Name: Bhavik Ransubhe
Class : TE(B) COMP
ROLL NO:39055

Android: Design a mobile app for media player.

Aim: Design a mobile app for media player.

Learning Objective:

To study and implementation of mobile application for media player

Software Requiremenrts:

- Android Studio
- Windows 10 64 bit
- Java JDK1.7 or later version

Hardware Requirements:

PC/Laptop with min 4GB RAM , 500GB HDD, Intelcore i5 Processor.

Theory:

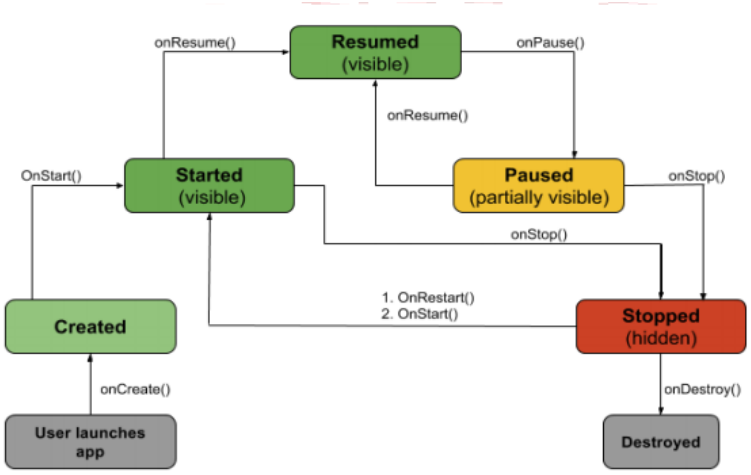
Actually, the first thing you do is create an activity. These are where all the action happens, because they are the screens that allow the user to interact with your app. In short, activities are one of the basic building blocks of an Android application. In this introduction to Android activities tutorial, you'll dive into how to work with activities. You'll work through creating a to-do list app named Forget Me Not and along the way learn:

- The process for creating, starting and stopping an activity and handle navigation between activities.
- The various stages in the lifecycle of an activity and how to handle each stage gracefully.
- The way to manage configuration changes and persist data within your activity.

Activity Lifecycle:

Before firing up your fingers, indulge yourself in a bit of theory. As mentioned earlier, activities are the foundations upon which you build screens for your app. They encompass multiple components the user can interact with, and it's likely that your app will have multiple activities to handle the various screens you create.

Having all these activities doing different things makes it sound like developing an Android app is a complicated undertaking. Fortunately, Android handles this with specific callback methods that initiate code only when it's needed. This system is known as the activity lifecycle. Handling the various lifecycle stages of your activities is crucial to creating a robust and reliable app. The lifecycle of an activity is best illustrated as a step pyramid of different stages linked by the core callback methods:



Following the diagram above, you can picture the lifecycle in action as it courses through your code. Take a closer look at each of the callbacks:

• onCreate(): When you first create an activity, you also call this method. This also where you initialize any UI elements or data objects. You also have the savedInstanceState of the activity that contains its previously saved state, and you can use it to recreate that state.

• onStart(): Just before presenting the user with an activity, this method is called. It's always followed by onResume() and very rarely by onStop(). In here, you generally should start UI animations, audio based content or anything else that requires the activity's contents to be on screen.

• onResume(): Before bringing an activity back to the foreground, you call this method. Here you have a good place to restart animations, update UI elements, restart camera previews, resume audio/video playback or initialize any components that you release during onPause().

• onPause(): Before sliding into the background, this method is called. Here you should stop any visuals or audio associated with the activity such as UI animations, music playback or the camera. This method is followed by onResume() if the activity returns to the foreground or by onStop() if it becomes hidden.

• onStop(): You call this method right after onPause(), but before the activity goes into the background, and it's a good place to save data that you want to commit to the disk. It's followed by either onRestart(), if this activity is coming back to the foreground, or onDestroy() if it's being released from memory.

• onRestart(): After stopping an activity, but just before starting it again, you call this method. It's always followed by onStart().

• onDestroy(): This is the final callback you'll receive from the system before the activity is destroyed. A call to finish() is one way to destroy the activity, or it can be triggered by the system when it needs to recoup memory. If your activity includes any background threads or other long-running resources, destruction could lead to a memory leak if they're not released, so you need to remember stop these processes here as well.

So many methods to remember! In the next section, you'll see some of these lifecycle methods in action, and then it'll be a lot easier to remember what everything does.

Creating an Activity:

Keeping the activity lifecycle in mind, take a look at an activity in the sample project. Open MainActivity, you'll see that onCreate. Here's a play-by-play of what's happening above:

- You call onCreate() on the superclass; remember that this is always the first thing you should do in a callback method.
- You tell the WindowManager to make your activity's window full screen.
- You set the content view of your activity with the corresponding layout file resource.
- Here you initialize all the UI and data variables. In this case, you're using a TextView to show the current date and time, a button to add tasks to your list, a ListView to display your list, and an ArrayList to hold your data. You can find the implementation of all these UI elements in the activity_main.xml file.
- Here you initialize and set the Adapter that will handle the data for your ListView.
- You set an OnItemClickListener() for the ListView to capture the user's tap on individual list entries.

Starting an Activity:

In its current stat, the app is a fairly useless lump of ones and zeros because you can't add anything to the to-do list. You have the power to change that, and that's exactly what you'll do next. In the MainActivity file you have open, add a field to the top of the class: private final int ADD_TASK_REQUEST = 1; You'll use this variable to reference your request to add new tasks later on. You can assign any int value to ADD_TASK_REQUEST. If it returns the value you entered on the request, you can assume your request was handled successfully. More on this later. Then add the following implementation for addTaskClicked: Intent intent = new Intent(MainActivity.this, TaskDescriptionActivity.class); startActivityForResult(intent, ADD_TASK_REQUEST); When the user taps the add a task button, you call addTaskClicked. Here you create an Intent to launch the TaskDescriptionActivity from MainActivity. It needs to produce a result from TaskDescriptionActivity, because the app needs to know if there is a new task to add to your list or not. To accomplish this, you start the activity using startActivityForResult(...). When the TaskDescriptionActivity finishes, it returns a result in an intent to the onActivityResult(...) method. Implement this callback to receive the result by adding the following method to the bottom of MainActivity: All appears to be well and good, but is it really? Not quite. Forget Me Not needs to be able to move to another activity and extract task descriptions from it, so the next thing you'll do is make one.

ADD an Activity:

Android Studio makes it very easy to create an activity. All you need do is right-click on the package where you want to add the activity; in this case, the package is com.raywenderlich.todolist). Then navigate to New\Activity, and the choose Empty Activity, which is a basic template for an activity. On the next screen, you enter the Activity Name and Android Studio will automatically fill the other fields based on that. Enter the activity name as TaskDescriptionActivity. Android Studio will automatically generate the corresponding resources needed to create the activity. These are:

• Class: The class file is named TaskDescriptionActivity.java and located in your Java package. This is where you implement the activity's behavior. This class must subclass the Activity class or an existing subclass of it.

• Layout: The layout file is located under res/layout and named activity_task_description.xml. It defines the placement of different UI elements on the screen when the activity is created.

CODE:

FILE 1: `MainActivity.java :-`

```
package com.br14x.mediaplayer;

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.PermissionDeniedResponse;
import com.karumi.dexter.listener.PermissionGrantedResponse;
import com.karumi.dexter.listener.PermissionRequest;
import com.karumi.dexter.listener.single.PermissionListener;

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

public class MainActivity extends AppCompatActivity {

    ListView mylistview;
    String[] items;


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

        mylistview = findViewById(R.id.mylistview);
        runtimePermission();


    }


    // Get read permission from user.
    private void runtimePermission(){
```

```
        Dexter.withContext(this)
                .withPermission(Manifest.permission.READ_EXTERNAL_STORAGE)
                .withListener(new PermissionListener() {
                    @Override
                    public void onPermissionGranted(PermissionGrantedResponse
permissionGrantedResponse) {
                        displayInListView(); // display song list, when permission
granted.
                    }

                    @Override
                    public void onPermissionDenied(PermissionDeniedResponse
permissionDeniedResponse) {
                        Toast.makeText(MainActivity.this, "Please grant
permission", Toast.LENGTH_SHORT).show();
                    }

                    @Override
                    public void
onPermissionRationaleShouldBeShown(PermissionRequest permissionRequest,
PermissionToken permissionToken) {
                        permissionToken.continuePermissionRequest(); // if user
denied permission.
                    }
                }).check();
    }


    // Read songs from device
    private ArrayList<File> findSong(File file) {

        ArrayList<File> arrayList = new ArrayList<>();

        File[] files = file.listFiles(); // create array object of File

        for(File singleFile : files){
            if(singleFile.isDirectory() && !singleFile.isHidden()){
                arrayList.addAll(findSong(singleFile));
            }
            else{
                if(singleFile.getName().endsWith(".mp3") ||
singleFile.getName().endsWith(".wav")){
                    arrayList.add(singleFile);
                }
            }
        }
        return arrayList;
    }

    // After reading, display song in list view
    private void displayInListView() {

        final ArrayList<File> mysong =
findSong(Environment.getExternalStorageDirectory()); // read songs by findSong
method.

        items = new String[mysong.size()];  // declare size
```

```
        for(int i=0;i<mysong.size();i++){
            items[i] =
mysong.get(i).getName().toString().replace(".mp3","").replace(".wav","");
        }

        ArrayAdapter<String> myAdapter = new
ArrayAdapter<>(this,android.R.layout.simple_list_item_1, items);
        mylistview.setAdapter(myAdapter); // plugin adapter with list view.



        // Perform action on item click
        mylistview.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view, int
position, long l) {
                String songName =
mylistview.getItemAtPosition(position).toString();

                startActivity(new Intent(MainActivity.this, PlayerActivity.class)
                        .putExtra("songs", mysong).putExtra("songname",songName)
                        .putExtra("pos",position));
                // Pass selected (song) item to another activity and also position
of song in list.
            }
        });

    }

}
```

FILE2:activity_main.xml:-

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#E17AF3"
    android:padding="10dp"
    tools:context=".MainActivity">

    <ListView
        android:id="@+id/mylistview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:cacheColorHint="#ED1A1A"
        android:listSelector="#F44336"
        tools:layout_editor_absoluteX="10dp"
        tools:layout_editor_absoluteY="10dp" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

FILE3: PlayerActivity.java:-

```
package com.br14x.mediaplayer;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.PorterDuff;
import android.media.MediaPlayer;
import android.net.Uri;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;

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

public class PlayerActivity extends AppCompatActivity {

    Button btnNext, btnPrevious, btnPause;
    TextView songLabel;
    SeekBar songSeekbar;

    static MediaPlayer myMediaPlayer;
    int position;

    String sname;

    ArrayList<File> mySongs;
    Thread updateSeekBar;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_player);
        getSupportActionBar().setTitle("Now Playing");
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);

        btnNext = (Button)findViewById(R.id.next);
        btnPrevious = (Button)findViewById(R.id.previous);
        btnPause = (Button)findViewById(R.id.pause);
        songLabel = (TextView)findViewById(R.id.songLabel);
        songSeekbar = (SeekBar)findViewById(R.id.seekBar);


        // update seekbar
        updateSeekBar = new Thread(){
            @Override
            public void run() {
                int totalDuration = myMediaPlayer.getDuration();  // know selected
song duration.
                int currentPosition = 0;

                while(currentPosition < totalDuration){
```

```
                    try {
                        sleep(500); // step of seek bar
                        currentPosition = myMediaPlayer.getCurrentPosition();
                        songSeekbar.setProgress(currentPosition);
                    }catch (InterruptedException e){
                        e.printStackTrace();
                    }
                }



            }
        };

        if(myMediaPlayer != null){
            myMediaPlayer.stop();
            myMediaPlayer.release();
        }

        Intent intent = getIntent();
        Bundle bundle = intent.getExtras();
        assert bundle != null;
        mySongs = (ArrayList) bundle.getParcelableArrayList("songs");

        sname = mySongs.get(position).getName().toString();

        String songName = intent.getStringExtra("songname");

        songLabel.setText(songName); // display song title
        songLabel.setSelected(true);

        position = bundle.getInt("pos", 0);

        Uri u = Uri.parse(mySongs.get(position).toString());

        myMediaPlayer = MediaPlayer.create(getApplicationContext(), u);
        myMediaPlayer.start();
        songSeekbar.setMax(myMediaPlayer.getDuration());

        updateSeekBar.start();  // increasing seekBar(made seekBar movable, when
song play) <------

        // change color of seek-bar

songSeekbar.getProgressDrawable().setColorFilter(getResources().getColor(R.color.c
olorPrimary), PorterDuff.Mode.MULTIPLY);

songSeekbar.getThumb().setColorFilter(getResources().getColor(R.color.colorPink),
PorterDuff.Mode.SRC_IN);


        // After playing song, Now apply listener on seekBar
        songSeekbar.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int i, boolean b) {
            }
```

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

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                myMediaPlayer.seekTo(seekBar.getProgress()); // apply change on
duration, when seek bar move.
            }
        });


        // At last, Apply listener on pause,next and previous button.
        btnPause.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                songSeekbar.setMax(myMediaPlayer.getDuration());

                if(myMediaPlayer.isPlaying()){
                    btnPause.setBackgroundResource(R.drawable.icon_play);
                    myMediaPlayer.pause();
                }else{
                    btnPause.setBackgroundResource(R.drawable.icon_pause);
                    myMediaPlayer.start();
                }
            }
        });

        btnNext.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                myMediaPlayer.stop(); // first stop the music
                myMediaPlayer.release();
                // then, increase position of song
                position = (position+1) % mySongs.size(); // also consider
boundary cases.

                Uri u = Uri.parse(mySongs.get(position).toString());
                myMediaPlayer = MediaPlayer.create(getApplicationContext(), u);
                sname = mySongs.get(position).getName().toString();
                songLabel.setText(sname);

                myMediaPlayer.start(); // then, start song
            }
        });

        btnPrevious.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                myMediaPlayer.stop(); // first stop the music
                myMediaPlayer.release();
                // then, decrease position of song
                position = ((position-1) < 0) ? (mySongs.size()-1):position-1; //
also consider boundary case less than 0.

                Uri u = Uri.parse(mySongs.get(position).toString());
                myMediaPlayer = MediaPlayer.create(getApplicationContext(), u);
                sname = mySongs.get(position).getName().toString();
                songLabel.setText(sname);
```

```
                myMediaPlayer.start(); // then, start song
            }
        });

    }


    // Perform action on go back
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if(item.getItemId() == android.R.id.home){
            onBackPressed();
        }

        return super.onOptionsItemSelected(item);
    }
}
```

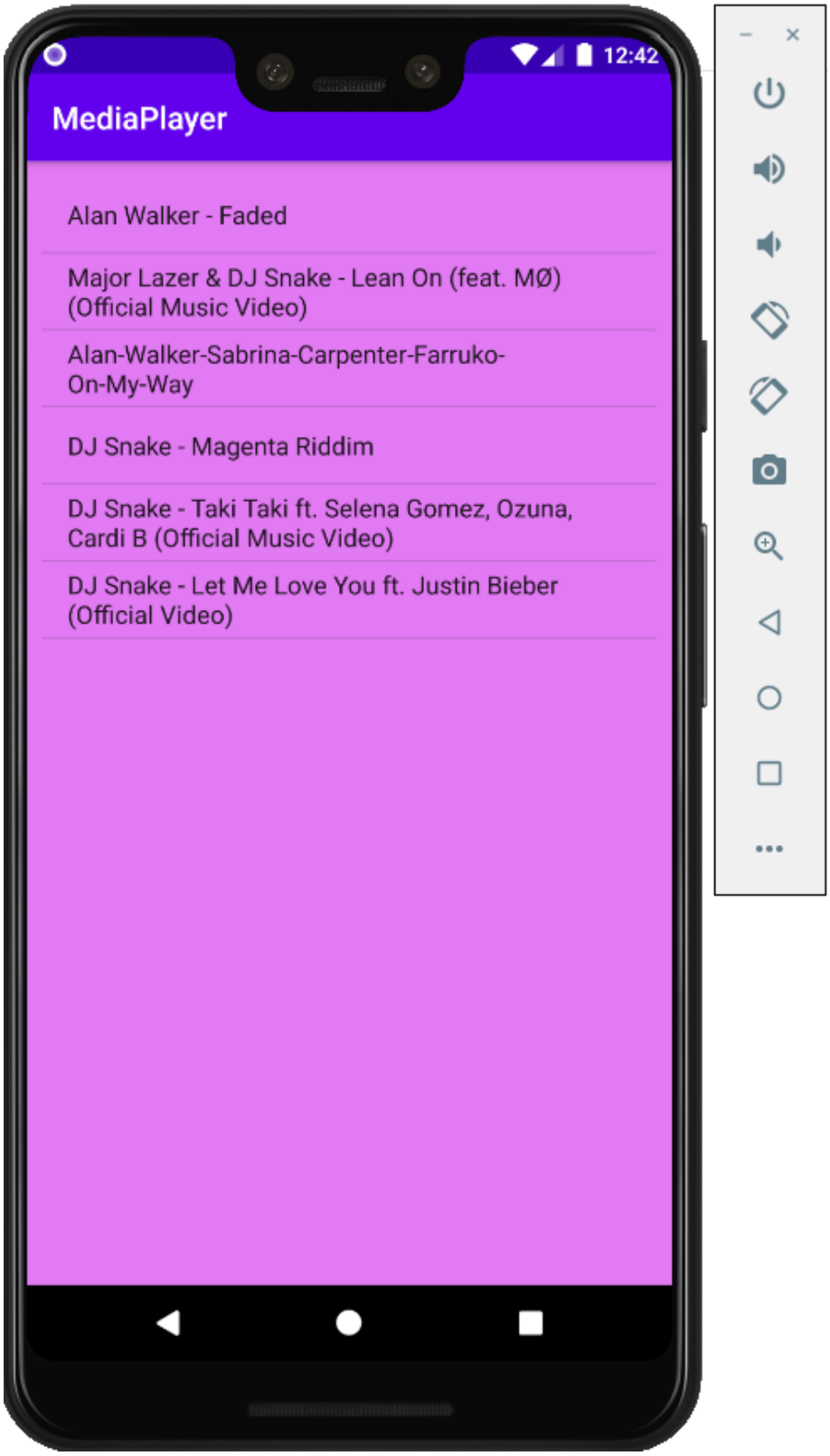
FILE4:activity_Player.xml:-

```
<?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:background="#20193E"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:weightSum="10"
    android:orientation="vertical"
    tools:context=".PlayerActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="vertical"
        android:gravity="center"
        android:layout_weight="7">

        <ImageView
            android:layout_width="260dp"
            android:layout_height="260dp"
            android:src="@drawable/coverimage"/>

        <TextView
            android:id="@+id/songLabel"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="20dp"
            android:ellipsize="marquee"
            android:marqueeRepeatLimit="marquee_forever"
            android:scrollHorizontally="true"

            android:singleLine="true"
            android:text="Song Name"
            android:textAlignment="center"
```

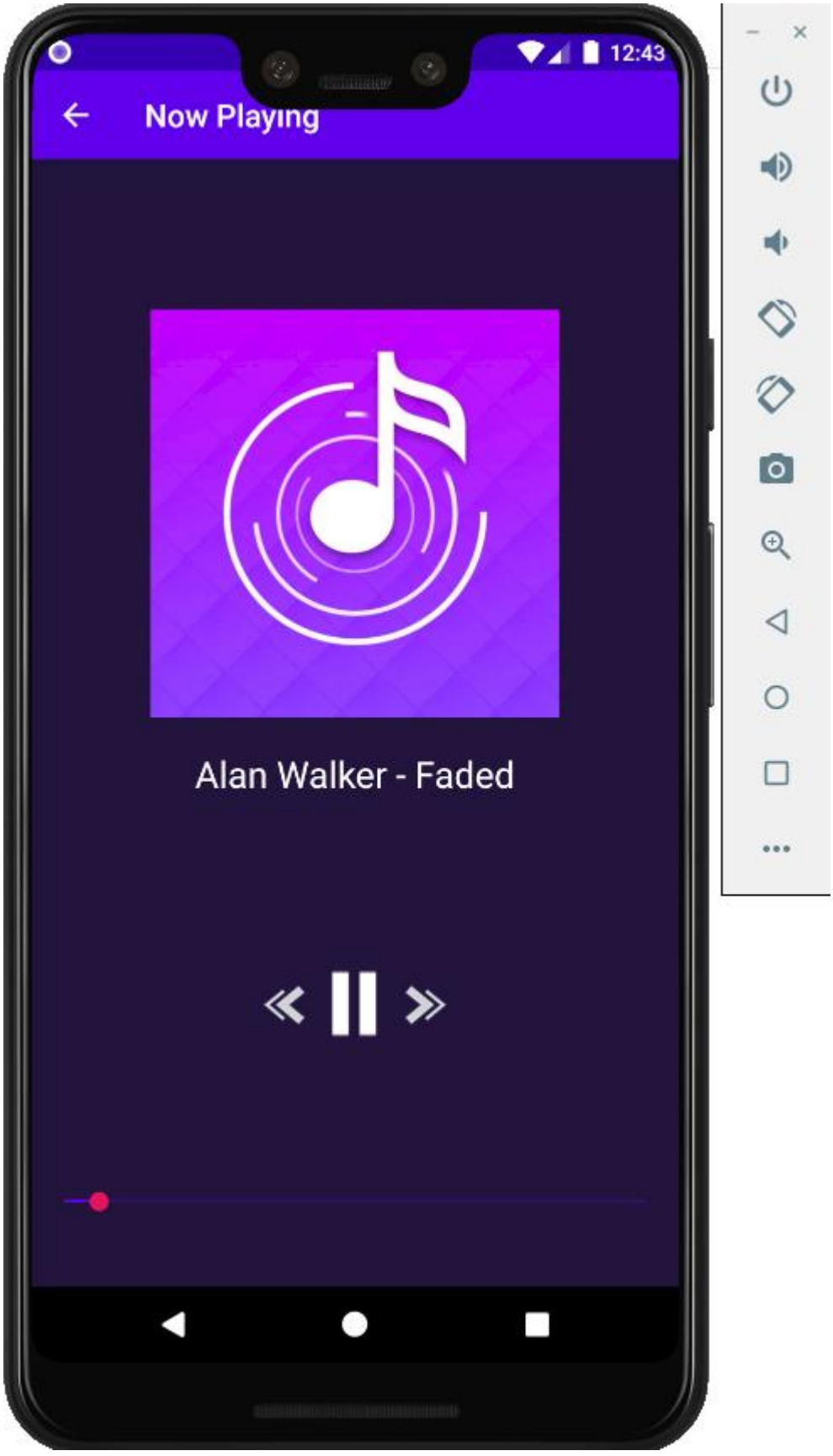
```
            android:textColor="#FFFEFF"
            android:textSize="23sp" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="3">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:padding="5dp">

            <SeekBar
                android:id="@+id/seekBar"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_alignParentBottom="true"
                android:layout_marginBottom="40dp"/>

            <Button
                android:id="@+id/pause"
                android:layout_width="52dp"
                android:layout_height="50dp"
                android:layout_centerHorizontal="true"
                android:background="@drawable/icon_pause"
                android:layout_marginTop="5dp"/>

            <Button
                android:id="@+id/next"
                android:layout_width="38dp"
                android:layout_height="38dp"
                android:layout_toRightOf="@+id/pause"
                android:layout_marginTop="12dp"
                android:background="@drawable/icon_next"/>


            <Button
                android:id="@+id/previous"
                android:layout_width="38dp"
                android:layout_height="38dp"
                android:layout_toLeftOf="@+id/pause"
                android:layout_marginTop="12dp"
                android:background="@drawable/icon_previous"/>




        </RelativeLayout>


    </LinearLayout>


</LinearLayout>
```

OUTPUT:-

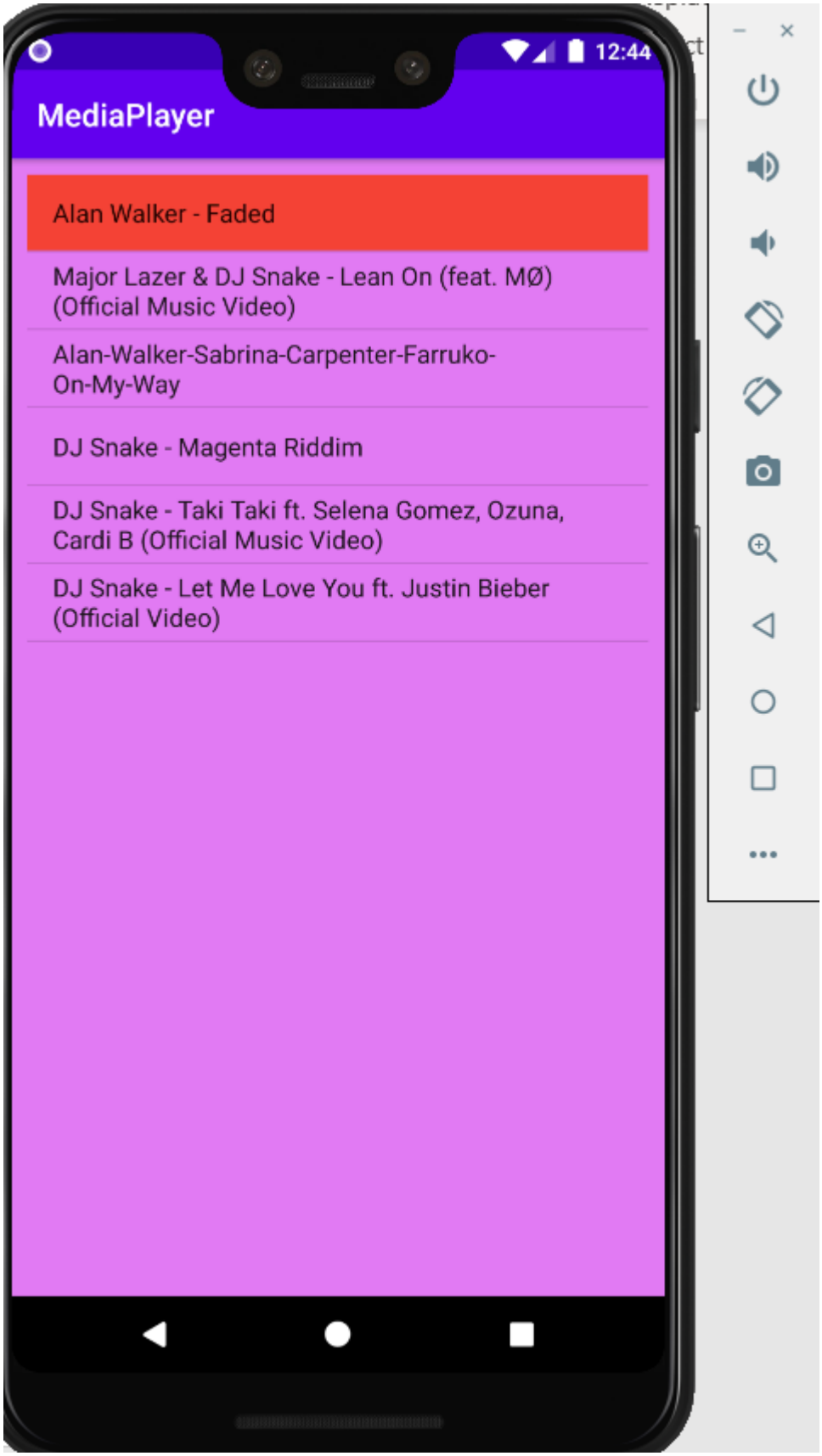
1)


12:42
MediaPlayer
Alan Walker - Faded
Major Lazer & DJ Snake - Lean On (feat. MØ) (Official Music Video)
Alan-Walker-Sabrina-Carpenter-Farruko-On-My-Way
DJ Snake - Magenta Riddim
DJ Snake - Taki Taki ft. Selena Gomez, Ozuna, Cardi B (Official Music Video)
DJ Snake - Let Me Love You ft. Justin Bieber (Official Video)

2)



3)Ongoing Music Highlighted :


12:44
MediaPlayer
Alan Walker - Faded
Major Lazer & DJ Snake - Lean On (feat. MØ) (Official Music Video)
Alan-Walker-Sabrina-Carpenter-Farruko-On-My-Way
DJ Snake - Magenta Riddim
DJ Snake - Taki Taki ft. Selena Gomez, Ozuna, Cardi B (Official Music Video)
DJ Snake - Let Me Love You ft. Justin Bieber (Official Video)