

# COVER PAGE

## **Declaration**

We hereby declare that the project work entitled “**LifeSource Blood Bank** ” is an authentic record of our own work carried out as requirements of Capstone Project for the award of B.Tech degree in Computer Science and Engineering from **Lovely Professional University, Phagwara**, under the guidance of **Assistant Professor Ms. Manjit Kaur**, from June 2019 to October 2019. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

**Name of Student:** Sonukumar Meena

**Roll. No. :** 11607098

**Course:** B.Tech CSE

**Semester:** 7th

**Name of Student:** Sai Venkata Seshu Reddy

**Roll. No. :** 11611983

**Course:** B.Tech CSE

**Semester:** 7th

**Name of Student:** Mali Rakesh Reddy

**Roll. No. :** 11606966

**Course:** B.Tech CSE

**Semester:** 7th

## **Certificate**

This is to clarify that the project entitled “**LifeSource Blood Bank**” carried out in Capstone Project is a bonafide work done by **Sonukumar Meena (11607098)**, **Sai Venkata Seshu Reddy (11611983)** and **Mali Rakesh Reddy (11606966)** is submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

**Name of Supervisor:** Manjit Kaur

**Date:**

## **Acknowledgement**

It gives us a great sense of pleasure to present the report of the B.Tech Capstone Project undertaken during B.Tech fourth year. This project in itself is an acknowledgement to the inspiration, drive and technical assistance contributed to it by many individuals. This project would never have seen the light of day without the help and guidance that we have received.

Our heartiest thanks to -----, Head of Dept., Department of CSE for providing us with an encouraging platform to develop this project, which thus helped us in shaping our abilities towards a constructive goal.

We owe a special debt of gratitude to **Ms. Manjit Kaur**, Assistant Professor Department of CSE, for her constant support and guidance throughout the course of our work. Her sincerity, thoroughness and perseverance has been a constant source of inspiration for us. She has showered us with all his extensively experienced ideas and insightful comments at virtually all stages of the project and has also taught us about the latest industry-oriented technologies.

We also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind guidance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Sonukumar Meena

Sai Venkata Seshu Reddy

Mali Rakesh Reddy

# **Table of Contents**

<b>1. INTRODUCTION</b>	<b>6</b>
1.1. PROJECT OVERVIEW	6
1.2. PROJECT DESCRIPTION	6
<b>2. PROBLEM STATEMENT</b>	<b>7</b>
<b>3. EXISTING SYSTEM</b>	<b>8</b>
3.1. DFD FOR PRESENT SYSTEM	8
3.2. WHAT'S NEW IN THE SYSTEM TO BE DEVELOPED	9
<b>4. PROBLEM ANALYSIS</b>	<b>10</b>
4.1. PRODUCT DEFINITION	10
4.1.1. ADMIN:	10
4.1.2. DONORS:	10
4.1.3. DONOR REGISTRATION:	11
4.1.4. MODIFYING DONOR INFORMATION:	11
4.1.5. ACCEPTORS:	12
4.1.6. DONOR SEARCH:	12
4.2. FEASIBILITY ANALYSIS	13
4.2.1. TECHNICAL FEASIBILITY	13
4.2.2. OPERATIONAL FEASIBILITY	14
4.2.3. ECONOMIC FEASIBILITY	14
4.3. Project Plan	15
<b>5. SOFTWARE REQUIREMENT ANALYSIS</b>	<b>16</b>
5.1. INTRODUCTION	16
5.2. GENERAL DESCRIPTION	16
5.2.1. PRODUCT PERSPECTIVE	16
5.2.2. PRODUCT FUNCTIONS	16
5.2.3. USER CHARACTERISTICS	17
5.2.4. GENERAL CONSTRAINTS	17
5.3. REQUIREMENTS ANALYSIS	17
<b>6. DESIGN</b>	<b>18</b>
6.1. SYSTEM DESIGN	18
6.2.1. LEVEL 0 DFD:-	19

6.2.2. LEVEL 1 DFD:-	20
6.2.3. USE CASE MODEL	21
6.2.4. E-R Diagram	22
6.3. FLOWCHART	23
7. TESTING	24
7.1. FUNCTIONAL TESTING	24
7.1.1. REGISTRATION PAGE TESTING	24
7.1.2. LOGIN PAGE TESTING	25
7.2. STRUCTURAL TESTING	26
7.3. LEVELS OF TESTING	26
7.3.1 UNIT TESTING	27
7.3.2 INTEGRITY TESTING	27
8. IMPLEMENTATION	28
8.1. IMPLEMENTATION OF PROJECT	28
9. USER MANUAL	37
9.1. HOME PAGE	37
9.2. REGISTRATION	38
9.3. REQUEST FOR BLOOD	39
9.5. CHANGE PASSWORD	41
9.6. BLOOD DONATED	42
10. SOURCE CODE	43
11. BIBLIOGRAPHY	44

# **1. INTRODUCTION**

## **1.1. PROJECT OVERVIEW**

The Blood Donation App is a platform where any person who is interested in donating the blood can register himself and if any general consumer wants to make request blood online he can also take the help of this App. Admin is the main authority who can do addition, deletion, and modification if required.

## **1.2. PROJECT DESCRIPTION**

This project is aimed to develop an online Blood Donation App. The entire project has been developed keeping in view of the distributed client server computing technology, in mind.

The project has been planned to be having the view of distributed architecture, with centralized storage of the database. The application for the data has been planned. #####

# **2. PROBLEM STATEMENT**

In present scenario searching for blood donors can take place through blood bank centers or by toll free numbers. So far it is a time taken process. Because it is having lots of manual work. It is a waste of time to go to blood bank if the blood of a particular group is not available and most of the time user has to wait in queue.

The percentage of people donating blood is increasing day by day due to awareness to donate blood for those needed. The blood received have to be managed thoroughly so that there will be no negative effect to the blood receiver once they received blood. The blood donation event schedule is normally advertised to the public so that they are aware of the blood donation campaign period. At the blood house unit, the staff and nurses only are

informed about the blood donation schedule for each month on the whiteboard at the blood house. So they are using manual way in informing the schedule. The problem arises when the space provided is not enough.

The medium used to inform the staff about the schedule of the month is using whiteboard and it is written by using whiteboard markers. Therefore, the writing tends to become unclear. The public did not have knowledge about blood donation. There are brochures distributed to the donor but not to the public because they only available at blood donation house. Hence, the public are not getting any details information about blood donation unless they go to the blood donation house.

One of the factors of the public afraid to donate their blood is they believe in myths. The myths that they always believe are, if they donate their blood they will become fat and if they donate their blood, their blood will become less in total amount and they will become pale. This LifeSource Blood Donation system should provide more information in order to educate the public so that they know blood donation will not give bad effects. By giving awareness to the public, this will increase volunteers to donate their blood.

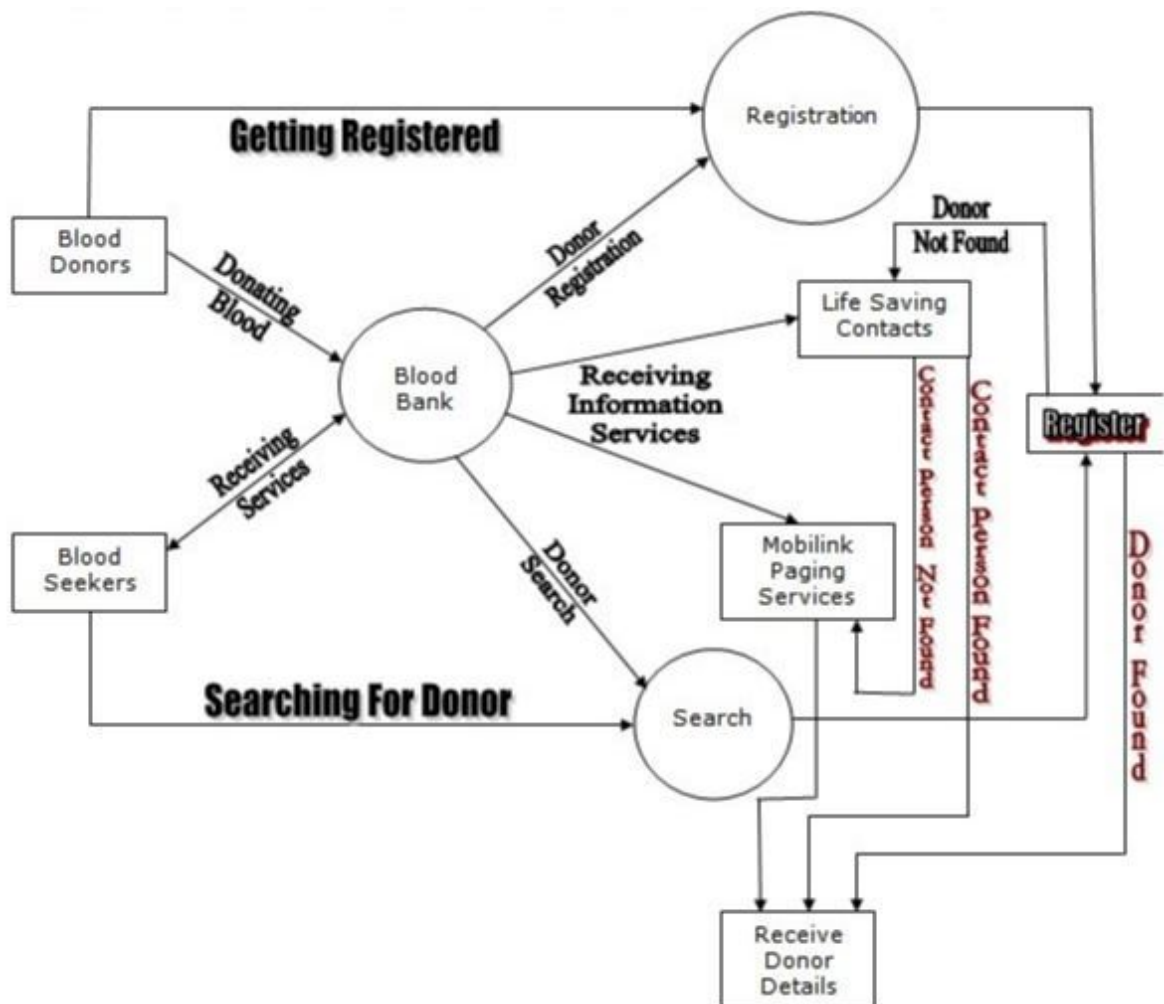
### **3. EXISTING SYSTEM**

There are several flaws associated with existing existing system includes:

1. The app doesn't allow to search for blood donors. The blood group is chosen but when selecting a country, a message pops up saying "Select any blood group".
2. When trying to register as Blood donor OTP does not come on time and time runs out.
3. Interface is not user friendly as it seems very confusing when we select a particular blood group, country or state.
4. Listed members are not helpful when needed, not dependable.
5. Sometimes app crash during registration process.



### 3.1. DFD FOR PRESENT SYSTEM



### 3.2. WHAT'S NEW IN THE SYSTEM TO BE DEVELOPED

The proposed Blood Bank App helps the people who are in need of a blood by giving them all the details of blood by giving them all the details of blood group availability or regarding the donors with the same blood.

The people in need of blood can search for the donors by giving their blood group and city name. It saves time as he can search donors online without going anywhere. Using this App user can get blood in time and can save his relative or friend life. This App work 24x7 so user can get information of blood donor any time. Blood donor can also get registered and save the life of another person. The main benefit of this App is the information of available blood group. When blood is needed in the operation then people have very less time to get

the blood available so if get the information like who can give him blood in time in his city is life saving. And here our App work, whenever a person needs blood he get information of the person who has the same blood group he needs.

### **Advantages:**

- User friendliness, we provided in the application with various controls.
- The system makes the overall project management much easier and flexible.
- Readily upload the latest updates, allows user to edit his/her profile.
- It provides a high level of security with different level of authentication.
- App interface is very reliable and very easy to use for anyone without any confusion.

## **4. PROBLEM ANALYSIS**

### **4.1. PRODUCT DEFINITION**

Online Blood Bank management system is to provide services for the people who are in need of blood by getting help from the donors who are interested in donating blood for the people. There are seven main modules in this system.

#### **4.1.1. ADMIN:**

Admin can manage both donors & acceptors. He can add or remove any user from the system. Each member in a donor & acceptor is given a user id and password, which identifies him uniquely. From admin module use can change donor details, delete donor or change the password.

- Change Password
- Modify donor details
- delete donor details
- Logout

Whenever a user wants to change his / her password he can select the change password option. The system displays the form, which asks him for his old password and new password. The system then compares the old password with the existing password in the database and if they match then the password is set to the new password in the database.

#### **4.1.2. DONORS:**

From this module user can create their account, when user create his account the user get a user id and password, which identifies him uniquely. From this module user can search donor for blood and can also refer his friend to become a donor. Donor can also get information like when he donated blood or when he will be able to donate blood.

#### **4.1.3. DONOR REGISTRATION:**

In this module, people who are interested in donating blood get registered in my site and give his overall details related to him, i.e. he fills in a registration form by giving the total details such as name, address, city, sex, wt, dob, blood group, telephone numbers, e-mail address, etc. He was also given two fields' username and password to fill such that he was a registered donor and he can enter the login form with his username and password and can modify his details if needed.

#### **4.1.4. MODIFYING DONOR INFORMATION:**

The registered donor only is able to modify his details; no other person can modify his details as there was a login form which restricts others from entering the username and

password providing high security for the details given by the donor. If at all the donor wants to modify his details, he was forced to give his username and password to enter in. After giving the username and password it checks for the donor whether he is an existing donor or not and if the username and password matches, he can then able to modify his total details. If the username and password do not exist then he gets a message as 'Wrong ID and Password Entered, Try Again'.

Following links are available on donor and acceptor module.

- Why donate blood
- Who needs blood
- Find A Donor.
- Refer A Friend.
- Change password.
- Logout

#### **4.1.5. ACCEPTORS:**

This module helps user to find blood group. When user click on find a blood group system ask him to enter blood group he want to search. After entering the blood group, system search for the availability of the blood group and give him the list of the donors who has the same blood group. Whenever a user wants to change password he can select the change password option. Then the system asks the user to enter old username and password then system check the credentials and change the password. Clicking on the logout button user can log out from the system.

- Find A Donor.
- Refer A Friend.
- Change password.

- Find a Blood group.

- Logout

#### **4.1.6. DONOR SEARCH:**

The people who are in need of blood can search in our site for getting the details of donors having the same blood group and with in the same city. They can directly click on the link search for a donor and can select a city name as well as the blood group which he needs. He then gets the details of the donors who exist with in the city and the same blood group that he has selected. If no match was found for the city and group selected by him he gets a message 'SORRY DONORS ARE NOT AVAILABLE WITH THE FOLLOWING BLOOD GROUP AND AREA'.

### **4.2. FEASIBILITY ANALYSIS**

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation.

- Technical Feasibility
- Operation Feasibility
- Economical Feasibility

#### **4.2.1. TECHNICAL FEASIBILITY**

- The technical issue usually raised during the feasibility stage of the investigation includes the following:
- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

The software and hard requirements for the development of this project are not many and already available in-house at our university android lab or are available as free as open source. The work for the project is done with current equipment and existing software technology.

#### **4.2.2. OPERATIONAL FEASIBILITY**

Proposed projects are beneficial only if they can be turned into information system. That will meet the organization's operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following?

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well -planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

#### **4.2.3. ECONOMIC FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial must equal or exceed the costs.

The system is economically feasible. It does not require any additional hardware or software. Since the interface for this system is developed using the existing resources and technologies. There is nominal expenditure and economical feasibility for certain.

#### **4.3. Project Plan**

## **5. SOFTWARE REQUIREMENT ANALYSIS**

### **5.1. INTRODUCTION**

**Purpose:** The main purpose for preparing this document is to give a general insight into the analysis and requirement of the system or situation and for determining the operating characteristics of the system.

**Scope:** This Document plays a vital role in the development life cycle (SDLC) and it describes the complete requirement of the system. It is meant for use by the developers and



will be the basic during testing phase. Any changes made to the requirement in the future will have to go through a formal approval process.

## **5.2. GENERAL DESCRIPTION**

Through this section a description is given about the characteristics about the entire system.

### **5.2.1. PRODUCT PERSPECTIVE**

This project is mainly towards persons who are willing to donate blood to the patients. Through this system it will be easier to find a donor for exact blood type and easy to build the connection between donor and the blood bank authorities or consumer. The main intent of building this software is to formal the procedure of blood donation and motivate donors in order to donate blood.

### **5.2.2. PRODUCT FUNCTIONS**

- Admin
- Donor
- Seeker or Acceptor

### **5.2.3. USER CHARACTERISTICS**

In here the App admin and the donor are the system users. According to our assumptions the donor who will register to the App can understand simple facts and instructions about blood donation. He can fill his/her details and can use App effortlessly.

### **5.2.4 GENERAL CONSTRAINTS**

- The program will be written in xyz **language**
- The system will mainly running on Online BloodDonation App.

- Every donor will get a user name and a password in order to log into the system.

### 5.3. REQUIREMENTS ANALYSIS

This document is generated as output of requirement analysis. The requirement analysis involves obtaining a clear and thorough understanding of the product to be developed.

Requirement ID	Requirement Name	Requirement Description
1.	Registration of Donor	Requires an interface for the registration details of the donor.
2.	Donor Details	Attributes:- Donor ID, Name, Age, Address, Contact, Email, Blood Group, Gender, Date
3.	Health Information	Body, Pulse rate, Hemoglobin, Blood Pressure.
4.	Blood Bank App	Requires an interface to display the details of the blood bank Attributes- Name, Address, Contacts, Location.
5.	Login	Require interface for login, Change of password and Password recovery. Attributes:- Email, Password, New Password.

Each requirement in this section should be:

1. Correct
2. Reliable
3. Unambiguous
4. Verifiable
5. Prioritized
6. Complete
7. Consistent

## **6. DESIGN**

### **6.1. SYSTEM DESIGN**

A data flow diagram is a graphical tool used to describe and analyze the movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams.

The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purposes.

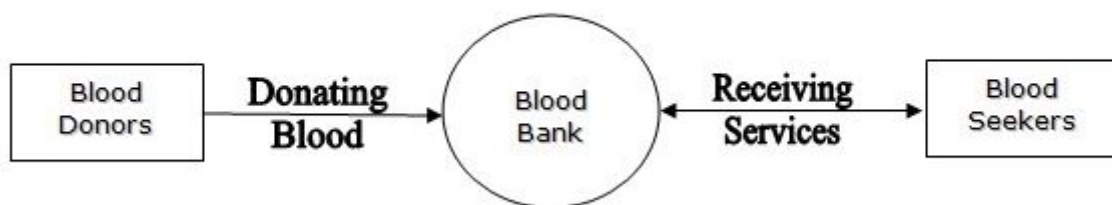
The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is

often called context diagram. It consists of a single process, which plays a vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD. The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for an analyst to understand the process.

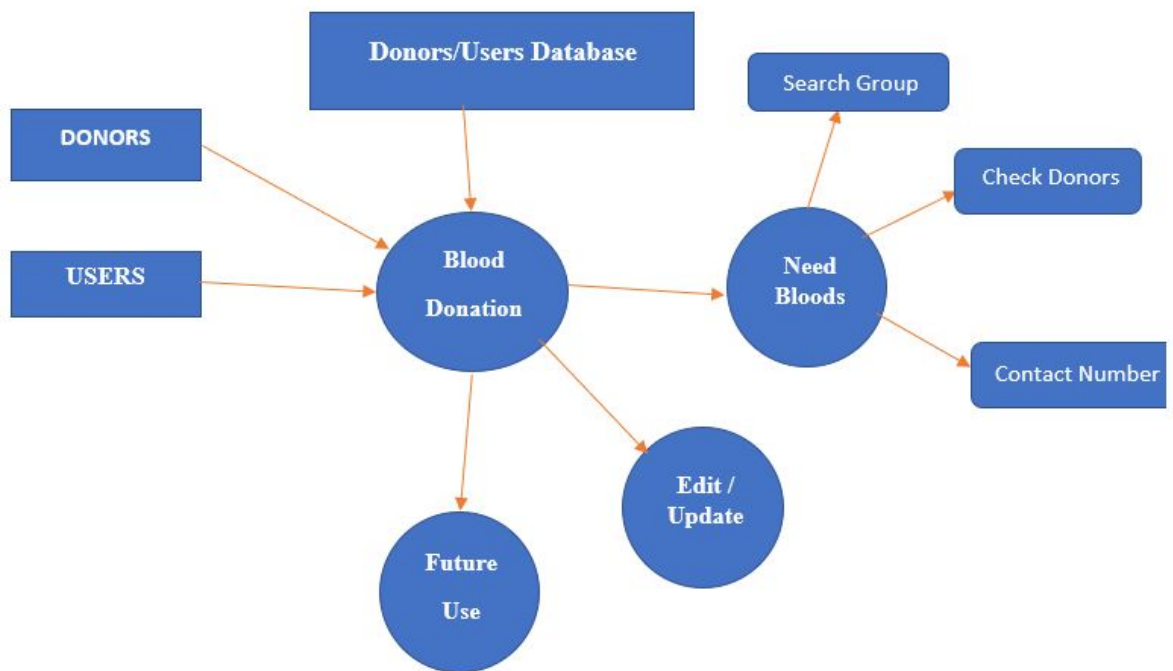
Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical form, this lead to the modular design.

A DFD is also known as a “bubble Chart” has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

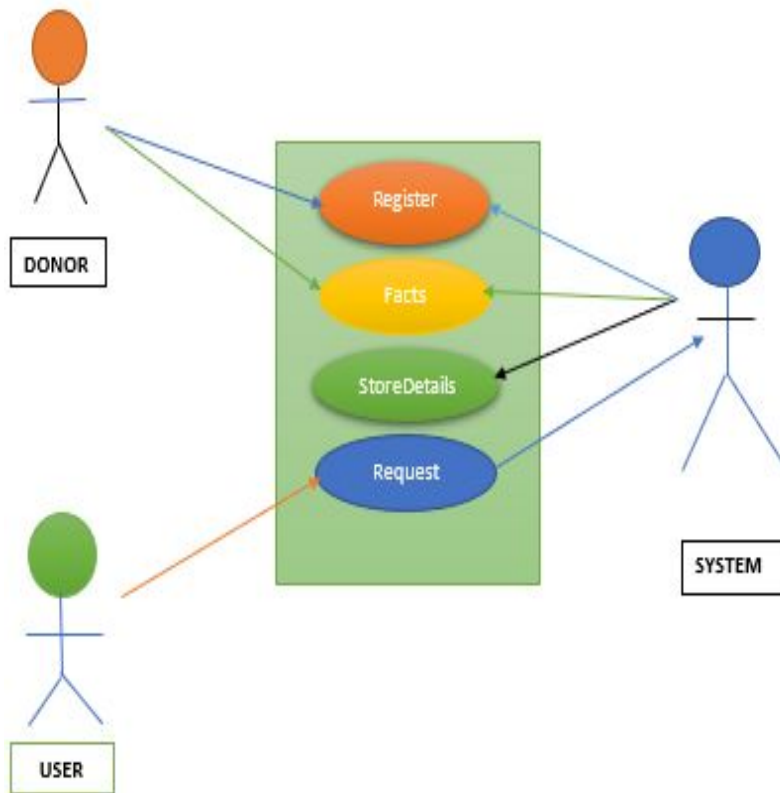
### 6.2.1. LEVEL 0 DFD:-



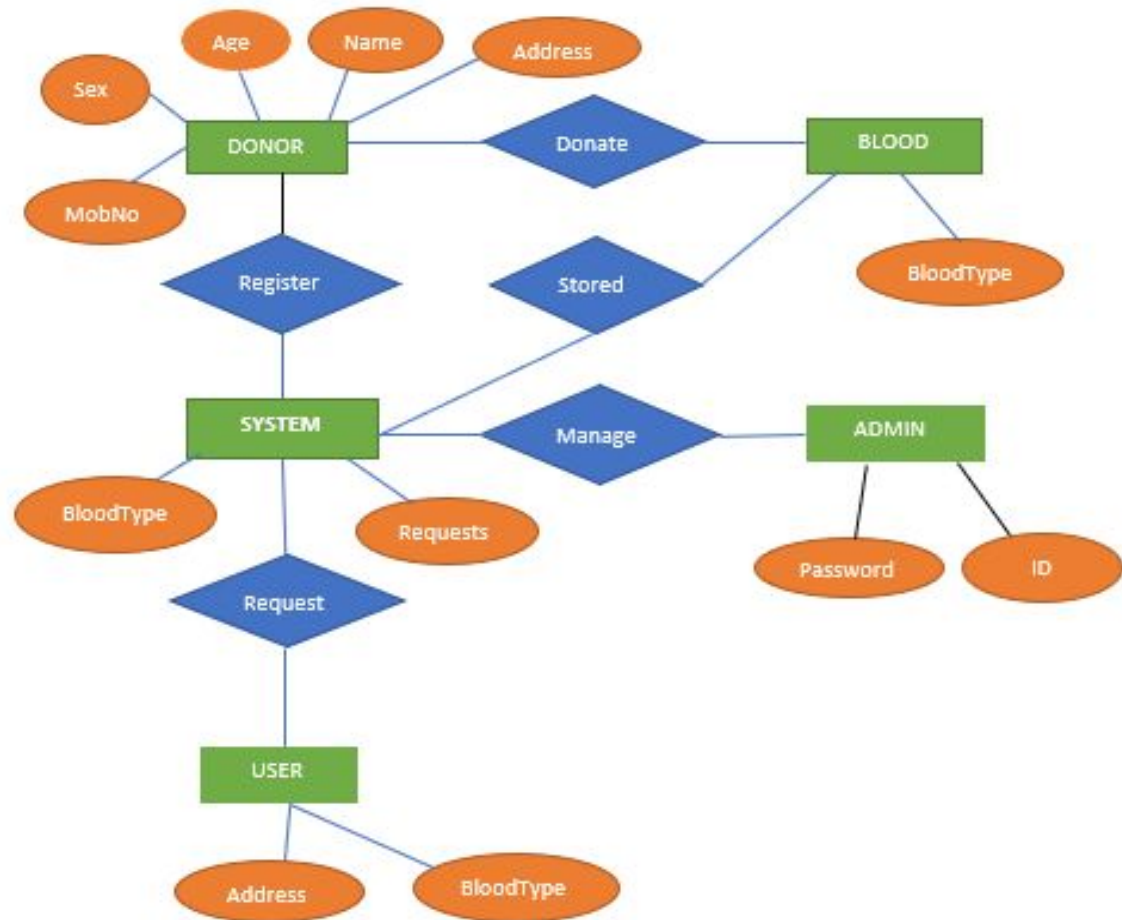
### 6.2.2. LEVEL 1 DFD:-



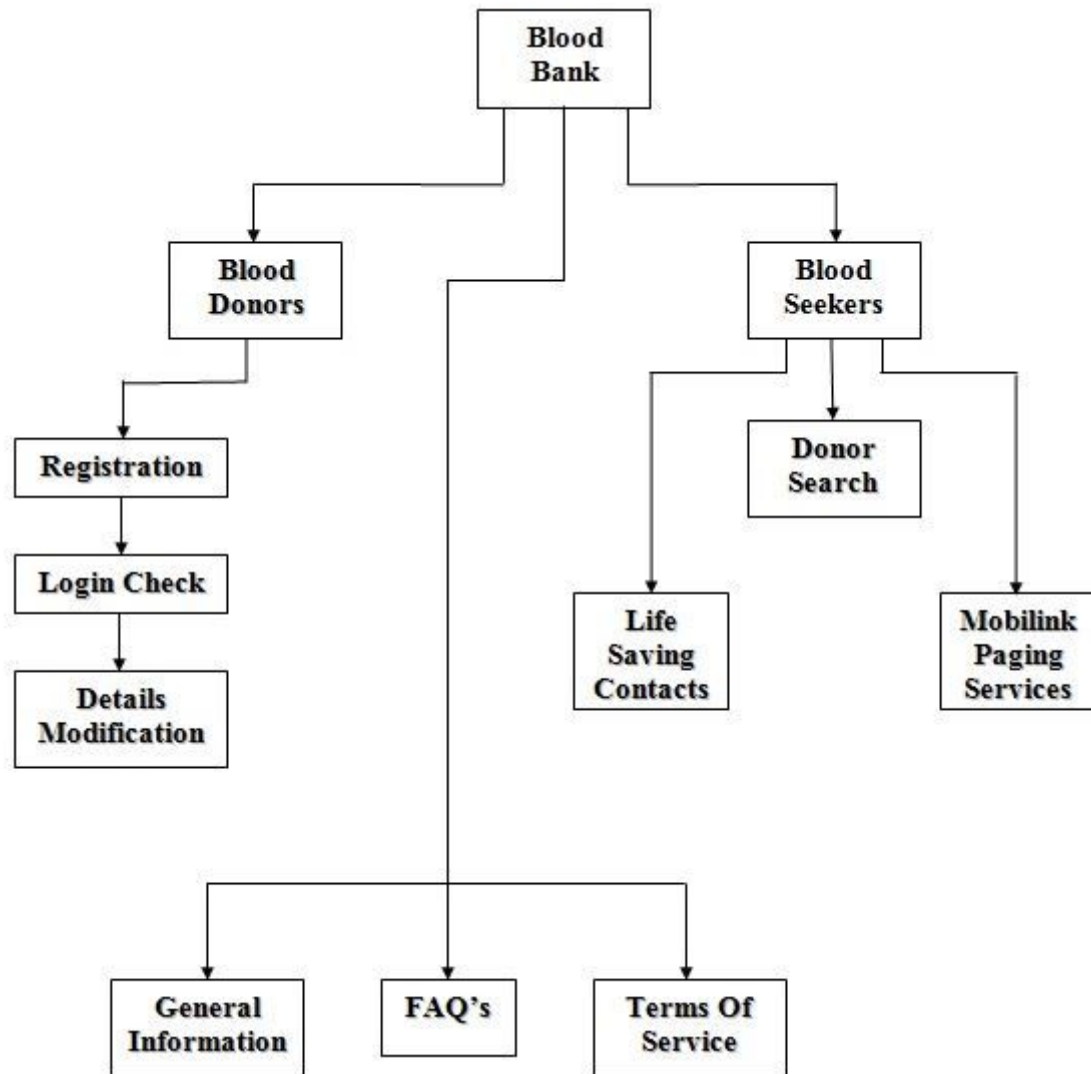
### 6.2.3. USE CASE MODEL



### 6.2.4. E-R Diagram



### 6.3. FLOWCHART





## **7. TESTING**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

### **7.1. FUNCTIONAL TESTING**

Here the system is a black box whose behavior is determined by studying its inputs and related outputs. The key problem is to select the inputs that have a huge probability of being members of a set in may case; the selection of these test cases is based on the previous studies.

### 7.1.1. REGISTRATION PAGE TESTING

Test Id	Test case title	Description	Expected outcome	Result
1.	Successful user Verification.	Registration to the system should be tried by admin with valid name, email, password, and phone no.	Registration should be successful and user should enter into the system.	Success
2.	Unsuccessful verification due to invalid name.	Registration to a system with invalid name.	Registration should fail and user will face again register page.	Success
3.	Unsuccessful verification due to invalid email.	Registration to a system with invalid email.	Registration should fail and user will face again register page.	Success
4.	Unsuccessful verification due to invalid Password.	Registration to a system with invalid Password.	Registration should fail and user will face again register page.	Success
5.	Unsuccessful verification due to invalid Mobile number	Registration to a system with invalid Mobile number	Registration should fail and user will face again register page.	Success

### 7.1.2. LOGIN PAGE TESTING

Test id	Test case title	Description	Expected outcome	Result
1.	Successful user Verification.	Login to the system should be tried by admin with correct username and password.	Login should be successful and user should enter into the system.	Success
2.	Unsuccessful verification due to wrong password.	Login to a system with the wrong password.	Login should fail and user will face again login page.	Success
3.	Unsuccessful verification due to invalid username.	Login to a system with invalid username.	Login should fail and user will face again login page.	Success

### 7.2. STRUCTURAL TESTING

A great deal can be learnt about the strength and the limitation of the application by examining the manner in which the system breaks. This type of testing has two limitations. It tests failure behavior of the system circumstances may arise through an unexpected combination of events where the node placed on the system exceeds the maximum anticipated load.

The structure of each module was checked at every step. Some structures were firstly wrong, which came to notice at the time of the connectivity.

### **7.3. LEVELS OF TESTING**

The aim of the testing process is to identify all the defects in the website. It is not practical to test the website with respect to each value that the input request data may assume. Testing provides a practical way of reducing defects in the website and increasing the user's confidence in a developed system. Testing consists of subjects the website to a set of test inputs and observing if the program fails to testing behave as expected then conditions under which failure occurs are noted for later debugging and correction. The following things are associated with testing.

Failure is a manifestation of an error. But the mere presence of an error may not necessarily lead to a failure. A test case is the triplet [I, S, O] where I is data input to the system. S is the state of the system at which the data is input. O is the expected output of the system A test suite is the set of all test cases with which a given software product is to be tested.

#### **7.3.1 UNIT TESTING**

In unit testing the entire individual functions and modules were tested independently. By following this strategy all the errors in coding were identified and connected. This method was applied in combination with the white and black box testing techniques to find the errors in each module.

Unit testing is normally considered an adjunct to the coding step. Unit test case design was started after source level code had been developed, reviewed, and verified for correct syntax. A review of design information provides guidance for establishing test cases that were likely to uncover errors in each of the categories above. Each test case was coupled with a set of expected results.

### **7.3.2 INTEGRITY TESTING**

Integrity phases the entire module using the bottom-up approach and tested them. Integrity testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective was to take unit tested modules and build a program structure that has been dictated by design.

The testing strategy has two different approaches namely the top-down approach in which the integration is carried out from the top-level module to the bottom and the bottom-up approach in which the integration is carried out from the low-level module to the top.

The modules were tested using the bottom-up approach by introducing stubs for the top level functions. This test was used to identify the errors in the interfaces, the errors in passing the parameters between the functions and to correct them.

## **8. IMPLEMENTATION**

Project implementation (or project execution) is the phase where visions and plans become reality. This is the logical conclusion, after evaluating, deciding, visioning, planning, applying for funds and finding the financial resources of a project.

### **8.1. IMPLEMENTATION OF PROJECT**

**Step 1:** If user is new then he has to register first with valid Email Id and password (minimum 6 characters).



Email

Password

.....

### Blood Group

**A-**

**Forgot your password?**

**Already registered. Login Me!**



**Step 2:** If user is a donor then click on DonateBlood at Home Page.

The screenshot shows a mobile application interface for blood donation. At the top, there is a teal header bar with a back arrow and the title "BloodDonation". Below this, the main content area is titled "Blood Donation Details". The form contains several fields: "Donor Name" with the value "Shiv Prasad Meena", "Blood Group" with a dropdown menu showing "A-", "State" with a dropdown menu showing "Rajasthan", "City" with a dropdown menu showing "Jaipur", "Address" with the value "Malviya Nagar", and "Phone Number" with the value "8769645421". Below the phone number field, there are two buttons: a grey button labeled "Donated" and a grey button labeled "DONATE". The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

VolTE 4G 78% 11:21

← BloodDonation

## Blood Donation Details

Donor Name

Shiv Prasad Meena

Blood Group

A- ▼

State

Rajasthan ▼

City

Jaipur ▼

Address

Malviya Nagar

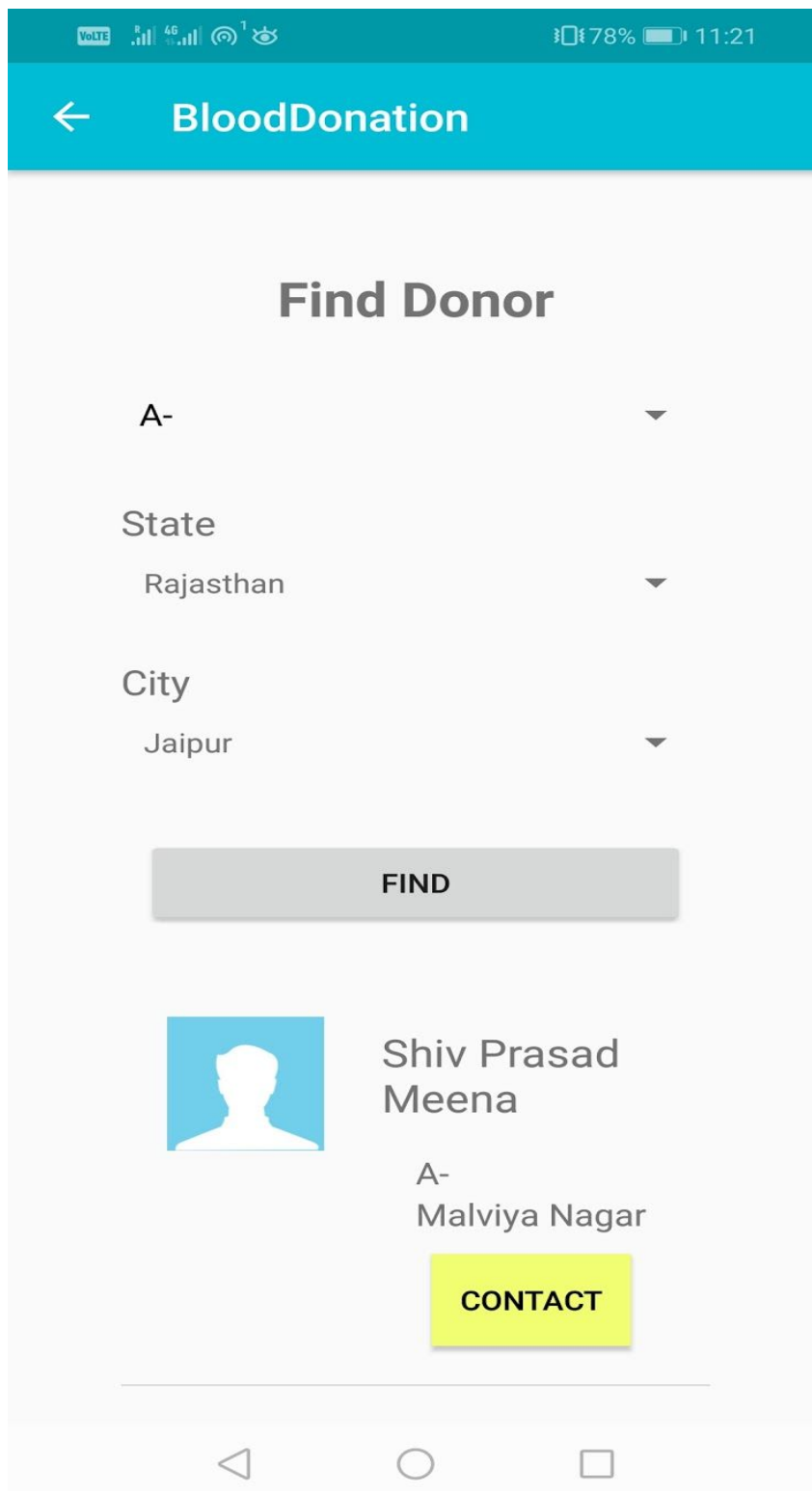
Phone Number

8769645421

Donated

DONATE

**Step 3:** If user is an acceptor.



The screenshot shows a mobile application interface for finding blood donors. At the top, there is a teal header bar with a back arrow and the text "BloodDonation". Below this, the main content area has a light gray background. The title "Find Donor" is centered in a large, bold, dark gray font. Underneath the title, there are three dropdown menus for filtering donors: "A-" for blood type, "Rajasthan" for state, and "Jaipur" for city. Each dropdown has a small downward arrow on its right. Below these filters is a gray button labeled "FIND". At the bottom of the main content area, there is a donor profile card. It features a blue square placeholder for a profile picture on the left. To the right of the picture, the name "Shiv Prasad Meena" is displayed in a bold, dark gray font. Below the name, the text "A- Malviya Nagar" is shown in a smaller, dark gray font. At the bottom right of the profile card is a yellow button labeled "CONTACT". The entire app interface is framed by a white border, and at the very bottom, there is a white bar containing three standard Android navigation icons: a triangle, a circle, and a square.

VoLTE 4G 78% 11:21

← BloodDonation


## Find Donor

A- ▼

State  
Rajasthan ▼

City  
Jaipur ▼

FIND

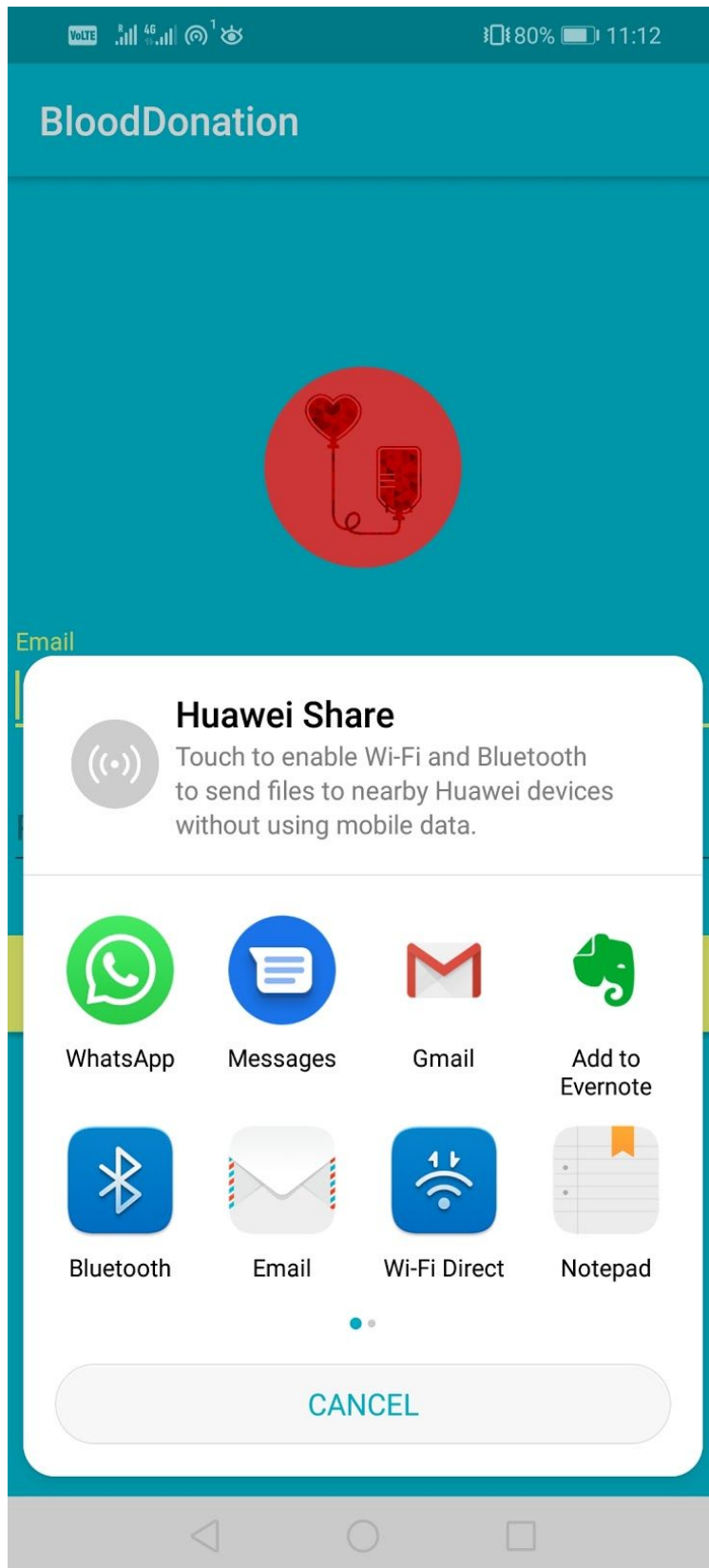
 Shiv Prasad Meena

A-  
Malviya Nagar

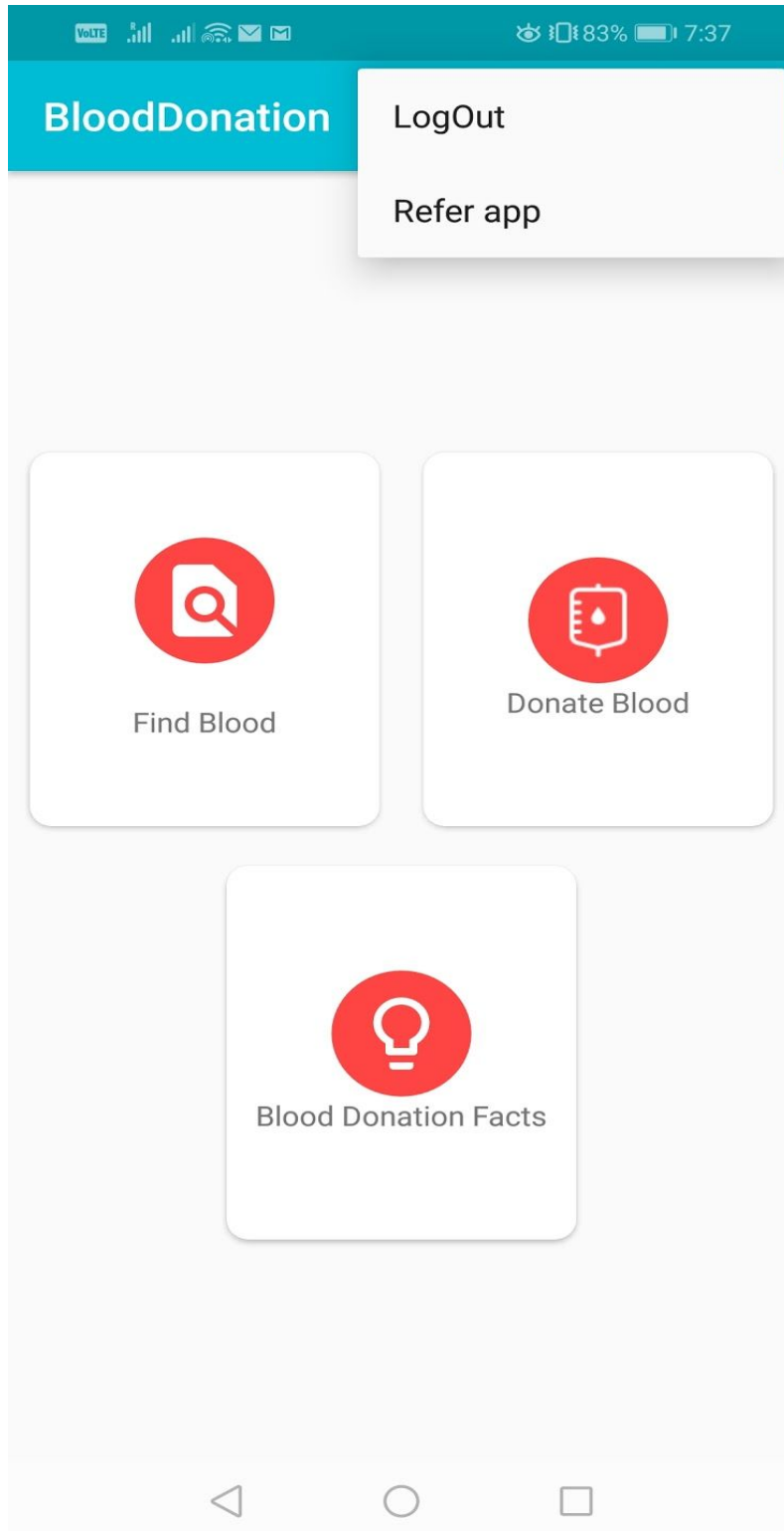
CONTACT



**Step 4:** Refer App.




**Step 5:** Log Out.



a). If password is Invalid, message “password too short, enter minimum 6 characters!” will pop up.


VoLTE 82% 7:41

← BloodDonation



Fullname  
sfhk

Email  
sonum4942@gmail.com

Password  
... | 

Blood Group  
A+

REGISTER


Forgot your password?

Already registered. Login Me!

b). If Email Id already exist, message “The email address is already in use by another account” will pop up.

VolTE 82% 7:42

← BloodDonation



Fullname  
sfhk

Email  
sonum4942@gmail.com

Password  
.....

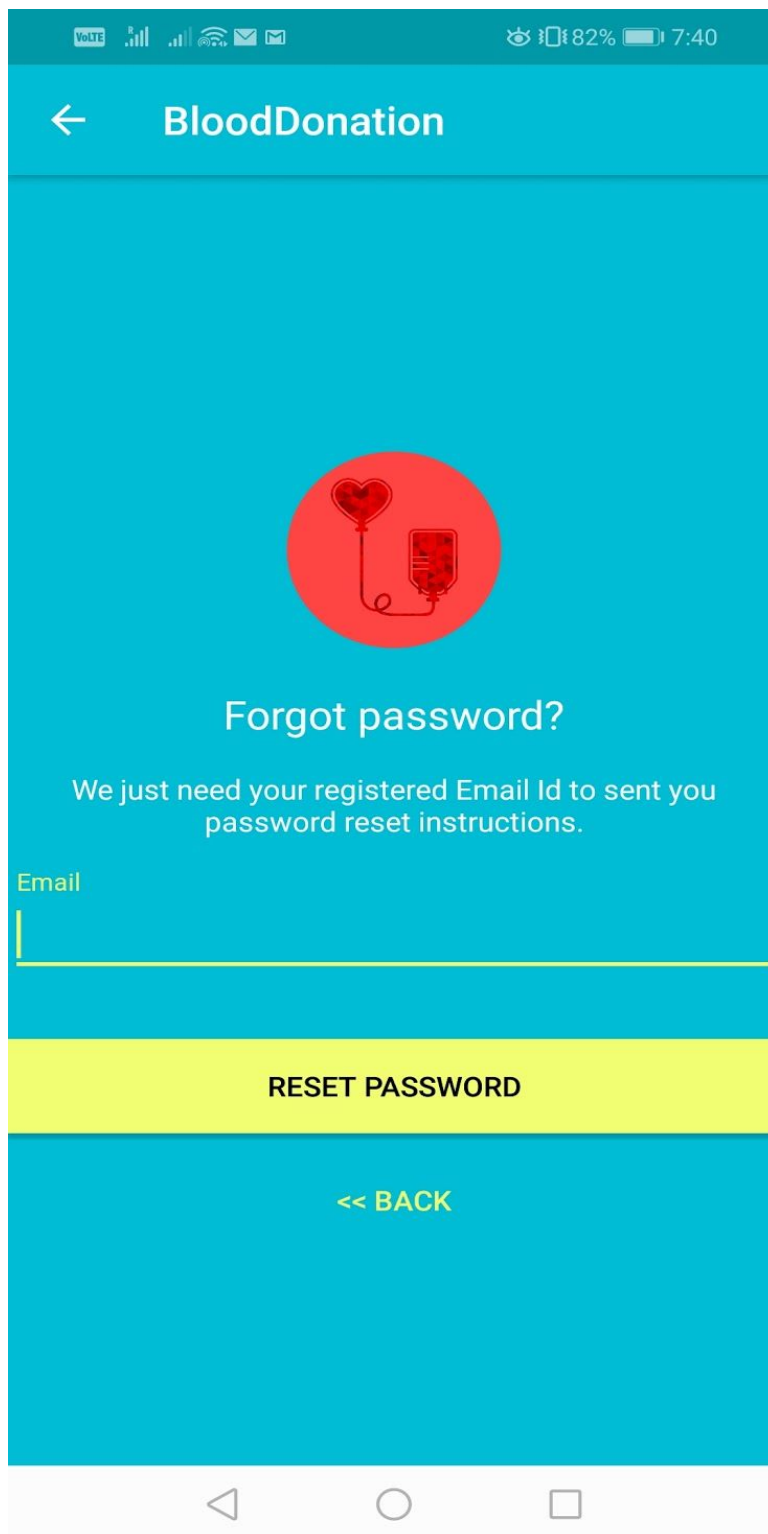
Blood Group  
A+

**REGISTER**

Authentication failed.com.google.firebase.auth.FirebaseAuthUserCollisionException: The email address is already in use by another account.


**Already registered. Login Me!**

c). If you want to change or forgot password then enter your registered email id so that you can get an email of password reset instructions.



VoLTE 82% 7:40

← BloodDonation



Forgot password?

We just need your registered Email Id to sent you password reset instructions.

Email

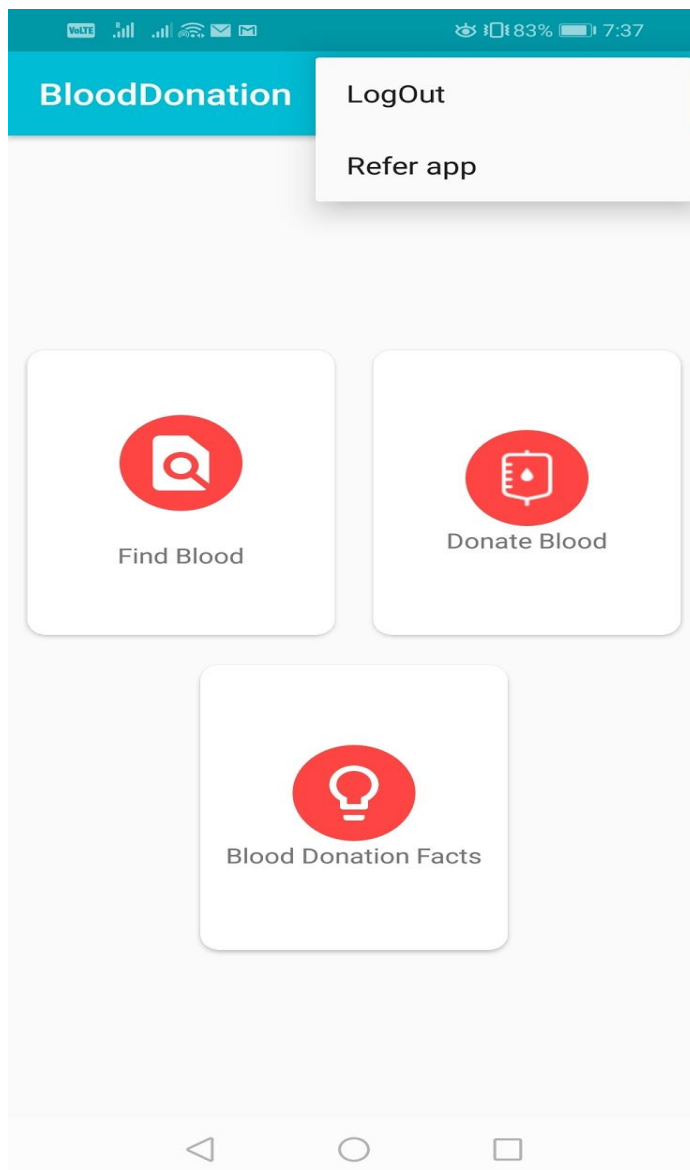
RESET PASSWORD

<< BACK

## 9. USER MANUAL

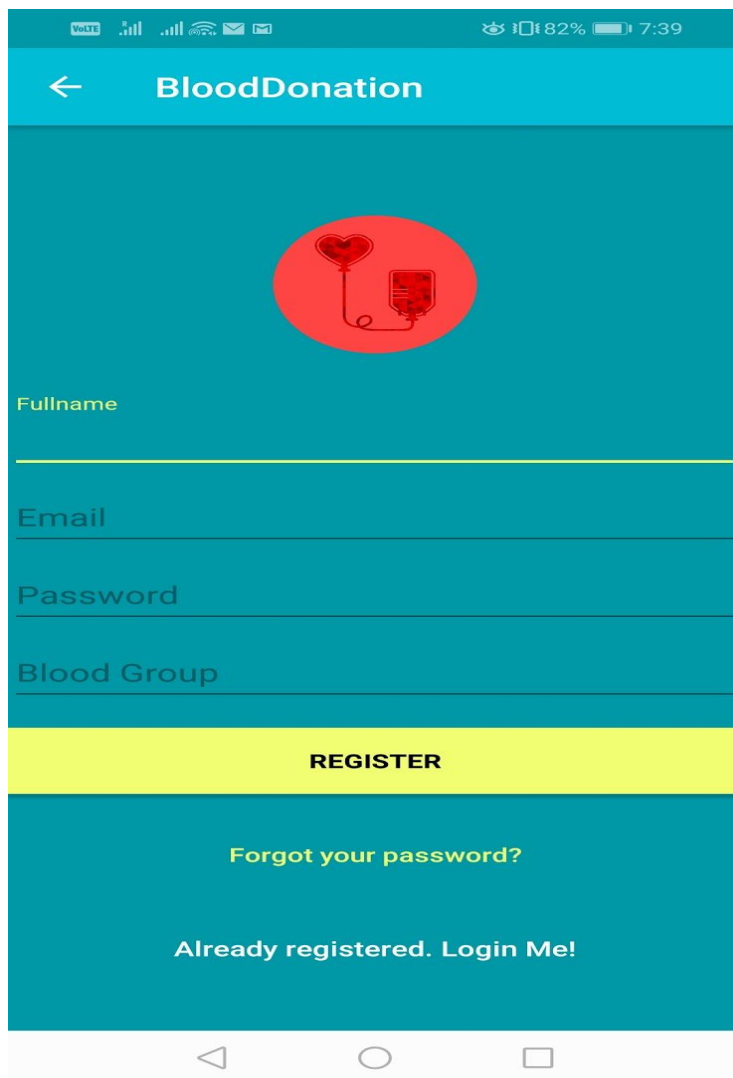
### 9.1. HOME PAGE

This is the home page or the main page of “LifeSource Blood Bank”. This is the main page of a client side. This page defines all about related to project.




## 9.2. REGISTRATION

Registration page includes the information of the donor/Acceptor who want to register. Donor can register the account by clicking new register. He/she can add the account for further enquiry of the blood donation.



The image shows a mobile application interface for a blood donation system. At the top, there is a status bar with various icons and a battery level of 82% at 7:39. Below this is a blue header bar with a back arrow and the text "BloodDonation". The main content area has a teal background. In the center, there is a red circular icon containing a white heart and a blood bag. Below the icon, there are four text input fields labeled "Fullname", "Email", "Password", and "Blood Group". A yellow button labeled "REGISTER" is positioned below the input fields. At the bottom of the form, there are two links: "Forgot your password?" and "Already registered. Login Me!". The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps buttons.

← BloodDonation



Fullname

Email

Password

Blood Group

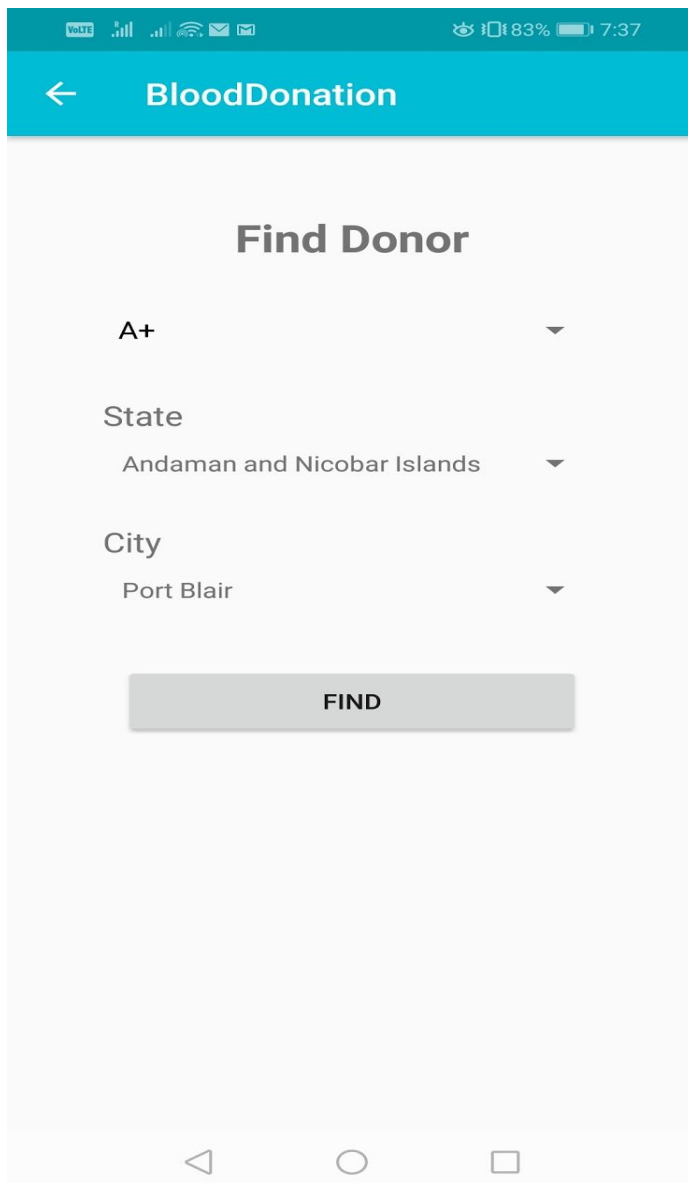
**REGISTER**

[Forgot your password?](#)

[Already registered. Login Me!](#)

### 9.3. REQUEST FOR BLOOD

Request for blood page includes blood groups list, states and cities. Acceptor needs to select blood group he require, his state and city.

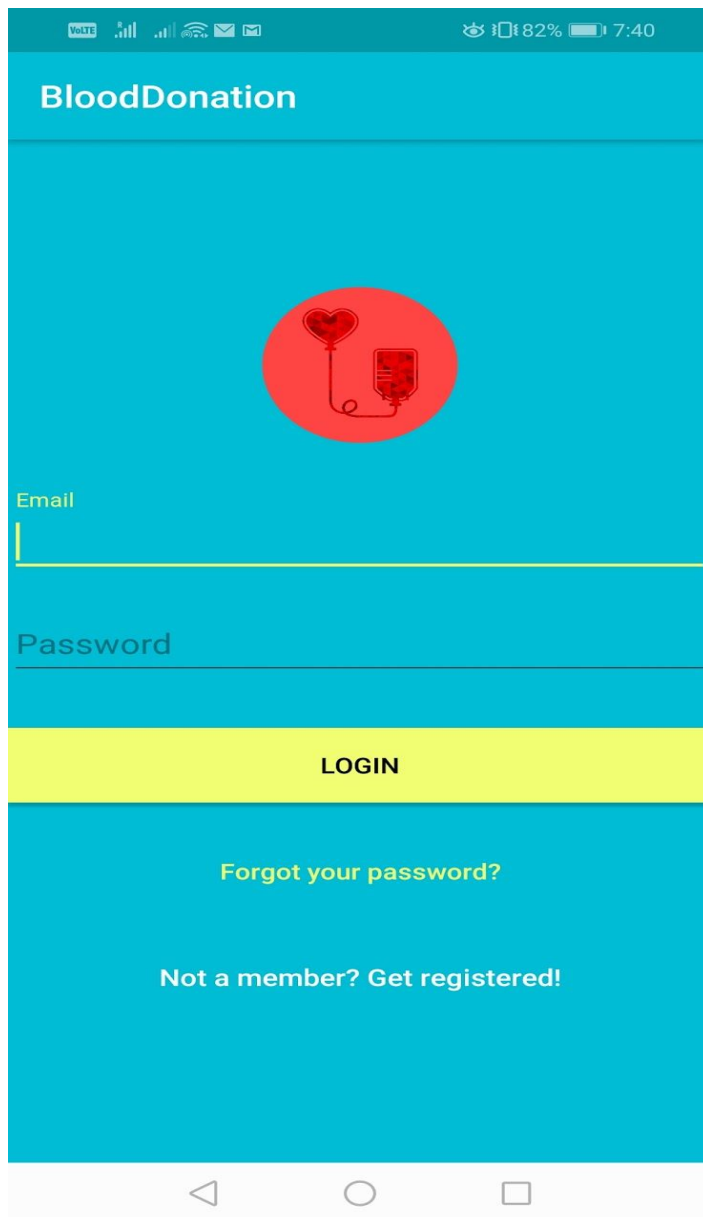


The screenshot shows a mobile application interface for finding blood donors. At the top, there is a teal header bar with a back arrow and the text "BloodDonation". Below this, the title "Find Donor" is centered. The form consists of three dropdown menus: "A+" for blood group, "Andaman and Nicobar Islands" for state, and "Port Blair" for city. Each dropdown has a downward arrow on its right. Below these fields is a grey button labeled "FIND". The status bar at the top shows "83%" battery and "7:37" time. The bottom of the screen shows the Android navigation bar with back, home, and recent apps icons.




## 9.4. DONOR/ACCEPTOR LOGIN

It is menu in which user has to enter his/her username and password, if username and password are correct then it will allow you to move to the next menu otherwise it will show error message and ask user to invalid mail id and password.



**BloodDonation**



Email

Password

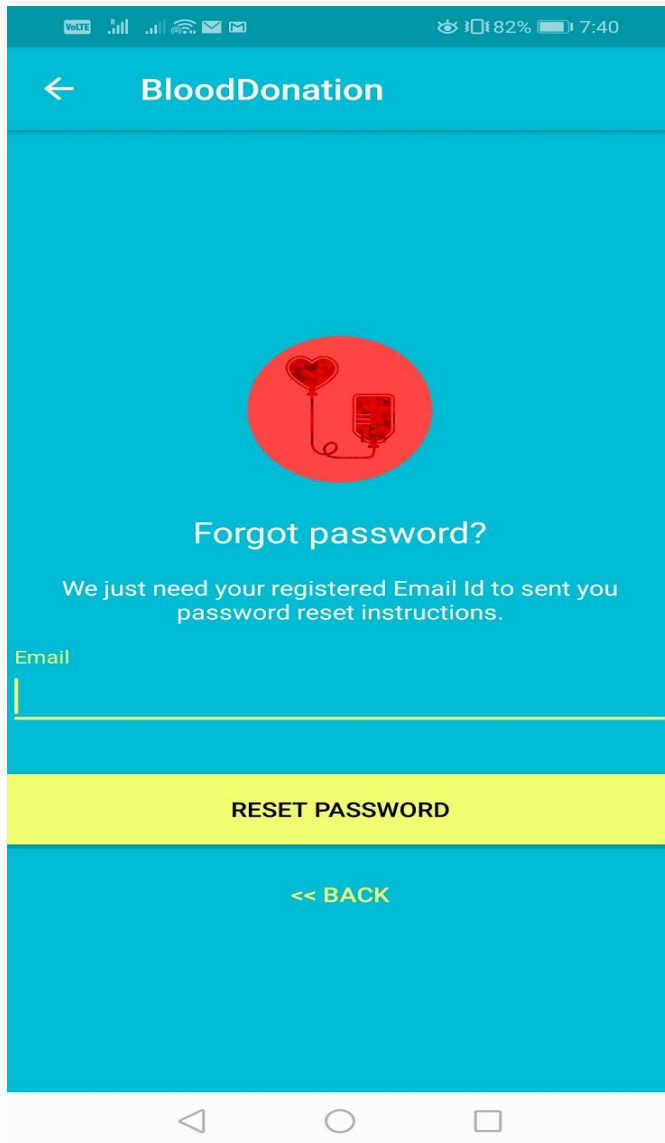
**LOGIN**

[Forgot your password?](#)

[Not a member? Get registered!](#)


## 9.5. CHANGE PASSWORD

When donor /Acceptor forgets his password then he can change his password. For that, he just needs to enter his registered Email Id to get password instructions.



Volte 82% 7:40

← BloodDonation



Forgot password?

We just need your registered Email Id to sent you password reset instructions.

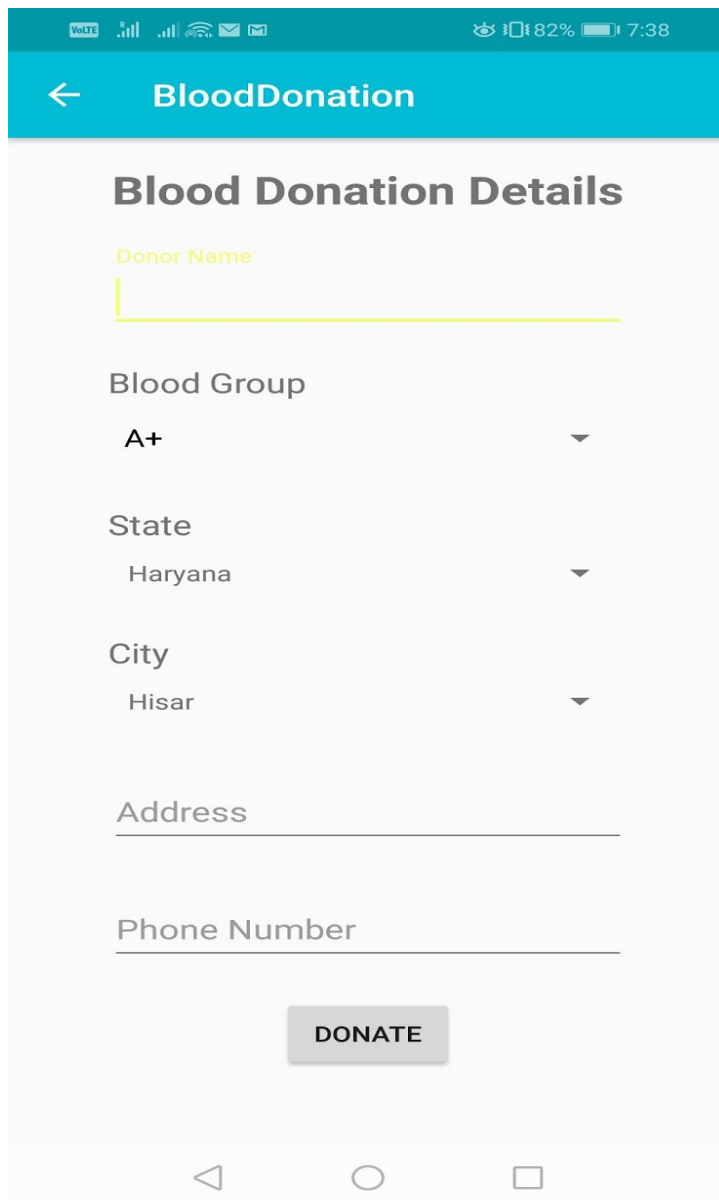
Email

RESET PASSWORD

<< BACK

## 9.6. BLOOD DONATED

Above snapshot describe about donation from donors. This is only used by a donor. To donate blood, donor has to provide some details like his full name, his blood group, his state name, address, and phone number



The screenshot shows a mobile application interface for blood donation. At the top, there is a teal header bar with a back arrow and the text "BloodDonation". Below this, the title "Blood Donation Details" is displayed in bold. The form contains several input fields: "Donor Name" (with a yellow underline), "Blood Group" (a dropdown menu showing "A+" with a downward arrow), "State" (a dropdown menu showing "Haryana" with a downward arrow), "City" (a dropdown menu showing "Hisar" with a downward arrow), "Address" (with a grey underline), and "Phone Number" (with a grey underline). At the bottom of the form is a grey button labeled "DONATE". The status bar at the very top shows "VoLTE", signal strength, Wi-Fi, battery at 82%, and the time 7:38. The bottom of the screen shows the standard Android navigation bar with back, home, and recent apps icons.

## 10. SOURCE CODE

//DonateBlood

```
package com.capstone.blooddonation;
import androidx.appcompat.app.AppCompatActivity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;
import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.JsonArrayRequest;
import com.capstone.blooddonation.Spinner.MySingleton;
import com.capstone.blooddonation.Spinner.State;
import com.capstone.blooddonation.Spinner.StateAdapter;
import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import java.util.ArrayList;
import java.util.List;

public class DonateBlood extends AppCompatActivity {

    private static final String KEY_STATE = "state";
    private static final String KEY_CITIES = "cities";
    private EditText editTextBloodGroup;
    Spinner stateSpinner;
    Spinner citiesSpinner;
    private ProgressDialog pDialog;
    private String cities_url = "http://api.androiddeft.com/cities/cities\_array.json";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_donate_blood);

        stateSpinner = findViewById(R.id.stateSpinner);
        citiesSpinner = findViewById(R.id.citiesSpinner);
    }
}
```

```

editTextBloodGroup = findViewById(R.id.editTextBloodGroup);

displayLoader();
loadStateCityDetails();

Button submitButton = findViewById(R.id.buttonSubmit);

//Display state and city name when submit button is pressed
submitButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String bloodGroup = editTextBloodGroup.getText().toString();
        State state = (State) stateSpinner.getSelectedItem();
        String city = citiesSpinner.getSelectedItem().toString();
        Toast.makeText(getApplicationContext(), "Selected State: " +
state.getStateName()
        + " City: " + city, Toast.LENGTH_SHORT).show();

    }
});
}

private void displayLoader() {
    pDialog = new ProgressDialog(DonateBlood.this);
    pDialog.setMessage("Loading Data.. Please wait...");
    pDialog.setIndeterminate(false);
    pDialog.setCancelable(false);
    pDialog.show();

}

private void loadStateCityDetails() {
    final List<State> statesList = new ArrayList<>();
    final List<String> states = new ArrayList<>();
    JsonRequest jsArrayRequest = new JsonRequest
        (Request.Method.GET, cities_url, null, new
Response.Listener<JSONArray>() {
        @Override
        public void onResponse(JSONArray responseArray) {
            pDialog.dismiss();
            try {
                //Parse the JSON response array by iterating over it
                for (int i = 0; i < responseArray.length(); i++) {
                    JSONObject response = responseArray.getJSONObject(i);
                    String state = response.getString(KEY_STATE);

```

```

        JSONArray cities = response.getJSONArray(KEY_CITIES);
        List<String> citiesList = new ArrayList<>();
        for (int j = 0; j < cities.length(); j++) {
            citiesList.add(cities.getString(j));
        }
        statesList.add(new State(state, citiesList));
        states.add(state);

    }
    final StateAdapter stateAdapter = new
StateAdapter(DonateBlood.this,
        R.layout.state_list, R.id.spinnerText, statesList);
    stateSpinner.setAdapter(stateAdapter);

    stateSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {

            //Populate City list to the second spinner when
            // a state is chosen from the first spinner
            State cityDetails = stateAdapter.getItem(position);
            List<String> cityList = cityDetails.getCities();
            ArrayAdapter citiesAdapter = new ArrayAdapter<>(DonateBlood.this,
                R.layout.city_list, R.id.citySpinnerText, cityList);
            citiesSpinner.setAdapter(citiesAdapter);
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {

        }
    });

} catch (JSONException e) {
    e.printStackTrace();
}
}, new Response.ErrorListener() {

    @Override
    public void onErrorResponse(VolleyError error) {
        pDialog.dismiss();
    }
}

```

```

        //Display error message whenever an error occurs
        Toast.makeText(getApplicationContext(),
            error.getMessage(), Toast.LENGTH_SHORT).show();

    }
});

// Access the RequestQueue through your singleton class.
MySingleton.getInstance(this).addToRequestQueue(jsArrayRequest);
}
}

```

//HomeActivity

```

package com.capstone.blooddonation;

import androidx.appcompat.app.AppCompatActivity;
import androidx.cardview.widget.CardView;

import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class HomeActivity extends AppCompatActivity {

    CardView btnDonateBlood,btnGetBlood,btnBloodFact;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_home);

        init();
    }
    public void init(){
        btnGetBlood=findViewById(R.id.btnGetBlood);
        btnDonateBlood=findViewById(R.id.btnDonateBlood);
        btnBloodFact=findViewById(R.id.btnBloodFact);

        btnGetBlood.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {

```

```

        Intent intent = new Intent(HomeActivity.this,FindDonor.class);
        startActivity(intent);
    }
});

btnDonateBlood.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(HomeActivity.this,DonateBlood.class);
        startActivity(intent);
    }
});
}
}
}

```

//LoginActivity

```

package com.capstone.blooddonation;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class LoginActivity extends AppCompatActivity {

    private EditText inputEmail, inputPassword;
    private FirebaseAuth auth;
    private ProgressBar progressBar;

```



```

private Button btnSignup, btnLogin, btnReset;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_login);

    // auth = FirebaseAuth.getInstance();

    /* if (auth.getCurrentUser() != null) {
        //startActivity(new Intent(LoginActivity.this, MainActivity.class));
        finish();
    }*/

    // set the view now
    setContentView(R.layout.activity_login);

    inputEmail = (EditText) findViewById(R.id.email);
    inputPassword = (EditText) findViewById(R.id.password);
    progressBar = (ProgressBar) findViewById(R.id.progressBar);
    btnSignup = (Button) findViewById(R.id.btn_signup);
    btnLogin = (Button) findViewById(R.id.btn_login);
    btnReset = (Button) findViewById(R.id.btn_reset_password);

    //Get Firebase auth instance
    auth = FirebaseAuth.getInstance();

    btnSignup.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            startActivity(new Intent(LoginActivity.this, SignupActivity.class));
            finish();
        }
    });

    btnReset.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //startActivity(new Intent(LoginActivity.this, ResetPasswordActivity.class));
        }
    });

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

```

```

public void onClick(View v) {
    String email = inputEmail.getText().toString();
    final String password = inputPassword.getText().toString();

    if (TextUtils.isEmpty(email)) {
        Toast.makeText(getApplicationContext(), "Enter email address!",
Toast.LENGTH_SHORT).show();
        return;
    }

    if (TextUtils.isEmpty(password)) {
        Toast.makeText(getApplicationContext(), "Enter password!",
Toast.LENGTH_SHORT).show();
        return;
    }

    progressBar.setVisibility(View.VISIBLE);

    //authenticate user
    auth.signInWithEmailAndPassword(email, password)
        .addOnCompleteListener(LoginActivity.this, new
OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            // If sign in fails, display a message to the user. If sign in succeeds
            // the auth state listener will be notified and logic to handle the
            // signed in user can be handled in the listener.
            progressBar.setVisibility(View.GONE);
            if (!task.isSuccessful()) {
                // there was an error
                if (password.length() < 6) {

inputPassword.setError(getString(R.string.minimum_password));
                } else {
                    Toast.makeText(LoginActivity.this,
getString(R.string.auth_failed), Toast.LENGTH_LONG).show();
                }
            } else {
                // Intent intent = new Intent(LoginActivity.this, MainActivity.class);
                //startActivity(intent);
                finish();
            }
        }
    });
}

```

```
    });  
    }  
}
```

```
//MapsActivity
```

```
package com.capstone.blooddonation;
```

```
import androidx.annotation.NonNull;  
import androidx.appcompat.app.AlertDialog;  
import androidx.core.app.ActivityCompat;  
import androidx.core.content.ContextCompat;  
import androidx.fragment.app.FragmentActivity;
```

```
import android.Manifest;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.IntentSender;  
import android.content.pm.PackageManager;  
import android.location.Address;  
import android.location.Criteria;  
import android.location.Geocoder;  
import android.location.Location;  
import android.location.LocationManager;  
import android.os.Build;  
import android.os.Bundle;  
import android.util.Log;  
import android.view.View;  
import android.widget.EditText;  
import android.widget.Toast;
```

```
import com.google.android.gms.common.ConnectionResult;  
import com.google.android.gms.common.api.GoogleApiClient;  
import com.google.android.gms.common.api.PendingResult;  
import com.google.android.gms.common.api.ResultCallback;  
import com.google.android.gms.common.api.Status;  
import com.google.android.gms.location.FusedLocationProviderClient;  
import com.google.android.gms.location.LocationListener;  
import com.google.android.gms.location.LocationRequest;  
import com.google.android.gms.location.LocationServices;  
import com.google.android.gms.location.LocationSettingsRequest;  
import com.google.android.gms.location.LocationSettingsResult;
```

```

import com.google.android.gms.location.LocationSettingsStatusCodes;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.tasks.OnSuccessListener;

import java.io.IOException;
import java.util.List;
import java.util.Locale;

import static android.Manifest.permission_group.LOCATION;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
    LocationListener, GoogleApiClient.ConnectionCallbacks,
    GoogleApiClient.OnConnectionFailedListener {

    private GoogleMap mMap;
    Location mLastLocation;
    Marker mCurrLocationMarker;
    GoogleApiClient mGoogleApiClient;
    LocationRequest mLocationRequest;
    LocationManager locationManager;
    String provider;
    EditText locationSearch;
    PendingResult<LocationSettingsResult> result;
    static final Integer GPS_SETTINGS = 0x7;
    FusedLocationProviderClient fusedLocationProviderClient;
    private Location lastLocation;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be
        used.
        SupportMapFragment mapFragment = (SupportMapFragment)
        getSupportFragmentManager()
            .findFragmentById(R.id.map);
        init();
        mapFragment.getMapAsync(this);
    }

```

```

    }
    public void init(){
        locationSearch = findViewById(R.id.editTextSearch);
        locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
        fusedLocationProviderClient=
LocationServices.getFusedLocationProviderClient(this);
        provider = locationManager.getBestProvider(new Criteria(), false);
    }

    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

}

    protected synchronized void buildGoogleApiClient() {
        mGoogleApiClient = new GoogleApiClient.Builder(this)
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(LocationServices.API).build();
        mGoogleApiClient.connect();
    }

    @Override
    public void onConnected(Bundle bundle) {

        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);

        mLocationRequest.setPriority(LocationRequest.PRIORITY_BALANCED_POWER_ACCURACY);
        if (ContextCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_FINE_LOCATION)
            == PackageManager.PERMISSION_GRANTED) {
            LocationServices.FusedLocationApi.requestLocationUpdates(mGoogleApiClient,
            mLocationRequest, this);
        }

    }

    @Override
    public void onConnectionSuspended(int i) {

```

```

    }

    @Override
    public void onLocationChanged(Location location) {

        mLastLocation = location;
        if (mCurrLocationMarker != null) {
            mCurrLocationMarker.remove();
        }
        //Place current location marker
        LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(latLng);
        markerOptions.title("Current Position");

        markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_
_GREEN));
        mCurrLocationMarker = mMap.addMarker(markerOptions);

        //move map camera
        mMap.moveCamera(CameraUpdateFactory.newLatLng(latLng));
        mMap.animateCamera(CameraUpdateFactory.zoomTo(11));

        //stop location updates
        if (mGoogleApiClient != null) {
            LocationServices.FusedLocationApi.removeLocationUpdates(mGoogleApiClient,
this);
        }

    }

    @Override
    public void onConnectionFailed(ConnectionResult connectionResult) {

    }

    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In
this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted to
install

```

\* it inside the SupportMapFragment. This method will only be triggered once the user has

\* installed Google Play services and returned to the app.

\*/

// Add a marker in Sydney and move the camera

public void searchLocation(View view) {

String location = locationSearch.getText().toString();

List<Address> addressList = null;

if (location != null || !location.equals("")) {

Geocoder geocoder = new Geocoder(this);

try {

addressList = geocoder.getFromLocationName(location, 1);

} catch (IOException e) {

e.printStackTrace();

}

Address address = addressList.get(0);

LatLng latLng = new LatLng(address.getLatitude(), address.getLongitude());

mMap.addMarker(new MarkerOptions().position(latLng).title(location));

mMap.animateCamera(CameraUpdateFactory.newLatLng(latLng));

Toast.makeText(getApplicationContext(),address.getLatitude()+"

"+address.getLongitude(),Toast.LENGTH\_LONG).show();

}

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

switch (requestCode) {

case 99:

if (ActivityCompat.checkSelfPermission(this, permissions[0]) == PackageManager.PERMISSION\_GRANTED) {

getLocation();

}

else{

Toast.makeText(this, "Permission denied", Toast.LENGTH\_SHORT).show();

}

return;

}

}

```

        void getLocation(){
            if(ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)!=
PackageManager.PERMISSION_GRANTED){
                ActivityCompat.requestPermissions(this,new String[]
{Manifest.permission.ACCESS_FINE_LOCATION},99);
            }
            else{
                Log.d("TAG","Get Location permission granted");

                fusedLocationProviderClient.getLastLocation().addOnSuccessListener(new
OnSuccessListener<Location>() {
                    @Override
                    public void onSuccess(Location location) {
                        if(location!=null){
                            lastLocation=location;
                            double latitude=lastLocation.getLatitude();
                            double longitude=lastLocation.getLongitude();

                            // tvLatitude.setText(""+latitude);
                            //tvLongitude.setText(""+longitude);

                            Geocoder geocoder=new Geocoder(MapsActivity.this, Locale.getDefault());

                            try {
                                List<Address>
locationList=geocoder.getFromLocation(latitude,longitude,1);

                                if(locationList.size(>0){
                                    Address address=locationList.get(0);

                                    //tvPhysicalAddress.setText("Address is: "+address);
                                }
                            }catch (IOException e){
                                e.printStackTrace();
                            }
                        }
                    }
                });
            }
        }
    }
}

```



```
//MySingleton
```

```
package com.capstone.blooddonation.Spinner;
```

```
import android.content.Context;
```

```
import com.android.volley.Request;
```

```
import com.android.volley.RequestQueue;
```

```
import com.android.volley.toolbox.Volley;
```

```
public class MySingleton {
```

```
    private static MySingleton mInstance;
```

```
    private RequestQueue mRequestQueue;
```

```
    private static Context mCtx;
```

```
    private MySingleton(Context context) {
```

```
        mCtx = context;
```

```
        mRequestQueue = getRequestQueue();
```

```
    }
```

```
    public static synchronized MySingleton getInstance(Context context) {
```

```
        if (mInstance == null) {
```

```
            mInstance = new MySingleton(context);
```

```
        }
```

```
        return mInstance;
```

```
    }
```

```
    private RequestQueue getRequestQueue() {
```

```
        if (mRequestQueue == null) {
```

```
            // getApplicationContext() is key, it keeps you from leaking the
```

```
            // Activity or BroadcastReceiver if someone passes one in.
```

```
            mRequestQueue = Volley.newRequestQueue(mCtx.getApplicationContext());
```

```
        }
```

```
        return mRequestQueue;
```

```
    }
```

```
    public <T> void addToRequestQueue(Request<T> req) {
```

```
        getRequestQueue().add(req);
```

```
    }
```

```
}
```

```
//ResetPasswordActivity
```

```
package com.capstone.blooddonation;
```

```
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.Toast;
```

```
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.FirebaseAuth;
```

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
```

```
public class ResetPasswordActivity extends AppCompatActivity {
```

```
    private EditText inputEmail;
    private Button btnReset, btnBack;
    private FirebaseAuth auth;
    private ProgressBar progressBar;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_reset_password);
```

```
        inputEmail = findViewById(R.id.email);
        btnReset = findViewById(R.id.btn_reset_password);
        btnBack = findViewById(R.id.btn_back);
        progressBar = findViewById(R.id.progressBar);
```

```
        auth = FirebaseAuth.getInstance();
```

```
        btnBack.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
```

```
            public void onClick(View v) {
                finish();
            }
        });
```



```

import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ProgressBar;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class SignupActivity extends AppCompatActivity {

    private EditText inputEmail, inputPassword, inputFullName, inputBloodGroup;
    private Button btnSignIn, btnSignUp, btnResetPassword;
    private ProgressBar progressBar;
    private FirebaseAuth auth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_signup);

        auth = FirebaseAuth.getInstance();

        btnSignIn = findViewById(R.id.sign_in_button);
        btnSignUp = findViewById(R.id.sign_up_button);
        inputEmail = findViewById(R.id.editTextEmail);
        inputPassword = findViewById(R.id.editTextPassword);
        inputFullName = findViewById(R.id.editTextName);
        inputBloodGroup = findViewById(R.id.editTextBloodGroup);
        progressBar = findViewById(R.id.progressBar);
        btnResetPassword = findViewById(R.id.btn_reset_password);

        btnResetPassword.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // Toast.makeText(SignupActivity.this, "Signed Up",
                Toast.LENGTH_SHORT).show();
                //startActivity(new Intent(SignupActivity.this, ResetPasswordActivity.class));
            }
        });
    }
}

```

```

btnSignIn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(SignupActivity.this, "Prssed",
Toast.LENGTH_SHORT).show();
        Intent intent=new Intent(getApplicationContext(), LoginActivity.class);
        startActivity(intent);
        //finish();
    }
});

btnSignUp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String name = inputFullName.getText().toString().trim();
        String email = inputEmail.getText().toString().trim();
        String password = inputPassword.getText().toString().trim();
        String bloodGroup = inputBloodGroup.getText().toString().trim();

        if(TextUtils.isEmpty(name)){
            inputFullName.setError("Enter Full Name!");
            return;
        }
        if (TextUtils.isEmpty(email)) {
            inputEmail.setError("Enter Email Address!");
            return;
        }

        if (TextUtils.isEmpty(password)) {
            inputPassword.setError("Enter Password!");
            return;
        }

        if (password.length() < 6) {
            inputPassword.setError("Password too short, enter minimum 6
characters!");
            return;
        }

        progressBar.setVisibility(View.VISIBLE);
        auth.createUserWithEmailAndPassword(email, password)
            .addOnCompleteListener(SignupActivity.this, new
OnCompleteListener<AuthResult>() {

```

```

        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            Toast.makeText(SignupActivity.this,
                "createUserWithEmail:onComplete:" + task.isSuccessful(),
                Toast.LENGTH_SHORT).show();
            progressBar.setVisibility(View.GONE);
            if (!task.isSuccessful()) {
                Toast.makeText(SignupActivity.this, "Authentication failed." +
                    task.getException(),
                        Toast.LENGTH_LONG).show();
            } else {
                Toast.makeText(SignupActivity.this, "Signed Up",
                    Toast.LENGTH_SHORT).show();
                startActivity(new Intent(SignupActivity.this, LoginActivity.class));
                finish();
            }
        }
    });

}
});
}

@Override
protected void onResume() {
    super.onResume();
    progressBar.setVisibility(View.GONE);
}
}

```

//SplashActivity

```
package com.capstone.blooddonation;
```

```
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.content.Intent;
```

```
import android.os.Bundle;
```

```
import android.os.Handler;
```

```
public class SplashActivity extends AppCompatActivity {
```

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_splash);

    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            //SharedPreferences myPreferences =
PreferenceManager.getDefaultSharedPreferences(SplashActivity.this);
            Intent intent=new Intent(SplashActivity.this,LoginActivity.class);
            startActivity(intent);
            finish();
        }
    },1000);
}
}

```

//State

```
package com.capstone.blooddonation.Spinner;
```

```
import java.util.List;
```

```

public class State {
    private String stateName;
    private List<String> cities;

    public State(String stateName, List<String> cities) {
        this.stateName = stateName;
        this.cities = cities;
    }

    public String getStateName() {
        return stateName;
    }

    public List<String> getCities() {
        return cities;
    }
}

```

```
}
```

```
//StateAdapter
```

```
package com.capstone.blooddonation.Spinner;
```

```
import android.content.Context;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
import android.widget.AdapterView;  
import android.widget.TextView;
```

```
import com.capstone.blooddonation.R;
```

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

```
import androidx.annotation.NonNull;  
import androidx.annotation.Nullable;
```

```
public class StateAdapter extends ArrayAdapter<State> {  
    private List<State> stateList = new ArrayList<>();  
  
    public StateAdapter(@NonNull Context context, int resource, int spinnerText,  
@NonNull List<State> stateList) {  
        super(context, resource, spinnerText, stateList);  
        this.stateList = stateList;  
    }  
  
    @Override  
    public State getItem(int position) {  
        return stateList.get(position);  
    }  
  
    @NonNull  
    @Override  
    public View getView(int position, @Nullable View convertView, @NonNull  
ViewGroup parent) {  
        return initView(position);  
    }  
}
```



```

@Override
public View getDropDownView(int position, @Nullable View convertView,
@NonNull ViewGroup parent) {
    return initView(position);
}

/**
 * Gets the state object by calling getItem and
 * Sets the state name to the drop-down TextView.
 *
 * @param position the position of the item selected
 * @return returns the updated View
 */
private View initView(int position) {
    State state = getItem(position);
    LayoutInflater inflater = (LayoutInflater)
getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View v = inflater.inflate(R.layout.state_list, null);
    TextView textView = v.findViewById(R.id.spinnerText);
    textView.setText(state.getStateName());
    return v;
}
}

```

## 11. BIBLIOGRAPHY

