Development of a Messenger Based on Android

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

Bachelor of Technology.

in

Computer Science and Engineering

Under the Supervision of

Mr. Amol Vasudeva

Senior Lecturer, CSE & IT

By

Abhik Mittal (091210)

Garima Gautam (091213)

Mahak Verma (091224)

to

JaypeeUniversity of Information and Technology

Waknaghat, Solan– 173234, Himachal Pradesh

CERTIFICATE

This is to certify that project report entitled "**Development of a Messenger Based on Android**", submitted by **Abhik Mittal (091210), Garima Gautam (091213), Mahak Verma (091224)** in partial fulfillment for the award of degree of Bachelor of Technology in Computer Science Engineering in 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:

Amol Vasudeva

Senior Lecturer, CSE & IT

ACKNOWLEDGEMENT

No venture can be completed without the blessings of the Almighty. We consider it our bounded duty to bow to Almighty whose kindness has always inspired us on the right path. We would like to take the opportunity to thank all the people who helped us in any form in the completion of our project "**Development of a Messenger Based on Android**".

It would not have been possible to see through the project without the guidance and constant support of our project guide Mr. Amol Vasudeva. For his coherent guidance, careful supervision, critical suggestions and encouragement we feel fortunate to be taught by him.

We would express our gratitude towards Brig. (Retd) Balbir Singh, Director JUIT, for having trust in us. We owe our heartiest thanks to Brig. (Retd.) Prof.S.P. Ghrera (HOD, CSE & IT Dept.), who has always inspired us to work towards our aim. Also, we are grateful to the whole faculty who helped us in making us believe that we could do this project.

We extend our gratitude towards Mr. Anshul Soon for helping us with the laboratory work. We are grateful to them for the support they provided us in doing things at our pace and for being patient for our mistakes.

Date:

Abhik Mittal

Garima Gautam

Mahak Verma

Table of Contents

List of Figures

ABSTRACT

In this fast and furious era ubiquitous computing is growing so extensively and every manual works are shifting towards computerized work. People don’t want to invest their time in processing they just want to give an input and take an output immediately. And by the help of machines we can reduce the processing time.

Nowadays our lifestyles have become so fast that we tend to miss many of our important events in our lives. Relationships are very important and they need constant refueling to build and maintain them. We generally require greeting our friends and family members on their birthdays, anniversaries or on some festive occasions. You may also require sending messages to your colleagues or to your boss. Due to busy work schedules or due to some other engagements, you may forget to send these messages to your loved ones.
MY MESSENGER is an app that helps you to send messages and emails to desired contacts and send your location to them too. It helps you to save the messages in advance and automatically send SMS messages the preset date to the desired recipients. Now no need to worry about remembering to send SMS messages and finally forgetting to send them as this application will certainly do it all on your behalf.

Automate the recurring events and tasks right on your mobile phone. Even if your phone is not with you it is still there in your alternative calendar. Think of it as your very own “private secretary” to schedule future text messages in a simple yet efficient manner.
In a single application you can send SMS, e-mail and your location to a desired contact and no need to switch applications.

Chapter 1
INTRODUCTION

In the era of smartphones and busy people, we need a messenger that helps people save time and their effort. My Messenger is an application that will help the users to send SMS and e-mails from the same application and schedule SMS for sending at future times too. There are times when people forget important events of their loved ones because of the busy schedule and this is the time when this application will help the users to not lose track of important events.

It is undeniable that every human heart murmurs "Hey I am not wired to remember every single thing" but let's admit it, knowingly or unknowingly you have disappointed a loved one or messed up an important business event. We do set calendar events and reminders, but does that really ensure that task will be completed? Actually, this is not the case. This Messenger will help us enable to schedule our messages or emails to be sent on a reset date. The contacts etc. can be synced with the application easily.

1.1 Features of the Messenger

1.1.1. SMS Sender:

This is a feature in the application that will help the user to send a SMS to a desired phone number. This includes three conditions, whether we want to directly send it or save and send in the database of the phone or send it via the messaging app of the phone.

1.1.2. Email Sender:

This is a feature in the application that enables the user to send an email to a desired email address.

1.1.3. SMS Scheduler:

SMS Scheduler is a simple tool for automatic SMS sending with chosen frequency.

1.1.4. Send Your Location:

This feature enables the user to send his location to any desired contact and receive his/her location. The receiver will receive the location in the form of a SMS.

1.1.5. Show Your Location:

The user will get to know his location using this option. He can see the nearest places on the map.

1.1.6. See Contacts:

The user can view his phonebook using this option.

ASSUMPTIONS AND DEPENDENCIES

- The user of the messaging application has a mobile device running Android 2.1+.
- The user of the messaging application has a smart phone with SMS and data transferring available.

MOTIVATION

- The need to keep up with our relations along with our busy schedules.
- A common human tendency is to forget. Once we remember we need to make a note of it.
- In the fast running world today, everyone has a smartphone. So its the best way to relate to a common man.

1.2 <u>Why Android is better?</u>

- ***Applications***

<u>Google applications</u>

Android includes most of the time many Google applications like Gmail, YouTube or Maps.

These applications are delivered with the machine most of the time, except in certain cases, such as some phones running android on which the provider has replaced Google applications by its own applications.

- ***Widgets***

With android, it is possible to use widgets which are small tools that can most often get information. These widgets are directly visible on the main window.

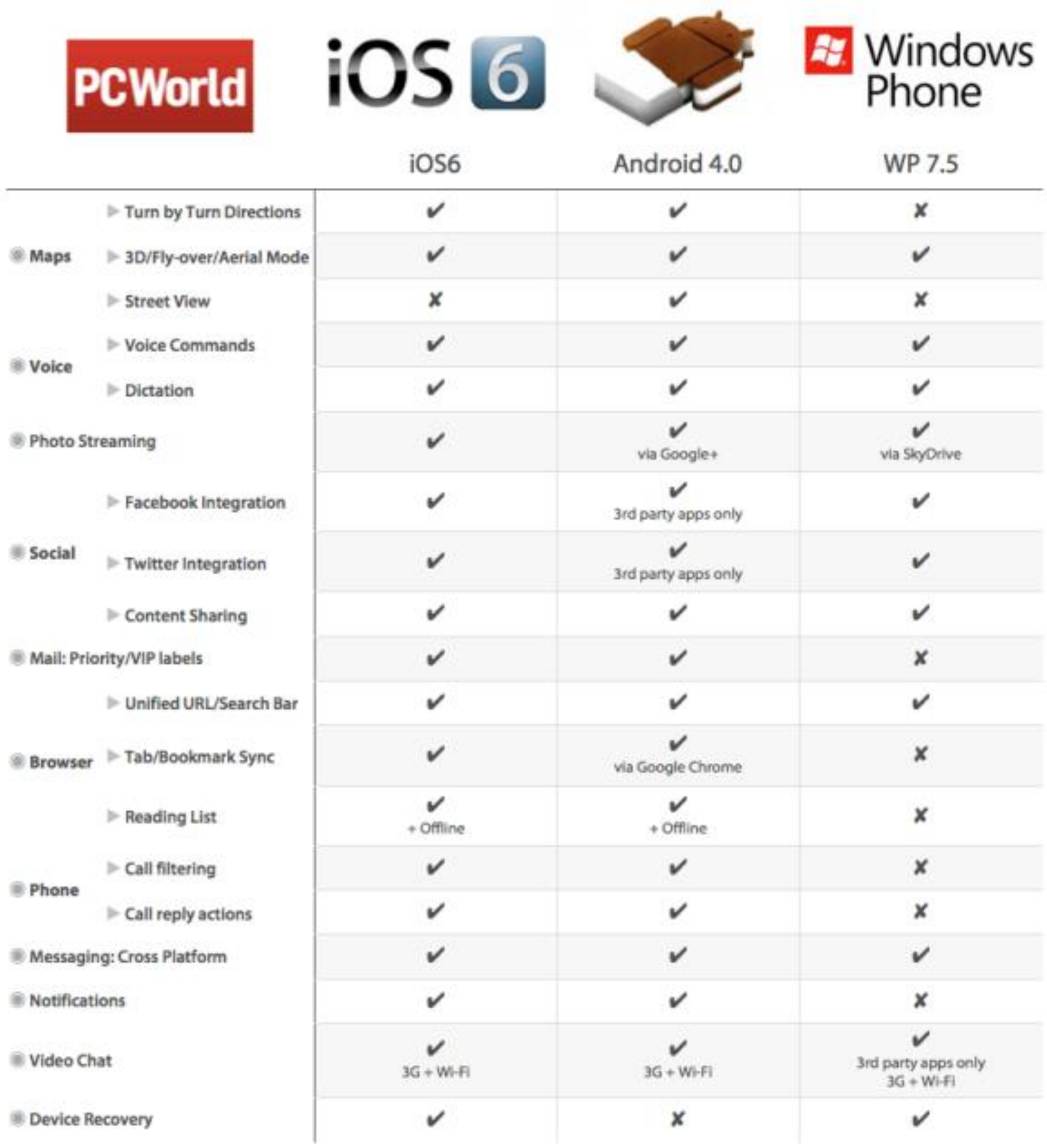
- ***Android Market***

This is an online software store to buy applications. Developers who created application scan add them into the store, and these applications can be downloaded by users, they can be both free and paid.

- ***Multitasking***

Android allows multitasking in the sense that multiple applications can run simultaneously.

With Task Manager it is possible view all running tasks and to switch from one to another easily.



PCWorld		iOS6	Android 4.0	WP 7.5
	Turn by Turn Directions	✔	✔	✘
Maps	3D/Fly-over/Aerial Mode	✔	✔	✔
	Street View	✘	✔	✘
Voice	Voice Commands	✔	✔	✔
	Dictation	✔	✔	✔
Photo Streaming		✔	✔ via Google+	✔ via SkyDrive
	Facebook Integration	✔	✔ 3rd party apps only	✔
Social	Twitter Integration	✔	✔ 3rd party apps only	✔
	Content Sharing	✔	✔	✔
Mail: Priority/VIP labels		✔	✔	✘
	Unified URL/Search Bar	✔	✔	✔
Browser	Tab/Bookmark Sync	✔	✔ via Google Chrome	✘
	Reading List	✔ + Offline	✔ + Offline	✘
Phone	Call filtering	✔	✔	✘
	Call reply actions	✔	✔	✘
Messaging: Cross Platform		✔	✔	✔
Notifications		✔	✔	✘
Video Chat		✔ 3G + Wi-Fi	✔ 3G + Wi-Fi	✔ 3rd party apps only 3G + Wi-Fi
Device Recovery		✔	✘	✔

Figure 1: Comparison of various operating systems available

1.3 ANDROID vs. IOS

ANDROID IS THE SUPERIOR SMARTPHONE DEVICE. IOS IS THE SUPERIOR PLATFORM

Android and iOS have different inherent strengths and weaknesses and instead of fruitlessly trying to decide which operating system is going to win everywhere, we should be focusing our efforts on determining which markets each OS is destined to dominate. Neither Android nor iOS is going away. Instead, each OS is going to go their separate ways.

Android's value is in the device. iOS's value is in the platform. Android will take the low end of the market. iOS will take the high end. Android will appeal to third-world nations, emerging markets, tech aficionados who admire the virtues of "open", those who require more freedom, those who require more options, those who require more diversity, those who use a single device, the cost conscious, and those who admire the value of free. iOS will appeal to more established nations, maturing markets, non-technical users who admire the virtues of easy and intuitive, those who require more security, those who require more consistency, those who require more integration, those who need multi-device management across multiple device form factors, the quality conscious, and those who fear Google's ad-supported business model.

iOS will appeal to Enterprise, businesses, governments, institutions, organizations, and other entities that require more structure and control. (As one who lived through the Windows v. Mac wars, the irony of this statement is not lost on me.)

1.4 ADVANTAGES

- Very handy and useful application.
- Helps you send SMS and E-mails using the same application.
- Helps you select contacts to send sms.
- You can view the location anytime anywhere.
- Important for those who cannot take time for their personal life due to their busy schedule.
- Helps you maintain cordial and fruitful relationships with your family members, friends and colleagues.

CHAPTER 2

TECHNOLOGIES USED

2.1 J2SE

Java Platform, Standard Edition or Java SE is a widely used platform for programming in the Java language. It is the Java Platform used to deploy portable applications for general use. In practical terms, Java SE consists of a virtual machine, which must be used to run Java programs, together with a set of libraries (or "packages") needed to allow the use of file systems, networks, graphical interfaces, and so on, from within those programs.

Java SE Overview

There are two principal products in the Java SE platform family: Java SE Runtime Environment (JRE) and Java Development Kit (JDK).

Java Runtime Environment (JRE)

The Java Runtime Environment (JRE) provides the libraries, the Java Virtual Machine, and other components to run applets and applications written in the Java programming language. In addition, two key deployment technologies are part of the JRE: Java Plug-in, which enables applets to run in popular browsers; and Java Web Start, which deploys standalone applications over a network. It is also the foundation for the technologies in the Java 2 Platform, Enterprise Edition (J2EE) for enterprise software development and deployment. The JRE does not contain tools and utilities such as compilers or debuggers for developing applets and applications.

Java Development Kit (JDK)

The JDK is a superset of the JRE, and contains everything that is in the JRE, plus tools such as the compilers and debuggers necessary for developing applets and applications. The conceptual diagram above illustrates all the component technologies in Java SE platform and how they fit together.

Java SE API

The Java SE application programming interface (API) defines the manner by which an applet or application can make requests to and use the functionality available in the compiled Java SE class libraries. (The Java SE class libraries are also part of the Java SE platform). The Java SE API consists of core technologies, Desktop (or client) technologies, and other technologies.

- Core components provide essential functionality for writing powerful enterprise-worthy programs in key areas such as database access, security, remote method invocation (RMI), and communications.
- Desktop components add a full range of features to help build applications that provide a rich user experience – deployment products such as Java Plug-in, component modeling APIs such as JavaBeans, and a graphical user interface.
- Other components round out the functionality.

Java Virtual Machine

The Java Virtual Machine is responsible for the hardware- and operating system-independence of the Java SE platform, the small size of compiled code (byte code), and platform security.

Java Platform Tools

The Java SE platform works with an array of tools, including Integrated Development Environments (IDEs), performance and testing tools, and performance monitoring tools.

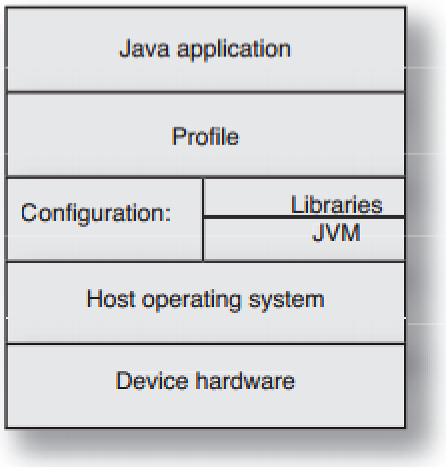


Figure 2: J2SE Layers

2.2 <u>ANDROID</u>

Android Incorporation was founded in Palo Alto, California, United States in October, 2003 by Andy Rubin : co-founder of Danger (Danger Incorporation was a company exclusively in platforms, software, design, and services for mobile computing devices), Rich Miner : co-founder of Wildfire Communications, Incorporation, Nick Sears : once VP at T-Mobile, and Chris White : headed design and interface development at Web TV. From starting Android Incorporation operated secretly, expose only that it was working on mobile software's. On that same year, Rubin had some sort of financial problems and Steve Perlman gave him $10,000 cash in an envelope and refused a stake in Android Incorporation.

Google took over Android Incorporation in August 2005, making Android Incorporation a entire owned property of Google Incorporation main employees of Android Incorporation, including Andy Rubin, Rich Miner and Chris White, stayed at the company after the possession of Google. Not much was known about Android Incorporation at the time of the acquisition, but people conclude that Google was planning to penetrate the mobile phone market with their weapon i.e. Android.

What is android?

Android is basically an operating system for smartphones. But we find now integrated into touch pads or televisions, even cars (trip computer) or netbooks. The OS was created by the start-up of the same name, which is owned by Google since 2005.

Specifications:

This operating system is based on version 2.6 of Linux, so it has a monolithic system kernel, what means that all system functions and drivers are grouped into one block of code.

Architecture

Android consists of five layers:

- The Linux kernel 2.6-which includes useful drivers that allow for example Wi-Fi or Bluetooth.
- The library written in C/C++ that provides higher level functionality such as an HTML engine, or a database (SQLite).
- A runtime environment for applications based on a virtual machine, made for inefficient machines such as telephones. The aim is to translate JAVA in machine language understood by Android.
- A JAVA framework that allows applications running on the virtual machine to organize and cooperate.

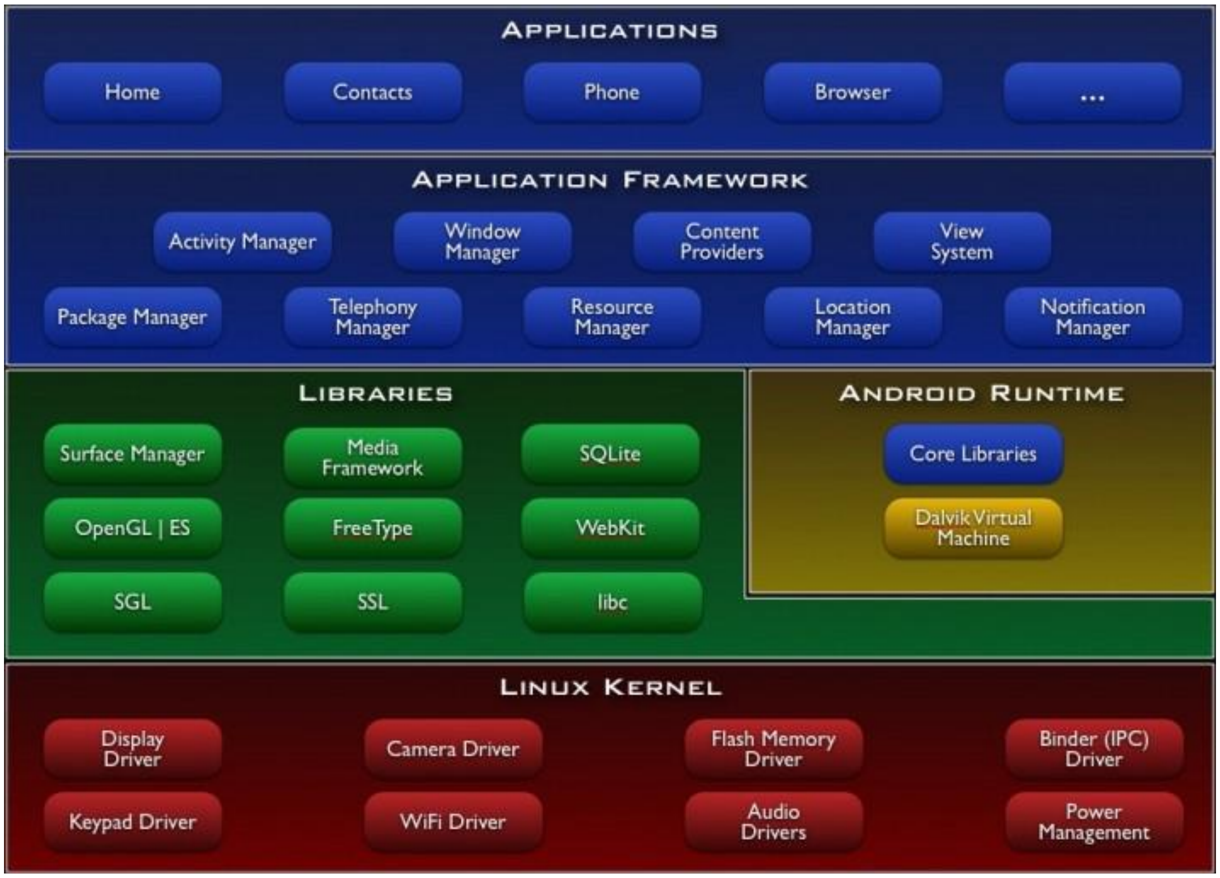


Figure 3: Android Architecture

Community-based firmware

There is a community of open-source enthusiasts that build and share Android-based firmware with a number of customizations and additional features, such as FLAC lossless audio support and the ability to store downloaded applications on the micro SD card. This usually involves rooting the device. Rooting allows users root access to the operating system, enabling full control of the phone. In order to use custom firmwares the device's boot loader must be unlocked. Rooting alone does not allow the flashing of custom firmware. Modified firmwares allow users of older phones to use applications available only on newer releases.

Those firmware packages are updated frequently, incorporate elements of Android functionality that haven't yet been officially released within a carrier-sanctioned firmware, and tend to have fewer limitations. CyanogenMod and OMFGB are examples of such firmware.

On 24 September 2009, Google issued a cease and desist letter to the modder Cyanogen, citing issues with the re-distribution of Google's closed-source applications within the custom firmware. Even though most of Android OS is open source, phones come packaged with closed-source Google applications for functionality such as the Android Market and GPS navigation. Google has asserted that these applications can only be provided through approved distribution channels by licensed distributors. Cyanogen has complied with Google's wishes and is continuing to distribute this mod without the proprietary software. He has provided a method to back up licensed Google applications during the mod's install process and restore them when it is complete.

2.2.1. Understanding the Android Software Stack

The Android software stack is composed of the elements shown in **Figure 4** and described in further detail below it. Put simply, a Linux kernel and a collection of C/C++ libraries are exposed through an application framework that provides services for, and management of, the run time and applications.

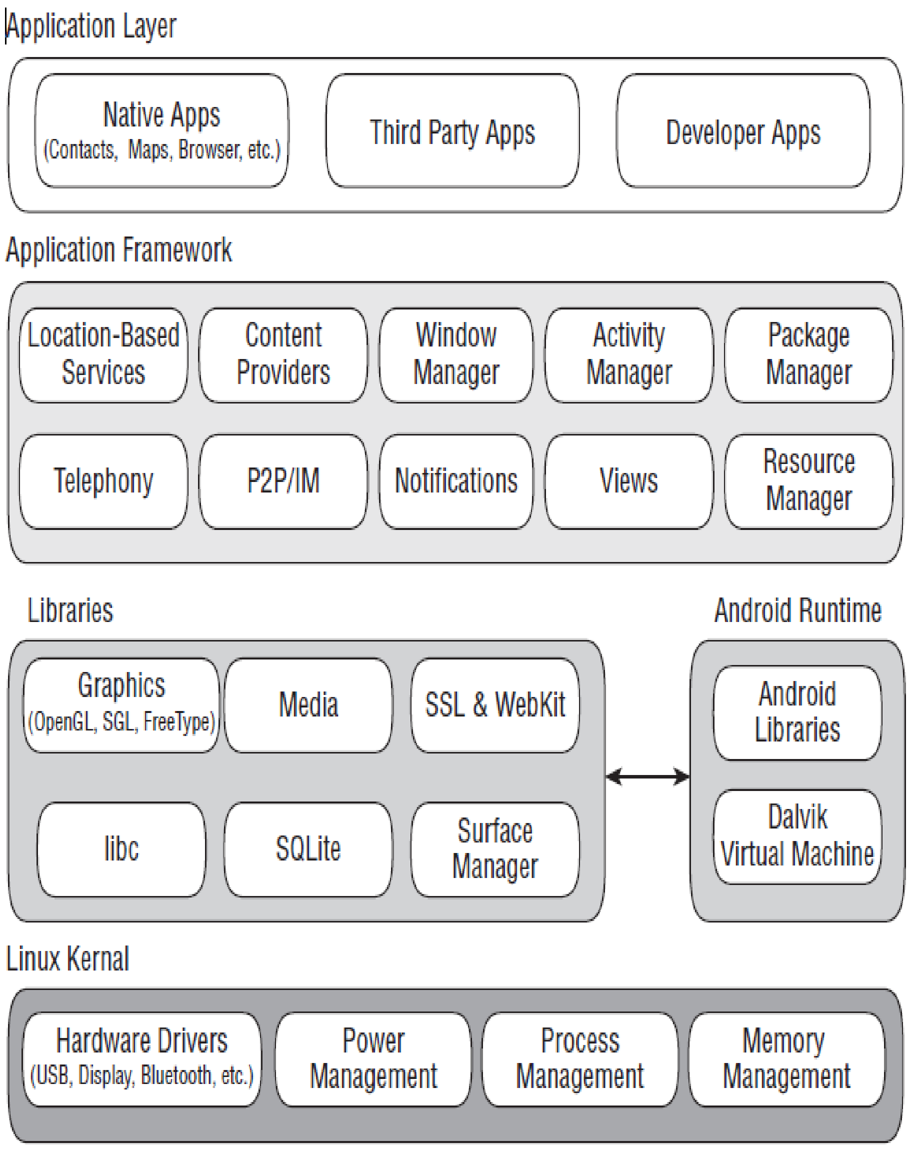


Figure: 4 Android Software Stack

2.2.2. Characteristic of the market:

Competitors

- The principal competitor is iPhone OS. It is mainly for competing with Apple that Android has been created.
- Palm OS devices on PDA.
- Blackberry: which team the same name smartphones
- Windows Mobile: which team smartphones and PDAs.
- Symbian: Current Market Leader

Key partners

To help launch Android, Google has created an alliance of thirty companies in order to develop standards for mobile devices. There is, among others:

- Operators such as NTT Dokomo, T-Mobile or Bouygues Telecom
- Of-equipment manufacturers like Sony Ericsson or Samsug
- Manufacturers of semiconductors, including Intel and Nvidia
- Corporate businesses.

Market share

The android market share continues to increase since its inception, and is likely to continue climbing because it is favored by big players like HTC, Sony Ericsson, Samsung, LG, Motorola, Dell and Acer. Moreover, according to IDC, android will be the 2nd mobile operating system used of the market in 2013. Here is the state of the market from 2006 to 2009. You have to know that the first mobile phone appeared in android date October 2008.

What Androids Are Made Of

When you write a desktop application, you are "master of your own domain." You launch your main window and any child windows—like dialog boxes—that are needed. From your standpoint, you are your own world, leveraging features supported by the operating system, but largely ignorant of any other program that may be running on the computer at the same time. If you do interact with other programs, it is typically through an application programming interface (API), such as Java Database Connectivity (JDBC), or frameworks atop it, to communicate with MySQL or another database. Android has similar concepts, but they are packaged differently and structured to make phones more crash-resistant:

Activities**:** The building block of the user interface is the activity. You can think of an activity as being the Android analogue for the window or dialog box in a desktop application or the page in a classic web application. Android is designed to support lots of cheap activities, so you can allow users to keep tapping to open new activities and tapping the Back button to back up, just like they do in a web browser.

Services**:** Activities are short-lived and can be shut down at any time. Services, on the other hand, are designed to keep running, if needed, independent of any activity, akin to the notion of services or daemons on other operating systems. You might use a service to check for updates to an RSS feed or to play back music even if the controlling activity is no longer operating. You will also use services for scheduled tasks ("cron jobs") and for exposing custom APIs to other applications on the device, though those are relatively advanced capabilities.

Content providers**:** Content providers provide a level of abstraction for any data stored on the device that is accessible by multiple applications. The Android development model encourages you to make your own data available to other applications, as well as your own applications. Building a content provider lets you do that, while maintaining complete control over how your data gets accessed. Content providers can be anything from web feeds, to local SQLite databases, and beyond.

***Intents*:** Intents are system messages that run around the inside of the device and notify applications of various events, from hardware state changes (e.g., an SD card was inserted), to incoming data (e.g., a Short Message Service [SMS] message arrived), to application events (e.g., your activity was launched from the device's main menu). Intents are much like messages or events on other operating systems. Not only can you respond to an Intent, but you can create your own to launch other activities or to let you know when specific situations arise (e.g., raise such-and-so Intent when the user gets within 100 meters of this-and-such location).

The Activity base class defines a series of events that governs the life cycle of an activity in Android. The Activity class defines the following events:

• ***onCreate()*** —	Called when the activity is first created
• ***onStart()*** —	Called when the activity becomes visible to the user
• ***onResume()*** —	Called when the activity starts interacting with the user
• ***onPause()*** —	Called when the current activity is being paused and the previous activity is being resumed
• ***onStop()*** —	Called when the activity is no longer visible to the user
• ***onDestroy()*** —	Called before the activity is destroyed by the system (either manually or by the system to conserve memory)
• ***onRestart()*** —	Called when the activity has been stopped and is restarting again

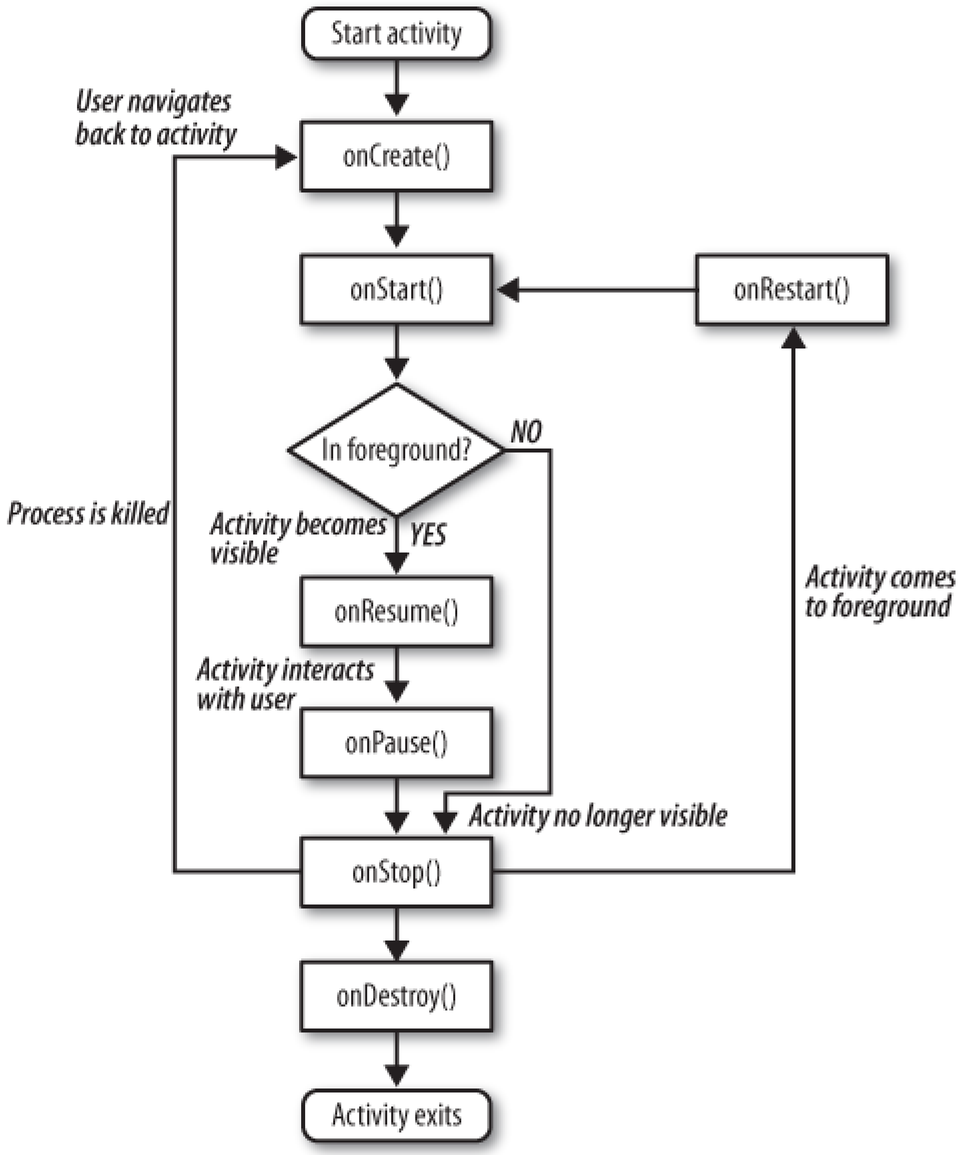


Figure: 5 Android Activity Lifecycle

2.3 XML

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. It is defined in the XML 1.0 Specification and several other related specifications. The design goals of XML emphasize simplicity, generality, and usability over the Internet. It is a textual data format with strong support via Unicode for the languages of the world.

Key Terminology

The material in this section is based on the XML Specification. This is not an exhaustive list of all the constructs that appear in XML; it provides an introduction to the key constructs most often encountered in day-to-day use.

(Unicode) character

By definition, an XML document is a string of characters. Almost every legal Unicode character may appear in an XML document.

Processor and application

The *processor* analyzes the markup and passes structured information to an *application*. The specification places requirements on what an XML processor must do and not do, but the application is outside its scope. The processor (as the specification calls it) is often referred to colloquially as an *XML parser*.

Markup and content

The characters making up an XML document are divided into *markup* and *content*, which may be distinguished by the application of simple syntactic rules. Generally, strings that constitute markup either begin with the character < and end with a >, or they begin with the character & and end with a ;. Strings of characters that are not markup are content. However, in a CDATA section, the delimiters <![CDATA[and]]> are classified as markup, while the text between them is classified as content. In addition, whitespace before and after the outermost element, is classified as markup.

Tag

A markup construct that begins with < and ends with >. Tags come in three flavors:

- *start-tags*; for example: <section>
- *end-tags*; for example: </section>
- *empty-element tags*; for example: <line-break />

Element

A logical document component either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag. The characters between the start- and end-tags, if any, are the element's *content*, and may contain markup, including other elements, which are called *child elements*. An example of an element is <Greeting>Hello, world.</Greeting> . Another is <line-break />.

Attribute

A markup construct consisting of a name/value pair that exists within a start-tag or empty-element tag. In the example (below) the element *img* has two attributes, *src* and *alt*:

```
<imgsrc="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
```

XML Declaration

XML documents may begin by declaring some information about them, as in the following example:

```
<?xml version="1.0"encoding="UTF-8"?>
```

XML example :

```
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Chapter 3
SOFTWARE MODEL

3.1.What is a software life cycle model?

A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed.

What are the benefits of software life cycle model?

- It provides guideline to organize, plan, staff, budget, schedule and manage software project work over organizational time, space, and computing environments.
- It provides prescriptive outline for what documents to produce for delivery to client.
- It provides basis for determining what software engineering tools and methodologies will be most appropriate to support different life cycle activities.
- It provides the framework for analysing or estimating patterns of resource allocation and consumption during the software life cycle
- It provides basis for conducting empirical studies to determine what affects software productivity, cost, and overall quality.

The waterfall model often considered as classic approach to the system development life cycle, the waterfall model describes a development method that is linear and sequential. Waterfall development has distinct goals for each phase of development. Imagine a waterfall on the cliff of a steep mountain. Once the water has flowed over the edge of the cliff and has begun its journey down the side of the mountain, it cannot turn back. It is the same with waterfall development. Once a phase of development is completed, the development proceeds to the next phase and there is no turning back. Waterfall Model is one of the most widely used Software Development Process. It is also called as "**Linear Sequential model**" or the "classic life cycle" or iterative model. It is widely used in the commercial development projects. It is called so because here, we move to next phase (step) after getting input from previous phase, like in a waterfall, water flows down to from the upper steps.

3.2 <u>Phases</u>

Analysis

- Feasibility study
- Requirements gathering
- Requirements specification preparation

Design

- Architecture
- Database design
- Prototypes and wireframe development
- Test plan preparation

Development

- Coding
- Unit testing

Testing & Quality Assurance

- Test cases preparation
- Testing and issue tracking
- Bug resolution

Launch and Maintenance

- Application delivery
- User acceptance

- Maintenance
- Support

Although, the phase name may differ for every software organization, the basic implementation steps remain the same.

Requirement Analysis and Software Definition

This is the first phase of waterfall model which includes a meeting with the customer to understand his requirements. This is the most crucial phase as any misinterpretation at this stage may give rise to validation issues later. The software definition must be detailed and accurate with no ambiguities. It is very important to understand the customer requirements and expectations so that the end product meets his specifications.

System Design

The customer requirements are broken down into logical modules for the ease of implementation. Hardware and software requirements for every module are identified and designed accordingly. Also the inter relation between the various logical modules is established at this stage. Algorithms and diagrams defining the scope and objective of each logical model are developed. In short, this phase lays a fundamental for actual programming and implementation.

System Implementation

This is the software process in which actual coding takes place. A software program is written based upon the algorithm designed in the system design phase. A piece of code is written for every module and checked for the output.

System Testing

The programmatically implemented software module is tested for the correct output. Bugs, errors are removed at this stage. In the process of software testing, a series of tests and test cases are performed to check the module for bugs, faults and other errors. Erroneous codes are rewritten and tested again until desired output is achieved.

System Deployment and Maintenance

This is the final phase of the waterfall model, in which the completed software product is handed over to the client after alpha, beta testing. After the software has been deployed on the client site, it is the duty of the software development team to undertake routine maintenance activities by visiting the client site.

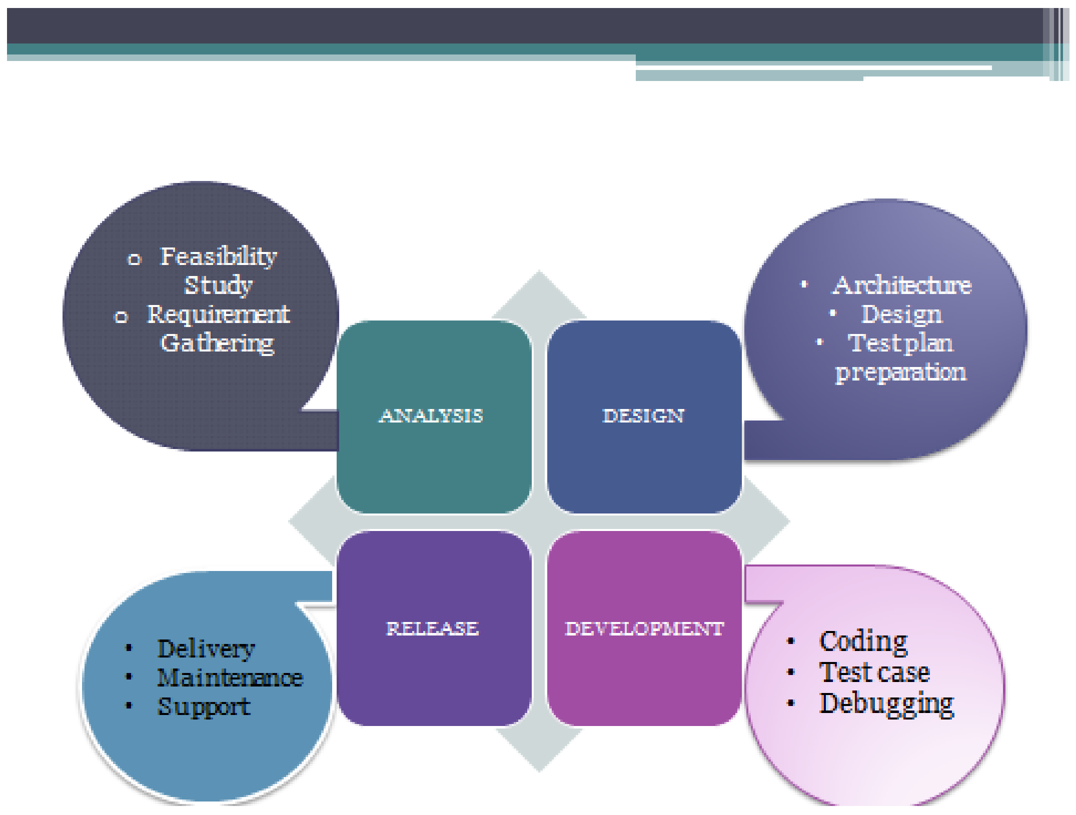
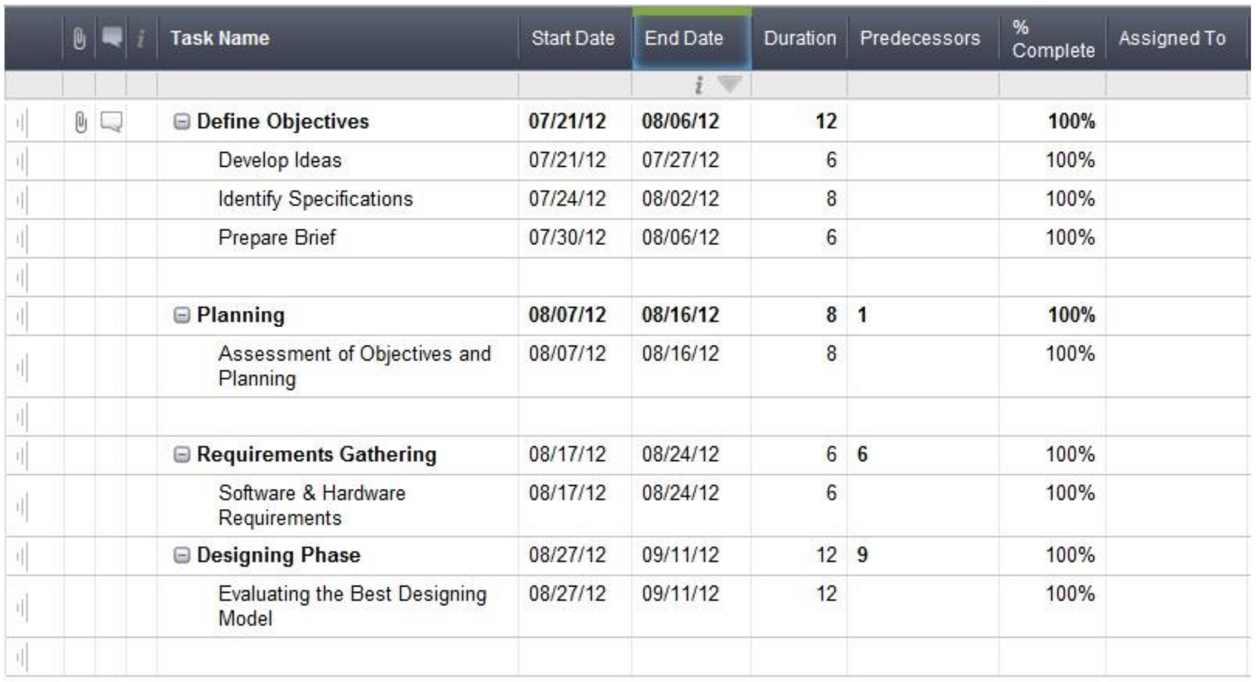


Figure: 6 Waterfall model

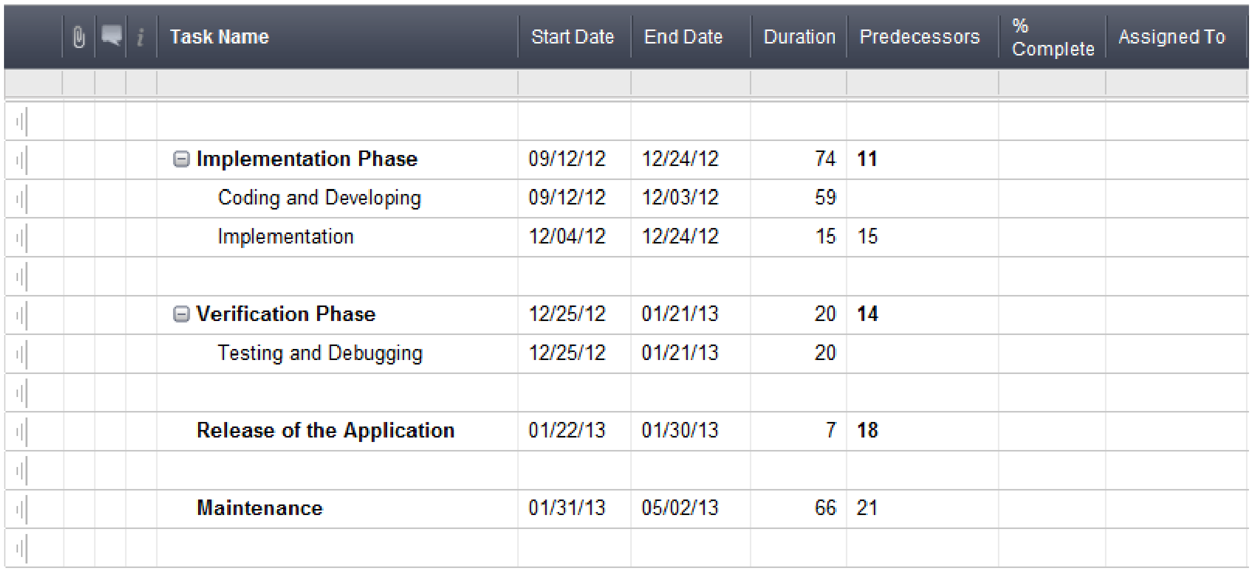
3.3. Feasibility Study

- Main aim: design a mobile application with the mentioned features using the mentioned requirements.
- Technical feasibility: similar applications are available in the market with some of the features. We would try and incorporate our innovative ideas in the project.
- Economic feasibility
- Schedule feasibility: the project will be done in the given time frame.

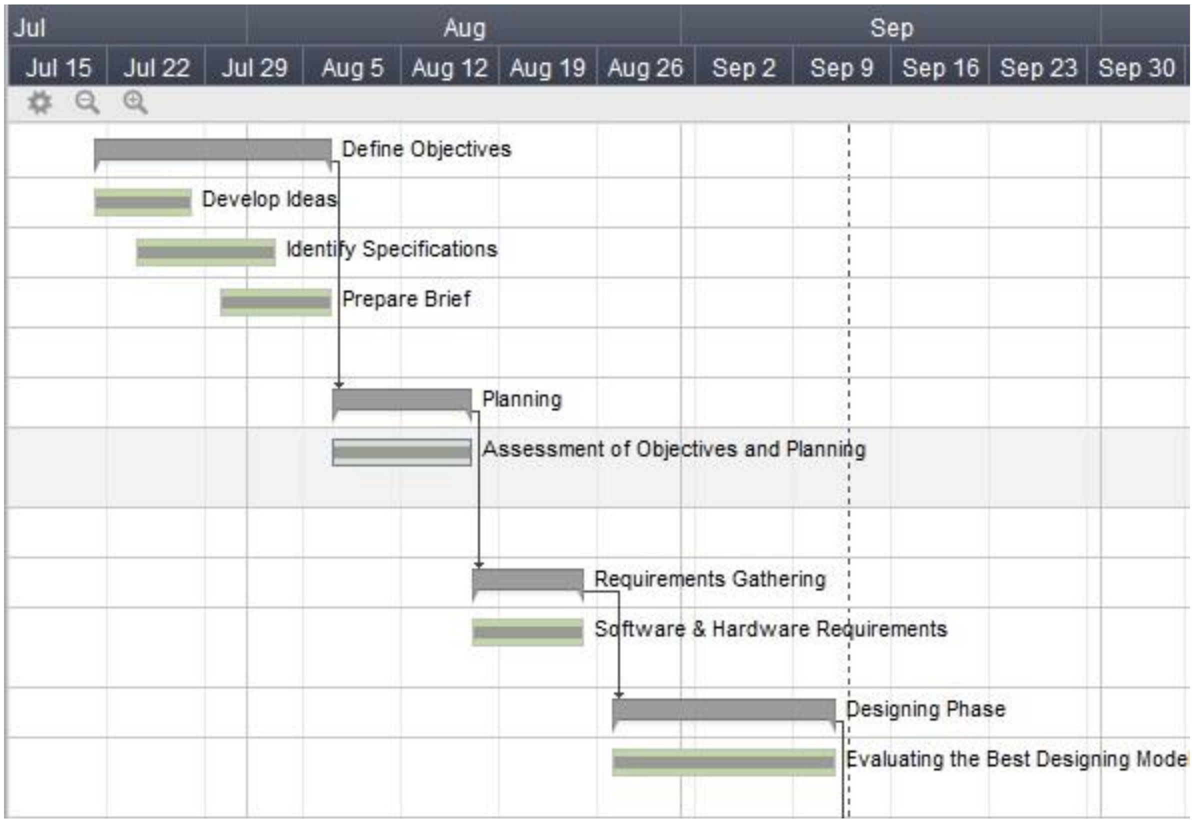
GANTT CHART

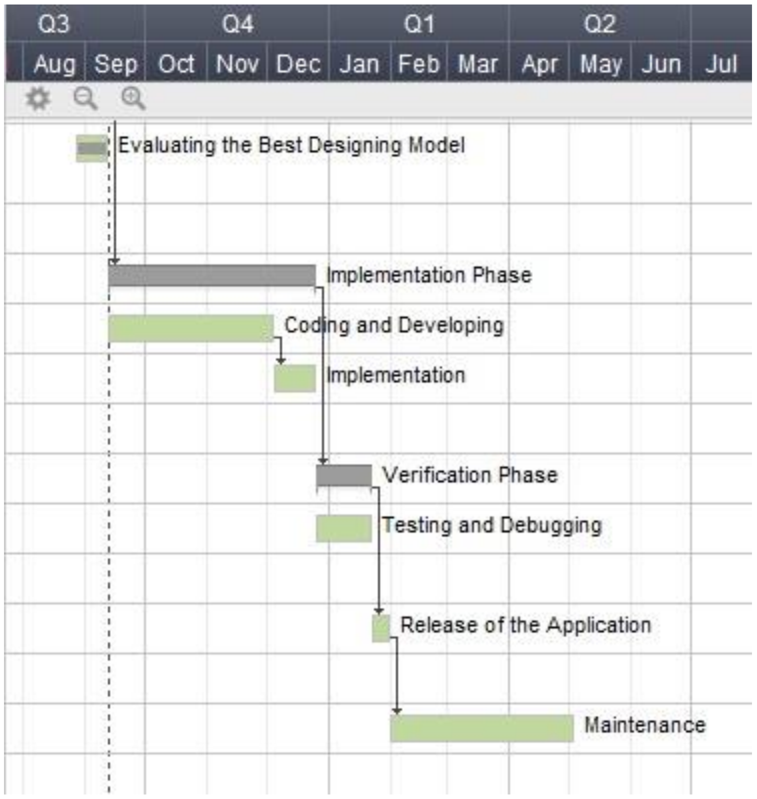


Task Name	Start Date	End Date	Duration	Predecessors	% Complete	Assigned To
Define Objectives	**07/21/12**	**08/06/12**	**12**		**100%**	
Develop Ideas	07/21/12	07/27/12	6		100%	
Identify Specifications	07/24/12	08/02/12	8		100%	
Prepare Brief	07/30/12	08/06/12	6		100%	
Planning	**08/07/12**	**08/16/12**	**8**	**1**	**100%**	
Assessment of Objectives and Planning	08/07/12	08/16/12	8		100%	
Requirements Gathering	08/17/12	08/24/12	6	**6**	100%	
Software & Hardware Requirements	08/17/12	08/24/12	6		100%	
Designing Phase	08/27/12	09/11/12	12	**9**	100%	
Evaluating the Best Designing Model	08/27/12	09/11/12	12		100%	



Task Name	Start Date	End Date	Duration	Predecessors	% Complete	Assigned To
Implementation Phase	09/12/12	12/24/12	74	**11**		
Coding and Developing	09/12/12	12/03/12	59			
Implementation	12/04/12	12/24/12	15	15		
Verification Phase	12/25/12	01/21/13	20	**14**		
Testing and Debugging	12/25/12	01/21/13	20			
Release of the Application	01/22/13	01/30/13	7	**18**		
Maintenance	01/31/13	05/02/13	66	21		





3.4. <u>**Software and Hardware Requirements**</u>

The My Messenger employs both standard database management system and user interface management system in its implementation.

- Eclipse
- Android Development Tool Plugin for Eclipse
- Android Software Development Kit Manager with Emulator (android version 2.1)
- HTML5 for layout
- Android Device with android version 2.1 and above and minimum 600MHz of processor.

3.5. Functional Requirements

1. The application has to provide user with the ability to interact with the application in order to select whether he/she wants to send SMS, e-mail, his/her location, schedule the SMS or view contacts.
2. The application has to provide user with the ability of sending SMS using various options.
3. The SMS sender must have an option to be saved in the sent folder for later reference.
4. The user must be able to interact with the application through tapping gestures.
5. The application should also allow the user to see his/her own location and navigate to the location that he/she gets.

3.6. Non-Functional Requirements

Performance - The system should be prompt in response to user actions, the application should have minimum lag time.

Usability – The UI of the application should fairly simple for any user to understand. The wordings and buttons should be of visible size to the user.

Dependency – The application must use the minimum amount of user phone data and stories deleted must be removed from phone instantaneously.

Backward Compatibility – The application should function without any faults from the latest version of Android till version 4.2

Recovery – The application should be able to recover from an error upon re-launch and the data should rollback to the state prior to the error.

Scalability – The application should be open and able to scale according to future upgrades.

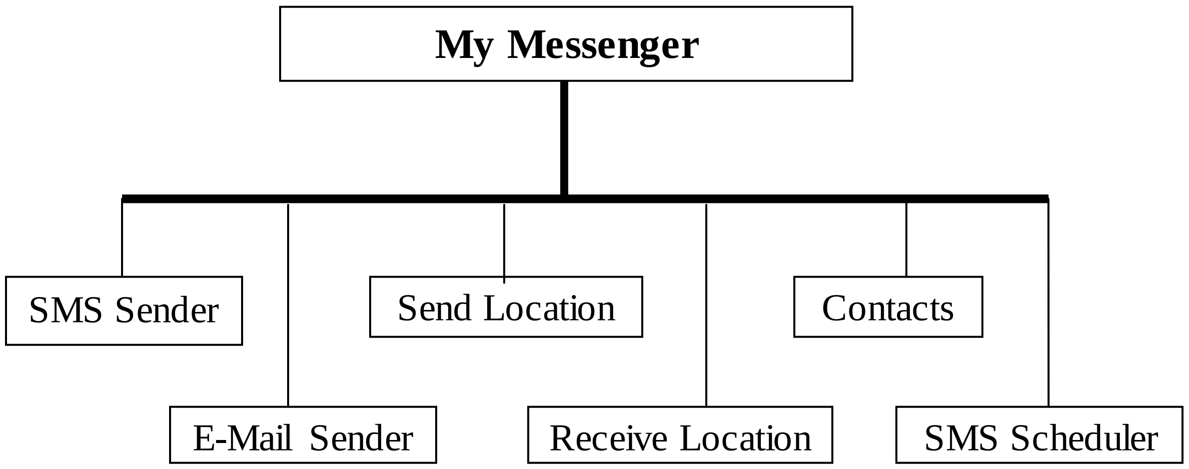


Figure: 7 Modules of Messenger

3.7 Use Cases

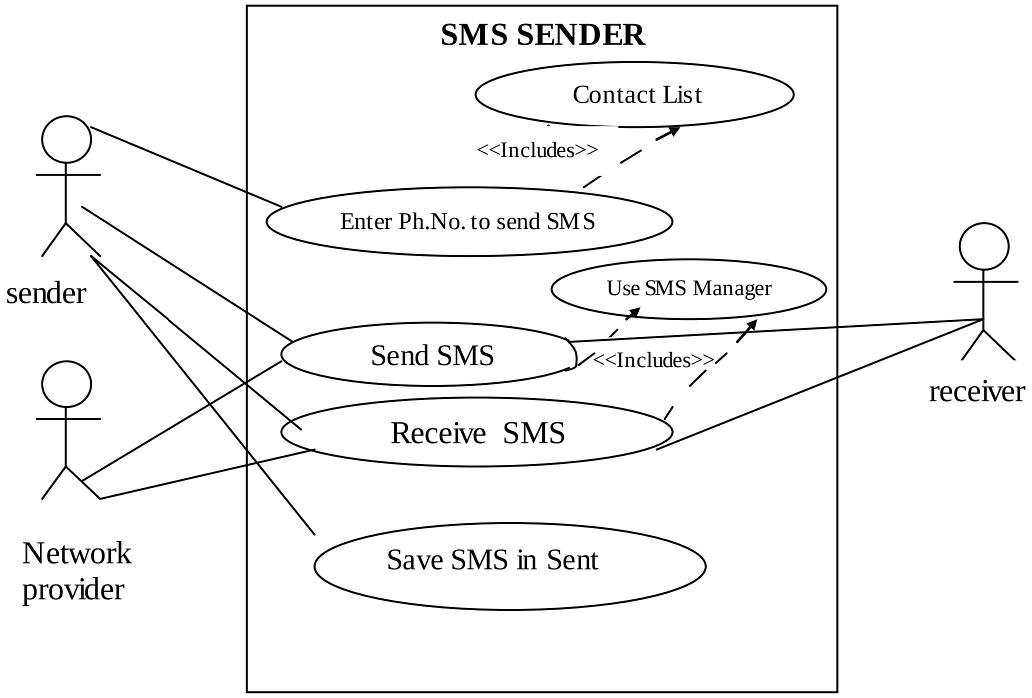


Figure: 8

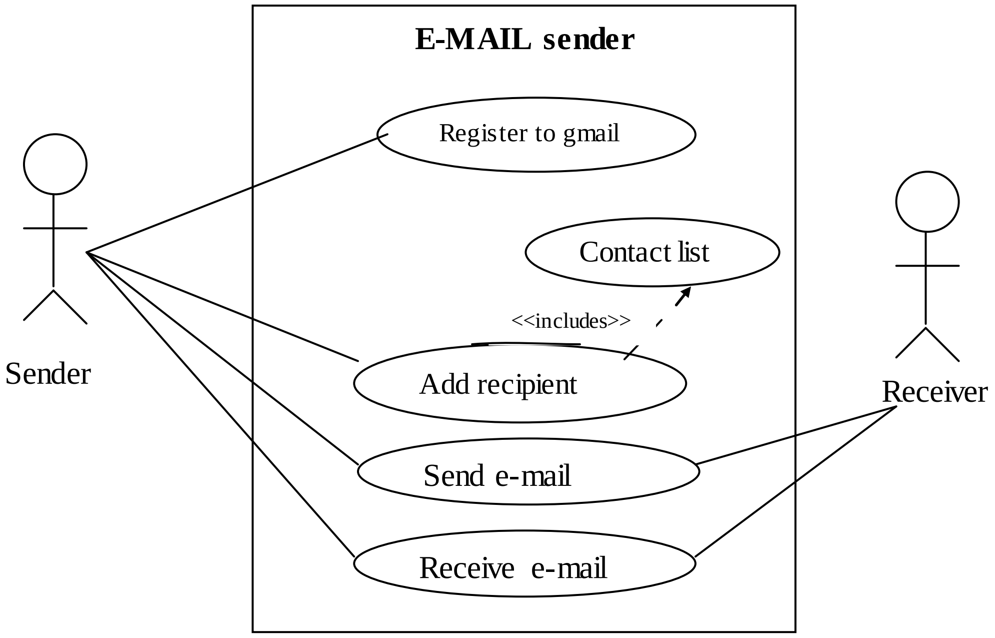


Figure: 9

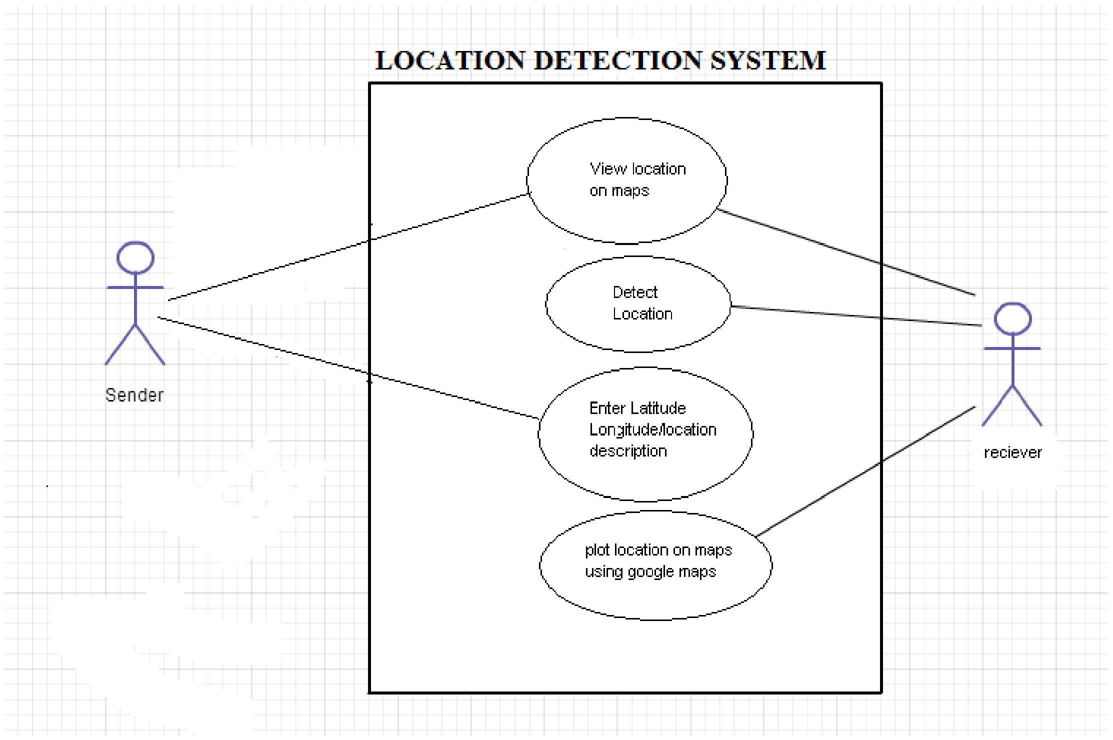

LOCATION DETECTION SYSTEM
View location on maps
Detect Location
Enter Latitude Longitude/location description
plot location on maps using google maps
Sender
reciever

Figure: 10

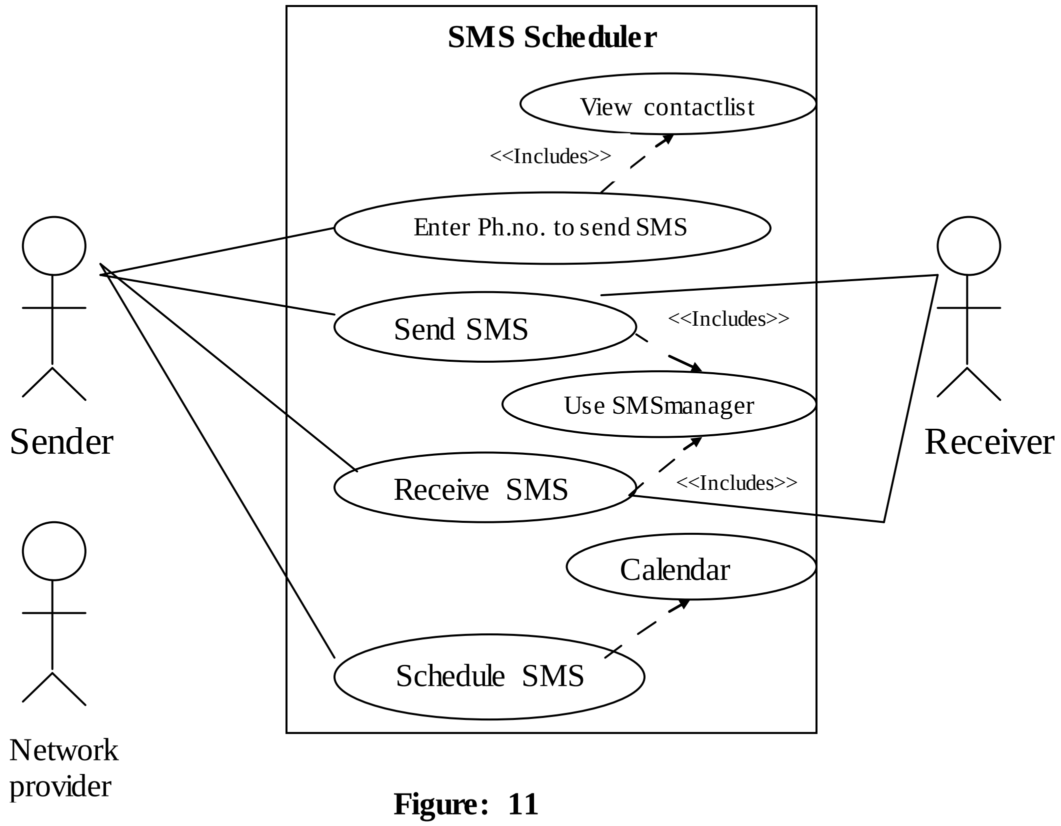


Figure: 11

Chapter 4
SMS SENDER

SMS messaging

SMS messaging is one of the main killer applications on a mobile phone today — for some users as necessary as the phone itself. Any mobile phone you buy today should have at least SMS messaging capabilities, and nearly all users of any age know how to send and receive such messages. Android comes with a built-in SMS application that enables you to send and receive SMS messages. However, in some cases you might want to integrate SMS capabilities into your own Android application. For example, you might want to write an application that automatically sends a SMS message at regular time intervals. For example, this would be useful if you wanted to track the location of your kids — simply give them an Android device that sends out an SMS message containing its geographical location every 30 minutes.

This section describes how you can programmatically send and receive SMS messages in your Android applications. The good news for Android developers is that you don't need a real device to test SMS messaging: The free Android Emulator provides that capability.

Sending SMS Messages Incognito

This describes how to send SMS messages programmatically from within the application. Using this approach, this application can automatically send an SMS message to a recipient without user intervention.

How It Works

Android uses a permissions-based policy whereby all the permissions needed by an application must be specified in the AndroidManifest.xml file. This ensures that when the application is installed, the user knows exactly which access permissions it requires. Because sending SMS messages incurs additional costs on the user's end, indicating the SMS permissions in the AndroidManifest.xml file enables users to decide whether to allow the application to install or not.

To send an SMS message programmatically, you use the SmsManager class. Unlike other classes, you do not directly instantiate this class; instead, you call the getDefault() static method to obtain a SmsManager object. You then send the SMS message using the sendTextMessage() method:

```
private void sendSMS(String phoneNumber, String message)
{
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(phoneNumber, null, message, null, null);
}
```

Following are the five arguments to the sendTextMessage() method:

- **destinationAddress** — Phone number of the recipient
- **scAddress** — Service center address; use null for default SMSC
- **text** — Content of the SMS message
- **sentIntent** — Pending intent to invoke when the message is sent (discussed in more detail in the next section)
- **deliveryIntent** — Pending intent to invoke when the message has been delivered (discussed in more detail in the next section)

Chapter 5

E-MAIL

Electronic mail, most commonly referred to as **email** or **e-mail** since approximately 1993, is a method of exchanging digital messages from an author to one or more recipients. Modern email operates across the Internet or other computer networks.

Email is the most defining feature of Smartphone's. Mobile email used to matter only to business people, but it's now a must-have for consumers as well.

Phone email clients provide access to both corporate and personal email, including Windows Live Mail, Gmail, and Yahoo. Corporate email is mostly handled through Lotus Notes or Microsoft's Exchange ActiveSync protocol that "pushes" email calendar, contact, and to do items on your phone within couple of seconds of their arrival to the company's server. The changes you make to these items on your phone are pushed back over the air to the corporate server.

Android phones integrate with Gmail services in real-time, so any changes you make to your Gmail, Google Calendar, and Google Contact items are synced with the Google cloud. As a result, you have up-to-date information on both mobile and desktop. Google provides a similar functionality for BlackBerry, iPhone, Windows Mobile, and Symbian devices through the free Google Sync service.

Chapter 6
LOCATION DETECTION

6.1.Maps, Geocoding, and Location-Based Services

One of the defining features of mobile phones is their portability, so it’s not surprising that some of the most enticing Android features are the services that let you find, contextualize, and map physical locations. You can create map-based Activities using Google Maps as a User Interface element. You have full access to the map, allowing you to control display settings, alter the zoom level, and move the centered location. Using Overlays, you can annotate maps and handle user input to provide map contextualized information and functionality.

Also covered in this chapter are the location-based services (LBS) — the services that let you find the device’s current location. They include technologies like GPS and Google’s cell-based location technology. You can specify which location-sensing technology to use explicitly by name, or implicitly by defining a set of criteria in terms of accuracy, cost, and other requirements.

Maps and location-based services use latitude and longitude to pinpoint geographic locations, but your users are more likely to think in terms of an address. Android provides a Geocoder that supports forward and reverse geocoding. Using the Geocoder, you can convert back and forth between latitude/longitude values and real-world addresses.

Used together, the mapping, geocoding, and location-based services provide a powerful toolkit for incorporating your phone’s native mobility into your mobile applications. The main tasks of the module are:

- Find and track the device location.
- Create and customize map-based Activities using MapView and MapActivity.
- Add Overlays to your maps.

6.1.1. Using Location-Based Services

Location-based services (LBS) is an umbrella term used to describe the different technologies used to find the device's current location. The two main LBS elements are:

- **LocationManager** Provides hooks to the location-based services.
- **LocationProviders** Each of which represents a different location-finding technology used to determine the device's current location.

Using the Location Manager, one can

- Obtain your current location.
- Track movement.
- Set proximity alerts for detecting movement into and out of a specified area.

6.1.2. Setting up the Emulator with Test Providers

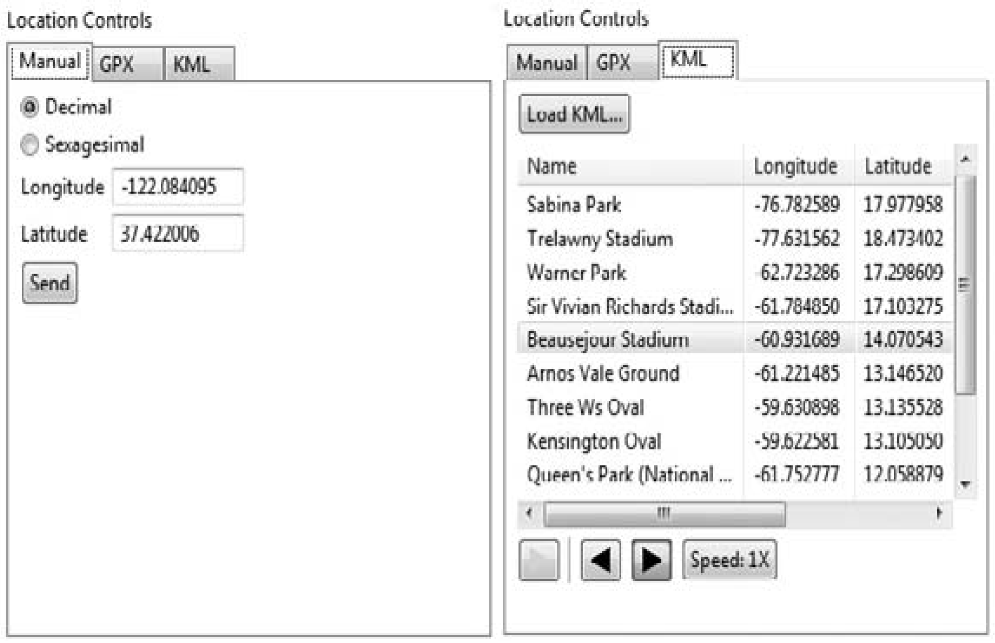
Location-based services are dependent on device hardware for finding the current location. When developing and testing with the emulator, your hardware is virtualized, and you're likely to stay in pretty much the same location. To compensate, Android includes hooks that let you emulate Location Providers for testing location based applications. In this section, you'll learn how to mock the position of the supported GPS provider.

6.1.2.1. Updating Locations in Emulator Location Providers

Use the Location Controls available from the DDMS perspective in Eclipse to push location changes directly into the test GPS_PROVIDER.

Figure 12 shows the Manual and KML tabs. Using the Manual tab, you can specify particular latitude/ longitude pairs. Alternatively, the KML and GPX tabs let you load KML (Keyhole Markup Language) and GPX (GPS Exchange Format) fi les, respectively. Once loaded, you can jump to particular waypoints (locations) or play back each location sequentially.

All location changes applied using the DDMS Location Controls will be applied to the GPS receiver, which must be enabled and active. Note that the GPS values returned by getLastKnownLocation will not change unless at least one application has requested location updates.



(● Decimal) Sexagesimal

Longitude -122.084095

Latitude 37.422006

Name	Longitude	Latitude
Sabina Park	-76.782589	17.977958
Trelawny Stadium	-77.631562	18.473402
Warner Park	-62.723286	17.298609
Sir Vivian Richards Stadi...	-61.784850	17.103275
Beausejour Stadium	-60.931689	14.070543
Arnos Vale Ground	-61.221485	13.146520
Three Ws Oval	-59.630898	13.135528
Kensington Oval	-59.622581	13.105050
Queen's Park (National ...	-61.752777	12.058879

Figure: 12 Updation of location in Emulator

6.1.3. Selecting a Location Provider

Depending on the device, there may be several technologies that Android can use to determine the current location. Each technology, or Location Provider, will offer different capabilities including power consumption, monetary cost, accuracy, and the ability to determine altitude, speed, or heading information. To get an instance of a specifi c provider, call getProvider, passing in the name:

```
String providerName = LocationManager.GPS_PROVIDER;
LocationProvider gpsProvider;
gpsProvider = locationManager.getProvider(providerName);
```

This is generally only useful for determining the abilities of a particular provider. Most Location Manager methods require only a provider name to perform location-based services.

6.1.3.1. Finding the Available Providers

The LocationManager class includes static string constants that return the provider name for the two most common Location Providers:

- LocationManager.GPS_PROVIDER
- LocationManager.NETWORK_PROVIDER

To get a list of names for all the providers available on the device, call getProviders, using a Boolean to indicate if you want all, or only the enabled, providers to be returned:

```
boolean enabledOnly = true;
List<String> providers = locationManager.getProviders(enabledOnly);
```

6.1.3.2. Finding Providers Based on Requirement Criteria

In most scenarios, it's unlikely that you will want to explicitly choose the Location Provider to use. More commonly, you'll specify the requirements that a provider must meet and let Android determine the best technology to use.

Use the Criteria class to dictate the requirements of a provider in terms of accuracy (fi ne or coarse), power use (low, medium, high), cost, and the ability to return values for altitude, speed, and bearing. The following code creates Criteria that require coarse accuracy, low power consumption, and no need for altitude, bearing, or speed. The provider is permitted to have an associated cost.

```
Criteria criteria = new Criteria();
criteria.setAccuracy(Criteria.ACCURACY_COARSE);
criteria.setPowerRequirement(Criteria.POWER_LOW);
criteria.setAltitudeRequired(false);
criteria.setBearingRequired(false);
criteria.setSpeedRequired(false);
criteria.setCostAllowed(true);
```

Having defi ned the required Criteria, you can use getBestProvider to return the best matching Location Provider or getProviders to return all the possible matches. The following snippet demonstrates using getBestProvider to return the best provider for your criteria where the Boolean lets you restrict the result to a currently enabled provider:

```
String bestProvider = locationManager.getBestProvider(criteria, true);
```

If more than one Location Provider matches your criteria, the one with the greatest accuracy is returned. If no Location Providers meet your requirements, the criteria are loosened, in the following order, until a provider is found:

- Power use
- Accuracy
- Ability to return bearing, speed, and altitude

The criterion for allowing a device with monetary cost is never implicitly relaxed. If no provider is found, null is returned. To see a list of names for all the providers that match your criteria, you can use getProviders. It accepts Criteria and returns a filtered String list of all available Location Providers that match them. As with the getBestProvider call, if no matching providers are found, this call returns null.

```
List<String> matchingProviders = locationManager.getProviders(criteria, false);
```

6.1.3.3. Finding Your Location

The purpose of location-based services is to fi nd the physical location of the device. Access to the location-based services is handled using the Location Manager system Service. To access the Location Manager, request an instance of the LOCATION_SERVICE using the getSystemService method, as shown in the following snippet:

```
String serviceString = Context.LOCATION_SERVICE;
LocationManager locationManager;
locationManager = (LocationManager)getSystemService(serviceString);
```

Before you can use the Location Manager, you need to add one or more uses-permission tags to your manifest to support access to the LBS hardware. The following snippet shows the *fine* and *coarse* permissions. Of the default providers, the GPS provider requires fine permission, while the Network provider requires only coarse. An application that has been granted fi ne permission will have coarse permission granted implicitly.

```
<uses-permission android:name=”android.permission.ACCESS_FINE_LOCATION”/>
```

```
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
```

You can find the last location fi x determined by a particular Location Provider using the getLastKnownLocation method, passing in the name of the Location Provider. The following example fi nds the last location fi x taken by the GPS provider:

```
String provider = LocationManager.GPS_PROVIDER;
Location location = locationManager.getLastKnownLocation(provider);
```

The Location object returned includes all the position information available from the provider that supplied it. This can include latitude, longitude, bearing, altitude, speed, and the time the location fi x was taken. All these properties are available using get methods on the Location object. In some instances, additional details will be included in the extras Bundle.

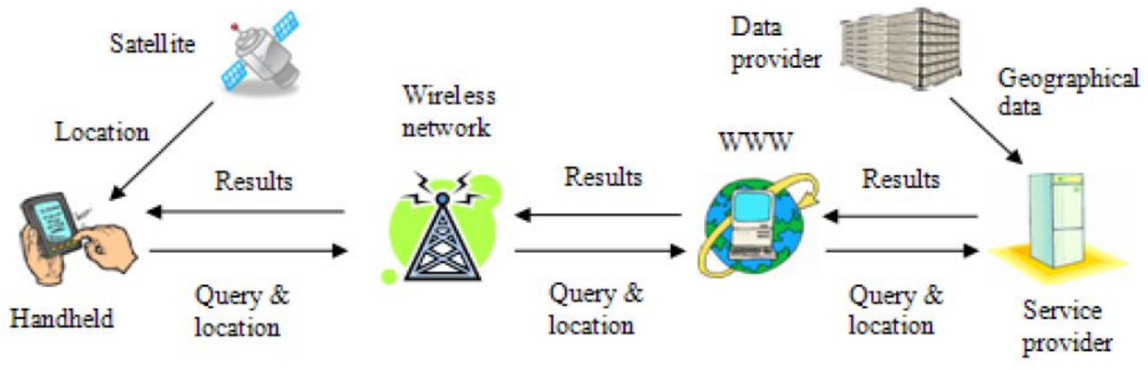


Figure: 13 System structure of generic location based services

6.1.3. Creating Map -Based Activities

The MapView provides a compelling User Interface option for presentation of geographical data. One of the most intuitive ways of providing context for a physical location or address is to display it on a map. Using a MapView, you can create Activities that feature an interactive map. Map Views support annotation using both Overlays and by pinning Views to geographical locations. Map Views offer full programmatic control of the map display, letting you control the zoom, location, and display modes — including the option to display satellite, street, and traffi c views. In the following sections, you'll see how to use Overlays and the MapController to create dynamic map-based Activities. Unlike online mash-ups, your map Activities will run natively on the device, allowing you to leverage its hardware and mobility to provide a more customized and personal user experience.

6.1.3.1. Introducing MapView and MapActivity

This section introduces several classes used to support Android maps:

- MapView is the actual Map View (control).
- MapActivity is the base class you extend to create a new Activity that can include a Map View. The MapActivity class handles the application life cycle and background service management required for displaying maps. As a result, you can only use a MapView within MapActivity-derived Activities.
- Overlay is the class used to annotate your maps. Using Overlays, you can use a Canvas to draw onto any number of layers that are displayed on top of a Map View.
- MapController is used to control the map, allowing you to set the center location and zoom levels.
- MyLocationOverlay is a special overlay that can be used to display the current position and orientation of the device.
- ItemizedOverlays and OverlayItems are used together to let you create a layer of map markers, displayed using drawable with associated text.

6.1.3.2 Creating a Map-Based Activity

To use maps in your applications, you need to create a new Activity that extends MapActivity. Within it, add a MapView to the layout to display a Google Maps interface element. The Android map library is not a standard package; as an optional API, it must be explicitly included in the application manifest before it can be used. Add the library to your manifest using a uses-library tag within the application node, as shown in the XML snippet below:

```
<uses-library android:name=”com.google.android.maps”/>
```

Google Maps downloads the map tiles on demand; as a result, it implicitly requires permission to use the Internet. To see map tiles in your Map View, you need to add a uses-permission tag to your application manifest for android.permission.INTERNET, as shown below:

```
<uses-permission android:name=”android.permission.INTERNET”/>
```

Once you’ve added the library and confi gured your permission, you’re ready to create your new map based Activity. MapView controls can only be used within an Activity that extends MapActivity. Override the onCreate method to lay out the screen that includes a MapView, and override isRouteDisplayed to return true if the Activity will be displaying routing information (such as traffi c directions).

At the time of publication, it was unclear how developers would apply for map keys. Invalid or disabled API keys will result in your MapView not loading the map image tiles. Until this process is revealed, you can use any text as your API key value.

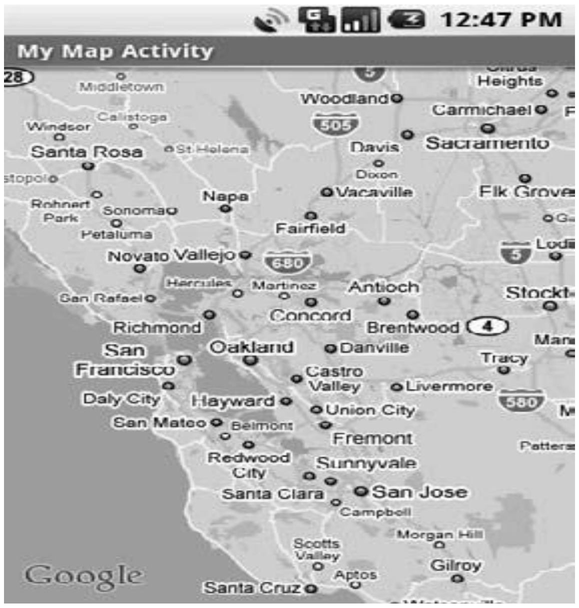


Figure: 14 Basic map activity

Android currently recommends that you include no more than one MapActivity *and one* MapView *in each application.*

6.1.3.3 Configuring and Using Map Views

The MapView class is a View that displays the actual map; it includes several options for deciding how the map is displayed.

By default, the Map View will show the standard street map. In addition, you can choose to display a satellite view, StreetView, and expected traffic, as shown in the code snippet below:

```
mapView.setSatellite(true);
mapView.setStreetView(true);
mapView.setTraffic(true);
```

You can also query the Map View to fi nd the current and maximum available zoom level, as well as the center point and currently visible longitude and latitude span (in decimal degrees). The latter (shown below) is particularly useful for performing geographically limited Geocoder lookups:

```
GeoPoint center = mapView.getMapCenter();
int latSpan = mapView.getLatitudeSpan();
int longSpan = mapView.getLongitudeSpan();
```

You can also optionally display the standard map zoom controls. The following code snippet shows how to get a reference to the Zoom Control View and pin it to a screen location. The Boolean parameter lets you assign focus to the controls once they're added.

```
int y = 10;
int x = 10;
MapView.LayoutParams lp;
lp = new MapView.LayoutParams(MapView.LayoutParams.WRAP_CONTENT,
MapView.LayoutParams.WRAP_CONTENT,
x, y,
MapView.LayoutParams.TOP_LEFT);
View zoomControls = mapView.getZoomControls();
mapView.addView(zoomControls, lp);
mapView.displayZoomControls(true);
```

6.1.3.4 Using the Map Controller

You use the Map Controller to pan and zoom a MapView. You can get a reference to a MapView's controller using getController, as shown in the following code snippet:

```
MapController mapController = myMapView.getController();
```

Map locations in the Android mapping classes are represented by GeoPoint objects, which contain latitude and longitude measured in microdegrees (i.e., degrees multiplied by 1E6 [or 1,000,000]). Before you can use the latitude and longitude values stored in the Location objects used by the locationbased services, you'll need to convert them to microdegrees and store them as GeoPoints, as shown in the following code snippet:

```
Double lat = 37.422006*1E6;
Double lng = -122.084095*1E6;
GeoPoint point = new GeoPoint(lat.intValue(), lng.intValue());
```

Re-center and zoom the MapView using the setCenter and setZoom methods available on the MapView's MapController, as shown in the snippet below:

```
mapController.setCenter(point);
mapController.setZoom(1);
```

When using setZoom, 1 represents the widest (or furthest away) zoom and 21 the tightest (nearest) view.

The actual zoom level available for a specifi c location depends on the resolution of Google's maps and imagery for that area. You can also use zoomIn and zoomOut to change the zoom level by one step. The setCenter method will "jump" to a new location; to show a smooth transition, use animateTo as shown in the code below:

```
mapController.animateTo(point);
```

6.1.3.5 Pinning Views to the Map and Map Positions

Previously in this chapter, you saw how to add the Zoom View to a Map View by pinning it to a specific screen location. You can pin any View-derived object to a Map View (including layouts and other View Groups), attaching it to either a screen position or a geographical map location.

In the latter case, the View will move to follow its pinned position on the map, effectively acting as an interactive map marker. As a more resource-intensive solution, this is usually reserved for supplying the detail "balloons" often displayed on mashups to provide further detail when a marker is clicked. Both pinning mechanisms are implemented by calling addView on the MapView, usually from the onCreate or onRestore methods within the MapActivity containing it. Pass in the View you want to pin and the layout parameters to use.

The MapView.LayoutParams parameters you pass in to addView determine how, and where, the View is added to the map.
To add a new View to the map relative to the screen, specify a new MapView.LayoutParams including arguments that set the height and width of the View, the x/y screen coordinates to pin to, and the alignment to use for positioning, as shown below:

```
int y = 10;
int x = 10;
MapView.LayoutParams screenLP;
screenLP = new MapView.LayoutParams(MapView.LayoutParams.WRAP_CONTENT,
MapView.LayoutParams.WRAP_CONTENT,
x, y,
MapView.LayoutParams.TOP_LEFT);
EditText editText1 = new EditText(getApplicationContext());
editText1.setText("Screen Pinned");
mapView.addView(editText1, screenLP);
```

To pin a View relative to a physical map location, pass four parameters when constructing the new MapView LayoutParams, representing the height, width, GeoPoint to pin to, and the layout alignment.

```
Double lat = 37.422134*1E6;
Double lng = -122.084069*1E6;
GeoPoint geoPoint = new GeoPoint(lat.intValue(), lng.intValue());
MapView.LayoutParams geoLP;
geoLP = new MapView.LayoutParams(MapView.LayoutParams.WRAP_CONTENT,
MapView.LayoutParams.WRAP_CONTENT,
geoPoint,
MapView.LayoutParams.TOP_LEFT);
EditText editText2 = new EditText(getApplicationContext());
editText2.setText("Location Pinned");
mapView.addView(editText2, geoLP);
```

Panning the map will leave the fi rst TextView stationary in the upper left corner, while the second TextView will move to remain pinned to a particular position on the map. To remove a View from a MapView, call removeView, passing in the View instance you wish to remove, as shown below:

```
mapView.removeView(editText2);
```

Chapter 7

CONTACTS

This section lets users pick a contact and then view the contact, via separate buttons. The View button is enabled only after the user picks a contact via the Pick button. Android makes the full database of contact information available to any application that has been granted the READ_CONTACTS permission. The Contacts Contract Provider provides an extensible database of contact-related information. This allows users to specify multiple sources for their contact information. More importantly, it allows developers to arbitrarily extend the data stored against each contact, or even become an alternative provider for contacts and contact details.

Let's take a closer look at how this feat is accomplished. When the user clicks the Pick button, we call startActivityForResult(). This is a variation on startActivity(), designed for activities that are set up to return some sort of result—a user's choice of file, contact, or whatever. Relatively few activities are set up this way, so you cannot expect to call startActivityForResult() and get answers from any activity you choose. In this case, we want to pick a contact. There is an ACTION_PICK Intent action available in Android that is designed for this sort of scenario. An ACTION_PICK Intent indicates to Android that we want to pick...something. That "something" is determined by the Uri we put in the Intent. In our case, it turns out that we can use an ACTION_PICK Intent for certain systemdefined Uri values to let the user pick a contact from the device's list of contacts. In particular, on Android 2.0 and higher, we can use:
android.provider.ContactsContract.Contacts.CONTENT_URI for this purpose:

```
public void pickContact(View v) {
Intent i=new Intent(Intent.ACTION_PICK,
Contacts.CONTENT_URI);
startActivityForResult(i, PICK_REQUEST);
}
```

When the user taps a contact, the picker activity ends (e.g., via finish()), and control returns to our activity. At that point, our activity is called with onActivityResult(). Android supplies us with three pieces of information:

- The identifying number we supplied to startActivityForResult(), so we can match this result to its original request
- A result status, either RESULT_OK or RESULT_CANCELED, to indicate whether the user made a positive selection or abandoned the picker (e.g., by pressing the Back button)
- An Intent that represents the result data itself, for a RESULT_OK response

The details of what is in the Intent will need to be documented by the activity that you called. In the case of an ACTION_PICK Intent for the Contacts. CONTENT_URI, the returned Intent has its own Uri (via getData()) that represents the chosen contact. In the RotationOne example, we stick that in a data member of the activity and enable the View button:

```
@Override
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
if (requestCode==PICK_REQUEST) {
if (resultCode==RESULT_OK) {
contact=data.getData();
viewButton.setEnabled(true);
}
}
}
```

If the user clicks the now-enabled View button, we create an ACTION_VIEW Intent on the
contact's Uri, and call startActivity() on that Intent:

```
public void viewContact(View v) {
startActivity(new Intent(Intent.ACTION_VIEW, contact));
}
```

This will bring up an Android-supplied activity to view details of that contact.

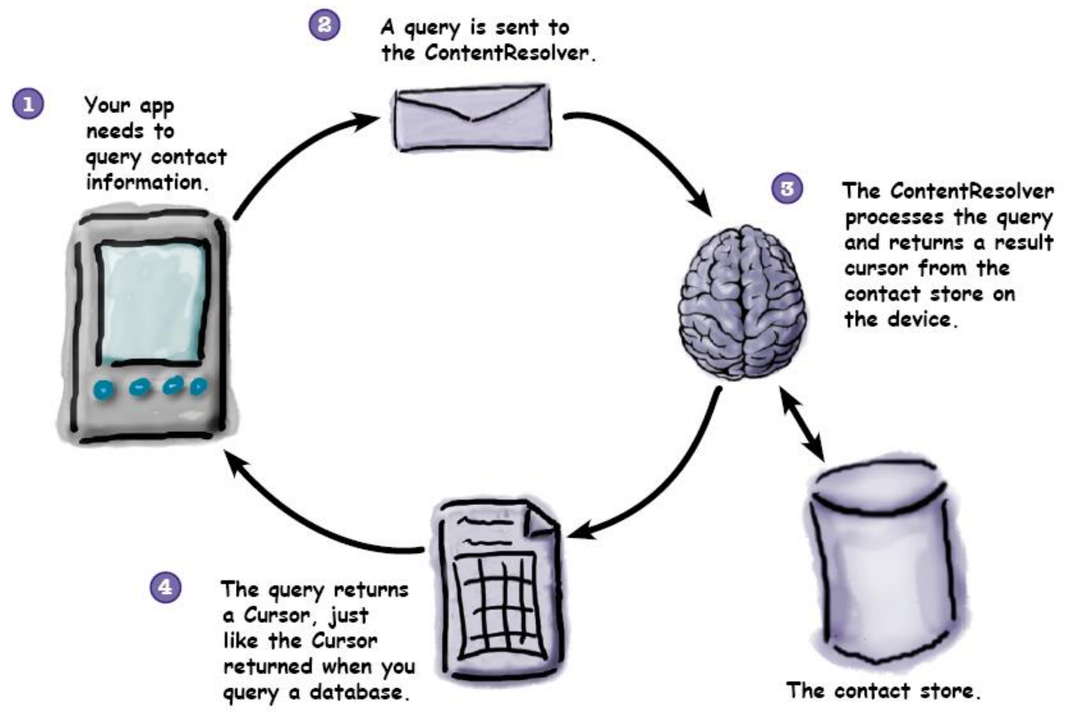

1
Your app needs to query contact information.
2
A query is sent to the ContentResolver.
3
The ContentResolver processes the query and returns a result cursor from the contact store on the device.
4
The query returns a Cursor, just like the Cursor returned when you query a database.
The contact store.

Figure: 15 Contact Picking Activity

Chapter 8
SCHEDULING

The key concept present in any operating system which allows the system to support multitasking, multiprocessing, etc. is Task Scheduling. Task Scheduling is the core, which refers to the way the different processes are allowed to share the common CPU. Scheduler and dispatcher are the software which help to carry out this assignment,.

Android operating system uses O (1) scheduling algorithm as it is based on Linux Kernel 2.6. Therefore the scheduler is names as Completely Fair Scheduler as the processes can schedule within a constant amount of time, regardless of how many processes are running on the operating system.

Pre-emptive task scheduling involves interrupting the low priority tasks when high priority tasks are present in the queue. This scheduling is particularly used for mobile operating system as the CPU utilization is medium, turnaround time and response time is high. Mobile phones are required to meet specific time deadlines for the tasks to occur.

Algorithm

1. Start the SMS application

2. check if there are any messages for the day.

3. SMS in the queue are sent by the SMSManager class.

4. if queue is empty, continue with normal work.

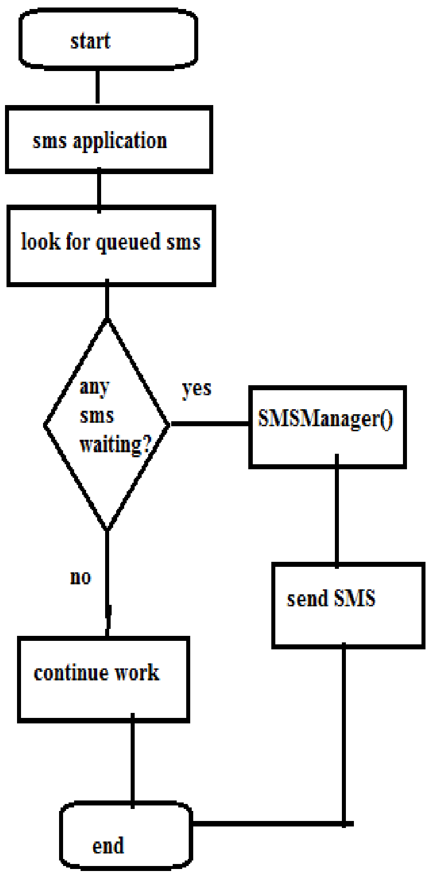
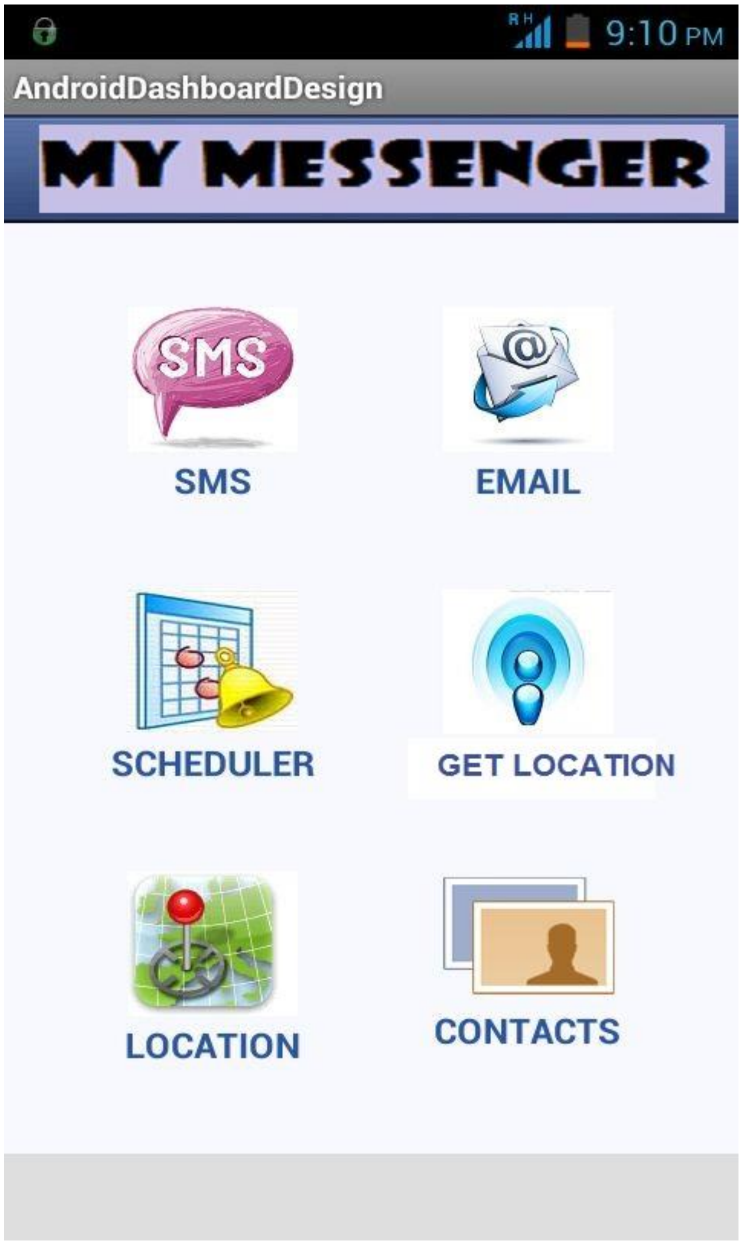


Figure: 16 Flowchart of SMS scheduler

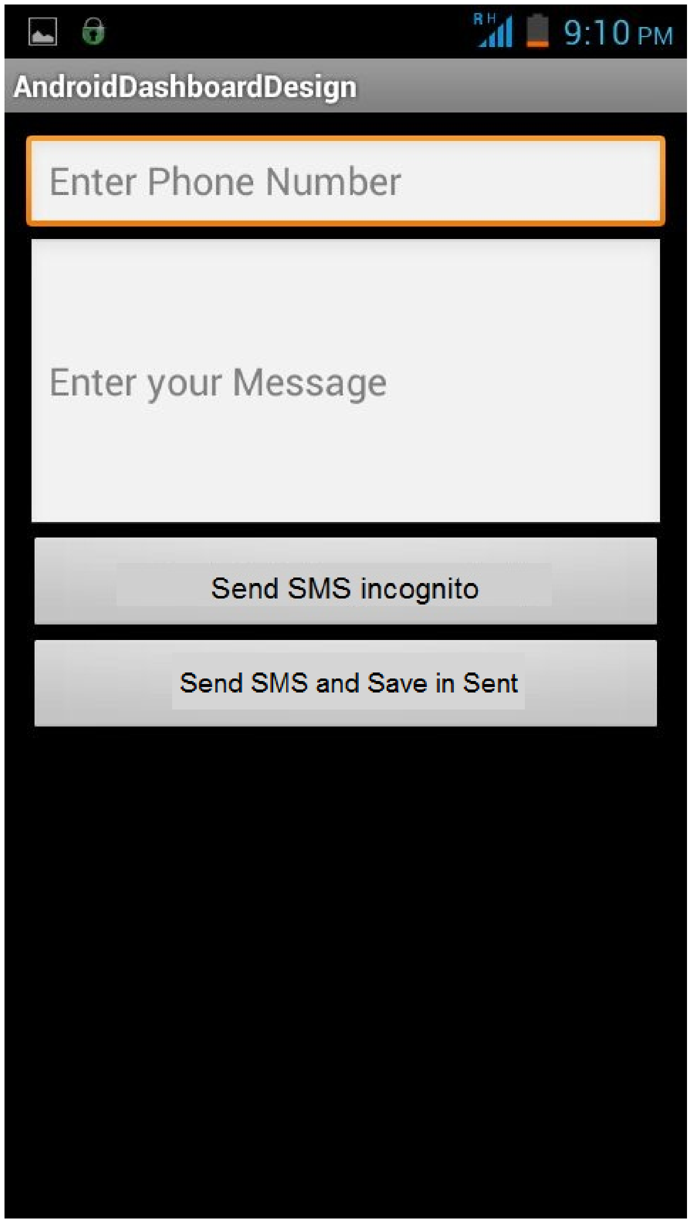
Chapter 9

SNAPSHOTS OF THE APPLICATION

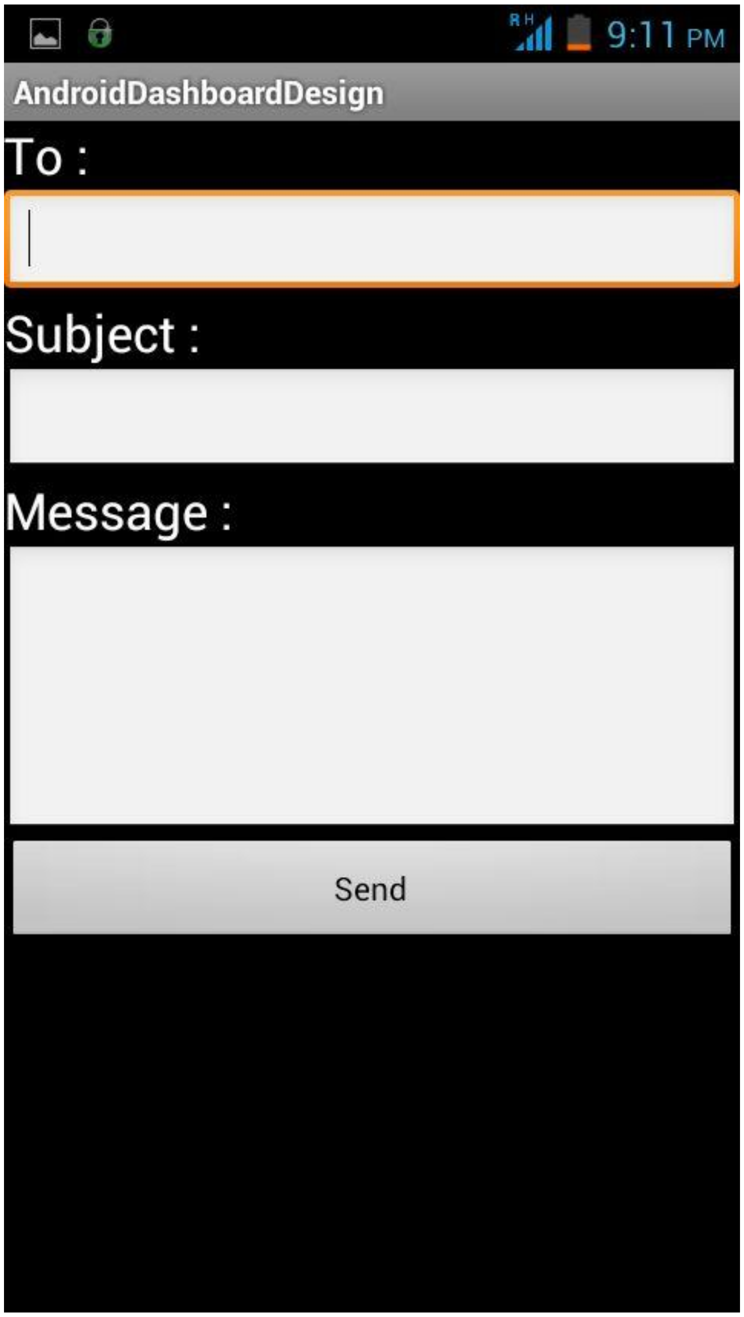
Main dashboard design



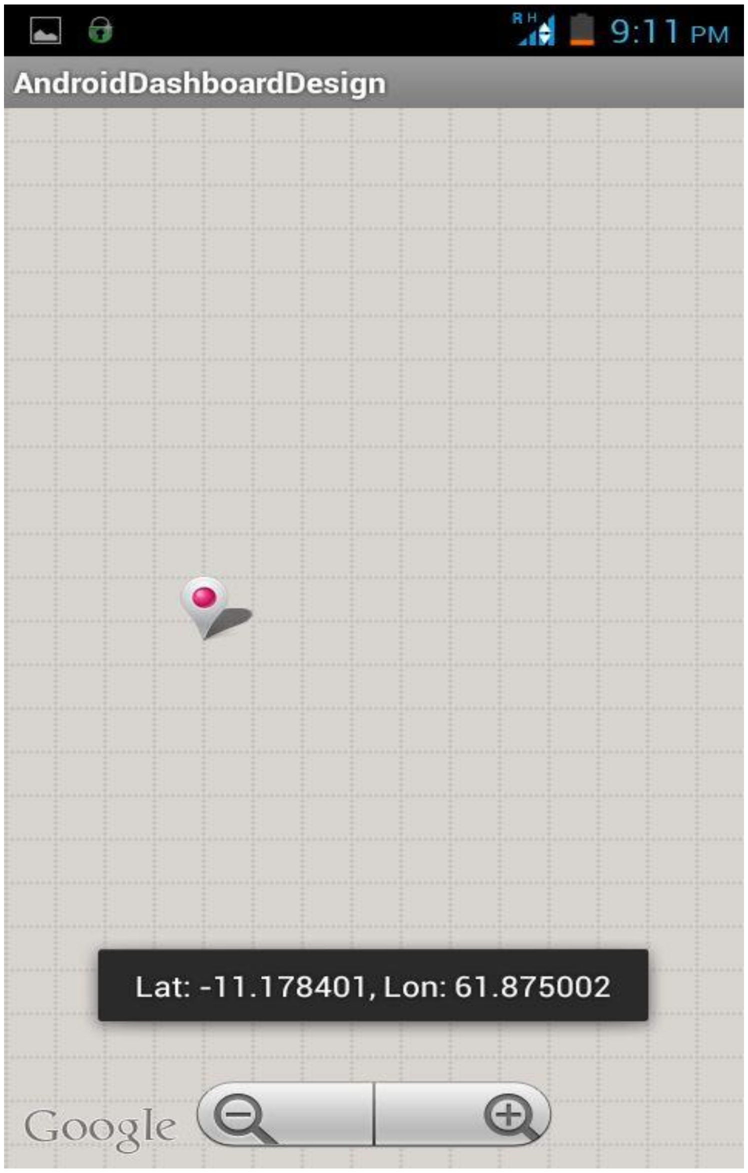
Send SMS


9:10 PM
AndroidDashboardDesign
Enter Phone Number
Enter your Message
Send SMS incognito
Send SMS and Save in Sent

SEND E-MAIL

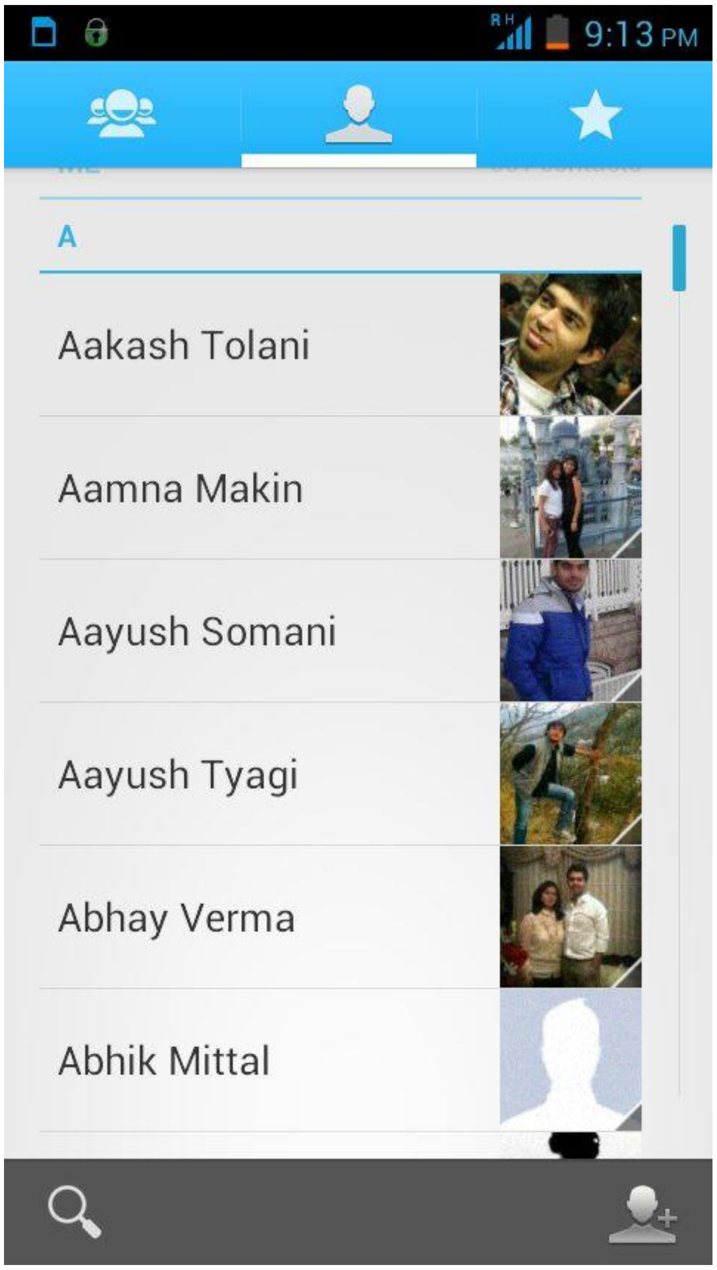


SHOW LOCATION



Lat: -11.178401, Lon: 61.875002

CONTACTS


9:13 PM
A
Aakash Tolani
Aamna Makin
Aayush Somani
Aayush Tyagi
Abhay Verma
Abhik Mittal

Chapter 10
IMPLEMENTATION

AndroidDashboardDesignActivity.java

```
package com.androidhive.dashboard;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import androidhive.dashboard.R;

public class AndroidDashboardDesignActivity extends Activity {

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

      /**
       * Creating all buttons instances
       * */

      Button btn_SMS = (Button) findViewById(R.id.btn_SMS);



      Button btn_EMAIL = (Button) findViewById(R.id.btn_EMAIL);
```

```
        Button btn_GPS = (Button) findViewById(R.id.btn_GPS);



        btn_SMS.setOnClickListener(new View.OnClickListener() {

                        @Override
                        public void onClick(View view) {
                                // Launching News Feed Screen
                                Intent i = new Intent(getApplicationContext(),
SMSDemo.class);
                                startActivity(i);
                        }
                });

        btn_EMAIL.setOnClickListener(new View.OnClickListener() {

                        @Override
                        public void onClick(View view) {
                                Intent i = new Intent(getApplicationContext(),
FriendsActivity.class);
                                startActivity(i);
                        }
                });



        btn_places.setOnClickListener(new View.OnClickListener() {

                        @Override
                        public void onClick(View view) {
                                // Launching News Feed Screen
                        Intent i = new Intent(getApplicationContext(),
GPSTracker.class);
                        startActivity(i);
                    }
                });

    btn_GPS.setOnClickListener(new View.OnClickListener() {

                    @Override
                    public void onClick(View view) {
                        Intent i = new Intent(getApplicationContext(),
AndroidGPSTrackingActivity.class);
                        startActivity(i);
                    }
                });

    btn_scheduler.setOnClickListener(new View.OnClickListener() {

                    @Override
                    public void onClick(View view) {
                        // Launching News Feed Screen
                        Intent i = new Intent(getApplicationContext(),
SchdulerActivity.class);
                        startActivity(i);
                    }
                });
  }
}
```

10.1 Sending SMS Messages

1. Using Eclipse, create a new Android project and name it.

2. SMSDemo.java

```
package com.androidhive.dashboard;
import java.util.ArrayList;

import android.app.Activity;
import android.content.ContentValues;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import androidhive.dashboard.R;


public class SMSDemo extends Activity {

        private EditText phoneNumber, messageText;
        private Button codeButton, intentButton, sendAndSaveButton;


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

    phoneNumber = (EditText)findViewById(R.id.phoneNumber);
```

```
        messageText = (EditText)findViewById(R.id.messageText);

        Button codeButton = (Button) findViewById(R.id.codeButton);
        Button intentButton = (Button) findViewById(R.id.intentButton);
        Button sendAndSaveButton = (Button) findViewById(R.id.sendAndSaveButton);

        codeButton.setOnClickListener(new View.OnClickListener() {
                public void onClick(View view) {
                        if(phoneNumber.getText().toString().trim().length() == 0) {
                                Toast.makeText(getApplicationContext(), "Please enter a
Phone Number.", Toast.LENGTH_LONG).show();
                                return;
                        }

                        if(messageText.getText().toString().trim().length() == 0) {
                                Toast.makeText(getApplicationContext(), "Please enter
your message.", Toast.LENGTH_LONG).show();
                                return;
                        }

                        if(messageText.getText().toString().trim().length() > 160) {
                                sendLongSMS()       ;
                        }
                        else {
                                sendSMS();
                        }
                }
        });

        intentButton.setOnClickListener(new View.OnClickListener() {
                public void onClick(View view) {
```

```
                if(phoneNumber.getText().toString().trim().length() == 0) {
                        Toast.makeText(getApplicationContext(), "Please enter a
Phone Number.", Toast.LENGTH_LONG).show();
                        return;
                }

                if(messageText.getText().toString().trim().length() == 0) {
                        Toast.makeText(getApplicationContext(), "Please enter
your message.", Toast.LENGTH_LONG).show();
                        return;
                }

                invokeSMSApp();
        }
    });


    sendAndSaveButton.setOnClickListener(new View.OnClickListener() {
        public void onClick(View view) {
                if(phoneNumber.getText().toString().trim().length() == 0) {
                        Toast.makeText(getApplicationContext(), "Please enter a
Phone Number.", Toast.LENGTH_LONG).show();
                        return;
                }

                if(messageText.getText().toString().trim().length() == 0) {
                        Toast.makeText(getApplicationContext(), "Please enter
your message.", Toast.LENGTH_LONG).show();
                        return;
                }
                if(messageText.getText().toString().trim().length() > 160) {
                    sendLongSMS();
                    //Save in SENT folder
                    saveInSent();

                }
                else {
                    sendSMS();
                    //Save in SENT folder
                    saveInSent();
                }
            }
     });

  }



  public void sendSMS() {
     //String phoneNo = "0123456789";
     //String message = "Hello World!";

     SmsManager smsManager = SmsManager.getDefault();
     smsManager.sendTextMessage(phoneNumber.getText().toString(),              null,
messageText.getText().toString(), null, null);

     Toast.makeText(getApplicationContext(),          "Message          Sent!",
Toast.LENGTH_LONG).show();
  }
```

```
    public void sendLongSMS() {
        //String phoneNo = "0123456789";
        //String message = "Hello World! Now we are going to demonstrate how to send a
message with more than 160 characters from your Android application.";

        SmsManager smsManager = SmsManager.getDefault();
        ArrayList<String>                    parts                    =
smsManager.divideMessage(messageText.getText().toString());
        smsManager.sendMultipartTextMessage(phoneNumber.getText().toString(),          null,
parts, null, null);

        Toast.makeText(getApplicationContext(),          "Message          Sent!",
Toast.LENGTH_LONG).show();
    }


    public void invokeSMSApp() {
        Intent smsIntent = new Intent(Intent.ACTION_VIEW);

        smsIntent.putExtra("sms_body",        messageText.getText().toString());        //"Hello
World!");
        smsIntent.putExtra("address", phoneNumber.getText().toString()); //"0123456789");
        smsIntent.setType("vnd.android-dir/mms-sms");

        startActivity(smsIntent);
    }


    public void saveInSent() {
            ContentValues values = new ContentValues();
        values.put("address", phoneNumber.getText().toString());

        values.put("body", messageText.getText().toString());

        getContentResolver().insert(Uri.parse("content://sms/sent"), values);
    }
}
```

3.messages_layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#f8f9fe"
    android:orientation="vertical" >
    <include layout="@layout/actionbar_layout" />
    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" >
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:padding="15dip"
            android:text="SCHEDULER"
            android:textColor="#ff29549f"
            android:textSize="25dip"
            android:textStyle="bold" />
    </LinearLayout>

</LinearLayout>
```

10.2 E-MAIL

Email_act.java

```
package com.example.email;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class MainActivity extends Activity {

        Button buttonSend;
        EditText textTo;
        EditText textSubject;
        EditText textMessage;

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

                buttonSend = (Button) findViewById(R.id.buttonSend);
                textTo = (EditText) findViewById(R.id.editTextTo);
                textSubject = (EditText) findViewById(R.id.editTextSubject);
                textMessage = (EditText) findViewById(R.id.editTextMessage);
```

```
                    buttonSend.setOnClickListener(new OnClickListener() {

                              @Override
                              public void onClick(View v) {

                                String to = textTo.getText().toString();
                                String subject = textSubject.getText().toString();
                                String message = textMessage.getText().toString();

                                Intent email = new Intent(Intent.ACTION_SEND);
                                email.putExtra(Intent.EXTRA_EMAIL, new String[]{ to});
                                //email.putExtra(Intent.EXTRA_CC, new String[]{ to});
                                //email.putExtra(Intent.EXTRA_BCC, new String[]{to});
                                email.putExtra(Intent.EXTRA_SUBJECT, subject);
                                email.putExtra(Intent.EXTRA_TEXT, message);

                                //need this to prompts email client only
                                email.setType("message/rfc822");

                                startActivity(Intent.createChooser(email, "Choose an Email client
:"));

                              }
                    });
          }
}
```

Layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/linearLayout1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textViewPhoneNo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="To : "
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/editTextTo"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress" >

        <requestFocus />
    </EditText>

    <TextView
        android:id="@+id/textViewSubject"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Subject : "
        android:textAppearance="?android:attr/textAppearanceLarge" />
    <EditText
        android:id="@+id/editTextSubject"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
         >
    </EditText>

    <TextView
        android:id="@+id/textViewMessage"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Message  : "
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/editTextMessage"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="top"
        android:inputType="textMultiLine"
        android:lines="5" />

    <Button
        android:id="@+id/buttonSend"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Send" />

</LinearLayout>
```

10.3. Location detection

Gps.xml

```
<?xml version="1.0" encoding="utf-8"?>

<com.google.android.maps.MapView

    xmlns:android="http://schemas.android.com/apk/res/android"

    android:id="@+id/mapView"

    android:layout_width="fill_parent"

    android:layout_height="fill_parent"

    android:clickable="true"

    android:apiKey="AIzaSyCZjrgDdTuSuWLnkXiWtErAuMuHQSAnaU8"

/>
```

GpsTrackingActivity.java

```
package com.androidhive.dashboard;

import java.util.List;

import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import com.google.android.maps.OverlayItem;

import android.app.Activity;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import androidhive.dashboard.R;

public class AndroidGPSTrackingActivity extends MapActivity {

      MapView mapView;

      // Map overlay items
      List<Overlay> mapOverlays;

      AddItemizedOverlay itemizedOverlay;

      GeoPoint geoPoint;
      // Map controllers
      MapController mc;

      double latitude;
      double longitude;
      OverlayItem overlayitem;


  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

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



mapView = (MapView) findViewById(R.id.mapView);
          mapView.setBuiltInZoomControls(true);

          mapOverlays = mapView.getOverlays();

          GPSTracker gps = new GPSTracker(AndroidGPSTrackingActivity.this);
          latitude = gps.getLatitude();
          longitude = gps.getLongitude();

          // Geopoint to place on map
          geoPoint = new GeoPoint((int) (latitude * 1E6),
                         (int) (longitude * 1E6));

          // Drawable marker icon
          Drawable drawable_user = this.getResources()
                         .getDrawable(R.drawable.mark_red);

          itemizedOverlay = new AddItemizedOverlay(drawable_user, this);

          // Map overlay item
          overlayitem = new OverlayItem(geoPoint, "Your Location",
                         "That is you!");

          itemizedOverlay.addOverlay(overlayitem);

          mapOverlays.add(itemizedOverlay);
          itemizedOverlay.populateNow();
          mc = mapView.getController();
```

```
                    mc.animateTo(geoPoint);
                    mapView.postInvalidate();
   }

          @Override
          protected boolean isRouteDisplayed() {
                    // TODO Auto-generated method stub
                    return false;
          }

}
```

10.4 SMS scheduler

ReminderTask.java

```
package com.androidhive.dashboard.smsscheduler;

import android.content.ContextWrapper;

import android.telephony.SmsManager;


import com.android.scheduler.NotificationMessage;

import com. android.scheduler.Task;

import com..android.scheduler.TaskResult;

public class ReminderTask implements Task {

                    //@Override

   public String getTitle() {
    return "Reminder";
}


//@Override
public String getId() {
    return "reminder"; // give it an ID
}


//@Override
public TaskResult doWork(ContextWrapper ctx) {
    TaskResult res = new TaskResult();


    // TODO implement your business logic here
    // i.e. query the DB, connect to a web service using HttpUtils, etc..


    SmsManager sm = SmsManager.getDefault();
    //here is where the destination of the text should go
    //String number = "5556";
    //sm.sendTextMessage(number, null, "Test SMS Message", null, null);
   String number = SMSScheduler.getInstance().getPhoneNumber();
   String smsText = SMSScheduler.getInstance().getSMSText();
   sm.sendTextMessage(number, null,smsText, null, null);
```

```
        NotificationMessage notification = new NotificationMessage(
                        "SMS Sent...",
                        "Don't forget to open Hello World App");
        notification.notificationIconResource = R.drawable.icon_notification_cards_clubs;
        notification.setNotificationClickIntentClass(SMSScheduler.class);

        res.addMessage( notification );

        return res;
    }

}
```

SmsScheduler.java

```
package com.androidhive.dashboard.smsscheduler;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.widget.Button;
import android.widget.EditText;
import com.android.scheduler.SchedulerManager;

import com.android.scheduler.analytics.AnalyticsManager;


public class SMSScheduler extends Activity {


        private EditText smsText;

        private EditText phoneNumber;

        private static SMSScheduler instance;

        private String smsTextStr;

        private String phoneNumberString;

        //String smsText;


        @Override

        protected void onCreate(Bundle savedInstanceState) {

                super.onCreate(savedInstanceState);


                requestWindowFeature(Window.FEATURE_INDETERMINATE
        _PROGRESS);

                setContentView(R.layout.main);



                // 1. call BuzzBox Analytics

                int openAppStatus = AnalyticsManager.onOpenApp(this);

                // 2. add the Task to the Scheduler

                if
        (openAppStatus==AnalyticsManager.OPEN_APP_FIRST_TIME) {
```

```
                    // register  the Task when the App in installed

                    SchedulerManager.getInstance().saveTask(this,

                              "*/1  * * * *",   // a cron string

                              ReminderTask.class);

                    SchedulerManager.getInstance().restart(this,
          ReminderTask.class  );

                }                              else                              if
          (openAppStatus==AnalyticsManager.OPEN_APP_UPGRADE){

                    // restart on upgrade

                   SchedulerManager.getInstance().restartAll(this,  0);

                }

                smsText  = (EditText)findViewById(R.id.editTextSMSText);

                phoneNumber                                                       =
          (EditText)findViewById(R.id.editTextPhoneNumber);



                // 3. set up UI buttons

Button  settingsButton  = (Button)  findViewById(R.id.settings);

settingsButton.setOnClickListener(new  View.OnClickListener()  {

                      //@Override

                      public  void  onClick(View  v) {

                              SchedulerManager.getInstance()

                .startConfigurationActivity(SMSScheduler.this,
          ReminderTask.class);

                      }

                });
```

```
 /* Button log = (Button) findViewById(R.id.log);

log.setOnClickListener(new View.OnClickListener() {

                        //@Override

                        public void onClick(View v) {

                                Intent intent = new Intent(SMSScheduler.this,
          SchedulerLogActivity.class);

                                startActivity(intent);

                        }

                });  */

 Button refresh = (Button) findViewById(R.id.notify);

refresh.setOnClickListener(new View.OnClickListener() {

                        //@Override

                        public void onClick(View v) {

                                smsTextStr = smsText.getText().toString();

                                phoneNumberString                                 =
          phoneNumber.getText().toString();


                SchedulerManager.getInstance().runNow(SMSScheduler.this,
          ReminderTask.class, 0);


                SchedulerManager.getInstance().restart(SMSScheduler.this,
          ReminderTask.class);

                        }

                });

instance = this;

          }
```

```
    @Override

    protected void onActivityResult(int requestCode, int resultCode, Intent data) {

        super.onActivityResult(requestCode, resultCode, data);

        if (SchedulerManager.SCHEDULER_CONFIG_REQ_CODE == requestCode &&
                data!=null) {

            SchedulerManager.getInstance()

                .handleConfigurationResult(this, data);

        }

    }

        public String getSMSText(){

                return smsTextStr;

    }

       public String getPhoneNumber(){

                return phoneNumberString;

    }

        public void setSMSText(String strSMSText){

                smsText.setText(strSMSText);

}


    public void setPhoneNumber(String strPhNumber){

                phoneNumber.setText(strPhNumber);

    }

    public static SMSScheduler getInstance(){

                return instance;
    }
}
```

Mainlayout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
    android:layout_height="fill_parent">
            <LinearLayout
                android:orientation="vertical"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent">

            <LinearLayout
                        android:orientation="horizontal"
                        android:layout_width="fill_parent"
                android:layout_height="fill_parent">
                        <TextView android:text="Message" android:id="@+id/textViewMessage"
android:layout_width="wrap_content"
android:layout_height="wrap_content"></TextView>
                        <EditText android:layout_width="fill_parent"
android:id="@+id/editTextSMSText" android:text=""
android:layout_height="wrap_content" android:layout_margin="7dip"></EditText>
            </LinearLayout>

            <LinearLayout
                        android:orientation="horizontal"
                        android:layout_width="fill_parent"
                android:layout_height="fill_parent">
```

```
                    <TextView  android:text="Number"
android:id="@+id/textViewPhoneNumber"  android:layout_width="wrap_content"
android:layout_height="wrap_content"></TextView>
                    <EditText  android:layout_width="fill_parent"
android:id="@+id/editTextPhoneNumber"  android:text=""
android:layout_height="wrap_content"  android:layout_margin="7dip"></EditText>
          </LinearLayout>
          <Button  android:id="@+id/settings"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="Schedule  SMS Settings"
             android:layout_margin="7dip"/>
          <!--<Button android:id="@+id/log"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="Scheduler  Log"
             android:layout_margin="7dip"/>

          --><Button android:id="@+id/notify"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="Send  SMS Now!"
             android:layout_margin="7dip"/>
          </LinearLayout>
</ScrollView>
```

CONCLUSION

MY MESSENGER is a multipurpose application which allows you to do various tasks at the same time. You can send SMS, e-mails and location at the same time. It also allows you to schedule your messages and send them at a later date. There is no doubt about the fact that such applications exist in the market but still all these features in one application itself is the new approach. This is a user-friendly application with no problems in installing and running the application. This application is useful for tracking another person whereabouts. To know where the person is when the location is sent is an altogether a different approach towards the latest phase of innovations.

References

1. Android. (N.D.). *Android Developers*. Retrieved March 6, 2012, from http://developer.android.com/
2. Kolodziej, K. & Hjelm, J. (2006). *Local Positioning Systems: LBS Applications and Services*, CRC Taylor & Francis.
3. Kupper, A. (2005). *Location-Based Services: Fundamentals and Operation*. Wiley.
4. Steiniger, S., Neun, M., & Edwardes, A. (2006). *Foundations of Location-Based Services*. Retrieved January 13, 2012, from http://www.geo.unizh.ch/publications/cartouche/lbs_lecturenotes_steinigeretal2006.pdf
5. Wang, S., Min, J., & Yi, B. K. (2008, May 19-23). Location based services for mobiles: technologies and standards. In *Proceedings of the IEEE International Conference on Communication (ICC)*, Beijing, China.
6. Sumit Kumar, Design and Research Implementation of Android Based WebServer
7. Framework, International Conference on Control, Communication and Computer Technology, 19th November 2011, IRNet, Bangalore.
8. http://www.pearsonhighered.com/samplechapter/0321197984.pdf
9. http://www.griet.ac.in/mca/ppt/J2ME%20Chapter%203.pdf
10. http://staff.um.edu.mt/__data/assets/pdf_file/0003/57171/sd_1.pdf
11. http://www.peerbits.com/waterfall-software-development-model.html
12. http://www.techhew.com/apps-2/sms-scheduler-app/