East West University

Submitted by

Md. Asif Iqbal

ID: 2009-1-60-005

Supervised by

Dr. Shaikh Muhammad Allayear

Assistant Professor

Department of Computer Science and Engineering,

East West University, Dhaka, Bangladesh

A project Submitted in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science & Engineering to the Department of Computer Science & Engineering

At the

East West University

Dhaka, Bangladesh
December 2014

ABSTRACT

Blood Donor's Diary is an application based on Android platform. It premises is to make save life. With this application anyone can make his/her own collection of blood donor's, Able to see donor's address location on google map.

This report provides detailed description on how I created the application and also how I eased the process of android development while creating the application.

Letter of Acceptance

The project entitled "Blood Donor's Diary" submitted by Md.Asif Iqbal, ID No. 2009-1-60-005, to the Department of Computer Science and Engineering , East West University, Dhaka-1212, Bangladesh is accepted by the Department for the partial fulfillment of requirements for the degree of Bachelor of Science in Computer Science and Engineering on December7,2014.

Board of Examiners

_____

Dr. Shaikh Muhammad Allayear,

Assistant Professor,

Department of Computer Science and Engineering,

East West University, Dhaka-1212, Bangladesh

_____

Dr. Shamim Hasnat Ripon,

Chairperson and Associate Professor,

Department of Computer Science and Engineering,

East West University Dhaka-1212, Bangladesh

Acknowledgements

The application Blood Donor's Diary would never run successfully without the valuable encouragement and guidance from my supervisor Dr. Shaikh Muhammad Allayear, Assistant Professor, Department of Computer Science and Engineering, East West University. He enlightened, encouraged and provided us with the ingenuity to transform my vision into reality. I am particularly grateful to Dr. Shamim Hasnat Ripon, Chairperson and Associate Professor, Department of Computer Science and Engineering, East West University, for his encouragement. I am also grateful to many of my friends who provided me with necessary hardware to test the software as it developed. I would like to thank Google for creating android platform and keeping it free.

Table of Contents **PAGES**

Chapter 1

Introduction

1.1 Introduction

Android is the world's most popular operating system for mobile devices and tablets. It is an open source operating system, created by Google, and available to all kinds of developers with various expertise levels, ranging from rookie to professional. Android Inc was founded in Palo Alto of California, U.S. by Andy Rubin, Rich Miner, Nick Sears and Chris White in 2003. Later Android Inc. was acquired by Google in 2005. After original release there have been a number of updates in the original version of Android.Android has a large community of developers writing applications ("apps") that extends the functionality of the devices. Developers write primarily in a customized version of Java. As of July 2013, the Google Play store officially reached over 1 million apps published and over 50 billion downloads. Android apps can be downloaded from third-party sites or through online stores such as Android Market, the app store run by Google. The operating system itself is installed on 400 million total devices. [2].My creativity is to make an android based application that would be an efficient app for smart phone and also an entertaining app for user. So I started working to create an android application for the user for entertainment and life became too easy.

1.2. Motivations

We have usually need blood when an emergency occurs. If we can create or develop our personal blood donor's list by collecting information about friends, relative's and others blood group we can save this list by using Blood Donor's Diary application. When we need blood we can easily contact with donor's and try to save life by donating blood. It's very hard to keep memorize all blood donor's details and sometime we forget when emergency occurs. With concerning all of those problems I have motivated to make this application.

1.3. Objectives

The main objectives of my application are:

· Donor's list in google map

· Donor's search in google map

· Registration

· Login

· Manual search

· Google map view type

.

1.4. Contribution

In survey of existing model I discuss the details about the android. I collect all the requirements to develop an android application there. Software and hardware requirements are also discussed in the chapter to initialize an android development environment. I then move into my project to discuss about it in details. I mention proposed model, its database structure, class diagrams, class descriptions, data flow model, ER diagram, requirements, testing results and requirements to run my application. At last I provide a user manual in Chapter User Manual that describes the proper way of using my application.

The way I followed to reach my goal –

- I collected the necessary information about Android.
- I learned Android programming technique.
- I also collected requirements for my project.
- I used SQLite database system for the application.
- I made all necessary diagrams of my project like Data flow diagram.
- I tested my application and it passed in all the method I applied.

Chapter 2

Literature Review and Survey of Existing Models

2.1. Android

Android is one of the most popular as well as most widely used mobile operating system. It is a software stack for mobile devices that includes an operating system, middleware and key applications [1]. The android SDK provides the tools and APIs necessary to create application on the android platform using JAVA programming language. Android is an open source mobile OS platform that's running on top of Linux kernel. It uses a non-standard Java Virtual Machine called Dalvik, specialized to handle mobile device processes. The main programming language used for Android is JAVA with many libraries from J2SE and third party open source projects, such as Apache commons, SQLite, web kit, etc.

Google purchased the initial developer of the software, Android Inc., in 2005. The unveiling of the Android distribution on November 5, 2007 was announced with the founding of the Open Handset Alliance, a consortium of 84 hardware, software and telecommunication companies devoted to advancing open standards for mobile devices. Google released most of the Android code under the Apache License, a free software license. The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android.

Android consists of a kernel based on the Linux kernel, with middleware, libraries and APIs written in C and application software running on an application framework which includes Java-compatible libraries based on Apache Harmony. Developers write primarily in a customized version of Java.Android became the world's leading smart phone platform at the end of 2010. For the first quarter of 2012, Android had a 59% smart phone market share worldwide. At the half of 2012, there were 400 million devices activated and 1 million activations per day. Analysts point to the advantage to Android of being a multi-channel, multi-carrier OS. [2] At the beginning of 2013, Android captured 70% of smart phone market share worldwide.

2.2. History of Android

2.2.1. Foundation

Android, Inc. was founded in Palo Alto, California, United States in October 2003 by Andy Rubin (co-founder of Danger), Rich Miner (co-founder of Wildfire Communications, Inc.), Nick Sears (once VP at T-Mobile), and Chris White (headed design and interface development at WebTV) to develop, in Rubin's words "...smarter mobile devices that are more aware of its owner's location and preferences". Despite the obvious past accomplishments of the founders and early employees, Android Inc. operated secretly, revealing only that it was working on software for mobile phones. That same year, Rubin ran out of money. Steve Perlman, a close friend of Rubin, brought him $10,000 in cash in an envelope and refused a stake in the company.

2.2.2. Acquisition by Google

Google acquired Android Inc. on August 17, 2005, making Android Inc. a wholly owned subsidiary of Google. Key employees of Android Inc., including Andy Rubin, Rich Miner and Chris White, stayed at the company after the acquisition.

Not much was known about Android Inc. at the time of the acquisition, but many assumed that Google was planning to enter the mobile phone market with this move.

2.2.3. Post-acquisition by Google

At Google, the team led by Rubin developed a mobile device platform powered by the Linux kernel. Google marketed the platform to handset makers and carriers on the promise of providing a flexible, upgradeable system. Google had lined up a series of hardware component and software partners and signaled to carriers that it was open to various degrees of cooperation on their part.

Speculation about Google's intention to enter the mobile communications market continued to build through December 2006. Reports from the BBC and The Wall Street Journal noted that Google wanted its search and applications on mobile phones and it was working hard to deliver that. Print and online media outlets soon reported rumors that Google was developing a Google-branded handset. Some speculated that as Google was defining technical specifications, it was showing prototypes to cell phone manufacturers and network operators.

In September 2007, InformationWeek covered an Evacuee serve study reporting that Google had filed several patent applications in the area of mobile telephony.

2.2.4. Open Handset Alliance

On November 5, 2007, the Open Handset Alliance, a consortium of several companies which

include Broadcom Corporation, Google, HTC, Intel, LG, Marvell Technology Group, Motorola, Nvidia, Qualcomm, Samsung Electronics, Sprint Nextel, T-Mobile and Texas Instruments unveiled itself. The goal of the Open Handset Alliance is to develop open standards for mobile devices. On the same day, the Open Handset Alliance also unveiled their first product, Android, a mobile device platform built on the Linux kernel version 2.6.

On December 9, 2008, 14 new members joined, including ARM Holdings, Atheros Communications, Asustek Computer Inc, Garmin Ltd, Huawei Technologies, PacketVideo, Softbank, Sony Ericsson, Toshiba Corp, and Vodafone Group Plc. [2]

2.2.5. Version history

Android has been updated frequently since the original release of "Astro", with each fixing bugs and adding new features. Each version is named in alphabetical order, with 1.5 "Cupcake" being the first named after a dessert and every update since following this naming convention. [2]

List of Android version names:

1. Cupcake
2. Donut
3. Eclair
4. Froyo
5. Gingerbread
6. Honeycomb

7. Ice Cream Sandwich

8. Android 4.2 Jelly Bean(API level 17)

9. Android 4.3 Jelly Bean (API level 18)

10. Android 4.4 Kit Kat (API level 19)

11.Android 5 Lollipop (API level 21)

2.3 Gingerbread refined the user interface, improved the soft keyboard and copy/paste features, improved gaming performance, SIP support (VoIP calls), and added support for Near Field Communication.

3.0 Honeycomb was a tablet-oriented release which supports larger screen devices and introduces many new user interface features, and supports multicore processors and hardware acceleration for graphics. The Honeycomb SDK has been released and the first device featuring this version, the Motorola Xoom tablet, went on sale in February 2011.

3.1 Honeycomb was announced at the 2011 Google I/O on 10 May 2011. One feature focuses on allowing Honeycomb devices to directly transfer content from USB devices.

3.2 Honeycomb released at July 15 2011, is "an incremental release that adds several new capabilities for users and developers". Highlights include optimization for a broader range of screen sizes; new "zoom-to-fill" screen compatibility mode; capability to load media files directly from the SD card; and an extended screen support API, providing developers with more precise control over the UI. Android 3.2 Honeycomb is the latest Android version that is available to tablets.

4.0.x Ice Cream Sandwich released at December 16, 2011, it's easy multitasking, rich notifications, customizable home screens, resizable widgets, and deep interactivity and adds powerful new ways of communicating and sharing.

4.1.x Jelly Bean released at July 9, 2012 Based on Linux kernel 3.0.31, Jelly Bean was an incremental update with the primary aim of improving the functionality and performance of the user interface. The performance improvement involved "Project Butter", which uses touch anticipation, triple buffering, and extended vsync timing and a fixed frame rate of 60 fps to create a fluid and "buttery-smooth" UI. Android 4.1 Jelly Bean was released to the Android Open Source Project on 9 July 2012, and the Nexus 7 tablet, the first device to run Jelly Bean

4.2. x Jelly Bean released at November 13, 2012 its API level is 17.

4.3. x Jelly Bean released at July 24, 2013API level is 18.

4.4 Kit Kat released at October 31, 2013 API level is 19.

5 Lollipop released at November 12, 2014 API level 21

2.3. Application

Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

Application Framework:

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more. Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities (subject to security constraints enforced by the framework). This same mechanism allows components to be replaced by the user.

Underlying all applications is a set of services and systems, including:

1. A rich and extensible set of Views that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser.

2. Content Providers that enable applications to access data from other applications (such as Contacts), or to share their own data.

3. A Resource Manager, providing access to non-code resources such as localized strings, graphics, and layout files.

4. A Notification Manager that enables all applications to display custom alerts in the status bar.

5. An Activity Manager that manages the lifecycle of applications and provides a common navigation back stack.

2.4 Libraries

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

" System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices.

" Media Libraries - based on Packet Video's Open CORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG.

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

" Lib Web Core - a modern web browser engine which powers both the Android browser and an embeddable web view.

" SGL - the underlying 2D graphics engine.

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

" FreeType - bitmap and vector font rendering.

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

2.5. Linux Kernel

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack. [4]

2.6 Features of Android

Features and current specifications of android are as follow:

Handset layouts

The platform is adaptable to larger, VGA, 2D graphics library, 3D graphics library based on OpenGL ES 2.0 specifications, and traditional Smartphone layouts.

Storage

SQLite, a lightweight relational database, is used for data storage purposes.

Connectivity

Android supports connectivity technologies including GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.

Messaging

SMS and MMS are available forms of messaging, including threaded text messaging and now Android Cloud to Device Messaging Framework (C2DM) is also a part of Android Push Messaging service.

Multiple language support

Android supports multiple human languages. The number of languages more than doubled for the platform 2.3 Gingerbread, 4.0.x Ice Cream Sandwich, 4.1.x Jelly Bean and 4.2.x Jelly Bean

Web browser

The web browser available in Android is based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine. The browser scores a 95/100 on the Acid3 Test.

Java support

While most Android applications are written in Java, there is no Java Virtual Machine in the platform and Java byte code is not executed. Java classes are compiled into Dalvik executables and run on Dalvik, a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. J2ME support can be provided via third-party applications.

Media support

Android supports the following audio/video/still media formats: WebM, H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, OggVorbis, FLAC, WAV, JPEG, PNG, GIF, BMP.

Streaming media support

RTP/RTSP streaming (3GPP PSS, ISMA), HTML progressive download (HTML5 <video> tag). Adobe Flash Streaming (RTMP) and HTTP Dynamic streaming are supported by the Flash plug-in. Apple HTTP Live Streaming is supported by RealPlayer for Mobile, and by the operating system in Android 3.0 (Honeycomb).

Additional hardware support

Android can video/still cameras, touch screens, GPS, accelerometers, gyroscopes, magnetometers, dedicated gaming controls, proximity and pressure sensors, thermometers, accelerated 2D bit blitz (use with hardware orientation, scaling, pixel format conversion) and accelerated 3D graphics.

Multi-touch

Android has native support for multi-touch which was initially made available in handsets such as the HTC One X. The feature was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology at the time). Google has since released an update for the Nexus One and the Motorola Droid which enables multi-touch natively.

Bluetooth

Supports A2DP, AVRCP, sending files (OPP), accessing the phone book (PBAP), voice dialing and sending contacts between phones. Keyboard, mouse and joystick (HID) support is available in Android 3.1+, and in earlier versions through manufacturer customizations and third-party applications.

Video calling

Android does not support native video calling, but some handsets have a customized version of the operating system that supports it, either via the UMTS network (like the Samsung Galaxy S) or over IP. Video calling through Google Talk is available in Android 2.3.4 and later. Gingerbread allows Nexus S to place Internet calls with a SIP account. This allows for enhanced VoIP dialing to other SIP accounts and even phone numbers. Skype 2.1 offers video calling in Android 2.3, 4.0.x, 4.1.x, 4.2.x, including front camera support.

Multitasking

Multitasking of applications is available.

Voice based features

Google search through voice has been available since initial release. Voice actions for calling, texting, navigation, etc. are supported on Android 2.2 onwards.

Tethering

Android supports tethering, which allows a phone to be used as a wireless/wired Wi-Fi hotspot. Before Android 2.2 this was supported by third-party applications or manufacturer customizations.

Screen capture

Android does not support screenshot capture as of 2011. This is supported by manufacturer and third-party customizations. Screen Capture is available in 4.0.x+ and through a PC connection using the DDMS developer's tool. [2]

2.7 Google Play

Google Play is an online software store developed by Google for Android devices. An application program ("app") called "Play Store" is preinstalled on most Android devices and allows users to browse and download apps published by third-party developers, hosted on Google Play. As of February 2013, there were more than 800,000 apps available for Android, and the estimated number of applications downloaded from the Play Store exceeded 20 billion. The operating system itself is installed on 500 million total devices.

Only devices that comply with Google's compatibility requirements are allowed to preinstall and access the Play Store. The app filters the list of available applications to those that are compatible with the user's device, and developers may restrict their applications to particular carriers or countries for business reasons.

Google offers many free applications in the Play Store including Google Voice, Google Goggles, Gesture Search, Google Translate, Google Shopper, Listen and My Tracks. In August 2010, Google launched "Voice Actions for Android", which allows users to search, write messages, and initiate calls by voice.

2.8 Security of Application

Android applications run in a sandbox, an isolated area of the operating system that does not have access to the rest of the system's resources, unless access permissions are granted by the user when the application is installed. Before installing an application, the Play Store displays all required permissions. A game may need to enable vibration, for example, but should not need to read messages or access the phonebook.

After reviewing these permissions, the user can decide whether to install the application. The sandboxing and permissions system weakens the impact of vulnerabilities and bugs in applications, but developer confusion and limited documentation has resulted in applications routinely requesting unnecessary permissions, reducing its effectiveness. The complexity of inter-application communication implies Android may have opportunities to run unauthorized code.

Several security firms have released antivirus software for Android devices, in particular, Lookout Mobile Security, AVG Technologies, Avast!, F-Secure, Kaspersky, McAfee and Symantec. This software is ineffective as sandboxing also applies to such applications, limiting their ability to scan the deeper system for threats.

A useful type of security applications program and service, often described as "Find My Phone", is available for Android, as well as for Microsoft Windows Phone and for Apple iPhone, whereby a registered user can find the approximate location of the phone, if switched on, over the Internet. This helps to locate lost or stolen phones. At least one of these can be installed on a phone after it has gone missing.

2.9 Software Development Tools

2.9.1 Android SDK

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator based on QEMU, documentation, sample code, and tutorials. Currently supported development platforms include computers running Linux (any modern desktop Linux distribution), Mac OS X 10.5.8 or later, Windows XP or later. The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) Plug-in, though developers may use any text editor to edit Java and XML files then use command line tools (Java Development Kit and Apache Ant are required) to create, build and debug Android applications as well as control attached Android devices (e.g., triggering a reboot, installing software package(s) remotely). Enhancements to Android's SDK go hand in hand with the overall Android platform development. The SDK also supports older versions of the Android platform in case developers wish to target their applications at older devices. Development tools are downloadable components, so after one has downloaded the latest version and platform, older platforms and tools can also be downloaded for compatibility testing.

2.9.2 Native Development Kit

Libraries written in C and other languages can be compiled to ARM or x86 native code and installed using the Android Native Development Kit. Native classes can be called from Java code running under the Dalvik VM using the System.loadLibrary call, which is part of the standard Android Java classes.

Complete applications can be compiled and installed using traditional development tools. The ADB debugger gives a root shell under the Android Emulator which allows native ARM code or x 86 codes to be uploaded and executed. ARM or x 86 codes can be

compiled using GCC on a standard PC. Running native code is complicated by the fact that Android uses a non-standard C library (libc, known as Bionic). The underlying graphics device is available as a frame buffer at /dev/graphics/fb0. The graphics library that Android uses to arbitrate and control access to this device is called the Skia Graphics Library (SGL), and it has been released under an open source license. Skia has backend for both win32 and UNIX, allowing the development of cross-platform applications, and it is the graphics engine underlying the Google Chrome web browser.

Unlike Java App development based on the Eclipse IDE, the NDK is based on command-line tools and requires invoking them manually to build, deploy and debug the apps. Several third-party tools allow integrating the NDK into Eclipse and Visual Studio.

2.9.3 Android Open Accessory Development Kit

The Android 3.1 platform (also back ported to Android 2.3.4) introduces Android Open Accessory support, which allows external USB hardware (an Android USB accessory) to interact with an Android-powered device in a special "accessory" mode.

When an Android-powered device is in accessory mode, the connected accessory acts as the USB host (powers the bus and enumerates devices) and the Android-powered device acts as the USB device. Android USB accessories

2.9.4 App Inventor for Android

On 12 July 2010, Google announced the availability of App Inventor for Android, a Web-based visual development environment for novice programmers, based on MIT's Open Blocks Java library and providing access to Android devices' GPS, accelerometer and orientation data, phone functions, text messaging, speech-to-text conversion, contact data, persistent storage, and Web services, initially including Amazon and Twitter. "We could

only have done this because Android's architecture is so open," said the project director, MIT's Hal Abelson. Under development for over a year, the block-editing tool has been taught to non-majors in computer science at Harvard, MIT, Wellesley, Trinity College (Hartford,) and the University of San Francisco, where Professor David Wolber developed an introductory computer science course and tutorial book for non-computer science students based on App Inventor for Android.

2.9.5 Hyper Next Android Creator

Hyper Next Android Creator (HAC) is a software development system aimed at beginner programmers that can help them create their own Android apps without knowing Java and the Android SDK. It is based on HyperCard that treated software as a stack of cards with only one card being visible at any one time and so is well suited to mobile phone applications that have only one window visible at a time. Hyper Next Android Creator's main programming language is simply called Hyper Next and is loosely based on HyperCard's Hyper Talk language.

Hyper Next is an interpreted English-like language and has many features that allow creation of Android applications. It supports a growing subset of the Android SDK including its own versions of the GUI control types and automatically runs its own background service so apps can continue to run and process information while in the background.

2.10 The Simple Project

The goal of Simple is to bring an easy-to-learn-and-use language to the Android platform. Simple is a BASIC dialect for developing Android applications. It targets professional and non-professional programmers alike in that it allows programmers to quickly write Android applications that use the Android runtime components.

Similar to Microsoft Visual Basic 6, Simple programs are form definitions (which contain components) and code (which contains the program logic). The interaction between the components and the program logic happens through events triggered by the components. The program logic consists of event handlers which contain code reacting to the events.

The Simple project is not very active, the last source code update being in August 2009. [5]

2.11. Application Components

Application components are the essential building blocks of an Android application. Each component is a different point through which the system can enter your application. Not all components are actual entry points for the user and some depend on each other, but each one exists as its own entity and plays a specific role-each one is a unique building block that helps define your application's overall behavior.

There are four different types of application components. Each type serves a distinct purpose and has a distinct lifecycle that defines how the component is created and destroyed.

Here are the four types of application components:

- Activities

An activity represents a single screen with a user interface. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. Although the activities work together to form a cohesive user experience in the email application, each one is independent of the others. As such, a different application can start any one of these activities (if the email application allows it). For example, a camera application can start the activity in the email application that composes new mail, in order for the user to share a picture. An activity is implemented as a subclass of Activity and you can learn more about it in the Activities developer guide.

- Services

  A service is a component that runs in the background to perform long-running operations or to perform work for remote processes. A service does not provide a user interface. For example, a service might play music in the background while the user is in a different application, or it might fetch data over the network without blocking user interaction with an activity. Another component, such as an activity, can start the service and let it run or bind to it in order to interact with it. A service is implemented as a subclass of Service and you can learn more about it in the Services developer guide.

- Content providers

  A content provider manages a shared set of application data. You can store the data in the file system, a SQLite database, on the web, or any other persistent storage location your application can access. Through the content provider, other applications can query or even modify the data (if the content provider allows it). For example, the Android system provides a content provider that manages the user's contact information.

As such, any application with the proper permissions can query part of the content provider (such as ContactsContract.Data) to read and write information about a particular person. Content providers are also useful for reading and writing data that is private to your application and not shared. For example, the Note Pad sample application uses a content provider to save notes. A content provider is implemented as a subclass of Content Provider and must implement a standard set of APIs that enable other applications to perform transactions. For more information, see the Content Providers developer guide.

- Broadcast receivers

  A broadcast receiver is a component that responds to system-wide broadcast announcements. Many broadcasts originate from the system-for example, a broadcast announcing that the screen has turned off, the battery is low, or a picture was captured. Applications can also initiate broadcasts-for example, to let other applications know that some data has been downloaded to the device and is available for them to use. Although broadcast receivers don't display a user interface, they may create a status bar notification to alert the user when a broadcast event occurs. More commonly, though, a broadcast receiver is just a "gateway" to other components and is intended to do a very minimal amount of work. For instance, it might initiate a service to perform some work based on the event. A broadcast receiver is implemented as a subclass of Broadcast Receiver and each broadcast is delivered as an Intent object. For more information, see the Broadcast Receiver class.

A unique aspect of the Android system design is that any application can start another application's component. For example, if you want the user to capture a photo with the device camera, there's probably another application that does that and your application can use it, instead of developing an activity to capture a photo yourself.

You don't need to incorporate or even link to the code from the camera application. Instead, you can simply start the activity in the camera application that captures a photo. When complete, the photo is even returned to your application so you can use it. To the user, it seems as if the camera is actually a part of your application. When the system starts a component, it starts the process for that application (if it's not already running) and instantiates the classes needed for the component. For example, if your application starts the activity in the camera application that captures a photo, that activity runs in the process that belongs to the camera application, not in your application's process.

Therefore, unlike applications on most other systems, Android applications don't have a single entry point (there's no main() function, for example).

Because the system runs each application in a separate process with file permissions that restrict access to other applications, your application cannot directly activate a component from another application. The Android system, however, can. So, to activate a component in another application, you must deliver a message to the system that specifies your intent to start a particular component. The system then activates the component for you.

2.12. Activating Components

Three of the four component types-activities, services, and broadcast receivers-are activated by an asynchronous message called an intent. Intents bind individual components to each other at runtime (you can think of them as the messengers that request an action from other components), whether the component belongs to your application or another.

Intent is created with an Intent object, which defines a message to activate either a specific component or a specific type of component-an intent can be either explicit or implicit, respectively.

For activities and services, intent defines the action to perform (for example, to "view" or "send" something) and may specify the URI of the data to act on (among other things that the component being started might need to know). For example, intent might convey a request for an activity to show an image or to open a web page. In some cases, you can start an activity to receive a result, in which case, the activity also returns the result in an Intent (for example, you can issue an intent to let the user pick a personal contact and have it returned to you-the return intent includes a URI pointing to the chosen contact).

For broadcast receivers, the intent simply defines the announcement being broadcast (for example, a broadcast to indicate the device battery is low includes only a known action string that indicates "battery is low").

The other component type, content provider, is not activated by intents. Rather, it is activated when targeted by a request from a Content Resolver. The content resolver handles all direct transactions with the content provider so that the component that's performing transactions with the provider doesn't need to and instead calls methods on the Content Resolver object. This leaves a layer of abstraction between the content provider and the component requesting information (for security).

There are separate methods for activating each type of component:

- You can start an activity (or give it something new to do) by passing an Intent to startActivity() or startActivityForResult() (when you want the activity to return a result).

- You can start a service (or give new instructions to an ongoing service) by passing an Intent to startService(). Or you can bind to the service by passing an Intent to bindService().

- You can initiate a broadcast by passing an Intent to methods like sendBroadcast(), sendOrderedBroadcast(), or sendStickyBroadcast().

- You can perform a query to a content provider by calling query() on a ContentResolver.

For more information about using intents, see the Intents and Intent Filters document. More information about activating specific components is also provided in the following documents: Activities, Services, Broadcast Receiver and Content Providers.

2.13. The Manifest File

Before the Android system can start an application component, the system must know that the component exists by reading the application's AndroidManifest.xml file (the "manifest" file). Your application must declare all its components in this file, which must be at the root of the application project directory.

The manifest does a number of things in addition to declaring the application's components, such as:

- Identify any user permissions the application requires, such as Internet access or read-access to the user's contacts.

- Declare the minimum API Level required by the application, based on which APIs the application uses.

- Declare hardware and software features used or required by the application, such as a camera, Bluetooth services, or a multi touch screen.

- API libraries the application needs to be linked against (other than the Android framework APIs), such as the Google Maps library.

- And more

2.14. Declaring components

The primary task of the manifest is to inform the system about the application's components. For example, a manifest file can declare an activity as follows:

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

<manifest ... >

<applicationandroid:icon="@drawable/app_icon.png" ... >

<activityandroid:name="com.example.project.ExampleActivity"

android:label="@string/example_label" ... >

</activity>

...

</application>

</manifest>

In the <application> element, the android: icon attribute points to resources for an icon that identifies the application.

In the <activity> element, the android: name attribute specifies the fully qualified class name of the Activity subclass and the android: label attributes specifies a string to use as the user-visible label for the activity.
You must declare all application components this way:

- <activity> elements for activities
- <service> elements for services
- <receiver> elements for broadcast receivers
- <provider> elements for content providers

Activities, services, and content providers that you include in your source but do not declare in the manifest are not visible to the system and, consequently, can never run. However, broadcast receivers can be either declared in the manifest or created dynamically in code (as BroadcastReceiver objects) and registered with the system by calling registerReceiver().

2.15 Declaring components capabilities

As discussed above, in Activating Components, you can use an Intent to start activities, services, and broadcast receivers. You can do so by explicitly naming the target component (using the component class name) in the intent. However, the real power of intents lies in the concept of intent actions. With intent actions, you simply describe the type of action you want to perform (and optionally, the data upon which you'd like to perform the action) and allow the system to find a component on the device that can perform the action and start it. If there are multiple components that can perform the

action described by the intent, then the user selects which one to use. The way the system identifies the components that can respond to an intent is by comparing the intent received to the intent filters provided in the manifest file of other applications on the device.

When you declare a component in your application's manifest, you can optionally include intent filters that declare the capabilities of the component so it can respond to intents from other applications. You can declare an intent filter for your component by adding an <intent-filter> element as a child of the component's declaration element.
For example, an email application with an activity for composing a new email might declare an intent filter in its manifest entry to respond to "send" intents (in order to send email). An activity in your application can then create an intent with the ―send‖ action (ACTION_SEND), which the system matches to the email application's ―send‖ activity and launches it when you invoke the intent with startActivity().

2.16 Declaring application requirements

There are a variety of devices powered by Android and not all of them provide the same features and capabilities. In order to prevent your application from being installed on devices that lack features needed by your application, it's important that you clearly define a profile for the types of devices your application supports by declaring device and software requirements in your manifest file. Most of these declarations are informational only and the system does not read them, but external services such as Google Play do read them in order to provide filtering for users when they search for applications from their device.
For example, if any application requires a camera and uses APIs introduced in Android 2.1 (API Level 7), you should declare these as requirements in your manifest file. That way, devices that do not have a camera and have an Android version lower than 2.1 cannot install your application from Google Play. However, you can also declare that your application uses the camera, but does not require it. In that case, your application

must perform a check at runtime to determine if the device has a camera and disable any features that use the camera if one is not available.
Here are some of the important device characteristics that you should consider as you design and develop your application:

- Screen size and density

In order to categorize devices by their screen type, Android defines two characteristics for each device: screen size (the physical dimensions of the screen) and screen density (the physical density of the pixels on the screen, or dpi—dots per inch). To simplify all the different types of screen configurations, the Android system generalizes them into select groups that make them easier to target. The screen sizes are: small, normal, large, and extra large. The screen densities are: low density, medium density, high density, and extra high density. By default, your application is compatible with all screen sizes and densities, because the Android system makes the appropriate adjustments to your UI layout and image resources. However, you should create specialized layouts for certain screen sizes and provide specialized images for certain densities, using alternative layout resources, and by declaring in your manifest exactly which screen sizes your application supports with the <supports-screens> element. For more information, see the Supporting Multiple Screens document.

- Input configurations

Many devices provide a different type of user input mechanism, such as a hardware keyboard, a trackball, or a five-way navigation pad. If your application requires a particular kind of input hardware, then you should declare it in your manifest with the <uses-configuration> element. However, it is rare that an application should require a certain input configuration.

- Device features

There are many hardware and software features that may or may not exist on a given Android-powered device, such as a camera, a light sensor, bluetooth, a certain version of

OpenGL, or the fidelity of the touch screen. You should never assume that a certain feature is available on all Android-powered devices (other than the availability of the standard Android library), so you should declare any features used by your application with the <uses-feature> element.

- Platform Version

Different Android-powered devices often run different versions of the Android platform, such as Android 1.6 or Android 2.3. Each successive version often includes additional APIs not available in the previous version. In order to indicate which set of APIs are available, each platform version specifies an API Level (for example, Android 1.0 is API Level 1 and Android 2.3 is API Level 9). If you use any APIs that were added to the platform after version 1.0, you should declare the minimum API Level in which those APIs were introduced using the <uses-sdk> element. It's important that you declare all such requirements for your application, because, when you distribute your application on Google Play, the store uses these declarations to filter which applications are available on each device. As such, your application should be available only to devices that meet all your application requirements.

2.17 Google Map

Google map is a desktop and mobile web mapping service application and technology provided by Google, offering satellite imagery, street maps, and Street View perspectives, as well as functions such as a route planner for traveling by foot, car, bicycle (beta test), or with public transportation. Also supported are maps embedded on third-party websites via the Google Maps API and a locator for urban businesses and other organizations in numerous countries around the world. Google Maps satellite images are not updated in real time; however, Google adds data to their Primary Database on a regular basis, and most of the images are no more than 3 years old.

The opt-in redesigned version of the desktop application has been available since 2013, alongside the "classic" (pre-2013) version. The redesigned version was met by user criticism regarding slowness, hiding some common functions, removing a scale bar.
Google Maps uses a close variant of the Mercator projection, and therefore cannot accurately show areas around the poles. Google Maps for mobile is the world's most popular app for smart phones.[9]

2.18 Google Map Navigation

Google Maps Navigation is a mobile application developed by Google for the Android and iOS operating systems that was later integrated into the Google Maps mobile app. The application uses an internet connection to a GPS navigation system to provide turn-by-turn voice-guided instructions on how to arrive at a given destination. This application is only available for Android devices.
The application requires connection to Internet data (e.g. 3G,4G,WiFi etc.) and normally uses a GPS satellite connection to determine its location. A user can enter a destination into the application, which will plot a path to it. The app displays the user's progress along the route and issues instructions for each turn.

Chapter 3

Proposed Models

3.1 Design

My application is android application based software. My project contains...

- Donor's list in google map
- Donor's search in google map
- Registration
- Login
- Manual search
- Google map view type

3.2 Implementation Procedure

Creating a fair playing field for android development:
Here are some simple pre-requisites one must have to develop an android app.
Developer Requirement:

- Advanced knowledge of java

- Basic knowledge of XML

Hardware Requirement*:*

Development PC must be a fast one. We used a core i5 machine clocked at @ 3.2 GHz with 4GB ram clocked @1066Mhz. A big monitor or two is also helpful. During debugging it really eases the pain.

3.2.1 Android Development Environment

For developing application we had to create android development environment. Google basically supports the "Eclipse Classic" version. But there are also other IDEs. We have used Eclipse for our development. There are other IDEs like Intelligent Idea. But we choose Eclipse because we wanted such IDE in which we could write java code and at the same time using the same IDE we could work on GUI for android. Eclipse provides both of these features. So we used Eclipse for our project development.

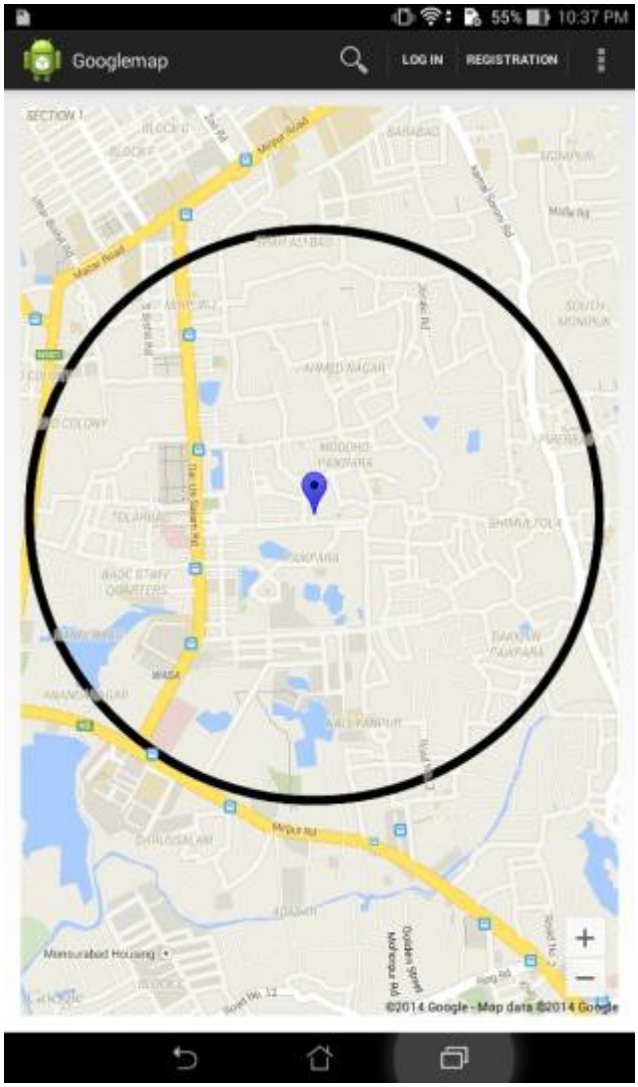

FIGURE 1: My virtual device

3.3.1 Donor's position in google map

Main activity is the basic part of any application. My main activity is located in the src file with the name mainactivity.java and also layout file for this activity is name mainactivity.xml file. After this I have created some button for my further option.

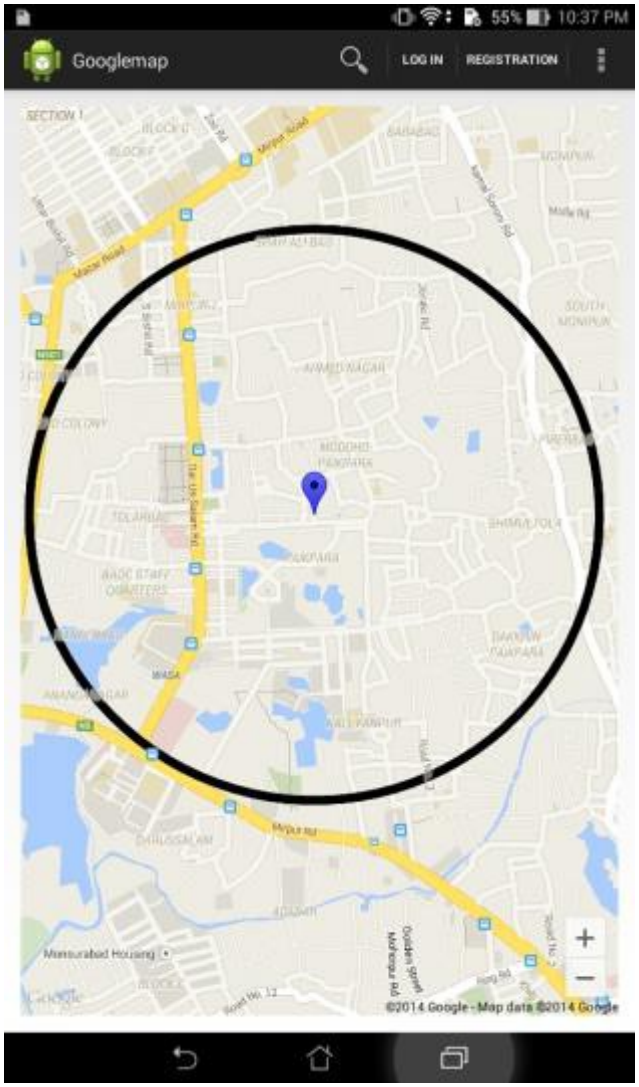

FIGURE 2: Main layout activity file

3.3.2 Donor's search in google map

After creating the main activity file, I have to concern about search click method. It was the hardest part in my project because of maintaining lot of search method I have to create lot of option and also keep maintaining database.

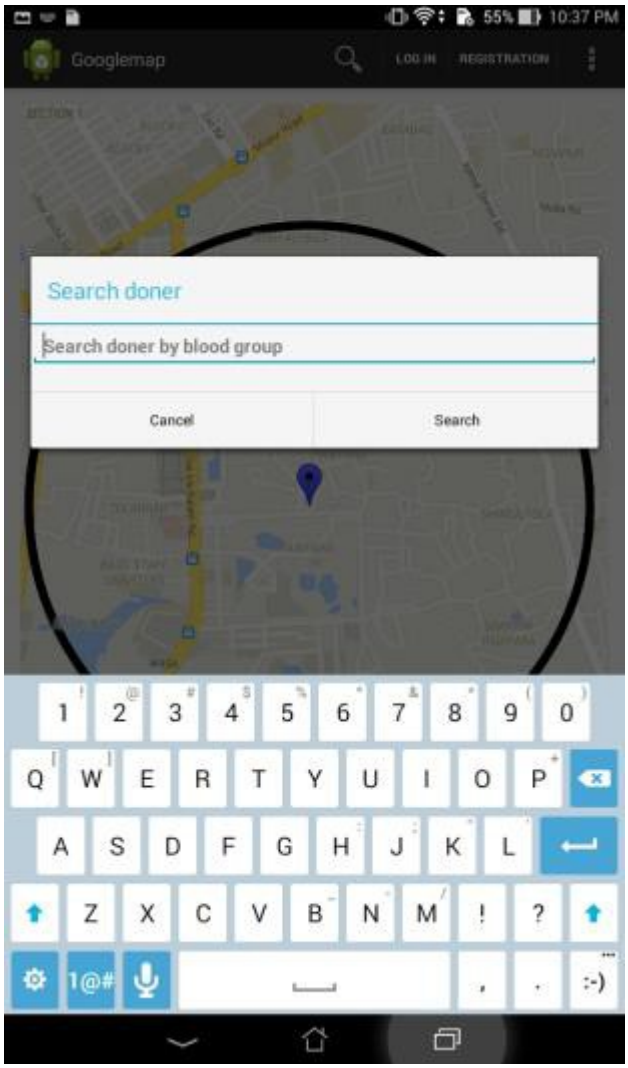

FIGURE 3: Donor Search

3.3.3 Registration

In this option there are lot of field are interconnected to bank account and every field are always connected to SQlite.

Figure 4: Registration

3.3.4 Login

Login is playing another vital activity in my application. Here alert box suggest to give email id & password. Login will be helpful when many user use this application for which we need a central database.

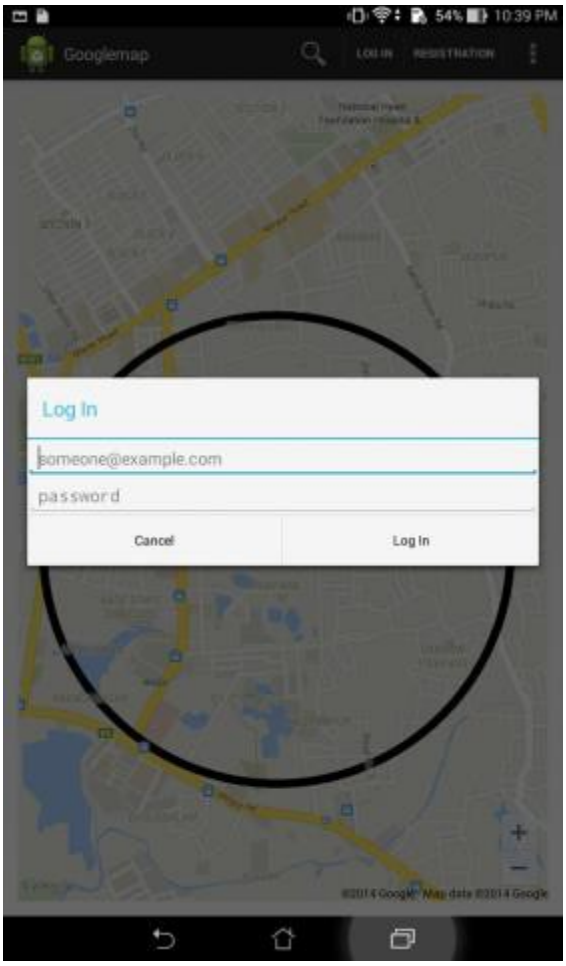

Figure 5 : Log in

3.3.5 Manual Search

In manual search user will be able to see data in a table form. From this table a user can easily call the donor & can send sms. In future we will try to enable email system.

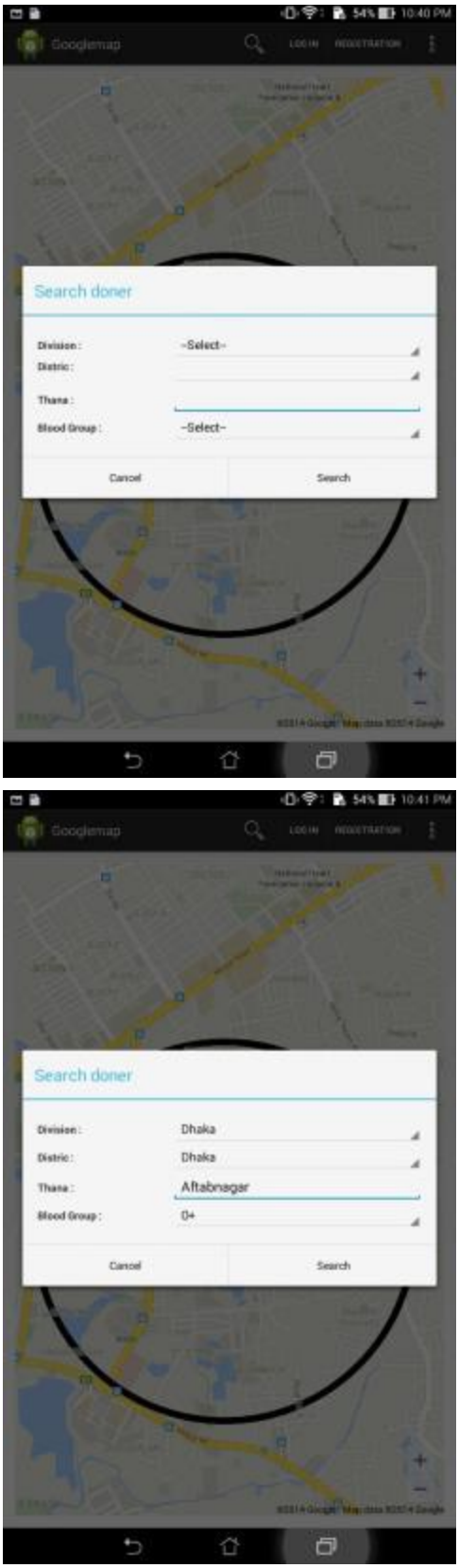

Figure 6 : Manual Search

3.3.6 Google Map view type

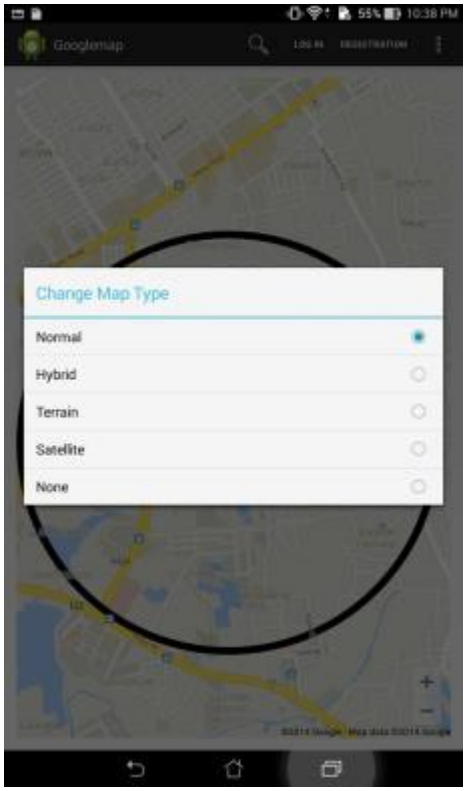

Figure 7 : Google Map Type

3.3.7 Hardware Specification

I have used following devices for development, debugging and testing purpose:

1. Asus Tab.

2. Symphony.

3. Google Nexus 4.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

Android was my choice because it is the most popular, user-friendly mobile operating system of the world. It is also most selling smart phone in the world .I think the application will be able to give the outcome that I wanted from the very beginning of my development process. In the process of developing the application I learned many things about the android operating system and the development related tricks.

I also adopted easier and fresh ways, tricks and techniques that would definitely help in future development. On the other hand it will help other android developers to develop it and modify it according to their way to make this application more fruitful and global. I will not say that I was perfect to make the application. I have also made a lot of mistakes. But I think this will help a lot to develop android application. Blood Donor's Diary application very soon takes a place in the android market. And for sure some modification will come to make it useable globally. It is really very hard task to fulfill all the requirements with an application of smart phone. So there are a lot of chances of further development of the application in future. I have made my application keeping some space for further development.

4.2. Future Work

Each mobile based application is built to be modified as time goes. Blood Donor's Diary has been also built, keeping this fact on mind. Some future modifications I want to make on this application are given below:

- I want to make an iOS version of this application, because besides android, I Phone is also widely used smart phone across the world.

- I also want to modify my application for the newer version of android operating system which will also be compatible to the newer version of Android operating system.

- I wanted to make a web database for Blood Donor's Diary by which when a user got registered a new donor will produced. After that central database will updated. So each & every donor can donate blood to save a life.

- Add more features in Blood Donor's Diary.

References

1. http://venturebeat.com/2013/01/28/android-captured-almost-70-global-smartphone-market-share-in-2012-apple-just-under-20/ [25-11-2014]

2. http://en.wikipedia.org/wiki/Google_Play [21-11-2014]

3. http://en.wikipedia.org/wiki/Android_(operating_system) [21-11-2014]

4. http://forum.xda-developers.com/showthread.php?t=1595487 [23-11-2014]

5. http://developer.android.com/guide/components/fundamentals.html [28-11-2014]

6. http://www.asus.com/Tablets/ASUS_MeMO_Pad_8_ME181C/ [30-11-2014]

7. http://www.gsmarena.com/lg_nexus_4_e960-5048.php [30-11-2014]

8. http://www.bdgsmarena.com/2014/07/symphony-xplorer-h100-price-specs-in.html [1-12-2014]

9. http://en.wikipedia.org/wiki/Google_Maps [21-11-2014]