

CHAPTER 1

INTRODUCTION

Emergency situations, such as accidents, create an immediate, critical need for specific blood type. In addition to emergency requirements, advances in medicine have increased the need for blood in many on-going treatments and elective surgeries. Despite increasing requirements for blood, only about 5% of the Indian population donates blood. In this paper we propose a new and efficient way to overcome such scenarios with our project. We have to create a new idea, just touch the button. Donor will be prompted to enter an individual's details, like name, phone number, and blood type. After that your contact details will appear in alphabetical order on the screen; the urgent time of a blood requirement, you can quickly check for contacts matching a particular or related blood group and reach out to them via Phone Call/SMS through the Blood donor App. Blood Donor App provides list of donors in your city/area. Use this app in case of emergency. A large number of blood donors are attracted using an Android application. Cloud-based services can prove important in emergency blood delivery since they can enable central and immediate access to donors' data and location from anywhere. Since almost everyone carries a mobile phone with him, it ensures instant location tracking and communication. The location-based app, operational on android platform, will help users easily find donors of matching blood groups in their location and access their mobile numbers for instant help. Only a registered person, with willingness to donate blood, will be able to access the service.

It is healthy to donate blood. So, we have created an application to simplify the blood donation process. The donor can easily find out the location where his/her blood group is needed. those locations can either be entities or individuals that urgently need the donor's blood group. When there is an urgent need for a particular blood group, you can use the app to message only the people having the required blood group This system that contains different modules to maintain blood and blood donors. Emergency situations, such as accidents, create an immediate, critical need for specific blood types. In addition to emergency requirements, advances in medicine have increased the need for blood in many on-going treatments and elective surgeries. Despite increasing requirements for blood, only about 5% of the Indian population donates

blood.our major objective is that Since almost everyone carries a mobile phone with him, it ensures instant location tracking and communication. Using GPS and we find donors nearer to the location from where the request is generated. Thus the Mobile Blood Bank can prove to be a boon for blood requesters. And also sending sms to the donors regarding the blood requirements. And also searching whether the particular blood group is available or not. And also getting the details of blood donor's etc.

CHAPTER 2

LITERATURE SURVEY

Santhanam et al extended the nominal definition based on a standard data set to derive a CART based decision tree model based on standard donor ship. This analysis helped identify the attributes that classify regular voluntary donor in the context of a standard dataset. This provided an extended RVD definition based on the donor definition (along with the application of CART) provides a standard model to determine the donor behavior and provides the capability to build a classification model. This additional nominal class can be easily computed based on the statistical definitions and help assist in decision making.

Chau et al have extensively analysed the link ages related to the blood donation to the location of the blood donation centers. This research was carried out using donor's past donation profiles to help setup a new blood donation center for the Hong Kong Red Cross. Their findings provide correlations between spatial distance and the incentive for the blood donors which is the uniqueness of this research. This is specifically helps in the effective setup of center with maximal donorship potential.

Bing et al extensively analysed the working and implementation of blood bank information Systems. Their research provides an extensive background of blood bank information systems. The research also talks about the importance of the decision making capability that is required for effectively running the operations in blood banks. The research also identifies various critical areas that are required for the systems to also have in order to enable decision making.

CHAPTER 3

EXISTING SYSTEM

In existing system the blood bank that store the information about their blood level stacks and the type of the blood available in blood bank. In existing system the high cost is needed for storing and maintain the blood stacks. In existing systems use need a building for storing a blood. So some times the blood stack may be damaged. The recruitment of blood donor when compared with other countries is very less in overall blood donating percentage annually. Besides this recruitment, the screening of donor and the management system is not well maintained. The details of the information of donors are given for the usage of the users for contacting them when in need of blood in case of any emergency. The problem which currently exists in the medical field is that blood is needed immediately for an injured person or for any major operation, it is not easily available even though blood banks are present. There are some websites present for donating blood where the phone numbers of the donors are present which are not reliable since they don't get often updated. At present there are no proper websites. The primary disadvantage of BTS is that there is a concern of many discomforts in immediately following the process. Discomfort in the process is typically minor. However, the users feel weak and light-headed for several hours following the procedure. There is no proper care of person who donates blood to patients. That is the medical history about the donor is not available with the websites.

3.1 DISADVANTAGES

- ☐ Difficulty in handling emergency situation.
- ☐ No proper security for personal details.
- ☐ Misuse by third parties.
- ☐ No proper update about recent details.

- ☐ Needs an intermediate to work manually on information update.
- ☐ Leads to error prone results.

CHAPTER4

PROPOSED SYSTEM

In the present scenario the donor and acceptor communication is illustrated. The bloods are used in case of emergency such as accidents and major operations. Scenario of the proposed method. The proposed method is to create an android application so that the blood donors are available easily within the required time. The donors who are nearby location are searched and nearby blood bank are tracked by the android application using GPS. In this project A Mobile phone with android operating system where the android app is installed. The proposed system is used for maintaining whole information about blood donor.

A. Donor

Each member in a Donor is given a user id and password, which identifies him uniquely. The member given a login form. He enters the login details user id and password.

B. Acceptor

Description about acceptors. Find blood group, which needs blood, A search blood based on blood group and pin code. Acceptor is the one who needs blood for someone related with him. And search traverse nearby blood bank

C. System Database

Stores all the details about the donor, acceptor. the data based used is SQLITE

D. Blood Donation App

An android application created for searching the available donor in a smart phone and traverse nearby blood bank through GPS functionality. Thus the proposed system can help everyone who is need of blood anytime and anywhere. The information given by the donors are completed true so that there will availability of donor at emergency situation.

4.1 OBJECTIVE ON PROBLEM STATED

our major objective is that Since almost everyone carries a mobile phone with him, it ensures instant location tracking and communication. Using GPS and we find donors nearer to the location from where the request is generated. Thus the Mobile Blood Bank can prove to be a boon for blood requesters. And also sending sms to the donors regarding the blood requirements. And also searching whether the particular blood group is available or not. And also getting the details of blood donor's etc .

CHAPTER 5

SYSTEM REQUIREMENTS

Android is a software stack for mobile devices that includes an operating system, middleware and key applications. Google Inc. purchased the initial developer of the software, Android Inc., in 2005. Android's mobile operating system is based on the Linux kernel. Google and other members of the Open Handset Alliance collaborated on Android's development and release.

The Android Open Source Project (AOSP) is tasked with the maintenance and further development of Android. The Android operating system is the world's best-selling Smartphone platform.¹

The Android SDK provides the tools and APIs necessary to begin developing applications Android platform using the Java programming language. Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. There are currently over 250,000 apps available for Android.

Features

- Application framework enabling reuse and replacement of components
- Dalvik virtual machine optimized for mobile devices
- Integrated browser based on the open source WebKit engine
- Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- SQLite for structured data storage
- Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)

- GSM Telephony (hardware dependent)
- Bluetooth, EDGE, 3G, and WiFi (hardware dependent)
- Camera, GPS, compass, and accelerometer (hardware dependent)
- Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

5.1 Android Architecture



Figure 5.1: Android Architecture

5.1.1 Libraries

Android includes a set of C/C++ libraries used by various components of the Android system as shown in figure 5.1. 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
- **LibWebCore** - a modern web browser engine which powers both the Android browser and an embeddable web view
- **SGL** - the underlying 2D graphics engine
- **3D libraries** - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- **FreeType** - bitmap and vector font rendering
- **SQLite** - a powerful and lightweight relational database engine available to all applications

5.1.2 Android Runtime

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

5.1.3 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.

The Linux kernel is an operating system kernel used by the Linux family of Unix-like operating systems. It is one of the most prominent examples of free and open source software. The Linux kernel is released under the GNU General Public License version 2 (GPLv2), (plus some firmware images with various licenses), and is developed by contributors worldwide. Day-to-day development takes place on the Linux kernel mailing list.

The Linux kernel was initially conceived and created by Finnish computer science student Linus Torvalds in 1991. Linux rapidly accumulated developers and users who adapted code from other free software projects for use with the new operating system. The Linux kernel has received contributions from thousands of programmers.^[10] Many Linux distributions have been released based upon the Linux kernel.

The Linux kernel has extensive support for and runs on many virtual machine architectures both as the host operating system and as a guest operating system. The virtual machines usually emulate Intel x86 family of processors, though in a few cases PowerPC or ARM processors are also emulated.

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 premise of providing a flexible, upgradable 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.

5.2 Hardware running Android

The main supported platform for Android is the ARM architecture.

The Android OS can be used as an operating system for cellphones, netbooks and tablets, including the Dell Streak, Samsung Galaxy Tab, TV and other devices. The first commercially available phone to run the Android operating system was the HTC Dream, released on 22 October 2008. In early 2010 Google collaborated with HTC to launch its flagship Android device, the Nexus One. This was followed later in 2010 with the Samsung-made Nexus S.

The early feedback on developing applications for the Android platform was mixed. Issues cited include bugs, lack of documentation, inadequate QA infrastructure, and no public issue-tracking system. In December 2007, MergeLab mobile startup founder Adam MacBeth stated, "Functionality is not there, is poorly documented or just doesn't work... It's clearly not ready for prime time." Despite this, Android-targeted applications began to appear the week after the platform was announced. The first publicly available application was the Snake game. The Android Dev Phone is a SIM-unlocked and hardware-unlocked device that is designed for advanced developers. While developers can use regular consumer devices purchased at retail to test and use their applications, some developers may choose not to use a retail device, preferring an unlocked or no-contract device.

The Android software development kit (SDK) includes a comprehensive set of development tools. These include a debugger, libraries, a handset emulator, documentation, sample code, and tutorials. The SDK is downloadable on the android developer website. Currently supported development platforms include computers running Linux, Mac OS X 10.4.9 or later, Windows XP or later. The officially supported integrated development environment (IDE) is Eclipse using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit Java and XML files then use command line tools to create, build and debug Android applications as well as control attached Android devices.

Android applications are packaged in .apk format and stored under /data/app folder on the Android OS .APK package contains .dex files(compiled byte code files called Dalvikexecutables), resource files, etc.

5.2.1 Android Operation System

Android is an operating system based on Linux with a Java programming interface. It provides tools, e.g. a compiler, debugger and a device emulator as well as its own Java Virtual machine (Dalvik Virtual Machine - DVM). Android is created by the Open Handset Alliance which is lead by Google.

Android uses a special virtual machine, e.g. the Dalvik Virtual Machine. Dalvik uses special bytecode. Therefore you cannot run standard Java bytecode on Android. Android provides a tool "dx" which allows to convert Java Class files into "dex" (Dalvik Executable) files. Android applications are packed into an .apk (Android Package) file by the program "aapt" (Android Asset Packaging Tool) To simplify development Google provides the Android Development Tools (ADT) for Eclipse . The ADT performs automatically the conversion from class to dex files and creates the apk during deployment.

Android supports 2-D and 3-D graphics using the OpenGL libraries and supports data storage in a SQLite database.

Every Android applications runs in its own process and under its own userid which is generated automatically by the Android system during deployment. Therefore the application is isolated from other running applications and a misbehaving application cannot easily harm other Android applications.

Important Android components

An Android application consists out of the following parts:

- Activity - Represents the presentation layer of an Android application, e.g. a screen which the user sees. An Android application can have several activities and it can be switched between them during runtime of the application.
- Views - The User interface of an Activities is build with widgets classes which inherent from "android.view.View". The layout of the views is managed by "android.view.ViewGroups".
- Services - perform background tasks without providing an UI. They can notify the user via the notification framework in Android.
- Content Provider - provides data to applications, via a content provider your application can share data with other applications. Android contains a SQLite DB which can serve as data provider
- Intents are asynchronous messages which allow the application to request functionality from other services or activities. An application can call directly a service or activity (explicit intent) or asked the Android system for registered services and applications for an intent (implicit intents). For example the application could ask via an intent for a contact application. Application register themself to an intent via an IntentFilter. Intents are a powerful concept as they allow to create loosely coupled applications.
- Broadcast Receiver - receives system messages and implicit intents, can be used to react to changed conditions in the system. An application can register as a broadcast receiver for certain events and can be started if such an event occurs.
- A Java Virtual Machine (JVM) enables a set of computer software programs and data structures to use a virtual machine model for the execution of other computer programs and scripts. The model used by a JVM accepts a form of computer intermediate language commonly referred to as Java bytecode. This language conceptually represents the instruction set of a stack-oriented, capability architecture. Sun Microsystems states there are over 4.5 billion JVM-enabled devices
- A JVM can also execute bytecode compiled from programming languages other than Java. For example, Ada source code can be compiled to execute on a JVM. JVMs can also be released by other companies besides Oracle (the developer of Java) — JVMs

using the "Java" trademark may be developed by other companies as long as they adhere to the JVM specification published by Oracle and to related contractual obligations.

- Java was conceived with the concept of WORA: "write once, run anywhere". This is done using the Java Virtual Machine. The JVM is the environment in which java programs execute. It is software that is implemented on non-virtual hardware and on standard operating systems.
- JVM is a crucial component of the Java platform, and because JVMs are available for many hardware and software platforms, Java can be both middleware and a platform in its own right,^[clarification needed] hence the trademark write once, run anywhere. The use of the same bytecode for all platforms allows Java to be described as "compile once, run anywhere", as opposed to "write once, compile anywhere", which describes cross-platform compiled languages. A JVM also enables such features as automated exception handling, which provides "root-cause" debugging information for every software error (exception), independent of the source code.
- A JVM is distributed along with a set of standard class libraries that implement the Java application programming interface (API). Appropriate APIs bundled together form the Java Runtime Environment (JRE).
- Java's execution environment is termed the Java Runtime Environment, or JRE.
- Programs intended to run on a JVM must be compiled into a standardized portable binary format, which typically comes in the form of .class files. A program may consist of many classes in different files. For easier distribution of large programs, multiple class files may be packaged together in a .jar file (short for Java archive).
- The Java application launcher, java, offers a standard way of executing Java code. Compare javaw.
- The JVM runtime executes .class or .jar files, emulating the JVM instruction set by interpreting it, or using a just-in-time compiler (JIT) such as Oracle's HotSpot. JIT compiling, not interpreting, is used in most JVMs today to achieve greater speed. There are also ahead-of-time compilers that enable developers to precompile class files into native code for particular platforms.

- Like most virtual machines, the Java Virtual Machine has a stack-based architecture akin to a microcontroller/microprocessor. However, the JVM also has low-level support for Java-like classes and methods, which amounts to a highly idiosyncratic memory model and capability-based architecture.

Download the Android SDK

Welcome Developers! If you are new to the Android SDK, please read the steps below, for an overview of how to set up the SDK.

Here an overview of the steps you must follow to set up the Android SDK:

1. Prepare your development computer and ensure it meets the system requirements.
2. Install the SDK starter package from the table above. (If you're on Windows, download the installer for help with the initial setup.)
3. Install the ADT Plugin for Eclipse (if you'll be developing in Eclipse).
4. Add Android platforms and other components to your SDK.
5. Explore the contents of the Android SDK (optional).

To get started, download the appropriate package from the table above, then read the guide to Installing the SDK.

Installing the SDK

Step 1:-Preparing Your Development Computer

Before getting started with the Android SDK, take a moment to confirm that your development computer meets the System Requirements. In particular, you might need to install the JDK, if you don't have it already.

If you will be developing in Eclipse with the Android Development Tools (ADT) Plugin—the recommended path if you are new to Android—make sure that you have a suitable

version of Eclipse installed on your computer as described in the System Requirements document. If you need to install Eclipse, you can download it from this location:

The "Eclipse Classic" version is recommended. Otherwise, a Java or RCP version of Eclipse is recommended. Use the Eclipse update manager to install all available plugins for the Android Development Tools (ADT) from the URL <https://dl-ssl.google.com/android/eclipse/>. Configuration

In Eclipse open the Preferences dialog as shown in figure 5.2 via Windows -> Preferences. Select Android and maintain the installation path of the Android SDK.

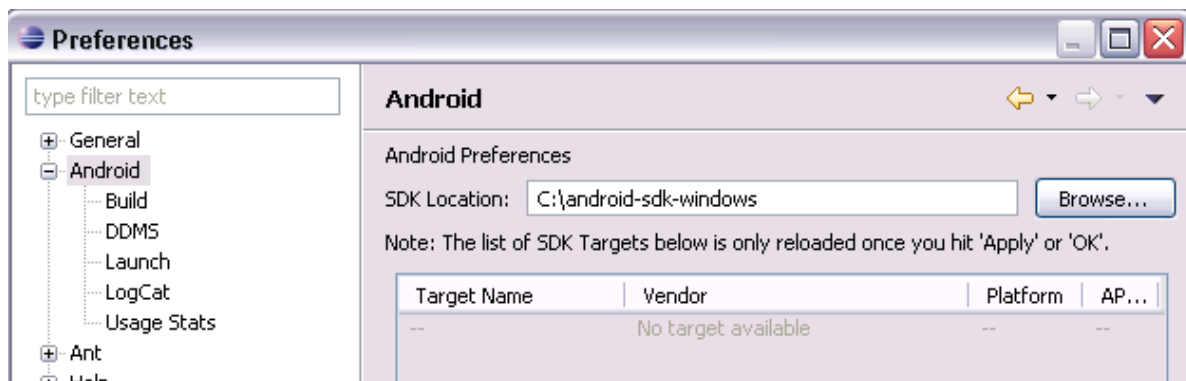
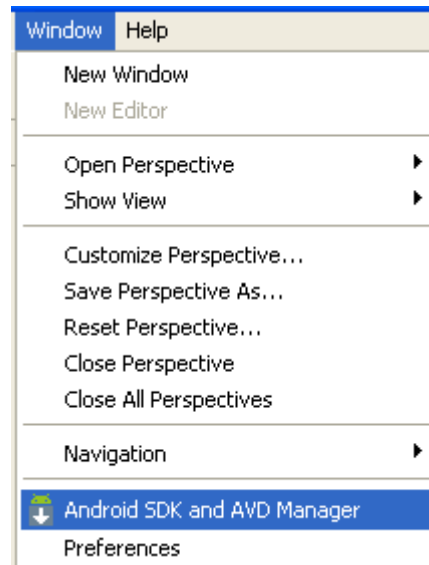


Figure 5.2: Preferences dialog

Select Window -> Android SDK and AVD Manager from the menu.



Select available packages and select the latest version of the SDK.

Step 2:- Downloading the SDK Starter Package

The SDK starter package is not a full development environment—it includes only the core SDK Tools, which you can use to download the rest of the SDK components (such as the latest Android platform). If you haven't already, get the latest version of the SDK starter package from the SDK download page.

If you downloaded a .zip or .tgz package (instead of the SDK installer), unpack it to a safe location on your machine. By default, the SDK files are unpacked into a directory named android-sdk-<machine-platform>.

If you downloaded the Windows installer (.exe file), run it now and it will check whether the proper Java SE Development Kit (JDK) is installed, then install the SDK Tools into a default location.

Make a note of the name and location of the SDK directory on your system as shown in figure 5.3—you will need to refer to the SDK directory later, when setting up the ADT plugin and when using the SDK tools from the command line.

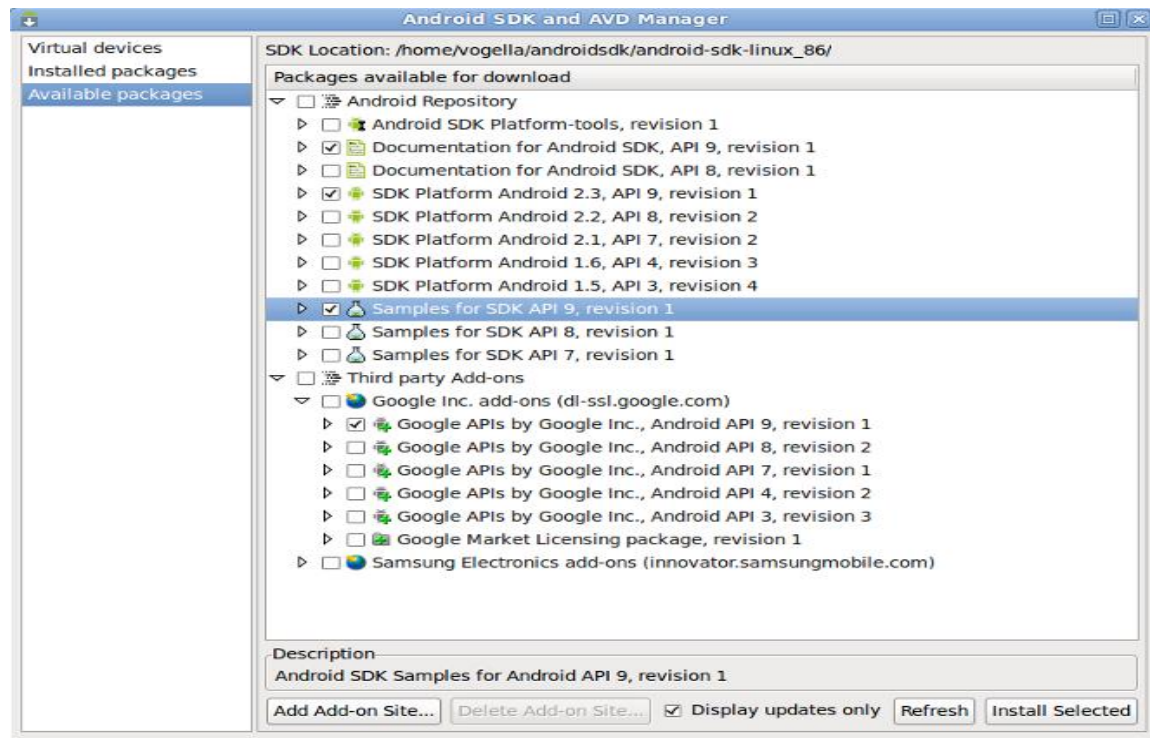


Figure 5.3: Android SDK

Step 3:- Installing the ADT Plugin for Eclipse

Android offers a custom plugin for the Eclipse IDE, called Android Development Tools (ADT), that is designed to give you a powerful, integrated environment in which to build Android applications. It extends the capabilities of Eclipse to let you quickly set up new Android projects, create an application UI, debug your applications using the Android SDK tools, and even export signed APKs in order to distribute your application. In general, developing in Eclipse with ADT is a highly recommended approach and is the fastest way to get started with Android.

If you'd like to use ADT for developing Android applications, install it now. Read Installing the ADT Plugin for step-by-step installation instructions, then return here to continue the last step in setting up your Android SDK.

If you prefer to work in a different IDE, you do not need to install Eclipse or ADT. Instead, you can directly use the SDK tools to build and debug your application as shown in

figure 5.4. The Introduction to Android application development outlines the major steps that you need to complete when developing in Eclipse or other IDEs.

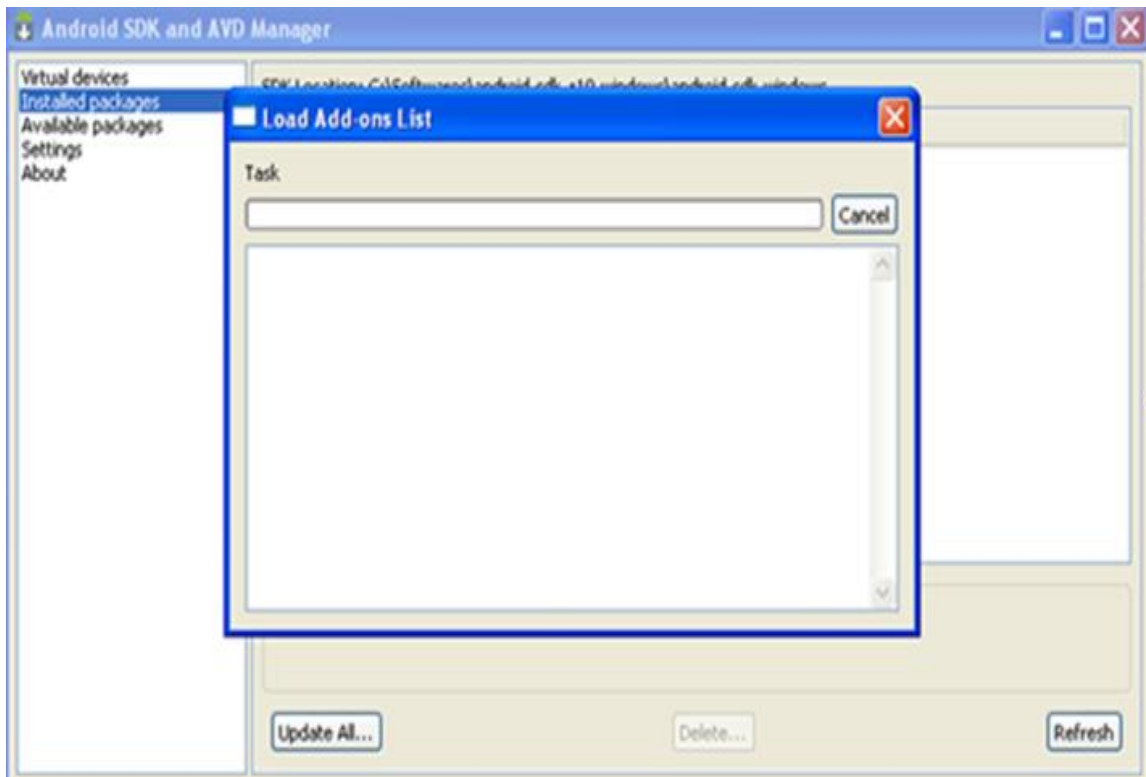


Figure 5.4: load Add.ons list

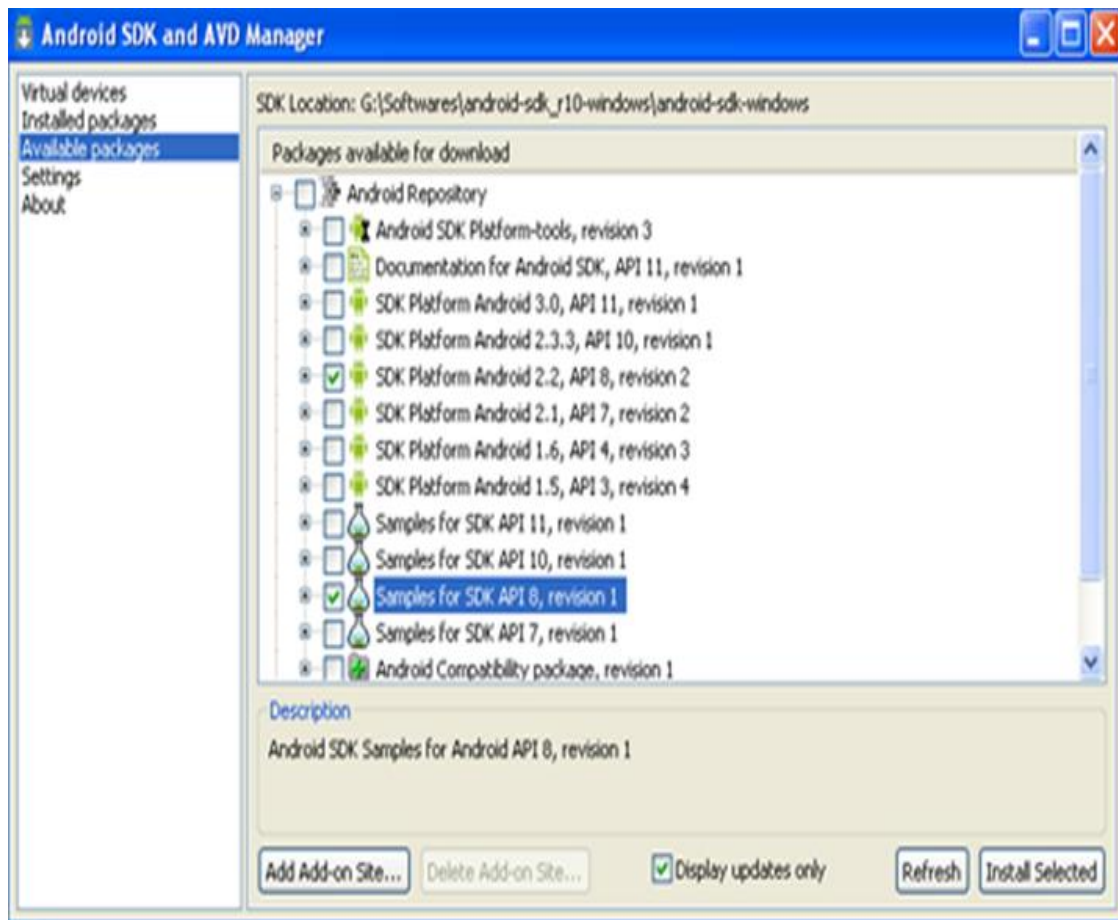


Figure 5.5: Available packages

Step 4:- Adding Platforms and Other Components

The last step in setting up your SDK is using the Android SDK and AVD Manager (a tool included in the SDK starter packages shown in figure 5.5) to download essential SDK components into your development environment.

The SDK uses a modular structure that separates the major parts of the SDK—Android platform versions, add-ons, tools, samples, and documentation—into a set of separately installable components. The SDK starter package, which you've already downloaded, includes only a single component: the latest version of the SDK Tools. To develop an Android application, you also need to download at least one Android platform and the associated platform

tools as shown in figure 5.7. You can add other components and platforms as well, which is highly recommended.

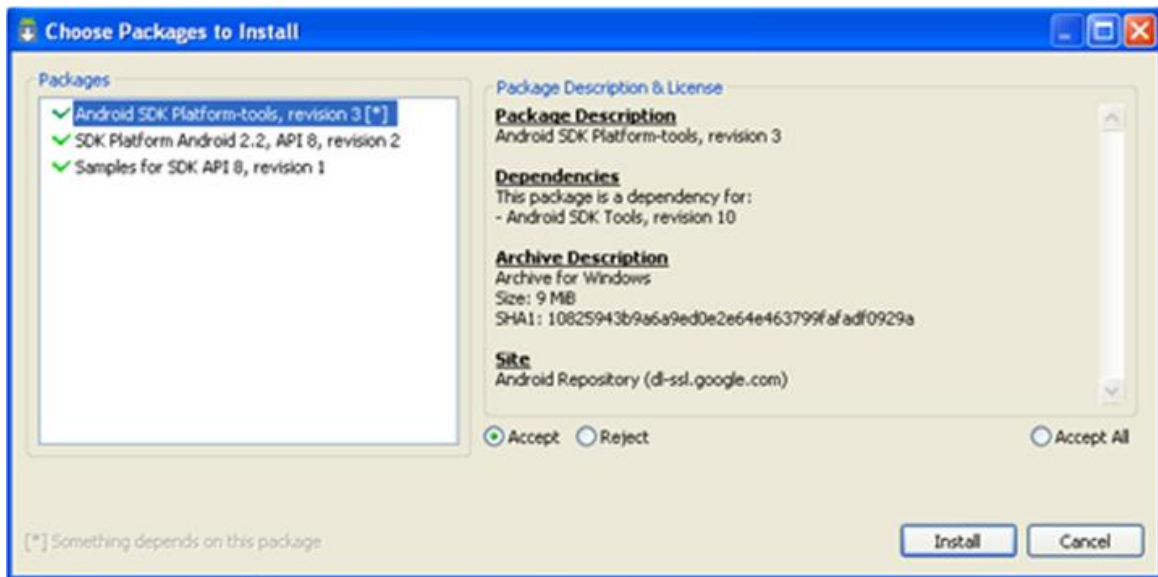


Figure 5.6: Choosing packages

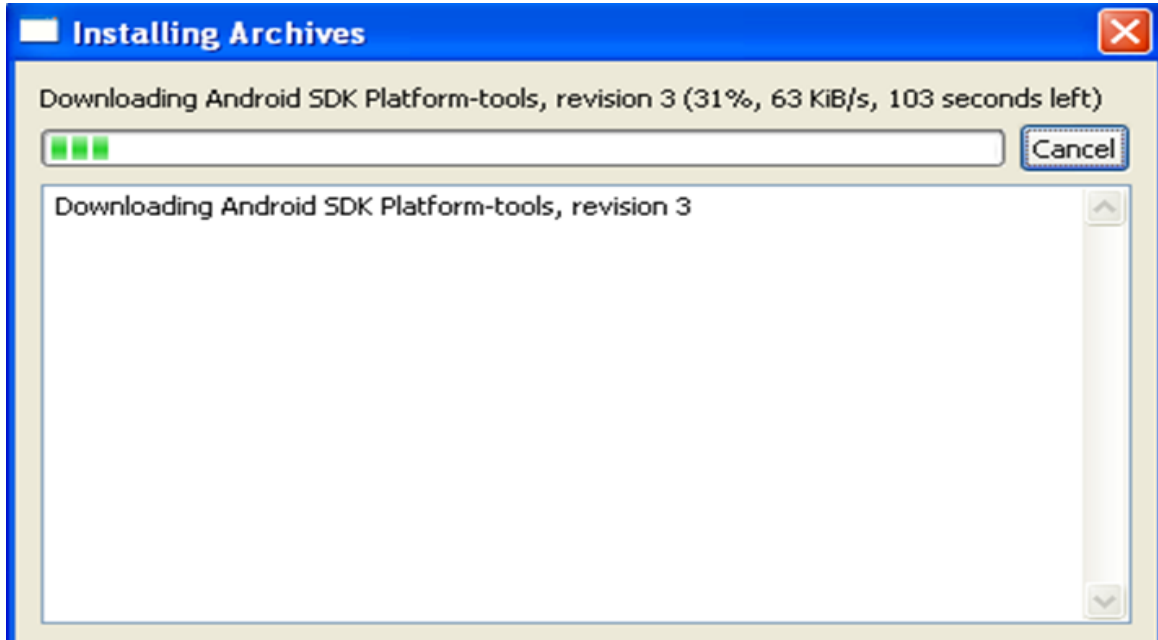


Figure 5.7: Installing Archives

If you used the Windows installer, when you complete the installation wizard, it will launch the Android SDK and AVD Manager with a default set of platforms and other components selected for you to install. Simply click install to accept the recommended set of components and install them. You can then skip to Step 5, but we recommend you first read the section about the Available Components to better understand the components available from the Android SDK and AVD Manager.

You can launch the Android SDK and AVD Manager in one of the following ways:

- From within Eclipse, select Window > Android SDK and AVD Manager.
- On Windows, double-click the SDK Manager.exe file at the root of the Android SDK directory.
- On Mac or Linux, open a terminal and navigate to the tools/ directory in the Android SDK, then execute:

To download components, use the graphical UI of the Android SDK and AVD Manager to browse the SDK repository and select new or updated components (see figure 1). The Android SDK and AVD Manager installs the selected components in your SDK environment. For information about which components you should download, see Recommended Components.

The *Android Repository* offers these types of components:

- **SDK Tools** — Contains tools for debugging and testing your application and other utility tools. These tools are installed with the Android SDK starter package and receive periodic updates. You can access these tools in the <sdk>/tools/ directory of your SDK. To learn more about them, see SDK Tools in the developer guide.
- **SDK Platform-tools** — Contains platform-dependent tools for developing and debugging your application. These tools support the latest features of the Android platform and are typically updated only when a new platform becomes available. You can access these tools in the <sdk>/platform-tools/ directory. To learn more about them, see Platform Tools in the developer guide.

- **Android platforms** — An SDK platform is available for every production Android platform deployable to Android-powered devices. Each SDK platform component includes a fully compliant Android library, system image, sample code, and emulator skins. To learn more about a specific platform, see the list of platforms that appears under the section "Downloadable SDK Components" on the left part of this page.
- **USB Driver for Windows** (Windows only) — Contains driver files that you can install on your Windows computer, so that you can run and debug your applications on an actual device. You *do not* need the USB driver unless you plan to debug your application on an actual Android-powered device. If you develop on Mac OS X or Linux, you do not need a special driver to debug your application on an Android-powered device. See Using Hardware Devices for more information about developing on a real device.
- **Samples** — Contains the sample code and apps available for each Android development platform. If you are just getting started with Android development, make sure to download the samples to your SDK.
- **Documentation** — Contains a local copy of the latest multiversion documentation for the Android framework API.

The *Third party Add-ons* provide components that allow you to create a development environment using a specific Android external library (such as the Google Maps library) or a customized (but fully compliant) Android system image. You can add additional Add-on repositories by clicking Add Add-on Site.

5.3 ECLIPSE

Eclipse is an open source community whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software lifecycle. Many people know us, and hopefully love us, as a Java IDE but Eclipse is much more than a Java IDE.

The Eclipse open source community has over 60 open source projects. These projects can be conceptually organized into seven different "pillars" or categories:

1. Enterprise Development
2. Embedded and Device Development
3. Rich Client Platform
4. Rich Internet Applications
5. Application Frameworks
6. Application Lifecycle Management (ALM)
7. Service Oriented Architecture (SOA)

The Eclipse community is also supported by a large and vibrant ecosystem of major IT solution providers, innovative start-ups, universities and research institutions and individuals that extend, support and complement the Eclipse Platform.

The exciting thing about Eclipse is many people are using Eclipse in ways that we have never imagined. The common thread is that they are building innovative, industrial strength software and want to use great tools, frameworks and runtimes to make their job easier.

Eclipse is a multi-language software development environment comprising an integrated development environment (IDE) and an extensible plug-in system. It is written mostly in Java and can be used to develop applications in Java and, by means of various plug-ins, other programming languages including Ada, C, C++, COBOL, Perl, PHP, Python, Ruby (including Ruby on Rails framework), Scala, Clojure, and Scheme. The IDE is often called Eclipse ADT for Ada, Eclipse CDT for C/C++, Eclipse JDT for Java, and Eclipse PDT for PHP.

5.3.1 ARCHITECTURE

Eclipse employs plug-ins in order to provide all of its functionality on top of (and including) the runtime system, in contrast to some other applications where functionality is typically hard coded. The runtime system of Eclipse is based on Equinox, an OSGi standard compliant implementation.

This plug-in mechanism is a lightweight software componentry framework. In addition to allowing Eclipse to be extended using other programming languages such as C and Python, the

plug-in framework allows Eclipse to work with typesetting languages like LaTeX,^[2] networking applications such as telnet, and database management systems. The plug-in architecture supports writing any desired extension to the environment, such as for configuration management. Java and CVS support is provided in the Eclipse SDK, with Subversion support provided by third-party plug-ins.

With the exception of a small run-time kernel, everything in Eclipse is a plug-in. This means that every plug-in developed integrates with Eclipse in exactly the same way as other plug-ins; in this respect, all features are "created equal". Eclipse provides plug-ins for a wide variety of features, some of which are through third parties using both free and commercial models. Examples of plug-ins include a UML plug-in for Sequence and other UML diagrams, a plug-in for DB Explorer, and many others.

The Eclipse SDK includes the Eclipse Java Development Tools (JDT), offering an IDE with a built-in incremental Java compiler and a full model of the Java source files. This allows for advanced refactoring techniques and code analysis. The IDE also makes use of a workspace, in this case a set of metadata over a flat filesystem allowing external file modifications as long as the corresponding workspace "resource" is refreshed afterwards.

Eclipse implements widgets through a widget toolkit for Java called SWT, unlike most Java applications, which use the Java standard Abstract Window Toolkit (AWT) or Swing. Eclipse's user interface also uses an intermediate GUI layer called JFace, which simplifies the construction of applications based on SWT.

5.3.2 Rich Client Platform

- Equinox OSGi – a standard bundling framework
- Core platform – boot Eclipse, run plug-ins
- Standard Widget Toolkit (SWT) – a portable widget toolkit
- JFace – viewer classes to bring model view controller programming to SWT, file buffers, text handling, text editors
- Eclipse Workbench – views, editors, perspectives, wizards

5.3.3 History

Eclipse began as an IBM Canada project. It was developed by Object Technology International (OTI) as a Java-based replacement for the Smalltalk based VisualAge family of IDE products,^[4] which itself had been developed by OTI. In November 2001, a consortium was formed to further the development of Eclipse as open source. In January 2004, the Eclipse Foundation was created.

Eclipse 3.0 (released on 21 June 2004) selected the OSGi Service Platform specifications as the runtime architecture. Eclipse was originally released under the Common Public License, but was later relicensed under the Eclipse Public License. The Free Software Foundation has said that both licenses are free software licenses, but are incompatible with the GNU General Public License (GPL). Mike Milinkovich, of the Eclipse Foundation commented that moving to the GPL would be considered when version 3 of the GPL was released.

According to Lee Nackman, Chief Technology Officer of IBM's Rational division at that time and later head of Rational software development and support, the name "Eclipse" was chosen to target Microsoft's Visual Studio product, and not Sun Microsystems.^[9] Ironically, Nackman is now himself a Microsoft employee.

5.3.4 Eclipse (SDK)

Eclipse Software Development Kit (SDK) is a Java based open-source integrated development environment (IDE) which combines a number of different Eclipse projects including Platform, Java Development Tools (JDT) and the Plug-in Development Environment (PDE).

Eclipse can be used to create a large array of software applications using languages ranging from PHP, C++ programs, to Java. It is one of the most popular development tools in both the open-source and commercial worlds.

It provides Java editing with validation, incremental compilation, cross-referencing, code assist; an XML Editor; Mylyn; and much more.

Eclipse is released under the Eclipse Foundation, a commercially friendly license that allows organizations to include Eclipse software in their commercial products, while at the same time asking those who create derivative works of EPL code to contribute back to the community.

Eclipse 3.0 (released on 21 June 2004) selected the OSGi Service Platform specifications as the runtime architecture as shown in figure 5.8. Eclipse was originally released under the Common Public License, but was later relicensed under the Eclipse Public License. The Free Software Foundation has said that both licenses are free software licenses, but are incompatible with the GNU General Public License (GPL). Mike Milinkovich, of the Eclipse Foundation commented that moving to the GPL would be considered when version 3 of the GPL was released.

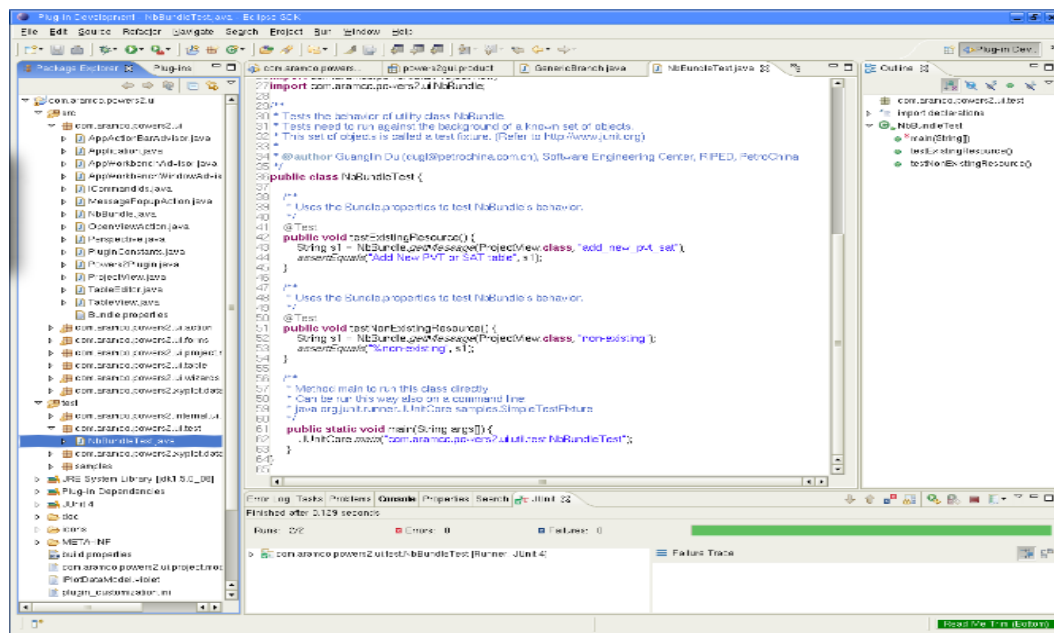


Figure 5.8: Eclipse SDK

5.3.5 Eclipse Platform

The Eclipse Platform provides the core frameworks and services upon which all plug-in extensions are created. It also provides the runtime in which plug-ins are loaded, integrated, and

executed. The primary purpose of the Platform is to enable other tool developers to easily build and deliver integrated tools.

Features include:

- Supports the construction of a variety of tools for application development
- Supports an unrestricted set of tool providers, including independent software vendors (ISVs)
- Supports tools to manipulate arbitrary content types (e.g., HTML, Java, C, JSP, EJB, XML, and GIF)
- Facilitates seamless integration of tools within and across different content types and tool providers
- Supports both GUI and non-GUI-based application development environments

Java Development Tools (JDT)

The JDT project provides the tool plug-ins that implement a Java IDE supporting the development of any Java application, including Eclipse plug-ins. It adds a Java project nature and Java perspective to the Eclipse Workbench as well as a number of views, editors, wizards, builders, and code merging and refactoring tools. The JDT project allows Eclipse to be a development environment for itself.

Features include:

- Java projects with source files arranged in package directories
- Editing with keyword and syntax coloring, outline showing declaration structure
- Code formatter
- Refactoring
- Search
- Compare
- Compile - JCK-compliant Java compiler
- Run Java programs in a separate target Java virtual machine

- Debug programs with JPDA-compliant Java virtual machine

5.4 Android Source Code

The following step is optional.

During Android development it is very useful to have the Android source code available as Android uses a lot of defaults. HarisPeco maintains plugins with provides access to the Android Source code. Use the Eclipse update manager to install two of his plugins.

Create an Android Emulator Device

The Android tools include an emulator. This emulator behaves like a real Android device in most cases and allow you to test your application without having a real device. You can emulate one or several devices with different configurations. Each configuration is defined via an "Android Virtual Device" (AVD) as shown in figure 5.9. To define an AVD press the device manager button, press "New" and maintain the following.

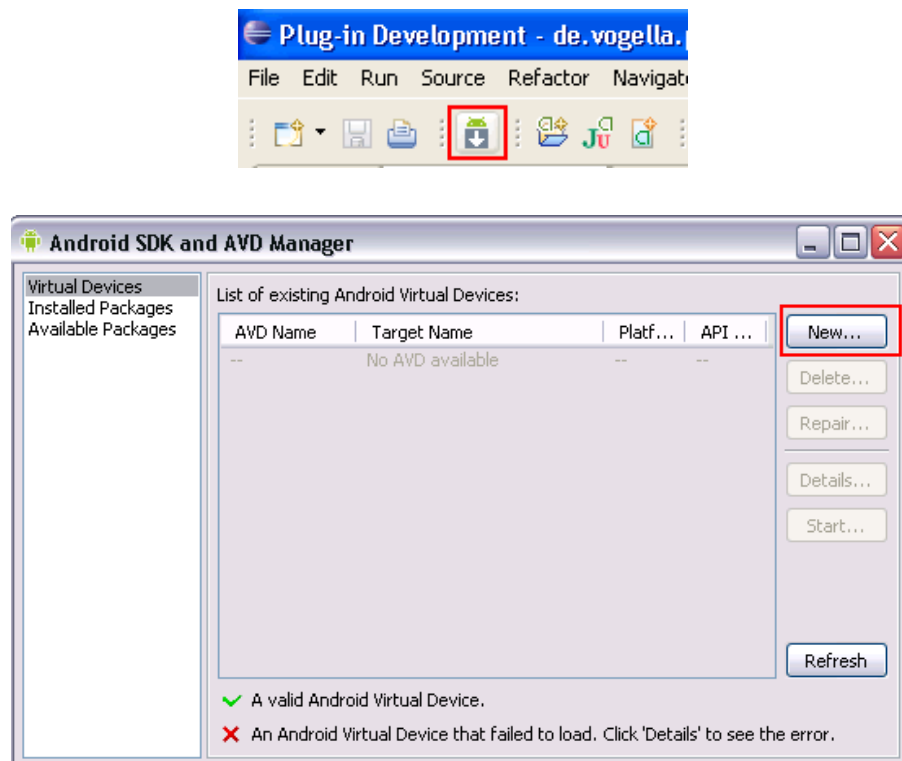
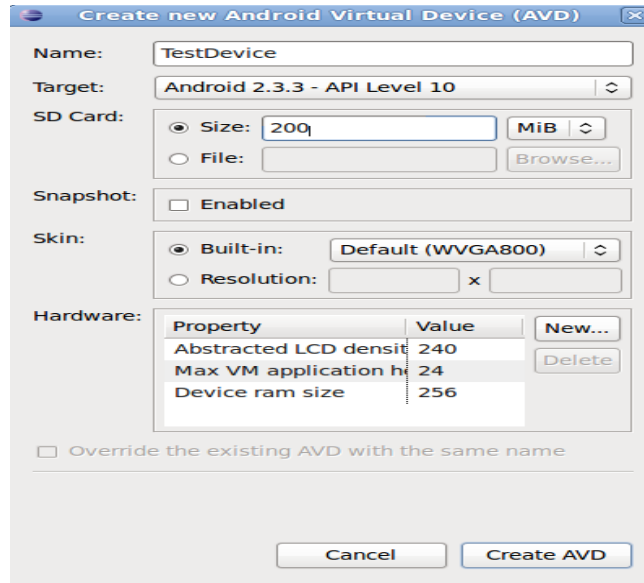


Figure 5.9: Android Virtual Devices**Figure 5.10: New AVD**

Press "Create AVD" as shown in figure 5.10. This will create the device and display it under the "Virtual devices". To test if your setup is correct, select your device and press "Start".

CHAPTER 6

IMPLEMENTATION

In the design phase the architecture is established. This phase starts with the requirement document delivered by the requirement phase and maps the requirement into the architecture the architecture defines the components, there interfaces and behaviours. The deliverable design document is the architecture. The design document describes a plan to implement the requirements.

6.1 System Analysis

Analysis is the process of understanding the problem and its domain. The main objectives of analysis is to capture a complete, ambiguous and consistent picture of the requirements of the system and what the system must do to satisfy the user needs and requirements. The principle objective of the system analysis phase is the specification of what the system is required to do. The system development data input and output forms and conventions

6.2 System Design

System design is an interactive process through which the requirements are translated into a “blue print” for constructing the software as shown in figure 6.1. The three characteristics that serve as a guide for the evaluation of a good design of software are as follows.

- The design must implement all the explicit requirements contained in the analysis mode, and it must accommodate all of the implicit requirements desired by the user.
- The design must be reliable, and act as an understandable guide for those, who generate code and test and subsequently maintain the software.
- The design should provide a complete picture of the software, addressing the data, functional and behavioural domains from an implementing perspective.

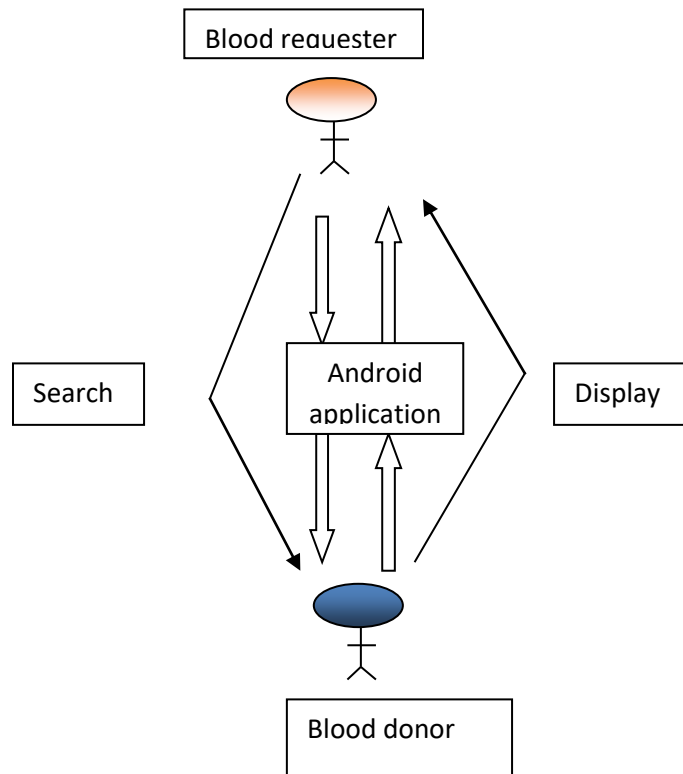


Figure 6.1: System Architecture

Modules:

1. **Donor:** donor is the one who is willing to donate the blood in this there are two modules
 - **Donor registration:** initially the donor needs to get register for the application by filling the details.
 - **Donor login:** to edit the details the donor can login to the app by using id and password and edit.

```
package com.example.blood_group_app;
```

```
import java.util.ArrayList;
```

```
import java.util.HashMap;
```



```
import android.annotation.SuppressLint;

import android.app.ActionBar;

import android.app.Activity;

import android.app.AlertDialog;

import android.content.DialogInterface;

import android.content.Intent;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.graphics.Color;

import android.graphics.drawable.ColorDrawable;

import android.os.Bundle;

import android.provider.BaseColumns;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.ListAdapter;

import android.widget.ListView;

import android.widget.Toast;

import static com.example.blood_group_app.Constants.FIRST_COLUMN;

import static com.example.blood_group_app.Constants.SECOND_COLUMN;

import static com.example.blood_group_app.Constants.THIRD_COLUMN;

import static com.example.blood_group_app.Constants.FOURTH_COLUMN;

public class DonarlistScreenActivity extends Activity {

    ListView listView1;
```

```
ArrayList<BeanDetails> ar_bean;

private ArrayList<HashMap> list;

BeanDetails bean = new BeanDetails();

DatabaseHelper helper;

SQLiteDatabase database;

Cursor cursor;

Activity act;

static String str_blood_gp;

String str_pincode;

private static final String fields[] = { BaseCol._ID, "username", "addr", "mobilen" };

ListAdapter adapter;

String str_uname;

String str_id;

@SuppressWarnings("NewApi")

@Override

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.donarlistscreen);

    ActionBar actionBar = getActionBar();

    actionBar.setBackgroundDrawable(new ColorDrawable(Color

        .parseColor("#000000")));

    listView1 = (ListView) findViewById(R.id.listView1);

    str_blood_gp = getIntent().getStringExtra("BloodGroup");
```

```
str_pincode = getIntent().getStringExtra("PinCode");

str_uname = getIntent().getStringExtra("username");

str_id = getIntent().getStringExtra("Id");

act = this;

helper = new DatabaseHelper(this);

ar_bean = new ArrayList<BeanDetails>();

list = new ArrayList<HashMap>();

try{

database = helper.getReadableDatabase();

ar_bean = getData();

if(ar_bean.size()>0){

for(int i=0 ; i<ar_bean.size();i++)

{ BeanDetails data = ar_bean.get(i);

HashMap temp = new HashMap();

temp.put(FIRST_COLUMN, data.Value1);

temp.put(SECOND_COLUMN, data.Value2);

temp.put(THIRD_COLUMN, data.Value3);

temp.put(FOURTH_COLUMN, data.Value4);

list.add(temp);}

for(int i = 0; i < list.size(); i++) {

HashMap temp =list.get(i);}

adapter= new ListviewAdapterDonarList(act,list);

listView1.setAdapter(adapter);} }
```

```
catch(Exception e){

    Toast.makeText(getApplicationContext(),"error"+e.getMessage(),

    Toast.LENGTH_SHORT).show();}}

public ArrayList<BeanDetails> getData(){

    try{

        String selection = "bloodgrp=? AND pinno=? AND donar=?";

        String[] selectionArgs = { str_blood_gp, str_pincode, "YES" };

        cursor=database.query(DatabaseHelper.User_details_TABLE_NAME,

                                fields, selection, selectionArgs, null, null, null);

        cursor.moveToFirst();

        do{

            bean.Value1 = cursor.getString(cursor.getColumnIndex("_id"));

            bean.Value2=cursor.getString(cursor.getColumnIndex("username"));

            bean.Value3 = cursor.getString(cursor.getColumnIndex("address"));

            bean.Value4 = cursor.getString(cursor.getColumnIndex("mobilen"));

            ar_bean.add(new BeanDetails(bean.Value1,bean.Value2, bean.Value3, bean.Value4));

        }while(cursor.moveToNext());

        return ar_bean;}

    catch(Exception e){

        Toast.makeText(getApplicationContext(),"SorryNoDataExists",

        Toast.LENGTH_SHORT).show();

        return null;}}

@Override
```

```
protected void onStop() {  
    super.onStop();  
    finish();}  
  
@Override  
  
public void onBackPressed() {  
    super.onBackPressed();  
  
    finish();  
  
    IntentMainScreen=new Intent(getApplicationContext(),SearchdonarActivity.class);  
  
    MainScreen.putExtra("username", str_uname);  
  
    MainScreen.putExtra("Id", str_id);  
  
    startActivity(MainScreen);}}
```

2. Receiver: are the one who is in search for the blood even hear we divide into two modules they are

- **Receiver registration:** initially the receiver needs to get register for the application by filling the details.
- **Receiver login:** to search for the donor or blood bank details the receiver has to login to the app by using id and perform various operations.

```
package com.example.blood_group_app;  
  
import android.app.Activity;  
  
import android.content.ContentValues;  
  
import android.content.Intent;  
  
import android.database.sqlite.SQLiteDatabase;  
  
import android.os.Bundle;
```

```
import android.view.View;

import android.view.View.OnClickListener;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Spinner;

import android.widget.Toast;

public class RegistrationScreenActivity extends Activity {

    EditText et_firstname;

    EditText et_lastname;

    EditText et_username;

    EditText et_password;

    EditText et_confirmpwd;

    EditText et_mobilenno;

    EditText et_address;

    EditText et_pinno;

    Spinner spn_gender;

    Spinner spn_bloodgrp;

    Button btn_register;

    Button btn_existinguser;

    DatabaseHelper helper;
```

```
SQLiteDatabase database;

String[] str_gender = { "Male", "Female" };

String[] str_blood_group = { "A+ve", "A-ve", "B+ve", "B-ve", "O+ve", "O-ve" };

String gender="Male";

String bloodgrp="A+ve";

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    setContentView(R.layout.registrationscreen);

    et_firstname = (EditText) findViewById(R.id.et_firstname);
    et_lastname = (EditText) findViewById(R.id.et_lastname);
    et_username = (EditText) findViewById(R.id.et_username);
    et_password = (EditText) findViewById(R.id.et_password);
    et_confirmpwd = (EditText) findViewById(R.id.et_confirmpwd);
    et_mobilenno = (EditText) findViewById(R.id.et_mobilenno);
    et_address = (EditText) findViewById(R.id.et_address);
    et_pinno = (EditText) findViewById(R.id.et_pinno);
    spn_gender = (Spinner) findViewById(R.id.spn_gender);
    spn_bloodgrp = (Spinner) findViewById(R.id.spn_bloodgrp);
    btn_register = (Button) findViewById(R.id.btn_register);

    helper = new DatabaseHelper(this);

    ArrayAdapter aa = new ArrayAdapter(this,
    android.R.layout.simple_spinner_item, str_gender);

    aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

```
spn_gender.setAdapter(aa);

spn_gender.setOnItemClickListener(new OnItemSelectedListener(){

@Override

public void onItemClick(AdapterView<?> parent, View view,int position, long id) {

gender = parent.getItemAtPosition(position).toString().trim();}

@Override

public void onNothingSelected(AdapterView<?> parent) {

gender = "Male";});

ArrayAdapter aa1 = new ArrayAdapter(this,

android.R.layout.simple_spinner_item, str_blood_group);

aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

spn_bloodgrp.setAdapter(aa1);

spn_bloodgrp.setOnItemClickListener(new OnItemSelectedListener(){

@Override

public void onItemClick(AdapterView<?> parent, View view,

int position, long id) {

bloodgrp = parent.getItemAtPosition(position).toString().trim();}

@Override

public void onNothingSelected(AdapterView<?> parent) {

bloodgrp = "A+ve";});

btn_register.setOnClickListener(new OnClickListener() {

private Intent loginScreen;
```



```
@Override

public void onClick(View v) {

    if (et_firstname.getText().toString().trim().equalsIgnoreCase(""))

        et_firstname.getText().toString().trim().length() == 0) {

        Toast.makeText(getApplicationContext(), "enter firstname",

        Toast.LENGTH_LONG).show();

    } else if (et_lastname.getText().toString().trim().equalsIgnoreCase(""))

        et_lastname.getText().toString().trim().length() == 0) {

        Toast.makeText(getApplicationContext(), "enter lastname",

        Toast.LENGTH_LONG).show();

    } else if (et_username.getText().toString().trim().equalsIgnoreCase(""))

        et_username.getText().toString().trim().length() == 0) {

        Toast.makeText(getApplicationContext(), "enter username",

        Toast.LENGTH_LONG).show();

    } else if (et_password.getText().toString().trim().equalsIgnoreCase(""))

        et_password.getText().toString().trim().length() == 0) {

        Toast.makeText(getApplicationContext(), "enter password",

        Toast.LENGTH_LONG).show();

    } else if (et_confirmpwd.getText().toString().trim()

        .equalsIgnoreCase(""))

        et_confirmpwd.getText().toString().trim().length() == 0) {

        Toast.makeText(getApplicationContext(), "enter confirmpwd",

        Toast.LENGTH_LONG).show();
```

```
} else if (et_mobilenno.getText().toString().trim()

.equalsIgnoreCase(""))

et_mobilenno.getText().toString().trim().length() == 0) {

Toast.makeText(getApplicationContext(), "enter mobilenno",

Toast.LENGTH_LONG).show();

} else if (et_address.getText().toString().trim().equalsIgnoreCase(""))

et_address.getText().toString().trim().length() == 0) {

Toast.makeText(getApplicationContext(), "enter address",

Toast.LENGTH_LONG).show();

} else if (et_pinno.getText().toString().trim().equalsIgnoreCase(""))

et_pinno.getText().toString().trim().length() == 0) {

Toast.makeText(getApplicationContext(), "enter pinno",

Toast.LENGTH_LONG).show();}

else{

database = helper.getWritableDatabase();

ContentValues values = new ContentValues();

values.put("firstname", et_firstname.getText().toString().trim());

values.put("lastname", et_lastname.getText().toString().trim());

values.put("username", et_username.getText().toString().trim());

values.put("pwd", et_password.getText().toString().trim());

values.put("mobilenno", et_mobilenno.getText().toString().trim());

values.put("address", et_address.getText().toString().trim());

values.put("pinno", et_pinno.getText().toString().trim());
```

```
values.put("gender", gender);

values.put("bloodgrp", bloodgrp);

values.put("donar", "NO");

try{

long i = database.insert(DatabaseHelper.User_details_TABLE_NAME, null, values);

if(i>0){

Toast.makeText(getApplicationContext(), "DataInsertedSuccessfully",

Toast.LENGTH_SHORT ). show();

Intent LoginScreen = new Intent(getApplicationContext(), LoginScreenActivity.class);

startActivity(LoginScreen);}

else{

Toast.makeText(getApplicationContext(), "Data Insertion failed" ,

Toast.LENGTH_SHORT).show();}

}catch(Exception e){

Toast.makeText(getApplicationContext(), "Inserting Data failed: " + e.getMessage() ,

Toast.LENGTH_SHORT).show();} } } });

/*btn_existinguser.setOnClickListener(new OnClickListener() {

public void onClick(View v) {

}});*/

}

@Override

protected void onStop() {

super.onStop();
```

```
finish();

}

@Override

public void onBackPressed() {

    super.onBackPressed();

    finish();

    Intent LoginScreen = new Intent(getApplicationContext(), LoginScreenActivity.class);

        startActivity(LoginScreen);}}
```

3. **Search based on blood group:** receiver can search the blood based on blood group.

```
package com.example.blood_group_app;

import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

public class SearchdonarActivity extends Activity {

    Spinner spn_bloodgrp;
    EditText et_pinno;
    Button btn_search;

    String[] str_blood_group = { "A+ve", "A-ve", "B+ve", "B-ve", "O+ve", "O-ve" };
    String bloodgrp="A+ve";
```

```
String str_username;
String str_id;
protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);
    setContentView(R.layout.searchdonar);

    spn_bloodgrp = (Spinner)findViewById(R.id.spn_bloodgrp);

    et_pinno = (EditText)findViewById(R.id.et_pinno);
    btn_search = (Button)findViewById(R.id.btn_search);
    str_username = getIntent().getStringExtra("username");
    str_id = getIntent().getStringExtra("Id");
    ArrayAdapter aa = new ArrayAdapter(this,
        android.R.layout.simple_spinner_item, str_blood_group);
    aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    spn_bloodgrp.setAdapter(aa);
    spn_bloodgrp.setOnItemSelectedListener(new OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view,
            int position, long id) {
            bloodgrp = parent.getItemAtPosition(position).toString().trim();
        }
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            bloodgrp = "A+ve";
        }
    });
    btn_search.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            try {
                if(et_pinno.getText().toString().trim().equalsIgnoreCase("")) ||
                et_pinno.getText().toString().trim().length() == 0 {
                    Toast.makeText(getApplicationContext(), "enter pinno", Toast.LENGTH_LONG).show();
                }
            }
        }
    });
}
```

```

Intent donarList = new Intent(getApplicationContext(), DonarlistScreenActivity.class);
donarList.putExtra("BloodGroup", bloodgrp);
donarList.putExtra("PinCode", et_pinno.getText().toString().trim());
donarList.putExtra("username", str_uname);
donarList.putExtra("Id", str_id);
startActivity(donarList);}}
catch(Exception e){
Toast.makeText(getApplicationContext(),"erroris:"+e.getMessage(),
Toast.LENGTH_LONG).show();}}});}

@Override
protected void onStop() {
super.onStop();
finish();}

@Override
public void onBackPressed() {
super.onBackPressed();
finish();

Intent MainScreen = new Intent(getApplicationContext(),SearchScreenActivity.class);
MainScreen.putExtra("username", str_uname);
MainScreen.putExtra("Id", str_id);
startActivity(MainScreen);}}

```

4. **Search based on pin code:** the receiver can also search for the donor based on pin code.
5. **Call/ sms to donor:** the receiver can make a call or can send a message to the donor.
6. **Blood bank info:** the receiver will be able to get the information regarding the blood bank.

```

package com.example.blood_group_app;
import java.util.ArrayList;
import java.util.HashMap;
import android.annotation.SuppressLint;

```

```
import android.app.ActionBar;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import android.provider.BaseColumns;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import static com.example.blood_group_app.Constants.FIRST_COLUMN;
import static com.example.blood_group_app.Constants.SECOND_COLUMN;
import static com.example.blood_group_app.Constants.THIRD_COLUMN;
import static com.example.blood_group_app.Constants.FOURTH_COLUMN;
import static com.example.blood_group_app.Constants.FIFTH_COLUMN;
import static com.example.blood_group_app.Constants.SIXTH_COLUMN;

public class BloodBankListScreenActivity extends Activity {
    ListView listView1;
    ArrayList<BeanDetails> ar_bean;
    private ArrayList<HashMap> list;
    BeanDetails bean = new BeanDetails();
    DatabaseHelper helper;
```

```
SQLiteDatabase database;
Cursor cursor;
Activity act;
private static final String fields[] = { BaseColumns._ID, "name","address", "mobilen",
"latitude", "longitude" };
ListAdapter adapter;
String str_username;
String str_id;
@SuppressLint("NewApi")
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.bloodbanklistscreen);
    ActionBar actionBar = getSupportActionBar();
    actionBar.setBackgroundDrawable(new ColorDrawable(Color.parseColor("#000000")));
    listView1 = (ListView) findViewById(R.id.listView1);
    str_username = getIntent().getStringExtra("username");
    str_id = getIntent().getStringExtra("Id");
    act = this;
    helper = new DatabaseHelper(this);
    ar_bean = new ArrayList<BeanDetails>();
    list = new ArrayList<HashMap>();
    try{
        database = helper.getReadableDatabase();
        ar_bean = getData();
        if(ar_bean.size()>0){
            for(int i=0 ; i<ar_bean.size();i++){
                BeanDetails data = ar_bean.get(i);
                HashMap temp = new HashMap();
                temp.put(FIRST_COLUMN, data.Value1);
```



```
temp.put(SECOND_COLUMN, data.Value2);
temp.put(THIRD_COLUMN, data.Value3);
temp.put(FOURTH_COLUMN, data.Value4);
temp.put(FIFTH_COLUMN, data.Value5);
temp.put(SIXTH_COLUMN, data.Value6);
list.add(temp);}

for(int i = 0; i < list.size(); i++) {
    HashMap temp =list.get(i);}
adapter= new ListviewAdapterBloodBankList(act,list);
listView1.setAdapter(adapter);} }
catch(Exception e){
    Toast.makeText(getApplicationContext(),"error"+e.getMessage(),
    Toast.LENGTH_SHORT).show();} }
public ArrayList<BeanDetails> getData(){
    try{
        cursor = database.query(DatabaseHelper.BloodBank_details_TABLE_NAME,
                                fields, null, null, null, null, null);

        cursor.moveToFirst();
        do{
            bean.Value1 = cursor.getString(cursor.getColumnIndex("_id"));
            bean.Value2 = cursor.getString(cursor.getColumnIndex("name"));
            bean.Value3 = cursor.getString(cursor.getColumnIndex("address"));
            bean.Value4 = cursor.getString(cursor.getColumnIndex("mobilenumber"));
            bean.Value5 = cursor.getString(cursor.getColumnIndex("latitude"));
            bean.Value6 = cursor.getString(cursor.getColumnIndex("longitude"));
            ar_bean.add(new BeanDetails(bean.Value1, bean.Value2, bean.Value3, bean.Value4,
            bean.Value5, bean.Value6));
        }while(cursor.moveToNext());
        return ar_bean;}
```

```

catch(Exception e){
    Toast.makeText(getApplicationContext(),"SorryNoDataExists",
    Toast.LENGTH_SHORT).show();
    return null;}}
@Override
public void onBackPressed() {
    super.onBackPressed();
    finish();
    Intent mainScreen = new Intent(getApplicationContext(),SearchScreenActivity.class);
    mainScreen.putExtra("username", str_uname);
    mainScreen.putExtra("Id", str_id);
    startActivity(mainScreen);}}

```

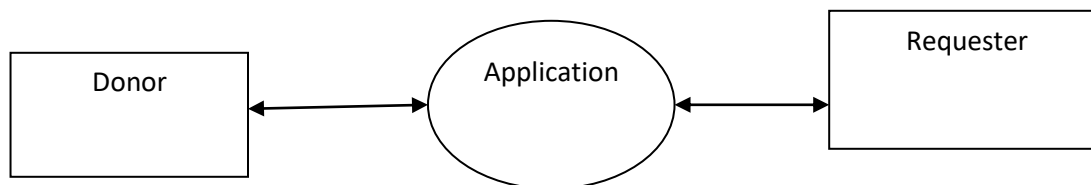
7. Blood donor list: the receiver will be able to see the list of blood donor.

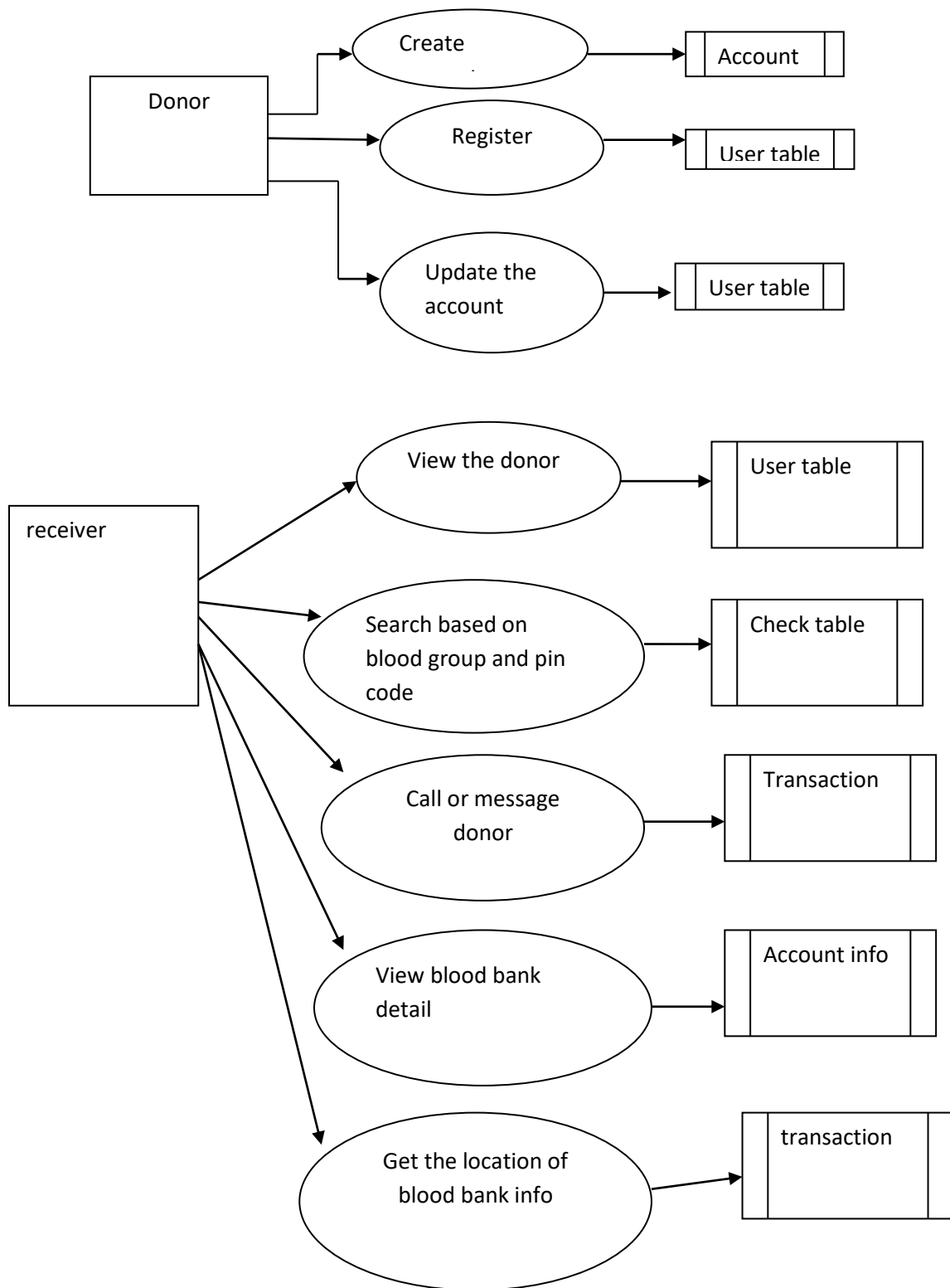
8. Exit application: once the operation is performed the donor or receiver will terminate the application.

6.3 Data flow diagram

Data flow diagrams are the basic building blocks that define the flow of data in a system to the particular destination and difference in the flow when any transformation happens as shown in figure 6.2 and 6.3. It makes whole procedure like a good document and makes simpler and easy to understand for both programmers and non-programmers by dividing into the sub process.

The data flow diagrams start from source and ends at the destination level i.e., it decomposes from high level to lower level and hence, it indicates the data flow for one way but not for loop structures and it doesn't indicate the time factors.



**Figure 6.2: Data Flow Diagram**

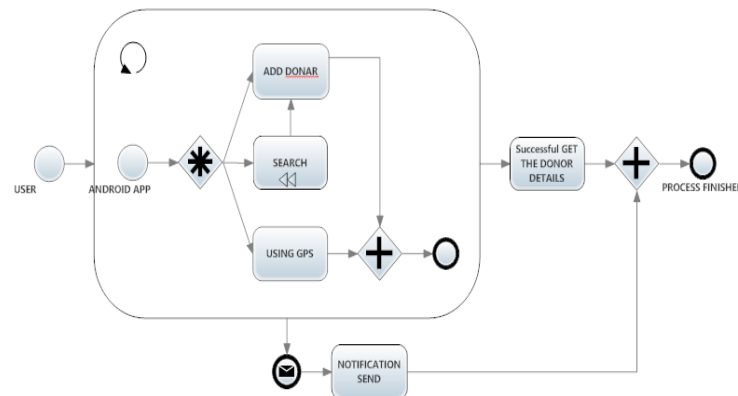


Figure 6.3: Overall Data Flow Diagram

CHAPTER 7

SYSTEM STUDY

FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well

within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 8

INPUT AND OUTPUT DESIGN

8.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.

- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

8.2 OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- ❖ Convey information about past activities, current status or projections of the
- ❖ Future.
- ❖ Signal important events, opportunities, problems, or warnings.
- ❖ Trigger an action.
- ❖ Confirm an action.

CHAPTER 9

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

9.1 TYPES OF TESTS

- **Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the

application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

- **Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

- **Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

- **System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

- **White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

- **Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.2 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

6.3 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

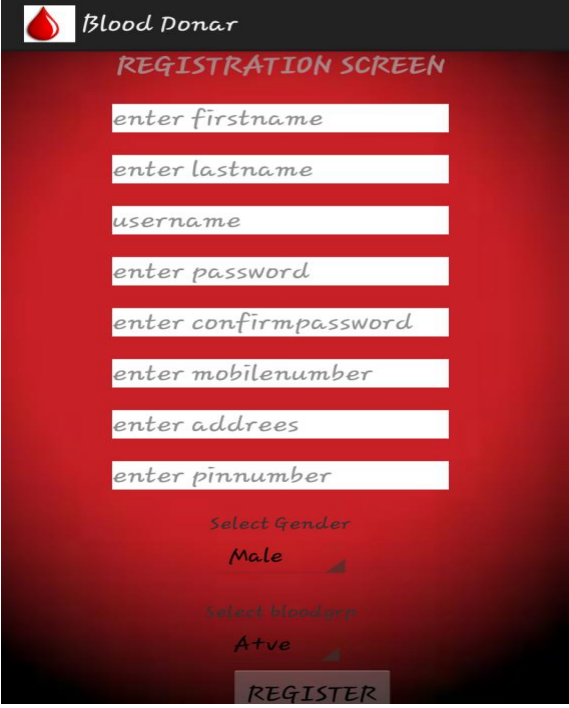
6.4 Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

CHAPTER 10

RESULTS



The screenshot shows the 'REGISTRATION SCREEN' of the 'Blood Donor' application. It features a dark red background with a blood drop icon in the top left corner. The screen contains several input fields for registration: 'enter firstname', 'enter lastname', 'username', 'enter password', 'enter confirmpassword', 'enter mobilenumber', 'enter addrees', and 'enter pinnumber'. Below these fields are two dropdown menus: 'Select Gender' with 'Male' selected, and 'Select bloodgrp' with 'A+ve' selected. A 'REGISTER' button is located at the bottom center.

Figure 10.1: Register Screen



The screenshot shows the 'MAINSCREEN' of the 'Blood Donor' application. It features a dark red background with a blood drop icon in the top left corner. The screen displays two large buttons: 'DONAR' and 'RECEIVER'.

Figure 10.2: Main Screen

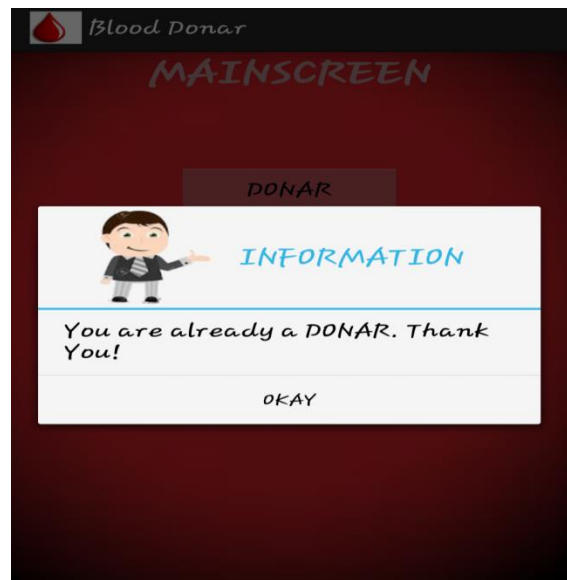


Figure 10.3: Information Screen

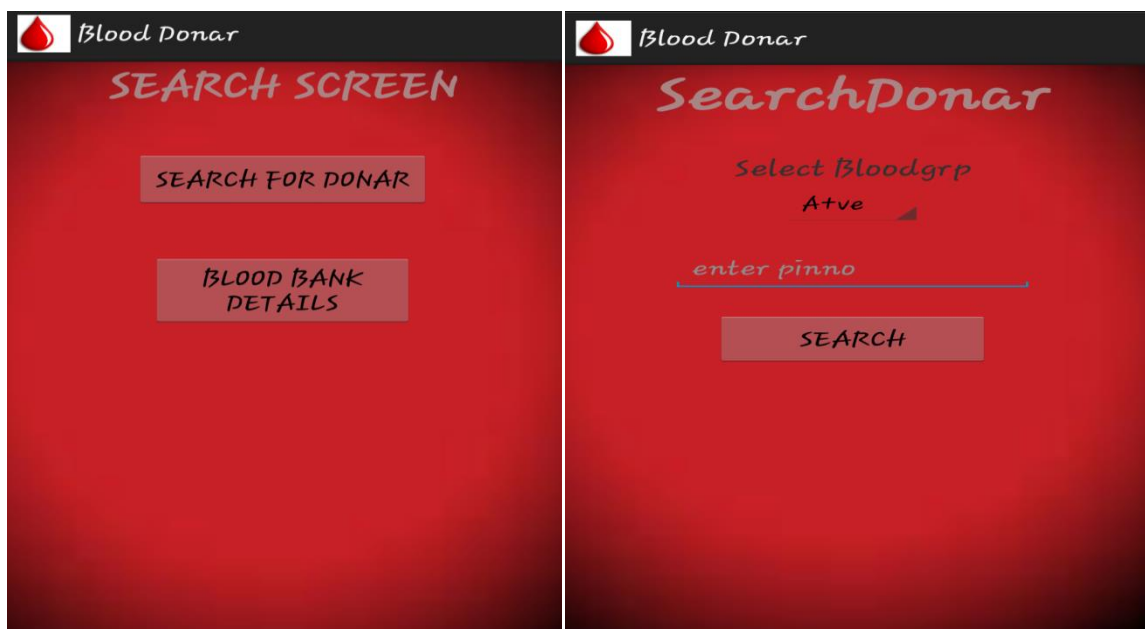


Figure 10.4: Search Screen

Figure 10.5: Search Donor



DONAR LIST				
ID	Name	Address	Mobileno	Action
	suji	anjanagar	8147508870	Action
	niki	anjanagar	9035759902	Action

Figure 10.6: Donor List

CONCLUSION

Initially mobile phones were developed only for voice communication but now days the scenario has changed, voice communication is just one aspect of a mobile phone. There are other aspects which are major focus of interest. one such major factors are GPS services. Both of these functionalities are already implemented but are only in the hands of manufacturers not in the hands of users because of proprietary issues, the system does not allow the user to access the mobile hardware directly. But now, after the release of android based open source mobile phone a user can access the hardware directly and design customized native applications to develop GPS enabled services.

In this study, presented a smart phone's application for the volunteer blood donor to increase the willingness and accessibility with the purpose of providing a continuous blood supply. This application helps receiver to get the blood as quick as possible when their stocks are insufficient. The application sends periodically actual location information of available blood

banks and the blood requests to the donors. In this way, it provides an uninterrupted communication between the receiver and volunteer donors.

REFERENCES

- [1] Turhan, S., Ozdemir, U, Yasar, A., 2012, KanBağısıVeTeminiBilgiSistemi: Türkiye _çin Mobil Modelleme, AkıllıSistemlerdeYeniliklerveUygulamalarıSempozyumu, Trabzon, Turkey, pp.192-198
- [2] Dutta, D.A. et al., 2011,Artificial Human Blood , Inventi Impact: Pharm Tech, Vol. 2015,No.1
- [3] Gillespie, T. W., &Hillyer, C. D. (2002). Blood donors and factors impacting the blood donation decision. Transfusion Medicine Reviews, 16(2), 115-130.
- [4] V. Bosnes, M. Aldrin, H. E. Heier, “Predicting Blood Donor Arrival.” , Transfusion, Cilt: 45, Sayı:2, 2014, s. 162-170.
- [5] Prcmasudha, B.G., et al., 2010, Application of Spatial Decision Support System to Blood Bank Information Systems, International Journal of Geoinformatics, Vol.6, No. 2, pp.51 – 58.

- [6] Sime, S.L. et al., 2015, Strengthening The Service Continuum Between Transfusion Providers and Suppliers: Enhancing the Blood Services Network., *Transfusion*, Vol.45, No.4, pp.206S-223S.
- [7] France, C. R., France, J. L., Wissel, M. E., Kowalsky, J. M., Bolinger, E. M., &Huckins, J. L. (2012).Enhancing blood donation intentions using multimedia donor education materials. *Transfusion*, 51(8).
- [8] Stanger, S. H., Yates, N., Wilding, R., & Cotton, S. (2012). Blood inventory management: hospital best practice. *Transfusion medicine reviews*, 26(2), 153-163.
- [9] Williamson, L. M., & Devine, D. V. (2013). Challenges in the management of the blood supply. *The Lancet*, 381(9880).
- [10] TürkiyeKanMerkezleriveTransfüzyonDerneği, UlusalKanveKanÜrünleriRehberi, Haziran 2013, www.kmtd.org.tr
- [11] S.Hablemitoglu, Y. Özkan, F. Yıldırım, “BirFedakârlıkÖrneğiOlarak “KanBağısı” ” ,Aileve Toplum, Ocak – Subat – Mart 2010, s.67 – 77
- [12] Nilsson Sojka, B., &Sojka, P. (2003). The blood-donation experience: perceived physical, psychological and social impact of blood donation on the donor. *VoxSanguinis*, 84(2), 120-128.
- [13] Erl, Thomas. *Soa: principles of service design*. Vol. 1. Upper Saddle River: Prentice Hall, 2010.
- [14] The Optimization of Blood Donor Information and Management System by Technopedia P. Priya¹, V. Saranya², S. Shabana³,Kavitha Subramani⁴ Department of Computer Science and Engineering, Panimalar Engineering college, Chennai, India^{1, 2, 3, 4}
- [15] MBB: A Life Saving Application Narendra Gupta¹, RamakantGawande² and Nikhil thengadi³ ^{1, 2, 3} Final Year, CSE Dept.,JDIET, Yavatmal, India.
- [16] AN ANDROID APPLICATION FOR VOLUNTEER BLOODDONORS by Sultan Turhan.
- [17] Arif. M. Sreevas. S. Nafseer. K. and Rahul. R.(2012), 'Automated online Blood bankdatabase', India Conference (INDICON),Annual IEEE, Print ISBN: 978-1-4673-2270-6, pp. 012 - 017.
- [18] Spyropoulos. B., Botsivaly. M., Tzavaras. A., andSpyropoulou, P (2012), 'Towards digital blood-banking', ITU-TKaleidoscope: Innovations for Digital Inclusions, .K-IDIE-ISBN:978-92-61-12891-3, Print ISBN: 978-92-61-12891-3, pp.I- 8.

[19] A Survey Paper on E-Blood Bank and an Idea to use on Smartphone TusharPandit, SatishNiloor and A.S. Shinde, Dept. of I.T SinhgadAcademy of Engineering, Pune, India