Project Report

on

“Vibein”

2022-2023

FACULTY OF ENGINEERING & COMPUTING SCIENCES
TEERTHANKER MAHAVEER UNIVERSITY, MORADABAD

PROJECT GUIDE:
Mr. Harjinder Singh
Mr. Ajay Chakravarthy

SUBMITTED BY:
Udit Gupta
(TCA2009044)

ACKNOWLEDGEMENT

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

I would like to express my appreciation to my supervisor, Mr. Harjinder Singh and Mr. Ajay Chakravarthy 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.

Udit Gupta

Date:

Place:

DECLARATION

I hereby declare that this Project Report titled “VIBEIN” submitted by me and approved by mine project guide, the College of Computing Sciences and Information Technology (CCSIT), Teerthanker Mahaveer University, Moradabad, is a bonafide work undertaken by us and it is not submitted to any other University or Institution for the award of any degree diploma / certificate or published any time before.

Student Name: Udit Gupta Student Signature

Project Guide: Mr. Ajay Chakravarthy Signature
Mr. Harjinder Singh

Table of Contents

1 PROJECT TITLE 1

2 PROBLEM STATEMENT 1

3 PROJECT DESCRIPTION 3

3.1 SCOPE OF THE WORK 4

3.2 PROJECT MODULES 6

4 IMPLEMENTATION METHODOLOGY 26

5 TECHNOLOGIES TO BE USED 27

5.1 SOFTWARE PLATFORM 27

5.2 HARDWARE PLATFORM 27

5.3 TOOLS 27

6 ADVANTAGES OF THIS PROJECT 29

7 FUTURE SCOPE AND FURTHER ENHANCEMENT OF THE PROJECT 30

8 CONCLUSION 31

9 REFERENCES 32

Appendix

A: Entity-Relation(ER) Diagram

B: Screen Shots

1 Project Title

"Vibein"

The purpose of this project is to develop a offline music mp3 player which can play the mainstream file format. To browse and query the storage space as well as operation of playing can be realized. Meanwhile, this software can play, pause and select songs with previous button and next button according to sets requirement as well as set up songs. It also provides a positive vibes environment for which music is most popular that's why it has a project title names as "Vibein".

2 Problem Statement

Android is open-source code mobile phone operating system that comes out by Google. Music player in this project is application software based on Google Android. Music is one of the best ways to relieve pressure in stressful modern society life. The purpose of this project is to develop a player which can play the mainstream file format. To browse and query the storage space as well as operation of playing can be realized . Meanwhile, this software can play, pause and select songs with latest button and next button according to sets requirement as well as set up songs.

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

Economic feasibility:

To design Android mobile phone music player as long as a computer has the Android development and the application development of Android is free. In addition, mobile phone music player is basic needs for 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 saved the spending. Therefore, the whole process of development doesn't need to spend any money that is economic feasibility.

Technical feasibility:

To design a music player which meets the basic requirements, a deep understand of JAVA language, 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.

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 life, which derived the development of all kinds of mobile phone player. But a lot of players 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 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.

Target Audience:

- This app is mainly focused on, but not limited to individuals with low income.
- This application can also be useful to explorer and adventurers.
- For the music lovers who want and interactive UI music player app.

Saturation Overview:

This section describes requirements of the system based on basic control functions of players, and system setup function of the player according to research results of the project demand. According to the research results of project demand, the basic requirements of project system and its function structure are presented. And describe the demand of the system through the different angles. The project is divided into the following parts by using diagram: the basic control functions of the player, the playlist management function of the player and system setting function of the player. The player interface requires rational layout, comfortable color, friendly control buttons and concise and beautiful images. According to the Android system requires, the non- response time is 5 seconds. The following is requirements in the music player development application: Application response time shall not exceed 5 seconds in music playing. Application response time shall not exceed 5 seconds as the music is suspended. Application response time shall not exceed 5 seconds as the music is stopped. Application response time shall not exceed 5 seconds as Move Next/Move Previous music. Application response time shall not exceed 5 seconds as system listing is required.

3 Project Description

System design

The App Starting module of the player in the project is introduced, as well as the Android engineering program structure, etc. Any App Starting needs Android Manifest. XML file to start. And any new project content will automatically generate an Android Manifest. XML file. Configuration files are the core of the whole program, which contains the Android SDK version, and the default Activity in program running. The systems will automatically looking for a logo in Android Manifest to react the corresponding operation when any component of the program triggers events. To define the system, the first thing is launching the Activity: Android Activity. There are properties such as action and category in < intent - filter >. Most of these are the default values of the system. Setting the action and category realize the switch between different Activities. When any components of the program is about to use, declaration must be in the Android Manifest. Xml files. To be clear that authorities must be illustrated as the statement of provider. Each component has a lot of attributes; the program will define different attributes according to different needs. The basic structure content of Android project includes: the SRC (source code), gen (constant that Android system automatically generates), res (resource file), and the layout of file and pictures in the main storage program interface.

Part of the function design

The main play interface design. Convenience and practical should be fully considered in the design of the main interface. Every Android interface is a visual interface, which has its unique layout configuration files. We can configure various layout and resources files according to the requirements, such as images, text and color reference, which can form different visual interface and glaring effect. Interface design of adding songs. There are no corresponding songs for the first time login entering the program; users need to add songs to play. Therefore, you need to enter the adding songs' interface. The empty playlist needs to add songs which can choose from the SD card to add.

Function design of play and Next/Move Previous music

- When need to use the player to play appropriate music, click the play button to realize the function.
- When need to use the player to switch to the previous song, click on "Move Previous music" button to realize the function.
- When need to use the player to play the next song, click on "the next music" button to realize the function.

3.1 Scope of the Work

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 life, which derived the development of all kinds of mobile phone player. But a lot of players 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 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.

To design a music player which meets the basic requirements, a deep understand of JAVA language, 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.

Music is not just entertainment; it is more of a therapy. It is like our escape from whatever bad and stressful things are happening in our lives.

Do you remember those days when we used to buy cassettes and DVDs for listening to music, then the internet came, and we got the luxury to download all types of music from the pirate bay, the most popular torrent site?

Now, the technology has advanced, and we get applications that are dedicated only to music lovers. There we get both the options of listening to music online and offline. Here, we will talk about the advantages and also the disadvantages of listening to music online and offline.

Nowadays, smartphone has already to replace the normal mobile phone device. Mobile phone device is no longer only just can act as communication tool but also one of the daily needs for most of the people. Android system, one of the mobile operating system which also known as open-source platform are now getting more and more popular, especially in the smartphone market. Because of the open-source platform, there are a lot of applications had generated.

- Music is what feelings sound like and because of music are so inspiring people, it’s so important to get a music player application that can work well for the user.
- With the rising of the smartphone, the natural that now the one mobile device we carry with us is also our primary media player. Over the years music fans have gradually changed how they listen to their music and what they listen to it on. Therefore, music application has been being one of the important part for the mobile device.

These are the advantages of offline music listening.

•The biggest advantage of listening to music offline is you do not need an internet connection to listen to the music. Once you download the music, you can listen to it as much as you want to without having an internet connection.

•Offline music listening and playing also come with the advantages of being transferred between music playing devices without an active internet connection.

•As offline music playing does not need an active internet connection, you can listen to them from all those remote places as well where internet connection is not available, or the network is poor.

It totally depends on you and the situation.

You will be able to decide on the basis of your requirements whether you are going to listen to music online or offline. If your device does not have much space and you have a proper internet connection, online music will be best for you.

On the other hand, if you are in some remote places where you do not have a proper internet connection, you must go with offline music. After all, your main purpose is listening to music, and not how you are doing it.

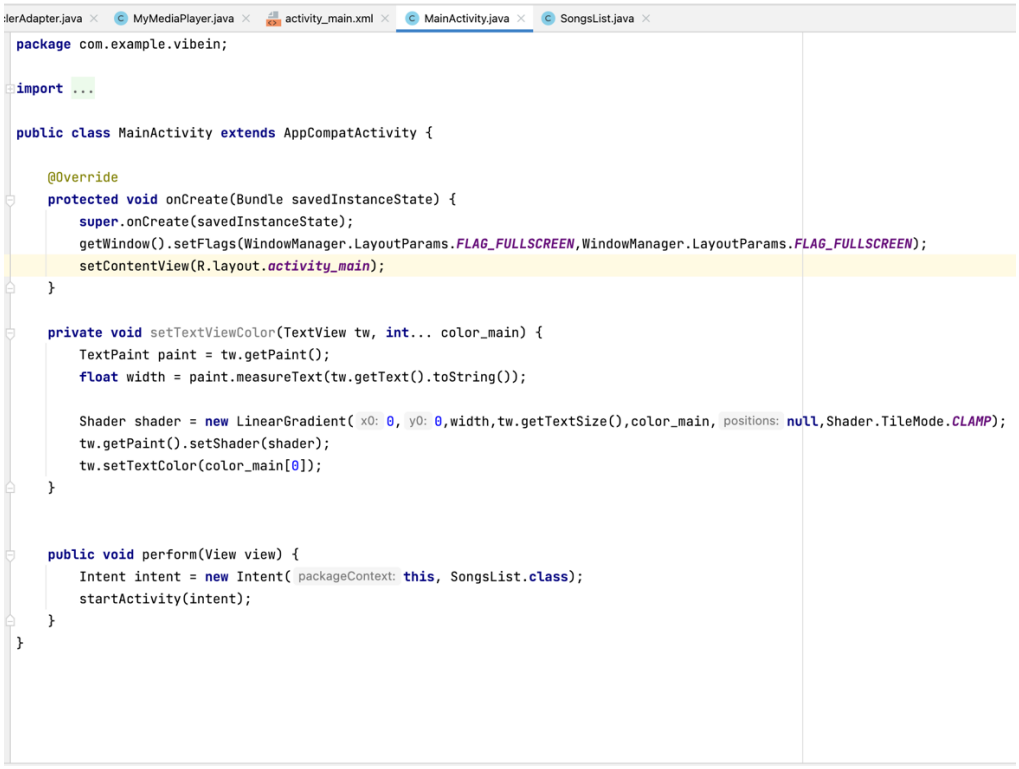
Target Audience:

- This app is mainly focused on, but not limited to individuals with low income.
- This application can also be useful to explorer and adventurers.

3.2 Project Modules

- **MainActivity.java**

  -> In this module the app is given a beautiful welcoming screen of the app where a button with a text “Let’s Vibe” to take the user to the next activity.



```
package com.example.vibein;

import ...

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,WindowManager.LayoutParams.FLAG_FULLSCREEN);
        setContentView(R.layout.activity_main);
    }

    private void setTextViewColor(TextView tw, int... color_main) {
        TextPaint paint = tw.getPaint();
        float width = paint.measureText(tw.getText().toString());

        Shader shader = new LinearGradient( x0: 0, y0: 0,width,tw.getTextSize(),color_main, positions: null,Shader.TileMode.CLAMP);
        tw.getPaint().setShader(shader);
        tw.setTextColor(color_main[0]);
    }


    public void perform(View view) {
        Intent intent = new Intent( packageContext: this, SongsList.class);
        startActivity(intent);
    }
}
```

- **SongsList.java**

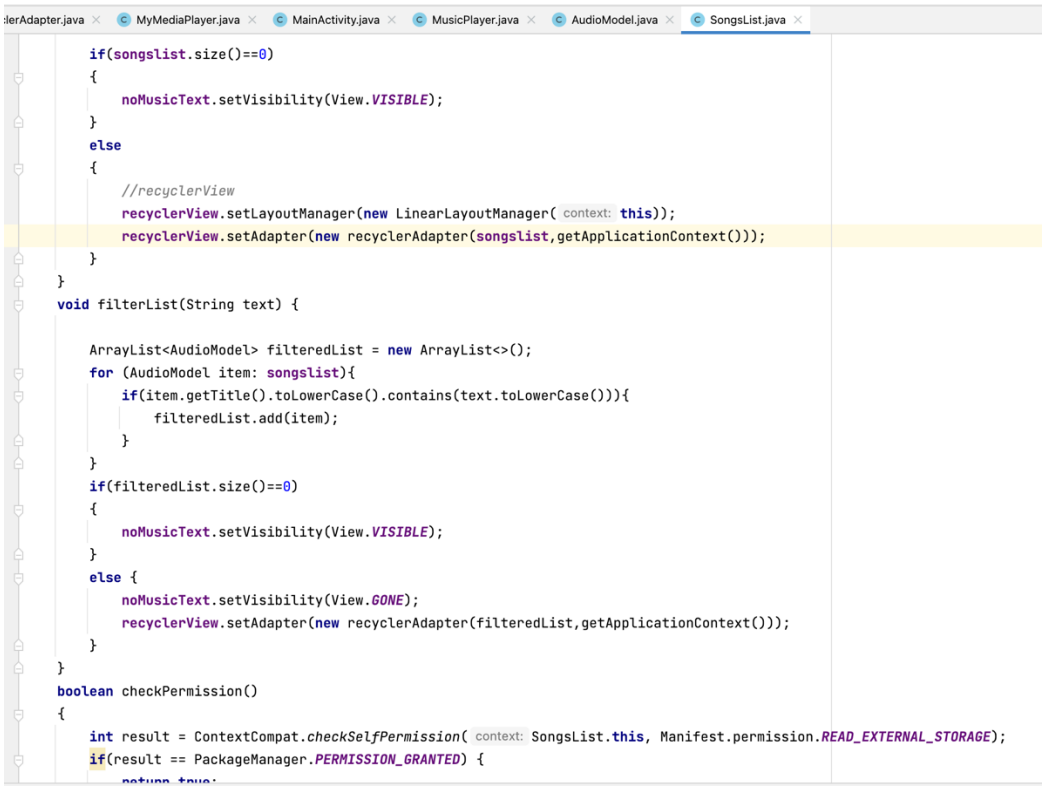
  - In this module the app is given functionality to read the external storage of the device.
    (MediaStore.Audio.Media.***EXTERNAL_CONTENT_URI***,projection,selection,**null**,**null**);
  - And then assigning recycler view to show the names of all the songs from the list.
  - A Search box is also present there for the user to search their song directly from the list using a filterList function that stores all the matched songs in the list named as filteredList.
    ArrayList<AudioModel> filteredList = **new** ArrayList<>();

lerAdapter.java | MyMediaPlayer.java | MainActivity.java | MusicPlayer.java | AudioModel.java | SongsList.java

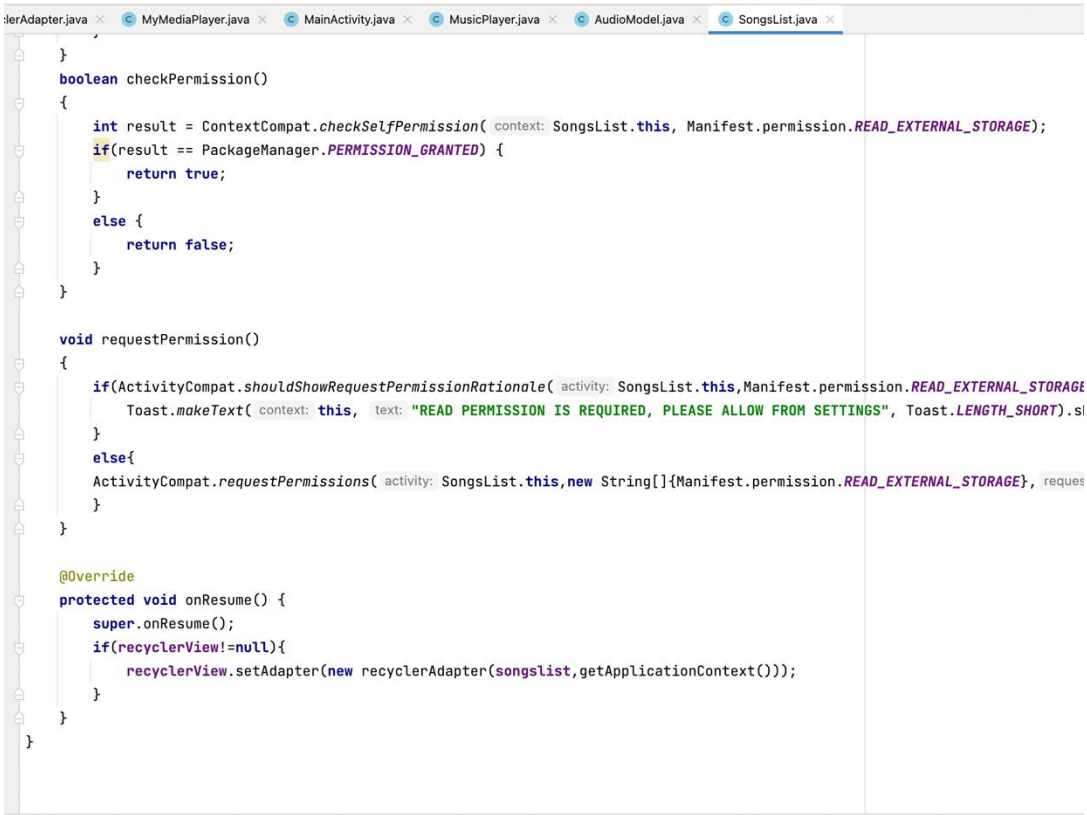
```java
package com.example.vibein;

import ...

public class SongsList extends AppCompatActivity {

    RecyclerView recyclerView;
    TextView noMusicText;
    ArrayList<AudioModel> songslist = new ArrayList<>();
    SearchView searchView;


    @SuppressLint("SuspiciousIndentation")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_songs_list);
        recyclerView = findViewById(R.id.list_of_songs);
        noMusicText = findViewById(R.id.no_songs_text);


        searchView = findViewById(R.id.searchbox);
        searchView.clearFocus();
        searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String query) {
                filterList(query);
                return false;
            }

            @Override
            public boolean onQueryTextChange(String newText) {
                filterList(newText);
```

lerAdapter.java | MyMediaPlayer.java | MainActivity.java | MusicPlayer.java | AudioModel.java | SongsList.java

```java
            @Override
            public boolean onQueryTextChange(String newText) {
                filterList(newText);
                return false;
            }
        });

        if(checkPermission()==false)
        {
            requestPermission();
            return;
        }
        String[] projection = {
                MediaStore.Audio.Media.TITLE,
                MediaStore.Audio.Media.DATA,
                MediaStore.Audio.Media.DURATION
        };
        String selection = MediaStore.Audio.Media.IS_MUSIC + " != 0";

        Cursor cursor = getContentResolver().query(MediaStore.Audio.Media.EXTERNAL_CONTENT_URI,projection,selection, selectionArg
        while(cursor.moveToNext())
        {
            AudioModel songData = new AudioModel(cursor.getString( i: 1),cursor.getString( i: 0),cursor.getString( i: 2));
            if(new File(songData.getPath()).exists())
            songslist.add(songData);

        }

        if(songslist.size()==0)
        {
            noMusicText.setVisibility(View.VISIBLE);
        }
        else
```



```
if(songslist.size()==0)
{
    noMusicText.setVisibility(View.VISIBLE);
}
else
{
    //recyclerView
    recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
    recyclerView.setAdapter(new recyclerAdapter(songslist,getApplicationContext()));
}
}
void filterList(String text) {

    ArrayList<AudioModel> filteredList = new ArrayList<>();
    for (AudioModel item: songslist){
        if(item.getTitle().toLowerCase().contains(text.toLowerCase())){
            filteredList.add(item);
        }
    }
    if(filteredList.size()==0)
    {
        noMusicText.setVisibility(View.VISIBLE);
    }
    else {
        noMusicText.setVisibility(View.GONE);
        recyclerView.setAdapter(new recyclerAdapter(filteredList,getApplicationContext()));
    }
}
boolean checkPermission()
{
    int result = ContextCompat.checkSelfPermission( context: SongsList.this, Manifest.permission.READ_EXTERNAL_STORAGE);
    if(result == PackageManager.PERMISSION_GRANTED) {
```



```
}
boolean checkPermission()
{
    int result = ContextCompat.checkSelfPermission( context: SongsList.this, Manifest.permission.READ_EXTERNAL_STORAGE);
    if(result == PackageManager.PERMISSION_GRANTED) {
        return true;
    }
    else {
        return false;
    }
}

void requestPermission()
{
    if(ActivityCompat.shouldShowRequestPermissionRationale( activity: SongsList.this,Manifest.permission.READ_EXTERNAL_STORAGE
        Toast.makeText( context: this,  text: "READ PERMISSION IS REQUIRED, PLEASE ALLOW FROM SETTINGS", Toast.LENGTH_SHORT).sh
    }
    else{
    ActivityCompat.requestPermissions( activity: SongsList.this,new String[]{Manifest.permission.READ_EXTERNAL_STORAGE}, reques
    }
}

@Override
protected void onResume() {
    super.onResume();
    if(recyclerView!=null){
        recyclerView.setAdapter(new recyclerAdapter(songslist,getApplicationContext()));
    }
}
}
```

- **recyclerAdapter.java**

recyclerAdapter.java | MyMediaPlayer.java | MainActivity.java | MusicPlayer.java | AudioModel.java | SongsList.java

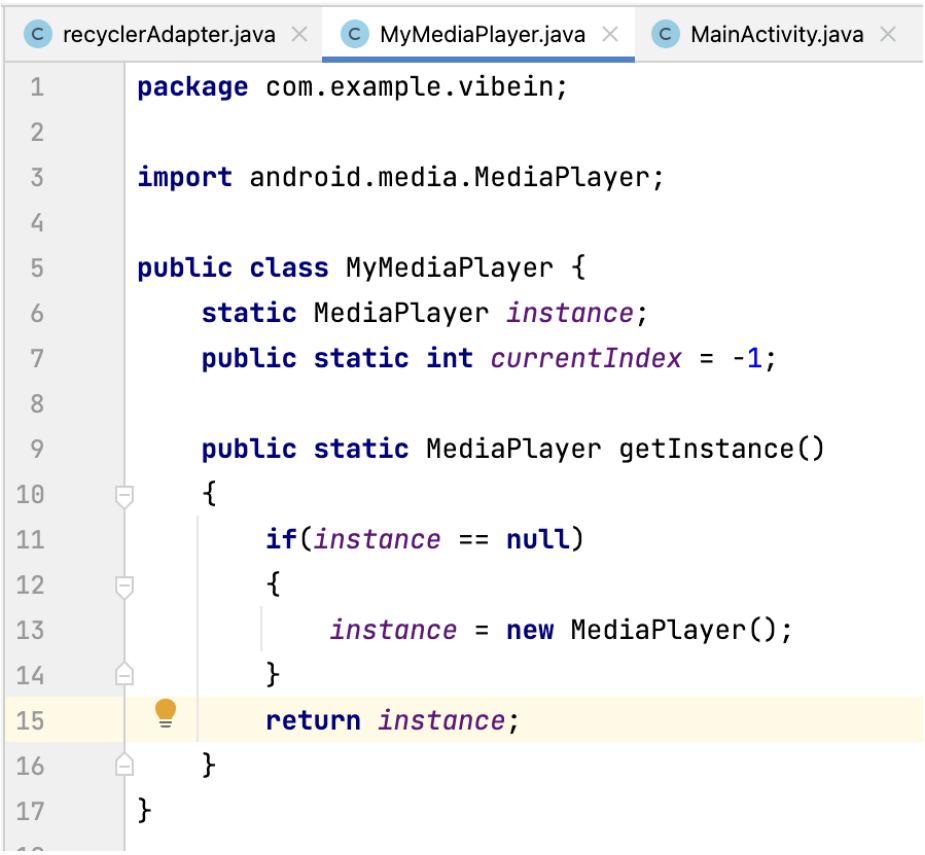
```java
package com.example.vibein;

import ...

public class recyclerAdapter extends RecyclerView.Adapter<recyclerAdapter.myViewHolder> {
    private ArrayList<AudioModel> songlist;
    Context context;

    public recyclerAdapter(ArrayList<AudioModel> songlist, Context context) {
        this.songlist = songlist;
        this.context = context;
    }

    @NonNull
    @Override
    public recyclerAdapter.myViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.list_items,parent, attachToRoot: false);
        return new recyclerAdapter.myViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull recyclerAdapter.myViewHolder holder, @SuppressLint("RecyclerView") int position) {
        AudioModel songData = songlist.get(position);
        holder.songName.setText(songData.getTitle());

        if(MyMediaPlayer.currentIndex==position){
            holder.songName.setTextColor(Color.parseColor( colorString: "#FF0000"));
        }
        else{
            holder.songName.setTextColor(Color.parseColor( colorString: "#000000"));
        }

        holder.itemView.setOnClickListener(new View.OnClickListener() {
```

recyclerAdapter.java | MyMediaPlayer.java | MainActivity.java | MusicPlayer.java | AudioMod

```java
        }

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                MyMediaPlayer.getInstance().reset();
                MyMediaPlayer.currentIndex = position;
                Intent intent = new Intent(context,MusicPlayer.class);
                intent.putExtra( name: "List",songlist);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(intent);
            }
        });

    }

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

    public class myViewHolder extends RecyclerView.ViewHolder{
        TextView songName;
        ImageView imageView;

        public myViewHolder(View itemView)
        {
            super(itemView);
            songName = itemView.findViewById(R.id.music_title);
            imageView = itemView.findViewById(R.id.music_note);
            imageView.setImageResource(R.drawable.musical_note);
        }
    }
}
```

- **AudioModel.java**

- **MyMediaPlayer.java**



- **MusicPlayer.java**

recyclerAdapter.java | MyMediaPlayer.java | MainActivity.java | MusicPlayer.java | AudioModel.java | SongsList.java

```java
package com.example.vibein;

import ...

public class MusicPlayer extends AppCompatActivity {

    TextView title,currentTime,totalTime;
    SeekBar seekBar;
    ImageView pausePlay,nextBtn,previousBtn;
    ArrayList<AudioModel> songslist;
    AudioModel currentSong;
    MediaPlayer mediaPlayer = MyMediaPlayer.getInstance();


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

        title = findViewById(R.id.music_title);
        currentTime = findViewById(R.id.current_time);
        totalTime = findViewById(R.id.total_time);
        seekBar = findViewById(R.id.seekbar);
        pausePlay = findViewById(R.id.pause_play);
        nextBtn = findViewById(R.id.next);
        previousBtn = findViewById(R.id.previous);

        title.setSelected(true);
        songslist = (ArrayList<AudioModel>) getIntent().getSerializableExtra( name: "List");

        setMusic();

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

recyclerAdapter.java | MyMediaPlayer.java | MainActivity.java | MusicPlayer.java | AudioModel.java | SongsList.java

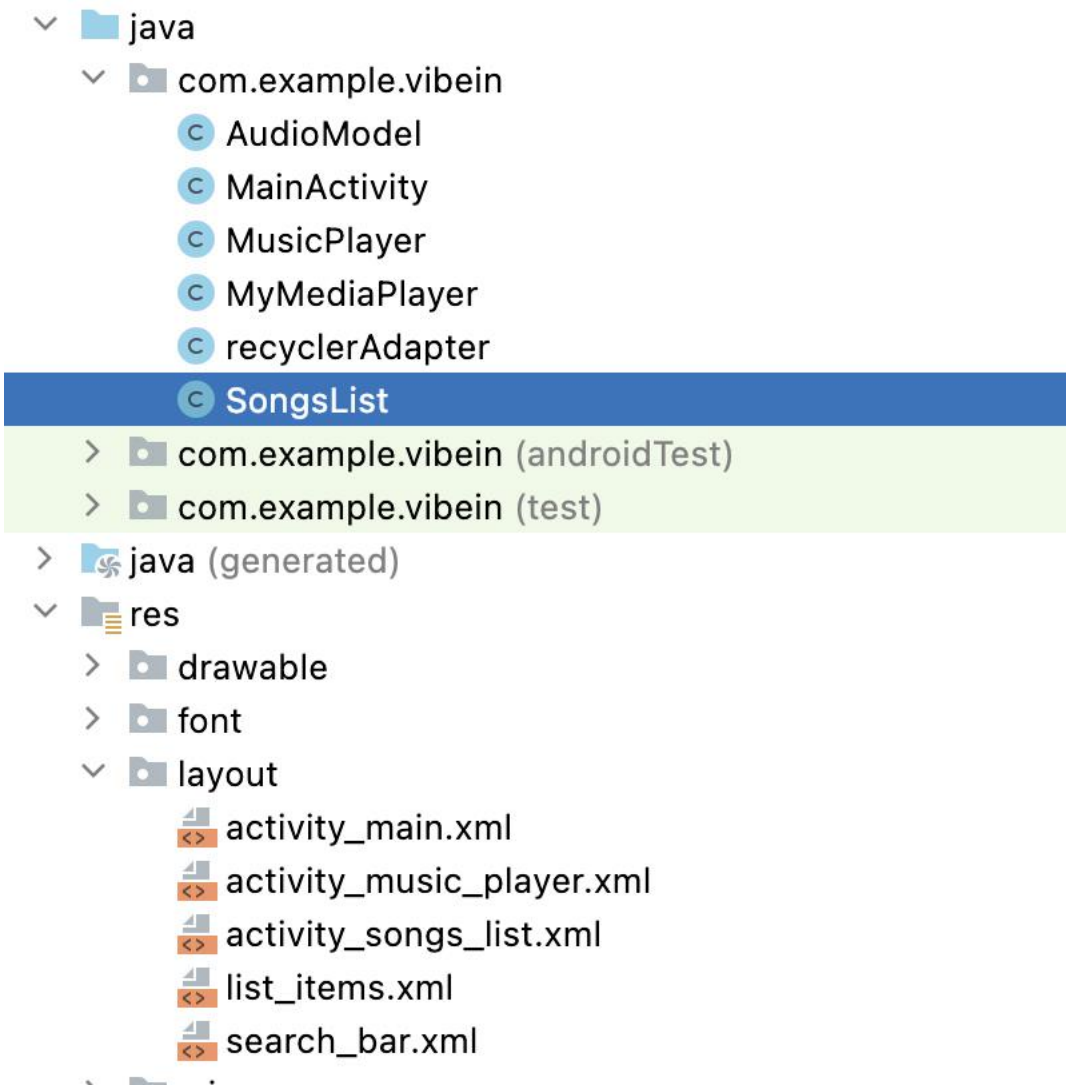
```java
        MusicPlayer.this.runOnUiThread(new Runnable() {
            @Override
            public void run() {
                if(mediaPlayer!=null){
                    seekBar.setProgress(mediaPlayer.getCurrentPosition());
                    currentTime.setText(convertToMin( time: mediaPlayer.getCurrentPosition()+""));

                    if(mediaPlayer.isPlaying()){
                        pausePlay.setImageResource(R.drawable.pause);
//                        musicIcon.setRotation(x++);
                    }else{
                        pausePlay.setImageResource(R.drawable.play);
//                        musicIcon.setRotation(0);
                    }

                }
                new Handler().postDelayed( r: this, delayMillis: 100);
            }
        });


        seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
                if(mediaPlayer!=null && fromUser){
                    mediaPlayer.seekTo(progress);
                }
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
```

recyclerAdapter.java | MyMediaPlayer.java | MainActivity.java | MusicPlayer.java | AudioModel.java | SongsList.java

```
78              @Override
79              public void onStopTrackingTouch(SeekBar seekBar) {
80
81              }
82          });
83      }
84      void setMusic(){
85          currentSong = songslist.get(MyMediaPlayer.currentIndex);
86
87          title.setText(currentSong.getTitle());
88          totalTime.setText(convertToMin(currentSong.getDuration()));
89
90          pausePlay.setOnClickListener(v-> pausePlay());
91          nextBtn.setOnClickListener(v-> playNextSong());
92          previousBtn.setOnClickListener(v-> playPreviousSong());
93
94          playMusic();
95      }
96
97      private void playMusic(){
98
99          mediaPlayer.reset();
100         try {
101             mediaPlayer.setDataSource(currentSong.getPath());
102             mediaPlayer.prepare();
103             mediaPlayer.start();
104             seekBar.setProgress(0);
105             seekBar.setMax(mediaPlayer.getDuration());
106         } catch (IOException e) {
107             e.printStackTrace();
108         }
109
110
```

recyclerAdapter.java | MyMediaPlayer.java | MainActivity.java | MusicPlayer.java | AudioModel.java | SongsList.java

```
112     private void playNextSong(){
113
114         if(MyMediaPlayer.currentIndex== songslist.size()-1)
115             return;
116         MyMediaPlayer.currentIndex +=1;
117         mediaPlayer.reset();
118         setMusic();
119
120     }
121     private void pausePlay(){
122         if(mediaPlayer.isPlaying())
123             mediaPlayer.pause();
124         else
125             mediaPlayer.start();
126     }
127
128     private void playPreviousSong(){
129         if(MyMediaPlayer.currentIndex== 0)
130             return;
131         MyMediaPlayer.currentIndex -=1;
132         mediaPlayer.reset();
133         setMusic();
134     }
135
136 @   public static String convertToMin(String time){
137         Long milis = Long.parseLong(time);
138         return String.format("%02d:%02d",
139                 TimeUnit.MILLISECONDS.toMinutes(milis) % TimeUnit.HOURS.toMinutes( duration: 1),
140                 TimeUnit.MILLISECONDS.toSeconds(milis) % TimeUnit.MINUTES.toSeconds( duration: 1));
141     }
142
143 }
```



- **Xml layout Modules**
  - activity_main.xml
  - activity_music_player.xml
  - activity_songs_list.xml
  - list_items.xml
  - search_bar.xml

- **activity_main.xml**

recyclerAdapter.java | MyMediaPlayer.java | MainActivity.java | MusicPlayer.java | activity_main.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.and
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:background="@drawable/purple_smoke"
8      tools:context=".MainActivity">
9
10     <ImageView
11         android:id="@+id/imageView2"
12         android:layout_width="306dp"
13         android:layout_height="306dp"
14         android:layout_marginStart="30dp"
15         android:layout_marginTop="92dp"
16         android:layout_marginEnd="30dp"
17         app:layout_constraintEnd_toEndOf="parent"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toTopOf="parent"
20         app:srcCompat="@drawable/headset" />
21
22
23     <Button
24         android:id="@+id/get_start_btn"
25         android:layout_width="250dp"
26         android:layout_height="wrap_content"
27         android:layout_marginVertical="100dp"
28         android:layout_marginStart="100dp"
29         android:layout_marginTop="70dp"
30         android:layout_marginEnd="100dp"
31         android:layout_marginBottom="100dp"
```

recyclerAdapter.java | MyMediaPlayer.java | MainActivity.java | MusicPlayer.java | activity_main.xml

```
30          android:layout_marginEnd="100dp"
31          android:layout_marginBottom="100dp"
32          android:background="@drawable/button1"
33          android:fontFamily="@font/audiowide"
34          android:onClick="perform"
35          android:padding="5dp"
36          android:text="Let's Vibe ->"
37          android:textColor="@android:color/white"
38          android:textSize="20sp"
39          app:layout_constraintBottom_toBottomOf="parent"
40          app:layout_constraintEnd_toEndOf="parent"
41          app:layout_constraintStart_toStartOf="parent"
42          app:layout_constraintTop_toBottomOf="@+id/imageView2"
43          app:layout_constraintVertical_bias="0.723" />
44
45      <ImageView
46          android:id="@+id/imageView"
47          android:layout_width="254dp"
48          android:layout_height="274dp"
49          android:layout_marginStart="28dp"
50          android:layout_marginTop="24dp"
51          android:layout_marginEnd="24dp"
52          app:layout_constraintBottom_toBottomOf="parent"
53          app:layout_constraintEnd_toEndOf="@+id/imageView2"
54          app:layout_constraintHorizontal_bias="0.0"
55          app:layout_constraintStart_toEndOf="@+id/imageView2"
56          app:layout_constraintStart_toStartOf="@+id/imageView2"
57          app:layout_constraintTop_toTopOf="parent"
58          app:layout_constraintVertical_bias="0.512"
59          app:srcCompat="@drawable/vibein_logo" />
60
```

- **activity_music_player.xml**

activity_music_player.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/purple_smoke"
    tools:context=".MusicPlayer">


    <TextView
        android:id="@+id/music_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingHorizontal="10dp"
        android:layout_marginStart="40dp"
        android:layout_marginTop="50dp"
        android:layout_marginEnd="40dp"
        android:layout_marginBottom="50dp"
        android:ellipsize="marquee"
        android:singleLine="true"
        android:text="TextView"
        android:textColor="@color/white"
        android:textSize="20dp" />


    <pl.droidsonroids.gif.GifImageView
        android:layout_width="match_parent"
        android:layout_height="323dp"
        android:layout_above="@id/controls"
        android:layout_alignParentEnd="true"
```

activity_music_player.xml

```
30          android:layout_above="@id/controls"
31          android:layout_alignParentEnd="true"
32          android:layout_marginTop="100dp"
33          android:layout_marginEnd="0dp"
34          android:src="@drawable/visualizer" />
35
36      <RelativeLayout
37          android:id="@+id/controls"
38          android:layout_width="match_parent"
39          android:layout_height="wrap_content"
40          android:layout_alignParentBottom="true">
41
42          <SeekBar
43              android:id="@+id/seekbar"
44              android:layout_width="match_parent"
45              android:layout_height="15dp"
46              android:layout_marginStart="40dp"
47              android:layout_marginTop="40dp"
48              android:layout_marginEnd="40dp"
49              android:layout_marginBottom="0dp"
50              android:background="@drawable/button1"
51              android:foregroundTint="#FFFFFF"
52              android:progress="50"
53              android:progressBackgroundTint="#FFFFFF"
54              android:progressTint="#FFFFFF"
55              android:secondaryProgressTint="#FFFFFF"
56              android:thumbTint="#FFFFFF"
57              app:tickMarkTint="#FFFFFF" />
58
59          <TextView
60              android:id="@+id/current_time"
```

activity_music_player.xml

```
58
59      <TextView
60          android:id="@+id/current_time"
61          android:layout_width="wrap_content"
62          android:layout_height="wrap_content"
63          android:layout_below="@id/seekbar"
64          android:layout_alignParentStart="true"
65          android:layout_marginStart="40dp"
66          android:layout_marginTop="10dp"
67          android:layout_marginEnd="40dp"
68          android:layout_marginBottom="20dp"
69          android:text="0:00"
70          android:textColor="@color/white" />
71
72      <TextView
73          android:id="@+id/total_time"
74          android:layout_width="wrap_content"
75          android:layout_height="wrap_content"
76          android:layout_below="@id/seekbar"
77          android:layout_alignParentEnd="true"
78          android:layout_marginStart="40dp"
79          android:layout_marginTop="10dp"
80          android:layout_marginEnd="40dp"
81          android:layout_marginBottom="20dp"
82          android:text="0:00"
83          android:textColor="@color/white" />
84
85      <RelativeLayout
86          android:layout_width="match_parent"
87          android:layout_height="wrap_content"
88          android:layout_below="@id/total_time"
```

activity_music_player.xml

```
88          android:layout_below="@id/total_time"
89          android:layout_marginBottom="120dp"
90          android:padding="20dp">
91
92          <ImageView
93              android:id="@+id/previous"
94              android:layout_width="38dp"
95              android:layout_height="38dp"
96              android:layout_alignParentStart="true"
97              android:layout_centerVertical="true"
98              android:layout_marginLeft="36dp"
99              android:src="@drawable/prev" />
100
101         <ImageView
102             android:id="@+id/next"
103             android:layout_width="38dp"
104             android:layout_height="38dp"
105             android:layout_alignParentEnd="true"
106             android:layout_centerVertical="true"
107             android:layout_marginRight="36dp"
108             android:src="@drawable/next1" />
109
110         <ImageView
111             android:id="@+id/pause_play"
112             android:layout_width="64dp"
113             android:layout_height="64dp"
114             android:layout_centerInParent="true"
115             android:src="@drawable/pause" />
116     </RelativeLayout>
117 </RelativeLayout>
118 </RelativeLayout>
```

- **activity_songs_list.xml**

activity_music_player.xml | activity_songs_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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=".SongsList"
    android:background="@drawable/purple_smoke"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:orientation="horizontal"
        android:layout_marginTop="10dp"
        android:paddingLeft="7dp">


        <ImageView
            android:id="@+id/imageView4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="6dp"
            android:layout_marginEnd="86dp"
            android:layout_marginBottom="1dp"
            app:srcCompat="@drawable/playlist" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
```

activity_music_player.xml | activity_songs_list.xml

```
30      <LinearLayout
31          android:layout_width="match_parent"
32          android:layout_height="wrap_content">
33
34          <androidx.appcompat.widget.SearchView
35              android:id="@+id/searchbox"
36              android:layout_width="match_parent"
37              android:layout_height="45dp"
38              android:layout_marginHorizontal="10dp"
39              android:layout_marginBottom="20dp"
40              android:background="@drawable/searchbox_bg"
41              app:closeIcon="@drawable/close_logo"
42              app:iconifiedByDefault="false"
43              app:queryHint="Search Here..."
44              app:searchIcon="@drawable/search_logo"
45              />
46
47      </LinearLayout>
48      <TextView
49          android:id="@+id/no_songs_text"
50          android:layout_width="match_parent"
51          android:layout_height="match_parent"
52          android:layout_margin="10dp"
53          android:background="@drawable/border"
54          android:gravity="center"
55          android:text="NO SONGS FOUND"
56          android:textAlignment="gravity"
57          android:textSize="20dp"
58          android:visibility="gone"
59          />
60
61      <androidx.recyclerview.widget.RecyclerView
62          android:id="@+id/list_of_songs"
63          android:layout_width="match_parent"
64          android:layout_height="match_parent" />
```

LinearLayout › LinearLayout

activity_music_player.xml | activity_songs_list.xml

```
30      <LinearLayout
31          android:layout_width="match_parent"
32          android:layout_height="wrap_content">
33
34          <androidx.appcompat.widget.SearchView
35              android:id="@+id/searchbox"
36              android:layout_width="match_parent"
37              android:layout_height="45dp"
38              android:layout_marginHorizontal="10dp"
39              android:layout_marginBottom="20dp"
40              android:background="@drawable/searchbox_bg"
41              app:closeIcon="@drawable/close_logo"
42              app:iconifiedByDefault="false"
43              app:queryHint="Search Here..."
44              app:searchIcon="@drawable/search_logo"
45              />
46      </LinearLayout>
47      <TextView
48          android:id="@+id/no_songs_text"
49          android:layout_width="match_parent"
50          android:layout_height="match_parent"
51          android:layout_margin="10dp"
52          android:background="@drawable/border"
53          android:gravity="center"
54          android:text="NO SONGS FOUND"
55          android:textAlignment="gravity"
56          android:textSize="20dp"
57          android:visibility="gone"
58          />
59      <androidx.recyclerview.widget.RecyclerView
60          android:id="@+id/list_of_songs"
61          android:layout_width="match_parent"
62          android:layout_height="match_parent" />
63  </LinearLayout>
```

LinearLayout › LinearLayout › androidx.appcompat.widget.SearchView

- **activity_songs_list.xml**

activity_music_player.xml | list_items.xml

```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="60dp"
    android:background="@drawable/list_design">
    <ImageView
        android:id="@+id/music_note"
        android:layout_width="45dp"
        android:layout_height="45dp"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:layout_alignParentStart="true"
        android:layout_marginLeft="10dp"
        android:visibility="visible"
        app:srcCompat="@drawable/musical_note"/>

    <TextView
        android:id="@+id/music_title"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_toEndOf="@id/music_note"
        android:gravity="center_vertical"
        android:paddingLeft="10dp"
        android:text="TextView"
        android:textSize="15dp"
        android:maxLines="1"
        android:ellipsize="end"
        android:textColor="@color/black"/>
</RelativeLayout>
```

- **activity_songs_list.xml**

list_items.xml search_bar.xml

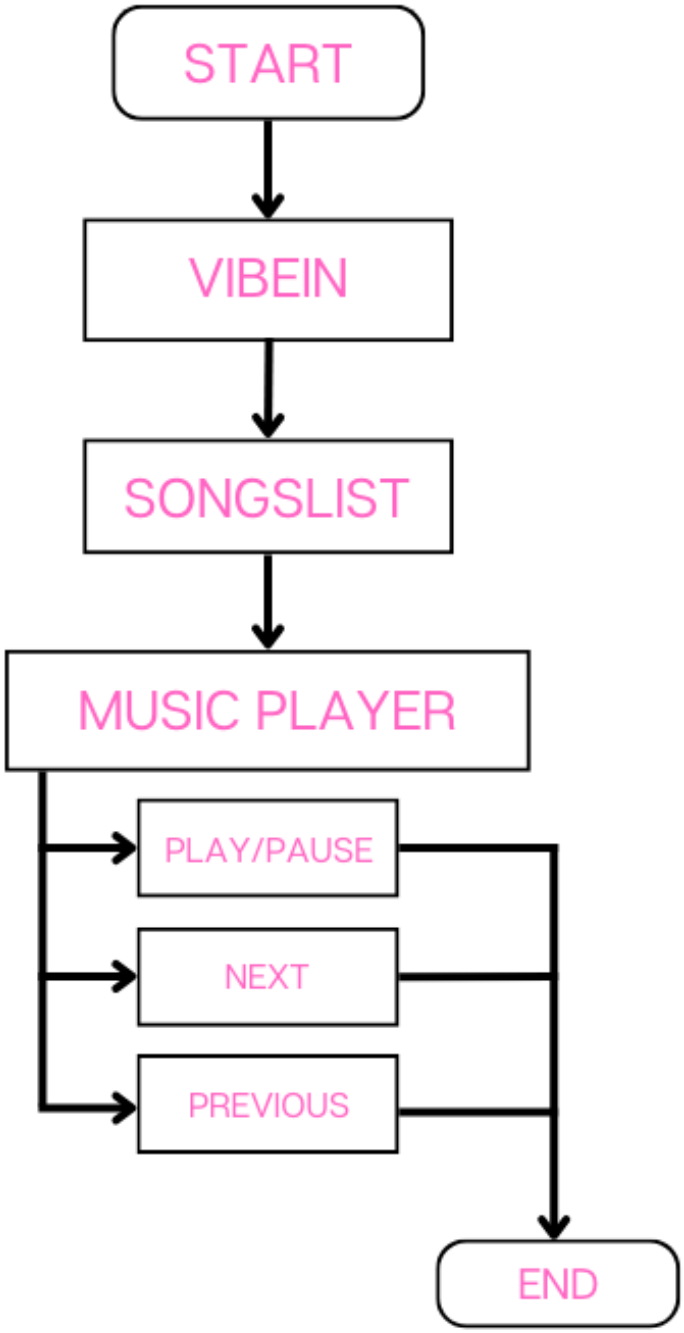
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout

        android:layout_width="match_parent"
        android:layout_height="50dp"
        android:layout_marginStart="20dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="20dp"
        android:layout_marginBottom="20dp"
        android:background="@color/white"
        android:backgroundTint="#B3000000"
        android:backgroundTintMode="screen"
        android:gravity="center_horizontal">


        <EditText
            android:id="@+id/editTextTextPersonName"
            android:layout_width="200dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="20dp"
            android:ems="15"
            android:inputType="textPersonName"
            android:text="Search"
            android:padding="5dp"
            android:textSize="25dp"
            />

        <ImageView
            android:id="@+id/logo_search"
            android:layout_width="40dp"
```

list_items.xml search_bar.xml

```
16      android:backgroundTint="#B3000000"
17      android:backgroundTintMode="screen"
18      android:gravity="center_horizontal">
19
20
21      <EditText
22          android:id="@+id/editTextTextPersonName"
23          android:layout_width="200dp"
24          android:layout_height="match_parent"
25          android:layout_marginLeft="20dp"
26          android:ems="15"
27          android:inputType="textPersonName"
28          android:text="Search"
29          android:padding="5dp"
30          android:textSize="25dp"
31          />
32
33      <ImageView
34          android:id="@+id/logo_search"
35          android:layout_width="40dp"
36          android:layout_height="40dp"
37          android:layout_marginStart="5dp"
38          android:layout_marginTop="5dp"
39          android:layout_marginEnd="5dp"
40          android:layout_marginBottom="5dp"
41          android:src="@drawable/search_logo"/>
42
43  </LinearLayout>
44
45  </LinearLayout>
```

4 Implementation Methodology

- **Context Diagram**



5 Technologies to be used

5.1 Software Platform

a) Front-end

The required software of the developing environment

- Operating system: Windows 10 and above
- Software：Android SDK(Software Development Kit), ADT(Android Development Tool)
- 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 – 18
    - Step 2: install the Android SDK: first download the Android SDK
    - Download address: http://developer-android- com/sdk /index-html
    - Input SDK tools path in the SDK location: D: \ android \ software \ android SDK–Windows and click OK.
    - The Android environment is set up successfully.

b) Back-end

Java programming is being used in developing the back-end part of the application And XML is being used in the development of back-end.

5.2 Hardware Platform

- This APP meets minimum hardware Requirements:
- >1 GHz Processor
- >512MB of RAM
- >50MB of Internal Storage

5.3 Tools

Android continues to be the most used operating system worldwide. And that means Android apps are extremely popular. Most companies who build mobile apps, create apps for multiple devices – both Android devices and iOS devices. To build an excellent Android app you need to use the best tools. Here's out top picks when it comes to Android development tools:

Android Studio

There's no talking about android app development without the Android Studio. It's the most basic tool for Android developers. Created by Google in 2013, it has pretty much become the

standard software for Android Developers. It's a great tool because it has the support of Google as well as a large community of developers.

Android Debug Bridge (ADB)

Android Debug Bridge is included in Android Studio and it's basically a line of communication between Android devices and other computers that developers use for QA and testing purposes. Android Developers can connect their Android device to their computer and make necessary changes to both devices at the same time.

Android Virtual Device (AVD) Manager

Another great feature of Android Studio is the AVD. This is an emulator that will run your Android app on your computer so that you have a better inside into what your code looks like. It's great for actually seeing the work you've done and making any adjustments as needed.

BUILD DEVELOPING ENEVIRONMENT OF ANDROID

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

The required software of the developing environment

Operating system: Windows 10 and above

Software ： Android SDK (Software Development Kit), ADT (Android Development Tool)

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 – 7

Step 2: install the Android SDK: first download the Android SDK

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

Input SDK tools path in the SDK location: D: \ android \ software \ android SDK– Windows and click OK.

The Android environment is set up successfully.

6 Advantages of this Project

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 life, which derived the development of all kinds of mobile phone player. But a lot of players 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 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.

Target Audience:

- This app is not limited to individuals with low income.
- This application can also be useful to explorer and adventurers

Nowadays, smartphone has already to replace the normal mobile phone device. Mobile phone device is no longer only just can act as communication tool but also one of the daily needs for most of the people. Android system, one of the mobile operating system which also known as open-source platform are now getting more and more popular, especially in the smartphone market. Because of the open-source platform, there are a lot of applications had generated. [1] Music is what feelings sound like and because of music are so inspiring people, it's so important to get a music player application that can work well for the user. [2] With the rising of the smartphone, the natural that now the one mobile device we carry with us is also our primary media player. Over the years music fans have gradually changed how they listen to their music and what they listen to it on. Therefore, music application has been being one of the important part for the mobile device.

These are the advantages of offline music listening.

- The biggest advantage of listening to music offline is you do not need an internet connection to listen to the music. Once you download the music, you can listen to it as much as you want to without having an internet connection.
- Offline music listening and playing also come with the advantages of being transferred between music playing devices without an active internet connection.
- As offline music playing does not need an active internet connection, you can listen to them from all those remote places as well where internet connection is not available, or the network is poor.

  It totally depends on you and the situation.

You will be able to decide on the basis of your requirements whether you are going to listen to music online or offline. If your device does not have much space and you have a proper internet connection, online music will be best for you.

On the other hand, if you are in some remote places where you do not have a proper internet connection, you must go with offline music. After all, your main purpose is listening to music, and not how you are doing it.

7 Future Scope and further enhancement of the Project

Every application needs to be improved according to the requirements and demands. Following are the improvements that are expected to be applied in future:

- The next thing this project needs is features. Our Mp3 Radio project only implements the minimal functions of a music player such as play, stop, next, and previous. Other players have things like fast forward, reverse, much more playlist control, or current song information. Much of this is supported in some way or another by VIBEIN; it would just be a matter of extending the project. Unfortunately, I didn't have the time to do that.

- Seeing the song play time displayed on the phone, along with the bit rate and other common Mp3 header information, would be a definite plus, and should be one of the first things to improve on the phone's interface. Real time song information would mean sending constant updates to the phone, which may or may not affect the efficiency of the player. However, it is likely that the extra messages will not be too much of a burden, and some packet loss would definitely be acceptable since the data could easily be interpolated.

- As far as extending playlist control, it would be nice to have full control over playlists. Using the phone, a user should be able to create new lists, load saved ones, and modify them on the fly. Currently

- Another feature of our project that could be done in the future is a file transfer system. Currently, there is no way to add or remove music from the compact flash except by removing the memory card and reading it on some other device. We need some sort of front end for the users to access the files with.

- There are two routes to take on this, either a network approach or a web approach. A front end could be built that would create an SSH session to the personal server via its wireless connection, or even use FTP protocols to transfer files in and out. This approach would be efficient and secure, but it is downfall is portability. The goal of Mp3 Radio is

portability. Taking your music anywhere means a user shouldn't be confined to a single computer with installed software to transfer files.

- A better approach is a web solution. The personal server runs an Apache web server, so why not take advantage of that? A web application could be created that would let you upload and manage files from any internet connection.

- Another thing to consider would be a peer-to-peer connection. Perhaps users could exchange files directly from personal server to personal server. This introduces more security issues and would be a whole other project, but would be an interesting thing to tackle.

- Something we could have done better if we did the project over again, would be to learn Symbian. A guide to programming in Symbian would have been a great thing to have, but we learned it all the hard way, and spent many frustrating hours not getting anywhere. Similarly, another valuable lesson can be learned here.

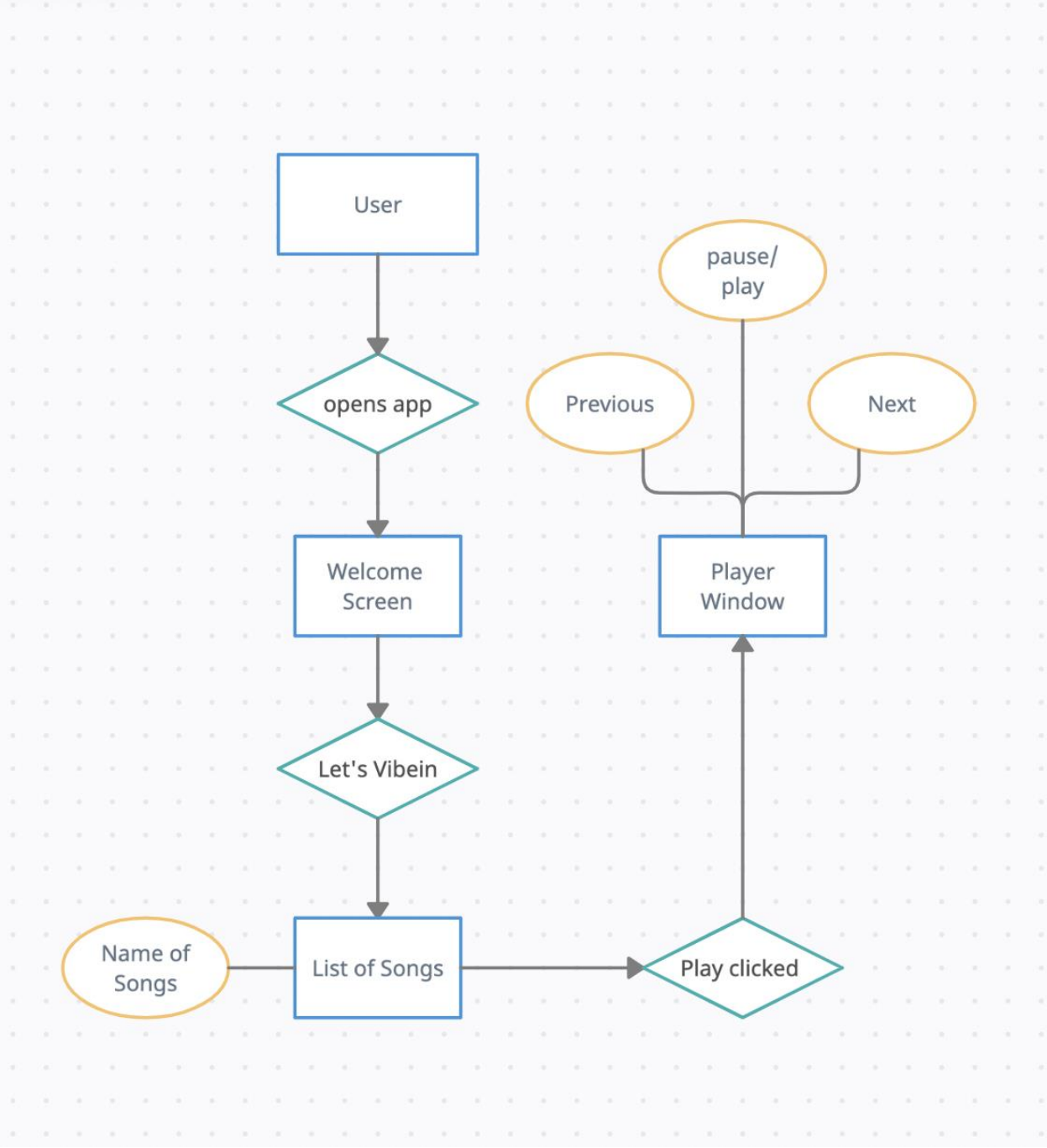
8 Conclusion

Through the development of music player on Android platform, we get a clear understanding of overall process of the system. The core part of the music player is mainly composed of main interface, file browsing and song listing, Grasping the development of the music player has had the preliminary scale small features. Music player system realized the basic function of player: play, pause, rewind and fast-forward a, volume adjustment is performed through the Android System Itself, play mode, song search, seek bar, this development implicated the popular mobile terminal development technology. This is the combination management of Java language in the open-source mobile platform based on Linux system configuration file. The system realized the music player programming. This design of music player based on Android system requires elaborate design of the music player framework, by adopting ANDROID STUDIO 2021.3.1 + Java language as technical support of this system, with the Android plug-in tools, and combination of Latest Android SDK version led to the comprehensive and smoothly design and development of the mobile terminalTime and money are one of the most important factors to any organization. Implementing such software in the college stationery department can surely be a profitable deal as this application helps to carry out tasks with ease and thereby reduces time and money on manpower and materials. This is an open-source application so that others can edit and transform this system application according to their needs.

9 References

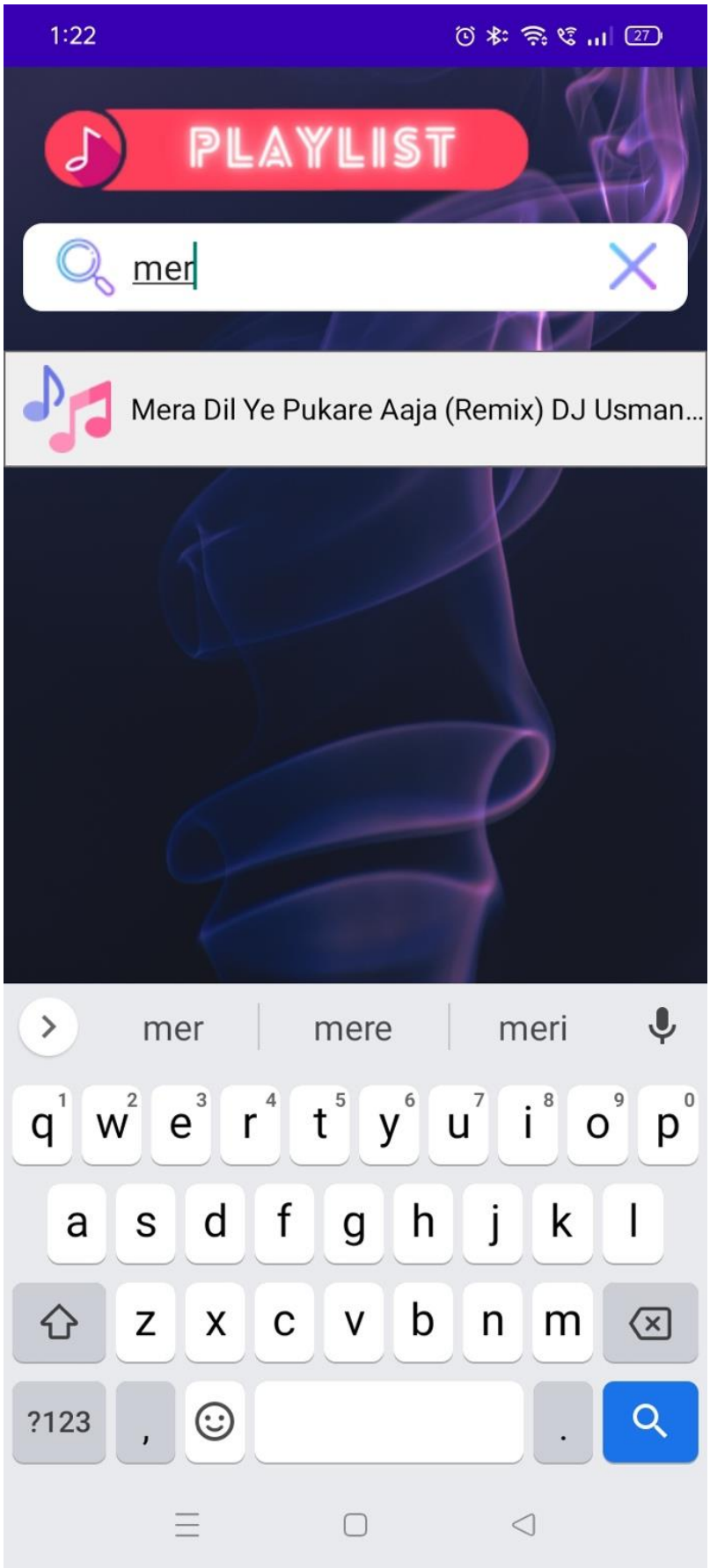
- Official Android Studio Docs address : https://developer.android.com/docs
- GeeksforGeeks address: https://www.geeksforgeeks.org/android-tutorial/
- Different Websites from Google.
- Existing music player apps

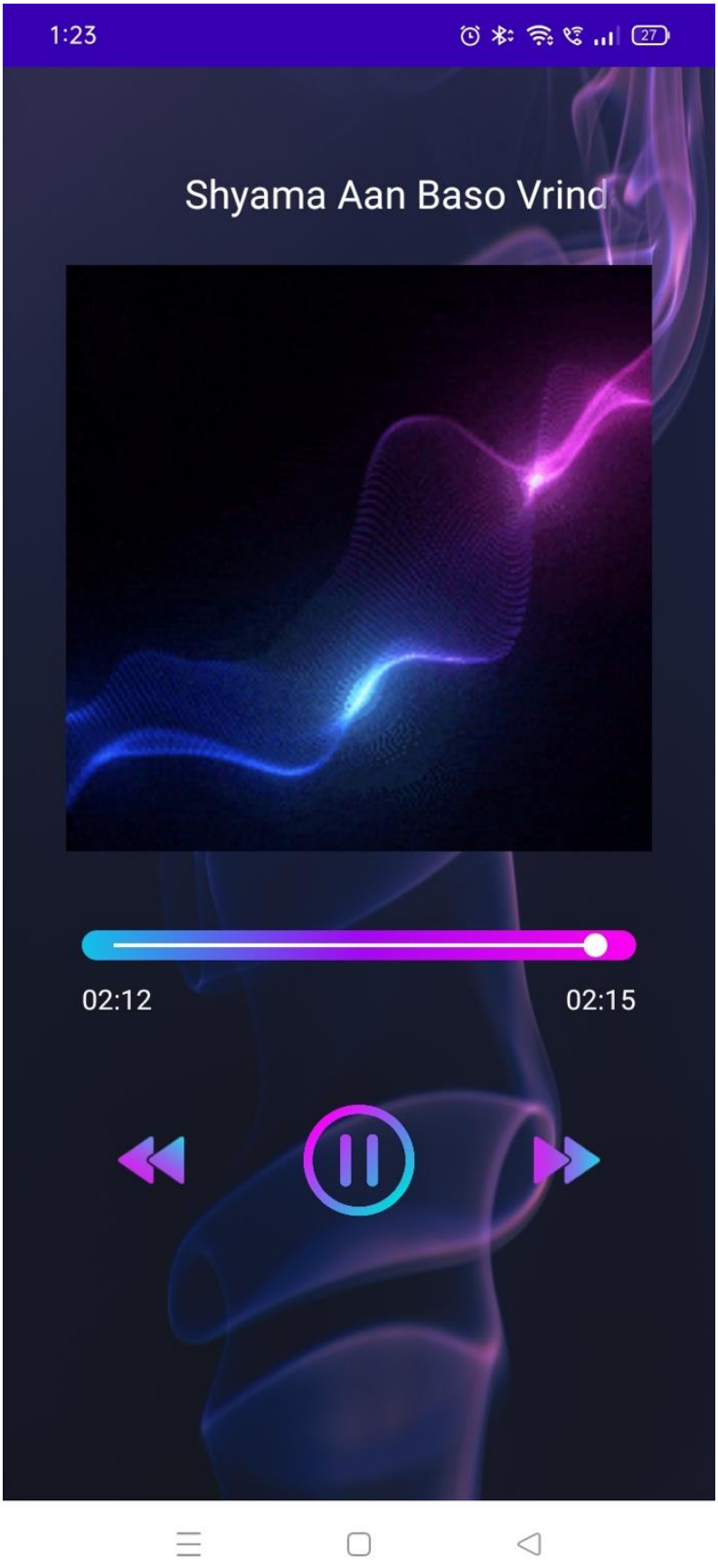
Annexure A
ER Diagram



Screenshots of Application


1:22
PLAYLIST
Search Here...
It's realme
Shyama Aan Baso Vrindavan Mein(PagalW...
Senorita
Ram Ram(PagalWorld.com.se)
2 Woofer(PagalWorldl)
Jaadugar(PagalWorld.com.se)
Mera Dil Ye Pukare Aaja (Remix) DJ Usman...
Piya Tu Aab To Aaja (Remix) DJ Shreya, SA...
Money(PaglaSongs)
How You Like That(PagalWorldl)


1:22
PLAYLIST
mer
Mera Dil Ye Pukare Aaja (Remix) DJ Usman...
mer
mere
meri


1:23
Shyama Aan Baso Vrind
02:12
02:15