BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC THĂNG LONG**

A red and blue logo

Description automatically generated

**BÁO CÁO BÀI TẬP LỚN**

**GỢI Ý ÂM NHẠC THEO CẢM XÚC**

**SINH VIÊN THỰC HIỆN: NHÓM 3**

**MÔN: TRÍ TUỆ NHÂN TẠO**

**GIẢNG VIÊN: NGÔ MẠNH CƯỜNG**

**MÃ LỚP: 251IS22401**

**HÀ NỘI – 2025**

**PHÂN CHIA NHIỆM VỤ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Thành viên** | **MSV** | **Vai trò chính** | **Cụ thể** |
| Hà Phạm Mai Linh | A49612 | Database + Content | * - Chuẩn hoá playlist\_nhac.csv: title/description/link, id không trùng. * - Viết guideline mô tả vibe (từ khoá mood/energy/genre/hoàn cảnh) để embedding hiệu quả. * - Mở rộng dataset (1000 bài), kiểm tra link YouTube hợp lệ |
| Dương Thị Thùy Linh | A49651 | UI/UX + Deploy | * - Xây Gradio Blocks: input, quick moods, nút “Next”, “Reset history”. * - Hiển thị top-3 lựa chọn (tuỳ chọn) + phần giải thích ngắn. * - Styling (CSS) + xử lý lỗi (API key thiếu, link lỗi, quota lỗi). * - Đóng gói chạy: Colab demo / local / HuggingFace Spaces |
| Nguyễn Mạnh Thái | A49587 | Embedding + VectorDB | * - Implement embedding function (Gemini) đúng task\_type: document vs query. * - Nạp dữ liệu vào Chroma, đảm bảo metadata (title/link) đi kèm. * - (Tuỳ chọn) chuyển sang PersistentClient để không mất index khi restart. * - Viết script “rebuild index” (xóa + tạo lại collection) & kiểm tra chất lượng truy hồi. |
| Nguyễn Đức Nam | A50484 | Recommendation Logic | * - Hoàn thiện logic lịch sử nghe (không mutate state trực tiếp, reset/next). * - Fallback khi hết bài mới: chiến lược (top-1 / random trong top-k / theo thời gian). * - (Tuỳ chọn) thêm “explain”: vì sao chọn bài này (dựa trên keywords/score). * - Chuẩn hoá output: song\_title, song\_link, video\_id, ai\_response. |

**LỜI CẢM ƠN**

Nhóm chúng tôi xin bày tỏ lòng biết ơn sâu sắc tới thầy Ngô Mạnh Cường, giảng viên hướng dẫn học phần Trí tuệ nhân tạo, Trường Đại học Thăng Long. Trong suốt quá trình thực hiện bài tập lớn, thầy đã tận tình hướng dẫn, định hướng phương pháp tiếp cận đề tài, đồng thời đưa ra những góp ý mang tính chuyên môn cao giúp nhóm hoàn thiện cả về mặt kỹ thuật lẫn tư duy khoa học.

Những trao đổi và nhận xét của thầy không chỉ giúp nhóm hiểu rõ hơn về cách xây dựng một hệ thống ứng dụng gắn với cơ sở lý thuyết, mà còn hỗ trợ nhóm rèn luyện kỹ năng trình bày báo cáo một cách logic, chặt chẽ và phù hợp với chuẩn mực học thuật. Sự hỗ trợ và đồng hành của thầy là yếu tố quan trọng góp phần vào việc hoàn thành đề tài này.

Nhóm xin chân thành cảm ơn thầy và kính chúc thầy nhiều sức khỏe, thành công trong công tác giảng dạy và nghiên cứu khoa học.

Hà Nội, ngày 10 tháng 01 năm 2026

|  |  |  |  |
| --- | --- | --- | --- |
| **SINH VIÊN** | **SINH VIÊN** | **SINH VIÊN** | **SINH VIÊN** |
|  |  |  |  |
|  |  |  |  |
| **Hà Phạm Mai Linh** | **Dương Thị Thùy Linh** | **Nguyễn Mạnh Thái** | **Nguyễn Đức Nam** |

**LỜI CAM ĐOAN**

Chúng tôi xin cam đoan rằng toàn bộ nội dung của báo cáo Gợi ý âm nhạc theo cảm xúc là kết quả của quá trình nghiên cứu, tìm hiểu và thực hiện nghiêm túc của nhóm dưới sự hướng dẫn của giảng viên. Các số liệu, hình ảnh, thuật toán và mã nguồn trình bày trong báo cáo đều được nhóm tự xây dựng hoặc đã được trích dẫn rõ ràng từ các tài liệu tham khảo hợp pháp.

Chúng tôi xin chịu hoàn toàn trách nhiệm trước Nhà trường và giảng viên hướng dẫn về tính trung thực của nội dung báo cáo. Nếu có bất kỳ sai sót hoặc vi phạm nào liên quan đến bản quyền và đạo văn, chúng tôi xin hoàn toàn chịu trách nhiệm theo các quy định hiện hành.

. Hà Nội, ngày 10 tháng 01 năm 2026

|  |  |  |  |
| --- | --- | --- | --- |
| **SINH VIÊN** | **SINH VIÊN** | **SINH VIÊN** | **SINH VIÊN** |
|  |  |  |  |
|  |  |  |  |
| **Hà Phạm Mai Linh** | **Dương Thị Thùy Linh** | **Nguyễn Mạnh Thái** | **Nguyễn Đức Nam** |

**LỜI MỞ ĐẦU**

Âm nhạc từ lâu đã không chỉ là một hình thức giải trí đơn thuần mà còn là liều thuốc tinh thần, là "người bạn đồng hành" trong mọi cung bậc cảm xúc của con người. Có những ngày mưa buồn, chúng ta tìm đến những bản Ballad sâu lắng; có những buổi sáng đầy năng lượng, ta lại cần những nhịp điệu EDM sôi động để bắt đầu ngày mới.

Tuy nhiên, giữa kho tàng hàng triệu bài hát trên các nền tảng số hiện nay, nghịch lý thay, người dùng ngày càng gặp khó khăn trong việc tìm kiếm giai điệu thực sự "chạm" đến cảm xúc hiện tại của mình. Các công cụ tìm kiếm truyền thống chủ yếu hoạt động dựa trên từ khóa (keyword) hoặc tên bài hát cụ thể. Chúng trở nên bất lực trước những yêu cầu mang tính ngữ nghĩa phức tạp như: *"Tìm cho tôi một bài hát để xoa dịu nỗi buồn thất tình"* hay *"Nhạc không lời để tập trung chạy deadline vào đêm khuya"*. Máy móc truyền thống hiểu từ khóa, nhưng chưa hiểu được con người.

Xuất phát từ thực tế đó, cùng với sự bùng nổ mạnh mẽ của Trí tuệ nhân tạo tạo sinh (Generative AI), em quyết định thực hiện đề tài: "Xây dựng ứng dụng Music Mood AI – Hệ thống gợi ý âm nhạc thấu hiểu cảm xúc".

Dự án này là nỗ lực kết hợp giữa nhu cầu giải trí và công nghệ tiên tiến. Bằng cách ứng dụng mô hình ngôn ngữ lớn Google Gemini kết hợp với kỹ thuật tìm kiếm ngữ nghĩa (Semantic Search) qua Vector Database (ChromaDB), hệ thống được xây dựng không chỉ để "tìm nhạc", mà là để "lắng nghe" người dùng. Ứng dụng đóng vai trò như một DJ ảo, có khả năng phân tích những câu nói đời thường, thấu hiểu tâm trạng ẩn sau đó và đưa ra những gợi ý âm nhạc chính xác nhất, kèm theo những lời dẫn chuyện được cá nhân hóa.

Báo cáo này sẽ trình bày chi tiết quy trình xây dựng hệ thống, từ việc thu thập dữ liệu, xử lý kỹ thuật RAG (Retrieval-Augmented Generation), cho đến việc thiết kế giao diện tương tác người dùng, nhằm minh chứng cho tiềm năng to lớn của AI trong việc nâng cao trải nghiệm giải trí số.

MỤC LỤC

**LỜI CẢM ƠN..............................................................................................................1**

**LỜI CAM ĐOAN........................................................................................................2**

**LỜI MỞ ĐẦU..............................................................................................................3**

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI 8

1.1. Lý do chọn đề tài 8

1.1.1. Đặt vấn đề 8

1.1.2. Mục tiêu 8

1.2. Cơ sở lý thuyết 9

1.2.1. Gợi ý nhạc theo nội dung và theo ngữ cảnh 9

1.2.2. Biểu diễn ngữ nghĩa bằng Embedding và tìm kiếm ngữ nghĩa 9

1.2.3. Vector database và truy hồi 9

1.2.4. Đa dạng và giảm lặp 9

1.2.5. Lập luận về độ tin cậy theo hướng truy hồi 11

1.3. Công nghệ sử dụng 12

1.3.1. Google Gemini API và Google GenAI SDK 12

1.3.2. ChromaDB 12

1.3.3. Gradio 12

CHƯƠNG 2. PHÂN TÍCH THIẾT KẾ VÀ GIẢI THÍCH MÃ NGUỒN 13

2.1. Phân tích thiết kế 13

2.1.1. Kiến trúc Hệ thống 13

2.1.2. Thiết kế Luồng dữ liệu 13

2.1.3. Thiết kế Dữ liệu 14

2.1.4. Thiết kế Giải thuật 14

2.2. Giải thích mã nguồn 14

2.2.1. Cài đặt và Import thư viện 14

2.2.2. Cấu hình API (Authentication) 15

2.2.3. Cấu hình Embedding (Vector hóa) 16

2.2.4. Class bọc (Wrapper) cho ChromaDB 17

2.2.5. Tạo dữ liệu giả lập (Seed Data) 17

2.2.6. Khởi tạo Database Vector 18

2.2.7. Nạp dữ liệu (Ingestion) & Feature Engineering 18

2.2.8. Hàm trích xuất ID YouTube 19

2.2.9. Hàm logic đề xuất nhạc 20

2.2.10. Giao diện hoá 22

CHƯƠNG 3. TRÌNH BÀY KẾT QUẢ 24

3.1. Giao diện người dùng hoàn thiện 24

3.2. Kịch bản thử nghiệm (Test Cases) 24

3.2.1. Kịch bản 1: Cảm xúc tiêu cực (Buồn/Cô đơn) 24

3.2.2. Kịch bản 2: Hoạt động năng lượng cao (Tập luyện) 26

3.3. 3. Đánh giá tính năng "Smart Shuffle" (Chống lặp) 28

CHƯƠNG 4. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 31

4.1. Kết luận chung 31

4.2. Hạn chế của đề tài 31

4.3. Hướng phát triển trong tương lai 32

**TÀI LIỆU THAM KHẢO...........................................................................................33**

**DANH MỤC BẢNG, HÌNH ẢNH, ĐỒ THỊ**

[Bảng 2.1 Cài đặt và Import thư viện 15](#_Toc219547057)

[Bảng 2.2 Mã nguồn Cấu hình API 15](#_Toc219547058)

[Bảng 2.3 Mã nguồn cấu hình Embedding 16](#_Toc219547059)

[Bảng 2.4Mã nguồn class bọc cho ChromaDB 17](#_Toc219547060)

[Bảng 2.5Mã nguồn tạo dữ liệu giải lập 17](#_Toc219547061)

[Bảng 2.6 Mã nguồn khởi tạo Database Vector 18](#_Toc219547062)

[Bảng 2.7Mã nguồn nạp dữ liệu vào trong ChromaDB 19](#_Toc219547063)

[Bảng 2.8 Hàm trích xuất ID từ YouTube 19](#_Toc219547064)

[Bảng 2.9 Hàm logic đề xuất nhạc 20](#_Toc219547065)

[Bảng 2.10 Module tìm kiếm ngữ nghĩa trong hàm 20](#_Toc219547066)

[Bảng 2.11 Module chống lặp bài hát 21](#_Toc219547067)

[Bảng 2.12 Module sinh lời dẫn DJ 21](#_Toc219547068)

[Bảng 2.13 Modul tạo sinh tăng cường 22](#_Toc219547069)

[Bảng 2.14 Giao diện hoá 22](#_Toc219547070)

[Bảng 2.15 Sự kiện nút bấm nhanh 23](#_Toc219547071)

[Bảng 2.16 Chạy Gradio 23](#_Toc219547072)

[Ảnh 1.1 Sơ đồ kiến trúc của Mrecuri – Mô hình gợi ý nhạc theo hướng đa dạng hóa 10](#_Toc218781593)

[Ảnh 1.2 Kiến trúc RAG (Retrieval - Augmented Generation) – Truy hồi tăng cường sinh 11](#_Toc218781594)

[Ảnh 3.1 Giao diện hoàn thiện 24](#_Toc218781595)

[Ảnh 3.2 Bài hát: “Breathe (In the Air)”, mô tả: “giai điệu êm ái, thư giãn như một hơi thở nhẹ, hoàn hảo để xoa dịu tâm hồn sau ngày dài mệt mỏi”. 25](#_Toc218781596)

[Ảnh 3.3 Bài hát: “T.N.T”, mô tả: “bài hát đầy năng lượng, nhịp điệu mạnh mẽ, tinh thần bùng nổ, phù hợp nghe để xua tan cảm giác mệt mỏi chán nản.” 25](#_Toc218781597)

[Ảnh 3.4 Bài hát: On My Way, Mô tả: EDM hiện đại, giai điệu truyền cảm hứng mạnh mẽ, phù hợp nghe khi cần động lực làm việc hoặc đi phượt xa. 26](#_Toc218781598)

[Ảnh 3.5 Bài hát “Stronger”, mô tả “Electronic-rap mạnh mẽ, lời ca nghị lực, là bài hát không thể thiếu trong danh sách nhạc tập gym để bứt phá giới hạn.” 27](#_Toc218781599)

[Ảnh 3.6 Bài hát: “Sugar”, mô tả: “Giai điệu cuồng nhiệt, tràn đầy năng lượng thô sơ, phù hợp nghe trong các buổi tập gym hoặc khi lái xe.” 28](#_Toc218781600)

[Ảnh 3.7 Bài hát: “Cold Heart - PNAU Remix”, mô tả: “Disco-pop hiện đại, giai điệu chill và bắt tai, lý tưởng nghe khi đang thư giãn vào cuối ngày hoặc dọn dẹp nhà cửa gọn gàng.” 29](#_Toc218781601)

[Ảnh 3.8 Bài hát: “Calma - Remix”, mô tả: “Nhạc Tropical Pop tươi mát, mang hơi thở biển cả, hoàn hảo cho những chuyến du lịch hè hoặc lúc nghỉ ngơi chill.” 30](#_Toc218781602)

[Ảnh 3.9 Những lần thử tiếp theo 30](#_Toc218781603)

# GIỚI THIỆU ĐỀ TÀI

## Lý do chọn đề tài

### Đặt vấn đề

Sự bùng nổ của nền tảng nghe nhạc trực tuyến tạo ra một nghịch lý phổ biến: người dùng có quyền truy cập kho nội dung rất lớn nhưng lại gặp khó khăn khi diễn đạt nhu cầu nghe nhạc theo cách mà hệ thống có thể hiểu và đáp ứng tức thời. Trong thực tế, nhu cầu thường được biểu đạt dưới dạng cảm xúc hoặc bối cảnh như buồn sau chia tay, cần tập trung học tập, hoặc muốn tăng động lực khi luyện tập. Các hệ gợi ý truyền thống dựa nhiều vào hành vi lịch sử và tương tác cộng đồng thường gặp vấn đề cold start, đồng thời khó xử lý truy vấn ngôn ngữ tự nhiên giàu ngữ cảnh.

Trong nghiên cứu hệ gợi ý nhạc, các khảo sát gần đây nhấn mạnh sự dịch chuyển từ các mô hình thuần collaborative filtering hoặc thuần content based sang các mô hình khai thác đồng thời dữ liệu nội dung và dữ liệu tương tác, đồng thời nêu bật các thách thức như tăng đa dạng và mới lạ, giải thích khuyến nghị, nhận biết ngữ cảnh, khuyến nghị theo chuỗi, mở rộng hiệu năng, và giảm cold start (Khởi đầu lạnh) – xảy ra khi hệ thống không thể đưa ra một gợi ý đáng tin cậy với quá ít dữ liệu.[1]

Trên hiện thực đó, đề tài Music Mood AI được lựa chọn nhằm xây dựng một nguyên mẫu hệ gợi ý dựa trên cảm xúc, trong đó đầu vào là mô tả tâm trạng bằng ngôn ngữ tự nhiên và đầu ra là nội dung nhạc phù hợp. Hướng tiếp cận này phù hợp với xu thế prompt based recommendation trong đó người dùng đưa ra yêu cầu bằng văn bản và hệ thống cần ánh xạ trực tiếp từ prompt sang đối tượng âm nhạc. Một hướng nghiên cứu tiêu biểu là Text2Tracks, đề xuất coi bài toán gợi ý nhạc theo prompt như một bài toán truy hồi sinh, nhằm ánh xạ prompt trực tiếp sang định danh bài hát để nâng cao hiệu quả và giảm các bước trung gian.[2]

### Mục tiêu

Xây dựng ứng dụng Music Mood AI cho phép người dùng nhập trạng thái cảm xúc bằng ngôn ngữ tự nhiên (Tiếng Việt). Hệ thống sẽ:

* Hiểu ngữ nghĩa của câu nói.
* Tìm kiếm bài hát phù hợp nhất trong kho dữ liệu.
* Tránh lặp lại bài hát vừa nghe (Smart Shuffle).
* Đóng vai DJ ảo để viết lời dẫn nhập phù hợp với tâm trạng.

## Cơ sở lý thuyết

### Gợi ý nhạc theo nội dung và theo ngữ cảnh

Hệ gợi ý nhạc có thể được phân loại theo các hướng như collaborative filtering, content based recommendation và context aware recommendation. Trong phạm vi đề tài, trọng tâm nằm ở hướng gợi ý dựa trên nội dung và ngữ cảnh cảm xúc, bởi đầu vào là mô tả tâm trạng và bối cảnh nghe. Các nghiên cứu về yếu tố cảm xúc và ngữ cảnh trong gợi ý nhạc cho thấy việc đưa thông tin cảm xúc và ngữ cảnh vào mô hình có thể cải thiện mức độ phù hợp trải nghiệm nghe so với các hệ chỉ dựa trên lịch sử.[3]

### Biểu diễn ngữ nghĩa bằng Embedding và tìm kiếm ngữ nghĩa

Embedding là phép ánh xạ văn bản sang không gian vector sao cho các biểu thức gần nghĩa có vector gần nhau. Đây là nền tảng của tìm kiếm ngữ nghĩa, cho phép truy hồi theo nghĩa thay vì chỉ theo từ khóa. Tài liệu của Gemini API mô tả embedding như một thành phần nền tảng cho các tác vụ như semantic search, phân loại, gom cụm và đặc biệt phù hợp để xây dựng hệ truy hồi tăng cường sinh. [4]

Trong thực hành, các hệ embedding hiện đại thường hỗ trợ nhãn loại tác vụ nhằm tối ưu vector cho mục đích sử dụng, ví dụ tối ưu cho truy hồi tài liệu hoặc truy hồi truy vấn. [5]

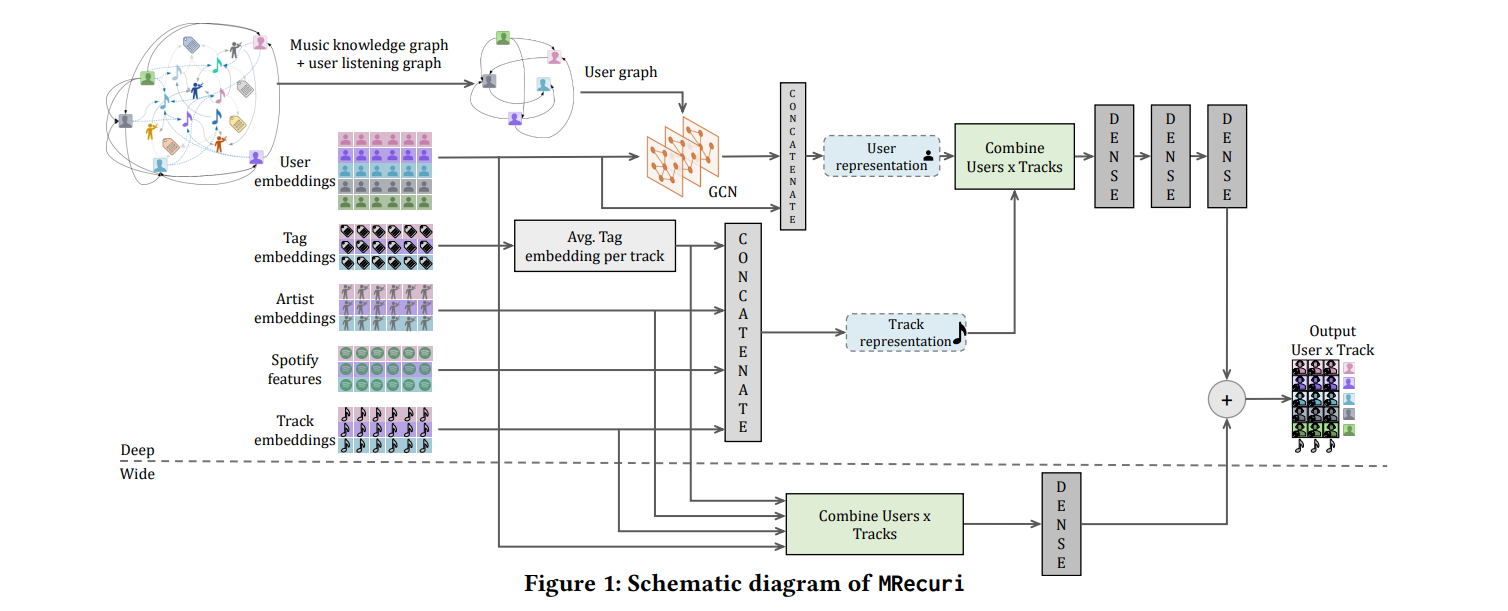
### Vector database và truy hồi

Vector database lưu trữ embedding của các mục nội dung kèm metadata, cho phép truy vấn tương tự bằng khoảng cách vector và trả về danh sách top k kết quả gần nhất. Chroma là một hệ cơ sở dữ liệu truy hồi và tìm kiếm theo vector, trong đó các thao tác cốt lõi gồm tạo collection, thêm dữ liệu và truy vấn tương tự để lấy các kết quả liên quan. [6]

### Đa dạng và giảm lặp

Bên cạnh độ liên quan, các thuộc tính như đa dạng và mới lạ có vai trò quan trọng trong trải nghiệm nghe nhằm hạn chế lặp lại và tránh hình thành bong bóng sở thích. Trong lĩnh vực truy hồi thông tin, Maximal Marginal Relevance là tiêu chuẩn kinh điển để cân bằng giữa liên quan và mới lạ bằng cách giảm trùng lặp trong danh sách kết quả [7]

Các nghiên cứu trong gợi ý nhạc cũng nhấn mạnh tác động của đa dạng đối với hành vi khám phá và thái độ của người nghe trong dài hạn. Như trong bài báo Exploring Diversity in Music Recommendations[8] đã chỉ ra rằng nếu hệ gợi ý chỉ tập trung đưa ra các bài quá giống những gì người dùng đã nghe, người nghe dễ bị quẩn quanh trong một nhóm thể loại quen thuộc và ít có cơ hội khám phá điều mới. Nghiên cứu này đề xuất cách tăng đa dạng và tính mới lạ bằng cách nhận diện vùng sở thích hiện tại của người dùng từ lịch sử nghe và mối liên hệ giữa các bài hát, sau đó chủ động gợi ý thêm những bài vẫn liên quan nhưng khác hơn một chút để mở rộng trải nghiệm nghe về lâu dài.



Ảnh 1.1 Sơ đồ kiến trúc của Mrecuri – Mô hình gợi ý nhạc theo hướng đa dạng hóa

Cách hoạt động của mô hình như sau:

* Thu thập nhiều nguồn đặc trưng: lịch sử nghe, knowledge graph, tag, artist, audio features, track embeddings
* Dùng GCN trên user graph để học user representation giàu ngữ cảnh từ quan hệ đồ thị
* Tạo track representation bằng cách ghép (concatenate) các embedding/feature của track
* Kết hợp user × track rồi đưa qua các tầng Dense để tính điểm phù hợp
* Song song có nhánh Wide (đơn giản/tuyến tính) và nhánh Deep (GCN + Dense), sau đó cộng hai nhánh để ra điểm xếp hạng cuối

Ý nghĩa chính:

* Gợi ý nhạc vừa liên quan vừa tăng đa dạng, giảm hiệu ứng bong bóng lọc
* Tận dụng cấu trúc đồ thị và đặc trưng nội dung để học sở thích người dùng tốt hơn

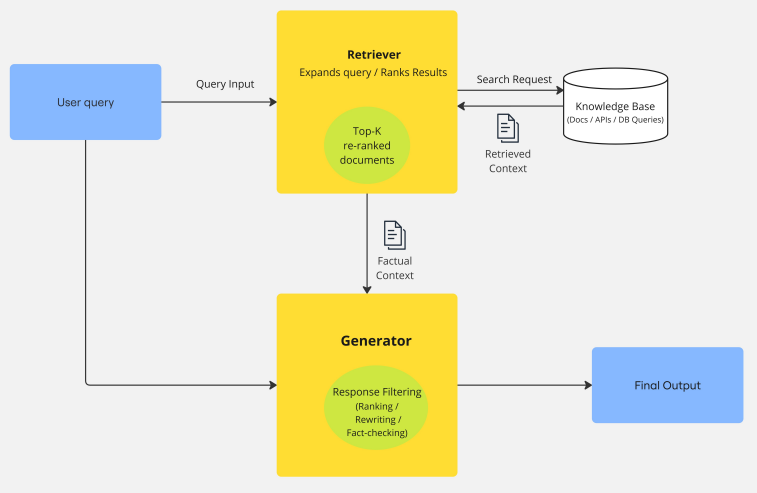
Liên hệ vào mô hình Music Mood AI:

* Nhóm em dùng embedding + truy hồi top-k và lịch sử nghe để giảm lặp (cách nhẹ, dễ triển khai)
* MRecuri là hướng học sâu cho cùng mục tiêu: tối ưu gợi ý user–track và tăng đa dạng dựa trên đồ thị và nhiều đặc trưng hơn

### Lập luận về độ tin cậy theo hướng truy hồi

Trong các hệ kết hợp mô hình sinh và truy hồi, nguyên tắc truy hồi trước rồi mới sinh giúp tăng mức độ bám sát dữ liệu và hạn chế sai lệch do mô hình sinh tạo ra. Các khảo sát và tổng quan về RAG cũng thảo luận rộng rãi về vai trò của truy hồi trong việc cải thiện độ bám chứng cứ và giảm hiện tượng thông tin bịa đặt, đồng thời chỉ ra rằng chất lượng truy hồi và cơ chế grounding là các yếu tố quyết định. [9]

Liên hệ với mô hình của nhóm, nguyên tắc truy hồi trước rồi mới sinh được áp dụng rõ ràng: hệ thống trước hết chuyển tâm trạng người dùng thành embedding và truy hồi top k bài phù hợp từ Chroma, trong đó mỗi bài có metadata cố định gồm tiêu đề và liên kết. Vì vậy, việc chọn bài hát được quyết định bởi bước truy hồi trên kho dữ liệu đã kiểm soát, còn mô hình sinh chỉ tạo câu dẫn ngắn để tăng trải nghiệm. Cách thiết kế này tạo cơ chế grounding trực tiếp vào playlist, giúp giảm rủi ro mô hình sinh bịa ra bài hoặc link không tồn tại, phù hợp với lập luận của RAG rằng truy hồi và grounding là yếu tố then chốt để tăng độ bám chứng cứ và hạn chế sai lệch



Ảnh 1.2 Kiến trúc RAG (Retrieval - Augmented Generation) – Truy hồi tăng cường sinh

**Mô hình hoạt động như sau:**

1. **User query**: Người dùng nhập câu hỏi/yêu cầu.
2. **Retriever**: Hệ thống đi **tìm trong kho dữ liệu** (Knowledge Base: tài liệu/API/DB) các đoạn liên quan nhất (top-k). Có thể mở rộng query, xếp hạng lại kết quả.
3. **Generator**: Mô hình sinh nhận **ngữ cảnh đã truy hồi** rồi mới tạo câu trả lời, đồng thời có thể lọc/kiểm tra lại nội dung.
4. **Final output**: Trả kết quả cuối.

Ý nghĩa chính

1. **Tăng độ tin cậy**: câu trả lời “bám” vào dữ liệu được truy hồi (grounding).
2. **Giảm hallucination**: hạn chế việc mô hình tự bịa thông tin.
3. **Cập nhật dễ**: muốn kiến thức mới chỉ cần cập nhật kho dữ liệu, không cần huấn luyện lại model.

Liên hệ với mô hình Music Mood AI

1. Knowledge Base = playlist CSV (mô tả bài hát).
2. Retriever = embedding + Chroma truy hồi top-k bài phù hợp mood.
3. Generator = Gemini chỉ viết câu dẫn DJ.
4. Final output = bài hát được phát (link YouTube) + câu dẫn.

## Công nghệ sử dụng

### Google Gemini API và Google GenAI SDK

Hệ thống sử dụng Gemini API cho hai chức năng: tạo embedding văn bản và sinh văn bản. Tài liệu Gemini API về embeddings mô tả rõ vai trò của embedding trong semantic search và các kịch bản RAG. [Google AI for Developers](https://ai.google.dev/gemini-api/docs/embeddings?utm_source=chatgpt.com) [10]

Về thư viện triển khai, Google cung cấp hướng dẫn chuyển đổi sang Google GenAI SDK, nhấn mạnh mô hình client thống nhất và khuyến nghị di trú từ các SDK cũ. [10]

Ngoài ra, trang PyPI của gói google generativeai cũng ghi nhận trạng thái deprecated, phản ánh xu hướng hợp nhất SDK. [11]

### ChromaDB

Chroma được sử dụng như vector store để lưu embedding của mô tả bài hát và thực hiện truy vấn top k theo độ tương tự. Tài liệu Chroma mô tả cơ chế truy vấn và truy xuất dữ liệu trong collection. [12]

### Gradio

Thư viện Python, Gradio được chọn để xây dựng giao diện tương tác nhanh cho nguyên mẫu, cho phép quản lý trạng thái theo phiên thông qua State trong Blocks, phù hợp với yêu cầu lưu lịch sử nghe để giảm lặp.[13]

# PHÂN TÍCH THIẾT KẾ VÀ GIẢI THÍCH MÃ NGUỒN

## Phân tích thiết kế

### Kiến trúc Hệ thống

Ứng dụng được thiết kế theo mô hình RAG (Retrieval-Augmented Generation) – (Truy xuất thông tin - Tăng cường ngữ cảnh - Sinh nội dung).

Các thành phần chính:

* Frontend (Giao diện): Sử dụng thư viện Gradio. Đóng vai trò nhận input từ người dùng và hiển thị trình phát nhạc.
* Backend Controller: Mã nguồn Python chạy trên Google Colab. Đóng vai trò điều phối luồng dữ liệu.
* Knowledge Base (Cơ sở tri thức): Sử dụng ChromaDB (Vector Database) để lưu trữ thông tin bài hát dưới dạng vector số học.
* AI Services (Trí tuệ nhân tạo): Sử dụng Google Gemini API cho hai nhiệm vụ:
* *Embedding Model:* Chuyển đổi văn bản sang vector.
* *Generative Model:* Tạo lời dẫn chuyện (Chat/Completion).

### Thiết kế Luồng dữ liệu

Quy trình xử lý dữ liệu đi qua 5 bước tuần tự:

* Input: Người dùng nhập tâm trạng (Ví dụ: "Tôi đang thất tình") hoặc chọn nút gợi ý nhanh.
* Vectorization (Mã hóa):
* Hệ thống gửi câu "Tôi đang thất tình" đến Gemini API.
* Nhận về một vector embedding (dãy số đa chiều).
* Semantic Search (Tìm kiếm ngữ nghĩa):
* Vector câu hỏi được so sánh với hàng ngàn vector bài hát trong ChromaDB.
* Hệ thống sử dụng thuật toán tính khoảng cách (thường là Cosine Similarity) để tìm ra các vector gần nhất (các bài hát có mô tả tương đồng nhất).
* Business Logic (Xử lý nghiệp vụ):
* Áp dụng bộ lọc "Smart Shuffle": Kiểm tra ID bài hát tìm được với lịch sử nghe (history\_state).
* Chọn ra bài hát chưa nghe hoặc bài phù hợp nhất.
* Output Generation (Sinh kết quả):
* Gửi thông tin bài hát + tâm trạng người dùng đến Gemini (Generative Model) để viết lời dẫn DJ.
* Tạo mã HTML nhúng YouTube và trả về giao diện Gradio.

### Thiết kế Dữ liệu

Hệ thống sử dụng Vector Database:

* Cấu trúc dữ liệu đầu vào (CSV):
* id: Định danh duy nhất.
* title: Tên bài hát.
* description: Từ khóa mô tả cảm xúc (Dữ liệu quan trọng nhất để AI hiểu).
* link: Đường dẫn YouTube.

### Thiết kế Giải thuật

Sử dụng thuật toán nhằm tránh lặp lại bài hát:

* Mục Đích:
* Tránh việc AI trả về 1 bài nhiều lần liên tiếp.
* Giải pháp:
* Truy xuất Top 10 kết quả (thay vì Top 1).
* Duyệt vòng lặp qua Top 10.
* So sánh với played\_history (Lịch sử phiên làm việc).
* Ưu tiên chọn bài có độ phù hợp cao nhưng chưa từng xuất hiện trong lịch sử.
* Nếu đã nghe hết Top 10, mới chấp nhận phát lại bài Top 1 và cập nhật lại vị trí của nó trong lịch sử.

## Giải thích mã nguồn

### Cài đặt và Import thư viện

|  |
| --- |
| !pip install -q -U google-generativeai chromadb gradio pandas  import google.generativeai as genai  import chromadb  from chromadb import Documents, EmbeddingFunction, Embeddings  import pandas as pd  import os  import re  import gradio as gr  from google.colab import userdata |

Bảng 2.1 Cài đặt và Import thư viện

* !pip install ...: Lệnh cài đặt các thư viện cần thiết trên môi trường Google Colab.
* google.generativeai: Thư viện chính để gọi API của Google Gemini.
* chromadb: Thư viện cơ sở dữ liệu Vector để lưu trữ bài hát.
* pandas: Dùng để tạo và đọc file CSV chứa danh sách nhạc.
* os: Tương tác với hệ điều hành (kiểm tra file tồn tại).
* re: Regular Expression (Biểu thức chính quy) để xử lý chuỗi (tách ID YouTube).
* gradio: Dùng để vẽ giao diện web.
* userdata: Dùng để lấy API Key được lưu bảo mật trong Google Colab.

### Cấu hình API (Authentication)

|  |
| --- |
| try:  GOOGLE\_API\_KEY = userdata.get('API\_KEY')  except:  GOOGLE\_API\_KEY = input("Dán Google AI Studio Key của bạn vào đây: ")  genai.configure(api\_key=GOOGLE\_API\_KEY) |

Bảng 2.2 Mã nguồn Cấu hình API

* Mục đích: Xác thực quyền truy cập vào dịch vụ AI của Google.
* Logic: Code thử lấy Key từ userdata (bảo mật). Nếu không thấy (lỗi), nó sẽ hiện ô input để người dùng nhập thủ công.
* genai.configure: Thiết lập kết nối với key đã nhận được.

### Cấu hình Embedding (Vector hóa)

|  |
| --- |
| def get\_gemini\_embedding(text, task\_type="retrieval\_document"):  model = "models/text-embedding-004"  try:  response = genai.embed\_content(  model=model,  content=text,  task\_type=task\_type,  title="Music Embedding" if task\_type == "retrieval\_document" else None  )  return response['embedding']  except Exception as e:  print(f"❌ Lỗi Embedding: {e}")  return [] |

Bảng 2.3 Mã nguồn cấu hình Embedding

* Embedding là quá trình biến đổi một đoạn văn bản (ví dụ: "Tôi buồn") thành một dãy số (vector) gồm 768 con số. Máy tính sẽ so sánh các dãy số này để biết câu nào có ý nghĩa gần giống nhau.
* def get\_gemini\_embedding(...): Khai báo một hàm nhận vào văn bản (text) và trả về vector.
* model = "models/text-embedding-004": Chỉ định chính xác phiên bản model. Đây là model chuyên dụng để chuyển đổi văn bản thành vector số học (768 chiều), khác với model Gemini Pro dùng để chat.
* task\_type: Tham số này báo cho Google biết mục đích sử dụng vector để tối ưu hóa thuật toán:
* retrieval\_document: Dùng khi ta đang vector hóa cơ sở dữ liệu (các bài hát).
* retrieval\_query: Dùng khi ta đang vector hóa câu hỏi của người dùng.
* genai.embed\_content(...): Lệnh gửi request lên server Google.
* return response['embedding']: Trích xuất phần dữ liệu vector (mảng số thực [0.1, -0.5, ...]) từ kết quả JSON trả về.
* except Exception as e: Nếu mất mạng hoặc Key sai, hàm sẽ in lỗi và trả về mảng rỗng [] để chương trình không bị dừng đột ngột.

### Class bọc (Wrapper) cho ChromaDB

|  |
| --- |
| class GeminiEmbeddingFunction(EmbeddingFunction):  def \_\_call\_\_(self, input: Documents) -> Embeddings:  embeddings = []  for text in input:  emb = get\_gemini\_embedding(text, task\_type="retrieval\_document")  embeddings.append(emb)  return embeddings |

Bảng 2.4Mã nguồn class bọc cho ChromaDB

* class GeminiEmbeddingFunction(EmbeddingFunction): Khai báo một lớp kế thừa từ lớp cha EmbeddingFunction của thư viện ChromaDB.
* def \_\_call\_\_(self, input: Documents): Phương thức đặc biệt (Magic Method). Khi ta đưa class này vào ChromaDB, ChromaDB sẽ gọi nó như một hàm số.
* Vòng lặp for text in input: ChromaDB thường xử lý theo lô (batch). input là một danh sách các văn bản. Vòng lặp này duyệt qua từng văn bản, gọi hàm API ở trên để lấy vector, và gom lại thành danh sách embeddings để trả về cho Database lưu trữ.

### Tạo dữ liệu giả lập (Seed Data)

|  |
| --- |
| csv\_filename = 'playlist\_nhac.csv'  if not os.path.exists(csv\_filename):  data = { ... } # Dictionary chứa dữ liệu  df\_sample = pd.DataFrame(data)  df\_sample.to\_csv(csv\_filename, index=False, encoding='utf-8') |

Bảng 2.5Mã nguồn tạo dữ liệu giải lập

* if not os.path.exists(...): Kiểm tra xem file CSV đã có chưa. Điều này giúp tránh việc ghi đè lại file nếu người dùng đã chỉnh sửa dữ liệu thủ công trước đó.
* pd.DataFrame(data): Chuyển đổi dữ liệu từ dạng Dictionary (Python thuần) sang dạng bảng (Pandas DataFrame) để dễ xử lý.
* to\_csv(..., index=False): Lưu ra file Excel/CSV. index=False nghĩa là không lưu cột số thứ tự dòng (0, 1, 2...) vào file, giúp file gọn hơn. encoding='utf-8' để đảm bảo hiển thị đúng tiếng Việt.

### Khởi tạo Database Vector

|  |
| --- |
| try:  chroma\_client = chromadb.Client()  try: chroma\_client.delete\_collection("music\_mood\_db\_v3")  except: pass  collection = chroma\_client.create\_collection(name="music\_mood\_db\_v3", embedding\_function=GeminiEmbeddingFunction()) |

Bảng 2.6 Mã nguồn khởi tạo Database Vector

* chromadb.Client(): Tạo một máy chủ cơ sở dữ liệu chạy ngay trên RAM (In-memory). Ưu điểm là nhanh, nhược điểm là tắt code sẽ mất dữ liệu.
* delete\_collection(...): Dòng lệnh dọn dẹp. Trước khi chạy, ta xóa collection cũ đi để đảm bảo dữ liệu nạp vào là mới tinh, không bị trùng lặp (duplicate) do chạy code nhiều lần.
* create\_collection(...): Tạo bảng mới. Quan trọng nhất là tham số embedding\_function=GeminiEmbeddingFunction(). Cấy bộ não AI (Class đã viết ở Phần 2) vào Database. Kể từ giờ, cứ nạp chữ vào là Database tự động dùng bộ não này để chuyển thành số.

### Nạp dữ liệu (Ingestion) & Feature Engineering

|  |
| --- |
| df = pd.read\_csv(csv\_filename)  df['id'] = df['id'].astype(str)  df['ai\_content'] = "Bài hát: " + df['title'] + ". Vibe: " + df['description']  collection.add(  documents=df['ai\_content'].tolist(),  ids=df['id'].tolist(),  metadatas=df[['title', 'link']].to\_dict('records')  ) |

Bảng 2.7Mã nguồn nạp dữ liệu vào trong ChromaDB

* astype(str): ChromaDB yêu cầu ID phải là chuỗi ký tự (String), nên ta ép kiểu cột ID từ số sang chữ.
* df['ai\_content'] = ...: Đây là kỹ thuật Feature Engineering (Kỹ thuật đặc trưng).
* *Vấn đề:* Nếu chỉ vector hóa mỗi phần mô tả, AI có thể bỏ sót ngữ nghĩa trong tên bài hát.
* *Giải pháp:* Ta ghép chuỗi Tên bài + Mô tả thành một đoạn văn bản thống nhất (ai\_content). Ví dụ: *"Bài hát: Mưa Trong Lòng. Vibe: Ballad buồn, thất tình"*. Điều này cung cấp ngữ cảnh đầy đủ nhất cho AI hiểu.
* collection.add(...): Lệnh nạp dữ liệu vào kho.
* documents: Dữ liệu dùng để tìm kiếm (đã gộp ở trên).
* metadatas: Dữ liệu đi kèm (Tên, Link) để hiển thị lại cho người dùng sau khi tìm thấy.

### Hàm trích xuất ID YouTube

|  |
| --- |
| def extract\_youtube\_id(url):  if not isinstance(url, str): return None  regex = r"(?:v=|\/)([0-9A-Za-z\_-]{11})"  match = re.search(regex, url)  return match.group(1) if match else None |

Bảng 2.8 Hàm trích xuất ID từ YouTube

* Mục đích: Link YouTube có nhiều dạng (youtube.com/watch?v=ABC, youtu.be/ABC). Để nhúng vào web, ta chỉ cần đoạn mã ABC (11 ký tự).
* regex = ...: Biểu thức chính quy.
* (?:v=|\/): Tìm chuỗi bắt đầu bằng v= HOẶC dấu gạch chéo /. ?: nghĩa là không cần lấy phần này, chỉ dùng để định vị.
* ([0-9A-Za-z\_-]{11}): Nhóm chính cần lấy. Tìm chuỗi gồm 11 ký tự bao gồm chữ, số, gạch dưới, gạch ngang.
* match.group(1): Lấy kết quả trong ngoặc đơn đầu tiên.

### Hàm logic đề xuất nhạc

|  |
| --- |
| def recommend\_music\_logic(user\_mood, played\_history): |

Bảng 2.9 Hàm logic đề xuất nhạc

* Input**:**
* user\_mood: Câu văn người dùng nhập (VD: "Tôi buồn").
* played\_history: Danh sách các ID bài hát đã nghe trong phiên làm việc này (VD: ['1', '5']).

Bước 1: Tìm kiếm ngữ nghĩa (Semantic Search)

|  |
| --- |
| query\_vec = get\_gemini\_embedding(user\_mood, task\_type="retrieval\_query")  n\_results = 30  results = collection.query(  query\_embeddings=[query\_vec],  n\_results=n\_results  ) |

Bảng 2.10 Module tìm kiếm ngữ nghĩa trong hàm

* Dùng task\_type="retrieval\_query" để vector hóa câu hỏi của người dùng.
* collection.query: So sánh vector câu hỏi với vector trong Database.
* n\_results=30: Lấy ra Top 30 bài hát phù hợp nhất.

Bước 2: Chống lặp bài hát (Thuật toán Smart Shuffle)

|  |
| --- |
| found\_index = -1  # 1. Ưu tiên tìm bài CHƯA từng nghe  for i, song\_id in enumerate(ids\_list):  if song\_id not in played\_history:  found\_index = i  break |

Bảng 2.11 Module chống lặp bài hát

* Logic: Hệ thống duyệt qua danh sách 10 bài tìm được (đã xếp hạng từ phù hợp nhất).
* Điều kiện: Nếu gặp một bài có ID *không nằm trong* played\_history, hệ thống chọn bài đó ngay lập tức (break) và dừng tìm kiếm. Điều này đảm bảo người dùng luôn nghe bài mới nếu có thể.

Bước 3: Sinh lời dẫn DJ (Generative AI - RAG)

|  |
| --- |
| # 3. Nếu KHÔNG tìm thấy bài mới (Đã nghe hết Top 30)  else:  found\_index = 0 # Quay lại bài đầu tiên (Top 1)  selected\_id = ids\_list[found\_index]    # Cập nhật lại vị trí trong lịch sử (LRU logic)  if selected\_id in played\_history:  played\_history.remove(selected\_id)  played\_history.append(selected\_id) |

Bảng 2.12 Module sinh lời dẫn DJ

* Logic: Nếu vòng lặp chạy hết mà không tìm thấy bài nào chưa nghe (tức là người dùng đã nghe hết cả 10 bài gợi ý), hệ thống buộc phải phát lại bài phù hợp nhất (vị trí 0).
* Xử lý lịch sử: Nó xóa ID bài đó ở vị trí cũ trong danh sách và thêm lại vào cuối danh sách. Thao tác này giúp cập nhật trạng thái "mới nhất" cho bài hát.

|  |
| --- |
| prompt = f"""  Bạn là DJ âm nhạc.  Người dùng: "{user\_mood}".  Bài hát chọn: "{song\_title}" (Mô tả: {found\_text}).  Viết câu dẫn ngắn (dưới 20 từ) thật chill.  """  model = genai.GenerativeModel('gemini-2.5-pro')  ai\_response = model.generate\_content(prompt).text |

Bảng 2.13 Modul tạo sinh tăng cường

* Đây là bước Augmented Generation (Tạo sinh tăng cường).
* Ta không chỉ đưa bài hát, mà còn đưa cả ngữ cảnh (người dùng buồn, bài hát mô tả là thất tình) vào Prompt.
* Model ngôn ngữ (gemini-2.5-pro) sẽ đóng vai DJ, sáng tạo ra một câu nói phù hợp. Ví dụ: "Gửi bạn một chút giai điệu để xoa dịu nỗi buồn này." thay vì chỉ hiện tên bài hát khô khan.

Bước 4: Trả về HTML

* Code tạo một chuỗi HTML chứa thẻ <iframe> của YouTube. Thẻ này được nhúng CSS (style) để bo tròn góc, tạo bóng (shadow) cho đẹp mắt.

### Giao diện hoá

|  |
| --- |
| with gr.Blocks(theme=gr.themes.Soft(primary\_hue="indigo"), css=css, title="Music Mood AI") as demo:  history\_state = gr.State([]) |

Bảng 2.14 Giao diện hoá

* gr.Blocks: Khung chứa bố cục trang web.
* history\_state = gr.State([]): (Cực kỳ quan trọng). Trong lập trình web, mỗi lần bấm nút là một lần "reset". Biến thông thường sẽ mất giá trị. gr.State là một biến đặc biệt dùng để lưu trữ trạng thái phiên làm việc. Nó chính là nơi lưu danh sách played\_history (các bài đã nghe) để truyền qua lại giữa các lần bấm nút mà không bị mất.

|  |
| --- |
| with gr.Row():  moods = [("🌧️ Buồn", "Tôi buồn..."), ...]  for label, prompt\_text in moods:  btn = gr.Button(label)  btn.click(  fn=lambda x=prompt\_text: x, outputs=input\_txt  ).then(  fn=recommend\_music\_logic,  inputs=[input\_txt, history\_state],  outputs=[output\_html, history\_state]  ) |

Bảng 2.15 Sự kiện nút bấm nhanh

* Sự kiện liên hoàn (.then):
* btn.click: Khi bấm nút (ví dụ nút "Buồn"), một hàm lambda chạy để điền chữ "Tôi buồn..." vào ô nhập liệu (input\_txt).
* .then(...): Ngay sau khi điền xong, nó tự động kích hoạt hàm tìm kiếm (recommend\_music\_logic).
* Data Flow trong giao diện: Hàm tìm kiếm nhận đầu vào là input\_txt và history\_state. Sau khi chạy xong, nó trả kết quả vào output\_html (để hiện nhạc) và cập nhật lại history\_state mới (để nhớ bài vừa nghe).

|  |
| --- |
| demo.launch(share=True, debug=True) |

Bảng 2.16 Chạy Gradio

* share=True: Gradio sẽ tạo một đường hầm (tunnel) public, cung cấp một đường link https://xxxx.gradio.live có thể truy cập từ bất kỳ đâu (điện thoại, máy tính khác) trong vòng 72h.
* debug=True: Nếu có lỗi, nó in chi tiết ra màn hình Colab để dễ sửa.

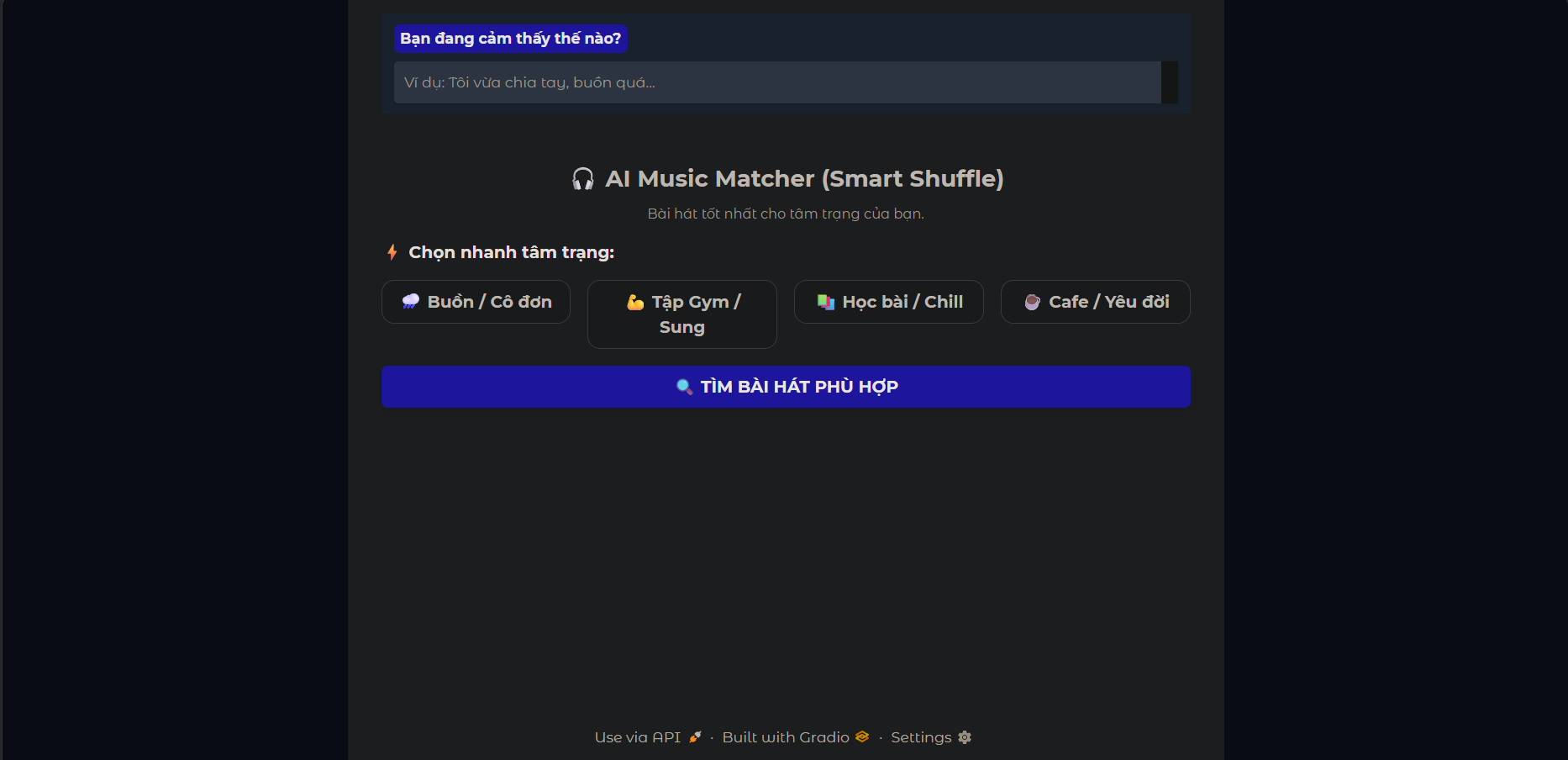
# TRÌNH BÀY KẾT QUẢ

Sau khi cài đặt môi trường và triển khai mã nguồn trên Google Colab, hệ thống Music Mood AI đã hoạt động ổn định. Dưới đây là các kết quả thử nghiệm chi tiết dựa trên các kịch bản người dùng khác nhau.

## Giao diện người dùng hoàn thiện

Giao diện được xây dựng bằng thư viện Gradio với phong cách hiện đại (Theme Soft/Indigo). Bố cục bao gồm:

* Khu vực nhập liệu: Cho phép nhập văn bản tự do hoặc chọn nhanh qua các nút gợi ý (Buồn, Gym, Cafe...).
* Khu vực hiển thị: Trình phát nhạc YouTube được nhúng trực tiếp kèm theo lời dẫn của AI DJ.



Ảnh 3.1 Giao diện hoàn thiện

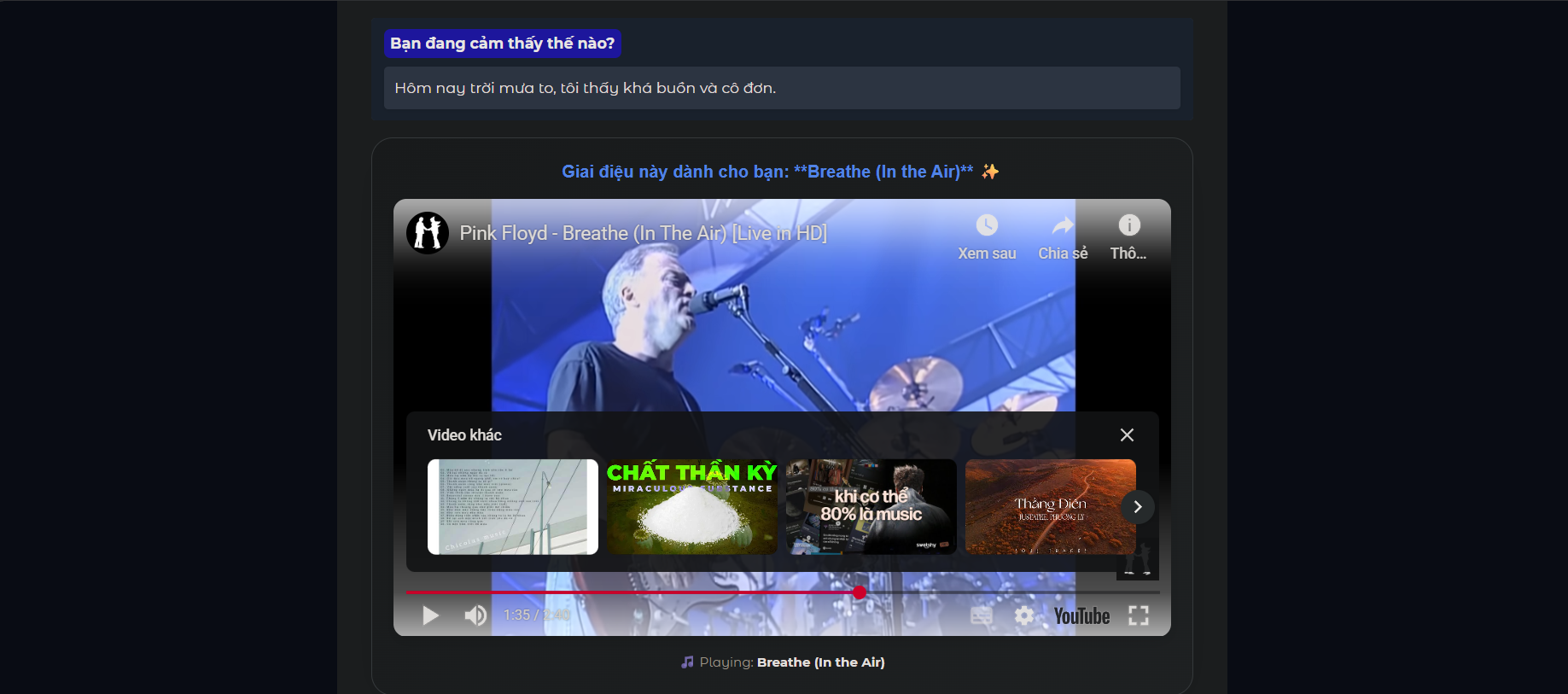
## Kịch bản thử nghiệm (Test Cases)

Để đánh giá độ chính xác của mô hình Embedding và logic RAG, nhóm thực hiện đã tiến hành thử nghiệm với 3 trường hợp điển hình:

### Kịch bản 1: Cảm xúc tiêu cực (Buồn/Cô đơn)

Lần 1:

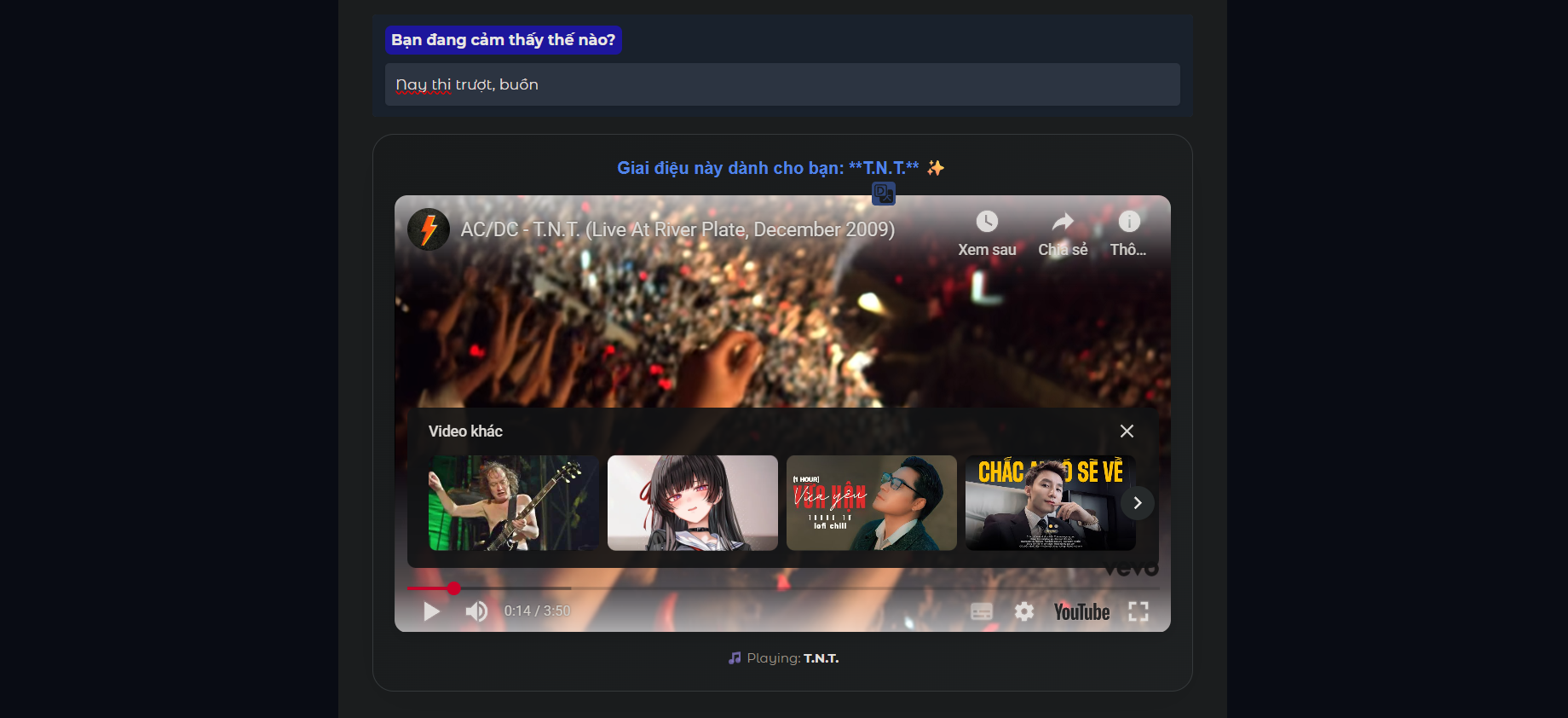
* Input: "Hôm nay trời mưa to, tôi thấy khá buồn và cô đơn.”
* Phân tích hệ thống: Từ khóa "mưa", "buồn", "cô đơn" được Vector hóa và tìm kiếm trong ChromaDB.
* Kết quả trả về:



Ảnh 3.2 Bài hát: “Breathe (In the Air)”, mô tả: “giai điệu êm ái, thư giãn như một hơi thở nhẹ, hoàn hảo để xoa dịu tâm hồn sau ngày dài mệt mỏi”.

Lần 2:

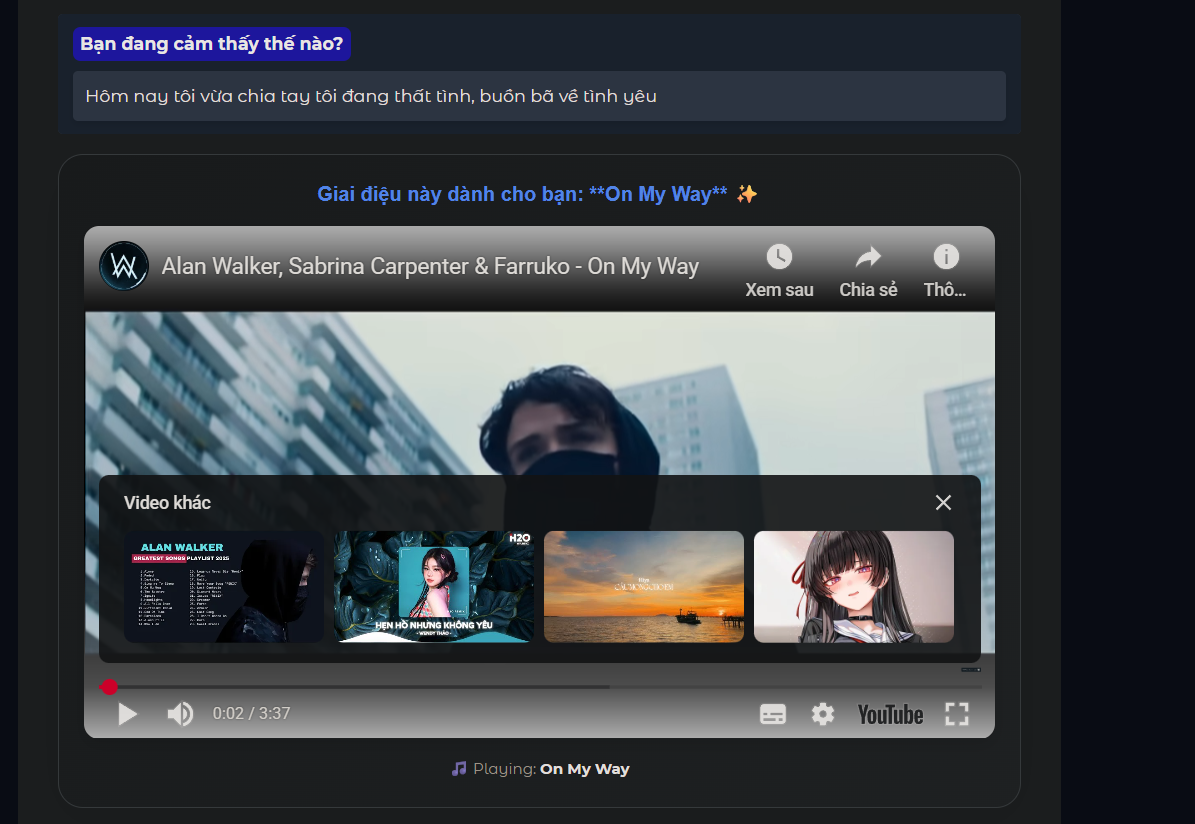
* Input: " Nay thi trượt, buồn.”
* Phân tích hệ thống: Từ khóa "Thi trượt", "buồn", được Vector hóa và tìm kiếm trong ChromaDB.
* Kết quả trả về:



Ảnh 3.3 Bài hát: “T.N.T”, mô tả: “bài hát đầy năng lượng, nhịp điệu mạnh mẽ, tinh thần bùng nổ, phù hợp nghe để xua tan cảm giác mệt mỏi chán nản.”

Lần 3:

* Input: "Hôm nay tôi vừa chia tay, tôi đang thất tình"
* Phân tích hệ thống: Từ khóa “chia tay", "thất tình", được Vector hóa và tìm kiếm trong ChromaDB.
* Kết quả trả về:

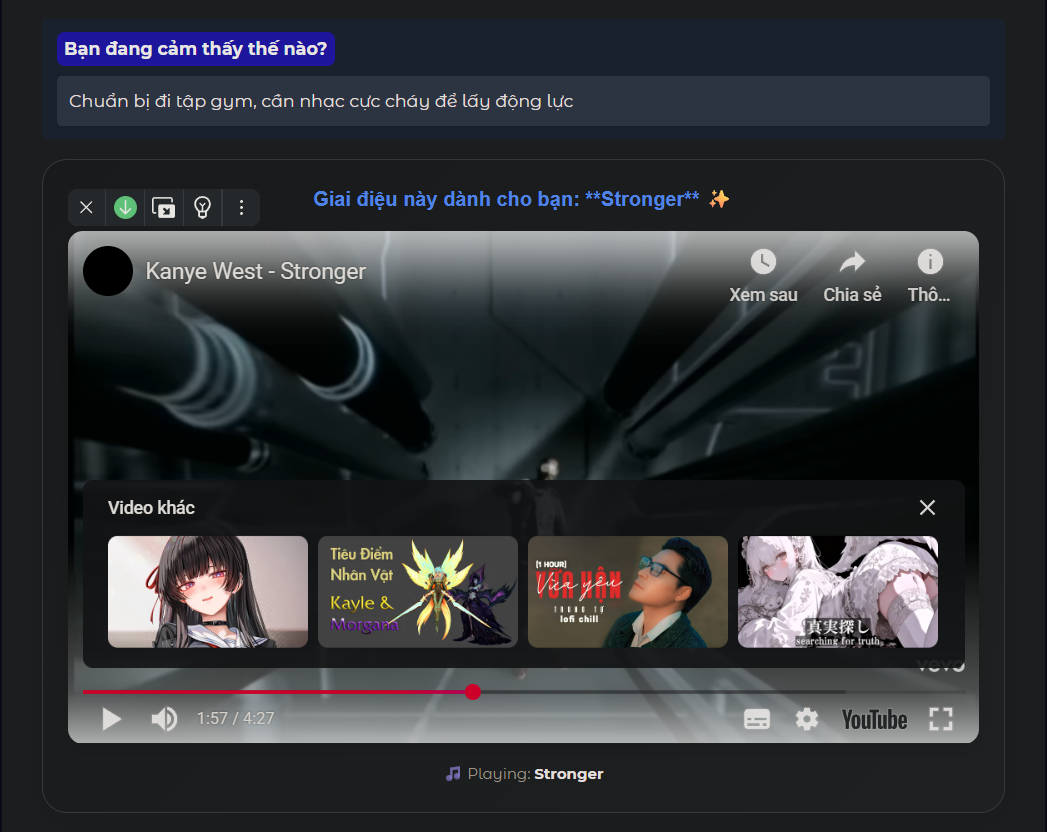


Ảnh 3.4 Bài hát: On My Way, Mô tả: EDM hiện đại, giai điệu truyền cảm hứng mạnh mẽ, phù hợp nghe khi cần động lực làm việc hoặc đi phượt xa.

### Kịch bản 2: Hoạt động năng lượng cao (Tập luyện)

Lần 1:

* Input: " Chuẩn bị đi tập gym, cần nhạc cực cháy để lấy động lực"
* Phân tích hệ thống: Từ khóa "Tập gym", "nhạc cực cháy", "lấy động lực " được Vector hóa và tìm kiếm trong ChromaDB.
* Kết quả trả về:



Ảnh 3.5 Bài hát “Stronger”, mô tả “Electronic-rap mạnh mẽ, lời ca nghị lực, là bài hát không thể thiếu trong danh sách nhạc tập gym để bứt phá giới hạn.”

Lần 2:

* Input: "Tôi chuẩn bị đi tập gym, cần nhạc lofi để tập"
* Phân tích hệ thống: Từ khóa "", "", được Vector hóa và tìm kiếm trong ChromaDB.
* Kết quả trả về:

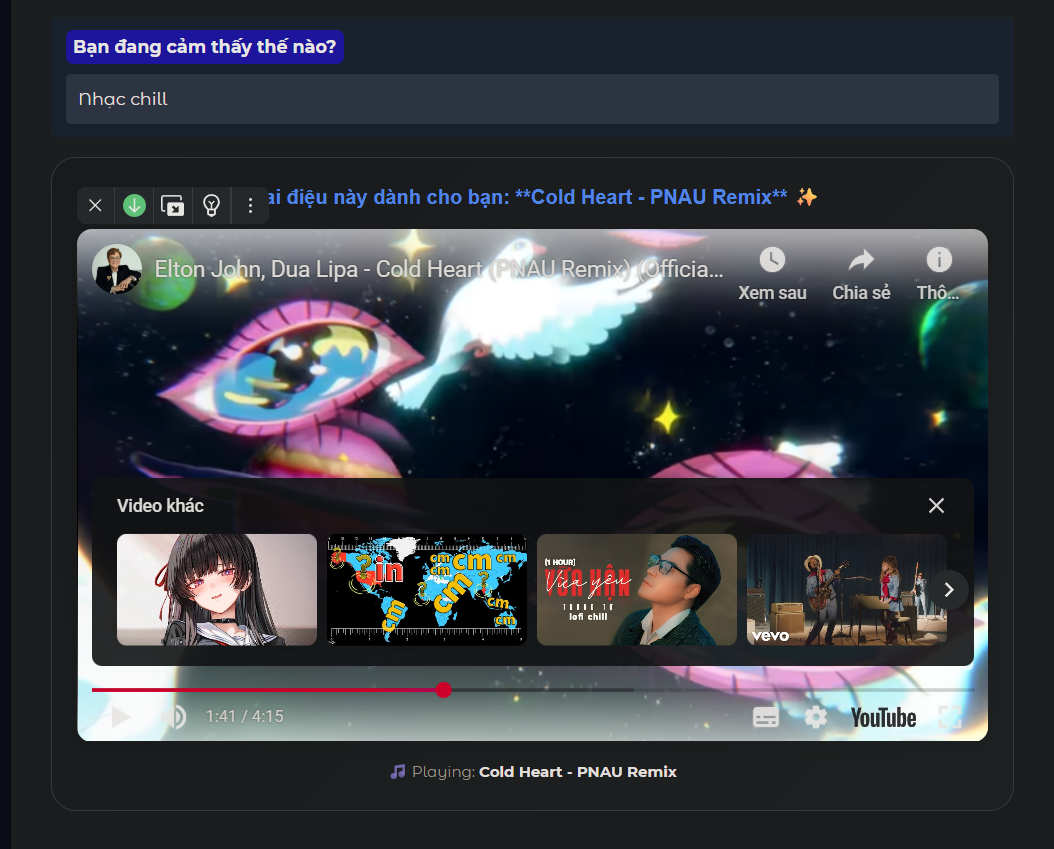


Ảnh 3.6 Bài hát: “Sugar”, mô tả: “Giai điệu cuồng nhiệt, tràn đầy năng lượng thô sơ, phù hợp nghe trong các buổi tập gym hoặc khi lái xe.”

## 3. Đánh giá tính năng "Smart Shuffle" (Chống lặp)

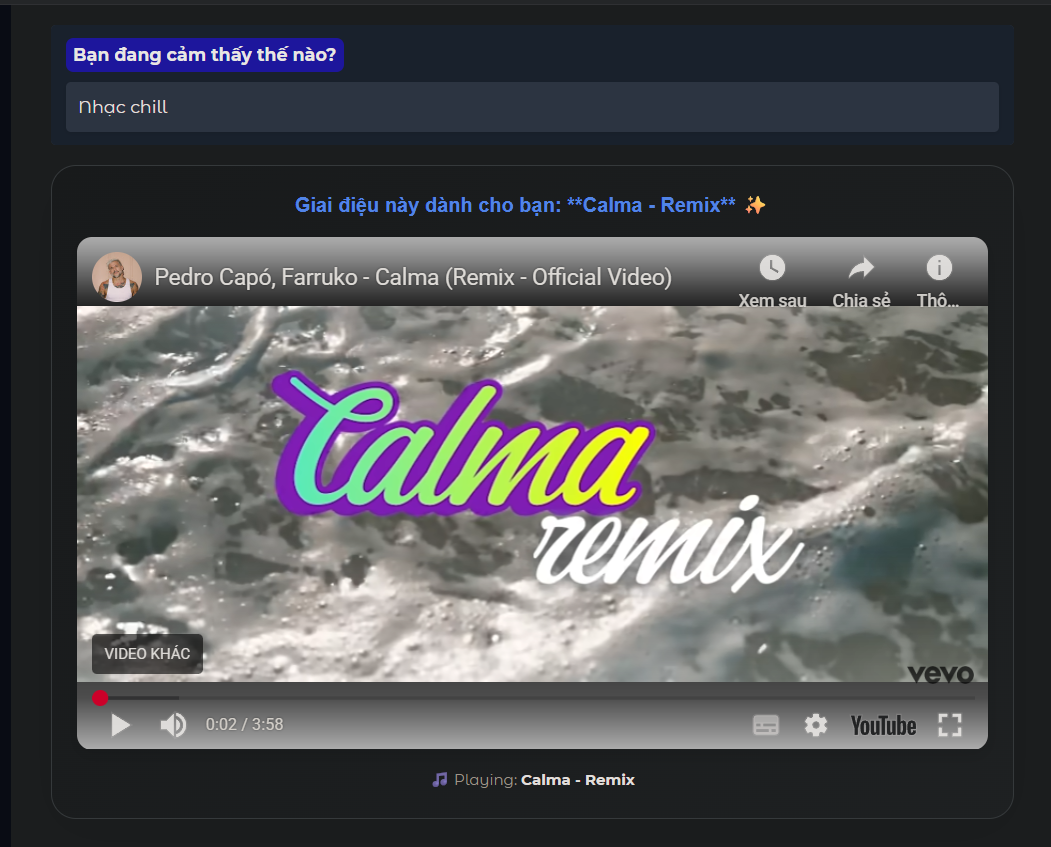
Đây là tính năng kỹ thuật quan trọng nhằm cải thiện trải nghiệm người dùng.

* Lần tìm kiếm 1:
* Input: "Nhạc chill"
* Output:



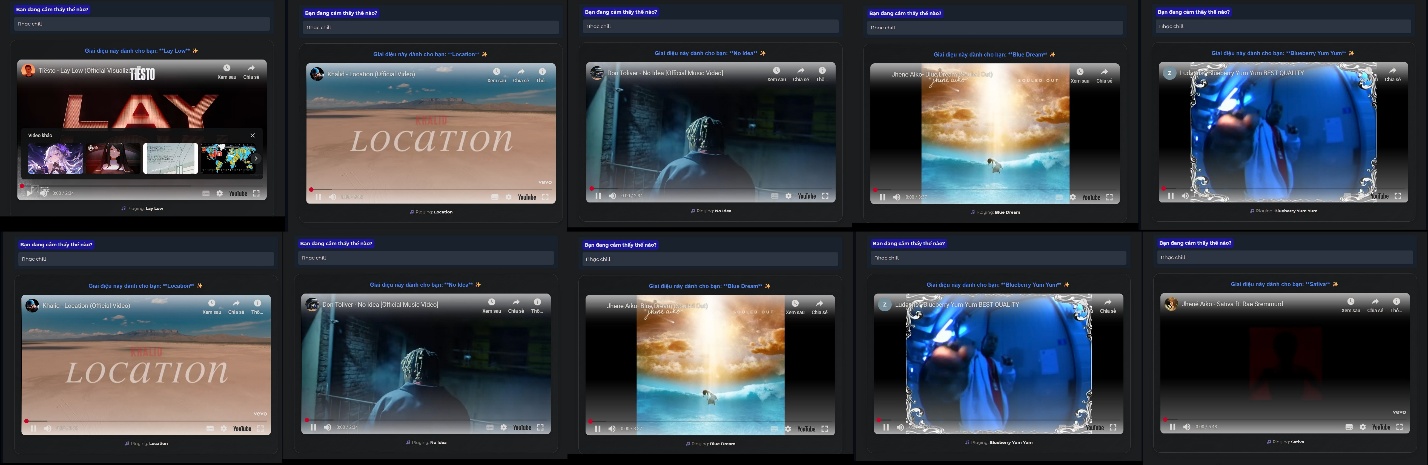
Ảnh 3.7 Bài hát: “Cold Heart - PNAU Remix”, mô tả: “Disco-pop hiện đại, giai điệu chill và bắt tai, lý tưởng nghe khi đang thư giãn vào cuối ngày hoặc dọn dẹp nhà cửa gọn gàng.”

* Lần tìm kiếm 2 (Ngay sau đó):
* Input: " Nhạc chill"
* Xử lý: Hệ thống tìm thấy là kết quả tốt nhất (Top 1), nhưng kiểm tra thấy ID '1' đã có trong lịch sử. Hệ thống tự động bỏ qua và chọn kết quả tốt tiếp theo (Top 2).
* Output:



Ảnh 3.8 Bài hát: “Calma - Remix”, mô tả: “Nhạc Tropical Pop tươi mát, mang hơi thở biển cả, hoàn hảo cho những chuyến du lịch hè hoặc lúc nghỉ ngơi chill.”

* Trạng thái hệ thống: played\_history = ['293', '404'].



Ảnh 3.9 Những lần thử tiếp theo

* Kết luận: Thuật toán hoạt động, đảm bảo người dùng không bị nghe lại một bài hát liên tục dù nhập cùng một ý định tìm kiếm.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận chung

Sau quá trình nghiên cứu và thực hiện, nhóm đã xây dựng thành công ứng dụng Music Mood AI – hệ thống gợi ý âm nhạc dựa trên cảm xúc sử dụng các công nghệ tiên tiến nhất hiện nay là Google Gemini API và ChromaDB. Dự án đã giải quyết được vấn đề cốt lõi đặt ra ban đầu là sự hạn chế của các công cụ tìm kiếm nhạc theo từ khóa truyền thống.

Các kết quả đạt được bao gồm:

* Xây dựng thành công kiến trúc RAG: Ứng dụng đã vận hành trơn tru quy trình Truy xuất (Retrieval) bài hát từ cơ sở dữ liệu vector và Tăng cường (Augmented) ngữ cảnh để AI sinh lời dẫn.
* Hiểu ngôn ngữ tự nhiên: Hệ thống có khả năng phân tích chính xác ý định của người dùng thông qua các câu lệnh phức tạp (ví dụ: "buồn nhưng muốn tập trung học") nhờ mô hình Embedding.
* Trải nghiệm người dùng thông minh: Tính năng "Smart Shuffle" đã hoạt động hiệu quả trong việc loại bỏ sự lặp lại bài hát, đảm bảo sự tươi mới cho trải nghiệm nghe nhạc.
* Cá nhân hóa tương tác: Vai trò "DJ ảo" được thực hiện tốt, mang lại cảm giác gần gũi thông qua các lời dẫn chuyện phù hợp với từng ngữ cảnh cảm xúc cụ thể.

## Hạn chế của đề tài

Bên cạnh những kết quả đạt được, nhóm cũng nhận thấy hệ thống còn tồn tại một số hạn chế cần khắc phục:

* Dữ liệu tĩnh: Hiện tại, danh sách bài hát đang được nạp thủ công từ file CSV. Hệ thống chưa có khả năng tự động cập nhật các bài hát mới thịnh hành (Trending) trên thị trường.
* Lưu trữ phiên làm việc: Do sử dụng ChromaDB ở chế độ In-memory (chạy trên RAM) và biến history\_state của Gradio, lịch sử nghe nhạc của người dùng sẽ bị mất khi tắt trình duyệt hoặc khởi động lại server.
* Phụ thuộc vào bên thứ ba: Hoạt động của ứng dụng phụ thuộc hoàn toàn vào tốc độ và giới hạn (Quota) của Google Gemini API. Nếu kết nối mạng không ổn định hoặc hết hạn mức API, hệ thống sẽ không thể phản hồi.

## Hướng phát triển trong tương lai

Để đưa Music Mood AI trở thành một sản phẩm hoàn thiện và có tính thực tiễn cao hơn, nhóm đề xuất các hướng phát triển tiếp theo:

* Mở rộng kho dữ liệu: Tích hợp các công cụ thu thập dữ liệu tự động (Crawler) để làm giàu kho nhạc với hàng nghìn bài hát mới mỗi ngày thay vì dữ liệu tĩnh.
* Lưu trữ bền vững (Persistent Storage): Chuyển đổi từ ChromaDB in-memory sang chế độ lưu trữ trên ổ cứng hoặc sử dụng các dịch vụ Cloud Database để lưu giữ lịch sử nghe nhạc và sở thích của người dùng lâu dài.
* Tích hợp giọng nói (Voice Interaction): Bổ sung tính năng Speech-to-Text để người dùng có thể trò chuyện trực tiếp bằng giọng nói với DJ ảo thay vì phải gõ phím.
* Phân tích đa phương thức: Không chỉ dựa vào văn bản mô tả, hệ thống có thể được nâng cấp để phân tích trực tiếp tín hiệu âm thanh (Audio Analysis) của bài hát để trích xuất các đặc trưng như nhịp độ (BPM), cao độ, giúp việc gợi ý chính xác hơn về mặt kỹ thuật âm nhạc.

**TÀI LIỆU THAM KHẢO**

[1] Y. Deldjoo, M. Schedl, P. Knees, Content-driven music recommendation: Evolution, state of the art, and challenges, Computer Science Review, vol. 51, 2024, 100618.

[2] E. Palumbo, G. Penha, A. Damianou, J. L. Redondo García, T. C. Heath, A. Wang, H. Bouchard, M. Lalmas, Text2Tracks: Prompt-based Music Recommendation via Generative Retrieval

[3] M. Schedl, C. Bauer, W. Reisinger, D. Kowald, E. Lex, Listener Modeling and Context-Aware Music Recommendation Based on Country Archetypes, Frontiers in Artificial Intelligence, 2020.

[4] Google AI for Developers, Embeddings, Gemini API Documentation

[5] Google Cloud, Text embeddings API, Generative AI on Vertex AI

[6] Chroma Docs, Add Data (Adding Data to Chroma Collections

[7] J. Carbonell, J. Goldstein, The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries, 1998.

[8] A. Tommasel, J. M. Rodriguez, D. Godoy, Haven’t I just Listened to This?: Exploring Diversity in Music Recommendations, UMAP 2022 Adjunct Proceedings, 2022

[9] Y. Li, X. Fu, G. Verma, P. Buitelaar, M. Liu, Mitigating Hallucination in Large Language Models (LLMs): An Application-Oriented Survey on RAG, Reasoning, and Agentic Systems

[10] Google AI for Developers, Migrate to the Google GenAI SDK, Gemini API

[11] PyPI, google-generativeai (Deprecated) - Google AI Python SDK for the Gemini API

[12] Chroma Docs, Query and Get (collection.query, query\_embeddings, n\_results)

[13] Gradio, State in Blocks (Quản lý trạng thái trong gr.Blocks và gr.State)