Music Player In Android

Project Report submitted in partial fulfillment of the requirement for the degree of

Bachelor of Technology.

in

Computer Science & Engineering

under the Supervision of

Mr. Arvind Kumar

Assistant Professor, Dept. of CSE

By

Shubham Kajaria

Enrollment No: 111294

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled "Music Player In Android", submitted by Shubham Kajaria in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science & Engineering to Jaypee University of Information Technology, Waknaghat, Solan has been carried out under my supervision.

This work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

Date:

Mr. Arvind Kumar
(Assistant Professor)

Acknowledgement

I would like to express my gratitude to all those who gave me the possibility to do this project. I want to thank the Department of CSE & IT in JUIT for giving us the permission to commence this project in the first instance, to do the necessary research work.

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success. I am grateful to my project guide Mr. Arvind Kumar for the guidance, inspiration and constructive suggestions that helpful me in the preparation of this project. I also thank myfaculty who have helped in successful completion of the project.

Date: **Shubham Kajaria(111294)**

Abstract

This software is used for android operating system, it can be used easily for playing music and it's convenient and quick , use simple UI and unique menu can give the users perfect experience.

This report gives an idea on how to design a music player based on Android OS. The music player, which uses the front-back end architecture, is divided into the part of music playback and the part of player interface and music list.

The Android platform provides resources for handling media playback, which your apps can use to create an interface between the user and their music files. Through this report, we will create a basic music player application for Android. The app will present a list of songs on the user device, so that the user can select songs to play.

Application will be written using Android SDK in Java and should run on all Android OS handsets.

The music player runs stably and conveniently during testing.

Table of Content

S. No.	Topic	Page No.
1.	Motivation	1
2.	Introduction to Android	2
2.1	The birth of Android	3
2.1.1	Open Handset alliance found	3
2.1.2	Hardware	3
2.2	Features of android OS	4
2.3	Interface Of android	4
2.4	Sensors In Android devices	5
3.	Literature Review	6
3.1	Literature Survey	6
3.1.1	Research Paper1	6
3.1.2	Research paper 2	7

4.		Android architecture	8
	4.1	Application framework	8
	4.2	Libraries	9
	4.3	Android Runtime	10
	4.4	Android Versions	11
	4.5	Symbol of Android OS	12
5.		Security and permissions in Android	13
	5.1	Development Tools	14
	5.2	Android Emulator	14
	5.3	Memory management of Android OS	15
6.		Design	16
	6.1	Android Development Life Cycle	17
	6.2	State Diagram	18
	6.3	Requirements	19
	6.4	Level-0 DFD	20
	6.5	Level-1 DFD	21

	6.6	System Chart	22
	6.7	Flow Chart	23
	6.8	Building Blocks of Android Code	24
	6.9	Screenshots	25
7.		Implementation	27
8.		Result	33
	8.1	Notification panel	33
	8.2	On pressing fast forward button	34
	8.3	On releasing fast forward button	35
	8.4	On pressing record button	36
	8.5	On pressing play button	37
	8.6	On pressing stop button	38
	8.7	No album cover	39
9.		Future Enhancements & Conclusion	40

List of Figures

S.No.		Title	Page No.
1.	Fig 2.3.1:	Interface of Android	5
2.	Fig 4.1:	Android Architecture	8
3.	Fig 4.3.1:	Android Runtime	10
4.	Fig 4.4.1:	Android Version Distribution	11
5.	Fig 4.5.1:	Symbol of Android OS	12
6.	Fig 6.1:	Design	16
7.	Fig 6.1.1:	Android Development Lifecycle	17
8.	Fig 6.2.1:	State Diagram	18
9.	Fig 6.4.1:	Level-0 DFD	20
10.	Fig 6.5.1:	Level-1 DFD	21
11.	Fig 6.6.1:	System Chart	22
12.	Fig 6.7.1:	Flow Chart	23
13.	Fig 6.8.1:	Building blocks of Android code	24

14.	Fig 6.9.1:	Screenshot1	25
15.	Fig 6.9.2:	Screenshot 2	26
16.	Fig 8.1:	Notification panel	33
17.	Fig 8.2:	On pressing fast forward button	34
18.	Fig 8.3:	On releasing fast forward button	35
19.	Fig 8.4:	On pressing record button	36
20.	Fig 8.5:	On pressing play button	37
21.	Fig 8.6:	On pressing stop button	38
22.	Fig 8.7:	Default album cover	39

Chapter 1: Motivation

Use of **Android OS** with more public interest and make it more user friendly so all can use it. As today's market speaks lot of android and lots of app are being built on it and most common of them all is music player application. Everyone loves to listen music but apart from a good music, a good music player is required which will enhance the quality of played music.

So, by making one of those player I will add it in Google Play Store.

Chapter 2: Introduction to Android

Android is an operating system based on the Linux , and designed primarily for touch screen mobile devices such as Smartphone's and tablet computers. The first Android-powered phone was sold in October 2008.

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 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. Applications can further send notifications to the user to inform them of relevant information, such as new emails and text messages.

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.

Android's open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which add new features for advanced usersor bring Android to devices which were officially released running other operating systems.

Android has a large community of developers writing applications that extend the functionality of devices, written primarily in the Java programming language

Android is popular with technology companies who require a ready-made, low-cost, customizable and lightweight operating system for high tech devices.

2.1 THE BIRTH OF ANDROID

Google Acquires Android Inc. In July 2005, Google acquired Android Inc., a small startup company based in Palo Alto, CA. Android's co-founders who went to work at Google included Andy Rubin (co-founder of Danger), Rich Miner (co-founder of Wildfire Communications, Inc), Nick Sears (once VP at T-Mobile), and Chris White (one of the first engineers at WebTV). At the time, little was known about the functions of Android Inc. other than they made software for mobile phones.

2.1.1 Open Handset Alliance Founded

On 5 November 2007, the Open Handset Alliance, a consortium of several companies which include Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, Sprint Nextel and NVIDIA, was unveiled with the goal to develop open standards for mobile devices. Along with the formation of the Open Handset Alliance, the OHA also unveiled their first product, Android, an open source mobile device platform based on the Linux operating system.

2.1.2 Hardware

Google has unveiled at least three prototypes for Android, at the Mobile World Congress on February 12, 2008. One prototype at the ARM booth displayed several basic Google applications. A 'd-pad' control zooming of items in the dock with a relatively quick response.

2.2 Features Of Android OS

- Application framework enabling reuse and replacement of components
- Dalvik virtual machine optimized for mobile devices
- Integrated browser based on the open source Web Kit engine Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- SQLite for structured data storage
- Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM Telephony (hardware dependent)
- Bluetooth, EDGE, 3G, and WiFi (hardware dependent)
- Camera, GPS, compass, and accelerometer (hardware dependent)
- Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE.

2.3 Interface of Android

Android home screens are typically made up of app icons and widgets. A home screen may be made up of several pages that the user can swipe back and forth between, though Android's home screen interface is heavily customizable, allowing the user to adjust the look and feel of the device to their taste.

Present along the top of the screen is a status bar, showing information about the device and its connectivity. This status bar can be "pulled" down to reveal a notification screen where apps display important information or updates, such as a newly received email or SMS text, in a way that does not immediately interrupt or inconvenience the user.

Fig 2.3.1: Interface of Android

2.4 Sensors in Android Devices

Interesting for music making:

• Gyro Sensors

• Accelerometer

• Geomagnetic Sensors

• Touchscreen

• Microphone

• Camera

• Motion Sensors

• Environmental Sensors

• Position Sensors

• Gesture Sensor

Chapter 3: Literature Review

3.1 Literature Survey:

3.1.1 Paper 1: Research and Development of Mobile Application for Android Platform

Abstract:

Today, as the developing of hardware of mobile is getting better, the performance index is much higher than the actual requirements of the software configuration. Phone's features more depend on software. As the Android operating system is getting more popular, the application based on Android SDK attracts much more attention. But now, some of the Android application interface is too cumbersome, pop-up ads is overmuch and the function is too single, these cause some inconvenience to the users. This article presents the application by eliminating the redundancy. Three kinds of applications are developed base on Java and Android SDK --- Weibo client, video player and audio player. The audio player uses the ContentResolver and Curor to obtain music files and plays the music by using the Service Components to call the Media Player class in the background. The video player uses the Media Player class provided by Android SDK. This class loads the file through URL, realize the multimedia file parsing by calling the OpenCore Library, which is at the bottom of Android, through JNI and by calling the SurfaceFlinger interface to realize the video files' playback. The users' data is collected through the Sina open platform called by Sina client and the data will be returned under the format of JSON by the Sina server. The system uses the authentication method for user authorization to complete the login process. The specific functions of this system are developed based on Android Weibo SDK. The interfaces of these Android apps are pretty and the operation is smooth. What's more, the cumbersomeinterface and excessive advertising are eliminated, so that users are able to manipulate these apps more conveniently and smoothly[1].

Keywords: Android, Weibo client, Video Player, audio player, Android SDK

Introduction

In recent years, the emergence of smart phones has changed the definition of mobile phones. Phone is no longer just a communication tool, but also an essential part of the people's communication and daily life. Various applications added unlimited fun for people's lives. It is certain that the future of the network will be the mobile terminal. Now the Android system in the electronics market is becoming more and more popular, especially in the smart phone market. Because of the open source, some of the development tools are free, so there are plenty of applications generated

3.1.2 Paper 2: Design of Android based Media Player

Abstract:

Many users like to watch video by a mobile phone, but the media player has many limitations. With a rapid 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.

Keywords: media player, FFmpeg, file browser, file server.

Introduction

With the continuous development of science and technology, mobile phone is no longer just communication, but a multimedia platform that provides multimedia capabilities. Playing a video on media player becomes basic function, but the media player has many limitations since there’re limited format supported by media player. At present, the decode module of most of the media players is based on FF mpeg decode library which supports more than 90 kinds of decoders, such as Storm Codec, KMP Codec, etc and the display module is based on SDL (Simple DirectMedia Layer) . This paper shows different approaches for design of media player. First is plug-in extension technology on android multimedia player software platform. Second is media player based on hierarchy

Chapter 4: Android Architecture

The following diagram shows the major components of Android. [3]

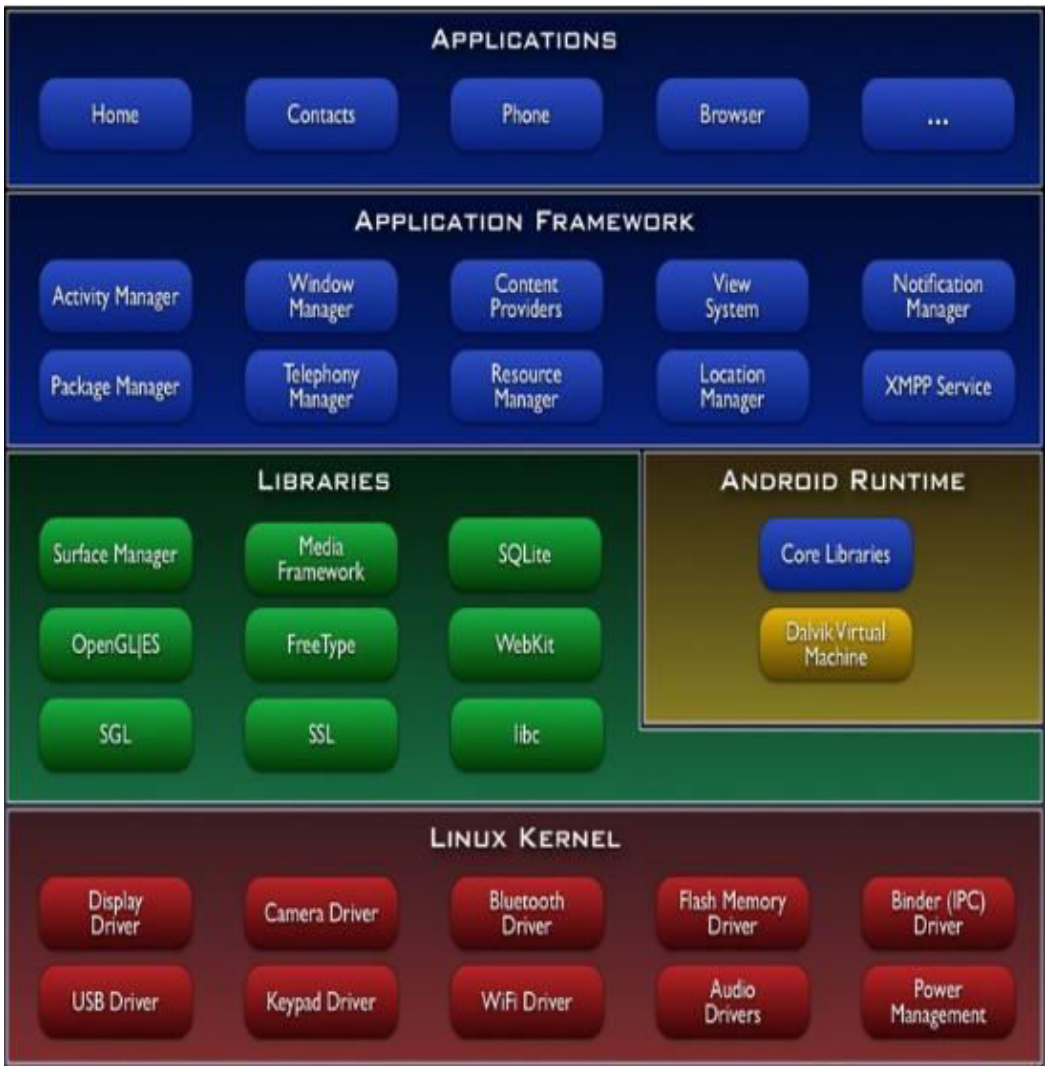

Fig 4.1: Architecture of Android OS

4.1 Application Framework

Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application

may then make use of those capabilities (subject to security constraints enforced by the framework).

This same mechanism allows components to be replaced by the user. Underlying all applications is a set of services and systems, including: A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data ·A Resource Manager, providing access to non-code resources such as localized strings, graphics, and lat files.

A Notification Manager that enables all applications to display custom alerts in the status bar An Activity Manager that manages the life cycle of applications and provides a common navigation backstack.

4.2 Libraries

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework.

Some of the core libraries are listed below: System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices Media Libraries - based on PacketVideo's Open CORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG.

Surface Manager - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications

LibWebCore - a modern web browser engine which powers both the Android browser and an embeddable web view

SGL - the underlying 2D graphics engine

3D libraries - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer

Free Type - bitmap and vector font rendering

SQLite - a powerful and lightweight relational database engine available to all applications.

4.3 Android Runtime

Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. The VM is register-based, and runs classes compiled by a Java language compiler that have been transformed into the .dex format by the included "dx" tool. The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

At the same level there is Android Runtime, where the main component Dalvik Virtual Machine is located. It was designed specifically for Android running in limited environment, where the limited battery, CPU, memory and data storage are the main issues. Android gives an integrated tool “dx”, which converts generated byte code from .jar to .dex file, after this byte code becomes much more efficient to run on the small processors.

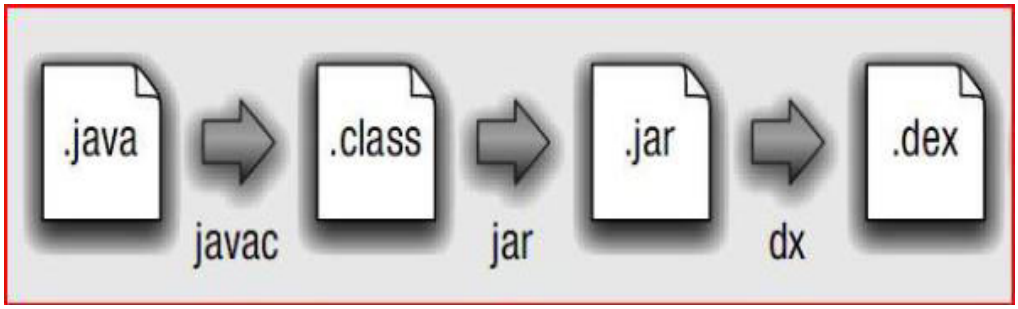

Fig 4.3.1: Conversion from .java to .dex file

As the result, it is possible to have multiple instances of Dalvik virtual machine running on the single device at the same time. The Core libraries are written in Java language and contains of the collection classes, the utilities, IO and other tools.

4.4 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 Eclair (API level 5)

Android 2.1 Eclair (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 4.3 Jelly Bean (API level 18)

Android 4.4 KitKat (API level 19)

Android 5.0 Lollipop(API level 21)

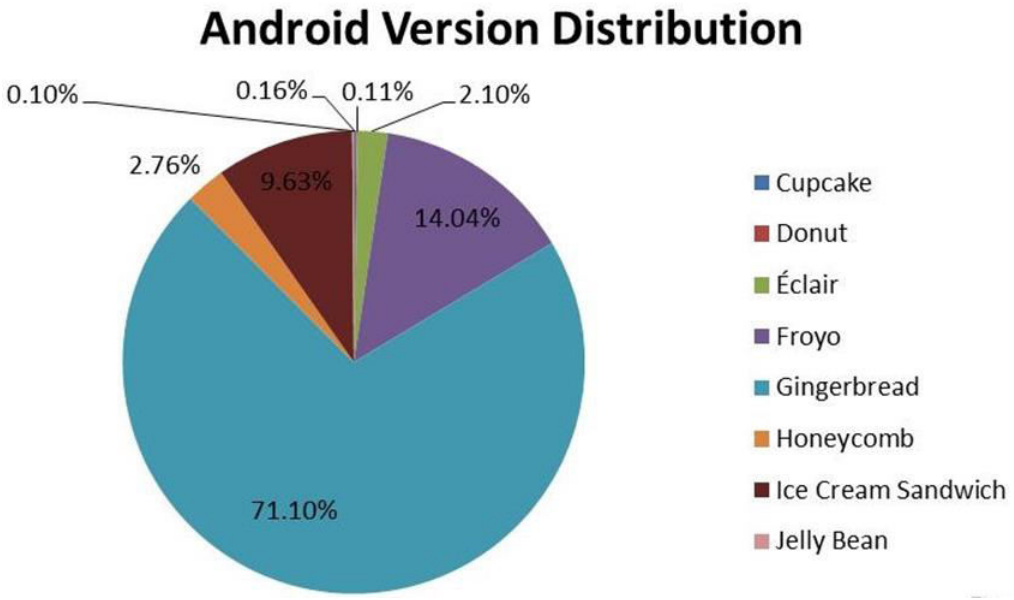

Fig 4.4.1: Android version Distribution

4.5 Symbol OF Android OS

Fig 4.5.1: Symbol of Android OS

Chapter 5: Security and permissions in Android

Android is a multi-process system, where each application (and parts of the system) runs in its own process. Most security between applications and the system is enforced at the process level through standard Linux facilities, such as user and group IDs that are assigned to applications. Additional finer-grained security features are provided through a "permission" mechanism that enforces restrictions on the specific operations that a particular process can perform.

Android mobile phone platform is going to be more secure than Apple's iPhone or any other device in the long run. There are several solutions nowadays to protect Google phone from various attacks. One of them is security vendor McAfee, a member of Linux Mobile (LiMo) Foundation. This foundation joins particular companies to develop an open mobile-device software platform. Many of the companies listed in the LiMo Foundation have also become members of the Open Handset Alliance (OHA).

As a result, Linux secure coding practice should successfully be built into the Android development process. However, open platform has its own disadvantages, such as source code vulnerability for black-hat hackers. In parallel with great opportunities for mobile application developers, there is an expectation for exploitation and harm. Stealthy Trojans hidden in animated images, particular viruses passed from friend to friend, used for spying and identity theft, all these threats will be active for a long run.

Another solution for such attacks is SMobile Systems mobile package. Security Shield –an integrated application that includes anti-virus, anti-spam, firewall and other mobile protection is up and ready to run on the Android operating system. Currently, the main problem is availability for viruses to pose as an application and do things like dial phone numbers, send text messages or multi-media messages or make connections to the Internet during normal device use. It is possible for somebody to use the GPS feature to track a person's location without their knowledge. Hence SMobile Systems is ready to notify and block these secure alerts. But the truth is that it is not possible to secure r mobile device or personal computer completely, as it connects to the internet. And neither the Android phone nor other devices will prove to be the exception. [5]

5.1 Development Tools

The Android SDK includes a variety of custom tools that help develop mobile applications on the Android platform. The most important of these are the Android Emulator and the Android Development Tools plugin for Eclipse, but the SDK also includes a variety of other tools for debugging, packaging, and installing r applications on the emulator.

5.2 Android Emulator

A virtual mobile device that runs on computer use the emulator to design, debug, and test r applications in an actual Android run-time environment.

Android Development Tools Plugin for the Eclipse IDE.The ADT plugin adds powerful extensions to the Eclipse integrated environment, making creating and debugging r Android applications easier and faster. If use Eclipse, the ADT plugin gives an incredible boost in developing Android applications:

It gives access to other Android development tools from inside the Eclipse IDE. For example, ADT lets access the many capabilities of the DDMS tool — taking screenshots, managing port-forwarding, setting breakpoints, and viewing thread and process information — directly from Eclipse.

It provides a New Project Wizard, which helps quickly create and set up all of the basic files'll need for a new Android application. It automates and simplifies the process of building r Android application.

It provides an Android code editor that helps write valid XML for r Android manifest and resource files.

5.3 Memory Management of Android OS

Since Android devices are usually battery-powered, Android is designed to manage memory (RAM) to keep power consumption at a minimum.

When an Android app is no longer in use, the system will automatically suspend it in memory – while the app is still technically "open," suspended apps consume no resources (e.g. battery power or processing power)

Android manages the apps stored in memory automatically: when memory is low, the system will begin killing apps and processes that have been inactive for a while, in reverse order since they were last used (i.e. oldest first).

This process is designed to be invisible to the user, such that users do not need to manage memory or the killing of apps themselves. [6]

Chapter 6: Design

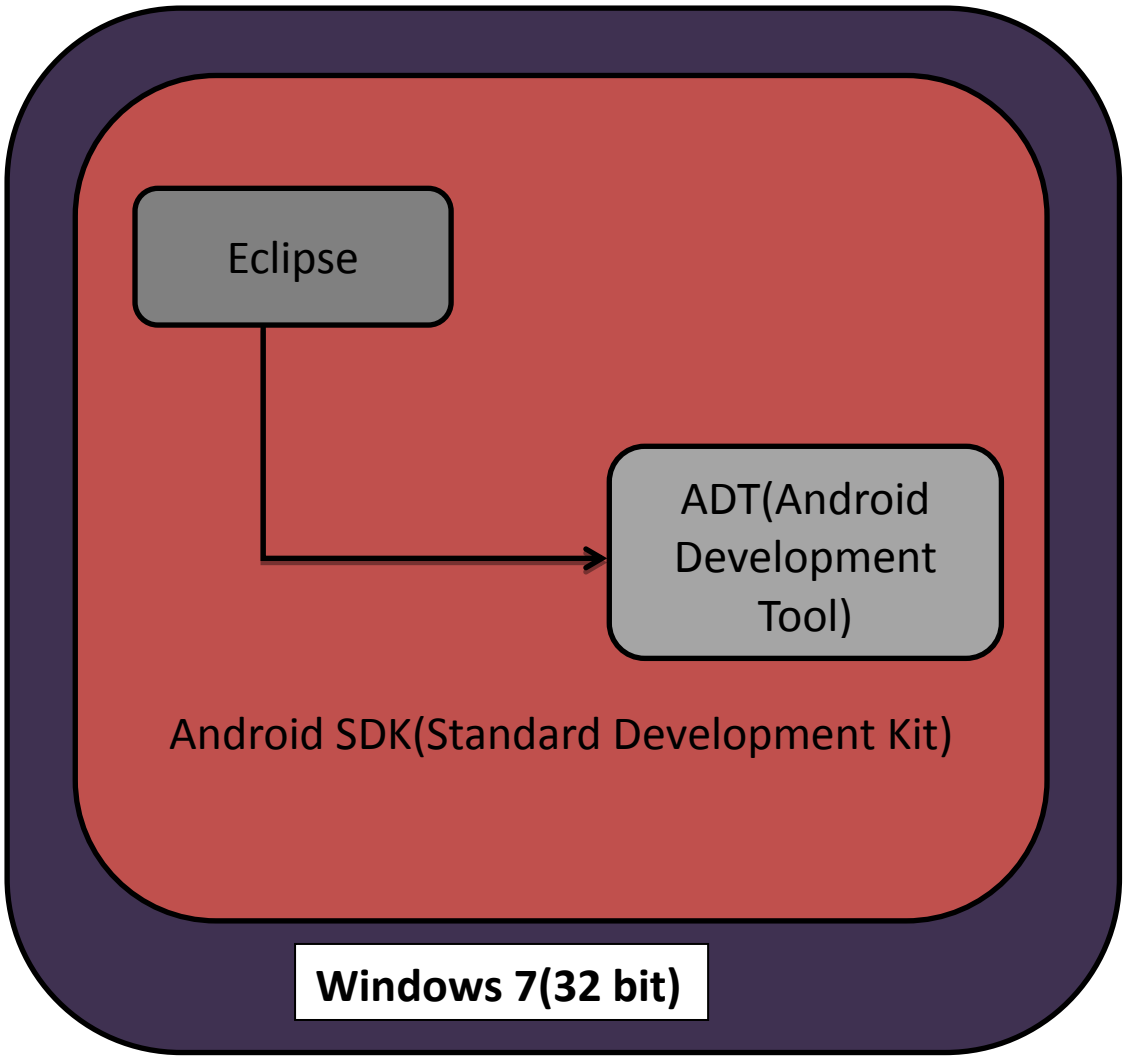

Fig 6.1: Design

6.1 Android Development Lifecycle

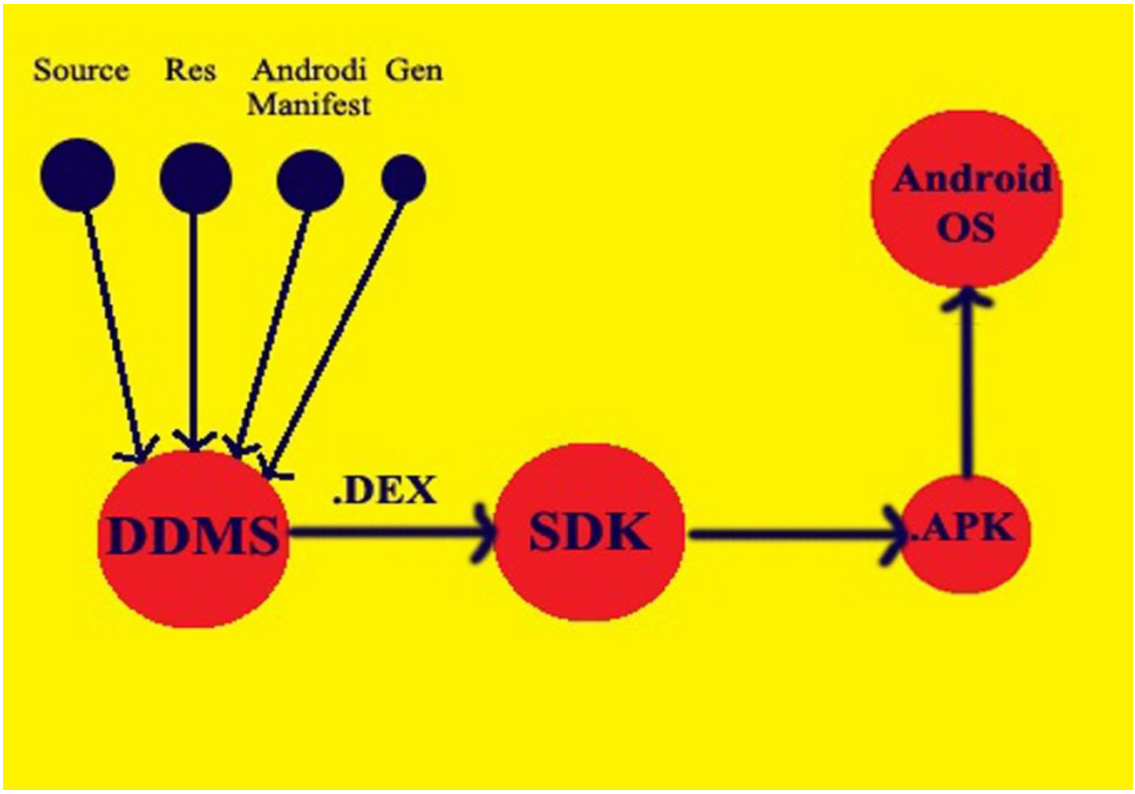

Fig 6.1.1: Android Development Life Cycle

Android ships with a debugging tool called the Dalvik Debug Monitor Server (DDMS), which provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process, and radio state information, incoming call and SMS spoofing, location data spoofing, and more. The Android SDK includes a variety of tools that help you develop mobile applications for the Android platform. The tools are classified into two groups: SDK tools and platform tools. SDK tools are platform independent and are required no matter which Android platform you are developing on. The most important SDK tools include the Android SDK Manager (`android sdk`), the AVD Manager (`android avd`) the emulator (`emulator`), and the Dalvik Debug Monitor Server (`ddms`).

6.2 State Diagram

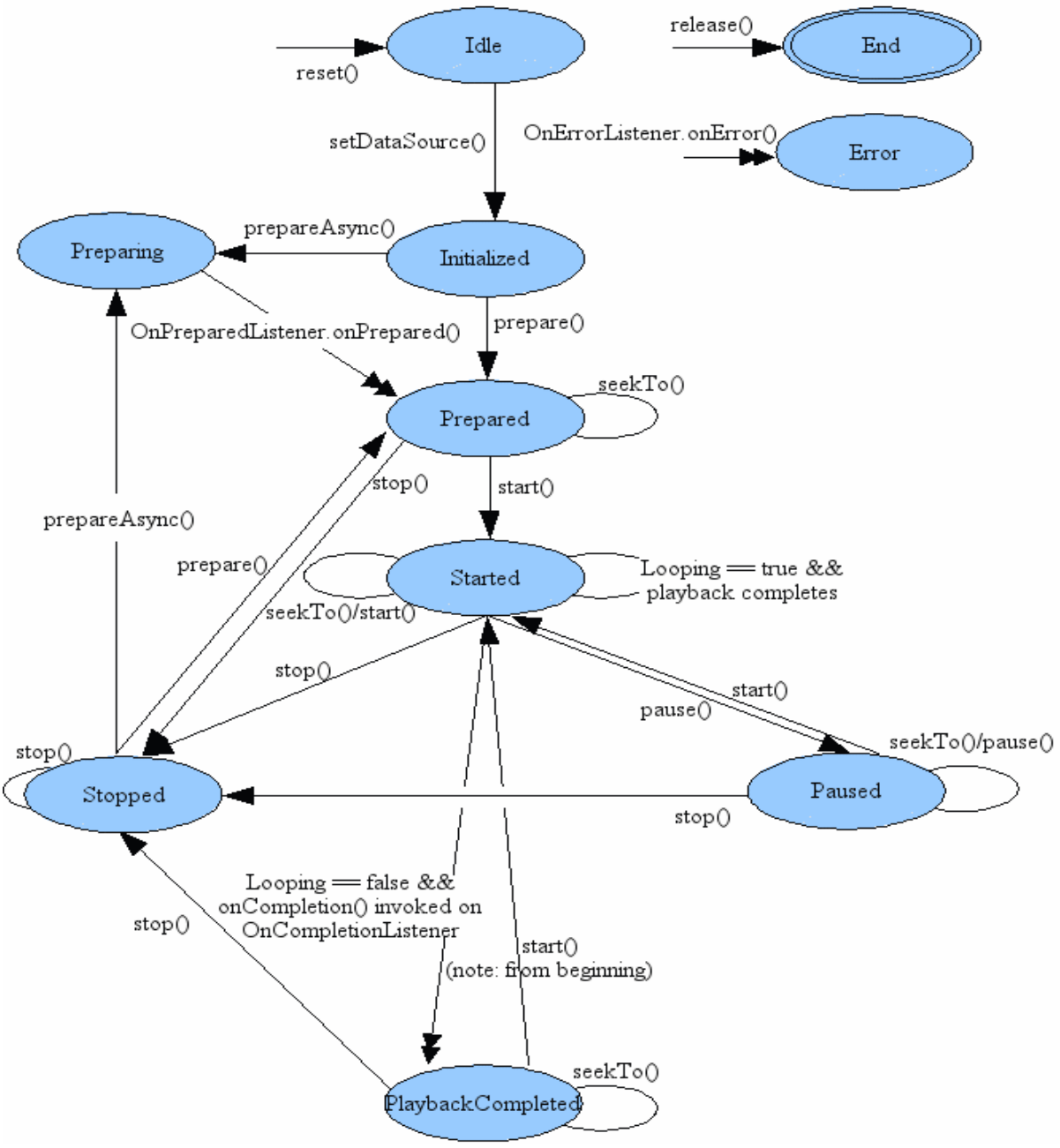

Fig 6.2.1:State Diagram

This diagram used to describe the behavior of systems. State diagrams are used to give an abstract description of the behavior of a system. This behavior is analyzed and represented in series of events, that could occur in one or more possible states. Hereby "each diagram usually represents objects of a single class and track the different states of its objects through the system".

6. 3 Requirements

Functional Requirements:.

- Android operating system on the Smartphone.
- The target device should be sound enabled.
- The android version should not be less than 2.3.5

External Interface Requirements:

User Interface Tested on:

- Android emulator version 4.3
- Samsung Duos mobile

Software Requirement:

- Android SDK Manager
- Eclipse
- ADT(Android Development Tool)

6.4 Level-0 DFD

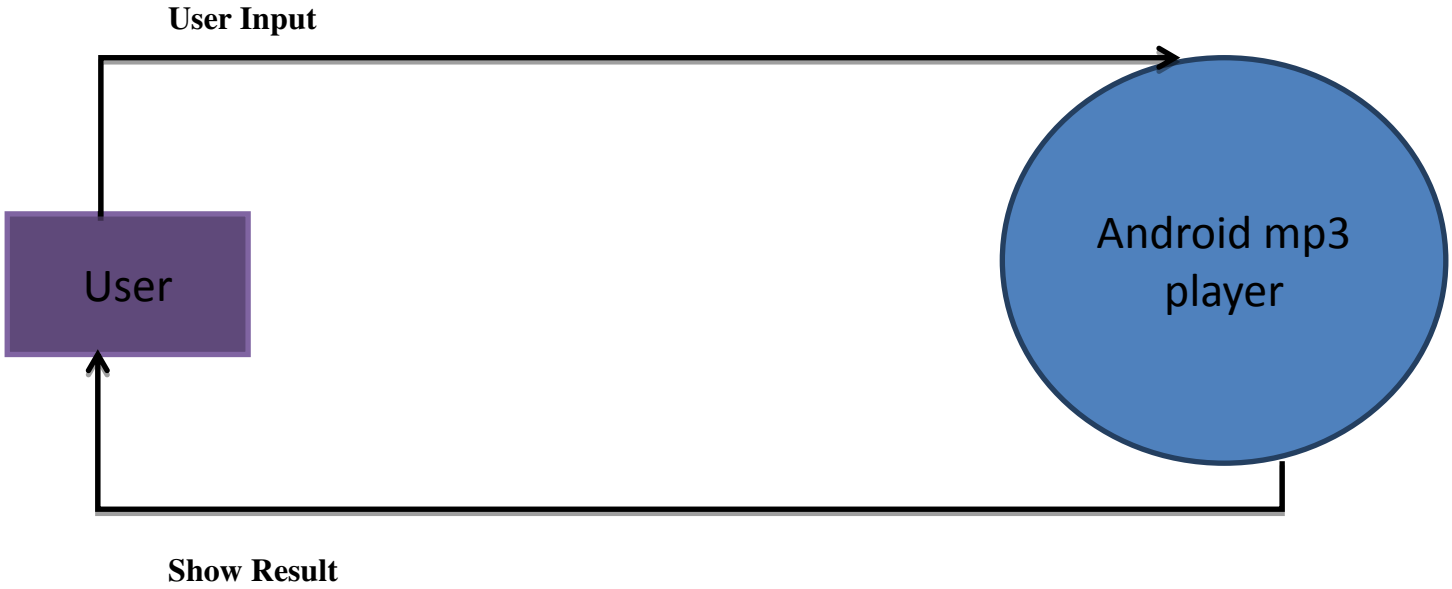

Fig 6.4.1: level-0 DFD

Here user will give input to the mp3 player and in turn player will play the requested music file as an output.

6.5 Level-1 DFD

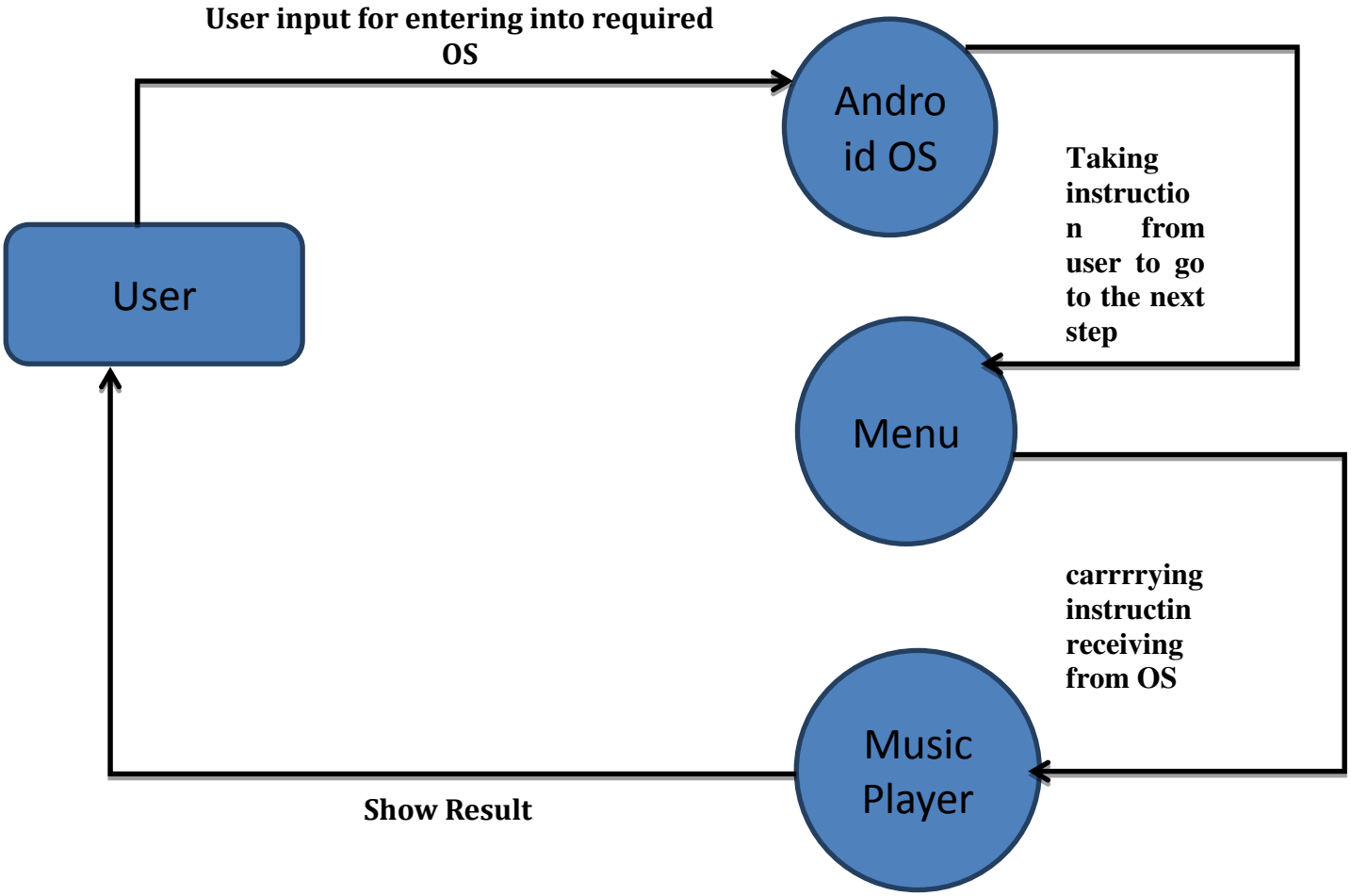

Fig 6.5.1: level-0 DFD

This is a detailed view of level-1 DFD where user will give instructions to mp3 player and inside that player, OS will take instructions and move to the next step and similarly requested file will be played as an output.

6.6 System Chart

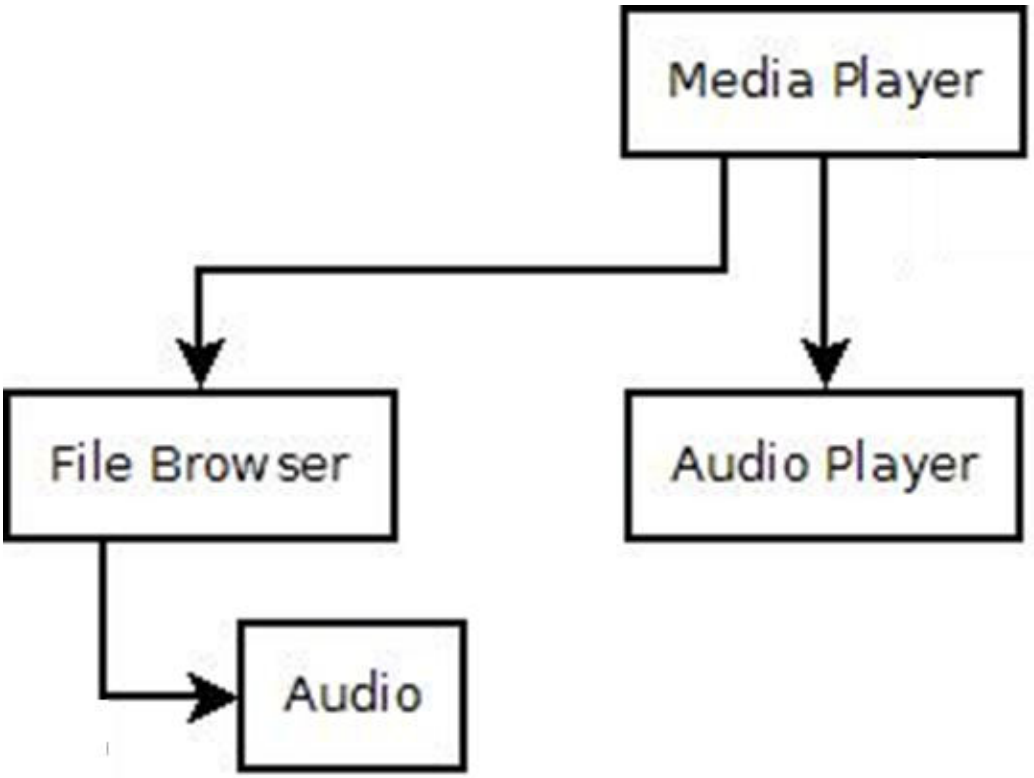

Fig 6.6.1: State Chart

This chart shows the breakdown of a system to its lowest manageable levels like MediaPlayer here is broken down into file browser and audio player.

6.7 Flow Chart

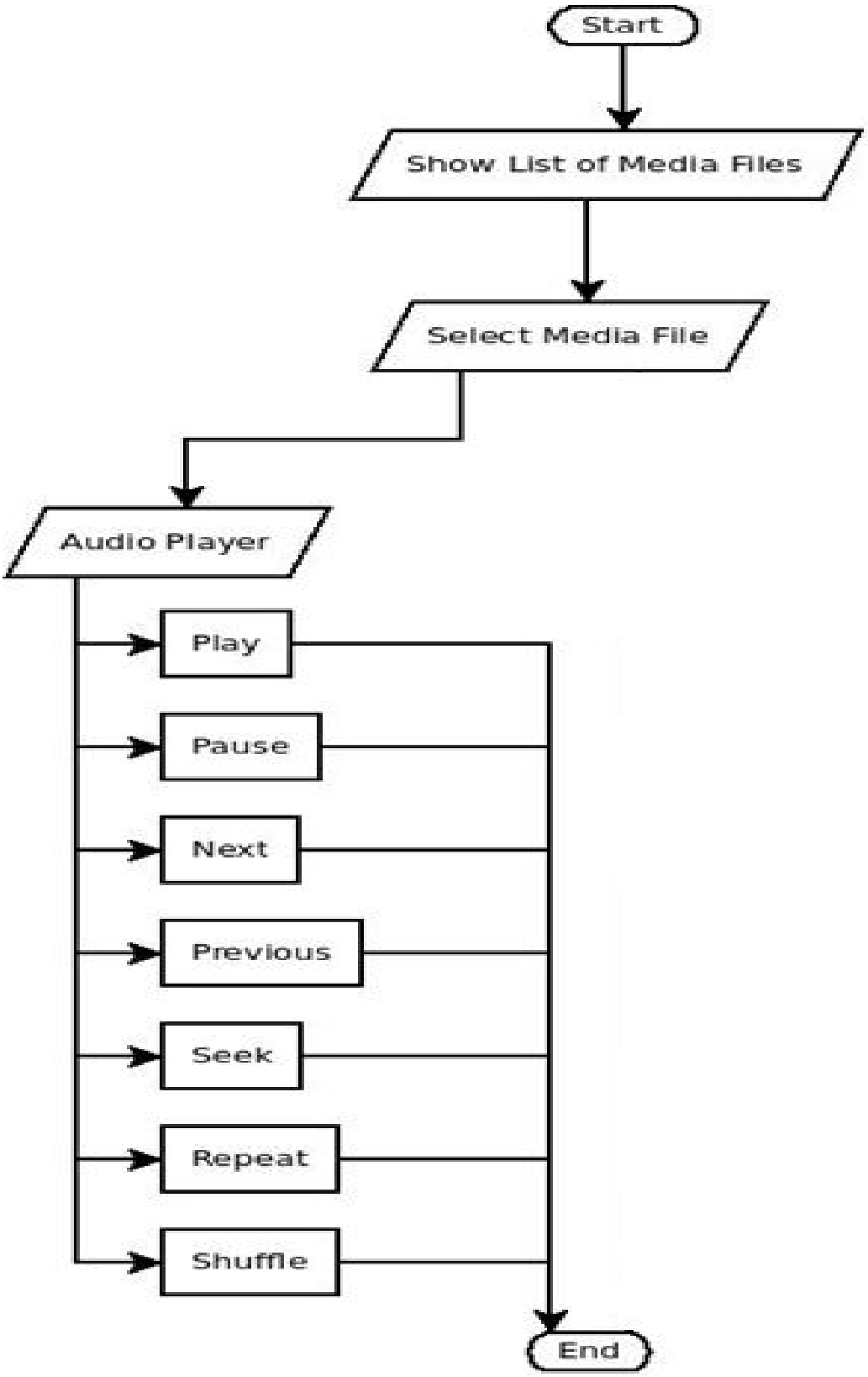

Fig 6.7.1: Flow Chart

This diagram that represents workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows.

6.8 Building Blocks of Android code

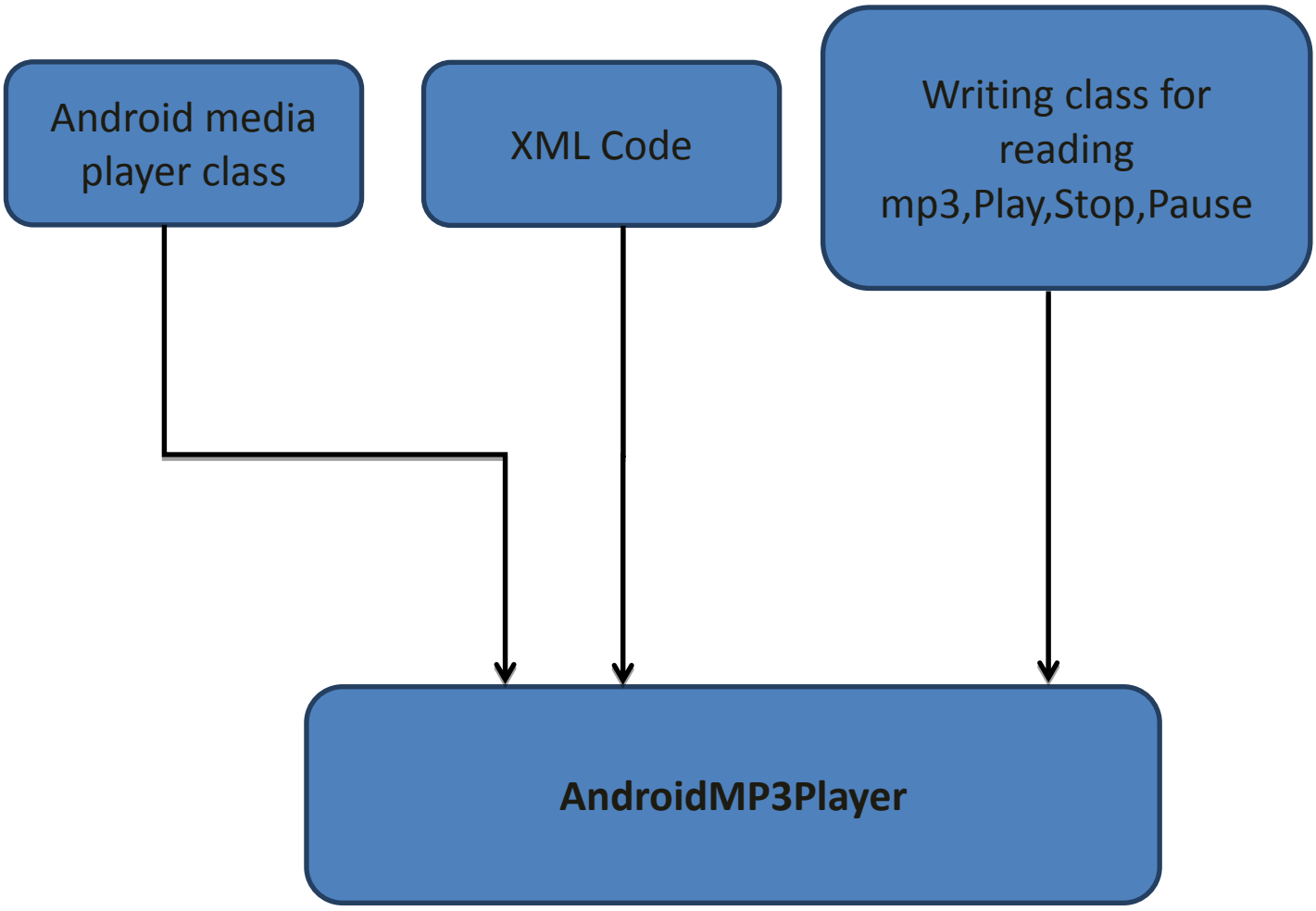

Fig 6.8.1: Building Blocks of Android Code

MediaPlayer class, XML code and various classes and functions for writing the code for various operations like play/pause, stop, rewind, etc. are the major building blocks for this music player.

6.9 Screenshots:

There are several songs which can be played by the user.

Fig 6.9.1: Screenshot1

After being played any song, screen goes into the next intent where several operations can be applied to it.

Fig 6.9.2: Sscreenshot2

Chapter 7: Implementation

Music player has several widgets and one of those is button. Buttons has to be created as per the requirement of music player application and giving their identifications (ids) through xml file created in the project.

```
Button record, rewind, fastfwd, pause, stop;

record=(Button) findViewById(R.id.button1);
pause=(Button) findViewById(R.id.button2);
rewind=(Button) findViewById(R.id.button3);
fastfwd=(Button) findViewById(R.id.button4);
stop=(Button) findViewById(R.id.button5);
```

After the button has been set, path has to be set where list of songs can be added i.e. from external sdcard.

```
private static final String sd_path=new String("/storage/extSdCard/");
	.
```

Before playing any song updateplaylist() will be called in order to add new songs to the list.

```
private void updatePlaylist() {
		// TODO Auto-generated method stub
		File home=new File(sd_path);
		if(home.listFiles(new Mp3Filter()).length>0)
				{
			for(File file :home.listFiles(new Mp3Filter()))
			{
				songs.add(file.getName());

			}

			//ListAdapter songList;
			ArrayAdapter<String> songList=new
ArrayAdapter<String>(this,R.layout.song_item,songs);
				setListAdapter(songList);

			}

	}
```

Various OnClickListener() on different buttons is to be set in order to perform a particular task like:- rewind.setOnClickListener(), fastforward.setOnClickListener(), etc.

```
pause.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View v) {
				// TODO Auto-generated method stub
				if(musicplayer.getCurrentPosition()!=0){
				if(i%2==0)
```

```
                        {
                        musicplayer.pause();
                        i++;
                        }
                        else if(i%2!=0)
                        {
                        musicplayer.start();
                        i++;
                        }

                        }
                        else if(musicplayer.getCurrentPosition()==0)
                        {
                              try{
                              musicplayer.reset();
                              musicplayer.setDataSource(abc);
                              musicplayer.prepare();
                              musicplayer.start();
                              updateseekbar.start();

      sb.setProgress(musicplayer.getCurrentPosition());

                              }
                              catch(Exception e)
                              {}

                        }
                        }
            });


rewind.setOnClickListener(new View.OnClickListener() {

      @Override
      public void onClick(View v) {
            if(i%2==0)
            {// TODO Auto-generated method stub
            int currentPosition = musicplayer.getCurrentPosition();
             if (currentPosition - seekBackwardTime >= 0) {
                   //d.setText("Rewinding...");
                  musicplayer.seekTo(currentPosition - seekBackwardTime);
             } else {
                  musicplayer.seekTo(0);
                  //d.setText("Rewinding...");
             }
      }
}
});


fastfwd.setOnClickListener(new View.OnClickListener() {

      @Override

      public void onClick(View v) {
            // TODO Auto-generated method stub
            if(i%2==0)
            {
            int currentPosition = musicplayer.getCurrentPosition();
```

```
        if (currentPosition + seekForwardTime <= musicplayer.getDuration())
{
            //c.setText("Forwarding...");
           musicplayer.seekTo(currentPosition + seekForwardTime);
        } else {
            //c.setText("Forwarding...");
           musicplayer.seekTo(musicplayer.getDuration());
        }

      }
}
});



stop.setOnClickListener(new View.OnClickListener() {

      @Override
      public void onClick(View v) {
            // TODO Auto-generated method stub
          musicplayer.seekTo(0);
          sb.setProgress(musicplayer.getCurrentPosition());
          cancelNotification();
          musicplayer.stop();
          i=0;
          TelephonyManager mgr = (TelephonyManager)
getSystemService(TELEPHONY_SERVICE);
          if(mgr != null) {

mgr.listen(phoneStateListener,PhoneStateListener.LISTEN_NONE);
          }
      }

      });


        // return true;



record.setOnClickListener(new View.OnClickListener() {

      @Override
      public void onClick(View v) {
            // TODO Auto-generated method stub
          if(record.getText().toString().equals("Rec."))
            {
                  try {
                        startRecord();
                  } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                  }
            record.setText("End");
            }

      else if(record.getText().toString().equals("End"))
      {
```

```
                stopRecord();
                record.setText("Play");
                }


        else if(record.getText().toString().equals("Play"))
        {

                try {
                        startPlayback();
                } catch (Exception e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                record.setText("Stop");
        }


        else if(record.getText().toString().equals("Stop"))
        {
                stopPlayback();
                record.setText("Rec.");
        }
                }
        });
```

Here MediaPlayer and MediaRecorder object has been created in order to play and record any music file. For showing the current time index of any song, SeekBar widget has been imported .Notification for each song being played is shown by importing Notification and NotificationManager class.

```
private void initNotification() {
                    //Song currSong = songs.get(position);

                  //get title and artist strings (Mapping the strings to
views in song Layout file)
                    //String songView=currSong.getTitle();

                 String ns = Context.NOTIFICATION_SERVICE;
                    NotificationManager mNotificationManager =
(NotificationManager) getSystemService(ns);
                    int icon = R.drawable.ic_launcher;
                    CharSequence tickerText = "Music In Play";
                    long when = System.currentTimeMillis();
                    Notification notification = new Notification(icon,
tickerText, when);
                    notification.flags = Notification.FLAG_ONGOING_EVENT;
                    Context context = getApplicationContext();
                    CharSequence contentTitle = "Music In play";
                    CharSequence contentText = "Listen To Music While
Performing Other Tasks";
                    Intent notificationIntent = new Intent(this,
MainActivity.class);
                    PendingIntent contentIntent =
PendingIntent.getActivity(context, 0,notificationIntent, 0);
```

```
notification.setLatestEventInfo(context, contentTitle, abc,contentIntent);
                mNotificationManager.notify(NOTIFICATION_ID,
notification);
            }

            // Cancel Notification
            private void cancelNotification() {
                String ns = Context.NOTIFICATION_SERVICE;
                NotificationManager mNotificationManager =
(NotificationManager) getSystemService(ns);
                mNotificationManager.cancel(NOTIFICATION_ID);
            }
```

If while listening a song any call comes on your phone or you have to make any call than automatically song will be paused and after making a call or receiving a call song will be played, this can be done by importing PhoneStateListener and TelephonyManager class.

```
final PhoneStateListener phoneStateListener = new PhoneStateListener() {
                @Override
                public void onCallStateChanged(int state, String
incomingNumber) {
                    if (state == TelephonyManager.CALL_STATE_RINGING) {
                        //Incoming call: Pause music
                         musicplayer.pause();
                         i++;
                    } else if(state == TelephonyManager.CALL_STATE_IDLE) {
                        //Not in call: Play music
                         musicplayer.start();
                    } else if(state ==
TelephonyManager.CALL_STATE_OFFHOOK) {
                        //A call is dialing, active or on hold
                    musicplayer.pause();
                    i++;
                    }
                    super.onCallStateChanged(state, incomingNumber);
                }
            };
            TelephonyManager mgr = (TelephonyManager)
getSystemService(TELEPHONY_SERVICE);
            if(mgr != null) {
                mgr.listen(phoneStateListener,
PhoneStateListener.LISTEN_CALL_STATE);
            }
```

Headset functionality is also implemented by importing broadcastReciever class.

```
// If headset gets unplugged, stop music and service.
            private BroadcastReceiver headsetReceiver = new
BroadcastReceiver() {
                private boolean headsetConnected = false;

                @Override
                public void onReceive(Context context, Intent intent) {
                    // TODO Auto-generated method stub
                    // Log.v(TAG, "ACTION_HEADSET_PLUG Intent
received");
```

```
                        if (intent.hasExtra("state")) {
                                if (headsetConnected &&
intent.getIntExtra("state", 0) == 0) {
                                        headsetConnected = false;
                                        headsetSwitch = 0;
                                        // Log.v(TAG, "State =  Headset
disconnected");
                                        // headsetDisconnected();
                                } else if (!headsetConnected
                                                &&
intent.getIntExtra("state", 0) == 1) {
                                        headsetConnected = true;
                                        headsetSwitch = 1;
                                        // Log.v(TAG, "State =  Headset
connected");
                                }

                        }

                        switch (headsetSwitch) {
                        case (0):
                               headsetDisconnected();
                               break;
                        case (1):
                               break;
                        }
                }

        };

        private void headsetDisconnected() {
               musicplayer.pause();
               i++;
        }
```

After clicking a particular song from the list, it will play that song and several other functionalities can be applied to it by using different buttons. Functionalities will be performed according to the task designated to the button like:-
Play/Pause, Record, Fast forward, Rewind a particular song. As the song is being played it will move into the next intent and album cover associated with that particular song will be shown else default gray layout will be set.

```
metaRetriever = new MediaMetadataRetriever();
                    metaRetriver.setDataSource(abc);
                    try {
                           art = metaRetriver.getEmbeddedPicture();
                           Bitmap songImage = BitmapFactory
.decodeByteArray(art, 0, art.length);
                           album_art.setImageBitmap(songImage);

}
                    catch (Exception e)
                    {
                    album_art.setBackgroundColor(Color.GRAY);
                      }
```

Chapter 8: Result

Notification panel will add automatically showing the path and name of the song after the song is played.

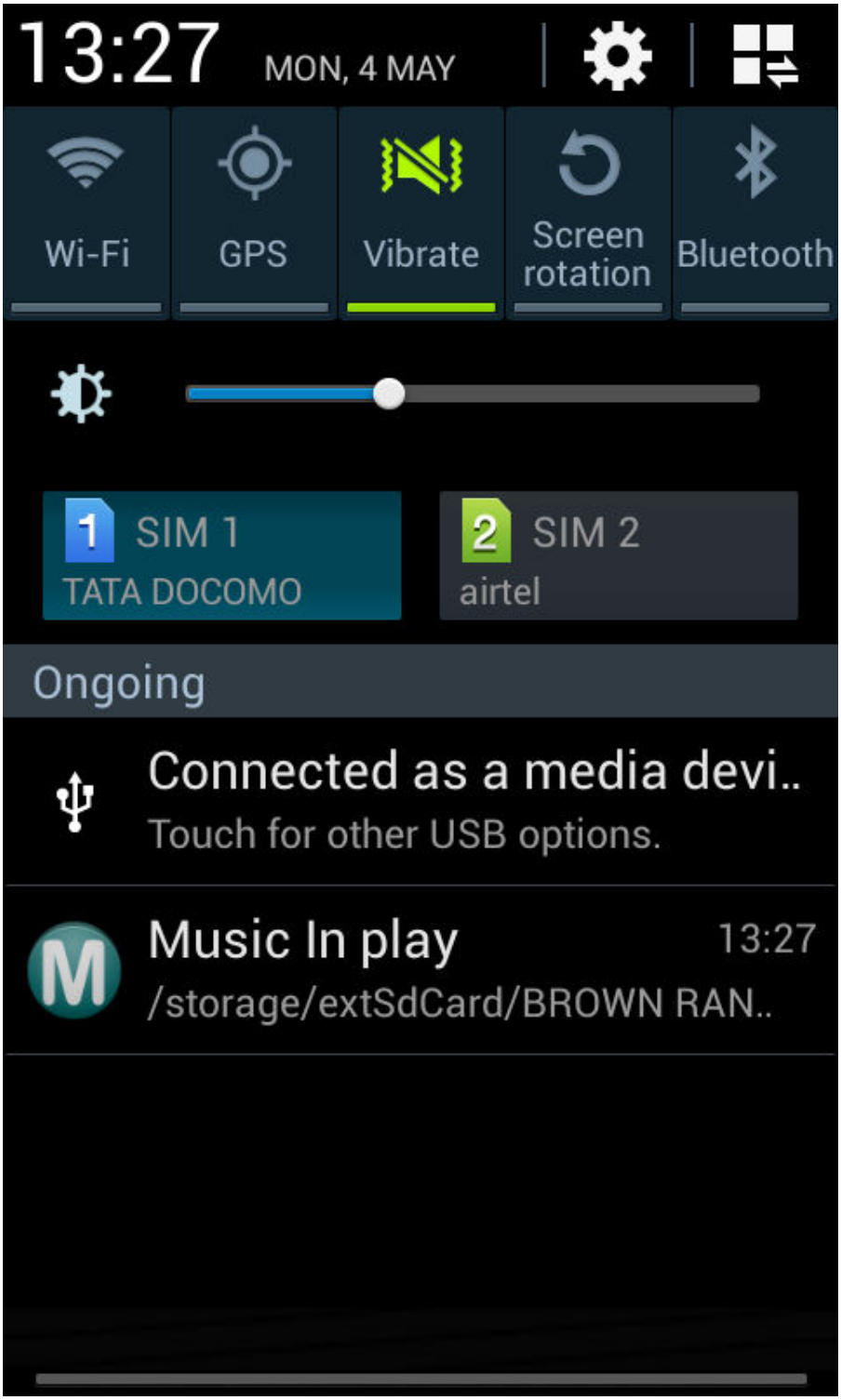

Fig 8.1: Notification panel

On pressing the fast forward button, the track moves forward by 5 sec, each time the user clicks on it.

Fig 8.2: On pressing fast forward button

On pressing fast forward button once the current track moves forward by 5 sec.

Fig 8.3: On releasing fast forward button

On pressing the record button, song starts getting recorded and in order to end that recording, end button is to be pressed.

Fig 8.4: On pressing Record button

On pressing the end button, play button will appear automatically and if you want to listen that recording, you can do so by clicking on that button.

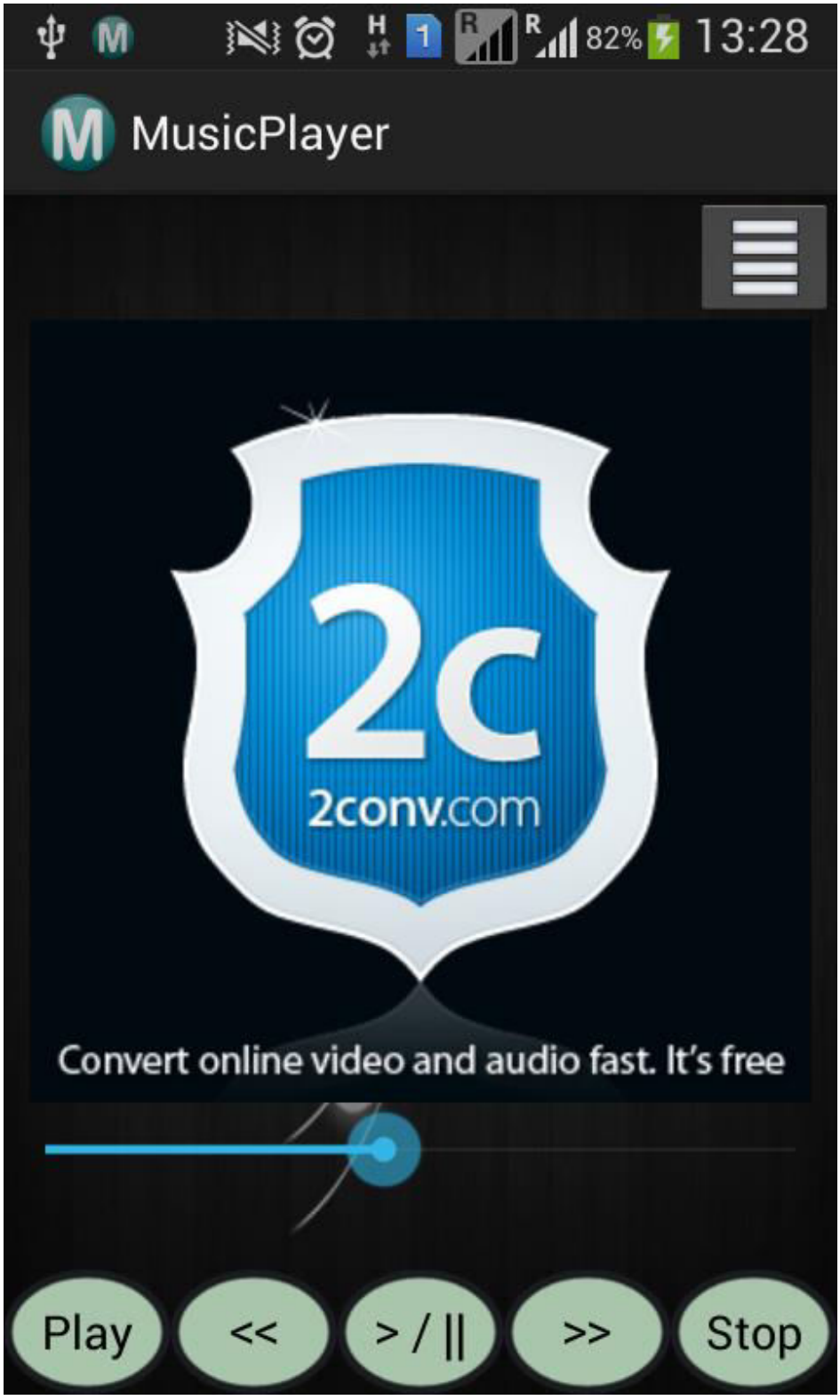

Fig 8.5: On pressing play button

In between if you want to stop listening the recording, you can simply press the stop button on the left side of the bottom of the screen.

Fig 8.6: On pressing stop button

If any song does not have any album cover than by default gray layout will be set else album cover associated with a particular song will be set.

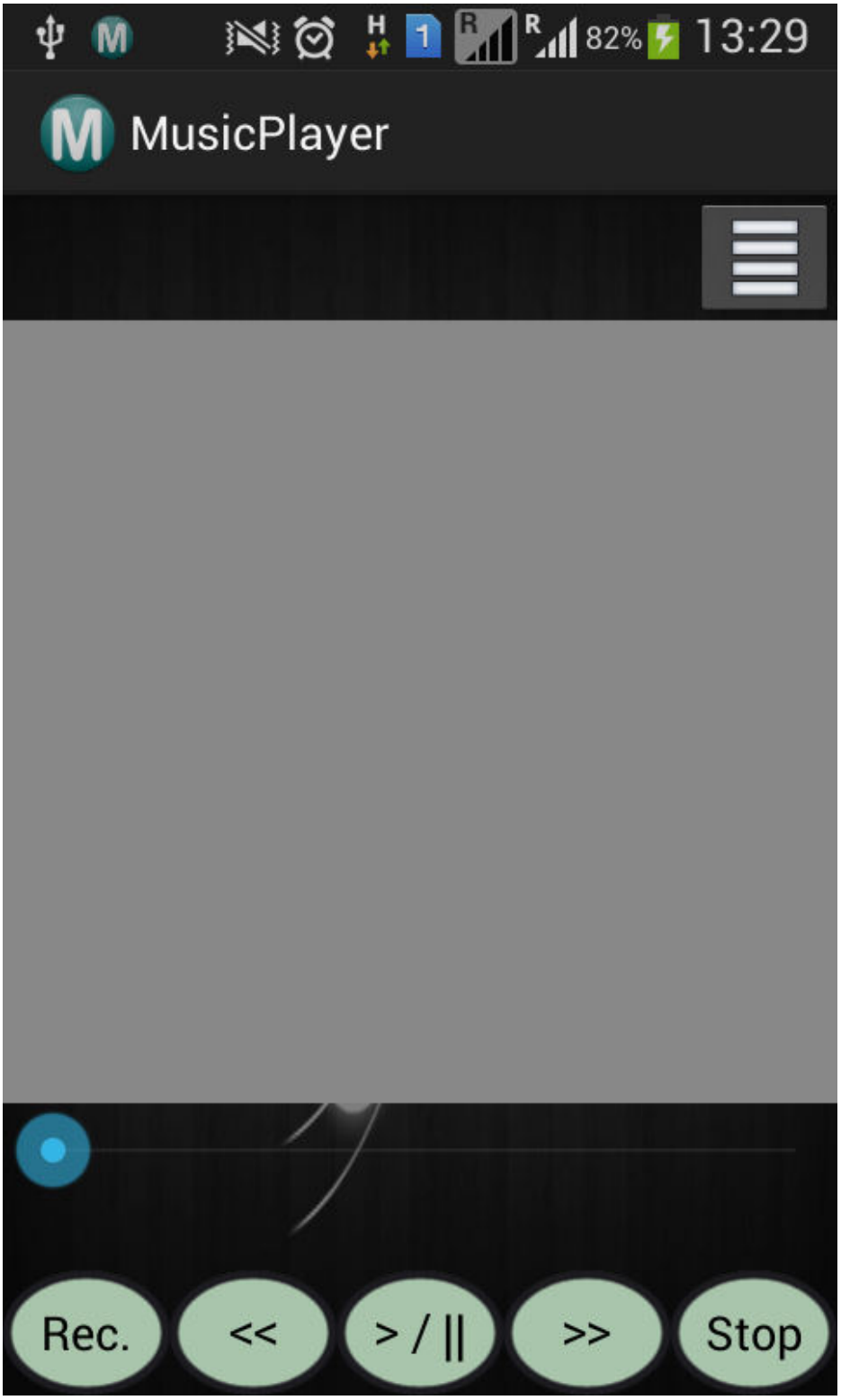

Fig 8.7: Default album cover

Chapter 9: Future Enhancements & Conclusion

Android is a truly open, free development platform based on Linux and open source. Handset makers can use and customize the platform without paying a royalty. Like most people, music is a huge part of everyone's life and our tastes are constantly changing based on how we feel or what we're doing .

Several future enhancements can be done in this player like:-

- Polish interface i.e. working on interface to make it more attractive and user friendly.
- Creating an additional field such as Star like songs in order to review later.
- Thumbs Up/ Thumbs Down for taking any song input.
- Add music visualizations and be able to share stations with a friends list.
- Ability to sync stations between friends so that the same song is played for both users with song selection taken from a mix of both users.
- Gesture recognition can also be added so that if you shake your android mobile phone, it will move to the next song in the current playlist.

This music player application in android will not only help the user to listen the music but also it will lead to a better listening with enhanced effects. This app has several functionalities beside play/pause like- recording, fast forwarding, rewinding, play/pause ,etc. It has also a catchy background layout along with album cover of each song, which will attract users to use it.

List of References Used:

[1] Li Ma, Lei Gu and Jin Wang, ” Research and Development of Mobile Application for Android Platform,” International Journal of Multimedia and Ubiquitous Engineering Vol.9, No.4 (2014), pp.187-198.

[2] Akshay R. Mukadam, Darshal N. Manchekar, Gaurav G. Panchal, Prasad P. Kanade, Atul Yadav, ”Android Based Media Player,” International Journal of Engineering Science and Innovative Technology (IJESIT) Volume 2, Issue 5, September 2013.

[3] Herbert Schildt, Complete Reference JAVA, 5th edition, Tata McGraw Hill.

[4] Android Programming: The Big Nerd Ranch Guide.

[5] Introduction to Android: http://developer.android.com/guide/index.html.

[5] Android Architecture:http://www.tutorialspoint.com/android_architecture.htm

[6] Application Framework: http://developer.android.com/guide/faq/framework.html

[7] Security Permissions:http://developer.android.com/train/articles/security-tips.html

[8] Memory Management:http://mobworld.wordpress.com/2010/07/05/memory-management-in-android/