# LifeBlood

**(A Blood Donation System Android App)**

## Technical Report

**BSSE 7th Batch**

**Institute of Information Technology**

| | |
|---|---|
| Submittted to | SPL 3 Committee<br>Institute of Information Technology<br>University of Dhaka |
| Submittted by | Asadullah Hill Galib BSSE0712 |
| Supervised by | Dr. Mohammed Shafiul Alam Khan<br>Associate Professor<br>Institute of Information Technology<br>University of Dhaka |

Submission date: 29th September 2018

# Letter of Transmittal

29th September 2018

Dr. Mohammed Shafiul Alam Khan

Associate Professor

Institute of Information Technology

University of Dhaka

**Subject: Submission of Technical report on “LifeBlood – A blood donation system android app”.**

Sir,

With due respect, I am submitting the technical report on the above project. In this report, I have given our best effort albeit some shortcomings. I earnestly hope that you would excuse our errors and oblige thereby.

Yours sincerely

Asadullah Hill Galib BSSE0712

4th Year, 8th Semester, 7th Batch

Institute of Information Technology

University of Dhaka

Session: 2014-15

## Acknowledgement

I am highly indebted for getting such a tremendous opportunity to prepare the technical report on **LifeBlood –** a blood donation system android app. I would like to thank whole-heartedly my supervisor, Dr. Mohammed Shafiul Alam Khan, Associate Professor, Institute of Information Technology, University of Dhaka, for giving me guidelines about how I can prepare this report.

## 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 of **LifeBlood-** a blood donation system android app.

# Contents

***Table of Figures:***

# 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. 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 of LifeBlood

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 Eliciting Requirements

We have seen Question and Answer (Q&A) approach in the previous chapter, where the inception phase of requirement engineering has been described. The main task of this phase is to combine the elements of problem solving, elaboration, negotiation and specification. The collaborative working approach of the stakeholders is required to elicit the requirements. We have finished the following tasks for eliciting requirements-

- Collaborative Requirements Gathering
- Quality Function Deployment
- Usage Scenarios
- Elicitation work products

### 2.2.1 Collaborative Requirements Gathering

We have met with many stakeholders in the Inception phase such as many people and developers. These meetings created an indecisive state for us to elicit the requirements. To solve this problem, we have met with the stakeholders (who are acting a vital rule in the whole

process) again to elicit the requirements. A slightly different scenario from these approaches has been found.
Following activities have been completed to accomplish this task.

1. A google form survey was being performed to acknowledge about requirements.
2. They were asked about the problems they were facing with the current existing system.
3. Lastly, we selected our final requirement list from the meetings

### 2.2.2 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.

#### 2.2.2.1 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 through map
4. Getting nearest and accepted blood donors' list and details
5. Viewing donor's profile

#### 2.2.2.2 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. Efficient search through map
3. Allow only valid users to login.

### 2.2.2.3 Exciting Requirements

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

1. Providing personal account for user
2. Efficient search through map

## 2.2.3 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 who can make request for blood. She can also become a blood donor by activating it after registration. A donor is both a donor and receiver.

**b. Authentication**

User can log in to her account using username 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, reviews, health status, ranking, availability status. Ranking, health status and availability status are only applicable for the donor. Each user can see other donor’s profile data.

**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 patient’s physical status, level of emergency, expected time of arrival, location etc. via that request. This request will search through the customized map to sort out nearer donors of that desired blood group. If any nearer donor accepts the request, receiver will then connect to the donor via phone call.

**e. Donor**

Donor is one who donates the blood. A user can become a blood donor by simple activation. While activating, donor must give her health status and availability status. Donor gets notification from nearer receiver's request. S/he can accept or reject the request. After accepting the request, receiver will get a notification of acceptance. And then receiver can connect with the donor via phone call. A donor can also give review on a receiver.

**f. Ranking System**

After a donation, receiver can rate the donor on a scale of 5. A ranking system will be developed according to the rating. This may appreciate best and effective donor and encourages other donors.

# 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.

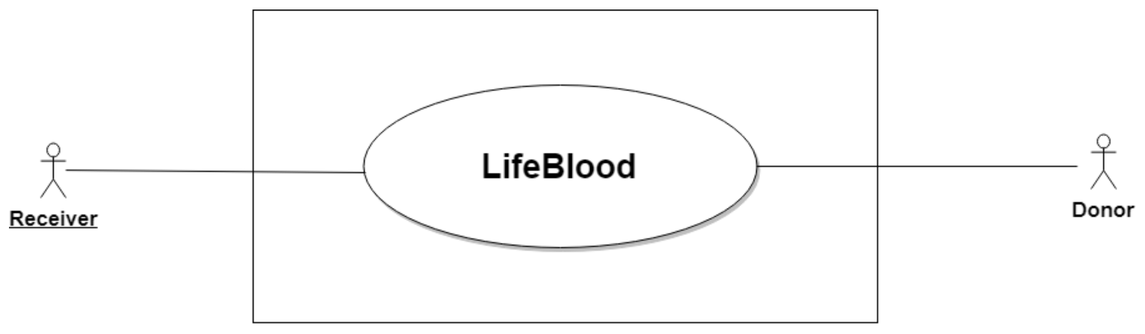
**Secondary Actor**

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



Level 0: LifeBlood

**Name:** LifeBlood

**ID:** Level-0 Use Case

**Primary Actors:** Receiver, Donor

**Secondary Actors:** None

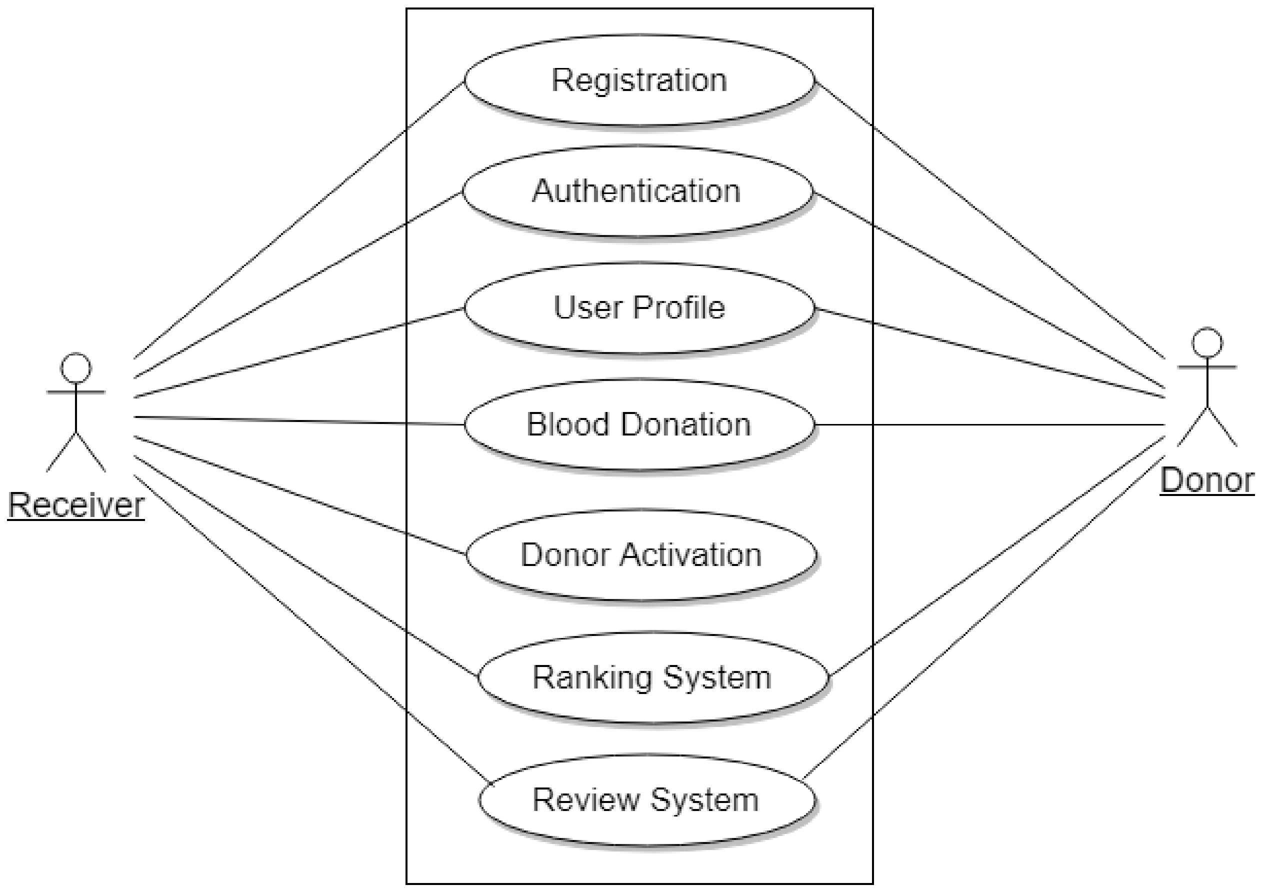
**Description of Level 0 Use Case Diagram:**

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



**Level 1: Subsytems**

**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

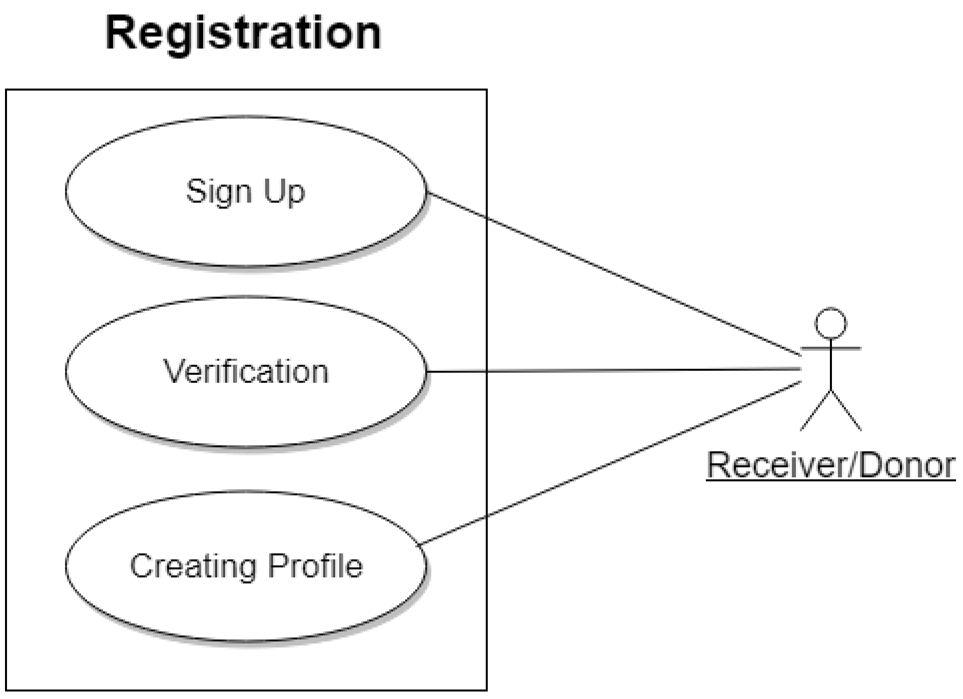
Reply 13: Profile created

**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



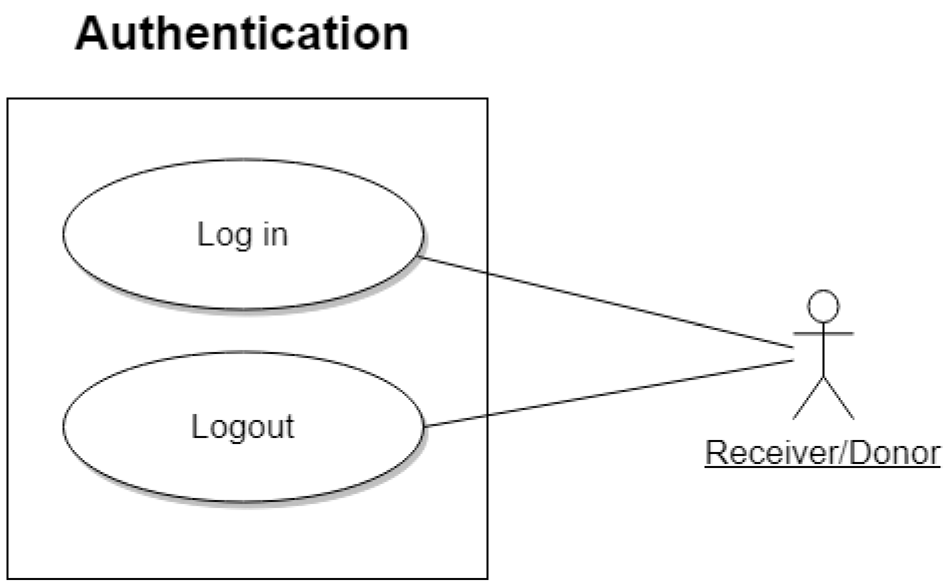
Level 1.1: Registration

**Name:** Registration

**ID:** Level-1.1 Use Case

**Primary Actors:** Receiver, Donor

## 3.3.4 Level-1.2 Use Case Diagram-Authentication



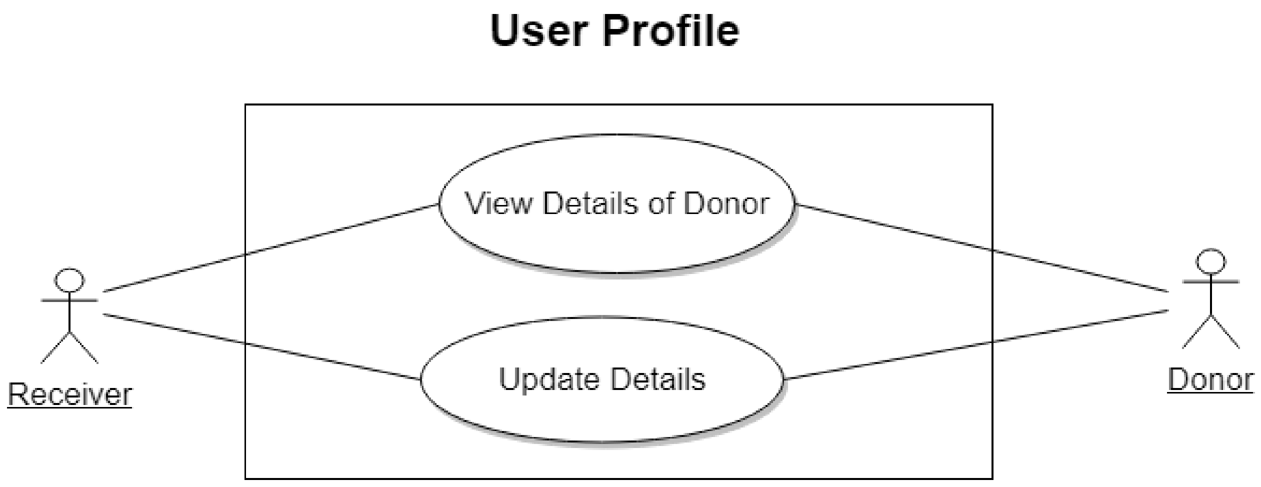
**Level 1.2: Authentication**

**Name:** Authentication

**ID:** Level-1.2 Use Case

**Primary Actors:** Receiver, Donor

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



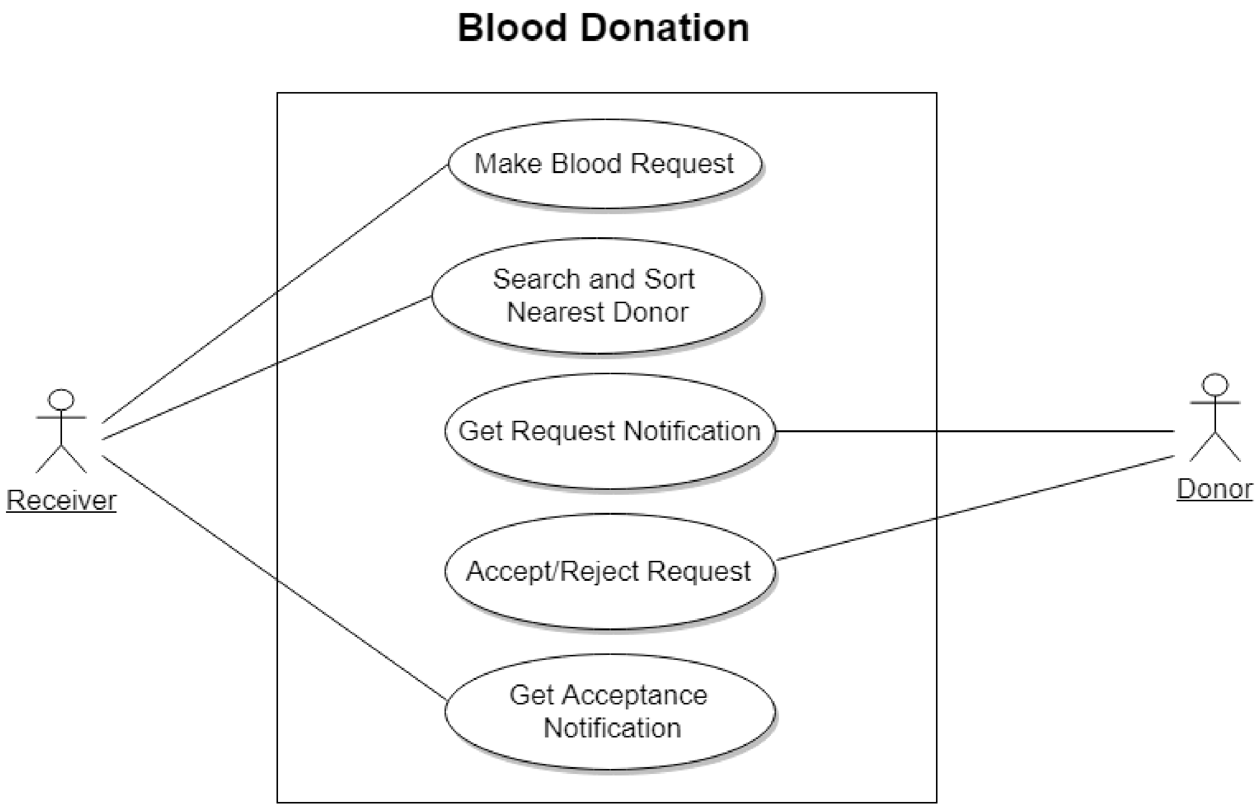
**Level 1.3: User Profile**

**Name:** User Profile

**ID:** Level-1.3 Use Case

**Primary Actors:** Receiver, Donor

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



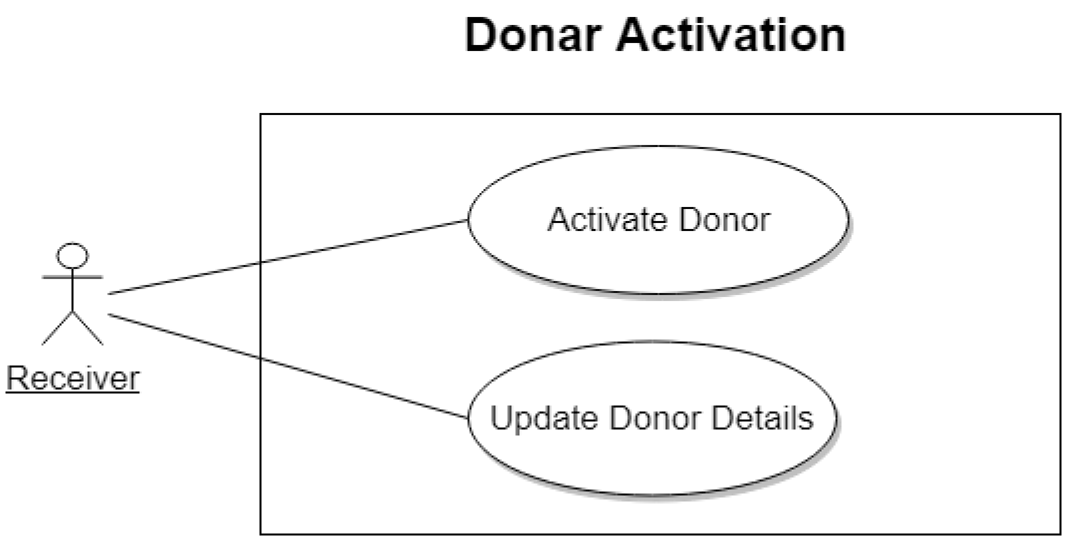
**Level 1.4: Blood Donation**

**Name:** Blood Donation

**ID:** Level-1.4 Use Case

**Primary Actors:** Receiver, Donor

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



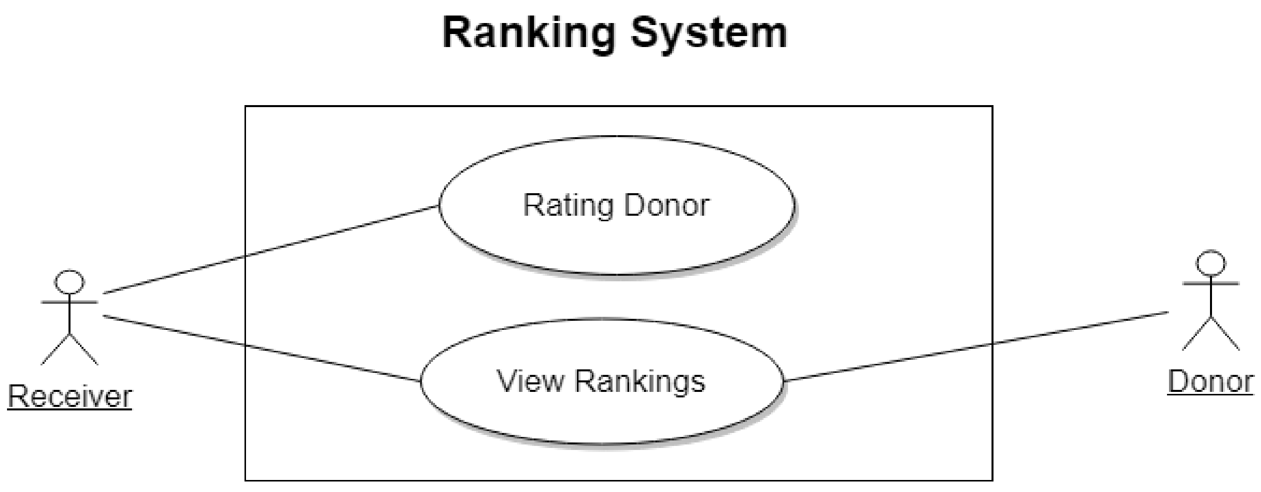
**Level 1.2: Donor Activation**

**Name:** Donor Activation

**ID:** Level-1.5 Use Case

**Primary Actors:** Receiver, Donor

## 3.3.8 Level-1.6 Use Case Diagram-Ranking System



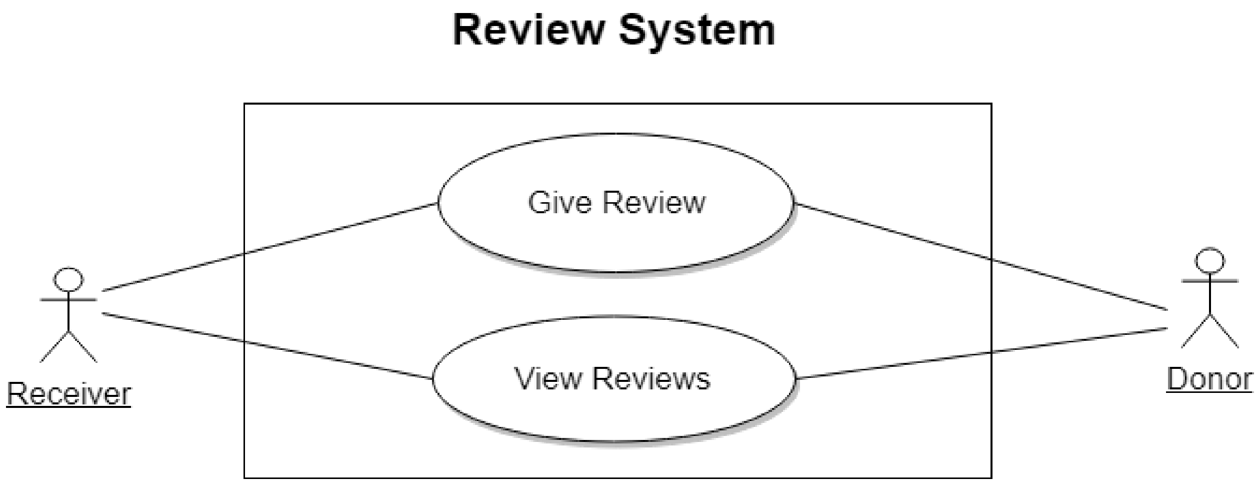
**Level 1.6: Ranking System**

**Name:** Ranking System

**ID:** Level-1.6 Use Case

**Primary Actors:** Receiver, Donor

## 3.3.9 Level-1.7 Use Case Diagram-Review System



**Level 1.7: Review System**

**Name:** Review System

**ID:** Level-1.7 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, Review, Notification, Health Status, Ranking, Availability status, Rating |
| 3 | Blood Request | **Request ID**, Receiver ID, Donor ID, Name, Mobile Number, Blood Group, Patient status, Level of emergency, Expected time of arrival |

## 4.7 Schema Table

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

| **2. Donor** | | |
|---|---|---|
| **Attributes** | **Types** | **Size** |
| User ID | NUMBER | 5 |
| Username | VARCHAR2 | 25 |
| Password | VARCHAR2 | 20 |
| Name | VARCHAR2 | 40 |
| Age | VARCHAR2 | 10 |
| Blood Group | VARCHAR2 | 10 |
| Mobile number | VARCHAR2 | 15 |
| Address | VARCHAR2 | 80 |
| Review | VARCHAR2 | 512 |
| Notification | VARCHAR2 | 40 |
| Health Status | VARCHAR2 | 80 |
| Ranking | VARCHAR2 | 10 |
| Availability status | VARCHAR2 | 40 |
| Rating | VARCHAR2 | 10 |

| **3. Blood Request** | | |
|---|---|---|
| **Attributes** | **Types** | **Size** |
| **Request ID** | NUMBER | 5 |
| Receiver ID | NUMBER | 5 |
| Donor ID | NUMBER | 5 |
| Name | VARCHAR2 | 40 |
| Mobile Number | VARCHAR2 | 15 |
| Blood Group | VARCHAR2 | 10 |
| Patient status | VARCHAR2 | 80 |
| Level of emergency | VARCHAR2 | 200 |
| Expected time of arrival | DATETIME | 30 |

## 4.8 Relationship among Data Objects

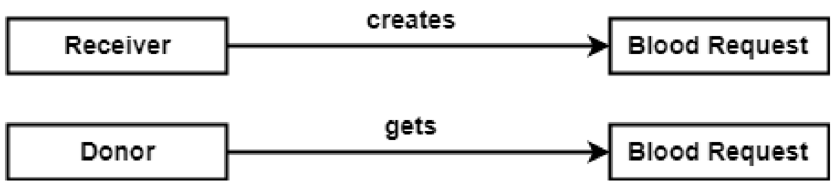


*Figure 1: Relationship among Data Objects*

## 4.9 ER Diagram

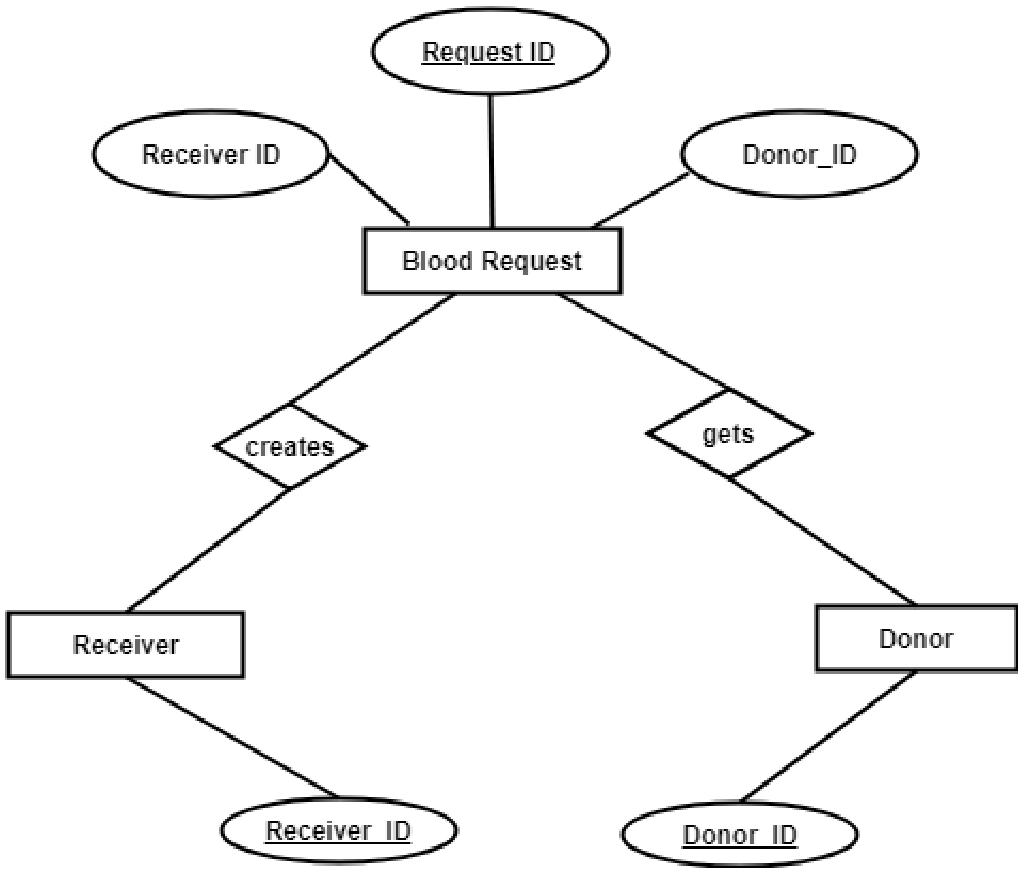


*Figure 2: ER Diagram*

# 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 + | callRequest() + searchMap() + sortDonors()+ getAcceptanceNotification() |

| | | | |
|---|---|---|---|
| | | mobileNumber + address + review + notification + healthStatus + ranking + availabilityStatus + rating | + 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
   a. 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 + | checkIfUserNameAvailable()+ takeInput()+ signUp()+ profileCreation() |

| | | | |
|---|---|---|---|
| | | mobileNumber + address | + 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<br>Getting authenticated | Receiver<br>Receiver |

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

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

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

## 5.11 CRC Diagram

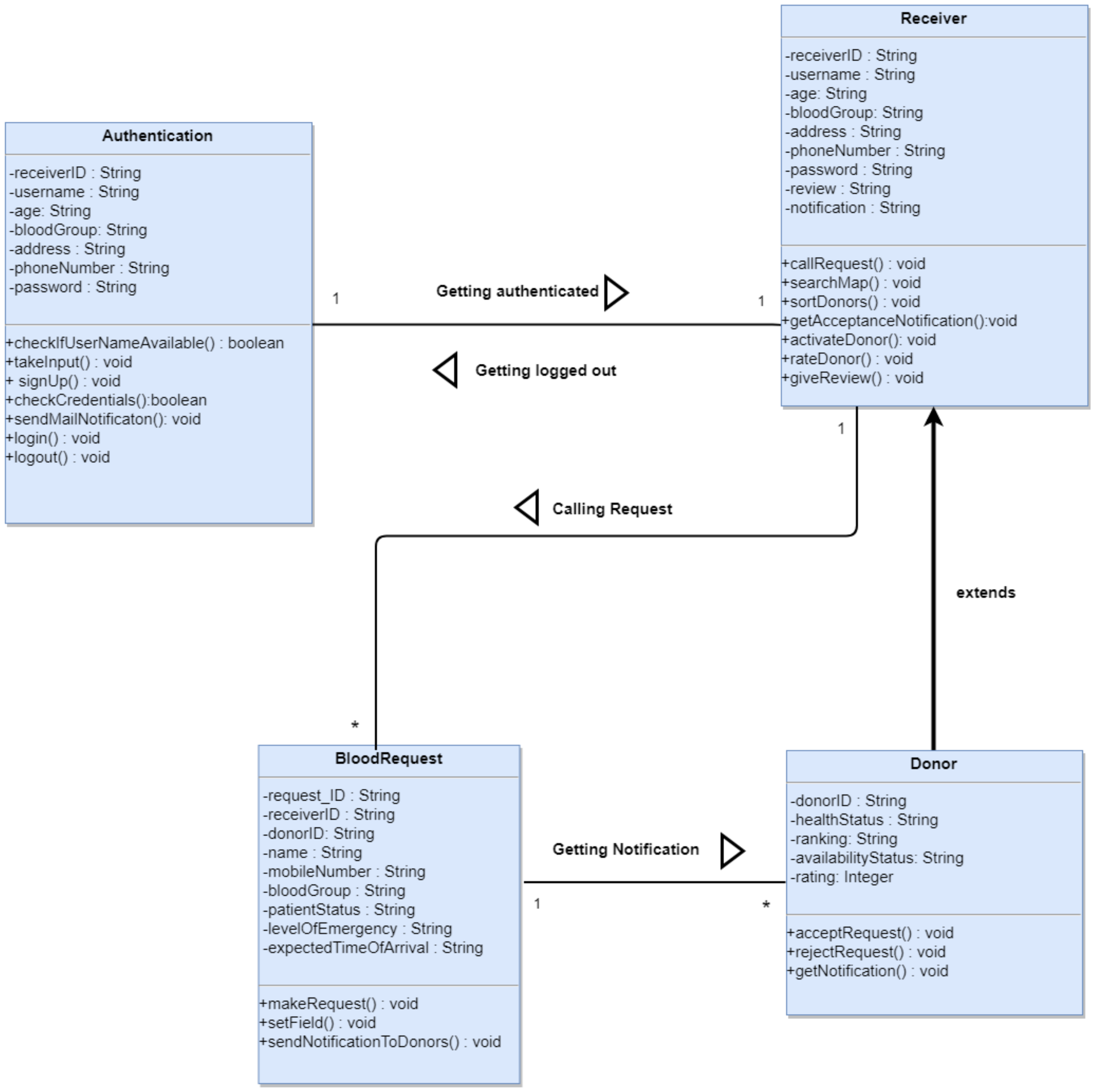


*Figure 3: CRC Diagram*

# 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.

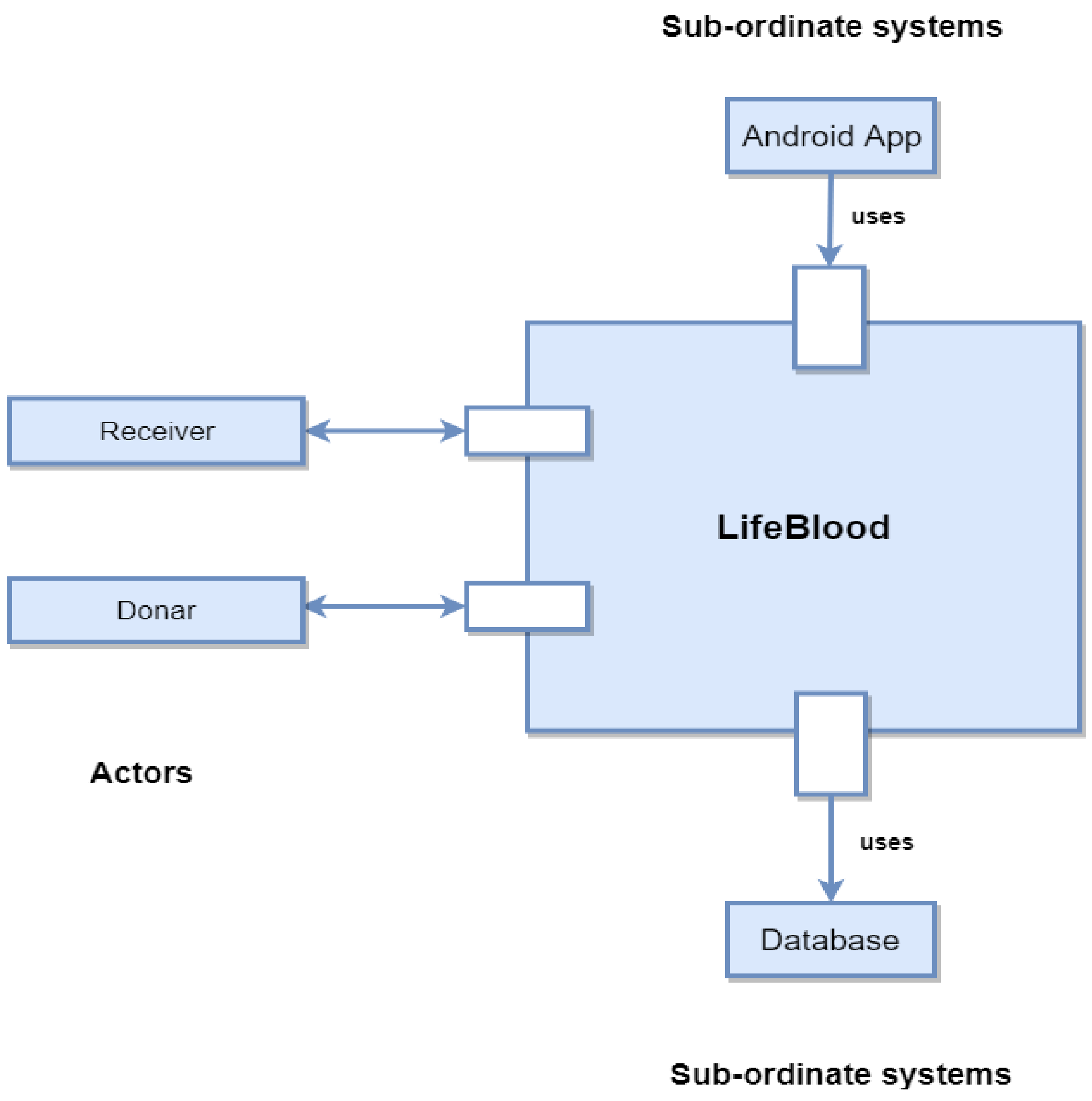


*Figure 4: Architectural Context Diagram*

## 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.

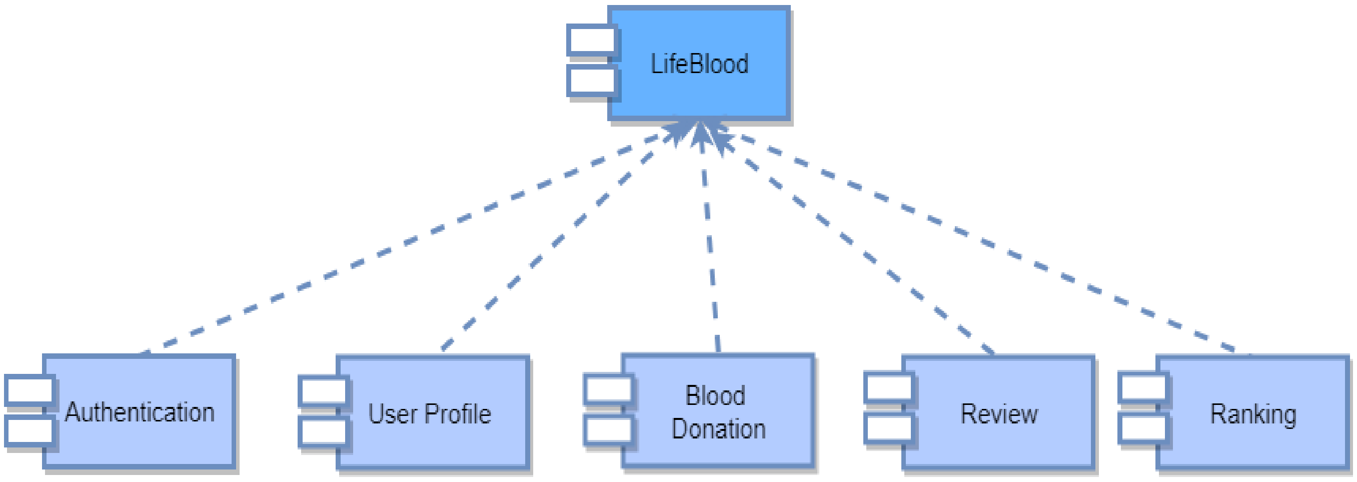


*Figure 5: System Components*

## 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.

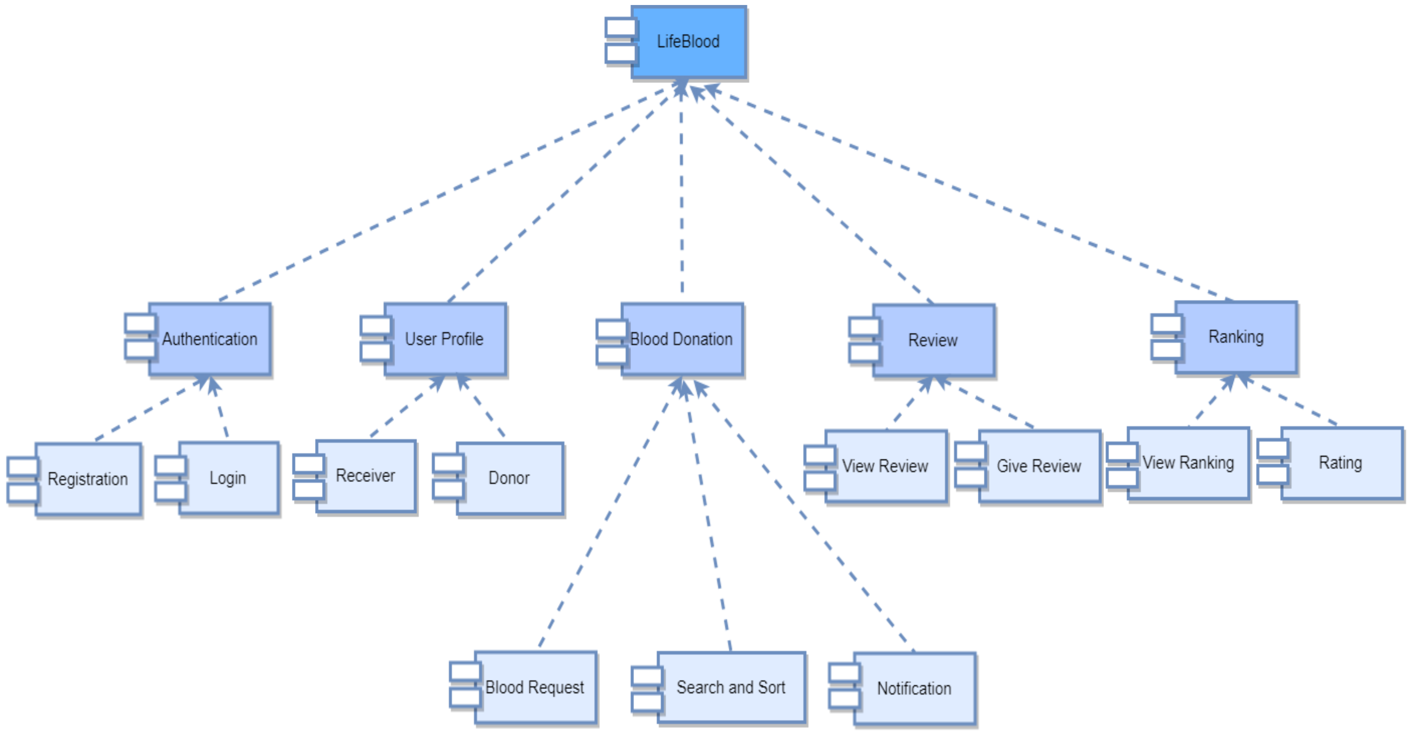


*Figure 6: Componet Elaboration*

# Chapter 7: Conclusion

This technical report draws a path for implementing the application. With the help of this technical report, any developer can understand the whole system requirement and design. So, it acts as a guideline for the developers who will build the real application. Besides, this technical report also helps the quality assurance team to check whether the built system is valid or not.