IBM – NALAIYA THIRAN PROJECT

PLASMA DONOR APPLICATION

PROJECT REPORT

INDUSTRY MENTOR : NAVYA

FACULTY MENTOR:KARTHIKEYAN P

TEAM ID : PNT2022TMID05166

TEAM LEAD : CINTHAMANI KN(921319106039)

TEAM MEMBER : BRINDHA V(921319106036)

TEAM MEMBER : BRUNISHA P(921319106037)

TEAM MEMBER : DIVYADHARSHINI R(921319106053)

PSNA COLLEGE OF ENGINEERING AND TECHNOLOGY
(An Autonomous Institution, Affiliated to Anna University Chennai - 600 025)

NOVEMBER 2022

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

1.1 OVERVIEW

During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low. The main goal of our project is to design a user- friendly web application which helps those who are need of plasma , plasma therapy is an experimental approach to treat those COVID-positive patients and help them recover faster. Therapy, which is considered reliable and safe. If a particular person has fully recovered from COVID19, they are eligible to donate their plasma. As we all know, the traditional methods of finding plasma, one has to find out for oneself by looking at hospital records and contacting donors have been recovered, sometimes may not be available at home and move to other places. In this type of scenario, the health of those who are sick becomes disastrous. Therefore, it is not considered a rapid process to find plasma.

1.2 PURPOSE

A user friendly and responsive interface with a quick notification system which instantly notifies the donor upon receiving a request.The main purpose of the proposed system, the donor who wants to donate plasma can register and can donate the plasma to the blood bank, the recipient can request for the donor and once the donor has accepted the request, the donor can donate blood at blood bank and the recipient can also track the status of the request for plasma and can take the plasma from the blood bank.

LITERATURE SURVEY

2.1 EXISTING PROBLEM

During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low. Saving the donor information and helping the needy by notifying the current donors list, would be a helping hand . Alternatively, now a day‟s plasma transplant surgery is also being performed rapidly. At this present time plasma banks are in short supply. Not only that, but the number of plasma donors is lowtoo. And some people do not know what plasma donation is and where to donateplasma.

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; Maryanna Quon 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. Retrieved May 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 Medicine and Commerce. New York:Quill.

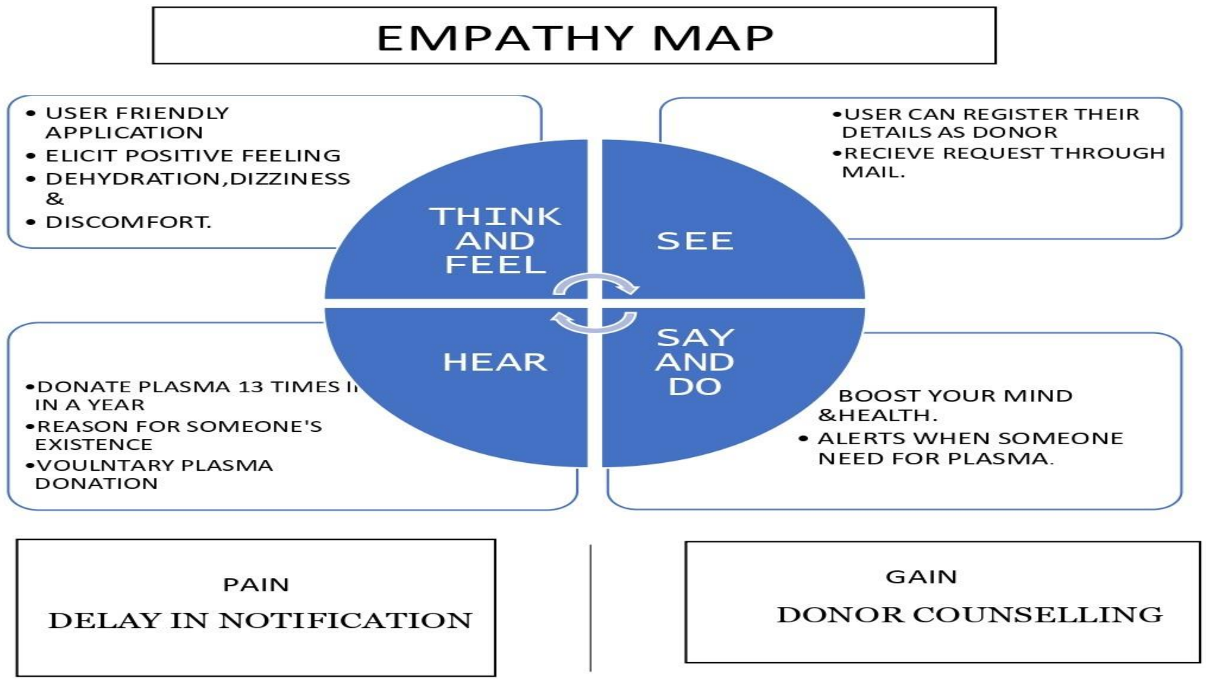
2.3 PROBLEM STATEMENT DEFINITION

During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low. Saving the donor information and helping the needyby notifying the current donors list, would be a helping hand. In regard to the problem faced,an application is to be built which would take the donor details, store them and inform them upon a request**.** The requirement of plasma became a high priority and the donor count has become low. Saving the donor information and helping the needy by notifying the current donors list, would be a helping hand. An application should be developed which would take the donor details, store them and notify them upon a request.

IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

- An empathy map is a simple, easy-to-digest visual that captures knowledge about a user‟s behaviours 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
• USER FRIENDLY APPLICATION
• ELICIT POSITIVE FEELING
• DEHYDRATION,DIZZINESS &
• DISCOMFORT.
•USER CAN REGISTER THEIR DETAILS AS DONOR
•RECIEVE REQUEST THROUGH MAIL.
THINK AND FEEL
SEE
HEAR
SAY AND DO
•DONATE PLASMA 13 TIMES IN A YEAR
•REASON FOR SOMEONE'S EXISTENCE
•VOULNTARY PLASMA DONATION
BOOST YOUR MIND &HEALTH.
• ALERTS WHEN SOMEONE NEED FOR PLASMA.
PAIN
DELAY IN NOTIFICATION
GAIN
DONOR COUNSELLING

Fig 1.1.Empathy Map

3.2 IDEATION & BRAINSTORMING

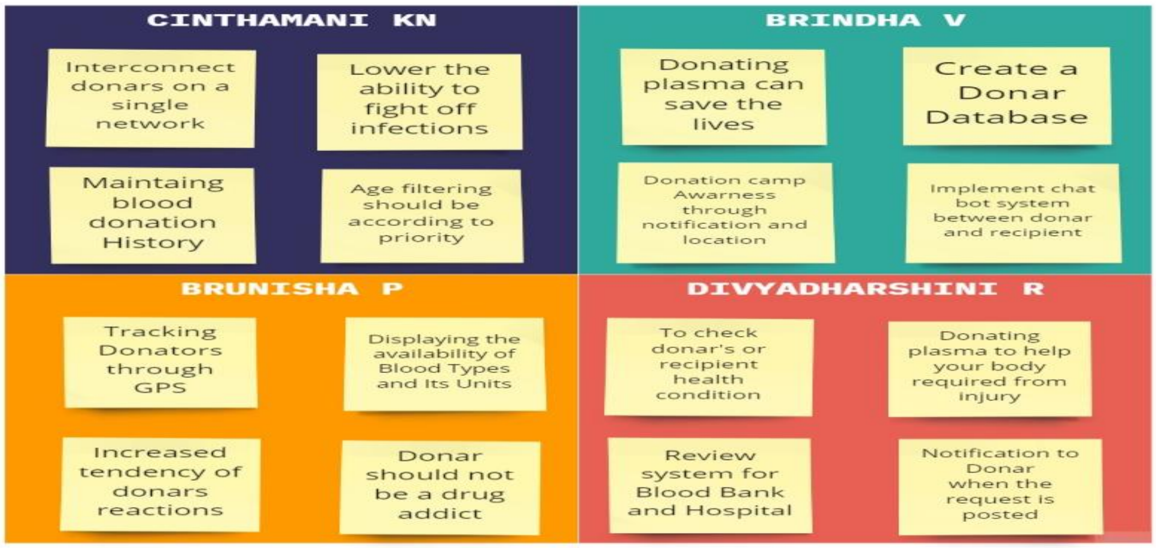

CINTHAMANI KN
Interconnect donars on a single network
Lower the ability to fight off infections
Maintaing blood donation History
Age filtering should be according to priority
BRINDHA V
Donating plasma can save the lives
Create a Donar Database
Donation camp Awarness through notification and location
Implement chat bot system between donar and recipient
BRUNISHA P
Tracking Donators through GPS
Displaying the availability of Blood Types and Its Units
Increased tendency of donars reactions
Donar should not be a drug addict
DIVYADHARSHINI R
To check donar's or recipient health condition
Donating plasma to help your body required from injury
Review system for Blood Bank and Hospital
Notification to Donar when the request is posted

Fig 1.2.Problem Statement & Brainstorm

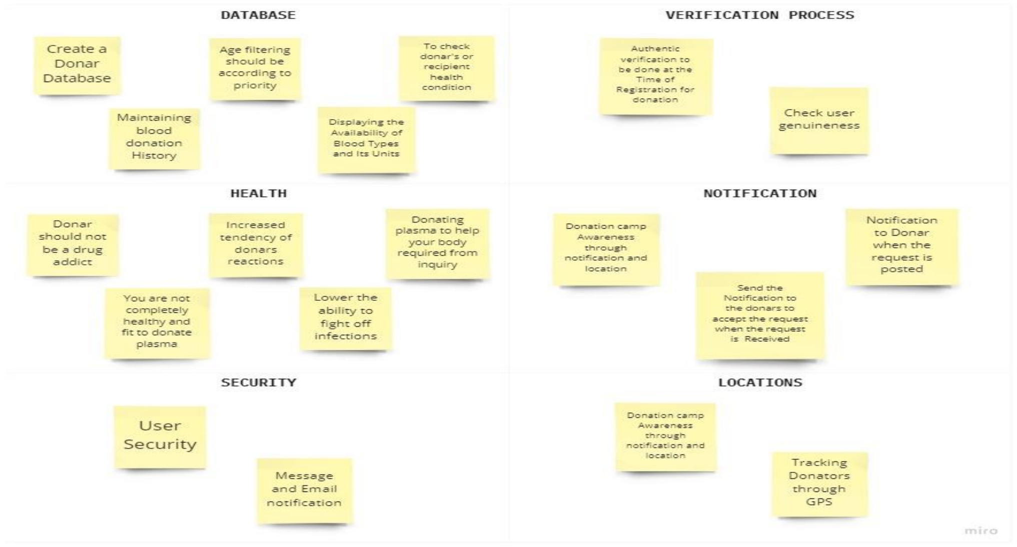
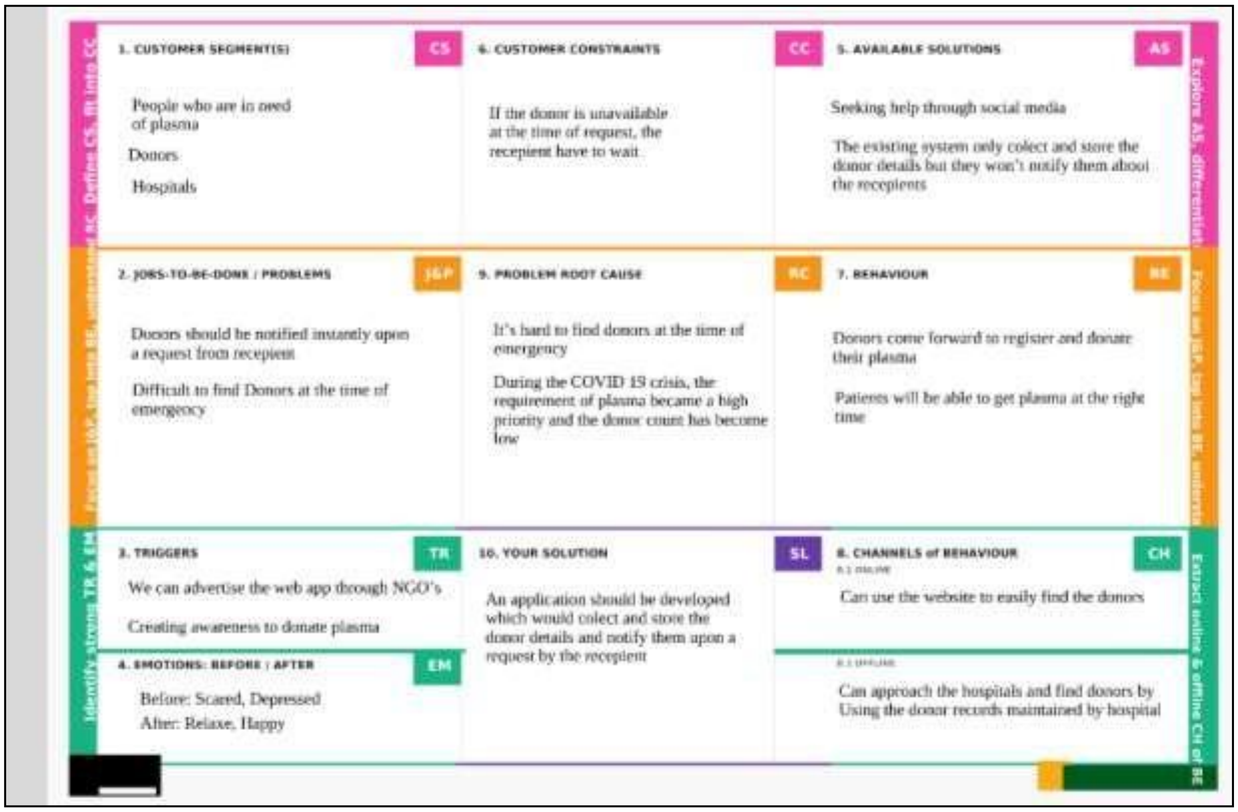

DATABASE
Create a Donar Database
Age filtering should be according to priority
To check donar's or recipient health condition
Maintaining blood donation History
Displaying the Availability of Blood Types and Its Units
VERIFICATION PROCESS
Authentic verification to be done at the Time of Registration for donation
Check user genuineness
HEALTH
Donar should not be a drug addict
Increased tendency of donars reactions
Donating plasma to help your body required from inquiry
You are not completely healthy and fit to donate plasma
Lower the ability to fight off infections
NOTIFICATION
Donation camp Awareness through notification and location
Notification to Donar when the request is posted
Send the Notification to the donars to accept the request when the request is Received
SECURITY
User Security
Message and Email notification
LOCATIONS
Donation camp Awareness through notification and location
Tracking Donators through GPS
miro

Fig 1.3.Group Ideas & Prioritize

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 anawareness 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


Explore AS, differentiate
Identify strong TR & EM
Extract online & offline CH of BE

1. CUSTOMER SEGMENT(S) CS

People who are in need of plasma

Donors

Hospitals

6. CUSTOMER CONSTRAINTS CC

If the donor is unavailable at the time of request, the recepient have to wait

5. AVAILABLE SOLUTIONS AS

Seeking help through social media

The existing system only colect and store the donor details but they won't notify them about the recepients

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

Donors should be notified instantly upon a request from recepient

Difficult to find Donors at the time of emergency

9. PROBLEM ROOT CAUSE RC

It's hard to find donors at the time of emergency

During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low

7. BEHAVIOUR BE

Donors come forward to register and donate their plasma

Patients will be able to get plasma at the right time

3. TRIGGERS TR

We can advertise the web app through NGO's

Creating awareness to donate plasma

4. EMOTIONS: BEFORE / AFTER EM

Before: Scared, Depressed

After: Relaxe, Happy

10. YOUR SOLUTION SL

An application should be developed which would colect and store the donor details and notify them upon a request by the recepient

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

Can use the website to easily find the donors

8.2 OFFLINE

Can approach the hospitals and find donors by Using the donor records maintained by hospital

Fig 1.4.Problem Solution fit

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

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 Website
FR-2	User Confirmation	Confirmation via Email
FR-3	User Login	Login through registered email id
FR-4	Send Request	Patient should fill their details and make a request
FR-5	Contact Donor	Donor and Patient contact by the details shared via email

4.2 NON-FUNCTIONAL REQUIREMENTS

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

NFR No.	**Non Functional Requirement (Epic)**	**Description**
NFR-1	Usability	The plasma Donor application is user friendly and does not involve any complex process
NFR-2	Security	The donor/recipient details are stored in a secured cloud based database.
NFR-3	Reliability	The application will have no down time so that you can always rely on and the information provided by it are so reliable
NFR-4	Performance	The application will work efficiently in emergency situations with an instant notification system.
NFR-5	Availability	The application will be available online 24x7
NFR-6	Scalability	The application can be accessed by multiple users at the same time and it has the ability to increase or decrease the IT resources as needed.

PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

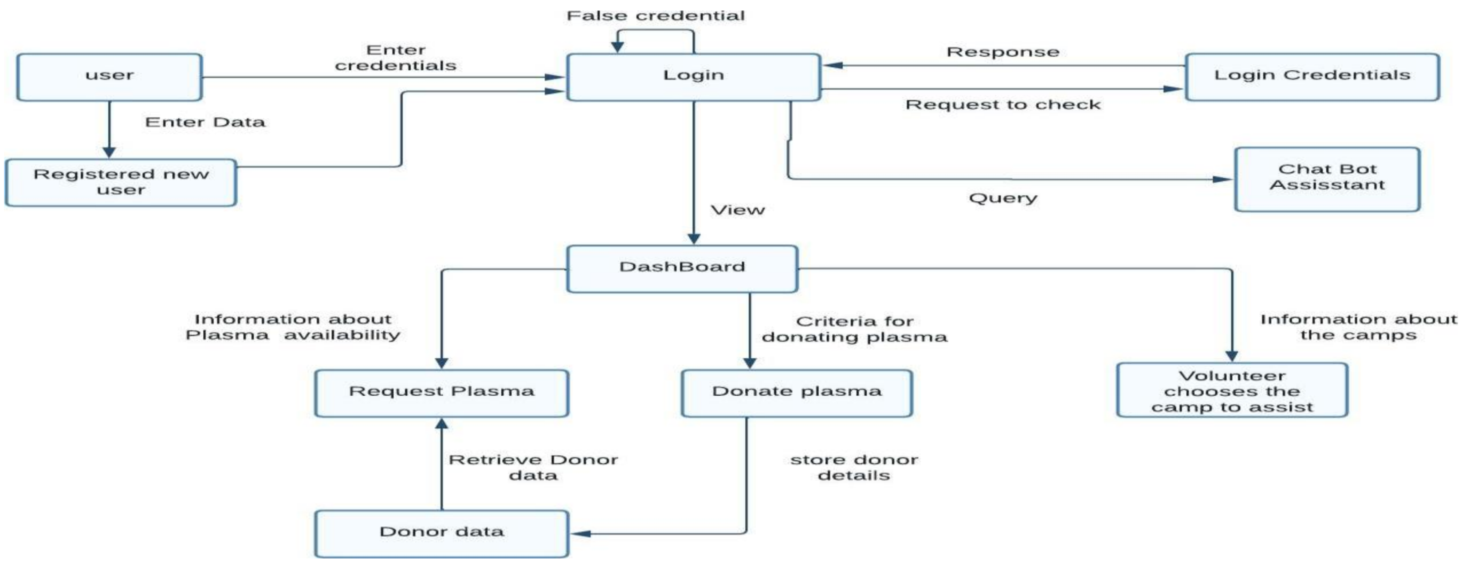


Fig 2.1.Data Flow Diagram

5.2 SOLUTION & TECHNICAL ARCHITECTURE

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. When the recipient requests for plasma, if there is lack of plasma at the time of request, automatically the recipient will be added to the waiting list. Later when there is availability of plasma, the recipient will be notified by email.

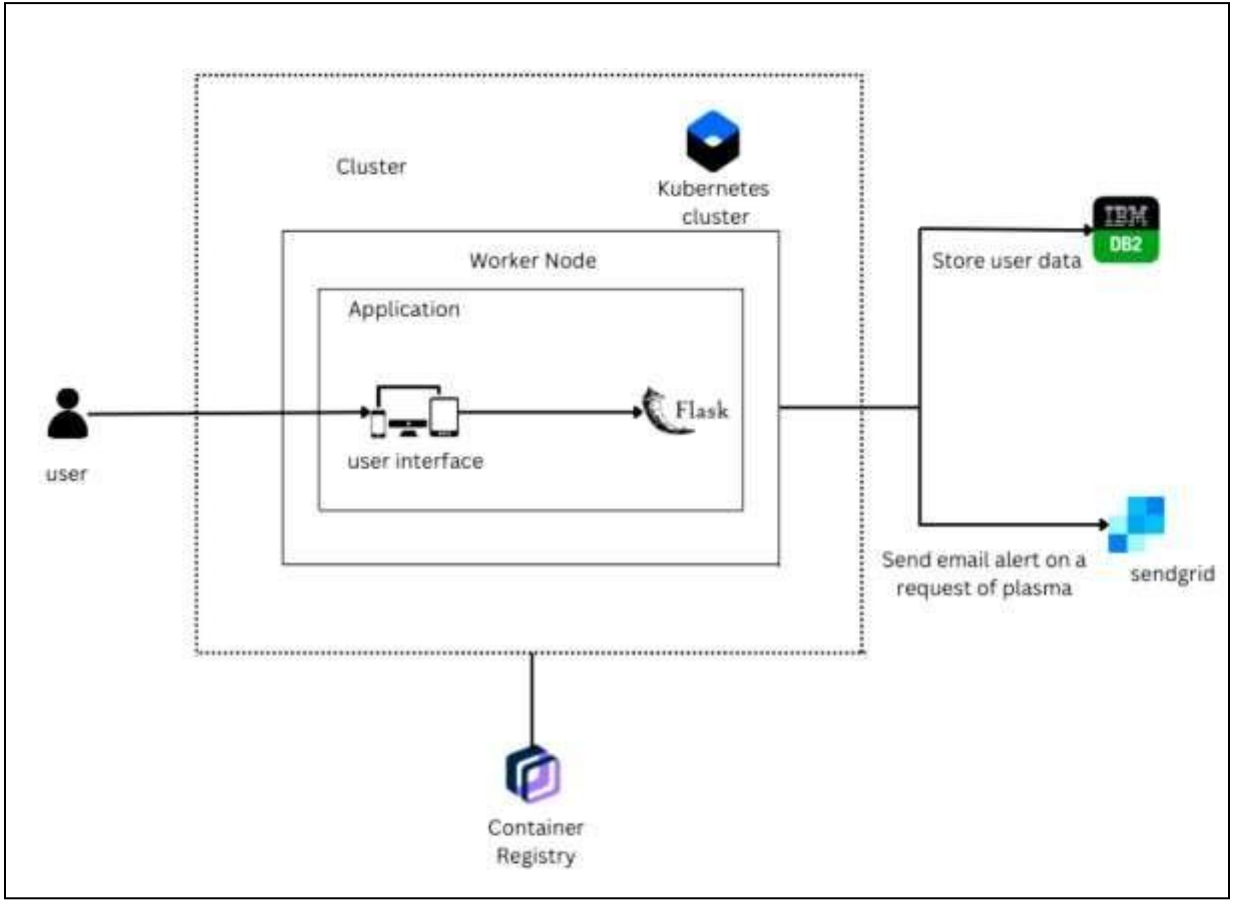


Fig 2.2. Architecture diagram

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	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 / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through	I can receive confirmation	Medium	Sprint-1

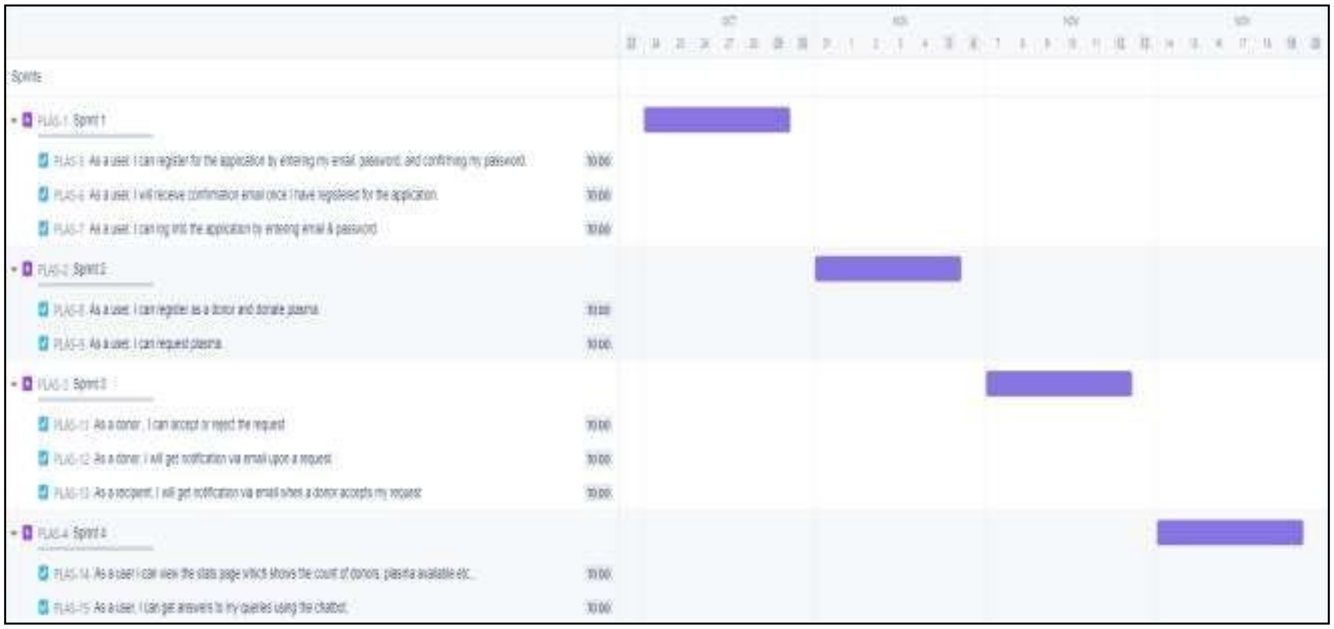
			Gmail	notifications through Gmail		
	Login	USN-4	As a user, I can log into the application by entering email & password	I can access into my User profile and view details in dashboard	High	Sprint-1
	Dashboard	USN-5	As a user,I can donate and request plasma.	I can receive appropriate notifications through email	High	Sprint-1
Customer (Web user)	Login	USN-6	As a user, I can register and log into the application by entering email & password to view the profile	I can access into my user profile and view details in dashboard	High	Sprint-1
	Dashboard	USN-7	As a user,I can donate and request plasma.	I can receive appropriate notifications through email	High	Sprint-1
Customer Care Executive	Application	USN-8	As a customer care executive, I can try to address user‟s concerns and questions	I can view and address their concerns andquestions	Medium	Sprint-2
Administrator	Application	USN-9	As an administrator, I can listen to feedbacks and make the user interface more friendly and make complex process simple.	I can change	Medium	Sprint-3
		USN-10	As an administrator, I can involve working with the technical side of websites.	I can help with troubleshooting bugs and provid e a seamless experience.	Medium	Sprint-1
Chatbot	Dashboard	USN-11	In addition the Customer care executive, chatbot can try to address user‟s concerns and questions	It can reply to all the queries related to our application	Medium	Sprint-3

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	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirmingmy password.	2	High	Cinthamani, Brindha
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Cinthamani, Brindha
Sprint-1	Login	USN-3	As a user, I can log into the application by entering email & password	2	High	Cinthamani, Brindha
Sprint- 2	Dashboard	USN-4	As a user, I can register as a donor anddonate plasma	3	High	Cinthamani Brunisha
Sprint- 2		USN-5	As a user, I can request plasma	2	High	Cinthamani Brunisha
Sprint- 4		USN-6	As a user I can view the statspage which shows the count ofdonors, plasma available etc.,	2	Medium	Cinthamani, Brindha
Sprint- 3		USN-7	As a donor , I can accept or reject the request	2	High	Cinthamani
Sprint - 4	Chatbot	USN-8	As a user, I can get answers to my queriesusing the chatbot	2	Medium	Divyadharshi
Sprint-3	Notification	USN-9	As a donor, I will get notification via email upon a request	1	High	Cinthamani Brunisha
Sprint-3		USN-10	As a recipient, I will get notification once my request is accepted	1	High	

6.2 SPRINT DELIVERY SCHEDULE



Sprints

PLAS-1 Sprint 1

PLAS-5 As a user, I can register for the application by entering my email, password, and confirming my password. 10:00

PLAS-6 As a user, I will receive confirmation email once I have registered for the application. 10:00

PLAS-7 As a user, I can log into the application by entering email & password 10:00

PLAS-2 Sprint 2

PLAS-8 As a user, I can register as a donor and donate plasma 10:00

PLAS-9 As a user, I can request plasma 10:00

PLAS-3 Sprint 3

PLAS-11 As a donor , I can accept or reject the request 10:00

PLAS-12 As a donor, I will get notification via email upon a request 10:00

PLAS-13 As a recipient, I will get notification via email when a donor accepts my request 10:00

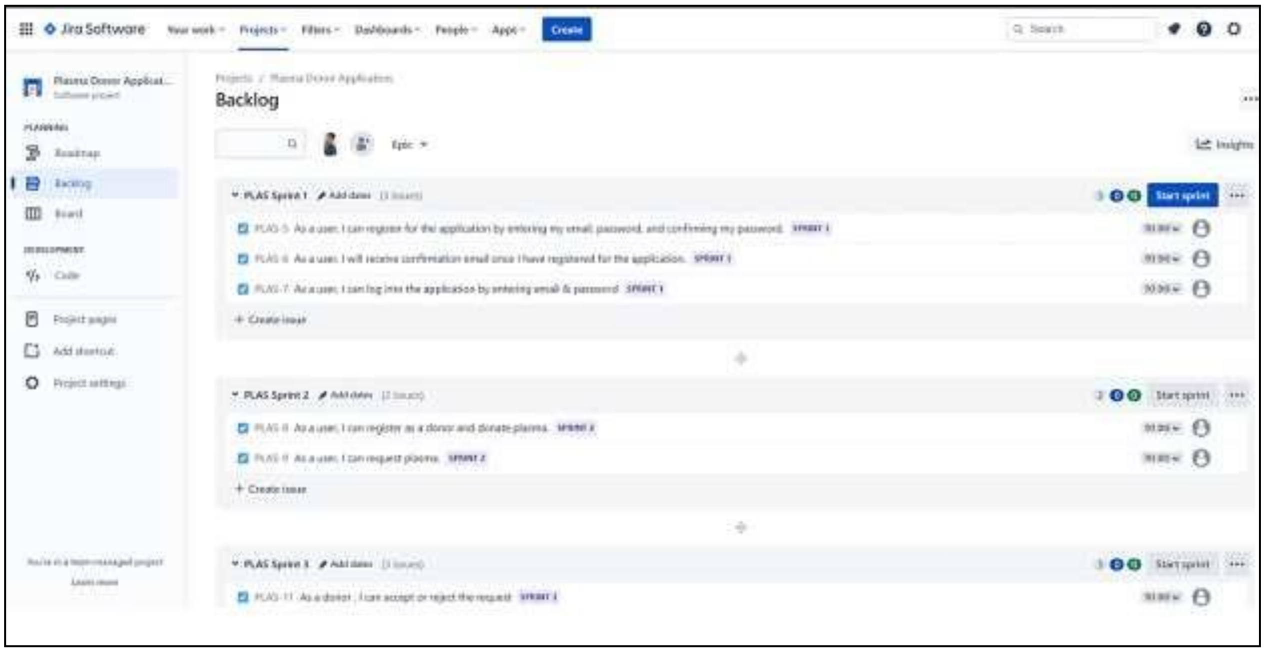
PLAS-4 Sprint 4

PLAS-14 As a user I can view the stats page which shows the count of donors, plasma available etc. 10:00

PLAS-15 As a user, I can get answers to my queries using the chatbot. 10:00

Fig 3.1. Sprint Delivery Schedule

6.3 REPORTS FROM JIRA



PLAS Sprint 1 Add dates (3 issues) 3 0 0 Start sprint

PLAS-5 As a user, I can register for the application by entering my email, password, and confirming my password. SPRINT 1 10:00

PLAS-6 As a user, I will receive confirmation email once I have registered for the application. SPRINT 1 10:00

PLAS-7 As a user, I can log into the application by entering email & password SPRINT 1 10:00

+ Create issue

PLAS Sprint 2 Add dates (2 issues) 2 0 0 Start sprint

PLAS-8 As a user, I can register as a donor and donate plasma. SPRINT 2 10:00

PLAS-9 As a user, I can request plasma. SPRINT 2 10:00

+ Create issue

PLAS Sprint 3 Add dates (3 issues) 3 0 0 Start sprint

PLAS-11 As a donor , I can accept or reject the request SPRINT 3 10:00

Fig 3.2. Sprint report

CODING & SOLUTIONING

7.1 FEATURE 1 – DONOR REGISTRATION

PYTHON SNIPPET :

```
@app.route('/add_donor', methods=['POST', 'GET'])
def add_donor():
    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=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0i', '', '')
            sql = "insert into donors values(?,?,?,?,?)"
            param = name, email,blood_group,contact_no, location,
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.execute(stmt, param)
            msg = "You're successfully registered as donor"
        except Exception as e:
            print("exception occured!",e)
            msg = e
        finally:
            return render_template('donor_registration_status.html', msg = msg)
```

7.2 FEATURE 2 – REQUEST

PYTHON SNIPPET :

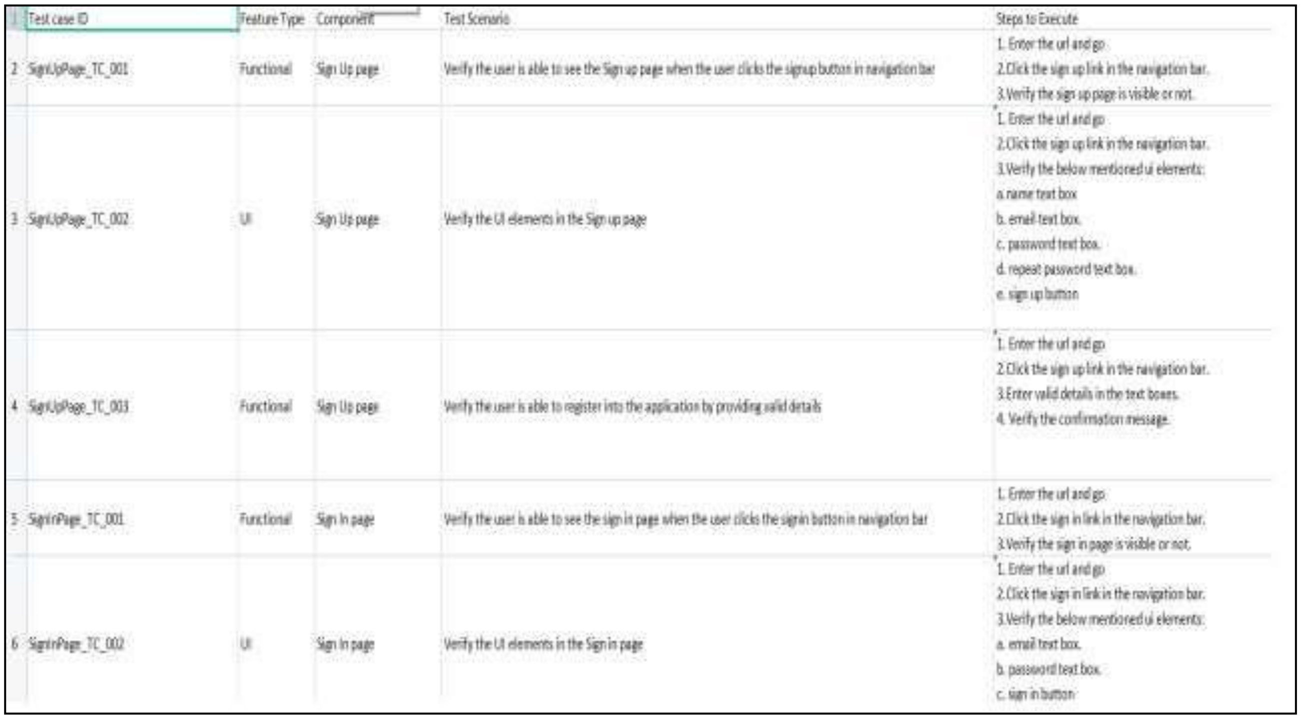
```
@app.route(„/create_request‟, methods=[„POST‟, „GET‟])
def create_request():
    if request.method == „POST‟:
        try:
            name = request.form[„name‟]
            email  =  request.form[„email‟]
            blood_group = request.form[„blood_group‟]
            contact_no = request.form[„contact_no‟]
            location = request.form[„city‟]
             conn      =      ibm_db.connect(„DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0i‟, „‟, „‟)
            sql = "insert into requests (name, email, blood_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 15ccurred!",e)
            msg = e
        finally:
            return render_template(„donor_registration_status.html‟, msg = msg)
```

TESTING

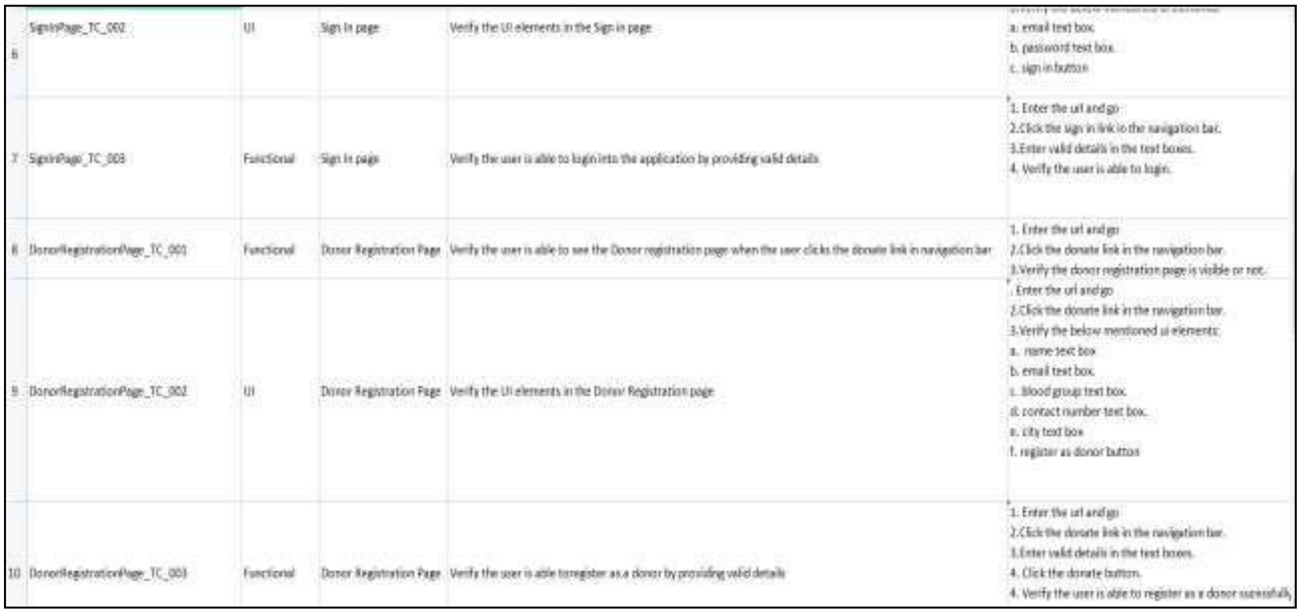
8.1 TEST CASES

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
8	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
15	Verify the donor gets a email notification when they make a request	Positive
16	Verify the donor and recipient gets a email notification when the donor accepts the 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

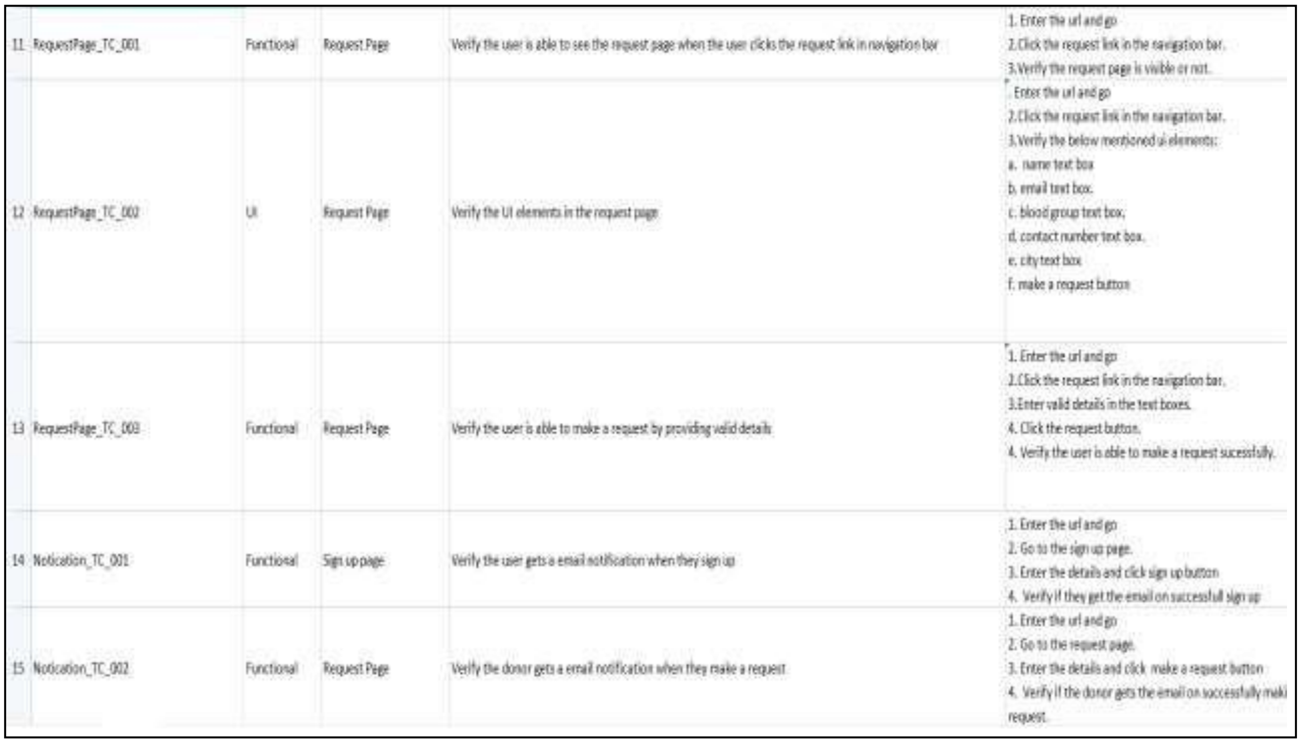
8.2 USER ACCEPTANCE TESTING



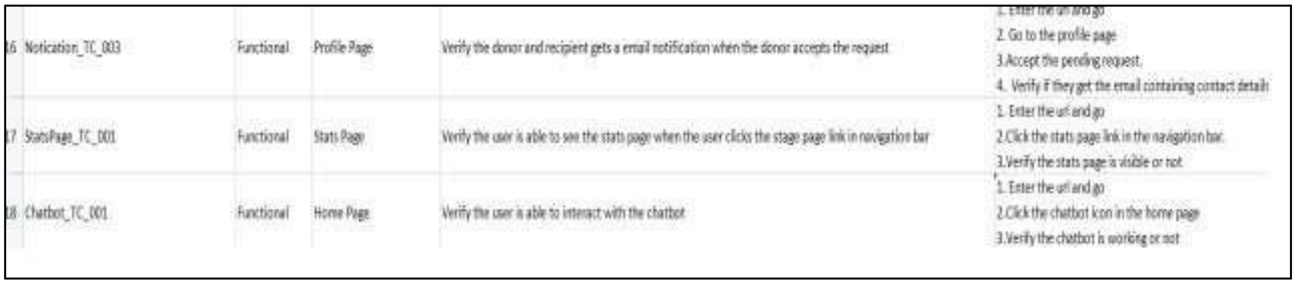
1	Test case ID	Feature Type	Component	Test Scenario	Steps to Execute
2	SignUpPage_TC_001	Functional	Sign Up page	Verify the user is able to see the Sign up page when the user clicks the signup button in navigation bar	1. Enter the url and go 2.Click the sign up link in the navigation bar. 3.Verify the sign up page is visible or not.
3	SignUpPage_TC_002	UI	Sign Up page	Verify the UI elements in the Sign up page	1. Enter the url and go 2.Click the sign up link in the navigation bar. 3.Verify the below mentioned ui elements: a.name text box b. email text box. c. password text box. d. repeat password text box. e. sign up button
4	SignUpPage_TC_003	Functional	Sign Up page	Verify the user is able to register into the application by providing valid details	1. Enter the url and go 2.Click the sign up link in the navigation bar. 3.Enter valid details in the text boxes. 4. Verify the confirmation message.
5	SigninPage_TC_001	Functional	Sign In page	Verify the user is able to see the sign in page when the user clicks the signin button in navigation bar	1. Enter the url and go 2.Click the sign in link in the navigation bar. 3.Verify the sign in page is visible or not.
6	SigninPage_TC_002	UI	Sign In page	Verify the UI elements in the Sign in page	1. Enter the url and go 2.Click the sign in link in the navigation bar. 3.Verify the below mentioned ui elements: a. email text box. b. password text box. c. sign in button



6	SignInPage_TC_002	UI	Sign In page	Verify the UI elements in the Sign in page	[illegible] 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



11	RequestPage_TC_001	Functional	Request Page	Verify the user is able to see the request page when the user clicks the request link in navigation bar	1. Enter the url and go 2.Click the request link in the navigation bar. 3.Verify the request page is visible or not.
12	RequestPage_TC_002	UI	Request Page	Verify the UI elements in the request page	. Enter the url and go 2.Click the request 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. make a request button
13	RequestPage_TC_003	Functional	Request Page	Verify the user is able to make a request by providing valid details	1. Enter the url and go 2.Click the request link in the navigation bar. 3.Enter valid details in the text boxes. 4. Click the request button. 4. Verify the user is able to make a request sucessfully.
14	Notication_TC_001	Functional	Sign up page	Verify the user gets a email notification when they sign up	1. Enter the url and go 2. Go to the sign up page. 3. Enter the details and click sign up button 4. Verify if they get the email on successfull sign up
15	Notication_TC_002	Functional	Request Page	Verify the donor gets a email notification when they make a request	1. Enter the url and go 2. Go to the request page. 3. Enter the details and click make a request button 4. Verify if the donor gets the email on successfully maki request.



16	Notication_TC_003	Functional	Profile Page	Verify the donor and recipient gets a email notification when the donor accepts the request	1. Enter the url and go 2. Go to the profile page 3.Accept the pending request. 4. Verify if they get the email containing contact details
17	StatsPage_TC_001	Functional	Stats Page	Verify the user is able to see the stats page when the user clicks the stage page link in navigation bar	1. Enter the url and go 2.Click the stats page link in the navigation bar. 3.Verify the stats page is visible or not
18	Chatbot_TC_001	Functional	Home Page	Verify the user is able to interact with the chatbot	1. Enter the url and go 2.Click the chatbot icon in the home page 3.Verify the chatbot is working or not

RESULTS

9.1 PERFORMANCE METRICS

Web application performance metrics help determine certain aspects that impact the performance of an application. There are eight key metrics, including: User Satisfaction—also known as Apdex Scores, uses a mathematical formula in order to determine user satisfaction.

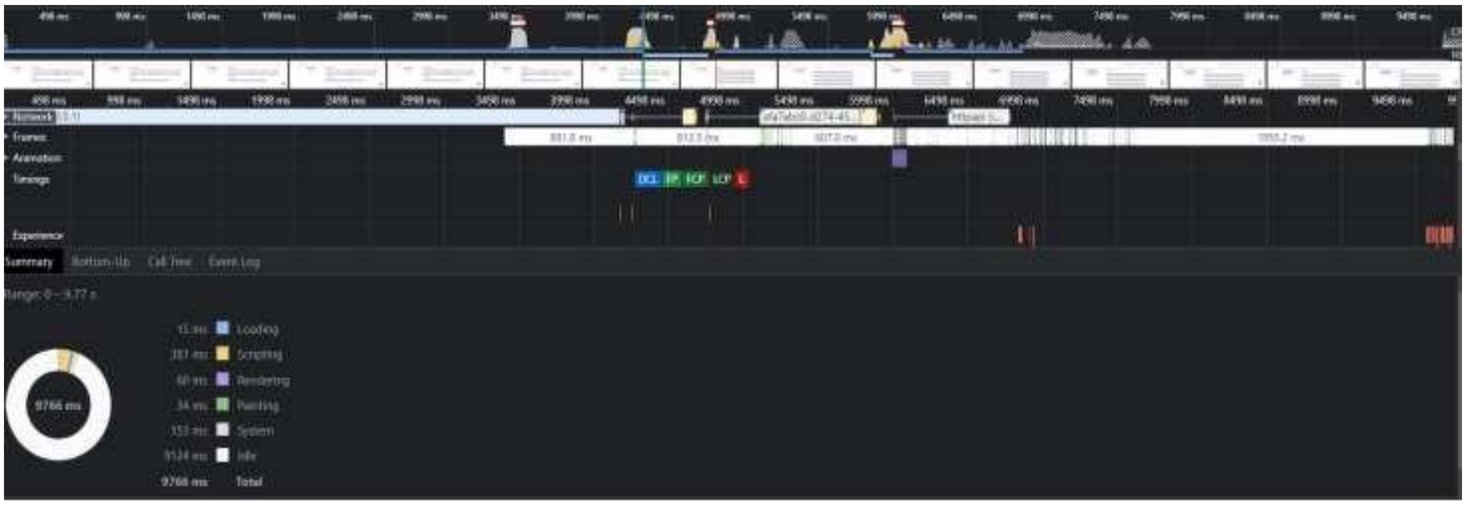


Fig 4.1. Performance Metrics

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 Plasma donation and Plasma Requesting.
- Attracts more, number of users as it is available in the form of Mobile application instead 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.

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 increasethe 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 areeligible 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 developedthe Android mobile application to ease the process of becoming a donor and recipient ofPMB bank.

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.

APPENDIX

SOURCE CODE

request.html :

```
<!doctype html>
<html lang=”en”>
  <head>
    <!—Required meta tags →
    <meta charset=”utf-8”>
    <meta name=”viewport” content=”width=device-width, initial-scale=1, shrink-to-fit=no”>
    <link                 href=”https://fonts.googleapis.com/css?family=Roboto:400,700,900&display=swap”
rel=”stylesheet”>
<!—Vendor CSS Files →
<link href=”../static/vendor/aos/aos.css” rel=”stylesheet”>
<link href=”../static/vendor/boxicons/css/boxicons.min.css” rel=”stylesheet”>
<link href=”../static/vendor/glightbox/css/glightbox.min.css” rel=”stylesheet”>
<link href=”../static/vendor/swiper/swiper-bundle.min.css” rel=”stylesheet”>
<link href=”../static/css/style.css” rel=”stylesheet”>
     <!—Bootstrap CSS →
   <link href=”https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css” rel=”stylesheet”
integrity=”sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC”
crossorigin=”anonymous”>
   <script                    src=”https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js”
integrity=”sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM”
crossorigin=”anonymous”></script>
    <!—Style →
```

```
<link rel=”stylesheet” href=”../static/css/request.css”>
<script>
  function validateForm() {
    let name = document.forms[“form”][“name”].value;
    let email = document.forms[“form”][“email”].value;
    let blood_group = document.forms[“form”][“blood_group”].value;
    let contact_no = document.forms[“form”][“contact_no”].value;
    let city = document.forms[“form”][“city”].value;
   if(validname(name)     &&     validemail(email)     &&     validblood_group(blood_group)     &&
validcontact_no(contact_no) &&  validcity(city)){
    return true;
   }
   else{
     return false;
   }
   }
   function validname(name){
     document.getElementById(“name_err”).innerHTML=””
     if (name != “”) {
       var letters = /^[a-zA-Z ]*$/;
         if(name.match(letters))
         {
        return true;
         }
         else
         {
        document.getElementById(“name_err”).innerHTML=”Name should contain only Alphabets”
         return false;
         }
     }
     else{
       document.getElementById(“name_err”).innerHTML=”Name should not be empty”
         return false;
     }
   }
   function validemail(email){
    console.log(email)
     document.getElementById(“email_err”).innerHTML=””;
     if (email != “”) {
       var emailfor = /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/;
         if(email.match(emailfor))
         {
         return true;
         }
         else
         {
         document.getElementById(“email_err”).innerHTML=”Invalid Email”
         return false;
         }
     }
     else{
       document.getElementById(“email_err”).innerHTML=”Email Should not be empty”
         return false;
     }
   }
   function validblood_group(blood_group){
     document.getElementById(“blood_err”).innerHTML=””;
     if (blood_group == “”) {
    document.getElementById(“blood_err”).innerHTML=”Blood group Should not be empty”
         return false;
     }
     else{
       return true;
     }
   }
   function validcontact_no(contact_no){
 document.getElementById(“contact_err”).innerHTML=””;
  if (contact_no == “”) {
   document.getElementById(“contact_err”).innerHTML=”Contact number Should not be empty”
      return false;
  }
  else{
    return true;
  }
}
function validcity(city){
 document.getElementById(“city_err”).innerHTML=””;
 if (city == “”) {
   document.getElementById(“city_err”).innerHTML=”City should not be empty”
      return false;
  }
  else{
    return true;
  }
}
      </script>
  <title>Planor</title>
</head>
<body>
<!-- ======= Header ======= →
<header id=”header” class=”fixed-top d-flex align-items-center shadow”>
  <div class=”container d-flex align-items-center justify-content-between”>
    <div class=”logo”>
      <h1 class=”text-light”><a href=”/”><span>Planor</span></a></h1>
    </div>
    <nav id=”navbar” class=”navbar”>
      <ul>
        <li><a class=”nav-link scrollto” href=”/”>Home</a></li>
        <li><a class=”nav-link scrollto” href=”#services”>Services</a></li>
```

```
          <li><a class=”nav-link scrollto” href=”/request”>Request</a></li>
          <li><a class=”nav-link scrollto” href=”/donor_registration”>Donate</a></li>
          <li><a class=”nav-link scrollto” href=”/sign_in”>Sign In</a></li>
          <li><a class=”nav-link scrollto” href=”/sign_up”>Sign Up</a></li>
          <li><a class=”nav-link scrollto” href=”/profile”>My Profile</a></li>
          <li><a class=”nav-link scrollto” href=”/logout”>Logout</a></li>
        </ul>
        <i class=”bi bi-list mobile-nav-toggle”></i>
      </nav><!-- .navbar →
    </div>
  </header><!—End Header →
    {% if logged_in == True %}
  <div class=”content”>
    <div class=”container”>
      <div class=”row justify-content-center”>
        <div class=”col-md-10”>
          <div class=”row justify-content-center”>
            <div class=”col-md-6”>
              <h3 class=”heading mb-4”>Let‟s make a request!</h3>
             <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptas debitis, fugit natus?</p>
              <p><img src=”../static/img/request.jpeg” alt=”Image” class=”img-fluid”></p>
            </div>
            <div class=”col-md-6”>
              <form action=”{{ url_for(„create_request‟) }}” onsubmit=”return validateForm()”
method=”post” id=”contactForm” name=”form”>
                <div class=”row”>
                  <div class=”col-md-12 form-group “>
                    <input onkeyup=”validname(this.value)” type=”text” class=”form-control mb-3”
name=”name” id=”name” placeholder=”Your name” value={{name}}>
                  </div>
                  <div style=”margin-top: -15px;”>
             <label style=”position: static; color:red” id=”name_err” class=”text-danger” ></label>
                  </div>
                </div>
                <div class="row">
                  <div class="col-md-12 form-group">
                    <input onkeyup="validemail(this.value)" type="text" class="form-control mb-3"
name="email" id="email" placeholder="Email" value={{email}}>
                  </div>
                  <div style="margin-top: -15px;">
              <label style="position: static; color:red" id="email_err" class="text-danger" ></label>
                  </div>
                </div>
                <div class="row">
                  <div class="col-md-12 form-group">
                    <input onkeyup="validblood_group(this.value)" type="text" class="form-control mb-3"
name="blood_group" id="subject" placeholder="Blood group">
                  </div>
                  <div style="margin-top: -15px;">
                    <label style="position: static; color:red" id="blood_err" class="text-danger" ></label>
                  </div>
                </div>
                <div class="row">
                  <div class="col-md-12 form-group">
                    <input onkeyup="validcontact_no(this.value)" type="text" class="form-control mb-3"
name="contact_no" id="subject" placeholder="Contact number">
                  </div>
                  <div style="margin-top: -15px;">
                    <label style="position: static; color:red" id="contact_err" class="text-danger" ></label>
                  </div>
                </div>
                <div class="row">
                  <div class="col-md-12 form-group">
                    <input onkeyup="validcity(this.value)" type="text" class="form-control mb-3"
name="city" id="subject" placeholder="City">
                  </div>
                  <div style=”margin-top: -15px;”>
                    <label style=”position: static; color:red” id=”city_err” class=”text-danger” ></label>
                  </div>
                </div>
                  <div class=”row”>
                  <div class=”col-12”>
           <input type=”submit” value=”Register as Donor” class=”btn btn-primary rounded-0 py-2 px-4”>
                  <span class=”submitting”></span>
                  </div>
                </div>
              </form>
              <div id=”form-message-warning mt-4”></div>
              <div id=”form-message-success”>
                Your message was sent, thank you!
              </div></div> </div></div></div></div></div>
    {% else %}
    <div>
      <div class=”d-flex flex-column align-items-center justify-content-center “ style=”height: 400px;”>
         Please, Sign in to make a request
         <a  class=”btn btn-dark mt-2” style=”height: initial;” href=”/sign_in”>Sign in</a>
      </div>
    </div>
    {% endif %}
    <script                    src=”https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js”
integrity=”sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM”
crossorigin=”anonymous”></script>
    </body>
</html>
```

Script.py :

```
import ibm_db
from flask import *
app = Flask(_name_)
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/sign_up')
def signUp():
    return render_template('sign_up.html')
@app.route('/sign_in')
def signIn():
    return render_template('sign_in.html')
@app.route('/request')
def requests():
    email = request.cookies.get('email')
    name = request.cookies.get('name')
    if email != None:
        resp = make_response(render_template('request.html',email = email, name = name, logged_in =
True))
    else:
        resp = make_response(render_template('request.html',email = email, name = name, logged_in =
False))
    return resp


@app.route('/donor_registration')
def donor_registration():
    email = request.cookies.get('email')
    name = request.cookies.get('name')
    if email != None:
        resp = make_response(render_template('donor_registration.html',email = email, name = name,
logged_in = True))
    else:
        resp = make_response(render_template('donor_registration.html',email = email, name = name,
logged_in = False))
    return resp
@app.route('/profile')
def profile():
    email = request.cookies.get('email')
    name = request.cookies.get('name')
    if email != None:
        conn = ibm_db.connect( 'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0i', '', '')
        sql = 'select * from requests where email='+'\''+email+'\''
        stmt = ibm_db.exec_immediate(conn, sql)
        requests = []
        dictionary = ibm_db.fetch_assoc(stmt)
        while dictionary != False:
            print(dictionary["ID"])
            requests.append(dictionary)
            dictionary = ibm_db.fetch_assoc(stmt)
        print(requests)
        sql = 'select * from donors where email='+'\''+email+'\''
        stmt = ibm_db.exec_immediate(conn, sql)
        dictionary = ibm_db.fetch_assoc(stmt)
        isDonor = False
        pending_requests = []
        if dictionary != False:
            isDonor = True
            donor_location = dictionary["LOCATION"]
            donor_bloodgroup = dictionary["BLOOD_GROUP"]
            sql = "select * from requests where blood_group="+"'"+donor_bloodgroup+"'"+"and
location= "+"'"+donor_location+"'"+"and request_status= "+"'Pending'"
            stmt = ibm_db.exec_immediate(conn, sql)
            dictionary = ibm_db.fetch_assoc(stmt)
            while dictionary != False:
                pending_requests.append(dictionary)
                dictionary = ibm_db.fetch_assoc(stmt)
            print(pending_requests)
        accepted_requests= []
        if isDonor:
            sql = 'select * from requests where accepted_by='+'\''+email+'\''
            stmt = ibm_db.exec_immediate(conn, sql)
            dictionary = ibm_db.fetch_assoc(stmt)
            while dictionary != False:
                accepted_requests.append(dictionary)
                dictionary = ibm_db.fetch_assoc(stmt)
            print(accepted_requests)
        return render_template('profile.html', name =name, email = email,requests_len = len(requests)
,requests = requests, pending_requestslen = len(pending_requests), pending_requests =
pending_requests, accepted_requestslen = len(accepted_requests), accepted_requests =
accepted_requests, logged_in=True)
    else:
        return render_template('profile.html', logged_in= False)
@app.route('/add_user', methods=['POST', 'GET'])
def add_user():
   if request.method == 'POST':
      try:

        name = request.form['name']
        email = request.form['email']
        password = request.form['pass']
        conn = ibm_db.connect(
           'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0i', '', '')
        sql = "insert into users values(?,?,?)"
        param = name, email, password,
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.execute(stmt, param)
            msg = "You're successfully signed up!"
        except Exception as e:
            print("exception occured!",e)
            msg = e
        finally:
            return render_template('post_signup.html', msg = msg)
@app.route('/validate_user',methods = ['POST', 'GET'])
def validate_user():
   if request.method == 'GET':
      try:
            args = request.args
            email = args.get('email')
            password = args.get('password')
               conn = ibm_db.connect(
                'DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0i', '', '')
            sql = 'select * from users where email='+'\''+email+'\''
            stmt = ibm_db.exec_immediate(conn, sql)
            dictionary = ibm_db.fetch_assoc(stmt)
            print("executed")
            print(dictionary)
            if dictionary != False:
                        if(dictionary["PASSWORD"]== password):
                               print("success")
                               resp = make_response(render_template("post_signin.html"))
                               resp.set_cookie('email', dictionary["EMAIL"])
                               resp.set_cookie('name',dictionary["NAME"])
                               print("success")
                               return resp
                        else:
```

```
                  return "Incorrect Password"
            else:
                return "User does not exists"
        except Exception as e :
          print("error",e)
          return repr(e)
@app.route('/add_donor', methods=['POST', 'GET'])
def add_donor():
    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=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0i', '', '')
            sql = "insert into donors values(?,?,?,?,?)"
            param = name, email,blood_group,contact_no, location,
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.execute(stmt, param)
            msg = "You're successfully registered as donor"
        except Exception as e:
            print("exception occured!",e)
            msg = e
        finally:
            return render_template('donor_registration_status.html', msg = msg)
@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=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0i', '', '')
            sql = "insert into requests (name, email, blood_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 occured!",e)
            msg = e
        finally:
            return render_template('donor_registration_status.html', msg = msg)
@app.route('/accept_request', methods=['POST', 'GET'])
def accept_request():
    if request.method == 'POST':
        try:

            id = request.form['id']
            email = request.cookies.get('email')
            conn = ibm_db.connect('DATABASE=bludb;HOSTNAME=b1bc1829-6f45-4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32304;SECURITY=SSL;SSL
ServerCertificate=DigiCertGlobalRootCA.crt;UID=gfn00031;PWD=LITZUQj2tpFc3t0i', '', '')
           sql = "update requests set request_status = 'Accepted' , accepted_by ="+"'"+email+"'"+"where
id ="+"'"+id+"'"
            stmt = ibm_db.exec_immediate(conn, sql)
        except Exception as e:
            print("exception occured!",e)
        finally:
            return redirect(url_for('profile'))
@app.route('/logout')
def logout():
    email = request.cookies.get('email')
    if email != None:
        resp = make_response(render_template('logout.html',loggedin = True))
        resp.set_cookie('name', '', expires=0)
        resp.set_cookie('email', '', expires=0)
        resp.set_cookie('dob', '', expires=0)
    else:
        resp = make_response(render_template('logout.html',loggedin = False))
    return resp
if __name__ == '_main_':
    app.run(debug=True)
```

GITHUB & PROJECT DEMO LINK

Github Link :

https://github.com/IBM-EPBL/IBM-Project-8620-1658926060