CAPSTONE PROJECT REPORT

(Project Term July-November 2019)

BLOOD COUNT APPLICATION

Submitted by

Gojur Ravikanth	**Registration Number: 11616140**
Tokala Rajesh	**Registration Number: 11601616**
Kannaboina Vinod Yadav	**Registration Number: 11608060**

Project Group Number : CSERGC0062

Course Code : CSE 439

School of Computer Science & Engineering

Under the Guidance of

Mr. Kiran Kumar Kaki

PAC FORM

TOPIC APPROVAL PERFORMA

School of Computer Science and Engineering (SCSE)

Program : P132::B.Tech. (Computer Science & Engineering)

COURSE CODE : CSE439 **REGULAR/BACKLOG :** Regular **GROUP NUMBER :** CSERGC0062

Supervisor Name : Kiran Kumar Kaki **UID :** 12307 **Designation :** Assistant Professor
Qualification : _________________________ **Research Experience :** _________________________

SR.NO.	NAME OF STUDENT	REGISTRATION NO	BATCH	SECTION	CONTACT NUMBER
1	Gojur Ravikanth	11616140	2016	K1612	8500059020
2	MD Viqhar	11608140	2016	K1601	9491081003
3	Kannaboina Vinod Yadav	11608060	2016	K1629	7659861344
4	Tokala Rajesh	11601616	2016	K1628	7287919261

SPECIALIZATION AREA : Program Methodology and Design **Supervisor Signature:** ____________________

PROPOSED TOPIC : Blood Count(React Native Based App)

Qualitative Assessment of Proposed Topic by PAC		
Sr.No.	**Parameter**	**Rating (out of 10)**
1	Project Novelty: Potential of the project to create new knowledge	7.18
2	Project Feasibility: Project can be timely carried out in-house with low-cost and available resources in the University by the students.	7.18
3	Project Academic Inputs: Project topic is relevant and makes extensive use of academic inputs in UG program and serves as a culminating effort for core study area of the degree program.	6.36
4	Project Supervision: Project supervisor’s is technically competent to guide students, resolve any issues, and impart necessary skills.	8.18
5	Social Applicability: Project work intends to solve a practical problem.	7.00
6	Future Scope: Project has potential to become basis of future research work, publication or patent.	7.18

PAC Committee Members		
PAC Member (HOD/Chairperson) Name: Gaurav Pushkarna	UID: 11057	Recommended (Y/N): Yes
PAC Member (Allied) Name: Pradeep Kumar	UID: 16473	Recommended (Y/N): Yes
PAC Member 3 Name: Dr. Vijay Kumar Garg	UID: 14085	Recommended (Y/N): Yes

Final Topic Approved by PAC: **Blood Count(React Native Based App)**

Overall Remarks: Approved

PAC CHAIRPERSON Name: 11024::Amandeep Nagpal **Approval Date:** 29 Apr 2019

DECLARATION

We hereby declare that the project work entitled (“ Blood Count Application ”) 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 Mr. Kiran Kumar Kaki during January to April 2019. All the information furnished in this capstone project report is based on our own intensive work and is genuine.

Project Group Number: CSERGC0062

Name of Student 1: Gojur Ravikanth
Registration Number: 11616140

Name of Student 2: Tokala Rajesh
Registration Number: 11601616

Name of Student 3: Kannaboina Vinod Yadav
Registration Number: 11608060

(Signature of Student 1)
Date:

(Signature of Student 2)
Date:

(Signature of Student 3)
Date:

CERTIFICATE

This is to certify that the declaration statement made by this group of students is correct to the best of my knowledge and belief. They have completed this Capstone Project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of the work has ever been submitted for any other degree at any University. The Capstone Project is fit for the submission and partial fulfilment of the conditions for the award of B. Tech degree in Computer Science and Engineering from Lovely Professional University, Phagwara.

Signature and Name of the Mentor:

Mr.Kiran Kumar Kaki
Designation: Assistant Professor
School of Computer Science and Engineering
Lovely Professional University,
Phagwara, Punjab.

Date :

ACKNOWLEDGEMENT

We are truly thankful to our mentor who guided us throughout the project and motivated us to learn about the technologies which are used in this project. A Computer Science student mainly focuses on programming applications of concept learned projects like this one provided us with an ideal opportunity to do the same. We acknowledge the valuable inputs provided by out mentor Mr. Kiran Kumar Kaki.

Later, we are thankful to all our faculties of Lovely Professional University for timely cooperation and support extended to us all the way.

TABLE OF CONTENTS

1.INTRODUCTION:

REACT NATIVE is a JavaScript framework used for developing a real, native mobile applications for iOS and Android. It uses JavaScript to design an application. It is like React, which uses native component rather than using web components as building blocks. React Native is based on React, JavaScript library of Facebook, and XML-esque markup (JSX) for developing the user interface. It targets the mobile platform rather than the browser. React native apps will run on a mobile device. React Native apps are the real native app, the JavaScript code stays as JavaScript, and they run in some extra thread by the compiled app. The user interface and everything are compiled to native code.

React Native is different from those types of frameworks. While a framework like PhoneGap works by wrapping web content in a WebView resulting in UI elements that don’t quite have a native feel to them, React native uses JavaScript components backed by native iOS or Android components so the app you build is fully native.

The BLOOD COUNT is both Android and iOS based project built using REACT NATIVE. This application is used in urgency or emergency, in which the mobile user can get all the information regarding the availability of blood, blood donation. The persons who are willing to donate the blood are allowed to register on this application using User Id, Password, blood group. The people who are in need of blood of various group can get into our application and find the availability of blood in near by blood banks. The acceptor can get access to the details of donor and he can contact the donor with the help of mobile number provided during the registration.

This Application provides administration access to head(who maintains the Blood bank) owing the Blood bank. The administrator regularly updates the information regarding the availability of blood. And he is provided with Admin Id and password, provided with access to various functionalities apart from donors, receivers.

This Application has numerous functionalities like registration, Login, Blood donation, Blood receiving , Generation of certificate, notice near Blood banks.

2. PROFILE OF THE PROBLEM, RATIONALE/SCOPE OF THE STUDY (PROBLEM STATEMENT):

In the present situation searching for blood donors is done through blood bank centres or through toll free numbers. So far it is a time consuming process. Because it involves lots of manual work. Lots of time is wasted for going to blood bank if the particular type of blood is not available in case of urgency or emergency it leads to a bad situation. In most of the cases user has to wait in the queue. The project is all about making the blood services available to the user.

2.1. OBJECTIVE:

The goals and objectives of the BLOOD COUNT Application are as follows:

1.To allow probable recipients to make search by current location and get nearby blood banks.

2.To provide and efficient donor and blood stock management functions to the blood bank by recording the donor and blood details.

3.To provide synchronized and centralized donor and blood stock database.

4. To provide immediate storage and retrieval of data and information.

5. The use of e-certificate for future use.

6. To help the people regarding blood during urgency and emergency situations.

2.2. DESCRIPTION:

The system we developed is BLOOD COUNT. This is mobile application based system that is to be used by blood banks and blood centres to update the information regarding donors, receivers and availability of blood. The system keeps the record of all the donors, recipients. This system also has the ability to keep track of the donors and blood stock in the blood bank. It also generates a e-Certificate for the future use of donor when he requests the blood. This project intends to computerize the blood availability.

There are some applications related to our project but they are having some drawbacks. The certificate generated, when the donor donates the blood is of no use in other systems. But in our application the certificate can be used to get free blood from the blood bank in the future. The receiver who is in need of blood can contact a donor through our application by verifying the contact details of the donor.

2.3. SCOPE:

This application provides the required information in less time and also helps out in quicker decision making. The scope of this project is that in a very short span it provides user with many facilities. It provides active management of blood, nearby blood banks and online donors. The main purpose of this project is to interconnect all the blood banks, donors on to a single network, store various data and information regarding blood and donors.

3. EXISTING SYSTEM:

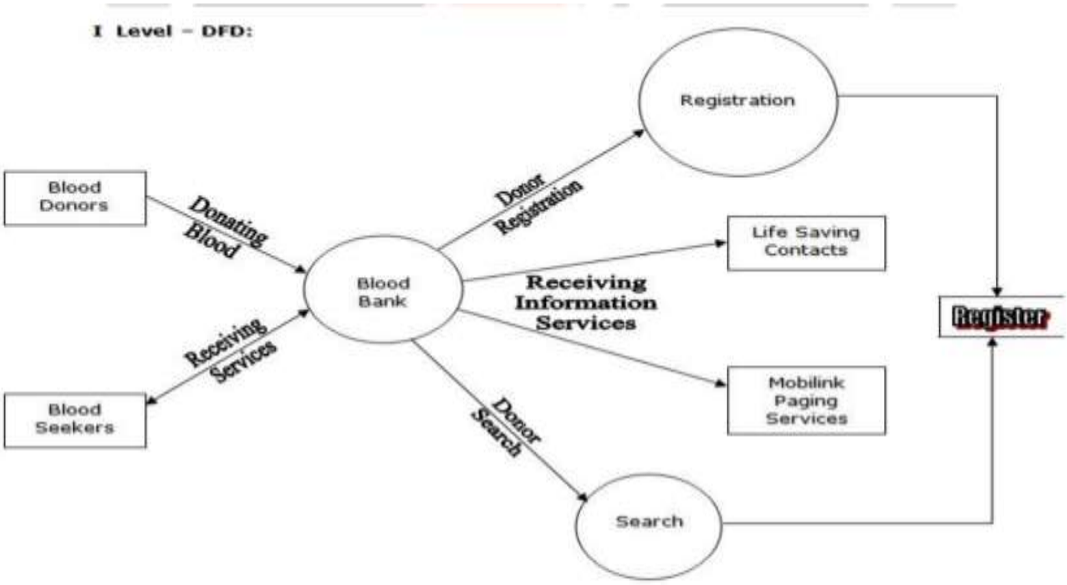
3.1. INTRODUCTION

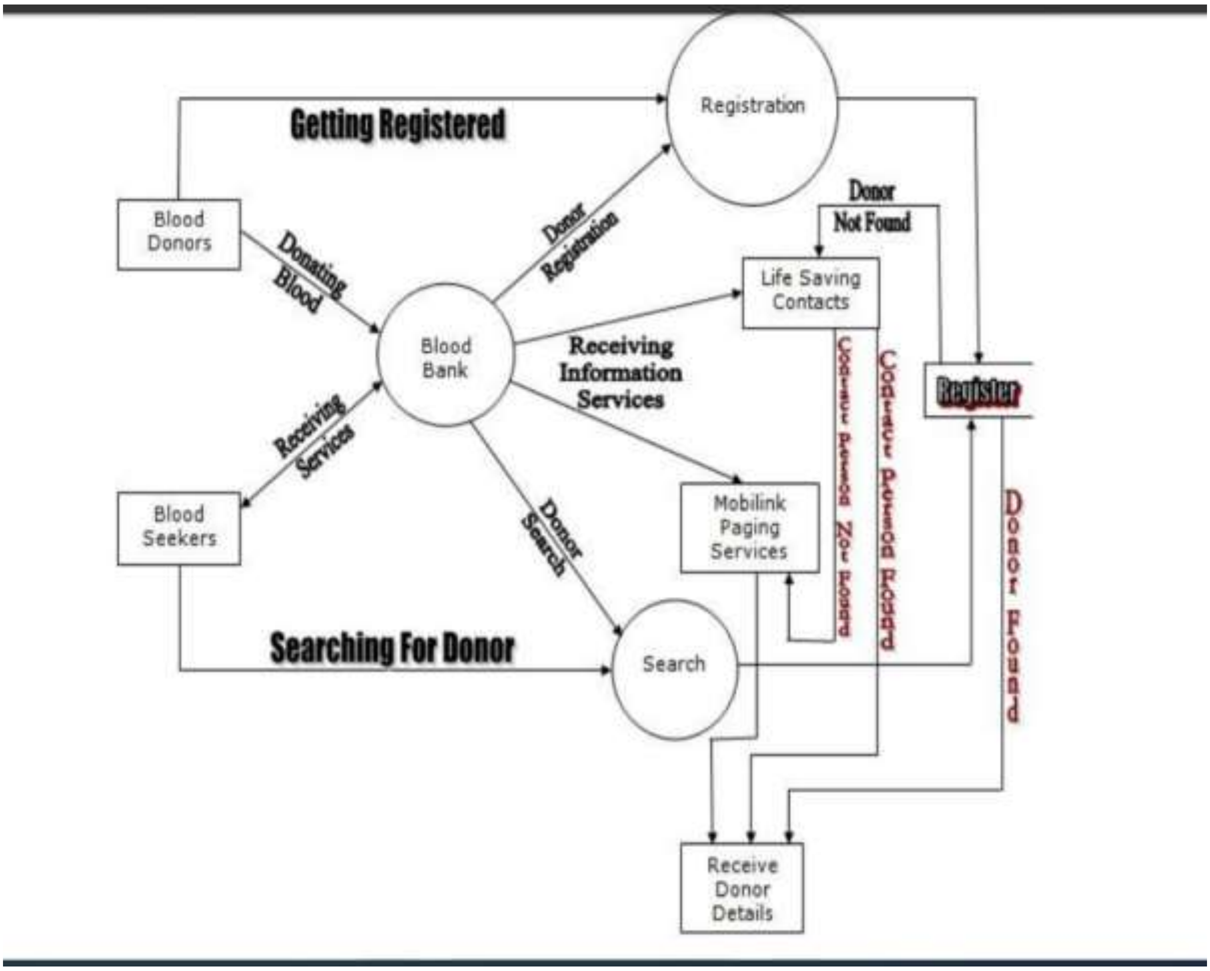
Blood bank management system is an web based application that allows users toAccess all the information regarding blood bank management software. It is readily scalable and adaptable to meet the emergency need of blood banks which is the key factor for the health care sector; it implements all the functionalities required for blood bank.

3.2. EXISTING SOFTWARE:

In the existing system the BLOOD BANK MANAGEMENT system exhibited at a lot of ineffectiveness an inefficiency of various existing software. It doesn’t integrate the blood banks that will lead to time consuming while searching of a particular group of blood. It doesn’t have proper information which is very difficult to supply the blood to the required people. The sharing of information among the various blood banks

3.3. DFD OF PRESENT SYSTEM:





3.4. What’s new in the system to be developed:

The other features in which the blood banks provide should also be incorporated in this application. The use of certificate in which the donor can get free blood from the blood bank using his certificate issued during blood donation. The available quantity of each particular type of blood is shown in the application. The user will find the near by blood banks .

4. PROBLEM ANALYSIS:

4.1. PRODUCT DEFINITION:

This is blood bank management system for blood banks and hospitals. This is an mobile app for admin and the users to track the availability of blood, donors and receivers. A user is added after registering on the app. The admin of that particular blood bank will add the user to the database along with contact details. Unlike the other blood bank management systems, this system allows the user to get the details from the mobile app. The user after being registered on the app he/she can donate or receive blood from the blood bank. After the user authenticates into the mobile app, he should check for functionality either receiving or donating and the process goes on.

4.2. FEASIBILITY ANALYSIS:

The Feasibility analysis is done before proceeding to implementing the proposed project idea. This is done to know the extent of possibility of the proposed project idea. This study gives us an idea of possibility of the project in terms of cost, technical resources and operations as the main aspects of the study.

4.2.1. COST FEASIBILITY:

This is a very simple software that can be developed very fast with even a single developer who knows mobile app development. This is very cost effective as there is no need of many developers for the software to be developed and takes a very less effort when compared to developing other big software and apps. Hence, this idea is economically feasible to implement.

4.2.2. TECHNICAL FEASIBILITY:

The technologies required for the building of this BLOOD COUNT system is quite basic. They are . This implies the need of very less number of developers for the development of the parking management system

4.2.3. OPERATIONAL FEASIBILITY:

The BLOOD COUNT will be easy to control as there is no complicatedimplementations for anyone to manage the system with updates and improvements. Not only it is easy to operate and control but also very efficient to use in terms of ease that the system provides to its user with elegant user experience and the details are provided with a single click of sign in into the BLOOD COUNT for users which displays the details of a particular user. Therefore, the system is easy to control, efficient and provides a quality service to its users.

4.3. PROJECT PLAN:

ACTIVITIES	JUNE	JULY	AUGUST	SEPTEMBER	OCTOBER
Complete UI Design	🟩	🟩			
Adding Functionality			🟩		
Creating Database				🟩	
Connecting App and Database				🟩	
Testing and Debugging					🟩

5. SOFTWARE REQUIREMENT ANALYSIS:

5.1. INTRODUCTION:

It is the process of gathering the requirements to build a software. In this we have to perform certain activities like

- Identify customer's needs.
- Evaluate system for feasibility.
- Perform economic and technical analysis.
- Allocate functions to system elements.
- Establish schedule and constraints.
- Create system definitions.

REQUIREMENTS GATHERING:

a) Hardware Requirements:

- Computer/ Laptop
- Configuration

  i. i5/i7 Core

  ii. Minimum 8GB RAM

b) Software Requirements:

- Windows
- Node.js
- Visual studio code
- React native CLI
- MySQL

5.3. SOFTWARE REQUIREMENT SPECIFICATIONS (SRS):

It is a description of software system to be developed. It helps to prevent the failure of the software project. It is a document where the requirements of the software is listed. It has two types of requirements they are

i. **Functional requirements:** It defines a system. It has the functions of a system. It tells what a software system will do.

ii. **Non-Functional Requirements:** It is used to judge the operation of a system.

FUNCTIONAL REQUIREMENTS:

Management

1) Sign Up
2) Login
3) Update availability of Blood
4) Generate unique id

User

1) Sign Up
2) Login
3) Find Nearby Blood Banks
4) Donate blood

NON-FUNCTIONAL REQUIRMENTS

1) USABILITY
2) SECURITY
3) RELAIBALITY
4) PORTABILITY

MANAGEMENT:

FR 1: Sign Up

DESCRIPTIOPN: If the user doesn’t have an account then he will be asked to create an account.

FR 1.1: Select Sign Up option.

I/P: Click on the Sign Up option.

O/P: A new window appears asking the details like name of the blood Bank, mobile number, email id and set password.

FR 1.2: Select create account option.

I/P: click on create account option after filling all the details

O/P: A new window appears asking for enter the OTP.

PROCESSING: It will checks whether the entered OTP is correct or not. if it corrects then, it will show you home page otherwise it shows an error message.

FR 2: Login

DESCRIPTION: If the user wants to get access to all the functionalities of Blood Count, he should login using his registered email id and password.

FR 2.1: Select Login option.

I/P: Click on Login option.

O/P: A new window appears asking for enter email id and password.

FR 2.2: Email id and password.

I/P: Enter Email id and password

O/P: A new window appears showing Blood Count homepage or error message occurs

PROCESSING: It will checks whether the entered email id and password is correct or not. If it is correct a homepage will appears otherwise an error message will be shown.

FR 3: Availability of Blood Units

DESCRIPTION: It shows availability of number of blood units present in there blood bank**.** If there is change in the availability of blood group units corresponding blood banks will update blood group units.

FR 3.1: Select availability of blood units option.

I/P: Click on availability of blood units option.

O/P: A new window appears displaying the blood groups with their availability.

PROCESSING: It will fetch the details from the data base.

FR 3.2: Select Edit option

I/P: Click on edit option.

O/P: A dialogue box appears to update availability of blood units.

FR 3.2.1: Select Save option.

I/P: Click on save option.

O/P: A toast appears which shows data updated successfully.

.

PROCESSING: The data will be updated in the data base.

FR 4: Generate Unique id

DESCRIPTION: It will generate unique id which will help the user to generate e-certificate

FR 4.1: select generate unique id option

I/P: click on generate unique id option

O/P: A new window appears which will display the unique id with donor name

USER:

FR 1: Sign Up

DESCRIPTIOPN: If the user doesn't have an account then he will be asked to create an account.

FR 1.1: Select Sign Up option.

I/P: Click on the Signup option.

O/P: A new window appears asking the details like username, mobile Number, email id, blood group and set password.

FR 1.2: Select create account option.

I/P: click on create account option after filling all the details

O/P: A new window apperas asking for enter the OTP.

PROCESSING: It will checks whether the entered OTP is correct or not. if it corrects then, it will show you home page otherwise it shows an error message.

FR 2: Login

DESCRIPTION: If the user wants to get access to all the functionalities of Blood Count, he should login using his registered mobile number and password.

FR 2.1: Select Login option.

I/P: Click on Login option.

O/P: A new window appears asking for enter email and password.

FR 2.2: Email address and password.

I/P: Enter email and password

O/P: A new window appears showing Blood Count homepage or an error message occurs.

PROCESSING: It will check whether the entered mobile number an password is correct or not. If it is correct a homepage will appears otherwise an error message will be shown.

FR 3: Nearby Blood banks

DESCRIPTION: It will display the nearby blood banks and availability of blood group units in the blood bank.

FR 3.1: Select Nearby Blood Bank option.

I/P: Click on Nearby Blood Bank option.

O/P: A new window appears asking the user to enter the area name.

FR 3.2: Select search option

I/P: Click on Search option.

O/P: A new window appears displaying the nearby blood banks with availability of blood group units

PROCESSING: It will fetch the blood banks details by area name from the data base.

FR 4: Donate Blood

DESCRIPTION: If user want to donate blood he or she can donate blood in nearby blood banks through this option

FR 4.1 : Select Donate Blood option:

I/P: click on donate blood option

O/P: A new window appears and asking the user to select time slot

FR 4.2 : Generate Certificate

I/P: click on generate certificate option

O/P: A new window appears asking the user to enter unique id

FR 4.3 : Select Print option

I/P: click on print option

O/P: e-certificate will be generated

PROCESSING: It will check the unique id, if it correct then it will generate e- certificate otherwise not

Non-Functional Requirements:

1.Usability

As the system is simple to handle and navigates within the most expected manner with no delays. therein case the computer programme reacts consequently and transverses quickly between its states.

2.Security

The most security concern is for users account thence correct login mechanism ought to be wont to avoid hacking. The pill id registration is much to spam check for increasing the safety. Hence, security is provided from unwanted use of recognition package.

3.Realiabilty

As the system give the proper tools for discussion, drawback finding it should be created positive that the system is reliable in its operations and for securing the sensitive details

4.Portability

Portability in high level programming is that the usability of an equivalent package in numerous environments. The pre demand for movability is generalized abstraction between the appliance logic and system interfaces. once package with an equivalent practicality is created for many computing platforms, movability is that the key issue for development price reduction.

6. DESIGN:

6.1. INTRODUCTION:

Design is a process for deliberately creating a product to meet a set of needs. Design considers user and consumers by asking what the user wants in a product. There are different types of designs used to develop a project. They are

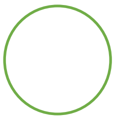
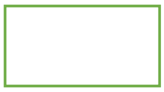
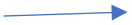
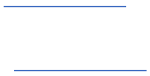
- Data flow diagrams (DFD)
- Entity-Relationship Diagram
- Use case Diagram
- Flow chart

6.2. Data Flow Diagram (DFD):

It is a way of representing a flow of data of a process or a system. It provides output and inputs of each entity.

Rules for creating a DFD:

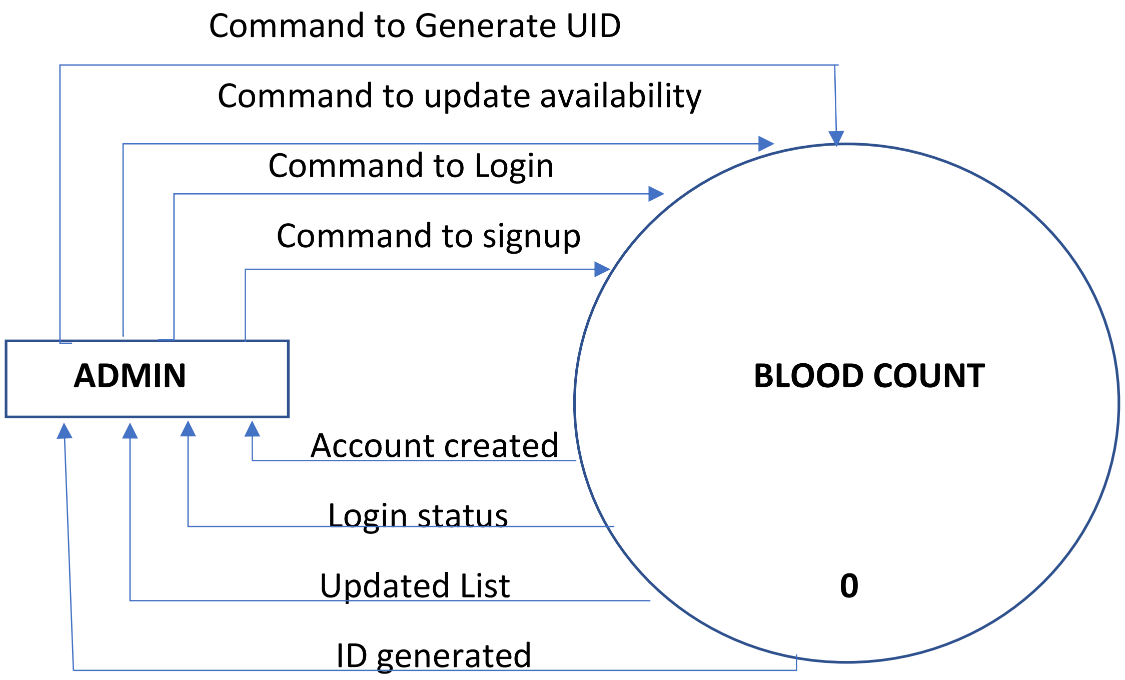
- Entity names should be general.
- Process should be numbered for easier mapping.
- Minimum three process in all level DFD except the zero level DFD.
- Maximum number of process in one DFD is recommended to be from 6 to 9.
- Every process must have an input and output.

Symbol	Meaning
	Process
	Entity
	Data Flow
	Data Store

DATA FLOW DIAGRAMS

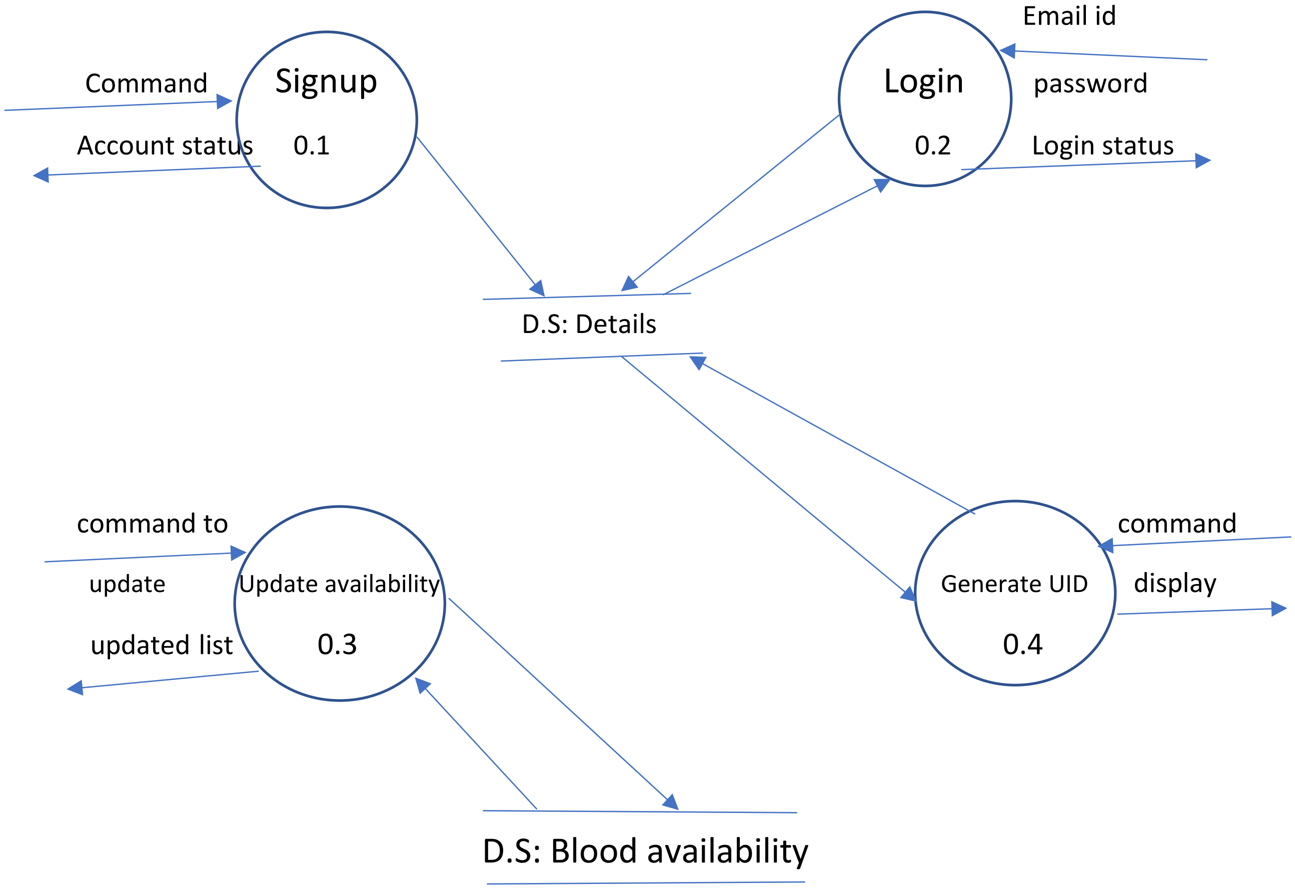
Management:

Level-0



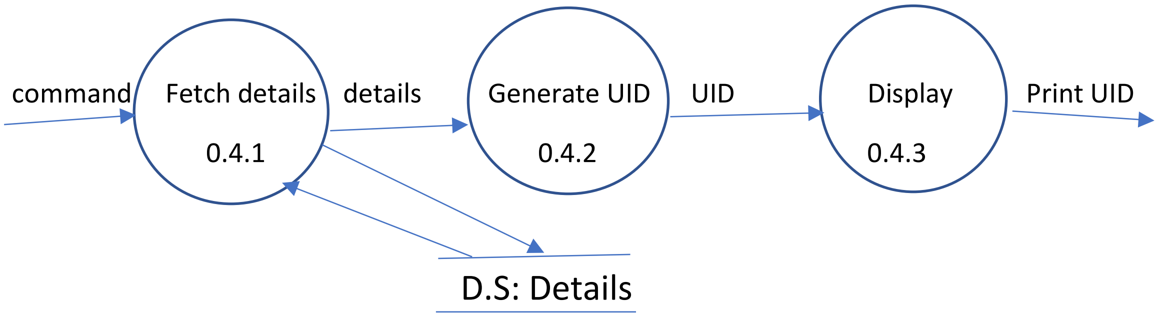
Description: It is level-0 DFD in which we will know about the functional requirements and their output for admins. We will divide it further in next levels. When we give command to Signup we will get the output as account created and for login we will get about login status and for update availability we will get output as updated list and for generate UID we will get UID as output.

Level-1



Description: It is level-1 DFD in which we divided the level 0 DFD. We will give command to the signup, login , update availability and generate UID they will fetch details from data store and generate appropriate output.

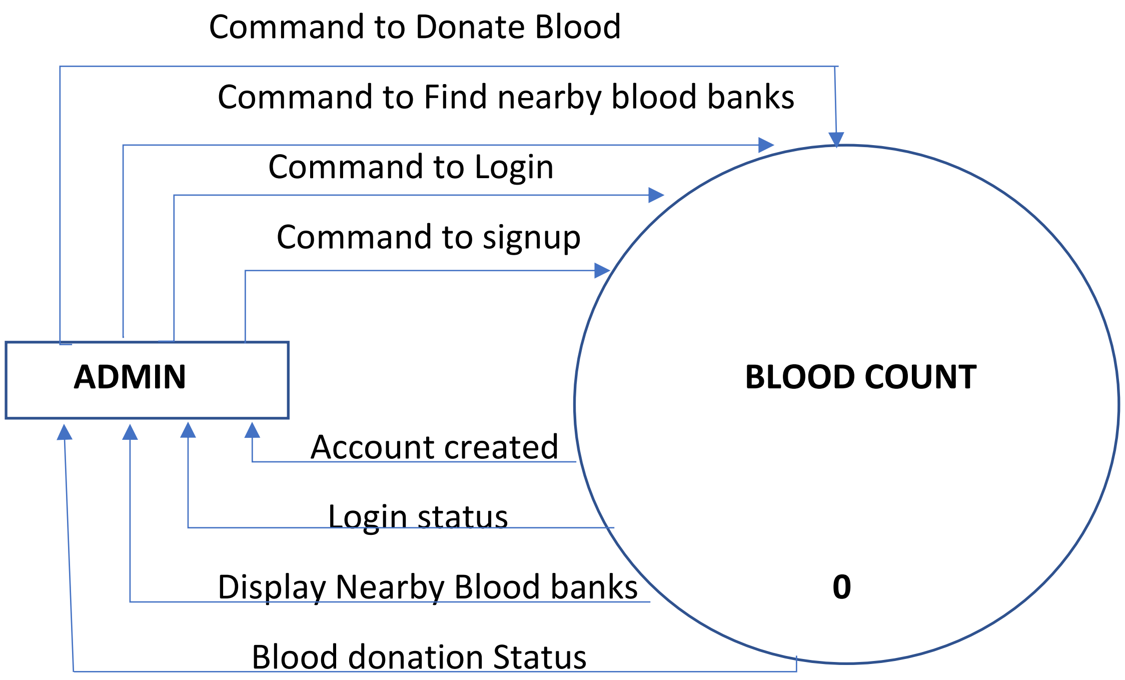
Level-2



Description: It is level 2 DFD in which we divided the Generate UID module. When we give the command it will fetch details of user and with that details it will generate UID and finally we will display it and also print it.

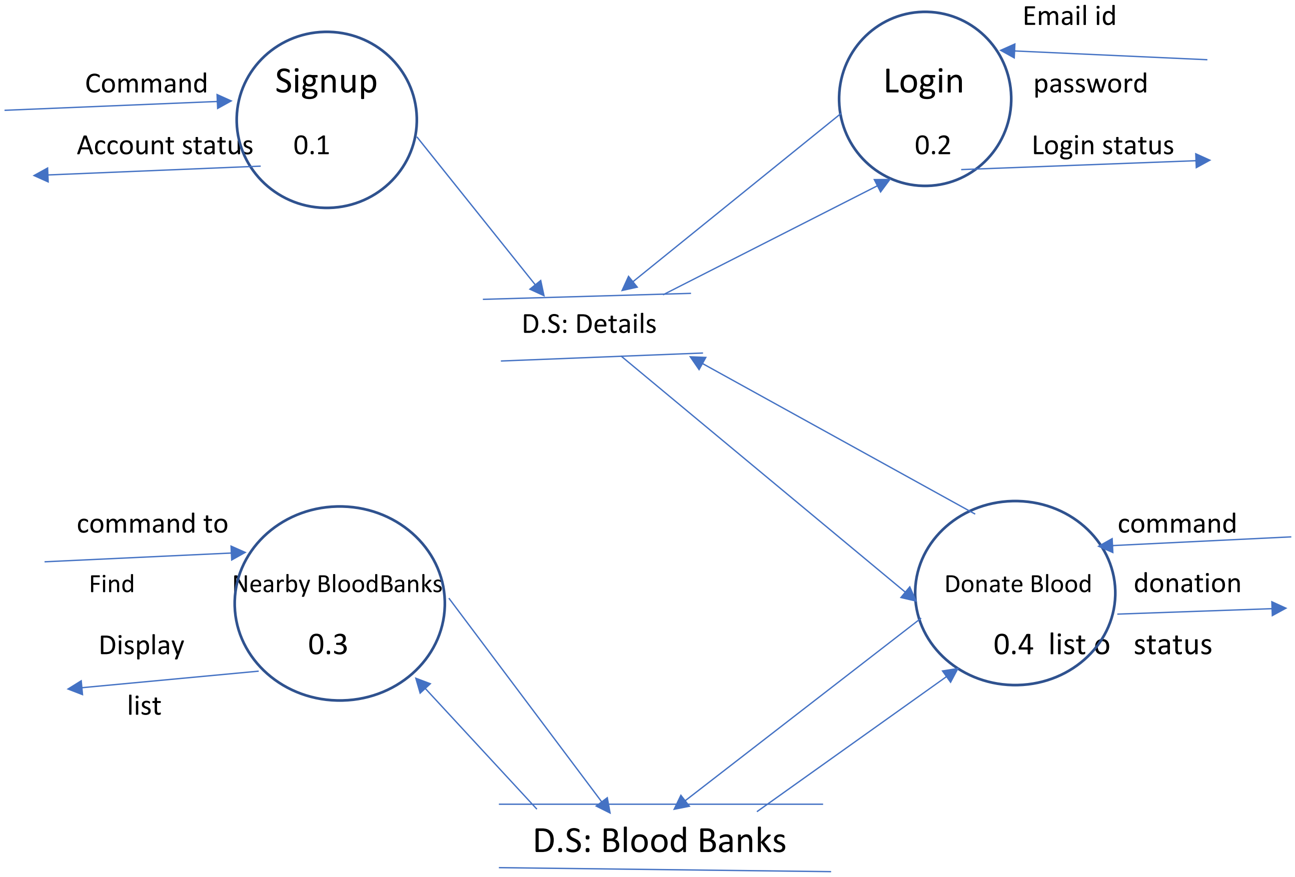
USER:

Level-0



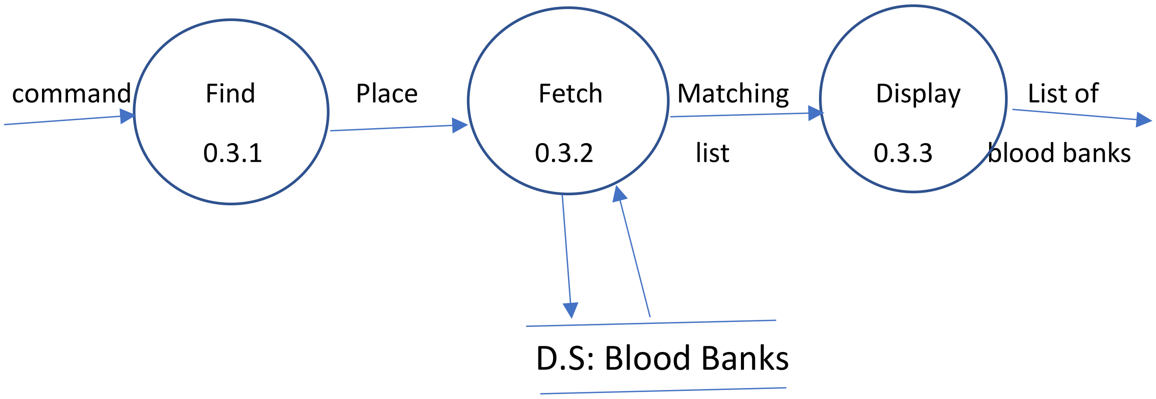
Description: It is level-0 DFD in which we will know about the functional requirements and their output for users. We will divide it further in next levels. When we give command to Signup we will get the output as account created and for login we will get about login status and for finding nearby blood banks we get output displaying the nearby blood banks and for donate blood he will get the blood donation status.

Level-1

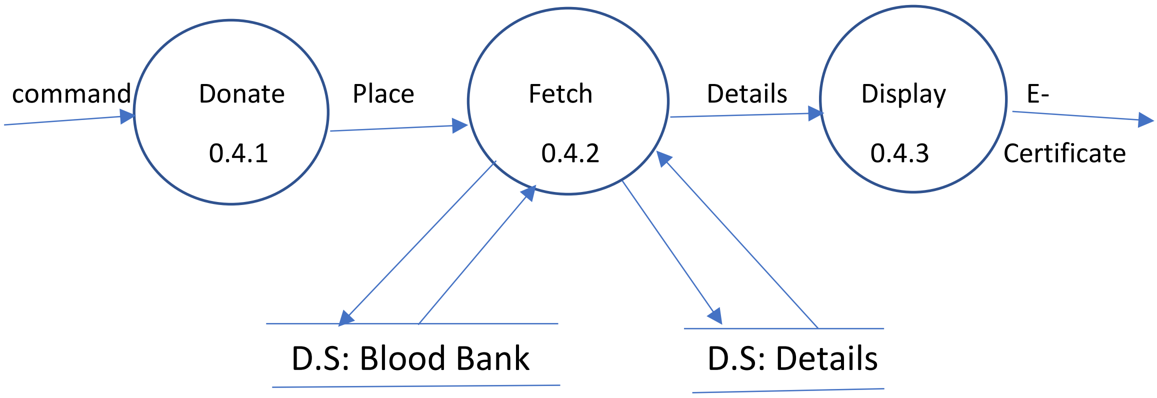


Description: It is level-1 DFD, we got this DFD by dividing the level-0 DFD. In this we will give the command to signup where the user will enter the details and he will get a message as account created as output. We will give the command to login and we enter details there and finally we will get output as login status. We will give commands Find Nearby blood banks and donate blood we will get output as blood banks and donation status. In these we have data stores like details and blood banks.

Level-2



Description: It is level 2 DFD we have divided the Find nearby blood banks module. In this firstly we will give command to find and we will enter details the details are send and we will fetch from data store blood banks and display the list to the user.



Description: It is also level 2 DFD, we have divided the Donate blood module. In this we will give command then we will enter place afterwards it will send request to the blood bank and also fetch details from data store and finally after completion of donation it will generate E-certificate.

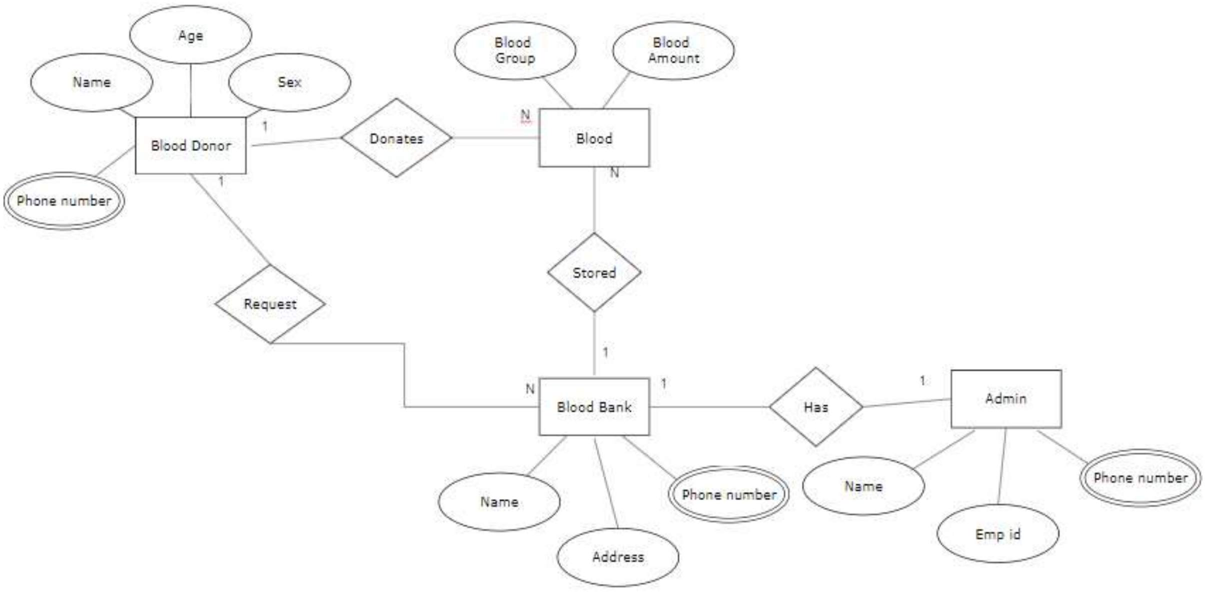
6.3. E-R Diagram:

Entity Relationship model is based on the notion of real-world entities and relationship among them.

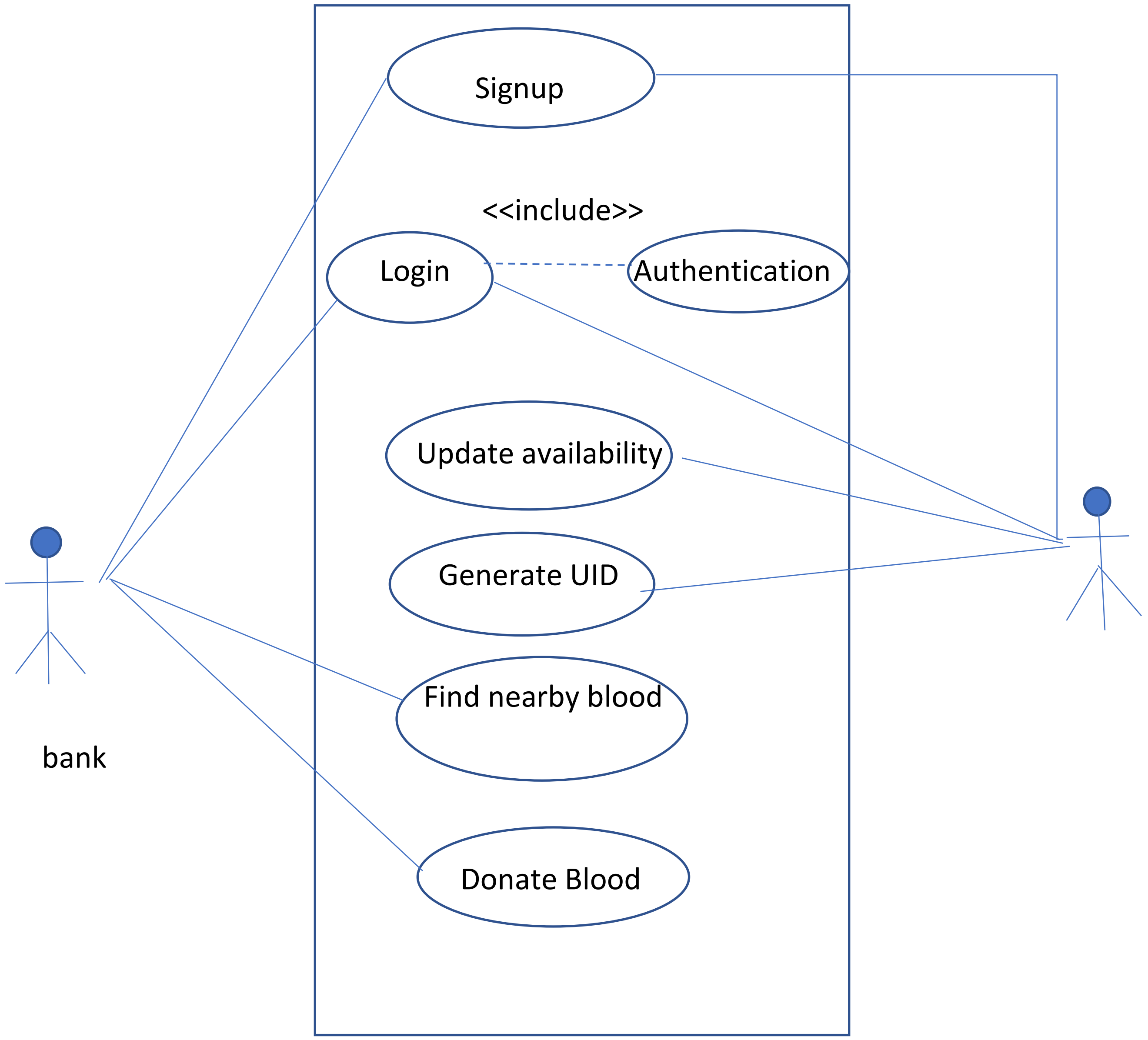
It has entities and attributes.

i. Entity is like class in C++.
ii. Attributes is like object of class.

Symbol	Name	Meaning
⬭	Oval	Attributes
▭	Rectangle	Entity sets
◇	Diamond	Relationship among entity sets
◎	Ovals	Multi-valued attributes
─	Line	Links entity sets to attributes and entity sets to relationship

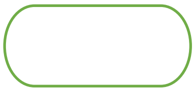
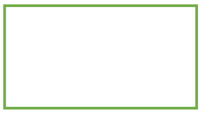
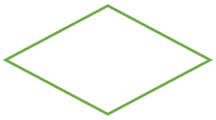
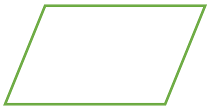
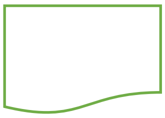
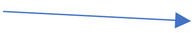

Age
Name
Sex
Blood Donor
Phone number
1
Donates
N
Blood Group
Blood Amount
Blood
N
1
Stored
Request
1
N
Blood Bank
1
Has
1
Admin
Name
Address
Phone number
Name
Emp id
Phone number

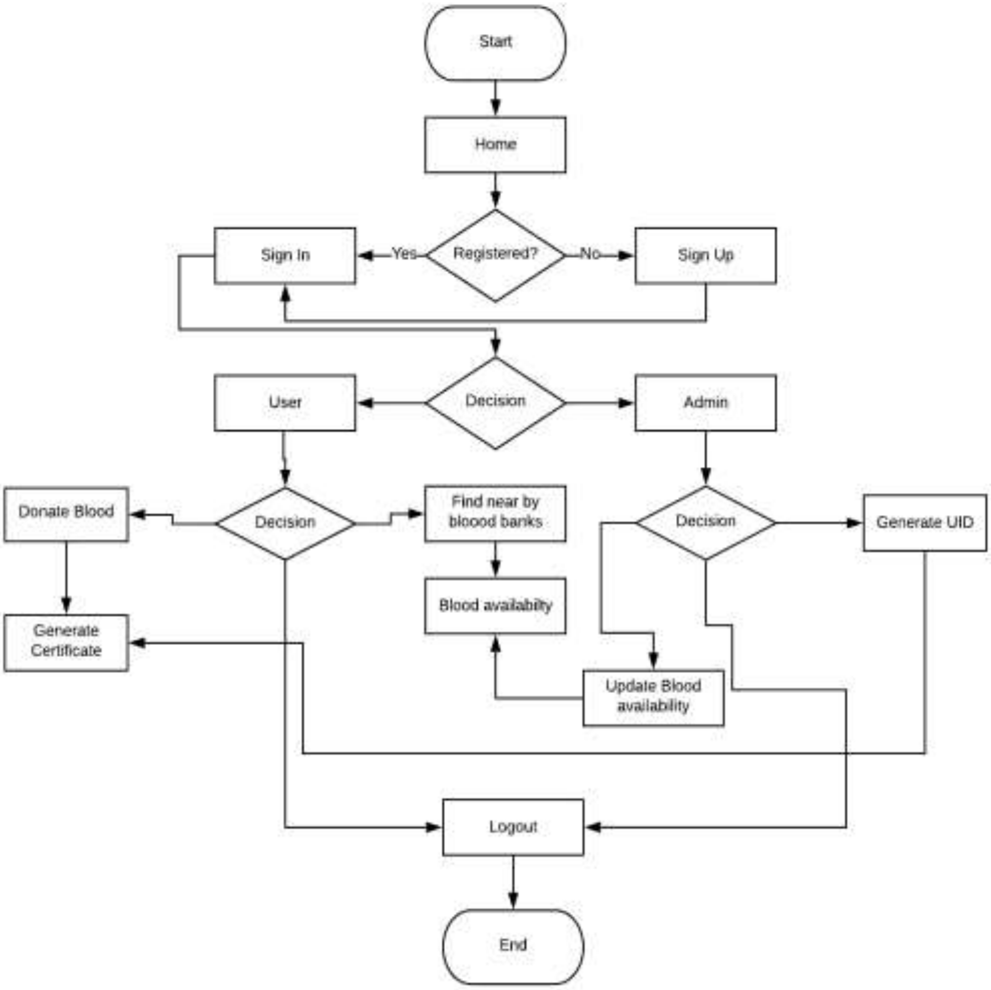
6.4. USE CASE DIAGRAM



6.5. Flow Charts:

It is a graphical or symbolic representation of a process. It is a tool for representing algorithms and programming logic. There are different symbols which are used to represent that process. The symbols are linked with the arrows which represent the process flow direction.

Symbol	Name	Function
	Terminator	Starting or ending point of the system.
	Process	It indicates particular operation.
	Decision	Decision or branching point.
	Data	Information entering or leaving the system
	Document	Printout
	Flow	Flow of the sequence and direction of the process.


Start
Home
Registered?
Yes
No
Sign In
Sign Up
Decision
User
Admin
Donate Blood
Decision
Find near by bloood banks
Decision
Generate UID
Generate Certificate
Blood availabilty
Update Blood availability
Logout
End

7. TESTING:

Testing is the process of executing a system with the intent of finding errors. It is an important phase in an application/software development. It is defined as the activity to check whether the actual output matches the expected output. It is used to identify the defects(errors) in the software. The testing can be done manually or using the automated tools like selenium, Appium etc.

7.1 Functional Testing:

It is a type of software testing where we will use it to test the functional requirements of the software. The functions are tested by giving them input and examining the output. It ensures that the functional requirements are properly satisfied by the application.

We will use mainly black box testing technique in which the internal logic of the system being tested is not known to the tester.

Functional testing involves the following steps:

- Identify the functions that the software/application is expected to perform.
- Create input data based on the function's specifications.
- Determine the output based on the function's specifications.
- Execute the test case.
- Compare the actual and expected outputs.

7.2 Structural Testing:

It is used to test the structure of the code. It is also called as White Box Testing or Glass Box Testing. It requires the knowledge of the internal implementation (code). So mainly it will be done by the developers. It will mainly concerned with how the system performs rather than functionality of the system.

Structural Testing Techniques:

a) **Statement Coverage:** This technique is mainly used for covering all programming statements with minimal tests.

b) **Branch Coverage:** This technique will run a series of tests to ensure that all the branches are tested at least once.

c) **Path Coverage:** This technique will test the all the possible paths through which each statement and branch are covered.

7.3) Levels of Testing:

There are four recognized levels of testing. They are

i. **Unit Testing:** It means testing the modules or a section of code individually. It is performed at the development stage only.

ii. **Integration Testing:** After the Unit testing we will perform integration testing. We will combine the modules and perform integration testing.

iii. **System Testing:** The name itself indicates that all the components of a software are tested as whole. We will use this to ensure that the overall product meets the requirements mentioned.

iv. **Acceptance Testing:** It is the final level test which is conducted to determine whether the system is ready for release. It will performed by making pre-written testcases that will be used to test the application.

7.4) Testing the Project:

- Enter username and password and click on login. If it is login then login function is working perfectly.
- Verify whether the password is displayed or not when it is entered into the application.
- Verify whether other than numbers in username if it is login or not.
- If a user got access to update blood availability, then there is an error.
- If user generate an e-certificate without donating blood, then there is an error.
- If the unique id is repeated, then there is an error.

8. IMPLEMENTATION:

Implementation is the final and important phase. The most critical stage is in achieving a successful new system and in giving the users confidence that the new system will work and be effective. The system can be implemented only after thorough testing is done and if it found to working according to the specification. Implementation is the stage in the project where the theoretical design is turned into the working system and is giving confidence to the new system for the users i.e. will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of method to achieve the changeover, an evaluation of change over methods. A part from planning, major task of preparing the implementation is to educate the targeted users. The more complex system is designed, the more complex will be the system implementation and publicity effort required.

8.1. IMPLEMENTATION OF PROJECT:

The implementation type of this project falls under the first type of different categories of implementation discussed in the above introduction of this section i.e. implementation of a BLOOD COUNT to replace a manual system. After completing the designing, development and unit & integrated testing phases of the application; the implementation phase will be carried out. During the implementation phase we faced several problems like less RAM, low internet connectivity and updates of packages. We overcome this problems by increasing the RAM, Processor and Bandwidth.

8.2 .CONVERSION PLAN:

The steps followed to make an REACT NATIVE app:

1. Set up the React Native Environment and install all the required packages.
2. Create components folder inside the root(Use visual studio).
3. Import individual files inside the app.js folder
   Design the interface in the individual files using stylesheets, etc.,

4. set up the database (Firebase) and connect with the app and deploy it.
5.Now save the file and open the simulator and load it
   You will see your designed interface

8.3. POST_IMPLEMENTATION AND SOFTWARE MAINTENANCE:

Post the implementation the project is tested in terms of whether it is behaving and giving the results as required and as expected. We can perform the maintenance of the project by adding new features and any bugs as observed by the team.

Maintenance is the enigma of the project development. Analysts and programmers spend more time maintaining programs than writing them. Though maintenance is not considered a part of software development, but it’s an activity that’s extremely important in the life of BLOOD COUNT project.

i. Corrective Maintenance: After the Implementation, correcting the residual errors if any. If such errors are discovered, the source of it should be detected and removed. This phenomenon falls under corrective maintenance.

ii. Perfective Maintenance: Sometimes changes have to be done according to the user requirements. This type of changes to the BLOOD COUNT is called perfective maintenance.

iii. Adaptive Maintenance: BLOOD COUNT often must be upgraded and enhanced to include more features and provide more services. This also requires modification of BLOOD COUNT.

9. PROJECT LEGACY:

9.1. CURRENT STATUS OF THE PROJECT:

Our project BLOOD COUNT application is working properly. It is helping the people by donating the blood when they are free, finding the nearby blood banks and the availability of blood units in that blood bank in emergency. It is also generating a e-certificate which will be used to get the blood from blood bank when he or his family members require blood.

9.2. REMAINING AREAS OF CONCERN:

Censorship: The most important area of concern is to prohibit or restrict of uploading incorrect and illegal contents over the application and no one signup as an admin without contacting with us. The concern of identifying and learning which contents aren't appropriate is the major challenge for us. For now, to identify and restrict such kind of publishing activities, we will take the starting users reports and then taking action. But in near future we, as the administrator of the app, want and will try to identify and restrict such kind of contents from publishing beforehand.

Data Manageability: Another important area of our concern is to manage the generated data. Being just in the initial stage, the management of data is very much an easy task but with every passing day, with every new user being added and with ever increasing generation of data the complexity and manageability of the database becomes a complicated task. Therefore, to avoid such kind of situation, we've to start looking for better manageability and scalability options like using cloud.

Data Security: Any administrator can never forget about the privacy and protection of the available data to guard it against illegal usage. Similarly, we totally care and in concern with our user confidential data security. We treat all the data same and try to keep respective level of security for the respective type of data. Even after implementing many security measures to protect the user credentials, there always remains a security concern over such issues and all we can do is to keep making our application software more secure with every updates.

.

Future Scope of Improvements: So far what we have developed and deployed is just a basic structural project. It is a prototype to work for demonstration purpose presently. Using this they can know whether the blood group which they want is present in which blood bank. So that they can get blood unit easily during emergency. There may be some questions regarding the registering of blood banks we have to work on it to improve the security. We want to add events to that which will help the users to know about the blood donation camps happening around them.

9.3. TECHNICAL AND MANAGERIAL LESSONS LEARNT:

This project has benefited us in many ways. We are exposed to a new technical knowledge and new environment (React Native). It has helped us to sharpen our skills and knowledge and we can build an application for iOS and Android at a time. This wonderful experience will immensely help each of us in further learning of advance concepts of React Native Framework.

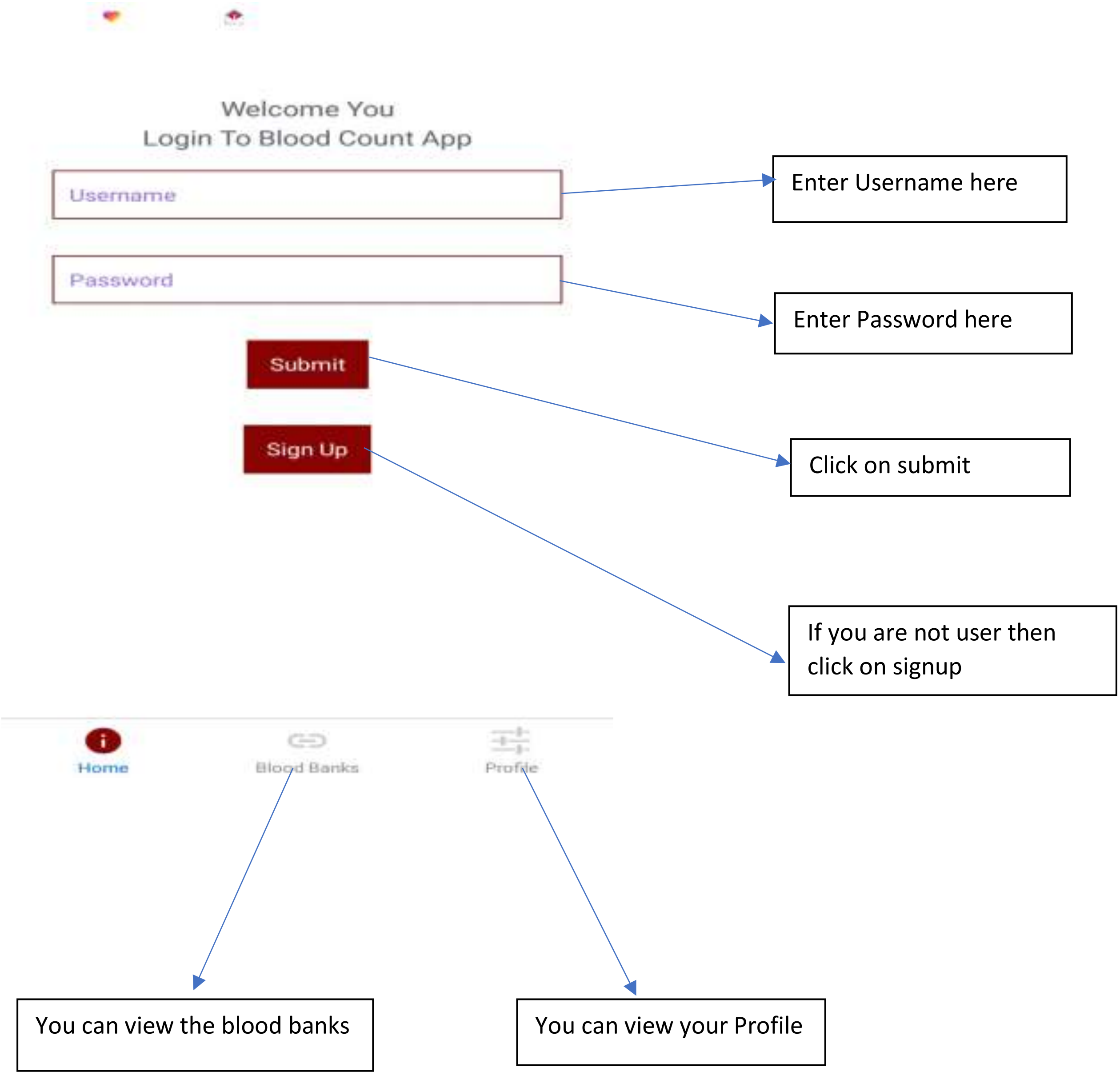
Technical lessons learnt:

- Installation and using of React Native environment.
- Installation and using of Node.js
- Configuration of Firebase
- Using React Native CLI

Managerial lessons learnt:

- Planning of Duration and schedule of the project
- Coordination
- Integrating individual work to make it collaborative work
- Strategic planning to avoid miscommunication among the team members

10. USER GUIDE:



Welcome You
Login To Blood Count App

Enter Username here

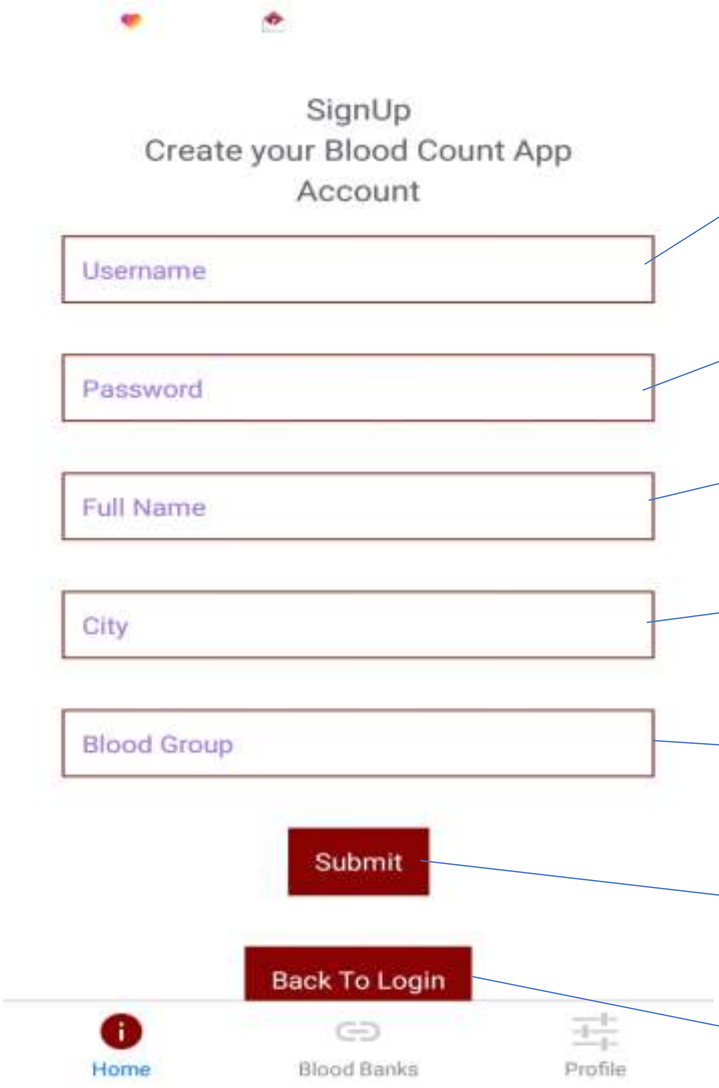
Enter Password here

Click on submit

If you are not user then click on signup

You can view the blood banks

You can view your Profile



SignUp
Create your Blood Count App Account

Username

Password

Full Name

City

Blood Group

Submit

Back To Login

Home
Blood Banks
Profile

Enter Username here

Enter Password here

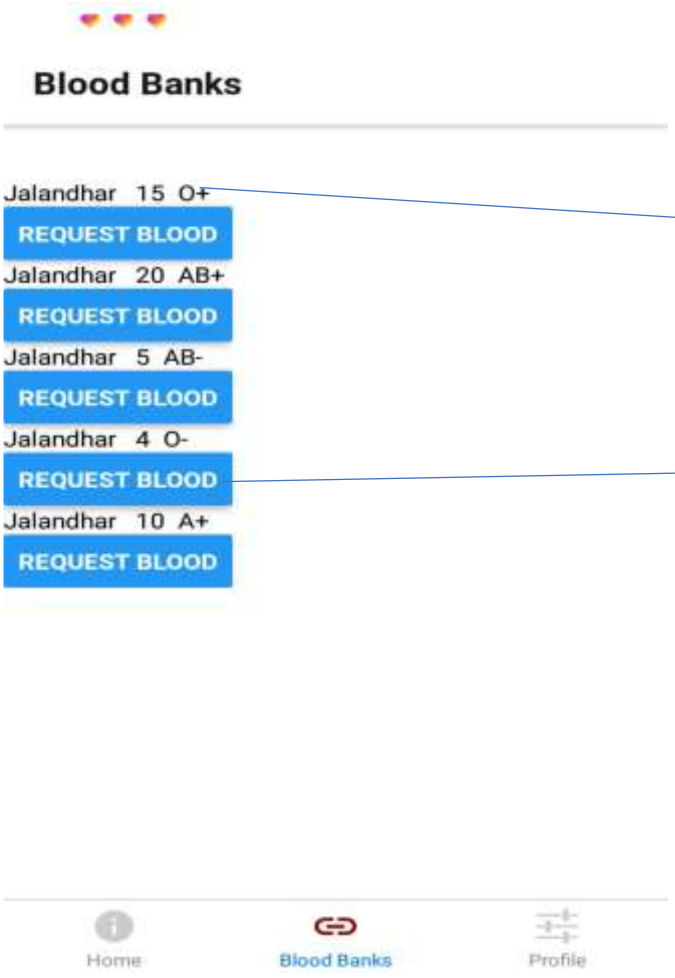
Enter Full name here

Enter City of your

Enter the Blood group of your

Click on submit

After creating account click on back to login



Blood Banks

Jalandhar 15 O+

Jalandhar 20 AB+

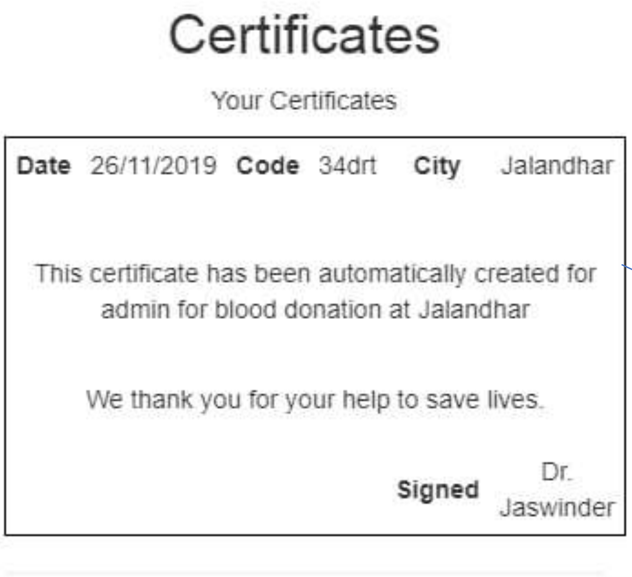
Jalandhar 5 AB-

Jalandhar 4 O-

Jalandhar 10 A+

You can see the blood banks and availability of blood units

You can request the blood



Certificates

Your Certificates

Date 26/11/2019 **Code** 34drt **City** Jalandhar

This certificate has been automatically created for admin for blood donation at Jalandhar

We thank you for your help to save lives.

Signed Dr. Jaswinder

You can get certificate like this after donating blood

11. SORCE CODE:

App.js

```
import { AppLoading } from 'expo';

import { Asset } from 'expo-asset';

import * as Font from 'expo-font';

import React, { useState } from 'react';

import { Platform, StatusBar, StyleSheet, View } from 'react-native';

import { Ionicons } from '@expo/vector-icons';

import AppNavigator from './navigation/AppNavigator';

export default function App(props) {

  const [isLoadingComplete, setLoadingComplete] = useState(false);

  if (!isLoadingComplete && !props.skipLoadingScreen) {

    return (  <AppLoading

      startAsync={loadResourcesAsync}

        onError={handleLoadingError}

        onFinish={() => handleFinishLoading(setLoadingComplete)}

      /> );

  } else {

    return ( <View style={styles.container}>

        {Platform.OS === 'ios' && <StatusBar barStyle="default" />}

        <AppNavigator />

      </View>   );

  }

}


async function loadResourcesAsync() {await Promise.all([ Asset.loadAsync([
require('./assets/images/robot-dev.png'),  require('./assets/images/robot-prod.png'),   ]),
    Font.loadAsync({
      // This is the font that we are using for our tab bar
      ...Ionicons.font,
      // We include SpaceMono because we use it in HomeScreen.js. Feel free to
      // remove this if you are not using it in your app
      'space-mono': require('./assets/fonts/SpaceMono-Regular.ttf'),  }),  ]);
}
function handleLoadingError(error) {
  // In this case, you might want to report the error to your error reporting
  // service, for example Sentry
  console.warn(error);
} function handleFinishLoading(setLoadingComplete) {
  setLoadingComplete(true);
}
const styles = StyleSheet.create({ container: {   flex: 1,   backgroundColor: '#fff',  },});
```

For Signup

```
import * as WebBrowser from 'expo-web-browser';
import React from 'react';
import { Image, Platform,  ScrollView,  StyleSheet, Text, TouchableOpacity, View, TextInput
AsyncStorage, } from 'react-native';
import { MonoText } from '../components/StyledText';
export default class SignUpScreen extends React.Component
{
static navigationOptions = { header: null }
  state = { email: '',password: '', fullname: '', city: '', address: '', bloodgroup: '',}
 _storeData = async (id) => {
  try
```

```
{
await AsyncStorage.setItem('userid', JSON.stringify(id));
} catch (error) {
   // Error saving data
  }
};
_getData = async () => {
  try
{ const existingProducts =await AsyncStorage.getItem('userid');
console.log(existingProducts); }
catch (error) {
   // Error saving data
  }
};
 handleEmail = (text) => { this.setState({ email: text }) }
 handlePassword = (text) => { this.setState({ password: text }) }
 handleFullName = (text) => { this.setState({ fullname: text }) }
handleCity = (text) => { this.setState({ city: text }) }
handleAddress = (text) =>{ this.setState({ address: text }) }
handleBloodGroup = (text) => { this.setState({ bloodgroup: text }) } back =()=>
 {
  this.props.navigation.navigate('Home', { itemId: "" ,  otherParam: 'anything you want here',});
 }
signup = (email, pass,fullname,city,address,bloodgroup) => {
 const LOGINurl="http://blookbank.flashcontacts.org/Home/AddUser?
username="+email+"&password="+pass+"&fullname="+fullname+"&city="+city+"&addres
s="+address+"&bloodgroup="+bloodgroup+"&type=USER";
fetch(LOGINurl) .then((response) => response.json()).then((responseJson) => { //return
responseJson.movies;
  if(responseJson!=null)
    {
```

```
    //alert(responseJson.Id);
    // setValue(responseJson.Id);
    this._storeData(responseJson.Id);
   //AsyncStorage.setItem('userid', JSON.stringify(responseJson.Id));
   //this._getData();
 //console.log(AsyncStorage.getItem('userid'));
 this.props.navigation.navigate('Home', { itemId: responseJson.Id ,otherParam: 'anything you want here', });}
else { alert("Wrong Username or Password") }  })
 .catch((error) => { alert("Wrong Username or Password"); }); }
render()
{ return (
   <View style={styles.container}> <ScrollView
 style={styles.container}contentContainerStyle={styles.contentContainer}>
 <View style={styles.getStartedContainer}>
 <Text style={styles.getStartedText}>SignUp</Text>
<Text style={styles.getStartedText}> Create your Blood Count App Account </Text>
<TextInput style = {styles.input} underlineColorAndroid = "transparent"
placeholder = "Username" placeholderTextColor = "#9a73ef" autoCapitalize = "none"
 onChangeText = {this.handleEmail}/>
<TextInput style = {styles.input} underlineColorAndroid = "transparent" placeholder = "Password" placeholderTextColor = "#9a73ef" autoCapitalize = "none" onChangeText = {this.handlePassword}/>
<TextInput style = {styles.input} underlineColorAndroid = "transparent"  placeholder = "Full Name" placeholderTextColor = "#9a73ef"autoCapitalize = "none" onChangeText = {this.handleFullName}/>
<TextInput style = {styles.input} underlineColorAndroid = "transparent"  placeholder = "City"placeholderTextColor = "#9a73ef"autoCapitalize = "none" onChangeText = {this.handleCity}/>
<TextInput style = {styles.input}underlineColorAndroid = "transparent"placeholder = "Blood Group" placeholderTextColor = "#9a73ef" autoCapitalize = "none"  onChangeText = {this.handleBloodGroup}/>
  <TouchableOpacity  style = {styles.submitButton}
```

```
onPress={()=>this.signup(this.state.email,this.state.password,this.state.fullname,this.state.city
,this.state.address,this.state.bloodgroup)}>

 <Text style = {styles.submitButtonText}> Submit  </Text>

</TouchableOpacity>

<TouchableOpacity style = {styles.submitButton}

onPress = { () => this.back()  }>

<Text style = {styles.submitButtonText}> Back To Login </Text>

 </TouchableOpacity> </View>  </ScrollView>  </View>  );}

}

SignUpScreen.navigationOptions = { header: null,};

function DevelopmentModeNotice() { if (__DEV__) {

 const learnMoreButton = (

<Text onPress={handleLearnMorePress} style={styles.helpLinkText}>  Learn more </Text>
); return

 <Text style={styles.developmentModeText}> Development mode is enabled: your app will
be slower but you can use useful development tools. {learnMoreButton} </Text>  ); } else

 { return ( <Text style={styles.developmentModeText}>You are not in development mode:
your app will run at full speed.</Text>  );  }

} function handleLearnMorePress() {

WebBrowser.openBrowserAsync('https://docs.expo.io/versions/latest/workflow/development
-mode/' );} function handleHelpPress() { WebBrowser.openBrowserAsync(
'https://docs.expo.io/versions/latest/workflow/up-and-running/#cant-see-your-changes' ); }
const styles = StyleSheet.create({ container: { flex: 1, backgroundColor: '#fff', paddingTop:
},developmentModeText: { marginBottom: 20, color: 'rgba(0,0,0,0.4)', fontSize: 14,
lineHeight: 19, textAlign: 'center',}, contentContainer: { paddingTop: 30, },

 welcomeContainer: { alignItems: 'center',marginTop: 10, marginBottom: 20, },

welcomeImage: {width: 100, height: 80, resizeMode: 'contain', marginTop: 3,marginLeft: -10
}, getStartedContainer: { alignItems: 'center', marginHorizontal: 50,  }, homeScreenFilename:
{ marginVertical: 7, },codeHighlightText: { color: 'rgba(96,100,109, 0.8)',}
codeHighlightContainer: { backgroundColor: 'rgba(0,0,0,0.05)',borderRadius: 3,
paddingHorizontal: 4, }, getStartedText: { fontSize: 17, color: 'rgba(96,100,109, 1)',
lineHeight: 24, textAlign: 'center',  }, tabBarInfoContainer: { position: 'absolute', bottom: 0,left:
0, right: 0, ...Platform.select({ ios: {  shadowColor: 'black', shadowOffset: { width: 0, height: -
3 }, shadowOpacity: 0.1, shadowRadius: 3,  },  android: { elevation: 20, },  }),alignItems:
'center',  backgroundColor: '#fbfbfb', paddingVertical: 20,}, tabBarInfoText: { fontSize: 17,
color: 'rgba(96,100,109, 1)', textAlign: 'center', },navigationFilename: {marginTop: 5,},
helpContainer: {  marginTop: 15, alignItems: 'center', },helpLink: { paddingVertical: 15, },
helpLinkText: { fontSize: 14,  color: '#2e78b7',}, input: {margin: 15, padding: 10,height: 40, width:300, borderColor: '#8a0303',borderWidth:  },

submitButton: { backgroundColor: '#8a0303', padding: 10, margin: 15, height: 40 }, submitButtonText:{ color: 'white' } });
```

For Home Screen and Login

```
import * as WebBrowser from 'expo-web-browser';

import React from 'react';

import { Image,Platform, ScrollView, StyleSheet, Text, TouchableOpacity, View, TextInput, AsyncStorage, } from 'react-native';

import { MonoText } from '../components/StyledText';

export default class HomeScreen extends React.Component {

static navigationOptions = { header: null } state = { email: ", password: " } _storeData = async (id) => {

 try {

  await AsyncStorage.setItem('userid', JSON.stringify(id));

 } catch (error) {

 // Error saving data

 }

};

_getData = async () => {

  try {

    const existingProducts =await AsyncStorage.getItem('userid');

    console.log(existingProducts);

  } catch (error) {

    // Error saving data

  }

};

 handleEmail = (text) => { this.setState({ email: text }) }

 handlePassword = (text) => { this.setState({ password: text }) }

 signup = () => { this.props.navigation.navigate('SignUp', { itemId: "" ,   otherParam: 'anything you want here',}); }
login = (email, pass) => { const LOGINurl="http://qrcode.flashcontacts.org/Home/Login?username="+email+"&password=" +pass;

fetch(LOGINurl)  .then((response) => response.json()) .then((responseJson) => { //return responseJson.movies;

if(responseJson!=null) { alert(responseJson.Id);

setValue(responseJson.Id);

this._storeData(responseJson.Id);

AsyncStorage.setItem('userid', JSON.stringify(responseJson.Id));

this._getData();

 console.log(AsyncStorage.getItem('userid'));

 this.props.navigation.navigate('Links', { itemId: responseJson.Id , otherParam: 'anything you want here', }); }

else{  alert("Wrong Username or Password"); } })

   .catch((error) => {   alert("Wrong Username or Password"); });

 } render() {

  return ( <View style={styles.container}> <ScrollView style={styles.container} contentContainerStyle={styles.contentContainer}>
<Viewstyle={styles.getStartedContainer}>

   <Text style={styles.getStartedText}>Welcome You</Text>

  <Text style={styles.getStartedText}> Login To Blood Count App </Text>

<TextInput style = {styles.input}  underlineColorAndroid = "transparent" placeholder = "Username" placeholderTextColor = "#9a73ef" autoCapitalize = "none" onChangeText = {this.handleEmail}/>

<TextInput style = {styles.input}  underlineColorAndroid = "transparent" placeholder = "Password" placeholderTextColor = "#9a73ef" autoCapitalize = "none" onChangeText = {this.handlePassword}/>

<TouchableOpacity style = {styles.submitButton} onPress = {  () => this.login(this.state.email, this.state.password)  }>

 <Text style = {styles.submitButtonText}> Submit </Text>  </TouchableOpacity>

 <TouchableOpacity style = {styles.submitButton} onPress = { () => this.signup()  }>

 <Text style = {styles.submitButtonText}> Sign Up </Text>  </TouchableOpacity>

</View>  </ScrollView> </View>  );}}

HomeScreen.navigationOptions = { header: null, };
```

```
function DevelopmentModeNotice() {
 if (__DEV__) {
   const learnMoreButton = (
     <Text onPress={handleLearnMorePress} style={styles.helpLinkText}>Learn more
</Text> );
return ( <Text style={styles.developmentModeText}> Development mode is enabled: your app
will be slower but you can use useful development tools. {learnMoreButton} </Text>  );
  } else {
   return ( <Text style={styles.developmentModeText}>  You are not in development mode:
your app will run at full speed  </Text>   );
  }
}
function handleLearnMorePress() {
WebBrowser.openBrowserAsync('https://docs.expo.io/versions/latest/workflow/development
-mode/' );} function handleHelpPress() { WebBrowser.openBrowserAsync(
'https://docs.expo.io/versions/latest/workflow/up-and-running/#cant-see-your-changes'   );  }
const styles = StyleSheet.create({ container: { flex: 1, backgroundColor: '#fff', paddingTop:
},developmentModeText: { marginBottom: 20, color: 'rgba(0,0,0,0.4)', fontSize: 14,
lineHeight: 19, textAlign: 'center',}, contentContainer: { paddingTop: 30, },


welcomeContainer: { alignItems: 'center',marginTop: 10, marginBottom: 20, },
welcomeImage: {width: 100, height: 80, resizeMode: 'contain', marginTop: 3,marginLeft: -10
}, getStartedContainer: { alignItems: 'center', marginHorizontal: 50,  }, homeScreenFilename:
{ marginVertical: 7, },codeHighlightText: { color: 'rgba(96,100,109, 0.8)',}
codeHighlightContainer: { backgroundColor: 'rgba(0,0,0,0.05)',borderRadius: 3,
paddingHorizontal: 4, }, getStartedText: { fontSize: 17, color: 'rgba(96,100,109, 1)',
lineHeight: 24, textAlign: 'center',  }, tabBarInfoContainer: { position: 'absolute', bottom: 0,left:
0, right: 0, ...Platform.select({ ios: {  shadowColor: 'black', shadowOffset: { width: 0, height: -
3 }, shadowOpacity: 0.1, shadowRadius: 3,  },  android: { elevation: 20, },   }),alignItems:
'center',  backgroundColor: '#fbfbfb', paddingVertical: 20,}, tabBarInfoText: { fontSize: 17,
color: 'rgba(96,100,109, 1)', textAlign: 'center', },navigationFilename: {marginTop: 5,},
helpContainer: {   marginTop: 15, alignItems: 'center', },helpLink: { paddingVertical: 15,  },
helpLinkText: { fontSize: 14,  color: '#2e78b7',}, input: {margin: 15, padding: 10,height: 40,
width:300, borderColor: '#8a0303',borderWidth:1  },
submitButton: { backgroundColor: '#8a0303', padding: 10, margin: 15, height: 40,},
submitButtonText:{ color: 'white'} });
```

Blood Banks and Linking

```
import React from 'react';

import { ScrollView, StyleSheet,AsyncStorage , FlatList, ActivityIndicator, View,
Text,Button,Image} from 'react-native';

import { Linking } from 'expo';

 export default class LinksScreen extends React.Component {

 state = { userid: null, username: null,  dataSource: [], };

 async componentDidMount() {

const existingProducts = await AsyncStorage.getItem('userid');

  this.setState({ userid : existingProducts  }); this.fetcdata(); }

 fetcdata(){const
LOGINurl="http://blookbank.flashcontacts.org/Home/GetBloodBanks?city=Jalandhar&bloo
dgroup="";

console.log(LOGINurl); fetch(LOGINurl) .then((response) => response.json())
.then((responseJson) => {   console.log("----"+JSON.stringify(responseJson));

   this.setState({ dataSource:responseJson  }); });

  }  render() {

 btncheckout = () => { const { userid, username } = this.state;

 console.log(userid);const
LOGINurl="http://blookbank.flashcontacts.org/Home/GetBloodBanks?city=&bloodgroup=";

 fetch(LOGINurl) .then((res) => res.json()) .then((data) => { // this.setState({ //
dataSource:data // })

<View style={{flex: 1, paddingTop:20}}>

 <FlatList   data={this.state.dataSource}

     renderItem={({item}) => {

const imgurl ='http://qrcode.flashcontacts.org/'+item.Image;

 return ( <View>

 <Textstyle={styles.info}>{item.City}{item.NumberOfUnits}{item.BloodGroup}  </Text>

    <View style={{flexDirection: "row"}}>

    <Buttonstyle={styles.image} title="Request Blood" onPress={() => btnplus(item.Id)}/>
```

```
        </View>  </View> )
          }}
          keyExtractor={(item, index) => index.toString()} />
  </View>
  </ScrollView>  );
      }
}
LinksScreen.navigationOptions = { title: 'Blood Banks', };
const styles = StyleSheet.create({
  container: { flex: 1,   paddingTop: 15,   backgroundColor: '#fff',  },
  image: { height:50,  width:50, alignItems: 'center',  justifyContent:'center',  } ,
  btnqty:
  { width:50,  margin:5,  } });
```

Main Tab Navigator:

```
import React from 'react';
import { Platform } from 'react-native';
import { createStackNavigator, createBottomTabNavigator } from 'react-navigation';
import TabBarIcon from '../components/TabBarIcon';
import HomeScreen from '../screens/HomeScreen';
import LinksScreen from '../screens/LinksScreen';
import SettingsScreen from '../screens/SettingsScreen';
import SignUpScreen from '../screens/SignUpScreen';
const config = Platform.select({ web: { headerMode: 'screen' }, default: {}, });
const HomeStack = createStackNavigator(
  {
Home: HomeScreen,
SignUp: SignUpScreen,

  }, config );

HomeStack.navigationOptions = {   tabBarLabel: 'Home',   tabBarIcon: ({ focused }) => ( <TabBarIcon  focused={focused}

name={ Platform.OS === 'ios' ? `ios-information-circle${focused ? " : '-outline'}`   : 'md-information-circle'  } /> ),

};

HomeStack.path = "; const LinksStack = createStackNavigator(

  {

    Links: LinksScreen,

  }, config );

LinksStack.navigationOptions = { tabBarLabel: 'Blood Banks',  tabBarIcon: ({ focused }) => ( <TabBarIcon focused={focused} name={Platform.OS === 'ios' ? 'ios-link' : 'md-link'} />  ),

};

LinksStack.path = ";

const SettingsStack = createStackNavigator( { Settings: SettingsScreen, },

  config );

SettingsStack.navigationOptions = {  tabBarLabel: 'Profile',

  tabBarIcon: ({ focused }) => (   <TabBarIcon focused={focused} name={Platform.OS === 'ios' ? 'ios-options' : 'md-options'} />  ),

};

SettingsStack.path = ";

const tabNavigator = createBottomTabNavigator({  HomeStack,  LinksStack,  SettingsStack, });

tabNavigator.path = ";

export default tabNavigator;
```

12. BIBLIOGRAPHY

https://facebook.github.io/react-native/docs/getting-started

https://github.com/facebook/react-native

https://react-native.org/doc/getting-started.html#content