

## **Final Report**

### **PLASMA DONOR APPLICATION**

Logapradeep S

Melwin vino A

Attotasaikeerthana

Kannan K

## Project Report Format

1. INTRODUCTION
  - 1.1 Project Overview
  - 1.2 Purpose
2. LITERATURE SURVEY
  - 2.1 Existing problem
  - 2.2 References
  - 2.3 Problem Statement Definition
3. IDEATION & PROPOSED SOLUTION
  - 3.1 Empathy Map Canvas
  - 3.2 Ideation & Brainstorming
  - 3.3 Proposed Solution
  - 3.4 Problem Solution fit
4. REQUIREMENT ANALYSIS
  - 4.1 Functional requirement
  - 4.2 Non-Functional requirements
5. PROJECT DESIGN
  - 5.1 Data Flow Diagrams
  - 5.2 Solution & Technical Architecture
  - 5.3 User Stories
6. PROJECT PLANNING & SCHEDULING
  - 6.1 Sprint Planning & Estimation
  - 6.2 Sprint Delivery Schedule
  - 6.3 Reports from JIRA
7. CODING & SOLUTIONING (EXPLAIN THE FEATURES ADDED IN THE PROJECT ALONG WITH CODE)
  - 7.1 Layer between Donor and Recipient
  - 7.2 Chat Bot Integration
  - 7.3 SendGrid Output
  - 7.4 Database Schema (if Applicable)
8. TESTING
  - 8.1 Test Cases
  - 8.2 User Acceptance Testing
9. RESULTS
  - 9.1 Performance Metrics

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.

## 1.2 Purpose:

The main purpose of the proposed system, the donor can donate the plasma to the blood bank, the blood bank can apply for the donor and once the donor has accepted the request, the blood bank can add the units they need and the admin can also send the request to the blood bank that urgently needs the plasma for the patient and can take the plasma from the blood bank.

# 2. LITERATURE SURVEY

## 2.1 Existing problem

There are many people who are willing to donate plasma and who need plasma. But there is not any accessible way to help them to find plasma donation centers in real- time. So, the problem is not the lack of donors, but finding the right sponsor at the right time. If someone needs plasma, they seek plasma first from family members, then from hospitals and the nearest plasma bank. If they can't process plasma in these ways, it's very difficult for them to contact another for a short-term plasma draw. This is a problem that I want to solve through this application. Instead of just providing plasma to people in need with an outdated list of regular plasma donors who may or may not be available to help, This application reaches the right people the moment users find Out.

## 2.2 References

S NO	TITLE	AUTHORS	ABSTRACT	DRAWBACKS
------	-------	---------	----------	-----------

1	Instant Plasma Donor Recipient Connector Web Application	Aishwarya R Gowri, Jain University Department of MCA, computer science	<p>A plasma is a liquid portion of the blood, over 55% of human blood is plasma. Plasma is used to treat various infectious diseases and it is one of the oldest methods known as plasma therapy. Plasma therapy is a process where blood is donated by recovered patients in order to establish antibodies that fights the infection.</p> <p>In this project plasma donor application is being developed by using AWS services. The services used are AWS Lambda, API gateway, DynamoDB, AWS Elastic Compute Cloud with the help of these AWS services, it eliminates the need of configuring the servers and reduces the infrastructural costs associated with it and helps to achieve serverless computing. Situations like if the donor count is very low, it is very important to get the information about the plasma donors. Saving the donor information and notifying about the current donors would be a helping hand as it can save time and help the users to track down the necessary information about the donors.</p>	<p>It cannot auto verify user genuineness.</p> <p>It requires an active internet connection.</p>
2	Plasma Donation App	Jenny Shersten	Motivation for further plasma collection from donors for recipients, as well as fast communication with them. For both groups - always upto-date information and the ability to	Reports are not verified

			follow statistics and data in the city and in the country	
--	--	--	---	--

3	Plasma Donation Website using MERN stack	Neha Soni , Software Engineering Intern at FICO   Technical Blogger	<p>The person who wants to donate his/her plasma needs to register in our application providing required information which are name, age, blood group, phone number, and location, etc.</p> <p>Patients who need plasma can also fill the form to request the plasma. Patients can directly call the donor by taking his/her contact number from the application. The user can also view the total active cases, recovered cases, vaccine centres in their area, hospital location, and helpline number.</p>	Auto-Verification: It cannot automatically verify the genuine users.
4	Instant Plasma Donor Recipient Connector Web Application	Ripathi S, Kumar V, Prabhakar A.	<p>The world is suffering from COVID 19 crisis, and we haven't found any vaccine yet. But there is another scientific way from which we can help to lower the death ratio or help the COVID 19 affected person is by donating Plasma from recovered patients. With no approved antiviral treatment plan for the deadly COVID19 infection, plasma therapy is an experimental approach to treat COVID positive patients and help them recover faster. The therapy considered to be safe and promising. If a particular person is fully recovered from COVID 19 he/she is applicable to donate their plasma. In the proposed system, donors who need to donate plasma can donate by uploading covid-19 certificate and blood bank can view donors and can raise requests to donors and the</p>	<p>Tedious work.</p> <p>Expensive.</p> <p>Requires more manpower</p> <p>Time Consuming.</p>

			hospital can register/login and can search for plasma, they can raise requests to blood bank and can get the plasma.	
--	--	--	--	--

5	Plasma-Donor-App	Dheeraj Kotwani, Pragathi Verma , Sitam Sardar, Vatsal Kesarwani Nakul Sharma Nuh Koca Harsh Rajgor	An Open-Source App which fills the gap between the patients and the Plasma Donors.	No search filter available  Cannot login through Chrome  UI improvement in Login page
---	------------------	---	--	---

### 2.3 Problem Statement Definition

This system 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. Similar to blood donors there also exist plasma donors where there

exists problems like in case of emergency needs the most important life saver necessity is plasma.

### 3. IDEATION & PROPOSED SOLUTION

#### 3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

Empathy Map:




#### 3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich number of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

## Step-1: Team Gathering, Collaboration and Select the Problem Statement

Template



### Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 10 minutes to prepare
- 1 hour to collaborate
- 2-8 people recommended

**Before you collaborate**

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

A

**Team Members:**

- Keralembale Dinesh
- Munegasser Maheshbabu
- Malavansu Rajitha
- Vushtulamuri Bhoomika

B

**Set the goal**

To develop an plasma donor application that enables user to join as a donor anywhere anytime and the information can be used whenever necessary situation arises .

C

**Tools used for development**

- DB2
- SendGrid

Open article →

- Docker
- Kubernetes
- Flask

**1 Define your problem statement**

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 helping to the problem faced, an application is to be built which would take the donor details, store them and inform them upon a request

**Process**

To develop an plasma donor application that enables user to join as a donor anywhere anytime and the information can be used whenever necessary situation arises .

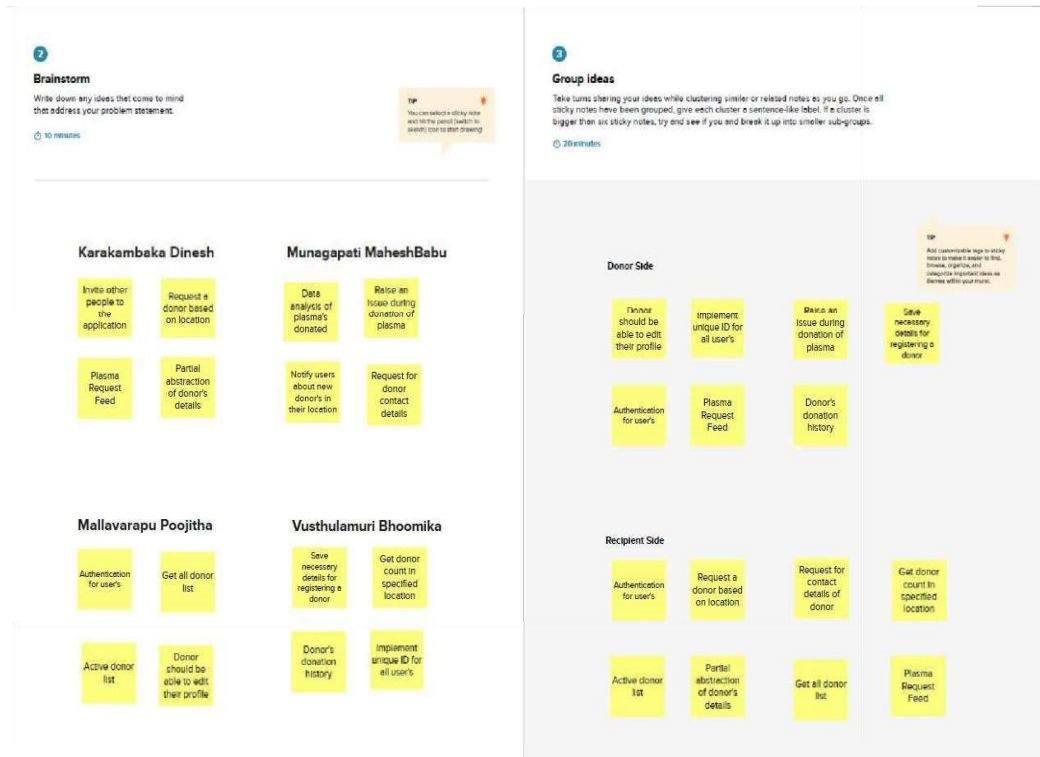
**Key rules of brainstorming**

To run an smooth and productive session

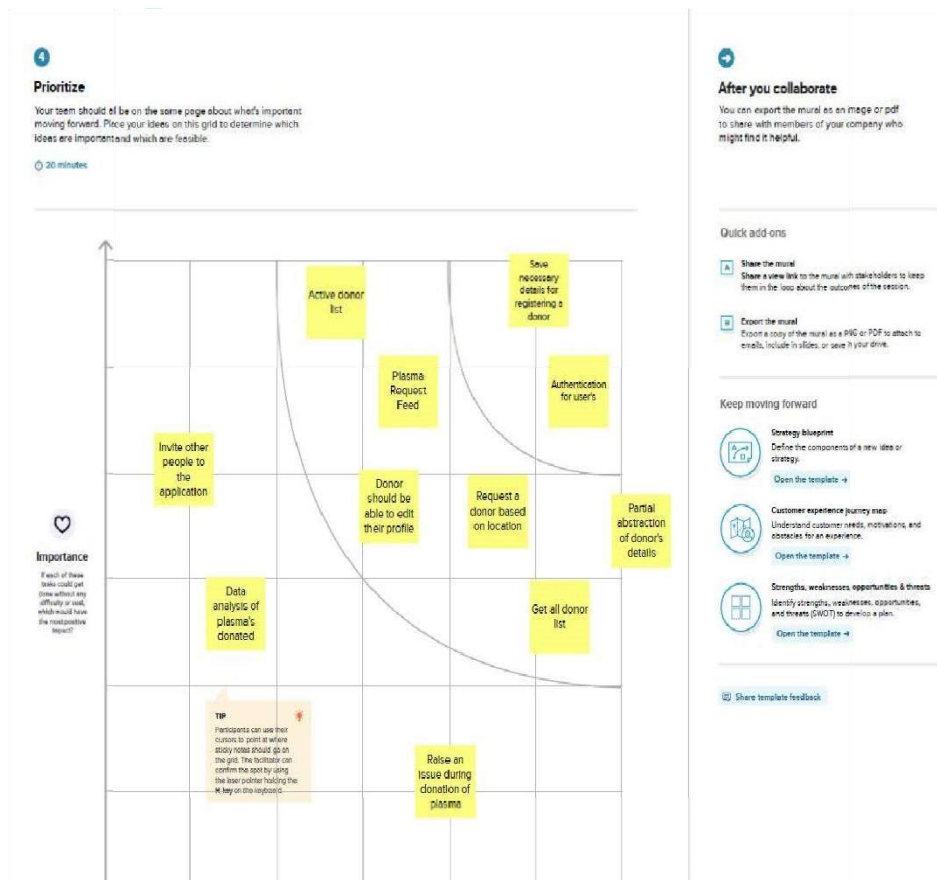
- Stay in topic.
- Defer judgment.
- Go for volume.
- Encourage wild ideas.
- Listen to others.
- If possible, be visual.

## Step-2: Brainstorm, Idea Listing and Grouping





## Step-3: Idea Prioritization



### 3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement	To meet the high demand for plasma in emergency situations and to make it possible of availability of plasma to the utmost.
2.	Solution description	The details of the donor are stored in the application database and when there is a need for plasma, the donors are get notified and also the needy is also informed about the availability of the plasma.
3.	Uniqueness	Faster Reachability, Simple to use, Quick Access.
4.	Customer Satisfaction	The end users can immediately come to know the availability of plasma when they expose their request and this ensures the immediate response to the needer.
5.	Business Model (Revenue Model)	This is completely a service based applications and meets the emergency needs of the people.
6.	Scalability of the Solution	When there is an emergency need the application finds the donor with the parameters like nearby location, availability and matching blood group.

### 3.4 Problem Solution fit

Define CS, fit into CC	<b>1. CUSTOMER SEGMENT(S)</b> <span>CS</span> • People who need plasma and donate plasma. • Hospitals and clinics	<b>6. CUSTOMER CONSTRAINTS</b> <span>CC</span> • Unavailability of plasma. • Availability of plasma types. • Donors within the nearest location.	<b>5. AVAILABLE SOLUTIONS</b> <span>AS</span> • Posting the situation in the social media like What's app, Instagram, Twitter etc. • The existing application used only collecting details of donors, but it does not notify them at the right time	Explore AS, differentiate
	<b>2. JOBS-TO-BE-DONE / PROBLEMS</b> <span>JRP</span> 1.Helps the needy or plasma seeker to find the donors available to their nearest location. 2. Provide a platform to volunteer donors to help the needy. Lack of information about the donors.. 3.Plasma demand and supply gap has grown even bigger	<b>9. PROBLEM ROOT CAUSE</b> <span>RC</span> 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.	<b>7. BEHAVIOUR</b> <span>BE</span> 1.The user/patient finds the right plasma donor application and interacts with the application. Registers by giving the details as a donor. 2. The database will have all the details and if a user posts a request then the concerned blood group donors will get notified about it. Calculate the usage and accuracy in finding the donor details.	
Focus on JRP, fit into BE, understand RC	<b>3. TRIGGERS</b> <span>TR</span> Advertising plan for blood donor app, that is we have to spread the need of plasma donor app over the specific hospitals which needs the plasma mostly and this will trigger people to use plasma web application. When they read our efficient solutions in the popular magazines and ne	<b>10. YOUR SOLUTION</b> <span>SL</span> Finding respective donors, alerting recipient via email when the plasma is available.	<b>8. CHANNELS OF BEHAVIOUR</b> <span>CH</span> The donor will register the details of his/her donation. And the user/patient will request on the application and the application will inform them as a response via mail.	Focus on JRP, fit into BE, understand RC
	<b>4. EMOTIONS: BEFORE / AFTER</b> <span>EM</span> Before: Confused, Anxious, Exhausted, Scared. Aft Relaxed, Motivated.			

## 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	Access Website	Should be able to access web applications on anydevice.
FR-2	User Registration	Acknowledgement sent to registered email id with the help of sendgrid.
FR-3	User Login	Login through the registered email id.
FR-4	Send Request	Users can request plasma and donors will be notified.

FR-5	Contact donor	Donor details will be provided to the recipient.
------	---------------	--

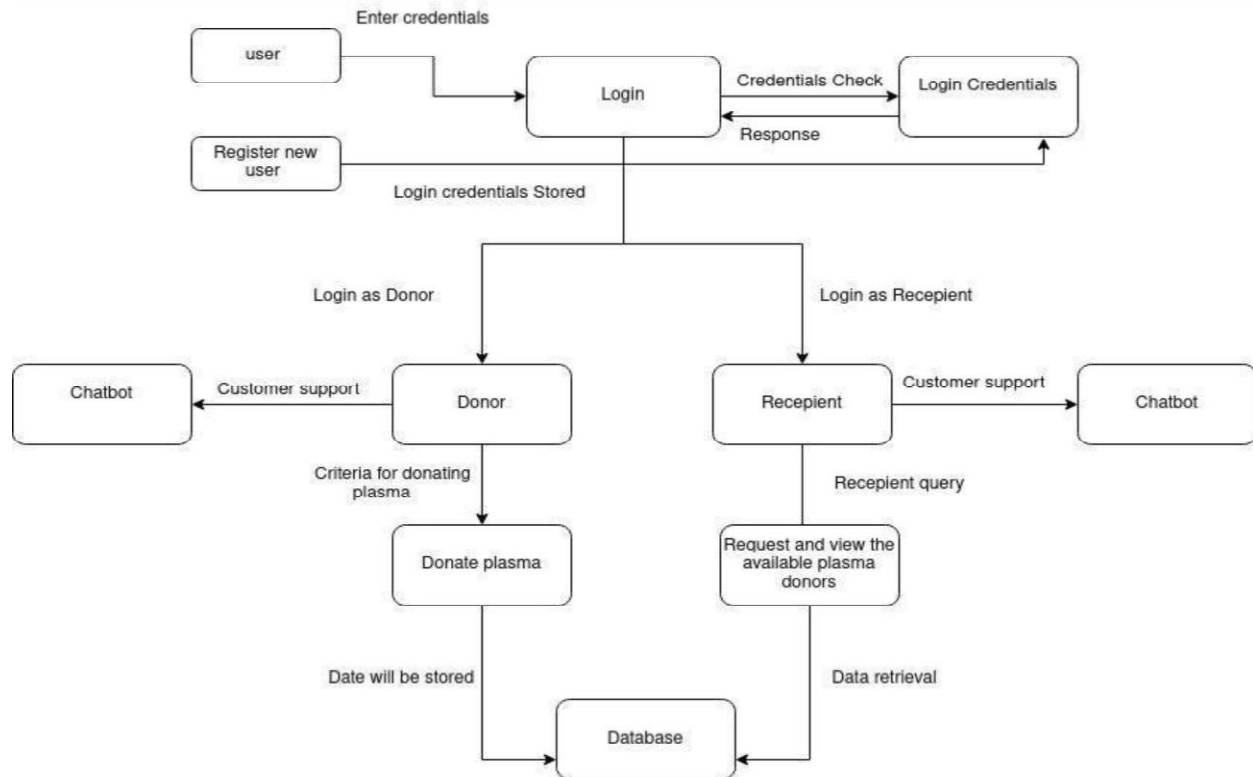
## 4.2 Non-Functional requirements

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

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	The Plasma donor application is user friendly and easy to access.
NFR-2	Security	The user/donor details are stored in cloud.
NFR-3	Reliability	The application have the ability to work all the time without failure .
NFR-4	Performance	The plasma donor application works well in all type of situations.
NFR-5	Availability	The plasma donor application is an 24/7 online web application.
NFR-6	Scalability	The application can be extended to provide plasma donor availability based on the recipient's location.

## 5. PROJECT DESIGN

### 5.1 Data Flow Diagrams



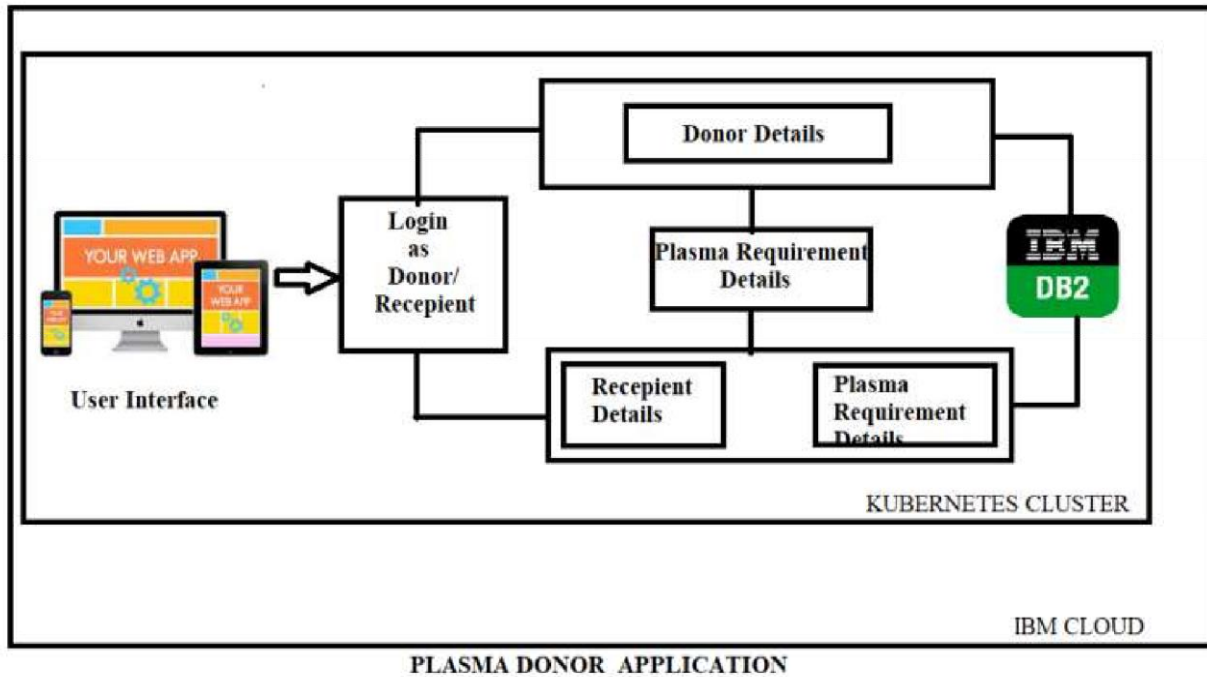
## 5.2 Solution & Technical Architecture

### Solution Architecture:

Solution architecture is a complex process \_ with many sub-processes that bridges the gap between business problems and technology solutions.

Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behaviour, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.



#### Technology Architecture:

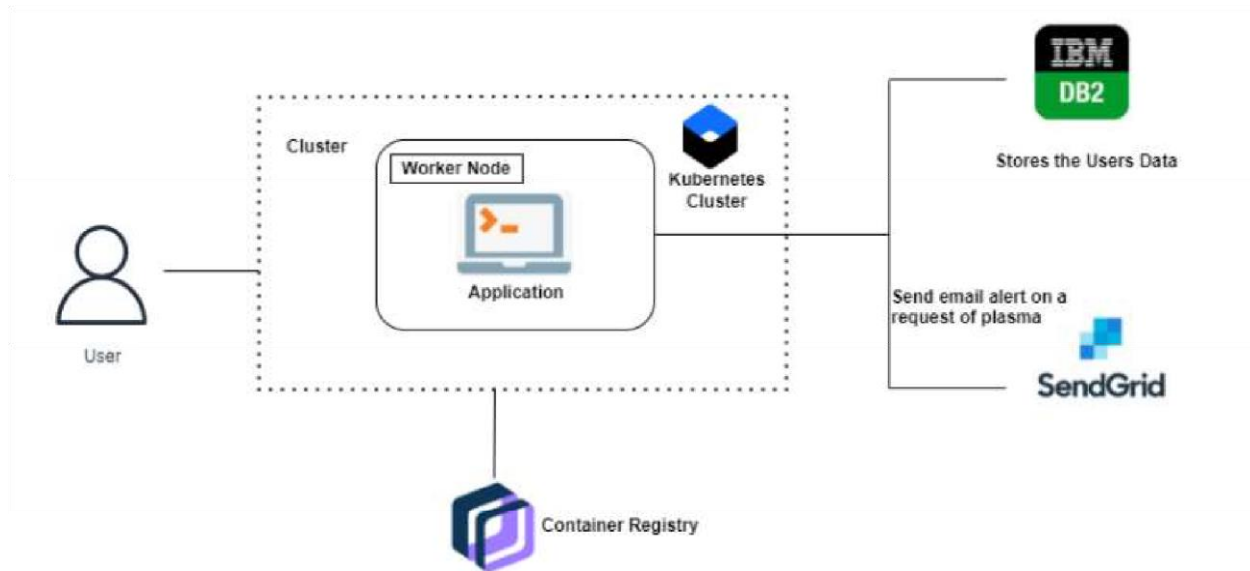


Table-1: Components & Technologies:

SNO	Component	Description	Technology
1.	User Interface	How user interacts with application e.g.Web UI, Mobile App, Chatbot etc.	HTML, CSS, JavaScript
2.	Application Logic	Logic for a process in the application	Python flask

3.	Cloud Database	Database Service on Cloud	IBM DB2
4.	File Storage	File storage requirements	IBM Block Storage
5.	External API	Trigger notification	Send grid

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Backend is built using flask and frontend usinghtml,Css,Javascript.	Html , Css , Javascript , Flask
2.	Security Implementations	Database storage and access would be limited.	IBM DB2
3.	Scalable Architecture	Docker containers allows multiple containers to bedeployed at the same time.	Docker
4.	Availability	Multiple kubernetes containers will be deployed.	Kubernetes
5.	Performance	Backend would be able to handle multiple clients	Flask

### 5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register as donor or recipient by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email	Medium	Sprint-4

Donor	Donor's Login	USN-3	As a donor, I can login into the donor's page	As a donor, I can access into my profile and donate plasma	High	Sprint-1
Recipient	Recipient's Login	USN-4	As a recipient, I can login into recipient page	As a recipient, I can access into my profile and request plasma	High	Sprint-1
Chat bot	Dashboard	USN-5	For the customer convenience, There is a chat bot for the queries	I can get the related queries from the bot	Medium	Sprint-2
Administrator	Administration	USN-6	As an administrator I can enumerate the user data and manage them	I can manage the data which provides data integrity to the user	High	Sprint-3

## 6. PROJECT PLANNING & SCHEDULING

### 6.1 Sprint Planning & Estimation Project

Tracker, Velocity & Burndown Chart:

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

Velocity:

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

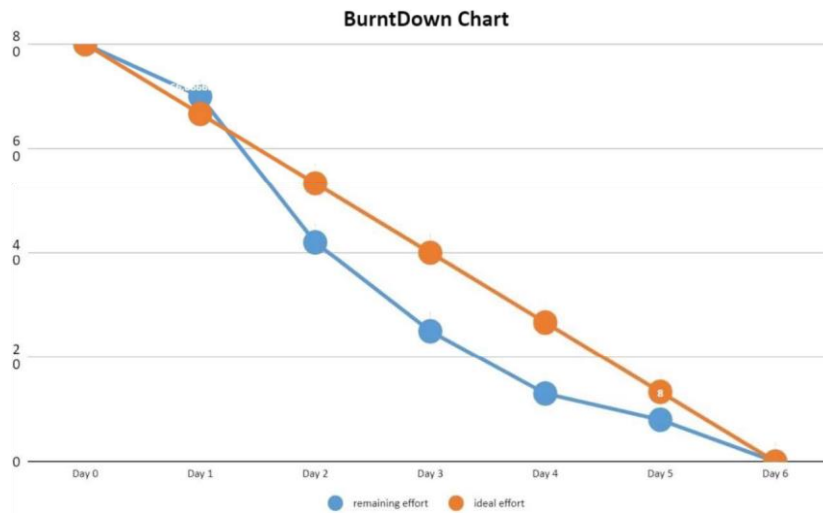
$$AV = \text{Sprint Duration} / \text{Velocity}$$



$$= 20/6 = 4$$

### 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.



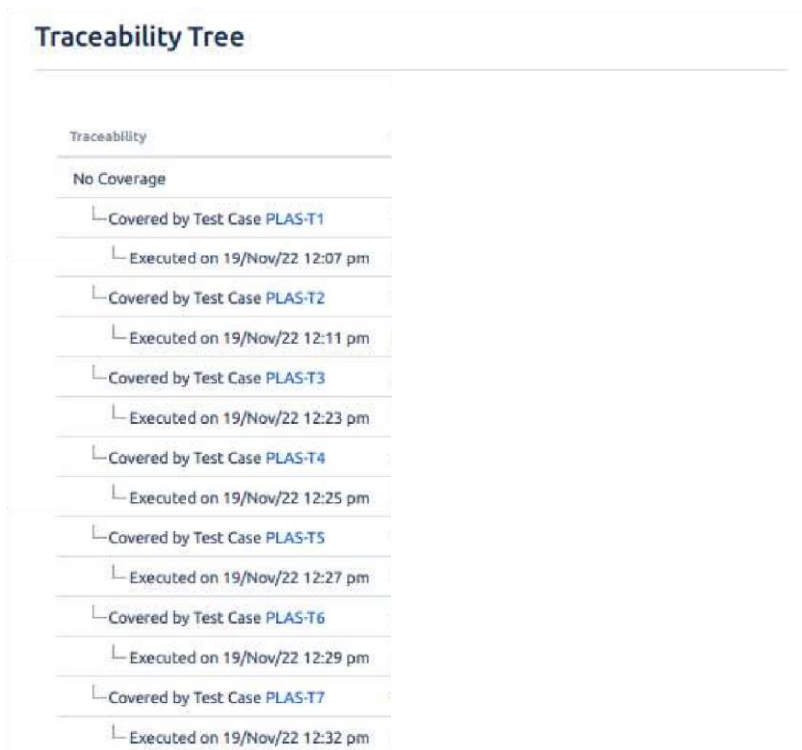
## 6.2 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register as donor or recipient by entering my email, password and confirming my password	10	High	Dinesh K
Sprint-4	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	20	Medium	Mahesh M

Sprint-1	Donor's Login	USN-3	As a donor, I can login into the donor's page	5	High	Poojitha M
Sprint-1	Recipient's Login	USN-4	As a recipient, I can login into recipient page	5	High	BhoomikaV
Sprint-1	Recipient's Login	USN-4	As a recipient, I can login into recipient page	5	High	Dinesh K
Sprint-3	Administration	USN-6	As an administrator I can enumerate the user data and manage them	20	High	Mahesh M

## 6.3 Reports from JIRA

### COVERAGE REPORT



## Coverage Report

Coverage	Test Cases
No Coverage	<div>7</div> <div>PLAS-T1 <span>APPROVED</span> Welcomepage_TC_001</div> <div>PLAS-T2 <span>APPROVED</span> Welcomepage_TC_002</div> <div>PLAS-T3 <span>APPROVED</span> about_page_TC_001</div> <div>PLAS-T4 <span>APPROVED</span> request_plasma_TC_001</div> <div>PLAS-T5 <span>APPROVED</span> Sign to speech_TC_002</div> <div>PLAS-T6 <span>APPROVED</span> donation_history_TC_003</div> <div>PLAS-T7 <span>APPROVED</span> sign_out_TC_001</div>

## Traceability Tree Report

Traceability Matrix Report

Traceability matrix

Coverage		Test Cases
No Coverage		PLAS-T1 - Welcomepage_TC_001
		PLAS-T2 - Welcomepage_TC_...
		PLAS-T3 - about_page_TC_001
		PLAS-T4 - request_plasma_TC...
		PLAS-T5 - Sign to speech_TC_...
		PLAS-T6 - donation_history_T...
		PLAS-T7 - sign_out_TC_001

Displaying (1 of 1)

Last test execution: Pass

Coverage	Test Cases
No Coverage <span>7</span>	PLAS-T1 <span>APPROVED</span> Welcomepage_TC_001 <span>1</span>
	PLAS-T2 <span>APPROVED</span> <span>1</span> Welcomepage_TC_002
	PLAS-T3 <span>APPROVED</span> <span>1</span> about_page_TC_001
	PLAS-T4 <span>APPROVED</span> <span>1</span> request_plasma_TC_001
	PLAS-T5 <span>APPROVED</span> <span>1</span> Sign to speech_TC_002
	PLAS-T6 <span>APPROVED</span> <span>1</span> donation_history_TC_003
	PLAS-T7 <span>APPROVED</span> <span>1</span> sign_out_TC_001

Traceability Report

## 7. CODING & SOLUTIONING

### 7.1 Layer between Donor and Recipient

The application serves as a layer between donor and the recipient, donor can expose their information to the application, and they will be notified via email if there's a request for plasma with the same blood group

### 7.2 Chat bot integration

The application has a chatbot integrated with it to help with basic user queries and to interact with the user. The chatbot feature is added to the application by using IBM Watson assistant in IBM cloud. This chatbot can interact with the user and guide them for simple queries.

## 7.4 Database Schema

The screenshot shows the IBM Db2 on Cloud interface. The top navigation bar includes 'Load Data', 'Load History', 'Tables' (selected), 'Views', 'Indexes', 'Aliases', 'MQTs', 'Sequences', and 'Application objects'. A search bar 'Find schemas or tables' is present. The left sidebar shows 'Schemas' and 'SQL' options. The main area is divided into two panels: 'Tables' and 'Table definition'.

**Tables Panel:**

Name	Schema	Properties
<input type="checkbox"/> DONORS	RSJ36993	...
<input type="checkbox"/> USERS	RSJ36993	...

Total: 2, selected: 0

**Table definition Panel:**

DONORS

No statistics available.

Name	Data type	Nullable	Length	Scale
NAME	VARCHAR	Y	256	0
EMAIL	VARCHAR	Y	256	0
NUMBER	VARCHAR	Y	256	0
BLOODG ROUP	VARCHAR	Y	256	0

View data

This is a duplicate of the screenshot above, showing the IBM Db2 on Cloud interface with the 'Tables' tab selected. The 'Table definition' panel for the 'DONORS' table is visible, showing columns: NAME, EMAIL, NUMBER, and BLOODG ROUP, all of type VARCHAR with a length of 256 and a scale of 0.

## 8. TESTING

### 8.1 Test Cases

- Verify if the buttons in web page are responsive
- Verify if the UI elements are getting displayed properly
- Verify if the user can upload files from his system
- Verify if the output is displayed
- Verify if the user can login using his credentials
- Verify if the model predicts the input accurately
- Verify if the user is getting redirected to home page after sign in
- Verify if the UI elements are being displayed
- Verify if the user can navigate to other pages in navigation bar
- 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).

### Defect Analysis

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

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	2	0	0	0	2
Duplicate	1	0	0	0	1
External	0	0	0	0	0
Fixed	3	0	0	0	3
Not Reproduced	2	0	0	0	2
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	8	0	0	0	8

### Test Case Analysis

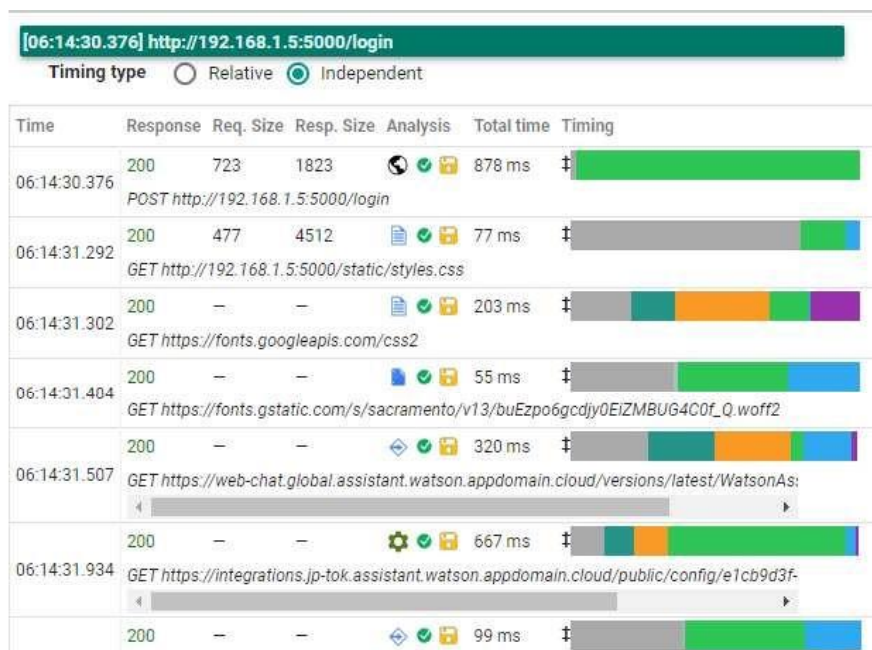
This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
---------	-------------	------------	------	------

Print Engine	0	0	0	0
Client Application	3	0	0	3
Security	0	0	0	0
Outsource Shipping	0	0	0	0
Exception Reporting	0	0	0	0
Final Report Output	0	0	0	0
Version Control	0	0	0	0

## 9. RESULTS

### 9.1 Performance Metrics







more than 3 million individuals have died as a result of the coronavirus. Aside from immunisation, there is another scientific approach for treating a covid infected individual and lowering the chance of mortality. This plasma treatment is an experimental strategy to treating and recovering corona-positive individuals. This plasma treatment is thought to be both safe and promising. 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 technique suggested here tries to connect donors and patients using an internet application. Users can use this application to make a request for plasma donation or a necessity. Both parties have the option to accept or reject the request. To donate plasma, the user must provide a Covid Negative report. If somebody need a Plasma Donor, this system is employed. Blood and plasma donation is a type of citizen's social duty in which a person can voluntarily donate blood/plasma using our app. This application was built with the idea of ensuring that donors contribute blood/plasma to the community. This approach is designed to be user-friendly so that anybody may access and manage his or her account. This application will disrupt the blood/plasma supply chain and assist the poor in finding free donors. This project will assist new blood and plasma banks in improving their services and transitioning from traditional to user-friendly frameworks.

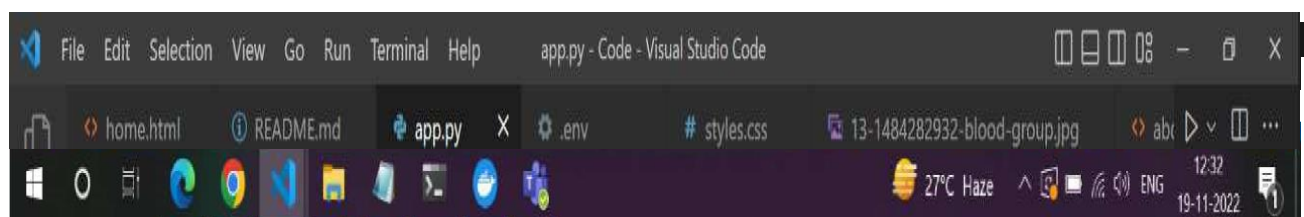
## 12. FUTURE SCOPE

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

- To add location of the donor on request
- Implement industry standards Oauth protocols
- Requesting donor within the neighbouring location

## 13. APPENDIX

Source Code App.py:



```

import os
import ibm_db
from flask_mail import Mail, Message

app.py > ...
1  from flask import Flask, render_template, request, redirect, url_for, session
2  from flask_mail import Mail, Message
3  import ibm_db
4  import os

conn= ibm_db.connect("DATABASE=bludb;HOSTNAME=764264db-9824-4b7c-82df-
5
6
7
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=rsj36993;PWD=9Yvt1EV6cb3DgOb
g",",")
8  conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90108kqb1od81cg.database
9
10

app.config['SECRET_KEY'] = 'top-secret!'
11  app = Flask(__name__)
12
13  app.config['SECRET_KEY'] = 'top-secret!'

app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
14  app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
15  app.config['MAIL_PORT'] = 587
16  app.config['MAIL_USE_TLS'] = True

app.config['MAIL_DEFAULT_SENDER'] = os.environ.get('MAIL_DEFAULT_SENDER')
mail = Mail(app)
17  app.config['MAIL_USERNAME'] = 'apikey'
18  app.config['MAIL_PASSWORD'] = os.environ.get('SENDGRID_API_KEY')
19  app.config['MAIL_DEFAULT_SENDER'] = os.environ.get('MAIL_DEFAULT_SENDER')

@app.route('/', methods=['GET'])
def home():
20  mail = Mail(app)
21
22  print(os.environ.get('MAIL_DEFAULT_SENDER'))

if 'email' not in session:
    return redirect(url_for('login'))

```

```
return render_template('home.html',name='Home')

23 @app.route("/",methods=['GET'])
24 def home():

Ln 15, Col 30 Spaces: 2 UTF-8 LF Python 3.10.7 64-bit
```

```
def Donorabout():
    return render_template('Donorabout.html')

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

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

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

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

        try:
            email = request.form['email']
            username = request.form['username']
            password = request.form['password']
            userType = request.form['type']
```

```

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)

print("entering1")
if not isUser:
    insert_sql = "INSERT INTO Users(Name,email,PASSWORD,usertype) VALUES
(?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, username)
    ibm_db.bind_param(prepare_stmt, 2, email)
    ibm_db.bind_param(prepare_stmt, 3, password)
    ibm_db.bind_param(prepare_stmt, 4, userType)
    ibm_db.execute(prepare_stmt)
    # print("entering2")

    return render_template('register.html',success="You can login")
else:
    return render_template('register.html',error='Invalid Credentials')

except Exception as e:

```

```

        print("error",e)

    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)

        # userType = isUser.USERTYPE
        # print("## ",isUser["USERTYPE"])

        if(isUser and isUser["USERTYPE"]=="Donor"):
            return render_template('Donorhome.html',error='Invalid Credentials')

```

```

        if(isUser and isUser["USERTYPE"]=="Receipient"):
            return render_template('Receipienthome.html',error='Invalid Credentials')

        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'))

@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['BloodGroup']
    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)

```

```

donor_list = []
for i in listt:
    donor_list.append(i["EMAIL"])

try:
    msg = Message('Urgent!! Plasma needed!!!', recipients=donor_list)

    content = "This email is sent from Plasma Donor Application! \n"+"Request
for plasma was made by: \n"+"Name: "+name+"\n Email: "+email+"\n Phone:
"+phone+"\n Address: "+Address+"\n Please contact for further communications!"

    msg.body = content
    msg.html = f'<p>This email is sent from Plasma Donor Application!</p> \n
<p>Request for plasma was made by: {name}</p> \n <p>Phone number: {phone}</p>\n
<p>Address: {Address}</p> '
    mail.send(msg)
    print("mail successfully sent")
except Exception as e:
    print("error: ",e)

    return render_template('reqReplyS.html',name='reqReplyS',total=len(listt))
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(prepare_stmt, 1, name)
ibm_db.bind_param(prepare_stmt, 2, email)
ibm_db.bind_param(prepare_stmt, 3, phone)
ibm_db.bind_param(prepare_stmt, 4, BloodGroup)
ibm_db.bind_param(prepare_stmt, 5, Address)
ibm_db.execute(prepare_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)

app.debug = True

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

```

GitHub Link:

<https://github.com/IBM-EPBL/IBM-Project-50760-1660923542>

video link:

[https://drive.google.com/file/d/1ufNYLa\\_pjIFy8A6QEmK-Ed\\_Yh-Eu2MG8/view?usp=drivesdk](https://drive.google.com/file/d/1ufNYLa_pjIFy8A6QEmK-Ed_Yh-Eu2MG8/view?usp=drivesdk)