|  |
| --- |
|  |
| LifeBlood: A Blood Donation System Android App |
| Software Project Lab 3 |
|  |
|  |
|  |

|  |
| --- |
| **NOVEMBER 25, 2018**  **INSTITUTE OF INFORMATION TECHNOLOGY**  **UNIVERSITY OF DHAKA** |

LifeBlood

A Blood Donation System Android App

**Supervised by**

Dr. Mohammed Shafiul Alam Khan

Associate Professor

Institute of Information Technology

University of Dhaka

**Submitted by**

Asadullah Hill Galib

BSSE-0712

Letter of Transmittal

November 25, 2018

BSSE 4th Year Exam Committee

Institute of Information Technology

University of Dhaka

Dear Sir,

I have prepared the report on LifeBlood: A Blood Donation System Android App. This report includes the technical document, implementation detail, test plans for testing the project and user manual to run the app.

The primary purpose of this report is to summarize my findings from the work that I completed. Therefore, I submit it to you for your kind approval.

Yours sincerely

Asadullah Hill Galib

BSSE0712

4th Year, 8th Semester, 7th Batch

Institute of Information Technology

University of Dhaka

Document Authentication

This project document has been approved by the following persons.

--------------------------------

**Prepared by**

Asadullah Hill Galib

BSSE-0712

-------------------------------

**Approved by**

Dr. Mohammed Shafiul Alam Khan

Associate Professor

Institute of Information Technology

University of Dhaka

Letter of Endorsement

November 25, 2018

**To Whom It May Concern**

Subject: Approval of the report.

This letter is to certify that all the information mentioned in this document is true. The  
project mentioned here have had successful involvement of Asadullah Hill Galib,  
BSSE0712, Institute of Information Technology, University of Dhaka.

I wish him all the best and hope that he will lead a successful career.

**Project Supervisor**

Dr. Mohammed Shafiul Alam Khan  
Associate Professor  
Institute of Information technology  
University of Dhaka

Abstract

This project will aim to build a blood donation system android app, which can ease our life. The scope of the application is to be serviceable and easy-to-use. The object of this study is to develop a technical report, implementation detail, test plans for testing the project and user manual to run the app.

Contents

[Chapter 1: Introduction 12](#_Toc530891129)

[1.1 Purpose 12](#_Toc530891130)

[1.2 Intended Audience 12](#_Toc530891131)

[1.3 Conclusion 13](#_Toc530891132)

[Chapter 2: Elicitation 14](#_Toc530891133)

[2.1 Introduction 14](#_Toc530891134)

[2.2 Quality Function Deployment 14](#_Toc530891135)

[2.2.1 Normal Requirements 14](#_Toc530891136)

[2.2.2 Expected Requirements 15](#_Toc530891137)

[2.2.3 Exciting Requirements 15](#_Toc530891138)

[2.3 Usage Scenario 15](#_Toc530891139)

[Chapter 3: Scenario Based Modeling 17](#_Toc530891140)

[3.1 Introduction 17](#_Toc530891141)

[3.2 Definition of Use Case 17](#_Toc530891142)

[3.3 Use Case Diagram 17](#_Toc530891143)

[3.3.1 Level-0 Use Case Diagram-LifeBlood 18](#_Toc530891144)

[3.3.2 Level-1 Use Case Diagram-Subsystems 19](#_Toc530891145)

[3.3.3 Level-1.1 Use Case Diagram-Registration 21](#_Toc530891146)

[3.3.4 Level-1.2 Use Case Diagram-Authentication 22](#_Toc530891147)

[3.3.5 Level-1.3 Use Case Diagram-User Profile 23](#_Toc530891148)

[3.3.6 Level-1.4 Use Case Diagram-Blood Donation 24](#_Toc530891149)

[3.3.7 Level-1.5 Use Case Diagram-Donor Activation 25](#_Toc530891150)

[Chapter 4: Data Modeling 26](#_Toc530891151)

[4.1 Data Modeling Concept 26](#_Toc530891152)

[4.2 Data Object Concept 26](#_Toc530891153)

[4.3 Noun Identification 26](#_Toc530891154)

[4.4 Potential Data Object 28](#_Toc530891155)

[4.5 Analysis for Finalizing Data Objects 28](#_Toc530891156)

[4.6 Final Data Objects 29](#_Toc530891157)

[4.7 Schema Table 29](#_Toc530891158)

[4.8 Relationship among Data Objects 30](#_Toc530891159)

[4.9 ER Diagram 31](#_Toc530891160)

[Chapter 5: Class Based Modeling 32](#_Toc530891161)

[5.1 Class Based Modeling Concept 32](#_Toc530891162)

[5.2 General Classifications 32](#_Toc530891163)

[5.3 Selection Criteria 34](#_Toc530891164)

[5.4 Potential Class 34](#_Toc530891165)

[5.5 Verb Identification 35](#_Toc530891166)

[5.6 Attributes and Methods of Potential Classes 36](#_Toc530891167)

[5.7 Analysis of Potential Classes 37](#_Toc530891168)

[5.8 Final Classes 37](#_Toc530891169)

[5.9 Attributes and Methods of Final classes 37](#_Toc530891170)

[5.10 Class Cards 38](#_Toc530891171)

[5.11 CRC Diagram 40](#_Toc530891172)

[Chapter 6: Architectural Design 41](#_Toc530891173)

[6.1 Introduction 41](#_Toc530891174)

[6.2 Represent the system in context 42](#_Toc530891175)

[6.3 Refine the architecture into components 43](#_Toc530891176)

[6.4 Instantiations of the system 44](#_Toc530891177)

[Chapter 7: Implementation Overview 45](#_Toc530891178)

[7.1 Front-End Implementation 45](#_Toc530891179)

[7.2 Back-End Implementation 45](#_Toc530891180)

[7.2.1 Database System: Cloud Firestore 45](#_Toc530891181)

[7.2.2 Google Maps SDK for Android 46](#_Toc530891182)

[7.3 Development Tools and Library 46](#_Toc530891183)

[7.3.1 Android Studio 46](#_Toc530891184)

[7.3.2 Android SDK 46](#_Toc530891185)

[7.3.3 Google Maps Android API Utility Library 46](#_Toc530891186)

[7.3.4 LocationManager 46](#_Toc530891187)

[Chapter 8: Testing 47](#_Toc530891188)

[8.1 Test Plan Identifier 47](#_Toc530891189)

[8.2 Introduction 47](#_Toc530891190)

[8.3 Items to be Tested 47](#_Toc530891191)

[8.4 Features to be Tested 47](#_Toc530891192)

[8.5 Reference to Related Documents 48](#_Toc530891193)

[8.6 Approach 48](#_Toc530891194)

[8.7 Item Pass/ Fail Criteria 48](#_Toc530891195)

[8.8 Suspension Criteria and Resumption Requirements 48](#_Toc530891196)

[8.9 Test Deliverables 48](#_Toc530891197)

[8.10 Environmental Need 49](#_Toc530891198)

[8.11 Testing Cost 49](#_Toc530891199)

[8.12 Test Cases 49](#_Toc530891200)

[Chapter 9: User Manual 56](#_Toc530891201)

[9.1 Login 57](#_Toc530891202)

[9.2 Registration 58](#_Toc530891203)

[9.3 Account Settings 59](#_Toc530891204)

[9.4 Main Page 60](#_Toc530891205)

[9.5 My Location 61](#_Toc530891206)

[9.6 Blood Request 62](#_Toc530891207)

[9.7 Nearer Donors List, Location and Calling 63](#_Toc530891208)

[9.8 Logout 64](#_Toc530891209)

[Chapter 10: Conclusion 65](#_Toc530891210)

***Table of Figures:***

[Figure 1: Relationship among Data Objects 30](#_Toc530890731)

[Figure 2: ER Diagram 31](#_Toc530890732)

[Figure 3: CRC Diagram 40](#_Toc530890733)

[Figure 4: Architectural Context Diagram 42](#_Toc530890734)

[Figure 5: System Components 43](#_Toc530890735)

[Figure 6: Componet Elaboration 44](#_Toc530890736)

[Figure 7: User Manual: Splash Screen 56](#_Toc530890737)

[Figure 8: User Manual: Login 57](#_Toc530890738)

[Figure 9: User Manual: Registration 58](#_Toc530890739)

[Figure 10: User Manual: Account Settings 59](#_Toc530890740)

[Figure 11: User Manual: Main Page 60](#_Toc530890741)

[Figure 12: User Manual: My Location 61](#_Toc530890742)

[Figure 13: User Manual: Blood Request 62](#_Toc530890743)

[Figure 14: User Manual: Donor List 63](#_Toc530890744)

[Figure 15: User Manual: Logout 64](#_Toc530890745)

***List of Tables:***

[Table 1: Test Cases for Registration 49](#_Toc530890721)

[Table 2: Test Cases for Authentication 50](#_Toc530890722)

[Table 3: Test Cases for Account Settings 51](#_Toc530890723)

[Table 4: Test Cases for User Profile 52](#_Toc530890724)

[Table 5: Test Cases for Blood Request Raising 53](#_Toc530890725)

[Table 6: Test Cases for Getting nearer donors’ list: 54](#_Toc530890726)

[Table 7: Test Cases for Calling specific donor 54](#_Toc530890727)

[Table 8: Test Cases for Donors’ position on map 55](#_Toc530890728)

[Table 9: Test Cases for Own location on map 55](#_Toc530890729)

[Table 10: Test Cases for Logout 55](#_Toc530890730)

# Chapter 1: Introduction

## 1.1 Purpose

This document briefly describes the overall system of proposed android app- **LifeBlood**. It contains functional, non-functional and supporting requirements and establishes a requirement baseline for the development of the system. Besides it also includes the implementation techniques, test plan for testing and user manual. The requirements contained in the technical report are independent, uniquely numbered and organized by topic. The technical report serves as an official means of communicating user requirements to the developer and provides a common reference point for both the developer team and the stakeholder community. The technical report will evolve over time as users and developers work together to validate, clarify and expand its contents.

## 1.2 Intended Audience

This technical report is intended for several audiences including the customers as well as the project managers, designers, developers, and testers.

1. The customer will use this technical report to verify that the developer team has created a product that is acceptable to the customer.
2. The project managers of the developer team will use this technical report to plan milestones and a delivery date and ensure that the developing team is on track during development of the system.
3. The designers will use this technical report as a basis for creating the system’s design. The designers will continually refer to this technical report to ensure that the system they are designing will fulfill the customer’s needs.
4. The developers will use this technical report as a basis for developing the system’s functionality. The developers will link the requirements defined in this technical report to the software they create to ensure that they have created a software that will fulfill all the customer’s documented requirements.
5. The testers will use this technical report to derive test plans and test cases for each documented requirement. When portions of the software are complete, the testers will run their tests on that software to ensure that the software fulfills the requirements documented in this SRS. The testers will again run their tests on the entire system when it is complete and ensure that all requirements documented in this technical report have been fulfilled.

## 1.3 Conclusion

This analysis of the audience helped me to focus on the users who will be using our analysis. This overall document will help each person related to this project to have a better idea about the project.

# Chapter 2: Elicitation

After discussing on the Inception phase, we need to focus on the Elicitation phase. So, this chapter specifies the Elicitation phase.

## 2.1 Introduction

Requirements Elicitation is a part of requirements engineering that is the practice of gathering requirements from the users, customers and other stakeholders. We have faced many difficulties, like understanding the problems, making questions for the stakeholders, limited communication with the stakeholders due to a short amount of time and volatility. Though it is not easy to gather requirements within a very short time, we have surpassed these problems in an organized and systematic manner.

## 2.2 Quality Function Deployment

Quality Function Deployment (QFD) is a technique that translates the needs of the customer into technical requirements for software. Ultimately the goal of QFD is to translate subjective quality criteria into objective ones that can be quantified and measured, and which can then be used to design and manufacture the product. It is a methodology that concentrates on maximizing customer satisfaction from the software engineering process. So, we have followed this methodology to identify the requirements for the project. The requirements, which are given below, are identified successfully by the QFD.

### 2.2.1 Normal Requirements

Normal requirements consist of objectives and goals that are stated during the meeting with the customers. Normal requirements of our project are:

1. User-friendly design.
2. Signup and login system of users
3. Requesting blood and searching donors according to blood group
4. Getting nearest and accepted blood donors’ sorted list according to distance
5. Viewing neared donors on map
6. Viewing own location on map
7. Updating profile

### 2.2.2 Expected Requirements

These requirements are implicit to the system and may be so fundamental that the customer does not explicitly state them. Their absence will be a cause for dissatisfaction.

1. Providing personal account for user
2. Allow only valid users to login.

### 2.2.3 Exciting Requirements

These requirements are for features that go beyond the customer's expectation and prove to be very satisfying when present.

1. Calling nearer donors
2. Custom markers for donors on map

## 2.3 Usage Scenario

“**LifeBlood**” is an android app for simplifying blood donation system. This system includes the following:

**a.  Registration**

A new user can sign up by giving the following information: username, password, name, age, mobile number, address and blood group. System check whether the username is unique. After successful registration a new profile will be created. By default, a user is a receiver as well as a donor who can make request for blood. She can also become only a blood receiver by activating it after registration.

**b.  Authentication**

User can log in to her account using email and password. And she also can log out from her account at any time.

**c.  User Profile**

User profile holds the details of that user. This contains user details, profile image, name, blood group, phone number and email id. Any user can update his profile data any time.

**d.  Receiver**

Receiver is one who receives blood from donor. S/he can make a blood request of urgent blood. S/he can also acknowledge some specific issues like hospital name, request details and blood group via that request. This request will search through the customized map to sort out nearer donors of that desired blood group. This list is sorted according to distance. Then it shows all the neared donor list and markers on map. Receiver will then connect to any specific donor via phone call.

**e.  Donor**

Donor is one who donates the blood. A user is by default a blood receiver as well as a donor. Any user can deactivate his donorship by deactivating it. Then h/she only receive blood but not act as a donor.

**f. Map**

Any user can view his/her current location on map. Besides when a blood request raised, all nearer donors’ position is being marked on the map. This is a custom maker which also holds the user profile image and distance.

# Chapter 3: Scenario Based Modeling

This chapter describes the Scenario Based Model for “LifeBlood”.

## 3.1 Introduction

Although the success of a computer-based system or product is measured in many ways, user satisfaction resides at the top of the list. If we understand how end users (and other actors) want to interact with a system, our software team will be better able to properly characterize requirements and build meaningful analysis and design models. Hence, requirements modeling begins with the creation of scenarios in the form of Use Cases, activity diagrams and swim lane diagrams.

## 3.2 Definition of Use Case

A Use Case captures a contract that describes the system behavior under various conditions as the system responds to a request from one of its stakeholders. A Use Case tells a stylized story about how an end user interacts with the system under a specific set of circumstances. A Use Case diagram simply describes a story using corresponding actors who perform important roles in the story and makes the story understandable for the users.

The first step in writing a Use Case is to define that set of “actors” that will be involved in the story. Actors are the different people that use the system or product within the context of the function and behavior that is to be described. Actors represent the roles that people play as the system operators. Every user has one or more goals when using system.

**Primary Actor**

Primary actors interact directly to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software.

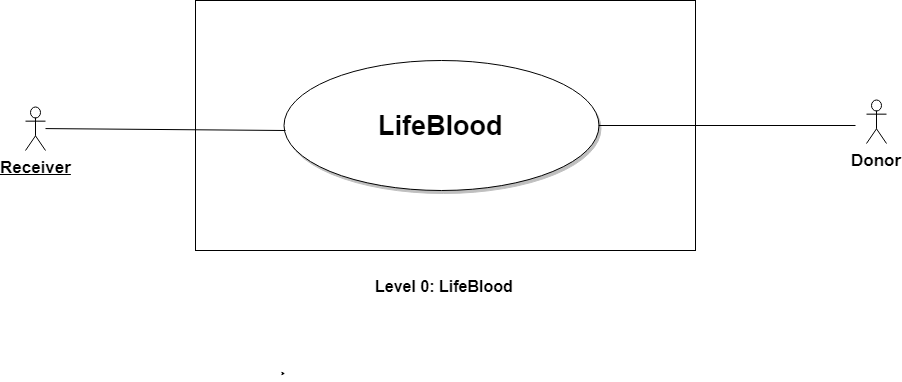
**Secondary Actor**

Secondary actors support the system so that primary actors can do their work. They either produce or consume information.

## 3.3 Use Case Diagram

Use Case diagrams give the non-technical view of overall system.

### 3.3.1 Level-0 Use Case Diagram-LifeBlood



**Name:** LifeBlood

**ID:** Level-0 Use Case

**Primary Actors:** Receiver, Donor

**Secondary Actors:** None

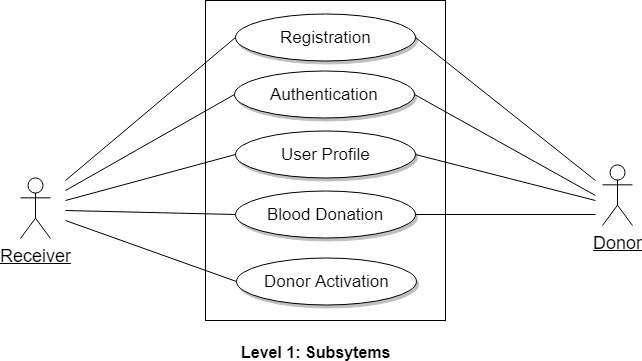
**Description of Level 0 Use Case Diagram:**

After analyzing the user story, we found that there are two actors who will use this system as a system operator.

Those actors are given below:

* Receiver
* Donor

### 3.3.2 Level-1 Use Case Diagram-Subsystems



**Name:** Subsystems of LifeBlood

**ID:** Level-1 Use Case

**Primary Actors:** Receiver, Donor

**Secondary Actors:** None

**Action-Reply:**

**Action 1: Provided registration Information**

Reply 1: Registration successful

Action 2: Enter login credentials

Reply 2: Login successful

Action 3: Activate Donor

Reply 3: Donor Activated

Action 4: Make Blood Request

Reply 4: Blood Request Made

Action 5: Logout

Reply 5: Logged out

Action 6: Accept Blood Request

Reply 6: Blood Request Accepted

Action 7: Sending Notification

Reply 7: Sent Notification

Action 8: Search Map

Reply 8: Search result shown

Action 9: Rate Donor

Reply 9: Donor rated

Action 10: Give Review

Reply 10: Review given

Action 11: Reject Request

Reply 11: Request Rejected

Action 12: Show Ranking

Reply 12: Ranking Shown

Action 13: Create profile

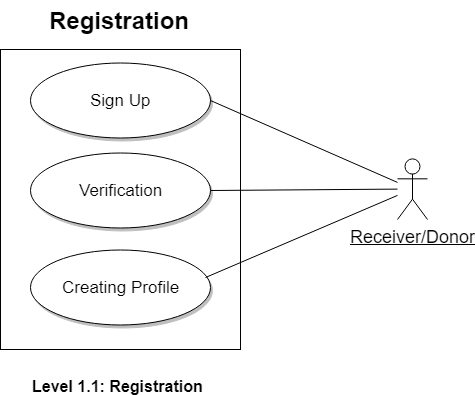
Reply 13: Profile created

**Description of Level 1 Use Case Diagram:**

There are seven subsystems:

* Authentication
* Registration
* User Profile
* Blood Donation
* Donor Activation
* Review System
* Ranking System

### 3.3.3 Level-1.1 Use Case Diagram-Registration

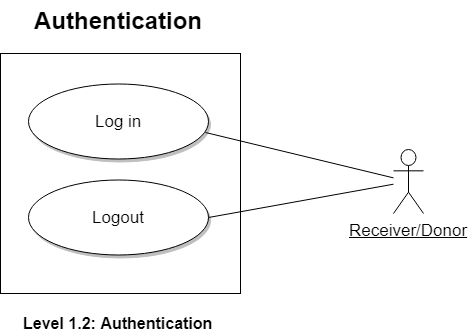


**Name:** Registration

**ID:** Level-1.1 Use Case

**Primary Actors:** Receiver, Donor

### 3.3.4 Level-1.2 Use Case Diagram-Authentication

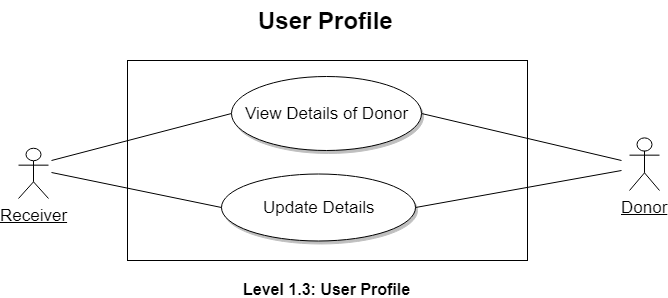


**Name:** Authentication

**ID:** Level-1.2 Use Case

**Primary Actors:** Receiver, Donor

### 3.3.5 Level-1.3 Use Case Diagram-User Profile

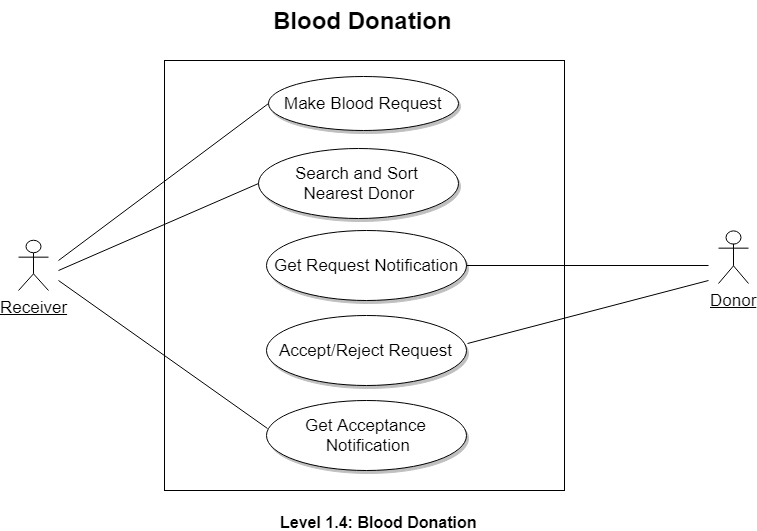


**Name:** User Profile

**ID:** Level-1.3 Use Case

**Primary Actors:** Receiver, Donor

### 3.3.6 Level-1.4 Use Case Diagram-Blood Donation

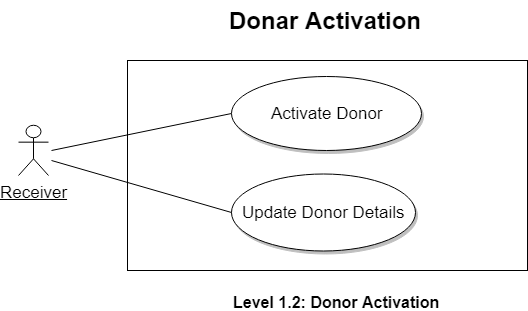


**Name:** Blood Donation

**ID:** Level-1.4 Use Case

**Primary Actors:** Receiver, Donor

### 3.3.7 Level-1.5 Use Case Diagram-Donor Activation



**Name:** Donor Activation

**ID:** Level-1.5 Use Case

**Primary Actors:** Receiver, Donor

# Chapter 4: Data Modeling

This Chapter is intended to describe data modeling of LifeBlood.

## 4.1 Data Modeling Concept

If software requirements include the necessity to create, extend or interact with a database or complex data structures need to be constructed and manipulated, then the software team chooses to create data models as part of overall requirements modeling. The entity-relationship diagram (ERD) defines all data objects that are processed within the system, the relationships between the data objects and the information about how the data objects are entered, stored, transformed and produced within the system.

## 4.2 Data Object Concept

A data object is a representation of composite information that must be understood by the software. Here, composite information means an information that has a number of different properties or attributes. A data object can be an external entity, a thing, an occurrence, a role, an organizational unit, a place or a structure.

## 4.3 Noun Identification

|  |  |  |  |
| --- | --- | --- | --- |
| **SL**  **No.** | **Nouns** | **P/S** | **Attributes** |
| 1 | LifeBlood | P |  |
| 2 | Android app | P |  |
| 3 | Blood donation system | P |  |
| 4 | User | S | 6-12, 27, 30, 33 |
| 5 | Information | P |  |
| 6 | Username | S |  |
| 7 | Password | S |  |
| 8 | Name | S |  |
| 9 | Age | S |  |
| 10 | Mobile number | S |  |
| 11 | Address | S |  |
| 12 | Blood group | S |  |
| 13 | System | P |  |
| 14 | Profile | P |  |
| 15 | Receiver | S | 6-12, 27, 33 |
| 16 | Donor | S | 6-12, 27-30, 33, 36 |
| 17 | Blood Request | S | 8, 10, 12, 18-21 |
| 18 | Patient status | S |  |
| 19 | Level of emergency | S |  |
| 20 | Expected time of arrival | S |  |
| 21 | Location | S |  |
| 22 | Account | P |  |
| 23 | Registration | S |  |
| 24 | Authentication | S | 6,7 |
| 25 | User profile | P |  |
| 26 | User details | P |  |
| 27 | Review | S |  |
| 28 | Health Status | S |  |
| 29 | Ranking | S |  |
| 30 | Availability status | S |  |
| 31 | Profile data | P |  |
| 32 | Activation | P |  |
| 33 | Notification | S |  |
| 34 | Phone | P |  |
| 35 | Rate | P |  |
| 36 | Rating | S |  |

## 4.4 Potential Data Object

**User:** Username, Password, Name, Age, Blood Group, Mobile number, Address, Review, Notification.

**Receiver:** Username, Password, Name, Age, Blood Group, Mobile number, Address, Review, Notification.

**Donor:** Username, Password, Name, Age, Blood Group, Mobile number, Address, Review, Notification, Health Status, Ranking, Availability status, Rating.

**Authentication:** Username, Password.

**Blood Request:** Name, Mobile Number, Blood Group, Patient status, Level of emergency, Expected time of arrival, Location

## 4.5 Analysis for Finalizing Data Objects

1. User and Receiver have same set of attributes. So, we can merge these into one object called Receiver.
2. Authentication has attributes which we can get from the Receiver object. So, we merge these two objects into a single object called Receiver.

## 4.6 Final Data Objects

|  |  |  |
| --- | --- | --- |
| **No.** | **Entity** | **Attributes** |
| 1 | Receiver | **Receiver ID**, Username, Password, Name, Age, Blood Group, Mobile number, Address, Review, Notification |
| 2 | Donor | **Donor ID**, Username, Password, Name, Age, Blood Group, Mobile number, Address, Notification |
| 3 | Blood Request | **Request ID**, Receiver ID, Donor ID, Name, Mobile Number, Blood Group, Hospital Name, Request Details |

## 4.7 Schema Table

|  |  |  |
| --- | --- | --- |
| 1. **Receiver** | | |
| **Attributes** | **Types** | **Size** |
| User ID  Username  Password  Name  Age  Blood Group  Mobile number  Address  Review  Notification | NUMBER  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2 | 5  25  20  40  10  10  15  80  250  40 |

|  |  |  |
| --- | --- | --- |
| 1. **Donor** | | |
| **Attributes** | **Types** | **Size** |
| User ID  Username  Password  Name  Age  Blood Group  Mobile number  Address | NUMBER  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2 | 5  25  20  40  10  10  15  80 |

|  |  |  |
| --- | --- | --- |
| 1. **Blood Request** | | |
| **Attributes** | **Types** | **Size** |
| **Request ID**  Receiver ID  Donor ID  Name  Mobile Number  Blood Group  Hospital Name  Request Details | NUMBER  NUMBER  NUMBER  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2  VARCHAR2 | 5  5  5  40  15  10  80  200 |

## 4.8 Relationship among Data Objects

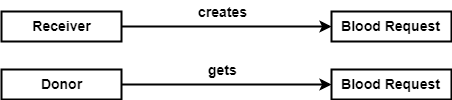


Figure 1: Relationship among Data Objects

## 4.9 ER Diagram

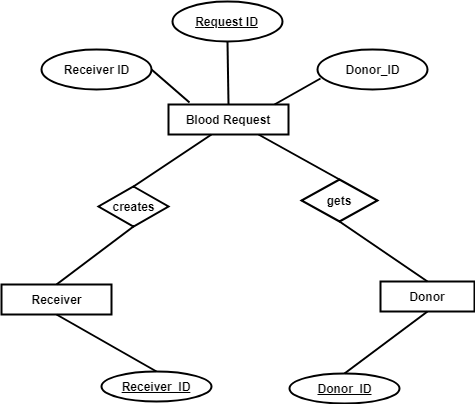


Figure 2: ER Diagram

# Chapter 5: Class Based Modeling

This Chapter is intended to describe class-based modeling of **LifeBlood**.

## 5.1 Class Based Modeling Concept

Class-based modeling represents the objects that the system will manipulate, the operations that will applied to the objects, relationships between the objects and the collaborations that occur between the classes that are defined.

## 5.2 General Classifications

To identify the potential classes, we have first selected the nouns from the solution space of the story. These were then characterized in seven general classifications. The seven general characteristics are as follows:

1. External entities
2. Things
3. Events
4. Roles
5. Organizational units
6. Places
7. Structures

Following are the specifications of the nouns according to the general classifications:

|  |  |  |  |
| --- | --- | --- | --- |
| **SL**  **No.** | **Potential Class** | **P/S** | **General Classification (GC)** |
| 1 | LifeBlood | P |  |
| 2 | Android app | P |  |
| 3 | Blood donation system | P |  |
| 4 | User | S | 2,4,5,7 |
| 5 | Information | P |  |
| 6 | Username | S | 2 |
| 7 | Password | S | 2 |
| 8 | Name | S | 2 |
| 9 | Age | S | 2 |
| 10 | Mobile number | S | 2 |
| 11 | Address | S | 2, 6 |
| 12 | Blood group | S | 2 |
| 13 | System | P |  |
| 14 | Profile | P |  |
| 15 | Receiver | S | 2,4,5,7 |
| 16 | Donor | S | 2,4,5,7 |
| 17 | Blood Request | S | 3,5,7 |
| 18 | Patient status | S | 2 |
| 19 | Level of emergency | S | 2 |
| 20 | Expected time of arrival | S | 2 |
| 21 | Location | S | 2,6 |
| 22 | Account | P |  |
| 23 | Registration | S | 3,5,7 |
| 24 | Authentication | S | 3,5,7 |
| 25 | User profile | P |  |
| 26 | User details | P |  |
| 27 | Review | S | 2 |
| 28 | Health Status | S | 2 |
| 29 | Ranking | S | 2 |
| 30 | Availability status | S | 2 |
| 31 | Profile data | P |  |
| 32 | Activation | P |  |
| 33 | Notification | S | 2,3 |
| 34 | Phone | P |  |
| 35 | Rate | P |  |
| 36 | Rating | S | 2 |

## 5.3 Selection Criteria

The potential classes were then selected as classes by six Selection Criteria. A potential class becomes a class when it fulfills all six characteristics.

1. Retained Information

2. Needed Services

3. Multiple Attributes

4. Common attributes

5. Common operations

6. Essential requirements

|  |  |  |  |
| --- | --- | --- | --- |
| **SL No** | **Potential Class** | **Special Classification (SC)** | |
| **Accepted** | **Rejected** |
| 1 | User | 1-6 |  |
| 2 | Receiver | 1-6 |  |
| 3 | Donor | 1-6 |  |
| 4 | Registration | 2,3,4,6 | 1,5 |
| 5 | Authentication | 2,3,4,6 | 1,5 |
| 6 | Blood Request | 1-3, 6 | 4,5 |

## 5.4 Potential Class

From above table, we have taken the nouns who passed three or more accepted criteria. So there are fourteen candidate classes who are selected primarily. Those are-

1. User
2. Receiver
3. Donor
4. Registration
5. Authentication
6. Blood Request

## 5.5 Verb Identification

|  |  |  |
| --- | --- | --- |
| **SL No** | **Verbs** | **Remarks** |
| 1 | Sign up | S |
| 2 | Give info | S |
| 3 | Check username | S |
| 4 | Create profile | S |
| 5 | Log in | S |
| 6 | Log out | S |
| 7 | Make Request | S |
| 8 | Activate donor | S |
| 9 | Hold details | P |
| 10 | Contain details | P |
| 11 | See data | P |
| 12 | Receive blood | P |
| 13 | Acknowledge issues | P |
| 14 | Search map | S |
| 15 | Sort donors | S |
| 16 | Accept request | S |
| 17 | Connect phone | P |
| 18 | Donate blood | P |
| 19 | Get notification | S |
| 20 | Reject Request | S |
| 21 | Give review | S |
| 22 | Rate donor | S |

## 5.6 Attributes and Methods of Potential Classes

Analyzing the above table, we have categorized the verbs and convert them into method names. We put them to their respective classes and showed them in the following table-

|  |  |  |  |
| --- | --- | --- | --- |
| **SL No** | **Preliminary Class** | **Attributes** | **methods()** |
| **1** | **Registration** | receiverID + username + password + name + age + bloodGroup + mobileNumber + address | checkIfUserNameAvailable()+ takeInput()+ signUp()+ profileCreation() |
| **2** | **Authentication** | Username + password | takeInput()+ checkCredentials() +  logIn() + logout() |
| **3** | **User** | receiverID + username + password + name + age + bloodGroup + mobileNumber + address + review + notification | callRequest() + searchMap() + sortDonors()+ getAcceptanceNotification() + activateDonor() + rateDonor()+ giveReview() |
| **4** | **Receiver** | receiverID + username + password + name + age + bloodGroup + mobileNumber + address + review + notification | callRequest() + searchMap() + sortDonors()+ getAcceptanceNotification() + activateDonor() + rateDonor()+ giveReview() |
| **5** | **Donor** | donorID + username + password + name + age + bloodGroup + mobileNumber + address + review + notification + healthStatus + ranking + availabilityStatus + rating | callRequest() + searchMap() + sortDonors()+ getAcceptanceNotification() + activateDonor() + rateDonor()+ giveReview() + accpetRequest() + rejectRequest() + getNotification() |
| **6** | **BloodRequest** | requestID +  receiverID + donorID + name + mobileNumber + bloodGroup + patientStatus + levelOfEmergency + expectedTimeOfArrival | makeRequest()+ getField() + setField() + sendNotificationToDonors()+ |

## 5.7 Analysis of Potential Classes

1. **User** and **Receiver** have almost same attributes and methods since we are making it a common class name **Receiver**.
2. Since the **Donor** have same attributes and methods with **Receiver** class and **Donor** have some additional attributes and methods, it is wise to make it a subclass of **Receiver** class.
3. **Registration** class can be merged with **Authentication** class to remove complexity.

## 5.8 Final Classes

There are three final classes and one inherited class.

1. Authentication
2. Receiver
   1. Donor
3. BloodRequest

## 5.9 Attributes and Methods of Final classes

|  |  |  |  |
| --- | --- | --- | --- |
| **SL No** | **Final Class** | **Attributes** | **Methods ()** |
| **1** | **Authentication** | receiverID + username + password + name + age + bloodGroup + mobileNumber + address | checkIfUserNameAvailable()+ takeInput()+ signUp()+ profileCreation() + takeInput()+ checkCredentials() +  logIn() + logout() |
| **2** | **Receiver** | receiverID + username + password + name + age + bloodGroup + mobileNumber + address + review + notification | callRequest() + searchMap() + sortDonors()+ getAcceptanceNotification() + activateDonor() + rateDonor()+ giveReview() |
| **2.a** | **Donor** | donorID + healthStatus + ranking + availabilityStatus + rating | accpetRequest() + rejectRequest() + getNotification() |
| **3** | **BloodRequest** | requestID +  receiverID + donorID + name + mobileNumber + bloodGroup + patientStatus + levelOfEmergency + expectedTimeOfArrival | makeRequest()+ getField() + setField() + sendNotificationToDonors()+ |

## 5.10 Class Cards

After identifying our final classes, we have generated the following class cards.

|  |  |
| --- | --- |
| 1. **Authentication** | |
| **Responsibilities** | **Collaborative Classes** |
| Getting registered  Getting authenticated | Receiver  Receiver |

|  |  |
| --- | --- |
| **2. Receiver** | |
| **Responsibilities** | **Collaborative Classes** |
| Calling Request  Searching Map  Sorting Donors  Activating Donor  Rating Donor  Giving Review  Getting Acceptance Notification | BloodRequest  BloodRequest  BloodRequest, Donor  -  Donor  Donor  Donor |

|  |  |
| --- | --- |
| **2.a. Donor** | |
| **Responsibilities** | **Collaboration** |
| Accepting Request  Rejecting Request  Getting Notification | BloodRequest  BloodRequest  BloodRequest |

|  |  |
| --- | --- |
| **3. BloodRequest** | |
| **Responsibilities** | **Collaboration** |
| Making Request  Getting Field  Setting Field  Sending Notification | Receiver  Receiver  -  Donor |

## 5.11 CRC Diagram

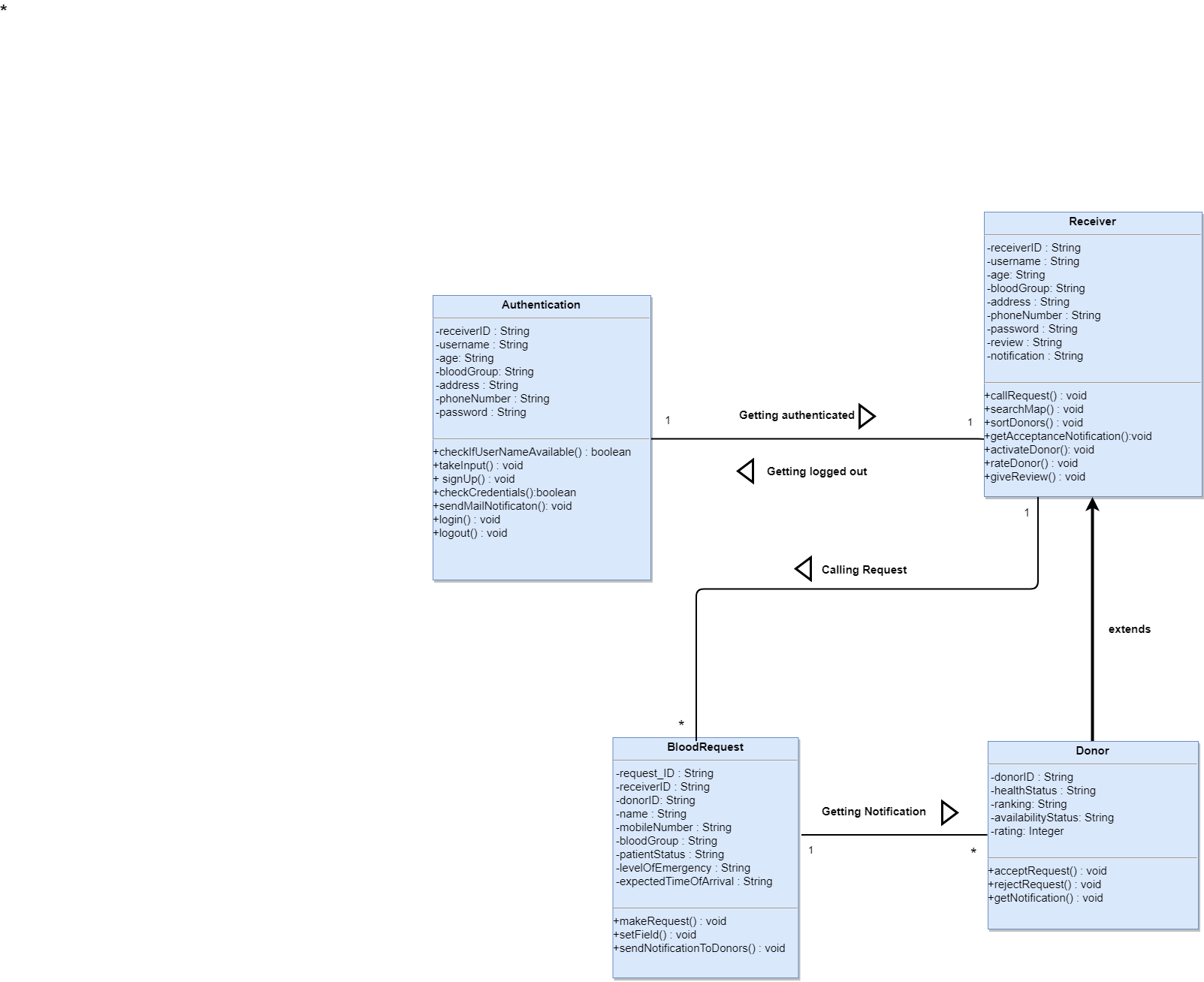


Figure 3: CRC Diagram

# Chapter 6: Architectural Design

## 6.1 Introduction

This chapter discusses architectural design of the **LifeBlood**.

The software architecture of a program or computing system is the structure or structures of the system which comprise-

• The software components

• The externally visible properties of those components

• The relationships among the components

Software architectural design represents the structure of the data and program components that are required to build a computer-based system

Architectural design contains the following steps:

1. Represent the system in context

2. Refine the architecture into components

3. Describe instantiations of the system

## 6.2 Represent the system in context

The ACD models the way software interacts with entities external to its boundaries. It also identifies systems that interoperate with the target system.

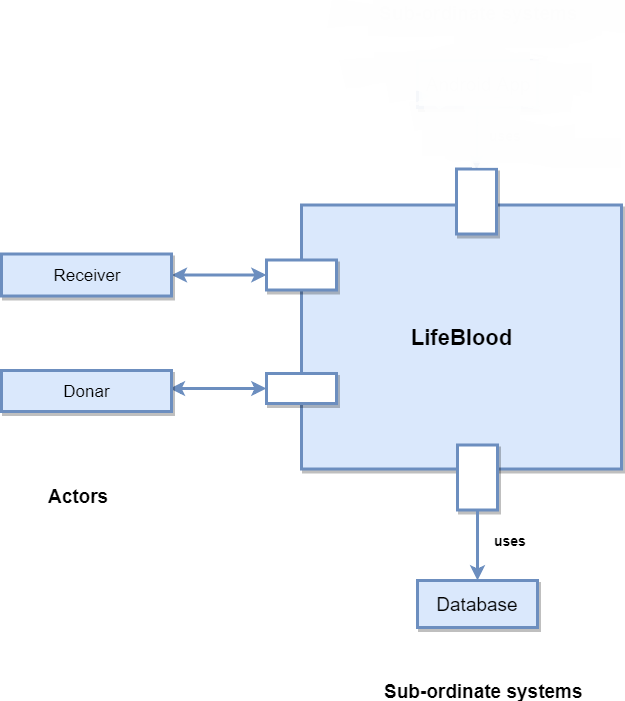


Figure 4: Architectural Context Diagram

## 6.3 Refine the architecture into components

Based on the context, the architectural designer refines the software architecture into components to illustrate the overall structure and architectural style of the system.

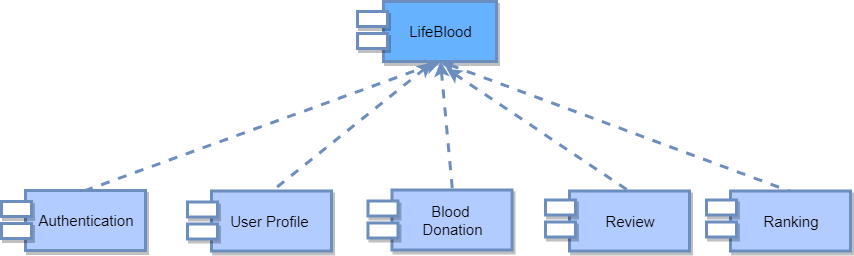


Figure 5: System Components

## 6.4 Instantiations of the system

An actual instantiation of the architecture is developed by applying it to a specific problem. This demonstrates that the architectural structure, style and components are appropriate.

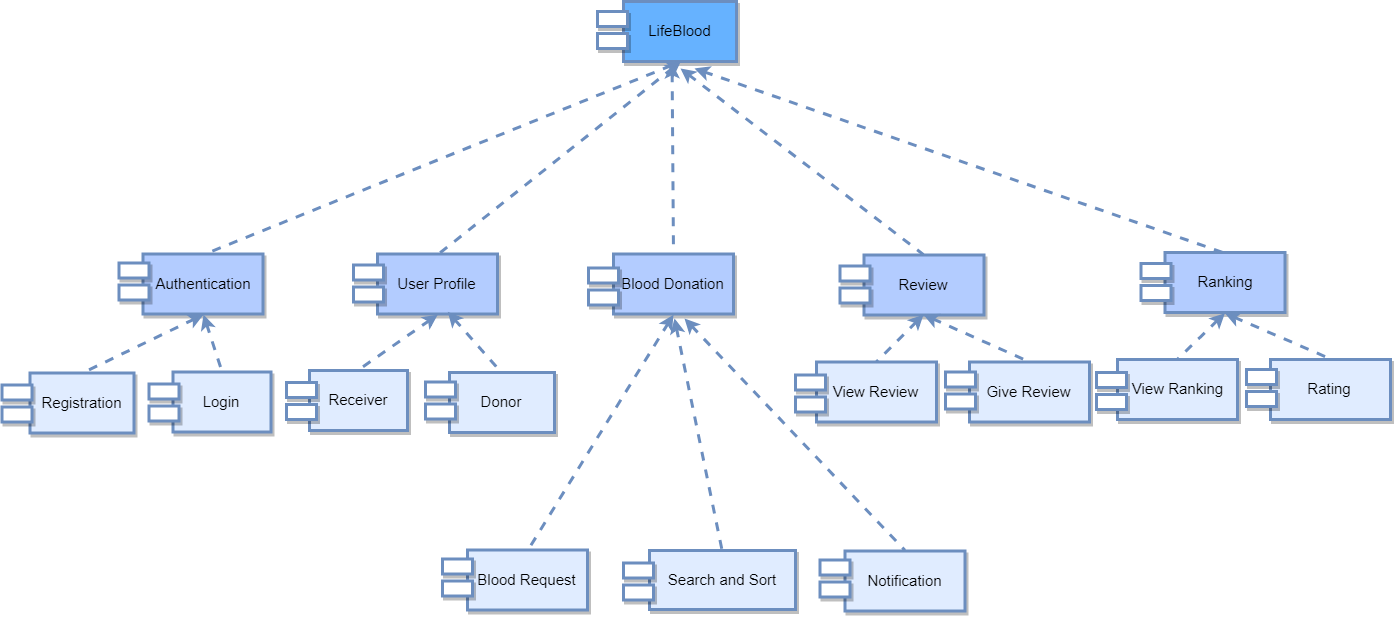


Figure 6: Componet Elaboration

# Chapter 7: Implementation Overview

LifeBlood is a mobile application built for the android platform. It is a native app in nature. Android apps can be written using Kotlin, Java, and C++ languages using the Android software development kit (SDK), while using other languages is also possible.

LifeBlood is built only for the android platform. The details of implementation are given below:

## 7.1 Front-End Implementation

For Designing User Interface, I have used traditional XML. XML is used to create the user interface of the app. XML stands for Extensible Markup Language. XML is a markup language much like HTML which is used to describe data. XML is readable both by human and machine. The layout of each page of the app was made by using the XML.

Besides some library and external dependencies also needed for designing user interface. Some of those are:

* CircleImageView - for circle shaped image on profile picture
* MapView - for rendering map
* Glide - for rendering and manipulating images more efficiently
* Android-Image-Cropper – for cropping images etc.

Java, the core language of Android is also used to create the front end. Java is in the heart of the android the app. It is used to render the layouts made by using the XML to the screen. All the business logic of the app is also written in Java.

## 7.2 Back-End Implementation

### 7.2.1 Database System: Cloud Firestore

For the database system I used Firestore Database System which is new, excellent and too much easy to use.

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud Platform. Like Firebase Realtime Database, it keeps your data in sync across client apps through real-time listeners and offers offline support for mobile and web, so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud Platform products, including Cloud Functions. These are some of the best features of Firestore:

* Build serverless apps
* Sync data across devices, on or offline
* Simple and effortless
* Enterprise-grade, scalable NoSQL

### 7.2.2 Google Maps SDK for Android

For using and rendering map, I mainly used Maps SDK for Android. With the Maps SDK for Android, you can add maps based on Google Maps data to your application. The API automatically handles access to Google Maps servers, data downloading, map display, and response to map gestures. You can also use API calls to add markers, polygons, and overlays to a basic map, and to change the user's view of a map area. These objects provide additional information for map locations and allow user interaction with the map. The API allows you to add these graphics to a map:

* Icons anchored to specific positions on the map (Markers).
* Sets of line segments (Polylines).
* Enclosed segments (Polygons).
* Bitmap graphics anchored to specific positions on the map (Ground Overlays).
* Sets of images which are displayed on top of the base map tiles (Tile Overlays).

## 7.3 Development Tools and Library

### 7.3.1 Android Studio

Android Studio is the official integrated development environment (IDE) for the Android platform since 2015. It was made by Google and powered by IntelliJ. Android Studio supports all the same programming languages of IntelliJ, and CLion e.g. Java (programming language), and C++; and Android Studio 3.0 or later supports Kotlin and Java 7 language features and a subset of Java 8 language features that vary by platform version. Android Studio latest version 3.2.1 was used to develop this project.

### 7.3.2 Android SDK

A software development kit that enables developers to create applications for the Android platform. The Android SDK includes sample projects with source code, development tools, an emulator, and required libraries to build Android applications. Applications are written using the Java programming language and run on Dalvik, a custom virtual machine designed for embedded use which runs on top of a Linuxkernel. Minimum SDK version of this app is 17

### 7.3.3 Google Maps Android API Utility Library

For custom markers on map and distance measurement I used Google Maps Android API Utility Library. Google Maps Android API Utility Library is an open-source library of classes that are useful for a range of applications.

### 7.3.4 LocationManager

For real-time device location I used LocationManager class. This class provides access to the system location services. These services allow applications to obtain periodic updates of the device's geographical location, or to fire an application-specified Intent when the device enters the proximity of a given geographical location.

# Chapter 8: Testing

## 8.1 Test Plan Identifier

The Test Plan has been created to communicate the test approach to readers. This is the Master Test Plan of my project. As the software will be released once I will stick to the master test plan. The id of this test plan is LifeBlood-bsse712-v1.0.

## 8.2 Introduction

LifeBlood is an android application which tries to handle and simplify the incident of blood finding manual approach. It introduces google maps and real-time location to sort out donors more efficiently and effectively. The app was divided into following modules:

* User Registration
* User Authentication
* User Profile
* Updating account setting
* Blood Request Raising
* Getting nearer donors’ list
* Calling specific donor
* Marking the donors’ position on custom google maps
* Getting user’s own location and showing on custom google maps

This test plan describes the testing approach and overall framework that will drive the testing of the LifeBlood-bsse712-v1.0.

## 8.3 Items to be Tested

“LifeBlood” the android application is the only test item to be tested here as no independent system are connected with the system.

## 8.4 Features to be Tested

1. Registration: User getting registered using email and password.
2. Authentication: User getting authenticated using email and password.
3. Account Settings: After registration user must have set profile image, profile name, phone number and blood group. Besides he/she can edit these things any time.
4. User Profile: After authenticated user sees his/her profile image, name, phone number and blood group on screen.
5. Blood Request Raising: Authenticated user make blood request giving hospital name, request description and blood group.
6. Getting nearer donors’ list: According to user’s current position, user gets nearer donors list in sorted order. This sort is according to distance measurement. This list contains every donors’ profile image, profile name, phone number.
7. Calling specific donor: User can directly call any of the donor from the list.
8. Donors’ position on map: User can see the donors’ current or most recent position on a custom google maps.
9. Own location on map: Seeing own location on custom google maps.
10. Logout: Logout from the account.

## 8.5 Reference to Related Documents

I have used Software Requirement Specification and Design document of this project for this test plan.

## 8.6 Approach

1. JUnit has been used for the unit testing of the application. *JUnit* is a unit testing framework for the Java programming language. *JUnit* has been important in the development of test-driven development. It has been used for the unit testing part.
2. Instrumentation tests are used for testing Android Frameworks such as UI, SharedPreferences and so on. Espresso has been used for the instrumented testing. Espresso is an instrumentation Testing framework made available by Google for the ease of **UI Testing**.
3. Manual testing has been done on the application.

## 8.7 Item Pass/ Fail Criteria

Stipulating the criteria that I will use to determine whether each test item of my project has passed or failed. The planning criteria gives the framework for how the project will be assessed and under what circumstances it will be released.

## 8.8 Suspension Criteria and Resumption Requirements

Suspension criteria will be used when it is needed to suspend all or a percentage of the testing activities when the testing has no value and the build is not working properly which is overall a wasting of resources. On the other hand, resumption criteria specify when testing can resume after it has been suspended. These will be applied in such situations.

Suspension when,

1. Unavailability of external dependent systems during execution.
2. A defect is introduced that cannot allow any further testing.

Resumption when

1. When the external dependent systems become available again.
2. When a fix is successfully implemented.

## 8.9 Test Deliverables

I will provide some deliverables after testing the application at the end of the project. These are given below:

1. Test plan document
2. Test cases

## 8.10 Environmental Need

1. Android device with minimum version 4.0 must be ensured to run and test the app.
2. Test data which will be provided is an environmental need as I need data to test. Except provided data I cannot perform any phase of testing.
3. I will perform a manual unit test, one phase of regression test on each module.

## 8.11 Testing Cost

As this is an academic project, so no need to estimate any testing cost

8.12 Test Cases

Table : Test Cases for Registration

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | **Test Cases** | **Preconditions** | **Input Test Data** | **Steps to Be Executed** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 01 | Test “Need an Account?” | Users not logged in |  | Click on the “Need an Account?” | Redirected to the registration page | Redirected to the registration link | Pass |
| 02 | Test with empty fields | Users not logged in |  | 1. Leave all field in the registration page empty  2. Click “Create Account” button. | Show warning message | Show warning message | Pass |
| 03 | Test with one empty field | Users not logged in | Provide value to all fields except one | 1. Enter all field except one field in the registration page  2. Click “Create Account” button. | Show warning message | Show warning message | Pass |
| 04 | Test with valid fields | Users not logged in | Provide valid values on all fields | 1.Enter all valid field in the registration page  2. Click “Create Account” button. | Redirected to the main page | Redirected to the main page | Pass |

Table : Test Cases for Authentication

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Cases** | **Preconditions** | **Input Test Data** | **Steps to Be Executed** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 01 | Test “Log into Account” | Users not logged in |  | Click on the app icon | Redirected to the login page | Redirected to the login page | Pass |
| 02 | Test with empty email and password field | Users not logged in |  | 1. Leave the email and password field in the login page empty  2. Click “Log into Account” button. | Show warning message | Show warning message | Pass |
| 03 | Test with one empty field | Users not logged in | Provide either the email or password value | 1. Enter either email or password value and leave the other one empty  2. Click “Log into Account” button. | Show warning message | Show warning message | Pass |
| 04 | Test with valid fields | Users not logged in | Provide valid email and password | 1.Enter all valid field in the login page  2. Click “Log into Account” button. | Redirected to the main page | Redirected to the main page | Pass |

Table : Test Cases for Account Settings

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Cases** | **Preconditions** | **Input Test Data** | **Steps to Be Executed** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 01 | Test “Account Settings” | Users logged in |  | Click on the option “Account Settings” in the menu | Redirected to the “Account Settings” page | Redirected to the “Account Settings” page | Pass |
| 02 | Test with all field empty | Users not logged in |  | 1. Leave all field in the account settings page empty  2. Click “Save Settings” button. | Show warning message | Show warning message | Pass |
| 03 | Test with one empty field | Users not logged in | Provide all field except any of the one: photo, name, phone number, blood group | 1. Enter all field except any of the one: photo, name, phone number, blood group  2. Click “Save Settings” button. | Show warning message | Show warning message | Pass |
| 04 | Test with valid fields | Users not logged in | Provide photo from gallery, name, phone number, blood group | 1.Enter photo from gallery, name, phone number, blood group  2. Click “Save Settings” button. | Redirected to the main page | Redirected to the main page | Pass |

Table : Test Cases for User Profile

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Cases** | **Preconditions** | **Input Test Data** | **Steps to Be Executed** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 01 | Test “User Profile” | Users logged in |  | Logged in to an existing account | Showing profile photo, name, phone number, blood group | Showing profile photo, name, phone number, blood group | Pass |

Table : Test Cases for Blood Request Raising

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Cases** | **Preconditions** | **Input Test Data** | **Steps to Be Executed** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 01 | Test “Make a Blood Request” | Users logged in |  | Click on the “Make a Blood Request” button | Redirected to the “Add Blood Request Page” page | Redirected to the “Add Blood Request Page” page | Pass |
| 02 | Test with empty field | Users logged in | Provide all field empty | 1. Leave all the field in the page empty  2. Click “Confirm Request” button. | Show warning message | Show warning message | Pass |
| 03 | Test with one empty field | Users logged in | Provide all field except one | 1. Fill up all the field except one  2. Click “Confirm Request” button. | Show warning message | Show warning message | Pass |
| 04 | Test with valid fields | Users logged in | Provide all field | 1.Fill up all field  2. Click “Confirm Request” button. | Redirected to the “Donors List” page | Redirected to the “Donors List” page | Pass |
| 05 | Test “Cancel Request” | Users logged in |  | Click on the “Cancel Request” button | Redirected to the main page | Redirected to the main page | Pass |

Table : Test Cases for Getting nearer donors’ list:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Cases** | **Preconditions** | **Input Test Data** | **Steps to Be Executed** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 01 | Test “Donors List” | Make a successful blood request |  |  | Showing list of users with profile image, name and phone number | Showing list of users with profile image, name and phone number | Pass |

Table : Test Cases for Calling specific donor

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Cases** | **Preconditions** | **Input Test Data** | **Steps to Be Executed** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 01 | Test “Calling Donor” | Make a successful blood request |  | Click on the phone icon on a specific donor from the list | Phone call to the donor | Phone call to the donor | Pass |

Table : Test Cases for Donors’ position on map

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Cases** | **Preconditions** | **Input Test Data** | **Steps to Be Executed** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 01 | Test “Donors Map” | Make a successful blood request |  |  | Showing marker on donors’ positions on map | Showing marker on donors’ positions on map | Pass |

Table : Test Cases for Own location on map

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Cases** | **Preconditions** | **Input Test Data** | **Steps to Be Executed** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 01 | Test “Own Location on Map” | Users logged in |  | Click on “Show on Map” button | Showing marker on user’s position on map | Showing marker on user’s position on map | Pass |

Table : Test Cases for Logout

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **ID** | **Test Cases** | **Preconditions** | **Input Test Data** | **Steps to Be Executed** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 01 | Test “Logout” | Users logged in |  | Click on “Logout” option from the menu | Redirecting to the login page | Redirecting to the login page | Pass |

# Chapter 9: User Manual

A user guide is commonly known as a user manual. User manual is usually prepared by a technical person like developer or others who technically involved in software project. User manual helps a non-technical person to run a system and make easy to use it at first time.

User  
should know that this app requires minimum SDK level 17 and Android version 5.0 (Lollipop). It will work properly on higher Android versions, such as- Marshmallow, Nougat or Oreo. The app works offline which means it does not require internet connection. User manual is explained here with necessary snapshots of the current version of the app.

On start up the following splash screen is shown:



Figure : User Manual: Splash Screen

## 9.1 Login

User can get logged in providing valid email and password. Besides user can go to Registration Page by clicking “Need a new Account?”.

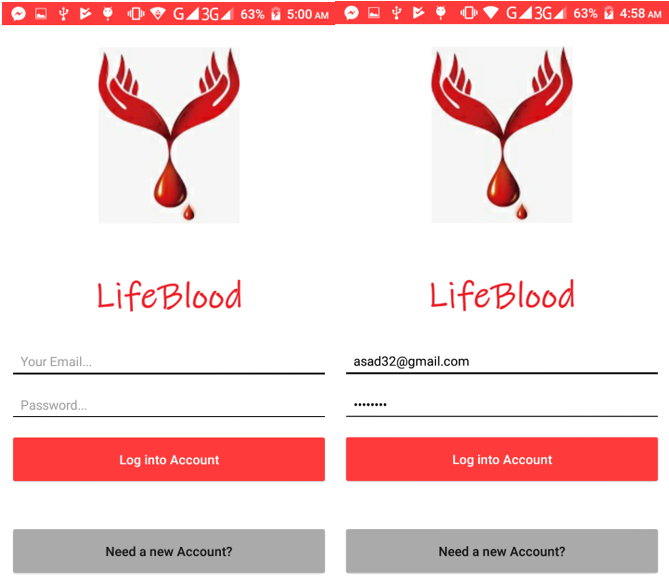


Figure : User Manual: Login

## 9.2 Registration

User can get registered providing email and password. Besides user can go to Login Page again by clicking “Already have an Account?”.

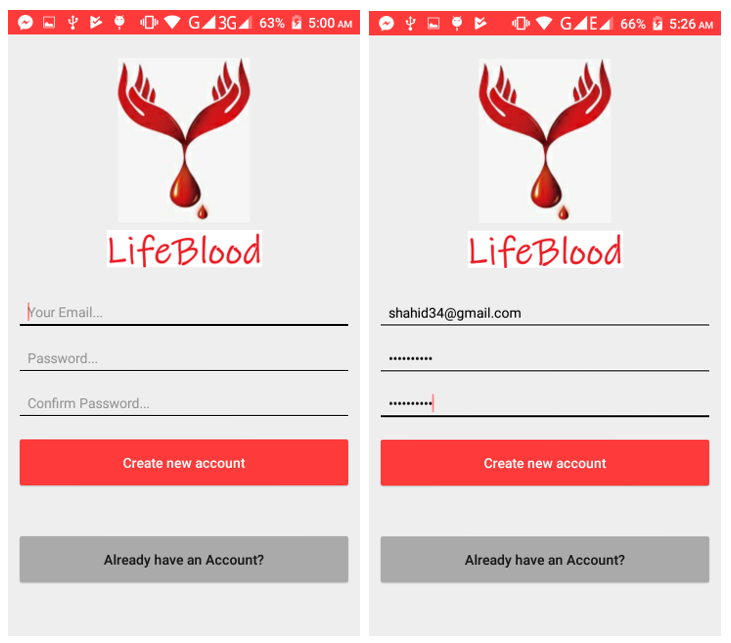


Figure : User Manual: Registration

## 9.3 Account Settings

After Registration a user must have set his/her profile image, profile name, phone number and blood group. After that h/she can save his/her account settings by clicking “Save Settings” button.

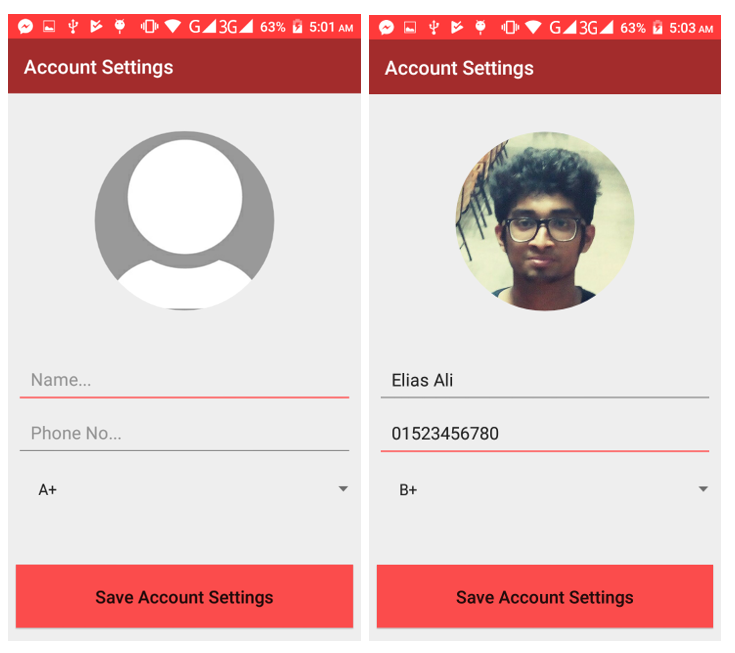


Figure : User Manual: Account Settings

## 9.4 Main Page

After account settings completed users is redirected to main page. Besides if a user logged into his/her account h/she also get this main page. This main page contains several buttons and a menu bar icon. A user can see his/her profile image, profile name, phone number and blood group.

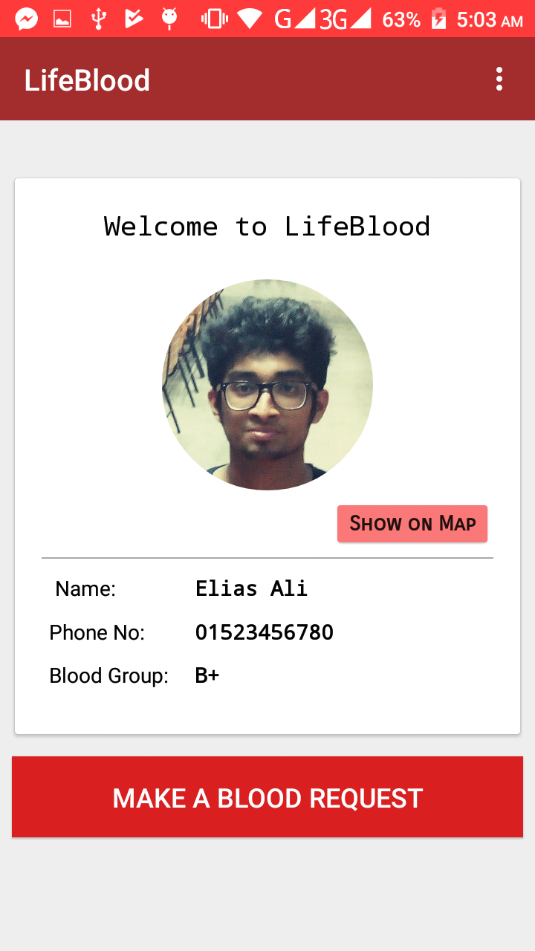


Figure : User Manual: Main Page

## 9.5 My Location

By clicking on “Show on Map” from main page, one gets his current location on map.



Figure : User Manual: My Location

## 9.6 Blood Request

By clicking on “Make A Blood Request” from main page, one is redirected to this “Add Blood Request” page. In this page h/she can raise a blood request providing blood group, hospital name and details. After clicking “Confirm Request” a user gets the donors list. Accordingly, by clicking “Cancel Request” h/she is redirected to the main page.

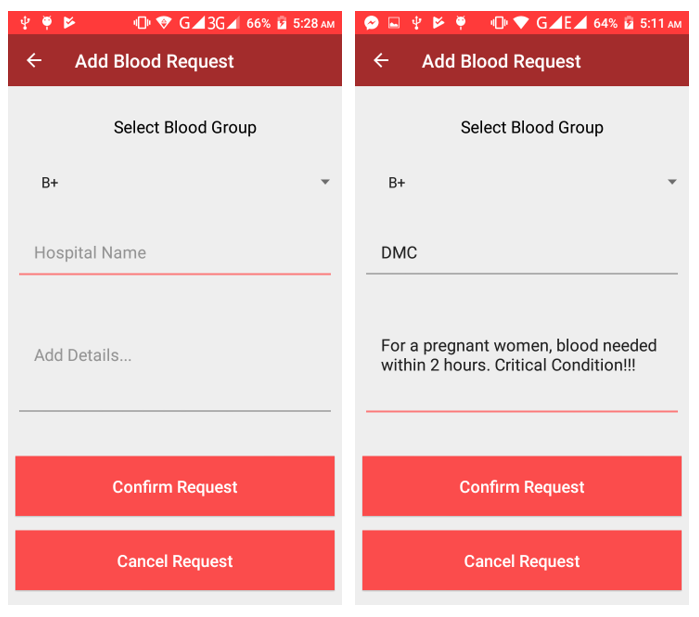


Figure : User Manual: Blood Request

## 9.7 Nearer Donors List, Location and Calling

By confirming request, a user gets the nearer donor list. This list is sorted according to distance from the user. From the list user can call any of the donor by clicking on the call icon. Besides a map view of all the donors from the list is also shown below.

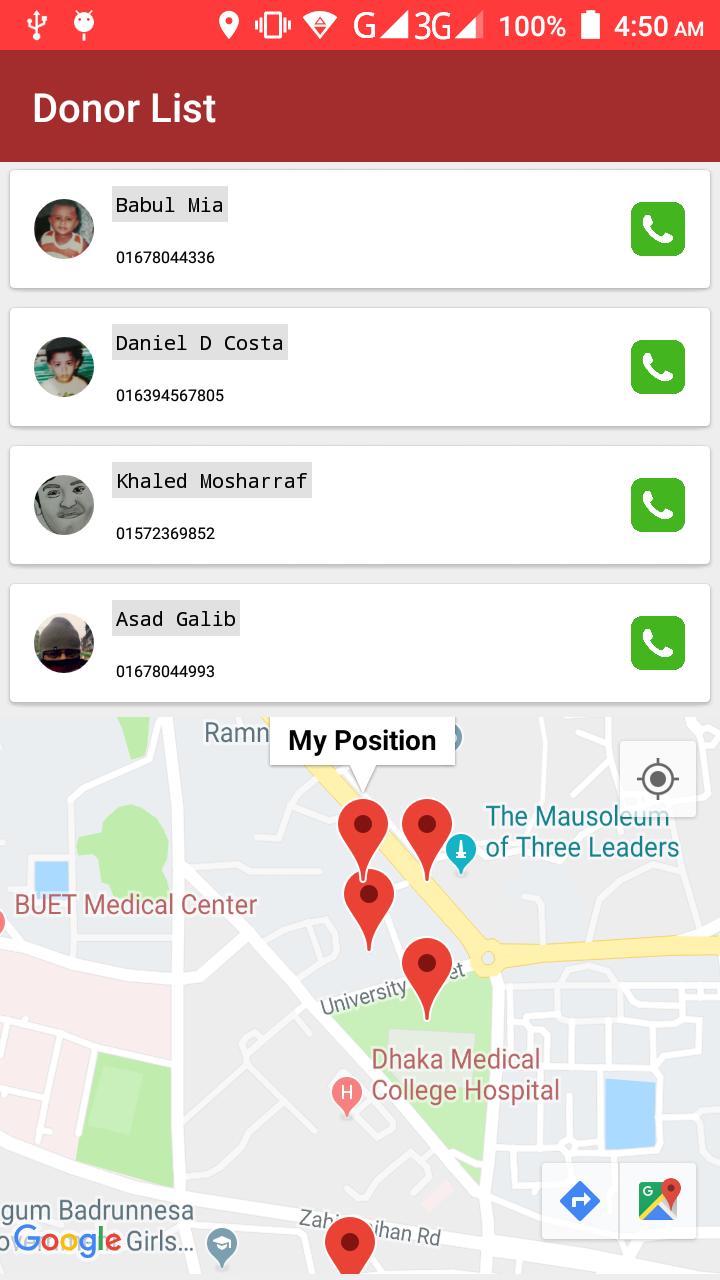


Figure : User Manual: Donor List

## 9.8 Logout

By clicking on “Logout” from the menu bar options, user can logout from the app. Besides by clicking on “Account Settings” h/she is redirected to Account Settings page.

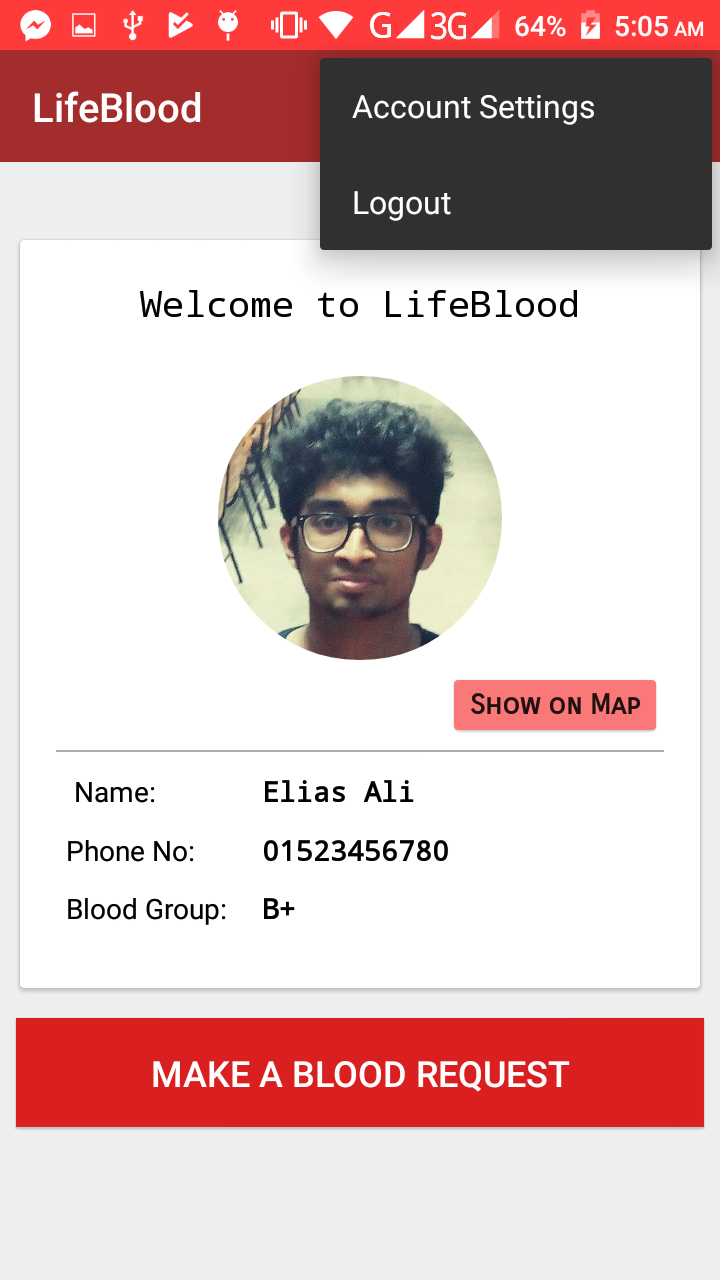


Figure : User Manual: Logout

# Chapter 10: Conclusion

This report’s requirement specification and design document draw a path for implementing the application. Besides it includes the real implementation overview, testing plan and user manual.

The primary aim of this report is to fulfill the Software Project Lab 3 course objective. Though there are many restrictions and constraints, this project has finished successfully. My future goal is to update this app according users’ reviews and debug those unseen bugs.