PLASMA DONOR APPLICATION

1 . INTRODUCTION

1.1 Project Overview

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.

2.2Purpose

The main goal of our project is 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. Therapy, which is considered reliable and safe. If a particular person has fully recovered from COVID19, they are eligible to donate their plasma.

2 . LITERATURE SURVEY

2.1 Existing problem

In recent days, it is noticed the increase in blood request posts on social media such as Facebook, Twitter, and Instagram. Interestingly there are many people across the world interested in donating blood when there is a need, but those donors don't have access to know about the blood donation requests in their local area. This is because there is no platform to connect local blood donors with patients. Plasma donor application solves the problem and creates a communication channel through authorized clinics whenever a patient needs blood donation. It is a useful tool to find compatible blood donors who can receive blood request posts in their local area. Clinics can use this web application to maintain the blood donation activity.

2.2 Reference

[1] A. Hossain, H. Rahaman, A. Jamil, and Dr. M. A. Khan, An algorithm for securing user credentials by combining Encryption and Hashing method, International Journal of Electrical Engineering and Applied Sciences (IJEEAS), vol. 3, no. 2, pp. 35–42, Dec. 2020.

[2] J. A. Khan and M. R. Alony, A new concept of blood bank management system using cloud computing for rural area (INDIA), International Journal of Electrical, Electronics and Computer Engineering, 4(1), pp.20-26, 2015.

[3] Javed Akhtar Khan and M. R. Aloney, Blood donor information filter based on seeker voice, International Conference on Inventive Computation Technologies (ICICT), vol. 3, pp. 1-3, 2016.

[4] S. Dhond, P. Randhavan, B. Munde, R. Patil, and V. Patil, Android

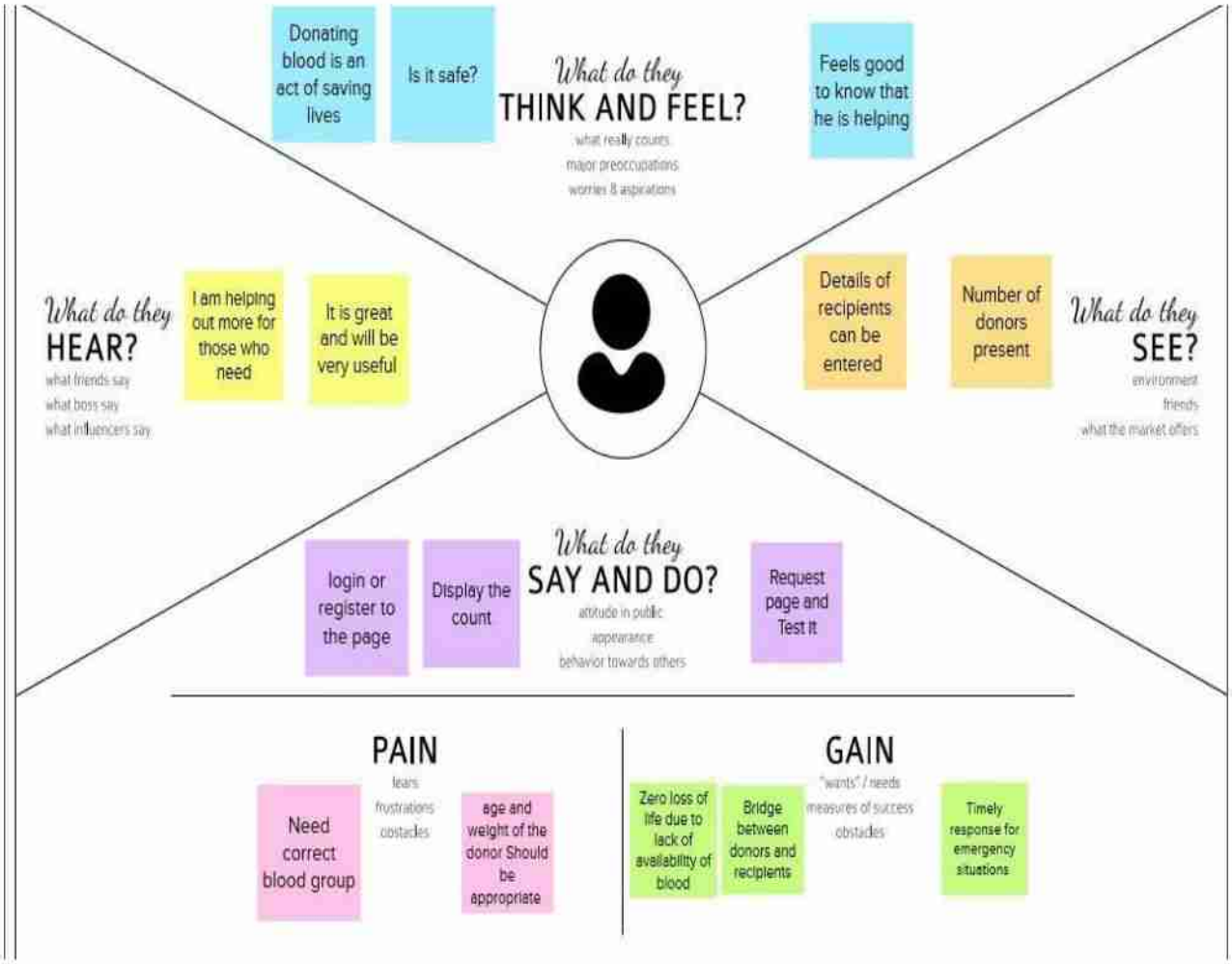
based health application in cloud computing for blood bank, International Engineering Research Journal (IERJ), 1(9),
pp.868-870, 2015.
[5] H. Jemal, Z. Kechaou, M. B. Ayed, and A. M. Alimi, Cloud computing and mobile devices based system for healthcare application, IEEE International Symposium on Technology and Society (ISTAS), pp. 1-5, 2015

2.3 Problem Statement Definition

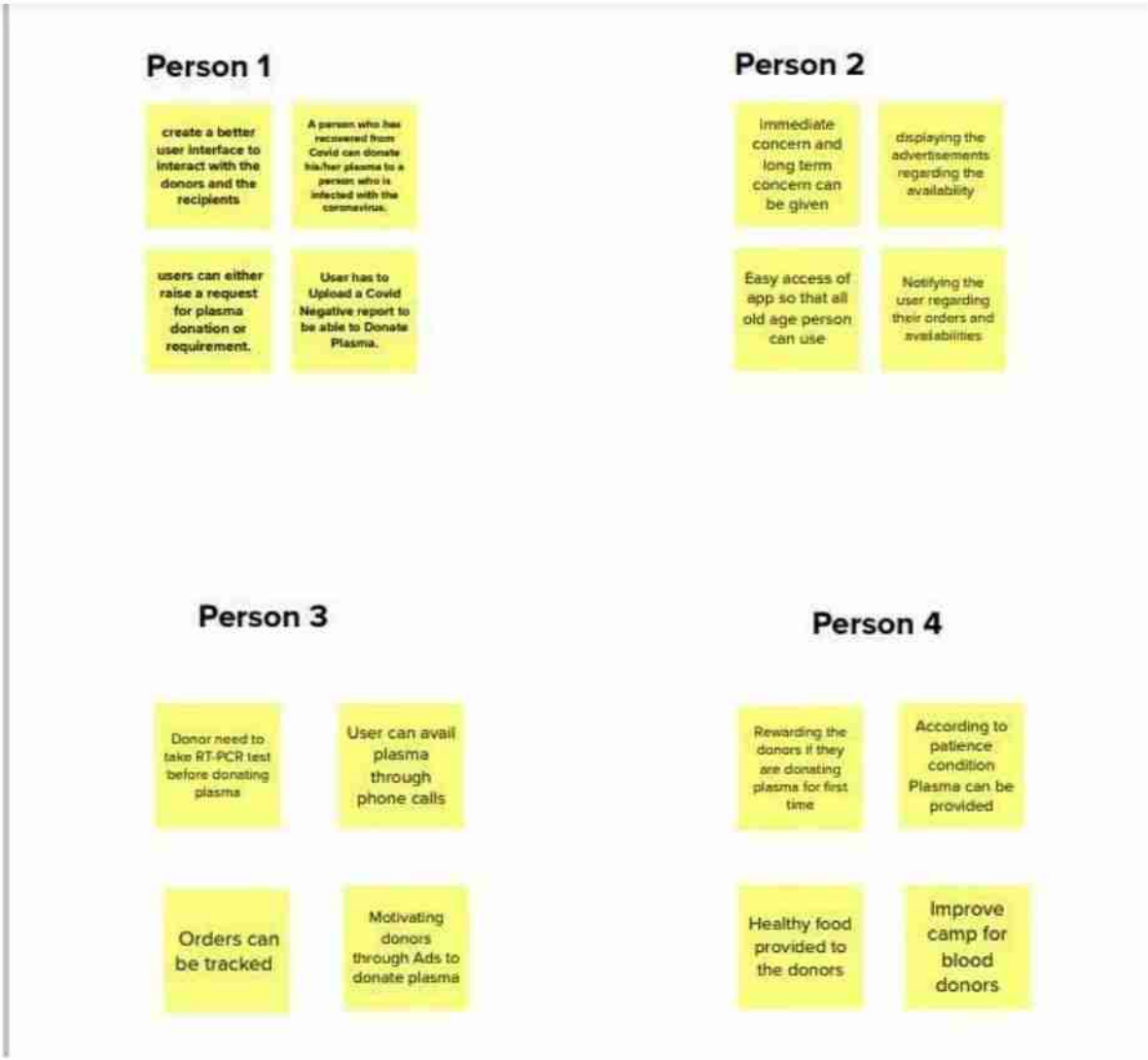
During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low. Saving the donor information and helping the needy by notifying the current donors list, would be a helping hand. In regard to the problem faced, an application is to be built which would take the donor details, store them and inform them upon a request
Documents regarding the medical details/ blood group test documents/Corona negative documents need to be uploaded in register page
Notification send to recipient if there is a donor and Donor will receive response if there is a need

3 . IDEATION & PROPOSED SOLUTION

3.1Empathy Map



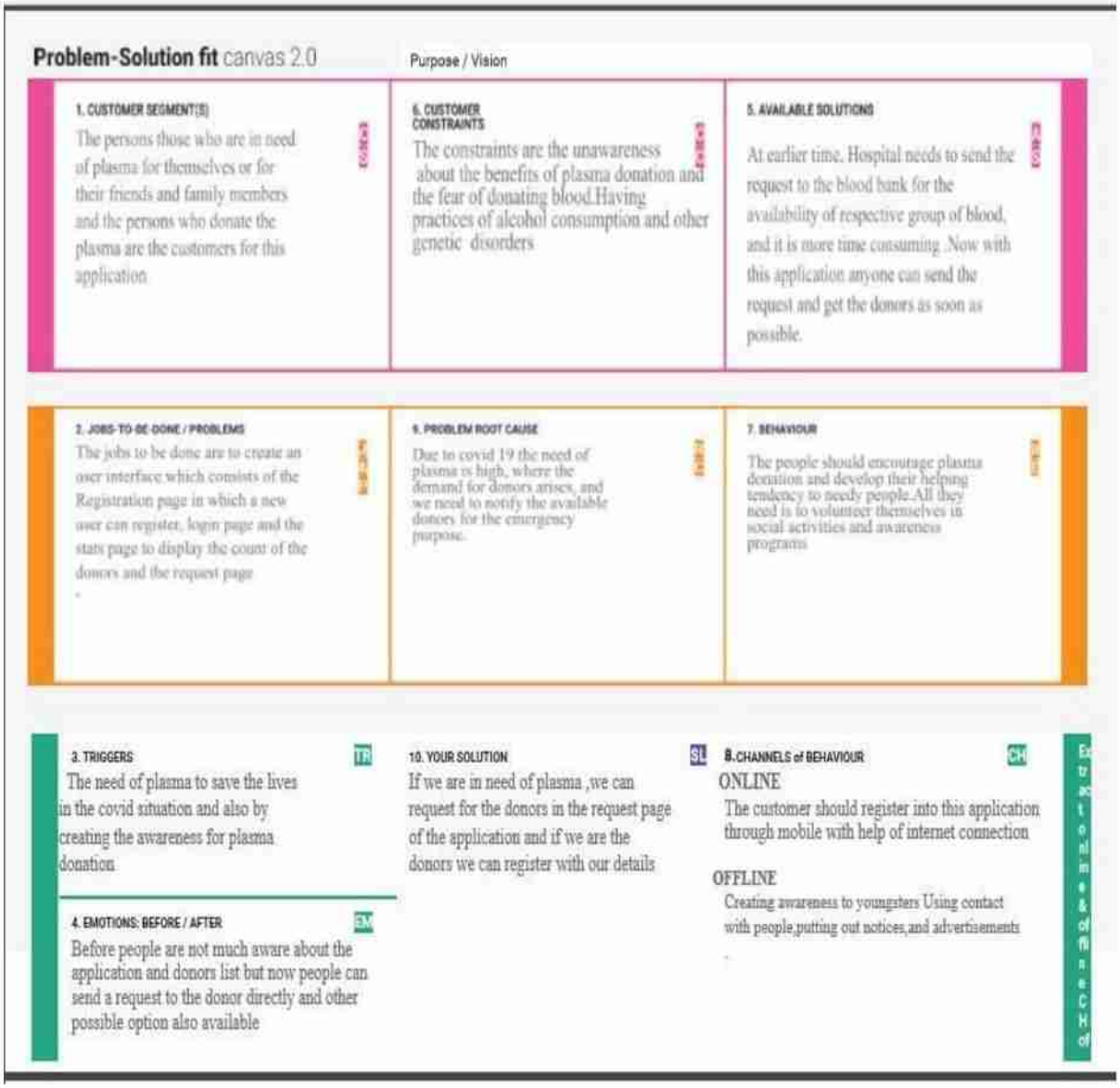
3.2 Ideation & Brainstorming



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	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 is to be built which would take the donor details, store them and inform them upon a request.
2.	Idea / Solution description	Creating web applications with UI to interact with the user for getting the donor details and providing them upon the recipient's request.
3.	Novelty / Uniqueness	It is a reliable platform to connect local donors with patients and helps ensure the best use of your valuable contribution.
4.	Social Impact / Customer Satisfaction	By using this application,we can get respective donors in an emergency situation
5.	Business Model (Revenue Model)	This model doesn't need any cost of expenses and can be easily used .
6.	Scalability of the Solution	This application can handle many numbers of users by either raising a request for plasma donation and requirement.

3.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional requirements.

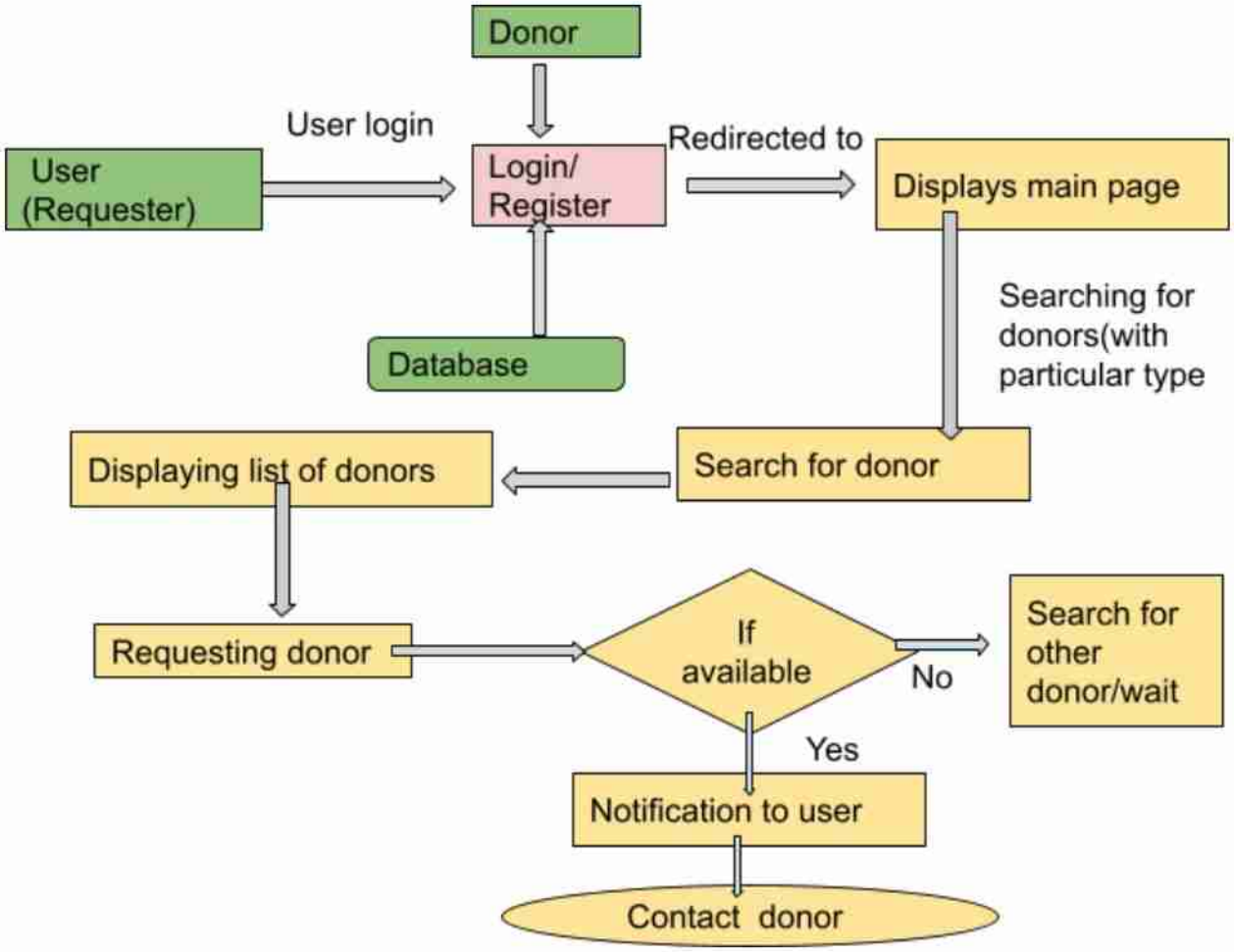
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through website
FR-2	User Confirmation	Confirmation via Email
FR-3	User Login	Login through registered email id
FR-4	Send Request	If plasma required then donor get the notification
FR-5	Contact Donor	Contact donor directly if emergency

4.2 Non-Functional requirements:

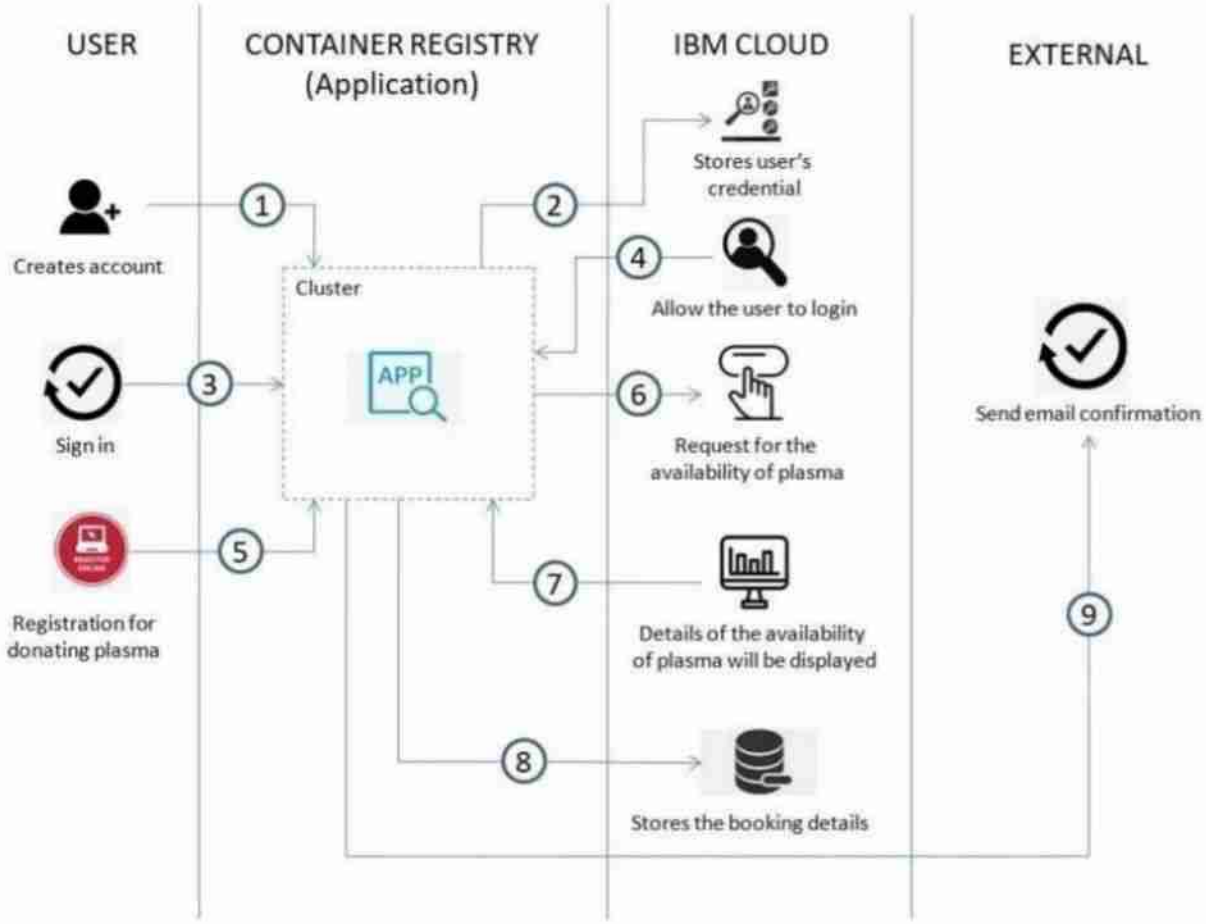
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The plasma Donor application is user friendly and easy to access
NFR-2	**Security**	The users/donor details are stored in the cloud and it is secured with the user email id and password
NFR-3	**Reliability**	The system have the ability to work all the times without failure apart from network failure.The contact list of the donor are provided
NFR-4	**Performance**	The plasma donor application works well in every emergency situation.The easy interactive with the user and less interrupts
NFR-5	**Availability**	The plasma Application is an online web application and it monitor 24/7
NFR-6	**Scalability**	The application offers multiple users and it is designed to protect the users information and details.

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (plasma donor)	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 Gmail		Medium	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password		High	Sprint-1
	Verification	USN-4	As a donor i can verify my donor eligibility criteria	il can check my eligibility	medium	Sprint-2
	Dashboard	USN-6	User can provide their personal details and location	I can complete my donor profile	low	Sprint-2
Customer (Plasma Receiver)	Registration	USN-1	As a receiver, I can register for the application by entering my email /Phone number, password, and confirming my password	I can create receiver account	High	Sprint-1
	Login	USN-2	Registered receiver can log into the application by entering receiver email & password	I can access my account / dashboard	High	Sprint-1
	Verification	USN-3	As a receiver, I can enter my details to check the receiver eligibility criteria	I can check my eligibility to receive plasma	Medium	Sprint-2
Administrat or	login	USN-1	Admin can log into the application by entering email & password	I can access my account / dashboard	High	Sprint-1
	Dashboard	USN-2	Admin can modify, add and remove features from the database and application	I can modify, add and remove features from the database and application	Low	Sprint-3

6.PROJECT PLANNING AND 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	**USER:** I can register for the application by entering my email and password	3	High	Jesila foumiya Jemima blessy Aishwarya P Gayathri
		USN-2	**USER:** I will receive a confirmation email once I have registered for the application	2	High	Jesila foumiya Jemima blessy Aishwarya P Gayathri
	Login	USN-3	**USER:** I can log into the application by entering my email & password	3	High	Jesila foumiya Jemima blessy Aishwarya P Gayathri

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
SPRINT-2	Dashboard	USN-4	**USER:** I Can view the donors and their blood group	5	High	Jesila foumiya Jemima blessy Aishwarya P Gayathri
		USN-5	**USER:** Donor can view if there any need for plasma	5	High	Jesila foumiya Jemima blessy Aishwarya P Gayathri
	Service	USN-6	**USER:** Can view no of available donor	5	High	Jesila foumiya Jemima blessy Aishwarya P Gayathri

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
SPRINT-3	Service	USN-7	**USER:** Request Specific Blood plasma	4	Medium	Jesila foumiya Jemima blessy Aishwarya P Gayathri
		USN-8	**ADMIN:** Notify the donor regarding request	3	Medium	Jesila foumiya Jemima blessy Aishwarya P Gayathri

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
SPRINT-4	Service	USN-9	**ADMIN:** I need to alert the donor when there is a need through SMS or mail	5	High	Jesila foumiya Jemima blessy Aishwarya P Gayathri
		USN-11	**ADMIN:** I need to check donor's medical reports	3	High	Jesila foumiya Jemima blessy Aishwarya P Gayathri
	Data collection	USN-12	**ADMIN:** I need to store user details on the cloud	5	High	Jesila foumiya Jemima blessy Aishwarya P Gayathri
		USN-13	**ADMIN:** I need to collect details about covid-19 cases from verified sources	5	High	Jesila foumiya Jemima blessy Aishwarya P Gayathri

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

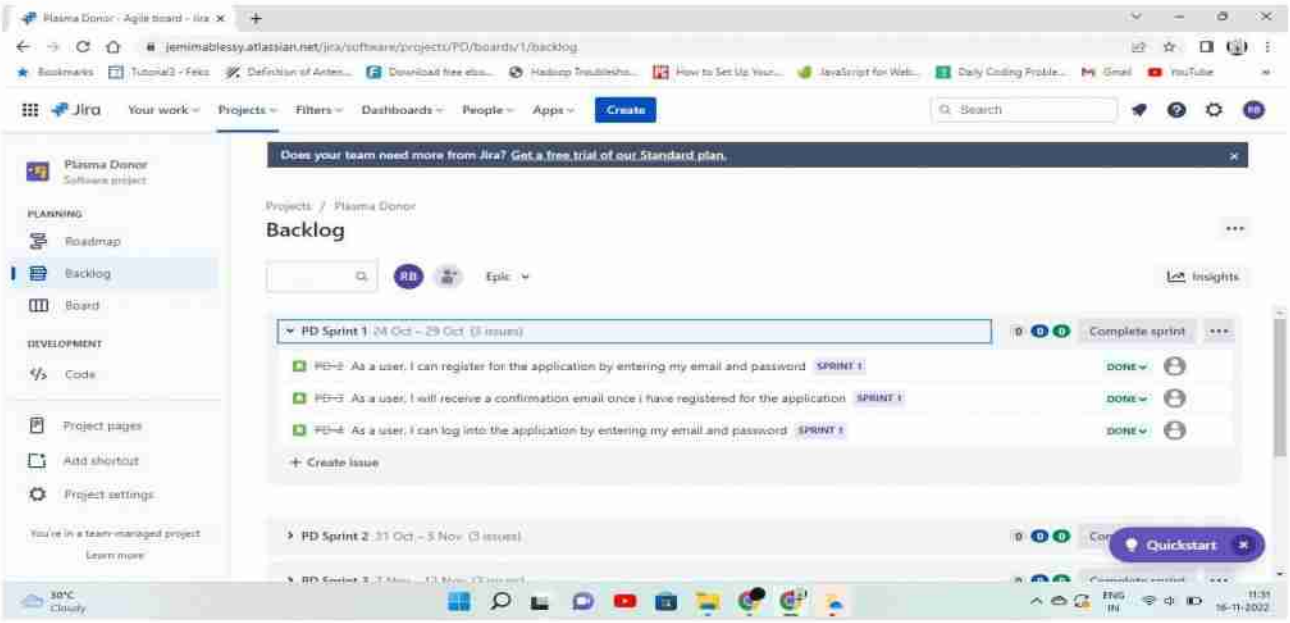
Burndown Chart:

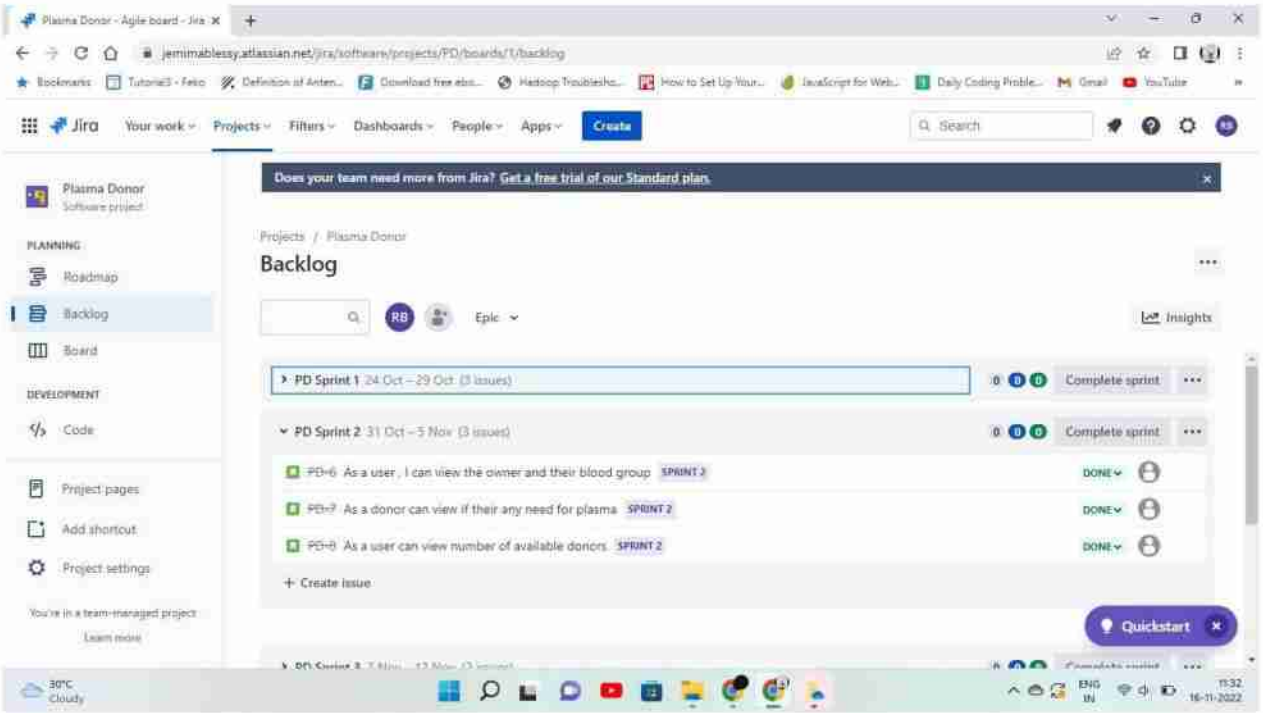
A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn-down charts can be applied to any project containing measurable progress over time.

6.2 Sprint Delivery Schedule

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

6.3 Reports from JIRA

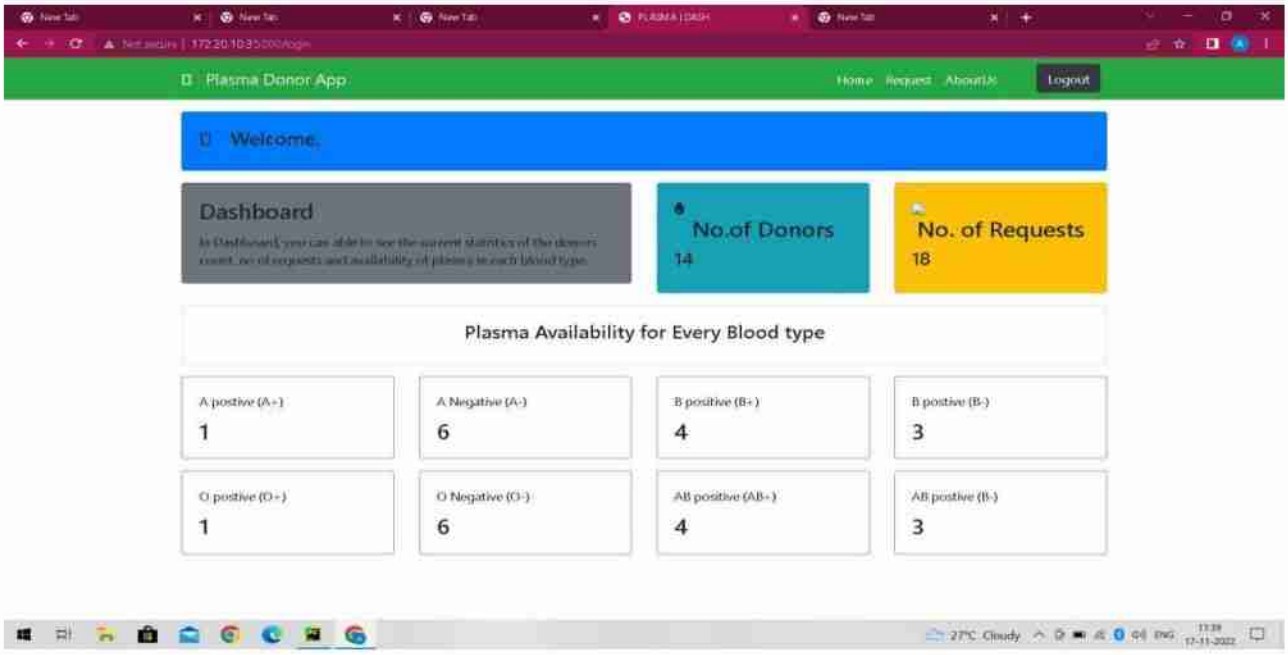




7. CODING & SOLUTIONING

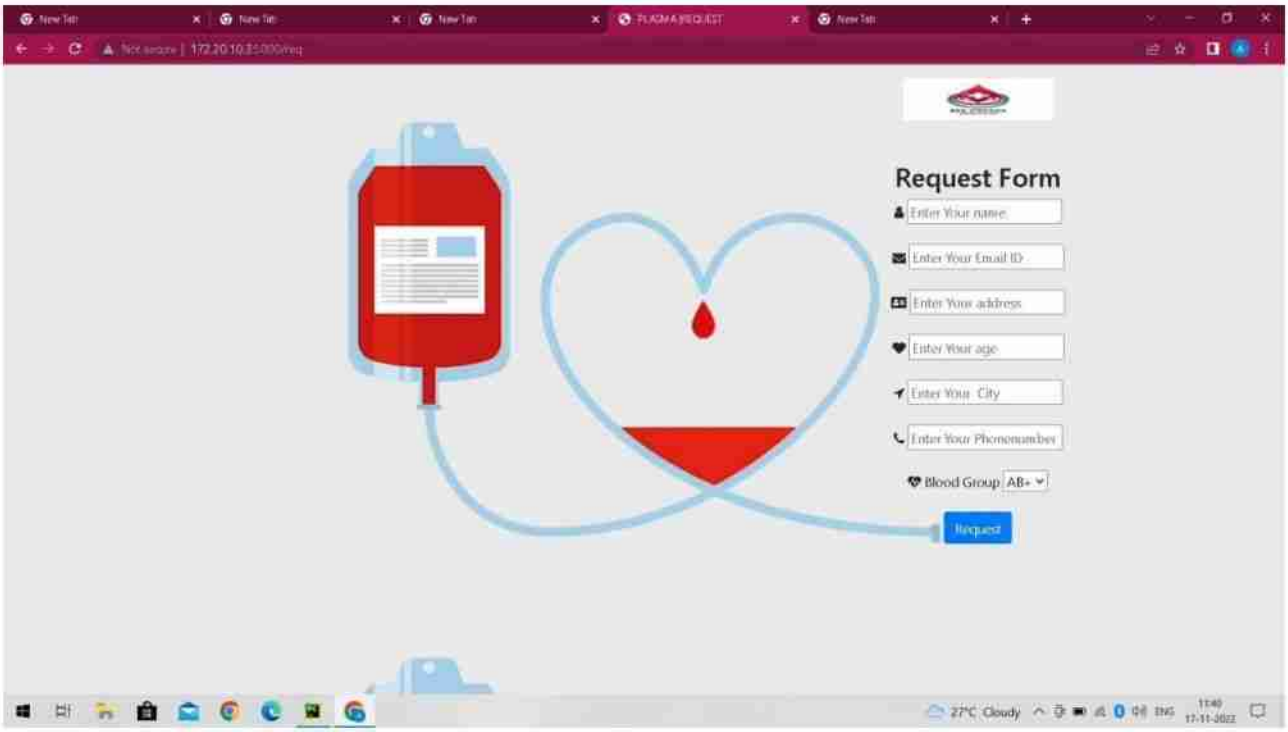
7.1 Feature 1

Plasma Donor application displays the static page for the availability .



7.2 Feature 2

Plasma Donor application is for different types of users including requestor, donor



Request.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>PLASMA |REQUEST</title>
        <!-- favicon -->
        <!-- <link rel="shortcut icon" href="/assets/img/favicon.ico"
type="image/x-icon"> -->
        <!-- <link rel="icon" href="/assets/img/favicon.ico" type="image/x-icon">
-->
        <link rel="icon" type="image/png" sizes="16x16"
href="/assets/img/favicon-32x32.png">
        <!-- bootstrap css cdn -->
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5
t9UJ0Z" crossorigin="anonymous">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.css">
        <!-- css stylesheet -->
        <link rel="stylesheet" href="css/style.css">
        <!-- font styles cdn -->
        <link rel="preconnect" href="https://fonts.gstatic.com">
        <link
href="https://fonts.googleapis.com/css2?family=Alegreya&display=swap"
rel="stylesheet">
        <link
href="https://fonts.googleapis.com/css2?family=Alegreya:wght@600&display=
swap" rel="stylesheet">
        <style>
        body {
          background-image:url('static/imgs/myimage.gif');
          background-repeat:repeat;
          background-size:cover;margin-left: 800px; }
        </style>
```

```
    </head>
<body class="main">
    <!-- bootstrap navbar -->
    <div class="logo mt-3 text-center">
        <a class="main-logo-img mt-5" href="#"><img
src="static/imgs/log2.jpg" alt="sheep-logo" height="50px" width="180px">
            <!-- <a class="navbar-brand" href="index.html">JobPortal</a> -->
          </a>
    </div>
      <!-- navbar ends -->
    <!-- Login form -->
    <div class="login text-center mt-5">
        <h2> Request Form </h2>
        <form action="/req" method="post">
        <div class="msg"><b>{{ msg }}</b></div>
            <!-- <input type="text" placeholder="fullname" id="fullname">
</br></br> -->
            <i class="fa fa-user icon"></i>
            <input type="text" name="name" placeholder="Enter Your
name" id="name" required></br></br>
            <i class="fa fa-envelope icon"></i>
          <input type="email" name="email" placeholder="Enter Your Email
ID" id="email" required></br></br>
            <i class="fa fa-address-card icon"></i>
          <input type="text" name="address" placeholder="Enter Your
address" id="address" required></br></br>
            <i class="fa fa-heart icon"></i>
            <input type="text" name="age" placeholder="Enter Your age"
id="age" required></br></br>
            <i class="fa fa-location-arrow icon"></i>
            <input type="text" name="city" placeholder="Enter Your
City" id="city" required></br></br>
            <i class="fa fa-phone icon"></i>
            <input type="text" name="phone" placeholder="Enter Your
Phonenumber" id="phone" required>

            <br>
            <br>
```

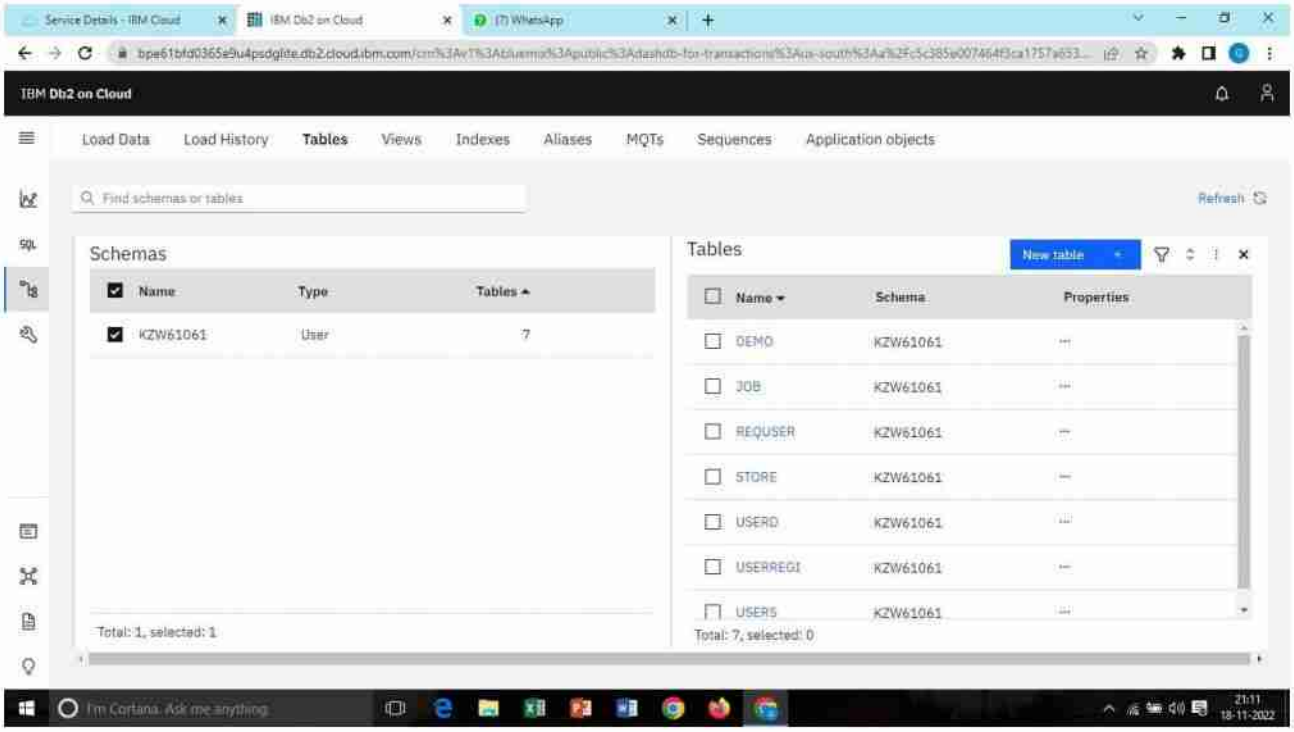
```
            <i class="fa fa-heartbeat icon"></i>
                    Blood Group <select id="blood" name="blood">
                     <option value="AB+">AB+</option>
                     <option value="AB-">AB-</option>
                     <option value="A+">A+</option>
                         <option value="A-">A-</option>
                         <option value="B+">B+</option>
                         <option value="B-">B-</option>
                         <option value="O+">O+</option>
                         <option value="O-">O-</option>
                    </select>

            </br>
            </br>


        <button type="submit" id="button" class="btn btn-primary"> Request
</button>
        </form>
    </div>


</body>
</html>
```

7.3 Database Schema




```
app=Flask(_name_)
app.secret_key='a'



hostname="98538591-7217-4024-b027-
8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain
.cloud" uid="kzw61061"
pwd="8ktYpfXGlIuP5Pp8"
driver="{IBM DB2 ODBC
DRIVER}"
db="bludb"
port="
30875"
protocol="TC
PIP"
certificate="crt.crt"

dsn=(
    " DATABASE={0};"
    " HOSTNAME={1};"
    "PORT={2};"
```

```
    "UID={3};"
    "SECURIT
    Y=SSL;"
    "SSLServerCertificate={4};"
    "PWD={5};").format(db,hostname,port,uid,certificate,pwd)
print(ds
n) try:
    conn = ibm_db.connect(dsn, '', '')
    print("Connected to db") except:
    print("unable
") @app.route('/')

insert_sql ="INSERT INTO userd VALUES(?,?,?,?,?,?,?,?,?)"
prep_stmt =
ibm_db.prepare(conn,insert_sql)
ibm_db.bind_param(prep_stmt,1,usename)
ibm_db.bind_param(prep_stmt,2,email)
ibm_db.bind_param(prep_stmt,3,password
) ibm_db.bind_param(prep_stmt, 4, name)
ibm_db.bind_param(prep_stmt, 5, city)
ibm_db.bind_param(prep_stmt, 6, gender)
ibm_db.bind_param(prep_stmt, 7, phone)
ibm_db.bind_param(prep_stmt, 8, blood)
ibm_db.bind_param(prep_stmt, 9, report)



ibm_db.execute(prep_stmt)
msg='you have successfully registered !'
```

8.TESTING

8.1 Test Cases

Feature type	Component	Test scenario	Test data	Expected result	Actual result	Status
Functional	Welcome page	user is able to see the Login and Signup popup when enters url.	https://plasmadonor.com/	Login and Signup popup should display	Working as expected	Pass
UI	Home page	Verify the UI elements in Login/Signup popup	https://plasmadonor.com/	Application should show below UI elements: a.email text box b.password text c.Login button d.New user?create an account	Working as expected	Pass
Functional	Home page	Verify user is able to log into app with Valid credentials	Username: Aish Email:aish@gmail.com password: Testing123	User should navigate to user account homepage	The user navigated to Home page.	Pass
Functional	Login page	Verify user is able to log into app with InValid credentials	Username: Aish Email:aish@gmail.com password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass
Functional	Login page	Verify user is able to log into app with InValid credentials	Username: Aish Email:aish@gmail.com password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass

7.2 User Acceptance Testing

1.Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [ProductName] project at the time of the release to User Acceptance Testing (UAT).

2.Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

3.Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

9.RESULTS

We show screenshots for this application for different types of users including requester, donor, and administrator. Various features of the application are described and their needs of use are analyzed. If a patient needs a blood at a clinic, blood donors in vicinity can be contacted through using a clinic management service provided in this application. Registered donors will get notification for the blood requests only if their blood group is compatible with the requested blood type and in the same city/region. Then matching blood donors can go to the requesting clinic and donate.

10.ADVANTAGES & DISADVANTAGES

- App automated processes the client had to run, and as a result reduced the amount of tasks and time consumed for arranging the users' donations.
- Client took a big step to get closer to its customers, offering them a solution they have at their fingertips, anytime they need.
- It cannot auto verify user genuineness.
- It requires an active internet connection.

11.CONCLUSION

This application provides a reliable platform to connect local blood donors with patients. It creates a communication channel through authenticated clinics whenever a patient needs blood donation. It is a useful tool to find compatible blood donors who can receive blood request posts in their local area. Clinics can use this web application to maintain the blood donation activity.

12.FUTURE SCOPE

Upgrading the UI that is more user-friendly which will help many users to access the website and also ensures that many plasma donors can be added into the community.

Using elastic load balancer, it helps to handle multiple requests at the same time which will maintain the uptime of the website with negligible downtime.

APPENDIX

Source code

```
from flask import Flask, render_template, request, redirect,url_for,session
import ibm_db
import re app=Flask(_name_) app.secret_key='a'
hostname="98538591-7217-4024-b027-
8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain
.cloud" uid="kzw61061"
pwd="8ktYpfXGlIuP5Pp8"
driver="{IBM DB2 ODBC
DRIVER}"
db="bludb"
port="
30875"
protocol="TC
PIP"
certificate="crt.crt"
dsnDATABASE={
0};"
    " HOSTNAME={1};"
    "PORT={2};"
    "UID={3};"
    "SECURIT
    Y=SSL;"
    "SSLServerCertificate={4};"
    "PWD={5};").format(db,hostname,port,uid,certificate,pwd)
print(ds
n) try:
    conn = ibm_db.connect(dsn, '', '')
    print("Connected to db")
except:
    print("unable ")
@app.route('
```

```
/') def
home():
    return render_template('home.html')
    return render_template('home.html')
@app.route('/login',methods=['GET','POS
T']) def login():
    global
    userid
    msg=''

    if request.method == 'POST':
        usename=request.form['username'] #form in
        html password=request.form['password']
        sql="SELECT * FROM userd WHERE username =? AND
        password=?" stmt=ibm_db.prepare(conn,sql)
        ibm_db.bind_param(stmt,1,usename)
        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 !'

            msg='logge in successfully !'
            return render_template("dashboard.html",msg
        =msg) else:
            msg= 'Incorrect username/ password !'
            return
    render_template('login.html',msg=msg) return
    render_template('login.html')
@app.route('/register',methods
=['GET','POST']) def register():
```

```
msg=''
if request.method =='POST':
    usename = request.form['username'] # form in htmlmail =
    request.form['email'] password = request.form['password']
    name=request.form['name'] city=request.form['city']
    gender=request.form['gender'] phone=request.form['phone']
    blood=request.form['blood'] report=request.form['report']

    sql = "SELECT * FROM userd WHERE
    username =? " stmt = ibm_db.prepare(conn,
    sql) ibm_db.bind_param(stmt, 1, usename)
    ibm_db.execute(stmt)
    account =
    ibm_db.fetch_assoc(stmt)
    print(account)
    if account:
        msg='Account already exist !'
        return render_template("login.html",
    msg=msg) elif not
    re.match(r'[^@]+@[^@]+\.[^@]+',email):
        msg='Invalid emmail address'
    elif not re.match(r'[A-Za-z0-9]+',usename):
        msg='name must contain character and num
        only'

    else:
        insert_sql ="INSERT INTO userd VALUES(?,?,?,?,?,?,?,?,?)"
        prep_stmt =
        ibm_db.prepare(conn,insert_sql)
        ibm_db.bind_param(prep_stmt,1,usename
        ) ibm_db.bind_param(prep_stmt,2,email)
        ibm_db.bind_param(prep_stmt,3,password
        ) ibm_db.bind_param(prep_stmt, 4, name)
        ibm_db.bind_param(prep_stmt, 5, city)
        ibm_db.bind_param(prep_stmt, 6, gender)
        ibm_db.bind_param(prep_stmt, 7, phone)
        ibm_db.bind_param(prep_stmt, 8, blood)
```

```
            ibm_db.bind_param(prep_stmt, 9, report)



            ibm_db.execute(prep_stmt)
            msg='you have successfully registered !'
            return render_template("dashboard.html", msg=msg)

    elif request.method =='post':
        msg ='please fill out details'
    msg = 'you have successfully registered !'
    return render_template('register.html')



@app.route('/dashboar
d') def dash():

    return render_template('dashboard.html')
    @app.route('/index') def index():
    return render_template('index.html')


@app.route('/req',methods =
['GET','POST']) def req():
    msg = ''
    if request.method == 'POST':
        name = request.form['name'] # form in html
        email = request.form['email']
        address=
        request.form['address'] age =
        request.form['age']
        city = request.form['city']
        phone =
        request.form['phone'] blood
        = request.form['blood']
        insert_sql = "INSERT INTO requser
```

```
        VALUES(?,?,?,?,?,?,?)" prep_stmt =
        ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, name)
        ibm_db.bind_param(prep_stmt, 2, email)
        ibm_db.bind_param(prep_stmt, 3, address)
        ibm_db.bind_param(prep_stmt, 4, age)
        ibm_db.bind_param(prep_stmt, 5, city)
        ibm_db.bind_param(prep_stmt, 6, phone)
        ibm_db.bind_param(prep_stmt, 7,
        blood) ibm_db.execute(prep_stmt)
        msg = 'you have successfully registered !'
        return render_template("home.html",
        msg=msg)
    else:
        msg ='please fill out details'
    return
    render_template('req.html')

@app.route('/contactus' ,methods =
['GET','POST']) def contactus():
    return render_template('contactus.html')

@app.route('/aboutus' ,methods =
['GET','POST']) def aboutus():
    return render_template('aboutus.html')
@app.route('/apply',methods =
['GET','POST'])
def
    apply(
    ):
    msg=''
    if request.method =='POST':
        usename = request.form['username'] # form in html
        email = request.form['email']

        qualification =request.form['qualification']
        skills =request.form['skills']
```

```
        jobs = request.form['s']
        sql = "SELECT * FORM users WHERE username
        =?" insert_sql ="INSERT INTO job values(?,?,?,?,?)"
        prep_stmt =ibm_db.prepare(conn,insert_sql)
        ibm_db.bind_param(prep_stmt, 1, usename)
        ibm_db.bind_param(prep_stmt, 2, email)
        ibm_db.bind_param(prep_stmt, 3, qualification)
        ibm_db.bind_param(prep_stmt, 4,
        skills)
        ibm_db.bind_param(prep_stmt,5,jobs)
        ibm_db.execute(prep_stmt)
        msg='You have successfully applied for job !'

    elif request.method =='POST':
        msg='Please fill out the form !'
    return render_template('apply.html'
,msg=msg) @app.route('/display')
def display():
    print(session["username"],session['id'])

    cursor =mysql.connection.cursor()
    cursor.execute('SELECT &FROM job WHERE userid
    =%s',(session['id'],)) account = cursor.fetchone()
    print("account display,account")

    return
render_template('display.html',account=account)
@app.route('/logout')
def logout():
    session.pop('loggedin',None)
    session.pop('id',None)
    session.pop('username',None)
    return
    render_template('home.html')
```

```
if __name__=="__main__":
    app.debug=True
    app.run(host = '0.0.0.0',port=5000)
```

Github link

https://github.com/IBM-EPBL