MP3 MUSIC PLAYER APPLICATION DEVELOPMENT USING ANDROID

By

Tan Siang Kian

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology
(Kampar Campus)

JAN 2020

UNIVERSITI TUNKU ABDUL RAHMAN

REPORT STATUS DECLARATION FORM

Title: MP3 MUSIC PLAYER APPLICATION DEVELOPMENT USING ANDROID

Academic Session: JAN 2020

I, TAN SIANG KIAN

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.

Verified by,

(Author's signature)

(Supervisor's signature)

Address:
7,JALAN ROS MERAH 3/19,
TAMAN JOHOR JAYA,
81100 JOHOR BAHRU, JOHOR.

Ooi Chek Yee
Supervisor's name

Date: 20/04/2020

Date: 20/04/2020

MP3 MUSIC PLAYER APPLICATION DEVELOPMENT USING ANDROID

By

Tan Siang Kian

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF INFORMATION TECHNOLOGY (HONS)

COMPUTER ENGINEERING

Faculty of Information and Communication Technology
(Kampar Campus)

JAN 2020

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 Siang Kian

Date : 20/04/2020

ACKNOWLEDGEMENT

I would like to express my sincere thanks to all those individuals who assisted me in completing this Final Year Project. The project could not be going smoothly without the help of these people.

First of all, I would like to express my appreciation to my supervisor, Ts Dr. Ooi Chek Yee. Dr. Ooi who has given me this bright opportunity to engage in this MP3 Music Player app project that is very suitable for fresher developers to enhance my own skill, I feel a million thanks to you. It is the first time to establish an Android application project in my life. You are my enlightenment mentor, and I will never forget your help to me in the future.

Furthermore, I would like to thanks to my moderator, Ts Dr. Khor Siak Wang. Dr. Khor has given me valuable advice to learn from my mistakes and improve on them. His valuable advice made me have more ideas for improvement after Final Year Project 1.

Finally, I must say thanks to my whole family especially my parents for their love, comfort, support, and encouragement for me to keep moving forward in the face of obstacles in my university life.

ABSTRACT

This project is about the mp3 music player application development using Android. The biggest difference between the music player and existing applications is that it is completely free for users to use. It will integrate the advantages of existing music players on the market, as far as possible to mining out the existing music players' function, and then do the filtering in order to eliminate function that not practical or low cost-effective. Also, it will be keep improved based on user feedback.

In addition, depending on the user's usage scenario, the music player will also add some modes, such as driving mode and night mode, to allow users to use the application in any situation or environment. Moreover, the music player will have audio trim features, allowing users to trim the best part of their favorite song into phone ringtone or alarm. On the other hand, the existing music players pay less attention to the control of gestures. Therefore, the music player will solve the limitation by adding more gestures and shake the phone feature for media control to make it more user-friendly and humanity.

In a nutshell, the methodology for developing the mp3 music application used in this project is the agile development cycle. The agile development cycle consists of six phases, which is requirements analysis, planning, design, implementation or development, testing, and deployment. Due to the iterative and flexible nature of this approach, it is able to effectively adapt to users with changing requirements.

TABLE OF CONTENTS

TITLE PAGE ..... **i**
DECLARATION OF ORIGINALITY ..... **ii**
ACKNOWLEDGEMENT ..... **iii**
ABSTRACT ..... **iv**
TABLE OF CONTENTS ..... **v**
LIST OF TABLES ..... **ix**
LIST OF FIGURES ..... **x**
LIST OF ABBREVIATIONS ..... **xii**
Chapter 1 Introduction **1**
1.1 Problem Statement 1
1.2 Background Information and Motivation 2
1.3 Objectives 3
1.4 Proposed Approach/Study 4
1.5 Highlight of What Have Been Achieved 5
1.6 Report Organization 5
Chapter 2 Literature Review **6**
2.1 Review on existing application 6
2.1.1 Review on YY Music 6
2.1.2 Review on Xiami Music 14
2.1.3 Review on JOOX Music 20
2.2 Critical Remarks of previous works 27
2.2.1 Strength and Weaknesses of previous works 27
2.2.2 Application Comparison 28
Chapter 3 System Design **29**
3.1 Site Map 29
3.2 Use Case Diagram 30
3.2.1 Use Case Description 31
3.3 Activity Diagram 36
3.3.1 Listen to music 36
3.3.2 Audio Trim 37
3.3.3 Sleep timer 38
3.3.4 Night mode 39
3.3.5 Change theme 40
3.3.6 Download song 41
3.4 System Wireframe 42
Chapter 4 Methodology and Tools **55**
4.1 Design Specifications 55

4.1.1 Methodology 55
4.2 Tool to use 57
4.2.1 Software Requirement 57
4.2.2 Hardware Requirement 57
4.3 User requirements 58
4.4 System Performance Definition 59
4.5 Timeline 60
Chapter 5 Implementation and Testing 61
5.1 Implementation 61
5.2 Testing 62
Chapter 6 Conclusion 68
6.1 Project Review, Discussions and Conclusion 68
6.1.1 Project Achievement 69
6.1.2 Problem Encountered 70
6.2 Future Work 70
BIBLIOGRAPHY 71
APPENDICES A A-1
A.1 MusicAdapter.java A-1
A.2 MusicPagerAdapter.java A-4
A.3MyApplication.java A-5
A.4 ActivityScope.java A-7
A.5 AppComponent.java A-7
A.6 AppModule.java A-8
A.7 BaseActivity.java A-9
A.8 BaseFragment.java A-11
A.9 BasePresenter.java A-13
A.10 IBaseView.java A-13
A.11 EasyRecyclerViewAdapter.java A-14
A.12 CommonConstant.java A-18
A.13 FFTData.java A-18
A.14 AudioData.java A-18
A.15 CommonDialog.java A-19
A.16 CircleBarRenderer.java A-24
A.17 Renderer.java A-27
A.18 VisualizerView.java A-29
A.19 DensityUtils.java A-34
A.20 LogUtils.java A-34
A.21 FileUtils.java A-35

A.22 Md5Utils.java A-37
A.23 Mp3ScanUtils.java A-38
A.24 ScreenUtils.java A-39
A.25 RingTools.java A-40
A.26 Music.java A-43
A.27 DetectSwipeGestureDriver A-44
A.28 DetectSwipeGestureListener A-45
A.29 FileInfo.java A-46
A.30 FileChooseComponent.java A-47
A.31 HomeComponent.java A-47
A.32 MusicInfo.java A-48
A.33 HomeModule.java A-52
A.34 FileChooseModule.java A-52
A.35 FileChooseContract.java A-53
A.36 HomeContract.java A-54
A.37 FileChoosePresenter.java A-56
A.38 HomePresenter A-58
A.39 CutFragment.java A-66
A.40 FileChooserActivity.java A-76
A.41 Mp3NameConvertUtils.java A-81
A.42 Mp3InfoUtils.java A-82
A.43 Mp3CutLogic.java A-84
A.44 AllFragment.java A-88
A.45 DriverMusicActivity.java A-97
A.46 MainActivity.java A-104
A.47 MusicActivity.java A-117
A.48 ThemeEnum.java A-128
A.49 MusicFragment.java A-129
A.50 RootActivity.java A-132
A.51 ThemeColor.java A-133
A.52 SplashActivity.java A-134
A.53 ShakeListener.java A-139
A.54 ShakeService.java A-142
A.55 ThemeColorAdapter.java A-144
A.56 BasePreference.java A-145
A.57 ThemeColorSelectDialog.java A-146
A.58 AppPreference.java A-148
A.59 AuxiliaryPreference.java A-150

A.60 BroadcastManager.java A-151
A.61 ThemeChangeable.java A-152
A.62 Common.java A-152
A.63 PeriodicTask.java A-153
A.64 Utils.java A-154
A.65 TimeSleepActivity.java A-155
A.66 AnimationUtils.java A-164
A.67 BlurUtil.java A-165
A.68 ColorUtils.java A-171
A.69 MergeImage.java A-175
A.70 activity_driver.xml A-176
A.71 activity_filechooser_show.xml A-178
A.72 activity_main.xml A-180
A.73 activity_music.xml A-181
A.74 activity_splash.xml A-185
A.75 activity_time_sleep.xml A-190
A.76 dialog_common.xml A-191
A.77 dialog_theme_color.xml A-193
A.78 fragment_all.xml A-194
A.79 fragment_cut.xml A-196
A.80 fragment_music.xml A-199
A.81 item_musicfile.xml A-201
A.82 item_theme_color.xml A-202
A.83 music_item.xml A-203
A.84 time_sleep_content.xml A-205
A.85 time_sleep_content_picker.xml A-211

POSTER

PLAGIARISM CHECK RESULT

LIST OF TABLES

Table Number	Title	Page
Table 2-2-1-1	Strength and Weaknesses of reviewed application	27
Table 2-2-2-1	Comparison among reviewed and proposed application	28
Table 3-2-1-1	Use Case Description of Listen to music	31
Table 3-2-1-2	Use Case Description of Progress bar	31
Table 3-2-1-3	Use Case Description of Driving mode	32
Table 3-2-1-4	Use Case Description of Night mode	32
Table 3-2-1-5	Use Case Description of Shake Control	33
Table 3-2-1-6	Use Case Description of Gesture Control	33
Table 3-2-1-7	Use Case Description of Audio trim	34
Table 3-2-1-8	Use Case Description of Set as ringtone or alarm	34
Table 3-2-1-9	Use Case Description of Search music online	35
Table 5-2-1	Unit Testing of Music Player Module	62
Table 5-2-2	Unit Testing of Media Icon Button Playback Control Module	63
Table 5-2-3	Unit Testing of Driver Mode Module	64
Table 5-2-4	Unit Testing of Swipe Gesture Playback Control Module	64
Table 5-2-5	Unit Testing of Shaking Playback Control Module	65
Table 5-2-6	Unit Testing of Audio Trim Module	65
Table 5-2-7	Unit Testing of Sleep Timer Module	66
Table 5-2-8	Unit Testing of Night Mode Module	67
Table 5-2-9	Unit Testing of Change Theme Color Module	67

LIST OF FIGURES

Figure Number	Title	Page
Figure 1-1	System Flowchart of MP3 Music Player	4
Figure 2-1-1-1	Homepage of YY Music	6
Figure 2-1-1-2	Playlists recommended by YY Music	7
Figure 2-1-1-3	Search page of YY Music	8
Figure 2-1-1-4	Music playback control interface and download song option	9
Figure 2-1-1-5	Music play control interface and download song option	10
Figure 2-1-1-6	FM radio page of YY Music	11
Figure 2-1-1-7	Setting page and Floating Window Feature	12
Figure 2-1-1-8	Timer feature	13
Figure 2-1-2-1	Main page of Xiami Music	14
Figure 2-1-2-2	Four categories of the main page and subcategories for each genre	15
Figure 2-1-2-3	Discover page of Xiami Music	16
Figure 2-1-2-4	Video and News page of Xiami Music	17
Figure 2-1-2-5	Driving Mode of Xiami Music	18
Figure 2-1-2-6	Night Mode of Xiami Music	19
Figure 2-1-3-1	Main page of JOOX	20
Figure 2-1-3-2	Me page and local song on the user device	21
Figure 2-1-3-3	My Karaoke feature and recording editor	22
Figure 2-1-3-4	Playlist and Artists category under Discover page	23
Figure 2-1-3-5	Radio and Live page of JOOX	24
Figure 2-1-3-6	Music playback control page and sharing song features	25
Figure 2-1-3-7	VIP privileges are required	26
Figure 3-1-1	Site Map Diagram	29
Figure 3-2-1	Use case diagram of MP3 Music Player Application	30
Figure 3-3-1-1	Activity Diagram for User to Listen to Music	36
Figure 3-3-2-1	Activity Diagram for User Using Audio Trim Feature	37
Figure 3-3-3-1	Activity Diagram for User Enable Sleep Timer	38
Figure 3-3-4-1	Activity Diagram for User Enable Night Mode	39
Figure 3-3-5-1	Activity Diagram for User to Change Theme of Tool Bar	40
Figure 3-3-6-1	Activity Diagram for User to Download Song	41
Figure 3-4-1	Splash screen	42
Figure 3-4-2	Home page	41
Figure 3-4-3	English Song Playlist	43

Figure 3-4-4	Others Song Playlist	42
Figure 3-4-5	Filtering Song Using the Search Bar and Alphabet Quick Scrollbar	44
Figure 3-4-6	Song Playing Page	45
Figure 3-4-7	Song Playback Mode	46
Figure 3-4-8	Driver Mode	47
Figure 3-4-9	Audio Trim Feature	48
Figure 3-4-10	Select Music Page	48
Figure 3-4-11	The Process of Cut Song	49
Figure 3-4-12	Set the Trimmed Song as Ringtone	50
Figure 3-4-13	Generated Trimmed Song	50
Figure 3-4-14	Enable Sleep Timer	51
Figure 3-4-15	Disable Sleep Timer	52
Figure 3-4-16	Enable Night Mode	53
Figure 3-4-17	Change Theme Color of Toolbar	54
Figure 3-4-18	Download Song	54
Figure 4-1-1-1	Agile Development Cycle (Devcrew.io, 2017)	55
Figure 4-5-1	Gantt chart for FYP1	60
Figure 4-5-2	Gantt chart for FYP2	60

LIST OF ABBREVIATIONS

Apps	Applications
CPU	Central processing unit
IOS	iPhone Operating System
MV	Music Video
MP3	MPEG 1 Audio Layer 3
MP4	MPEG LAYER 4
UI	User interface
UTAR	Universiti Tunku Abdul Rahman
VIP	Very Important Person
XML	Full extensible markup language
APK	Android Application Package

Chapter 1 Introduction

1.1 Problem Statement

The problem domains on this project are:

1. Bloated software and user interfaces

Due to the fierce competition between music player applications, many developers tried to add many features, advertise and content to their respective music player in order to retain their users and attract new users. This trend has made it harder for users to get content from their music player, which also means it's harder to filter the content that they want. With the continuous iteration of application and a growing number of features, the music player will become even more bloated and the user's experience will become less smooth. Based on Mehul (2018), users tend to feel frustrated and angry if they take a long time to get a reply from the mobile application, so they will never return to the same application, and 48% of users will simply uninstall or stop using it.

2. Lack of gestures to control

Most music player apps use touch buttons to play, pause and switch between previous and next songs while ignoring the convenience of using gesture swiping to control the music player. For instance, when a user is working and intends to skip to the next song in the music player, he/she have to switch their attention to the console from work and click the button. This problem does not affect music player properly work, but it does have some inconvenience. However, according to Scacca (2020) said that as our physical devices and appliances develop the button-free design, consumers will become more comfortable and confident in this way of interaction, so we should consider using gesture control on more mobile applications.

3. Lack of sorting and searching features

When users continuously to add new songs into the playlist, the difficulty of the songs the user wants to filter will increase. After the songs in the playlist are added to reach hundreds of songs, the user can only search song by continuously swipe up or down. If not carefully check the content, it is possible to miss the songs that the user wants to filter, and then repeat the behavior until the result is found. Therefore, it is an extremely poor experience for users.

1.2 Background Information and Motivation

In modern society, people live a fast-paced life, and pressure is constantly present in lives. Due to the wide use of mobile phones, music has become the daily essential spiritual food, everyone's mobile phone inside there must be a music player. An application like MP3 music players is used to balance stress and happiness. It accompanies people anytime, anywhere and anyplace such as when people taking the bus and exercising.

The mobile MP3 music player application is designed to allow users to listen to music in a more convenient and comfortable way without too much restriction. Moreover, it can play the music properly without interference from advertisements and offline.

Since many developers realize that modern urbanites are living in a stressful situation, they have captured the commercial opportunity, therefore many similar applications have emerged in the market. These applications have easy-to-use interfaces and features that make the user experience better.

However, these existing music players blindly pursue fancy appearance and huge features, resulting in the high utilization rate of users' mobile phones, such as CPU and memory. Whereas, for most normal users, these kind of huge and many features are meaningless. Therefore, this project is designed to dedicate to MP3 music player based on the Android mobile phone platform to optimize performance and simplify to meet user needs.

1.3 Objectives

The objective of this thesis is to propose development of android that:

1. **Make it with a simple feature and run smoothly**

   By using this mp3 music player will make users fell comfortable and relaxed because it will pay more attention to the features commonly used by users, excluding some rarely used features that occupy a large of system processors, making the music player lightweight, simple, but also has powerful basic features.

2. **Support gesture control**

   The MP3 music player will add features triggered by gestures to make it easier for users to use as well as less dependent on touch buttons. For example, a user can skip next or previous songs by simply swiping left and right on the anywhere of the screen in the playing interface.

3. **Support quick search**

   The lack of a search bar in the music list is unacceptable. Therefore, the mp3 music player will use the search bar as well as fast scroll using alphabets on the right side of the screen, allowing users to quickly filter through hundreds of songs to find the ones users want to play.

1.4 Proposed Approach/Study

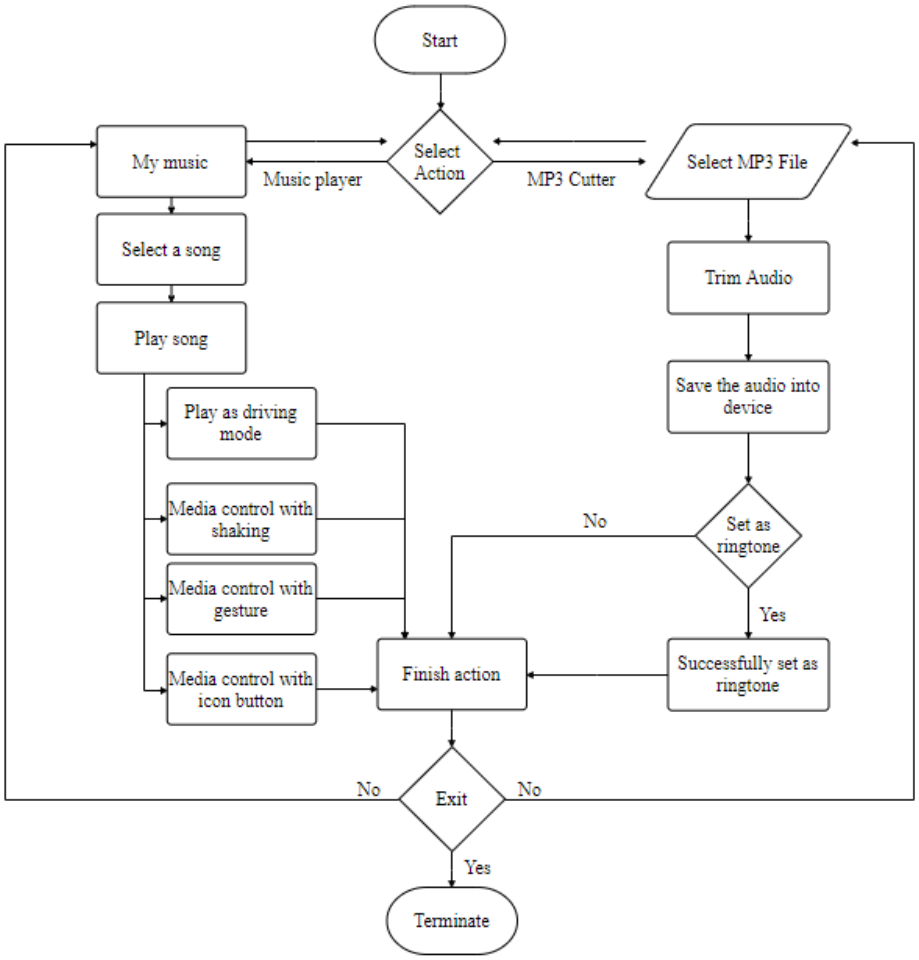


Figure 1-1 System Flowchart of MP3 Music Player

Figure 1-1 shows the system flowchart of the mp3 music player. When the user starts the application, they can select two types of the main action of the application. The main action is divided into the music player and mp3 cutter. The first case is a music player, where users can select a song they want to listen to under the "My music" fragment and click it to play. In the music playing interface, users have a variety of options for action which are playing as driver mode, media control with shaking, media control with the gesture, media control with the icon button. The second case is an mp3 cutter. After the user selects the song he or she wants to trim, the application will start to trim the music and save it to the mobile device. At this point, the application will ask the user if they want to set the trimmed audio to ringtone and then finish the action. After the completion of the action, the user can choose whether to exit the application or not, if "no" back to select action interface, if "yes" terminate the application.

1.5 Highlight of What Have Been Achieved

The main highlight of the project is to make the proposed application become a high learnability application without too many complex features, enhance the interaction between the user and the media control so that the user can have a better experience to achieve real pressure relief. It is worth mentioning that the music player has the audio trim function. Users can trim the best parts of the audio by setting the starting and ending points of the audio, which can be used as a ringtone. In addition, the ability to enhance the interaction between users and media control is that the application can skip songs by shaking the phone under the lock screen status of the phone. Also, the application utilizes the gesture controls to get rid of its reliance on touch buttons. For example, a song can be switched when the user slides left or right on the music playing interface.

1.6 Report Organization

The project report is divided into six chapters. Chapter 1 is about the background and motivation of the proposed application, the problem statement, and objectives to solve the problem statement, proposed approach or study, and highlighting of proposed applications that have been achieved. Chapter 2 contains a review of the three existing music player applications on the market, along with comparisons, strengths, and weaknesses of each. Chapter 3 is the system design including the site map, use case diagram, activity diagram, and system wireframe which is the user interface of the proposed application. Chapter 4 is about software design methodology, tools used, requirements, system performance definitions, and timelines. Chapter 5 is about implementation and testing. Chapter 6 is the project review, discussion, and conclusion, the achievements of the project, the problems encountered in the development process and the future improvement.

Chapter 2 Literature Review

2.1 Review on existing application

2.1.1 Review on YY Music

YY Music is a free music player that can play all music on YouTube, support background playing music, and download the song into local storage for free. Its music library is extremely powerful that covers almost all songs, such as English, Chinese, Cantonese, Japanese and even Korean songs. It performs extremely well in terms of functionality and interaction between users and application.

The homepage of YY Music (Figure 2-1-1-1) is 'Discover', in which random lists and popularity rankings of various music themes are provided such as rank, top 100, latest songs, and weekly 20. When users feel confused in selecting songs or tired of listening to classic songs, this feature allows users to directly click, listen to the latest music.

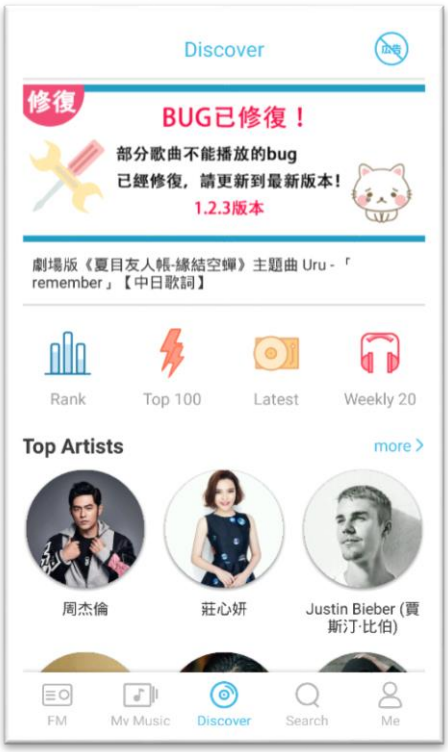


Figure 2-1-1-1 Homepage of YY Music

In addition, the discovery page will also be a display variety of popular artists and playlists (Figure 2-1-1-2), allowing users to quickly listen to favorite artists' music. When the user clicks into the playlist, all the songs will be listed for the user's reference. If the user is satisfied with all the songs in the playlist, just click "Play All" on the top right to start listening to all the songs in the playlist.

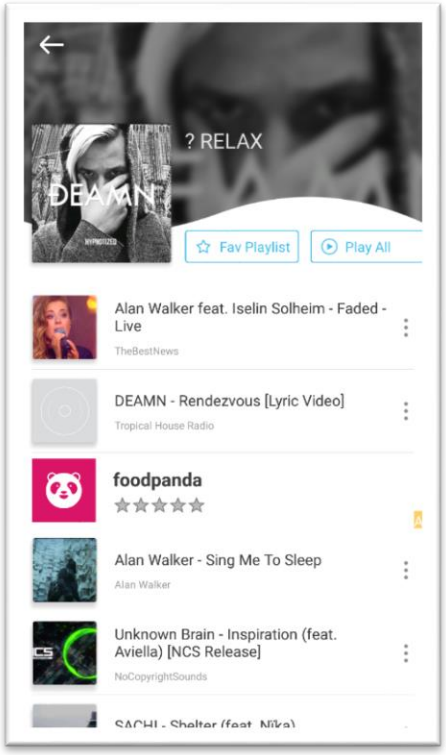


Figure 2-1-1-2 Playlists recommended by YY Music

Furthermore, the search page of YY Music (Figure 2-1-1-3) supports searching according to the singer, song name, album, and others, enabling users to quickly find favorite music songs. The search page also has popular search keywords and recent search history records, make search feature more user-friendly and easy.

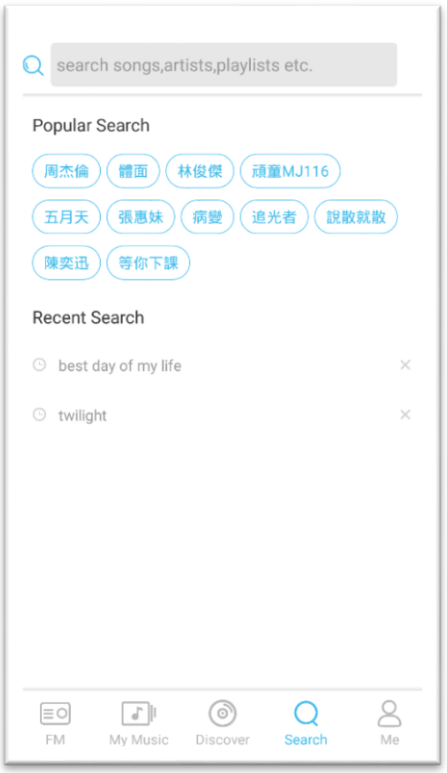


Figure 2-1-1-3 Search page of YY Music

In the playback control interface (Figure 2-1-1-4), the top is music video playback. However, it is not supported by playing the music only, but there is a "Save traffic" on the top right. Below is the playlist of songs, if users prefer to the particular music, it is allowing to download the song by clicking "Download" button which next to the song title and selecting the file format. There are two formats can be chosen, which are MP3 and MP4. The downloaded MP4 files can be viewed offline. However, the MP4 only support to be download 360p resolution, but it is enough to satisfy most of the users.

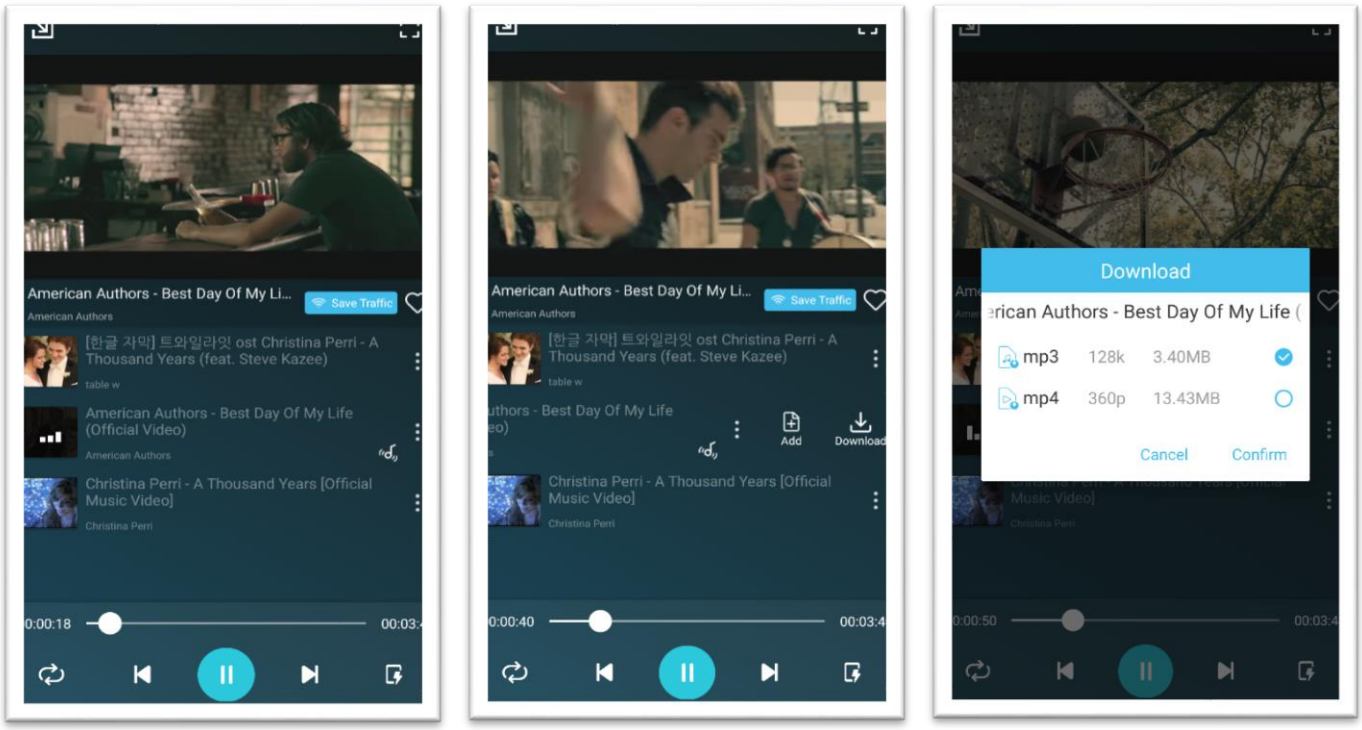


Figure 2-1-1-4 Music playback control interface and download song option

There is an option which is the 'Add' button next to the song title, it represents to add the song to the playlist created by users. In "My Music" page (Figure 2-1-1-5), users can create their own music world by adding favorite songs into the playlist. All playlists are free to create and unlimited in number. In the meantime, users can decide on the cover page of the playlist by placing the particular song on the first ordering. Unfortunately, local songs in the storage of the phone cannot be added in the playlist, only songs downloaded from YY Music can be added.

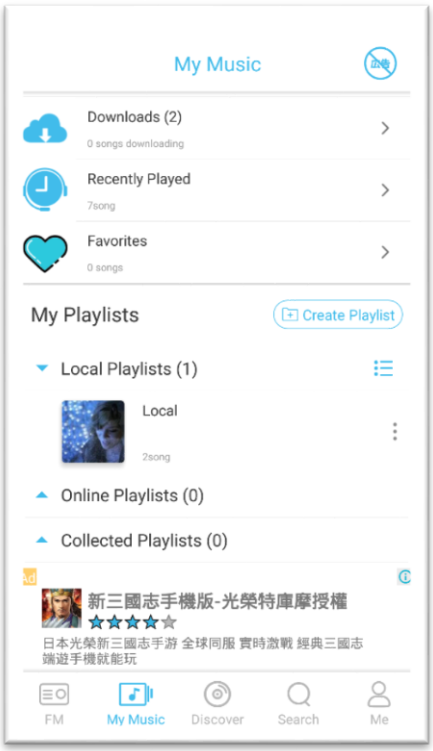


Figure 2-1-1-5 Music play control interface and download song option

Moreover, there is an FM radio (Figure 2-1-1-6) feature in YY Music. Users can directly choose channels to listen to music recommended by the system and may hear some unexpected and favorite songs here.

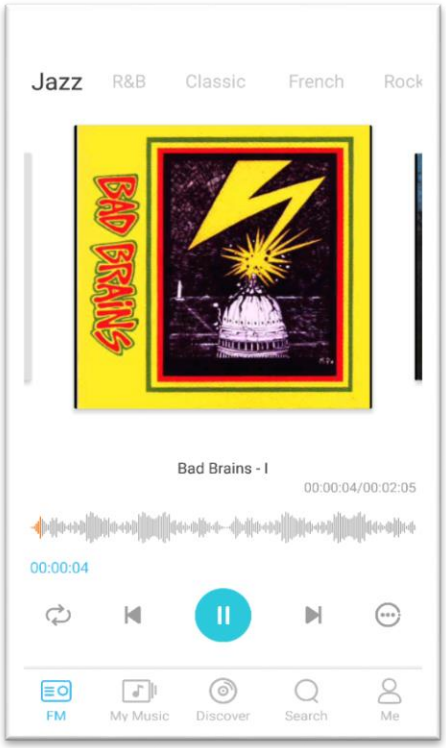


Figure 2-1-1-6 FM radio page of YY Music

In YY Music setting (Figure 2-1-1-7), users can choose whether to enable the floating window display features. The floating window is achieved by covering the top of all applications, so no matter on the desktop or running any applications, the MV screen of music will be displayed, and the position can be dragged arbitrarily.

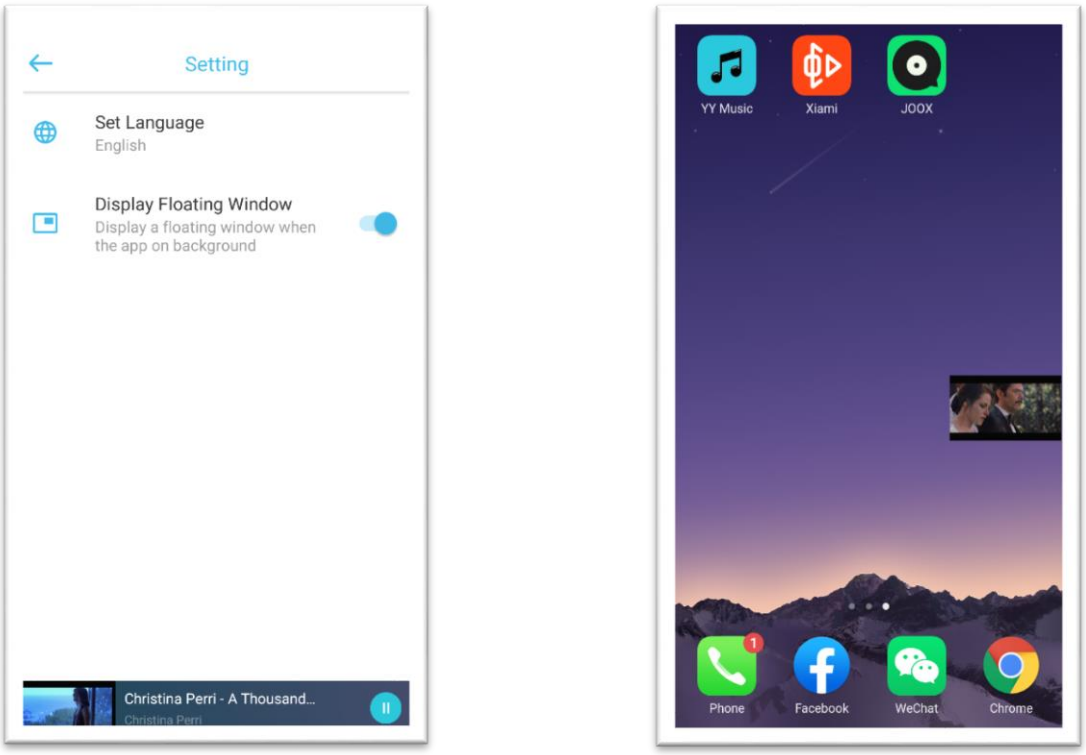


Figure 2-1-1-7 Setting page and Floating Window Feature

In addition, there is a timer feature (Figure 2-1-1-8) that allows users to fall asleep while playing their favorite songs without worrying about the huge amount of cellular data that might be consumed by continuous playing the whole night.

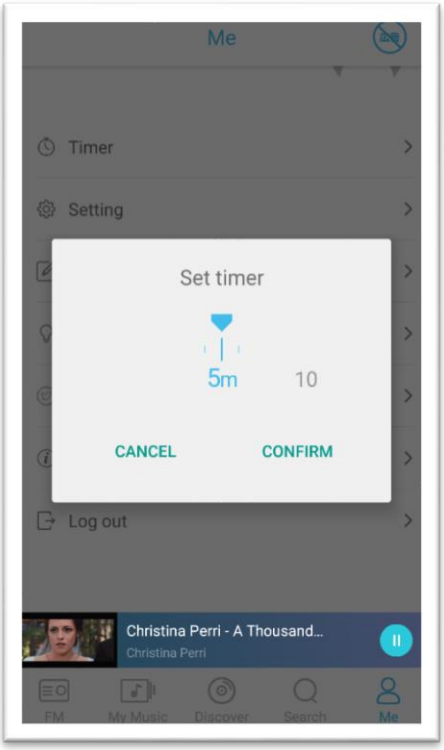


Figure 2-1-1-8 Timer feature

In conclusion, YY Music is an excellent music player that can meet many requirements of users and run smoothly. YY Music has a neat user interface, making simplicity exquisite and beautiful. However, it's a completely free music player, so there is a lot of advertisement on many pages. But it can be turn off advertisement by pressing back button on the phone, which is better than having to watch an advertisement for a mandatory 30 seconds in many applications.

2.1.2 Review on Xiami Music

Xiami music is a music player application owned by Hangzhou Alibaba Music Technology Co., Ltd. Xiami's design team is a group of people who love music and life. They started the development of Xiami music in the year 2006. In the early days, Xiami music was also called "EMUMO", meaning earn music and money. So far, Xiami music's efforts and professionalism have been recognized by a large number of users, with 14.4 million active users in China every month. In addition, Xiami has also been committed to supporting original music for a long time. It takes the lead in launching original music supporting projects in China to explore and cultivate a new generation of artists for the music industry.

Xiami music main page (Figure 2-1-2-1) is to adopt the frameless and blank-leaving design looks very comfortable.

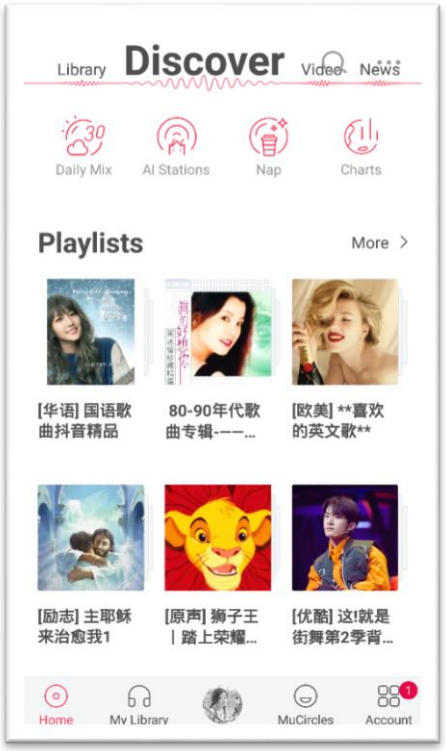


Figure 2-1-2-1 Main page of Xiami Music

However, due to the number of information on likes and comments in the music community, a large number of blank-leaving design make the visual experience lack hierarchy and increase reading difficulties. Therefore, the divider line can be used in the music community to divide each user's messages. The dividing line represents a wall, which can be used to categorize disorganized information.

Four categories are found at the top of Xiami music main page, which are a library, discover, video and news (Figure 2-1-2-2). The different genres in a music library are categorized clearly, which makes people feel the professionalism and dedication of Xiami music team. It's almost like an encyclopedia of the music genre, there are even several subcategories for each genre.

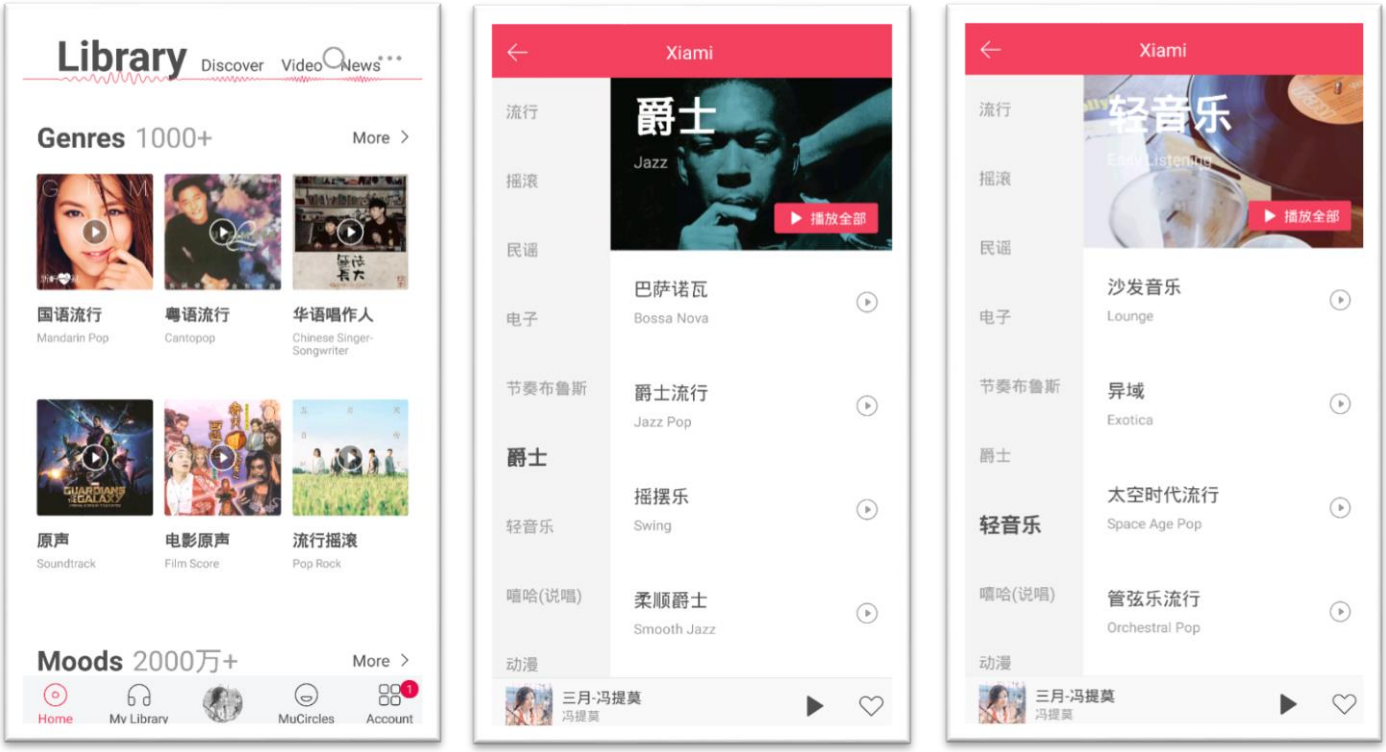


Figure 2-1-2-2 Four categories of the main page and subcategories for each genre

In addition, the discover page (Figure 2-1-2-3) is to recommend various playlists to users by system. In the discovery page, the users can be swipe down indefinitely, different content will be automatically loaded and refreshed, which makes users have an endless sense of freshness in the music world.

Figure 2-1-2-3 Discover page of Xiami Music

However, as we see that Figure 2-1-2-3 due to the recommended content occupies too much space of screen so that users need to constantly swipe down to see more content, which makes filtering content less efficient. Hence, it can be improved by narrowing the space of each content. For example, there is around two or three new content on a discovery page, so users can explore and access more music libraries, and can find what they want more quickly and conveniently.

In addition, video page (Figure 2-1-2-4) is about music video of popular artists. This is because the short video clips are popular now, therefore, it is a feature that launched to meet some users. Finally, the news page (Figure 2-1-2-4) is about some famous stories of some composers and singers as well as the latest news in the music field.

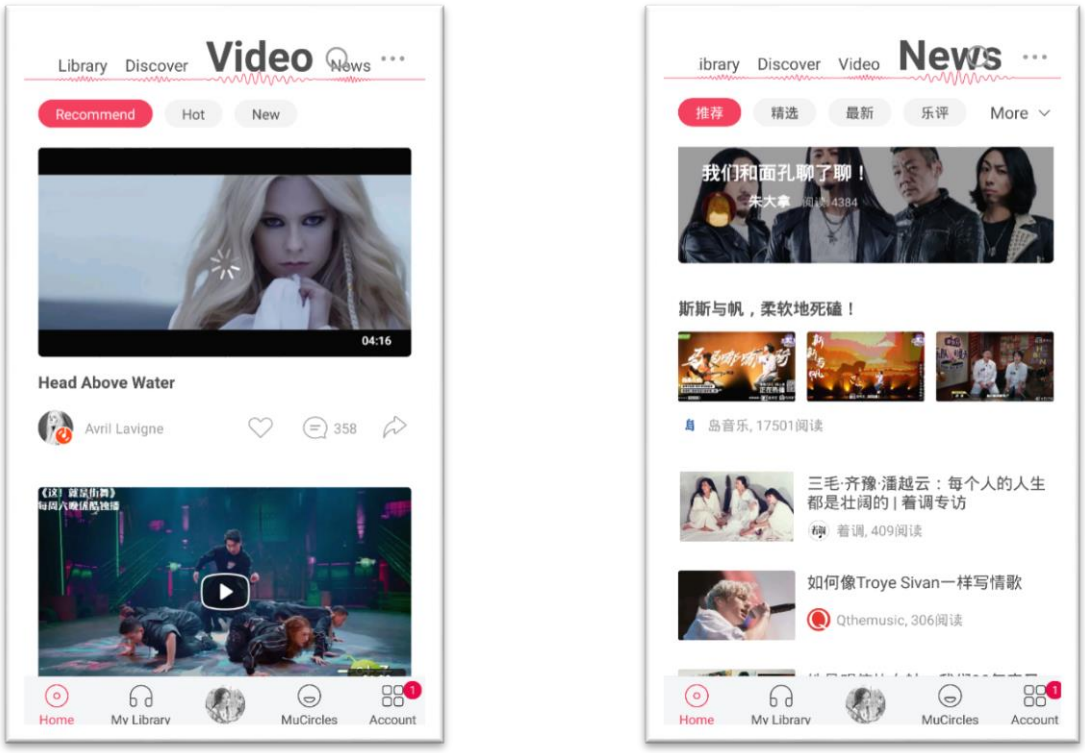


Figure 2-1-2-4 Video and News page of Xiami Music

The feature which is driving mode (Figure 2-1-2-5) and night mode in Xiami music is noteworthy. When the user enabled driving mode, the buttons in the playback control screen will become very large, which allows the driver to control the music player without distraction while driving and more convenience. In addition, there are three song lists can be chosen in this mode that is the radio station, recently played songs and the local music.

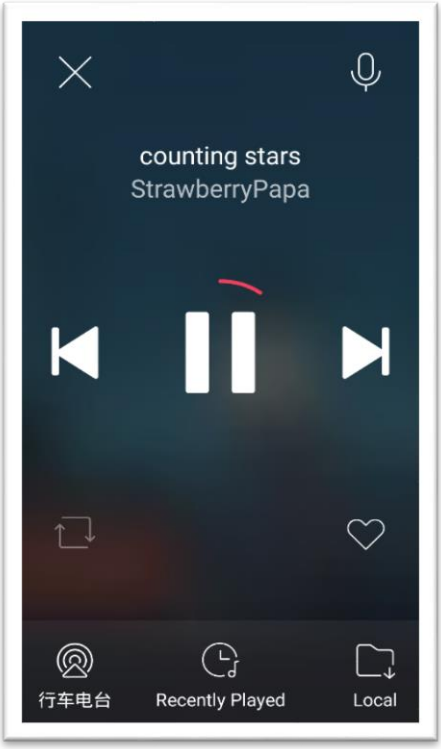


Figure 2-1-2-5 Driving Mode of Xiami Music

Moreover. night mode (Figure 2-1-2-6) is about the background color of the music player interface will be changed to black, which is to meet the bad habit of modern people still playing mobile phone after turning off the light, so that when they use the music player in the dark environment, the light in the phone will not be so dazzling.

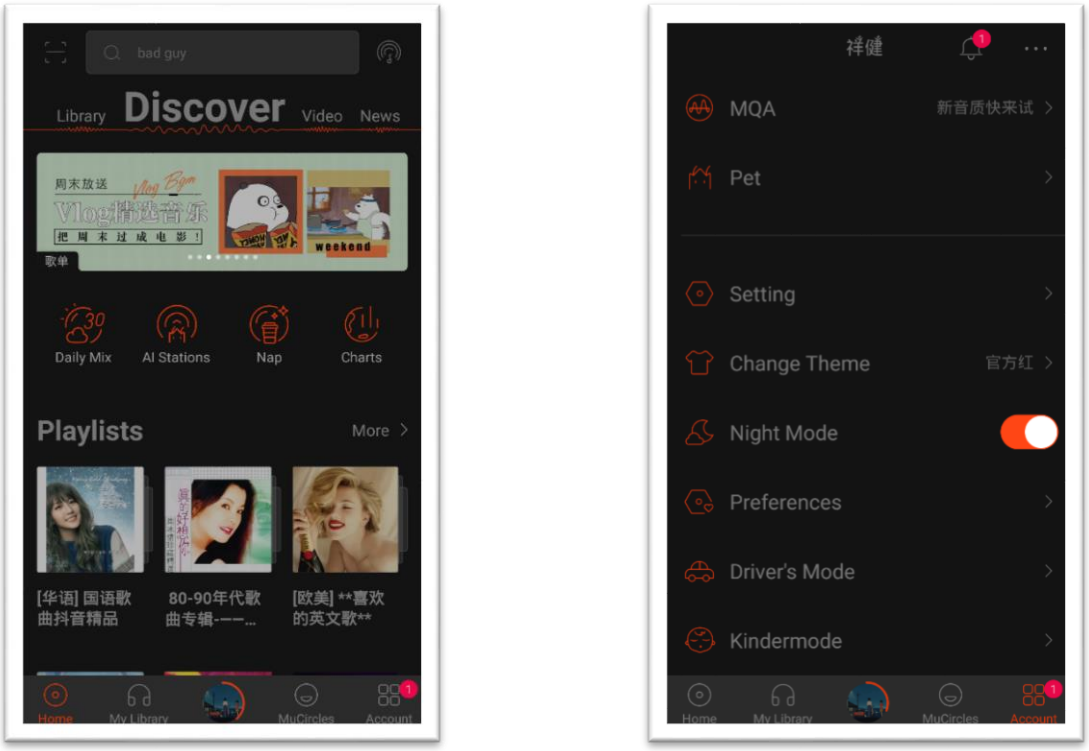


Figure 2-1-2-6 Night Mode of Xiami Music

Summary, Xiami music compared to other music apps, it is more emphasized on the experience of listening to music, like a specially prepared for those who love music. It has a neat user interface, no additional unnecessary features due to it is a music player, so listening to music is the most main features.

2.1.3 Review on JOOX Music

JOOX, owned by Tencent Mobility Limited is a streaming music service APP launched in 2015. Its main markets are Hong Kong, Malaysia, Indonesia, and Thailand, and it is the most downloads music player application across Google Play and IOS (Mulia, 2018). In addition to the existing multiple features, JOOX continuously listens to the voice and feedback of customers and launches new features to satisfy users.

The JOOX user interface (Figure 2-1-3-1) is simple and grouped into three groups which is me, discovery and live. On top of each interface, there will be a search bar, allowing users to search favorite songs at any time, has a good interactive.

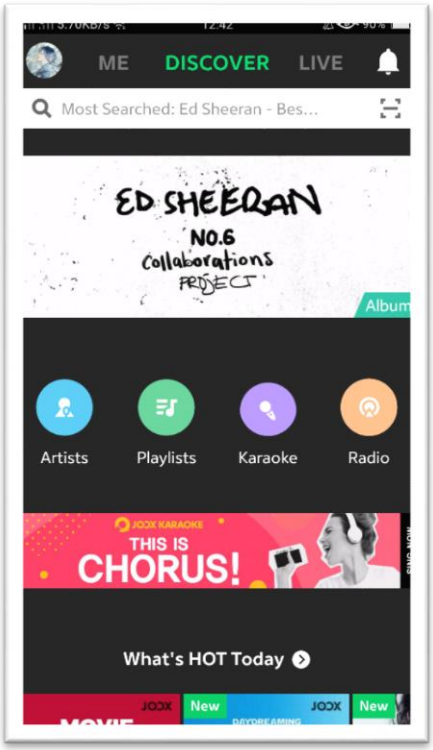


Figure 2-1-3-1 Main page of JOOX

In the "ME" page (Figure 2-1-3-2), users can create a new playlist to manage played and downloaded songs. In addition, JOOX will automatically import song on the user's device.

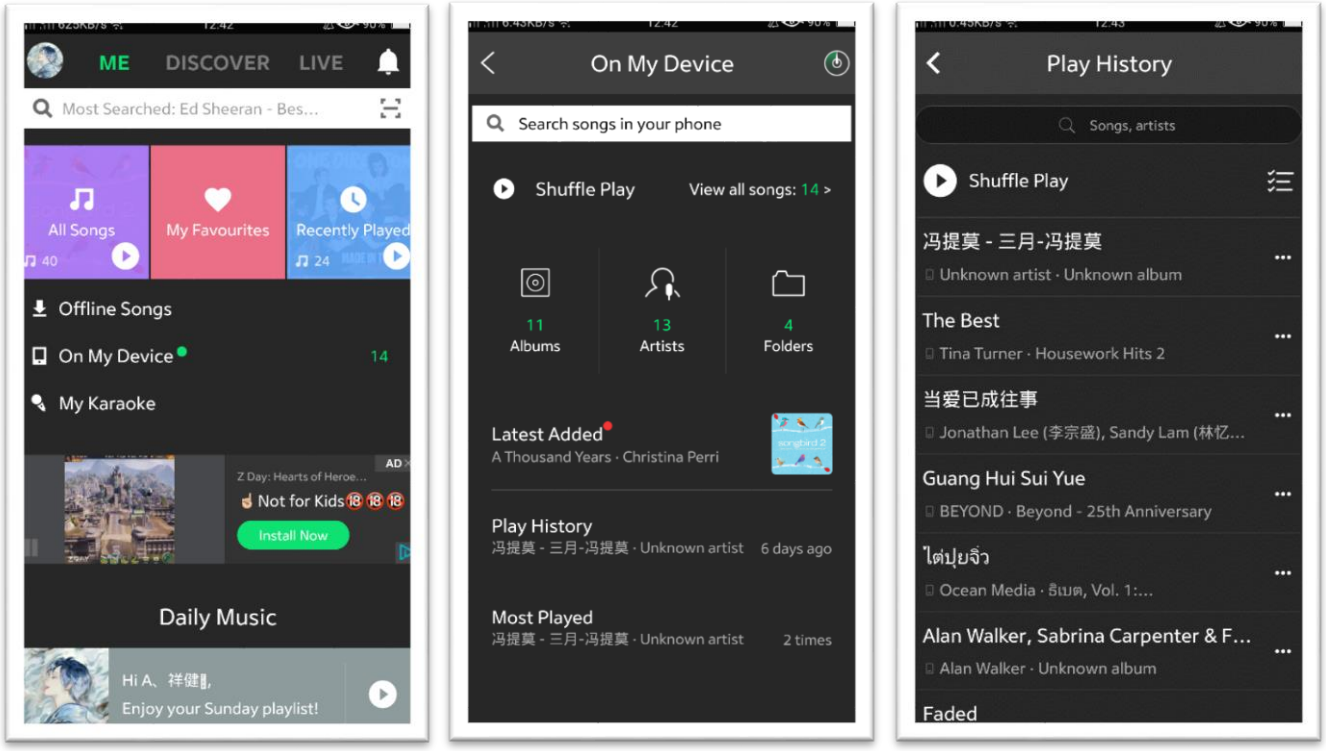


Figure 2-1-3-2 Me page and local song on the user device

Moreover, one of JOOX's features, "My Karaoke" (Figure 2-1-3-3), is a boon for karaoke fans. My karaoke features are very comprehensive after users record their nice songs, they are allowed to preview just recorded songs before saving. Also, users are allowed to edit the recorded songs such as sound offset, vocal or music volume and add effects.

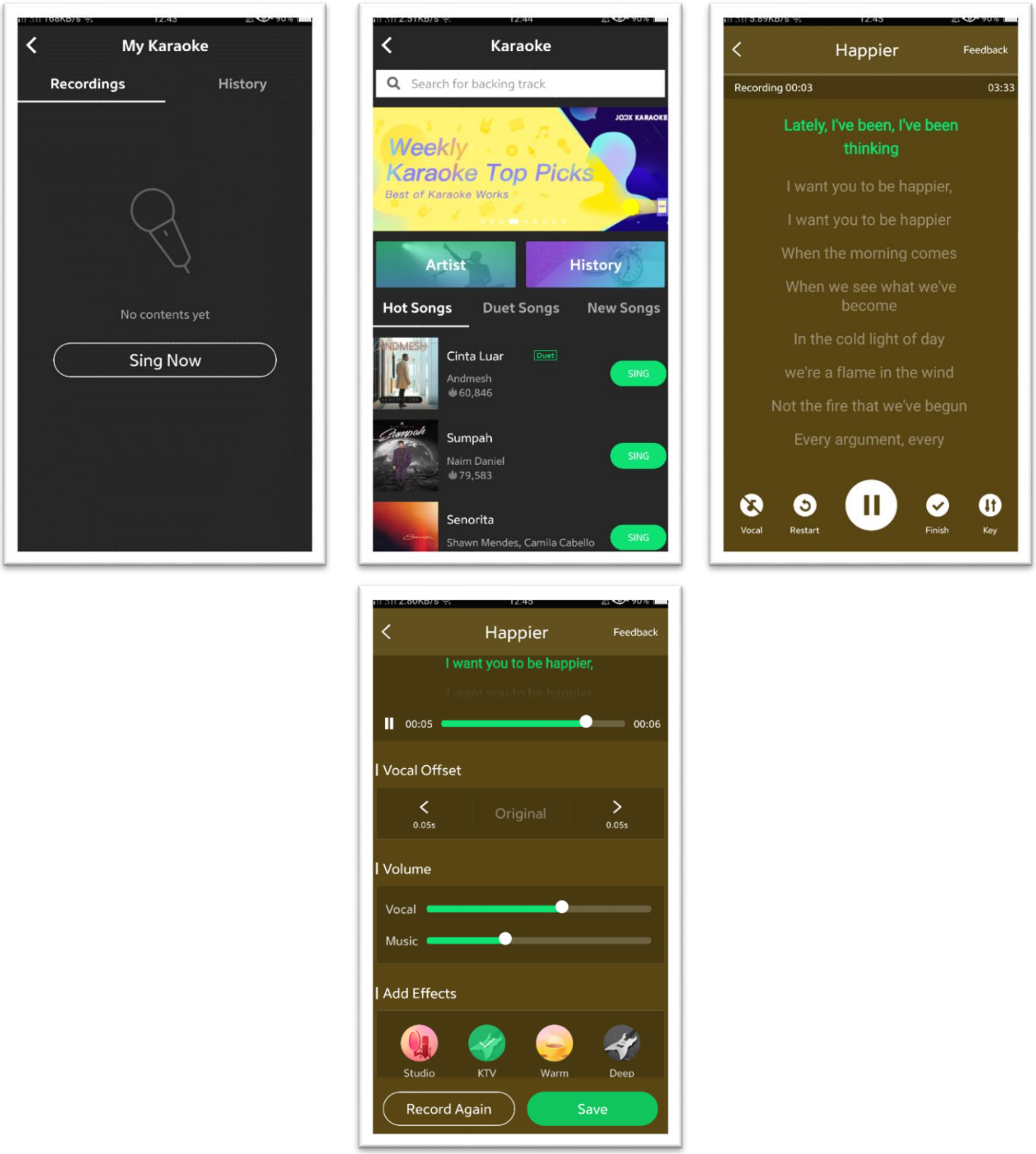


Figure 2-1-3-3 My Karaoke feature and recording editor

Next, users can search for their favorite songs through various categories on the discovery page. In the artists and playlists (Figure 2-1-3-4) under the discovery page, the categories are detailed classification and the layout is neat, just like users stay in a real karaoke room, allowing users to immerse themselves in the music world.

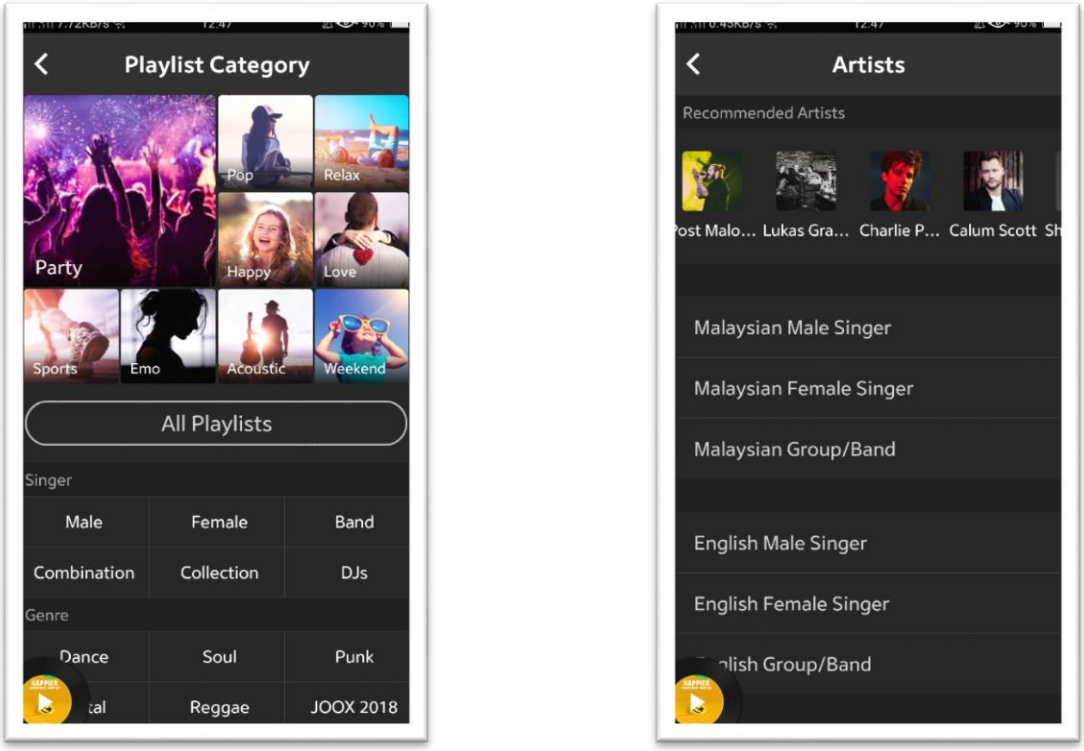


Figure 2-1-3-4 Playlist and Artists category under Discover page

In the radio page (Figure 2-1-3-5), there are around more than 50 theme stations classified according to different song styles, and users can play songs by only one click. Also, the live page is about some artist' interview, live, and their music video.

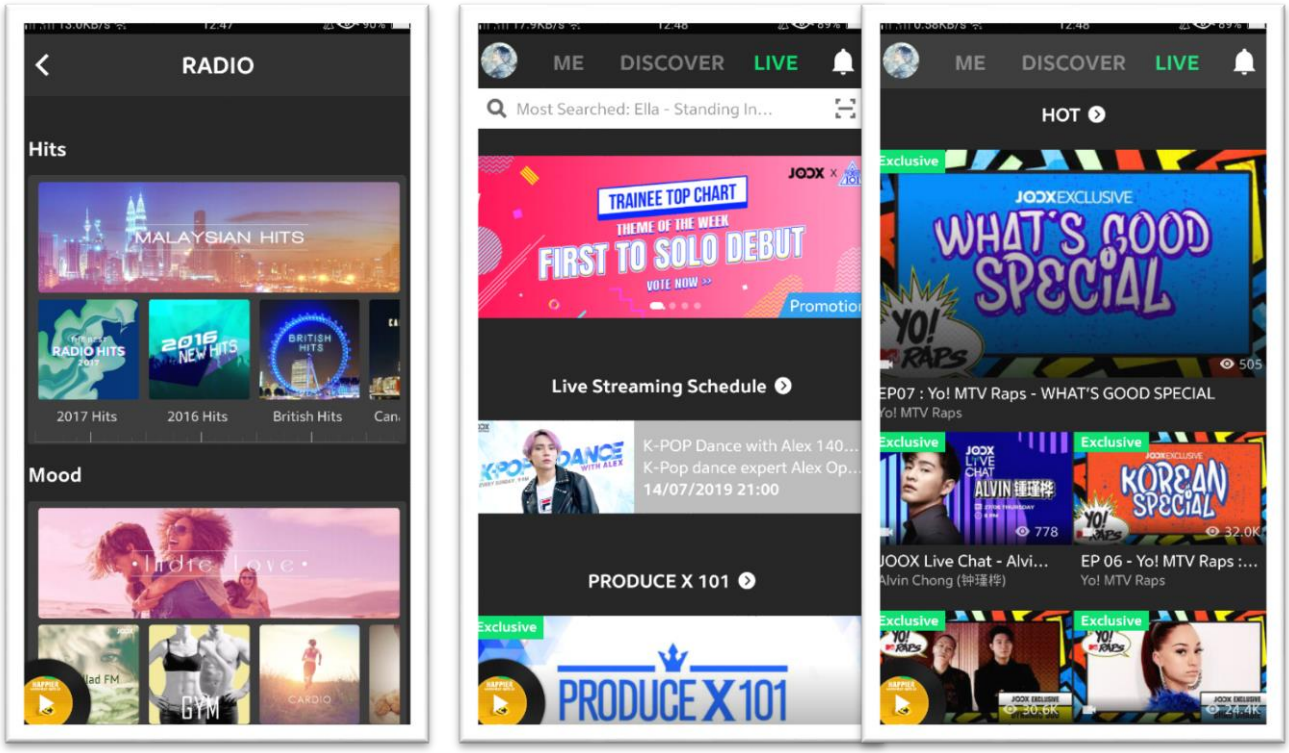


Figure 2-1-3-5 Radio and Live page of JOOX

In addition, the music playback control page with basic functions that users generally need. The layout of the interface is easy to understand and the operation is very simple, which can be mastered immediately after the first use. From Figure 2-1-3-6 show that users can share their favorite songs with their friends at any time. But it has a flaw that users can't rewind and fast-forward freely (Figure 2-1-3-7) when playing popular songs because of it is a VIP's privilege.

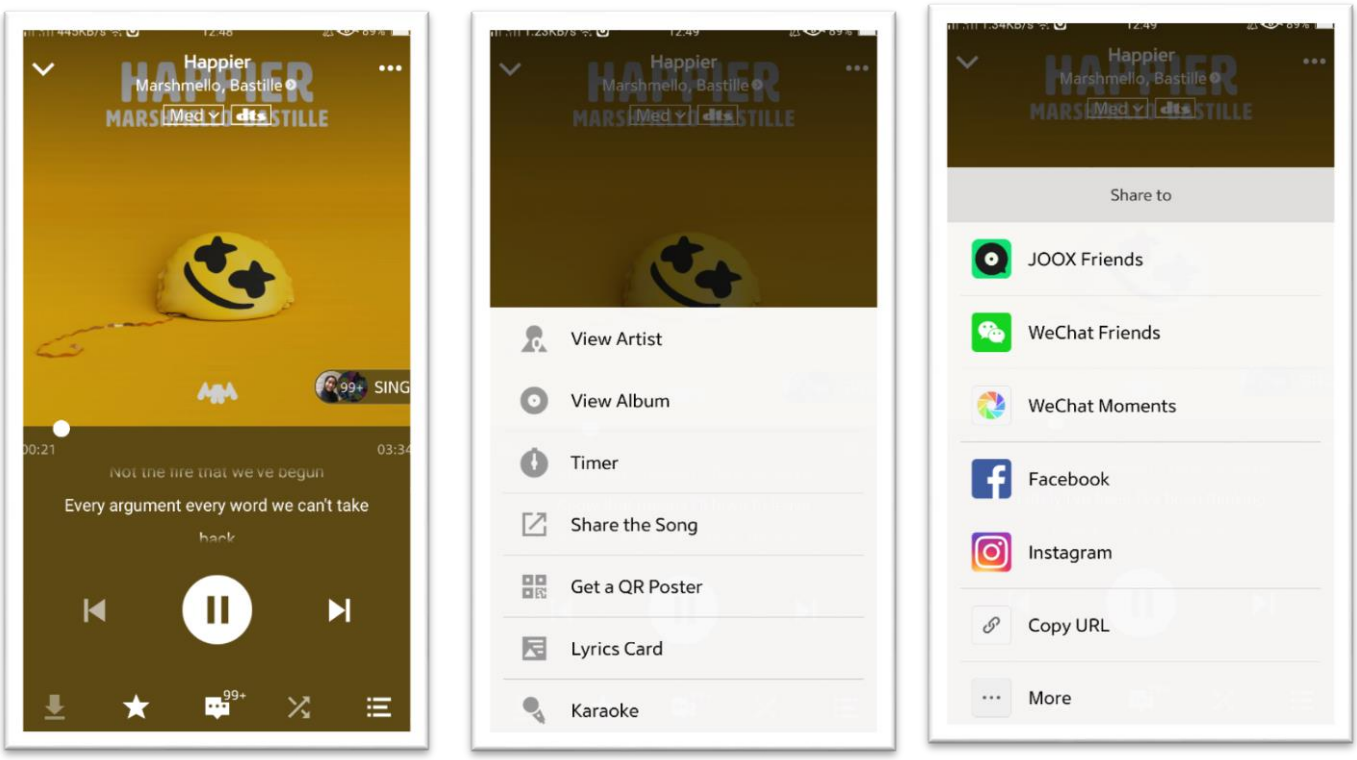


Figure 2-1-3-6 Music playback control page and sharing song features

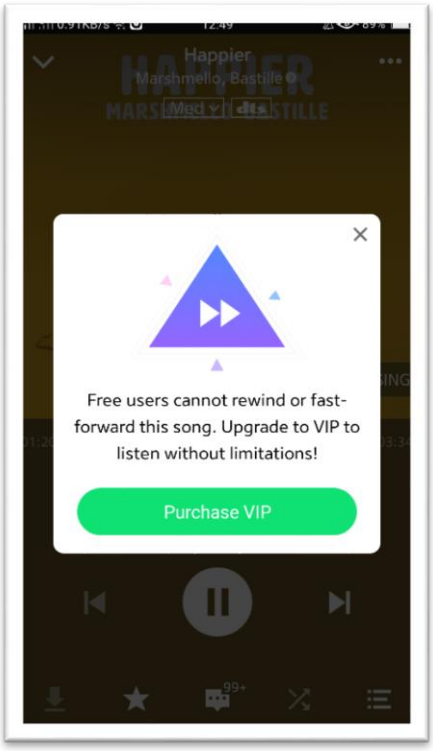


Figure 2-1-3-7 VIP privileges are required

The biggest drawback of JOOX is that many operations require to become VIP, such as downloading or listening to most of the songs. With so many apps out there that offer free downloads or listen to songs that could drive some users away. So, JOOX can introduce some tasks or mission features to relieve the limit once they have done the tasks so that allows users to gain VIP's privileges within a limited time.

To conclude, JOOX is comfortable to use, friendly in music classification, and it also allows users to change JOOX interface skin freely. It's a great paid music player.

2.2 Critical Remarks of previous works

2.2.1 Strength and Weaknesses of previous works

Application	Strength	Weaknesses
YY Music	Able to download music on YouTube	Annoying ads keep pop up during using
Xiami Music	Lightweight application and emphasize the experience of listening to music	Lack of hierarchy in some user interface
JOOX	With useful features such as My Karaoke	A large number of features require VIP privileges

Table 2-2-1-1 Strength and Weaknesses of reviewed application

2.2.2 Application Comparison

Application / Criteria	YY Music	Xiami Music	JOOX	Proposed Application
User Interface	Common	Good	Good	Good
Sorting and searching local song	Poor	Common	Good	Good
Sleep Timer	√	√	√	√
Shake phone to switch songs	X	X	X	√
Touch gesture to switch songs	X	X	X	√
Background Play	√	√	√	√
Driving Mode	X	√	X	√
Night Mode	X	√	X	√

√: Yes X: No

Table 2-2-2-1 Comparison among reviewed and proposed application

Chapter 3 System Design

3.1 Site Map

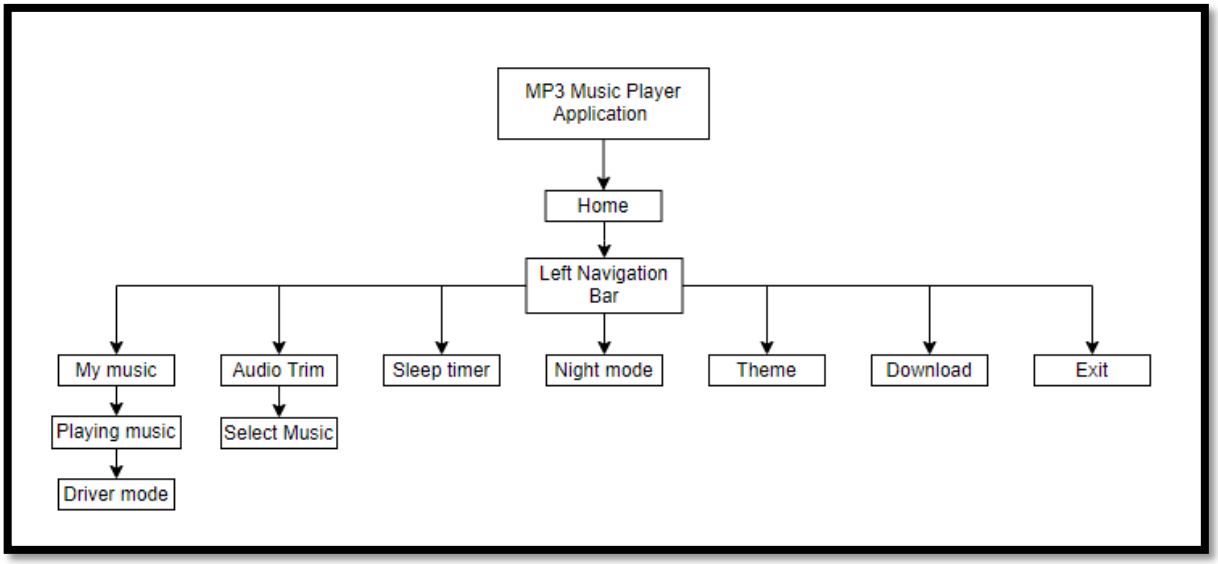


Figure 3-1-1 Site Map Diagram

The site map in this project is a method for displaying information on the layout. According to the figure 3-1-1 above, it shows that the project is consist of the left navigation bar, which leads to "My music", "Audio trim", "Sleep timer", "Night mode", "Theme", "Download" and "Exit".

Under the "My music" page as the home page, the user can select a song from the playlist to play and then turn into the “Playing music” interface. On this page, the user is allowed to switch to the driver mode to play the song.

Next, under the “Audio Trim”, the user can cut the mp3 after selecting the song file. In addition, users can change the layout into a dark color under “Night mode”.

In addition, the user is free to change the theme color of the toolbar under the “theme”. Moreover, users can download songs under the “Download” page. Lastly, users who want to terminate the application can simply click “Exit”.

3.2 Use Case Diagram

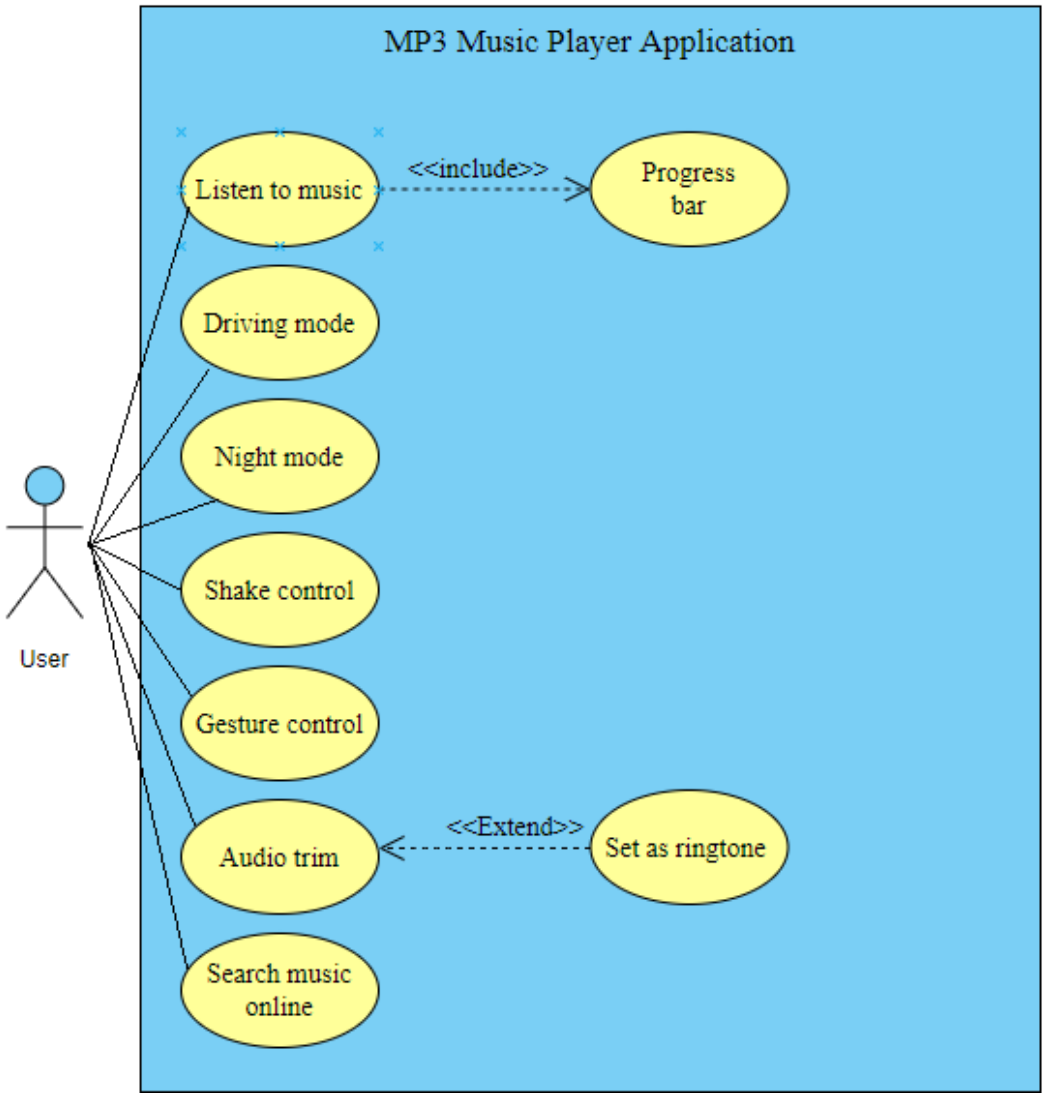


Figure 3-2-1 Use case diagram of MP3 Music Player Application

3.2.1 Use Case Description

Use case name: Listen to music	ID: UC001
Actor: User	
Description: User able listen to music on this application	
Trigger: 1. User select any song from the playlist 2. User click the play button to start playing the music	
Precondition: 1. The user should import the song from their mobile device into the music player 2. The user should add the song into a playlist by search music online feature	
Normal flow of events: 1. User select a song from the playlist 2. The song playing until the end	
Alternate / Exceptional flows: 1a. The application will display a message "No local music" if the playlist does not have any song	

Table 3-2-1-1 Use Case Description of Listen to music

Use case name: Progress bar	ID: UC002
Actor: User	
Description: The progress bar shows the progress of the song	
Trigger: 1. The progress bar becomes active when the user starts playing the song	
Precondition: 1. The song must in playing status	
Normal flow of events: 1. Song is playing 2. The progress bar will show the duration of the song 3. The progress bar will be finished if the song is finished playing	
Alternate / Exceptional flows: 1a. The progress bar will reset if user play other song 2a. If the user drags the progress bar to left or right, the song's progress will changes 3a. The progress bar will reset if finished playing	

Table 3-2-1-2 Use Case Description of Progress bar

Use case name: Driver mode	ID: UC003
Actor: User	
Description: Enables the driver to control music playback more effectively and safely while driving	
Trigger: The user click the driving mode button	
Precondition: -	
Normal flow of events: 1. The user can enable the driving mode on music playback control page 2. Select song playlist which is the local song or recently played	
Alternate / Exceptional flows: 2a. The system will display the message "No any songs to play" if the playlist is empty	

Table 3-2-1-3 Use Case Description of Driving mode

Use case name: Night mode	ID: UC004
Actor: User	
Description: Reduces user interface brightness to protect the user's eyes if they using the music player in dark environments	
Trigger: The user can enable night mode in left navigation bar	
Precondition: -	
Normal flow of events: 1. The user click the left navigation bar 2. Click the night mode button to enable it	
Alternate / Exceptional flows:-	

Table 3-2-1-4 Use Case Description of Night mode

Use case name: Shake control	ID: UC005
Actor: User	
Description: Users can switch songs by shaking their phones	
Trigger: 1. The user can shake the phone to perform media control while the song is playing	
Precondition: -	
Normal flow of events: 1. Select any song in playlist 2. Enter to song playing interface 3. Shake the phone 4. The song will switch to the next song	
Alternate / Exceptional flows:-	

Table 3-2-1-5 Use Case Description of Shake Control

Use case name: Gesture control	ID: UC006
Actor: User	
Description: The user able to use gestures to perform some action in the playback control page	
Trigger: 1. When the user swipes left or right on the screen	
Precondition: 1. The user should enter song playing page	
Normal flow of events: 1. The user enters the playing music page 2. The user swipe left or right on screen	
Alternate / Exceptional flows: 2a. When the user swipe left will switch to the previous song 2b. When the user swipe right will switch to the next song	

Table 3-2-1-6 Use Case Description of Gesture Control

Use case name: Audio trim	ID: UC007
Actor: User	
Description: Allows users to perform music trim on music application	
Trigger: 1. The user click the audio trim button in left navigation bar to perform the action	
Precondition: 1. Audio files in MP3 format must available on the user's device	
Normal flow of events: 1. Select the audio file that needs to be trim 2. Set the start and end points of the song 3. Save the song to the user's device	
Alternate / Exceptional flows: 1a. The system will display the message "Please select an audio file to trim" if a user no select any audio file	

Table 3-2-1-7 Use Case Description of Audio trim

Use case name: Set as ringtone	ID: UC008
Actor: User	
Description: Allow user to set the trim audio as a ringtone or alarm	
Trigger: 1. There will be options for users to choose set as an alarm or ringtone after audio trim	
Precondition: 1. Use audio trim feature	
Normal flow of events: -	
Alternate / Exceptional flows: -	

Table 3-2-1-8 Use Case Description of Set as ringtone or alarm

Use case name: Search music online	ID: UC009
Actor: User	
Description: Allows users to search for the song from YouTube and download it into a mobile device	
Trigger: 1. The user click the Download icon button under left navigation bar	
Precondition: -	
Normal flow of events: 1. Click the left navigation bar 2. Select “Download”	
Alternate / Exceptional flows: 2a. The user can add the song into playlist 2b. The user can download the song into device	

Table 3-2-1-9 Use Case Description of Search music online

3.3 Activity Diagram

3.3.1 Listen to music

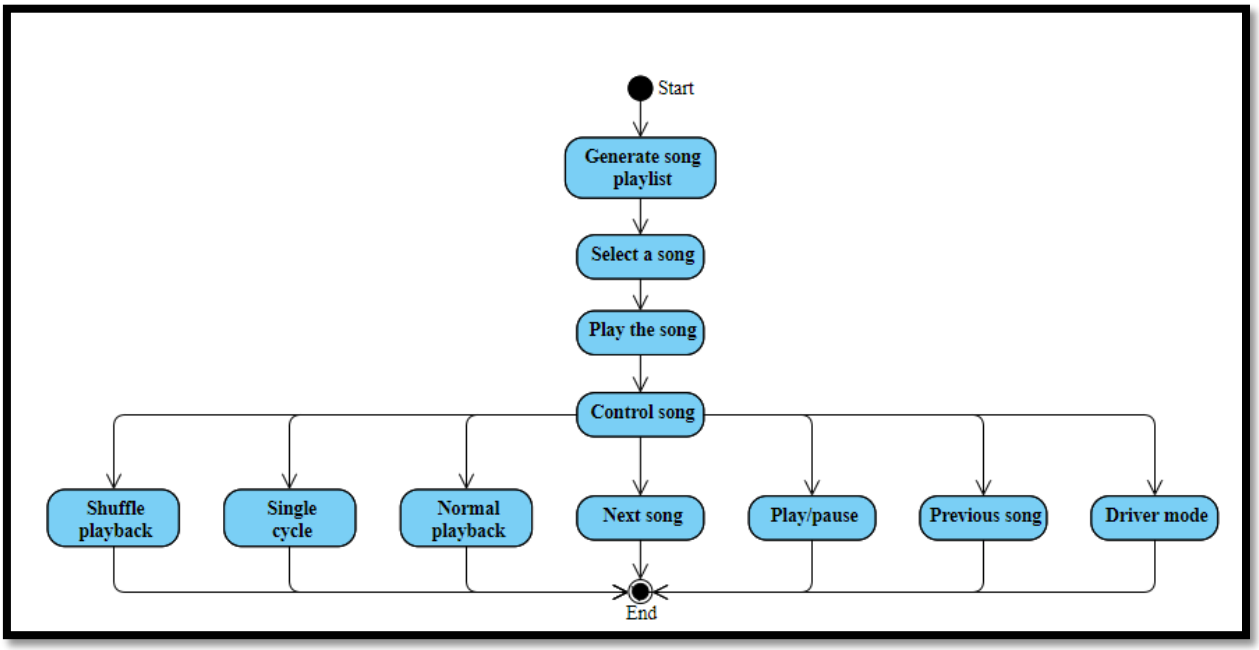


Figure 3-3-1-1 Activity Diagram for User to Listen to Music

The user starts the application and then it will go to the home page which is the "My music" page. The application will read the device's local songs and generate a playlist. After that, the user selects the song and plays it, the page will jump into the playing music interface, where the user is allowed to control the music. Playing modes are allowed to control music play orders which are shuffle playback, single cycle, and normal playback. Moreover, the user can also skip to the next song, back to the previous song and play or pause the current music to control music. Lastly, the user can enable the driver mode to play the song.

3.3.2 Audio Trim

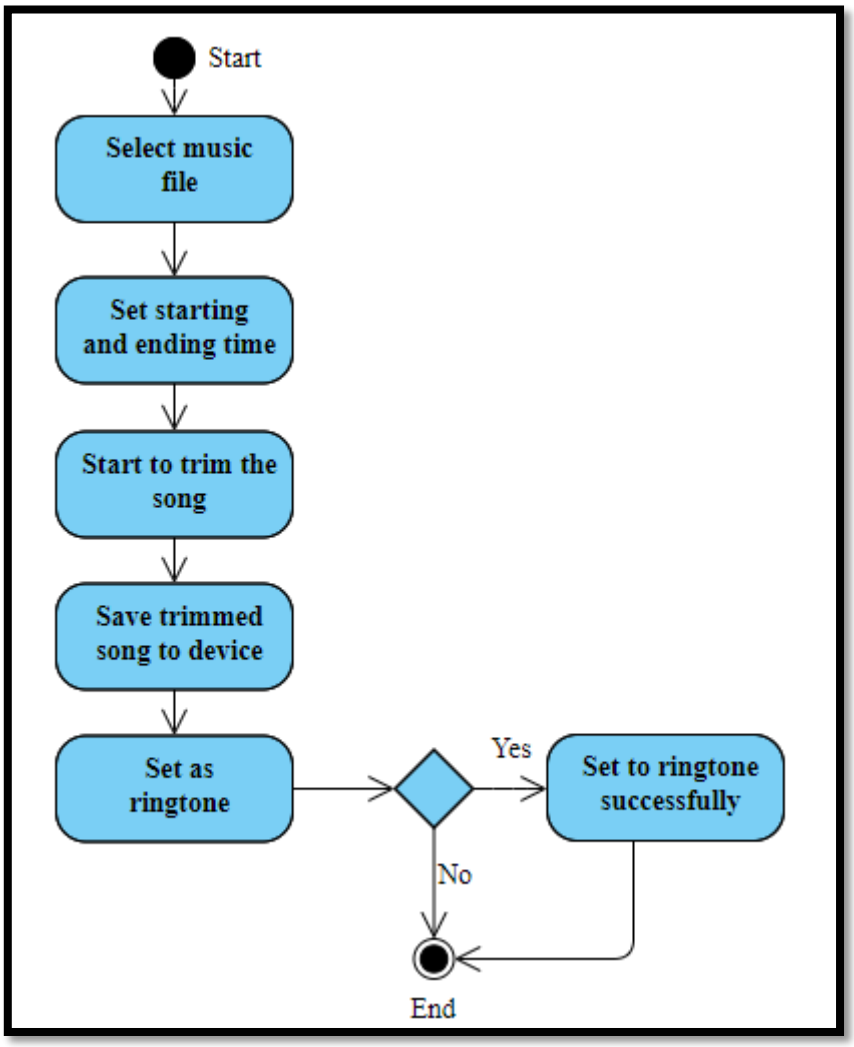


Figure 3-3-2-1 Activity Diagram for User Using Audio Trim Feature

In order to cut the MP3 song, the user needs to select an MP3 song to be trim after entering the audio trim page. After that, the user can drag the progress bar to set the start and end times. A dialog box will pop up after successfully trimmed the song to ask the user whether to set the trimmed song as the ringtone. If the user agrees, the app will successfully set the ringtone and end the activity. If users don't want to be set as a ringtone, they can click on "No" to end the activity.

3.3.3 Sleep timer

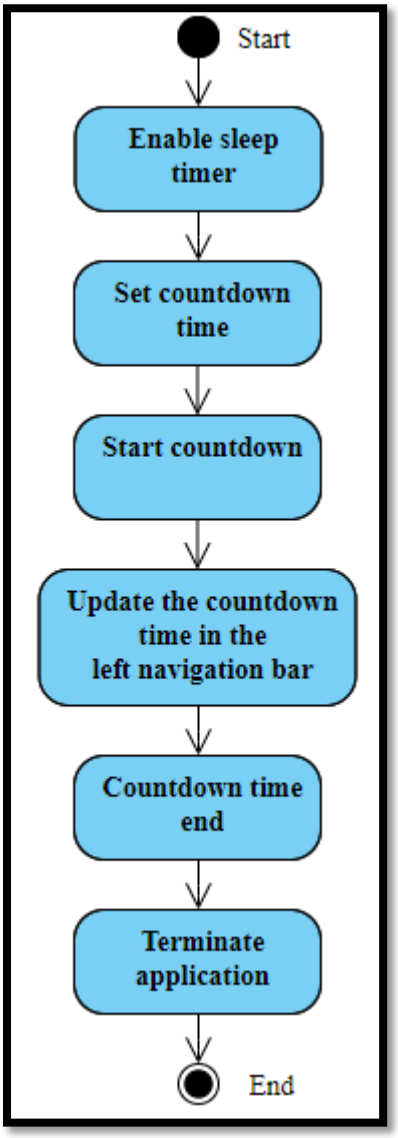


Figure 3-3-3-1 Activity Diagram for User Enable Sleep Timer

Users can start it by clicking on the "Sleep timer" in the left navigation bar. After that, the user should set the countdown time, and the countdown will begin after it enabled. The countdown timer will be updated in real-time, the user can check the remaining time in the left navigation bar. When the countdown timer is over, the application will terminate.

3.3.4 Night mode

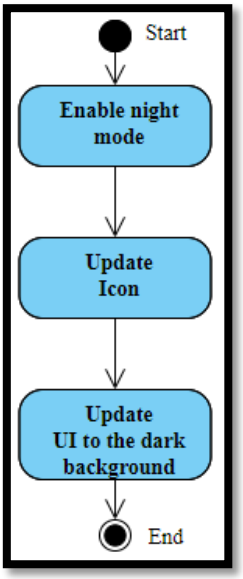


Figure 3-3-4-1 Activity Diagram for User Enable Night Mode

Users can enable night mode by clicking the "Night Mode" button in the left navigation bar. When it is enabled, the moon icon will change to a sun icon, and the layout of each page will update to dark color background.

3.3.5 Change theme

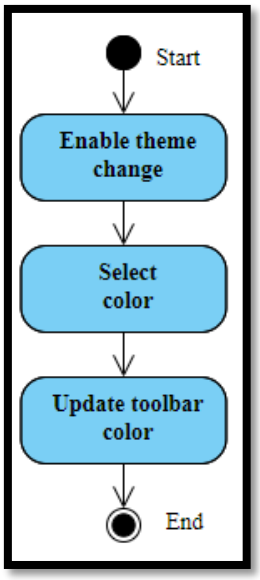


Figure 3-3-5-1 Activity Diagram for User to Change Theme of Tool Bar

In order to change the theme color of the toolbar, the user can click "Theme" in the left navigation bar. At this movement, a theme setting dialog box will pop up, which provides eight colors for users to select. The selected color will be updated to the background of the toolbar.

3.3.6 Download song

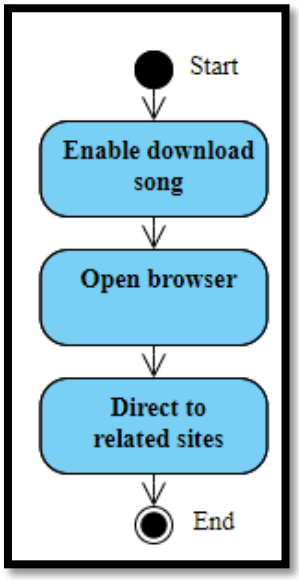


Figure 3-3-6-1 Activity Diagram for User to Download Song

Users can download songs by clicking "Download" in the left navigation bar. The application will open the mobile browser and direct it to a website that downloads MP3 songs. Users can search the song by entering song, album and artist names.

3.4 System Wireframe

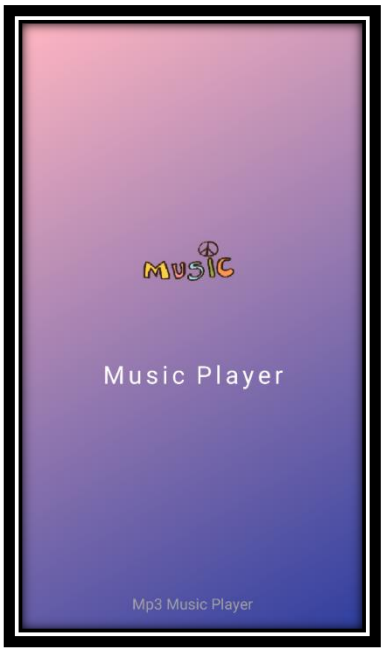


Figure 3-4-1 Splash screen

Figure 3-4-2 Home page

Once the user starts the application, the first screen will be a splash screen (Figure 3-4-1) to show out. Then, figure 3-4-2 which is the home page, set to "My music" by default, reads songs from the local device and generates a playlist.

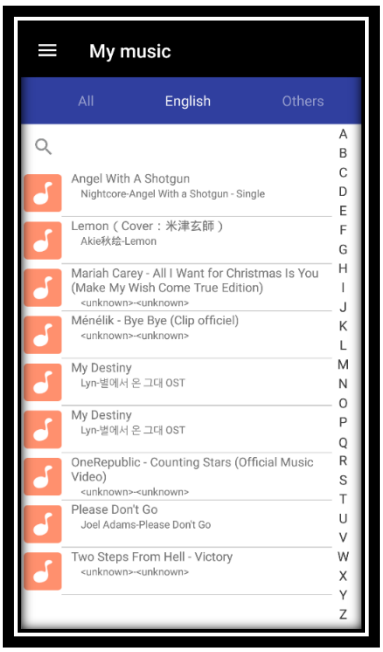


Figure 3-4-3 English Song Playlist

Figure 3-4-4 Others Song Playlist

The song playlists are divided into three categories which are the first is all songs, the second is English songs, and the third is other songs which including Japanese, Chinese and so on. Song playlists are primarily categorized by the first word of the song name.

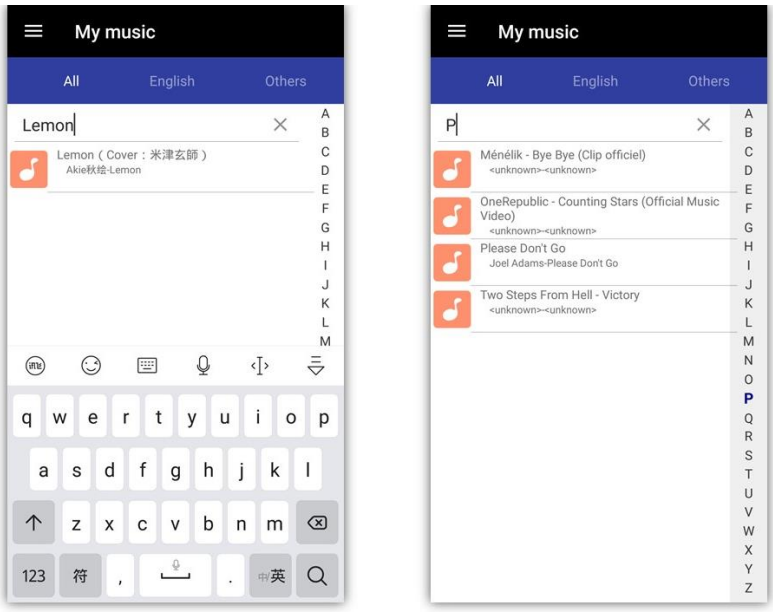


Figure 3-4-5 Filtering Song Using the Search Bar and Alphabet Quick Scrollbar

From figure 3-4-5, the user can quickly find the song by entering the song name in the search bar at the top. In addition, when the right side alphabet quick scrollbar is being clicked, the top search bar also will be updated. For example, if the user drags to the 'P' character, the search bar at the top will also be updated to 'P'. In fact, filtering song is according to the song name, as long as the song name contains the user input information, then the result will be displayed.

Figure 3-4-6 Song Playing Page

As the song begins to play, the disk in the center of the screen will begin to rotate and the button on bottom center will be updated to pause button. The pause button will be updated to the play button if the song stops playing. In addition, the user can control the progress of the song by dragging the progress bar.

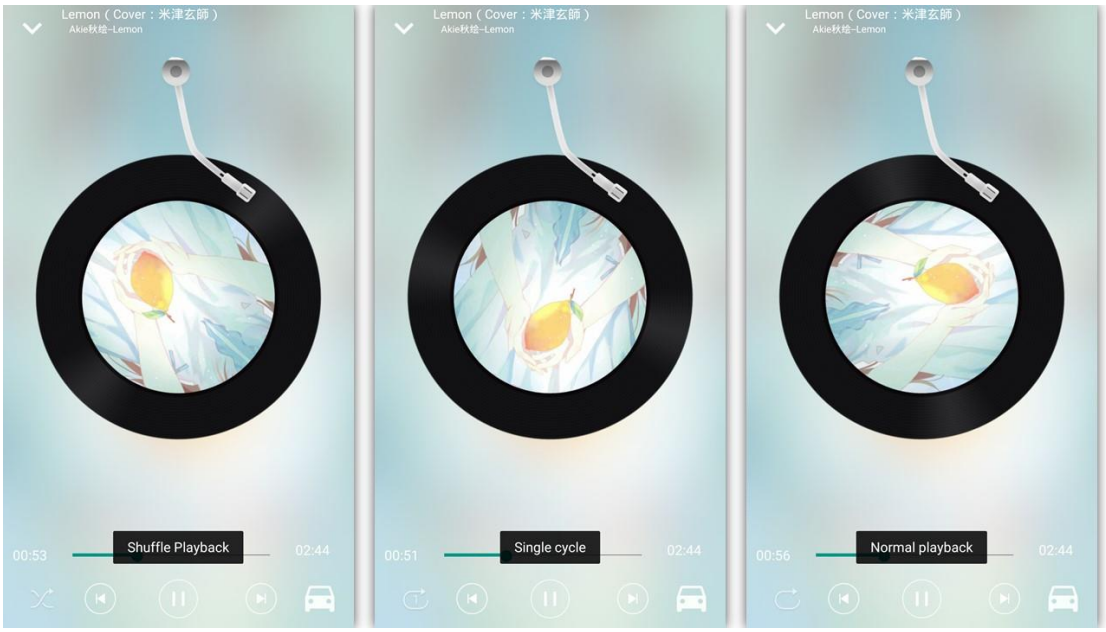


Figure 3-4-7 Song Playback Mode

The user can switch the song playback mode by clicking the button at the bottom left. There are three modes of playback. The first one is the shuffle playback. When the song is completed to play, the next song will be selected randomly from the playlist. The second is a single cycle, where the current song is completed to play and the same song will be playing again. The third is the normal playback. It means that the application will continue to play the songs in the playlist in sequence until the user moves on to the next action.

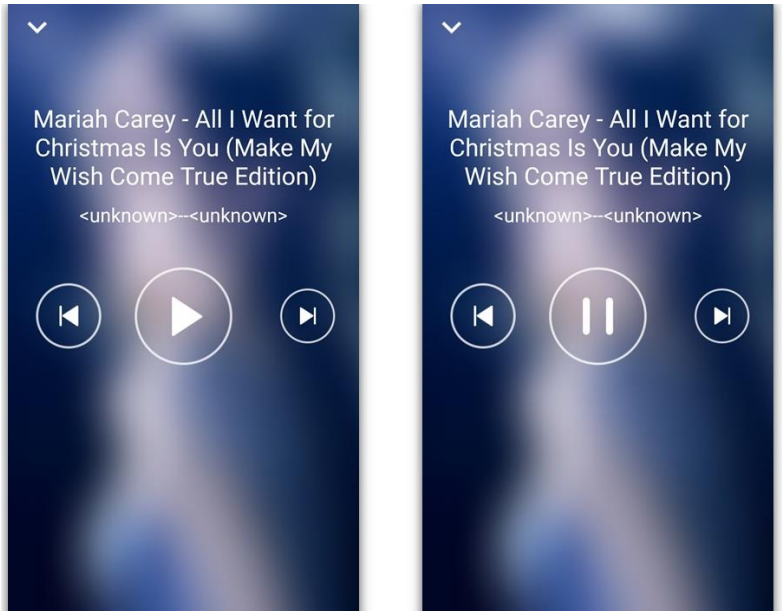


Figure 3-4-8 Driver Mode

Users can click the car icon at the bottom right side in Figure 3-4-7 to enable the driver playing mode, where all buttons and size of song names will be enlarged, the output will be like Figure 3-4-8.

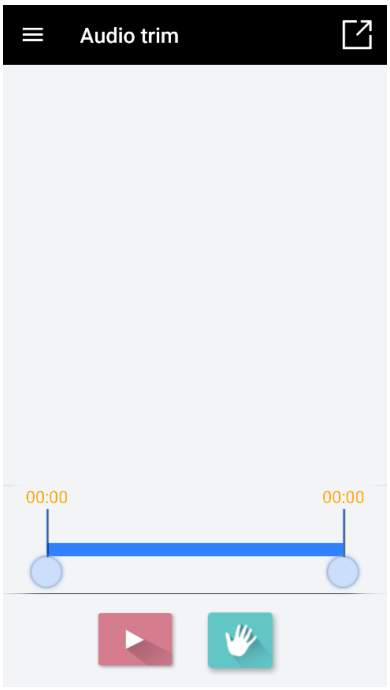


Figure 3-4-9 Audio Trim Feature

This is the audio trim page, the user should click on the top right side button to select music files in order to cut the mp3 song.

Figure 3-4-10 Select Music Page

The user should select a song on this list to trim the song. If there is no user desire song on the list, the user can click the button on the bottom right side in order to scan the local device's song to update the list.

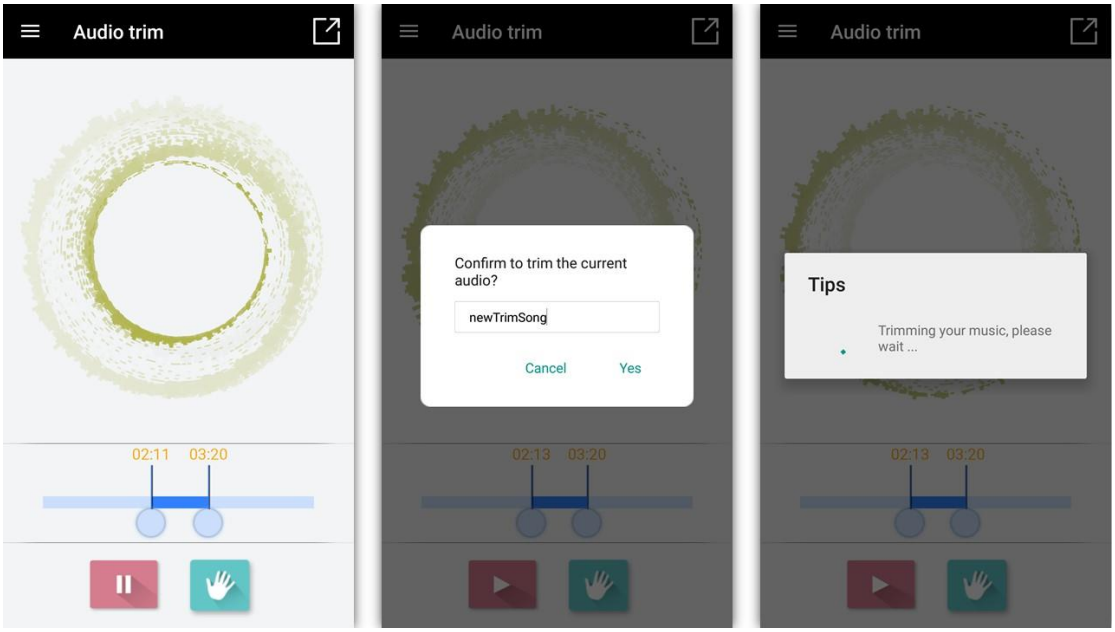


Figure 3-4-11 The Process of Cut Song

The user can set the start time by dragging the left progress bar and the end time by dragging the right progress bar. After setting the start and end time, click on the green background color icon. At this time, a dialog box will pop up asking the user to enter the new name of the trimmed song. Click "Yes" to start the trim, and "No" to cancel the action.

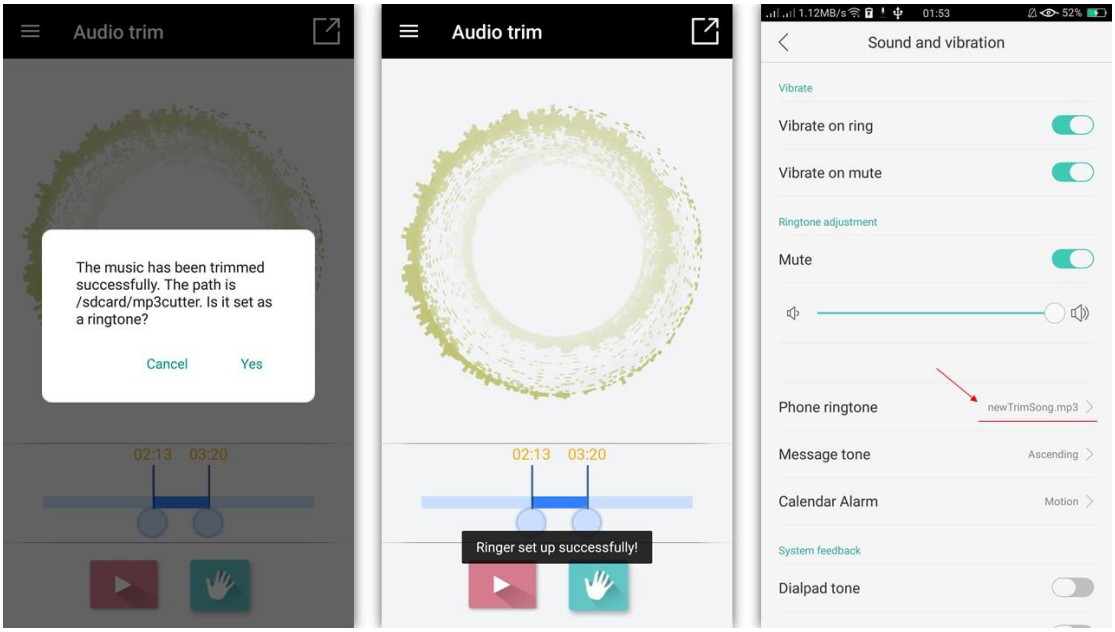


Figure 3-4-12 Set the Trimmed Song as Ringtone

A dialog box will pop up after trimmed successfully in order to ask the user if they want to set it as a ringtone. Click "Yes" and the toast message will pop up to show that the set-up is successful. The user can check whether it has been successfully set as the ringtone in the phone settings. Click "Cancel" to return and cancel action.

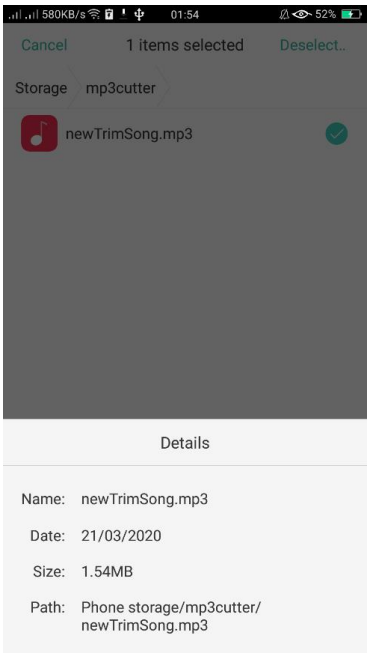


Figure 3-4-13 Generated Trimmed Song

Users can view the new MP3 trimmed songs in the relevant path after trimmed.

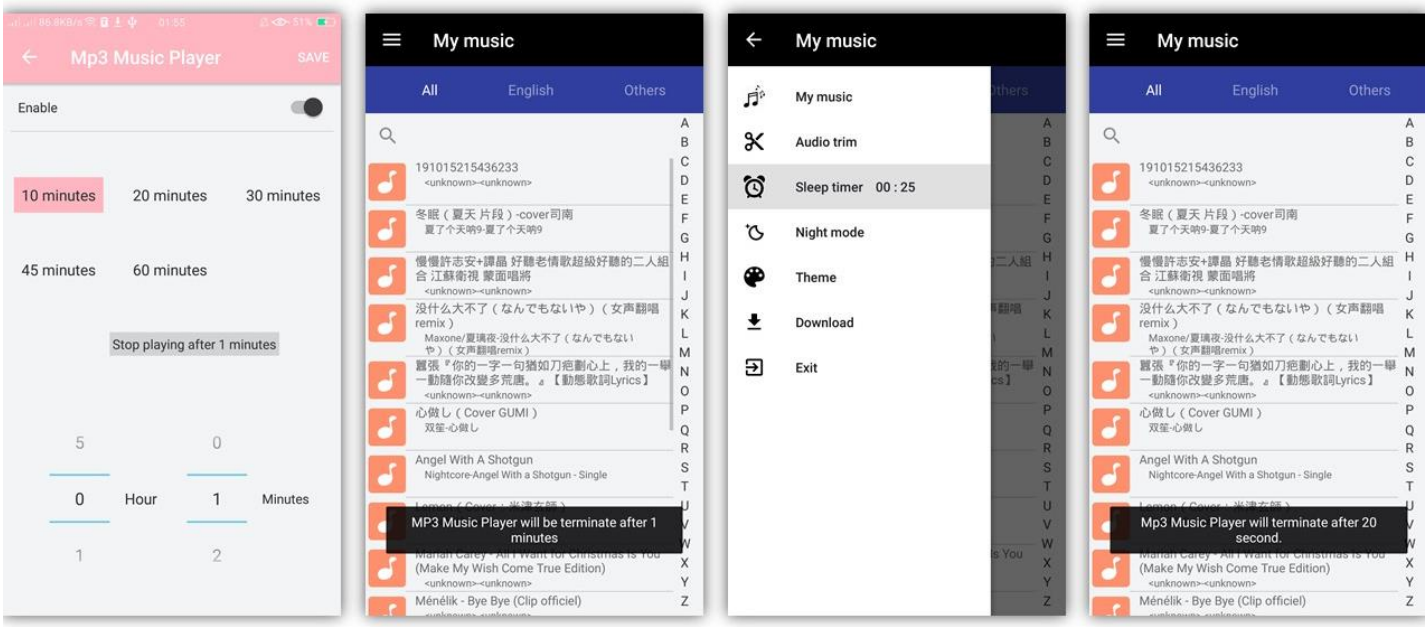


Figure 3-4-14 Enable Sleep Timer

In order to enable sleep timer, users can click on the left navigation bar and select "sleep timer". In the sleep timer setting, the user can start to set the countdown time after clicking enable. The countdown times provide some option to let the user choose which are 10 minutes, 20 minutes, 30 minutes, 45 minutes, and 60 minutes. Also, users are allowed to customize the countdown times. After setting the countdown time, the user will return to the previous page and a toast message will pop up to let the user know that the sleep timer is enabled and set up successfully. In addition, users can click on the left navigation bar to see how much countdown time is left. When the countdown time left 20 seconds, the toast message will pop up again to inform the user that the app will be closed after 20 seconds.

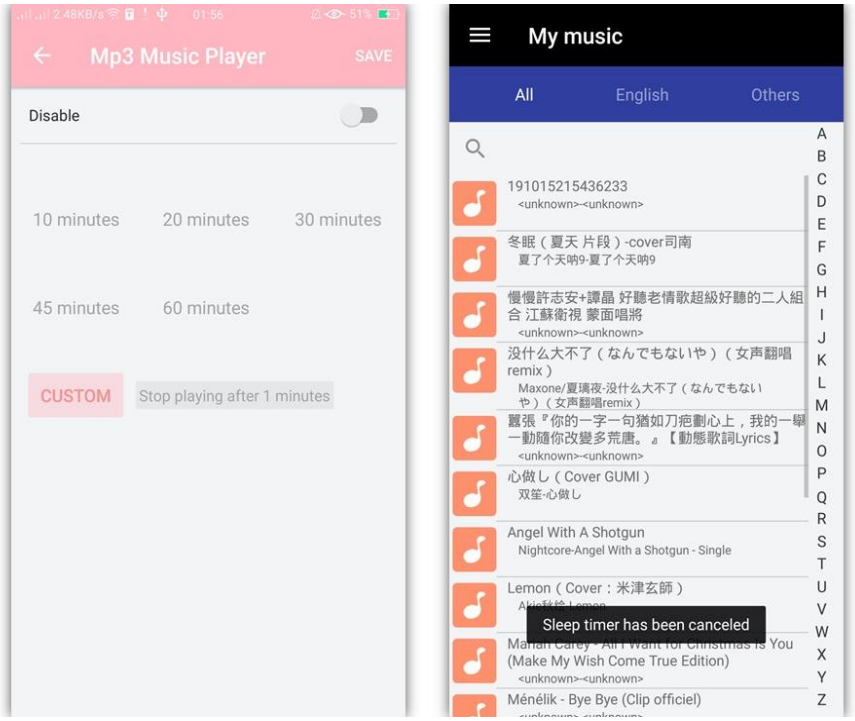


Figure 3-4-15 Disable Sleep Timer

Moreover, users are allowed to disable the feature after they set up the sleep timer. Users can simply go back to the sleep timer settings, drag to the left and click the "Save" button on the upper right side then successfully cancel the sleep timer.

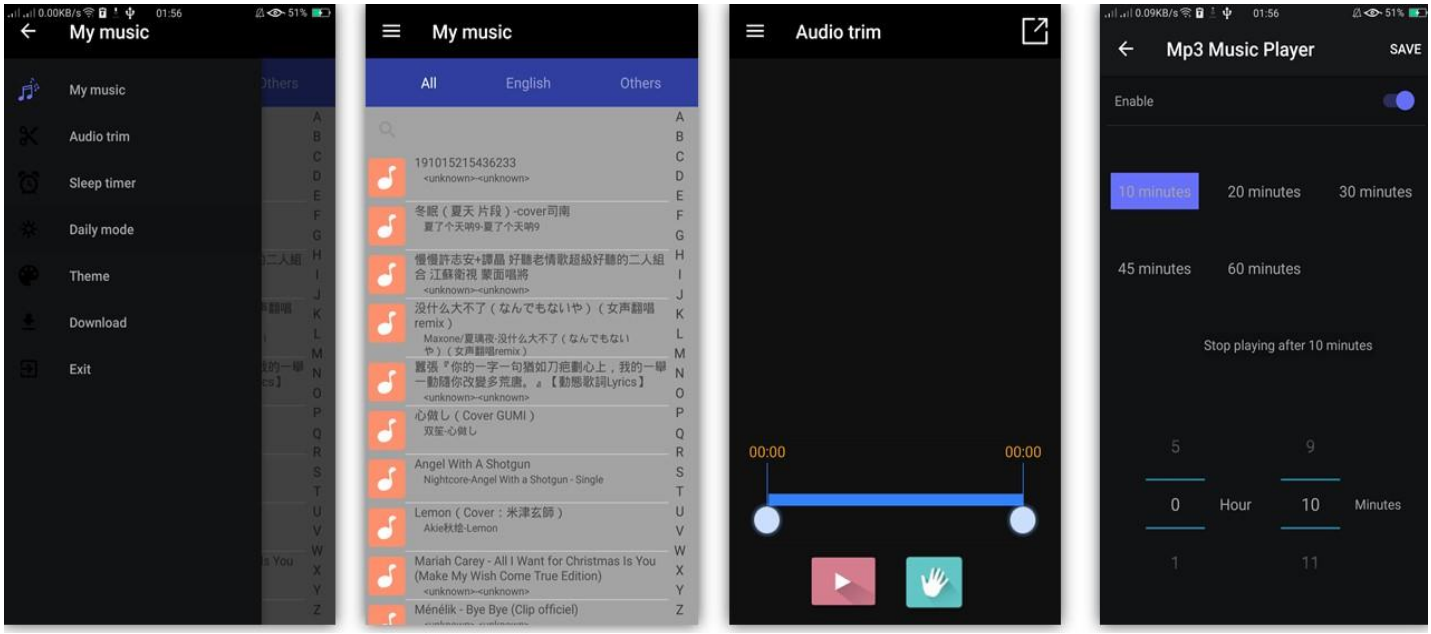


Figure 3-4-16 Enable Night Mode

Night mode can be enabled from the left navigation bar by clicking on the moon icon which also called "Night mode". When enabled, the moon icon will update to the sun icon and change its name to "Daily mode", then the user can click again to cancel the night mode. When night mode is enabled, the background color of the layout in "My music", "Audio trim" and sleep timer settings will be changed to dark color.

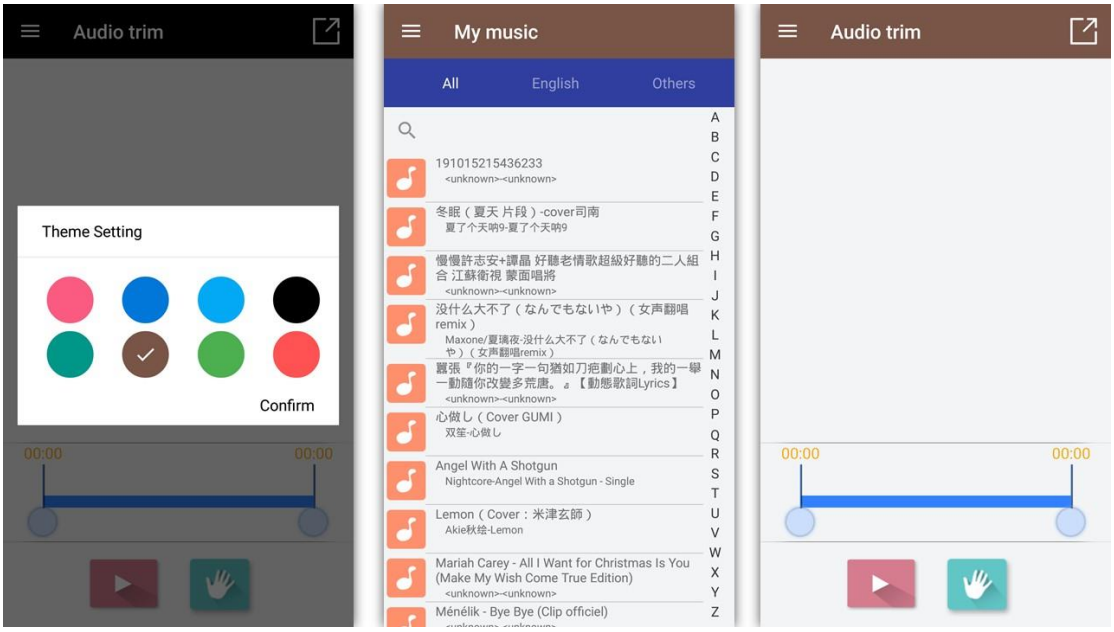


Figure 3-4-17 Change Theme Color of Toolbar

There are eight types of theme colors for users to select. After the user changes the theme color, the theme on "My music" and "Audio trim" will be updated to the selected color.

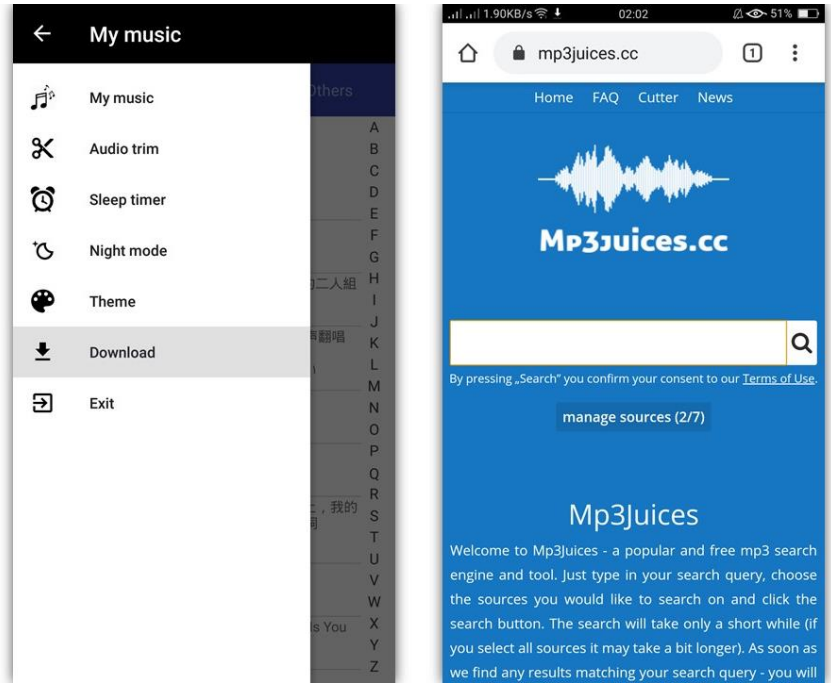


Figure 3-4-18 Download Song

Users can download songs by clicking "Download" in the left navigation bar, but this is not an in-app download. It just provides a hyperlink that allows users to download songs by direct to this specific URL

Chapter 4 Methodology and Tools

4.1 Design Specifications

4.1.1 Methodology

In this project, the agile development cycle will be used to guide the development process. The reason for using agile methods is that mobile applications have a short software life cycle and rapidly changing technologies, so users will constantly change their requirement and needs in response to technological changes. Therefore, the agile development cycle are more suitable for android application development because of iterative and flexible, so it can adapt effectively to changing customers.

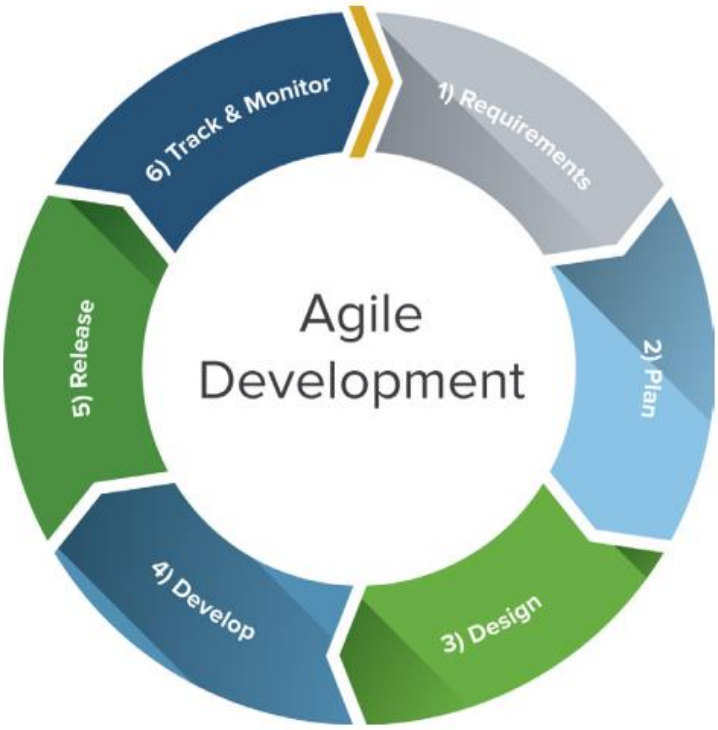


Figure 4-1-1-1 Agile Development Cycle (Devcrew.io, 2017)

The agile development cycle contains 6 phrase which is requirement analysis, planning, design, implementation or development, testing, and deployment.

Requirement analysis

At this stage, we will review existing MP3 music players on the market. After the review, we will find out what current users need and idea to improve the existing music players and collect their comments and suggestions for further analysis.

Planning

In the planning stage, we should first try to explore out the features that the music player can have. Next, we will eliminate the features that users feel no really useful or low cost-effective. Finally, each feature is prioritized and assigned to an iteration.

Design

The design stage is prepared according to the requirements of users. Since there are many details and problems encountered during development to be considered for each feature. Therefore, we will discuss and formulate solutions and test strategies to verify the product at this stage.

Implementation or Development

During the development phase, we will iteratively implement each of the features listed during the planning phase. At this stage, there will be many setbacks and obstacle, so the team needs to constantly overcome these obstacles. Moreover, we will prioritize the most important features and need to make intelligent trade-offs between the depth of completeness of a single feature and the breadth of implementation of multiple features.

Testing

In this stage, we will test the performance of each feature in order to check whether it meets the requirements of users. For example, we will test whether the application can be properly installed and run on a real device, and check whether any errors occur in the running process and each feature is up to standard.

Deployment

In this final phase, we will begin to deliver this application to the customer. For instance, we will upload this MP3 music player application in the Google Play Store, or posting download links on Utar Confession which is on Facebook in order to allow UTAR students to use it. In addition, we will anticipate that users will encounter unpredictable problems when using the player in this process, so we will solve these problems in a future version.

4.2 Tool to use

4.2.1 Software Requirement

1. Android Studio
2. Java SE 8
3. Visual Paradigm

4.2.2 Hardware Requirement

1. Laptop
   - Processor: Intel(R) Core(TM) i7-4500U CPU @ 1.80GHz 2.40 GHz
   - RAM: 12.00 GB
   - Graphic Card: NVIDIA GeForce GT 740M
   - Hard Disk storage: 1TB
   - Operating System: Windows 10 Professional Edition

2. Smartphone device
   - Processor make: Qualcomm Snapdragon 652 (MSM8976)
   - RAM: 4.0 GB
   - Phone Storage: 64 GB
   - Operating System: Android version 5.1.1 (Lollipop)

4.3 User requirements

Functional Requirements:

- Able to import the MP3 format files on the device into the music player
- Able to play audio files smoothly
- Able to download or listen to the song by searching for the name of the song, album, or artist
- Driving mode
- Night mode
- Songs in playlists can be quickly searched and filtered through the alphabet and search bar

Non-functional Requirements:

- Simplify user interface
- Optimize the design to display the information in a better ways

4.4 System Performance Definition

To achieve targeted system performance for this project, there are criteria's must be achieved as the list shown below:

1. Hardware/Software Variation

   The MP3 music player application needs to ensure the application can work and installed properly and smoothly on different devices. For example, the hardware and operating system of each phone will be different, just like Oppo and Huawei phone is ColorOS and EMUI based on android respectively, so need to test on different devices. In addition, the application should adapt to different devices to ensure that no errors occur during execution to avoid different results on different devices (Guru99, n.d.).

2. App in background

   Since this project is developing a music player, the running state of this application in the background is very important. If it does not work properly in the background, this will cause the user to have to input data from scratch or restart the application when retrieving the application, resulting in data loss and so on (Guru99, n.d.).

4.5 Timeline

ID	Task Name	Duration (Week)	Start (dd/mm/yyyy)	Finish (dd/mm/yyyy)	Week													
					1	2	3	4	5	6	7	8	9	10	11	12	13	14
	Analysis																	
1	Review previous work	2	27/05/2019	09/06/2019														
2	Define project scope	1	10/06/2019	16/06/2019														
3	Define project objective	1	10/06/2019	16/06/2019														
4	Gathering user requirement	1	10/06/2019	16/06/2019														
	Design																	
5	Design system architecture	2	17/06/2019	30/06/2019														
6	Identify development tools	2	17/06/2019	30/06/2019														
	Implementation																	
7	Coding	3	01/07/2019	21/06/2019														
8	Testing	1	22/07/2019	28/07/2019														
9	Maintaining	2	29/07/2019	11/08/2019														
10	Documentation	1	12/08/2019	18/08/2019														

Figure 4-5-1 Gantt chart for FYP1

ID	Task Name	Duration (Week)	Start (dd/mm/yyyy)	Finish (dd/mm/yyyy)	Week													
					1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Review previous work	1	13/01/2020	19/01/2020														
2	Design Prototype 2	1	20/01/2020	26/01/2020														
3	Development in Prototype 2	2	27/01/2020	09/02/2020														
4	Testing and Debugging on the Prototype	1	10/02/2020	16/02/2020														
5	Review Prototype	1	17/02/2020	23/02/2020														
6	Design Final Prototype	2	24/02/2020	08/03/2020														
7	Full System Development	3	09/03/2020	12/04/2020														
8	Testing and Debugging on Final system	1	13/04/2020	19/03/2020														
9	Finalize Report	2	13/04/2020	26/04/2020														
10	Project Submission	1	20/04/2020	26/04/2020														
11	Presentation	2	20/04/2020	03/05/2020														

Figure 4-5-2 Gantt chart for FYP2

Chapter 5 Implementation and Testing

5.1 Implementation

The proposed application completed the debugging task during the testing phase, then it should enter the deployment phase. In the deployment phase, the developer needs to publish the application's installation package which is the "APK" file, to a platform such as Google PlayStore for users to download. However, due to the number of users are limit so far and the proposed application is not in the final public version, there are still many modules that should be improved and updated. Therefore, it will be uploaded to the relevant platform to promote to users after the final public version is released. In addition, users can execute the app in a non-network state, but the "download" module requires an Internet connection to open the relevant web page to download the song.

Below are the steps to describe how a new user will execute the proposed application:

1. The user first execute the application, he or she needs to give the proposed application the permissions it needs to read local songs on the phone and load them into the song playlist.
2. Users can play a song by clicking on one of the songs on the playlist.
3. In the song playback interface, the user is allowed to drag the progress bar, as well as perform media control through the icon buttons, gesture and shaking the phone.
4. Back on the home page, users can click on the left navigation bar to select the "Audio Trim" module to trim music.
5. The user clicks the "open" icon button in the upper right corner and selects a song to begin trimming the song.

6. The user can use the progress bar to set the start and end times and click the start trim button.

7. After trimmed successfully, the user can decide to set the trimmed song to the ringtone or not.
8. In addition, users can click on the left navigation bar to enable "sleep timer," "night mode," "change theme color," and "download" features.

5.2 Testing

Unit Testing 1: Music Player

Test Objective: To ensure that the song selected by the user can be played normally, the selected song information is displayed normally, and the song playlist can be import and show properly

Input	Expected Output	Actual Output
Click "All" in the category	The song playlist under All category was successfully read and contains English songs and other songs	Pass
Click "English" in the category	The song playlist under the English category has been successfully read and contains only English songs	Pass
Click "Others" in the category	The song playlist under the category of Others was successfully read and contains only other songs	Pass
Choose any song from the playlist	Enter the song playing interface, the song can be playing properly, and successfully display the selected song name, album name and artist name	Pass
Play any song and click the Home button to make the app run in the background	Songs still playing in the background	Pass

Table 5-2-1 Unit Testing of Music Player Module

Unit Testing 2: Media Icon Playback Control

Test Objective: To ensure that all playback control icon buttons under playing song interface can work and perform properly

Input	Expected Output	Actual Output
Click the song playback mode to switch to shuffle playback	When the current song finished playing, and it will randomly choose a song from the playlist to play, rather than playing the song in the order of the playlist	Pass
Click the song playback mode to switch to the single cycle	When the current song finished playing, just repeat the current song	Pass
Click the song playback mode to switch to normal playback	When the current song finished playing, play the song in the order of the playlist	Pass
Click the pause button	Stop play the song, the pause button is updated to the play button	Pass
Click the play button	Start play the song and update play the button to pause button	Pass
Click the next button to switch to the next song	Switch to the next song, the song name, album name, and artist name are also successfully updated to the next song's information	Pass
Click the previous button to return to the previous song	Switch to the previous song, the song name, album name, and artist name are also successfully updated to the previous song's information	Pass
Click the driver play mode button	Enter into driver play mode interface	Pass
Click the back button	Stop playing the song and return to the home page "my music"	Pass

Table 5-2-2 Unit Testing of Media Icon Button Playback Control Module

Unit Testing 3: Driver Mode

Test Objective: To ensure all button and song info under driver playing mode interface will be enlarged and can play the song properly

Input	Expected Output	Actual Output
Click driver mode button in the song playing interface	Successfully switch to the driver mode interface, and successfully update the current song playing information	Pass

Table 5-2-3 Unit Testing of Driver Mode Module

Unit Testing 4: Swipe Gesture Playback Control

Test Objective: To ensure that swipe left or right under playing song interface and driver playing mode can properly switch song

Input	Expected Output	Actual Output
Swipe left in the song playing interface	Successfully switch to the next song and update the song information	Pass
Swipe right in the song playing interface	Successfully switch to the previous song and update the song information	Pass
Swipe left in driver mode interface	Successfully switch to the next song and update the song information	Pass
Swipe right in the driver mode interface	Successfully switch to the previous song and update the song information	Pass

Table 5-2-4 Unit Testing of Swipe Gesture Playback Control Module

Unit Testing 5: Shaking Playback Control

Test Objective: To ensure that shaking the phone can properly switch to next song

Input	Expected Output	Actual Output
Shake the phone in the song playing interface	Successfully switch to the next song and update the song information	Pass

Table 5-2-5 Unit Testing of Shaking Playback Control Module

Unit Testing 6: Audio Trim

Test Objective: To ensure that the selected song can be trim and set as ringtone

Input	Expected Output	Actual Output
Click the "Select Music" button	The playlist under "Select music" interface successfully reads the local songs on the device	Pass
Select any song file and set the start and end time before clicking the trim icon button	Successful trim and a dialog box pops up asking for entering the name of the new song file	Pass
Click "Yes" to set the trimmed song as ringtone	Successfully set the trimmed song as ringtone	Pass
Click "Cancel" to refuse to set the trimmed song as a ringtone	Failed set the trimmed song as ringtone and remains unchanged	Pass
Click the icon button to start the trim before do not select any song file	Invalid trim and pop up toast message prompt "Please select a song file"	Pass
Click the icon button to start playing before do not select any song file	Invalid play and pop up toast message prompt "Please select a song file"	Pass

Table 5-2-6 Unit Testing of Audio Trim Module

Unit Testing 7: Sleep Timer

Test Objective: To ensure that sleep time can be enable or disable and set up countdown time successfully

Input	Expected Output	Actual Output
Click the "Sleep timer" icon button in the left navigation bar	Successfully enter the sleep timer setting interface	Pass
Click the Enable button	Allows the user to proceed to the next step, such as setting the countdown time	Pass
Set the countdown time	Update the prompt message "Stop playing after x seconds" after the countdown time is set	Pass
After enabled sleep timer and setting the countdown time, click the "Save" button	The toast message prompts that the sleep timer has been set successfully, the remaining time can be checked in the left navigation bar, and the application will be terminated after the countdown time ends	Pass
After successfully setting the sleep timer, go back to the sleep timer settings interface and click "Disable" and then "Save" button	Successfully disable sleep timer, cancel the countdown time and pop up the toast message "Sleep timer has been successfully canceled"	Pass

Table 5-2-7 Unit Testing of Sleep Timer Module

Unit Testing 8: Night Mode

Test Objective: To ensure that night mode can be enable or disable and set the background of each layout to dark color after enabled

Input	Expected Output	Actual Output
Enable night mode	Icon button updated to the sun icon, "My music", "Audio trim" and "Sleep timer settings" interface changed to a dark color background	Pass
Disable night mode	Icon button update back to the moon icon, "My music", "Audio trim" and "Sleep timer settings" interface update into a brighter color background	Pass

Table 5-2-8 Unit Testing of Night Mode Module

Unit Testing 9: Change Theme Color

Test Objective: To ensure that theme color of toolbar can be change

Input	Expected Output	Actual Output
Choose one out of eight colors and click the “Confirm” button	Successfully updated the theme color of the toolbar to the color selected by the user	Pass

Table 5-2-9 Unit Testing of Change Theme Color Module

Chapter 6 Conclusion

6.1 Project Review, Discussions and Conclusion

In a nutshell, when users hold the mentality of venting and relaxation to expect the music player to bring them relief pressure, in result the application with a dazzling and complex interface, a variety of multifarious functions, from time to time prompt out of the advertising, as well as the function that requires be a members to use, which will only make users feel more depressed and feel the pressure.

Moreover, most people who use a music player, usually don't leave the music player open in the foreground, but start playing music and then go on to do something else at hand such as take a break, read a book and news, or play a game. As a result, they can't focus on the various functions and buttons in the app's interface. For instance, users who are lying down to take a break and tried to switch to the next song but they need lots of action like unlocking the phone, open the app again in the background and look for the switch button.

In addition, the specific song is overwhelmed by a large number of songs and cause information overload, users can only spend more energy and time to find it. For example, searching for a book in the library, and realize that there is no library catalog is mean to looking for a needle in a haystack.

In short, the proposed application will combine the strengths of most music players on the existing market and eliminate some unrealistic features, allowing users to focus on listening to music rather than store, communities or various VIP packages or features. The proposed MP3 music player will focus on improving the experience of users of the music player experience.

6.1.1 Project Achievement

Firstly, the proposed music player had achieved its first objective, which is to make the music player become a simple, easy-to-use, and well-run application. The proposed application had become faster startup, smaller size, and less memory usage by eliminating some unrealistic features. The application also adding some useful features like audio trim.

Next, the proposed music player achieves a second objective which is to reduce the use of button controls and enhance the way the app interacts with the user, such as using gestures and shaking controls. Using gestures, the user doesn't have to pay full attention to the phone, but simply swipes right or left at any place in the playing interface to switch the songs. In addition, if the app is running on the lock screen or in the background, users can successfully switch to the next song by simply shaking the phone, completely eliminating the use of buttons.

Lastly, the proposed music player achieves a third objective which is a quick search. The application will use the search bar and the alphabet fast scroll, allowing users to quickly traverse the song playlist and find the songs they want, which is an efficient way. For example, if a user wants to search the playlist for a song called "Lemon", he or she just enters a character that is 'm' into the search bar and the result appears. This is because the name of the song contains the characters entered in the search bar.

6.1.2 Problem Encountered

The main problem encounter in this project is the structure of code which the structure no build well at first and led to keep modification during developing night mode. Since the layout interface of each activity is not build as uniformly managed and updated at the beginning, only one layout is updated to the dark color after the night mode is enabled, and the layout of the other remains unchanged. Fortunately, these problems were solved in the final development stage, but the solution is informal and no efficiently, as it is achieved by using a lot of duplicate coding to solve.

6.2 Future Work

I. Enhanced interactivity, allowing users to open song playlist when they swipe up from the music playing interface
II. Implement of the in-app download of songs, rather than the current use of a specific website as a hyperlink
III. Refactoring code, rebuild the coding structure to make the coding look cleaner, easier to understand and perform efficiently
IV. Cross-platform, running the app on IOS, not only Android

BIBLIOGRAPHY

Guru99.com. (2019). *Mobile App Performance Testing: CheckList, Tools (Andriod & iOS)*. [online] Available at: https://www.guru99.com/mobile-app-performance-testing-strategy-tools.html#1 [Accessed 15 Jul. 2019].

Joox.com. (2019). *JOOX - Music Anytime Anywhere*. [online] Available at: https://www.joox.com/my-en [Accessed 28 Jun. 2019].

Kr-asia.com. (2019). *Spotify facing growth pains in Asia, where Tencent's Joox is more popular – KrASIA*. [online] Available at: https://kr-asia.com/spotify-facing-growth-issue-in-asia-where-tencents-joox-is-more-popular [Accessed 29 Jun. 2019].

Mehul, R. (2018). *Top Reasons Why Your Mobile App is Slow and How to Fix it*. [online] Available at: https://www.freecodecamp.org/news/top-reasons-why-your-mobile-app-is-slow-and-how-to-fix-it-f0f7ce524934/ [Accessed 19 Mar. 2020].

Play.google.com. (2019). [online] Available at: https://play.google.com/store/apps/details?id=com.gugu.music&hl=en [Accessed 25 Jun. 2019].

Play.google.com. (2019). [online] Available at: https://play.google.com/store/apps/details?id=fm.xiami.main&hl=en [Accessed 26 Jun. 2019].

Sohail, A. (2019). *Agile Project Management: Understanding and Benefits - DevCrew I/O. [online] DevCrew I/O*. Available at: https://devcrew.io/2017/08/03/agile-basics/ [Accessed 17 Jul. 2019].

Scacca, S. (2020). *When to Design with Gestures vs. Buttons*. [online] Available at: https://www.telerik.com/blogs/when-to-design-with-gestures-vs-buttons [Accessed 19 Mar. 2020].

Xiami.com. (2019). *虾米音乐 - 发现音乐新世界*. [online] Available at: https://www.xiami.com/ [Accessed 27 Jun. 2019].

APPENDICES A

A.1 MusicAdapter.java

```
package com.example.fyp2_tsk.adapter;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.TextView;
import com.example.fyp2_tsk.utils.Common;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.entity.Music;
import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

//Inherit the BaseAdapter to implement a custom adapter and rewrite the code of BaseAdapter
public class MusicAdapter extends BaseAdapter{
    private Context context;
    private List<Music> musicList;
    private ArrayList<Music> MusicarrayList;

    public MusicAdapter(Context context, List<Music> musicList) {
        this.context = context;
        this.musicList = musicList;
        this.MusicarrayList = new ArrayList<Music>();
        this.MusicarrayList.addAll(Common.musicList);
    }

    @Override
    public int getCount() {
        return Common.musicList.size();
    }

    @Override
    public Object getItem(int position) {
        return null;
    }
```

```
    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = null;
        ViewHolder viewHolder = new ViewHolder();
        //Cache principle,to determine whether convertView is empty
        if (convertView == null) {
            //binding layout file
            view = LayoutInflater.from(context).inflate(R.layout.music_item, null);

            viewHolder.titleTv = (TextView) view.findViewById(R.id.musicitem_title_tv);
            viewHolder.artistTv = (TextView) view.findViewById(R.id.musicitem_artist_tv);
            viewHolder.albumImgv = (ImageView) view.findViewById(R.id.musicitem_album_imgv);
            viewHolder.isPlayingView = view.findViewById(R.id.musicitem_playing_v);
            view.setTag(viewHolder);
        } else {
            view = convertView;
            viewHolder = (ViewHolder) view.getTag();
        }

        viewHolder.titleTv.setText(Common.musicList.get(position).title);
        viewHolder.artistTv.setText(Common.musicList.get(position).artist + "-" +
Common.musicList.get(position).album);

        if (Common.musicList.get(position).isPlaying) {
            viewHolder.isPlayingView.setVisibility(View.VISIBLE);
        } else {
            viewHolder.isPlayingView.setVisibility(View.INVISIBLE);
        }

        return view;
    }

    public void filter(String charText){
        charText = charText.toLowerCase(Locale.getDefault());
        Common.musicList.clear();
        if (charText.length()==0){
            Common.musicList.addAll(MusicarrayList);
        }
```

```
        else {
            for (Music music : MusicarrayList){
                if (music.getTitle().toLowerCase(Locale.getDefault())
                        .contains(charText)){
                    Common.musicList.add(music);
                }
            }
        }
        notifyDataSetChanged();
    }

    //Used to store the controls in music_item.xml
    class ViewHolder {
        TextView titleTv;
        TextView artistTv;
        ImageView albumImgv;
        View isPlayingView;
    }

    public void notifyDataSetChanged() {
        super.notifyDataSetChanged();
    }

}
```

A.2 MusicPagerAdapter.java

```
package com.example.fyp2_tsk.adapter;

import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;

import java.util.List;

public class MusicPagerAdapter extends FragmentPagerAdapter {
    private List<Fragment> fragmentList ;

    public MusicPagerAdapter(FragmentManager fm,List<Fragment> fragmentList) {
        super(fm);
        this.fragmentList = fragmentList;
    }


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

    @Override
    public int getCount() {
        return fragmentList.size();
    }
}
```

A.3MyApplication.java

```
package com.example.fyp2_tsk.common.app;

import android.app.Application;
import com.example.fyp2_tsk.common.utils.LogUtils;
import com.orhanobut.logger.AndroidLogAdapter;
import com.orhanobut.logger.Logger;
import com.umeng.analytics.MobclickAgent;
import com.example.fyp2_tsk.common.app.di.AppComponent;
import com.example.fyp2_tsk.common.app.di.AppModule;
import com.example.fyp2_tsk.common.app.di.DaggerAppComponent;
import com.example.fyp2_tsk.home.bean.MyObjectBox;
import io.objectbox.BoxStore;
import skin.support.SkinCompatManager;
import skin.support.design.app.SkinMaterialViewInflater;

public class MyApplication extends Application {
    private AppComponent mAppComponent;
    public static MyApplication instances;
    private BoxStore mBoxStore;
    @Override
    public void onCreate() {
        super.onCreate();
        instances = this;
        mAppComponent = DaggerAppComponent
                .builder()
                .appModule(new AppModule(this))
                .build();
        initDatabase();

        //logger
        if(LogUtils.isApkDebugable(this)) {
            Logger.addLogAdapter(new AndroidLogAdapter());
        }

        //umeng analytics
        MobclickAgent.setScenarioType(this, MobclickAgent.EScenarioType.E_UM_NORMAL);

        SkinCompatManager.withoutActivity(this)                         // Basic control
skin initialization
                .addInflater(new SkinMaterialViewInflater())            // Material design
control skinning initialization [optional]
                .setSkinStatusBarColorEnable(false)                     // Close the status
```

```
bar skinning, open by default [optional]
                .setSkinWindowBackgroundEnable(false)                    //Close
windowBackground skinning, open by default [optional]
                .loadSkin();
    }

    public AppComponent getAppComponent() {
        return mAppComponent;
    }



    public static MyApplication getInstances() {
        return instances;
    }

    private void initDatabase() {
        mBoxStore = MyObjectBox.builder().androidContext(this).build();
    }

    public BoxStore getBoxStore(){
        return mBoxStore;
    }
}
```

A.4 ActivityScope.java

```
package com.example.fyp2_tsk.common.app.di;

import java.lang.annotation.Retention;
import javax.inject.Scope;
import static java.lang.annotation.RetentionPolicy.RUNTIME;

@Scope
@Retention(RUNTIME)
public @interface ActivityScope {
}
```

A.5 AppComponent.java

```
package com.example.fyp2_tsk.common.app.di;

import javax.inject.Singleton;
import dagger.Component;

@Singleton
@Component(modules = AppModule.class)
public interface AppComponent {

}
```

A.6 AppModule.java

```
package com.example.fyp2_tsk.common.app.di;

import android.app.Application;
import javax.inject.Singleton;
import dagger.Module;
import dagger.Provides;

@Module
public class AppModule {
    private Application mApplication;
    public AppModule(Application application) {
        this.mApplication = application;
    }

    @Singleton
    @Provides
    public Application provideApplication() {
        return mApplication;
    }
}
```

A.7 BaseActivity.java

```
package com.example.fyp2_tsk.common.base;

import android.content.Context;
import android.databinding.DataBindingUtil;
import android.databinding.ViewDataBinding;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import com.umeng.analytics.MobclickAgent;
import com.example.fyp2_tsk.common.app.MyApplication;
import com.example.fyp2_tsk.common.app.di.AppComponent;
import javax.inject.Inject;

public abstract class BaseActivity<V extends IBaseView,T extends BasePresenter<V>, B extends
ViewDataBinding>
        extends AppCompatActivity implements IBaseView {
    protected MyApplication myApplication;

    //Dependency Injection
    protected abstract void ComponentInject(AppComponent appComponent);
    protected abstract int initLayoutResId();
    protected abstract void initData(Bundle savedInstanceState);

    @Inject
    public T mPresenter;
    protected  B mDataBinding;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mDataBinding = DataBindingUtil.setContentView(this, initLayoutResId());
        myApplication = (MyApplication) getApplication();
        ComponentInject(myApplication.getAppComponent());//Dependency Injection
        initData(savedInstanceState);
    }

    @Override
    public Context getContext(){
        return this;
    }

    @Override
```

```
    protected void onResume() {
        super.onResume();
        MobclickAgent.onResume(this);
    }

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

A.8 BaseFragment.java

```
package com.example.fyp2_tsk.common.base;

import android.content.Context;
import android.databinding.DataBindingUtil;
import android.databinding.ViewDataBinding;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import com.example.fyp2_tsk.common.app.MyApplication;
import com.example.fyp2_tsk.common.app.di.AppComponent;
import javax.inject.Inject;

public abstract class BaseFragment<V extends IBaseView,T extends BasePresenter<V>, B extends
ViewDataBinding> extends Fragment implements IBaseView {
    protected MyApplication myApplication;
    @Inject
    public T mPresenter;
    protected  B mDataBinding;
    View view;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        if (view != null){
            ViewGroup parent = (ViewGroup) view.getParent();
            if (parent != null){
                parent.removeView(view);
            }
            return  view;
        }

        myApplication = (MyApplication)getActivity().getApplication();
        ComponentInject(myApplication.getAppComponent());

        mDataBinding = DataBindingUtil.inflate(inflater, initLayoutResId(), container,
false);
        view = mDataBinding.getRoot();

        init(view);
        return view;
```

```
    }

    @Override
    public Context getContext() {
        return super.getContext();
    }

    protected abstract void ComponentInject(AppComponent appComponent);

    protected abstract void init(View view);

    protected abstract int initLayoutResId();
}
```

A.9 BasePresenter.java

```
package com.example.fyp2_tsk.common.base;

public class BasePresenter<V extends IBaseView>{
    protected V mView;

    public BasePresenter(V rootView) {
        this.mView = rootView;
    }
}
```

A.10 IBaseView.java

```
package com.example.fyp2_tsk.common.base;

import android.content.Context;

public interface IBaseView {
     Context getContext();
}
```

A.11 EasyRecyclerViewAdapter.java

```
package com.example.fyp2_tsk.common.base;

import android.support.v7.widget.GridLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.StaggeredGridLayoutManager;
import android.view.View;
import android.view.ViewGroup;
import java.util.ArrayList;
import java.util.List;


public abstract class EasyRecyclerViewAdapter<T> extends
RecyclerView.Adapter<RecyclerView.ViewHolder> {
    private static final int TYPE_HEAD = 0;
    private static final int TYPE_BODY = 1;
    private static final int TYPE_FOOT = 2;
    private View mHeaderView;
    private View mFooterView;
    private List<T> mDatas = new ArrayList<>();
    private OnItemClickListener mItemClickListener;
    private OnItemLongClickListener mItemLongClickListener;

    public void setOnItemClickListener(OnItemClickListener listener) {
        mItemClickListener = listener;
    }

    public void setDatas(List<T> mDatas) {
        this.mDatas = mDatas;
        notifyDataSetChanged();
    }

    public EasyRecyclerViewAdapter() {
    }

    @Override
    public int getItemViewType(int position) {
        if (mFooterView == null && mHeaderView == null) {
            return TYPE_BODY;
        }
        if (mHeaderView != null && position == 0) {
            return TYPE_HEAD;
        }
```

```
        if (mFooterView != null && mHeaderView == null && position == mDatas.size()) {
            return TYPE_FOOT;
        }
        if (mFooterView != null && mHeaderView != null && position == mDatas.size() + 1) {
            return TYPE_FOOT;
        }
        return TYPE_BODY;
    }

    @Override
    public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        if (viewType == TYPE_HEAD) {
            return new EasyViewHolder(mHeaderView);
        }
        if (viewType == TYPE_FOOT) {
            return new EasyViewHolder(mFooterView);
        }
        return onCreate(parent, viewType);//get resources file
    }

    @Override
    public void onBindViewHolder(RecyclerView.ViewHolder holder, int position) {
        if (getItemViewType(position) == TYPE_HEAD || getItemViewType(position) ==
TYPE_FOOT) {
            return;
        }

        final int dataPosition = getDataPosition(holder);
        final T data = mDatas.get(dataPosition);
        onBind(holder, dataPosition, data);

        if (mItemClickListener != null) {
            holder.itemView.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    mItemClickListener.OnItemClick(v, dataPosition, data);
                }
            });
        }

        if (mItemLongClickListener != null) {
            holder.itemView.setOnLongClickListener(new View.OnLongClickListener() {
                @Override
                public boolean onLongClick(View v) {
```

```
                    mItemLongClickListener.OnItemLongClick(v, dataPosition, data);
                    return true;
                }
            });
        }

    }

    @Override
    public void onAttachedToRecyclerView(RecyclerView recyclerView) {
        super.onAttachedToRecyclerView(recyclerView);
        final RecyclerView.LayoutManager layoutParams = recyclerView.getLayoutManager();
        if (layoutParams instanceof GridLayoutManager) {
            ((GridLayoutManager) layoutParams).setSpanSizeLookup(new
GridLayoutManager.SpanSizeLookup() {
                @Override
                public int getSpanSize(int position) {
                    int type = getItemViewType(position);
                    if (type == TYPE_HEAD || type == TYPE_FOOT)
                        return ((GridLayoutManager) layoutParams).getSpanCount();
                    return 1;
                }
            });
        }

    }

    @Override
    public void onViewAttachedToWindow(RecyclerView.ViewHolder holder) {
        super.onViewAttachedToWindow(holder);
        ViewGroup.LayoutParams layoutParams = holder.itemView.getLayoutParams();
        if (layoutParams instanceof StaggeredGridLayoutManager.LayoutParams) {
            StaggeredGridLayoutManager.LayoutParams params =
(StaggeredGridLayoutManager.LayoutParams) layoutParams;
            if (mHeaderView != null && holder.getLayoutPosition() == 0) {
                params.setFullSpan(true);
            } else if (mFooterView != null && holder.getLayoutPosition() ==
getFooterPosition()) {
                params.setFullSpan(true);
            } else {
                params.setFullSpan(false);
            }
        }
    }
```

```
    public int getDataPosition(RecyclerView.ViewHolder holder) {
        int position = holder.getLayoutPosition();
        return mHeaderView == null ? position : position - 1;
    }

    private int getFooterPosition() {
        if (mFooterView == null) {
            return -1;
        }
        if (mHeaderView == null) {
            return mDatas.size();
        }
        if (mHeaderView != null) {
            return mDatas.size() + 1;
        }
        return -1;
    }

    @Override
    public int getItemCount() {
        if (mFooterView != null && mHeaderView != null) {
            return mDatas.size() + 2;
        } else if (mFooterView != null || mHeaderView != null) {
            return mDatas.size() + 1;
        }
        return mDatas.size();
    }

    public abstract RecyclerView.ViewHolder onCreate(ViewGroup parent, final int viewType);

    public abstract void onBind(RecyclerView.ViewHolder viewHolder, int RealPosition, T
data);

    public class EasyViewHolder extends RecyclerView.ViewHolder {

        public EasyViewHolder(View itemView) {
            super(itemView);
        }
    }

    public interface OnItemClickListener<T> {
        public void OnItemClick(View view, int position, T data);
    }
```

```
    interface OnItemLongClickListener<T> {
        public void OnItemLongClick(View view, int position, T data);
    }
}
```

A.12 CommonConstant.java

```
package com.example.fyp2_tsk.common.constant;

public class CommonConstant {
    // Ringtone path
    public static String RING_FOLDER = "/sdcard/mp3cutter";
}
```

A.13 FFTData.java

```
package com.example.fyp2_tsk.common.ui.view.visualizer;

// Data class to explicitly indicate that these bytes are the FFT of audio data
public class FFTData {
    public byte[] bytes;

    public FFTData() {
    }

    public FFTData setBytes(byte[] bytes) {
        this.bytes = bytes;
        return this;
    }
}
```

A.14 AudioData.java

```
package com.example.fyp2_tsk.common.ui.view.visualizer;
```

```
// Data class to explicitly indicate that these bytes are raw audio data
public class AudioData {
    public byte[] bytes;

    public AudioData() {
    }

    public AudioData setBytes(byte[] bytes) {
        this.bytes = bytes;
        return this;
    }
}
```

A.15 CommonDialog.java

```
package com.example.fyp2_tsk.common.ui.view;

import android.app.Activity;
import android.app.Dialog;
import android.content.Context;
import android.text.InputType;
import android.text.TextUtils;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

import com.example.fyp2_tsk.R;

public class CommonDialog extends Dialog {
    private TextView mTitleTv;
    private TextView mContentTv;
    private Context mContext;
    public TextView mConfirmTv;
    public TextView mCancelTv;
    private EditText mInputEt;
    private String mContentStr;
    private String mTitleStr;
    private int mRightTextColor;
    private boolean mIsShowOne;
    private boolean mIsShowInput;
    private String mHintContent;
    private boolean mIsPasswordInput;
```

```
    private int mCancelColor;// = Color.parseColor("#009688");
    public OnDialogClickListener mOnClickListener;

    private CommonDialog(Builder builder) {
        super(builder.context, R.style.dialog_normal);
        mContext = builder.context;
        setContentView(R.layout.dialog_common);
        initView();
        initForBuilder(builder);
        refreshView();
    }

    private void initForBuilder(Builder builder) {
        mTitleStr = builder.titleStr;
        mContentStr = builder.contentStr;
        mOnClickListener = builder.mOnDialogClickListener;
        mCancelColor = builder.cancelColor;
        mRightTextColor = builder.rightTextColor;
        mIsShowOne = builder.isShowOne;
        mIsShowInput = builder.isShowInput;
        mHintContent = builder.hintContent;
        mIsPasswordInput = builder.isPasswordInput;
        setCancelable(builder.canCancel);
    }

    private void initView() {
        this.mTitleTv = (TextView) this.findViewById(R.id.tv_dialog_title);
        this.mContentTv = (TextView) this.findViewById(R.id.tv_dialog_content);
        this.mConfirmTv = (TextView) this.findViewById(R.id.tv_dialog_confirm);
        this.mCancelTv = (TextView) this.findViewById(R.id.tv_dialog_cancle);
        this.mInputEt = (EditText) this.findViewById(R.id.et_dialog_input);
    }

    private void refreshView() {
        if (mTitleStr != null) {
            mTitleTv.setVisibility(View.VISIBLE);
            mTitleTv.setText(mTitleStr);
        } else {
            mTitleTv.setVisibility(View.GONE);
        }
        if (mContentStr != null) {
            this.mContentTv.setVisibility(View.VISIBLE);
            this.mContentTv.setText(mContentStr);
        } else {
```

```
            mContentTv.setVisibility(View.GONE);
        }
        if (mRightTextColor != 0) {
            this.mConfirmTv.setTextColor(mRightTextColor);
        }
        if (mCancelColor != 0) {
            this.mCancelTv.setTextColor(this.mCancelColor);
        }
        if (mIsShowOne) {
            this.mCancelTv.setVisibility(View.GONE);
        } else {
            this.mCancelTv.setVisibility(View.VISIBLE);
        }
        if (mIsShowInput) {
            this.mInputEt.setVisibility(View.VISIBLE);
            this.mInputEt.setHint(mHintContent);
            this.mInputEt.setInputType(mIsPasswordInput ?
InputType.TYPE_TEXT_VARIATION_PASSWORD :
                    InputType.TYPE_CLASS_TEXT);
        } else {
            this.mInputEt.setVisibility(View.GONE);
        }
        this.setDataAndListener();
    }

    private void setDataAndListener() {
        this.mConfirmTv.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                CommonDialog.this.close();
                if (CommonDialog.this.mOnClickListener != null) {
                    if (TextUtils.isEmpty(CommonDialog.this.getEditText()))
                        (CommonDialog.this.mOnClickListener).doOk();
                    else {

(CommonDialog.this.mOnClickListener).doOk(CommonDialog.this.getEditText());
                    }
                }

            }
        });
        this.mCancelTv.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                CommonDialog.this.close();
                if (CommonDialog.this.mOnClickListener != null) {
```

```
                CommonDialog.this.mOnClickListener.doCancle();
            }

        }
    });
}

public void close() {
    if (!((Activity) this.mContext).isFinishing()) {
        ((Activity) this.mContext).runOnUiThread(new Runnable() {
            public void run() {
                if (CommonDialog.this.isShowing()) {
                    CommonDialog.this.dismiss();
                }

            }
        });
    }

}

public String getEditText() {
    return this.mInputEt.getText().toString().trim();
}

public static class OnDialogClickListener {
    public void doOk() {
    }

    public void doOk(String text) {
    }

    public void doCancle() {
    }
}


public static final class Builder {
    private Context context;
    private String titleStr;
    private String contentStr;
    private int cancelColor;
    private int rightTextColor;
    private boolean isShowOne;
```

```
        private boolean isShowInput;
        private String hintContent;
        private boolean isPasswordInput;
        private boolean canCancel = true;
        private OnDialogClickListener mOnDialogClickListener;

        public Builder() {
        }

        public Builder setContext(Context context) {
            this.context = context;
            return this;
        }

        public Builder setContentStr(String val) {
            contentStr = val;
            return this;
        }

        public Builder setIsShowOne(boolean val) {
            isShowOne = val;
            return this;
        }

        public Builder setIsShowInput(boolean val) {
            isShowInput = val;
            return this;
        }

        public Builder setOnDialogListener(OnDialogClickListener val) {
            mOnDialogClickListener = val;
            return this;
        }

        public CommonDialog build() {
            return new CommonDialog(this);
        }
    }
}
```

A.16 CircleBarRenderer.java

```
package com.example.fyp2_tsk.common.ui.view.visualizer.renderer;

import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Rect;
import com.example.fyp2_tsk.common.ui.view.visualizer.AudioData;
import com.example.fyp2_tsk.common.ui.view.visualizer.FFTData;

public class CircleBarRenderer extends Renderer {
    private int mDivisions;
    private Paint mPaint;
    private boolean mCycleColor;

    /**
     * Renders the FFT data onto a pulsing, rotating circle
     * @param paint - Paint to draw lines with
     */
    public CircleBarRenderer(Paint paint, int divisions) {
        this(paint, divisions, false);
    }

    /**
     * Renders the audio data onto a pulsing circle
     * @param
     * @param paint - Paint to draw lines with
     * @param divisions - must be a power of 2. Controls how many lines to draw
     * @param cycleColor - If true the color will change on each frame
     */
    public CircleBarRenderer(Paint paint, int divisions, boolean cycleColor) {
        super();
        mPaint = paint;
        mDivisions = divisions;
        mCycleColor = cycleColor;
    }

    @Override
    public void onRender(Canvas canvas, AudioData data, Rect rect) {
        // Do nothing, we only display FFT data
    }

    @Override
```

```
public void onRender(Canvas canvas, FFTData data, Rect rect) {
    if (mCycleColor) {
        cycleColor();
    }

    for (int i = 0; i < data.bytes.length / mDivisions; i++) {
        // Calculate dbValue
        byte rfk = data.bytes[mDivisions * i];
        byte ifk = data.bytes[mDivisions * i + 1];
        float magnitude = (rfk * rfk + ifk * ifk);
        float dbValue = 75 * (float) Math.log10(magnitude);

        float[] cartPoint = {
                (float) (i * mDivisions) / (data.bytes.length - 1),
                rect.height() / 2 - dbValue / 4
        };

        float[] polarPoint = toPolar(cartPoint, rect);
        mFFTPoints[i * 4] = polarPoint[0];
        mFFTPoints[i * 4 + 1] = polarPoint[1];

        float[] cartPoint2 = {
                (float) (i * mDivisions) / (data.bytes.length - 1),
                rect.height() / 2 + dbValue
        };

        float[] polarPoint2 = toPolar(cartPoint2, rect);
        mFFTPoints[i * 4 + 2] = polarPoint2[0];
        mFFTPoints[i * 4 + 3] = polarPoint2[1];
    }

    canvas.drawLines(mFFTPoints, mPaint);

    // Controls the pulsing rate
    modulation += 0.13;
    angleModulation += 0.28;
}

float modulation = 0;
float modulationStrength = 0.4f; // 0-1
float angleModulation = 0;
float aggresive = 0.4f;

private float[] toPolar(float[] cartesian, Rect rect) {
```

```
        double cX = rect.width() / 2;
        double cY = rect.height() / 2;
        double angle = (cartesian[0]) * 2 * Math.PI;
        double radius = ((rect.width() / 2) * (1 - aggresive) + aggresive * cartesian[1] /
2) * ((1 - modulationStrength) + modulationStrength * (1 + Math.sin(modulation)) / 2);
        float[] out = {
                (float) (cX + radius * Math.sin(angle + angleModulation)),
                (float) (cY + radius * Math.cos(angle + angleModulation))
        };
        return out;
    }

    private float colorCounter = 0;

    private void cycleColor() {
        int r = (int) Math.floor(128 * (Math.sin(colorCounter) + 1));
        int g = (int) Math.floor(128 * (Math.sin(colorCounter + 2) + 1));
        int b = (int) Math.floor(128 * (Math.sin(colorCounter + 4) + 1));
        mPaint.setColor(Color.argb(128, r, g, b));
        colorCounter += 0.03;
    }
}
```

A.17 Renderer.java

```
package com.example.fyp2_tsk.common.ui.view.visualizer.renderer;

import android.graphics.Canvas;
import android.graphics.Rect;
import com.example.fyp2_tsk.common.ui.view.visualizer.AudioData;
import com.example.fyp2_tsk.common.ui.view.visualizer.FFTData;

abstract public class Renderer {
    // Have these as members, so we don't have to re-create them each time
    protected float[] mPoints;
    protected float[] mFFTPoints;

    public Renderer() {
    }

    // As the display of raw/FFT audio will usually look different, subclasses
    // will typically only implement one of the below methods
    /**
     * Implement this method to render the audio data onto the canvas
     * @param canvas - Canvas to draw on
     * @param data - Data to render
     * @param rect - Rect to render into
     */
    abstract public void onRender(Canvas canvas, AudioData data, Rect rect);

    /**
     * Implement this method to render the FFT audio data onto the canvas
     * @param canvas - Canvas to draw on
     * @param data - Data to render
     * @param rect - Rect to render into
     */
    abstract public void onRender(Canvas canvas, FFTData data, Rect rect);


    // These methods should actually be called for rendering
    /**
     * Render the audio data onto the canvas
     * @param canvas - Canvas to draw on
     * @param data - Data to render
     * @param rect - Rect to render into
     */
    final public void render(Canvas canvas, AudioData data, Rect rect) {
```

```
        if (mPoints == null || mPoints.length < data.bytes.length * 4) {
            mPoints = new float[data.bytes.length * 4];
        }

        onRender(canvas, data, rect);
    }

    /**
     * Render the FFT data onto the canvas
     * @param canvas - Canvas to draw on
     * @param data - Data to render
     * @param rect - Rect to render into
     */
    final public void render(Canvas canvas, FFTData data, Rect rect) {
        if (mFFTPoints == null || mFFTPoints.length < data.bytes.length * 4) {
            mFFTPoints = new float[data.bytes.length * 4];
        }
        onRender(canvas, data, rect);
    }
}
```

A.18 VisualizerView.java

```
package com.example.fyp2_tsk.common.ui.view.visualizer;

import android.annotation.SuppressLint;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.PorterDuff.Mode;
import android.graphics.PorterDuffXfermode;
import android.graphics.Rect;
import android.media.MediaPlayer;
import android.media.audiofx.Visualizer;
import android.util.AttributeSet;
import android.util.Log;
import android.view.View;
import com.example.fyp2_tsk.common.ui.view.visualizer.renderer.Renderer;
import java.util.HashSet;
import java.util.Set;

/**
 * A class that draws visualizations of data received from a
 * {@link Visualizer.OnDataCaptureListener#onWaveFormDataCapture } and
 * {@link Visualizer.OnDataCaptureListener#onFftDataCapture }
 */
public class VisualizerView extends View {
    private byte[] mBytes;
    private byte[] mFFTBytes;
    private Rect mRect = new Rect();
    private Visualizer mVisualizer;
    private Set<Renderer> mRenderers;
    private Paint mFlashPaint = new Paint();
    private Paint mFadePaint = new Paint();
    AudioData mAudioData = new AudioData();
    FFTData mFftData = new FFTData();

    public VisualizerView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs);
        init();
    }
```

```
    public VisualizerView(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public VisualizerView(Context context) {
        this(context, null, 0);
    }

    private void init() {
        mBytes = null;
        mFFTBytes = null;

        mFlashPaint.setColor(Color.argb(122, 255, 255, 255));
        mFadePaint.setColor(Color.argb(238, 255, 255, 255)); // Adjust alpha to change how
quickly the image fades
        mFadePaint.setXfermode(new PorterDuffXfermode(Mode.MULTIPLY));

        mRenderers = new HashSet<Renderer>();
    }

    /**
     * Links the visualizer to a player
     *
     * @param player - MediaPlayer instance to link to
     */
    @SuppressLint("NewApi")
    public void link(MediaPlayer player) {
        if (player == null) {
            throw new NullPointerException("Cannot link to null MediaPlayer");
        }

        // Create the Visualizer object and attach it to our media player.
        try {
            mVisualizer = new Visualizer(player.getAudioSessionId());
            Log.d("mp3cutter:", "size:" + Visualizer.getCaptureSizeRange()[1]);
            mVisualizer.setCaptureSize(Visualizer.getCaptureSizeRange()[1]);

            // Pass through Visualizer data to VisualizerView

            mVisualizer.setDataCaptureListener(new Visualizer.OnDataCaptureListener() {
                                                   @Override
                                                   public void
onWaveFormDataCapture(Visualizer visualizer, byte[] bytes, int samplingRate) {
                                                       updateVisualizer(bytes);
```

```
                                                }

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

            // Enabled Visualizer and disable when we're done with the stream
            //mVisualizer.setEnabled(false);
            player.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
                @Override
                public void onCompletion(MediaPlayer mediaPlayer) {
                    //mVisualizer.setEnabled(false);
                }
            });
        } catch (UnsupportedOperationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalStateException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (RuntimeException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    @SuppressLint("NewApi")
    public void setEnabled(boolean enabled) {
        if (mVisualizer == null)
            return;
        mVisualizer.setEnabled(enabled);
    }

    public void addRenderer(Renderer renderer) {
        if (renderer != null) {
            mRenderers.add(renderer);
        }
    }

    public void clearRenderers() {
```

```
        mRenderers.clear();
    }


    /**
     * Pass data to the visualizer. Typically this will be obtained from the
     * Android Visualizer.OnDataCaptureListener call back. See
     * {@link Visualizer.OnDataCaptureListener#onWaveFormDataCapture }
     *
     * @param bytes
     */
    public void updateVisualizer(byte[] bytes) {
        mBytes = bytes;
        invalidate();
    }

    /**
     * Pass FFT data to the visualizer. Typically this will be obtained from the
     * Android Visualizer.OnDataCaptureListener call back. See
     * {@link Visualizer.OnDataCaptureListener#onFftDataCapture }
     *
     * @param bytes
     */
    public void updateVisualizerFFT(byte[] bytes) {
        mFFTBytes = bytes;
        invalidate();
    }

    Bitmap mCanvasBitmap;
    Canvas mCanvas;


    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        if (isInEditMode()) {
        }
        // Create canvas once we're ready to draw
        mRect.set(0, 0, getWidth(), getHeight());

        if (mCanvasBitmap == null) {
            mCanvasBitmap = Bitmap.createBitmap(canvas.getWidth(), canvas.getHeight(),
Config.ARGB_8888);
        }
```

```
        if (mCanvas == null) {
            mCanvas = new Canvas(mCanvasBitmap);
        }

        if (mBytes != null) {
            // Render all audio renderers
            mAudioData.setBytes(mBytes);
            for (Renderer r : mRenderers) {
                r.render(mCanvas, mAudioData, mRect);
            }
        }

        if (mFFTBytes != null) {
            // Render all FFT renderers
            mFftData.setBytes(mFFTBytes);
            for (Renderer r : mRenderers) {
                r.render(mCanvas, mFftData, mRect);
            }
        }

        // Fade out old contents
        mCanvas.drawPaint(mFadePaint);

        canvas.drawBitmap(mCanvasBitmap, 0, 0, null);
    }
}
```

A.19 DensityUtils.java

```
package com.example.fyp2_tsk.common.utils;

import android.content.Context;

public class DensityUtils {
    //Convert the dip or dp value to the px value to keep the size constant
    public static int dp2px(Context context, float dipValue) {
        final float scale = context.getResources().getDisplayMetrics().density;
        return (int) (dipValue * scale + 0.5f);
    }
}
```

A.20 LogUtils.java

```
package com.example.fyp2_tsk.common.utils;

import android.content.Context;
import android.content.pm.ApplicationInfo;

public class LogUtils {
    /**
     * When the debug attribute is not set in AndroidManifest.xml:
     * When using Eclipse to run this way of packaging, its debug property is true, when
using Eclipse to export this way, its debug property is false.
     * When using ant packaging, its value depends on whether ant's packaging parameters are
release or debug.
     * Therefore, it is best not to set the android: debuggable attribute in Android
Mainifest.xml, but to determine its value by the packaging method.
     */
    public static boolean isApkDebugable(Context context) {
        try {
            ApplicationInfo info = context.getApplicationInfo();
            return (info.flags & ApplicationInfo.FLAG_DEBUGGABLE) != 0;
        } catch (Exception e) {

        }
        return false;
    }
}
```

A.21 FileUtils.java

```
package com.example.fyp2_tsk.common.utils;

import android.text.TextUtils;

import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.text.DecimalFormat;
import java.util.List;

public class FileUtils {
    //creates if folder does not exist
    public static boolean bFolder(String strFolder) {
        boolean btmp = false;
        File f = new File(strFolder);
        if (!f.exists()) {
            if (f.mkdirs()) {
                btmp = true;
            } else {
                btmp = false;
            }
        } else {
            btmp = true;
        }
        return btmp;
    }

    //Check folder exist or not, if yes then delete it
    public static void checkAndDelFile(File file){
        if(file.exists()){
            file.delete();
        }
    }

    //Gets the file size in bytes
    public static long getFileSizes(File f) throws Exception {
        long s = 0;
        if (f.exists()) {
            FileInputStream fis = null;
```

```
            fis = new FileInputStream(f);
            s = fis.available();
        } else {
            System.out.println("File does not exist.");
        }
        return s;
    }

    //Gets the name of the file title without suffix
    public static String getFileTitle(String fileName) {
        if (TextUtils.isEmpty(fileName)) {
            return null;
        } else {
            int indexDot = fileName.lastIndexOf(".");
            return fileName.substring(0, indexDot);
        }
    }

    //Format file size
    public static String formetFileSize(long fileS) {
        DecimalFormat df = new DecimalFormat("#.00");
        String fileSizeString = "";
        if (fileS < 1024) {
            fileSizeString = df.format((double) fileS) + "B";
        } else if (fileS < 1048576) {
            fileSizeString = df.format((double) fileS / 1024) + "K";
        } else if (fileS < 1073741824) {
            fileSizeString = df.format((double) fileS / 1048576) + "M";
        } else {
            fileSizeString = df.format((double) fileS / 1073741824) +"G";
        }
        return fileSizeString;
    }
}
```

A.22 Md5Utils.java

```
package com.example.fyp2_tsk.common.utils;

import java.security.MessageDigest;



public class Md5Utils {
    public static String md5(String s) {
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            byte[] bytes = md.digest(s.getBytes("utf-8"));
            return toHex(bytes);
        }
        catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private static String toHex(byte[] bytes) {
        final char[] HEX_DIGITS = "0123456789ABCDEF".toCharArray();
        StringBuilder ret = new StringBuilder(bytes.length * 2);
        for (int i=0; i<bytes.length; i++) {
            ret.append(HEX_DIGITS[(bytes[i] >> 4) & 0x0f]);
            ret.append(HEX_DIGITS[bytes[i] & 0x0f]);
        }
        return ret.toString();
    }
}
```

A.23 Mp3ScanUtils.java

```
package com.example.fyp2_tsk.common.utils;

import com.example.fyp2_tsk.mp3cut.logic.Mp3InfoUtils;
import com.example.fyp2_tsk.home.bean.FileInfo;
import com.example.fyp2_tsk.home.bean.MusicInfo;
import java.io.File;
import java.util.List;

public class Mp3ScanUtils {
    private static final String TAG = "Mp3Utils";

    //Scan mp3 files recursively way
    public static void scanMp3File(List<MusicInfo> list, String filePath) throws Exception {
        File[] files = folderScan(filePath);
        if (files == null)
            return;
        File file;
        for (int i = 0; i < files.length; i++) {
            if (files[i].isHidden())
                continue;
            String fileAbsolutePath = files[i].getAbsolutePath();
            String fileName = files[i].getName();
            boolean isDirectory = false;
            if (files[i].isDirectory()) {
                isDirectory = true;
            }
            FileInfo fileInfo = new FileInfo(fileAbsolutePath, fileName,
                    isDirectory);
            if (fileInfo.isDirectory())
                scanMp3File(list, fileInfo.getFilePath());
            else if (fileInfo.isMUSICFile()) {
                MusicInfo music = new MusicInfo();
                music.setFilepath(fileInfo.getFilePath());
                music.setTitle(FileUtils.getFileTitle(fileInfo.getFileName()));
                music.setFilename(fileInfo.getFileName());
                music.setCoverPath(Mp3InfoUtils.getCoverPath(music.getFilepath(),
music.getTitle()));
                String path = fileInfo.getFilePath();
                file = new File(path);
                long size = FileUtils.getFileSizes(file);
                music.setFileSize(size);
                list.add(music);
```

```
            }
        }
    }

    private static File[] folderScan(String path) {
        File file = new File(path);
        File[] files = file.listFiles();
        return files;
    }
}
```

A.24 ScreenUtils.java

```
package com.example.fyp2_tsk.common.utils;

import android.content.Context;
import android.util.DisplayMetrics;
import android.view.WindowManager;

public class ScreenUtils {
    public static int[] getScreenSize(Context context) {
        WindowManager wm = (WindowManager) context.getSystemService(Context.WINDOW_SERVICE);
        DisplayMetrics outMetrics = new DisplayMetrics();
        wm.getDefaultDisplay().getMetrics(outMetrics);
        return new int[]{outMetrics.widthPixels, outMetrics.heightPixels};
    }
}
```

A.25 RingTools.java

```
package com.example.fyp2_tsk.common.utils;

import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Build;
import android.provider.MediaStore;
import android.provider.Settings;
import android.widget.Toast;
import com.orhanobut.logger.Logger;
import com.example.fyp2_tsk.R;

import java.io.File;

public class RingTools {

    /**
     * set ringtone
     * @param type RingtoneManager.TYPE_RINGTONE is phone ringtone
     *             RingtoneManager.TYPE_NOTIFICATION  is Notification ringtone
     *             RingtoneManager.TYPE_ALARM is alarm ringtone
     * @param path Download down the mp3 full path
     */
    public static void setRing(Activity context, int type, String path) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            // check whether have WRITE_SETTINGS permission or not
            if (!Settings.System.canWrite(context)) {
                Toast.makeText(context,
context.getResources().getString(R.string.homefragment_systemsetting_denied),
                        Toast.LENGTH_SHORT).show();
            } else {
                setRingHasPermission(context, type, path);
            }
        } else {
            setRingHasPermission(context, type, path);
        }
    }

    private static void setRingHasPermission(Activity context, int type, String path) {
        Uri oldRingtoneUri = RingtoneManager.getActualDefaultRingtoneUri(context,
```

```
RingtoneManager.TYPE_RINGTONE); //current phone call ringtone
        Uri oldNotification = RingtoneManager.getActualDefaultRingtoneUri(context,
RingtoneManager.TYPE_NOTIFICATION); //current notification ringtone
        Uri oldAlarm = RingtoneManager.getActualDefaultRingtoneUri(context,
RingtoneManager.TYPE_ALARM); //current alarm
        File sdfile = new File(path);
        ContentValues values = new ContentValues();
        values.put(MediaStore.MediaColumns.DATA, sdfile.getAbsolutePath());
        values.put(MediaStore.MediaColumns.TITLE, sdfile.getName());
        values.put(MediaStore.MediaColumns.MIME_TYPE, "audio/mp3");
        values.put(MediaStore.Audio.Media.IS_RINGTONE, true);
        values.put(MediaStore.Audio.Media.IS_NOTIFICATION, true);
        values.put(MediaStore.Audio.Media.IS_ALARM, true);
        values.put(MediaStore.Audio.Media.IS_MUSIC, true);

        Uri uri = MediaStore.Audio.Media.getContentUriForPath(sdfile.getAbsolutePath());
        Uri newUri = null;
        String deleteId = "";
        try {
            Cursor cursor = context.getContentResolver().query(uri, null,
MediaStore.MediaColumns.DATA + "=?", new String[]{path}, null);
            if (cursor.moveToFirst()) {
                deleteId = cursor.getString(cursor.getColumnIndex("_id"));
            }
            Logger.d(" + deleteId" + deleteId);

            context.getContentResolver().delete(uri,
                    MediaStore.MediaColumns.DATA + "=\"" + sdfile.getAbsolutePath() + "\"",
null);
            newUri = context.getContentResolver().insert(uri, values);
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (newUri != null) {
            String ringStoneId = "";
            String notificationId = "";
            String alarmId = "";
            if (null != oldRingtoneUri) {
                ringStoneId = oldRingtoneUri.getLastPathSegment();
            }
            if (null != oldNotification) {
                notificationId = oldNotification.getLastPathSegment();
            }
            if (null != oldAlarm) {
```

```
                alarmId = oldAlarm.getLastPathSegment();
            }
            Uri setRingStoneUri;
            Uri setNotificationUri;
            Uri setAlarmUri;
            if (type == RingtoneManager.TYPE_RINGTONE || ringStoneId.equals(deleteId)) {
                setRingStoneUri = newUri;
            } else {
                setRingStoneUri = oldRingtoneUri;
            }
            if (type == RingtoneManager.TYPE_NOTIFICATION ||
notificationId.equals(deleteId)) {
                setNotificationUri = newUri;
            } else {
                setNotificationUri = oldNotification;
            }
            if (type == RingtoneManager.TYPE_ALARM || alarmId.equals(deleteId)) {
                setAlarmUri = newUri;
            } else {
                setAlarmUri = oldAlarm;
            }
            RingtoneManager.setActualDefaultRingtoneUri(context,
RingtoneManager.TYPE_RINGTONE, setRingStoneUri);
            RingtoneManager.setActualDefaultRingtoneUri(context,
RingtoneManager.TYPE_NOTIFICATION, setNotificationUri);
            RingtoneManager.setActualDefaultRingtoneUri(context, RingtoneManager.TYPE_ALARM,
setAlarmUri);
            switch (type) {
                case RingtoneManager.TYPE_RINGTONE:
                    Toast.makeText(context.getApplicationContext(), "Ringer set up
successfully!", Toast.LENGTH_SHORT).show();
                    break;
                case RingtoneManager.TYPE_NOTIFICATION:
                    Toast.makeText(context.getApplicationContext(), "Set notification ring
successfully!", Toast.LENGTH_SHORT).show();
                    break;
                case RingtoneManager.TYPE_ALARM:
                    Toast.makeText(context.getApplicationContext(), "Alarm set
successfully!", Toast.LENGTH_SHORT).show();
                    break;
            }
        }
    }
}
```

A.26 Music.java

```
package com.example.fyp2_tsk.entity;

import android.graphics.Bitmap;

public class Music {

    public String title;
    public String artist;
    public String album;
    public int length;
    public Bitmap albumBip;
    public String path;
    public boolean isPlaying;

    public String getTitle() {
        return title;
    }
}
```

A.27 DetectSwipeGestureDriver

```
package com.example.fyp2_tsk.Gesture;

import android.view.GestureDetector;
import android.view.MotionEvent;

import com.example.fyp2_tsk.other.DriverMusicActivity;
import com.example.fyp2_tsk.other.MusicActivity;

public class DetectSwipeGestureDriver extends GestureDetector.SimpleOnGestureListener {
    private static int MIN_SWIPE_DISTANCE_X = 100;

    private static int MAX_SWIPE_DISTANCE_X = 1000;

    private DriverMusicActivity activity = null;

    public DriverMusicActivity getActivity() {
        return activity;
    }

    public void setActivity(DriverMusicActivity activity) {
        this.activity = activity;
    }
    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)
{
        float deltaX = e1.getX() - e2.getX();
        float deltaXAbs = Math.abs(deltaX);

        if (deltaXAbs >= MIN_SWIPE_DISTANCE_X && deltaXAbs <= MAX_SWIPE_DISTANCE_X) {
            if (deltaX > 0) {
                this.activity.gestureSwipeLeft();
            }
            else {
                this.activity.gestureSwipeRight();
            }
        }
        return true;
    }
}
```

A.28 DetectSwipeGestureListener

```
package com.example.fyp2_tsk.Gesture;

import android.view.GestureDetector;
import android.view.MotionEvent;
import com.example.fyp2_tsk.other.DriverMusicActivity;
import com.example.fyp2_tsk.other.MusicActivity;

public class DetectSwipeGestureListener extends GestureDetector.SimpleOnGestureListener {
    private static int MIN_SWIPE_DISTANCE_X = 100;
    private static int MAX_SWIPE_DISTANCE_X = 1000;

    private MusicActivity activity = null;

    public MusicActivity getActivity() {
        return activity;
    }

    public void setActivity(MusicActivity activity) {
        this.activity = activity;
    }

    @Override
    public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY)
{
        float deltaX = e1.getX() - e2.getX();
        float deltaXAbs = Math.abs(deltaX);

        if (deltaXAbs >= MIN_SWIPE_DISTANCE_X && deltaXAbs <= MAX_SWIPE_DISTANCE_X) {
            if (deltaX > 0) {
                this.activity.gestureSwipeLeft();
            } else {
                this.activity.gestureSwipeRight();
            }
        }
        return true;
    }
}
```

A.29 FileInfo.java

```
package com.example.fyp2_tsk.home.bean;

import java.util.ArrayList;

public class FileInfo {
    private FileType fileType;
    private String fileName;
    private String filePath;

    public FileInfo(String filePath, String fileName, boolean isDirectory) {
        this.filePath = filePath;
        this.fileName = fileName;
        fileType = isDirectory ? FileType.DIRECTORY : FileType.FILE;
    }

    public boolean isMUSICFile() {
        if (fileName.lastIndexOf(".") < 0) // Don't have the suffix
            return false;
        String fileSuffix = fileName.substring(fileName.lastIndexOf("."));
        if (!isDirectory() && MUSIC_SUFFIX.contains(fileSuffix))
            return true;
        else
            return false;
    }

    public boolean isDirectory() {
        if (fileType == FileType.DIRECTORY)
            return true;
        else
            return false;
    }

    public String getFileName() {
        return fileName;
    }

    public String getFilePath() {
        return filePath;
    }

    @Override
    public String toString() {
```

```
        return "FileInfo [fileType=" + fileType + ", fileName=" + fileName
                + ", filePath=" + filePath + "]";
    }

    private static ArrayList<String> MUSIC_SUFFIX = new ArrayList<String>();

    static {
        MUSIC_SUFFIX.add(".mp3");
    }

    enum FileType {
        FILE, DIRECTORY
    }
}
```

A.30 FileChooseComponent.java

```
import com.example.fyp2_tsk.common.app.di.ActivityScope;
import com.example.fyp2_tsk.common.app.di.AppComponent;
import com.example.fyp2_tsk.home.ui.FileChooserActivity;
import dagger.Component;

@ActivityScope
@Component(modules = FileChooseModule.class, dependencies = AppComponent.class)
public interface FileChooseComponent {
    void inject(FileChooserActivity activity);
}
```

A.31 HomeComponent.java

```
package com.example.fyp2_tsk.home.di;
import com.example.fyp2_tsk.common.app.di.ActivityScope;
import com.example.fyp2_tsk.common.app.di.AppComponent;
import com.example.fyp2_tsk.home.ui.CutFragment;
import dagger.Component;

@ActivityScope
@Component(modules = HomeModule.class, dependencies = AppComponent.class)
public interface HomeComponent {
    void inject(CutFragment fragment);
}
```

A.32 MusicInfo.java

```
package com.example.fyp2_tsk.home.bean;

import android.os.Parcel;
import android.os.Parcelable;
import io.objectbox.annotation.Entity;
import io.objectbox.annotation.Id;

@Entity
public class MusicInfo implements Parcelable, Cloneable {
    @Id()
    private Long id = null;
    private String filepath;
    private String filename;
    private long fileSize;
    private int storageType;
    private String coverPath;
    private String album;
    private String artist;
    private String title; //Music title

    public MusicInfo() {
    }

    public String getFilepath() {
        return this.filepath;
    }

    public void setFilepath(String filepath) {
        this.filepath = filepath;
    }

    public String getFilename() {
        return this.filename;
    }

    public void setFilename(String filename) {
        this.filename = filename;
    }

    public Long getId() {
        return this.id;
    }
```

```
public void setId(Long id) {
    this.id = id;
}

public long getFileSize() {
    return this.fileSize;
}

public void setFileSize(long fileSize) {
    this.fileSize = fileSize;
}

public int getStorageType() {
    return storageType;
}

public MusicInfo setStorageType(int storageType) {
    this.storageType = storageType;
    return this;
}

public interface Type {
    int LOCAL = 0;
}

public String getCoverPath() {
    return coverPath;
}

public MusicInfo setCoverPath(String coverPath) {
    this.coverPath = coverPath;
    return this;
}

public String getAlbum() {
    return album;
}

public MusicInfo setAlbum(String album) {
    this.album = album;
    return this;
}
```

```
public String getTitle() {
    return title;
}

public MusicInfo setTitle(String title) {
    this.title = title;
    return this;
}

public String getArtist() {
    return artist;
}

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

@Override
public int describeContents() {
    return 0;
}

@Override
public void writeToParcel(Parcel dest, int flags) {
    dest.writeValue(this.id);
    dest.writeString(this.filepath);
    dest.writeString(this.filename);
    dest.writeLong(this.fileSize);
    dest.writeInt(this.storageType);
    dest.writeString(this.coverPath);
    dest.writeString(this.album);
    dest.writeString(this.title);
    dest.writeString(this.artist);
}

protected MusicInfo(Parcel in) {
    this.id = (Long) in.readValue(Long.class.getClassLoader());
    this.filepath = in.readString();
    this.filename = in.readString();
    this.fileSize = in.readLong();
    this.storageType = in.readInt();
    this.coverPath = in.readString();
    this.album = in.readString();
```

```
        this.title = in.readString();
        this.artist = in.readString();
    }

    public static final Parcelable.Creator<MusicInfo> CREATOR = new
Parcelable.Creator<MusicInfo>() {
        @Override
        public MusicInfo createFromParcel(Parcel source) {
            return new MusicInfo(source);
        }

        @Override
        public MusicInfo[] newArray(int size) {
            return new MusicInfo[size];
        }
    };

    @Override
    public Object clone() throws CloneNotSupportedException {
        MusicInfo musicInfo = null;
        try{
            musicInfo = (MusicInfo) super.clone();
        }
        catch(CloneNotSupportedException e) {
            e.printStackTrace();
        }
        return musicInfo;
    }
}
```

A.33 HomeModule.java

```
package com.example.fyp2_tsk.home.di;

import com.example.fyp2_tsk.home.presenter.HomeContract;
import dagger.Module;
import dagger.Provides;

@Module
public class HomeModule {
    private HomeContract.View view;
    public HomeModule(HomeContract.View view){
        this.view = view;
    }
    @Provides
    public HomeContract.View providerHomeView(){
        return this.view;
    }
}
```

A.34 FileChooseModule.java

```
package com.example.fyp2_tsk.home.di;

import com.example.fyp2_tsk.home.presenter.FileChooseContract;
import dagger.Module;
import dagger.Provides;

@Module
public class FileChooseModule {
    private FileChooseContract.View view;
    public FileChooseModule(FileChooseContract.View view){
        this.view = view;
    }
    @Provides
    public FileChooseContract.View providerHomeView(){
        return this.view;
    }
}
```

A.35 FileChooseContract.java

```
package com.example.fyp2_tsk.home.presenter;

import com.example.fyp2_tsk.common.base.IBaseView;
import com.example.fyp2_tsk.home.bean.MusicInfo;
import java.util.List;

public class FileChooseContract {
    public interface View extends IBaseView {
        void getMusicList(List<MusicInfo> musiclist);
    }



    public interface Presenter{
        void loadFile(final boolean isforceScan);
    }
}
```

A.36 HomeContract.java

```
package com.example.fyp2_tsk.home.presenter;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.media.MediaPlayer;
import com.example.fyp2_tsk.common.base.IBaseView;

public class HomeContract {
    public interface View extends IBaseView {
        //Enable of disable spectrum
        void setVisualizerViewEnaled(boolean enabled);

        //Detect recording permission, spectrum
        void checkRecordPermission(MediaPlayer mediaPlayer);

        float getSeekBarAbsoluteMaxValue();

        //Gets the corresponding value for the seekbar min thumb slider
        int getSeekbarSelectedMinValue();

        //Gets the corresponding value for the seekbar Max thumb slider
        int getSeekbarSelectedMaxValue();

        //set the background of the play button according to the play status
        void setPlayBtnWithStatus(boolean isPlayingStatus);

        //Add spectrum rendering
        void addBarGraphRenderers();

        //Seekbar sets whether the seekbar slider is available (it can be slid)
        void setSeekBarEnable(boolean isEnable);

        //Sets the current playback value
        boolean setSeekBarProgressValue(int value, boolean isMin);

        // set max value of seekbar
        void setSeekBarMaxValue(int value);

        //trim successful
        void doCutterSucc(String path);
```

```
        //trim failed
        void doCutterFail();

        //set back to default of seekbar
        void resetSeekBarSelValue();
    }


    interface Presenter{
        //switch between play and pause
        void playToggle(Activity activity);

        void pause();

        void play(Activity activity);

        void onDestroy();

        void onActivityResult(int requestCode, int resultCode, Intent data);

        //Set the system sound value
        void setStreamVolume(int progress);

        //get the max sound value of system
        int getStreamMaxVolume();

        //get the min sound value of system
        int getStreamVolume();

        //do trim song task
        void doCutter(final String fileName, final long minValue, final long maxValue);

        //Check if mp3 is selected
        boolean isSelectedMp3(Context context);

        //Mediaplayer jumps to the position corresponding to the specified slider (min or
Max) based on the incoming parameter flag.
        void seekToForIsMin(boolean isMinBar);
    }
}
```

A.37 FileChoosePresenter.java

```
package com.example.fyp2_tsk.home.presenter;

import android.os.Environment;
import com.example.fyp2_tsk.common.app.MyApplication;
import com.example.fyp2_tsk.common.base.BasePresenter;
import com.example.fyp2_tsk.common.utils.Mp3ScanUtils;
import com.example.fyp2_tsk.home.bean.MusicInfo;
import java.util.ArrayList;
import java.util.List;
import javax.inject.Inject;
import io.objectbox.Box;
import io.reactivex.Observable;
import io.reactivex.ObservableEmitter;
import io.reactivex.ObservableOnSubscribe;
import io.reactivex.android.schedulers.AndroidSchedulers;
import io.reactivex.functions.Consumer;
import io.reactivex.schedulers.Schedulers;

public class FileChoosePresenter extends BasePresenter<FileChooseContract.View> implements
FileChooseContract.Presenter {
    private String mSdcardRootPath;
    Box<MusicInfo> mBox;
    @Inject
    public FileChoosePresenter(FileChooseContract.View view) {
        super(view);
        init();
    }

    private void init(){
        mBox = MyApplication.getInstances().getBoxStore().boxFor(MusicInfo.class);
        mSdcardRootPath = Environment.getExternalStorageDirectory()
                .getAbsolutePath();
    }


    //Force rescan and update the database, if the database has the data then update from
the existing database
    public void loadFile(final boolean isforceScan){
        Observable.create(new ObservableOnSubscribe() {
            @Override
            public void subscribe(ObservableEmitter e) throws Exception {
```

```
                List<MusicInfo> datas = mBox.getAll();
                if(datas==null||datas.size()<=0||isforceScan) {
                    datas = new ArrayList<>();
                }
                if(isforceScan) {
                    addDataToDB(datas);
                }
                else{
                    if(datas.size()<=0)
                        addDataToDB(datas);
                }
                e.onNext(datas);
            }
        }).observeOn(AndroidSchedulers.mainThread()).subscribeOn(Schedulers.newThread())
                .subscribe(new Consumer<List<MusicInfo>>() {
                    @Override
                    public void accept(List<MusicInfo> datas) throws Exception {
                        if(mView!=null)
                            mView.getMusicList(datas);
                    }
                });
    }

    private void addDataToDB(List<MusicInfo> datas) throws Exception {
        mBox.removeAll();
        Mp3ScanUtils.scanMp3File(datas, mSdcardRootPath);
        if(datas!=null) {
            mBox.put(datas);
        }
    }
}
```

A.38 HomePresenter

```
package com.example.fyp2_tsk.home.presenter;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.text.TextUtils;
import android.widget.Toast;

import com.example.fyp2_tsk.common.app.MyApplication;
import com.example.fyp2_tsk.common.base.BasePresenter;
import com.example.fyp2_tsk.common.constant.CommonConstant;
import com.example.fyp2_tsk.common.utils.FileUtils;
import com.example.fyp2_tsk.home.bean.MusicInfo;
import com.example.fyp2_tsk.home.ui.CutFragment;
import com.example.fyp2_tsk.home.ui.FileChooserActivity;
import com.example.fyp2_tsk.mp3cut.logic.Mp3CutLogic;
import com.example.fyp2_tsk.mp3cut.logic.Mp3InfoUtils;
import com.example.fyp2_tsk.mp3cut.util.Mp3NameConvertUtils;
import com.example.fyp2_tsk.R;

import java.io.File;
import java.io.IOException;
import java.util.concurrent.TimeUnit;

import javax.inject.Inject;

import io.reactivex.Observable;
import io.reactivex.ObservableEmitter;
import io.reactivex.ObservableOnSubscribe;
import io.reactivex.Observer;
import io.reactivex.android.schedulers.AndroidSchedulers;
import io.reactivex.disposables.Disposable;
import io.reactivex.functions.Consumer;
import io.reactivex.schedulers.Schedulers;

import static android.app.Activity.RESULT_CANCELED;
import static android.app.Activity.RESULT_OK;

public class HomePresenter extends BasePresenter<HomeContract.View> implements
HomeContract.Presenter {
```

```
    public MediaPlayer mMediaPlayer;
    private String mSelMp3Path = "";
    private MusicInfo mSelMp3Info;
    // Identifies the current playback slider as min or Max
    private boolean mIsMinFlag = true;
    // Gets the system audio object
    private AudioManager mAudioManager;

    @Inject
    public HomePresenter(HomeContract.View view) {
        super(view);
        init();
    }

    private void init() {
        mMediaPlayer = new MediaPlayer();
        mAudioManager = (AudioManager)
mView.getContext().getSystemService(Context.AUDIO_SERVICE);
    }

    private Disposable mDisposable;
    private Observable<Long> mUpdateProgressObservable = Observable.interval(0, 1,
TimeUnit.SECONDS);
    private Consumer mUpdateProgressConsumer = new Consumer() {
        @Override
        public void accept(Object o) throws Exception {
            if (mView == null)
                return;
            int curPosition = getCurPosition();
            Number maxValue = mView.getSeekBarAbsoluteMaxValue();

            if (!mView.setSeekBarProgressValue(curPosition, mIsMinFlag) || curPosition >=
maxValue
                    .intValue()) {
                pause();
            }
        }
    };


    //Cancel the update of the seekbar rx polling event
    private void cancelUpdateProgress() {
        if (mDisposable != null) {
            mDisposable.dispose();
```

```
            mDisposable = null;
        }
    }

    //trim music
    @Override
    public void doCutter(final String mp3Title, final long minValue, final long maxValue) {
        Observable.create(new ObservableOnSubscribe() {
            @Override
            public void subscribe(ObservableEmitter e) throws Exception {
                Mp3CutLogic helper = new Mp3CutLogic(new File(mSelMp3Path));
                if (FileUtils.bFolder(CommonConstant.RING_FOLDER)) {
                    if (!TextUtils.isEmpty(mp3Title)) {
                        String targetMp3FilePath = CommonConstant.RING_FOLDER + "/" +
mp3Title + ".mp3";

                        try {
                            helper.generateNewMp3ByTime(targetMp3FilePath, minValue,
maxValue);
                            addMp3ToDb(mp3Title, targetMp3FilePath);
                            e.onNext(targetMp3FilePath);
                        } catch (Exception e1) {
                            e.onError(e1);
                        }
                    }
                }
            }
        }).observeOn(AndroidSchedulers.mainThread()).subscribeOn(Schedulers.newThread())
                .subscribe(new Observer<String>() {
                    @Override
                    public void onSubscribe(Disposable d) {

                    }

                    @Override
                    public void onNext(String value) {
                        String cutterPath = value;
                        if (mView != null) {
                            mView.doCutterSucc(cutterPath);
                        }
                    }

                    @Override
                    public void onError(Throwable e) {
```

```
                        if (mView != null) {
                            mView.doCutterFail();
                        }
                    }

                    @Override
                    public void onComplete() {

                    }
                });
    }

    private void addMp3ToDb(String title, String cutterPath) throws
CloneNotSupportedException {
        File file = new File(cutterPath);
        MusicInfo info = new MusicInfo();
        info.setStorageType(MusicInfo.Type.LOCAL);
        info.setFilepath(cutterPath);
        info.setFilename(Mp3NameConvertUtils.titleConvertName(title));
        info.setTitle(title);
        info.setCoverPath(Mp3InfoUtils.getCoverPath(cutterPath, title));
        try {
            info.setFileSize(FileUtils.getFileSizes(file));
        } catch (Exception e) {
            e.printStackTrace();
        }
        MyApplication.getInstances().
                getBoxStore().boxFor(MusicInfo.class).put(info);
    }

    @Override
    public void playToggle(Activity activity) {
        if (isPlaying()) {
            pause();
        } else {
            play(activity);
        }

    }

    @Override
    public void pause() {
        if (isPlaying()) {
            mMediaPlayer.pause();
```

```
            mView.setPlayBtnWithStatus(false);
            mView.setVisualizerViewEnaled(false);
            cancelUpdateProgress();
        }
    }

    private void seekTo(int progress) {
        mMediaPlayer.seekTo(progress);
    }

    @Override
    public void play(Activity activity) {
        if (TextUtils.isEmpty(mSelMp3Path)) {
            Toast.makeText(activity, activity.getString(R.string.dialog_cutter_warning_sel),
Toast.LENGTH_SHORT).show();
            return;
        }
        mMediaPlayer.start();
        mView.setPlayBtnWithStatus(true);
        mView.setVisualizerViewEnaled(true);
        seekToForIsMin();
        //start the progress rx polling event
        mDisposable = mUpdateProgressObservable.observeOn(AndroidSchedulers.mainThread()).
                subscribe(mUpdateProgressConsumer);
    }

    private void reset() {
        mMediaPlayer.reset();
    }

    private void setDataSource(String path) {
        try {
            mMediaPlayer.setDataSource(path);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void prepare() {
        try {
            mMediaPlayer.prepare();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
    }

    private MediaPlayer getMediaPlayer() {
        return mMediaPlayer;
    }

    private int getDuration() {
        return mMediaPlayer.getDuration();
    }

    private boolean isPlaying() {
        return mMediaPlayer.isPlaying();
    }

    private int getCurPosition() {
        return mMediaPlayer.getCurrentPosition();
    }

    //check whether has select the mp3 or not
    @Override
    public boolean isSelectedMp3(Context context) {
        if (TextUtils.isEmpty(mSelMp3Path)) {
            Toast.makeText(context, context.getString(R.string.dialog_cutter_warning_sel),
Toast.LENGTH_SHORT).show();
            return false;
        }
        return true;
    }

    private void seekToForIsMin() {
        int curValue;
        if (mIsMinFlag) {
            curValue = mView.getSeekbarSelectedMinValue();
        } else {
            curValue = mView.getSeekbarSelectedMaxValue();
        }
        seekTo(curValue);
    }

    @Override
    public void seekToForIsMin(boolean isMinBar) {
        if (judgeIsPlayingThumb(isMinBar)) {
            int curValue;
            if (isMinBar) {
```

```
                curValue = mView.getSeekbarSelectedMinValue();
            } else {
                curValue = mView.getSeekbarSelectedMaxValue();
            }
            seekTo(curValue);
        }
    }

    /**
     * //determines whether the sliding slider is the currently playing slider
     * @param isMinBar Sliding slider true: min  false: max
     */

    private boolean judgeIsPlayingThumb(boolean isMinBar) {
        boolean isPlaying = false;
        if (mIsMinFlag) {
            if (isMinBar) {
                isPlaying = true;
            }
        } else {
            if (!isMinBar) {
                isPlaying = true;
            }
        }
        return isPlaying;
    }

//Select file then return
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        if (resultCode == RESULT_CANCELED) {
            return;
        }
        if (resultCode == RESULT_OK && requestCode == CutFragment.REQUEST_CODE) {
            mSelMp3Path = data
                    .getStringExtra(FileChooserActivity.EXTRA_FILEPATH_CHOOSER);
            mSelMp3Info = data
                    .getParcelableExtra(FileChooserActivity.EXTRA_FILE_CHOOSER);
            try {
                if (!TextUtils.isEmpty(mSelMp3Path)) {
                    mView.resetSeekBarSelValue();
                    mView.setPlayBtnWithStatus(false);
                    mView.setSeekBarEnable(true);
                    pause();
```

```
                    reset();
                    setDataSource(mSelMp3Path);
                    prepare();
                    mView.setSeekBarMaxValue(getDuration());
                    mView.checkRecordPermission(getMediaPlayer());
                    mView.addBarGraphRenderers();
                }
            } catch (IllegalArgumentException e) {   // Auto-generated catch block
                e.printStackTrace();
            } catch (SecurityException e) {
                // Auto-generated catch block
                e.printStackTrace();
            } catch (IllegalStateException e) {// Auto-generated catch block
                e.printStackTrace();
            }
        }
    }

    @Override
    public void onDestroy() {
        if (mDisposable != null) {
            mDisposable.dispose();
            mDisposable = null;
        }
    }

    @Override
    public void setStreamVolume(int progress) {
        mAudioManager.setStreamVolume(AudioManager.STREAM_MUSIC progress, 0);
    }

    @Override
    public int getStreamMaxVolume() {
        return mAudioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
    }

    @Override
    public int getStreamVolume() {
        return mAudioManager
                .getStreamVolume(AudioManager.STREAM_MUSIC);
    }
}
```

A.39 CutFragment.java

```
package com.example.fyp2_tsk.home.ui;

import android.Manifest;
import android.app.ActivityOptions;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.Paint;
import android.media.MediaPlayer;
import android.media.RingtoneManager;
import android.os.Build;
import android.os.Environment;
import android.preference.PreferenceManager;
import android.support.annotation.NonNull;
import android.view.View;
import android.widget.SeekBar;
import android.widget.Toast;
import com.example.fyp2_tsk.common.app.di.AppComponent;
import com.example.fyp2_tsk.common.base.BaseFragment;
import com.example.fyp2_tsk.common.constant.CommonConstant;
import com.example.fyp2_tsk.common.ui.view.CommonDialog;
import com.example.fyp2_tsk.common.ui.view.visualizer.renderer.CircleBarRenderer;
import com.example.fyp2_tsk.common.utils.FileUtils;
import com.example.fyp2_tsk.common.utils.RingTools;
import com.example.fyp2_tsk.databinding.FragmentCutBinding;
import com.example.fyp2_tsk.home.di.DaggerHomeComponent;
import com.example.fyp2_tsk.home.di.HomeModule;
import com.example.fyp2_tsk.home.presenter.HomeContract;
import com.zyl.customrangeseekbar.CustomRangeSeekBar;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.home.presenter.HomePresenter;
import java.util.concurrent.TimeUnit;
import io.reactivex.Observable;
import io.reactivex.android.schedulers.AndroidSchedulers;
import io.reactivex.functions.Consumer;
import permissions.dispatcher.NeedsPermission;
import permissions.dispatcher.OnNeverAskAgain;
import permissions.dispatcher.OnPermissionDenied;
import permissions.dispatcher.OnShowRationale;
import permissions.dispatcher.PermissionRequest;
```

```
import permissions.dispatcher.RuntimePermissions;

@RuntimePermissions
public class CutFragment extends BaseFragment<HomeContract.View, HomePresenter,
FragmentCutBinding> implements HomeContract.View, View.OnClickListener {
    public static final int REQUEST_CODE = 1010;
    private ProgressDialog mProgressDialog;
    private SharedPreferences mPreferences;
    private SharedPreferences.Editor mEditor;
    String status;
    // Music slider event
    private CustomRangeSeekBar.ThumbListener mThumbListener = new
CustomRangeSeekBar.ThumbListener() {

        @Override
        public void onClickMinThumb(Number max, Number min) {
        }

        @Override
        public void onClickMaxThumb() {
        }

        @Override
        public void onMinMove(Number max, Number min) {
            mPresenter.seekToForIsMin(true);
        }

        @Override
        public void onMaxMove(Number max, Number min) {
            mPresenter.seekToForIsMin(false);
        }

        @Override
        public void onUpMinThumb(Number max, Number min) {
        }

        @Override
        public void onUpMaxThumb() {
        }
    };

    //Sound slider slide event
    private SeekBar.OnSeekBarChangeListener mVoiceChangeListener = new
SeekBar.OnSeekBarChangeListener() {
```

```
        public void onStopTrackingTouch(SeekBar seekBar) {
        }

        public void onStartTrackingTouch(SeekBar seekBar) {
        }

        public void onProgressChanged(SeekBar seekBar, int progress,
                                      boolean fromUser) {
            if (seekBar.getId() == R.id.voice_seekbar) {
                mPresenter.setStreamVolume(progress);
                mDataBinding.visualView.invalidate();
            }
        }
    };

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


    // Display trim successful dialog window
    private void showCutterSuccessDialog(final String path) {
        String format =
String.format(getResources().getString(R.string.homefragment_cut_success),
CommonConstant.RING_FOLDER);
        new CommonDialog.Builder().setContext(getActivity()).setContentStr(format)
                .setOnDialogListener(new CommonDialog.OnDialogClickListener() {
                    @Override
                    public void doOk() {

CutFragmentPermissionsDispatcher.setRingWithPermissionCheck(CutFragment.this, path);
                    }
                })
                .build().show();
    }

    @NeedsPermission(Manifest.permission.WRITE_SETTINGS)
    public void setRing(final String path) {
        if (FileUtils.bFolder(CommonConstant.RING_FOLDER)) {
            RingTools.setRing(getActivity(), RingtoneManager.TYPE_RINGTONE, path);
        }
    }
```

```
@Override
public void onDetach() {
    super.onDetach();
}

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

@Override
protected void ComponentInject(AppComponent appComponent) {
    DaggerHomeComponent
            .builder()
            .appComponent(appComponent)
            .homeModule(new HomeModule(this))
            .build()
            .inject(this);
}

@Override
protected void init(View view) {
    mPreferences = PreferenceManager.getDefaultSharedPreferences(getActivity());
    mEditor = mPreferences.edit();
    init();
    initListener();
}

@Override
protected int initLayoutResId() {
    return R.layout.fragment_cut;
}

private void init() {
    mDataBinding.voiceSeekbar.setMax(mPresenter.getStreamMaxVolume());
    mDataBinding.voiceSeekbar.setProgress(mPresenter.getStreamVolume());
}

private void initListener() {
    mDataBinding.btnPlay.setOnClickListener(this);
    mDataBinding.btnCutterSure.setOnClickListener(this);
    mDataBinding.rangeSeekbar.setThumbListener(mThumbListener);
```

```
        mDataBinding.rangeSeekbar.setEnabled(false);
        mDataBinding.voiceSeekbar.setOnSeekBarChangeListener(mVoiceChangeListener);
    }

    @Override
    public void setVisualizerViewEnaled(boolean enabled) {
        mDataBinding.visualView.setEnabled(enabled);
    }

    @Override
    public void checkRecordPermission(MediaPlayer mediaPlayer) {

CutFragmentPermissionsDispatcher.linkMediaPlayerForVisualViewWithPermissionCheck(this,
mediaPlayer);
    }

    @Override
    public int getSeekbarSelectedMaxValue() {
        Number number = mDataBinding.rangeSeekbar.getSelectedAbsoluteMaxValue();
        return number.intValue();
    }

    @Override
    public float getSeekBarAbsoluteMaxValue() {
        return mDataBinding.rangeSeekbar.getAbsoluteMaxValue();
    }

    @Override
    public int getSeekbarSelectedMinValue() {
        Number number = mDataBinding.rangeSeekbar.getSelectedAbsoluteMinValue();
        return number.intValue();
    }

    @Override
    public void setPlayBtnWithStatus(boolean isPlayingStatus) {
        if (isPlayingStatus) {
            mDataBinding.btnPlay.setBackgroundResource(R.drawable.selector_pause_btn);
        } else {
            mDataBinding.btnPlay.setBackgroundResource(R.drawable.selector_play_btn);
        }
    }

    public void setSeekBarEnable(boolean isClickable) {
        mDataBinding.rangeSeekbar.setEnabled(isClickable);
```

```
    }

    @NeedsPermission(Manifest.permission.RECORD_AUDIO)
    public void linkMediaPlayerForVisualView(MediaPlayer player) {
        mDataBinding.visualView.link(player);
    }

    //Add spectrum rendering
    @Override
    public void addBarGraphRenderers() {
        Paint paint2 = new Paint();
        paint2.setStrokeWidth(12f);
        paint2.setAntiAlias(true);
        paint2.setColor(Color.argb(240, 172, 175, 64));
        CircleBarRenderer barGraphRendererTop = new CircleBarRenderer(paint2, 4);
        mDataBinding.visualView.clearRenderers();
        mDataBinding.visualView.addRenderer(barGraphRendererTop);
    }

    @Override
    public boolean setSeekBarProgressValue(int value, boolean isMin) {
        if (isMin) {
            return mDataBinding.rangeSeekbar.setSelectedAbsoluteMinValue(value);
        } else {
            return mDataBinding.rangeSeekbar.setSelectedAbsoluteMaxValue(value);
        }
    }

    //reset seekbar initial value
    @Override
    public void resetSeekBarSelValue() {
        mDataBinding.rangeSeekbar.restorePercentSelectedMinValue();
        mDataBinding.rangeSeekbar.restorePercentSelectedMaxValue();
    }

    @Override
    public void setSeekBarMaxValue(int value) {
        mDataBinding.rangeSeekbar.setAbsoluteMaxValue(value);
    }

    @Override
    public void doCutterSucc(final String path) {
        Observable.timer(2000, TimeUnit.MILLISECONDS)
                .observeOn(AndroidSchedulers.mainThread())
```

```
                .subscribe(new Consumer<Long>() {
                    @Override
                    public void accept(Long aLong) throws Exception {
                        if (mProgressDialog != null && getActivity() != null
&& !getActivity().isFinishing()) {
                            mProgressDialog.dismiss();
                            showCutterSuccessDialog(path);
                        }
                    }
                });

    }

    @Override
    public void doCutterFail() {
        if (mProgressDialog != null && getActivity() != null
&& !getActivity().isFinishing()) {
            mProgressDialog.dismiss();
        }
        Toast.makeText(getActivity(),
getResources().getString(R.string.homefragment_cut_fail), Toast.LENGTH_LONG).show();
    }

    /**
     * Prompt out dialog box while trimming audio
     */
    public void showCutterPromptDialog() {
        final Number minNumber = mDataBinding.rangeSeekbar.getSelectedAbsoluteMinValue();
        final Number maxNumber = mDataBinding.rangeSeekbar.getSelectedAbsoluteMaxValue();
        if (maxNumber.intValue() <= minNumber.longValue()) {
            Toast.makeText(getActivity(),
                    getString(R.string.dialog_cutter_warning_length),
                    Toast.LENGTH_LONG).show();
            return;
        }
        mPresenter.pause();
        new
CommonDialog.Builder().setContext(getActivity()).setContentStr(getString(R.string.dialog_cut
ter_msg))
                .setIsShowInput(true)
                .setOnDialogListener(new CommonDialog.OnDialogClickListener() {
                    @Override
                    public void doOk(String filename) {
                        mProgressDialog = ProgressDialog.show(getActivity(),
```

```
getResources().getString(R.string.homefragment_cutting_tip),
                                getResources().getString(R.string.homefragment_cutting));
                        mPresenter.doCutter(filename, minNumber.longValue(),
                                maxNumber.longValue());
                    }
                })
                .build().show();
    }

    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        mPresenter.onActivityResult(requestCode, resultCode, data);
    }

    @Override
    public void onStop() {
        super.onStop();
        mPresenter.pause();
        setPlayBtnWithStatus(false);
    }

    @OnShowRationale(Manifest.permission.RECORD_AUDIO)
    void showRationaleForRecord(final PermissionRequest request) {
        new CommonDialog.Builder().setContext(getActivity()).setContentStr(
                getResources().getString(R.string.homefragment_permission_prompt))
                .setOnDialogListener(new CommonDialog.OnDialogClickListener() {
                    @Override
                    public void doOk() {
                        request.proceed();
                    }
                }).setIsShowOne(true).build().show();
    }

    @OnPermissionDenied(Manifest.permission.RECORD_AUDIO)
    void showRecordDenied() {
        Toast.makeText(getActivity(),
getResources().getString(R.string.homefragment_permission_denied),
                Toast.LENGTH_LONG).show();
    }

    @OnNeverAskAgain(Manifest.permission.RECORD_AUDIO)
    void onRecordNeverAskAgain() {
    }

    @Override
```

```
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        // NOTE: delegate the permission handling to generated method
        CutFragmentPermissionsDispatcher.onRequestPermissionsResult(this, requestCode,
grantResults);
    }

    @Override
    public boolean shouldShowRequestPermissionRationale(@NonNull String permission) {
        return super.shouldShowRequestPermissionRationale(permission);
    }

    @Override
    public void onClick(View v) {
        int id = v.getId();
        switch (id) {
            case R.id.btn_play:
                mPresenter.playToggle(getActivity());
                break;
            case R.id.btn_cutter_sure:
                if (mPresenter.isSelectedMp3(getActivity()))
                    showCutterPromptDialog();
                break;
            default:
                break;
        }
    }

    public void openFile() {
        Intent intent = new Intent(getActivity(), FileChooserActivity.class);

        if (getActivity() == null || getActivity().isFinishing()) {
            return;
        }
        if (Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                startActivityForResult(intent, REQUEST_CODE,
ActivityOptions.makeSceneTransitionAnimation(getActivity()).toBundle());
            } else {
                startActivityForResult(intent, REQUEST_CODE);
            }
        } else {
            Toast.makeText(getActivity(),
```

```
                    R.string.sdcard_unmonted_hint, Toast.LENGTH_SHORT).show();
        }
    }

    public void updateUI(String status) {
        if (status.equals("true")) {
            mDataBinding.visualView.setBackgroundResource(R.color.cutter_theme_black);
            mDataBinding.rlMain.setBackgroundResource(R.color.cutter_theme_black);
        } else {
            mDataBinding.visualView.setBackgroundResource(R.color.white_main_bg);
            mDataBinding.rlMain.setBackgroundResource(R.color.white_main_bg);
        }
    }

    @Override
    public void onResume() {
        super.onResume();
        status = mPreferences.getString(getString(R.string.statusNightMode), "false");
        if (status.equals("true"))
            updateUI(status);
        else
            updateUI(status);
    }
}
```

A.40 FileChooserActivity.java

```
package com.example.fyp2_tsk.home.ui;

import android.Manifest;
import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.content.Intent;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.v7.widget.DividerItemDecoration;
import android.support.v7.widget.LinearLayoutManager;
import android.text.TextUtils;
import android.transition.TransitionInflater;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageView;
import android.widget.Toast;
import com.bumptech.glide.Glide;
import com.bumptech.glide.request.RequestOptions;
import com.example.fyp2_tsk.common.app.di.AppComponent;
import com.example.fyp2_tsk.common.base.BaseActivity;
import com.example.fyp2_tsk.common.utils.DensityUtils;
import com.example.fyp2_tsk.common.utils.FileUtils;
import com.example.fyp2_tsk.common.utils.ScreenUtils;
import com.example.fyp2_tsk.databinding.ActivityFilechooserShowBinding;
import com.example.fyp2_tsk.home.di.DaggerFileChooseComponent;
import com.example.fyp2_tsk.home.presenter.FileChoosePresenter;
import com.jaeger.library.StatusBarUtil;
import com.zhy.adapter.recyclerview.CommonAdapter;
import com.zhy.adapter.recyclerview.base.ViewHolder;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.home.bean.MusicInfo;
import com.example.fyp2_tsk.home.di.FileChooseModule;
import com.example.fyp2_tsk.home.presenter.FileChooseContract;
import java.util.ArrayList;
import java.util.List;
import permissions.dispatcher.NeedsPermission;
import permissions.dispatcher.OnNeverAskAgain;
import permissions.dispatcher.RuntimePermissions;
import skin.support.widget.SkinCompatToolbar;
```

```
@RuntimePermissions
public class FileChooserActivity extends BaseActivity<FileChooseContract.View,
FileChoosePresenter, ActivityFilechooserShowBinding> implements OnClickListener,
FileChooseContract.View{
    private CommonAdapter mAdapter;
    private ArrayList<MusicInfo> mMusicList = new ArrayList<>();
    public static final String EXTRA_FILEPATH_CHOOSER = "filepath_chooser";
    public static final String EXTRA_FILE_CHOOSER = "file_chooser";
    private ObjectAnimator mMoveAnim;
    private int mUpdateBtnLeft;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            android.transition.Transition transition =
TransitionInflater.from(this).inflateTransition(R.transition.explode);
            getWindow().setEnterTransition(transition);
        }

        StatusBarUtil.setColor(this, Color.TRANSPARENT);
        mDataBinding.btnUpdate.setOnClickListener(this);
        mDataBinding.btnUpdate.measure(0, 0);
        mUpdateBtnLeft = ScreenUtils.getScreenSize(this)[0] -
                mDataBinding.btnUpdate.getMeasuredWidth() - DensityUtils.dp2px(this ,10);
        initToolbar();

        mDataBinding.rlMusice.setLayoutManager(new LinearLayoutManager(this));
        mAdapter = new CommonAdapter<MusicInfo>(this, R.layout.item_musicfile, mMusicList) {
            @Override
            protected void convert(ViewHolder holder, final MusicInfo musicInfo, int
position) {
                holder.setText(R.id.tv_name, musicInfo.getTitle());
                holder.setText(R.id.tv_size,
FileUtils.formetFileSize(musicInfo.getFileSize()));
                if(TextUtils.isEmpty(musicInfo.getCoverPath())){
                    holder.setImageDrawable(R.id.iv_icon,
getResources().getDrawable(R.mipmap.music_icon));
                }
                else {
                    RequestOptions options = new
RequestOptions().placeholder(R.mipmap.music_icon);
                    Glide.with(FileChooserActivity.this).load(musicInfo.getCoverPath())
                            .apply(options).into((ImageView) holder.getView(R.id.iv_icon));
```

```
                }

                holder.itemView.setOnClickListener(new OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        clickItem(musicInfo);
                    }
                });
            }
        };
        mDataBinding.rlMusice.setAdapter(mAdapter);
        mDataBinding.rlMusice.addItemDecoration(new DividerItemDecoration(this,
DividerItemDecoration.VERTICAL));
        FileChooserActivityPermissionsDispatcher.refreshDataWithPermissionCheck(this,
false);
    }

    @Override
    protected void ComponentInject(AppComponent appComponent) {
        DaggerFileChooseComponent
                .builder()
                .appComponent(appComponent)
                .fileChooseModule(new FileChooseModule(this))
                .build()
                .inject(this);
    }

    @Override
    protected int initLayoutResId() {
        return R.layout.activity_filechooser_show;
    }

    @Override
    protected void initData(Bundle savedInstanceState) {

    }

    private void clickItem(MusicInfo info) {
        Intent intent = new Intent();
        intent.putExtra(EXTRA_FILEPATH_CHOOSER, info.getFilepath());
        intent.putExtra(EXTRA_FILE_CHOOSER, info);
        setResult(RESULT_OK, intent);
        finish();
    }
```

```
    private void initToolbar() {
        SkinCompatToolbar toolbar = (SkinCompatToolbar) findViewById(R.id.toolbar);
        toolbar.setBackgroundResource(R.color.theme_color);
        setSupportActionBar(toolbar);
        toolbar.setNavigationOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
    }

    @NeedsPermission({Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.WRITE_EXTERNAL_STORAGE})
    void refreshData(final boolean isforce) {
        if (mDataBinding.aviLoading.isShown())
            return;
        startLoadingAnim();
        mPresenter.loadFile(isforce);
    }

    private void startLoadingAnim() {
        mDataBinding.aviLoading.setVisibility(View.VISIBLE);
        mMoveAnim  = ObjectAnimator.ofFloat(mDataBinding.aviLoading, "translationX", 0f,
mUpdateBtnLeft);
        mMoveAnim.setDuration(2000);
        mMoveAnim.setRepeatCount(ValueAnimator.INFINITE);
        mMoveAnim.start();
    }

    private void stopLoadingAnim(){
        mMoveAnim.cancel();
        mDataBinding.aviLoading.setVisibility(View.GONE);
    }

    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (event.getAction() == KeyEvent.ACTION_DOWN
                && event.getKeyCode() == KeyEvent.KEYCODE_BACK) {
            backProcess();
            return true;
        }
        return super.onKeyDown(keyCode, event);
    }
```

```
    public void backProcess() {
        setResult(RESULT_CANCELED);
        finish();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    @Override
    public void onClick(View v) {
        int id = v.getId();
        switch (id) {
            case R.id.btn_update:

FileChooserActivityPermissionsDispatcher.refreshDataWithPermissionCheck(this, true);
                break;
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        FileChooserActivityPermissionsDispatcher.onRequestPermissionsResult(this,
requestCode, grantResults);
    }

    @OnNeverAskAgain({Manifest.permission.READ_EXTERNAL_STORAGE,
            Manifest.permission.WRITE_EXTERNAL_STORAGE})
    void onRecordNeverAskAgain() {
        Toast.makeText(FileChooserActivity.this,
getResources().getString(R.string.filechoose_permission_denied),
                Toast.LENGTH_SHORT).show();
    }

    @Override
    public void getMusicList(List<MusicInfo> musiclist) {
        mMusicList.clear();
        mMusicList.addAll(musiclist);
        stopLoadingAnim();
        if (mAdapter != null) {
```

```
            mAdapter.notifyDataSetChanged();
        }
    }
}
```

A.41 Mp3NameConvertUtils.java

```
package com.example.fyp2_tsk.mp3cut.util;

public class Mp3NameConvertUtils {
    private static final String SUFFIX_LOWER = ".mp3";
    public static String titleConvertName(String title){
        if(title==null)
            return null;
        return title.concat(SUFFIX_LOWER);
    }
}
```

A.42 Mp3InfoUtils.java

```
package com.example.fyp2_tsk.mp3cut.logic;

import com.example.fyp2_tsk.common.app.MyApplication;
import com.example.fyp2_tsk.common.utils.Md5Utils;
import org.jaudiotagger.audio.mp3.MP3File;
import org.jaudiotagger.tag.id3.AbstractID3v2Frame;
import org.jaudiotagger.tag.id3.AbstractID3v2Tag;
import org.jaudiotagger.tag.id3.framebody.FrameBodyAPIC;
import java.io.File;
import java.io.FileOutputStream;

public class Mp3InfoUtils {

    public static String getCoverPath(String path, String title){
        return saveMP3Image(new File(path), title,
MyApplication.instances.getExternalCacheDir().getAbsolutePath(),
                false);
    }

    //Get mp3 picture and save it to the specified path
    public static String saveMP3Image(File mp3File, String title, String mp3ImageSavePath,
boolean cover) {
        // Generate mp3 picture path
        String mp3ImageFullPath = mp3ImageSavePath + "/" + (Md5Utils.md5(title) + ".jpg");

        //If it is in non-overlay mode, the image will be returned directly (no longer
created)
        if (!cover) {
            File tempFile = new File(mp3ImageFullPath);
            if (tempFile.exists()) {
                return mp3ImageFullPath;
            }
        }

        //Generate mp3 storage directory
        File saveDirectory = new File(mp3ImageSavePath);
        saveDirectory.mkdirs();

        //Get mp3 picture
        byte imageData[] = getMP3Image(mp3File);
        //If the picture does not exist, it returns null directly
        if (null == imageData || imageData.length == 0) {
```

```
            return null;
        }
        //Save mp3 picture file
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(mp3ImageFullPath);
            fos.write(imageData);
            fos.close();
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
        return mp3ImageFullPath;
    }

    public static byte[] getMP3Image(File mp3File) {
        byte[] imageData = null;
        try {
            MP3File mp3file = new MP3File(mp3File);
            AbstractID3v2Tag tag = mp3file.getID3v2Tag();
            AbstractID3v2Frame frame = (AbstractID3v2Frame) tag.getFrame("APIC");
            FrameBodyAPIC body = (FrameBodyAPIC) frame.getBody();
            imageData = body.getImageData();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return imageData;
    }
}
```

A.43 Mp3CutLogic.java

```
package com.example.fyp2_tsk.mp3cut.logic;

import com.example.fyp2_tsk.common.utils.FileUtils;
import org.jaudiotagger.audio.mp3.MP3AudioHeader;
import org.jaudiotagger.audio.mp3.MP3File;
import java.io.File;
import java.io.IOException;
import java.io.RandomAccessFile;

public class Mp3CutLogic {
    private static final int BUFFER_SIZE = 1024 * 1024;
    private File mSourceMp3File;

    public Mp3CutLogic(File mp3File) {
        this.mSourceMp3File = mp3File;
    }

    //Generate new mp3 files according to time (support VBR and CBR respectively)
    public void generateNewMp3ByTime(String targetFileStr, long beginTime, long endTime)
throws Exception {
        MP3File mp3 = new MP3File(this.mSourceMp3File);
        MP3AudioHeader header = (MP3AudioHeader) mp3.getAudioHeader();
        if (header.isVariableBitRate()) {
            generateMp3ByTimeAndVBR(header, targetFileStr, beginTime, endTime);
        } else {
            generateMp3ByTimeAndCBR(header, targetFileStr, beginTime, endTime);
        }
    }

    //Generate MP3 files based on time and source files (source file mp3 bit rate is vbr
variable bit rate)
    private void generateMp3ByTimeAndVBR(MP3AudioHeader header, String targetFileStr, long
beginTime, long endTime) throws IOException {
        long frameCount = header.getNumberOfFrames();
        int sampleRate = header.getSampleRateAsNumber();
        int sampleCount = 1152;//header.getNoOfSample();
        int paddingLength = header.isPadding() ? 1 : 0;

        //Frame size = (number of samples per frame × bit rate (bit / s) ÷ 8 ÷ sampling
rate) + Padding
        //getBitRateAsNumber The returned is kbps, so * 1000
        float frameSize = sampleCount * header.getBitRateAsNumber() / 8f / sampleRate * 1000
```

```
+ paddingLength;

        //Get audio track duration
        int trackLengthMs = header.getTrackLength() * 1000;

        // Ratio of start time to total time
        float beginRatio = (float) beginTime / (float) trackLengthMs;

        //Ratio of end time to total time
        float endRatio = (float) endTime / (float) trackLengthMs;
        long startFrameSize = (long) (beginRatio * frameCount * frameSize);
        long endFrameSize = (long) (endRatio * frameCount * frameSize);

        //Returns the first byte of music data
        long firstFrameByte = header.getMp3StartByte();
        generateTargetMp3File(targetFileStr, startFrameSize, endFrameSize, firstFrameByte);
    }

    //Generate MP3 files based on time and source files (source file mp3 bit rate is cbr
constant bit rate)
    private void generateMp3ByTimeAndCBR(MP3AudioHeader header, String targetFileStr, long
beginTime, long endTime) throws IOException {
        //Get audio track duration
        int trackLengthMs = header.getTrackLength() * 1000;
        long bitRateKbps = header.getBitRateAsNumber();

        //1KByte/s=8Kbps, bitRate *1024L / 8L / 1000L Convert to bps per millisecond
        //Calculate the start byte position
        long beginBitRateBpm = convertKbpsToBpm(bitRateKbps) * beginTime;

        //Returns the first byte of music data
        long firstFrameByte = header.getMp3StartByte();

        //Get the byte position of the file where the start time is
        long beginByte = firstFrameByte + beginBitRateBpm;

        //Calculate the end byte position
        long endByte = beginByte + convertKbpsToBpm(bitRateKbps) * (endTime - beginTime);
        if (endTime > trackLengthMs) {
            endByte = this.mSourceMp3File.length() - 1L;
        }
        generateTargetMp3File(targetFileStr, beginByte, endByte, firstFrameByte);
    }
```

```
    private void generateTargetMp3File(String targetFileStr, long beginByte, long endByte,
long firstFrameByte) throws IOException {
        File file = new File(targetFileStr);
        //Delete if it exists
        FileUtils.checkAndDelFile(file);
        RandomAccessFile targetMp3File = null;
        RandomAccessFile sourceFile = null;
        try {
            targetMp3File = new RandomAccessFile(targetFileStr, "rw");
            sourceFile = new RandomAccessFile(mSourceMp3File, "rw");
            //write mp3 header info
            writeSourceToTargetFileWithBuffer(targetMp3File, sourceFile, firstFrameByte, 0);
            //write mp3 frame info
            int size = (int) (endByte - beginByte);
            writeSourceToTargetFileWithBuffer(targetMp3File, sourceFile, size, beginByte);
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            if (sourceFile != null)
                sourceFile.close();
            if (targetMp3File != null)
                targetMp3File.close();
        }
    }

    //KBPS kilobytes per second convert into BPM bytes per millisecond
    private long convertKbpsToBpm(long bitRate) {
        return bitRate * 1024L / 8L / 1000L;
    }

    //Write source file to target file in cache according to file and size
    private static void writeSourceToTargetFileWithBuffer(RandomAccessFile targetFile,
RandomAccessFile sourceFile, long totalSize, long offset) throws Exception {
        //buffer size, each time the specified data is written to prevent memory leaks
        int buffersize = BUFFER_SIZE;
        long count = totalSize / buffersize;
        if (count <= 1) {
            //The total file length is less than the buffer size
            writeSourceToTargetFile(targetFile, sourceFile, new byte[(int) totalSize],
offset);
        } else {
            // Size remaining after writing count
            long remainSize = totalSize % buffersize;
            byte data[] = new byte[buffersize];
```

```
            //The offset of seek when a file is read in
            for (int i = 0; i < count; i++) {
                writeSourceToTargetFile(targetFile, sourceFile, data, offset);
                offset += BUFFER_SIZE;
            }
            if (remainSize > 0) {
                writeSourceToTargetFile(targetFile, sourceFile, new byte[(int) remainSize],
offset);
            }
        }
    }

    //Writes to the target file based on the file and size
    private static void writeSourceToTargetFile(RandomAccessFile targetFile,
RandomAccessFile sourceFile, byte data[], long offset) throws Exception {
        sourceFile.seek(offset);
        sourceFile.read(data);
        long fileLength = targetFile.length();
        // Moves the write file pointer to the end of the file.
        targetFile.seek(fileLength);
        targetFile.write(data);
    }
}
```

A.44 AllFragment.java

```
package com.example.fyp2_tsk.other;

import android.Manifest;
import android.content.ContentResolver;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Build;
import android.provider.MediaStore;
import android.support.annotation.NonNull;
import android.support.annotation.RequiresApi;
import android.support.v4.app.Fragment;
import android.os.Bundle;
import android.text.TextUtils;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.ListView;
import android.widget.SearchView;
import android.widget.Toast;
import com.lwkandroid.widget.indexbar.IndexBar;
import com.example.fyp2_tsk.utils.Common;
import com.master.permissionhelper.PermissionHelper;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.adapter.MusicAdapter;
import com.example.fyp2_tsk.entity.Music;
import java.util.List;

public class AllFragment extends Fragment {
    private String TAG = "HelloActivity";
    private PermissionHelper permissionHelper;
    private ListView listView;
    private List<Music> musicList;
    private MusicAdapter adapter;
    private SearchView svSearch;
    private LinearLayout linearLayout;
```

```
    public AllFragment() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_all, container, false);
//Create a View object and return to the view

        // ask for permission
        permissionHelper = new PermissionHelper(this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, 100);
        permissionHelper.request(new PermissionHelper.PermissionCallback() {
            @Override
            public void onPermissionGranted() {
                initListView();                                    //Scan the info
from database after get permission
                Log.d(TAG, "onPermissionGranted() called");
            }

            @Override
            public void onIndividualPermissionGranted(String[] grantedPermission) {
                Log.d(TAG, "onIndividualPermissionGranted() called with: grantedPermission =
[" + TextUtils.join(",", grantedPermission) + "]");
            }

            @Override
            public void onPermissionDenied() {
                Log.d(TAG, "onPermissionDenied() called");
            }

            @Override
            public void onPermissionDeniedBySystem() {
                Log.d(TAG, "onPermissionDeniedBySystem() called");
            }
        });

        listView = (ListView) view.findViewById(R.id.logic_lv);
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @RequiresApi(api = Build.VERSION_CODES.JELLY_BEAN)
            @Override
            public void onItemClick(AdapterView<?> parent, View view, int position, long id)
{
```

```
                for (Music m : Common.musicList) {
                    m.isPlaying = false;
                }
                Common.musicList.get(position).isPlaying = true;

                //update UI
                adapter.notifyDataSetChanged();

                Intent intent = new Intent(getActivity(), MusicActivity.class);
                //transfer position to music activity
                intent.putExtra("position", position); //get position so can play right song
                startActivity(intent);
            }
        });

        adapter = new MusicAdapter(getActivity(), musicList);
        listView.setAdapter(adapter);

        //filter
        svSearch = (SearchView) view.findViewById(R.id.edSearch);
        svSearch.setQueryHint("Search here");
        svSearch.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String s) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String s) {
                if (TextUtils.isEmpty(s)){
                    adapter.filter("");
                    listView.clearTextFilter();
                }
                else {
                    adapter.filter(s);
                }
                return true;
            }
        });

        int searchCloseButtonId =
svSearch.getContext().getResources().getIdentifier("android:id/search_close_btn", null,
null);
        ImageView closeButton = (ImageView) view.findViewById(searchCloseButtonId);
```

```
        closeButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                svSearch.onActionViewCollapsed();
            }
        });

        //fast search using alphabet
        IndexBar mIndexBar = (IndexBar) view.findViewById(R.id.indexBar);
        mIndexBar.setOnIndexLetterChangedListener(new
IndexBar.OnIndexLetterChangedListener(){
            @Override
            public void onTouched(boolean touched)
            {
                // TODO's finger press and lift will call back here
            }

            @Override
            public void onLetterChanged(CharSequence indexChar, int index, float y)
            {
                // TODO will call back when the index letter changes
                svSearch.setQuery(indexChar,false);
                svSearch.onActionViewExpanded();
            }
        });

        linearLayout = (LinearLayout) view.findViewById(R.id.bg_listview);
        return view;
    }

    // permission
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        if (permissionHelper != null) {
            permissionHelper.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        }
    }

    private void initListView() {
        Common.musicList.clear();
        ContentResolver resolver = getActivity().getContentResolver();
```

```
        //Create the cursor MediaStore.Audio.Media.EXTERNAL_CONTENT_URI to get the audio
file, the latter is about selecting the filter conditions, fill in null here.
        Cursor cursor = resolver.query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, null,
null, null, MediaStore.Audio.Media.DEFAULT_SORT_ORDER);
        if(cursor.moveToFirst()){
            do {
                String title =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.TITLE));
                String artist =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.ARTIST));
                String album =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.ALBUM));
                int albumID =
cursor.getInt(cursor.getColumnIndex(MediaStore.Audio.Media.ALBUM_ID));
                int length =
cursor.getInt(cursor.getColumnIndex(MediaStore.Audio.Media.DURATION));
                String path =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.DATA));

                Music music = new Music();
                music.length = length;
                music.title = title;
                music.artist = artist;
                music.album = album;
                music.path = path;
                music.albumBip = getAlbumArt(albumID);

                Common.musicList.add(music);
                Common.musicList_backup.add(music);
            } while (cursor.moveToNext());
        }
        else {
            Toast.makeText(getActivity(), "No local music", Toast.LENGTH_SHORT).show();
        }
        cursor.close();
    }

    private Bitmap getAlbumArt(int album_id) {                          //get the album
image by id
        String mUriAlbums = "content://media/external/audio/albums";
        String[] projection = new String[]{"album_art"};
        Cursor cur = getActivity().getContentResolver().query(Uri.parse(mUriAlbums + "/" +
Integer.toString(album_id)), projection, null, null, null);
```

```
        String album_art = null;
        if (cur.getCount() > 0 && cur.getColumnCount() > 0) {
            cur.moveToNext();
            album_art = cur.getString(0);
        }
        cur.close();
        Bitmap bm = null;
        if (album_art != null) {
            bm = BitmapFactory.decodeFile(album_art);
        } else {
            bm = BitmapFactory.decodeResource(getResources(), R.mipmap.touxiang1);
        }
        return bm;
    }

    @Override
    public void onResume() {
        super.onResume();
        adapter.notifyDataSetChanged();
    }

    public void updateEnglish(){
            Common.musicList.clear();
            ContentResolver resolver = getActivity().getContentResolver();
            Cursor cursor = resolver.query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,
null, null, null, MediaStore.Audio.Media.DEFAULT_SORT_ORDER);

            if(cursor.moveToFirst()){
                do {
                    String title =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.TITLE));
                    String artist =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.ARTIST));
                    String album =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.ALBUM));
                    int albumID =
cursor.getInt(cursor.getColumnIndex(MediaStore.Audio.Media.ALBUM_ID));
                    int length =
cursor.getInt(cursor.getColumnIndex(MediaStore.Audio.Media.DURATION));
                    String path =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.DATA));

                    String firstC = String.valueOf(title.charAt(0));
```

```
                    Music music = new Music();
                    music.length = length;
                    music.title = title;
                    music.artist = artist;
                    music.album = album;
                    music.path = path;
                    music.albumBip = getAlbumArt(albumID);

                    if (firstC.equals("A") || firstC.equals("B")|| firstC.equals("C")||
firstC.equals("D")|| firstC.equals("E")|| firstC.equals("F")
                            || firstC.equals("G")|| firstC.equals("H")||
firstC.equals("I")|| firstC.equals("J")|| firstC.equals("K")|| firstC.equals("L")
                            || firstC.equals("M")|| firstC.equals("N")||
firstC.equals("O")|| firstC.equals("P")|| firstC.equals("Q")|| firstC.equals("R")
                            || firstC.equals("S")|| firstC.equals("T")||
firstC.equals("U")|| firstC.equals("V")|| firstC.equals("W")|| firstC.equals("X")
                            || firstC.equals("Y")|| firstC.equals("Z")||
firstC.equals("a")|| firstC.equals("b")|| firstC.equals("c")|| firstC.equals("d")
                            || firstC.equals("e")|| firstC.equals("f")||
firstC.equals("g")|| firstC.equals("h")|| firstC.equals("i")|| firstC.equals("j")
                            || firstC.equals("k")|| firstC.equals("l")||
firstC.equals("m")|| firstC.equals("n")|| firstC.equals("o")|| firstC.equals("p")
                            || firstC.equals("q")|| firstC.equals("r")||
firstC.equals("s")|| firstC.equals("t")|| firstC.equals("u")|| firstC.equals("v")
                            || firstC.equals("w")|| firstC.equals("x")||
firstC.equals("y")|| firstC.equals("z")) {
                        Common.musicList.add(music);
                        Common.musicList_backup.add(music);
                    }
                    else ;
                } while (cursor.moveToNext());
            }else {
                Toast.makeText(getActivity(), "Local No Music", Toast.LENGTH_SHORT).show();
            }
            cursor.close();
        adapter = new MusicAdapter(getActivity(), musicList);
        listView.setAdapter(adapter);
    }

    public void updateOthers(){
        Common.musicList.clear();
        ContentResolver resolver = getActivity().getContentResolver();
        Cursor cursor = resolver.query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI, null,
null, null, MediaStore.Audio.Media.DEFAULT_SORT_ORDER);
```

```
        if(cursor.moveToFirst()){
            do {
                String title =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.TITLE));
                String artist =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.ARTIST));
                String album =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.ALBUM));
                int albumID =
cursor.getInt(cursor.getColumnIndex(MediaStore.Audio.Media.ALBUM_ID));
                int length =
cursor.getInt(cursor.getColumnIndex(MediaStore.Audio.Media.DURATION));
                String path =
cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.DATA));

                String firstC = String.valueOf(title.charAt(0));

                Music music = new Music();
                music.length = length;
                music.title = title;
                music.artist = artist;
                music.album = album;
                music.path = path;
                music.albumBip = getAlbumArt(albumID);

                if (!firstC.equals("A")
&& !firstC.equals("B")&& !firstC.equals("C")&& !firstC.equals("D")&& !firstC.equals("E")&& !
firstC.equals("F")

&& !firstC.equals("G")&& !firstC.equals("H")&& !firstC.equals("I")&& !firstC.equals("J")&& !
firstC.equals("K")&& !firstC.equals("L")

&& !firstC.equals("M")&& !firstC.equals("N")&& !firstC.equals("O")&& !firstC.equals("P")&& !
firstC.equals("Q")&& !firstC.equals("R")

&& !firstC.equals("S")&& !firstC.equals("T")&& !firstC.equals("U")&& !firstC.equals("V")&& !
firstC.equals("W")&& !firstC.equals("X")

&& !firstC.equals("Y")&& !firstC.equals("Z")&& !firstC.equals("a")&& !firstC.equals("b")&& !
firstC.equals("c")&& !firstC.equals("d")
&& !firstC.equals("e")&& !firstC.equals("f")&& !firstC.equals("g")&& !firstC.equals("h")&& !
firstC.equals("i")&& !firstC.equals("j")
&& !firstC.equals("k")&& !firstC.equals("l")&& !firstC.equals("m")&& !firstC.equals("n")&& !
```

```
firstC.equals("o")&& !firstC.equals("p")
&& !firstC.equals("q")&& !firstC.equals("r")&& !firstC.equals("s")&& !firstC.equals("t")&& !
firstC.equals("u")&& !firstC.equals("v")
&& !firstC.equals("w")&& !firstC.equals("x")&& !firstC.equals("y")&& !firstC.equals("z")){
                    Common.musicList.add(music);
                    Common.musicList_backup.add(music);
                }
                else ;
            }  while (cursor.moveToNext());
        }else {
            Toast.makeText(getActivity(), "Local No Music", Toast.LENGTH_SHORT).show();
        }
        cursor.close();
        adapter = new MusicAdapter(getActivity(), musicList);
        listView.setAdapter(adapter);
    }

    public void updateAll(){
        initListView();
        adapter = new MusicAdapter(getActivity(), musicList);
        listView.setAdapter(adapter);
    }

    public void updateUI(String status){
        if (status.equals("true")){
            listView.setBackgroundResource(R.color.dark_main_text);
            svSearch.setBackgroundResource(R.color.dark_main_text);
            linearLayout.setBackgroundResource(R.color.dark_main_text);
        }
        else if (status.equals("false")){
            listView.setBackgroundResource(R.color.theme_white_main_bg);
            svSearch.setBackgroundResource(R.color.theme_white_main_bg);
            linearLayout.setBackgroundResource(R.color.theme_white_main_bg);
        }
    }
}
```

A.45 DriverMusicActivity.java

```
package com.example.fyp2_tsk.other;

import android.annotation.TargetApi;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.media.MediaPlayer;
import android.os.Build;
import android.support.annotation.RequiresApi;
import android.support.v4.view.GestureDetectorCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.ImageView;
import android.widget.TextView;
import com.example.fyp2_tsk.Gesture.DetectSwipeGestureDriver;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.entity.Music;
import com.example.fyp2_tsk.utils.BlurUtil;
import com.example.fyp2_tsk.utils.Common;
import java.io.IOException;

public class DriverMusicActivity extends AppCompatActivity implements View.OnClickListener {
    private int i = 0;
    private int playMode = 0;
    private int buttonWitch = 0;
    private ImageView bgImgvDriver;
    private TextView titleTvDriver;
    private TextView artistTvDriver;
    private ImageView prevImgvDriver;
    private ImageView nextImgvDriver;
    private int position;
    private MediaPlayer mediaPlayer;
    private ImageView pauseImgvDriver;
    private ImageView downImgDriver;
    private boolean isStopDriver;

    private GestureDetectorCompat gestureDetectorCompat = null;
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //create full screen Activity
        requestWindowFeature(Window.FEATURE_NO_TITLE);

getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParams.F
LAG_FULLSCREEN);
        setContentView(R.layout.activity_driver);
        bingID();

        //Swipe gesture control
        DetectSwipeGestureDriver detectSwipeGestureDriver = new DetectSwipeGestureDriver();
        detectSwipeGestureDriver.setActivity(this);
        gestureDetectorCompat = new GestureDetectorCompat(this, detectSwipeGestureDriver);

        Intent intent = getIntent();
//Through the getIntent () method to obtain intent information
        position = intent.getIntExtra("position", 0);            //get position

        mediaPlayer = new MediaPlayer();
        prevAndnextplaying(Common.musicList.get(position).path);
    }

    public void gestureSwipeLeft(){
        buttonWitch = 2;
        setBtnMode();
    }

    public void gestureSwipeRight(){
        buttonWitch = 1;
        setBtnMode();
    }

    private void prevAndnextplaying(String path) {
        isStopDriver = false;
        mediaPlayer.reset();
        titleTvDriver.setText(Common.musicList.get(position).title);
        artistTvDriver.setText(Common.musicList.get(position).artist + "--" +
Common.musicList.get(position).album);
        pauseImgvDriver.setImageResource(R.mipmap.ic_play_btn_pause);

        if (Common.musicList.get(position).albumBip != null) {
            Bitmap bgbm = BlurUtil.doBlur(Common.musicList.get(position).albumBip, 10,
```

```
5);//Blur the album pic
            bgImgvDriver.setImageBitmap(bgbm);//Set the blurred album picture as background
        } else {
            Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
R.mipmap.touxiang1);
            bgImgvDriver.setImageBitmap(bitmap);
        }

        try {
            mediaPlayer.setDataSource(path);
            mediaPlayer.prepare();
            mediaPlayer.start();
            mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
                @RequiresApi(api = Build.VERSION_CODES.KITKAT)
                @Override
                public void onCompletion(MediaPlayer mp) {
                    if (!mediaPlayer.isPlaying()) {
                        setPlayMode();
                    }
                }
            });
        } catch (IllegalArgumentException | SecurityException | IllegalStateException
                | IOException e) {
            e.printStackTrace();
        }

        MusicThread musicThread = new MusicThread();
        new Thread(musicThread).start();

    }

    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    private void setPlayMode() {
        //Loop all in orderly
        if (playMode == 0) {
            if (position == Common.musicList.size() - 1)//Loop by default
            {
                position = 0;// first song
                mediaPlayer.reset();
                prevAndnextplaying(Common.musicList.get(position).path);

            } else {
                position++;
                mediaPlayer.reset();
```

```
                prevAndnextplaying(Common.musicList.get(position).path);
            }
        }

        //single cycle
        else if (playMode == 1){
            //position no need to change
            mediaPlayer.reset();
            prevAndnextplaying(Common.musicList.get(position).path);
        }

        else if (playMode == 2)//shuffle playback
        {
            position = (int) (Math.random() * Common.musicList.size()); //generate random
number and the number no more than total no. of song
            mediaPlayer.reset();
            prevAndnextplaying(Common.musicList.get(position).path);
        }
    }

    @TargetApi(Build.VERSION_CODES.KITKAT)
    private void setBtnMode() {
        if (playMode == 0)//Loop all in orderly
        {
            if (position == Common.musicList.size() - 1)//Loop by default
            {
                if (buttonWitch == 1) {
                    position--;
                    mediaPlayer.reset();
                    prevAndnextplaying(Common.musicList.get(position).path);
                } else if (buttonWitch == 2) {
                    position = 0;
                    mediaPlayer.reset();
                    prevAndnextplaying(Common.musicList.get(position).path);
                }
            } else if (position == 0) {
                if (buttonWitch == 1) {
                    position = Common.musicList.size() - 1;
                    mediaPlayer.reset();
                    prevAndnextplaying(Common.musicList.get(position).path);
                } else if (buttonWitch == 2) {
                    position++;
                    mediaPlayer.reset();
                    prevAndnextplaying(Common.musicList.get(position).path);
```

```
            }
        } else {
            if (buttonWitch == 1) {
                position--;
                mediaPlayer.reset();
                prevAndnextplaying(Common.musicList.get(position).path);

            } else if (buttonWitch == 2) {
                position++;
                mediaPlayer.reset();
                prevAndnextplaying(Common.musicList.get(position).path);
            }
        }
    }

    else if (playMode == 1)//single cycle
    {

        mediaPlayer.reset();
        prevAndnextplaying(Common.musicList.get(position).path);
    }

    else if (playMode == 2)//shuffle playback
    {
        position = (int) (Math.random() * Common.musicList.size());
        mediaPlayer.reset();
        prevAndnextplaying(Common.musicList.get(position).path);
    }
}

private void bingID() {
    titleTvDriver = (TextView)findViewById(R.id.music_title_tv_driver);
    artistTvDriver = (TextView) findViewById(R.id.music_artist_tv_driver);
    bgImgvDriver = (ImageView) findViewById(R.id.music_bg_imgv_driver);
    prevImgvDriver = (ImageView) findViewById(R.id.music_prev_imgv_driver);
    nextImgvDriver = (ImageView) findViewById(R.id.music_next_imgv_driver);
    pauseImgvDriver = (ImageView) findViewById(R.id.music_pause_imgv_driver);
    downImgDriver = (ImageView) findViewById(R.id.music_down_imgv_driver);
    pauseImgvDriver.setOnClickListener(this);
    prevImgvDriver.setOnClickListener(this);
    nextImgvDriver.setOnClickListener(this);
    downImgDriver.setOnClickListener(this);
}
```

```
@RequiresApi(api = Build.VERSION_CODES.KITKAT)
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.music_prev_imgv_driver:
            buttonWitch = 1;
            setBtnMode();
            break;

        case R.id.music_next_imgv_driver:
            buttonWitch = 2;
            setBtnMode();
            break;

        case R.id.music_pause_imgv_driver:
            if (mediaPlayer.isPlaying()) {
                mediaPlayer.pause();
                pauseImgvDriver.setImageResource(R.mipmap.ic_play_btn_play);
            } else {
                mediaPlayer.start();
                pauseImgvDriver.setImageResource(R.mipmap.ic_play_btn_pause);
            }
            break;
        case R.id.music_down_imgv_driver:
            this.finish();
            Intent intent = new Intent(DriverMusicActivity.this, MusicActivity.class);
            intent.putExtra("position", position);
            startActivity(intent);
            break;
        default: break;
    }
}

@Override
protected void onPause() {
    super.onPause();
    for (Music music : Common.musicList) {
        music.isPlaying = false;
    }
    Common.musicList.get(position).isPlaying = true;
}

@Override
protected void onStop() {
```

```
        super.onStop();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        i = 0;
        isStopDriver = false;

        if (mediaPlayer.isPlaying()) {
            mediaPlayer.stop();
        }

        if (mediaPlayer != null) {
            mediaPlayer.stop();
        }
    }

    //Create class like MusicThread to implement Runnable interface and multithreading
    class MusicThread implements Runnable {
        @Override
        public void run() {
            while (!isStopDriver && Common.musicList_backup.get(position) != null) {
                try {
                    Thread.sleep(1000);
                }
                catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        gestureDetectorCompat.onTouchEvent(event);
        return true;
    }
}
```

A.46 MainActivity.java

```
package com.example.fyp2_tsk.other;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.res.ColorStateList;
import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.net.Uri;
import android.os.Build;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.annotation.NonNull;
import android.support.design.widget.NavigationView;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentTransaction;
import android.support.v7.app.ActionBarDrawerToggle;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.widget.Toast;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.SleepTimer.AppPreference;
import com.example.fyp2_tsk.SleepTimer.AuxiliaryPreference;
import com.example.fyp2_tsk.SleepTimer.BroadcastManager;
import com.example.fyp2_tsk.SleepTimer.PeriodicTask;
import com.example.fyp2_tsk.SleepTimer.ThemeChangeable;
import com.example.fyp2_tsk.SleepTimer.ThemeEnum;
import com.example.fyp2_tsk.SleepTimer.TimeSleepActivity;
import com.example.fyp2_tsk.common.app.di.AppComponent;
import com.example.fyp2_tsk.common.base.BaseActivity;
import com.example.fyp2_tsk.common.base.BasePresenter;
import com.example.fyp2_tsk.common.base.IBaseView;
import com.example.fyp2_tsk.databinding.ActivityMainBinding;
import com.example.fyp2_tsk.skin.ThemeColorSelectDialog;
import com.example.fyp2_tsk.utils.ColorUtils;
import com.jaeger.library.StatusBarUtil;
import com.orhanobut.logger.Logger;
```

```
import com.example.fyp2_tsk.home.ui.CutFragment;

public class MainActivity extends BaseActivity<IBaseView, BasePresenter<IBaseView>,
ActivityMainBinding> implements ThemeChangeable {
    private Fragment mCurFragment;
    private Fragment mCutFragment;
    private Fragment mMusicFragment;
    private long time = 0;

    //Sleep timer broadcast
    private AuxiliaryPreference auxiliaryPreference;
    private BroadcastReceiver appThemeChangeAutomaticReceiver;
    private BroadcastReceiver appQuitTimeCountdownReceiver;
    private BroadcastManager broadcastManager;
    protected AppPreference appPreference;

    //shared preferences
    private SharedPreferences mPreferences;
    private SharedPreferences.Editor mEditor;

    Bundle b;
    Boolean nightFlag = false;
    String statusFlag;

    //Used to sleep timers,stop() no call during execute, only finish of countdown
    private PeriodicTask quitCountDown = null;
    private CountDownTask task = null;

    @Override
    protected void ComponentInject(AppComponent appComponent) {

    }

    @Override
    protected int initLayoutResId() {
        //create full screen Activity
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
        return R.layout.activity_main;
    }

    @Override
    protected void initData(Bundle savedInstanceState) {
```

```
        initView();
        mCutFragment = new CutFragment();
        mMusicFragment = new MusicFragment();
        switchToMusic();
        StatusBarUtil.setColorForDrawerLayout(MainActivity.this, mDataBinding.drawerlayout,
Color.TRANSPARENT);
        auxiliaryPreference = new AuxiliaryPreference(this);
    }

    private void initView() {
        //create shared preferences to record status of night mode
        mPreferences = PreferenceManager.getDefaultSharedPreferences(this);
        mEditor = mPreferences.edit();
        statusFlag = nightFlag.toString(); //night mode flag to check enable or disable
        mEditor.putString(getString(R.string.statusNightMode),statusFlag);
        mEditor.commit();

        b = new Bundle();

        broadcastManager = BroadcastManager.getInstance();
        appPreference = new AppPreference(this); // to get theme color for night mode
feature
        checkTheme(); // use appPreference to decide show white or black color background

        //register broadcast
        initBroadcastReceivers();
        initToolBar();
        initDrawer();
        initNavigationView();
    }

    private void initToolBar() {
        mDataBinding.toolbar.setTitle(getResources().getString(R.string.main_tab_music));
        mDataBinding.toolbar.setTitleTextColor(Color.parseColor("#ffffff"));
        mDataBinding.toolbar.setBackgroundResource(R.color.theme_color);

        setSupportActionBar(mDataBinding.toolbar);
    }

    private void initDrawer() {
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(this,
mDataBinding.drawerlayout, mDataBinding.toolbar, R.string.app_name, R.string.app_name);
        mDataBinding.drawerlayout.addDrawerListener(toggle);
        toggle.syncState();
```

```
    }

    private void initNavigationView() {
        mDataBinding.navigationView.setItemIconTintList(null);
        mDataBinding.navigationView.setNavigationItemSelectedListener(new
NavigationView.OnNavigationItemSelectedListener() {
            @Override
            public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {
                boolean isNeedChecked = true;
                switch (menuItem.getItemId()) {
                    case R.id.item_theme:
                        isNeedChecked = false;
                        ThemeColorSelectDialog mdf = new ThemeColorSelectDialog();
                        FragmentTransaction ft =
getSupportFragmentManager().beginTransaction();
                        ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
                        mdf.show(ft, "df");
                        break;

                    case R.id.item_cut:
                        switchToCutPage();
                        break;

                    case R.id.item_download:
                        switchToDownload();
                        break;

                    case R.id.item_music:
                        switchToMusic();
                        break;

                    case R.id.item_sleeptimer:
                        startActivity(new Intent(MainActivity.this,
TimeSleepActivity.class));
                        break;

                    case R.id.item_night_mode:
                        handleModeSwitch();
                        break;

                    case R.id.item_exit:
                        android.os.Process.killProcess(android.os.Process.myPid());
                        finish();
                        break;
```

```
                }
                invalidateOptionsMenu();
                menuItem.setChecked(isNeedChecked);
                mDataBinding.drawerlayout.closeDrawers();
                return true;
            }
        });
    }

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

    private void switchToCutPage() {
        mCurFragment = mCutFragment;
        b.putBoolean("nightFlag", nightFlag);
        mCutFragment.setArguments(b);
        getSupportFragmentManager().beginTransaction().replace(R.id.frame_content,
mCutFragment).commitNow();
        mDataBinding.toolbar.setTitle(getResources().getString(R.string.main_tab_trim));
    }

    private void switchToMusic() {
        mCurFragment = mMusicFragment;
        b.putBoolean("nightFlag", nightFlag);
        mMusicFragment.setArguments(b);
        getSupportFragmentManager().beginTransaction().replace(R.id.frame_content,
mMusicFragment).commitNow();
        mDataBinding.toolbar.setTitle(getResources().getString(R.string.main_tab_music));
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main_menu, menu);
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
        if (mCurFragment instanceof CutFragment) {
            menu.findItem(R.id.home_item_open).setVisible(true);
```

```
        } else {
            menu.findItem(R.id.home_item_open).setVisible(false);
        }
        Logger.d("onPrepareOptionsMenu: visible" + (mCurFragment instanceof CutFragment));
        return super.onPrepareOptionsMenu(menu);
    }

    //cut page click open file icon
    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.home_item_open:
                if (mCurFragment instanceof CutFragment) {
                    ((CutFragment) mCurFragment).openFile();
                }
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    //press back button to quit application
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE_BACK && event.getAction() == KeyEvent.ACTION_DOWN) {
            if ((System.currentTimeMillis() - time > 1000)) {
                Toast.makeText(this, getResources().getString(R.string.main_finish_tip),
                        Toast.LENGTH_SHORT).show();
                time = System.currentTimeMillis();
            } else {
                moveTaskToBack(false);
            }
            return true;
        } else {
            return super.onKeyDown(keyCode, event);
        }
    }

    private void initBroadcastReceivers() {
        //sleep timer
        appQuitTimeCountdownReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                int status =
intent.getIntExtra(BroadcastManager.Countdown.APP_QUIT_TIME_COUNTDOWN_STATUS,
```

```
BroadcastManager.Countdown.STOP_COUNTDOWN);
                if (status == BroadcastManager.Countdown.STOP_COUNTDOWN) {
                    stopQuitCountdown(true);
                } else {
                    startQuitCountdown();
                }
            }
        };

        //nighmode
        appThemeChangeAutomaticReceiver = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                int va =
intent.getIntExtra(BroadcastManager.APP_THEME_CHANGE_AUTOMATIC_TOKEN,
BroadcastManager.APP_THEME_CHANGE_AUTOMATIC_WHITE);
                ThemeEnum theme = va == BroadcastManager.APP_THEME_CHANGE_AUTOMATIC_WHITE ?
ThemeEnum.WHITE : ThemeEnum.DARK;
                appPreference.updateTheme(theme);
                switchThemeMode(theme);
            }
        };

        broadcastManager.registerBroadReceiver(this, appQuitTimeCountdownReceiver,
BroadcastManager.FILTER_APP_QUIT_TIME_COUNTDOWN);
        broadcastManager.registerBroadReceiver(this, appThemeChangeAutomaticReceiver,
BroadcastManager.FILTER_APP_THEME_CHANGE_AUTOMATIC);
    }

    public void startQuitCountdown() {
        if (quitCountDown == null) {
            task = new CountDownTask();
            quitCountDown = new PeriodicTask(task, 1000);
        } else {
            task.reset();
        }
        quitCountDown.start();
    }

    public void stopQuitCountdown(boolean resetText) {
        if (quitCountDown != null && quitCountDown.isSchedule()) {
            quitCountDown.stop();
        }
```

```
        auxiliaryPreference.setTimeSleepDisable();
        if (resetText) {
            resetText();
        }
    }

    //reset sleep timer item at left navigation bar after disable
    private void resetText() {
        Menu menu = mDataBinding.navigationView.getMenu();
        if (menu != null) {
            MenuItem item = menu.findItem(R.id.item_sleeptimer);
            item.setTitle("Sleep timer");
        }
    }

    private void unregisterReceiver() {
        if (appQuitTimeCountdownReceiver != null) {
            broadcastManager.unregisterReceiver(this, appQuitTimeCountdownReceiver);
        }
        if (appThemeChangeAutomaticReceiver != null) {
            broadcastManager.unregisterReceiver(this, appThemeChangeAutomaticReceiver);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        unregisterReceiver();
    }

    private class CountDownTask implements PeriodicTask.Task {
        final MenuItem item;
        int dur; // dur unit is minute
        long startTime;
        final String title;

        public CountDownTask() {
            reset();

            Menu menu = mDataBinding.navigationView.getMenu();
            if (menu != null) {
                item = menu.findItem(R.id.item_sleeptimer);
                title = item.getTitle().toString();
            } else {
```

```
                item = null;
                title = "";
            }
        }

        int sec = 59;

        @Override
        public void execute() {
            if (dur >= 0 && item != null) {
                if (dur == 0) {
                    if (sec == 0) {
                        countDownFinish(false);
                    } else if (sec == 23) { // count down 23 second and then notice user
                        mDataBinding.navigationView.post(new Runnable() {
                            @Override
                            public void run() {
                                Toast.makeText(MainActivity.this, "Mp3 Music Player will
terminate after 20 second.", Toast.LENGTH_SHORT).show();
                            }
                        });
                    }
                }

                if (sec == 0) {
                    sec = 59;
                    dur--;
                } else {
                    sec--;
                }
                countDown();
            } else {
                countDownFinish(false);
            }
        }

        private void countDownFinish(boolean resetText) {
            stopQuitCountdown(resetText);

            //time reached and terminated
            finish();
            android.os.Process.killProcess(android.os.Process.myPid());
            System.exit(1);
            Toast.makeText(getApplicationContext(), "Closed Completely and Safely",
```

```
Toast.LENGTH_LONG).show();
        }

        private void countDown() {
            mDataBinding.navigationView.post(new Runnable() {
                @Override
                public void run() {
                    String newTitle = title + "    " + getString();
                    item.setTitle(newTitle);
                }
            });
        }

        private String getString() {
            String sq = " : ";
            String time;
            if (dur >= 60) {
                int h = dur / 60;
                int m = dur % 60;
                time = get2Str(h) + sq + get2Str(m) + sq + get2Str(sec);
            } else {
                int m = dur % 60;
                time = get2Str(m) + sq + get2Str(sec);
            }
            return time;
        }

        private String get2Str(int m) {
            return m < 10 ? ("0" + m) : (m + "");
        }

        private void reset() {
            dur = auxiliaryPreference.getTimeSleepDuration();
            startTime = auxiliaryPreference.getTimeSleepStartTime();
            // dur = 2 ,start count down from 1.59
            dur--;
        }
    }

    /**
     * Switch to daily mode and night mode
     */
    protected void checkTheme() {
        ThemeEnum themeEnum = appPreference.getTheme();
```

```
        if (themeEnum == ThemeEnum.DARK) {
            this.setTheme(R.style.Theme_DARK);
        } else {
            this.setTheme(R.style.Theme_WHITE);
        }
    }

    private void handleModeSwitch() {
        ThemeEnum theme = ThemeEnum.reversal(appPreference.getTheme());

        appPreference.updateTheme(theme);
        switchThemeMode(theme);
    }

    public void switchThemeMode(final ThemeEnum theme) {
        int[] colors = ColorUtils.get10ThemeColors(this, theme);
        int to = colors[3];

        View view = getWindow().getDecorView();
        view.setBackgroundColor(to);
        themeChange(theme, null);
    }

    @Override
    public void themeChange(ThemeEnum themeEnum, int[] colors) {
        ThemeEnum th = appPreference.getTheme();
        int[] cs = ColorUtils.get10ThemeColors(MainActivity.this, th);
        int mainBC = cs[3];
        int mainTC = cs[5];
        int accentC = cs[2];

        mDataBinding.navigationView.setItemTextColor(ColorStateList.valueOf(mainTC));
        mDataBinding.navigationView.setBackgroundColor(mainBC);

        updateSwitchMenuIconAndText();

        Menu menu = mDataBinding.navigationView.getMenu();
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            for (int i = 0; i < menu.size(); i++) {
                MenuItem item = menu.getItem(0);
                Drawable icon = item.getIcon();
                if (icon != null) {
                    icon.setTint(accentC);
                }
```

```
            }
        }
    }

    private void updateSwitchMenuIconAndText() {
        MenuItem item;
        Menu menu = mDataBinding.navigationView.getMenu();
        if (menu != null) {
            item = menu.findItem(R.id.item_night_mode);
            if (item == null) {
                return;
            }
        } else {
            return;
        }

        ThemeEnum theme = appPreference.getTheme();

        Drawable icon;
        String title;

        if (theme == ThemeEnum.WHITE || theme == ThemeEnum.VARYING) { // Switch to night
mode
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                icon = MainActivity.this.getDrawable(R.drawable.ic_night);
            } else {
                icon = MainActivity.this.getResources().getDrawable(R.drawable.ic_night);
            }
            title = MainActivity.this.getString(R.string.setting_night_mode);
            nightFlag = false;
            statusFlag = nightFlag.toString();
            mEditor.putString(getString(R.string.statusNightMode),statusFlag);
            mEditor.commit();
            final FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
            ft.detach(mCurFragment);
            ft.attach(mCurFragment);
            ft.commit();
        }
        else { // Switch to daily mode
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
                icon = MainActivity.this.getDrawable(R.drawable.ic_daytime);
            } else { // Switch to daily mode
                icon = MainActivity.this.getResources().getDrawable(R.drawable.ic_daytime);
            }
```

```
            title = MainActivity.this.getString(R.string.setting_daytime_mode);
            nightFlag = true;
            statusFlag = nightFlag.toString();
            mEditor.putString(getString(R.string.statusNightMode),statusFlag);
            mEditor.commit();
            final FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
            ft.detach(mCurFragment);
            ft.attach(mCurFragment);
            ft.commit();
        }

        item.setIcon(icon);
        item.setTitle(title);
    }
}
```

A.47 MusicActivity.java

```
package com.example.fyp2_tsk.other;

import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.annotation.TargetApi;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.media.MediaPlayer;
import android.os.Build;
import android.os.Handler;
import android.os.Message;
import android.support.annotation.RequiresApi;
import android.support.v4.view.GestureDetectorCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
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.LinearInterpolator;
import android.view.animation.RotateAnimation;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;
import android.widget.Toast;
import com.example.fyp2_tsk.Gesture.DetectSwipeGestureListener;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.ShakeControl.ShakeListener;
import com.example.fyp2_tsk.entity.Music;
import com.example.fyp2_tsk.utils.BlurUtil;
import com.example.fyp2_tsk.utils.Common;
import com.example.fyp2_tsk.utils.MergeImage;

import java.io.IOException;

public class MusicActivity extends AppCompatActivity implements View.OnClickListener {
    private int i = 0;
    private int playMode = 0;
    private int buttonWitch = 0;
    private ImageView bgImgv;
```

```
    private TextView titleTv;
    private TextView artistTv;
    private TextView currrentTv;
    private TextView totalTv;
    private ImageView prevImgv;
    private ImageView nextImgv;
    private int position;
    private ImageView discImagv;
    private ImageView needleImagv;
    private MediaPlayer mediaPlayer;
    private ImageView driveMode;
    private ImageView pauseImgv;
    private ImageView downImg;
    private ImageView styleImg;
    private SeekBar seekBar;
    private boolean isStop;
    private ObjectAnimator objectAnimator = null;
    private RotateAnimation rotateAnimation = null;
    private RotateAnimation rotateAnimation2 = null;
    private String TAG = "MusicActivity";

    private GestureDetectorCompat gestureDetectorCompat = null;
    private ShakeListener shakeListener;

    //Handler implementation to pass values to the main thread
    private Handler handler = new Handler() {
        @Override
        public void handleMessage(Message msg) {
            super.handleMessage(msg);
            seekBar.setProgress((int) (msg.what));
            currrentTv.setText(formatTime(msg.what));
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //create full screen Activity
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_music);

        bingID();
```

```
        //Swipe gesture control
        DetectSwipeGestureListener swipeGestureListener = new DetectSwipeGestureListener();
        swipeGestureListener.setActivity(this);
        gestureDetectorCompat = new GestureDetectorCompat(this, swipeGestureListener);

        Intent intent = getIntent();
        position = intent.getIntExtra("position", 0);            //get position from last
activity

        mediaPlayer = new MediaPlayer();
        prevAndnextplaying(Common.musicList.get(position).path);
        seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
                if (fromUser) {
                    mediaPlayer.seekTo(progress);
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {

            }
        });

        shakeListener = new ShakeListener(MusicActivity.this);
        shakeListener.setOnShakeListener(new ShakeListener.OnShakeListener() {
            @Override
            public void onShake() {
                if (mediaPlayer.isPlaying())
                    mediaPlayer.reset();
                buttonWitch = 2;
                setBtnMode();
            }
        });
    }

    public void gestureSwipeLeft() {
```

```
        buttonWitch = 2;
        setBtnMode();
    }

    public void gestureSwipeRight() {
        buttonWitch = 1;
        setBtnMode();
    }

    private void prevAndnextplaying(String path) {
        isStop = false;
        mediaPlayer.reset();
        titleTv.setText(Common.musicList.get(position).title);
        artistTv.setText(Common.musicList.get(position).artist + "--" +
Common.musicList.get(position).album);
        pauseImgv.setImageResource(R.mipmap.ic_play_btn_pause);

        if (Common.musicList.get(position).albumBip != null) {
            Bitmap bgbm = BlurUtil.doBlur(Common.musicList.get(position).albumBip, 10,
5);//Blur the album image
            bgImgv.setImageBitmap(bgbm);                                    //set it as bg
            Bitmap bitmap1 = BitmapFactory.decodeResource(getResources(),
R.mipmap.play_page_disc);//BitmapFactory.decodeResource is used to parse and create a Bitmap
object from the specified resource file according to the given resource ID
            Bitmap bm = MergeImage.mergeThumbnailBitmap(bitmap1,
Common.musicList.get(position).albumBip);//put bg image on disc
            discImagv.setImageBitmap(bm);
        } else {
            Bitmap bitmap = BitmapFactory.decodeResource(getResources(),
R.mipmap.touxiang1);
            bgImgv.setImageBitmap(bitmap);
            Bitmap bitmap1 = BitmapFactory.decodeResource(getResources(),
R.mipmap.play_page_disc);
            Bitmap bm = MergeImage.mergeThumbnailBitmap(bitmap1, bitmap);
            discImagv.setImageBitmap(bm);
        }
        try {
            mediaPlayer.setDataSource(path);
            mediaPlayer.prepare();
            mediaPlayer.start();
            mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
                @RequiresApi(api = Build.VERSION_CODES.KITKAT)
                @Override
                public void onCompletion(MediaPlayer mp) {
```

```
                    if (!mediaPlayer.isPlaying()) {
                        setPlayMode();
                    }

                }
            });
        } catch (IllegalArgumentException | SecurityException | IllegalStateException
                | IOException e) {
            e.printStackTrace();
        }

        totalTv.setText(formatTime(Common.musicList.get(position).length));
        seekBar.setMax(Common.musicList.get(position).length);

        MusicThread musicThread = new MusicThread();
//start thread
        new Thread(musicThread).start();

        //Instantiate, set the rotation object, cd rotation animation
        objectAnimator = ObjectAnimator.ofFloat(discImagv, "rotation", 0f, 360f);
        //set time of one round
        objectAnimator.setDuration(8000);
        //set speed of one round
        objectAnimator.setInterpolator(new LinearInterpolator());
        //Set the number of cycles ,-1 is repeat
        objectAnimator.setRepeatCount(-1);
        //Set how to turn after one turn
        objectAnimator.setRepeatMode(ValueAnimator.RESTART);
        objectAnimator.start();

        rotateAnimation = new RotateAnimation(-25f, 0f, Animation.RELATIVE_TO_SELF, 0.3f,
Animation.RELATIVE_TO_SELF, 0.1f);
        rotateAnimation.setDuration(500);
        rotateAnimation.setInterpolator(new LinearInterpolator());
        rotateAnimation.setRepeatCount(0);
        rotateAnimation.setFillAfter(true);
        rotateAnimation.setStartOffset(500);
        needleImagv.setAnimation(rotateAnimation);
        rotateAnimation.cancel();

        rotateAnimation2 = new RotateAnimation(0f, -25f, Animation.RELATIVE_TO_SELF, 0.3f,
Animation.RELATIVE_TO_SELF, 0.1f);
        rotateAnimation2.setDuration(500);
        rotateAnimation2.setInterpolator(new LinearInterpolator());
```

```
        rotateAnimation2.setRepeatCount(0);
        rotateAnimation2.setFillAfter(true);
        needleImagv.setAnimation(rotateAnimation2);
        rotateAnimation2.cancel();
    }

    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    private void setPlayMode() {
        //All repeat
        if (playMode == 0) {
            if (position == Common.musicList.size() - 1)//last song position
            {
                position = 0;// first song
                mediaPlayer.reset();
                objectAnimator.pause();
                needleImagv.startAnimation(rotateAnimation2);
                prevAndnextplaying(Common.musicList.get(position).path);
            } else {
                position++;
                mediaPlayer.reset();
                objectAnimator.pause();
                needleImagv.startAnimation(rotateAnimation2);
                prevAndnextplaying(Common.musicList.get(position).path);
            }
        }
        //Single cycle
        else if (playMode == 1) {
            //position no nid to change
            mediaPlayer.reset();
            objectAnimator.pause();
            needleImagv.startAnimation(rotateAnimation2);
            prevAndnextplaying(Common.musicList.get(position).path);
        }
        else if (playMode == 2)//Shuffle playback
        {
            position = (int) (Math.random() * Common.musicList.size()); //generate random
number
            mediaPlayer.reset();
            objectAnimator.pause();
            needleImagv.startAnimation(rotateAnimation2);
            prevAndnextplaying(Common.musicList.get(position).path);
        }
    }
```

```
@TargetApi(Build.VERSION_CODES.KITKAT)
private void setBtnMode() {
    //Repeat playlist
    if (playMode == 0) {
        // if current play is last song in playlist
        if (position == Common.musicList.size() - 1) {
            //skip to previous song
            if (buttonWitch == 1) {
                position--;
                mediaPlayer.reset();
                objectAnimator.pause();
                needleImagv.startAnimation(rotateAnimation2);
                prevAndnextplaying(Common.musicList.get(position).path);
            }
            //skip next song which is first song
            else if (buttonWitch == 2) {
                position = 0;// First song
                mediaPlayer.reset();
                objectAnimator.pause();
                needleImagv.startAnimation(rotateAnimation2);
                prevAndnextplaying(Common.musicList.get(position).path);
            }
        } else if (position == 0) {
            if (buttonWitch == 1) {
                position = Common.musicList.size() - 1;
                mediaPlayer.reset();
                objectAnimator.pause();
                needleImagv.startAnimation(rotateAnimation2);
                prevAndnextplaying(Common.musicList.get(position).path);
            } else if (buttonWitch == 2) {
                position++;
                mediaPlayer.reset();
                objectAnimator.pause();
                needleImagv.startAnimation(rotateAnimation2);
                prevAndnextplaying(Common.musicList.get(position).path);
            }
        } else {
            if (buttonWitch == 1) {
                position--;
                mediaPlayer.reset();
                objectAnimator.pause();
                needleImagv.startAnimation(rotateAnimation2);
                prevAndnextplaying(Common.musicList.get(position).path);
```

```
            } else if (buttonWitch == 2) {
                position++;
                mediaPlayer.reset();
                objectAnimator.pause();
                needleImagv.startAnimation(rotateAnimation2);
                prevAndnextplaying(Common.musicList.get(position).path);
            }
        }
    }

    else if (playMode == 1)//single cycle
    {
        //position no nid to change
        mediaPlayer.reset();
        objectAnimator.pause();
        needleImagv.startAnimation(rotateAnimation2);
        prevAndnextplaying(Common.musicList.get(position).path);
    }

    else if (playMode == 2)//random
    {
        position = (int) (Math.random() * Common.musicList.size());//random play
        mediaPlayer.reset();
        objectAnimator.pause();
        needleImagv.startAnimation(rotateAnimation2);
        prevAndnextplaying(Common.musicList.get(position).path);
    }
}

private String formatTime(int length) {
    int min = length / 1000 / 60;
    int sec = (length / 1000) % 60;
    String minStr = min < 10 ? "0" + min : min + "";
    String secStr = sec < 10 ? "0" + sec : sec + "";
    return minStr + ":" + secStr;
}

private void bingID() {
    titleTv = (TextView) findViewById(R.id.music_title_tv);
    artistTv = (TextView) findViewById(R.id.music_artist_tv);
    bgImgv = (ImageView) findViewById(R.id.music_bg_imgv);
    currrentTv = (TextView) findViewById(R.id.music_current_tv);
    totalTv = (TextView) findViewById(R.id.music_total_tv);
    prevImgv = (ImageView) findViewById(R.id.music_prev_imgv);
```

```
        nextImgv = (ImageView) findViewById(R.id.music_next_imgv);
        discImagv = (ImageView) findViewById(R.id.music_disc_imagv);
        needleImagv = (ImageView) findViewById(R.id.music_needle_imag);
        pauseImgv = (ImageView) findViewById(R.id.music_pause_imgv);
        downImg = (ImageView) findViewById(R.id.music_down_imgv);
        seekBar = (SeekBar) findViewById(R.id.music_seekbar);
        styleImg = (ImageView) findViewById(R.id.music_play_btn_loop_img);
        driveMode = (ImageView) findViewById(R.id.driver_Mode);
        pauseImgv.setOnClickListener(this);
        prevImgv.setOnClickListener(this);
        nextImgv.setOnClickListener(this);
        downImg.setOnClickListener(this);
        styleImg.setOnClickListener(this);
        driveMode.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(MusicActivity.this, DriverMusicActivity.class);
                intent.putExtra("position", position);
                startActivity(intent);
                MusicActivity.this.finish();
            }
        });
    }

    @RequiresApi(api = Build.VERSION_CODES.KITKAT)
    @Override
    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.music_prev_imgv:
                buttonWitch = 1;
                setBtnMode();
                break;

            case R.id.music_next_imgv:
                buttonWitch = 2;
                setBtnMode();
                break;

            case R.id.music_pause_imgv:
                if (mediaPlayer.isPlaying()) {
                    mediaPlayer.pause();
                    objectAnimator.pause();
                    needleImagv.startAnimation(rotateAnimation2);
                    pauseImgv.setImageResource(R.mipmap.ic_play_btn_play);
```

```
                } else {
                    mediaPlayer.start();
                    objectAnimator.resume();
                    needleImagv.startAnimation(rotateAnimation);
                    pauseImgv.setImageResource(R.mipmap.ic_play_btn_pause);
                }
                break;

            case R.id.music_play_btn_loop_img:
                i++;
                if (i % 3 == 1) {
                    Toast.makeText(MusicActivity.this, "Single cycle",
Toast.LENGTH_SHORT).show();
                    playMode = 1;
                    styleImg.setImageResource(R.mipmap.ic_play_btn_one);
                }
                if (i % 3 == 2) {
                    Toast.makeText(MusicActivity.this, "Shuffle Playback",
Toast.LENGTH_SHORT).show();
                    playMode = 2;
                    styleImg.setImageResource(R.mipmap.ic_play_btn_shuffle);
                }
                if (i % 3 == 0) {
                    Toast.makeText(MusicActivity.this, "Normal playback",
Toast.LENGTH_SHORT).show();
                    playMode = 0;
                    styleImg.setImageResource(R.mipmap.ic_play_btn_loop);
                }
                break;
            case R.id.music_down_imgv:
                this.finish();
                break;
            default:
                break;
        }
    }

    @Override
    protected void onPause() {
        super.onPause();
        for (Music music : Common.musicList
        ) {
            music.isPlaying = false;
        }
```

```
        Common.musicList.get(position).isPlaying = true;
    }

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

    @Override
    protected void onDestroy() {
        super.onDestroy();
        i = 0;
        isStop = false;
        if (mediaPlayer.isPlaying()) {
            mediaPlayer.stop();
        }
        if (mediaPlayer != null) {
            mediaPlayer.stop();
        }
    }

    //MusicThread class to implement Runnable interface
    class MusicThread implements Runnable {
        @Override
        public void run() {
            while (!isStop && Common.musicList_backup.get(position) != null) {
                try {
                    //let thread sleep 1000 milliseconds
                    Thread.sleep(1000);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                //Send the current time to the Handler to update the UI
                handler.sendEmptyMessage(mediaPlayer.getCurrentPosition());
            }
        }
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        gestureDetectorCompat.onTouchEvent(event);
        return true;
    }
}
```

A.48 ThemeEnum.java

```
package com.example.fyp2_tsk.SleepTimer;

public enum ThemeEnum {
    WHITE,

    DARK,

    //Change with album image
    VARYING;

    public static ThemeEnum reversal(ThemeEnum theme) {
        if (theme == WHITE || theme == VARYING) {
            return DARK;
        } else {
            return WHITE;
        }
    }
}
```

A.49 MusicFragment.java

```
package com.example.fyp2_tsk.other;

import android.content.Context;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.support.v4.app.Fragment;
import android.view.View;
import com.example.fyp2_tsk.adapter.MusicPagerAdapter;
import com.example.fyp2_tsk.common.app.di.AppComponent;
import com.example.fyp2_tsk.common.base.BaseFragment;
import com.example.fyp2_tsk.common.base.BasePresenter;
import com.example.fyp2_tsk.common.base.IBaseView;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.databinding.FragmentMusicBinding;
import java.util.ArrayList;
import java.util.List;

public class MusicFragment extends BaseFragment<IBaseView, BasePresenter<IBaseView>,
FragmentMusicBinding>{
    private List<Fragment> fragmentList = new ArrayList<>();
    private AllFragment allFragment;
    private SharedPreferences mPreferences;
    private SharedPreferences.Editor mEditor;
    String status;
    Boolean firstTime = true;

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

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

    @Override
    protected void ComponentInject(AppComponent appComponent) {

    }

    @Override
```

```
    protected void init(View view) {
        init();
    }


    private void init(){
        mPreferences = PreferenceManager.getDefaultSharedPreferences(getActivity());
        mEditor = mPreferences.edit();

        allFragment = new AllFragment();

        fragmentList.add(allFragment);

        MusicPagerAdapter musicPagerAdapter = new
MusicPagerAdapter(getActivity().getSupportFragmentManager(), fragmentList);

        //viewPager binding the adapter
        mDataBinding.mainVp.setAdapter(musicPagerAdapter);

        mDataBinding.engLogicTv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                allFragment.updateEnglish();

mDataBinding.mainLogicTv.setTextColor(getResources().getColor(R.color.white_60P));

mDataBinding.engLogicTv.setTextColor(getResources().getColor(R.color.white));

mDataBinding.chineseLogicTv.setTextColor(getResources().getColor(R.color.white_60P));
            }
        });

        mDataBinding.chineseLogicTv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

mDataBinding.mainLogicTv.setTextColor(getResources().getColor(R.color.white_60P));

mDataBinding.engLogicTv.setTextColor(getResources().getColor(R.color.white_60P));

mDataBinding.chineseLogicTv.setTextColor(getResources().getColor(R.color.white));
                allFragment.updateOthers();
            }
        });
```

```
        mDataBinding.mainLogicTv.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

mDataBinding.mainLogicTv.setTextColor(getResources().getColor(R.color.white));

mDataBinding.engLogicTv.setTextColor(getResources().getColor(R.color.white_60P));

mDataBinding.chineseLogicTv.setTextColor(getResources().getColor(R.color.white_60P));
                allFragment.updateAll();
            }
        });
    }

    @Override
    protected int initLayoutResId() { return R.layout.fragment_music;}

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

    @Override
    public void onDetach() {
        super.onDetach();
    }

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

    @Override
    public void onResume() {
        super.onResume();
        status = mPreferences.getString(getString(R.string.statusNightMode),"false");
        if (!firstTime) {
            if (status.equals("true"))
                allFragment.updateUI(status);
            else
                allFragment.updateUI(status);
        }
        firstTime = false;
    }
}
```

A.50 RootActivity.java

```
package com.example.fyp2_tsk.other;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.app.AppCompatActivity;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.SleepTimer.AppPreference;
import com.example.fyp2_tsk.SleepTimer.AuxiliaryPreference;
import com.example.fyp2_tsk.SleepTimer.ThemeEnum;

public class RootActivity extends AppCompatActivity {
    protected AppPreference appPreference;
    protected AuxiliaryPreference auxiliaryPreference;

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

        appPreference = new AppPreference(this);
        auxiliaryPreference = new AuxiliaryPreference(this);

        checkTheme();
    }

    protected void checkTheme() {
        ThemeEnum themeEnum = appPreference.getTheme();
        if (themeEnum == ThemeEnum.DARK) {
            this.setTheme(R.style.Theme_DARK);
        } else {
            this.setTheme(R.style.Theme_WHITE);
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

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

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

A.51 ThemeColor.java

```
package com.example.fyp2_tsk.skin;

public class ThemeColor {
    int drawableResId;
    String name;
    boolean isChosen = false;

    public ThemeColor(int drawableResId, String name) {
        this.drawableResId = drawableResId;
        this.name = name;
    }

    public int getDrawableResId() {
        return drawableResId;
    }

    public boolean isChosen() {
        return isChosen;
    }

    public void setChosen(boolean chosen) {
        isChosen = chosen;
    }

    public String getName() {
        return name;
    }

    public ThemeColor setName(String name) {
        this.name = name;
        return this;
    }
}
```

A.52 SplashActivity.java

```
package com.example.fyp2_tsk.other;

import android.animation.Animator;
import android.animation.AnimatorSet;
import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.content.Intent;
import android.databinding.DataBindingUtil;
import android.graphics.Color;
import android.graphics.drawable.GradientDrawable;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.DisplayMetrics;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.animation.AccelerateDecelerateInterpolator;
import android.view.animation.LinearInterpolator;
import android.widget.ImageView;
import android.widget.TextView;
import com.example.fyp2_tsk.databinding.ActivitySplashBinding;
import com.example.fyp2_tsk.utils.Utils;
import com.jaeger.library.StatusBarUtil;
import com.example.fyp2_tsk.R;
import java.util.Random;

public class SplashActivity extends AppCompatActivity {
    private ActivitySplashBinding mBinding;
    private TextView[] tv;
    private View container;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //create full screen Activity
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        StatusBarUtil.setColor(this, Color.TRANSPARENT);
        mBinding = DataBindingUtil.setContentView(this, R.layout.activity_splash);
```

```
        initViews();
    }

    private void goToMainActivity() {
        Intent intent = new Intent();
        intent.setClass(SplashActivity.this, MainActivity.class);
        startActivity(intent);
        SplashActivity.this.finish();
        overridePendingTransition(R.anim.fade_in, R.anim.fade_out); // switch scene using
fade in and out effects
    }

    private void initViews() {
        container = findViewById(R.id.splash_container);
        GradientDrawable gd = new GradientDrawable(GradientDrawable.Orientation.TL_BR, new
int[]{
                getResources().getColor(R.color.colorPrimary),
                getResources().getColor(R.color.colorPrimaryDark)
        });
        container.setBackground(gd);
        container.setClickable(false);

        tv = new TextView[]{
                (TextView) findViewById(R.id.splash_m),
                (TextView) findViewById(R.id.splash_u),
                (TextView) findViewById(R.id.splash_s),
                (TextView) findViewById(R.id.splash_i),
                (TextView) findViewById(R.id.splash_c),
                (TextView) findViewById(R.id.splash_p),
                (TextView) findViewById(R.id.splash_l),
                (TextView) findViewById(R.id.splash_a),
                (TextView) findViewById(R.id.splash_y),
                (TextView) findViewById(R.id.splash_e),
                (TextView) findViewById(R.id.splash_r)
        };

        tv[0].post(new Runnable() {
            @Override
            public void run() {
                for (TextView t : tv) {
                    t.setVisibility(View.VISIBLE);
                    startTextInAnim(t);
                }
            }
```

```
        });
    }

    private void startTextInAnim(TextView t) {
        Random r = new Random();
        DisplayMetrics metrics = Utils.getMetrics(this);
        int x = r.nextInt(metrics.widthPixels * 4 / 3);
        int y = r.nextInt(metrics.heightPixels * 4 / 3);
        float s = r.nextFloat() + 4.0f;
        ValueAnimator tranY = ObjectAnimator.ofFloat(t, "translationY", y - t.getY(), 0);
        ValueAnimator tranX = ObjectAnimator.ofFloat(t, "translationX", x - t.getX(), 0);
        ValueAnimator scaleX = ObjectAnimator.ofFloat(t, "scaleX", s, 1.0f);
        ValueAnimator scaleY = ObjectAnimator.ofFloat(t, "scaleY", s, 1.0f);
        ValueAnimator alpha = ObjectAnimator.ofFloat(t, "alpha", 0.0f, 1.0f);

        AnimatorSet set = new AnimatorSet();
        set.setDuration(1000); //char's animation period
        set.setInterpolator(new AccelerateDecelerateInterpolator());
        set.play(tranX).with(tranY).with(scaleX).with(scaleY).with(alpha);
        if (t == findViewById(R.id.splash_r)) {
            set.addListener(new Animator.AnimatorListener() {
                @Override
                public void onAnimationStart(Animator animation) {

                }

                @Override
                public void onAnimationEnd(Animator animation) {
                    startFinalAnim();
                }

                @Override
                public void onAnimationCancel(Animator animation) {

                }

                @Override
                public void onAnimationRepeat(Animator animation) {

                }
            });
        }
        set.start();
    }
```

```
    private void startFinalAnim() {
        final ImageView image = (ImageView) findViewById(R.id.splash_logo);
        final TextView name = (TextView) findViewById(R.id.splash_name);

        ValueAnimator alpha = ObjectAnimator.ofFloat(image, "alpha", 0.0f, 1.0f);
        alpha.setDuration(1000);

        ValueAnimator alphaN = ObjectAnimator.ofFloat(name, "alpha", 0.0f, 1.0f);
        alphaN.setDuration(1000);

        ValueAnimator tranY = ObjectAnimator.ofFloat(image, "translationY", -
image.getHeight() / 3, 0);
        tranY.setDuration(1000);

        ValueAnimator wait = ObjectAnimator.ofInt(0, 100);
        wait.setDuration(1000); //Stay period
        wait.addListener(new Animator.AnimatorListener() {
            @Override
            public void onAnimationStart(Animator animation) {
            }

            @Override
            public void onAnimationEnd(Animator animation) {
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        goToMainActivity();
                    }
                });
            }

            @Override
            public void onAnimationCancel(Animator animation) {

            }

            @Override
            public void onAnimationRepeat(Animator animation) {

            }
        });

        AnimatorSet set = new AnimatorSet();
```

```
        set.setInterpolator(new LinearInterpolator());
        set.addListener(new Animator.AnimatorListener() {
            @Override
            public void onAnimationStart(Animator animation) {
                image.setVisibility(View.VISIBLE);
                name.setVisibility(View.VISIBLE);
            }

            @Override
            public void onAnimationEnd(Animator animation) {

            }

            @Override
            public void onAnimationCancel(Animator animation) {

            }

            @Override
            public void onAnimationRepeat(Animator animation) {

            }
        });

        set.play(alpha).with(alphaN).with(tranY).before(wait);
        set.start();
    }
}
```

A.53 ShakeListener.java

```
package com.example.fyp2_tsk.ShakeControl;

import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;

public class ShakeListener implements SensorEventListener {

    private static final int SPEED_SHAKEHOLD = 2500;

    private static final int UPTATE_INTERVAL_TIME = 100;

    private SensorManager sensorManager;

    private Sensor sensor;

    private OnShakeListener onShakeListener;

    private Context mContext;

    private float lastX;
    private float lastY;
    private float lastZ;

    private long lastUpdateTime;

    public ShakeListener(Context mContext) {
        super();
        this.mContext = mContext;

        start();
    }

    public void start(){
        sensorManager=(SensorManager) mContext.getSystemService(Context.SENSOR_SERVICE);
        if(sensorManager!=null){

            sensor=sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        }
```

```
        if(sensor!=null){

            sensorManager.registerListener(this, sensor, SensorManager.SENSOR_DELAY_NORMAL);
        }
    }

    public void setOnShakeListener(OnShakeListener listener) {
        onShakeListener = listener;
    }

    public void stop(){
        sensorManager.unregisterListener(this);
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        // TODO Auto-generated method stub
        long currentUpdateTime=System.currentTimeMillis();

        long timeInterval=currentUpdateTime-lastUpdateTime;

        if(timeInterval<UPTATE_INTERVAL_TIME){
            return;
        }

        lastUpdateTime=currentUpdateTime;

        float x=event.values[0];
        float y=event.values[1];
        float z=event.values[2];

        float deltaX=x-lastX;
        float deltaY=y-lastY;
        float deltaZ=z-lastZ;

        lastX=x;
        lastY=y;
        lastZ=z;

        double
speed=Math.sqrt(deltaX*deltaX+deltaY*deltaY*deltaZ*deltaZ)/timeInterval*10000;

        if(speed>SPEED_SHAKEHOLD){
            onShakeListener.onShake();
```

```
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        // TODO Auto-generated method stub
    }

    public interface OnShakeListener {
         void onShake();
    }
}
```

A.54 ShakeService.java

```
package com.example.fyp2_tsk.ShakeControl;

import android.app.Service;
import android.content.Context;
import android.content.Intent;
import android.os.IBinder;
import android.os.Vibrator;
import android.util.Log;

public class ShakeService extends Service {
    private static final String TAG = "ShakeService";
    private ShakeListener mShakeListener;
    private Vibrator vibrator;

    @Override
    public void onCreate() {
        // TODO Auto-generated method stub
        super.onCreate();

        vibrator = (Vibrator) getBaseContext().getSystemService(Context.VIBRATOR_SERVICE);
        mShakeListener = new ShakeListener(getBaseContext());
        mShakeListener.setOnShakeListener(new ShakeListener.OnShakeListener() {

            @Override
            public void onShake() {
                // TODO Auto-generated method stub
                mShakeListener.stop();
                startVibrator();
                //vibrator.cancel();
                mShakeListener.start();
            }
        } );
    }


    void startVibrator(){
        Log.i(TAG,"shake");

        vibrator.vibrate(500);
    }

    @Override
```

```
    public IBinder onBind(Intent intent) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        // TODO Auto-generated method stub
        mShakeListener.start();
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public void onDestroy() {
        // TODO Auto-generated method stub
        super.onDestroy();
        mShakeListener.stop();
    }
}
```

A.55 ThemeColorAdapter.java

```
package com.example.fyp2_tsk.skin;

import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import com.example.fyp2_tsk.common.base.EasyRecyclerViewAdapter;
import com.example.fyp2_tsk.R;
import de.hdodenhof.circleimageview.CircleImageView;

public class ThemeColorAdapter extends EasyRecyclerViewAdapter<ThemeColor> {
    private int position;

    @Override
    public RecyclerView.ViewHolder onCreate(ViewGroup parent, int viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_theme_color, parent, false);
        return new ThemeColorViewHolder(view);
    }

    @Override
    public void onBind(final RecyclerView.ViewHolder viewHolder, final int RealPosition,
ThemeColor data) {
        ((ThemeColorViewHolder)
viewHolder).them_color.setImageResource(data.getDrawableResId());
        if (data.isChosen()) {
            ((ThemeColorViewHolder) viewHolder).chosen.setVisibility(View.VISIBLE);
            position=RealPosition;
        } else {
            ((ThemeColorViewHolder) viewHolder).chosen.setVisibility(View.GONE);
        }
    }


    class ThemeColorViewHolder extends EasyViewHolder {
        CircleImageView them_color;
        ImageView chosen;

        public ThemeColorViewHolder(View itemView) {
            super(itemView);
            them_color = (CircleImageView) itemView.findViewById(R.id.them_color);
```

```
            chosen = (ImageView) itemView.findViewById(R.id.choose);
        }
    }

    public int getPosition() {
        return position;
    }
}
```

A.56 BasePreference.java

```
package com.example.fyp2_tsk.SleepTimer;

import android.content.Context;
import android.content.SharedPreferences;

public class BasePreference {

    protected SharedPreferences.Editor editor;
    protected final SharedPreferences preferences;
    protected final Context context;

    public enum Preference {
        APP_PREFERENCE, //App settings
        AUXILIARY_PREFERENCE //Apply auxiliary settings, temporary data
    }

    public BasePreference(Context context, Preference which) {
        this.context = context;
        String name = which.toString().toLowerCase();
        this.preferences = context.getSharedPreferences(name, Context.MODE_PRIVATE);
    }
}
```

A.57 ThemeColorSelectDialog.java

```
package com.example.fyp2_tsk.skin;

import android.content.res.TypedArray;
import android.databinding.DataBindingUtil;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.DialogFragment;
import android.support.v7.widget.GridLayoutManager;
import android.util.DisplayMetrics;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import com.example.fyp2_tsk.common.base.EasyRecyclerViewAdapter;
import com.example.fyp2_tsk.common.utils.DensityUtils;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.databinding.DialogThemeColorBinding;
import java.util.ArrayList;
import skin.support.SkinCompatManager;

public class ThemeColorSelectDialog extends DialogFragment {
    private ThemeColorAdapter mThemeColorAdapter = new ThemeColorAdapter();
    private ArrayList<ThemeColor> mThemeColorList = new ArrayList<>();

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
@Nullable Bundle savedInstanceState) {

        getDialog().requestWindowFeature(Window.FEATURE_NO_TITLE);
        final Window window = getDialog().getWindow();
        final DisplayMetrics metrics = getResources().getDisplayMetrics();
        final DialogThemeColorBinding binding = DataBindingUtil.inflate(inflater,
R.layout.dialog_theme_color, ((ViewGroup) window.findViewById(android.R.id.content)),
false);

        binding.recyclerTheme.setLayoutManager(new GridLayoutManager(getActivity(), 4));

        init(binding);
        initListener(binding);
```

```
        window.setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));//注意此处
        window.setLayout(metrics.widthPixels - DensityUtils.dp2px(getContext(), 30), -2);
        return binding.getRoot();
    }

    private void init(DialogThemeColorBinding binding){
        TypedArray colors = getResources().obtainTypedArray(R.array.theme_colors);
        String[] names = getResources().getStringArray(R.array.theme_names);
        int titleLength = colors.length();
        mThemeColorList.clear();
        for( int i = 0; i < titleLength; i++ ){
            mThemeColorList.add(new ThemeColor(colors.getResourceId(i, 0)
                    , names[i]));
        }
        mThemeColorAdapter.setDatas(mThemeColorList);
        binding.recyclerTheme.setAdapter(mThemeColorAdapter);
    }

    private void initListener(DialogThemeColorBinding binding){
        mThemeColorAdapter.setOnItemClickListener(new
EasyRecyclerViewAdapter.OnItemClickListener() {
            @Override
            public void OnItemClick(View view, int position, Object data) {
                for (ThemeColor themeColor : mThemeColorList) {
                    themeColor.setChosen(false);
                }
                mThemeColorList.get(position).setChosen(true);
                mThemeColorAdapter.notifyDataSetChanged();
            }
        });
        binding.tvSure.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                SkinCompatManager.getInstance().loadSkin(
                        mThemeColorList.get(mThemeColorAdapter.getPosition()).getName(),
                        null, SkinCompatManager.SKIN_LOADER_STRATEGY_BUILD_IN);
                dismiss();
            }
        });
    }
}
```

A.58 AppPreference.java

```
package com.example.fyp2_tsk.SleepTimer;

import android.content.Context;
import android.os.Build;
import android.support.annotation.ColorInt;

import com.example.fyp2_tsk.R;

public class AppPreference extends BasePreference {

    public static final String KEY_THEME = "key_theme";
    public static final String KEY_THEME_ACTIONBAR_COLOR = "key_theme_actionbar_color";
    public static final String KEY_THEME_STATUS_BAR_COLOR = "key_theme_statusBar_color";
    public static final String KEY_THEME_ACCENT_COLOR = "key_theme_accent_color";

    public AppPreference(Context context) {
        super(context, Preference.APP_PREFERENCE);
    }

    public ThemeEnum getTheme() {
        String pa = preferences.getString(KEY_THEME, ThemeEnum.WHITE.name());
        return ThemeEnum.valueOf(pa);
    }

    @ColorInt
    public int getActionbarColor() {
        int deC;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            deC = context.getColor(R.color.colorPrimary);
        } else {
            deC = context.getResources().getColor(R.color.colorPrimary);
        }
        return preferences.getInt(KEY_THEME_ACTIONBAR_COLOR, deC);
    }

    @ColorInt
    public int getStatusBarColor() {
        int deC;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            deC = context.getColor(R.color.colorPrimary);
        } else {
            deC = context.getResources().getColor(R.color.colorPrimary);
```

```
        }
        return preferences.getInt(KEY_THEME_STATUS_BAR_COLOR, deC);
    }

    @ColorInt
    public int getAccentColor() {
        int deC;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            deC = context.getColor(R.color.colorPrimary);
        } else {
            deC = context.getResources().getColor(R.color.colorPrimary);
        }
        return preferences.getInt(KEY_THEME_ACCENT_COLOR, deC);
    }

    public void updateTheme(ThemeEnum themeEnum) {
        if (themeEnum == ThemeEnum.VARYING) return;

        editor = preferences.edit();
        editor.putString(KEY_THEME, themeEnum.name());
        editor.apply();
    }
}
```

A.59 AuxiliaryPreference.java

```
package com.example.fyp2_tsk.SleepTimer;
import android.content.Context;
public class AuxiliaryPreference extends BasePreference{
    public AuxiliaryPreference(Context context) {
        super(context, Preference.AUXILIARY_PREFERENCE);
    }
    //used as terminate the application regularly
    private class TimeSleep {
        static final String TIME_SLEEP_ENABLE = "time_sleep_enable";
        static final String TIME_SLEEP_START_TIME = "time_sleep_start_time";
        static final String TIME_SLEEP_DURATION = "time_sleep_duration";
    }
    //update the countdown time in real time
    public void updateTimeSleep(int duration) {
        editor = preferences.edit();
        editor.putBoolean(TimeSleep.TIME_SLEEP_ENABLE, true);
        editor.putInt(TimeSleep.TIME_SLEEP_DURATION, duration);
        editor.putLong(TimeSleep.TIME_SLEEP_START_TIME, System.currentTimeMillis());
        editor.apply();
    }
    //call when countdown time end to set sleep time into disable status
    public void setTimeSleepDisable() {
        editor = preferences.edit();
        editor.putBoolean(TimeSleep.TIME_SLEEP_ENABLE, false);
        editor.putInt(TimeSleep.TIME_SLEEP_DURATION, -1);
        editor.putLong(TimeSleep.TIME_SLEEP_START_TIME, System.currentTimeMillis());
        editor.apply();
    }

    public boolean getTimeSleepEnable() {
        return preferences.getBoolean(TimeSleep.TIME_SLEEP_ENABLE, false);
    }

    public int getTimeSleepDuration() {
        return preferences.getInt(TimeSleep.TIME_SLEEP_DURATION, -1);
    }

    public long getTimeSleepStartTime() {
        return preferences.getLong(TimeSleep.TIME_SLEEP_START_TIME, -1);
    }
}
```

A.60 BroadcastManager.java

```
package com.example.fyp2_tsk.SleepTimer;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.support.annotation.Nullable;

public class BroadcastManager {
    // show countdown time left in the left navigation bar
    public static final String FILTER_APP_QUIT_TIME_COUNTDOWN =
"filter_app_quit_time_countdown";
    private static BroadcastManager mInstance;

    public static final class Countdown {
        public static final String APP_QUIT_TIME_COUNTDOWN_STATUS =
"app_quit_time_countdown_status";
        public static final int START_COUNTDOWN = 1;
        public static final int STOP_COUNTDOWN = 2;
    }
    private BroadcastManager() {
    }

    public static BroadcastManager getInstance() {
        if (mInstance == null) {
            mInstance = new BroadcastManager();
        }
        return mInstance;
    }

    public void registerBroadReceiver(Context context, BroadcastReceiver receiver, String
identity) {
        IntentFilter filter = new IntentFilter(identity);
        context.registerReceiver(receiver, filter);
    }

    public void sendBroadcast(Context context, String identity, @Nullable Bundle extras) {
        Intent intent = new Intent();
        if (extras != null) {
            intent.putExtras(extras);
        }
```

```
        intent.setAction(identity);
        context.sendBroadcast(intent);
    }

    public void unregisterReceiver(Context context, BroadcastReceiver receiver) {
        context.unregisterReceiver(receiver);
    }

    // Theme change
    public static final String FILTER_APP_THEME_CHANGE_AUTOMATIC =
"filter_app_theme_change_automatic";
    public static final String APP_THEME_CHANGE_AUTOMATIC_TOKEN =
"filter_app_theme_change_automatic_token";
    public static final int APP_THEME_CHANGE_AUTOMATIC_WHITE = 1;
}
```

A.61 ThemeChangeable.java

```
package com.example.fyp2_tsk.SleepTimer;

public interface ThemeChangeable {
    void themeChange(ThemeEnum themeEnum, int[] colors);
}
```

A.62 Common.java

```
package com.example.fyp2_tsk.utils;

import com.example.fyp2_tsk.entity.Music;
import java.util.ArrayList;
import java.util.List;

public class Common {
    public static List<Music> musicList = new ArrayList<>();
    public static List<Music> musicList_backup = new ArrayList<>();
}
```

A.63 PeriodicTask.java

```
package com.example.fyp2_tsk.SleepTimer;

import java.util.Timer;
import java.util.TimerTask;

public class PeriodicTask {

    private TimerTask progressUpdateTask;
    private final Task task;
    private final int period;
    private boolean isSchedule = false;

    public interface Task {
       ////Perform periodic tasks, note that this method is not executed on the main thread,
and cannot access UI controls
        void execute();
    }

    //Perform periodic tasks
    public PeriodicTask(Task task, int period) {
        this.task = task;
        this.period = period;
    }

    public void stop() {
        if (!isSchedule()) {
            return;
        }

        if (progressUpdateTask != null) {
            progressUpdateTask.cancel();
            isSchedule = false;
        }
    }

    public void start() {
        if (isSchedule()) {
            return;
        }

        Timer timer = new Timer();
        progressUpdateTask = new TimerTask() {
```

```
            @Override
            public void run() {
                task.execute();
            }
        };
        timer.schedule(progressUpdateTask, 0, period);
        isSchedule = true;
    }

    public boolean isSchedule() {
        return isSchedule;
    }
}
```

A.64 Utils.java

```
package com.example.fyp2_tsk.utils;

import android.app.Activity;
import android.util.DisplayMetrics;
import android.view.Display;

public class Utils {
    public static DisplayMetrics getMetrics(Activity activity) {
        Display display = activity.getWindowManager().getDefaultDisplay();
        DisplayMetrics metrics = new DisplayMetrics();
        display.getMetrics(metrics);
        return metrics;
    }
}
```

A.65 TimeSleepActivity.java

```
package com.example.fyp2_tsk.SleepTimer;

import android.content.res.ColorStateList;
import android.graphics.Color;
import android.os.Build;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.CompoundButton;
import android.widget.NumberPicker;
import android.widget.Switch;
import android.widget.TextView;
import android.widget.Toast;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.other.RootActivity;
import com.example.fyp2_tsk.utils.AnimationUtils;
import com.example.fyp2_tsk.utils.ColorUtils;

public class TimeSleepActivity extends RootActivity implements ThemeChangeable {
    private ViewHolder viewHolder;
    private NumberPickerHolder numberPickerHolder;
    private Toolbar toolbar;
    private boolean enable;
    private int time;

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

        initToolbar();

    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        viewHolder = null;
```

```
        numberPickerHolder = null;
    }

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

        viewHolder = new ViewHolder();
        numberPickerHolder = new NumberPickerHolder();

        viewHolder.initViews();
        themeChange(null, null);
        initData();
    }

    private void initData() {
        // True if it has been set and the time has not yet arrived
        enable = auxiliaryPreference.getTimeSleepEnable();
        if (enable) {
            time = auxiliaryPreference.getTimeSleepDuration();
        } else {
            time = 10;
        }
        viewHolder.initData(enable);
    }

    private void initToolbar() {
        toolbar = (Toolbar) findViewById(R.id.time_sleep_toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setTitle(getResources().getString(R.string.main_tab_sleep));
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        final int[] ta = ColorUtils.get2ActionStatusBarColors(this);
        toolbar.setBackgroundColor(ta[0]);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            getWindow().setStatusBarColor(ta[0]);
        }

    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_save, menu);
        return super.onCreateOptionsMenu(menu);
```

```
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                finish();
                break;
            case R.id.action_save:
                if (time == 0) {
                    String msg =
getString(R.string.error_time_sleep_time_must_more_then_zero);
                    Toast.makeText(TimeSleepActivity.this, msg, Toast.LENGTH_SHORT).show();
                } else {
                    if (enable) {
                        handleSave();
                    } else {
                        handleCancelIfEnableBefore();
                    }
                }
                break;
            default:
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    private void handleCancelIfEnableBefore() {
        boolean enableBefore = auxiliaryPreference.getTimeSleepEnable();
        if (enableBefore) {
            Bundle bundle = new Bundle();
            bundle.putInt(BroadcastManager.Countdown.APP_QUIT_TIME_COUNTDOWN_STATUS,
BroadcastManager.Countdown.STOP_COUNTDOWN);
            BroadcastManager.getInstance().sendBroadcast(this,
BroadcastManager.FILTER_APP_QUIT_TIME_COUNTDOWN, bundle);

            String msg = getString(R.string.info_time_sleep_is_canceled);
            Toast.makeText(TimeSleepActivity.this, msg, Toast.LENGTH_SHORT).show();
        }
        finish();
    }

    private void handleSave() {
        auxiliaryPreference.updateTimeSleep(time);
```

```
        String ti = getString(R.string.replace_time_sleep_app_quit_at);
        String re;
        if (time > 60) {
            re = time / 60 + " " + getString(R.string.hour) + " " + time % 60 + " " +
getString(R.string.minute);
        } else {
            re = time + " " + getString(R.string.minute);
        }
        String str = ti.replace("*", re);
        Toast.makeText(this, str, Toast.LENGTH_LONG).show();
        Bundle bundle = new Bundle();
        bundle.putInt(BroadcastManager.Countdown.APP_QUIT_TIME_COUNTDOWN_STATUS,
BroadcastManager.Countdown.START_COUNTDOWN);
        BroadcastManager.getInstance()
                .sendBroadcast(this, BroadcastManager.FILTER_APP_QUIT_TIME_COUNTDOWN,
bundle);

        finish();
    }

    @Override
    public void themeChange(ThemeEnum themeEnum, int[] colors) {

        int acc = ColorUtils.getAccentColor(this);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            viewHolder.custom.setBackgroundTintList(ColorStateList.valueOf(acc));
        } else {
            viewHolder.custom.setDrawingCacheBackgroundColor(acc);
        }

        int tcs[] = ColorUtils.get2ThemeTextColor(this, appPreference.getTheme());
        int mC = tcs[0];
        viewHolder.status.setTextColor(mC);
        for (View v : viewHolder.vs) {
            ((TextView) v).setTextColor(mC);
        }
        viewHolder.line.setBackgroundColor(tcs[1]);
        viewHolder.show.setTextColor(mC);

    }

    private class ViewHolder implements
            CompoundButton.OnCheckedChangeListener {
        TextView status;
```

```
        Switch cwitch;
        TextView m10;
        TextView m20;
        TextView m30;
        TextView m45;
        TextView m60;
        Button custom;
        TextView show;
        View line;

        View[] vs;

        final int color;

        public ViewHolder() {
            color = ColorUtils.getAccentColor(TimeSleepActivity.this);
        }

        View.OnClickListener checkListener = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                time = tagToInt(v);

                int dur = getResources().getInteger(R.integer.anim_default_duration) * 2 /
3;
                AnimationUtils.startScaleAnim(v, dur, null, 0.7f, 1.0f);

                updateCurrentTimeText();
                updateSelected();
            }
        };

        View.OnClickListener customListener = new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!numberPickerHolder.custom) {
                    numberPickerHolder.init();
                    AnimationUtils.startScaleAnim(v, 300, null, 1.0f, 0.0f);
                }
            }
        };

        void initViews() {
            status = (TextView) findViewById(R.id.time_sleep_status);
```

```
        cwitch = (Switch) findViewById(R.id.time_sleep_status_switch);
        m10 = (TextView) findViewById(R.id.time_sleep_10m);
        m20 = (TextView) findViewById(R.id.time_sleep_20m);
        m30 = (TextView) findViewById(R.id.time_sleep_30m);
        m45 = (TextView) findViewById(R.id.time_sleep_45m);
        m60 = (TextView) findViewById(R.id.time_sleep_60m);
        custom = (Button) findViewById(R.id.time_sleep_custom);
        show = (TextView) findViewById(R.id.time_sleep_custom_show);
        line = findViewById(R.id.time_sleep_line);

        vs = new View[]{
                m10,
                m20,
                m30,
                m45,
                m60,
        };

        cwitch.setOnCheckedChangeListener(this);
        m10.setOnClickListener(checkListener);
        m20.setOnClickListener(checkListener);
        m30.setOnClickListener(checkListener);
        m45.setOnClickListener(checkListener);
        m60.setOnClickListener(checkListener);

        custom.setOnClickListener(customListener);

    }

    private void initTexts() {
        String s = getString(R.string.minute);
        int dur = getResources().getInteger(R.integer.anim_default_duration);
        for (View v : vs) {
            ((TextView) v).setText(tagToInt(v) + " " + s);
            AnimationUtils.startScaleAnim(v, dur, null, 0.0f, 1.0f);
            AnimationUtils.startAlphaAnim(v, dur, null, 0.0f, enable ? 1.0f : 0.4f);
        }
    }

    private void initData(boolean enable) {
        cwitch.setChecked(enable);
        initTexts();
        updateCurrentTimeText();
        updateSelected();
```

```
            setEnable(enable);
        }

        void setEnable(boolean enable) {
            String en = enable ? getString(R.string.enable) : getString(R.string.disable);
            status.setText(en);
            for (View v : vs) {
                v.setEnabled(enable);
            }
            custom.setEnabled(enable);
            show.setEnabled(enable);

            float alpha = 1.0f;
            if (!enable) {
                alpha = 0.4f;
            }
            for (View v : vs) {
                v.setAlpha(alpha);
            }
            custom.setAlpha(alpha);
            show.setAlpha(alpha);

            if (numberPickerHolder.custom) {
                numberPickerHolder.setEnable(enable);
            }

        }

        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            enable = isChecked;
            setEnable(isChecked);
        }

        private void updateSelected() {
            for (View view : vs) {
                int tag = tagToInt(view);
                if (tag == time) {
                    view.setBackgroundColor(color);
                } else {
                    view.setBackgroundColor(Color.TRANSPARENT);
                }
            }
        }
```

```
    void updateCurrentTimeText() {
        String txt = getString(R.string.replace_time_sleep);
        String s = txt.replace("*", String.valueOf(time));
        show.setText(s);
    }

    int tagToInt(View v) {
        String tag = (String) v.getTag();
        int r = 0;
        if (tag != null && !TextUtils.isEmpty(tag)) {
            r = Integer.valueOf(tag);
        }
        return r;
    }
}

private class NumberPickerHolder implements
        NumberPicker.OnValueChangeListener {

    TextView tHour;
    TextView tMinute;
    NumberPicker minute;
    NumberPicker hour;

    boolean custom = false;

    void init() {
        custom = true;
        View v = findViewById(R.id.time_sleep_pickers);
        v.setVisibility(View.VISIBLE);

        tHour = (TextView) findViewById(R.id.time_sleep_custom_num_left_t);
        tMinute = (TextView) findViewById(R.id.time_sleep_custom_num_right_t);
        minute = (NumberPicker) findViewById(R.id.time_sleep_custom_num_right);
        hour = (NumberPicker) findViewById(R.id.time_sleep_custom_num_left);
        minute.setOnValueChangedListener(this);
        hour.setOnValueChangedListener(this);

        initTheme();
        initPickersData();
    }

    private void initPickersData() {
```

```
            minute.setMinValue(0);
            hour.setMinValue(0);
            minute.setMaxValue(59);
            hour.setMaxValue(5);
            hour.setValue(0);
            minute.setValue(time);
        }

        private void initTheme() {
            int[] cs = ColorUtils.get2ThemeTextColor(TimeSleepActivity.this,
appPreference.getTheme());
            tHour.setTextColor(cs[0]);
            tMinute.setTextColor(cs[0]);
        }

        @Override
        public void onValueChange(NumberPicker picker, int oldVal, int newVal) {
            int min = minute.getValue();
            int hou = hour.getValue();
            time = hou * 60 + min;
            viewHolder.updateCurrentTimeText();
        }

        public void setEnable(boolean enable) {
            tHour.setEnabled(enable);
            tMinute.setEnabled(enable);
            minute.setEnabled(enable);
            hour.setEnabled(enable);

            float alpha = 1.0f;
            if (!enable) {
                alpha = 0.4f;
            }
            tHour.setAlpha(alpha);
            tMinute.setAlpha(alpha);
            minute.setAlpha(alpha);
            hour.setAlpha(alpha);
        }
    }

}
```

A.66 AnimationUtils.java

```
package com.example.fyp2_tsk.utils;

import android.animation.Animator;
import android.animation.AnimatorSet;
import android.animation.ObjectAnimator;
import android.animation.ValueAnimator;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.View;
import android.view.animation.OvershootInterpolator;
public class AnimationUtils {
    public static void startAlphaAnim(@NonNull final View view, int duration, @Nullable
Animator.AnimatorListener listener, float... values) {
        ValueAnimator alphaAnim = ObjectAnimator.ofFloat(values);
        alphaAnim.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
            @Override
            public void onAnimationUpdate(ValueAnimator animation) {
                float alpha = (float) animation.getAnimatedValue();
                view.setAlpha(alpha);
            }
        });
        if (listener != null) {
            alphaAnim.addListener(listener);
        }
        alphaAnim.setDuration(duration);
        alphaAnim.start();
    }

    public static void startScaleAnim(@NonNull View view, int duration, @Nullable
Animator.AnimatorListener listener, float... values) {
        ValueAnimator animSX = ObjectAnimator.ofFloat(view, "scaleX", values);
        ValueAnimator animSY = ObjectAnimator.ofFloat(view, "scaleY", values);
        AnimatorSet set = new AnimatorSet();
        set.setDuration(duration);
        set.setInterpolator(new OvershootInterpolator());
        set.play(animSX).with(animSY);
        if (listener != null) {
            set.addListener(listener);
        }
        set.start();
    }
}
```

A.67 BlurUtil.java

```
package com.example.fyp2_tsk.utils;

import android.graphics.Bitmap;

public class BlurUtil {
    private static final String TAG = BlurUtil.class.getSimpleName();
    /**
     * Frosted glass of the picture
     * @param radius BLur level
     */
    public static Bitmap doBlur(Bitmap sentBitmap, int radius, boolean canReuseInBitmap) {

        Bitmap bitmap;
        if (canReuseInBitmap) {
            bitmap = sentBitmap;
        } else {
            bitmap = sentBitmap.copy(sentBitmap.getConfig(), true);
        }

        if (radius < 1) {
            return (null);
        }

        int w = bitmap.getWidth();
        int h = bitmap.getHeight();

        int[] pix = new int[w * h];
        bitmap.getPixels(pix, 0, w, 0, 0, w, h);

        int wm = w - 1;
        int hm = h - 1;
        int wh = w * h;
        int div = radius + radius + 1;

        int r[] = new int[wh];
        int g[] = new int[wh];
        int b[] = new int[wh];
        int rsum, gsum, bsum, x, y, i, p, yp, yi, yw;
        int vmin[] = new int[Math.max(w, h)];

        int divsum = (div + 1) >> 1;
        divsum *= divsum;
```

```
        int dv[] = new int[256 * divsum];
        for (i = 0; i < 256 * divsum; i++) {
            dv[i] = (i / divsum);
        }

        yw = yi = 0;

        int[][] stack = new int[div][3];
        int stackpointer;
        int stackstart;
        int[] sir;
        int rbs;
        int r1 = radius + 1;
        int routsum, goutsum, boutsum;
        int rinsum, ginsum, binsum;

        for (y = 0; y < h; y++) {
            rinsum = ginsum = binsum = routsum = goutsum = boutsum = rsum = gsum = bsum = 0;
            for (i = -radius; i <= radius; i++) {
                p = pix[yi + Math.min(wm, Math.max(i, 0))];
                sir = stack[i + radius];
                sir[0] = (p & 0xff0000) >> 16;
                sir[1] = (p & 0x00ff00) >> 8;
                sir[2] = (p & 0x0000ff);
                rbs = r1 - Math.abs(i);
                rsum += sir[0] * rbs;
                gsum += sir[1] * rbs;
                bsum += sir[2] * rbs;
                if (i > 0) {
                    rinsum += sir[0];
                    ginsum += sir[1];
                    binsum += sir[2];
                } else {
                    routsum += sir[0];
                    goutsum += sir[1];
                    boutsum += sir[2];
                }
            }
            stackpointer = radius;

            for (x = 0; x < w; x++) {

                r[yi] = dv[rsum];
                g[yi] = dv[gsum];
```

```
            b[yi] = dv[bsum];

            rsum -= routsum;
            gsum -= goutsum;
            bsum -= boutsum;

            stackstart = stackpointer - radius + div;
            sir = stack[stackstart % div];

            routsum -= sir[0];
            goutsum -= sir[1];
            boutsum -= sir[2];

            if (y == 0) {
                vmin[x] = Math.min(x + radius + 1, wm);
            }
            p = pix[yw + vmin[x]];

            sir[0] = (p & 0xff0000) >> 16;
            sir[1] = (p & 0x00ff00) >> 8;
            sir[2] = (p & 0x0000ff);

            rinsum += sir[0];
            ginsum += sir[1];
            binsum += sir[2];

            rsum += rinsum;
            gsum += ginsum;
            bsum += binsum;

            stackpointer = (stackpointer + 1) % div;
            sir = stack[(stackpointer) % div];

            routsum += sir[0];
            goutsum += sir[1];
            boutsum += sir[2];

            rinsum -= sir[0];
            ginsum -= sir[1];
            binsum -= sir[2];

            yi++;
        }
        yw += w;
```

```
        }
        for (x = 0; x < w; x++) {
            rinsum = ginsum = binsum = routsum = goutsum = boutsum = rsum = gsum = bsum = 0;
            yp = -radius * w;
            for (i = -radius; i <= radius; i++) {
                yi = Math.max(0, yp) + x;

                sir = stack[i + radius];

                sir[0] = r[yi];
                sir[1] = g[yi];
                sir[2] = b[yi];

                rbs = r1 - Math.abs(i);

                rsum += r[yi] * rbs;
                gsum += g[yi] * rbs;
                bsum += b[yi] * rbs;

                if (i > 0) {
                    rinsum += sir[0];
                    ginsum += sir[1];
                    binsum += sir[2];
                } else {
                    routsum += sir[0];
                    goutsum += sir[1];
                    boutsum += sir[2];
                }

                if (i < hm) {
                    yp += w;
                }
            }
            yi = x;
            stackpointer = radius;
            for (y = 0; y < h; y++) {
                // Preserve alpha channel: ( 0xff000000 & pix[yi] )
                pix[yi] = (0xff000000 & pix[yi]) | (dv[rsum] << 16) | (dv[gsum] << 8) |
dv[bsum];

                rsum -= routsum;
                gsum -= goutsum;
                bsum -= boutsum;
```

```
                stackstart = stackpointer - radius + div;
                sir = stack[stackstart % div];

                routsum -= sir[0];
                goutsum -= sir[1];
                boutsum -= sir[2];

                if (x == 0) {
                    vmin[y] = Math.min(y + r1, hm) * w;
                }
                p = x + vmin[y];

                sir[0] = r[p];
                sir[1] = g[p];
                sir[2] = b[p];

                rinsum += sir[0];
                ginsum += sir[1];
                binsum += sir[2];

                rsum += rinsum;
                gsum += ginsum;
                bsum += binsum;

                stackpointer = (stackpointer + 1) % div;
                sir = stack[stackpointer];

                routsum += sir[0];
                goutsum += sir[1];
                boutsum += sir[2];

                rinsum -= sir[0];
                ginsum -= sir[1];
                binsum -= sir[2];

                yi += w;
            }
        }

        bitmap.setPixels(pix, 0, w, 0, 0, w, h);

        return (bitmap);
    }
```

```
    public static Bitmap doBlur(Bitmap originBitmap, int scaleRatio, int blurRadius){
        Bitmap scaledBitmap = Bitmap.createScaledBitmap(originBitmap,
                originBitmap.getWidth() / scaleRatio,
                originBitmap.getHeight() / scaleRatio,
                false);
        Bitmap blurBitmap = doBlur(scaledBitmap, blurRadius, false);
        scaledBitmap.recycle();
        return blurBitmap;
    }
}
```

A.68 ColorUtils.java

```
package com.example.fyp2_tsk.utils;

import android.content.Context;
import android.os.Build;
import com.example.fyp2_tsk.R;
import com.example.fyp2_tsk.SleepTimer.AppPreference;
import com.example.fyp2_tsk.SleepTimer.ThemeEnum;

public class ColorUtils {
     //0 background color of Status bar, 1 background color of Title bar
    public static int[] get2ActionStatusBarColors(Context context) {
        AppPreference preference = new AppPreference(context);

        int actionColor;
        int statusColor;

        if (preference.getTheme() != ThemeEnum.DARK) {
            actionColor = preference.getActionbarColor();
            statusColor = preference.getStatusBarColor();
        } else {
            statusColor = context.getResources().getColor(R.color.theme_dark_primary);
            actionColor = context.getResources().getColor(R.color.theme_dark_primary_dark);
        }

        return new int[]{statusColor, actionColor};
    }

    public static int getAccentColor(Context context) {
        AppPreference preference = new AppPreference(context);
        if (preference.getTheme() != ThemeEnum.DARK) {
            return preference.getAccentColor();
        } else {
            return context.getResources().getColor(R.color.theme_dark_accent);
        }
    }

    //0 primary font color, 1 secondary font color
    public static int[] get2WhiteThemeTextColor(Context context) {
        int[] colors = new int[2];

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            colors[0] = context.getColor(R.color.theme_white_main_text);
```

```
            colors[1] = context.getColor(R.color.theme_white_vic_text);
        } else {
            colors[0] = context.getResources().getColor(R.color.theme_white_main_text);
            colors[1] = context.getResources().getColor(R.color.theme_white_vic_text);
        }

        return colors;
    }

    //0 primary font color, 1 secondary font color
    public static int[] get2ThemeTextColor(Context context, ThemeEnum themeEnum) {
        if (themeEnum == ThemeEnum.DARK) {
            return get2DarkThemeTextColor(context);
        } else {
            return get2WhiteThemeTextColor(context);
        }
    }

    //0 primary font color, 1 secondary font color
    public static int[] get2DarkThemeTextColor(Context context) {
        int[] colors = new int[2];

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            colors[0] = context.getColor(R.color.theme_dark_main_text);
            colors[1] = context.getColor(R.color.theme_dark_vic_text);
        } else {
            colors[0] = context.getResources().getColor(R.color.theme_dark_main_text);
            colors[1] = context.getResources().getColor(R.color.theme_dark_vic_text);
        }
        return colors;
    }


    /**
     * 0 Status bar background color <br>
     * 1 Background color of title bar <br>
     * 2 Preferred color of controls <br>
     * 3 main background color <br>
     * 4 auxiliary background color <br>
     * 5 main font color <br>
     * 6 auxiliary font color <br>
     * 7 bottom navigation background color <br>
     * 8 Main font color of title bar <br>
     * 9 Supplementary font color of title bar <br>
     */
```

```
    public static int[] get10ThemeColors(Context context, ThemeEnum themeEnum) {
        if (themeEnum == ThemeEnum.WHITE || themeEnum == ThemeEnum.VARYING) {
            return get10WhiteThemeColors(context);
        } else {
            return get10DarkThemeColors(context);
        }
    }

    //Day mode
    public static int[] get10WhiteThemeColors(Context context) {
        int[] colors = new int[10];
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            colors[0] = context.getColor(R.color.theme_white_primary);
            colors[1] = context.getColor(R.color.theme_white_primary_dark);
            colors[2] = context.getColor(R.color.theme_white_accent);
            colors[3] = context.getColor(R.color.theme_white_main_bg);
            colors[4] = context.getColor(R.color.theme_white_vic_bg);
            colors[5] = context.getColor(R.color.theme_white_main_text);
            colors[6] = context.getColor(R.color.theme_white_vic_text);
            colors[7] = context.getColor(R.color.theme_white_nav);
            colors[8] = context.getColor(R.color.theme_white_toolbar_main_text);
            colors[9] = context.getColor(R.color.theme_white_toolbar_vic_text);
        } else {
            colors[0] = context.getResources().getColor(R.color.theme_white_primary);
            colors[1] = context.getResources().getColor(R.color.theme_white_primary_dark);
            colors[2] = context.getResources().getColor(R.color.theme_white_accent);
            colors[3] = context.getResources().getColor(R.color.theme_white_main_bg);
            colors[4] = context.getResources().getColor(R.color.theme_white_vic_bg);
            colors[5] = context.getResources().getColor(R.color.theme_white_main_text);
            colors[6] = context.getResources().getColor(R.color.theme_white_vic_text);
            colors[7] = context.getResources().getColor(R.color.theme_white_nav);
            colors[8] =
context.getResources().getColor(R.color.theme_white_toolbar_main_text);
            colors[9] =
context.getResources().getColor(R.color.theme_white_toolbar_vic_text);
        }

        AppPreference preference = new AppPreference(context);
        colors[2] = preference.getAccentColor();

        return colors;
    }

    //Night Mode
```

```
    public static int[] get10DarkThemeColors(Context context) {
        int[] colors = new int[10];
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            colors[0] = context.getColor(R.color.theme_dark_primary);
            colors[1] = context.getColor(R.color.theme_dark_primary_dark);
            colors[2] = context.getColor(R.color.theme_dark_accent);
            colors[3] = context.getColor(R.color.theme_dark_main_bg);
            colors[4] = context.getColor(R.color.theme_dark_vic_bg);
            colors[5] = context.getColor(R.color.theme_dark_main_text);
            colors[6] = context.getColor(R.color.theme_dark_vic_text);
            colors[7] = context.getColor(R.color.theme_dark_nav);
            colors[8] = context.getColor(R.color.theme_dark_toolbar_main_text);
            colors[9] = context.getColor(R.color.theme_dark_toolbar_vic_text);
        } else {
            colors[0] = context.getResources().getColor(R.color.theme_dark_primary);
            colors[1] = context.getResources().getColor(R.color.theme_dark_primary_dark);
            colors[2] = context.getResources().getColor(R.color.theme_dark_accent);
            colors[3] = context.getResources().getColor(R.color.theme_dark_main_bg);
            colors[4] = context.getResources().getColor(R.color.theme_dark_vic_bg);
            colors[5] = context.getResources().getColor(R.color.theme_dark_main_text);
            colors[6] = context.getResources().getColor(R.color.theme_dark_vic_text);
            colors[7] = context.getResources().getColor(R.color.theme_dark_nav);
            colors[8] =
context.getResources().getColor(R.color.theme_dark_toolbar_main_text);
            colors[9] =
context.getResources().getColor(R.color.theme_dark_toolbar_vic_text);
        }
        return colors;
    }
}
```

A.69 MergeImage.java

```
package com.example.fyp2_tsk.utils;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffXfermode;

public class MergeImage {
        /**
         * Combine disc pictures
         * @param discBitmap  Vinyl disc base map
         * @param albumBitmap Album cover image
         * @return
         */
        public static Bitmap mergeThumbnailBitmap(Bitmap discBitmap, Bitmap albumBitmap) {
            //Get the width and height of the vinyl disc
            int w = discBitmap.getWidth();
            int h = discBitmap.getHeight();
            //According to the width and height of the vinyl disc base image, zoom the album
picture
            albumBitmap = Bitmap.createScaledBitmap(albumBitmap, w, h, true);
            Bitmap bm = Bitmap.createBitmap(w, h, Bitmap.Config.ARGB_8888);
            Canvas canvas = new Canvas(bm);
            Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
            //Here need to draw a circle
            canvas.drawCircle(w / 2, h / 2, w / 3 +10, paint);
            //After the circle is drawn, reset the brush
            paint.reset();
            //Set the image synthesis mode, which is to draw the source image only where the
source image and the target image intersect
            paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
            canvas.drawBitmap(albumBitmap, 0, 0, paint);
            paint.reset();
            canvas.drawBitmap(discBitmap, 0, 0, null);
            return bm;
        }
    }
```

A.70 activity_driver.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="com.example.fyp2_tsk.other.MusicActivity">

    <ImageView
        android:id="@+id/music_bg_imgv_driver"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        android:src="@mipmap/touxiang1" />

    <RelativeLayout
        android:id="@+id/music_relativelayout_driver"
        android:layout_width="match_parent"
        android:layout_height="50dp">

        <ImageView
            android:id="@+id/music_down_imgv_driver"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_alignParentLeft="true"
            android:layout_margin="5dp"
            android:src="@mipmap/ic_arrow_down" />
    </RelativeLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/music_relativelayout"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true">

        <TextView
            android:id="@+id/music_title_tv_driver"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="100dp"
```

```
            android:text="周杰伦"
            android:gravity="center"
            android:textColor="#ffffff"
            android:textSize="25dp" />

        <TextView
            android:id="@+id/music_artist_tv_driver"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/music_title_tv_driver"
            android:layout_marginTop="10dp"
            android:gravity="center"
            android:text="告白气球"
            android:textColor="#ffffff"
            android:textSize="20dp" />

        <ImageView
            android:id="@+id/music_prev_imgv_driver"
            android:layout_width="125dp"
            android:layout_height="200dp"
            android:layout_centerVertical="true"
            android:src="@drawable/seach_btn_pre" />

        <ImageView
            android:id="@+id/music_pause_imgv_driver"
            android:layout_width="150dp"
            android:layout_height="200dp"
            android:layout_centerHorizontal="true"
            android:layout_centerVertical="true"
            android:src="@drawable/seach_btn_pause" />

        <ImageView
            android:id="@+id/music_next_imgv_driver"
            android:layout_width="175dp"
            android:layout_height="200dp"
            android:layout_centerVertical="true"
            android:layout_toRightOf="@id/music_pause_imgv_driver"
            android:src="@drawable/seach_btn_next" />

    </RelativeLayout>

</RelativeLayout>
```

A.71 activity_filechooser_show.xml

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

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:fitsSystemWindows="true">

        <skin.support.widget.SkinCompatToolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            app:navigationIcon="@mipmap/toolbar_back"
            app:title="Select Music"
            app:titleTextColor="#ffffff" />

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="#ffffff"
            android:orientation="horizontal"
            android:padding="5.0dip">
            <RelativeLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="10dp"
                android:layout_marginRight="10dp"
                android:layout_alignParentBottom="true"
                android:id="@+id/fl_main">
                <com.wang.avi.AVLoadingIndicatorView
                    android:id="@+id/avi_loading"
                    style="@style/AVLoadingIndicatorView"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:visibility="gone"
                    android:layout_centerVertical="true"
                    app:indicatorColor="@color/cutter_theme_black"
                    app:indicatorName="PacmanIndicator"/>
                <android.support.design.widget.FloatingActionButton
                    android:id="@+id/btn_update"
```

```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:src="@mipmap/btn_update" />
        </RelativeLayout>


        <android.support.v7.widget.RecyclerView
            android:id="@+id/rl_musice"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:clickable="true"
            android:columnWidth="90dp"
            android:horizontalSpacing="10dp"
            android:numColumns="auto_fit"
            android:stretchMode="columnWidth"
            android:verticalSpacing="10dp"
            android:layout_above="@id/fl_main"/>


    </RelativeLayout>
</LinearLayout>
</layout>
```

A.72 activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
        <android.support.design.widget.AppBarLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:fitsSystemWindows="true"
            android:orientation="vertical">

            <!--Toolbar-->
            <skin.support.widget.SkinCompatToolbar
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:fitsSystemWindows="true"
                android:minHeight="?attr/actionBarSize"
                android:popupTheme="@style/ThemeOverlay.AppCompat.Light"
                app:theme="@style/ThemeOverlay.AppCompat.ActionBar" />

            <!--DrawerLayout-->
            <android.support.v4.widget.DrawerLayout
                android:id="@+id/drawerlayout"
                android:layout_width="match_parent"
                android:layout_height="match_parent">
                <!--MainLayout-->
                <FrameLayout
                    android:id="@+id/frame_content"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:layout_below="@+id/appbar"
                    android:scrollbars="none"
                    app:layout_behavior="@string/appbar_scrolling_view_behavior" />
                <android.support.design.widget.NavigationView
                    android:id="@+id/navigation_view"
                    android:layout_width="wrap_content"
                    android:layout_height="match_parent"
                    android:layout_gravity="start"
                    android:background="#ffffff"
                    app:menu="@menu/activity_main_drawer" />
            </android.support.v4.widget.DrawerLayout>
        </android.support.design.widget.AppBarLayout>
</layout>
```

A.73 activity_music.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="com.example.fyp2_tsk.other.MusicActivity">

    <ImageView
        android:id="@+id/music_bg_imgv"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scaleType="centerCrop"
        android:src="@mipmap/touxiang1" />

    <RelativeLayout
        android:id="@+id/music_relativelayout"
        android:layout_width="match_parent"
        android:layout_height="50dp">

        <ImageView
            android:id="@+id/music_down_imgv"
            android:layout_width="50dp"
            android:layout_height="50dp"
            android:layout_alignParentLeft="true"
            android:layout_margin="5dp"
            android:src="@mipmap/ic_arrow_down" />

        <TextView
            android:id="@+id/music_title_tv"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toRightOf="@+id/music_down_imgv"
            android:text="周杰伦"
            android:textColor="#ffffff"
            android:textSize="13dp" />

        <TextView
            android:id="@+id/music_artist_tv"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
```

```
            android:layout_below="@+id/music_title_tv"
            android:layout_marginLeft="8dp"
            android:layout_toRightOf="@+id/music_down_imgv"
            android:text="告白气球"
            android:textColor="#ffffff"
            android:textSize="10dp" />

    </RelativeLayout>
    <RelativeLayout
        android:layout_width="350dp"
        android:layout_height="500dp"
        android:layout_above="@+id/music_linlayout"
        android:layout_below="@+id/music_relativelayout"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true">

        <ImageView
            android:id="@+id/music_disc_imagv"
            android:layout_width="match_parent"
            android:layout_height="300dp"
            android:layout_marginTop="100dp"
            android:src="@mipmap/play_page_disc" />

        <ImageView
            android:id="@+id/music_needle_imag"
            android:layout_width="150dp"
            android:layout_height="159dp"
            android:layout_marginLeft="132dp"
            android:src="@mipmap/play_page_needle" />
    </RelativeLayout>

    <LinearLayout
        android:id="@+id/music_linlayout"
        android:layout_width="match_parent"
        android:layout_height="90dp"
        android:layout_alignParentBottom="true"
        android:orientation="vertical">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="5"
            android:orientation="horizontal">
```

```
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <TextView
                android:id="@+id/music_current_tv"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
                android:layout_centerVertical="true"
                android:layout_marginLeft="10dp"
                android:text="00:00"
                android:textColor="#ffffff" />

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_centerVertical="true"
                android:layout_marginLeft="10dp"
                android:layout_marginRight="10dp"
                android:layout_toLeftOf="@+id/music_total_tv"
                android:layout_toRightOf="@+id/music_current_tv">

                <SeekBar
                    android:id="@+id/music_seekbar"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent" />

            </LinearLayout>

            <TextView
                android:id="@+id/music_total_tv"
                android:layout_width="50dp"
                android:layout_height="wrap_content"
                android:layout_marginRight="10dp"
                android:layout_alignParentRight="true"
                android:text="00:00"
                android:textColor="#ffffff" />

        </RelativeLayout>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
```

```
        android:layout_height="0dp"
        android:layout_weight="10">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <ImageView
                android:id="@+id/music_play_btn_loop_img"
                android:layout_width="60dp"
                android:layout_height="60dp"
                android:layout_centerVertical="true"
                android:layout_marginLeft="10dp"
                android:src="@mipmap/ic_play_btn_loop" />

            <ImageView
                android:id="@+id/music_prev_imgv"
                android:layout_width="60dp"
                android:layout_height="60dp"
                android:layout_centerVertical="true"
                android:layout_toRightOf="@+id/music_play_btn_loop_img"
                android:src="@drawable/seach_btn_pre" />

            <ImageView
                android:id="@+id/music_pause_imgv"
                android:layout_width="60dp"
                android:layout_height="60dp"
                android:layout_centerHorizontal="true"
                android:layout_centerVertical="true"
                android:src="@drawable/seach_btn_pause" />

            <ImageView
                android:id="@+id/music_next_imgv"
                android:layout_width="60dp"
                android:layout_height="60dp"
                android:layout_centerVertical="true"
                android:layout_marginLeft="25dp"
                android:layout_toRightOf="@id/music_pause_imgv"
                android:src="@drawable/seach_btn_next" />

            <ImageView
                android:id="@+id/driver_Mode"
                android:layout_width="40dp"
                android:layout_height="40dp"
```

```
                    android:layout_centerVertical="true"
                    android:layout_marginLeft="10dp"
                    android:layout_toRightOf="@id/music_next_imgv"
                    android:src="@drawable/ic_drive_mode_white_24dp" />
            </RelativeLayout>
        </LinearLayout>
    </LinearLayout>
</RelativeLayout>
```

A.74 activity_splash.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <android.support.constraint.ConstraintLayout
        android:id="@+id/splash_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/splash_m"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:padding="2dp"
            android:text="M"
            android:textColor="@color/white"
            android:textSize="26sp"
            android:visibility="invisible"
            app:layout_constraintBottom_toBottomOf="@+id/splash_u"
            app:layout_constraintRight_toLeftOf="@+id/splash_u"
            app:layout_constraintTop_toTopOf="@+id/splash_u"
            app:layout_constraintVertical_bias="0.0" />

        <TextView
            android:id="@+id/splash_i"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="2dp"
            android:text="i"
            android:textColor="@color/white"
            android:textSize="26sp"
            android:visibility="invisible"
```

```
        app:layout_constraintBottom_toTopOf="@+id/guideline20"
        app:layout_constraintHorizontal_bias="0.418"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent" />

    <TextView
        android:id="@+id/splash_c"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:text="c"
        android:textColor="@color/white"
        android:textSize="26sp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="@+id/splash_i"
        app:layout_constraintLeft_toRightOf="@+id/splash_i"
        app:layout_constraintTop_toTopOf="@+id/splash_i"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/splash_p"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
        android:padding="2dp"
        android:text="P"
        android:textColor="@color/white"
        android:textSize="26sp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="@+id/splash_c"
        app:layout_constraintLeft_toRightOf="@+id/splash_c"
        app:layout_constraintTop_toTopOf="@+id/splash_c"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/splash_l"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:text="l"
        android:textColor="@color/white"
        android:textSize="26sp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="@+id/splash_p"
```

```
        app:layout_constraintLeft_toRightOf="@+id/splash_p"
        app:layout_constraintTop_toTopOf="@+id/splash_p"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/splash_a"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:text="a"
        android:textColor="@color/white"
        android:textSize="26sp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="@+id/splash_l"
        app:layout_constraintLeft_toRightOf="@+id/splash_l"
        app:layout_constraintTop_toTopOf="@+id/splash_l"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/splash_y"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:text="y"
        android:textColor="@color/white"
        android:textSize="26sp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="@+id/splash_a"
        app:layout_constraintLeft_toRightOf="@+id/splash_a"
        app:layout_constraintTop_toTopOf="@+id/splash_a"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/splash_e"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:text="e"
        android:textColor="@color/white"
        android:textSize="26sp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="@+id/splash_y"
        app:layout_constraintLeft_toRightOf="@+id/splash_y"
        app:layout_constraintTop_toTopOf="@+id/splash_y"
```

```
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/splash_r"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:text="r"
        android:textColor="@color/white"
        android:textSize="26sp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="@+id/splash_e"
        app:layout_constraintLeft_toRightOf="@+id/splash_e"
        app:layout_constraintTop_toTopOf="@+id/splash_e"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/splash_s"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:text="s"
        android:textColor="@color/white"
        android:textSize="26sp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="@+id/splash_i"
        app:layout_constraintRight_toLeftOf="@+id/splash_i"
        app:layout_constraintTop_toTopOf="@+id/splash_i"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        android:id="@+id/splash_u"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="2dp"
        android:text="u"
        android:textColor="@color/white"
        android:textSize="26sp"
        android:visibility="invisible"
        app:layout_constraintBottom_toBottomOf="@+id/splash_s"
        app:layout_constraintRight_toLeftOf="@+id/splash_s"
        app:layout_constraintTop_toTopOf="@+id/splash_s"
        app:layout_constraintVertical_bias="0.0" />
```

```
        <TextView
            android:id="@+id/splash_name"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="16dp"
            android:layout_marginLeft="8dp"
            android:layout_marginRight="8dp"
            android:text="@string/app_name"
            android:textColor="@color/white_d_d_d"
            android:textSize="16sp"
            android:visibility="invisible"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent" />

        <ImageView
            android:id="@+id/splash_logo"
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:layout_marginBottom="0dp"
            android:src="@drawable/img_0"
            android:visibility="invisible"
            app:layout_constraintBottom_toTopOf="@+id/splash_i"
            app:layout_constraintHorizontal_bias="0.501"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent" />

        <android.support.constraint.Guideline
            android:id="@+id/guideline19"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            app:layout_constraintGuide_percent="0.5" />

        <android.support.constraint.Guideline
            android:id="@+id/guideline20"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            app:layout_constraintGuide_percent="0.61" />

    </android.support.constraint.ConstraintLayout>
</layout>
```

A.75 activity_time_sleep.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.v7.widget.Toolbar
            android:id="@+id/time_sleep_toolbar"
            android:layout_width="match_parent"
            android:layout_height="?actionBarSize" />
    </android.support.design.widget.AppBarLayout>

    <include layout="@layout/time_sleep_content" />

</android.support.design.widget.CoordinatorLayout>
```

A.76 dialog_common.xml

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:layout_margin="20dip"
    android:background="@drawable/commondialog_bg"
    android:orientation="vertical">
    <TextView
        android:id="@+id/tv_dialog_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginTop="20dp"
        android:textColor="#202020"
        android:textSize="18dp"
        android:visibility="gone" />

    <TextView
        android:id="@+id/tv_dialog_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="18dp"
        android:layout_marginTop="12dp"
        android:textColor="#202020"
        android:textSize="15dp" />

    <EditText
        android:id="@+id/et_dialog_input"
        android:layout_width="fill_parent"
        android:layout_height="32dp"
        android:layout_marginLeft="20dp"
        android:layout_marginRight="20dp"
        android:layout_marginTop="12dp"
        android:background="@mipmap/commondialog_et_bg"
        android:inputType="textPassword"
        android:maxLength="20"
        android:maxLines="1"
        android:paddingLeft="15dp"
        android:textColor="#000000"
```

```
        android:textColorHint="#ACB3BF"
        android:textSize="13dp"
        android:visibility="gone" />

    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="48dp"
        android:layout_marginTop="12dp">

        <TextView
            android:id="@+id/tv_dialog_confirm"
            android:layout_width="75dp"
            android:layout_height="36dp"
            android:layout_alignParentRight="true"
            android:layout_centerVertical="true"
            android:layout_marginRight="12dp"
            android:gravity="center"
            android:text="@string/dialog_btn_sure"
            android:textColor="#009688"
            android:textSize="14dp" />

        <TextView
            android:id="@+id/tv_dialog_cancle"
            android:layout_width="75dp"
            android:layout_height="36dp"
            android:layout_centerVertical="true"
            android:layout_marginRight="12dp"
            android:layout_toLeftOf="@id/tv_dialog_confirm"
            android:gravity="center"
            android:text="@string/dialog_btn_cancel"
            android:textColor="#009688"
            android:textSize="14dp" />

    </RelativeLayout>
</LinearLayout>
```

A.77 dialog_theme_color.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="#ffffff"
        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        android:orientation="vertical">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Theme Setting"
            android:textColor="@color/theme_color"
            android:layout_marginLeft="25dp"
            android:layout_marginTop="15dp"
            android:textSize="16dp"/>
        <View
            android:layout_width="match_parent"
            android:layout_height="2dp"
            android:layout_marginTop="15dp"
            android:background="#f1f2f3"/>
        <android.support.v7.widget.RecyclerView
            android:id="@+id/recycler_theme"
            android:layout_width="match_parent"
            android:layout_height="120dp"
            android:layout_gravity="center"
            android:layout_marginLeft="10dp"
            android:layout_marginTop="15dp"
            android:layout_marginRight="10dp"/>
        <TextView
            android:id="@+id/tv_sure"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Confirm"
            android:textSize="16dp"
            android:layout_gravity="right"
            android:layout_marginRight="25dp"
            android:layout_marginBottom="10dp"
            android:layout_marginTop="5dp"    android:textColor="@color/theme_color"/>
    </LinearLayout>
</layout>
```

A.78 fragment_all.xml

```
<FrameLayout 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="com.example.fyp2_tsk.other.AllFragment">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/bg_listview"
        android:orientation="vertical">

        <SearchView
            android:id="@+id/edSearch"
            android:layout_width="320dp"
            android:layout_height="wrap_content"
            android:hint="Search" />

        <android.support.constraint.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <LinearLayout
                android:id="@+id/linearLayout"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginEnd="32dp"
                android:orientation="horizontal"
                app:layout_constraintEnd_toEndOf="parent">

                <!-- TODO: Update blank fragment layout -->
                <ListView
                    android:id="@+id/logic_lv"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:background="@color/cutter_theme_white"
                    android:divider="@null"></ListView>
            </LinearLayout>
        </android.support.constraint.ConstraintLayout>
    </LinearLayout>
```

```
    <com.lwkandroid.widget.indexbar.IndexBar
        android:id="@+id/indexBar"
        android:layout_width="40dp"
        android:layout_height="match_parent"
        app:bg_color_normal="@android:color/transparent"
        app:bg_color_pressed="#10000000"
        app:text_color_normal="#3c3c3c"
        app:text_color_pressed="#000093"
        app:text_size_normal="14sp"
        android:layout_gravity="end"
        app:text_size_pressed="16sp" />

</FrameLayout>
```

A.79 fragment_cut.xml

```
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:rangeseekbar="http://schemas.android.com/apk/res-auto">

    <RelativeLayout
        android:id="@+id/rl_main"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:background="#ffffff">

        <!-- voice bar -->
        <RelativeLayout
            android:id="@+id/rl_player_voice"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:layout_marginTop="10dp"
            android:visibility="gone">

            <ImageView
                android:id="@+id/iv_player_min_voice"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
                android:layout_centerVertical="true"
                android:layout_marginLeft="10dip"
                android:background="@mipmap/volume_min" />

            <ImageView
                android:id="@+id/iv_player_max_voice"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:layout_centerVertical="true"
                android:layout_marginRight="10dp"
                android:background="@mipmap/volume_max" />

            <SeekBar
                android:id="@+id/voice_seekbar"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_centerVertical="true"
                android:layout_toLeftOf="@id/iv_player_max_voice"
```

```
            android:layout_toRightOf="@id/iv_player_min_voice" />

    </RelativeLayout>

    <!-- playerbar view -->
    <LinearLayout
        android:id="@+id/ll_play_toolbar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:gravity="center_vertical"
        android:orientation="horizontal"
        android:weightSum="4">

        <ImageButton
            android:id="@+id/btn_play"
            android:layout_width="80dp"
            android:layout_height="55dp"
            android:layout_weight="1"
            android:background="@drawable/selector_play_btn" />

        <ImageButton
            android:id="@+id/btn_cutter_sure"
            android:layout_width="114dp"
            android:layout_height="67dp"
            android:layout_weight="1"
            android:background="@drawable/selector_sure_btn" />
    </LinearLayout>

    <!-- playbar view -->
    <RelativeLayout
        android:id="@+id/rl_player_center"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_above="@id/ll_play_toolbar"
        android:layout_marginBottom="10dp">

        <ImageView
            android:id="@+id/topborder"
            android:layout_width="fill_parent"
            android:layout_height="1dp"
```

```
                android:layout_alignParentTop="true"
                android:background="@mipmap/border_bg" />

            <RelativeLayout
                android:id="@+id/rl_playerbar"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_below="@id/topborder">

                <com.zyl.customrangeseekbar.CustomRangeSeekBar
                    android:id="@+id/range_seekbar"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_centerVertical="true"
                    rangeseekbar:progressBarBg="@mipmap/seekbar_bg"
                    rangeseekbar:progressBarSelBg="@mipmap/seekbar_sel_bg"
                    rangeseekbar:thumbImage="@mipmap/btn_seekbar_normal"
                    rangeseekbar:progressTextSize="16dp"
                    rangeseekbar:progressTextFormat="timeFormat"
                    rangeseekbar:startMinPercent="0"
                    rangeseekbar:startMaxPercent="1"
                    />
            </RelativeLayout>

            <ImageView
                android:id="@+id/bottomborder"
                android:layout_width="fill_parent"
                android:layout_height="1dp"
                android:layout_below="@id/rl_playerbar"
                android:background="@mipmap/border_bg" />
        </RelativeLayout>

        <com.example.fyp2_tsk.common.ui.view.visualizer.VisualizerView
            android:id="@+id/visual_view"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:layout_above="@id/rl_player_center"/>
    </RelativeLayout>
</layout>
```

A.80 fragment_music.xml

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

    <LinearLayout
        android:id="@+id/main_top_linlayout"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:background="@color/colorPrimaryDark"
        android:orientation="vertical">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="50dp">

                <TextView
                    android:id="@+id/main_logic_tv"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_centerVertical="true"
                    android:text="All"
                    android:textColor="@color/white"
                    android:layout_marginLeft="60dp"
                    android:textSize="15dp" />

                <TextView
                    android:id="@+id/eng_logic_tv"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_centerVertical="true"
                    android:text="English"
                    android:layout_marginLeft="75dp"
                    android:layout_toRightOf="@+id/main_logic_tv"
                    android:textColor="@color/white_60P"
                    android:textSize="15dp" />

                <TextView
                    android:id="@+id/chinese_logic_tv"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_centerVertical="true"
                    android:text="Others"
```

```
                    android:layout_marginLeft="75dp"
                    android:textAlignment="center"
                    android:textColor="@color/white_60P"
                    android:layout_toRightOf="@+id/eng_logic_tv"
                    android:textSize="15dp" />
        </RelativeLayout>

        <android.support.v4.view.ViewPager
            android:id="@+id/main_vp"
            android:layout_width="match_parent"
            android:layout_height="match_parent"

android:background="@color/cutter_theme_white"></android.support.v4.view.ViewPager>
    </LinearLayout>

</layout>
```

A.81 item_musicfile.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:id="@+id/iv_icon"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:src="@mipmap/music_icon"
        android:layout_marginTop="5dp"
        android:layout_marginBottom="5dp"
        android:layout_marginLeft="10dp"/>

    <TextView
        android:id="@+id/tv_name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="14dp"
        android:maxLines="1"
        android:ellipsize="end"
        android:textColor="#000000"
        android:layout_marginLeft="5dp"
        android:layout_toRightOf="@id/iv_icon"
        android:layout_alignTop="@id/iv_icon"/>

    <TextView
        android:id="@+id/tv_size"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="10dp"
        android:maxLines="1"
        android:ellipsize="end"
        android:textColor="#808080"
        android:layout_alignLeft="@id/tv_name"
        android:layout_alignBottom="@id/iv_icon" />

</RelativeLayout>
```

A.82 item_theme_color.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android">
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_marginLeft="16dp">

        <de.hdodenhof.circleimageview.CircleImageView
            android:id="@+id/them_color"
            android:layout_width="48dp"
            android:layout_height="48dp"
            android:layout_gravity="center"
            android:layout_margin="4dp" />

        <ImageView
            android:id="@+id/choose"
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_gravity="center"
            android:src="@mipmap/ic_done_white_36dp"
            android:visibility="gone" />

    </FrameLayout>
</layout>
```

A.83 music_item.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="50dp">

    <View
        android:id="@+id/musicitem_playing_v"
        android:layout_width="3dp"
        android:layout_height="50dp"
        android:visibility="invisible"
        android:background="@color/colorAccent"
        />

    <ImageView
        android:id="@+id/musicitem_album_imgv"
        android:layout_marginLeft="3dp"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:layout_marginTop="5dp"
        android:layout_marginBottom="5dp"
        android:src="@mipmap/music_icon"/>

    <TextView
        android:id="@+id/musicitem_title_tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_toRightOf="@+id/musicitem_album_imgv"
        android:textSize="13dp"
        android:text="成都"/>
    <TextView
        android:id="@+id/musicitem_artist_tv"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="11dp"
        android:layout_toRightOf="@+id/musicitem_album_imgv"
        android:layout_below="@+id/musicitem_title_tv"
        android:layout_marginLeft="20dp"
        android:layout_gravity="bottom"
        android:text="赵雷"/>

    <View
```

```
        android:layout_width="match_parent"
        android:layout_height="1sp"
        android:layout_toRightOf="@+id/musicitem_album_imgv"
        android:background="#cccccc"
        android:layout_alignParentBottom="true"/>

</RelativeLayout>
```

A.84 time_sleep_content.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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"
    app:layout_behavior="@string/appbar_scrolling_view_behavior">

    <TextView
        android:id="@+id/time_sleep_status"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="0dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="0dp"
        android:gravity="start|center_vertical"
        android:padding="16dp"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/time_sleep_status_switch"
        app:layout_constraintTop_toTopOf="parent" />

    <Switch
        android:id="@+id/time_sleep_status_switch"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="0dp"
        android:layout_marginRight="0dp"
        android:layout_marginTop="0dp"
        android:paddingRight="16dp"
        app:layout_constraintBottom_toBottomOf="@+id/time_sleep_status"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="@+id/time_sleep_status" />

    <View
        android:id="@+id/time_sleep_line"
        android:layout_width="0dp"
        android:layout_height="1px"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="0dp"
        android:background="#41000000"
```

```
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/time_sleep_status" />

    <TextView
        android:id="@+id/time_sleep_10m"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:clickable="true"
        android:padding="8dp"
        android:tag="10"
        android:textSize="16sp"
        app:layout_constraintBottom_toTopOf="@+id/guideline14"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/guideline11"
        app:layout_constraintTop_toTopOf="@+id/guideline13" />

    <TextView
        android:id="@+id/time_sleep_20m"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:clickable="true"
        android:padding="8dp"
        android:tag="20"
        android:textSize="16sp"
        app:layout_constraintBottom_toTopOf="@+id/guideline14"
        app:layout_constraintLeft_toLeftOf="@+id/guideline11"
        app:layout_constraintRight_toLeftOf="@+id/guideline9"
        app:layout_constraintTop_toTopOf="@+id/guideline13" />

    <TextView
        android:id="@+id/time_sleep_30m"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
```

```
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:clickable="true"
        android:padding="8dp"
        android:tag="30"
        android:textSize="16sp"
        app:layout_constraintBottom_toTopOf="@+id/guideline14"
        app:layout_constraintLeft_toLeftOf="@+id/guideline9"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="@+id/guideline13" />

    <TextView
        android:id="@+id/time_sleep_45m"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:clickable="true"
        android:padding="8dp"
        android:tag="45"
        android:textSize="16sp"
        app:layout_constraintBottom_toTopOf="@+id/guideline12"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/guideline11"
        app:layout_constraintTop_toTopOf="@+id/guideline14" />

    <TextView
        android:id="@+id/time_sleep_60m"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="8dp"
        android:clickable="true"
        android:padding="8dp"
        android:tag="60"
        android:textSize="16sp"
        app:layout_constraintBottom_toTopOf="@+id/guideline12"
        app:layout_constraintLeft_toLeftOf="@+id/guideline11"
        app:layout_constraintRight_toLeftOf="@+id/guideline9"
        app:layout_constraintTop_toTopOf="@+id/guideline14" />
```

```
<Button
    android:id="@+id/time_sleep_custom"
    style="@style/Base.Widget.AppCompat.Button.Colored"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginTop="8dp"
    android:padding="15dp"
    android:text="@string/custom"
    android:textSize="16sp"
    app:layout_constraintBottom_toTopOf="@+id/guideline15"
    app:layout_constraintHorizontal_bias="0.466"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toLeftOf="@+id/guideline11"
    app:layout_constraintTop_toTopOf="@+id/guideline12" />

<TextView
    android:id="@+id/time_sleep_custom_show"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="8dp"
    android:layout_marginLeft="56dp"
    android:layout_marginRight="8dp"
    android:layout_marginTop="8dp"
    android:background="#22000000"
    android:clickable="false"
    android:gravity="center_vertical|start"
    android:padding="3dp"
    app:layout_constraintBottom_toTopOf="@+id/guideline15"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="@+id/guideline12" />

<include
    android:id="@+id/time_sleep_pickers"
    layout="@layout/time_sleep_content_pickers"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginBottom="0dp"
    android:layout_marginLeft="0dp"
    android:layout_marginRight="0dp"
```

```
        android:layout_marginTop="0dp"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="@+id/guideline15" />

    <android.support.constraint.Guideline
        android:id="@+id/guideline9"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.66" />


    <android.support.constraint.Guideline
        android:id="@+id/guideline11"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.33" />

    <android.support.constraint.Guideline
        android:id="@+id/guideline12"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.42" />

    <android.support.constraint.Guideline
        android:id="@+id/guideline15"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.56" />

    <android.support.constraint.Guideline
        android:id="@+id/guideline14"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.28" />

    <android.support.constraint.Guideline
```

```
        android:id="@+id/guideline13"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.14" />

</android.support.constraint.ConstraintLayout>
```

A.85 time_sleep_content_picker.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.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"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:layout_editor_absoluteX="0dp"
    tools:layout_editor_absoluteY="81dp">

    <NumberPicker
        android:id="@+id/time_sleep_custom_num_left"
        style="30sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginRight="16dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/time_sleep_custom_num_left_t"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/time_sleep_custom_num_left_t"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginRight="16dp"
        android:layout_marginTop="8dp"
        android:text="Hour"
        android:textSize="16sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/guideline18"
        app:layout_constraintTop_toTopOf="parent" />

    <NumberPicker
        android:id="@+id/time_sleep_custom_num_right"
        style="30sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
```

```
        android:layout_marginLeft="16dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="@+id/guideline18"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/time_sleep_custom_num_right_t"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:text="Minutes"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/time_sleep_custom_num_right"
        app:layout_constraintTop_toTopOf="parent" />

    <android.support.constraint.Guideline
        android:id="@+id/guideline18"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        app:layout_constraintGuide_percent="0.5" />

</android.support.constraint.ConstraintLayout>
```

POSTER

UNIVERSITI TUNKU ABDUL RAHMAN

Faculty of Information and Communication Technology

Name: Tan Siang Kian

Student ID: 16ACB05288

Programme: Information System Engineering

Supervisor: Dr. Ooi Chek Yee

MP3 Music Player Application Development Using Android

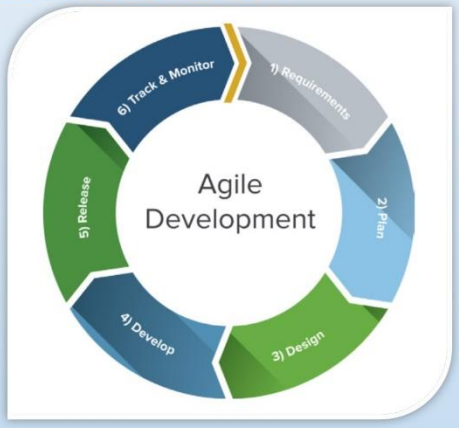
INTRODUCTION

An integrate the advantages of existing music players on the market by eliminating function that not practical or low cost-effective and then add some useful features such as driving mode and audio trim.

OBJECTIVE

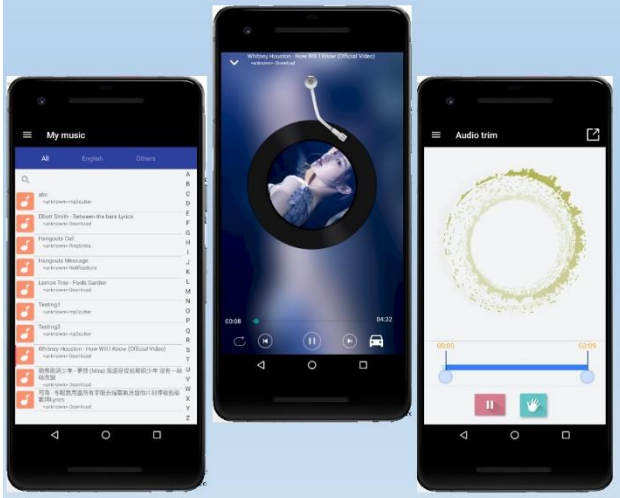
- Make it with a simple feature and run smoothly
- Support gesture control
- Support quick search

METHODOLOGY



The agile development cycle contains 6 phrase which is requirement analysis, planning, design, implementation or development, testing, and deployment.

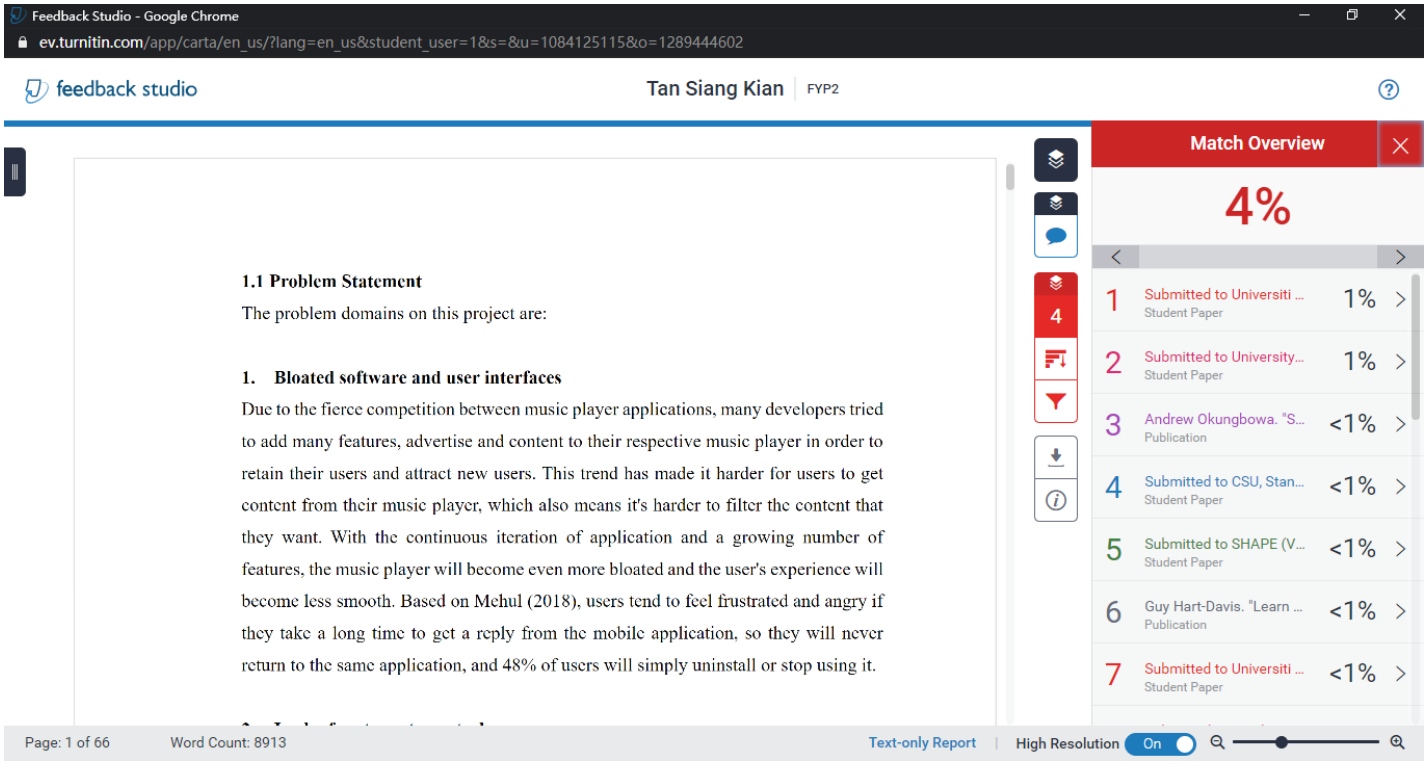
RESULTS

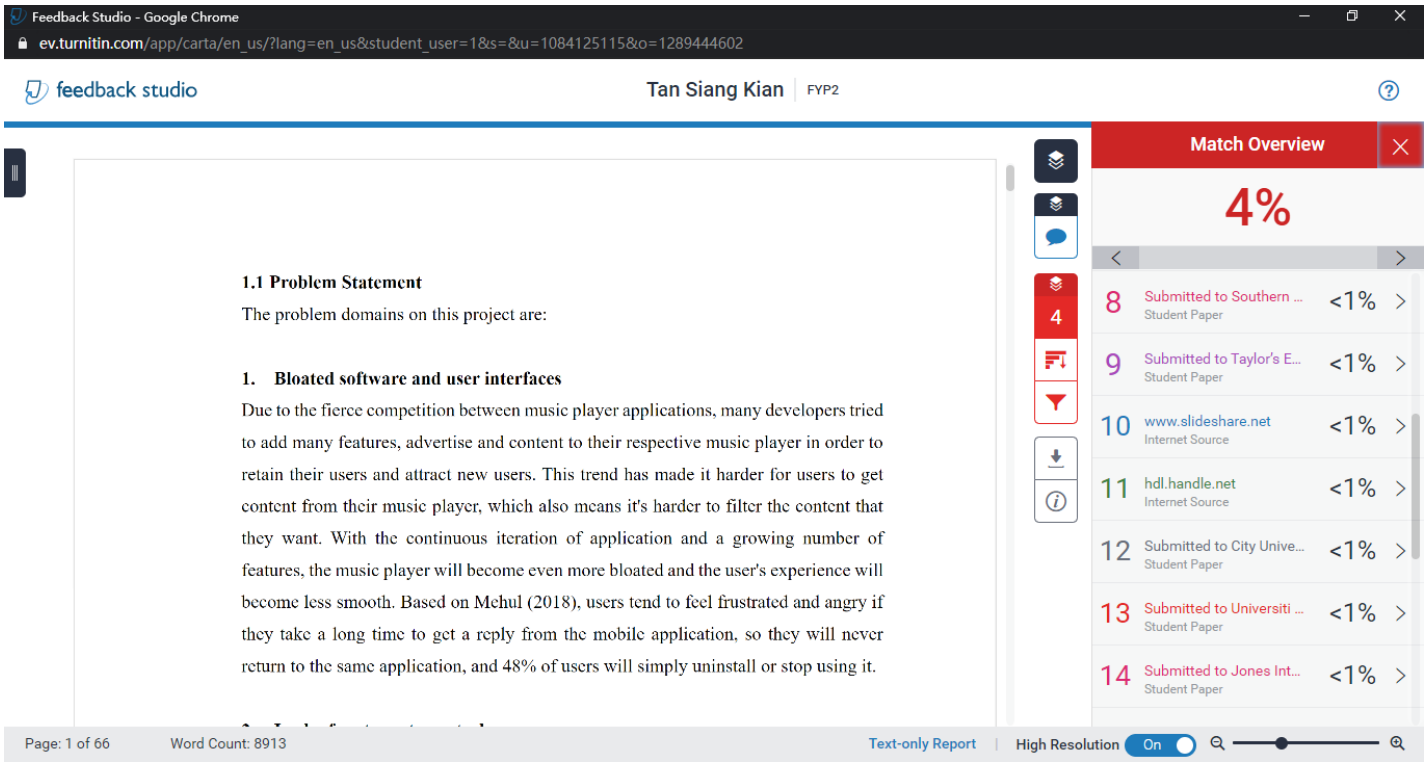


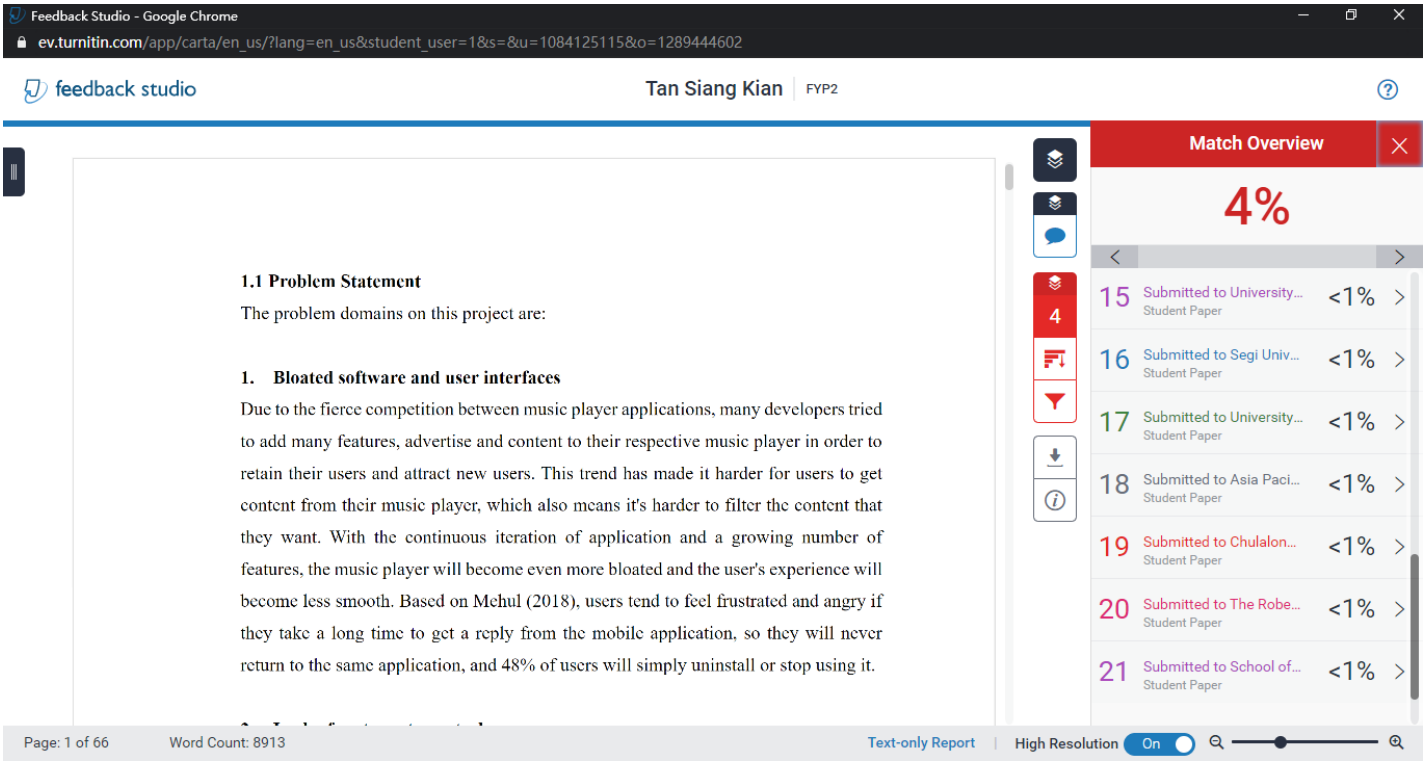
CONCLUSION

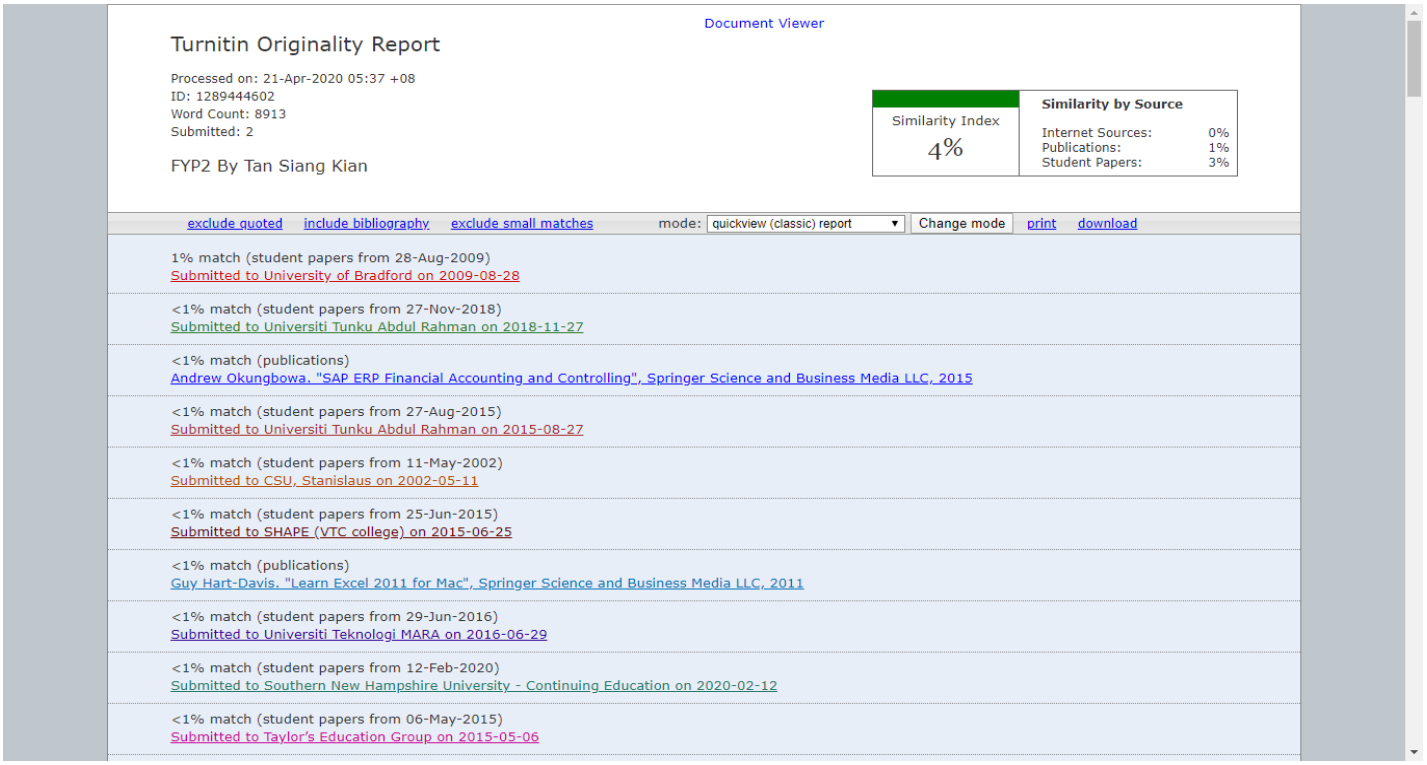
This application is a lightweight application and runs smoothly without consuming too many system resources. In addition, the application completely got rid of only using icon buttons for media control and allow the user to utilize quick search to find song in playlist faster.

PLAGIARISM CHECK RESULT






Feedback Studio - Google Chrome
ev.turnitin.com/app/carta/en_us/?lang=en_us&student_user=1&s=&u=1084125115&o=1289444602
feedback studio
Tan Siang Kian
FYP2
1.1 Problem Statement
The problem domains on this project are:
1. Bloated software and user interfaces
Due to the fierce competition between music player applications, many developers tried to add many features, advertise and content to their respective music player in order to retain their users and attract new users. This trend has made it harder for users to get content from their music player, which also means it's harder to filter the content that they want. With the continuous iteration of application and a growing number of features, the music player will become even more bloated and the user's experience will become less smooth. Based on Mehul (2018), users tend to feel frustrated and angry if they take a long time to get a reply from the mobile application, so they will never return to the same application, and 48% of users will simply uninstall or stop using it.
Match Overview
4%
15 Submitted to University... <1%
Student Paper
16 Submitted to Segi Univ... <1%
Student Paper
17 Submitted to University... <1%
Student Paper
18 Submitted to Asia Paci... <1%
Student Paper
19 Submitted to Chulalon... <1%
Student Paper
20 Submitted to The Robe... <1%
Student Paper
21 Submitted to School of... <1%
Student Paper
Page: 1 of 66
Word Count: 8913
Text-only Report
High Resolution
On


Document Viewer
Turnitin Originality Report
Processed on: 21-Apr-2020 05:37 +08
ID: 1289444602
Word Count: 8913
Submitted: 2
FYP2 By Tan Siang Kian
Similarity Index
4%
Similarity by Source
Internet Sources: 0%
Publications: 1%
Student Papers: 3%
exclude quoted
include bibliography
exclude small matches
mode: quickview (classic) report
Change mode
print
download
1% match (student papers from 28-Aug-2009)
Submitted to University of Bradford on 2009-08-28
<1% match (student papers from 27-Nov-2018)
Submitted to Universiti Tunku Abdul Rahman on 2018-11-27
<1% match (publications)
Andrew Okungbowa, "SAP ERP Financial Accounting and Controlling", Springer Science and Business Media LLC, 2015
<1% match (student papers from 27-Aug-2015)
Submitted to Universiti Tunku Abdul Rahman on 2015-08-27
<1% match (student papers from 11-May-2002)
Submitted to CSU, Stanislaus on 2002-05-11
<1% match (student papers from 25-Jun-2015)
Submitted to SHAPE (VTC college) on 2015-06-25
<1% match (publications)
Guy Hart-Davis, "Learn Excel 2011 for Mac", Springer Science and Business Media LLC, 2011
<1% match (student papers from 29-Jun-2016)
Submitted to Universiti Teknologi MARA on 2016-06-29
<1% match (student papers from 12-Feb-2020)
Submitted to Southern New Hampshire University - Continuing Education on 2020-02-12
<1% match (student papers from 06-May-2015)
Submitted to Taylor's Education Group on 2015-05-06

FYP2

ORIGINALITY REPORT

4%	0%	1%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Universiti Tunku Abdul Rahman Student Paper	1%
2	Submitted to University of Bradford Student Paper	1%
3	Andrew Okungbowa. "SAP ERP Financial Accounting and Controlling", Springer Science and Business Media LLC, 2015 Publication	<1%
4	Submitted to CSU, Stanislaus Student Paper	<1%
5	Submitted to SHAPE (VTC college) Student Paper	<1%
6	Guy Hart-Davis. "Learn Excel 2011 for Mac", Springer Science and Business Media LLC, 2011 Publication	<1%
7	Submitted to Universiti Teknologi MARA Student Paper	<1%
	Submitted to Southern New Hampshire	

8	University - Continuing Education Student Paper	<1%
9	Submitted to Taylor's Education Group Student Paper	<1%
10	www.slideshare.net Internet Source	<1%
11	hdl.handle.net Internet Source	<1%
12	Submitted to City University of Hong Kong Student Paper	<1%
13	Submitted to Universiti Tenaga Nasional Student Paper	<1%
14	Submitted to Jones International University Student Paper	<1%
15	Submitted to University of Bedfordshire Student Paper	<1%
16	Submitted to Segi University College Student Paper	<1%
17	Submitted to University of Hertfordshire Student Paper	<1%
18	Submitted to Asia Pacific Instutute of Information Technology Student Paper	<1%
	Submitted to Chulalongkorn University	

19	Student Paper	<1%
20	Submitted to The Robert Gordon University Student Paper	<1%
21	Submitted to School of Accounting & Management Student Paper	<1%

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	On		

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 SIANG KIAN
ID Number(s)	16ACB05288
Programme / Course	BACHELOR OF INFORMATION SYSTEMS (HONS) INFORMATION SYSTEM ENGINEERING
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: 4 % **Similarity by source** Internet Sources: 0 % Publications: 1 % Student Papers: 3 %	
Number of individual sources listed of more than 3% similarity: 0	

Parameters of originality required and limits approved by UTAR are as Follows:

(i) Overall similarity index is 20% and below, and

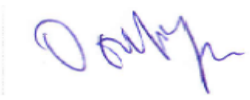
(ii) Matching of individual sources listed must be less than 3% each, and

(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.



Signature of Supervisor

Name: Ooi Chek Yee

Date: 20/04/2020

Signature of Co-Supervisor

Name: _____

Date: _____

UNIVERSITI TUNKU ABDUL RAHMAN

FACULTY OF INFORMATION & COMMUNICATION TECHNOLOGY (KAMPAR CAMPUS)

CHECKLIST FOR FYP2 THESIS SUBMISSION

Student Id	16ACB05288
Student Name	TAN SIANG KIAN
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 Cover
√	Signed Report Status Declaration Form
√	Title Page
√	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)
√	Poster
√	Signed Turnitin Report (Plagiarism Check Result – Form Number: FM-IAD-005)

*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: 20/04/2020	Supervisor verification. Report with incorrect format can get 5 mark (1 grade) reduction. _____ (Signature of Supervisor) Date: 20/04/2020

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan, 2020	**Study week no.:** 3
Student Name & ID: Tan Siang Kian & 16ACB05288	
Supervisor: Dr. Ooi Chek Yee	
Project Title: MP3 Music Player Application Development Using Android	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Review the FYP1 report and improve some chapters that need improvement.

2. WORK TO BE DONE

- Design and attempt to complete the music playing page module.

3. PROBLEMS ENCOUNTERED

- Not familiar with how to make a black vinyl disc into a rotating animation in the music playing interface

4. SELF EVALUATION OF THE PROGRESS

- Speed up the work progress, so the vinyl disc can rotate while the song is playing

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan, 2020	**Study week no.:** 6
Student Name & ID: Tan Siang Kian & 16ACB05288	
Supervisor: Dr. Ooi Chek Yee	
Project Title: MP3 Music Player Application Development Using Android	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Chapter 1 and Chapter 2 done
- Music playing page module

2. WORK TO BE DONE

- Gesture media control and shaking phone media control
- Driver mode
- Categorize song playlists into "All", "English", and "Others" categories

3. PROBLEMS ENCOUNTERED

- Shaking the phone's media control is not sensitive enough, and it takes a lot of effort to switch songs

4. SELF EVALUATION OF THE PROGRESS

- Speed up coding part in the development progress

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan, 2020	**Study week no.:** 9
Student Name & ID: Tan Siang Kian & 16ACB05288	
Supervisor: Dr. Ooi Chek Yee	
Project Title: MP3 Music Player Application Development Using Android	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Chapter 1 and 2 done
- Music playing page and driver mode
- Gesture media control and shaking phone media control
- Categorize song playlist

2. WORK TO BE DONE

- Sleep timer
- Audio trim
- Night mode

3. PROBLEMS ENCOUNTERED

- Only change to a dark background color in a single layout interface when night mode is enabled

4. SELF EVALUATION OF THE PROGRESS

- Catch up in writing report part

Supervisor's signature

Student's signature

FINAL YEAR PROJECT WEEKLY REPORT

(Project II)

Trimester, Year: Jan, 2020	**Study week no.:** 11
Student Name & ID: Tan Siang Kian & 16ACB05288	
Supervisor: Dr. Ooi Chek Yee	
Project Title: MP3 Music Player Application Development Using Android	

1. WORK DONE

[Please write the details of the work done in the last fortnight.]

- Chapter 1, 2, 3, 4, and 5
- The main module is totally completed

2. WORK TO BE DONE

- Chapter 6 – Conclusion
- Documentation
- Testing and Debugging on Final system

3. PROBLEMS ENCOUNTERED

- A small bug occurred in the single module of the test application

4. SELF EVALUATION OF THE PROGRESS

- Summarize the report and prepare the slide in order to presentation

Supervisor's signature

Student's signature