A Project Report

On

Media Player

Submitted in partial fulfillment of the

requirement for the award of the degree of

BACHELORS OF COMPUTER APPLICATION

DEGREE

Session 2023-24

in

Capstone project

By

Kirti Kumar (21SCSE1040037)

Surlay Rao (21SCSE1040025)

Mahak Mustafa (21SCSE1040073)

Under the guidance of

Dr Neha Singh

SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY

GALGOTIAS UNIVERSITY, GREATER NOIDA

INDIA

APRIL, 2024

SCHOOL OF COMPUTER APPLICATION AND TECHNOLOGY
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the project, entitled **"Media Player"** in partial fulfillment of the requirements for the award of the BCA (Bachelores of Computer Application) submitted in the School of Computer Application and Technology of Galgotias University, Greater Noida, is an original work carried out during the period of Jan 2024 to May 2024, under the supervision of Dr Neha Singh Department of Computer Science and Engineering/School of Computer Application and Technology , Galgotias University, Greater Noida.

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Student Names (Admission No.)

Kirti Kumar (21SCSE1040037)

Surlay Rao (21SCSE1040025)

Mahak Mustafa (21SCSE1040073)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Guide Name

Dr Neha Singh

CERTIFICATE

This is to certify that Project Report entitled “Media Player” which is submitted by Kirti Kumar 21SCSE1040037, Surlay Rao 21SCSE1040025, Mahak Mustafa 21SCSE1040073 in partial fulfillment of the requirement for the award of degree BCA. in Department of SCSE School of Computer Application and Technology Galgotias University, Greater Noida, India is a record of the candidate own work carried out by him/them under my supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.

Signature of Examiner(s) **Signature of Supervisor(s)**

Date: April, 2024

Place: Greater Noida

TABLE OF CONTENTS

Page

CHAPTER 1 : INTRODUCTION

This report provides an overview of a Media Player Android application, highlighting its features, user experience, market analysis, and future prospects. Android media player applications have become integral tools for users to manage and enjoy multimedia content on mobile devices, making it essential to understand their impact on the digital landscape.

In today's digital age, the consumption of multimedia content has become an integral part of our daily lives. Whether it's music , podcasts, or any other form of media, we rely on media player applications to effortlessly and efficiently manage and enjoy these digital treasures. A media player application is a software tool designed to play, organize, and sometimes even edit various types of media files, providing users with a seamless and immersive multimedia experience.

Welcome to the next generation of music listening! We are thrilled to introduce **Musify**, a revolutionary music player designed to enhance your audio journey. As passionate music enthusiasts ourselves, we understand the importance of a seamless and immersive music experience.

Musify is not just another player; it's a comprehensive platform crafted with precision and care. Whether you're a casual listener or a die-hard audiophile, our application is tailored to meet your unique needs and preferences.

1.1.1 Android

“Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets”. “Android's user interface is mainly based on direct manipulation, using touch gestures that loosely correspond to real-world actions, such as swiping, tapping and pinching, to manipulate on-screen objects, along with a virtual keyboard for text input. In addition to touchscreen devices, Google has further developed Android TV for televisions, Android Auto for cars, and Android Wear for wrist watches, each with a specialized user interface”.

Figure1.Android Logo

1.1.2 Android Architecture

“We studied the Android system architecture. Android system is a Linux-based system, Use of the software stack architecture design patterns . As shown in Figure 1, the Android architecture consists of four layers: Linux kernel, Libraries and Android runtime, Application framework

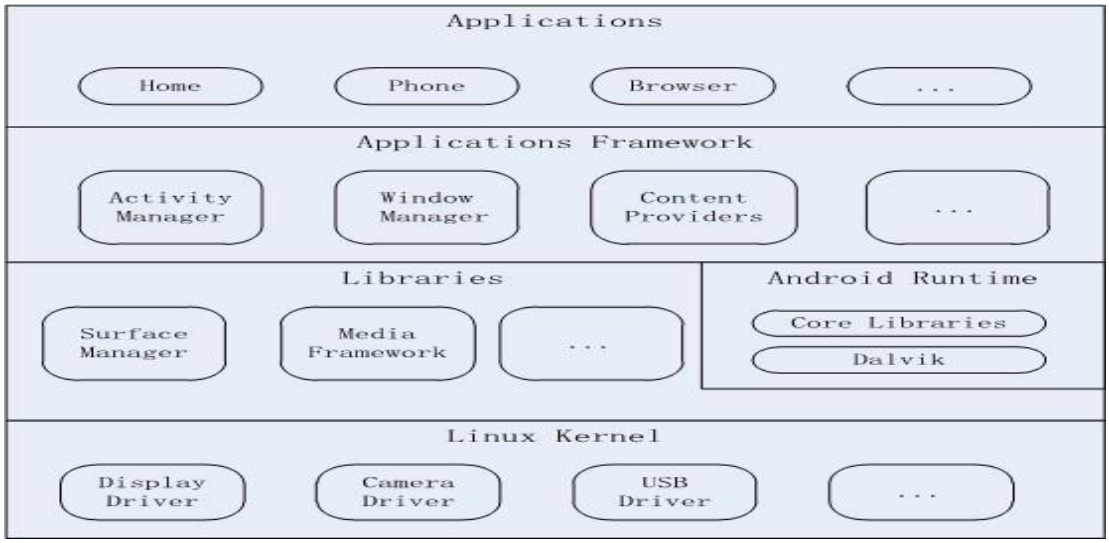


Figure2. Android Architecture

A) Applications:

“Android app will be shipped with a set of core applications including client, SMS program, calendar, maps, browser, contacts, and others. All these application programs are developed in Java”.

B) Application Framework :

“The developer is allowed to access all the API framework of the core programs. The application framework simplifies the reuse of its components. Any other app can release its functional components and all other apps can access and use this component (but have to follow the security of the framework). Same as the users can be able to substitute the program components with this reuse mechanism”

C) Libraries and Android Runtime

"The library is divided in to two components: Android Runtime and Android Library. Android Runtime is consisted of a Java Core Library and Dalvik virtual machine. The Core Library provides Java core library with most functions. Dalvik virtual machine is register virtual machine and makes some specific improvements for mobile device. Android system library is support the application framework, it is also an important link connecting between application framework and Linux Kernel. This system library is developed in C or C++ language. These libraries can also be utilized by the different components in the Android system. They provide service for the developers through the application framework".

D) Linux Kernel

"The kernel system service provided by Android inner nuclear layer is based on Linux 2.6 kernel, Operations like internal storage, process management, internet protocol, bottom- drive and other core service are all based on Linux kernel".

1.1.3 Software Development Kit(SDK)

"A software development kit (SDK or "devkit") has many definitions but it is usually callded a set of software development tools which help us in many ways or allows the creation of applications which can be useful in future in a lot of things as seen from this report for a certain softwarepackage, software framework, hardware platform, computer system, video game console, operating system, or similar development platform. To create applications you have to download this software development kit". "For example, if you want to create an Android app you require an SDK with java programming, for iOS apps you require an iOS SDK with swift language, and to develop MS Windows apps you require thepackage, software framework, hardware platform, computer system, video game console, operating

system, or similar development platform. To create applications you have to download this software development kit". "For example, if you want to create an Android app you require an SDK with java programming, for iOS apps you require an iOS SDK with swift language, and to develop MS Windows apps you require the .net language. There are also SDKs that are installed in apps to provide analytics and data about activity. Prominent examples include Google and Facebook".

"Android Studio is an integrated development environment (IDE) for developing for the Android platform. It was announced on May 16, 2013 at the Google I/Oconference". "Android Studio is freely available under the Apache License 2.0. Android Studio was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from versionconference". "Android Studio is freely available under the Apache License 2.0. Android Studio was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014". "The first stable build was released in December 2014, starting from version 1.0. Based on JetBrains' software, Android Studio is designed specifically for Android development. It is available for download on Windows, Mac OS Xand Linux, and replaced Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development".

Figure3.Android Studio Logo

CHAPTER 2 : LITERATURE REVIEW

Paper 1: Music player Android

“The authors have tested the app in three environments including hardware, software and network. Test hardware environment is Lenovo Y460 laptop and millet M1 phone; software environment is windows 11 and phone system environment is android 12.0”. “Network environment is China Mobile which is 10M broadband, WIFI LAN and China Mobile GPRS network.By testing each function on mobile phone and the computer simulator, the result showed that video player and audio player run well and no advertising”. “Sina weibo client can successfully complete OAuth2.0 certificate authority and login and collect the basic data of the user information from sina server and no redundant information. Expected effect is achieved after testing all the functions”. “They says that since the Weibo client has to access to the network, when tested on an android phone, the performance under the environment of WIFI network and mobile 2G GPRS network can meet the expected requirements”. Paper 2: Research on Development of android Applications.

“This article gives a detailed introduction of android application framework and the working principal of android applications. Finally, a music player on the android platform was put forward as an example to illustrate this mechanism”.

“The authors say that android application development college challenge has only been held two times, but it greatly encourages and promotes the creativity of the college students”. “With more and more competitive teams participating the contest, it will be harder to win an award. However, many exciting applications will be presented in the contest. This challenge gives us an opportunity to learn about that a lot of ideas we think about can be implemented on android platform. At the same time, the contest provides a stage for android developer to discuss and communicate with each other. This can effectively

promote the development of android and attract more software engineers to develop applications on android platform".

Paper 4: "A model driven approach for android applications development"

"This paper proposes a MDE approach for android applications development, which addresses how to model specific aspects of android applications, as intent and a data/service request, using standard UML notations". "Moreover, it supports static and behavioral code generation from UML class and sequence diagrams, according to the rules imposed by the android platform. To demonstrate our approach, a case study was conducted, in which an android application was modeled in UML and code was generated from it. To generate code, the extension of GenCode was used". "However, the actual version of GenCode tool that supports the proposed approach, only made an automatic transformation from UML class and sequence diagrams to the target android Java code, without consider any optimization in the generated code. As future work, we plan to extend this tool in order to consider the good practices for android development , and thus generating efficient code".

Paper 5: Design of Android based Media Player

This paper[5] tells us that many or a lot of users use their mobile phone but obviously, but the media player has many limitations. The increase in development of communication and network, multimedia based technology is adopted in media player. Different approaches of media player "shown in this paper are plug-in extension technology, multimedia based on hierarchy, media player based on file browser, media player based on FFmpeg (Fast Forward Moving Picture Expert Group), media player based on file server".

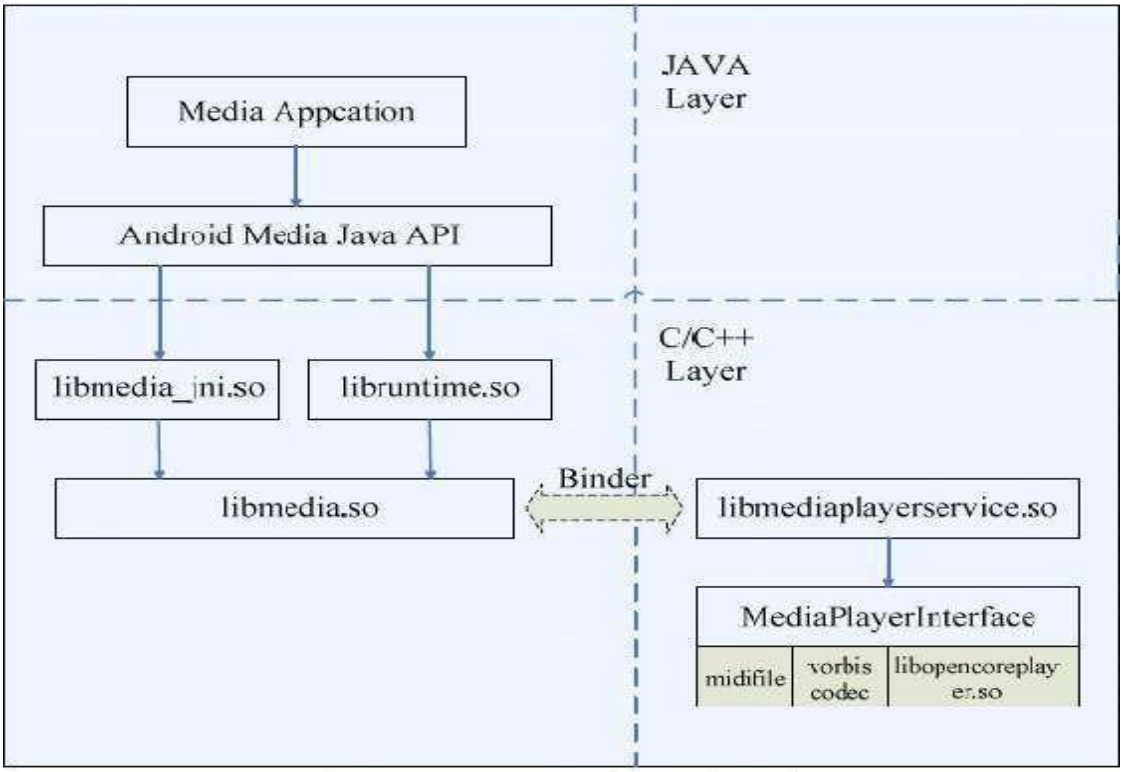


Figure 4. Architecture of multimedia player software platform

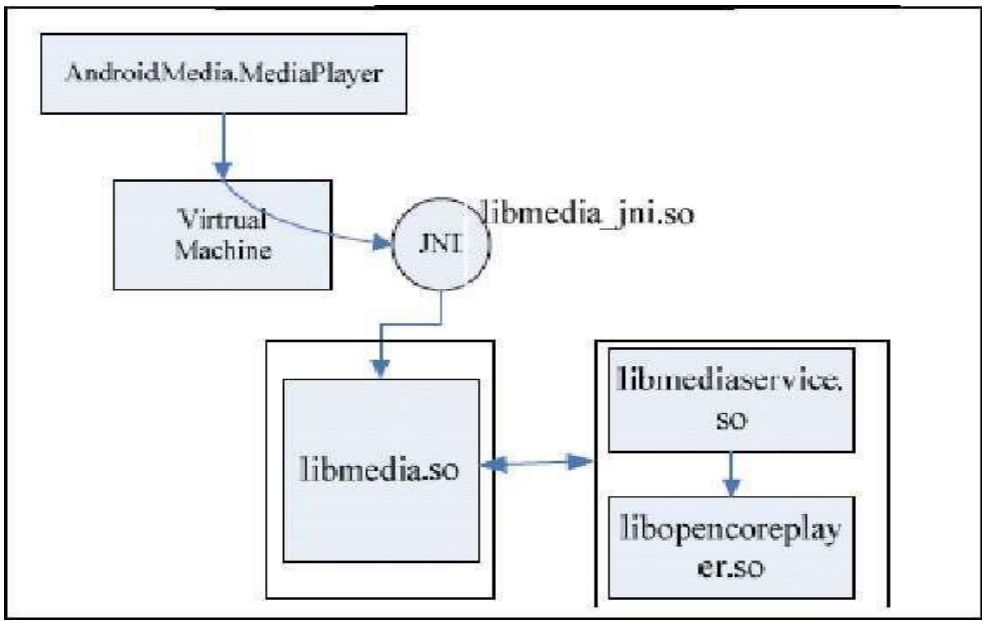


Figure 5. Android media framework

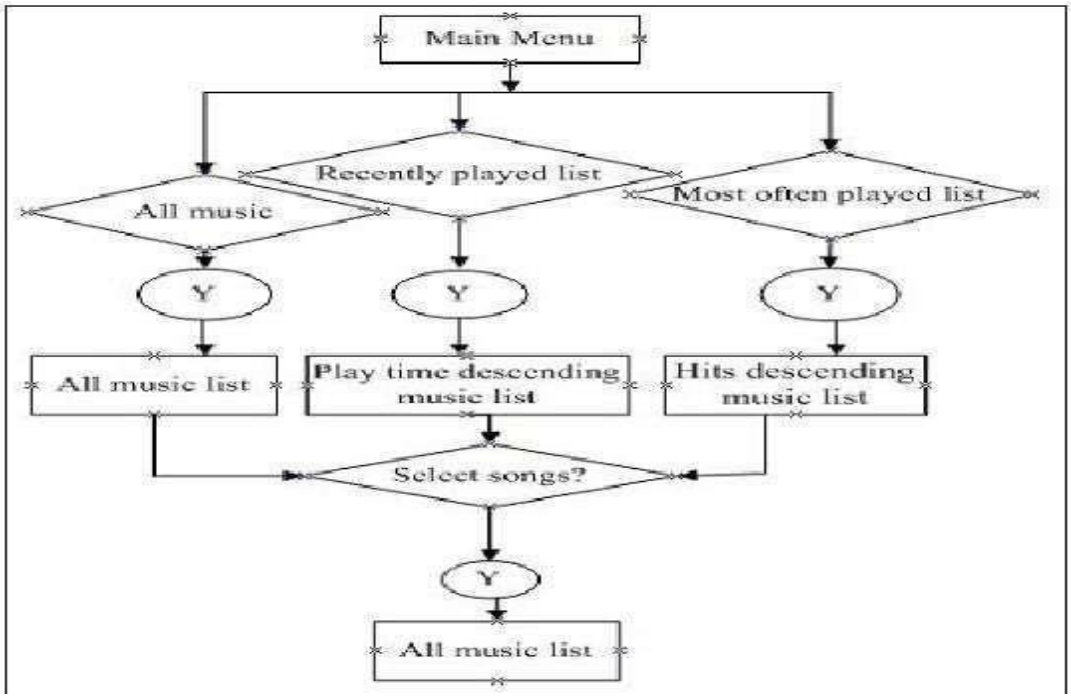


Figure 6. System processes of media player

Paper 6: The Android - A Widely Growing Mobile Operating System With its Mobile based

Applications[6]

"Android operating system is one of the most widely used mobile Operating System these days and also enhancing its use for making betterment in different areas of life. Android mobile operating system is based on the Linux kernel and is developed by Google and primarily designed for smartphones and tablets. Android Operating System consist of four main layers, the specifying architecture is given in this paper. The advanced Smart applications of android in mobile, real-time and wireless sensor network are widening their service areas. Android is a disruptive technology, which was introduced initially on mobile handsets, but has much wider potential. In this paper we are studying, one of the smart and enhancing Android operating

system application which are based on Automated and tracking from remote distance. These application helps students, teachers, parents, patients and users of home appliance as anytime and anywhere basis. Being part of today's advance world, using fastest acceptable and mobile Android Operating System it's possible to develop automated attendance system, secure transferring of medical data and automated home appliance monitoring system".

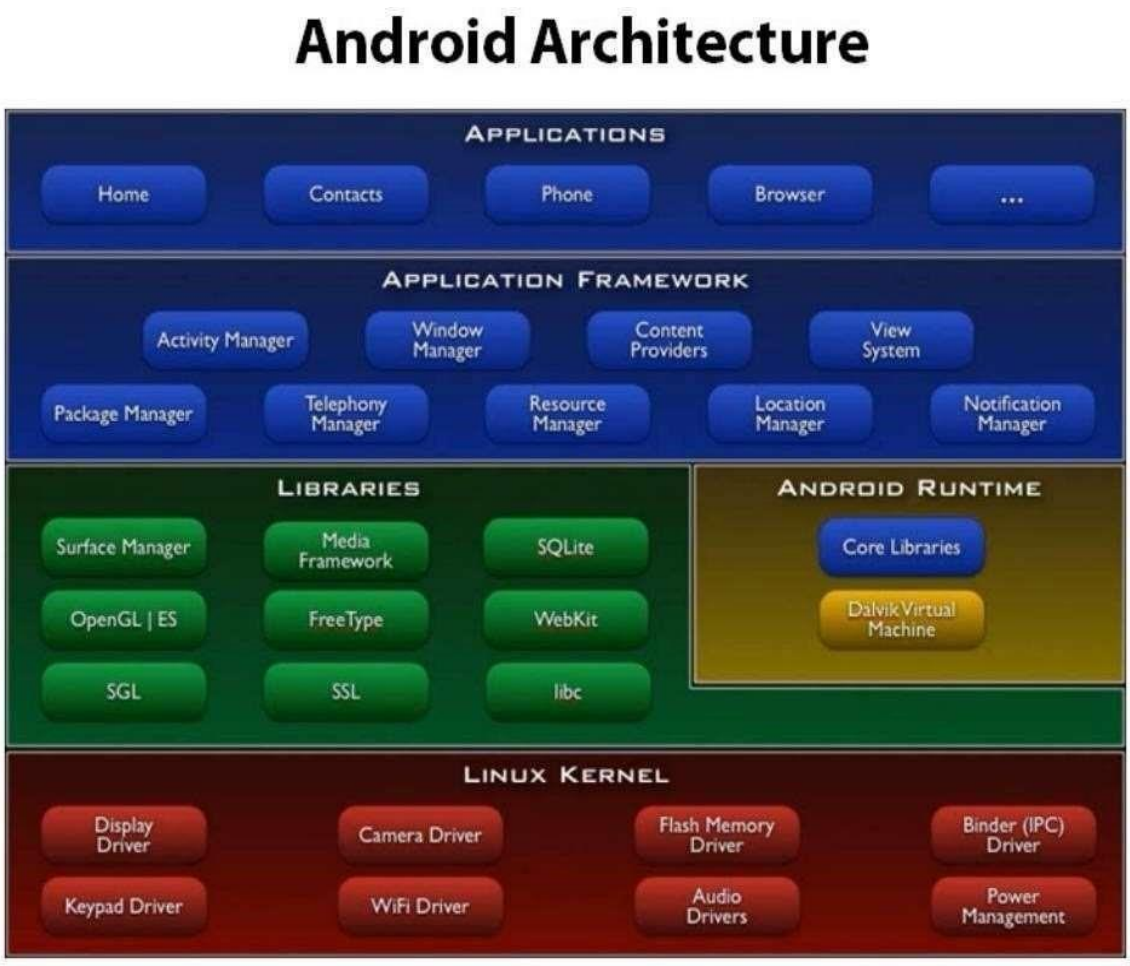


Figure 7.Android Architecture

CHAPTER 3:SYSTEM DEVELOPMENT

3.1 Designing navigation drawer

To add a navigation drawer, declare wer user interface with aDrawerLawet object as the root view of werlawet. “Inside theDrawerLawet, add one view that contains the main content for the screen (wer primary lawet when the drawer is hidden) and another view that contains the contents of the navigation drawer”.

“For example, the following lawet uses a DrawerLawet with two child views: a FrameLawet to contain the main content (populated by a Fragment at runtime), and a ListView for the navigation drawer”.

```
<FrameLawet
    android:id="@+id/content_frame"
    android:lawet_width="match_parent"
    android:lawet_height="match_parent"/>
<ListViewandroid:id="@+id/left_drawer"
    android:lawet_width="240dp"
    android:lawet_height="match_parent"
    android:lawet_gravity="start"
    android:choiceMode="singleChoice"
    android:divider="@android:color/transparent"
    android:dividerHeight="0dp"
    android:background="#111"/>
```

This lawet demonstrates some important lawet characteristics:

- "The main content view (the FrameLawet above) must be the first child in the DrawerLawet because the XML order implies z-ordering and the drawer must be on top of the content".
- "The main content view is set to match the parent view's width and height, because it represents the entire UI when the navigation drawer is hidden".
- "The drawer view specifies its width in dp units and the height matches the parent view. The drawer width should be no more than 320dp so the user can always see a portion of the main content".

Initialize the Drawer List

"In wer activity, one of the first things to do is initialize the navigation drawer's list of items. How we do so depends on the content of wer app, but a navigation drawer often consists of a ListView, so the list should be populated by an Adapter (such as ArrayAdapter or SimpleCursorAdapter)".

For example, here's how we can initialize the navigation list with a string array:

```
mPlanetTitles=getResources().getStringArray(R.array.planets_array);
mDrawerLawet=(DrawerLawet)findViewById(R.id.drawer_lawet);
mDrawerList=(ListView)findViewById(R.id.left_drawer);


mDrawerList.setAdapter(newArrayAdapter<String>(this,
      R.lawet.drawer_list_item,mPlanetTitles));

mDrawerList.setOnItemClickListener(newDrawerItemClickListener());

...
```

"This code also calls setOnItemClickListener() to receive click events in the navigation drawer's list. The next section shows how to implement this interface and change the content view when the user selects an item".

Handle Navigation Click Events

"When the user selects an item in the drawer's list, the system calls onItemClick() on theOnItemClickListener given to setOnItemClickListener()".

"What we do in the onItemClick() method depends on how we've implemented wer app structure. In the following example, selecting each item in the list inserts a different Fragment into the main content view (theFrameLawet element identified by the R.id.content_frame ID)":

```
privateclassDrawerItemClickListenerimplementsListView.OnItemClickListener{
    @Override
    publicvoidonItemClick(AdapterView parent,Viewview,int position,long id){
        selectItem(position);
    }
}

privatevoidselectItem(int position){
    Fragmentfragment=newPlanetFragment();
    Bundleargs=newBundle();
    args.putInt(PlanetFragment.ARG_PLANET_NUMBER, position);
    fragment.setArguments(args);

    FragmentManagerfragmentManager=getFragmentManager();
    fragmentManager.beginTransaction()
                .replace(R.id.content_frame, fragment)
```

```
            .commit();

    mDrawerList.setItemChecked(position,true);
    setTitle(mPlanetTitles[position]);
    mDrawerLawet.closeDrawer(mDrawerList);
}

@Override
publicvoidsetTitle(CharSequence title){
    mTitle= title;
    getActionBar().setTitle(mTitle);
}
```

Listen for Open and Close Events

"To listen for drawer open and close events, call setDrawerListener() on wer DrawerLawet and pass it an implementation of DrawerLawet.DrawerListener. This interface provides callbacks for drawer events such asonDrawerOpened() and onDrawerClosed()"

"However, rather than implementing the DrawerLawet.DrawerListener, if wer activity includes the action bar we can instead extend the ActionBarDrawerToggle class. The ActionBarDrawerToggle implementsDrawerLawet.DrawerListener so we can still override those callbacks, but it also facilitates the proper interaction behavior between the action bar icon and the navigation drawer (discussed further in the next section)". we can instead extend the ActionBarDrawerToggle class. The ActionBarDrawerToggle implementsDrawerLawet.DrawerListener so we can still override those callbacks, but it also facilitates the proper interaction behavior between the

action bar icon and the navigation drawer (discussed further in the next section)".

"As discussed in the Navigation Drawer design guide, we should modify the contents of the action bar when the drawer is visible, such as to change the title and remove action items that are contextual to the main content. The following code shows how we can do so by overriding

`DrawerLawet.DrawerListener` callback methods with an instance of the ActionBarDrawerToggle

```
    };
    mDrawerLawet.setDrawerListener(mDrawerToggle);
}

@Override
publicbooleanonPrepareOptionsMenu(Menu menu){
        booleandrawerOpen=mDrawerLawet.isDrawerOpen(mDrawerList);
    menu.findItem(R.id.action_websearch).setVisible(!drawerOpen);
    returnsuper.onPrepareOptionsMenu(menu);
```

```
    }
}
```

3.2 Creating another activity

Respond to the Send Button

1. “In Android Studio, from the res/lawet directory, edit thecontent_my.xml file”.
2. “Add the android:onClick attribute to the <Button>element”.
   res/lawet/content_my.xml

```
<Button
   android:lawet_width="wrap_content"
   android:lawet_height="wrap_content"
   android:text="@string/button_send"
   android:onClick="sendMessage"/>
```

“The android:onClick attribute’s value, "sendMessage", is the name of a method in wer activity that the system calls when the user clicks the button”.

3. “In the java/com.shashwat.iciip directory, open the MainActivity.java file”.

4. “Within the MainActivity class, add the sendMessage() method stub shown below”.

   MainActivity.java

```
publicvoidsendMessage(View view){
}
```

“In order for the system to match this method to the method name given to android:onClick, the signature must be exactly as shown. Specifically, the method

must”: o Be public

- Have a void return value
- "Have a View as the only parameter (this will be the View that was clicked)"

Next, we'll fill in this method to read the contents of the text field and deliver that text to another activity.

Build an Intent

1. "In MainActivity.java, inside the sendMessage() method, create an Intent to start an activity calledDisplayMessageActivity with the following code":

   java/com.shashwat.iciip/MainActivity.java

```
publicvoidsendMessage(View view){
  Intentintent=newIntent(this,DisplayMessageActivity.class);
}
```

   The constructor used here takes two parameters: o "A Context as its first parameter (this is used because the Activity class is a subclass of Context)"

   - "The Class of the app component to which the system should deliver the Intent (in this case, the activity that should be started)"

   "Android Studio indicates that we must import the Intent class".

2. At the top of the file, import the Intent class:

   MainActivity.java

```
importandroid.content.Intent;
```

3. Inside the sendMessage() method, use findViewById() to get the EditText element.

MainActivity.java

```
publicvoidsendMessage(View view){
  Intentintent=newIntent(this,DisplayMessageActivity.class);
 EditTexteditText=(EditText)findViewById(R.id.edit_message);
}
```

4. “At the top of the file, import the EditText class.

   In Android Studio, press Alt + Enter (option + return on Mac) to import missing classes”.

5. Assign the text to a local message variable, and use the putExtra() method to add its text value to the intent.

   MainActivity.java

   “An Intent can carry data types as key-value pairs called extras. The putExtra() method takes the key name in the first parameter and the value in the second parameter”.

6. At the top of the MyActivity class, add the EXTRA_MESSAGE definition as follows:

MainActivity.java

```
publicclassMyActivityextendsAppCompatActivity{
    publicfinalstaticString EXTRA_MESSAGE ="shashwat.com.iciip.MESSAGE";
    ...
}
```

"For the next activity to query the extra data, we should define the key for wer intent's extra using a public constant. It's generally a good practice to define keys for intent extras using wer app's package name as a prefix. This ensures the keys are unique, in case wer app interacts with other apps".

7. "In the sendMessage() method, to finish the intent, call the startActivity() method, passing it theIntent object created in step 1".

With this new code, the complete sendMessage() method that's invoked by the Send button now looks like this:

MainActivity.java

```
publicvoidsendMessage(View view){
    Intentintent=newIntent(this,DisplayMessageActivity.class);
    EditTexteditText=(EditText)findViewById(R.id.edit_message);
    String message =editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```

“The system receives this call and starts an instance of the Activity specified by the Intent. Now we need to create the DisplayMessageActivity class in order for this to work”.

Create the Second Activity

1. “In Android Studio, in the java directory, select the package, com.mycompany.iciip, right- click, and select New > Activity > Blank Activity”.

2. In the Choose options window, fill in the activity details:

   - Activity Name: DisplayMessageActivity
   - Lawet Name: activity_display_message
   - Title: My Message
   - Hierarchical Parent: com.mycompany.myfirstapp.MyActivity
   - Package name: com.mycompany.myfirstapp

   Click Finish.

3. Open the DisplayMessageActivity.java file.
   “The class already includes an implementation of the required onCreate() method. We update the implementation of this method later”.

Receive the Intent

“Every Activity is invoked by an Intent, regardless of how the user navigated there. We can get the Intentthat started wer activity by calling getIntent() and retrieve the data contained within the intent”.

1. “In the mainactivity directory, edit the DisplayMessageActivity.java file”.

2. “Get the intent and assign it to a local variable”.

```
Intentintent=getIntent();
```

3. “At the top of the file, import the Intent class”.

4. Extract the message delivered by MyActivity with the getStringExtra() method.

```
String message =intent.getStringExtra(MyActivity.EXTRA_MESSAGE);
```

Display the Message

1. “In the res/lawet directory, edit the content_display_message.xml file”.

2. “Add an android:id attribute to the RelativeLawet. We need this attribute to reference the object from wer app code”.

```
<RelativeLawetxmlns:android="http://schemas.android.com/apk/res/android"
...
android:id="@+id/content">
</RelativeLawet>
```

3. “Switch back to editing DisplayMessageActivity.java”.

4. “In the onCreate() method, create a TextView object”.

```
TextViewtextView=newTextView(this);
```

5. “Set the text size and message with setText()”.

```
textView.setTextSize(40);  textView.setText(message);
```

6. “Add the TextView to the RelativeLawet identified by R.id.content”.

```
RelativeLawetlawet=(RelativeLawet)findViewById(R.id.content);  lawet.addView(textView);
```

7. “At the top of the file, import the TextView class”.

“In Android Studio, press Alt + Enter (option + return on Mac) to import missing classes”. 3.3 Video Player

Video Player is achieved through the Android Studio platform.It begins with the study of operating “mechanism, Android platform media layer structure, xml customizable interface”, Content Providers achieves file scanning to get a list of media files, MediaPlayer class, file parsing, “Surface Flinger interface. After that, we could develop an Android-based mobile video player. Realize media library, video player, file opening, audio, video, photographs and other functions. Figure below is system flow chart”.

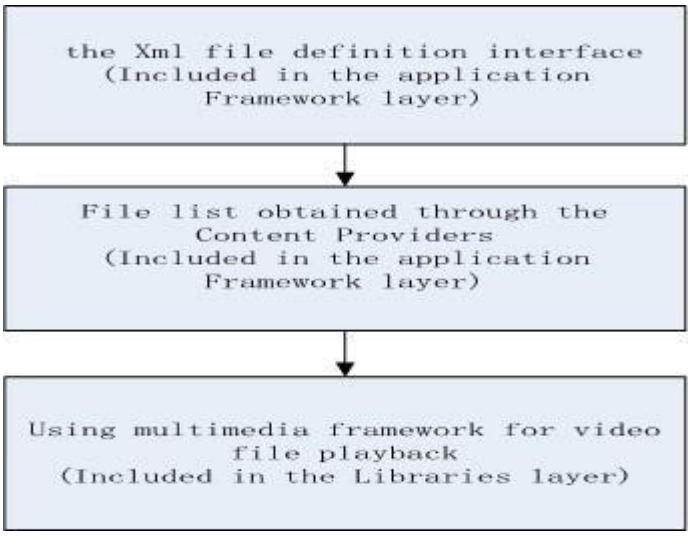


Figure 8.System Flow Chart

“The software interface is defined through XML files. XML layout files control view, is not only simple, but also isolated the View control logic from

Java code and controlled by inserted into XML files. Reflects the MVC principle in a better way and also reflects the principle of

separation of logic and views. This software obtains the list of media files by scanning through

Content Providers. Content Providers is recognized as a bridge between the data storing and searching across programs. The function is to achieve data sharing among different Apps, it is the only way to share data with other apps. Figure below shows the media layer structure".

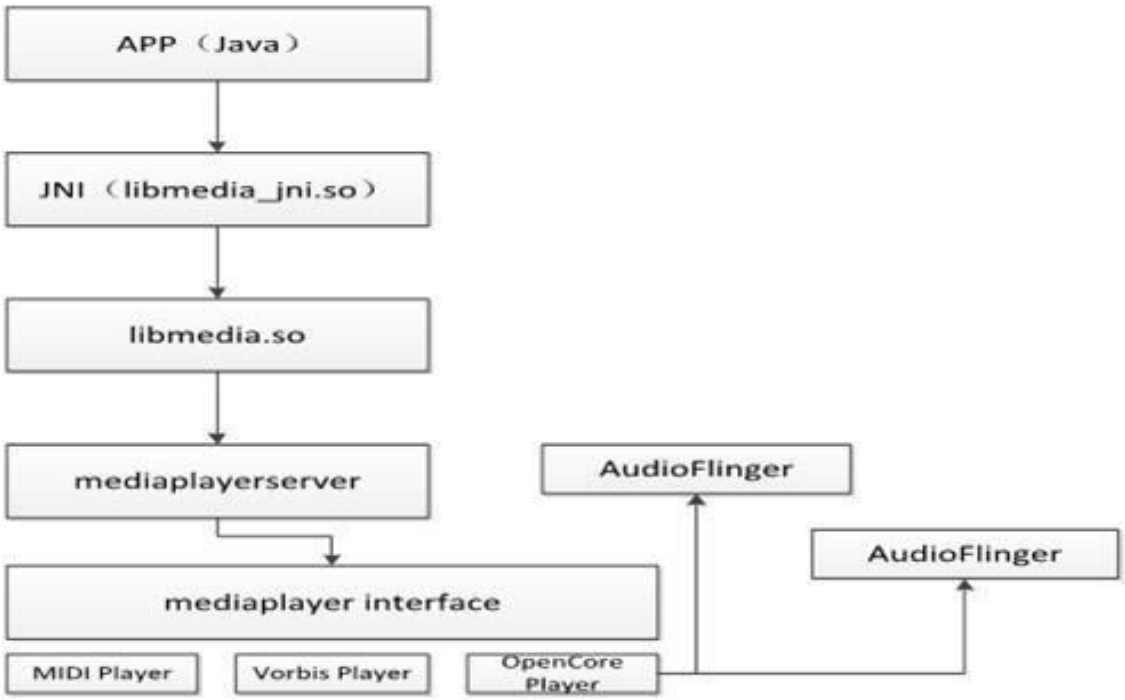


Figure 9.Media Layer Structure

"The upper applications of Android-MediaPlayer are implemented by JAVA, realized logic processing. JAVA program realizes the playback of video file and online video by calling the underlying media library libmedi.so through JNI interface. MediaPlayer can be roughly divided into two parts at run time: Client and Server. They are running in two separated processes. Binder used between them to achieve IPC communication. Mediaplayerservice in Figure 3 is a server- side implementation repository. MediaPlayer calls media

playback capabilities provided by Opencore to implement video file playback, Opencore responsible media file format parsing, decoding audio and video data, and outputs the media data. Opencore calls SurfaceFlinger" interface to realize the showing of video data and by applying AudioFlinger interface to realize the playback of audio data.

"In the Android media layer, the most important class is MediaPlayer.

MediaPlayer class and its associated structures are shown in Figure below".

"Vitamio is an open multimedia framework for Android, with hardware accelerated decoder and renderer. Vitamio can play 720p/1080p HD mp4,mkv,m4v,mov,flv,avi,rmvb,rm,ts,tp and many other video formats in Android and iOS. Almost all popular streaming protocols are supported by Vitamio, including HLS(m3u8), MMS, RTSP, RTMP, and HTTP".

Integrating SDK into your application

- Create a New Android project
- import vitamiolibrary.jar into your applicaiton project /libs directory
- Add libvitamio.so into your application project /libs directory
- Copy the recourse like class,picture from Demo into app project

AUDIO PLAYER

MediaPlayer class can be used to control playback of audio files and streams.

State Diagram

“Playback control of audio/video files and streams is managed as a state machine. The following diagram shows the life cycle and the states of a MediaPlayer object driven by the supported playback control operations. The ovals represent the states a MediaPlayer object may reside in. The arcs represent the playback control operations that drive the object state transition. There are two types of arcs. The arcs with a single arrow head represent synchronous method calls, while those with a double arrow head represent asynchronous method calls”.

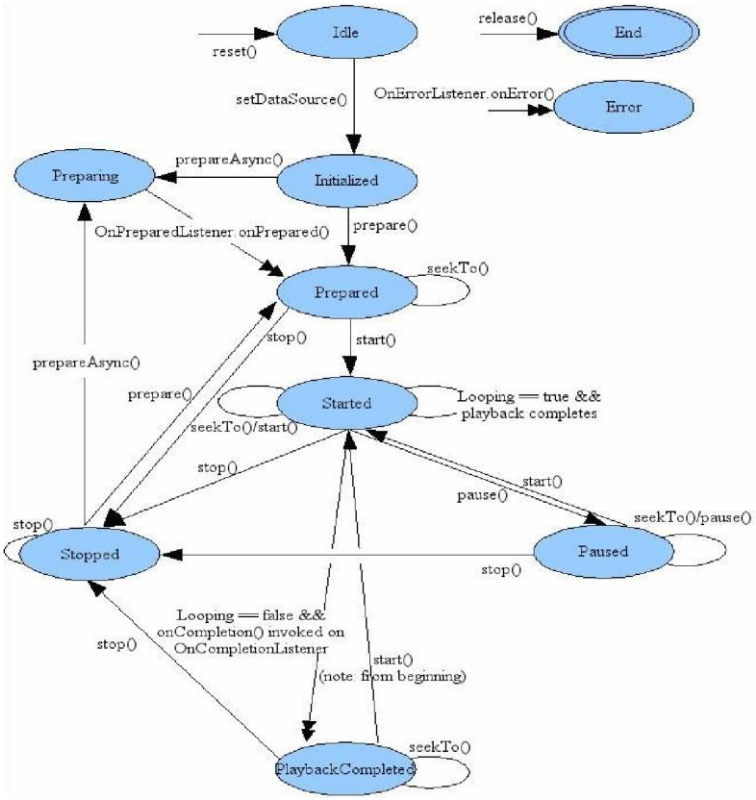


Figure 10.State Diagram of audio player

“From this state diagram, one can see that a MediaPlayer object has the following states”:

- “When a MediaPlayer object is just created using new or after reset() is called, it is in the Idle state; and after release() is called, it is in the

Endstate. Between these two states is the life cycle of the MediaPlayer object". o "There is a subtle but important difference between a newly constructed MediaPlayer object and the MediaPlayer object after reset() is called. It is a programming error to invoke methods such"

"as getCurrentPosition(), getDuration(), getVideoHeight(), getVideoWidth(),setAudioStrea mType(int), setLooping(boolean), setVolume(float, float), pause(), start(), stop(), seekTo(int), prepare() orprepareAsync() in the Idle state for both cases. If any of these methods is called right after a MediaPlayer object is constructed, the user supplied

callback method OnErrorListener.onError() won't be called"

"by the internal player engine and the object state remains unchanged; but if these methods are called right after reset(), the user supplied callback method OnErrorListener.onError() will be invoked by the internal player engine and the object will be transfered to the Error state".

- "It is also recommended that once a MediaPlayer object is no longer being used, call release() immediately so that resources used by the internal player engine associated with the MediaPlayer object can be released immediately. Resource may include singleton resources such as hardware acceleration components and failure to call release() may cause subsequent instances of MediaPlayer objects to fallback to software implementations or fail altogether. Once the MediaPlayer object is in the End state, it can no longer be used and there is no way to bring it back to any other state". o "Furthermore, the MediaPlayer objects created using new is in the Idle state, while those created with one of the overloaded convenient createmethods are NOT in the Idle state". fact, the objects are in the Prepared state if the creation using create method is successful.

- “In general, some playback control operation may fail due to various reasons, such as unsupported audio/video format, poorly interleaved audio/video, resolution too high, streaming timeout, and the like. Thus, error reporting and recovery is an important concern under these circumstances. Sometimes, due to programming errors, invoking a playback control operation in an invalid state may also occur. Under all these error conditions, the internal player engine invokes a user supplied OnErrorListener.onError() method if an OnErrorListener has been registered beforehand” via setOnErrorListener(android.media.MediaPlayer.OnErrorListener).

  - “It is important to note that once an error occurs, the MediaPlayer object enters the Error state (except as noted above), even if an error listener has not been registered by the application”.
  - “In order to reuse a MediaPlayer object that is in the Error state and recover from the error, reset() can be called to restore the object to its Idlestate”.

  - “It is good programming practice to have your application register a OnErrorListener to look out for error notifications from the internal player engine”. o “IllegalStateException is thrown to prevent programming errors such as calling prepare(), prepareAsync(), or one of the overloadedsetDataSource methods in an invalid state”.

- “Calling setDataSource(FileDescriptor), or setDataSource(String), or setDataSource(Context, Uri) or setDataSource(FileDescriptor, long, long), or setDataSource(FileDescriptor, long, long), or setDataSource(MediaDataSource) transfers a MediaPlayer object in the Idle state to the Initialized state”. o “An

IllegalStateException is thrown if setDataSource() is called in any other state".

- "It is good programming practice to always look out for IllegalArgumentException and IOException that may be thrown from the overloadedsetDataSource methods".

- "A MediaPlayer object must first enter the Prepared state before playback can be started".
  - "There are two ways (synchronous vs. asynchronous) that the Prepared state can be reached: either a call to prepare() (synchronous) which transfers the object to the Prepared state once the method call returns, or a call to prepareAsync() (asynchronous) which first transfers the object to the Preparing state after the call returns (which occurs almost right way) while the internal player engine continues working on the rest of preparation work until the preparation work completes. When the preparation completes or when prepare() call returns, the internal player engine then calls a user supplied callback method, onPrepared() of the OnPreparedListener interface, if an OnPreparedListener is registered beforehand via setOnPreparedListener(android.media.MediaPlayer.OnPreparedListener)".
  - "It is important to note that the Preparing state is a transient state, and the behavior of calling any method with side effect while a MediaPlayer object is in the Preparing state is undefined". o "An IllegalStateException is thrown if prepare() or prepareAsync() is called in any other state". o "While in the Prepared state, properties

such as audio/sound volume, screenOnWhilePlaying, looping can be adjusted by invoking the corresponding set methods".

- "To start the playback, start() must be called. After start() returns successfully, the MediaPlayer object is in the Started state. isPlaying() can be called to test whether the MediaPlayer object is in the Started state". o "While in the Started state, the internal player engine calls a user supplied OnBufferingUpdateListener.onBufferingUpdate() callback method if a OnBufferingUpdateListener has been registered beforehand via setOnBufferingUpdateListener(OnBufferingUpdateListener). This callback allows applications to keep track of the buffering status while streaming audio/video".

  - Calling start() has not effect on a MediaPlayer object that is already in the Started state.

- "Playback can be paused and stopped, and the current playback position can be adjusted. Playback can be paused via pause(). When the call topause() returns, the MediaPlayer object enters the Paused state. Note that the transition from the Started state to the Paused state and vice versa happens asynchronously in the player engine. It may take some time before the state is updated in calls to and it can be a number of seconds in the case of streamed content". and it can be a number of seconds in the case of streamed content". o "Calling start() to resume playback for a paused MediaPlayer object, and the resumed playback position is the same as where it was paused. When the call to start() returns, the paused MediaPlayer object goes back to the Started state".

  - Calling pause() has no effect on a MediaPlayer object that is already in the Paused state.
- "Calling stop() stops playback and causes a MediaPlayer in the Started, Paused, Prepared or PlaybackCompleted state to enter the Stopped state".

- “Once in the Stopped state, playback cannot be started until prepare() or prepareAsync() are called to set the MediaPlayer object to the Preparedstate again”. o “Calling stop() has no effect on a MediaPlayer object that is already in the Stopped state”.

- The playback position can be adjusted with a call to seekTo(int).

  - “Although the asynchronuous seekTo(int) call returns right way, the actual seek operation may take a while to finish, especially for audio/video being streamed. When the actual seek operation completes, the internal player engine calls a user supplied OnSeekComplete.onSeekComplete() if an OnSeekCompleteListener has been registered beforehand via setOnSeekCompleteListener(OnSeekCompleteListener)”.

  - “Please note that seekTo(int) can also be called in the other states, such as Prepared, Paused and PlaybackCompleted state”.
  - “Furthermore, the actual current playback position can be retrieved with a call to getCurrentPosition(), which is helpful for applications such as a Music player that need to keep track of the playback progress”.

- When the playback reaches the end of stream, the playback completes.

  - “If the looping mode was being set to truewith setLooping(boolean), the MediaPlayer object shall remain in the Started state”.

  - “If the looping mode was set to false , the player engine calls a user supplied callback method, OnCompletion.onCompletion(), if a OnCompletionListener is registered beforehand via setOnCompletionListener(OnCompletionListener). The invoke of the callback signals that the object is now in the PlaybackCompleted state”.

interface	“MediaStore.Audio.Albu mColumns Columns representing an album”
Class	“MediaStore.Audio. Albums Contains artists for audio files”
interface	“MediaStore.Audio.ArtistColum ns Columns representing an artist”
Class	“MediaStore.Audio.Artists Contains artists for audio files”
interface	“MediaStore.Audio.AudioColumns Columns for audio file that show up in multiple tables.”
Class	“MediaStore.Audio.Genres Contains all genres for audio files”
interface	“MediaStore.Audio.GenresColumns Columns representing an audio genre”
Class	“MediaStore.Audio.Media”
Class	“MediaStore.Audio.Playlists Contains playlists for audio files”
Classes for audio player	

interface	“MediaStore.Audio.PlaylistsColum ns Columns representing a playlist”
Class	“MediaStore.Audio.Radio”
Public constructors	
“MediaStore.Audio()”	
Public methods	
staticString	“keyFor(String name)” “Converts a name to a "key" that can be used for grouping, sorting and searching.”

Features : -

1.Equalizer

“An Equalizer is used to alter the frequency response of a particular music source or of the main output mix”.

“An application creates an Equalizer object to instantiate and control an Equalizer engine in the audio framework. The application can either simply use predefined presets or have a more precise control of the gain in each frequency band controlled by the equalizer”.

“The methods, parameter types and units exposed by the Equalizer implementation are directly mapping those defined by the OpenSL ES 1.0.1

Specification

(http://www.khronos.org/opensles/) for the SLEqualizerItf interface. Please refer to this specification for more details".

"To attach the Equalizer to a particular AudioTrack or MediaPlayer, specify the audio session ID of this AudioTrack or MediaPlayer when constructing the
Equalizer".

creating an equalizer

```
Equalizer equalizer = new Equalizer(0,mediaplayer.getAudioSessionId());
equalizer.setEnabled(true);
equalizer.getNumberOfBands();
```

```
    float volu
me =
1; float speed = 0.05f;
```

Classes for equalizer	
interface	Equalizer.OnParameterChangeListener The OnParameterChangeListener interface defines a method called by the Equalizer when a param
Class	Equalizer.Settings The Settings class regroups all equalizer parameters.

Control

```
package com.grifball.info;

import com.grifball.info.ShakeDetector.OnShakeListener;

import android.app.Activity;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.media.MediaPlayer;
import android.os.Bundle;

public class HammerActivity extends Activity {

private ShakeDetector mShakeDetector;
private SensorManager mSensorManager;

// Declare the MediaPlayer object
private MediaPlayer mMediaPlayer;

@Override
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.hammer_page);

   // ShakeDetector initialization
   mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

```
mShakeDetector = new ShakeDetector();

mShakeDetector.setOnShakeListener(new ShakeDetector.OnShakeListener() {
   public void onShake() {
      // Initialize media player
      mMediaPlayer = MediaPlayer.create(HammerActivity.this, R.raw.hammer);

      // Add OnCompletionListener to release the
      mMediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
         @Override
         public void onCompletion(MediaPlayer mp) {
            mMediaPlayer.release();
}
      });
      mMediaPlayer.start();
 });
 };

 @Override
 protected void onResume() {
  super.onResume();
  mSensorManager.registerListener(mShakeDetector,
     mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
     SensorManager.SENSOR_DELAY_UI);
 }

 @Override
 protected void onPause() {
  mSensorManager.unregisterListener(mShakeDetector);
  super.onStop();
 }}
```

This is my ShakeDetector code.

```
package com.grifball.info;

import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;

/**Listener that detects shake gesture.
 */
public class ShakeDetector implements SensorEventListener {
```

```
/** Minimum movement force to consider. */
private static final int MIN_FORCE = 10;


/**Minimum times in a shake gesture that the direction of movement needs tochange.
 */
private static final int MIN_DIRECTION_CHANGE = 3;

/** Maximum pause between movements. */
private static final int MAX_PAUSE_BETHWEEN_DIRECTION_CHANGE = 200;

/** Maximum allowed time for shake gesture. */
private static final int MAX_TOTAL_DURATION_OF_SHAKE = 400;

/** Time when the gesture started. */
private long mFirstDirectionChangeTime = 0;

/** Time when the last movement started. */
private long mLastDirectionChangeTime;

/** How many movements are considered so far. */
private int mDirectionChangeCount = 0;

/** The last x position. */
private float lastX = 0;

/** The last y position. */
private float lastY = 0;

/** The last z position. */
private float lastZ = 0;

/** OnShakeListener that is called when shake is detected. */
private OnShakeListener mShakeListener;
```

OnCompletionListener code. package
com.grifball.info;

```
import android.media.MediaPlayer; public class OnCompletionListener {

public void onCompletion(MediaPlayer mp) {
// TODO Auto-generated method stub

}

}
```

2. Sleep timer

- Create one service, which is going to be used for countdown of time.
- By default android broadcasts one intent call ACTION_TIME_TICK at every minute Register that intent in your service.
- Increment your count at every minute.

```
public int onStartCommand (Intent intent, int flags, int startId)
{

    context.registerReceiver(new TickReceiver(), new
ntentFilter(Intent.ACTION_TIME_TICK));




    public class TickReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
    if(intent.getAction().compareTo(Intent.ACTION_TIME_TICK) == 0){
        count++;
        if(count==5 && (PlayActivity.mediaPlayer!=null) ){
           PlayActivity.mediaPlayer.stop();
           PlayActivity.mediaPlayer.release();
        }
    }
}
```

PlayActivity.java

```
static MediaPlayer mediaPlayer;




@Override

public void onCreate(){

super.onCreate();

     mediaPlayer=new MediaPlayer();
```

```
mediaPlayer.setDataSource(YOUR_PATH);
mediaPlayer.prepare();
mediaPlayer.exception();
Intent ServiceIntent=new Intent(this,TimeCounterService.class);
StartService(ServiceIntent);
```

.

3.4 System Requirements For Android Studio

WINDOWS

- "Microsoft® Windows® 8/7/Vista (32 or 64-bit)".
- "2 GB RAM minimum, 4 GB RAM recommended".
- "400 MB hard disk space".
- "At least 1 GB for Android SDK, emulator system images"
- 1280 x 800 minimum screen resolution

4. Add to Favorites

```
import android.content.SharedPreferences;

import android.os.Bundle;

import android.widget.Button;

import android.widget.Toast;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

    private Button favoriteButton;

    private boolean isFavorite = false;

    private String songId;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
```

```
        songId = "song_1"; // Replace with actual song ID

        favoriteButton = findViewById(R.id.favoriteButton);

        isFavorite = isFavorite(songId);

        favoriteButton.setActivated(isFavorite);

        favoriteButton.setOnClickListener(v -> {

            isFavorite = !isFavorite;

            favoriteButton.setActivated(isFavorite);

            saveFavoriteStatus(songId, isFavorite);

            Toast.makeText(MainActivity.this, isFavorite ? "Added to
Favorites" : "Removed from Favorites", Toast.LENGTH_SHORT).show();

        });

    }



    private boolean isFavorite(String songId) {

        SharedPreferences sharedPreferences =
getSharedPreferences("Favorites", MODE_PRIVATE);
```

```
        return sharedPreferences.getBoolean(songId, false);

    }

    private void saveFavoriteStatus(String songId, boolean isFavorite) {

        SharedPreferences sharedPreferences =
getSharedPreferences("Favorites", MODE_PRIVATE);

        SharedPreferences.Editor editor = sharedPreferences.edit();

        editor.putBoolean(songId, isFavorite);

        editor.apply();

    }

}
```

5. Search

```
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.ListView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    private EditText searchEditText;
    private ListView listView;
    private ArrayAdapter<String> adapter;
    private List<String> songs;

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

        searchEditText = findViewById(R.id.searchEditText);
        listView = findViewById(R.id.listView);
```

```
        // Initialize your song list (songs) here

        adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, songs);
        listView.setAdapter(adapter);

        searchEditText.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {}

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                adapter.getFilter().filter(s);
            }

            @Override
            public void afterTextChanged(Editable s) {}
        });
    }
}
```

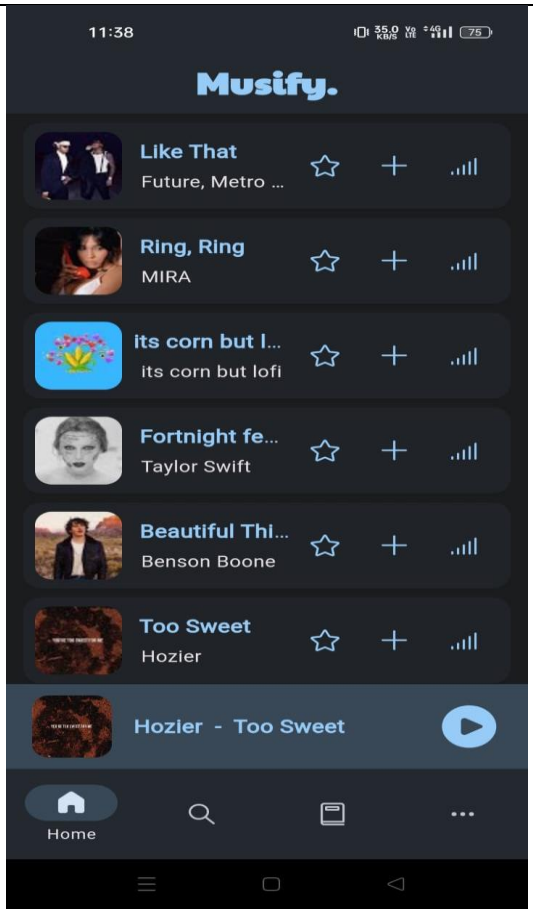
6. Add to Play List

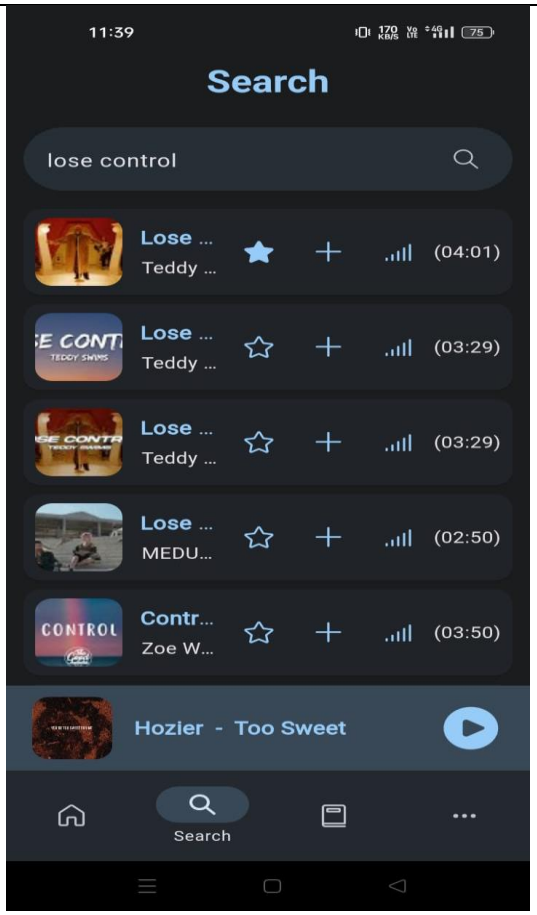
```
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;

import androidx.appcompat.app.AppCompatActivity;

import java.util.ArrayList;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    private ListView listView;
    private ArrayAdapter<String> adapter;
    private List<String> playlist;

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

        listView = findViewById(R.id.listView);
        Button addToPlaylistButton = findViewById(R.id.addToPlaylistButton);
```

```
        // Initialize your playlist (playlist) here

        adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1, playlist);
        listView.setAdapter(adapter);

        addToPlaylistButton.setOnClickListener(v -> {
            // Add the selected song to the playlist
            String selectedSong = "Selected Song"; // Replace with the selected song
            playlist.add(selectedSong);
            adapter.notifyDataSetChanged();
        });
    }
}
```

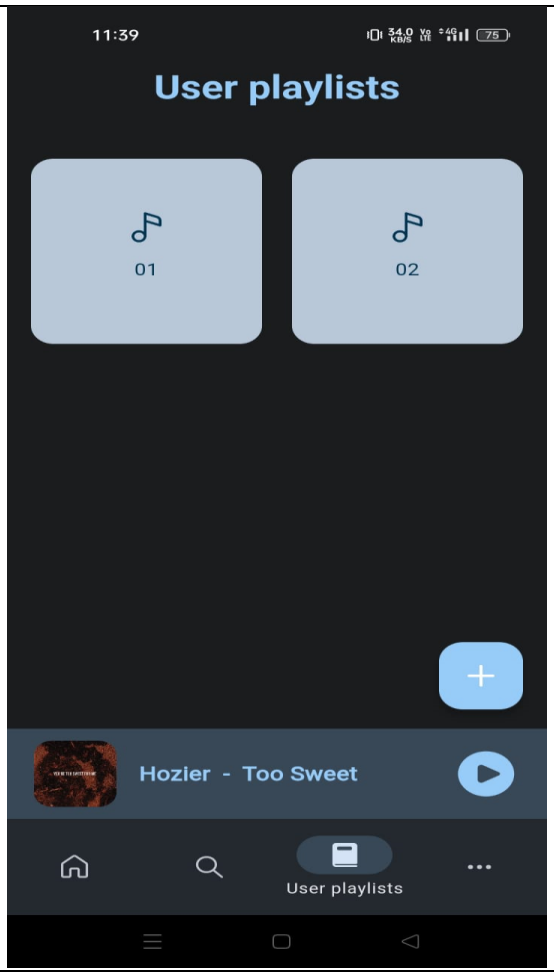
CHAPTER 4: PERFORMANCE ANALYSIS

A) Screenshots of Conference Application






11:38
Now playing
YOU'RE TOO SWEET FOR ME
Too Sweet
Hozier
00:52
04:11


11:39
User playlists
01
02
Hozier - Too Sweet
User playlists

CHAPTER 5: CONCLUSION

"Android as a full, open and free mobile device platform, with its powerful function and good user experience rapidly developed into the most popular mobile operating system. This report gives an overview of the different challenges and issues faced in android app development The experience of developing an android app is quite challenging, motivating as well as satisfying".

"This report shows an approach for designing of media player. Media player should consider the improvement in scenario such as decode efficiency needs to be improved, synchronization between multiple media streams, and display of the original data. Use of FFmpeg decode library seems to be an alternative method. Research shows FFmpeg supports most media formats which gives a high decode efficiency. Different approaches that can be considered are plug-in extension technology, multimedia based on hierarchy, media player based on file browser, media player based on FFmpeg, media player based on file server, etc". There is a vast scope of improvement in this field.

CHAPTER 6: REFRENCES

[1] Ma, Li, Lei Gu, and Jin Wang. "Research and Development of Mobile Application for android Platform." (2014).

[2] Liu, Jianye, and Jiankun Yu. "Research on Development of android Applications."

Fourth International conference on Intelligent Networks and Intelligent Systems. 2011.

[3] Peng, Bin, Jinming Yue, and Chen Tianzhou. "The android Application Development College Challenge." High Performance Computing and

Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on. IEEE, 2012

[4] Parada, Abilio G., and Lisane B. de Brisolara. "A model driven approach for android applications development." Computing System Engineering (SBESC), 2012 Brazilian Symposium on. IEEE, 2012.

[5] Nikhil S. Sakhare , R. W. Jasutkar. “Design of Android based Media Player”.

International Journal of Science and Research (IJSR), India Online ISSN: 23197064, February 2013.

[6] Amit M. Farkade, Miss. Sneha. R. Kaware. “The Android - A Widely

Growing Mobile Operating System With its Mobile based Applications” International Journal of Computer Science and Mobile Applications, Vol.3 Issue.

1, January- 2015, pg. 39-45