MP3-Music Player Application Development

BY

TAN JIA YI

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION SYSTEMS (HONOURS) INFORMATION SYSTEMS ENGINEERING

Faculty of Information and Communication Technology

(Kampar Campus)

JUN 2022

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: __MP3 - MUSIC PLAYER APPLICATION DEVELOPMENT _____

_____

_____

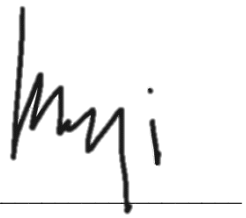
Academic Session: ___JUN 2022_____

I ___TAN JIA YI_____

(CAPITAL LETTER)

declare that I allow this Final Year Project Report to be kept in
Universiti Tunku Abdul Rahman Library subject to the regulations as follows:

1. The dissertation is a property of the Library.
2. The Library is allowed to make copies of this dissertation for academic purposes.



_____
(Author's signature)

Verified by,

Ooi Chek Yee

_____
(Supervisor's signature)

Address:
_22, LORONG SAMBAU 15B,
_TAMAN CHI LIUNG_____
41200 KLANG, SELANGOR

_____Ooi Chek Yee
Supervisor's name

Date: __11/7/2022_____

Date: _1/9/2022

FACULTY/INSTITUTE* OF __INFORMATION AND COMMUNICATION TECHNOLOGY_

UNIVERSITI TUNKU ABDUL RAHMAN

Date: _11/7/2022_____

SUBMISSION OF FINAL YEAR PROJECT /DISSERTATION/THESIS

It is hereby certified that _____***TAN JIA YI***_____ (ID No: _***19ACB05612***___) has completed this final year project/ dissertation/ thesis* entitled "*MP3 - MUSIC PLAYER APPLICATION DEVELOPMENT*_____" under the supervision of __TS DR. OOI CHEK YEE_____ (Supervisor) from the Department of _COMPUTER AND COMMUNICATION TECHNOLOGY__, Faculty/Institute* of _INFORMATION AND COMMUNICATION TECHNOLOGY___ , and _TS. LIM JIT THEAM_____ (Co-Supervisor)* from the Department of _DIGITAL ECONOMY TECHNOLOGY__, Faculty/Institute* of _INFORMATION AND COMMUNICATION TECHNOLOGY___.

I understand that University will upload softcopy of my final year project / dissertation/ thesis* in pdf format into UTAR Institutional Repository, which may be made accessible to UTAR community and public.

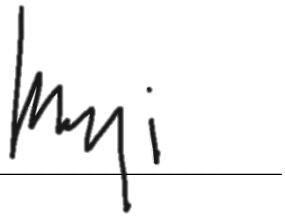
Yours truly,

__TAN JIA YI_____
(*Student Name*)

*Delete whichever not applicable

DECLARATION OF ORIGINALITY

I declare that this report entitled "**MP3 - MUSIC PLAYER APPLICATION DEVELOPMENT USING ANDROID**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : 

Name : TAN JIA YI

Date : 11/7/2022

ACKNOWLEDGEMENTS

I would like to express thanks and appreciation to my supervisor, Dr. Ooi Chek Yee and my moderator, Ts. Lim Jit Theam who have given me a golden opportunity to involve on the an MP3-Music Player Application Development project. Besides that, they have given me a lot of guidance in order to complete this project. When I was facing problems in this project, the advice from them always assists me in overcoming the problems. Again, a million thanks to my supervisor and moderator.

To a very special person in my life, Chin Hiew Yan and Wong Hew Tong, for their patience, unconditional support, and love, and for standing by my side during hard times. Finally, I must say thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

ABSTRACT

This project is an MP3 Music Player design project for academic purpose. It will provide the methodology, concept and design of a music player. This project is about developing an MP3 music player application for Android. The most significant distinction between the music player and other MP3 application is that it is absolutely free to use. It will incorporate the benefits of existing music players on the market, mining out as many functions as possible from existing music players, and then filtering to exclude functions that are not practical or pricy. It will also be enhanced depending on user comments. Audio is a crucial mode of communication, equally as significant as text nowadays. As we all know that the audio files are digital data, so we'll need a tool to execute the digital files, or to play the files. We can never listen to music, movies, or the contents of an audio file without this tool or player. As a result, we require music players. This gadget allows us to listen to music and other digital audio files. We can accomplish that without downloading and installing paid music players. The music player GUI project concept attempts to simulate a physical music player. This software enables us to listen to songs, music, and all music files on our mobile phone. The music player application must be capable of playing a song, creating, and displaying a playlist, pausing and restarting a long song, changing the song, and play the previous or subsequent song. The agile development cycle was employed to create the mp3 music application for this project. The agile development cycle contained six phases which is requirements elicitation, planning, design, development and implementation, testing, and deployment. Because of the iterative and flexible character of this technique, it can efficiently adapt to changing user requirements.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

APK	Android Application Package
apps	Applications
FYP	Final Year Project
GB	Gigabyte
IOS	iPhone Operating System
MP3	MPEG 1 Audio Layer 3
MV	Music Video
UI	User interface
VIP	Very Important Person
XML	Full Extensible Markup Language

Chapter 1
Introduction

One of the quotes quoted by an American writer, Kahlil Gibran used to say that “Music is the language of the spirit. It opens the secret of life bringing peace, abolishing strife.” Music has been an integral part of human culture from the dawn of time, a world without music or melody would be absolutely empty. Especially in today’s society, because we live in a fast-paced culture, stress is a norm in our lives. So, music has become one of the important components of our lives. People nowadays really need a space to release their stress and temporary escape from reality. Then music is the best way for them to calm and relax. A researcher from Stanford University has indicated that “listening to music seems to be able to change brain functioning to the same extent as medication.” They found that almost everyone has easy access to music, which makes it an effective stress reduction strategy. (Emily Saarman, 2006) As we have seen, people nowadays no matter on the bus, subway or gym they would like to listen to music in order to enjoy “me time” or chill themselves. Therefore, a great functioning MP3 music player application is needed, and I would like to propose this title as my FYP project to improve and develop a better MP3 music player application to obtain a better user experience. Why do we use the MP3 format? The MP3 audio format has been adopted by the market and it has become the main format for streaming music as it is extensively supported on multiple platforms. Although MP3 is a lossy file format that affects the quality of sound, it required lesser space compared to other file formats which are suitable for storing music on smartphones and other music devices, such as iPods. A few years after the invention of the smartphone, many mobile applications appeared one another, and MP3 music player applications are not an exception. There are some popular music player applications in the market such as JOOX, Spotify and etc., which is quite outstanding compared to other similar applications. However, there are still many aspects that can be improved in terms of functions, interface design and so on. So, this report will review and study the applications of some MP3 music players, point out the existing problems, and propose the expected innovations.

1.1 Problem Statement and Motivation

1.1.1 Problem Statement

i. Restricted Only for Listening to Music

Due to the needs and demands of different users, a music player application cannot come just for listening to music. This will make the user get bored and choose another app with other special features. Just listening and controlling music, the app can't catch on and compete with those popular and famous music apps.

ii. Limited Only on Button Control

As we all know, almost every music player application uses play/pause, skip/previous and shuffle/loop buttons to operate the music player. Although this is a common setting in every music player, it can cause inconvenience and danger to users. This can be said because when the user wants to skip or play/pause a song, they need to swipe the focus to the music player to press the exact button at the exact spot. When the user is driving, distraction may result, and accidents may occur. According to Suzanne Scacca, gestures might make more sense, as they don’t always such precise touch-precision. Let's take Instagram as an example. Users can like, comment on, share, or bookmark posts by clicking on the corresponding buttons. However, as you can see, there isn't a lot of space between options, especially for users with larger fingers. So, Instagram has adopted a double-tap gesture for likes. In this way, the user can interact with the image or video by simply double tapping the image or video somewhere with their finger. (Suzanne, 2020)

iii. Song Libraries Are Limited

There is a lot of different genres of songs we can easily find online these days. Because online songs are scattered and difficult to concentrate on, users wish to gather all the songs in the same place for convenience, and they can listen to the songs they want as soon as they open the applications. But in some music player applications, there is still the problem of a limited song library in which some songs cannot be found on the application. For example, classic old songs or some unpopular songs most probably cannot be searched in the application, which is an extremely inconvenient situation for users. This will directly lead the users to switch to other music player applications.

iv. Do Not Have Download Option

Due to the rise of the use of smart phones, there are many applications comings out, MP3 music players are also one of them. Many people started to use this MP3 music player to listen to music which to chill their day. But there is a problem which users cannot download their favourite song directly from most of these MP3 music players. They needed to go to other sources to find the song then only import the song to those applications. It is a very tough and troublesome process before enjoying the music. So that, today's music player application should stay up with the times, allowing users the ease of directly downloading their favourite songs from the application itself rather than requiring them to obtain them in another means.

1.1.2 Motivation

The aim of the thesis is to propose an improved version of MP3 music player application. Although there are already many popular MP3 music player in the market but there are still problems contain in there. As mentioned above, common issue such as restricted only for listening to music, limited only on button control, and also song libraries are limited which will affect the user experience. The user uses the program to enjoy the music atmosphere and relax, but frequent problems will only make the user more irritable. Therefore, improving and adding functions to the MP3 music player application is important and should be considered.

1.2 Objectives

The goal of this thesis is to propose the creation of an android that:

- **Add-on Special Function**

  To enrich the music player application, this music player application will add some special functions such as trim music and equalizer control. Users can have different experiences on this special music player app and also interact more with this application, so they don't get bored and dull.

- **Add In Gesture Control**

  To make it easier for users to control the music player, gesture controls will be added to this MP3 application. It allows users to control the music player with gestures in inconvenient situations. Users can change or control songs by simply swiping somewhere on the screen without pressing the exact button.

- **Expand Song Collection Library**

  To ensure the all the music can be found in this application and also the music collection library is complete, we will always be updating the library. Also, we can expand the library size to store more genres of a song to fulfil all age groups of users.

- **Provide Download Function**

  The problem of not offering a download function will be solved by implementing a third-party MP3 music download website, since if we want to lawfully download music from the internet, it will require a lot of difficult processes and budget to acquire the approval of copyright, license, and so on. So, in this scenario, a third-party MP3 music download website will be employed to achieve the goal of “Settle All Things in Just One App”. After the download function is introduced, the user may pick the download icon, which will allow them to a third-party MP3 music download website where they can download their favourite music.

1.3 Project Scope and Direction

Develop a fully functional and free MP3 music player application whose basic functions are similar to other successful music player applications on the market. The project will use the useful functions evaluated in the literature to build an upgraded version of the program to provide users with a better user experience. After the project is completed, the MP3 music player application will be able to provide a luxurious user experience and an interesting application with various special features. This application expands and updates the song library by adding classic old songs and unpopular songs that are hardly found in modern music players. This project will also realize the freedom to download user's favourite songs. With gesture control, users can control the music flow easier and convenient compared to button pressing. Trimming and equalizer mode can increase the interesting index of this application to prevent users to drive away from the application and. Also, the equalizer can alter the music output based on the user need.

1.4 Contributions

The end result of this project is that all the basic functions in similar applications will be available in the MP3 music player application, allowing users to download their favourite songs without any restrictions. Additionally, the app supports gesture controls, allowing users to control the flow of songs simply by swiping somewhere on the screen, rather than focusing on pressing specific buttons. This will reduce accidents and increase convenience for users. Additionally, the app will collect and update a library of songs of any genre and any generation, allowing people of all ages to download their favorite genres of songs. Last but not least, the app will add trimming and equalizer features to enrich the entire app and give users a different experience of using MP3 apps.

1.5 Report Organization

The project report consists of 6 chapters. Chapter 1 discusses the background and motivation for making the application, the problem statement, and the goals based on the problem statement, recommended methods or research, and highlights of a successful application. Chapter 2 provides an overview of the three music player applications currently on the market, along with the comparison, advantages, and limitations of each. System design is covered in Chapter 3, which includes sitemaps, use-case diagrams, activity diagrams, and system wireframes, which are the proposed user interface for the application. Chapter 4 covers software design techniques, tools, requirements, system performance definitions, and timescales. Chapter 5 is about implementation and testing. Chapter 6 includes an overview, discussion, and conclusion of project achievements, challenges encountered during development, and future improvements.

Chapter 2

Literature Review

2.1 Previous Works on MP3 Music Player Applications

2.1.1 Spotify

Spotify is the world's largest music streaming service as of September 2021, with over 381 million monthly active users, including 172 million premium customers. Spotify provides digitally limited recorded music and podcasts from record labels and media firms, with over 70 million tracks available. (Spotify For The Record, 2021) Basic functions are free with advertisements and limited restrictions as a freemium service, but more capabilities, like as offline listening and ad-free listening, are accessible through premium subscriptions.

As shown in Figure 2.1.1.1, the freemium service of Spotify does not have the authority to listen to your favourite music. For example, if a user searches for “mama’s boy” by LANY in Spotify, it will return a song list that includes the song that the user searched for. Users can only select the option 'Shuffle Play', so it's all about luck to listen to the specific song that the user wants.

Figure 2.1.1.1 User cannot directly listen to selected song

Furthermore, with the freemium of Spotify, advertisements will be played in the middle of the music that the user is listening to, which has a negative impact on the user experience. In other words, it has ruined the most basic purpose of a music player application, which is to allow the user to listen to their song to the finish without interruption, which is a very poor experience. Spotify is available on a variety of devices, including computers, phones, tablets, speakers, televisions, and automobiles, and you can effortlessly switch between them using Spotify Connect. Spotify exclusively provides music and podcasts via their applications. Because of their license, we are unable to export our material outside of the app. People don't like to pay for anything by nature, it's best if it's free. In any case, these are the reasons why consumers are willing to spend money on Spotify's premium price.

Spotify's UI design is modern and clean. Each of the functions can be seen easily on the main page and make it easy to operate. Users may search for music by artist, album, or genre, and they can also build, update, and share playlists. (Figure 2.1.1.2) Moreover, the transition animation in Spotify improves the beauty of the entire program, and the operation of the whole application is also very smooth which cannot compare with other competitors. Spotify's UI design and stability are benefit those other developers may learn from, resulting in a highly positive user experience.

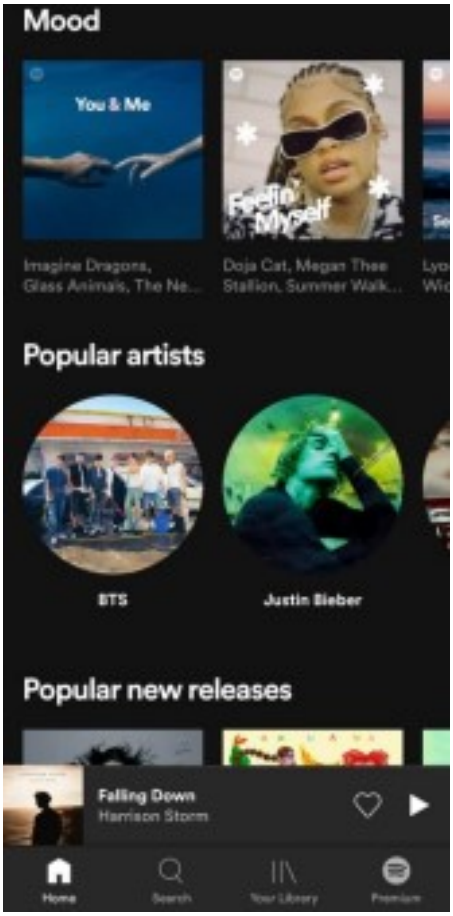
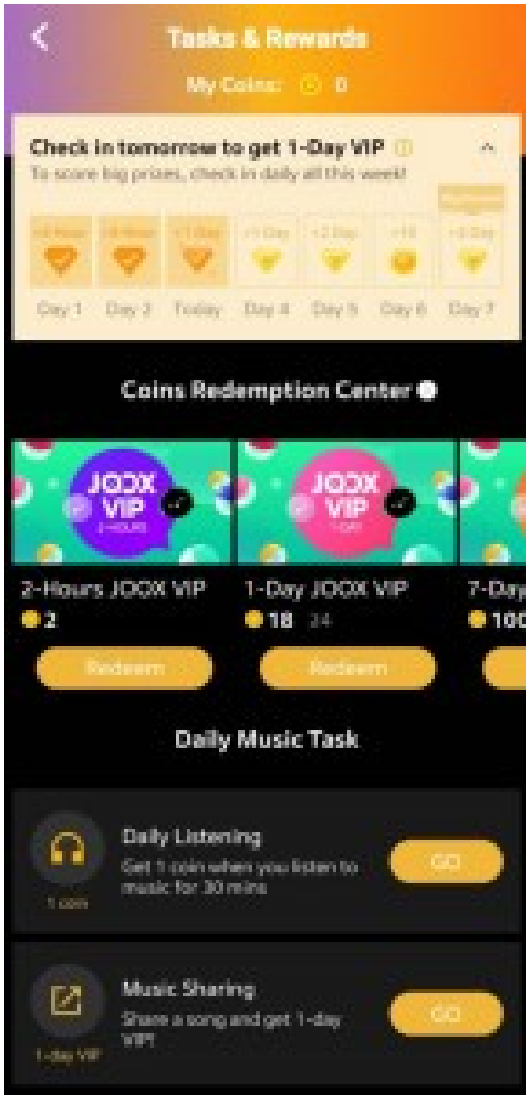


Figure 2.1.1.2 Spotify Homepage UI

However, owing to Spotify's aim of encouraging users to acquire the premium version, Spotify will receive a penalty point, because the most crucial feature of the music player application is uninterrupted music listening from beginning to end. Spotify may really add advertisements at the conclusion of music or before the commencement of music to make customers feel more at ease.

2.1.2 JOOX

Tencent's JOOX is a music streaming service that was introduced in January 2015. JOOX is the most popular music streaming app in Asian regions including Hong Kong, Macau, Indonesia, Malaysia, Myanmar, Thailand, and South Africa. (Wikipedia, 2021) JOOX app is now available for download for iOS and Android devices on App Store, Google Play and Huawei App Gallery, respectively. The service went online on June 24, and the app has had over 100,000 downloads since then. (Edwin Yapp, 2015) JOOX is a freemium service, with the majority of its songs available for free, while some are exclusively available to premium users, who may obtain them through paid memberships or by completing certain activities such as unlimited skips, ad-free, offline listening, quality streaming and play on demand. JOOX VIP is the only one who can use Auto-Download. Open the appropriate playlist and check the box next to "Auto-Download." When you use Wi-Fi, JOOX will automatically download songs from your playlist. Without JOOX VIP, it can’t perform offline listening which is data consumption. JOOX can be upgraded to the premium version without paying. Users only need to sign in every day to enjoy VIP benefits for a limited time. Another way to get JOOX VIP is that users can share the song to WeChat Moments or Facebook to get it as a reward. This is beneficial for users with lower living costs, such as students. JOOX offers various music qualities to meet the needs of the users. (Figure 2.1.2.1 & Figure 2.1.2.2)



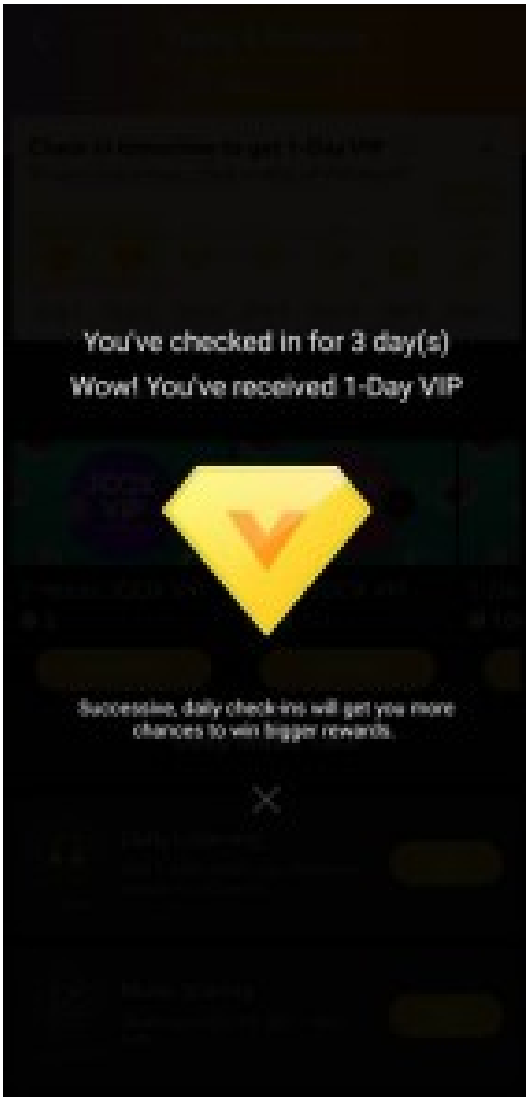


Figure 2.1.2.1 Check-in daily/share music **Figure 2.1.2.2 Get VIP reward**

Furthermore, JOOX provides normal music quality at 64Kbps, which is poor and produces loud sound, but it saves a lot of data, especially for customers in rural regions. For VIP members, the portal also provides medium music quality at 128Kbps and 320Kbps. Users may select the music quality that best suits their needs. The music discovery option helps users to quickly and easily locate their favourite tunes. It becomes another important aspect to consider while searching for the best music streaming experience. JOOX also has amazing music discovery skills. Radio, Featured Artists, New Releases, and Editor's Picks are some of the music recommendations available. (Figure 2.1.2.3)

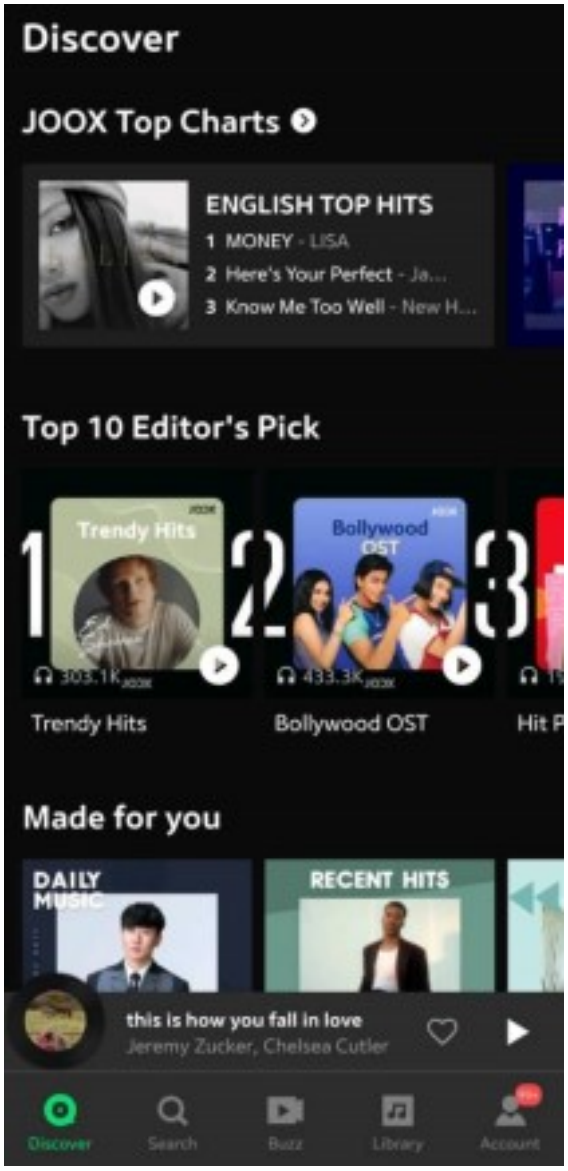


Figure 2.1.2.3 Music recommendations

However, the music collection library of JOOX is slightly smaller, some of the songs that cannot be found in JOOX. As a good music player application, it must have a large base of music collection library which contain as many songs as it can to meet all the user's needs. In addition, JOOX is not available worldwide, it is not available in some country's Google Play Store or App Store. To users in those countries who wanted to use JOOX, they need to use VPN to access it which is a very complex procedure to use it. Like other applications, JOOX also has ads disturbance problems.

2.1.3 YouTube Music

YouTube Music is a music streaming service created by YouTube, a Google company. YouTube Music is a famous music video streaming software that has grown in popularity in recent years. The application had almost 70 million official tracks. The most significant feature of YouTube Music is its huge music collection, which includes not only formally published songs but also remixes and covers. Users not only can listen to the songs of famous artist but also can enjoy the works by internet singers. It differs from other typical music player applications in that it does not permit playing songs from internal storage. Instead, it exclusively plays music straight from its database. (Google Play Store, 2022)

YouTube Music's homepage (Figure 2.1.3.1) displayed a large number of recommendations in many genres of music styles for users to pick from. It also researched the user's favourite music preferences and then generated several sorts of playlists that the user would enjoy, therefore resolving the problem of a user who has difficulty making a decision. In addition, the user's most often listened to music will be gathered as a playlist, reducing the time the user spends searching for their favourite music and saving the process of manually adding their music to a playlist. Because YouTube Music obtains information directly from the internet, it does not suffer from the absence of music data such as picture and artist that traditional music applications suffer from; instead, every aspect of music, including MV and lyrics, can be accessed in the programme.

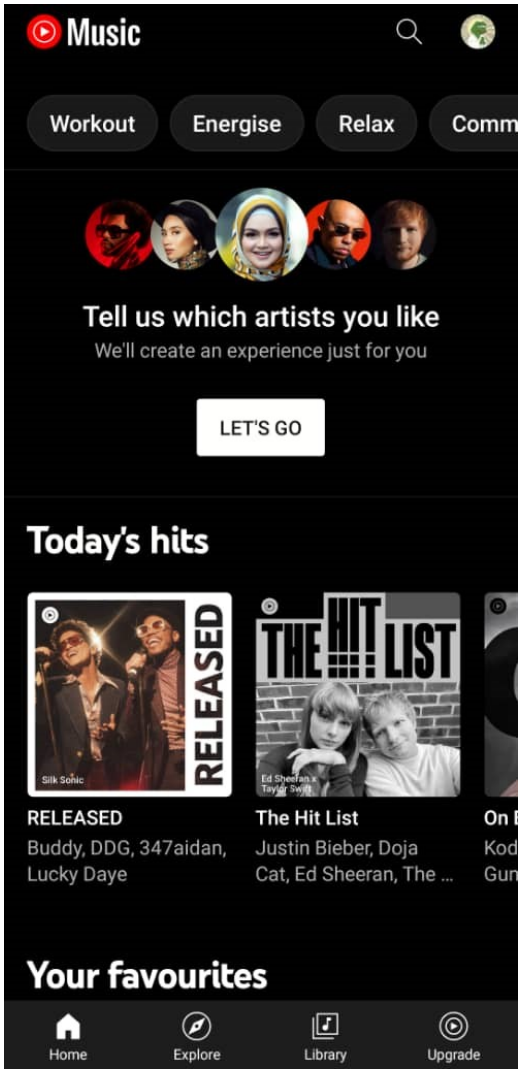


Figure 2.1.3.1 Homepage

Furthermore, on the Explore page (Figure 2.1.3.2), the music was divided into albums based on moods and genres. There was a New Releases area for users to find new music in the business; it was a useful tool for those who enjoy novelty. Under addition, in the chart function (Figure 2.1.3.3), YouTube Song featured the top music, artist, and album during the previous week or month, making it easier for users to find popular music and artists currently. Furthermore, the entire UI design of YouTube Music was biased towards black color, and the font size utilised was just appropriate for a comfortable visual experience. The transition animations for each portion were incredibly smooth to use, which is an additional advantage over other music applications on the market. A search option is also given, allowing users to search for music all around the globe using only their fingertips on the Internet.

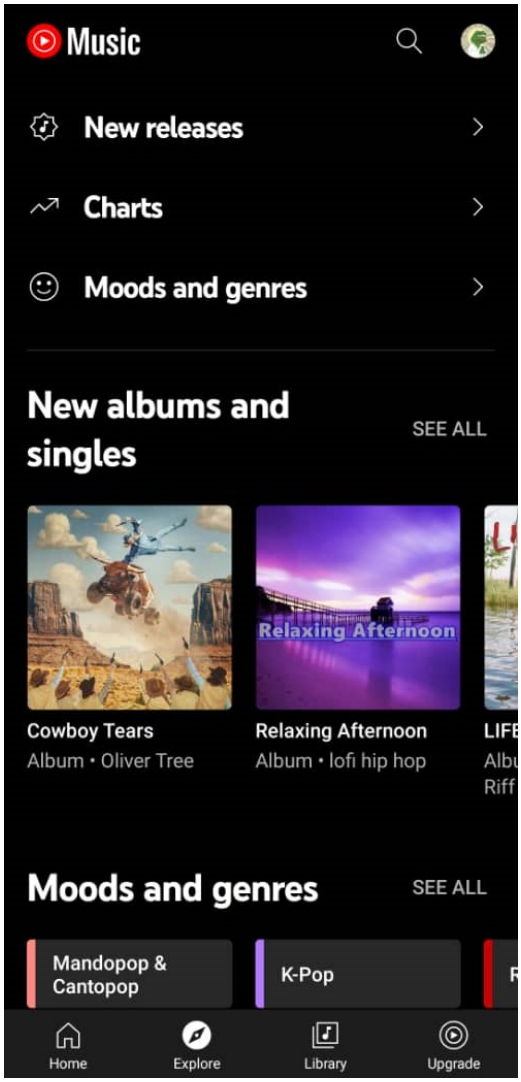


Figure 2.1.3.2 Explorer page

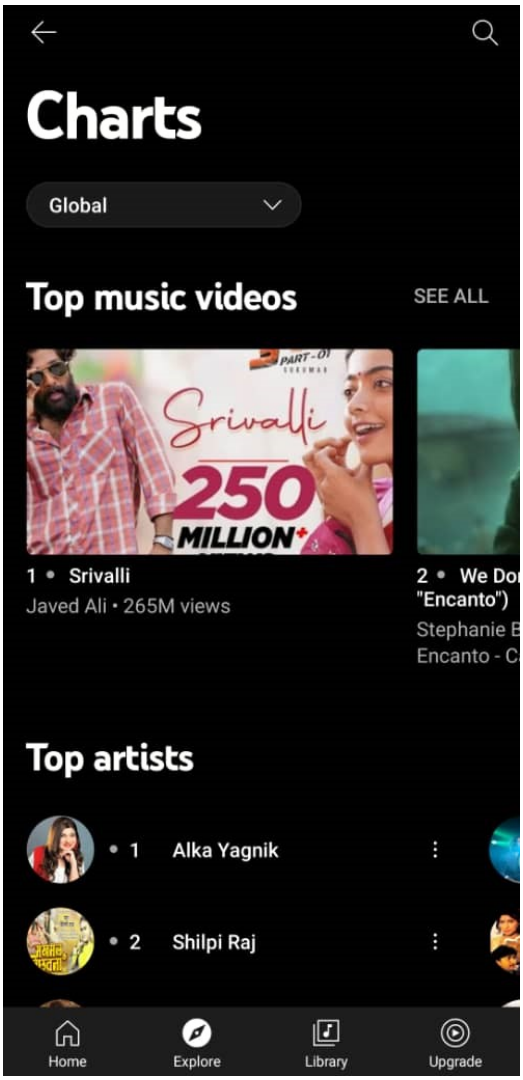


Figure 2.1.3.3 Chart

Furthermore, in the playback interface (Figure 2.1.3.4), YouTube Music did present the song video together with the music, which will add some new criteria into standard music player application, but it may eat up more user's mobile data in order to play. However, YouTube Music provided another alternative for users to listen to music without seeing a music video, but it required them to upgrade to the premium version. In addition, the interface had a clean, basic, and black theme UI that provided users a premium sense. It is worth noting that YouTube Music does offer lyrics for users to peruse, which is a really useful function (Figure 2.1.3.5). Aside from the lyrics option, there is “Up Next” which allows the user to preview the next song that will be played, and “Related” which will propose similar types of music based on the music the user was listening to.

Figure 2.1.3.4 Playback Interface

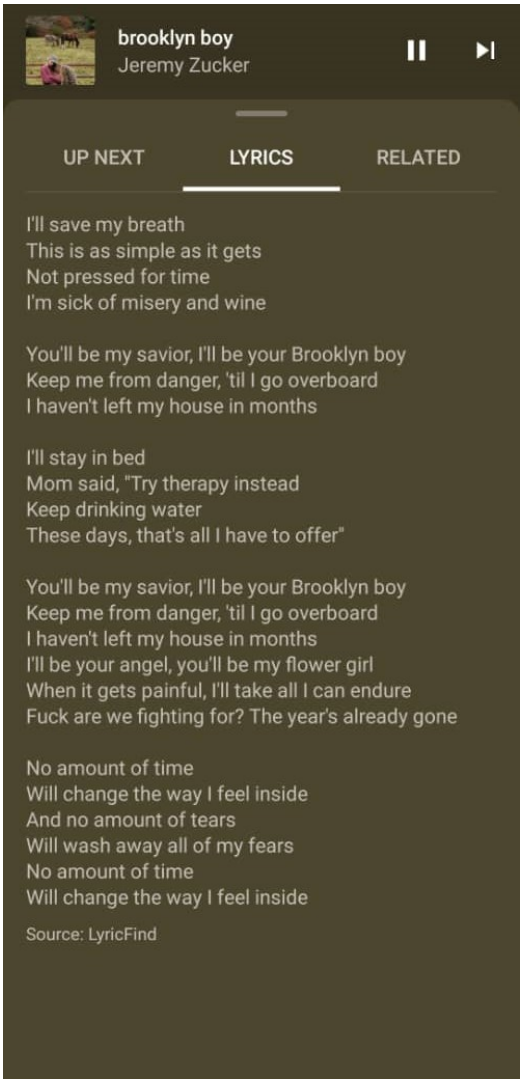


Figure 2.1.3.5 Lyrics

Besides, YouTube Music did include a feature called 'History' enabling users to evaluate what music they had recently listened to with highly detailed information such as song name, artist, and time. (Figure 2.1.3.6) Aside from that, users may subscribe to their favourite artists, and after doing so, they will receive notifications anytime the artist they subscribe to releases new music, which is similar to YouTube. (Figure 2.1.3.7) Surprisingly, not only official artists, but also amateur artists and even unknown musicians, are permitted to subscribe as long as the user likes it. The search tool in YouTube Music not only searches for new music but also works in any portion of the playlist, such as the history, top chart, and other sorts of playlists, making it a strong search engine.

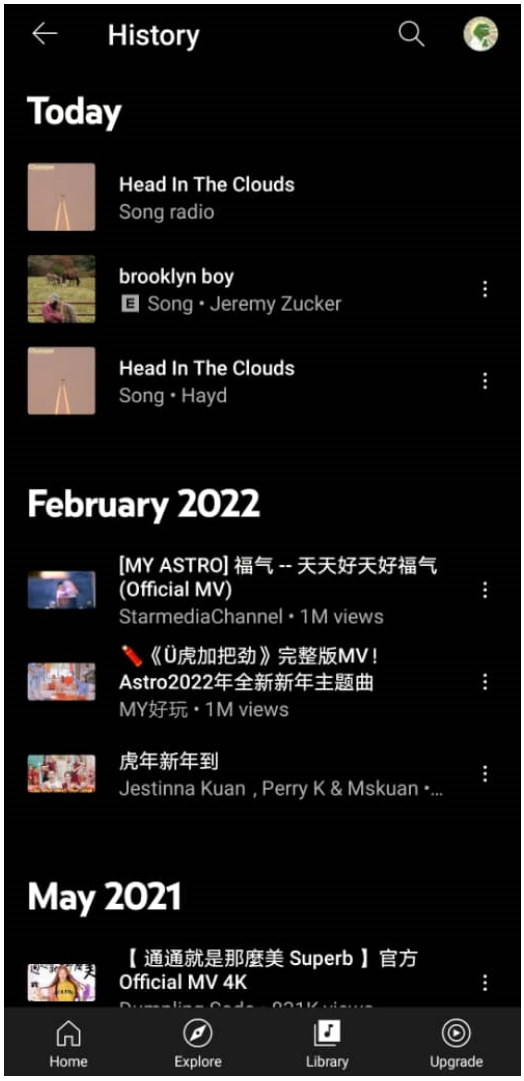


Figure 2.1.3.6 History page

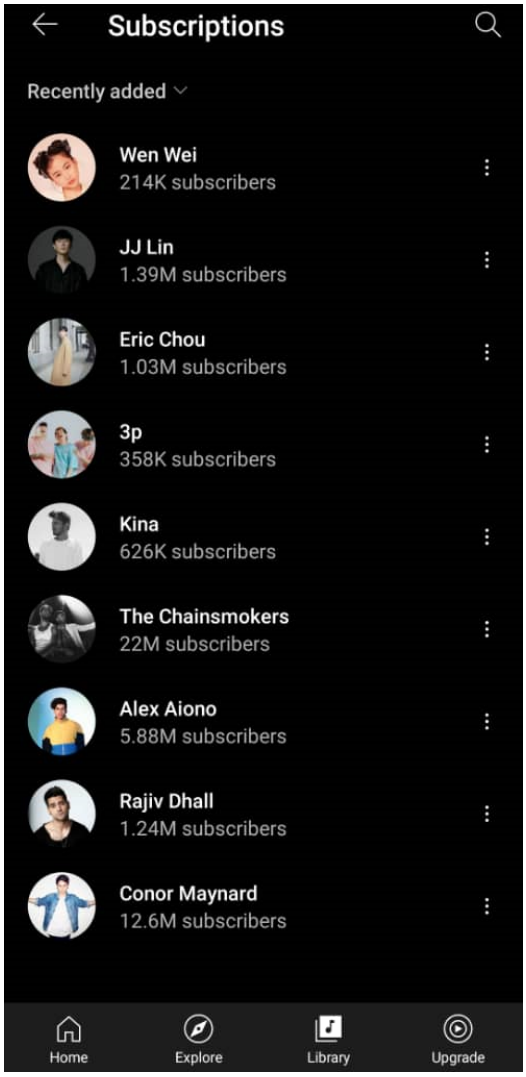


Figure 2.1.3.7 Subscription page

However, some of the features on YouTube Music are only available if the user subscribed the premium version. (Figure 2.1.3.8) For example, in order to enjoy ad-free music, the user must subscribe to the premium version; otherwise, the user would be forced to listen to advertisements in the midst of the song. Furthermore, the premium version will allow users to listen to music in the background, which means they can continue playing their music outside of the application or even on the lock screen, whereas the standard version requires users to stay inside the application in order to continue listening to music. Aside from that, users may download their favourite songs in the premium version. In any case, users may also choose a family or student plan to enjoy a lower membership charge and a 7-day trial period.

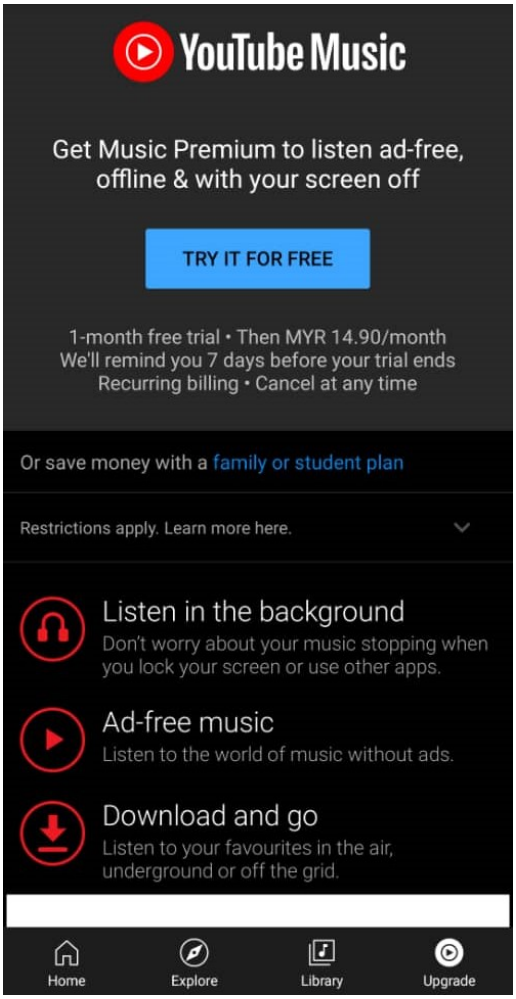


Figure 2.1.3.8 YouTube Music Premium

To summarise, YouTube Music is a very user-friendly music video streaming programme with a plethora of useful functions. Even without the premium version, the functions and capabilities supplied by YouTube Music are more than adequate for regular usage. Unfortunately, customers who do not choose to pay for the premium edition will have to put up with the intrusive adverts that display throughout the song's playback, preventing them from completely enjoying the music.

On the plus side, YouTube Music has a very effective suggestion system that allows users to continue exploring their favourite kind of music. Furthermore, the lyrics feature increased user involvement with the programme, allowing users to enjoy their music while also understanding the meaning of the song by seeing a preview of the lyrics at the same time. Others classic music programmes should take note of the seamless transition animation and pleasant dark tone of the UI design. Overall, YouTube Music's UI is well-designed, functional, and provides a fantastic user experience.

2.2 Critical Remark of Previous Works

2.2.1 Strength and Weakness of Previous Works

Application Name	Strength	Weakness
JOOX	• Able to select song quality • Provide excellent song finding capabilities • Share or check-in to get free VIP	• Need JOOX VIP to perform certain function • Music collection library are small • Country restriction
Spotify	• UI modern and clean • Provide excellent song finding capabilities • Stable app	• Must subscribe premium version to perform certain function • Does not have the authority to listen to your favourite music
YouTube Music	• Nice UI • Lyrics Shown • Great Music Recommendation Features	• Must subscribe premium version to access certain features

Table 2.2.1.1 Strength and Weakness of Reviewed Application

2.2.2 Application Comparison

Criteria	Spotify	JOOX	YouTube Music	Proposed Application
User Interface	Good	Average	Great	Good
Download Music	✓	✓	×	✓
Sleep Timer	×	✓	×	✓
Notification	✓	✓	✓	✓
Background Play	✓	✓	×	✓
Gesture Control	×	×	×	✓
Equalizer	×	×	×	✓
Trimming	×	×	×	✓

✓: Yes ×: No

Table 2.2.2.1 Comparison of Reviewed and Proposed Applications

Chapter 3

System Design

3.1 Site Map

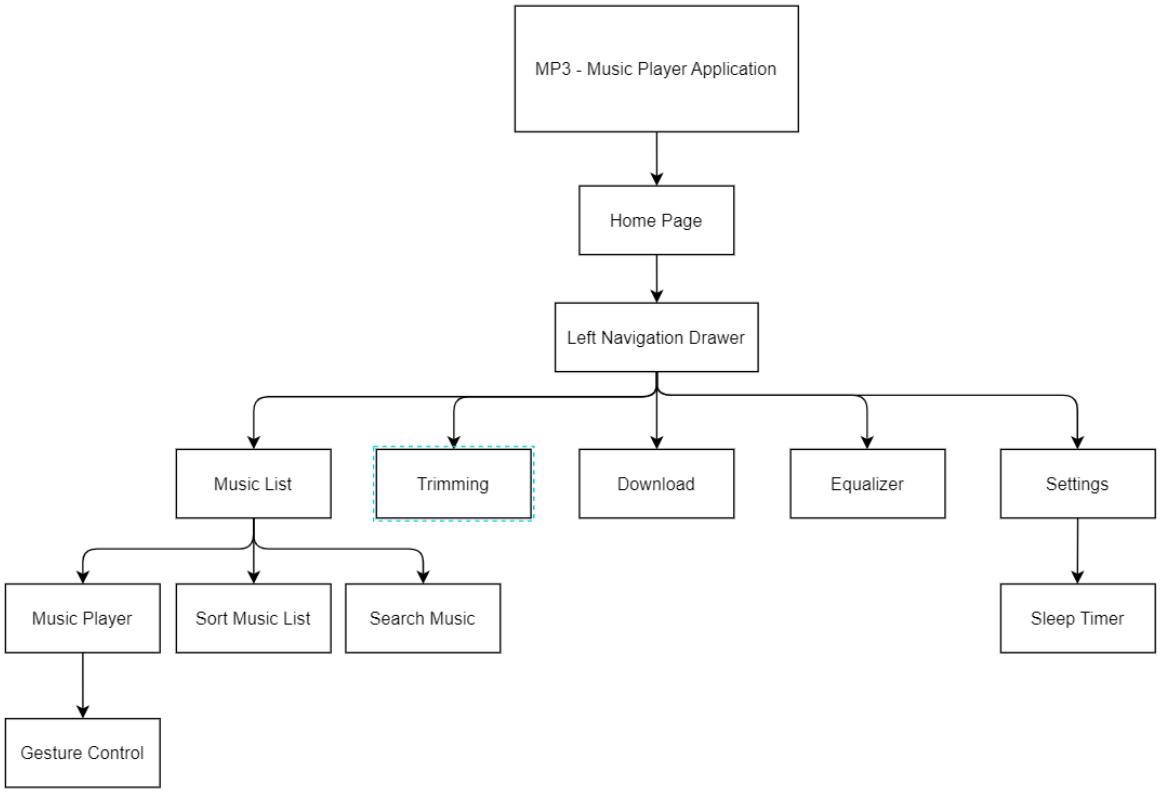


Figure 3.1.1 Site Map Diagram

A visual sitemap gives us an at-a-glance look at the layout of your application, including the hierarchy of elements and the relationship between content and pages. The site map above (Figure 3-1-1) show that this application consists of the left navigation drawer which contain “Music List”, “Trimming”, “Download” and “Settings”. It allows user to choose a feature and will execute each functions relatively.

As shown in the sitemap, the first element "Music List" also serves as the home page of the app. From this page, the user can perform search, sort or select a song from the displayed music list. When user clicked on a song, it will direct them to the "Music Player" interface. On the music player activity page, the user can control all the control buttons such as

play/pause, skip/previous and others. Also, by using button to control, users are also allowed to control by using gesture.

Moreover, under the “Trimming” feature, the user can trim the selected song and save it inside the phone. Also, users can download the songs they wanted in the “Download” page.

On the equalizer page, the user can choose the output of the music from a given list or adjust it according to their own preferences.

There is a function under "Settings" namely "Sleep Timer". Users can set the sleep time by selecting the “Sleep Timer” option. After the set time is reached, the song will stop playing and the application will terminate.

3.2 Use Case Diagram

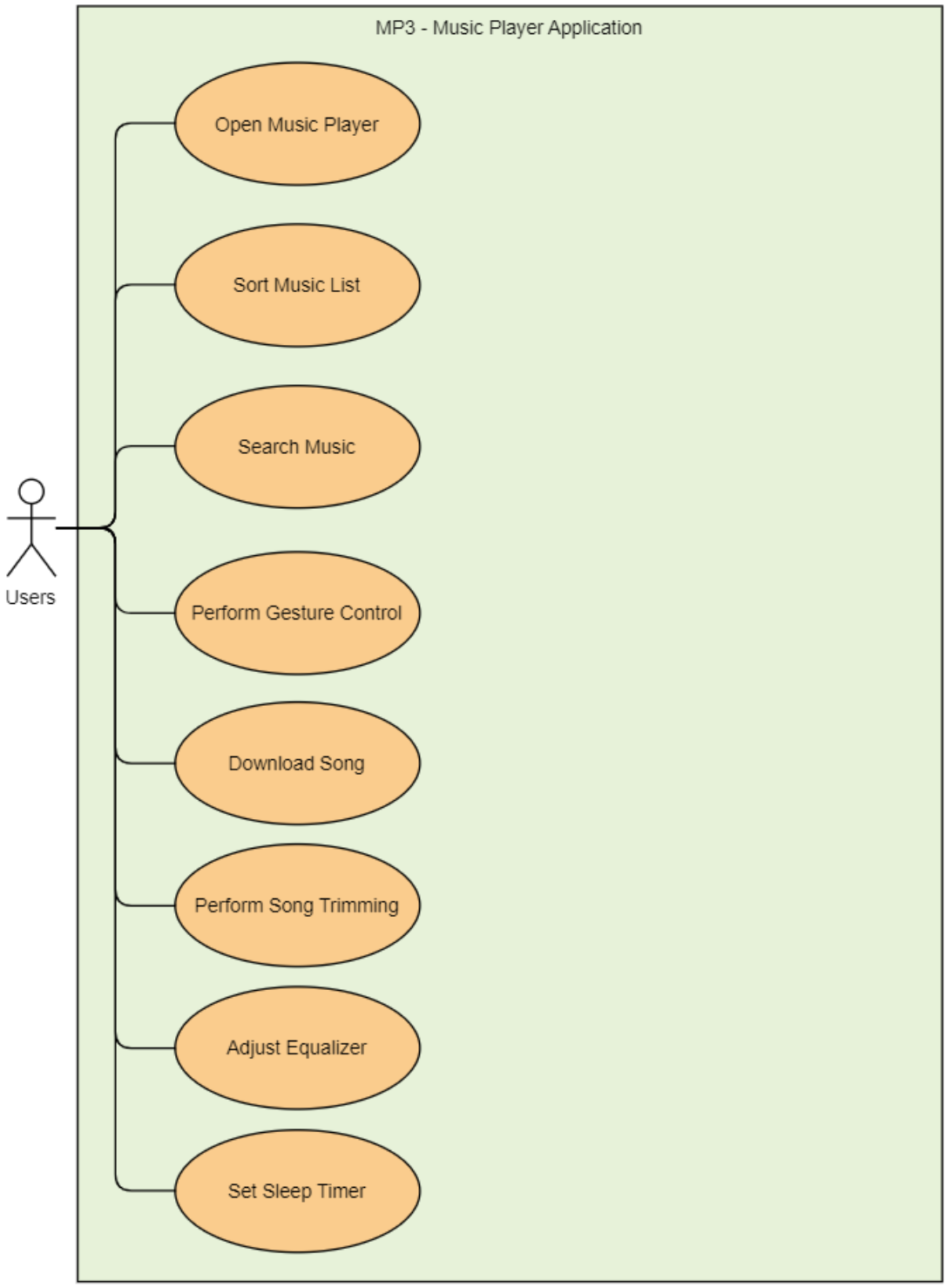


Figure 3.2.1 Use case diagram of MP3 - Music Player Application

Use Case Name: Open Music Player	ID: UC 001
Actor: User	
Description: User able to listen to music which they have downloaded	
Trigger: User select a song to play	
Precondition: There is a playlist exist	
Normal Flow of Events: 1. Users select a song to play	
Alternate / Expectational Flows: 1a. The application will display "No Song Exist" if there is no song exist in the playlist	

Table 3.2.1.1 Use Case Description of Open Music Player

Use Case Name: Sort Music List	ID: UC 002
Actor: User	
Description: User able to sort the list displayed in music list	
Trigger: User click the dot menu	
Precondition: There are more than 1 music exist in the list	
Normal Flow of Events: 1. Users click the dot menu on the top right corner of the screen 2. Users select sort by what method	
Alternate / Expectational Flows: 2a. When users select to sort by name, the music list will present in alphabetically 2b. When users select to sort by date, the music list will present in earliest download date at the top 2c. When users select to sort by size, the music list will present in smallest file size first.	

Table 3.2.1.2 Use Case Description of Sort Music List

Use Case Name: Search Music	ID: UC 003
Actor: User	
Description: User able to search the artist's name or title of the song in the search bar	
Trigger: User click the magnifier icon	
Precondition: There is 1 music exist in the list	
Normal Flow of Events: 1. Users click the magnifier icon that beside the dot icon 2. Users type the artist's name or title of the song in the search bar	
Alternate / Expectational Flows: -	

Table 3.2.1.3 Use Case Description of Search Music

Use Case Name: Perform Gesture Control	ID: UC 004
Actor: User	
Description: The user able to perform certain actions in Music Player page by using gestures	
Trigger: When the user performs the gesture pattern on the screen	
Precondition: The user should be in the Music Player page	
Normal Flow of Events: 1. Users select a song 2. Users direct to Music Player page 3. Users perform the gesture action on the screen	
Alternate / Expectational Flows: 3a. When the user swipe to left on the screen, it will switch to the next song 3b. When the user swipe to right on the screen, it will switch to the previous song 3c. When the user double tap on the screen, it will pause/play the song	

Table 3.2.1.4 Use Case Description of Perform Gesture Control

Use Case Name: Download Song	ID: UC 005
Actor: User	
Description: Allow user to download music from third-party site	
Trigger: User search for a music and press “Download” button	
Precondition: User’s device is connected to mobile data or Wi-Fi	
Normal Flow of Events: 1. Users go to left navigation drawer 2. Users click “Download” 3. User accesses to third-party music download site 4. Users search the artist or song by its name 5. Users select the song they wanted 6. Users download the song they found	
Alternate / Expectational Flows: 2a. The application will display “No Internet Connection” when user’s device is not connected to mobile data or Wi-Fi	

Table 3.2.1.5 Use Case Description of Download Song

Use Case Name: Perform Song Trimming	ID: UC 006
Actor: User	
Description: Allow user to trim the song	
Trigger: User select “Song Trimming” in left navigation drawer	
Precondition: MP3 format audio files must be available on the user's device	
Normal Flow of Events: 1. Users go to left navigation drawer 2. Users click “Song Trimming” 3. Users click “Add Audio” 4. Users select a song that wanted to trim 5. Set the start and end point 6. Users click “Cut Audio” 7. Users enter the filename that wanted to save 8. Save the edited song into user’s device	
Alternate / Expectational Flows: 7a. Users click “Cancel”, the operation will terminate	

Table 3.2.1.6 Use Case Description of Perform Song Trimming

Use Case Name: Adjust Equalizer	ID: UC 007
Actor: User	
Description: Allow user to change the output of the songs	
Trigger: Select “Equalizer” in left navigation drawer	
Precondition: -	
Normal Flow of Events: 1. Users press the left navigation drawer 2. Select “Equalizer” 3. Change the output of the songs	
Alternate / Expectational Flows: 3a. Users choose the drop-down menu to choose the pre-setting 3b. Users self-adjustment	

Table 3.2.1.7 Use Case Description of Adjust Equalizer

Use Case Name: Set Sleep Timer	ID: UC 008
Actor: User	
Description: Allow user to set Sleep Timer which will automatically stop the app when reached the set time	
Trigger: User select the “Sleep Time” option	
Precondition: -	
Normal Flow of Events: 1. Users go to left navigation drawer 2. Users select “Sleep Time” 3. Users slide the number picker to set the hour and minute 4. Users click “Start” button to start the countdown	
Alternate / Expectational Flows: 4a. Users click “Cancel” button to cancel the countdown	

Table 3.2.1.8 Use Case Description of Set Sleep Timer

3.3 Activity Diagram

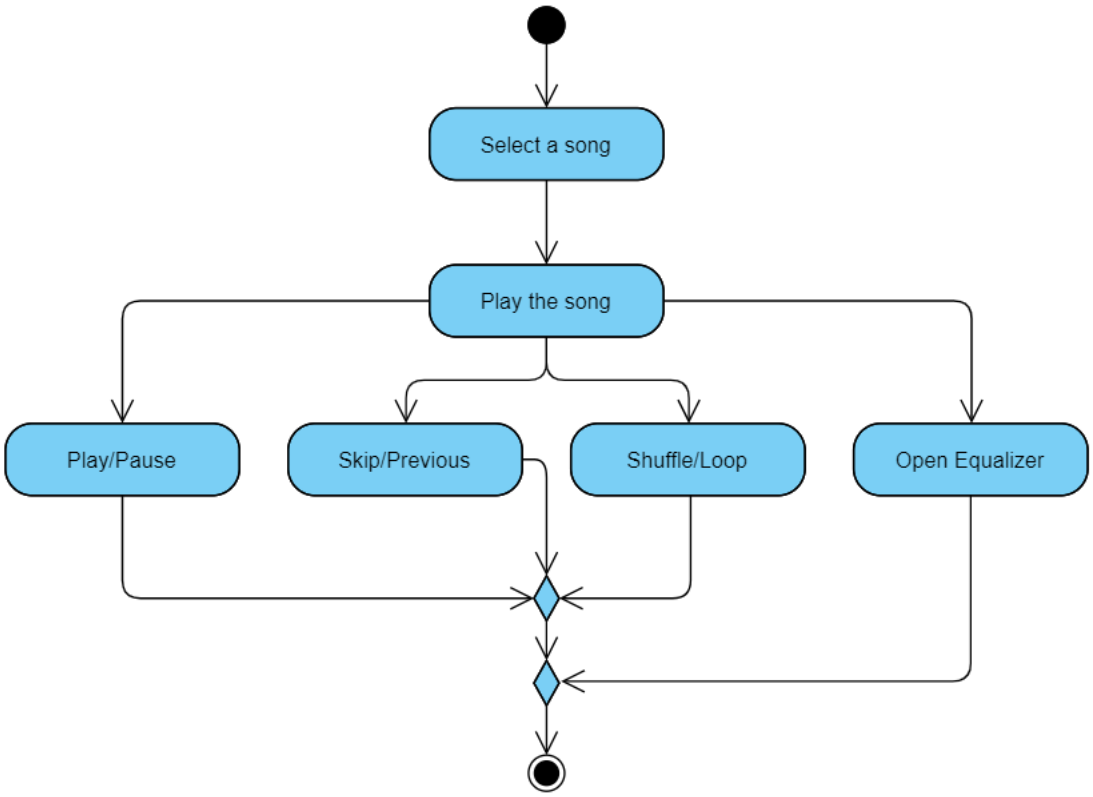


Figure 3.3.1.1 Activity Diagram for Open Music Player

First, when user launch the application, it will direct them to the main page which is the "Music List" page. Then, there will be a list of songs that read from the local phone storage and users can select a song to play. After the users select a song to play, the application will lead the user to music player interfaces which allowed user to control the music. Users can perform play or pause the song, skip to the next song or go back to previous song and loop a single song or random playing without following the sequences. In this page, users also can play with the equalizer to tune the music to different outputs.

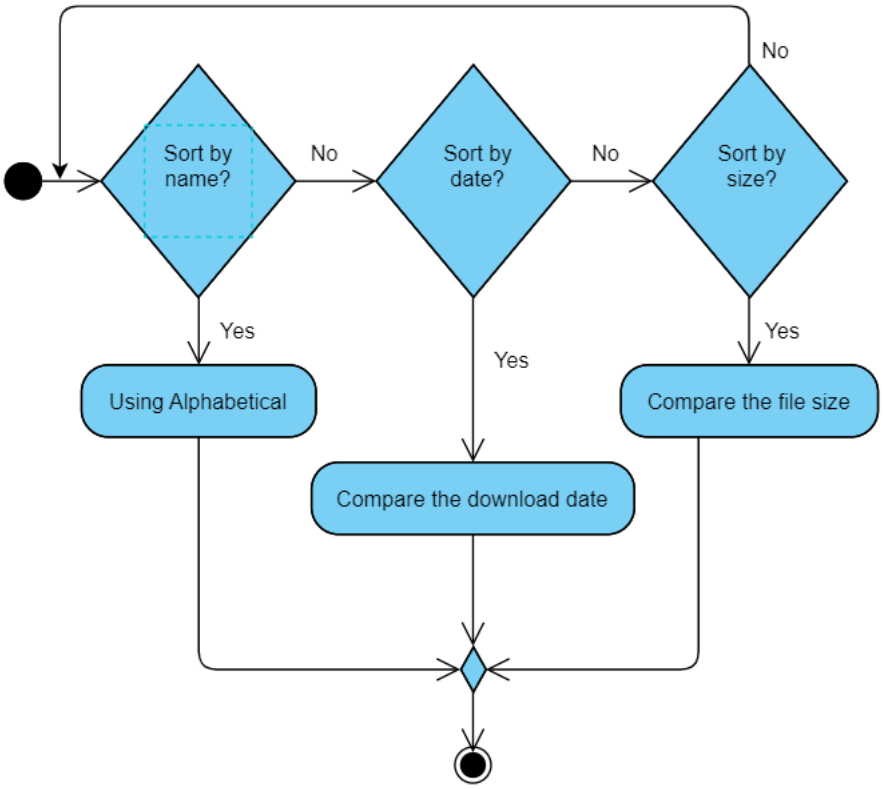


Figure 3.3.1.2 Activity Diagram for Sort Music List

On the home page, there is a sort menu in the upper right corner in the form of a dot menu. A drop-down menu with three alternatives will show when the user clicks on it, giving them a choice. Users have the option of sorting by size (smallest file size first), date (earliest download date at the top), or name (alphabetically).

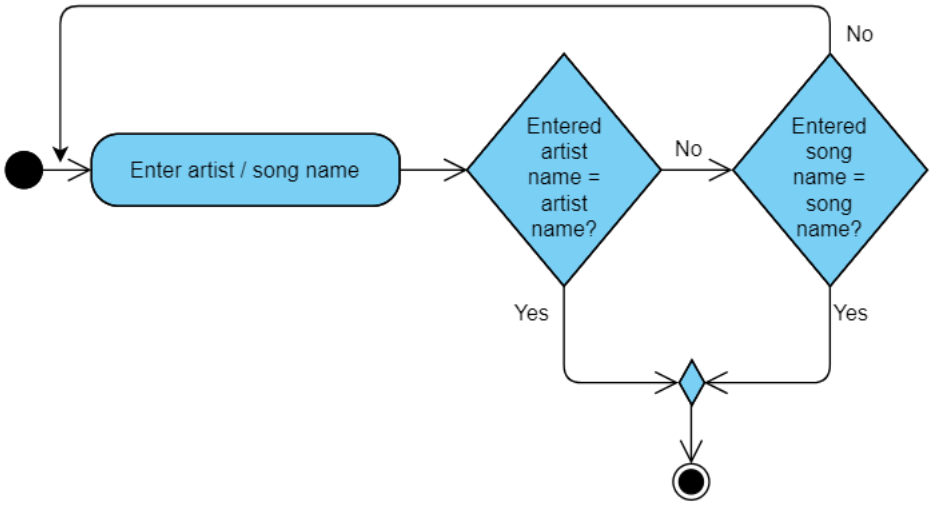


Figure 3.3.1.3 Activity Diagram for Search Music

User can type the song name or artist name that they wanted to find in the bar. Then, user can click to the music and start playing.

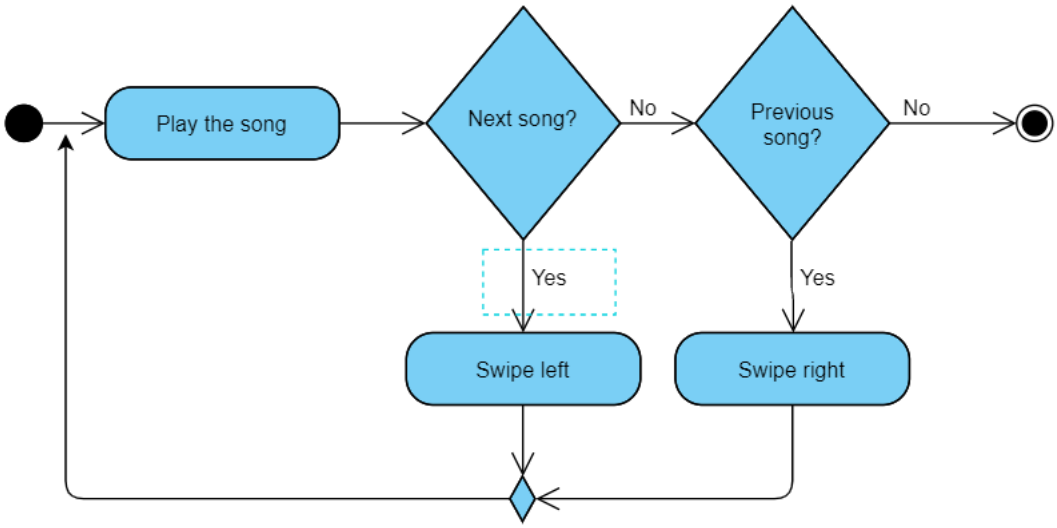


Figure 3.3.1.4 Activity Diagram for Perform Gesture Control

To perform gesture control, users need to select a song from the song list and the system will then direct the user to the music player page. In this page, users can perform gesture control such as swipe left on the screen to skip to next song or swipe to right to go back to previous song. Additionally, users can double-tap the screen to play and pause the music.

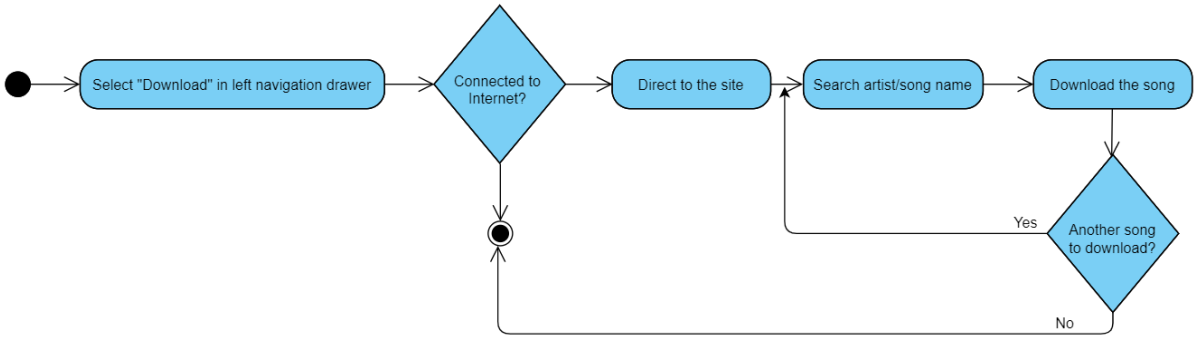


Figure 3.3.1.5 Activity Diagram for Download Song

To download songs, users need to browse the navigation drawer and select the “Download”. Users must be connected to the Internet first to access the site. Later, when users reach the site, they are allowed to search for artist or song name at the search bar. Then, they choose the desired song from the long list and download it.

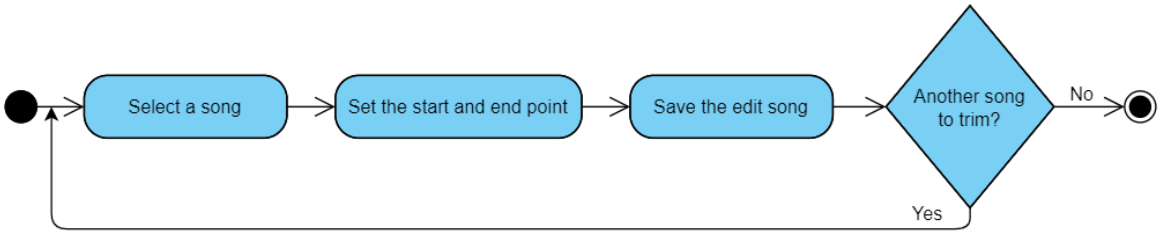


Figure 3.3.1.6 Activity Diagram for Song Trimming

To perform song trimming, users need to select a song to trim. After that, the user can drag the displayed progress bar to specify the start and end points. After the song is trimmed, a dialog box will pop up asking the user whether to save the trimmed song.

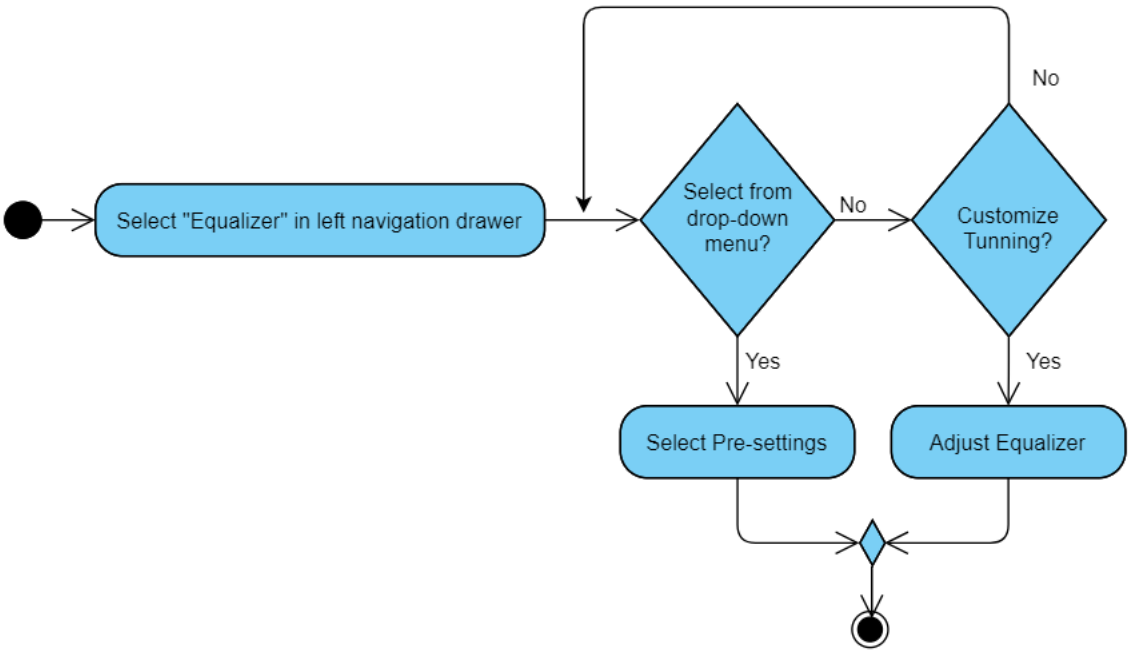


Figure 3.3.1.7 Activity Diagram for Adjust Equalizer

To perform different output of the song, user need to browse to navigation drawer and select “Equalizer”. Inside the page user can either select the pre-setting from the drop-down menu or they can tune it via adjusting the bars and spinner.

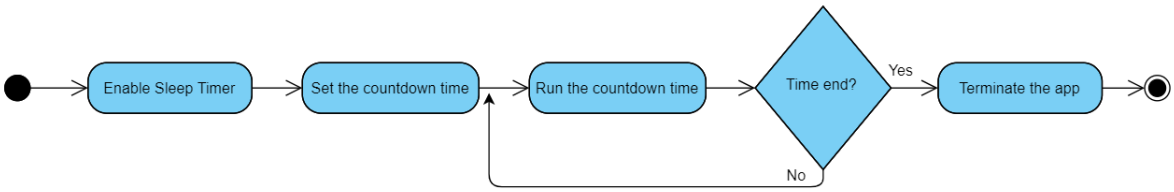


Figure 3.3.1.8 Activity Diagram for Set Sleep Timer

To set the sleep timer, user need to enable the sleep timer by set the countdown time then the timer will start to countdown. After when to time is over, the application will terminate automatically.

3.4 System Wireframe

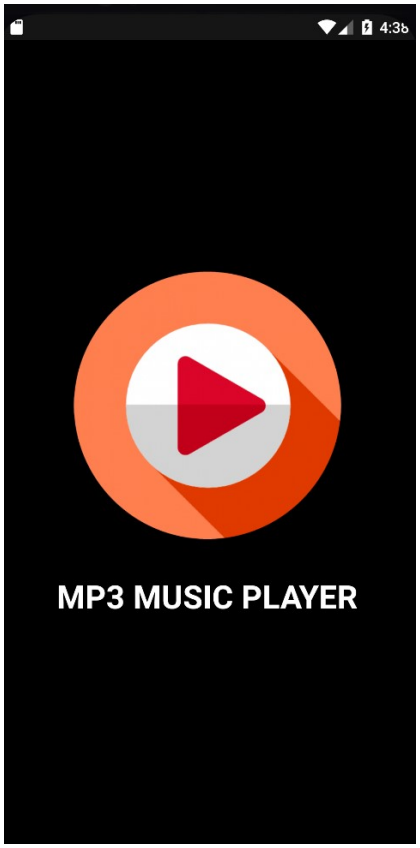


Figure 3.4.1 Splash Screen

When users launch the application, they first see a splash screen titled "MP3 Music Player" and a "Play" icon (Figure 3.4.1). The splash screen will appear for a few seconds and then become the home page of the application.

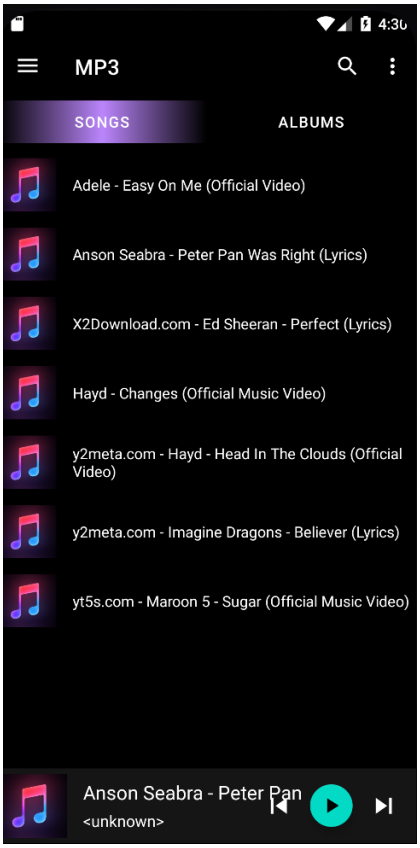


Figure 3.4.2 Home Page & Songs Fragment

After the splash screen, the user will go directly to the main page or home page. As shown in the figure above (Figure 3.4.2), there are 2 fragments, one is a song fragment and the other is an album fragment. By default, it will be in the song fragment. In addition, a list of songs read from the user's phone storage is listed. So, when the user launches the app for the first time, the system will get the user permission to read the storage.

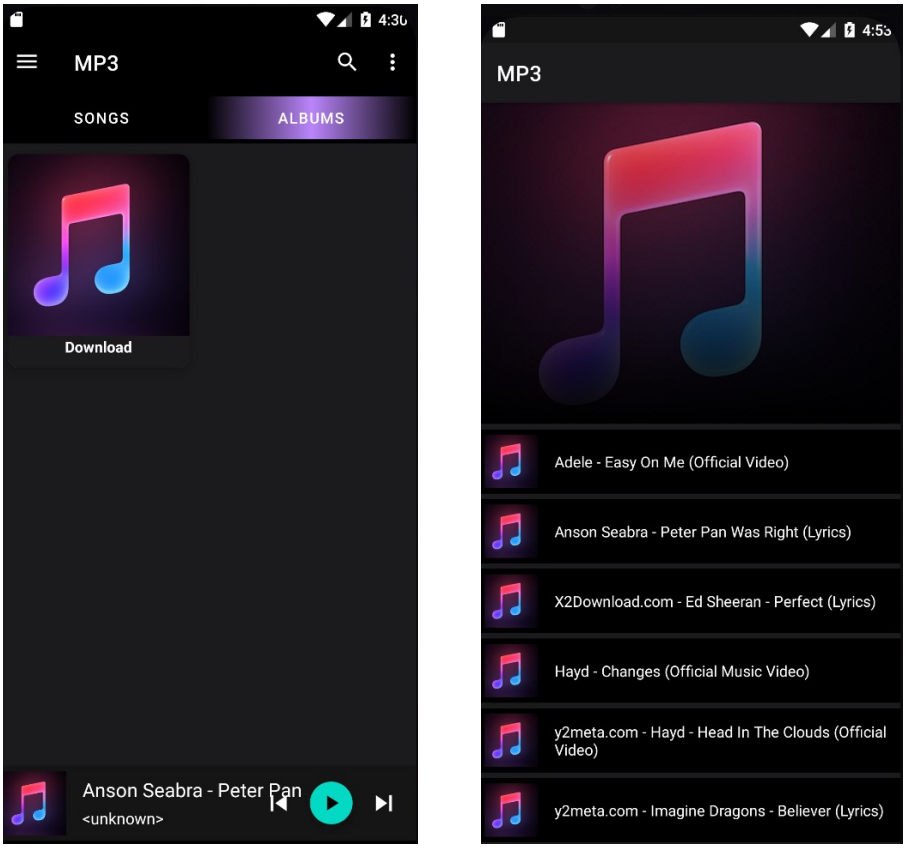


Figure 3.4.3 Albums Fragment

When user select the song fragment, the song will be categorized into an album. As shown in above (Figure 3.4.3), there is a “Download” album. This is because all the songs in this project are download from the same source, so they would be categorized into the same album which is “Download” album. When user selected the album, there will be a list of songs that belongs to that album. User can select a song from the list and play.

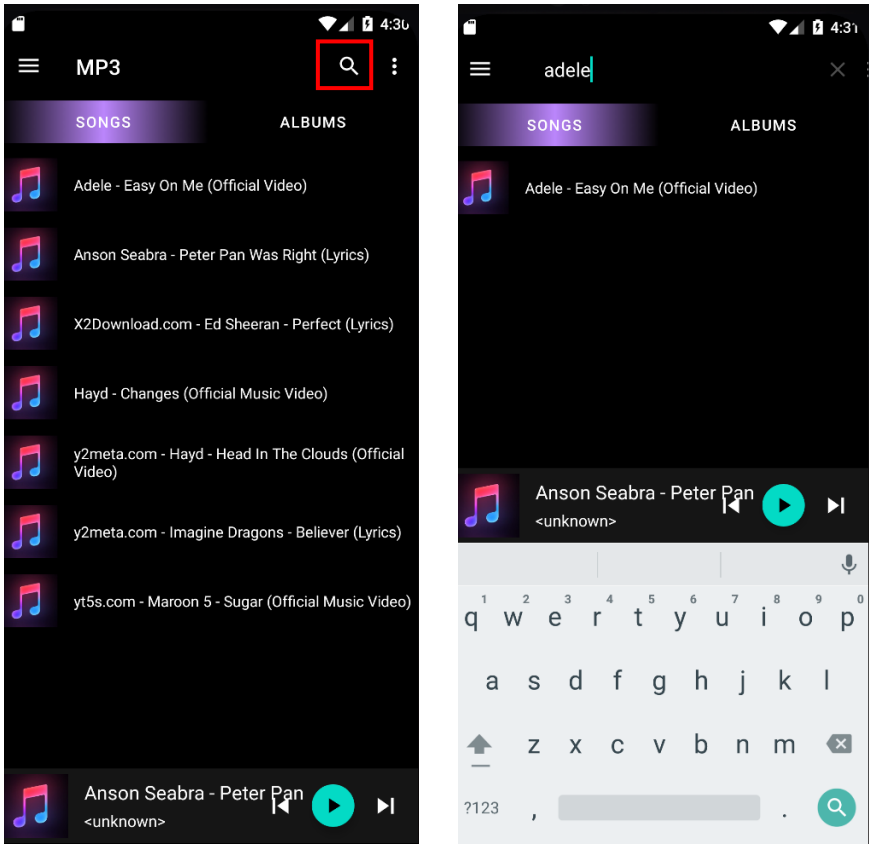


Figure 3.4.4 Search Bar

When user click the magnifier in the home page, they can type the song name or artist name that they wanted to find in the bar. As shown in figure 3.4.4 when user type “Adele” all the songs that belongs to Adele will listed out. Then, user can click to the music and start playing.

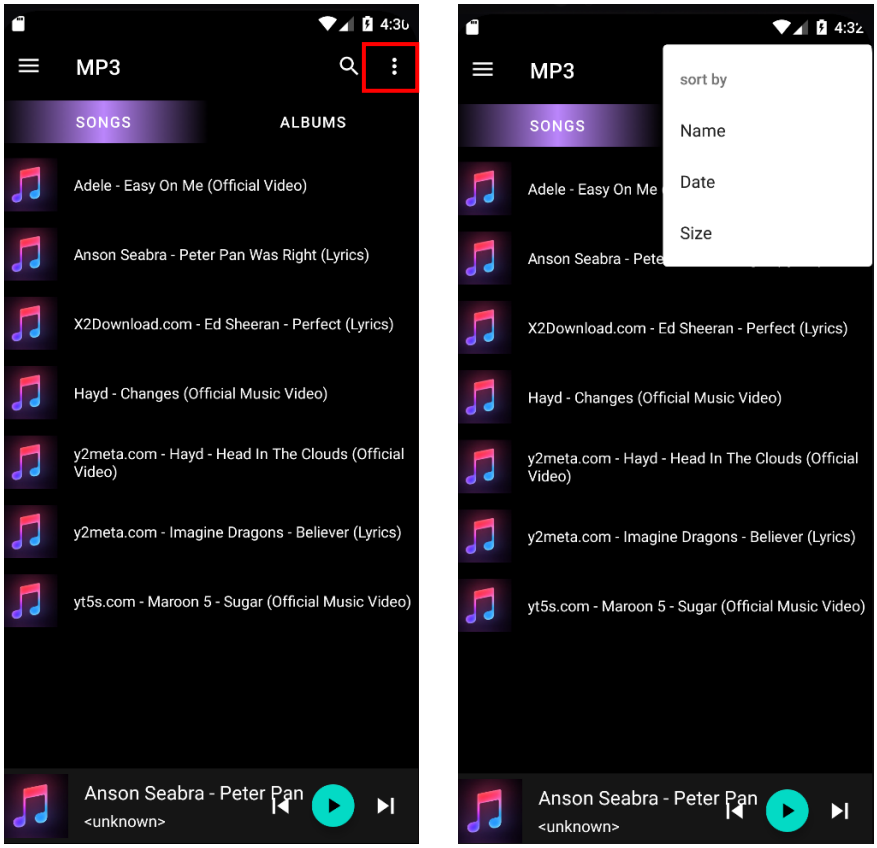


Figure 3.4.5 Sort Menu

There is a dot menu in the upper right corner of the home page, which is a sort menu. When the user clicks on it, a drop-down menu will appear with 3 options for the user to choose from. Users can choose to sort by name (alphabetically) or by date (earliest download date at the top) or by size (smallest file size first).

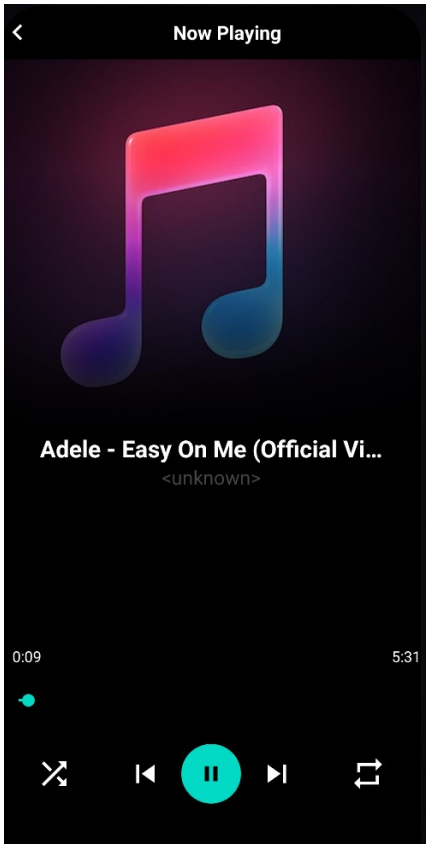


Figure 3.4.6 Music Player

When a user selects a song, whether from a song fragment or album fragment, or from a search result, they are directed to a music player page. At the top of the page is the song's cover photo. If there is no specific cover photo, the default photo is used. The user can then see the name of the song being played. On this page, the user can perform many controls. First, the user can play or pause the song by clicking the teal color button in the middle. Second, the user can skip or go to the previous song. The button on the right is to skip a song, and the button on the left is to go back to the previous song. Third, users can shuffle or loop songs by clicking the left-most and right-most buttons at the bottom of the page. The right button is for looping songs and the left button is for shuffling. Fourth, the user can drag in the seek bar to go to a specific part of the song. Finally, users can also perform gesture control on this page. To play or pause music, users can double-tap the screen. To skip to the next song, users can swipe to left on the screen, and swipe to right on the screen to go back to the previous song.

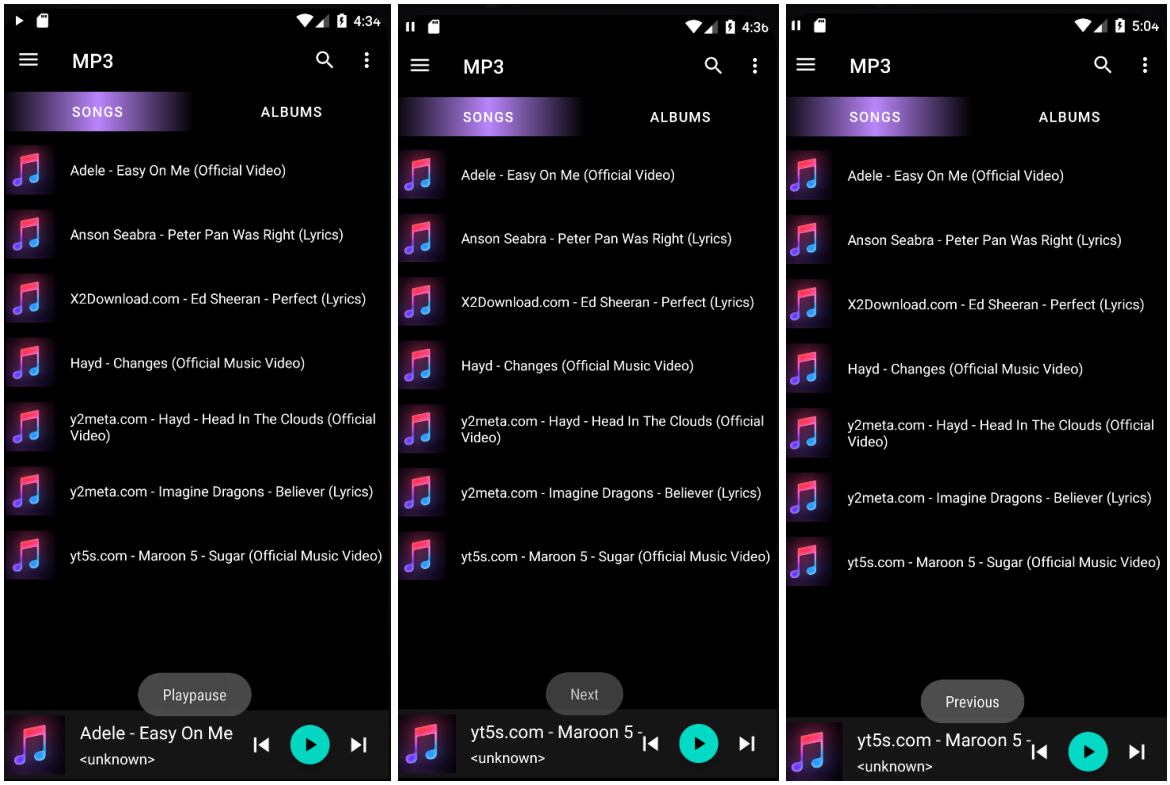


Figure 3.4.7 Bottom Music Player

When user leave the music player page, the music will still be playing. To control it, user do not need to go back to the music player page, they can just control it through the bottom music player bar (Figure 3.4.7). It can also allow user to play/pause, next and previous the song. When user clicked to button, there will be a toast message to let user know which button they have been clicked.

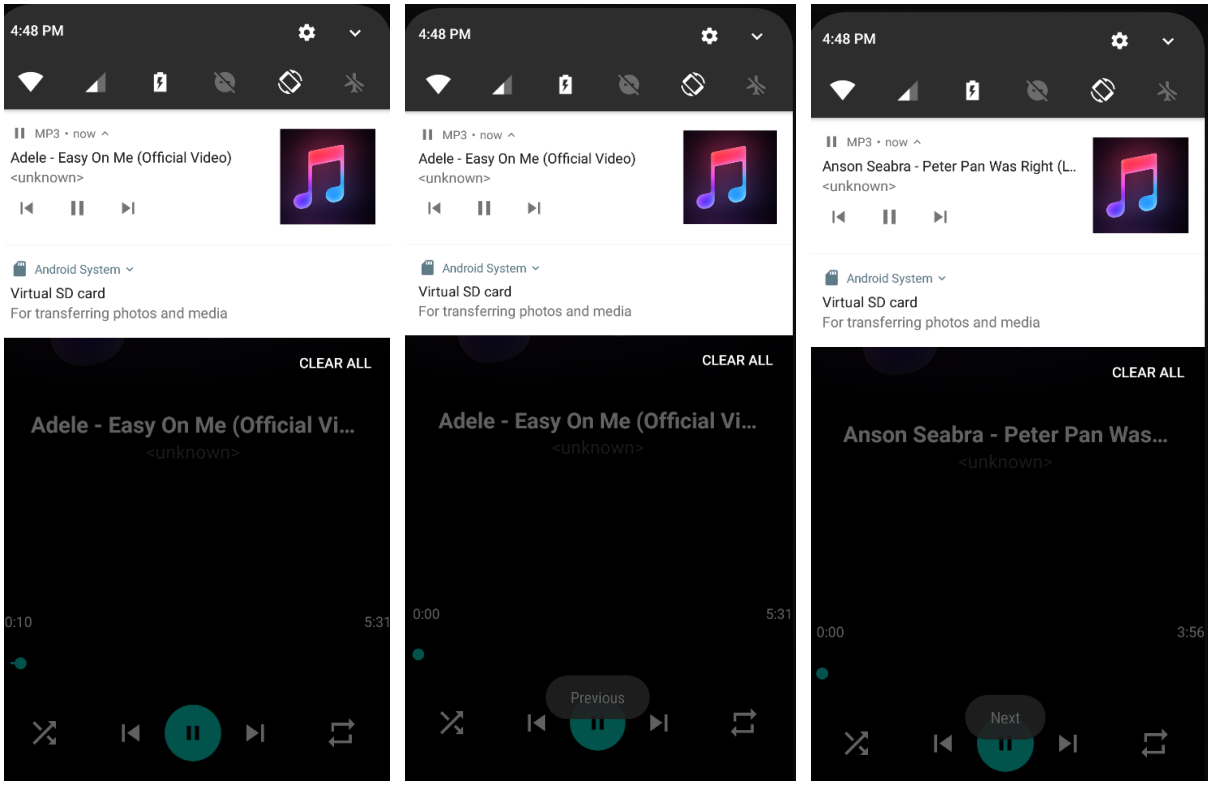


Figure 3.4.8 Notification Music Player

There will also be a notification music player in the notification bar of the phone itself when the song is playing. Users can also control songs through the notification bar, just pull down the notification bar and have tabs to control. When the user exits the app, the song will still play in the background. Well, this notification bar music player will be a great place to control your music.

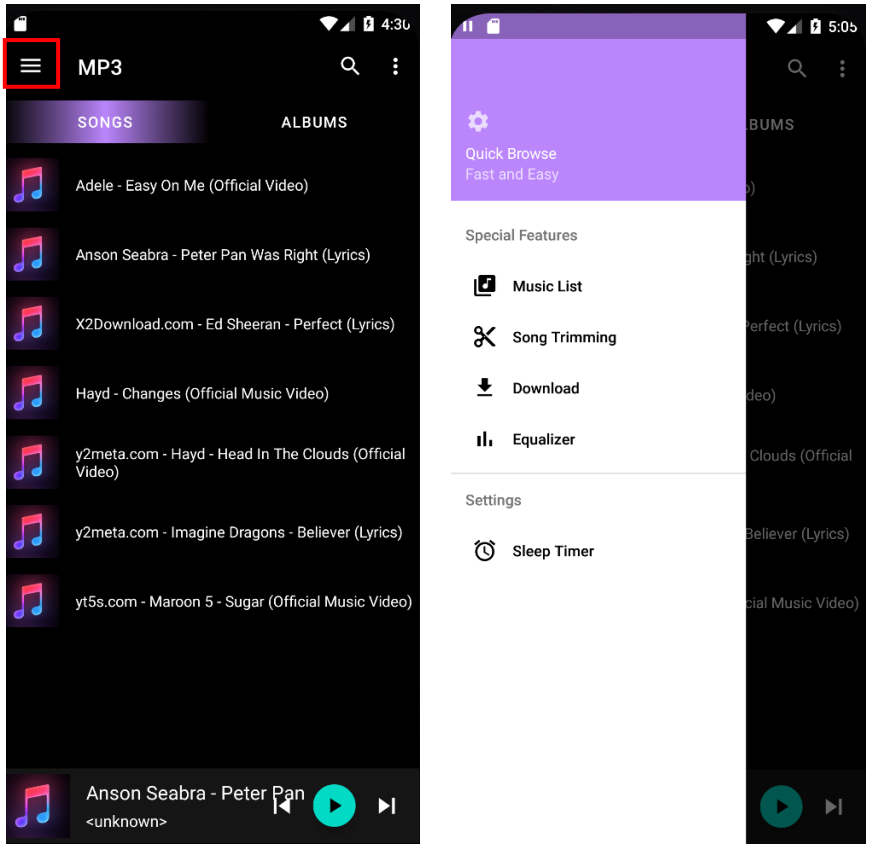


Figure 3.4.9 Navigation Drawer

On the home page users can see a three-line icon at the top left of the screen, which is a navigation drawer to easily navigate to another functional page. When the user clicks the icon, a menu drawer will appear on the left side of the screen with 2 sections, Special Features and Settings. As soon as the user selects a specific feature or setting, it directs them to the page immediately.

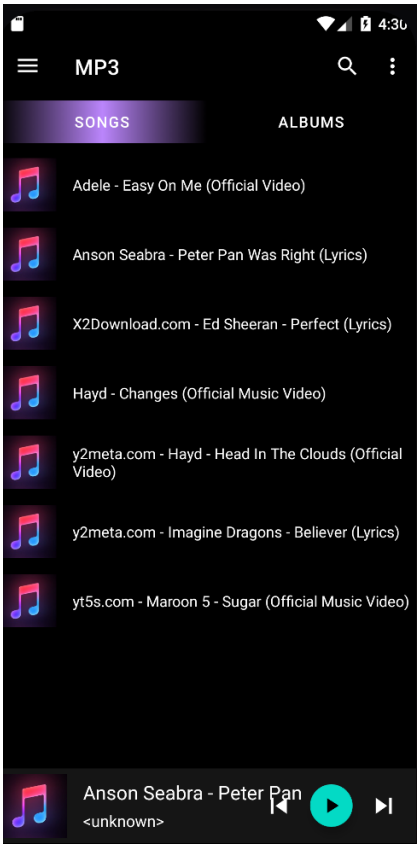


Figure 3.4.10 Music List

When user select the “Music List” in the navigation drawer, it will direct them here (Figure 3.4.10) which is the same as home page that contain all the songs that user download.

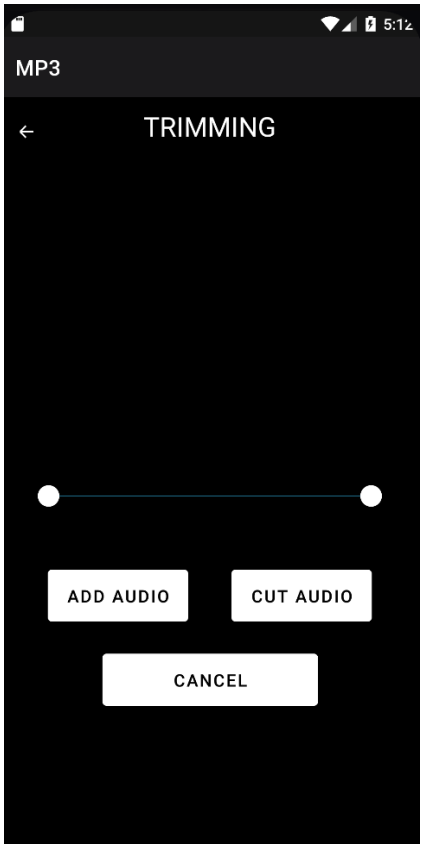


Figure 3.4.11 Song Trimming Features

When the user selects “Song Trimming” in the navigation drawer, it directs them to the trim activity page. Users can perform song trimming on their downloaded songs. First, users need to click the Add Audio button to add songs for trimming.

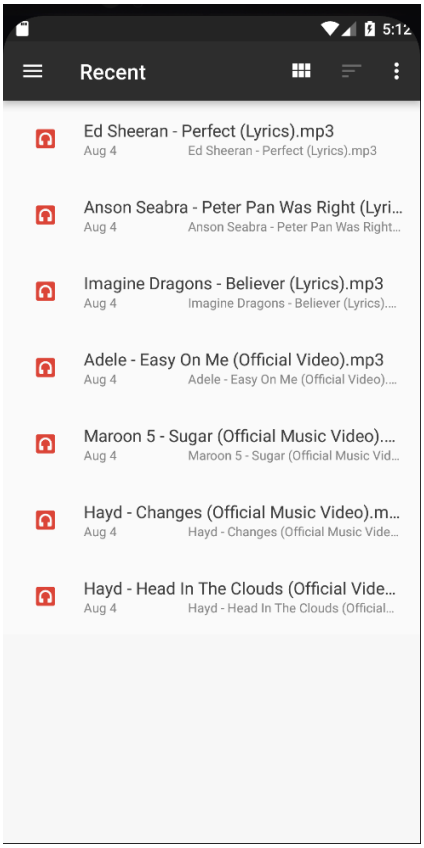


Figure 3.4.12 Select Song

After user clicked the “Add Audio” button, the system will direct them to the song directory and user can start to choose a song that they wanted to trim.

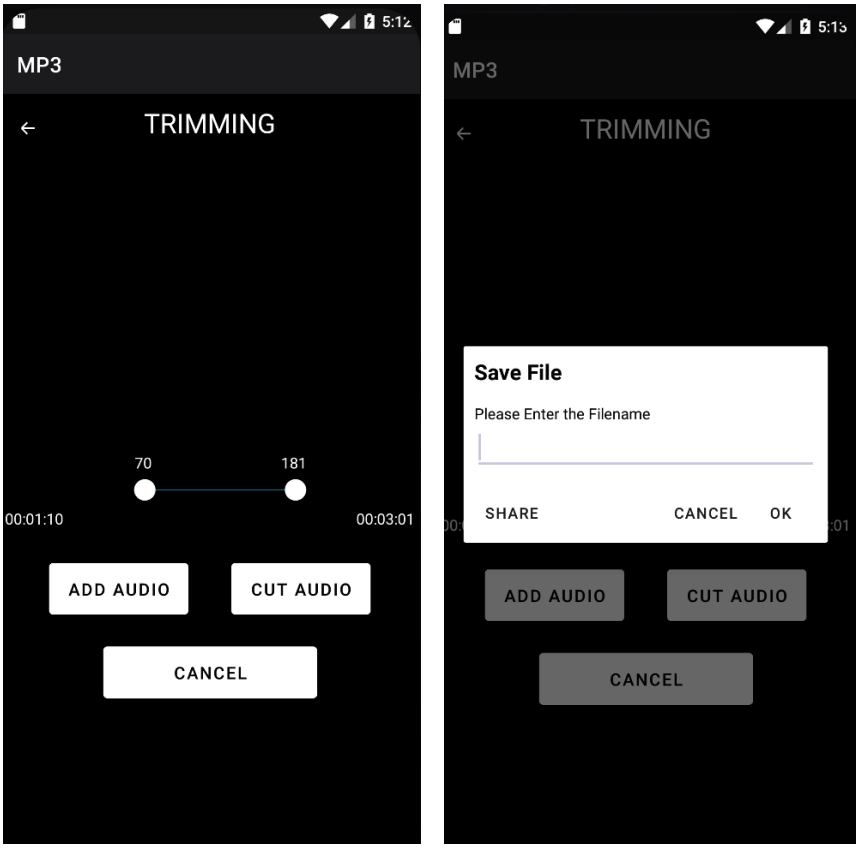


Figure 3.4.13 Trim and Save Song

After the user selects a song, the system will direct the user back to the trimming page, and the selected song will start playing. The user then drags the seek bar to trim the part they want and clicks the “Cut Audio” button to save the file. If user click the “Cut Audio” button, a dialog will pop up asking them to enter a filename to save. After that, if the user chooses “OK”, the trimmed song will be saved, but if the user chooses “Cancel”, the operation will be canceled. There’s also a “Share” button in the dialog that allows users to share the trimmed song to other places.

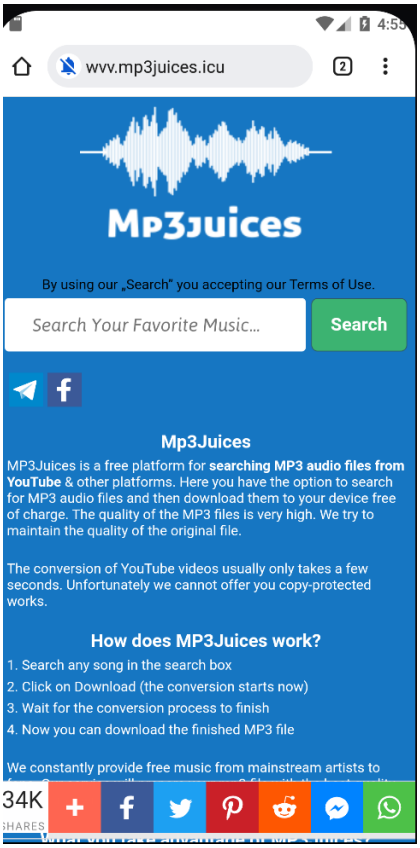


Figure 3.4.14 Download

When users select “Downloads” in the navigation drawer, they are directed to a web page, which is a third-party download site on the Internet. Users can download the song they want by searching the song title or artist name in the search bar. Once the download is complete, the app will read the downloaded songs into our Mp3 app.

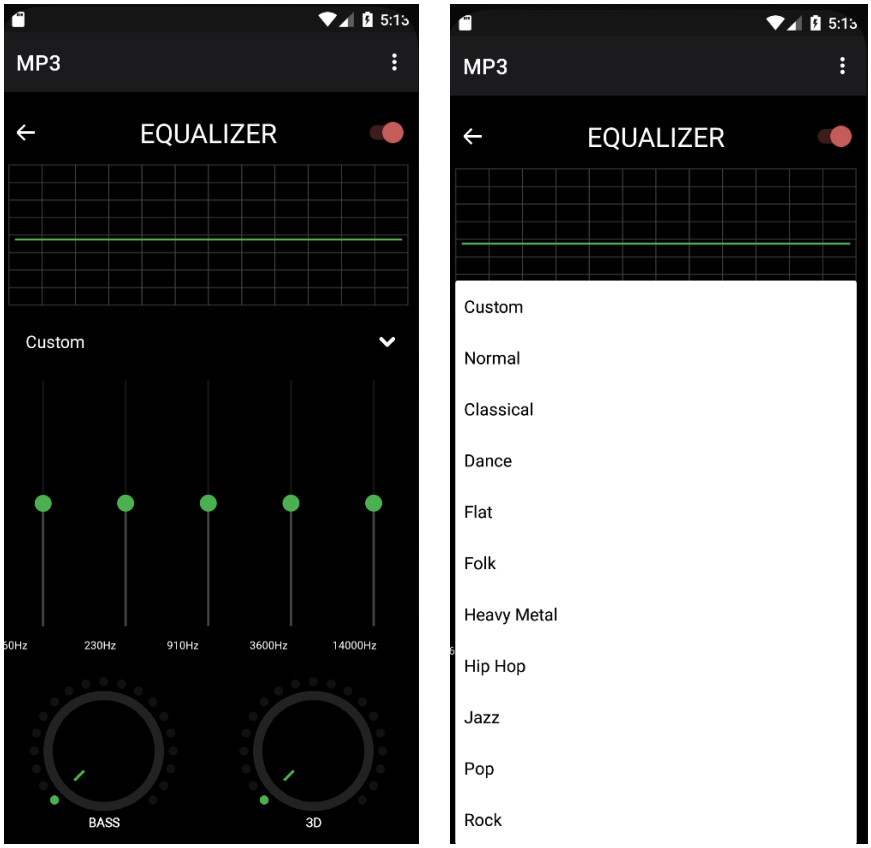


Figure 3.4.15 Equalizer

When the user selects “Equalizer” in the navigation drawer, they are directed to the equalizer page. In this page, users can change the song output according to their preferences. Users can adjust all sliders or tuners to adjust the output or select a preset via the drop-down arrow. There will be a list of output types for the user to choose from (Figure 3.4.15).

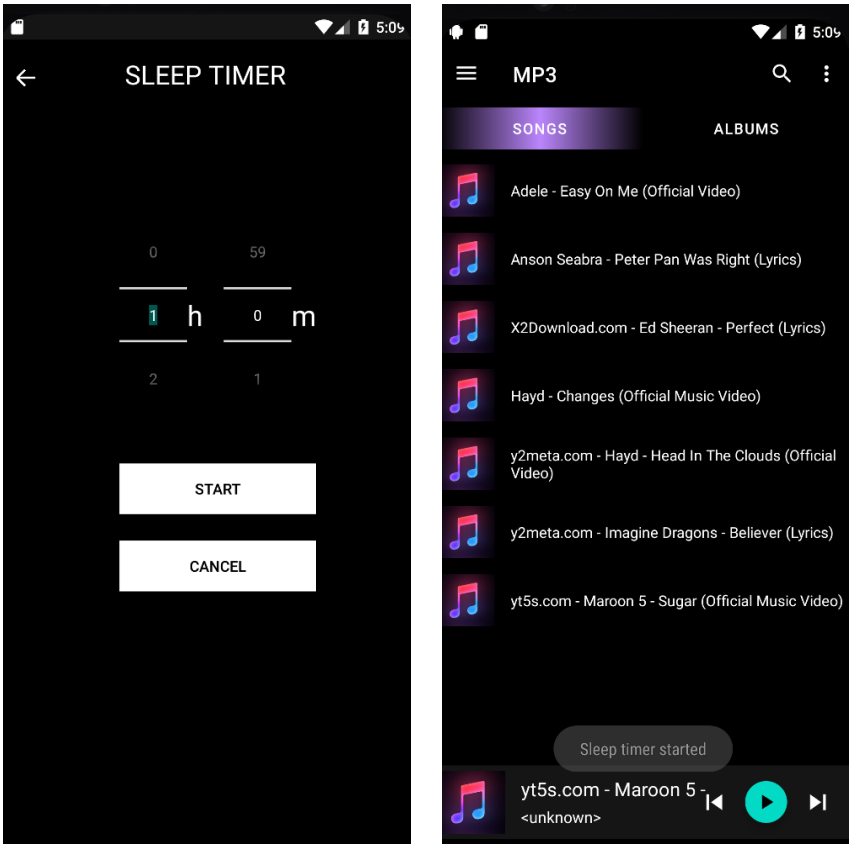


Figure 3.4.16 Set Sleep Timer and Start Countdown

When the user selects “Sleep Timer” in the navigation drawer, the system will direct them to the set sleep timer page. In this page, the user can set a sleep timer, and when the countdown is complete, the application will terminate itself. Users can set the sleep timer to set hours or minutes by sliding the number selector up and down. After adjusting the time, the user can press the “Start” button to start the countdown. If the user presses “Cancel”, the system will cancel the operation. Once the user presses the “Start” button, it will direct them back to the home page with a toast message that says, “Sleep timer started”.

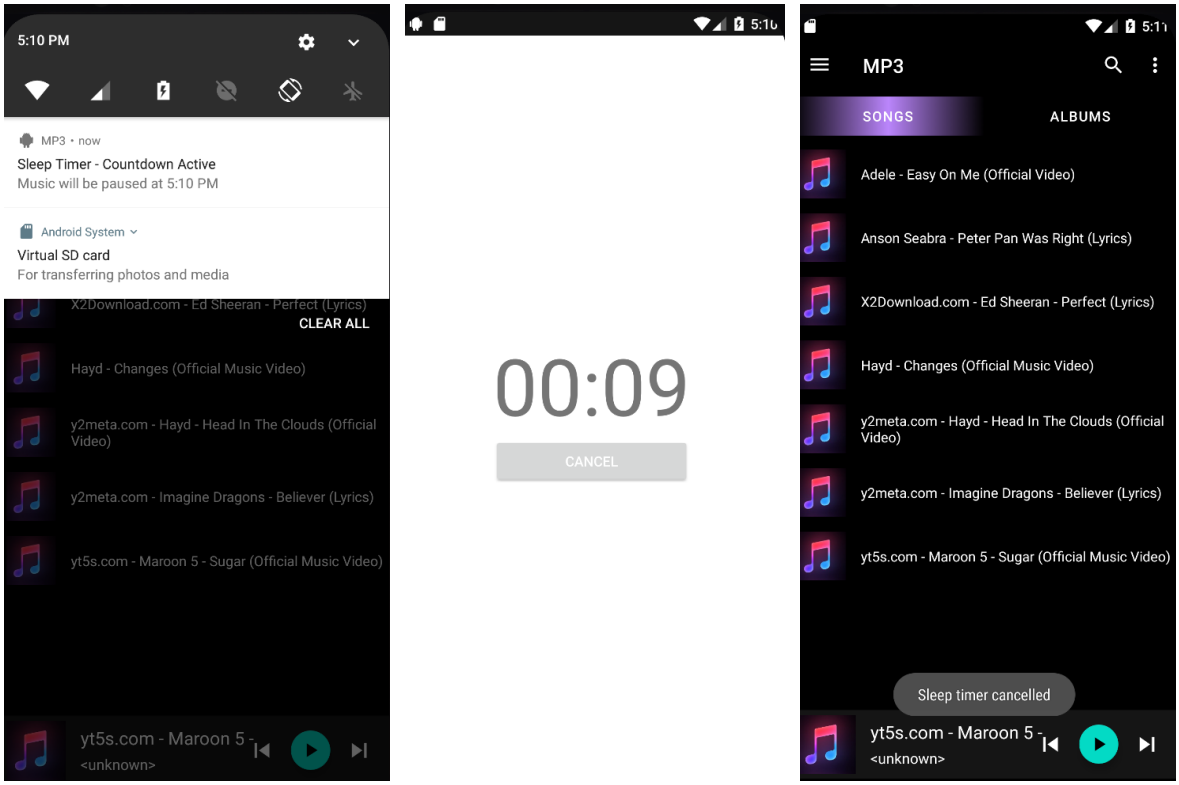


Figure 3.4.17 Notification and Countdown Cancelled

After the countdown is start, the notification bar will show the user that countdown is active, and music will be stop at what time. Also, when user select again the “Sleep timer” at the navigation bar, inside the page will change to countdown timer instead of let user to set again. User can choose to cancel the countdown by just clicking the “Cancel” button. Then, the system will direct the user back to the home page and pop a toast message that state, “Sleep Timer Cancelled”.

Chapter 4

System Methodology/Approach

The processes of the project were categorized into different phases, which were project planning, analysis, design, development and testing.

4.1 System Requirement

4.1.1 Hardware

The hardware involved in this project is laptop and Android smartphone. A laptop and smartphone are used for all the process of development and testing.

Table 4.1.1.1 Specifications of laptop

Description	Specifications
Model	Huawei MateBook 13
Processor	Intel(R) Core (TM) i5-8265U
Operating System	Windows 11
Graphic	NVIDIA® GeForce MX150
Memory	8GB RAM
Storage	256GB SSD

Table 4.1.1.2 Specifications of Smartphone

Description	Specifications
Model	Huawei Nova 3i
Processor	Hisilicon Kirin 710
Operating System	Android Version 9
Memory	4GB RAM
Storage	128GB

4.1.2 Software

- Android Studio Bumblebee (2021.1.1)

  It is an Integrated Development Environment (IDE) created primarily for the development of Android applications. Google and Jetbrains collaborated to create this compiler, which is based on Jetbrains' IntelliJ IDEA and is written in Java, Kotlin, and C++. Android Studio is accessible for free on the official website and is supported by Windows, MacOS, and Linux-based operating systems. (Android Studio, n.d.)

- Visual Paradigm

  Diagrams like as use case diagrams and flow charts are also created using this programme. This programme is available online or as downloaded software.

4.1.3 Methodology

The Agile Methods will be used in this project's process. The key reason is that, in comparison to other approaches, the Agile Model is more adaptable to development processes. The application's functions can be separated into time boxes to supply certain features for a release. As a result, the project will be able to develop function by function before combining all of the functions to complete the project. Also, Agile Methods is suitable for those who don't have a detailed specification and design before moving to implementation which will be frequently changed. As our system are considered as a small project and not going to outsource to other company so Agile Methods is the most effective method.

Figure 4.1.3.1 Agile Method (Maanas, n.d.)

There are six phase in agile methodology which are requirement elicitation, planning, design, implementation and development, testing, and deployment.

Requirement Elicitation

In this phase, we are going to find out what users really need by review the existing MP3 Music Player on the market now. Also, we collect as many as the idea, comment, and suggestion through the existing application to make the improvement on next.

Planning

Agile planning is all about determining how long it will take to complete a given job and to complete an entire project. It takes a project and divides it into sprints to do this. Project objective and project scope will also be determined at this phase.

Design

In design phase, we will come out with UML diagram and prototype that is useful for implementation phase.

Implementation and Development

In implementation phase, the prototype is installed. Because of the preparation done in the preceding phases, this is usually the quickest phase. Prototypes are being developed by using the software and hardware mentioned above.

Testing

In testing phase, we are going to test each component that we have deployed in the previous stage to ensure all the components are perform well and meets the needs. We can run the testing this by using some approaches such as Black Box Testing.

Deployment

We will start making this app available to the public in the final step. For example, in order to make it available to all the public, we will upload this MP3 music player app to the Google Play Store or share the app link to various social media. We expect users to experience unexpected issues with the player along the way, which we will address in future releases.

4.1.4 User Requirements

Functional Requirements:

- As a user, I can play the song
- As a user, I can skip the song
- As a user, I can download the song
- As a user, I can play the songs in background
- As a user, I can adjust the output of a song
- As a user, I can set sleep timer
- As a user, I can trim the song
- As a user, I control the flow using gesture
- As a user, I can search for specific song
- As a user, I can sort the music list

Non-Functional Requirements:

- This application will operate in Android OS
- Respond time must be less than 10 second
- Well organised user interface

4.1.5 System Performance Definition

A number of criteria must be satisfied in order to achieve the system performance required by the project, which are given below:

- Response Time

  Response times should normally be as quick as feasible. The most comfortable response time interval is 0.1 - 1 second. People may adjust to shorter response times, but they are often unsatisfied with delays greater than 2 seconds. (WAPT, n.d.)

- Different Devices Display and Performance

  The MP3 music player application must ensure that it can be used and installed properly and smoothly on a variety of devices. For example, the hardware and operating system of each phone will change, as Samsung and Huawei phones are Android10 and EMUI based on Android, respectively, necessitating testing on several devices. Furthermore, the application should be adaptable to different devices to ensure that no mistakes occur during execution and that different results are obtained on different devices.

4.1.6 Timeline

ID	Task Name	Duration Days	Start Date	End Date	Week 1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	**Planning and Requirement Specification**	30	24/1/2022	25/2/2022														
1.1	Review Similar Application	7	24/1/2022	31/1/2022														
1.2	Determine the Problem Statement	7	1/2/2022	8/2/2022														
1.3	Define Project Scope	7	9/2/2022	16/2/2022														
1.4	Determine Project Objective	7	9/2/2022	16/2/2022														
1.5	Gantt Chart	4	17/2/2022	21/2/2022														
1.6	Proposed Method / Approach	3	22/2/2022	25/2/2022														
1.7	Part 1 Completed	0	25/2/2022	25/2/2022														
2	**Analysis**	14	26/2/2022	12/3/2022														
2.1	Literature Review	14	26/2/2022	12/3/2022														
2.2	Revise the Problem Statement & Project Objective	14	26/2/2022	12/3/2022														
3	**Design**	14	13/3/2022	27/3/2022														
3.1	Develop Flowchart	5	13/3/2022	18/3/2022														
3.2	Develop Use-Cases Diagram	8	19/3/2022	27/3/2022														
3.3	Use-Cases Description	8	19/3/2022	27/3/2022														
3.4	Part 2 Completed	0	27/3/2022	27/3/2022														
4	**Development**	13	28/3/2022	10/4/2022														
4.1	Interface Design and Basic User Interaction	13	28/3/2022	10/4/2022														
4.2	Develop Basic Function	13	28/3/2022	10/4/2022														
5	**Testing**	8	4/4/2022	12/4/2022														
5.1	Perform Application Testing	8	4/4/2022	11/4/2022														
5.2	Discuss Issue Found	8	4/4/2022	11/4/2022														
5.3	Provide Solution to Issue Found	8	4/4/2022	11/4/2022														
5.4	Part 3 Completed	0	12/4/2022	12/4/2022														
6	**Delivery**	2	13/4/2022	15/4/2022														
6.1	Prepare Presentation	2	13/4/2022	14/4/2022														
6.2	Presentation and Demo	0	15/4/2022	15/4/2022														

Figure 3.4.1 Gantt Chart for FYP1

ID	Task Name	Duration Days	Start Date	End Date	Week 1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Review Previous Work	7	13/6/2022	19/6/2022														
2	Design Prototype 2	7	20/6/2022	26/6/2022														
3	Develop Prototype	14	27/6/2022	10/7/2022														
4	Testing and Debugging On Prototype	7	11/7/2022	17/7/2022														
5	Solution for Issue Found	7	18/7/2022	24/7/2022														
6	Finalize Application Prototype	7	25/7/2022	31/7/2022														
7	Full System Development	21	1/8/2022	21/8/2022														
8	Final Application Testing	7	22/8/2022	28/8/2022														
9	Finalize Report	7	29/8/2022	4/9/2022														
10	Report Submission	7	5/9/2022	11/9/2022														
11	Presentation	7	12/9/2022	18/9/2022														

Figure 3.4.2 Gantt Chart for FYP2

Chapter 5
Implementation and Testing

5.1 Implementation

When the testing stage of the proposed application's development was complete, it should move on to the deployment stage. During the deployment stage, the developer must upload the "APK" file, the installation package for the application, to a site like Google Play Store so that consumers may download it. However, there are still many modules that need to be updated and enhanced because the number of users is still limited, and the planned application is not in its final public form. Therefore, upon the publication of the final public version, it will be published to the appropriate platform for promotion. Moreover, users can run the programme without a network, but the “download” features need Internet connection to access to the appropriate web page where they can download the song.

The following steps outline how a new user can run the application:

1. When users first launch the apps, there is a dialog pop-up to ask permission from the user. The users must grant the suggested application the necessary permissions before it can read local songs from the phone and add them to the song playlist.

2. There are two fragments on the home page/music list page: album fragment and song fragment users can choose a song to play in either of these two fragments. To play a song, the users just click a specific song in the playlist

3. In the music player interface, the users are allowed to control the music flow through icon button such as play/pause, progress bar and so on or using gesture control.

4. Users go back to the home page, and they can select the “Song Trimming” feature from the left navigation bar by clicking the three-line icon on the top left corner of the page.

5. Users press the “Add Audio” button first to add a song from the local storage into the apps to perform trimming

6. Users drag the seek bar to set the starting point and the ending point of the song and click the “Cut Audio” button.

7. After the “Cut Audio” button is clicked, the users can enter the filename to save the trimmed song to the external storage.

8. Furthermore, users also can open the navigation drawer to perform download, equalizer, set sleep timer features.

5.2 Testing

Unit Testing 1: Music List

Test Objective: To guarantee that the user's chosen song can play normally, the details for that song can be seen normally, and the song playlist can be imported and shown appropriately

Input	Expected Output	Actual Output
Select “Songs” Fragment	The music list under the “Songs” fragment are read in successfully and contain all the download song	pass
Select “Albums” Fragment	The music list under the “Albums” fragment are read in successfully and contain all the download song categorize by albums	pass
Select any song from “Songs” fragment’s list	Select any song from the “Songs” fragment music list and it can direct to music player and play the selected music	pass
Select any song from “Albums” fragment’s album’s music list	Select any song from the “Albums” fragment music list and it can direct to music player and play the selected music	pass
Play any song and click the simulator’s home button to make the app run in the background	The songs are still playing in the background after user exits the app	pass
Select the magnifier icon	User do search by enter the artist or song name in the search bar and the correct result display for the user	pass

Table 5.2.1 Unit Testing of Music List Features

Unit Testing 2: Sort List

Test Objective: To guarantee that the music list is sort appropriate based on user's choice

Input	Expected Output	Actual Output
Select the 3-dot icon	List a drop-down menu to let user choose how to sort the music list whether by name, date or size	pass
Select "name" from the drop-down menu	The music list is successfully sort by name which follow the alphabetical order	pass
Select "date" from the drop-down menu	The music list is successfully sort by date which the smallest date will come first	pass
Select "size" from the drop-down menu	The music list is successfully sort by size which the smallest file will be on the top of the list	pass

Table 5.2.2 Unit Testing of Sort List Features

Unit Testing 3: Music Player

Test Objective: To guarantee that all the playback control buttons inside the music player interface are functional and operate as intended

Input	Expected Output	Actual Output
Click the "PlayPause" button	Play the song and update play the button to pause button or pause the current playing song and update the pause button to play button	pass
Click the "Skip" button	Skip to the next song by followed the music list's sequences (if user does not choose shuffle play)	pass
Click the "Previous" button	Go back to the previous song by followed the music list's sequences (if user does not choose shuffle play)	pass
Click the "Shuffle" button	Play the song by not follow the sequences of the music list	pass
Click the "Loop" button	Keep repeat playing a same song	pass
Drag the seek bar	Drag the seek bar and the time of the song will be update and the song will jump to the following time dragged	pass

Table 5.2.3 Unit Testing of Music Player Features

Unit Testing 4: Gesture Control

Test Objective: To guarantee that all gesture control perform in music player interface are working properly

Input	Expected Output	Actual Output
Swipe to left on the screen	Will skip to the next song	pass
Swipe to right on the screen	Will go back to the previous song	pass
Double tap on the screen	The music will play or pause	pass

Table 5.2.4 Unit Testing of Gesture Control Features

Unit Testing 5: Song Trimming

Test Objective: To guarantee that the selected song can be trim and save

Input	Expected Output	Actual Output
Click the “Add Audio” button	Direct user to choose the song to trim	pass
Click the “Cut Audio” button	Display the pop-up dialog	pass
Click “Yes” on the dialog box	Save the file to external storage and the filename set to the name user type at the textbox	pass
Click “Cancel” on the dialog box	Close the pop-up dialog	pass
Click “Share” on the dialog box	Display bottom pop-up dialog and allow the user to choose where to share the trimmed song	pass
Click the “Cancel” button	Cancel the trim action and back to home page	pass
Drag the seek bar	The time update and the song play within the start point and end point	pass

Table 5.2.5 Unit Testing of Song Trimming Features

Unit Testing 6: Download

Test Objective: To guarantee that the user can be direct to the download site and download the song

Input	Expected Output	Actual Output
Click "Download" in the navigation drawer	Direct user to third-party download page	pass

Table 5.2.6 Unit Testing of Download Features

Unit Testing 7: Equalizer

Test Objective: To guarantee that the output of the song can be change based on user setting or adjustment

Input	Expected Output	Actual Output
Click the drop-down icon	Display the pre-set output type to let user choose	pass
Spin the tuner	The tuner can be spin successfully and the output sound will be change	pass
Drag the seek bar	The seek bar can be drag successfully and the output of sound will be change	pass
Song output type	The graph on the top of the screen will successfully change based on the different option and settings	pass

Table 5.2.7 Unit Testing of Equalizer Features

Unit Testing 8: Sleep Timer

Test Objective: To guarantee that the output of the song can be change based on user setting or adjustment

Input	Expected Output	Actual Output
Pull the number picker of hours	Display and stay at the number the user pulled	pass
Pull the number picker of minutes	Display and stay at the number the user pulled	pass
Click the “Start” button	The countdown is start. There will be a toast message display on the home screen and there is also a notification display in the simulator’s notification bar	pass
Click the “Cancel” button	Discard the action and go back to home screen	pass
After sleep timer is set, go back to the page again and click “Cancel” button	The countdown will be stopped and there will be a toast message display on the home screen saying that the countdown had cancelled	pass

Table 5.2.8 Unit Testing of Sleep Timer Features

Unit Testing 9: Bottom Music Player

Test Objective: To guarantee that all the playback control buttons in the bottom music player is functional and operate as intended

Input	Expected Output	Actual Output
Click the “PlayPause” button	Play the song and update play the button to pause button or pause the current playing song and update the pause button to play button.	pass
Click the “Skip” button	Skip to the next song by followed the music list’s sequences (if user does not choose shuffle play in music player page)	pass
Click the “Previous” button	Go back to the previous song by followed the music list’s sequences (if user does not choose shuffle play in music player page)	pass
Select a song from music list	Play the song that are selected by user and there is same with the song that playing in the music player page	pass

Table 5.2.9 Unit Testing of Bottom Music Player

Unit Testing 10: Notification Bar's Music Player

Test Objective: To guarantee that all the playback control buttons in the notification bar's music player is functional and operate as intended

Input	Expected Output	Actual Output
Click the "PlayPause" button	Play the song and update play the button to pause button or pause the current playing song and update the pause button to play button.	pass
Click the "Skip" button	Skip to the next song by followed the music list's sequences (if user does not choose shuffle play in music player page)	pass
Click the "Previous" button	Go back to the previous song by followed the music list's sequences (if user does not choose shuffle play in music player page)	pass
Select a song from music list	Play the song that are selected by user and there is same with the song that playing in the music player page	pass
Exit the application	The notification music player is still at the notification bar and the music is still continue playing	pass

Table 5.2.10 Unit Testing of Notification Bar's Music Player

Chapter 6
Conclusion

6.1 Project Review, Discussions and Conclusion

Will people choose to use tedious software or interesting software? I believe that people nowadays are more pursuing special things, and they all want to customize some things according to their own preferences to meet their own needs. The tedious software is fine for the first use, but over time the user will lose the freshness and gradually move to better and more functional software.

Most people usually don't keep their phone's screen awake all the time, as this drains the battery faster. Also, it is difficult for users to focus on the application to make control when they are too busy with something. For example, when users are driving or riding, it is difficult for them to aim for buttons to control. If they turn their attention to their phone, then they could cause an accident. As we know, a lot of accidents now happen because of drivers playing on their phones when they are driving.

Today's people choose to use the MP3 software of mobile phones to connect Bluetooth to various devices more often than to choose radio stations. They choose to complete various procedures in one software, such as downloading music and playing music in the same software. If separated into different software will make the whole process cumbersome. Also, if a software only has some kinds of songs, it will also bring inconvenience to the user. Whether it is an older song or the latest trending song, it must be easy for users to download.

In conclusion, the proposed application will combine the strengths of majority of music players currently available in the market and eliminate some unrealistic features, allowing users to have a user experience when using the MP3 music player application. The proposed MP3 music player will focus on improving the experience of users of the music player experience.

6.1.1 Project Achievement

First, the proposed music player had achieved its first objective which is add-on special features. The special features in this proposed music player are song trimming and equalizer. Two of these features are rarely can be seen in most of the music player in the market. By implemented this feature, it allows user to have more interactivity with the application and also make the music player more unique and interesting. Song trimming allow user to trim a specific song to the length or to the part they wanted, and equalizer allow user to change the output of the selected song. These two features are more on user customize based on their needs.

Second, the proposed music player had achieved its second objective which is add in gesture control. By using gesture control, user doesn't have to pay full attention to the phone when they wanted to control the music flow in an inconvenience situation such as driving. This function eliminated the risk of accident for user when they are using the music player application which is a win-win situation. User just simply swipes right or left or double tap within the phone's screen then it will successfully trigger the function and the effect is same as using button to control.

Third, the proposed music player had achieved its third objective which is expand song collection library. Now the user can download their favorite song no matter in what genre. This is because the third-party download site provides a large size of the song library and all genre of song include classic, pop, jazz and others.

Lastly, the proposed music player had achieved its fourth objective which is provide download function. The download function allow user to download the song they wanted by just clicking the "Download" button. Although, the download function are no in-apps download but it is also making user easy to get a music in one app. They do not need to find another place to download the music they wanted, which is a time-wasting process.

6.1.2 Problem Encountered

The main problem encounter in this project is unable to implement the in-apps download function. We are just using a third part download site from web. Although all the songs are available at the site but there are all illegal as in copyright. To solve this problem, we need to have a stronger assets and knowledge. So, for this project we are temporary using the third-party site.

6.2 Future Work

I. Implementing in-app song downloads as opposed to the present practice of using a third-party website which is some sort of illegal.

II. Refactoring code involves rebuilding the coding structure to make it clearer, simpler to understand, and more effective.

III. Build more special function to make the application more interesting and enhance the interactivity.

IV. Cross-platform, making the app available on iOS and Android.

REFERENCES

Suzanne, S. (2020, March). *When to Design with Gestures vs. Buttons*. [Online]. Available at: <https://www.telerik.com/blogs/when-to-design-with-gestures-vs-buttons> [Accessed 17 June 2022]

Edwin, Y. (2015). *WeChat's Tencent enters music-streaming fray with Joox*. [Online]. Digital News Asia. Available at: <https://www.digitalnewsasia.com/personal-tech/wechat-tencent-enters-music-streaming-fray-with-joox> [Accessed 20 June 2022]

Emily, S. (2006). *Feeling the beat: Symposium explores the therapeutic effects of rhythmic music*. [Online]. Available at: <https://news.stanford.edu/news/2006/may31/brainwave-053106.html> [Accessed 13 June 2022]

Wikipedia, 2021. *JOOX*. [Online]. Available at: <https://en.wikipedia.org/wiki/Joox#cite_note-6> [Accessed 20 June 2022]

Maanas, n.d. Agile software development methodology. [Online]. Available at: <Agile Software Development Methodology Stock Vector (Royalty Free) 1232798980 (shutterstock.com)> [Accessed 24 June 2022]

WAPT, n.d.. Response Time. [Online]. Available at: <https://www.loadtestingtool.com/help/response-time.shtml> [Accessed 24 June 2022]

Android Studio, n.d.. Meet Android Studio. [Online]. Available at: <https://developer.android.com/studio/intro> [Accessed 24 June 2022]

Spotify For the Record, n.d. Company Info. [Online]. Available at: <Spotify — About Spotify> [Accessed 20 June 2022]

Google Play (2021). Play Music – MP3 Music Player. [Online]. Available at: < https://play.google.com/store/apps/details?id=com.google.android.apps.youtube.music&hl=en&gl=US > [Accessed 20 June 2022]

APPENDICES

A.1 ActionPlaying.java

```
package com.example.mp3;

public interface ActionPlaying
{
    void playpausebtnCliked();
    void prevbtnCliked();
    void nextbtnCliked();
}
```

A.2 AlbumAdapter.java

```
package com.example.mp3;

import android.content.Context;
import android.content.Intent;
import android.media.MediaMetadataRetriever;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;

import java.util.ArrayList;

public class AlbumAdapter extends
RecyclerView.Adapter<AlbumAdapter.MyHolder>
{
    private Context mContext;
    private ArrayList<MusicFiles> albumFile;
    View view;

    public AlbumAdapter(Context mContext, ArrayList<MusicFiles> albumFile)
{
        this.mContext = mContext;
        this.albumFile = albumFile;
    }

    @NonNull
    @Override
    public MyHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType)
    {
        view = LayoutInflater.from(mContext).inflate(R.layout.albumitem,
parent, false);
        return new MyHolder(view);
    }

    @Override
```

```
    public void onBindViewHolder(@NonNull MyHolder holder, int position)
    {
        holder.albumname.setText(albumFile.get(position).getAlbum());
        byte[] image = getAlbumArt(albumFile.get(position).getPath());
        if(image != null)
        {
            Glide.with(mContext).asBitmap()
                    .load(image)
                    .into(holder.albumimage);
        }
        else
        {
            Glide.with(mContext)
                    .load(R.drawable.musicpic)
                    .into(holder.albumimage);
        }
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(mContext, AlbumDetails.class);
                intent.putExtra("albumName",
albumFile.get(position).getAlbum());
                mContext.startActivity(intent);
            }
        });
    }

    @Override
    public int getItemCount() {
        return albumFile.size();
    }

    public class MyHolder extends RecyclerView.ViewHolder
    {
        ImageView albumimage;
        TextView albumname;

        public MyHolder(@NonNull View itemView)
        {
            super(itemView);
            albumimage = itemView.findViewById(R.id.albumimage);
            albumname = itemView.findViewById(R.id.albumname);
        }
    }

    private byte[] getAlbumArt(String uri)
    {
        MediaMetadataRetriever retriever = new MediaMetadataRetriever();
        retriever.setDataSource(uri);
        byte[] art = retriever.getEmbeddedPicture();
        retriever.release();
        return art;

    }
}
```

A.3 AlbumDetails.java

```
package com.example.mp3;

import static com.example.mp3.MainActivity.musicFiles;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.media.MediaMetadataRetriever;
import android.os.Bundle;
import android.widget.ImageView;

import com.bumptech.glide.Glide;

import java.util.ArrayList;

public class AlbumDetails extends AppCompatActivity {

    RecyclerView recyclerView;
    ImageView albumPhoto;
    String albumName;
    ArrayList<MusicFiles> albumSongs = new ArrayList<>();
    AlbumDetailsAdapter albumDetailsAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_album_details);
        recyclerView = findViewById(R.id.recycleview);
        albumPhoto = findViewById(R.id.albumphoto);
        albumName = getIntent().getStringExtra("albumName");
        int j = 0;
        for (int i = 0; i < musicFiles.size(); i++)
        {
            if(albumName.equals(musicFiles.get(i).getAlbum()))
            {
                albumSongs.add(j, musicFiles.get(i));
                j++;
            }
        }
        byte[] image = getAlbumArt(albumSongs.get(0).getPath());
        if (image != null)
        {
            Glide.with(this)
                    .load(image)
                    .into(albumPhoto);
        }
        else
        {
            Glide.with(this)
                    .load(R.drawable.musicpic)
                    .into(albumPhoto);
        }
    }

    @Override
    protected void onResume() {
        super.onResume();
```

```
        if (!(albumSongs.size() < 1))
        {
            albumDetailsAdapter = new AlbumDetailsAdapter(this, albumSongs);
            recyclerView.setAdapter(albumDetailsAdapter);
            recyclerView.setLayoutManager(new LinearLayoutManager(this,
RecyclerView.VERTICAL, false));
        }
    }

    private byte[] getAlbumArt(String uri)
    {
        MediaMetadataRetriever retriever = new MediaMetadataRetriever();
        retriever.setDataSource(uri);
        byte[] art = retriever.getEmbeddedPicture();
        retriever.release();
        return art;

    }
}
```

A.4 AlbumDetailsAdapter.java

```
package com.example.mp3;

import android.content.Context;
import android.content.Intent;
import android.media.MediaMetadataRetriever;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;

import java.util.ArrayList;

public class AlbumDetailsAdapter extends
RecyclerView.Adapter<AlbumDetailsAdapter.MyHolder>
{
    private Context mContext;
    static ArrayList<MusicFiles> albumFile;
    View view;

    public AlbumDetailsAdapter(Context mContext, ArrayList<MusicFiles>
albumFile) {
        this.mContext = mContext;
        this.albumFile = albumFile;
    }

    @NonNull
    @Override
    public MyHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType)
    {
        view = LayoutInflater.from(mContext).inflate(R.layout.music_items,
parent, false);
        return new MyHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull MyHolder holder, final int
position)
    {
        holder.albumname.setText(albumFile.get(position).getTiles());
        byte[] image = getAlbumArt(albumFile.get(position).getPath());
        if(image != null)
        {
            Glide.with(mContext).asBitmap()
                    .load(image)
                    .into(holder.albumimage);
        }
        else
        {
            Glide.with(mContext)
                    .load(R.drawable.musicpic)
                    .into(holder.albumimage);
```

```
        }
        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(mContext, PlayerActivity.class);
                intent.putExtra("sender", "albumDetails");
                intent.putExtra("position", position);
                mContext.startActivity(intent);
            }
        });
    }

    @Override
    public int getItemCount() {
        return albumFile.size();
    }

    public class MyHolder extends RecyclerView.ViewHolder
    {
        ImageView albumimage;
        TextView albumname;

        public MyHolder(@NonNull View itemView)
        {
            super(itemView);
            albumimage = itemView.findViewById(R.id.music_img);
            albumname = itemView.findViewById(R.id.musicfilename);
        }
    }

    private byte[] getAlbumArt(String uri)
    {
        MediaMetadataRetriever retriever = new MediaMetadataRetriever();
        retriever.setDataSource(uri);
        byte[] art = retriever.getEmbeddedPicture();
        retriever.release();
        return art;

    }
}
```

A.5 AlbumsFragment.java

```
package com.example.mp3;

import static com.example.mp3.MainActivity.albums;
import static com.example.mp3.MainActivity.musicFiles;

import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.GridLayoutManager;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class AlbumsFragment extends Fragment {

    RecyclerView recyclerView;
    AlbumAdapter albumAdapter;

    public AlbumsFragment() {
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_albums, container,
false);
        recyclerView = view.findViewById(R.id.recycleview);
        recyclerView.setHasFixedSize(true);
        if(!(albums.size() < 1))
        {
            albumAdapter = new AlbumAdapter(getContext(), albums);
            recyclerView.setAdapter(albumAdapter);
            recyclerView.setLayoutManager(new
GridLayoutManager(getContext(), 2));
        }
        return view;
    }
}
```

A.6 AnalogController.java

```
package com.example.mp3;

import android.view.View;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.MotionEvent;

public class AnalogController extends View {

    float midx, midy;
    Paint textPaint, circlePaint, circlePaint2, linePaint;
    String angle;
    float  currdeg, deg = 3, downdeg;

    int progressColor, lineColor;

    onProgressChangedListener mListener;

    String label;

    public interface onProgressChangedListener {
        void onProgressChanged(int progress);
    }

    public void setOnProgressChangedListener(onProgressChangedListener
listener) {
        mListener = listener;
    }

    public AnalogController(Context context) {
        super(context);
        init();
    }

    public AnalogController(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public AnalogController(Context context, AttributeSet attrs, int
defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }

    void init() {
        textPaint = new Paint();
        textPaint.setColor(Color.WHITE);
        textPaint.setStyle(Paint.Style.FILL);
        textPaint.setTextSize(33);
        textPaint.setFakeBoldText(true);
        textPaint.setTextAlign(Paint.Align.CENTER);
        circlePaint = new Paint();
        circlePaint.setColor(Color.parseColor("#222222"));
        circlePaint.setStyle(Paint.Style.FILL);
```

```
        circlePaint2 = new Paint();
        circlePaint2.setColor(EqualizerFragment.themeColor);
        circlePaint2.setStyle(Paint.Style.FILL);
        linePaint = new Paint();
        linePaint.setColor(EqualizerFragment.themeColor);
        linePaint.setStrokeWidth(7);
        angle = "0.0";
        label = "Label";
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        midx = canvas.getWidth() / 2;
        midy = canvas.getHeight() / 2;

        int   ang    = 0;
        float x      = 0, y = 0;
        int   radius = (int) (Math.min(midx, midy) * ((float) 14.5 / 16));
        float deg2   = Math.max(3, deg);
        float deg3   = Math.min(deg, 21);
        for (int i = (int) (deg2); i < 22; i++) {
            float tmp = (float) i / 24;
            x = midx + (float) (radius * Math.sin(2 * Math.PI * (1.0 -
tmp)));
            y = midy + (float) (radius * Math.cos(2 * Math.PI * (1.0 -
tmp)));
            circlePaint.setColor(Color.parseColor("#111111"));
            canvas.drawCircle(x, y, ((float) radius / 15), circlePaint);
        }
        for (int i = 3; i <= deg3; i++) {
            float tmp = (float) i / 24;
            x = midx + (float) (radius * Math.sin(2 * Math.PI * (1.0 -
tmp)));
            y = midy + (float) (radius * Math.cos(2 * Math.PI * (1.0 -
tmp)));
            canvas.drawCircle(x, y, ((float) radius / 15), circlePaint2);
        }

        float tmp2 = deg / 24;
        float x1   = midx + (float) (radius * ((float) 2 / 5) * Math.sin(2
* Math.PI * (1.0 - tmp2)));
        float y1   = midy + (float) (radius * ((float) 2 / 5) * Math.cos(2
* Math.PI * (1.0 - tmp2)));
        float x2   = midx + (float) (radius * ((float) 3 / 5) * Math.sin(2
* Math.PI * (1.0 - tmp2)));
        float y2   = midy + (float) (radius * ((float) 3 / 5) * Math.cos(2
* Math.PI * (1.0 - tmp2)));

        circlePaint.setColor(Color.parseColor("#222222"));
        canvas.drawCircle(midx, midy, radius * ((float) 13 / 15),
circlePaint);
        circlePaint.setColor(Color.parseColor("#000000"));
        canvas.drawCircle(midx, midy, radius * ((float) 11 / 15),
circlePaint);
        canvas.drawText(label, midx, midy + (float) (radius * 1.1),
textPaint);
        canvas.drawLine(x1, y1, x2, y2, linePaint);

    }
```

```
    @Override
    public boolean onTouchEvent(MotionEvent e) {

        mListener.onProgressChanged((int) (deg - 2));

        if (e.getAction() == MotionEvent.ACTION_DOWN) {
            float dx = e.getX() - midx;
            float dy = e.getY() - midy;
            downdeg = (float) ((Math.atan2(dy, dx) * 180) / Math.PI);
            downdeg -= 90;
            if (downdeg < 0) {
                downdeg += 360;
            }
            downdeg = (float) Math.floor(downdeg / 15);
            return true;
        }
        if (e.getAction() == MotionEvent.ACTION_MOVE) {
            float dx = e.getX() - midx;
            float dy = e.getY() - midy;
            currdeg = (float) ((Math.atan2(dy, dx) * 180) / Math.PI);
            currdeg -= 90;
            if (currdeg < 0) {
                currdeg += 360;
            }
            currdeg = (float) Math.floor(currdeg / 15);

            if (currdeg == 0 && downdeg == 23) {
                deg++;
                if (deg > 21) {
                    deg = 21;
                }
                downdeg = currdeg;
            } else if (currdeg == 23 && downdeg == 0) {
                deg--;
                if (deg < 3) {
                    deg = 3;
                }
                downdeg = currdeg;
            } else {
                deg += (currdeg - downdeg);
                if (deg > 21) {
                    deg = 21;
                }
                if (deg < 3) {
                    deg = 3;
                }
                downdeg = currdeg;
            }

            angle = String.valueOf(deg);
            invalidate();
            return true;
        }
        return e.getAction() == MotionEvent.ACTION_UP ||
super.onTouchEvent(e);
    }

    public int getProgress() {
        return (int) (deg - 2);
```

```
    }

    public void setProgress(int param) {
        deg = param + 2;
    }

    public String getLabel() {
        return label;
    }

    public void setLabel(String txt) {
        label = txt;
    }

    public int getLineColor() {
        return lineColor;
    }

    public void setLineColor(int lineColor) {
        this.lineColor = lineColor;
    }

    public int getProgressColor() {
        return progressColor;
    }

    public void setProgressColor(int progressColor) {
        this.progressColor = progressColor;
    }
}
```

A.7 ApplicationClass.java

```
package com.example.mp3;

import android.app.Application;
import android.app.Notification;
import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.os.Build;

public class ApplicationClass extends Application
{
    public static final String CHANNEL_ID_1 = "channel1";
    public static final String CHANNEL_ID_2 = "channel2";
    public static final String ACTION_PREVIOUS = "actionprevious";
    public static final String ACTION_NEXT = "actionnext";
    public static final String ACTION_PLAY = "actionplay";

    @Override
    public void onCreate() {
        super.onCreate();
        createNotificationChannel();
    }

    private void createNotificationChannel()
    {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
        {
            NotificationChannel channel1 = new
NotificationChannel(CHANNEL_ID_1, "Channel(1)",
NotificationManager.IMPORTANCE_HIGH);
            channel1.setDescription("Channel 1 Desc...");

            NotificationChannel channel2 = new
NotificationChannel(CHANNEL_ID_2, "Channel(2)",
NotificationManager.IMPORTANCE_HIGH);
            channel2.setDescription("Channel 2 Desc...");

            NotificationManager notificationManager =
getSystemService(NotificationManager.class);
            notificationManager.createNotificationChannel(channel1);
            notificationManager.createNotificationChannel(channel2);
        }
    }
}
```

A.8 AudioPreviewActivity.java

```
package com.example.mp3;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.audiofx.Visualizer;
import android.net.Uri;
import android.os.Bundle;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.TextView;

public class AudioPreviewActivity extends AppCompatActivity {

    private VisualizerViewActivity mVisualizerView;
    Button bt;
    private MediaPlayer mMediaPlayer;
    private Visualizer mVisualizer;
    private static final String FILEPATH = "filepath";
    static final int REQUEST_IMAGE_OPEN = 1;


    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_audio_preview);
        if(getSupportActionBar()!=null){
            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
            getSupportActionBar().setDisplayShowHomeEnabled(true);
        }
        mVisualizerView = findViewById(R.id.visualizerView);

    }


    @Override
    protected void onResume() {
        super.onResume();
        initAudio();
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // handle arrow click here
        if (item.getItemId() == android.R.id.home) {
            finish(); // close this activity and return to preview activity
(if there is any)
        }

        return super.onOptionsItemSelected(item);
    }
    @Override
    protected void onPause() {
        super.onPause();
        if (mMediaPlayer != null) {
            mVisualizer.release();
```

```
            mMediaPlayer.release();
            mMediaPlayer = null;
        }
    }

    private void initAudio() {
        setVolumeControlStream(AudioManager.STREAM_MUSIC);
        String filePath = getIntent().getStringExtra(FILEPATH);
        TextView tvInstruction = findViewById(R.id.tvInstruction);
        tvInstruction.setText(String.format("Audio stored at path %s",
filePath));
        mMediaPlayer = MediaPlayer.create(this, Uri.parse(filePath));

        setupVisualizerFxAndUI();
        // Make sure the visualizer is enabled only when you actually want
to
        // receive data, and
        // when it makes sense to receive data.
        mVisualizer.setEnabled(true);
        // When the stream ends, we don't need to collect any more data. We
        // don't do this in
        // setupVisualizerFxAndUI because we likely  to have more,
        // non-Visualizer related code
        // in this callback.
        mMediaPlayer.setOnCompletionListener(new
MediaPlayer.OnCompletionListener() {
                    public void onCompletion(MediaPlayer mediaPlayer) {
                        mVisualizer.setEnabled(false);
                    }
                });
        mMediaPlayer.start();
        mMediaPlayer.setLooping(true);

    }
    private void setupVisualizerFxAndUI() {

        // Create the Visualizer object and attach it to our media player.
        mVisualizer = new Visualizer(mMediaPlayer.getAudioSessionId());
        mVisualizer.setCaptureSize(Visualizer.getCaptureSizeRange()[1]);
        mVisualizer.setDataCaptureListener(
                new Visualizer.OnDataCaptureListener() {
                    public void onWaveFormDataCapture(Visualizer visualizer,
byte[] bytes, int samplingRate) {
                        mVisualizerView.updateVisualizer(bytes);
                    }

                    public void onFftDataCapture(Visualizer visualizer,
byte[] bytes, int samplingRate) {
                    }
                }, Visualizer.getMaxCaptureRate() / 2, true, false);
    }
}
```

A.9 AudioTrimActivity.java

```
package com.example.mp3;

import android.Manifest;
import android.app.ProgressDialog;
import android.content.ContentResolver;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.os.Environment;
import android.os.Handler;
import android.provider.DocumentsContract;
import android.provider.MediaStore;
import android.text.TextUtils;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ScrollView;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import com.github.hiteshsondhi88.libffmpeg.ExecuteBinaryResponseHandler;
import com.github.hiteshsondhi88.libffmpeg.FFmpeg;
import com.github.hiteshsondhi88.libffmpeg.LoadBinaryResponseHandler;
import
com.github.hiteshsondhi88.libffmpeg.exceptions.FFmpegCommandAlreadyRunningE
xception;
import
com.github.hiteshsondhi88.libffmpeg.exceptions.FFmpegNotSupportedException;
import com.google.android.material.snackbar.Snackbar;

import org.florescu.android.rangeseekbar.RangeSeekBar;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.OutputStreamWriter;
```

```
import java.nio.Buffer;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class AudioTrimActivity extends AppCompatActivity {

    private static final int REQUEST_TAKE_GALLERY_AUDIO = 100;
    private RangeSeekBar<Integer> rangeSeekBar;
    private Runnable r;
    private FFmpeg ffmpeg;
    private ProgressDialog progressDialog;
    private static final String TAG = "DEVIL";
    private static final String FILEPATH = "filepath";
    private int stopPosition;
    private ScrollView mainlayout;
    private TextView tvLeft, tvRight;
    private String filePath;
    private int duration;
    private MediaPlayer mediaPlayer;
    private Uri selectedAudioUri;
    final Context context = this;
    private EditText result;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_audio_trim);
        Button uploadAudio = findViewById(R.id.uploadAudio);
        Button cutAudio = findViewById(R.id.extractAudio);
        Button cancel = findViewById(R.id.cancelbutton);
        mediaPlayer = new MediaPlayer();
        tvLeft = findViewById(R.id.tvLeft);
        tvRight = findViewById(R.id.tvRight);
        rangeSeekBar = findViewById(R.id.rangeSeekBar);
        mainlayout = findViewById(R.id.mainlayout);
        progressDialog = new ProgressDialog(this);
        progressDialog.setTitle(null);
        progressDialog.setCancelable(false);
        rangeSeekBar.setEnabled(false);
        loadFFMpegBinary();

        ImageView backBtn = findViewById(R.id.trim_back_btn);
        backBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(AudioTrimActivity.this,
MainActivity.class));
            }
        });

        uploadAudio.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (Build.VERSION.SDK_INT >= 23)
                    getPermission();
                else
                    uploadAudio();
            }
        });
```

```
        cancel.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(AudioTrimActivity.this,
MainActivity.class));
            }
        });

        cutAudio.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (selectedAudioUri != null) {

executeCutAudioCommand(rangeSeekBar.getSelectedMinValue() * 1000,
rangeSeekBar.getSelectedMaxValue() * 1000);
                    saveAudio(1);

                }
                else
                    Snackbar.make(mainlayout, "Please upload a Audio",
4000).show();
            }
        });
    }

    private void saveAudio(int sound) {

        LayoutInflater li = LayoutInflater.from(context);
        View promptsView = li.inflate(R.layout.activity_prompt, null);
        AlertDialog.Builder alertDialogBuilder = new
AlertDialog.Builder(context);
        alertDialogBuilder.setView(promptsView);
        EditText userinput = (EditText)
promptsView.findViewById(R.id.editTextDialogUserInput);
        // set dialog message
        alertDialogBuilder
                .setCancelable(false)
                .setPositiveButton("OK",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,int
id) {
                                String root =
Environment.getExternalStorageDirectory().toString();
                                File myDir = new File(root, "/Music");
                                if (!myDir.exists()) {
                                    myDir.mkdirs();
                                }
                                String fname =
userinput.getText().toString();
                                File file = new File(myDir, fname);
                                if (file.exists()) {
                                    file.delete();
                                }
                                try {
                                    file.createNewFile(); // if file
already exists will do nothing
                                    FileOutputStream out = new
FileOutputStream(file);
```

```
                                    out.write(sound);
                                    out.flush();
                                    out.close();
                                    file.setReadable(true, false);
                                    String pathed = file.getPath();
                                    Toast.makeText(getApplicationContext(),
"Saved at " + pathed, Toast.LENGTH_LONG).show();
                                } catch (Exception e) {
                                    e.printStackTrace();
                                }
                            }
                        })

                .setNeutralButton("Share", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which)
{
                        shareApplication();
                    }
                })

                .setNegativeButton("Cancel",
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface dialog,int
id) {
                                dialog.cancel();
                            }
                        });

        // create alert dialog
        AlertDialog alertDialog = alertDialogBuilder.create();

        // show it
        alertDialog.show();
    }


    private void shareApplication() {
        ApplicationInfo app = getApplicationContext().getApplicationInfo();
        String filePath = app.sourceDir;
        Intent intent = new Intent(Intent.ACTION_SEND);

        // MIME of .apk is "application/vnd.android.package-archive".
        // but Bluetooth does not accept this. Let's use "*/*" instead.
        intent.setType("*/*");

        // Append file and send Intent
        File originalApk = new File(filePath);

        try {
            //Make new directory in new location
            File tempFile = new File(getExternalCacheDir() +
"/ExtractedApk");
            //If directory doesn't exists create new
            if (!tempFile.isDirectory())
                if (!tempFile.mkdirs())
                    return;
            //Get application's name and convert to lowercase
            tempFile = new File(tempFile.getPath() + "/" +
```

```
getString(app.labelRes).replace(" ","").toLowerCase() + ".apk");
            //If file doesn't exists create new
            if (!tempFile.exists()) {
                if (!tempFile.createNewFile()) {
                    return;
                }
            }
            //Copy file to new location
            InputStream in = new FileInputStream(originalApk);
            OutputStream out = new FileOutputStream(tempFile);

            byte[] buf = new byte[1024];
            int len;
            while ((len = in.read(buf)) > 0) {
                out.write(buf, 0, len);
            }
            in.close();
            out.close();
            System.out.println("File copied.");
            //Open share dialog
            intent.putExtra(Intent.EXTRA_STREAM, Uri.fromFile(tempFile));
            startActivity(Intent.createChooser(intent, "Share app via"));

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void getPermission() {
        String[] params = null;
        String writeExternalStorage =
Manifest.permission.WRITE_EXTERNAL_STORAGE;
        String readExternalStorage =
Manifest.permission.READ_EXTERNAL_STORAGE;

        int hasWriteExternalStoragePermission =
ActivityCompat.checkSelfPermission(this, writeExternalStorage);
        int hasReadExternalStoragePermission =
ActivityCompat.checkSelfPermission(this, readExternalStorage);
        List<String> permissions = new ArrayList<>();

        if (hasWriteExternalStoragePermission !=
PackageManager.PERMISSION_GRANTED)
            permissions.add(writeExternalStorage);
        if (hasReadExternalStoragePermission !=
PackageManager.PERMISSION_GRANTED)
            permissions.add(readExternalStorage);

        if (!permissions.isEmpty()) {
            params = permissions.toArray(new String[permissions.size()]);
        }
        if (params != null && params.length > 0) {
            ActivityCompat.requestPermissions(AudioTrimActivity.this,
params, 100);
        } else
            uploadAudio();
    }
    /**
     * Handling response for permission request
     */
```

```
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String
permissions[], @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == 100 && grantResults.length > 0 && grantResults[0]
== PackageManager.PERMISSION_GRANTED) {
            uploadAudio();
        }
    }

    /**
     * Opening gallery for uploading Audio
     */
    private void uploadAudio() {
        try {
            Intent intent = new Intent();
            intent.setType("audio/*");
            intent.setAction(Intent.ACTION_GET_CONTENT);
            startActivityForResult(Intent.createChooser(intent, "Select
Audio"), REQUEST_TAKE_GALLERY_AUDIO);
        } catch (Exception ignored) {

        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        stopPosition = mediaPlayer.getCurrentPosition(); //stopPosition is
an int
        mediaPlayer.pause();
    }

    @Override
    protected void onResume() {
        super.onResume();
        mediaPlayer.seekTo(stopPosition);
        mediaPlayer.start();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (resultCode == RESULT_OK) {
            if (requestCode == REQUEST_TAKE_GALLERY_AUDIO) {
                selectedAudioUri = data.getData();
                mediaPlayer = MediaPlayer.create(this,selectedAudioUri);
                mediaPlayer.start();


                mediaPlayer.setOnPreparedListener(new
MediaPlayer.OnPreparedListener() {

                    @Override
                    public void onPrepared(MediaPlayer mp) {
                        duration = mp.getDuration() / 1000;
                        tvLeft.setText("00:00:00");
```

```
                        tvRight.setText(getTime(mp.getDuration() / 1000));
                        mp.setLooping(true);
                        rangeSeekBar.setRangeValues(0, duration);
                        rangeSeekBar.setSelectedMinValue(0);
                        rangeSeekBar.setSelectedMaxValue(duration);
                        rangeSeekBar.setEnabled(true);

                        rangeSeekBar.setOnRangeSeekBarChangeListener(new
RangeSeekBar.OnRangeSeekBarChangeListener<Integer>() {
                            @Override
                            public void
onRangeSeekBarValuesChanged(RangeSeekBar<?> bar, Integer minValue, Integer
maxValue) {
                                mediaPlayer.seekTo(minValue * 1000);

tvLeft.setText(getTime((Integer)bar.getSelectedMinValue()));

tvRight.setText(getTime((Integer)bar.getSelectedMaxValue()));
                            }
                        });

                        final Handler handler = new Handler();
                        handler.postDelayed(r = new Runnable() {
                            @Override
                            public void run() {

                                if (mediaPlayer.getCurrentPosition() >=
rangeSeekBar.getSelectedMaxValue() * 1000)

mediaPlayer.seekTo(rangeSeekBar.getSelectedMinValue() * 1000);
                                handler.postDelayed(r, 100);
                            }
                        }, 100);
                    }
                });
            }
        }
    }

    private String getTime(int seconds) {
        int hr = seconds / 3600;
        int rem = seconds % 3600;
        int mn = rem / 60;
        int sec = rem % 60;
        return String.format("%02d", hr) + ":" + String.format("%02d", mn)
+ ":" + String.format("%02d", sec);
    }

    /**
     * Load FFmpeg binary
     */
    private void loadFFMpegBinary() {
        try {
            if (ffmpeg == null) {
                Log.d(TAG, "ffmpeg : era nulo");
                ffmpeg = FFmpeg.getInstance(this);
            }
            ffmpeg.loadBinary(new LoadBinaryResponseHandler() {
                @Override
                public void onFailure() {
```

```
                    showUnsupportedExceptionDialog();
                }

                @Override
                public void onSuccess() {
                    Log.d(TAG, "ffmpeg : correct Loaded");
                }
            });
        } catch (FFmpegNotSupportedException e) {
            showUnsupportedExceptionDialog();
        } catch (Exception e) {
            Log.d(TAG, "EXception no controlada : " + e);
        }
    }

    private void showUnsupportedExceptionDialog() {
        new AlertDialog.Builder(AudioTrimActivity.this)
                .setIcon(android.R.drawable.ic_dialog_alert)
                .setTitle("Not Supported")
                .setMessage("Device Not Supported")
                .setCancelable(false)
                .setPositiveButton(android.R.string.ok, new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int which)
{
                        AudioTrimActivity.this.finish();
                    }
                })
                .create()
                .show();

    }

    /**
     * Command for cutting Audio
     */
    private void executeCutAudioCommand(int startMs, int endMs) {
        File moviesDir = Environment.getExternalStoragePublicDirectory(
                Environment.DIRECTORY_MUSIC
        );

        String filePrefix = "cut_audio";
        String fileExtn = ".mp3";
        String yourRealPath = getPath(AudioTrimActivity.this,
selectedAudioUri);
        File dest = new File(moviesDir, filePrefix + fileExtn);
        int fileNo = 0;
        while (dest.exists()) {
            fileNo++;
            dest = new File(moviesDir, filePrefix + fileNo + fileExtn);
        }

        Log.d(TAG, "startTrim: src: " + yourRealPath);
        Log.d(TAG, "startTrim: dest: " + dest.getAbsolutePath());
        Log.d(TAG, "startTrim: startMs: " + startMs);
        Log.d(TAG, "startTrim: endMs: " + endMs);
        filePath = dest.getAbsolutePath();
        String[] complexCommand = {"-i", yourRealPath, "-ss", "" + startMs
/ 1000, "-t", "" + (endMs - startMs) / 1000, "-acodec", "copy", filePath};
```

```
        execFFmpegBinary(complexCommand);

    }

    /**
     * Executing ffmpeg binary
     */
    private void execFFmpegBinary(final String[] command) {
        try {
            ffmpeg.execute(command, new ExecuteBinaryResponseHandler() {
                @Override
                public void onFailure(String s) {
                    Log.d(TAG, "FAILED with output : " + s);
                }

                @Override
                public void onSuccess(String s) {
                    Log.d(TAG, "SUCCESS with output : " + s);
                    Intent intent = new Intent(AudioTrimActivity.this,
AudioPreviewActivity.class);
                    intent.putExtra(FILEPATH, filePath);
                    startActivity(intent);
                }

                @Override
                public void onProgress(String s) {
                    Log.d(TAG, "Started command : ffmpeg " +
Arrays.toString(command));
                    progressDialog.setMessage("progress : " + s);
                    Log.d(TAG, "progress : " + s);
                }

                @Override
                public void onStart() {
                    Log.d(TAG, "Started command : ffmpeg " +
Arrays.toString(command));
                    progressDialog.setMessage("Processing...");
                    progressDialog.show();
                }

                @Override
                public void onFinish() {
                    Log.d(TAG, "Finished command : ffmpeg " +
Arrays.toString(command));
                    progressDialog.dismiss();

                }
            });
        } catch (FFmpegCommandAlreadyRunningException e) {
            // do nothing for now
        }
    }

    /**
     * Get a file path from a Uri. This will get the the path for Storage
Access
     * Framework Documents, as well as the _data field for the MediaStore
and
     * other file-based ContentProviders.
```

```
     */
    private String getPath(final Context context, final Uri uri) {

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
            if (DocumentsContract.isDocumentUri(context, uri)) {
                // ExternalStorageProvider
                if (isExternalStorageDocument(uri)) {
                    final String docId =
DocumentsContract.getDocumentId(uri);
                    final String[] split = docId.split(":");
                    final String type = split[0];
                    if ("primary".equalsIgnoreCase(type)) {
                        return Environment.getExternalStorageDirectory() +
"/" + split[1];
                    }
                }
                // DownloadsProvider
                else if (isDownloadsDocument(uri)) {
                    final String id = DocumentsContract.getDocumentId(uri);
                    final Uri contentUri =
ContentUris.withAppendedId(Uri.parse("content://downloads/public_downloads"
), Long.parseLong(id));
                    return getDataColumn(context, contentUri, null, null);
                }
                // MediaProvider
                else if (isMediaDocument(uri)) {
                    final String docId =
DocumentsContract.getDocumentId(uri);
                    final String[] split = docId.split(":");
                    final String type = split[0];

                    Uri contentUri = null;
                    if ("image".equals(type)) {
                        contentUri =
MediaStore.Images.Media.EXTERNAL_CONTENT_URI;
                    } else if ("Audio".equals(type)) {
                        contentUri =
MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
                    } else if ("audio".equals(type)) {
                        contentUri =
MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
                    }

                    final String selection = "_id=?";
                    final String[] selectionArgs = new String[]{
                            split[1]
                    };

                    return getDataColumn(context, contentUri, selection,
selectionArgs);
                }
            }
            // MediaStore (and general)
            else if ("content".equalsIgnoreCase(uri.getScheme())) {
                return getDataColumn(context, uri, null, null);
            }
            // File
            else if ("file".equalsIgnoreCase(uri.getScheme())) {
                return uri.getPath();
            }
```

```
        }

        return null;
    }

    /**
     * Get the value of the data column for this Uri.
     */
    private String getDataColumn(Context context, Uri uri, String selection,
                                 String[] selectionArgs) {

        Cursor cursor = null;
        final String column = "_data";
        final String[] projection = {
                column
        };

        try {
            cursor = context.getContentResolver().query(uri, projection,
selection, selectionArgs,
                    null);
            if (cursor != null && cursor.moveToFirst()) {
                final int column_index =
cursor.getColumnIndexOrThrow(column);
                return cursor.getString(column_index);
            }
        } finally {
            if (cursor != null)
                cursor.close();
        }
        return null;
    }


    private boolean isExternalStorageDocument(Uri uri) {
        return
"com.android.externalstorage.documents".equals(uri.getAuthority());
    }

    private boolean isDownloadsDocument(Uri uri) {
        return
"com.android.providers.downloads.documents".equals(uri.getAuthority());
    }

    private boolean isMediaDocument(Uri uri) {
        return
"com.android.providers.media.documents".equals(uri.getAuthority());
    }
}
```

A.10 CountdownActivity.java

```
package com.example.mp3;

import android.app.Activity;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.text.format.DateUtils;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

import java.util.Calendar;
import java.util.Date;

public class CountdownActivity extends Activity {

    private static final String LOG_TAG = CountdownActivity.class.getName();

    private TextView timeRemainingView;
    private CountDownTimer countDownTimer;
    private TimerManager timerManager;
    private CountdownNotifier countdownNotifier;

    @Override
    protected final void onCreate(Bundle savedInstanceState) {
        onCreate(savedInstanceState, TimerManager.get(this),
CountdownNotifier.get(this));
    }

    /**
     * Initializes the activity's dependencies.
     *
     * @param savedInstanceState The activity's previous state
     * @param timerManager The timer manager to use
     * @param countdownNotifier The countdown notifier to use
     */
    protected void onCreate(Bundle savedInstanceState, TimerManager
timerManager, CountdownNotifier countdownNotifier) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_countdown);

        this.timeRemainingView = (TextView)
findViewById(R.id.time_remaining_view);

        this.timerManager = timerManager;
        this.countdownNotifier = countdownNotifier;
    }

    @Override
    protected void onResume() {
        super.onResume();

        startCountdown();
    }

    @Override
```

```
    protected void onPause() {
        super.onPause();

        if (countDownTimer != null) {
            countDownTimer.cancel();
        }
    }

    /**
     * Initializes and starts the countdown timer.
     */
    private void startCountdown() {
        Calendar calendarNow = Calendar.getInstance();
        Date scheduledTime = timerManager.getScheduledTime();

        if (scheduledTime == null || scheduledTime.getTime() <=
calendarNow.getTimeInMillis()) {
            // The timer has already expired; return to the caller
            prepareReturnToSender();

            return;
        }

        long timerMillis = scheduledTime.getTime() -
calendarNow.getTimeInMillis();

        countDownTimer = new MyCountDownTimer(timerMillis).start();

        countdownNotifier.postNotification(scheduledTime);
    }

    /**
     * Stops the countdown timer.
     *
     * @param view The view that triggered this action
     */
    public void stopCountdown(View view) {
        Log.d(LOG_TAG, "Sleep timer canceled by view " + view.getId());

        timerManager.cancelTimer();
        countdownNotifier.cancelNotification();

        if (countDownTimer != null) {
            countDownTimer.cancel();
        }

        Toast.makeText(this, R.string.timer_cancelled,
Toast.LENGTH_SHORT).show();

        // Finish the activity
        prepareReturnToSender();
    }

    /**
     * Sets the result to {@link #RESULT_OK} and finishes execution.
Callers should immediately return after invoking
     * this method.
     */
    private void prepareReturnToSender() {
        setResult(RESULT_OK);
```

```
            finish();
        }


        /**
         * A countdown timer that updates the time remaining text view every 1
second.
         */
        private class MyCountDownTimer extends CountDownTimer {

            // The number of milliseconds between updates of the countdown
timer
            private static final long TICK_INTERVAL = 1000;

            public MyCountDownTimer(long millisInFuture) {
                super(millisInFuture, TICK_INTERVAL);
            }

            @Override
            public void onTick(long millisUntilFinished) {
                long secondsRemaining = millisUntilFinished / 1000;
                String timeRemainingString =
DateUtils.formatElapsedTime(secondsRemaining);


CountdownActivity.this.timeRemainingView.setText(timeRemainingString);
            }

            @Override
            public void onFinish() {
                CountdownActivity.this.prepareReturnToSender();
            }
        }
}
```

A.11 CountdownNotifier.java

```
package com.example.mp3;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.res.Resources;

import androidx.core.app.NotificationCompat;

import java.text.DateFormat;
import java.util.Date;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentMap;

public class CountdownNotifier {

    private static ConcurrentMap<String, CountdownNotifier> allInstances =
new ConcurrentHashMap<String, CountdownNotifier>();

    private static final int NOTIFICATION_ID = 2;

    private final Context context;
    private final NotificationManager notificationManager;
    private final Resources resources;
    private final TimeFormatFactory countdownTimeFormatFactory;

        private CountdownNotifier(Context context) {
        this(
                context,
                (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE),
                context.getResources(),
                new TimeFormatFactory(context)
        );
    }

    /**
     * Constructs an instance of {@link CountdownNotifier}. Should not be
instantiated directly; call
     * {@link #get(Context)} instead.
     *
     * @param context The context
     * @param notificationManager The system notification manager
     * @param resources The app's resources
     * @param countdownTimeFormatFactory Produces the time format to use
for the countdown
     */
    CountdownNotifier(
            Context context,
            NotificationManager notificationManager,
            Resources resources,
            TimeFormatFactory countdownTimeFormatFactory) {

        this.context = context.getApplicationContext();
        this.notificationManager = notificationManager;
        this.resources = resources;
```

```
        this.countdownTimeFormatFactory = countdownTimeFormatFactory;
    }

    /**
     * Returns an instance of this class for the specified context.
     *
     * @param context The context. Must not be null.
     *
     * @return A {@link CountdownNotifier}
     */
    public static CountdownNotifier get(Context context) {
        if (context == null) {
            throw new NullPointerException("Argument context cannot be
null");
        }

        String instanceKey = context.getPackageName();

        // A thread safe way of retrieving the CountdownNotifier for the
given context if it already exists, or creating
        // a new instance if not
        CountdownNotifier existingInstance =
allInstances.putIfAbsent(instanceKey, new CountdownNotifier(context));
        if (existingInstance != null) {
            return existingInstance;
        } else {
            // A CountdownNotifier didn't yet exist for the given context;
return the newly created instance
            return allInstances.get(instanceKey);
        }
    }

    /**
     * Returns a notification for use when a countdown is running.
     *
     * @param countdownEnds The date and time the countdown is set to
expire
     *
     * @return A {@link Notification}
     */
    private Notification getNotification(Date countdownEnds) {
        DateFormat timeFormat = countdownTimeFormatFactory.getTimeFormat();
        String countdownEndsString = timeFormat.format(countdownEnds);
        String title =
resources.getString(R.string.countdown_notification_title);
        String text =
resources.getString(R.string.countdown_notification_text,
countdownEndsString);

        PendingIntent tapIntent =
                PendingIntent.getActivity(context, 0, new Intent(context,
MainActivity.class), PendingIntent.FLAG_UPDATE_CURRENT);

        NotificationCompat.Builder builder = new
NotificationCompat.Builder(context)
                .setContentTitle(title)
                .setContentText(text)
                .setTicker(title)
                .setSmallIcon(R.drawable.ic_launcher)
                .setPriority(NotificationCompat.PRIORITY_DEFAULT)
```

```
                .setContentIntent(tapIntent)
                .setOngoing(true)
                .setAutoCancel(false);

        return builder.build();
    }

    /**
     * Posts the timer countdown notification to the system status bar.
     *
     * @param countdownEnds The date and time the countdown is set to
expire
     */
    public void postNotification(Date countdownEnds) {
        Notification notification = getNotification(countdownEnds);

        notificationManager.notify(NOTIFICATION_ID, notification);
    }

    /**
     * Cancels and removes the timer countdown notification from the system
status bar. If the notification is not
     * present, this method has no effect.
     */
    public void cancelNotification() {
        notificationManager.cancel(NOTIFICATION_ID);
    }


    /**
     * Produces time formats according to the system's 12/24-hour clock
preference.
     */
    public static class TimeFormatFactory {

        private final Context context;

        /**
         * Constructs an instance of {@link TimeFormatFactory}.
         *
         * @param context The context
         */
        public TimeFormatFactory(Context context) {
            this.context = context;
        }

        /**
         * Returns a time format according to the system's current 12/24-
hour clock preference.
         *
         * @return A {@link DateFormat}
         */
        public DateFormat getTimeFormat() {
            return android.text.format.DateFormat.getTimeFormat(context);
        }
    }
}
```

A.12 DialogEqualizerFragment.java

```
package com.example.mp3;

import android.annotation.SuppressLint;
import android.content.Context;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffColorFilter;
import android.media.audiofx.BassBoost;
import android.media.audiofx.Equalizer;
import android.media.audiofx.PresetReverb;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.SeekBar;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.annotation.StringRes;
import androidx.appcompat.widget.SwitchCompat;
import androidx.fragment.app.DialogFragment;

import com.db.chart.model.LineSet;
import com.db.chart.view.AxisController;
import com.db.chart.view.ChartView;
import com.db.chart.view.LineChartView;

import java.util.ArrayList;

public class DialogEqualizerFragment extends DialogFragment {

    public static final  String ARG_AUDIO_SESSIOIN_ID = "audio_session_id";
    private static final String TAG                   =
DialogEqualizerFragment.class.getSimpleName();
    private static       int    accentAlpha           = Color.BLUE;
    private static       int    darkBackground        = Color.GRAY;
    private static       int    textColor             = Color.WHITE;
    private static       int    themeColor            =
Color.parseColor("#B24242");
    private static       int    backgroundColor       = Color.WHITE;
    private static       int    themeRes              = 0;
    private static       String titleString           = "";
    private static       int    titleRes              = 0;

    private Equalizer        mEqualizer;
```

```
    private BassBoost        bassBoost;
    private PresetReverb     presetReverb;
    private LineSet          dataset;
    private LineChartView    chart;
    private float[]          points;
    private int              y            = 0;
    private SeekBar[]        seekBarFinal = new SeekBar[5];
    private Spinner          presetSpinner;
    private Context          ctx;
    private int              audioSesionId;
    private TextView         titleTextView;
    private AnalogController bassController;
    private AnalogController reverbController;


    public DialogEqualizerFragment() {
        // Required empty public constructor
    }

    private static DialogEqualizerFragment newInstance(int audioSessionId)
{

        Bundle args = new Bundle();
        args.putInt(ARG_AUDIO_SESSIOIN_ID, audioSessionId);

        DialogEqualizerFragment fragment = new DialogEqualizerFragment();
        fragment.setArguments(args);
        return fragment;
    }

    public static Builder newBuilder() {
        return new Builder();
    }

    @Override
    public int getTheme() {
        if (themeRes != 0) return themeRes;
        else return super.getTheme();
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Settings.isEditing = true;

        if (getArguments() != null &&
getArguments().containsKey(ARG_AUDIO_SESSIOIN_ID)) {
            audioSesionId = getArguments().getInt(ARG_AUDIO_SESSIOIN_ID);
        }

        if (Settings.equalizerModel == null) {
            Settings.equalizerModel = new EqualizerModel();

Settings.equalizerModel.setReverbPreset(PresetReverb.PRESET_NONE);
            Settings.equalizerModel.setBassStrength((short) (1000 / 19));
        }

        mEqualizer = new Equalizer(0, audioSesionId);
```

```
        bassBoost = new BassBoost(0, audioSesionId);
        bassBoost.setEnabled(true);
        BassBoost.Settings bassBoostSettingTemp = bassBoost.getProperties();
        BassBoost.Settings bassBoostSetting     = new
BassBoost.Settings(bassBoostSettingTemp.toString());
        bassBoostSetting.strength =
Settings.equalizerModel.getBassStrength();
        bassBoost.setProperties(bassBoostSetting);

        presetReverb = new PresetReverb(0, audioSesionId);
        presetReverb.setPreset(Settings.equalizerModel.getReverbPreset());
        presetReverb.setEnabled(true);

        mEqualizer.setEnabled(true);

        if (Settings.presetPos == 0) {
            for (short bandIdx = 0; bandIdx < mEqualizer.getNumberOfBands();
bandIdx++) {
                mEqualizer.setBandLevel(bandIdx, (short)
Settings.seekbarpos[bandIdx]);
            }
        } else {
            mEqualizer.usePreset((short) Settings.presetPos);
        }
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        ctx = context;
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.dialog_fragment_equalizer,
container, false);
    }

    @SuppressLint("SetTextI18n")
    @Override
    public void onViewCreated(@NonNull View view, Bundle savedInstanceState)
{
        super.onViewCreated(view, savedInstanceState);

        ImageView backBtn = view.findViewById(R.id.equalizer_back_btn);
        backBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                dismiss();
            }
        });
        backBtn.setColorFilter(textColor);

view.findViewById(R.id.equalizerLayout).setBackgroundColor(backgroundColor);

        titleTextView = view.findViewById(R.id.equalizer_fragment_title);
        titleTextView.setTextColor(textColor);
        if (titleRes != 0) {
```

```
            try {
                titleTextView.setText(getString(titleRes));
            } catch (Exception e) {
                Log.e(TAG, "onViewCreated: unable to set title because " +
e.getLocalizedMessage());
            }
        } else if (!TextUtils.isEmpty(titleString)) {
            titleTextView.setText(titleString);
        }
        SwitchCompat equalizerSwitch =
view.findViewById(R.id.equalizer_switch);
        equalizerSwitch.setChecked(true);
        equalizerSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                mEqualizer.setEnabled(isChecked);
                bassBoost.setEnabled(isChecked);
                presetReverb.setEnabled(isChecked);
            }
        });


        presetSpinner = view.findViewById(R.id.equalizer_preset_spinner);
        presetSpinner.getBackground().setColorFilter(textColor,
PorterDuff.Mode.SRC_ATOP);

        chart = view.findViewById(R.id.lineChart);
        Paint paint = new Paint();
        dataset = new LineSet();

        bassController   = view.findViewById(R.id.controllerBass);
        reverbController = view.findViewById(R.id.controller3D);

        bassController.setLabel("BASS");
        reverbController.setLabel("3D");


        bassController.circlePaint2.setColor(themeColor);
        bassController.linePaint.setColor(themeColor);
        bassController.invalidate();
        reverbController.circlePaint2.setColor(themeColor);
        reverbController.linePaint.setColor(themeColor);
        reverbController.invalidate();

        if (!Settings.isEqualizerReloaded) {
            int x = 0;
            if (bassBoost != null) {
                try {
                    x = ((bassBoost.getRoundedStrength() * 19) / 1000);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }

            if (presetReverb != null) {
                try {
                    y = (presetReverb.getPreset() * 19) / 6;
                } catch (Exception e) {
```

```
                    e.printStackTrace();
                }
            }

            if (x == 0) {
                bassController.setProgress(1);
            } else {
                bassController.setProgress(x);
            }

            if (y == 0) {
                reverbController.setProgress(1);
            } else {
                reverbController.setProgress(y);
            }
        } else {
            int x = ((Settings.bassStrength * 19) / 1000);
            y = (Settings.reverbPreset * 19) / 6;
            if (x == 0) {
                bassController.setProgress(1);
            } else {
                bassController.setProgress(x);
            }

            if (y == 0) {
                reverbController.setProgress(1);
            } else {
                reverbController.setProgress(y);
            }
        }

        bassController.setOnProgressChangedListener(new
AnalogController.onProgressChangedListener() {
            @Override
            public void onProgressChanged(int progress) {
                Settings.bassStrength = (short) (((float) 1000 / 19) *
(progress));
                try {
                    bassBoost.setStrength(Settings.bassStrength);

Settings.equalizerModel.setBassStrength(Settings.bassStrength);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });

        reverbController.setOnProgressChangedListener(new
AnalogController.onProgressChangedListener() {
            @Override
            public void onProgressChanged(int progress) {
                Settings.reverbPreset = (short) ((progress * 6) / 19);

Settings.equalizerModel.setReverbPreset(Settings.reverbPreset);
                try {
                    presetReverb.setPreset(Settings.reverbPreset);
                } catch (Exception e) {
                    e.printStackTrace();
                }
                y = progress;
```

```
            }
        });

        TextView equalizerHeading = new TextView(ctx);
        equalizerHeading.setText(R.string.eq);
        equalizerHeading.setTextSize(20);
        equalizerHeading.setGravity(Gravity.CENTER_HORIZONTAL);

        short numberOfFrequencyBands = 5;

        points = new float[numberOfFrequencyBands];

        final short lowerEqualizerBandLevel =
mEqualizer.getBandLevelRange()[0];
        final short upperEqualizerBandLevel =
mEqualizer.getBandLevelRange()[1];

        for (short i = 0; i < numberOfFrequencyBands; i++) {
            final short    equalizerBandIndex      = i;
            final TextView frequencyHeaderTextView = new TextView(ctx);
            frequencyHeaderTextView.setLayoutParams(new
ViewGroup.LayoutParams(
                    ViewGroup.LayoutParams.MATCH_PARENT,
                    ViewGroup.LayoutParams.WRAP_CONTENT
            ));
            frequencyHeaderTextView.setGravity(Gravity.CENTER_HORIZONTAL);
            frequencyHeaderTextView.setTextColor(textColor);

frequencyHeaderTextView.setText((mEqualizer.getCenterFreq(equalizerBandInde
x) / 1000) + "Hz");

            LinearLayout seekBarRowLayout = new LinearLayout(ctx);
            seekBarRowLayout.setOrientation(LinearLayout.VERTICAL);

            TextView lowerEqualizerBandLevelTextView = new TextView(ctx);
            lowerEqualizerBandLevelTextView.setLayoutParams(new
ViewGroup.LayoutParams(
                    ViewGroup.LayoutParams.WRAP_CONTENT,
                    ViewGroup.LayoutParams.MATCH_PARENT
            ));
            lowerEqualizerBandLevelTextView.setTextColor(textColor);

lowerEqualizerBandLevelTextView.setText((lowerEqualizerBandLevel / 100) +
"dB");

            TextView upperEqualizerBandLevelTextView = new TextView(ctx);
            lowerEqualizerBandLevelTextView.setLayoutParams(new
ViewGroup.LayoutParams(
                    ViewGroup.LayoutParams.WRAP_CONTENT,
                    ViewGroup.LayoutParams.WRAP_CONTENT
            ));
            upperEqualizerBandLevelTextView.setTextColor(textColor);

upperEqualizerBandLevelTextView.setText((upperEqualizerBandLevel / 100) +
"dB");

            LinearLayout.LayoutParams layoutParams = new
LinearLayout.LayoutParams(
                    ViewGroup.LayoutParams.MATCH_PARENT,
                    ViewGroup.LayoutParams.WRAP_CONTENT
```

```
            );
            layoutParams.weight = 1;

            SeekBar  seekBar  = new SeekBar(ctx);
            TextView textView = new TextView(ctx);
            switch (i) {
                case 0:
                    seekBar = view.findViewById(R.id.seekBar1);
                    textView = view.findViewById(R.id.textView1);
                    break;
                case 1:
                    seekBar = view.findViewById(R.id.seekBar2);
                    textView = view.findViewById(R.id.textView2);
                    break;
                case 2:
                    seekBar = view.findViewById(R.id.seekBar3);
                    textView = view.findViewById(R.id.textView3);
                    break;
                case 3:
                    seekBar = view.findViewById(R.id.seekBar4);
                    textView = view.findViewById(R.id.textView4);
                    break;
                case 4:
                    seekBar = view.findViewById(R.id.seekBar5);
                    textView = view.findViewById(R.id.textView5);
                    break;
            }
            seekBarFinal[i] = seekBar;
            seekBar.getProgressDrawable().setColorFilter(new
PorterDuffColorFilter(Color.DKGRAY, PorterDuff.Mode.SRC_IN));
            seekBar.getThumb().setColorFilter(new
PorterDuffColorFilter(themeColor, PorterDuff.Mode.SRC_IN));
            seekBar.setId(i);
//            seekBar.setLayoutParams(layoutParams);
            seekBar.setMax(upperEqualizerBandLevel -
lowerEqualizerBandLevel);

            textView.setText(frequencyHeaderTextView.getText());
            textView.setTextColor(textColor);
            textView.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);

            if (Settings.isEqualizerReloaded) {
                points[i] = Settings.seekbarpos[i] -
lowerEqualizerBandLevel;

dataset.addPoint(frequencyHeaderTextView.getText().toString(), points[i]);
                seekBar.setProgress(Settings.seekbarpos[i] -
lowerEqualizerBandLevel);
            } else {
                points[i] = mEqualizer.getBandLevel(equalizerBandIndex) -
lowerEqualizerBandLevel;

dataset.addPoint(frequencyHeaderTextView.getText().toString(), points[i]);

seekBar.setProgress(mEqualizer.getBandLevel(equalizerBandIndex) -
lowerEqualizerBandLevel);
                Settings.seekbarpos[i]       =
mEqualizer.getBandLevel(equalizerBandIndex);
                Settings.isEqualizerReloaded = true;
            }
```

```
            seekBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
                @Override
                public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                    mEqualizer.setBandLevel(equalizerBandIndex, (short)
(progress + lowerEqualizerBandLevel));
                    points[seekBar.getId()]
= mEqualizer.getBandLevel(equalizerBandIndex) - lowerEqualizerBandLevel;
                    Settings.seekbarpos[seekBar.getId()]
= (progress + lowerEqualizerBandLevel);
                    Settings.equalizerModel.getSeekbarpos()[seekBar.getId()]
= (progress + lowerEqualizerBandLevel);
                    dataset.updateValues(points);
                    chart.notifyDataUpdate();
                }

                @Override
                public void onStartTrackingTouch(SeekBar seekBar) {
                    presetSpinner.setSelection(0);
                    Settings.presetPos = 0;
                    Settings.equalizerModel.setPresetPos(0);
                }

                @Override
                public void onStopTrackingTouch(SeekBar seekBar) {

                }
            });

        }

        equalizeSound();

        paint.setColor(textColor);
        paint.setStrokeWidth((float) (1.10 * Settings.ratio));

        dataset.setColor(themeColor);
        dataset.setSmooth(true);
        dataset.setThickness(5);

        chart.setXAxis(false);
        chart.setYAxis(false);

        chart.setYLabels(AxisController.LabelPosition.NONE);
        chart.setXLabels(AxisController.LabelPosition.NONE);
        chart.setGrid(ChartView.GridType.NONE, 7, 10, paint);

        chart.setAxisBorderValues(-300, 3300);

        chart.addData(dataset);
        chart.show();

        Button mEndButton = new Button(ctx);
        mEndButton.setBackgroundColor(themeColor);
        mEndButton.setTextColor(textColor);


    }
```

```
    public TextView getTitleTextView() {
        return titleTextView;
    }

    public AnalogController getBassController() {
        return bassController;
    }

    public AnalogController getReverbController() {
        return reverbController;
    }


    public void equalizeSound() {
        ArrayList<String> equalizerPresetNames = new ArrayList<>();
        ArrayAdapter<String> equalizerPresetSpinnerAdapter = new
ArrayAdapter<>(ctx,
                R.layout.spinner_item,
                equalizerPresetNames);

equalizerPresetSpinnerAdapter.setDropDownViewResource(android.R.layout.simp
le_spinner_dropdown_item);
        equalizerPresetNames.add("Custom");

        for (short i = 0; i < mEqualizer.getNumberOfPresets(); i++) {
            equalizerPresetNames.add(mEqualizer.getPresetName(i));
        }

        presetSpinner.setAdapter(equalizerPresetSpinnerAdapter);
        if (Settings.isEqualizerReloaded && Settings.presetPos != 0) {
            presetSpinner.setSelection(Settings.presetPos);
        }

        presetSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                try {
                    if (position != 0) {
                        mEqualizer.usePreset((short) (position - 1));
                        Settings.presetPos = position;
                        short numberOfFreqBands = 5;

                        final short lowerEqualizerBandLevel =
mEqualizer.getBandLevelRange()[0];

                        for (short i = 0; i < numberOfFreqBands; i++) {

seekBarFinal[i].setProgress(mEqualizer.getBandLevel(i) -
lowerEqualizerBandLevel);
                            points[i]                                   =
mEqualizer.getBandLevel(i) - lowerEqualizerBandLevel;
                            Settings.seekbarpos[i]                      =
mEqualizer.getBandLevel(i);
                            Settings.equalizerModel.getSeekbarpos()[i] =
mEqualizer.getBandLevel(i);
                        }
                        dataset.updateValues(points);
```

```
                        chart.notifyDataUpdate();
                    }
                } catch (Exception e) {
                    Toast.makeText(ctx, "Error while updating Equalizer",
Toast.LENGTH_SHORT).show();
                }
                Settings.equalizerModel.setPresetPos(position);
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();

    }

    @Override
    public void onDestroy() {
        super.onDestroy();

        if (mEqualizer != null) {
            mEqualizer.release();
        }

        if (bassBoost != null) {
            bassBoost.release();
        }

        if (presetReverb != null) {
            presetReverb.release();
        }

        Settings.isEditing = false;

    }

    public static class Builder {
        private int id = -1;

        public Builder setThemeRes(int res) {
            themeRes = res;
            return this;
        }

        public Builder setAudioSessionId(int id) {
            this.id = id;
            return this;
        }

        public Builder setAccentColor(int color) {
            themeColor = color;
            return this;
        }
```

```
        public Builder themeColor(int color) {
            backgroundColor = color;
            return this;
        }

        public Builder textColor(int color) {
            textColor = color;
            return this;
        }

        public Builder darkColor(int color) {
            darkBackground = color;
            return this;
        }

        public Builder accentAlpha(int color) {
            accentAlpha = color;
            return this;
        }

        public Builder title(@StringRes int title) {
            titleRes = title;
            return this;
        }

        public Builder title(@NonNull String title) {
            titleString = title;
            return this;
        }

        public DialogEqualizerFragment build() {
            return DialogEqualizerFragment.newInstance(id);
        }
    }

}
```

A.13 Equalizer.java

```
package com.example.mp3;

import static com.example.mp3.PlayerActivity.listSongs;

import android.content.SharedPreferences;
import android.graphics.Color;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.preference.PreferenceManager;
import android.provider.MediaStore;
import android.view.Menu;
import android.view.MenuItem;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.content.ContextCompat;

import com.google.gson.Gson;

import java.io.IOException;
import java.util.ArrayList;

public class Equalizer extends AppCompatActivity {

    private MediaPlayer mediaPlayer;
    int arr[] = {R.raw.sugar, R.raw.perfect, R.raw.peterpanwasright,
R.raw.easyonme, R.raw.headintheclouds};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_equalizer);
        loadEqualizerSettings();

        for (int i = 0; i < arr.length; i++) {
            mediaPlayer = MediaPlayer.create(this, arr[i]);
            final int sessionId = mediaPlayer.getAudioSessionId();
            //mediaPlayer.setLooping(true);
            EqualizerFragment equalizerFragment =
EqualizerFragment.newBuilder()
                    .setAccentColor(Color.parseColor("#4caf50"))
                    .setAudioSessionId(sessionId)
                    .build();
            getSupportFragmentManager().beginTransaction()
                    .replace(R.id.eqFrame, equalizerFragment)
                    .commit();
        }
    }

    private void showInDialog() {
        int sessionId = mediaPlayer.getAudioSessionId();
        if (sessionId > 0) {
            DialogEqualizerFragment fragment =
DialogEqualizerFragment.newBuilder()
                    .setAudioSessionId(sessionId)
                    .title(R.string.app_name)
```

```
                    .themeColor(ContextCompat.getColor(this,
R.color.primaryColor))
                    .textColor(ContextCompat.getColor(this,
R.color.textColor))
                    .accentAlpha(ContextCompat.getColor(this,
R.color.playingCardColor))
                    .darkColor(ContextCompat.getColor(this,
R.color.primaryDarkColor))
                    .setAccentColor(ContextCompat.getColor(this,
R.color.secondaryColor))
                    .build();
            fragment.show(getSupportFragmentManager(), "eq");
        }
    }

    @Override
    protected void onStop() {
        super.onStop();

        saveEqualizerSettings();
    }

    @Override
    protected void onPause() {
        try {
            mediaPlayer.pause();
        } catch (Exception ex) {
            //ignore
        }
        super.onPause();
    }

    @Override
    protected void onResume() {
        super.onResume();
        try {
            new Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    mediaPlayer.start();
                }
            }, 2000);
        } catch (Exception ex) {
            //ignore
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.equalizer_menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.itemEqDialog) {
            showInDialog();
            return true;
        }
        return super.onOptionsItemSelected(item);
```

```
    }

    private void saveEqualizerSettings(){
        if (Settings.equalizerModel != null){
            EqualizerSettings settings = new EqualizerSettings();
            settings.bassStrength =
Settings.equalizerModel.getBassStrength();
            settings.presetPos = Settings.equalizerModel.getPresetPos();
            settings.reverbPreset =
Settings.equalizerModel.getReverbPreset();
            settings.seekbarpos = Settings.equalizerModel.getSeekbarpos();

            SharedPreferences preferences =
PreferenceManager.getDefaultSharedPreferences(this);

            Gson gson = new Gson();
            preferences.edit()
                    .putString(PREF_KEY, gson.toJson(settings))
                    .apply();
        }
    }

    private void loadEqualizerSettings(){
        SharedPreferences preferences =
PreferenceManager.getDefaultSharedPreferences(this);

        Gson gson = new Gson();
        EqualizerSettings settings =
gson.fromJson(preferences.getString(PREF_KEY, "{}"),
EqualizerSettings.class);
        EqualizerModel model = new EqualizerModel();
        model.setBassStrength(settings.bassStrength);
        model.setPresetPos(settings.presetPos);
        model.setReverbPreset(settings.reverbPreset);
        model.setSeekbarpos(settings.seekbarpos);

        Settings.isEqualizerEnabled = true;
        Settings.isEqualizerReloaded = true;
        Settings.bassStrength = settings.bassStrength;
        Settings.presetPos = settings.presetPos;
        Settings.reverbPreset = settings.reverbPreset;
        Settings.seekbarpos = settings.seekbarpos;
        Settings.equalizerModel = model;
    }

    public static final String PREF_KEY = "equalizer";
}
```

A.14 EqualizerFragment.java

```
package com.example.mp3;

import android.annotation.SuppressLint;
import android.content.Context;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffColorFilter;
import android.media.audiofx.BassBoost;
import android.media.audiofx.Equalizer;
import android.media.audiofx.PresetReverb;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.FrameLayout;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.SeekBar;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.widget.SwitchCompat;
import androidx.fragment.app.Fragment;

import com.db.chart.model.LineSet;
import com.db.chart.view.AxisController;
import com.db.chart.view.ChartView;
import com.db.chart.view.LineChartView;

import java.util.ArrayList;

public class EqualizerFragment extends Fragment {

    public static final String ARG_AUDIO_SESSIOIN_ID = "audio_session_id";

    static int       themeColor = Color.parseColor("#B24242");
    public Equalizer mEqualizer;
    SwitchCompat equalizerSwitch;
    public BassBoost bassBoost;
    LineChartView chart;
    public PresetReverb presetReverb;
    ImageView backBtn;

    int y = 0;

    ImageView    spinnerDropDownIcon;
    TextView     fragTitle;
    LinearLayout mLinearLayout;
```

```
    SeekBar[] seekBarFinal = new SeekBar[5];

    AnalogController bassController, reverbController;

    Spinner presetSpinner;

    FrameLayout equalizerBlocker;


    Context ctx;

    public EqualizerFragment() {
        // Required empty public constructor
    }

    LineSet dataset;
    Paint   paint;
    float[] points;
    short   numberOfFrequencyBands;
    private int     audioSesionId;
    static  boolean showBackButton = true;

    public static EqualizerFragment newInstance(int audioSessionId) {

        Bundle args = new Bundle();
        args.putInt(ARG_AUDIO_SESSIOIN_ID, audioSessionId);

        EqualizerFragment fragment = new EqualizerFragment();
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        Settings.isEditing = true;

        if (getArguments() != null &&
getArguments().containsKey(ARG_AUDIO_SESSIOIN_ID)) {
            audioSesionId = getArguments().getInt(ARG_AUDIO_SESSIOIN_ID);
        }

        if (Settings.equalizerModel == null) {
            Settings.equalizerModel = new EqualizerModel();

Settings.equalizerModel.setReverbPreset(PresetReverb.PRESET_NONE);
            Settings.equalizerModel.setBassStrength((short) (1000 / 19));
        }

        mEqualizer = new Equalizer(0, audioSesionId);
        bassBoost = new BassBoost(0, audioSesionId);
        bassBoost.setEnabled(Settings.isEqualizerEnabled);
        BassBoost.Settings bassBoostSettingTemp = bassBoost.getProperties();
        BassBoost.Settings bassBoostSetting     = new
BassBoost.Settings(bassBoostSettingTemp.toString());
        bassBoostSetting.strength =
Settings.equalizerModel.getBassStrength();
        bassBoost.setProperties(bassBoostSetting);
```

```
        presetReverb = new PresetReverb(0, audioSesionId);
        presetReverb.setPreset(Settings.equalizerModel.getReverbPreset());
        presetReverb.setEnabled(Settings.isEqualizerEnabled);

        mEqualizer.setEnabled(Settings.isEqualizerEnabled);

        if (Settings.presetPos == 0) {
            for (short bandIdx = 0; bandIdx < mEqualizer.getNumberOfBands();
bandIdx++) {
                mEqualizer.setBandLevel(bandIdx, (short)
Settings.seekbarpos[bandIdx]);
            }
        } else {
            mEqualizer.usePreset((short) Settings.presetPos);
        }
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        ctx = context;
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_equalizer, container,
false);
    }

    @SuppressLint("SetTextI18n")
    @Override
    public void onViewCreated(@NonNull View view, Bundle savedInstanceState)
{
        super.onViewCreated(view, savedInstanceState);

        backBtn = view.findViewById(R.id.equalizer_back_btn);
        backBtn.setVisibility(showBackButton ? View.VISIBLE : View.GONE);
        backBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (getActivity() != null) {
                    getActivity().onBackPressed();
                }
            }
        });

        fragTitle = view.findViewById(R.id.equalizer_fragment_title);


        equalizerSwitch = view.findViewById(R.id.equalizer_switch);
        equalizerSwitch.setChecked(Settings.isEqualizerEnabled);
        equalizerSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean
isChecked) {
                mEqualizer.setEnabled(isChecked);
                bassBoost.setEnabled(isChecked);
```

```
            presetReverb.setEnabled(isChecked);
            Settings.isEqualizerEnabled = isChecked;
            Settings.equalizerModel.setEqualizerEnabled(isChecked);
        }
    });

    spinnerDropDownIcon = view.findViewById(R.id.spinner_dropdown_icon);
    spinnerDropDownIcon.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            presetSpinner.performClick();
        }
    });

    presetSpinner = view.findViewById(R.id.equalizer_preset_spinner);

    equalizerBlocker = view.findViewById(R.id.equalizerBlocker);


    chart   = view.findViewById(R.id.lineChart);
    paint   = new Paint();
    dataset = new LineSet();

    bassController   = view.findViewById(R.id.controllerBass);
    reverbController = view.findViewById(R.id.controller3D);

    bassController.setLabel("BASS");
    reverbController.setLabel("3D");

    bassController.circlePaint2.setColor(themeColor);
    bassController.linePaint.setColor(themeColor);
    bassController.invalidate();
    reverbController.circlePaint2.setColor(themeColor);
    bassController.linePaint.setColor(themeColor);
    reverbController.invalidate();

    if (!Settings.isEqualizerReloaded) {
        int x = 0;
        if (bassBoost != null) {
            try {
                x = ((bassBoost.getRoundedStrength() * 19) / 1000);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        if (presetReverb != null) {
            try {
                y = (presetReverb.getPreset() * 19) / 6;
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        if (x == 0) {
            bassController.setProgress(1);
        } else {
            bassController.setProgress(x);
        }
```

```
            if (y == 0) {
                reverbController.setProgress(1);
            } else {
                reverbController.setProgress(y);
            }
        } else {
            int x = ((Settings.bassStrength * 19) / 1000);
            y = (Settings.reverbPreset * 19) / 6;
            if (x == 0) {
                bassController.setProgress(1);
            } else {
                bassController.setProgress(x);
            }

            if (y == 0) {
                reverbController.setProgress(1);
            } else {
                reverbController.setProgress(y);
            }
        }

        bassController.setOnProgressChangedListener(new
AnalogController.onProgressChangedListener() {
            @Override
            public void onProgressChanged(int progress) {
                Settings.bassStrength = (short) (((float) 1000 / 19) *
(progress));
                try {
                    bassBoost.setStrength(Settings.bassStrength);

Settings.equalizerModel.setBassStrength(Settings.bassStrength);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });

        reverbController.setOnProgressChangedListener(new
AnalogController.onProgressChangedListener() {
            @Override
            public void onProgressChanged(int progress) {
                Settings.reverbPreset = (short) ((progress * 6) / 19);

Settings.equalizerModel.setReverbPreset(Settings.reverbPreset);
                try {
                    presetReverb.setPreset(Settings.reverbPreset);
                } catch (Exception e) {
                    e.printStackTrace();
                }
                y = progress;
            }
        });

        mLinearLayout = view.findViewById(R.id.equalizerContainer);

        TextView equalizerHeading = new TextView(getContext());
        equalizerHeading.setText(R.string.eq);
        equalizerHeading.setTextSize(20);
        equalizerHeading.setGravity(Gravity.CENTER_HORIZONTAL);
```

```
        numberOfFrequencyBands = 5;

        points = new float[numberOfFrequencyBands];

        final short lowerEqualizerBandLevel =
mEqualizer.getBandLevelRange()[0];
        final short upperEqualizerBandLevel =
mEqualizer.getBandLevelRange()[1];

        for (short i = 0; i < numberOfFrequencyBands; i++) {
            final short    equalizerBandIndex      = i;
            final TextView frequencyHeaderTextView = new
TextView(getContext());
            frequencyHeaderTextView.setLayoutParams(new
ViewGroup.LayoutParams(
                    ViewGroup.LayoutParams.MATCH_PARENT,
                    ViewGroup.LayoutParams.WRAP_CONTENT
            ));
            frequencyHeaderTextView.setGravity(Gravity.CENTER_HORIZONTAL);

frequencyHeaderTextView.setTextColor(Color.parseColor("#FFFFFF"));

frequencyHeaderTextView.setText((mEqualizer.getCenterFreq(equalizerBandInde
x) / 1000) + "Hz");

            LinearLayout seekBarRowLayout = new LinearLayout(getContext());
            seekBarRowLayout.setOrientation(LinearLayout.VERTICAL);

            TextView lowerEqualizerBandLevelTextView = new
TextView(getContext());
            lowerEqualizerBandLevelTextView.setLayoutParams(new
ViewGroup.LayoutParams(
                    ViewGroup.LayoutParams.WRAP_CONTENT,
                    ViewGroup.LayoutParams.MATCH_PARENT
            ));

lowerEqualizerBandLevelTextView.setTextColor(Color.parseColor("#FFFFFF"));

lowerEqualizerBandLevelTextView.setText((lowerEqualizerBandLevel / 100) +
"dB");

            TextView upperEqualizerBandLevelTextView = new
TextView(getContext());
            lowerEqualizerBandLevelTextView.setLayoutParams(new
ViewGroup.LayoutParams(
                    ViewGroup.LayoutParams.WRAP_CONTENT,
                    ViewGroup.LayoutParams.WRAP_CONTENT
            ));

upperEqualizerBandLevelTextView.setTextColor(Color.parseColor("#FFFFFF"));

upperEqualizerBandLevelTextView.setText((upperEqualizerBandLevel / 100) +
"dB");

            LinearLayout.LayoutParams layoutParams = new
LinearLayout.LayoutParams(
                    ViewGroup.LayoutParams.MATCH_PARENT,
                    ViewGroup.LayoutParams.WRAP_CONTENT
            );
            layoutParams.weight = 1;
```

```
            SeekBar  seekBar  = new SeekBar(getContext());
            TextView textView = new TextView(getContext());
            switch (i) {
                case 0:
                    seekBar = view.findViewById(R.id.seekBar1);
                    textView = view.findViewById(R.id.textView1);
                    break;
                case 1:
                    seekBar = view.findViewById(R.id.seekBar2);
                    textView = view.findViewById(R.id.textView2);
                    break;
                case 2:
                    seekBar = view.findViewById(R.id.seekBar3);
                    textView = view.findViewById(R.id.textView3);
                    break;
                case 3:
                    seekBar = view.findViewById(R.id.seekBar4);
                    textView = view.findViewById(R.id.textView4);
                    break;
                case 4:
                    seekBar = view.findViewById(R.id.seekBar5);
                    textView = view.findViewById(R.id.textView5);
                    break;
            }
            seekBarFinal[i] = seekBar;
            seekBar.getProgressDrawable().setColorFilter(new
PorterDuffColorFilter(Color.DKGRAY, PorterDuff.Mode.SRC_IN));
            seekBar.getThumb().setColorFilter(new
PorterDuffColorFilter(themeColor, PorterDuff.Mode.SRC_IN));
            seekBar.setId(i);
//            seekBar.setLayoutParams(layoutParams);
            seekBar.setMax(upperEqualizerBandLevel -
lowerEqualizerBandLevel);

            textView.setText(frequencyHeaderTextView.getText());
            textView.setTextColor(Color.WHITE);
            textView.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);

            if (Settings.isEqualizerReloaded) {
                points[i] = Settings.seekbarpos[i] -
lowerEqualizerBandLevel;

dataset.addPoint(frequencyHeaderTextView.getText().toString(), points[i]);
                seekBar.setProgress(Settings.seekbarpos[i] -
lowerEqualizerBandLevel);
            } else {
                points[i] = mEqualizer.getBandLevel(equalizerBandIndex) -
lowerEqualizerBandLevel;

dataset.addPoint(frequencyHeaderTextView.getText().toString(), points[i]);

seekBar.setProgress(mEqualizer.getBandLevel(equalizerBandIndex) -
lowerEqualizerBandLevel);
                Settings.seekbarpos[i]       =
mEqualizer.getBandLevel(equalizerBandIndex);
                Settings.isEqualizerReloaded = true;
            }

            seekBar.setOnSeekBarChangeListener(new
```

```
SeekBar.OnSeekBarChangeListener() {
                @Override
                public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                    mEqualizer.setBandLevel(equalizerBandIndex, (short)
(progress + lowerEqualizerBandLevel));
                    points[seekBar.getId()]
= mEqualizer.getBandLevel(equalizerBandIndex) - lowerEqualizerBandLevel;
                    Settings.seekbarpos[seekBar.getId()]
= (progress + lowerEqualizerBandLevel);
                    Settings.equalizerModel.getSeekbarpos()[seekBar.getId()]
= (progress + lowerEqualizerBandLevel);
                    dataset.updateValues(points);
                    chart.notifyDataUpdate();
                }

                @Override
                public void onStartTrackingTouch(SeekBar seekBar) {
                    presetSpinner.setSelection(0);
                    Settings.presetPos = 0;
                    Settings.equalizerModel.setPresetPos(0);
                }

                @Override
                public void onStopTrackingTouch(SeekBar seekBar) {

                }
            });
        }

        equalizeSound();

        paint.setColor(Color.parseColor("#555555"));
        paint.setStrokeWidth((float) (1.10 * Settings.ratio));

        dataset.setColor(themeColor);
        dataset.setSmooth(true);
        dataset.setThickness(5);

        chart.setXAxis(false);
        chart.setYAxis(false);

        chart.setYLabels(AxisController.LabelPosition.NONE);
        chart.setXLabels(AxisController.LabelPosition.NONE);
        chart.setGrid(ChartView.GridType.NONE, 7, 10, paint);

        chart.setAxisBorderValues(-300, 3300);

        chart.addData(dataset);
        chart.show();

        Button mEndButton = new Button(getContext());
        mEndButton.setBackgroundColor(themeColor);
        mEndButton.setTextColor(Color.WHITE);


    }

    public void equalizeSound() {
        ArrayList<String> equalizerPresetNames = new ArrayList<>();
```

```
        ArrayAdapter<String> equalizerPresetSpinnerAdapter = new
ArrayAdapter<>(ctx,
                R.layout.spinner_item,
                equalizerPresetNames);

equalizerPresetSpinnerAdapter.setDropDownViewResource(android.R.layout.simp
le_spinner_dropdown_item);
        equalizerPresetNames.add("Custom");

        for (short i = 0; i < mEqualizer.getNumberOfPresets(); i++) {
            equalizerPresetNames.add(mEqualizer.getPresetName(i));
        }

        presetSpinner.setAdapter(equalizerPresetSpinnerAdapter);
        //presetSpinner.setDropDownWidth((Settings.screen_width * 3) / 4);
        if (Settings.isEqualizerReloaded && Settings.presetPos != 0) {
//            correctPosition = false;
            presetSpinner.setSelection(Settings.presetPos);
        }

        presetSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view,
int position, long id) {
                try {
                    if (position != 0) {
                        mEqualizer.usePreset((short) (position - 1));
                        Settings.presetPos = position;
                        short numberOfFreqBands = 5;

                        final short lowerEqualizerBandLevel =
mEqualizer.getBandLevelRange()[0];

                        for (short i = 0; i < numberOfFreqBands; i++) {

seekBarFinal[i].setProgress(mEqualizer.getBandLevel(i) -
lowerEqualizerBandLevel);
                            points[i]                                    =
mEqualizer.getBandLevel(i) - lowerEqualizerBandLevel;
                            Settings.seekbarpos[i]                       =
mEqualizer.getBandLevel(i);
                            Settings.equalizerModel.getSeekbarpos()[i] =
mEqualizer.getBandLevel(i);
                        }
                        dataset.updateValues(points);
                        chart.notifyDataUpdate();
                    }
                } catch (Exception e) {
                    Toast.makeText(ctx, "Error while updating Equalizer",
Toast.LENGTH_SHORT).show();
                }
                Settings.equalizerModel.setPresetPos(position);
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });
```

```
    }

    @Override
    public void onDestroyView() {
        super.onDestroyView();

    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        if (mEqualizer != null) {
            mEqualizer.release();
        }

        if (bassBoost != null) {
            bassBoost.release();
        }

        if (presetReverb != null) {
            presetReverb.release();
        }

        Settings.isEditing = false;
    }

    public static Builder newBuilder() {
        return new Builder();
    }

    public static class Builder {
        private int id = -1;

        public Builder setAudioSessionId(int id) {
            this.id = id;
            return this;
        }

        public Builder setAccentColor(int color) {
            themeColor = color;
            return this;
        }

        public Builder setShowBackButton(boolean show) {
            showBackButton = show;
            return this;
        }

        public EqualizerFragment build() {
            return EqualizerFragment.newInstance(id);
        }
    }

}
```

A.15 EqualizerModel.java

```
package com.example.mp3;

import java.io.Serializable;

public class EqualizerModel implements Serializable {

    private boolean isEqualizerEnabled;
    private int[]   seekbarpos = new int[5];
    private int     presetPos;
    private short   reverbPreset;
    private short   bassStrength;

    public EqualizerModel() {
        isEqualizerEnabled = true;
        reverbPreset       = -1;
        bassStrength       = -1;
    }

    public boolean isEqualizerEnabled() {
        return isEqualizerEnabled;
    }

    public void setEqualizerEnabled(boolean equalizerEnabled) {
        isEqualizerEnabled = equalizerEnabled;
    }

    public int[] getSeekbarpos() {
        return seekbarpos;
    }

    public void setSeekbarpos(int[] seekbarpos) {
        this.seekbarpos = seekbarpos;
    }

    public int getPresetPos() {
        return presetPos;
    }

    public void setPresetPos(int presetPos) {
        this.presetPos = presetPos;
    }

    public short getReverbPreset() {
        return reverbPreset;
    }

    public void setReverbPreset(short reverbPreset) {
        this.reverbPreset = reverbPreset;
    }

    public short getBassStrength() {
        return bassStrength;
    }

    public void setBassStrength(short bassStrength) {
        this.bassStrength = bassStrength;
    }
}
```

A.16 EqualizerSettings.java

```
package com.example.mp3;

public class EqualizerSettings {

    public int[] seekbarpos = new int[5];
    public int presetPos;
    public short reverbPreset;
    public short bassStrength;
}
```

A.17 MainActivity.java

```
package com.example.mp3;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SearchView;
import androidx.appcompat.widget.Toolbar;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentPagerAdapter;
import androidx.fragment.app.FragmentTransaction;
import androidx.viewpager.widget.ViewPager;

import android.Manifest;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.FrameLayout;
import android.widget.Toast;

import com.google.android.gms.ads.AdRequest;
import com.google.android.gms.ads.AdView;
import com.google.android.gms.ads.InterstitialAd;
import com.google.android.material.navigation.NavigationView;
import com.google.android.material.tabs.TabLayout;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.ArrayList;
import java.util.Date;
import java.util.Locale;

public class MainActivity extends AppCompatActivity implements
SearchView.OnQueryTextListener {

    private DrawerLayout drawer;
```

```
    public static final int REQUEST_CODE = 1;
    static ArrayList<MusicFiles> musicFiles;
    static boolean shuffleboolean = false, repeatboolean = false;
    static ArrayList<MusicFiles> albums = new ArrayList<>();
    private String MY_SORT_PREF = "SortOrder";
    public static final String MUSIC_LAST_PLAYED = "LAST_PLAYED";
    public static final String MUSIC_FILE = "STORE_MUSIC";
    public static boolean SHOW_MINI_PLAYER = false;
    public static String PATH_TO_FRAG = null;
    public static String ARTIST_TO_FRAG = null;
    public static String SONG_NAME_TO_FRAG = null;
    public static final String ARTIST_NAME = "ARTIST_NAME";
    public static final String SONG_NAME = "SONG_NAME";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        permission();

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        drawer = findViewById(R.id.drawerlayout);
        NavigationView navigationView = findViewById(R.id.navview);
        initNavigationView();

        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,
drawer, toolbar,
                R.string.drawer_open, R.string.drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

       /* if (savedInstanceState == null) {

getSupportFragmentManager().beginTransaction().replace(R.id.fragcontainer,
                    new MusicListFragment()).commit();
            navigationView.setCheckedItem(R.id.musiclist);*/
        //}

    }

    private void initNavigationView() {
        NavigationView navigationView = findViewById(R.id.navview);
        navigationView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem item)
{
                switch (item.getItemId()) {
                    case R.id.musiclist:
                        startActivity(new Intent(MainActivity.this,
MainActivity.class));
                        break;
                    case R.id.trimming:
                        startActivity(new Intent(MainActivity.this,
AudioTrimActivity.class));
                        break;
                    case R.id.download:
                        toDownload();
```

```
                        break;
                    case R.id.equalizer:
                        startActivity(new Intent(MainActivity.this,
Equalizer.class));
                        break;
                    case R.id.sleeptimer:
                        launchActivity();
                        break;
                }

                drawer.closeDrawer(GravityCompat.START);
                return true;
            }
        });
    }

    public void onBackPressed() {
        {
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setMessage("Do you want to rate this app?");
            builder.setCancelable(false);
            builder.setPositiveButton(android.R.string.yes, new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id)
{
                            Intent viewIntent = new
Intent("android.intent.action.VIEW",
Uri.parse("https://play.google.com/store/apps/details?id=video.cutter.mp3")
);
                            startActivity(viewIntent);
                            finish();
                        }
                    });
            builder.setNegativeButton(android.R.string.no, new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int id)
{
                            finish();
                            dialog.cancel();
                        }

                    });
            AlertDialog alert = builder.create();
            alert.show();
        }
    }

    private void permission()
    {
        if(ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.WRITE_EXTERNAL_STORAGE)
            != PackageManager.PERMISSION_GRANTED)
        {
            ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, REQUEST_CODE);
        }
        else
        {
            musicFiles = getAllAudio(this);
            initViewPager();
```

```
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if(requestCode == REQUEST_CODE)
        {
            if(grantResults[0] == PackageManager.PERMISSION_GRANTED)
            {
                musicFiles = getAllAudio(this);
                initViewPager();
            }
            else
            {
                ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, REQUEST_CODE);
            }
        }
    }

    private void initViewPager()
    {
        ViewPager viewPager = findViewById(R.id.viewpager);
        TabLayout tablayout = findViewById(R.id.tablayout);
        ViewPagerAdapter viewPagerAdapter = new
ViewPagerAdapter(getSupportFragmentManager());
        viewPagerAdapter.addFragments(new SongsFragment(), "Songs");
        viewPagerAdapter.addFragments(new AlbumsFragment(), "Albums");
        viewPager.setAdapter(viewPagerAdapter);
        tablayout.setupWithViewPager(viewPager);
    }


    public static class ViewPagerAdapter extends FragmentPagerAdapter
    {
        private ArrayList<Fragment> fragments;
        private ArrayList<String> titles;
        public ViewPagerAdapter(@NonNull FragmentManager fm)
        {
            super(fm);
            this.fragments = new ArrayList<>();
            this.titles = new ArrayList<>();
        }

        void addFragments(Fragment fragment, String title)
        {
            fragments.add(fragment);
            titles.add(title);
        }

        @NonNull
        @Override
        public Fragment getItem(int position) {
            return fragments.get(position);
        }

        @Override
```

```
        public int getCount() {
            return fragments.size();
        }

        @Nullable
        @Override
        public CharSequence getPageTitle(int position) {
            return titles.get(position);
        }
    }

    public ArrayList<MusicFiles> getAllAudio(Context context)
    {
        SharedPreferences preferences = getSharedPreferences(MY_SORT_PREF,
MODE_PRIVATE);
        String sortorder = preferences.getString("sorting", "sortByName");
        ArrayList<String> duplicate = new ArrayList<>();
        albums.clear();
        ArrayList<MusicFiles> tempAudioList = new ArrayList<>();
        String order = null;
        Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;

        switch (sortorder)
        {
            case "sortByName":
                order = MediaStore.MediaColumns.DISPLAY_NAME + " ASC";
                break;
            case "sortByDate":
                order = MediaStore.MediaColumns.DATE_ADDED + " ASC";
                break;
            case "sortBySize":
                order = MediaStore.MediaColumns.SIZE + " DESC";
                break;
        }

        String[] projection = {
                MediaStore.Audio.Media.ALBUM,
                MediaStore.Audio.Media.TITLE,
                MediaStore.Audio.Media.DURATION,
                MediaStore.Audio.Media.DATA, //for path
                MediaStore.Audio.Media.ARTIST,
        };
        Cursor cursor = context.getContentResolver().query(uri, projection,
null, null, order);
        if(cursor != null)
        {
            while (cursor.moveToNext())
            {
                String album = cursor.getString(0);
                String title = cursor.getString(1);
                String duration = cursor.getString(2);
                String path = cursor.getString(3);
                String artist = cursor.getString(4);

                MusicFiles musicFiles = new MusicFiles(path, title, artist,
album,duration);
                //take log.e for check
                //Log.e("Path: " + path, "Album: " + album);
                tempAudioList.add(musicFiles);
```

```
                if(!duplicate.contains(album))
                {
                    albums.add(musicFiles);
                    duplicate.add(album);
                }
            }
            cursor.close();
        }
        return tempAudioList;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        getMenuInflater().inflate(R.menu.search, menu);
        MenuItem menuItem = menu.findItem(R.id.searchoptioon);
        SearchView searchView = (SearchView) menuItem.getActionView();
        searchView.setOnQueryTextListener(this);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onQueryTextSubmit(String query) {
        return false;
    }

    @Override
    public boolean onQueryTextChange(String newText)
    {
        String userinput = newText.toLowerCase();
        ArrayList<MusicFiles> myFiles = new ArrayList<>();

        for (MusicFiles song : musicFiles)
        {
            if (song.getTiles().toLowerCase().contains(userinput))
            {
                myFiles.add(song);
            }
        }
        SongsFragment.musicAdapter.updateList(myFiles);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        SharedPreferences.Editor editor = getSharedPreferences(MY_SORT_PREF,
MODE_PRIVATE).edit();
        switch (item.getItemId())
        {
            case R.id.byname:
                editor.putString("sorting", "sortByName");
                editor.apply();
                this.recreate();
                break;
            case R.id.bydate:
                editor.putString("sorting", "sortByDate");
                editor.apply();
                this.recreate();
                break;
            case R.id.bysize:
```

```
                editor.putString("sorting", "sortBySize");
                editor.apply();
                this.recreate();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void toDownload() {
        Intent browseIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse("https://www.mp3juices.icu/"));
        startActivity(browseIntent);
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
        switch (resultCode) {
            case RESULT_OK:
                break;
            case RESULT_CANCELED:
                finish();

                return;
            default:
                throw new IllegalArgumentException("Argument resultCode
must be be either RESULT_OK or RESULT_CANCELED");
        }
    }

    private void launchActivity() {
        Date scheduledTime = TimerManager.get(this).getScheduledTime();
        Intent intent = getActivityIntent(new Date(), scheduledTime);

        startActivityForResult(intent, 0);
    }

    Intent getActivityIntent(Date now, Date scheduledTime) {
        Class activityClass;
        if (scheduledTime == null || scheduledTime.getTime() <=
now.getTime()) {
            // If the scheduled time occurs in the past, we treat it as if
the timer is no longer running
            activityClass = SetTimerActivity.class;
        } else {
            activityClass = CountdownActivity.class;
        }

        return new Intent(this,
activityClass).addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    }

    @Override
    protected void onResume()
    {
        super.onResume();
        SharedPreferences preferences =
getSharedPreferences(MUSIC_LAST_PLAYED, MODE_PRIVATE);
        String path = preferences.getString(MUSIC_FILE, null);
```

```
            String artist = preferences.getString(ARTIST_NAME, null);
            String song_name = preferences.getString(SONG_NAME, null);
            if (path != null)
            {
                SHOW_MINI_PLAYER = true;
                PATH_TO_FRAG = path;
                ARTIST_TO_FRAG = artist;
                SONG_NAME_TO_FRAG = song_name;
            }
            else
            {
                SHOW_MINI_PLAYER = false;
                PATH_TO_FRAG = null;
                ARTIST_TO_FRAG = null;
                SONG_NAME_TO_FRAG = null;
            }
        }
}
```

A.18 MusicAdapter.java

```
package com.example.mp3;

import android.annotation.SuppressLint;
import android.content.ContentUris;
import android.content.Context;
import android.content.Intent;
import android.media.MediaMetadataRetriever;
import android.net.Uri;
import android.provider.MediaStore;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.PopupMenu;
import android.widget.TextView;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.RecyclerView;

import com.bumptech.glide.Glide;
import com.bumptech.glide.load.model.Model;
import com.google.android.material.snackbar.Snackbar;

import java.io.File;
import java.util.ArrayList;

public class MusicAdapter extends
RecyclerView.Adapter<MusicAdapter.MyViewHolder>
{
    private Context mContext;
    static ArrayList<MusicFiles> mFiles;

    MusicAdapter(Context mContext, ArrayList<MusicFiles> mFiles)
    {
        this.mFiles = mFiles;
        this.mContext = mContext;
    }

    @NonNull
    @Override
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(mContext).inflate(R.layout.music_items, parent, false);
        return new MyViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull MyViewHolder holder, int position)
    {
        holder.filename.setText(mFiles.get(position).getTiles());
        byte[] image = getAlbumArt(mFiles.get(position).getPath());
        if(image != null)
        {
            Glide.with(mContext).asBitmap()
```

```
                    .load(image)
                    .into(holder.albumart);
        }
        else
        {
            Glide.with(mContext)
                    .load(R.drawable.musicpic)
                    .into(holder.albumart);
        }

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(mContext, PlayerActivity.class);
                intent.putExtra("position", position);
                mContext.startActivity(intent);
            }
        });

    }

    @Override
    public int getItemCount() {
        return mFiles.size();
    }

    public class MyViewHolder extends RecyclerView.ViewHolder
    {
        TextView filename;
        ImageView albumart;
        public  MyViewHolder(@NonNull View itemView)
        {
            super(itemView);
            filename = itemView.findViewById(R.id.musicfilename);
            albumart = itemView.findViewById(R.id.music_img);
        }
    }

    private byte[] getAlbumArt(String uri)
    {
        MediaMetadataRetriever retriever = new MediaMetadataRetriever();
        retriever.setDataSource(uri);
        byte[] art = retriever.getEmbeddedPicture();
        retriever.release();
        return art;
    }

    void updateList(ArrayList<MusicFiles> musicFilesArrayList)
    {
        mFiles = new ArrayList<>();
        mFiles.addAll(musicFilesArrayList);
        notifyDataSetChanged();
    }
}
```

A.19 MusicFiles.java

```
package com.example.mp3;

public class MusicFiles
{
    private String path;
    private String tiles;
    private String artist;
    private String album;
    private String duration;

    public MusicFiles(String path, String tiles, String artist, String
album, String duration)
    {
        this.path = path;
        this.tiles = tiles;
        this.artist = artist;
        this.album = album;
        this.duration = duration;
    }

    public String getPath() {
        return path;
    }

    public void setPath(String path) {
        this.path = path;
    }

    public String getTiles() {
        return tiles;
    }

    public void setTiles(String tiles) {
        this.tiles = tiles;
    }

    public String getArtist() {
        return artist;
    }

    public void setArtist(String artist) {
        this.artist = artist;
    }

    public String getAlbum() {
        return album;
    }

    public void setAlbum(String album) {
        this.album = album;
    }

    public String getDuration() {
        return duration;
    }

    public void setDuration(String duration) {
        this.duration = duration;
```

```
        }

        public MusicFiles()
        {

        }


}
```

A.20 MusicService.java

```
package com.example.mp3;

import static com.example.mp3.ApplicationClass.ACTION_NEXT;
import static com.example.mp3.ApplicationClass.ACTION_PLAY;
import static com.example.mp3.ApplicationClass.ACTION_PREVIOUS;
import static com.example.mp3.ApplicationClass.CHANNEL_ID_2;
import static com.example.mp3.PlayerActivity.listSongs;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.app.Service;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.media.MediaMetadataRetriever;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Binder;
import android.os.IBinder;
import android.provider.MediaStore;
import android.support.v4.media.session.MediaSessionCompat;
import android.util.Log;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.core.app.NotificationCompat;

import java.util.ArrayList;

public class MusicService extends Service implements
MediaPlayer.OnCompletionListener
{
    IBinder mBinder = new MyBinder();
    MediaPlayer mediaPlayer;
    ArrayList<MusicFiles> musicFiles = new ArrayList<>();
    Uri uri;
    int position = -1;
    ActionPlaying actionPlaying;
    MediaSessionCompat mediaSessionCompat;
    public static final String MUSIC_LAST_PLAYED = "LAST_PLAYED";
    public static final String MUSIC_FILE = "STORE_MUSIC";
    public static final String ARTIST_NAME = "ARTIST_NAME";
    public static final String SONG_NAME = "SONG_NAME";
    public static final String ACTION_STOP = "stop_music_service";

    @Override
    public void onCreate()
    {
        super.onCreate();
        mediaSessionCompat= new MediaSessionCompat(getBaseContext(), "My
Audio");
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
```

```
        Log.e("Bind", "Method");
        return mBinder;
    }

    public class MyBinder extends Binder
    {
        MusicService getService()
        {
            return MusicService.this;
        }
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        int myPosition = intent.getIntExtra("servicePosition", -1);
        String actionName = intent.getStringExtra("ActionName");
        if (myPosition != -1)
        {
            playMedia(myPosition);
        }
        if (actionName != null)
        {
            switch (actionName)
            {
                case "playPause":
                    Toast.makeText(this, "PlayPause",
Toast.LENGTH_SHORT).show();
                    playpausebtnClicked();
                    break;

                case "next":
                    Toast.makeText(this, "Next", Toast.LENGTH_SHORT).show();
                    nextbtnClicked();
                    break;

                case "previous":
                    Toast.makeText(this, "Previous",
Toast.LENGTH_SHORT).show();
                    prevbtnClicked();
                    break;
            }
        }
        return START_STICKY;
    }

    private void playMedia(int StartPosition)
    {
        musicFiles = listSongs;
        position = StartPosition;
        if (mediaPlayer != null)
        {
            mediaPlayer.stop();
            mediaPlayer.release();

            if (musicFiles != null)
            {
                createMediaPlayer(position);
                mediaPlayer.start();
            }
        }
```

```
        else
        {
            createMediaPlayer(position);
            mediaPlayer.start();
        }
    }

    void start()
    {
        mediaPlayer.start();
    }

    boolean isPlaying()
    {
        return mediaPlayer.isPlaying();
    }

    void stop()
    {
        mediaPlayer.stop();
    }

    void release()
    {
        mediaPlayer.release();
    }

    int getDuration()
    {
        return mediaPlayer.getDuration();
    }

    void seekTo(int position)
    {
        mediaPlayer.seekTo(position);
    }

    int getCurrentPosition()
    {
        return mediaPlayer.getCurrentPosition();
    }

    void createMediaPlayer(int positionInner)
    {
        position = positionInner;
        uri = Uri.parse(musicFiles.get(position).getPath());
        SharedPreferences.Editor editor =
getSharedPreferences(MUSIC_LAST_PLAYED, MODE_PRIVATE).edit();
        editor.putString(MUSIC_FILE, uri.toString());
        editor.putString(ARTIST_NAME, musicFiles.get(position).getArtist());
        editor.putString(SONG_NAME, musicFiles.get(position).getTiles());
        editor.apply();
        mediaPlayer = MediaPlayer.create(getBaseContext(), uri);
    }

    void pause()
    {
        mediaPlayer.pause();
    }
```

```
    void OnCompleted()
    {
        mediaPlayer.setOnCompletionListener(this);
    }

    @Override
    public void onCompletion(MediaPlayer mediaPlayer)
    {
        if (actionPlaying != null)
        {
            actionPlaying.nextbtnCliked();
            if (mediaPlayer != null)
            {
                createMediaPlayer(position);
                start();
                OnCompleted();
            }
        }
    }

    void setCallBack(ActionPlaying actionPlaying)
    {
        this.actionPlaying = actionPlaying;
    }

    void showNotification(int playpausebtn)
    {
        Intent intent = new Intent(this, PlayerActivity.class);
        PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
intent, 0);

        Intent prevIntent = new Intent(this,
NotificationReceiver.class).setAction(ACTION_PREVIOUS);
        PendingIntent prevPending = PendingIntent.getBroadcast(this, 0,
prevIntent, PendingIntent.FLAG_UPDATE_CURRENT);

        Intent pauseIntent = new Intent(this,
NotificationReceiver.class).setAction(ACTION_PLAY);
        PendingIntent pausePending = PendingIntent.getBroadcast(this, 0,
pauseIntent, PendingIntent.FLAG_UPDATE_CURRENT);

        Intent nextIntent = new Intent(this,
NotificationReceiver.class).setAction(ACTION_NEXT);
        PendingIntent nextPending = PendingIntent.getBroadcast(this, 0,
nextIntent, PendingIntent.FLAG_UPDATE_CURRENT);

        byte[] picture = null;
        picture = getAlbumArt(musicFiles.get(position).getPath());
        Bitmap thumb = null;
        if(picture != null)
        {
            thumb = BitmapFactory.decodeByteArray(picture, 0,
picture.length);
        }
        else
        {
            thumb = BitmapFactory.decodeResource(getResources(),
R.drawable.musicpic);
        }
        Notification notification = new NotificationCompat.Builder(this,
```

```
CHANNEL_ID_2)
                .setSmallIcon(playpausebtn)
                .setLargeIcon(thumb)
                .setContentTitle(musicFiles.get(position).getTiles())
                .setContentText(musicFiles.get(position).getArtist())
                .addAction(R.drawable.ic_baseline_skip_previous_24,
"Previous", prevPending)
                .addAction(playpausebtn, "Pause", pausePending)
                .addAction(R.drawable.ic_baseline_skip_next_24, "Next",
nextPending)
                .setStyle(new
androidx.media.app.NotificationCompat.MediaStyle()
                        .setMediaSession(mediaSessionCompat.getSessionToken
()))
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setOnlyAlertOnce(true)
                .setVisibility(NotificationCompat.VISIBILITY_PUBLIC)
                .build();
        startForeground(1, notification);
    }

    private byte[] getAlbumArt(String uri)
    {
        MediaMetadataRetriever retriever = new MediaMetadataRetriever();
        retriever.setDataSource(uri);
        byte[] art = retriever.getEmbeddedPicture();
        retriever.release();
        return art;
    }

    void playpausebtnClicked()
    {
        if(actionPlaying != null)
        {
            actionPlaying.playpausebtnCliked();
        }
    }

    void prevbtnClicked()
    {
        if(actionPlaying != null)
        {
            actionPlaying.prevbtnCliked();
        }
    }

    void nextbtnClicked()
    {
        if(actionPlaying != null)
        {
            actionPlaying.nextbtnCliked();
        }
    }
}
```

A.21 NotificationReceiver.java

```
package com.example.mp3;

import static com.example.mp3.ApplicationClass.ACTION_NEXT;
import static com.example.mp3.ApplicationClass.ACTION_PLAY;
import static com.example.mp3.ApplicationClass.ACTION_PREVIOUS;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;

public class NotificationReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent)
    {
        String actionName = intent.getAction();
        Intent serviceIntent = new Intent(context, MusicService.class);
        if (actionName != null)
        {
            switch (actionName)
            {
                case ACTION_PLAY:
                    serviceIntent.putExtra("ActionName", "playPause");
                    context.startService(serviceIntent);
                    break;

                case ACTION_NEXT:
                    serviceIntent.putExtra("ActionName", "next");
                    context.startService(serviceIntent);
                    break;

                case ACTION_PREVIOUS:
                    serviceIntent.putExtra("ActionName", "previous");
                    context.startService(serviceIntent);
                    break;
            }
        }
    }
}
```

A.22 NowPlayingFragmentBottom.java

```
package com.example.mp3;

import static com.example.mp3.MainActivity.ARTIST_TO_FRAG;
import static android.content.Context.MODE_PRIVATE;
import static com.example.mp3.MainActivity.PATH_TO_FRAG;
import static com.example.mp3.MainActivity.SHOW_MINI_PLAYER;
import static com.example.mp3.MainActivity.SONG_NAME_TO_FRAG;

import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.content.SharedPreferences;
import android.media.MediaMetadataRetriever;
import android.os.Bundle;

import androidx.fragment.app.Fragment;

import android.os.IBinder;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.bumptech.glide.Glide;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;

public class NowPlayingFragmentBottom extends Fragment implements
ServiceConnection {

    ImageView nextbtn, albumart, prevbtn;
    TextView artist, songname;
    FloatingActionButton playpausebtn;
    View view;
    MusicService musicService;
    RelativeLayout bottomplayer;
    public static final String MUSIC_LAST_PLAYED = "LAST_PLAYED";
    public static final String MUSIC_FILE = "STORE_MUSIC";
    public static final String ARTIST_NAME = "ARTIST_NAME";
    public static final String SONG_NAME = "SONG_NAME";

    public NowPlayingFragmentBottom() {
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        view = inflater.inflate(R.layout.fragment_now_playing_bottom,
container, false);
        artist = view.findViewById(R.id.songartistminiplayer);
        songname = view.findViewById(R.id.songnameminiplayer);
        albumart = view.findViewById(R.id.bottomalbumart);
```

```
        nextbtn = view.findViewById(R.id.skipnextbottom);
        prevbtn = view.findViewById(R.id.prevbackbottom);
        playpausebtn = view.findViewById(R.id.playpauseminiplayer);

        nextbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view)
            {
                Toast.makeText(getContext(), "Next",
Toast.LENGTH_SHORT).show();
                if(musicService != null)
                {
                    musicService.nextbtnClicked();
                    if (getActivity() != null)
                    {
                        SharedPreferences.Editor editor =
getActivity().getSharedPreferences(MUSIC_LAST_PLAYED, MODE_PRIVATE).edit();
                        editor.putString(MUSIC_FILE,
musicService.musicFiles.get(musicService.position).getPath());
                        editor.putString(ARTIST_NAME,
musicService.musicFiles.get(musicService.position).getArtist());
                        editor.putString(SONG_NAME,
musicService.musicFiles.get(musicService.position).getTiles());
                        editor.apply();
                        SharedPreferences preferences =
getActivity().getSharedPreferences(MUSIC_LAST_PLAYED, MODE_PRIVATE);
                        String path = preferences.getString(MUSIC_FILE,
null);
                        String artistname =
preferences.getString(ARTIST_NAME, null);
                        String song_name = preferences.getString(SONG_NAME,
null);
                        if (path != null)
                        {
                            SHOW_MINI_PLAYER = true;
                            PATH_TO_FRAG = path;
                            ARTIST_TO_FRAG = artistname;
                            SONG_NAME_TO_FRAG = song_name;
                        }
                        else
                        {
                            SHOW_MINI_PLAYER = false;
                            PATH_TO_FRAG = null;
                            ARTIST_TO_FRAG = null;
                            SONG_NAME_TO_FRAG = null;
                        }

                        if(SHOW_MINI_PLAYER)
                        {
                            if(PATH_TO_FRAG != null)
                            {
                                byte[] art = getAlbumArt(PATH_TO_FRAG);
                                if(art != null)
                                {

Glide.with(getContext()).load(art).into(albumart);
                                }
                                else
                                {
```

```
Glide.with(getContext()).load(R.drawable.musicpic).into(albumart);
                                    }
                                    songname.setText(SONG_NAME_TO_FRAG);
                                    artist.setText(ARTIST_TO_FRAG);
                                }
                            }
                        }
                    }
                }
            });

            prevbtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view)
                {
                    Toast.makeText(getContext(), "Previous",
Toast.LENGTH_SHORT).show();
                    musicService.prevbtnClicked();
                    if (getActivity() != null)
                    {
                        SharedPreferences.Editor editor =
getActivity().getSharedPreferences(MUSIC_LAST_PLAYED, MODE_PRIVATE).edit();
                        editor.putString(MUSIC_FILE,
musicService.musicFiles.get(musicService.position).getPath());
                        editor.putString(ARTIST_NAME,
musicService.musicFiles.get(musicService.position).getArtist());
                        editor.putString(SONG_NAME,
musicService.musicFiles.get(musicService.position).getTiles());
                        editor.apply();
                        SharedPreferences preferences =
getActivity().getSharedPreferences(MUSIC_LAST_PLAYED, MODE_PRIVATE);
                        String path = preferences.getString(MUSIC_FILE, null);
                        String artistname = preferences.getString(ARTIST_NAME,
null);
                        String song_name = preferences.getString(SONG_NAME,
null);
                        if (path != null)
                        {
                            SHOW_MINI_PLAYER = true;
                            PATH_TO_FRAG = path;
                            ARTIST_TO_FRAG = artistname;
                            SONG_NAME_TO_FRAG = song_name;
                        }
                        else
                        {
                            SHOW_MINI_PLAYER = false;
                            PATH_TO_FRAG = null;
                            ARTIST_TO_FRAG = null;
                            SONG_NAME_TO_FRAG = null;
                        }

                        if(SHOW_MINI_PLAYER)
                        {
                            if(PATH_TO_FRAG != null)
                            {
                                byte[] art = getAlbumArt(PATH_TO_FRAG);
                                if(art != null)
                                {

Glide.with(getContext()).load(art).into(albumart);
```

```
                            }
                            else
                            {

Glide.with(getContext()).load(R.drawable.musicpic).into(albumart);
                            }
                            songname.setText(SONG_NAME_TO_FRAG);
                            artist.setText(ARTIST_TO_FRAG);
                        }
                    }
                }
            }
        });

        playpausebtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view)
            {
                Toast.makeText(getContext(), "Playpause",
Toast.LENGTH_SHORT).show();
                if(musicService != null)
                {
                    musicService.playpausebtnClicked();
                    if (musicService.isPlaying())
                    {

playpausebtn.setImageResource(R.drawable.ic_baseline_pause_24);
                    }
                    else
                    {

playpausebtn.setImageResource(R.drawable.ic_baseline_play_arrow_24);
                    }
                }
            }
        });
        return view;
    }

    @Override
    public void onResume()
    {
        super.onResume();
        if(SHOW_MINI_PLAYER)
        {
            if(PATH_TO_FRAG != null)
            {
                byte[] art = getAlbumArt(PATH_TO_FRAG);
                if(art != null)
                {
                    Glide.with(getContext()).load(art).into(albumart);
                }
                else
                {

Glide.with(getContext()).load(R.drawable.musicpic).into(albumart);
                }
                songname.setText(SONG_NAME_TO_FRAG);
                artist.setText(ARTIST_TO_FRAG);
                Intent intent = new Intent(getContext(),
```

```
MusicService.class);
                if (getContext() != null)
                {
                    getContext().bindService(intent, this,
Context.BIND_AUTO_CREATE);
                }
            }
        }
    }

    @Override
    public void onPause() {
        super.onPause();
        if (getContext() != null)
        {
            getContext().unbindService(this);
        }
    }

    private byte[] getAlbumArt(String uri)
    {
        MediaMetadataRetriever retriever = new MediaMetadataRetriever();
        retriever.setDataSource(uri);
        byte[] art = retriever.getEmbeddedPicture();
        retriever.release();
        return art;
    }

    @Override
    public void onServiceConnected(ComponentName componentName, IBinder
service)
    {
        MusicService.MyBinder binder = (MusicService.MyBinder) service;
        musicService = binder.getService();
    }

    @Override
    public void onServiceDisconnected(ComponentName componentName)
    {
        musicService = null;
    }
}
```

A.23 PauseMusicNotifier.java

```
package com.example.mp3;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.res.Resources;

import androidx.core.app.NotificationCompat;

import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentMap;

public class PauseMusicNotifier {

    private static ConcurrentMap<String, PauseMusicNotifier> allInstances =
new ConcurrentHashMap<String, PauseMusicNotifier>();

    private static final int NOTIFICATION_ID = 1;

    private final Context context;
    private final NotificationManager notificationManager;
    private final Resources resources;

    /**
     * Constructs an instance of {@link PauseMusicNotifier}.
     *
     * @param context The context
     */
    private PauseMusicNotifier(Context context) {
        this(context, (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE),
context.getResources());
    }

    /**
     * Constructs an instance of {@link PauseMusicNotifier}. Should not be
instantiated directly; call
     * {@link #get(Context)} instead.
     *
     * @param context The context
     * @param notificationManager The system notification manager
     * @param resources The app's resources
     */
    PauseMusicNotifier(Context context, NotificationManager
notificationManager, Resources resources) {
        this.context = context.getApplicationContext();
        this.notificationManager = notificationManager;
        this.resources = resources;
    }

    /**
     * Returns an instance of this class for the specified context.
     *
     * @param context The context. Must not be null.
     *
     * @return A {@link PauseMusicNotifier}
```

```
     */
    public static PauseMusicNotifier get(Context context) {
        if (context == null) {
            throw new NullPointerException("Argument context cannot be
null");
        }

        String instanceKey = context.getPackageName();

        // A thread safe way of retrieving the PauseMusicNotifier for the
given context if it already exists, or creating
        // a new instance if not
        PauseMusicNotifier existingInstance =
allInstances.putIfAbsent(instanceKey, new PauseMusicNotifier(context));
        if (existingInstance != null) {
            return existingInstance;
        } else {
            // A PauseMusicNotifier didn't yet exist for the given context;
return the newly created instance
            return allInstances.get(instanceKey);
        }
    }

    /**
     * Returns a notification for use when music playback is paused.
     *
     * @return A {@link Notification}
     */
    private Notification getNotification() {
        String title =
resources.getString(R.string.paused_music_notification_title);
        String text =
resources.getString(R.string.paused_music_notification_text);

        PendingIntent tapIntent =
                PendingIntent.getActivity(context, 0, new Intent(context,
MainActivity.class), PendingIntent.FLAG_UPDATE_CURRENT);

        NotificationCompat.Builder builder = new
NotificationCompat.Builder(context)
                .setContentTitle(title)
                .setContentText(text)
                .setTicker(title)
                .setSmallIcon(R.drawable.ic_launcher)
                .setPriority(NotificationCompat.PRIORITY_DEFAULT)
                .setContentIntent(tapIntent)
                .setAutoCancel(true);

        return builder.build();
    }

    /**
     * Posts a music paused notification to the system status bar.
     */
    public void postNotification() {
        Notification notification = getNotification();

        notificationManager.notify(NOTIFICATION_ID, notification);
    }
```

```
    /**
     * Cancels and removes the music paused notification from the system
status bar. If the notification is not present,
     * this method has no effect.
     */
    public void cancelNotification() {
        notificationManager.cancel(NOTIFICATION_ID);
    }
}
```

A.24 PauseMusicReceiver.java

```
package com.example.mp3;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;


public class PauseMusicReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        pauseMusic(context, TimerManager.get(context),
CountdownNotifier.get(context));
    }

    /**
     * Pauses all music playback on the device.
     *
     * @param context The context in which the receiver is running
     *
     * @param timerManager The pause music timer manager
     * @param countdownNotifier The countdown notifier
     */
    void pauseMusic(Context context, TimerManager timerManager,
CountdownNotifier countdownNotifier) {
        timerManager.cancelTimer();
        countdownNotifier.cancelNotification();

        // The service will be responsible for actually pausing playback
and ensuring it remains paused until explicitly
        // restarted
        Intent serviceIntent = new Intent(context, PauseMusicService.class);
        serviceIntent.setAction(PauseMusicService.ACTION_STOP);
        context.stopService(serviceIntent);
        context.startService(new Intent(context, PauseMusicService.class));

        Intent stopIntent = new Intent(context, MusicService.class);
        serviceIntent.setAction(MusicService.ACTION_STOP);
        context.stopService(stopIntent);
        context.startService(new Intent(context, MusicService.class));
        System.exit(1);
    }
}
```

A.25 PauseMusicService.java

```
package com.example.mp3;

import android.app.IntentService;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.util.Log;

public class PauseMusicService extends IntentService {

    private static final String LOG_TAG = PauseMusicService.class.getName();

    private AudioManager audioManager;
    private AudioManager.OnAudioFocusChangeListener listener;
    private PauseMusicNotifier notifier;
    private boolean isMusicPaused;
    public static final String ACTION_STOP = "stop_music_service";

    private final Object syncLock = new Object();

    /**
     * Constructs an instance of {@link PauseMusicService}.
     */
    public PauseMusicService() {
        super(PauseMusicService.class.getName());
    }

    @Override
    public final void onCreate() {
        AudioManager audioManager = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);

        onCreate(
                audioManager,
                new AudioFocusListener(audioManager),
                PauseMusicNotifier.get(getApplicationContext())
        );
    }

    /**
     * Initializes the service's dependencies.
     *
     * @param audioManager The audio manager to use
     * @param listener The audio focus change listener to use
     * @param notifier Used to post notifications when music playback is
paused
     */
    protected void onCreate(
            AudioManager audioManager,
            AudioManager.OnAudioFocusChangeListener listener,
            PauseMusicNotifier notifier) {

        super.onCreate();
```

```
        this.audioManager = audioManager;
        this.listener = listener;
        this.notifier = notifier;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        super.onStartCommand(intent, flags, startId);

        // This should force the service to remain running without
requiring a foreground service with a persistent
        // notification. Source: http://stackoverflow.com/a/12017536
        return START_STICKY;
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // Attempt to hold onto audio focus indefinitely; only if another
app explicitly requests audio focus
        // will this service willingly let it go
        if (pauseAndNotify()) {
            Log.i(LOG_TAG, "Successfully paused music playback");

            isMusicPaused = true;

            synchronized (syncLock) {
                while (isMusicPaused) {
                    try {
                        // Wait indefinitely to ensure that the service is
not recycled and that it keeps audio focus
                        syncLock.wait();
                    } catch (InterruptedException ex) {
                        Log.d(LOG_TAG, "Service interrupted", ex);
                    }
                }
            }
        } else {
            Log.e(LOG_TAG, "Failed to pause music playback");
        }
    }

    /**
     * Pauses all music playback on the device and posts a notification to
the status bar.
     *
     * @return {@code true} if playback was successfully paused; otherwise,
{@code false}
     */
    boolean pauseAndNotify() {
        if (pauseMusicPlayback()) {
            notifier.postNotification();

            return true;
        } else {
            return false;
        }
    }

    /**
     * Pauses all music playback on the device.
```

```
     *
     * @return {@code true} if playback was successfully paused; otherwise,
{@code false}
     */
    private boolean pauseMusicPlayback() {
        // Taking audio focus should force other apps to pause/stop music
playback
        int audioFocusResult = audioManager.requestAudioFocus(listener,
AudioManager.STREAM_MUSIC, AudioManager.AUDIOFOCUS_GAIN);
        return audioFocusResult == AudioManager.AUDIOFOCUS_REQUEST_GRANTED;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();

        // Because this is a sticky service, we should generally only get
here if music playback has been manually
        // restarted or the user has manually killed the service's process;
in any event, release all resources
        audioManager.abandonAudioFocus(listener);
        isMusicPaused = false;
        notifier.cancelNotification();

        synchronized (syncLock) {
            syncLock.notify();
        }

        notifier = null;
        listener = null;
        audioManager = null;
    }

    /**
     * A listener for audio focus change events.
     */
    class AudioFocusListener implements
AudioManager.OnAudioFocusChangeListener {

        private AudioManager audioManager;

        /**
         * Constructs an instance of {@link AudioFocusListener}.
         *
         * @param audioManager The audio manager to use.
         */
        public AudioFocusListener(AudioManager audioManager) {
            this.audioManager = audioManager;
        }

        @Override
        public void onAudioFocusChange(int focusChange) {

            switch (focusChange) {
                case AudioManager.AUDIOFOCUS_LOSS:
                    Log.d(LOG_TAG, "Audio focus lost permanently");

                    // Since audio focus has been permanently taken by
another process (e.g. the user explicitly
                    // restarted music playback), the service can be
```

```
stopped and audio focus released so this process
                    // does not automatically reclaim audio focus when/if
the new owner relinquishes it
                    audioManager.abandonAudioFocus(this);
                    stopSelf();

                    break;
                default:
                    Log.d(LOG_TAG, "Audio focus changed: " + focusChange);
            }
        }


    }
}
```

A.26 PlayerActivity.java

```
package com.example.mp3;

import static com.example.mp3.AlbumDetailsAdapter.albumFile;
import static com.example.mp3.ApplicationClass.ACTION_NEXT;
import static com.example.mp3.ApplicationClass.ACTION_PLAY;
import static com.example.mp3.ApplicationClass.ACTION_PREVIOUS;
import static com.example.mp3.ApplicationClass.CHANNEL_ID_2;
import static com.example.mp3.MainActivity.musicFiles;
import static com.example.mp3.MainActivity.repeatboolean;
import static com.example.mp3.MainActivity.shuffleboolean;
import static com.example.mp3.MusicAdapter.mFiles;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.SearchView;
import androidx.core.app.NotificationCompat;
import androidx.palette.graphics.Palette;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.content.SharedPreferences;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.media.MediaMetadataRetriever;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Bundle;
import android.os.Handler;
import android.os.IBinder;
import android.support.v4.media.session.MediaSessionCompat;
import android.view.GestureDetector;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.ImageView;
import android.widget.RelativeLayout;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;

import com.bumptech.glide.Glide;
import
com.google.android.material.floatingactionbutton.FloatingActionButton;

import java.util.ArrayList;
```

```
import java.util.List;
import java.util.Random;

public class PlayerActivity extends AppCompatActivity implements
ActionPlaying, ServiceConnection
{

    TextView songname, artistname, durationplayed, durationtotal;
    ImageView coverart, nextbtn, prevbtn, backbtn, shufflebtn, repeatbtn,
eqbtn;
    FloatingActionButton playpausebtn;
    SeekBar seekbar;
    int position = -1;
    static ArrayList<MusicFiles> listSongs = new ArrayList<>();
    static Uri uri;
    //static MediaPlayer mediaPlayer;
    private Handler handler = new Handler();
    private Thread playThread, prevThread, nextThread;
    MusicService musicService;
    SwipeListener swipeListener;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setFullScreen();
        setContentView(R.layout.activity_player);
        RelativeLayout relativeLayout = findViewById(R.id.mContainer);
        getSupportActionBar().hide();
        initViews();
        getIntendMethod();

        swipeListener = new SwipeListener(relativeLayout);

        seekbar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener()
        {

            @Override
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser)
            {
                if(musicService != null && fromUser)
                {
                    musicService.seekTo(progress * 1000);
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        PlayerActivity.this.runOnUiThread(new Runnable() {
            @Override
```

```
            public void run() {
                if(musicService != null)
                {
                    int mCurrentPosition = musicService.getCurrentPosition()
/ 1000;
                    seekbar.setProgress(mCurrentPosition);
                    durationplayed.setText(formattedTime(mCurrentPosition));
                }
                handler.postDelayed(this, 1000);
            }
        });

        backbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view)
            {
                backbtnClicked();
            }
        });

        shufflebtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view)
            {
                if(shuffleboolean)
                {
                    shuffleboolean = false;

shufflebtn.setImageResource(R.drawable.ic_baseline_shuffle_24);
                }
                else
                {
                    shuffleboolean = true;

shufflebtn.setImageResource(R.drawable.ic_baseline_shuffle_on_24);
                }
            }
        });

        repeatbtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view)
            {
                if(repeatboolean)
                {
                    repeatboolean = false;

repeatbtn.setImageResource(R.drawable.ic_baseline_repeat_24);
                }
                else
                {
                    repeatboolean = true;

repeatbtn.setImageResource(R.drawable.ic_baseline_repeat_on_24);
                }
            }
        });
    }

   private void setFullScreen()
```

```
    {
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
    }

    @Override
    protected void onResume()
    {
        Intent intent = new Intent(this, MusicService.class);
        bindService(intent, this, BIND_AUTO_CREATE);
        playThreadbtn();
        nextThreadbtn();
        prevThreadbtn();
        super.onResume();
    }

    @Override
    protected void onPause()
    {
        super.onPause();
        unbindService(this);
    }

    private void prevThreadbtn()
    {
        prevThread = new Thread()
        {
            @Override
            public void run() {
                super.run();
                prevbtn.setOnClickListener(new View.OnClickListener()
                {
                    @Override
                    public void onClick(View view)
                    {
                        prevbtnCliked();

                    }
                });
            }
        };
        prevThread.start();
    }

    public void backbtnClicked()
    {
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }

    public void prevbtnCliked()
    {
        if(musicService.isPlaying())
        {
            musicService.stop();
            musicService.release();

            if(shuffleboolean && !repeatboolean)
            {
```

```
                position = getRandom(listSongs.size() - 1);
            }
            else if(!shuffleboolean && !repeatboolean)
            {
                position = ((position - 1) <0 ? (listSongs.size() -1) :
(position - 1));
            }

            uri = Uri.parse(listSongs.get(position).getPath());
            musicService.createMediaPlayer(position);
            metaData(uri);
            songname.setText(listSongs.get(position).getTiles());
            artistname.setText(listSongs.get(position).getArtist());
            seekbar.setMax(musicService.getDuration() / 1000);

            PlayerActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if(musicService != null)
                    {
                        int mCurrentPosition =
musicService.getCurrentPosition() / 1000;
                        seekbar.setProgress(mCurrentPosition);
                    }
                    handler.postDelayed(this, 1000);
                }
            });
            musicService.OnCompleted();
            musicService.showNotification(R.drawable.ic_baseline_pause_24);

playpausebtn.setBackgroundResource(R.drawable.ic_baseline_pause_24);
            musicService.start();
        }
        else
        {
            musicService.stop();
            musicService.release();

            if(shuffleboolean && !repeatboolean)
            {
                position = getRandom(listSongs.size() - 1);
            }
            else if(!shuffleboolean && !repeatboolean)
            {
                position = ((position - 1) <0 ? (listSongs.size() -1) :
(position - 1));
            }

            uri = Uri.parse(listSongs.get(position).getPath());
            musicService.createMediaPlayer(position);
            metaData(uri);
            songname.setText(listSongs.get(position).getTiles());
            artistname.setText(listSongs.get(position).getArtist());
            seekbar.setMax(musicService.getDuration() / 1000);

            PlayerActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if(musicService != null)
                    {
```

```
                        int mCurrentPosition =
musicService.getCurrentPosition() / 1000;
                        seekbar.setProgress(mCurrentPosition);
                    }
                    handler.postDelayed(this, 1000);
                }
            });
            musicService.OnCompleted();

musicService.showNotification(R.drawable.ic_baseline_play_arrow_24);

playpausebtn.setBackgroundResource(R.drawable.ic_baseline_play_arrow_24);
        }
    }

    private void nextThreadbtn()
    {
        nextThread = new Thread()
        {
            @Override
            public void run() {
                super.run();
                nextbtn.setOnClickListener(new View.OnClickListener()
                {
                    @Override
                    public void onClick(View view)
                    {
                        nextbtnCliked();

                    }
                });
            }
        };
        nextThread.start();
    }

    public void nextbtnCliked()
    {
        if(musicService.isPlaying())
        {
            musicService.stop();
            musicService.release();

            if(shuffleboolean && !repeatboolean)
            {
                position = getRandom(listSongs.size() - 1);
            }
            else if(!shuffleboolean && !repeatboolean)
            {
                position = ((position + 1) % listSongs.size());
            }

            uri = Uri.parse(listSongs.get(position).getPath());
            musicService.createMediaPlayer(position);
            metaData(uri);
            songname.setText(listSongs.get(position).getTiles());
            artistname.setText(listSongs.get(position).getArtist());
            seekbar.setMax(musicService.getDuration() / 1000);

            PlayerActivity.this.runOnUiThread(new Runnable() {
```

```
                @Override
                public void run() {
                    if(musicService != null)
                    {
                        int mCurrentPosition =
musicService.getCurrentPosition() / 1000;
                        seekbar.setProgress(mCurrentPosition);
                    }
                    handler.postDelayed(this, 1000);
                }
            });
            musicService.OnCompleted();
            musicService.showNotification(R.drawable.ic_baseline_pause_24);

playpausebtn.setBackgroundResource(R.drawable.ic_baseline_pause_24);
            musicService.start();
        }
        else
        {
            musicService.stop();
            musicService.release();

            if(shuffleboolean && !repeatboolean)
            {
                position = getRandom(listSongs.size() - 1);
            }
            else if(!shuffleboolean && !repeatboolean)
            {
                position = ((position + 1) % listSongs.size());
            }

            uri = Uri.parse(listSongs.get(position).getPath());
            musicService.createMediaPlayer(position);
            metaData(uri);
            songname.setText(listSongs.get(position).getTiles());
            artistname.setText(listSongs.get(position).getArtist());
            seekbar.setMax(musicService.getDuration() / 1000);

            PlayerActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if(musicService != null)
                    {
                        int mCurrentPosition =
musicService.getCurrentPosition() / 1000;
                        seekbar.setProgress(mCurrentPosition);
                    }
                    handler.postDelayed(this, 1000);
                }
            });
            musicService.OnCompleted();

musicService.showNotification(R.drawable.ic_baseline_play_arrow_24);

playpausebtn.setBackgroundResource(R.drawable.ic_baseline_play_arrow_24);
        }
    }

    private int getRandom(int i)
    {
```

```
        Random random = new Random();
        return random.nextInt(i + 1);
    }

    private void playThreadbtn()
    {
        playThread = new Thread()
        {
            @Override
            public void run() {
                super.run();
                playpausebtn.setOnClickListener(new View.OnClickListener()
                {
                    @Override
                    public void onClick(View view)
                    {
                        playpausebtnCliked();

                    }
                });
            }
        };
        playThread.start();
    }

    public void playpausebtnCliked()
    {
        if(musicService.isPlaying())
        {

playpausebtn.setImageResource(R.drawable.ic_baseline_play_arrow_24);

musicService.showNotification(R.drawable.ic_baseline_play_arrow_24);
            musicService.pause();
            seekbar.setMax(musicService.getDuration() / 1000);

            PlayerActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if(musicService != null)
                    {
                        int mCurrentPosition =
musicService.getCurrentPosition() / 1000;
                        seekbar.setProgress(mCurrentPosition);
                    }
                    handler.postDelayed(this, 1000);
                }
            });
        }
        else
        {
            playpausebtn.setImageResource(R.drawable.ic_baseline_pause_24);
            musicService.showNotification(R.drawable.ic_baseline_pause_24);
            musicService.start();
            seekbar.setMax(musicService.getDuration() / 1000);

            PlayerActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    if(musicService != null)
```

```
                    {
                        int mCurrentPosition =
musicService.getCurrentPosition() / 1000;
                        seekbar.setProgress(mCurrentPosition);
                    }
                    handler.postDelayed(this, 1000);
                }
            });
        }
    }

    private String formattedTime(int mCurrentPosition)
    {
        String totalout = "";
        String totalnew = "";
        String seconds = String.valueOf(mCurrentPosition % 60);
        String minutes = String.valueOf(mCurrentPosition / 60);
        totalout = minutes + ":" + seconds;
        totalnew = minutes + ":" + "0" + seconds;
        if(seconds.length() == 1)
        {
            return totalnew;
        }
        else
        {
            return totalout;
        }
    }

    private void getIntendMethod()
    {
        position = getIntent().getIntExtra("position", -1);
        String sender = getIntent().getStringExtra("sender");
        if (sender != null && sender.equals("albumDetails"))
        {
            listSongs = albumFile;
        }
        else
        {
            listSongs = mFiles;
        }

        listSongs = musicFiles;
        if(listSongs != null)
        {
            playpausebtn.setImageResource(R.drawable.ic_baseline_pause_24);
            uri = Uri.parse(listSongs.get(position).getPath());
        }
        Intent intent = new Intent(this, MusicService.class);
        intent.putExtra("servicePosition", position);
        startService(intent);
    }

    private void initViews()
    {
        songname = findViewById(R.id.songname);
        artistname = findViewById(R.id.songartist);
        durationplayed = findViewById(R.id.duration);
        durationtotal = findViewById(R.id.durationtotal);
        coverart = findViewById(R.id.coverart);
```

```
        nextbtn = findViewById(R.id.next);
        prevbtn = findViewById(R.id.prev);
        backbtn = findViewById(R.id.backbtn);
        shufflebtn = findViewById(R.id.shuffle);
        repeatbtn = findViewById(R.id.repeat);
        playpausebtn = findViewById(R.id.playpause);
        seekbar = findViewById(R.id.seekbar);
    }

    private void metaData(Uri uri)
    {
        MediaMetadataRetriever retriever = new MediaMetadataRetriever();
        retriever.setDataSource(uri.toString());
        int durationTotal =
Integer.parseInt(listSongs.get(position).getDuration()) / 1000;
        durationtotal.setText(formattedTime(durationTotal));
        byte[] art = retriever.getEmbeddedPicture();
        Bitmap bitmap;

        if(art != null)
        {
            bitmap = BitmapFactory.decodeByteArray(art, 0, art.length);
            ImageAnimation(this, coverart, bitmap);
            Palette.from(bitmap).generate(new Palette.PaletteAsyncListener()
{
                @Override
                public void onGenerated(@Nullable Palette palette) {
                    Palette.Swatch swatch = palette.getDominantSwatch();
                    if(swatch != null)
                    {
                        ImageView gredient =
findViewById(R.id.layoutviewgredient);
                        RelativeLayout mContainer =
findViewById(R.id.mContainer);

gredient.setBackgroundResource(R.drawable.gredientbg);
                        mContainer.setBackgroundResource(R.drawable.mainbg);
                        GradientDrawable gradientDrawable = new
GradientDrawable(GradientDrawable.Orientation.BOTTOM_TOP,
                                new int[]{swatch.getRgb(), 0x00000000});
                        gredient.setBackground(gradientDrawable);
                        GradientDrawable gradientDrawableBg = new
GradientDrawable(GradientDrawable.Orientation.BOTTOM_TOP,
                                new int[]{swatch.getRgb(),
swatch.getRgb()});
                        mContainer.setBackground(gradientDrawableBg);
                        songname.setTextColor(swatch.getTitleTextColor());
                        artistname.setTextColor(swatch.getBodyTextColor());
                    }
                    else
                    {
                        ImageView gredient =
findViewById(R.id.layoutviewgredient);
                        RelativeLayout mContainer =
findViewById(R.id.mContainer);

gredient.setBackgroundResource(R.drawable.gredientbg);
                        mContainer.setBackgroundResource(R.drawable.mainbg);
                        GradientDrawable gradientDrawable = new
GradientDrawable(GradientDrawable.Orientation.BOTTOM_TOP,
```

```
                                new int[]{0xff000000, 0x00000000});
                        gredient.setBackground(gradientDrawable);
                        GradientDrawable gradientDrawableBg = new
GradientDrawable(GradientDrawable.Orientation.BOTTOM_TOP,
                                new int[]{0xff000000, 0xff000000});
                        mContainer.setBackground(gradientDrawableBg);
                        songname.setTextColor(Color.WHITE);
                        artistname.setTextColor(Color.DKGRAY);
                    }
                }

            });
        }
        else
        {
            Glide.with(this)
                    .asBitmap()
                    .load(R.drawable.musicpic)
                    .into(coverart);
            ImageView gredient = findViewById(R.id.layoutviewgredient);
            RelativeLayout mContainer = findViewById(R.id.mContainer);
            gredient.setBackgroundResource(R.drawable.gredientbg);
            mContainer.setBackgroundResource(R.drawable.mainbg);
            songname.setTextColor(Color.WHITE);
            artistname.setTextColor(Color.DKGRAY);
        }
    }

    public void ImageAnimation(final Context context, final ImageView
imageview, final Bitmap bitmap)
    {
        Animation aniOut = AnimationUtils.loadAnimation(context,
android.R.anim.fade_out);
        final Animation aniIn = AnimationUtils.loadAnimation(context,
android.R.anim.fade_in);
        aniOut.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {

            }

            @Override
            public void onAnimationEnd(Animation animation)
            {
                Glide.with(context).load(bitmap).into(imageview);
                aniIn.setAnimationListener(new Animation.AnimationListener()
{
                    @Override
                    public void onAnimationStart(Animation animation) {

                    }

                    @Override
                    public void onAnimationEnd(Animation animation) {

                    }

                    @Override
                    public void onAnimationRepeat(Animation animation) {
```

```
                    }
                });
                imageview.startAnimation(aniIn);
            }

            @Override
            public void onAnimationRepeat(Animation animation) {

            }
        });
        imageview.startAnimation(aniOut);
    }

    @Override
    public void onServiceConnected(ComponentName componentName, IBinder
service)
    {
        MusicService.MyBinder myBinder = (MusicService.MyBinder) service;
        musicService = myBinder.getService();
        //Toast.makeText(this, "Connected" + musicService,
Toast.LENGTH_SHORT).show();
        musicService.setCallBack(this);
        seekbar.setMax(musicService.getDuration() / 1000);
        metaData(uri);
        songname.setText(listSongs.get(position).getTiles());
        artistname.setText(listSongs.get(position).getArtist());
        musicService.OnCompleted();
        musicService.showNotification(R.drawable.ic_baseline_pause_24);
    }

    @Override
    public void onServiceDisconnected(ComponentName componentName)
    {
        musicService = null;
    }

    private class SwipeListener implements View.OnTouchListener{
       GestureDetector gestureDetector;

       SwipeListener(View view) {
           int threshold = 100;
           int velocity_threshold = 100;

           GestureDetector.SimpleOnGestureListener listener = new
GestureDetector.SimpleOnGestureListener() {
               @Override
               public boolean onDown(MotionEvent e) {
                   return true;
               }

               @Override
               public boolean onDoubleTap(MotionEvent e) {
                   playpausebtnCliked();
                   return true;
               }

               @Override
               public boolean onFling(MotionEvent e1, MotionEvent e2, float
velocityX, float velocityY) {
                   float xDiff = e2.getX() - e1.getX();
```

```
                    float yDiff = e2.getY() - e1.getY();
                    try {
                        if (Math.abs(xDiff) > Math.abs(yDiff)) {
                            if (Math.abs(xDiff) > threshold &&
Math.abs(velocityX) > velocity_threshold) {
                                if (xDiff > 0) {
                                    prevbtnCliked();
                                }else {
                                    nextbtnCliked();
                                }
                                return true;
                            }
                        }
                    }catch (Exception e) {
                        e.printStackTrace();
                    }
                    return false;
                }
            };
            gestureDetector = new GestureDetector(listener);
            view.setOnTouchListener(this);
        }

        @Override
        public boolean onTouch(View v, MotionEvent event) {
            return gestureDetector.onTouchEvent(event);
        }
    }
}
```

A.27 SetTimerActivity.java

```
package com.example.mp3;

import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.util.Log;
import android.view.View;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.NumberPicker;
import android.widget.Toast;

public class SetTimerActivity extends Activity {

    private static final String LOG_TAG = SetTimerActivity.class.getName();

    private static final String HOURS_KEY = MainActivity.class.getName() +
".hours";
    private static final String MINUTES_KEY = MainActivity.class.getName()
+ ".minutes";

    private static final int MIN_HOURS = 0;
    private static final int MAX_HOURS = 9;
    private static final int MIN_MINUTES = 0;
    private static final int MAX_MINUTES = 59;

    // By default, the timer will be set to one hour
    private static final int DEFAULT_HOURS = 1;
    private static final int DEFAULT_MINUTES = 0;

    private NumberPicker hoursPicker;
    private NumberPicker minutesPicker;
    private TimerManager timerManager;
    private SharedPreferences sharedPreferences;
    private CountdownNotifier countdownNotifier;
    private PauseMusicNotifier pauseMusicNotifier;


    @Override
    protected final void onCreate(Bundle savedInstanceState) {
        onCreate(
                savedInstanceState,
                TimerManager.get(this),
                PreferenceManager.getDefaultSharedPreferences(this),
                CountdownNotifier.get(this),
                PauseMusicNotifier.get(this));

        ImageView backBtn = findViewById(R.id.sleep_back_btn);
        backBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(SetTimerActivity.this,
MainActivity.class));
            }
        });
    }
```

```
    /**
     * Initializes the activity's dependencies.
     *
     * @param savedInstanceState The activity's previous state
     * @param timerManager The timer manager to use
     * @param sharedPreferences The shared preferences to use
     * @param countdownNotifier The countdown notifier to use
     * @param pauseMusicNotifier The pause music notifier to use
     */
    protected void onCreate(
            Bundle savedInstanceState,
            TimerManager timerManager,
            SharedPreferences sharedPreferences,
            CountdownNotifier countdownNotifier,
            PauseMusicNotifier pauseMusicNotifier) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_set_timer);

        // Prevent the soft keyboard from appearing until explicitly
launched by the user

getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_AL
WAYS_HIDDEN);

        hoursPicker = (NumberPicker) findViewById(R.id.hours_picker);
        hoursPicker.setMinValue(MIN_HOURS);
        hoursPicker.setMaxValue(MAX_HOURS);
        hoursPicker.setValue(sharedPreferences.getInt(HOURS_KEY,
DEFAULT_HOURS));

        minutesPicker = (NumberPicker) findViewById(R.id.minutes_picker);
        minutesPicker.setMinValue(MIN_MINUTES);
        minutesPicker.setMaxValue(MAX_MINUTES);
        minutesPicker.setValue(sharedPreferences.getInt(MINUTES_KEY,
DEFAULT_MINUTES));

        this.timerManager = timerManager;
        this.sharedPreferences = sharedPreferences;
        this.countdownNotifier = countdownNotifier;
        this.pauseMusicNotifier = pauseMusicNotifier;
    }

    /**
     * Starts a countdown timer based on the current settings.
     *
     * @param view The view that triggered this action
     */
    public void startTimer(View view) {
        Log.d(LOG_TAG, "Sleep timer started by view " + view.getId());
        hoursPicker.clearFocus();
        minutesPicker.clearFocus();

        int hours = hoursPicker.getValue();
        int minutes = minutesPicker.getValue();

        // The currently selected hour and minute values should become the
new defaults
```

```
        setDefaultTimerLength(hours, minutes);

        // It is possible that a previous music paused notification is
still active; remove it
        pauseMusicNotifier.cancelNotification();

        timerManager.setTimer(hours, minutes);

        countdownNotifier.postNotification(timerManager.getScheduledTime());

        Toast.makeText(this, R.string.timer_started,
Toast.LENGTH_SHORT).show();

        setResult(RESULT_OK);
        finish();
    }

    public void cancelTimer(View view) {
        startActivity(new Intent(SetTimerActivity.this,
MainActivity.class));
    }

    /**
     * Sets the default timer length to the specified number of hours and
minutes.
     *
     * @param hours The number of hours
     * @param minutes The number of minutes
     */
    private void setDefaultTimerLength(int hours, int minutes) {
        sharedPreferences.edit()
                .putInt(HOURS_KEY, hours)
                .putInt(MINUTES_KEY, minutes)
                .commit();
    }

}
```

A.28 Settings.java

```
package com.example.mp3;

public class Settings {

    public static boolean        isEqualizerEnabled  = true;
    public static boolean        isEqualizerReloaded = true;
    public static int[]          seekbarpos          = new int[5];
    public static int            presetPos;
    public static short          reverbPreset        = -1;
    public static short          bassStrength        = -1;
    public static EqualizerModel equalizerModel;
    public static double         ratio               = 1.0;
    public static boolean        isEditing           = false;
}
```

A.29 SongsFragment.java

```
package com.example.mp3;

import static com.example.mp3.MainActivity.musicFiles;
import static com.example.mp3.MusicAdapter.mFiles;

import android.os.Bundle;

import androidx.fragment.app.Fragment;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

public class SongsFragment extends Fragment {

    RecyclerView recyclerView;
    static MusicAdapter musicAdapter;

    public SongsFragment() {
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_songs, container,
false);
        recyclerView = view.findViewById(R.id.recycleview);
        recyclerView.setHasFixedSize(true);
        if(!(musicFiles.size() < 1))
        {
            musicAdapter = new MusicAdapter(getContext(), musicFiles);
            recyclerView.setAdapter(musicAdapter);
            recyclerView.setLayoutManager(new
LinearLayoutManager(getContext(), RecyclerView.VERTICAL, false));
        }
        return view;
    }
}
```

A.30 SplashScreen.java

```
package com.example.mp3;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;

public class SplashScreen extends Activity
{
    Handler handler;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_splash_screen);

        handler=new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent=new Intent(SplashScreen.this,
MainActivity.class);
                startActivity(intent);
                finish();
            }
        },3000);
    }
}
```

A.31 TimerManager.java

```
package com.example.mp3;

import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.preference.PreferenceManager;

import java.util.Calendar;
import java.util.Date;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentMap;

public class TimerManager {

    private static final String SCHEDULED_TIME_KEY =
TimerManager.class.getName() + ".scheduledTime";

    private static ConcurrentMap<String, TimerManager> allInstances = new
ConcurrentHashMap<String, TimerManager>();

    private final Context context;
    private final SharedPreferences sharedPreferences;
    private final AlarmManager alarmManager;

    /**
     * Constructs an instance of {@link TimerManager}.
     *
     * @param context The context
     */
    private TimerManager(Context context) {
        this(
                context,
                PreferenceManager.getDefaultSharedPreferences(context),
                (AlarmManager)
context.getSystemService(Context.ALARM_SERVICE)
        );
    }

    /**
     * Constructs an instance of {@link TimerManager}. Should not be
instantiated directly; call
     * {@link TimerManager#get(Context)} instead.
     *
     * @param context The context
     * @param sharedPreferences The shared preferences to use
     * @param alarmManager The alarm manager to use
     */
    TimerManager(Context context, SharedPreferences sharedPreferences,
AlarmManager alarmManager) {
        this.context = context.getApplicationContext();
        this.sharedPreferences = sharedPreferences;
        this.alarmManager = alarmManager;
    }

    /**
     * Gets an instance of this class for the specified context.
```

```
     *
     * @param context The context. Must not be null.
     *
     * @return A {@link TimerManager}
     */
    public static TimerManager get(Context context) {
        if (context == null) {
            throw new NullPointerException("Argument context cannot be
null");
        }

        String instanceKey = context.getPackageName();

        // A thread safe way of retrieving the TimerManager for the given
context if it already exists, or creating
        // a new instance if not
        TimerManager existingInstance =
allInstances.putIfAbsent(instanceKey, new TimerManager(context));
        if (existingInstance != null) {
            return existingInstance;
        } else {
            // A TimerManager didn't yet exist for the given context;
return the newly created instance
            return allInstances.get(instanceKey);
        }
    }

    /**
     * Sets a timer to pause music playback the given number of hours and
minutes in the future.
     * If a timer is already set for the current context, this will replace
it.
     *
     * @param hours The number of hours. Must be non-negative.
     * @param minutes The number of minutes. Must be non-negative.
     */
    public void setTimer(int hours, int minutes) {
        if (hours < 0) {
            throw new IllegalArgumentException("Argument hours cannot be
negative");
        } else if (minutes < 0) {
            throw new IllegalArgumentException("Argument minutes cannot be
negative");
        }

        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.HOUR, hours);
        calendar.add(Calendar.MINUTE, minutes);

        // NOTE: If an alarm has already been set in this context, this
intent will automatically replace it
        PendingIntent intent = getAlarmIntent();

        // Using the RTC alarm type means the alarm manager will not wake
the device if it is asleep when the time is
        // reached. This is intentional; if the device is asleep, then it
means there is no music currently playing
        // and there's no point in waking it up.
        alarmManager.set(AlarmManager.RTC, calendar.getTimeInMillis(),
intent);
```

```
        // Save the currently scheduled time
        sharedPreferences.edit()
                .putLong(SCHEDULED_TIME_KEY, calendar.getTimeInMillis())
                .commit();
    }

    /**
     * Cancels the timer for the current context. If no timer is currently
set, this will do nothing.
     */
    public void cancelTimer() {
        PendingIntent intent = getAlarmIntent();

        alarmManager.cancel(intent);

        sharedPreferences.edit()
                .remove(SCHEDULED_TIME_KEY)
                .commit();
    }

    /**
     * Returns a {@link android.app.PendingIntent} that can be used to
create or cancel a pending pause music alarm.
     *
     * @return A {@link android.app.PendingIntent}
     */
    private PendingIntent getAlarmIntent() {
        return PendingIntent.getBroadcast(context, 0, new Intent(context,
PauseMusicReceiver.class), 0);
    }

    /**
     * Returns the date and time that the timer is set to expire.
     *
     * @return A {@link Date}, or null if no timer is currently set
     */
    public Date getScheduledTime() {
        if (!sharedPreferences.contains(SCHEDULED_TIME_KEY)) {
            return null;
        }

        long millis = sharedPreferences.getLong(SCHEDULED_TIME_KEY,
Long.MIN_VALUE);

        return new Date(millis);
    }
}
```

A.32 VisualizerViewActivity.java

```
package com.example.mp3;

import android.view.View;
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import android.util.AttributeSet;

public class VisualizerViewActivity extends View {

    private byte[] mBytes;
    private float[] mPoints;
    private Rect mRect = new Rect();
    private Paint mForePaint = new Paint();

    public VisualizerViewActivity(Context context) {
        super(context);
        init();
    }

    public VisualizerViewActivity(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public VisualizerViewActivity(Context context, AttributeSet attrs, int
defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }

    private void init() {
        mBytes = null;
        mForePaint.setStrokeWidth(1f);
        mForePaint.setAntiAlias(true);
        mForePaint.setColor(Color.rgb(0, 128, 255));
    }

    public void updateVisualizer(byte[] bytes) {
        mBytes = bytes;
        invalidate();
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        if (mBytes == null) {
            return;
        }
        if (mPoints == null || mPoints.length < mBytes.length * 4) {
            mPoints = new float[mBytes.length * 4];
        }
        mRect.set(0, 0, getWidth(), getHeight());
        for (int i = 0; i < mBytes.length - 1; i++) {
            mPoints[i * 4] = mRect.width() * i / (mBytes.length - 1);
            mPoints[i * 4 + 1] = mRect.height() / 2
```

```
                    + ((byte) (mBytes[i] + 128)) * (mRect.height() / 2) /
128;
            mPoints[i * 4 + 2] = mRect.width() * (i + 1) / (mBytes.length -
1);
            mPoints[i * 4 + 3] = mRect.height() / 2
                    + ((byte) (mBytes[i + 1] + 128)) * (mRect.height() / 2)
                    / 128;
        }
        canvas.drawLines(mPoints, mForePaint);
    }
}
```

A.33 activity_album_details.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary"
    tools:context=".AlbumDetails">

    <ImageView
        android:id="@+id/albumphoto"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:src="@drawable/musicpic"
        android:scaleType="centerCrop"/>

    <ImageView
        android:id="@+id/gradient"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:src="@drawable/gredientbg"/>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycleview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/albumphoto"/>


</RelativeLayout>
```

A.34 activity_audio_preview.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/background_light"
    android:orientation="vertical">

    <TextView
        android:id="@+id/tvInstruction"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginTop="10dp"
        android:gravity="center"
        android:textColor="#000000" />

    <com.example.mp3.VisualizerViewActivity
        android:id="@+id/visualizerView"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:layout_marginBottom="10dp" />

</LinearLayout>
```

A.35 activity_audio_trim.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:rsb="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/mainlayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fadeScrollbars="false"
    android:layout_weight="1"
    android:fillViewport="true"
    android:background="@color/colorWindowBackground">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="?android:attr/actionBarSize"
        android:layout_marginTop="10dp"
        android:background="@android:color/transparent">

        <ImageView
            android:id="@+id/trim_back_btn"
            android:layout_width="53dp"
            android:layout_height="44dp"
            android:layout_alignParentStart="true"
            android:background="@drawable/custom_ripple_2"
            android:clickable="true"
            android:focusable="true"
            android:padding="12dp"
            android:src="@drawable/back2" />

        <TextView
            android:id="@+id/trim_fragment_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerHorizontal="true"
            android:text="@string/trim"
            android:textAllCaps="true"
            android:textColor="#FFFFFF"
            android:textSize="25sp" />

        <TextView
            android:id="@+id/name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_above="@id/rangeSeekBar"
            android:layout_centerHorizontal="true"
            android:layout_margin="20dp" />

        <org.florescu.android.rangeseekbar.RangeSeekBar
            android:id="@+id/rangeSeekBar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_marginStart="5dp"
            android:layout_marginTop="5dp"
            android:layout_marginEnd="5dp"
            android:layout_marginBottom="5dp"
            app:alwaysActive="true"
```

```
        app:showLabels="false"
        app:thumbDisabled="@drawable/ic_baseline_circle_24"
        app:thumbNormal="@drawable/ic_baseline_circle_24"
        app:thumbPressed="@drawable/ic_baseline_circle_24"
        rsb:activeColor="#a4c639"
        rsb:textAboveThumbsColor="#FFFFFFFF" />

    <TextView
        android:id="@+id/tvLeft"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/rangeSeekBar"
        android:layout_alignStart="@+id/rangeSeekBar"
        android:layout_below="@+id/rangeSeekBar"
        android:textColor="#FFFFFFFF" />

    <TextView
        android:id="@+id/tvRight"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignEnd="@+id/rangeSeekBar"
        android:layout_alignRight="@+id/rangeSeekBar"
        android:layout_below="@+id/rangeSeekBar"
        android:textColor="#FFFFFFFF" />

    <Button
        android:id="@+id/uploadAudio"
        android:layout_width="130dp"
        android:layout_height="60dp"
        android:layout_below="@+id/tvLeft"
        android:layout_marginTop="26dp"
        android:layout_toLeftOf="@+id/name"
        android:backgroundTint="#FFFFFFFF"
        android:gravity="center"
        android:padding="5dp"
        android:singleLine="true"
        android:text="Add Audio"
        android:textColor="@color/black"
        android:textSize="16sp" />

    <Button
        android:id="@+id/extractAudio"
        android:layout_width="130dp"
        android:layout_height="60dp"
        android:layout_alignTop="@+id/uploadAudio"
        android:layout_toEndOf="@+id/name"
        android:layout_toRightOf="@+id/name"
        android:gravity="center"
        android:padding="2dp"
        android:singleLine="true"
        android:text="Cut Audio"
        android:backgroundTint="#FFFFFFFF"
        android:textColor="@color/black"
        android:textSize="16sp" />

    <Button
        android:id="@+id/cancelbutton"
        android:layout_width="200dp"
        android:layout_height="60dp"
        android:layout_alignTop="@+id/uploadAudio"
```

```
            android:layout_centerInParent="true"
            android:layout_marginLeft="-120dp"
            android:layout_marginTop="78dp"
            android:layout_toEndOf="@+id/name"
            android:layout_toRightOf="@+id/name"
            android:backgroundTint="#FFFFFFFF"
            android:gravity="center"
            android:padding="2dp"
            android:singleLine="true"
            android:text="Cancel"
            android:textColor="@color/black"
            android:textSize="16sp" />

    </RelativeLayout>
</ScrollView>
```

A.36 activity_circle_bar.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.chibde.visualizer.CircleBarVisualizer
        android:id="@+id/visualizer"
        android:layout_width="match_parent"
        android:layout_height="350dp"/>

</RelativeLayout>
```

A.37 activity_countdown.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_vertical|center_horizontal"
    tools:context=".CountdownActivity">

    <TableRow>
        <TextView
            android:id="@+id/time_remaining_view"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="36pt" />
    </TableRow>

    <TableRow>
        <Button
            android:id="@+id/cancel_timer_button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:onClick="stopCountdown"
            android:text="@string/cancel_timer_button_label" />
    </TableRow>

</TableLayout>
```

A.38 activity_equalizer.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <FrameLayout
        android:id="@+id/eqFrame"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

A.39 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawerlayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:openDrawer="start">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:elevation="4dp"
            app:titleTextColor="@color/white"
            android:background="@color/black"
            android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />

        <com.google.android.material.tabs.TabLayout
            android:id="@+id/tablayout"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@+id/toolbar"
            app:tabIndicator="@drawable/tab_indicator"
            app:tabIndicatorColor="@color/purple_200"
            app:tabIndicatorFullWidth="true"
            app:tabIndicatorGravity="center"
            app:tabIndicatorHeight="40dp"
            android:background="@color/black"
            app:tabTextColor="@color/colorAccent" />

        <androidx.viewpager.widget.ViewPager
            android:id="@+id/viewpager"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@+id/tablayout"
            android:layout_alignBottom="@+id/fragbottomplayer" />

        <FrameLayout
            android:id="@+id/fragbottomplayer"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true">

            <fragment
                android:name="com.example.mp3.NowPlayingFragmentBottom"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                tools:layout="@layout/fragment_now_playing_bottom" />
```

```
        </FrameLayout>

    </RelativeLayout>

    <com.google.android.material.navigation.NavigationView
        android:id="@+id/navview"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="start"
        app:headerLayout="@layout/nav_header"
        app:menu="@menu/drawer_menu" />

</androidx.drawerlayout.widget.DrawerLayout>
```

A.40 activity_player.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/mContainer"
    android:background="@drawable/mainbg"
    android:orientation="vertical"
    tools:context=".PlayerActivity">

    <RelativeLayout
        android:id="@+id/layouttopbtn"
        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_alignParentTop="true"
        android:background="@drawable/gredientbg">

        <ImageView
            android:id="@+id/backbtn"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_alignParentStart="true"
            android:layout_centerVertical="true"
            android:src="@drawable/ic_baseline_chevron_left_24" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_centerVertical="true"
            android:layout_toEndOf="@id/backbtn"
            android:background="@android:color/transparent"
            android:gravity="center_horizontal"
            android:text="Now Playing"
            android:textColor="@color/colorAccent"
            android:textSize="18sp"
            android:textStyle="bold" />

    </RelativeLayout>

    <RelativeLayout
        android:id="@+id/card"
        android:layout_width="match_parent"
        android:layout_height="350dp"
        android:layout_below="@id/layouttopbtn">

        <ImageView
            android:id="@+id/coverart"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:scaleType="centerCrop"
            android:src="@drawable/musicpic" />

        <ImageView
            android:id="@+id/layoutviewgredient"
            android:layout_width="match_parent"
            android:layout_height="250dp"
            android:layout_alignParentBottom="true" />
```

```
</RelativeLayout>

<TextView
    android:id="@+id/songname"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="This is Song Name"
    android:layout_below="@id/card"
    android:textColor="@color/colorAccent"
    android:gravity="center_horizontal"
    android:textSize="22sp"
    android:textStyle="bold"
    android:singleLine="true"
    android:ellipsize="end"
    android:layout_marginStart="32dp"
    android:layout_marginEnd="32dp"/>

<TextView
    android:id="@+id/songartist"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="This is Song Artist"
    android:layout_below="@id/songname"
    android:textColor="@color/colorAccent"
    android:gravity="center_horizontal"
    android:textSize="18sp"
    android:singleLine="true"
    android:ellipsize="end"
    android:layout_marginStart="40dp"
    android:layout_marginEnd="40dp"/>

<RelativeLayout
    android:id="@+id/relativelayoutforbottom"
    android:layout_width="match_parent"
    android:layout_height="60dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="30dp"
    android:background="@drawable/gredientbg">

    <ImageView
        android:id="@+id/shuffle"
        android:layout_width="35dp"
        android:layout_height="35dp"
        android:layout_alignParentStart="true"
        android:layout_centerVertical="true"
        android:layout_marginStart="32dp"
        android:src="@drawable/ic_baseline_shuffle_24" />

    <ImageView
        android:id="@+id/prev"
        android:layout_width="35dp"
        android:layout_height="35dp"
        android:layout_centerVertical="true"
        android:layout_marginEnd="16dp"
        android:layout_toStartOf="@+id/playpause"
        android:src="@drawable/ic_baseline_skip_previous_24" />
```

```
<com.google.android.material.floatingactionbutton.FloatingActionButton
            android:id="@+id/playpause"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:clickable="true"
            android:focusable="true"
            android:src="@drawable/ic_baseline_play_arrow_24" />

        <ImageView
            android:id="@+id/next"
            android:layout_width="35dp"
            android:layout_height="35dp"
            android:layout_centerVertical="true"
            android:layout_marginStart="16dp"
            android:layout_toEndOf="@+id/playpause"
            android:src="@drawable/ic_baseline_skip_next_24" />

        <ImageView
            android:id="@+id/repeat"
            android:layout_width="35dp"
            android:layout_height="35dp"
            android:layout_alignParentEnd="true"
            android:layout_centerVertical="true"
            android:layout_marginEnd="32dp"
            android:layout_toStartOf="@+id/playpause"
            android:src="@drawable/ic_baseline_repeat_24" />

    </RelativeLayout>

    <RelativeLayout
        android:id="@+id/seekbarlayout"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_above="@+id/relativelayoutforbottom"
        android:layout_marginBottom="30dp">

        <TextView
            android:id="@+id/duration"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_marginStart="10dp"
            android:text="1:28"
            android:textColor="@color/colorAccent" />

        <TextView
            android:id="@+id/durationtotal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true"
            android:layout_marginStart="10dp"
            android:text="4:28"
            android:textColor="@color/colorAccent" />

        <SeekBar
            android:id="@+id/seekbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
```

```
            android:layout_marginEnd="20dp" />


    </RelativeLayout>
</RelativeLayout>
```

A.41 activity_prompt.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/layout_root"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="10dp" >

    <TextView
        android:id="@+id/textView2"
        android:theme="@style/AlertDialogCustom"
        android:text="Save File"
        android:textSize="20dp"
        android:textStyle="bold"
        android:paddingBottom="16dp"
        android:textColor="@color/black"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <TextView
        android:id="@+id/textView1"
        android:textColor="@color/black"
        android:text="Please Enter the Filename"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <EditText
        android:id="@+id/editTextDialogUserInput"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AlertDialogCustom"
        android:textColor="@color/black">

        <requestFocus />

    </EditText>

</LinearLayout>
```

A.42 activity_set_timer.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    android:layout_gravity="center_vertical|center_horizontal"
    tools:context=".SetTimerActivity">

    <ImageView
        android:id="@+id/sleep_back_btn"
        android:layout_width="53dp"
        android:layout_height="44dp"
        android:layout_marginTop="14dp"
        android:layout_alignParentStart="true"
        android:background="@drawable/custom_ripple_2"
        android:clickable="true"
        android:focusable="true"
        android:padding="8dp"
        android:src="@drawable/back2" />

    <TextView
        android:id="@+id/trim_fragment_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="16dp"
        android:text="@string/sleep"
        android:textAllCaps="true"
        android:textColor="#FFFFFF"
        android:textSize="25sp" />

    <TableLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="170dp"
        android:layout_marginLeft="115dp"
        android:layout_gravity="center_vertical|center_horizontal">

        <TableRow>
            <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="center_horizontal|center_vertical"
                android:orientation="horizontal"
                tools:ignore="UselessParent">

            <NumberPicker
                android:id="@+id/hours_picker"
                android:theme="@style/DefaultNumberPickerTheme"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="12pt"
                android:textColor="@color/white"
                android:text="@string/hours_label" />
```

```
            <Space
                android:layout_width="20dp"
                android:layout_height="match_parent" />

            <NumberPicker
                android:id="@+id/minutes_picker"
                android:theme="@style/DefaultNumberPickerTheme"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content" />
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="12pt"
                android:textColor="@color/white"
                android:text="@string/minutes_label" />
            </LinearLayout>
        </TableRow>

        <TableRow>
            <Button
                android:id="@+id/start_timer_button"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="50dp"
                android:onClick="startTimer"
                android:background="@color/white"
                android:textColor="@color/black"
                android:text="@string/start_timer_button_label" >

                <requestFocus />
            </Button>
        </TableRow>

        <TableRow>

            <Button
                android:id="@+id/cancel_timer_button"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginTop="25dp"
                android:background="@color/white"
                android:onClick="cancelTimer"
                android:text="Cancel"
                android:textColor="@color/black">

                <requestFocus />
            </Button>

        </TableRow>

    </TableLayout>

</RelativeLayout>
```

A.43 activity_splash_screen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:background="@color/black">

    <ImageView
        android:id="@+id/logo_id"
        android:layout_width="250dp"
        android:layout_height="250dp"
        android:layout_centerInParent="true"
        android:src="@drawable/music"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/logo_id"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="33dp"
        android:text="MP3 MUSIC PLAYER"
        android:textStyle="bold"
        android:textColor="@color/white"
        android:textSize="30dp" />

</RelativeLayout>
```

A.44 albumitem.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="170dp"
    android:layout_height="200dp"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/albumitem"
    app:cardCornerRadius="10dp"
    app:cardElevation="10dp"
    android:layout_margin="10dp">

    <RelativeLayout
        android:id="@+id/relativelayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/colorPrimary">

        <ImageView
            android:id="@+id/albumimage"
            android:layout_width="match_parent"
            android:layout_height="170dp"
            android:src="@mipmap/ic_launcher"/>

        <TextView
            android:id="@+id/albumname"
            android:layout_width="140dp"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:text="Album"
            android:textColor="@color/white"
            android:singleLine="true"
            android:gravity="center_horizontal"
            android:layout_centerHorizontal="true"
            android:layout_marginBottom="10dp"
            android:layout_marginTop="10dp"
            android:textStyle="bold"/>

    </RelativeLayout>

</androidx.cardview.widget.CardView>
```

A.45 dialog_fragment_equalizer.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/equalizerLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:clickable="true"
    android:focusable="true">

    <RelativeLayout
        android:id="@+id/equalizer_action_container"
        android:layout_width="match_parent"
        android:layout_height="?android:attr/actionBarSize"
        android:layout_marginTop="10dp">

        <ImageView
            android:id="@+id/equalizer_back_btn"
            android:layout_width="50dp"
            android:layout_height="match_parent"
            android:layout_alignParentStart="true"
            android:layout_centerVertical="true"
            android:clickable="true"
            android:focusable="true"
            android:padding="12dp"
            android:src="@drawable/ic_close_black_24dp" />

        <TextView
            android:id="@+id/equalizer_fragment_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:text="@string/eq"
            android:textAllCaps="true"
            android:textSize="25sp" />

        <androidx.appcompat.widget.SwitchCompat
            android:id="@+id/equalizer_switch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true"
            android:layout_centerVertical="true"
            android:layout_marginEnd="10dp" />

    </RelativeLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/equalizer_action_container"
        android:orientation="vertical">

        <com.db.chart.view.LineChartView
            android:id="@+id/lineChart"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="3"
            android:paddingLeft="15dp"
            android:paddingRight="15dp"
```

```
            android:paddingBottom="5dp" />

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_marginLeft="9dp"
            android:layout_marginRight="9dp"
            android:layout_weight="1"
            android:layoutDirection="ltr"
            >

            <View
                android:layout_width="50dp"
                android:layout_height="match_parent"
                android:layout_alignParentStart="true"
                android:layout_marginStart="20dp" />

            <Spinner
                android:id="@+id/equalizer_preset_spinner"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginStart="5dp" />

        </RelativeLayout>


        <LinearLayout
            android:id="@+id/equalizerContainer"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="6"
            android:orientation="horizontal"
            android:padding="3dp">

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:orientation="vertical">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="8"
                    android:layoutDirection="ltr"
                    >


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBar
                        android:id="@+id/seekBar1"
                        android:layout_width="0dp"
                        android:layout_height="0dp"
                        android:layout_marginTop="20dp"
                        android:padding="10dp"
                        android:thumb="@drawable/custom_equalizer_thumb"
                        app:seekBarRotation="CW270" />

</com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper>
```

```
                <TextView
                    android:id="@+id/textView1"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="1"
                    android:textSize="10dp" />
            </LinearLayout>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:orientation="vertical">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="8"
                    android:layoutDirection="ltr"
                    >


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBar
                        android:id="@+id/seekBar2"
                        android:layout_width="0dp"
                        android:layout_height="0dp"
                        android:layout_marginTop="20dp"
                        android:padding="10dp"
                        android:thumb="@drawable/custom_equalizer_thumb"
                        app:seekBarRotation="CW270" />

</com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper>

                <TextView
                    android:id="@+id/textView2"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="1"
                    android:textSize="10dp" />
            </LinearLayout>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:orientation="vertical">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="8"
                    android:layoutDirection="ltr"
                    >


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBar
                        android:id="@+id/seekBar3"
                        android:layout_width="0dp"
```

```
                        android:layout_height="0dp"
                        android:layout_marginTop="20dp"
                        android:padding="10dp"
                        android:thumb="@drawable/custom_equalizer_thumb"
                        app:seekBarRotation="CW270" />

</com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper>

                <TextView
                    android:id="@+id/textView3"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="1"
                    android:textSize="10dp" />
            </LinearLayout>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:orientation="vertical">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="8"
                    android:layoutDirection="ltr"
                    >


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBar
                        android:id="@+id/seekBar4"
                        android:layout_width="0dp"
                        android:layout_height="0dp"
                        android:layout_marginTop="20dp"
                        android:padding="10dp"
                        android:thumb="@drawable/custom_equalizer_thumb"
                        app:seekBarRotation="CW270" />

</com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper>

                <TextView
                    android:id="@+id/textView4"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="1"
                    android:textSize="10dp" />
            </LinearLayout>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:orientation="vertical">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
```

```
                    android:layout_weight="8"
                    android:layoutDirection="ltr"
                    >


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBar
                        android:id="@+id/seekBar5"
                        android:layout_width="0dp"
                        android:layout_height="0dp"
                        android:layout_marginTop="20dp"
                        android:padding="10dp"
                        android:thumb="@drawable/custom_equalizer_thumb"
                        app:seekBarRotation="CW270" />

</com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper>

                <TextView
                    android:id="@+id/textView5"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="1"
                    android:textSize="10dp" />
            </LinearLayout>

        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_marginBottom="10dp"
            android:layout_weight="3"
            android:orientation="horizontal">

            <com.example.mp3.AnalogController
                android:id="@+id/controllerBass"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1" />

            <com.example.mp3.AnalogController
                android:id="@+id/controller3D"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1" />
        </LinearLayout>

    </LinearLayout>


</RelativeLayout>
```

A.46 dialog_single_option.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginLeft="9dp"
        android:layout_marginRight="9dp"
        android:background="@drawable/background_dialog"
        android:focusable="true"
        android:focusableInTouchMode="true"
        android:orientation="vertical">

        <TextView
            android:id="@+id/tvDialogHeading"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@drawable/background_redheaderdialog"
            android:gravity="center"
            android:paddingBottom="13dp"
            android:paddingTop="11dp"
            android:textColor="@android:color/white"
            android:textSize="17sp" />


        <TextView
            android:id="@+id/tvDialogText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_marginBottom="46dp"
            android:layout_marginLeft="18dp"
            android:layout_marginRight="18dp"
            android:layout_marginTop="46dp"
            android:gravity="center"
            android:textColor="#000000">


        </TextView>

        <View
            android:layout_width="match_parent"
            android:layout_height="1dp"
            android:background="#cbcbcc" />


        <TextView
            android:id="@+id/tvDialogSubmit"
            android:layout_width="match_parent"
            android:layout_height="49dp"

android:background="@drawable/background_dialog_button_selector"
            android:gravity="center"
            android:paddingBottom="16dp"
            android:paddingTop="16dp"
            android:textColor="#000000" />
```

```
    </LinearLayout>
</ScrollView>
```

A.47 fragment_albums.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".AlbumsFragment">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycleview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/colorPrimary"/>

</RelativeLayout>
```

A.48 fragment_equalizer.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorWindowBackground"
    android:clickable="true"
    android:focusable="true">

    <RelativeLayout
        android:id="@+id/equalizer_action_container"
        android:layout_width="match_parent"
        android:layout_height="?android:attr/actionBarSize"
        android:layout_marginTop="10dp"
        android:background="@android:color/transparent">

        <ImageView
            android:id="@+id/equalizer_back_btn"
            android:layout_width="50dp"
            android:layout_height="match_parent"
            android:layout_alignParentStart="true"
            android:layout_centerVertical="true"
            android:background="@drawable/custom_ripple_2"
            android:clickable="true"
            android:focusable="true"
            android:padding="12dp"
            android:src="@drawable/back2" />

        <TextView
            android:id="@+id/equalizer_fragment_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:text="@string/eq"
            android:textAllCaps="true"
            android:textColor="#FFFFFF"
            android:textSize="25sp" />

        <androidx.appcompat.widget.SwitchCompat
            android:id="@+id/equalizer_switch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true"
            android:layout_centerVertical="true"
            android:layout_marginEnd="10dp"
            app:theme="@style/CustomSwitch" />

    </RelativeLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/equalizer_action_container"
        android:orientation="vertical">

        <com.db.chart.view.LineChartView
            android:id="@+id/lineChart"
            android:layout_width="match_parent"
```

```
            android:layout_height="0dp"
            android:layout_weight="3"
            android:background="@drawable/graph_back_2"
            android:paddingBottom="5dp"
            android:paddingLeft="15dp"
            android:paddingRight="15dp" />

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_marginLeft="9dp"
            android:layout_marginRight="9dp"
            android:layout_weight="1"
            android:background="#33000000"
            android:layoutDirection="ltr"
            >

            <View
                android:id="@+id/showcase_view_equalizer"
                android:layout_width="50dp"
                android:layout_height="match_parent"
                android:layout_alignParentStart="true"
                android:layout_marginStart="20dp" />

            <Spinner
                android:id="@+id/equalizer_preset_spinner"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:background="@android:color/transparent" />

            <ImageView
                android:id="@+id/spinner_dropdown_icon"
                android:layout_width="20dp"
                android:layout_height="20dp"
                android:layout_alignParentEnd="true"
                android:layout_centerVertical="true"
                android:layout_marginEnd="10dp"
                android:src="@drawable/dropdown_icon"
                app:tint="#FFFFFF" />
        </RelativeLayout>


        <LinearLayout
            android:id="@+id/equalizerContainer"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="6"
            android:orientation="horizontal"
            android:padding="3dp">

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:orientation="vertical">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
```

```
                    android:layout_weight="8"
                    android:layoutDirection="ltr"
                    >


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBar
                        android:id="@+id/seekBar1"
                        android:layout_width="0dp"
                        android:layout_height="0dp"
                        android:layout_marginTop="20dp"
                        android:padding="10dp"
                        android:thumb="@drawable/custom_equalizer_thumb"
                        app:seekBarRotation="CW270" />

</com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper>

                <TextView
                    android:id="@+id/textView1"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="1"
                    android:textSize="10dp" />
            </LinearLayout>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:orientation="vertical">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="8"
                    android:layoutDirection="ltr">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBar
                        android:id="@+id/seekBar2"
                        android:layout_width="0dp"
                        android:layout_height="0dp"
                        android:layout_marginTop="20dp"
                        android:padding="10dp"
                        android:thumb="@drawable/custom_equalizer_thumb"
                        app:seekBarRotation="CW270" />

</com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper>

                <TextView
                    android:id="@+id/textView2"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="1"
                    android:textSize="10dp" />
            </LinearLayout>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
```

```
                android:layout_weight="2"
                android:orientation="vertical">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="8"
                    android:layoutDirection="ltr"
                    >


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBar
                        android:id="@+id/seekBar3"
                        android:layout_width="0dp"
                        android:layout_height="0dp"
                        android:layout_marginTop="20dp"
                        android:padding="10dp"
                        android:thumb="@drawable/custom_equalizer_thumb"
                        app:seekBarRotation="CW270" />

</com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper>

                <TextView
                    android:id="@+id/textView3"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="1"
                    android:textSize="10dp" />
            </LinearLayout>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:orientation="vertical">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="8"
                    android:layoutDirection="ltr"
                    >


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBar
                        android:id="@+id/seekBar4"
                        android:layout_width="0dp"
                        android:layout_height="0dp"
                        android:layout_marginTop="20dp"
                        android:padding="10dp"
                        android:thumb="@drawable/custom_equalizer_thumb"
                        app:seekBarRotation="CW270" />

</com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper>

                <TextView
                    android:id="@+id/textView4"
                    android:layout_width="match_parent"
```

```
                    android:layout_height="0dp"
                    android:layout_weight="1"
                    android:textSize="10dp" />
            </LinearLayout>

            <LinearLayout
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="2"
                android:orientation="vertical">


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="8"
                    android:layoutDirection="ltr"
                    >


<com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBar
                        android:id="@+id/seekBar5"
                        android:layout_width="0dp"
                        android:layout_height="0dp"
                        android:layout_marginTop="20dp"
                        android:padding="10dp"
                        android:thumb="@drawable/custom_equalizer_thumb"
                        app:seekBarRotation="CW270" />

</com.h6ah4i.android.widget.verticalseekbar.VerticalSeekBarWrapper>

                <TextView
                    android:id="@+id/textView5"
                    android:layout_width="match_parent"
                    android:layout_height="0dp"
                    android:layout_weight="1"
                    android:textSize="10dp" />
            </LinearLayout>

        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_marginBottom="10dp"
            android:layout_weight="3"
            android:background="@android:color/transparent"
            android:orientation="horizontal">

            <com.example.mp3.AnalogController
                android:id="@+id/controllerBass"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1" />

            <com.example.mp3.AnalogController
                android:id="@+id/controller3D"
                android:layout_width="0dp"
                android:layout_height="match_parent"
                android:layout_weight="1"/>
```

```
        </LinearLayout>

    </LinearLayout>

    <FrameLayout
        android:id="@+id/equalizerBlocker"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/equalizer_action_container"
        android:alpha="0.7"
        android:background="@android:color/transparent"
        android:clickable="true"
        android:visibility="invisible" />

</RelativeLayout>
```

A.49 fragment_music_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <com.google.android.material.tabs.TabLayout
        android:id="@+id/tablayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimaryDark"
        app:tabIndicator="@drawable/tab_indicator"
        app:tabIndicatorColor="@color/purple_200"
        app:tabIndicatorFullWidth="true"
        app:tabIndicatorGravity="center"
        app:tabIndicatorHeight="40dp"
        app:tabTextColor="@color/colorAccent" />

    <androidx.viewpager.widget.ViewPager
        android:id="@+id/viewpager"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/tablayout"
        android:layout_alignBottom="@+id/fragbottomplayer" />

    <FrameLayout
        android:id="@+id/fragbottomplayer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true">

        <fragment
            android:name="com.example.mp3.NowPlayingFragmentBottom"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            tools:layout="@layout/fragment_now_playing_bottom" />

    </FrameLayout>

</RelativeLayout>
```

A.50 fragment_now_playing_bottom.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    tools:context=".NowPlayingFragmentBottom">

    <RelativeLayout
        android:id="@+id/cardbottomplayer"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimaryDark"
        android:padding="5dp">

        <ImageView
            android:id="@+id/bottomalbumart"
            android:layout_width="60dp"
            android:layout_height="60dp"
            android:scaleType="centerCrop"
            android:src="@drawable/musicpic" />

        <ImageView
            android:id="@+id/skipnextbottom"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_alignParentEnd="true"
            android:layout_centerVertical="true"
            android:layout_marginEnd="10dp"
            android:src="@drawable/ic_baseline_skip_next_24" />

        <ImageView
            android:id="@+id/prevbackbottom"
            android:layout_width="30dp"
            android:layout_height="30dp"
            android:layout_centerVertical="true"
            android:layout_marginEnd="10dp"
            android:layout_toStartOf="@+id/playpauseminiplayer"
            android:src="@drawable/ic_baseline_skip_previous_24" />


<com.google.android.material.floatingactionbutton.FloatingActionButton
            android:id="@+id/playpauseminiplayer"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_baseline_play_arrow_24"
            android:layout_toStartOf="@+id/skipnextbottom"
            android:layout_marginEnd="10dp"
            app:fabSize="mini"
            android:layout_centerVertical="true"/>

        <TextView
            android:id="@+id/songnameminiplayer"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="15dp"
            android:layout_marginTop="5dp"
            android:layout_toStartOf="@+id/playpauseminiplayer"
```

```
            android:layout_toEndOf="@+id/bottomalbumart"
            android:maxLines="1"
            android:text="Song Name Is Here"
            android:textColor="@color/white"
            android:textSize="18sp" />

        <TextView
            android:id="@+id/songartistminiplayer"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@+id/songnameminiplayer"
            android:layout_marginStart="15dp"
            android:layout_marginTop="5dp"
            android:layout_toStartOf="@+id/playpauseminiplayer"
            android:layout_toEndOf="@+id/bottomalbumart"
            android:maxLines="1"
            android:text="Artist Name"
            android:textColor="@color/white"
            android:textSize="15sp" />

    </RelativeLayout>

</FrameLayout>
```

A.51 fragment_songs.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SongsFragment">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycleview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/black"/>

</RelativeLayout>
```

A.52 music_items.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="5dp"
    android:background="@color/black"
    android:id="@+id/audio_item" >

    <ImageView
        android:id="@+id/music_img"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:padding="5dp"
        android:src="@mipmap/ic_launcher_round" />

    <TextView
        android:id="@+id/musicfilename"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="File Name"
        android:layout_toEndOf="@+id/music_img"
        android:layout_marginStart="10dp"
        android:layout_centerVertical="true"
        android:textColor="@color/colorAccent"/>

</RelativeLayout>
```

A.53 nav_header.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="176dp"
    android:background="@color/purple_200"
    android:gravity="bottom"
    android:orientation="vertical"
    android:padding="16dp"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_baseline_settings_24" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="8dp"
        android:text="Quick Browse"
        android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Fast and Easy" />

</LinearLayout>
```

A.54 spinner_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/text1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:minHeight="?android:attr/listPreferredItemHeightSmall"
    android:paddingEnd="?android:attr/listPreferredItemPaddingEnd"
    android:paddingLeft="?android:attr/listPreferredItemPaddingLeft"
    android:paddingRight="?android:attr/listPreferredItemPaddingRight"
    android:paddingStart="?android:attr/listPreferredItemPaddingStart"
    android:textAppearance="?android:attr/textAppearanceListItemSmall"
    android:textColor="#FFF" />
```

A.55 drawer_menu.xml

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

    <group android:checkableBehavior="single">

    <item android:title="Special Features">
        <menu>
            <item
                android:id="@+id/musiclist"
                android:icon="@drawable/ic_baseline_library_music_24"
                android:title="Music List" />
            <item
                android:id="@+id/trimming"
                android:icon="@drawable/ic_baseline_content_cut_24"
                android:title="Song Trimming" />
            <item
                android:id="@+id/download"
                android:icon="@drawable/ic_baseline_download_24"
                android:title="Download" />
            <item
                android:id="@+id/equalizer"
                android:icon="@drawable/ic_baseline_bar_chart_24"
                android:title="Equalizer" />
        </menu>
    </item>
    </group>

    <item android:title="Settings">
        <menu>
            <item
                android:id="@+id/sleeptimer"
                android:icon="@drawable/ic_baseline_alarm_24"
                android:title="Sleep Timer" />
        </menu>
    </item>

</menu>
```

A.56 equalizer_menu.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/itemEqDialog"
        android:title="@string/show_eq_dialog" />
</menu>
```

A.57 search.xml

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

    <item
        android:id="@+id/searchoptioon"
        android:title="search"
        android:icon="@drawable/ic_baseline_search_24"
        app:showAsAction="ifRoom"
        app:actionViewClass="androidx.appcompat.widget.SearchView"/>

    <item
        android:id="@+id/sortoption"
        android:title="sort by"
        app:showAsAction="never">

        <menu>
            <item
                android:id="@+id/byname"
                android:title="Name"
                app:showAsAction="never"/>

            <item
                android:id="@+id/bydate"
                android:title="Date"
                app:showAsAction="never"/>

            <item
                android:id="@+id/bysize"
                android:title="Size"
                app:showAsAction="never"/>

        </menu>

    </item>

</menu>
```

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 2**
Student Name & ID: Tan Jia Yi 19ACB05612	
Supervisor: Dr. Ooi Chek Yee	
Project Title: MP3-Music Player Application Development	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Review back FYP 1 and do some changes and improvement based on comment given in chapter 1 and 2.

2. WORK TO BE DONE

- Chapter 3 and 4 – UML diagram and Method/Technology Used

3. PROBLEMS ENCOUNTERED

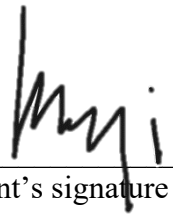
Objective and Contribution are still lack of "wow" effect

4. SELF EVALUATION OF THE PROGRESS

Come out with more special idea based on the review and research on existing application because now are still lack of the special places in this proposed application

Ooi Chek Yee

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 4**
Student Name & ID: Tan Jia Yi 19ACB05612	
Supervisor: Dr. Ooi Chek Yee	
Project Title: MP3-Music Player Application Development	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Chapter 3 and 4 – UML diagram and Method/Technology Used

2. WORK TO BE DONE

- Setting drawer Navigation Menu
- Sleep Timer

3. PROBLEMS ENCOUNTERED

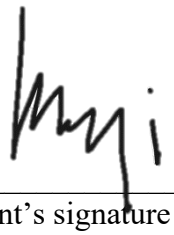
Still not very sure how to perform a complete and detailed UML diagram

4. SELF EVALUATION OF THE PROGRESS

Stuck on UML Diagram drawing for too long

Ooi Chek Yee

Supervisor's signature



Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 6**
Student Name & ID: Tan Jia Yi 19ACB05612	
Supervisor: Dr. Ooi Chek Yee	
Project Title: MP3-Music Player Application Development	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Setting drawer Navigation Menu
- Sleep Timer

2. WORK TO BE DONE

- Gesture Control
- Trimming Function

3. PROBLEMS ENCOUNTERED

The UI is not beautiful and consistent enough. A lot of time wasted when placing the drawer menu because it wasn't fixed in the ideal position.

4. SELF EVALUATION OF THE PROGRESS

Can be speed up the coding pace, wasting too much time in those tiny error - careless

Ooi Chek Yee

_____

Supervisor's signature

_____

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 8**
Student Name & ID: Tan Jia Yi 19ACB05612	
Supervisor: Dr. Ooi Chek Yee	
Project Title: MP3-Music Player Application Development	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Gesture Control
- Trimming Function

2. WORK TO BE DONE

- Download Function
- Equalizer

3. PROBLEMS ENCOUNTERED

Gesture and Trimming are new thing for me, so need a little more time to explore even though there are a lot of info out there but will also make confuse.

4. SELF EVALUATION OF THE PROGRESS

Know more theory behind about the gesture control. Everything is on the right way.

Ooi Chek Yee

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 10**
Student Name & ID: Tan Jia Yi 19ACB05612	
Supervisor: Dr. Ooi Chek Yee	
Project Title: MP3-Music Player Application Development	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Download Function
- Equalizer

2. WORK TO BE DONE

- Testing and Debugging

3. PROBLEMS ENCOUNTERED

Only can download the music from 3rd party website which is illegal. Moreover, there are too little resources about equalizer development out there.

4. SELF EVALUATION OF THE PROGRESS

Everything goes well but there are also some imperfect functions to amend

Ooi Chek Yee

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Y3T3	**Study week no.: 12**
Student Name & ID: Tan Jia Yi 19ACB05612	
Supervisor: Dr. Ooi Chek Yee	
Project Title: MP3-Music Player Application Development	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Chapter 1, 2, 3, 4, and 5
- The Whole MP3 Music Player Application

2. WORK TO BE DONE

- Chapter 6 - Conclusion
- Final Check on Report
- Presentation Slide

3. PROBLEMS ENCOUNTERED

A minor flaw in one of the test modules

4. SELF EVALUATION OF THE PROGRESS

Everything is almost done but the application is still not the perfect one due to limited time and knowledge

Ooi Chek Yee

Supervisor's signature

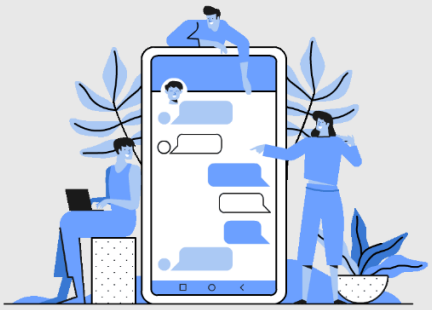
Student's signature

POSTER

UTAR
UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION COMMUNICATION AND TECHNOLOGY

Project Developer: Tan Jia Yi
Project Supervisor: Dr Ooi Chek Yee



MP3 MUSIC PLAYER APPLICATION DEVELOPMENT USING ANDROID

An MP3 music player combines the advantages of the existing music players on the market, removes the unpractical functions and adds practical functions such as downloading music and music scanning.

FEATURES

DOWNLOAD MUSIC

Download Music From 3rd Party Site

MUSIC TRIMMING

Trimm the favourite part of the music

BACKGROUND PLAY

Music will still playing when the screen is off

EQUALIZER

Change the output of the song based on your mood

GESTURE CONTROL

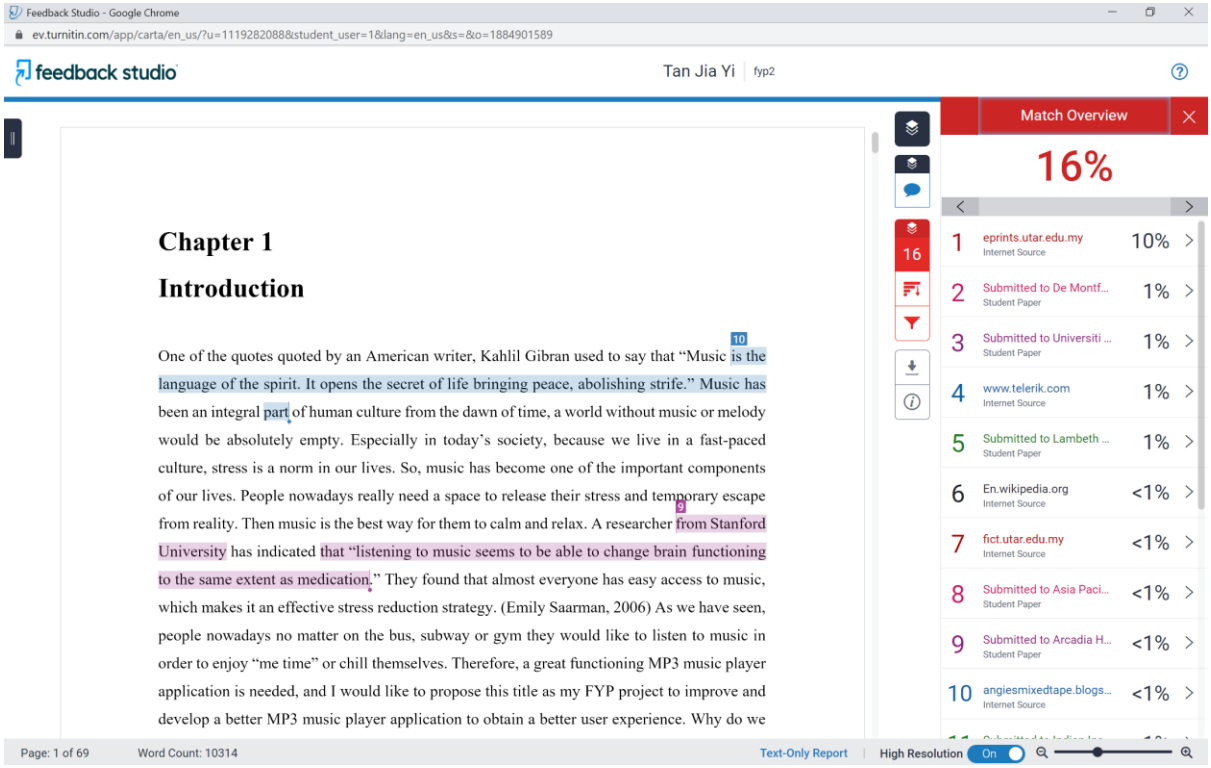
PROPOSED METHOD

Requirements Elicitation → Planning → Design → Development and implementation → Testing → Deployment

BETTER THAN OTHERS?

YES! More customise features for you! Trimm as you like, Tune as you like and Play the songs as you like! Gesture for easy control too!

PLAGIARISM CHECK RESULT



fyp2

ORIGINALITY REPORT

16%	14%	0%	4%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	eprints.utar.edu.my Internet Source	10%
2	Submitted to De Montfort University Student Paper	1%
3	Submitted to Universiti Tunku Abdul Rahman Student Paper	1%
4	www.telerik.com Internet Source	1%
5	Submitted to Lambeth College Student Paper	1%
6	En.wikipedia.org Internet Source	<1%
7	fict.utar.edu.my Internet Source	<1%
8	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
9	Submitted to Arcadia High School Student Paper	<1%

10	angiesmixedtape.blogspot.com Internet Source	<1%
11	Submitted to Indian Institute of Management, Indore Student Paper	<1%
12	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
13	www.prnewswire.com Internet Source	<1%
14	Submitted to Jaipuria Institute of Management Student Paper	<1%
15	weblizar.com Internet Source	<1%
16	fdpp.dilg.gov.ph Internet Source	<1%
17	wawolifu.cf Internet Source	<1%

Universiti Tunku Abdul Rahman			
Form Title : Supervisor's Comments on Originality Report Generated by Turnitin for Submission of Final Year Project Report (for Undergraduate Programmes)			
Form Number: FM-IAD-005	Rev No.: 0	Effective Date: 01/10/2013	Page No.: 1of 1

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

Full Name(s) of Candidate(s)	Tan Jia Yi
ID Number(s)	19ACB05612
Programme / Course	IA
Title of Final Year Project	MP3 - MUSIC PLAYER APPLICATION DEVELOPMENT USING ANDROID

Similarity	**Supervisor's Comments (Compulsory if parameters of originality exceeds the limits approved by UTAR)**
Overall similarity index: 16 % **Similarity by source** Internet Sources: 14 % Publications: 0 % Student Papers: 4 %	References are properly cited.
Number of individual sources listed of more than 3% similarity: 1	
Parameters of originality required and limits approved by UTAR are as Follows: **(i) Overall similarity index is 20% and below, and** **(ii) Matching of individual sources listed must be less than 3% each, and** **(iii) Matching texts in continuous block must not exceed 8 words** ***Note: Parameters (i) – (ii) shall exclude quotes, bibliography and text matches which are less than 8 words.***	

Note Supervisor/Candidate(s) is/are required to provide softcopy of full set of the originality report to Faculty/Institute

Based on the above results, I hereby declare that I am satisfied with the originality of the Final Year Project Report submitted by my student(s) as named above.

Ooi Chek Yee

Signature of Supervisor

Name: Ooi Chek Yee

Date: 23 August 2022

Signature of Co-Supervisor

Name: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY
(KAMPAR CAMPUS)

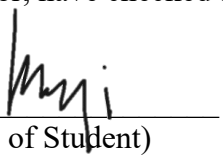
CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	19ACB05612
Student Name	Tan Jia Yi
Supervisor Name	Dr. Ooi Chek Yee

TICK (√)	DOCUMENT ITEMS Your report must include all the items below. Put a tick on the left column after you have checked your report with respect to the corresponding item.
√	Front Plastic Cover (for hardcopy)
√	Title Page
√	Signed Report Status Declaration Form
√	Signed FYP Thesis Submission Form
√	Signed form of the Declaration of Originality
√	Acknowledgement
√	Abstract
√	Table of Contents
√	List of Figures (if applicable)
√	List of Tables (if applicable)
√	List of Symbols (if applicable)
√	List of Abbreviations (if applicable)
√	Chapters / Content
√	Bibliography (or References)
√	All references in bibliography are cited in the thesis, especially in the chapter of literature review
√	Appendices (if applicable)
√	Weekly Log
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result - Form Number: FM-IAD-005)
√	I agree 5 marks will be deducted due to incorrect format, declare wrongly the ticked of these items, and/or any dispute happening for these items in this report.

*Include this form (checklist) in the thesis (Bind together as the last page)

I, the author, have checked and confirmed all the items listed in the table are included in my report.



(Signature of Student)
Date: 23 August 2022