Design Document

for

Central Blood Bank Management System

Version 1.0

Prepared by

Group:14 Group Name: Dead Inside

Nibir Baruah	19807545	nibir@iitk.ac.in
Adhiraj Sinha	190053	adhirajs@iitk.ac.in
Aditya Prakash	190066	prakasha@iitk.ac.in
Amit Kumar Singh	190117	amitsgh@iitk.ac.in
Harsh Patel	190363	harshp@iitk.ac.in
Anuj Chaudhary	190165	anujch@iitk.ac.in
Ayush Singh	190217	ayushsgh@iitk.ac.in
Manish Mayank	190482	manmay@iitk.ac.in
Kartik Jhanwar	190418	kartikjh@iitk.ac.in
Nikhil Mehta	190549	nikhilme@iitk.ac.in
Nishima Panwar	190562	niship@iitk.ac.in

Course: CS253

Mentor TA: *Ashutosh*

Date: 10-02-2023

CONTENTS II

REVISIONS II

1 CONTEXT DESIGN 4

1.1 CONTEXT MODEL 4

1.2 HUMAN INTERFACE DESIGN 4

2 ARCHITECTURE DESIGN 5

3 OBJECT-ORIENTED DESIGN 7

3.1 USE CASE DIAGRAM 7

3.2 CLASS DIAGRAM 10

3.3 SEQUENCE DIAGRAM 11

3.4 STATE DIAGRAM 15

4 PROJECT PLAN 17

5 OTHER DETAILS 20

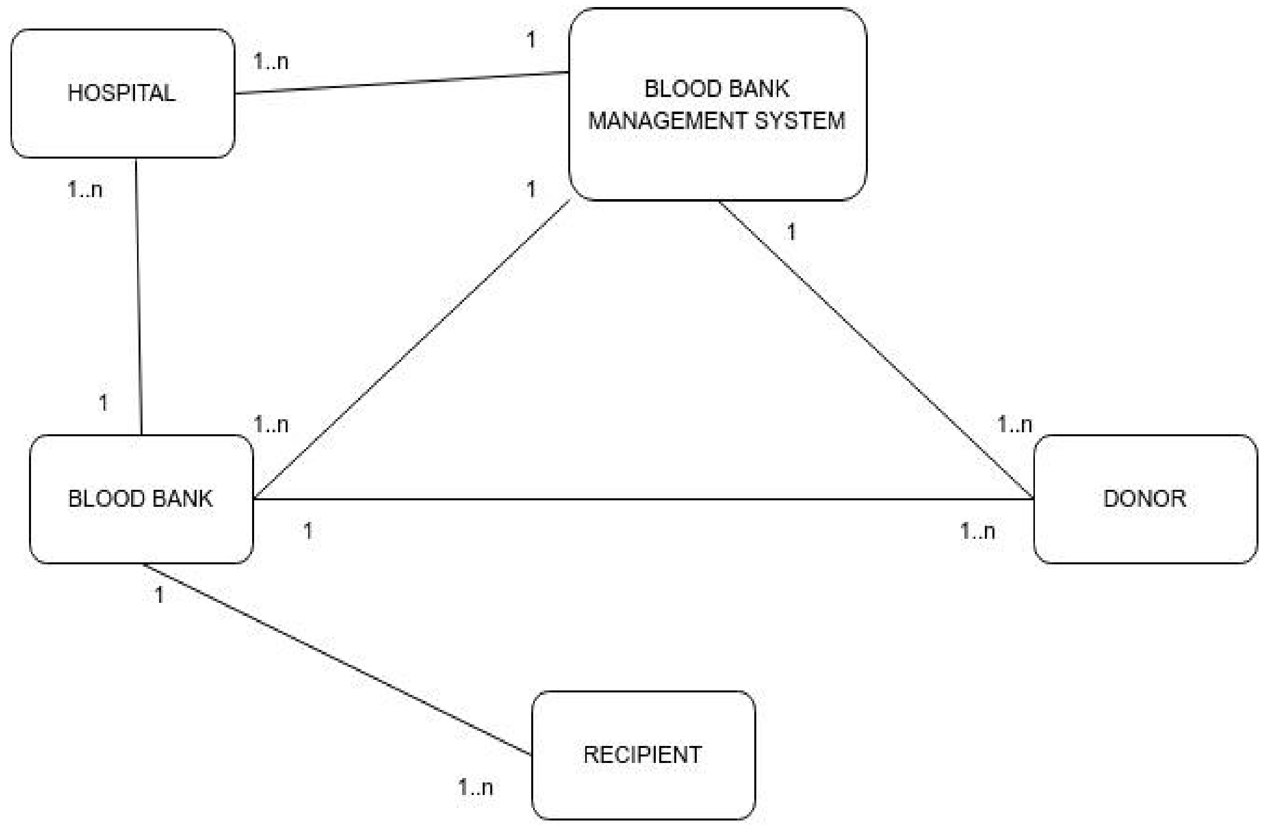
APPENDIX A - GROUP LOG 21

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Nibir Baruah Aditya Prakash Amit Kumar Singh Harsh Patel Anuj Chaudhary Ayush Singh Manish Mayank Kartik Jhanwar Nikhil Mehta Nishima Panwar Adhiraj Sinha	Initial Draft	10/02/23

1 Context Design

1.1 Context Model



1.2 Human Interface Design

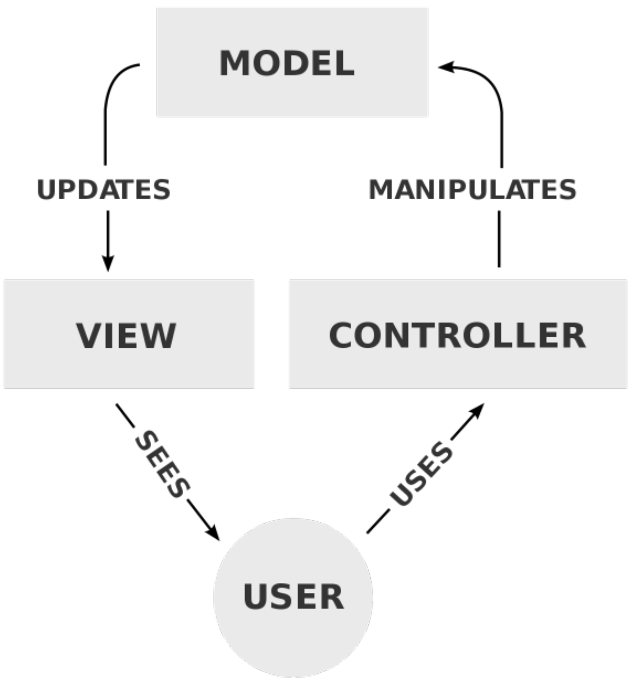
Donors: Interface for the interaction of donors with the software. This interface provides functionality to select a particular state and city. After selecting, this will provide all the information about the blood camps available in that city.

Looking for blood: Interface for the interaction of people who are looking for blood with the software. This interface provides functionality to select state, city, blood group and blood components(RBC, Platelets, Plasma, Cryo AHF, Granulocytes). After selecting, this will provide all the information about the available blood and its quantity with the blood bank name and its location.

Blood Bank Login: Interface for the interaction of blood bank with the software. This interface provides for sign-up for new blood banks to register. After successfully logging in gives access to the features like viewing and updating blood details currently available.

2 Architecture Design

WHAT?



The architecture of the system can be thought of as **Model-View-Controller.**
It is a software architectural pattern commonly used for developing user interfaces that divide the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted by the user. In MVC application development, the controller accepts all application requests and advises the model to prepare any data requested by the view. The view brings the final visual output using the data produced by the controller.

WHY?

The pattern of the architecture was determined not by pure chance, but by eliminating other possibilities. Our previous knowledge provided us with other kinds of architectural patterns: Layered, Repository, Client-Server, Pipeline, and Bus.

1. The **repository architecture** would have only divided our software into components, and would not have depicted any kind of functionalities or inter-component interactions, and hence was ruled out.
2. **Pipeline system** was again a bad choice for the pattern since our software did not have a stepwise input processing mechanism.
3. **Bus architecture** was again a problem due to the lack of a bus system in our software.

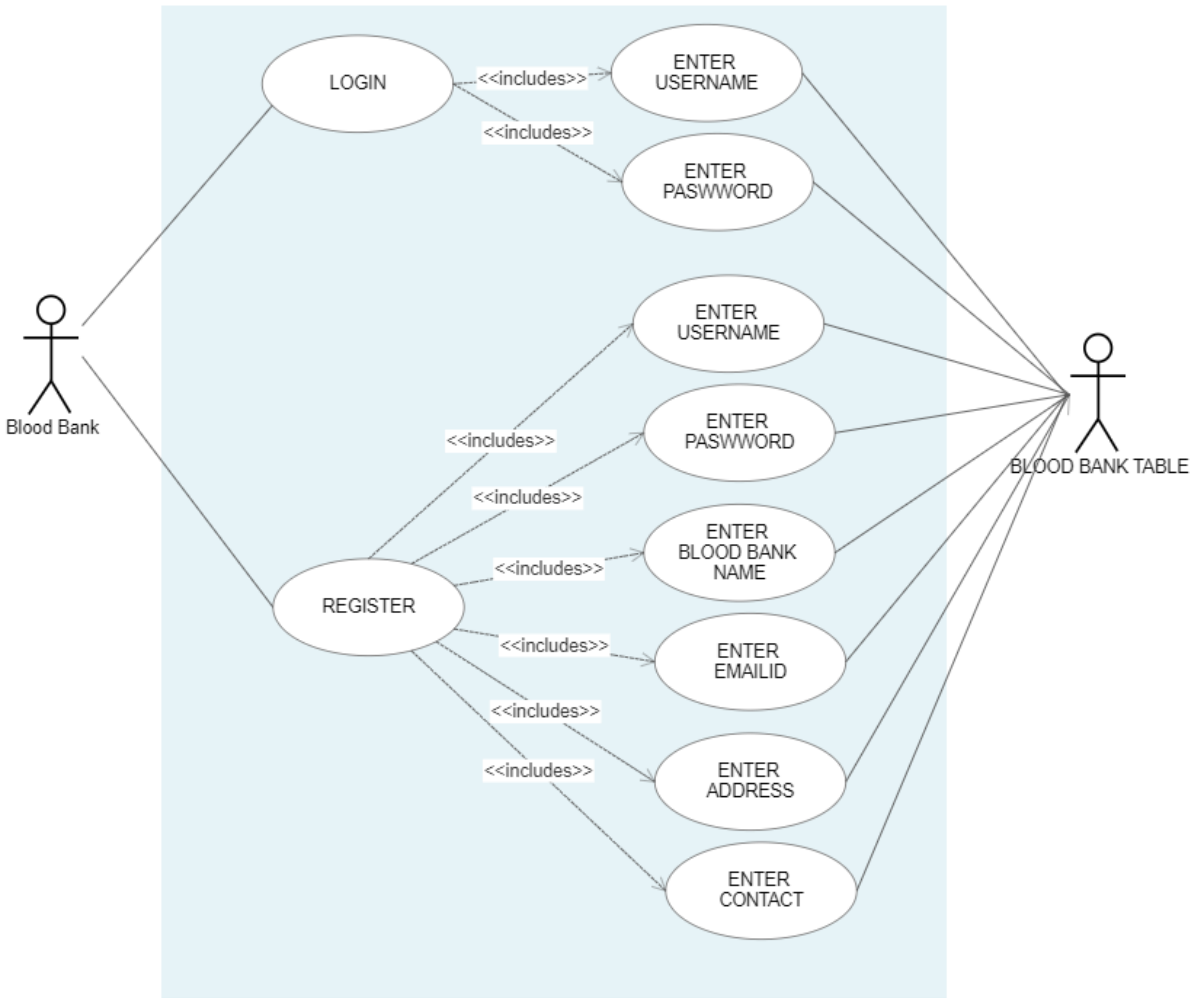
Model-View-Controller was chosen for the following reasons:-

- There is only one kind of user - Blood Bank.
- Faster development process
- Facilitate multiple views, reducing code duplication by separating data and business logic
- supports test-driven development(TDD)
- easier to divide and arrange web application functionality into large-scale apps.
- The MVC pattern is helpful during the application’s early design phase

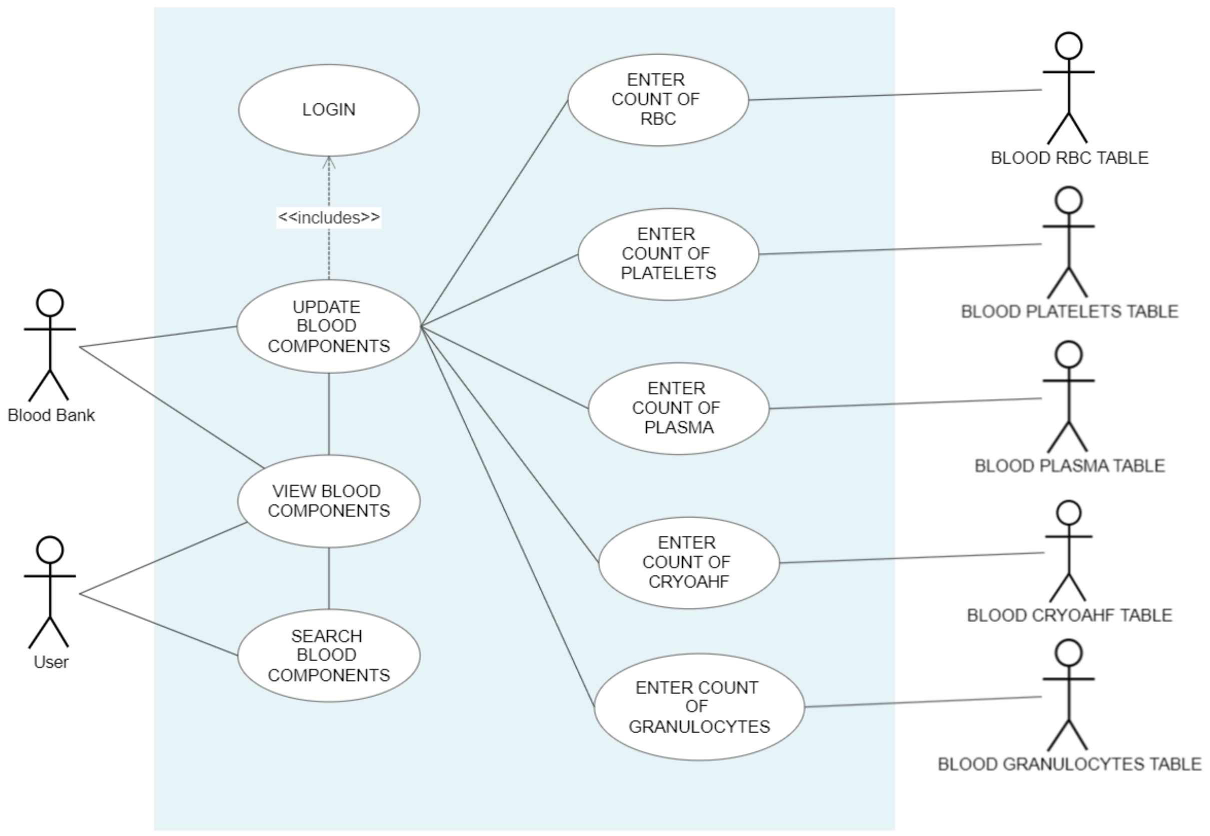
3 Object Oriented Design

3.1 Use Case Diagrams

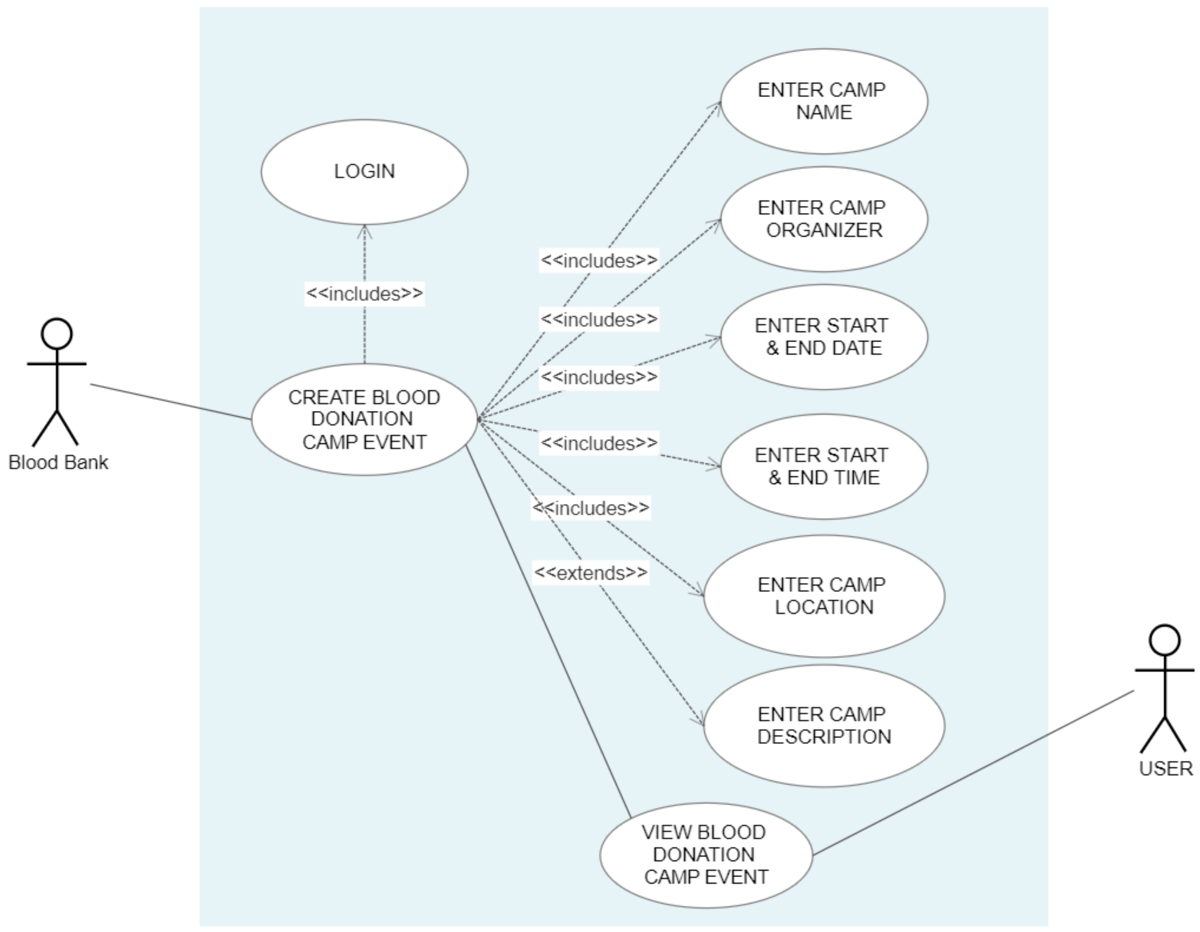
LOGIN & REGISTER



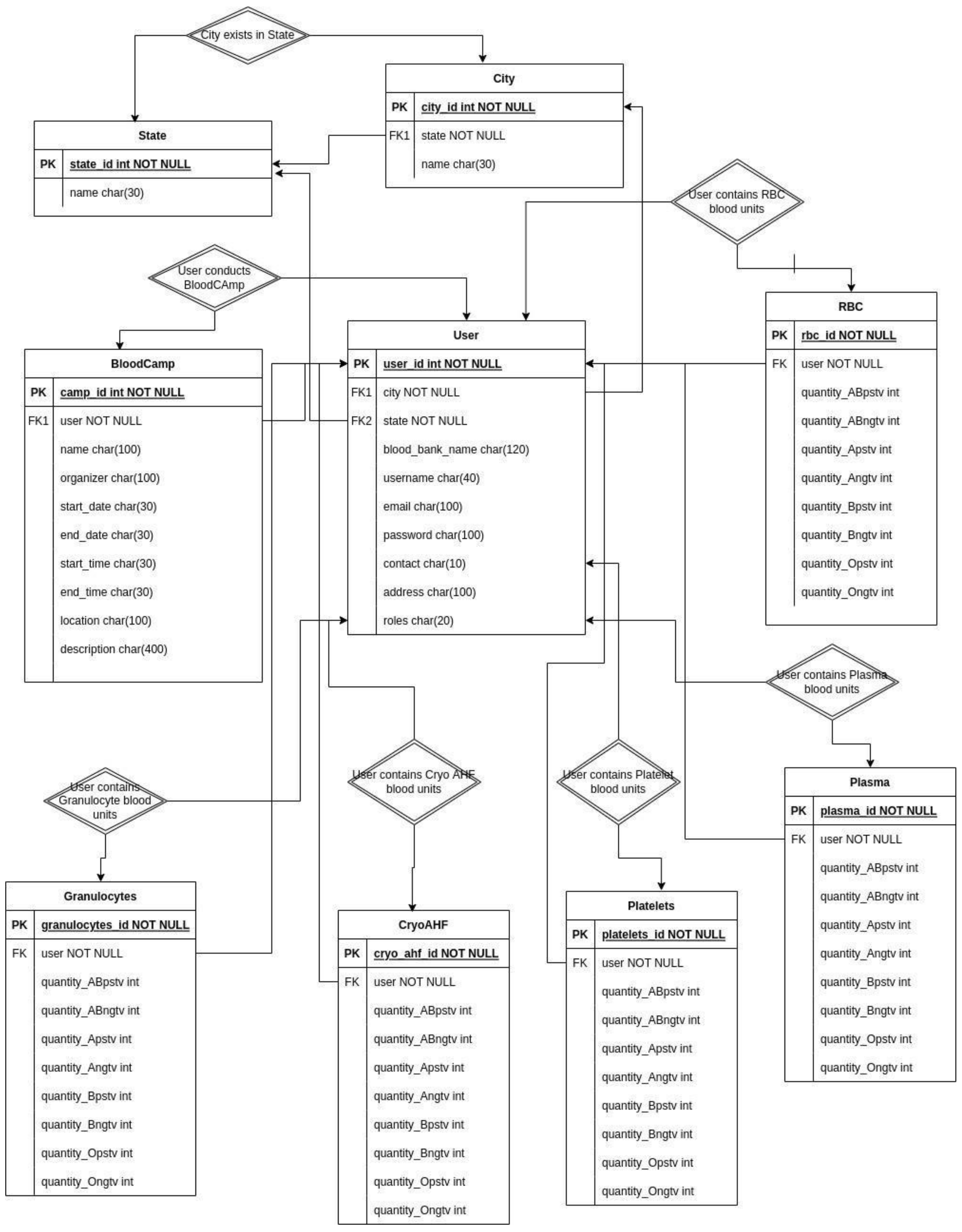
UPDATE & VIEW BLOOD COMPONENTS COUNT



VIEW & CREATE BLOOD DONATION CAMP EVENT

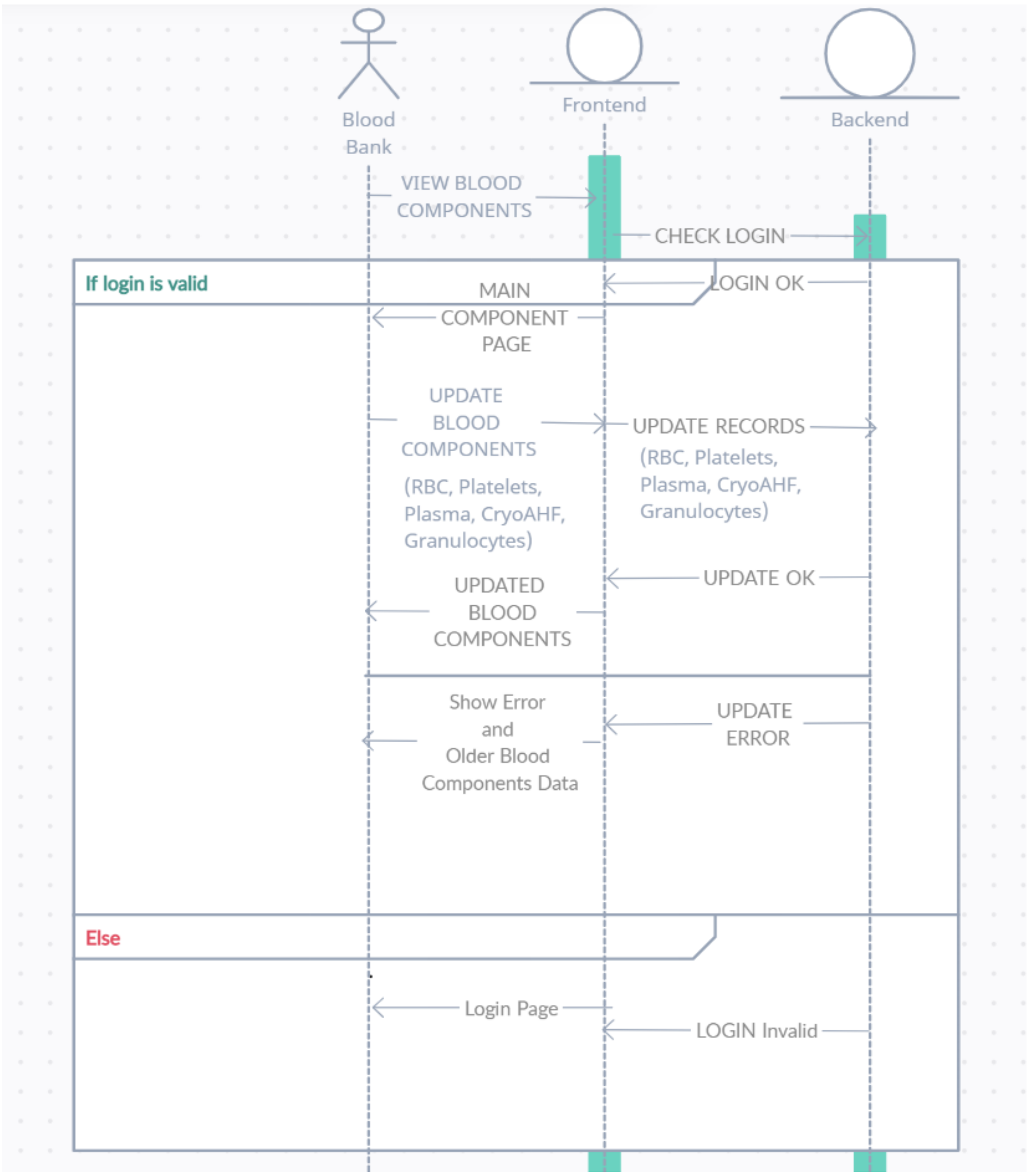

LOGIN
<<includes>>
Blood Bank
CREATE BLOOD DONATION CAMP EVENT
ENTER CAMP NAME
<<includes>>
ENTER CAMP ORGANIZER
<<includes>>
ENTER START & END DATE
<<includes>>
ENTER START & END TIME
<<includes>>
<<includes>>
ENTER CAMP LOCATION
<<extends>>
ENTER CAMP DESCRIPTION
VIEW BLOOD DONATION CAMP EVENT
USER

3.2 Class Diagrams

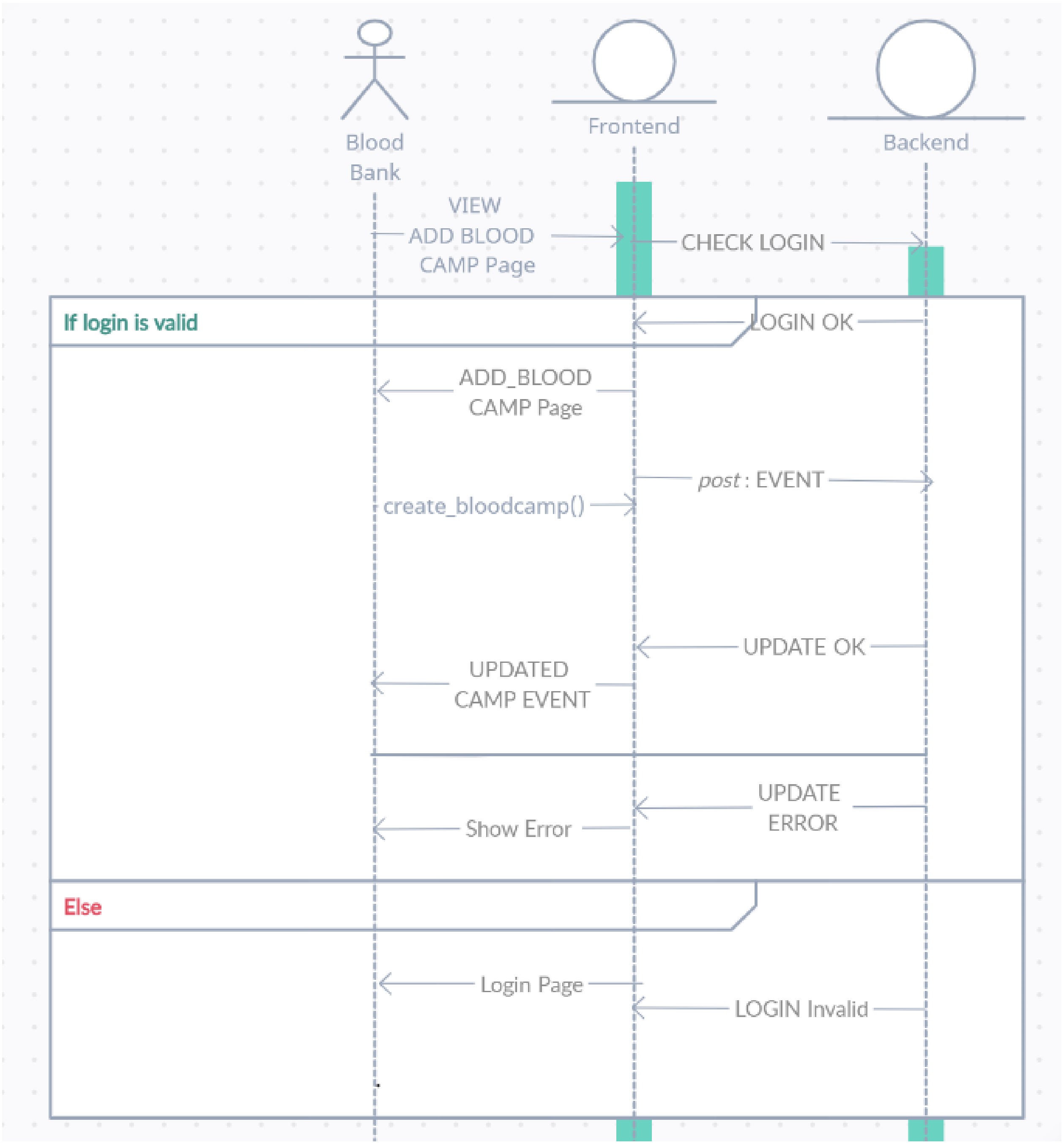


3.3 Sequence Diagrams

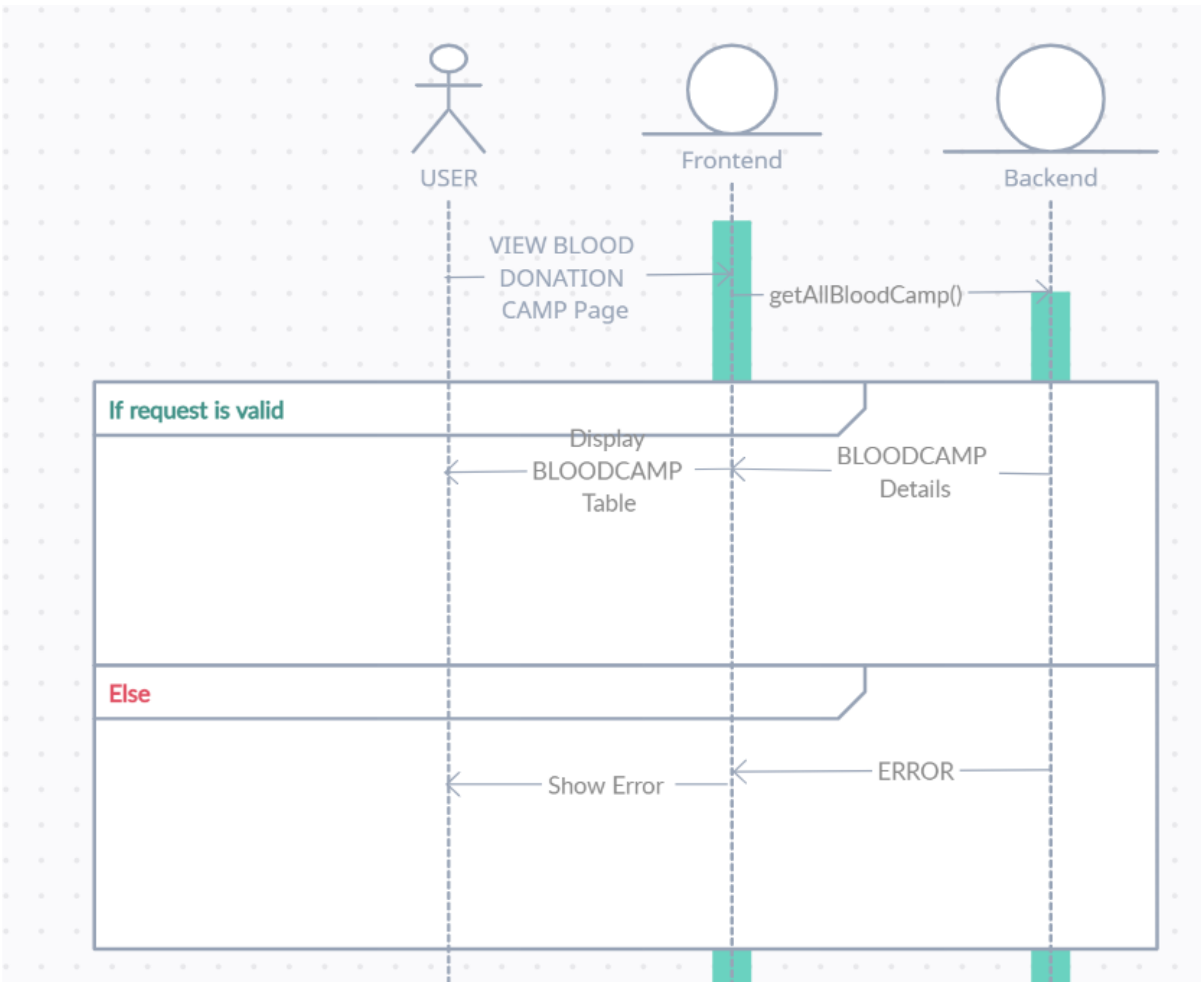
VIEW AND UPDATE BLOOD COMPONENTS


Blood Bank
Frontend
Backend
VIEW BLOOD COMPONENTS
CHECK LOGIN
If login is valid
LOGIN OK
MAIN COMPONENT PAGE
UPDATE BLOOD COMPONENTS (RBC, Platelets, Plasma, CryoAHF, Granulocytes)
UPDATE RECORDS (RBC, Platelets, Plasma, CryoAHF, Granulocytes)
UPDATE OK
UPDATED BLOOD COMPONENTS
Show Error and Older Blood Components Data
UPDATE ERROR
Else
Login Page
LOGIN Invalid

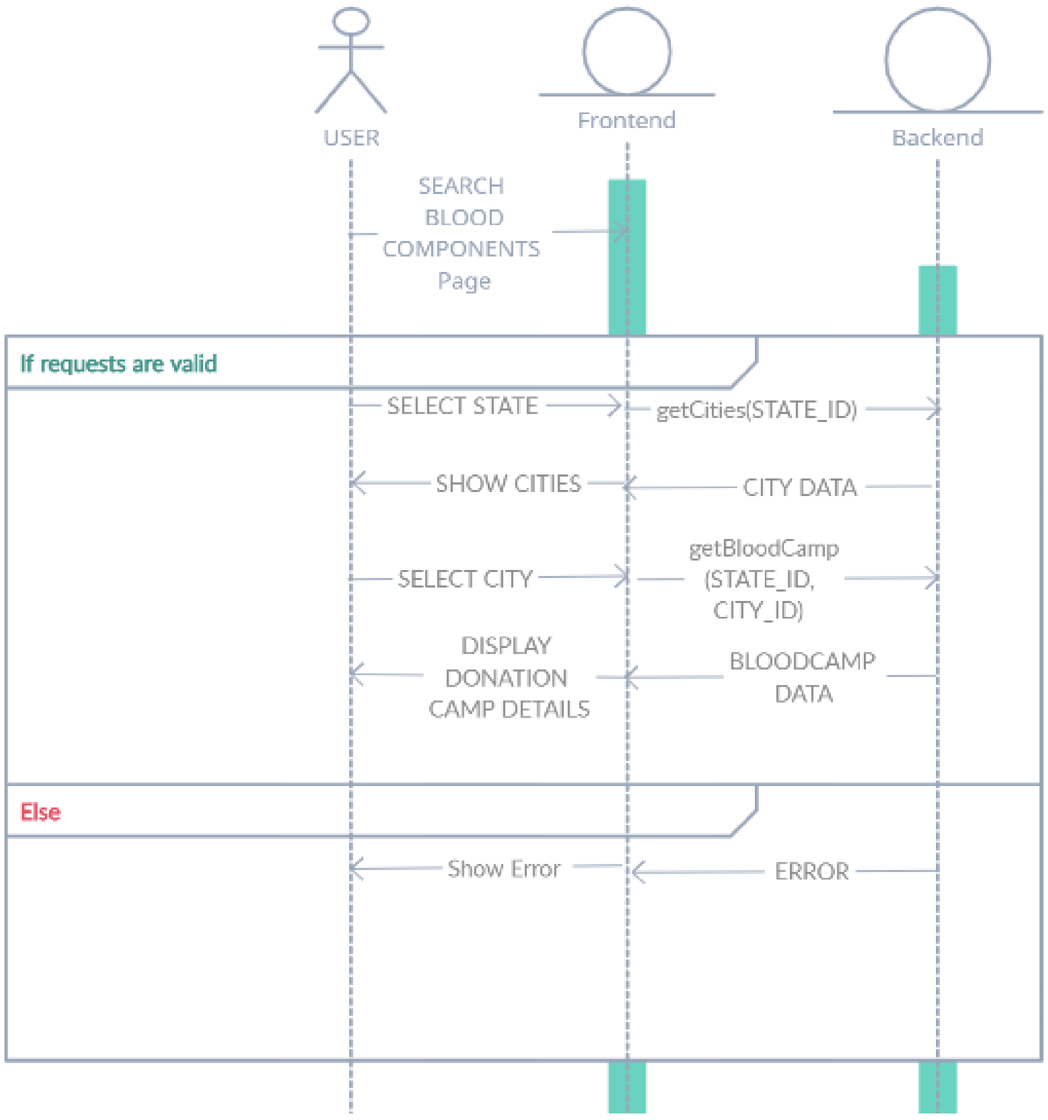
CREATE BLOOD DONATION CAMP



VIEW BLOODCAMP

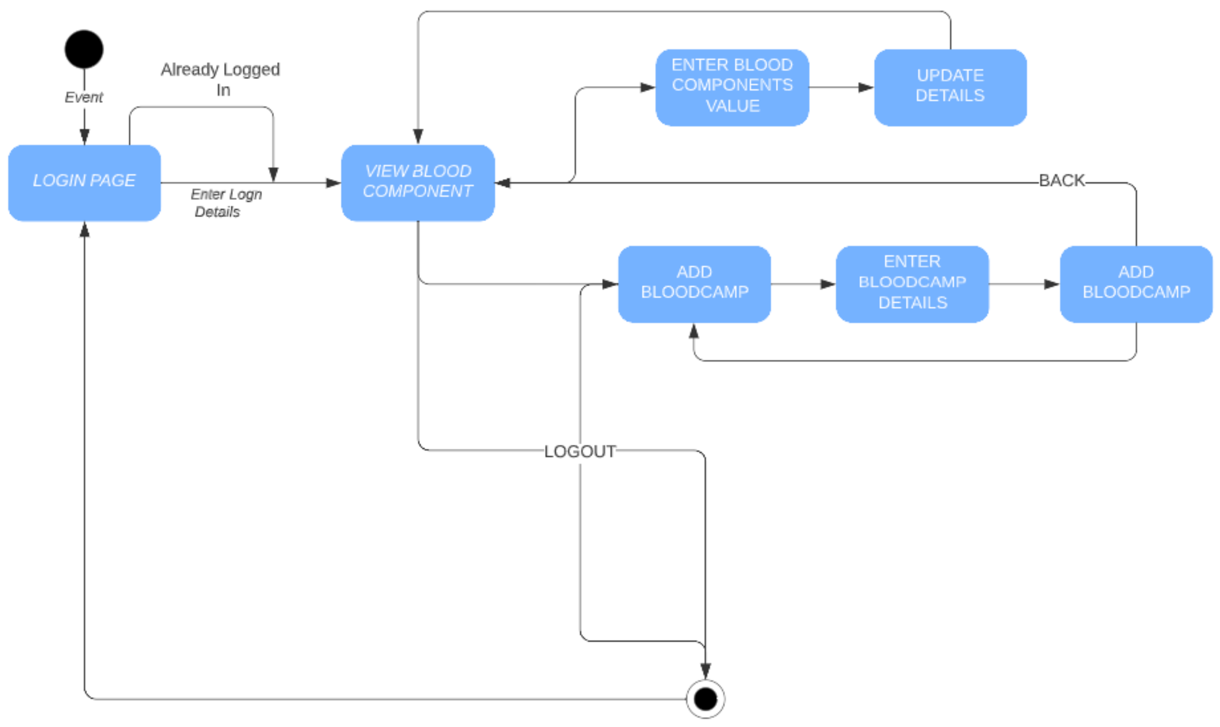


SEARCH BLOOD COMPONENTS

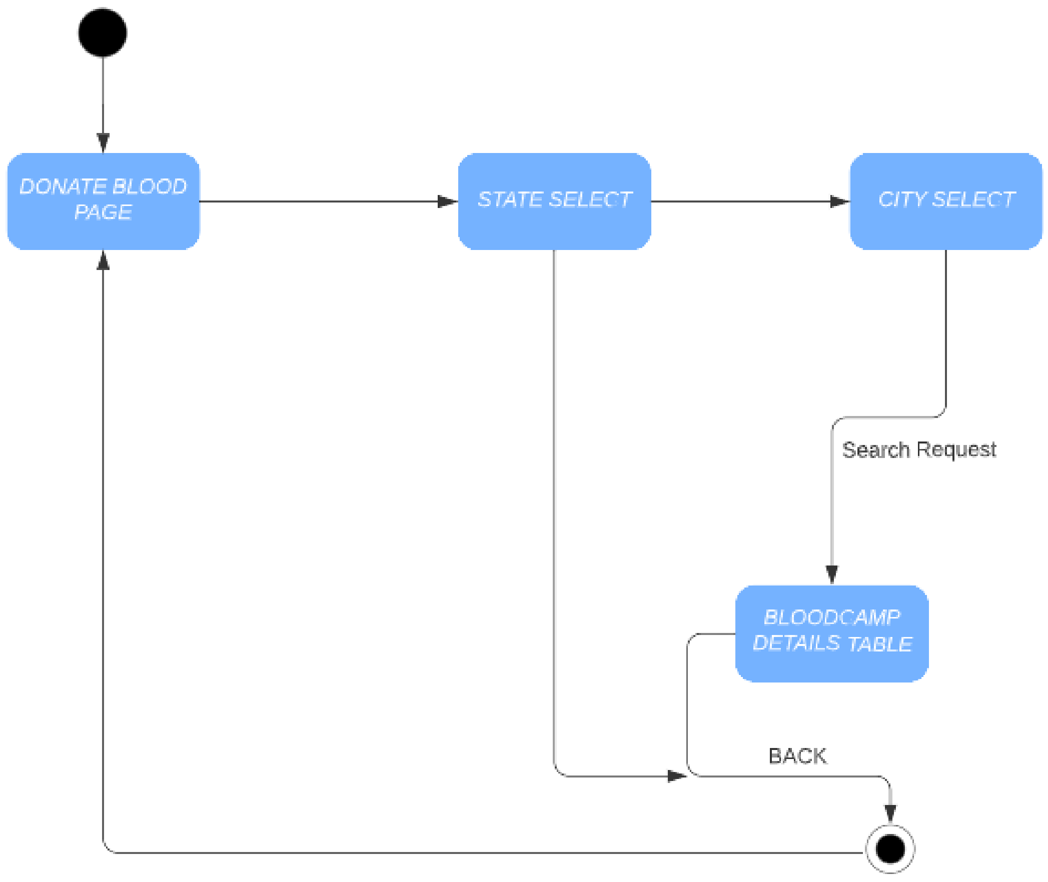

USER
Frontend
Backend
SEARCH BLOOD COMPONENTS Page
If requests are valid
SELECT STATE
getCities(STATE_ID)
SHOW CITIES
CITY DATA
SELECT CITY
getBloodCamp (STATE_ID, CITY_ID)
DISPLAY DONATION CAMP DETAILS
BLOODCAMP DATA
Else
Show Error
ERROR

3.4 State Diagrams

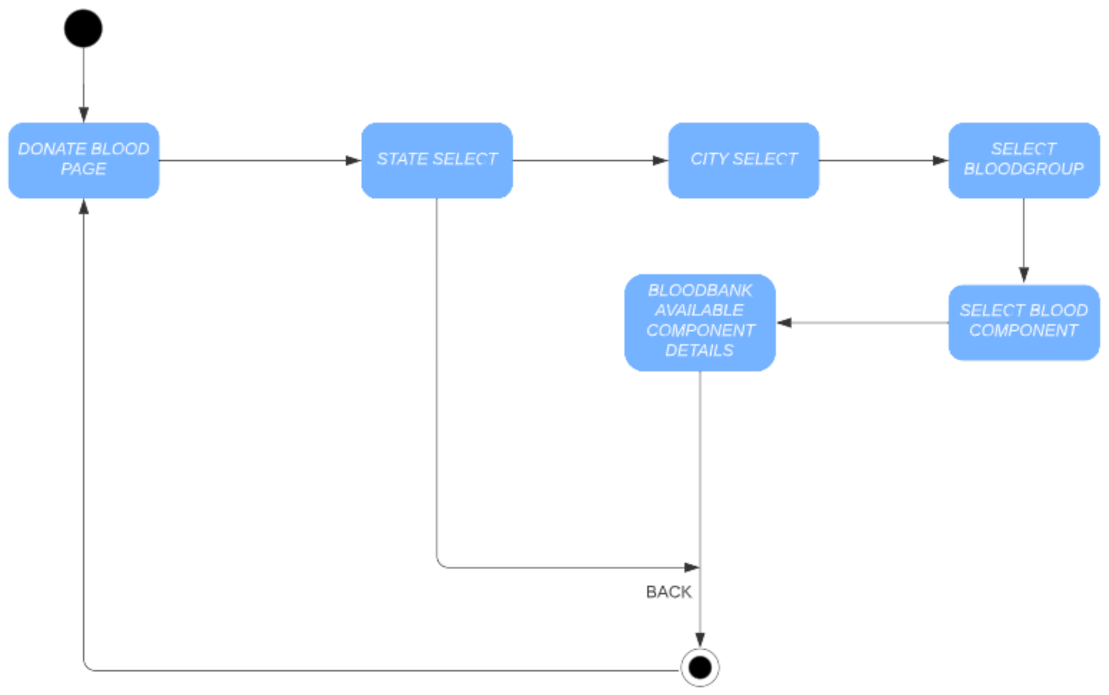
BLOOD BANK



DONOR


DONATE BLOOD PAGE
STATE SELECT
CITY SELECT
Search Request
BLOODCAMP DETAILS TABLE
BACK

RECIPIENT & HOSPITAL/LOOKING FOR BLOOD



4 Project Plan

1. **Setting Up the Source Control: to manage changes of our code over time**
   Software Used: **Git**
   We need to manage changes in our code over time, through useful operations such as creating alternative copies (*branches*) on which we can work without touching the other copies, reverting the code to previous states (*commits*) whenever needed.
   Source control gives us granular access to our project's code (*repository*) at certain points in its development, allowing for more detailed editing.

2. **Remote repository: GitHub**
   GitHub is a hosting service for Git repositories. We can **host a copy of our repository online,** and it provides **powerful collaboration tools**.

3. **IDE: VSCode / SublimeText**

Front End Development Workflow

- **Wireframing** is a set of screen images showing the total structure of the user interface. At the same point, a so-called storyboard for the interrelations between individual screens is also built. A storyboard helps to see whether all interface elements have been planned and whether all connections are clear and logical
- **UI design** - Actual UI screens will be created in the form of mockups. The mockups are prototypes of the application UI and will be used in further UI programming
- **UI development-** The code corresponding to the UI design is written

Backend Development Workflow

- **Database management-** Data storage setup includes configuring databases serving the application
- **User management-** defining the methods of user authentication and access and the security measures to be implemented in the app
- **Server-side logic-** representing the flows executed by the application server and the procedures for handling user's requests sent from the UI
- **Data integration-** enabling data exchange between different resources including third-party ones

Software Testing Life Cycle (STLC)

Software Testing Life Cycle is a sequence of different activities performed by the testing team to ensure the quality of the software or the product. It defines a series of activities conducted to perform Software Testing.

Different Phases of STLC:-

- **Requirement Analysis:-** In this step, we understand the requirements for what we will be testing & figure out all the testable needs.

- **Test Planning:-** In this Phase of Testing, we define the objective and scope of the project.

- **Test Case development:-** Test cases and dummy data for Testing are prepared in this testing phase. Other team members review the test cases before Testing.

- **Test Execution:-** It executes the code on the test cases with a prepared data set and then compares the expected and actual results.

- **Test Cycle Closure:-** It is the last Phase of STLC; it involves meeting team members and evaluating cycle completion criteria based on Test coverage that our software system fulfils all the requirements mentioned in the required document.

Functional Testing

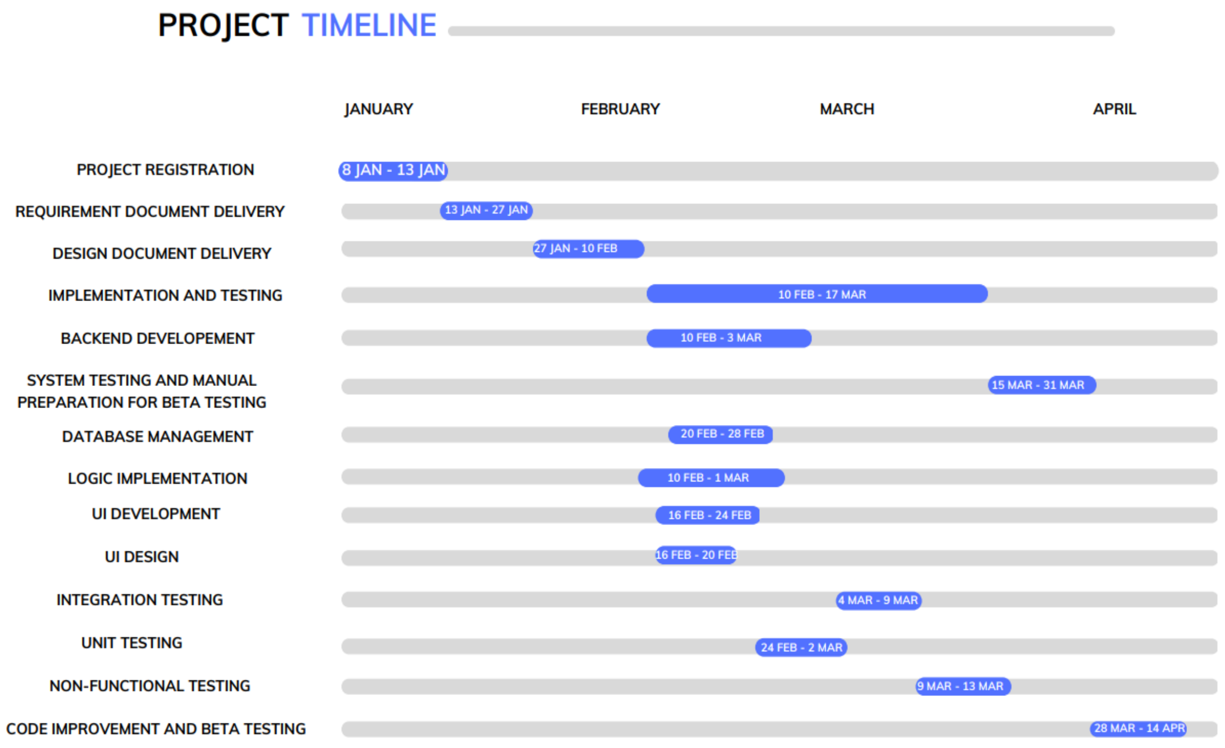
1. **Unit Testing:-** In this level of Testing, we test the individual units /components of the software. Each module should work as intended. It usually has a few inputs and a single output. All developers will do unit testing of the modules they have written themselves.
2. **Integration Testing:-** At this level, we will test different combined and tested modules as a group. This level of Testing exposes faults in interaction between different individual units and ensures that all modules interact with other modules as intended.
3. **System Testing:-** In this level of testing, complete and integrated software is tested. This testing checks if the system complies with the specified requirements.
4. **Interface Testing:-** In this testing, we test the connection which integrates and facilitates the communication between these components like server, database, etc. It verifies that communication between the systems is done correctly.

Non-Functional Testing:-

1. **Performance Testing:-** This Testing will evaluate the system's behavior at an increasing workload. It will ensure that the system will not lag when it increases load.
2. **Reliability Testing:-** This testing ensures that the product is reliable for its intended purpose. It is about exercising an application to discover failures before the system is deployed.
3. **Security Testing:-** This testing ensures that the system is free from loopholes. No user should be able to see any other user's data. Lab or Pharmacy should only see the data concerning their department. Admin should not be able to see the user's medical history.

S.No.	Major Tasks	Assigned To
1	Wireframing	Amit Kumar Singh
2	UI Design	Adhiraj Sinha
3	UI Development	Harsh Patel
4	Database Management	Nibir Baruah
5	User Management	Aditya Prakash

6	Server-side-logic	Anuj Chaudhary
7	Data Integration	Kartik Jhanwar
8	Unit Testing	Manish Mayank
9	Integration Testing	Ayush Singh
10	System Testing	Nikhil Mehta
11	Interface Testing	Nishima Panwar



5 Other Details

<*This section is <u>Optional</u>. Please provide any other details that are suitable for being included in the design document .*>

Appendix A - Group Log

<*Please include here all the minutes from your group meetings, your group activities, and any other relevant information that will assist in determining the effort put forth to produce this document*>

Had 2 online zoom meetings to discuss various work assigned to group members. All group members contributed equally, according to the work assigned to them.