BlooDonate

Connecting Blood Donors and Recipients

Final Year Project Report

by

Hamna Zahid

Maryem Fatima

Menobia Arif

In Partial Fulfillment

Of the Requirements for the degree

Bachelors of Engineering in Software Engineering (BESE)

School of Electrical Engineering and Computer Science

National University of Sciences and Technology

Islamabad, Pakistan

(2018)

DECLARATION

We hereby declare that this project report entitled “BlooDonate” submitted to the SEECS is a record of an original work done by us under the guidance of Supervisor Dr Muhammad Ali Tahir and that no part has been plagiarized without citations. Also, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelors of Engineering in Software Engineering**.**

Team Members **Signature**

Hamna Zahid ______________

Maryem Fatima ______________

Menobia Arif ______________

Supervisor **Signature**

Dr Ali Tahir ______________

Date:

____

Place:

____

Dedicated to

Allah Almighty

&

Our Parents and Teachers

ACKNOWLEDGEMENTS

We thank Dr. Muhammad Ali Tahir our project advisor, Assistant professor at National University of Sciences and Technology Islamabad, for all his kind support throughout the project development phase.

We also like to thank Amazon development team because their products make the whole project a lot easier and efficient.

We also acknowledge the help and support of our Co-Advisor Dr. Muneeb Ullah, Assistant professor at National University of Sciences and Technology, Islamabad throughout the year.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

BlooDonate is an android application that will make the process of blood donation process easier and efficient by connecting donors and recipients. It is a solution to a problem of increased blood demand especially in emergency situations as it is said that after every two seconds someone is in need of blood and death rate has been increased due to unavailability of blood in emergency situations. There exist some solutions including blood banks, websites and applications that provide similar functionality but they do not involve any kind of authentication from user side. Moreover, they just provide recipient a list of donors and he himself has to contact each and every donor to ask for their availability. Thus our application is designed to cater all the problems faced by a recipient when he is in need of blood in emergency situations

Chapter 1

INTRODUCTION

Every now and then we hear that blood is urgently required by someone of some type at some place and when someone needs it so bad he is just contacting every possible person they can think of not knowing their blood type or whether they can easily get it from them or not.

Blood Donation Application, as the name suggests is an Android application which aims to ease the blood donation procedure. This is an application which will provide a platform where donors and recipients can communicate efficiently and effectively. Donors will be registered based on their blood type and their location which will make it easy for the recipients to find blood of the type that they want and within their reach. Recipient will simply request blood mentioning the type, quantity and their current location and based on these attributes, donors of nearby locations and matching blood type will be notified. On receiving the notification, if a donor accepts the request he will be given an instant blood donation eligibility form (to verify if he can donate blood right now or not) which will include checks like last blood donation date, on medication or not or having any disease at the moment or not etc. Once a donor is verified, the recipient is informed of his availability and then the recipient gets to choose one of the eligible donors that has accepted the request. This way the recipient can select a donor based on their rating and how near they are. Once a donor is selected, he is directly connected to the recipient and they can now communicate via regular call or SMS. Since the donor has to reach the recipient, he will be able to navigate to the recipient via navigation facility provided in our application. Once the donor donates the blood, the recipient has to rate him before he can request again this will complete the donation.

This project will help the people in need to get the required blood in limited amount of time. They will not have to waste time calling people whose blood type doesn’t even match or who are far from their reach. Also, with the help of the eligibility check, it will help filter out the donors which are not eligible to donate blood which will eventually save the time of the donor and the recipient both. Therefore, we think it is of great value to the people of our community.

This chapter covered the introduction to our final year project in a non-technical term. The following chapters include literature review and problem definition which covers what is the previous work done and why have we done this project. It also includes our solution to the problem. Then it covers the detailed design and architecture of the system as well as the implementation and testing that we have done. After that is shows the results of the system and usability testing that we have performed finally it ends with the conclusion and future work that we have proposed.

Chapter 2

LITERATURE REVIEW

There are already some applications that provide similar functionality in many ways like they provide recipient a list of donors’ phone numbers on the basis of blood group and location only. Main limitations with existing solutions include many problems. For example

1. Recipient has to contact each and every donor to confirm their availability in the very emergency situation without even knowing whether he is eligible for the donation or not.
2. They don't require any authentication method. So there is a chance of false donors just to spam the system.
3. Around 60% of them do not notify user about the request. That’s why the result is fewer user.

In this project we aimed to solve the above mentioned problems by providing complete solution through an android application which will connect donor and recipient directly. Moreover, it has different eligibility checks which decides whether donor can donate or not. We have made reception process for recipient a lot easier than other apps where recipient had to put a lot of effort in finding blood.

Chapter 3

PROBLEM DEFINITION

We often hear that someone is in need of blood but due to absence of any specific platform the patient / recipient has to suffer a lot. He has to call everyone for their eligibility and availability. Applications that are present these days just provide user with a bunch of donors’ contact numbers and recipient has to call each and every one for their eligibility. This thing takes too time much that the recipient or patient who is in need of might lose his or his loved one’s life. Moreover, existing systems do not involve any kind of authentication method or eligibility checks. In short, we can say that there is an absence of a proper platform for blood donors and recipients to communicate their needs and get blood of required type effectively and efficiently in case of urgent situations.

Who needs it? How many would benefit? Blood requirement occurs mostly as a result of emergency situations which can involve anyone. Therefore, our application targets everyone and anyone in need of blood. According to a report 364/10,000 people donate blood in high-income countries, so we aim to reach at least this target.

Chapter 4

METHODOLOGY

4.1 MODULES

Our whole application comprises of many independent modules that combines to give full functionality of the system. These independent modules are as follows:

1. Live EC2 machine
2. Live database
3. Apache web server
4. Android mobile

4.1.1 Amazon EC2

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) cloud. It eliminates the need of investing in hardware upfront and in this way helps in faster development of applications. It allows you to scale up or down the machines according to your requirement.

4.1.1.1 Features of EC2:

- It provides virtual computing environments, that is an *instance.*
- You can configure by yourself the storage, CPU, memory and networking capacity for the instance.
- Secure login information for your instances using *key pairs* (AWS stores the public key, and you store the private key in a secure place)
- It also facilitates the storage of temporary data and deletes it when you stop of terminate the machine.
- It provides persistent storage using amazon elastic block store. (Amazon EBS)
- There are multiple physical locations for the resources which ensures security

- A firewall that enables you to specify the protocols, ports, and source IP ranges that can reach your instances using *security groups*

4.1.1.2 Features of Amazon database

It makes it easy to set up, operate and scale up relational database in cloud. It provides cost efficient solutions to users and allow them to resize their capacity when they need to. It frees you from any kind of hardware management so that you can solely focus on development and its security of the system. It provides you with 6 database engines that include Amazon Aurora, PostgreSQL, MySQL, Maria DB, Oracle, and Microsoft SQL Server. We are using MYSQL for our application.

4.1.2 MYSQL

It is the world's most popular open source database. It is used for either in the transactions of a small corporation or in large enterprises. It is cost-effectively help you deliver high performance, scalable database applications.

4.1.3 Apache Web Server

It is the most widely used web server software. Developed and maintained by Apache Software Foundation, Apache is an open source software available for free. It runs on 67% of all web servers in the world.

4.2 METHODS

Methods and techniques include HTTPS protocol for connection between android mobile app (User interface) and Apache server, which receives the messages using PHP code running on it, this sends queries written in SQL to MySQL Server which sends a reply which is turn sent back to the server which transfers it to the android user. Below figure sums up the connections between the above mentioned separate modules and how they communicate with each other. When a certain user requests for the blood data flow takes place which is summed up by the following diagram.

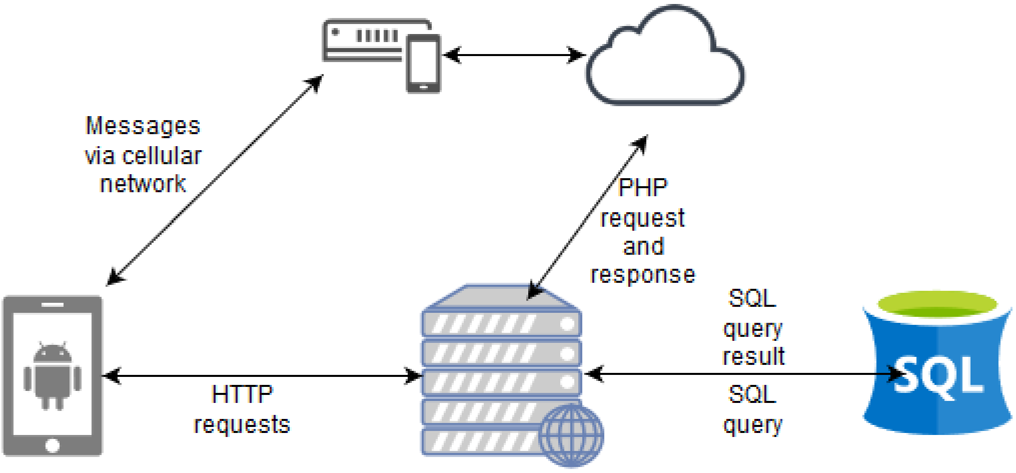

Figure 1: Blood Request Data Flow

4.3 TOOLS USED

- **Android Studio:** used for the front-end development of our application which includes all the screens and front end functionality. Google maps API for android was also used for the location, direction, driving distance and navigation services.
- **Firebase:** it is used for cloud messaging.
- **EC2 live virtual machine by Amazon Web Services:** A live running virtual machine where our server resides which handles all the requests sent to it at any time of the day.
- **Relational Database by Amazon Web Services:** Storage space provided by AWS where our data is stored as a relational database.
- **PHP:** The PHP files are used for the communication between the client (android app) and the server (Residing at EC2 virtual machine). These files are placed on Apache web server.

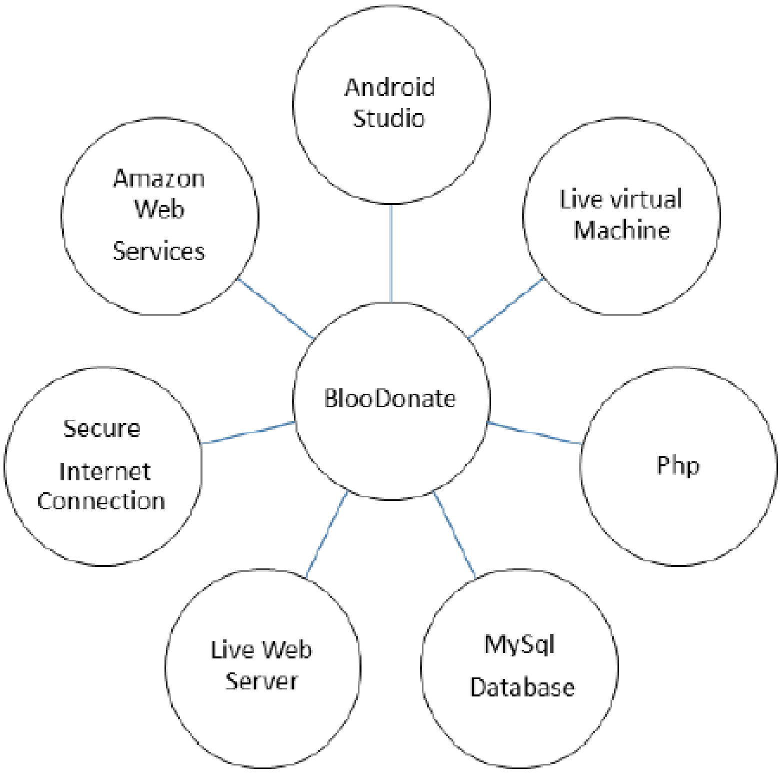

Figure 2: Tools Used

Chapter 5

DETAILED DESIGN AND ARCHITECTURE

5.1 SYSTEM ARCHIECTURE

5.1.1 Major Components

Components/ Software and their main functionality is mentioned as below:

1. **Amazon EC2:** We used amazon to get a live server for 24/7 blood required requests by recipients.
2. **Amazon database base:** Amazon database as Amazon EC2 was used for the 24/7 transaction of blood requests.
3. **Firebase:** Firebase is used for cloud messaging i-e to send targeted messages to donors or recipients.
4. **SMS gateway:** SMS gateway is used to send messages via cellular network to multiple donors. That in case if someone is not connected to internet he'll still receive notification via cellular network.
5. **Apache web server:** Our Amazon EC2 machine has apache web server installed on it.
6. **MySQL Workbench:** it is used to access database placed on amazon database servers.

Android Mobile: mobile with android operating system is required for the application.

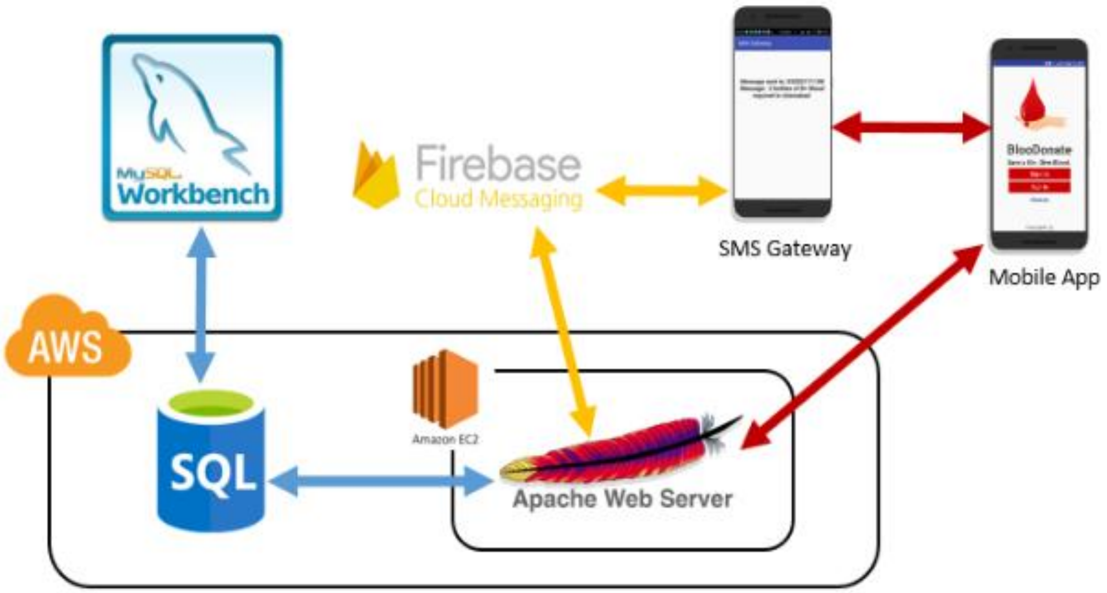

Figure 3: Components of the System

5.1.2 Architecture Design Approach

The system has been developed using the layered architectural and client server approach. The layers include database, application layer and the front end mobile application while the client is the user mobile application and the server is the running on the live EC2 machine using AWS.

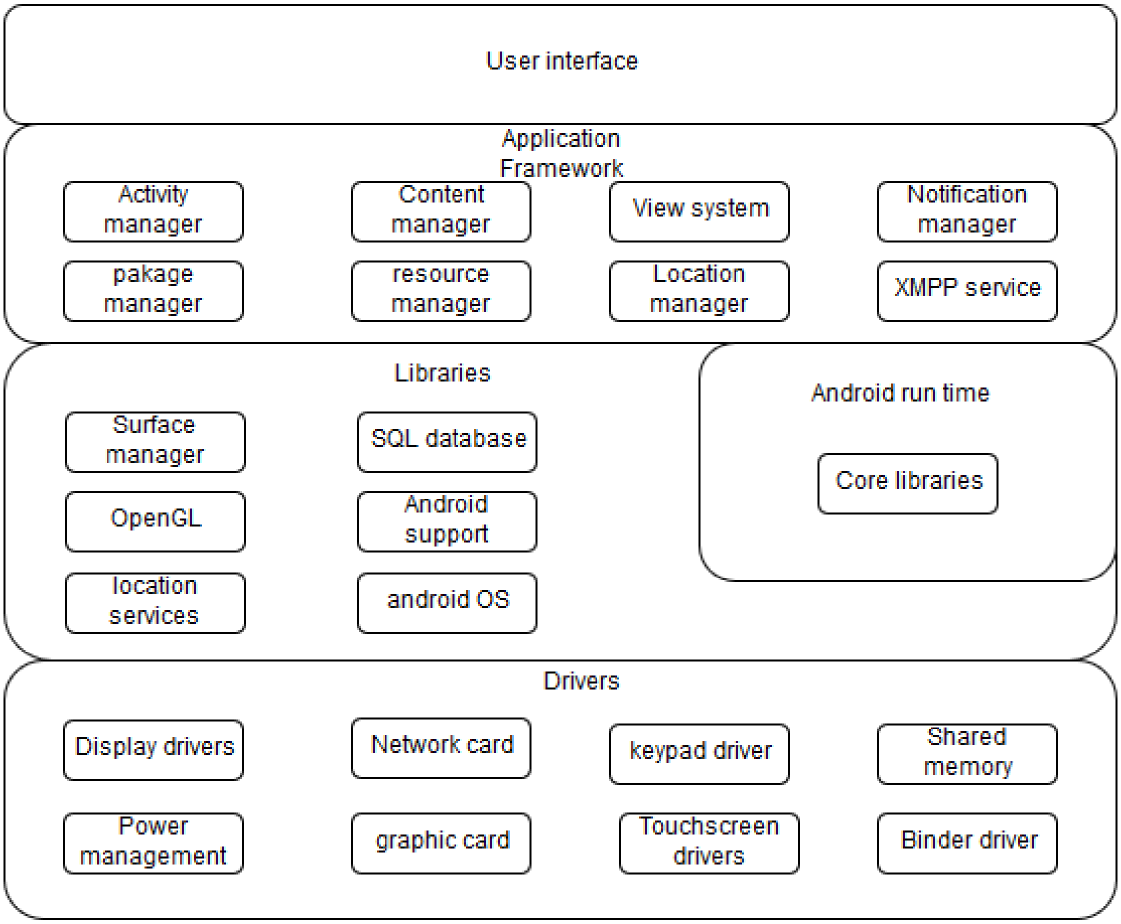

Figure 4: Architectural Diagram

5.1.3 Architecture Design

As we discussed earlier the flow of our application, the architecture which it depicts is client server architecture because our whole application after when recipient requests for the blood is completely client server and database centered. It involves a server, database server, Cloud messaging server, SMS gateway, and users.

User will interact with the app using GUI of android. After requesting for the blood, server will check the blood type and sends notification via Cloud messaging server and SMS gateway. Every eligible user will receive and respond to the request and the notification is again send back to server. And server again send this notification to user.

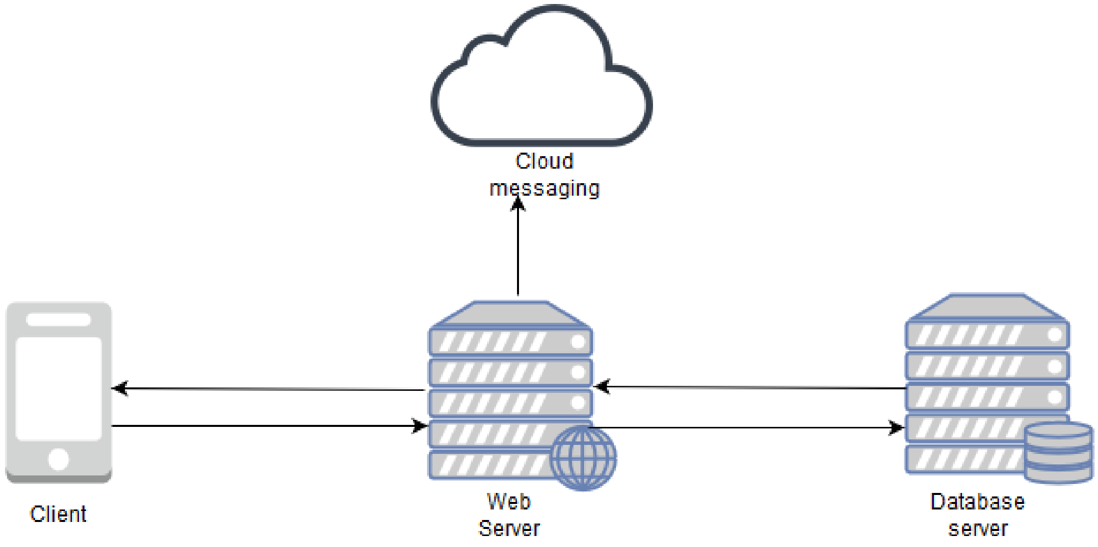

Figure 5 : System Architecture Approach

Whole system involves the following sub level systems in it:

- **Web server:** All the queries are directed to the web server that is in listening state for 24/7. Whenever a user make a request server listens to it and get data from the database and perform the whole functionality.
- **Database server:** Data about the donor and recipient including their blood type address and details like height weight is placed in database server. Webserver gets all its data for performing different activities from the database server.
- **Cloud messaging:** Cloud messaging is used to send messages to users.

5.1.4 Subsystem Architecture

The major function apart from user login and registration of our application is request generation and how the system cater it when recipient request for the blood. From the below mentioned sequence diagram we can understand in a clearer way that how system acts in request generation state.

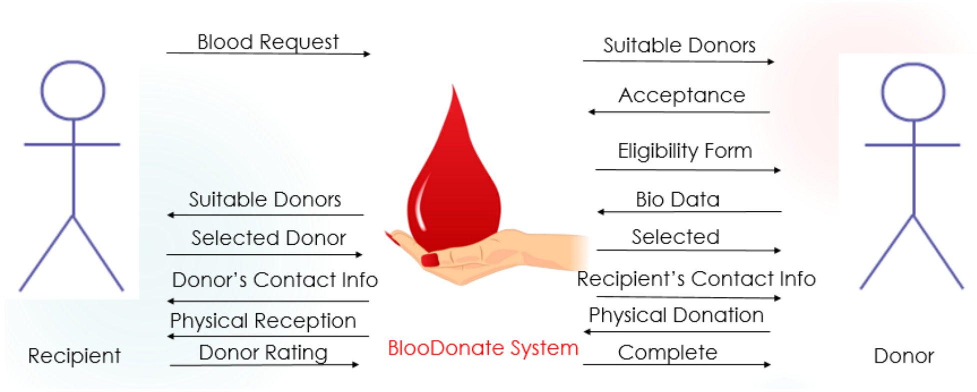

Figure 6 : Flow of Actions

The system architecture in terms of the sequence of actions and activities taking place is shown in the sequence diagrams in the next section

5.1.5 Sequence diagrams

5.1.5.1 Sign up

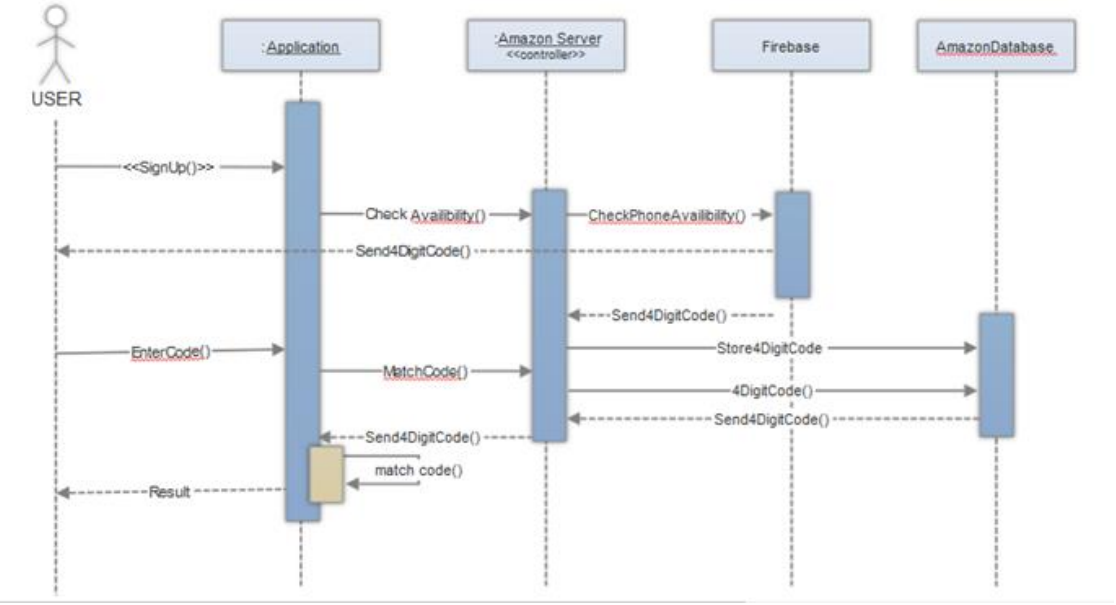

Figure 7 : Sign up Sequence Diagram

5.1.5.2 Sign in

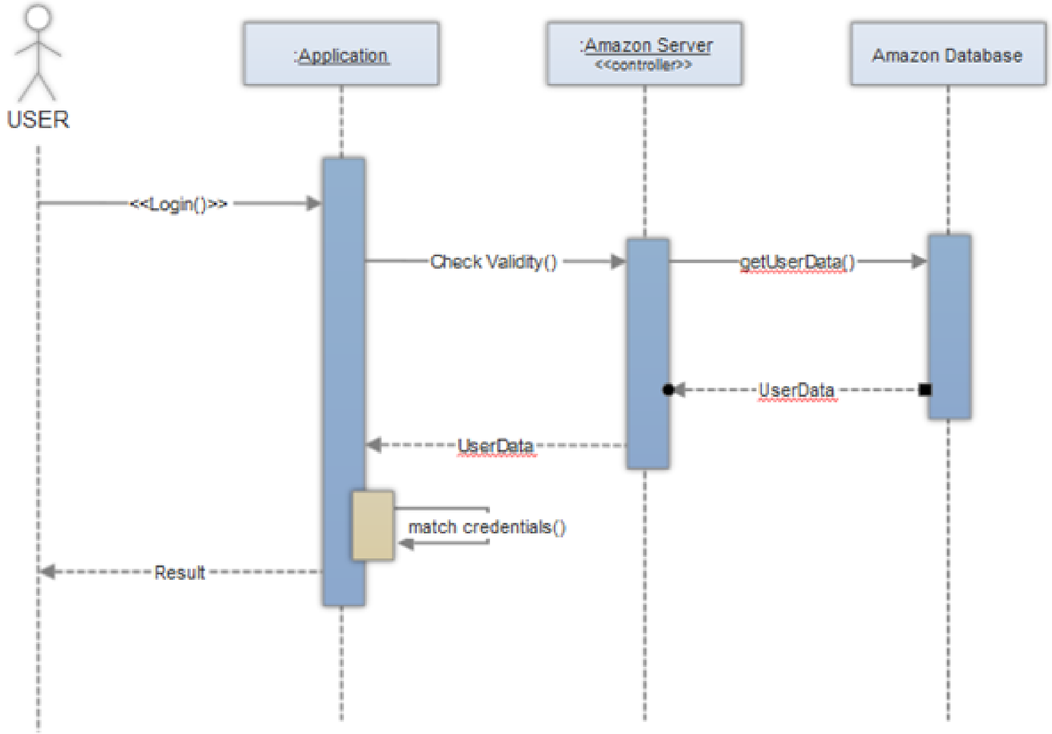

Figure 8 : Sign in Sequence Diagram

5.1.5.3 Request generation

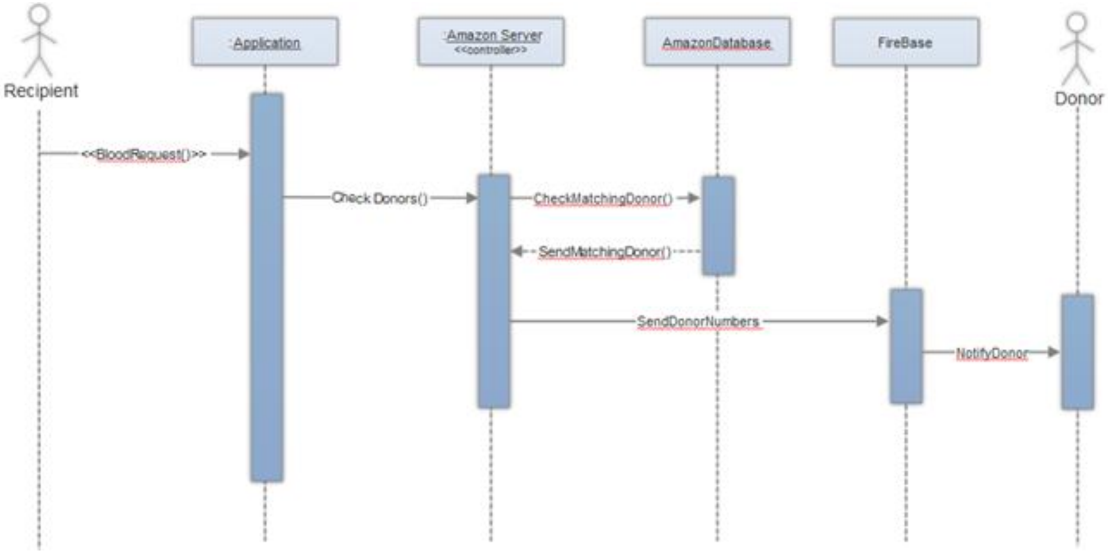

Figure 9 : Request Generation Sequence Diagram

5.1.5.4 Accept Request

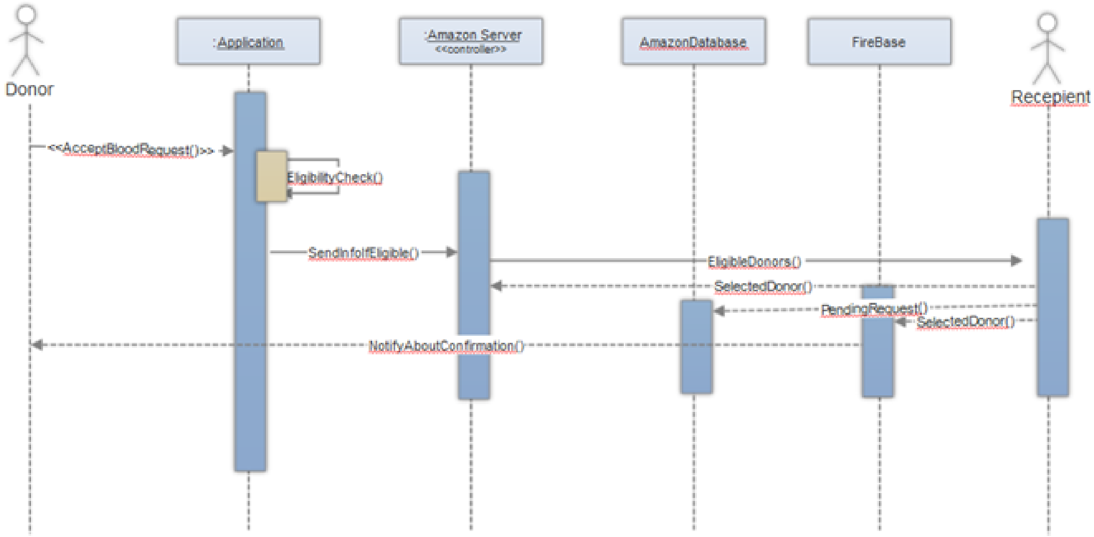

Figure 10 : Accept Request Sequence Diagram

5.2 DETAILED SYSTEM DESING

5.2.1 Amazon Web Server

Classification

Amazon web server has live EC2 machine. It receives all types of requests related to blood request and queries related to blood response.

Definition

The main purpose of server is it listens to the requests. It is live and listening to users 24/7. It has all the queries related to each functionality and direct them to database for data retrieval.

Responsibilities

Whenever someone is in need of blood, that person generate a request by entering the blood type he is in need of, the amount he requires and his position on google maps. This request is directed to the server and it performs the following actions:

1. Query database for the eligible donors.

2. Receives the query results and redirect it to cloud messaging server in order to send notifications to the eligible donor.
3. When a donor responds back to the request, it send notifications to the recipient.
4. When recipient receives the notification about the donor acceptance, it confirms the donor. This confirmation is again redirected to the server and server again send this confirmation to database and cloud messaging
5. Moreover, it also checks for the notification that are completed by the donor. Removes those completed requests.

Constraints

For the final year project, we used the free version of amazon web server for one year only. From onward we'll have to pay Moreover it only provide 1GB storage space. If we want to increase the storage space, we have to pay more.

Composition

This EC2 machine has apache server installed on it. It has PHP code for the application stored on it.

Uses/Interactions

EC2 has apache server installed on it. This machine in short server perform or complete all the user requests and listens to user response in case when donor response back. Moreover, it also communicates with the cloud messaging server via SMS Gateway.

Resources

Live machine needs the following resources:

1. Apache web server
2. A live database
3. Processors to process the events and requests.

Processing

Whenever the server receives an http request from a user, it finds the Php code file and run that code. That Php code has MySQL query in it which is then directed to database and fetch the results from it.

5.2.2 Amazon Database Server

Classification

Amazon database server is a live server which can be queried 24/7.

Definition

The main purpose of database server is it listens to the all MySQL queries. It is live and listening to users 24/7.

Responsibilities

Amazon database has the following responsibilities:

1. To store data in an efficient way so that it doesn't take much time while fetching the records.
2. It much makes the data storage secure.
3. It must fetch the results fast.

Constraints

For the final year project, we used the free version of amazon database server for one year only. From onward we'll have to pay. Moreover, it only provides 1GB storage space. If we want to increase the storage space, we have to pay more.

Composition

We are using MySQL database for querying Amazon database. This query is carried out by Php code.

Uses/Interactions

Amazon database uses various database search engine. We are using MySQL for our project.

Resources

Database component stores and backup the data in it.

Processing

Whenever the server receives an http request from a user, it finds the Php code file and run that code. That Php code has MySQL query in it which is then directed to database and fetch the results from it.

5.2.3 Firebase

Classification

Firebase is used for cloud messaging.

Definition

Firebase is being used by our system for cloud messaging. It is directly connected to SMS Gateway which is connected to the server. It is used for sending messages to the users.

Responsibilities

It has the following responsibilities

1. It lets the user reliably send the messages at no costs.
2. It notifies the user about any new notification in the app.

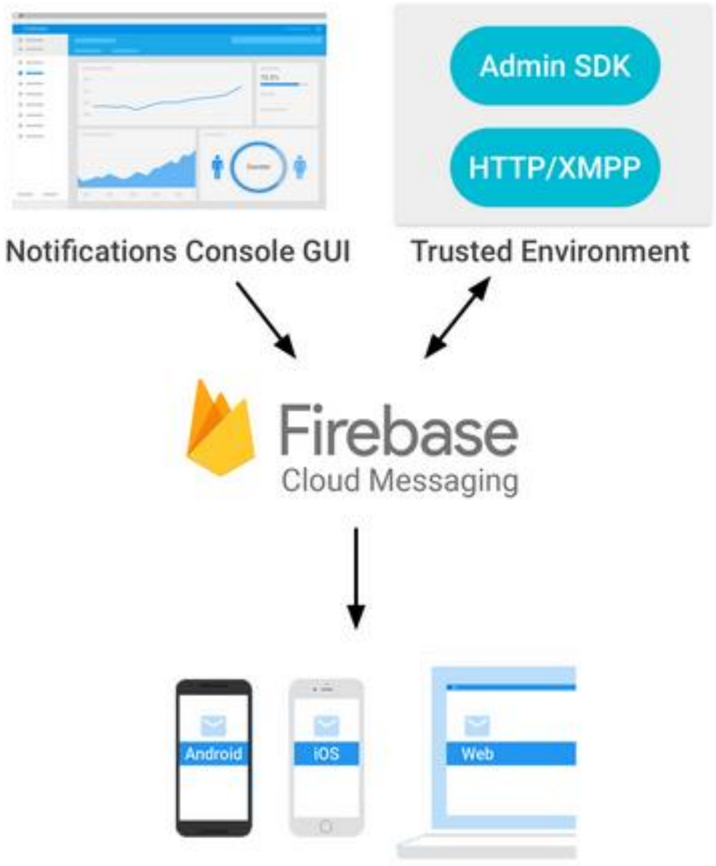

Figure 11 : Firebase Connections

Constraints

Firebase services are free of cost while the SMS gateway has been programmed locally by us so does not require finances for its continuous running. The only costs incurred are that for sending SMS messages to the other mobile phones.

Composition

Firebase messaging console is responsible for sending direct push notifications to our blood application as well as to the SMS gateway mobile application. SMS gateway mobile application receives these notifications and then forwards these in the form of SMS messages using the SIM in the mobile phone.

Resources

- An Android mobile phone
- A secure Internet connection
- Cloud computing power
- Firebase configured account
- A charged sim card

Processing

When the Php server sends request to the firebase server, it forwards this to the SMS gateway, which in turn sends the notifications to the user mobile phones.

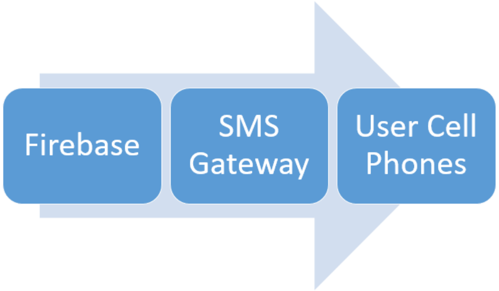

Figure 12 : Firebase Processing

5.3 CLASS DIAGRAM

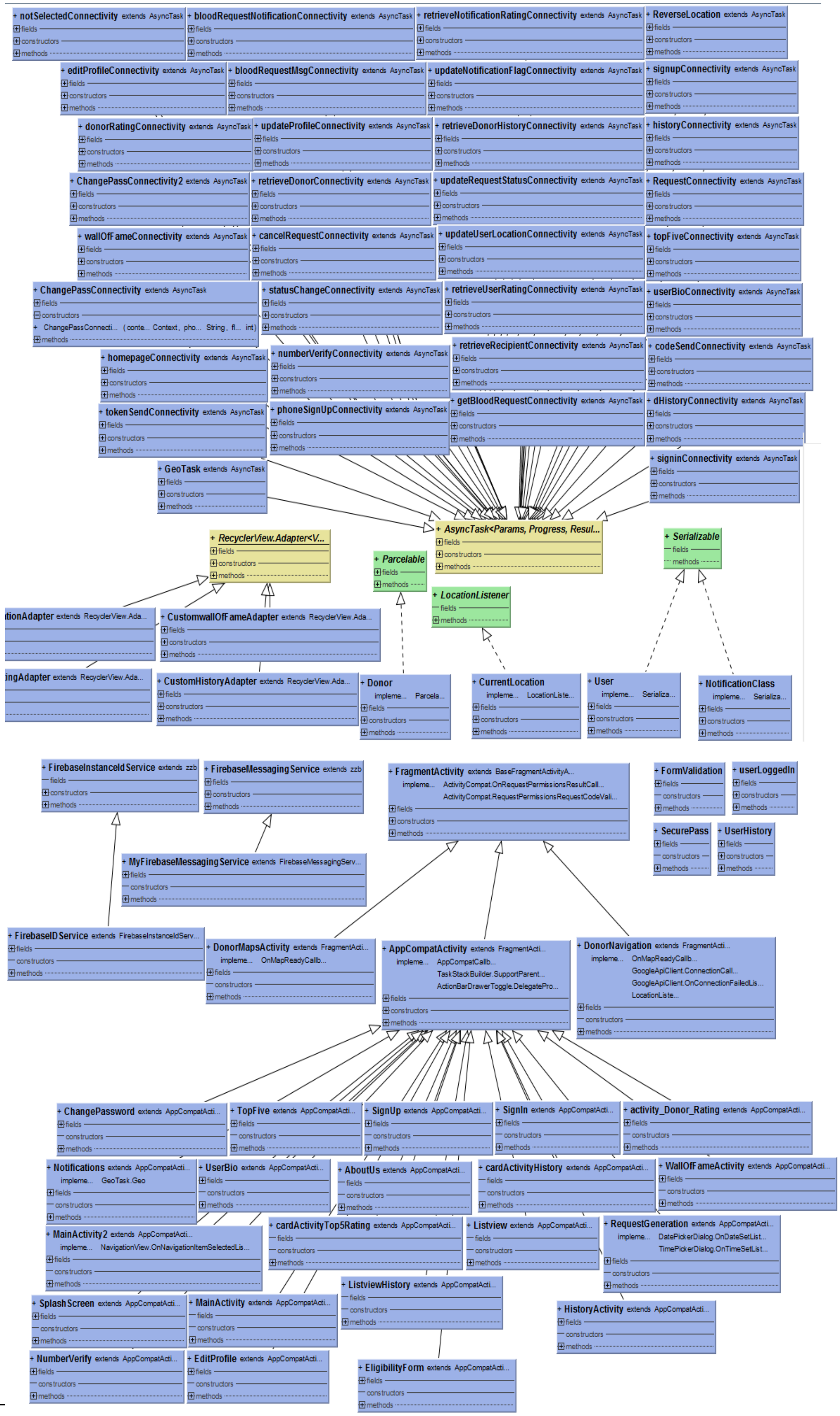
notSelectedConnectivity extends AsyncTask
bloodRequestNotificationConnectivity extends AsyncTask
retrieveNotificationRatingConnectivity extends AsyncTask
ReverseLocation extends AsyncTask
editProfileConnectivity extends AsyncTask
bloodRequestMsgConnectivity extends AsyncTask
updateNotificationFlagConnectivity extends AsyncTask
signupConnectivity extends AsyncTask
donorRatingConnectivity extends AsyncTask
updateProfileConnectivity extends AsyncTask
retrieveDonorHistoryConnectivity extends AsyncTask
historyConnectivity extends AsyncTask
ChangePassConnectivity2 extends AsyncTask
retrieveDonorConnectivity extends AsyncTask
updateRequestStatusConnectivity extends AsyncTask
RequestConnectivity extends AsyncTask
wallOfFameConnectivity extends AsyncTask
cancelRequestConnectivity extends AsyncTask
updateUserLocationConnectivity extends AsyncTask
topFiveConnectivity extends AsyncTask
ChangePassConnectivity extends AsyncTask
ChangePassConnecti... (conte... Context, pho... String, fl... int)
statusChangeConnectivity extends AsyncTask
retrieveUserRatingConnectivity extends AsyncTask
userBioConnectivity extends AsyncTask
homepageConnectivity extends AsyncTask
numberVerifyConnectivity extends AsyncTask
retrieveRecipientConnectivity extends AsyncTask
codeSendConnectivity extends AsyncTask
tokenSendConnectivity extends AsyncTask
phoneSignUpConnectivity extends AsyncTask
getBloodRequestConnectivity extends AsyncTask
dHistoryConnectivity extends AsyncTask
signinConnectivity extends AsyncTask
GeoTask extends AsyncTask
AsyncTask<Params, Progress, Resul...
RecyclerView.Adapter<V...
Parcelable
Serializable
LocationListener
itionAdapter extends RecyclerView.Ada...
CustomwallOfFameAdapter extends RecyclerView.Ada...
ingAdapter extends RecyclerView.Ada...
CustomHistoryAdapter extends RecyclerView.Ada...
Donor impleme... Parcela...
CurrentLocation impleme... LocationListe...
User impleme... Serializa...
NotificationClass impleme... Serializa...
FirebaseInstanceIdService extends zzb
FirebaseMessagingService extends zzb
FragmentActivity extends BaseFragmentActivityA... impleme... ActivityCompat.OnRequestPermissionsResultCall... ActivityCompat.RequestPermissionsRequestCodeVali...
FormValidation
userLoggedIn
SecurePass
UserHistory
MyFirebaseMessagingService extends FirebaseMessagingServ...
FirebaseIDService extends FirebaseInstanceIdServ...
DonorMapsActivity extends FragmentActi... impleme... OnMapReadyCallb...
AppCompatActivity extends FragmentActi... impleme... AppCompatCallb... TaskStackBuilder.SupportParent... ActionBarDrawerToggle.DelegatePro...
DonorNavigation extends FragmentActi... impleme... OnMapReadyCallb... GoogleApiClient.ConnectionCall... GoogleApiClient.OnConnectionFailedLis... LocationListe...
ChangePassword extends AppCompatActi...
TopFive extends AppCompatActi...
SignUp extends AppCompatActi...
SignIn extends AppCompatActi...
activity_Donor_Rating extends AppCompatActi...
Notifications extends AppCompatActi... impleme... GeoTask.Geo
UserBio extends AppCompatActi...
AboutUs extends AppCompatActi...
cardActivityHistory extends AppCompatActi...
WallOfFameActivity extends AppCompatActi...
MainActivity2 extends AppCompatActi... impleme... NavigationView.OnNavigationItemSelectedLis...
cardActivityTop5Rating extends AppCompatActi...
Listview extends AppCompatActi...
RequestGeneration extends AppCompatActi... impleme... DatePickerDialog.OnDateSetList... TimePickerDialog.OnTimeSetList...
ListviewHistory extends AppCompatActi...
SplashScreen extends AppCompatActi...
MainActivity extends AppCompatActi...
HistoryActivity extends AppCompatActi...
NumberVerify extends AppCompatActi...
EditProfile extends AppCompatActi...
EligibilityForm extends AppCompatActi...
fields
constructors
methods

Figure 13 : Class Diagram

5.4 ER DIAGRAM

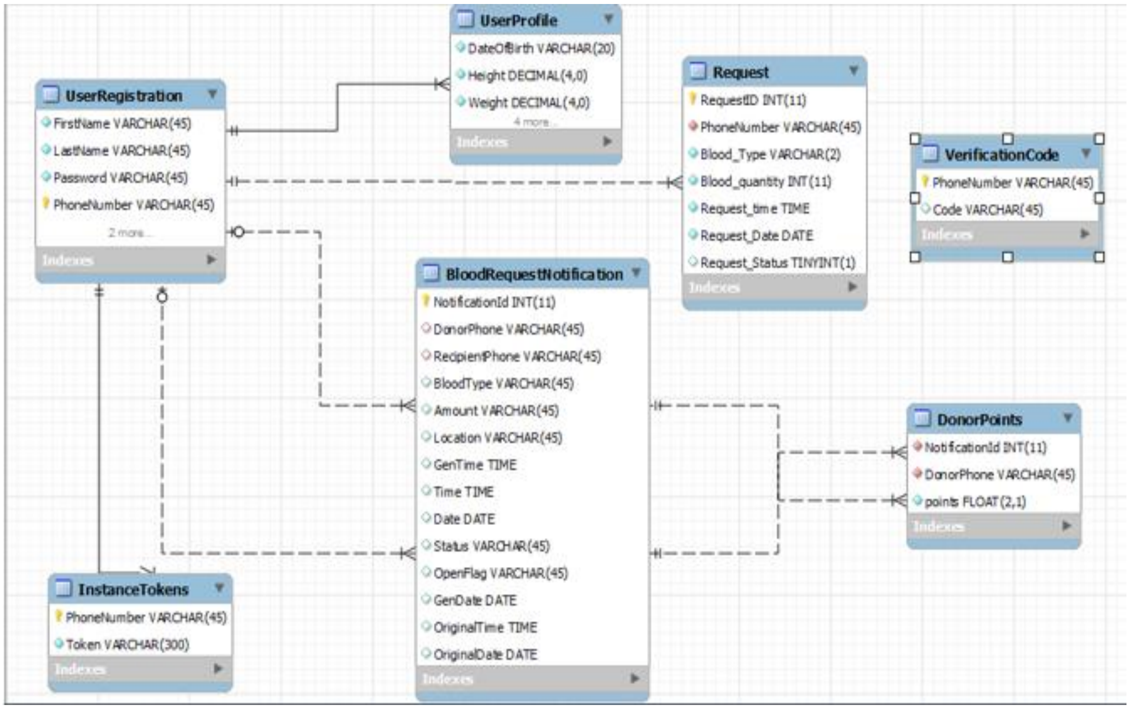

Figure 14 : Entity Relation Diagram

Chapter 6

IMPLEMENTATION AND TESTING

The system consists of an android application at the front end developed using Android Studio and google location services, this is connected at the application layer with an Apache Web Server running on an EC2 live machine configured using Amazon Web Services (AWS). This application server is in turn connected to a live SQL database server also configured using AWS. This database server has a database managed on it using MySQL workbench. The application server also integrates with Firebase cloud messaging services for the purpose of sending notifications to the mobile front end. Firebase is also connected to an SMS gateway, coded in the form of an android application. This SMS gateway receives notifications from the firebase Messaging services containing the content of the message and the number to which it is to be sent. It is the responsibility of this SMS gateway to send SMS notifications to the user mobile phones.

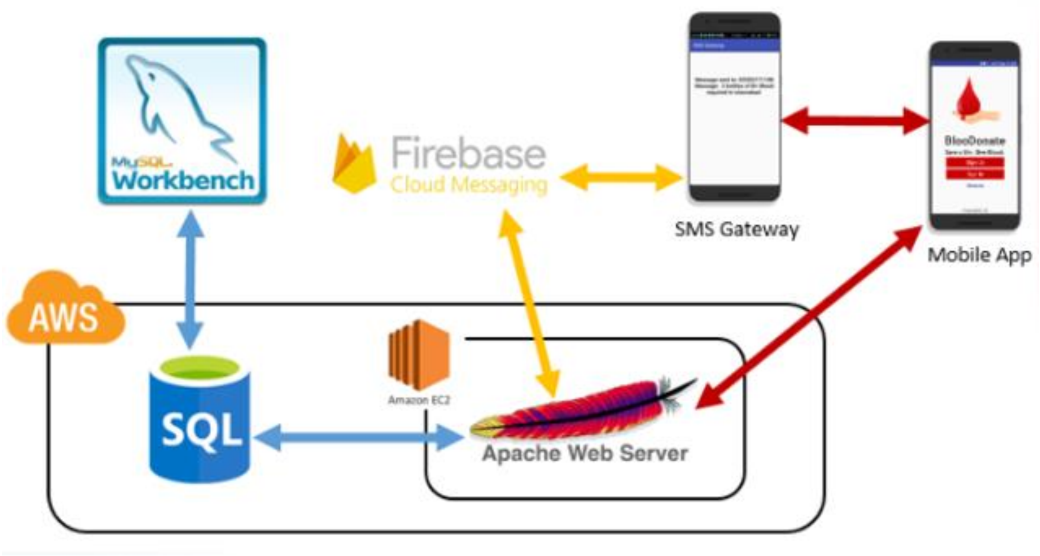

Figure 15 System Components

6.1 METHODS, TOOLS AND TECHNIQUES

6.1.1 For Front End:

- Junit Test Framework
- Android Studio
- Internet Connection
- GPS Connection
- An Android Mobile Phone

6.1.2 For Web Server

- Amazon Web Services
- Live Online Virtual Machine
- Live Web Server
- HTTP Connection
- PHP Setup
- Putty (Shell Connector)
- Firebase Services
- SMS Gateway

6.1.3 For Database

- Amazon Web Services
- Live virtual Machine
- Live Database Server
- MySQL Workbench
- Secure Internet Connection

6.1.4 Main Resources Required

- An Android Mobile Phone

- Secure Internet Connection
- GPS Connection
- MySQL Workbench
- Live Database server
- Amazon Web services

6.2 CORE FUNCTIONALITIES

- User sign up using a phone number which is unique among all users, this phone number is verified using SMS code verification method. The user also has to enter bio data such as weight, height etc. in order to register.
- Sing in which includes password hash matching so that the passwords are secure.
- Home page display which is the main activity hub displaying the user profile data.
- Request generation which allows the user to request blood on the basis of blood type, location, amount required and the time at which it is required.
- History which shows the status of all ongoing and previous blood donations.
- Notifications which show the status of newly received messages.
- Edit profile using which a user can alter the bio data.
- Change password which allows a user to store a new password by entering the previous one.
- Availability status change using which a user can tell the system whether he/she is available for donation or not.

6.3 TESTING

The system has been tested using unit, integration, system and usability testing. This section presents the testing process in detail and the next section evaluates the results obtained.

6.3.1 Unit Testing

All the modules and units in the system have been tested independently in order to verify that everything is working according to specification. Each unit, it's expected and actual outcome were verified using complete test cases which have been mentioned in detail in the tables included in integration test plan. The test cases used for unit and integration test cases were same, and after unit testing integration testing was done in iterations in order to make sure that the system is working as expected.

6.3.2 Scope of Testing

There are three systems in the project which are in the form of layers. Namely the front end (an android application), a web server and a database server. The modules in these systems, after being tested in isolation in unit testing will be integrated in incremental fashion in integration testing. These systems will further be tested using the bottom up approach. A high level architecture of the system is shown in the diagrams below:

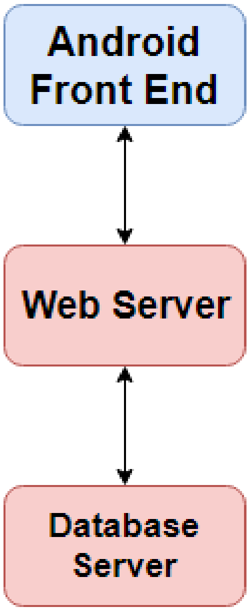

Figure 16 : Layered Architecture

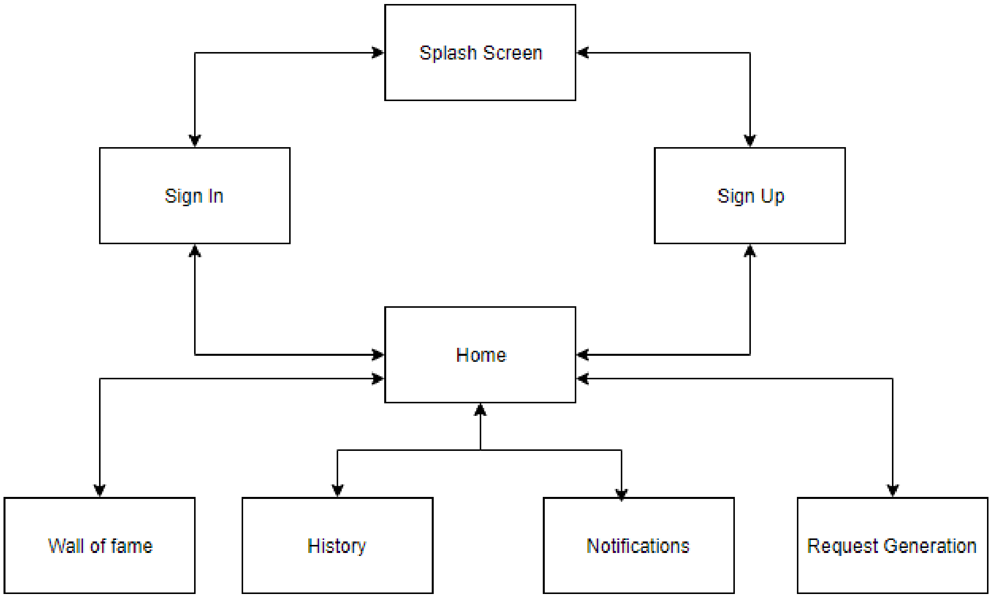

Figure 17 : Front End Sub Systems

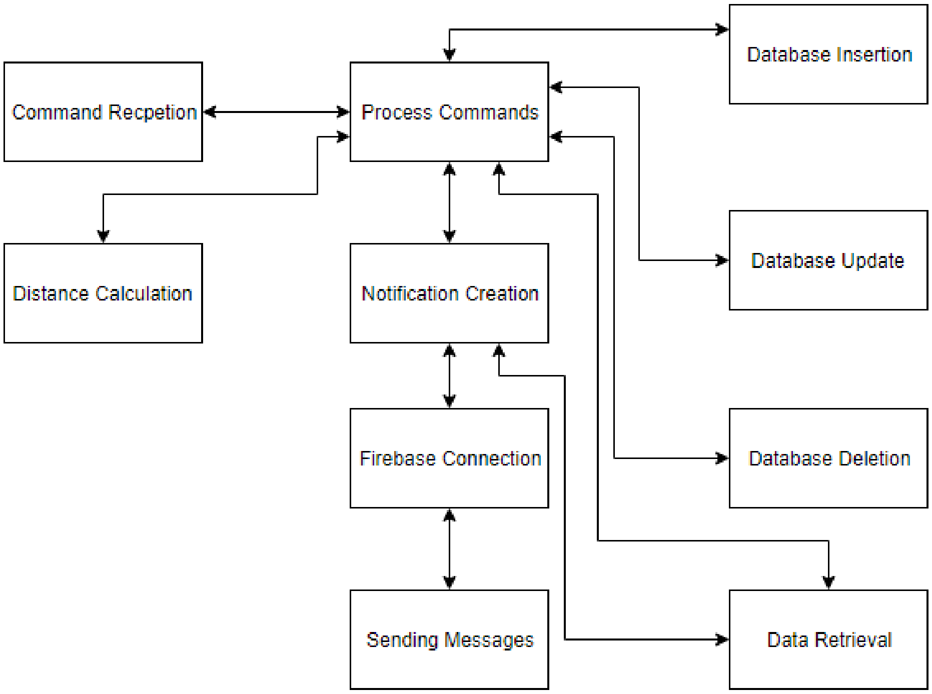

Figure 18 : Application Logic Sub Systems

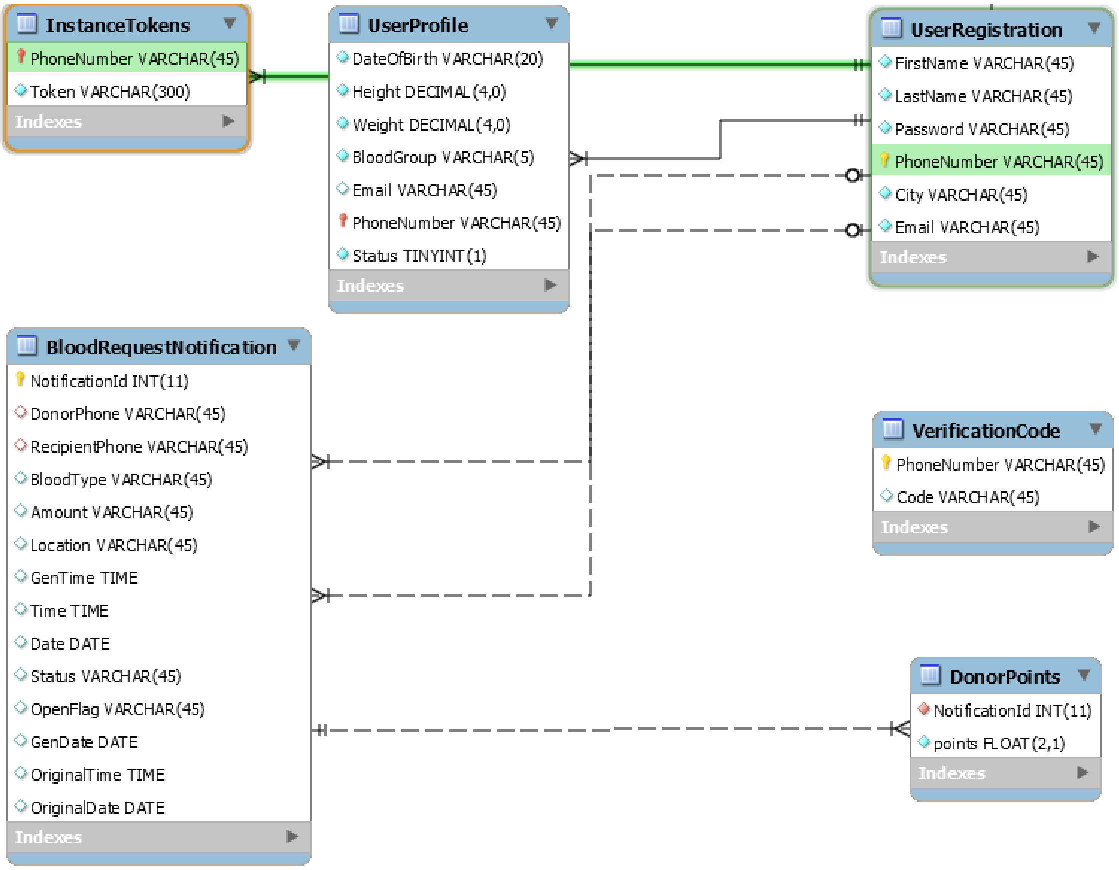

Figure 19 : Database Modules

6.3.3 Characteristics to Be Tested

- The interface is working properly and is not taking more than a specified amount of time to respond.
- The interface is taking user input.
- The input data is being processed correctly.
- The connection with web server is well established.
- The web-serve is performing functions on the database in the specified amount of time.
- The database is performing queries in the specified time.
- The operations being performed by the database are correct.

6.3.4 Integration Techniques

Integration Techniques being used are Incremental testing for Intra-system and Bottom up testing for Inter-system testing.

Justification: The selected project consists of modules which cannot be broken be considered in the form of a hierarchy, there are three systems in the project which are in the form of layers. These layers have modules which are interacting with each other but not in a hierarchical fashion, instead some represent the core functionality and some represent the additional services being provided. Hence, it will be more suitable to test these systems using an incremental approach, but testing the main modules first and then adding modules into the systems one by one.

Bottom up approach for inter system testing is being used as the stubs for lower layer will be very far away from the top layer and it would lead to increased time spent writing stubs for this. Also given the dynamic nature of this application, any defect in the lower layer can cause the whole system to crash, hence it is necessary to test the lower layer first in order to proceed further.

6.3.5 Structure of integration level

6.3.5.1 Integration testing phases

1. Functional testing will be carried out first for each module after it is integrated with the system in order to confirm the correct functioning of the system as it is integrated.
2. Interface testing will be performed as to test that each module integrated into the system is interfacing correctly with the other modules
3. End-to-end will be carried out to ensure the correct output from the systems when given a specific input.
4. Endurance testing will be performed at the end on the whole system in order to see the number of connections the system can bear for a long time.

Modules or subsystems to be integrated in each phase:

Front end: Android application

- Current Location
- Donor Navigation
- Reverse Location
- Application Security
- About us
- Donor Rating
- Data Parser
- History
- Edit Profile
- Eligibility Form
- Geo Task
- Home Page
- Number Verify
- Request Generation
- Sign In
- Sign Up
- Splash Screen
- User Bio
- User Logged In
- Wall of Fame
- Form Validation
- Firebase Messaging Service

Connectivity Modules:

- Request Message Connectivity
- Request Notification Connectivity
- Cancel Request Connectivity
- Change Password Connectivity
- Code Send Connectivity

- History Connectivity
- Donor Rating Connectivity
- Edit Profile Connectivity
- Get Blood Request Connectivity
- History Connectivity
- Home Page Connectivity
- Not Selected Connectivity
- Number Verify Connectivity
- Phone Sign Up Connectivity
- Retrieve Donor Connectivity
- Sign in Connectivity
- Sign Up Connectivity
- Status Change Connectivity
- Token Send Connectivity
- Update Notification Flag Connectivity
- Update Profile Connectivity
- Update Request Status Connectivity
- Update User Location Connectivity
- User Biodata Connectivity
- Wall of Fame Connectivity

Modules to be integrated in each phase (functional and interface verification will be carried out side by side):

Table 1 : Modules to be Integrated 1

ID	**Functional**	**Interface**
1	Splash Screen	User Logged In
2	Sign Up	Phone Sign Up Connectivity Code Send Connectivity
3	Number Verify	Number Verify Connectivity
4	User Bio	User Bio Connectivity

	Secure Pass Form Validation	
5	Sign In	Sign in Connectivity Token Send Connectivity
6	Home Page Current Location Reverse Location	Home Page Connectivity Update User Location Connectivity
7	Edit Profile	Update Profile Connectivity
8	Request Generation	Blood Request Message Connectivity Blood Request Notification Connectivity
9	Notifications	Get Blood Request Connectivity Cancel Request Connectivity Retrieve Donor Connectivity Update Notification Flag Connectivity Geo Task Data Parser
10	Eligibility Form	Update Blood Request Status Connectivity
11	History	History Connectivity Donor Rating Connectivity Not Selected Connectivity Retrieve Donor History Connectivity
12	Wall of Fame	Wall of fame Connectivity

Web server: the application logic

Sub-systems for web-server are:

- Cancel Request
- History User
- Distance calculation
- Donor Rating
- Edit profile Get Data
- Match User

- Mobile Sign Up
- Notification Cleanup
- Not selected
- Receive Instance token
- Retrieve blood request
- Retrieve donors
- Retrieve Registration
- Retrieve Token
- Send Firebase Message
- Send custom notification
- Send notification to donors
- Send text
- Send Text to donors
- Send verification Text
- Sign In
- Sign Up
- Store Bio Data
- Store Blood Request Notification
- Store Code
- Store Instance Token
- Update Blood Request Status
- Update Notification Flag
- Update Profile
- Update Status
- Update User Location
- Verify code
- Wall of fame points

Modules to be integrated each phase:

Table 2 : Modules to be Integrated 2

ID	Functional	Interface
1	Sign Up Verify code	Store Bio Data Mobile Sign Up Store code
2	Send Firebase Message	Send Instance Token Send Custom Notification Send Notification to Donors
3	Send Text	Send Verification Text Sent Text to Donors
4	Sign In	Receive Instance Token Update User Location Retrieve Registration
5	Send Blood Request	Match User Retrieve Token
6	Store Blood Request Notification Retrieve Blood Request	Update Status Update Blood Request Status Update Notification Flag Distance Calculation Cancel Request
7	Notification Clean Up	Delete Notification
8	Edit Profile	Update Profile
9	User History	Retrieve Donor Cancel Request Not Selected
10	Wall of Fame Points	Donor Rating

Database server: the data storage for the whole system

Modules in database server are:

- Blood Request Notification
- Donor Points

- Instance Tokens
- User Profile
- User Registration
- Verification Code

Modules to be integrated each phase:

Table 3 : Modules to be Integrated 3

ID	Functional	Interface
1	User Registration	Phone Insertion Procedure Phone Deletion Procedure
2	User Profile	User Insertion Procedure User Deletion Procedure User Update Procedure User Retrieval Procedure
3	Verification Code	Code Insertion Procedure Code Retrieval Procedure Code Update Procedure
4	Instance Token	Token Insertion Procedure Token Update Procedure Token Retrieval Procedure
5	Blood Request Notification	Request Insertion Procedure Request Retrieval Procedure Request Update Procedure
6	Donor Points	Points Insertion Procedure Points Retrieval Procedure

6.3.6 End to end Testing

End to end testing of sub-systems (it also involves integrating the three systems together):

Table 4 : Sub Systems

ID	**Sub-systems**
1	Mobile Sign Up
2	Sign Up
3	Sign In
4	Home
5	Blood Request
6	Notifications
7	User History
8	Wall of Fame
9	Log Out
10	Change Availability

For ***endurance testing*** the number of requests in a particular time frame will be generated in order to determine the number of concurrent connections the system can endure.

6.3.6.2 Environment to be setup and resources required in each phase

For front end functional and integration testing the following resources are required:

- Junit Test Framework
- Android Studio
- Internet Connection
- GPS Connection
- An Android Mobile Phone

For web server functional and integration testing the following resources are required:

- Amazon Web Services
- Live Online Virtual Machine
- Live Web Server

- HTTP Connection
- PHP Setup
- Putty (Shell Connector)
- Firebase Services
- SMS Gateway

For Database functional and integration testing the following resources are required:

- Amazon Web Services
- Live virtual Machine
- Live Database Server
- MySQL Workbench
- Secure Internet Connection

For End-to-End and Endurance Testing:

- An Android Mobile Phone
- Secure Internet Connection
- GPS Connection
- MySQL Workbench
- Live Database server
- Amazon Web services

6.3.6.3 Criteria for each integration test phase

For Functional Testing

Entry criteria:

- Environment is setup
- Resources are available
- Code for every module is working
- All unit tests have been passed
- Bugs from previous phase have been fixed and verified

Exit criteria:

- All test cases pass

Test configuration set-up:

- Using Junit framework
- SQL Procedures

For Interface Testing

Entry criteria:

- Interfaces are developed
- Resources are available
- Code for every module is working
- All unit tests have been passed
- Bugs from previous phase have been fixed and verified

Exit criteria:

- Information is being communicated in the right format

Integration Techniques to be used:

- Incremental

Test configuration set-up:

- Using Junit framework
- SQL Procedures

For end-to-end testing:

Entry criteria:

- All systems have been tested separately
- Integrated interfaces of the systems
- Bugs from previous phase have been fixed and verified
- Application install ability has been checked

Exit criteria:

- End result at the other end is expected given the input

Integration Techniques to be used:

- Bottom Up

Test configuration set-up:

- Using Junit framework
- SQL Procedures

For endurance testing:

Entry criteria:

- Environment is setup
- All the systems have been integrated and pass end-to end testing
- Bugs from previous phase have been fixed and verified

Exit criteria:

- The application can ensure a specific number of connected users in a specified time frame

Integration Techniques to be used:

- Bottom Up

Test configuration set-up:

- Manual
- Junit Testing Framework

Test cases for front end

Table 5 : Test Cases for Front End

ID	Input Data	Initial Condition	Expected Result	Test Procedure
1	User click on application icon	Application installed	App opens and splash screen is displayed for 1.5 seconds	Observe time of splash screen display
2	Call Menu Display	Splash screen exits	Screen is displayed showing buttons for Sign Up and Sing In	Manual checking
3	User clicks on Sign Up button	Mobile device is connected to the network and Menu is displayed	Mobile entry field is displayed	Manual checking
4	Mobile phone number entry in correct format	Mobile entry filed is displayed	System accepts the number format	Observation that the system is accepting the number in correct format
5	Mobile phone number entry in incorrect format	Mobile entry field is displayed	System rejects the number format	Observation that the system is rejecting the number in incorrect format
6	Mobile phone number entry which already exists	System accepted the number format	System returns saying that use another number	Observation that the system is checking if the system returns

				number already exists flag
7	Mobile phone number entry which is new	System accepted the number format	System accepts the number	Observation that the system is checking if the system returns number accepted flag
8	Number accepted flag	System accepts the number	Send verification code module is called and code entry screen is displayed	Observation that the verification code module is called and verification code entry screen is displayed
9	User enters the correct code	Verification code entry screen is displayed	User bio data entry screen is displayed	Observe that the bio data screen is displayed
10	User enters the incorrect code	Verification code entry screen is displayed	User is told that the code is incorrect	Observe that the module returns incorrect code flag
11	User enters correct format bio data	User bio data entry screen is displayed	User is taken to the Home screen	Observe that the module calls the home screen
12	User enters incorrect format bio data	User bio data entry screen is displayed	User is told to correct the format of the entries	Observe that the module returns incorrect format bio data flag

13	User enters incomplete bio data and presses the submit button	User bio data entry screen is displayed	User is told to complete the form	Observe that the module returns incomplete data flag
14	User presses the Sign In button	Mobile device is connected to the network and Menu is displayed	Sign In in form is displayed	Observe that the module calls sing in form
15	User enters incorrect phone number and incorrect password	Sign In in form is displayed	Sign In unsuccessful message is displayed	Observe that the module returns sign in unsuccessful flag
16	User enters correct phone number and incorrect password	Sign In in form is displayed	Sign In unsuccessful message is displayed	Observe that the module returns sign in unsuccessful flag
17	User enters incorrect phone number and correct password	Sign In in form is displayed	Sign In unsuccessful message is displayed	Observe that the module returns sign in unsuccessful flag
18	User enters correct phone number and correct password	Sign In in form is displayed	Sign In successful message is displayed	Observe that the module returns sign in successful flag

19	Home screen is called	Sign In successful message is displayed	Home screen shows correct bio data information of the user logged in	Observe that the module displays correct user bio data information
17	Edit profile is clicked	Home screen is displayed	Edit profile form is displayed	Observe that the module calls the edit profile form
18	Edit profile form is called	Edit profile has been clicked	Form is auto filled with correct existing information	Observe that the module displays the correct information
19	User edits the information in incorrect format	Edit profile displays correct existing data	System rejects the information	Observe that the system returns rejected information flag
20	User edits the information in correct format	Edit profile displays correct existing data	System accepts the information	Observe that the system returns accepted information flag
21	User clicks the submit edit profile button while the information is incomplete	Edit profile displays correct existing data	System returns incomplete information flag	Observe that the system returns incomplete information tag
22	User clicks the submit edit profile button while the information is complete	Edit profile displays correct existing data	System goes back to the home screen	Observe that the system calls the home screen

23	User clicks the request button	Home screen is displayed	Request Screen is displayed	Observe that the system calls the request screen
24	User selects the blood amount, blood type and location	Request screen is displayed	System returns the number of donor to which request is sent	Observe that the system returns the correct number of donors
25	User clicks the notification screen	Home screen is displayed	System displays the notifications for the user including any requests accepted by the donors, requests received by the user, if a recipient has selected the current donor and if a current donation has been cancelled or completed	Observe that the system returns the correct notifications for the relevant user
26	User clicks the blood required notification	Notification screen is being displayed	System displays the eligibility form	Observe that the system calls the eligibility form
27	Donor fills the eligibility form and meets the criteria	Eligibility form is displayed	System accepts the donor	Observe that the system returns the donor accepted flag
28	Donor fills the eligibility form and does not meet the criteria	Eligibility form is displayed	System rejects the donor	Observe that the system returns the donor rejected flag
29	Recipient clicks the donor accepted notification	Notification screen is displayed	System displays the map showing all the donors accepted	Observe that the module displays all the donors accepted

30	Recipient clicks on a donor to select him/her for donation	Map having accepted donors is being displayed	Systems marks the donor as accepted	Observe that the module returns the donor accepted flag
31	Donor clicks on the selected for donation notification	Notification screen is displayed	System displays the recipient contact information along with the location on a map	Observe that the module returns the correct recipient information
32	User clicks on the history icon	Home screen is displayed	History screen is displayed showing the previous completed donations, any pending blood requests by the recipient, any pending selection requests by the donor and any donation in process	Observe that the module returns the correct history information for the relevant user
33	User clicks on an request generated	History screen is displayed	Dialog for cancel request is displayed	Observe that the module calls the dialog for request cancellation
34	User clicks on cancel request	Request cancellation dialog is displayed	The request is marked as cancelled	Observe that the module returns the cancelled request flag
35	User clicks on do not cancel	Request cancellation dialog is displayed	The request status is unchanged	Observe that the request status remains unchanged
36	User clicks on an ongoing donation	History screen is displayed	Complete request dialog is displayed	Observe that the module calls the dialog for donation completion

37	User clicks on complete request	Complete request dialog is displayed	Donor rating screen is displayed	Observe that the donor rating screen is displayed
38	User gives rating to donor	Donor rating screen is displayed	Rating for donor is saved and History screen is displayed	Observe that the module returns rating saved flag and displays the history screen
39	User clicks on the Wall of fame icon	Home screen is displayed	Wall of fame is displayed showing top 10 donors having highest rating	Observe that the module returns the top 10 donors
40	User clicks on the availability icon	Home screen is displayed and availability icon is true	User availability is set to false	Observe that the module returns the availability flag as false
41	User clicks on the availability icon	Home screen is displayed and availability icon is false	User availability is set to true	Observe that the module returns the availability flag as true
42	User clicks on the logout icon	Home screen is displayed and user is logged in	User is logged out	Observe that the module returns the logged out tag

Test cases for web server

Table 6 : Test Cases for Web Servers

ID	Input Data	Initial Condition	Expected Result	Test Procedure
1	Mobile number	Mobile number already exist in database	Return rejected flag	Observe if a rejected is returned

2	Mobile number	Mobile number does not already exist in the database	Return accepted flag	Observe if an accepted flag is returned
3	Call to get code	Mobile number has been checked	A random code is returned	Observe if a random code is being returned
4	Send code through text	Random code has been generated	Return successful sent flag	Observe if a successful sent flag is returned
5	A correct code and phone number	Verification code has been sent and stored in database	Return a verified code flag	Observe if a verified flag is returned
6	An incorrect code and phone number	Verification code has been sent and stored in database	Return a not verified code flag	Observe if a not verified flag is returned
7	User profile data	User code has been verified	Return successful entry flag	Observe if the successful entry flag was returned
8	Request for blood group, amount and location	A blood request has been generated by the user	Returns the matching donors in 20 Km radius area	Observe if the module returns correct donors
9	2 points having a latitude and longitude value each	User has asked for donor distance information	Distance between the points is returned	Observe if the correct distance is returned
10	Notify donors	Matching donors have been found	Returns successful notification flag	Observe if a successful notification flag is returned
11	Store blood request	Suitable donors have been notified	Return successful entry flag	Observe if a successful entry flag is returned
12	Return notifications	User has notifications	Return the correct notification data	Observe if the data returned is correct according the relevant user
13	Return notifications	User does not have any notifications	Return no notification tag	Observe if a no notification tag is returned

14	Get history	User has history item	Return correct history	Observe if correct history has been returned according to the relevant user
15	Get rating	User has some donations	Returns the average of all donation made	Observe if correct average is returned
16	Get top ten donors	Connection established with database	Returns the top then donors in the database	Observe if the correct donors are returned
17	Change notification status	A notification already exists in the database	Returns success status changed flag	Observe if success states changed flag is returned
18	Send firebase notification	Connection with firebase is established	Returns successful notification sent flag	Observe if successful notification sent flag is returned

Test cases for database

Table 7 : Test Cases for Database

ID	**Input Data**	**Initial Condition**	**Expected Result**	**Test Procedure**
1	Phone number and insert flag	User number table exists	Phone number inserted	Observation for checking successful entry
2	Phone number and deletion flag	User table exists	User row with the corresponding phone number is deleted	Observation for checking successful deletion
3	User information and insertion flag	User table exists	User data inserted	Observation for checking successful entry
4	User information and update flag	User table exists	User data updated	Observation for checking successful update
5	User information and deletion flag	User table exists	User data deleted	Observation for checking successful deletion

6	Phone number and retrieval flag	User table exists	User data retrieved	Observation for checking successful retrieval
7	Instance Token and insertion flag	Instance token table exists	Instance token inserted	Observation for checking successful entry
8	Instance Token and update flag	Instance token table exists	Instance token update	Observation for checking successful update
9	Phone number and retrieval flag	Instance token table exists	Instance token retrieved	Observation for checking successful retrieval
10	Blood Request Notification and insertion flag	User and Request table exists	Request information stored	Observation for checking successful entry
11	Blood Request Notification and update flag	User and Request table exists	Request information updated	Observation for checking successful update
12	Phone number and retrieval flag	User and Request table exists	Request information retrieved	Observation for checking successful retrieval
13	Notification id, donor points and insertion flag	User, request and donor points table exits	Donor points stored	Observation for checking successful entry
14	Notification id, donor points and retrieval flag	User, request and donor points table exits	Donor points retrieved	Observation for checking successful retrieval
15	Phone number, verification code and insertion flag	Verification code table exists	Verification code stored	Observation for checking successful entry
16	Phone number, verification code and update flag	Verification code table exists	Verification code updated	Observation for checking successful update
17	Phone number and retrieval flag	Verification code table exists	Verification code retrieved	Observation for checking successful retrieval

6.3.7 System Testing:

After testing each and every unit and then performing integration testing unit all the actual results were in accordance with the expected results, testing of the whole system was performed. The main results are mentioned below:

Table 8 : System Testing

ID	Modules	Successful Operations
1	Splash Screen	User Logged In
2	Sign Up	Phone Sign Code Send
3	Number Verify	Number Verification
4	User Bio Secure Pass Form Validation	User Bio Data Storage and retrieval
5	Sign In	Sign In Token Communication
6	Home Page Current Location Reverse Location	Home Page Update User Location
7	Edit Profile	Update Profile
8	Request Generation	Blood Request Message Blood Request Notification
9	Notifications	Get Blood Request Cancel Request Retrieve Donor Update Notification Flag Geo Task Data Parser
10	Eligibility Form	Update Blood Request Status

11	History	History Donor Rating Not Selected Retrieve Donor History
12	Wall of Fame	Wall of fame display of top 5 donors

6.3.8 Usability Testing

Total of users involved: 15

Type of Tasks: Scenarios, open-ended

Brief Description: 3 groups of people were gathered and were given hypothetical scenarios to execute. The tasks were open ended and we monitored the task performance, errors encountered and the successful completions.

Major scenarios are mentioned below:

Scenarios:

1. Request generation
2. Request form filling by donor
3. Donor selection by recipient
4. Donor notified to go for donation
5. Recipient completes the donation giving points to donor
6. Donor cancels the request after being selected
7. Recipient cancels the request before any donor is accepted
8. Recipient cancels the request after acceptance but before selection of donor
9. Recipient cancels the request after selection of donor
10. Wall of fame shows correct donors
11. Update user profile
12. Change password
13. Sign out
14. History Display
15. Notifications Display

Chapter 7

RESULTS AND DISCUSSOIN

This section will show the results of executing the test cases mentioned in the testing section in form of tables and graphs. And also the results of usability testing based on different scenarios. Appendix-A in chapter 10 also shows the use cases of our system and it also shows the sequence of activities performed to achieve the functionality through various activity diagrams.

7.1 RESULTS FROM TEST CASES

7.1.1 Results from Front End

ID	Input Data	Actual Result	Expected Result	Pass/ Fail
1	User click on application icon	App opens and splash screen is displayed for 1.5 seconds	App opens and splash screen is displayed for 1.5 seconds	Pass
2	Call Menu Display	Screen is displayed showing buttons for Sign Up and Sing In	Screen is displayed showing buttons for Sign Up and Sing In	Pass
3	User clicks on Sign Up button	Mobile entry field is displayed	Mobile entry field is displayed	Pass
4	“0320-3171108”	Accepted	Accepted	Pass
5	“03203171108”	Rejected	Rejected	Pass
6	“0315-5899825”	Use Another other Number	Use Another other Number	Pass
7	“0333-5168056”	Number Accepted	Number Accepted	Pass
8	Number Accepted	Verification code sent	Verification code sent	Pass
9	9136	User bio data entry screen is displayed	User bio data entry screen is displayed	Pass
10	9236	Incorrect code	Incorrect code	Pass
11	User enters correct format bio data	User is taken to the Home screen	User is taken to the Home screen	Pass

Table 9 : Front End Test Results

12	User enters incorrect format bio data	Incorrect format	Incorrect format	Pass
13	User enters incomplete bio data and presses the submit button	Incomplete form	Incomplete form	Pass
14	User presses the Sign In button	Sign In in form is displayed	Sign In in form is displayed	Pass
15	Username: “0320-5168056” Password: “abc”	Sign In unsuccessful	Sign In unsuccessful	Pass
16	Username: “0320-3171108” Password: “abc”	Sign In unsuccessful	Sign In unsuccessful	Pass
17	Username: “0320-5168056” Password: “Peanut12*”	Sign In unsuccessful	Sign In unsuccessful	Pass
18	Username: “0320-3171108” Password: “Peanut12*”	Sign In successful	Sign In successful	Pass
19	Home screen is called	Home screen shows correct bio data information of the user logged in	Home screen shows correct bio data information of the user logged in	Pass
17	Edit profile is clicked	Edit profile form is displayed	Edit profile form is displayed	Pass
18	Edit profile form is called	Form is auto filled with correct existing information	Form is auto filled with correct existing information	Pass

19	User edits the information in incorrect format	Rejected	Rejected	Pass
20	User edits the information in correct format	Accepted	Accepted	Pass
21	User clicks the submit edit profile button while the information is incomplete	Incomplete information	Incomplete information	Pass
22	User clicks the submit edit profile button while the information is complete	System goes back to the home screen	System goes back to the home screen	Pass
23	User clicks the request button	Request Screen is displayed	Request Screen is displayed	Pass
24	User selects the blood amount: 2, blood type: AB- and location: Islamabad	Your request has been sent to 2 nearby donors	Your request has been sent to 2 nearby donors	Pass
25	User clicks the notification screen	System displays the notifications for the user including any requests accepted by the donors, requests received by the user, if a recipient has selected the current donor and if a current donation has been cancelled or completed	System displays the notifications for the user including any requests accepted by the donors, requests received by the user, if a recipient has selected the current donor and if a current donation has been	Pass

			cancelled or completed	
26	User clicks the blood required notification	System displays the eligibility form	System displays the eligibility form	Pass
27	Donor fills the eligibility form and meets the criteria	Donor Accepted	Donor Accepted	Pass
28	Donor fills the eligibility form and does not meet the criteria	Donor Rejected	Donor Rejected	Pass
29	Recipient clicks the donor accepted notification	System displays the map showing all the donors accepted	System displays the map showing all the donors accepted	Pass
30	Recipient clicks on a donor to select him/her for donation	Donor Selected	Donor Selected	Pass
31	Donor clicks on the selected for donation notification	System displays the recipient contact information along with the location on a map	System displays the recipient contact information along with the location on a map	Pass
32	User clicks on the history icon	History screen is displayed showing the previous completed donations, any pending blood requests by the recipient, any pending selection requests by the	History screen is displayed showing the previous completed donations, any pending blood requests by the recipient, any pending selection requests by	Pass

		donor and any donation in process	the donor and any donation in process	
33	User clicks on an request generated	Dialog for cancel request is displayed	Dialog for cancel request is displayed	Pass
34	User clicks on cancel request	Request Cancelled	Request Cancelled	Pass
35	User clicks on do not cancel	The request status is unchanged	The request status is unchanged	Pass
36	User clicks on an ongoing donation	Complete request dialog is displayed	Complete request dialog is displayed	Pass
37	User clicks on complete request	Donor rating screen is displayed	Donor rating screen is displayed	Pass
38	User gives rating to donor: 4.0	Rating for donor is saved and History screen is displayed	Rating for donor is saved and History screen is displayed	Pass
39	User clicks on the Wall of fame icon	Wall of fame is displayed showing top 10 donors having highest rating	Wall of fame is displayed showing top 10 donors having highest rating	Pass
40	User clicks on the availability icon	User availability is set to false	User availability is set to false	Pass
41	User clicks on the availability icon	User availability is set to true	User availability is set to true	Pass
42	User clicks on the logout icon	User is logged out	User is logged out	Pass

7.1.2 Test Cases for Web Server:

Table 10 : Web Server Test Results

ID	Input Data	Actual Result	Expected Result	Pass/Fail
1	“0320-3171108”	Already Exists	Already Exists	Pass
2	“0333-5168056”	Accepted	Accepted	Pass

3	Call to get code	9236	9236	Pass
4	Send code through text	Successfully Sent	Successfully Sent	Pass
5	Code: 9236, Phone: 0333-5168056	Verified	Verified	Pass
6	Code: 0236, Phone: 0333-5168056	Not Verified	Not Verified	Pass
7	Enter user profile data	Successful Entry	Successful Entry	Pass
8	Request for blood group: AB-, amount: 2 and location: Islamabad	Returns the matching 2 donors in 20 Km radius area	Returns the matching 2 donors in 20 Km radius area	Pass
9	32, 73 30, 73	222 Km	222 Km	Pass
10	Notify donors	Successful Notification	Successful Notification	Pass
11	Store blood request	Successful Entry	Successful Entry	Pass
12	Return notifications	Return the correct notification data	Return the correct notification data	Pass
13	Return notifications	No notification	No notification	Pass
14	Get history	Return correct history	Return correct history	Pass
15	Get rating	Returns the average of all donation made	Returns the average of all donation made	Pass
16	Get top ten donors	Returns the top then donors in the database	Returns the top then donors in the database	Pass
17	Change notification status	Successful Status Change	Successful Status Change	Pass
18	Send firebase notification	Successful Notification Sent	Successful Notification Sent	Pass

7.1.3 Test Case Results for Database:

ID	Input Data	Actual Result	Expected Result	Pass/Fail

1	“0320-3171108” and INSERT	Phone number inserted	Phone number inserted	Pass
2	“0320-3171108” and DELETE	User row with the corresponding phone number is deleted	User row with the corresponding phone number is deleted	Pass
3	User information and INSERT	User data inserted	User data inserted	Pass
4	User information and UPDATE	User data updated	User data updated	Pass
5	User information and DELETE	User data deleted	User data deleted	Pass
6	“0320-3171108” and RETRIEVE	User data retrieved	User data retrieved	Pass
7	Instance Token and INSERT	Instance token inserted	Instance token inserted	Pass
8	Instance Token and UPDATE	Instance token updated	Instance token updated	Pass
9	“0320-3171108” and RETRIEVE	Instance token retrieved	Instance token retrieved	Pass
10	Blood Request Notification and INSERT	Request information stored	Request information stored	Pass
11	Blood Request Notification and UPDATE	Request information updated	Request information updated	Pass
12	“0320-3171108” and RETRIEVE	Request information retrieved	Request information retrieved	Pass
13	68, 4.0 and INSERT	Donor points stored	Donor points stored	Pass
14	68, 4.0 and RETRIEVE	Donor points retrieved	Donor points retrieved	Pass
15	“0320-3171108”, 9236, INSERT	Verification code stored	Verification code stored	Pass
16	“0320-3171108”, 9036, UPDATE	Verification code updated	Verification code updated	Pass

17	“0320-3171108”, RETRIEVE	Verification code retrieved	Verification code retrieved	Pass

7.2 USABILITY TEST RESULTS

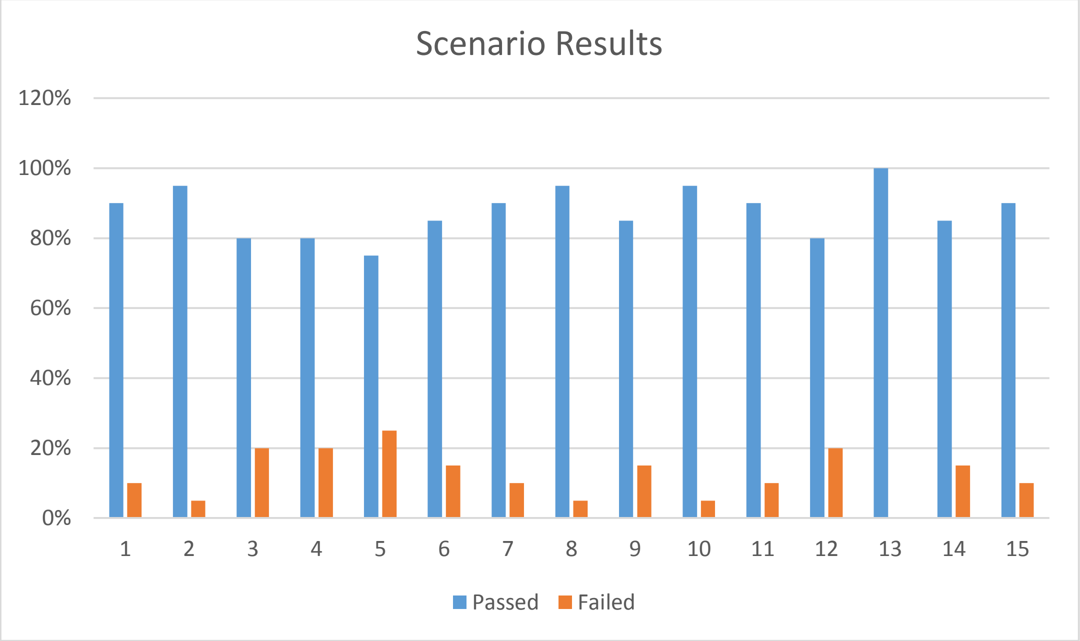

Figure 20 : Scenario Based Testing Results

The results of testing show that the system is fulfilling all the core functionalities required, the modules have been improved in the form of iterations until they are giving the expected results. Usability testing results also show that the system is easy to use and the scenarios have an average success rate of 85% to 90%. This also suggests that there is room for improvement in the process being followed for ‘blood request generation and acceptance’ so that it is more intuitive. The lowest success rate was for scenario number five in which the recipient has to give points to the donor for donating blood. The probable reason that has been identified is that after donation, the recipient thinks that the donation process is complete and forgets to give points. In order to cater for this the system should not allow the recipient to make any further request until all the previous donations have been rated. The rest of the scenarios have an acceptable success rate and can only be improved by making major changes in the donation process flow being followed.

Chapter 8

CONCLUSION AND FUTURE WORK

The problem was how can we ease the process of finding a nearby blood donor who is also eligible for blood donation. Through BlooDonate, we are making the process easier because required blood is just 2 clicks away from the recipient. Whenever someone is in need of blood he has to request the blood quantity, type, and place and the whole system will take care of his whole donor connection process.

For the time being, system scope is limited to a few number of users but its next versions can handle more users. Moreover, we want to make separate accounts for blood banks and hospitals so that they can also take part in making this process easier for humanity.

Chapter 9

REFERENCES

[1] Ouhbi, S., Fernández-Alemán, J., Toval, A., Idri, A. and Pozo, J. (2015). Free Blood Donation Mobile Applications. Journal of Medical Systems, 39(5).

[2] Rogers, R. (2009). Android application development. Sebastopol: O'Reilly.

[3] Soni, R. (2013). A Study on Android Application Development. Journal of Telematics and Informatics, 1(2).

[4] Yuan, S., Chang, S., Uyeno, K., Almquist, G. and Wang, S. (2015). Blood donation mobile applications: are donors ready?. Transfusion, 56(3), pp.614-621.

[5] Lu, J. and Gokhale, S. (2008). Performance Analysis of a Web Server. International Journal of Information Technology and Web Engineering, 3(3), pp.50-65.

[6] Schiano, W., Waguespack, L. and Yates, D. (2013). Apache Web Server. The International Journal of Design Management and Professional Practice, 6(1), pp.1-12.

[7] Lee, J. and Ware, B. (2003). Open source development with LAMP. Boston: Addison-Wesley.

[8] Glass, M. (2005). Beginning PHP5, Apache, MySQL web development. Indianapolis, Ind.: Wiley.

[9] Welling, L. and Thomson, L. (2008). PHP and MySQL Web development. Upper Saddle River, NJ: Addison-Wesley.

[10] Bulger, B., Greenspan, J. and Wall, D. (2004). MySQL/PHP database applications. Indianapolis, IN: Wiley Pub.

[11] Brown, J., Shipman, B. and Vetter, R. (2007). SMS: The Short Message Service. Computer, 40(12), pp.106-110.

APPENDIX

10.1 USE CASE DIAGRAMS

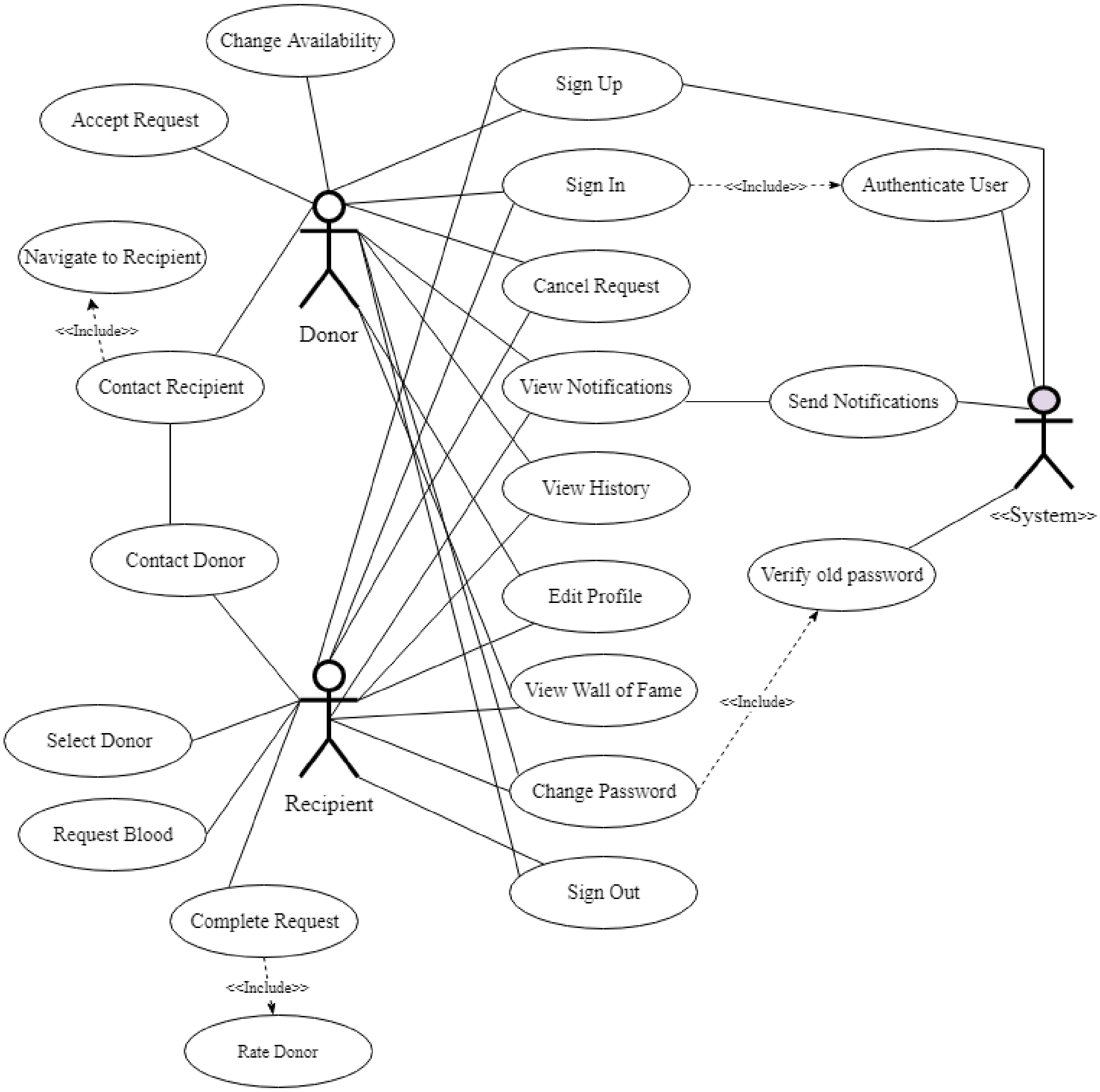

Figure 21 : System Use Case

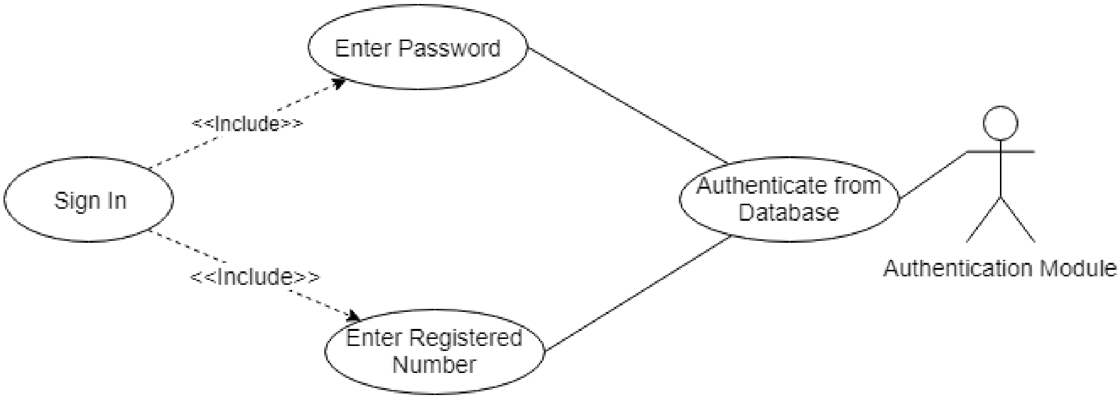

Figure 22 : Sign in Sub System Use Case Diagram

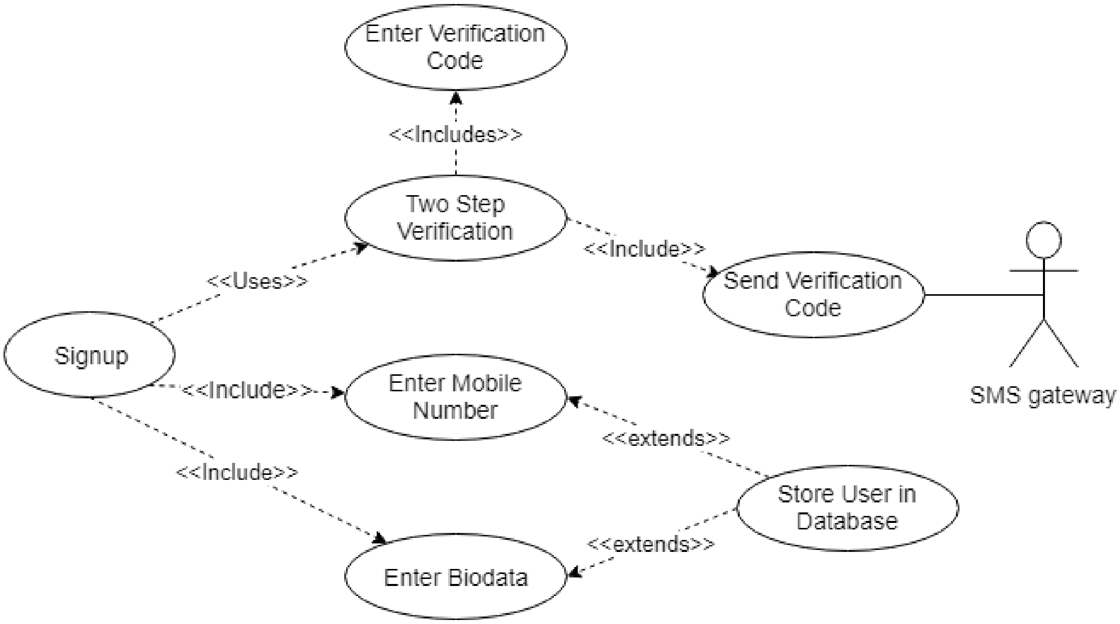

Figure 23 : Sign up Sub System Use Case Diagram

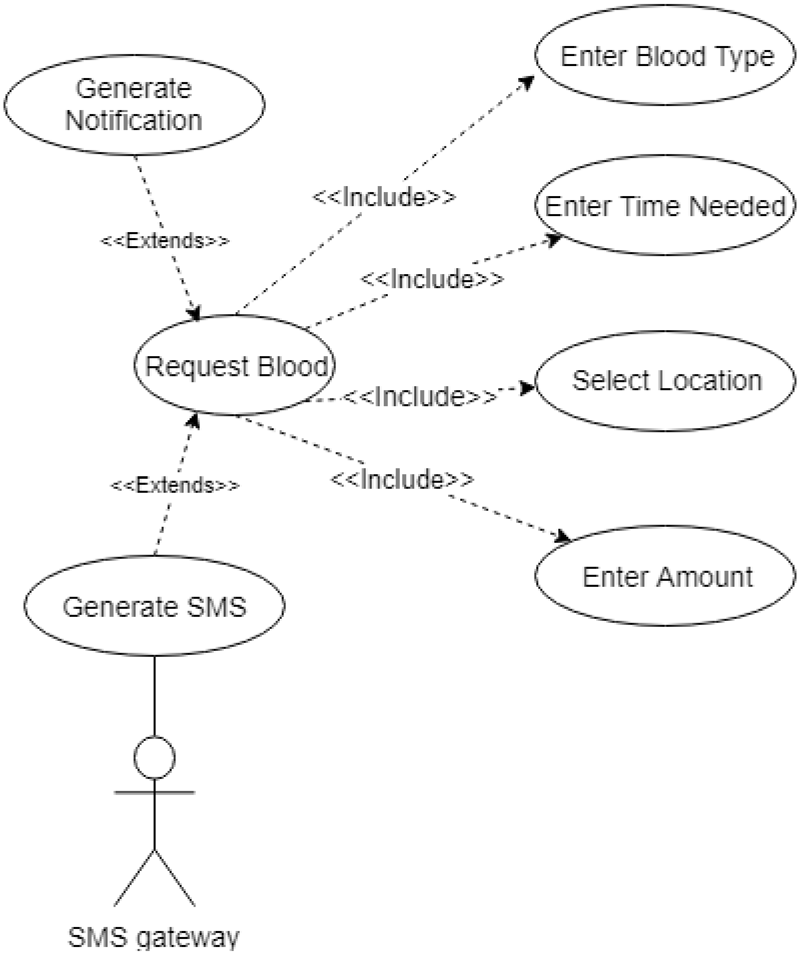

Figure 24 : Request Generation Sub System Use Case Diagram

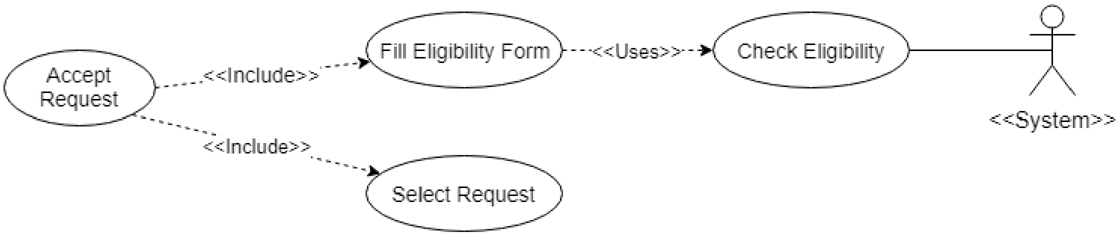

Figure 25 : : Accept Request Sub System Use Case Diagram

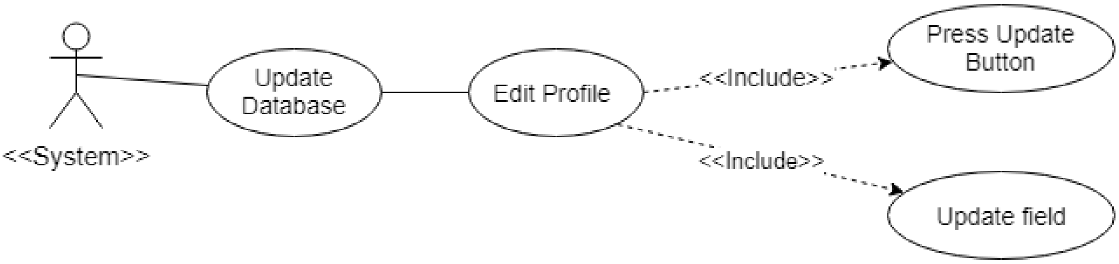

Figure 26 : Edit Profile Sub System Use Case Diagram

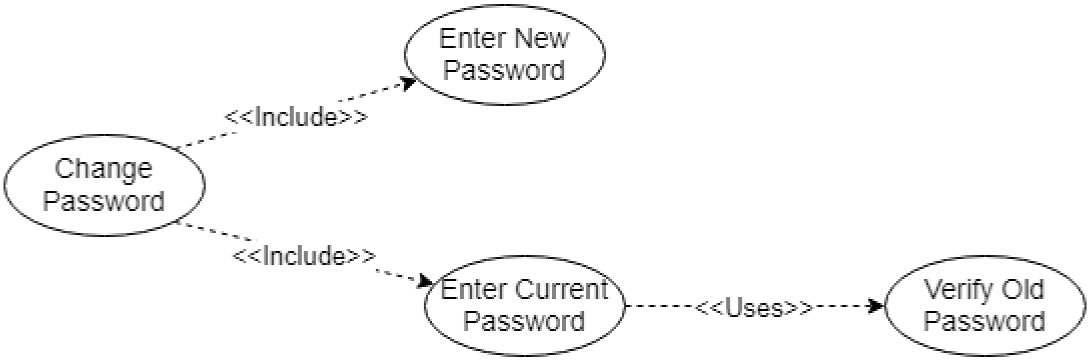

Figure 27 : Change Password Sub System Use Case Diagram

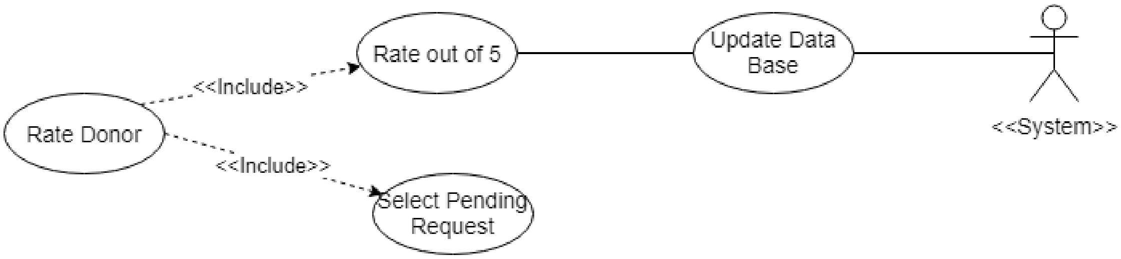

Figure 28 :Rate Donor Sub System Use Case Diagram

10.2 ACTIVITY DIAGRAMS

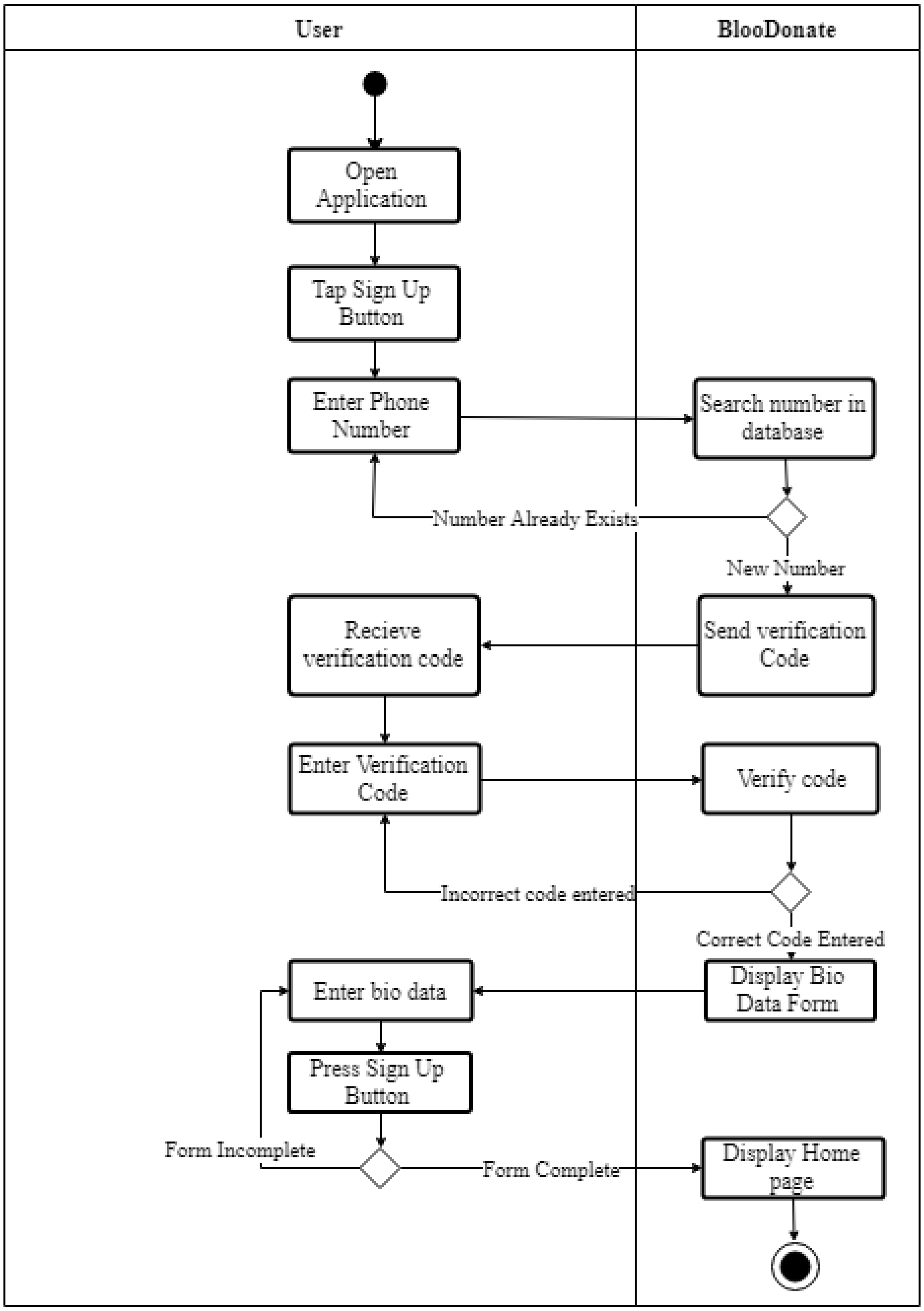

Figure 29 : Sign Up Activity Diagram

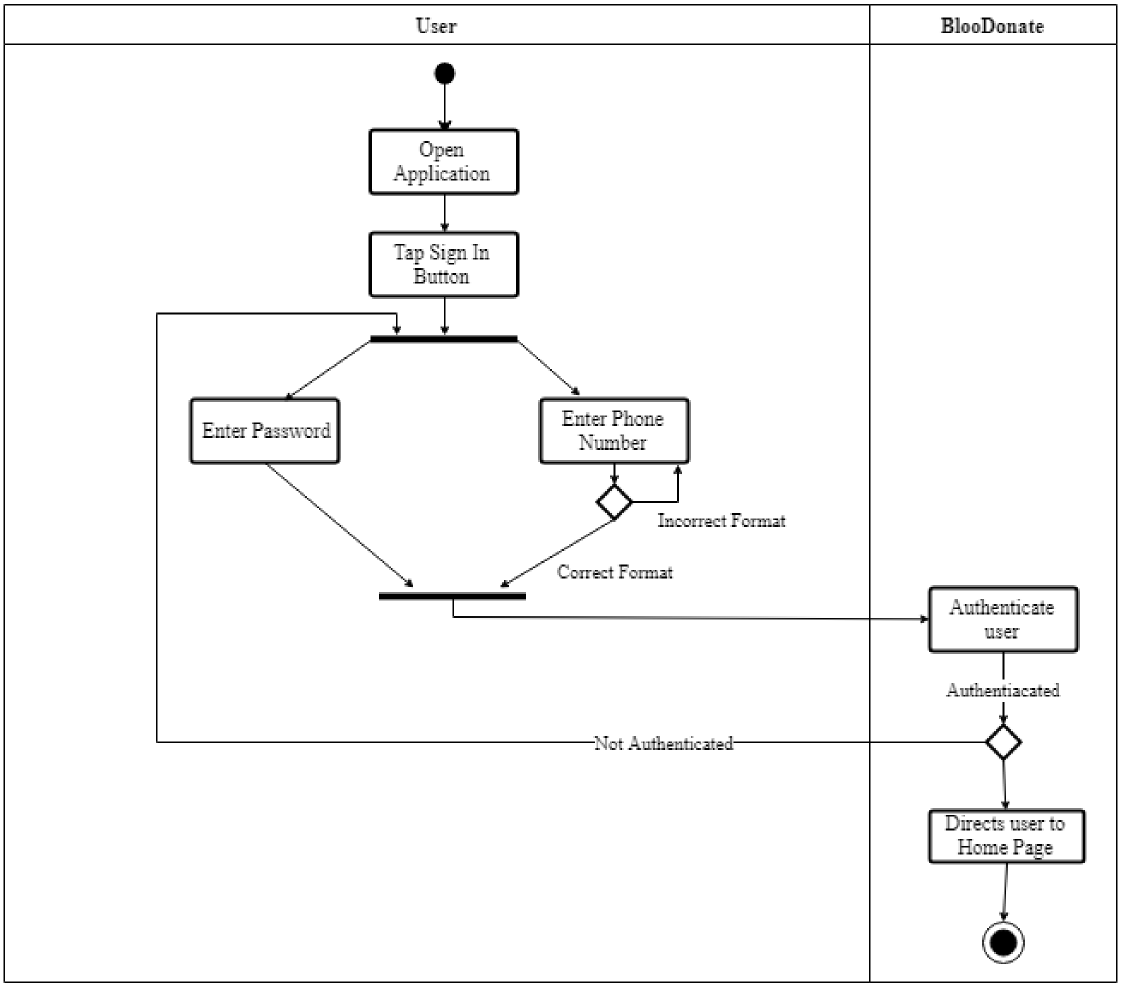

Figure 30 : Sign In Activity Diagram

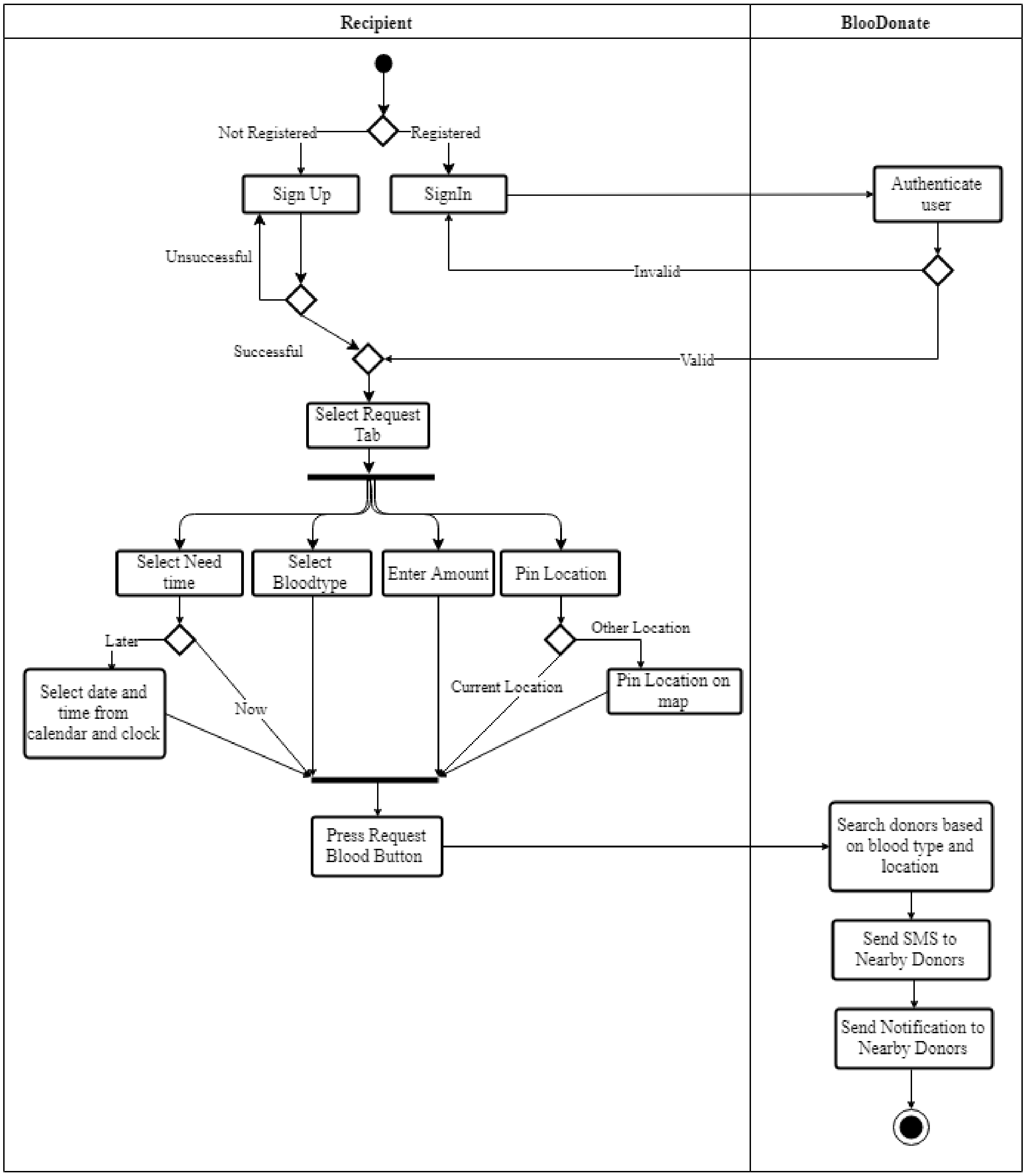

Figure 31 : Generate Request Activity Diagram

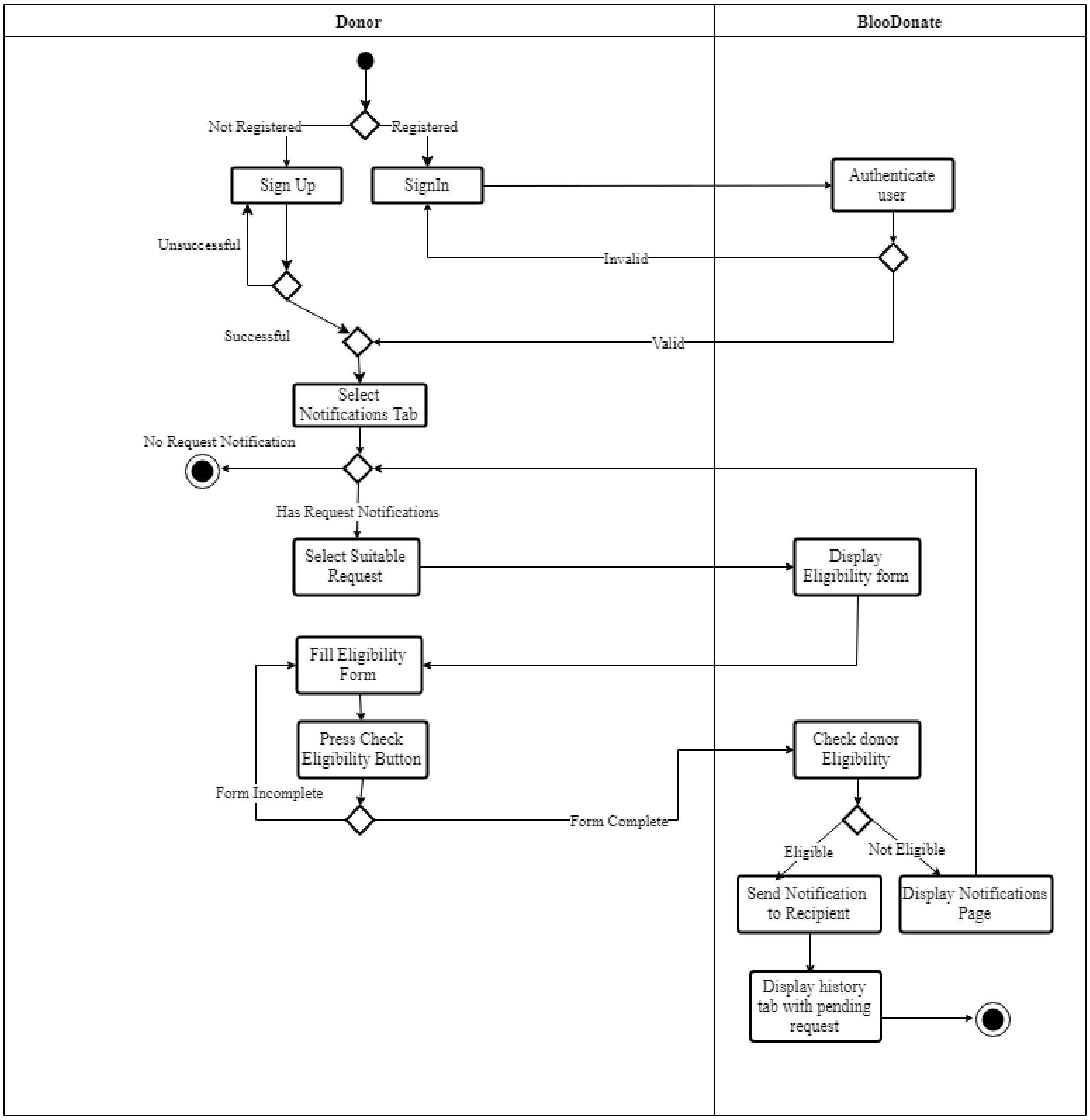

Figure 32 Accept Request Activity Diagram

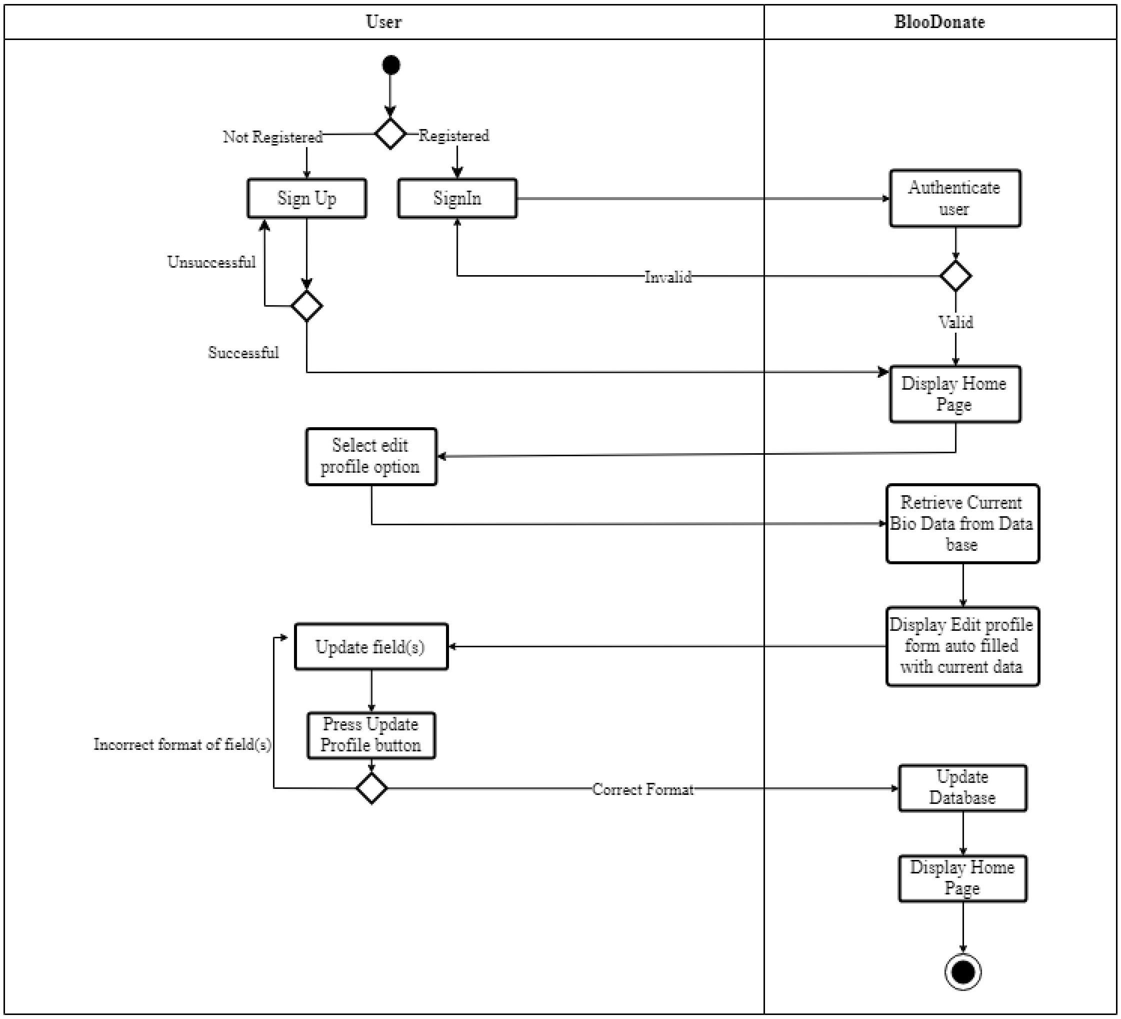

Figure 33 : Edit Profile Activity Diagram

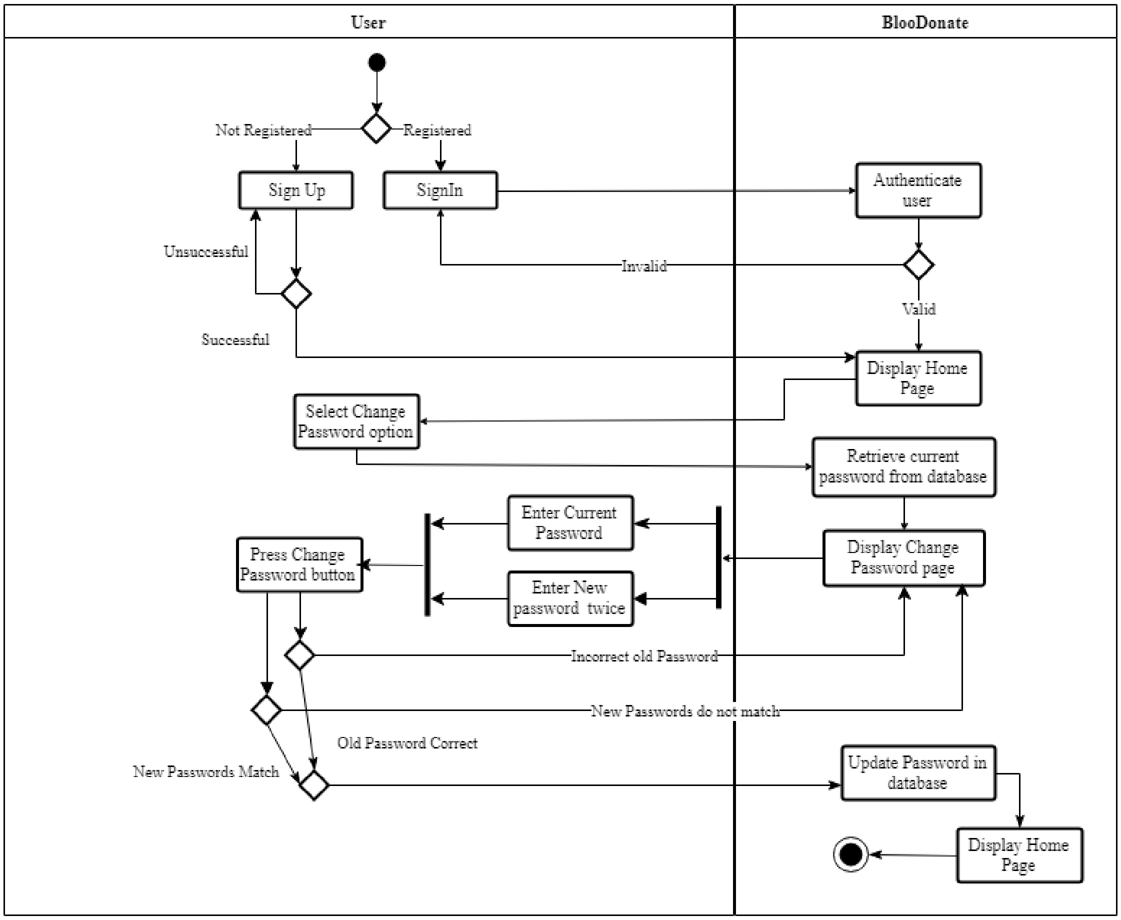

Figure 34 : Change Password Activity Diagram

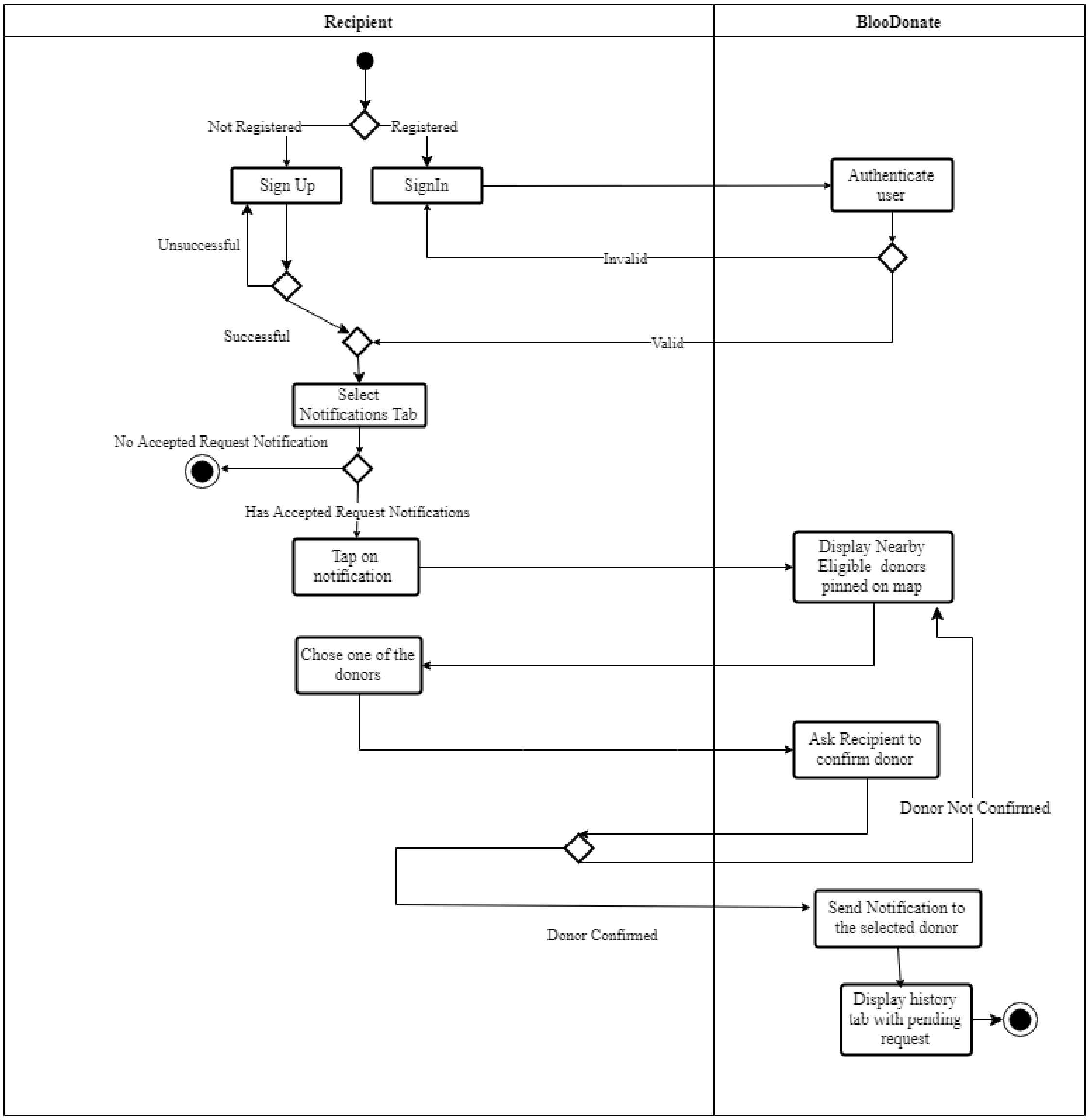
Recipient
BlooDonate
Not Registered
Registered
Sign Up
SignIn
Authenticate user
Unsuccessful
Invalid
Successful
Valid
Select Notifications Tab
No Accepted Request Notification
Has Accepted Request Notifications
Tap on notification
Display Nearby Eligible donors pinned on map
Chose one of the donors
Ask Recipient to confirm donor
Donor Not Confirmed
Send Notification to the selected donor
Donor Confirmed
Display history tab with pending request

Figure 35 : Select Donor Activity Diagram

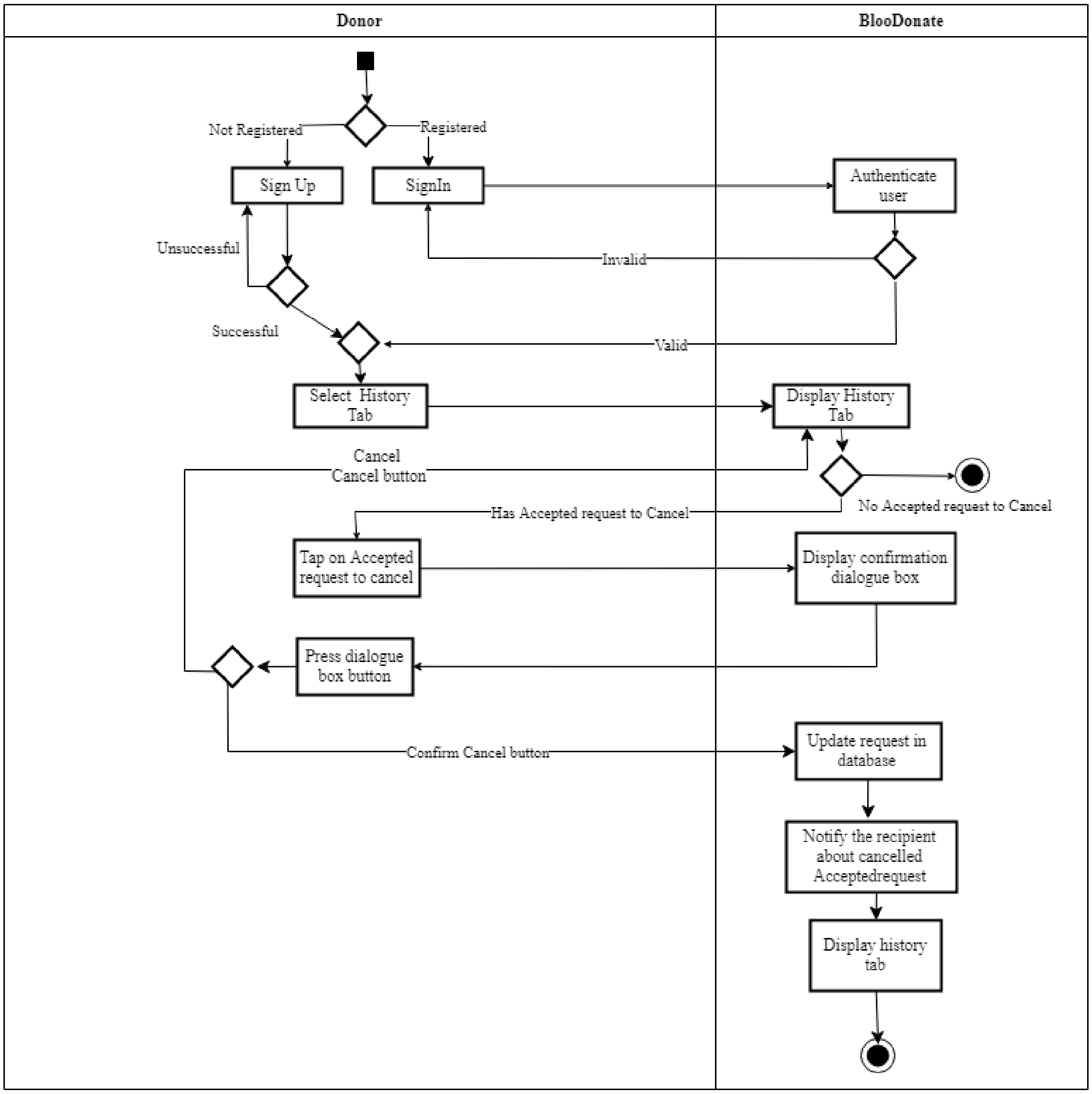
Donor
BlooDonate
Not Registered
Registered
Sign Up
SignIn
Authenticate user
Unsuccessful
Invalid
Successful
Valid
Select History Tab
Display History Tab
Cancel
Cancel button
No Accepted request to Cancel
Has Accepted request to Cancel
Tap on Accepted request to cancel
Display confirmation dialogue box
Press dialogue box button
Confirm Cancel button
Update request in database
Notify the recipient about cancelled Acceptedrequest
Display history tab

Figure 36 :Donor Request Cancel Activity Diagram

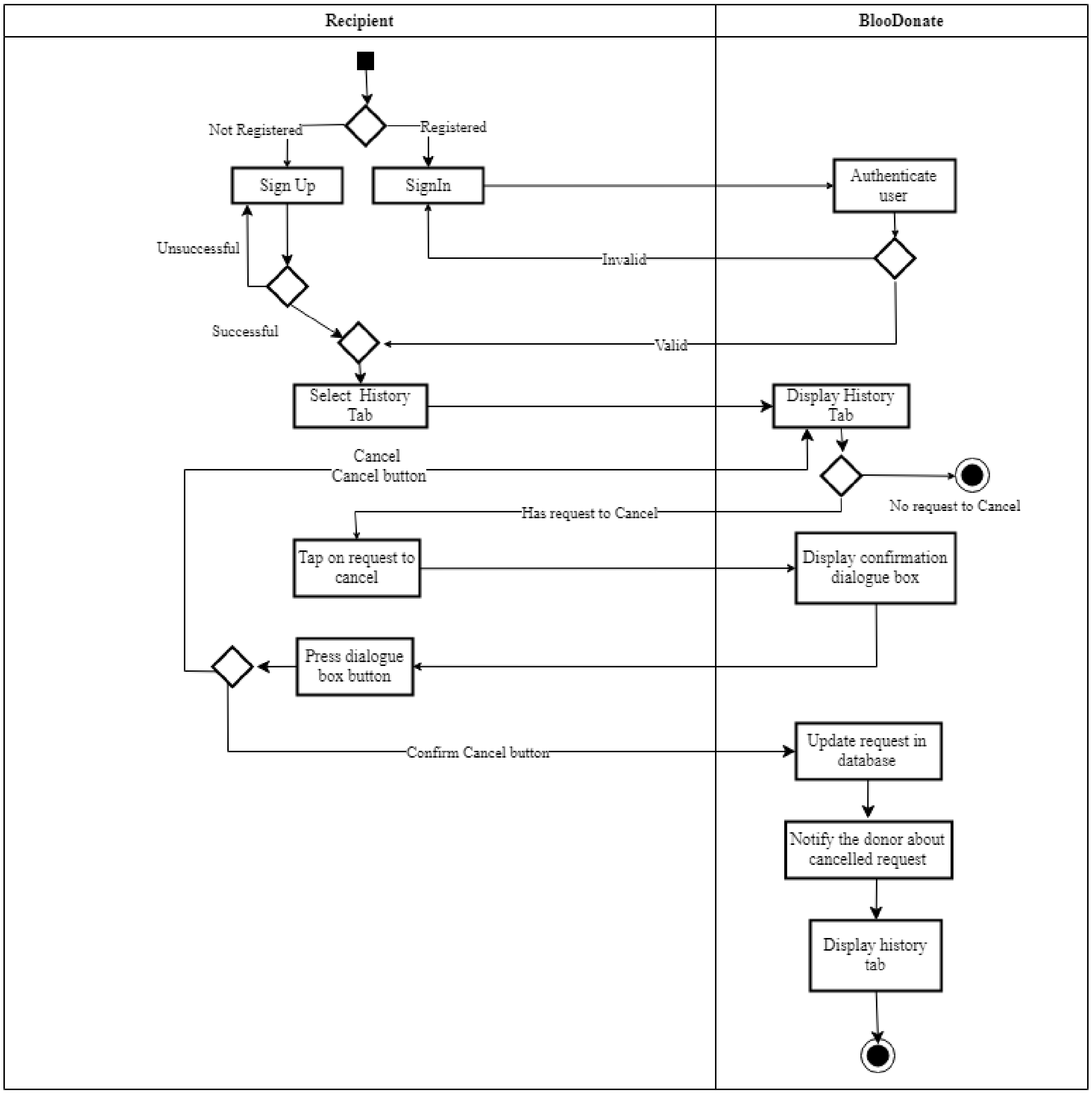

Figure 37 :Recipient Request Cancel Activity Diagram

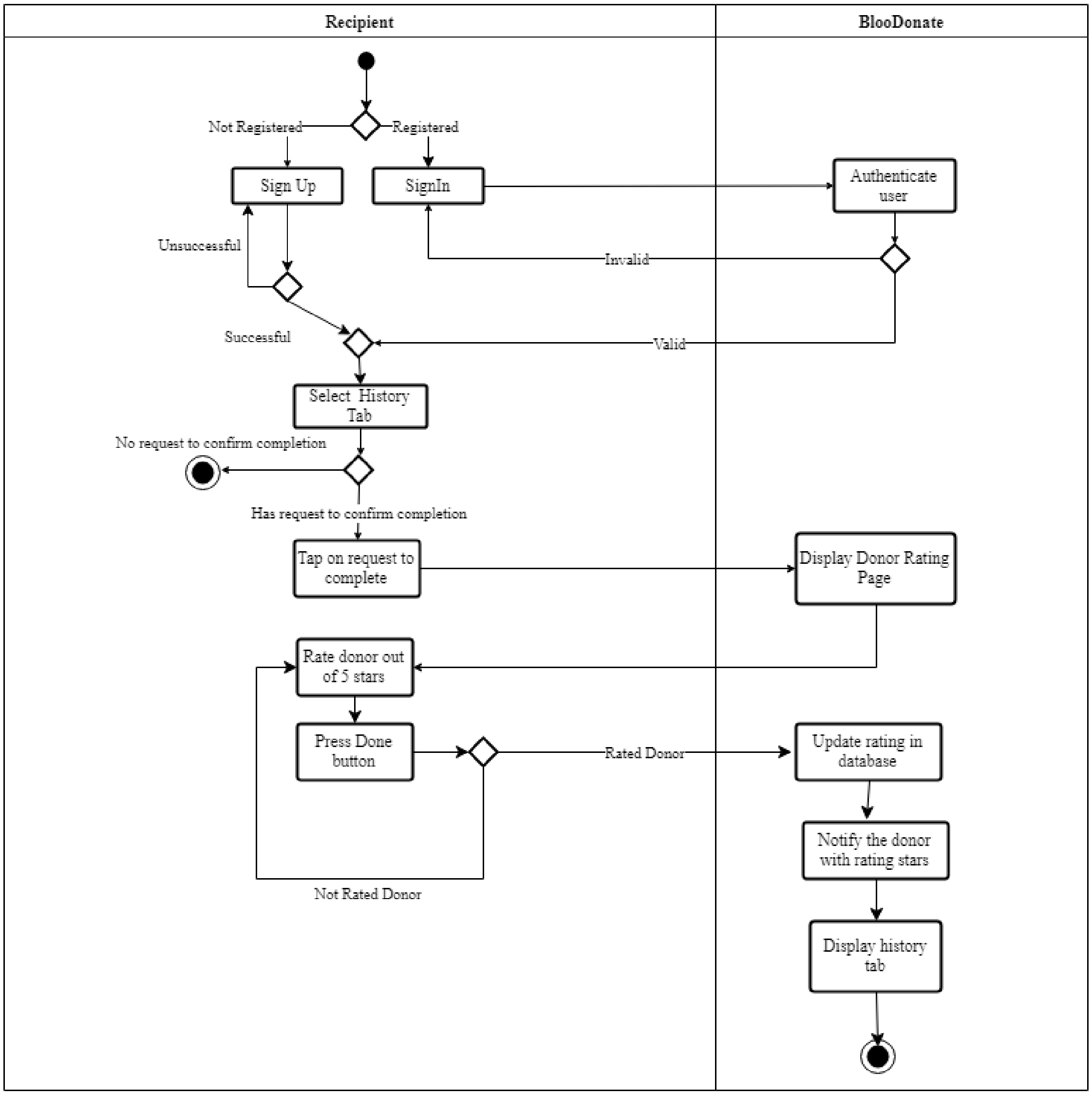
Recipient
BlooDonate
Not Registered
Registered
Sign Up
SignIn
Authenticate user
Unsuccessful
Invalid
Successful
Valid
Select History Tab
No request to confirm completion
Has request to confirm completion
Tap on request to complete
Display Donor Rating Page
Rate donor out of 5 stars
Press Done button
Rated Donor
Update rating in database
Notify the donor with rating stars
Not Rated Donor
Display history tab

Figure 38 : Rate Donor Activity Diagram