BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN
TRƯỜNG ĐẠI HỌC THỦY LỢI

BÀI TẬP LỚN

HỌC PHẦN: LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

ĐỀ TÀI: Xây dựng ứng dụng Todo App (Java Console App)

Mã Sinh Viên	Họ và Tên	Ngày Sinh	Lớp
2151173744	Nguyễn Đại An	18/01/2003	63KTPM1
112	Nguyễn Duy Thái	1/11/2003	63KTPM1
2151170588	Nguyễn Ngọc Quý	24/03/2003	63KTPM1

Hà Nội, năm 2024

LỜI NÓI ĐẦU

MỤC LỤC

MỤC LỤC HÌNH ẢNH

MỤC LỤC BẢNG

BẢNG CÁC TỪ VIẾT TẮT

STT	TỪ VIẾT TẮT	VIẾT ĐẦY ĐỦ
1	CSDL	Cơ sở dữ liệu
2		

CHƯƠNG 1. MÔ TẢ BÀI TOÁN

1.1. Giới thiệu

Âm nhạc đóng vai trò quan trọng trong cuộc sống con người, giúp chúng ta giải tỏa những áp lực trong cuộc sống hằng ngày, là sợi dây liên kết giữa con người với con người. Do đó nhu cầu nghe nhạc cũng cũng như chia sẻ những bài hát mình yêu thích cho mọi người khác cùng thưởng thức hiện nay đang rất cao và các ứng dụng nghe nhạc khác vẫn vẫn chưa có thể hoàn toàn đáp ứng ứng được. Vì vậy Vii Music ra đời.

1.2. Chức năng chính

Ứng dụng Vii Music cần có các chức năng sau:

- Chức năng phát nhạc: Cốt lõi của chức năng này là giúp người dùng phát các bài hát từ danh sách cá nhân hoặc từ thư viện nhạc trực tuyến, cho phép họ tôn vinh âm nhạc mọi lúc, mọi nơi
- Quản lý phát danh sách(danh sách phát): Khi người dùng chọn một danh sách phát (danh sách phát), ứng dụng sẽ phát lần lượt như các bài hát trong danh sách đó mà không cần thiết phải thủ công. Điều này giúp tiết kiệm thời gian và mang lại trải nghiệm nghệ thuật liền mạch.
- Tìm kiếm nhạc: Cho phép người tìm kiếm những bản nhạc yêu thích theo mong muốn.
- Tài khoản và quản lý cá nhânhân: Cho phép người dùng kiểm tra và quản lý thông tin cá nhân của mình.
- Lưu trữ dữ liệu: Ứng dụng cần kết nối và lưu trữ dữ liệu trên SQLite.

1.3. Yêu cầu phi chức năng

- Dễ sử dụng: Giao diện console cần rõ ràng, dễ hiểu và dễ sử dụng.
- Hiệu suất: Thời gian phản hồi nhanh, có thời gian tải nhạc nhanh và bộ nhớ tối ưu.
- Quản lý thư viện nhạc lớn : Khả năng lưu trữ và quản lý thư viện nhạc lớn mà vẫn duy trì tốc độ truy cập nhanh hơn.
- Bảo mật thông tin cá nhân: Đảm bảo rằng chỉ những người dùng hợp lệ mới có thể truy cập vào các nội dung bảo mật hoặc tính năng cao cấp của ứng dụng.
- Khả năng hoạt động ngoại tuyến: Đảm bảo rằng chỉ những người dùng hợp lệ mới có thể truy cập vào các nội dung bảo mật hoặc tính năng cao cấp của ứng dụng.

CHƯƠNG 2. PHÂN TÍCH YÊU CẦU VÀ THIẾT KẾ HỆ THỐNG

2.1. Phân tích yêu cầu:

Xác định người dùng:

- Người dùng cuối là bất kỳ ai muốn sử dụng ứng dụng để sử dụng vào mục đích yêu thích cá nhân. Họ có thể là học sinh, sinh viên, nhân viên văn phòng, hoặc bất kỳ ai cần tổ chức công việc hàng ngày.
- Người dùng có thể có các mức độ am hiểu về âm nhạc khác nhau, do đó ứng dụng cần dễ sử dụng và thân thiện với người dùng.

Thu thập yêu cầu:

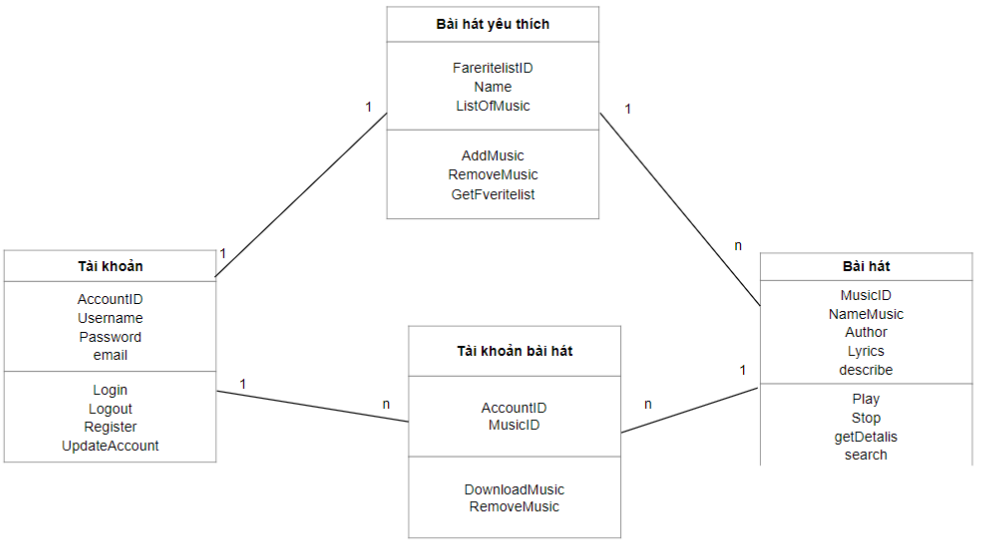
- Dựa trên mô tả bài toán, ta đã xác định được các chức năng chính (Chức năng phát nhạc, Quản lý phát danh sách, Tìm kiếm nhạc, Tài khoản và quản lý cá nhân) và yêu cầu phi chức năng (dễ sử dụng, hiệu năng, Thư viện lớn, Bảo mật, Hoạt động ngoại tuyến) của ứng dụng.

Phân tích yêu cầu:

- **Chức năng phát nhạc:** Chức năng phát nhạc là một phần cốt lõi của ứng dụng nhạc, cho phép người dùng phát những bản nhạc mình yêu thích
- **Hiển thị danh sách yêu thích:** Hiển thị danh sách các bản nhạc yêu thích của người dùng được thêm vào
- **Danh sách nhạc tải về:** Người dùng có thể tải về những bản nhạc yêu thích của mình để nghe nó khi không có kết nối mạng.
- **Chỉnh sửa thông tin cá nhân:** Người dùng update thông tin của mình trong trang mục tài khoản.

2.2. Thiết kế hệ thống:

2.2.1 Biểu đồ lớp cho mô hình miền:



- Tập trung vào các thực thể nghiệp vụ và mối quan hệ giữa chúng.
- Thường không chứa các lớp liên quan đến giao diện người dùng, cơ sở dữ liệu, hay các chi tiết kỹ thuật khác.
- Quan hệ:
    - Một tài khoản có nhiều bản nhạc trong danh sách yêu thích (thông qua thực thể trung gian Tài khoản - Bản nhạc).
    - Một tài khoản có thể có một danh sách yêu thích
    - Danh sách yêu thích có nhiều bản nhạc
    - Nhiều bản nhac nằm trong 1 danh sách yêu thích
    - Một tài khoản có thể thêm nhiều bản nhạc vào danh sách yêu thích.
    - Nhiều bản nhạc nằm trong một danh sách yêu thích
- Mối quan hệ giữa các thực thể
    - Tài khoản có mối quan hệ nhiều-nhiều với Bản nhạc thông qua thực thể Tài khoản - Bản nhạc
    - Tài khoản có một danh sách yêu thích

2.2.2 Biểu đồ lớp cho MVC:

- Mô tả kiến trúc của ứng dụng, bao gồm các lớp trong Model, View và Controller.
- Thể hiện mối quan hệ giữa các lớp trong MVC và cách chúng tương tác với nhau.
- Ví dụ: trong ứng dụng App MusiMusic, biểu đồ lớp cho MVC sẽ chứa các lớp như: Account, Favoritelist, Song

2.2.3 Thiết kế kiến trúc (MVC):

- **Model:**
    - **Account**: Class đại diện cho một tài khoản người dùng chứa các thuộc tính: ID, name, email, password
    - FavoriteList: Class quản lý danh sách các bản nhạc yêu thích, chứa các phương thức: thêm, xóa
- **View:**
    - **TaskView**: Class hiển thị thông tin công việc trên console: hiển thị menu, danh sách yêu thích, chơi nhạc, ...
- **Controller:**
    - **TaskControlle**r: Class tiếp nhận yêu cầu từ người dùng thông qua TaskView, xử lý yêu cầu bằng cách gọi các phương thức của Model, sau đó trả kết quả về TaskView để hiển thị.

Dựa trên kiến trúc MVC đã chọn, ta có thể xác định các lớp sau:

- Lớp Task:
    - Thuộc tính:ID, name, email, password.
    - Phương thức:
        - Getters và setters cho các thuộc tính.
        - toString(): trả về chuỗi mô tả thuộc tính.
- Lớp TaskList:
    - Thuộc tính: List<String> tasks chứa các tác vụ mà người dùng có thể thực hiện, như "Thêm bài hát yêu thích", "Xóa bài hát yêu thích", hoặc "Chơi nhạc").
    - Phương thức:

        - `addTask(task: String)`: Thêm một tác vụ mới vào danh sách `tasks`.
        - `removeTask(task: String)`: Xóa một tác vụ khỏi danh sách `tasks`.
        - `listTasks()`: Lấy danh sách các tác vụ hiện có.
        - `addSongToFavorites(song: String)`: Thêm một bài hát vào danh sách yêu thích thông qua đối tượng `FavoriteList`.
        - `removeSongFromFavorites(song: String)`: Xóa một bài hát khỏi danh sách yêu thích.
        - `playSong(song: String)`: Chơi bài hát được chỉ định, có thể kèm theo việc hiển thị thông tin hoặc gửi yêu cầu lên `TaskView` để hiển thị.
- Lớp TaskView:
    - Phương thức:
        - `displayMenu(): void` – Hiển thị menu chính của ứng dụng
        - `displayFavorite(songs: List<Song>): void` – Hiển thị danh sách các bài hát yêu thích.
        - `displayPlaySong(message: String): void` – Hiển thị giao diện phát bài hát.
- Lớp TaskController:
    - Thuộc tính:
        - Account account: đối tượng Account.
        - FavouriteList favouriteList: đối tượng TaskView.
    - Phương thức:

- `addSongToFavorites(accountId: int, songId: int): void` – Xử lý việc thêm bài hát vào danh sách yêu thích của tài khoản.
- `removeSongFromFavorites(accountId: int, songId: int): void` – Xử lý việc xóa bài hát khỏi danh sách yêu thích.
- handleUserInput:void - Xử lý các tương tác của người dùng với hệ thống
- playSong(accountId: int):void - Xử lý phát bài hát

Mối quan hệ giữa các lớp:

- TaskView:

- Liên kết một chiều: Lớp TaskView sử dụng các phương thức của TaskController để thực hiện các tác vụ liên quan đến tài khoản và danh sách yêu thích. Ví dụ, TaskView có thể gọi các phương thức như addSongToFavorites(), removeSongFromFavorites(), và playSong() từ TaskController để quản lý các hành động của người dùng.
- Liên kết một chiều: Lớp TaskView sử dụng các thuộc tính và phương thức của lớp Account để hiển thị thông tin người dùng. Điều này có thể bao gồm việc hiển thị tên, email, và các thông tin cá nhân khác của người dùng thông qua các phương thức như getName(), getEmail(), và getPassword().
- Liên kết một chiều: Lớp TaskView sử dụng các thuộc tính và phương thức của lớp FavoriteList để hiển thị danh sách yêu thích của người dùng. Phương thức displayFavorite() của TaskView sẽ hiển thị danh sách các bài hát hoặc mục yêu thích từ FavoriteList.

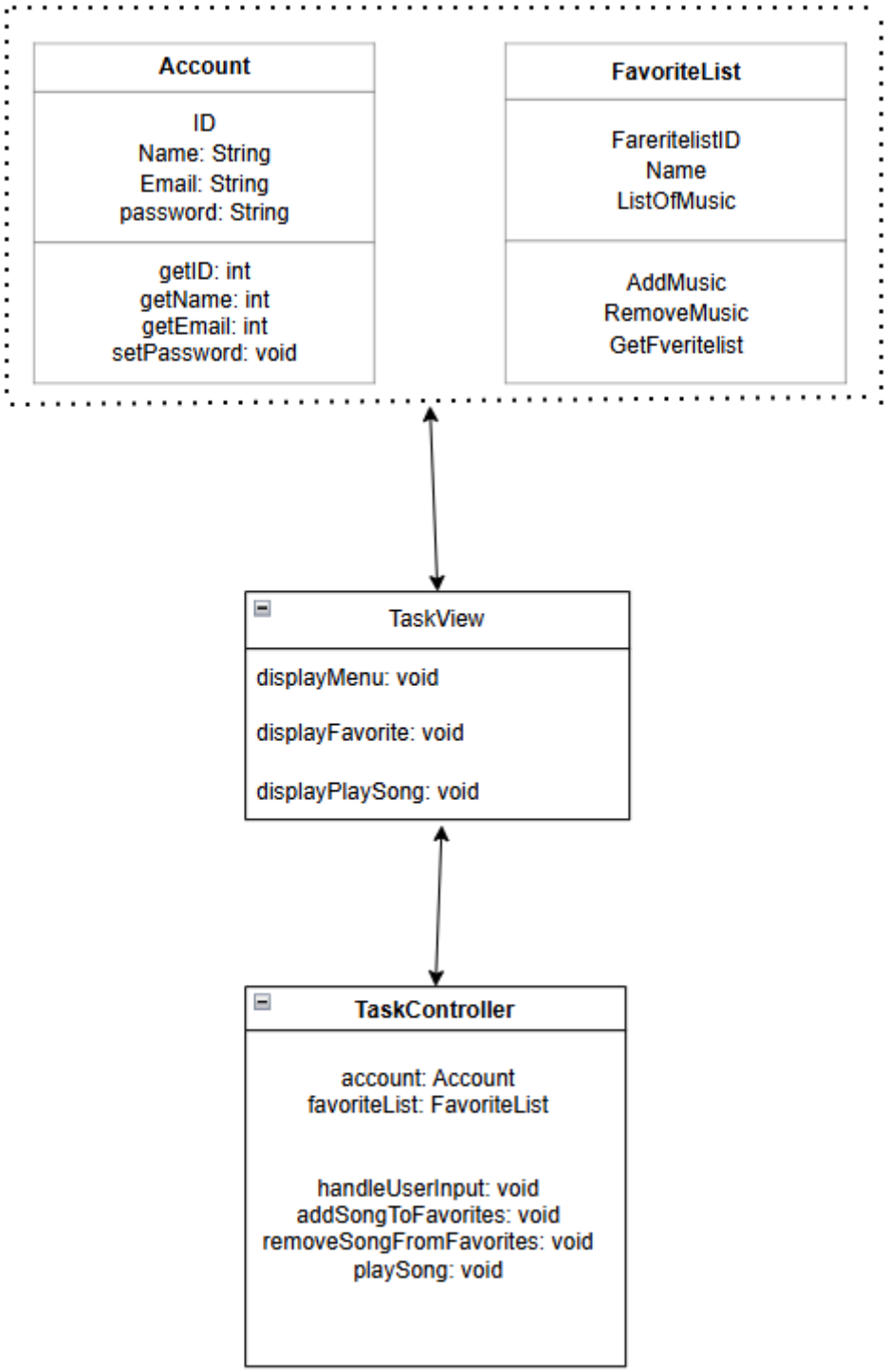
- TaskController:

- Liên kết một chiều: Lớp TaskController cung cấp các phương thức mà TaskView sử dụng để thực hiện các tác vụ liên quan đến tài khoản và danh sách yêu thích. Ví dụ, TaskView có thể gọi các phương thức như addSongToFavorites(), removeSongFromFavorites(), và playSong() từ TaskController để quản lý các hành động của người dùng.
- Liên kết một chiều: Lớp TaskController có thuộc tính account kiểu Account, cho thấy rằng TaskController sử dụng các thuộc tính và phương thức của Account để quản lý thông tin tài khoản người dùng. Điều này bao gồm việc truy cập và cập nhật thông tin tài khoản như tên, email, và mật khẩu.

- Liên kết một chiều: Lớp TaskController có thuộc tính favoriteList kiểu FavoriteList, cho thấy rằng TaskController sử dụng FavoriteList để quản lý danh sách yêu thích của người dùng. Điều này bao gồm việc thêm, xóa và truy xuất các mục yêu thích từ danh sách.

- Account:
- Liên kết một chiều: Lớp TaskController có thuộc tính account kiểu Account, cho thấy rằng TaskController sử dụng các thuộc tính và phương thức của Account để quản lý các tác vụ liên quan đến tài khoản người dùng.
- Liên kết một chiều: Lớp TaskController cũng có thuộc tính favoriteList kiểu FavoriteList, cho thấy rằng TaskController sử dụng FavoriteList để quản lý danh sách yêu thích của người dùng.
- Liên kết một chiều: Lớp TaskView có thể hiển thị thông tin từ Account thông qua các phương thức như displayMenu(), displayFavorite(), và displayPlaySong(), mặc dù không có thuộc tính trực tiếp liên kết với Account.

- FavouriteList:
- Liên kết một chiều: Lớp TaskController có thuộc tính favoriteList kiểu FavoriteList, cho thấy rằng TaskController sử dụng FavoriteList để quản lý danh sách yêu thích của người dùng.
- Liên kết một chiều: Lớp TaskView có phương thức displayFavorite(), cho thấy rằng TaskView sử dụng FavoriteList để hiển thị danh sách yêu thích của người dùng.
- Liên kết một chiều: Lớp Account không có liên kết trực tiếp với FavoriteList trong biểu đồ này, nhưng TaskController sử dụng cả Account và FavoriteList, cho thấy rằng Account có thể gián tiếp liên quan đến FavoriteList thông qua TaskController.



S

2.3 Thiết kế cơ sở dữ liệu:

- Tạo một Sqlite mới để lưu trữ dữ liệu.
- Xác định tên cheat sheet và các cột tương ứng với các thuộc tính của dữ liệu.
- Kết nối cơ sở dữ liệu với app.

2.4 Thiết kế giao diện:

- Hiển thị menu chính với các lựa chọn:
    1. Trang chủ
    2. Hiển thị danh sách các bài hát yêu
    3. Chơi nhạc
    4. Danh sách tải về
    5. Trang thông tin cá nhân
- Sử dụng các thông báo rõ ràng để hướng dẫn người dùng nhập liệu và hiển thị kết quả.

2.5. Triển khai:

- **Viết code:** Sử dụng Java để cài đặt các class trong mô hình MVC, đọc/ghi file, xử lý dữ liệu và hiển thị giao diện console.
- **Kiểm thử:** Viết các test case để kiểm tra các chức năng của ứng dụng, đảm bảo ứng dụng hoạt động đúng theo yêu cầu.

2.6. Vận hành và bảo trì:

- Cài đặt và triển khai:
    - Hướng dẫn người dùng cách sử dụng ứng dụng .
- 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

3.1. Công nghệ đã sử dụng

- Ngôn ngữ lập trình: Java
- Công cụ: Android studio
- Thư viện: (tùy chọn) có thể sử dụng các thư viện hỗ trợ đọc/ghi file, xử lý dữ liệu,

3.2. Tiến độ thực hiện

Link github tới dự án:https://github.com/Annsobig/CSE441_PROJECT

B1. Tạo dự án mới:

- Mở Android studio
- Chọn "Create New Project".
- Chọn "Java" làm ngôn ngữ lập trình.
- Chọn JDK phù hợp (ví dụ: JDK 11 hoặc mới hơn).
- Nhập tên dự án (ví dụ: "Appmusic").
- Chọn vị trí lưu trữ dự án.
- Nhấn "Finish" để tạo dự án.

B2. Tạo các package:

- Trong cửa sổ "Project", click chuột phải vào thư mục "src".
- Chọn "New" -> "Package".

B3. Tạo các lớp:

- Trong mỗi package, click chuột phải và chọn "New" -> "Java Class" để tạo các lớp tương ứng.
- Cài đặt các thuộc tính và phương thức cho từng lớp dựa trên thiết kế đã phân tích.

B4. Cài đặt thư viện API:

- Mở file build.gradle (nếu bạn đang sử dụng Gradle).
- Thêm dependency cho thư viện API Client Library for Java.
- Các thư viện:

- androidx.appcompat:appcompat:1.4.1:

  Thư viện hỗ trợ tương thích ngược cho các thành phần giao diện người dùng của Android, giúp đảm bảo ứng dụng hoạt động trên nhiều phiên bản Android khác nhau1.

- com.google.android.material:material:1.6.0:

Thư viện Material Design của Google, cung cấp các thành phần giao diện người dùng theo phong cách Material Design1.

- androidx.constraintlayout:constraintlayout:2.1.3:

  Thư viện cho phép tạo bố cục giao diện người dùng phức tạp và linh hoạt bằng cách sử dụng các ràng buộc.

- androidx.legacy:legacy-support-v4:1.0.0:

  Thư viện hỗ trợ tương thích ngược cho các thành phần của Android API cấp 4 trở lên.

- androidx.navigation:navigation-fragment:2.4.2:

  Thư viện hỗ trợ điều hướng giữa các fragment trong ứng dụng Android..

- androidx.navigation:navigation-ui:2.4.2:

  Thư viện hỗ trợ điều hướng giao diện người dùng, giúp quản lý các hành động điều hướng trong ứng dụng.

- androidx.annotation:annotation:1.3.0:

  Thư viện cung cấp các chú thích (annotations) giúp cải thiện chất lượng mã nguồn và hỗ trợ các công cụ phát triển.

- junit:junit:4.13.2:

  Thư viện kiểm thử đơn vị (unit testing) cho Java, giúp viết và chạy các bài kiểm thử tự động.

- androidx.test.ext:junit:1.1.3:

  Thư viện mở rộng của JUnit cho Android, hỗ trợ kiểm thử đơn vị trên nền tảng Android.

- androidx.test.espresso:espresso-core:3.4.0:

  Thư viện kiểm thử giao diện người dùng (UI testing) cho Android, giúp viết và chạy các bài kiểm thử tự động cho giao diện người dùng.

- com.android.volley:volley:1.2.1:

  Thư viện mạng của Google, giúp thực hiện các yêu cầu HTTP một cách dễ dàng và hiệu quả

-

- Các API sẽ tự động tải về và thêm thư viện vào dự án.

B5. Viết code:

- Bắt đầu viết code cho từng lớp, thực hiện các chức năng của ứng dụng:
  - **Task:** cài đặt các thuộc tính và phương thức cơ bản.
  - **TaskList:** cài đặt các phương thức để quản lý menu công việc

- **TaskView:** cài đặt các phương thức để hiển thị giao diện console và tương tác với người dùng.
- **TaskController:** cài đặt logic xử lý yêu cầu của người dùng, điều phối các lớp khác để hoàn thành các chức năng.

B6. Chạy và kiểm thử:

- Chạy ứng dụng từ Android studio bằng cách click chuột chọn **"Run"**.
- Kiểm tra các chức năng của ứng dụng, sửa lỗi và hoàn thiện code.

3.3. Hình ảnh sản phẩm

KẾT LUẬN

Ưu điểm:

App Vii Music được phát triển dựa trên nhu cầu của giới trẻ để bắt kịp thời đại công nghệ số hiện nay. App cho phép mỗi người dùng như mỗi người đóng góp, vừa nghe nhạc vừa là người đóng góp cho kho thư viện nhạc trở nên phong phú. Người dùng có thể tùy chỉnh thoải mái dữ liệu cá nhân của mình cũng như các bản nhạc. Và đặc biệt là không có quảng cáo - 1 trải nghiệm tuyệt vời

Nhược điểm:

App Vii Music được phát triển bởi đội ngũ non kinh nghiệm, chưa lành nghề nên trải nghiệm UX/UI chưa được sát với trải nghiệm người dùng. App nghe nhạc thiếu kinh phí do đó thư viện nhạc còn hạn chế cần có sự đóng góp của người dùng.

Hướng phát triển:

App Vii Music hướng tới trải nghiệm người dùng 1 cách tiện nghi nhất, sẽ là app tương thích với mọi loại thiết bị để hướng tới nhiều khách hàng nhất, đưa sản phẩm ra thị trường được nhiều người săn đón.

DANH MỤC TÀI LIỆU THAM KHẢO

[1]. Nguyễn Hồng Sơn (2007), *Giáo trình hệ thống Mạng máy tính CCNA* (Semester 1), NXB Lao động xã hội.

[2]. Phạm Quốc Hùng (2017), *Đề cương bài giảng Mạng máy tính*, Đại học SPKT Hưng Yên.

[3]. James F. Kurose and Keith W. Ross (2013), *Computer Networking: A top-down approach sixth Edition*, Pearson Education.