Title Page

Design and Implementation of Android Music Player

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

Bachelor of Technology

in

Computer Science & Engineering

under the Supervision of

Mrs. Sanjana Singh

By

Pranav Chugh (101250)

to

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled "Design and Implementation of Android Music Player", submitted by "Pranav Chugh (101250)" 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: **Mrs. Sanjana Singh**

(Assistant Professor, Dept. of CSE)

Acknowledgement

I would like to express my gratitude to all those who helped me to do this project. I want to thank the Department of CSE and IT in JUIT for giving me the permission to commence this project and do the necessary research work.

I am thankful to my project guide Mrs. Sanjana Singh whose ceaseless cooperation and constant guidance and encouragement made it possible for me to complete this project.

Date: Pranav Chugh
101250

Table of Content

List of Figures

ABSTRACT

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

This report gives us 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.

CHAPTER 1: MOTIVATION

Android is the most popular mobile operating system. There are many separate apps in the Android Market that provide audio playback or online radio streaming. The aim of my project is to provide both these functionalities in a single app with audio filters and equalizer.

CHAPTER 2: LITERATURE REVIEW

1 Background to Android

1.4 Introduction

Android is a mobile operating system (OS) based on the Linux kernel and currently developed by Google. With a user interface based on direct manipulation, Android is designed primarily for touchscreen mobile devices such as smartphones and tablet computers, with specialized user interfaces for televisions (Android TV), cars (Android Auto), and wrist watches (Android Wear). The OS uses touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, and a virtual keyboard. Despite being primarily designed for touchscreen input, it also has been used in game consoles, digital cameras, regular PCs (e.g. the HP Slate 21) and other electronics.

Android is the most widely used mobile OS and, as of 2013, the highest selling OS overall. Android devices sell more than Windows, iOS, and Mac OS X devices combined, with sales in 2012, 2013 and 2014 close to the installed base of all PCs. As of July 2013 the Google Play store has had over 1 million Android apps published, and over 50 billion apps downloaded. A developer survey conducted in April–May 2013 found that 71% of mobile developers develop for Android. At Google I/O 2014, the company revealed that there were over 1 billion active monthly Android users, up from 538 million in June 2013.

Android’s source code is released by Google under open source licenses, although most Android devices ultimately ship with a combination of open source and proprietary software. Initially developed by Android, Inc., which Google backed financially and later bought in 2005, Android was unveiled in 2007 along with the founding of the Open Handset Alliance—a consortium of hardware, software, and telecommunication companies devoted to advancing open standards for mobile devices.

Android is popular with technology companies which require a ready-made, low-cost and customizable operating system for high-tech devices. 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 users or bring Android to devices which were officially released running other operating systems. The operating system’s success has made it a target for patent litigation as part of the so-called “smartphone wars” between technology companies.

1.2 Features

1.2.1 Interface

Android’s default user interface 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, and a virtual keyboard. The response to user input is designed to be immediate and provides a fluid touch interface, often using the vibration capabilities of the device to provide haptic feedback to the user. Internal hardware such as accelerometers, gyroscopes and proximity sensors are used by some applications to respond to additional user actions, for example adjusting the screen from portrait to landscape depending on how the device is oriented, or allowing the user to steer a vehicle in a racing game by rotating the device, simulating control of a steering wheel.

Android devices boot to the homescreen, the primary navigation and information point on the device, which is similar to the desktop found on PCs. Android homescreens are typically made up of app icons and widgets; app icons launch the associated app, whereas widgets display live, auto-updating content such as the weather forecast, the user’s email inbox, or a news ticker directly on the homescreen. A homescreen may be made up of several pages that the user can swipe back and forth between, though Android’s homescreen interface is heavily customizable, allowing the user to adjust the look and feel of the device to their tastes. Third-party apps available on Google Play and other app stores can extensively re-theme the homescreen, and even mimic the look of other operating systems, such as Windows Phone. Most manufacturers, and some wireless carriers, customize the look and feel of their Android devices to differentiate themselves from their competitors.

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. Notifications are persistent until read (by tapping, which opens the relevant app) or dismissed by sliding it off the screen. Beginning on Android 4.1, “expanded notifications” can display expanded details or additional functionality; for instance, a music player can display playback controls, and a “missed call” notification provides buttons for calling back or sending the caller an SMS message.

Android provides the ability to run applications which change the default launcher and hence the appearance and externally visible customize of Android. These appearance changes include a multi-page dock or no dock, and many more changes to fundamental features of the user interface.

1.2.2 Applications

Applications (“apps”), that extend the functionality of devices, are developed primarily in the Java programming language using the Android software development kit (SDK). The SDK includes a comprehensive set of development tools, including a debugger, software libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) plugin. Other development tools are available, including a Native Development Kit for applications or extensions in C or C++, Google App Inventor, a visual environment for novice programmers, and various cross platform mobile web applications frameworks. In January 2014, Google unveiled an Apache Cordova–based framework for porting Chrome HTML 5 applications to Android, wrapped in a native application shell.

Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application’s APK file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices. Google Play Store is the primary application store installed on Android devices that comply with Google’s compatibility requirements and license the Google Mobile Services software. Google Play Store allows users to browse, download and update applications published by Google and third-party developers; As of July 2013, there are more than one million applications available for Android in Play Store. As of May 2013, 48 billion applications have been installed from Google Play Store and in July 2013, 50 billion applications were installed. Some carriers offer direct carrier billing for Google Play application purchases, where the cost of the application is added to the user's monthly bill.

Due to the open nature of Android, a number of third-party application marketplace also exist for Android, either to provide a substitute for devices that are not allowed to ship with Google Play Store, provide applications that cannot be offered on Google Play Store due to policy violations, or for other reasons. Examples of these third-party stores have included the Amazon Appstore, GetJar, and SlideMe. F-Droid, another alternative marketplace, seeks to only provide applications that are distributed under free and open sourcelicenses.

1.2.3 Memory management

Since Android devices are usually battery-powered, Android is designed to manage memory (RAM) to keep power consumption at a minimum, in contrast to desktop operating systems which generally assume they are connected to unlimited mains electricity. When an Android application is no longer in use, the system will automatically suspend it in memory; while the application is still technically "open", suspended applications consume no resources (for example, battery power or processing power) and sit idly in the background until needed again. This brings a dual benefit by increasing the general responsiveness of Android devices, since applications do not need to be closed and reopened from scratch each time, and by ensuring that background applications do not consume power needlessly.

Android manages the applications stored in memory automatically: when memory is low, the system will begin killing applications and processes that have been inactive for a while, in reverse order since they were last used (oldest first). This process is designed to be invisible to the user, so that users do not need to manage memory or the killing of applications themselves. As of 2011, third-party task killers were reported by Lifehacker as doing more harm than good.

1.3 Hardware

The main hardware platform for Android is the ARM architecture (ARMv7 or later, Android 5.0 also supports ARMv8-A), with x86 and MIPS architectures also officially supported. Both 64-bit and 32-bit variants of all three architectures are supported since the release of Android 5.0. Since 2012, Android devices with Intel processors began to appear, including phones and tablets.

As of November 2013, Android 4.4 recommends at least 512 MB of RAM, while for "low RAM" devices 340 MB is the required minimum amount that does not include memory dedicated to various hardware components such as the baseband processor. Android 4.4 requires a 32-bit ARMv7, MIPS or x86 architecture processor (latter two through unofficial ports), together with an OpenGL ES 2.0 compatible graphics processing unit (GPU). Android supports OpenGL ES 1.1, 2.0, 3.0 and 3.1. Some applications may explicitly require a certain version of the OpenGL ES, and suitable GPU hardware is required to run such applications.

Android devices incorporate many optional hardware components, including still or video cameras, GPS, orientation sensors, dedicated gaming controls, accelerometers,gyroscopes, barometers, magnetometers, proximity sensors, pressure sensors, thermometers, and touchscreens. Some hardware components are not required, but became standard in certain classes of devices, such as smartphones, and additional requirements apply if they are present. Some other hardware was initially required, but those requirements have been relaxed or eliminated altogether. For example, as Android was developed initially as a phone OS, hardware such as microphones were required, while over time the phone function became optional. Android used to require an autofocus camera, which was relaxed to a fixed-focus camera if it is even present at all, since the camera was dropped as a requirement entirely when Android started to be used on set-top boxes.

In addition to running on smartphones and tablets, several vendors run Android natively on regular PC hardware with a keyboard and mouse. In addition to their availability on commercially available hardware, similar PC hardware–friendly versions of Android are freely available from the Android-x86 project, including customized Android 4.4. Using the Android emulator that is part of the Android SDK, or by using BlueStacks or Andy, Android can also run non-natively on x86. Chinese companies are building a PC and mobile operating system, based on Android, to “compete directly with Microsoft Windows and Google Android”. The Chinese Academy of Engineering noted that “more than a dozen” companies were customizing Android following a Chinese ban on the use of Windows 8 on government PCs.

1.4 Development

Android is developed in private by Google until the latest changes and updates are ready to be released, at which point the source code is made available publicly. This source code will only run without modification on select devices, usually the Nexus series of devices. The source code is, in turn, adapted by OEMs to run on their hardware. Android’s source code does not contain the often proprietary device drivers that are needed for certain hardware components.

The green Android logo was designed for Google in 2007 by graphic designer Irina Blok. The design team was tasked with a project to create a universally identifiable icon with the specific inclusion of a robot in the final design. After numerous design developments based on science-fiction and space movies, the team eventually sought inspiration from the human symbol on restroom doors and modified the figure into a robot shape. As Android is open-sourced, it was agreed that the logo should be likewise, and since its launch the green logo has been reinterpreted into countless variations on the original design.

1.4.1 Update Schedule

Google provides major upgrades, incremental in nature, to Android every six to nine months, which most devices are capable of receiving over the air. The latest major release is Android 5.0 “Lollipop”.

Compared to its chief rival mobile operating system, namely iOS, Android updates are typically slow to reach actual devices. For devices not under the Nexus brand, updates often arrive months from the time the given version is officially released. This is partly due to the extensive variation in hardware of Android devices, to which each upgrade must be specifically

tailored, as the official Google source code only runs on their
flagship Nexus devices. Porting Android to specific hardware is a time- and resource-consuming process for device manufacturers, who prioritize their newest devices and often leave older ones behind. Hence, older smartphones are frequently not updated if the manufacturer decides it is not worth their time, regardless of whether the phone is capable of running the update. This problem is compounded when manufacturers customize Android with their own interface and apps, which must be reapplied to each new release. Additional delays can be introduced by wireless carriers who, after receiving updates from manufacturers, further customize and brand Android to their needs and conduct extensive testing on their networks before sending the upgrade out to users.

The lack of after-sale support from manufacturers and carriers has been widely criticized by consumer groups and the technology media. Some commentators have noted that the industry has a financial incentive not to upgrade their devices, as the lack of updates for existing devices fuels the purchase of newer ones, an attitude described as “insulting”. The Guardian has complained that the method of distribution for updates is complicated only because manufacturers and carriers have designed it that way. In 2011, Google partnered with a number of industry players to announce an “Android Update Alliance”, pledging to deliver timely updates for every device for 18 months after its release; however, there has not been another official word about that alliance.

In 2012, Google began decoupling certain aspects of the operating system (particularly core applications) so they could be updated through Google Play Store, independently of Android itself. One of these components, Google Play Services, is a closed-source system-level process providing APIs for Google services, installed automatically on nearly all devices running Android version 2.2 and higher. With these changes, Google can add new operating system functionality through Play Services and application updates without having to distribute an upgrade to the operating system itself. As a result, Android 4.2 and 4.3 contained relatively fewer user-facing changes, focusing more on minor changes and platform improvements.

1.4.2 Linux Kernel

Android's kernel is based on one of the Linux kernel's long-term support (LTS) branches. As of April 2014, older Android devices use version 3.4 of the Linux kernel (or newer), while the devices sold with Android 5.0 use 3.10. The specific kernel version depends on the actual Android device and its hardware platform; Android has used various kernel versions since the version 2.6.25 that was used in Android 1.0.

Android's variant of the Linux kernel has further architectural changes that are implemented by Google outside the typical Linux kernel development cycle, such as the inclusion of components like Binder, ashmem, pmem, logger, wakelocks, and different out-of-memory (OOM) handling. Certain features that Google contributed back to the Linux kernel, notably a power management feature called "wakelocks", were rejected by mainline kernel developers partly because they felt that Google did not show any intent to maintain its own code. Google announced in April 2010 that they would hire two employees to work with the Linux kernel community, but Greg Kroah-Hartman, the current Linux kernel maintainer for the stable branch, said in December 2010 that he was concerned that Google was no longer trying to get their code changes included in mainstream Linux. Some Google Android developers hinted that "the Android team was getting fed up with the process," because they were a small team and had more urgent work to do on Android.

In August 2011, Linus Torvalds said that "eventually Android and Linux would come back to a common kernel, but it will probably not be for four to five years". In December 2011, Greg Kroah-Hartman announced the start of Android Mainlining Project, which aims to put some Android drivers, patches and features back into the Linux kernel, starting in Linux 3.3. Linux included the autosleep and wakelocks capabilities in the 3.5 kernel, after many previous attempts at merger. The interfaces are the same but the upstream Linux implementation allows for two different suspend modes: to memory (the traditional suspend that Android uses), and to disk (hibernate, as it is known on the desktop). Google maintains a public code repository that contains their experimental work to re-base Android off the latest stable Linux versions.

The flash storage on Android devices is split into several partitions, such as /system for the operating system itself, and /data for user data and application installations. In contrast to desktop Linux distributions, Android device owners are not given root access to the operating system and sensitive partitions such as /system are read-only. However, root access can be obtained by exploiting security flaws in Android, which is used frequently by the open-source community to enhance the capabilities of their devices, but also by malicious parties to install viruses and malware.

Android is a Linux distribution according to the Linux Foundation, Google's open-source chief Chris DiBona, and several journalists. Others, such as Google engineer Patrick Brady, say that Android is not Linux in the traditional Unix-like Linux distribution sense; Android does not include the GNU C Library and some of other components typically found in Linux distributions.

1.4.3 Software Stack

On top of the Linux kernel, there are the middleware, libraries and APIs written in C, and application software running on an application framework which includes Java-compatible libraries based on Apache Harmony. Development of the Linux kernel continues independently of other Android's source code bases. Android uses the Dalvik virtual machine with just-in-time compilation (JIT) to run Dalvik "dex-code" (Dalvik Executable), which is usually translated from the Java bytecode. Android 4.4 also supports new experimental runtime, Android Runtime (ART), which is not enabled by default.

Android's standard C library, Bionic, was developed by Google specifically for Android, as a derivation of the BSD's standard C library code. Bionic itself has been designed with several major features specific to the Linux kernel. The main benefits of using Bionic instead of the GNU C Library (glibc) or uClibc are its different licensing model, smaller runtime footprint, and optimization for low-frequency CPUs.

Aiming for a more suitable licensing model, toward the end of 2012 Google switched the Bluetooth stack in Android from the GPL-licensed BlueZ to the Apache-licensed BlueDroid.

Android does not have a native X Window System by default, nor does it support the full set of standard GNUlibraries. This made it difficult to port existing Linux applications or libraries to Android, until version r5 of the Android Native Development Kit brought support for applications written completely in C or C++. Libraries written in C may also be used in Java application by injection of a small Java shim and usage of the JNI.

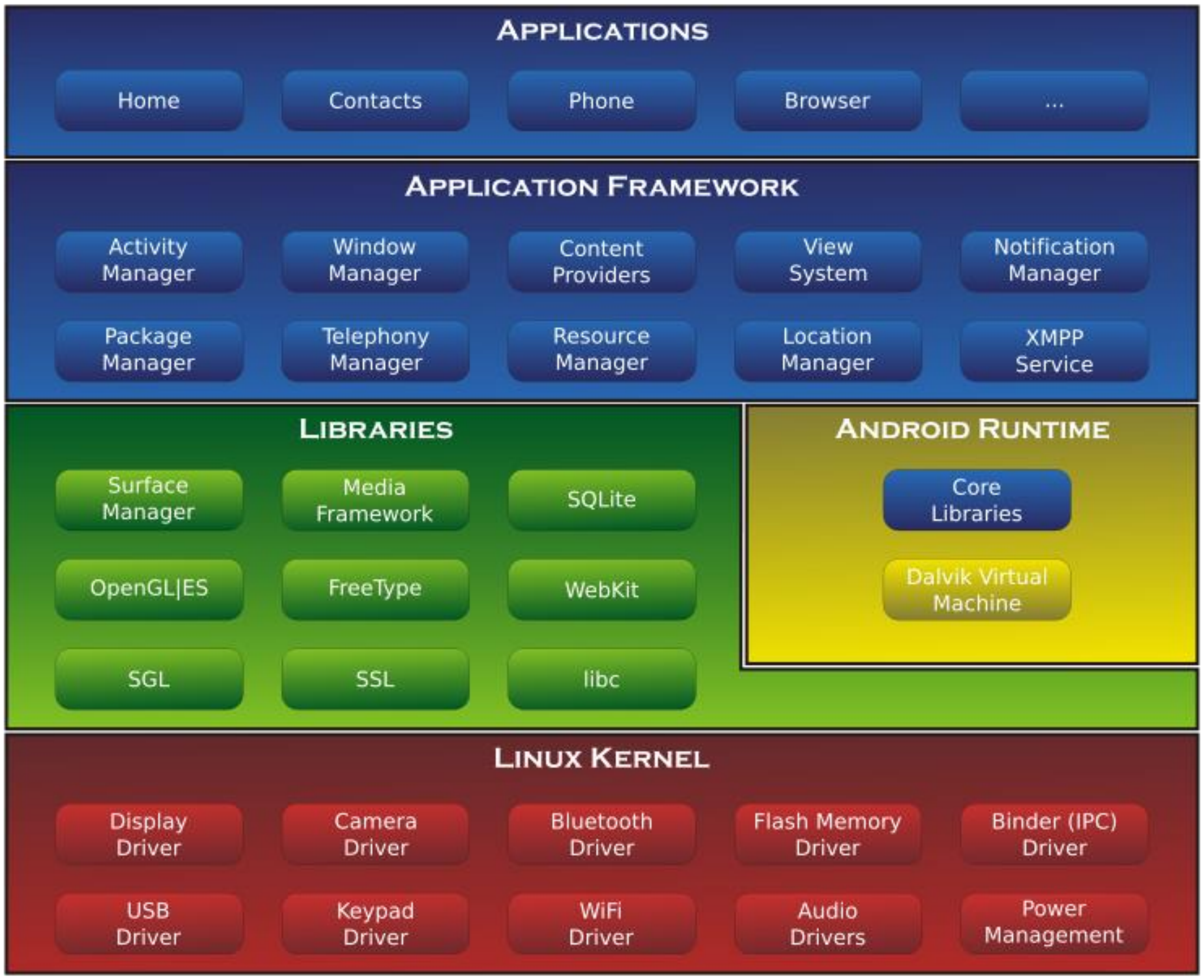


Fig 1: Android Architecture **[2]**

1.4.4 Open-Source Community

Android has an active community of developers and enthusiasts who use the Android Open Source Project (AOSP) source code to develop and distribute their own modified versions of the operating system. These community-developed releases often bring new features and updates to devices faster than through the official manufacturer/carrier channels, albeit without as extensive testing or quality assurance; provide continued support for older devices that no longer receive official updates; or bring Android to devices that were officially released running other operating systems, such as the HP TouchPad. Community releases often come pre-rooted and contain modifications unsuitable for non-technical users, such as the ability to overclock or over/undervolt the device's processor. CyanogenMod is the most widely used community firmware, and acts as a foundation for numerous others.

Historically, device manufacturers and mobile carriers have typically been unsupportive of third-party firmware development. Manufacturers express concern about improper functioning of devices running unofficial software and the support costs resulting from this. Moreover, modified firmwares such as CyanogenMod sometimes offer features, such as tethering, for which carriers would otherwise charge a premium. As a result, technical obstacles including locked bootloaders and restricted access to root permissions are common in many devices. However, as community-developed software has grown more popular, and following a statement by the Librarian of Congress in the United States that permits the "jailbreaking" of mobile devices, manufacturers and carriers have softened their position regarding third party development, with some, including HTC, Motorola, Samsung and Sony, providing support and encouraging development. As a result of this, over time the need to circumvent hardware restrictions to install unofficial firmware has lessened as an increasing number of devices are shipped with unlocked or unlockable bootloaders, similar to Nexus series of phones, although usually requiring that users waive their devices' warranties to do so. However, despite manufacturer acceptance, some carriers in the US still require that phones are locked down, frustrating developers and customers.

1.5 Security and Privacy

Android applications run in a sandbox, an isolated area of the system that does not have access to the rest of the system's resources, unless access permissions are explicitly granted by the user when the application is installed. Before installing an application, Play Store displays all required permissions: a game may need to enable vibration or save data to an SD card, for example, but should not need to read SMS messages or access the phonebook. After reviewing these permissions, the user can choose to accept or refuse them, installing the application only if they accept. The sandboxing and permissions system lessens the impact of vulnerabilities and bugs in applications, but developer confusion and limited documentation has resulted in applications routinely requesting unnecessary permissions, reducing its effectiveness. Google has now pushed an update to Android Verify Apps feature, which will now run in background to detect malicious processes and crack them down.

The "App Ops" privacy and application permissions control system, used for internal development and testing by Google, was introduced in Google's Android 4.3 release for the Nexus devices. Initially hidden, the feature was discovered publicly; it allowed users to install a management application and approve or deny permission requests individually for each of the applications installed on a device. Access to the App Ops was later restricted by Google starting with Android 4.4.2 with an explanation that the feature was accidentally enabled and not intended for end-users; for such a decision, Google received criticism from the Electronic Frontier Foundation. Individual application permissions management, through the App Ops or third-party tools, is currently only possible with root access to the device.

2 Literature Survey

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

2.1.1 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 OAuth 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 cumbersome interface and excessive advertising are eliminated, so that users are able to manipulate these apps more conveniently and smoothly.

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 smartphone market. Because of the open source, some of the development tools are free, so there are plenty of applications generated.

2.2 Paper 2: Design of Android based Media Player

2.2.1 Abstract

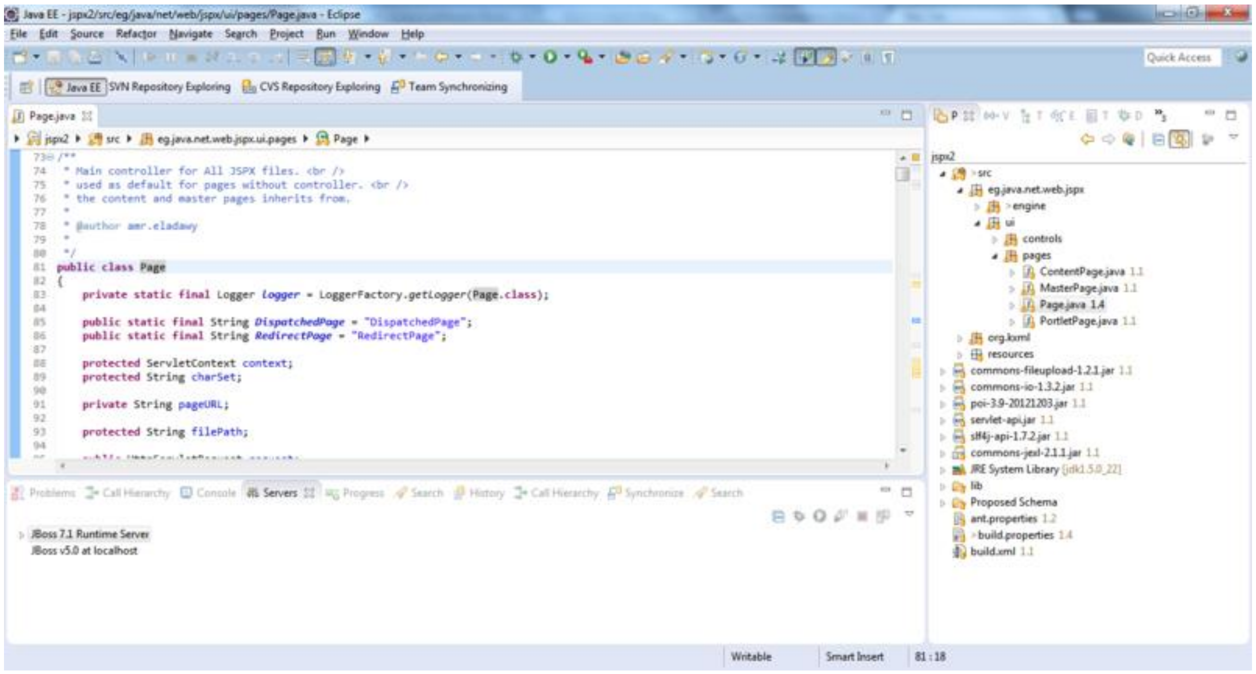
Media Player forms an integral part of today’s Smartphone. It is generally used by users to view media files of various formats. Many users like to watch video by a mobile phone, but the media player has many limitations. With a rapid development of communication and network, multimedia based technology is adopted in media player. Android is an open-source and has powerful APIs which has attracted large number of developers. The papers discuss about the study of the media player with the help of the existing media players which are available in the Android Market and proposed system for the media player which will provide the uninterrupted enjoyment for the user

With the continuous development in Science and Technology, mobile is no longer just a device used for communication but a multimedia platform that provides the ability to play the media. Playing the audio and video is just a basic thing, due the limitations it has, there are limited formats etc. Present scenario for media players provide support for some media format and recently facilities for providing the subtitles is included in the existing system. This paper demonstrates about proposed system which will provide the rich features with the help of existing features.

2.3 Development Tools

2.3.1 Eclipse

In computer programming, Eclipse is an integrated development environment (IDE). It contains a base workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications. By means of various plug-ins, Eclipse may also be used to develop applications in other programming languages: Ada, ABAP, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Natural, Perl, PHP, Prolog, Python, R, Ruby(including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.



```
/**
 * Main controller for All JSPX files. <br />
 * used as default for pages without controller. <br />
 * the content and master pages inherits from.
 *
 * @author amr.eladawy
 *
 */
public class Page
{
    private static final Logger logger = LoggerFactory.getLogger(Page.class);

    public static final String DispatchedPage = "DispatchedPage";
    public static final String RedirectPage = "RedirectPage";

    protected ServletContext context;
    protected String charSet;

    private String pageURL;

    protected String filePath;
```

```
JBoss 7.1 Runtime Server
JBoss v5.0 at localhost
```

```
jspx2
  >src
    eg.java.net.web.jspx
      >engine
      ui
        controls
        pages
          ContentPage.java 1.1
          MasterPage.java 1.1
          Page.java 1.4
          PortletPage.java 1.1
    org.kxml
    resources
  commons-fileupload-1.2.1.jar 1.1
  commons-io-1.3.2.jar 1.1
  poi-3.9-20121203.jar 1.1
  servlet-api.jar 1.1
  slf4j-api-1.7.2.jar 1.1
  commons-jexl-2.1.1.jar 1.1
  JRE System Library [jdk1.5.0_22]
  lib
  Proposed Schema
  ant.properties 1.2
  >build.properties 1.4
  build.xml 1.1
```

Fig 2: Eclipse

2.3.2 Android Development Tools (ADT) Plugin

Android Development Tools (ADT) is a plugin for the Eclipse IDE that is designed to provide an integrated environment in which to build Android applications. ADT extends the capabilities of Eclipse to let developers set up new Android projects, create an application UI, add packages based on the Android Framework API, debug their applications using the Android SDK tools, and export signed (or unsigned) .apk files in order to distribute their applications. It is a freeware available to download. It was official IDE for Android but was replaced by Android Studio (based on IntelliJ IDEA Community Edition).

2.3.3 Android SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger,libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, and Windows XP or later. For the moment one can also develop Android software on Android itself by using the AIDE - Android IDE - Java, C++ app and the Java editor app. The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) Plugin, though IntelliJ IDEA IDE (all editions) fully supports Android development out of the box, andNetBeans IDE also supports Android development via a plugin. Additionally, developers may use any text editor to edit Java and XML files, then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely).

Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (the folder is accessible only to the root user for security reasons). APK package contains .dex files (compiled byte code files called Dalvik executables), resource files, etc.

2.4 Software Development Lifecycle

The general methodology in developing a system is involved in different phases, which describe the system's life cycle model for developing software project. The concept includes not only forward motion but also have the possibility to return that is cycle back to an activity previously completed. This cycle back or feedback may occur as a result of the failure with the system to meet a performance objective or as a result of changes in redefinition of system activities. Like most systems, the life cycle of the computer based system also exhibits distinct phases.

Those are,

1. REQUIREMENT ANALYSIS PHASE
2. DESIGN PHASE
3. DEVELOPMENT PHASE
4. CODING PHASE
5. TESTING PHASE

2.4.1 Requirement Analysis Phase

This phase includes the identification of the problem, in order to identify the problem, we have to know information about the problem, the purpose of the evaluation for problem to be known. We have to clearly know about the client's requirements and the objectives of the project.

2.4.2 System Analysis Phase

Feasibility analysis involves the benefits of various approaches and the determination of the alternative approaches a\through methods like questionnaires and interviews etc., different data about the project is collected and the data throughout the project is represented in the form of UML Diagrams.

2.4.3 Design Phase

S/W design is a process through which the requirements are translated into a representation of a s/w. One of the software requirements have been analyzed and specified, the s/w design involves three technical activities: design, coding generation and testing. The design of the system is in modular form i.e., the s/w is logically partitioned into components that perform specific functions and sub functions. The design phase leads to modules that exhibit independent functional characteristics. It even leads to interfaces that reduce the complexity of the connections between modules and with the external environment. The design phase is of main importance because in this activity, decisions ultimately affect the success of s/w implementation and maintenance.

2.4.4 Development Phase

The development phase includes choosing of a suitable s/w to solve the particular problem given. The various facilities and the sophistication in the selected s/w give a better development of the problem.

2.4.5 Coding Phase

The coding phase is for translating the design of the system produced during the design phase into code in a given programming language, which can be executed by a computer and which performs the computation specified by the design.

2.4.6 Testing Phase

Testing is done in various ways such as testing the algorithm, programming code, sample data debugging is also one of following the above testing.

CHAPTER 3: REQUIREMENTS

1 Functional Requirements

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

2 External Interface Requirements

- Android emulator version 4.3
- Android operating System Smartphone

3 Software Requirements

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

CHAPTER 4: ANALYSIS, DESIGN AND MODELING

1 Integration of Development Tools

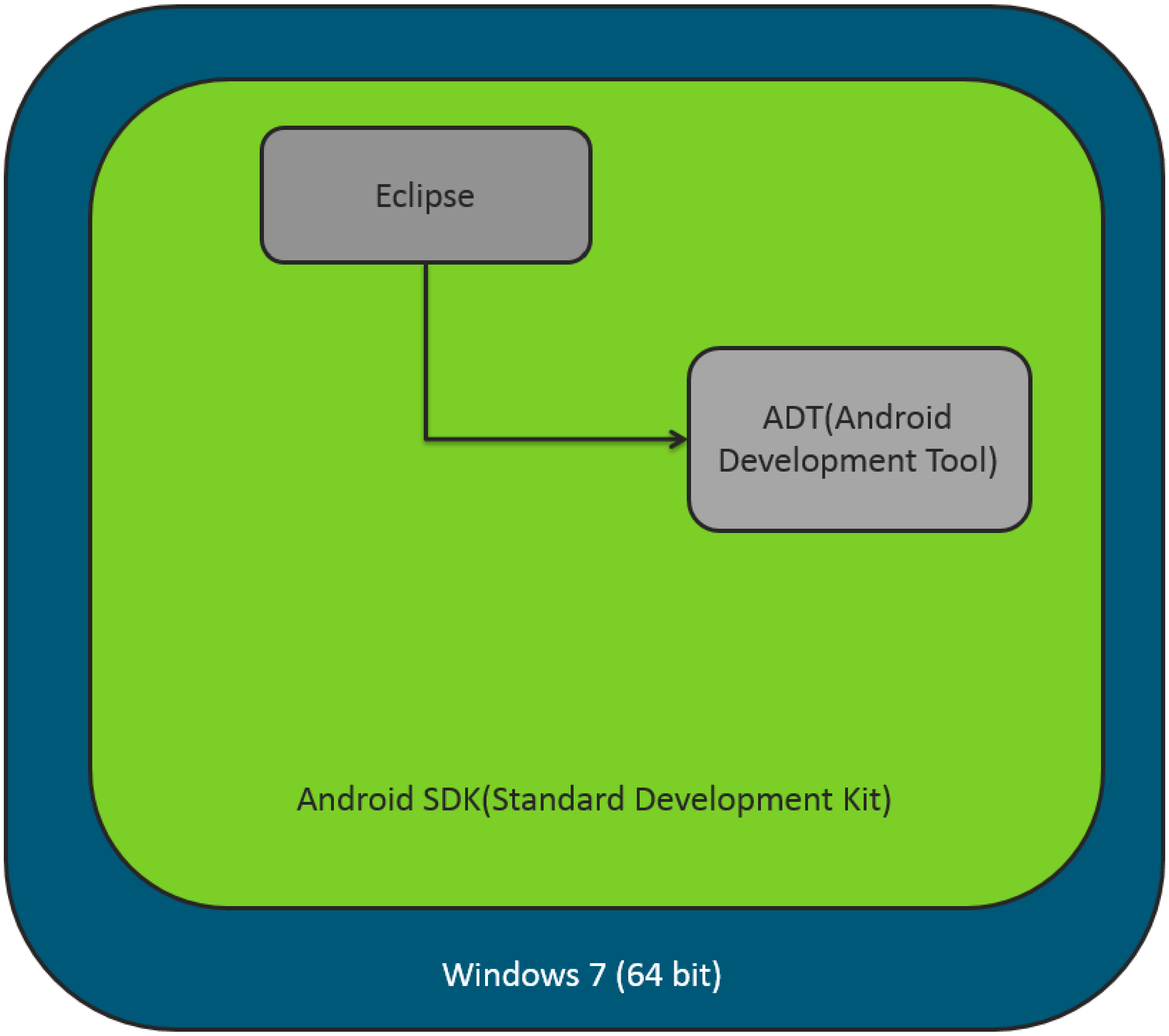

Eclipse
ADT(Android Development Tool)
Android SDK(Standard Development Kit)
Windows 7 (64 bit)

Fig 3: Integration of Development Tools

2 State Diagram

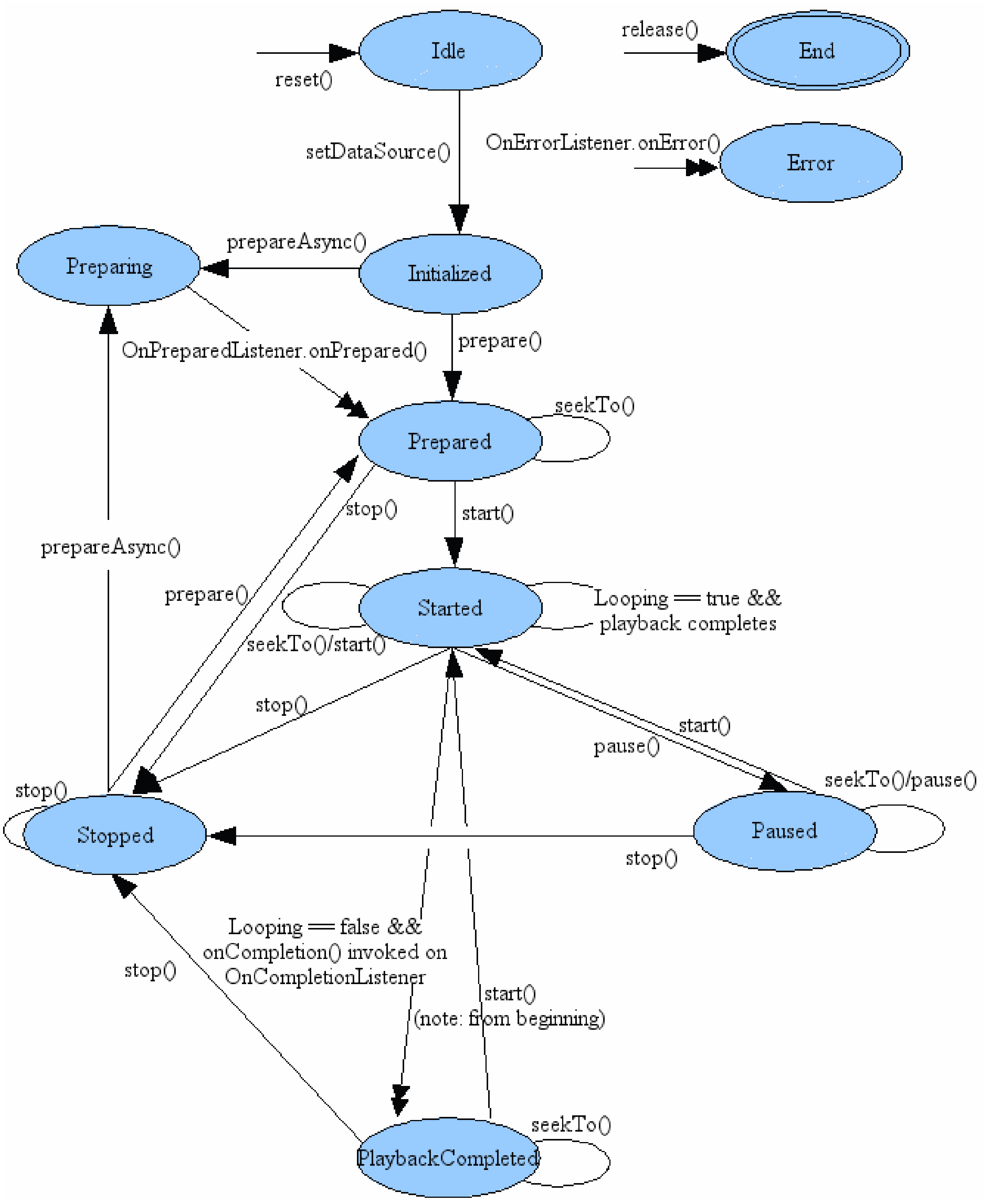
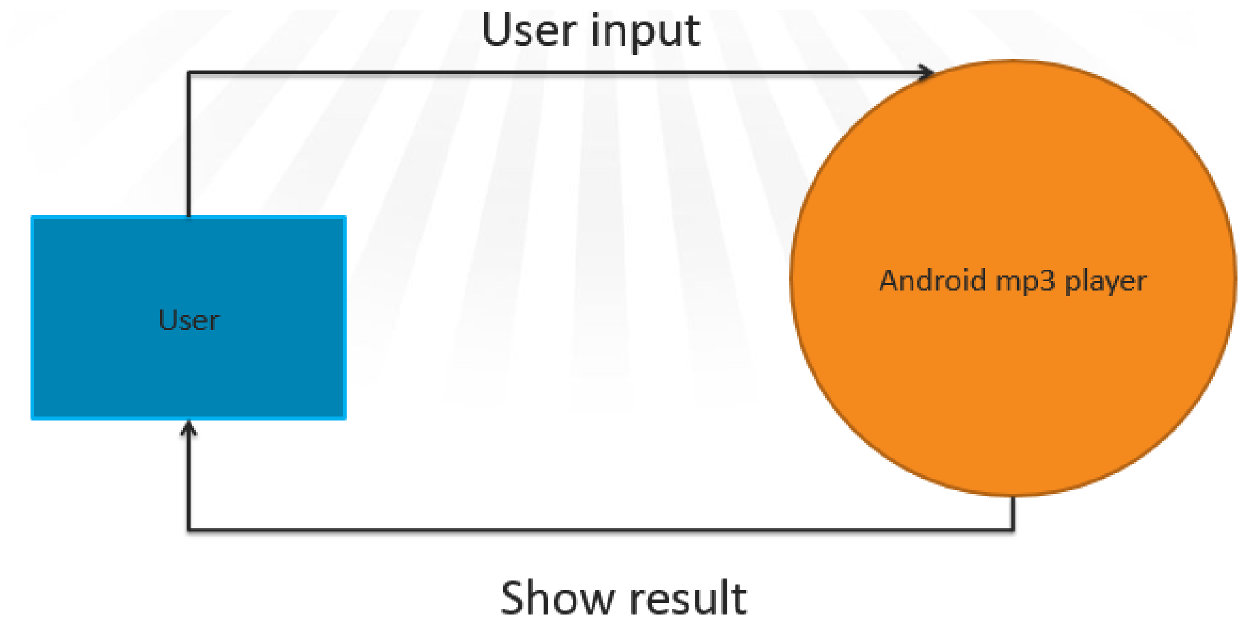


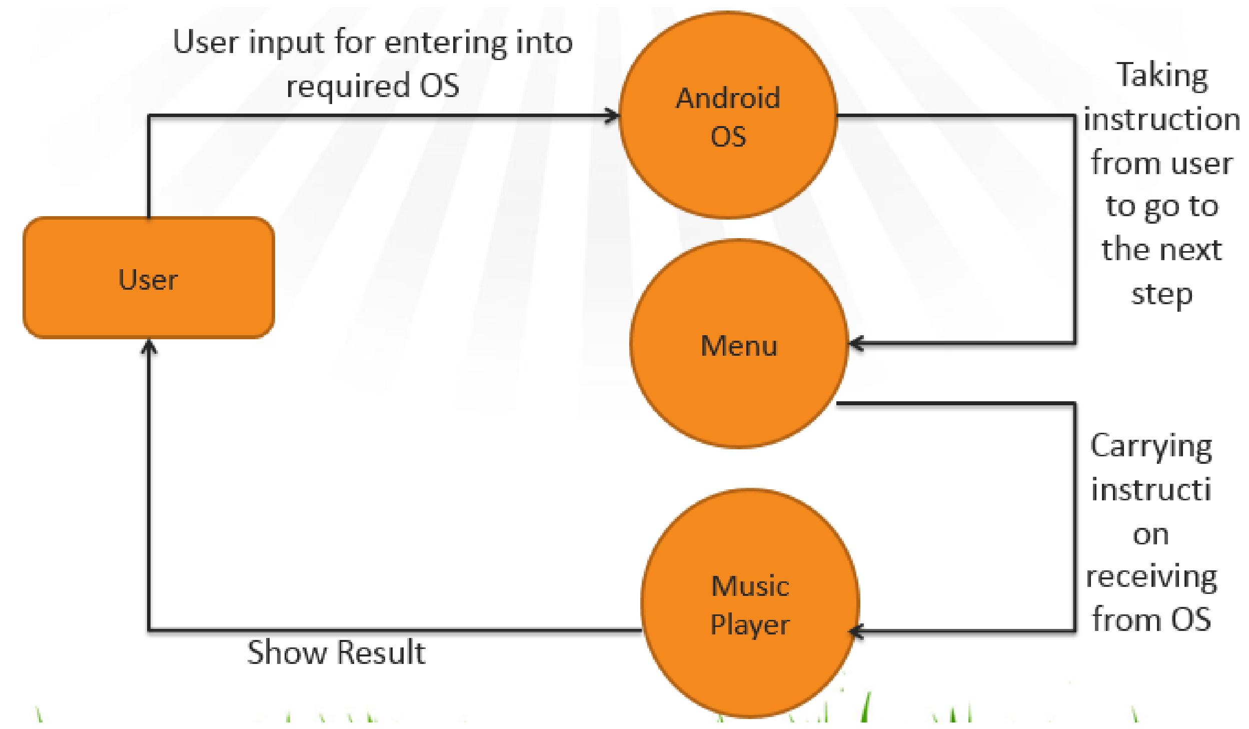
Fig 4: Media Player State Diagram

- MediaPlayer class is used to control playback of audio files.
- Playback control of audio files and streams is managed as a state machine.
- When a MediaPlayer object is just created using new or after reset() is called, it is in the Idle state.
- After release() is called, it is in the End state.
- Between these two states is the life cycle of the MediaPlayer object.
- 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.
- Under all these error conditions, the internal player engine invokes a user supplied OnErrorListener.onError() method if an OnErrorListener has been registered beforehand.
- Calling one of the overloaded setDataSource() method transfers a MediaPlayer object in the Idle state to the Initialized state.
- A MediaPlayer object must first enter the Prepared state before playback can be started.
- There are two ways that the *Prepared* state can be reached: Synchronous and Asynchronous
- prepare() (Synchronous): Transfers the object to the Prepared state.
- prepareAsync() (Asynchronous): Transfers the object to the Preparing state while the internal player engine continues to work on the rest of preparation work. When the preparation completes or when prepare() call returns, the internal player engine then calls the onPrepared() method of the OnPreparedListener interface.
- To start the playback, start() must be called. After start() returns successfully, the MediaPlayer object is in the Started state.
- Playback can be paused via the pause() method. It then enters the Paused state.
- Playback can be stopped via the stop() method. It then enters the Stopped state.
- After completion of playback, it enters the PlaybackCompleted state.

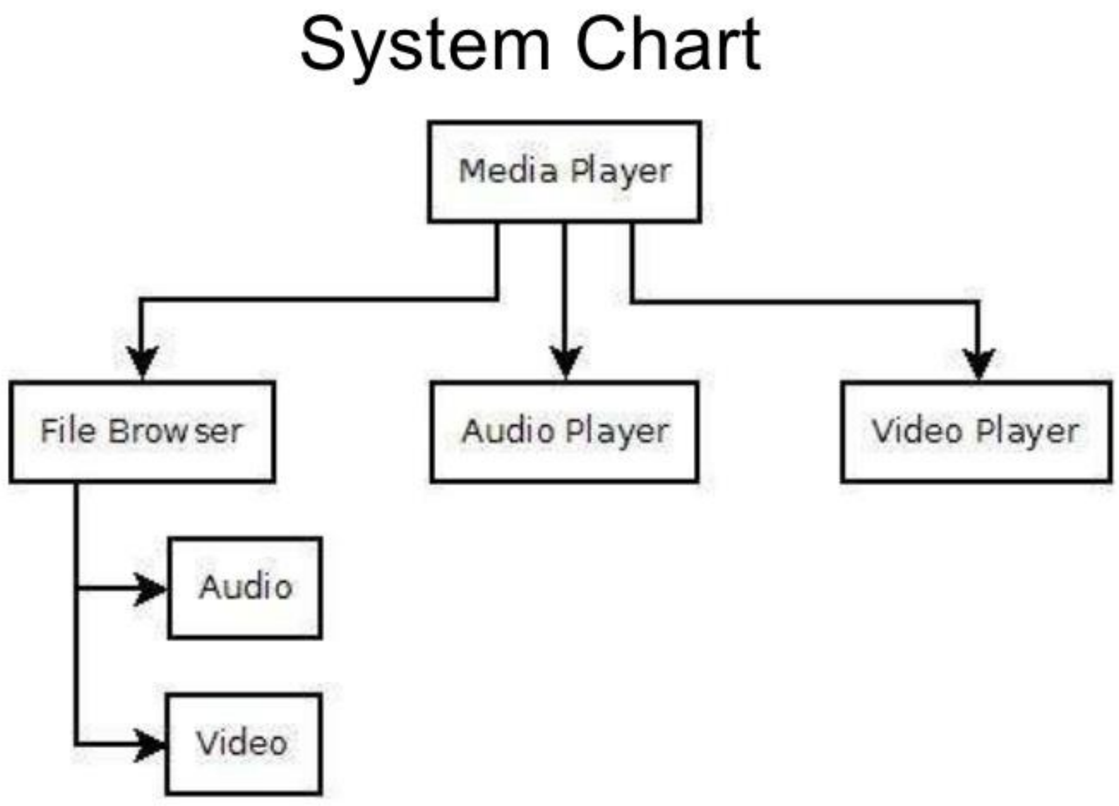
3 Level 0 DFD



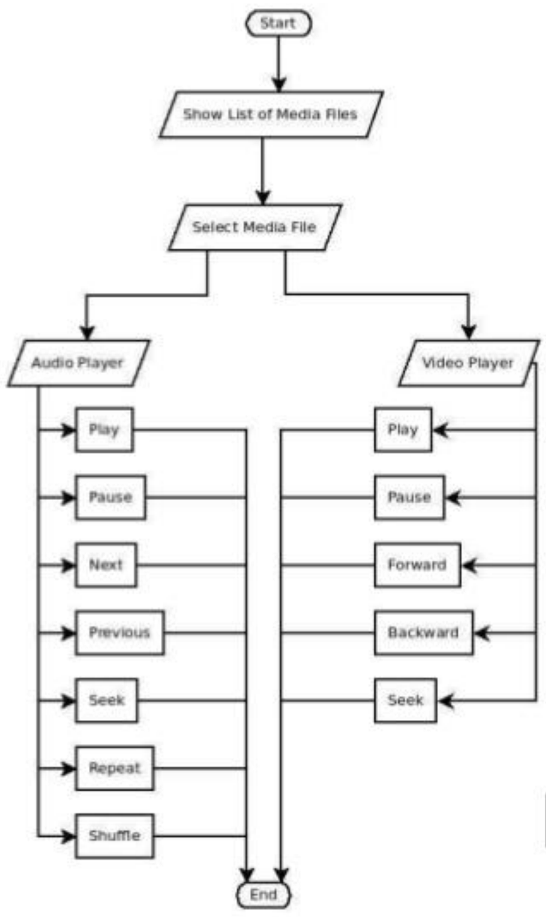
4 Level 1 DFD


User input for entering into required OS
Android OS
Taking instruction from user to go to the next step
User
Menu
Carrying instructi on receiving from OS
Music Player
Show Result

5 System Chart

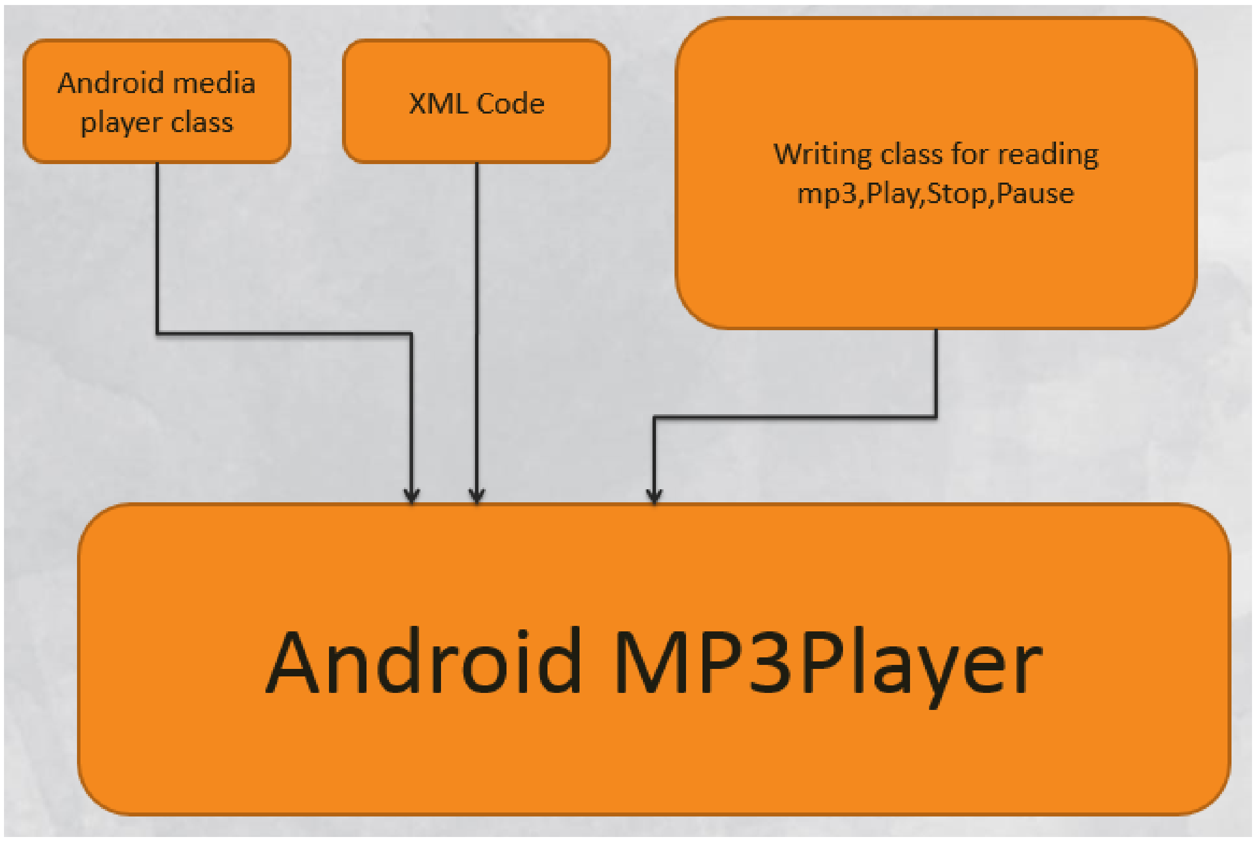


6 Flow Chart

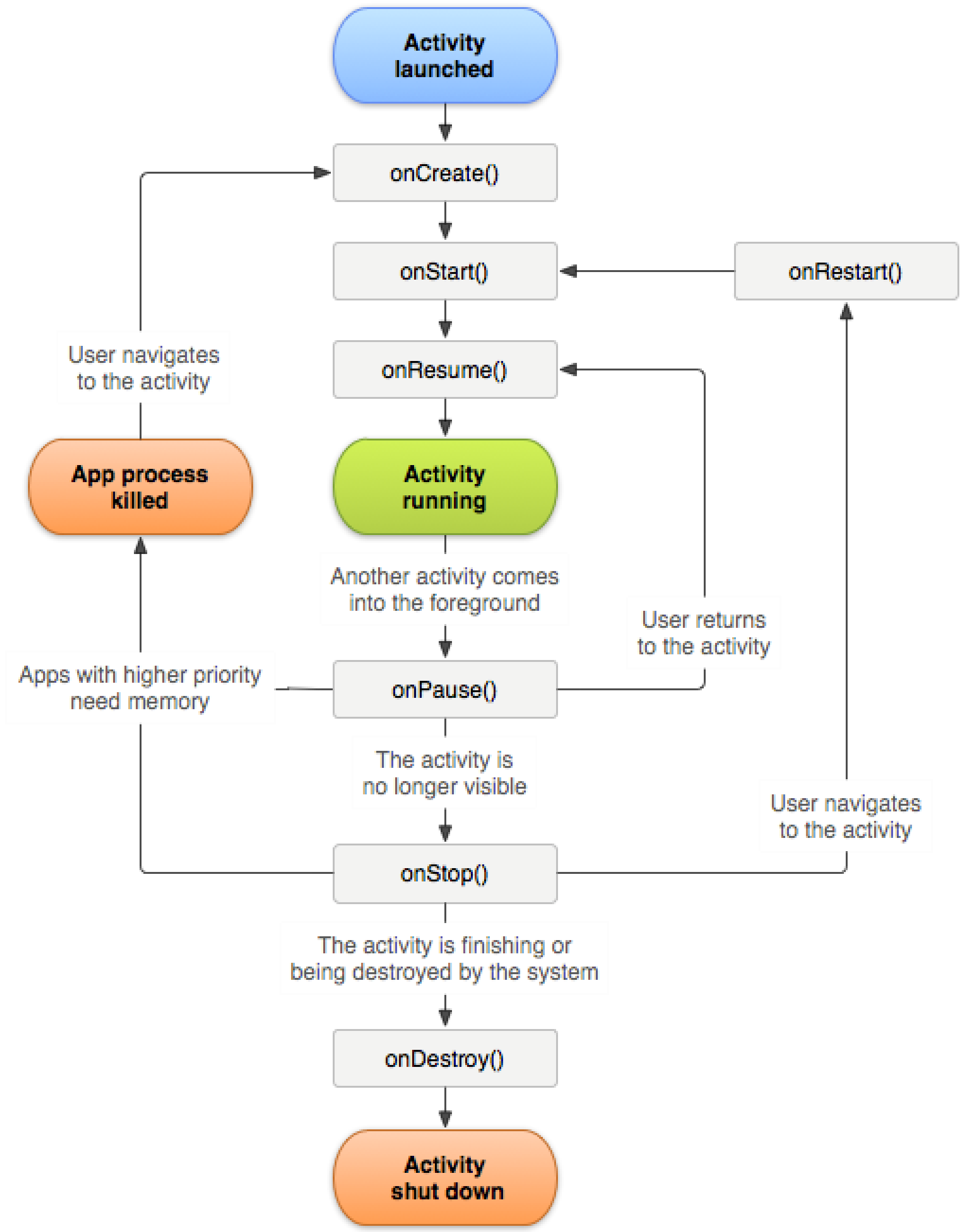


Flow Chart

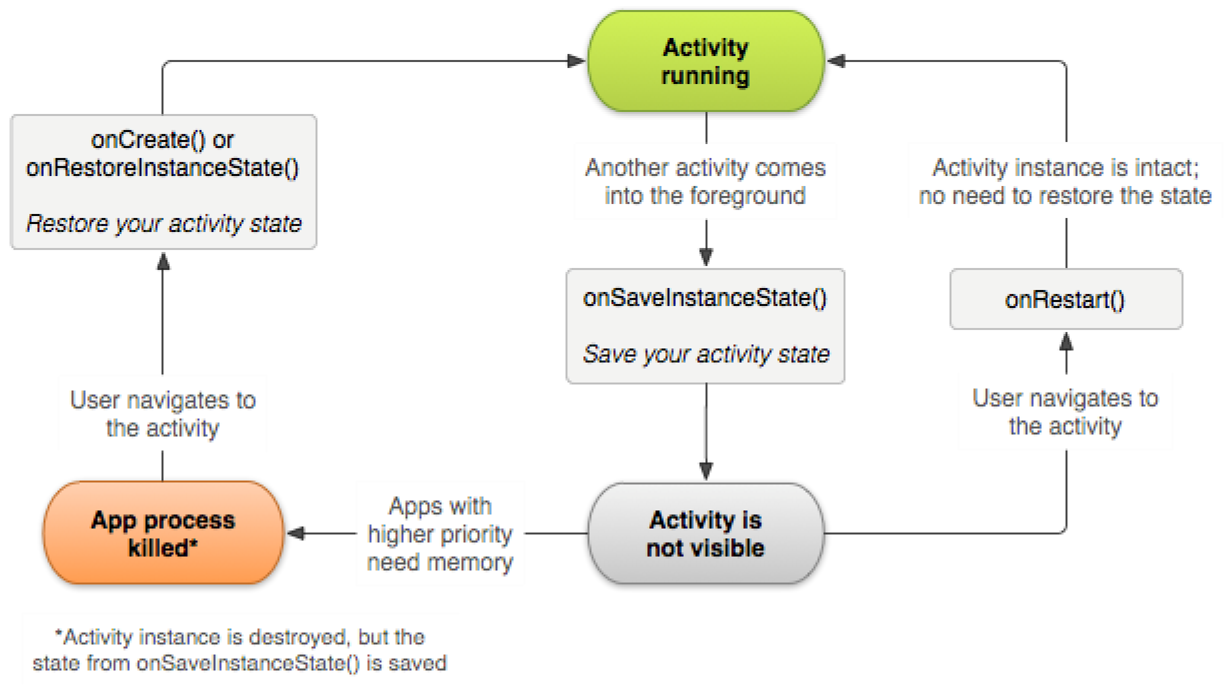
7 Building Blocks of Android Code


Android media player class
XML Code
Writing class for reading mp3,Play,Stop,Pause
Android MP3Player

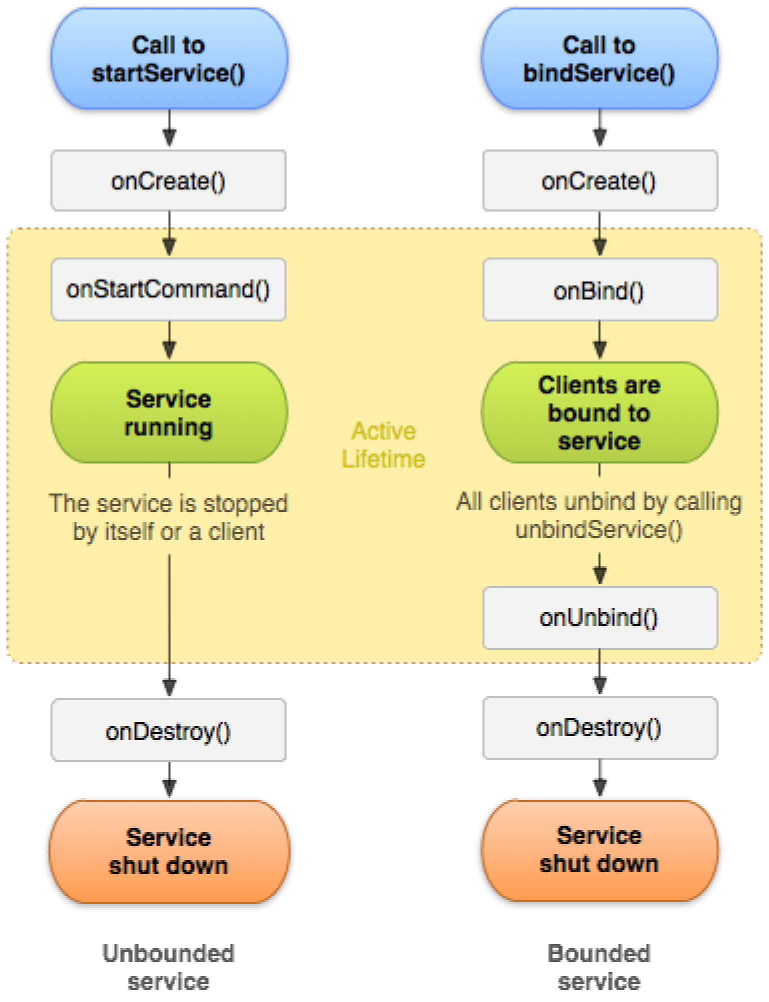
8 Activity Lifecycle


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

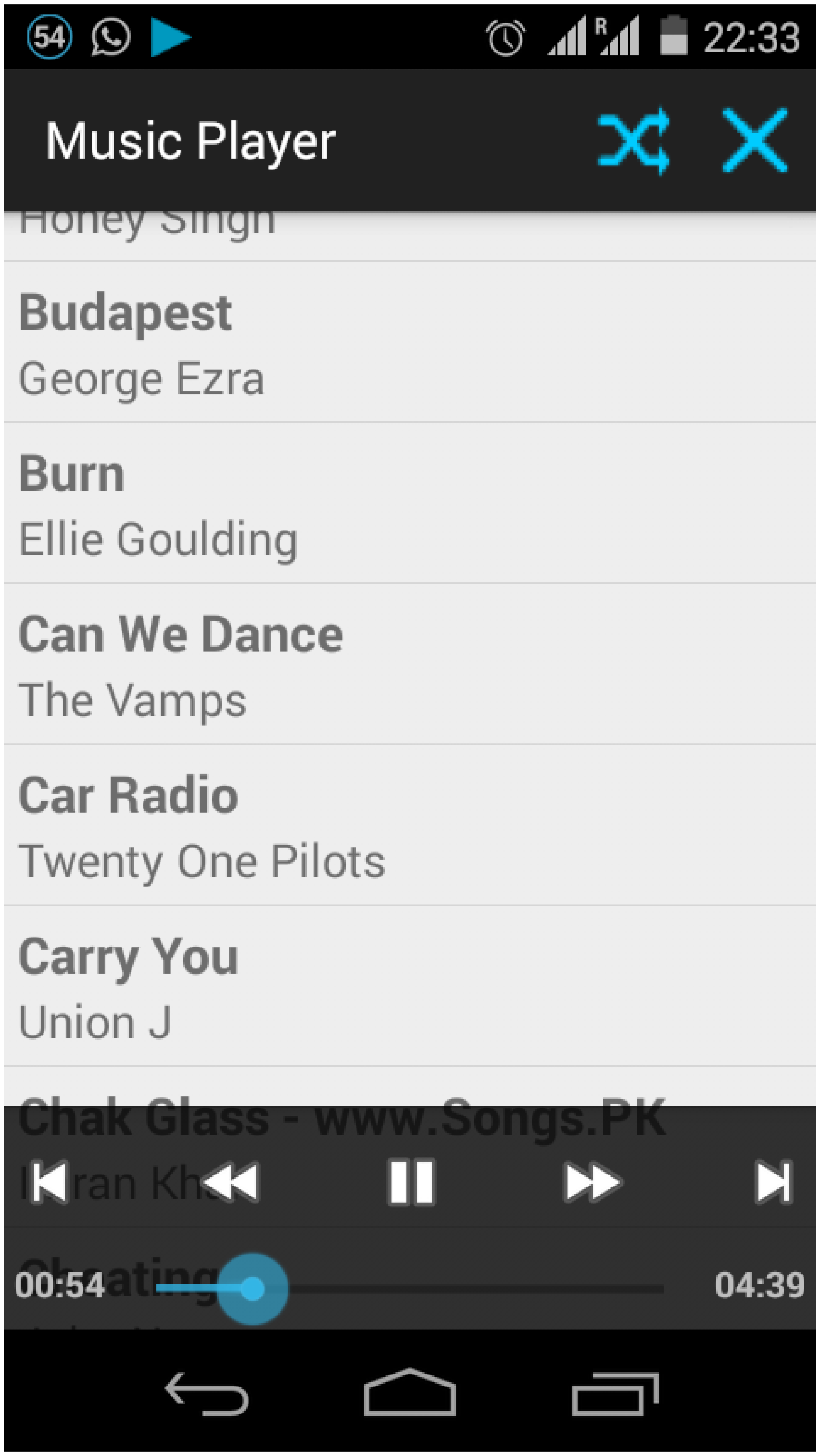
9 Activity State Diagram


Activity running
onCreate() or onRestoreInstanceState()
Restore your activity state
Another activity comes into the foreground
Activity instance is intact; no need to restore the state
onSaveInstanceState()
Save your activity state
onRestart()
User navigates to the activity
User navigates to the activity
App process killed*
Apps with higher priority need memory
Activity is not visible
*Activity instance is destroyed, but the state from onSaveInstanceState() is saved

10 Service Lifecycle



11 Screenshots


22:33

Music Player

Honey Singh

Budapest
George Ezra

Burn
Ellie Goulding

Can We Dance
The Vamps

Car Radio
Twenty One Pilots

Carry You
Union J

Chak Glass - www.Songs.PK
Imran Kh

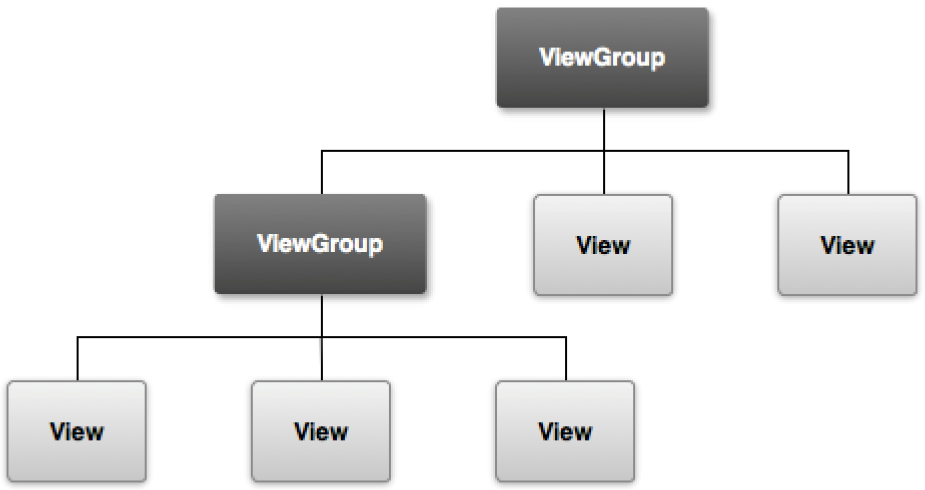
Cheating

00:54 04:39

CHAPTER 5: DEVELOPMENT PHASE

1 User Interface

1.1 Layout

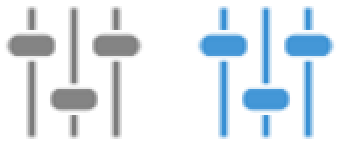


The user interface for each component of your app is defined using a hierarchy of View and ViewGroup objects. Each view group is an invisible container that organizes child views, while the child views may be input controls or other widgets that draw some part of the UI.

1.2 Buttons

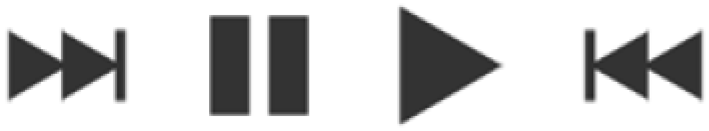
A button consists of text or an icon (or both text and an icon) that communicates what action occurs when the user touches it.

1.2.1 Equalizer Button



1.2.2 Repeat Button

1.2.3 Basic Control Buttons



1.2.4 Shuffle Button

1.2.5 Playlist Button

1.2.6 Launcher Icon

CHAPTER 5: FUTURE ENHANCEMENTS

Search facility in library with search performed on title, artist and album fields. Equalizer feature with presets and user configurable. Online radio station stream feature with presets and user provided URL support. Preset playlists like Top 25 Most Played, Recently Added and Recently Played. Option to create playlists.

CHAPTER 6: CONCLUSION

Android is a truly open, free development platform based on Linux kernel. The app will provide the user with audio playback from external storage as well as an option to access online radio stations over the internet significantly increasing the user's music library.

CHAPTER 7: LIST OF REFERENCES

[1] Android Official Developers Guide – http://developer.android.com

[2] Wikipedia Android - http://en.wikipedia.org/wiki/Android_(operating_system)

[3] Research and Development of Mobile Application for Android Platform – Li Ma, Lei Gu and Jin Wang - International Journal of Multimedia and Ubiquitous Engineering (2014)

[4] Android Based Media Player - Akshay R. Mukadam, Darshal N. Manchekar, Gaurav G. Panchal, Prasad P. Kanade, Atul Yadav - International Journal of Engineering Science and Innovative Technology (2013)