A

PROJECT REPORT

ON

"MUSIC PLAYER APP - SING WITH ME"

Submitted in the partial fulfillment of the requirement for the Award of degree

BACHELOR OF COMPUTER APPLICATION

FROM

H.N.B.G.U SRINAGAR

SESSION : 2020-21

PROJECT GUIDE : **Prof. SHEKHAR SINHA**
Asst Professor (DEPT. OF IT)

SUBMITTED BY :
BHARAT THAPA (LEADER)

BHUWAN CHANDRA PANT

BCA [BATCH: 2018-21]

DEPARTMENT OF INFORMATION TECHNOLOGY

INSTITUTE OF TECHNOLOGY & MANAGEMENT

DEHRADUN

CANDIDATE'S DECLARATION

This is to certify that work, which is being presented in the project entitled **"MUSIC PLAYER APP - SING WITH ME"** submitted by undersigned students of BCA 6th sem in partial fulfillment for award of degree of **Bachelor in Computer Application** is a record of their own work carried out by them under the guidance and supervision of **Mr. SHEKHAR SINHA, Asst Prof, Department of Information Technology.**

This work has not been submitted elsewhere for the award of any other degree.

Name & Signature of students

- ❖ BHARAT THAPA (Roll No: 18224512030)
- ❖ BHUWAN CHANDRA PANT (Roll No: 18224512032)

Date:

Place: ITM, Dehradun

INSTITUTE OF TECHNOLOGY & MANAGEMENT

DEHRADUN

CERTIFICATE

This is to certify that the work, which is being presented in the project entitled **"MUSIC PLAYER APP - SING WITH ME"** submitted by **Bharat Thapa, Bhuwan Chandra Pant,** students of final year of BCA in partial fulfillment for award of degree of **Bachelor of Computer Application** is a record of student's work carried out by them under our guidance and supervision.

This work has not been submitted elsewhere for award of any other degree.

Date

Mudit Mittal (HOD)

ACKNOWLEDGEMENT

We the students of BCA [Batch: 2018-21] in the Institute of Technology & Management, Dehradun feel glad and believe that we are very lucky to get the opportunity to develop an in house project. It gives us great pleasure to express our sincere gratitude to Mr. MUDIT MITTAL, the HOD-IT, who helped us in the progress of our project work.

We express our heartfelt gratitude to all the faculties of IT Deptt in ITM, Dehradun for providing all the facilities to complete and make our project very easy and successful. Our deep knowledge of gratitude is extended to Mr. Shekhar Sinha (Project Guide).

DATE:-

PLACE: - ITM, Dehradun

BHARAT THAPA

BHUWAN CHANDRA PANT

INDEX

S No **Topics** **Page No**

Title of the Project

“SING WITH ME” under the category of “MUSIC PLAYER APP” an app based project.

Introduction of the Project

- **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, advertisements 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

applications and a growing number of features, the music player will become even more bloated and the user's experience will become less smooth. 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 sorting**

   When users continuously 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 songs by continuously swiping 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.

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 functions that are not practical or low cost-effective.

Also, it will be kept improved based on user feedback. On the other hand, the existing music players pay less attention to all this. Therefore, the music player will solve the limitation by adding required features 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.

Background Information and Motivation

In modern society, people live a fast-paced life, and pressure is constantly present in their 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 are 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 kinds of huge and many features are meaningless. Therefore, this project

is designed to dedicate to MP3 music players based on the Android mobile phone platform to optimize performance and simplify to meet user needs.

Objectives

The objective of this thesis is to propose development of android that: Make it with a simple feature and run smoothly. Using this mp3 music player will make users feel 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. A user can skip next or previous songs by simply clicking on the skip left and skip right icon.

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. In addition, the ability to enhance the interaction between users and media control is that the application can skip songs by just clicking the next button on the phone under the lock screen status of the phone.

Problem Encountered

The main problem encountered in this project is the structure of code which the structure did not build well at first and led to keep modification during its development. Since the layout interface of each activity is not built as uniformly managed and updated as in the beginning, only one layout is updated and the layout of the other remains unchanged. Fortunately, these problems were solved in the final development stage, but the solution is informal and not efficient, as it is achieved by using a lot of duplicate coding to solve.

Project Achievement

Firstly, the proposed music player had achieved its first objective, which is to make the music player becomes 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 adds some useful features.

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 touch controls. Using gestures, the user doesn't have to pay full attention to the phone, but simply click right or

left 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 clicking on left or right button on the screen of 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 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 scroll down to the letter l and the result appears. This is because the name of the song contains the characters entered in the alphabetical order.

Use Case Description

1. Actor: User

   Description: User able listen to music on this application

   Trigger:

   A. User select any song from the playlist

   B. User click the play button to start playing the music

   Precondition:

A. The user should have the songs downloaded on their mobile device.

Normal flow of events:

A. User select a song from the playlist

B. The song playing until the end

Alternate / Exceptional flows:

A. The application will display a message "No local music" if the playlist does not have a song.

2. Actor: User

   Description: The progress bar shows the progress of the song

   Trigger:

   A. The progress bar becomes active when the user starts playing the song

   Precondition:

   A. The song must be in playing status

   Normal flow of events:

   A. Song is playing

   C. The progress bar will be finished if the song is finished playing

   Alternate / Exceptional flows:

   A. The progress bar will reset if user play other song

B. If the user drags the progress bar to left or right, the song's progress will change

C. The progress bar will reset if finished playing.

System Design

Once the user starts the application, the first screen will be the home page, reads songs from the local device and generates a playlist.

The user will now choose one of the music from the playlist and start playing that. As the song begins to play, the piano image at the top of the screen will be seen and the button on bottom center will be updated to the 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.

Software Requirement Specification

The Software Requirements Specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional and behavioral description, an indication of performance requirements and design

constraints, appropriate validation criteria, and other data pertinent to requirements

Introduction of developing environment of Android

The applications of Android need to run based on the Android environment. The following is the configuration requirement and installation steps of the Android development environment.

The required software of the developing environment :

Operation system: Windows XP Linux Windows 7

Software：Android SDK(Software Development Kit)、ADT(Android Development Tool) IDE environment：Eclipse IDE + ADT Eclipse3 or higher

JDK：Java Runtime Environment virtual machine、Java Development Kit(JDK) Installation steps of the developing environment

Step 1: install the Java virtual machine JDK version - 6

Step 2: install Eclipse3-5 tools; download address:

http://www-eclipse-org/downloads/

Step 3: install the Android SDK: first download the Android SDK

Download address: http://developer-android-com/sdk/index-html

Step 4: Install Android ADT plug-in, run Eclipse and select help - > install new software and select add.

Input SDK tools path in the SDK location: D: \ android \ software \ android SDK – Windows and click OK. The Android environment is set up successfully.

The design principle of android application.

Twice the result with half the effort will get an overall study of the principles done before the design and follow them in the operation.

The principle of software design mainly includes the following points:

(1) Reliability

The reliability of the software design must be determined. The reliability of the software system refers to the ability to avoid fault occurred in the process of system running, as well as the ability to remedy troubles once the fault occurs.

(2) Reusability

Look for commonness of similar codes, and come up with new methods abstractly and reasonably. Pay attention to the generic design.

(3) Understandability

The understandability of software not only require clear and readable document, but the simplified structure of software itself, which requires the designer to possess keen insight and creativity, and know well about the design objects.

(4) Simple program

To keep the program simple and clear, good programmers can use simple programs to solve complex problems.

(5) Testability

Testability means that the created system has a proper data collection to conduct a comprehensive test of the entire system.

(6) The Open-Closed Principle

Module is extensible but cannot be modified. That is to say, extension is open to the existing code in order to adapt to the new requirements. While modify is closed to the categories. Once the design is completed, the categories cannot be modified.

Function and structure design of Android system.

This system adopts the modularized program design, and system function is correspondingly divided into function modules, the main modules include:

(1)UI function module design of mobile terminal: the index screen, play screen, music adding page, file management page are realized.

(2) Backstage function module design of mobile terminal: the specific function, music file data storage function and other functions are implemented.

The feasibility analysis

This section verified that it is feasible to add music player on the Android system from the aspects of economic, technical and social feasibility.

1. Economic feasibility

To design an Android mobile phone music player as long as a computer has Android development and the application development of Android is free. In addition, mobile phone music players are basic needs for the public. The information that which functions are necessary form all the

consumers , which functions are needed for some people, and which features are seldom to use is easy to understand. And a lot of research is eliminated, thus saving the spending. Therefore, the whole process of development doesn't need to spend any money that is economically feasible.

2. Technical feasibility

To design a music player which meets the basic requirements, a deep understand of JAVA language, the Eclipse development tools, SQLite databases, the Android system architecture, application of framework and other technical knowledge are needed.(framework is the core of the application, and rules that all the programmers participating in the development must abide by). Based on the related technology information and resources for Android on the market, and equipped with technical personnel of technology and the spirit of willing to learn, the technology is feasible.

3. Social feasibility

With the rapid development of the mobile phone market, all kinds of audio and video resources are widely circulated on the Internet. These resources seem ordinary, but have gradually become an indispensable part of people's lives, which resulted in the development of all kinds of

mobile phone players. But a lot of players are devoted to fancy appearance, strong function causing a lot of wasted resources to the user's mobile phone and bringing a lot of inconvenience to the user as multitasking operation is needed. Some functions are useless to ordinary people. Powerful player is a good thing, but a lot of functions are actually useless for most users. Aimed at these problems, developing a multiplied audio player which owns the features of simplified functions, common play function, meeting the needs of most users, less required memory and high quality of playing music, maximizes the optimization in performance.

Design Specifications

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 requirements 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.

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

A. 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.

B. 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.

C. 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.

D. 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.

E. 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.

F. 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

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.

Tool to use

1. Software Requirement

   A. Android Studio

   B. Java SE 8

   C. Visual Paradigm

2. Hardware Requirement

   A. 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

   B. Smartphone device

- Processor make: Qualcomm Snapdragon 652 (MSM8976)

- RAM: 4.0 GB

- Phone Storage: 64 GB

- Operating System: Android version 5.1.1 (Lollipop)

Functional Requirements:

1. Able to import the MP3 format files on the device into the music player
2. Able to play audio files smoothly
3. Able to listen to the song.
4. Songs in playlists can be quickly searched and filtered through the alphabet.

Non-functional Requirements:

1. Simplify user interface
2. Optimize the design to display the information in a better way.

System Performance Definition

To achieve targeted system performance for this project, there are criteria must be

achieved as the list shown below:

1. Hardware/Software Variation

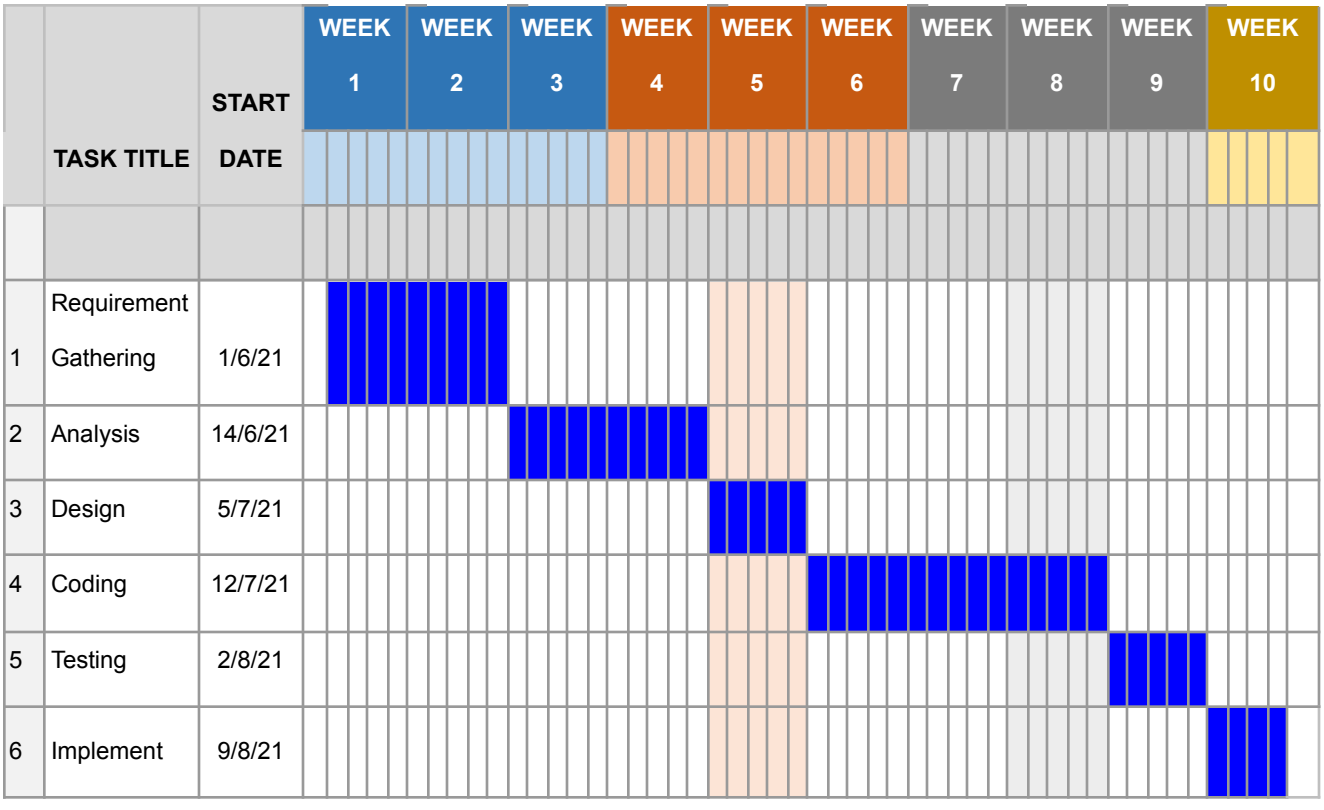
The MP3 music player application needs to ensure the application can work and install properly and smoothly on different devices. For example, the hardware and operating system of each phone will be different, just like Oppo and Huawei phones ColorOS and EMUI based on android respectively, so need to 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.

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.

Project Scheduling

An elementary Gantt chart or Timeline chart for the development plan is given below:



	TASK TITLE	START DATE
1	Requirement Gathering	1/6/21
2	Analysis	14/6/21
3	Design	5/7/21
4	Coding	12/7/21
5	Testing	2/8/21
6	Implement	9/8/21

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 executes 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 to perform media control through the icon buttons.

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: Play any song and click the Home button to make the app run in the background

Expected Output: Songs still playing in the background

Actual Output: Pass

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: Click the pause button

Expected Output: Stop play the song, the pause button is updated to the

play button

Actual Output: Pass

Input: Click the play button

Expected Output: Start play the song and update play the button to pause button

Actual Output: Pass

Input: Click the next button to

switch to the next song

Expected Output: Switch to the next song, the song name, album name, and artist name are also successfully updated to the next song's information

Actual Output: Pass

Input: Click the previous button to return to the previous song

Expected Output: Switch to the previous song, the song name, album name, and artist name are also successfully updated to the previous song's information

Actual Output: Pass

CODING OF SOME IMPORTANT PAGES

1. **MainActivity.java**

```
package com.example.singwithme;


import

androidx.appcompat.app.AppCompatActivity;
```

```
import android.Manifest;
import android.content.Intent;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

import com.karumi.dexter.Dexter;
import com.karumi.dexter.PermissionToken;
import
com.karumi.dexter.listener.PermissionDeniedRes
ponse;
import
com.karumi.dexter.listener.PermissionGrantedRe
sponse;
import
com.karumi.dexter.listener.PermissionRequest;
```

```
import
com.karumi.dexter.listener.single.PermissionLi
stener;

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

public class MainActivity extends
AppCompatActivity {

   @Override
         protected void onCreate(Bundle
savedInstanceState) {
       ListView listView;

       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_main);
       listView= findViewById(R.id.listView);
       Dexter.withContext(this)
```

```
.withPermission(Manifest.permission.READ_EXTER
NAL_STORAGE)
                                    .withListener(new
PermissionListener() {
                        @Override
                                                public void
onPermissionGranted(PermissionGrantedResponse
permissionGrantedResponse) {


//
Toast.makeText(MainActivity.this,     "Runtime
permission                                  given",
Toast.LENGTH_SHORT).show();
                                        ArrayList<File>
mySongs=
fetchSongs(Environment.getExternalStorageDirec
tory());
                                    String[] items= new
String[mySongs.size()];
```

```
                                    for (int i=0;i<
mySongs.size();i++){

items[i]=mySongs.get(i).getName().replace(".mp
3","");

                                }
                                    ArrayAdapter<String>
adapter=                                        new
ArrayAdapter<String>(MainActivity.this,
android.R.layout.simple_list_item_1,items);

listView.setAdapter(adapter);

listView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {
                                    @Override
                                        public void
onItemClick(AdapterView<?> parent, View view,
int position, long id) {
```

```
                                        Intent intent=
new Intent(MainActivity.this,PlaySong.class);
                                                String
currentSong=
listView.getItemAtPosition(position).toString(
);


intent.putExtra("songList", mySongs);


intent.putExtra("currentSong", currentSong);


intent.putExtra("position", position);


startActivity(intent);
                                    }
                                });


                            }


                            @Override
```

```
                                        public void
onPermissionDenied(PermissionDeniedResponse
permissionDeniedResponse) {

                    }

                    @Override
                                        public void
onPermissionRationaleShouldBeShown(PermissionR
equest   permissionRequest,   PermissionToken
permissionToken) {

permissionToken.continuePermissionRequest();
                    }
                })
                .check();
    }

        public   ArrayList<File>fetchSongs(File
file){
        ArrayList arrayList= new ArrayList();
```

```
        File[] songs= file.listFiles();
        if(songs!=null){
            for (File myFile: songs){
                    if  (!myFile.isHidden()  &&
myFile.isDirectory()){

arrayList.addAll(fetchSongs(myFile));
                }
                else{
                                                    if
(myFile.getName().endsWith(".mp3")              &&
!myFile.getName().startsWith(".")){
                            arrayList.add(myFile);

                    }
                }
            }
        }
        return arrayList;
    }
}
```

2. activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintL
ayout
xmlns:android="http://schemas.android.com/apk
/res/android"

xmlns:app="http://schemas.android.com/apk/res
-auto"

xmlns:tools="http://schemas.android.com/tools
"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/listView"
        android:layout_width="0dp"
        android:layout_height="0dp"
```

```
app:layout_constraintBottom_toBottomOf="paren
t"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.Constraint
Layout>
```

3. **PlaySong.java**

```
package com.example.singwithme;

import
androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.media.MediaPlayer;
import android.net.Uri;
```

```
import android.os.Bundle;

import android.view.View;

import android.widget.ImageView;

import android.widget.SeekBar;

import android.widget.TextView;


import java.io.File;

import java.util.ArrayList;


public class PlaySong extends
AppCompatActivity {
    @Override
    protected void onDestroy() {
        super.onDestroy();
        mediaPlayer.stop();
        mediaPlayer.release();
        updateSeek.interrupt();
    }


    TextView textView;
    ImageView play, previous,next;
```

```
    ArrayList<File>songs;
    MediaPlayer mediaPlayer;
    String textContent;
    int position;
    SeekBar seekBar;
    Thread updateSeek;

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

setContentView(R.layout.activity_play_song);
        textView= findViewById(R.id.textView);
        play= findViewById(R.id.play);
        previous= findViewById(R.id.previous);
        next= findViewById(R.id.next);
        seekBar= findViewById(R.id.seekBar);


        Intent intent= getIntent();
```

```
        Bundle bundle=intent.getExtras();
                                songs=    (ArrayList)
bundle.getParcelableArrayList("songList");
                                        textContent=
intent.getStringExtra("currentSong");
        textView.setText(textContent);
        textView.setSelected(true);
                                            position=
intent.getIntExtra("position",0);
                                        Uri     uri=
Uri.parse(songs.get(position).toString());
                                        mediaPlayer=
MediaPlayer.create(this,uri);
        mediaPlayer.start();


seekBar.setMax(mediaPlayer.getDuration());



          seekBar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
              @Override
```

```
                                        public   void
onProgressChanged(SeekBar     seekBar,     int
progress, boolean fromUser) {

            }

            @Override
                                        public   void
onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
                                        public   void
onStopTrackingTouch(SeekBar seekBar) {

mediaPlayer.seekTo(seekBar.getProgress());
            }
        });

        updateSeek= new Thread(){
```

```
            @Override
            public void run() {
                int currentPosition=0;
                try {
                                                    while
(currentPosition<mediaPlayer.getDuration()){
                                        currentPosition=
mediaPlayer.getCurrentPosition();

seekBar.setProgress(currentPosition);
                                sleep(800);
                            }
                        }
                        catch (Exception e){
                            e.printStackTrace();
                        }
                    }
                };
                updateSeek.start();
```

```
                play.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (mediaPlayer.isPlaying()){

play.setImageResource(R.drawable.play);
                    mediaPlayer.pause();

                }
                else {

play.setImageResource(R.drawable.pause);
                    mediaPlayer.start();
                }

            }
        });

                previous.setOnClickListener(new
View.OnClickListener() {
```

```
            @Override
            public void onClick(View v) {
                mediaPlayer.stop();
                mediaPlayer.release();
                if (position!=0){
                    position= position-1;
                }
                else {
                    position=songs.size()-1;
                }
                                        Uri   uri=
Uri.parse(songs.get(position).toString());
                                        mediaPlayer=
MediaPlayer.create(getApplicationContext(),uri
);
                mediaPlayer.start();

play.setImageResource(R.drawable.pause);

seekBar.setMax(mediaPlayer.getDuration());
```

```
                                                textContent=
songs.get(position).getName().toString();
                    textView.setText(textContent);


                }
            });


                        next.setOnClickListener(new
View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    mediaPlayer.stop();
                    mediaPlayer.release();
                    if (position!= songs.size()-1){
                        position= position+1;
                    }
                    else {
                        position=0;
                    }
                                                Uri   uri=
Uri.parse(songs.get(position).toString());
```

```
                                        mediaPlayer=
MediaPlayer.create(getApplicationContext(),uri
);
                    mediaPlayer.start();

play.setImageResource(R.drawable.pause);

seekBar.setMax(mediaPlayer.getDuration());
                                        textContent=
songs.get(position).getName().toString();
                    textView.setText(textContent);

                }
            });



    }
}
```

4. activity_play_song.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintL
ayout
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=".PlaySong">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="256dp"
        android:layout_height="0dp"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="114dp"
```

```
app:layout_constraintBottom_toTopOf="@+id/tex
tView"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/logo" />

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="314dp"
        android:text="TextView"

android:marqueeRepeatLimit="marquee_forever"
        android:ellipsize="marquee"
        android:scrollHorizontally="true"
```

```
        android:singleLine="true"
        android:textSize="24sp"
        android:textStyle="bold"
        android:fadingEdge="horizontal"

app:layout_constraintBottom_toBottomOf="paren
t"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="0.498"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/ima
geView2" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="314dp"
        android:layout_height="198dp"
```

```
        android:orientation="horizontal"

app:layout_constraintBottom_toBottomOf="paren
t"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="0.5"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/tex
tView">

        <ImageView
            android:id="@+id/previous"
            android:layout_width="104dp"

android:layout_height="match_parent"
             app:srcCompat="@drawable/previous"
/>
```

```
        <ImageView
            android:id="@+id/play"
            android:layout_width="104dp"

android:layout_height="match_parent"
            app:srcCompat="@drawable/pause" />

        <ImageView
            android:id="@+id/next"
            android:layout_width="104dp"

android:layout_height="match_parent"
            app:srcCompat="@drawable/next" />
    </LinearLayout>

    <SeekBar
        android:id="@+id/seekBar"
        android:layout_width="371dp"
        android:layout_height="21dp"
```

```
app:layout_constraintBottom_toTopOf="@+id/lin
earLayout"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/tex
tView" />

</androidx.constraintlayout.widget.Constraint
Layout>
```

5. **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintL
ayout
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=".PlaySong">

    <ImageView
        android:id="@+id/imageView2"
        android:layout_width="256dp"
        android:layout_height="0dp"
        android:layout_marginTop="16dp"
        android:layout_marginBottom="114dp"

app:layout_constraintBottom_toTopOf="@+id/tex
tView"

app:layout_constraintEnd_toEndOf="parent"
```

```
app:layout_constraintStart_toStartOf="parent"


app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/logo" />


    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="314dp"
        android:text="TextView"


android:marqueeRepeatLimit="marquee_forever"
        android:ellipsize="marquee"
        android:scrollHorizontally="true"
        android:singleLine="true"
        android:textSize="24sp"
        android:textStyle="bold"
        android:fadingEdge="horizontal"
```

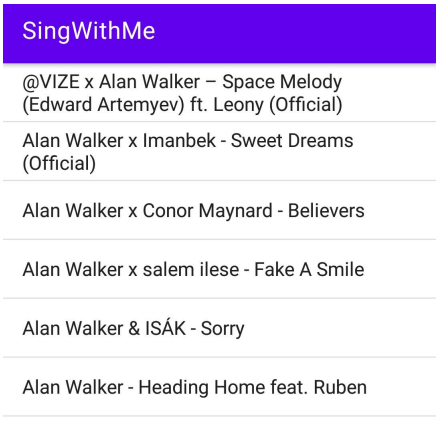
```
app:layout_constraintBottom_toBottomOf="paren
t"

app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="0.498"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/ima
geView2" />

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="314dp"
        android:layout_height="198dp"
        android:orientation="horizontal"

app:layout_constraintBottom_toBottomOf="paren
t"
```

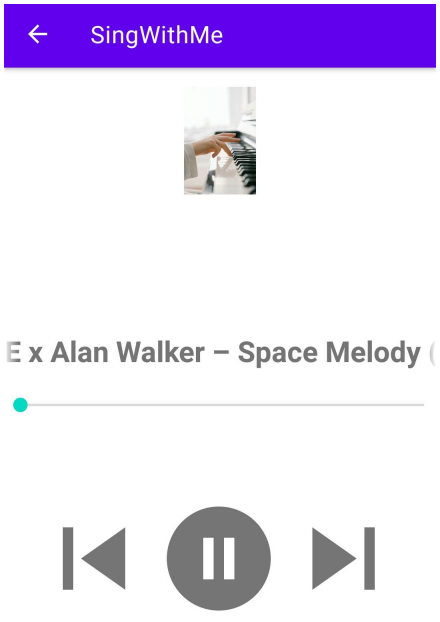
```
app:layout_constraintEnd_toEndOf="parent"

app:layout_constraintHorizontal_bias="0.5"

app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/tex
tView">

        <ImageView
            android:id="@+id/previous"
            android:layout_width="104dp"

android:layout_height="match_parent"
             app:srcCompat="@drawable/previous"
/>

        <ImageView
            android:id="@+id/play"
            android:layout_width="104dp"
```

```
android:layout_height="match_parent"
            app:srcCompat="@drawable/pause" />

        <ImageView
            android:id="@+id/next"
            android:layout_width="104dp"

android:layout_height="match_parent"
            app:srcCompat="@drawable/next" />
    </LinearLayout>

    <SeekBar
        android:id="@+id/seekBar"
        android:layout_width="371dp"
        android:layout_height="21dp"

app:layout_constraintBottom_toTopOf="@+id/lin
earLayout"

app:layout_constraintEnd_toEndOf="parent"
```

```
app:layout_constraintStart_toStartOf="parent"


app:layout_constraintTop_toBottomOf="@+id/tex
tView" />


</androidx.constraintlayout.widget.Constraint
Layout>
```

SCREEN SHOTS





SingWithMe

x Imanbek - Sweet Dreams (Offic

SingWithMe

x Conor Maynard - Believers.mp3

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 try to switch to the next song but they need lots of action like unlocking the phone, opening the app again in the background and looking for the switch button. In addition, the specific song

is overwhelmed by a large number of songs and causes information overload, users can only spend more energy and time to find it. For example, searching for a book in the library, and realizing 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.