CONTENTS

Main Report

INTRODUCTION TO MOBILE APPLICATION DEVELOPMENT

Every day the new devices are incoming to the market with innovative options thanks to growing technology. The evolution of Mobile Application Development technology with new devices made our lives much easier.

In the smartphone world, simply having a running web site is not enough. Regarding a recent study, it has shown that about 45% and more of Google search happens using smartphones.

The number is spectacular and there is a growth within the mobile business. Being obtainable on an internet-enabled device is needed for every and each business which has given the kicking start to mobile application development

What is Mobile Development?

Mobile development, which is not about building phone apps, though it is a huge part of it. Actually, it's doing any reasonably development for any kind of mobile devices such as developing apps for phones, tablets, smartwatches, and every form of wearable devices that run any kind of mobile operating system.

Mobile development presents a reasonably distinctive chance for a oneperson development team to build an actual, usable, significant app end-to-end during a comparatively short period. However, Mobile Apps Development represents more than just a chance for the solo-developer to create their own project as it is arguably the longer term of development, as mobile devices are getting larger and bigger parts of our lives.

Android is the dominant player in mobile development platforms space, it was a bit later participant to the game, first being released in Sept 2008, virtually a year later than iOS but it has managed to achieve a reasonably massive share of the mobile market.

Technically, Android is the mobile OS with the largest most dominant share of the market with around 80% share compared to iOS's 18 % share. Those numbers are a bit deceiving since android may be a fragmented market consisting of the many different devices created by different manufacturers, running completely different versions of the Android OS.

Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine Web Kit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

Android Libraries

- This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the application framework libraries in addition to those that facilitate user interface building, graphics drawing and database access. A summary of some key core Android libraries available to the Android developer is as follows –

- **android.app** − Provides access to the application model and is the cornerstone of all Android applications.
- **android.content** − Facilitates content access, publishing and messaging between applications and application components.
- **android.database** − Used to access data published by content providers and includes SQLite database management classes.
- **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android.text** − Used to render and manipulate text on a device display.
- **android.view** − The fundamental building blocks of application user interfaces.

- **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications. Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++ based libraries contained in this layer of the Android software stack.

Android Runtime

- This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android.

- The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

- The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language

INTRODUCTION TO PROJECT:-

This project is associated with any **Android Media Player**. The Project is as per the required curriculums of the **B.E**

Android Introduction…

- Android is an operating system based on the Linux kernel, and designed primarily for touch screen mobile devices such as Smartphone's and tablet computers.
- Android allows users to customize their home screens with shortcuts to applications and widgets, which allow users to display live content, such as emails and weather information, directly on the home screen.

Purpose…

Explains the functional features, design…

Scope…

This application can run anonymously in any Android based Smart- phones, not less than version 2.3.5

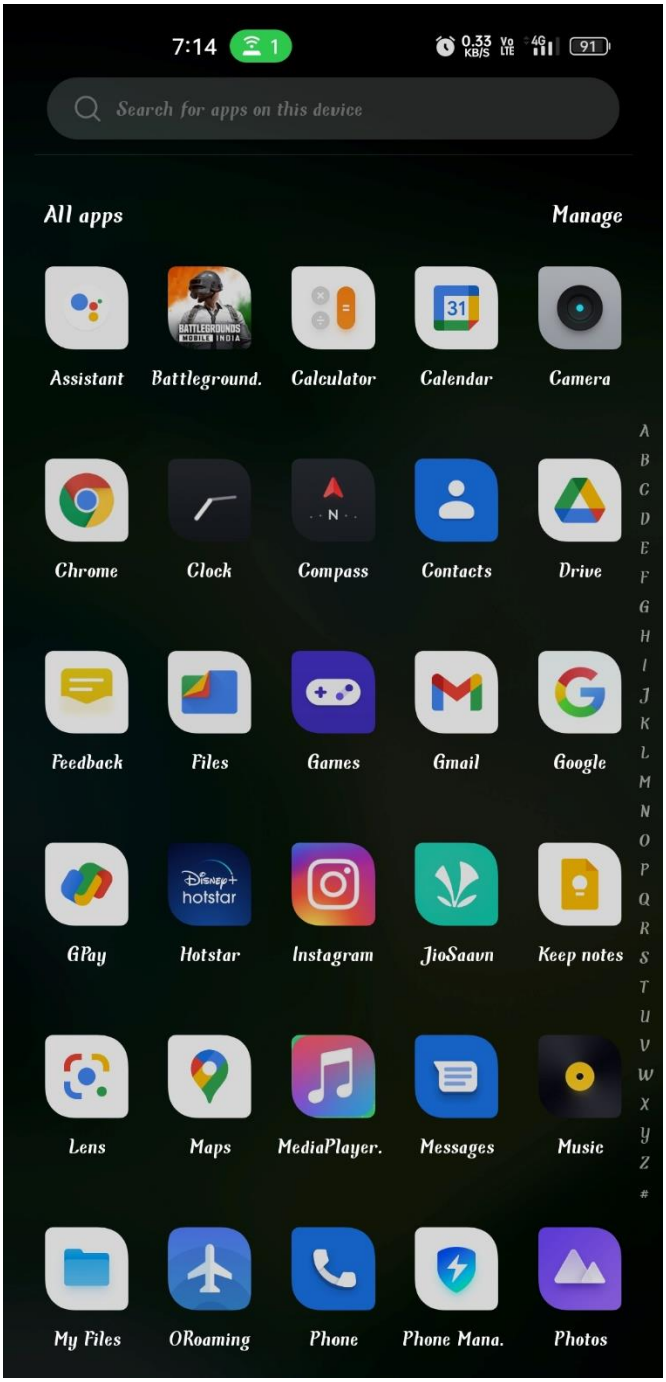
Objectives…

- Application will be written using Android SDK in Java and should run on all Android OS handsets.
- The application will play audio files with format of MP3, AAC, 3GP, M4A, MIDI, RTX, OGG, and WAV.
- The application will play video files with format of 3GP, MP4, WEBM.
- Background playing options.
- Notification on the home screen.
- Android is open source and Google releases the code under the Apache License.
- Android has a large community of developers writing applications ("apps") that extend the functionality of devices, written primarily in the Java programming language
- Android is the world's most widely used smart phone platform,[overtaking Symbian in the fourth quarter of 2010. Android is popular with technology companies who require a ready-made, low- cost, customizable and lightweight operating system for high tech devices.
- Despite being primarily designed for phones and tablets, it also has been used in

televisions, games consoles, digital cameras and other electronics.

- The user interface of Android is based on direct manipulation, using touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching and reverse pinching to manipulate on- screen objects
- Android home screens are typically made up of app icons and widgets; app icons launch the associated app, whereas widgets display live, auto updating content such as the weather forecast, the user's email inbox, or a news ticker directly on the home screen.

Interface of Android .....



Android Versions

- Android 1.0 (API level 1)
- Android 1.1 (API level 2)
- Android 1.5 Cupcake (API level 3)
- Android 1.6 Donut (API level 4)
- Android 2.0 Éclair (API level 5)
- Android 2.1 Éclair (API level 7)
- Android 2.2–2.2.3 Froyo (API level 8)
- Android 2.3–2.3.2 Gingerbread (API level 9)
- Android 2.3.3–2.3.7 Gingerbread (API level 10)
- Android 3.0 Honeycomb (API level 11)
- Android 3.1 Honeycomb (API level 12)
- Android 3.2 Honeycomb (API level 13)
- Android 4.0–4.0.2 Ice Cream Sandwich (API level 14)
- Android 4.0.3–4.0.4 Ice Cream Sandwich (API level 15)
- Android 4.1 Jelly Bean (API level 16)
- Android 4.2 Jelly Bean (API level 17)
- Android 4.3 Jelly Bean (API level 18)
- Android 4.4 KitKat (API level 19)

Symbols of android

Media Player

Introduction to Media player

Media Player is android application that can play various audio and video files. To make use of android OS with more public interest and make it more users friendly so all can use it. This project is to design and implement platform independent media player which can play most of the audio files like .mp3, .wav etc. and some video files in addition to view images.

Objectives…

- Application will be written using Android SDK in Java and should run on all Android OS handsets.
- The application will play audio files with format of MP3, AAC, 3GP, M4A, MIDI, RTX, OGG, and WAV.
- The application will play video files with format of 3GP, MP4, WEBM.
- Background playing options.

Functional requirements…

- Android operating system on the Smartphone.
- The target device should be sound enabled.
- Ability to play Audio file
- Playlist Screen
- Main Screen
- Player Screen

Playlist menu…

- Play
- Stop
- Pause
- Songs list
- Next
- Previous

Extrnal interface requirments

- User Interface Tested on Android Version 12
- Any device which have Android platform will be accesed

Snapshots :-

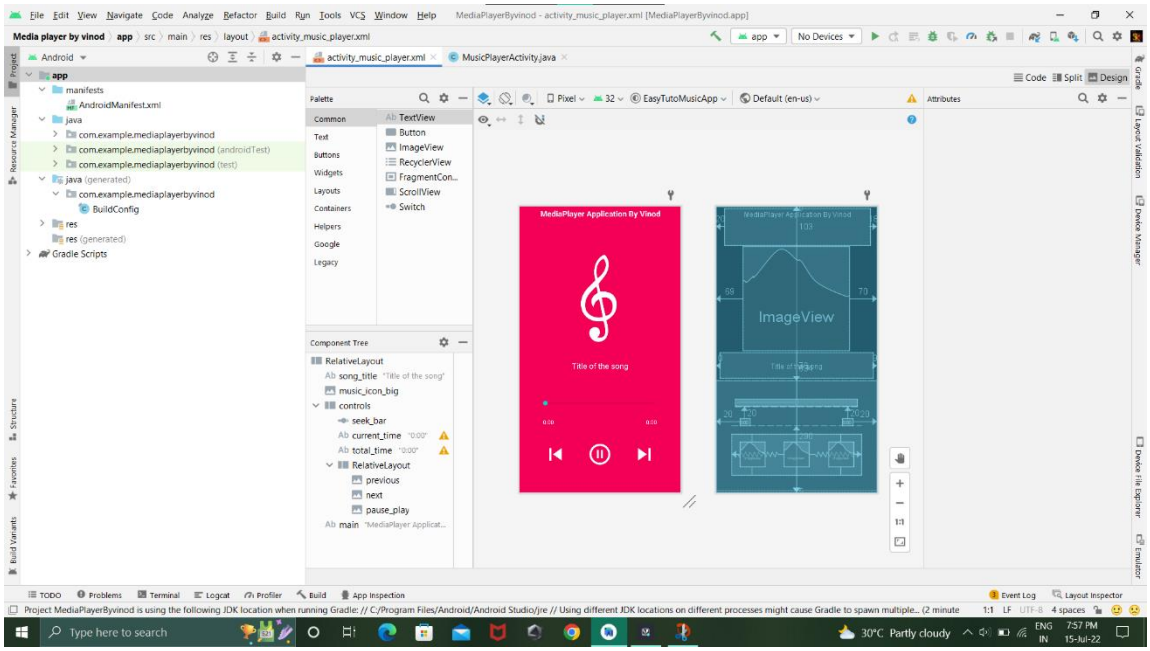


Fig-1.1 : Frontend Design

- **For designing of this application the foregroung design and Blueprint is very important that is shown in the fig 1.1**

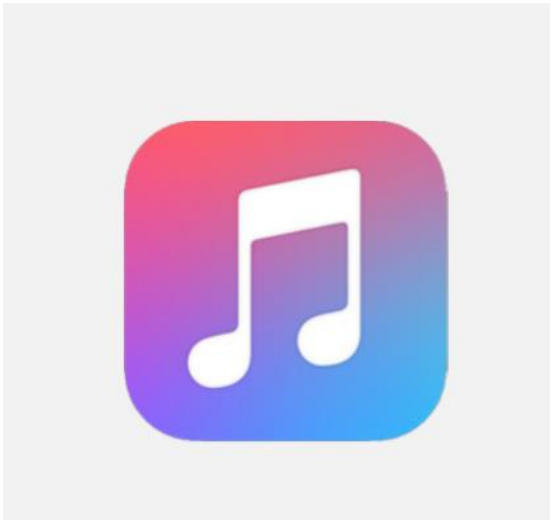
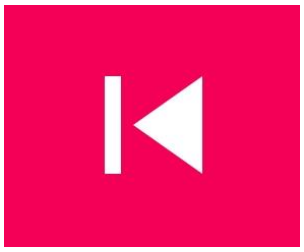


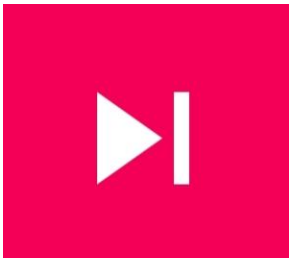
Fig -1.2 : Application Interface

- **This is the icon of the music player which is set by the user accordingly**

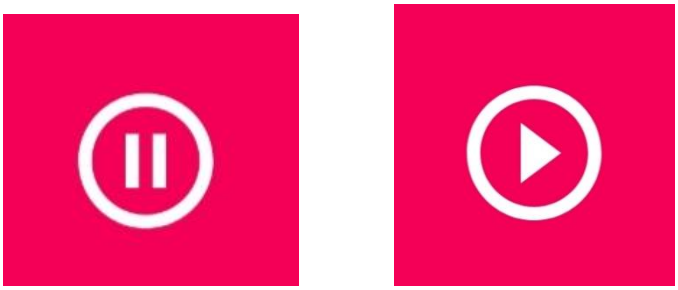
MUSIC BUTTONS :-



- **The Button shown above is used for the play the previous song played**



- **The above Button is used for the play the next song in the queue**



- **The above Button which are show here is used for pause and play the songs**

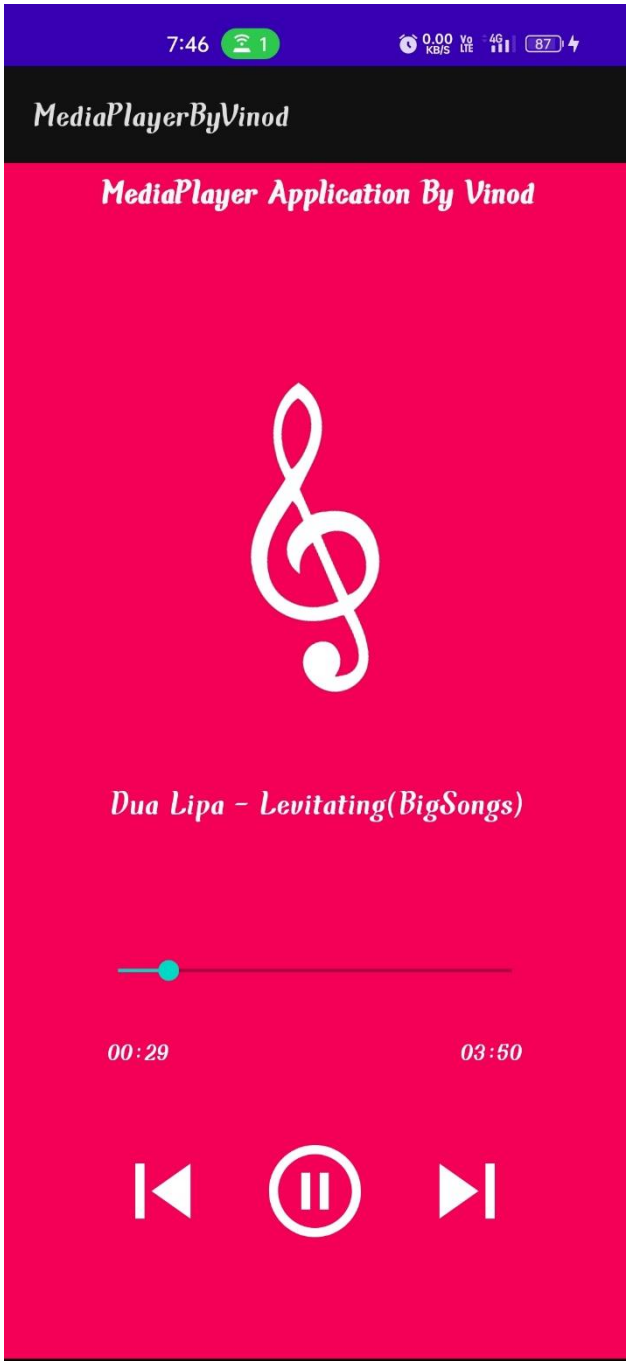
- **This is the seekbar where the start and end of the song will be visible if we want to forward press on seekbar then go to a particular time**

Fig – 1.3 : Media Buttons

Fig – 1.4 :Playlist view

Fig- 1.5 : Playlist while playing

- **These are the playlist view where the Fig -1.4 is before playing the song and Fig -1.5 is while playing the particular song then the playing song title will be visible in the red coloured one which is shown above**



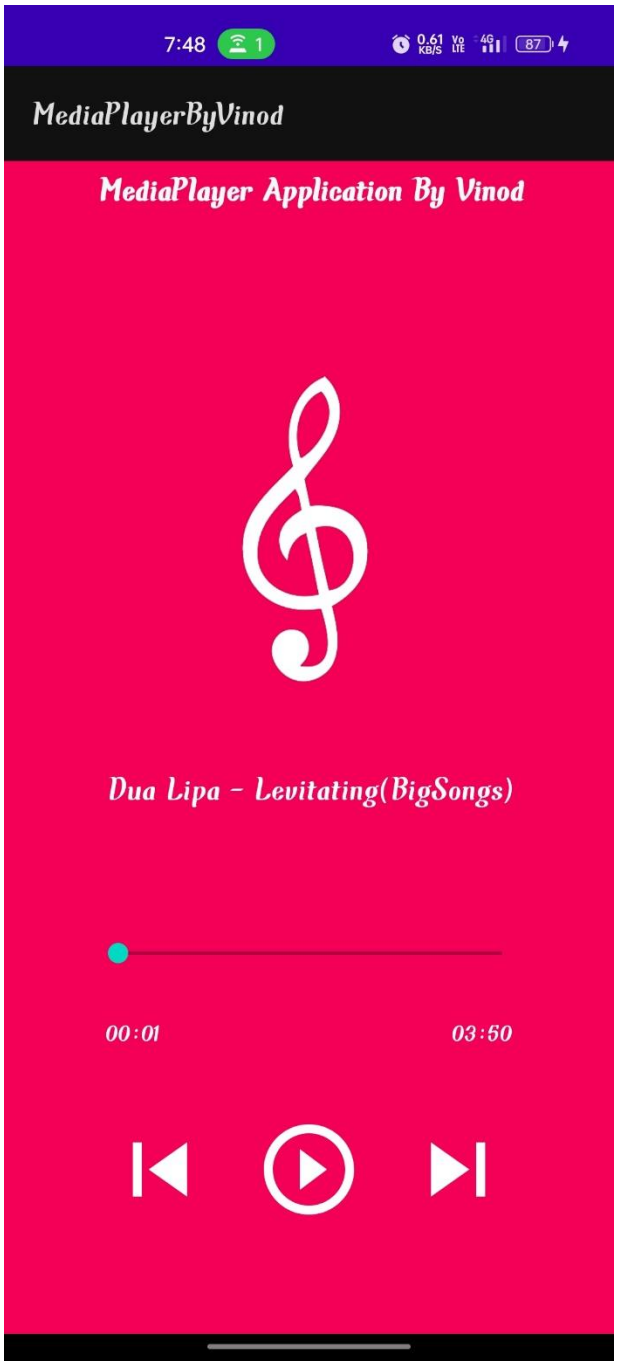


Fig – 1.5 : Interface while playing

Fig - 1.6 : Interface while pause

- **While playing a song in the list will appear the next screen of main Media player interface where the media buttons are customized here i.e play button,stop button,previous and next button in the screen**

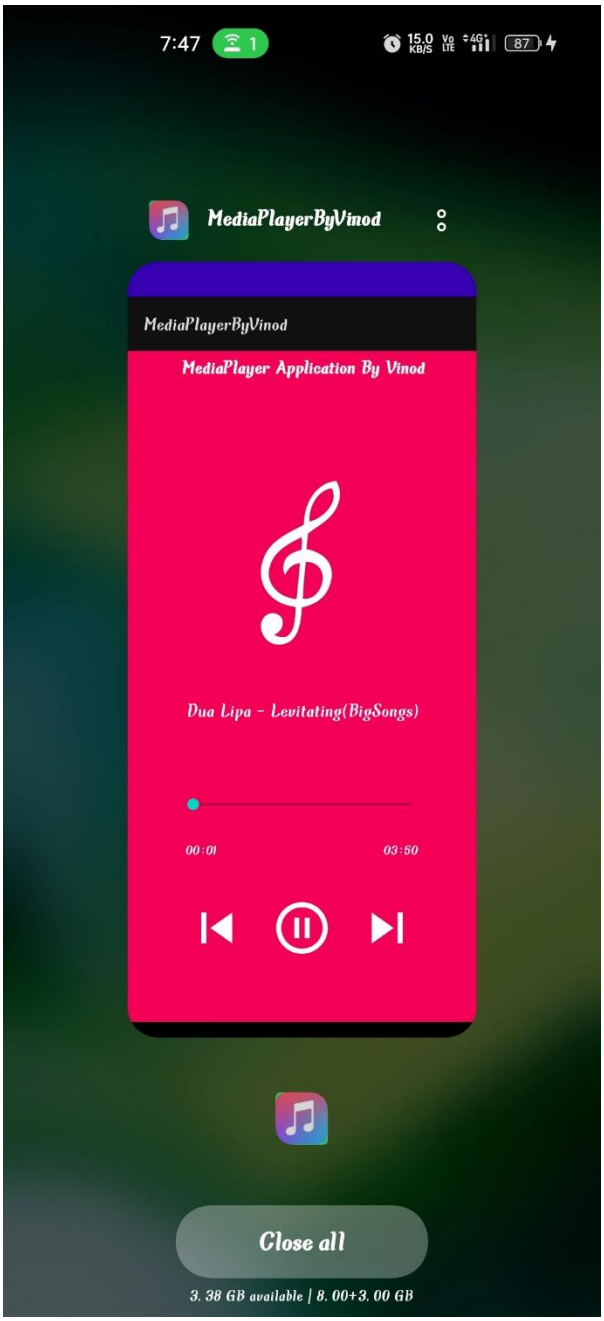


Fig – 1.7 : Bagroundplay

- The baground play screen is here when the user is using another application while Media player is in the baground that is appears like this

Source Code :-

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="#F50057"
    tools:context=".MusicPlayerActivity">

    <TextView
        android:id="@+id/song_title"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="10dp"
        android:layout_marginTop="20dp"
        android:layout_marginEnd="9dp"
        android:layout_marginBottom="290dp"
        android:ellipsize="marquee"
        android:padding="20dp"
        android:singleLine="true"
        android:text="Title of the song"
        android:textAlignment="center"
        android:textColor="@color/white"
        android:textSize="20dp" />

    <ImageView
        android:id="@+id/music_icon_big"
        android:layout_width="240dp"
        android:layout_height="240dp"
        android:layout_above="@id/controls"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_alignParentEnd="true"
        android:layout_marginStart="69dp"
        android:layout_marginTop="103dp"
        android:layout_marginEnd="70dp"
        android:layout_marginBottom="76dp"
        android:padding="20dp"
        android:src="@drawable/music_icon_big" />
```

```
<RelativeLayout
   android:id="@+id/controls"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:layout_alignParentBottom="true"
   android:layout_marginBottom="5dp"
   android:padding="40dp">

   <SeekBar
      android:id="@+id/seek_bar"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginStart="10dp"
      android:layout_marginTop="10dp"
      android:layout_marginEnd="10dp"
      android:layout_marginBottom="10dp"
      android:backgroundTint="@color/white"
      android:progressBackgroundTint="#000000" />

   <TextView
      android:id="@+id/current_time"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/seek_bar"
      android:layout_alignParentStart="true"
      android:layout_margin="20dp"
      android:text="0:00"
      android:textColor="@color/white" />

   <TextView
      android:id="@+id/total_time"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@id/seek_bar"
      android:layout_alignParentEnd="true"
      android:layout_margin="20dp"
      android:text="0:00"
      android:textColor="@color/white" />

   <RelativeLayout
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/total_time"
      android:padding="20dp">
```

```
        <ImageView
            android:id="@+id/previous"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentStart="true"
            android:layout_centerVertical="true"
            android:src="@drawable/ic_baseline_skip_previous_24" />

        <ImageView
            android:id="@+id/next"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentEnd="true"
            android:layout_centerVertical="true"
            android:src="@drawable/ic_baseline_skip_next_24" />

        <ImageView
            android:id="@+id/pause_play"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_centerInParent="true"
            android:src="@drawable/ic_baseline_pause_circle_outline_24" />

    </RelativeLayout>
</RelativeLayout>

<TextView
    android:id="@+id/main"
    android:layout_width="wrap_content"
    android:layout_height="92dp"
    android:layout_alignParentStart="true"
    android:layout_alignParentTop="true"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginStart="20dp"
    android:layout_marginTop="5dp"
    android:layout_marginEnd="18dp"
    android:layout_marginBottom="634dp"
    android:singleLine="false"
    android:soundEffectsEnabled="false"
    android:text="MediaPlayer Application By Vinod"
    android:textAlignment="center"
    android:textColor="#F6FBF9"
    android:textSize="20sp"
    android:textStyle="bold" />

</RelativeLayout>
```

Activity Main.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"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/songs_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:padding="10dp"
        android:text="SONGS"
        android:textColor="@color/black"
        android:textSize="20dp"
        android:textStyle="bold" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/no_songs_text"
        android:text="NO SONGS FOUND"
        android:layout_centerInParent="true"
        android:visibility="gone"/>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/songs_text" />



</RelativeLayout>
```

Recycler_item.xml :-

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#EFEFEF"
    android:layout_marginTop="5dp"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:paddingTop="10dp"
    android:paddingBottom="10dp"
    xmlns:tools="http://schemas.android.com/tools">


    <ImageView
        android:layout_width="36dp"
        android:layout_height="36dp"
        android:id="@+id/icon_view"
        android:layout_centerVertical="true"
        android:src="@drawable/music_icon"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:id="@+id/music_title_text"
        android:layout_toEndOf="@id/icon_view"
        android:padding="10dp"
        android:maxLines="1"
        android:ellipsize="end"
        tools:text="Music"
        android:textColor="@color/black"/>

</RelativeLayout>
```

MusicplayerActivity.java :-

```
package com.example.mediaplayerbyvinod;

import androidx.appcompat.app.AppCompatActivity;

import android.media.MediaPlayer;
import android.os.Bundle;
import android.os.Handler;
import android.widget.ImageView;
import android.widget.SeekBar;
import android.widget.TextView;

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

public class MusicPlayerActivity extends AppCompatActivity {

    TextView titleTv,currentTimeTv,totalTimeTv;
    SeekBar seekBar;
    ImageView pausePlay,nextBtn,previousBtn,musicIcon;
    ArrayList<AudioModel> songsList;
    AudioModel currentSong;
    MediaPlayer mediaPlayer = MyMediaPlayer.getInstance();
    int x=0;

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

        titleTv = findViewById(R.id.song_title);
        currentTimeTv = findViewById(R.id.current_time);
        totalTimeTv = findViewById(R.id.total_time);
        seekBar = findViewById(R.id.seek_bar);
        pausePlay = findViewById(R.id.pause_play);
        nextBtn = findViewById(R.id.next);
        previousBtn = findViewById(R.id.previous);
        musicIcon = findViewById(R.id.music_icon_big);

        titleTv.setSelected(true);

        songsList = (ArrayList<AudioModel>) getIntent().getSerializableExtra("LIST");
        setResourcesWithMusic();
        MusicPlayerActivity.this.runOnUiThread(new Runnable()
```

```
        @Override
        public void run() {
            if(mediaPlayer!=null){
                seekBar.setProgress(mediaPlayer.getCurrentPosition());
                currentTimeTv.setText(convertToMMSS(mediaPlayer.getCurrentPosition()+""));

                if(mediaPlayer.isPlaying()){
                    pausePlay.setImageResource(R.drawable.ic_baseline_pause_circle_outline_24);
                    musicIcon.setRotation(x++);
                }else{
                    pausePlay.setImageResource(R.drawable.ic_baseline_play_circle_outline_24);
                    musicIcon.setRotation(0);
                }
            }
            new Handler().postDelayed(this,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) {
        }

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

void setResourcesWithMusic(){
    currentSong = songsList.get(MyMediaPlayer.currentIndex);
    titleTv.setText(currentSong.getTitle());
    totalTimeTv.setText(convertToMMSS(currentSong.getDuration()));
    pausePlay.setOnClickListener(v-> pausePlay());
    nextBtn.setOnClickListener(v-> playNextSong());
    previousBtn.setOnClickListener(v-> playPreviousSong());

    playMusic();
}
```

```
    private void playMusic(){

        mediaPlayer.reset();
        try {
            mediaPlayer.setDataSource(currentSong.getPath());
            mediaPlayer.prepare();
            mediaPlayer.start();
            seekBar.setProgress(0);
            seekBar.setMax(mediaPlayer.getDuration());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private void playNextSong(){

        if(MyMediaPlayer.currentIndex== songsList.size()-1)
            return;
        MyMediaPlayer.currentIndex +=1;
        mediaPlayer.reset();
        setResourcesWithMusic();
    }

    private void playPreviousSong(){
        if(MyMediaPlayer.currentIndex== 0)
            return;
        MyMediaPlayer.currentIndex -=1;
        mediaPlayer.reset();
        setResourcesWithMusic();
    }

    private void pausePlay(){
        if(mediaPlayer.isPlaying())
            mediaPlayer.pause();
        else
            mediaPlayer.start();
    }

    public static String convertToMMSS(String duration){
        Long millis = Long.parseLong(duration);
        return String.format("%02d:%02d",
                TimeUnit.MILLISECONDS.toMinutes(millis) % TimeUnit.HOURS.toMinutes(1),
                TimeUnit.MILLISECONDS.toSeconds(millis) %
TimeUnit.MINUTES.toSeconds(1));
    }
}
```

MainActivity.java :-

```
package com.example.mediaplayerbyvinod;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.Manifest;
import android.content.pm.PackageManager;
import android.database.Cursor;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.TextView;
import android.widget.Toast;

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

public class MainActivity extends AppCompatActivity {

    RecyclerView recyclerView;
    TextView noMusicTextView;
    ArrayList<AudioModel> songsList = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        recyclerView = findViewById(R.id.recycler_view);
        noMusicTextView = findViewById(R.id.no_songs_text);

        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,projectio
n,selection,null,null);
        while(cursor.moveToNext()){
            AudioModel songData = new
AudioModel(cursor.getString(1),cursor.getString(0),cursor.getString(2));
            if(new File(songData.getPath()).exists())
                songsList.add(songData);
        }
        if(songsList.size()==0){
            noMusicTextView.setVisibility(View.VISIBLE);
        }else{
            //recyclerview
            recyclerView.setLayoutManager(new LinearLayoutManager(this));
            recyclerView.setAdapter(new MusicListAdapter(songsList,getApplicationContext()));
        }
    }
    boolean checkPermission(){
        int result = ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.READ_EXTERNAL_STORAGE);
        if(result == PackageManager.PERMISSION_GRANTED){
            return true;
        }else{
            return false;
        }
    }

    void requestPermission(){

if(ActivityCompat.shouldShowRequestPermissionRationale(MainActivity.this,Manifest.permissi
on.READ_EXTERNAL_STORAGE)){
            Toast.makeText(MainActivity.this,"READ PERMISSION IS REQUIRED,PLEASE
ALLOW FROM SETTTINGS",Toast.LENGTH_SHORT).show();
        }else
            ActivityCompat.requestPermissions(MainActivity.this,new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE},123);
    }

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

MyMediaPlayer.java :-

```
package com.example.mediaplayerbyvinod;

import android.media.MediaPlayer;

public class MyMediaPlayer {
   static MediaPlayer instance;

   public static MediaPlayer getInstance(){
      if(instance == null){
         instance = new MediaPlayer();
      }
      return instance;
   }

   public static int currentIndex = -1;
}
```

MusicListAdapter.java :-

```
package com.example.mediaplayerbyvinod;

import android.content.Context;
import android.content.Intent;
import android.graphics.Color;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.ArrayList;

import androidx.recyclerview.widget.RecyclerView;

public class MusicListAdapter extends
RecyclerView.Adapter<MusicListAdapter.ViewHolder>{
   ArrayList<AudioModel> songsList;
   Context context;

   public MusicListAdapter(ArrayList<AudioModel> songsList, Context context) {
      this.songsList = songsList;
      this.context = context;
   }
```

```
    @Override
    public ViewHolder onCreateViewHolder( ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.recycler_item,parent,false);
        return new MusicListAdapter.ViewHolder(view);
    }

    @Override
    public void onBindViewHolder( MusicListAdapter.ViewHolder holder, int position) {
        AudioModel songData = songsList.get(position);
        holder.titleTextView.setText(songData.getTitle());

        if(MyMediaPlayer.currentIndex==position){
            holder.titleTextView.setTextColor(Color.parseColor("#FF0000"));
        }else{
            holder.titleTextView.setTextColor(Color.parseColor("#000000"));
        }
        holder.itemView.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {

                MyMediaPlayer.getInstance().reset();
                MyMediaPlayer.currentIndex = position;
                Intent intent = new Intent(context,MusicPlayerActivity.class);
                intent.putExtra("LIST",songsList);
                intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                context.startActivity(intent);
            }
        });
    }

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

    public class ViewHolder extends RecyclerView.ViewHolder{
        TextView titleTextView;
        ImageView iconImageView;
        public ViewHolder(View itemView) {
            super(itemView);
            titleTextView = itemView.findViewById(R.id.music_title_text);
            iconImageView = itemView.findViewById(R.id.icon_view);
        }
    }
}
```

AudioModel.java :-

```
package com.example.mediaplayerbyvinod;

import java.io.Serializable;

public class AudioModel implements Serializable {
    String path;
    String title;
    String duration;

    public AudioModel(String path, String title, String duration) {
        this.path = path;
        this.title = title;
        this.duration = duration;
    }

    public String getPath() {
        return path;
    }

    public void setPath(String path) {
        this.path = path;
    }

    public String getTitle() {
        return title;
    }

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

    public String getDuration() {
        return duration;
    }

    public void setDuration(String duration) {
        this.duration = duration;
    }
}
```

Conclusion :-

Building a media player is a challenging but not impossible task. The challenge lies in understanding AWT application architecture, how to create a JMF player, and how to manage the JMF player by appropriately responding to the player's controller events. This article has shed some light on each of these areas. Not only do you have an improved understanding of what is needed to create a media player; you also have a useful media player tool for your developer toolbox (and pleasure)!