A picture containing graphical user interface

Description automatically generated

**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**

1. **INTRODUCTION**

1.1 Project Overview

1.2 Purpose

1. **LITERATURE SURVEY**

2.1 Existing problem

* 1. References
  2. Problem Statement Definition

1. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit

**4. REQUIREMENT ANALYSIS**

* 1. Functional requirement
  2. Non-Functional requirements

1. **PROJECT DESIGN**
   1. Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

1. **PROJECT PLANNING & SCHEDULING**
   1. Sprint Planning & Estimation
   2. Sprint Delivery Schedule
   3. Reports from JIRA
2. **CODING & SOLUTIONING** 
   1. Feature 1
   2. Feature 2
   3. Database Schema
3. **TESTING** 
   1. Test Cases
4. **RESULTS**
   1. Authentication Module
   2. Service provider module
   3. Screen layout
5. **ADVANTAGES & DISADVANTAGES**
6. **CONCLUSION**
7. **FUTURE SCOPE**

**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.

A picture containing timeline

Description automatically generated

**3. IDEATION AND PROPOSED SOLUTION**

**3.1 EMPATHY MAP CANVAS**

Diagram, timeline

Description automatically generated

**3.2 IDEATION AND BRAINSTORMING**

Graphical user interface, application

Description automatically generated

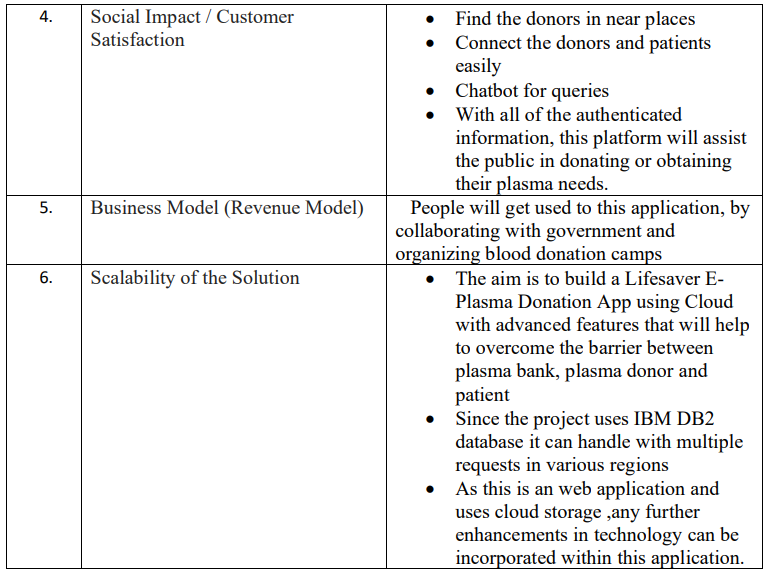
**3.3 PROPOSED SOLUTION**

Table

Description automatically generated

Text

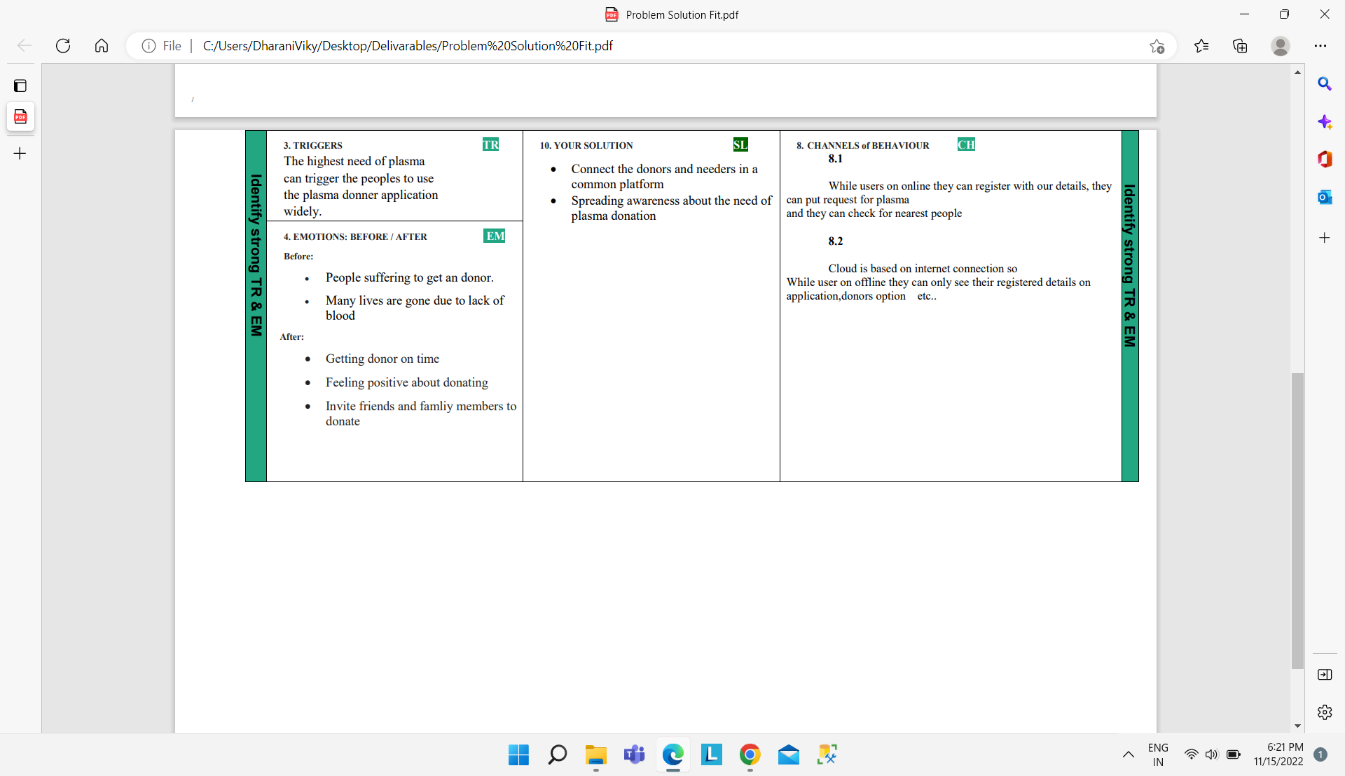
Description automatically generated



**3.4 PROBLEM SOLUTION FIT**

**Calendar

Description automatically generated**



**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:**

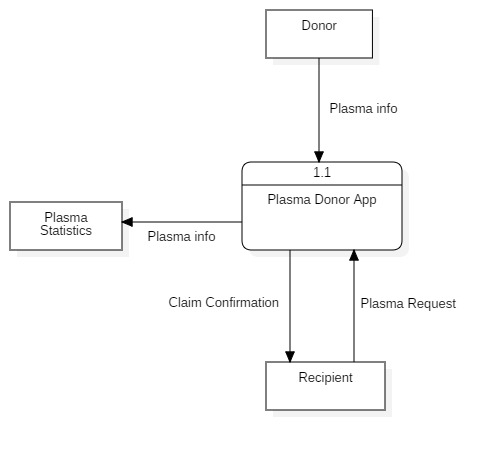
Following are the non-functional requirements of the proposed solution.

|  |  |  |
| --- | --- | --- |
| **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 |

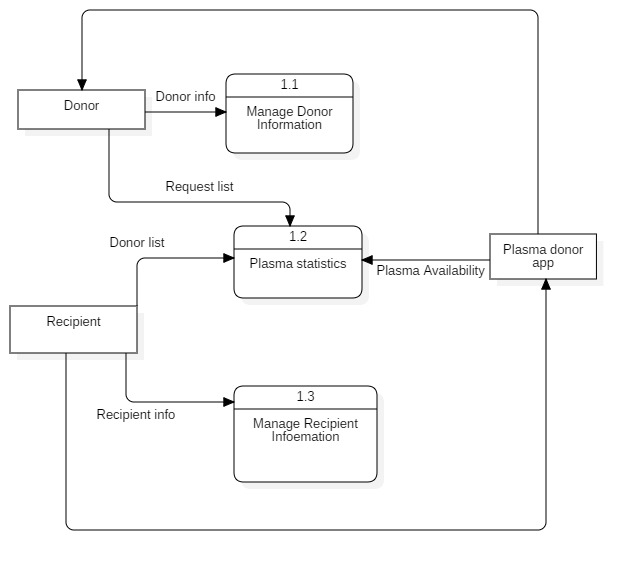
**5.PROJECT DESIGN**

**5.1 DATA FLOW DIAGRAMS**

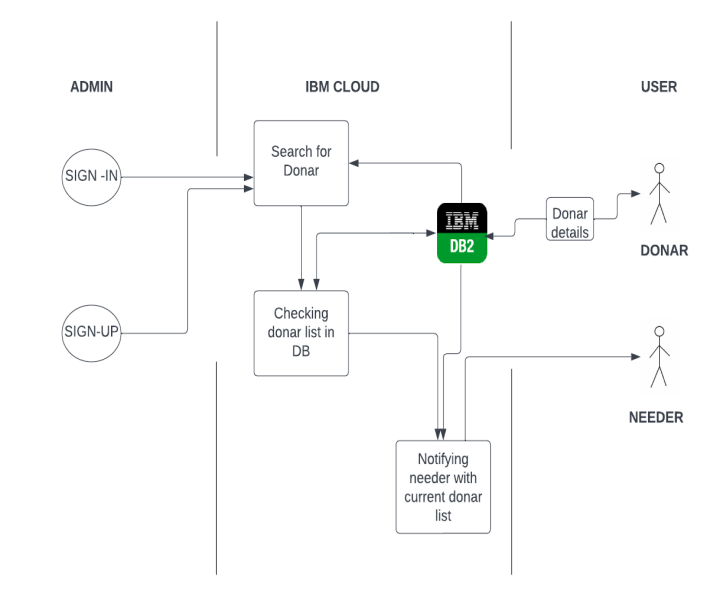
**Level 0**



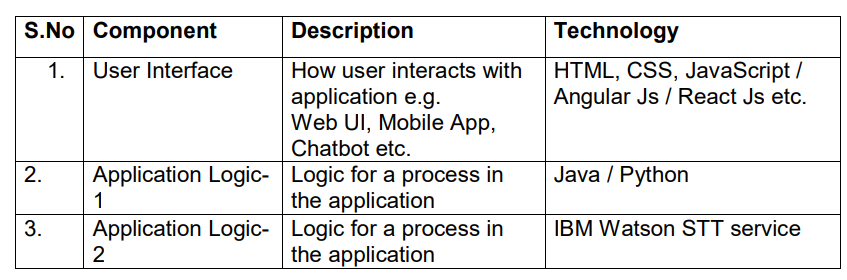
**Level 1**

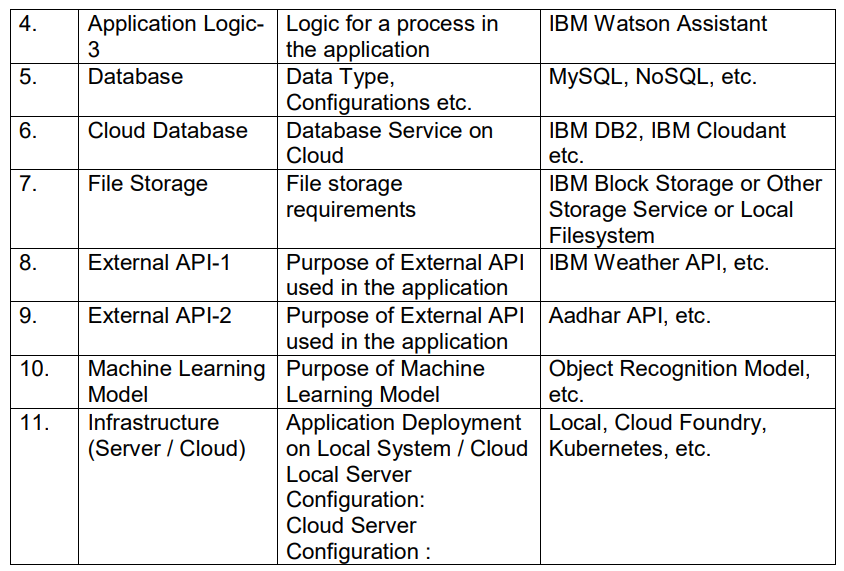


**5.2 SOLUTION AND TECHNOLOGY ARCHITECTURE**

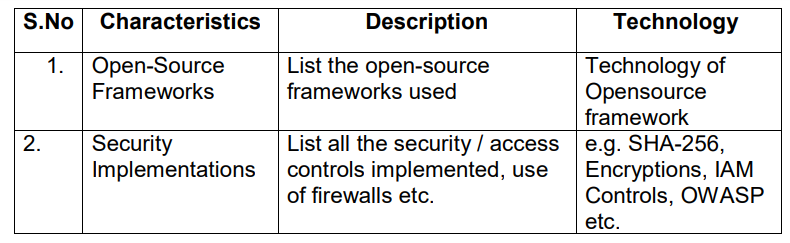


**COMPONENTS AND TECHNOLOGIES**





**APPLICATION CHARACTERISTICS**

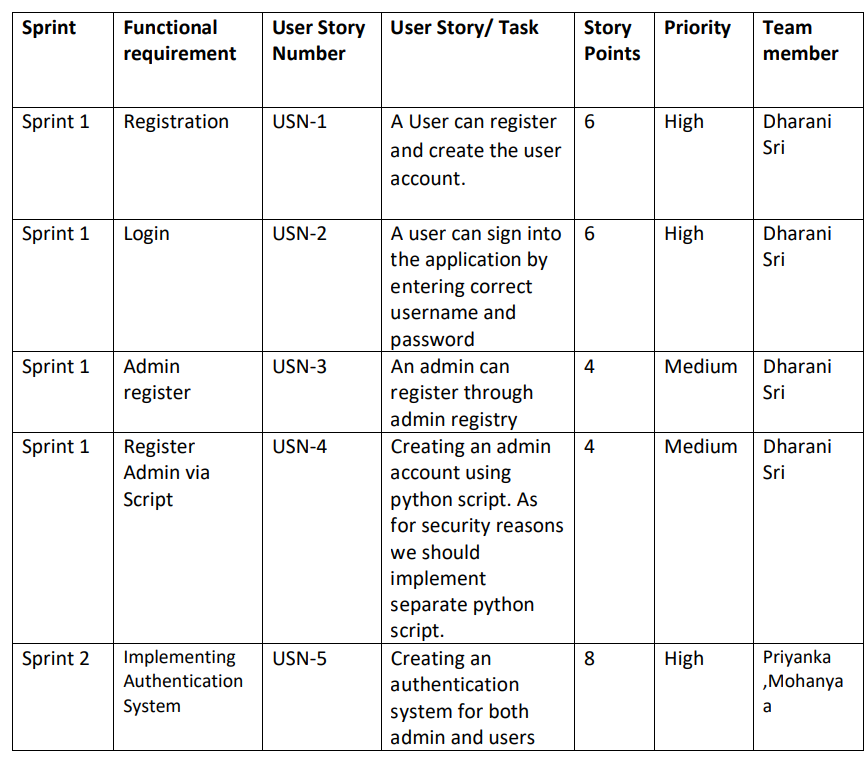
****

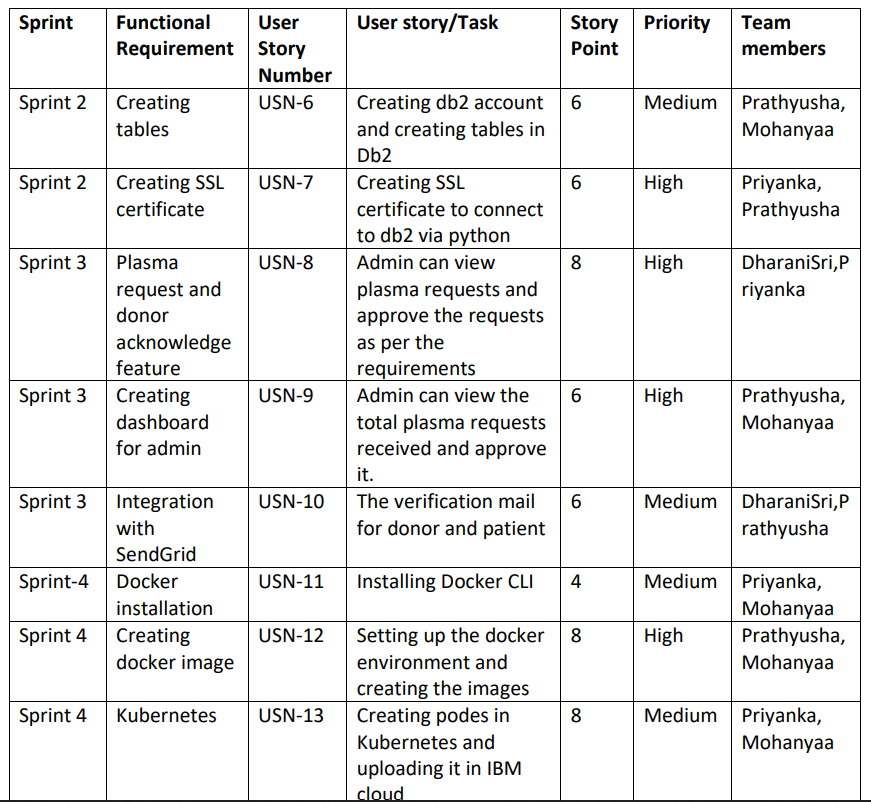
**5.3 USER STORIES**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 |

1. **PROJECT PLANNING & SCHEDULING**
   1. **SPRINT PLANNING AND ESTIMATION**

****



**6.2 SPRINT DELIVERY SCHEDULE**

**Table

Description automatically generated**

**6.3 REPORTS FROM JIRA**

**A picture containing graphical user interface

Description automatically generated**

**Graphical user interface, application, Teams

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**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](http://smtp.sendgrid.net/)'

app.config['MAIL\_PORT'] = 587

app.config['MAIL\_USE\_TLS'] = True

app.config['MAIL\_USERNAME'] = 'apikey'

app.config['MAIL\_PASSWORD'] = 'SG.awCtlHRgR4axIysEvvskxQ.hhmozXUcHXtMZ4kQyz\_VU1jjZChjAmnV8ZMKKtnKpG8'

app.config['MAIL\_DEFAULT\_SENDER'] = '[ZrPlasmaDonor@outlook.com](mailto: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=30119;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"])

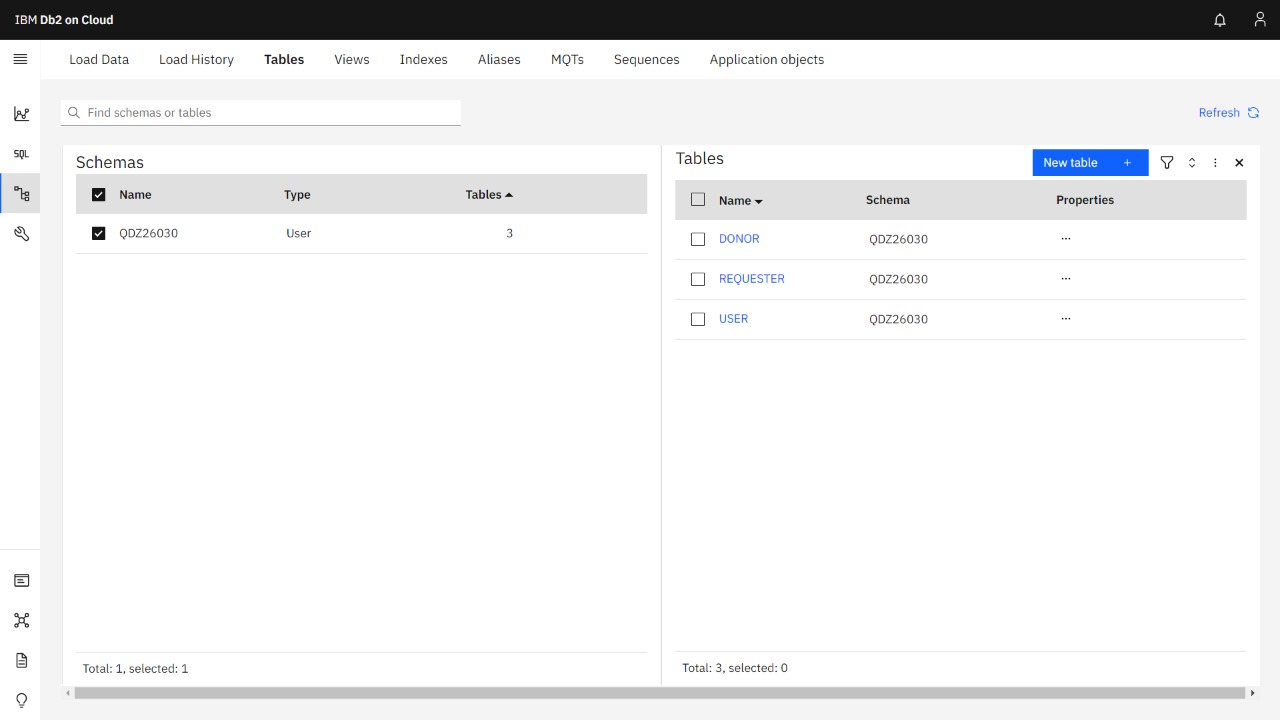
    print ("The Phone is : ", dictionary["PHNUM"])

    print(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ")

    values.update(dictionary)

    dictionary = ibm\_db.fetch\_both(stmt)

**7.3 DATABASE SCHEME**

****

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

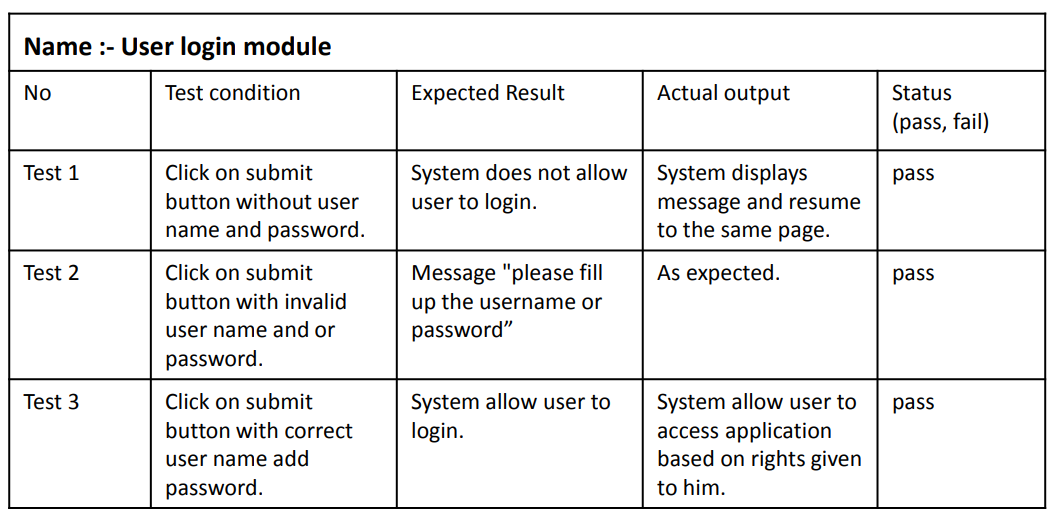
Description automatically generated**

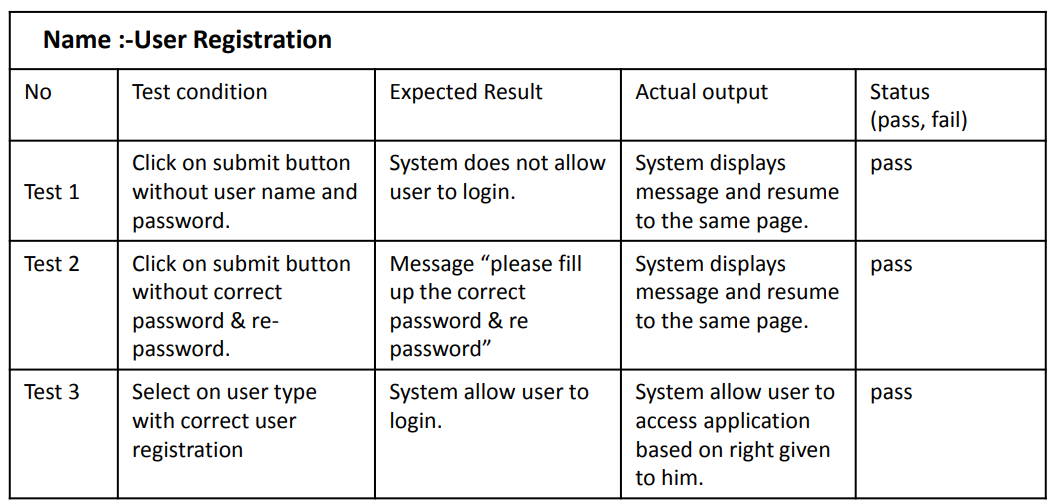
**Graphical user interface, application

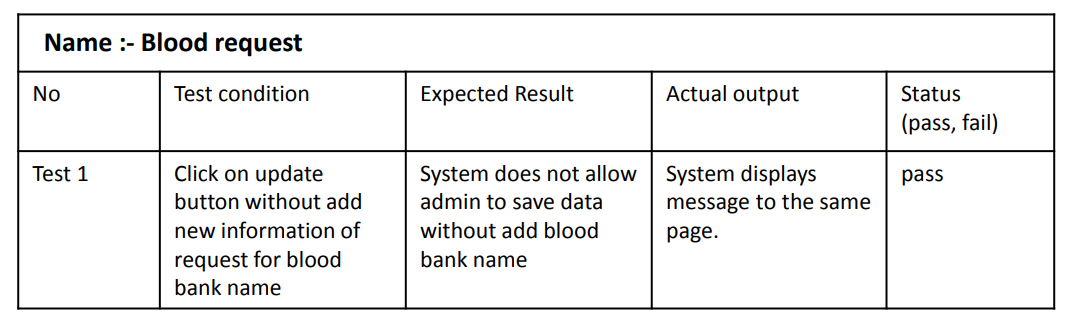
Description automatically generated**

**8.TESTING**

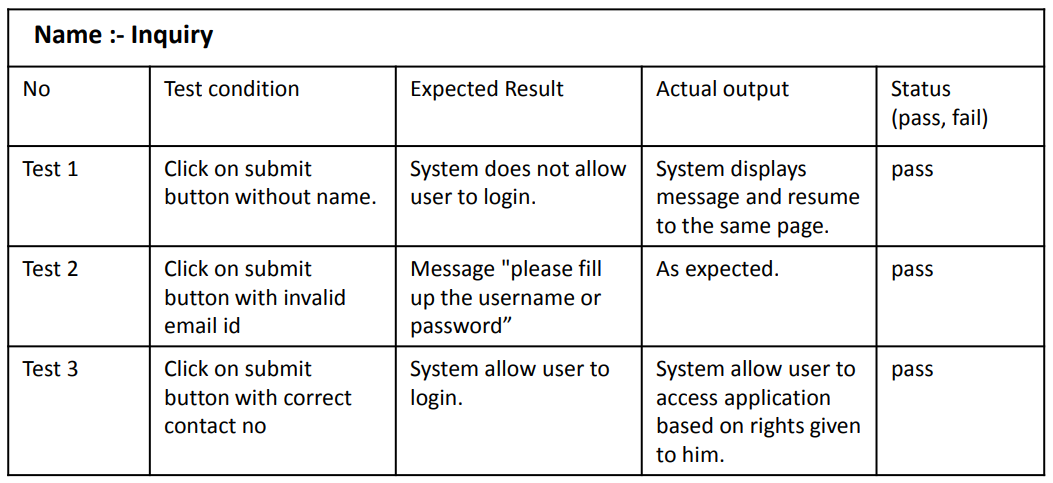
**8.1 TEST CASES**

****

****

**Table

Description automatically generated**

****

**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.

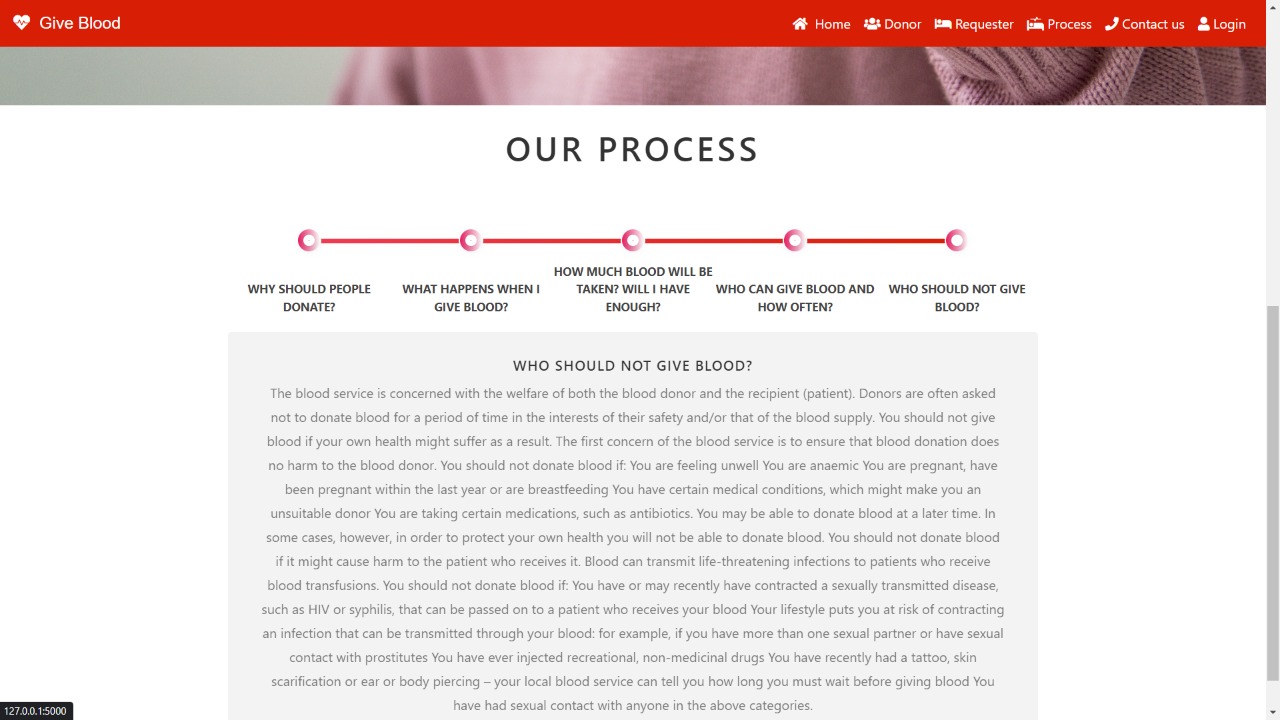
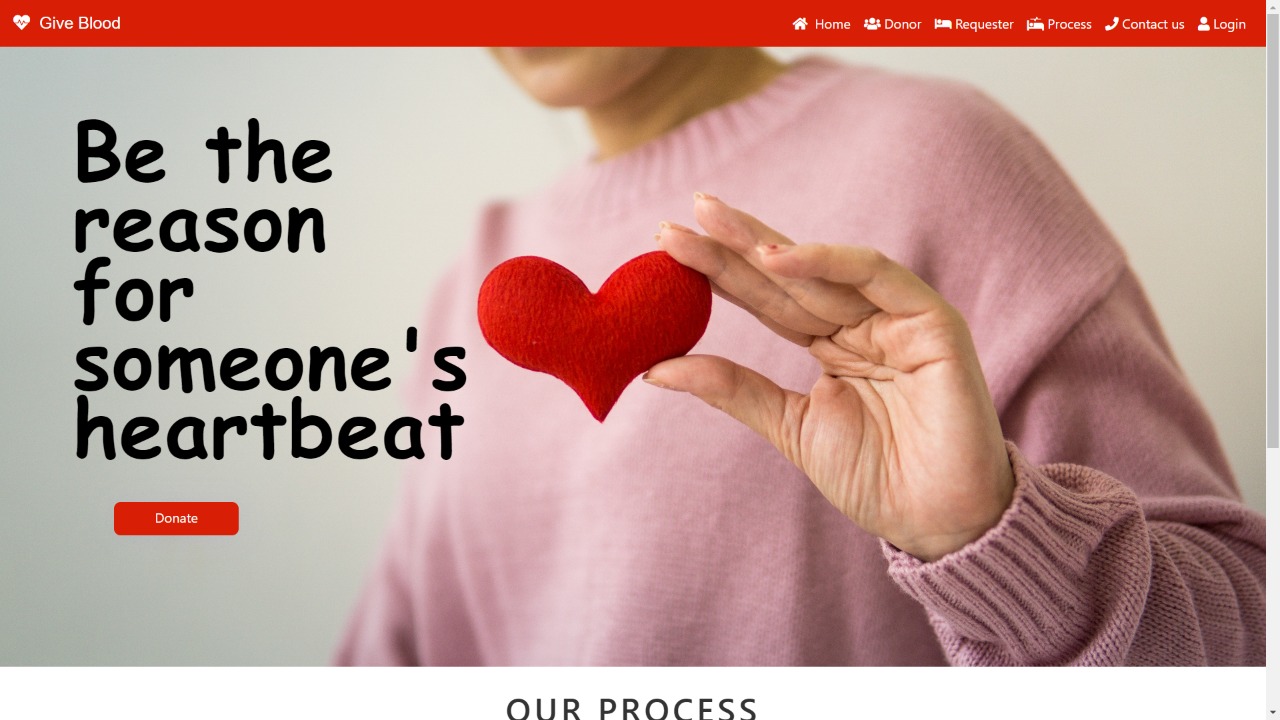
**9.2 SERVICE PROVIDER MODULE**

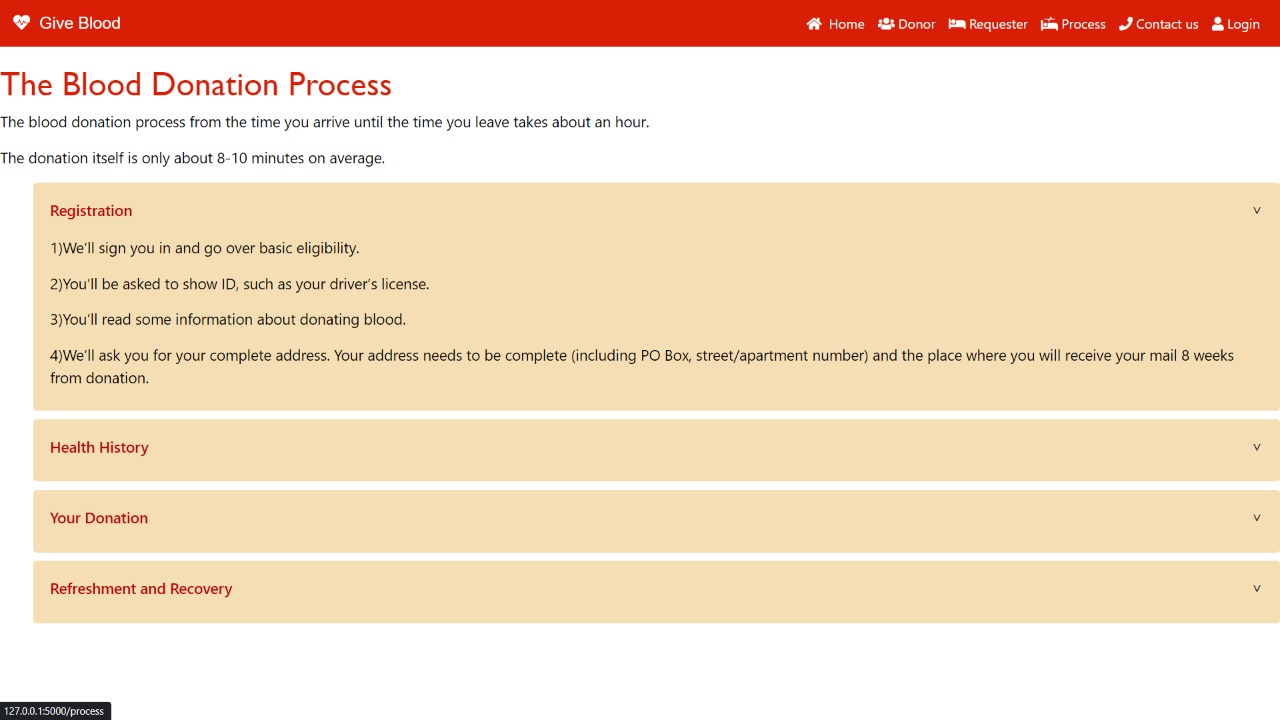
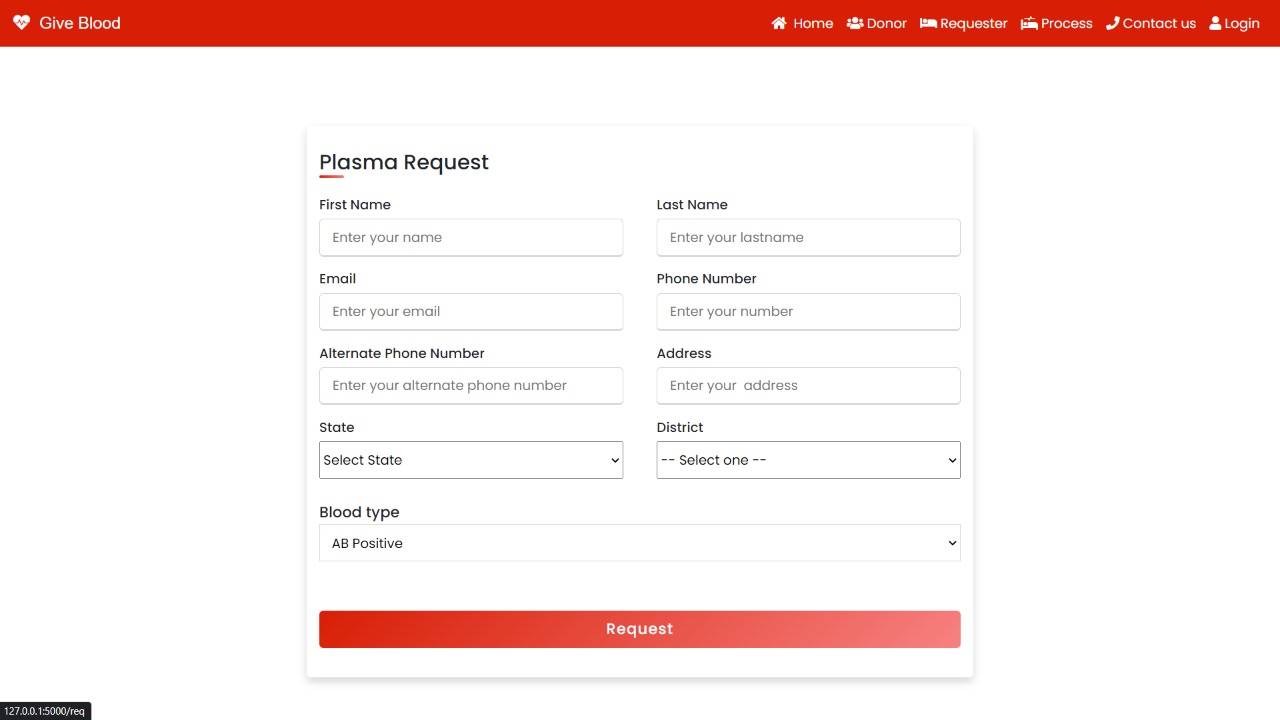
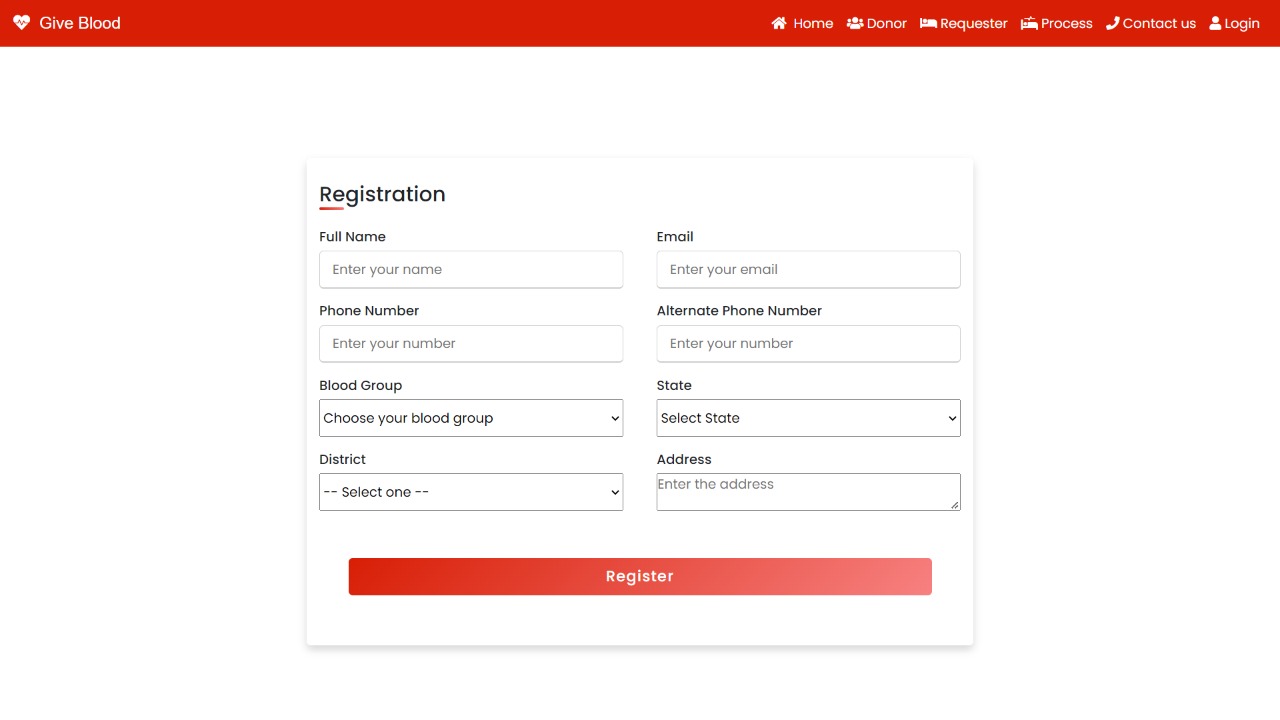
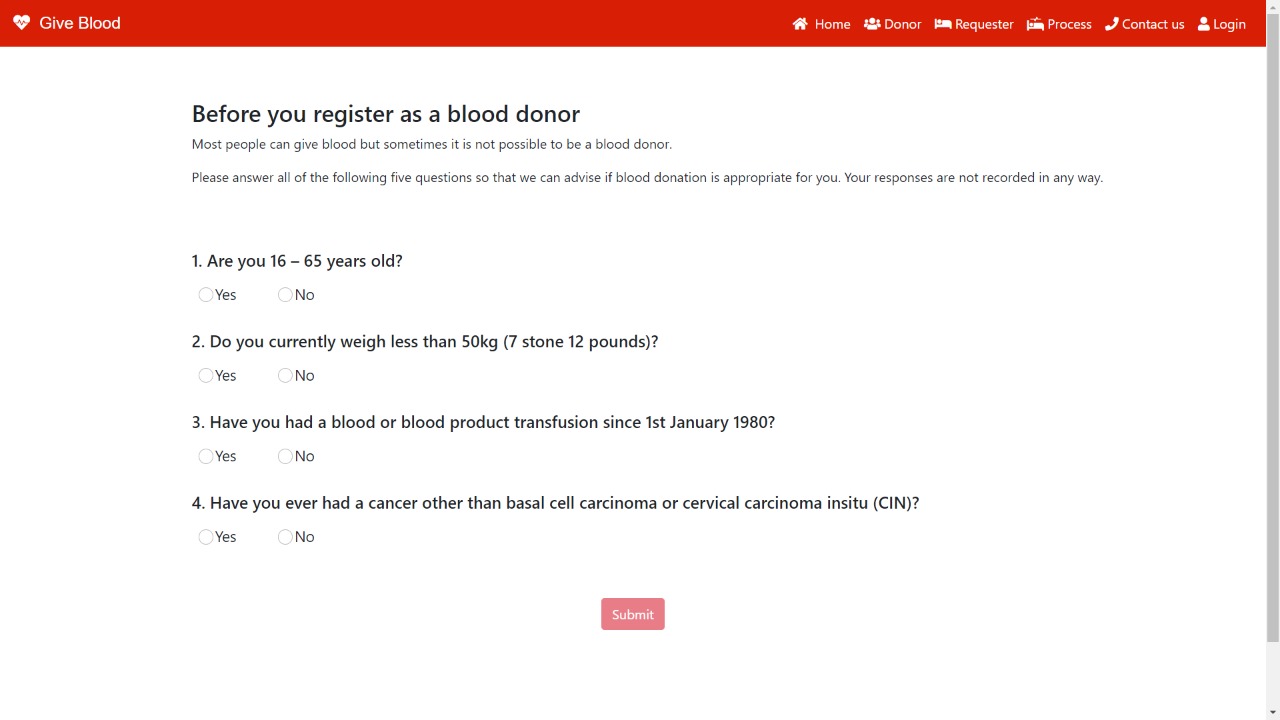
• Add New Donor- User can be able to register to add donor details .

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

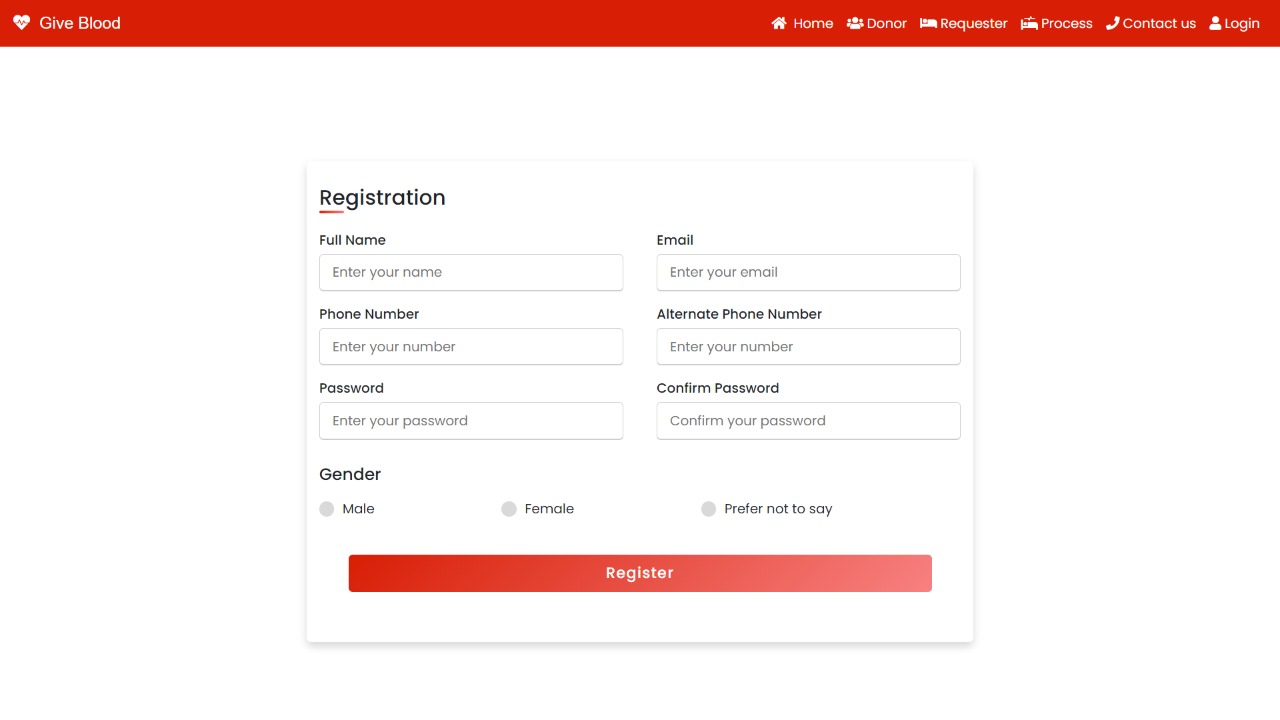
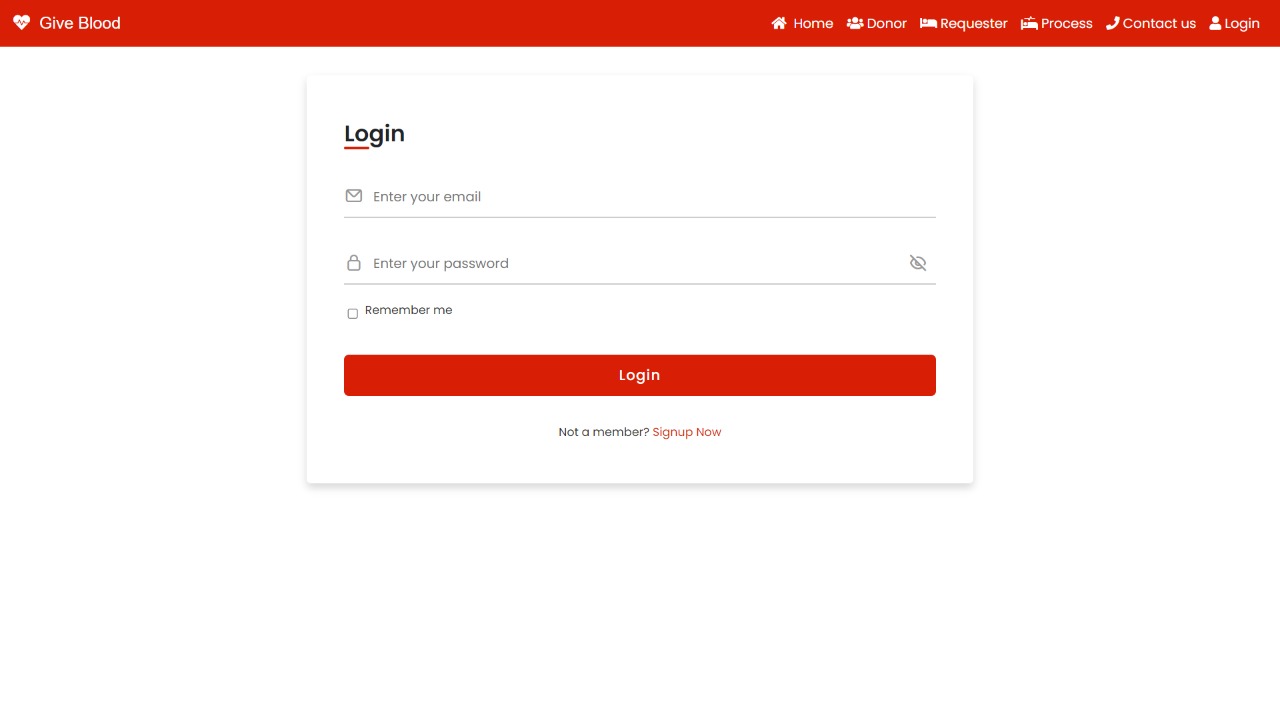
• Edit Customer Plan Details User can be able to edit the existing Donor details as the Donor wish.

**9.3 SCREEN LAYOUT**

****

****

**Graphical user interface, application

Description automatically generated**

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