Plasma Donor Application

Project Report

Team ID	PNT2022TMID54477
Team Members	Nilesh Kumar P
	Dineshkumar B
	Akash RV
	Akash C
Batch ID	B8-2A4E
College Name	Easwari Engineering College,Chennai

Content

1.Introduction:

PROJECT OVERVIEW:

Plasma donation, also known as apheresis, can help save lives. It is a relatively safe procedure, but there can be minor side effects. Plasma is the liquid part of the blood. It contains proteins and antibodies that are crucial for clotting and immunity. Around 55% of the blood is plasma. Plasma donation involves drawing blood, extracting the plasma, and returning what is left of the blood to the person, all through a single needle that remains in the arm throughout the process. Plasma is in high demand, as it helps treat cancer and other health issues. In May 2020, the Food and Drug Administration (FDA) asked people who had recovered from COVID19 to donate plasma. Experts believe that the plasma may contain antibodies forSARS-CoV2, the virus behind the disease. Receiving plasma with these antibodies could help a person fight off the infection. People with AB blood have a universal type of plasma, which means that a person with any blood type can receive this plasma safely. This is different from having the universal blood type, which is O negative. The American Red Cross urge people with AB blood to donate plasma. A person can do every 28 days, or up to 13 times a year. Research shows that plasma donation is safe, and the National Institutes of Health (NIH) emphasize that there is no risk of getting the wrong blood back. Also, the FDA and other health authorities regulate the equipment and procedure of plasma donation. However, a person who donates plasma may experience minor adverse effects, and as with any other procedure involving a puncture, certain risks are involved. So, it is highly necessary and equally important to create an application to maintain the donors list and details to contact and track them during emergency situations. When talking about an application, it needs to be easy to handle and user friendly. To be more precise, this application should have all the facilities from registering to donating, and login to request satisfying. And one of the interesting facts is that this application runs on Cloud. This might be a useful application during critical times.

PURPOSE:

When the world is struck by deadly diseases, there is a high risk of mass death of populations across the world. These diseases give no enough time for the surgeons to find medicine and so there is a need to find a quick remedy to reduce mass death of people to such illness. One of the best methods, which is highly effective is the donation of blood plasma of cured individuals to sick persons. This can possibly cure the illness of the infected person. Plasma donation was one of the best methods which was adopted to cure people during the recent global pandemic, COVID-19. The recovery rates were high during these times when death was ultimately increasing as no medicine was found across the globe. Plasma Donation also helps to increase immunity. Another issue was that no cured patient came forward to donate blood plasma, so the infected ones were highly worried as they can’t find anyone to help them. So, we are in need of an application that stores donor details, tracks and informs them upon request from a patient

2.Literature Survey:

“A Systematic Review & Design of Web-Based Blood Management System, year 2021

AUTHORS: Gokul Dudani, Tanushree, Kajal Singh, Anushka Singh Chauhan.

When blood is needed in a hospital, it is usually not available in time, leading to inconsistencies. Both patients and sponsors are unaware that the donor is being hospitalized due to a lack of communication and other services.

A system like this is needed to close the communication gap between hospitals, blood banks, donors, and receptors. The main purpose of a web-based blood donation program is to ensure compliance with blood stock.A web-based blood donation system is a good place to monitor whether a particular type of blood is available in a stack or not, as well as to provide a place where blood can be accessed.

“Web Based Online Blood Donation System”, year 2021

AUTHORS: Ramakant Gawande, Narendra Gupta, Nikhil Thengadi.

They come up with a system to link all donors and help in controlling blood transfusion process. Their system will also maintain database which hold data of donors and blood according to their city and further by their locality.they have proposed a machine so that it will hyperlink all donors.

The machine will help to control the blood transfusion service and create a database to maintain records on shares of blood in every place as records on donors in every city. They will be able to check in as donors and as a result acquire a request from their nearby customers who desires blood to donate blood in instances of want. The online blood donation administration Framework application is an approach to synchronize blood donation centers with Emergency clinics with the assistance of the web.

Anybody willing to give blood can be found at the closest blood donation center Utilizing the android bank the executive framework. Blood donation center can Be followed utilizing maps. The android application is simply accessible to Benefactors to look for blood gifts and ask blood donation centers and clinics to Search out blood donation center and close by givers.

"Towards an Efficient and Secure Blood Bank Management System" ,year 2020

AUTHORS: P.A.J. Sandaruwan, U.D.L. Dolapihilla, D.W.N.R. Karunathilaka; W.A.D.T.L. Wijayaweera, W.H. Rankoth.

We have proposed a management platform for the Blood bank operations with the following modules: (1) forecast blood demand, (2) suggest blood donation campaign locations and (3) secure blood supply chain. The proposed platform has been implemented using techniques such as Long Short-Term Memory (LSTM), k-means clustering, Geographic Information
Systems (GIS), and block chain. Our results show that using our proposed
Modules, we can minimize the imbalance between supply and demand of blood,
Find the most suitable donor in an emergency, and enhance the privacy of data.

"Automated blood bank system using Raspberry PI" ,year 2018

AUTHORS: Ashlesha C. Adsul, V. K. Bhosale, R. M. Autee .

"Raspberry pi based blood bank system" proposed to bring blood donors to the
One place. The aim of this system is fulfil every blood request by using android
Application and raspberry pi. In the proposed system, data about the donors will
Be collected by using android application and raspberry pi by installing systems
At places such as hospitals, blood banks etc. These data will be stored in the
Database. User/Patients needs to access application and needs to enter his
Requirements about the blood in the application the requirements are matched

With the database and message will be to that particular blood donor through GSM Modem.

“Short message service (SMS) based blood bank”, year 2016

AUTHORS: G. Muddu Krishna & S. Nagaraju.

They proposed a system in which services of blood bank will be accessed via SMS. If someone needed blood then they have to request for blood via SMS and Then packet count module of their system will check for availability of blood and Response will be given by data processing module.

“A Health-IOT Platform Based on the Integration of Intelligent Packaging, Unobtrusive Bio-Sensor and Intelligent Medicine Box”,year 2015

AUTHORS: Geng Yang, Li Xie, Matti Mantysalo, Xiaolin Zhou, Zhibo Pang,

Li Da Xu, Sharon Kao-Walter, Qiang Chen, Lirong Zheng. In this paper, an intelligent home-based healthcare platform is proposed and Implemented. It involves iMedBox with connectivity, iMedPack with Communication capability enabled by RFID, Bio-Patch and SOC. It fuses with IOT. The body-worn Bio-Patch can detect and transmit the user bio-signals to the iMedBox in real time. The only limitations are, comprehensive platform missing. And the Physical size, rigid nature and short battery become limitation for long Term use.

References:

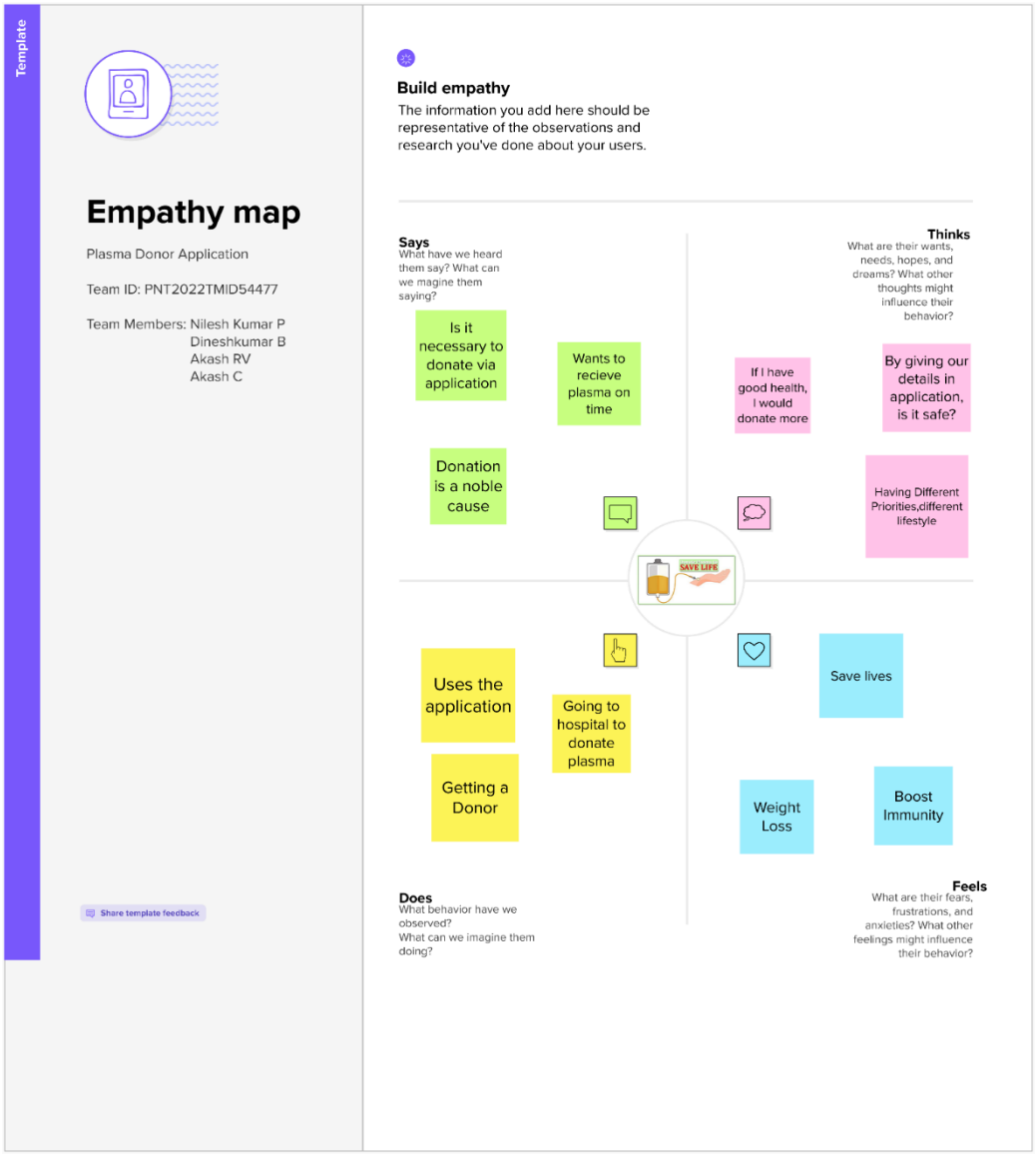
[1] X. Chen, “Commercial plasma donation and individual health in impoverished rural china,” Health economics review, vol. 4, no. 1, p. 30, 2014.
[2] “Blood safety and availability,” https://www.who.int/news- room/fact-sheets/detail/blood-safety-and-availability, (Accessed on 09/05/2020).
[3] T. Alanzi and B. Alsaeed, “Use of social media in the blood donation process in saudi arabia,” Journal of Blood Medicine, vol. 10, p. 417, 2019.
[4] T. Wangchuk, K. Wangmo, U. Wangchuk, P. Gyem, P. R. Dhungyel et al., “Need of medium for finding blood donor in bhutan,” Asian Journal For Convergence In Technology (AJCT), 2018.
[5] V. K. Tatikonda and H. El-Ocla, “Bloodr: blood donor and requester mobile application,” Mhealth, vol. 3, 2017.
[6] M. S. Hossain, N. Das, M. K. H. Patwary, and M. Al-Hasan, “Finding the nearest blood donors using dijkstra algorithm,” SISFORMA: Journal of Information Systems (e-Journal), vol. 5, no. 2, pp. 40–44, 2019.
[7] H. D. Das, R. Ahmed, N. Smrity, and L. Islam, “Bdonor: A geo-localised blood donor management system using mobile crowdsourcing,” in 2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT). IEEE, 2020, pp. 313–317.
[8] S.-Q. Wang and D.-M. Zhu, “Research on selecting initial points for k-means clustering,” in 2008 International Conference on Machine Learning and Cybernetics, vol. 5. IEEE, 2008, pp. 2673–2677.
[9] S. Na, L. Xumin, and G. Yong, “Research on k-means clustering algorithm: An improved k-means clustering algorithm,” in 2010 Third International Symposium on intelligent information technology and security informatics. IEEE, 2010, pp. 63–67.
[10] K. Sasirekha and P. Baby, “Agglomerative hierarchical clustering algorithm-a,” International Journal of Scientific and Research Publications, vol. 83, p. 83, 2013.
[11] D. Müllner, “Modern hierarchical, agglomerative clustering algorithms,” arXiv preprint arXiv:1109.2378, 2011.
[12] “Opencage geocoder.” [Online]. Available: https://opencagedata.com/

Problem Statement:

During COVID 19 crisis the requirement for plasma increased drastically as there were no vaccinations found in order to treat the infected patients. In such situation it was very difficult to find the plasma donor, check whether the donor was infected previously and was recovered, and which donor is eligible to donate plasma was a challenging task.

Who does the problem affect?	Donor, receiver.
What are the boundaries of the problem?	Plasma bank, hospitals.
What is the issue?	In some situation it was very difficult to find the plasma donor, check whether the donor was infected previously and was recovered, and which donor is eligible to donate plasma was a challenging task.
When does the issue occurs?	In case of emergency, it becomes difficult to approach the right donor. When it cannot automatically verify the genuine users. When there is no valid information regarding the plasma donation or managing programs available on any of the portals.
Where is the issue occurring?	The issue occurs in hospital and plasma bank while searching for donors.
Why is it important that we fix the problem?	By solving this issue, donors can easily donate and receiver can get plasma so that we can able to treat the infected patients as soon as possible.

3.Ideation & Proposed Solution:



1

PROBLEM STATEMENT

PROBLEM

How might we handle to create an application fully

dedicated for Plasma Donation?

2

BRAINSTROM

Nilesh Kumar P

Using hospital/clinic organization

to improve the service.

Chatbot service can be included to clear doubts about plasma

donation.

If the blood groupis suitable with the

requested blood , then only the notification are send.

Organizing various activities

to promote the application's interest among the people.

Dineshkumar B

Features like history of the donations made, finding

donor's location using GPS

Only donors from the age of 18 having a weight of 50kg

can register in the application.

Rapid contacting features can be

added when there is a immediate need of plasma.

By using location detecting

features, one can able to find accurate location of the donor.

AKASH C

Details such as gender, D.O.B, age, contact detailsare collected and stored in database.

Plasma of the body are to be examined by a medical expert before the donation.

Verifications are to be made at registration stage in order tomake the donation.

To ensure and verify whether the donor is free from any other cautionary diseases.

Akash R V

Blood group description from both the parties (Donor & collector) is collected.

If there is any misinformation and proved that he/she had any other diseases, immediate action of removing his/her profile are to made.

The exact date of the plasma extraction must be mentioned on his profile.

The user during the initial stage of registration should given whether it is his/her first time at donating or already donated person.

3

GROUP IDEAS

APP INTERFACE

Chatbot service can be included to clear doubts about plasma donation.

Using hospital/clinic organization to improve the service.

If the blood group is suitable with the requested blood type ,then only the notification are send.

Verifications are to be made at registration stage in order to make the donation

FEATURES

Features like history of the donations made, finding donor's location using GPS

Rapid contacting features can be added when there is a immediate need of plasma.

By using location detecting features, one can able to find accurate location of the donor.

Organizing various activities to promote the application's interest among the people.

REGISTRATION STAGES

Details such as age, contact detailsare collected and stored in database.

Blood group description from both the parties (Donor & collector) is collected.

Only donors from the age of18 having a weight of 50kgcan register in the application.

The user during the initial stage of registration should given whether it is his/her first time at donating or already donated person.

SAFETY MEASURES

Plasma of the body are to be examined by a medical expert before the donation.

To ensure and verify whether the donor is free from any other cautionary diseases.

The exact date of the plasma extraction must be mentioned on his profile.

If there is any misinformation and proved that he/she had any other diseases, immediate action of removing his/her profile are to made.

4

PRIORITIZE

Proposed Solution :

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	During COVID19 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. Plasma is a critical part of the treatment for many serious health problems. This is why there are blood drives asking people to donate blood plasma. The current donors list, would be a helping hand.
2.	Idea / Solution description	In regard to the problem faced, an application is to be built which would take the donor details, store them and inform upon a request. This way, one who in need in plasma can able to make a request, then application can able to read the information of donors that are stored database and informing up the donors regarding that request.
3.	Novelty / Uniqueness	This application can able to perform certain functionality and possess certain feature which are unique. Those are listed below: • Those who want to donate their plasma can do by simply register by uploading

		their covid-19 recovery certificate. • It can able to find donors who are located close to the needy by using GPS location tracking. A chat bot to answer frequently asked question about plasma donation.
4.	Social Impact / Customer Satisfaction	By using the application one can easily able to find the donor at emergency situations and the one who willing to donate their plasma can easily be connected with the needy. Since this process takes place continuously, we can build a healthy society of tomorrow.
5.	Business Model (Revenue Model)	We can provide some additional medical services in order to generate some revenue. Medical services like blood test, medical record management, medical transportation service and some other healthcare service.
6.	Scalability of the Solution	Since the whole application is developed based on micro-services architecture, the scalability of the application is made easy. The application can able scale as the users grow and handle the traffic at any situations.

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- Donors
- Patient
- Hospitals

6. CUSTOMER CONSTRAINTS CC

- Regular Internet connection
- Donor health condition
- Unavailability of plasma

5. AVAILABLE SOLUTIONS AS

The existing application used only collecting data collecting details of donors but it does not notify them at the right time.

Our solution is building a website that notifies the donor at the same time.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

- Difficult to find donors at the right time / at the time of emergency.
- Donors not aware of plasma requirements.

9. PROBLEM ROOT CAUSE RC

- Not able to find the donors at the time of emergency.
- Count of donors has been tremendously decreasing since hospital management couldn't contact them or get them notified at the right.

7. BEHAVIOUR BE

The customer comes forward to:

- Attend plasma donation camps.
- Donate plasma
- The hospital management/ patient is able to find plasma donors at the right time.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

Blood donation improves or saves lives and enhances social solidarity. It is also influenced by increasing deaths due to unavailability of plasma at required times.

4. EMOTIONS: BEFORE / AFTER EM

Before:

Patient/ hospital find it hard to get a right resource to get plasma leaving them upset.

After:

The donors and customers have a feeling of satisfaction.

10. YOUR SOLUTION SL

Creating website which will provide information about available donors and plasma. If not available, the customer will be notified when plasma is available.

8.CHANNELS of BEHAVIOUR CH

ONLINE

Can use the website to find donors.

OFFLINE

Can use the record maintain by the hospital.

Identify strong TR & EM

4.Requirement Analysis:

Functional Requirements:

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 Email and Social media accounts
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User Login	Login through registered email id
FR-4	Recipient Request	The recipient makes request for blood type for plasma
FR-5	Donor Request Alert	The Donor gets alerted through email
FR-6	Closed Request Verification	Donor gets an e-certificate and notification for next donation once donation is completed
FR-7	Awareness and Information	Users can look up the benefits of plasma donation and information related
FR-8	Chat Assistant	Helps to solve queries related to donation within the app

Non-functional Requirements:

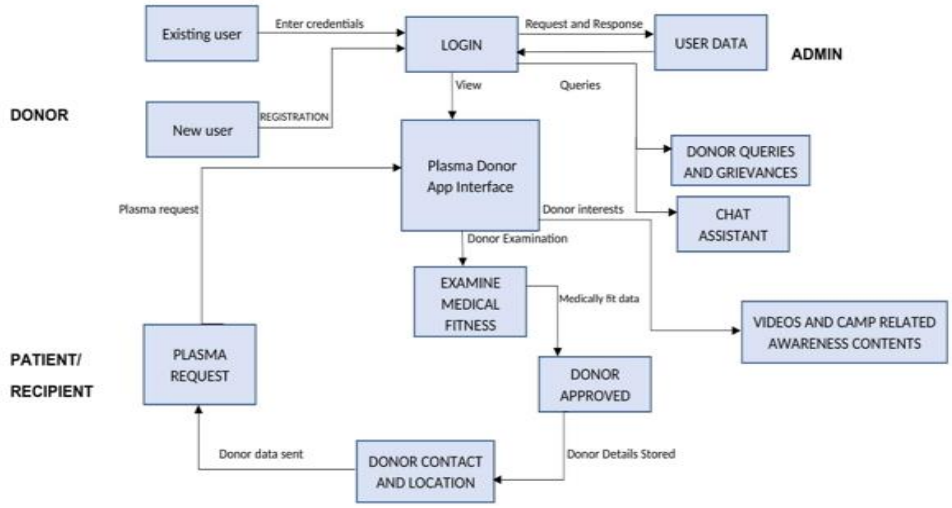
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	This app is easy to use, easy to learn and navigate. Tasks such as booking a donation appointment could be completed in few steps and no instructions and training are required and this app is usable by people of all age group.
NFR-2	**Security**	This is a secure web application plus a secure database system that provides a safe environment for patients, doctors and transplant centers to create online profile for patients seeking living donors of plasma. Fake login and bots are carefully removed.
NFR-3	**Reliability**	All information that the user enters into the app is voluntary and the user can cease the usage at any time and

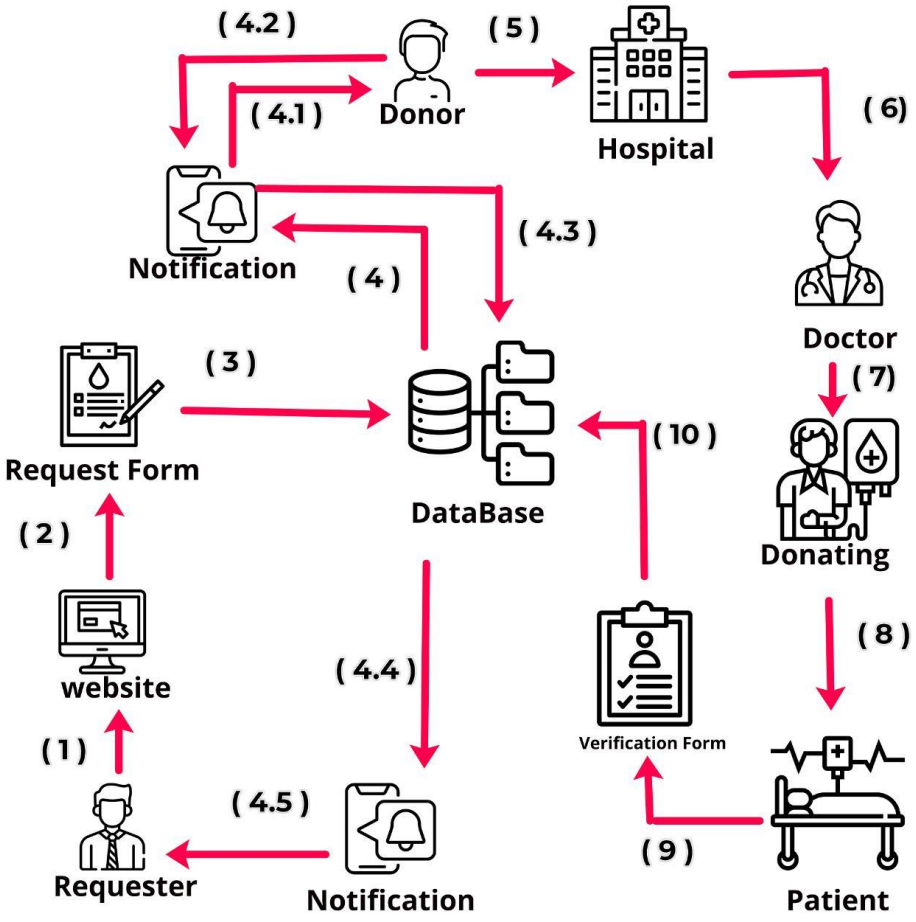
		delete their profile. If the user has shared any information through social network portals, it can also be removed. This app creates a friendly bond with the donors.
NFR-4	**Performance**	There is no lag during usage and the user can experience a glitch free usage. The user also gets route and tips on how to travel conveniently to the donation point.
NFR-5	**Availability**	This App will be available on Google Play store and App Store and also in web.
NFR-6	**Scalability**	This Website has ability to handle multiple donors at a time and provides users with good user experience and reacts fast according to growing number of requests.

5.Project Design:

Data Flow Diagram:



Solution Architecture:



Technology Architecture:

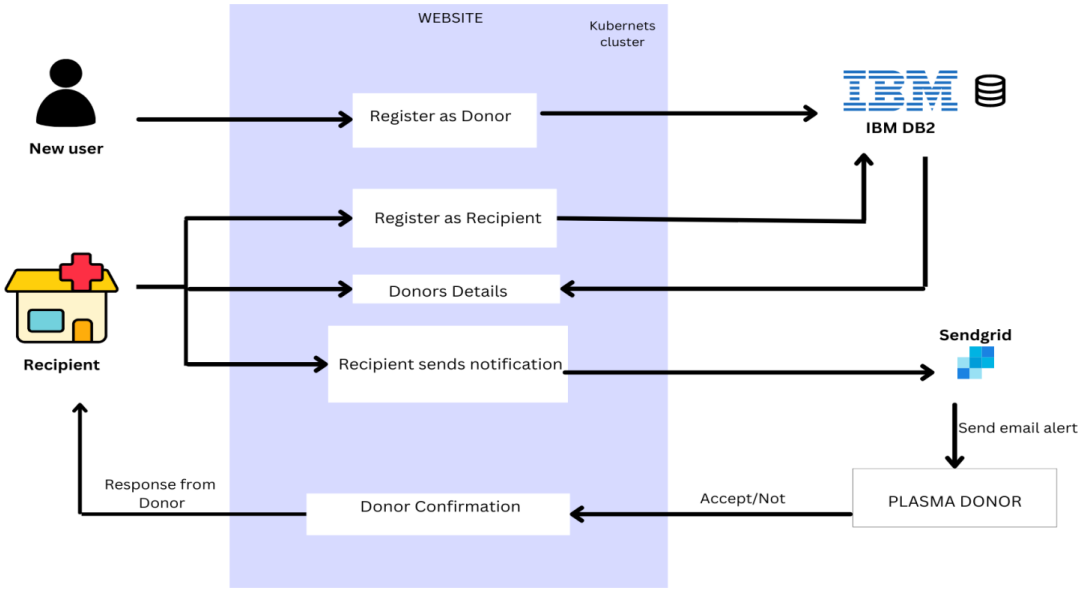


Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	User interact with application using form, login,Request notification	Python FLASK, HTML, CSS
2.	Registration(Donor)	Donors register in the application to Donate the plasma	Python FLASK, HTML, CSS, IBM DB2
3.	Registration(Recipient)	Recipients register in the application to receive a Donor	Python FLASK, HTML, CSS, IBM DB2
4.	Notification	Sends the notification to the Donors	Python FLASK, HTML, CSS, IBM DB2
5.	SendGrid	Sends email alert to recipients	SendGrid
6.	Cloud Database	Database Service on Cloud	IBM DB2
7.	Kubernetes Database	Run Containerized application	IBM Kubernetes

Table-2: Application Characteristics:

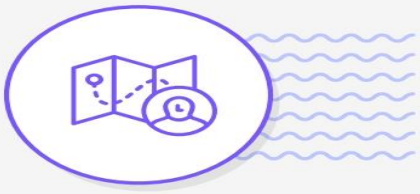
S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	Docker is used for Open Source Framework	DOCKER
2.	Scalable Architecture	It connected with Scalable Architecture	IBM DB2
3.	Availability	This application is anytime accessible	Python FLASK

User Stories:

User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user) Donor	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Social media accounts	I can register & access the app with Social media account	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail other Email services	I can register the app with email account	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can register and access user profile with Gmail account	High	Sprint-1
Patient	Recipient	USN-6	As a requester, I can request the blood group for which I need plasma	I can get plasma from donors when available	High	Sprint-2
Customer (Web user) Donor	Profile	USN-7	As a user, I can see registration page, login page and chat bot for which the user can access to donate and to request for the required blood group plasma.	I can login through email and social media account for registration.	Medium	Sprint-2
Customer Care Executive	Help desk /User support for App	USN-8	As a helpdesk supporter, I can solve the queries and grievances of the user	I can reply to queries and give solutions to problems	High	Sprint-3
Administrator	Registration support	USN-9	As an admin, I can view the database of the registered user	I can check and verify the registered user's login credentials	Medium	Sprint-4
	Dashboard	USN-9	As an admin, I can manage plasma requests and other technical glitches in the app	I can check request numbers and troubleshoot problems in the app	Medium	Sprint-4
Chat Assistant	Dashboard	USN-10	In addition to customer care executive, I can help with user's queries within the app	I can reply to user's queries in the app	Medium	Sprint-4

Template



Customer experience journey map

Plasma Donor Application

Team ID: PNT2022TMID54477

Team Members:
Nilesh Kumar P
Akash RV
Dineshkumar B
Akash C

Created in partnership with

Product School

Document an existing experience

Narrow your focus to a specific scenario or process within an existing product or service. In the **Steps** row, document the step-by-step process someone typically experiences, then add detail to each of the other rows.

SCENARIO

Searching,Requesting,Registering,Recieving Notification about details of Plasma Donor

Entice

How does someone initially become aware of this process?

Steps

What does the person (or group) typically experience?

- **Searching for Plasma**
- **Recomendation (others experience)**
- **Discovers our app**
- Utilizing a search engine, people have looked for plasma.
- Through camps, blood banks, family, friends, magazine etc.
- learns about our website and app

Interactions

What interactions do they have at each step along the way?

- **People:** Who do they see or talk to?
- **Places:** Where are they?
- **Things:** What digital touchpoints or physical objects would they use?

- Patient first interacts with the app, locates, and contacts the donor
- Any desktop or mobile device with internet connectivity and a browser is acceptable for use by the user.
- Users can access the app from any place

Goals & motivations

At each step, what is a person's primary goal or motivation? ("Help me..." or "Help me avoid...")

- Educate donors about the procedure of plasma donation
- Help me locate a suitable plasma donor.
- Help me to get the donor at the right time

Positive moments

What steps does a typical person find enjoyable, productive, fun, motivating, delightful, or exciting?

- Regarding the data and their organisational systems for finding contributors, it is quite clear.
- Positive realisation when you consider how much time it takes compared to other sources.
- Expressing belief and faith in previous reviews.

Negative moments

What steps does a typical person find frustrating, confusing, angering, costly, or time-consuming?

- Uncertain about the after effects of plasma donation
- Sometimes people neglect to correctly submit their information. This causes incorrect donor information.

Areas of opportunity

How might we make each step better? What ideas do we have? What have others suggested?

- Periodically, statistics on the availability of plasma can be updated.
- Delivering a responsive and welcoming user experience.

Enter

What do people experience as they begin the process?

Registration

The initial step is to log in and enter the patient's information, such as age, blood type, how much is required etc.

Contact Details

They must input contact information, such as a phone number, address, and other data, after providing their personal information.

Confirmation

A notice follows the last request for confirmation after providing both the personal and contact information.

Information about donors is gathered using a registration form.

To inform people about their plasma donation eligibility

Confirmation sent through email or SMS.

Assist in connecting with the Donor.

Educate donors about the procedure of plasma donation

Assist recipient in locating plasma donor who is compatible

It is straightforward and easy to enter information and conformation.

For safety, emails and verification are conducted.

If the registration process is unsuccessful, the user may become angry.

Not always feeling hopeful about their needs.

Obtain only the information you require from the Donor. Make the procedure easy and leave out any unnecessary information.

Secured environment as a result of several authentication procedures

Engage

In the core moments in the process, what happens?

Reach Location

Following confirmation, the volunteer donor will receive your details and travel to your location.

Meet up

The donor will get in touch with the requester after visiting your location and follow up.

The user engages with the application's or website's request-making area.

The user engages with the application's or website's search layer

Interacts with the notification that was received through any available means.

Assists as the user look over donor search application.

Helps me receive the necessary information through notice.

Helps me begin the procedure and discover the appropriate donor.

Feel certain when making a request that the right donor will be found.

When we meet the donors, our hunt for donors tends to go well and we feel more secure.

Feel bad if there is a glitch in the request area

Experience disappointment when notice is not received.

Consist of services which the make request procedure on an easy path.

Include methods for receiving notifications across various media.

Exit

What do people typically experience as the process finishes?

Exit the application

The user may close the website after receiving the notification.

Prompt for review

After an hour, an email, and message notification Encourage the tour participant in an evaluation

Review and rating

The reviewer assigns a star rating of 1 to 5 and comments on the app or website

Interacts with the app's exit process and meets the plasma donor in person to obtain the necessary plasma.

Model for "Leave a review" window inside of the a website profile or app.

Assists in completing this application in a manner that is favourable and satisfied.

Tells people about the fantastic services.

The new applicant truly feels inspired and motivated to donate plasma after reading this.

Motivated by receiving a certificate of appreciation for a donor

A user claims that the procedure of leaving a review is difficult.

Feeling bad about the application since the donor had the wrong plasma type after seeing them in person

Include features like an application review system.

How might we gradually reveal the whole review so that each step feels more straightforward.

Accessible in other languages.

Extend

What happens after the experience is over?

history of donation

Detailed information on the completed donations appears on the user's profile along with some information about that contribution.

recommendation

Pop-up recommendations for plasma donations and user need were displayed on the page.

cool off notification

After a few months, it ought to suggest plasma donations.

The website's or Android app's section describing successful donations

Window for recommendations on the website

Interaction that involves asking both the donor and the recipient broad questions about their health.

Enhances more features for the donors' accessibility.

Helps me to realize what I previously did introducing this application.

With the recommendation window, users can both assist others and learn about willing donors.

People prefer to look considering their past completed accomplishments

Even after a contribution, bad ratings might cause ongoing grief.

Maintaining privacy about donor contact details after donation.

Making applicant to remember about their past request.

6.Project Planning & Schedule:

Sprint Planning & Estimation:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	A User can register and create the user account.	6	High	Nilesh Kumar P Dineshkumar B Akash RV Akash C
Sprint-1	Login	USN-2	A User can sign-in to the application by entering the registered email id and password.	6	High	Nilesh Kumar P Dineshkumar B Akash C
Sprint-1	Admin Register	USN-3	An admin can register through the admin registry.	4	Medium	Nilesh Kumar P Akash RV
Sprint-1	Register Admin Via Script	USN-4	Creating an Admin Account using a python script. As for security reasons we should implement a separate python script.	4	High	Dineshkumar B Akash RV Akash C
Sprint-2	Implementing Authentication System	USN-5	creating an authentication system for both admin and users using flask application	6	High	Nilesh Kumar P Dineshkumar B
Sprint-2	Creating Tables	USN-6	Creating Db2 account and creating the tables in DB2 in IBM cloud db2	4	Medium	Nilesh Kumar P Dineshkumar B

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-2	Creating SSL certificate and integrating with python code	USN-7	Creating the SSL certificate to connect db2 via python code.	6	High	Nilesh Kumar P Dineshkumar B Akash RV Akash C
Sprint-2	Creating dashboard	USN-8	Admin and Donor can interact with our application.	4	Medium	Akash RV
Sprint-3	Plasma request and donor acknowledge feature	USN-9	Admin can create plasma requests which will be shown in the user portal.	6	High	Nilesh Kumar P Dineshkumar B Akash RV Akash C
Sprint-3	Creating dashboard for admin	USN-10	Admin dashboard, admin can view the total request has been requested for plasma by the recipient/user.	6	High	Nilesh Kumar P Akash RV Akash C
Sprint-3	Integrating the Watson chat bot	USN-11	Users can use the chat bot for basic clarification using the chat bot.	4	Medium	Nilesh Kumar P Dineshkumar B
Sprint-3	Integration with SendGrid.	USN-12	The source/verification mail for both user (donor and recipient) .	4	Medium	Nilesh Kumar P Akash C
Sprint-4	Docker installation	USN-13	Installing Docker CLI	4	Low	Nilesh Kumar P

Sprint Delivery Schedule:

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

7.Coding and Solution:

Main Python-Flask program:

```
from
distutils.log
import debug
                from sendgridmail import sendmail
                from flask import Flask, render_template, request, redirect, url_for, session
                import ibm_db
                import re
                import os
                from dotenv import load_dotenv

                load_dotenv()


                app = Flask(__name__)

                app.secret_key = 'a'

                conn=ibm_db.connect(os.getenv('DB_KEY'),"","")

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


                @app.route('/loginpage',methods=['GET', 'POST'])
                def loginpage():
                    global userid
                    msg = ''

                    if request.method == 'POST' :
                        username = request.form['username']
                        password = request.form['password']
```

```
        sql = "SELECT * FROM donors WHERE username =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print (account)
        if account:
            session['loggedin'] = True
            session['id'] = account['USERNAME']
            userid=  account['USERNAME']
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'
            sendmail(account['EMAIL'],'Plasma donor App login','You are
successfully logged in!')
            return redirect(url_for('dash'))
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

@app.route('/registration')
def home():
    return render_template('register.html')

@app.route('/register',methods=['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        phone = request.form['phone']
        city = request.form['city']
        infect = request.form['infect']
        blood = request.form['blood']
        sql = "SELECT * FROM donors WHERE username =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            insert_sql = "INSERT INTO  donors 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.bind_param(prep_stmt, 4, city)
```

```
            ibm_db.bind_param(prep_stmt, 5, infect)
            ibm_db.bind_param(prep_stmt, 6, blood)
            ibm_db.bind_param(prep_stmt, 7, phone)

            ibm_db.execute(prep_stmt)
            msg = 'You have successfully registered !'
            sendmail(email,'Plasma donor App Registration','You are
successfully Registered {}!'.format(username))

    elif request.method == 'POST':
        msg = 'Please fill out the form !'
    return render_template('register.html', msg = msg)

@app.route('/dashboard')
def dash():
    if session['loggedin'] == True:
        sql = "SELECT COUNT(*), (SELECT COUNT(*) FROM DONORS WHERE blood= 'O
Positive'), (SELECT COUNT(*) FROM DONORS WHERE blood='A Positive'), (SELECT
COUNT(*) FROM DONORS WHERE blood='B Positive'), (SELECT COUNT(*) FROM DONORS
WHERE blood='AB Positive'), (SELECT COUNT(*) FROM DONORS WHERE blood='O
Negative'), (SELECT COUNT(*) FROM DONORS WHERE blood='A Negative'), (SELECT
COUNT(*) FROM DONORS WHERE blood='B Negative'), (SELECT COUNT(*) FROM DONORS
WHERE blood='AB Negative') FROM donors"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        return render_template('dashboard.html',b=account)
    else:
        msg = 'Please login!'
        return render_template('login.html', msg = msg)

@app.route('/requester')
def requester():
    if session['loggedin'] == True:
        return render_template('request.html')
    else:
        msg = 'Please login!'
        return render_template('login.html', msg = msg)

@app.route('/requested',methods=['POST'])
def requested():
    bloodgrp = request.form['bloodgrp']
    address = request.form['address']
    name=  request.form['name']
    email=  request.form['email']
    phone= request.form['phone']
    insert_sql = "INSERT INTO  requested VALUES (?, ?, ?, ?, ?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, bloodgrp)
    ibm_db.bind_param(prep_stmt, 2, address)
    ibm_db.bind_param(prep_stmt, 3, name)
    ibm_db.bind_param(prep_stmt, 4, email)
    ibm_db.bind_param(prep_stmt, 5, phone)
```

```
            ibm_db.execute(prep_stmt)
            sendmail(email,'Plasma donor App plasma request','Your request for plasma
        is recieved.')
            return render_template('request.html', pred="Your request is sent to the
        concerned people.")


        @app.route('/logout')

        def logout():
           session.pop('loggedin', None)
           session.pop('id', None)
           session.pop('username', None)
           return render_template('login.html')

        if __name__ == '__main__':
           app.run(host='0.0.0.0',debug='TRUE')
```

SendGrid:

```
# using SendGrid's Python Library
# https://github.com/sendgrid/sendgrid-python
import os
from dotenv import load_dotenv

load_dotenv()
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail

def sendmail(usermail,subject,content):
    message =
Mail(from_email='maryada@student.tce.edu',to_emails=usermail,subject
=subject,html_content='<strong> {} </strong>'.format(content))
    try:
        sg = SendGridAPIClient(os.getenv('SENDGRID_API_KEY'))
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e.message)
```

Style.css:

```
@import
url(https://fonts.googleapis.com/css?fami
ly=Open+Sans);
```

```
.btn {
	display: inline-block;
	*display: inline;
	*zoom: 1; padding:
	4px 10px 4px;
	margin-bottom: 0;
	font-size: 13px;
	line-height: 18px;
	color: #333333;
	text-align: center;
	text-shadow: 0 1px 1px rgba(255, 255, 255,
0.75);
	vertical-align: middle;
	background-color: #d70c0c;
	background-image: -moz-linear-
gradient(top, #ffffff, #e6e6e6);
	background-image: -ms-linear-gradient(top,
#ffffff, #e6e6e6);
	background-image: -webkit-gradient(linear,
0 0, 0 100%, from(#ffffff), to(#e6e6e6));
	background-image: -webkit-linear-
gradient(top, #ffffff, #e6e6e6);
	background-image: -o-linear-gradient(top,
#ffffff, #e6e6e6);
	background-image: linear-gradient(top,
#ffffff, #e6e6e6);
	background-repeat: repeat-x;
	filter:
progid:dximagetransform.microsoft.gradient(startC
olorstr=#ffffff, endColorstr=#e6e6e6,
GradientType=0);
	border-color: #e6e6e6 #e6e6e6 #e6e6e6;
	border-color: rgba(0, 0, 0, 0.1) rgba(0,
0, 0, 0.1) rgba(0, 0, 0, 0.25);
	border: 1px solid #e6e6e6;
	-webkit-border-radius: 4px;
	-moz-border-radius: 4px;
	border-radius: 4px;
	-webkit-box-shadow: inset 0 1px 0
rgba(255, 255, 255, 0.2), 0 1px 2px rgba(0, 0, 0,
0.05);
	-moz-box-shadow: inset 0 1px 0 rgba(255,
255, 255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);
	box-shadow: inset 0 1px 0 rgba(255, 255,
255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.05);
	cursor: pointer; *margin-left: .3em;
	}
```

```
.btn:hover, .btn:active, .btn.active,
.btn.disabled, .btn[disabled] { background-color:
#e6e6e6; }

.btn-large {
       padding: 9px 14px;
       font-size: 15px;
       line-height: normal;
       -webkit-border-radius: 5px;
       -moz-border-radius: 5px;
       border-radius: 5px;
       }

.btn:hover {
       color: #333333;
       text-decoration: none;
       background-color: #e6e6e6;
       background-position: 0 -15px;
       -webkit-transition: background-position
0.1s linear;
       -moz-transition: background-position 0.1s
linear;
       -ms-transition: background-position 0.1s
linear;
       -o-transition: background-position 0.1s
linear;
       transition: background-position 0.1s
linear;
       }

.btn-primary, .btn-primary:hover {
       text-shadow: 0 -1px 0 rgba(0, 0, 0, 0.25);
       color: #ffffff;
       }

.btn-primary.active { color: rgba(255, 255, 255,
0.75); }

.btn-primary {
       background-color: #d70c0c;
       background-image: -moz-linear-
gradient(top, #6eb6de, #4a77d4);
       background-image: -ms-linear-gradient(top,
#6eb6de, #4a77d4);
       background-image: -webkit-gradient(linear,
0 0, 0 100%, from(#6eb6de), to(#4a77d4));
       background-image: -webkit-linear-
gradient(top, #6eb6de, #4a77d4);
       background-image: -o-linear-gradient(top,
#6eb6de, #4a77d4);
       background-image: linear-gradient(top,
#6eb6de, #4a77d4);
       background-repeat: repeat-x;
```

```
        filter:
progid:dximagetransform.microsoft.gradient(startC
olorstr=#6eb6de, endColorstr=#4a77d4,
GradientType=0);
        border: 1px solid #3762bc;
        text-shadow: 1px 1px 1px rgba(0,0,0,0.4);
        box-shadow: inset 0 1px 0 rgba(255, 255,
255, 0.2), 0 1px 2px rgba(0, 0, 0, 0.5);
        }

.btn-primary:hover, .btn-primary:active, .btn-
primary.active, .btn-primary.disabled, .btn-
primary[disabled] {
        filter: none;
        background-color: #d70c0c
        }

.btn-block { width: 100%; display:block; }

* { -webkit-box-sizing:border-box; -moz-box-
sizing:border-box; -ms-box-sizing:border-box; -o-
box-sizing:border-box; box-sizing:border-box; }

html { width: 100%; height:100%; overflow:hidden;
}

body {
        width: 100%;
        height:100%;
        font-family: 'Open Sans', sans-serif;
        color: #000000;
        font-size: 18px;
        text-align:center;
        letter-spacing:1.2px;


}
.header {
                        top:0;
                        margin:0px;
                        left: 0px;
                        right: 0px;
                        position: fixed;
                        background: #d44a4a;
                        color: black;
                        box-shadow: 0px 8px 4px
grey;
                        overflow: hidden;
                        padding: 15px;
                        font-size: 1.5vw;
                        width: 100%;
                        text-align: center;
                }
.login {
```

```
        position: absolute;
        top: 70%;
        left: 50%;
        margin: -25px 0 0 -150px;
        width:400px;
        height:400px;
}

.header div { color: #fff; text-shadow: 0 0 10px
rgba(0,0,0,0.3); letter-spacing:1px; text-
align:center; float:left; padding-left:150px;}

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  padding-right:150px;
  overflow: hidden;
 }


li {
  float: right;
}

li a {
  display: block;
  color: white;
  text-align: center;
  padding: 0px 15px;
  text-decoration: none;
}


input {
        width: 100%;
        margin-bottom: 10px;
        background: rgba(255,255,255,255);
        border: none;
        outline: none;
        padding: 10px;
        font-size: 13px;
        color: black;
        text-shadow: black;
        border: 1px solid rgba(0,0,0,0.3);
        border-radius: 4px;
        box-shadow: inset 0 -5px 45px
rgba(100,100,100,0.2), 0 1px 1px
rgba(255,255,255,0.2);
        -webkit-transition: box-shadow .5s ease;
        -moz-transition: box-shadow .5s ease;
        -o-transition: box-shadow .5s ease;
        -ms-transition: box-shadow .5s ease;
        transition: box-shadow .5s ease;
```

```
}
input:focus { box-shadow: inset 0 -5px 45px
rgba(100,100,100,0.4), 0 1px 1px
rgba(255,255,255,0.2); }


textarea {
	width: 100%;
	margin-bottom: 10px;
	background: rgba(255,255,255,255);
	border: none;
	outline: none;
	padding: 10px;
	font-size: 13px;
	color: black;
	text-shadow: black;
	border: 1px solid rgba(0,0,0,0.3);
	border-radius: 4px;
	box-shadow: inset 0 -5px 45px
rgba(100,100,100,0.2), 0 1px 1px
rgba(255,255,255,0.2);
	-webkit-transition: box-shadow .5s ease;
	-moz-transition: box-shadow .5s ease;
	-o-transition: box-shadow .5s ease;
	-ms-transition: box-shadow .5s ease;
	transition: box-shadow .5s ease;
}
textarea:focus { box-shadow: inset 0 -5px 45px
rgba(100,100,100,0.4), 0 1px 1px
rgba(255,255,255,0.2); }

select {
	width: 100%;
	margin-bottom: 10px;
	background: rgba(255,255,255,255);
	border: none;
	outline: none;
	padding: 10px;
	font-size: 13px;
	color: #000000;
	text-shadow: 1px 1px 1px rgba(0,0,0,0.3);
	border: 1px solid rgba(0,0,0,0.3);
	border-radius: 4px;
	box-shadow: inset 0 -5px 45px
rgba(100,100,100,0.2), 0 1px 1px
rgba(255,255,255,0.2);
	-webkit-transition: box-shadow .5s ease;
	-moz-transition: box-shadow .5s ease;
	-o-transition: box-shadow .5s ease;
	-ms-transition: box-shadow .5s ease;
	transition: box-shadow .5s ease;
}
```

Dashboard:

```
<!DOCTYP
E html>
            <html lang="en">
            <head>
              <title>IBM Plasma Donar App</title>
              <meta charset="utf-8">
              <meta name="viewport" content="width=device-width, initial-scale=1">
              <link rel="stylesheet"
            href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
              <script
            src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
              <script
            src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></s
            cript>
              <script
            src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
              <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
            </head>
            <style>
                            .big{
                            top:70;
                            background-color:white;
                            margin-top:80px;
                            margin-left:550px;
                            margin-right:550px;
                            height:200px;
                            border-radius: 25px;
                            border: 3px solid #4a77d4;
                            box-shadow: 6px 8px 4px grey;
                            text-align:center;
                            }
                            .row{

                            height:150px;

                            }
                            .col{
                                    margin:10px;
                                    margin-left:50px;
                                    margin-right:50px;
                                    border-radius: 25px;
                                    border: 1px solid #4a77d4;
                                    box-shadow: 0px 8px 4px grey;
                                    text-align:center;
                            }
                            .ext{
                            margin-top:25px;
                            line-height:40px;
                            }
                            .ext1{
                            margin-top:40px;
```

```
                line-height:50px;
                font-size:25px;
                color:#f95450;
                }

</style>
<body>

<div class="container-fluid">
<div class="header">
<div><b>Plasma Donar App</b></div>
<ul>
                <li><a href="/requester">Request</a></li>
                <li><a class="active" href="/logout">Logout</a></li>

        </ul>
</div>
  <br>
  <div class="big">
    <div class="box">
                <div class="ext1"><font
size="20px">{{b['1']}}</font><br><b>Donors</b></div>
        </div>
  </div>
  <br>
  <div class="row">
    <div class="col" >
                <div class="ext">{{b['2']}}<br><b>O Positive</b></div>
        </div>
    <div class="col" >
                <div class="ext">{{b['3']}}<br><b>A Positive</b></div>
        </div>
    <div class="col" >
                <div class="ext">{{b['4']}}<br><b>B Positive</b></div>
        </div>
    <div class="col" >
                <div class="ext">{{b['5']}}<br><b>AB Positive</b></div>
        </div>
  </div>
  <br>
  <div class="row">
    <div class="col" >
                <div class="ext">{{b['6']}}<br><b>O Negative</b></div>
        </div>
    <div class="col" >
                <div class="ext">{{b['7']}}<br><b>A Negative</b></div>
        </div>
    <div class="col" >
                <div class="ext">{{b['8']}}<br><b>B Negative</b></div>
        </div>
    <div class="col" >
                <div class="ext">{{b['9']}}<br><b>AB Negative</b></div>
        </div>
  </div>
```

```
  <div style="height:200px"></div>
</div>
</body>
</html>
```

Login:

```
<!DOCTYPE
html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>IBM Donor App</title>
       <link href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
       <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
type='text/css'>
       <link href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>
       <link
href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
       <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">


<style>
.login{
top: 20%;
}
</style>
</head>

<body>
<div class="header">
<div>Plasma Donor App</div>
       <ul>



              <li><a href="/registration">Register</a></li>
              <li><a class="active" href="/login">Home</a></li>
       </ul>
</div>
 <div class="login" >
              <div>
              </div>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('loginpage')}}"method="post">
       <input type="text" name="username" placeholder="Enter UserName"
required="required" style="color:black" />
```

```
        <input type="password" name="password" placeholder="Enter Password"
required="required" style="color:black" />
        <button type="submit" class="btn btn-primary btn-block btn-
large">Login</button>

    </form>
<br><br>
<div style="color:black">
 {{ msg }}</div>
 </div>

</body>
</html>
```

Register:

```
<!DOCTYPE
html>
            <html >
            <!--From https://codepen.io/frytyler/pen/EGdtg-->
            <head>
              <meta charset="UTF-8">
              <title>IBM Plasma Donor App</title>
                  <link href='https://fonts.googleapis.com/css?family=Pacifico'
            rel='stylesheet' type='text/css'>
                  <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
            type='text/css'>
                  <link href='https://fonts.googleapis.com/css?family=Hind:300'
            rel='stylesheet' type='text/css'>
                  <link
            href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
            rel='stylesheet' type='text/css'>
                  <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
            }}">

            <style>
            .login{
            top: 20%;
            }
            </style>
            </head>

            <body>
            <div class="header">
            <div>Plasma Donor App</div>
                  <ul>



                        <li><a class="active" href="/login">Home</a></li>
                  </ul>
            </div>
             <div class="login">
```

```
     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('register')}}"method="post">
       <input type="text" name="username" placeholder="Enter Your Name"
required="required" style="color:black"/>
        <input type="email" name="email" placeholder="Enter Email"
required="required" style="color:black"/>
              <input type="text" name="phone" placeholder="Enter 10-digit mobile
number" required="required" style="color:black"/>
        <input type="city" name="city" placeholder="Enter Your City Name"
required="required" style="color:black"/>
              <select name="infect">

                                    <option value="select" selected>Select COVID
infection status</option>
                                    <option value="infected">Infected</option>
                                    <option
value="uninfected">Uninfected</option>
              </select>
        <select name="blood">
                                    <option value="select" selected>Choose your
blood group</option>
                                    <option value="O Positive">O
Positive</option>
                                    <option value="A Positive">A
Positive</option>
                                    <option value="B Positive">B
Positive</option>
                                    <option value="AB Positive">AB
Positive</option>
                                    <option value="O Negative">O
Negative</option>
                                    <option value="A Negative">A
Negative</option>
                                    <option value="B Negative">B
Negative</option>
                                    <option value="AB Negative">AB
Negative</option>
              </select>
        <input type="password" name="password" placeholder="Enter Password"
required="required" style="color:black"/>
        <button type="submit" class="btn btn-primary btn-block btn-
large">Register</button>

    </form>

 <br><br>
<div style="color:black">
 {{ msg }}</div>
 </div>



</body>
</html>
```

Request:

```
<!DOCTYPE
html>
            <html >
            <!--From https://codepen.io/frytyler/pen/EGdtg-->
            <head>
              <meta charset="UTF-8">
              <title>IBM Plasma Donor App</title>
                    <link href='https://fonts.googleapis.com/css?family=Pacifico'
            rel='stylesheet' type='text/css'>
                    <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'
            type='text/css'>
                    <link href='https://fonts.googleapis.com/css?family=Hind:300'
            rel='stylesheet' type='text/css'>
                    <link
            href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
            rel='stylesheet' type='text/css'>
                    <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
            }}">

            <style>
            .login{
            top: 20%;
            }
            </style>
            </head>

            <body>
            <div class="header">
            <div>Plasma Donor App</div>
                    <ul>


                            <li><a href="/requester">Request</a></li>
                            <li><a href="/registration">Register</a></li>
                            <li><a class="active" href="/dashboard">Home</a></li>
                    </ul>
            </div>
             <div class="login">
                            <div>
                            </div>

                 <!-- Main Input For Receiving Query to our ML -->
                <form action="{{ url_for('requested')}}"method="post">
                    <input type="text" name="name" placeholder="Enter Name" required="required"
            style="color:black" />
                     <input type="email" name="email" placeholder="Enter Email"
            required="required" style="color:black"/>
                            <input type="text" name="phone" placeholder="Enter 10-digit mobile
            number" required="required" style="color:black"/>
                            <select name="bloodgrp">
```

```
                                        <option value="select" selected>Choose your
blood group</option>
                                        <option value="O Positive">O
Positive</option>
                                        <option value="A Positive">A
Positive</option>
                                        <option value="B Positive">B
Positive</option>
                                        <option value="AB Positive">AB
Positive</option>
                                        <option value="O Negative">O
Negative</option>
                                        <option value="A Negative">A
Negative</option>
                                        <option value="B Negative">B
Negative</option>
                                        <option value="AB Negative">AB
Negative</option>
                </select>
                <textarea rows="4" placeholder="Enter the address"
required="required" style="color:black" name="address"></textarea>
          <button type="submit" class="btn btn-primary btn-block btn-large">Submit
the request</button>

      </form>


 <br><br>
<div style="color:black">
 {{ pred }}</div>

 </div>


</body>
</html>
```

Docker:

```
FROM
python:3.6
 WORKDIR /app
 ADD . /app
 COPY requirements.txt /app
 RUN python3 -m pip install -r requirements.txt
 EXPOSE 5000
 CMD ["python","app.py"]
```

8.Testing:

User Acceptance Testing:

Defect Analysis:

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	0	0	0	5
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	10	0	0	0	10

Test Case Analysis:

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	0	0	0	0
Client Application	5	0	0	5
Security	0	0	0	0
Outsource Shipping	0	0	0	0

Test Cases:

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID
1	Functional	Login Page	Verify user is able to Login into the Application		1) Open the Plasma Donor Applicaion 2) Login with user Credentials 3) Verify logged in to user account	UserName:Bhuvi Password:Testing	Login Successfully	Working as expected	Pass		N	
2	Functional	Sign up Page	Verify the User is able to Signup in th eapplication		1) Open the Plasma Donor Applicaion 2) Enter the Details and Create a new User 3) Verify if User is created and inserted into DB Table	Username:Kumar Password:Testing Name:Kumar DOB:03/06/2001 Password:Testing	Account Created Successfully	Working as expected	Pass		N	
3	Functional	Personal Details page	Verify if all the user details are stored in Database		1) Open the Plasma Donor Applicaion 2) Enter the Details 3) Verify if user is created and inserted into DB Table	Username:Chidambaram Password:Testing Name:Chidambaram DOB:20/08/2001 Age:21 Availability:Available Contact No:9047353651 City:Chennai State:Tamil Nadu Country:India Blood Type: A+ Description: Happy to Donate	User Details must be stored in the database	Working as expected	Pass		N	
4	Functional	Search page	Search users based on blood type, city and availability		1) Log in to Plasma Donor Application 2) Enter City and Blood type 3) All the donor details with city and blood type is displayed.	City:Chennai Blood Type:A+	Donor details with matching details must be displayed	Working as expected	Pass		N	
5	Functional	Request page	Verify the Request is displayed		1) Log in to Plasma Donor Application 2) All the requests for the user is received		All the request received by the user must be displayed	Working as expected	Pass		N	

9. Result:

Performance Testing:

NFT - Risk Assessment

S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Voluem Changes	Risk Score	Justification
1	Plasma Donor	New	Low	No Changes	Moderate	Yes, 2hrs	>10 to 30%	GREEN	

NFT - Detailed Test Plan

S.No	Project Overview	NFT Test approach	sumptions/Dependencies/Ri	Approvals/SignOff
1	Login Page	1) Open the Plasma Donor App 2) Login with User Credentials	No Risks	N/A
2	Sign Up Page	1) Open the Plasma Donor App 2) Enter the Details and Create a New User	No Risks	N/A
3	Personal Details Page	1) Log in to Plasma Donor Application 2) Enter all the Personal Details and availability Details	No Risks	N/A
4	Search Donor Page	1) Log in to Plasma Application 2) Enter City and Blood type 3) All the Donor details with city and Blood type is displayed	No Risks	N/A
5	Request Page	1) Log in to Plasma Donor Application 2) All the Requests for the User is Received	No Risks	N/A
6	Email Acknowledgement	1) Mails are Sent to the requested user 2) Mails are sent to the reuest user	No Risks	N/A

End Of Test Report

S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	Plasma Donor	2) Test for all Testcases 3) Log out of the Plasma Donor Application	YES	Test Passed	GO	N/A	None	N/A

10.ADVANTAGES & DISADVANTAGES :

ADVANTAGES:

- Very much helpful at times of emergency as this app helps us to find donors easily.
- User friendly interface of the app makes it easier to interact
- Helps very much in voluntary activities.
- Clear details of donors and acceptors can be found once request/donation is placed
- Does not consume much space as it runs in the cloud

DISADVANTAGES:

- Since it is a beta version some user troubles couldn’t be handled
- Verification of details from Admin side could make delay.

11.CONCLUSION:

“Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction.” Cloud makes hardware resources readily available and quick to configure, which shortens the time required for developers to show a working version of their products. Also, cloud allows the reuse of the same resources for multiple successive projects, which is more cost-efficient. As this is a cloud based app it is easy to handle and use.We will hope to look after the updates of this Plasma Donor Application in the future!

12.Appendix:

Source Code:

```
From
distutils.log
import debug
                from sendgridmail import sendmail
                from flask import Flask, render_template, request, redirect, url_for,
                session
                import ibm_db
                import re
                import os
                from dotenv import load_dotenv

                load_dotenv()



                app = Flask(__name__)

                app.secret_key = 'a'

                conn=ibm_db.connect(os.getenv('DB_KEY'),"","")

                @app.route('/')
                @app.route('/login')
```

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


@app.route('/loginpage',methods=['GET', 'POST'])
def loginpage():
    global userid
    msg = ''

    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']
        sql = "SELECT * FROM donors WHERE username =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.bind_param(stmt,2,password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print (account)
        if account:
            session['loggedin'] = True
            session['id'] = account['USERNAME']
            userid=  account['USERNAME']
            session['username'] = account['USERNAME']
            msg = 'Logged in successfully !'
            sendmail(account['EMAIL'],'Plasma donor App login','You are
successfully logged in!')
            return redirect(url_for('dash'))
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)

@app.route('/registration')
def home():
    return render_template('register.html')

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

```
        phone = request.form['phone']
        city = request.form['city']
        infect = request.form['infect']
        blood = request.form['blood']
        sql = "SELECT * FROM donors WHERE username =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            insert_sql = "INSERT INTO  donors 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.bind_param(prep_stmt, 4, city)
            ibm_db.bind_param(prep_stmt, 5, infect)
            ibm_db.bind_param(prep_stmt, 6, blood)
            ibm_db.bind_param(prep_stmt, 7, phone)

            ibm_db.execute(prep_stmt)
            msg = 'You have successfully registered !'
            sendmail(email,'Plasma donor App Registration','You are
successfully Registered {}!'.format(username))

    elif request.method == 'POST':
        msg = 'Please fill out the form !'
    return render_template('register.html', msg = msg)

@app.route('/dashboard')
def dash():
    if session['loggedin'] == True:
        sql = "SELECT COUNT(*), (SELECT COUNT(*) FROM DONORS WHERE blood=
'O Positive'), (SELECT COUNT(*) FROM DONORS WHERE blood='A Positive'),
(SELECT COUNT(*) FROM DONORS WHERE blood='B Positive'), (SELECT COUNT(*)
FROM DONORS WHERE blood='AB Positive'), (SELECT COUNT(*) FROM DONORS WHERE
```

```
blood='O Negative'), (SELECT COUNT(*) FROM DONORS WHERE blood='A
Negative'), (SELECT COUNT(*) FROM DONORS WHERE blood='B Negative'), (SELECT
COUNT(*) FROM DONORS WHERE blood='AB Negative') FROM donors"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        return render_template('dashboard.html',b=account)
    else:
        msg = 'Please login!'
        return render_template('login.html', msg = msg)

@app.route('/requester')
def requester():
    if session['loggedin'] == True:
        return render_template('request.html')
    else:
        msg = 'Please login!'
        return render_template('login.html', msg = msg)

@app.route('/requested',methods=['POST'])
def requested():
    bloodgrp = request.form['bloodgrp']
    address = request.form['address']
    name=  request.form['name']
    email=  request.form['email']
    phone= request.form['phone']
    insert_sql = "INSERT INTO  requested VALUES (?, ?, ?, ?, ?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, bloodgrp)
    ibm_db.bind_param(prep_stmt, 2, address)
    ibm_db.bind_param(prep_stmt, 3, name)
    ibm_db.bind_param(prep_stmt, 4, email)
    ibm_db.bind_param(prep_stmt, 5, phone)
    ibm_db.execute(prep_stmt)
    sendmail(email,'Plasma donor App plasma request','Your request for
plasma is recieved.')
    return render_template('request.html', pred="Your request is sent to
the concerned people.")


@app.route('/logout')
```

```
def logout():
   session.pop('loggedin', None)
   session.pop('id', None)
   session.pop('username', None)
   return render_template('login.html')

if __name__ == '__main__':
   app.run(host='0.0.0.0',debug='TRUE')
```

GITHUB & DEMO LINK:

GITHUB:

https://github.com/IBM-EPBL/IBM-Project-42327-1667195805

DEMO LINK:

https://drive.google.com/file/d/1ARN7p1B3G-bvrneQQhBl1ma5onRy-tLD/view?usp=share_link