BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN

**TRƯỜNG ĐẠI HỌC THỦY LỢI**

A blue and white logo

Description automatically generated

**BÀI TẬP LỚN**

**HỌC PHẦN: PHÁT TRIỂN ỨNG DỤNG CHO CÁC THIẾT BỊ DI DỘNG**

**ĐỀ TÀI: Xây dựng ứng dụng Music App (Android Studio)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Lớp** |
| 2151173751 | Đặng Văn Bình | 1/11/2003 | 63KTPM1 |
| 2151173779 | Nguyễn Quang Huy | 03/04/2003 | 63KTPM1 |
| 2151170586 | Nguyễn Đức Huy | 27/11/2003 | 63KTPM1 |
| 2151170555 | Trần Huy Hoàng | 24/09/2003 | 63KTPM1 |

### 

**Hà Nội, năm 2024**

BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN

**TRƯỜNG ĐẠI HỌC THỦY LỢI**

A blue and white logo

Description automatically generated

**BÀI TẬP LỚN**

**HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

**ĐỀ TÀI: Xây dựng ứng dụng Music App (Android Studio)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| 2151173751 | Đặng Văn Bình | 1/11/2003 |  |  |
| 2151173779 | Nguyễn Quang Huy | 03/04/2003 |  |  |
| 2151170586 | Nguyễn Đức Huy | 27/11/2003 |  |  |
| 2151170555 | Trần Huy Hoàng | 24/09/2003 |  |  |

### 

**CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2**

**Hà Nội, năm 2024**

**LỜI NÓI ĐẦU**

Trong thời đại công nghệ số, nhu cầu lưu trữ và quản lý nội dung số, đặc biệt là âm nhạc, ngày càng trở nên phổ biến. Tuy nhiên, người dùng thường gặp khó khăn trong việc quản lý bộ sưu tập nhạc cá nhân một cách hiệu quả. Với sự phát triển của các nền tảng nghe nhạc trực tuyến, người dùng vẫn có nhu cầu lưu giữ các bài nhạc cá nhân, những bản thu âm tự sáng tác, hay các tác phẩm không có sẵn trên các nền tảng lớn.

Xuất phát từ nhu cầu này, ứng dụng [My Music] được phát triển nhằm giúp người dùng dễ dàng tải lên, lưu trữ và phân loại các bản nhạc của mình. Bên cạnh đó, ứng dụng còn cung cấp trải nghiệm nghe nhạc tiện dụng và thân thiện với người dùng. Mục tiêu của ứng dụng không chỉ là một công cụ lưu trữ mà còn là một nền tảng giúp người dùng tận hưởng âm nhạc theo cách cá nhân hóa nhất.

**MỤC LỤC**

[**CHƯƠNG 1: MÔ TẢ BÀI TOÁN** 4](#_Toc181305399)

[**1.1.** **Giới thiệu** 4](#_Toc181305400)

[**1.2.** **Chức năng chính** 4](#_Toc181305401)

[**1.3.** **Yêu cầu phi chức năng** 5](#_Toc181305402)

[**CHƯƠNG 2: PHÂN TÍCH YÊU CẦU HỆ THỐNG** 6](#_Toc181305403)

[**2.1.** **Phân tích yêu cầu:** 6](#_Toc181305404)

[**2.2.** **Thiết kế hệ thống** 9](#_Toc181305405)

[Biểu đồ lớp: 9](#_Toc181305406)

[Xác định các lớp: 10](#_Toc181305407)

[Mối quan hệ giữa các lớp: 11](#_Toc181305408)

[Thiết kế cơ sở dữ liệu: 11](#_Toc181305409)

[Thiết kế giao diện: 14](#_Toc181305410)

[**2.3.** **Triển khai:** 19](#_Toc181305411)

[ **Viết code:** Sử dụng App [Android Studio] và dùng ngôn ngữ [Java]. 19](#_Toc181305412)

[ **Kiểm thử:** 19](#_Toc181305413)

[**2.4.** **Vận hành và bảo trì:** 23](#_Toc181305414)

[**CHƯƠNG 3: KẾT QUẢ THỰC HIỆN** 24](#_Toc181305415)

[**3.1.** **Công nghệ đã sử dụng** 24](#_Toc181305416)

[**3.2.** **Tiến độ thực hiện** 24](#_Toc181305417)

[Link github tới dự án: [https://github.com/lich1101/CSE441\_PROJECT.git] 24](#_Toc181305418)

[KẾT LUẬN 25](#_Toc181305419)

[**PHỤ LỤC** 26](#_Toc181305420)

**CHƯƠNG 1: MÔ TẢ BÀI TOÁN**

* 1. **Giới thiệu**

Ứng dụng [My Music] là giải pháp cho việc lưu trữ, quản lý và phân loại các file nhạc cá nhân một cách tiện lợi. Ứng dụng cho phép người dùng tải lên các bản nhạc của riêng mình, dễ dàng phân loại theo các mục, thể loại và tạo ra không gian cá nhân giúp cá nhân hóa cho việc trải nghiệm nghe nhạc.

Ngoài việc lưu trữ, ứng dụng còn tập trung vào việc mang lại trải nghiệm nghe nhạc tối ưu, thân thiện với giao diện trực quan, giúp người dùng dễ dàng truy cập và thưởng thức âm nhạc mọi lúc, mọi nơi. Chúng tôi hy vọng [My Music] sẽ trở thành công cụ đắc lực cho những ai yêu âm nhạc và muốn tự do khám phá, quản lý bộ sưu tập nhạc cá nhân một cách chuyên nghiệp.

* 1. **Chức năng chính**

Ứng dụng [My Music] cần có những chức năng sau:

* Tải lên bài nhạc: Cho phép người dùng có thể tải lên bài nhạc của bản thân, gồm có file âm nhạc, tên bài nhạc, mô tả (nếu cần).
* Xóa bài nhạc: Cho phép người dùng có thể xóa đi các bài nhạc mà người dùng không muốn.
* Hiển thị danh sách Playlist: Cho phép người dùng có thể xem các bài nhạc có trong Playlist của người dùng.
* Đánh dấu bài hát yêu thích: Cho phép người dùng có thể tạo 1 danh sách các bài hát được yêu thích và tích chọn những bài hát mà mình yêu thích vào để thêm chúng vào danh sách này.
* Tạo Playlist riêng: Cho phép người dùng tạo ra 1 playlist cá nhân giúp người dùng phân loại và quản lý các bài hát 1 cách dễ dàng.
* Tìm kiếm bài nhạc: Cho phép người dùng có thể tìm kiếm theo tên các bài nhạc, tìm kiếm theo tên các Playlists.
* Phát bài nhạc: Cho phép người dùng chạy tiến trình nghe nhạc.
* Tạm dừng bài nhạc: Cho phép người dùng tạm dừng tiến trình nghe nhạc.
* Phát bài nhạc trước: Cho phép người dùng có thể quay lại bài hát trước đó.
* Phát bài nhạc tiếp theo: Cho phép người dùng có thể chuyển tiếp sang bài hát kế tiếp.
* Chỉnh timeline bài nhạc: Cho phép người dùng điều chỉnh thanh timeline của bài nhạc.
  1. **Yêu cầu phi chức năng**
* Khả năng sử dụng: Giao diện người dùng phải thân thiện, dễ sử dụng và không yêu cầu người dùng phải có kiến thức kỹ thuật cao. Người dùng phải có thể dễ dàng tải lên nhạc, sắp xếp và phát nhạc mà không gặp rắc rối về thao tác.
* Hiệu năng: Ứng dụng phải có khả năng xử lý và phản hồi nhanh chóng khi người dùng tải nhạc lên hoặc truy cập danh sách nhạc.
* Khả năng mở rộng: Hệ thống phải cho phép mở rộng dung lượng lưu trữ khi số lượng bài hát hoặc người dùng tăng lên.
* Bảo mật: Dữ liệu của người dùng, bao gồm các tệp nhạc và thông tin tài khoản, phải được bảo mật bằng cách mã hóa trong quá trình truyền tải và lưu trữ.
* Khả năng bảo trì: Mã nguồn và kiến trúc hệ thống phải được thiết kế để dễ bảo trì, với khả năng sửa lỗi hoặc nâng cấp nhanh chóng.

# **CHƯƠNG 2: PHÂN TÍCH YÊU CẦU HỆ THỐNG**

* 1. **Phân tích yêu cầu:**

Xác định người dùng:

* Người dùng cuối là những người không muốn phụ thuộc vào các nền tảng nhạc trực tuyến, những người sáng tạo nội dung âm thanh, những nghệ sĩ độc lập và nhạc sĩ, người yêu thích âm nhạc cá nhân.
* Người dùng có thể có các mức độ am hiểu về máy tính khác nhau, do đó ứng dụng cần dễ sử dụng và thân thiện với người dùng.

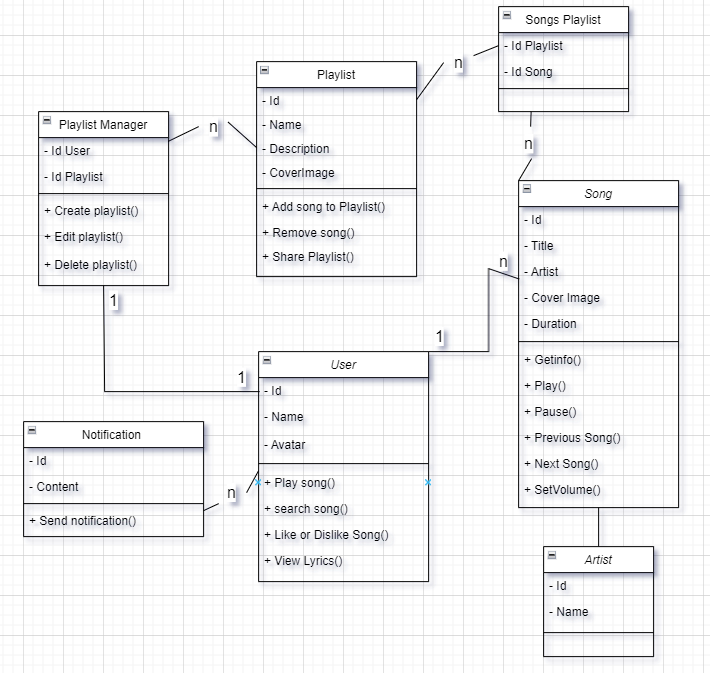
Thu thập yêu cầu:

* Dựa trên mô tả bài toán, ta đã xác định được các chức năng chính (tải nhạc, xóa nhạc, hiển thị danh sách Playlist, đánh dấu bài hát yêu thích, tạo playlist riêng, tìm kiếm bài nhạc, phát bài nhạc, tạm dùng bài nhạc, phát bài nhạc tiếp theo, chỉnh timeline bài nhạc) và yêu cầu phi chức năng (khả năng sử dụng, hiệu năng, khả năng mở công việc, bảo mật, khả năng bảo trì) của ứng dụng.

Phân tích yêu cầu:

* Tải nhạc:
  + Cho phép người dùng chọn tệp nhạc từ thiết bị.
  + Kiểm tra định dạng.
  + Hiển thị quá trình tải lên với thanh tiến trình và thông báo thành công thất bại.
  + Giới hạn dung lượng tệp nhạc và số lượng bài nhạc mà người dùng có thể tải lên dựa trên gói dịch vụ.
* Xóa nhạc:
  + Hiẻn thị nút xóa bên cạnh mỗi bài nhạc trong danh sách nhạc của người dùng.
  + Khi người dùng bấm nút xóa hệ thống sẽ yêu cầu xác nhận trước khi thực hiện.
  + Sau khi xác nhận bài nhạc được xóa khỏi bộ sư tập của người dùng và khỏi CSDL.
  + Thông báo cho người dùng biết kết quá xóa (Thành công hoặc thất bại).
* Hiển thị danh sách Playlist:
  + - * Hiển thị danh sách tất cả các Playlist mà người dùng sở hữu hoặc đã lưu vào thư viện cá nhân.
      * Cho phép người dùng nhấn vào tên Playlist để xem các bài nhạc trong đó.
      * Hiển thị các thông tin liên quan như số lượng bài nhạc, tên Playlist, ngày tạo.
      * Cần phân trang nếu danh sách Playlist quá dài, đảm bảo hiển thị mượt mà.
* Đánh giấu bài hát yêu thích:
* Hiểu thị biểu tượng yêu thích bên cạnh mỗi bài nhạc.
* Người dùng nhấn vào biển tượng để đánh dấu hoặc bỏ đánh dấu bài hát yêu thích.
* Tự động thêm bài hát vào danh sách các bài hát yêu thích của người dùng.
* Người dùng có thể tìm và truy cập dễ dàng danh sách các bài hát đã đánh dấu yêu thích.
* Tạo Playlist riêng:
* Người dùng nhập tên Playlist và nhấn nút tạo.
* Cho phép người dùng thêm các bài hát từ thư viện cá nhân vào Playlist.
* Cập nhập danh sách Playlist sau khi tạo thành công và hiển thị cho người dùng.
* Người dùng có thể tạo số lượng Playlist tối đa tùy theo gói dịch vụ hoặc dung lượng tài khoản.
* Tìm kiếm bài nhạc:
* Cung cấp thanh tìm kiếm để người dùng nhập từ khóa (tên bài hát, nghệ sĩ, album).
* Hiển thị kết quả tìm kiếm theo từ khóa dưới dạng danh sách sắp xếp theo mức độ liên quan.
* Người dùng có thể chọn bài hát từ kết quả để phát hoặc thêm vào Playlist.
* Hệ thống cần có khả năng tìm kiếm theo nhiều tiêu chí như tên bài hát, nghệ sĩ, album, thể loại.
* Phát bài nhạc:
* Người dùng nhấn vào bài nhạc muốn phát.
* Ứng dụng sẽ tải tệp nhạc từ server và bắt đầu phát.
* Hiển thị giao diện phát nhạc với các nút điều kiển (chơi, tạm dừng, tua nhanh/tua lại).
* Hiểu thị thông tin bài hát đang phát (tên, nghệ sĩ, album, thời gian phát).
* Đảm bảo chất lượng âm thanh ổn định và không bị gián đoạn.
* Tạm dừng bài nhạc
  + Cung cấp nút “Tạm dừng” trong giao diện phát nhạc.
  + Khi nhấn, bài nhạc sẽ tạm ngưng phát và lưu lại vị trí hiện tại.
  + Người dùng co thể nhấn “Phát tiếp” để tiếp tục phát từ vị trí đã dừng.
  + Lưu trữ trạng thái của bài nhạc khi tạm dùng không yêu cầu tải lại từ đầu khi phát tiếp.
* Phát bài nhạc trước:
  + Cung cấp nút “Phát trước” trong giao diện phát nhạc.
  + Khi nhấn, ứng dụng sẽ phát bài nhạc ngay trước bài hiện tại trong danh sách phát hoặc Playlist.
  + Nếu bài hiện tại là bài đầu tiên trong danh sách, hành động này sẽ phái bài cuối cùng hoặc không làm gì, tùy thuộc vào thiết lập.
  + Đảm bảo phát lại bài trước ngay lập tức mà không bị gián đoạn.
* Phát bài nhạc tiếp theo
  + Cung cấp nút “Phát tiếp” trong giao diện phát nhạc.
  + Khi nhấn, hệ thống sẽ phát bài nhạc tiếp theo trong danh sách hiện tại.
  + Nếu đang phát bài cuối cùng trong danh sách ứng dụng sẽ phát lại từ đầu hoặc dừng phát tùy theo cấu hình.
  + Đảm bảo chuyển bài nhanh chóng và không gián đoạn.
* Chỉnh timeline bài nhạc
  + Hiển thị thanh time line cho phép người dùng thấy tổng thời gian và thời gian hiện tịa của bài hát.
  + Người dùng có thể kéo thanh timeline để tua nhanh hoặc tua lại đến ví trí mong muốn.
  + Bài nhạc sẽ tiếp tục phát từ vị trí được chọn sau khi tua lại.
  + Ứng dụng cần phản hồi nhanh và chính xác khi người dùng thay đổi timeline mà không làm gián đoạn phát nhạc.
  1. **Thiết kế hệ thống**

### Biểu đồ lớp:



### Xác định các lớp:

* **Lớp User:**
  + **Thuộc tính:** ID, Name, Avatar.
  + **Phương thức:** 
    - PlaySong(): Chạy bài hát.
    - SearchSong(): Tìm kiếm bài hát.
    - LikeSong(): Yêu thích bài hát.
    - ViewLyrics(): Xem lời bài hát
* **Lớp Song:**
  + **Thuộc tính:** ID, Title, Lyrics, CoverImage, PlayCount.
  + **Phương thức:** 
    - GetInfo(): Thông tin bài hát.
    - Play(): Phát bài hát.
    - Pause(): Dừng phát bài hát.
    - PreviousSong(): Quay về bài hát trước.
    - NextSong(): Chuyển tiếp bài hát.
    - SetVolume(): Điều chỉnh âm thanh.
* **Lớp Artist:**
  + **Thuộc tính:** ID, Name.
* **Lớp Playlist:**
  + **Thuộc tính:** ID, Name, Description, CoverImage.
  + **Phương thức:** 
    - AddSongForPlaylist(): Thêm 1 bài hát vào Playlist.
    - RemoveSong(): Xóa bài hát khỏi Playlist.
    - SharePlaylist(): Chia sẻ Playlist.
* **Lớp Songs Playlist:**
  + **Thuộc tính:** ID Playlist, ID Song.
* **Lớp Playlist Manager:**
  + **Thuộc tính:** ID User, ID Playlist
  + **Phương thức:** 
    - CreatePlaylist(): Tạo 1 Playlist mới.
    - DeletePlaylist(): Xóa 1 Playlist đã có.
    - EditPlaylist(): Sửa 1 Playlist.
* **Lớp Notification:**
  + **Thuộc tính:** ID, Content.
  + **Phương thức:**
    - SendNotification(): Gửi thông báo.

### Mối quan hệ giữa các lớp:

* User sử dụng Song để có thể nghe nhạc.
* Song và Playlist có mối liên hệ với nhau thông qua lớp trung gian là Songs Playlist
* User sử dụng Playlist thông qua trung gian là lớp Playlist Manager
* User nhận thông báo từ lớp Notification

### Thiết kế cơ sở dữ liệu:

* Bảng User (Người dùng)

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| user\_id | INT | Khóa chính, định danh người dùng |
| name | VARCHAR | Tên người dùng |
| avatar | VARCHAR | Đường dẫn ảnh đại diện |

* Bảng Song (Bài hát)

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| song\_id | INT | Khóa chính, định danh bài hát |
| title | VARCHAR | Tên bài hát |
| cover\_image | VARCHAR | Đường dẫn ảnh bìa |
| duration | TIME | Thời gian phát nhạc |
| artist\_id | INT | Khóa ngoại, liên kết với bảng Artist |

* Bảng Playlist (Danh sách phát)

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| playlist\_id | INT | Khóa chính, định danh playlist |
| name | VARCHAR | Tên danh sách phát |
| description | TEXT | Mô tả danh sách phát |
| cover\_image | VARCHAR | Ảnh bìa của playlist |

* Bảng Songs Playlist

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| playlist\_id | INT | Khóa ngoại, liên kết với bảng Playlist |
| song\_id | INT | Khóa ngoại, liên kết với bảng Song |

* Bảng Playlist Manager

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| user\_id | INT | Khóa ngoại, liên kết với bảng User |
| playlist\_id | INT | Khóa ngoại, liên kết với bảng Playlist |

* Bảng Notification (Thông báo)

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| notification\_id | INT | Khóa chính |
| content | TEXT | Nội dung thông báo |
| user\_id | INT | Khóa ngoại, liên kết với bảng User |

* Bảng Artist (Nghệ sĩ)

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| playlist\_id | INT | Khóa chính, định danh nghệ sĩ |
| name | VARCHAR | Tên nghệ sĩ |

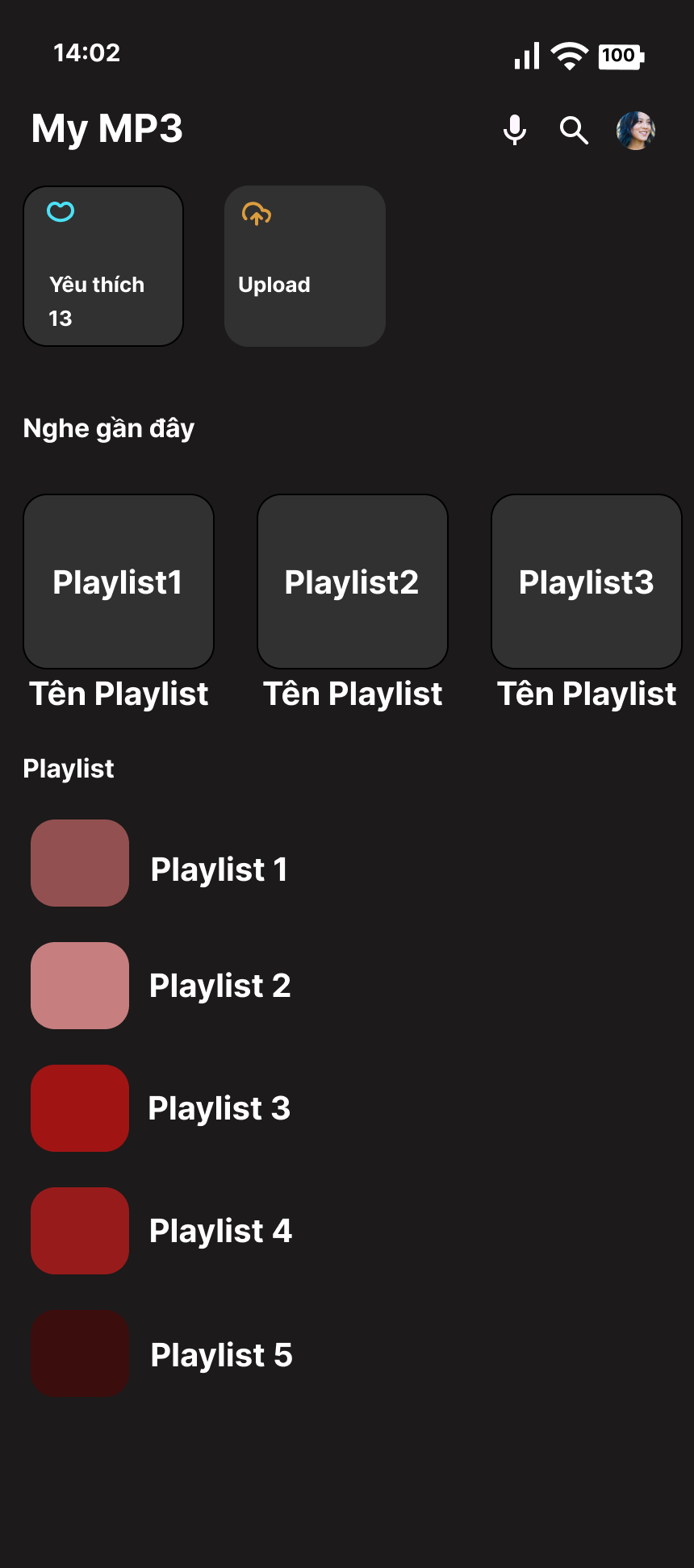
### 

### Thiết kế giao diện:

* Hiển thị Logo App:



* Hiển thị Menu chính



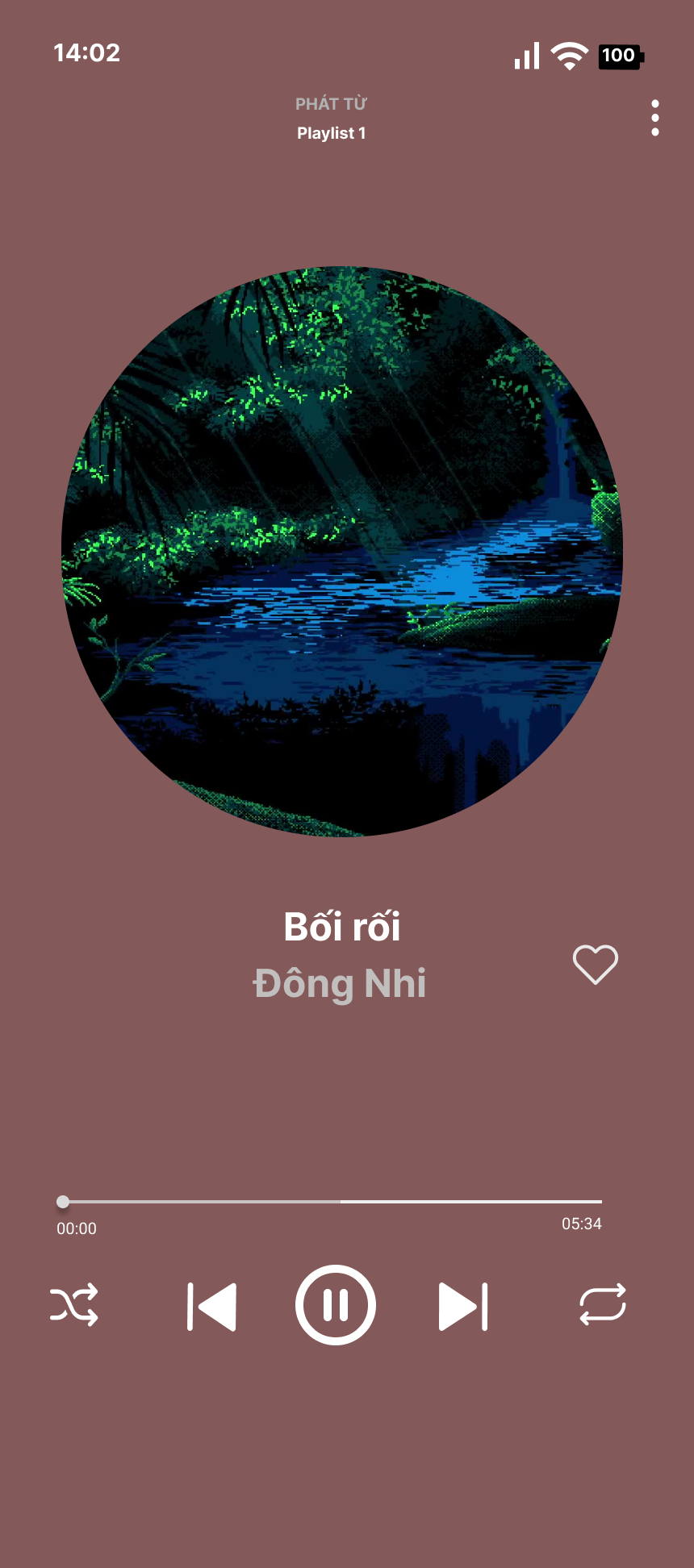
* Giao diện bài hát được yêu thích:



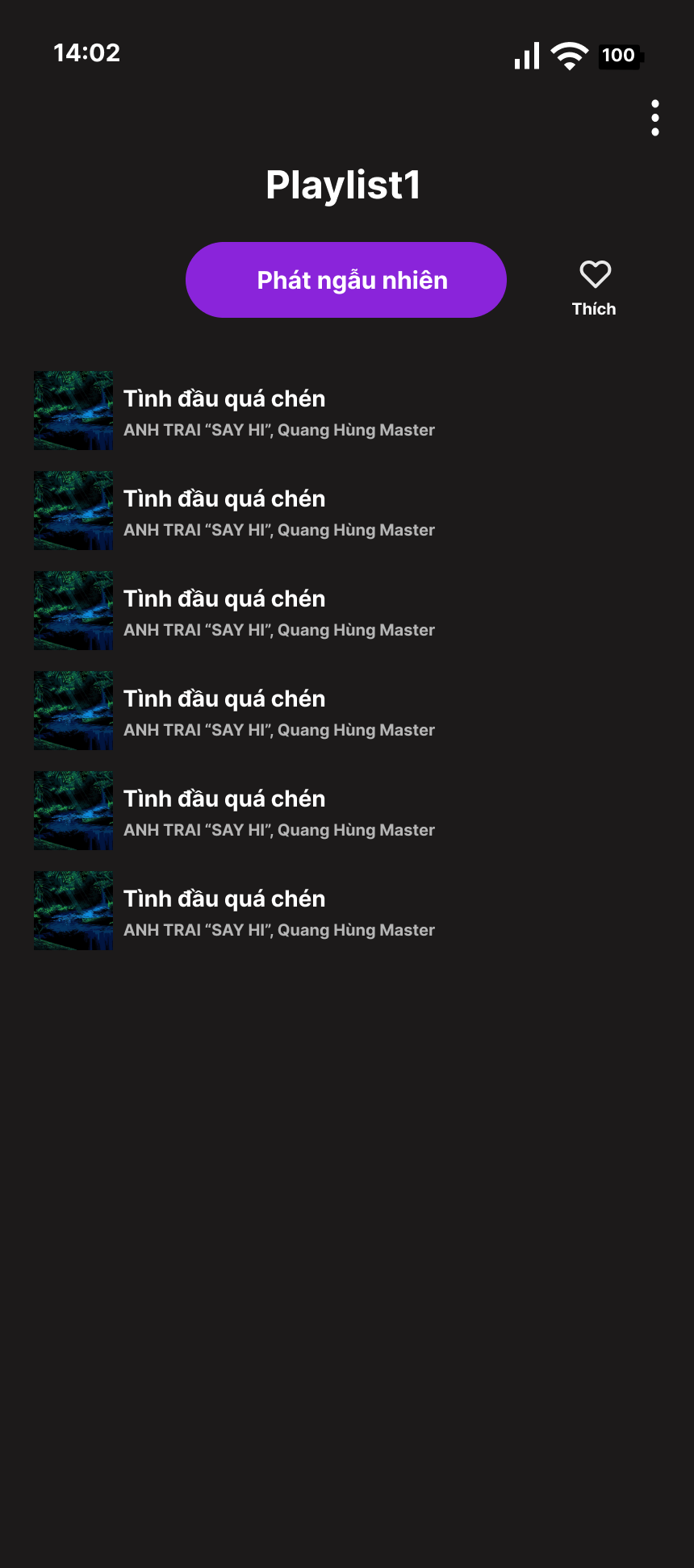
* Giao diện Tìm kiếm bài hát:



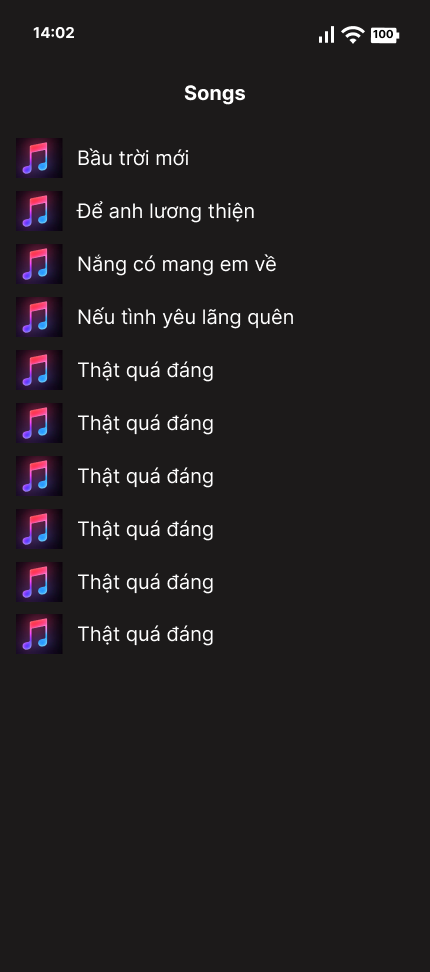
* Giao diện Phát bài hát:



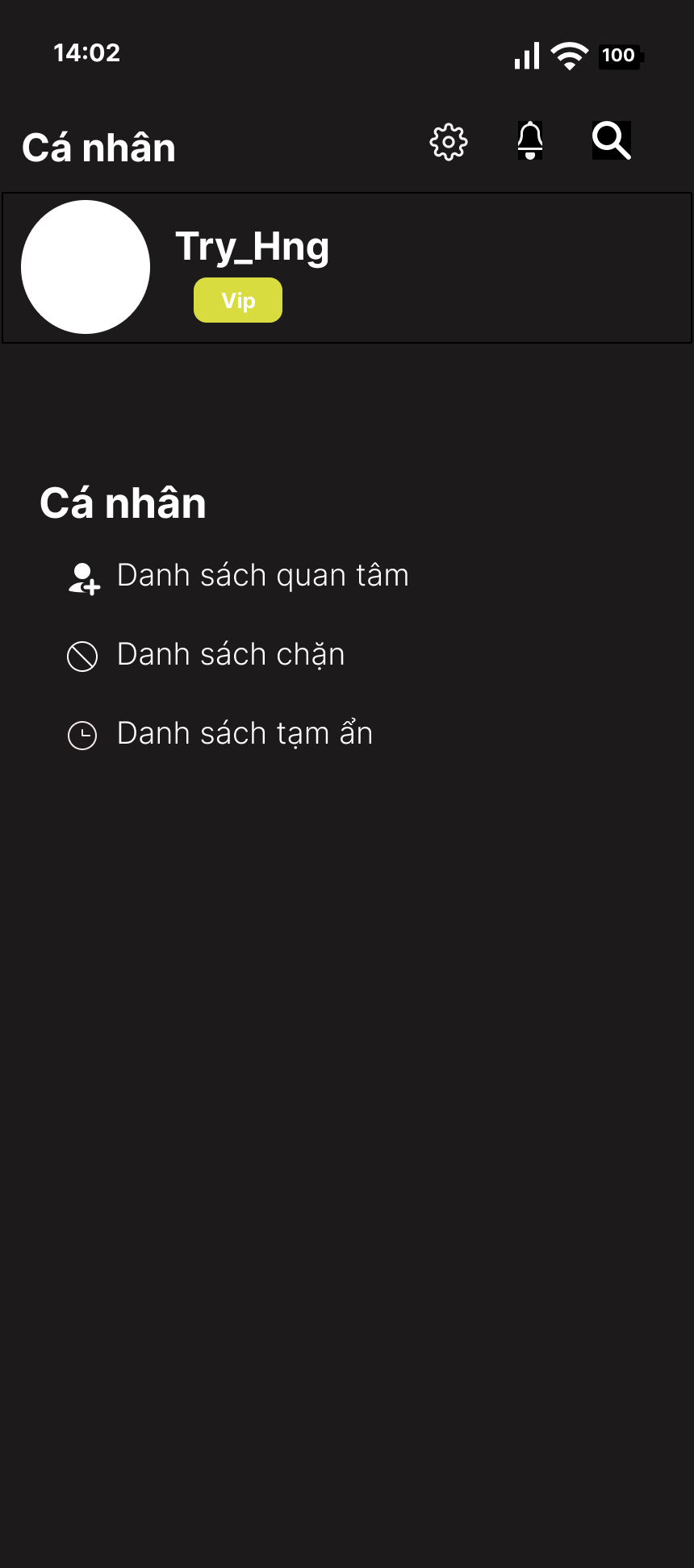
* Giao diện Hiển thị Playlist:



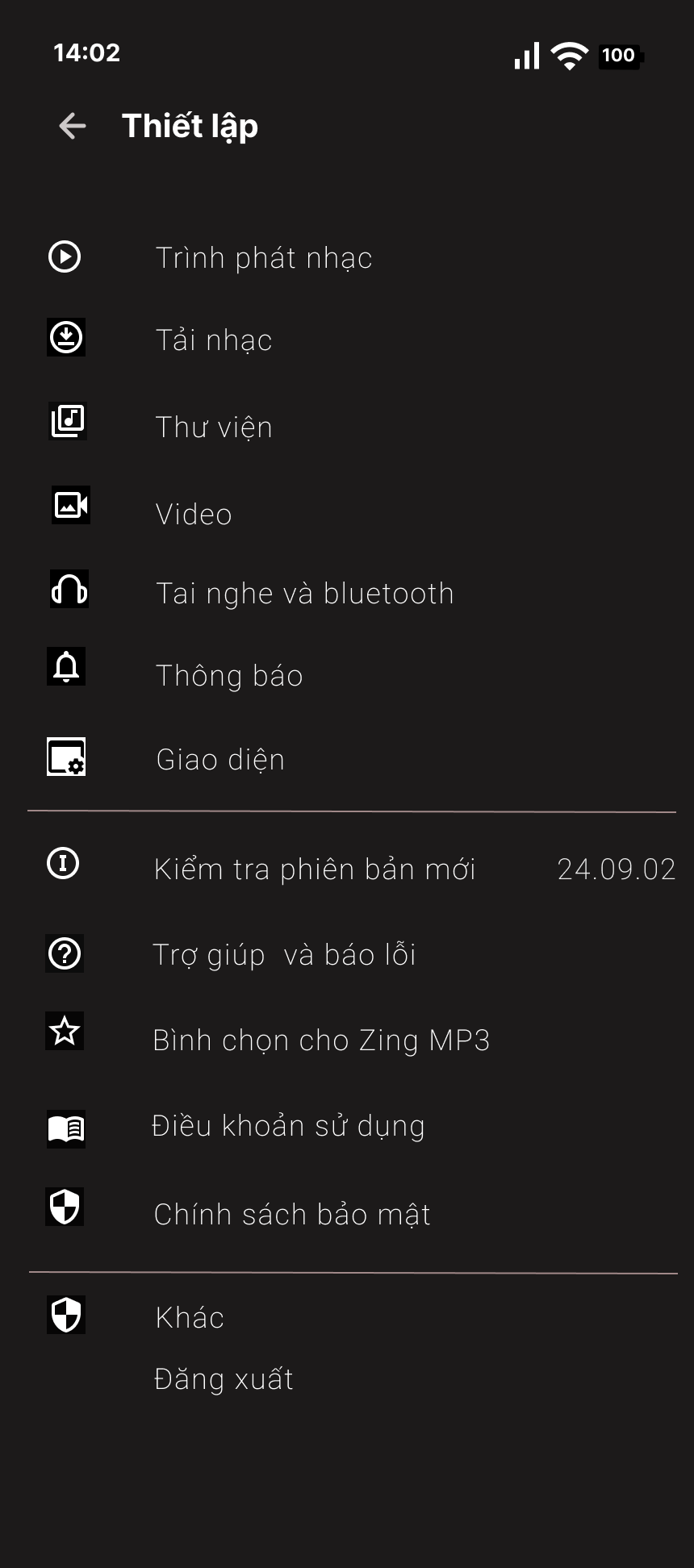
* Giao diện Hiển thị bài hát:



* Giao diện Tài khoản cá nhân:



* Giao diện Setting:



* 1. **Triển khai:**
* **Viết code:** Sử dụng App [Android Studio] và dùng ngôn ngữ [Java].
* **Kiểm thử:** 
  + **Chức năng Play Song (Phát nhạc)**

|  |  |  |  |
| --- | --- | --- | --- |
| **TestCase ID** | **Mô tả** | **Bước thực hiện** | **Kết quả mong đợi** |
| TC01 | Phát một bài hát từ playlist | 1. Mở ứng dụng  2. Chọn một playlist  3. Chọn một bài hát  4. Nhấn nút "Play" | Bài hát được phát thành công |
| TC02 | |  | | --- | |  |   Tạm dừng bài hát | 1. Chơi một bài hát  2. Nhấn nút "Pause" | Bài hát tạm dừng |
| TC03 | Phát bài hát tiếp theo | 1. Phát một bài hát  2. Nhấn nút "Next" | Bài hát tiếp theo trong playlist được phát |
| TC04 | Phát lại bài hát trước đó | 1. Phát một bài hát  2. Nhấn nút "Previous" | Bài hát trước đó được phát |
| TC05 | Điều chỉnh âm lượng | 1. Phát một bài hát  2. Điều chỉnh âm lượng qua thanh trượt | Âm lượng của bài hát thay đổi theo thiết lập |

* **Chức năng Quản lý Playlist**

|  |  |  |  |
| --- | --- | --- | --- |
| **TestCase ID** | **Mô tả** | **Bước thực hiện** | **Kết quả mong đợi** |
| TC06 | Tạo playlist mới | 1. Nhấn vào nút "Create Playlist"  2. Nhập tên và mô tả playlist  3. Nhấn "Save" | Bài hát được phát thành công |
| TC07 | |  | | --- | |  |   Thêm bài hát vào playlist | 1. Mở một playlist  2. Nhấn "Add Song"  3. Chọn bài hát từ danh sách bài hát  4. Nhấn "Add" | Bài hát được thêm vào playlist thành công |
| TC08 | Xóa bài hát khỏi playlist | 1. Mở một playlist  2. Chọn bài hát cần xóa  3. Nhấn "Remove" | Bài hát được xóa khỏi playlist thành công |
| TC09 | Chia sẻ playlist | 1. Mở playlist  2. Nhấn nút "Share"  3. Chọn phương thức chia sẻ (qua email, social media, v.v.) | Playlist được chia sẻ thành công |

* **Chức năng tìm kiếm bài hát**

|  |  |  |  |
| --- | --- | --- | --- |
| **TestCase ID** | **Mô tả** | **Bước thực hiện** | **Kết quả mong đợi** |
| TC10 | Tìm kiếm bài hát theo tên | 1. Nhập tên bài hát vào thanh tìm kiếm  2. Nhấn nút "Search" | Danh sách các bài hát phù hợp với từ khóa tìm kiếm xuất hiện |
| TC11 | |  | | --- | |  |   Tìm kiếm bài hát không tồn tại | 1. Nhập tên một bài hát không có trong danh sách  2. Nhấn nút "Search" | Thông báo "Không tìm thấy bài hát" hiển thị |

* **Chức năng Bài hát yêu thích**

|  |  |  |  |
| --- | --- | --- | --- |
| **TestCase ID** | **Mô tả** | **Bước thực hiện** | **Kết quả mong đợi** |
| TC12 | Thích một bài hát | 1. Phát một bài hát  2. Nhấn vào biểu tượng "Thích" (Like) | Biểu tượng "Thích" được hiển thị và bài hát được thêm vào danh sách yêu thích |
| TC13 | |  | | --- | |  |   Bỏ thích một bài hát | 1. Nhấn vào biểu tượng "Thích" (Like) trên bài hát đã thích trước đó | Biểu tượng "Thích" bị bỏ và bài hát được xóa khỏi danh sách yêu thích |

* **Chức năng thông báo (Notification)**

|  |  |  |  |
| --- | --- | --- | --- |
| **TestCase ID** | **Mô tả** | **Bước thực hiện** | **Kết quả mong đợi** |
| TC14 | Nhận thông báo về bài hát mới | 1. Mở ứng dụng  2. Thêm bài hát mới vào ứng dụng  3. Kiểm tra thông báo trên thanh trạng thái | Người dùng nhận được thông báo về bài hát mới được thêm vào |
| TC15 | |  | | --- | |  |   Nhận thông báo khi playlist được chia sẻ | 1. Chia sẻ playlist với một người dùng khác  2. Kiểm tra thông báo trên thanh trạng thái | Người dùng nhận được thông báo về playlist đã được chia sẻ |

* **Chức năng quản lý bài hát (Song Management)**

|  |  |  |  |
| --- | --- | --- | --- |
| **TestCase ID** | **Mô tả** | **Bước thực hiện** | **Kết quả mong đợi** |
| TC19 | Thêm bài hát mới | 1. Mở màn hình quản lý bài hát  2. Nhấn "Thêm bài hát"  3. Nhập thông tin bài hát (tên, lời bài hát, ảnh bìa)  4. Nhấn "Lưu" | Bài hát mới được thêm thành công vào ứng dụng |
| TC20 | |  | | --- | |  |   Xóa bài hát khỏi ứng dụng | 1. Mở danh sách bài hát  2. Chọn bài hát cần xóa  3. Nhấn "Xóa" | Bài hát được xóa khỏi ứng dụng thành công |

* 1. **Vận hành và bảo trì:**
* Cài đặt và triển khai:
  + Kết nối thiết bị Android và cài đặt trực tiếp từ Android Studio, hoặc tạo file APK và cài đặt thủ công.
  + Triển khai lên Google Play Store: Chuẩn bị Release APK, đăng ký tài khoản Google Play Developer, tải ứng dụng lên Google Play Console, và đợi xét duyệt.
* Bảo trì: Sửa lỗi phát sinh, cập nhật chức năng mới (nếu có) và cải thiện hiệu năng của ứng dụng.

# **CHƯƠNG 3: KẾT QUẢ THỰC HIỆN**

* 1. **Công nghệ đã sử dụng**
* **Ngôn ngữ lập trình:** Java, XML
* **Công cụ:** Android Studio, SQLite
* **Thư viện:** MediaPlayer
  1. **Tiến độ thực hiện**

### Link github tới dự án: [https://github.com/lich1101/CSE441\_PROJECT.git]

# **KẾT LUẬN**

Ứng dụng Music App mang lại nhiều **ưu điểm** vượt trội trong việc quản lý và nghe nhạc. Người dùng có thể tạo và quản lý các playlist của riêng mình, giúp cá nhân hóa trải nghiệm nghe nhạc theo sở thích. Giao diện trực quan, dễ sử dụng kết hợp với các chức năng như phát nhạc, tìm kiếm, và chia sẻ playlist giúp người dùng tận hưởng âm nhạc một cách tiện lợi. Bên cạnh đó, tính năng thông báo cũng giúp người dùng luôn cập nhật được những bài hát mới hoặc playlist được chia sẻ. Ứng dụng có thể hoạt động tốt trên nhiều thiết bị di động, mang đến trải nghiệm mượt mà.

Tuy nhiên, Music App cũng tồn tại một số **nhược điểm** cần cải thiện. Các chức năng như phát nhạc có thể tiêu tốn nhiều tài nguyên, gây ảnh hưởng đến hiệu suất trên các thiết bị cấu hình thấp. Ngoài ra, việc quản lý dữ liệu nhạc lớn có thể làm tăng dung lượng lưu trữ cần thiết, gây khó khăn cho người dùng có không gian bộ nhớ hạn chế. Một vấn đề khác là khả năng đồng bộ dữ liệu giữa các thiết bị chưa được tối ưu, khiến người dùng gặp rắc rối khi chuyển đổi giữa nhiều thiết bị khác nhau.

Trong tương lai, để **phát triển** và nâng cao trải nghiệm người dùng, ứng dụng có thể tích hợp các công nghệ hiện đại như AI để gợi ý bài hát dựa trên sở thích nghe nhạc của từng người. Thêm vào đó, tính năng đồng bộ đám mây cho phép lưu trữ playlist và bài hát trên các dịch vụ như Google Drive hoặc iCloud sẽ giúp người dùng truy cập và quản lý nhạc từ nhiều thiết bị dễ dàng hơn. Cuối cùng, ứng dụng có thể mở rộng sang việc hỗ trợ streaming nhạc trực tuyến, kết hợp với các nền tảng âm nhạc lớn để mang đến nhiều nội dung hơn cho người dùng.

# **PHỤ LỤC**

* **Code minh họa đầy đủ đã thực hiện:**
  + **MainActivity**

|  |
| --- |
| public class MainActivity extends AppCompatActivity {  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main*);   Button btn\_upload = findViewById(R.id.*btn\_upload*);  btn\_upload.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View view) {  Intent intent = new Intent(MainActivity.this, UploadActivity.class);  startActivity(intent);  }  });  } } |

* + **AudioModel**

|  |
| --- |
| public class AudioModel implements Serializable {  String path;  String title;  String duration;   public AudioModel(String path, String title, String duration) {  this.path=path;  this.title=title;  this.duration=duration;  }   public String getPath() { return path; }   public void setPath(String path) { this.path = path; }   public String getTitle(){ return title; }   public void setTitle(String title){  this.title = title;  }   public String getDuration(){ return duration; }   public void setDuration(String duration){ this.duration = duration; } } |

* + **MusicAdapter**

|  |
| --- |
| public class MusicAdapter extends RecyclerView.Adapter<MusicAdapter.ViewHolder>{   ArrayList<AudioModel>songsList;  Context context;   public MusicAdapter(ArrayList<AudioModel> songsList, Context context){  this.songsList = songsList;  this.context = context;  }   @Override  public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  View view = LayoutInflater.*from*(context).inflate(R.layout.*recycler\_item*,parent,false);  return new ViewHolder(view);  }   @Override  public void onBindViewHolder(ViewHolder holder , int position) {  AudioModel SongData = songsList.get(position);  holder.getAdapterPosition();  holder.titleTextView.setText(SongData.getTitle());    holder.itemView.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  MyMediaPlayer.*getInstance*().reset();  MyMediaPlayer.*currentIndex* = holder.getAdapterPosition();  Intent intent = new Intent(context, MusicPlayerActivity.class);  intent.putExtra("LIST",songsList);  intent.setFlags(Intent.*FLAG\_ACTIVITY\_NEW\_TASK*);  context.startActivity(intent);  }  });  }    @Override  public int getItemCount() {   return songsList.size();  }   public class ViewHolder extends RecyclerView.ViewHolder{   TextView titleTextView;  ImageView iconView;   public ViewHolder( View itemView) {  super(itemView);  titleTextView = itemView.findViewById(R.id.*title\_view*);  iconView = itemView.findViewById(R.id.*icon\_view*);  }  } } |

* + **MusicPlayerActivity**

|  |
| --- |
| public class MusicPlayerActivity extends AppCompatActivity{   TextView titleTv,currentTv,totalTv;  SeekBar seekBar;  ImageView pause,play,pause\_play,music\_Icon;  ArrayList<AudioModel> songsList;  AudioModel currentSong;  MediaPlayer mediaPlayer = MyMediaPlayer.*getInstance*();   int position = 0;  boolean isPlaying = false;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_music\_player*);   titleTv = findViewById(R.id.*Songtitle*);  currentTv = findViewById(R.id.*current\_time*);  totalTv = findViewById(R.id.*total\_time*);  seekBar = findViewById(R.id.*seek\_bar*);  pause = findViewById(R.id.*previous*);  play = findViewById(R.id.*next*);  pause\_play = findViewById(R.id.*pause\_play*);  music\_Icon = findViewById(R.id.*music\_image*);   titleTv.setSelected(true);   songsList = (ArrayList<AudioModel>) getIntent().getSerializableExtra("LIST");  setResourceWithMusic();   MusicPlayerActivity.this.runOnUiThread(new Runnable() {  @Override  public void run() {  if (mediaPlayer != null) {  seekBar.setProgress(mediaPlayer.getCurrentPosition());  currentTv.setText(*convertToMMSS*(mediaPlayer.getCurrentPosition() + ""));   if (mediaPlayer.isPlaying()) {  pause\_play.setImageResource(R.drawable.*ic\_baseline\_pause\_circle\_filled\_24*);  } else {  pause\_play.setImageResource(R.drawable.*ic\_baseline\_play\_circle\_filled\_24*);  }  }  new Handler(Looper.*getMainLooper*()).postDelayed(this, 100);   }  });  pause\_play.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  if(mediaPlayer.isPlaying()){  pausePlay();  }else{  pausePlay();  }  }  });   seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  @Override  public void onProgressChanged(SeekBar seekBar, int progress, boolean FromUser) {  if (mediaPlayer != null && FromUser) {  mediaPlayer.seekTo(progress);  }  }   @Override  public void onStartTrackingTouch(SeekBar seekBar) {   }   @Override  public void onStopTrackingTouch(SeekBar seekBar) {   }  });  }   void setResourceWithMusic(){  currentSong = songsList.get(MyMediaPlayer.*currentIndex*);  titleTv.setText(currentSong.getTitle());  totalTv.setText(*convertToMMSS*(currentSong.getDuration()));  pause\_play.setOnClickListener(v-> pausePlay());  play.setOnClickListener(v-> playNextSong());  pause.setOnClickListener(v-> playPreviousSong());   if(!mediaPlayer.isPlaying()) {  playMusic();  }  }  public void playMusic(){  mediaPlayer.reset();  try {  mediaPlayer.setDataSource(currentSong.getPath());  mediaPlayer.prepare();  mediaPlayer.start();  seekBar.setProgress(0);  seekBar.setMax(mediaPlayer.getDuration());  isPlaying = true;   setAlbumArt(currentSong.getPath());  startAnimation();  }  catch (IOException e){  e.printStackTrace();  }  }  public void playNextSong(){  if (MyMediaPlayer.*currentIndex* == songsList.size()-1)  return;   MyMediaPlayer.*currentIndex* +=1;  mediaPlayer.reset();  setResourceWithMusic();   }  public void playPreviousSong(){  if (MyMediaPlayer.*currentIndex*==0)  return;  MyMediaPlayer.*currentIndex* -=1;  mediaPlayer.reset();  setResourceWithMusic();   }  public void pausePlay(){  if (mediaPlayer.isPlaying()){  mediaPlayer.pause();  isPlaying = false;  pause\_play.setImageResource(R.drawable.*ic\_baseline\_play\_circle\_filled\_24*);  music\_Icon.clearAnimation();  }else{  mediaPlayer.start();  isPlaying = true;  pause\_play.setImageResource(R.drawable.*ic\_baseline\_pause\_circle\_filled\_24*);  startAnimation();  }  }   @SuppressLint("DefaultLocale")  public static String convertToMMSS(String Duration) {  long millis = Long.*parseLong*(Duration);   return String.*format*("%02d:%02d",  TimeUnit.*MILLISECONDS*.toMinutes(millis) % TimeUnit.*HOURS*.toMinutes(1),  TimeUnit.*MILLISECONDS*.toSeconds(millis) % TimeUnit.*MINUTES*.toSeconds(1));  }   private void startAnimation() {  RotateAnimation rotateAnimation = new RotateAnimation(  0f, 360f,  RotateAnimation.*RELATIVE\_TO\_SELF*, 0.5f,  RotateAnimation.*RELATIVE\_TO\_SELF*, 0.5f  );  rotateAnimation.setDuration(5000);  rotateAnimation.setRepeatCount(Animation.*INFINITE*);  rotateAnimation.setInterpolator(new LinearInterpolator());  music\_Icon.startAnimation(rotateAnimation);  }   private void setAlbumArt(String filepath) throws IOException {  MediaMetadataRetriever retriever = new MediaMetadataRetriever();  retriever.setDataSource(filepath);  byte[] art = retriever.getEmbeddedPicture();  if (art != null) {  music\_Icon.setImageBitmap(BitmapFactory.*decodeByteArray*(art, 0, art.length));  } else {  music\_Icon.setImageResource(R.drawable.*musicicon*);  }   retriever.release();  }  } |

* + **MyMediaPlayer**

|  |
| --- |
| public class MyMediaPlayer {  static MediaPlayer *instance*;   public static MediaPlayer getInstance(){  if(*instance* == null){  *instance* = new MediaPlayer();  }  return *instance*;  }  public static int *currentIndex* = -1; } |

* + **SplashActivity**

|  |
| --- |
| public class SplashActivity extends AppCompatActivity {  @Override  public void onCreate(@Nullable Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_splash*);   new Handler().postDelayed(new Runnable() {  @Override  public void run() {  *// Chuyển sang LoginActivity* Intent intent = new Intent(SplashActivity.this, MainActivity.class);  startActivity(intent);  finish(); *// Đóng SplashActivity để không thể quay lại* }  }, 3000);  } } |

* + **UploadActivity**

|  |
| --- |
| public class UploadActivity extends AppCompatActivity {  RecyclerView recyclerView;  TextView noMusicTextView;   ArrayList<AudioModel> SongsList = new ArrayList<>();   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_upload*);   recyclerView = findViewById(R.id.*recycle\_view*);  noMusicTextView = findViewById(R.id.*No\_Songs*);   if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*TIRAMISU*) {  if (!checkPermission()) {  requestPermission();  return;  }  }  String[] projection = {  MediaStore.Audio.Media.*TITLE*,  MediaStore.Audio.Media.*DATA*,  MediaStore.Audio.Media.*DURATION*,  MediaStore.Audio.Media.*ARTIST*,  MediaStore.Audio.Media.*\_ID*,  MediaStore.Audio.Media.*SIZE* };  String selection = MediaStore.Audio.Media.*IS\_MUSIC* + "!= 0";   Cursor cursor = getContentResolver().query(MediaStore.Audio.Media.*EXTERNAL\_CONTENT\_URI*, projection, selection, null, null  );   while (cursor.moveToNext()) {  AudioModel songData = new AudioModel(cursor.getString(1), cursor.getString(0), cursor.getString(2));  if (new File(songData.getPath()).exists())  SongsList.add(songData);  }  if (SongsList.size() == 0) {  noMusicTextView.setVisibility(View.*VISIBLE*);  } else {  *//recyclerView* recyclerView.setLayoutManager(new LinearLayoutManager(this));  recyclerView.setAdapter(new MusicAdapter(SongsList, getApplicationContext()));  }  }   @RequiresApi(api = Build.VERSION\_CODES.*TIRAMISU*)  boolean checkPermission(){  int result = ContextCompat.*checkSelfPermission*(UploadActivity.this, android.Manifest.permission.*READ\_MEDIA\_AUDIO*);  if(result == PackageManager.*PERMISSION\_GRANTED*){  return true;  }else {  return false;  }  }  @RequiresApi(api = Build.VERSION\_CODES.*TIRAMISU*)  void requestPermission(){  if(ActivityCompat.*shouldShowRequestPermissionRationale*(UploadActivity.this, android.Manifest.permission.*READ\_MEDIA\_AUDIO*)){  Toast.*makeText*(UploadActivity.this,"PERMISSION NEEDED",Toast.*LENGTH\_SHORT*).show();  }else  ActivityCompat.*requestPermissions*(UploadActivity.this,new String[]{Manifest.permission.*READ\_MEDIA\_AUDIO*},200);  } } |