TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU TẠI TP. HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI :QUẢN LÝ SỔ TIẾT KIỆM

Giảng viên hướng dẫn: ThS.TRẦN PHONG NHÃ

Sinh viên thực hiện: Lê Công Thuận

Nguyễn Ngô Thành Tài

Trịnh Nguyễn Vy Na

Lớp: CQ.62.CNTT

Khóa: 62

Tp. Hồ Chí Minh, năm 2024

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

PHÂN HIỆU TẠI TP. HỒ CHÍ MINH

BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

ĐỀ TÀI : QUẢN LÝ SỔ TIẾT KIỆN

Giảng viên hướng dẫn: ThS.TRẦN PHONG NHÃ

Sinh viên thực hiện: Lê Công Thuận - 6251071100

Trịnh Nguyễn Vy Na - 6251071001

Nguyễn Ngô Thành Tài - 6251071003

Lớp: CQ.62.CNTT

Khóa: 62

Tp. Hồ Chí Minh, năm 2024

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

PHÂN HIỆU TẠI TP. HỒ CHÍ MINH Độc lập - Tự do - Hạnh phúc

# NHIỆM VỤ THIẾT KẾ BÀI TẬP LỚN

BỘ MÔN: **CÔNG NGHỆ THÔNG TIN**

-------\*\*\*-------

**1. Đề tài.**

Thiết kế và xây dựng chương trình nghe nhạc trên điện thoại

**2. Mục đích, yêu cầu.**

1. Mục đích.

- Giúp người dùng có thể nghe nhạc, tạo playlist và upload lên cloud

1. Yêu cầu.

- Giao diện: Thân thiện, dễ sử dụng, cấu hình nhẹ.

- Tổ chức dữ liệu rõ ràng, nhất quán

**3. Nội dung và phạm vi đề tài.**

1. Nội dung đề tài.

- Tổng quan về bài toán

- Tổng quan về ngôn ngữ lập trình Java

- Tổng quan về Firestore

- Sử dụng Android Studio để thiết kế chương trình

- Áp dụng Firestore theo cấu trúc NoSQL để xây dựng database

1. Phạm vi đề tài.

- Tạo ra chương trình ứng dụng di động để phát nhạc cho người dùng.

**4. Công nghệ, công cụ và ngôn ngữ lập trình.**

1. Công nghệ: Github.
2. Công cụ: Android Studio, Firestore.
3. Ngôn ngữ lập trình: Java.

**5. Các kết quả chính dự kiến sẽ đạt được và ứng dụng.**

Quyển báo cáo đề tài bài tập lớn được chia thành 4 phần như sau:

-  **Chương 1**: Tổng quan về bài toán

+ Tổng quan về đề tài.

+ Mục tiêu nghiên cứu.

+ Đối tượng nghiên cứu

+ Phạm vi.

- **Chương 2**: Cơ sở lí thuyết.

+ Tổng quan về ngôn ngữ lập trình Java.

+ Tổng quan về Firestore và NoSQL.

+ Tổng quan về Android Studio.

- **Chương 3**: Phân tích bài toán

+ Mô tả bài toán

+ Sơ đồ phân rã chức năng

+ Sơ đồ Usecase

+ Sơ đồ hoạt động - activity diagram

+ Sơ đồ tuần tự

- **Chương 4**: Thiết kế và cài đặt chương trình

+ Thiết kế và xây dựng hệ thống.

+ Thiết kế giao diện

+ Code chương trình

* **Chương 5**: Kết luận và hướng phát triển:

+ Kết quả thực hiện được

+ Đề xuất phát triển

**6. Giáo viên và cán bộ hướng dẫn.**

Họ tên: Phan Thanh Hy

Đơn vị công tác: Bộ môn Công Nghệ Thông Tin – Trường Đại học Giao thông Vận tải phân hiệu tại TP HCM

Điện thoại: Email

# 

# LỜI CẢM ƠN

Lời đầu tiên em xin phép gửi đến Quý Thầy Cô của Bộ môn Công Nghệ Thông Tin trường Đại học Giao thông Vận tải - Phân hiệu tại Tp Hồ Chí Minh lời chúc sức khỏe và lời cảm ơn sâu sắc nhất. Thầy cô đã tận tình hướng dẫn, truyền đạt kiến thức, kinh nghiệm cho chúng em trong suốt quá trình học tập và rèn luyện ở trường, chúc thầy cô luôn mạnh khỏe và thu được nhiều thành công hơn nữa trong tương lai.

Đặc biệt, em xin gửi lời cảm ơn đến thầy ThS.Phan Thanh Hy, thầy đã tận tình hướng dẫn chỉ bảo để em có thể tiếp cận với nhiều bài toán thực tế để hoàn thành bài tập lớn ‘Ứng dụng nghe nhạc trên điện thoại’.

Mặc dù nhóm em đã cố gắng hết sức trong quá trình thực hiện bài tập lớn này, nhưng vì thời gian có hạn và nhóm em cũng chưa có kinh nghiệm làm việc với bài toán thực tế, nên chắc chắn sẽ có nhiều thiếu sót.

Lời sau cùng, em xin kính chúc thầy cô trong bộ môn Công Nghệ Thông Tin trường Đại học Giao thông Vận tải - Phân hiệu tại Tp Hồ Chí Minh luôn mạnh khỏe và thu được nhiều thành công hơn nữa trong tương lai.

Em xin chân thành cảm ơn!

# 

# NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Tp. Hồ Chí Minh, ngày ... tháng ... năm 2024

Giảng viên hướng dẫn

# MỤC LỤC

[NHIỆM VỤ THIẾT KẾ BÀI TẬP LỚN 1](#_gjdgxs)

[LỜI CẢM ƠN 3](#_30j0zll)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 4](#_1fob9te)

[MỤC LỤC 5](#_3znysh7)

[DANH MỤC HÌNH ẢNH 7](#_7phoau1sjagi)

[DANH MỤC THUẬT NGỮ 8](#_tyjcwt)

[CHƯƠNG 1.MỞ ĐẦU 1](#_3dy6vkm)

[1.1 Tổng quan về đề tài. 1](#_1t3h5sf)

[1.2. Mục tiêu nghiên cứu. 1](#_4d34og8)

[1.3 Đối tượng nghiên cứu 1](#_r1mlb6c75c4h)

[1.4. Phạm vi. 1](#_2s8eyo1)

[CHƯƠNG 2: CƠ SỞ LÍ THUYẾT 2](#_17dp8vu)

[2.1 Tổng quan về ngôn ngữ lập trình Java: 2](#_3rdcrjn)

[2.1.2 Ưu điểm: 2](#_lnxbz9)

[2.1.3 Nhược điểm 2](#_35nkun2)

[2.2 Tổng quan về Firestore và NoSQL. 2](#_1ksv4uv)

[2.2.1 Khái niệm: 2](#_44sinio)

[2.2.2 Ưu điểm: 3](#_2jxsxqh)

[2.2.3 Nhược điểm 3](#_d789bm9erui9)

[2.3. Tổng quan về Android Studio: 4](#_3j2qqm3)

[2.3.1 Khái niệm: 4](#_1y810tw)

[2.3.2 Ưu điểm: 4](#_4i7ojhp)

[2.3.3 Nhược điểm: 4](#_q6ofnsghrleo)

[CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG 5](#_1ci93xb)

[3.1. Mô tả bài toán: 5](#_3whwml4)

[3.2. Sơ đồ phân rã chức năng 5](#_2bn6wsx)

[3.3. Sơ đồ Usecase 6](#_4y3do7eljk0v)

[3.3.1. Sơ đồ Usecase tổng quát 6](#_9tij6kjxayu9)

[3.3.2 Sơ đồ Usecase đăng nhập 7](#_8edul7kmlf7s)

[3.3.3 Sơ đồ Usecase nghe nhạc 8](#_vxsw2y3at2fv)

[3.3.4 Sơ đồ Usecase lưu vào playlist 9](#_q5adwelmhsgy)

[3.4.Sơ đồ hoạt động - AD 10](#_3as4poj)

[3.4.1. Sơ đồ hoạt động miêu tả người dùng đăng nhập bằng tài khoản 10](#_1pxezwc)

[3.4.2. Sơ đồ hoạt động miêu tả người dùng nghe nhạc offline 11](#_49x2ik5)

[3.4.3. Sơ đồ hoạt động miêu tả người dùng thêm bài hát vào playlist 12](#_2p2csry)

[3.4.4. Sơ đồ hoạt động miêu tả người dùng xóa bài hát khỏi playlist 13](#_4bmb5a67nhcd)

[3.4.5. Sơ đồ hoạt động miêu tả người dùng xóa playlist: 14](#_u8eo2kxv8fzg)

[3.4.6. Sơ đồ hoạt động miêu tả người dùng nghe nhạc online: 15](#_ckgkc9118k7e)

[3.5. Sơ đồ tuần tự 16](#_djansxlvib99)

[3.5.1. Sơ đồ tuần tự đăng nhập 16](#_iago1awxrfvs)

[3.5.2. Sơ đồ tuần tự nghe nhạc 17](#_75s9dkod5v6s)

[3.5.3. Sơ đồ tuần tự thêm bài hát vào playlist 18](#_enu432igmwbu)

[3.5.4. Sơ đồ tuần tự xóa bài hát khỏi playlist 19](#_zhtbwm9cy6mf)

[3.5.5. Sơ đồ tuần tự xóa playlist 19](#_gtvv2yasnri3)

[3.5.6. Sơ đồ tuần tự chức năng tìm kiếm 20](#_hol6c8lbmfam)

[3.6.Class Diagram: 21](#_92b1grqwmbtu)

[CHƯƠNG 4: THIẾT KẾ HỆ THỐNG 22](#_147n2zr)

[4.1. Kiến trúc hệ thống 22](#_3o7alnk)

[4.2.Thiết kế giao diện: 22](#_2grqrue)

[4.2.1. Giao diện khi mở chương trình: 23](#_nxg0mzs69wma)

[4.2.2. Giao diện khi nghe nhạc offline 24](#_14eeokhd779o)

[4.2.3. Giao diện đăng nhập: 25](#_gvgtdnhz6s2)

[4.2.4:Giao diện khi nhấn vào bài hát để phát nhạc offline: 26](#_yx4vmabll2j)

[4.2.5: Giao diện Playlists: 27](#_5ccrv92jikls)

[4.2.6: Giao diện hiển thị danh sách bài hát trong playlist: 28](#_q8luqd36m096)

[4.2.7: Giao diện khi người dùng muốn xóa bài hát ra khỏi playlist: 29](#_2di3qmjskj6s)

[4.2.8: Giao diện người dùng khi nhấn vào 1 trong các playlist online: 30](#_qgarmji4k5qj)

[4.2.9: Giao diện người dùng khi nghe nhạc online: 31](#_rlhi0rye0yu5)

[4.3 Code API 32](#_9pglowjx5g0r)

[4.3.1. Spotify Online 32](#_h74uixt8up7p)

[4.3.1.1. Thực hiện gọi API các bài hát Top 50 trên Spotify 32](#_qo9aw7vr8lyf)

[4.3.1.2. Thực hiện việc thêm bài hát vào mục yêu thích 33](#_gylkc7cug0ir)

[4.3.1.3. Thực hiện việc tìm kiếm bài hát 34](#_n927o8y82ebv)

[4.3.1.4. Thực hiện xử lý giữa người dùng và ứng dụng 37](#_52eutldkzf05)

[4.3.2. Nghe nhạc offline có trong máy 48](#_ulbb8v6o76br)

[4.3.2.1. Khởi tạo MusicAdapter 48](#_68hj4kwph9u5)

[4.3.2.2. Thực hiện việc lấy nhạc ở thiết bị trong DashboardFragment 51](#_q0x9qcpxvxq2)

[4.3.3. Đăng nhập vào tài khoản google 54](#_19qcyoicnanx)

[4.3.4. Chức năng load bài hát lên cloud(Storage of Firestore) 58](#_ro4840s4cl99)

[4.3.5. Tải bài hát có trên cloud và xóa bài hát khỏi cloud 59](#_bga048eljdkq)

[KẾT LUẬN VÀ KIẾN NGHỊ 61](#_3fwokq0)

[TÀI LIỆU THAM KHẢO 63](#_1v1yuxt)

# DANH MỤC HÌNH ẢNH

[3.2.1.Sơ đồ phân rã chức năng 6](#_7u5dqhztozs)

[3.3.1.Sơ đồ UC tổng quát 6](#_8g2oyw6bsvwg)

[3.3.2.Sơ đồ UC đăng nhập 7](#_e77w58xy0eqh)

[3.3.3.Sơ đồ UC nghe nhạc 8](#_1xf1oujex0py)

[3.3.4.Sơ đồ UC lưu vào playlist 9](#_swd9vpe7q3xw)

[3.4.1. Sơ đồ hoạt động miêu tả người dùng đăng nhập bằng tài khoản 10](#_neub636d1p55)

[3.4.2 Sơ đồ hoạt động miêu tả người dùng nghe nhạc offline 11](#_t0orps313x1q)

[3.4.3. Sơ đồ hoạt động miêu tả người dùng thêm bài hát vào playlist 12](#_nz8hu3oc5d9i)

[3.4.4. Sơ đồ hoạt động miêu tả người dùng xóa bài hát khỏi playlist 13](#_1mdxjpliqq2h)

[3.4.5 Sơ đồ hoạt động miêu tả người dùng xóa playlist 14](#_9n5arnm7j8fh)

[3.4.6 Sơ đồ hoạt động miêu tả người dùng nghe nhạc online 15](#_wz3mrqa9clfq)

[3.5.1. Sơ đồ tuần tự đăng nhập 16](#_bvhw0bmdqb92)

[3.5.2. Sơ đồ tuần tự nghe nhạc offline 17](#_e3cqswydj8vs)

[3.5.3. Sơ đồ tuần tự thêm bài hát vào playlist 18](#_5qwr7cadfzqv)

[3.5.4. Sơ đồ tuần tự xóa bài hát khỏi playlist 19](#_naqp4dbtix4l)

[3.5.5. Sơ đồ tuần tự xóa playlist 19](#_7b8fobk2ohnw)

[3.5.6. Sơ đồ tuần tự chức năng tìm kiếm 20](#_8uy4j68wblz4)

[3.6. Class Diagram 21](#_mxv6wqsxopkr)

[4.2.1. Giao diện khi mở chương trình 23](#_8sl04swt9zha)

[4.2.2. Giao diện khi nghe nhạc offline 24](#_4g340ucs089n)

[4.2.3. Giao diện đăng nhập: 25](#_w614u3r9nbn6)

[4.2.4:Giao diện khi nhấn vào bài hát để phát nhạc offline 26](#_hw2lbjcszwyc)

[4.2.5: Giao diện Playlists 27](#_bu2igvv3n96r)

[4.2.6: Giao diện hiển thị danh sách bài hát trong playlist 28](#_r8hcl3qmg4uv)

[4.2.7: Giao diện khi người dùng muốn xóa bài hát ra khỏi playlist 29](#_i5oewuwn137d)

[4.2.8: Giao diện người dùng khi nhấn vào 1 trong các playlist online 30](#_yqgq91vnkq1i)

[4.2.9: Giao diện người dùng khi nghe nhạc online 31](#_gthwi0ym0les)

# 

# DANH MỤC THUẬT NGỮ

| STT | Mô tả | Ý nghĩa |
| --- | --- | --- |
| 1 | API | Application Programming Interface |
| 2 | SQL | Structured Query Language |
| 3 | NoSQL | Not Only SQL |
| 4 | UC | Uses case |
| 5 | AD | Activity diagram |

# CHƯƠNG 1.MỞ ĐẦU

## 1.1 Tổng quan về đề tài.

Trong thời đại số hóa hiện nay, việc sử dụng ứng dụng nghe nhạc trên điện thoại đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày. Theo các nghiên cứu, khoảng 95% người dùng smartphone sử dụng các ứng dụng nghe nhạc, và xu hướng này vẫn đang tiếp tục tăng lên mỗi ngày.

## 1.2. Mục tiêu nghiên cứu.

- Tổng quan về mobile app.

- Nghiên cứu về ngôn ngữ lập trình Java.

- Nghiên cứu về Firebase Cloud Firestore

- Phân tích và thiết kế: Giúp cho việc xây dựng và thực hiện trở nên dễ dàng và rõ ràng hơn.

- Cấu trúc và bố cục về ứng dụng “Ứng dụng nghe nhạc trên điện thoại”:

+ Khởi tạo và hiển thị các giao diện.

+ Lập trình xử lý sự kiện của các hành động.

+ Kết nối với database đã tạo ở Firestore

## 1.3 Đối tượng nghiên cứu

- Người dùng mục tiêu:

+ Độ tuổi: Người trưởng thành có nhu cầu nghe nhạc online và offline

+ Sở thích âm nhạc: Mọi thể loại

+ Thói quan nghe nhạc: bất cứ đâu và khi nào như: khi làm việc, khi tập thể dục, khi lái xe,...

+ Yêu cầu công nghệ: có điện thoại chạy hệ điều hành android

## 1.4. Phạm vi.

Xây dựng app mobile bằng ngôn ngữ Java, sử dụng API, Firestore để lưu trữ dữ liệu để có các chức năng đã được nêu ra trong phần nội dung.

# CHƯƠNG 2: CƠ SỞ LÍ THUYẾT

## 

## 2.1 Tổng quan về ngôn ngữ lập trình Java:

2.1.1 Khái niệm:  
 Java là một ngôn ngữ lập trình hướng đối tượng, đa nền tảng được phát triển bởi Sun Microsystems, hiện nay là một phần của Oracle. Java được thiết kế với mục tiêu “viết một lần, chạy mọi nơi”, có nghĩa là mã Java sau khi được biên dịch có thể chạy trên bất kỳ nền tảng nào hỗ trợ Java mà không cần sửa đổi. Java được sử dụng rộng rãi trong việc phát triển ứng dụng web, ứng dụng di động (đặc biệt là Android), hệ thống doanh nghiệp và nhiều ứng dụng khác.

### 2.1.2 Ưu điểm:

Đầu tiên, tính di động của Java là một điểm mạnh, với khả năng chạy trên nhiều nền tảng khác nhau mà không cần biên dịch lại mã nguồn..Tiếp theo, Java được biết đến với tính bảo mật cao, với hệ thống kiểm soát quyền truy cập vào tài nguyên của hệ thống, giúp ngăn chặn các lỗ hổng bảo mật và tấn công từ hacker. Cuối cùng, Java cũng nổi tiếng với tính đa luồng và hiệu suất cao, cho phép xử lý đa nhiệm một cách linh hoạt và hiệu quả.

### 2.1.3 Nhược điểm

* Hiệu suất chưa được tối ưu hóa tốt.
* Mã nguồn phức tạp

## 2.2 Tổng quan về Firestore và NoSQL.

### 2.2.1 Khái niệm:

Firestore: Firestore, còn được gọi là Cloud Firestore, là một giải pháp cơ sở dữ liệu NoSQL của Google Firebase được thiết kế để cung cấp cho các nhà phát triển một nền tảng linh hoạt và có thể mở rộng để xây dựng các ứng dụng web và di động hiện đại. Firestore cho phép đồng bộ hóa, lưu trữ và truy xuất dữ liệu theo thời gian thực đồng thời cung cấp chức năng mạnh mẽ, bao gồm hỗ trợ ngoại tuyến, tổ chức dữ liệu phân cấp và bộ khả năng truy vấn toàn diện.

NoSQL: NoSQL, viết tắt của “Not Only SQL”, là một loại cơ sở dữ liệu phi quan hệ, mô hình dữ liệu sơ đồ linh hoạt, được thiết kế để lưu trữ và truy xuất dữ liệu theo cách không tuân theo mô hình cơ sở dữ liệu quan hệ truyền thống. Cơ sở dữ liệu NoSQL được công nhận rộng rãi vì khả năng dễ phát triển, chức năng cũng như hiệu năng ở quy mô lớn. Chúng thường được sử dụng trong các ứng dụng cần quy mô lớn, khả năng mở rộng cao và khả năng lưu trữ dữ liệu không cấu trúc hoặc bán cấu trúc.

### 2.2.2 Ưu điểm:

Đầu tiên, Firebase cung cấp một cơ sở hạ tầng mạnh mẽ với các dịch vụ như lưu trữ dữ liệu realtime, authentication, messaging và analytics, giúp nhà phát triển tập trung vào việc xây dựng ứng dụng mà không phải lo lắng về việc triển khai và quản lý hạ tầng.

Thứ hai, Firebase hỗ trợ tích hợp dễ dàng với các công nghệ và framework phổ biến như React, Angular, hoặc Flutter, giúp tăng tốc độ phát triển và giảm thời gian và công sức cần thiết cho việc kết nối và xử lý dữ liệu.

Thứ ba, Firebase cung cấp các công cụ phân tích mạnh mẽ, từ theo dõi hoạt động người dùng đến hiểu rõ hơn về cách người dùng tương tác với ứng dụng.

Cuối cùng, Firebase là một giải pháp đáng tin cậy và linh hoạt, với khả năng mở rộng dễ dàng để đáp ứng nhu cầu của các ứng dụng từ nhỏ đến lớn.

### 2.2.3 Nhược điểm

Một trong những nhược điểm lớn nhất của Firebase là giới hạn về quy mô và sức mạnh của nền tảng. Ngoài ra, Firebase có thể trở nên tốn kém khi phát triển ứng dụng lớn với nhu cầu cao về tính tùy chỉnh và mở rộng. Cuối cùng, việc kháng nạp dữ liệu từ Firebase có thể là một vấn đề đối với một số ứng dụng yêu cầu hiệu suất cao và thời gian phản hồi nhanh chóng.

## 2.3. Tổng quan về Android Studio:

### 2.3.1 Khái niệm:

Android Studio là môi trường phát triển tích hợp chính thức (IDE) cho việc phát triển ứng dụng Android, dựa trên IntelliJ IDEA. Nó cung cấp một hệ thống xây dựng dựa trên Gradle linh hoạt, một máy giả lập nhanh và nhiều tính năng phong phú, cùng một môi trường thống nhất để phát triển cho tất cả các thiết bị Android

### 2.3.2 Ưu điểm:

Android Studio là một môi trường phát triển tích hợp (IDE) mạnh mẽ dành cho việc phát triển ứng dụng di động trên nền tảng Android. Đầu tiên, Android Studio tích hợp sâu vào hệ sinh thái Android, cung cấp cho người dùng các công cụ phát triển mạnh mẽ như Android SDK, các loại máy ảo để kiểm thử ứng dụng, và trình giả lập thiết bị để kiểm tra hiệu suất. Thứ hai, nó hỗ trợ nhiều ngôn ngữ lập trình như Java, Kotlin, và C++, mở ra nhiều lựa chọn cho nhà phát triển. Thứ ba, giao diện người dùng của Android Studio thân thiện và dễ sử dụng, với nhiều tính năng hỗ trợ tự động hoàn thiện mã, gỡ lỗi thông minh và tích hợp quản lý mã nguồn thông qua Git. Cuối cùng, Android Studio liên tục được cập nhật và hỗ trợ bởi cộng đồng phát triển Android rộng lớn.

### 2.3.3 Nhược điểm:

Một trong những vấn đề phổ biến mà người dùng gặp phải là tài nguyên hệ thống yêu cầu cao. Android Studio đòi hỏi một máy tính có cấu hình mạnh để chạy mượt mà, đặc biệt là khi xử lý các dự án lớn. Thêm vào đó, quá trình cài đặt và cấu hình ban đầu của Android Studio cũng có thể phức tạp đối với những người mới bắt đầu. Ngoài ra, một số người dùng cũng gặp phải vấn đề về hiệu suất và ổn định, với các lỗi phát sinh không mong muốn trong quá trình sử dụng.

# CHƯƠNG 3: PHÂN TÍCH THIẾT KẾ HỆ THỐNG

## 3.1. Mô tả bài toán:

Ứng dụng nghe nhạc trên điện thoại dành cho người dùng yêu âm nhạc. Các chức năng của ứng dụng sẽ được thao tác trực tiếp trên điện thoại android.

## 3.2. Sơ đồ phân rã chức năng


3.1.Sơ đồ phân rã chức năng

#### 3.2.1.Sơ đồ phân rã chức năng

## 3.3. Sơ đồ Usecase

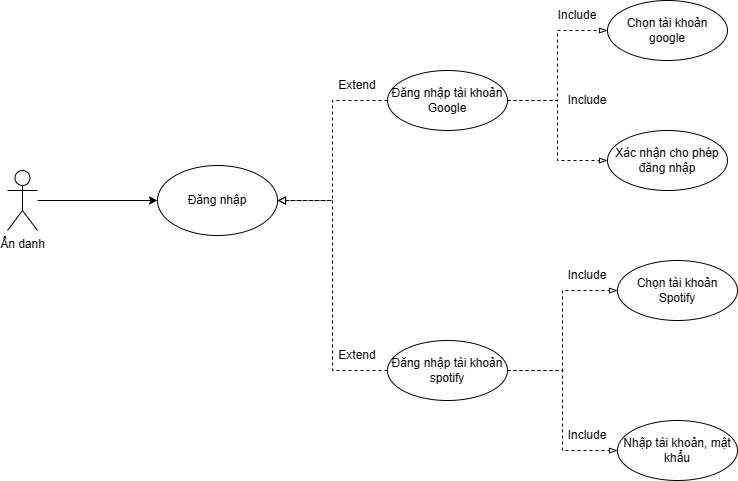
### 3.3.1. Sơ đồ Usecase tổng quát

#### 

#### 3.3.1.Sơ đồ UC tổng quát

#### 

### 3.3.2 Sơ đồ Usecase đăng nhập



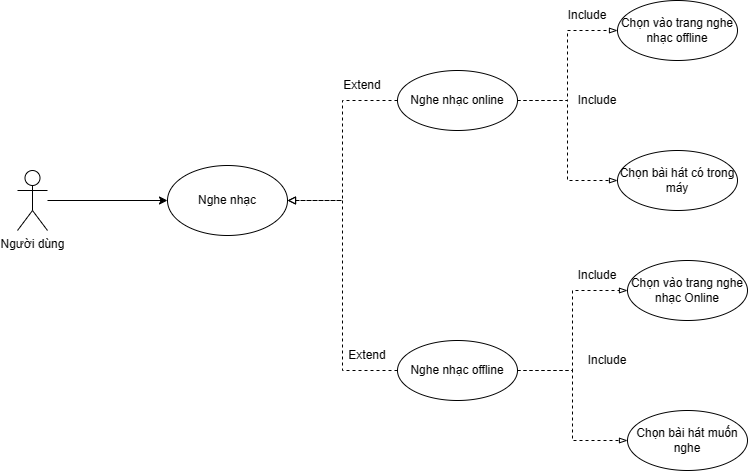
#### 3.3.2.Sơ đồ UC đăng nhập

### 

### 

### 

### 3.3.3 Sơ đồ Usecase nghe nhạc



#### 3.3.3.Sơ đồ UC nghe nhạc

### 

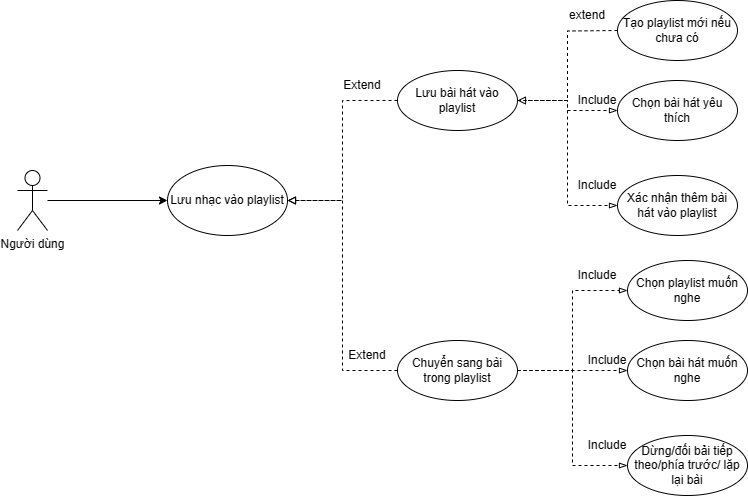
### 

### 

### 

### 

### 3.3.4 Sơ đồ Usecase lưu vào playlist



#### 3.3.4.Sơ đồ UC lưu vào playlist

## 3.4.Sơ đồ hoạt động - AD

### 3.4.1. Sơ đồ hoạt động miêu tả người dùng đăng nhập bằng tài khoản

#### 3.4.1. Sơ đồ hoạt động miêu tả người dùng đăng nhập bằng tài khoản

### 3.4.2. Sơ đồ hoạt động miêu tả người dùng nghe nhạc offline

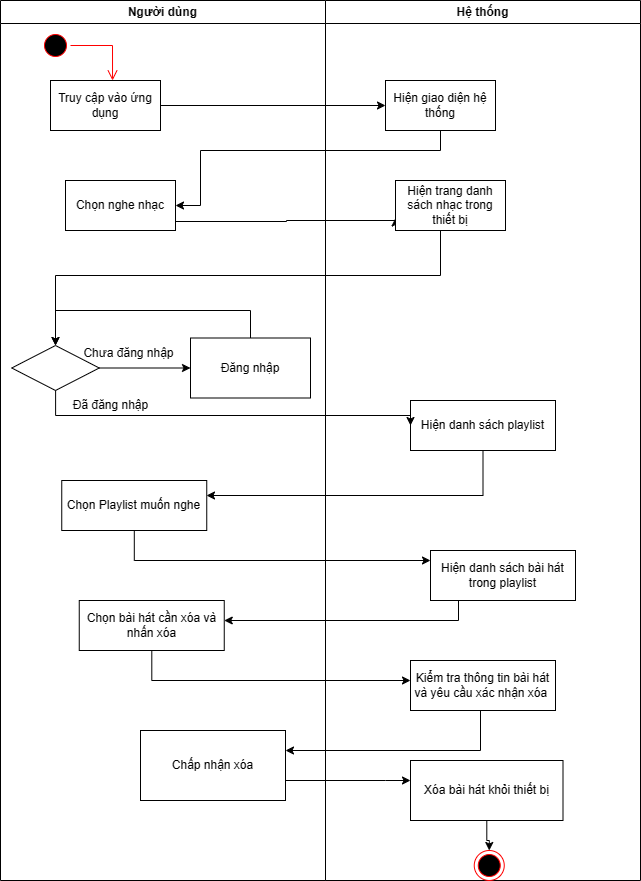
#### 3.4.2 Sơ đồ hoạt động miêu tả người dùng nghe nhạc offline

### 3.4.3. Sơ đồ hoạt động miêu tả người dùng thêm bài hát vào playlist

#### 3.4.3. Sơ đồ hoạt động miêu tả người dùng thêm bài hát vào playlist

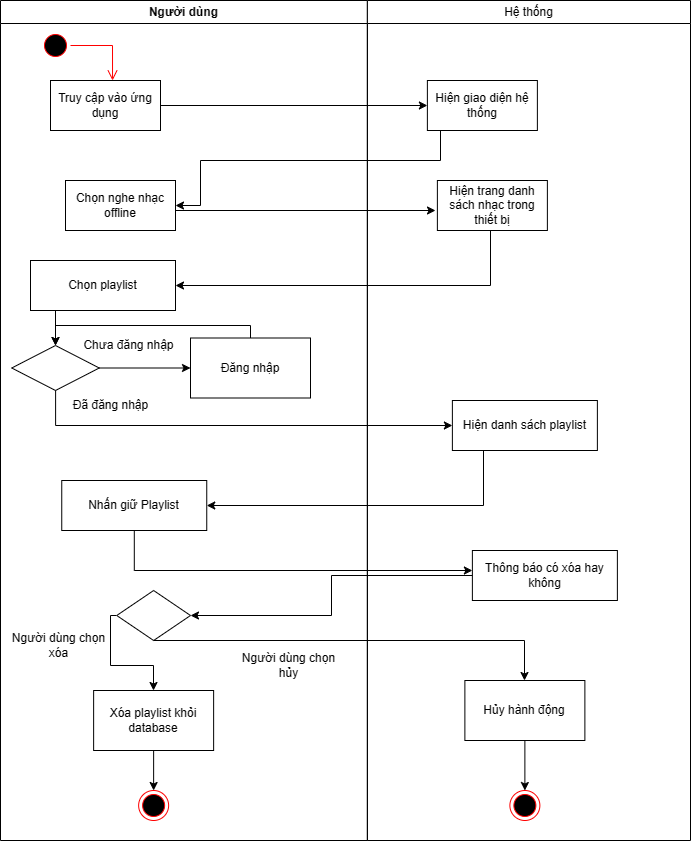
#### 

### 3.4.4. Sơ đồ hoạt động miêu tả người dùng xóa bài hát khỏi playlist



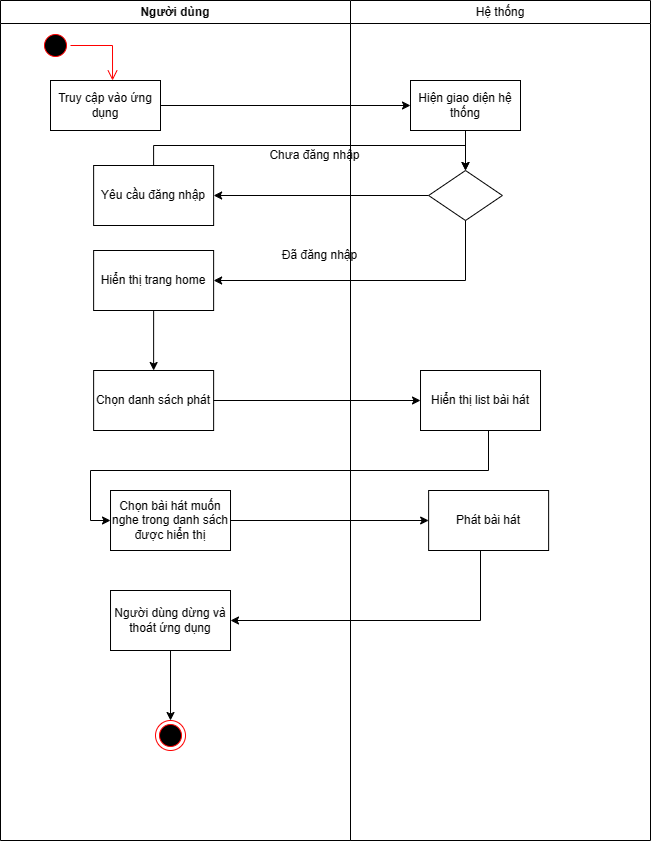
#### 3.4.4. Sơ đồ hoạt động miêu tả người dùng xóa bài hát khỏi playlist

### 3.4.5. Sơ đồ hoạt động miêu tả người dùng xóa playlist:



#### 3.4.5 Sơ đồ hoạt động miêu tả người dùng xóa playlist

### 3.4.6. Sơ đồ hoạt động miêu tả người dùng nghe nhạc online:

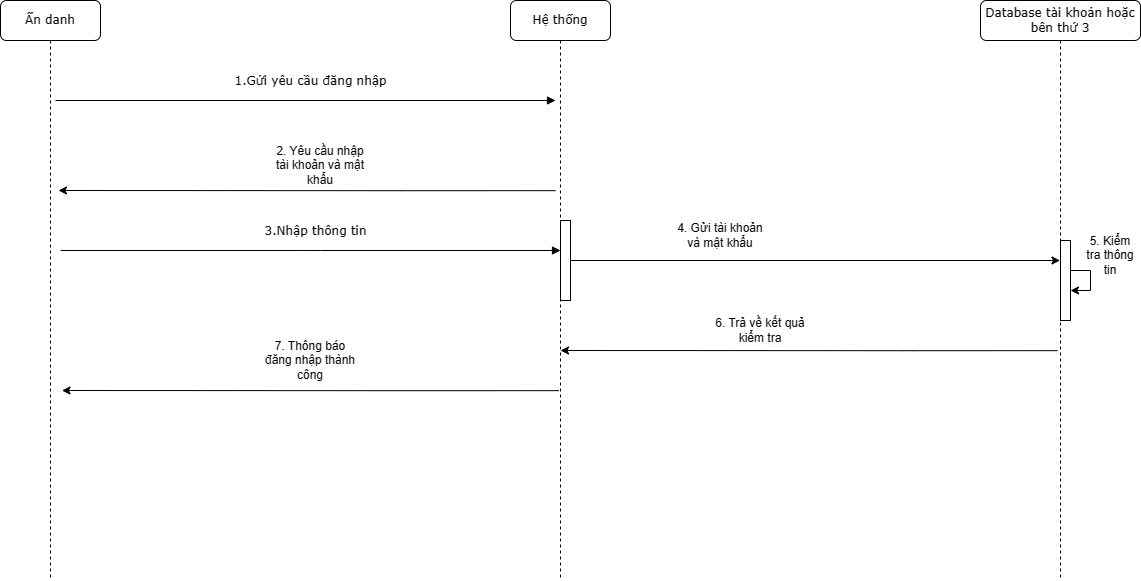


#### 3.4.6 Sơ đồ hoạt động miêu tả người dùng nghe nhạc online

#### 

## 3.5. Sơ đồ tuần tự

### 3.5.1. Sơ đồ tuần tự đăng nhập



#### 3.5.1. Sơ đồ tuần tự đăng nhập

#### 

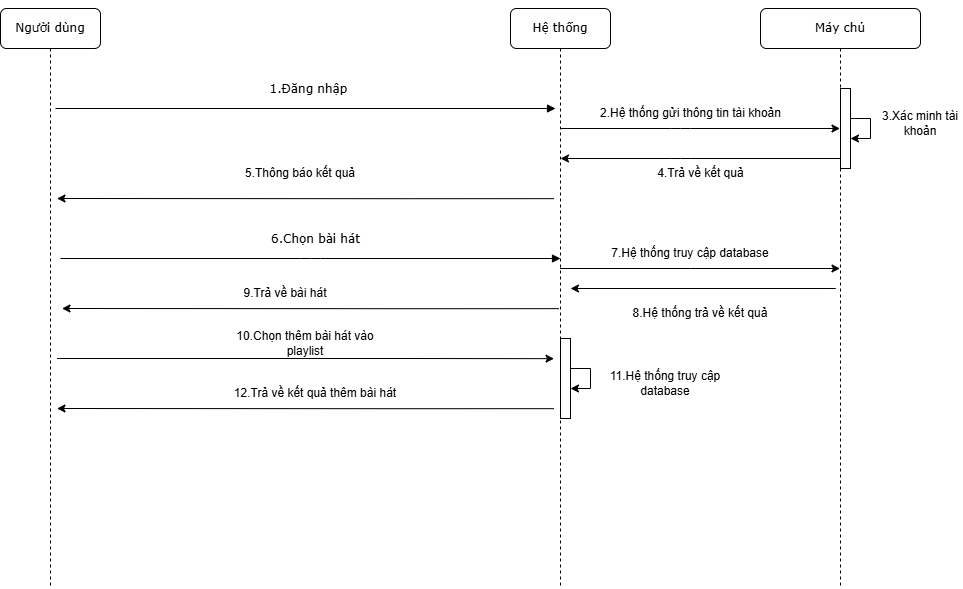
### 3.5.2. Sơ đồ tuần tự nghe nhạc

#### 3.5.2. Sơ đồ tuần tự nghe nhạc offline

### 

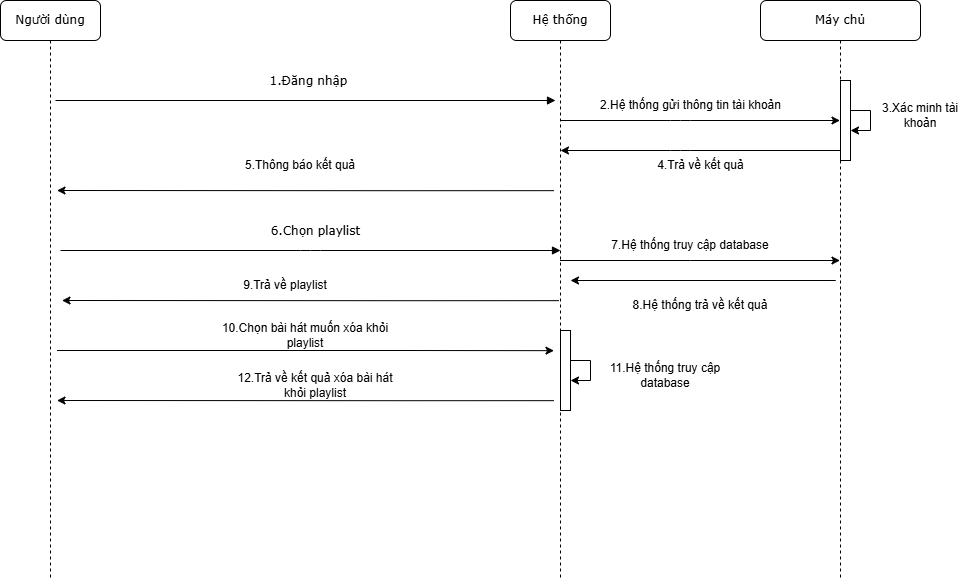
### 

### 3.5.3. Sơ đồ tuần tự thêm bài hát vào playlist



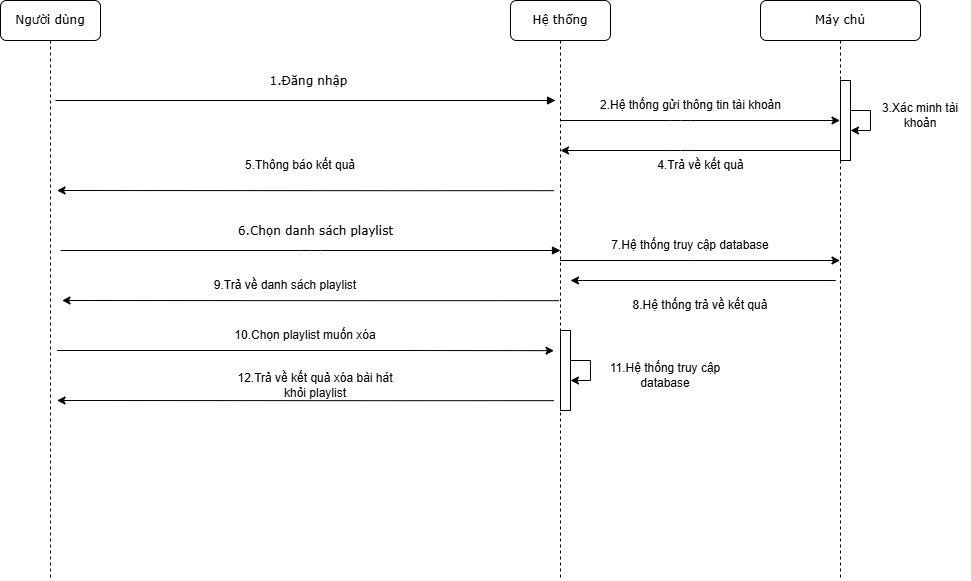
#### 3.5.3. Sơ đồ tuần tự thêm bài hát vào playlist

### 3.5.4. Sơ đồ tuần tự xóa bài hát khỏi playlist



#### 3.5.4. Sơ đồ tuần tự xóa bài hát khỏi playlist

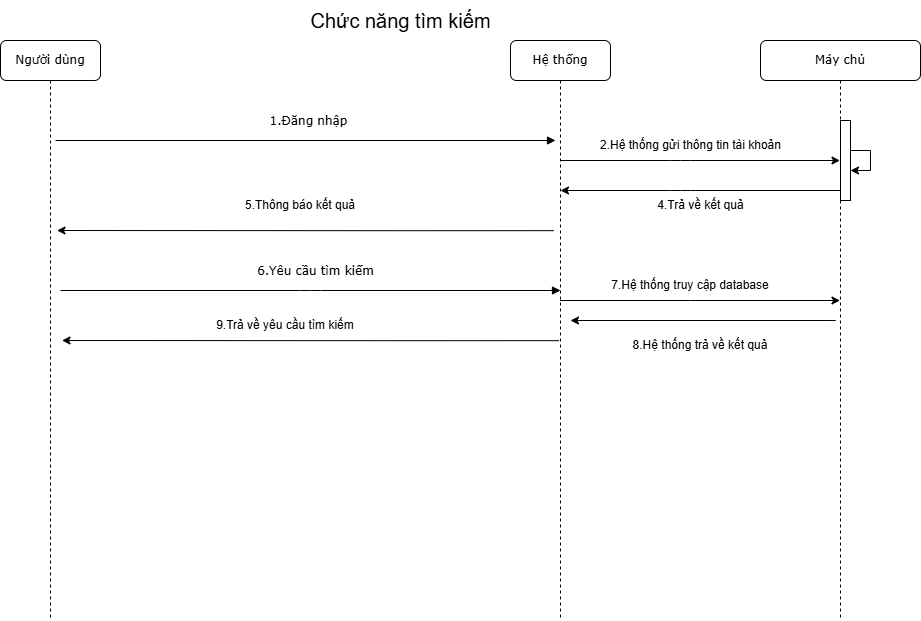
### 3.5.5. Sơ đồ tuần tự xóa playlist



#### 3.5.5. Sơ đồ tuần tự xóa playlist

#### 

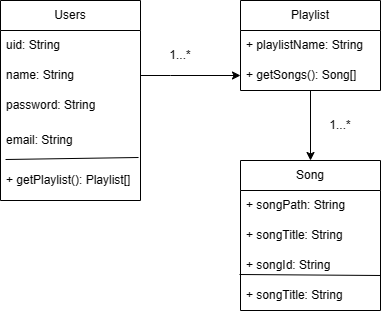
### 3.5.6. Sơ đồ tuần tự chức năng tìm kiếm



#### 3.5.6. Sơ đồ tuần tự chức năng tìm kiếm

#### 

## 3.6.Class Diagram:



#### 3.6. Class Diagram

# CHƯƠNG 4: THIẾT KẾ HỆ THỐNG

## 4.1. Kiến trúc hệ thống

Ứng dụng nghe nhạc trên điện thoại là hệ thống được thiết kế để cung cấp các dịch vụ liên quan đến việc nghe nhạc, quản lý playlist, khám phá bài hát mới, và chia sẻ âm nhạc với cộng đồng. Hệ thống được tối ưu hóa để hoạt động mượt mà trên các thiết bị di động như điện thoại thông minh và máy tính bảng, đồng thời cung cấp trải nghiệm người dùng thân thiện và tiện lợi cho người sử dụng ứng dụng.

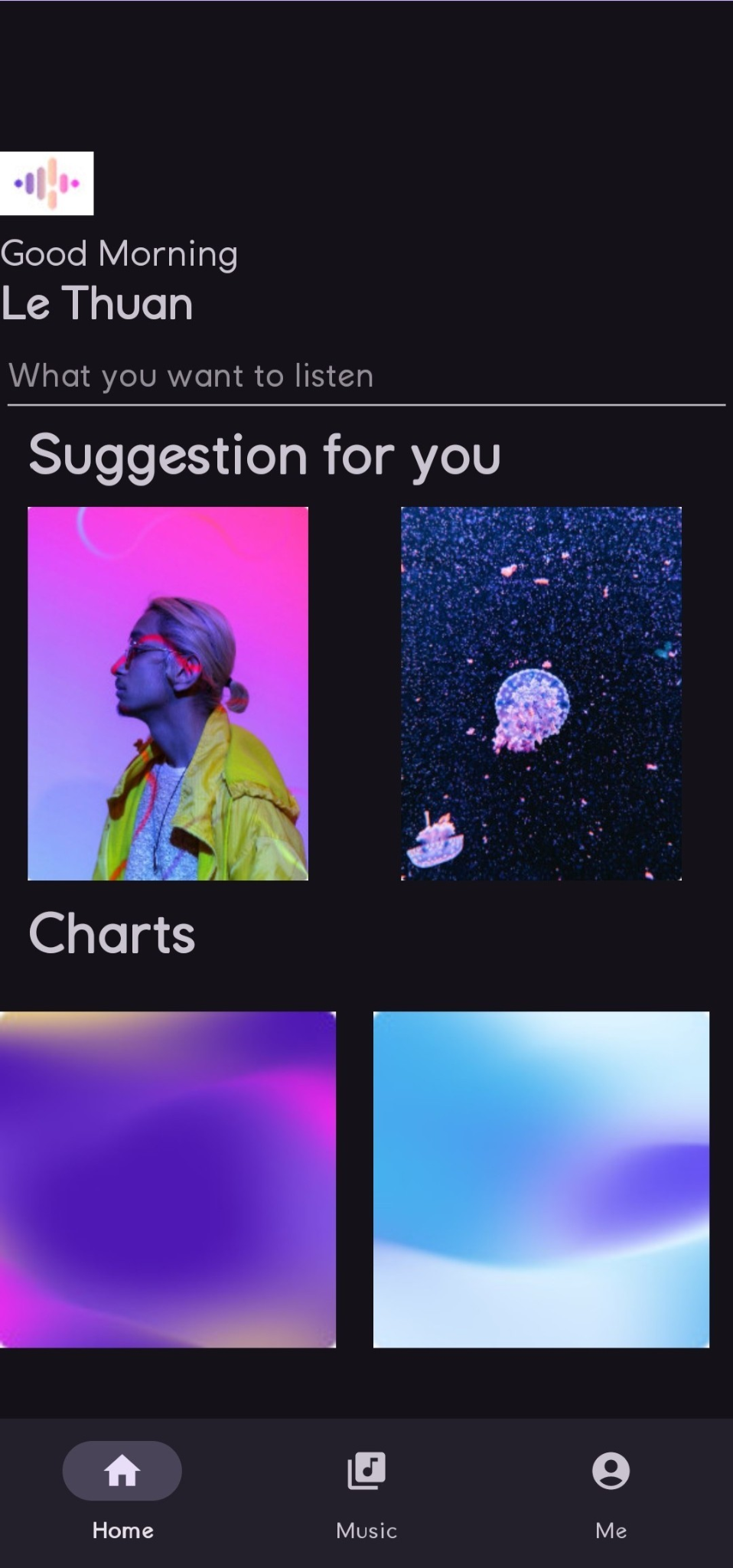
Các thành phần của kiến trúc hệ thống gồm có:

- Các đối tượng sử dụng: Mọi người dùng có nhu cầu nghe nhạc

- Thiết bị truy cập vào hệ thống: Máy tính bảng hoặc điện thoại di động thông minh.

## 4.2.Thiết kế giao diện:

### 4.2.1. Giao diện khi mở chương trình:



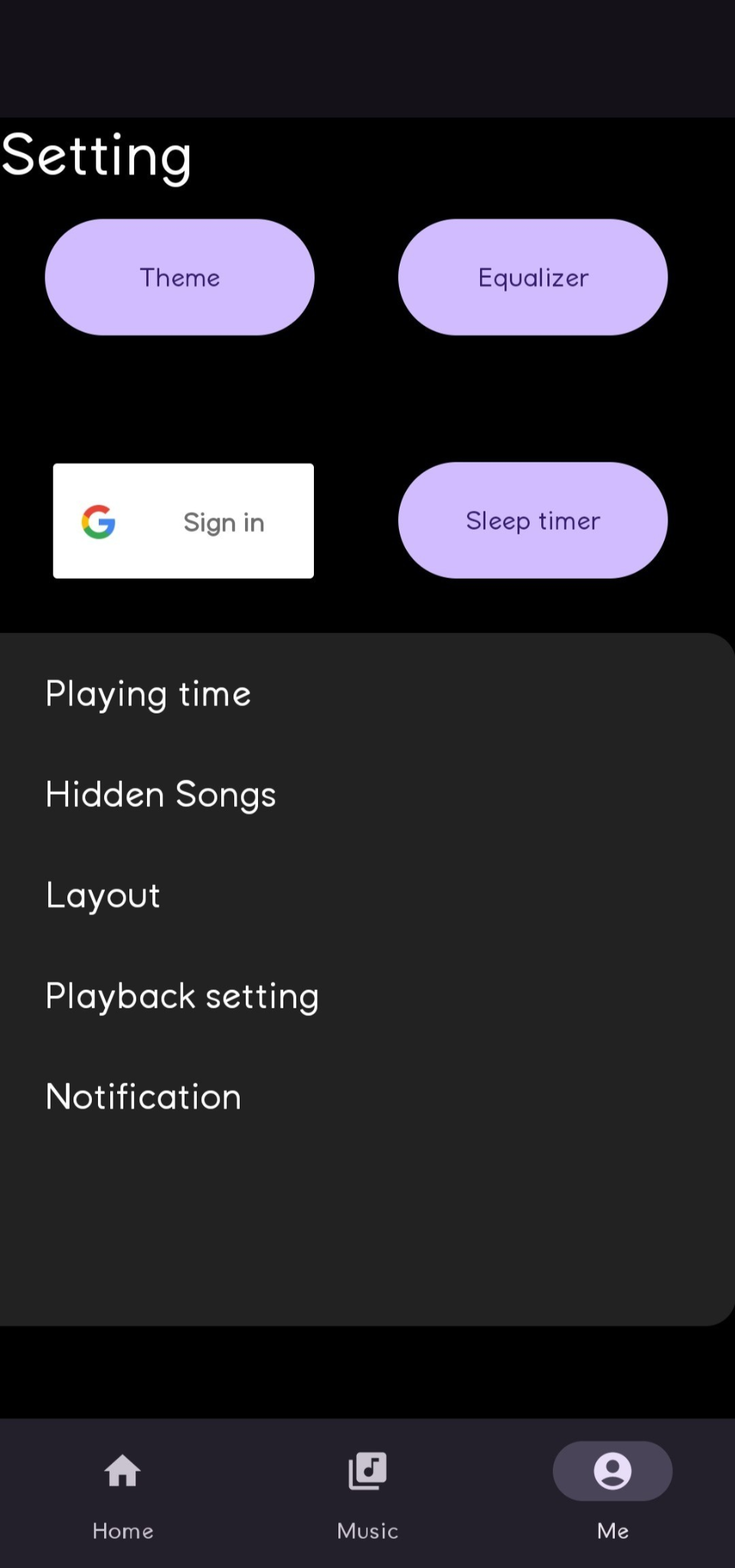
#### 4.2.1. Giao diện khi mở chương trình

### 4.2.2. Giao diện khi nghe nhạc offline



#### 4.2.2. Giao diện khi nghe nhạc offline

### 4.2.3. Giao diện đăng nhập:



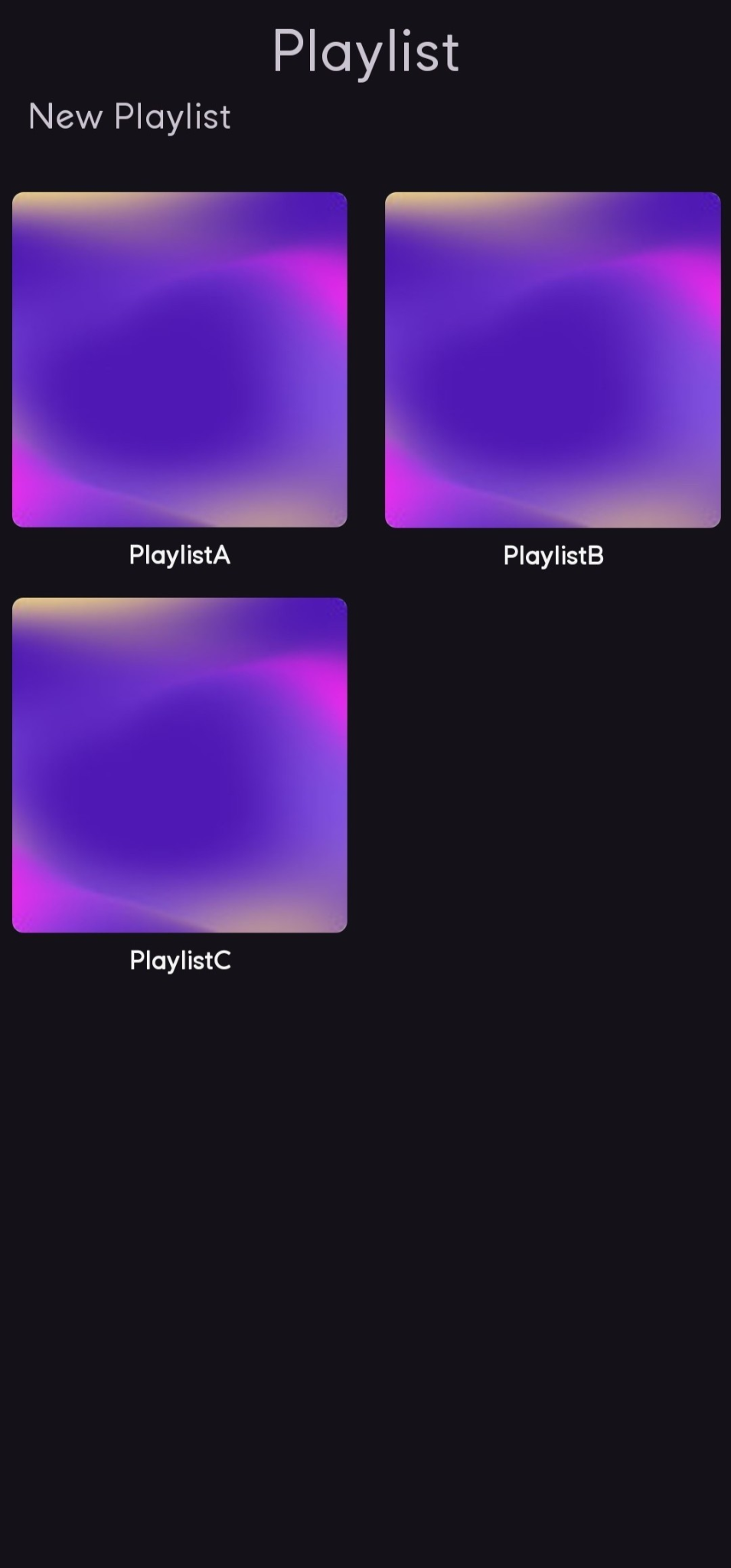
#### 4.2.3. Giao diện đăng nhập:

### 4.2.4:Giao diện khi nhấn vào bài hát để phát nhạc offline:



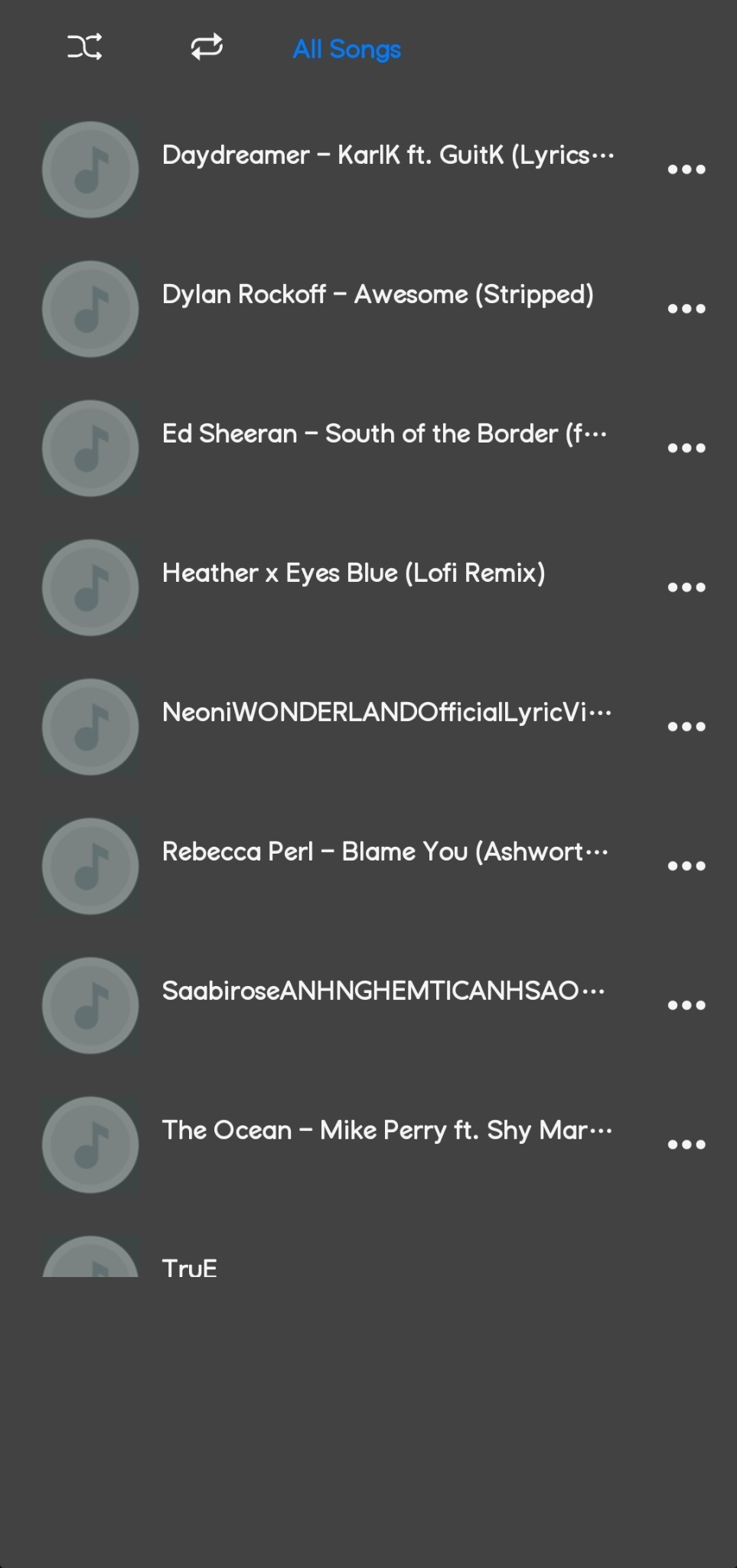
#### 4.2.4:Giao diện khi nhấn vào bài hát để phát nhạc offline

### 4.2.5: Giao diện Playlists:



#### 4.2.5: Giao diện Playlists

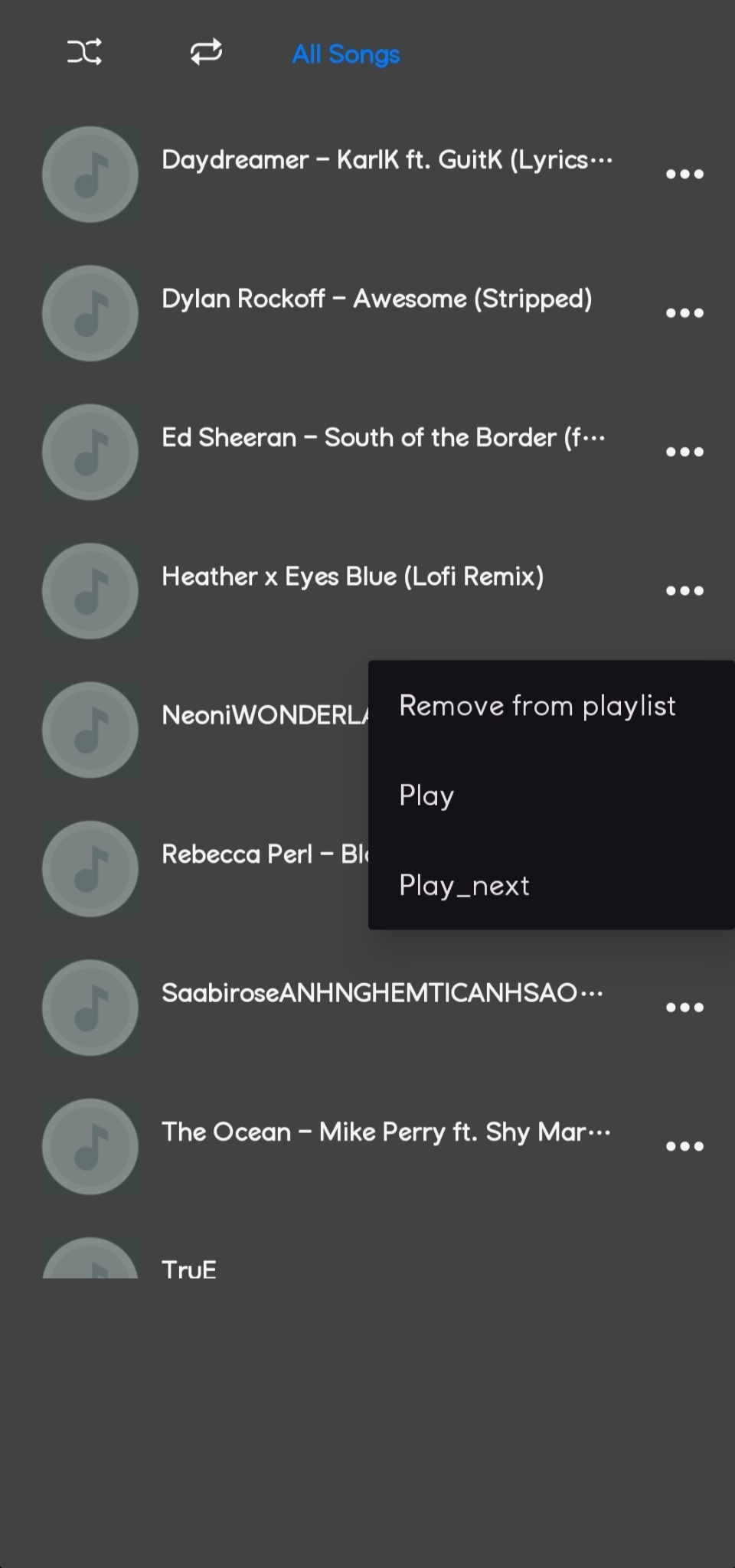
### 4.2.6: Giao diện hiển thị danh sách bài hát trong playlist:



#### 4.2.6: Giao diện hiển thị danh sách bài hát trong playlist

#### 

### 4.2.7: Giao diện khi người dùng muốn xóa bài hát ra khỏi playlist:



#### 4.2.7: Giao diện khi người dùng muốn xóa bài hát ra khỏi playlist

### 

### 4.2.8: Giao diện người dùng khi nhấn vào 1 trong các playlist online:

#### 4.2.8: Giao diện người dùng khi nhấn vào 1 trong các playlist online

### 4.2.9: Giao diện người dùng khi nghe nhạc online:

#### 4.2.9: Giao diện người dùng khi nghe nhạc online

### 

## 4.3 Code API

### 4.3.1. Spotify Online

##### 4.3.1.1. Thực hiện gọi API các bài hát Top 50 trên Spotify

public ArrayList<Song> getTop50(final VolleyCallBack callBack) {

String endpoint = "https://api.spotify.com/v1/playlists/37i9dQZEVXbMDoHDwVN2tF";

JsonObjectRequest jsonObjectRequest = new JsonObjectRequest

(Request.Method.GET, endpoint, null, response -> {

Gson gson = new Gson();

// Clear existing songs

songs.clear();

// Navigate JSON to retrieve tracks

JSONArray tracksArray = response.optJSONObject("tracks").optJSONArray("items");

if (tracksArray != null) {

for (int i = 0; i < tracksArray.length(); i++) {

JSONObject trackObject = tracksArray.optJSONObject(i);

if (trackObject != null) {

JSONObject trackInfo = trackObject.optJSONObject("track");

if (trackInfo != null) {

Song song = gson.fromJson(trackInfo.toString(), Song.class);

songs.add(song);

}

}

}

}

callBack.onSuccess();

}, error -> {

// TODO: Handle error

Log.d("ERROR", "getTop50: ");

}) {

@Override

public Map<String, String> getHeaders() throws AuthFailureError {

Map<String, String> headers = new HashMap<>();

String token = sharedPreferences.getString("token", "");

String auth = "Bearer " + token;

headers.put("Authorization", auth);

return headers;

}

};

queue.add(jsonObjectRequest);

return songs;

}

##### 4.3.1.2. Thực hiện việc thêm bài hát vào mục yêu thích

public void addSongToLibrary(Song song) {

JSONObject payload = preparePutPayload(song);

JsonObjectRequest jsonObjectRequest = prepareSongLibraryRequest(payload);

queue.add(jsonObjectRequest);

}

private JsonObjectRequest prepareSongLibraryRequest(JSONObject payload) {

return new JsonObjectRequest(Request.Method.PUT, "https://api.spotify.com/v1/me/tracks", payload, response -> {

}, error -> {

}) {

@Override

public Map<String, String> getHeaders() throws AuthFailureError {

Map<String, String> headers = new HashMap<>();

String token = sharedPreferences.getString("token", "");

String auth = "Bearer " + token;

headers.put("Authorization", auth);

headers.put("Content-Type", "application/json");

return headers;

}

};

}

##### 4.3.1.3. Thực hiện việc tìm kiếm bài hát

public ArrayList<Song> findTrack(final VolleyCallBack callBack) {

Log.d("findtrack", "12");

SharedPreferences sh = context.getApplicationContext().getSharedPreferences("FindTrackName", MODE\_PRIVATE);

String endpoint = "https://api.spotify.com/v1/search";

String type = "track";

String q = sh.getString("findTrackName" , "");

//String q = "Thu Cuoi";

Log.d("findtrack", q + type + endpoint);

JsonObjectRequest jsonObjectRequest = new JsonObjectRequest(

Request.Method.GET,

endpoint + "?q=" + q + "&type=" + type,

null,

response -> {

// Clear existing songs

songs.clear();

Log.d("findtrack", endpoint + "?q=" + q + "&type=" + type);

// Navigate JSON to retrieve tracks

JSONArray tracksArray = response.optJSONObject("tracks").optJSONArray("items");

if (tracksArray!= null) {

for (int i = 0; i < tracksArray.length(); i++) {

JSONObject trackObject = tracksArray.optJSONObject(i);

if (trackObject!= null) {

String trackName = trackObject.optString("name");

String trackId = trackObject.optString("id");

Song song = new Song(trackId, trackName , R.drawable.the\_strokes );

songs.add(song);

Log.d("testing" , songs.get(0).getId());

Log.d("testing" , songs.get(0).getName());

}

}

}

callBack.onSuccess();

Log.d("ERROR", "repeatTrack: " + response.toString());

},

error -> {

Log.d("ERROR2", "12121");

NetworkResponse networkResponse = error.networkResponse;

if (networkResponse!= null) {

int errorCode = networkResponse.statusCode;

Log.d("ERROR", "Error Code: " + errorCode);

} else {

Log.d("ERROR", "Error Code: Unknown");

}

}

) {

@Override

public Map<String, String> getHeaders() throws AuthFailureError {

Map<String, String> headers = new HashMap<>();

String token = sharedPreferences.getString("token", "");

String auth = "Bearer " + token;

headers.put("Authorization", auth);

return headers;

}

};

queue.add(jsonObjectRequest);

return songs;

}

##### 

##### 4.3.1.4. Thực hiện xử lý giữa người dùng và ứng dụng

public class Spotify\_Song\_Play extends AppCompatActivity {

ActivitySpotifySongPlayBinding binding;

SongService service;

ArrayList<Song> songs;

private static final String CLIENT\_ID ="4e5e333774674e6d9b9a34f9f68fe051";

private static final String REDIRECT\_URI = "http://localhost:8888/callback";

private SpotifyAppRemote mSpotifyAppRemote;

private Song song;

private PlayerApi playerApi;

int flag = 0;

int timer = 0; /// TIME

int flagLikedSong = 0;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

binding = ActivitySpotifySongPlayBinding.inflate(getLayoutInflater());

setContentView(binding.getRoot());

service = new SongService(getApplicationContext());

playCLickSong();

loadData();

event();

}

private void playCLickSong() {

SpotifyAppRemote.connect(this, connectionParams,

new Connector.ConnectionListener() {

@Override

public void onConnected(SpotifyAppRemote spotifyAppRemote) {

mSpotifyAppRemote = spotifyAppRemote;

Log.d("SongPlay", "Connected! Yay!");

connected();

}

@Override

public void onFailure(Throwable throwable) {

}

});

}

public void repeatTrack() {

mSpotifyAppRemote.getPlayerApi().setRepeat(1);

}

private void connected() {

SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs", Context.MODE\_PRIVATE);

String message = sharedPreferences.getString("TRACK\_ID\_KEY", "");

mSpotifyAppRemote.getPlayerApi().play("spotify:track:" + message);

Log.d("SongPlay","spotify:track:"+message);

}

ConnectionParams connectionParams =

new ConnectionParams.Builder(CLIENT\_ID)

.setRedirectUri(REDIRECT\_URI)

.showAuthView(true)

.build();

private void event() {

binding.songProgressbar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {

@Override

public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

if (fromUser){

int timePosition = (int) (progress \* getDuration() / 1000.0);

mSpotifyAppRemote.getPlayerApi().seekTo(timePosition);

}

}

@Override

public void onStartTrackingTouch(SeekBar seekBar) {

}

@Override

public void onStopTrackingTouch(SeekBar seekBar) {

}

});

binding.repeatButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

binding.repeatButton.setSelected(!binding.repeatButton.isSelected())

mSpotifyAppRemote.getPlayerApi().toggleRepeat();

}

});

binding.playPauseButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

if (flag == 0)

{

mSpotifyAppRemote.getPlayerApi().pause();

flag = 1;

}

else {

mSpotifyAppRemote.getPlayerApi().resume();

flag = 0;

}

binding.prevButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

mSpotifyAppRemote.getPlayerApi().skipPrevious();

updateSong();

}

});

binding.nextButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

mSpotifyAppRemote.getPlayerApi().skipNext();

updateSong();

}

});

binding.likeButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Log.d("ERROR1" , "stuff");

Intent intent = getIntent();

songs = intent.<Song>getParcelableArrayListExtra("SongList");

int position = getIntent().getIntExtra("Order" , 0);

if (songs.size() > 0 && flagLikedSong == 1 ) {

service.removeSongToLibrary(songs.get(position));

flagLikedSong = 0;

}

else {

service.addSongToLibrary(songs.get(position));

flagLikedSong = 1;

}

}

});

binding.buttonRight.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

timer += 10;

Toast.makeText(Spotify\_Song\_Play.this, "Nhạc sẽ dừng sau " + timer , Toast.LENGTH\_SHORT).show();

while (timer > 0){

try {

Thread.sleep(1000);

} catch (InterruptedException e) {

throw new RuntimeException(e);

}

timer -= 1;

}

if (timer == 0 ){

mSpotifyAppRemote.getPlayerApi().pause();

Toast.makeText(Spotify\_Song\_Play.this, "Nhạc đã dừng" , Toast.LENGTH\_SHORT).show();

}

}

});

}

});

}

private int getDuration()

{ return 200;}

private void updateSong(){

int position = getIntent().getIntExtra("Order" , 0);

binding.songName.setText((songs.get(position).getName()));

}

private void loadData() {

songs = new ArrayList<Song>();

Intent intent = getIntent();

songs = intent.<Song>getParcelableArrayListExtra("SongList");

int position = getIntent().getIntExtra("Order" , 0);

binding.songName.setText((songs.get(position).getName()));

//Play Song

if (songs.size() > 0) {

String trackID = songs.get(position).getId();

SharedPreferences sharedPreferences = getSharedPreferences("MyPrefs", Context.MODE\_PRIVATE);

SharedPreferences.Editor editor = sharedPreferences.edit();

editor.clear();

editor.putString("TRACK\_ID\_KEY", trackID);

editor.apply();

}

}

}

### 4.3.2. Nghe nhạc offline có trong máy

##### 4.3.2.1. Khởi tạo MusicAdapter

public class MusicAdapter extends RecyclerView.Adapter<MusicAdapter.ViewHolder> {

private ArrayList<HashMap<String, String>> songList;

ArrayList<String> playlists = getPlaylists();

FirebaseFirestore db = FirebaseFirestore.getInstance();

private List<String> playlistNames;

private Context context;

public MusicAdapter(ArrayList<HashMap<String, String>> songList) {

this.songList = songList;

}

@Override

public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item\_song\_normal, parent, false);

return new ViewHolder(view);

}

@Override

public void onBindViewHolder(ViewHolder holder, int position) {

HashMap<String, String> song = songList.get(position);

holder.songTitle.setText(song.get("songTitle"));

// Thiết lập OnClickListener cho ImageView menu\_button

holder.menuButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// Khởi tạo PopupMenu tại đây

PopupMenu popup = new PopupMenu(v.getContext(), holder.menuButton);

popup.getMenuInflater().inflate(R.menu.popup\_song, popup.getMenu());

popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {

public boolean onMenuItemClick(MenuItem item) {

// Xử lý sự kiện khi một item menu được chọn

if (item.getItemId() == R.id.popup\_song\_upload) {

// Hiển thị danh sách playlist hoặc tùy chọn tạo playlist mới

uploadSongToStorage(v.getContext(), song);

}

if (item.getItemId() == R.id.popup\_song\_add\_to\_playlist) {

// Hiển thị danh sách playlist hoặc tùy chọn tạo playlist mới

showPlaylistSelection(v.getContext(), song);}

return true;

}

});

popup.show(); // Hiển thị PopupMenu

}

});

}

@Override

public int getItemCount() {

return songList.size();

}

public class ViewHolder extends RecyclerView.ViewHolder {

ImageView menuButton;

TextView songTitle;

public ViewHolder(View itemView) {

super(itemView);

songTitle = itemView.findViewById(R.id.song\_title);

menuButton = itemView.findViewById(R.id.menu\_button);

// Set the click listener for the itemView

itemView.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

// Get the position of the clicked item

int position = getBindingAdapterPosition();

// Get the song path from the songList using the position

String songPath = songList.get(position).get("songPath");

Intent intent = new Intent(view.getContext(), screen\_now\_playing.class);

intent.putExtra("SONG\_PATH", songPath);

intent.putExtra("SONG\_LIST", songList);

intent.putExtra("SONG\_INDEX", position);

view.getContext().startActivity(intent);

}

});

}

}

}

##### 4.3.2.2.Thực hiện việc lấy nhạc ở thiết bị trong DashboardFragment

public class DashboardFragment extends Fragment {

private FragmentDashboardBinding binding;

private RecyclerView recyclerView;

private MusicAdapter musicAdapter;

private ArrayList<HashMap<String, String>> songList;

private static final int MY\_PERMISSIONS\_REQUEST\_WRITE\_STORAGE = 1;

public View onCreateView(@NonNull LayoutInflater inflater,

ViewGroup container, Bundle savedInstanceState) {

DashboardViewModel dashboardViewModel =

new ViewModelProvider(this).get(DashboardViewModel.class);

binding = FragmentDashboardBinding.inflate(inflater, container, false);

View root = binding.getRoot();

final TextView textView = binding.textDashboard;

dashboardViewModel.getText().observe(getViewLifecycleOwner(), textView::setText);

return root;

}

@Override

public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {

super.onViewCreated(view, savedInstanceState);

recyclerView = view.findViewById(R.id.recycler\_view);

recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));

// Lấy danh sách bài hát từ thiết bị

songList = getMusicFiles();

musicAdapter = new MusicAdapter(songList);

recyclerView.setAdapter(musicAdapter);

TextView playlistButton = view.findViewById(R.id.button\_playlist);

playlistButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

// Tạo Intent để khởi động Activity Playlists

Intent intent = new Intent(getActivity(), Playlists.class);

startActivity(intent);

}

});

}

public ArrayList<HashMap<String, String>> getMusicFiles() {

ArrayList<HashMap<String, String>> fileList = new ArrayList<>();

// Uri for external music files

Uri uri = MediaStore.Audio.Media.EXTERNAL\_CONTENT\_URI;

// Projection for the columns we want to retrieve

String[] projection = {

MediaStore.Audio.Media.\_ID,

MediaStore.Audio.Media.TITLE,

MediaStore.Audio.Media.DATA

};

// Query the content resolver

Cursor cursor = getContext().getContentResolver().query(uri, projection, null, null, null);

if (cursor != null) {

while (cursor.moveToNext()) {

HashMap<String, String> song = new HashMap<>();

song.put("songTitle", cursor.getString(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.TITLE)));

song.put("songPath", cursor.getString(cursor.getColumnIndexOrThrow(MediaStore.Audio.Media.DATA)));

// Adding each song to the list

fileList.add(song);

}

cursor.close();

}

return fileList;

}

@Override

public void onDestroyView() {

super.onDestroyView();

binding = null;

}

}

### 4.3.3. Đăng nhập vào tài khoản google

public class NotificationsFragment extends Fragment {

### 

private FragmentNotificationsBinding binding;

private GoogleSignInClient mGoogleSignInClient;

private int RC\_SIGN\_IN = 0;

FirebaseFirestore db = FirebaseFirestore.getInstance();

public View onCreateView(@NonNull LayoutInflater inflater,

ViewGroup container, Bundle savedInstanceState) {

binding = FragmentNotificationsBinding.inflate(inflater, container, false);

View root = binding.getRoot();

// Cấu hình Google Sign In

GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT\_SIGN\_IN)

.requestIdToken(getString(R.string.default\_web\_client\_id))

.requestEmail()

.build();

mGoogleSignInClient = GoogleSignIn.getClient(getActivity(), gso);

// Thêm nút đăng nhập Google vào layout và xử lý sự kiện click

SignInButton signInButton = binding.signInButton;

signInButton.setOnClickListener(v -> signIn());

Button updriveButton = root.findViewById(R.id.updrive);

updriveButton.setOnClickListener(new View.OnClickListener() {

@UnstableApi @Override

public void onClick(View v) {

Intent intent = new Intent(getActivity(), ActDrive.class);

startActivity(intent);

}

});

return root;

}

private void signIn() {

Intent signInIntent = mGoogleSignInClient.getSignInIntent();

startActivityForResult(signInIntent, RC\_SIGN\_IN);

}

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data) {

super.onActivityResult(requestCode, resultCode, data);

if (requestCode == RC\_SIGN\_IN) {

Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);

handleSignInResult(task);

}

}

private void handleSignInResult(Task<GoogleSignInAccount> completedTask) {

try {

GoogleSignInAccount account = completedTask.getResult(ApiException.class);

AuthCredential credential = GoogleAuthProvider.getCredential(account.getIdToken(), null);

Log.w("token", "token=" + account.getIdToken());

FirebaseAuth firebaseAuth = FirebaseAuth.getInstance();

firebaseAuth.signInWithCredential(credential)

.addOnCompleteListener(getActivity(), task -> {

if (task.isSuccessful()) {

FirebaseUser user = firebaseAuth.getCurrentUser();

Intent intent = new Intent(getActivity(), MainActivity2.class);

startActivity(intent);

} else {

Log.w("error", "signInWithCredential:failure", task.getException());

Toast.makeText(getActivity(), "Authentication failed.", Toast.LENGTH\_SHORT).show();

}

});

} catch (ApiException e) {

Log.w("error", "signInResult:failed code=" + e.getStatusCode());

}

}

@Override

public void onDestroyView() {

super.onDestroyView();

binding = null;

}

}

### 

### 4.3.4. Chức năng load bài hát lên cloud(Storage of Firestore)

private void uploadSongToStorage(Context context, HashMap<String, String> song) {

String userEmail = GlobalVars.getUserEmail();

if (userEmail != null && !userEmail.isEmpty()) {

// Người dùng đã đăng nhập, tiến hành tải lên

String songPath = song.get("songPath");

Uri file = Uri.fromFile(new File(songPath));

StorageReference storageRef = FirebaseStorage.getInstance().getReference();

String sanitizedEmail = userEmail.replace(".", "\_");

StorageReference songRef = storageRef.child("users/" + sanitizedEmail + "/" + file.getLastPathSegment());

UploadTask uploadTask = songRef.putFile(file);

uploadTask.addOnSuccessListener(new OnSuccessListener<UploadTask.TaskSnapshot>() {

@Override

public void onSuccess(UploadTask.TaskSnapshot taskSnapshot) {

Toast.makeText(context, "Tải lên thành công.", Toast.LENGTH\_SHORT).show();

}

}).addOnFailureListener(new OnFailureListener() {

@Override

public void onFailure(@NonNull Exception e) {

Toast.makeText(context, "Tải lên thất bại: " + e.getMessage(), Toast.LENGTH\_SHORT).show();

}

});

} else {

// Người dùng chưa đăng nhập hoặc email không hợp lệ

Toast.makeText(context, "Bạn cần đăng nhập để sử dụng chức năng này.", Toast.LENGTH\_SHORT).show();

}

}

### 4.3.5. Tải bài hát có trên cloud và xóa bài hát khỏi cloud

// Phương thức để tải bài hát

private void downloadSong(Song\_Dri song, Context context) {

// Tạo request để tải bài hát

DownloadManager.Request request = new DownloadManager.Request(Uri.parse(song.getUrl()));

request.setAllowedNetworkTypes(DownloadManager.Request.NETWORK\_WIFI | DownloadManager.Request.NETWORK\_MOBILE)

.setTitle(song.getName())

.setDescription("Downloading...")

.setNotificationVisibility(DownloadManager.Request.VISIBILITY\_VISIBLE\_NOTIFY\_COMPLETED)

.setDestinationInExternalPublicDir(Environment.DIRECTORY\_DOWNLOADS, song.getName() + ".mp3");

// Lấy DownloadManager từ context và thêm vào hàng đợi

DownloadManager manager = (DownloadManager) context.getSystemService(Context.DOWNLOAD\_SERVICE);

if (manager != null) {

manager.enqueue(request);

} else {

Toast.makeText(context, "Không thể tải xuống bài hát", Toast.LENGTH\_SHORT).show();

}

private Object getSystemService(String downloadService) {

return null;

}

// Phương thức để xóa bài hát

private void deleteSong(Song\_Dri song, int position, View view) {

// Lấy tham chiếu đến bài hát trong Firebase Storage

StorageReference songRef = FirebaseStorage.getInstance().getReferenceFromUrl(song.getUrl());

// Xóa bài hát

songRef.delete().addOnSuccessListener(aVoid -> {

// Xóa bài hát khỏi danh sách và thông báo cho adapter

songs.remove(position);

notifyItemRemoved(position);

notifyItemRangeChanged(position, songs.size());

Toast.makeText(view.getContext(), "Bài hát đã được xóa", Toast.LENGTH\_SHORT).show(); // Sử dụng view.getContext() thay vì v.getContext()

}).addOnFailureListener(e -> {

Toast.makeText(view.getContext(), "Lỗi khi xóa bài hát", Toast.LENGTH\_SHORT).show(); // Sử dụng view.getContext() thay vì v.getContext()

});

}}

# KẾT LUẬN VÀ KIẾN NGHỊ

1. Kết quả đạt được:

Trong quá trình nghiên cứu và hoàn thành đồ án tốt nghiệp với đề tài “Ứng dụng nghe nhạc bằng điện thoại”, chúng em đã đạt được các kết quả như sau:

Về mặt kiến thức:

* Hiểu được ngôn ngữ Java và cách hoạt động của Android Studio cũng như hệ quản trị cơ sở dữ liệu Firestore.
* Hiểu được các bước cần thiết để quản lý và phát triển một ứng dụng nghe nhạc trên điện thoại.
* Hiểu được sự quan trọng của quy trình phân tích nghiệp vụ trước khi bắt đầu thi hành dự án
* Hiểu được tầm quan trọng của sơ đồ trong khi thực thi dự án

Về mặt ứng dụng:

* Từ những kết quả có được về mặt kiến thức, chúng em đã xây dựng được các tài liệu cần thiết để triển khai một ứng dụng nghe nhạc. Từ những yêu cầu mà người dùng đưa ra, đã xây dựng thành công các quy trình, các luồng xử lý trong ứng dụng. Xây dựng được cơ sở dữ liệu trên Firestore để từ đó có thể triển khai hệ thống bằng Java và Android Studio.

Về mặt con người:

* Quá trình làm đồ án đã giúp chúng em học hỏi và rèn luyện khả năng tìm kiếm, nghiên cứu, và xử lý vấn đề. Chúng em đã phát triển kỹ năng lập trình, kiên nhẫn và khả năng trình bày văn bản, đây là những kỹ năng quan trọng cho tương lai nghề nghiệp.

1. Tồn tại

Bên cạnh những mặt tích cực, đề tài còn tồn tại một số hạn chế như:

* Các tài liệu về phân tích chức năng chưa có cơ hội tương tác trực tiếp với người dùng, nên còn thiếu sự chính xác.
* Tài liệu cơ sở dữ liệu còn thiếu sót.
* Do thời gian và kiến thức còn hạn chế, việc triển khai các quy trình chỉ mới ở mức độ tổng quát

1. Hướng phát triển

Trong tương lai nếu có điều kiện đồ án của em sẽ phát triển theo các hướng sau:

* Khảo sát ý kiến người dùng để xây dựng các quy trình và chức năng chính xác hơn.
* Nghiên cứu sâu hơn để xây dựng cơ sở dữ liệu phù hợp hơn trên Firestore.
* Triển khai tất cả các quy trình, tận dụng khả năng của Java, Android Studio và Firestore để phát triển ứng dụng nghe nhạc tối ưu và hiệu quả.
* Dự đoán hành vi, xu hướng của người dùng bằng database và trực quan hóa bằng Power BI/Tableau.
* Tích hợp phát triển song song thêm ứng dụng trên website và desktop.

# 

# 

# 

# 

# 

# 

# TÀI LIỆU THAM KHẢO

Java Introduction. [Introduction to Java (w3schools.com)](https://www.w3schools.com/java/java_intro.asp)

Developer guides: [Developer guides | Android Developers](https://developer.android.com/guide)

MediaPlayer: [MediaPlayer | Android Developers](https://developer.android.com/reference/android/media/MediaPlayer)

Firebase Firestore Migrations:  [Firebase Firestore Migrations - DEV Community](https://dev.to/mikgross/firebase-firestore-migrations-olg)

What is an API (Application Programming Interface): [What is an API (Application Programming Interface) - GeeksforGeeks](https://www.geeksforgeeks.org/what-is-an-api/)

How to build a simple music player app using Android Studio: [How to build a simple music player app using Android Studio - GeeksforGeeks](https://www.geeksforgeeks.org/how-to-build-a-simple-music-player-app-using-android-studio/)

New Google Sign-In API | Authentication | Google for Developers: [API Đăng nhập bằng Google mới | Authentication | Google for Developers](https://developers.google.com/identity/sign-in/android/sign-in-identity?hl=vi)

**Phân công công việc**

| Lê Công Thuận | BE, API spotify |
| --- | --- |
| Nguyễn Ngô Thành Tài | BA, FE giao diện chung |
| Trịnh Nguyễn Vy Na | BE FE offline, API Google, Firestore |