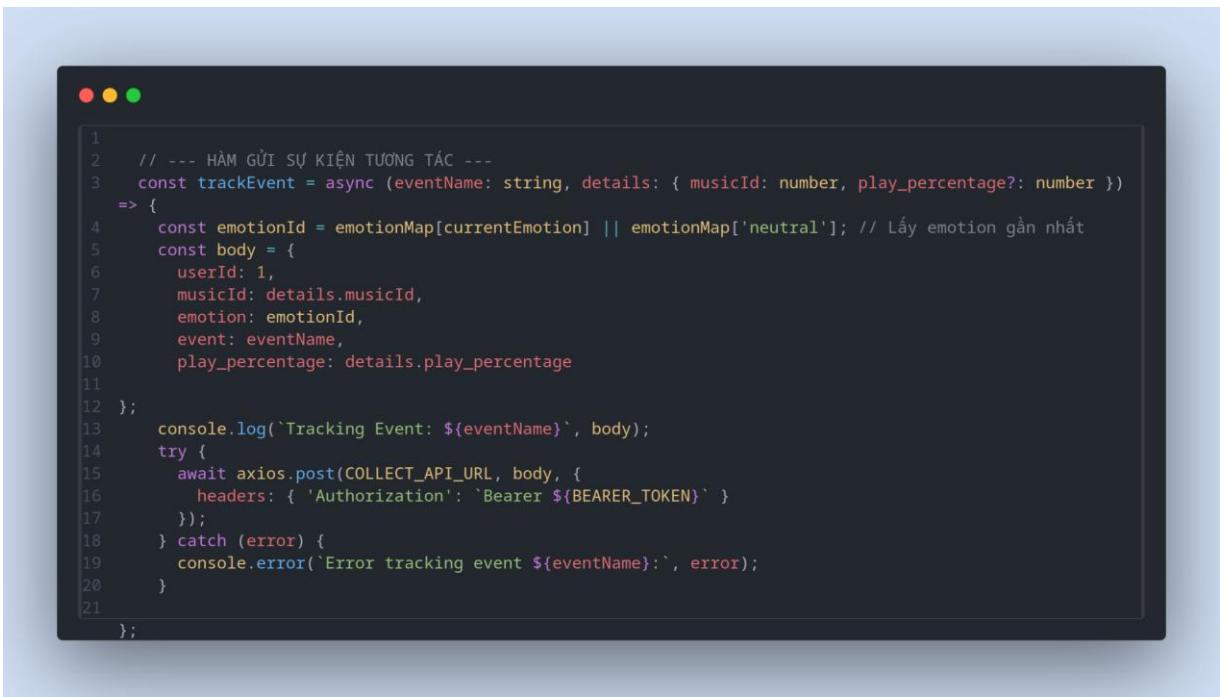
1 // 2. useEffect cho Job 10 giây: gửi cảm xúc và cập nhật gợi ý
2 useEffect(() => {
3   const jobInterval = setInterval(() => {
4     setEmotionHistory(currentHistory => {
5       if (currentHistory.length === 0) return [];
6
7       const emotionCounts = currentHistory.reduce((acc, emotion) => {
8         acc[emotion] = (acc[emotion] || 0) + 1;
9         return acc;
10      }, {} as Record<string, number>);
11      const dominantEmotion = Object.keys(emotionCounts).reduce((a, b) => emotionCounts[a] > emotionCounts[b] ? 
12        a : b);
13      const emotionId = emotionMap[dominantEmotion];
14
15      console.log('Starting 10s Job...');
16
17      if (emotionId) {
18        console.log(`10s Job: Dominant emotion is ${dominantEmotion}. Sending to backend...`);
19        const requestBody = { userId: 1, emotion: emotionId, event: "cron_collect"
20        };
21        axios.post(COLLECT_API_URL, requestBody, { headers: { 'Authorization': `Bearer ${BEARER_TOKEN}` } })
22          .then(() => {
23            console.log("Emotion data sent. Now refreshing recommendations...");
24            fetchRecommendedSongs();
25
26          })
27          .catch(error => console.error("Error sending emotion data:", error));
28      }
29    });
30  }, 10000);
31
32  return () => clearInterval(jobInterval);
33 }, [fetchRecommendedSongs]); // Thêm fetchRecommendedSongs vào dependency array

```

Luồng hoạt động:

1. Một `useEffect` khác thiết lập một `setInterval` chạy mỗi 10 giây.
2. Thực hiện lấy `emotionHistory` đã thu thập trong 10 giây qua và tìm ra cảm xúc nào xuất hiện nhiều nhất (cảm xúc chủ đạo).
3. Cảm xúc chủ đạo này được gửi lên server với sự kiện "`cron_collect`".
4. **Ngay sau khi** gửi dữ liệu thành công, hàm `fetchRecommendedSongs()` được gọi để lấy về một danh sách bài hát mới, đã được thuật toán phía backend điều chỉnh dựa trên dữ liệu cảm xúc vừa nhận được.

6.3. Thu thập và gửi sự kiện



```

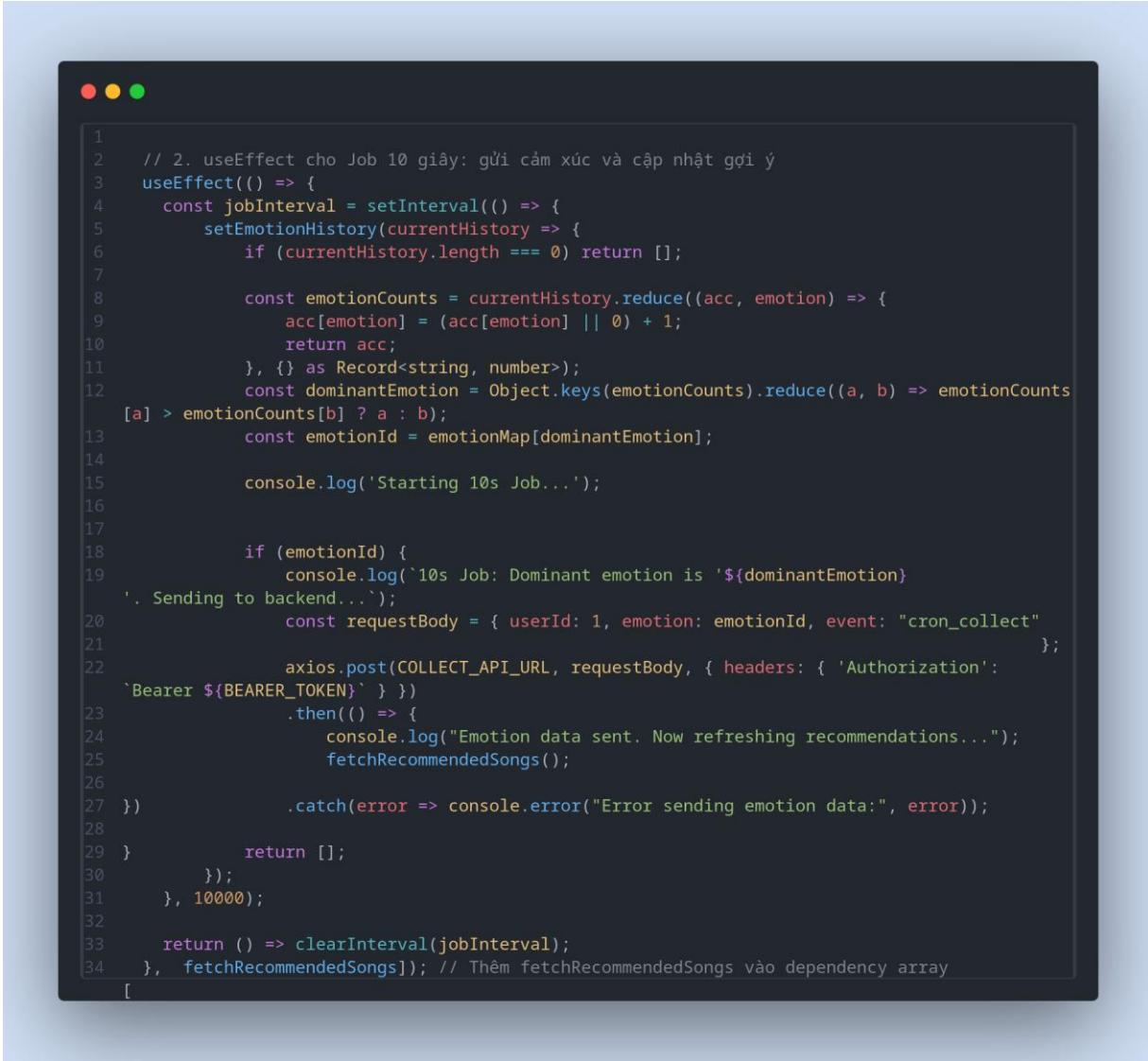
1 // --- HÀM GỬI SỰ KIỆN TƯỞNG TÁC ---
2 const trackEvent = async (eventName: string, details: { musicId: number, play_percentage?: number }) => {
3   const emotionId = emotionMap[currentEmotion] || emotionMap['neutral']; // Lấy emotion gần nhất
4   const body = {
5     userId: 1,
6     musicId: details.musicId,
7     emotion: emotionId,
8     event: eventName,
9     play_percentage: details.play_percentage
10   };
11   console.log(`Tracking Event: ${eventName}`, body);
12   try {
13     await axios.post(COLLECT_API_URL, body, {
14       headers: { 'Authorization': `Bearer ${BEARER_TOKEN}` }
15     });
16   } catch (error) {
17     console.error(`Error tracking event ${eventName}:`, error);
18   }
19 }
20
21

```

Hàm này sẽ thực hiện ghi lại các sự kiện được kích hoạt trực tiếp bởi hành động của người dùng trên giao diện, sau đó gửi cho backend phân tích:

- **start_playing**: Ghi lại thời điểm người dùng bắt đầu nghe một bài hát.

- **skip**: Ghi lại hành động người dùng bỏ qua bài hát hiện tại để nghe bài khác. Đặc biệt, sự kiện này còn gửi cả **play_percentage**, là phần trăm thời lượng mà người dùng đã nghe trước khi bỏ qua.
- **like**: Ghi lại khi người dùng nhấn nút "thích" một bài hát.
- **listen_through**: Ghi lại khi người dùng nghe hết một bài hát (một tín hiệu tích cực mạnh mẽ).



```
1 // 2. useEffect cho Job 10 giây: gửi cảm xúc và cập nhật gợi ý
2 useEffect(() => {
3     const jobInterval = setInterval(() => {
4         setEmotionHistory(currentHistory => {
5             if (currentHistory.length === 0) return [];
6
7             const emotionCounts = currentHistory.reduce((acc, emotion) => {
8                 acc[emotion] = (acc[emotion] || 0) + 1;
9                 return acc;
10            }, {} as Record<string, number>);
11            const dominantEmotion = Object.keys(emotionCounts).reduce((a, b) => emotionCounts
12 [a] > emotionCounts[b] ? a : b);
13            const emotionId = emotionMap[dominantEmotion];
14
15            console.log('Starting 10s Job...');
16
17            if (emotionId) {
18                console.log(`10s Job: Dominant emotion is '${dominantEmotion}
19 ' . Sending to backend...`);
20                const requestBody = { userId: 1, emotion: emotionId, event: "cron_collect"
21                };
22                axios.post(COLLECT_API_URL, requestBody, { headers: { 'Authorization':
23 `Bearer ${BEARER_TOKEN}` } })
24                    .then(() => {
25                        console.log("Emotion data sent. Now refreshing recommendations...");
26                        fetchRecommendedSongs();
27
28                    })
29                    .catch(error => console.error("Error sending emotion data:", error));
30
31            });
32        }, 10000);
33    }
34    return () => clearInterval(jobInterval);
35 }, [fetchRecommendedSongs]); // Thêm fetchRecommendedSongs vào dependency array
```

Ngoài các hành động cụ thể, hệ thống còn tự động thu thập "tâm trạng nền" của người dùng một cách bí động thông qua một cron job chạy định kỳ.

- **Mỗi 10 giây**, một `useEffect` sẽ được kích hoạt.
- Nó phân tích lịch sử cảm xúc trong 10 giây qua (`emotionHistory`) để tìm ra **cảm xúc chủ đạo** (dominant emotion).
- Cảm xúc chủ đạo này được gửi về máy chủ với tên sự kiện là `cron_collect`.

- Sau khi gửi thành công, hệ thống sẽ gọi hàm `fetchRecommendedSongs()` để làm mới danh sách gợi ý, đảm bảo playlist luôn phù hợp với tâm trạng gần nhất của người dùng.

6.4. Xử lý sự kiện

Luồng xử lý này sẽ được thực hiện ở Emotion Collector Service, tại đây frontend sẽ gửi dữ liệu đến thông qua API với request body sau:

```
{
  "userId": 1,
  "musicId": 2,
  "emotion": 3,
  "event": "cron_collect", // listen_through | skip | start_playing | like | cron_collect
  "play_percentage": 5
}
```

Trong đó:

"userId" : 1: Xác định người dùng. Có nghĩa sự kiện này thuộc về người dùng có ID là **1**.

"musicId" : 2: Xác định bài hát. Sự kiện này liên quan đến bài hát có ID là **2**.

"emotion" : 3: Đại diện cho cảm xúc của người dùng tại thời điểm sự kiện xảy ra. Giá trị **3** là một ID số, có thể được ánh xạ tới một cảm xúc cụ thể (ví dụ: 'sad' hoặc 'romantic' tùy thuộc vào `emotionMap` của hệ thống).

"play_percentage": 5: Cho biết phần trăm thời lượng của bài hát đã được phát.

Trong trường hợp này là **5%**.

"event": "cron_collect": Đây là loại sự kiện, và là trường quan trọng nhất để hiểu ngữ cảnh.

Event	Mô tả	Quy tắc xử lý	Ý nghĩa đối với recommend system
start_playing	Kích hoạt khi người dùng bắt đầu nghe một bài hát.	Tính điểm tương tác. Điểm số là một giá trị dương, cố định và nhỏ (score = 0.5).	Tín hiệu quan tâm ban đầu. Cho thấy người dùng có hứng thú hoặc tò mò về bài hát, nhưng chưa phải là một tín hiệu yêu thích mạnh mẽ.
like	Kích hoạt khi người dùng nhấn nút "Thích".	Tính điểm tương tác. Điểm số là một giá trị dương, cố định và cao nhất (score = 5.0).	Tín hiệu tích cực mạnh nhất. Đây là dữ liệu vàng, khẳng định rằng bài hát này rất phù hợp với sở thích và/hoặc cảm xúc của người dùng tại thời điểm đó.
skip	Kích hoạt khi người dùng chuyển bài hát trước khi nó kết thúc.	Tính điểm tương tác. Điểm số được tính toán động dựa trên play_percentage qua hàm calculatePlayScore. Điểm có thể âm (nếu skip sớm) hoặc dương (nếu skip muộn).	Tín hiệu tiêu cực hoặc trung tính. Nếu play_percentage thấp, đây là một tín hiệu tiêu cực rõ ràng. Nếu play_percentage cao (qua 55%), mức độ tiêu cực giảm dần, có thể chỉ là người dùng muốn đổi không khí.

listen_through	Kích hoạt khi bài hát tự chạy đến hết.	Tính điểm tương tác. Điểm số là một giá trị dương, cố định và tương đối cao (score = 2.0).	Tín hiệu tích cực mạnh. Cho thấy người dùng hài lòng và chấp nhận nghe hết bài hát. Mức độ yêu thích cao hơn start_playing nhưng không mạnh bằng một cú like chủ động.
cron_collect	Kích hoạt tự động sau mỗi 10 giây từ frontend để gửi cảm xúc chủ đạo.	Không tính điểm tương tác. Hành động duy nhất là cập nhật cảm xúc gần nhất của người dùng vào Redis với thời gian hết hạn là 24 giờ.	Cập nhật "tâm trạng nền" theo thời gian thực. Dữ liệu này giúp hệ thống gợi ý có thể làm mới playlist một cách linh hoạt để phù hợp với cảm xúc hiện tại của người dùng, ngay cả khi họ không thực hiện hành động nào.

6.4.1. Xử lý sự kiện cron_collect

Đối với sự kiện thu thập bị động (**cron_collect**), mục đích chính không phải là tính điểm tương tác mà là cập nhật "trạng thái cảm xúc hiện tại" của người dùng.

- Khi nhận được sự kiện này, hệ thống sẽ ngay lập tức cập nhật cảm xúc của người dùng vào **Redis**.
- Dữ liệu này được lưu với thời gian hết hạn là 24 giờ, giúp các dịch vụ khác (như hệ thống gợi ý) có thể truy cập nhanh chóng vào cảm xúc gần nhất của người dùng.
- Sau khi cập nhật Redis, hàm sẽ kết thúc ngay lập tức và **không thực hiện tính điểm hay cập nhật vào cơ sở dữ liệu chính**.

6.4.2. Xử lý sự kiện like

Sự kiện này sẽ được gán điểm số cao nhất (**5.0 điểm**), thể hiện sự yêu thích mạnh mẽ.

6.4.3. Xử lý sự kiện listen_through

Đây là sự kiện khi người dùng nghe hết bài hát, được gán **2.0 điểm**. Đánh dấu đây là một tín hiệu tích cực.

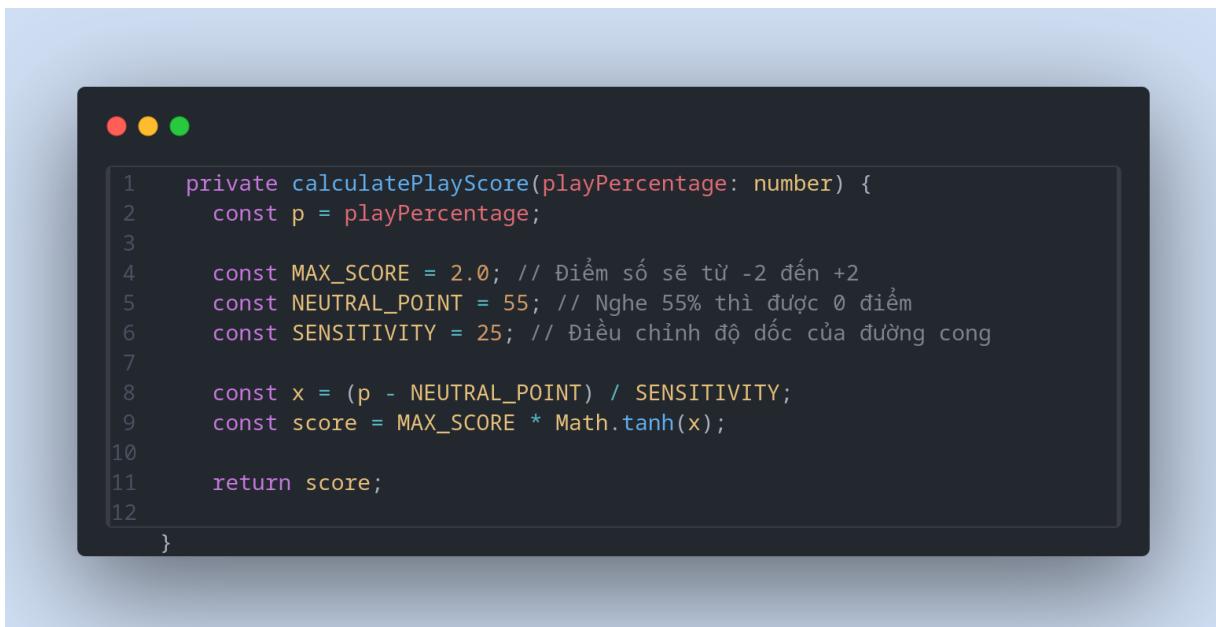
6.4.4. Xử lý sự kiện start_playing

Đây là sự kiện khi bắt đầu nghe một bài hát, được gán **0.5 điểm**. Đây là một tín hiệu cho thấy sự quan tâm ban đầu.

6.4.5. Xử lý sự kiện skip

Sự kiện **skip** (bỏ qua bài hát) được xử lý để phản ánh mức độ không hài lòng. Điểm số được tính dựa trên phần trăm thời lượng đã nghe (**play_percentage**) thông qua một hàm tanh:

$$\text{score} = \text{MAX_SCORE} \cdot \tanh\left(\frac{p - \text{NEUTRAL_POINT}}{\text{SENSITIVITY}}\right)$$



```
1 private calculatePlayScore(playPercentage: number) {
2     const p = playPercentage;
3
4     const MAX_SCORE = 2.0; // Điểm số sẽ từ -2 đến +2
5     const NEUTRAL_POINT = 55; // Nghe 55% thì được 0 điểm
6     const SENSITIVITY = 25; // Điều chỉnh độ dốc của đường cong
7
8     const x = (p - NEUTRAL_POINT) / SENSITIVITY;
9     const score = MAX_SCORE * Math.tanh(x);
10
11    return score;
12}
```

Sau khi điểm số được tính toán, hệ thống sẽ cập nhật nó vào cơ sở dữ liệu chính.

- Hệ thống kiểm tra xem đã có bản ghi nào cho sự kết hợp của (`userId`, `musicId`, `emotion`) hay chưa.
- **Nếu đã tồn tại:** Điểm số mới sẽ được cộng dồn vào điểm số cũ. Điều này cho phép các tương tác được tích lũy (ví dụ: `start_playing` (+0.5) rồi `like` (+5.0) sẽ cho tổng điểm là 5.5).
- **Nếu chưa tồn tại:** Một bản ghi mới sẽ được tạo với điểm số vừa tính được.

6.5. Tạo Recommendation

6.5.1. Retrieve dữ liệu



```
1 musicData = MusicRecommendation()
2
3 # Tải toàn bộ dữ liệu. `all_datasets` là một dict {emotion_id: Dataset}
4 # Thông tin chi tiết bài hát cũng được tải và lưu trong đối tượng musicData
5 print("Loading all datasets from databases...")
6 all_datasets = musicData.loadMusicData()
7 all_users = musicData.loadListeners()
```

Trước khi đi vào tính toán, ta sẽ tải toàn bộ dữ liệu cần thiết vào bộ nhớ.

- Một đối tượng `MusicRecommendation` được khởi tạo để quản lý việc truy cập dữ liệu.
- Toàn bộ dữ liệu tương tác của người dùng (`ratings`) được tải và phân tách thành các bộ dữ liệu riêng biệt cho từng cảm xúc. Kết quả là một dictionary

`all_datasets` nơi mỗi `key` là một `emotion_id` và `value` là bộ dữ liệu tương tác tương ứng.

- Danh sách tất cả người dùng cũng được tải để đảm bảo không bị bỏ sót.

6.5.2. Xử lý theo từng cảm xúc



```
● ● ●
1
2 for emotion_id, emotion_name in emotion_map.items():
3     try: # Bắt đầu khối xử lý lỗi
4         print(f"\n{'='*20} PROCESSING EMOTION: {emotion_name.upper()} (ID: {emotion_id}) {'='*20}")
5
6         # Lấy dữ liệu dành riêng cho cảm xúc này
7         evaluation_data_for_emotion = all_datasets.get(emotion_id)
8
9         # Nếu không có dữ liệu cho cảm xúc này, bỏ qua
10        if not evaluation_data_for_emotion:
11            print(f"No rating data found for emotion '{emotion_name}'. Skipping.")
12            continue
13
```

Chiến lược cốt lõi của hệ thống là không xây dựng một mô hình chung chung, mà thay vào đó, **xây dựng một mô hình gọi ý riêng biệt, chuyên biệt cho từng loại cảm xúc**.

- Hệ thống lặp qua một danh sách các cảm xúc đã định nghĩa (`emotion_map`).
- Trong mỗi vòng lặp, nó chỉ làm việc với bộ dữ liệu tương ứng với cảm xúc đó (ví dụ: chỉ các tương tác xảy ra khi người dùng đang `happy`, hoặc `sad`).
- Điều này giúp các gợi ý trở nên cực kỳ phù hợp với ngữ cảnh cảm xúc của người dùng.

6.5.3. Áp dụng Hybrid Algorithm



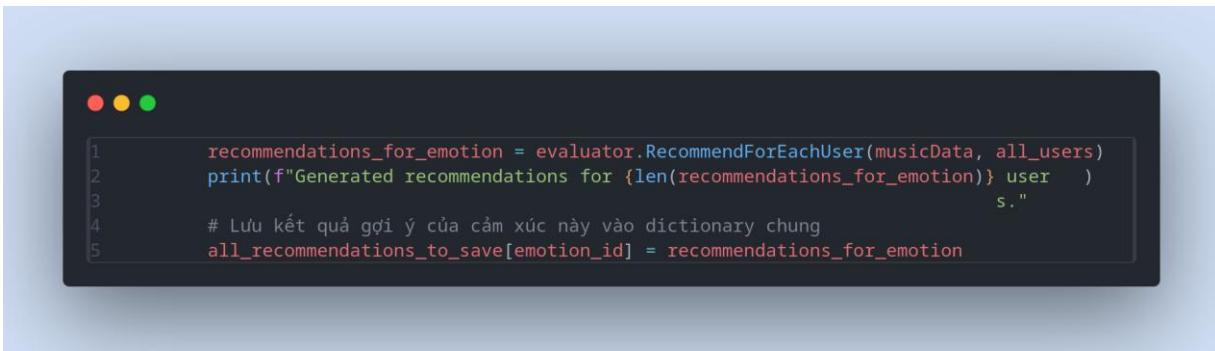
```
1 SimpleRBM = RBMAlgorithm(epochs=40)
2 ContentKNN = ContentKNNAlgorithm(10, {}, musicData)
3 Hybrid = HybridAlgorithm([SimpleRBM, ContentKNN], [0.2, 0.8])
4
5 # Thêm thuật toán vào Evaluator
6 evaluator.AddAlgorithm(Hybrid, "Hybrid")
```

Tại mỗi vòng lặp cảm xúc, hệ thống không chỉ sử dụng một thuật toán duy nhất mà áp dụng **Hybrid**, kết hợp sức mạnh của hai thuật toán khác nhau:

1. **RBMAlgorithm (Restricted Boltzmann Machine)**: Một thuật toán Lọc Cộng tác (Collaborative Filtering) dựa trên Deep Learning, có khả năng học các mẫu tương tác phức tạp giữa người dùng và bài hát. Nó trả lời câu hỏi: "Những người dùng có hành vi giống bạn đã thích bài hát nào?".
2. **ContentKNNAlgorithm (Content-Based K-Nearest Neighbors)**: Một thuật toán Lọc dựa trên Nội dung (Content-Based Filtering). Nó đề xuất các bài hát có đặc điểm âm thanh (như tempo, energy, acousticness...) tương tự như những bài hát mà người dùng đã thích trước đây. Nó trả lời câu hỏi: "Bài hát nào có nội dung giống với những bài bạn đã thích?".

Hai thuật toán này được kết hợp với trọng số **20% cho RBM** và **80% cho ContentKNN**. Điều này có nghĩa là hệ thống ưu tiên gợi ý các bài hát có nội dung tương tự, đồng thời bổ sung thêm các gợi ý đa dạng từ hành vi của cộng đồng.

6.5.4. Tạo và lưu trữ recommendations vào Redis



```
recommendations_for_emotion = evaluator.RecommendForEachUser(musicData, all_users)
print(f"Generated recommendations for {len(recommendations_for_emotion)} user{s}.")
# Lưu kết quả gợi ý của cảm xúc này vào dictionary chung
all_recommendations_to_save[emotion_id] = recommendations_for_emotion
```

Sau khi được huấn luyện trên dữ liệu của một cảm xúc, đối tượng **Evaluator** sẽ tạo danh sách gợi ý cho **tất cả người dùng** trong ngữ cảnh cảm xúc đó.

Kết quả gợi ý của tất cả các cảm xúc được tập hợp lại vào một dictionary lớn.

Cuối cùng, toàn bộ dictionary này được lưu vào **Redis** trong một lần ghi duy nhất. Key trong Redis thường có dạng **recommendations:<emotion_id>**, và value là danh sách gợi ý cho tất cả người dùng.

6.6. Truy xuất Recommendation

Luồng xử lý này sẽ được thực hiện ở Main Service, tại đây, ta sẽ lấy các recommendations sau đó trả ra cho frontend hiển thị tới người dùng.

Khi một người dùng yêu cầu danh sách gợi ý, hàm này sẽ thực hiện một chuỗi các bước theo thứ tự ưu tiên để tổng hợp ra kết quả cuối cùng:

Lớp 1: Gợi ý theo Cảm xúc (Highest Priority)

Đây là lớp logic được ưu tiên cao nhất và là bước đầu tiên trong quy trình. Hệ thống ngay lập tức cố gắng cung cấp các bài hát phù hợp với cảm xúc hiện tại của người dùng.

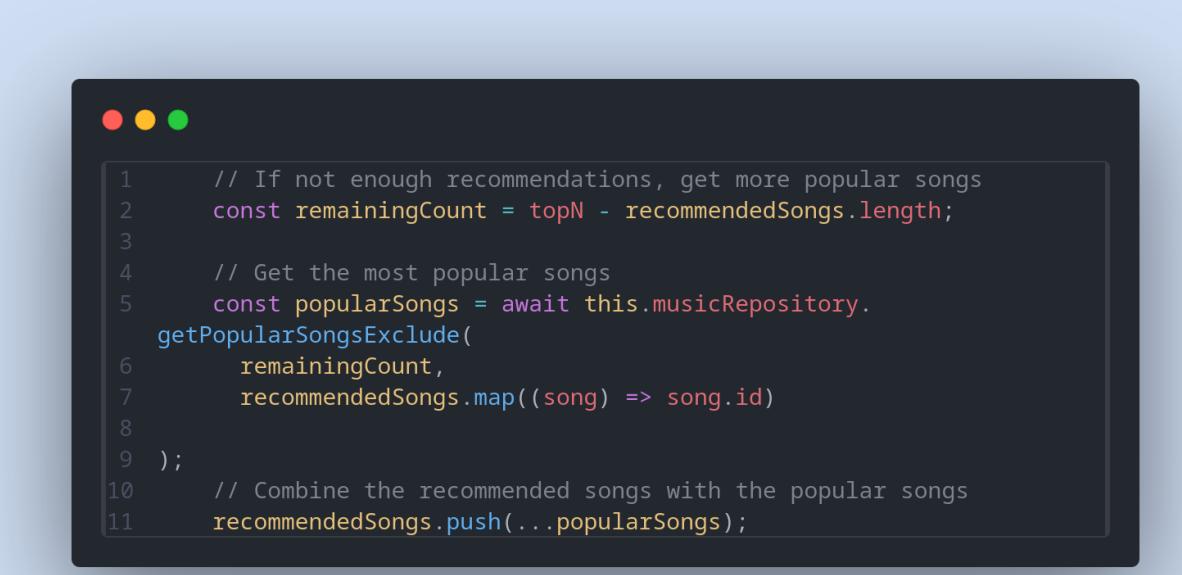
1. **Lấy Cảm xúc Hiện tại:** Hệ thống truy vấn Redis để lấy `listenerEmotion` – cảm xúc gần nhất của người dùng đã được lưu lại (thông qua sự kiện `cron_collect` từ frontend).
2. **Gọi hàm `getSongsByEmotion`:** Nếu tìm thấy cảm xúc, hàm chuyên biệt này sẽ được gọi để lấy danh sách bài hát. Hàm này lại có 2 bước logic nhỏ bên trong:
 - **a. Gọi ý Cá nhân hóa theo Cảm xúc:** Trước tiên, nó cố gắng truy vấn Redis với một `key` rất chi tiết (ví dụ: `emobeat_recommendations:listener:1:emotion:2`) để tìm xem có danh sách gợi ý nào đã được tạo sẵn cho chính người dùng này khi họ ở trong trạng thái cảm xúc đó không. Nếu có, đây là kết quả tốt nhất và sẽ được ưu tiên.
 - **b. Gọi ý Chung theo Thể loại Cảm xúc:** Nếu không có gợi ý cá nhân hóa theo cảm xúc, hệ thống sẽ sử dụng một logic ánh xạ (`switch` case) để tìm các **thể loại nhạc** phù hợp với cảm xúc của người dùng (ví dụ: người dùng `sad` -> gợi ý nhạc `SAD` và `CHILL`). Sau đó, nó sẽ truy vấn cơ sở dữ liệu để lấy các bài hát thuộc thể loại này. Logic ánh xạ sẽ thực hiện theo quy tắc dưới đây:

Listener Emotion	Music Emotion	Chiến lược
Disgusted (Chán ghét)	CHILL, ROMANTIC, OTHER	Chuyển hướng cảm xúc tiêu cực bằng các thể loại nhạc thư giãn, lãng mạn hoặc khác để cải thiện tâm trạng.
Fearful (Sợ hãi)	CHILL	Sử dụng nhạc thư giãn (Chill) để làm

		dịu đi cảm xúc lo lắng, sợ hãi của người dùng.
Angry (Tức giận)	ENERGY	Dùng nhạc sôi động (Energy) để giải tỏa hoặc chuyển hóa năng lượng của sự tức giận, giúp nâng cao tâm trạng.
Sad (Buồn)	SAD, CHILL	Cung cấp hai lựa chọn: nhạc buồn (Sad) để đồng điệu với cảm xúc, hoặc nhạc thư giãn (Chill) để xoa dịu nỗi buồn.
Happy (Vui vẻ)	HAPPY, ENERGY, ROMANTIC	Cung cấp và khuếch đại cảm xúc tích cực bằng các thể loại nhạc vui tươi (Happy), sôi động (Energy) hoặc lãng mạn (Romantic).
Neutral (Trung tính)	OTHER, CHILL, ROMANTIC	Gợi ý các thể loại nhạc dễ nghe, không quá thư giãn hoặc lãng mạn để duy trì tâm trạng hoặc khuyễn khích khám phá.
Surprised (Ngạc nhiên)	ENERGY, HAPPY	Phản ứng với trạng thái bất ngờ (thường có chỉ số energy cao) bằng các thể loại nhạc sôi động hoặc vui vẻ.

3. **Tổng hợp kết quả:** Các bài hát tìm được từ lớp này sẽ là nền tảng cho danh sách gợi ý cuối cùng.

Lớp 2: Gọi ý Phổ biến (Final Fallback)



```
1 // If not enough recommendations, get more popular songs
2 const remainingCount = topN - recommendedSongs.length;
3
4 // Get the most popular songs
5 const popularSongs = await this.musicRepository.
6     getPopularSongsExclude(
7         remainingCount,
8         recommendedSongs.map((song) => song.id)
9     );
10 // Combine the recommended songs with the popular songs
11 recommendedSongs.push(...popularSongs);
```

Đây là lớp dự phòng cuối cùng để đảm bảo hệ thống luôn trả về đủ số lượng bài hát theo yêu cầu (**topN**).

1. **Tính toán số lượng còn thiếu:** Sau khi lớp gọi ý theo cảm xúc hoạt động, hệ thống sẽ kiểm tra xem danh sách **recommendedSongs** đã đủ số lượng chưa.
2. **Truy vấn Bài hát Phổ biến:** Nếu còn thiếu, nó sẽ gọi hàm **getPopularSongsExclude** để lấy thêm các bài hát có lượt nghe (**listenCount**) và lượt yêu thích (**favoriteCount**) cao nhất từ cơ sở dữ liệu.
3. **Loại trừ trùng lặp:** Lời gọi này sẽ truyền vào danh sách ID của các bài hát đã có để đảm bảo các bài hát phổ biến được thêm vào không bị trùng lặp.

4. **Hoàn thiện danh sách:** Các bài hát phổ biến sẽ được thêm vào cuối danh sách để lấp đầy cho đủ số lượng **topN**.

CHƯƠNG 7. KẾT LUẬN

7.1. Kết quả đồ án

Đồ án được phát triển theo đúng quy trình vòng đời phát triển phần mềm, từ khâu phân tích, thiết kế đến triển khai và kiểm thử. Các kiến thức đã được học trong quá trình học tập tại trường như lập trình frontend, backend, thiết kế cơ sở dữ liệu, xử lý ảnh và trí tuệ nhân tạo đã được áp dụng hiệu quả để xây dựng nền hệ thống hoàn chỉnh.

Bên cạnh việc cung cấp kiến thức nền tảng, nhóm còn chủ động tìm hiểu và ứng dụng các công nghệ mới đang được sử dụng rộng rãi trên thế giới, nhằm nâng cao chất lượng và tính thực tiễn của sản phẩm. Các công nghệ nổi bật có thể kể đến như: **ReactJS** kết hợp **TypeScript** với **Vite** và **Tailwind CSS** để xây dựng giao diện hiện đại, mượt mà; **NodeJS** phục vụ phát triển các API nghiệp vụ chính; **FastAPI** xử lý logic gọi ý âm nhạc sử dụng trí tuệ nhân tạo; **PostgreSQL** đảm nhận vai trò lưu trữ dữ liệu chính; và **Redis** dùng để tối ưu truy xuất dữ liệu và tăng hiệu năng toàn hệ thống.

Hệ thống ứng dụng công nghệ trí tuệ nhân tạo thông qua thư viện **face-api.js** để thực hiện nhận diện cảm xúc của người dùng qua webcam theo thời gian thực. Việc tích hợp công nghệ này đã mở ra một trải nghiệm nghe nhạc hoàn toàn mới, nơi người dùng có thể nhận được những gợi ý âm nhạc phù hợp với cảm xúc hiện tại của mình, góp phần nâng cao sự kết nối cảm xúc và cá nhân hóa trong quá trình thưởng thức âm nhạc.

Giao diện ứng dụng được thiết kế trực quan, thân thiện với người dùng, đảm bảo dễ sử dụng cho cả những người không rành về công nghệ. Mọi thao tác từ khởi chạy camera, nhận diện cảm xúc, đến phát nhạc đều được thực hiện nhanh chóng và liền mạch.

Thông qua quá trình thực hiện đồ án, nhóm đã không chỉ hoàn thành một hệ thống đầy đủ chức năng cơ bản mà còn nâng cao được khả năng làm việc nhóm, phân chia công việc hợp lý, giải quyết vấn đề thực tế và tư duy xây dựng hệ thống toàn diện từ frontend, backend đến AI. Đây là một bước đệm quan trọng cho việc ứng dụng kiến thức vào môi trường làm việc chuyên nghiệp trong tương lai.

7.2. Ưu điểm và hạn chế

7.2.1. Ưu điểm

- Ứng dụng tích hợp công nghệ trí tuệ nhân tạo để nhận diện cảm xúc theo thời gian thực từ hình ảnh webcam, tạo ra trải nghiệm nghe nhạc mang tính cá nhân hóa cao.
- Giao diện được xây dựng bằng React kết hợp Tailwind CSS, mang lại trải nghiệm trực quan, hiện đại và thân thiện với người dùng.
- Hệ thống phân tách rõ ràng giữa các thành phần frontend, backend và AI giúp dễ dàng bảo trì, mở rộng và triển khai.
- Cơ sở dữ liệu sử dụng PostgreSQL đảm bảo tính ổn định, an toàn và hiệu quả trong lưu trữ và truy vấn dữ liệu. Redis được tích hợp nhằm tối ưu tốc độ phản hồi.
- Ứng dụng đạt hiệu năng tốt, phản hồi nhanh, xử lý cảm xúc mượt mà và đưa ra gợi ý phù hợp trong thời gian ngắn.

7.2.2. Hạn chế

- Tính chính xác của việc nhận diện cảm xúc còn phụ thuộc nhiều vào điều kiện ánh sáng, góc quay và chất lượng webcam, dẫn đến kết quả chưa thật sự ổn định trong một số tình huống.

- Ứng dụng hiện chưa có tính năng học máy tự động cải thiện chất lượng gợi ý theo thói quen người dùng, việc gợi ý chủ yếu dựa trên quy tắc cảm xúc – thẻ loại cố định.
- Hệ thống chưa hỗ trợ đa nền tảng (chỉ mới triển khai trên trình duyệt), chưa tối ưu cho thiết bị di động.

7.3. Hướng phát triển

- **Tăng cường độ chính xác trong nhận diện cảm xúc** bằng cách kết hợp nhiều mô hình học sâu khác nhau, huấn luyện trên tập dữ liệu lớn hơn và đa dạng hơn về sắc tộc, độ tuổi, ánh sáng và biểu cảm.
- **Phát triển phiên bản di động** (mobile app) trên nền tảng Android và iOS để tiếp cận người dùng rộng rãi hơn và hỗ trợ tính linh hoạt trong việc sử dụng.
- **Bổ sung các tính năng nâng cao**, như lưu lại cảm xúc theo thời gian, thống kê tâm trạng người dùng trong ngày/tuần/tháng, hoặc đề xuất bài hát để cải thiện tâm trạng theo hướng tích cực.

TÀI LIỆU THAM KHẢO

[1] Tìm hiểu thư viện face-api thông qua viết ứng dụng Face Recognition

<https://viblo.asia/p/tim-hieu-thu-vien-face-api-thong-quy-viet-ung-dung-face-recognition-4P856A1BIY3>

[2] Restricted Boltzmann machine

<https://viblo.asia/p/restricted-boltzmann-machine-an-overview-aWj531oQZ6m>

[3] KNN là gì

<https://viblo.asia/p/knn-k-nearest-neighbors-1-djeZ14ejKWz>

[4] DEAM Dataset

<https://www.kaggle.com/datasets/imsparsh/deam-mediaeval-dataset-emotional-analysis-in-music>

[5] Hybrid Recommender Systems Explained

<https://itresearches.com/blending-insights-hybrid-recommender-systems-explained/>

[6] Using the Arousal-Valence Model to Better Your Emotional Intelligence

<https://neurodivergentinsights.com/arousal-valence-model/>

[7] face-api.js documentation

<https://www.npmjs.com/package/face-api.js/v/0.22.2>

[8] A Survey on Multimodal Music Emotion Recognition

<https://arxiv.org/html/2504.18799v1>

[9] Hands-On Guide To Librosa For Handling Audio Files

<https://www.analyticsvidhya.com/blog/2024/01/hands-on-guide-to-librosa-for-handling-audio-files/>

[10] Music Emotion Recognition Papers

<https://paperswithcode.com/task/music-emotion-recognition>

[11] Music Emotion Recognition Algorithm using Deep Learning.

<https://github.com/rxng8/Music-Emotion-Recognition-Algorithm>

[12] Surprise documentation

<https://surpriselib.com/>