PLASMA DONOR APPLICATION PROJECT REPORT

Team Id	PNT2022TMID24834
Project Name	Plasma Donor Application
Team Members	M K Aravindan Deepak M Sathish G Seenu M

1. Introduction

1.1 Project Overview

Project Name: Plasma Donor Application

Problem Statement:

People who need plasma in Covid days are increased day by day. People who have disease or people who have gotten into accidents and run out of Immune need constant supply of plasma to sustain their life and there is not enough plasma available for them. It is not that people do not want to donate plasma, but because they have no idea where they can donate. It is important for the people who are excited to donate, but yet are very busy, to be sure where and when they can donate ,and therefore We are designing a system which contains all the information regarding plasma donation camps ongoing in a particular area so that people who want to donate plasma will get information regarding these camps. Our System is a mobile application which aims to serve as a communication tool between Plasma Donation camp Organizers and Plasma donors . To become a member of the system, donors need to create their profile by providing the information like name, blood group, email address, password, and exact location from “Google Map”. The mobile application always keeps updating the location of a donor. As a result, the system can automatically keep showing the nearby Plasma donation Camps to the registered donor wherever they go, and donors can easily get the idea of nearby blood donation camps. Also, users can get information regarding the type of blood which is available and information of past as well as future events

Goal

Donated blood helps meet many medical needs — including saving the life of a baby, restoring the strength of a cancer patient and providing a critical transfusion to someone who has been in a disorder.

Project Progress and Work done

We are a team of 4.Our plan is to create a application using Html,Css,Python Flask,IBM db2,Kubernates,Docker,Send-grid.We did the front end with the necessary things such as Home page,login page for admin, Registration page for donor Active and inactive members can be viewed,info of various blood groups.back end is done in Python flask and all the files are integrated with the IBM db2.

1.2 Purpose

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.

2.Literature Survey

1. Bloomberg School of Public Health staff report

Sex, age, and severity of disease may be useful in identifying COVID-19 survivors who are likely to have high levels of antibodies that can protect against the disease, according to a new study co-led by researchers at Johns Hopkins Bloomberg School of Public Health.

The findings suggest that older males who have recovered from COVID-19 after having been hospitalized are strong candidates for donating plasma for treating COVID-19 patients. Doctors have been using infusions of plasma—the part of blood that contains antibodies—from recovered COVID-19 patients to treat other patients and also as a possible prophylaxis for prevention.

Doctors have used convalescent plasma to treat patients or immunize persons at high risk of virus exposure during outbreaks of measles, mumps, polio, Ebola, and even the 1918 pandemic flu.

2.Blood Donors & Blood Collection by Christopher R.France.

With growing discussion about blood donor remuneration, the present study examined the level of payment that may be required to convince individuals to engage in whole blood, plasma, and platelet donations.

Level of payment needed to motivate whole blood, plasma, and platelet donation was examined as a function of donation history, sample, and gender. In addition, path analyses examined associations between donation motivators, barriers, and payment level.

Across all types of donation, history of whole blood donation was related to a greater willingness to donate without payment. At the same time, however, sizeable portions of prior donors indicated that monetary payment would convince them to donate whole blood (24%), plasma (51%), or platelets (57%).

3.Management of Blood Donation System by Seda Ba¸s, Giuliana Carello.

Applying optimization methods to healthcare management and logistics is a devel_oping research area with numerous studies. Specifically, facility location, staffrostering, patient allocation, and medical supply transportation are the main themesanalysed. Optimization approaches have been developed for several healthcare related problems, ranging from the resource management in hospitals to the delivery of care services in a territory. However, optimization approaches can also improve other services in the health system that have been only marginally addressed, yet.One of them is the Blood Donation (BD) system, aiming at providing an adequate supply of blood to Transfusion Centres (TCs) and hospitals.

4.Android Projects:

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.

Advantages

1.It is a user-friendly application.

2.It will help people to find plasma easily.

Disadvantages

1.It cannot auto verify user genuineness.

2.It requires an active internet connection.

2.2 References

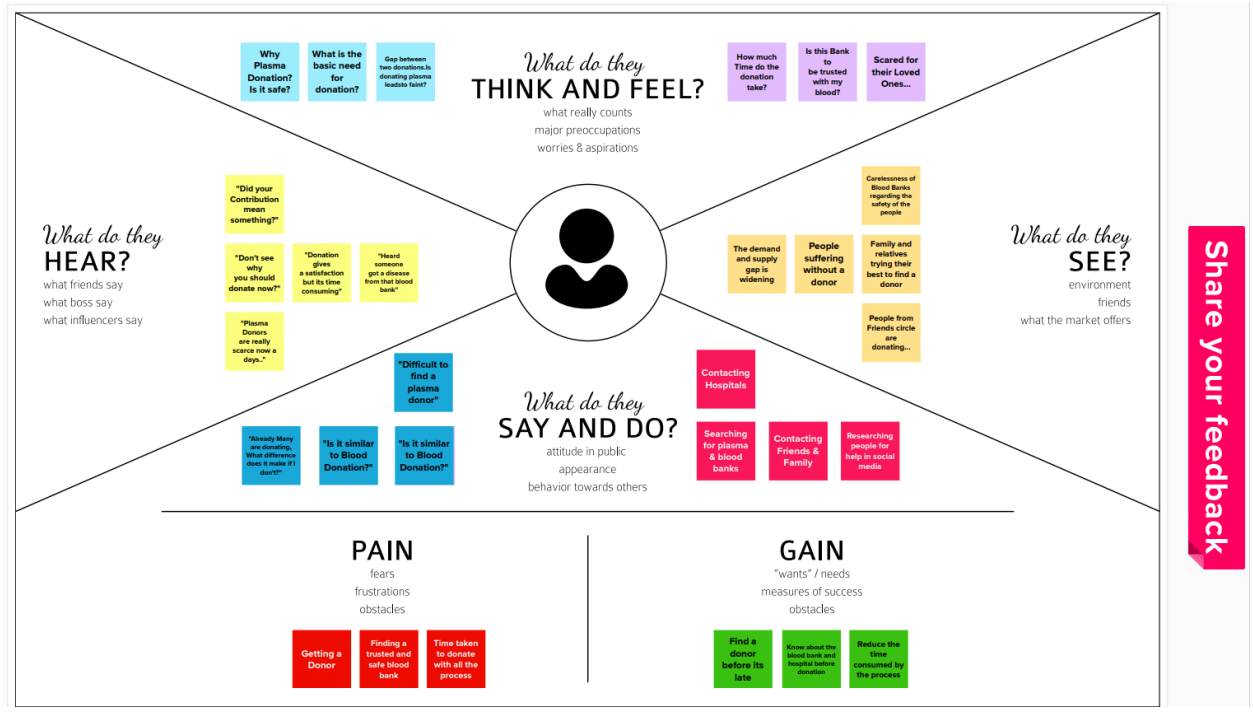
1. *Bloomberg School of Public Health staff report*
2. *Blood Donors & Blood Collection by Christopher R.France*
3. *Management of Blood Donation System by Seda Ba¸s, Giuliana Carello.*
4. *Android Projects*

2.3 Problem statement Defintion

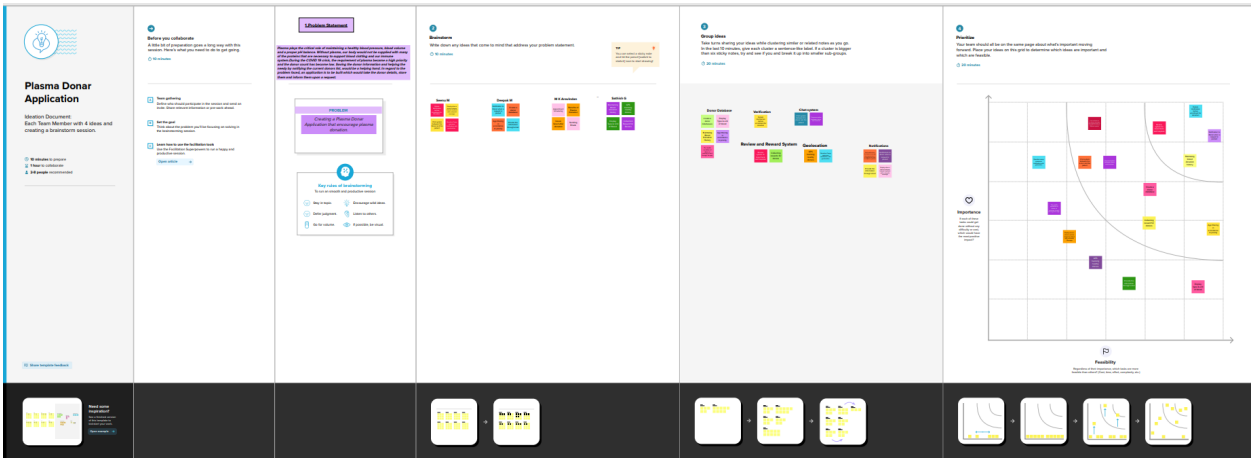
During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low. Saving the donor information and helping the needy by notifying the current donors list, would be a helping hand. In regard to the problem faced, an application is to be built which would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

3.Ideation & Proposed Solution

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming



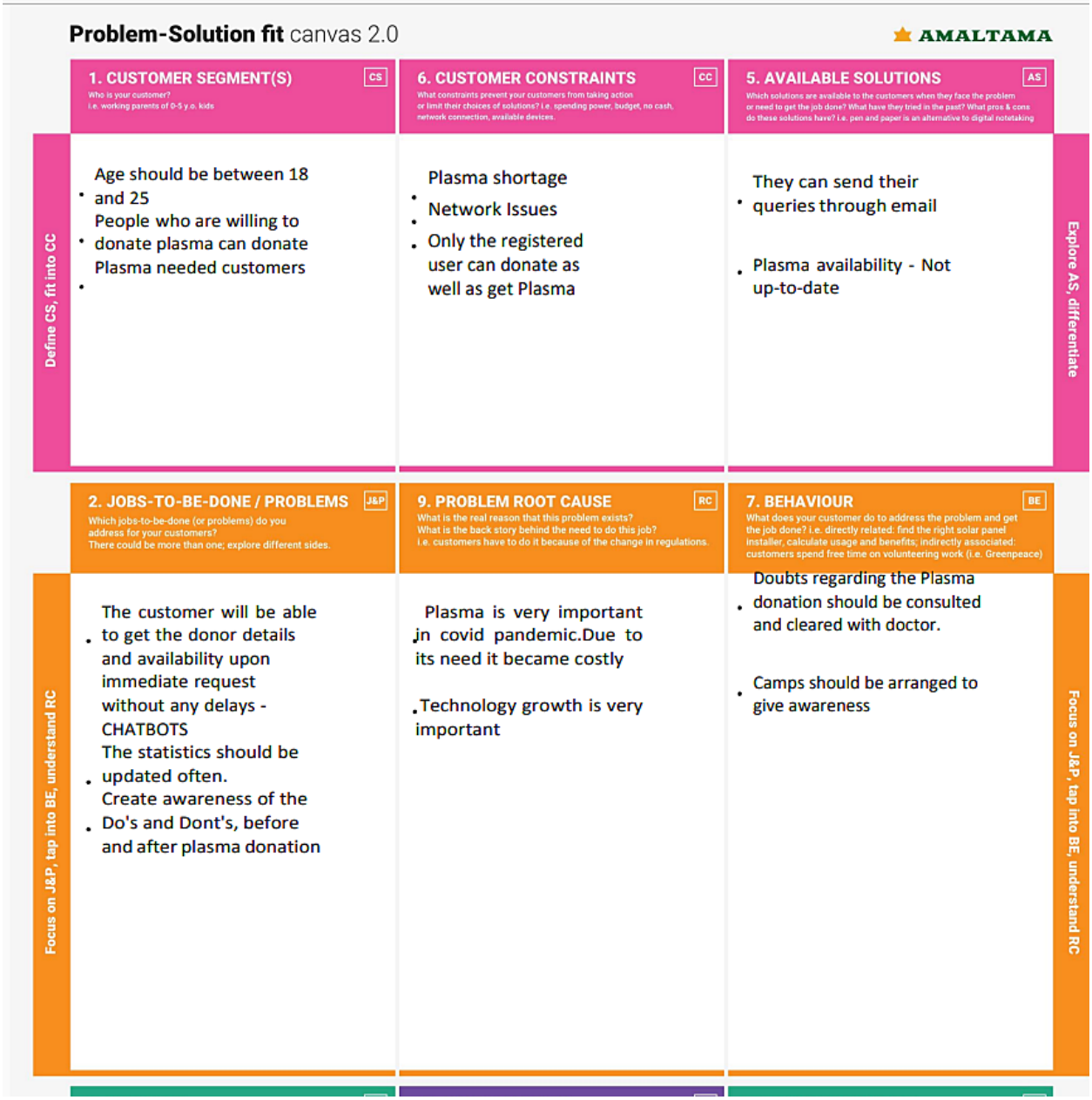
3.3 Proposed Solution

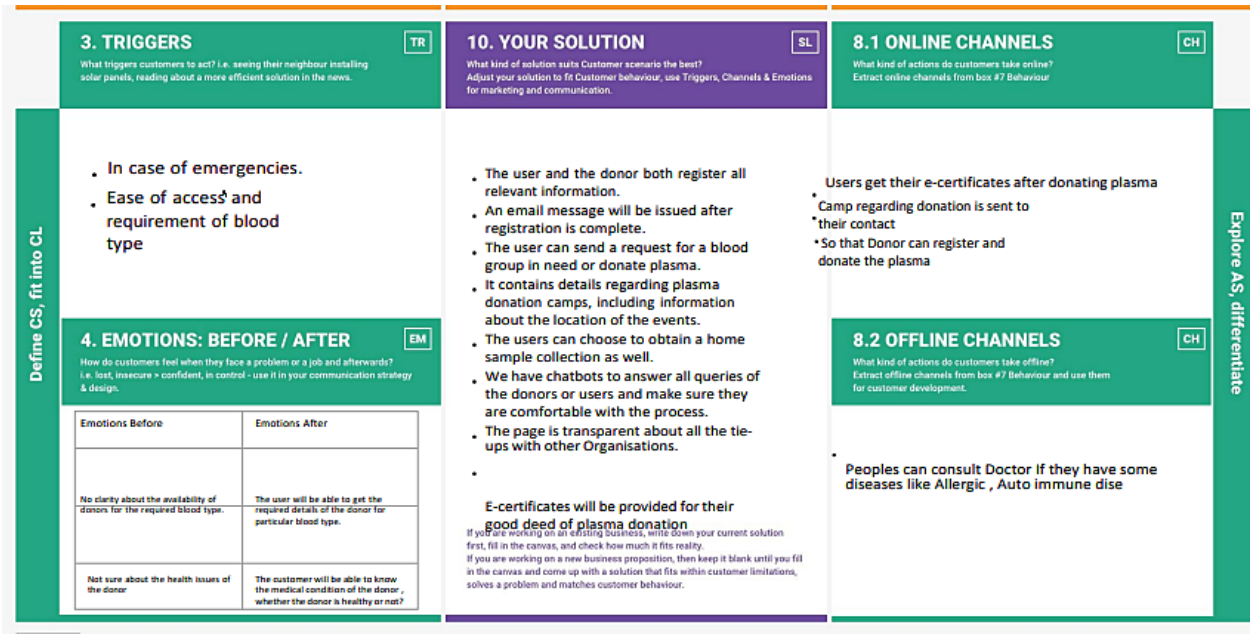
S.No.	Parameter	Description
1.	Problem Statement (That need to be solved)	The need for plasmaincreased, During the COVID 19 crisis, while the number of donors has decreased. Plasma is necessary for the survival of people without cancer, rare disorders,immunological problems, and genetic anomalies. Every blood bank claims to be out of blood, so we need to make people aware of the issue and offer support. Numerous camps, seminars, and applications can be of great help.
2.	Solution description	Plasma donor is an application which will make things easier and efficient at emergency and to solve our problem statement. Some of the features are : ● The user and the donor both register their full relevant information. ● An email message will be issued after registration is complete. ● The user has the option of sending a request for a blood group in need or donating plasma in this. ● It contains details regarding plasma donation camps, including information about the location of the events. ● The users can also able to know their necessary group of blood in their area.
3.	Uniqueness	A visual representation that is simple for users to understand will be used to display the statistics for the blood group availability data for plasma donation. The user can send a request for plasma if they are unsure about its availability in their immediate vicinity. Whether plasma is in short supply or is more readily available, users will receive an email notification within a short period of time. If individuals sign up for our application for plasma donors and decide they want to donate plasma, they can schedule an appointment. They will obtain their e-certification for donating plasma once they have completed their

		session according to schedule.These are the innovative elements included in this.
4.	Benefits user can obtain	Despite the apparent abundance of resources, there are still cases where hospitals or blood banks run out of essential resources, such as specific blood type shortages. One of the major issues health facilities run into is the shortage of certain blood types. An additional problem is facilities need access to patient data as quickly as possible before beginning patient blood transfer. This application, along with all the services it provides, also helps to eradicate certain spam messages and mails circulating around regarding fake or already satisfied blood emergency situations . A single platform for maintaining all genuine blood related activities and information increases the trust of the public to get involved in these activities, and to participate in blood donations.
5.	Business Model	An unpaid application exists for plasma donors. It is readily available and accessible by all. Due to the difficulty in locating donors who match a certain blood group, this application enables users to register people who wish to donate plasma and keep their information in a database. By informing the current donors of the need, saving the donor information would assist. The need for plasma increased significantly during the COVID 19 crisis, and the number of donors is limited. In the end, working with the government can use an app to aid those in need of plasma.
6.	Scalability of the Solution	This application assist users in finding the closest blood centre, knowing their eligibility to donate blood, receiving notifications when an urgent blood donation call comes in, and scheduling a convenient appointment utilising temporal and/or spatial information. A current donor profile will be used, containing details such as the donor's present location, blood type, and the date of their most recent donation, among other things. The right donors will be cleverly informed of the demand for blood donations, making it easier to locate a local suitable donor at the

		right time

3.4 Problem Solution Fit





4.Requirement Analysis

4.1 Functional Requirements

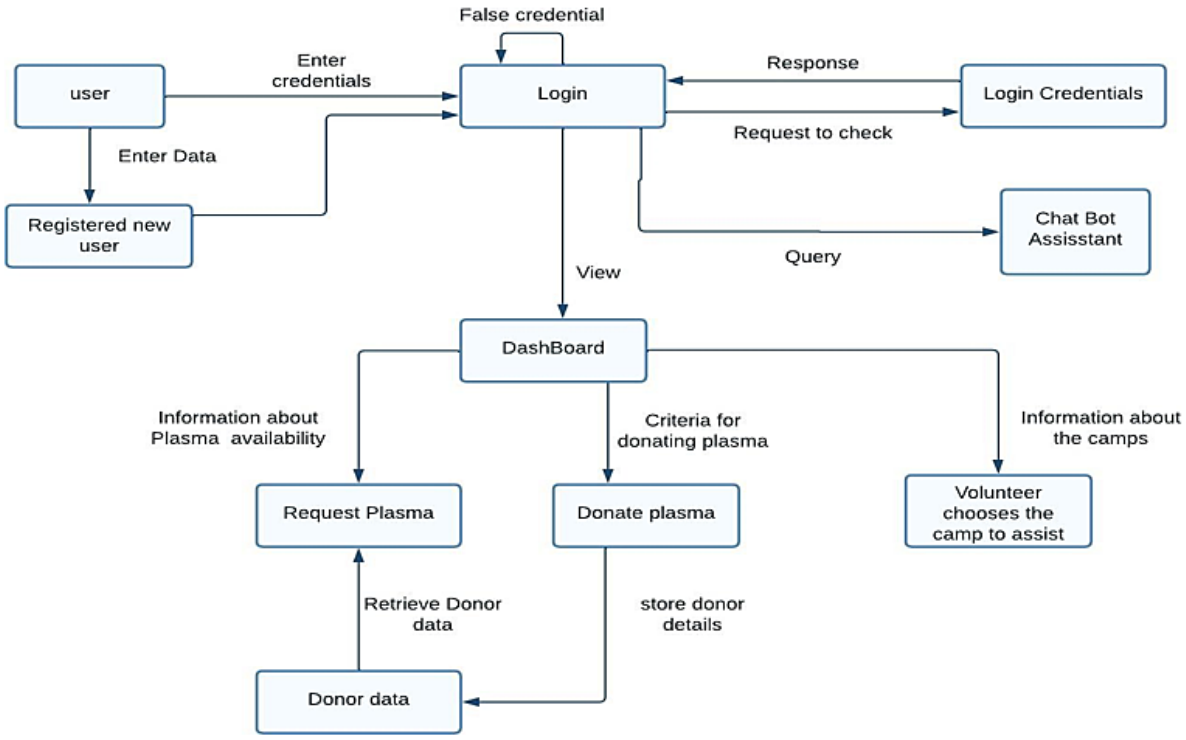
FR No	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form (WebApp)
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Certification	E certificate should be provided to the user to motivate them.
FR-4	Availability	The availability of plasma is given in the page as stats, which will be helpful for the users
FR-5	Plasma Request	Users can request to donate plasma by filling out the request form on the page.
FR-6	Searching requirements	Once the request is submitted, they will get an email Users can use the search bar to look up information
FR-7	Virtual Assistants	Virtual Assistants plays a vital role in clearing the doubts of the user .It surely helps the users to donate plasma in a right way and the necessary people may get plasma with help of Virtual Assistants.

4.2 Non Functional Requirements

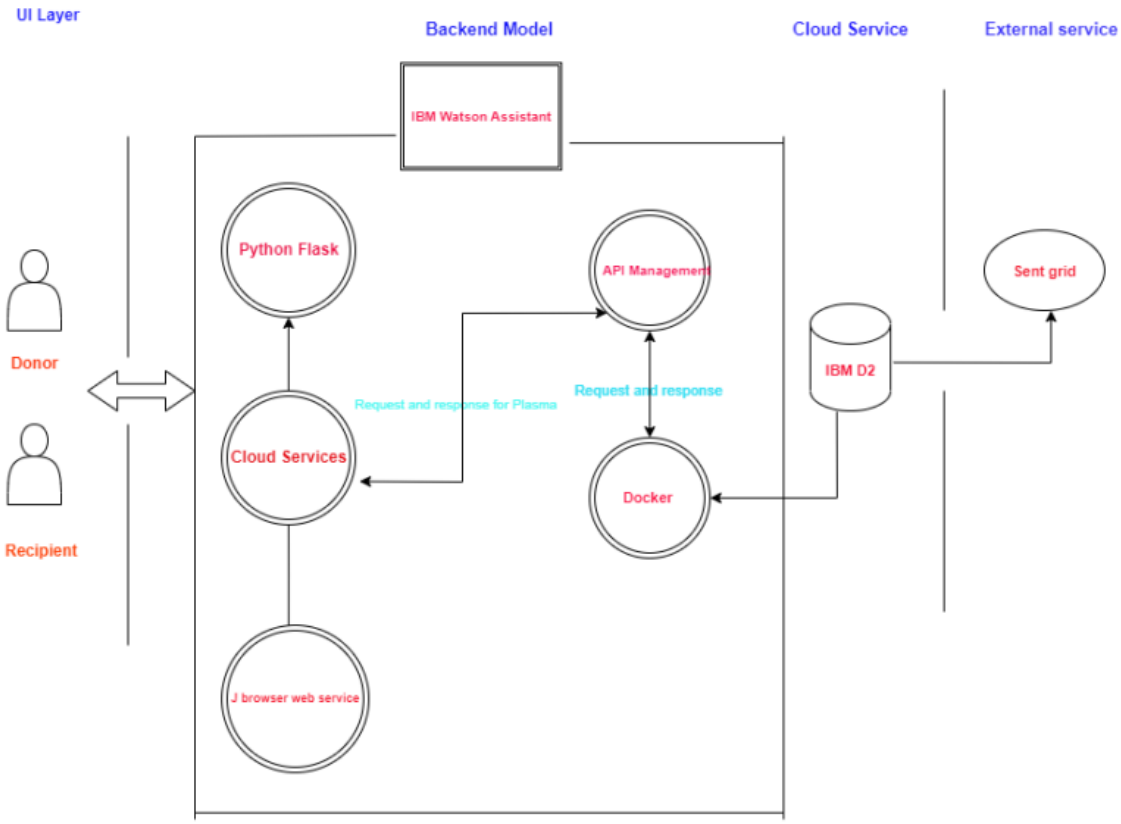
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Interface should be good.
NFR-2	Security	Safety measures such as firewall should be ensured
NFR-3	Reliability	Reliability is very important since user data is valuable one.
NFR-4	Performance	Usersshould have a proper Internet Connection.
NFR-5	Availability	The system including the online and offline components should be available 24/7.
NFR-6	Scalability	The system including the online and offline components should be available 24/7.

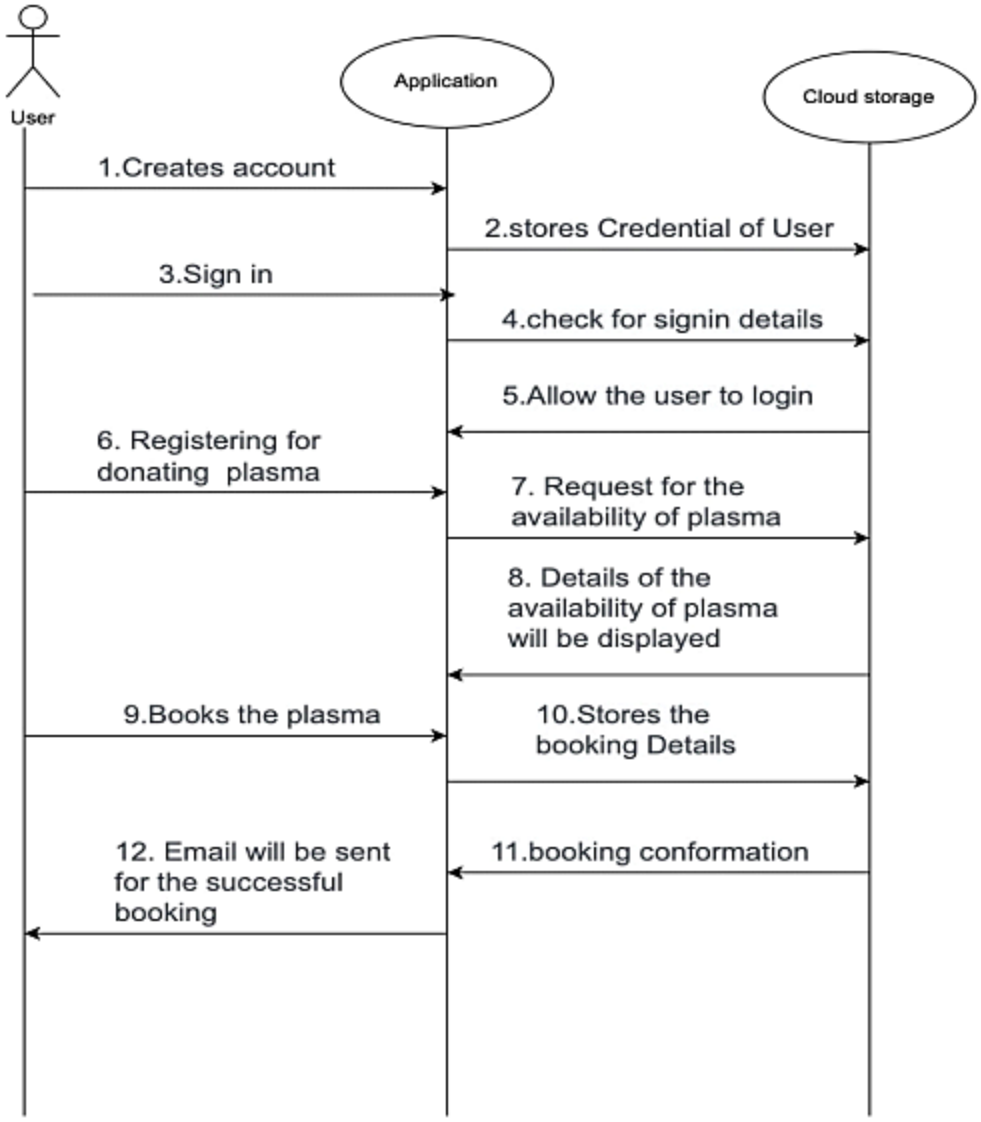
5.Project Design

5.1 Data Flow Diagrams



5.2 Solution and Technical Architecture





5.3 User Stories

User Type	Functional Requirement	User Story Numbe	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user	Registration Page	USN-1	User can register for the application by entering my email, password, and	Access the account/ dashboard	High	Sprint-1

			confirming my password			
		USN-2	User will receive confirmation email once I have registered for the application	Confirmation email & click confirm	High	Sprint-1
		USN-3	User can register for the application through Gmail	receive confirmation notifications through Gmai	Low	Sprint-1
	Login	USN-4	User can log into the application by entering email & password	Access into my User profile and view details in dashboard	High	Sprint-1
	Dashboard	USN-5	User can send the proper requests to donate and obtain plasma	Receive notifications through email	High	Sprint-2
Customer (Web user)	Login	USN-6	User can register and log into the application by entering email & password to view the profile	Access into my User profile and view details in dashboard questions	High	Sprint-2
	Dashboard	USN-7	User can register and log into the application by entering email & password to view the profile	Receive notifications through email	Low	Sprint-2
Customer Care Executive	Application	USN-8	A customer care executive,I can try to address user's concerns and questions	Receive notifications through email	Medium	Sprint-3
Administrator	Application	USN-9	Administrator can help with user-facing	User interface can	Medium	Sprint-2

			aspects of a website, like its appearance, navigation and use of media.	be modified		
		USN-10	Administrator can involve working with the technical side of websites.	Bugs and errors can be overcomed	Medium	Sprint-4
Chatbot	Dashboard	USN-11	In addition the Customer care Contact by the customer	I can reply to all Contact by the customer. queries related to the our application	Medium	Sprint-4

6.Project Planning and Scheduling

6.1 Sprint Planning and Estimation

User Type	Functional Requirement	User Story Number	User Story / Task	Story Points	Priority	Release	Team Members
Customer (Mobile user	Registration Page	USN-1	User can register for the application by entering my email, password, and confirming my password	5	High	Sprint-1	M K Aravindan, Seenu M, Deepak M, Sathish G
		USN-2	User will receive confirmation email once I have registered for the application	1	High	Sprint-1	M K Aravindan, Seenu M, Deepak M, Sathish G
		USN-3	User can register for the	2	Low	Sprint-1	M K Aravindan, Seenu M, Deepak M,

			application through Gmail				Sathish G
	Login	USN-4	User can log into the application by entering email & password	5	High	Sprint-1	M K Aravindan, Seenu M, Deepak M, Sathish G
	Dashboard	USN-5	User can send the proper requests to donate and obtain plasma	1	High	Sprint-2	M K Aravindan, Seenu M, Deepak M, Sathish G
Customer (Web user)	Login	USN-6	User can register and log into the application by entering email & password to view the profile	10	High	Sprint-2	M K Aravindan, Seenu M, Deepak M, Sathish G
	Dashboard	USN-7	User can register and log into the application by entering email & password to view the profile	10	Low	Sprint-2	M K Aravindan, Seenu M, Deepak M, Sathish G
Customer Care Executive	Application	USN-8	A customer care executive,I can try to address user's concerns and questions	20	Medium	Sprint-3	M K Aravindan, Seenu M, Deepak M, Sathish G
Administrator	Application	USN-9	Administrator can help with user-facing aspects of a website, like its appearance, navigation and use of media.	10	Medium	Sprint-2	M K Aravindan, Seenu M, Deepak M, Sathish G
		USN-10	Administrator can involve working with the technical side of websites.	10	Medium	Sprint-4	M K Aravindan, Seenu M, Deepak M, Sathish G
Chatbot	Dashboa	USN-	In addition the	10	Medi	Sprint-4	M K Aravindan,

	rd	11	Customer care Contact by the customer		um		Seenu M, Deepak M, Sathish G

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	12 Nov 2022

6.3 Reoports from JIRA

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)

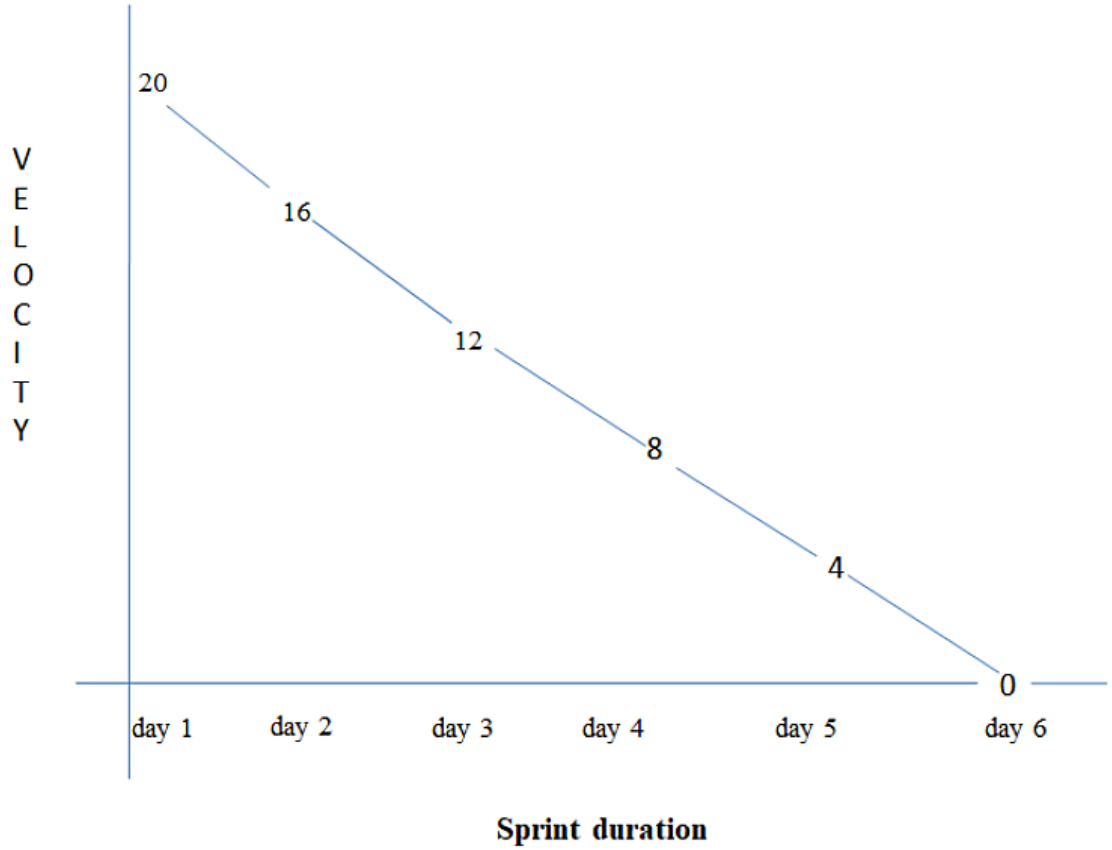
Sprint Duration=6 days Velocity of the team = 20 points

Average Velocity= Velocity/Sprint Duration

Av=20/6=3.34

Burndown Chart:

A burn down 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.



7.Coding and Solutioning

7.1 Feature

The following features can be added in the application in the future:

1.To add location the donor on request

2.Implement industry standards oauth protocols.

3.Requesting donor within the neighbouring location.

8.Testing

8.1 User Cases

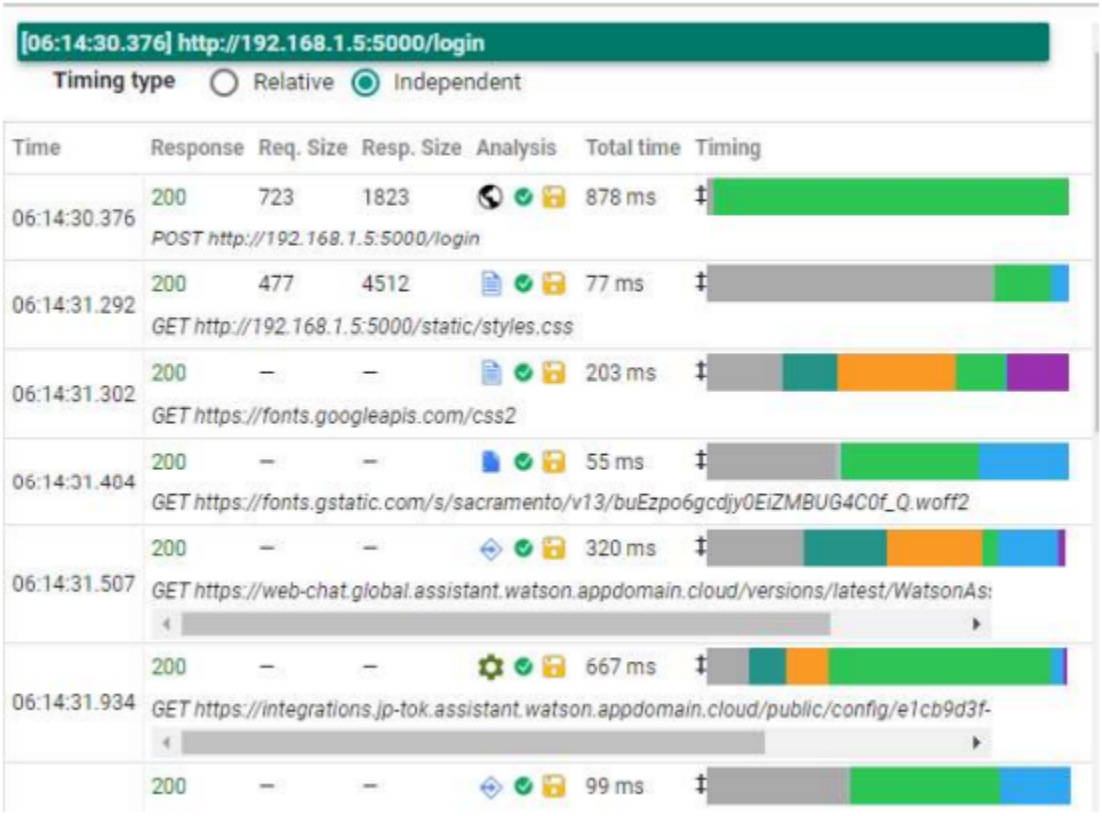
1. Verify if the buttons in web page are responsive Verifyif the UI elementsare getting displayed properly
2. Verify if the user can upload files from his system Verify if the outputis displayed
3. Verify if the user can login using his credentialsVerifyif the model predictsthe input accurately
4. Verify if the user is getting redirected to home page after sign in
5. Verify if the UI elements are being displayed
6. Verify if the user can navigateto other pages in navigation bar
7. Verify if the user can exit the home to sign page

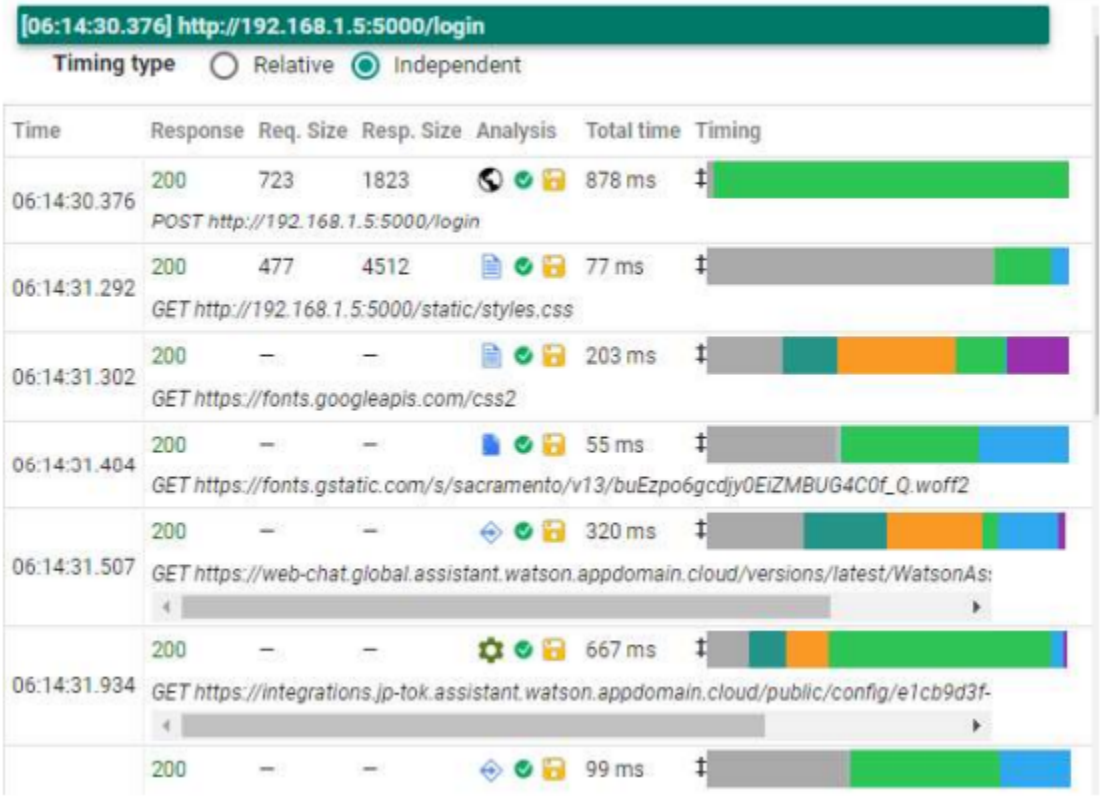
8.2 User Acceptance Testing

Purpose of Document The purpose of this document is to briefly explain the test coverage and open issues of the Skills/Job Recommender Application project at the time of the release to User Acceptance Testing (UAT).

9.Results

9.1 Performance Metrics


[06:14:30.376] http://192.168.1.5:5000/login
Timing type
Relative
Independent
Time
Response
Req. Size
Resp. Size
Analysis
Total time
Timing
06:14:30.376
200
723
1823
878 ms
POST http://192.168.1.5:5000/login
06:14:31.292
200
477
4512
77 ms
GET http://192.168.1.5:5000/static/styles.css
06:14:31.302
200
–
–
203 ms
GET https://fonts.googleapis.com/css2
06:14:31.404
200
–
–
55 ms
GET https://fonts.gstatic.com/s/sacramento/v13/buEzpo6gcdjy0EiZMBUG4C0f_Q.woff2
06:14:31.507
200
–
–
320 ms
GET https://web-chat.global.assistant.watson.appdomain.cloud/versions/latest/WatsonAs
06:14:31.934
200
–
–
667 ms
GET https://integrations.jp-tok.assistant.watson.appdomain.cloud/public/config/e1cb9d3f-
200
–
–
99 ms


[06:14:30.376] http://192.168.1.5:5000/login
Timing type
Relative
Independent
Time
Response
Req. Size
Resp. Size
Analysis
Total time
Timing
06:14:30.376
200
723
1823
878 ms
POST http://192.168.1.5:5000/login
06:14:31.292
200
477
4512
77 ms
GET http://192.168.1.5:5000/static/styles.css
06:14:31.302
200
–
–
203 ms
GET https://fonts.googleapis.com/css2
06:14:31.404
200
–
–
55 ms
GET https://fonts.gstatic.com/s/sacramento/v13/buEzpo6gcdjy0EiZMBUG4C0f_Q.woff2
06:14:31.507
200
–
–
320 ms
GET https://web-chat.global.assistant.watson.appdomain.cloud/versions/latest/WatsonAs
06:14:31.934
200
–
–
667 ms
GET https://integrations.jp-tok.assistant.watson.appdomain.cloud/public/config/e1cb9d3f-
200
–
–
99 ms

10.Advantages and Disadvantages

Advantages

- The main advantage of a Plasma bank management system is easy and effective information retrieval. Hence, the staff can view precise information quickly.
- The staff can now store all the details in thePlasma bank management system. Therefore, they can get rid of the manual procedures.

Disadvantages

- Manual document and data entry.
- Only web based system is available no mobile based system available.
- No proper coordination between different Applications and Users.
- Cannot Upload and Download the latest updates at right time.

11.Conclusion

We had completed the Plasma Donor Application using Html,Css in Frontend and We had used Python Flask Framework, Ibm Db2 as a Database,Container using Docker

12.Future Scope

The scope of a system means that which modules are being covered by the system. The scope clearly defines the boundaries of the proposed system. The functional areas of this application that lies under the scope of the proposed system are the management of the availability of donors, hospitals, Plasma banks to the user or member at any time. The system calculates the estimated locations of the donors, hospitals and blood banks and also provides online chat service between donors and consumers.

The client can also go through from the guidelines section to view the useful precautions needed before and after blood transfusion. To be a member of the Android Plasma Bank has to fill the registration form and provide the necessary information.

13.Appendix

12. FUTURE SCOPE
13. APPENDIX

Source Code
GitHub & Project Demo Link

Source code

```
from flask import Flask, render_template, request, redirect,
url_for, session

import ibm_db

import sendgrid
import os
from sendgrid.helpers.mail import *

SENDGRID_Key =
'SG.eRyF8WCWS4qWNeus8yIBMw.QtnTdBv5554HN45UpvjwhyineOr9zC91ya
ax6yH7_Lk'

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b1bc1829-6f45-
4cd4-bef4-
10cf081900bf.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;P
ORT=32304;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRoot
CA.crt;UID=zkd09477;PWD=KaR6irUnQiHwt8w1",'','')

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

@app.route("/",methods=['GET'])
```

```
def home():
  if 'email' not in session:
    return redirect(url_for('login'))
  return render_template('home.html',name='Home')



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


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


    if not email or not username or not password:
      return render_template('register.html',error='Please
fill all fields')


    #hash=bcrypt.hashpw(password.encode('utf-
8'),bcrypt.gensalt())


    query = "SELECT * FROM USERS WHERE Email=?"
    stmt = ibm_db.prepare(conn, query)
    ibm_db.bind_param(stmt,1,email)
    ibm_db.execute(stmt)
    isUser = ibm_db.fetch_assoc(stmt)
```

```
    if not isUser:
        insert_sql = "INSERT INTO Users(Name,email,PASSWORD)
VALUES (?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, username)
        ibm_db.bind_param(prep_stmt, 2, email)
        ibm_db.bind_param(prep_stmt, 3, password)
        ibm_db.execute(prep_stmt)
        return render_template('register.html',success="You can
login")
    else:
        return render_template('register.html',error='Invalid
Credentials')

  return render_template('register.html',name='Home')

@app.route("/login",methods=['GET','POST'])
def login():
  if request.method == 'POST':
    email = request.form['email']
    password = request.form['password']

    if not email or not password:
        return render_template('login.html',error='Please fill
all fields')
    query = "SELECT * FROM USERS WHERE Email=?"
    stmt = ibm_db.prepare(conn, query)
    ibm_db.bind_param(stmt,1,email)
```

```
    ibm_db.execute(stmt)
    isUser = ibm_db.fetch_assoc(stmt)
    print(isUser,password)

    if not isUser:
      return render_template('login.html',error='Invalid
Credentials')

    #isPasswordMatch = bcrypt.checkpw(password.encode('utf-
8'),isUser['PASSWORD'].encode('utf-8'))

    #if not isPasswordMatch:
    if(isUser['PASSWORD']!=password):
      return render_template('login.html',error='Invalid
Credentials')

    session['email'] = isUser['EMAIL']
    return redirect(url_for('home'))

  return render_template('login.html',name='Home')


@app.route('/logout')
def logout():
  session.pop('email', None)
  return redirect(url_for('login'))


sg =
```

```
sendgrid.SendGridAPIClient('SG.eRyF8WCWS4qWNeus8yIBMw.QtnTdBv
5554HN45UpvjwhyineOr9zC91yaax6yH7_Lk')
from_email = Email("plasmadonoribm@gmail.com")


@app.route('/request',methods=['GET','POST'])
def req():
  if request.method == 'GET':
    return render_template('request.html',name='request')
  email = request.form['email']
  name = request.form['Name']
  phone = request.form['phone']
  BloodGroupReq = request.form['BloodGroupReq']
  ADDRESS = request.form['ADDRESS']
  #to_email = To(email)
  print(email,name,phone,BloodGroupReq,ADDRESS)
  query = "SELECT * FROM DONORS WHERE BloodGroup=?"
  stmt = ibm_db.prepare(conn, query)
  ibm_db.bind_param(stmt,1,BloodGroupReq)
  ibm_db.execute(stmt)
  ll = ibm_db.fetch_assoc(stmt)
  if(ll):
    listt = []
    while(ll!=False):
      listt.append(ll)
      ll = ibm_db.fetch_assoc(stmt)
    print(listt)

    for i in listt:
```

```
        to_email = To(i['EMAIL'])
        subject = "REQUEST FOR BLOOD DONATION"
        content = Content("text/plain", "{} requests plasma
donation and has the same blood group {} as you.\nIf you wish
to really donate the blood, please contact them at the email
{}.\nThank you.".format(name,BloodGroupReq,email))
        mail = Mail(from_email, to_email, subject, content)
        response =
sg.client.mail.send.post(request_body=mail.get())
        print(response.status_code)
        print(response.body)
        print(response.headers)

    return
render_template('reqReplyS.html',name='reqReplyS',total=len(l
istt))
  else:
    return render_template('reqReplyF.html',name='reqReplyF')



@app.route('/donate',methods=['GET','POST'])
def donate():
  if request.method == 'GET':
    return render_template('donate.html',name='donate')
  email = request.form['email']
  name = request.form['Name']
  phone = request.form['phone']
  BloodGroup = request.form['BloodGroup']
  ADDRESS = request.form['ADDRESS']
```

```
  print(email,name,phone,BloodGroup,ADDRESS)
  insert_sql = "INSERT INTO
DONORS(Name,email,Number,BloodGroup,ADDRESS) 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, phone)
  ibm_db.bind_param(prep_stmt, 4, BloodGroup)
  ibm_db.bind_param(prep_stmt, 5, ADDRESS)
  ibm_db.execute(prep_stmt)
  return render_template('donSuccess.html',name='donSuccess')

@app.route('/stats',methods=['GET','POST'])
def stats():
  if request.method == 'GET':
    return render_template('stats.html',total=0,flag=1)
  email = request.form['email']
  query = "SELECT * FROM DONORS WHERE email=?"
  stmt = ibm_db.prepare(conn, query)
  ibm_db.bind_param(stmt,1,email)
  ibm_db.execute(stmt)
  ll = ibm_db.fetch_assoc(stmt)
  listt = []
  if(ll):
    while(ll!=False):
      listt.append(ll)
      ll = ibm_db.fetch_assoc(stmt)
    print(listt)
```

```
  return render_template('stats.html',total=len(listt),flag=0)

if __name__ == "__main__":
  app.run(host="0.0.0.0")
```

Github link

https://github.com/IBM-EPBL/IBM-Project-46059-1660735615

Demo link

https://drive.google.com/file/d/11u0TPEYCXaFQ8kbSiYifko7vxzNEHCGW/view?usp=drivesdk