Project Report

Date	26 November 2022
Team ID	PNT2022TMID25179
Project Name	Plasma Donor Application

1. INTRODUCTION

1.1 Project Overview

During COVID-19 crisis the requirement for plasma increased drastically as there were no vaccination found.With plasma therapy the recovery rates were high but the donor count was very low. It was very important to get information about the plasma donors.Saving the donor information and notifying about the current donors would be a helping hand.It can save time and help the users to track down the necessary information about the donors

1.2 Purpose

With rapid increase in the usage of social networks sites across the world, there is also a steady increase in blood donation requests as being noticed in the number of posts on these sites such as Facebook and twitter seeking blood donors. Finding blood donor is a challenging issue in almost every country. There are some blood donor finder applications in the market such as Blood app by Red Cross and Blood Donor Finder application by Neologix. However, more reliable applications that meet the needs of users need to be developed.

2. LITERATURE SURVEY

2.1 Existing problem

- Manual document and data entry.
- Only web based system is available no mobile based system available.
- Less Security.
- No proper coordination between different Applications and Users.
- Cannot Upload and Download the latest updates at right time.

2.2 References

1. Dennis O"Neil(1999). "Blood Components".Palomar College. Archived from the original on June 5,2013.
2. Tuskegee University(May 29, 2013)."Chapter 9 Blood".tuskegee.edu. Archived from the original on December 28, 2013.
3. "Ways to Keep Your Blood Plasma Healthy". Archived from the original on November 1, 2013.Retrieved November 10, 2011.
4. Jump up to Maton, Anthea; Jean Hopkins; Charles Wiliam McLaughlin; Susan Johnson; MaryannaQuon Warner LaHart;David LaHart; Jill D. Wright(1993), Human Biology and Health, Englewood Cliffs,New Jersey,USA.
5. The Physics Factbook— Density of Blood.[6]Basic Biology(2015)."Blood cells".
6. Elkassabany NM, Meny GM, Doria RR, Marcucci C (2008). "Green Plasma Revisited". Anesthesiology 108(4);
7. "19th WHO Model List of Essential Medicines(April 2015)"(PDF). WHO April 2015. RetrievedMay 10, 2015.
8. Tripathi S, Kumar V,Prabhakar A, Joshi S, Agarwal A(2015)."Passive blood plasma separation at the microscale; a review of design principles and microdevices". J.Micromech, Microeng 25(8); 083001.
9. Guo, Weijin; Hansson, Jonas; van der wijngaart, Wouter(2020)."Synthetic Paper Separates Plasma from Whole Blood with Low Protein Loss".Analytical Chemistry.92(9): 6194-6199.
10. Mani A, Poornima AP, Gupta D(2019) "Greenish discoloration of plasma: Is it really a matter of concern?", Asian Journal of Transfusion Science.
11. Starr, Douglas P.(2000), Blood:An Epic History of Medicineand Commerce. New York:Quill.

2.3 Problem Statement Definition

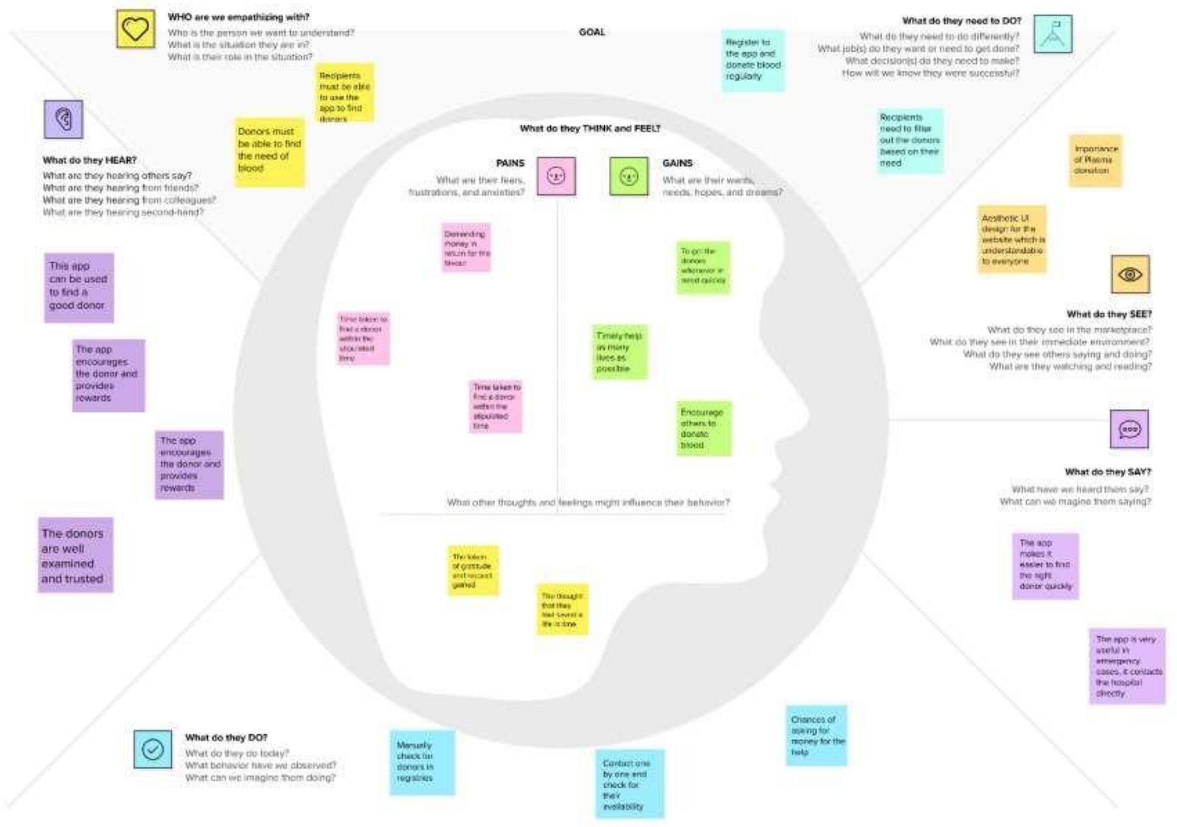
During the COVID 19 crisis, the requirement of plasma became high and the donor count being low. Saving the donor information and helping the need by notifying the current donors would be a helping hand. It is very difficult find the respective blood group donors when anyone is in need. In regard to the problem faced, an application is to be built which would take the donor details store it and inform them upon a request

3. **IDEATION & PROPOSED SOLUTION**

3.1 Empathy Map Canvas

Empathy Map Canvas: An empathy map is a simple, easy-to-digest visual that captures knowledge about a user’s behaviors and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user’s perspective along with his or her goals and challenges.

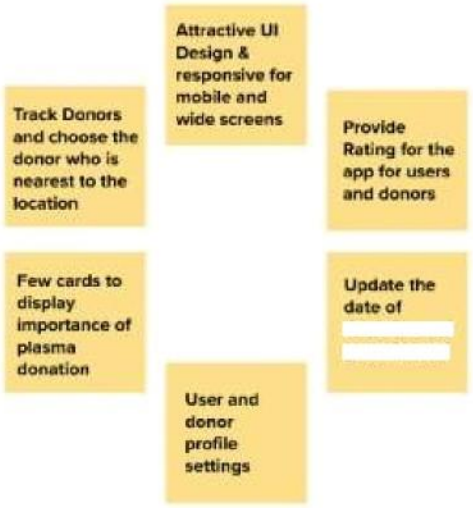
Empathy Map of Plasma Donor Application



WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
GOAL
Register to the app and donate blood regularly
What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
Recipients must be able to use the app to find donors
Donors must be able to find the need of blood
Recipients need to filter out the donors based on their need
Importance of Plasma donation
What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
GAINS
What are their wants, needs, hopes, and dreams?
Demanding money in return for the blood
To get the donors whenever in need quickly
Aesthetic UI design for the website which is understandable to everyone
This app can be used to find a good donor
Time taken to find a donor within the stipulated time
Timely help as many lives as possible
What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
The app encourages the donor and provides rewards
Time taken to find a donor within the stipulated time
Encourage others to donate blood
The app encourages the donor and provides rewards
What do they SAY?
What have we heard them say?
What can we imagine them saying?
What other thoughts and feelings might influence their behavior?
The donors are well examined and trusted
The token of gratitude and respect gained
The thought that they had saved a life in time
The app makes it easier to find the right donor quickly
The app is very useful in emergency cases, it contacts the hospital directly
Chances of asking for money for the help
What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
Manually check for donors in registries
Contact one by one and check for their availability

3.2 Ideation & Brainstorming

Gaunisha Gaanavi G



Attractive UI Design & responsive for mobile and wide screens

Track Donors and choose the donor who is nearest to the location

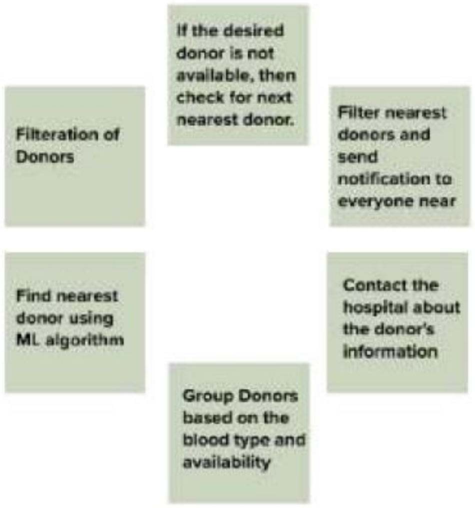
Provide Rating for the app for users and donors

Few cards to display importance of plasma donation

Update the date of

User and donor profile settings

Divya G S



If the desired donor is not available, then check for next nearest donor.

Filteration of Donors

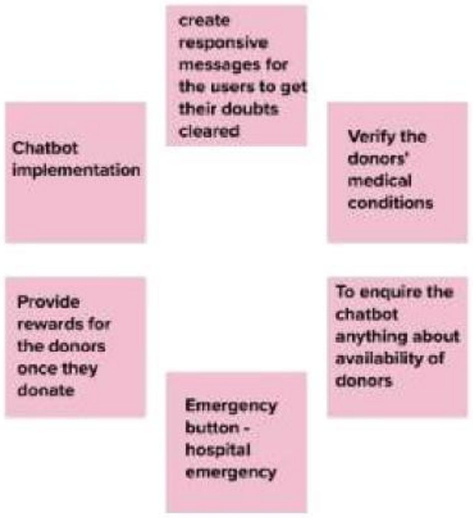
Filter nearest donors and send notification to everyone near

Find nearest donor using ML algorithm

Contact the hospital about the donor's information

Group Donors based on the blood type and availability

Gayathri J 5



create responsive messages for the users to get their doubts cleared

Chatbot implementation

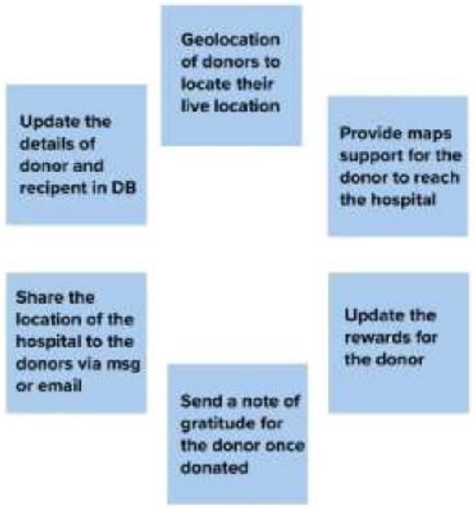
Verify the donors' medical conditions

Provide rewards for the donors once they donate

To enquire the chatbot anything about availability of donors

Emergency button - hospital emergency

Akshayaa D



Geolocation of donors to locate their live location

Update the details of donor and recipent in DB

Provide maps support for the donor to reach the hospital

Share the location of the hospital to the donors via msg or email

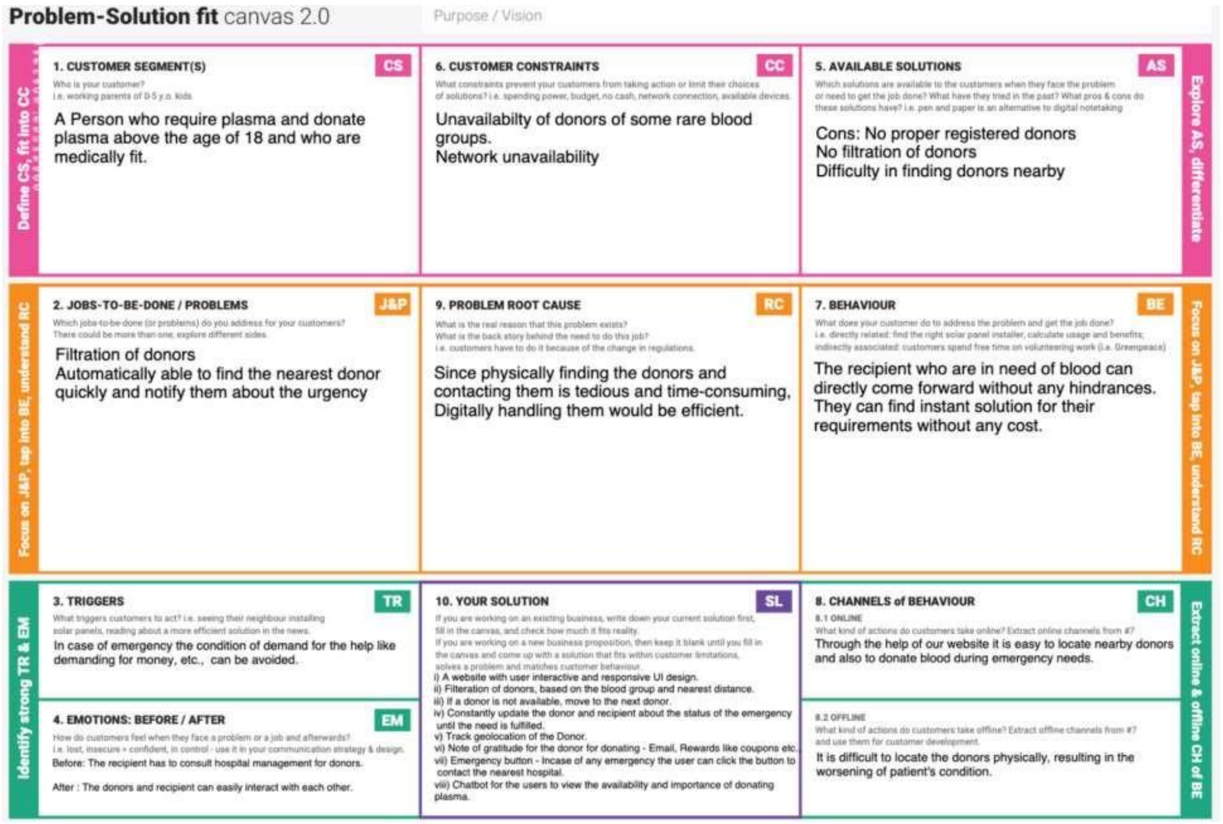
Update the rewards for the donor

Send a note of gratitude for the donor once donated

3.3 Proposed Solution

An application should be developed which would take the donor details, store them and notify them upon a request. A user friendly and responsive interface with a quick notification system which instantly notifies the donor upon receiving a request. The application seamlessly connects the donor and the recipient. It will create an awareness among the people about donation of plasma which will be done in an easy way of connecting the donor and the recipient. And for sure the patient will be satisfied.Since the app is going to be deploy in a cloud kubernetes cluster, it will continue to be efficient when large number of people uses it. There will be no down time.

3.4 Problem Solution fit



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

A Person who require plasma and donate plasma above the age of 18 and who are medically fit.

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices

Unavailabilty of donors of some rare blood groups.
Network unavailability

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

Cons: No proper registered donors
No filtration of donors
Difficulty in finding donors nearby

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

Filtration of donors
Automatically able to find the nearest donor quickly and notify them about the urgency

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

Since physically finding the donors and contacting them is tedious and time-consuming, Digitally handling them would be efficient.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

The recipient who are in need of blood can directly come forward without any hindrances. They can find instant solution for their requirements without any cost.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

In case of emergency the condition of demand for the help like demanding for money, etc., can be avoided.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?
i.e. lost, insecure > confident, in control - use it in your communication strategy & design.

Before: The recipient has to consult hospital management for donors.

After : The donors and recipient can easily interact with each other.

10. YOUR SOLUTION SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

i) A website with user interactive and responsive UI design.
ii) Filteration of donors, based on the blood group and nearest distance.
iii) If a donor is not available, move to the next donor.
iv) Constantly update the donor and recipient about the status of the emergency until the need is fulfilled.
v) Track geolocation of the Donor.
vi) Note of gratitude for the donor for donating - Email, Rewards like coupons etc.
vii) Emergency button - Incase of any emergency the user can click the button to contact the nearest hospital.
viii) Chatbot for the users to view the availability and importance of donating plasma.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from #7

Through the help of our website it is easy to locate nearby donors and also to donate blood during emergency needs.

8.2 OFFLINE

What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

It is difficult to locate the donors physically, resulting in the worsening of patient's condition.

Extract online & offline CH of BE

4. **REQUIREMENT** ANALYSIS

4.1 Functional requirement

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	**Functional Requirement** (Epic)	**Sub Requirement (Story / Sub-Task}**
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR 3	User Certification	Rewards in the form of coupons should be sent to donor email.
FR-4	Searching/reporting Requirements	Chatbot to filter and choose the required donor.
FR-5	Finding donors	Using Ml algorithm for finding nearby donor
FR-6	Eligibility of Donor/Recipient	Check if the donor/recipient is medically fit.
FR-7	Notification	Send notification to all donors in case of emergency

4.2 Non-Functional requirements

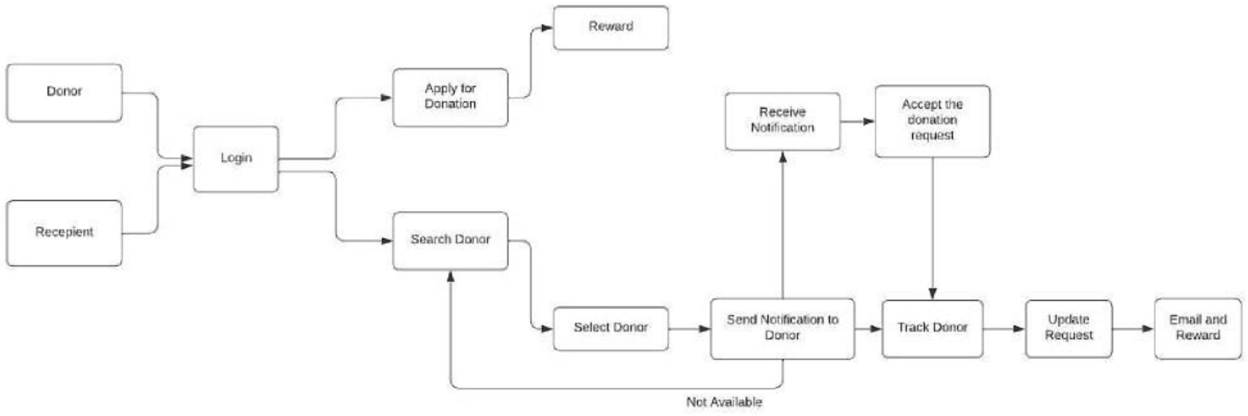
Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

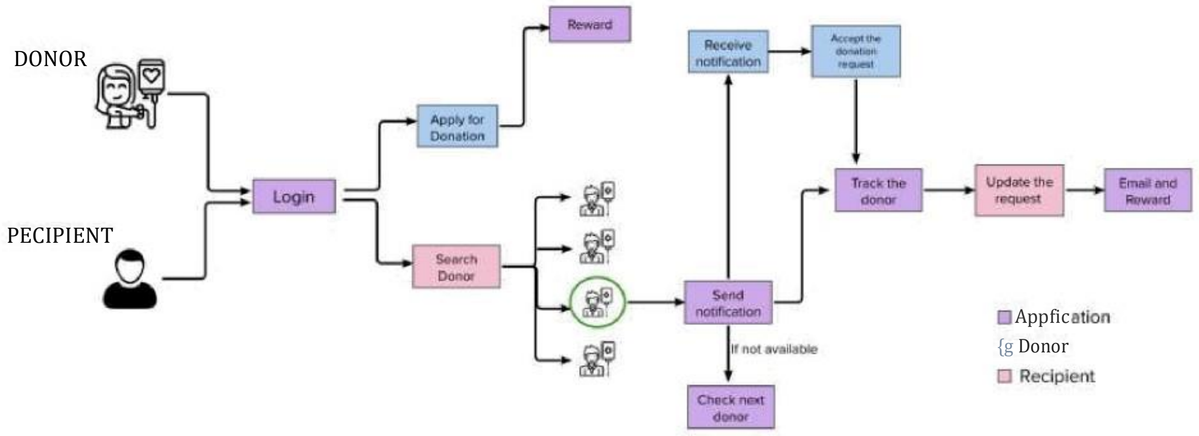
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	User friendly interface with easily accessible, well looking and interactive Ehatbots.
NFR—2	**Security**	Only registered user should be able to view the details of the donor/recipient. Details of the donor/recipient should be kept secure.
NFR—3	Reliability	The system shoul d be built in such a way that it is reliable in its operations as well as to secure the sensitive details.
NFR-4	Performance	The system shaul d be able to handle a large number of users and should nat get disrupted while using the system application.
NFR—5	Availability	The application should be available for all users at all times, and not be disrupted due to any internal issue or server issues
NFR-6	Scalability	The idea can be scaled far further requirements, by adding google calendar support for checking their date of donation.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

User Stories

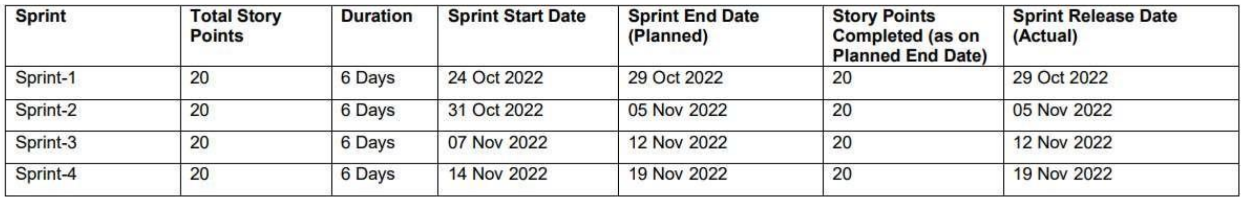
User Type	Functional Requirement **(Epic)**	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dash ooard	High	SprinI-1
		USN-2	As a user, I will receive confirmabon email once I have registered for the application	I can receive canfirmauon email	High	SprinI-1
	Logon	USN-3	As a user, I can log into me application by entering email & password	1 can access ltte portal	Higln	SprinI-1
	Dashboard	USN-4	As a user, I can view my dashboard on successful login	I can access my rewards and medical fitness	Medium	5print-2
	Chatbot	USN-5	As a recipient, 1 can track m y nearby donors	I can choose me donor nearby	High	SprinI-2
		USN-6	As a user, I want to get responsive messages to clear my doubts	I can get to know about tne availability of lhe donors	Medium	Sprint-2
	Navigation	USN-7	As a donor. I can check my meoical eligi oiiity	I can know the eligioilily criteria to donate blood	Medium	Sprint-3
		USA-B	As a recipient, I can access the location of my donor	I can access the geolocation of my donor	Medium	Sprint-3
		USN-9	As a recipient, I can group the donor based on blood type and availability	1 can access ltte avaliable donors easily	Higfi	SprinI-3
		USN-10	As a donor can accept the donation request	I c-an accept or re ect oased an my physical conditions	High	Sprint-4
		USN-11	As a recipient, I can verify the donors medical fimess	I can choose tne nght donor	High	Sprint-4
Admin		USN-12	As a admin, I can view ihe frequent donors	I can access all donors and recepients	Medium	SprinI-4

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement {Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registralion	USN-1	As a user, I can register For the aoplication by entering my email. password, and confirming my password.	2	High	Gaunisha Gaanavi G
Sprinf-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Akshayaa D
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	2	High	Oivya G S
Sprint-2	dashboard	USN-4	As a user, I can view my dasnboard on successful login		Medium	Gayalhn J S
Sprint-2	C halbot	USN-5	As a recipient, I can track my nearoy donors	1	High	Akshayaa D
Sprini-2		USN-6	As a user, 1 want to gel responsive messages to cearmydoubts		Medium	Oivya G S
Sprint-3	Navigalion	USN-7	As a donor, I can check my medical eligibility	3	Medium	Gaunisha Gaanavi G
Sprint-3		USN-8	As a recipient, I can access tne location of my donor	2	Medium	Gayalhn J S
Sprint-3		USN-9	As a recipient, I can group the donor based on blood lype and availability	2	High	Gaunisha Gaanavi G
Sprint-4		US N-10	As a donor I can accepl tne donation reouest	3	High	Gayalhn J S
Sprint-4		USN-11	As a recipient, I can verify the donors medical fitness	3	High	Akshayaa D
Sprint-4		USN-12	As a aomin. I can view the frequent donors	3	Medium	Oivya G S

6.2 Sprint Delivery Schedule



Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIRA

7. CODING & SOLUTIONING (Esplain the features added in the project along with code)

7.1 Feature 1:

ChatBot

A chatbot is instrumental in giving out vital information about blood donation. A simple checklist presented to the user can help them determine whether they are a good fit or not.

We have a pretty laid back attitude when it comes to most things but blood donation takes the prize.

One can donate blood once every three months. But, you do not see people walking in to blood banks to do that.

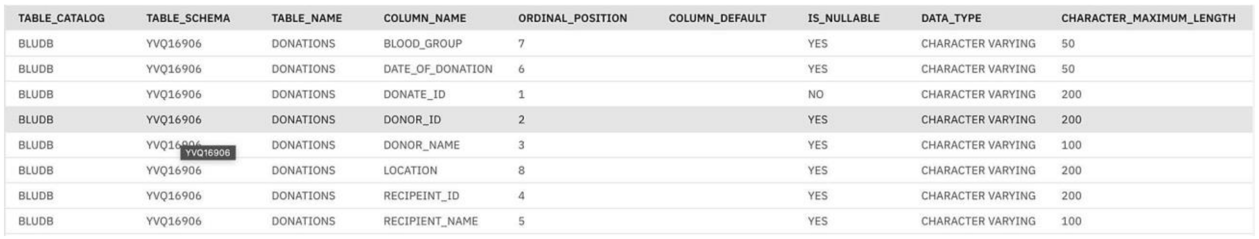
Chatbot are able to send out timely reminders to donors. These reminders can be made as non-intrusive as possible and location based services will be able to direct the donor to the nearest blood bank.

7.2 Feature 2

```
        @app.route(„/create_request”, methods=[„POST”, „GET”]) def create_request():
    if request.method == „POST”: try:
      name = request.form[„name”]email = request.form[„email”]
      blood_group = request.form[„blood_group”] contact_no = request.form[„contact_no”] location =
      request.form[„city”]
       conn = ibm_db.connect(

        „DATABASE=bludb;HOSTNAME=b1bc1829-6f4S-4cd4-bef4-

l0cf08l900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.c1oud;PORT=32304;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0i”, „”, „”)
      sql = “insen into requests (name, email, b1ood  _group, contact  no, location) values(?,?,?,?,?)”param =
      name, email,blood_group,contact_no, location,
      stmt = ibm_db.prepare(conn, sql)ibm_db.execute(stmt, param)
    msg = “You”re successfully made a request!” except Exception as e:
      print(“exception ccurred!”,e)msg = e
    finally:

      return render_template(„donor registration_status.html”, msg = msg)
```

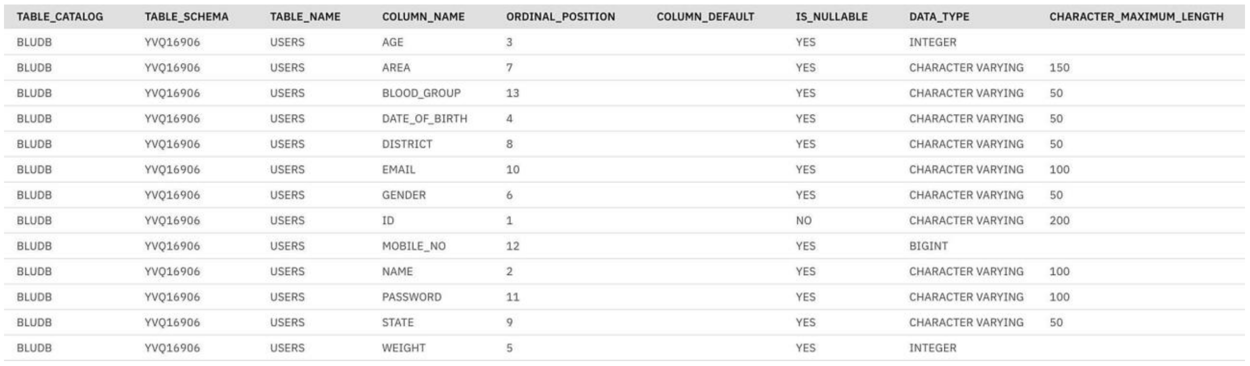
7.3 Database Schema (if Applicable)

SELECT * FROM SYSIBM.COLUMNS WHERE TABLE_NAME = 'DONATIONS”,



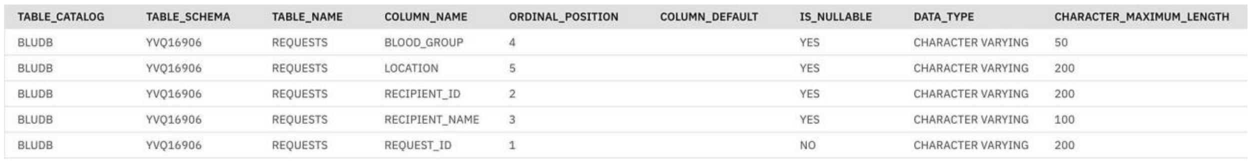
TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
BLUDB	YVQ16906	DONATIONS	BLOOD_GROUP	7		YES	CHARACTER VARYING	50
BLUDB	YVQ16906	DONATIONS	DATE_OF_DONATION	6		YES	CHARACTER VARYING	50
BLUDB	YVQ16906	DONATIONS	DONATE_ID	1		NO	CHARACTER VARYING	200
BLUDB	YVQ16906	DONATIONS	DONOR_ID	2		YES	CHARACTER VARYING	200
BLUDB	YVQ16906	DONATIONS	DONOR_NAME	3		YES	CHARACTER VARYING	100
BLUDB	YVQ16906	DONATIONS	LOCATION	8		YES	CHARACTER VARYING	200
BLUDB	YVQ16906	DONATIONS	RECIPEINT_ID	4		YES	CHARACTER VARYING	200
BLUDB	YVQ16906	DONATIONS	RECIPIENT_NAME	5		YES	CHARACTER VARYING	100

SELECT * FROM SYSIBM.COLUMNS WHERE TABLE_NAME = 'USERS';



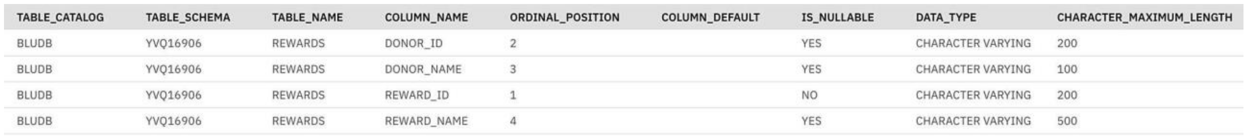
TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
BLUDB	YVQ16906	USERS	AGE	3		YES	INTEGER	
BLUDB	YVQ16906	USERS	AREA	7		YES	CHARACTER VARYING	150
BLUDB	YVQ16906	USERS	BLOOD_GROUP	13		YES	CHARACTER VARYING	50
BLUDB	YVQ16906	USERS	DATE_OF_BIRTH	4		YES	CHARACTER VARYING	50
BLUDB	YVQ16906	USERS	DISTRICT	8		YES	CHARACTER VARYING	50
BLUDB	YVQ16906	USERS	EMAIL	10		YES	CHARACTER VARYING	100
BLUDB	YVQ16906	USERS	GENDER	6		YES	CHARACTER VARYING	50
BLUDB	YVQ16906	USERS	ID	1		NO	CHARACTER VARYING	200
BLUDB	YVQ16906	USERS	MOBILE_NO	12		YES	BIGINT	
BLUDB	YVQ16906	USERS	NAME	2		YES	CHARACTER VARYING	100
BLUDB	YVQ16906	USERS	PASSWORD	11		YES	CHARACTER VARYING	100
BLUDB	YVQ16906	USERS	STATE	9		YES	CHARACTER VARYING	50
BLUDB	YVQ16906	USERS	WEIGHT	5		YES	INTEGER	

SELECT * FROM SYSIBM.COLUMNS WHERE TABLE_NAME = 'REQUESTS“,



TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
BLUDB	YVQ16906	REQUESTS	BLOOD_GROUP	4		YES	CHARACTER VARYING	50
BLUDB	YVQ16906	REQUESTS	LOCATION	5		YES	CHARACTER VARYING	200
BLUDB	YVQ16906	REQUESTS	RECIPIENT_ID	2		YES	CHARACTER VARYING	200
BLUDB	YVQ16906	REQUESTS	RECIPIENT_NAME	3		YES	CHARACTER VARYING	100
BLUDB	YVQ16906	REQUESTS	REQUEST_ID	1		NO	CHARACTER VARYING	200

SELECT * FROM SYSIBM.COLUMNS WHERE TABLE_NAME = 'REWARDS'



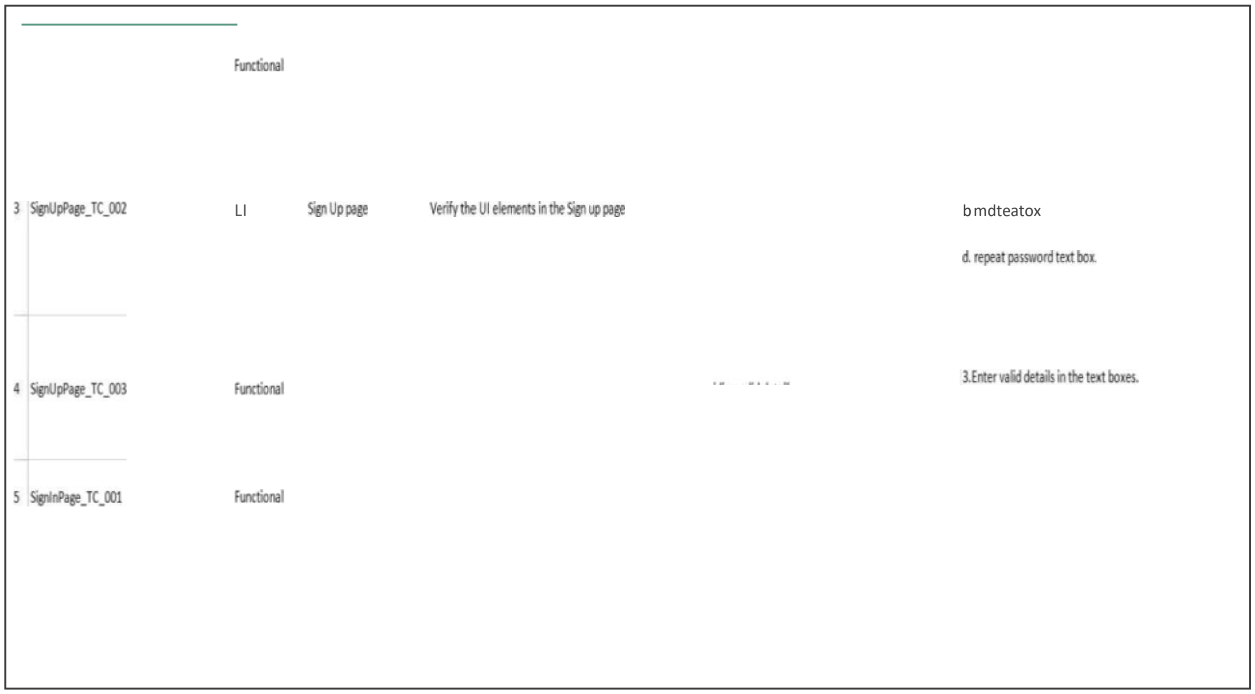
TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	COLUMN_NAME	ORDINAL_POSITION	COLUMN_DEFAULT	IS_NULLABLE	DATA_TYPE	CHARACTER_MAXIMUM_LENGTH
BLUDB	YVQ16906	REWARDS	DONOR_ID	2		YES	CHARACTER VARYING	200
BLUDB	YVQ16906	REWARDS	DONOR_NAME	3		YES	CHARACTER VARYING	100
BLUDB	YVQ16906	REWARDS	REWARD_ID	1		NO	CHARACTER VARYING	200
BLUDB	YVQ16906	REWARDS	REWARD_NAME	4		YES	CHARACTER VARYING	500

8. TESTING

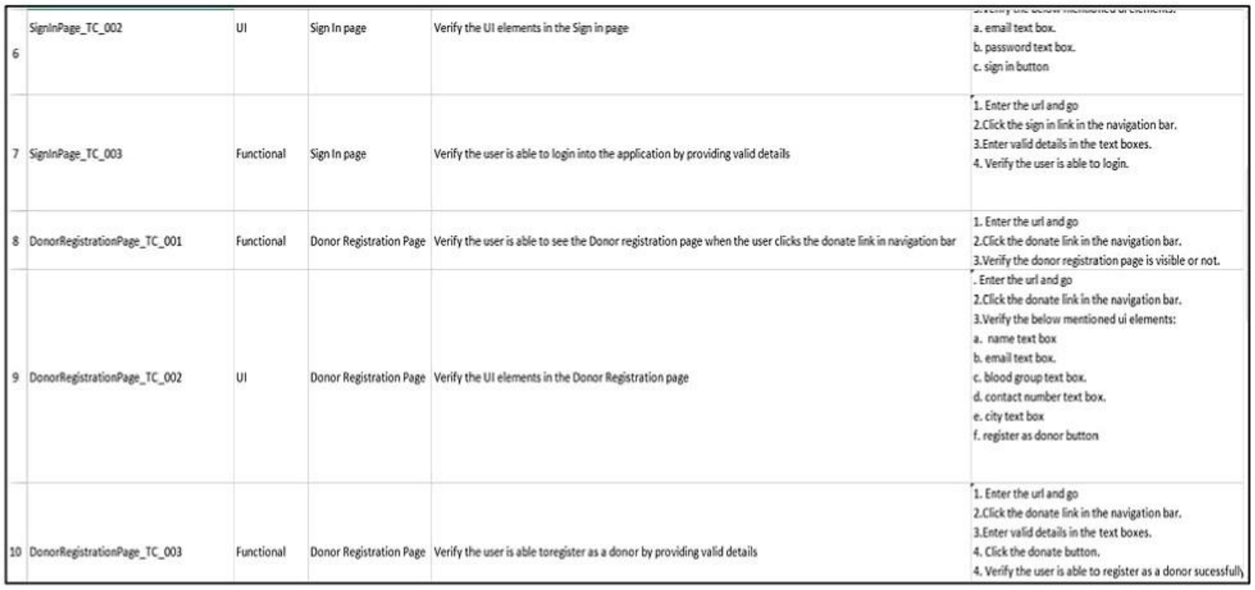
8.1 Test Cases

1	Test Cases	Result
2	Verify the user is able to see the Sign up page when the user clicks the signup button in navigation bar	Positive
3	Verify the UI elements in the Sign up page	Positive
4	Verify the user is able to register into the application by providing valid details	Positive
5	Verify the user is able to see the sign in page when the user clicks the signin button in navigation bar	Positive
6	Verify the UI elements in the Sign in page	Positive
7	Verify the user is able to login into the application by providing valid details	Positive
B	Verify the user is able to see the Donor registration page when the user clicks the donate link in navigation bar	Positive
9	Verify the UI elements in the Donor Registration page	Positive
10	Verify the user is able toregister as a donor by providing valid details	Positive
11	Verify the user is able to see the request page when the user clicks the request link in navigation bar	Positive
12	Verify the UI elements in the request page	Positive
13	Verify the user is able to make a request by providing valid details	Positive
14	Verify the user gets a email notification when they sign up	Positive
is	Verify the dOnor geMa email notification when they make a request	Positive
16	Verifythe donorandrecipientgetsa email notiflcationvvhenthe donor acceptsthe request	Positive
17	Verify the user is able to see the stats page when the user clicks the stage page link in navigation bar	Positive
18	Verify the user is able to interact with the chatbot	Positive

8.2 User Acceptance Testing



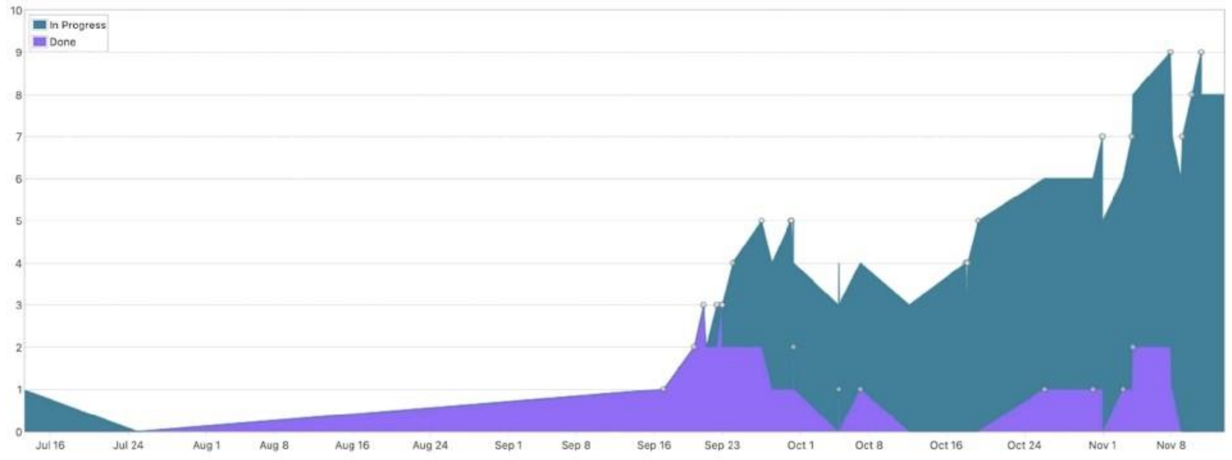
	Functional				
3 SignUpPage_TC_002	LI	Sign Up page	Verify the UI elements in the Sign up page		b mdteatox d. repeat password text box.
4 SignUpPage_TC_003	Functional				3.Enter valid details in the text boxes.
5 SignInPage_TC_001	Functional				



6	SignInPage_TC_002	UI	Sign In page	Verify the UI elements in the Sign in page	3.Verify the below mentioned ui elements: a. email text box. b. password text box. c. sign in button
7	SignInPage_TC_003	Functional	Sign In page	Verify the user is able to login into the application by providing valid details	1. Enter the url and go 2.Click the sign in link in the navigation bar. 3.Enter valid details in the text boxes. 4. Verify the user is able to login.
8	DonorRegistrationPage_TC_001	Functional	Donor Registration Page	Verify the user is able to see the Donor registration page when the user clicks the donate link in navigation bar	1. Enter the url and go 2.Click the donate link in the navigation bar. 3.Verify the donor registration page is visible or not.
9	DonorRegistrationPage_TC_002	UI	Donor Registration Page	Verify the UI elements in the Donor Registration page	. Enter the url and go 2.Click the donate link in the navigation bar. 3.Verify the below mentioned ui elements: a. name text box b. email text box. c. blood group text box. d. contact number text box. e. city text box f. register as donor button
10	DonorRegistrationPage_TC_003	Functional	Donor Registration Page	Verify the user is able toregister as a donor by providing valid details	1. Enter the url and go 2.Click the donate link in the navigation bar. 3.Enter valid details in the text boxes. 4. Click the donate button. 4. Verify the user is able to register as a donor sucessfully

9. RESULTS

9.1 Performance Metrics



10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- It is a user-friendly application.
- It will help people to find plasma easily.
- Simple User Interface
- It alleviates the burden of coordinator to manage Users and resources easily.
- Compared to all other mobile applications, it incorporates provisions for Plasmadonation and

Plasma Requesting.

- Attracts more, number of users as it is available in the form of Mobile applicationinstead of What's app group.
- Usage of this application will greatly reduce time in selecting the right donor.

DISADVANTAGES

- It requires an active internet connection.
- It relays on the details provided by the user.

11. CONCLUSION

Plasma is a liquid portion of blood; it is a mixture of water, proteins and salts. Antibodies are proteins made by the body in response to an infection. People fully rescued from COVID19 are encouraged to donate plasma, which can help to increase the lifespan of other patients because their plasma contains antigens which helps the affected person to recover faster. These immunoglobulin give your immune system a way to fight the virus when you are sick, so your plasma can be used to help others fightoff illness. Individuals must fully resolve symptoms for at least 14 days prior are eligible to donate. Enhanced mobile application for plasma has been developed to help the administrator to attract more donors and recipients and make user management an easy task. This mobile application will attract more users as it is user friendly and greatly reduces scalability issues.Thus, we have successfully designed and developed the Android mobile application to ease the process of becoming a donor and recipient ofPMB bank.

12. FUTURE SCOPE

- A chat widget to establish communication between a donor and recipient .
- To attract more users android application should also be developed in future.

13. APPENDIX

Source Code:

controller.py

```
from flask import *

from flask_mail import *

from datetime import date

from datetime import datetime
```

```
import uuid
from model.model import PlasmaModel


app=Flask(_name_)
app.secret_key = "div"


mail = Mail(app)


app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT']=465
app.config['MAIL_USERNAME'] = '19euit046@skcet.ac.in'
app.config['MAIL_PASSWORD'] = 'gauniganesh'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True


mail = Mail(app)



@app.route('/',methods=["POST","GET"])
def Home():
    if request.method=="GET":
        return  render_template(”Home.html”)


@app.route('/Login',methods=["POST”,"GET”])
def Login():
    obj = PlasmaModel()
    if request.method=="GET":
        return render_template("Login.html")
    elif request.method=="POST":
        email=request.form["email"]
        password=request.form["password"]
        out=obj.get_user_info_email(email)
        if out:
            if out['PASSWORD']==password:
                return redirect(url_for("Landing_home",id=out['ID'])   )
            else:
                flash("Password is wrong.Please enter correct password")
                return render_template("Login.html",email=out['EMAIL'])
        else:
            flash("Email you have entered has not been registered. Please register")
            return render_template("Login.html")

@app.route('/Register',methods=["POST","GET"])
def Register():
    obj = PlasmaModel()
    if request.method=="GET":
        return render_template("sign_up.html")
    elif request.method=="POST":
        Id=uuid.uuid1()
        if int(request.form['age'])<18:
            flash("Age is under than 18. Cannot register")
            return render_template("sign_up.html")
        if int(request.form['weight'])<50:
            flash("Weight is under 50. Cannot register")
            return  render_temp1ate("sign_up.html")


        data={
            ’ID”.str(Id),
            ’NAME“.request.form['username'],
            ’AGE”.request.form[’age'],
            ’DATE_OF_BIRTH”.request.form['dob'],
            ’WEIGHT“.request.form['weight'],
            'GENDER“.request.form['Gender'],
            'AREA“.request.form['area'],
            'DISTRICT”.request.form['District'],
            'STATE“.request.form['State'],
            ’EMAIL“.request.form['email'],
            ’PASSWORD”.request.form['password’],
            ’MOBILE_NO“.request.form['mobileno'],
            ’BLOOD_GROUP“.request.form['bloodgroup']


        obj.insert_into_users(data)
        flash("Successfully Registered! !")
        return render_template("Login.html")


@app.route('/Landing_home/<id>',methods=[”POST","GET"])
def  Landing_home(id):
    if request.method=="GET”:
        return render_template("Landing_Home.html”,id=id)
```

```
@app.route('/donorsearch/<id>',methods=[”POST”,"GET”])
def Donor_Search(id):
   if request.method=="GET":
      return render_template(”Donor_Search.html",id=id)
   elif request.method==”POST":
      obj = PlasmaModel()
      data={
         ’BLOOD_GROUP“.request.form['bloodgroup'],
         'STATE“.request.form['State'],
         'DISTRICT”.request.form['District']

      output=obj.get_user_info_b1oodgroup(data)
                                                                                                                return
     render_temp1ate("Donor_Filter.htm1",data=output,id=id,bloodgroup=request.form['bloodgroup’],state
     =request.form['State’],district=request.form['District'])

@app.route('/DonorFilter/<id>/<fi1ter>/<bloodgroup>/<state>/<district>',methods=["POST","GET"])
def Donor_Filter(id,fi1ter,bloodgroup,state,district):
   obj = PlasmaModel()
   data=(
      'BLOOD_GROUP“.bloodgroup,
      'STATE”.state,
      'DISTRICT“.district

   if request.method=="GET":
      output=obj.get_donor_filter(data,filter)
                                                                                                                return
```

```
    render_template("Donor_Filter.html",data=output,id=id,bloodgroup=bloodgroup,state=state,district=d
    istrict)

@app.route('/Recipient_Filter/<id>/<filter>',methods=["POST",”GET"])
def Recipient_Fi1ter(id):
   obj = PlasmaModel()
   if request.method=="GET”:
      output=obj.get_pendin  requests(id)
      return render_template("Recipient_Filter.htm1",id=id,data=output)

@app.route('/Donate/+id>',methods=["POST","GET"])
def Donate(id):
   obj = PlasmaModel()
   if request.method=="GET":
      output = obj.get_donations_info_id(id)
      return render_template("Recipient_Filter.htm1",id=id,data=output)

@app.route('/location_enter/<donor_id>/<donor_name>/+recipient_id>',methods=["POST","GET"])
def Location_enter(donor_id,donor_name,recipient_id):
   obj = PlasmaModel()
   recipient_info=obj.get_user_info_id(recipient_id)
   if request.method=="GET":
      data={
         ’DONOR_ID“.donor_id,
         ’DONOR_NAME“.donor_name,
         'RECIPIENT_ID”.recipient_id,
         'RECIPIENT_NAME”.recipient_info['NAME'],
         'DATE_OF_DONATION".str(date.today()),
         'BLOOD_GROUP".recipient_info['BLOOD_GROUP'],
         'MOBILE_NO".recipient_info['MOBILE_NO'],
         'DISTRICT".recipient_info['DISTRICT'],
         'STATE".recipient_info['STATE'],
         'STATUS"."Pending"

      return render_template("EnterLocation.html",id=recipient_id,data=data)
   if request.method=="POST":
      Id=uuid.uuidl()
      tableData=(
         'DONATE_ID".str(Id),
         'DONOR_ID".donor_id,
         'DONOR_NAME".donor_name,
         'RECIPIENT_ID".recipient_id,
         'RECIPIENT_NAME".recipient_info['NAME'],
         'DATE_OF_DONATION".str(date.today()),
         'BLOOD_GROUP".recipient_info['BLOOD_GROUP'],
         'LOCATION".request.form['location'],
         'STATUS"."Pending"

      obj.insert_into_donations(tableData)
      # notify donors about the request
                                          msg_to_donor=Message('WE4U    Plasma    Donor
    Application',sender='19euit046@skcet.ac.in',recipients=['19euit046@skcet.ac.in'])
        msg_to_donor.html="<h2>He1lo "+donor_name+",</h2><p>Hope you are doing wel1!</p><p>We
    hereby inform you that you have a request for Plasma by <b>  "+recipient_info['NAME']+"</b>
```

```
    residing at <b>"+recipient_info['AREA']+", ”+recipient_info['DISTRICT'] +”, "+
    recipient_info['STATE']+"</b> <h4>We offer you a sincere thanks! <br>Your contribution will help
    us change lives!</h4><p>If you have any questions or concerns, please don't hesitate to contact us
    we4u@gmail.com. Thank You</p>"
        mail.send(ms   to_donor)

        return render_template("Thankyou_request.html",id=recipient_id)

@app.route('/accept_request/side/<donate_id>/<recipient_id>’,methods=["POST","GET"])
def Accept_request(id,donate_id,recipient_id):
    obj = PlasmaModel()
    if request.method == "GET”:
        obj.update_status_accepted(donate_id)
        donor_info=obj.get_user_info_id(id)
        reward_id=uuid.uuidl()
        data={
            ’REWARD_ID“.str(reward_id),
            'DONOR_ID”.id,
            'DONOR_NAME”.donor_info['NAME'],
            'REWARD_NAME”.'20 Rs CashBack!!'

        obj.insert_into_rewards(data)
        recipient_info=obj.get_user_info_id(recipient_id)
        donate_info=obj.get_donations_info_id(donate_id)

        # send mobile number of donor to recipient
                                        msg_to_recepient=Message('WE4U Plasma Donor
Application',sender='19euit046@skcet.ac.in',recipients=['19euit046@skcet.ac.in'])
        msg_to_recepient.html="<h2>He1lo "+recipient_info['NAME']+",</h2>+p>Hope you are doing well!</p><p>We hereby inform you that the Donor you have requested for Plasma has accepted your request. </p><br><b>Here is the Mobile no of the donor - "+donor_info['MOBILE_NO']+"</b><br> h4>We offer you a sincere thanks and gratitude for choosing our service!</h4>+p>If you have any questions or concerns, please don't hesitate to contact us we4u@gmail.com. Thanks</p>"
 mail.send(msg_to_recepient)

 # send recipient information to donor
                                          msg_to_donor=Message('WE4U Plasma Donor Application',sender='19euit046@skcet.ac.in',recipients=['19euit046@skcet.ac.in'])
              msg_to_donor.html="<h2>Hello "+donor_info['NAME']+",</h2> p>Hope you are doing we1l!</p><p>Thank you coming forward to donate your blood.<b><b>Below mentioned is the address and contact number of the recepient</b><i>Location: "+donate_info['LOCATION']+" Mobile No: "+recipient_info['MOBILE_NO']+"<br><h4>We offer you a sincere thanks and gratitude for choosing our service!</h4><p>If you have any questions or concerns, please don't hesitate to contact us we4u@gmail.com. Thanks</pt"
 mail.send(msg_to_donor)

 # send rewards to donor
                                          msg_to_donor=Message('WE4U Plasma Donor Application',sender='19euit046@skcet.ac.in',recipients=['19euit046@skcet.ac.in'])
         msg_to_donor.html="<h2>Hello ”+donor_info['NAME']+",</h2><p>Thank you for your kind action</p><p>We hereby inform you that we have added a reward <br><b>"+data['REWARD_NAME']+”</b><h4>We offer you a sincere thanks for coming forward in donating plasma!</h4><p>If you have any questions or concerns, please don't hesitate to contact us
```

```
      we4u@gmail.com. Thanks</pt"
       mail.send(msg_to_donor)
       return render_template("Thankyou_request_accepted.html",id=id)

@app.route('/Profile/<id>',methods=["POST”,”GET”])
def Profile(id):
   obj=PlasmaModel()
   if request.method=="GET":
      output=obj.get_user_info_id(id)
      return render_template("Profile.htm1”,id=id,data=output)
   elif request.method=="POST":
      data=(
         ’NAME“.request.form['username'],
         ’AGE”.request.form[’age'],
         ’DATE_OF_BIRTH”.request.form['dob'],
         ’WEIGHT“.request.form['weight'],
         ’GENDER“.request.form['Gender'],
         'AREA“.request.form['area'],
         'DISTRICT”.request.form['District'],
         'STATE“.request.form['State'],
         ’EMAIL“.request.form['email'],
         ’PASSWORD”.request.form['password'],
         ’MOBILE_NO“.request.form['mobileno'],
         ’BLOOD_GROUP“.request.form['bloodgroup']

      data=obj.update_user_info(data,id)
      return render_template("Profile.htm1”,id=id,data=data)
```

```
@app.route('/donate_history/<id>',methods=["POST","GET"])
def Donated_history(id):
    obj=PlasmaModel()
    if request.method == "GET":
        output=obj.get_completed_donations(id)
        return render_template("Donated_History.html”,id=id,data=output)
@app.route('/Recipient_request/<id>',methods=["POST",”GET"])
def Recipient_request(id):
    obj = PlasmaModel()
    if request.method=="GET":
        output=obj.get_pending_requests(id)
        return render_template(”Recipient_requests.html”,id=id,data=output)

@app.route('/Get_rewards/<id>',methods=[”POST","GET"])
def Get_rewards(id):
    obj=PlasmaModel()
    if request.method=="GET":
        output = obj.get_rewards(id)
        return render_template("Rewards.html",id=id,data=output)



@app.route('/Logout',methods=["POST","GET"])
def Logout():
    if request.method=="GET":
        return render_template("Home.html”)
```

```
if(_name==”main_"):
    app.run(debug=True)
```

Model.py

```
import ibm_db


dsn_hostname                                                "815fa4db-dc03-4c70-869a-
      a9ccl3f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud"
dsn_uid = "yvql6906"
dsn_pwd = "ZMgfXgE7YvDXLbX4"
dsn_security="SSL"
dsn_SSLServerCertificate="DigiCertG1obalRootCA.crt"
dsn_database = "BLUDB"
dsn_port = "30367"


dsn = (
    "DATABASE=(0};"
    "HOSTNAME={1};"
    "PORT={2};"
    "SECURITY=(3};"
    "SSLServerCertificate=(4};"
    "UID={5};"
    "PWD={6};"
).format(dsn_database,dsn_hostname,dsn_port,dsn_security,dsn_SSLServerCertificate,dsn_uid,dsn_pwd)


try:
        conn        ibm_db.connect('DATABASE=bludb;HOSTNAME=815fa4db-dc03-4c70-869a-
      a9ccl3f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30367;SECURITY=SSL;S
      SLServerCertificate=DigiCertGlobalRootCA.crt;UID=yvq16906;PWD=ZMgfXgE7YvDXLbX4', '', '')
    print(" Connected to database : ",dsn_database,"as user: ", dsn_uid," on host: ”,dsn_hostname)
except:
    print("Unable to connect: ",ibm_db.conn_errormsg())


class PlasmaModel:
    def _init_(self):
        self.users=dsn_uid+".USERS"
        self.donations=dsn_uid+".DONATIONS"
        self.requests=dsn_uid+".REQUESTS"
        self.rewards=dsn_uid+".REWARDS"



    def insert_into_users(self,data):
                                        statement         "insert     into     "+self.users+"
      values('"+data['ID']+'",'"+data['NAME']+"',"+data['AGE']+",'"+data['DATE_OF_BIRTH']+"',"+data['
      WEIGHT']+",'"+data['GENDER']+"','"+data['AREA']+"','"+data['DISTRICT']+"','"+data['STATE']+"',
      '"+data['EMAIL']+"','"+data['PASSWORD']+"',"+data['MOBILE_NO']+",'"+data['BLOOD_GROUP']

        print(statement)
        result = ibm_db.exec_immediate(conn,statement)
        print("inserted---> to table ",self.users )

    def get_user_info_email(self,email):
        statement = "select * from "+self.users+" where EMAIL='"+emai1+"'"
        print(statement)
        result = ibm_db.exec_immediate(conn,statement)
        if result:
            resultset=ibm_db.fetch_both(result)
            print(resultset)
            return resultset
        else:
            return None

    def  get_user_info_id(se1f,id):
        statement = "select * from "+self.users+" where ID='"+id+"'"
        print(statement)
        result = ibm_db.exec_immediate(conn,statement)
        if result:
            resultset=ibm_db.fetch_both(result)
            print(resultset)
            return resultset
        else:
            return None

    def update_user_info(self,data,id):

      update_value="NAME='"+data['NAME’]+"',AGE="+data['AGE']+",DATE_OF_BIRTH=”’+data['DA
      TE_OF_BIRTH']+"',WEIGHT="+data['WEIGHT']+",GENDER='"+data['GENDER']+"',AREA='"+d
      ata['AREA']+"',DISTRICT='"+data['DISTRICT']+"',STATE='"+data['STATE']+"',EMAIL=’”+data['E
      MAIL’]+"',PASSWORD='"+data['PASSWORD']+"',MOBILE_NO=”+data[’MOBILE_NO']+”,BLOO
      D_GROUP='"+data['BLOOD_GROUP']+"'"
        statement = "update "+self.users+" set "+update_value+"where ID='"+id+’”;"
        print(statement)
        result = ibm_db.exec_immediate(conn,statement)
        if result:
            resultset=self.get_user_info_id(id)
            print(resultset)
            return resultset
        else:
            return None

    def get_user_info_bloodgroup(se1f,data):
           statement = "select * from "+self.users+" where BLOOD_GROUP=’"+data['BLOOD_GROUP']+"'
      and STATE=’"+data['STATE']+"' and DISTRICT='"+data['DISTRICT']+"’"
        print(statement)
        result = ibm_db.exec_immediate(conn,statement)
        result_fetch=ibm_db.fetch_both(result)
        resultset=[]
        if result_fetch:
            resultset.append(result_fetch)
            resultset=[dict(r) for r in resultset] if resultset else None
            print(resultset)
            return resultset
        else:
            return None

    def insert_into_donations(self,data):
                                            statement         "insert     into     "+self.donations+"
      values(’"+data['DONATE_ID']+"',’"+data['DONOR_ID']+"',’"+data['DONOR_NAME']+"',’"+data['RE
      CIPIENT_ID']+"',’"+data['RECIPIENT_NAME']+"',’"+data['DATE_OF_DONATION']+"',’"+data['B
    LOOD_GROUP']+"','"+data['LOCATION']+"','"+data['STATUS']+"')"
      print(statement)
      result = ibm_db.exec_immediate(conn,statement)
      print("inserted---> to table ",self.donations )

    def get_donor_filter(self,data,filter):
      if filter == "agelth":
          statement = "select * from "+self.users+" where BLOOD_GROUP='"+data['BLOOD_GROUP']+"' and STATE='"+data['STATE']+"' and DISTRICT='"+data['DISTRICT']+"' order by AGE desc"
      elif filter == "agehtl":
          statement = "select * from "+self.users+" where BLOOD_GROUP='"+data['BLOOD_GROUP']+"' and STATE='"+data['STATE']+"' and DISTRICT='"+data['DISTRICT']+"' order by AGE asc"
      elif filter == "genderm":
          statement = "select * from "+self.users+" where BLOOD_GROUP='"+data['BLOOD_GROUP']+"' and STATE='"+data['STATE']+"' and DISTRICT='"+data['DISTRICT']+"' and GENDER = 'Male'"
      elif filter == "genderf":
          statement = "select * from "+self.users+" where BLOOD_GROUP='"+data['BLOOD_GROUP']+"' and STATE='"+data['STATE']+"' and DISTRICT='"+data['DISTRICT']+"' and GENDER = 'Female'"
      else:
          statement = "select * from "+self.users+" where BLOOD_GROUP='"+data['BLOOD_GROUP']+"' and STATE='"+data['STATE']+"' and DISTRICT='"+data['DISTRICT']+"'"
      print(statement)
```

```
    result = ibm_db.exec_immediate(conn,statement)
    result_fetch=ibm_db.fetch_both(result)
    resultset=[]
    if result_fetch:
        resultset.append(result_fetch)
        resultset=[dict(r) for r in resultset] if resultset else None
        print(resultset)
        return resultset
    else:
        return None

def get_donations_info_id(se1f,id):
    statement = ”select * from "+self.donations+" where DONOR_ID='"+id+"'"
    print(statement)
    result = ibm_db.exec_immediate(conn,statement)
    result_fetch=ibm_db.fetch_both(result)
    resultset=[]
    if result_fetch:
        resultset.append(result_fetch)
        resultset=[dict(r) for r in resultset] if resultset else None
        print(resultset)
        return resultset
    else:
        return None

def update_status_accepted(self,donate_id):
                statement = ”update "+self.donations+" set STATUS='Completed' where
```

```
  DONATE_ID='"+donate_id+"'"
   print(statement)
   result = ibm_db.exec_immediate(conn,statement)
   print("Updated-->"+self.donations)

def insert_into_rewards(self,data):
                                          statement          "insert      into      "+self.rewards+"
  values('"+data['REWARD_ID']+"','"+data['DONOR_ID']+"','"+data['DONOR_NAME']+"',''+data['R
  EWARD_NAME']+"')"
   print(statement)
   result = ibm_db.exec_immediate(conn,statement)
   print("inserted---> to table ",self.rewards )

def get_completed_donations(self,id):
      statement = "select * from "+self.donations+" where DONOR_ID=''+id+"' and STATUS =
  'Completed","
   print(statement)
   result = ibm_db.exec_immediate(conn,statement)
   result_fetch=ibm_db.fetch_both(result)
   resultset=[]
   if result_fetch:
     resultset.append(result_fetch)
     resultset=[dict(r) for r in resultset] if resultset else None
     print(resultset)
     return resultset
   else:
     return None
```

```
def get_pending_requests(self,id):
        statement = "select * from "+self.donations+" where DONOR_ID='"+id+"' and STATUS =
  'Pending';"
   print(statement)
   result = ibm_db.exec_immediate(conn,statement)
   result_fetch=ibm_db.fetch_both(result)
   resultset=[]
   if result_fetch:
      resultset.append(result_fetch)
      resultset=[dict(r) for r in resultset] if resultset else None
      print(resultset)
      return resultset
   else:
      return None

def get_rewards(self,id):
   statement = "select * from "+self.rewards+" where DONOR_ID='"+id+"';"
   print(statement)
   result = ibm_db.exec_immediate(conn,statement)
   result_fetch=ibm_db.fetch_both(result)
   resultset=[]
   if result_fetch:
      resultset.append(result_fetch)
      resultset=[dict(r) for r in resultset] if resultset else None
      print(resultset)
      return resultset
```

```
    else:
        return None
```

GitHub & Project Demo Link

GitHub Link: https://github.com/IBM-EPBL/IBM-Project-24435-1659942826

Project Demo Link

https://drive.google.com/file/d/1FhbNSXuuTFQ19SP5L7QBnOstT6inb5La/view?usp=sharing