ALL IN ONE MANAGER APPLICATION FOR ANDROID OPERATING SYSTEM

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. Ravindara Bhatt

By

Karan Kapoor

Roll No.: 111217

Jaypee University of Information and Technology

Waknaghat, Solan – 173234, Himachal Pradesh

Certificate

This is to certify that project report entitled "**ALL IN ONE MANAGER APPLICATION FOR ANDROID OPERATING SYSTEM**", submitted by Karan Kapoor (111217) 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.Ravindara Bhatt

(Signature)

Assistant Professor CSE Department

Acknowledgement

It has been a great honor for me to work on “**AllinOneManager for android**" in this esteemed institute Jaypee University of Information Technology, Waknaghat Solan.

I am thankful to **Mr. Ravindra Bhatt** for providing me with a wonderful opportunity in the form of this project and letting me use the resources available at JUIT. He has been a great source of inspiration for me throughout my project. This experience enabled me to explore my hidden potentials. It gave me an opportunity for not only understanding but practically implementing the various concepts I have learnt in the classrooms.

I am sincerely thankful to my mentor, **Mr. Ravindra Bhatt** who have been helping me throughout my training in learning new concepts and software. He helped me in settling down and made me familiar with the working environment.

All the credit certainly goes to him and JUIT management. Their constant guidance, inspiration and constructive suggestions have helped me in the successful completion of my project successfully. I hope I've not missed out anyone who has helped me out during my training period, if so I express my sincere gratitude towards them.

KARAN KAPOOR

Abstract

Creating an all in one manager application, which not only stands apart from the rest of applications in the market, but also to create an application which can be helpful in daily working.

To create an **All in one manager** application, which provides user with rich features to manage the device on which it is deployed. Creating the application, which not only provides rich features but is optimized so that the device can be made more productive with simpler user environment. All in one manager is an application which is created with a purpose of providing user with a choice to perform several functions simultaneously.

All in one manager application provides features like **Battery Manager (**for device battery efficiency), **Application Manager** (Allowing user to full control of the applications running on the device), **Task Manager** (allowing user to manage task easily), **Contact Manager** (allowing user to manage contacts easily) and a **Note editor**(To manage user's note).

CONTENT ______________________________ Page No.

FIGURES____________________________Page No.

CHAPTER 1

INTRODUCTION

Mobile applications are becoming increasingly prevalent today, particularly in the world of business. With huge potential and free of cost nature android became quite popular, leading to a huge user base. With more than 1.3 billion applications present for android platform and thousands coming out every day android has become favorite of most developers.

1.1 The Android operating system

Android is an operating system based on the Linux kernel. The project responsible for developing the Android system is called the Android Open Source Project (AOSP) and is primarily lead by Google.

The Android Operating System was developed with handheld devices in mind utilizing touch interface. In beginning android was being developed for digital cameras interface but later it was observed that this operating system has much more potential in the smartphone market. With huge potential and free of cost nature android became quite popular, leading to a huge user base.

There are many advantages to developing for the Android platform:

- **Free of cost startup-**The development tools for the platform are free to download, and Google only charges a small fee to distribute applications on the Android Market.
- **Innovation and exploration freedom-**The Android OS is an open-source platform based on the Linux kernel and multiple open-source libraries. In addition to building applications to run on Android devices, developers are free to contribute to or extend the platform as well.
- **Collaboration freedom-** Android developers are not required to sign an NDA and are encouraged to collaborate and share source code with each other. According to a survey by Black Duck Software, the number of open source mobile apps and libraries grew at a rate of 168% from 2008 to 2009, faster on Android than any other platform. This means more code that you can reuse in your own projects to bring them to market much faster.
- **Open distribution model.** Very few restrictions are placed on the content or functionality allowed in Google's Android Market, and developers are free to distribute their applications through other distribution channels as well.

- **Multi-platform support.** There are a wide variety of hardware devices powered by the Android OS, including many different phones and tablet computers. Development for the platform can occur on Windows, Mac OS or Linux.
- **Multi-carrier support.** A large number of telecom carriers currently offer Android powered phones. [1]

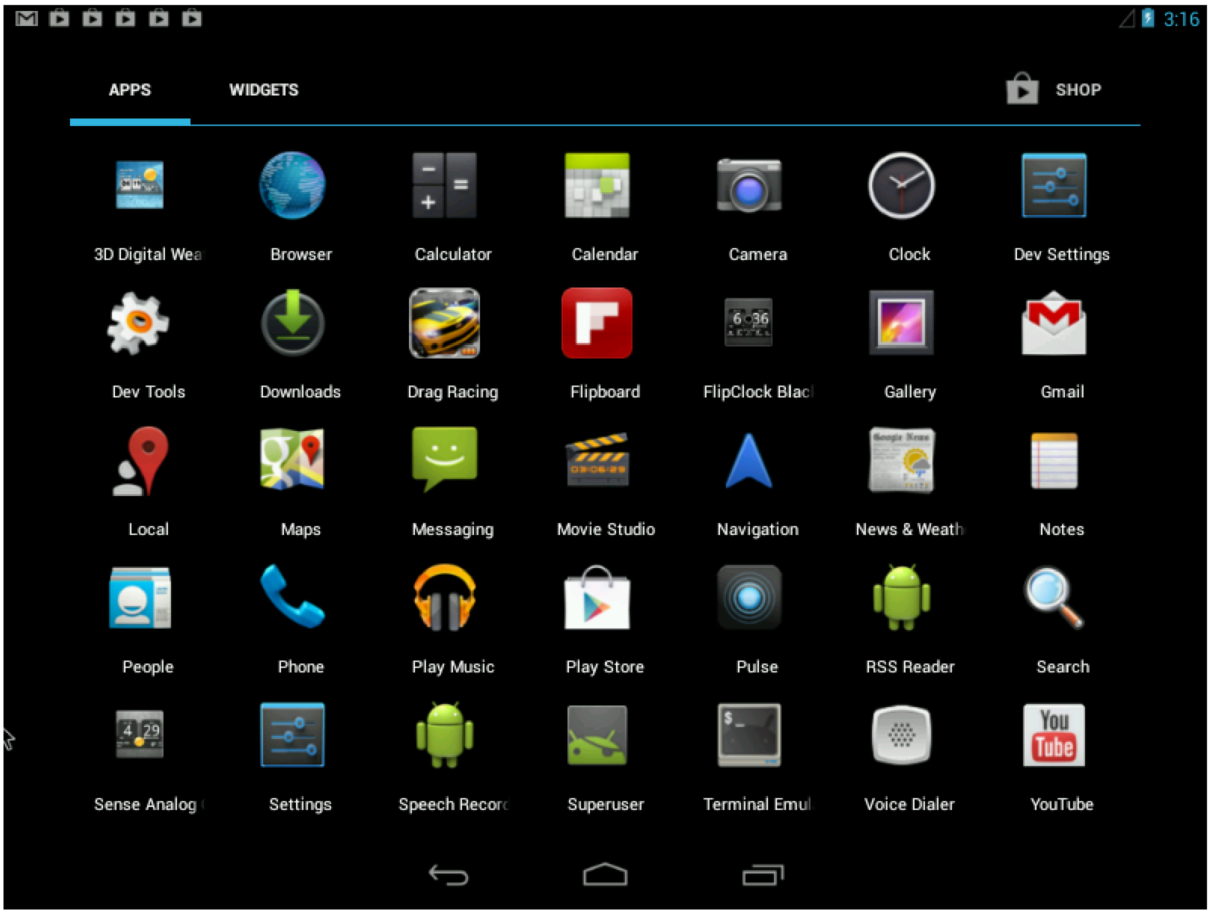


Fig 1 Android Interface

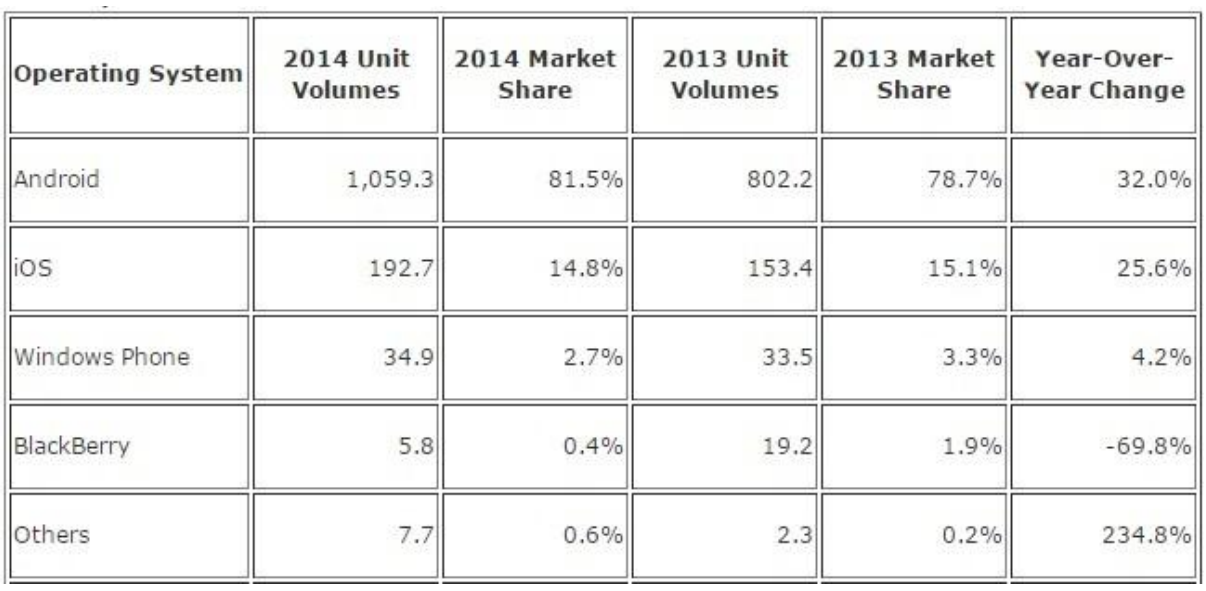
1.2 Android Applications and Market Scenario.

Android applications can be easily developed using various free to use Development kits available on the internet coupled with free nature and easy uploading of applications to Google Store makes android development a boon for application developers. In recent years a lot of development in smartphone technologies with dropping of prices of hardware has resulted in tremendous production of android applications varying from as simple as wallpaper changer to complex User Interfaces and enjoyable games. As the user base for android is growing day by day so is the need for new applications, there are thousands of similar applications from which user can choose as per his/her preference.

Being free of cost android development has lead to huge number of developers developing applications for android platform. With support from various smartphone vendors development for android has reached new heights. Android market share is constantly rising and is at its peak nowadays. With Android securing almost about 85% of market share[2].(as per the data by Tech Times).Android has been the most developers choice of development platform as it provides the developer with flexibility in development and requires low start up capital.

Major competition to android platform in smartphone market is from IOS, Windows and blackberry, all of them being proprietary platforms giving an edge to android platform over all these other platforms. With new innovations every day and hardware becoming cheap software costs need to come down too which is easier said than done, but being free of cost android is triumphant here.

With uses that can be found in various other devices has even further led to android stronghold in the market, these devices include Digital camera (Eg.Galaxy Camera), Gps navigators, netbooks, smartbooks, smart TVs (Android TV, Google TV). In addition, the Android operating system has seen applications on smart glasses (Google Glass), smartwatches, headphones, car CD and DVD players, portable media players. Ouya, a video game console running Android.



Operating System	2014 Unit Volumes	2014 Market Share	2013 Unit Volumes	2013 Market Share	Year-Over-Year Change
Android	1,059.3	81.5%	802.2	78.7%	32.0%
iOS	192.7	14.8%	153.4	15.1%	25.6%
Windows Phone	34.9	2.7%	33.5	3.3%	4.2%
BlackBerry	5.8	0.4%	19.2	1.9%	-69.8%
Others	7.7	0.6%	2.3	0.2%	234.8%

Fig 2. Top Four Smartphone Operating Systems, Unit Shipments, Market Share, and Year-Over-Year Growth, Calendar Year 2014 Data (Units in Millions).
[3](Source: http://www.idc.com/getdoc.jsp IDC Worldwide Quarterly)

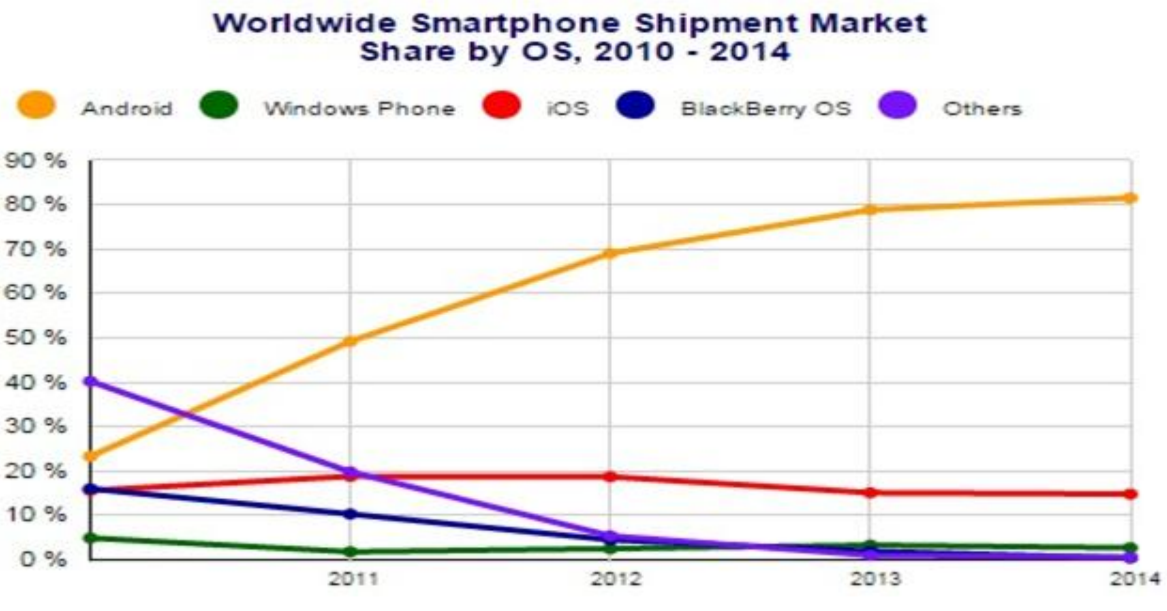


Fig 3.WorldWide Smartphone Shipment Market Share by Operating System 2010-2014.
[4](Source: http://www.idc.com/getdoc.jsp IDC Worldwide Quarterly)

1.3 Challenges And Issues

- **Software Fragmentation Issue**
  A few variants of Android working framework are accessible on diverse gadgets. Frequent updates change the version of Android OS that keeps running on the gadget. This suggests that the engineers can't simply concentrate on the latest version of the OS as not everybody updates their gadgets. It is inconvenient for many users to upgrade their operating systems.

- **Hardware fragmentation:**
  There are no less than 170 running Android, with generally varying features, from consoles to cameras, in addition to diverse screen shapes and sizes.

- **Stability of the applications:**
  As new applications are turning out regularly, there is no surety that the application will be stable or perform reliably over the wide assortment of gadget present, prompting poor or corrupted performance.

- **Huge number of similar applications:**
  As android development is free and publishing your application is also quiet easy leading to number of similar applications in case of android there can be thousands, which in turn leads to copyright infringements.

1.4 Motivation

- **Huge Market Share**

Android has a much greater and developing business with an overall about 80% share in the market beating the competition by quite a long stretch.

- **Huge chance of success**

With the expansion in Google Play market, the odds of created application making it successful are a lot more than its competitors.

- **Free of cost application publication**

Google Play provides its users over 12% more ad inventory than its competitors which makes it relatively easier and cheaper to advertise your applications.

- **Quicker Turnaround**

There is no perfect application - meaning that there is always room for improvement. Google Play provides a very helpful platform for beginners and their applications.

- **Huge Free Learning Resources**

Android community has grown quite big. It has contributed thousands of resources for quickly learning the development on this platform. You can locate rich and innovative media to learn android free of any expense.

- **The Java Advantage**

Since Android uses Java as its programming language, it has almost all the advantages of Java. Specially in terms of a lot of free and open source java libraries that can be used to develop any kind of app you are thinking of.

1.5 Objective

- To Develop applications which provide several functionalities.
- To provide simple interface for the developed applications.

1.6 Problem Statement

Today half the population of the world is using smartphones, smartphones that consume battery and resources. Everybody wants to maximize the performance and longevity of device working period. People want easy to use applications and applications that can accomplish several activities under one main application. Considering all this in mind **All in One Manager** for Android platform is being created, which contains, **Battery Manager(**for device battery efficiency), **Application manager + Task Killer**(Allowing user to full control of the applications running on the device and a way to destroy these applications), **Contact Manager**(allowing user to manage contacts easily) and a **Note editor**(To manage user's note).

1.7 Organization of work

Chapter 1 gives a brief introduction of android with challenges and issues associated with android, android applications and its market scenario, motivation and objective of project and underlying problem statement. **Chapter 2** is about the literature survey done about android platform ranging from android development tools to android version history, application life cycle, battery specifications and survey of similar applications to ones being developed. In **Chapter 3** Android architecture is discussed. In **Chapter 4** requirements for development of android applications such as need of IDE and SDK is discussed. **Chapter 5** is about proposed work describing the application development phases and flowcharts for development of applications. **In Chapter 6** we provide with simulation of android on computers and test devices used for testing the applications. **Chapter 7** is comprised of output snapshots of the applications. In **Chapter 8** Results of the project are given. In **Chapter 9** Conclusion and future works are given.

CHAPTER 2

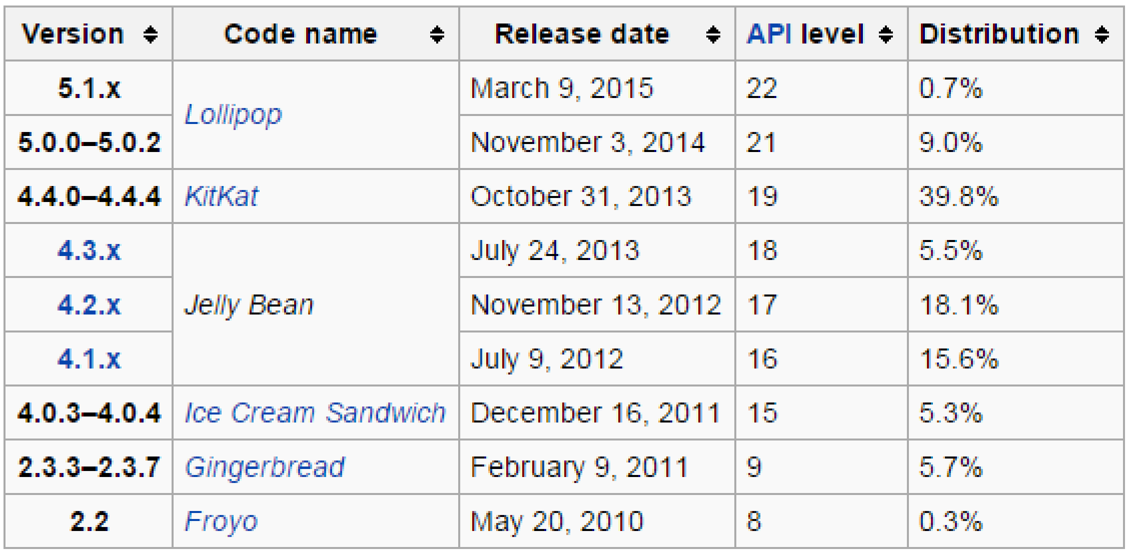
LITERATURE SURVEY

The objective behind making this application was to bring the rich functionalities of several applications onto a mobile device. So while surveying as to on which platform or rather operating system the project has to be implemented , android platform was selected for the following reasons:

- Android is an open source platform
- Supports multifunction
- Provides rich and free tools to make interactive application
- Downloading the software's required for making the application are absolutely free

Applications, that extend the functionality of devices, are written primarily in the Java programming language (without the usual "write once, run anywhere" claim of the Java platform) 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. Initially, Google's supported integrated development environment (IDE) was Eclipse using the Android Development Tools (ADT) plugin in December 2014, Google released Android Studio, based on IntelliJ IDEA, as its primary IDE for Android application development. 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.[5]

2.1 Android Versions:



Version	Code name	Release date	API level	Distribution
5.1.x	*Lollipop*	March 9, 2015	22	0.7%
5.0.0–5.0.2		November 3, 2014	21	9.0%
4.4.0–4.4.4	*KitKat*	October 31, 2013	19	39.8%
4.3.x	*Jelly Bean*	July 24, 2013	18	5.5%
4.2.x		November 13, 2012	17	18.1%
4.1.x		July 9, 2012	16	15.6%
4.0.3–4.0.4	*Ice Cream Sandwich*	December 16, 2011	15	5.3%
2.3.3–2.3.7	*Gingerbread*	February 9, 2011	9	5.7%
2.2	*Froyo*	May 20, 2010	8	0.3%

Fig 4. Android Version History(Source:Wikipedia.org\Androidversions\)

2.2 Application Life cycle:

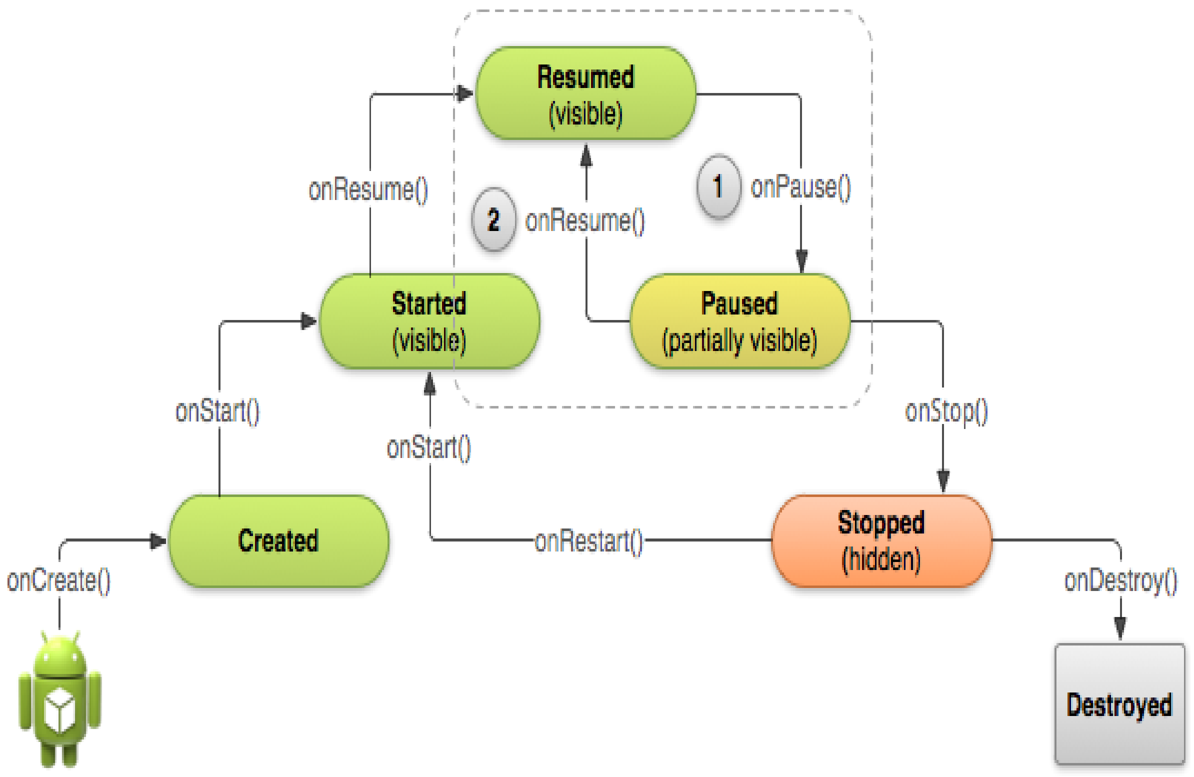


Fig 5 Android application life cycle(Source: en.wikipedia.org\android\)

- **Resumed**

  In this state, the activity is in the foreground and the user can interact with it. (Also sometimes referred to as the "running" state.

- **Paused**

  In this state, the activity is partially obscured by another activity—the other activity that's in the foreground is semi-transparent or doesn't cover the entire screen. The paused activity does not receive user input and cannot execute any code.

- **Stopped**

  In this state, the activity is completely hidden and not visible to the user; it is considered to be in the background. While stopped, the activity instance and all its state information such as member variables is retained, but it cannot execute any code. [6]

2.3 Battery Specifications:

A lithium-ion battery (sometimes Li-ion battery or LIB) is a member of a family of rechargeable battery types in which lithium ions move from the negative electrode to the positive electrode during discharge and back when charging. Li-ion batteries use an intercalated lithium compound as one electrode material, compared to the metallic lithium used in a non-rechargeable lithium battery. The electrolyte, which allows for ionic movement, and the two electrodes are the constituent components of a lithium-ion cell.
Lithium-ion batteries can be dangerous under some conditions and can pose a safety hazard since they contain, unlike other rechargeable batteries, a flammable electrolyte and are also kept pressurized. Because of this the testing standards for these batteries are more stringent than those for acid-electrolyte batteries, requiring both a broader range of test conditions and additional battery-specific tests. This is in response to reported accidents and failures, and there have been battery-related recalls by some companies.

Fig 6 Li-ion Battery(Source: www.shoptitans.com)

2.4 Applications providing similar services:

- **Juice Defender(Battery Manager)**

  Packed with seemingly endless options, Juice Defender is one of the best battery managers out there. The free client lets you manage common connections, such as mobile data as well as Wi-Fi and Bluetooth. Multiple preset modes, like "aggressive" and "balanced," allow for toggling and scheduling, background synchronization, and choosing which apps can keep your screen on.[7]

- **Task Manager(Application Viewer + Task Killer)**

  Task Manager kills all tasks and apps with one tap and it does it. If you are not willing to swift through multiple apps and tasks and wonder which to close and which to keep.[8]

- **Contacts+(Contact Manager)**

  Contacts+ is a very robust, well-designed address book. What makes Contacts+ stand out is that it integrates many of your social networks and messaging apps into one place, pulling metadata, information, and contact photos into an easy-to-use interface.[9]

- **Simplenote(Note Editor)**

  Simplenote is basically perfect for taking and syncing plain-text notes. It's just a nice-looking, very fast app that lets you type into it and then syncs what to type to other platforms and makes everything searchable.[10]

In next chapter, we are going to look into Android architecture and its components in detail.

CHAPTER 3
ANDROID ARCHITECTURE

Application architecture is the process of defining a structured solution that meets all of the technical and operational requirements, while optimizing common quality attributes such as performance, security, and manageability. It involves a series of decisions based on a wide range of factors, and each of these decisions can have considerable impact on the quality, performance, maintainability, and overall success of the application.

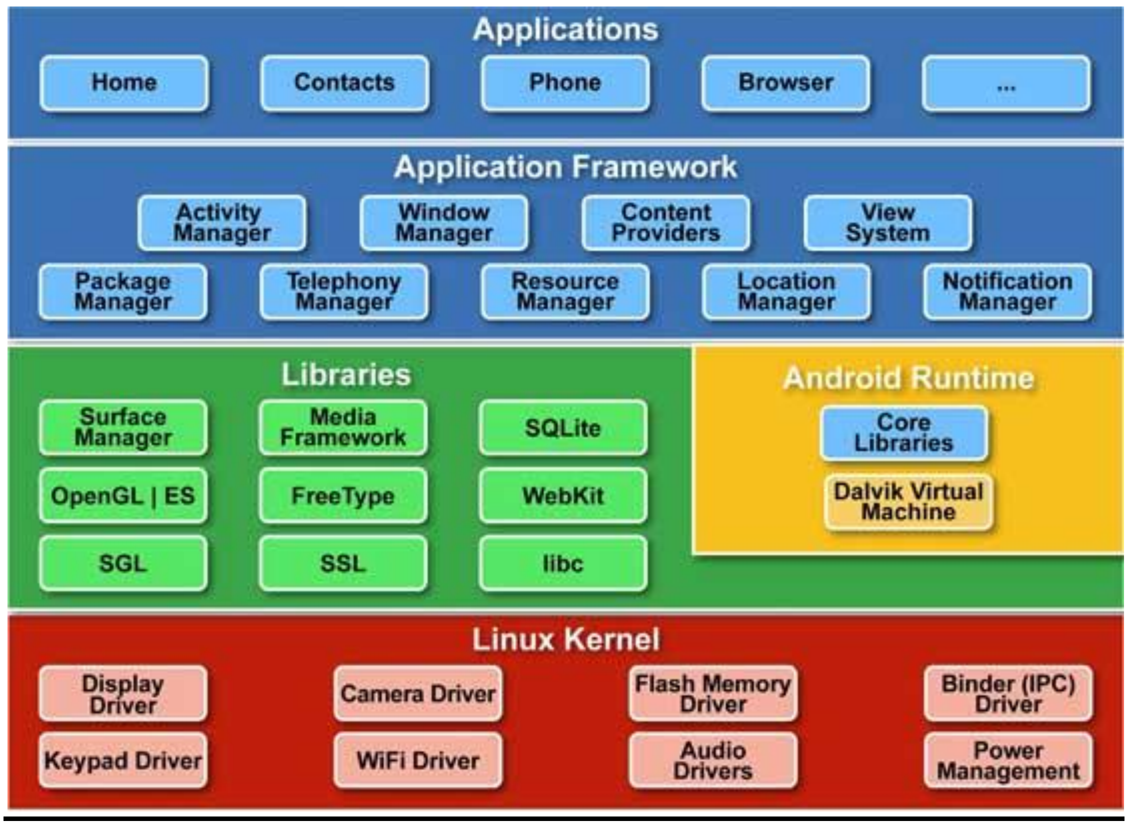


Fig 7. Android Architecture(Source:Tutorial point.in\android_arch\architecture.jpg)

3.1 Linux kernel

At the bottom of the layers is Linux - Linux 2.6 with approximately 115 patches. This provides basic system functionality like process management, memory management, device management like camera, keypad, display etc. Also, the kernel handles all the things that Linux is really good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware.

3.2 Libraries

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

3.3 Android Runtime

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

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

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

3.4 Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

3.5 Applications

You will find all the Android application at the top layer. You will write your application to be installed on this layer only. Examples of such applications are Contacts Books, Browser, Games etc.[11]

In next chapter, we are going to discuss about requirements related to the development of android applications.

CHAPTER 4
REQUIREMENTS

Development of an application for specific platform comes with its own specific requirements such as language requirements, Integrated Development Environment and other add-ons. For development of android application we have basic three requirements. These requirements are to be taken care of before starting the development of the applications.

4.1 Java Language:

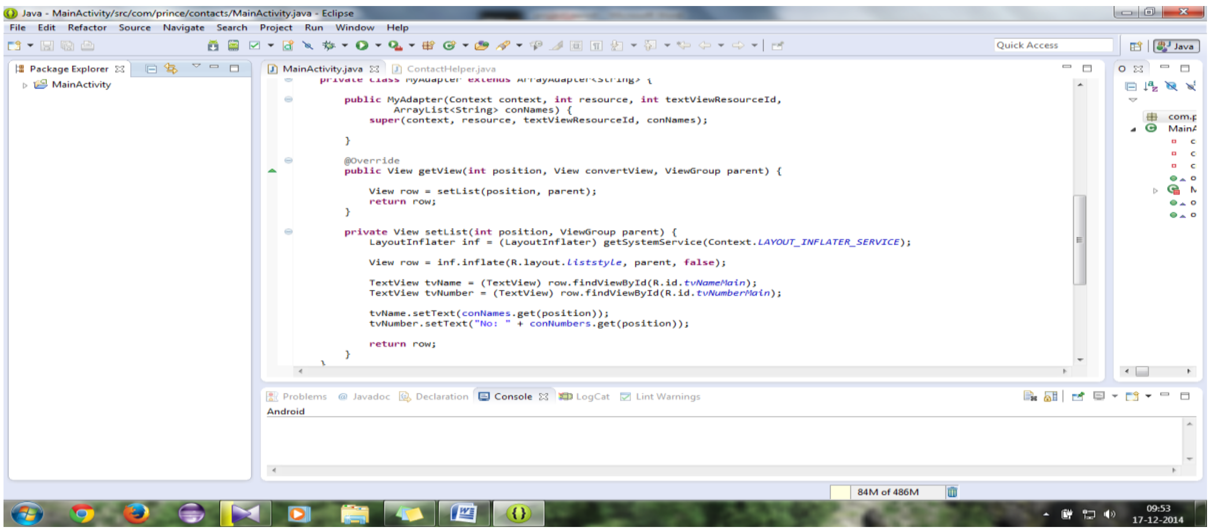
Writing in the Java programming language is the primary way to produce code that will be deployed as byte code in a Java Virtual Machine (JVM); byte code compilers are also available for other languages, including Ada, JavaScript, Python, and Ruby. In addition, several languages have been designed to run natively on the JVM, including Scala, Clojure and Groovy. Java syntax borrows heavily from C and C++, but object-oriented features are modeled after Smalltalk and Objective-C. Java eschews certain low-level constructs such as pointers and has a very simple memory model where every object is allocated on the heap and all variables of object types are references. Memory management is handled through integrated automatic garbage collection performed by the JVM.

Java is used to develop Android application. It runs in a Virtual Machine, so no need to recompile it for every phone out there. Large number of development tools for java.

4.2 Eclipse ADT(Android Development Environment)

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.
The initial codebase originated from IBM VisualAge. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.

Eclipse IDE is designed to provide an integrated environment in which to build Android applications. It is a freeware available to download.

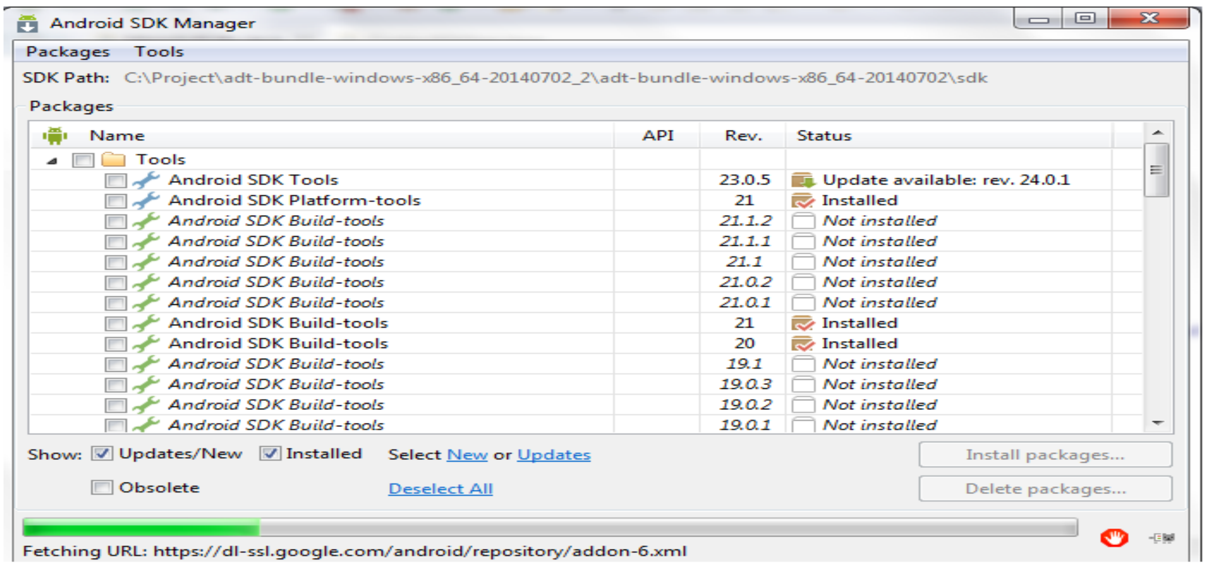


```
    private class MyAdapter extends ArrayAdapter<String> {

        public MyAdapter(Context context, int resource, int textViewResourceId,
                ArrayList<String> conNames) {
            super(context, resource, textViewResourceId, conNames);

        }

        @Override
        public View getView(int position, View convertView, ViewGroup parent) {

            View row = setList(position, parent);
            return row;
        }

        private View setList(int position, ViewGroup parent) {
            LayoutInflater inf = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);

            View row = inf.inflate(R.layout.liststyle, parent, false);

            TextView tvName = (TextView) row.findViewById(R.id.tvNameMain);
            TextView tvNumber = (TextView) row.findViewById(R.id.tvNumberMain);

            tvName.setText(conNames.get(position));
            tvNumber.setText("No: " + conNumbers.get(position));

            return row;
        }
```

Fig 8 Eclipse IDE in use

4.3 Android SDK/ADT

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).
SDK(Software Development Kit) Tools is a downloadable component for the Android SDK. It includes the complete set of development and debugging tools for the Android SDK. Android Development Tools (ADT) is a plug-in for the Eclipse IDE that is designed to give you a powerful, integrated environment in which to build Android applications



Android SDK Manager

Packages Tools

SDK Path: C:\Project\adt-bundle-windows-x86_64-20140702_2\adt-bundle-windows-x86_64-20140702\sdk

Packages

Name	API	Rev.	Status
Tools			
Android SDK Tools		23.0.5	Update available: rev. 24.0.1
Android SDK Platform-tools		21	Installed
Android SDK Build-tools		21.1.2	Not installed
Android SDK Build-tools		21.1.1	Not installed
Android SDK Build-tools		21.1	Not installed
Android SDK Build-tools		21.0.2	Not installed
Android SDK Build-tools		21.0.1	Not installed
Android SDK Build-tools		21	Installed
Android SDK Build-tools		20	Installed
Android SDK Build-tools		19.1	Not installed
Android SDK Build-tools		19.0.3	Not installed
Android SDK Build-tools		19.0.2	Not installed
Android SDK Build-tools		19.0.1	Not installed

Show: ✓ Updates/New ✓ Installed Select New or Updates Install packages...

Obsolete Deselect All Delete packages...

Fetching URL: https://dl-ssl.google.com/android/repository/addon-6.xml

Fig 9 Android Sdk manager

In the next chapter discussion will be focused on proposed work that is how to tackle the problem at hand and how to approach the problem.

CHAPTER 5
PROPOSED WORK

The applications to be created are battery manager, application manager and contact manager from there other applications can start forming and coalescing together. These sub applications are to create an environment that is easy to use as well as provide several functionalities to the user.

5.1 Basic components of the applications

- Battery manager will show current battery state of the battery and information about the battery.

- The application manager will show running application and installed applications, task killer will terminate running applications which will help in releasing device resources.

- Contact manager helps to manage the contacts on the device either by adding, deleting or by editing the current contacts.

- Note editor help to manage everyday notes.

These applications are to be created in Eclipse IDE under Android Application Project.
With Android ADT/SDK packages installed the Android applications can be created using GUI interface of Eclipse.

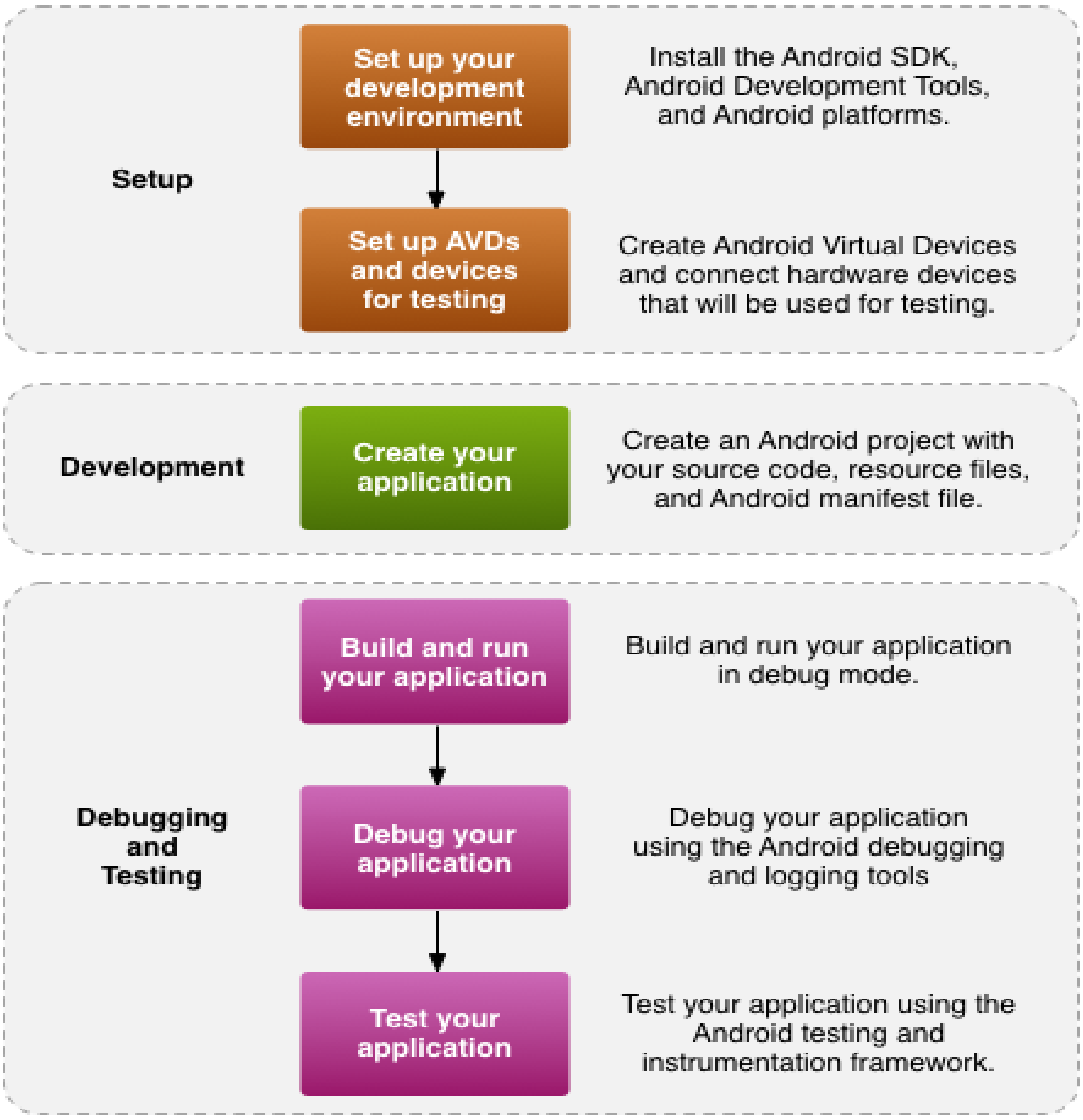

Setup
Set up your development environment
Install the Android SDK, Android Development Tools, and Android platforms.
Set up AVDs and devices for testing
Create Android Virtual Devices and connect hardware devices that will be used for testing.
Development
Create your application
Create an Android project with your source code, resource files, and Android manifest file.
Debugging and Testing
Build and run your application
Build and run your application in debug mode.
Debug your application
Debug your application using the Android debugging and logging tools
Test your application
Test your application using the Android testing and instrumentation framework.

Fig 10 The development model for Android applications(Source: wiki.dave.eu)

Fig 10 depicts the work flow for developing an android application in which we have 3 phases:

- Setting up application development softwares(IDE, SDK,AVD) in the system.
- Designing and development of the application. Setting up data items such as icons, thumbnails in file directory for use, creation of manifesta and writing code for the application.
- Testing and then debugging of the application created for the optimization.

5.2 The Flowcharts for Applications being created

1. Flowchart for Battery Manager Application:

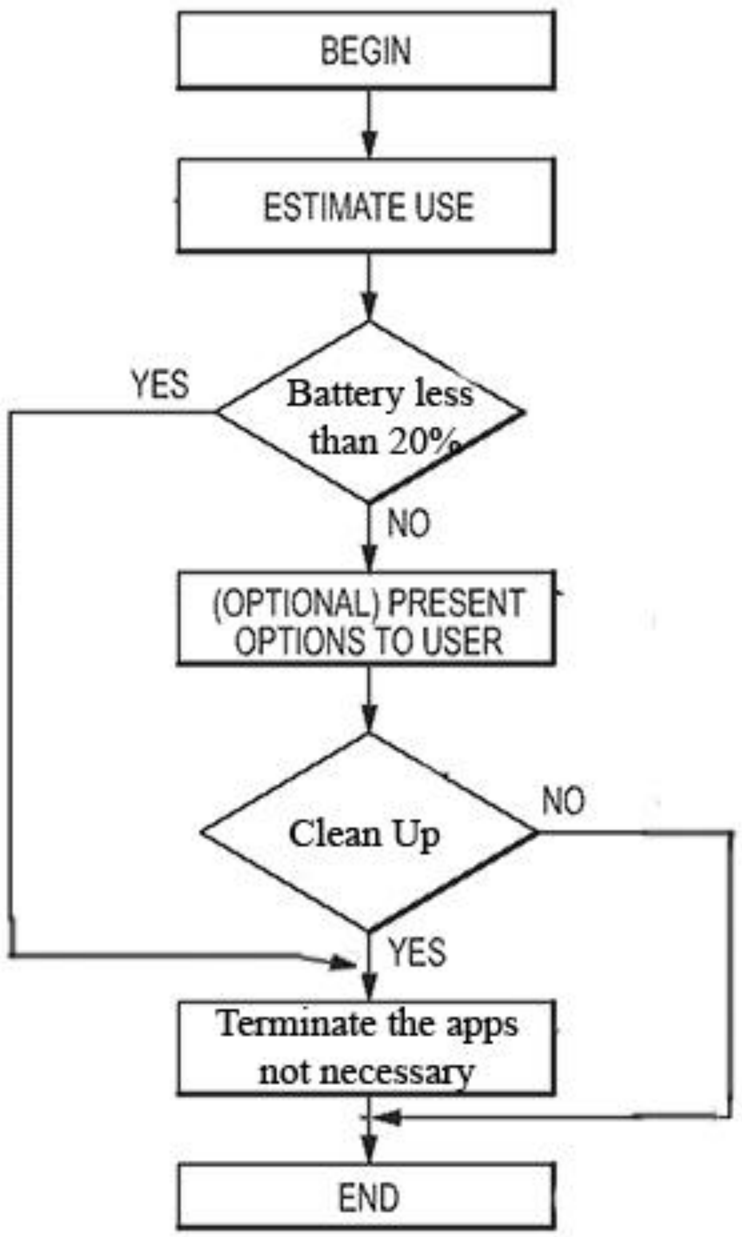

BEGIN
ESTIMATE USE
Battery less than 20%
YES
NO
(OPTIONAL) PRESENT OPTIONS TO USER
Clean Up
NO
YES
Terminate the apps not necessary
END

Fig 11 Flowchart Battery saver Application

Working of Battery manager application is based upon two criterias. The first criteria consists of if user selects to clean up all the applications which have been running in the background for more than an hour will be terminated. The second criteria is that if battery status falls below 20 % all the application which are not necessary for device functioning will be terminated.

2. Flowchart for Contact Manager Application:

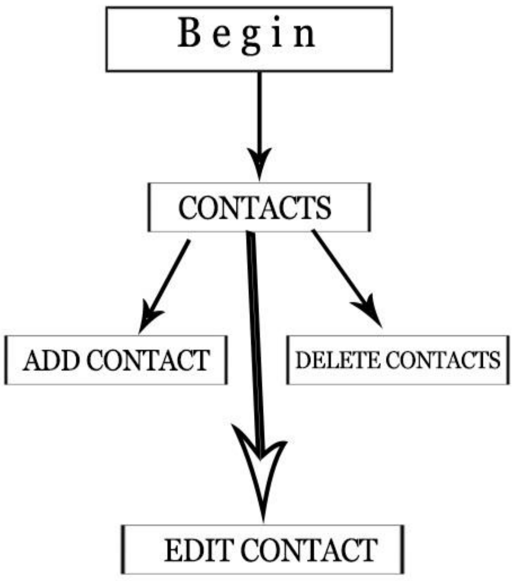


Fig 12 Flow Diagram for Contact Manager

Fig 12 depicts the flowchart of contact manger application in which a user is provided with 3 options when the application begins working that is **Add contact**, **Delete Contact** and **Edit contact**. From the options user can modify add or delete the existing contact.

3.Flowchart for Application Viewer + Task Killer Applications:

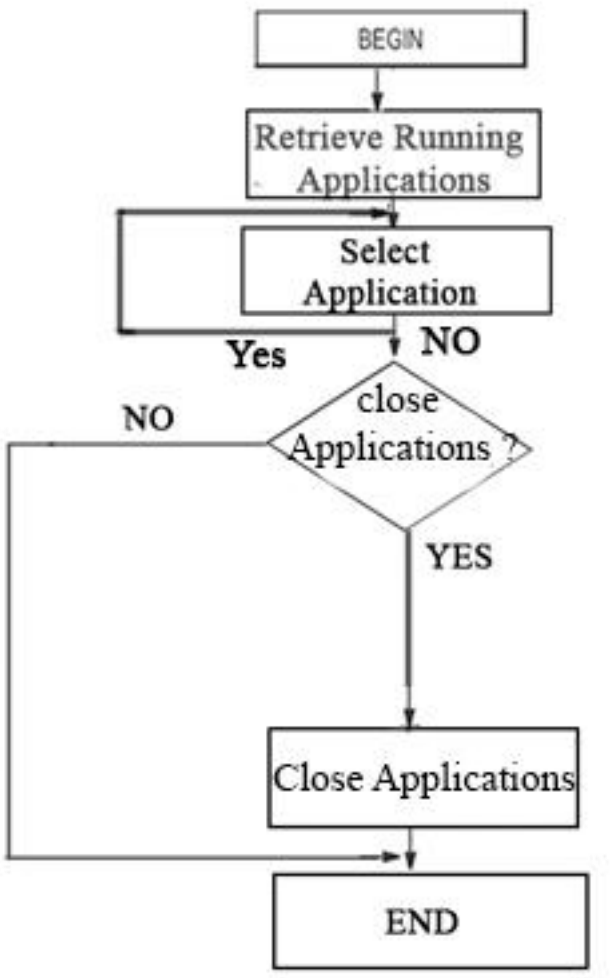


Fig 13 Flowchart for Application Viewer + Task Killer Applications

Fig 13 depicts the flowchart for Application Viewer+Task Killer the application which can be used to see various applications running on the device and the user is provided with an option to close the applications if he/she wants to.

In next chapter Emulation environment will be discussed in detail. Topics of discussion will be Android Virtual Devices(AVDs) and the testing devices used for the project.

CHAPTER-6
EMULATOR ENVIRONMENT

Testing is a very crucial part of application development. Two modes of testing are available for android application testing one of them is to transfer the application.apk file to a device and test it, the other way is to use Android virtual Devices.

6.1 AVD(Android Virtual Devices)

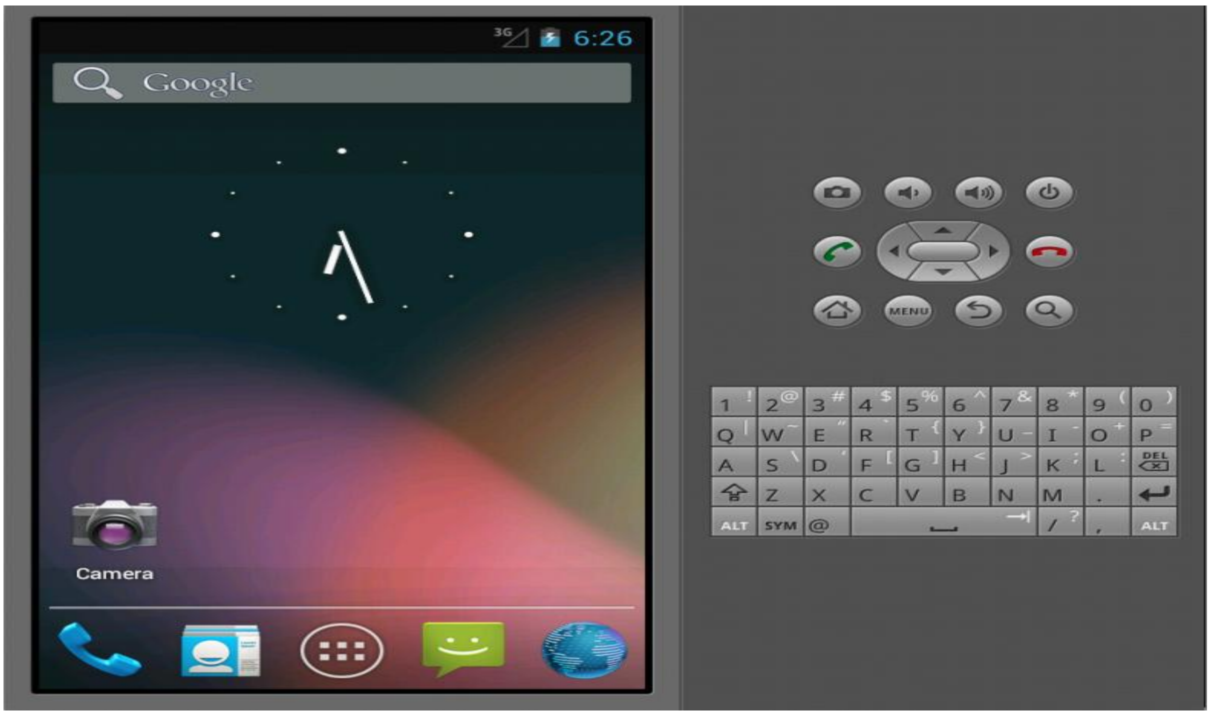


Fig 14 Android Emulator

An Android Virtual Device (AVD) is an emulator configuration that lets you model an actual device by defining hardware and software options to be emulated by the Android Emulator. The easiest way to create an AVD is to use the graphical AVD Manager, which you launch from Eclipse by clicking Window > AVD Manager.
Once the AVD is launched, you can see how it gives the complete look and feel of a real Android-based mobile phone, complete with keyboard and multi-touch support. It can also be used in a variety of configurations to test your app, such as landscape/portrait mode, network strength, and roaming network, etc. All of these options can be configured using the AVD manager. The AVD is self-sufficient to emulate different devices available. You can create different AVDs for different configurations and test your application on each of them to make sure it is compatible across device types.

6.2 Test Devices

1. Micromax Canvas Turbo Mini A200

- RAM 1GB
- Battery Li-Ion, 1800 mAh.

2.HTC Wildfire S

- RAM 512MB
- Battery: Li-Ion 1230 mAh.

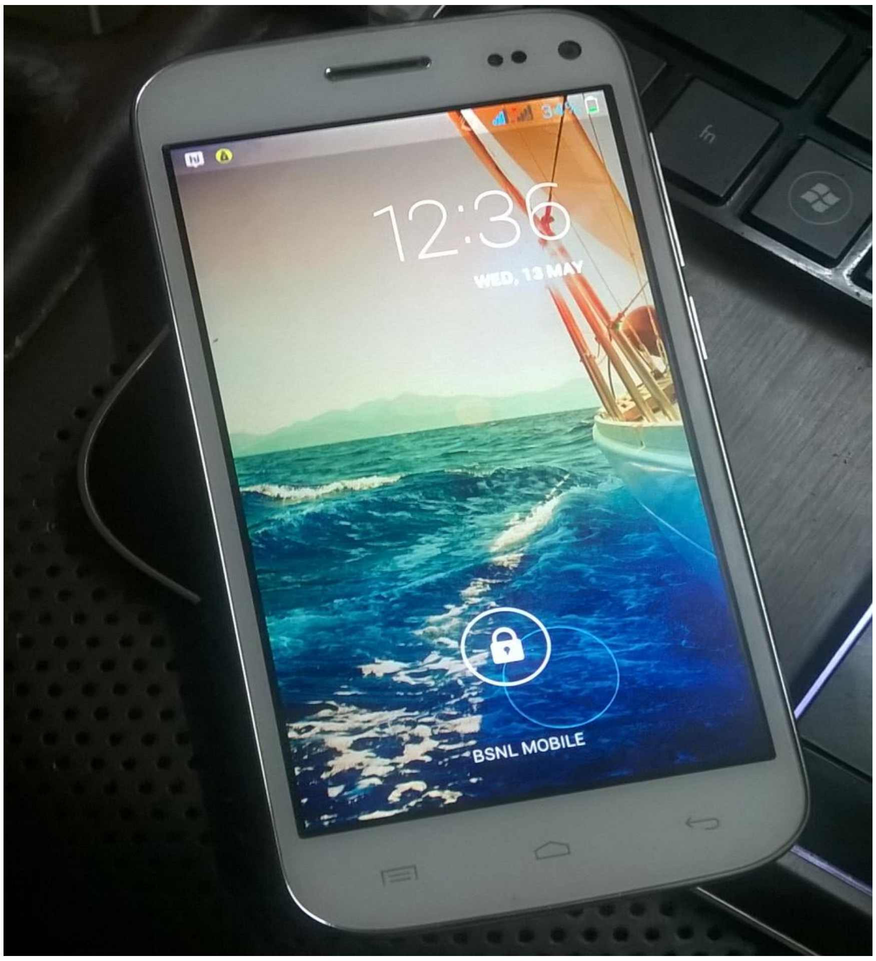


Fig 15 Micromax Canvas Turbo Mini A200

In next chapter Output screens of the applications will given.

CHAPTER-7
OUTPUTS

Four Applications were developed, the output snapshots of the four applications are provided. The snapshots depicts the applications in their running state.

7.1 Battery Manager

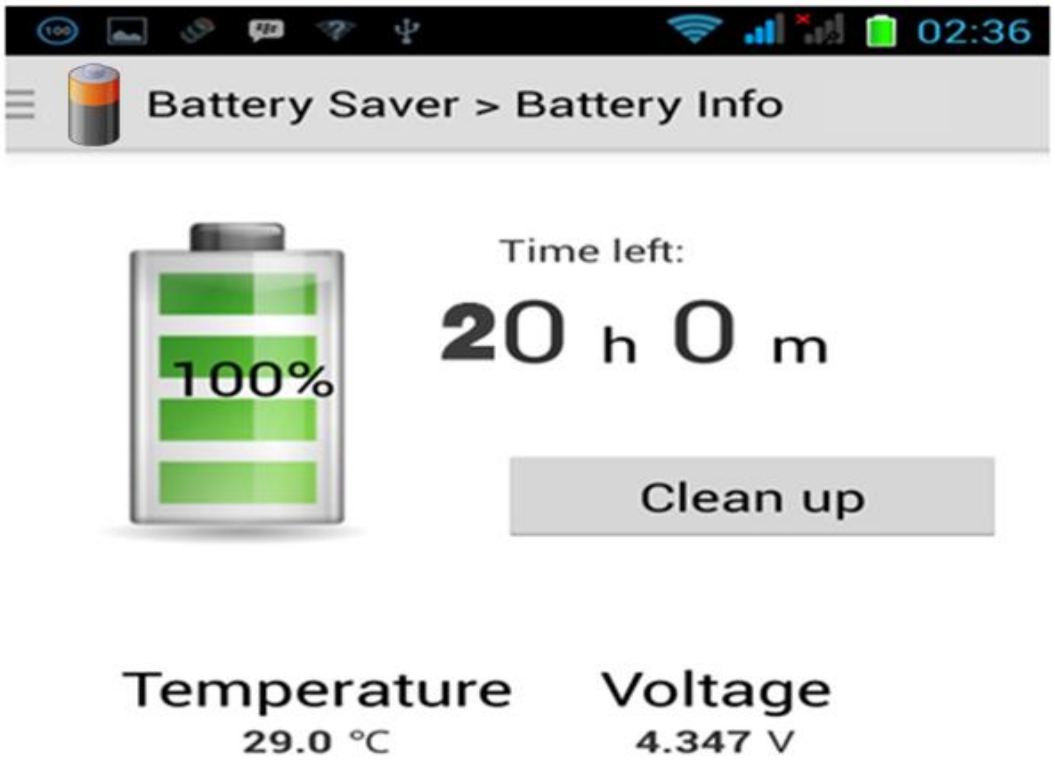
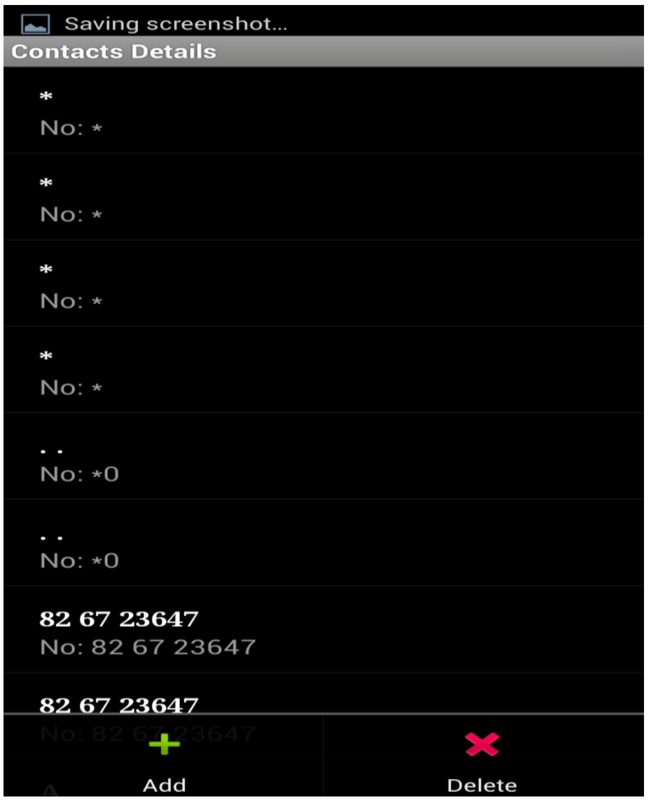


Fig 16 Battery Manager

Fig 16 is the output snapshot of Battery manager application showing various parameter such as temperature of the battery and voltage utilized by the battery. It als odepicts the time left till the battery can be used. Also an option to clean up is given so as to close down applications working in backgroud with not much use or not been used for some time.

7.2 Contact manager


Saving screenshot...

Contacts Details

*
No: *

*
No: *

*
No: *

*
No: *

. .
No: *0

. .
No: *0

82 67 23647
No: 82 67 23647

82 67 23647
No: 82 67 23647

Fig 17 Contacts Manager

Fig 17 is the output snapshot of the contact manager application depicting its three functionalities that are adding, deleting and managing contacts as per the user preference.
The application can be used to edit contacts or just to which contacts are stored on the device.

7.3 Application Viewer + Task Killer

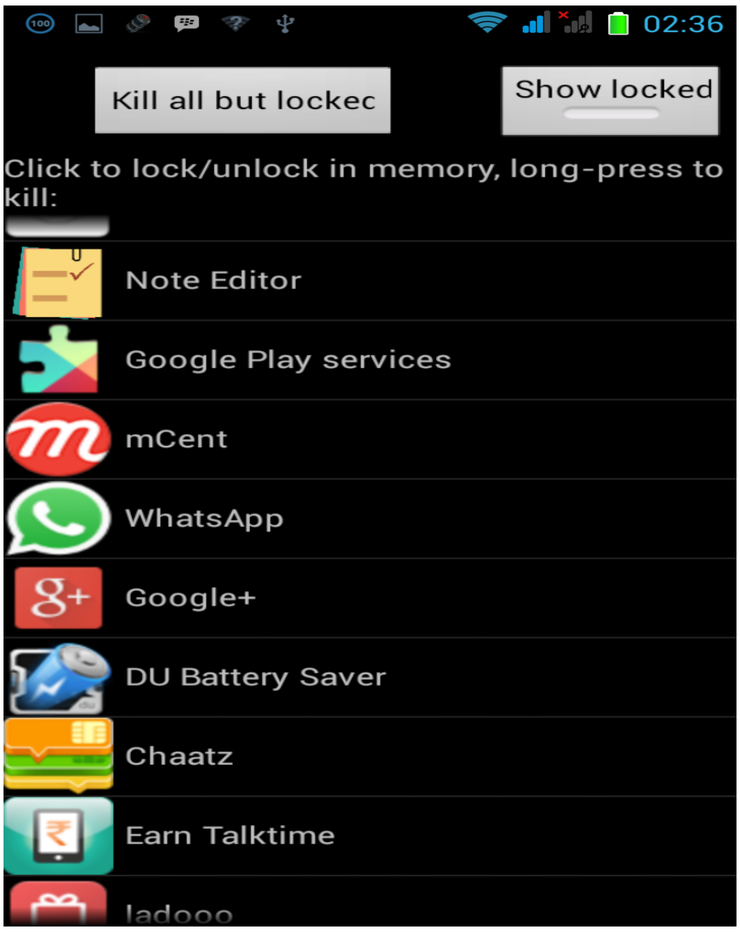
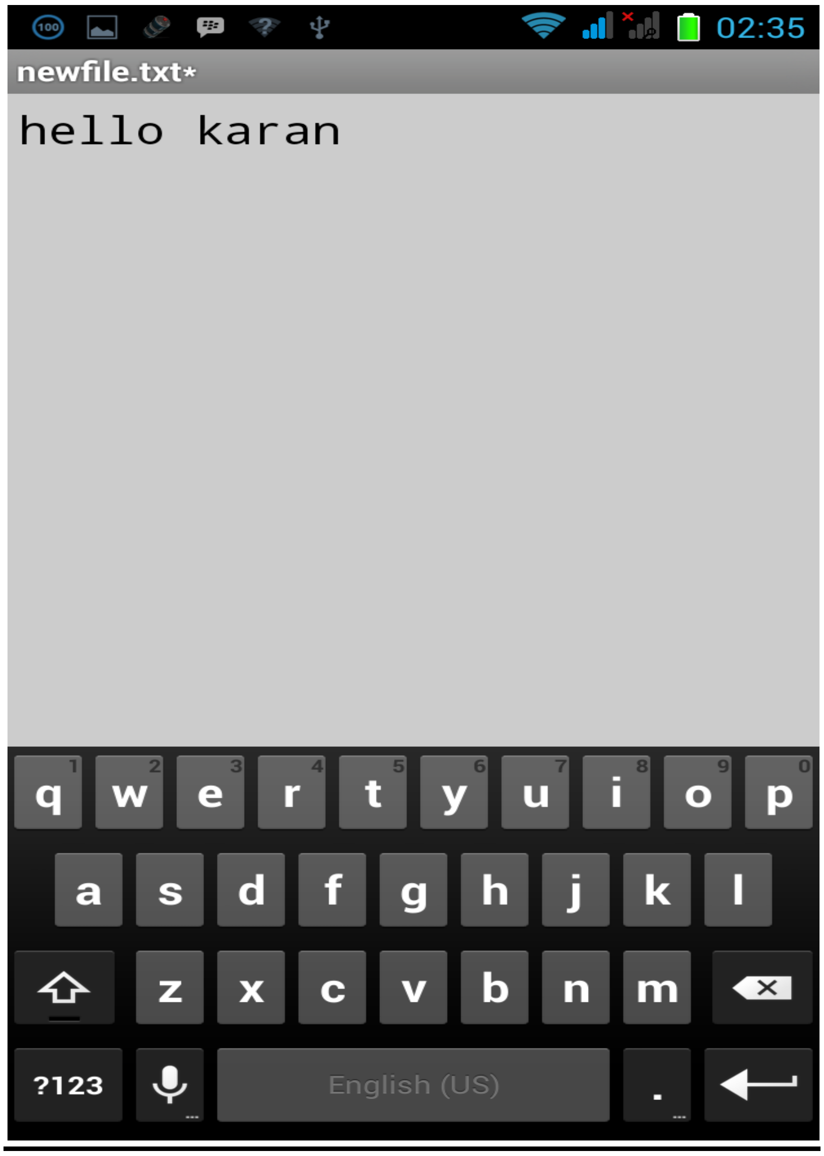


Fig 18 Application viewer +Task killer

Fig 18 is the snapshot of application viewer+Task Killer application which can be used to view various applications running on the device at any given time. User can lock the applications he/she does not want to be destroyed.

7.4 Note Editor



hello karan

Fig 19 Notes Editor

Fig 19 is the screenshot of the Note editor application in which user can create various notes of his/her requirements for later use and can save these notes in the devices memory. Note editor provides various fonts and colors for text to be written.

In next chapter the results of the project will be discussed.

CHAPTER-8
RESULTS

Results are generated in graphical format to compare the performance of the developed applications within various parameters. Also to compare the application with a third party application(of similar type).

8.1 Comparison of three applications before and after Installing developed applications.

1. Comparison of three applications before and after Installing developed applications:

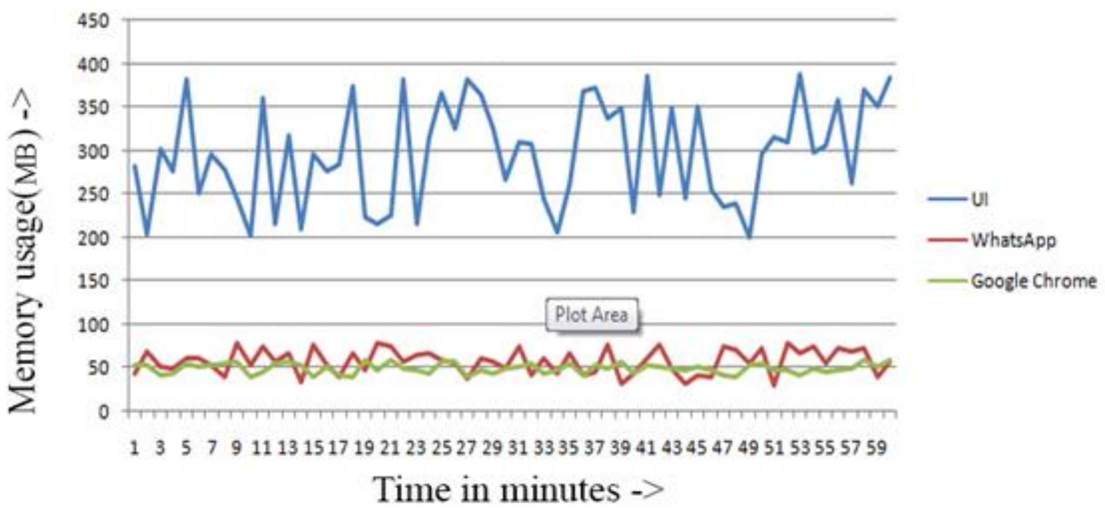


Fig 20 Comparison between three running applications before installing developed applications.

From the graph it can be seen that a lot of resources are being used up by these 3 applications

2. Comparison of three applications after Installing applications:

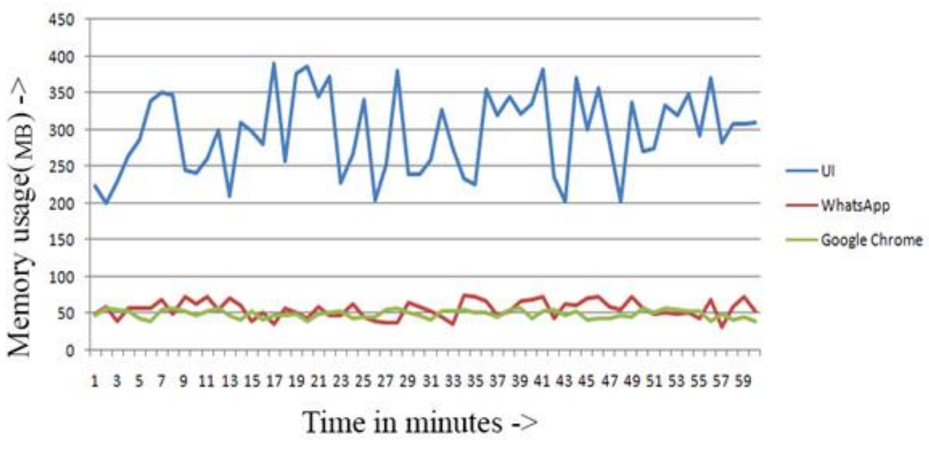


Fig 21 Comparison between three running applications after installing developed applications.

Fig 25 and fig 26 are depicting the use of memory by the device before and after installation of developed application. For test cases 3 different applications were selected namely User Interface(UI), WhatsApp , Google Chrome browser for testing the application as to assess how much performance gain do we get after installing the application.

The performance gain is not very high, application is able to save around 5mb of memory more as compared to the case when no application is installed. The applications worked smoothly on the testing devices with less than 3 crashes for all the time the application was used. Moreover the gain though not very significant the application was able to perform smoothly.

8.2 Comparison of three applications after Installing third party application:

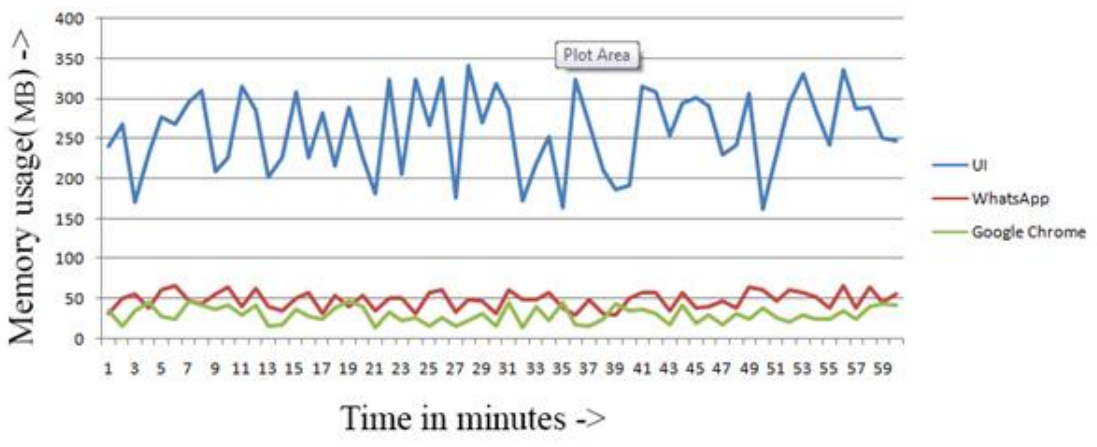


Fig 22 Comparison of three applications after Installing third party application

Fig 27 Depicts the graph showing usage of memory resources when a third party Battery management application was installed(that is DU battery Saver). The application enhanced the efficiency and performance of the device quite significantly.

It can be seen from the graph that in case of User Interface about 50mb of memory is being saved and in case of both the other applications the memory being saved is about 5MB to 10MB, leading to better performance of the device.

8.3 Line chart depicting battery working (in hrs) before and after using application:

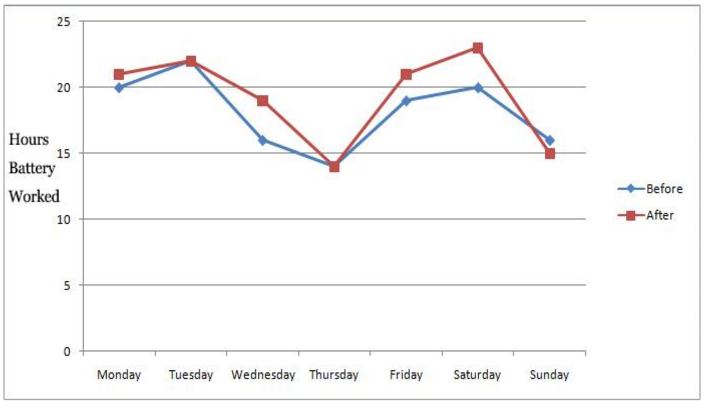


Fig 23 Line chart depicting battery working (in hrs) before and after using application.

Fig 26 depicts the battery performance of the device with and without the developed applications installation. The test were conducted over the week and seen how long the battery lasted. More hours lead to better performance.

Various applications were run simultaneously on the device for both before and after cases to create similar test environments to get the data that for how long the battery lasted. From the graph, it is seen that there is very slight difference in battery performance after installing the application as only very slight difference of battery

CHAPTER-9
CONCLUSION AND FUTURE WORK

9.1 CONCLUSION:

Application is showing a lot of potential and can render helpful to user in various ways either by letting the user control the phone through one application or by helping in improvement of phones performance.The applications are performing smoothly on various devices on which it was tested. Applications are quite stable and are not hindering the workings of the devices on which it was tested. The performance and efficiency gains were not high. The gain from using the applications was not more than that of 2%-3%. The applications can further be enhanced overtime to the level of performance gains provided by third party paid applications. Though getting various applications in single application is something new which has given a slight edge to the application as it makes performing several tasks easier and smooth.

9.2 Future Scope

In future more extensive work can be done on these applications to further enhance the workings, optimization and feel of the applications. In future this application can be expanded to incorporate even more sub applications to improve the user experience, make work space more easy to use and to enhance the devices efficiency. Moreover, the application can be expanded to incorporate not only basic but also other more specific requirements.

REFERENCES

BOOKS:

[1]. Android in Action by W. Frank Ableson.

[2]Development of Android Applications by Edison Jimenez and Kyle Davidson

[4]. Android OS: A robust, free, open-source operating system for mobile devices by Paul Michael Kilgo

Websites:

[4]. www.Developer.android.com/

[5].http://www.theappguruz.com/

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

[7]. Android API: http://developer.android.com/reference/packages.html

[8]. Android User Interfaces: http://developer.android.com/guide/topics/ui/index.html

[9]. The Java Tutorials: http://docs.oracle.com/javase/tutorial/

[10]. Sample Source Code: http://developer.android.com/resources/samples/get.html

Additional Resource References:

[1].developer.android.com/about/
[2].http://www.techtimes.com/articles/12006/20140803/android-85-global-smartphone-marketshare-report.htm
[3].http://www.idc.com/getdoc.jsp
[4]. http://www.idc.com/getdoc.jsp
[5].http://www.appsfreedom.com/best-way-extend-enterprise-asset-management-apps-mobile-devices/
[6]. http://developer.android.com/training/basics/activity-lifecycle
[7]. https://play.google.com/store/apps/details?id=com.latedroid.juicedefender&hl=en
[8].https://play.google.com/store/apps/details?id=com.rhythm.hexise.task&hl=en
[9].https://play.google.com/store/apps/details?id=com.contapps.android&hl=en
[10].https://play.google.com/store/apps/details?id=com.automattic.simplenote&hl=en
[11]. http://www.tutorialspoint.com/android/android_architecture.htm

APPENDIX
Codes

- **Battery manager:**

```
package com.manager_battery.batterydata;

import android.os.BatteryManager;
import android.os.Bundle;
import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.util.Log;
import android.widget.TextView;
public class MainActivity extends Activity {
	TextView textBatteryLevel = null;
	String batteryLevelInfo = "Battery Level";
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		textBatteryLevel = (TextView) findViewById(R.id.txtBatteryInfo);
		registerBatteryLevelReceiver();
	}
	@Override
	protected void onDestroy() {
		unregisterReceiver(battery_receiver);
		super.onDestroy();
	}
	private BroadcastReceiver battery_receiver = new BroadcastReceiver() {
		@Override
		public void onReceive(Context context, Intent intent) {
			boolean isPresent = intent.getBooleanExtra("present", false);
			String technology = intent.getStringExtra("technology");
			int plugged = intent.getIntExtra("plugged", -1);
			int scale = intent.getIntExtra("scale", -1);
			int health = intent.getIntExtra("health", 0);
			int status = intent.getIntExtra("status", 0);
			int rawlevel = intent.getIntExtra("level", -1);
			int voltage = intent.getIntExtra("voltage", 0);
			int temperature = intent.getIntExtra("temperature", 0);
			int level = 0;

			Bundle bundle = intent.getExtras();
			Log.i("BatteryLevel", bundle.toString());
			if (isPresent) {
				if (rawlevel >= 0 && scale > 0) {
					level = (rawlevel * 100) / scale;
				}
				String info = "Battery Level: " + level + "%\n";
				info += ("Technology: " + technology + "\n");
				info += ("Plugged: " + getPlugTypeString(plugged) + "\n");
				info += ("Health: " + getHealthString(health) + "\n");
				info += ("Status: " + getStatusString(status) + "\n");
				info += ("Voltage: " + voltage + "\n");
				info += ("Temperature: " + temperature + "\n");
				setBatteryLevelText(info + "\n\n" + bundle.toString());
			} else {
				setBatteryLevelText("Battery not present!!!");
			}
		}
	};
	private String getPlugTypeString(int plugged) {
		String plugType = "Unknown";
		switch (plugged) {
		case BatteryManager.BATTERY_PLUGGED_AC:
			plugType = "AC";
			break;
		case BatteryManager.BATTERY_PLUGGED_USB:
```

```
                        plugType = "USB";
                        break;
                }
                return plugType;
        }
        private String getHealthString(int health) {
                String healthString = "Unknown";
                switch (health) {
                case BatteryManager.BATTERY_HEALTH_DEAD:
                        healthString = "Dead";
                        break;
                case BatteryManager.BATTERY_HEALTH_GOOD:
                        healthString = "Good";
                        break;
                case BatteryManager.BATTERY_HEALTH_OVER_VOLTAGE:
                        healthString = "Over Voltage";
                        break;
                case BatteryManager.BATTERY_HEALTH_OVERHEAT:
                        healthString = "Over Heat";
                        break;
                case BatteryManager.BATTERY_HEALTH_UNSPECIFIED_FAILURE:
                        healthString = "Failure";
                        break;
                }

                return healthString;
        }

        private String getStatusString(int status) {
                String statusString = "Unknown";

                switch (status) {
                case BatteryManager.BATTERY_STATUS_CHARGING:
                        statusString = "Charging";
                        break;
                case BatteryManager.BATTERY_STATUS_DISCHARGING:
                        statusString = "Discharging";
                        break;
                case BatteryManager.BATTERY_STATUS_FULL:
                        statusString = "Full";
                        break;
                case BatteryManager.BATTERY_STATUS_NOT_CHARGING:
                        statusString = "Not Charging";
                        break;
                }

                return statusString;
        }

        private void setBatteryLevelText(String text) {
                textBatteryLevel.setText(text);
        }

        private void registerBatteryLevelReceiver() {
                IntentFilter filter = new IntentFilter(Intent.ACTION_BATTERY_CHANGED);

                registerReceiver(battery_receiver, filter);
        }
}
```

• APPLICATION VIEWER +TASK KILLER

```
package com.Karan.appkiller;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.ArrayList;
import java.util.List;

import com.Karan.appkiller.R;

import android.app.ActivityManager;
import android.app.ListActivity;
import android.app.ActivityManager.RunningAppProcessInfo;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.drawable.Drawable;
import android.net.Uri;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.ContextMenu;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.view.Window;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.ToggleButton;

public final class AppKiller extends ListActivity implements OnClickListener {
		private static final String PREFS_FILENAME = "GenKiller.conf";
		private static final String ShowLockedKey = "ShowLocked";
		private static final String PREFS_WHITE_LIST = "WhiteList";
		private static final int MENU_KILL = 0;
		private SharedPreferences prefs;
		private List<String> runningPackages = new ArrayList<String>();
		private List<String> reservedPackages = new ArrayList<String>();
		private List<String> whiteListPackages = new ArrayList<String>();
		private ActivityManager activityManager;
		private PackageAdapter adapter;
		private boolean showLocked;
		private Method killMethod;

		@Override
		public void onCreate(Bundle savedInstanceState) {
				super.onCreate(savedInstanceState);
				this.requestWindowFeature(Window.FEATURE_NO_TITLE);
				this.setContentView(R.layout.main);
				this.activityManager = (ActivityManager)this.getSystemService(ACTIVITY_SERVICE);
				this.prefs = this.getSharedPreferences(PREFS_FILENAME, MODE_PRIVATE);

				if (this.reservedPackages.isEmpty()) {
						this.reservedPackages.add("system");
						this.reservedPackages.add("com.google.process.gapps");
						this.reservedPackages.add("android.process.acore");
						this.reservedPackages.add("android.process.media");
				}

				this.loadWhiteList();
				this.initializeButtons();
				this.initializeKillMethod();
		this.adapter = new PackageAdapter(
				this.getApplicationContext(),
				this.runningPackages,
				this.whiteListPackages,
				this.showLocked);
		this.setListAdapter(this.adapter);
		ListView listView = this.getListView();
		listView.setTextFilterEnabled(false);
		this.registerForContextMenu(listView);
	}

	private void initializeButtons() {
		Button killButton = (Button) this.findViewById(R.id.btnKill);
		killButton.setOnClickListener(this);

		this.showLocked = this.prefs.getBoolean(ShowLockedKey, true);
		final ToggleButton showHideButton = (ToggleButton) this.findViewById(R.id.btnShowHide);
		showHideButton.setChecked(showLocked);
		showHideButton.setOnClickListener(this);
	}
```

- **Contact manager:**

```
package com.cmanager.contacts;
import java.util.ArrayList;
import android.app.ListActivity;
import android.content.Context;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;
public class MainActivity extends ListActivity {
	private ArrayList<String> conNames;
	private ArrayList<String> conNumbers;
	private Cursor crContacts;
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		conNames = new ArrayList<String>();
		conNumbers = new ArrayList<String>();
		crContacts = ContactHelper.getContactCursor(getContentResolver(), "");
		crContacts.moveToFirst();
		while (!crContacts.isAfterLast()) {
			conNames.add(crContacts.getString(1));
			conNumbers.add(crContacts.getString(2));
			crContacts.moveToNext();
		}
		setListAdapter(new MyAdapter(this, android.R.layout.simple_list_item_1,
					R.id.tvNameMain, conNames));
	}
	private class MyAdapter extends ArrayAdapter<String> {
		public MyAdapter(Context context, int resource, int textViewResourceId,
					ArrayList<String> conNames) {
			super(context, resource, textViewResourceId, conNames);

		}
		@Override
		public View getView(int position, View convertView, ViewGroup parent) {
			View row = setList(position, parent);
			return row;
		}

		private View setList(int position, ViewGroup parent) {
			LayoutInflater inf = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);

			View row = inf.inflate(R.layout.liststyle, parent, false);

			TextView tvName = (TextView) row.findViewById(R.id.tvNameMain);
			TextView tvNumber = (TextView) row.findViewById(R.id.tvNumberMain);

			tvName.setText(conNames.get(position));
			tvNumber.setText("No: " + conNumbers.get(position));

			return row;
		}
	}

	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		MenuInflater imf = getMenuInflater();
		imf.inflate(R.menu.main, menu);
		return true;
	}

	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		if (item.getItemId() == R.id.item1) {
			Intent intent = new Intent(MainActivity.this, AddContact.class);
			startActivity(intent);
		} else if (item.getItemId() == R.id.item2) {
			Intent intent = new Intent(MainActivity.this, DeleteContacts.class);
			startActivity(intent);
		}
		return super.onOptionsItemSelected(item);
	}
}
```

- **Note Editor**

```
package com.karan.noteeditor;

import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Locale;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.SearchManager;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Typeface;
import android.os.Bundle;
import android.os.Handler;
import android.preference.PreferenceManager;
import android.provider.SearchRecentSuggestions;
import android.text.Editable;
import android.text.InputType;
import android.text.TextWatcher;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

import com.karan.noteeditor.EditTextSelectable.OnSelectionChangedListener;
import com.maxistar.textpad.R;

public class EditorActivity extends Activity {
private static final int OPEN_FILE = 1;
private static final int SAVE_FILE = 2;
//private static final int SETTINGS = 3;
private static final int NEW_FILE = 4;
private static final int SAVE_AS = 5;
private static final int MENU_SEARCH = 6;

private static final int REQUEST_OPEN = 1;
private static final int REQUEST_SAVE = 2;
private static final int REQUEST_SETTINGS = 3;

private static final int DO_NOTHING = 0;
private static final int DO_OPEN = 1;
private static final int DO_NEW = 2;

private EditTextSelectable mText;
private TextWatcher watcher;
String filename = TPStrings.EMPTY;
boolean changed = false;

private int open_when_saved = DO_NOTHING; // to figure out better way

Handler handler = new Handler();

static int selectionStart = 0;


/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);

mText = (EditTextSelectable) this.findViewById(R.id.editText1);
//mScrollView = (ScrollView) this.findViewById(R.id.scroll);

applyPreferences();

if (savedInstanceState!=null){
restoreState(savedInstanceState);
}
else {
Intent i = this.getIntent();
if (TPStrings.ACTION_VIEW.equals(i.getAction())) {
android.net.Uri u = i.getData();
openNamedFile(u.getPath());


}
}

watcher = new TextWatcher() {
@Override
public void afterTextChanged(Editable s) {
// TODO Auto-generated method stub
}

@Override
```

```
public void beforeTextChanged(CharSequence s, int start, int count,
int after) {
// TODO Auto-generated method stub
}

@Override
public void onTextChanged(CharSequence s, int start, int before,
int count) {
if (!changed) {
changed = true;
updateTitle();
}
}
};
//mText.invalidate(); //
handler.postDelayed(new Runnable(){
@Override
public void run() {
mText.addTextChangedListener(watcher);

mText.addOnSelectionChangedListener(new OnSelectionChangedListener(){

@Override
public void onSelectionChanged(int selStart, int selEnd) {
// TODO Auto-generated method stub
selectionStart = mText.getSelectionStart();
}

});
```