Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Spring, Year: 2022), B.Sc. in CSE (Day)

PROJECT REPORT

Course Title: Mobile Application Development Lab
Course Code: CSE-402 Section: 191-DG

Lab Project Name: Create a Simple Music Player App in Android Studio.

Student Details

	Name	ID
1.	Khondokar Shad Bin Abed	191002390
2.	Md Abu Hasnat	191002133

Submission Date : 23/04/2022
Course Teacher's Name : Shifat Ara Rafiq

[For Teachers use only: Don't Write Anything inside this box]

Lab Project Status

Marks: …… **Signature: .....**

Comments: ..... **Date: .....**

Table of Contents

Chapter 1

Introduction

1.1 Introduction

Music is a part of every person's life. No matter what your mood is, you have a song to sustain that mood. If you wish to play your desired songs on Android devices, you need a music player. So, through this article, we will try to build our music player app using Android.

The music player app that we will develop in this article would allow the users to play the songs present on the device. You can download songs on your device and then use the music player to play those songs. Let's see the quite exciting features that you get along with this app.

Features of Music Player App:

Alluring user interface with loaded animations and music bars.

You can access all the songs present on your device.

Play your songs on the go without even having an internet connection.

Play or pause your music.

Listen to music even on your device background—no need to keep the app open all the time.

Music Visualizer that helps you to visualize the beats of a song.

1.2 Design Goals/Objective

The following things you will learn in this article:

- Implementing Media Player class and using its methods like pause, play and stop.
- Using external files (images, audios, etc.) in our project.
- Building the interface of our Music Player Android App.

Chapter 2

Design/Development/Implementation of the Project

2.1 Software Requirement

Implementation Tools:

- Android Studio.
- Android SDK.

2.2 Project Requirements

To build the music player applications, you need not be an expert but should be familiar with a few concepts. If you are good at designing layouts, you can make your music more alluring and attractive. Let's see the concepts that are essential to know to build the music player project.

i. Android Studio and its tools.

ii. Android XML Layout Designing.

iii. Android Activities and Lifecycle.

iv. Android Recycler View or List View.

v. Android Event Handler.

vi. Android Animations.

vii. Android Services.

viii. Java and Object-Oriented Programming.

Chapter 3
Performance Evaluation

3.1 Simulation Environment/ Simulation Procedure

Android Manifest: The AndroidManifest.xml file consists of all the required permissions and declarations essential for the music player app to run smoothly. Below are the permissions that we have kept in the music player application.

XML-Code:

```
<?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=".MainActivity"
   android:orientation="vertical"
   android:gravity="center_horizontal">

  <ImageView
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:src="@drawable/image"
     android:layout_marginTop="60dp"/>
  <SeekBar
      android:id="@+id/positionBar"
      android:layout_width="300dp"
      android:layout_height="wrap_content"
      android:layout_marginTop="30dp"/>
  <LinearLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content">
  <TextView
      android:id="@+id/elapsedTimeLabel"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="0.00"
      android:layout_marginLeft="40dp"/>
  <TextView
      android:id="@+id/remainingTimeLabel"
      android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="1.00"
        android:layout_marginLeft="240dp"/>



</LinearLayout>

    <Button
        android:id="@+id/playBtn"
        android:layout_width="30dp"
        android:layout_height="30dp"
        android:background="@drawable/play"
        android:layout_marginTop="40dp"
        android:onClick="playBtnClick"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_margin="40dp"
        android:gravity="center">
        <ImageView
            android:layout_width="18dp"
            android:layout_height="18dp"
            android:src="@drawable/sound"/>

        <SeekBar
            android:id="@+id/volumeBar"
            android:layout_width="300dp"
            android:layout_height="wrap_content"
            android:max="100"
            android:progress="50" />

        <ImageView
            android:layout_width="26dp"
            android:layout_height="26dp"
            android:src="@drawable/sound2"/>

    </LinearLayout>

    </LinearLayout>
```

- **Main Activity –** The MainActivity, as we know, is the parent activity for our music application that eventually loads up at the start of the application.
- **Android Resources –** Various Android resources are used in your layouts, apply distinct colors, and define static strings.

Java-Code:

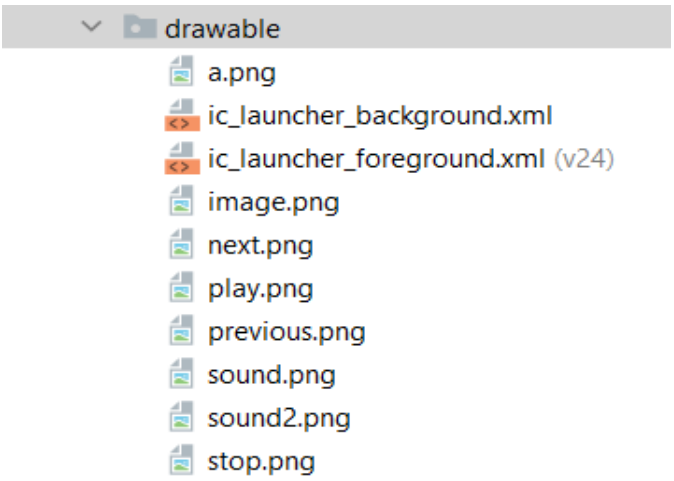
```
package com.example.musicplayer;

import androidx.appcompat.app.AppCompatActivity;

import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Message;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    Button playBtn;
    SeekBar positionBar;
    SeekBar volumeBar;
    TextView elapsedTimelabel;
    TextView remainingTimeLabel;
    MediaPlayer mp;
    int totalTime;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        playBtn=(Button) findViewById(R.id.playBtn);
        elapsedTimelabel=(TextView) findViewById(R.id.elapsedTimeLabel);
        remainingTimeLabel =(TextView) findViewById(R.id.remainingTimeLabel);

        mp =MediaPlayer.create(this,R.raw.d);
        mp.setLooping(true);
        mp.seekTo(0);
        mp.setVolume(0.5f,0.5f);
        totalTime=mp.getDuration();

        positionBar=(SeekBar) findViewById(R.id.positionBar);
        positionBar.setMax(totalTime);
        positionBar.setOnSeekBarChangeListener(
                new SeekBar.OnSeekBarChangeListener() {
                    @Override
                    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
                        if (fromUser) {
                            mp.seekTo(progress);
                            positionBar.setProgress(progress);
                        }
                    }
                    @Override
```

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

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        }
    );

    // Volume Bar
    volumeBar = findViewById(R.id.volumeBar);
    volumeBar.setOnSeekBarChangeListener(
        new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
                float volumeNum = progress / 100f;
                mp.setVolume(volumeNum, volumeNum);
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {
            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
            }
        }
    );

    // Thread (Update positionBar & timeLabel)
    new Thread(new Runnable() {
        @Override
        public void run() {
            while (mp != null) {
                try {
                    Message msg = new Message();
                    msg.what = mp.getCurrentPosition();
                    handler.sendMessage(msg);
                    Thread.sleep(1000);
                } catch (InterruptedException ignored) {}
            }
        }
    }).start();

}

private Handler handler = new Handler() {
    @Override
    public boolean handleMessage(Message msg) {
        int currentPosition = msg.what;
```

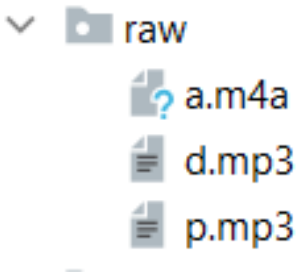
```
            // Update positionBar.
            positionBar.setProgress(currentPosition);

            // Update Labels.
            String elapsedTime = createTimeLabel(currentPosition);
            elapsedTimelabel.setText(elapsedTime);

            String remainingTime = "- " + createTimeLabel(totalTime - currentPosition);
            remainingTimeLabel.setText(remainingTime);

            return true;
        }
    };


    public String createTimeLabel(int time) {
        String timeLabel = "";
        int min = time / 1000 / 60;
        int sec = time / 1000 % 60;

        timeLabel = min + ":";
        if (sec < 10) timeLabel += "0";
        timeLabel += sec;

        return timeLabel;
    }

    public void playBtnClick(View view) {

        if (!mp.isPlaying()) {
            // Stopping
            mp.start();
            playBtn.setBackgroundResource(R.drawable.stop);

        } else {
            // Playing
            mp.pause();
            playBtn.setBackgroundResource(R.drawable.play);
        }
    }
}
```

3.2 Results and Discussions

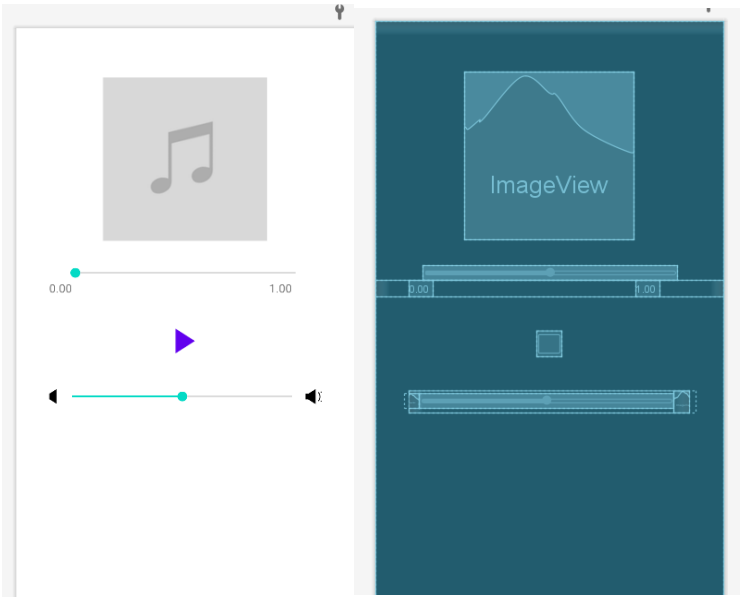
1st add picture in a file (drawable) for music application



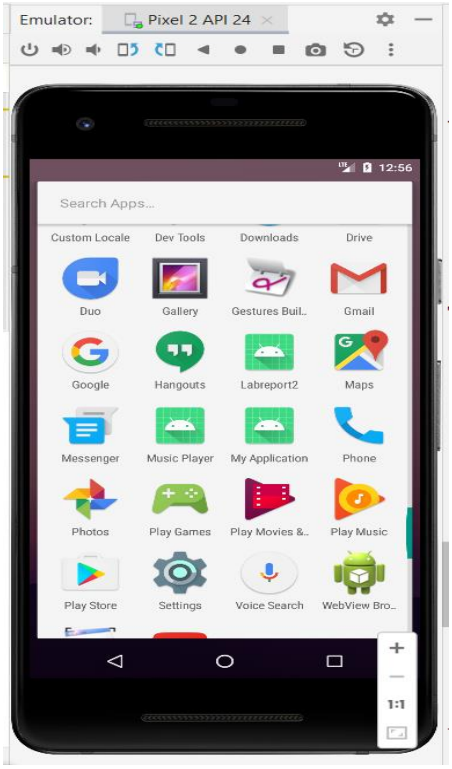
2nd add music in a file (raw) for music application

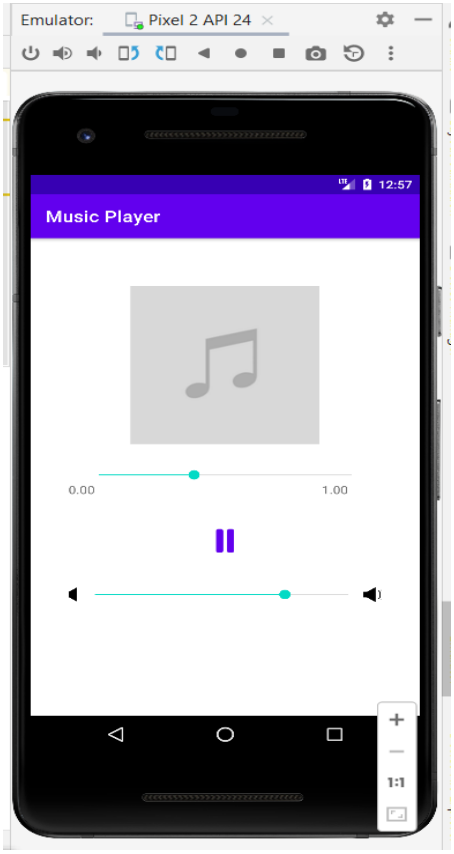


XML design



Music Play runs output:





Chapter 4

Conclusion

4.1 Introduction

From this project, you came to know about how a music player is built using Android Studio. In the objective of doing so, you got to know the features of the music player, its flow, and the requirements to build it. Further, you saw the things you need to know to get started with the development of the music player. Finally, you saw the actual implementation of the music player along with its source code.

References

1. https://www.youtube.com/watch?v=zCYQBIcePaw
2. https://techvidvan.com/tutorials/android-music-player-app-project/