CS6106 – DATABASE MANAGEMENT SYSTEM

Feb - May 2021

BLOOD BANK MANAGEMENT SYSTEM

Harish J- 2019103524

Bharat Kumar DP- 2019103512

TABLE OF CONTENTS:

ABSTRACT:

The Problem we address is the lack of blood needed for surgeries,Cancer treatment,chronic illnesses and other injuries. People with Sickle Cell Anemia need blood at regular intervals throughout their lives. Every 2 seconds someone in India needs blood and the ground truth there isn't enough to provide them. This Lifesaving care starts from a donation that we all make. And Blood can be stored only for maximum 36 days and so this isn't a onetime process. It needs to be followed and practiced throughout.

Blood requests aren't organized and many people aren't receiving the blood at proper time. Most people resort to asking their friends, sharing it in social media to get help. If they don't get it resolved within a time period, its very unfortunate.

To speed up and organize this process and to ensure everyone in need of blood gets the right attention and resource, we provide a Common interface to request for blood and also encourage people to donate blood.

INTRODUCTION:

A portal to manage Blood donation and transfusion processes in a Blood bank. A common interface between Hospitals, Blood banks, Registered donors to speed up the process of blood transfusion and to create social awareness. Blood donation cycle will be transparent and turnaround time will be reduced. If there is a lack of blood samples in Blood bank, the blood banks nearby the location will be contacted. To encourage donors, points and badges will be provided which can be redeemed for Free medical checkups in the respective hospital. Blood

banks can create **Camps** to collect Blood which will be advertised to the Users.

TOOLS USED:(TECH STACK)

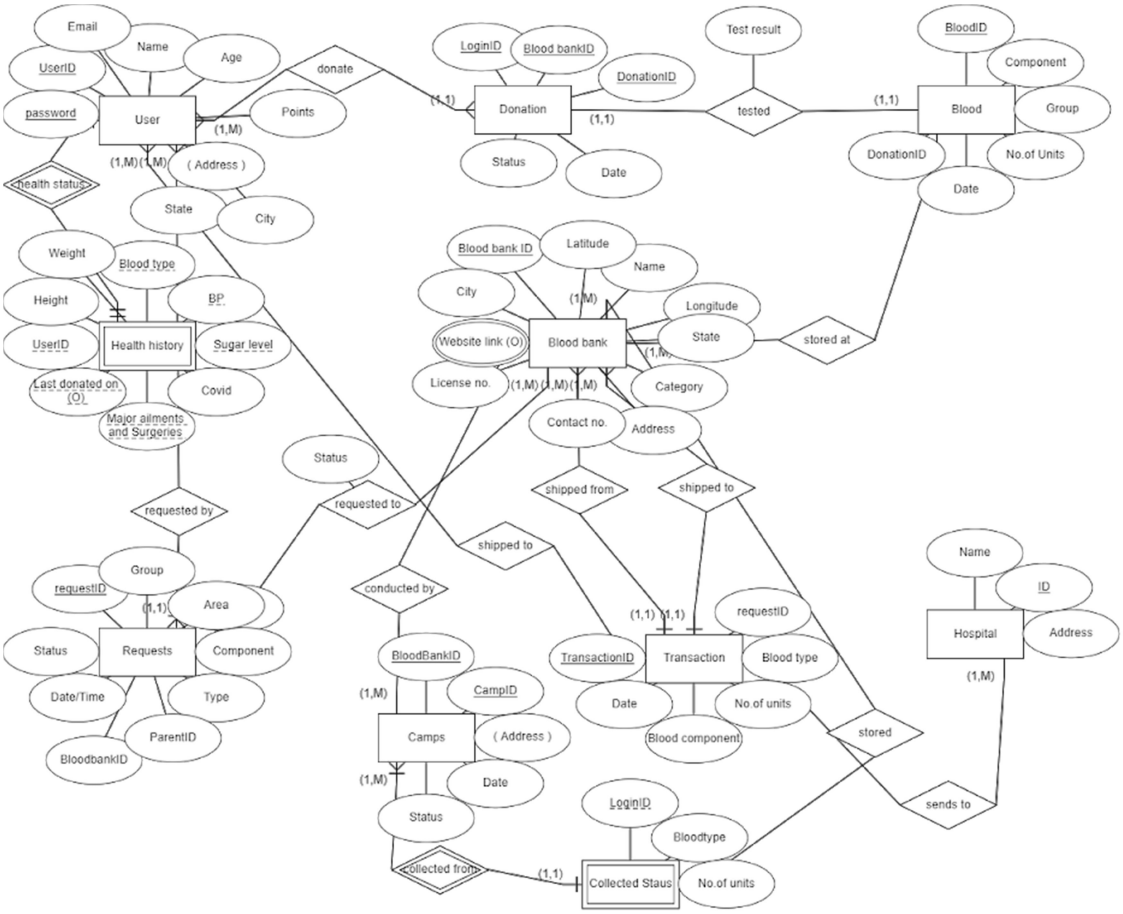
Front end: HTML,CSS,Java script

Back end: Node js 14.4.0,Express framework

Database: MySQL

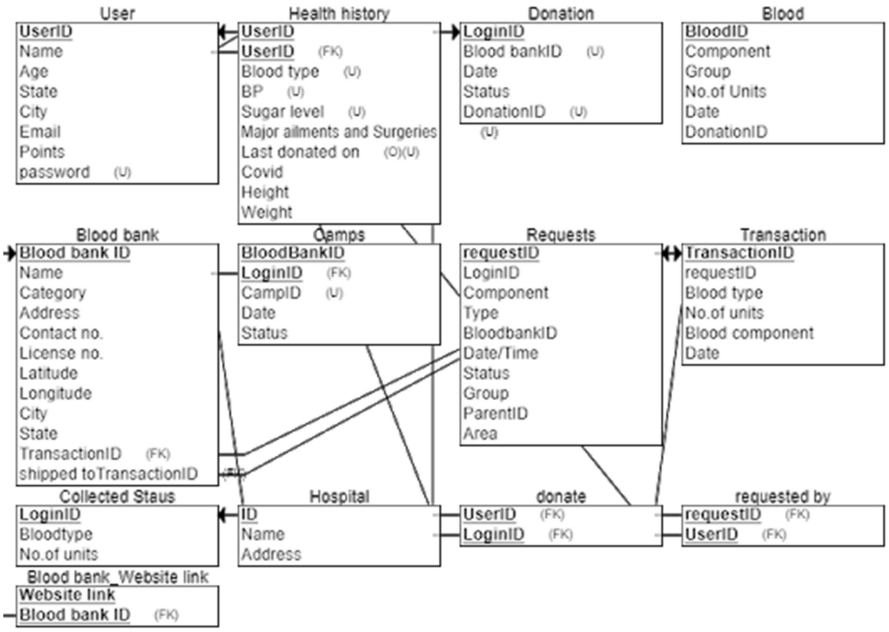
ER AND RELATIONAL SCHEMA:

ER diagram:



Relational schema:

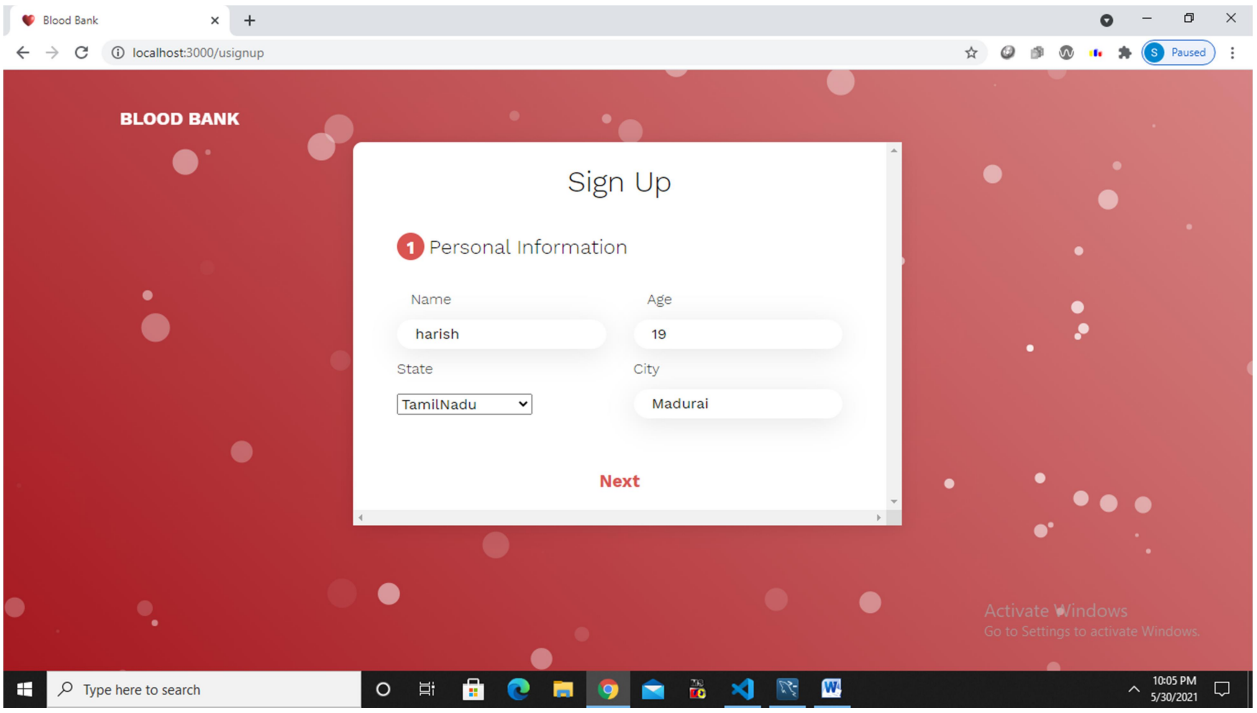
All relations are BCNF normalized.

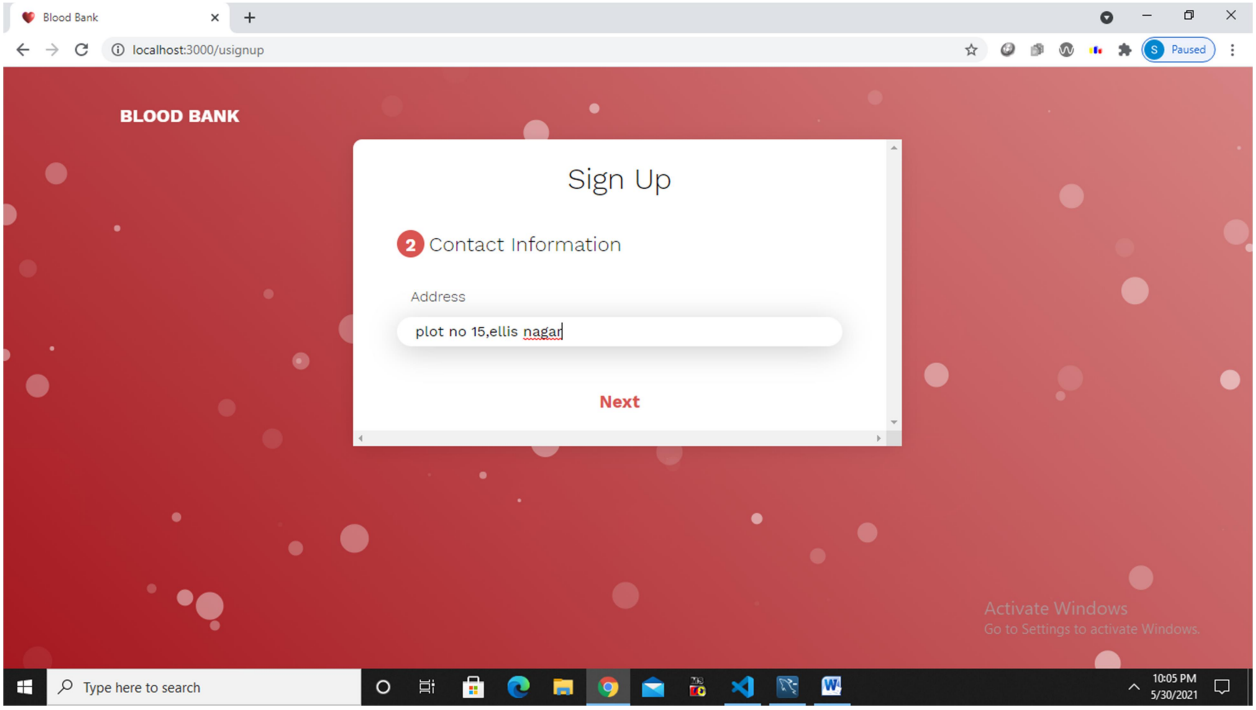


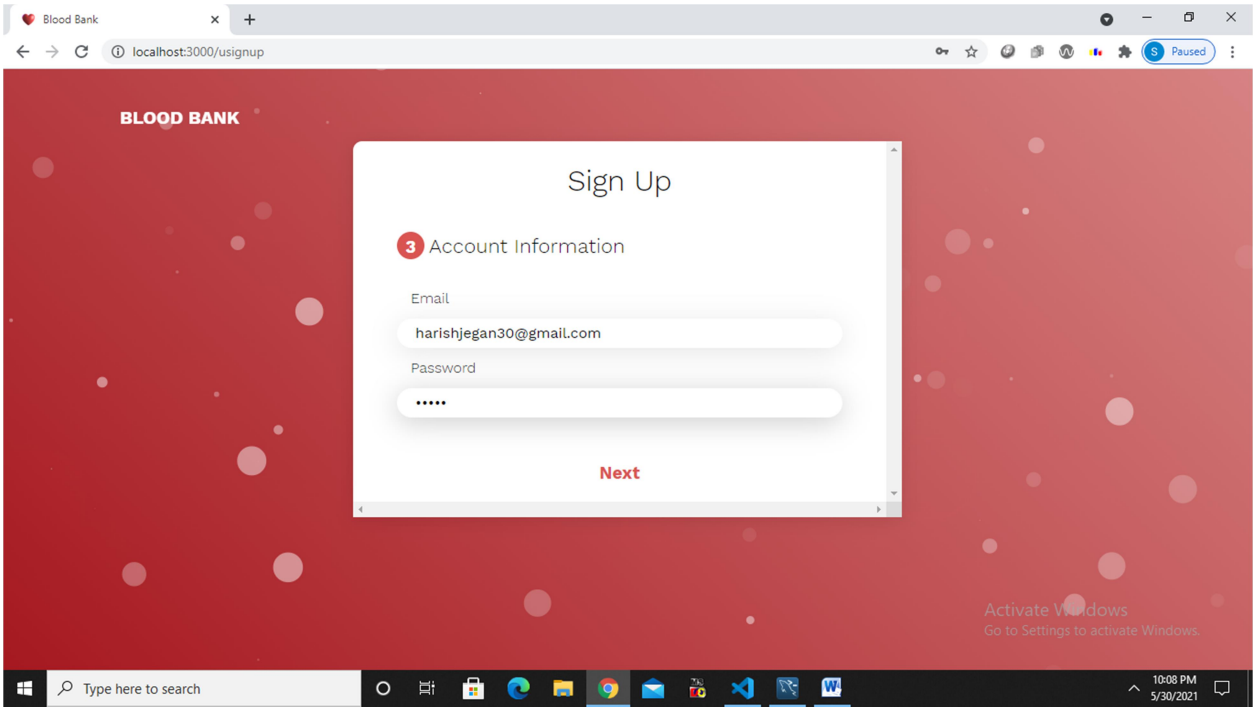
REGISTRATION AND LOGIN:

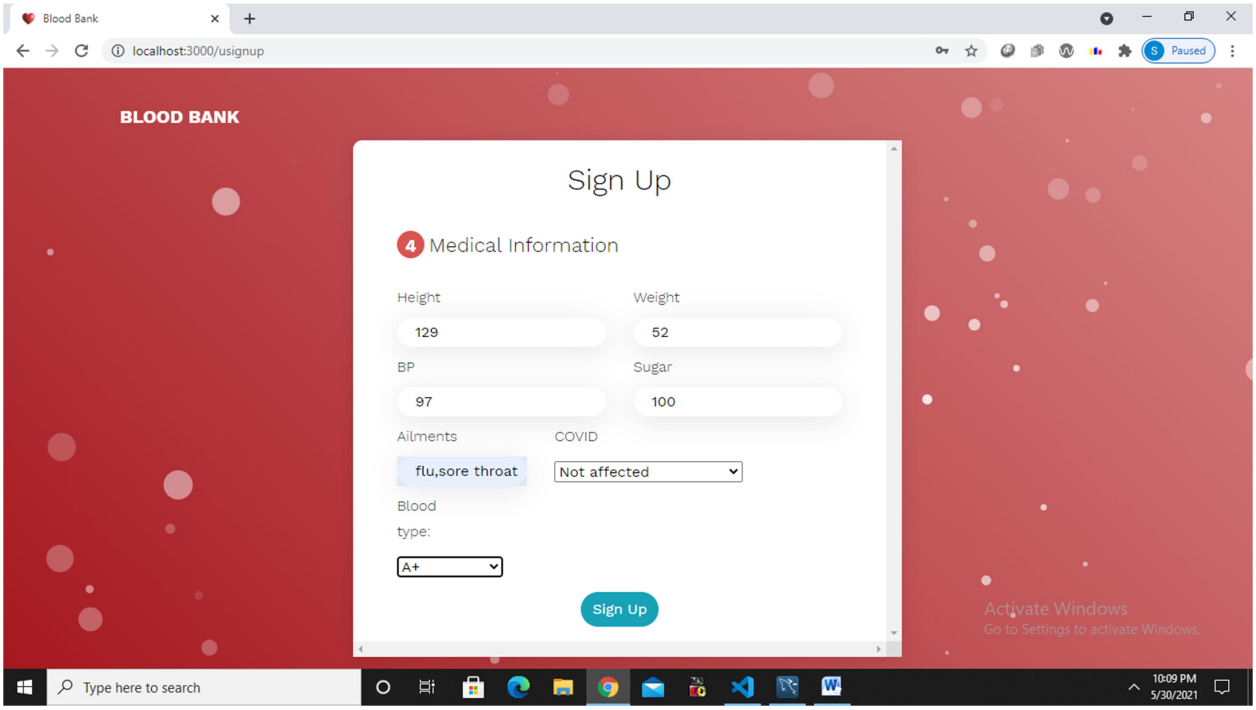
User side:

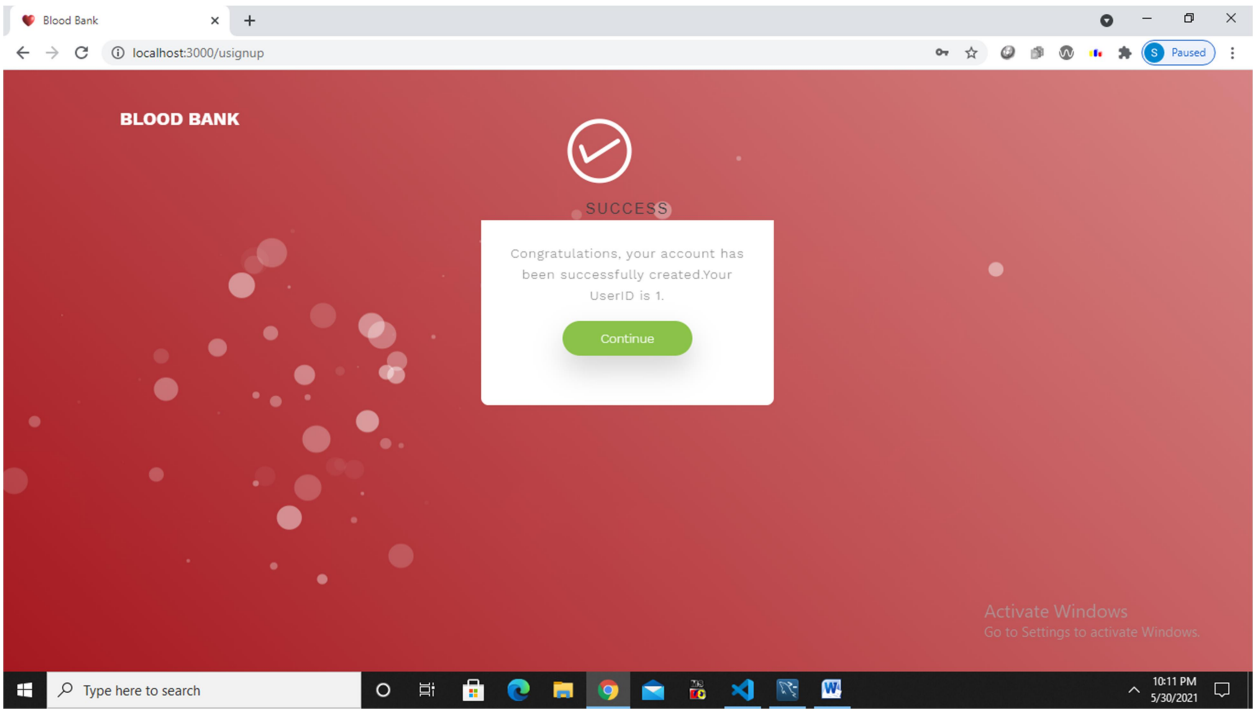
Sign up procedure:




Blood Bank
localhost:3000/usignup
Paused
BLOOD BANK
Sign Up
2 Contact Information
Address
plot no 15,ellis nagar
Next
Activate Windows
Go to Settings to activate Windows.
Type here to search
10:05 PM
5/30/2021

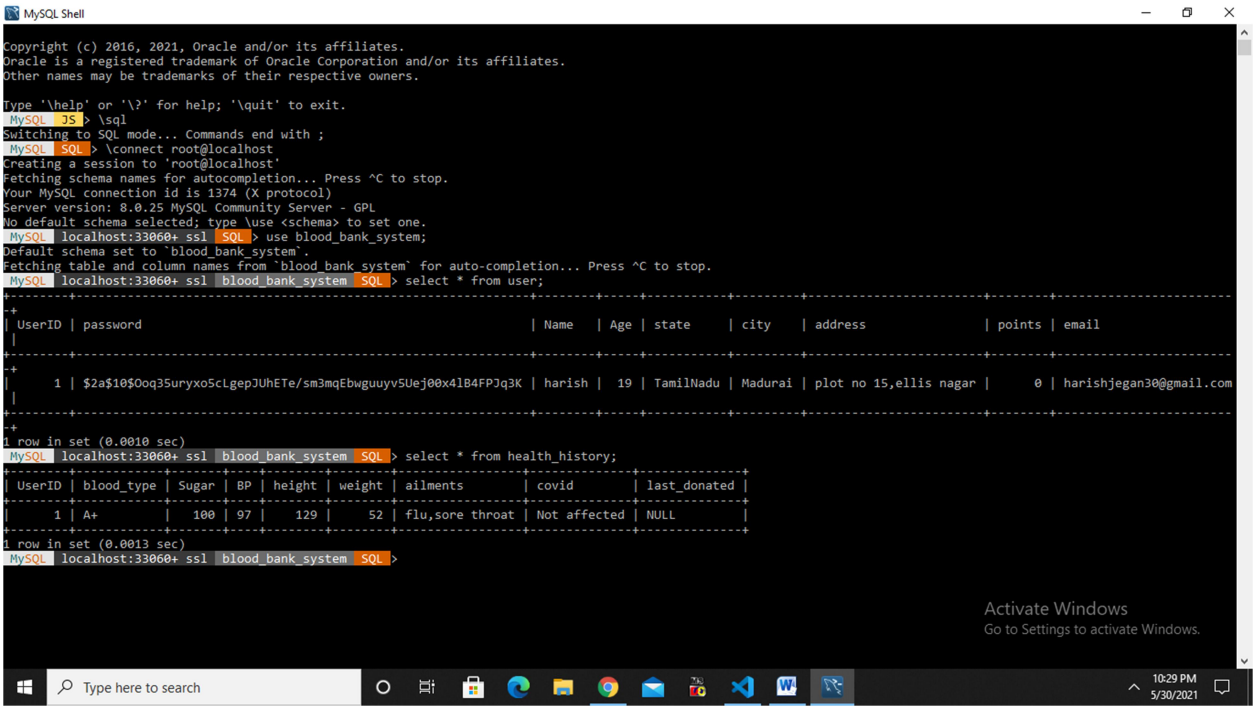

Blood Bank
localhost:3000/usignup
Paused
BLOOD BANK
Sign Up
3 Account Information
Email
harishjegan30@gmail.com
Password
.....
Next
Activate Windows
Go to Settings to activate Windows.
Type here to search
10:08 PM
5/30/2021


Blood Bank
localhost:3000/usignup
Paused
BLOOD BANK
Sign Up
4 Medical Information
Height
129
Weight
52
BP
97
Sugar
100
Ailments
flu,sore throat
COVID
Not affected
Blood type:
A+
Sign Up
Activate Windows
Go to Settings to activate Windows.
Type here to search
10:09 PM
5/30/2021

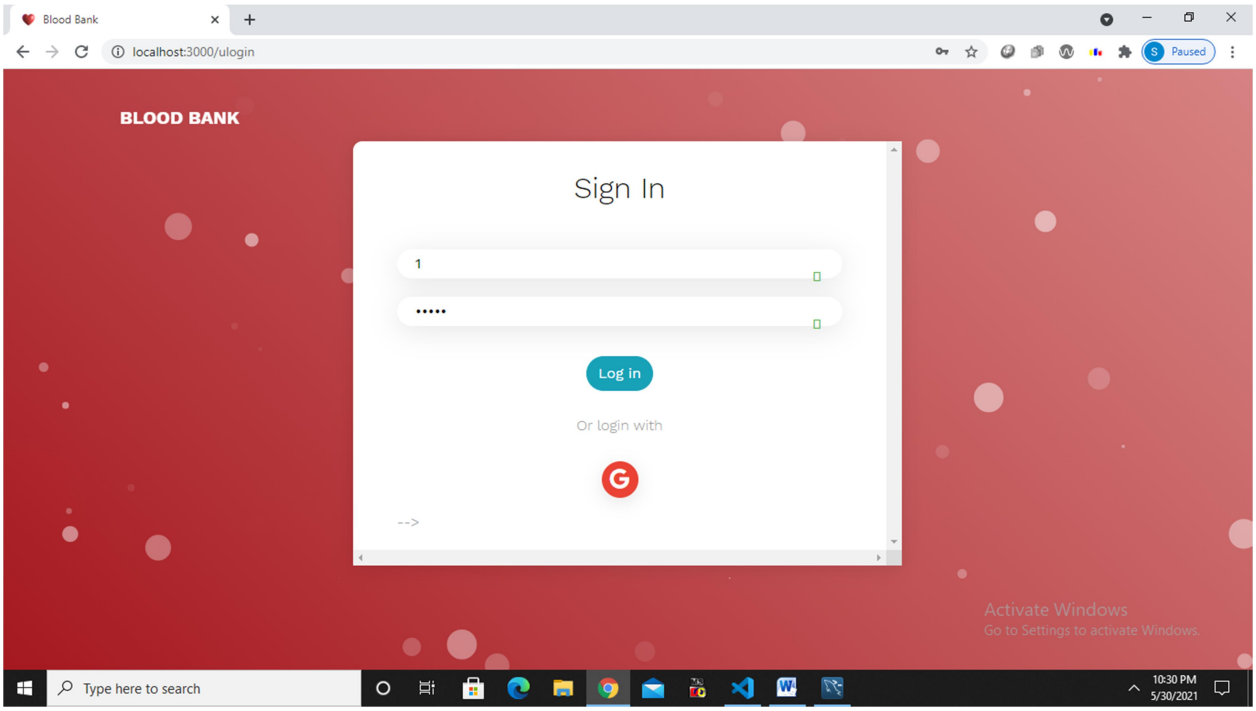


A new user has filled up the details in the signup form

and a success message is displayed showing the userid assigned to him.The corresponding details of the user are also stored in the user and health_history table.

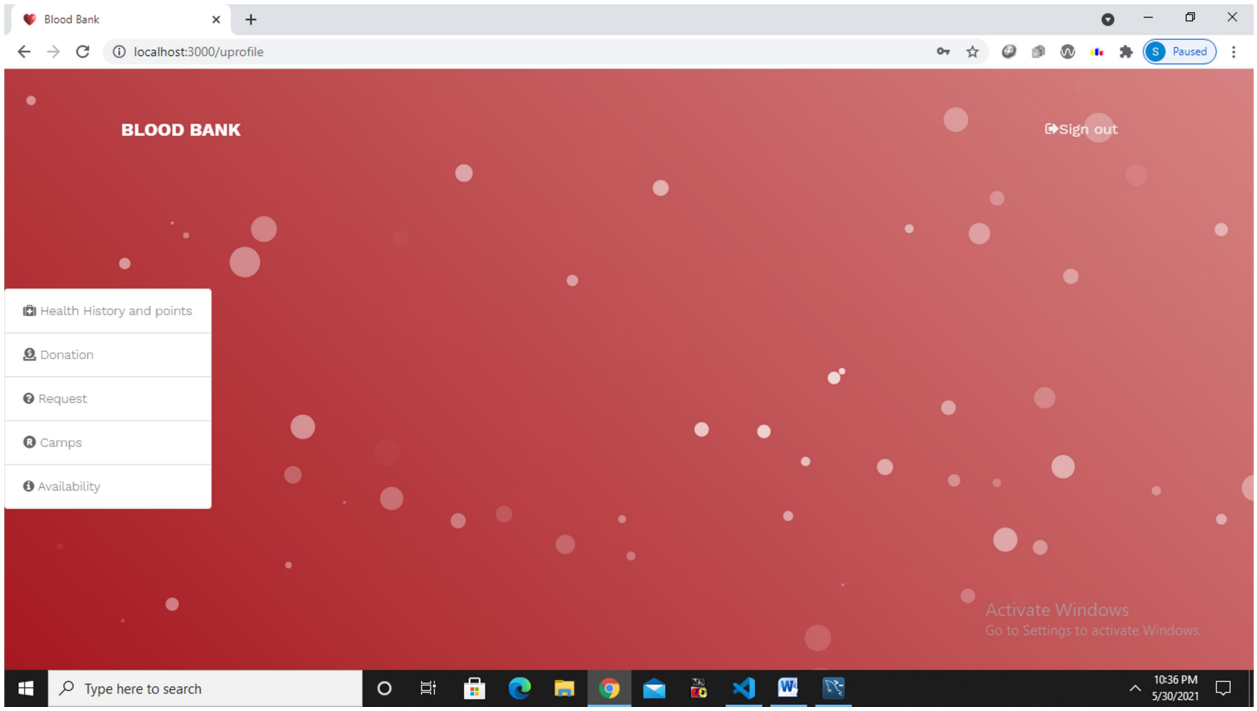


Log in procedure:



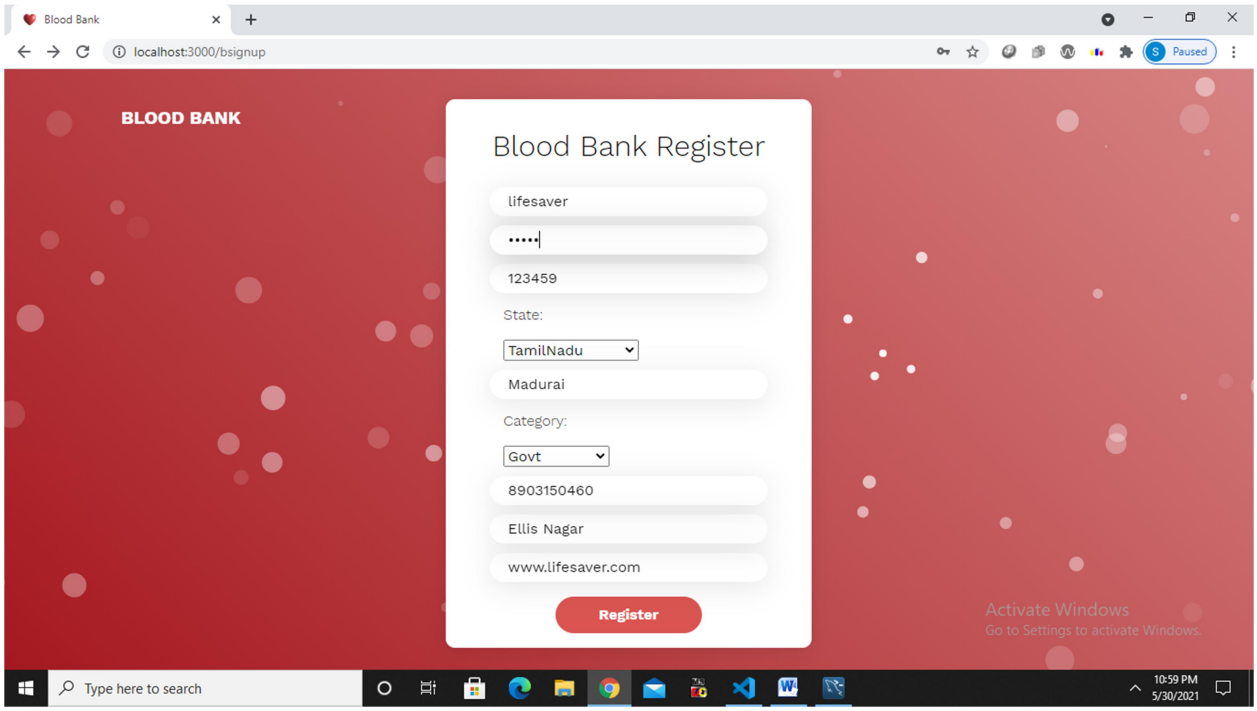
User who has signed up with the website can log in with password he/she had mentioned and the userid given to him after successful creation of his/her account.

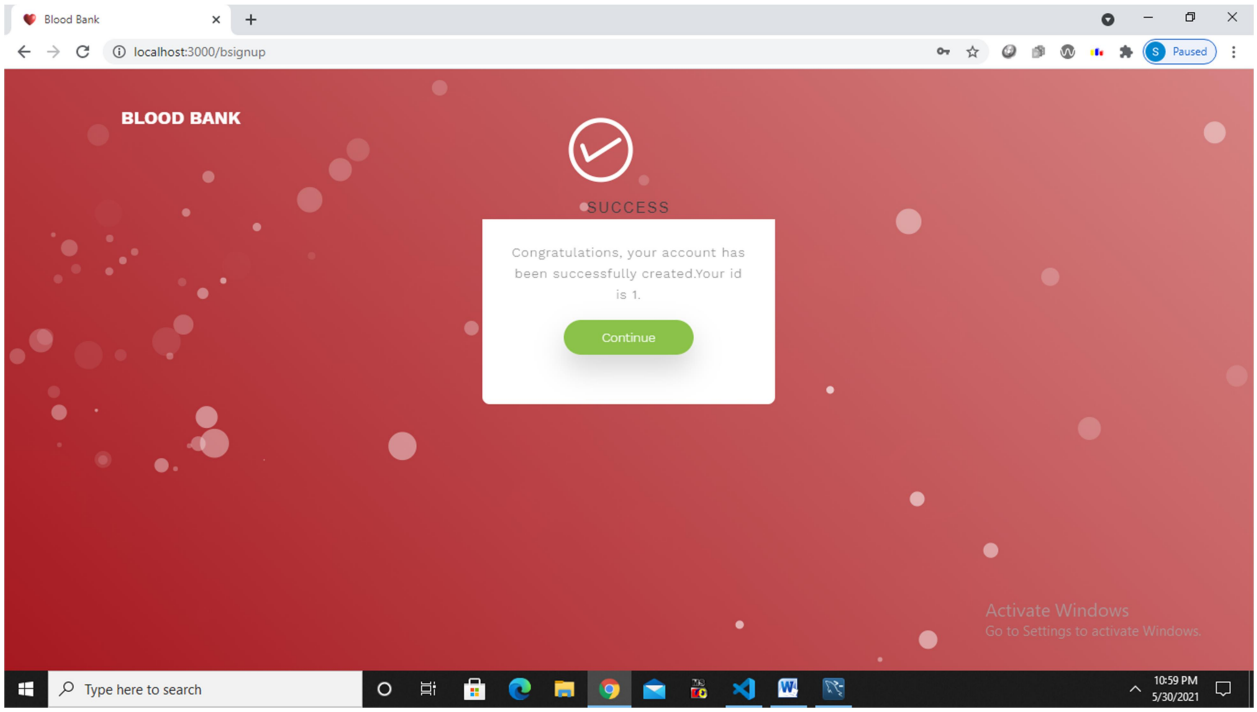
If the user's credentials match with the ones stored in database,the user is taken to his profile dashboard.



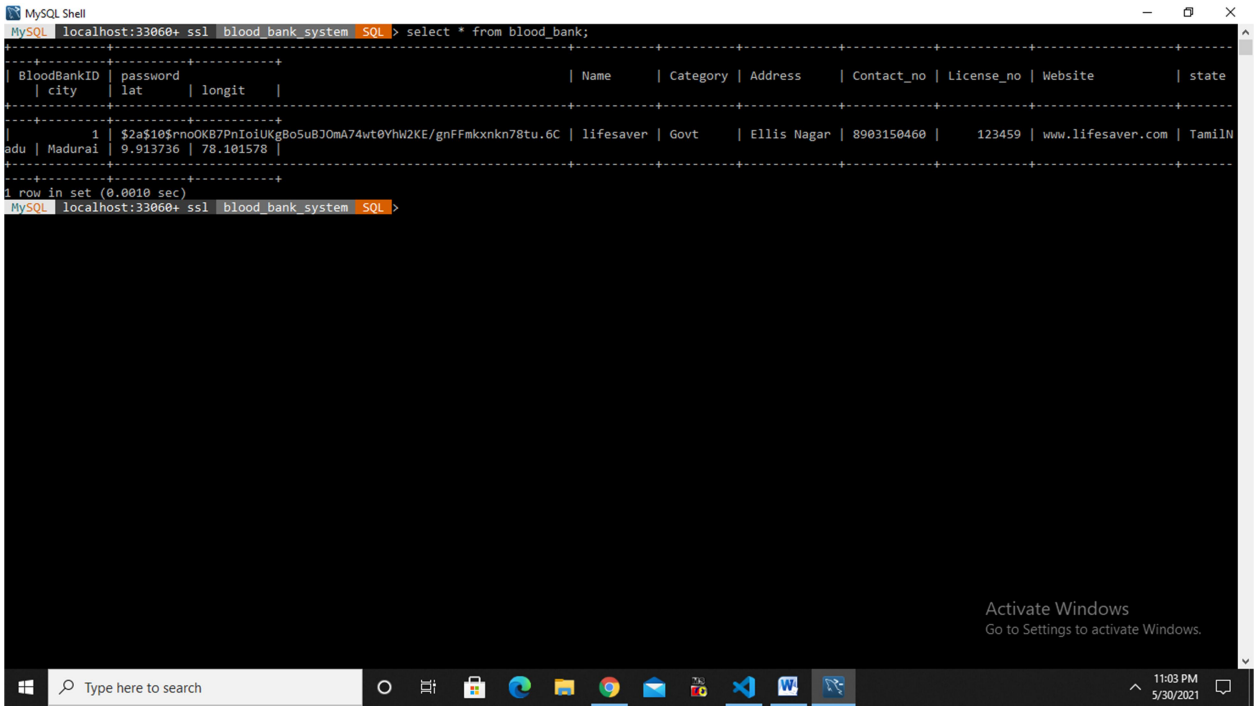
Blood Bank Side:

Sign up:

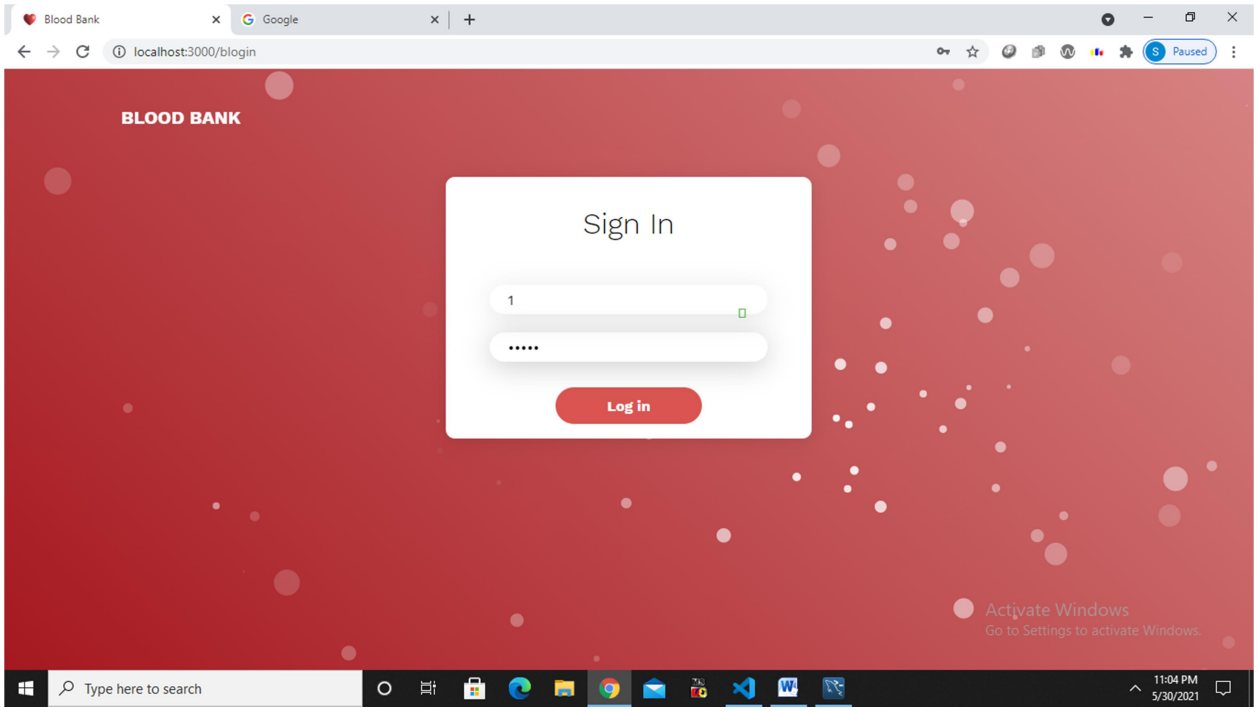

Blood Bank
localhost:3000/bsignup
Paused
BLOOD BANK
Blood Bank Register
lifesaver
123459
State:
TamilNadu
Madurai
Category:
Govt
8903150460
Ellis Nagar
www.lifesaver.com
Register
Activate Windows
Go to Settings to activate Windows.
Type here to search
10:59 PM
5/30/2021


Blood Bank
localhost:3000/bsignup
Paused
BLOOD BANK
SUCCESS
Congratulations, your account has been successfully created.Your id is 1.
Continue
Activate Windows
Go to Settings to activate Windows.
Type here to search
10:59 PM
5/30/2021

A blood bank agency can register in our website by filling the details as above.Then,a success message gets displayed showing the id assigned to the blood bank.The details of the blood bank also get stored in the database(blood_bank table).

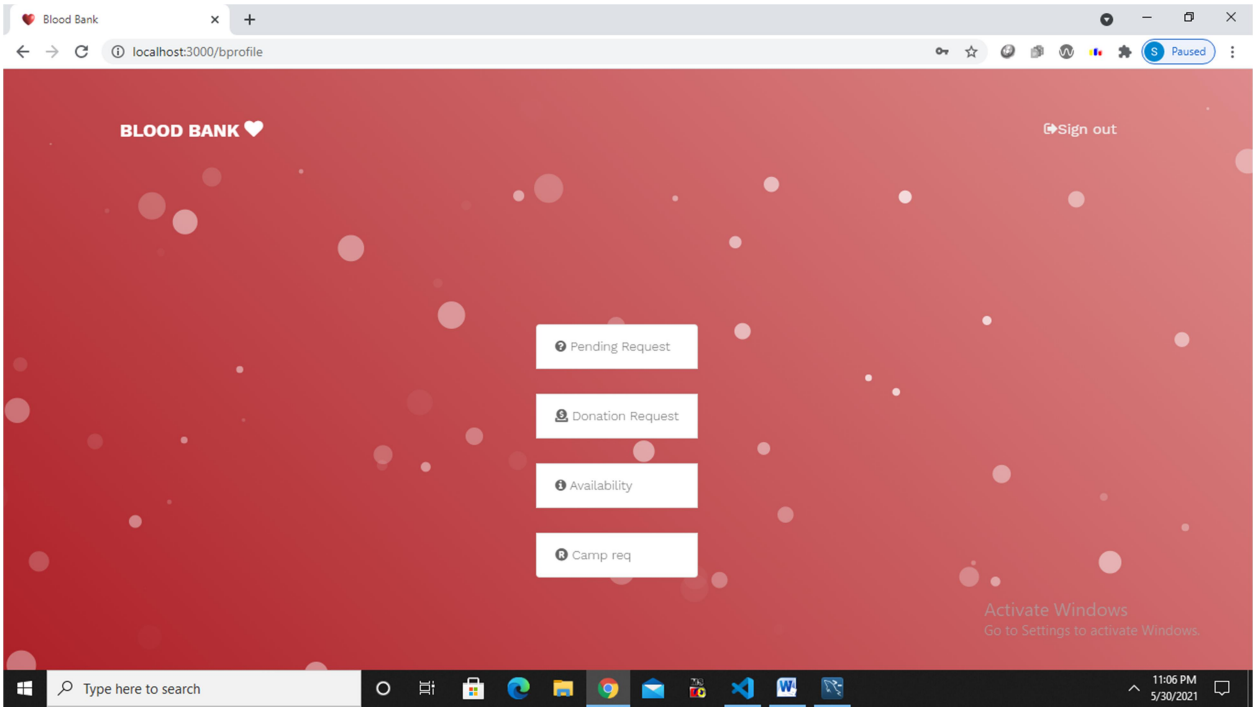


Log in procedure:



Blood banks who have signed up with the website can log in with password they had mentioned and the id given to them after successful creation of their account.

If the credentials match with the ones stored in database,they are taken to the profile dashboard.


Blood Bank
localhost:3000/bprofile
Paused
BLOOD BANK
Sign out
Pending Request
Donation Request
Availability
Camp req
Activate Windows
Go to Settings to activate Windows.
Type here to search
11:06 PM
5/30/2021

BACKEND FLOW:

User side endpoints:

```
//user signup
app.get('/usignup', user.signup);//to display form
app.post('/usignup', user.signup);//to send the details with POST req
//user login
app.get('/ulogin', user.login);//to display form
app.post('/ulogin', user.login);//to send login credentials through POST req
//display user profile
app.get('/uprofile', user.dashboard);
//display health history of user
app.get('/health_history',user.history);
//to logout
app.get('/ulogout', user.logout);

//donation register
app.get('/udonate',user.donate);
app.post('/udonate',user.donate);
//blood request register
app.get('/urequest',user.request);
app.post('/urequest',user.request);
//check blood availablity
app.get('/display_uavailable',user.avail_form);
app.get('/uavailable',user.available);
//check camp availablity
app.get('/campavail',user.campavail);
```

Blood bank side endpoints:

```
//Bbank
//bbank signup
app.get('/bsignup', bbank.bsignup);
app.post('/bsignup',  bbank.bsignup);
//bbank login
app.get('/blogin',  bbank.blogin);
app.post('/blogin',  bbank.blogin);
//display bank profile
app.get('/bprofile', bbank.bdashboard);
//display and fill form to register a camp
app.get('/campreg',  bbank.campreg);
app.post('/campreg',  bbank.campreg);
//to display blood availability
app.get('/bavailable',  bbank.bavailable);

//donations
app.get('/bdonations',  bbank.bdonations);//list donations
app.get('/bcomp_donations/:id',(req,res)=>{ //get units and comp for a part donation
    res.render('get_don_info.ejs',{message:req.params.id});
});
app.post('/bcomp_donations/:id',bbank.bcomp_donations);//complete the donation
//requests
app.get('/brequests',  bbank.brequests);//list requests
app.get('/bserve_requests',  bbank.bserve_requests);//serve requests

//logout
app.get('/blogout', bbank.blogout);
```

AUTHENTICATION:

Sessions and Cookies are used.

```
app.use(express.static(path.join(__dirname, 'public')));
//session
app.use(session({
    genid: (req) => {
        console.log('Inside the session middleware')
        console.log(req.sessionID)
        return uuid() // use UUIDs for session IDs
    },
    store: new FileStore({logFn: function(){}}),
    secret:'dbmsproj',
    resave: false,
    saveUninitialized: true
}))
```

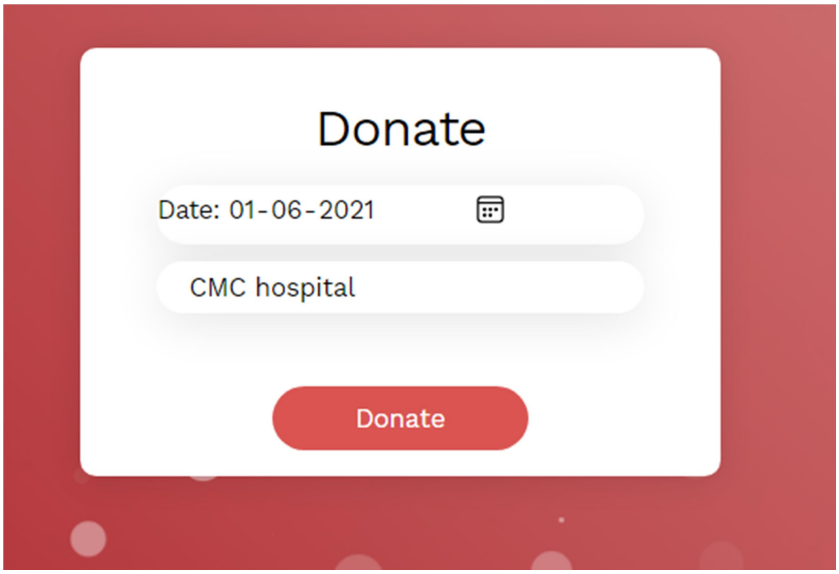
User id is stored in Sessions data once a User/Blood bank logs in and this serves as Proof that the subsequent requests with this SessionID(unique number) are authenticated.To differentiate User and Blood bank the prefix ‘a’ and ‘b’ is used.

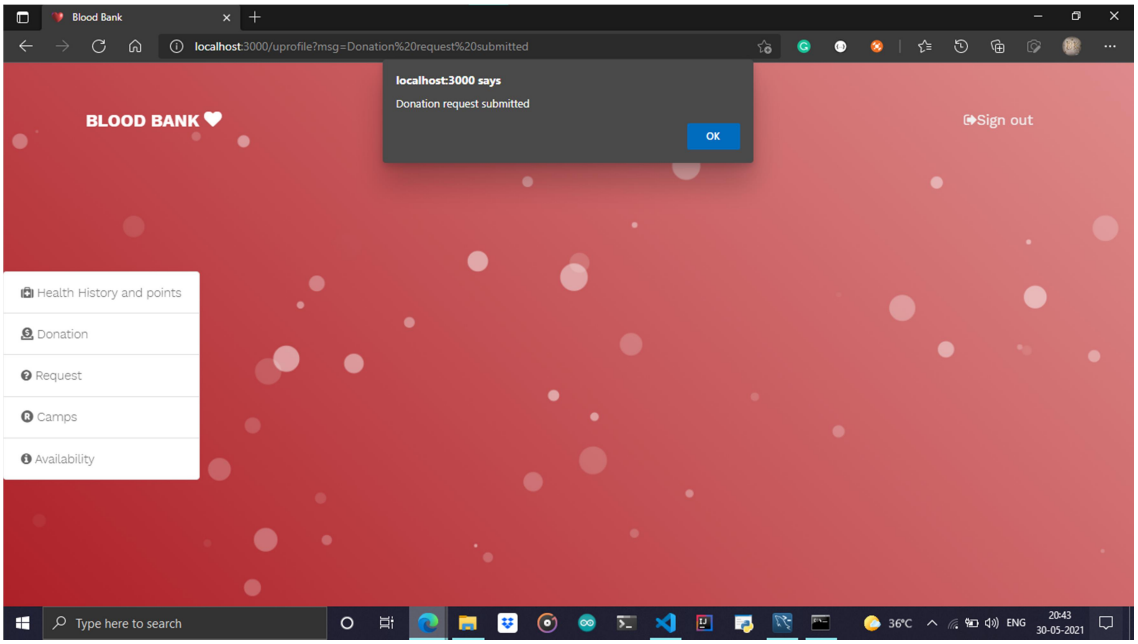
1.Donation flow

```
+------------+--------+-------------+---------------+----------------+---------+
20 rows in set (0.3618 sec)
MySQL  localhost:33060+ ssl  bbank  SQL > select * from donations;
+------------+--------+-------------+---------------+--------+
| DonationID | UserID | BloodBankID | date_donation | status |
+------------+--------+-------------+---------------+--------+
|          1 |      1 |           1 | 2021-05-11    | Done   |
|          2 |      2 |           1 | 2021-05-17    | Done   |
|          3 |      3 |           3 | 2021-05-18    | Done   |
|          4 |      4 |           4 | 2021-05-20    | Done   |
|          5 |      4 |           5 | 2021-05-27    | Done   |
+------------+--------+-------------+---------------+--------+
5 rows in set (0.5677 sec)
```

a. User registers for donation in a Blood bank on a particular date

Click the Donations tab and enter credentials.





Donation request is submitted.

```
MySQL  localhost:33060+ ssl  bbank  SQL > select * from donations;
+------------+--------+-------------+---------------+---------+
| DonationID | UserID | BloodBankID | date_donation | status  |
+------------+--------+-------------+---------------+---------+
|          1 |      1 |           1 | 2021-05-11    | Done    |
|          2 |      2 |           1 | 2021-05-17    | Done    |
|          3 |      3 |           3 | 2021-05-18    | Done    |
|          4 |      4 |           4 | 2021-05-20    | Done    |
|          5 |      4 |           5 | 2021-05-27    | Done    |
|          6 |      4 |           7 | 2021-06-01    | Pending |
+------------+--------+-------------+---------------+---------+
6 rows in set (0.0037 sec)
```

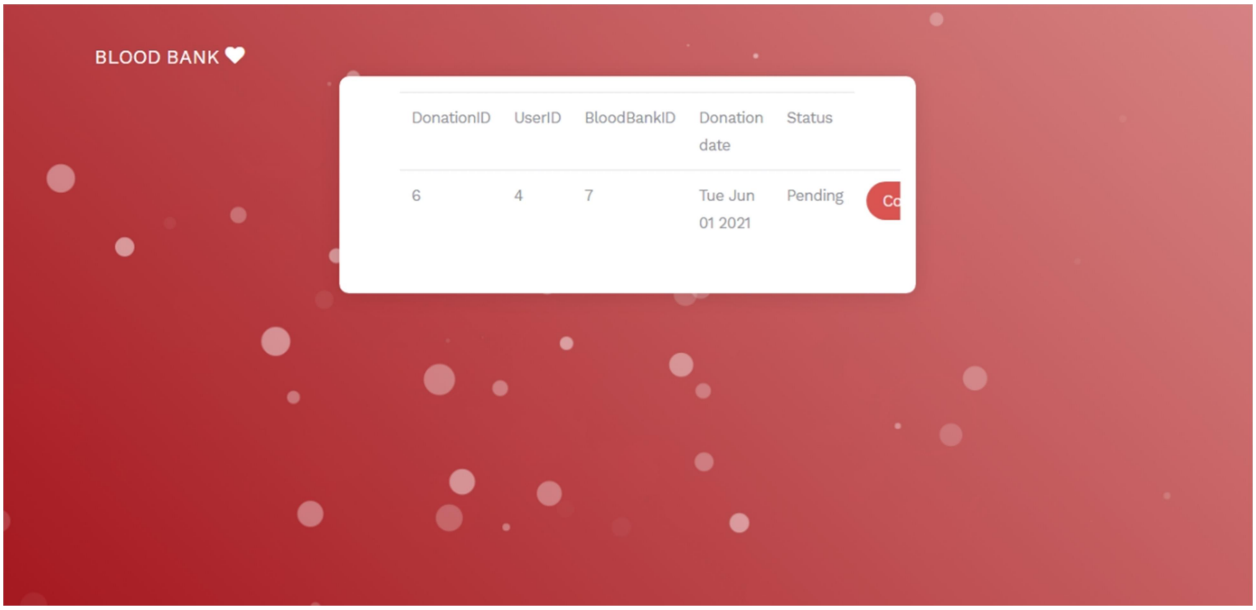
Added in Donations table with status as Pending.

```
MySQL  localhost:33060+ ssl  bbank  SQL > select * from user where userid=4;
+--------+--------------------------------------------------------------+------+-----+-----------+------------+-----------------+-----+-------+
| UserID | password                                                     | Name | Age | state     | city       | address         | pts | email |
+--------+--------------------------------------------------------------+------+-----+-----------+------------+-----------------+-----+-------+
|      4 | $2a$10$pWmLqVThvuZORfui99s6MeCfnnMK9biwSqy2awVRm54jwqv8nrhRq | Arun |  25 | Tamil Nadu | Coimbatore | Coimbatore South |   7 | NULL  |
+--------+--------------------------------------------------------------+------+-----+-----------+------------+-----------------+-----+-------+
1 row in set (0.3292 sec)
```

Points incremented for that User.

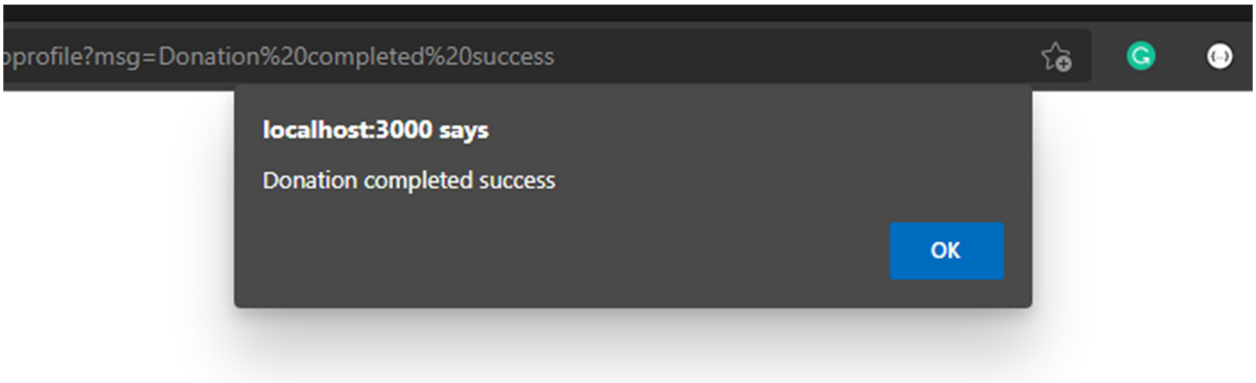
b. Donations get added to the Blood bank side.

Sign in to the Bloodbank side(id:7 and check Pending Donations).



The Pending donations for this blood bank are listed.

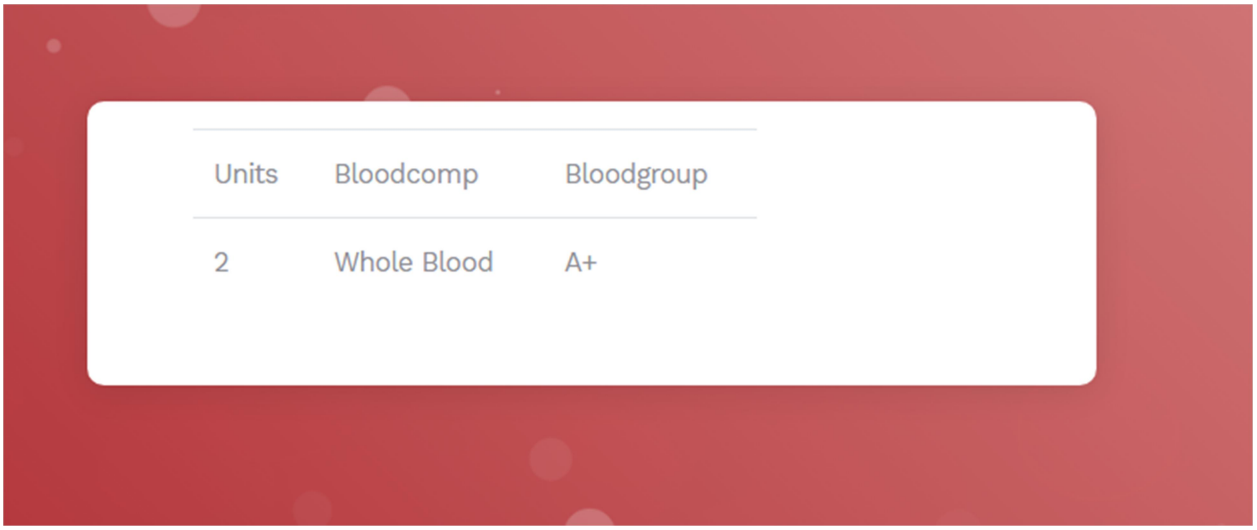
c. On that particular day if the person reaches out to the Bank and donates blood, its details are recorded and the donation completed.



```
MySQL  localhost:33060+ ssl  bbank  SQL > select * from donations;
+------------+--------+-------------+---------------+--------+
| DonationID | UserID | BloodBankID | date_donation | status |
+------------+--------+-------------+---------------+--------+
|          1 |      1 |           1 | 2021-05-11    | Done   |
|          2 |      2 |           1 | 2021-05-17    | Done   |
|          3 |      3 |           3 | 2021-05-18    | Done   |
|          4 |      4 |           4 | 2021-05-20    | Done   |
|          5 |      4 |           5 | 2021-05-27    | Done   |
|          6 |      4 |           7 | 2021-06-01    | Done   |
+------------+--------+-------------+---------------+--------+
6 rows in set (0.1339 sec)
```

Status is changed to done in Database

d. The received blood gets added to its Available blood.



```
6 rows in set (0.1339 sec)
MySQL  localhost:33060+ ssl  bbank  SQL > select * from blood;
+---------+------------+-------------+-----------------+-------------+
| BloodID | DonationID | blood_group | blood_component | blood_units |
+---------+------------+-------------+-----------------+-------------+
|       1 |          1 | A+          | Whole Blood     |           3 |
|       2 |          2 | B+          | Plasma          |           3 |
|       4 |          4 | A+          | Platelets       |           2 |
|       5 |          5 | A+          | Plasma          |           2 |
|       7 |          6 | A+          | Whole Blood     |           2 |
+---------+------------+-------------+-----------------+-------------+
5 rows in set (0.0023 sec)
```

It gets stored in Blood relation with BloodID 7 and DonationID holds the info about the user and blood bank.

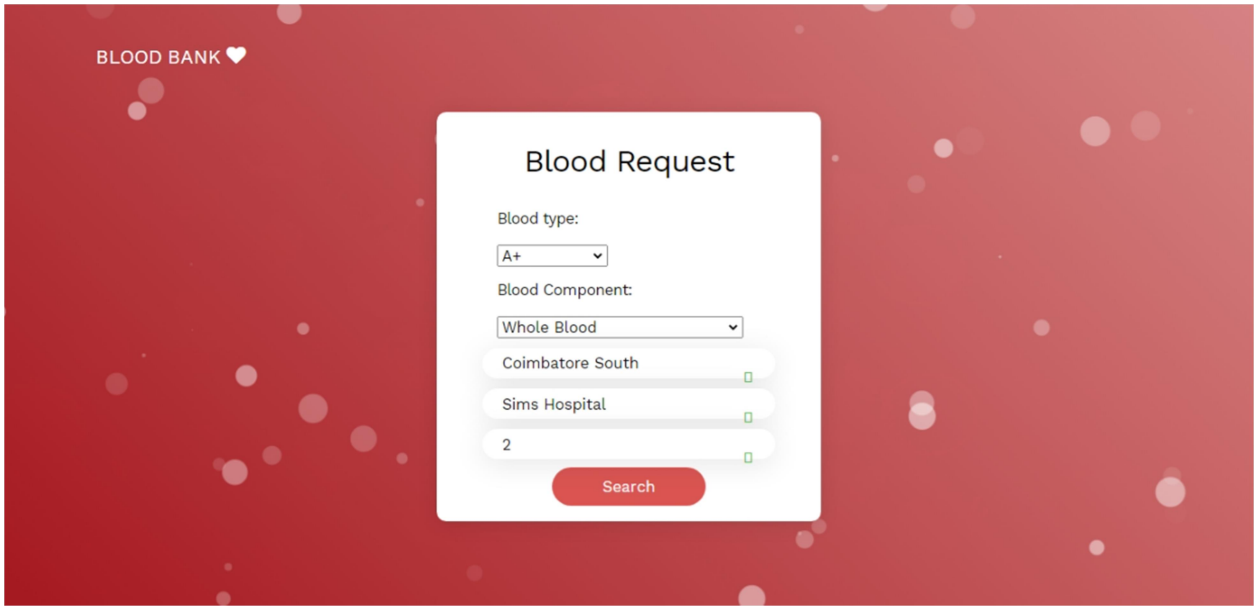
Donation cycle completed.

Now, let's see how these stored blood are utilized.

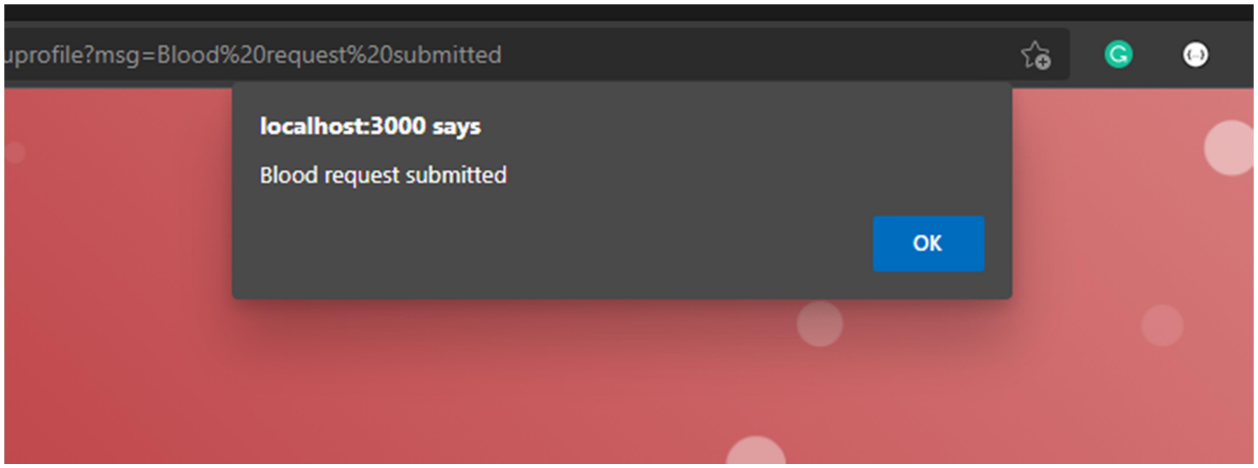
2.Requests flow:

Let's say we are in need of Blood, we can register a Request to a blood bank nearby.

a. Fill the Blood request form on the User side



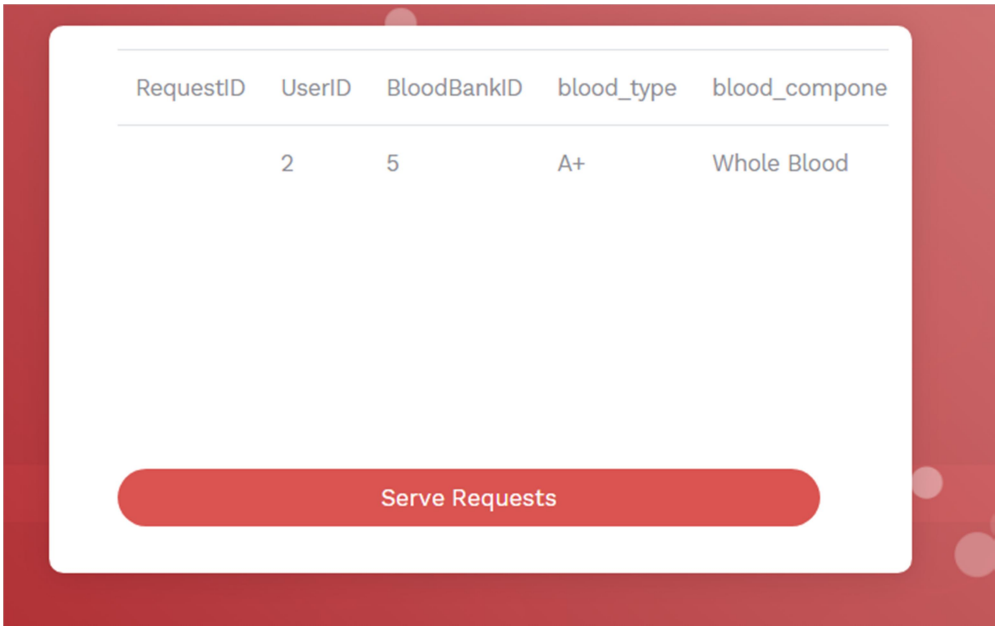
The User wants blood near Coimbatore South and so the Blood bank in that area Sims Hospital is first approached with this request.



b. The Request is added along with Date and Time to the Database and is reflected as Pending requests in the Blood bank side.

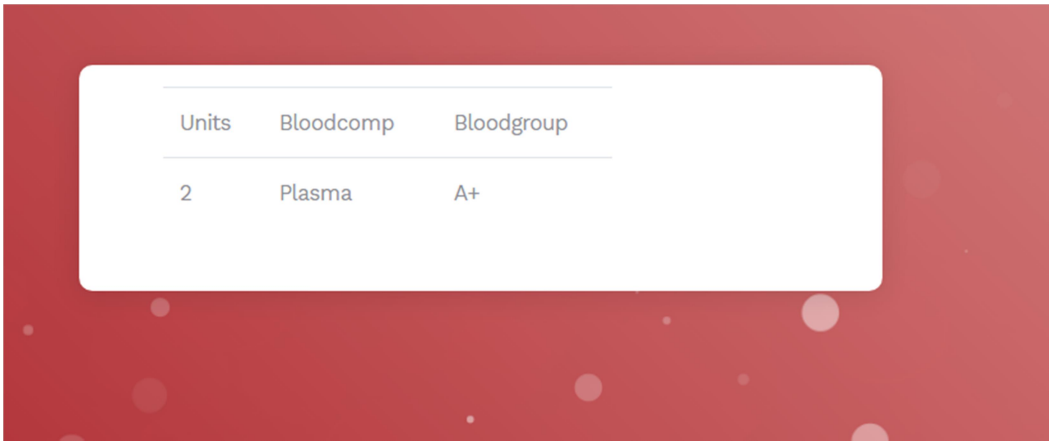
```
MySQL  localhost:33060+ ssl  bbank  SQL > select * from requests;
+-----------+--------+-------------+------------+----------------+---------+-------------------+---------------------+-------+
| RequestID | UserID | BloodBankID | blood_type | blood_component | status  | area              | date_time           | Units |
+-----------+--------+-------------+------------+----------------+---------+-------------------+---------------------+-------+
|         1 |      1 |           3 | A+         | RBC            | Pending | egmore            | NULL                |     2 |
|         2 |      2 |           3 | B+         | Whole blood    | Done    | Egmore            | NULL                |     4 |
|         4 |      4 |           3 | B+         | Whole blood    | Done    | Egmore            | NULL                |     1 |
|         5 |      3 |           3 | O-         | Plasma         | Pending | Egmore            | NULL                |     2 |
|         7 |      4 |           7 | O+         | Whole blood    | Pending | Coimbatore Central | 2021-05-27 21:18:01 |     2 |
|         8 |      2 |           5 | A+         | Whole Blood    | Pending | Coimbatore South  | 2021-05-30 21:17:50 |     2 |
+-----------+--------+-------------+------------+----------------+---------+-------------------+---------------------+-------+
6 rows in set (0.1794 sec)
```

In the blood bank side, it is shown as Pending requests.

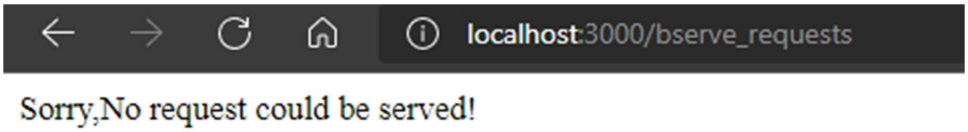


c. The Blood bank checks its available blood status and serves the requests if it possible.

This check is done automatically by clicking the **Serve Requests** option.

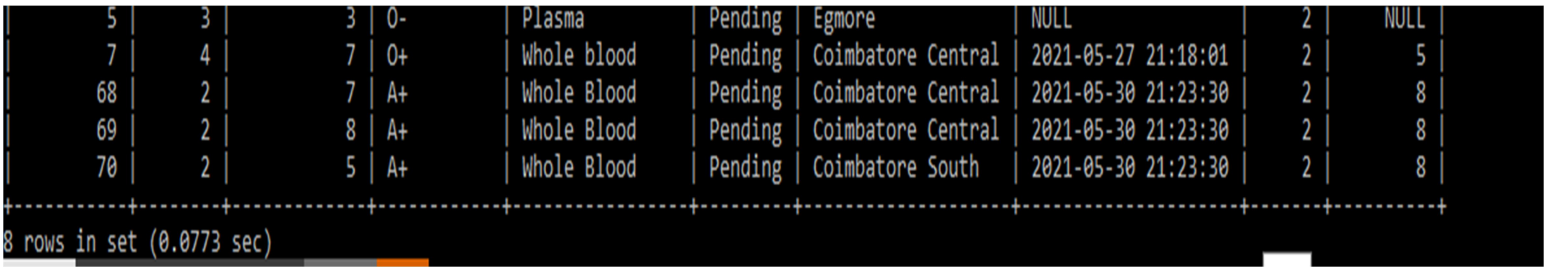


The Bank has only Plasma on its Available list currently and so it couldn't serve this request.



d. If the available blood is lower than the requested amount, it forwards the request to other blood banks nearby.

As it wasn't able to serve the request it forwards it to 5 other blood banks within a 25km radius.



5	3	3	O-	Plasma	Pending	Egmore	NULL	2	NULL
7	4	7	O+	Whole blood	Pending	Coimbatore Central	2021-05-27 21:18:01	2	5
68	2	7	A+	Whole Blood	Pending	Coimbatore Central	2021-05-30 21:23:30	2	8
69	2	8	A+	Whole Blood	Pending	Coimbatore Central	2021-05-30 21:23:30	2	8
70	2	5	A+	Whole Blood	Pending	Coimbatore South	2021-05-30 21:23:30	2	8

8 rows in set (0.0773 sec)

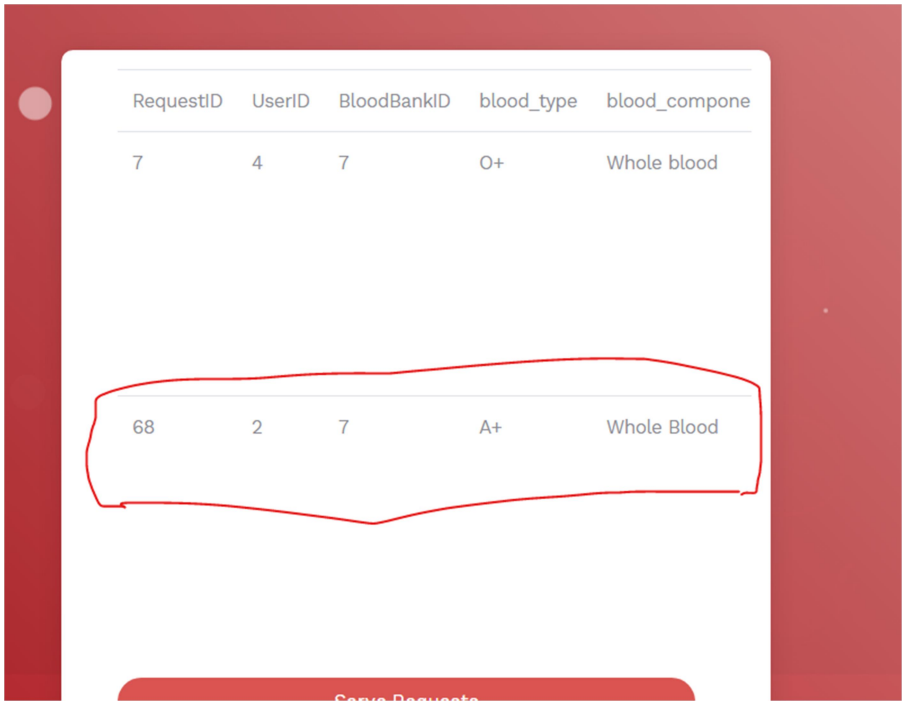
We can see that it has forwarded it to 2 other banks(as only 2 were within 25km radius).

Request IDs 69 and 70 are added by the previous request(8) and so its ParentID is 8.

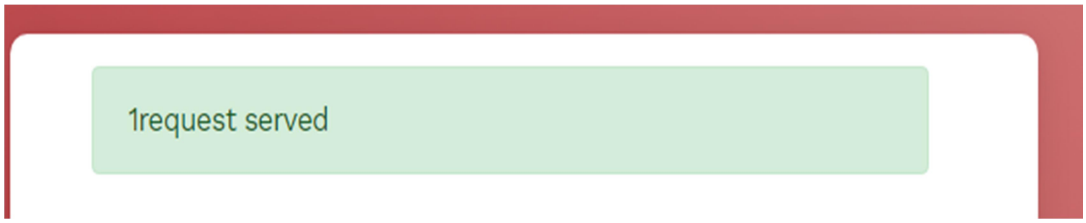
All Requests with the same ParentID denote the same request and so if any one of this group is served all others are redundant and can be removed.

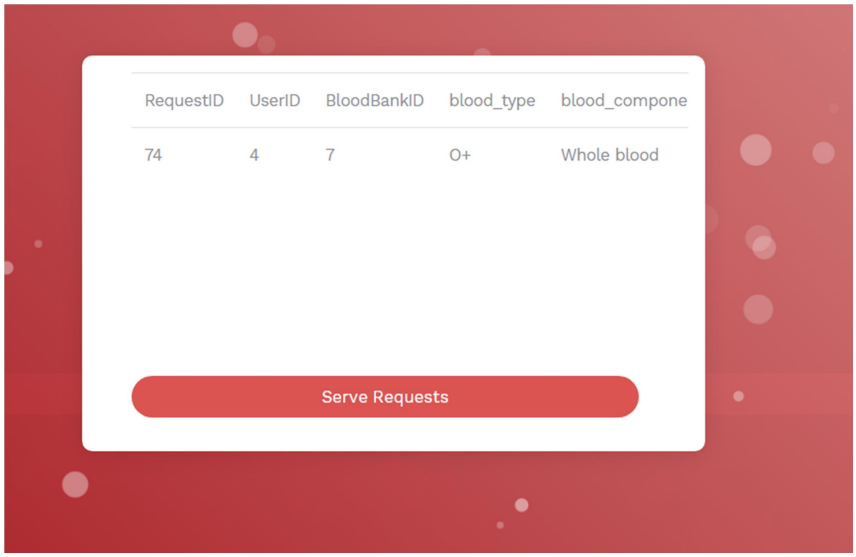
e.Now,Blood banks 7 and 8 can see this request in their pending list.

Let's take Blood bank 7 and check its Pending requests.



As this blood bank has 2 units of A+ Whole blood(got from donations),it can serve the request but the Request7 can't be served as O+ blood isn't available. So 1 request is served and the other is forwarded.





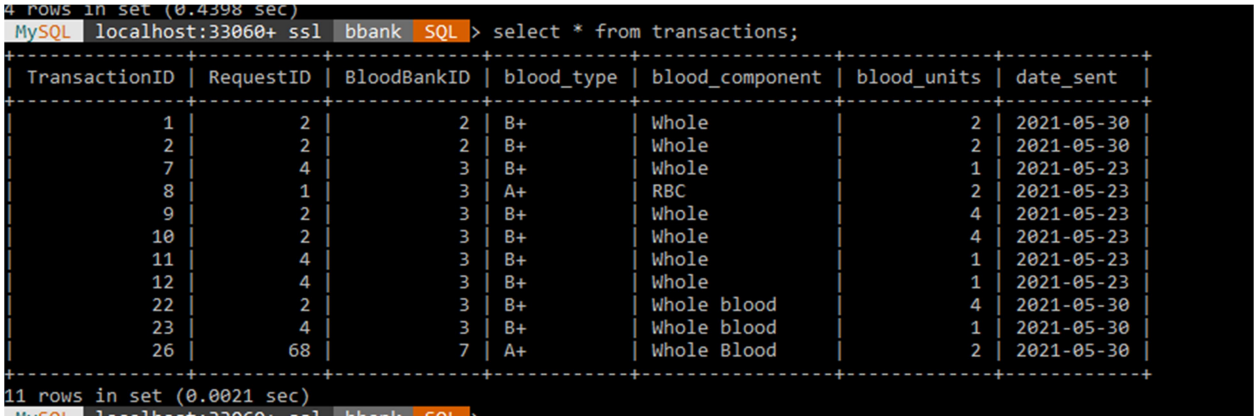
This Request74 is that O+ blood request which is forwarded to itself.

```
Query OK, 0 rows affected (0.8892 sec)
MySQL  localhost:33060+ ssl  bbank  SQL > select * from requests;
+-----------+--------+-------------+------------+----------------+---------+--------------------+---------------------+-------+----------+
| RequestID | UserID | BloodBankID | blood_type | blood_component | status  | area               | date_time           | Units | ParentID |
+-----------+--------+-------------+------------+----------------+---------+--------------------+---------------------+-------+----------+
|         1 |      1 |           3 | A+         | RBC            | Pending | egmore             | NULL                |     2 |     NULL |
|         2 |      2 |           3 | B+         | Whole blood    | Done    | Egmore             | NULL                |     4 |     NULL |
|         4 |      4 |           3 | B+         | Whole blood    | Done    | Egmore             | NULL                |     1 |     NULL |
|         5 |      3 |           3 | O-         | Plasma         | Pending | Egmore             | NULL                |     2 |     NULL |
|        68 |      2 |           7 | A+         | Whole Blood    | Removed | Coimbatore Central | 2021-05-30 21:23:30 |     2 |        8 |
|        69 |      2 |           8 | A+         | Whole Blood    | Removed | Coimbatore Central | 2021-05-30 21:23:30 |     2 |        8 |
|        70 |      2 |           5 | A+         | Whole Blood    | Removed | Coimbatore South   | 2021-05-30 21:23:30 |     2 |        8 |
|        74 |      4 |           7 | O+         | Whole blood    | Pending | Coimbatore Central | 2021-05-30 22:07:00 |     2 |        5 |
|        75 |      4 |           8 | O+         | Whole blood    | Pending | Coimbatore Central | 2021-05-30 22:07:00 |     2 |        5 |
|        76 |      4 |           5 | O+         | Whole blood    | Pending | Coimbatore South   | 2021-05-30 22:07:00 |     2 |        5 |
+-----------+--------+-------------+------------+----------------+---------+--------------------+---------------------+-------+----------+
10 rows in set (0.1157 sec)
```

Request 68,69,70 belong to this group and so they all are removed.

Requests 74,75,76 are created new for Request7 which couldn't be served now.

e. Once the request is served by any on the blood banks, all other requests of the same group are removed, so that a single request isn't served multiple times.
f. Blood availability status is updated as a request has been served and this Transaction is stored in Transactions relation.



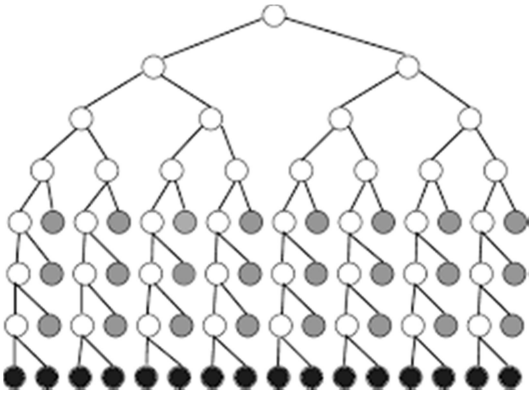
```
4 rows in set (0.4398 sec)
MySQL  localhost:33060+ ssl  bbank  SQL > select * from transactions;
+---------------+-----------+-------------+------------+-----------------+-------------+------------+
| TransactionID | RequestID | BloodBankID | blood_type | blood_component | blood_units | date_sent  |
+---------------+-----------+-------------+------------+-----------------+-------------+------------+
|             1 |         2 |           2 | B+         | Whole           |           2 | 2021-05-30 |
|             2 |         2 |           2 | B+         | Whole           |           2 | 2021-05-30 |
|             7 |         4 |           3 | B+         | Whole           |           1 | 2021-05-23 |
|             8 |         1 |           3 | A+         | RBC             |           2 | 2021-05-23 |
|             9 |         2 |           3 | B+         | Whole           |           4 | 2021-05-23 |
|            10 |         2 |           3 | B+         | Whole           |           4 | 2021-05-23 |
|            11 |         4 |           3 | B+         | Whole           |           1 | 2021-05-23 |
|            12 |         4 |           3 | B+         | Whole           |           1 | 2021-05-23 |
|            22 |         2 |           3 | B+         | Whole blood     |           4 | 2021-05-30 |
|            23 |         4 |           3 | B+         | Whole blood     |           1 | 2021-05-30 |
|            26 |        68 |           7 | A+         | Whole Blood     |           2 | 2021-05-30 |
+---------------+-----------+-------------+------------+-----------------+-------------+------------+
11 rows in set (0.0021 sec)
```

Transaction23 records this serving of request with the blood bank which served it.

How forwarding of requests helps?

As a single unserved request is forwarded to max 5 times, each of which can be served 5 times further , it grows exponentially till the request is served. This speeds up the process and reduces the waiting time of the user.

1->5->25->125->625->.....At each level the number of blood banks approached increases exponentially.



How nearby blood banks are found?

Latitude and Longitude of each blood bank is obtained using OpenStreetMap API and npm node module geocoder.

OpenStreetMap

node-geocoder - npm (npmjs.com)

To measure distance Haversine formula is used.

Haversine formula - Wikipedia

$$\Delta\hat{\sigma} = 2\arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_s\cos\phi_f\sin^2\left(\frac{\Delta\lambda}{2}\right)}\right)$$

The returned distance is sorted and the blood banks within 25 kms radius are chosen.

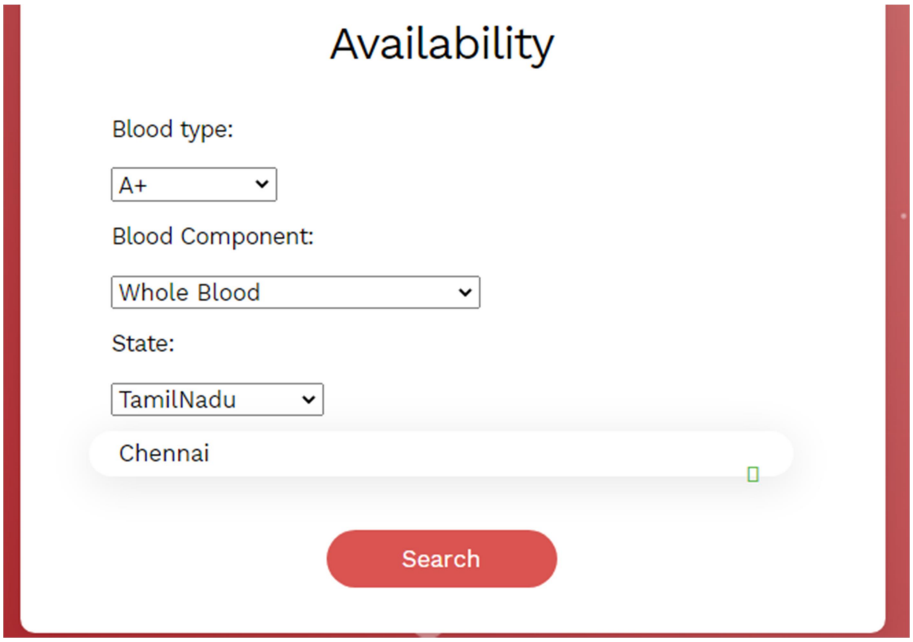
SQL FUNCTIONS:

1.proc_donations - procedure to complete donation status by updating Blood relation and incrementing Points of user.

proc_donations.sql

```
DROP PROCEDURE IF EXISTS UPDATE_DONATIONS;
DELIMITER $$
CREATE PROCEDURE UPDATE_DONATIONS
(IN did INT,IN bid INT,IN units INT,IN comp varchar(255))
BEGIN
    DECLARE b_type varchar(20);
    DECLARE d_date date;
    DECLARE u_id int;
    UPDATE Donations
    SET Status="Done"
    WHERE DonationID=did;

    SELECT date_donation,UserID into d_date,u_id FROM Donations where DonationID=did;

    UPDATE health_history
    SET last_donated=d_date
    where UserID=u_id;

    SELECT blood_type INTO b_type FROM HEALTH_HISTORY
    INNER JOIN User USING(UserID)
    WHERE UserID=u_id;

    INSERT INTO Blood Values
    (
        bid,did,b_type,comp,units
    );
END
$$
DELIMITER ;
```

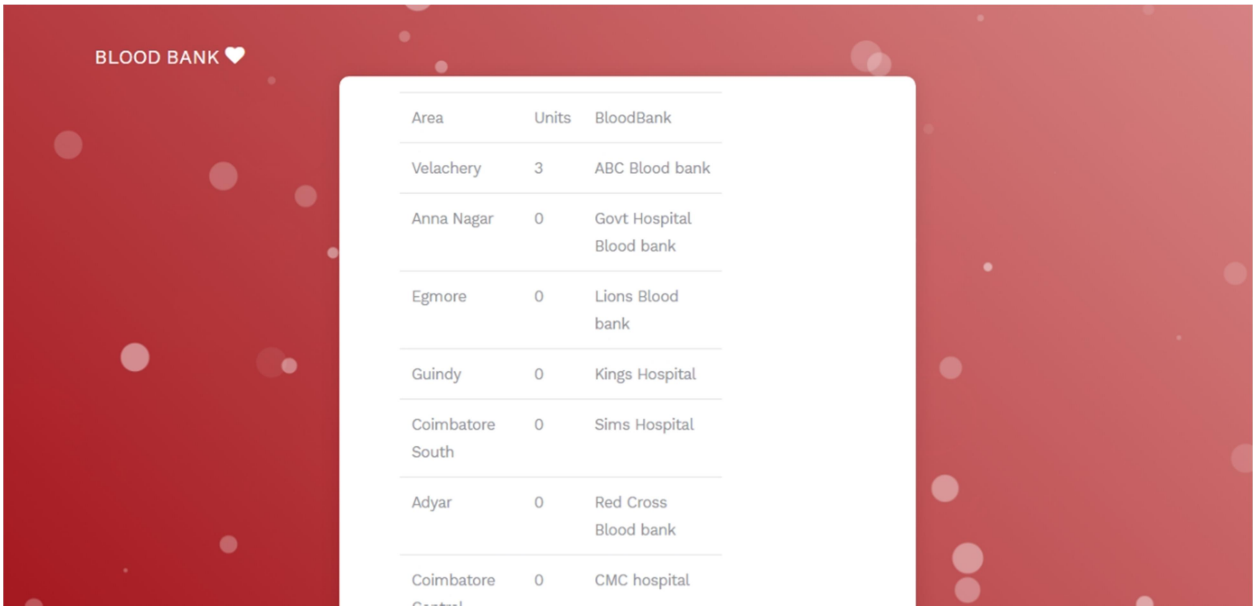
2.func_uavail - Function to check Blood availability from User side. Takes in the Blood type,component,City and State as input and outputs a JSON.



Function U_AVAIL

```
DROP FUNCTION IF EXISTS U_AVAIL;
DELIMITER $$
CREATE FUNCTION U_AVAIL
(city varchar(255),state varchar(255),b_type varchar(255),b_comp varchar(255))
RETURNS JSON
DETERMINISTIC
BEGIN
    DECLARE l_bbid INT;
    DECLARE l_name varchar(255);
    DECLARE l_address varchar(255);
    DECLARE l_sum INT;
    DECLARE l_bid INT;
    DECLARE l_unit INT;
    DECLARE temp INT;
    DECLARE data_obj JSON;
    DECLARE data_obj2 JSON;
    DECLARE FLAG1 INT DEFAULT 0;
    DECLARE FLAG2 INT DEFAULT 0;
    DECLARE bbid CURSOR FOR SELECT BloodbankID,Name,Address FROM blood_bank WHERE City=city AND State=state;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET FLAG1 = 1;

    OPEN bbid;

    bbid_loop:LOOP
        FETCH bbid into l_bbid,l_name,l_address;
        IF FLAG1 THEN
          CLOSE bbid;
          LEAVE bbid_loop;
        END IF;

          block1: BEGIN
           DECLARE blood_unit CURSOR FOR Select BloodID,blood_units FROM Blood b INNER JOIN Donations d USING(DonationID)
           DECLARE CONTINUE HANDLER FOR NOT FOUND SET FLAG2 = 1;
```

```
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET FLAG2 = 1;
        OPEN blood_unit;

        SET l_sum=0;
        blood_loop:LOOP
            FETCH blood_unit into l_bid,l_unit;
            IF FLAG2 THEN
                CLOSE blood_unit;
                SET FLAG2=0;
                LEAVE blood_loop;
            END IF;
            SET l_sum=l_sum+l_unit;
        END LOOP blood_loop;
        END block1;

        SELECT JSON_MERGE(
            JSON_OBJECT('Bloodbank',l_name),
            JSON_OBJECT('Area', l_address),
            JSON_OBJECT('Units',l_sum)
        ) INTO data_obj;

        SELECT JSON_DEPTH(data_obj2) INTO temp;
        IF temp IS NULL THEN
            SET data_obj2=data_obj;
        ELSE
            SELECT JSON_ARRAY_APPEND(data_obj2,'$',data_obj) INTO data_obj2;
        END IF;

    END LOOP bbid_loop;
    RETURN data_obj2;

END$$
```



Area	Units	BloodBank
Velachery	3	ABC Blood bank
Anna Nagar	0	Govt Hospital Blood bank
Egmore	0	Lions Blood bank
Guindy	0	Kings Hospital
Coimbatore South	0	Sims Hospital
Adyar	0	Red Cross Blood bank
Coimbatore	0	CMC hospital

3.func_bavail - Function to check Blood availability in Bankside. Takes only the BankID as input and returns a JSON with the number of Units of each blood type and component.

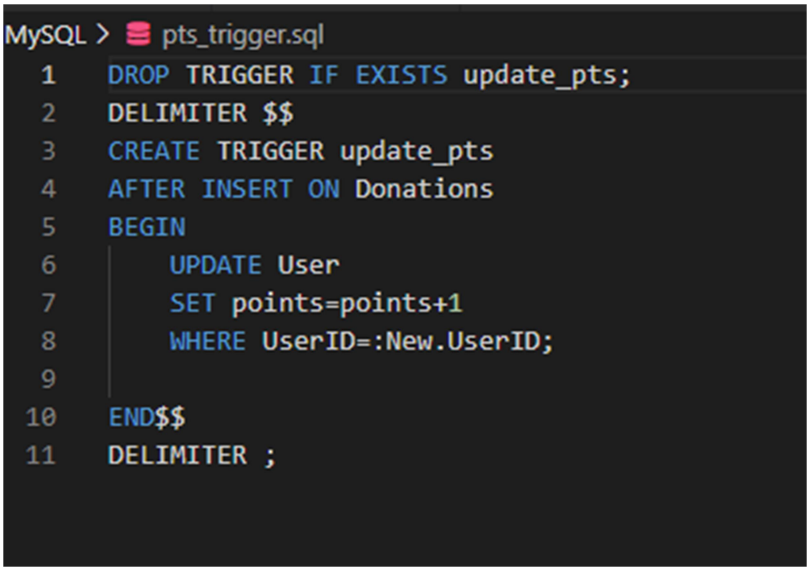
4.func_requests (BloodbankID) -serves requests belonging to this Blood bank

Uses Cursor to loop through requests and for each request checks the Available blood(uses b_Avail function) and decides if the the request can be satisfied.

Returns the number of requests satisfied.

TRIGGERS:

1.trigger_pts - Trigger to increment Points of User when he registers for a Donation.



2. trigger_tran - Trigger to create and insert Transactions once a request has been served, and this is done by checking the status.

QL > trigger_tran.sql

```
DROP TRIGGER IF EXISTS update_avail;
DELIMITER $$
CREATE TRIGGER update_avail
AFTER UPDATE  ON Requests
FOR EACH ROW
BEGIN
    DECLARE l_date datetime;
    IF (New.status="Done") THEN
        SELECT NOW() INTO l_date;
        INSERT INTO TRANSACTIONS(RequestID,BloodBankID,blood_type,blood_component,blood_units,date_sent) VALUES
        (New.RequestID,New.BloodBankID,New.blood_type,New.blood_component,New.Units,l_date);
    END IF;
END $$
DELIMITER ;
```

CONCLUSION:

Even though blood donation is considered important,there aren't many portals which coordinate the blood transfer process between blood banks and donors.

Keypoints:

1.We provide exponential forwarding of requests to ensure minimum waiting time.

2.We provide points and badges to encourage people which they can use for Medical checkups.

3.Covid recovered patients are encouraged to donate Plasma as its needed in the treatment of affected people.

Improvements:

1.Vaccination process can also be managed with this setup by storing the info of vaccinated people.

2. Currently, Requests are forwarded only to blood banks, this can be extended with forwarding to Donors nearby with alert messages.

We consider our project to be a head start in solving the issue of quick transfer of blood to people in need and encourage many people to come forward and donate blood.