**­**

**ABSTRACT**

Help Line is a voluntary and non-governmental organization. It maintains Online library of blood donors in India. Sometimes Doctors and Blood bank project have to face the difficulty in finding the blood group Donors at right time. Help Line has attempted to provide the answer by taking upon itself the task of collecting Blood bank project nationwide for the cause and care of people in need.

At any point of time the people who are in need can reach the donors through our search facility. By mobilizing people and organization who desire to make a difference in the lives of people in need. On the basis of humanity, everyone is welcome to register as a blood donor.

The Blood Alliance (TBA) is an app-based system that is designed to store, process, retrieve and analyse information concerned with the administrative and inventory management within a blood bank. This project aims at maintaining all the information pertaining to blood donors, different blood groups available in each blood bank and help them manage in a better way. Aim is to provide transparency in this field, make the process of obtaining blood from a blood bank hassle free and corruption free and make the system of blood bank management effective.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TOPIC** | **PAGE NO** |

|  |  |  |
| --- | --- | --- |
| **1** | **Introduction** | **7-10** |
| 1.1 | Project Description | 8 |
| 1.2 | Overview of the current state of technology | 8 |
| 1.3 | Project Objectives | 8 |
| 1.4 | Scope and Limitations of the Project | 9 |
| 1.5 | Significance of the Project | 9 |

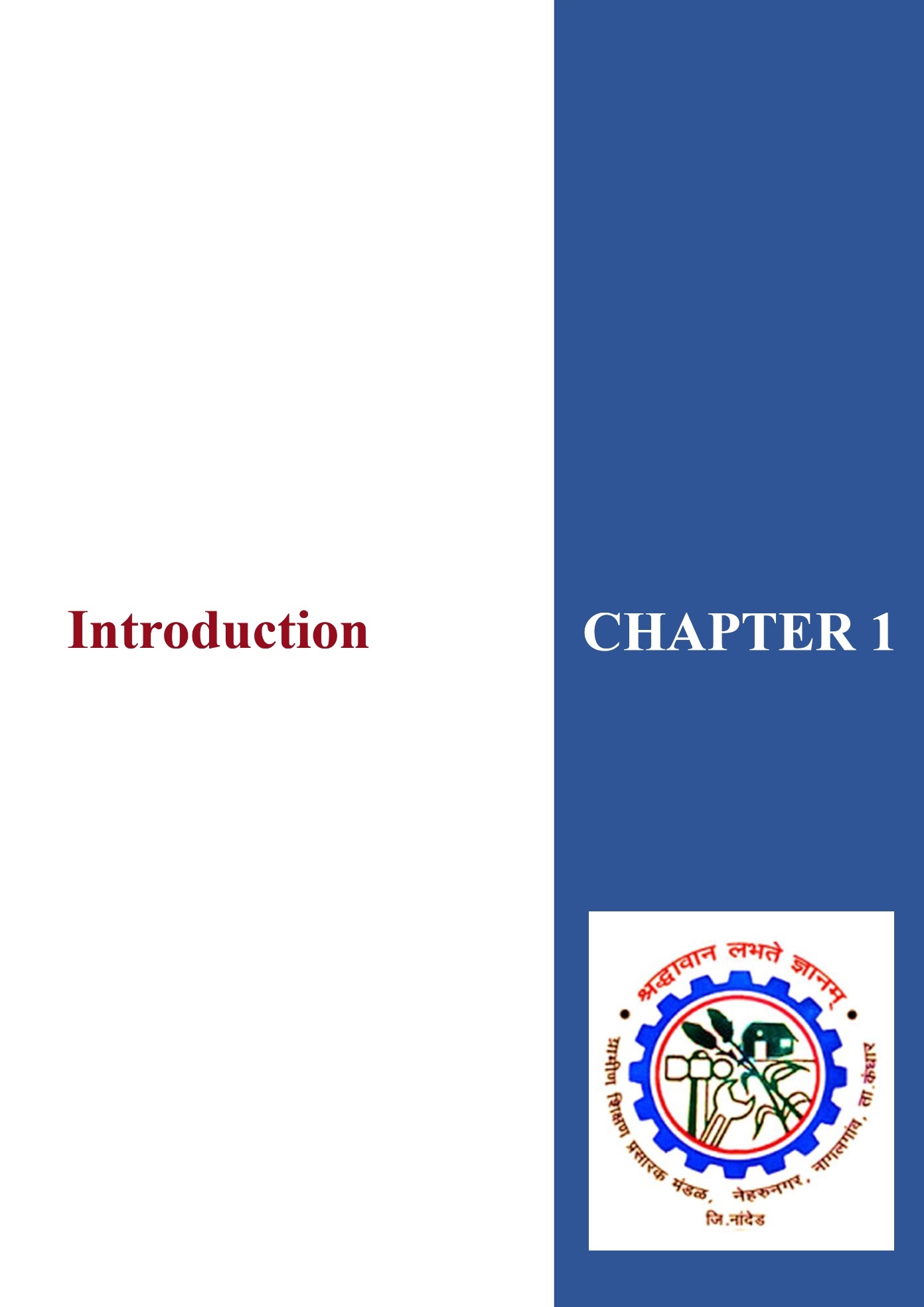
|  |  |  |
| --- | --- | --- |
| **2** | **Literature Survey** | **11-14** |
| 2.1 | Market Survey | 12 |
| 2.2 | Requirements | 12 |
| 2.3 | Literature Review | 13 |

|  |  |  |
| --- | --- | --- |
| **3** | **Overview of Android** | **15-20** |
| 3.1 | History | 16 |
| 3.2 | Application | 17 |
| 3.3 | Development | 17 |
| 3.4 | Testing | 18 |
| 3.4.1 | Testing Fundamentals | 18 |
| 2.5 | Features & Specifications | 20 |

|  |  |  |
| --- | --- | --- |
| **4** | **Requirement Analysis and Design** | **21-28** |
| 4.1 | Software Engineering Paradigm Used | 23 |
| 4.2 | The Software Process | 24 |
| 4.3 | The Linear Sequential Model/ Software Development Life Cycle | 25 |
| 4.4 | Software Development Process Model | 25 |

|  |  |  |
| --- | --- | --- |
| **5** **4** | **Design and Documentation** | **29-35** |
|  |  |  |
| **6** | **Software Development Environment** | **36-43** |
| 6.1 | Software Environment | 37 |
| 6.1.1 | Android Architecture | 37 |
| 6.1.2 | Application Framework | 38 |
| 6.1.3 | Libraries | 39 |
| 6.1.4 | Android Runtime | 40 |
| 6.1.5 | Linux Kernel | 40 |
| 6.1.6 | Android SDK | 40 |
| 6.2 | System Requirements | 41 |
| 6.3 | Eclipse IDE | 41 |
| 6.4 | Hardware Requirements for Android SDK | 41 |
| 6.5 | Version History | 42 |

|  |  |  |
| --- | --- | --- |
| **7** | **Testing and Result** | **44-50** |
| 7.1 | System Testing | 45 |
| 7.2 | Types of Tests | 45 |
| 7.2.1 | Unit Testing | 45 |
| 7.2.2 | Integration Testing | 46 |
| 7.2.3 | Functional Test | 46 |
| 7.2.4 | System Test | 46 |
| 7.2.5 | White Box Testing | 47 |
| 7.2.6 | Black Box Testing | 47 |
| 7.3 | Black Box Testing | 47 |
| 7.4 | Integration Testing | 47 |
| 7.5 | Acceptance Testing | 48 |
| 7.6 | Test Cases | 49 |
|  |  |  |
|  |  |  |
| **8** | **Coding** | **51-88** |
|  | |  |
| **9** | **Output Screen** | **89-104** |
|  | |  |
| **10** | **Advantages and Features** | **105-106** |
|  | |  |
| **11** | **Applications** | **107-108** |
|  | |  |
| **12** | **Conclusion** | **109-111** |



***Chapter 1***

## INTRODUCTION TO THE BLOOD ALLIANCE

### 1.1 Project Description

The number of persons who are in need of blood are increasing in large number day by day. In order to help people who are in need of blood, my Online Blood Bank can be used effectively for getting the details of blood donors having the same blood group and with in the same city. With the help of my Online Blood Bank people who are having the thought of donating blood gets registered in my Online Blood Bank giving his total details

### 1.2 Overview of the current state of technology

The net has become the critical medium powering a growing list of revenue generating business activities from e-commerce and e-supply The birth of Internet empowers today’s generation to maximize the use of technology.

In existing Blood Bank Android Project system, if someone needs the blood, then either the person has to go to the blood bank nearby to him or else have to buy the blood from the hospital but in case, if both the places don’t have the blood of the required group than finding blood of that group might be a tough task at the required time.

In existing Blood Bank Project system, wastage of time is done more and still there will be chances that you won’t get blood at the required time.

### 1.3 Project Objectives

In this part, the researchers discussed the general and specific objectives that were aimed to be applied by the researchers.

* + - **General Objective**

To design and develop a The Blood Alliance app

#### Specific Objectives

To develop and establish an efficient and effective way of communication between the blood donors and blood bank.

* The person’s time and work are reduced very much which prevails in the present system.
* To incorporate the following features to the system.
  + Registering the Donors.
  + Modification of Donor Information.
  + Searching a Donor.
  + Send request to blood bank.

### 1.4 Scope and Limitations of the Project

The people in need of blood can search for the donors by giving their blood group and city name. it saves time as he can search donors online without going anywhere. Using this system user can get blood in time and can save his relative or friend life. Our app work 24x7 so user can get information of blood donor any time. Blood donor can also get registered and save life of another person. The main benefit of this system is the information of available blood group. When blood is need in the operation then people have very less time to get the blood available so if he gets the information like who can give him blood in time in his city is lifesaving. And here our system work, whenever a person needs blood, he gets information of the person who has the same blood group he needs.

### 1.5 Significance of the Project

The main aim of developing this system is to provide blood to the people who are in need of blood. The number of persons who are in need of blood are increasing in large number day by day. Using this system user can search blood group available in the city and he can also get contact number of the donor who has the same blood group he needs. In order to help people who are in need of blood, this Online Blood Bank management system can be used effectively for getting the details of available blood groups and user can also get contact number of the blood donors having the same blood group and within the same city. So, if the blood group is not available in the blood bank user can request the donor to donate the blood to him and save someone life. Using this bank management system people can register himself or herself who want to donate blood. To register in the system, they have to enter their contact information like address mobile number etc.

This The Blood Alliance app is an online working platform so it is easily available to everyone. When a person wants to donate blood, he has to register to the system. Donor registration is very easy, to get register to the system he has to fill up registration form. After submitting the registration form, he can create username and password. Donor has to give on like blood group, contact details etc. donor can also change his account information when he wants using his username and password

• **Existing System:**

Using this blood bank app people can search blood group available which they are needed. They check it online using our “The Blood Alliance” app. If in case blood group is not available in blood bank, they can also get contact numbers of the persons who has the same blood group he is need. And he can request the person to donate the blood for saving

### • Proposed System

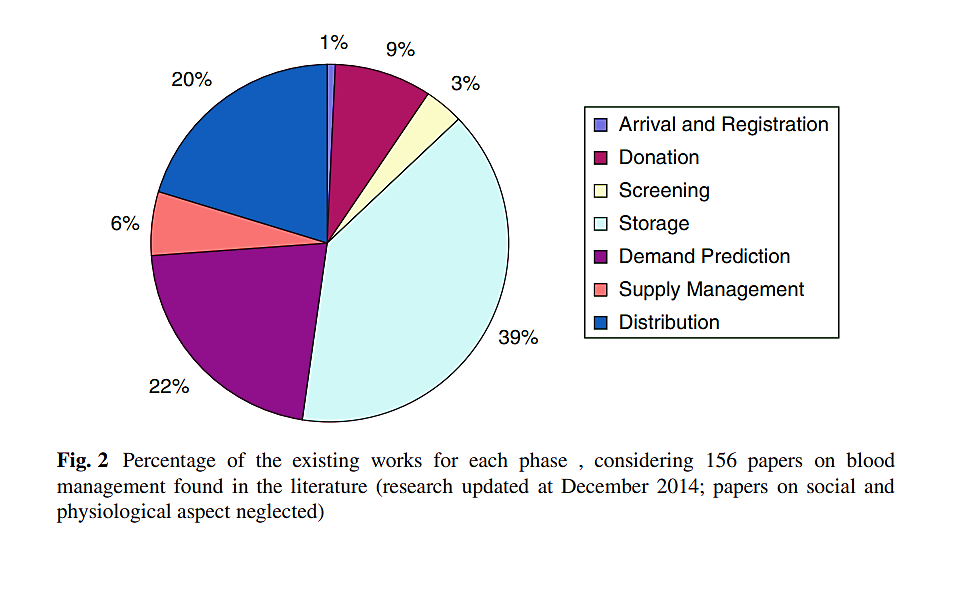
The number of persons who are in need of blood are increasing in large number day by day. In order to help people who are in need of blood, my Online Blood Bank can be used effectively for getting the details of blood donors having the same blood group and within the same city. With the help of my Online Blood Bank people who are having the thought of donating blood gets registered in my Online Blood Bank giving his total details.

***Chapter 2***

**LITERATURE SURVEY**

### 2.1 Market Survey:

Blood Banks are rarely using the digital Database systems which are now used by many computer systems and programs so we got to know that market really needs the computer-based Database sequences, so for that purpose we are implementing this project as per the market structure.



#### Figure 2.1: Market Analysis

### 2.2 Requirements:

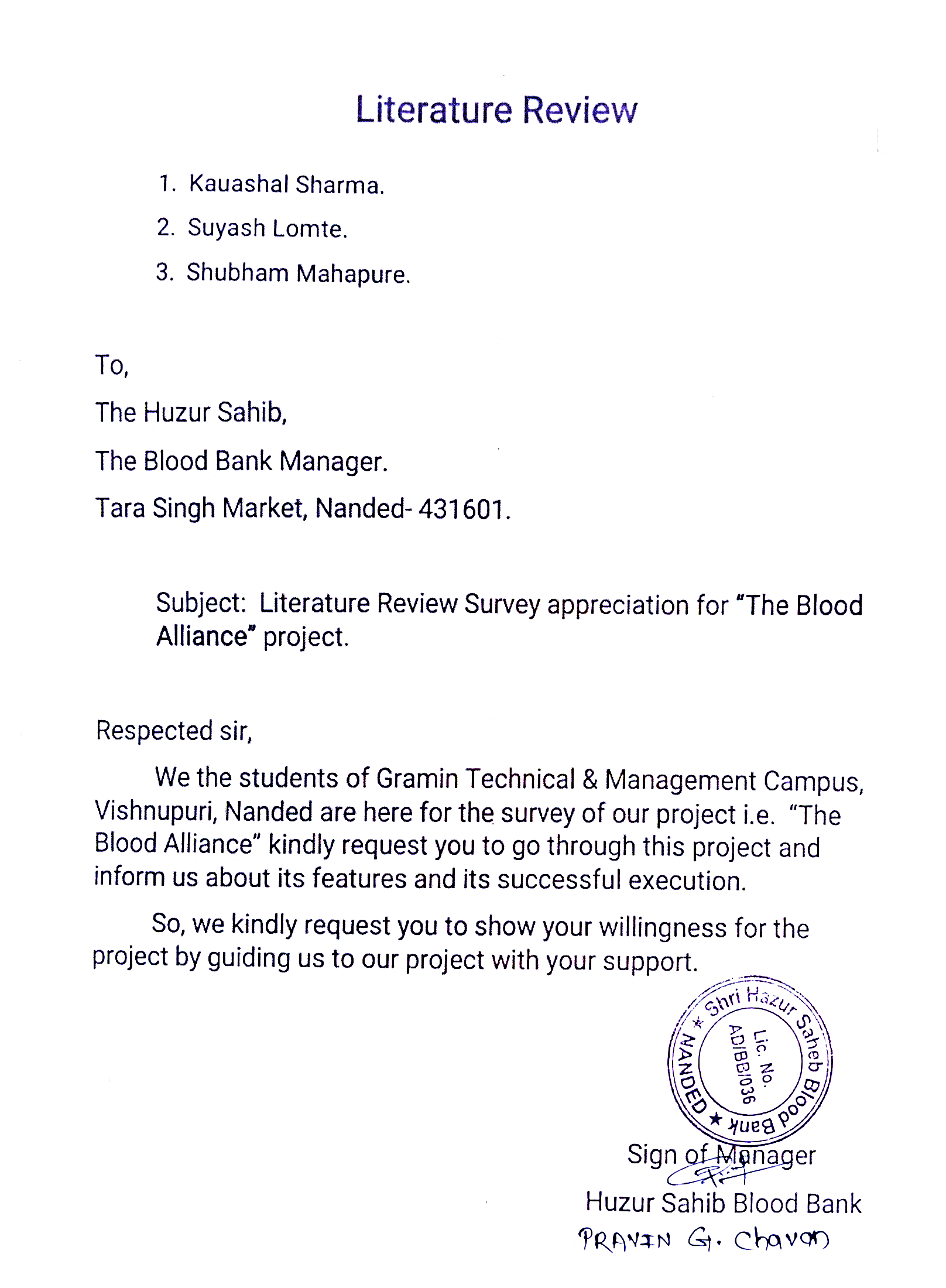
1. They needed a source which can minimize work load on the employees and staffs.
2. They needed a software-based program to store their all Data.
3. So according to their requirement, our project “The Blood Alliance” is the perfect solution for the use.



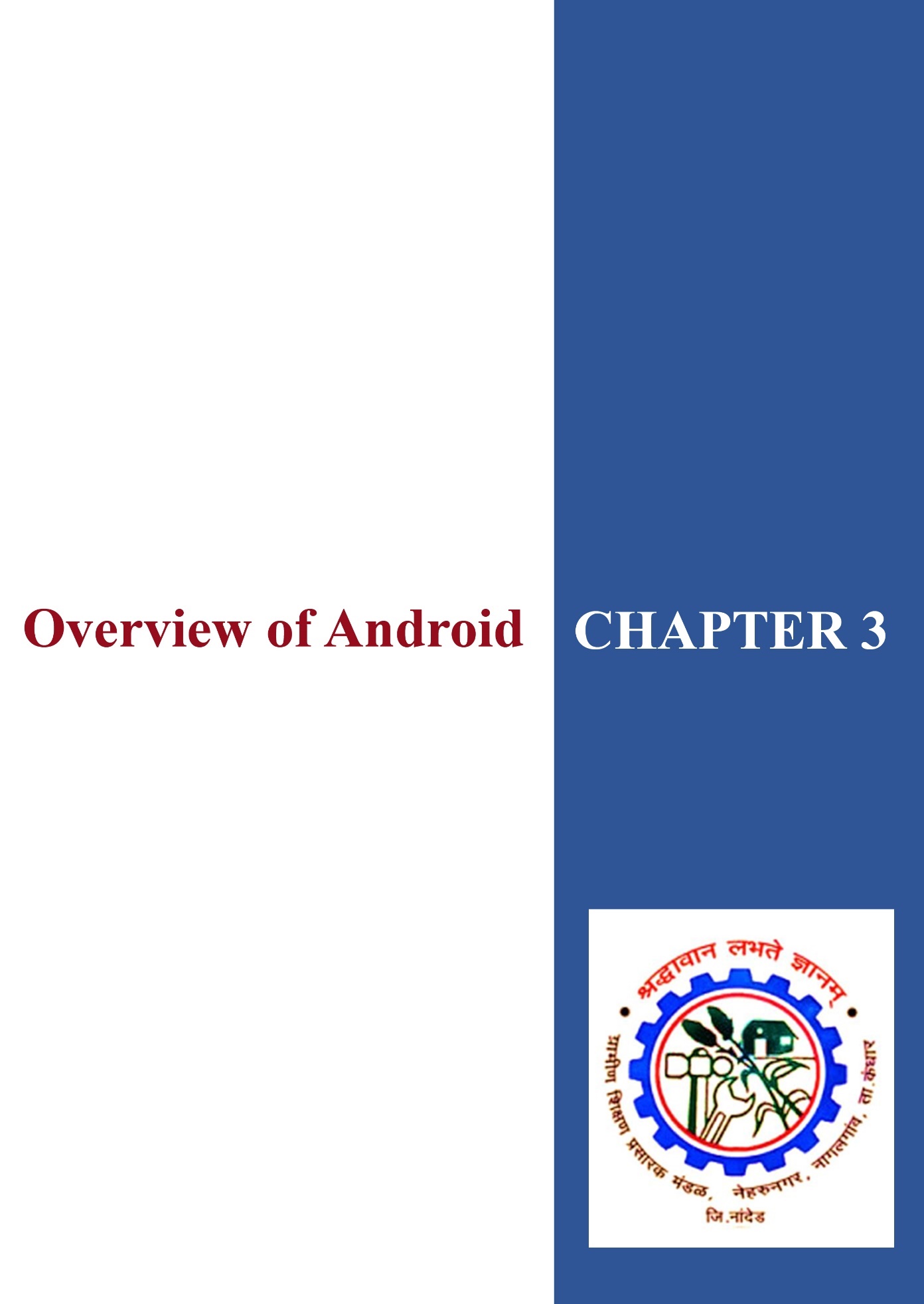
**Figure 2.2: Visit to the “Shri Hazur Sahib Blood Bank Nanded”.**

### 2.3 Literature Review:

So, in survey we got to know there is really need of newly high-quality software to manage all records for the management. Hence, they appreciated our project “The Blood Alliance”.



**Figure 2.3: Letter of Literature Review survey.**



***Chapter 3***

## OVERVIEW OF ANDROID

Android is a [mobile operating system](http://en.wikipedia.org/wiki/Mobile_operating_system) (OS) based on the [Linux kernel](http://en.wikipedia.org/wiki/Linux_kernel) and currently developed by [Google.](http://en.wikipedia.org/wiki/Google) With an interface based on [direct manipulation,](http://en.wikipedia.org/wiki/Direct_manipulation_interface) Android is designed primarily for [touch screen](http://en.wikipedia.org/wiki/Touchscreen) mobile devices such as [smart phone](http://en.wikipedia.org/wiki/Smartphone) and [tablet computers,](http://en.wikipedia.org/wiki/Tablet_computer) with specialized user interfaces for televisions ([Android TV)](http://en.wikipedia.org/wiki/Android_TV), cars ([Android Auto)](http://en.wikipedia.org/wiki/Android_Auto), and wrist watches ([Android Wear)](http://en.wikipedia.org/wiki/Android_Wear). The OS uses touch inputs that loosely correspond to real-world actions, like swiping, tapping, pinching, and reverse pinching to manipulate on-screen objects, and a [virtual keyboard.](http://en.wikipedia.org/wiki/Virtual_keyboard) Despite being primarily designed for touch screen input, it also has been used in [game consoles,](http://en.wikipedia.org/wiki/Video_game_console) [digital cameras,](http://en.wikipedia.org/wiki/Digital_camera) regular PCs (e.g. the HP Slate 21) and other electronics.

Android's [source code](http://en.wikipedia.org/wiki/Source_code) is released by Google under [open source](http://en.wikipedia.org/wiki/Open_source) licenses, although most Android devices ultimately ship with a combination of open source and proprietary software. Initially developed by Android, Inc., which Google backed financially and later bought in 2005, Android was unveiled in 2007 along with the founding of the [Open Handset Alliance](http://en.wikipedia.org/wiki/Open_Handset_Alliance) a consortium of hardware, software, and telecommunication companies devoted to advancing [open standards](http://en.wikipedia.org/wiki/Open_standard) for mobile devices.

Android is popular with technology companies which require a ready-made, low cost and customizable operating system for [high-tech](http://en.wikipedia.org/wiki/High-tech) devices. Android's open nature has encouraged a large community of developers and enthusiasts to use the open-source code as a foundation for community-driven projects, which add new features for advanced users or bring Android to devices which were officially, released running other operating systems. The operating system's success has made it a target for patent.

### 3.1 History

The early intentions of the company were to develop an advanced operating system for [digital cameras,](http://en.wikipedia.org/wiki/Digital_camera) when it was realized that the market for the devices was not large enough, and diverted their efforts to producing a Smartphone operating system to rival those of [Symbian](http://en.wikipedia.org/wiki/Symbian) and [Windows Mobile.](http://en.wikipedia.org/wiki/Windows_Mobile) Despite the 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,](http://en.wikipedia.org/wiki/Steve_Perlman) a close friend of Rubin, brought him $10,000 in cash in an envelope and refused a stake in the company.

### 3.2 Application

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

Android has a growing selection of third-party applications, which can be acquired by users by downloading and installing the application's APK file, or by downloading them using an application store program that allows users to install, update, and remove applications from their devices. Google Play Store is the primary application store installed on Android devices that comply with Google's compatibility requirements and license the Google Mobile Services software. Google Play Store allows users to browse, download and update applications published by Google and third-party developers.

### 3.3 Development

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

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

### 3.4 Testing

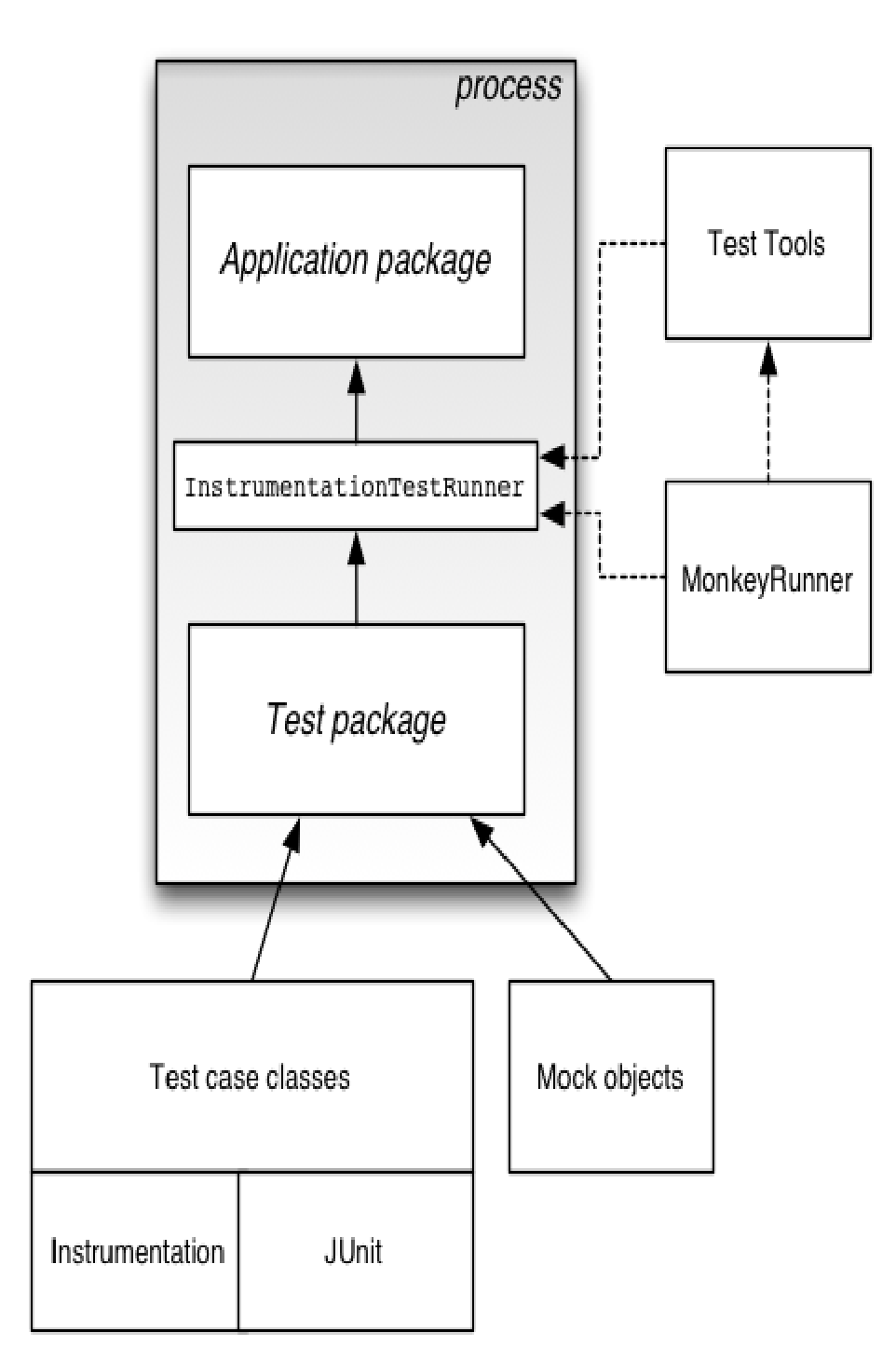
The Android framework includes an integrated testing framework that helps you test all aspects of your application and the SDK tools include tools for setting up and running test applications. Whether you are working in Eclipse with ADT or working from the command line, the SDK tools help you set up and run your tests within an emulator or the device you are targeting.

#### 3.4.1 Testing Fundamentals

The Android testing framework, an integral part of the development environment, provides architecture and powerful tools that help you test every aspect of your application at every level from unit to framework.

The testing framework has these key features:

* Android test suites are based on JUnit. You can use plain JUnit to test a class that doesn't call the Android API, or Android's JUnit extensions to test Android components. If you're new to Android testing, you can start with general-purpose test case classes such as Android Testcase and then go on to use more sophisticated classes.
* The Android JUnit extensions provide component-specific test case classes. These classes provide helper methods for creating mock objects and methods that help you control the lifecycle of a component.
* Test suites are contained in test packages that are similar to main application packages, so you don't need to learn a new set of tools or techniques for designing and building tests.
* The SDK tools for building and tests are available in Eclipse with ADT, and also in command-line form for use with other IDEs. These tools get information from the project of the application under test and use this information to automatically create the build files, manifest file, and directory structure for the test package.
* The SDK also provides [monkey runner,](http://developer.android.com/tools/help/monkeyrunner_concepts.html) an API for testing devices with Python programs, and [UI/Application Exerciser Monkey,](http://developer.android.com/tools/help/monkey.html) a command-line tool for stress testing UIs by sending pseudo-random events to a device. This document describes the fundamentals of the Android testing framework, including the structure of tests, the APIs that you use to develop tests, and the tools that you use to run tests and view results. The document assumes you have a basic knowledge of Android application programming and Unit testing methodology. The following diagram summarizes the testing framework:



**Figure 3.1: Testing Framework in Android**

### 3.5 Features & Specifications

Android is a powerful Operating System supporting a large number of applications in Smart Phones. These applications make life more comfortable and advanced for the users. Hardware that supports Android are mainly based on ARM architecture platform. Some of the current features and specifications of android are:

* Application Framework (It enables reuse and replacements of components)
* Dalvik Virtual machine (It is optimized for mobile device)
* Integrated Browser (It is based on the Open-Source Web kit Engine)
* Optimized Graphics
* SQLite
* Media support
* GSM technology
* Bluetooth, EDGE, Wi-Fi
* Camera, GPS, Compass etc.

Android comes with an Android market which is an online software store. It was developed by Google. It allows Android users to select, and download applications developed by third party developers and use them. There are around 2.0 lack+ games, application and widgets available on the market for users.

Android applications are written in java programming language. Android is available as open source for developers to develop applications which can be further used for selling in android market. There are around 200000 applications developed for android with over 3 billion+ downloads. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. For software development, Android provides Android SDK (Software development kit).



***Chapter 4***

## REQUIREMENT ANALYSIS AND DESIGN

### • Software requirements

* Windows XP/7/8/8.1 oAndroid Development Toolkit (ADT)

o Android Studio API 25

o Front End**:** Android xml o Back End: SQL

o HTTP

* Oracal 8

o VBScript

• **Hardware requirements** o Processor: I3

o Hard Disk: 500Gb

* RAM: 2Gb
* Android Device

Requirements engineering provides the appropriate mechanism for understanding what the customer wants, analyzing need, assessing feasibility, negotiating a reasonable solution, specifying the solution unambiguously, validating the specification, and managing the requirements as they are transformed into an operational system. It encompasses seven distinct tasks: inception, elicitation, elaboration, negotiation, specification, validation, and management. It is important to note that some of these tasks occur in parallel and all are adapted to the needs of the project.

Requirements analysis is the first stage in the systems engineering process and software development process. Requirements analysis in systems engineering and software engineering, encompasses those tasks that go into determining the needs or conditions to meet for a new or altered product, taking account of the possibly conflicting requirements of the various stakeholders, such as beneficiaries or users. Requirements analysis is critical to the success of a development project. Requirements must be actionable, measurable, testable, related to identified business needs or opportunities, and defined to a level of detail sufficient for system design. Requirements can be functional and non-functional.

#### 4.1 Software Engineering Paradigm Used

Like any other product, a software product completes a cycle from its inception to obsolescence/replacement/wearing out. The process of software development not only needs writing the program and maintains it, but also a detail study of the system to identify current requirements related to software development as well as to anticipate the future requirements. It also needs to meet the objective of low cost, and good quality along with minimum development time.

* Requirement analysis and specification for clear understanding of the problem.
* Software design for planning the solution of the problem.
* Coding (implementation) for writing program as per the suggested solution.
* Testing for verifying and validating the objective of the product.
* Operation and maintenance for use and to ensure its availability to users.

This application was also developed in phases for effective output. Each phase was given its due importance with respect to time and cost. The system development life

**Requirement Analysis and Specifications**

**Designing**

**Coding**

**Testing & validation**

**Operation and Maintenance**

**Figure 4.1: Life cycle of Project Management Information System.**

#### 4.2 The Software Process

A process is a collection of activities, actions, and tasks that are performed when some work product is to be created. An activity strives to achieve a broad objective. And is applied regardless of application domain, size of project, complexity of effort, or degree of rigor with which software engineering is to be applied. An action encompasses a set of tasks that produces a major work product. A task focuses on small but well-defined objective that produces tangible outcomes.

In the context of software engineering a process is not a rigid prescription for how to build computer software rather it is an adaptable approach that enables the people doing the work (the software team) to pick and choose the appropriate set of Work actions and tasks. The intent is always to deliver software in a timely manner and with sufficient quality to satisfy those who have sponsored its creation and those who will use it.

A process framework establishes the foundation for a complete software engineering process by identifying a small number of framework activities that are applicable to all software projects, regardless of their size or complexity. In addition, the process framework encompasses a set of umbrella activities that are applicable across the entire software process. A generic process framework for software engineering encompasses five activities:

* **Communication:** Before any technical work can commence, it is critically important to communicate and collaborate with the customer (and other stakeholders. The intent is to understand stakeholders, objectives for the project and to gather requirements that help define software features and functions.
* **Planning:** Any complicated journey can be simplified if a map exists. A software project is a complicated journey, and the planning activity creates a "map" that helps guide the team as it makes the journey. The map called a software project plan-defines the software engineering work by describing the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule.
* **Modelling:** Whether you're a landscaper, a bridge builder, an aeronautical engineer, a carpenter, or an architect, you work with models every day. You create a "sketch" of the thing so that you'll understand the big picture-what it will look like architecturally, how the constituent parts fit together, and many other characteristics. If required, you refine the sketch into greater and greater detail in an effort to better understand the problem and how you’re going to solve it. A software engineer does the same thing by creating models to better understand software requirements and the design that will achieve those requirements.
* **Construction:** This activity combines code generation (either manual or automated) and the testing that is required for uncovering errors in the code.
* **Deployment:** The software (as a complete entity or as a partially completed increment) is delivered to the customer who evaluates the delivered product and provides feedback based on the evaluation.

These five generic framework activities can be used during the development of smal1, simple programs, the creation of large web applications, and for the engineering of large, complex computer-based systems. The details of the software process will be quite different in each case, but the framework activities remain the same.

#### 4.3 The Linear Sequential Model/ Software Development Life Cycle

Sometimes called the classic life cycle or the waterfall model or software development life cycle (SDLC), the linear sequential model suggests a systematic, sequential approach to software development that begins at the system level and progresses through analysis, design, coding, testing, and support. A linear process flow executes each of the five framework activities in sequence, beginning with communication and culminating with deployment. Figure 2.3 illustrates the linear sequential model for software engineering. Modelled after a conventional engineering cycle, the linear sequential model encompasses the following activities.

##### 4.3.1 System/Information Engineering and Modelling

Because software is always part of a larger system (or business), work begins by establishing requirements for all system elements and then allocating some subset of these requirements to software. This system view is essential when software must interact with other elements such as hardware, people, and databases. System engineering and analysis encompass requirements gathering at the system level with a small amount of top level design and analysis. Information engineering encompasses requirements gathering at the strategic business level and at the business area level.

**Figure 4.2: Linear sequential models**

Analysis

Design

Code

Test

##### 4.3.2 Design

Software design is actually a multistep process that focuses on four distinct attributes of a program: data structure, software architecture, interface representations, and procedural (algorithmic) detail. The design process translates requirements into a representation of the software that can be assessed for quality before coding begins. Like requirements, the design is documented and becomes part of the software configuration.

##### 4.3.3 Code generation

The design must be translated into a machine-readable form. The code generation step performs this task. If design is performed in a detailed manner, code generation can be accomplished mechanistically.

##### 4.3.4 Testing

Once code has been generated, program testing begins. The testing process focuses on the logical internals of the software, ensuring that all statements have been tested, and on the functional externals; that is, conducting tests to uncover errors and ensure that defined input will produce actual results that agree with required results.

In [computer programming,](http://en.wikipedia.org/wiki/Computer_programming) unit testing is a method by which individual units of [source code,](http://en.wikipedia.org/wiki/Source_code) sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use.

##### 4.3.5 Support

Software will undoubtedly undergo change after it is delivered to the customer (a possible exception is embedded software). Change will occur because errors have been encountered, because the software must be adapted to accommodate changes in its external environment (e.g., a change required because of a new operating system or peripheral device), or because the customer requires functional or performance enhancements. Software support/maintenance reapplies each of the preceding phases to an existing program rather than a new one.

#### 4.4 Software Development Process Model

A software development process, also known as a software development lifecycle (SDLC), is a structure imposed on thedevelopment of a software produc[t.](http://en.wikipedia.org/wiki/Software_development) Similar terms include software life cycle and software process. It is often considered a subset of [systems development life cycle.](http://en.wikipedia.org/wiki/Systems_Development_Life_Cycle) There are several [models](http://en.wikipedia.org/wiki/Software_development_process#Software_development_models) for such processes, each describing approaches to a variety oftasks or activities that take place during the process. Some people consider a life-cycle model a more general term and a software development process a more specific term. For example, there are many specific software development processes that 'fit' the spiral life-cycle model. [ISO/IEC1220](http://en.wikipedia.org/wiki/ISO/IEC_12207) is an international standard for software life-cycle processes.

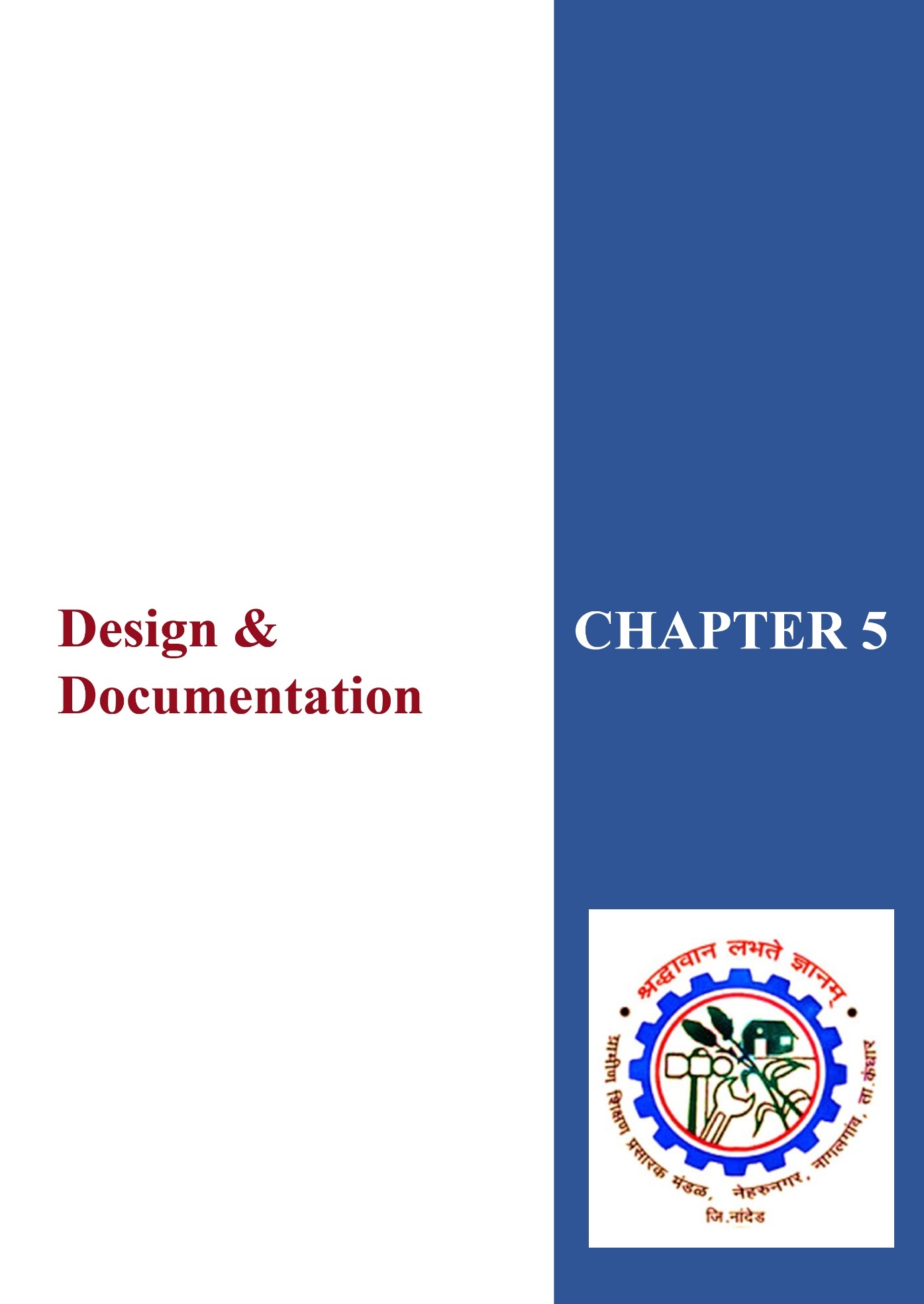
It aims to be the standard that defines all the tasks required for developing and maintaining software. A software development process model is a description of the work practices, tools and techniques used to develop software. Software models serve as standards as well provide guidelines while developing software. It is beneficial to one or more software development model while developing software.

##### 4.4.1 Waterfall Model

It includes a sequential approach to software development. It includes phases like task definition, analysis, design, implementation, testing and maintenance. The phases are always in order and are never overlapped. Then waterfall development model originates in the [manufacturing](http://en.wikipedia.org/wiki/Manufacturing) and construction industries; highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible. Since no formal software development methodologies existed at the time, this hardware-oriented model was simply adapted for software development. The phases are always in order and are never overlapped.

The waterfall model is the oldest paradigm for software engineering. However, over the past three decades, criticism of this process model has caused even ardent supporters to question its efficacy. The phases are always in order and are never overlapped. Among the problems that are sometimes encountered when the waterfall model is applied are:

* Real projects rarely follow the sequential flow that the model proposes. Although the linear model can accommodate iteration, it does so indirectly. As a result, changes can cause confusion as the project team proceeds.
* It is often difficult for the customer to state all requirements explicitly. The waterfall model requires this and has difficulty accommodating the natural uncertainty that exists at the beginning of many projects.
* The customer must have patience. A working version of the programs will not be available until late in the project time span. A major blunder, if undetected until the working program is reviewed, can be disastrous.



## *Chapter 5*

## DESIGN AND DOCUMENTATION

### • Algorithm

Step 1: Start.

Step 2: Show the splash screen for 5 seconds.

Step 3: Show home screen

Step 4: Open login page

Step 5: Register new user to create its account in app.

Step 6: Fill the detail related to user.

Step 7: Enter the field user are donor or not.

Step 8: Register the detail.

Step 9: After registration login your own account.

Step 10: Enter user name and password.

Step 11: After enter correct user name and password login your account.

Step 12: Display the blood bank search menu. You want to search blood.

Step 13: Select the blood group.

Step 14: Display the donor list of blood group.

Step 15: After click on donor name.

Step 16: Display the detail related to the donor.

Step 17: Second module is use to send request to blood bank. (Admin panel) Step 18: Admin is web-based detail information and authentication process Step 19: Also, can edit profile detail.

• **System - Flow Diagram:**

Blood

Bank

Blood

Donors

Blood

Seekers

Registration

Login Check

Details

Modification

Donor

Search

Life

Saving

Mobilink

Paging

FAQ’s

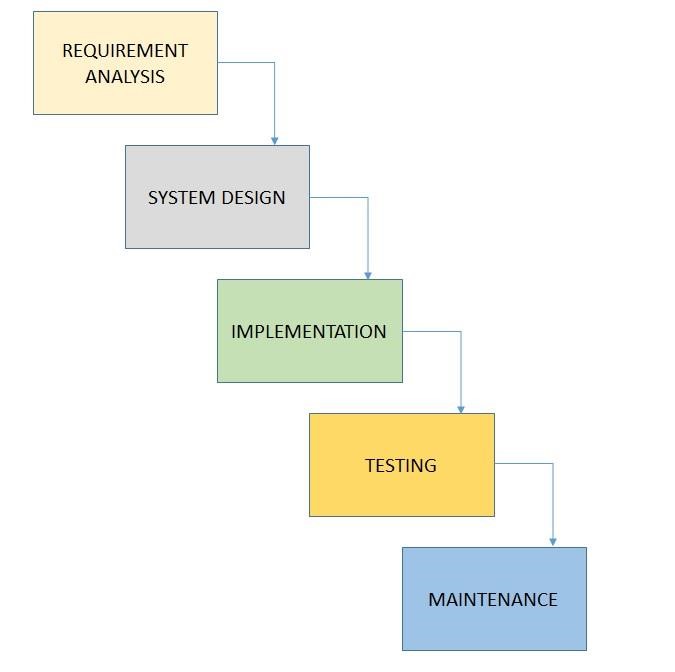
General

Information

Terms Of

Service

#### Figure 5.1: System Flow Diagram



#### Figure 5.2: Waterfall Model

• **Data - Flow Diagrams:**

Blood

Donors

Blood

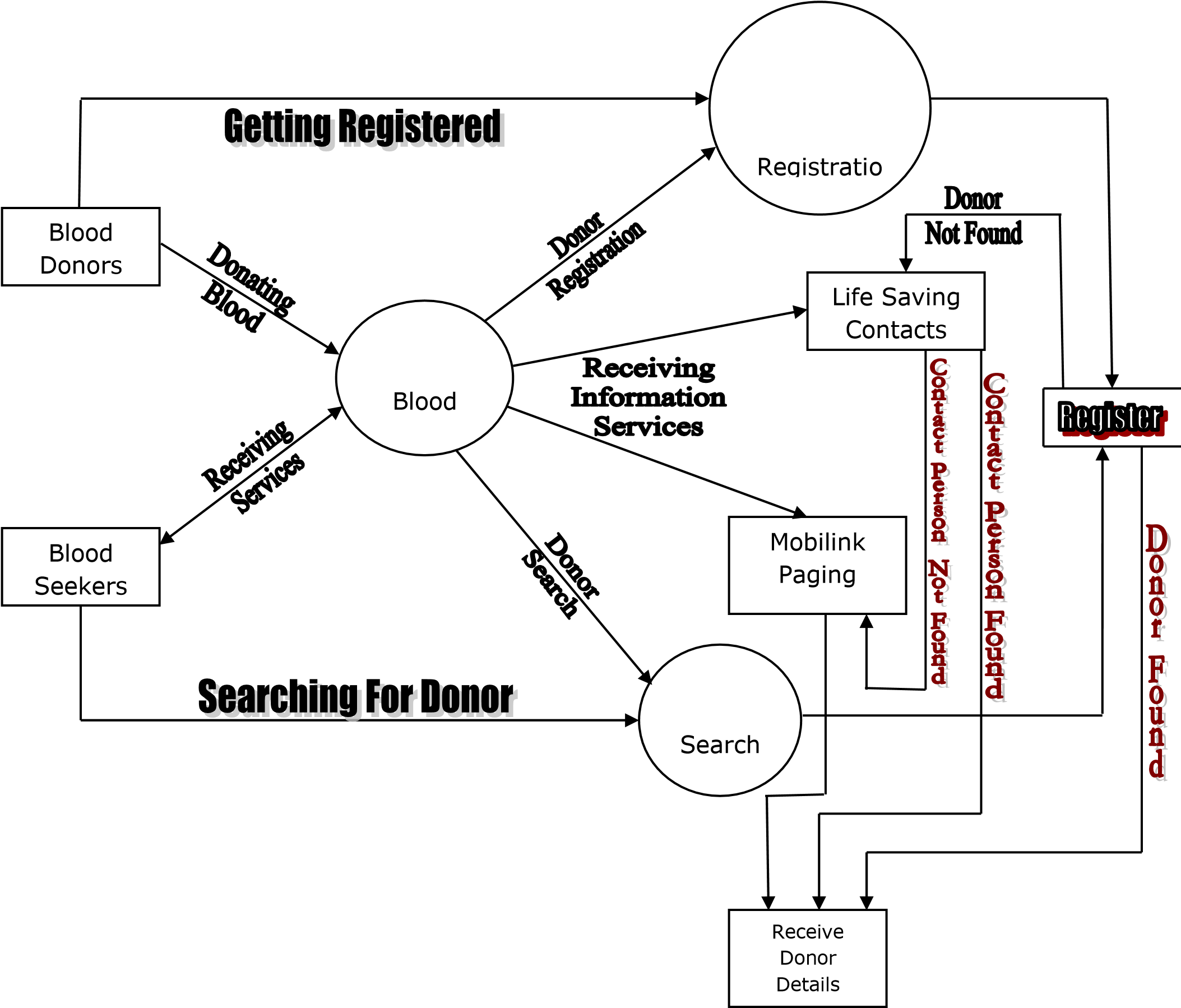
Blood

Seekers

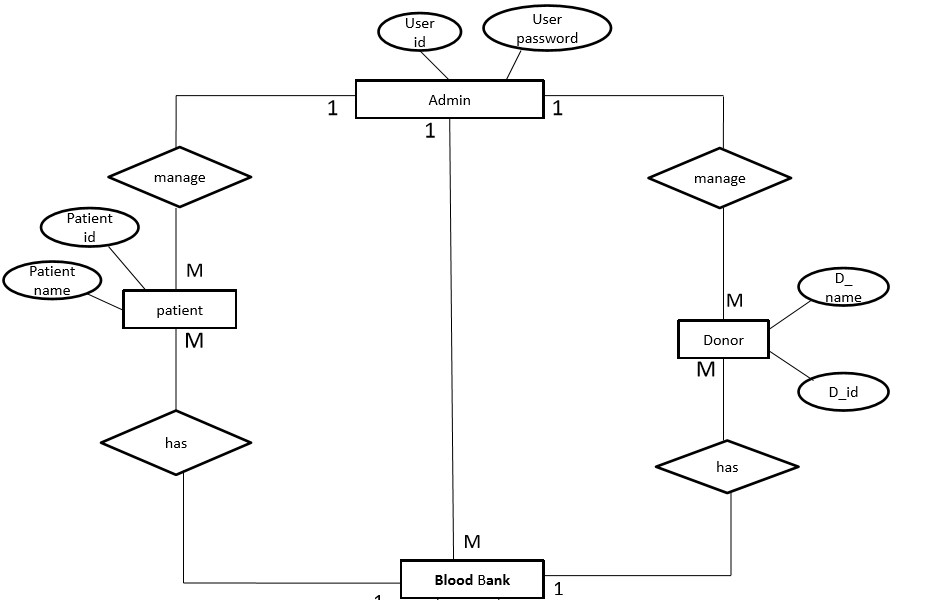
**Figure 5.3: I Context Level – DFD**



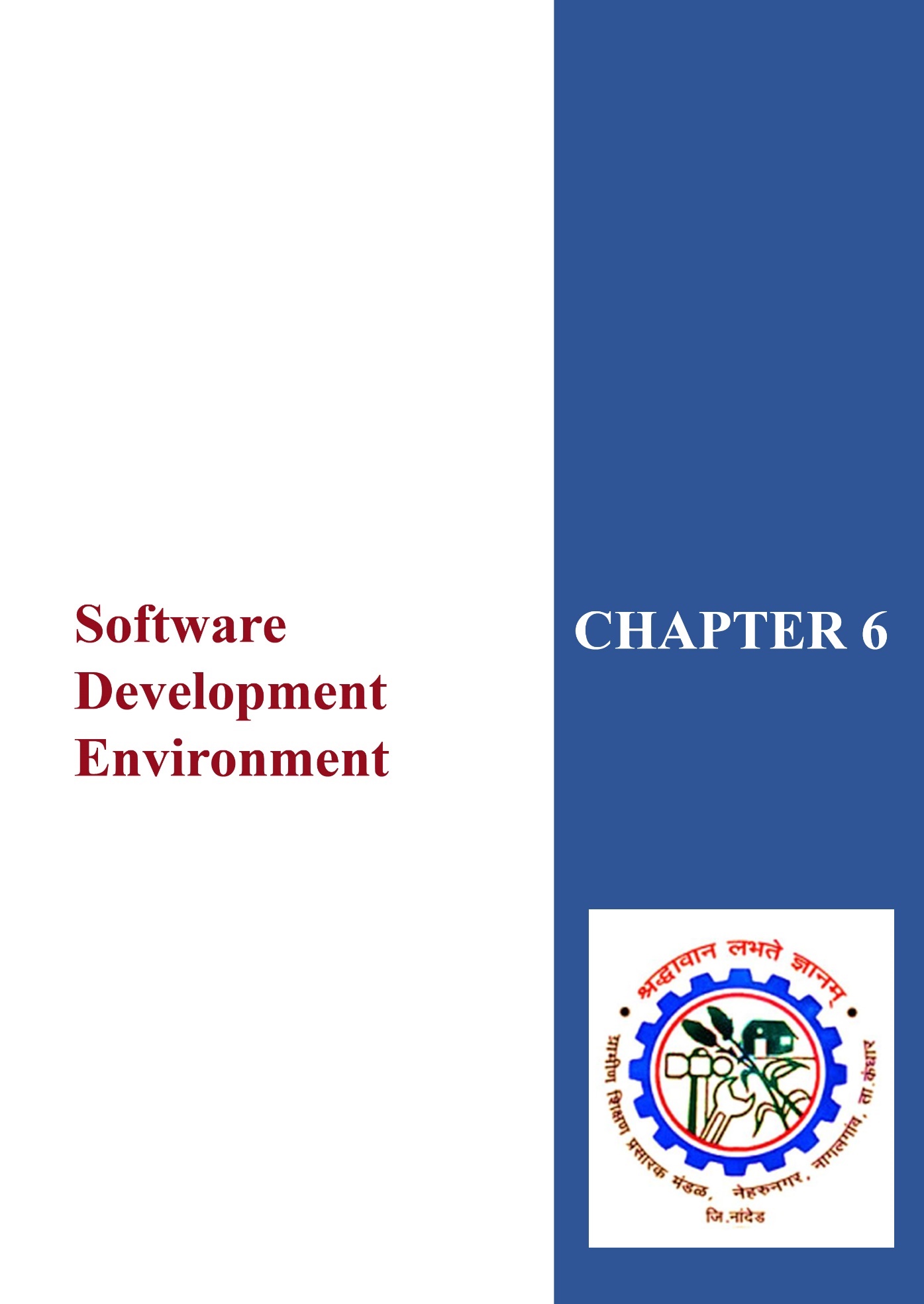
**Figure 5.4: I Level - DFD**



**Figure 5.5: Level – DFD**



**Figure 5.6: E-R Diagram**



#### *Chapter 6*

**SOFTWARE DEVELOPEMENT ENVIRONMENT**

##### 6.1 Software Environment

**Android** is a Linux-based operating system for mobile devices such as Smartphone’s and tablet computers. It is developed by the Open Handset Alliance led by Google . Android has a large community of developers writing applications ("apps") that extend the functionality of the devices. Developers write primarily in a customized version of Java.

###### 6.1.1 Android architecture

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. Android uses the Dalvik virtual machine with just-in-time compilation to run Dalvik dex-code (Dalvik Executable), which is usually translated from Java byte code.

The main hardware platform for Android is the ARM architecture. There is support for [x86](http://en.wikipedia.org/wiki/X86) from the [Android x86](http://en.wikipedia.org/wiki/Android_x86) project, and Google TV uses a special x86 version of Android.

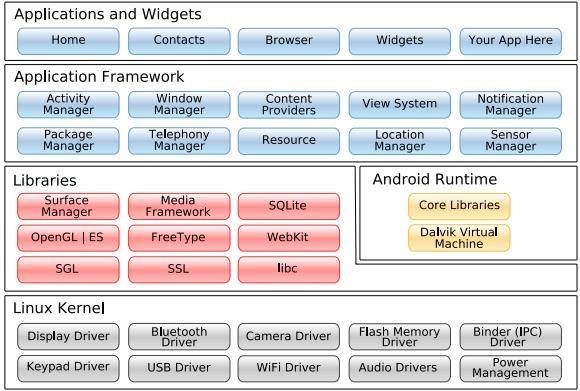


Figure 6.1. Android architecture

Android's kernel is based on the Linux kernel and has further architecture changes by Google outside the typical Linux kernel development cycle. Android does not have a native X Window System nor does it support the full set of standard GNU libraries, and this makes it difficult to port existing Linux applications or libraries to Android.

Certain features that Google contributed back to the Linux kernel, notably a power management feature called wake locks, were rejected by mainline kernel developers, partly because kernel maintainers felt that Google did not show any intent to maintain their own code. Even though Google announced in April 2010 that they would hire two employees to work with the Linux kernel community, Greg Kroah-Hartman, the current Linux kernel maintainer for the -stable branch, said in December 2010 that he was concerned that Google was no longer trying to get their code changes included in mainstream Linux.Some Google Android developers hinted that "the Android team was getting fed up with the process", because they were a small team and had more urgent work to do on Android.

However, in September 2010, Linux kernel developer Rafael J. Wysocki added a patch that improved the mainline Linux wakeup events framework. He said that Android device drivers that use wake locks can now be easily merged into mainline Linux, but that Android's opportunistic suspend features should not be included in the mainline kernel. In 2011 Linus Torvalds said that "eventually Android and Linux would come back to a common kernel, but it will probably not be for four to five years".

In December 2011, Greg Kroah-Hartman announced the start of the Android Mainlining Project, which aims to put some Android drivers, patches and features back into the Linux kernel, starting in Linux 3.3. further integration being expected for Linux Kernel 3.4.

6.1.2 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:

* A rich and extensible set of [Views](http://developer.android.com/resources/tutorials/views/index.html) that can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser
* [Content Providers](http://developer.android.com/guide/topics/providers/content-providers.html) that enable applications to access data from other applications (such as Contacts), or to share their own data
* A [Resource Manager,](http://developer.android.com/guide/topics/resources/resources-i18n.html) providing access to non-code resources such as localized strings, graphics, and layout files
* A [Notification Manager](http://developer.android.com/reference/android/app/NotificationManager.html) that enables all applications to display custom alerts in the status bar. An [Activity Manager](http://developer.android.com/reference/android/app/Activity.html) that manages the lifecycle of applications and provides a common navigation back stack

6.1.3 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
* **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
* **Free Type** - bitmap and vector font rendering
* **SQLite** - a powerful and lightweight relational database engine available to all applications

6.1.4 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.

6.1.5 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.

6.1.6 Android SDK

You should update to the latest tools or platform using the *Android SDK and AVD Manager*, rather than downloading a new SDK starter package.

http://dl.google.com/android/android-sdk\_r16-macosx.zip

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

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

###### 6.2 System Requirements

Supported Operating Systems

* Windows XP (32-bit), Vista (32- or 64-bit), or Windows 7 (32- or 64-bit)
* Mac OS X 10.5.8 or later (x86 only)
* Linux (tested on Ubuntu Linux, Lucid Lynx)
* GNU C Library (glibc) 2.7 or later is required.
* On Ubuntu Linux, version 8.04 or later is required.
* 64-bit distributions must be capable of running 32-bit applications. For information about how to add support for 32-bit applications supported Development Environments for Android SDK.

###### 6.3 Eclipse IDE

* Eclipse 3.6 (Helios) or greater
* Note: Eclipse 3.5 (Galileo) is no longer supported with the latest version of ADT.
* Eclipse JDT plug-in (included in most Eclipse IDE packages)
* If you need to install or update Eclipse, you can download it from [http://www.eclipse.org/downloads/.](http://www.eclipse.org/downloads/)

Several types of Eclipse packages are available for each platform. For developing Android applications, we recommend that you install one of these packages:

* Eclipse IDE for Java Developers
* Eclipse Classic
* Eclipse IDE for Java EE Developers
* [JDK 5 or JDK 6](http://www.oracle.com/technetwork/java/javase/downloads/index.html) (JRE alone is not sufficient)
* [Android Development Tools plug-in](http://developer.android.com/sdk/eclipse-adt.html) (recommended)
* Not compatible with Gnu Compiler for Java (gcj) Other development environments or IDEs
* [JDK 5 or JDK 6](http://www.oracle.com/technetwork/java/javase/downloads/index.html) (JRE alone is not sufficient)
* [Apache Ant](http://ant.apache.org/) 1.8 or later
* Not compatible with Gnu Compiler for Java (gcj)

###### 6.4 Hardware requirements for Android SDK

The Android SDK requires disk storage for all of the components that you choose to install. The table below provides a rough idea of the disk-space requirements to expect, based on the components that you plan to use.

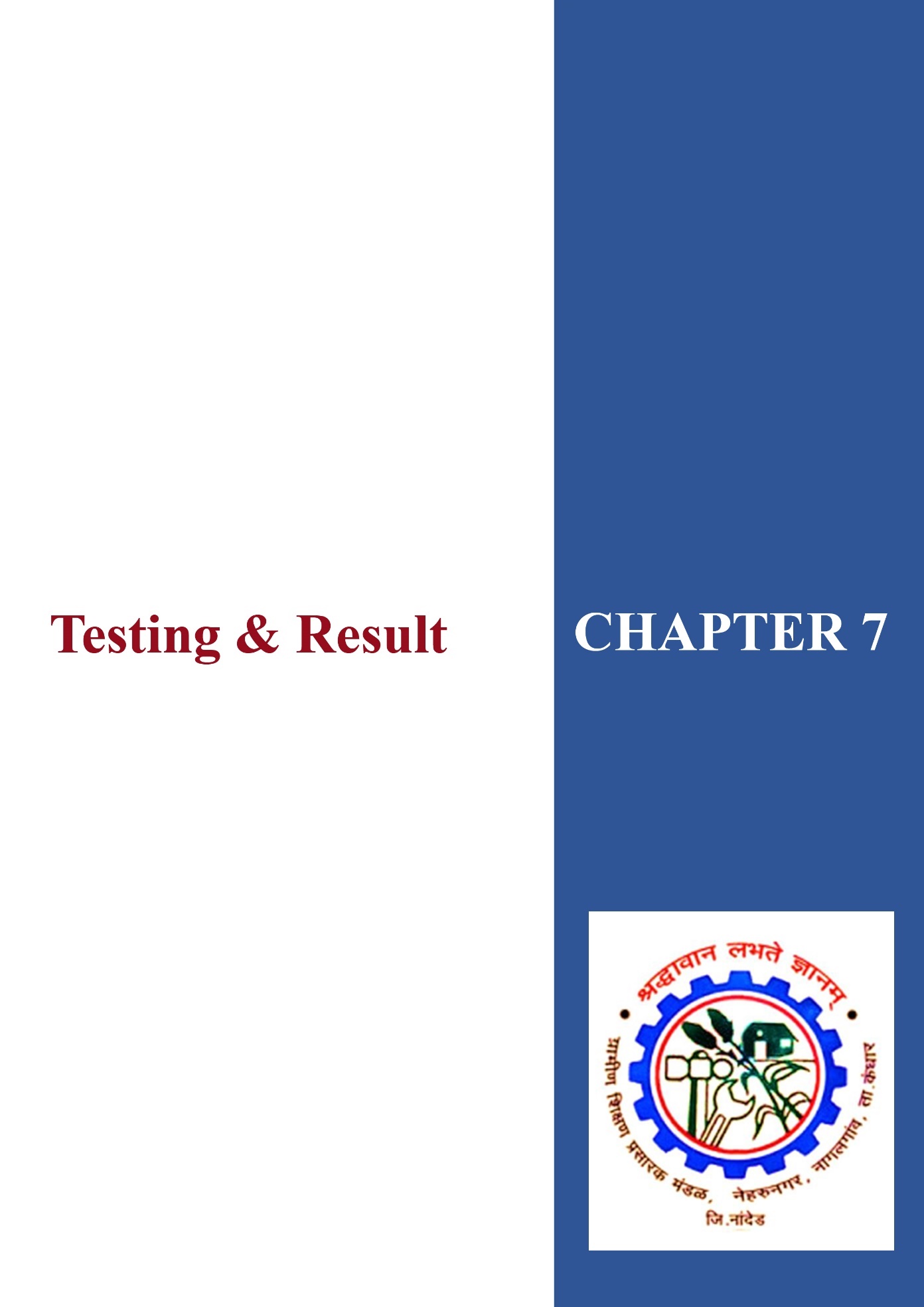
|  |  |  |
| --- | --- | --- |
| **Component type** | **Approximate size** | **Comments** |
| SDK Tools | 35 MB | Required. |
| SDK Platform-tools | 6 MB | Required. |
| Android platform (each) | 150 MB | At least one platform is required. |
| SDK Add-on (each) | 100 MB | Optional. |
| USB Driver for Windows | 10 MB | Optional. For Windows only. |
| Samples (per platform) | 10M | Optional. |
| Offline documentation | 250 MB | Optional. |

Notethat the disk-space requirements above are *in addition to* those of the Eclipse IDE, JDK, or other prerequisite tools that you may need to install on your development computer.

###### Version history

Android has seen a number of updates since its original release, each fixing bugs and adding new features. Each version is named, in alphabetical order, after a dessert.

* **2.3 Gingerbread** refined the user interface, improved the soft keyboard and copy/paste features, improved gaming performance, added 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, support for multi-core processors, hardware acceleration for graphics and full system encryption. The first device featuring this version, the Motorola Xoom tablet, went on sale in February 2011.
  + **3.1 Honeycomb**, released in May 2011, added support for extra input devices, USB host mode for transferring information directly from cameras and other devices, and the Google Movies and Books apps.
  + **3.2 Honeycomb**, released in July 2011, added optimization for a broader range of screen sizes, new "zoom-to-fill" screen compatibility mode, loading media files directly from SD card, and an extended screen support API. Huawei Media Pad is the first 7-inch tablet to use this version.
* **4.0 Ice Cream Sandwich**: Announced on October 19, 2011, brought Honeycomb features to smart phones and added new features including facial recognition unlock, network data usage monitoring and control, unified social networking contacts, photography enhancements, offline email searching, app folders, and information sharing using NFC. Android 4.0.3 Ice Cream Sandwich is the latest Android version that is available to phones. The source code of Android 4.0.1 was released on November 14, 2011.



***Chapter 7***

## TESTING AND RESULT

### 7.1 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.

### 7.2 Types Of Tests

* Unit testing
* Integration testing
* Functional test
* System Test
* White Box Testing
* Black Box Testing
* Unit Testing

#### 7.2.1 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.

#### 7.2.2 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.

#### 7.2.3 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 centred 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.

#### 7.2.4 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.

#### 7.2.5 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.

#### 7.2.6 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.

### 7.3 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**

o Verify that the entries are of the correct format.

o No duplicate entries should be allowed.

o All links should take the user to the correct page.

### 7.4 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.

### 7.5 Acceptance Testing

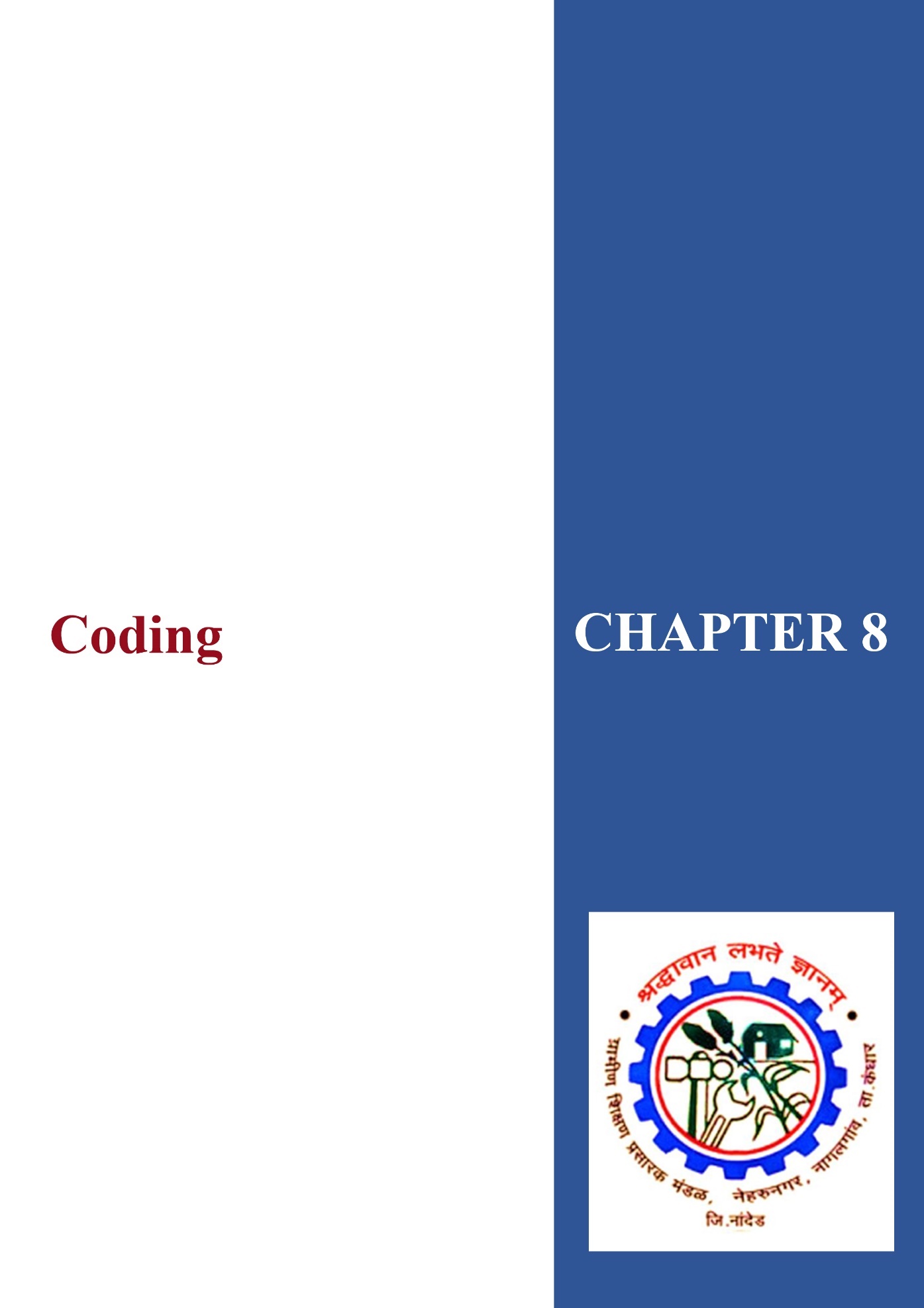
User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

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

### 7.6 Test Cases

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Scenario ID** | | Login-1 | | | **Test Case ID** | | Login-TBA | | |
| **Test Case Description** | | Login – Test Case TBA | | | **Test Priority** | | High | | |
| **Pre-Requisite** | | NA | | | **Post-Requisite** | | NA | | |
|  | |  | | |  | |  | | |
| Test Execution Steps: | | | | | | | | | | |
| **Test Case id** | **Objective** | | **Inputs** | **Expected Result** | **Actual Result** | **Test App** | | **Test Result** | **Test Comments** | |
| TC\_1 | Launch application  “The Blood Alliance”. | | Click/open the application “The Blood Alliance”. | The Blood Alliance homepage is launched successfully. | The Blood Alliance homepage is launched successfully. | IE -11 | | Pass | [V\_D 26/03/2022 09:44 PM]: Launch successful. | |
| TC\_2 | Enter invalid Username & any Password and hit login button. | | Username: invalid123  Password: \*\*\*\*\*\* | The username or password that you've entered doesn't match any account. [Sign up for an account.](https://www.facebook.com/r.php) | The Username or password number that you've entered doesn't match any account. [Sign up for an account.](https://www.facebook.com/r.php) | IE -11 | | Fail | [V\_D 26/03/2022 09:45 PM]: Invalid login attempt stopped. | |
| TC\_3 | Enter valid Username & incorrect Password and hit login button. | | Username: valid123.com  Password: \*\*\*\*\*\* | The password that you've entered is incorrect. [Forgotten password?](https://www.facebook.com/recover/initiate?lwv=120&lwc=1348092) | The password that you've entered is incorrect. [Forgotten password?](https://www.facebook.com/recover/initiate?lwv=120&lwc=1348092) | IE -11 | | Fail | [V\_D 26/03/2022 09:46 PM]: Invalid login attempt stopped. | |
| TC\_4 | Enter valid Username & valid Password and hit login button. | | Username: valid123.com  Password: \*\*\*\*\*\* | You have Logged in successfully! | You have Logged in successfully! | IE -11 | | Pass | [V\_D 26/03/2022 09:48 PM]: logged in. | |
| TC\_5 | Click register button. | | Click on register button. | The register page is opened. | The register page is opened. | IE -11 | | Pass | [V\_D 26/03/2022 09:50 PM]: Registration page opened. | |
| TC\_6 | Select valid name. | | Click on text field. | Text field enabled for name field. | Text field enabled for name field. | IE -11 | | Pass | [V\_D 26/03/2022 09:51 PM]: Name entered. | |
| TC\_7 | Select the gender. | | Select your gender. | The gender radio box is enabled. | The gender radio box is enabled. | IE -11 | | Pass | [V\_D 26/03/2022 09:52 PM]: Selected chosen gender. | |
| TC\_8 | Enter valid address. | | Click on text field. | Text field enabled for address field. | Text field enabled for address field. | IE -11 | | Pass | [V\_D 26/03/2022 09:53 PM]: Address entered. | |
| TC\_9 | Enter valid city. | | Click on text field. | Text field enabled for city. | Text field enabled for city. | IE -11 | | Pass | [V\_D 26/03/2022 09:51 PM]:City entered. | |
| TC\_10 | Enter valid email id. | | Click on text field. | Text field enabled for email id. | Text field enabled for email id. | IE -11 | | Pass | [V\_D 26/03/2022 09:54 PM]: Selected valid email id. | |
| TC\_11 | Enter valid contact number. | | Click on text field for contact number. | Text field enabled for contact number. | Text field enabled for contact number. | IE -11 | | Pass | [V\_D 26/03/2022 09:55 PM]: Selected valid contact number. | |
| TC\_12 | Create username. | | Click on text field. | Text field enabled for creating user name. | Text field enabled for creating user name. | IE -11 | | Pass | [V\_D 26/03/2022 09:57 PM]: Username created. | |
| TC\_13 | Create password. | | Click on text field. | Text field enabled for creating password. | Text field enabled for creating password. | IE -11 | | Pass | [V\_D 26/03/2022 09:59 PM]: Selected valid password. | |
| TC\_14 | Browse image | | Click on browse button to select image. | Browser button enabled to choose image. | Browser button enabled to choose image. | IE -11 | | Pass | [V\_D 26/03/2022 10:00 PM]: Selected profile image. | |
| TC\_15 | Choose blood group. | | Click on list option to select the blood group. | List items enabled and blood groups visible. | List items enabled and blood groups visible. | IE -1 | | Pass | [V\_D 26/03/2022 10:01 PM]: Selected chosen blood group. | |
| TC\_16 | Select donor box. | | Click on check box to check donor box. | Check box enabled. | Check box enabled. | IE -1 | | Pass | [V\_D 26/03/2022 10:02 PM]: Chosen box. | |
| TC\_17 | Skip any one step and click register button. | | Click on register button to proceed. | Register button disabled please fill all required fields and try again. | Register button disabled please fill all required fields and try again. | IE -1 | | Fail | [V\_D 26/03/2022 10:04 PM]: Filling all details is mandatory. | |
| TC\_18 | Fill all details and click register button. | | Click on register button. | Register button enabled, you are registered as donor. | Register button enabled, you are registered as donor. | IE -1 | | Pass | [V\_D 26/03/2022 10:06 PM]: Registered successfully. | |
| TC\_19 | Click on the rules and regulations button. | | Click on the rules and regulations button. | Rules and Regulations Button Enabled. | Rules and Regulations Button has been Enabled. | IE-1 | | Pass | V\_D 26/03/2022 10:10 PM]: Registered successfully. | |
| TC\_20 | After clicking the button we will be redirected to the rules and regulations interface | | After clicking the button we will be redirected to the rules and regulations interface. | Rules and Regulations Interface is opened. | Rules and Regulations Page is opened and displayed. | IE-1 | | Pass | V\_D 26/03/2022 10:12 PM]: Registered successfully. | |

### 



***Chapter 8***

## CODING

**HomePage.java**

package com.android.tabfragments;

import android.app.ProgressDialog;

import android.content.SharedPreferences; import android.os.AsyncTask;

import android.os.Bundle;

import android.support.v4.app.Fragment; import android.util.Log;

import android.view.LayoutInflater; import android.view.View;

import android.view.ViewGroup;

import android.widget.AdapterView;

import android.widget.ArrayAdapter; import android.widget.Button;

import android.widget.EditText;

import android.widget.Spinner;

import android.widget.TextView;

import android.widget.Toast;

import com.androidbloodbank.R;

import org.apache.http.HttpEntity;

import org.apache.http.HttpResponse; import org.apache.http.NameValuePair; import rg.apache.http.client.HttpClient;

import rg.apache.http.client.entity.UrlEncodedFormEntity; import org.apache.http.client.methods.HttpPost;

import org.apache.http.impl.client.DefaultHttpClient; import org.apache.http.message.BasicNameValuePair;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.io.BufferedReader; import java.io.IOException; import java.io.InputStream; import java.io.InputStreamReader; importjava.text.SimpleDateFormat; import java.util.ArrayList; import java.util.Date;

import static android.content.Context.*MODE\_PRIVATE*;

public class BloodBank extends Fragment {

View v;

String bloodgroup, bloodtype;

String result = "", line = "";

InputStream is = null; ProgressDialog pDialog = null; int code; int quantity;

String reqquantity;

Button b;

Spinner sp, sp1;

EditText etquantity;

TextView aplus, aminus, bplus, bminus, abplus, abminus, oplus, ominus;

SimpleDateFormat df;

String formattedDate;

SharedPreferences pref;

SharedPreferences.Editor edit;

String name, address, mbno, city;

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

v = inflater.inflate(R.layout.fragment\_blood\_bank, container, false);

df = new SimpleDateFormat("yyyy/MM/dd"); formattedDate = df.format(new Date());

pref =getActivity().getSharedPreferences("requestrecord", *MODE\_PRIVATE*); name = pref.getString("name", ""); mbno = pref.getString("mbno", ""); address = pref.getString("address", ""); city = pref.getString("city", "");

etquantity = (EditText) v.findViewById(R.id.etquantity); aplus = (TextView) v.findViewById(R.id.Aplusquantity); aminus = (TextView) v.findViewById(R.id.Aminusquantity); bplus = (TextView) v.findViewById(R.id.Bplusquantity); bminus = (TextView) v.findViewById(R.id.Bminusquantity); abplus = (TextView) v.findViewById(R.id.ABplusquantity); abminus = (TextView) v.findViewById(R.id.ABminusquantity); oplus = (TextView) v.findViewById(R.id.Oplusquantity); ominus = (TextView) v.findViewById(R.id.Ominusquantity); sp = (Spinner) v.findViewById(R.id.spbloodgrp); sp1 = (Spinner) v.findViewById(R.id.spbloodtype); b = (Button) v.findViewById(R.id.brequest); ArrayAdapter<CharSequence> adapter =

ArrayAdapter.*createFromResource*(getActivity(),

R.array.bloodgroup, android.R.layout.*simple\_spinner\_item*);

adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*);

sp.setAdapter(adapter);

ArrayAdapter<CharSequence> adapter1 =

ArrayAdapter.*createFromResource*(getActivity(), R.array.bloodtype, android.R.layout.*simple\_spinner\_item*);

adapter1.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)

;

sp1.setAdapter(adapter1);

sp.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() { public void onItemSelected(AdapterView<?> parent, View view, int pos, long id)

{

if (sp.getSelectedItem().toString().trim().equals("A")) { bloodgroup = "A";

} else if (sp.getSelectedItem().toString().trim().equals("A+"))

{

bloodgroup = "A+";

} else if

(sp.getSelectedItem().toString().trim().equals("AB+")) { bloodgroup = "AB+";

} else if (sp.getSelectedItem().toString().trim().equals("AB")) { bloodgroup = "AB-";

} else if (sp.getSelectedItem().toString().trim().equals("B-"))

{ bloodgroup = "B-";

} else if (sp.getSelectedItem().toString().trim().equals("B+"))

{

bloodgroup = "B+";

} else if (sp.getSelectedItem().toString().trim().equals("O-"))

{ bloodgroup = "O-";

} else if (sp.getSelectedItem().toString().trim().equals("O+"))

{

bloodgroup = "O+";

}

}

public void onNothingSelected(AdapterView<?> parent) {

*// Another interface callback*

}

});

sp1.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() { public void onItemSelected(AdapterView<?> parent, View view, int pos, long id)

{

if (sp1.getSelectedItem().toString().trim().equals("Red Blood Cell")) { bloodtype = "Red Blood Cell";

} else if

(sp1.getSelectedItem().toString().trim().equals("Plasma")) { bloodtype = "Plasma";

} else if

(sp1.getSelectedItem().toString().trim().equals("Platelets")) { bloodtype = "Platelets";

}

}

public void onNothingSelected(AdapterView<?> parent) {

*// Another interface callback*

}

});

b.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View view) {

reqquantity = etquantity.getText().toString();

Log.*d*("Name", name);

Log.*d*("MbNo", mbno);

Log.*d*("address", address);

Log.*d*("city", city);

Log.*d*("bloodgroup", bloodgroup);

Log.*d*("reqquantity", reqquantity); Log.*d*("Date", formattedDate); new Update().execute();

}

});

new Show().execute();

return v;

}

class Show extends AsyncTask<Void, Void, Void> {

@Override protected void onPreExecute() { *// TODO Auto-generated method stub* super.onPreExecute();

pDialog = new ProgressDialog(getActivity()); pDialog.setMessage("Searching"); pDialog.setIndeterminate(false); pDialog.setCancelable(false); pDialog.show();

}

@Override

protected Void doInBackground(Void... voids) { if (pDialog.isShowing()) { pDialog.dismiss(); } try {

HttpClient hc = new DefaultHttpClient();

HttpPost hp = new

HttpPost("http://192.168.43.68/AndroidBloodBank/bloodbank.php");

HttpResponse hr = hc.execute(hp);

HttpEntity he1 = hr.getEntity();

is = he1.getContent();

Log.*d*("OUTPUT:", is.toString());

} catch (Exception e) {

Log.*e*("Fail 1", e.toString());

} return null; }

@Override

protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid);

try {

BufferedReader reader = new BufferedReader(new

InputStreamReader(is, "ISO-8859-1"), 8);

StringBuilder sb = new StringBuilder(); while ((line = reader.readLine()) != null) { sb.append(line + "");

}

result = sb.toString();

Log.*d*("TAAAAAG", result);

} catch (Exception e) {

*// TODO: handle exception*

} try {

JSONObject jsonResponse = new JSONObject(result);

JSONArray jsonMainNode = jsonResponse.optJSONArray("quantity");

for (int i = 0; i < jsonMainNode.length(); i++) {

JSONObject jsonChildNode = jsonMainNode.getJSONObject(i); if (jsonMainNode.getJSONObject(i) != null) { String bloodgroup = jsonChildNode.optString("bloodgroup");

quantity = Integer.*parseInt*(jsonChildNode.optString("quantity"));

if (bloodgroup.equals("A+")) {

String aplusquantity = jsonChildNode.optString("quantity");

Log.*d*("A+", aplusquantity); aplus.setText(aplusquantity);

} else if (bloodgroup.equals("A-")) { String aminusquantity = jsonChildNode.optString("quantity");

Log.*d*("A-", aminusquantity); aminus.setText(aminusquantity);

} else if (bloodgroup.equals("B+")) {

String bplusquantity = jsonChildNode.optString("quantity");

Log.*d*("B+", bplusquantity); bplus.setText(bplusquantity);

} else if (bloodgroup.equals("B-")) { String bminusquantity = jsonChildNode.optString("quantity");

Log.*d*("B-", bminusquantity); bminus.setText(bminusquantity);

} else if (bloodgroup.equals("AB+")) {

String abplusquantity =

jsonChildNode.optString("quantity");

Log.*d*("AB+", jsonChildNode.optString("AB+")); abplus.setText(abplusquantity);

} else if (bloodgroup.equals("AB-")) { String abminusquantity =

jsonChildNode.optString("quantity");

Log.*d*("AB-", jsonChildNode.optString("AB-")); abminus.setText(abminusquantity);

} else if (bloodgroup.equals("O+")) { String obplusquantity =

jsonChildNode.optString("quantity");

Log.*d*("O+", jsonChildNode.optString("O+")); oplus.setText(obplusquantity);

} else if (bloodgroup.equals("O-")) { String obminusquantity =

jsonChildNode.optString("quantity");

Log.*d*("O-", jsonChildNode.optString("O-")); ominus.setText(obminusquantity);

}

}

}

} catch (JSONException e) {

Toast.*makeText*(getActivity(), "No Record Found",

Toast.*LENGTH\_SHORT*).show();

}

}

}

class Update extends AsyncTask<Void, Void, Void> { ArrayList<NameValuePair>nameValuePairs = new

ArrayList<NameValuePair>();

@Override protected void onPreExecute() { *// TODO Auto-generated method stub* super.onPreExecute();

pDialog = new ProgressDialog(getActivity()); pDialog.setMessage("Updating"); pDialog.setIndeterminate(false); pDialog.setCancelable(false); pDialog.show();

}

@Override

protected Void doInBackground(Void... voids) { if (pDialog.isShowing()) { pDialog.dismiss(); } try {

nameValuePairs.add(new BasicNameValuePair("name", name)); nameValuePairs.add(new BasicNameValuePair("address", address)); nameValuePairs.add(new BasicNameValuePair("mbno", mbno)); nameValuePairs.add(new BasicNameValuePair("city", city)); nameValuePairs.add(new BasicNameValuePair("bloodgroup", bloodgroup)); nameValuePairs.add(new BasicNameValuePair("bloodtype", bloodtype)); nameValuePairs.add(new BasicNameValuePair("quantity", reqquantity)); nameValuePairs.add(new BasicNameValuePair("date", formattedDate)); nameValuePairs.add(new BasicNameValuePair("status", "" + 0));

HttpClient hc = new DefaultHttpClient();

HttpPost hp = new

HttpPost("http://192.168.43.68/AndroidBloodBank/request.php"); hp.setEntity(new UrlEncodedFormEntity(nameValuePairs));

HttpResponse hr = hc.execute(hp);

HttpEntity he1 = hr.getEntity();

is = he1.getContent();

Log.*d*("OUTPUT:", is.toString());

} catch (Exception e) {

Log.*e*("Fail 1", e.toString());

} return null;

}

@Override

protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid);

try {

BufferedReader reader = new BufferedReader(new

InputStreamReader(is, "ISO-8859-1"), 8);

StringBuilder sb = new StringBuilder(); while ((line = reader.readLine()) != null) { sb.append(line + "");

}

result = sb.toString();

Log.*d*("TAAAAAG", result);

} catch (Exception e) {

*// TODO: handle exception*

} try {

JSONObject json\_data = new JSONObject(result + ""); code = (json\_data.getInt("code")); Log.*d*("COOODE", "" + code);

} catch (Exception e) {

*// TODO: handle exception*

Log.*d*("ERRRRRRRRR", "" + e.toString());

} if (code == 1) {

Toast.*makeText*(getActivity(), "Request sent.Wait for Admin

Approval" , Toast.*LENGTH\_LONG*).show();

try { is.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

**Signup page:**

findViewById(R.id.etemailid);

etmobileno = (EditText) findViewById(R.id.etcontactno); etusername = (EditText) findViewById(R.id.etusername); etpassword = (EditText) findViewById(R.id.etpassword); sp = (Spinner) findViewById(R.id.spbloodgrp);

public class SignUpActivity extends Activity {

RadioGroup rbg;

EditText etname, etmobileno, etemailid, etgender, etaddress, etcity, etusername, etpassword;

CheckBox cbdonar;

Spinner sp;

Button bpic, bregister;

ImageView iv;

RadioButton radiobutton;

String imagename;

Bitmap thumbnail;

SharedPreferences pref;

SharedPreferences.Editor edit;

private static final int *CAPTURE\_IMAGE\_ACTIVITY\_REQUEST\_CODE* = 1888;

String result = "", line, img = ""; int REQUEST\_CAMERA = 0, SELECT\_FILE = 1;

String userChoosenTask;

InputStream is = null; int code;

Bundle b = new Bundle();

ProgressDialog pDialog = null;

String name, city, mbno, address, emailid, username, password, donar, gender, bloodgroup;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState);

setContentView(R.layout.signup);

bregister = (Button) findViewById(R.id.bregister); bpic = (Button) findViewById(R.id.bpic); iv = (ImageView) findViewById(R.id.ivpic); cbdonar = (CheckBox) findViewById(R.id.cbdonar); etname = (EditText) findViewById(R.id.etname); etcity = (EditText) findViewById(R.id.etcity); etaddress = (EditText) findViewById(R.id.etaddress); etemailid = (EditText)

rbg = (RadioGroup) findViewById(R.id.radioGroup1);

bpic.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View view) { selectImage();

}

});

ArrayAdapter<CharSequence> adapter =

ArrayAdapter.*createFromResource*(SignUpActivity.this,

R.array.bloodgroup, android.R.layout.*simple\_spinner\_item*);

adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)

;

sp.setAdapter(adapter);

*// Inflate the layout for this fragment*

bregister.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View view) { name = etname.getText().toString(); city = etcity.getText().toString(); address = etaddress.getText().toString(); emailid = etemailid.getText().toString(); mbno = etmobileno.getText().toString(); username = etusername.getText().toString(); password = etpassword.getText().toString();

int selectedId = rbg.getCheckedRadioButtonId(); radiobutton = (RadioButton) findViewById(selectedId); gender = radiobutton.getText().toString();

if (cbdonar.isChecked()) donar = "true"; else donar = "false";

pref = getApplicationContext().getSharedPreferences("requestrecord",

*MODE\_PRIVATE*);

edit = pref.edit();

edit.putString("name", name); edit.putString("mbno", mbno); edit.putString("address", address); edit.putString("bloodgroup", bloodgroup); edit.putString("city", city); edit.commit();

new insert().execute();

}

});

sp.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() { public void onItemSelected(AdapterView<?> parent, View view, int pos, long id)

{

if (sp.getSelectedItem().toString().trim().equals("A")) { bloodgroup = "A";

} else if

(sp.getSelectedItem().toString().trim().equals("A+")) { bloodgroup = "A+";

} else if

(sp.getSelectedItem().toString().trim().equals("AB+")) { bloodgroup = "AB+";

} else if (sp.getSelectedItem().toString().trim().equals("AB-

")) {

bloodgroup = "AB-";

} else if (sp.getSelectedItem().toString().trim().equals("B-

")) {

bloodgroup = "B-";

} else if

(sp.getSelectedItem().toString().trim().equals("B+")) {

bloodgroup = "B+";

} else if (sp.getSelectedItem().toString().trim().equals("O-

")) {

bloodgroup = "O-";

} else if

(sp.getSelectedItem().toString().trim().equals("O+")) { bloodgroup = "O+";

}

}

public void onNothingSelected(AdapterView<?> parent) {

*// Another interface callback*

}

});

}

public class insert extends AsyncTask<Void, Void, Void> {

ArrayList<NameValuePair>nameValuePairs = new

ArrayList<NameValuePair>();

@Override protected void onPreExecute() { *// TODO Auto-generated method stub* super.onPreExecute();

pDialog = new ProgressDialog(SignUpActivity.this); pDialog.setMessage("Signing"); pDialog.setIndeterminate(false); pDialog.setCancelable(false); pDialog.show();

}

@Override

protected Void doInBackground(Void... params) { nameValuePairs.add(new BasicNameValuePair("name", name)); nameValuePairs.add(new BasicNameValuePair("gender", gender)); nameValuePairs.add(new BasicNameValuePair("emailid", emailid)); nameValuePairs.add(new BasicNameValuePair("mbno", mbno)); nameValuePairs.add(new BasicNameValuePair("address", address)); nameValuePairs.add(new BasicNameValuePair("city", city));

nameValuePairs.add(new BasicNameValuePair("bloodgroup", bloodgroup));

*//image path left*

nameValuePairs.add(new BasicNameValuePair("pic", convertBitmapToString(thumbnail)));

nameValuePairs.add(new BasicNameValuePair("imagename", imagename));

nameValuePairs.add(new BasicNameValuePair("donar", donar)); nameValuePairs.add(new BasicNameValuePair("username", username)); nameValuePairs.add(new BasicNameValuePair("password", password));

try {

HttpClient hc = new DefaultHttpClient();

HttpPost hp = new

HttpPost("http://192.168.43.68/AndroidBloodBank/register.php"); hp.setEntity(new UrlEncodedFormEntity(nameValuePairs));

HttpResponse hr = hc.execute(hp);

HttpEntity he1 = hr.getEntity();

is = he1.getContent();

Log.*d*("OUTPUT:", is.toString());

} catch (Exception e) {

Log.*e*("Fail 1", e.toString());

} return null; }

@Override

protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid); pDialog.dismiss();

try {

BufferedReader reader = new BufferedReader(new

InputStreamReader(is, "iso-8859-1"), 8);

StringBuilder sb = new StringBuilder();

while ((line = reader.readLine()) != null) { sb.append(line + "\n");

}

*/\* SignUp.this.runOnUiThread(new Runnable() { public void run() { // show message here*

*} });\*/* result = sb.toString();

Log.*d*("pass 2", result);

} catch (Exception e) {

Log.*e*("Fail 2", e.toString());

}

try {

JSONObject json\_data = new JSONObject(result + "");

code = (json\_data.getInt("code")); Log.*d*("COOODE", "" + code);

} catch (Exception e) {

*// TODO: handle exception*

Log.*d*("ERRRRRRRRR", "" + e.toString());

} if (code == 1) {

Toast.*makeText*(getApplicationContext(), "Sign Up Sucessful",

Toast.*LENGTH\_LONG*).show();

Intent intent = new Intent(SignUpActivity.this,

LoginActivity.class);

b.putString("username", username); intent.putExtra("bundle", b); startActivity(intent); try { is.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

public String convertBitmapToString(Bitmap bmp) {

ByteArrayOutputStream stream = new ByteArrayOutputStream(); bmp.compress(Bitmap.CompressFormat.*PNG*, 90, stream); *//compress to which format you want.*

byte[] byte\_arr = stream.toByteArray();

String imageStr = Base64.*encodeBytes*(byte\_arr); Log.*d*("Image in String------", imageStr); return imageStr;

}

public void selectImage() {

final CharSequence[] items = {"Take Photo", "Choose from Library",

"Cancel"};

AlertDialog.Builder builder = new AlertDialog.Builder(SignUpActivity.this); builder.setTitle("Add Photo!");

builder.setItems(items, new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int item) { boolean result = Utility.*checkPermission*(SignUpActivity.this);

if (items[item].equals("Take Photo")) { userChoosenTask = "Take Photo"; if (result)

cameraIntent();

} else if (items[item].equals("Choose from Library")) { userChoosenTask = "Choose from Library"; if (result)

galleryIntent();

} else if (items[item].equals("Cancel")) { dialog.dismiss();

}

}

});

builder.show();

}

@Override

public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) { switch (requestCode) { case Utility.*MY\_PERMISSIONS\_REQUEST\_READ\_EXTERNAL\_STORAGE*:

if (grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) { if (userChoosenTask.equals("Take Photo")) cameraIntent();

else if (userChoosenTask.equals("Choose from Library")) galleryIntent();

} else {

*//code for deny*

} break; }

}

private void galleryIntent() { Intent intent = new Intent(); intent.setType("image/\*"); intent.setAction(Intent.*ACTION\_GET\_CONTENT*);*//*

startActivityForResult(Intent.*createChooser*(intent, "Select File"),

SELECT\_FILE);

}

private void cameraIntent() {

Intent intent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*); startActivityForResult(intent, REQUEST\_CAMERA);

}

@Override

public void onActivityResult(int requestCode, int resultCode, Intent data) { super.onActivityResult(requestCode, resultCode, data);

if (resultCode == Activity.*RESULT\_OK*) { if (requestCode == SELECT\_FILE) {

Uri selectedImageUri = data.getData();

onSelectFromGalleryResult(data);

} else if (requestCode == REQUEST\_CAMERA) onCaptureImageResult(data);

}

}

private void onCaptureImageResult(Intent data) { thumbnail = (Bitmap) data.getExtras().get("data");

ByteArrayOutputStream bytes = new ByteArrayOutputStream(); thumbnail.compress(Bitmap.CompressFormat.*JPEG*, 90, bytes);

File destination = new File(Environment.*getExternalStorageDirectory*(),

System.*currentTimeMillis*() + ".jpg"); imagename = destination.getName(); Log.*d*("Name of image", imagename);

FileOutputStream fo; try {

destination.createNewFile(); fo = new FileOutputStream(destination); fo.write(bytes.toByteArray()); fo.close();

} catch (FileNotFoundException e) {

e.printStackTrace(); } catch (IOException e) {

e.printStackTrace();

}

iv.setImageBitmap(thumbnail); img = convertBitmapToString(thumbnail); Log.*d*("Img in String------", img);

}

@SuppressWarnings("deprecation")

private void onSelectFromGalleryResult(Intent data) {

if (data != null) { try { thumbnail =

MediaStore.Images.Media.*getBitmap*(SignUpActivity.this.getContentResolver(), data.getData());

imagename = data.getData().getLastPathSegment() + ".jpg";

} catch (IOException e) {

e.printStackTrace();

}

}

iv.setImageBitmap(thumbnail); img = convertBitmapToString(thumbnail);

}

}

**Login page:**

package com.androidbloodbank;

import android.app.ProgressDialog; import android.content.Intent; import android.content.SharedPreferences; import android.net.Uri; import android.os.AsyncTask; import android.os.Bundle;

import android.support.v7.app.AppCompatActivity; import android.util.Log; import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.TextView; import android.widget.Toast;

import com.google.android.gms.appindexing.Action; import com.google.android.gms.appindexing.AppIndex; import com.google.android.gms.appindexing.Thing; import com.google.android.gms.common.api.GoogleApiClient;

import org.apache.http.HttpEntity; import org.apache.http.HttpResponse; import org.apache.http.NameValuePair; import org.apache.http.client.HttpClient; import org.apache.http.client.entity.UrlEncodedFormEntity; import org.apache.http.client.methods.HttpPost; import org.apache.http.impl.client.DefaultHttpClient; import org.apache.http.message.BasicNameValuePair; import org.json.JSONArray; import org.json.JSONException; import org.json.JSONObject;

import java.io.BufferedReader; import java.io.InputStream; import java.io.InputStreamReader; import java.util.ArrayList;

import static com.androidbloodbank.R.id.etname;

public class LoginActivity extends AppCompatActivity {

EditText etusername, etpassword;

String username, password; TextView signup; int k;

SharedPreferences pref;

SharedPreferences.Editor edit;

String name;

String mbno;

String address;

String city;

String bloodgroup;

int code = 0;

String tempstring, line;

Button btnlogin;

ProgressDialog pDialog = null;

InputStream is = null;

Bundle b = new Bundle();

*/\*\**

* *ATTENTION: This was auto-generated to implement the App Indexing API.*
* *See https://g.co/AppIndexing/AndroidStudio for more information.*

*\*/*

private GoogleApiClient client;

@Override

protected void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); setContentView(R.layout.login); etusername = (EditText) findViewById(etname); etpassword = (EditText) findViewById(R.id.etpassword);

btnlogin = (Button) findViewById(R.id.blogin); btnlogin.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

username = etusername.getText().toString(); password = etpassword.getText().toString();

new Loginsync().execute();

}

});

*// ATTENTION: This was auto-generated to implement the App Indexing API. // See https://g.co/AppIndexing/AndroidStudio for more information.* client = new GoogleApiClient.Builder(this).addApi(AppIndex.*API*).build();

}

public void register(View v) {

startActivity(new Intent(LoginActivity.this, SignUpActivity.class));

}

*/\*\**

* *ATTENTION: This was auto-generated to implement the App Indexing API.*
* *See https://g.co/AppIndexing/AndroidStudio for more information.*

*\*/*

public Action getIndexApiAction() {

Thing object = new Thing.Builder()

.setName("Login Page") *// TODO: Define a title for the content shown.*

*// TODO: Make sure this auto-generated URL is correct.*

.setUrl(Uri.*parse*("http://[ENTER-YOUR-URL-HERE]"))

.build();

return new Action.Builder(Action.*TYPE\_VIEW*)

.setObject(object)

.setActionStatus(Action.*STATUS\_TYPE\_COMPLETED*)

.build();

}

@Override public void onStart() { super.onStart();

*// ATTENTION: This was auto-generated to implement the App Indexing API. // See https://g.co/AppIndexing/AndroidStudio for more information.* client.connect();

AppIndex.*AppIndexApi*.start(client, getIndexApiAction());

}

@Override public void onStop() { super.onStop();

*// ATTENTION: This was auto-generated to implement the App Indexing API.*

*// See https://g.co/AppIndexing/AndroidStudio for more information.* AppIndex.*AppIndexApi*.end(client, getIndexApiAction()); client.disconnect();

}

public class Loginsync extends AsyncTask<Void, Void, Void> {

ArrayList<NameValuePair>nameValuePairs = new

ArrayList<NameValuePair>();

@Override

protected void onPreExecute() { *// TODO Auto-generated method stub* super.onPreExecute();

pDialog = new ProgressDialog(LoginActivity.this); pDialog.setMessage("Logging in"); pDialog.setIndeterminate(false); pDialog.setCancelable(false); pDialog.show();

}

@Override

protected Void doInBackground(Void... params) {

nameValuePairs.add(new BasicNameValuePair("username", username)); nameValuePairs.add(new BasicNameValuePair("password", password));

try {

HttpClient hc = new DefaultHttpClient();

HttpPost hp = new

HttpPost("http://192.168.43.68/AndroidBloodBank/login.php"); hp.setEntity(new UrlEncodedFormEntity(nameValuePairs));

HttpResponse hr = hc.execute(hp);

HttpEntity he1 = hr.getEntity();

is = he1.getContent();

Log.*d*("OUTPUT:", is.toString());

} catch (Exception e) {

Log.*e*("Fail 1", e.toString());

Toast.*makeText*(getApplicationContext(), "Invalid IP Address",

Toast.*LENGTH\_LONG*).show();

} return null; }

@Override

protected void onPostExecute(Void result) { *// TODO Auto-generated method stub* super.onPostExecute(result); pDialog.dismiss();

try {

BufferedReader reader = new BufferedReader( new InputStreamReader(is, "ISO-8859-1"), 8);

StringBuilder sb = new StringBuilder(); while ((line = reader.readLine()) != null) { sb.append(line + "");

}

tempstring = sb.toString();

Log.*d*("TAAAAAG", tempstring);

} catch (Exception e) {

*// TODO: handle exception*

} try {

JSONObject jsonResponse = new JSONObject(tempstring); JSONArray jsonMainNode = jsonResponse.optJSONArray("record"); k = 0;

for (int i = 0; i < jsonMainNode.length(); i++) {

JSONObject jsonChildNode = jsonMainNode.getJSONObject(i); if (jsonMainNode.getJSONObject(i) != null) { k++;

name = jsonChildNode.optString("name");

mbno = jsonChildNode.optString("mbno");

address = jsonChildNode.optString("address");

city = jsonChildNode.optString("city");

bloodgroup = jsonChildNode.optString("bloodgroup");

*// code = Integer.parseInt(jsonChildNode.optString("flag"));*

Log.*d*("Name", name);

Log.*d*("MbNo", mbno);

Log.*d*("address", address);

Log.*d*("city", city);

Log.*d*("bloodgroup", bloodgroup);

Log.*d*("k", "" + k);

}

}

} catch (JSONException e) {

Toast.*makeText*(LoginActivity.this, "No Record Found",

Toast.*LENGTH\_SHORT*).show();

} if (k >0) {

pref = getApplicationContext().getSharedPreferences("requestrecord",

*MODE\_PRIVATE*);

edit = pref.edit();

edit.putString("name", name); edit.putString("mbno", mbno); edit.putString("address", address); edit.putString("bloodgroup", bloodgroup); edit.putString("city", city);

edit.commit();

Intent intent = new Intent(LoginActivity.this,

HomePage.class);

b.putString("username", username); intent.putExtra("bundle", b); startActivity(intent);

} else {

Toast.*makeText*(getApplicationContext(),

"unsuccessful", Toast.*LENGTH\_LONG*).show();

}

}

}

}

• **Search donor:**

package com.android.tabfragments;

import android.app.AlertDialog; import android.app.ProgressDialog; import android.content.DialogInterface; import android.os.AsyncTask; import android.os.Bundle; import android.support.v4.app.Fragment; import android.util.Log; import android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import android.widget.AdapterView; import android.widget.ArrayAdapter; import android.widget.ListView; import android.widget.Spinner; import android.widget.Toast;

import com.androidbloodbank.R;

import org.apache.http.HttpEntity; import org.apache.http.HttpResponse; import org.apache.http.NameValuePair; import org.apache.http.client.HttpClient;

import org.apache.http.client.entity.UrlEncodedFormEntity; import org.apache.http.client.methods.HttpPost; import org.apache.http.impl.client.DefaultHttpClient; import org.apache.http.message.BasicNameValuePair; import org.json.JSONArray;

import org.json.JSONException; import org.json.JSONObject;

import java.io.BufferedReader; import java.io.InputStream; import java.io.InputStreamReader; import java.util.ArrayList;

public class SearchDonar extends Fragment {

Spinner sp;

String bloodgroup;

String result = "", line = "";

InputStream is = null; ProgressDialog pDialog = null; int code;

ListView lv;

User u;

ArrayList<String>listitems = new ArrayList<String>(); ArrayList<User>listitemuser = new ArrayList<User>();

@Override public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

View v = inflater.inflate(R.layout.fragment\_search\_donar, container, false);

lv = (ListView) v.findViewById(R.id.listView1); sp = (Spinner) v.findViewById(R.id.spinner2);

ArrayAdapter<CharSequence> adapter =

ArrayAdapter.*createFromResource*(this.getActivity(),

R.array.bloodgroup, android.R.layout.*simple\_spinner\_item*);

adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_ite m*);

sp.setAdapter(adapter);

lv.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

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

Log.*d*("In Itemuser listen---", listitemuser.get(pos).getName());

Log.*d*("In Items listen---", listitems.get(pos));

set(pos);

}

}

);

sp.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() { public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) { listitems.clear();

listitemuser.clear();

if (sp.getSelectedItem().toString().trim().equals("A")) { bloodgroup = "A";

} else if

(sp.getSelectedItem().toString().trim().equals("A+")) { bloodgroup = "A+";

} else if

(sp.getSelectedItem().toString().trim().equals("AB+")) { bloodgroup = "AB+";

} else if

(sp.getSelectedItem().toString().trim().equals("AB-")) { bloodgroup = "AB-";

} else if (sp.getSelectedItem().toString().trim().equals("B-

")) {

bloodgroup = "B-";

} else if

(sp.getSelectedItem().toString().trim().equals("B+")) { bloodgroup = "B+";

} else if (sp.getSelectedItem().toString().trim().equals("O-

")) {

bloodgroup = "O-";

} else if

(sp.getSelectedItem().toString().trim().equals("O+")) { bloodgroup = "O+";

}

new Insert().execute();

}

public void onNothingSelected(AdapterView<?> parent) {

*// Another interface callback*

}

});

return v;

}

private void set(int pos) {

AlertDialog.Builder adb = new AlertDialog.Builder(getActivity()); adb.setTitle(" Contact Information"); adb.setIcon(R.drawable.man);

adb.setMessage("Name:\t" + listitemuser.get(pos).getName() +

"\nAddress:" + listitemuser.get(pos).getAddress() + "\nCOntact No:" + listitemuser.get(pos).getContact()); adb.setPositiveButton(getText(R.string.okay\_dismiss), new DialogInterface.OnClickListener() { public void onClick(DialogInterface dialog, int id) { dialog.dismiss();

} }); adb.show();

}

class Insert extends AsyncTask<Void, Void, Void> {

ArrayList<NameValuePair>nameValuePairs = new

ArrayList<NameValuePair>();

@Override

protected void onPreExecute() { *// TODO Auto-generated method stub* super.onPreExecute();

pDialog = new ProgressDialog(getActivity()); pDialog.setMessage("Searching"); pDialog.setIndeterminate(false); pDialog.setCancelable(false); pDialog.show();

}

@Override

protected Void doInBackground(Void... params) { if (pDialog.isShowing()) { pDialog.dismiss(); } try {

nameValuePairs.add(new BasicNameValuePair("bloodgroup", bloodgroup));

HttpClient hc = new DefaultHttpClient();

HttpPost hp = new

HttpPost("http://192.168.43.68/AndroidBloodBank/searchdonar.php");

hp.setEntity(new UrlEncodedFormEntity(nameValuePairs));

HttpResponse hr = hc.execute(hp);

HttpEntity he1 = hr.getEntity();

is = he1.getContent();

Log.*d*("OUTPUT:", is.toString());

} catch (Exception e) {

Log.*e*("Fail 1", e.toString());

} return null; }

@Override

protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid);

try {

BufferedReader reader = new BufferedReader(new

InputStreamReader(is, "ISO-8859-1"), 8);

StringBuilder sb = new StringBuilder(); while ((line = reader.readLine()) != null) { sb.append(line + "");

}

result = sb.toString();

Log.*d*("TAAAAAG", result);

} catch (Exception e) {

*// TODO: handle exception*

} try {

JSONObject jsonResponse = new JSONObject(result); JSONArray jsonMainNode = jsonResponse.optJSONArray("search\_donar");

for (int i = 0; i < jsonMainNode.length(); i++) { JSONObject jsonChildNode = jsonMainNode.getJSONObject(i);

if (jsonMainNode.getJSONObject(i) != null) {

String name = jsonChildNode.optString("name");

String donar = jsonChildNode.optString("donar");

String address = jsonChildNode.optString("address");

String contact = jsonChildNode.optString("mbno");

String city = jsonChildNode.optString("city");

*/\* Log.d("Name", name);*

*Log.d("Mb No", contact);*

*Log.d("Address", address);*

*Log.d("City", city);*

*\*/*

u = new User(name, address, city, contact);

String outPut = name;

if (donar.equals("true")) { listitemuser.add(u);

Log.*d*("Name---->", u.getName());

Log.*d*("Contact---->", u.getContact()); Log.*d*("Address---->", u.getAddress());

listitems.add(outPut);

}

*// employeeList.add(createEmployee("notice", outPut));*

ArrayAdapter<String> adapter = new ArrayAdapter<String> (getActivity(), android.R.layout.*simple\_list\_item\_1*, listitems); lv.setAdapter(adapter);

}

}

} catch (JSONException e) {

Toast.*makeText*(getActivity(), "No Record Found",

Toast.*LENGTH\_SHORT*).show();

}

}

}

}

• **Request page:**

mport java.io.BufferedReader; import java.io.IOException; import java.io.InputStream; import java.io.InputStreamReader; import java.text.SimpleDateFormat; import java.util.ArrayList; import java.util.Date;

import static android.content.Context.*MODE\_PRIVATE*;

public class BloodBank extends Fragment {

View v;

String bloodgroup, bloodtype;

String result = "", line = "";

InputStream is = null; ProgressDialog pDialog = null; int code; int quantity;

String reqquantity;

Button b;

Spinner sp, sp1;

EditText etquantity;

TextView aplus, aminus, bplus, bminus, abplus, abminus, oplus, ominus;

SimpleDateFormat df;

String formattedDate;

SharedPreferences pref;

SharedPreferences.Editor edit;

String name, address, mbno, city;

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

v = inflater.inflate(R.layout.fragment\_blood\_bank, container, false);

df = new SimpleDateFormat("yyyy/MM/dd"); formattedDate = df.format(new Date());

pref =getActivity().getSharedPreferences("requestrecord", *MODE\_PRIVATE*); name = pref.getString("name", ""); mbno = pref.getString("mbno", ""); address = pref.getString("address", ""); city = pref.getString("city", "");

etquantity = (EditText) v.findViewById(R.id.etquantity); aplus = (TextView) v.findViewById(R.id.Aplusquantity); aminus = (TextView) v.findViewById(R.id.Aminusquantity); bplus = (TextView) v.findViewById(R.id.Bplusquantity); bminus = (TextView) v.findViewById(R.id.Bminusquantity); abplus = (TextView) v.findViewById(R.id.ABplusquantity); abminus = (TextView) v.findViewById(R.id.ABminusquantity); oplus = (TextView) v.findViewById(R.id.Oplusquantity); ominus = (TextView) v.findViewById(R.id.Ominusquantity); sp = (Spinner) v.findViewById(R.id.spbloodgrp); sp1 = (Spinner) v.findViewById(R.id.spbloodtype); b = (Button) v.findViewById(R.id.brequest); ArrayAdapter<CharSequence> adapter =

ArrayAdapter.*createFromResource*(getActivity(),

R.array.bloodgroup, android.R.layout.*simple\_spinner\_item*);

adapter.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item*)

;

sp.setAdapter(adapter);

ArrayAdapter<CharSequence> adapter1 =

ArrayAdapter.*createFromResource*(getActivity(), R.array.bloodtype, android.R.layout.*simple\_spinner\_item*);

adapter1.setDropDownViewResource(android.R.layout.*simple\_spinner\_dropdown\_item* );

sp1.setAdapter(adapter1);

sp.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() { public void onItemSelected(AdapterView<?> parent, View view, int pos, long id)

{

if (sp.getSelectedItem().toString().trim().equals("A")) { bloodgroup = "A";

} else if

(sp.getSelectedItem().toString().trim().equals("A+")) { bloodgroup = "A+";

} else if

(sp.getSelectedItem().toString().trim().equals("AB+")) { bloodgroup = "AB+";

} else if (sp.getSelectedItem().toString().trim().equals("AB-

")) {

bloodgroup = "AB-";

} else if (sp.getSelectedItem().toString().trim().equals("B-

")) {

bloodgroup = "B-";

} else if

(sp.getSelectedItem().toString().trim().equals("B+")) { bloodgroup = "B+";

} else if (sp.getSelectedItem().toString().trim().equals("O-

")) { bloodgroup = "O-";

} else if

(sp.getSelectedItem().toString().trim().equals("O+")) { bloodgroup = "O+";

}

}

public void onNothingSelected(AdapterView<?> parent) {

*// Another interface callback*

}

});

sp1.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() { public void onItemSelected(AdapterView<?> parent, View view, int pos, long id)

{

if (sp1.getSelectedItem().toString().trim().equals("Red Blood Cell")) { bloodtype = "Red Blood Cell";

} else if

(sp1.getSelectedItem().toString().trim().equals("Plasma")) { bloodtype = "Plasma";

} else if

(sp1.getSelectedItem().toString().trim().equals("Platelets")) { bloodtype = "Platelets";

}

}

public void onNothingSelected(AdapterView<?> parent) {

*// Another interface callback*

}

});

b.setOnClickListener(new View.OnClickListener() { @Override

public void onClick(View view) {

reqquantity = etquantity.getText().toString();

Log.*d*("Name", name);

Log.*d*("MbNo", mbno);

Log.*d*("address", address);

Log.*d*("city", city);

Log.*d*("bloodgroup", bloodgroup);

Log.*d*("reqquantity", reqquantity); Log.*d*("Date", formattedDate); new Update().execute();

}

});

new Show().execute();

return v;

}

class Show extends AsyncTask<Void, Void, Void> {

@Override

protected void onPreExecute() { *// TODO Auto-generated method stub* super.onPreExecute();

pDialog = new ProgressDialog(getActivity()); pDialog.setMessage("Searching"); pDialog.setIndeterminate(false); pDialog.setCancelable(false); pDialog.show();

}

@Override

protected Void doInBackground(Void... voids) { if (pDialog.isShowing()) { pDialog.dismiss(); } try {

HttpClient hc = new DefaultHttpClient();

HttpPost hp = new

HttpPost("http://192.168.43.68/AndroidBloodBank/bloodbank.php");

HttpResponse hr = hc.execute(hp);

HttpEntity he1 = hr.getEntity();

is = he1.getContent();

Log.*d*("OUTPUT:", is.toString());

} catch (Exception e) {

Log.*e*("Fail 1", e.toString());

} return null; } @Override

protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid);

try {

BufferedReader reader = new BufferedReader(new

InputStreamReader(is, "ISO-8859-1"), 8);

StringBuilder sb = new StringBuilder(); while ((line = reader.readLine()) != null) { sb.append(line + "");

}

result = sb.toString();

Log.*d*("TAAAAAG", result);

} catch (Exception e) {

*// TODO: handle exception*

} try {

JSONObject jsonResponse = new JSONObject(result);

JSONArray jsonMainNode = jsonResponse.optJSONArray("quantity");

for (int i = 0; i < jsonMainNode.length(); i++) {

JSONObject jsonChildNode = jsonMainNode.getJSONObject(i); if (jsonMainNode.getJSONObject(i) != null) { String bloodgroup = jsonChildNode.optString("bloodgroup");

quantity = Integer.*parseInt*(jsonChildNode.optString("quantity"));

if (bloodgroup.equals("A+")) {

String aplusquantity = jsonChildNode.optString("quantity");

Log.*d*("A+", aplusquantity); aplus.setText(aplusquantity);

} else if (bloodgroup.equals("A-")) { String aminusquantity = jsonChildNode.optString("quantity");

Log.*d*("A-", aminusquantity); aminus.setText(aminusquantity);

} else if (bloodgroup.equals("B+")) { String bplusquantity = jsonChildNode.optString("quantity");

Log.*d*("B+", bplusquantity); bplus.setText(bplusquantity);

} else if (bloodgroup.equals("B-")) { String bminusquantity = jsonChildNode.optString("quantity");

Log.*d*("B-", bminusquantity); bminus.setText(bminusquantity);

} else if (bloodgroup.equals("AB+")) { String abplusquantity =

jsonChildNode.optString("quantity");

Log.*d*("AB+", jsonChildNode.optString("AB+")); abplus.setText(abplusquantity);

} else if (bloodgroup.equals("AB-")) { String abminusquantity =

jsonChildNode.optString("quantity");

Log.*d*("AB-", jsonChildNode.optString("AB-")); abminus.setText(abminusquantity);

} else if (bloodgroup.equals("O+")) { String obplusquantity =

jsonChildNode.optString("quantity");

Log.*d*("O+", jsonChildNode.optString("O+")); oplus.setText(obplusquantity);

} else if (bloodgroup.equals("O-")) { String obminusquantity =

jsonChildNode.optString("quantity");

Log.*d*("O-", jsonChildNode.optString("O-")); ominus.setText(obminusquantity);

}

}

}

} catch (JSONException e) {

Toast.*makeText*(getActivity(), "No Record Found",

Toast.*LENGTH\_SHORT*).show();

}

}

}

class Update extends AsyncTask<Void, Void, Void> { ArrayList<NameValuePair>nameValuePairs = new

ArrayList<NameValuePair>();

@Override protected void onPreExecute() { *// TODO Auto-generated method stub* super.onPreExecute();

pDialog = new ProgressDialog(getActivity()); pDialog.setMessage("Updating"); pDialog.setIndeterminate(false); pDialog.setCancelable(false); pDialog.show();

}

@Override

protected Void doInBackground(Void... voids) { if (pDialog.isShowing()) { pDialog.dismiss(); } try {

nameValuePairs.add(new BasicNameValuePair("name", name)); nameValuePairs.add(new BasicNameValuePair("address", address)); nameValuePairs.add(new BasicNameValuePair("mbno", mbno)); nameValuePairs.add(new BasicNameValuePair("city", city)); nameValuePairs.add(new BasicNameValuePair("bloodgroup", bloodgroup)); nameValuePairs.add(new BasicNameValuePair("bloodtype", bloodtype)); nameValuePairs.add(new BasicNameValuePair("quantity", reqquantity)); nameValuePairs.add(new BasicNameValuePair("date", formattedDate)); nameValuePairs.add(new BasicNameValuePair("status", "" + 0));

HttpClient hc = new DefaultHttpClient();

HttpPost hp = new

HttpPost("http://192.168.43.68/AndroidBloodBank/request.php"); hp.setEntity(new UrlEncodedFormEntity(nameValuePairs));

HttpResponse hr = hc.execute(hp);

HttpEntity he1 = hr.getEntity();

is = he1.getContent();

Log.*d*("OUTPUT:", is.toString());

} catch (Exception e) {

Log.*e*("Fail 1", e.toString());

} return null; }

@Override

protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid);

try {

BufferedReader reader = new BufferedReader(new

InputStreamReader(is, "ISO-8859-1"), 8);

StringBuilder sb = new StringBuilder(); while ((line = reader.readLine()) != null) { sb.append(line + "");

}

result = sb.toString();

Log.*d*("TAAAAAG", result);

} catch (Exception e) {

*// TODO: handle exception*

} try {

JSONObject json\_data = new JSONObject(result + "");

code = (json\_data.getInt("code")); Log.*d*("COOODE", "" + code);

} catch (Exception e) {

*// TODO: handle exception*

Log.*d*("ERRRRRRRRR", "" + e.toString());

} if (code == 1) {

Toast.*makeText*(getActivity(), "Request sent.Wait for Admin

Approval" , Toast.*LENGTH\_LONG*).show();

try { is.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

* **Profile page:**
* public class MyProfile extends Fragment {

View v;

InputStream is = null; int code;

ProgressDialog pDialog = null;

String name, city, mbno, address, emailid, username, password, donar, gender, imgpath;

EditText etname, etmobileno, etemailid, etgender, etaddress, etcity, etusername, etpassword;

ImageView iv;

String result = "", line, img = ""; int REQUEST\_CAMERA = 0, SELECT\_FILE = 1;

String userChoosenTask;

Button bupdate;

CheckBox cbdonar;

Intent intent;

Bundle b;

@Override public View onCreateView(LayoutInflater inflater, final ViewGroup container, Bundle savedInstanceState) {

v = inflater.inflate(R.layout.fragment\_my\_profile, container, false);

iv = (ImageView) v.findViewById(R.id.ivupload); bupdate = (Button) v.findViewById(R.id.bupdate); cbdonar = (CheckBox) v.findViewById(R.id.cbdonar); etname = (EditText) v.findViewById(R.id.etname); etcity = (EditText) v.findViewById(R.id.etcity); etaddress = (EditText) v.findViewById(R.id.etaddress); etemailid = (EditText) v.findViewById(R.id.etemailid); etmobileno = (EditText) v.findViewById(R.id.etmbno); etpassword = (EditText) v.findViewById(R.id.etpassword); etgender = (EditText) v.findViewById(R.id.etgender); intent = getActivity().getIntent(); b = intent.getBundleExtra("bundle"); username = b.getString("username"); new Insert().execute();

bupdate.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

name = etname.getText().toString(); gender = etgender.getText().toString(); emailid = etemailid.getText().toString(); mbno = etmobileno.getText().toString(); address = etaddress.getText().toString(); city = etcity.getText().toString(); password = etpassword.getText().toString(); if (cbdonar.isChecked()) donar = "true"; else

donar = "false";

new Update().execute();

}

});

return v;

}

public class Update extends AsyncTask<Void, Void, Void> {

@Override

protected void onPreExecute() {

*// TODO Auto-generated method stub* super.onPreExecute();

pDialog = new ProgressDialog(getActivity()); pDialog.setMessage("Signing"); pDialog.setIndeterminate(false); pDialog.setCancelable(false); pDialog.show();

}

@Override

protected Void doInBackground(Void... params) {

ArrayList<NameValuePair> nameValuePairs = new

ArrayList<NameValuePair>();

try {

HttpClient hc = new DefaultHttpClient();

nameValuePairs.add(new BasicNameValuePair("tag", "update")); nameValuePairs.add(new BasicNameValuePair("name", name)); nameValuePairs.add(new BasicNameValuePair("gender", gender)); nameValuePairs.add(new BasicNameValuePair("emailid", emailid)); nameValuePairs.add(new BasicNameValuePair("mbno", mbno)); nameValuePairs.add(new BasicNameValuePair("address", address)); nameValuePairs.add(new BasicNameValuePair("city", city)); nameValuePairs.add(new BasicNameValuePair("password", password));

nameValuePairs.add(new BasicNameValuePair("donar", donar)); nameValuePairs.add(new BasicNameValuePair("username", username));

HttpPost hp = new

HttpPost("http://192.168.43.68/AndroidBloodBank/myprofile.php"); hp.setEntity(new UrlEncodedFormEntity(nameValuePairs));

HttpResponse hr = hc.execute(hp);

HttpEntity he1 = hr.getEntity();

is = he1.getContent();

Log.*d*("OUTPUT:", is.toString());

} catch (Exception e) {

Log.*e*("Fail 1", e.toString());

*//Toast.makeText(getApplicationContext(), "Invalid IP Address",*

*Toast.LENGTH\_LONG).show();*

} return null; }

@Override

protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid); pDialog.dismiss();

try {

BufferedReader reader = new BufferedReader(new

InputStreamReader(is, "iso-8859-1"), 8);

StringBuilder sb = new StringBuilder();

while ((line = reader.readLine()) != null) { sb.append(line + "\n");

}

*/\* SignUp.this.runOnUiThread(new Runnable() { public void run() { // show message here*

*} });\*/* result = sb.toString();

Log.*d*("pass 2", result);

} catch (Exception e) {

Log.*e*("Fail 2", e.toString());

}

try {

JSONObject json\_data = new JSONObject(result + "");

code = (json\_data.getInt("code")); Log.*d*("COOODE", "" + code);

} catch (Exception e) {

*// TODO: handle exception*

Log.*d*("ERRRRRRRRR", "" + e.toString());

} if (code == 1) {

Toast.*makeText*(getActivity(), "Record Updated successfully" + result, Toast.*LENGTH\_LONG*).show();

etname.setText(""); etgender.setText(""); etemailid.setText(""); etmobileno.setText(""); etaddress.setText(""); etcity.setText(""); etpassword.setText("");

try { is.close();

} catch (IOException e) {

e.printStackTrace();

}

} else {

Toast.*makeText*(getActivity(), "Record Not Updated successfully"

+ result, Toast.*LENGTH\_LONG*).show();

}

}

}

public class Insert extends AsyncTask<Void, Void, Void> {

ArrayList<NameValuePair>nameValuePairs = new

ArrayList<NameValuePair>();

@Override

protected void onPreExecute() { *// TODO Auto-generated method stub* super.onPreExecute();

pDialog = new ProgressDialog(getActivity()); pDialog.setMessage("Signing"); pDialog.setIndeterminate(false); pDialog.setCancelable(false); pDialog.show();

}

@Override

protected Void doInBackground(Void... params) {

try {

HttpClient hc = new DefaultHttpClient(); nameValuePairs.add(new BasicNameValuePair("tag", "select")); nameValuePairs.add(new BasicNameValuePair("username", username));

HttpPost hp = new

HttpPost("http://192.168.43.68/AndroidBloodBank/myprofile.php"); hp.setEntity(new UrlEncodedFormEntity(nameValuePairs));

HttpResponse hr = hc.execute(hp);

HttpEntity he1 = hr.getEntity();

is = he1.getContent();

Log.*d*("OUTPUT:", is.toString());

} catch (Exception e) {

Log.*e*("Fail 1", e.toString());

*//Toast.makeText(getApplicationContext(), "Invalid IP Address",*

*Toast.LENGTH\_LONG).show();*

} return null; }

void imgpath(final String path) {

Thread background = new Thread() {

public void run() { try {

*/\* URL url = new URL(path); //you can write here any link*

*URLConnection ucon = url.openConnection();*

*InputStream is = ucon.getInputStream();*

*\*/*

is = new java.net.URL(path).openStream(); boolean supported = is.markSupported();

final BitmapFactory.Options options = new BitmapFactory.Options(); options.inJustDecodeBounds = true;

BufferedInputStream buffer = new BufferedInputStream(is); buffer.reset();

Bitmap bm = BitmapFactory.*decodeStream*(buffer, null, options); if (bm != null) { iv.setImageBitmap(bm);

Log.*d*("Bitmap", bm.toString());

}

} catch (Exception e) {

e.printStackTrace();

}

} };

background.start();

}

@Override

protected void onPostExecute(Void aVoid) { super.onPostExecute(aVoid); pDialog.dismiss(); try {

BufferedReader reader = new BufferedReader(new

InputStreamReader(is, "iso-8859-1"), 8);

StringBuilder sb = new StringBuilder();

while ((line = reader.readLine()) != null) { sb.append(line + "\n");

}

*/\* SignUp.this.runOnUiThread(new Runnable() { public void run() { // show message here*

*} });\*/* result = sb.toString();

Log.*d*("pass 2", result);

} catch (Exception e) {

Log.*e*("Fail 2", e.toString());

}

try {

JSONObject jsonResponse = new JSONObject(result); JSONArray jsonMainNode = jsonResponse.optJSONArray("search\_user");

for (int i = 0; i < jsonMainNode.length(); i++) {

JSONObject jsonChildNode = jsonMainNode.getJSONObject(i); if (jsonMainNode.getJSONObject(i) != null) { name = jsonChildNode.optString("name"); gender = jsonChildNode.optString("gender"); emailid = jsonChildNode.optString("emailid"); mbno = jsonChildNode.optString("mbno"); address = jsonChildNode.optString("address"); city = jsonChildNode.optString("city"); password = jsonChildNode.optString("password"); donar = jsonChildNode.optString("donar"); imgpath = jsonChildNode.optString("image");

imgpath = "http:// 192.168.43.80/AndroidBloodBank/" + imgpath;

Log.*d*("ImagePath", imgpath);

imgpath(imgpath); etname.setText(name); etgender.setText(gender); etemailid.setText(emailid); etmobileno.setText(mbno); etaddress.setText(address); etcity.setText(city); etpassword.setText(password);

if (donar.equals("true")) { cbdonar.setChecked(true); } else { cbdonar.setChecked(false);

}

}

}

} catch (JSONException e) {

Toast.*makeText*(getActivity(), "No Record Found",

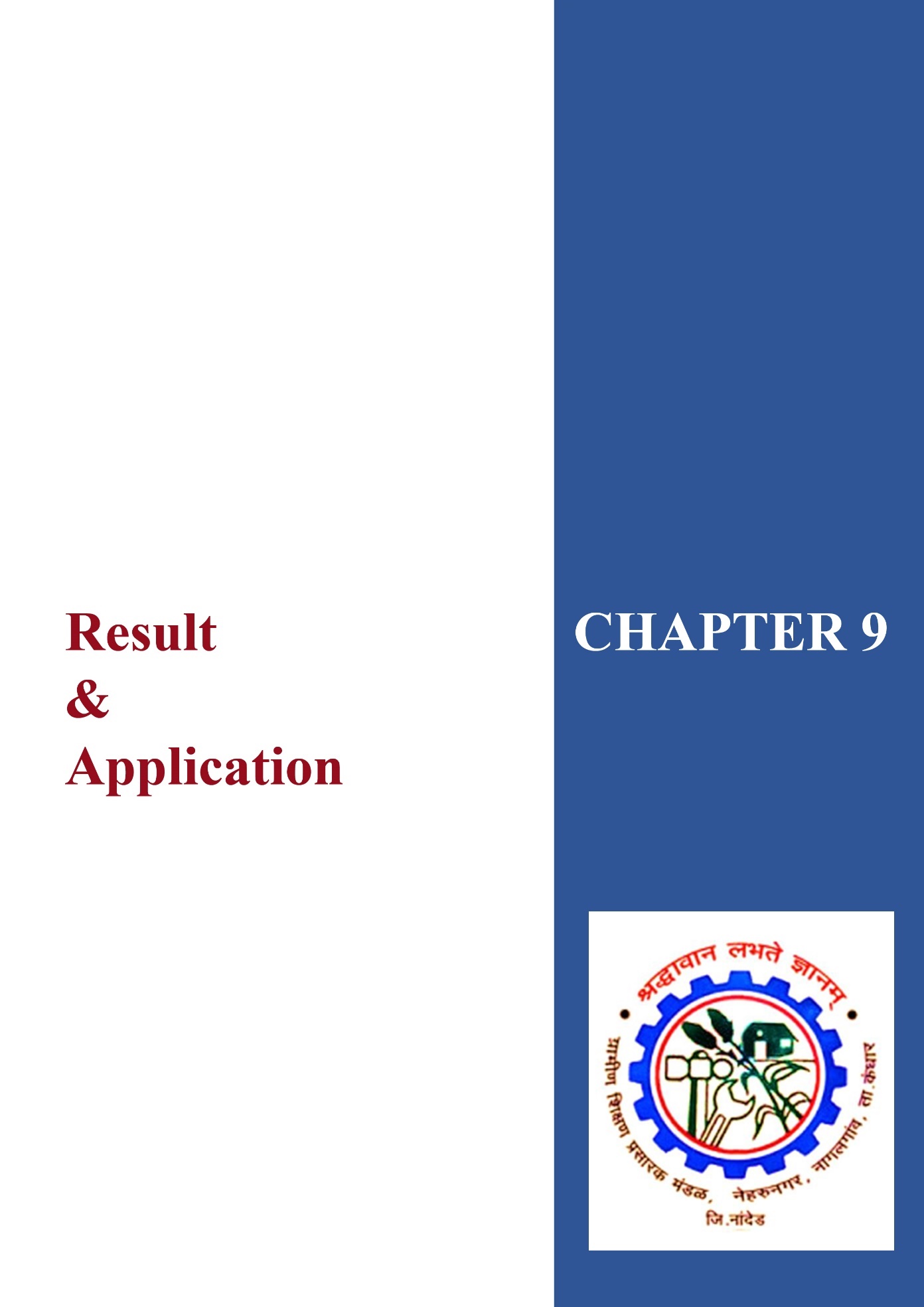
Toast.*LENGTH\_SHORT*).show();

}

}

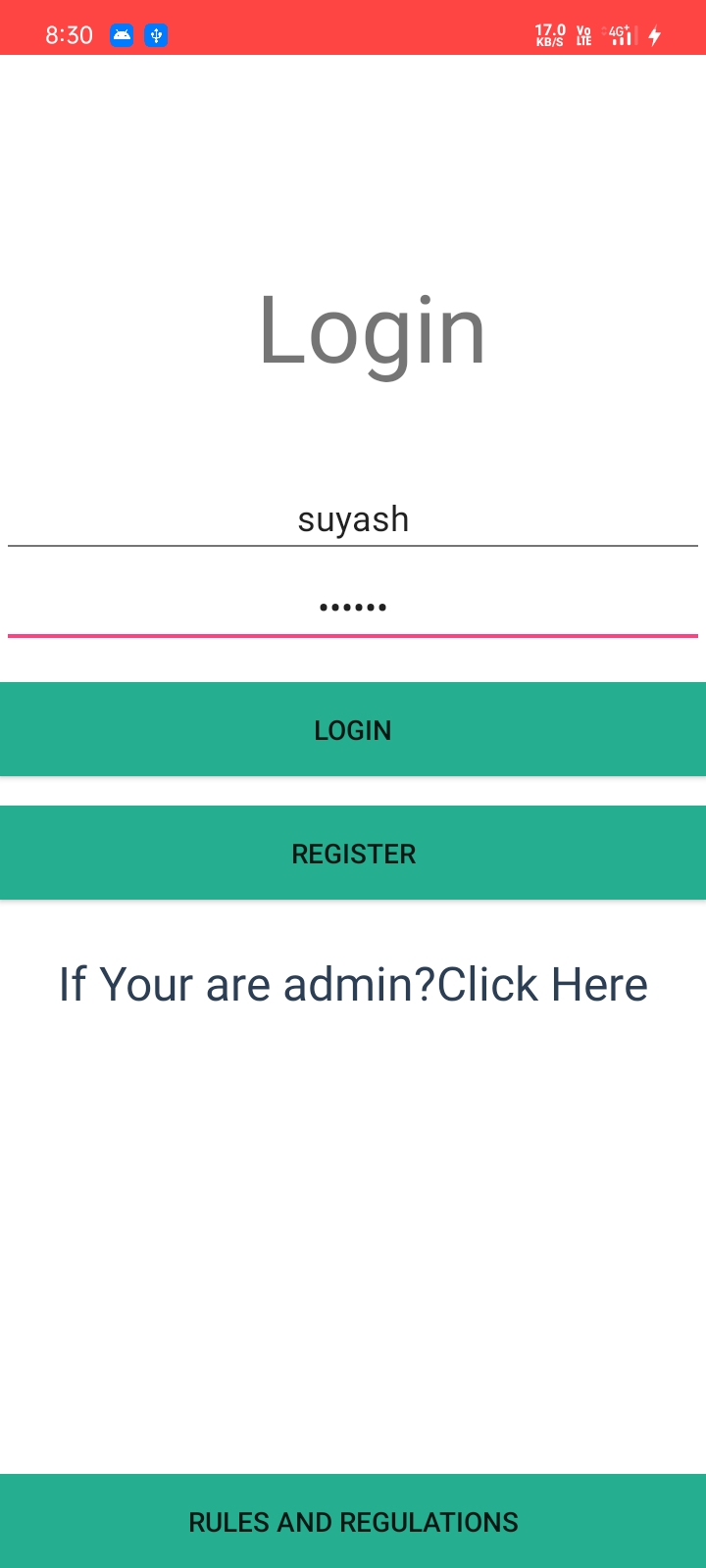
}

}

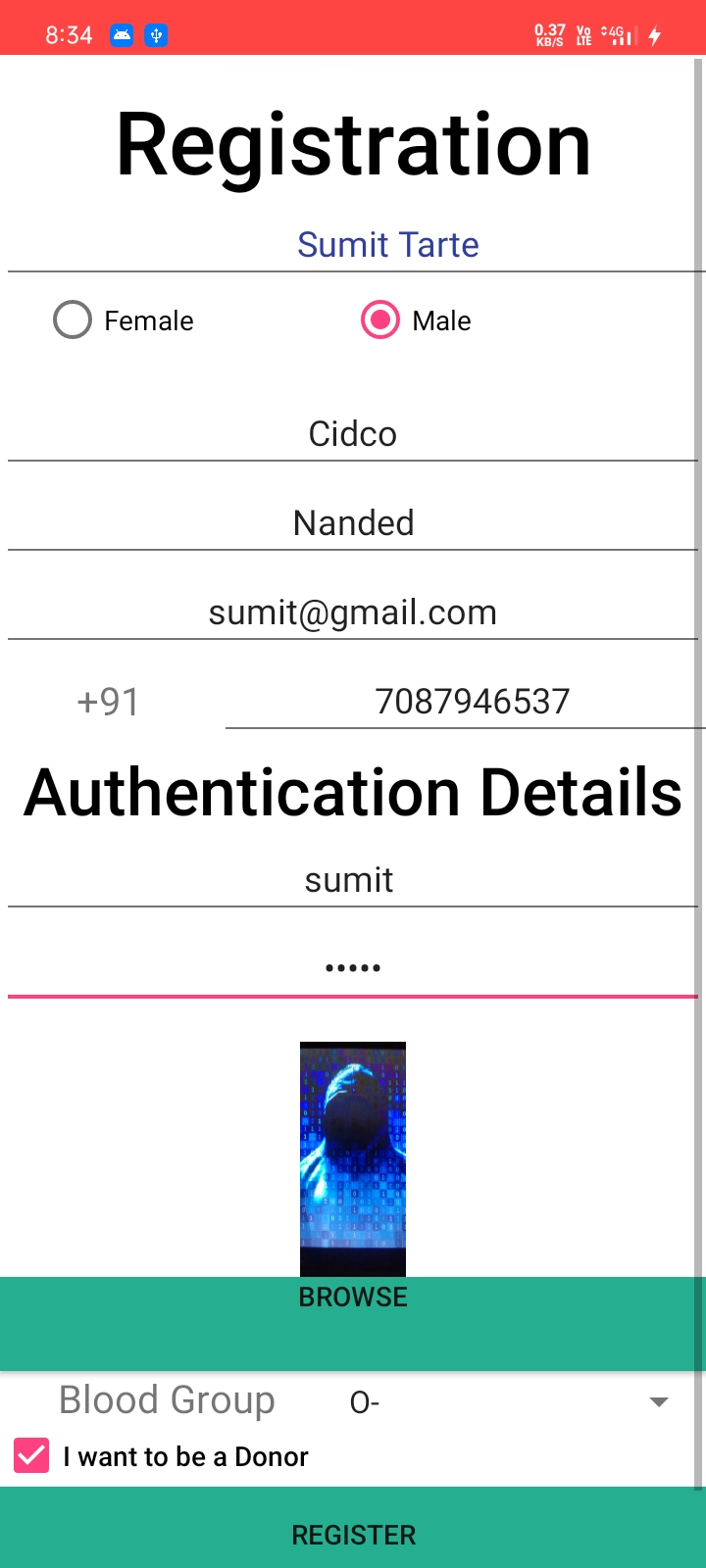


***Chapter 9***

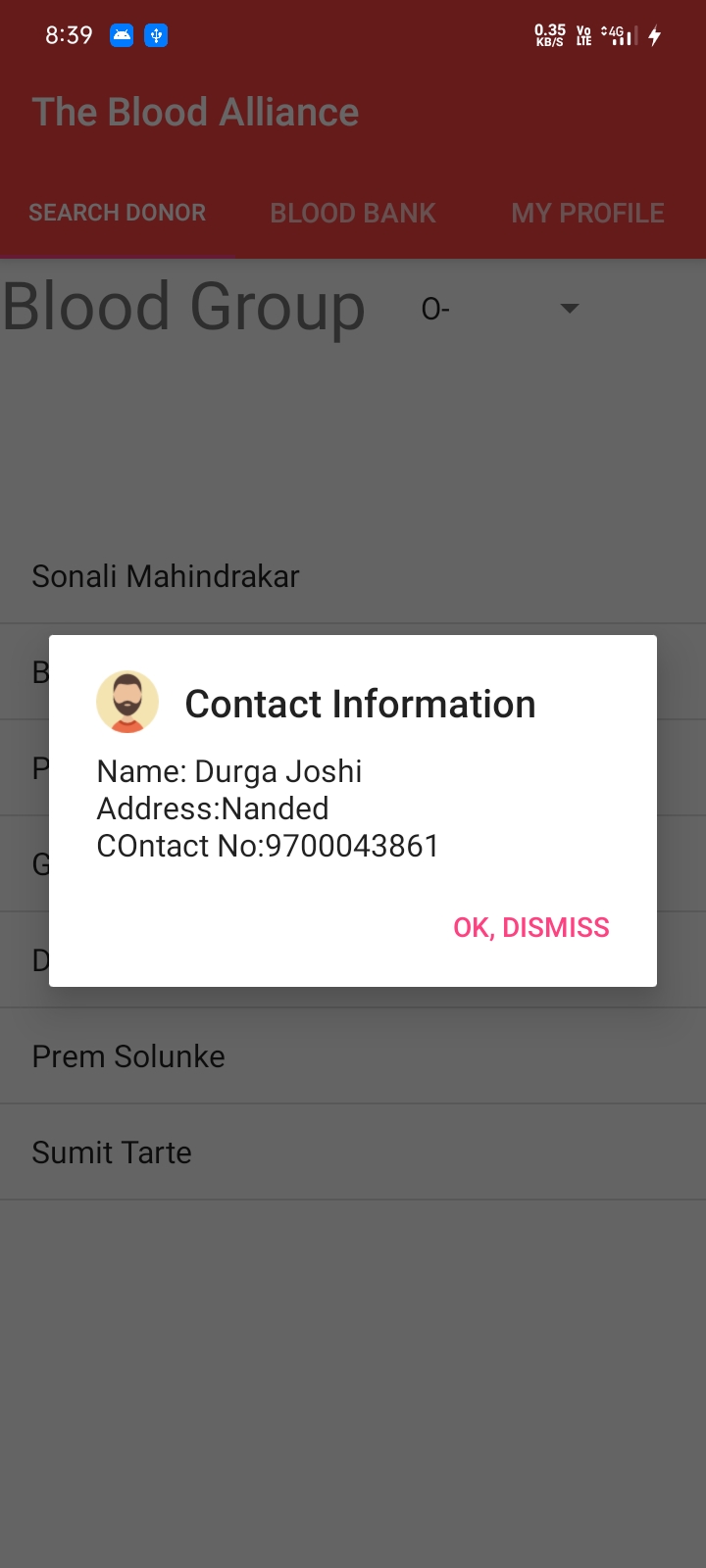
## • Result and Application



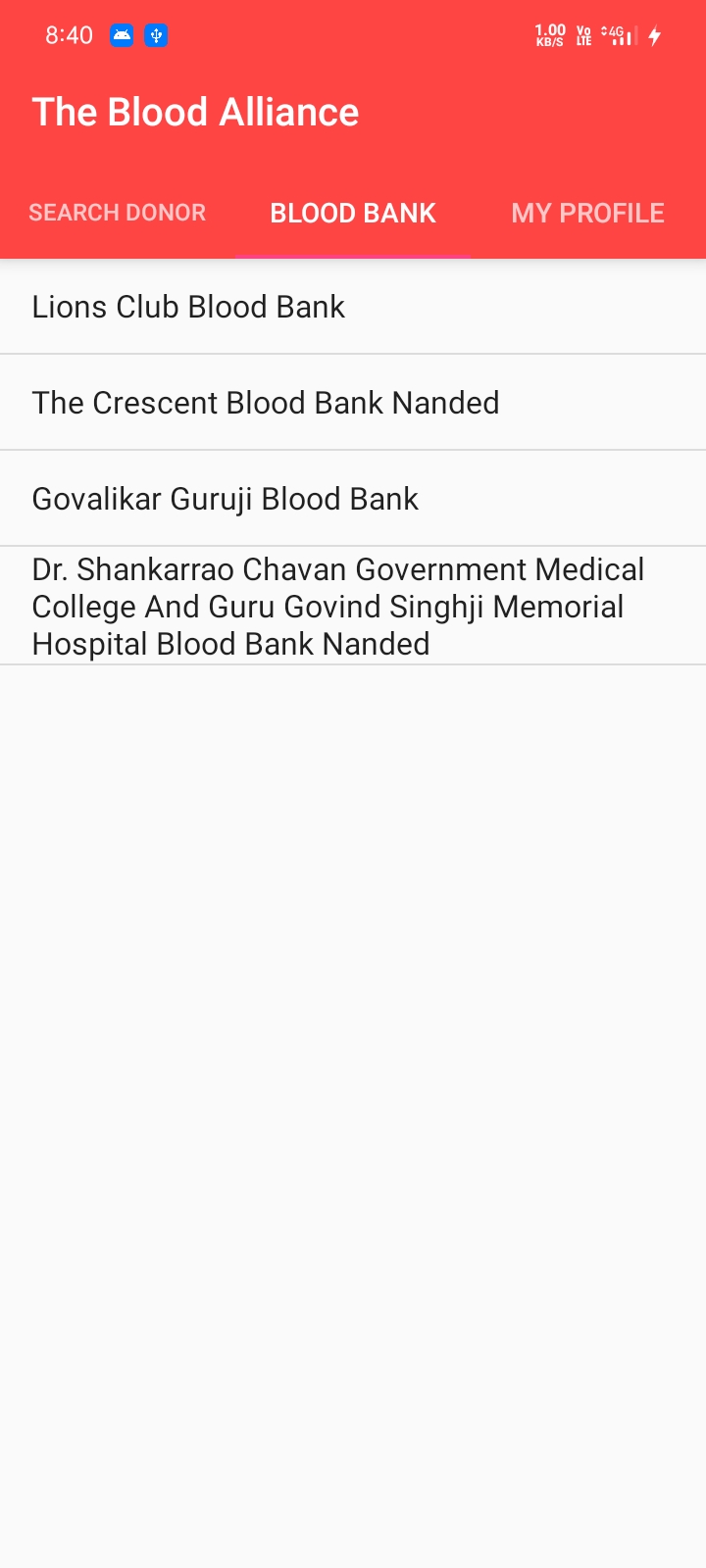
**Figure 9.1: Here Application is opened & Login, Registration, Admin & Rules and Regulations module is visible**



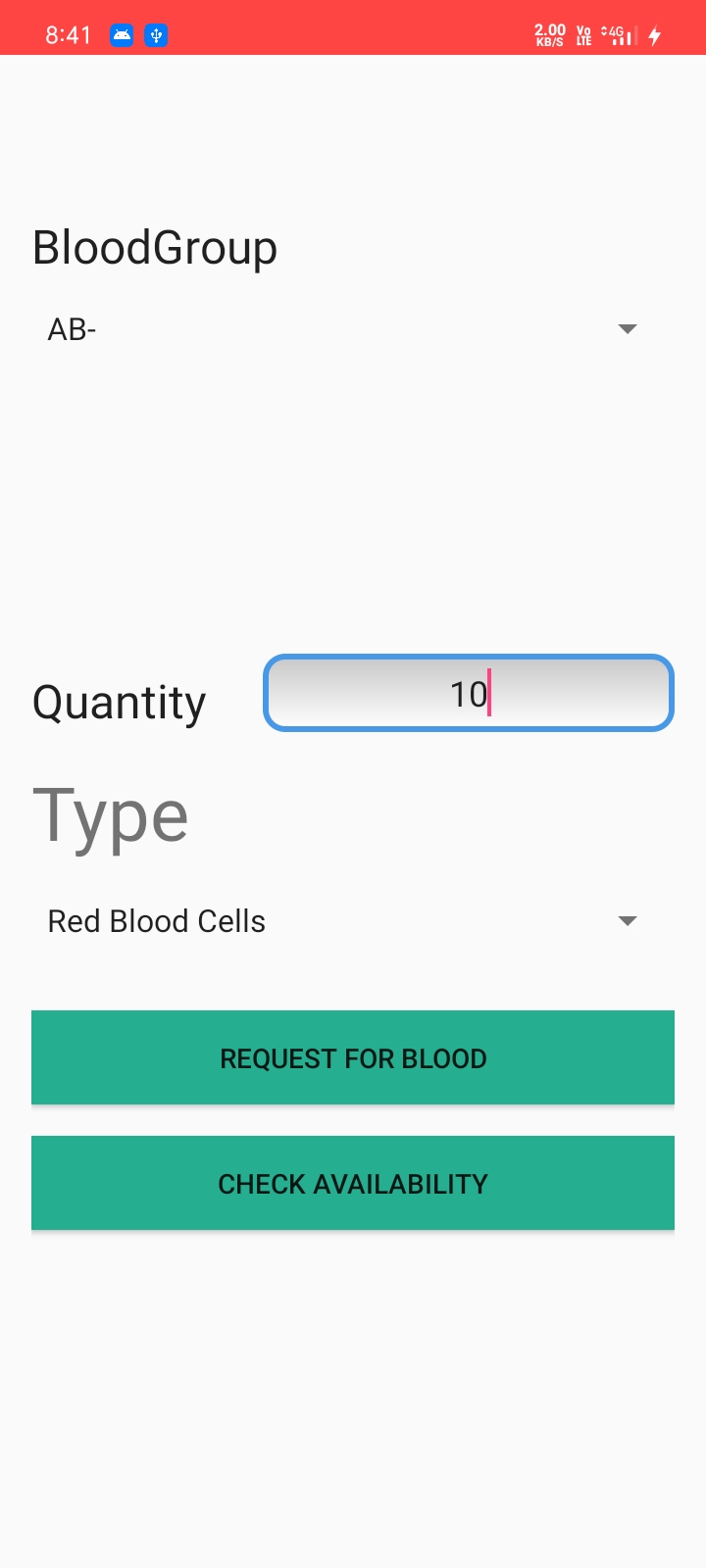
**Figure 9.2: Register the donor details**



**Figure 9.3: Search Donor by selecting the required blood group and user details shown**



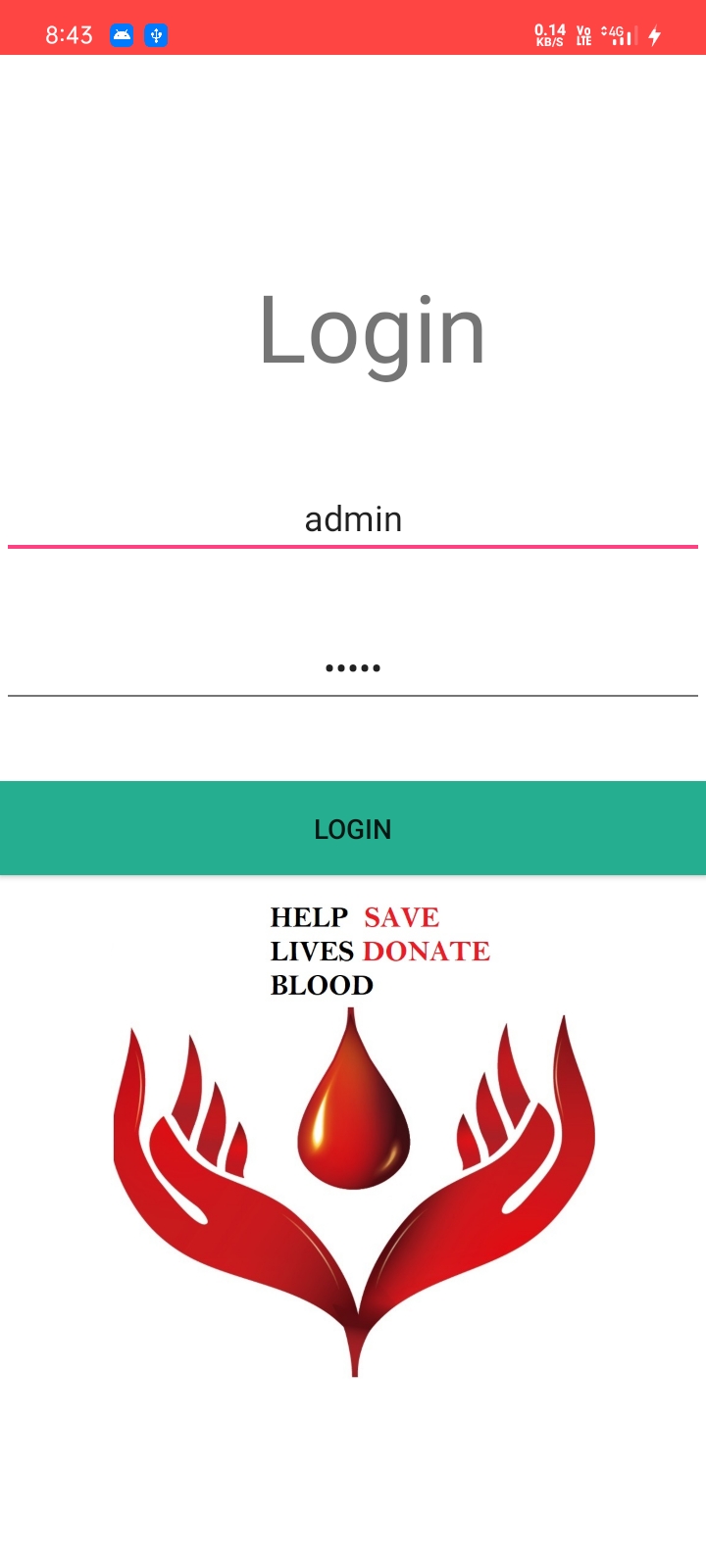
**Figure 9.4: Available blood banks**



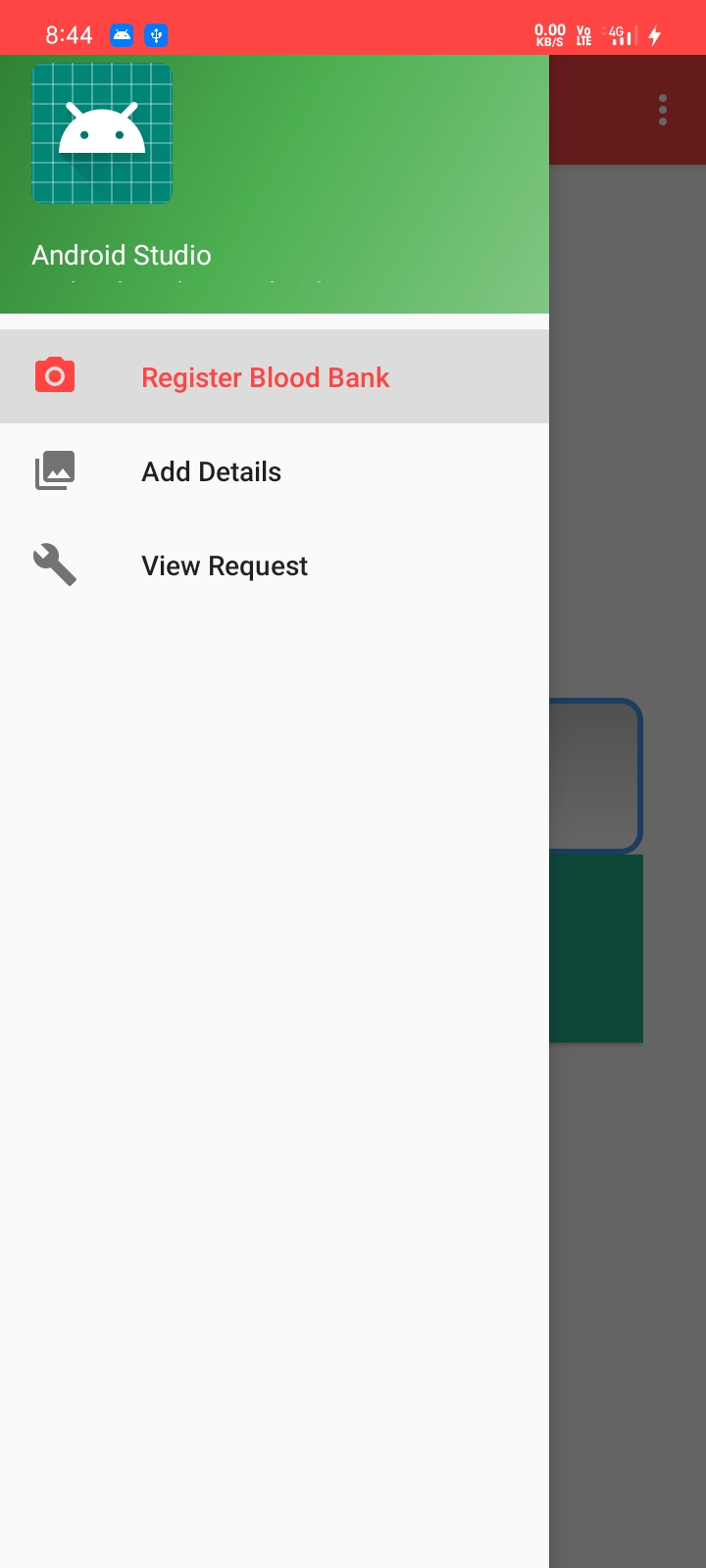
**Figure 9.5: Successfully request sent to the blood bank**



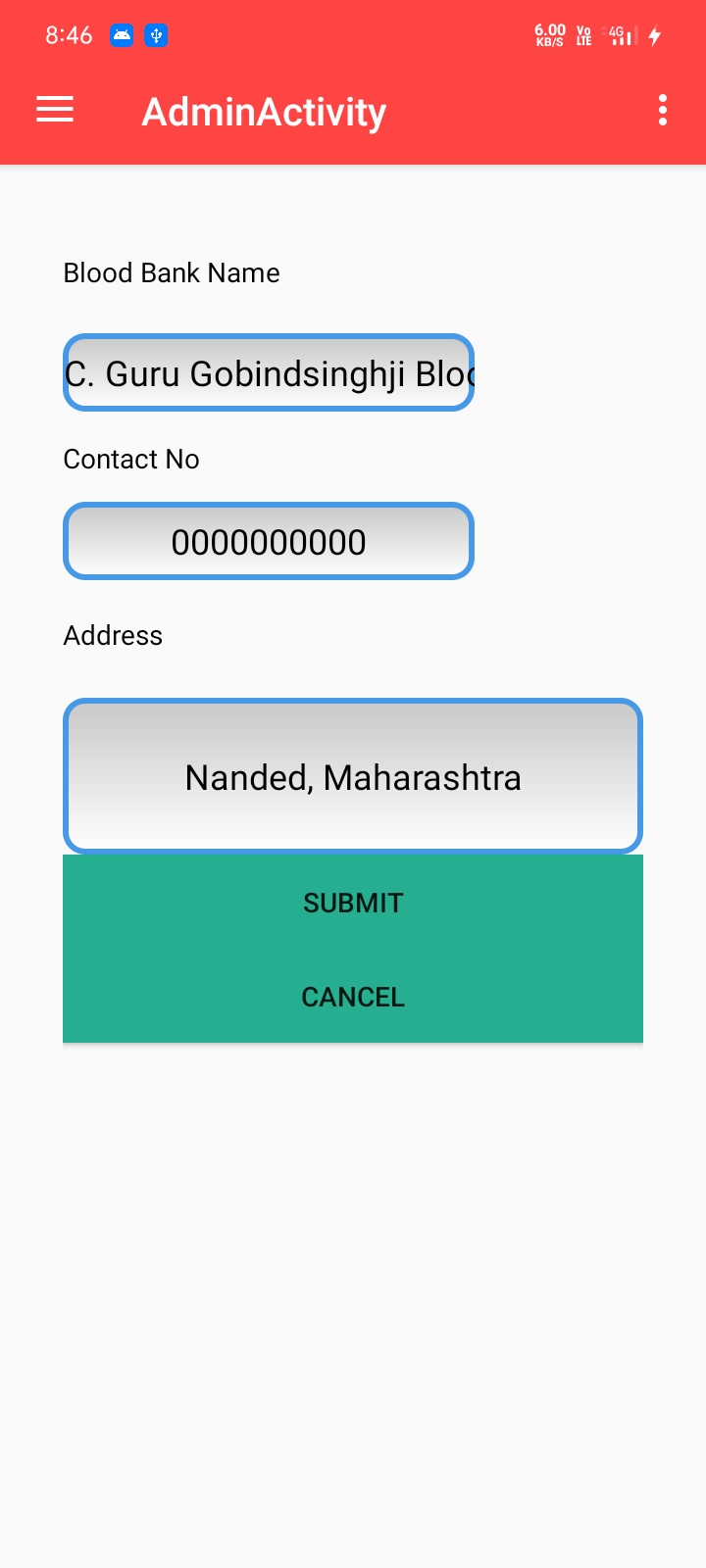
**Figure 9.6: User profile, user can also update to his credentials.**



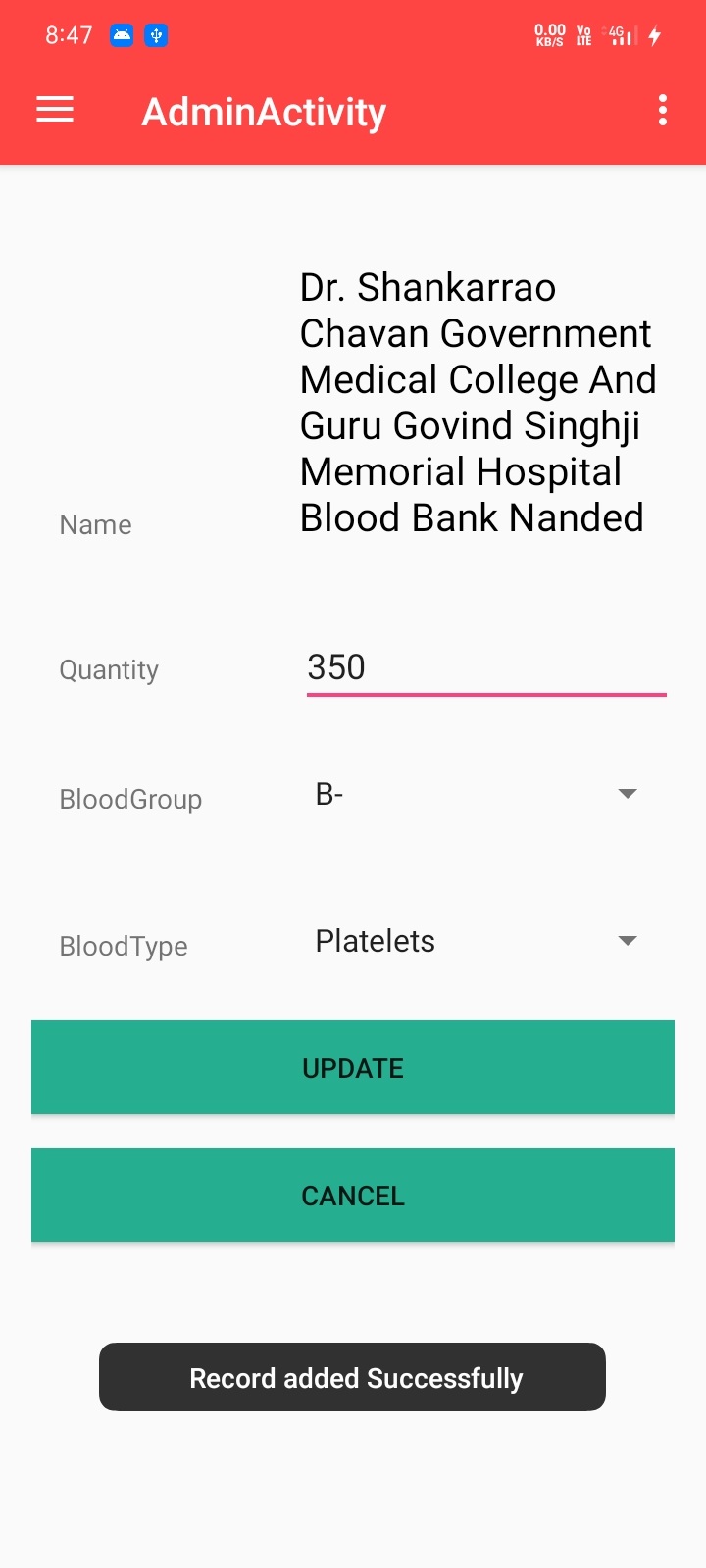
**Figure 9.7: Admin login**



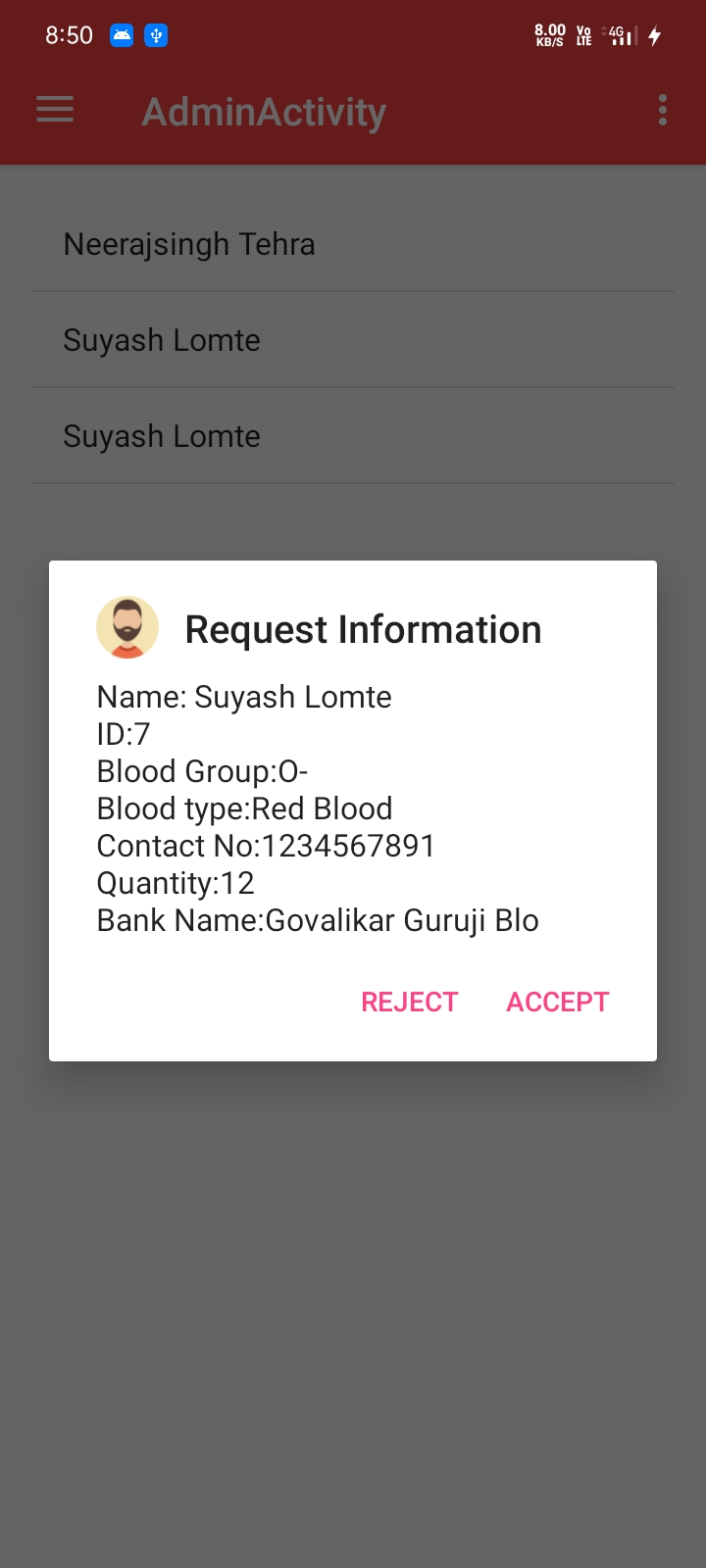
**Figure 9.8: Admin Activities**



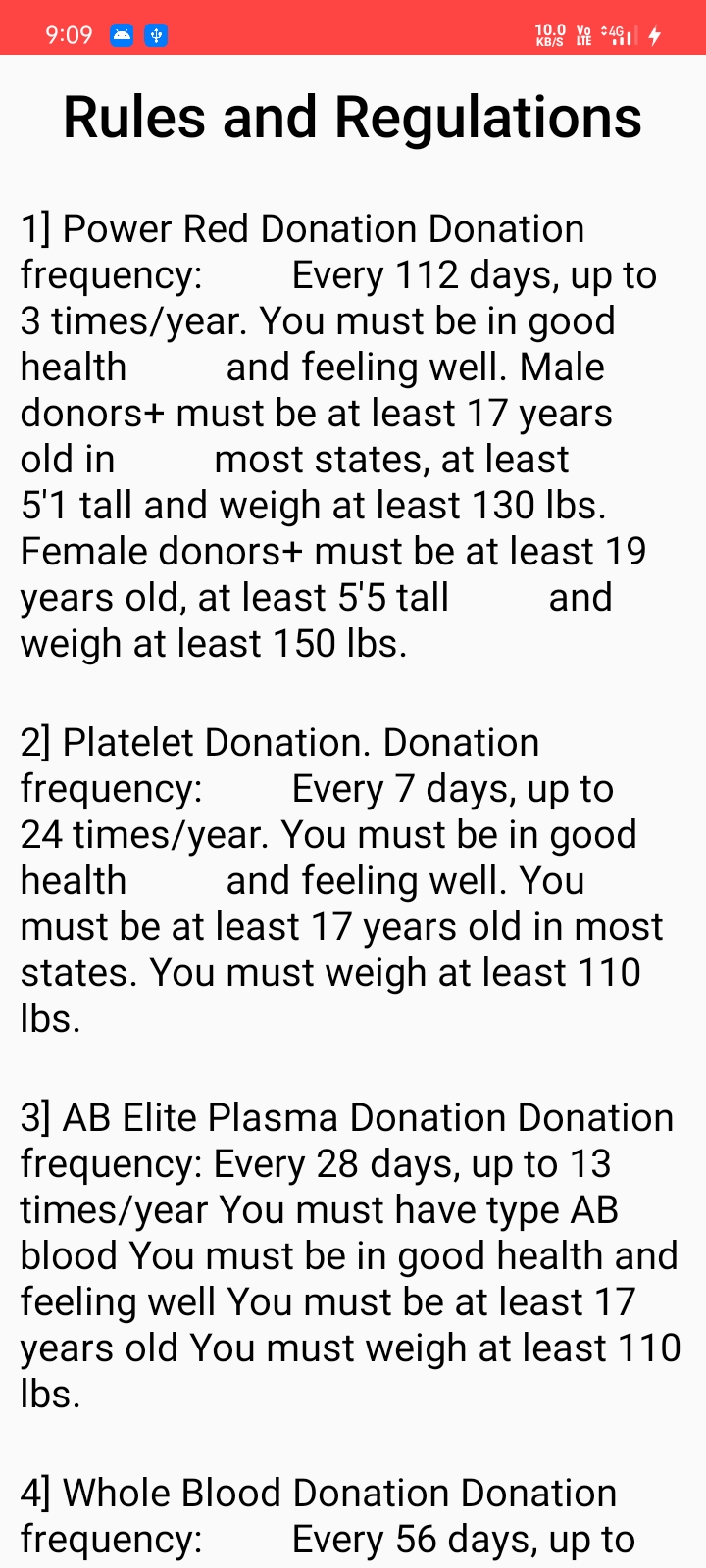
**Figure 9.9: Register a Blood Bank**



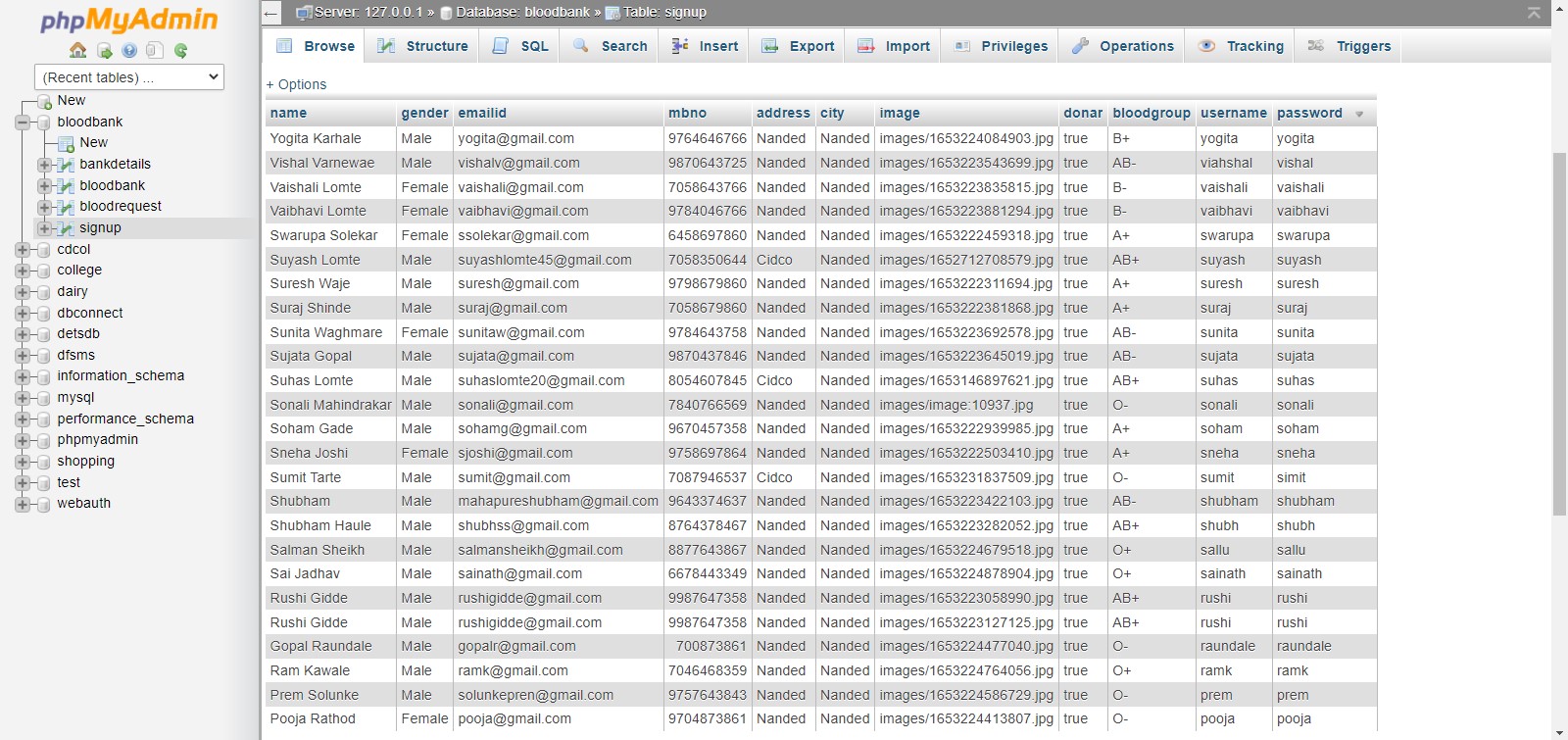
**Figure 9.10: Adding bank details**



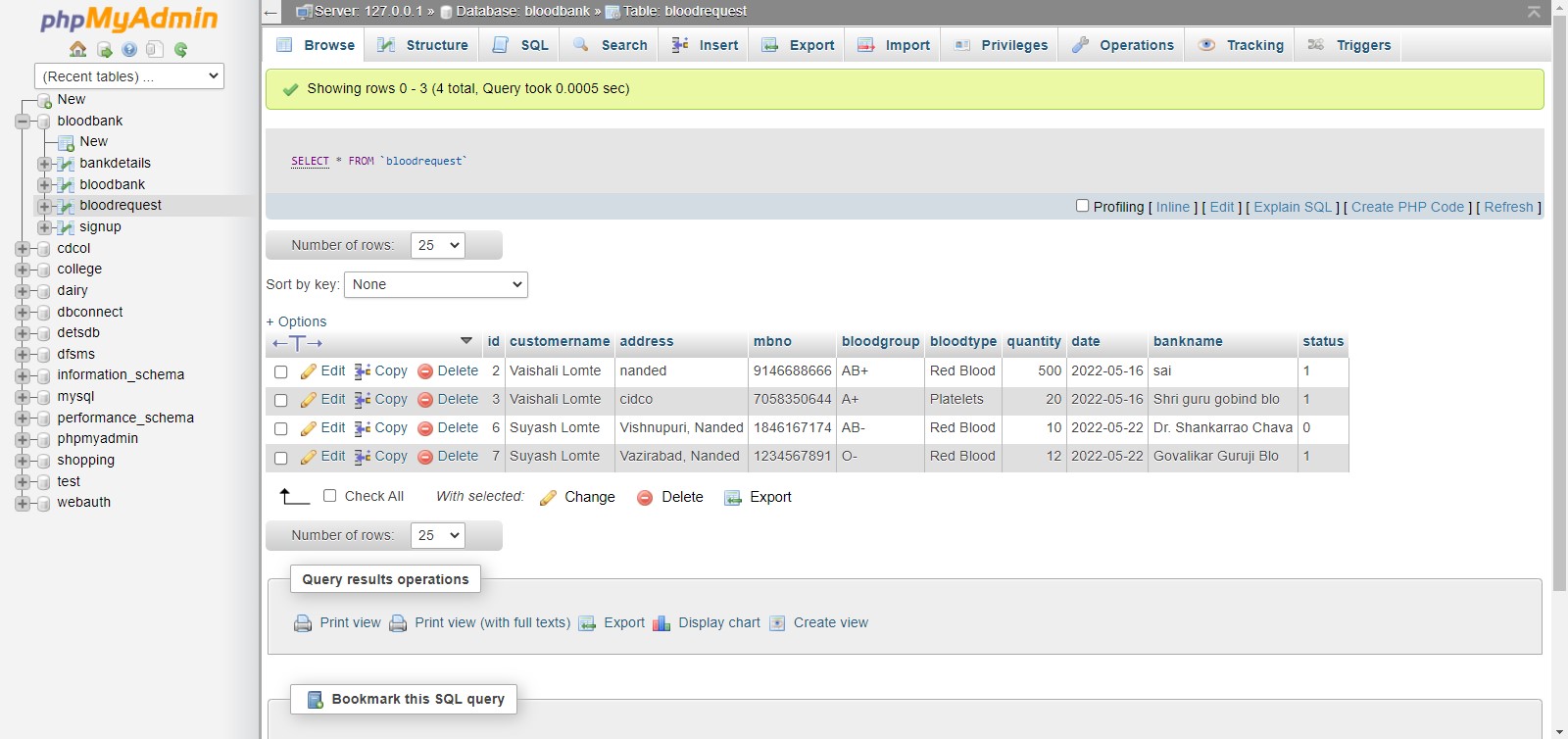
**Figure 9.11: Blood requests**



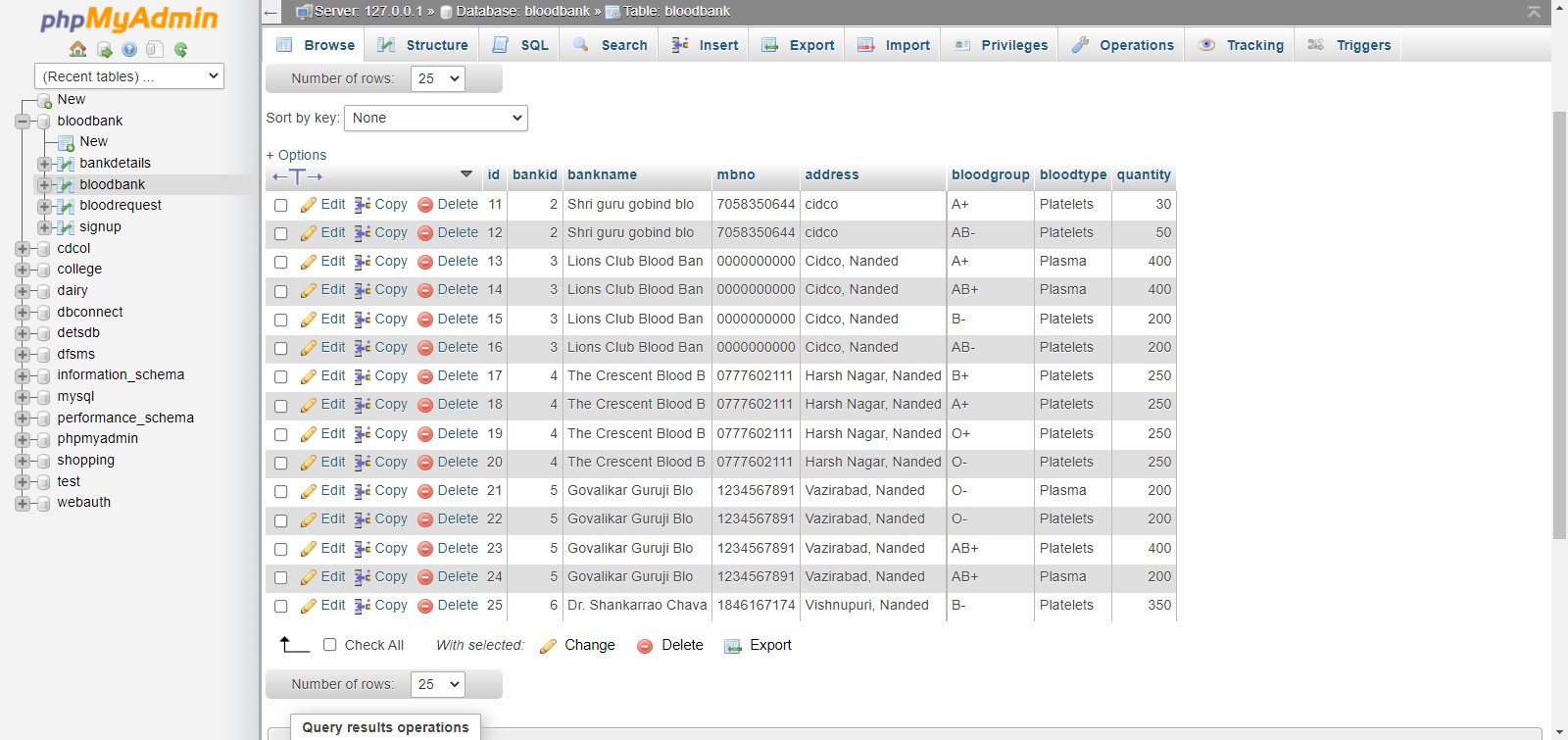
**Figure 9.12: Rules and Regulations module**



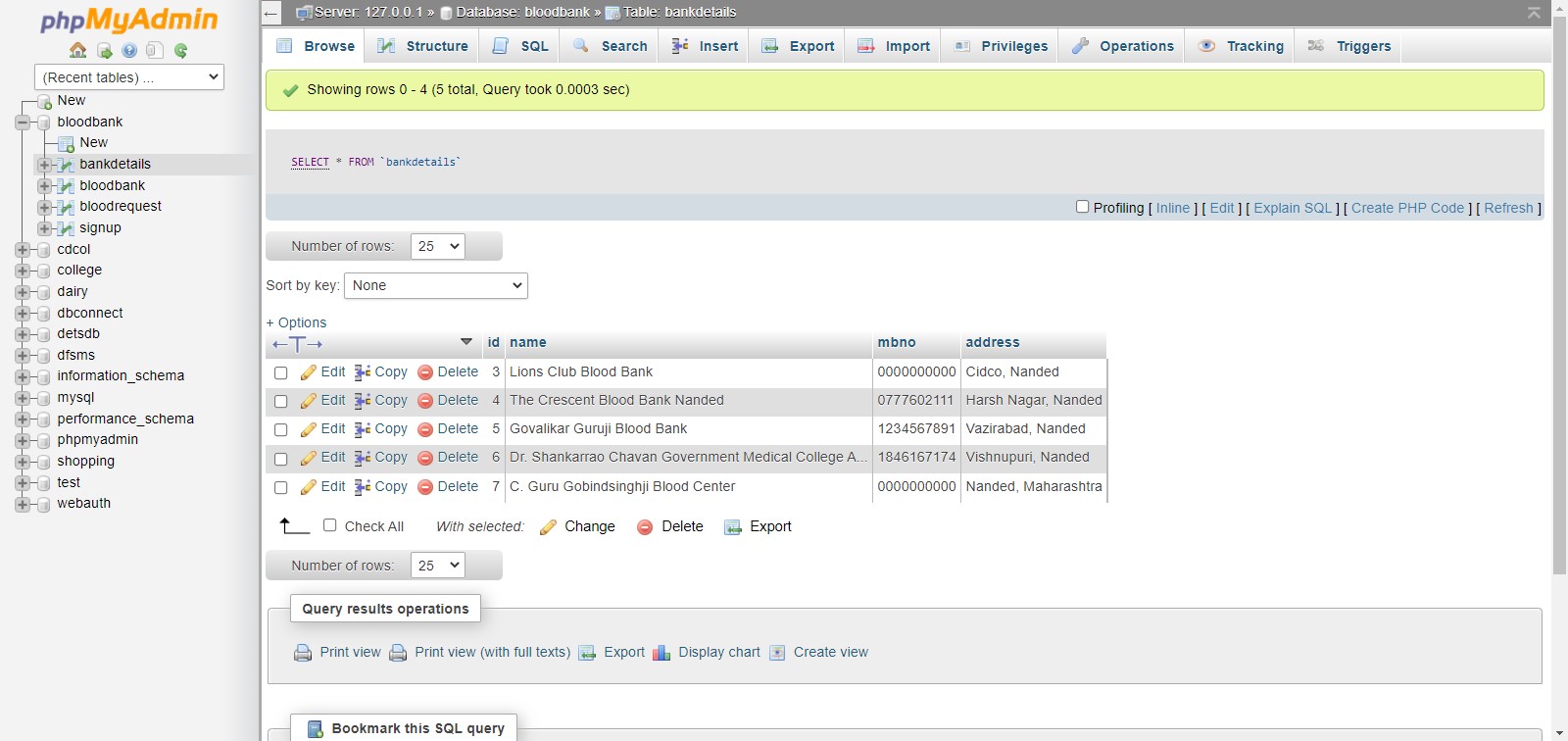
**Figure 9.13: Signup Database**



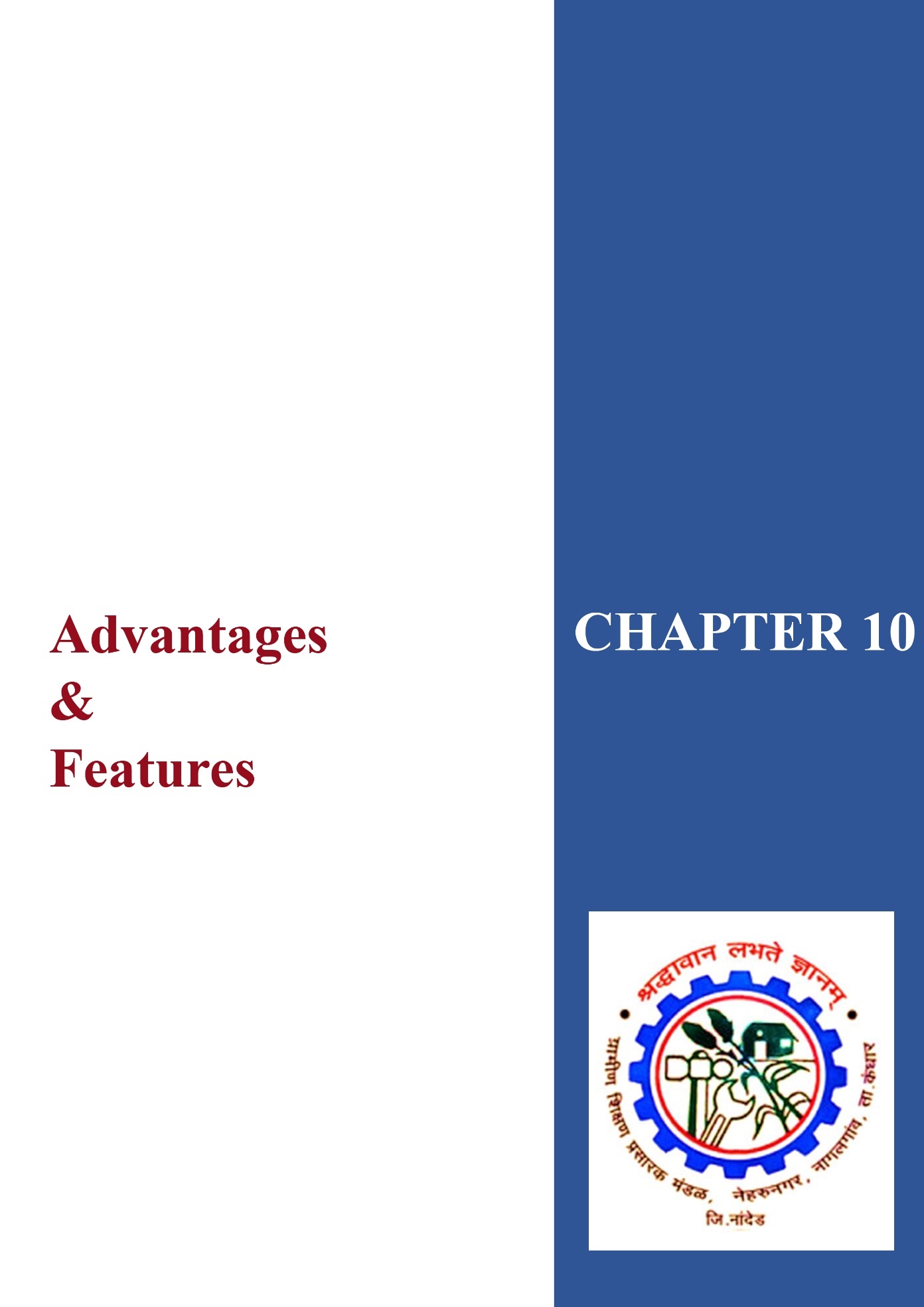
**Figure 9.14: Blood Requests**



**Figure 9.15: Blood Bank Database**



**Figure 9.16: Blood Bank Details**



***Chapter 10***

## ADVANTAGES AND FEATURES

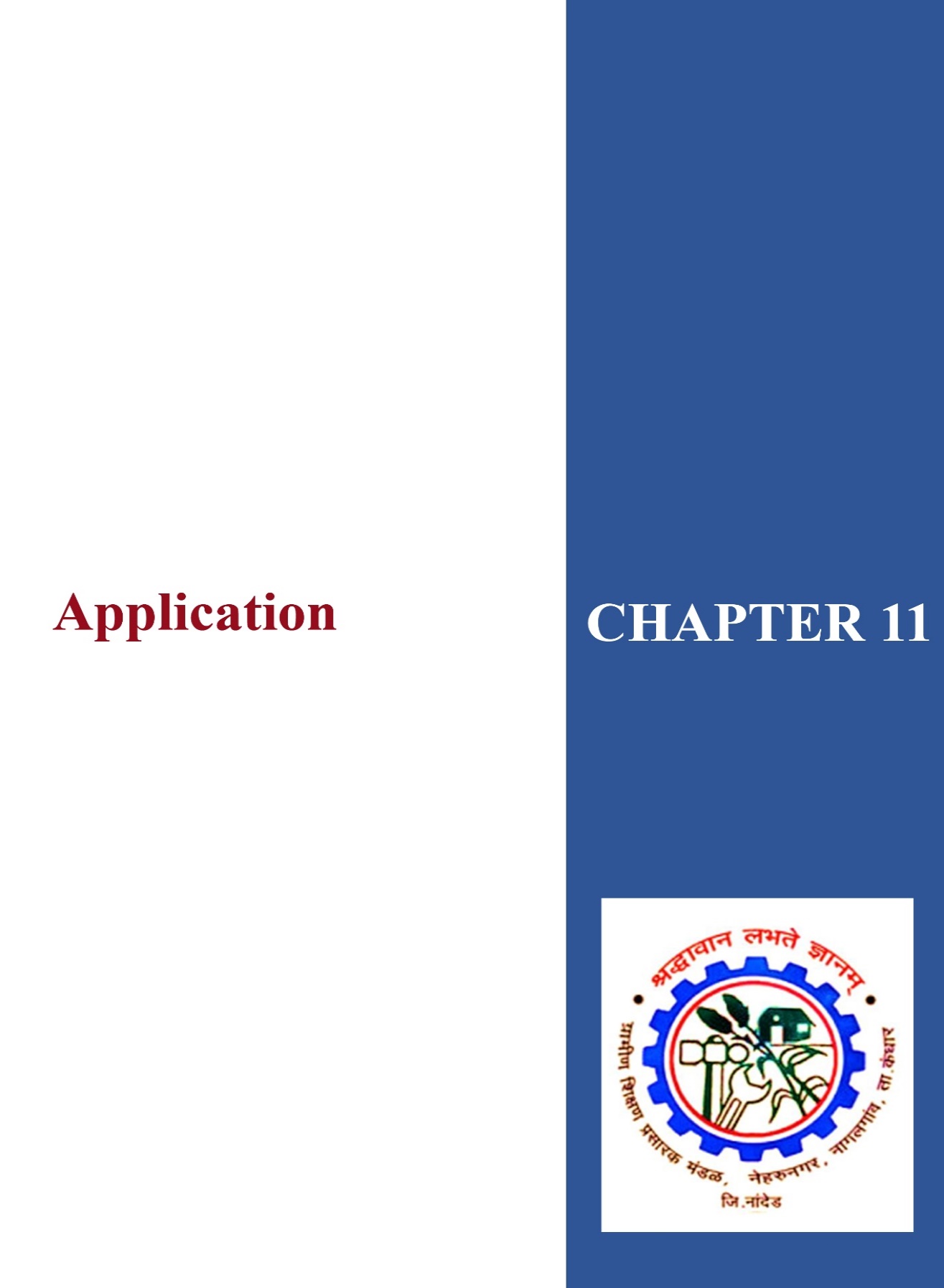
**10.1 Advantages:**

o This project has a login page which allows only the registered user to login and thereby preventing unauthorized access.

* This system can be used to view all the donor details and accordingly select the right donor.
* The android mobile user will be able make quick decision in selecting a donor. o Usage of this application will greatly reduce time in selecting the right donor.

**10.2 Features:**

* Donor Registration and Blood Collection
* Component Separation and Accounting
* Transfusion Reaction Detail
* Blood Donor Detail
* Collection of Record
* User Access Control
* Request to blood bank



***Chapter 11***

## APPLICATIONS

* The people in need of blood can search for the donors by giving their blood group and city name.
* It is very flexible and user friendly.
* The person’s time and work is reduced very much which prevails in the present system.
* Easy and Helpful.
* The people are not limited to receive or provide services in working hours of the branch only; he is serviced 24 hours a day, 7 days of week and 365 days of the year.



**CONCLUSION**

This project acts as an important role in saving life of human beings and which is also its main aim. The project The Blood Alliance system is developed so that users can view the information about registered blood donors such as name, address, and other such personal information along with their details of blood group and other medical information of donor. The project also has a login page where in the user is required to register and only then can view the availability of blood and may also register to donate blood if he/she wishes to. This project requires internet access and thus there is a disadvantage of internet failure. Thus, this application helps to select the right donor online instantly using medical details along with the blood group. The main aim of developing this application is to reduce the time to a great extent that is spent in searching for the right donor and the availability of blood required. Thus, this application provides the required information in no time and also helps in quicker decision making.

## FUTURE SCOPE

There is a scope of further improvement of the system. The system can be expanded with the following ideas:

* It is recommended to develop a system which is not limited to district only but can be used in other areas.
* We aim at including as many blood banks as possible and also as many donors as possible.

At present, the blood bank can only see the list of donors, but in future we will also include features by which blood banks can call, SMS as well as send an e-mail to the donor.

## REFERENCES

**Sites Referred:**

* http://www.codeproject.com/Articles/102065/Android-A-beginner-s-guide
* http://mobile.dzone.com/articles/fundamentals-android-tutorial
* http://mobile.tutsplus.com/tutorials/android/java-tutorial/
* http://developer.android.com/guide/index.html
* http://stackoverflow.android/codingmania.com/guide/index.html
* <http://w3schools.android.com/guide/index.html>
* <https://www.slideshare.net/JawharAli/full-report-on-blood-bank-management-system>
* <https://www.freeprojectz.com/project-source-code-database-download/blood-bank-management-system-project>
* <https://www.lovelycoding.org/blood-bank-management-system/>
* <https://www.filemakr.com/btech-final-year-project-report-blood-bank-management>
* <https://nevonprojects.com/online-blood-bank-project/>
* <https://projectworlds.in/free-projects/php-projects/online-blood-bank-management-system-in-php/>
* https://itsourcecode.com/fyp/blood-bank-management-system-project-report-documentation-pdf/