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 nghe nhạc Pophits (Android)

Mã SinhViên	Họ và Tên	Ngày Sinh	Lớp
2151061205	Sầm Gia Bảo	02/08/2003	63CNTT.NB
2151173775	Vũ Xuân Hoàng	1/11/2003	63KTPM1
2151062881	Hà Nhật Tiến	25/10/2003	63CNTT.NB
2151060252	Phạm Quốc Tuấn	29/08/2003	63CNTT.NB

Hà Nội, năm 2024

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 nghe nhạc Pophits (Android)

Mã SinhViên	Họ và Tên	Ngày Sinh	Điểm	
			Bằng Số	Bằng Chữ
2151061205	Sầm Gia Bảo	02/08/2003		
2151173775	Vũ Xuân Hoàng	1/11/2003		
2151062881	Hà Nhật Tiến	25/10/2003		
2151060252	Phạm Quốc Tuấn	29/08/2003		

CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2

Hà Nội, năm 2024

LỜI NÓI ĐẦU

Trong thời đại công nghệ số phát triển nhanh chóng, việc sử dụng các ứng dụng nghe nhạc trực tuyến đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày của con người. Nhằm đáp ứng nhu cầu giải trí đa dạng và tiện lợi, chúng tôi đã thực hiện dự án "Ứng dụng nghe nhạc" như một phần của môn học lập trình android với mong muốn cung cấp một nền tảng nghe nhạc đơn giản, hiệu quả và thân thiện với người dùng.

Mục tiêu của dự án là xây dựng một ứng dụng nghe nhạc hoàn chỉnh, cho phép người dùng dễ dàng tìm kiếm, phát và quản lý các bài hát yêu thích. Đồng thời, dự án còn giúp chúng tôi nắm vững hơn về các kiến thức đã học, như lập trình giao diện người dùng, xử lý âm thanh, quản lý dữ liệu và các phương pháp phát triển phần mềm hiện đại.

Lý do chúng tôi lựa chọn phát triển một ứng dụng nghe nhạc đến từ sự quan sát thực tế về nhu cầu ngày càng tăng của người dùng đối với các giải pháp giải trí trực tuyến, đồng thời mong muốn cung cấp một sản phẩm đơn giản nhưng hữu ích, phục vụ mục tiêu giải trí và giúp người dùng giảm căng thẳng sau những giờ làm việc, học tập căng thẳng. Ứng dụng này không chỉ là một công cụ giải trí, mà còn là một giải pháp công nghệ, giúp người dùng tiếp cận âm nhạc theo cách tiện lợi và hiệu quả nhất.

Chúng tôi xin gửi lời cảm ơn chân thành tới thầy Cù Tiến Dũng đã tận tình hướng dẫn và hỗ trợ chúng tôi trong suốt quá trình thực hiện dự án. Cùng với đó, chúng tôi cũng xin cảm ơn các bạn cùng lớp đã đóng góp ý kiến quý báu giúp dự án hoàn thiện hơn.

Chúng tôi rất mong nhận được sự đóng góp và phản hồi từ thầy/cô và các bạn để dự án có thể tiếp tục phát triển tốt hơn trong tương lai.

Bảng phân công công việc

Họ và tên	Công việc
Sầm Gia Bảo	• Thiết kế Fragment Trang chủ • Chức năng lọc danh sách nhạc theo top
Vũ Xuân Hoàng	• Chức tìm kiếm và hiển thị bài hát • Thiết kế Fragment Tìm kiếm • Phát nhạc
Hà Nhật Tiến	• Thiết kế chức năng gợi ý nhạc dựa trên Album • Thiết kế Fragment Thư viện
Phạm Quốc Tuấn	• Thiết kế giao diện Figma • Thiết kế khung giao diện cho App • Thiết kế Fragment Bộ sưu tập • Chức năng hiển thị nhạc từ bộ nhớ trong lên App

MỤC LỤC

MỤC LỤC HÌNH ẢNH

BẢNG CÁC TỪ VIẾT TẮT

STT	TỪ VIẾT TẮT	VIẾT ĐẦY ĐỦ
1	CSDL	Cơ sở dữ liệu
2	MVC	Model,View,Class
3	App	Application
4	JDK	Java Development Ki

CHƯƠNG 1. MÔ TẢ BÀI TOÁN

1.1. Giới thiệu

Ứng dụng nghe nhạc Pophits là một ứng dụng nghe nhạc đơn giản, cho phép người dùng nghe nhạc trực tuyến hoặc nghe nhạc đã được tải sẵn ở trong thiết bị. Ứng dụng sẽ được xây dựng bảng Java và chạy trên nền tảng Android.

1.2. Chức năng chính

Ứng dụng Nghe Nhạc cần có các chức năng sau:

- Phát nhạc: Người dùng có thể tìm kiếm và phát các bài hát từ kho nhạc có sẵn. Tính năng này hỗ trợ các thao tác cơ bản như phát, tạm dừng, tiếp tục và dừng bài hát.
- Tìm kiếm bài hát: Cho phép người dùng dễ dàng tìm kiếm bài hát yêu thích dựa trên tên bài hát, nghệ sĩ, hoặc album.
- Đề xuất nhạc dựa trên Album: Người dùng có thể được đề xuất những bản nhạc hay nhất có trong album, từ đó có thể có thêm những lựa chọn khi nghe nhạc.
- Hiển thị bài nhạc lưu trữ trong bộ nhớ trong của thiết bị: Người dùng có thể tải tài nguyên nhạc và tìm kiếm trong ứng dụng để có thể chơi nhạc.
- Chế độ phát liên tiếp và lặp lại: Cung cấp các chế độ phát nhạc đặc biệt như phát liên tiếp hoặc lặp lại danh sách phát để người dùng có trải nghiệm nghe nhạc linh hoạt hơn.

1.3. Yêu cầu phi chức năng

- Dễ sử dụng: Giao diện app cần rõ ràng, dễ hiểu và dễ sử dụng.
- Hiệu năng: Ứng dụng cần hoạt động nhanh chóng và hiệu quả, ngay cả khi có nhiều công việc.
- Độ tin cậy: Ứng dụng cần hiển thị các bài nhạc rõ rang, chơi nhạc và không để bị lỗi bài nào cả, nhạc trên máy cần được tự động tìm thấy và upload lên.

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 nghe nhạc. Họ có thể là học sinh, sinh viên, nhân viên văn phòng, hoặc bất kỳ ai có nhu cầu giải trí, thư giãn thông qua âm nhạc.

Người dùng có thể có các mức độ hiểu biết khác nhau về công nghệ, do đó ứng dụng cần đảm bảo dễ sử dụng, thân thiện với người dùng, và cung cấp trải nghiệm liền mạch trên mọi thiết bị. Điều này giúp đảm bảo rằng người dùng từ cơ bản đến nâng cao đều có thể dễ dàng thao tác và tận hưởng âm nhạc mà không gặp khó khăn.

Thu thập yêu cầu:

Dựa trên mô tả bài toán, ta đã xác định được các chức năng chính của ứng dụng Nghe Nhạc, bao gồm nghe nhạc trực tuyến, nghe nhạc có sẵn trong bộ nhớ thiết bị (Nhạc được tải xuống từ nhiều nguồn khác nhau)

Mô tả theo yêu cầu dưới dạng User Story

- Là một người dùng, tôi cần một ứng dụng nghe nhạc trực tuyến không cần đăng nhập đăng kí, có thể nghe bất cứ bài nhạc nào chỉ cần có internet
- Là một người dùng, tôi cần một ứng dụng có thể nghe nhạc ngoại tuyến thông qua những bài nhạc đã được tải xuống và lưu trữ trong bộ nhớ thiết bị của người dùng.
- Là một người dùng, tôi cần được đề xuất các bản nhạc dựa trên các album mà ứng dụng có sẵn.

Phân tích yêu cầu:

- Phát nhạc: Cho phép người dùng phát các bài hát từ thư viện nhạc có sẵn.
    - Yêu cầu: Nhạc phải được chạy khi người dùng phát nhạc
- Hiển thị nhạc có sẵn trong thiết bị lên để người dùng có thể nghe nhạc offline.
    - Yêu cầu: Người dùng phải tải nhạc, hoặc copy nhạc từ máy khác vào bộ nhớ trong để ứng dụng có thể tìm kiếm và hiển thị nhạc
- Tìm kiếm bài hát: Người dùng có thể tìm kiếm bài hát theo tên, nghệ sĩ hoặc album.
    - Yêu cầu Người dùng nhập tên bài hát cần tìm, và ấn nút tìm kiếm trên bàn phím để có thể lọc danh sách bài nhạc muốn tìm
    - Người dùng có thể chọn sort theo tên tác giả, album, bài hát và tất cả bài hát.
- Khám phá đề xuất nhạc: Ứng dụng sẽ đề xuất nhạc dựa trên album, và cung cấp ra

những bản nhạc được dề xuất.

- Yêu cầu: Phải click vào từng album nhạc nếu muốn hiển thị đề xuất nhạc

- Chế độ phát liên tiếp và lặp lại: Hỗ trợ các chế độ phát nhạc linh hoạt.
  - Yêu cầu: Khi chọn 1 trong 2 chế độ phải đáp ứng được yêu cầu của người dùng là phát liên tiếp nhạc hoặc là lặp lại 1 bài.

2.2. Thiết kế hệ thống:

2.2.1. Biểu đồ lớp cho MVC:

a) Thiết kế kiến trúc (MVC):

- **Model:**
  - Album
  - Playlist
  - Song
- **View:**
  - Fragment
    - FavoriteFragment
    - HomeFragment
    - LibraryFragment
    - SearchFragment
  - Adapter
    - SongAdapter
    - AlbumAdapter
  - MainActivity
  - MusicPlayerActivity

- **Controller:**
  - MusicPlayerController
  - SongController
  - AlbumAdapter
  - FavoriteAdaper
  - APIService

        - SongService
        - ApiService
    - BackgroundService
        - MusicService

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

- Lớp MainActivity
    - Thuộc tính (Attributes)
        - ActivityMainBinding binding: Sử dụng để liên kết giao diện người dùng (UI) với các thành phần trong file XML activity_main.xml.
        - LinearLayout media_player_layout: Lưu trữ layout của trình phát nhạc nhỏ (media player).
        - ImageButton btn_previous, btn_play_pause, btn_next: Các nút điều khiển cho trình phát nhạc:
        - btn_previous: Nút để phát bài hát trước.
        - btn_play_pause: Nút để tạm dừng hoặc tiếp tục phát bài hát.
        - btn_next: Nút để phát bài hát tiếp theo.
        - ArrayList<Song> list_song: Danh sách các bài hát hiện có.
        - int position: Chỉ mục của bài hát hiện tại trong list_song.
        - Song curent_song: Bài hát hiện tại đang phát.
        - boolean isPlaying: Biến trạng thái để kiểm tra xem nhạc có đang phát hay không (ban đầu là true).
    - Phương thức (Methods)
        - Lifecycle Methods
        - onCreate(Bundle savedInstanceState): Phương thức khởi tạo của Activity, được gọi khi Activity được tạo. Các công việc được thực hiện bao gồm:
        - Ánh xạ ActivityMainBinding với layout của Activity.
        - Kích hoạt chế độ Edge-to-Edge.
        - Thiết lập WindowInsetsCompat để điều chỉnh padding cho view.
        - Thiết lập sự kiện cho bottomNavigationView để thay thế các

fragment khi người dùng nhấn vào các mục trong navigation.

- Mặc định mở HomeFragment nếu savedInstanceState là null.
- UI Management Methods
- replaceFragment(Fragment fragment): Thay thế fragment hiện tại bằng fragment mới được cung cấp.
- Music Player Methods
- playNextSong(): Phát bài hát tiếp theo trong danh sách list_song. Nếu đến bài cuối cùng, hiển thị thông báo không thể phát tiếp.
- playPreviousSong(): Phát bài hát trước đó trong danh sách list_song. Nếu đến bài đầu tiên, hiển thị thông báo không thể lùi tiếp.
- togglePlayPause(): Chuyển đổi giữa chế độ phát và tạm dừng nhạc.
- Nếu isPlaying là true, tạm dừng nhạc, thay đổi biểu tượng nút thành play, và đặt isPlaying là false.
- Nếu isPlaying là false, bắt đầu phát nhạc, thay đổi biểu tượng nút thành pause, và đặt isPlaying là true.
- updateUIForCurrentSong(): Cập nhật giao diện trình phát nhỏ với thông tin bài hát hiện tại, bao gồm tên bài hát, nghệ sĩ, và ảnh bìa.
- Music Service Methods
- startMusicService(): Khởi động dịch vụ phát nhạc (MusicService) với URL của bài hát hiện tại.
- stopMusicService(): Dừng dịch vụ phát nhạc.

- Phương thức khác
  - onActivityResult(int requestCode, int resultCode, @Nullable Intent data): Phương thức nhận dữ liệu từ các Activity khác trả về sau khi kết thúc. Nó thực hiện các công việc sau:
  - Kiểm tra xem dữ liệu trả về có rỗng không.
  - Nếu có dữ liệu, cập nhật danh sách bài hát (list_song) và chỉ mục bài hát hiện tại (position).
  - Cập nhật giao diện trình phát nhỏ với bài hát hiện tại.
  - Thiết lập các sự kiện click cho các nút điều khiển (phát tiếp, phát trước, play/pause).

- Lớp MusicPlayerActivity
  - Thuộc tính

        - musicPlayerController: Điều khiển phát nhạc và quản lý danh sách bài hát hiện tại.
        - seekBar: Thanh điều chỉnh thời gian phát của bài hát.
        - currentTime: Hiển thị thời gian hiện tại của bài hát đang phát.
        - totalDuration: Hiển thị tổng thời lượng của bài hát.
        - songTitleView: Hiển thị tiêu đề của bài hát.
        - song_artist: Hiển thị tên nghệ sĩ của bài hát.
        - songImage: Hiển thị ảnh của bài hát.
        - statusSong: Nút điều khiển phát/tạm dừng bài hát.
        - imgview_one_song: Nút kích hoạt chế độ lặp một bài hát.
        - next_button: Nút chuyển đến bài hát tiếp theo.
        - pre_button: Nút quay lại bài hát trước đó.
        - rotateAnimator: Hiệu ứng xoay cho hình ảnh bài hát.
        - gestureDetector: Bộ nhận diện cử chỉ để xử lý các thao tác vuốt.
    - Các phương thức (Methods)
        - onCreate: Phương thức khởi tạo Activity, thiết lập các view và bộ điều khiển, gán sự kiện cho các nút bấm.
        - onTouchEvent: Xử lý sự kiện chạm trên màn hình, bao gồm thao tác vuốt.
        - updateUIForCurrentSong: Cập nhật giao diện hiển thị cho bài hát hiện tại.
        - formatTime: Định dạng thời gian từ mili-giây thành chuỗi "phút giây".
        - onStart: Đăng ký BroadcastReceiver khi Activity bắt đầu.
        - onStop: Hủy đăng ký BroadcastReceiver khi Activity dừng lại.
        - onDestroy: Gửi dữ liệu trở lại trước khi Activity bị hủy.
        - onBackPressed: Xử lý sự kiện khi người dùng nhấn nút quay lại và gửi dữ liệu về màn hình trước.
        - sendBackData: Gửi dữ liệu hiện tại của bài hát (vị trí, danh sách) về Activity gọi nó

- Lớp FavoriteFragment

- Thuộc tính (Attributes)
  - REQUEST_CODE_PERMISSION: Mã yêu cầu để xin quyền truy cập bộ nhớ.
  - recyclerViewFavorite: RecyclerView để hiển thị danh sách bài hát yêu thích.
  - favoriteAdapter: Adapter cho RecyclerView, dùng để liên kết dữ liệu bài hát yêu thích.
  - songList: Danh sách các bài hát (ArrayList<Song>) để hiển thị trong danh sách yêu thích.
  - emptyTextView: TextView để hiển thị thông báo khi danh sách bài hát yêu thích trống.
- Phương thức (Methods)
  - onCreateView: Phương thức khởi tạo View cho Fragment, thiết lập RecyclerView và Adapter, đồng thời kiểm tra và yêu cầu quyền truy cập bộ nhớ.
  - checkAndRequestPermissions: Phương thức kiểm tra và yêu cầu quyền truy cập bộ nhớ. Tùy thuộc vào phiên bản Android, phương thức này sẽ yêu cầu quyền khác nhau.
  - onRequestPermissionsResult: Xử lý kết quả khi người dùng cấp hoặc từ chối quyền truy cập bộ nhớ.
  - onActivityResult: Xử lý kết quả khi yêu cầu quyền đặc biệt trên Android 11 trở lên.
  - loadSongs: Phương thức tải danh sách bài hát từ bộ nhớ thiết bị. Phương thức này loại trừ các tệp âm thanh từ các thư mục không mong muốn (như Ringtones, Alarms, và Notifications) và thêm các bài hát vào danh sách songList.
  - getAlbumArtUri: Phương thức lấy URI của ảnh bìa album cho bài hát, dựa trên albumId

- Lớp HomeFragment
  - Thuôc tính:
    - recyclerViewTopYear - RecyclerView: Hiển thị danh sách các bài hát top của năm.
    - recyclerViewTopMonth - RecyclerView: Hiển thị danh sách các bài hát top của tháng.
    - recyclerViewTopWeek - RecyclerView: Hiển thị danh sách các bài

hát top của tuần.

- songAdapterYear - SongAdapter: Adapter cho recyclerViewTopYear.
- songAdapterMonth - SongAdapter: Adapter cho recyclerViewTopMonth.
- songAdapterWeek - SongAdapter: Adapter cho recyclerViewTopWeek.
- songListYear - List<Song>: Danh sách bài hát top của năm.
- songListMonth - List<Song>: Danh sách bài hát top của tháng.
- songListWeek - List<Song>: Danh sách bài hát top của tuần.
- apiService - ApiService: Đối tượng dùng để gọi API.
- songService - SongService: Đối tượng dùng để phân tích dữ liệu JSON từ API thành danh sách các bài hát.

- Phương thức:
  - HomeFragment() - Constructor: Khởi tạo HomeFragment, constructor mặc định bắt buộc của lớp Fragment.
  - onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) - View: Phương thức này tạo và trả về View cho Fragment. Nó khởi tạo RecyclerView, danh sách bài hát, và Adapter. Phương thức này cũng khởi tạo ApiService và SongService, và gọi các phương thức để lấy danh sách bài hát.
  - fetchTopTracksYear() - void: Phương thức này thực hiện trong một luồng riêng (Thread) để lấy danh sách các bài hát top của năm từ API. Sau khi lấy dữ liệu, nó cập nhật danh sách songListYear và gọi notifyDataSetChanged trên songAdapterYear để cập nhật RecyclerView.
  - fetchTopTracksMonth() - void: Tương tự như fetchTopTracksYear, phương thức này lấy danh sách bài hát top của tháng từ API và cập nhật songListMonth và songAdapterMonth.
  - fetchTopTracksWeek() - void: Tương tự như fetchTopTracksYear và fetchTopTracksMonth, phương thức này lấy danh sách bài hát top của tuần từ API và cập nhật songListWeek và songAdapterWeek.

- Lớp LibraryFragment
  - Thuộc tính
    - albumRecyclerView - RecyclerView: Hiển thị danh sách album

trong thư viện.

- trackRecyclerView - RecyclerView: Hiển thị danh sách bài hát trong album được chọn.
- albumAdapter - AlbumAdapter: Adapter cho albumRecyclerView để hiển thị các album.
- trackAdapter - TrackAdapter: Adapter cho trackRecyclerView để hiển thị các bài hát trong album.
- albumList - List<Album>: Danh sách chứa các album.
- trackList - List<Track>: Danh sách chứa các bài hát trong album.
- tracksTitle - TextView: Tiêu đề của danh sách bài hát, hiển thị khi có bài hát từ album được chọn.

- Phương thức
  - LibraryFragment() - Constructor: Constructor mặc định bắt buộc của lớp Fragment.
  - onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) - View: Phương thức này tạo và trả về View cho Fragment. Nó thiết lập RecyclerView cho cả album và bài hát, tạo Adapter và gọi phương thức initializeAlbumData() để tải dữ liệu album ban đầu.
  - initializeAlbumData() - void: Phương thức này khởi chạy một AsyncTask (lớp FetchAlbumsTask) để tải danh sách các album từ API.
  - FetchAlbumsTask - Lớp con của AsyncTask<Void, Void, List<Album>>: Lớp này được sử dụng để tải danh sách album từ API trong nền.
  - doInBackground(Void... voids) - List<Album>: Tải dữ liệu album từ API trong luồng nền. Phương thức này gọi fetchAlbums từ ApiService và phân tích dữ liệu JSON thành một danh sách các đối tượng Album.
  - onPostExecute(List<Album> albums) - void: Được gọi sau khi tải xong danh sách album. Nếu danh sách không rỗng, nó cập nhật danh sách album, thông báo cho albumAdapter và tự động tải danh sách bài hát từ album đầu tiên bằng cách gọi FetchTracksTask.
  - FetchTracksTask - Lớp con của AsyncTask<String, Void, List<Track>>: Lớp này được sử dụng để tải danh sách bài hát từ một album cụ thể.
  - doInBackground(String... albumIds) - List<Track>: Tải danh sách

bài hát từ API trong luồng nền dựa trên albumId được truyền vào. Phương thức này gọi fetchTracksByAlbum từ ApiService và phân tích dữ liệu JSON thành một danh sách các đối tượng Track.

- onPostExecute(List<Track> tracks) - void: Được gọi sau khi tải xong danh sách bài hát. Nếu danh sách bài hát không rỗng, nó cập nhật danh sách trackList, thông báo cho trackAdapter, và hiển thị trackRecyclerView và tracksTitle.

- Lớp SearchFragment
  - Thuộc tính
    - recyclerView: RecyclerView để hiển thị danh sách bài hát.
    - adapter: SongAdapter để quản lý dữ liệu cho RecyclerView.
    - songList: List<Song> chứa danh sách bài hát.
    - editTextSearch: EditText để người dùng nhập từ khóa tìm kiếm.
    - spinnerSearch: Spinner cho phép người dùng chọn loại tìm kiếm (ví dụ: bài hát, nghệ sĩ, album).
    - songController: SongController để quản lý các thao tác với bài hát (lấy dữ liệu bài hát).
    - isLoading: boolean để kiểm tra xem có đang tải dữ liệu hay không.
    - currentPage: int để theo dõi trang hiện tại của danh sách bài hát.
    - songsPerPage: final int xác định số bài hát cần tải mỗi lần.
    - selectedTag: String để lưu loại tìm kiếm đã chọn.
    - check_json: TextView có vẻ được định nghĩa nhưng không sử dụng trong mã nguồn.
  - Phương thức
    - onCreate(Bundle savedInstanceState): Phương thức được gọi khi fragment được tạo. Nó thường được sử dụng để khởi tạo các thành phần không phụ thuộc vào giao diện.
    - onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState): Phương thức để tạo và trả về giao diện cho fragment. Nó sử dụng LayoutInflater để làm phẳng giao diện XML thành một View.
    - fetchSongs(String query, String selectTag, int page): Phương thức lấy dữ liệu bài hát từ songController và cập nhật RecyclerView. Nó chạy trong một luồng riêng để không làm đông giao diện.

        - loadMoreSongs(): Phương thức được gọi để tải thêm bài hát khi người dùng cuộn đến cuối danh sách.
        - getRandomLetter(): Phương thức để lấy một chữ cái ngẫu nhiên, dùng để tìm bài hát ban đầu.

- Lớp AlbumAdapter
    - Thuộc tính
        - List<Album> albums: Danh sách các album cần hiển thị trong RecyclerView.
        - AlbumClickListener albumClickListener: Lắng nghe sự kiện khi người dùng nhấn vào một album, cho phép xử lý hành động khi album được chọn.
    - Phương thức
        - AlbumAdapter(List<Album> albums, AlbumClickListener albumClickListener): Constructor khởi tạo adapter với danh sách album và đối tượng lắng nghe sự kiện nhấn vào album.
        - onCreateViewHolder(@NonNull ViewGroup parent, int viewType): Khởi tạo ViewHolder cho album, ánh xạ layout album_item.xml vào RecyclerView.
        - onBindViewHolder(@NonNull ViewHolderAlbum holder, int position): Liên kết dữ liệu của album tại vị trí position trong danh sách albums vào ViewHolder. Hiển thị tên và ảnh của album, đồng thời thiết lập sự kiện nhấn vào album.
        - getItemCount(): Trả về tổng số album có trong danh sách albums.
    - Lớp con ViewHolder (Inner Class)
        - TextView albumName: Hiển thị tên của album.
        - ImageView albumImage: Hiển thị ảnh bìa của album.
        - ViewHolderAlbum(@NonNull View itemView): Constructor khởi tạo ViewHolder, ánh xạ các view thành phần như albumName và albumImage từ layout album_item.xml.
    - Giao diện lắng nghe sự kiện (Interface)
        - AlbumClickListener: Giao diện định nghĩa phương thức callback khi người dùng nhấn vào một album.
        - void onAlbumClick(String albumId): Phương thức xử lý sự kiện khi một album được nhấn, nhận albumId để xác định album nào được chọn.

- Lớp SongAdapter
    - Thuộc tính
        - List<Song> songList: Danh sách các bài hát cần hiển thị trong RecyclerView.
        - Song song: Bài hát hiện tại được hiển thị trong RecyclerView.
        - Map<String, Long> downloadMap: Bản đồ chứa thông tin về các bài hát đang tải xuống, với khóa là tên bài hát và giá trị là mã nhận diện quá trình tải xuống.
        - Context context: Ngữ cảnh của Activity hoặc Fragment nơi SongAdapter được sử dụng.
    - Phương thức
        - SongAdapter(List<Song> songList): Constructor để khởi tạo adapter với danh sách bài hát.
        - SongAdapter(Context context, List<Song> songList): Constructor để khởi tạo adapter với danh sách bài hát và ngữ cảnh hiện tại.
        - onCreateViewHolder(@NonNull ViewGroup parent, int viewType): Tạo SongViewHolder và ánh xạ layout item_song.xml vào RecyclerView.
        - onBindViewHolder(@NonNull SongViewHolder holder, int position): Liên kết dữ liệu của bài hát tại vị trí position trong songList vào SongViewHolder. Hiển thị tên bài hát, nghệ sĩ, album và ảnh bìa. Thiết lập sự kiện nhấn vào bài hát để mở MusicPlayerActivity và chuyển bài hát hiện tại.
        - getItemCount(): Trả về tổng số bài hát trong songList.
    - Lớp con DownloadFileTask (Inner Class)
        - doInBackground(String... urls): Tải tệp âm thanh từ URL và lưu vào bộ nhớ ngoài của thiết bị. Sử dụng AsyncTask để tải xuống trong nền nhằm tránh làm đông UI.
        - onPostExecute(String result): Hiển thị thông báo tải xuống sau khi hoàn thành.
    - Lớp con ViewHolder (Inner Class)
        - TextView songName: Hiển thị tên bài hát.
        - TextView artistName: Hiển thị tên nghệ sĩ.
        - TextView albumNameTextView: Hiển thị tên album của bài hát.

        - ImageView songImage: Hiển thị ảnh bìa của bài hát.
        - ImageButton img_dowload: Nút tải xuống bài hát nếu bài hát cho phép tải xuống.
        - SongViewHolder(View itemView): Constructor để khởi tạo ViewHolder và ánh xạ các thành phần UI từ layout item_song.xml.

- Lớp LibraryAdapter
    - Thuộc tính
        - List<Song> songList: Danh sách các bài hát địa phương cần hiển thị trong RecyclerView.
        - Context context: Ngữ cảnh của Activity hoặc Fragment nơi SongLocalAdapter được sử dụng.
    - Phương thức (Methods)
        - SongLocalAdapter(List<Song> songList, Context context): Constructor để khởi tạo adapter với danh sách bài hát và ngữ cảnh hiện tại.
        - onCreateViewHolder(@NonNull ViewGroup parent, int viewType): Tạo SongViewHolder và ánh xạ layout item_song.xml vào RecyclerView.
        - onBindViewHolder(@NonNull SongViewHolder holder, int position): Liên kết dữ liệu của bài hát tại vị trí position trong songList vào SongViewHolder. Hiển thị tên bài hát, nghệ sĩ, album và ảnh bìa. Thiết lập sự kiện nhấn vào bài hát để mở MusicPlayerActivity và chuyển bài hát hiện tại.
        - getItemCount(): Trả về tổng số bài hát trong songList.
    - Lớp con ViewHolder (Inner Class)
        - TextView songName: Hiển thị tên bài hát.
        - TextView artistName: Hiển thị tên nghệ sĩ.
        - TextView albumName: Hiển thị tên album của bài hát.
        - ImageView songImage: Hiển thị ảnh bìa của bài hát.
        - SongViewHolder(View itemView): Constructor để khởi tạo ViewHolder và ánh xạ các thành phần UI từ layout item_song.xml.

- Lớp SongService
    - Thuộc tính

        - ApiService apiService: Đối tượng ApiService để thực hiện các thao tác giao tiếp với API.
    - Phương thức
        - List<Song> parseJson(String json): Phương thức này phân tích dữ liệu JSON để tạo ra danh sách các đối tượng Song. Mỗi bài hát được lấy từ JSON với các thuộc tính như id, name, artistName, audioUrl, imageUrl, albumName, audiodownload, và audiodownload_allowed.
        - static List<Album> parseAlbumsJson(String json): Phương thức này phân tích dữ liệu JSON để tạo ra danh sách các đối tượng Album. Nó xử lý các thuộc tính như id, name, releaseDate, artistId, artistName, imageUrl, zipUrl, shortUrl, shareUrl, và zipAllowed.
        - static List<Track> parseTracksJson(String json): Phương thức này phân tích dữ liệu JSON để tạo ra danh sách các đối tượng Track. Mỗi Track được lấy từ JSON với các thuộc tính như trackId, position, trackName, duration, licenseUrl, audioUrl, audioDownloadUrl, và audioDownloadAllowed.

- Lớp ApiService
    - Thuộc tính
        - String BASE_URL: Chuỗi URL cơ bản của API Jamendo, được sử dụng để xây dựng các URL truy vấn API.
        - String CLIENT_ID: Chuỗi mã ID của ứng dụng khách (client_id) để xác thực yêu cầu API với Jamendo.
    - Phương thức
        - String buildSongUrl(String query, int page, int limit): Xây dựng URL truy vấn API để tìm kiếm các bài hát dựa trên từ khóa query, số trang page, và số lượng bài hát tối đa limit.
        - String fetchData(String apiUrl): Gửi yêu cầu đến API và lấy dữ liệu JSON từ URL cung cấp (apiUrl). Phương thức trả về dữ liệu JSON dạng chuỗi.
        - String fetchAlbums(int page, int limit): Gửi yêu cầu đến API để lấy danh sách album với phân trang theo page và limit. Phương thức trả về dữ liệu JSON dạng chuỗi của danh sách album.
        - String fetchTracksByAlbum(String albumId): Gửi yêu cầu đến API để lấy danh sách bài hát của một album dựa trên albumId. Phương thức trả về dữ liệu JSON dạng chuỗi của danh sách bài hát trong album.
        - String fetchTopYear(int page, int limit): Lấy danh sách các bài hát

hàng đầu của năm, với phân trang theo page và limit. Phương thức trả về dữ liệu JSON dạng chuỗi của các bài hát hàng đầu trong năm.

- String fetchTopMonth(int page, int limit): Lấy danh sách các bài hát hàng đầu của tháng, với phân trang theo page và limit. Phương thức trả về dữ liệu JSON dạng chuỗi của các bài hát hàng đầu trong tháng.
- String fetchTopWeek(int page, int limit): Lấy danh sách các bài hát hàng đầu của tuần, với phân trang theo page và limit. Phương thức trả về dữ liệu JSON dạng chuỗi của các bài hát hàng đầu trong tuần.

- Lớp MusicService
  - Thuộc tính
    - String ACTION_UPDATE_SEEKBAR: Tên hành động cho Intent được phát để cập nhật SeekBar khi vị trí phát nhạc thay đổi.
    - String EXTRA_CURRENT_POSITION: Khóa Intent chứa vị trí hiện tại của bài hát.
    - String EXTRA_DURATION: Khóa Intent chứa tổng thời lượng của bài hát.
    - String ACTION_SONG_COMPLETED: Tên hành động cho Intent được phát khi bài hát hoàn tất.
    - MediaPlayer mediaPlayer: Đối tượng phát nhạc.
    - Handler handler: Xử lý các cập nhật cho SeekBar.
    - IBinder binder: Đối tượng Binder cho phép các hoạt động khác kết nối với MusicService.
    - String audioUrl: Đường dẫn của file nhạc cần phát.
    - boolean isPaused: Xác định trạng thái của mediaPlayer (đang tạm dừng hay không).
  - Phương thức
    - public IBinder onBind(Intent intent): Trả về binder để cho phép các thành phần khác trong ứng dụng kết nối với MusicService.
    - public int onStartCommand(Intent intent, int flags, int startId): Nhận lệnh bắt đầu Service. Dựa trên các extras trong intent, phương thức này sẽ gọi playAudio(), pauseAudio(), hoặc seekTo().
    - private void playAudio(): Phát nhạc từ audioUrl. Nếu mediaPlayer chưa được khởi tạo, phương thức sẽ thiết lập mediaPlayer với nguồn phát audioUrl và bắt đầu phát nhạc. Nếu nhạc đang tạm dừng

(isPaused), phương thức sẽ tiếp tục phát và cập nhật SeekBar.

- private void pauseAudio(): Tạm dừng phát nhạc nếu mediaPlayer đang phát. Đồng thời, ngừng cập nhật SeekBar.
- private void seekTo(int position): Chuyển đến một vị trí cụ thể trong bài hát.
- private void startSeekBarUpdates(): Bắt đầu cập nhật vị trí của SeekBar mỗi giây.
- private final Runnable updateSeekBar: Runnable để cập nhật SeekBar. Gửi một Intent chứa vị trí hiện tại và thời lượng của bài hát đến Activity.
- public void onDestroy(): Giải phóng mediaPlayer và xóa tất cả các cập nhật SeekBar khi Service dừng lại.

- Lớp MusicPlayerController
  - Thuộc tính
    - Context context: Đối tượng context của Activity để tương tác với các dịch vụ hệ thống, như Toast và Service.
    - ArrayList<Song> list_song: Danh sách các bài hát mà người dùng có thể phát.
    - int position: Vị trí hiện tại của bài hát trong list_song.
    - Song current_song: Bài hát hiện tại dựa trên position.
    - boolean isPlaying: Trạng thái phát hiện tại (đang phát hoặc không).
    - boolean playOne: Xác định có đang phát một bài hát liên tục (chế độ lặp) hay không.
  - Phương thức
    - Constructor MusicPlayerController(Context, ArrayList<Song>, int position): Khởi tạo bộ điều khiển nhạc, nhận danh sách bài hát và vị trí của bài hát ban đầu.
    - Getter Methods:
      - getCurrentSong(): Trả về bài hát hiện tại.
      - isPlayOne(): Trả về trạng thái playOne.
      - getPosition(): Trả về vị trí hiện tại của bài hát.
      - getSongList(): Trả về danh sách các bài hát.
      - isPlaying(): Trả về trạng thái phát của bài hát hiện tại.

        - Phương thức điều khiển phát nhạc
            - playAudio(): Bắt đầu phát bài hát hiện tại bằng cách khởi động MusicService và gửi audioUrl của bài hát tới Service.
            - pauseAudio(): Tạm dừng bài hát đang phát bằng cách gửi tín hiệu pause tới MusicService.
            - stopAudio(): Dừng hẳn việc phát nhạc và ngừng MusicService.
        - Chuyển bài hát
            - playNextSong(): Phát bài hát tiếp theo trong danh sách nếu chưa phải bài cuối. Nếu đang phát bài cuối, hiển thị Toast thông báo.
            - playPreviousSong(): Phát bài hát trước đó nếu không phải là bài đầu. Nếu đang phát bài đầu, hiển thị Toast thông báo.
        - Chế độ lặp bài hát
            - togglePlayOne(): Thay đổi trạng thái playOne để bật/tắt chế độ lặp bài hát.

- Lớp SongController
    - Thuộc tính
        - ApiService apiService: Đối tượng của ApiService để gọi các API từ dịch vụ nhạc.
        - SongService songService: Đối tượng của SongService để phân tích dữ liệu JSON và chuyển đổi chúng thành các đối tượng Song.
    - Phương thức
        - Constructor SongController(): Khởi tạo SongController, đồng thời khởi tạo các đối tượng apiService và songService.
        - Phương thức lấy danh sách bài hát
            - getSongs(String query, String selectedTag, int page, int songsPerPage): Lấy danh sách các bài hát từ API theo từ khóa tìm kiếm (query), thẻ chọn lọc (selectedTag), và các thông tin phân trang.

c) Quan hệ giữa các lớp:

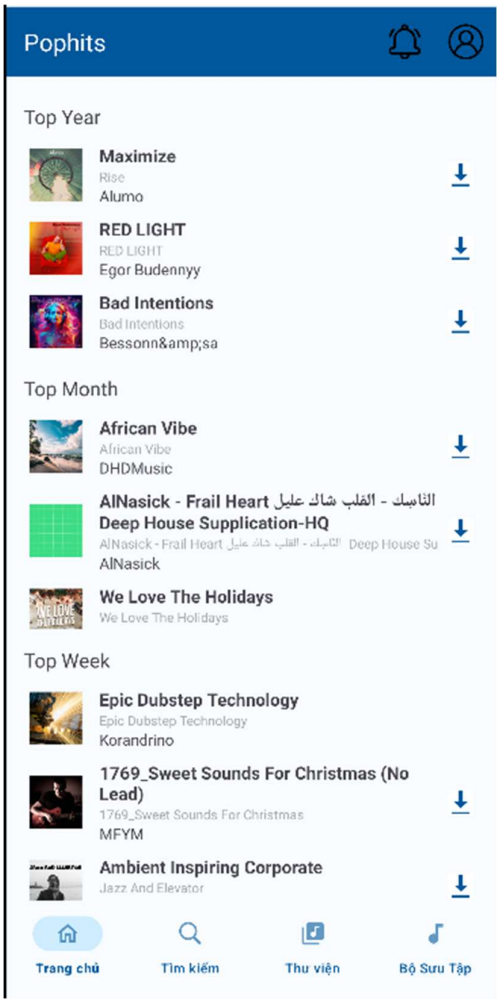
- MainActivity
    - Quản lý Fragments: MainActivity là lớp chính quản lý các fragment khác như HomeFragment, FavoriteFragment, LibraryFragment, và SearchFragment. Nó sử dụng phương thức replaceFragment(Fragment fragment) để chuyển đổi giữa các fragment khi người dùng nhấn vào các mục trong thanh điều hướng.
    - Tương tác với MusicPlayerActivity: MainActivity có thể tương tác với MusicPlayerActivity để phát nhạc, điều này có thể thông qua onActivityResult khi nhận dữ liệu từ MusicPlayerActivity.
- MusicPlayerActivity
    - Quản lý phát nhạc: MusicPlayerActivity điều khiển phát nhạc, sử dụng lớp MusicPlayerController để quản lý danh sách bài hát và tương tác với các thành phần như seekBar, songTitleView, và statusSong để cập nhật giao diện người dùng.
    - Tương tác với MainActivity: Khi người dùng hoàn tất các hành động trong MusicPlayerActivity, nó gửi dữ liệu trở lại MainActivity thông qua sendBackData.
- FavoriteFragment
    - Hiển thị danh sách yêu thích: FavoriteFragment sử dụng RecyclerView để hiển thị danh sách bài hát yêu thích và sử dụng favoriteAdapter để quản lý dữ liệu. Nó có thể lấy danh sách bài hát từ MainActivity hoặc một lớp quản lý dữ liệu khác.
    - Tương tác với Media Storage: FavoriteFragment yêu cầu quyền truy cập bộ nhớ để tải danh sách bài hát yêu thích từ thiết bị.
- HomeFragment
    - Lấy dữ liệu từ API: HomeFragment gọi các phương thức từ ApiService và SongService để lấy danh sách bài hát top theo năm, tháng, và tuần. Sau đó, nó cập nhật RecyclerView với dữ liệu mới.
    - Tương tác với MainActivity: HomeFragment có thể tương tác với MainActivity để điều hướng đến các fragment khác hoặc cập nhật giao diện.
- LibraryFragment
    - Quản lý danh sách album và bài hát: LibraryFragment quản lý RecyclerView cho album và bài hát, sử dụng albumAdapter và trackAdapter. Nó có thể tải dữ liệu album từ API thông qua ApiService.
    - Tương tác với Album và Track: Các lớp Album và Track có thể được sử

dụng để lưu trữ thông tin về album và bài hát mà LibraryFragment hiển thị.

- SearchFragment
    - Tìm kiếm bài hát: SearchFragment cho phép người dùng nhập từ khóa tìm kiếm và chọn loại tìm kiếm. Nó sử dụng songController để quản lý các thao tác với bài hát và cập nhật RecyclerView với danh sách bài hát phù hợp.
    - Tương tác với MainActivity: Giống như các fragment khác, SearchFragment cũng có thể tương tác với MainActivity để điều hướng hoặc cập nhật dữ liệu.

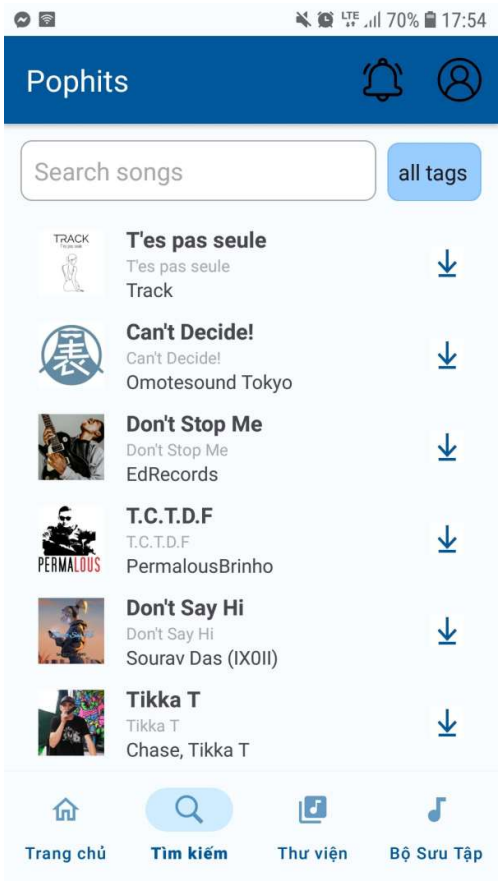
2.2.2 Thiết kế giao diện:

- Hiển thị tab chính là HomeFragment – Đưa ra các lựa chọn về các bài hát top ngày tuần tháng.



Ảnh 1: Giao diện trang chủ

- Giao diện tìm kiếm.

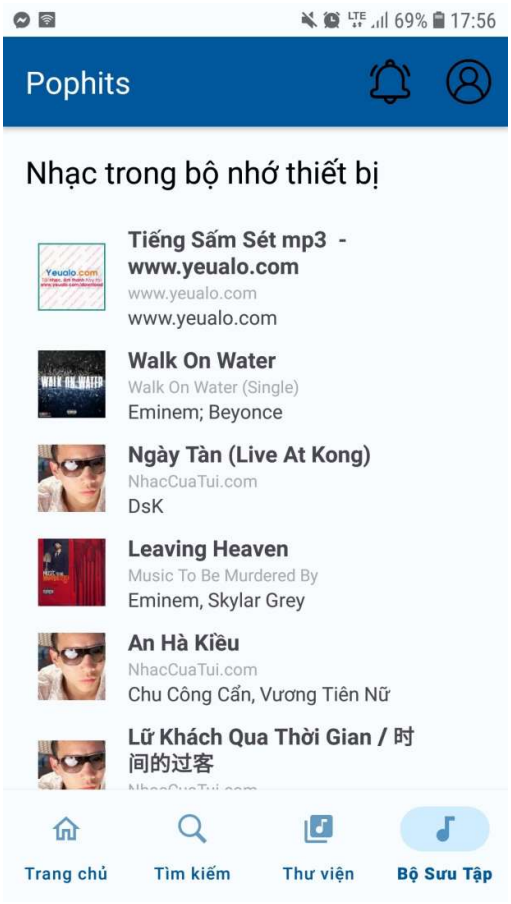


Ảnh 2: Giao diện tìm kiếm

- Giao diện thư viện(album)

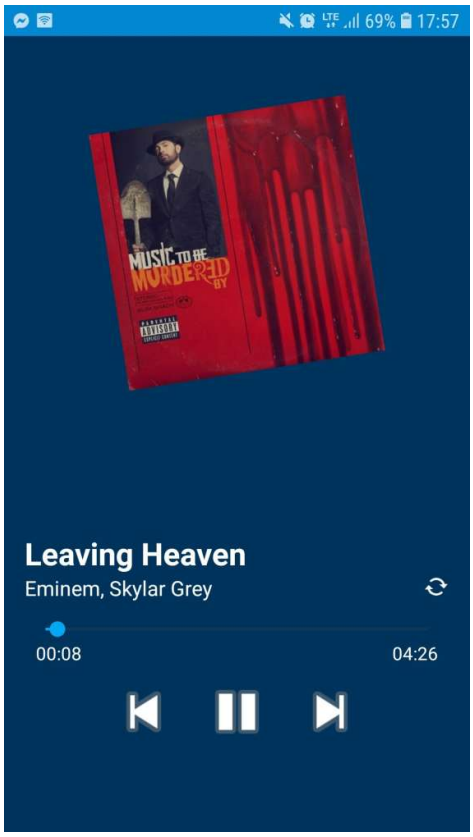
Ảnh 3: Giao diện thư viện

- Giao diện bộ sự tập



Ảnh 4: Giao diện bộ sưu tập

- Giao diện phát nhạc



Ảnh 5: Giao diện phát nhạc

2.3. Triển khai:

- Viết code: Chia 4 fragment lớn cho từng thành viên trong team, các chức năng nhỏ đươc phân công cụ thể.
- Kiểm thử: Kiểm tra và chạy phần mềm nghe nhạc, tìm lỗi.

2.4. Vận hành và bảo trì:

- Cài đặt và triển khai:
- 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: IntelliJ IDEA
- Thư viện:
    - Retrofit (com.squareup.retrofit2:retrofit:2.9.0)
        - Tác dụng: Retrofit là một thư viện HTTP client cho Android và Java, giúp dễ dàng giao tiếp với các API RESTful. Nó hỗ trợ tạo các yêu cầu mạng, quản lý phản hồi, và giúp chuyển đổi dữ liệu từ JSON sang các đối tượng Java một cách đơn giản.
    - Gson Converter (com.squareup.retrofit2:converter-gson:2.9.0)
        - Tác dụng: Đây là một converter cho Retrofit, cho phép tự động chuyển đổi dữ liệu JSON thành các đối tượng Java thông qua thư viện Gson. Điều này giúp giảm bớt công sức trong việc phân tích cú pháp và chuyển đổi dữ liệu.
    - Gson (com.google.code.gson:gson:2.10.1)
        - Tác dụng: Gson là một thư viện dùng để chuyển đổi giữa JSON và các đối tượng Java. Nó cung cấp các phương thức để phân tích cú pháp JSON và chuyển đổi các đối tượng Java thành định dạng JSON, hỗ trợ cho việc truyền tải và nhận dữ liệu giữa ứng dụng và API.
    - Glide (com.github.bumptech.glide:glide:4.15.1)
        - Tác dụng: Glide là một thư viện giúp tải, cache và hiển thị hình ảnh trong ứng dụng Android. Nó rất hiệu quả trong việc xử lý hình ảnh từ các nguồn như URL, giúp giảm thiểu độ trễ và sử dụng băng thông khi tải hình ảnh.
    - Glide Compiler (com.github.bumptech.glide:compiler:4.15.1)
        - Tác dụng: Đây là một annotation processor cho Glide, giúp tự động tạo ra mã nguồn cần thiết để Glide hoạt động hiệu quả. Nó thường được sử dụng để tối ưu hóa việc tải hình ảnh và cải thiện hiệu suất trong quá trình biên dịch.

3.2. Tiến độ thực hiện

Link github tới dự án: https://github.com/hanhattien2003/CSE441_MUSIC

Hướng dẫn các bước đã thực hiện:

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: "CSE441_MUSIC").
- Chọn vị trí lưu trữ dự án.
- Chọn Theme muốn sử dụng
- Nhấn "Finish" để tạo dự án.

B2. Tạo các package:

- Trong cửa sổ "Project", click chuột phải vào thư mục "src".
- Chọn "New" ->"Package".
- Tạo các package sau:
    - model: chứa các lớp Album, Song.
    - Fragment: chứa lớp AlbumFragment, HomeFragment, LibraryFragment, SearchFragment.
    - Adapter: chứa các lớp SongAdapter, LibraryAdapter, AlbumAdapter.
    - controller: chứa lớp SongController, MusicPlayerController.
    - APIService: chứa các lớp ApiService, SongService
    - BackgroundService: chứa lớp MusicService.

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. Đăng ký sử dụng Jamendo API

B5. Cài đặt các thư viện:

- Mở file build.gradle.
- Thêm dependency:

- implementation ("com.squareup.retrofit2:retrofit:2.9.0")
- implementation ("com.squareup.retrofit2:converter-gson:2.9.0")
- implementation ("com.google.code.gson:gson:2.10.1")
- implementation ("com.github.bumptech.glide:glide:4.15.1")
- annotationProcessor ("com.github.bumptech.glide:compiler:4.15.1")

B6. Viết code

B7. Chạy và kiểm thử

KẾT LUẬN

Ưu điểm

Ứng dụng nghe nhạc đơn giản mang lại cho người dùng trải nghiệm nghe nhạc trực tuyến và ngoại tuyến tiện lợi và hiệu quả. Một trong những ưu điểm nổi bật của ứng dụng là tính năng tìm kiếm bài hát, nghệ sĩ hoặc album dễ dàng, cho phép người dùng nhanh chóng tìm thấy nội dung âm nhạc mà họ yêu thích. Ứng dụng cũng cung cấp danh sách danh sách nhạc độc đáo, hướng người dùng trải nghiệm âm nhạc đặc biệt, độc nhất. Hơn nữa, giao diện người dùng được thiết kế thân thiện, đơn giản và dễ sử dụng, giúp người dùng không mất nhiều thời gian để làm quen và sử dụng các chức năng của ứng dụng.

Nhược điểm

Dù có nhiều ưu điểm, ứng dụng nghe nhạc đơn giản vẫn gặp một số nhược điểm. Một trong số đó là hiệu suất chưa tối ưu; khi xử lý danh sách bài hát lớn hoặc phát nhạc có chất lượng cao, ứng dụng có thể gặp tình trạng lag, ảnh hưởng đến trải nghiệm nghe nhạc. Ngoài ra, ứng dụng có thể thiếu một số tính năng cao cấp như chế độ xem lời bài hát, điều chỉnh tốc độ phát và điều chỉnh chất lượng âm thanh, điều này có thể làm giảm sự hấp dẫn đối với người dùng có nhu cầu cao hơn.

Hướng phát triển chủ đề

Để nâng cao giá trị và khả năng cạnh tranh của ứng dụng nghe nhạc đơn giản, các nhà phát triển nên tập trung vào việc tối ưu hóa hiệu suất và cải thiện trải nghiệm người dùng. Ngoài ra, phát triển các tính năng tương tác như bình luận, đánh giá bài hát hoặc tạo playlist chung với bạn bè sẽ tạo ra một môi trường cộng đồng sôi nổi và hấp dẫn hơn. Cuối cùng, tích hợp các tùy chọn tùy chỉnh như chế độ phát ngẫu nhiên, lặp lại bài hát, và phát nhạc theo tâm trạng sẽ mang lại trải nghiệm cá nhân hóa cho người dùng, giúp họ cảm thấy thoải mái và hài lòng hơn khi sử dụng ứng dụng..

DANH MỤC TÀI LIỆU THAM KHẢO

[1]. https://developer.jamendo.com/v3.0/docs

[2]. https://www.rcsdk12.org/cms/lib/NY01001156/Centricity/Domain/4951/Head_First_Java_Second_Edition.pdf

[3]. Margaret Kozak Polk , *Coding Android Apps*

[4]. Chat GPT by OpenAI

[5]. Gemini AI by Google

[6]. Copilot by Microsoft

PHỤ LỤC

- Các quyền cần được cấp khi dùng app: sử dụng internet và truy cập bộ nhớ.
- File Android Manifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">



<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission
android:name="android.permission.MANAGE_EXTERNAL_STORAGE"
        tools:targetApi="30" />


<application
        android:allowBackup="true"

android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.CSE441_MUSIC"
        android:enableOnBackInvokedCallback="true"
        tools:targetApi="31">
```

```
<!--        service-->
<service android:name=".BackgroundService.MusicService" />
<!--        service-->

<activity
            android:name=".MusicPlayerActivity"

android:configChanges="orientation|keyboardHidden|screenSize"
            android:exported="false"

android:label="@string/title_activity_music_player"

android:theme="@style/Theme.CSE441_MUSIC.Fullscreen" />
<activity
            android:name=".MainActivity"
            android:exported="true">
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>

</manifest>
```

- Mỗi phương thức của API Jamendo được diễn đạt một cách tổng quát dưới dạng thực thể hoặc thực thể+các thực thể con. Dạng URL GET tổng quát là như sau: http[s]://api.jamendo.com/<version>/<entity>/<subentity>/?<api_parameter>=<value>. Vì một số thông số kỹ thuật RESTful (như các hành động CRUD ánh xạ 1-1 qua các phương thức HTTP, hoặc ID thực thể nhận được dưới dạng tham số GET) không được triển khai trong frameworki, nên API ưu tiên được định nghĩalà

RESTlike.

- File gửi yêu cầu cho api và nhận về kết quả

```
package com.example.cse441_music.APIService;


import android.util.Log;


import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;


public class ApiService {
    private static final String BASE_URL = "https://api.jamendo.com/v3.0";
    public static final String CLIENT_ID = "900dafad";


    public String buildSongUrl(String query, int page, int limit) {
        StringBuilder apiUrl = new StringBuilder(BASE_URL + "/tracks/?client_id=" + CLIENT_ID);
        apiUrl.append("&limit=").append(limit);
        apiUrl.append("&offset=").append(page * limit);


        if (!query.isEmpty()) {
            apiUrl.append("&search=").append(query);
        }


        return apiUrl.toString();
```

```
    }


    public String fetchData(String apiUrl) throws Exception {
        URL url = new URL(apiUrl);
        HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();
        InputStream inputStream = new BufferedInputStream(urlConnection.getInputStream());
        BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));


        StringBuilder result = new StringBuilder();
        String line;
        while ((line = reader.readLine()) != null) {
            result.append(line);
        }


        reader.close();
        return result.toString();
    }



    // Phương thức lấy danh sách album với phân trang
    public String fetchAlbums(int page, int limit) throws Exception {
        String apiUrl = BASE_URL + "/albums/?client_id=" + CLIENT_ID + "&format=jsonpretty&limit=" + limit + "&offset=" + (page * limit);
        Log.d("API URL", apiUrl);
        return fetchData(apiUrl);
    }
```

```
    // Phương thức lấy bài hát theo album
    public String fetchTracksByAlbum(String albumId) throws Exception {
        String apiUrl = BASE_URL + "/albums/tracks/?client_id=" + CLIENT_ID + "&format=jsonpretty&id=" + albumId;
        return fetchData(apiUrl);
    }

    // Fetch top songs of the year
    public String fetchTopYear(int page, int limit) throws Exception {
        String apiUrl = BASE_URL + "/tracks/?client_id=" + CLIENT_ID + "&limit=" + limit + "&offset=" + (page * limit) + "&datebetween=2000-01-01_2023-12-31";
        return fetchData(apiUrl);
    }

    // Fetch top songs of the month
    public String fetchTopMonth(int page, int limit) throws Exception {
        String apiUrl = BASE_URL + "/tracks/?client_id=" + CLIENT_ID + "&limit=" + limit + "&offset=" + (page * limit) + "&datebetween=2023-10-01_2023-10-31";
        return fetchData(apiUrl);
    }

    // Fetch top songs of the week
    public String fetchTopWeek(int page, int limit) throws Exception {
```

```
        String apiUrl = BASE_URL + "/tracks/?client_id=" +
CLIENT_ID + "&limit=" + limit + "&offset=" + (page * limit) +
"&datebetween=2023-10-20_2023-10-27";
        return fetchData(apiUrl);
    }

    // Bạn có thể thêm các phương thức API khác nếu cần
}
```

- File chuyển kiểu dữ liệu trả về từ json sang list

```
package com.example.cse441_music.APIService.Song;

import com.example.cse441_music.APIService.ApiService;
import com.example.cse441_music.Model.Album;
import com.example.cse441_music.Model.Song;
import com.example.cse441_music.Model.Track;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import java.util.ArrayList;
import java.util.List;

public class SongService {
    private ApiService apiService;

    public List<Song> parseJson(String json) throws
JSONException {
        List<Song> songList = new ArrayList<>();
```

```
        JSONObject jsonObject = new JSONObject(json);
        JSONArray results = jsonObject.getJSONArray("results");

        for (int i = 0; i < results.length(); i++) {
            JSONObject track = results.getJSONObject(i);
            String id = track.getString("id");
            String name = track.getString("name");
            String artistName = track.getString("artist_name");
            String audioUrl = track.getString("audio");
            String imageUrl = track.getString("image");
            String albumName = track.optString("album_name");
            String audiodownload = track.optString("audiodownload", "");
            int audiodownload_allowed = track.optBoolean("audiodownload_allowed", false) ? 1 : 0;

            songList.add(new Song(id, name, artistName, audioUrl, imageUrl, albumName, audiodownload, audiodownload_allowed));
        }

        return songList;
    }

    public static List<Album> parseAlbumsJson(String json) throws Exception {
        JSONObject root = new JSONObject(json);
        JSONArray resultsArray = root.getJSONArray("results");
```

```
        List<Album> albums = new ArrayList<>();


        for (int i = 0; i < resultsArray.length(); i++) {
            JSONObject              albumObject              =
resultsArray.getJSONObject(i);


            String id = albumObject.getString("id");
            String name = albumObject.getString("name");
            String                releaseDate                =
albumObject.getString("releasedate");
            String                  artistId                 =
albumObject.getString("artist_id");
            String                artistName                 =
albumObject.getString("artist_name");
            String imageUrl = albumObject.getString("image");
            String zipUrl = albumObject.getString("zip");
            String                  shortUrl                 =
albumObject.getString("shorturl");
            String                  shareUrl                 =
albumObject.getString("shareurl");
            boolean                zipAllowed                =
albumObject.getBoolean("zip_allowed");


            Album album = new Album(id, name, releaseDate,
artistId, artistName, imageUrl, zipUrl, shortUrl, shareUrl,
zipAllowed);
            albums.add(album);
        }


        return albums;
    }
```

```
    public static List<Track> parseTracksJson(String json) throws Exception {
        JSONObject root = new JSONObject(json);
        JSONArray resultsArray = root.getJSONArray("results");

        // Assuming each album in the response contains a "tracks" array
        List<Track> tracks = new ArrayList<>();
        if (resultsArray.length() > 0) {
            JSONObject albumObject = resultsArray.getJSONObject(0); // First album object
            JSONArray tracksArray = albumObject.getJSONArray("tracks");

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

                String trackId = trackObject.getString("id");
                int position = trackObject.getInt("position");
                String trackName = trackObject.getString("name");
                int duration = trackObject.getInt("duration");
                String licenseUrl = trackObject.getString("license_ccurl");
                String audioUrl = trackObject.getString("audio");
                String audioDownloadUrl = trackObject.getString("audiodownload");
                boolean audioDownloadAllowed = trackObject.getBoolean("audiodownload_allowed");
```

```
                Track track = new Track(trackId, position,
trackName, duration, licenseUrl, audioUrl, audioDownloadUrl,
audioDownloadAllowed);
                tracks.add(track);
            }
        }

        return tracks;
    }

}
```