## Victor Daniel Bucur - 20211235

## Mobile Development Application

## My Music Application

GitHub Link: https://github.com/LiverpoolHopeUni/codingproject-Bvd88

**Purpose Statement**

The purpose of AltMusicPlayer is to provide users with an alternative music player that combines aesthetic appeal with functional excellence. The application allows users to enjoy their music library in a visually engaging environment with easy navigation and interactive controls. Another reason when I built this is because I wanted an alternative audio player, that doesn’t need internet, or bomb us with ads.

**Introduction**

AltMusicPlayer is an Android-based application developed in Kotlin, designed to offer a unique listening experience. The app features a sophisticated design with a rich, user-friendly interface including a welcome screen that smoothly transitions to a comprehensive music player interface. Users can play music from a predefined list or browse their device for audio files.

**Technical Requirements**

* **Android Studio Version**: Arctic Fox 2021.1.1 Patch 3
* **Programming Language**: Kotlin
* **Minimum SDK Version**: 21
* **Target SDK Version**: 30
* **Dependencies**:
  + **AndroidX Libraries**: Provides backward compatibility and additional support for newer Android features.
  + **Material Components for Android**: Utilized for modern UI elements such as buttons and image buttons.
  + **MediaPlayer API**: For media playback functionalities.
  + **ObjectAnimator**: For animating UI components like the vinyl record rotation.
  + **ActivityResultContracts**: Used for handling results from activities, especially for file browsing.

**User Interface UI**

O imagine care conține electronice, text, gadget, multimedia

Descriere generată automat**Screens**

1. **Welcome Screen (activity\_welcome.xml)**: Features a large welcome text and a START button that navigates to the main music player screen. The design uses RelativeLayout for flexibility in layout management.

O imagine care conține electronice, text, gadget, Dispozitiv electronic

Descriere generată automat

1. **Music Player Screen (activity\_main.xml)**: This screen includes:
   * **ImageView** for the vinyl record animation.
   * **TextView** for displaying the current track.
   * **SeekBar** for tracking and controlling playback.
   * **ListView** for displaying upcoming tracks.
   * ImageButtons (**play**, **next**, **previous**, **shuffle**, and **repeat**) that control playback and track navigation.

**Description**

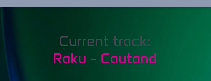
When you launch the app for the first time, you are greeted by a Welcome screen featuring the Hope University logo and a welcoming message, "Welcome to my Music Player." Below this text is a button which, when pressed, directs users to the main player interface. Upon accessing the main page, the current track's title is prominently displayed at the top of the screen. If a song is loaded and played, the track's name is dynamically fetched from the song's metadata.

Just beneath the track information, there is a visually engaging vinyl record graphic, animated using ObjectAnimator, which rotates whenever a song is played. Directly below the vinyl animation is a Seek Bar that offers full control over the song playback, allowing users to skip sections or replay their favourite parts.

Further down, a LinearLayout hosts five media control buttons (play, next, previous, shuffle, and repeat), providing comprehensive control over the playlist. Additionally, there's a preloaded playlist available for immediate listening. For those who prefer their own music, a "BROWSE Files" button allows users to navigate their external storage to select and play downloaded songs.

**Detailed Technical Overview**

**Metadata Extraction**:



The application utilizes the **MediaMetadataRetriever** class to extract metadata from audio files. This feature is integral to the functionality of displaying track names dynamically. When a user plays a song, the app retrieves and displays the song title from its metadata, which could include additional details such as artist and album if desired.

Specifically, when a new audio file is loaded via the "BROWSE Files" button, **MediaMetadataRetriever** is used to access the file's metadata and extract the title using **METADATA\_KEY\_TITLE**. If the title is not available, a default placeholder "Unknown Title" is used. This process ensures that users always have contextual information about the tracks they are playing.

O imagine care conține cerc, Dispozitiv de stocare date, disc compact, Disc de vinil

Descriere generată automatO imagine care conține cerc, disc compact, Dispozitiv de stocare date, Disc de vinil

Descriere generată automat**Object Animator**:

O imagine care conține cerc, disc compact, Dispozitiv de stocare date, Disc de vinil

Descriere generată automat

The app features a vinyl record animation on the main player screen, which serves as a dynamic visual cue that a song is playing. This animation is driven by **ObjectAnimator**, configured to rotate the vinyl record image.

The **ObjectAnimator** is set up to rotate the ImageView representing the vinyl record from 0 to 360 degrees continuously. This animation uses a **LinearInterpolator** for smooth rotation and is set to repeat indefinitely while a track is playing. The rotation is paused when playback stops, creating a realistic and responsive user experience that visually represents the playback status.

The control of the vinyl rotation is directly tied to the play and pause functionalities, aligning the visual effects with the actual playback state of the media player.

**SeekBar Functionality**:

The SeekBar allows users to visually and interactively manage playback. Users can drag the seek bar to any point in the song, effectively seeking to specific parts of the audio track.

The SeekBar's progress is updated in real-time with the song playback, thanks to a scheduled task that updates its position every second to reflect the current playback position of the MediaPlayer.

User interactions with the SeekBar (such as dragging the handle) directly adjust the playback position of the MediaPlayer, demonstrating a tight integration between the UI and the underlying audio playback system.

**Play/Pause and Other Media Control Buttons** functionality:



Play/Pause Button: This button toggles the playback state of the MediaPlayer. When the user taps 'play', the MediaPlayer starts playing the current track, and the button icon changes to 'pause'. Tapping 'pause' stops the playback, and the icon reverts to 'play'. This provides a clear visual cue about the playback status.

Next and Previous Buttons: These buttons allow users to navigate through the song list. Tapping 'next' advances to the next track in the playlist, and 'previous' returns to the prior song. These actions automatically update the displayed track information and reset the SeekBar to reflect the new track's duration.

Shuffle and Repeat Buttons: The shuffle button randomizes the order of tracks in the playlist, while the repeat button toggles the repeat mode. These features enhance user control over how they listen to their music, providing a personalized listening experience.

The play/pause functionality is controlled through conditional statements that check the current state of the MediaPlayer. If the media is not playing, the MediaPlayer is commanded to start, and vice versa.

Each button is linked to a specific function in the MediaPlayer API. For example, the 'next' button calls **MediaPlayer.create()** with the next track in the array, which is managed by updating the **currentSongIndex**.

These buttons utilize **ImageButton** in the layout, styled with custom drawables for each state (play, pause, next, previous, shuffle, repeat), ensuring the UI is intuitive and responsive.

**"BROWSE Files" Button functionality.**

The "BROWSE Files" button allows users to select audio files from their device's external storage. This expands the user’s ability to play music beyond the preloaded playlist, giving them access to their entire music library.

Once a file is selected, the app retrieves the file's URI and uses it to initialize the MediaPlayer with that specific audio track. This integration ensures that any file format supported by the MediaPlayer can be played.

O imagine care conține gadget, captură de ecran, Dispozitiv de comunicare, multimedia

Descriere generată automatThe app also handles errors and exceptions gracefully, such as unsupported file formats or read errors, ensuring the app remains stable and responsive even in case of incorrect user inputs.

O imagine care conține text, captură de ecran, Font, Grafică

Descriere generată automat