PLASMA DONOR APPLICATION

BONAFIDE CERTIFICATE

Certified that this Project report, “PLASMA DONOR APPLICATION”
is the bonafide work of

J. PRIYANKA (727619BIT007)
G.K. MOHANYAA (727619BIT013)
J. PRATHYUSHA (727619BIT021)
V.DHARANISRI (727619BIT037)

SYNOPSIS

13. APPENDIX

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

1.1 PROJECT OVERVIEW

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.

In this project, the user can interact with the application. The user can register by providing essential details. The database will have all the details and if a user posts a request then the concerned blood group donors will be notified.

1.2 PURPOSE

1.To develop a system that provides functions to support donors to view and

manage their information conveniently.

2. To maintain records of blood donors, plasma donation information and plasma

stocks in a centralized database system.

3.To inform donors about the availability of plasma in particular blood group.

2.LITERATURE SURVEY

2.1 EXISTING PROBLEM

When a new donor comes to donate blood, they are required to fill out their personal information during the registration process before making a donation. After the donation, the donor is given a donor identification card with their name, blood type and a barcode to be used as a reference for future donations. The barcode is used to retrieve the donor's record containing their personal information, medical history and donation information, including blood results. Only blood bank administrators have the authority to access the donor's records, since the system is only available for their use within the organization. This makes it difficult for donors to make changes to their personal information within the system. That is, for donors to update their personal information, such as their phone number, mailing address, or e-mail, they cannot update the information by themselves, but must contact the blood bank centre to update their information.

At the back the card is a table that contains number of donations, date, location, and the blood collector's signature. Existing donors can submit their donor ID cards to retrieve their personal information and donation records and start the blood donation process, and they will be given a new card after they have donated blood for a total of eight times. Having a donor ID card may be a tangible reminder to people that they are helping lives as a blood donor; however, possessing a physical card comes with drawbacks such as loss or damage. To ensure donors can still identify themselves with the system, other credentials, such as username and password, can be used as a safeguard if their donor ID card is lost or damaged. If the donated blood is disqualified, the donor will be notified through postal mail that their blood component is reactive to viruses, meaning that there is a positive result of the blood being infected, and the organization will also inform the donor to perform another blood test at the blood bank to confirm the result of blood. If the blood is qualified, the administrator then will deposit the blood into the inventory for future requests.

2.2 REFERENCES

1.Voluntary blood donations rising in Oman. (2014,November 21).Retrieved from https://timesofoman.com/article/43536

2.Teena.C. A, Sankar. K and Kannan. S(2014).A Study on Blood Bank Management. https://www.idosi.org/mejsr/mejsr19(8)14/21.pdf

3.Kumar.R, Singh. S and Ragavi. V.A(2017).Blood Bank Management System. http://ijariie.com/AdminUploadPdf/Blood_Bank_Management_System_ijariie6874.pdf

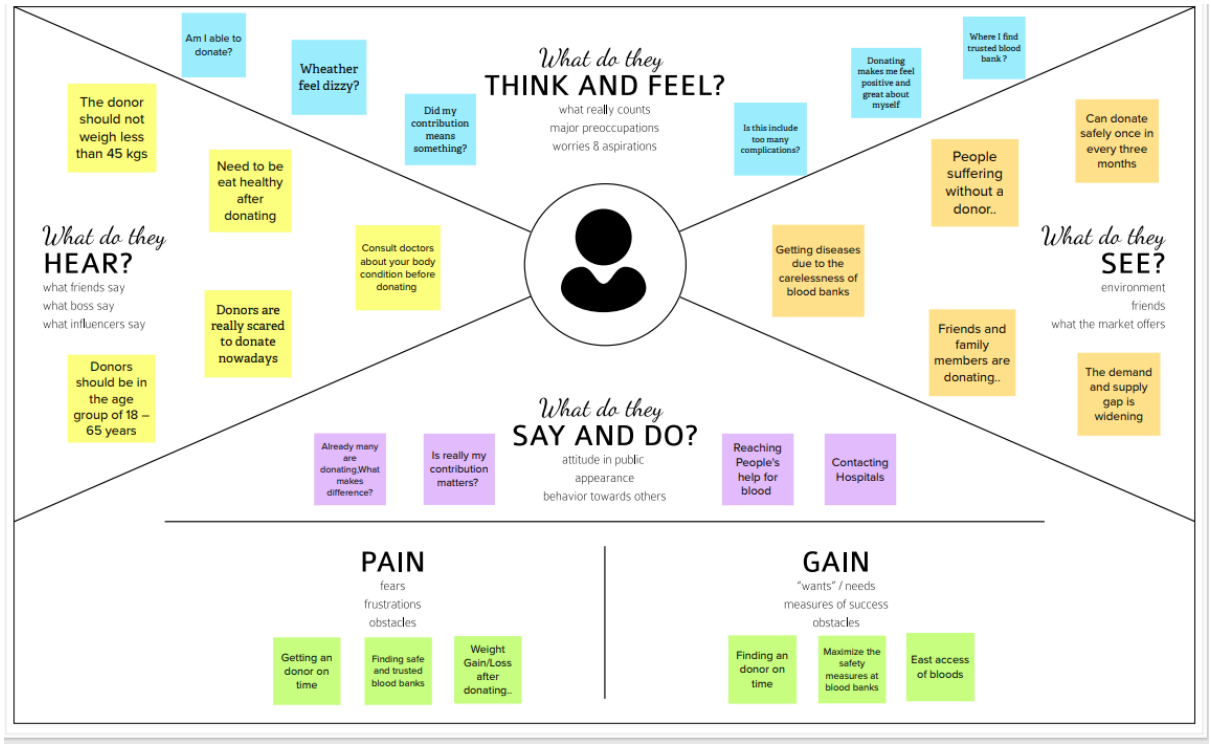
2.3 PROBLEM STATEMENT DEFINITION

In recent times, like blood donation the plasma donation is quite popular and have become essential. Blood plasma donations are used for slightly more specific purposes than a general blood donation. The most common uses of plasma donations include individuals who haveexperienced a severe trauma, burn or shock, adults or children with cancer, and people with liver or clotting factor disorders.

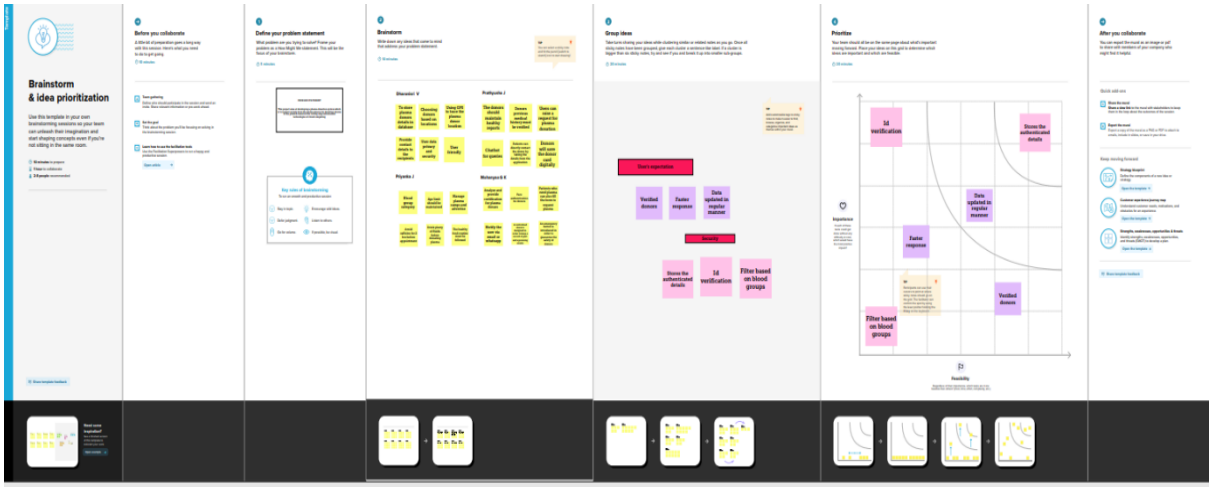
The plasma donation process usually consumes a lot of time and effort from both donors and medical staff since there is no concrete information system that allows donors and plasma donation centers communicate efficiently and coordinate with each other to minimize time and effort required for plasma donation process. This work aims at developing a Plasma Donation System based on the cutting-edge information technologies of cloud computing.

3. IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION AND BRAINSTORMING

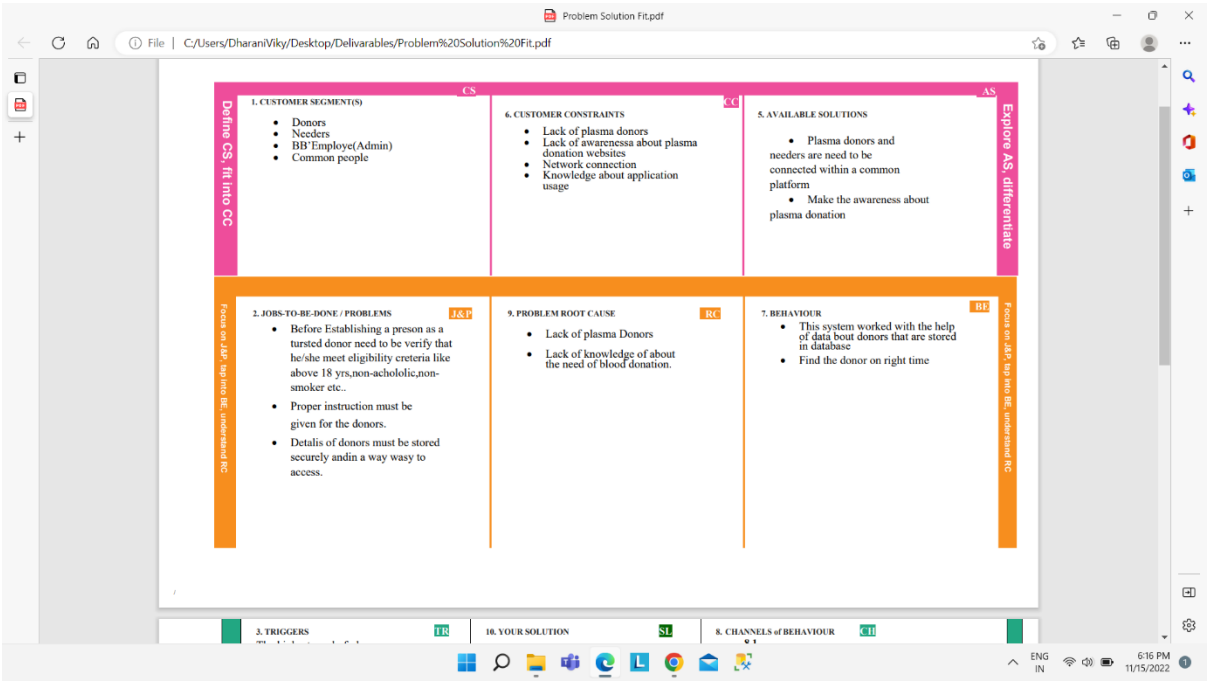


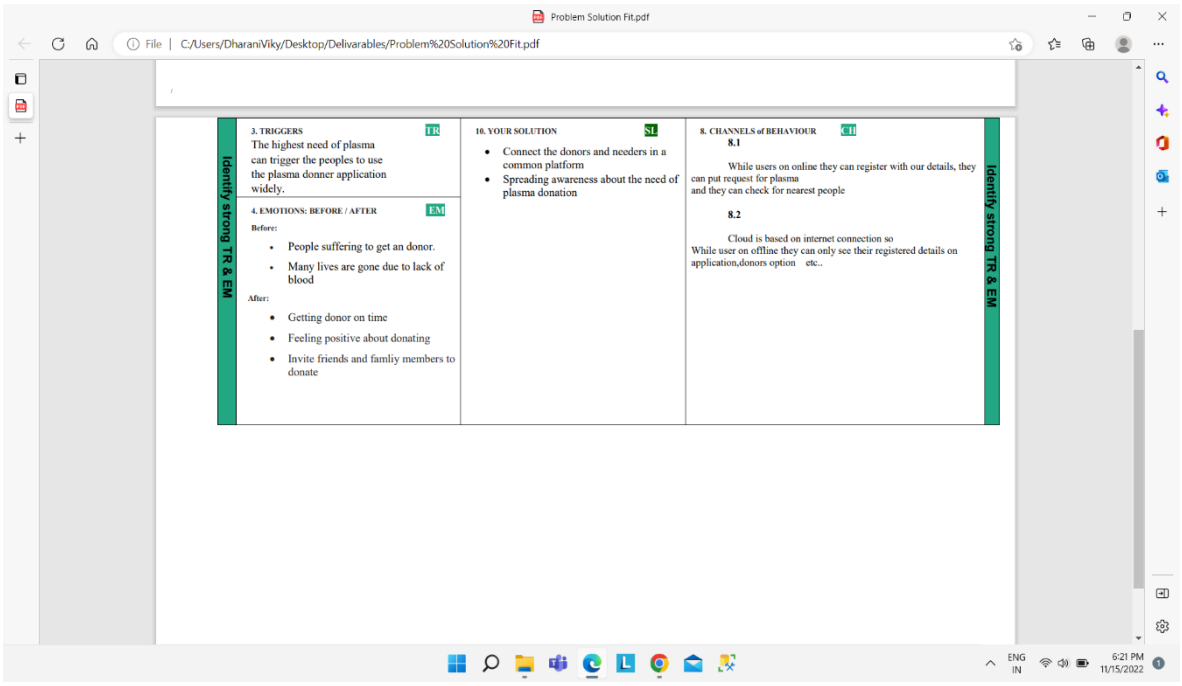
3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• When the details are maintained manually,it is complicated for donors and patients. • Making such an application which is user friendly as well as has more features for serving the people better, we are proposing a model in which we are going to connect the plasma donor and requester together in a unique way
2.	Idea / Solution description	• Both the donor and patients can register • Statistics on overall donation will be displayed • By using GPS easily track the plasma donor • Reduced workload by storing the details in cloud storage
3.	Novelty / Uniqueness	• User friendly UI to access the web application by all the people • If a user doesn't have a matching plasma based on their blood type, they can send a request for it. • The app will automatically scan the available database of users registered as donor to find a suitable match. • If a successful match is found then a chat box between the donor and recipient is established. • Else the request stays in place in the database until a suitable voluntary donor is found in future. • Voluntary donors can fill out an application form and make an appointment for plasma donation. • Once they have finished their donation, they will be given their e-certification for plasma donation.

4.	Social Impact / Customer Satisfaction	• Find the donors in near places • Connect the donors and patients easily • Chatbot for queries • With all of the authenticated information, this platform will assist the public in donating or obtaining their plasma needs.
5.	Business Model (Revenue Model)	People will get used to this application, by collaborating with government and organizing blood donation camps
6.	Scalability of the Solution	• The aim is to build a Lifesaver E-Plasma Donation App using Cloud with advanced features that will help to overcome the barrier between plasma bank, plasma donor and patient • Since the project uses IBM DB2 database it can handle with multiple requests in various regions • As this is an web application and uses cloud storage ,any further enhancements in technology can be incorporated within this application.

3.4 PROBLEM SOLUTION FIT




Problem Solution Fit.pdf
File | C:/Users/DharaniViky/Desktop/Delivarables/Problem%20Solution%20Fit.pdf
3. TRIGGERS
TR
The highest need of plasma can trigger the peoples to use the plasma donner application widely.
4. EMOTIONS: BEFORE / AFTER
EM
Before:
People suffering to get an donor.
Many lives are gone due to lack of blood
After:
Getting donor on time
Feeling positive about donating
Invite friends and famliy members to donate
10. YOUR SOLUTION
SL
Connect the donors and needers in a common platform
Spreading awareness about the need of plasma donation
8. CHANNELS of BEHAVIOUR
CH
8.1
While users on online they can register with our details, they can put request for plasma and they can check for nearest people
8.2
Cloud is based on internet connection so While user on offline they can only see their registered details on application,donors option etc..
Identify strong TR & EM
Identify strong TR & EM
ENG IN
6:21 PM 11/15/2022

4.REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form(WebApp) Registration through Gmail
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login using Registered email Id
FR-4	Searching/reporting requirements	Users can use the search bar to look up information about camps and other topics.
FR-5	User Plasma Request	Users can request to donate plasma by filling out the request form on the page. Once the request is submitted, they will notified through email
FR-6	Statistical data	The availability of plasma is given in the page as stats, which will be helpful for the users.
FR-7	Certification	After the donor donates plasma, we will give them a digital certificate of appreciation and authentication.
FR-8	View donation camps	View the list of donation camps happening nearby.
FR-9	Virtual Assistants(ChatBot)	A virtual assistant is a software agent that can carry out tasks or provide services on behalf of a person in response to commands or inquiries. When users enter their inquiries, the system will respond with pertinent information about plasma and details of plasma donation.

4.2 Non-Functional requirements:

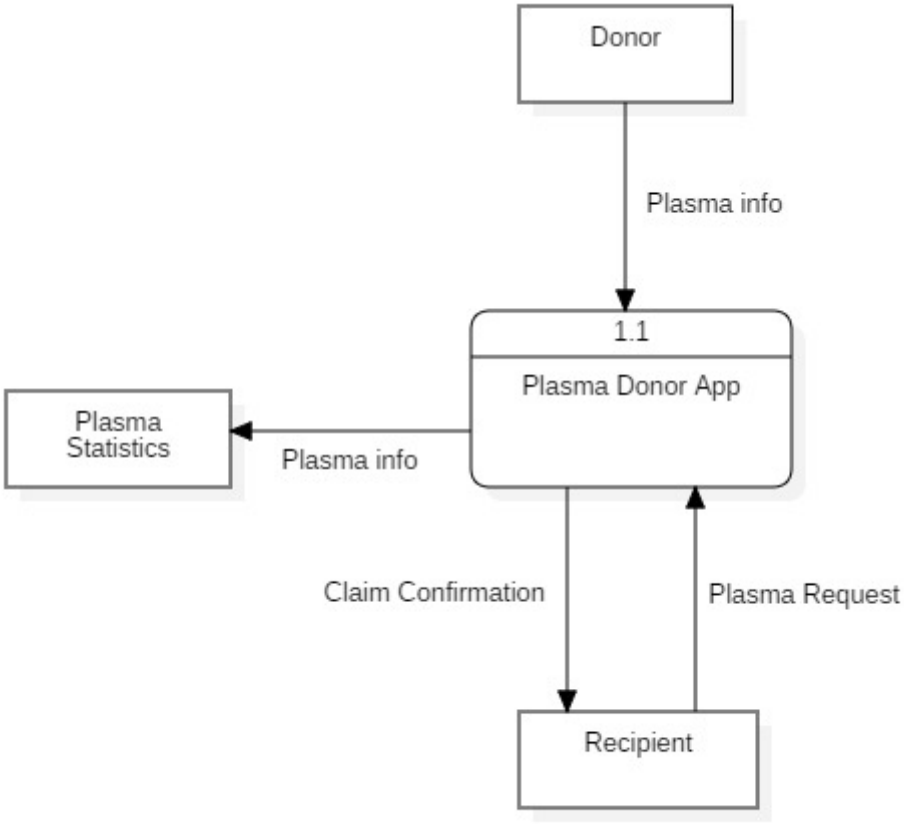
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The user interface of the plasma donor system must be well-designed and welcoming.
NFR-2	**Security**	Data storage is required to have high security systems, just like it is by many other applications. Databases are able to keep all the donor information that is viewed by applications. It must be secured with email Id and password.
NFR-3	**Reliability**	The system has the ability to work all the times without failures apart from network failure. A donor can have the faith on the system. The authorities will keeps the privacy of all donors in a proper manner
NFR-4	**Performance**	The Plasma donor System must perform well in different scenarios. The system is interactive and delays involved are less.
NFR-5	**Availability**	The system including the online and offline components should be available 24/7.
NFR-6	**Scalability**	The application should have the ability to handle growing numbers of users and load without compromising on performance and causing disruptions to user experience. The system offers the proper resources for issue solutions and is designed to protect sensitive information during all phases of operation

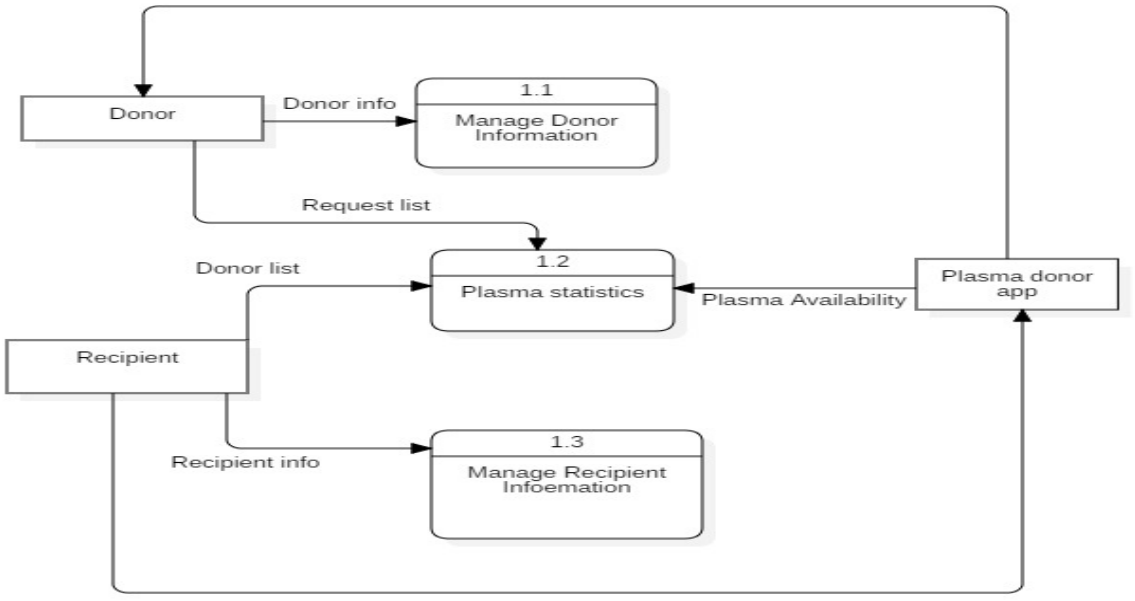
5.PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

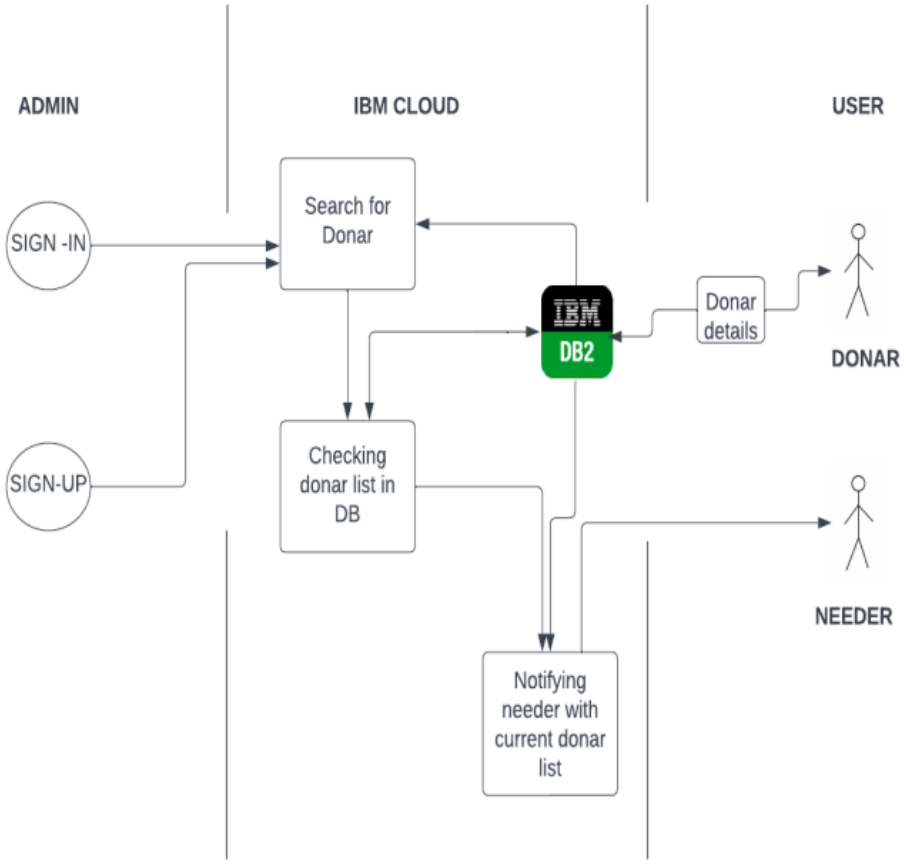
Level 0



Level 1



5.2 SOLUTION AND TECHNOLOGY ARCHITECTURE



COMPONENTS AND TECHNOLOGIES

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Logic for a process in the application	Java / Python
3.	Application Logic-2	Logic for a process in the application	IBM Watson STT service

4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL, NoSQL, etc.
6.	Cloud Database	Database Service on Cloud	IBM DB2, IBM Cloudant etc.
7.	File Storage	File storage requirements	IBM Block Storage or Other Storage Service or Local Filesystem
8.	External API-1	Purpose of External API used in the application	IBM Weather API, etc.
9.	External API-2	Purpose of External API used in the application	Aadhar API, etc.
10.	Machine Learning Model	Purpose of Machine Learning Model	Object Recognition Model, etc.
11.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

APPLICATION CHARACTERISTICS

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Technology of Opensource framework
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	e.g. SHA-256, Encryptions, IAM Controls, OWASP etc.

5.3 USER STORIES

User Type	Functional requirement	User Story Number	User story/task	Acceptance Criteria	Priority	Release
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
	Login	USN-2	As a user, I can log into the application by entering email & password	I can receive confirmation email & click confirm	High	Sprint-2

	Donation list	USN-3	As a user, I can log into the application and see the request and also receive request.	I can see the requests and accept or reject it.	High	Sprint-3
Recipient	App Registration	USN-4	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
	Login	USN-5	As a user, I can log into the application by entering email & password	I can receive confirmation email & click confirm	High	Sprint-2
	Plasma Request	USN-6	As a user, I can enter into the application and find the donor and request for plasma.	I can register & access the dashboard with Login and request plasma.	Medium	Sprint-3
	Find donor	USN-7	As a patient, I can directly access the application and find the plasma donor	I can access my account / dashboard	High	Sprint-3

User Type	**Functional Requirement**	**User Story Number**	**User story/Task**	**Acceptance criteria**	**Priority**	**Release**
Admin	Login	USN-8	As Administrator, I can login into the app.	I can access the app details	High	Sprint-1
	Maintain database	USN-9	As Administrator I can hold the exact details of donor and recipient and availability of plasma.	I can access database	Medium	Sprint-4
Bot	Help the user	USN-10	As AI bot, i can hold the good communication between bank and user also help the user	I can access the dashboard	Medium	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

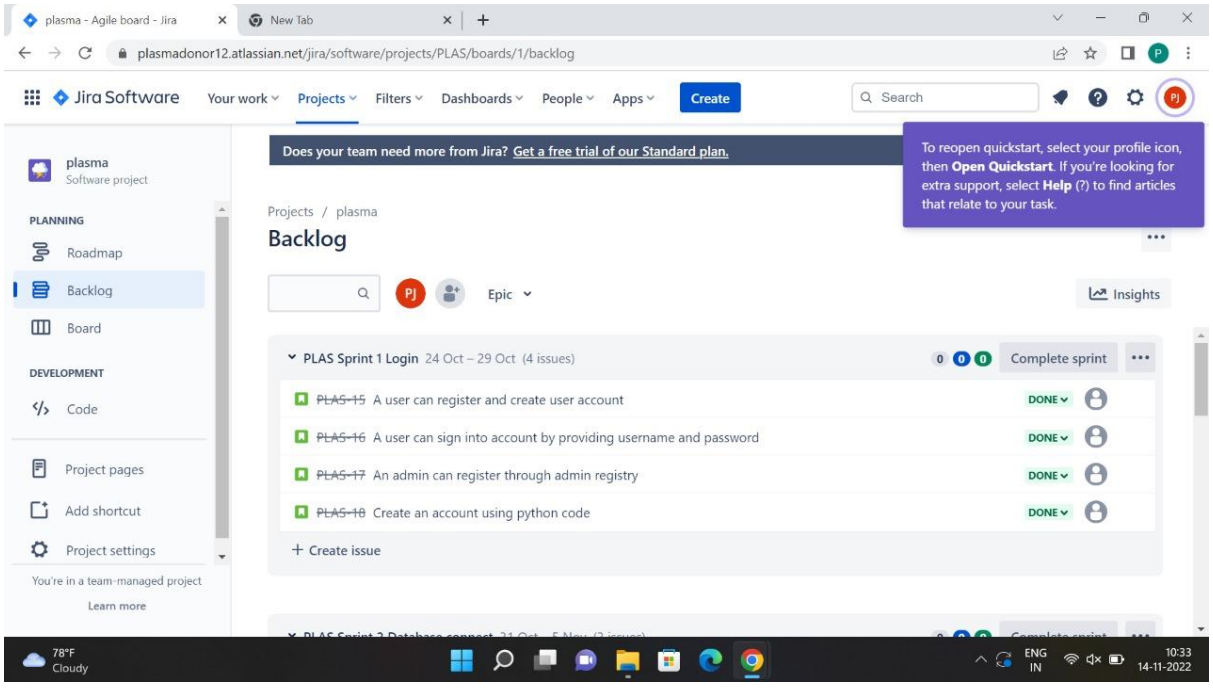
Sprint	Functional requirement	User Story Number	User Story/ Task	Story Points	Priority	Team member
Sprint 1	Registration	USN-1	A User can register and create the user account.	6	High	Dharani Sri
Sprint 1	Login	USN-2	A user can sign into the application by entering correct username and password	6	High	Dharani Sri
Sprint 1	Admin register	USN-3	An admin can register through admin registry	4	Medium	Dharani Sri
Sprint 1	Register Admin via Script	USN-4	Creating an admin account using python script. As for security reasons we should implement separate python script.	4	Medium	Dharani Sri
Sprint 2	Implementing Authentication System	USN-5	Creating an authentication system for both admin and users	8	High	Priyanka ,Mohanya a

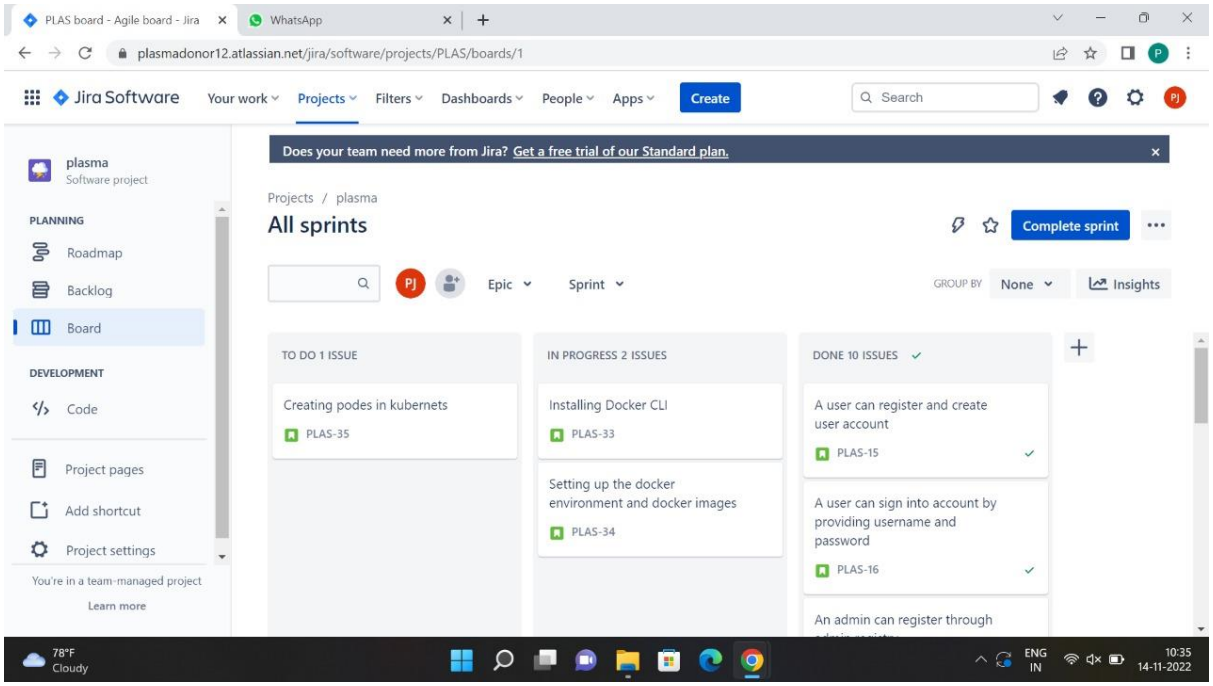
Sprint	Functional Requirement	User Story Number	User story/Task	Story Point	Priority	Team members
Sprint 2	Creating tables	USN-6	Creating db2 account and creating tables in Db2	6	Medium	Prathyusha, Mohanyaa
Sprint 2	Creating SSL certificate	USN-7	Creating SSL certificate to connect to db2 via python	6	High	Priyanka, Prathyusha
Sprint 3	Plasma request and donor acknowledge feature	USN-8	Admin can view plasma requests and approve the requests as per the requirements	8	High	DharaniSri,Priyanka
Sprint 3	Creating dashboard for admin	USN-9	Admin can view the total plasma requests received and approve it.	6	High	Prathyusha, Mohanyaa
Sprint 3	Integration with SendGrid	USN-10	The verification mail for donor and patient	6	Medium	DharaniSri,Prathyusha
Sprint-4	Docker installation	USN-11	Installing Docker CLI	4	Medium	Priyanka, Mohanyaa
Sprint 4	Creating docker image	USN-12	Setting up the docker environment and creating the images	8	High	Prathyusha, Mohanyaa
Sprint 4	Kubernetes	USN-13	Creating podes in Kubernetes and uploading it in IBM cloud	8	Medium	Priyanka, Mohanyaa

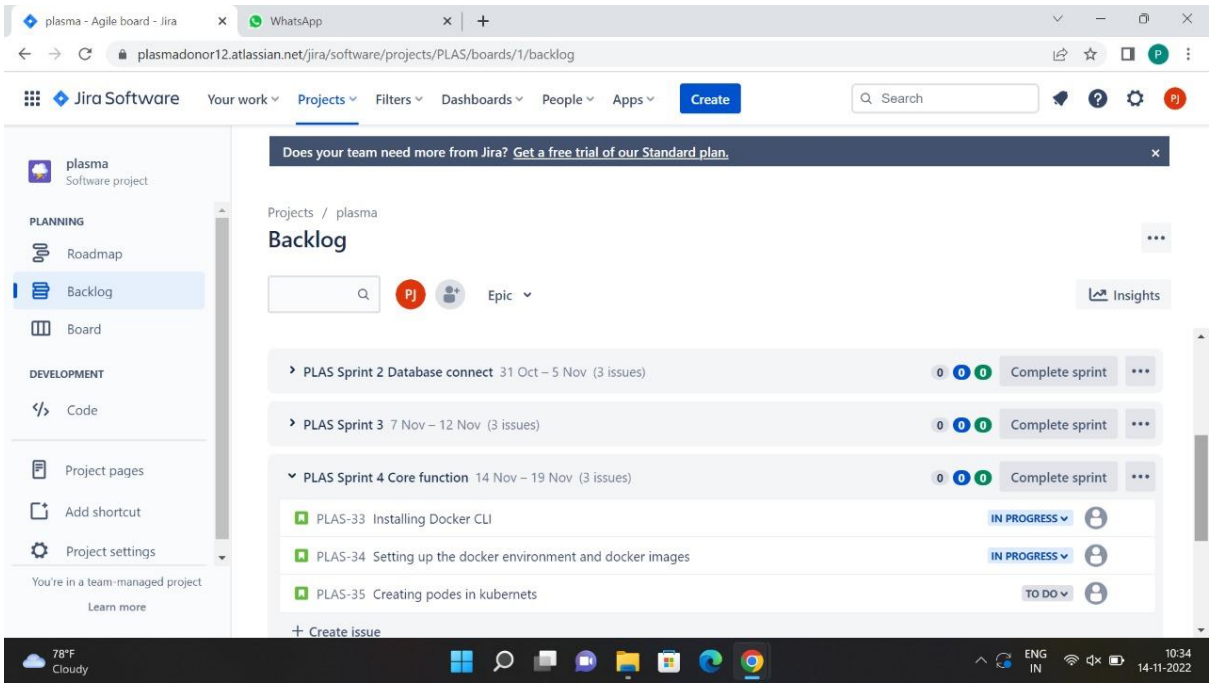
6.2 SPRINT DELIVERY SCHEDULE

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




PLAS board - Agile board - Jira
WhatsApp
plasmadonor12.atlassian.net/jira/software/projects/PLAS/boards/1
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
plasma
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Does your team need more from Jira? Get a free trial of our Standard plan.
Projects / plasma
All sprints
Complete sprint
Epic
Sprint
GROUP BY
None
Insights
TO DO 1 ISSUE
Creating podes in kubernets
PLAS-35
IN PROGRESS 2 ISSUES
Installing Docker CLI
PLAS-33
Setting up the docker environment and docker images
PLAS-34
DONE 10 ISSUES
A user can register and create user account
PLAS-15
A user can sign into account by providing username and password
PLAS-16
An admin can register through
78°F
Cloudy
ENG
IN
10:35
14-11-2022


plasma - Agile board - Jira
WhatsApp
plasmadonor12.atlassian.net/jira/software/projects/PLAS/boards/1/backlog
Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
plasma
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
You're in a team-managed project
Learn more
Does your team need more from Jira? Get a free trial of our Standard plan.
Projects / plasma
Backlog
Epic
Insights
PLAS Sprint 2 Database connect 31 Oct – 5 Nov (3 issues)
Complete sprint
PLAS Sprint 3 7 Nov – 12 Nov (3 issues)
Complete sprint
PLAS Sprint 4 Core function 14 Nov – 19 Nov (3 issues)
Complete sprint
PLAS-33 Installing Docker CLI
IN PROGRESS
PLAS-34 Setting up the docker environment and docker images
IN PROGRESS
PLAS-35 Creating podes in kubernets
TO DO
Create issue
78°F
Cloudy
ENG
IN
10:34
14-11-2022

7.CODING AND SOLUTIONING

7.1 FEATURE 1

```
from turtle import st
from flask import Flask, render_template, request, redirect, url_for, session
from markupsafe import escape
import sendgrid
import os
import sys
import ibm_db
from flask_mail import Mail, Message
from emailSender import *
from flask import Response


conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=824dfd4d-99de-440d-9991-
629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=30119;Security=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=qdz26030;PWD=hC55ak4dG6UPcHuX;","","")
app = Flask(__name__)


#Email
app.config['SECRET_KEY'] = 'top-secret!'
app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
```

```
app.config['MAIL_PASSWORD'] = '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'
app.config['MAIL_DEFAULT_SENDER'] = 'ZrPlasmaDonor@outlook.com'
mail = Mail(app)



@app.route('/')
def home():
  return render_template('home.html')
@app.route('/log')
def log():
  return render_template('login.html')
@app.route('/signup')
def signup():
  return render_template('register.html')
@app.route('/contact')
def contact():
  return render_template('contact.html')
@app.route('/donorpage')
def donorpage():
  return render_template('donor.html')
@app.route('/eligi')
def eligi():
  return render_template('eligibility.html')
@app.route('/req')
```

```
def req():
  return render_template('requester.html')


@app.route('/addrec',methods = ['POST', 'GET'])
def addrec():
  if request.method == 'POST':
    name = request.form['name']
    email = request.form['email']
    phnum = request.form['phnum']
    phnum2=request.form['phnum2']
    pas = request.form['pas']
    pas2=request.form['pas2']
    gen=request.form['gen']


    sql = "SELECT * FROM user WHERE name =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,name)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)


    if account:
        return render_template('home.html', msg="You are already a member, please login using your details")
    else:
      insert_sql = "INSERT INTO user 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, phnum)
    ibm_db.bind_param(prep_stmt, 4, phnum2)
    ibm_db.bind_param(prep_stmt, 5, pas)
    ibm_db.bind_param(prep_stmt, 6, pas2)
    ibm_db.bind_param(prep_stmt, 7, gen)
    ibm_db.execute(prep_stmt)

  return render_template('home.html', msg="Student Data saved successfuly.")

@app.route('/loginpage',methods=['POST'])

def loginpage():
  user = request.form['user']
  passw = request.form['passw']
  sql = "SELECT * FROM user WHERE email =? AND pas=?"
  stmt = ibm_db.prepare(conn, sql)
  ibm_db.bind_param(stmt,1,user)
  ibm_db.bind_param(stmt,2,passw)
  ibm_db.execute(stmt)
  account = ibm_db.fetch_assoc(stmt)
  if account:
      return render_template('home.html')
  else:
      return render_template('login.html', pred="Login unsuccessful. Incorrect username / password !")
```

```
@app.route('/donor',methods = ['POST', 'GET'])
def donor():
  if request.method == 'POST':

    name = request.form['name']
    email = request.form['email']
    phnum = request.form['phnum']
    phnum2=request.form['phnum2']
    blood=request.form['bloodgrp']
    states=request.form['state']
    district=request.form['district']
    address=request.form['address']



    sql = "SELECT * FROM donor WHERE name =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,name)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)

    if account:

      return render_template('home.html', msg="You are already a member as donor!!")
```

```
    else:
      insert_sql = "INSERT INTO donor 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, phnum)
      ibm_db.bind_param(prep_stmt, 4, phnum2)
      ibm_db.bind_param(prep_stmt, 5, blood)
      ibm_db.bind_param(prep_stmt, 6, states)
      ibm_db.bind_param(prep_stmt, 7, district)
      ibm_db.bind_param(prep_stmt, 8, address)



      ibm_db.execute(prep_stmt)


        return render_template('home.html', msg="Successfuly registered as donor.")



@app.route('/requested',methods = ['POST', 'GET'])
def requested():
  if request.method == 'POST':


    name = request.form['name']
    lname=request.form['lname']
    email = request.form['email']
```

```
    phnumr = request.form['phnumr']
    phnumr2=request.form['phnumr2']
    address=request.form['address']
    bloodgrp=request.form['blood']
    sql = "SELECT * FROM requester WHERE name =?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,name)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)

    if account:

      # return render_template('home.html', msg="You are already a member as requester!!")
      pass
    else:

      insert_sql = "INSERT INTO requester VALUES (?,?,?,?,?,?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, name)
      ibm_db.bind_param(prep_stmt, 2, lname)
      ibm_db.bind_param(prep_stmt, 3, email)
      ibm_db.bind_param(prep_stmt, 4, phnumr)
      ibm_db.bind_param(prep_stmt, 5, phnumr2)
      ibm_db.bind_param(prep_stmt, 6, address)
      ibm_db.bind_param(prep_stmt, 7, bloodgrp)
      ibm_db.execute(prep_stmt)
```

```
bloodgrp = request.form['blood']
state = request.form['state']
district = request.form['district']
ph = request.form['phnumr']

reqDetails = dict()
reqDetails['Name'] = name + " " + lname
reqDetails['Email'] = email
reqDetails['State'] =  state
reqDetails['City'] =  district
reqDetails['bloodgrp'] = bloodgrp
reqDetails['phone'] = ph

sql = "SELECT * FROM donor WHERE blood=? and states=? and district=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt,1,bloodgrp)
ibm_db.bind_param(stmt,2,state)
ibm_db.bind_param(stmt,3,district)
ibm_db.execute(stmt)
data = ibm_db.fetch_assoc(stmt)
donorFoundFlag = False
donorList = []
if data!= False:
  while data != False:
    donorList.append(data)
    data = ibm_db.fetch_assoc(stmt)
```

```
        sendEmail(email, successMail(reqDetails,donorList))  #Send an Email to
Requestor
      for i in donorList:
        mailTemplate = sendEmailToDoanar(i['NAME'],  reqDetails)
        sendEmail(i['EMAIL'], mailTemplate) #Send an Email to Donor



    else:
      #When no donor found
      pass
  return ('', 204)
if __name__ == '__main__':
  app.run(debug=True)


def sendEmail(email, data):
  recipient = email
  msg = Message('Plasma Donar', recipients=[recipient])
  msg.body = ('')
  msg.html = data
  mail.send(msg)
```

7.2 FEATURE 2

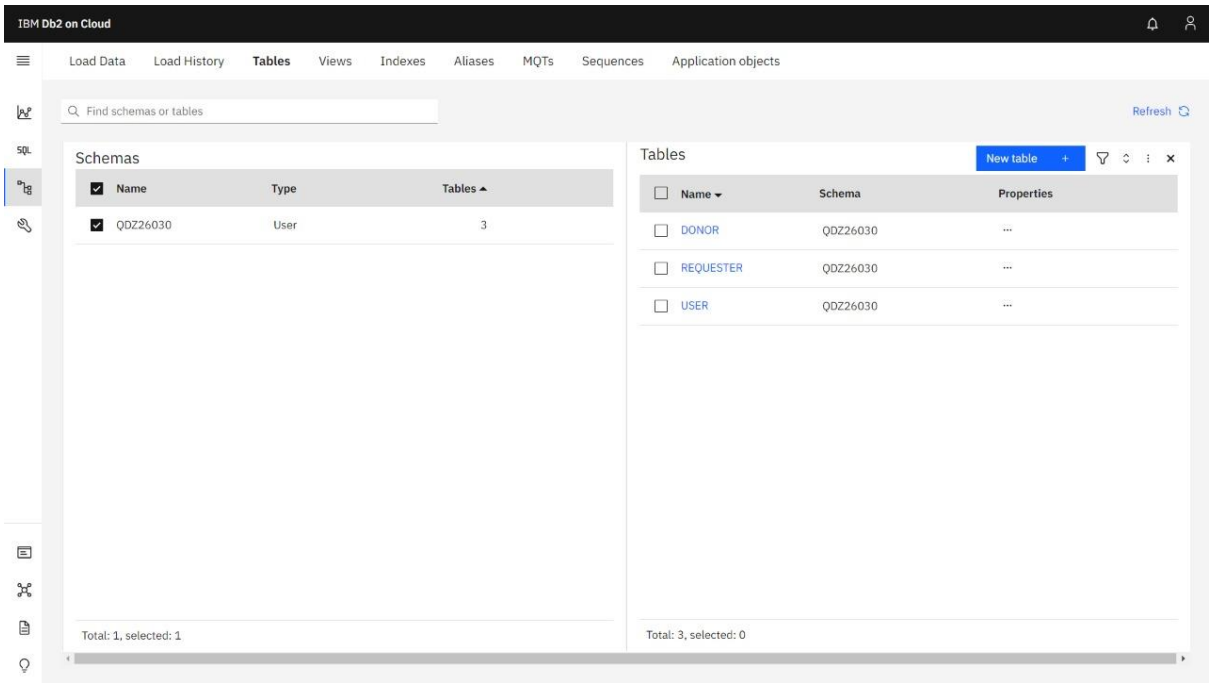
```
import ibm_db
conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=824dfd4d-99de-440d-
9991-
629c01b3832d.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=3011
```

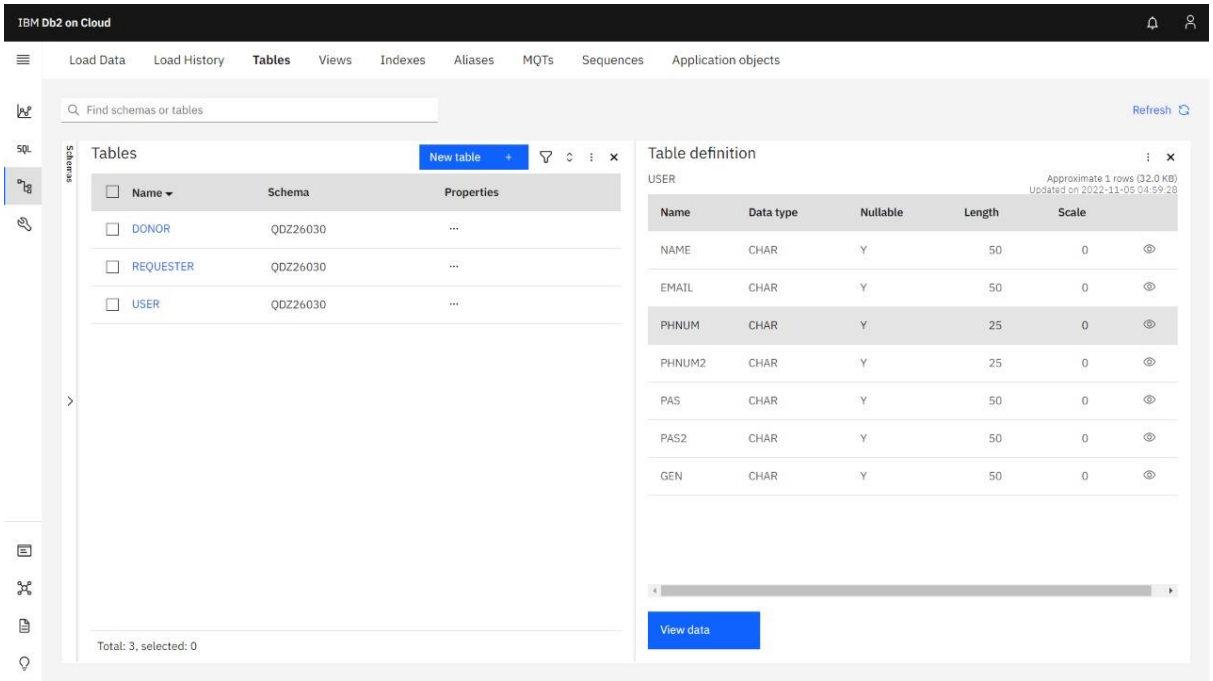
9;Security=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=qdz26030;PWD=hC55ak4dG6UPcHuX;","","")

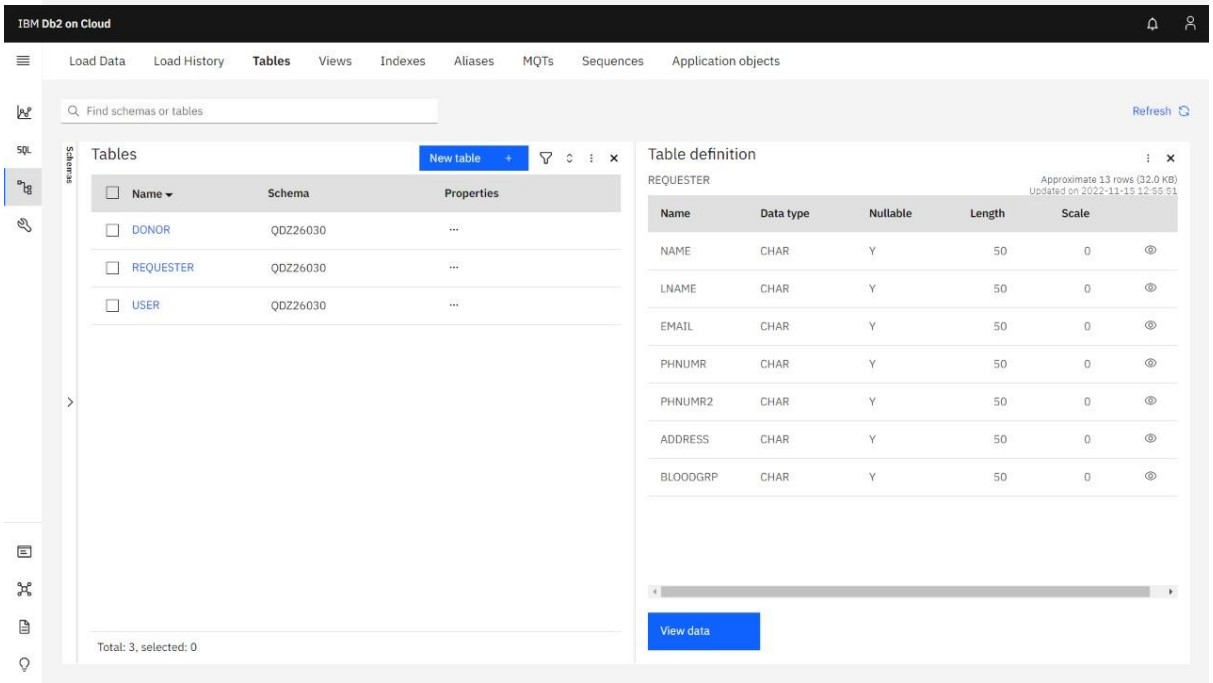
#To retrive all the records from DB2

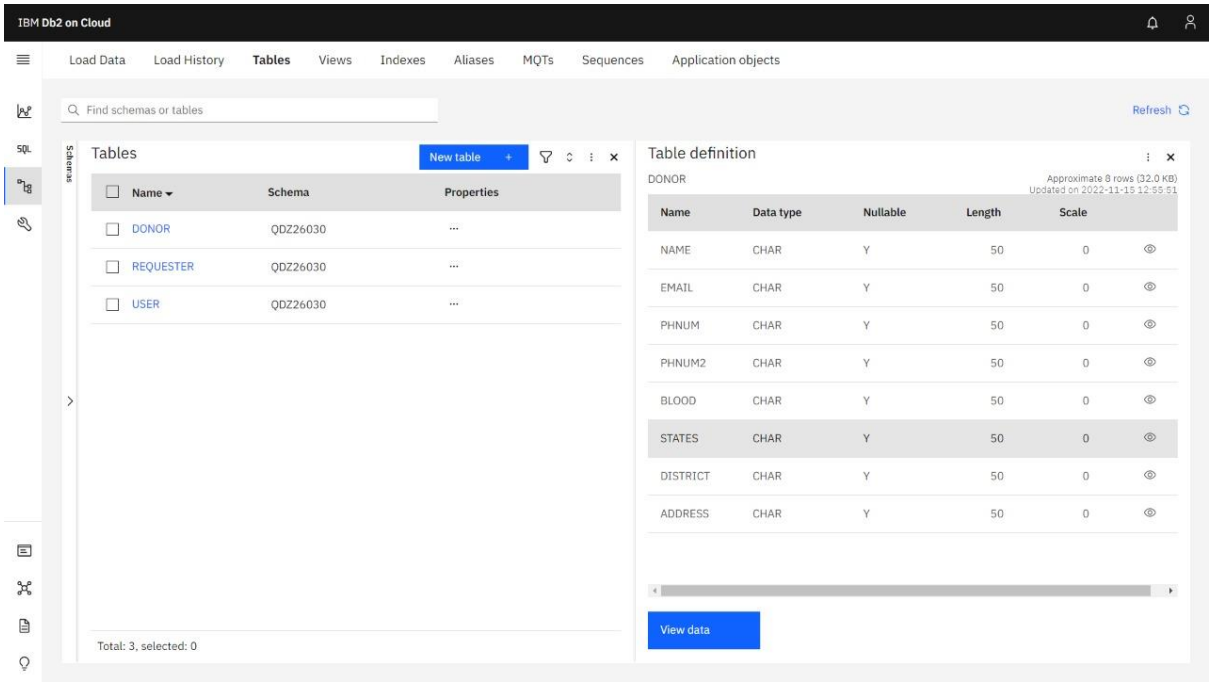
```
sql = "SELECT * FROM DONOR"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
values=dictionary
while dictionary != False:
   #print(dictionary)
   print ("The Name is : ",  dictionary["NAME"])
   print ("The Email is : ", dictionary["EMAIL"])
   print ("The Phone is : ", dictionary["PHNUM"])
   print(" ******************** ")
   values.update(dictionary)
      dictionary = ibm_db.fetch_both(stmt)
```

7.3 DATABASE SCHEME




IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Schemas
Tables
New table
Name
Schema
Properties
DONOR QDZ26030
REQUESTER QDZ26030
USER QDZ26030
Total: 3, selected: 0
Table definition
USER
Approximate 1 rows (32.0 KB)
Updated on 2022-11-05 04:59:28
Name
Data type
Nullable
Length
Scale
NAME CHAR Y 50 0
EMAIL CHAR Y 50 0
PHNUM CHAR Y 25 0
PHNUM2 CHAR Y 25 0
PAS CHAR Y 50 0
PAS2 CHAR Y 50 0
GEN CHAR Y 50 0
View data


IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Schemas
Tables
New table
Name
Schema
Properties
DONOR QDZ26030
REQUESTER QDZ26030
USER QDZ26030
Total: 3, selected: 0
Table definition
REQUESTER
Approximate 13 rows (32.0 KB)
Updated on 2022-11-15 12:55:51
Name
Data type
Nullable
Length
Scale
NAME CHAR Y 50 0
LNAME CHAR Y 50 0
EMAIL CHAR Y 50 0
PHNUMR CHAR Y 50 0
PHNUMR2 CHAR Y 50 0
ADDRESS CHAR Y 50 0
BLOODGRP CHAR Y 50 0
View data


IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Schemas
Tables
New table
Name
Schema
Properties
DONOR
QDZ26030
REQUESTER
QDZ26030
USER
QDZ26030
Total: 3, selected: 0
Table definition
DONOR
Approximate 8 rows (32.0 KB)
Updated on 2022-11-15 12:55:51
Name
Data type
Nullable
Length
Scale
NAME CHAR Y 50 0
EMAIL CHAR Y 50 0
PHNUM CHAR Y 50 0
PHNUM2 CHAR Y 50 0
BLOOD CHAR Y 50 0
STATES CHAR Y 50 0
DISTRICT CHAR Y 50 0
ADDRESS CHAR Y 50 0
View data

8.TESTING

8.1 TEST CASES

Name :- User login module				
No	Test condition	Expected Result	Actual output	Status (pass, fail)
Test 1	Click on submit button without user name and password.	System does not allow user to login.	System displays message and resume to the same page.	pass
Test 2	Click on submit button with invalid user name and or password.	Message "please fill up the username or password"	As expected.	pass
Test 3	Click on submit button with correct user name add password.	System allow user to login.	System allow user to access application based on rights given to him.	pass

Name :-User Registration				
No	Test condition	Expected Result	Actual output	Status (pass, fail)
Test 1	Click on submit button without user name and password.	System does not allow user to login.	System displays message and resume to the same page.	pass
Test 2	Click on submit button without correct password & re-password.	Message "please fill up the correct password & re password"	System displays message and resume to the same page.	pass
Test 3	Select on user type with correct user registration	System allow user to login.	System allow user to access application based on right given to him.	pass

Name :- Blood request				
No	Test condition	Expected Result	Actual output	Status (pass, fail)
Test 1	Click on update button without add new information of request for blood bank name	System does not allow admin to save data without add blood bank name	System displays message to the same page.	pass

Name :- Donor request				
No	Test condition	Expected Result	Actual output	Status (pass, fail)
Test 1	Click on update button without add new information of donor request for blood bank name.	System does not allow admin to save data without add blood bank name	System displays message to the same page.	pass

Name :- Inquiry				
No	Test condition	Expected Result	Actual output	Status (pass, fail)
Test 1	Click on submit button without name.	System does not allow user to login.	System displays message and resume to the same page.	pass
Test 2	Click on submit button with invalid email id	Message "please fill up the username or password”	As expected.	pass
Test 3	Click on submit button with correct contact no	System allow user to login.	System allow user to access application based on rights given to him.	pass

9.RESULTS

9.1 AUTHENTICATION MODULE

• Sign Up- New user or donor can create an account to use in the blood/plasma donor application and create a password for account verification and create an identity.

• Sign In- Donor Sign Into the account for viewing or editing location details and any other personal information.

• Account Verification -If donor changes their password or if they forget the password then we must verify their account using mail verification.

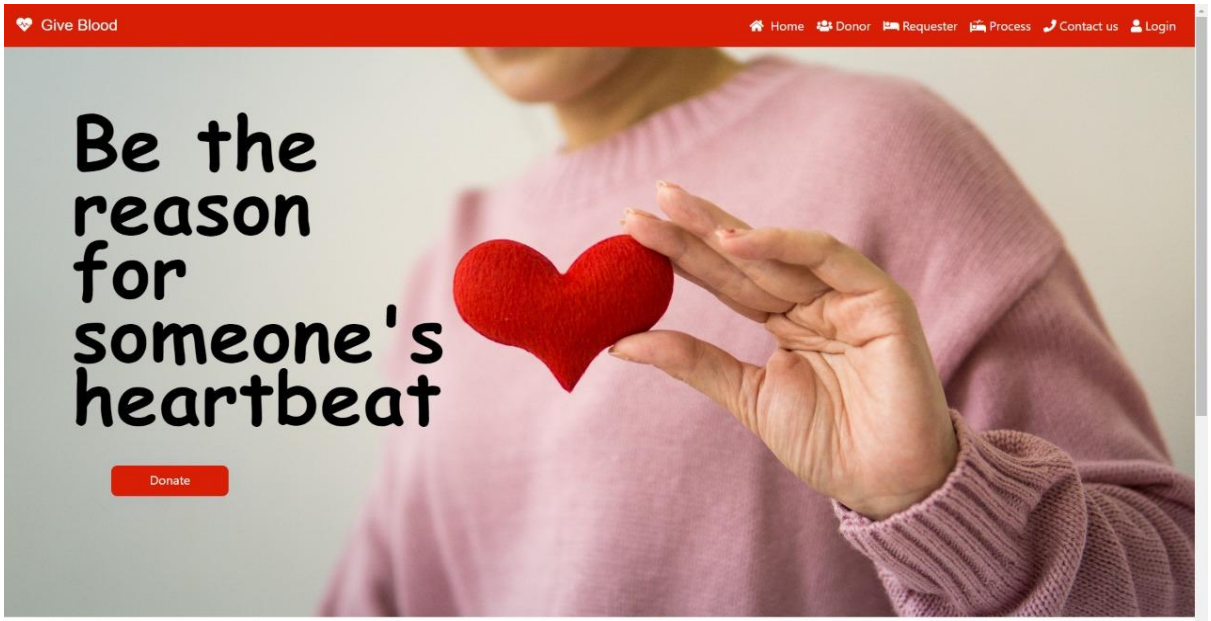
9.2 SERVICE PROVIDER MODULE

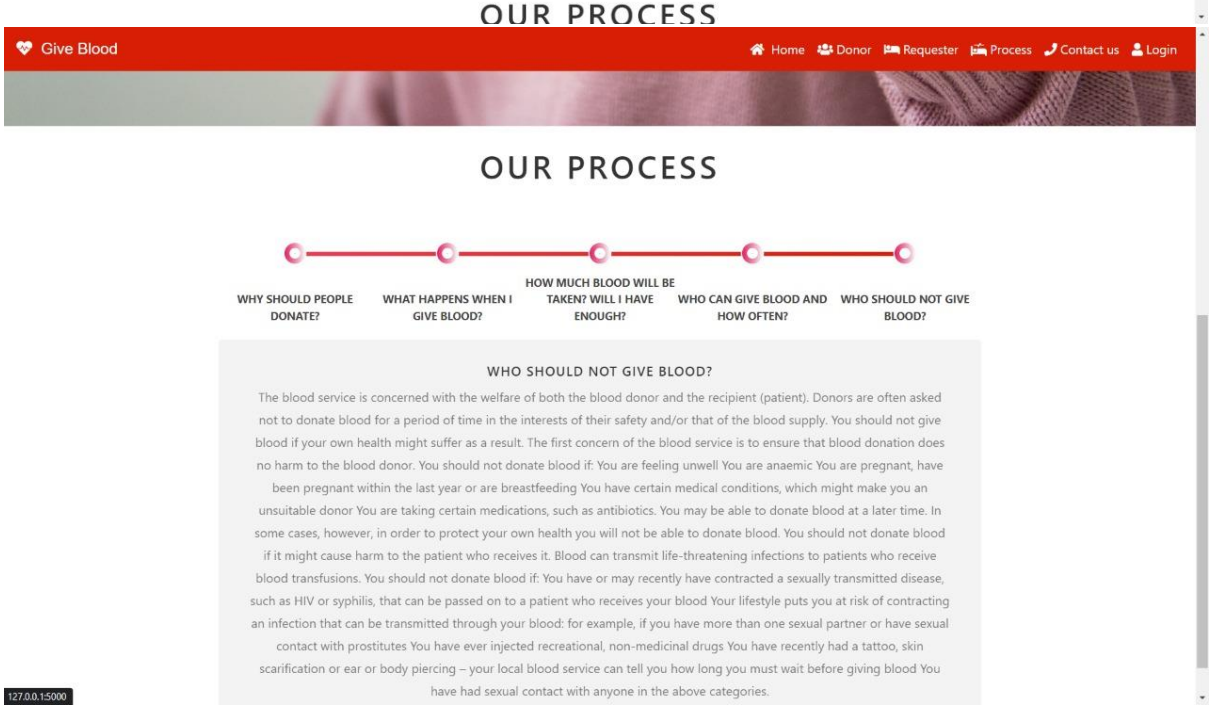
• Add New Donor- User can be able to register to add donor details .

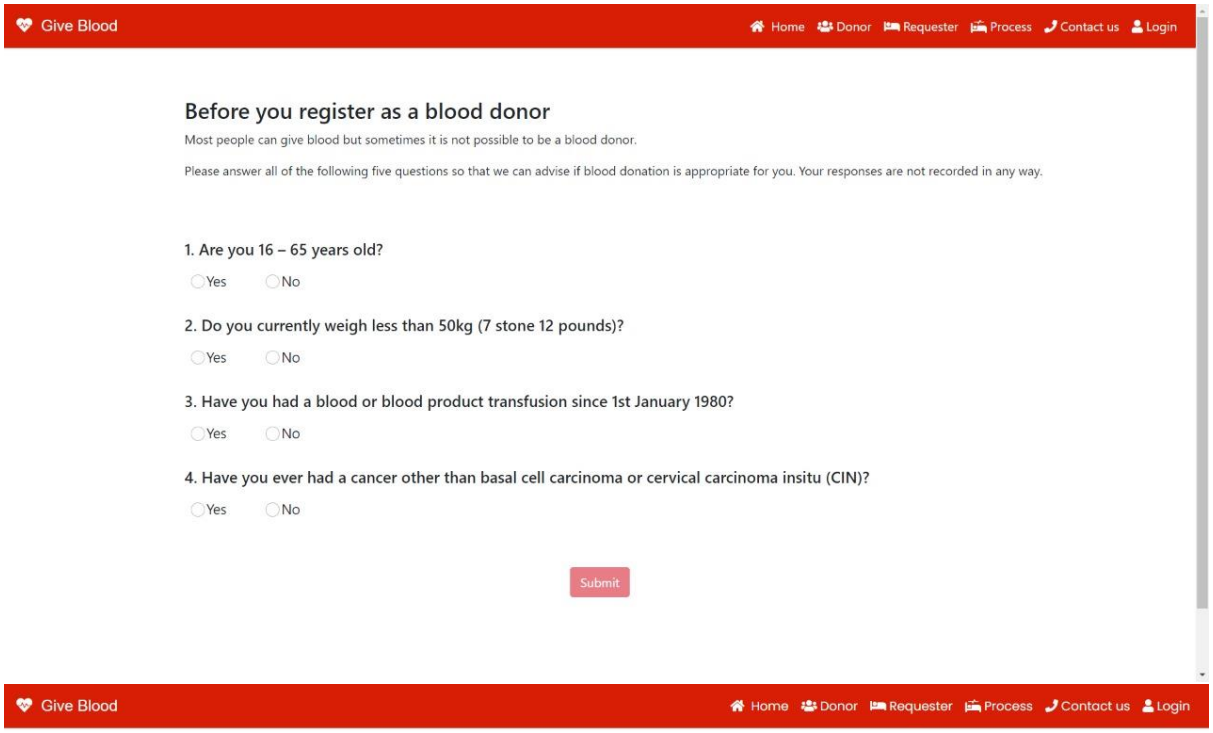
• List All Donor -User can be able to view all Donor who all use our Plasma Donor Application.

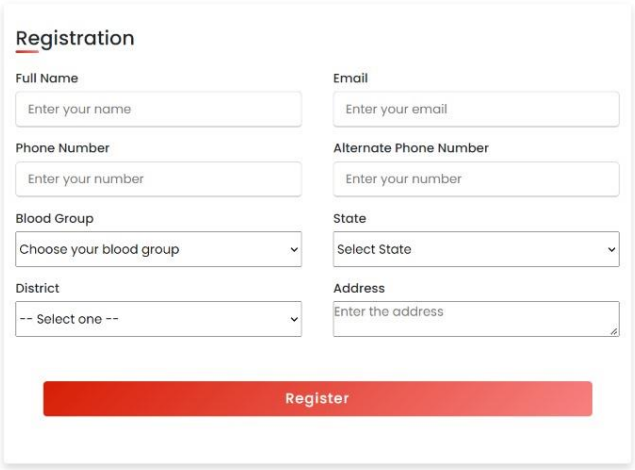
• Edit Customer Plan Details User can be able to edit the existing Donor details as the Donor wish.

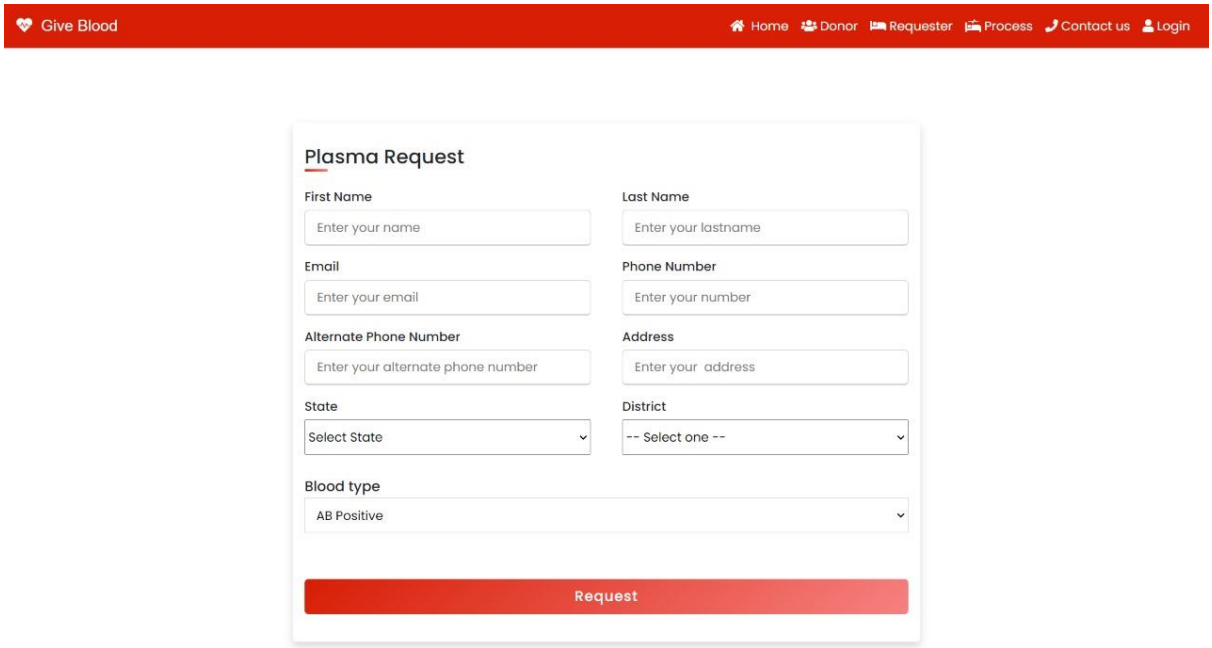
9.3 SCREEN LAYOUT



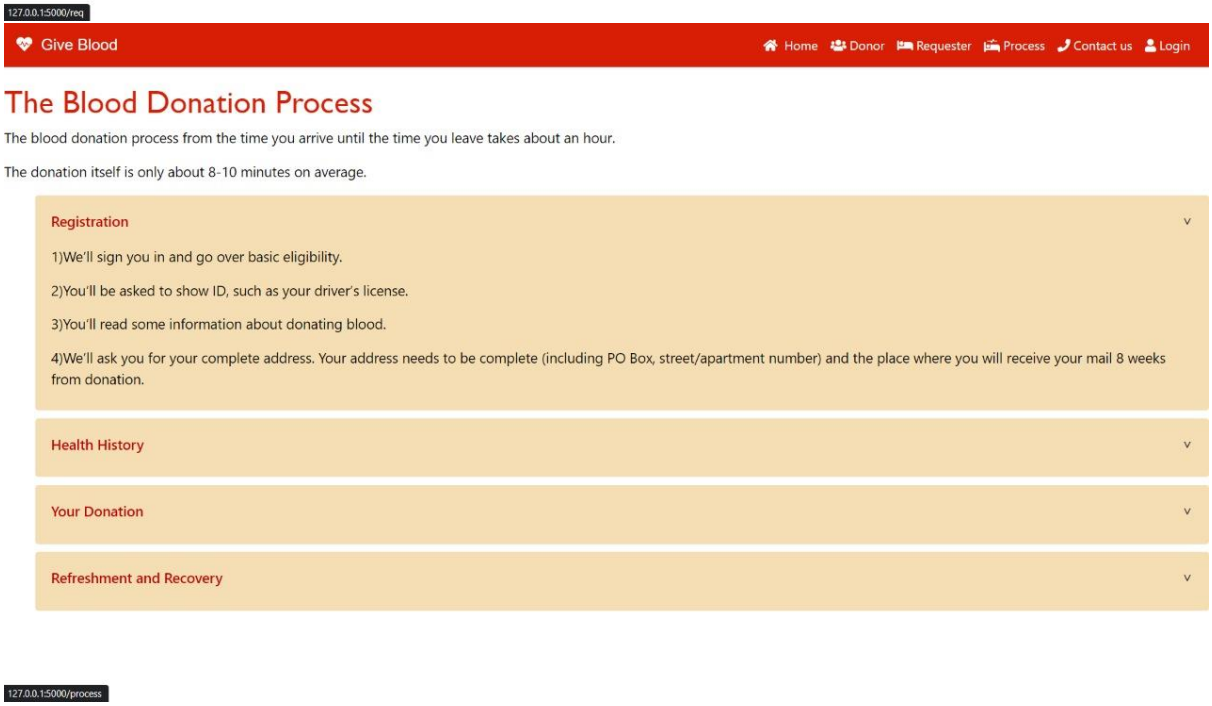



Give Blood
Home
Donor
Requester
Process
Contact us
Login
Before you register as a blood donor
Most people can give blood but sometimes it is not possible to be a blood donor.
Please answer all of the following five questions so that we can advise if blood donation is appropriate for you. Your responses are not recorded in any way.
1. Are you 16 – 65 years old?
Yes
No
2. Do you currently weigh less than 50kg (7 stone 12 pounds)?
Yes
No
3. Have you had a blood or blood product transfusion since 1st January 1980?
Yes
No
4. Have you ever had a cancer other than basal cell carcinoma or cervical carcinoma insitu (CIN)?
Yes
No
Submit
Give Blood
Home
Donor
Requester
Process
Contact us
Login

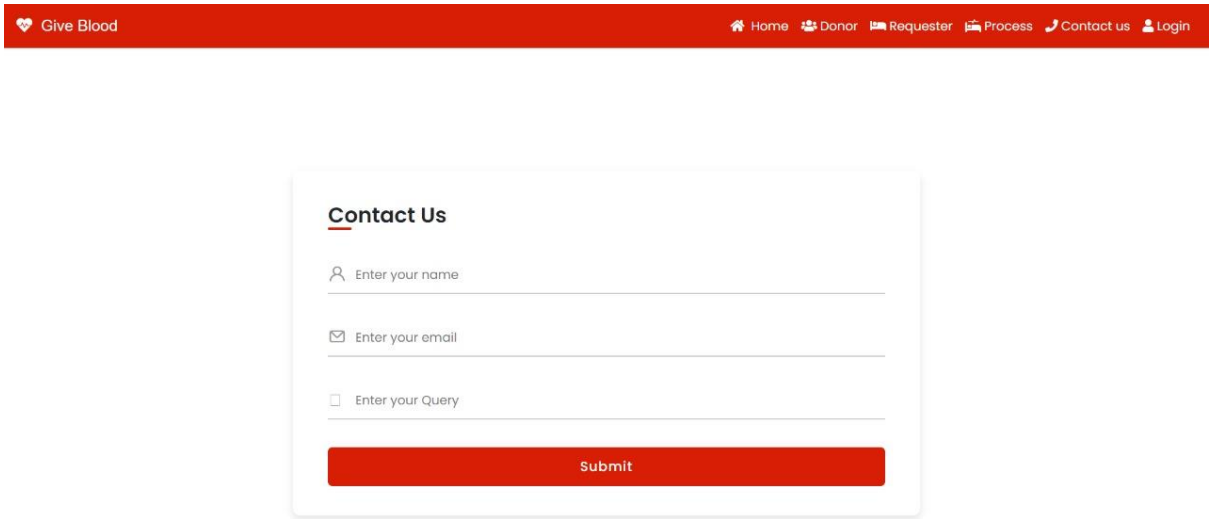

Registration
Full Name
Enter your name
Email
Enter your email
Phone Number
Enter your number
Alternate Phone Number
Enter your number
Blood Group
Choose your blood group
State
Select State
District
-- Select one --
Address
Enter the address
Register

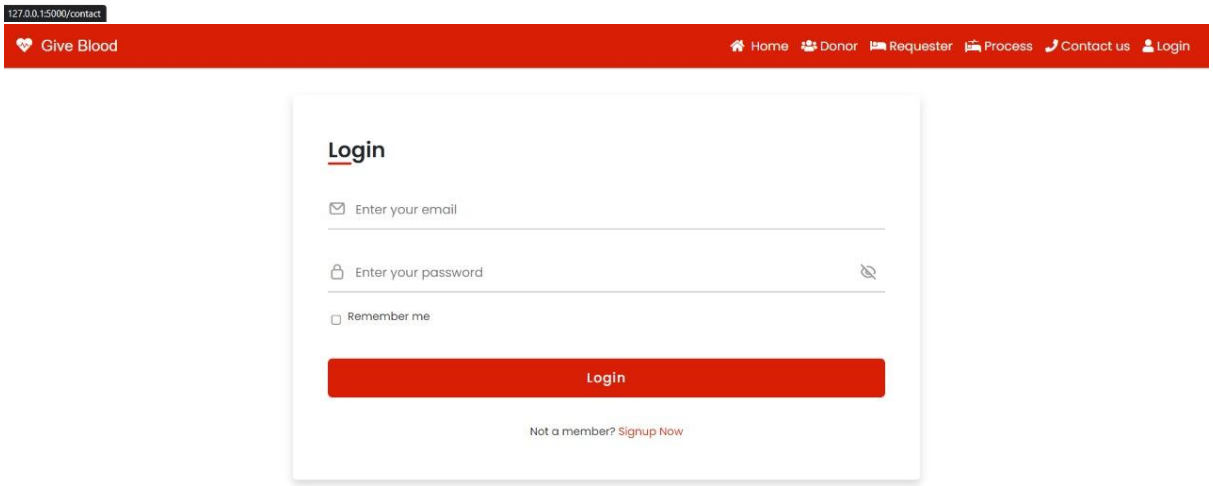


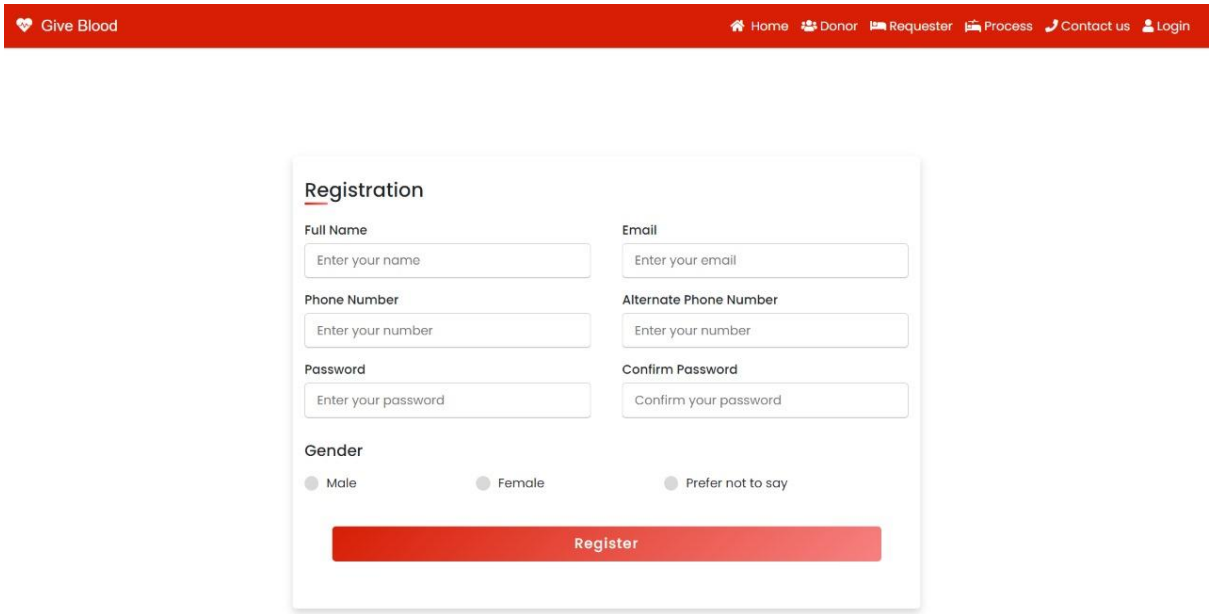
127.0.0.1:5000/req



127.0.0.1:5000/process


Give Blood
Home
Donor
Requester
Process
Contact us
Login
Contact Us
Enter your name
Enter your email
Enter your Query
Submit


127.0.0.1:5000/contact
Give Blood
Home
Donor
Requester
Process
Contact us
Login
Login
Enter your email
Enter your password
Remember me
Login
Not a member? Signup Now



The Donors can register their account using their email ID. Once registered, The Donor can sign-up by using his\her respective password. The login page for Plasma Donors is shown in the figure, which contains the E-mail and Password field. The profile of the Donor, where he/she needs to enter the required details. After registration Donor can maintain according to his availability. The registration page with Full Name, Email Address, Password, Contact Details, Blood Group, Location and all other details, which is illustrated. The details of the available donors can be displayed and viewed by other users.

10.ADVANTAGES AND DISADVANTAGES

Advantages

• Speed:

This website is fast and offers great accuracy as compared to manual registered keeping.

• Maintenance:

Less maintenance is required

• User Friendly:

It is very easy to use and understand. It is easily workable and accessible for everyone.

• Fast Results:

It would help you to provide plasma donors easily depending upon the availability of it.

Disadvantages

• Internet:

It would require an internet connection for the working of the website.

• Auto-Verification:

It cannot automatically verify the genuine users.

11.CONCLUSION

Although the government is carrying out Covid vaccination campaigns on a large scale, the number of vaccines produced is not enough for all the population to get vaccinated at present. And with the corona positive cases rising every day, saving lives has become the prime matter of concern. As per the data provided by WHO more than 3 million people have died due to the coronavirus. However, apart from vaccination, there is another scientific method by which a covid infected person can be treated and the death risk can be reduced. This plasma therapy is an experimental approach to treat corona positive patients and help them recover. This plasma therapy is considered to be safe & promising. A person who has recovered from Covid can donate his/her plasma to a person who is infected with the coronavirus. This 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 frameworks.

12.FUTURE SCOPE

Plasma Application can be developed to further improve user accessibility via integrating this application with various social networks application program interfaces (APIs). Consequently, users can login and sign up using various social networks. This would increase number of donors and enhances the process of blood donation. 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 experiences about 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 to his blood/plasma donation appointments.

13.APPENDIX

SOURCE CODE

https://drive.google.com/drive/folders/1fymiCUULtPJD98Bmzvu7MNPrvsvFwdUQ?usp=share_link

GITHUB LINK

https://github.com/IBM-EPBL/IBM-Project-2234-1658467693.git

DEMO LINK

https://drive.google.com/file/d/1p79Wtx3JhwzB0jmztcbdiUizUk74JHx2/view?usp=share_link