PLASMA DONAR APPLICATION

PROJECT REPORT

Submitted by

M. SABARINATH

S. SHREEKALA

M. VIGNESHKUMAR

S. SANJAY

In partial fulfilment for the award of the degree

Of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

VELALAR COLLEGE OF ENGINEERING AND TECHNOLOGY,

ERODE.

PROJECT REPORT FORMAT

7. CODING & SOLUTIONING

(Explain the features added in the project along with code) 7.1 Feature code

7.2 Database Schema

8..RESULTS

8.1 Performance Metrics

9. ADVANTAGES & DISADVANTAGES

10. CONCLUSION

11. FUTURE SCOPE 12.APPENDIX

Source code

Git Hub & Project Demo Link

1.INTRODUCTION

The world is suffering from the COVID 19 crisis and no vaccine has been found yet.. But there is another scientific way in which we can help reduce mortality or help people affected by COVID19 by donating plasma from recovered patients. In the absence of an approved antiviral treatment plan for a fatal COVID19 infection, plasma therapy is an experimental approach to treat COVID19-positive patients and help them faster recovery. Therapy is considered competent. In the recommendation system, the donor who wants to donate plasma can donate by uploading their COVID19 certificate and the blood bank can see the donors who have uploaded the certificate and they can make a request to the donor and the hospital can register/login and search for the necessary things. plasma from a blood bank and they can request a blood bank and obtain plasma from the blood bank.

1.1PROJECT OVERVIEW:

The main goal of our project is to make it easier for the COVID-19 patients to get a plasma donor easily as well as donate plasma if they have recovered. The system targets two types of users: the people who want to donate plasma and the people who need plasma. The main objective of developing the application is to make it easier for the COVID-19 patients to get a plasma donor easily and as soon as possible.

1.2 PURPOSE:

Plasma Donor Application deals with notifying concerned donor upon request by the recipient in need of plasma . This project provides quick access to donors for an immediate requirement of blood. In case of emergency/surgery. Blood procurement is always a major problem which consumes lot of time .

2.LITERATURESURVEY

AUTHOR :1.KalpanaDeviguntoju2.Tejaswini Jalli 3.Sreeja Uppala 4. Sanjay Mallisettai

*This paper present the design to design a user-friendly web application that is like a scientific vehicle from which we can help reduce mortality or help those affected by covid19 by donating plasma from patients who have recovered without approved antiretroviral therapy planning for a deadly covid19 infection, plasma therapy is an experimental approach to treat those covid-positive patients and help them recover faster.

*The main purpose of the proposed system, the donor who wants to donate plasma can simply upload their covid19 traced certificate and can donate the plasma to the blood bank, the blood bank can apply for the donor and once the donor has accepted the request, the blood bank can add the units they need and the hospital can also send the request to the blood bank that urgently needs the plasma for the patient and can take the plasma from the blood bank.

AUTHOR:1. Jerome Ah Lindeboom 2.Keshen R Mathura 3. Irene HaAartman 4. Frans Hm Kroon

*This paper present the study to describe and quantify the therapeutic value of platelet concentrate on the capillary density in oral mucosal wound healing.

*Ten patients, five males and five females, were included in the study with a mean age of 54.2±9.1 years for females and 57.6±6.9 years for males. donor platelet counts from whole blood had a mean value of 248.5±13.5 × 109/l, while the value of platelet counts in the prp had a mean of 975.9±97.9 × 109/l. wound healing was significantly accelerated in the prp-treated mucosal wounds during the first 10 postoperative days. after the second week, no obvious differences between the prp or placebo side could be noted.

AUTHOR :1.Samarth Gupta 2. Rakesh Kumar Jain

*This paper present that platelet-rich plasma (prp) is widely used for wound healing in medical care because of the numerous growth factors it contains. traditionally, donor sites are left to heal with a primary dressing so wounds are not left open. however, a delay in healing accompanied by pain at a donor site is often seen. this study primarily throws light on the use of autologous prp over split-thickness skin graft (stsg) donor sites to promote healing and reduce pain.

*A Total of 100 patients were included in the study. patients in the prp group showed statistically significant faster healing at postoperative day 14 compared with the control group ($p<0.05$), where required dressings for 3-4 weeks postoperatively. pain scale scores in the postoperative period were significantly less in the prp group at six hours postoperatively compared with the control group ($p<0.05$). there was a reduced incidence of hypertrophic scar formation in the small number of patients in the prp group who had developed hypertrophic scar previously.

AUTHOR :1.J. Alsousou2.M.Thompson 3. P. Hulley. A. Noble K. Willett

*Although mechanical stabilisation has been a hallmark of orthopaedicsurgicalmanagement, orthobiologics are now playing an increasing role. platelet-rich plasma (prp)is a volume of plasma fraction of autologous blood having platelet concentrations above Baseline.

AUTHOR :1.Nayan das 2. MD. Asif Iqbal

*This paper present the necessity of blood has become a significant concern in the present context all over the world. due to a shortage of blood, people couldn't save themselves or their friends and family members. a bag of blood can save a precious life. statistics show that a tremendous amount of blood is needed yearly because of major operations, road accidents, blood disorders, including anemia, hemophilia, and acute viral infections like dengue, etc.

* Approximately 85 million people require single or multiple blood transfusions for treatment. voluntary blood donors per 1,000 population of some countries are quite promising, such as Switzerland (113/1,000), japan (70/1,000), while others have an unsatisfying result like India has 4/1,000, and Bangladesh has 5/1000.

2.1 EXISTING PROBLEM:

The existing problem was finding the perfect fundraising platform ,finding donors, lack of easy access, donor relationship- donor retention ,lack of resources **.**

2.2 REFERENCE:

[1]INSTANT PLASMA DONOR RECIPIENTCONNECTOR WEBAPPLICATION Kalpana
Devi Guntoju*1, Tejaswini Jalli*2, Sreeja Uppala*3, Sanjay Mallisetti*4
*1,2,3,4Dept. Of CSE, CMR Technical Campus, India.

[2] Influence of the application of platelet-enriched plasma in oral mucosal wound healing Jérôme A H Lindeboom [1], Keshen R Mathura, Irene H A Aartman, Frans H M Kroon, Dan M J Milstein, Can Ince

[3] Application of autologous platelet-rich plasma to graft donor sites to reduce pain and promote healing Samarth Gupta [1], Rakesh Kumar Jain [1]

[4]The biology of platelet-rich plasma and its application in trauma and orthopaedic surgery: a review of the literature J Alsousou [1], M Thompson, P Hulley, A Noble, K Willett

[5] Nearest Blood & Plasma Donor Finding: A Machine Learning Approach Nayan Das; MD. Asif Iqbal

2.3 PROBLEM STATEMENT DEFINITION :

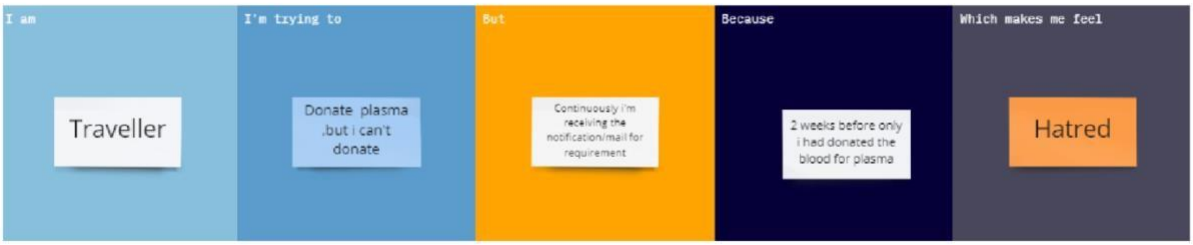
During COVID 19 crisis the requirement for plasma increased drastically as there were no vaccinations found in order to treat the infected patients. In such situation it was very difficult to find the plasma donor, check whether the donor was infected previously and was recovered, and which donor is eligible to donate plasma was a challenging task. As the plasma therapy was one of the ways to treat the infected patients getting the donor details played a major role.

PROBLEM STATEMENT- 1

PROBLEM STATEMENT-2

PROBLEM STATEMENT-3

PROBLEM STATEMENT-4



PROBLEM STATEMENT-5

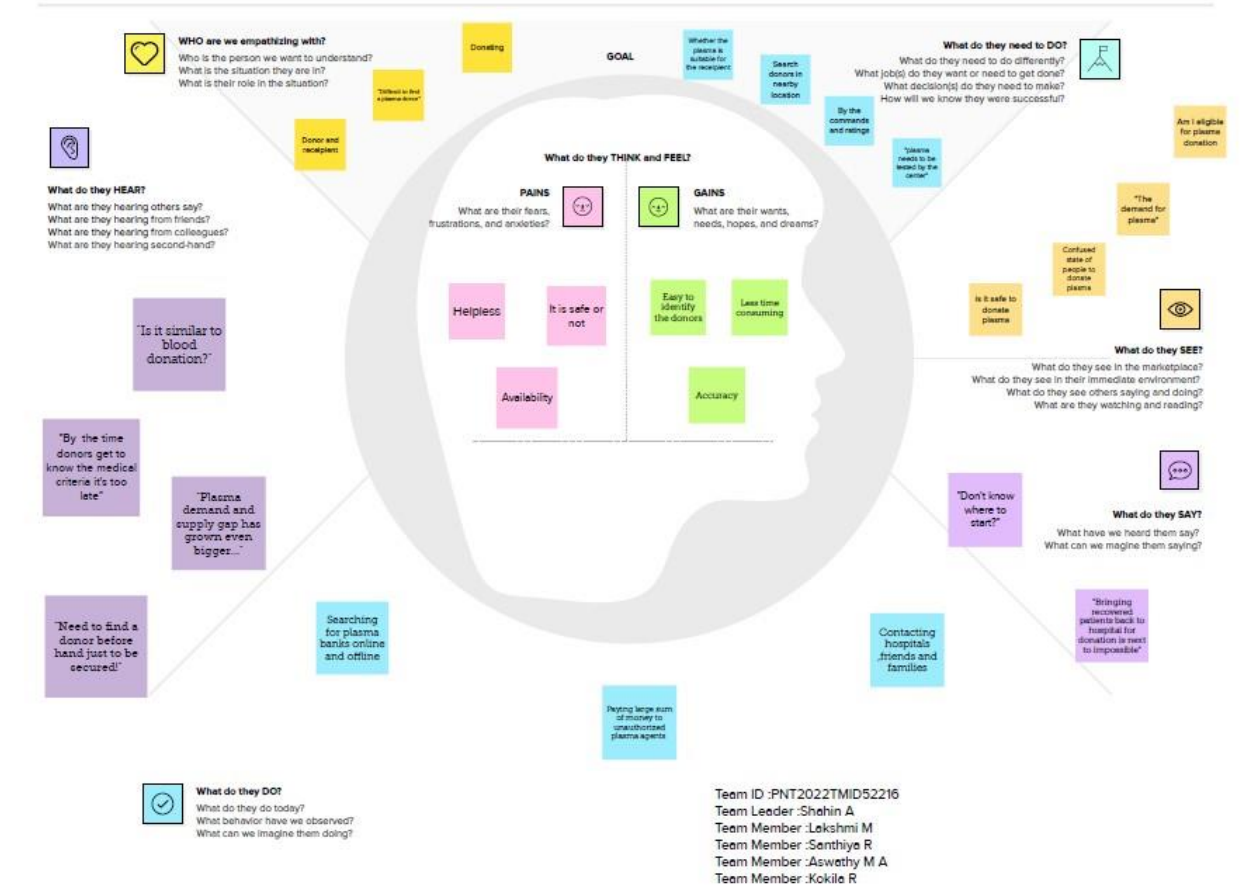
3.IDEATION &PROPOSED SOLUTION

Once all the data and insights were gathered from the research phase, the next step was to generate potential solution.The proposed method helps the users to check the availability of donors. A donor has to register to the application providing their details. The registered users can get the information about the donor count of each

blood group. The database will have all the details such as name, email, phone number, infected status. Whenever a user requests for a particular blood group then the concerned blood group donors will receive the notification regarding the requirement.

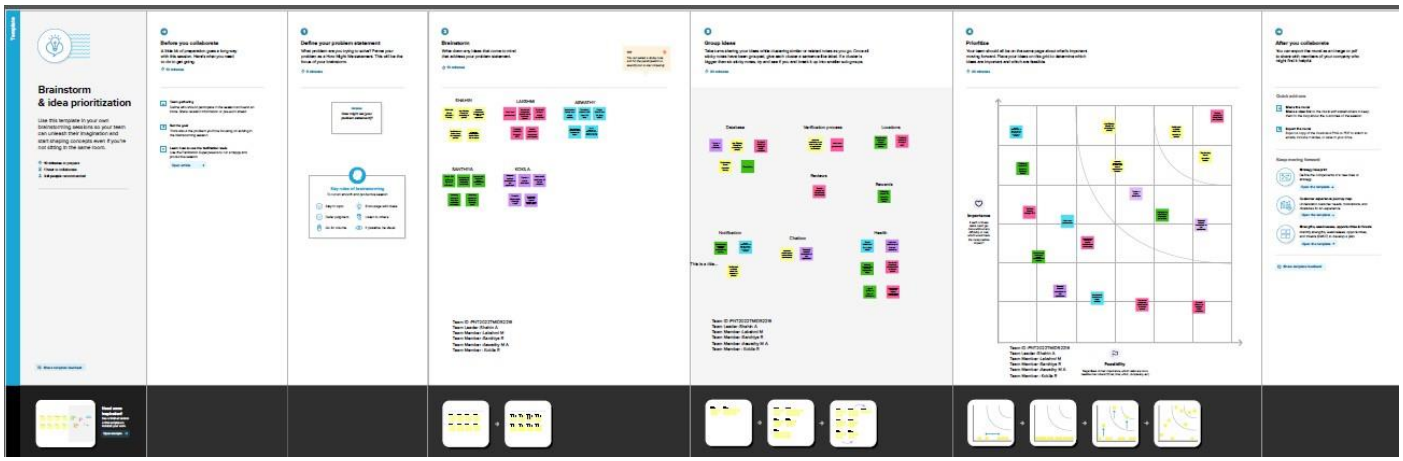
3.1 EMPATHY MAP CANVAS :

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behavior's 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.



3.2 IDEATION &BRAINSTORM :

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving .Prioritizing volume over value, out of the box ideas are welcome and build upon , and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solution.



3.3 PROPOSED SOLUTION :

1.

Parameter & Description:

1.Problem Statement (problem to be solved):

I'm a student I'm trying to find plasma because in the need of emergency, But I don't know whether the required unit of plasma is available or not which makes me feel disappointment.

2.Idea /Solution description:

The user should know the required unit of plasma and check the availability of plasma in the application.

3.Novelty /Uniqueness :

This problem is due to don't know how to use the application.

4.Social Impact /Customer Satisfaction :

User satisfied with the problem there will very few possibility of problem occur.

5.Business Model (Revenue Model):

On the revenue bases, this plasma donor application will be profit for Hospital ,NGO's and private sectors.

6.Scalability of the solution :

The problem of the solution were solved and also as per the user flexibility the requirement can be modified.

2.

1.Problem Statement (problem to be solved):

I'm Rural people when I'm trying to use d plasma donor application ,But I don't know how to use application & never used before but I want to use the application which makes me feel Anxiety. 2.Idea /Solution description:

The user should have basic knowledge about the application ,read the user manual or else use "chatbot" for guidance.

3.Novelty /Uniqueness:

This problem is common once they known to use there will be no issue.

4.Social impact /Customer satisfaction :

The user will be more satisfied with the solution and if once again the problem occur, it can be easily recovered.

5.Business Model (Revenue Model):

On the revenue bases, this donor application will be profit for Hospital, NGO's and private organizations. 6.Scalability of the solution:

The worst thought of the customer about the application will change and also as per the user flexibility the requirements can be modified.

3.

1.Problem Statement (Problem to be solved):

I’m the Doctor when I’m trying to use the plasma donor application because I’m expecting more specification, if more added which makes me feel enthusiastic.

2.Idea/Solution description:

Everyone will have different ideas and different queries but the most important suggestion will be added upon the application.

3. Novelty/Uniqueness :

Everyone will have different ideas and different queries but the most important suggestion will be added upon the application . 4.Social Impact/Customer Satisfaction :

User satisfied with the problem there will very few possibility of problem occur. It can be easily recovered.

5.Business Model(Revenue Model):

On the revenue bases, this donor application will be profit for Hospital, NGO’s and private organizations. 6.Scalability of the Solution:

The user mindset about the application will changed and also as per flexibility and requirements can be modified.

4.

1.Problem Statement (problem to be solved):

I’m Traveller when I’m trying to donate plasma but I can’t able to donate plasma because 2 weeks before only I had donated blood for plasma but continuously I’m receiving notification which makes me feel Hatred .

2.Ideas/Solution description:

The Traveller need to update his plasma donation details in the application, If still the issue occur contact us option in application

3.Novelty/Uniqueness :

The problem rarely occurs to user and not a common problem .it will be rectified by technical team.

4.Social impact/Customer Satisfaction :

The customer will be more satisfied with the solution and if once again problem occurs it can be easily recovered .

5.Business Model(Revenue Model):

On the revenue bases, this donor application will be profit for Hospitals, NGO’s and private organizations.

6.Scalability of the Solution :

The user mindset about the application will be changed and also as per user flexibility the requirements can be modified.

5.

1.Problem Statement(Problem to be solved):

I'm Developer I'm trying to develop an application Because most of them are uneducated there is no awareness which makes me feel frustrated.

2.Ideas /Solution description:

So started to develop an application through this a few of them are able to know it and they may able started to donate plasma.

3. Novelty/Uniqueness :

This problem is due to don't know how to use the application .

4. Social impact/Customer Satisfaction :

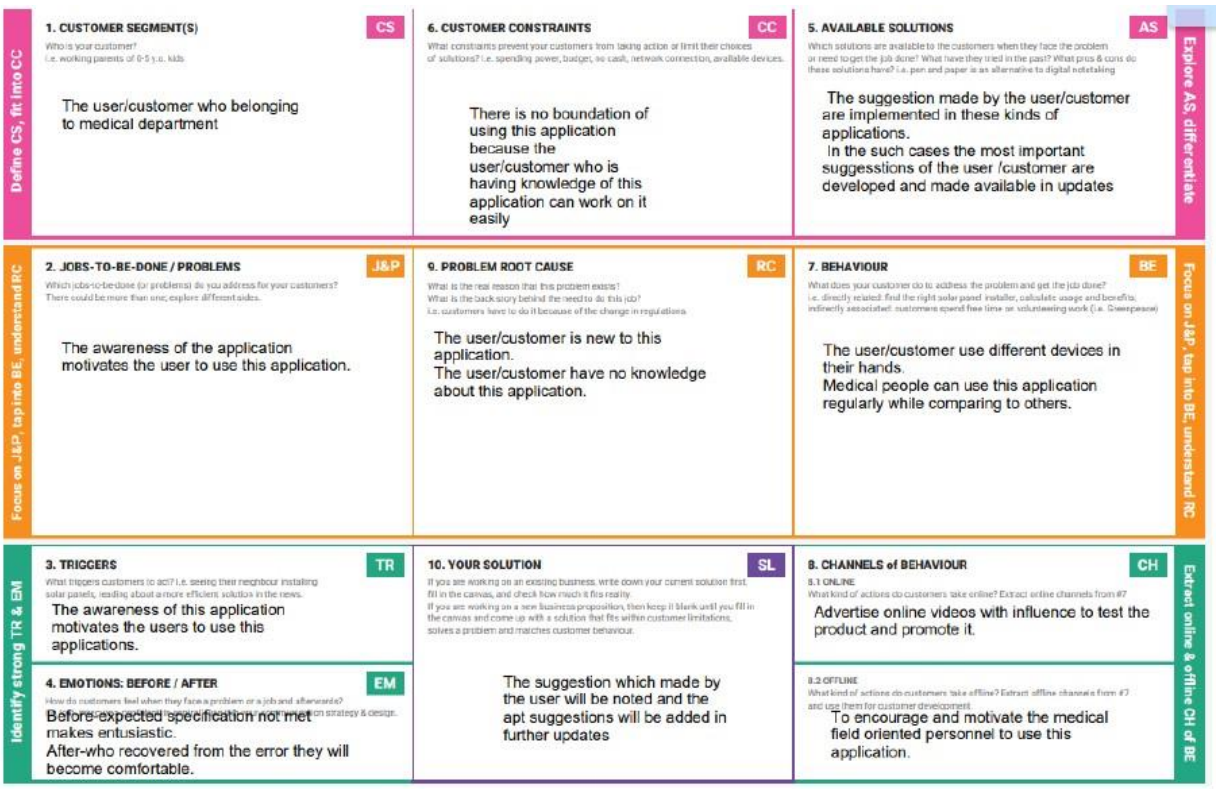
There is only very less chance of problem.

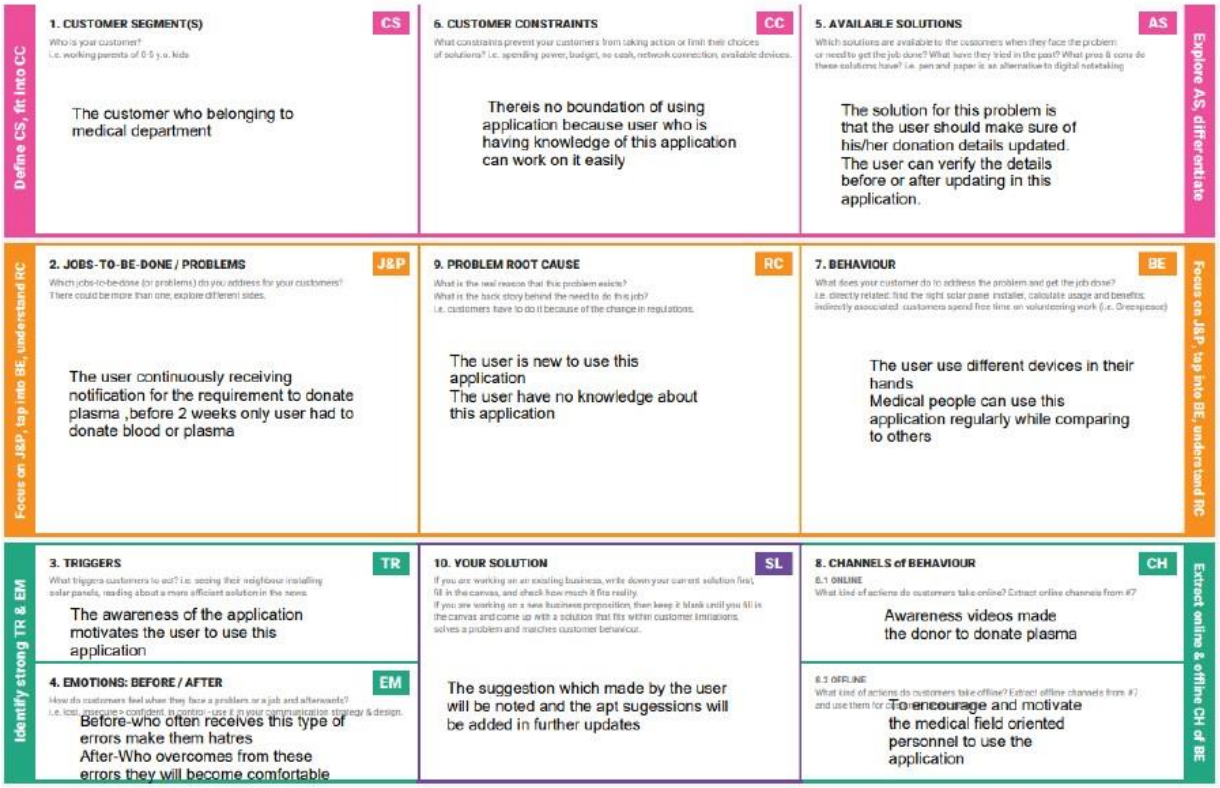
5.Business Model(Revenue Model) :

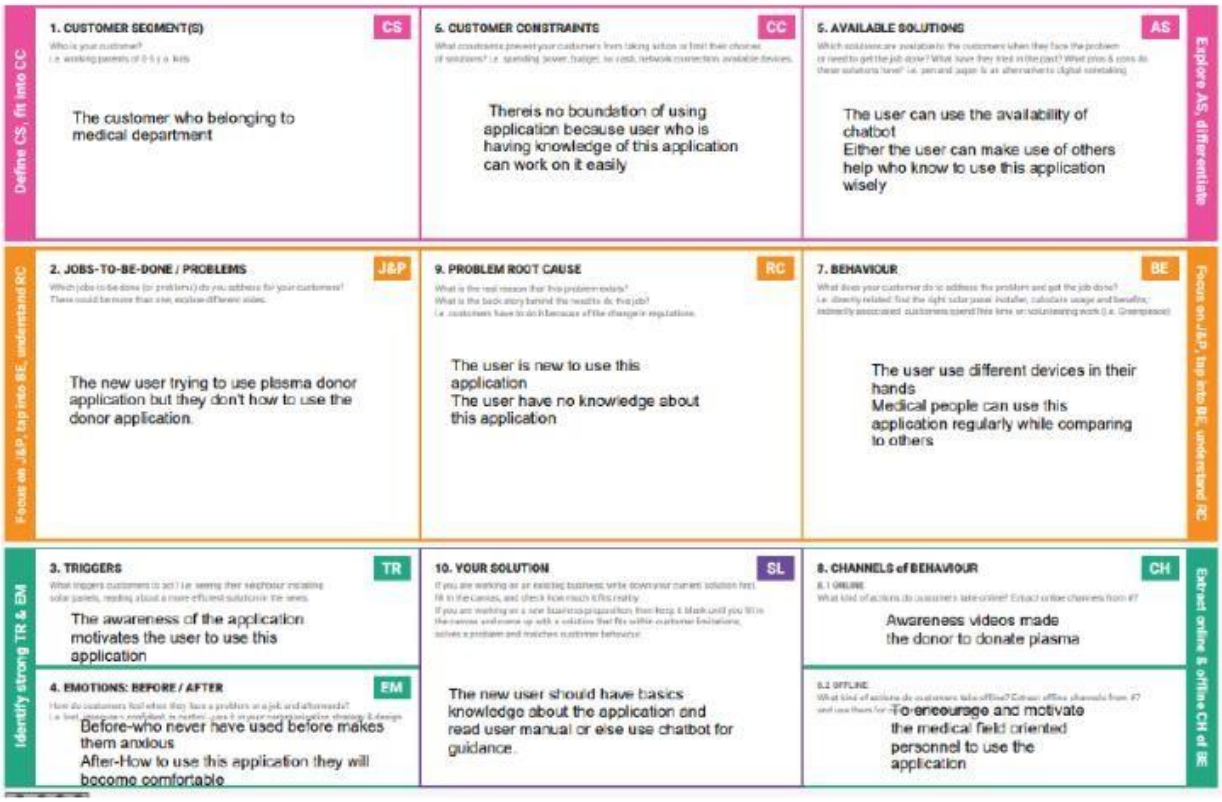
On the revenue bases, this donor application will be profit for Hospitals, NGO's and private organizations. 6.Scalability of the Solution:

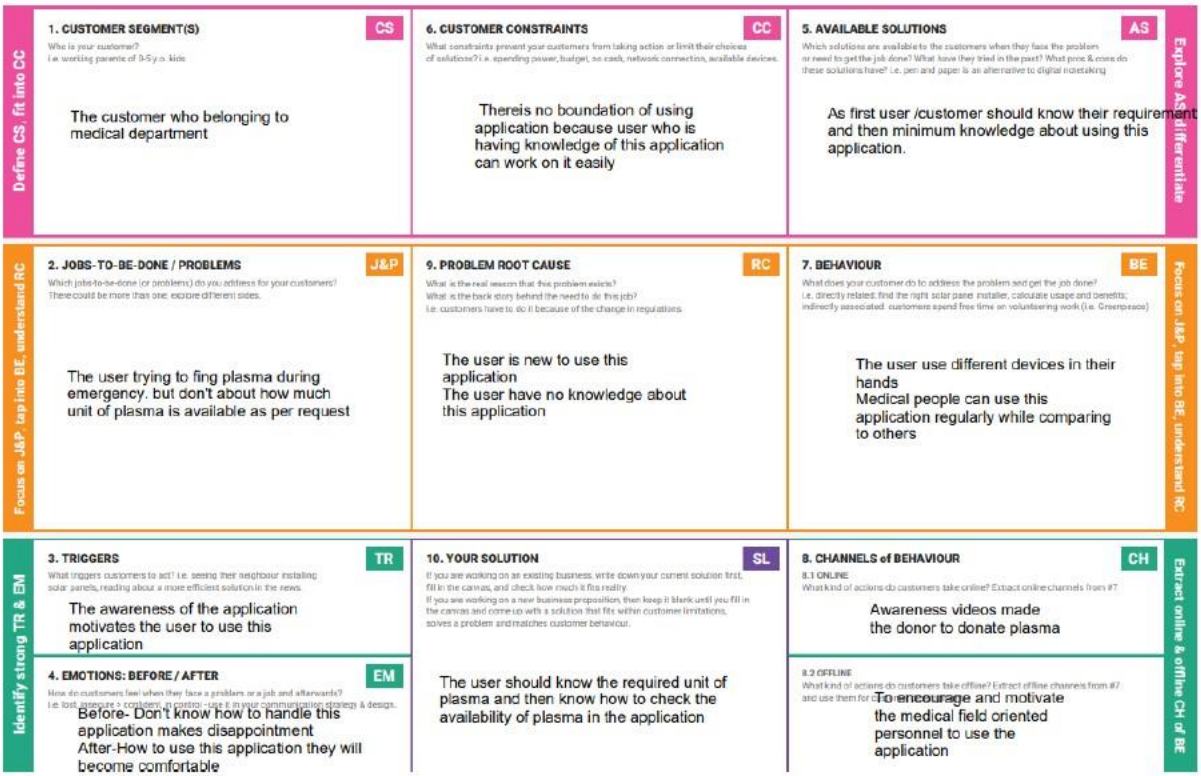
The user mindset about the application will be changed and also as per user flexibility the requirements can be modified.

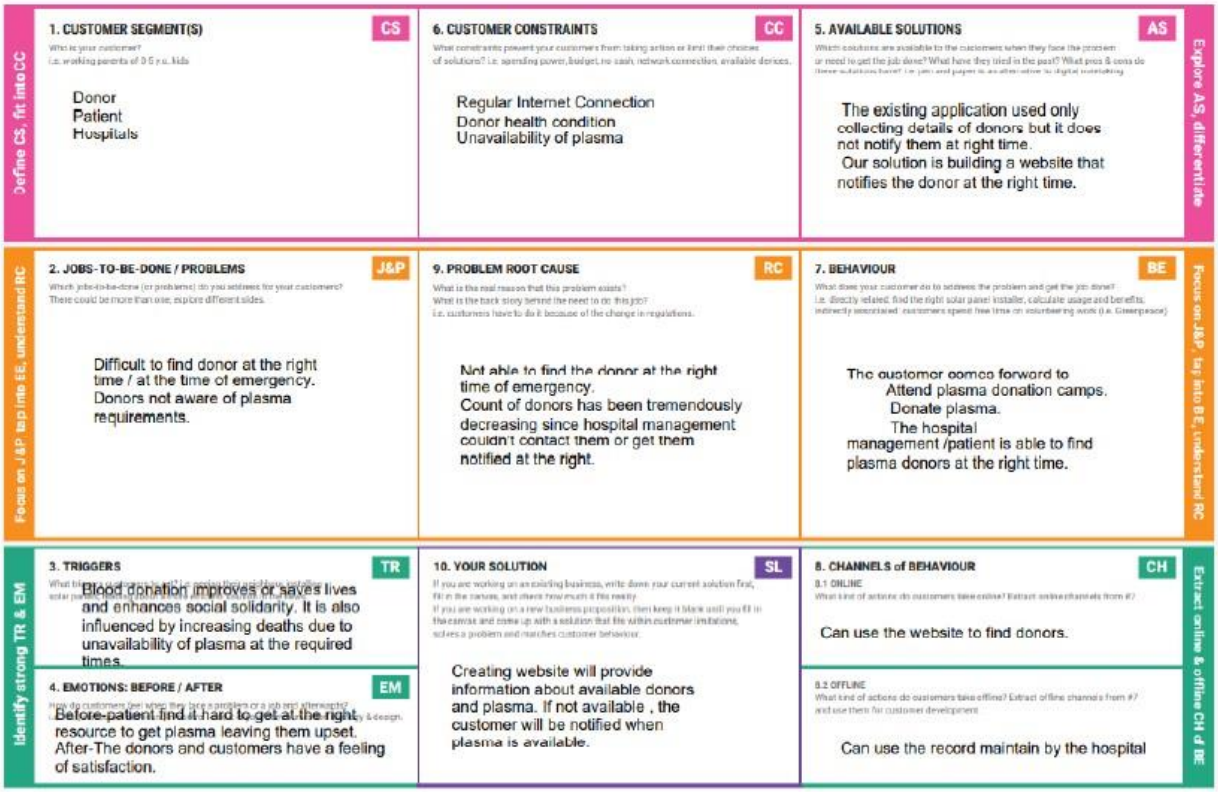
3.4 PROBLEM SOLUTION FIT :


Define CS, fit into CC
1. CUSTOMER SEGMENT(S)
CS
Who is your customer?
i.e. working parents of 0-5 y.o. kids
The user/customer who belonging to medical department
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.
There is no boundation of using this application because the user/customer who is having knowledge of this application can work on it easily
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
The suggestion made by the user/customer are implemented in these kinds of applications.
In the such cases the most important suggesstions of the user /customer are developed and made available in updates
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.
The awareness of the application motivates the user to use this application.
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
The user/customer is new to this application.
The user/customer have no knowledge about this application.
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 user/customer use different devices in their hands.
Medical people can use this application regularly while comparing to others.
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.
The awareness of this application motivates the users to use this applications.
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-expected specification not met
After-who recovered from the error they will become comfortable.
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.
The suggestion which made by the user will be noted and the apt suggestions will be added in further updates
8. CHANNELS of BEHAVIOUR
CH
8.1 ONLINE
What kind of actions do customers take online? Extract online channels from #7
Advertise online videos with influence to test the product and promote it.
8.2 OFFLINE
What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.
To encourage and motivate the medical field oriented personnel to use this application.
Extract online & offline CH of BE


Define CS, fit into CC
1. CUSTOMER SEGMENT(S)
CS
The customer who belonging to medical department
6. CUSTOMER CONSTRAINTS
CC
Thereis no boundation of using application because user who is having knowledge of this application can work on it easily
5. AVAILABLE SOLUTIONS
AS
The solution for this problem is that the user should make sure of his/her donation details updated. The user can verify the details before or after updating in this application.
Explore AS, differentiate
Focus on J&P, tap into BE, understand RC
2. JOBS-TO-BE-DONE / PROBLEMS
J&P
The user continuously receiving notification for the requirement to donate plasma ,before 2 weeks only user had to donate blood or plasma
9. PROBLEM ROOT CAUSE
RC
The user is new to use this application
The user have no knowledge about this application
7. BEHAVIOUR
BE
The user use different devices in their hands
Medical people can use this application regularly while comparing to others
Focus on J&P, tap into BE, understand RC
Identify strong TR & EM
3. TRIGGERS
TR
The awareness of the application motivates the user to use this application
4. EMOTIONS: BEFORE / AFTER
EM
Before-who often receives this type of errors make them hatres
After-Who overcomes from these errors they will become comfortable
10. YOUR SOLUTION
SL
The suggestion which made by the user will be noted and the apt sugessions will be added in further updates
8. CHANNELS of BEHAVIOUR
CH
8.1 ONLINE
Awareness videos made the donor to donate plasma
8.2 OFFLINE
To encourage and motivate the medical field oriented personnel to use the application
Extract online & offline CH of BE


Define CS, fit into CC
1. CUSTOMER SEGMENT(S)
CS
The customer who belonging to medical department
6. CUSTOMER CONSTRAINTS
CC
Thereis no boundation of using application because user who is having knowledge of this application can work on it easily
5. AVAILABLE SOLUTIONS
AS
The user can use the availability of chatbot
Either the user can make use of others help who know to use this application wisely
Explore AS, differentiate
Focus on J&P, tap into BE, understand RC
2. JOBS-TO-BE-DONE / PROBLEMS
J&P
The new user trying to use plasma donor application but they don't how to use the donor application.
9. PROBLEM ROOT CAUSE
RC
The user is new to use this application
The user have no knowledge about this application
7. BEHAVIOUR
BE
The user use different devices in their hands
Medical people can use this application regularly while comparing to others
Focus on J&P, tap into BE, understand RC
Identify strong TR & EM
3. TRIGGERS
TR
The awareness of the application motivates the user to use this application
4. EMOTIONS: BEFORE / AFTER
EM
Before-who never heve used before makes them anxious
After-How to use this application they will become comfortable
10. YOUR SOLUTION
SL
The new user should have basics knowledge about the application and read user manual or else use chatbot for guidance.
8. CHANNELS of BEHAVIOUR
CH
8.1 ONLINE
Awareness videos made the donor to donate plasma
8.2 OFFLINE
To encourage and motivate the medical field oriented personnel to use the application
Extract online & offline CH of BE





4. REQUIREMENT ANALYSIS :

The selection of plasma donors should be based on regularly viewed criteria without discrimination of any kind including gender, blood group. A prospective donors health updates should be evaluated for each donation on the day of donation prior to plasma collection.

4.1 FUNCTIONAL REQUIREMENTS :

1.User Registration

- ✦ Registration through Form
- ✦ Registration through Gmail
- ✦ Registration through Linked IN

2.User Confirmation

- ✦ Confirmation via Email
- ✦ Conformation via OTP

3.Userlogin

Operator has registered then the software operator should be able to login to the web application .The login information will be stored on the database for future use.

4.Request plasma

Should be able to request plasma in an emergency situation, software operators need to define plasma group, location, require data, contact. The plasma request will be sent to the plasma bank and then Inventory to check the availability

5.Plasma stock

Receiving the plasma request from the clinic the plasma stock in the plasma bank Inventory will be searched to match the requested plasma request.Thus match plasma units will be sent to the clinic.

6.Distribution status

If the distribution seems to be delayed then the clinic manager must be able to call the distribution person to get the update revise on the distribution.

4.2 NON-FUNCTIONAL REQUIREMENTS :

1.Usability

- The cost of the plasma units are standardized.

2.Security

- Any donor can't see any details of any other donor .If a donor does not manage to provide the user name and a password three times the user automatically will log out from the website.

3.Reliability

- The system has the ability to work all the time without failures apart from network failure. A donor can have faith in the system. When the doctors found any disease in the testing stage after providing details to the donor the system keeps the secret of the donor.

4.Performance

- The system is interactive and the delays involved are less .When connecting to the server the delay is based on the distance of the 2 systems and the configuration between them so there is high probability that there will be or not a successful connection in less than 20 seconds for the sake of good communication.

5.Availability

- The system should be available all times, meaning the user can access it using application. In case if a hardware failure or database corruption, are placement page will shown. Also in case of a Hardware failure or database corruption, backups of the database should be retrieved from the application data folder and saved by the administrator.

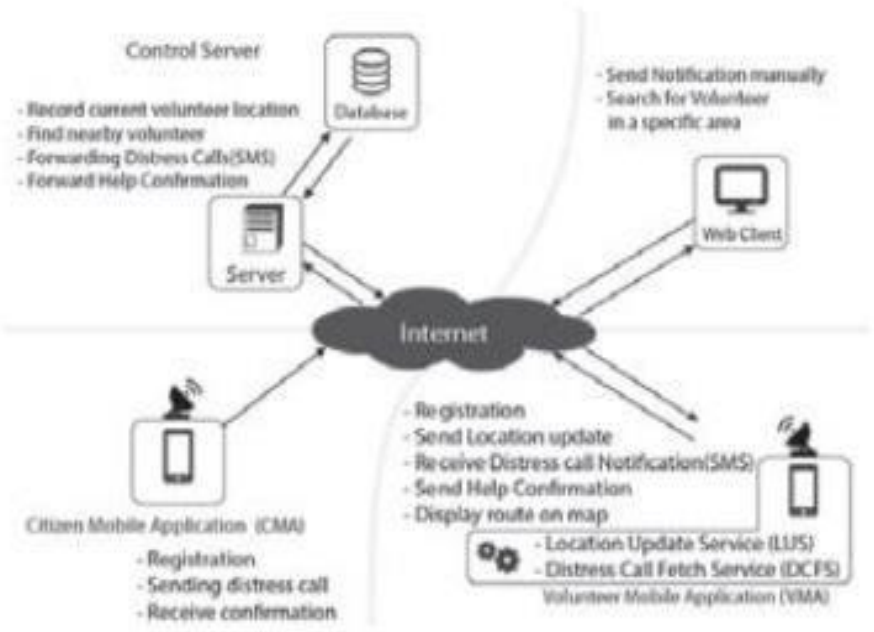
6.Scalability

- In the application to handle an increase in workload without performance dehydration ,or its ability to quickly enlarge. The solution must allow the hardware end of the deployed software service and components to be scaled horizontally as well as vertically.

5.PROJECT DESIGN

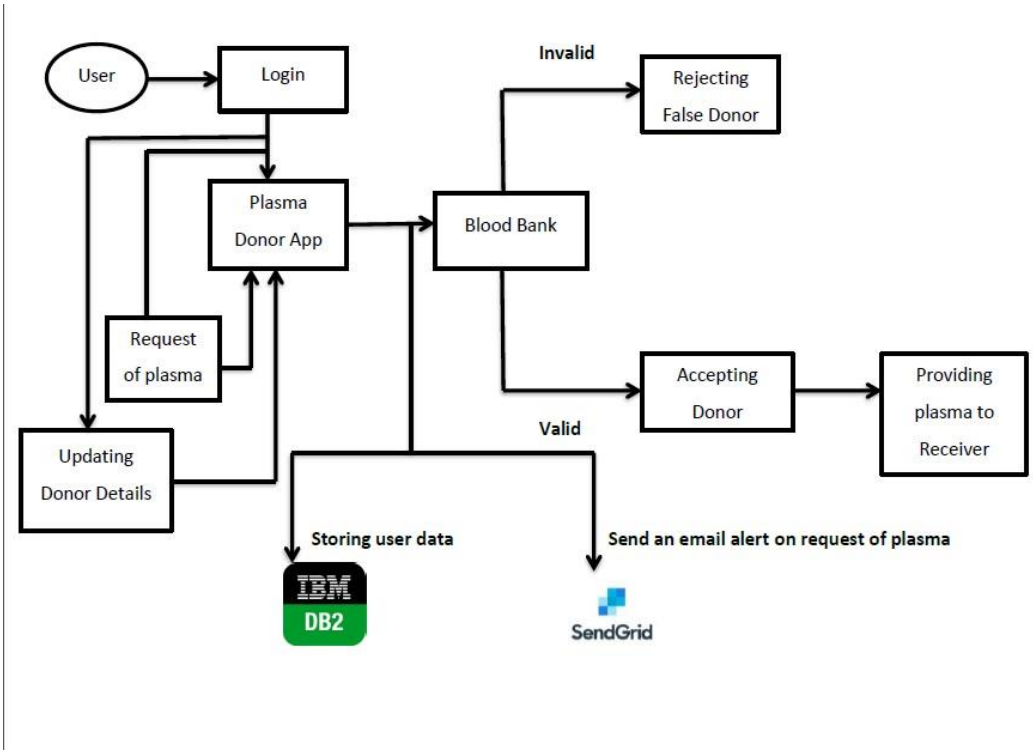
5.1 DATA FLOW DIAGRAMS :

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

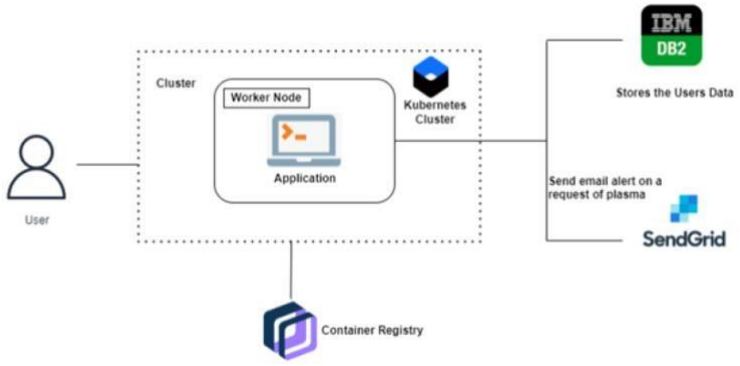


5.2 SOLUTION & TECHNICAL ARCHITECTURE :

SOLUTION ARCHITECTURE :



TECHNICAL ARCHITECTURE :



COMPONENTS & DESCRIPTION:

1. User Interface

The user register and login. See the UI.

Technology : HTML, CSS, Python Flask

2. Data maintenance

Store , maintain ,retrieve the user's details

Technology : MYSQL

3. Chatbot

Clarify user queries.

Technology : IBM Watson service

4. Confirmation Email

Sending the confirmation email to users they registered successfully

Technology :SendGrid

5. Cloud Database

Cloud database to store plasma information and View Plasma information

Technology : IBM DB2

6. File Storage

File storage requirements

Technology : IBM Block Storage

7. Infrastructure (Server / Cloud)

To deploy the application on Local System

Technology : Kubernetes

APPLICATION CHARACTERISTICS :

1. Open-Source Frameworks

Python Flask frameworks is used

Technology : Python Flask

2. Security Implementations

Mandatory Control(MAC) and kubernetes is used

Technology : SHA-256, Encryptions, IAM Controls,OWASPetc.

3. Scalable Architecture

3-Tier Architecture is used

Technology :Webserver-HTML,CSSapplication server-python flask,database server-IBM DB2

4. Availability

Using Load balancer to distribute network traffic across servers

Technology : IBM Load balancer

5. Performance

User friendly UI Request and response is faster

Technology : IBM content delivery network

5.3 USER STORIES :

USER TYPE : Customer (Mobile user)

Registration :

- As a user, I can register for the application by entering my email, password, and confirmation my password.
- As a user, I will receive confirmation email once I have registered for the application.
- As a user, I can register for the application through Facebook.
- As a user, I can register for the application through Gmail. **Login :**

✞ As a user I can register my health status.

Dashboard :

✞ As a User ,I can able to update or edit my profile.

Web page :

✞ As a user ,I able to access all the details from the web page for contact the server.

USER TYPE : Customer Care Executive **Data base:**

✞ As a user I can get all plasma donors details from the server.

USER TYPE : Administrator
Application :

- As a user I can fully satisfied with this idea

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION :

Registration &

Login : Sprint -1

- Create UI to interact with pages. To create the user and admin login functionalities.

  Story Points-2

  Priority :High

Cloud and Databases :

Sprint -2

- Connecting flask with database [IBMDB2] Implementing of IBM chatbot.

  Story points-2

  Priority :High

Deployment in Development phase :

Sprint -3

- Creating images with Docker, Deploying Kubernetes and the mailing service.

  Story points-2

  Priority :High

Testing and Development to user :

Sprint-4

- To make sure that the software is handy to users.

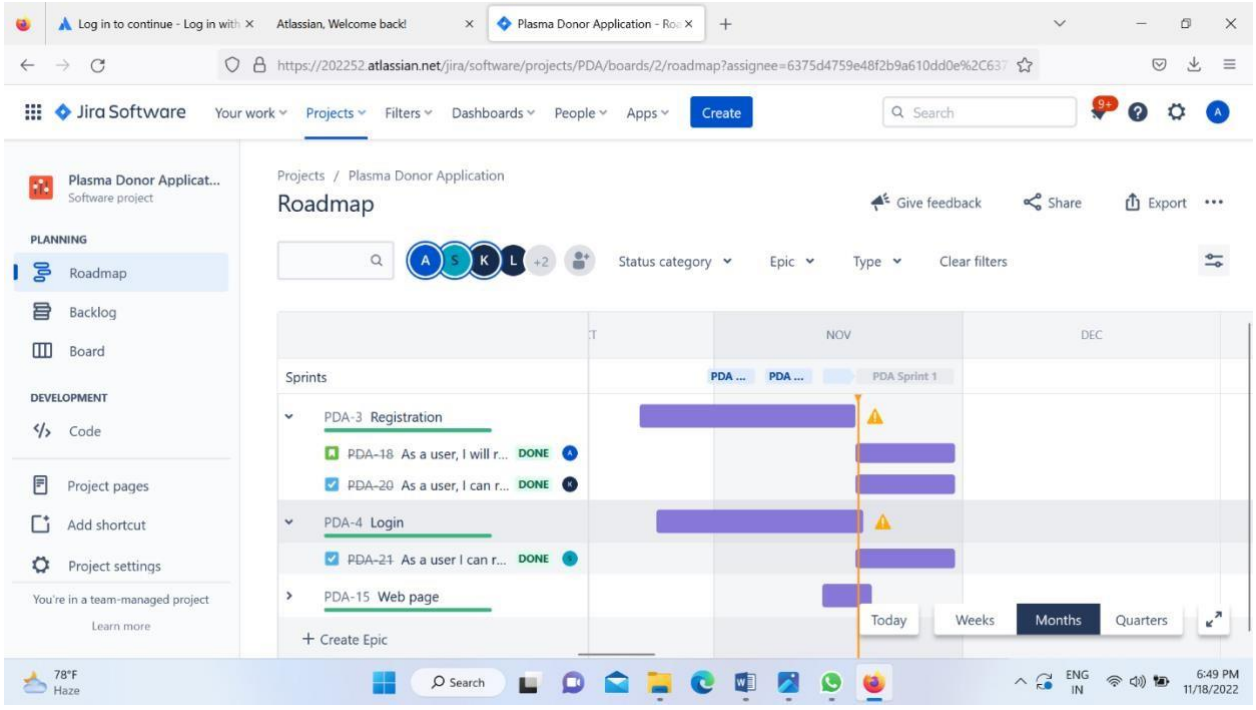
  Story points-2

  Priority :High

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 Oct2022	29 Oct2022	20	29 Oct2022
Sprint-2	20	6 Days	31 Oct2022	05 Nov2022	20	05 Nov2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov2022	20	12 Nov2022
Sprint-4	20	6 Days	14 Nov2022	19 Nov2022	20	19 Nov2022

6.3 REPORTS FROM JIRA :



7.CODING & SOLUTIONING

7.1 FEATURE CODE :

```
from distutils.log import debug # from sendgridmail import
sendmail from flask import Flask, render_template, request,
redirect, url_for, session from flask_mail import Mail, Message
import re import os import ibm_db from dotenv import
load_dotenv


load_dotenv()



app = Flask(__name__)


app.secret_key = 'a'
print("Try to
connect to Db2")
conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=2f3279a5-
73d1-485988f0-
a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PO
RT=
;UID=;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.
crt;PWD=
", ",")
print("Connected Successfully")
```

```
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT'] = 465
app.config['MAIL_USERNAME'] = 'example@gmail.com'
app.config['MAIL_PASSWORD'] = '*******'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = True mail = Mail(app)

@app.route('
/')
@app.route('
/login') def
login():

    return render_template('login.html')


@app.route('/loginpage',methods=['GE
T', 'POST']) def loginpage():     global
userid     msg = ''

    if request.method == 'POST' :
        username = request.form['username']         password =
request.form['password']         sql = "SELECT * FROM donors
WHERE username =? AND password=?"         stmt =
ibm_db.prepare(conn, sql)
```

```
ibm_db.bind_param(stmt,1,username)
ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
account                         =
ibm_db.fetch_assoc(stmt)
print (account)         if account:
            session['loggedin']  =  True                        session['id']  =
account['USERNAME']                 userid=  account['USERNAME']
session['username'] = account['USERNAME']           msg = 'Logged
in successfully !'              index(account['EMAIL'],'Plasma donor App
login','You  are  successfully  logged  in!')                       return
redirect(url_for('dash'))         else:
            msg = 'Incorrect username /
password !'     return
render_template('login.html', msg = msg)

@app.route('/regist
ration') def home():
    return render_template('register.html')

@app.route('/register',methods=['GET
', 'POST']) def register():     msg = ''
if request.method == 'POST' :
username = request.form['username']
email = request.form['email']
```

```
        password = request.form['password']
phone = request.form['phone']         city =
request.form['city']         infect =
request.form['infect']         blood =
request.form['blood']         sql = "SELECT *
FROM donors WHERE username =?"         stmt =
ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,username)
ibm_db.execute(stmt)         account =
ibm_db.fetch_assoc(stmt)         print("ac",account)
if account:
            msg = 'Account already exists !'
elif not re.match(r'[^@]+@[^@]+\.[^@]+',
email):
            msg = 'Invalid email address !'
elif not re.match(r'[A-Za-z0-9]+',
username):
            msg = 'name must contain only characters and
numbers !'         else:
            insert_sql = "INSERT INTO  donors VALUES (?, ?, ?, ?, ?, ?,
?)"            prep_stmt = ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prep_stmt, 1, username)
ibm_db.bind_param(prep_stmt, 2, password)
ibm_db.bind_param(prep_stmt, 3, email)
ibm_db.bind_param(prep_stmt, 4, phone)
ibm_db.bind_param(prep_stmt, 5, city)
```

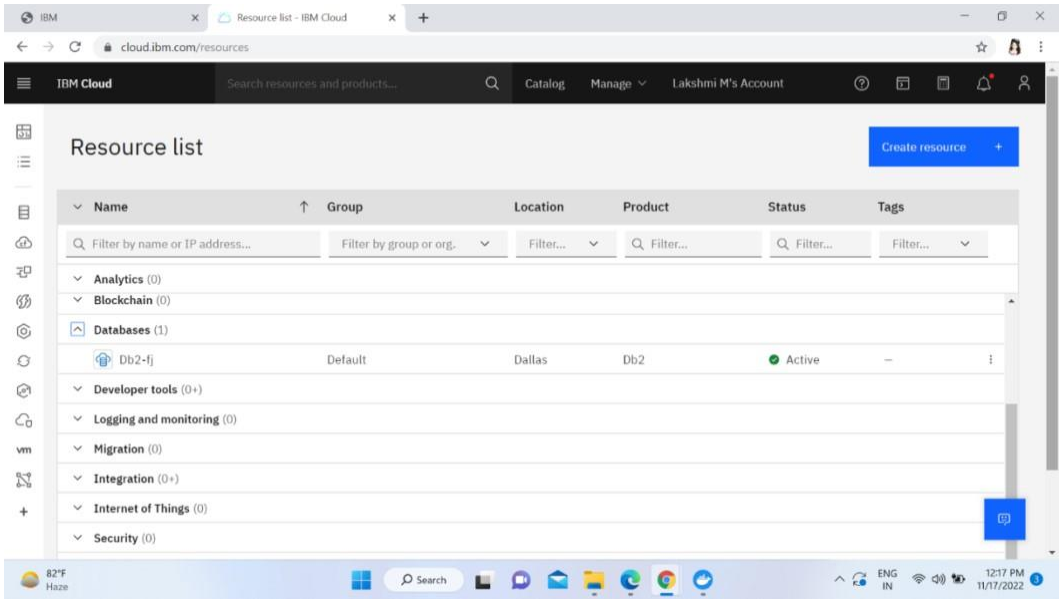
```
ibm_db.bind_param(prep_stmt, 6, infect)
ibm_db.bind_param(prep_stmt, 7, blood)

        ibm_db.execute(prep_stmt)
msg = 'You have successfully
registered, !'
        index(email,'Plasma donor App Registration','You are
successfully Registered {}!'.format(username))

   elif request.method == 'POST':
msg = 'Please fill out the form !'     return
render_template('register.html', msg =
msg)

@app.route('/dashboa
rd') def dash():     if
session['loggedin'] ==
True:
      sql = "SELECT COUNT(*), (SELECT COUNT(*) FROM
DONORS WHERE blood= 'O Positive'), (SELECT COUNT(*) FROM
DONORS WHERE blood='A Positive'), (SELECT COUNT(*) FROM
DONORS WHERE blood='B Positive'), (SELECT COUNT(*) FROM
DONORS WHERE blood='AB Positive'),
(SELECT COUNT(*) FROM DONORS WHERE blood='O Negative'),
(SELECT
COUNT(*) FROM DONORS WHERE blood='A Negative'), (SELECT
COUNT(*) FROM DONORS WHERE blood='B Negative'),
(SELECT COUNT(*) FROM DONORS WHERE blood='AB
Negative') FROM donors"        stmt = ibm_db.prepare(conn, sql)
```

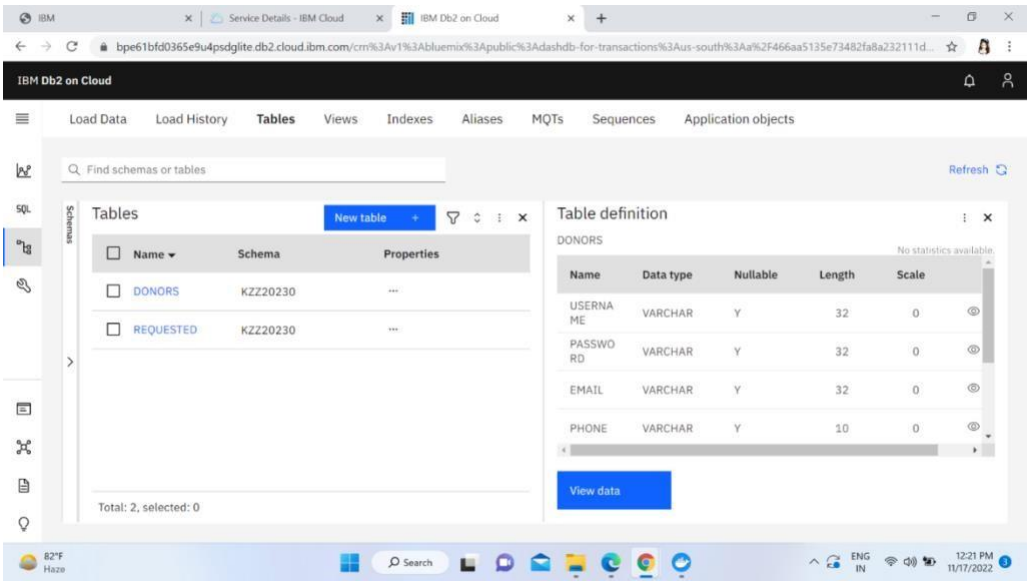
```
ibm_db.execute(stmt)        account = ibm_db.fetch_assoc(stmt)
print(account)        return
render_template('dashboard.html',b=account)
   else:
       msg = 'Please login!'        return
render_template('login.html', msg = msg)

@app.route('/requeste
r') def requester():
if session['loggedin']
== True:
       return
render_template('request.html')
else:
       msg = 'Please login!'        return
render_template('login.html', msg = msg)

@app.route('/requested',methods=['P
OST']) def requested():
   bloodgrp = request.form['bloodgrp']        address =
request.form['address']     name= request.form['name']
email=        request.form['email']                phone=
request.form['phone']      insert_sql = "INSERT INTO
requested VALUES (?, ?, ?, ?, ?)"        prep_stmt =
ibm_db.prepare(conn,                    insert_sql)
ibm_db.bind_param(prep_stmt,        1,        bloodgrp)
```

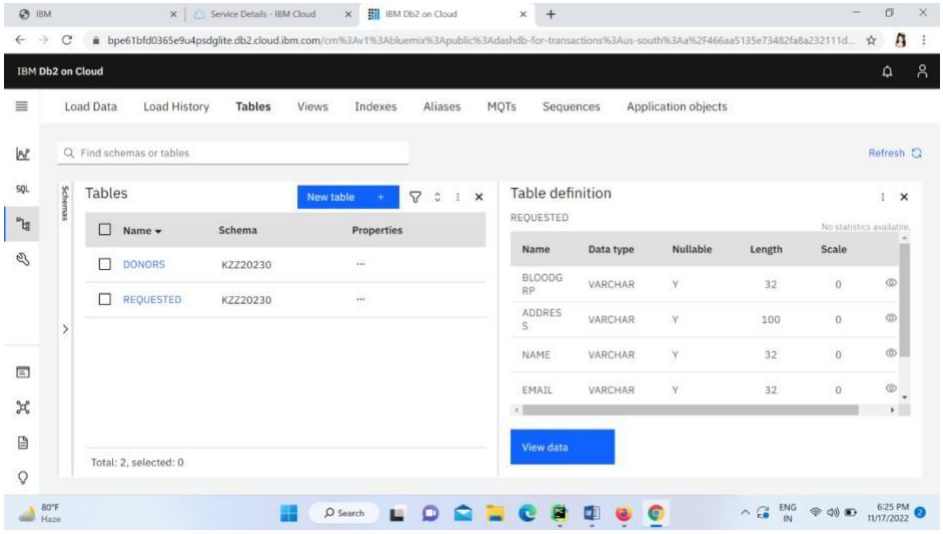
```
ibm_db.bind_param(prep_stmt, 2, address)
ibm_db.bind_param(prep_stmt, 3, name)
ibm_db.bind_param(prep_stmt, 4, email)
ibm_db.bind_param(prep_stmt, 5, phone)
ibm_db.execute(prep_stmt) index(email,'Plasma
donor App plasma request','Your request for plasma is
recieved.') return render_template('request.html',
pred="Your request is sent to the concerned people.")


def index(usermail,subject,content):

  msg = Message(subject, sender = 'example@gmail.com', recipients =
[usermail]) msg.body = format(content) mail.send(msg) return
"Sent"

@app.route('/l
ogout') def
logout():
  session.pop('loggedin',
None) session.pop('id',
None)
session.pop('username',
None) return
render_template('login.htm
```

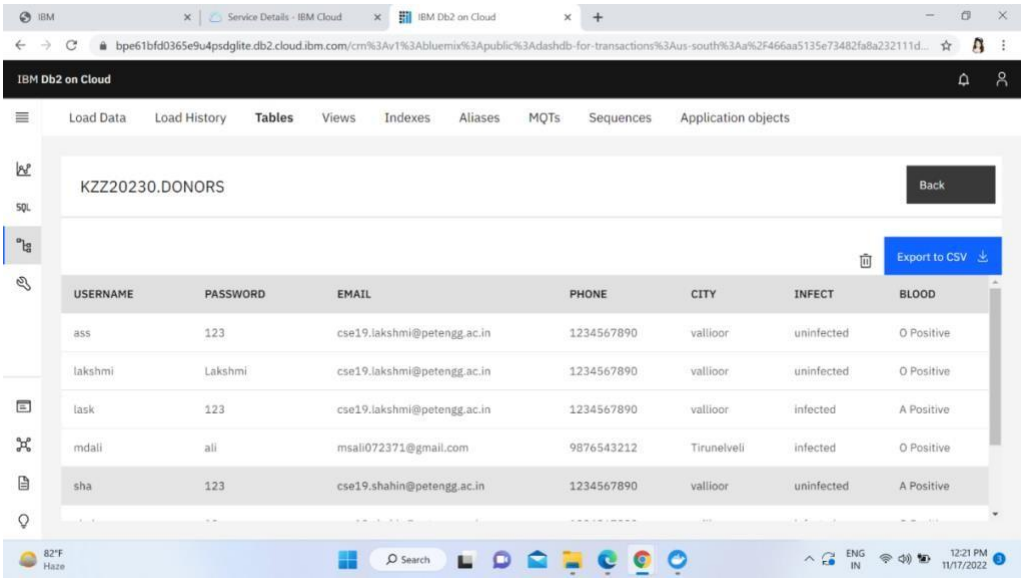
```
l')    if    __name__    ==
'__main__':
   app.run(host='0.0.0.0',debug='TRUE')
```

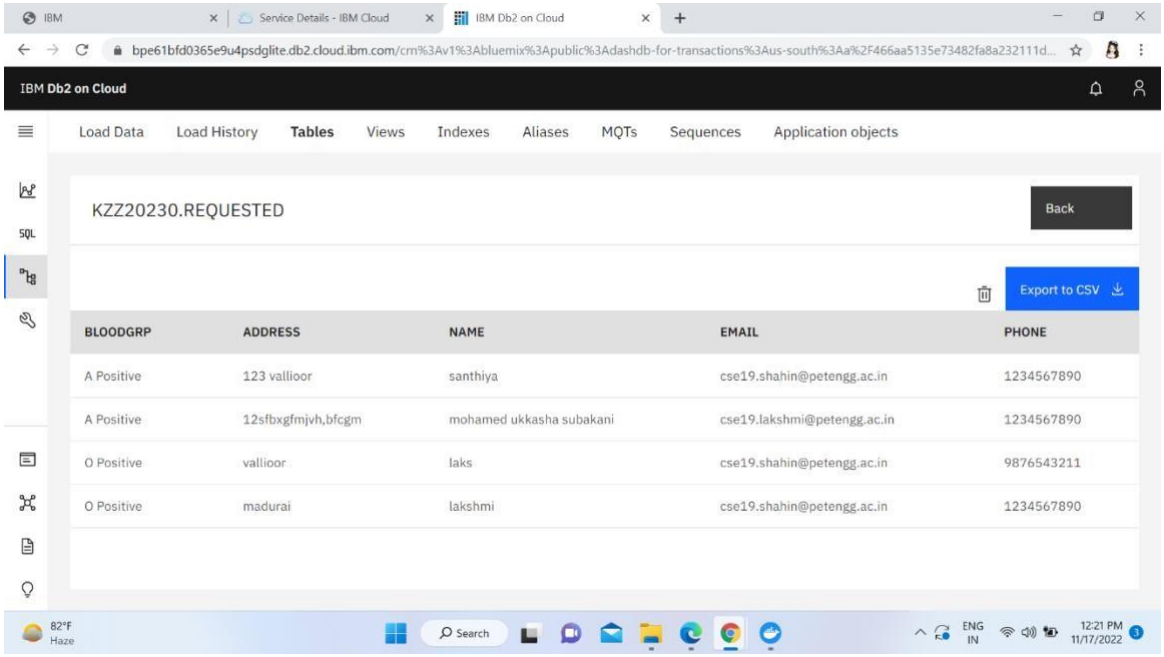
7.2 DATABASE SCHEMA :




IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Tables
New table
Name
Schema
Properties
DONORS
KZZ20230
REQUESTED
KZZ20230
Total: 2, selected: 0
Table definition
DONORS
No statistics available.
Name
Data type
Nullable
Length
Scale
USERNAME
VARCHAR
Y
32
0
PASSWORD
VARCHAR
Y
32
0
EMAIL
VARCHAR
Y
32
0
PHONE
VARCHAR
Y
10
0
View data


IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Tables
New table
Name
Schema
Properties
DONORS
KZZ20230
REQUESTED
KZZ20230
Total: 2, selected: 0
Table definition
REQUESTED
No statistics available.
Name
Data type
Nullable
Length
Scale
BLOODGRP
VARCHAR
Y
32
0
ADDRESS
VARCHAR
Y
100
0
NAME
VARCHAR
Y
32
0
EMAIL
VARCHAR
Y
32
0
View data

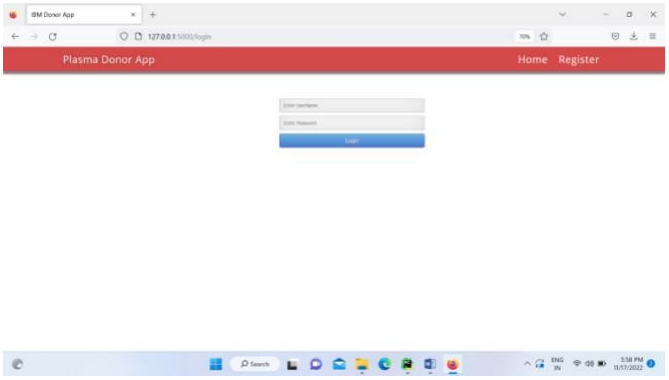

IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
KZZ20230.DONORS
Back
Export to CSV
USERNAME
PASSWORD
EMAIL
PHONE
CITY
INFECT
BLOOD
ass
123
cse19.lakshmi@petengg.ac.in
1234567890
vallioor
uninfected
O Positive
lakshmi
Lakshmi
cse19.lakshmi@petengg.ac.in
1234567890
vallioor
uninfected
O Positive
lask
123
cse19.lakshmi@petengg.ac.in
1234567890
vallioor
infected
A Positive
mdali
ali
msali072371@gmail.com
9876543212
Tirunelveli
infected
O Positive
sha
123
cse19.shahin@petengg.ac.in
1234567890
vallioor
uninfected
A Positive


IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
KZZ20230.REQUESTED
Back
Export to CSV
BLOODGRP
ADDRESS
NAME
EMAIL
PHONE
A Positive
123 vallioor
santhiya
cse19.shahin@petengg.ac.in
1234567890
A Positive
12sfbxgfmjvh,bfcgm
mohamed ukkasha subakani
cse19.lakshmi@petengg.ac.in
1234567890
O Positive
vallioor
laks
cse19.shahin@petengg.ac.in
9876543211
O Positive
madurai
lakshmi
cse19.shahin@petengg.ac.in
1234567890

8. RESULTS

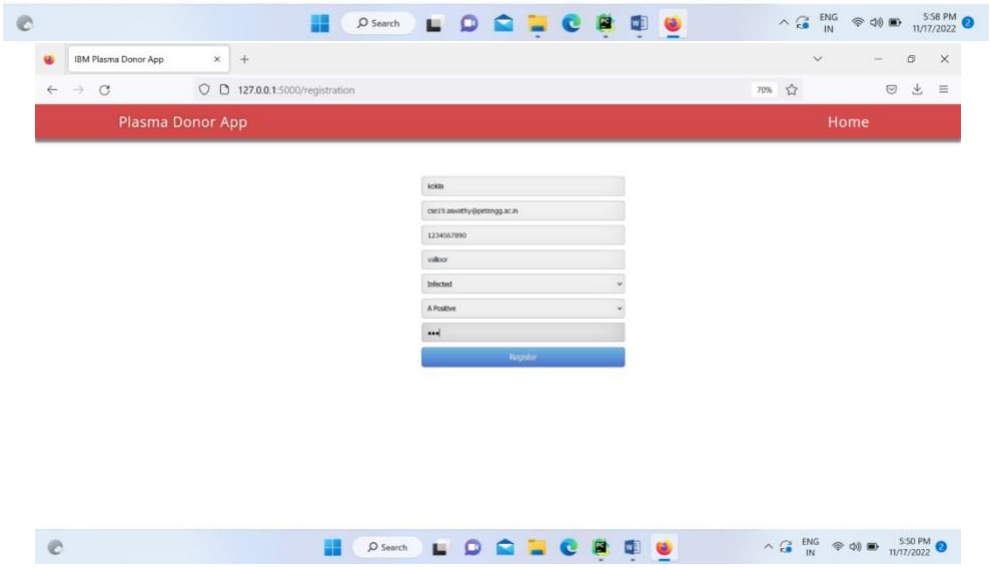
This results are represented with the capabilities of different login in mobile application. It tells the systematic process of each user login capabilities in a systematic way.

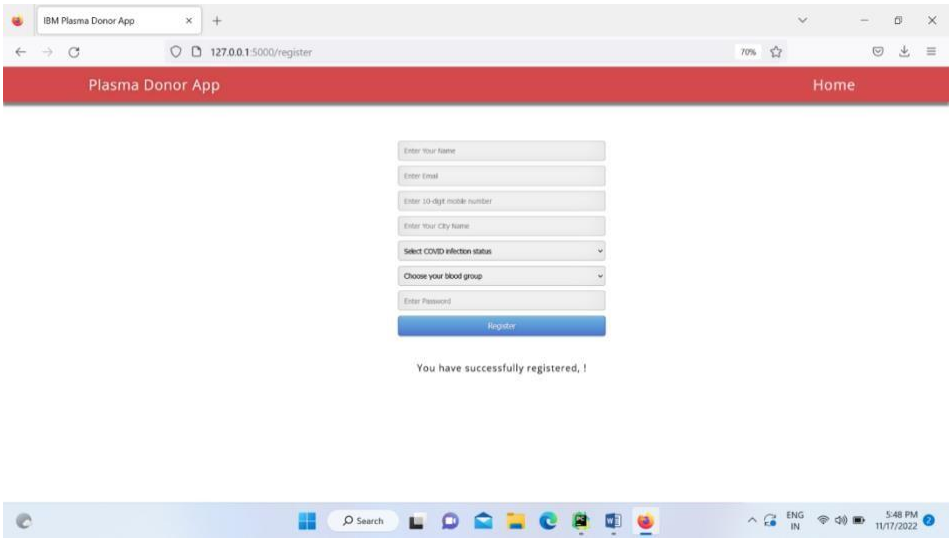
HOME PAGE:



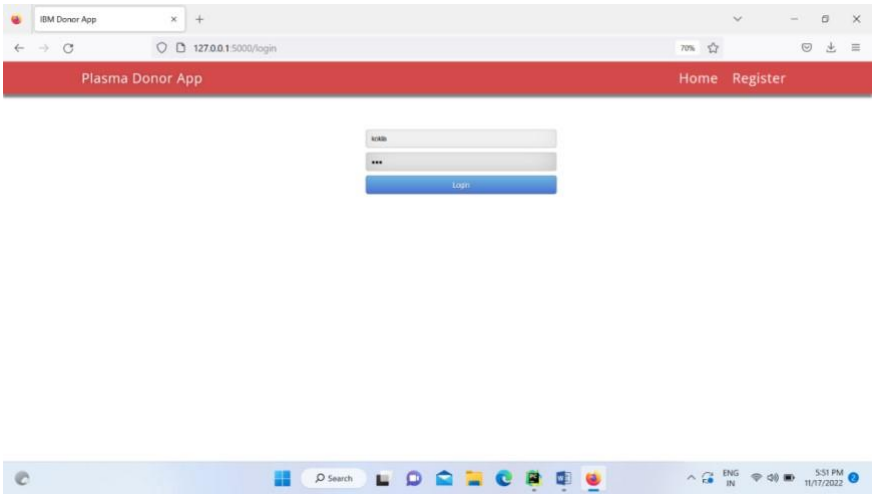
REGISTER PAGE:


IBM Plasma Donor App
127.0.0.1:5000/registration
Plasma Donor App
Home
Enter Your Name
Enter Email
Enter 10-digit mobile number
Enter Your City Name
Select COVID infection status
Choose your blood group
Enter Password
Register

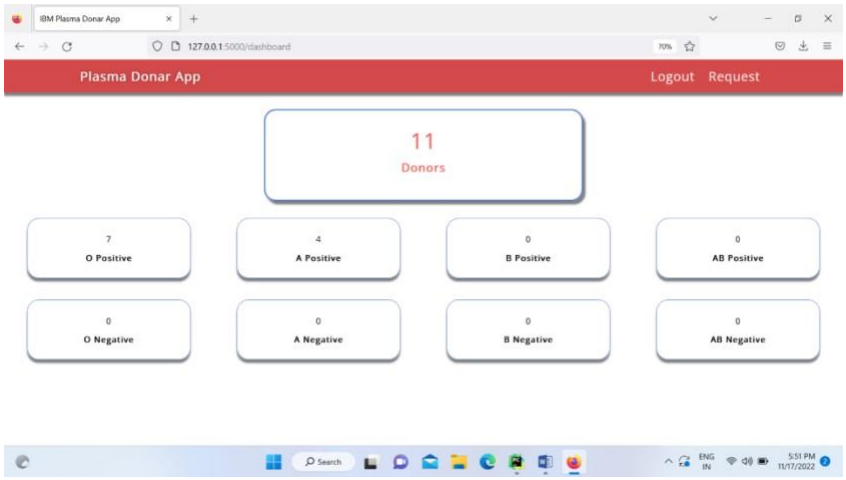

IBM Plasma Donor App
127.0.0.1:5000/registration
Plasma Donor App
Home
kokila
cse19.aswathy@pettengg.ac.in
1234567890
vallioor
Infected
A Positive
Register



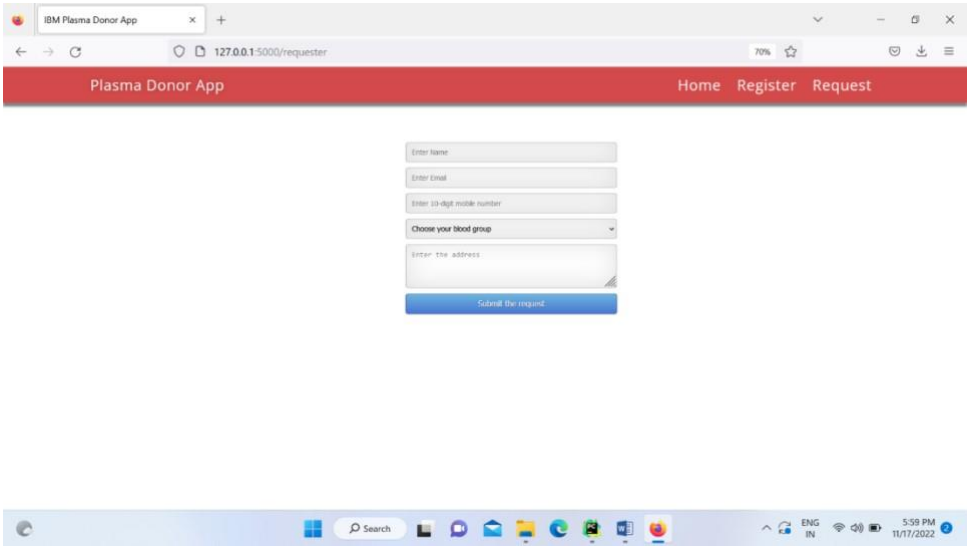
LOGIN PAGE:

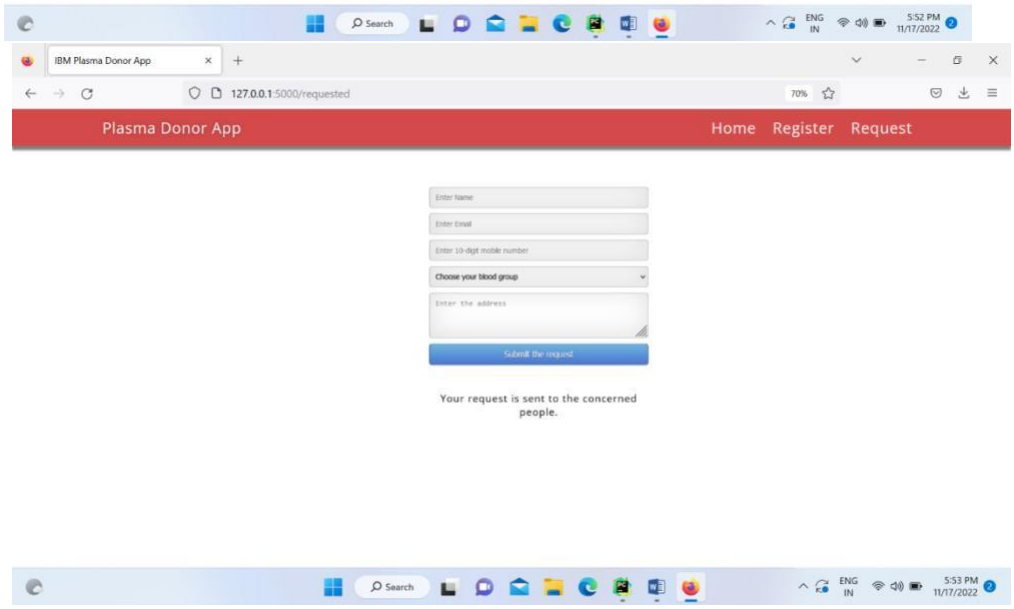


DASHBOARD:



REQUEST PAGE:





8.1 PERFORMANCE METRICS :

Being well-hydrated is also the best way to be efficient with your time. Since plasma is mostly water, drinking the recommended amount of water can help make the donation process go faster. They should be passionate, committed, curious, honest and reliable. They should also possess good communication skills, good ethics and be creative. Key factors to look for include past giving, wealth markers, business affiliations, and philanthropic tendencies. Overall, this information helps nonprofits determine a donor's ability and desire to donate to a specific cause.

9.ADVANTAGES & DISADVANTAGES

ADVANTAGES :

- ✞ Easy connecting donors and recipients makes plasma donation way more proficient.
- ✞ Prime motive of the app is to solve the perpetual shortfall of plasma donors.
- ✞ It connects plasma donors and recipients through a single and scalable platform. ✞ Effortless access : Users on this platform will be able to use the app with just one click.
- ✞ Easy registration through the mobile app will help getting quick access from both ends.

DISADVANTAGES :

- ✞ Only with proper internet connection to use this application.
- ✞ It cannot an auto verify user genuiness.

10. CONCLUSION

Enhanced mobile application for plasma has been developed to help the administrator to attend more donors and recipients and make user management an easy task. This mobile application will attract more users as it is user friendly. his system proposed here aims at connecting the donors & the patients by an online application.

By using this application, the users can either raise a request for plasma donation or requirement. Both parties can Accept or Reject the request. User has to Upload a Covid Negative report to be able to Donate Plasma. This system is used if anyone needs a Plasma Donor Blood and Plasma donation is a kind of citizen's social responsibility in which an individual can willingly donate blood/plasma via our app. This Application has been created with the concept and has sought to make sure that the donor gives blood/plasma to community.

This model is made user friendly so anybody can view and maintain his/her account. This application will break the chain of business through blood/plasma and help the poor to find donor at free of cost. This project will help new blood/plasma banks improve their services and progress from traditional to user-friendly framework

11.FUTURE SCOPE

The sole purpose of this project is to develop a computer system that will link all donors, control plasma transfusion service and create database to hold data on stocks of plasma in each area. Furthermore people will be able to see which patients need plasma supplies via android application.

User interface (UI) can be improved in future to accommodate global audience by supporting different languages across countries. Data scraping can be done from different social networks and can be shown in the Blood/Plasma Request Feeds.

Appointments can be synchronized with Google and Outlook calendars for the ease of users. Donor and Beneficiary Stories feature aims to create a sense of belonging to the community. Donors will be able to view and share personal experiencesabout their donation; Beneficiaries can share their experiences of receiving blood transfusion which contributed to their improved health and lives

. Live Check-in Process feature aims to provide a better experience with regards to the waiting time when the user is in the process of donation. We hypothesise that a more efficient experience will help the user look forward tohis blood/plasma donation appointments

12.APPENDIX

SOURCE CODE :

dashboard.html :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>IBM Plasma Donar App</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></sc
ript>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<style>
                .big{
                top:70px;
```

```
                background-
color:white;            margin-
top:80px;               margin-
left:550px;             margin-
right:550px;
height:200px;
border-radius: 25px;
border: 3px solid #4a77d4;
box-shadow: 6px 8px 4px grey;
        text-align:center;
                }
                .row{

                height:150px;

                }
                .col{
margin:10px;                    margin-
left:50px;
margin-right:50px;
border-radius: 25px;
border: 1px solid #4a77d4;
box-shadow: 0px 8px 4px grey;
        text-align:center;
                }
                .ext{
                margin-top:25px;
line-height:40px;
```

```
                }
                .ext1{
                margin-top:40px;
        line-height:50px;
font-size:25px;
color:#f95450;
                }

</style>
<body>

<div class="container-fluid">
<div class="header">
<div><b>Plasma Donar App</b></div>
<ul>
                <li><a href="/requester">Request</a></li>
                <li><a class="active" href="/logout">Logout</a></li>


        </ul>
</div>
  <br>
  <div class="big">
    <div class="box">
                <div     class="ext1"><font
size="20px">{{b['1']}}</font><br><b>Donors</b></div>
        </div>
  </div>
  <br>
  <div class="row">
```

```
  <div class="col" >
                <div class="ext">{{b['2']}}<br><b>O Positive</b></div>
        </div>
  <div class="col" >
                <div class="ext">{{b['3']}}<br><b>A Positive</b></div>
        </div>
  <div class="col" >
                <div class="ext">{{b['4']}}<br><b>B Positive</b></div>
        </div>
  <div class="col" >
                <div class="ext">{{b['5']}}<br><b>AB Positive</b></div>
        </div>
</div>
<br>
<div class="row">
  <div class="col" >
                <div class="ext">{{b['6']}}<br><b>O Negative</b></div>
        </div>
  <div class="col" >
                <div class="ext">{{b['7']}}<br><b>A Negative</b></div>
        </div>
  <div class="col" >
                <div class="ext">{{b['8']}}<br><b>B Negative</b></div>
        </div>
  <div class="col" >
                <div class="ext">{{b['9']}}<br><b>AB Negative</b></div>
        </div>
</div>
```

```
  <div style="height:200px"></div>
</div>
</body>
</html>
```

login.html

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>IBM Donor App</title>
        <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
        <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
        <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
        <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</style>
</head>
```

```
<body>
<div class="header">
<div>Plasma Donor App</div>
        <ul>




                <li><a href="/registration">Register</a></li>
                <li><a class="active" href="/login">Home</a></li>
        </ul>
</div>
 <div class="login" >
                <div>
                </div>


    <!-- Main Input For Receiving Query to our ML -->
   <form action="{{ url_for('loginpage')}}"method="post">
        <input type="text"  name="username"  placeholder="Enter
UserName" required="required" style="color:black" />
      <input type="password" name="password" placeholder="Enter Password"
required="required" style="color:black" />
      <button type="submit" class="btn btn-primary btn-block btn-
large">Login</button>


   </form>
<br><br>
<div style="color:black">
 {{ msg }}</div>
 </div>
```

```
</body>

</html>
```

register.html

```
<!DOCTYPE html>

<html >

<!--From https://codepen.io/frytyler/pen/EGdtg-->

<head>

  <meta charset="UTF-8">

  <title>IBM Plasma Donor App</title>

        <link    href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>

        <link    href='https://fonts.googleapis.com/css?family=Arimo'
rel='stylesheet' type='text/css'>

        <link    href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>

        <link
href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>

        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">


</style>

</head>


<body>

<div class="header">

<div>Plasma Donor App</div>

        <ul>
```

```
                <li><a class="active" href="/login">Home</a></li>
        </ul>
</div>
 <div class="login">

    <!-- Main Input For Receiving Query to our ML -->
   <form action="{{ url_for('register')}}"method="post">
        <input type="text" name="username" placeholder="Enter Your Name" required="required" style="color:black"/>
      <input type="email" name="email" placeholder="Enter Email" required="required" style="color:black"/>
  <input type="text" name="phone" placeholder="Enter 10-digit mobile number" required="required" style="color:black"/>
      <input type="city" name="city" placeholder="Enter Your City Name" required="required" style="color:black"/>
                <select name="infect">

                                        <option value="select" selected>Select COVID infection status</option>
                                        <option value="infected">Infected</option>
                                        <option value="uninfected">Uninfected</option>
                </select>
      <select name="blood">
                                        <option value="select" selected>Choose your blood group</option>
                                        <option value="O Positive">O Positive</option>
                                        <option value="A Positive">A Positive</option>
                                        <option value="B Positive">B Positive</option>
```

```
                                    <option value="AB Positive">AB Positive</option>
                                    <option value="O Negative">O Negative</option>
                                    <option value="A Negative">A Negative</option>
                                    <option value="B Negative">B Negative</option>
                                    <option value="AB Negative">AB Negative</option>
                </select>
        <input type="password" name="password" placeholder="Enter Password"
required="required" style="color:black"/>
        <button type="submit" class="btn btn-primary btn-block btn-
large">Register</button>


    </form>


 <br><br>
<div style="color:black">
 {{ msg }}</div>
 </div>




</body>
</html>
```

request.html

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>IBM Plasma Donor App</title>
```

```
        <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
        <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
        <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
        <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
        <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</style>
</head>

<body>
<div class="header">
<div>Plasma Donor App</div>
        <ul>

                <li><a href="/requester">Request</a></li>
                <li><a href="/registration">Register</a></li>
                <li><a class="active" href="/dashboard">Home</a></li>
        </ul>
</div>
<div class="login">
                <div>
                </div>

    <!-- Main Input For Receiving Query to our ML -->
   <form action="{{ url_for('requested')}}"method="post">
```

```
          <input type="text" name="name" placeholder="Enter Name"
required="required" style="color:black" />

      <input type="email" name="email" placeholder="Enter Email"
required="required" style="color:black"/>

  <input type="text" name="phone" placeholder="Enter 10-digit mobile number"
required="required" style="color:black"/>

                <select name="bloodgrp">
                                        <option value="select" selected>Choose your blood
group</option>
                                        <option value="O Positive">O Positive</option>
                                        <option value="A Positive">A Positive</option>
                                        <option value="B Positive">B Positive</option>
                                        <option value="AB Positive">AB Positive</option>
                                        <option value="O Negative">O Negative</option>
                                        <option value="A Negative">A Negative</option>
                                        <option value="B Negative">B Negative</option>
                                        <option value="AB Negative">AB Negative</option>
                </select>
                <textarea rows="4" placeholder="Enter the address"
required="required" style="color:black" name="address"></textarea>

      <button type="submit" class="btn btn-primary btn-block btn-large">Submit the
request</button>


   </form>

<br><br>

<div style="color:black">

 {{ pred }}</div>


 </div>
```

```
</body>
</html>
```

style.css

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
.btn {
        display: inline-block;
        *display: inline;
        *zoom: 1;
padding:        4px
10px 4px;
margin-bottom: 0;
font-size: 13px;
line-height: 18px;
color: #333333;
text-align: center;
 text-shadow: 0 1px 1px rgba(255, 255, 255,
0.75);   vertical-align: middle;   background-
color: #d70c0c;
        background-image: -moz-linear-gradient(top, #ffffff, #e6e6e6);
        background-image: -ms-linear-gradient(top, #ffffff, #e6e6e6);
        background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#ffffff),
        to(#e6e6e6));  background-image: -webkit-linear-gradient(top, #ffffff,
        #e6e6e6);
        background-image: -o-linear-gradient(top, #ffffff, #e6e6e6);
background-image: linear-gradient(top, #ffffff, #e6e6e6);    background-
repeat: repeat-x;
```

```
 filter:                progid:dximagetransform.microsoft.gradient(startColorstr=#ffffff,
endColorstr=#e6e6e6, GradientType=0);   border-color: #e6e6e6 #e6e6e6 #e6e6e6;
        border-color: rgba(0, 0, 0, 0.1) rgba(0, 0, 0, 0.1) rgba(0, 0, 0,
0.25);          border: 1px solid #e6e6e6;
        -webkit-border-radius: 4px;
-moz-border-radius: 4px;
border-radius: 4px;
        -webkit-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px
rgba(0, 0, 0, 0.05);      -moz-box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0
1px 2px rgba(0, 0, 0, 0.05);    box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0
1px 2px rgba(0, 0, 0, 0.05);    cursor: pointer; *margin-left: .3em;
        }

.btn:hover, .btn:active, .btn.active, .btn.disabled, .btn[disabled] { background-color:
#e6e6e6; }

.btn-large {
        padding: 9px
14px;           font-
size: 15px;     line-
height: normal;
        -webkit-border-radius: 5px;
        -moz-border-radius: 5px;
        border-radius: 5px;
        }

.btn:hover {
        color: #333333;
text-decoration: none;
```

```
background-color: #e6e6e6;
background-position: 0 -
15px;
        -webkit-transition: background-position 0.1s linear;
        -moz-transition: background-position 0.1s linear;
        -ms-transition: background-position 0.1s
linear;         -o-transition: background-position
0.1s linear;    transition: background-position 0.1s
linear;
        }

.btn-primary, .btn-primary:hover {
text-shadow: 0 -1px 0 rgba(0, 0, 0,
0.25);          color: #ffffff;
        }

.btn-primary.active { color: rgba(255, 255, 255, 0.75); }

.btn-primary {
        background-color: #d70c0c;
        background-image: -moz-linear-gradient(top, #6eb6de, #4a77d4);
background-image: -ms-linear-gradient(top, #6eb6de, #4a77d4);
        background-image: -webkit-gradient(linear, 0 0, 0 100%, from(#6eb6de),
to(#4a77d4));            background-image: -webkit-linear-gradient(top, #6eb6de,
#4a77d4);
        background-image: -o-linear-gradient(top, #6eb6de, #4a77d4);
        background-image: linear-gradient(top, #6eb6de, #4a77d4);
        background-repeat: repeat-x;
```

```
        filter: progid:dximagetransform.microsoft.gradient(startColorstr=#6eb6de,
endColorstr=#4a77d4,
GradientType=0);  border: 1px solid
#3762bc;  text-shadow: 1px 1px 1px
rgba(0,0,0,0.4);
        box-shadow: inset 0 1px 0 rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.5);
        }

.btn-primary:hover,  .btn-primary:active,  .btn-primary.active,  .btn-
primary.disabled,  .btnprimary[disabled] {  filter: none;  background-
color: #d70c0c
        }

.btn-block { width: 100%; display:block; }

* { -webkit-box-sizing:border-box; -moz-box-sizing:border-box; -ms-box-
sizing:border-box; -obox-sizing:border-box; box-sizing:border-box; }

html { width: 100%; height:100%; overflow:hidden; }

body {
        width: 100%;
height:100%;
        font-family: 'Open
Sans', sans-serif;  color:
#000000;  font-size: 18px;
        text-align:center; letter-
        spacing:1.2px;
```

```
}
.header {
                        top:0;
                margin:0px;
        left: 0px;
right: 0px;
position: fixed;
background: #d44a4a;
        color: black;
                        box-shadow: 0px
8px 4px grey;
overflow: hidden;
padding: 15px;
font-size: 1.5vw;
width: 100%;
                        text-align: center;
                }
.login {
        position:
absolute;       top:
70%;    left: 50%;
margin: -25px 0 0 -
150px;
width:400px;
height:400px;
}
```

```
.header div { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing:1px;
textalign:center; float:left; padding-left:150px;}


ul {
  list-
style-
type:
none;
margin:
0;
padding:
0;
padding-
right:150
px;
overflow:
hidden;
li a {
display:
block;
color:
white;
text-align:
center;
padding:
0px 15px;
text-
decoration
: none;
```

```
}


input {  width:
        100%;
        margin-
        bottom:
        10px;
        backgrou
        nd:
        rgba(255,
        255,255,
        255);
        border:
        none;
        outline:
        none;
        padding:
        10px;
        font-size:
        13px;
        color:
        black;
        text-
        shadow:
        black;
        border:
        1px solid
```

```
        rgba(0,0,
        0,0.3);
        border-
        radius:
        4px;
        box-
        shadow:
        inset 0 -
        5px 45px
        rgba(100,
        100,100,
        0.2), 0
        1px 1px
        rgba(255,
        255,255,
        0.2);
        -webkit-transition: box-shadow .5s ease;
        -moz-transition: box-shadow .5s ease;
        -o-transition: box-shadow .5s ease;
-ms-transition: box-shadow .5s ease;
transition: box-shadow .5s ease;
}
input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px
rgba(255,255,255,0.2); }


textarea {
        width: 100%;
margin-bottom: 10px;
```

```
background:
rgba(255,255,255,255);
border: none;   outline: none;
padding: 10px;
        font-size:
        13px;
        color:
        black;
        text-
        shadow:
        black;

        border: 1px solid rgba(0,0,0,0.3);
        border-radius: 4px;
        box-shadow: inset 0 -5px 45px rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);
        -webkit-transition: box-shadow .5s ease;
        -moz-transition: box-shadow .5s ease;
        -o-transition: box-shadow .5s ease;
-ms-transition: box-shadow .5s ease;
transition: box-shadow .5s ease;
}
textarea:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px
rgba(255,255,255,0.2); }

select {
        width: 100%;
margin-bottom: 10px;
background:
rgba(255,255,255,255);
```

```
border: none;   outline: none;
padding: 10px;          font-size:
13px;   color: #000000;
        text-shadow: 1px 1px 1px rgba(0,0,0,0.3);    border: 1px solid
rgba(0,0,0,0.3);          border-radius: 4px;     box-shadow: inset 0 -5px 45px
rgba(100,100,100,0.2), 0 1px 1px rgba(255,255,255,0.2);    -webkit-transition: box-
shadow .5s ease;
        -moz-transition: box-shadow .5s ease;
        -o-transition: box-shadow .5s ease;
        -ms-transition: box-shadow .5s
        ease; transition: box-shadow .5s
        ease;
}
```

app.py

```
from distutils.log import
debug # from
sendgridmail import
sendmail
from flask import Flask, render_template, request, redirect,
url_for, session from flask_mail import Mail, Message import
re import os import ibm_db from dotenv import load_dotenv

load_dotenv()
```

```
app = Flask(__name__)


app.secret_key =

'a' print("Try to

connect to Db2")

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=2f3279a5-73d1-4859-
88f0a6c3e6b4b907.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=
;UID=;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;PWD=",

",") print("Connected Successfully")

app.config['MAIL_SERVER']='smtp.gmail.co

m' app.config['MAIL_PORT'] = 465

app.config['MAIL_USERNAME'] =

'example@gmail.com'

app.config['MAIL_PASSWORD'] = '*******'

app.config['MAIL_USE_TLS'] = False

app.config['MAIL_USE_SSL'] = True mail =

Mail(app)


@app.route

('/')

@app.route

('/login')

def login():


    return render_template('login.html')
```

```
@app.route('/loginpage',methods=['G
ET', 'POST']) def loginpage():    global
userid    msg = ''

   if request.method == 'POST' :
      username =
request.form['username']        password
= request.form['password']
      sql = "SELECT * FROM donors WHERE username =? AND
password=?"        stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,username)
ibm_db.bind_param(stmt,2,password)        ibm_db.execute(stmt)

      account                    =
ibm_db.fetch_assoc(stmt)
print  (account)                 if
account:
         session['loggedin']      =      True
session['id']   =   account['USERNAME']
userid=             account['USERNAME']
session['username']                        =
account['USERNAME']              msg =
'Logged in successfully !'
         index(account['EMAIL'],'Plasma donor App login','You are
successfully logged in!')          return redirect(url_for('dash'))        else:
         msg = 'Incorrect username /
password !'    return
render_template('login.html', msg = msg)
```

```
@app.route('/regi
stration') def
home():
    return render_template('register.html')

@app.route('/register',methods=['G
ET', 'POST']) def register():     msg
= ''     if request.method == 'POST' :
username =
request.form['username']         email
= request.form['email']
password =
request.form['password']
phone = request.form['phone']
city = request.form['city']
infect = request.form['infect']
        blood = request.form['blood']
        sql = "SELECT * FROM donors WHERE
username =?"         stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,username)
ibm_db.execute(stmt)
        account                    =
ibm_db.fetch_assoc(stmt)
print("ac",account)             if
account:
            msg = 'Account already exists !'
elif not re.match(r'[^@]+@[^@]+\.[^@]+',
email):
```

```
            msg = 'Invalid email address
!'          elif not re.match(r'[A-Za-z0-
9]+', username):
            msg = 'name must contain only characters and
numbers !'          else:
            insert_sql = "INSERT INTO  donors VALUES (?, ?, ?, ?,
?, ?, ?)"             prep_stmt = ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prep_stmt, 1, username)
ibm_db.bind_param(prep_stmt, 2, password)
ibm_db.bind_param(prep_stmt, 3, email)
ibm_db.bind_param(prep_stmt, 4, phone)
ibm_db.bind_param(prep_stmt, 5, city)
ibm_db.bind_param(prep_stmt, 6, infect)
ibm_db.bind_param(prep_stmt, 7, blood)


            ibm_db.execute(prep_stmt)
msg = 'You have successfully
registered, !'
            index(email,'Plasma donor App Registration','You are successfully
Registered {}!'.format(username))

    elif request.method == 'POST':
msg = 'Please fill out the form !'
return render_template('register.html',
msg = msg)


@app.route('/dashb
oard') def dash():
if
```

```
session['loggedin']
== True:
    sql = "SELECT COUNT(*), (SELECT COUNT(*) FROM DONORS WHERE blood= 'O Positive'), (SELECT COUNT(*) FROM DONORS WHERE blood='A Positive'), (SELECT COUNT(*) FROM DONORS WHERE blood='B Positive'), (SELECT COUNT(*) FROM DONORS WHERE blood='AB Positive'), (SELECT COUNT(*) FROM DONORS WHERE blood='O Negative'), (SELECT COUNT(*) FROM DONORS WHERE blood='A Negative'), (SELECT COUNT(*) FROM DONORS WHERE blood='B Negative'), (SELECT COUNT(*)
FROM DONORS WHERE blood='AB Negative') FROM donors"
    stmt = ibm_db.prepare(conn, sql)
ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
print(account)                    return
render_template('dashboard.html',b=account
)   else:
    msg = 'Please login!'
    return render_template('login.html', msg = msg)


@app.route('/reques
ter') def requester():
if
session['loggedin']
== True:
    return
render_template('request.html')
else:
    msg = 'Please login!'      return
render_template('login.html', msg =
msg)
```

```
@app.route('/requested',methods=['POS
T']) def requested():
   bloodgrp                =
request.form['bloodgrp']
address                    =
request.form['address']
name=
request.form['name']
email=
request.form['email']
phone=
request.form['phone']
   insert_sql = "INSERT INTO  requested VALUES (?, ?,
?, ?, ?)"     prep_stmt = ibm_db.prepare(conn, insert_sql)
ibm_db.bind_param(prep_stmt, 1, bloodgrp)
ibm_db.bind_param(prep_stmt, 2, address)
ibm_db.bind_param(prep_stmt, 3, name)
ibm_db.bind_param(prep_stmt, 4, email)
ibm_db.bind_param(prep_stmt, 5, phone)
ibm_db.execute(prep_stmt)
   index(email,'Plasma donor App plasma request','Your request for plasma is
recieved.')     return render_template('request.html', pred="Your request is sent to
the concerned people.")


def index(usermail,subject,content):
```

```
  msg = Message(subject, sender = 'example@gmail.com', recipients
= [usermail])    msg.body = format(content)    mail.send(msg)    return
"Sent"


@app.route('/logout')
def logout():
  session.pop('loggedin',
None)    session.pop('id',
None)
session.pop('username',
None)    return
render_template('login.ht
ml')


if __name__ == '__main__':
  app.run(host='0.0.0.0',debug='TRUE')
```

Dockerfile

```
FROM python:3.9
WORKDIR /app
ADD . /app
COPY requirements.txt /app
RUN python3 -m pip install -r requirements.txt
EXPOSE 5000
CMD ["python","app.py"]
```

GITHUB LINK :

https://github.com/IBM-EPBL/IBM-Project-34725-1660275274