PLASMA DONOR APPLICATION

TEAM ID:PNT2022TMID47490

MENTOR: Gnana Jeslin J

COLLEGE: R.M.K COLLEGE OF ENGINEERING AND TECHNOLOGY , PUDUVOYAL

TEAM MEMBERS:

Lankipalli Nitheesh Kumar
Matcha Chandra Kiran
Pinnamareddy Sai Tharun Reddy
Mukesh NR

S.NO	TABLE OF CONTENTS	PAGE NO.
1	**INTRODUCTION**	
	1.1 Project Overview	
	1.2 Purpose	
2	**LITERATURE SURVEY**	
	2.1 Realtime Software for Existing Problems	
	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 Requirements	
	4.2 Non-Functional Requirements	
5	**PROJECT DESIGN**	
	5.1 Data Flow Diagram	
	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**	
	7.1 Feature 1	
	7.2 Feature 2	
	7.3 Database Schema (if Applicable)	
8	**TESTING**	
	8.1 Test Cases	
	8.2 User Acceptance Testing	
9	**RESULTS**	
	9.1 Performance Metrics	
10	**ADVANTAGES & DISADVANTAGES**	
11	**CONCLUSION**	
12	**APPENDIX** (Source code, GitHub and Demo link)	

1.INTRODUCTION

1.1 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 for SARS-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.

1.2 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

2.1 REALTIME SOFTWARE FOR EXISTING PROBLEMS

FLASK: There is a recent transformation into the development of multi-platform languages and frameworks. Flask is a small framework by most standards, small enough to be called a “microframework.” It is small enough that once you become familiar with it, you will likely be able to read and understand all of its source code. But being small does not mean that it does less than other frameworks. Flask was designed as an extensible framework from the ground up; it provides a solid core with the basic services, while extensions provide the rest. Because you can pick and choose the extension packages that you want, you end up with a lean stack that has no bloat and does exactly what you need. Flask has two main dependencies. The routing, debugging, and Web Server Gateway Interface (WSGI) subsystems come from Werkzeug, while template support is provided by Jinja2. Werkzeug and Jinja2 are authored by the core developer of Flask. There is no native support in Flask for accessing databases, validating web forms, authenticating users, or other high-level tasks. These and many other key services most web applications need are available through extensions that integrate with the core packages. As a developer, you have the power to cherry-pick the extensions that work best for your project or even write your own if you feel inclined to. This is in contrast with a larger framework, where most choices have been made for you and are hard or sometimes impossible to change.

DOCKER & KUBERNETES: Docker is an open-source engine that automates the deployment of applications into containers. It was written by the team at Docker, Inc (formerly dot Cloud Inc, an early player in the Platform-as-a-Service (PAAS) market), and released by them under the Apache 2.0 license. Docker adds an application deployment engine on top of a virtualized container execution environment. It is designed to provide a lightweight and fast environment in which to run your code as well as an efficient workflow to get that code from your laptop to your test environment and then into production. Docker is incredibly simple. Indeed, you can get started with Docker on a minimal host running nothing but a compatible Linux kernel and a Docker binary.

Kubernetes, or k8s for short, is an open-source container orchestrator. Originally developed by the engineers at Google, Kubernetes solves many problems involved with running a microservice architecture in production. Kubernetes automatically takes care of scaling, self-healing, load-balancing, rolling updates, and other tasks that used to be done manually by DevOps engineers. Since Kubernetes was open-sourced and managed by Cloud Native Computing Foundation in 2014, the development community has embraced its benefits to orchestrate container-based systems.

These Framework and Software help the application to gain a run-time structure and assist with all internal features of the app.

At times of pandemic, people should come forward to donate their blood plasma voluntarily to arrest the spread of disease and provide the cure that they have experienced. Our app will feature awareness videos and articles and will provide rewards for first time donors and regular donors. All These Humanitarian deeds are to be carried on regularly to maintain social well hood and supportive society. There are many incidents to be pointed out citing plasma donation during the past in Covid-19 Pandemic. One such incident where India's first Covid-19 plasma donor shares her story, urges other patients to do the same. Members of an Indian Islamic organisation are volunteering to donate blood for plasma therapy after their congregation sparked dozens of Covid-19 clusters across the country. All these articles are mentioned to encourage the donation of plasma in our application

2.2 REFERENCES

1) Philip J, Sarkar RS, Pathak A. Adverse events associated with apheresis procedures: Incidence and relative frequency. Asian J Transfus Sci. 2013 Jan;7(1):37-41. doi: 10.4103/0973-6247.106730. PMID: 23559763; PMCID: PMC3613659.

2) Flask Web Development by Miguel Grinberg Copyright © 2014 Miguel Grinberg. All rights reserved. Printed in the United States of America. Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

3) Version v1.2.0 (fba92ef) of The Docker Book by James Turnbul © Copyright 2014 - James Turnbull. 4) An Introduction to Kubernetes by LEVEREGE , First Edition © Leverege LLC.

5) White Paper by IDC Sponsored by: IBM Andrew Smith, February 2021.

6) Essentials of Application Development on IBM Cloud December 2017 Third Edition (December 2017) by Ahmed Azraq Hala A. Aziz Uzma Siddiqui.

7) From Ahmedabad, Smriti Thakkar is the first recovered Covid-19 patient in India who volunteered to donate her plasma. India Today Web Desk New Delhi, April 25, 2020 UPDATED: April 25, 2020 19:15 IST

8) Tablighi Jamaat gives blood for plasma therapy By Zubair Ahmed BBC Hindi, Delhi Published 28 April 2020.

2.2 PROBLEM STATEMENT DEFINITION

A customer problem statement outlines problems that your customers face. It helps you figure out how your product or service will solve this problem for them. The statement helps you understand the experience you want to offer your customers. It can also help you understand a new audience when creating a new product or service. A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face. Throughout the process, you'll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

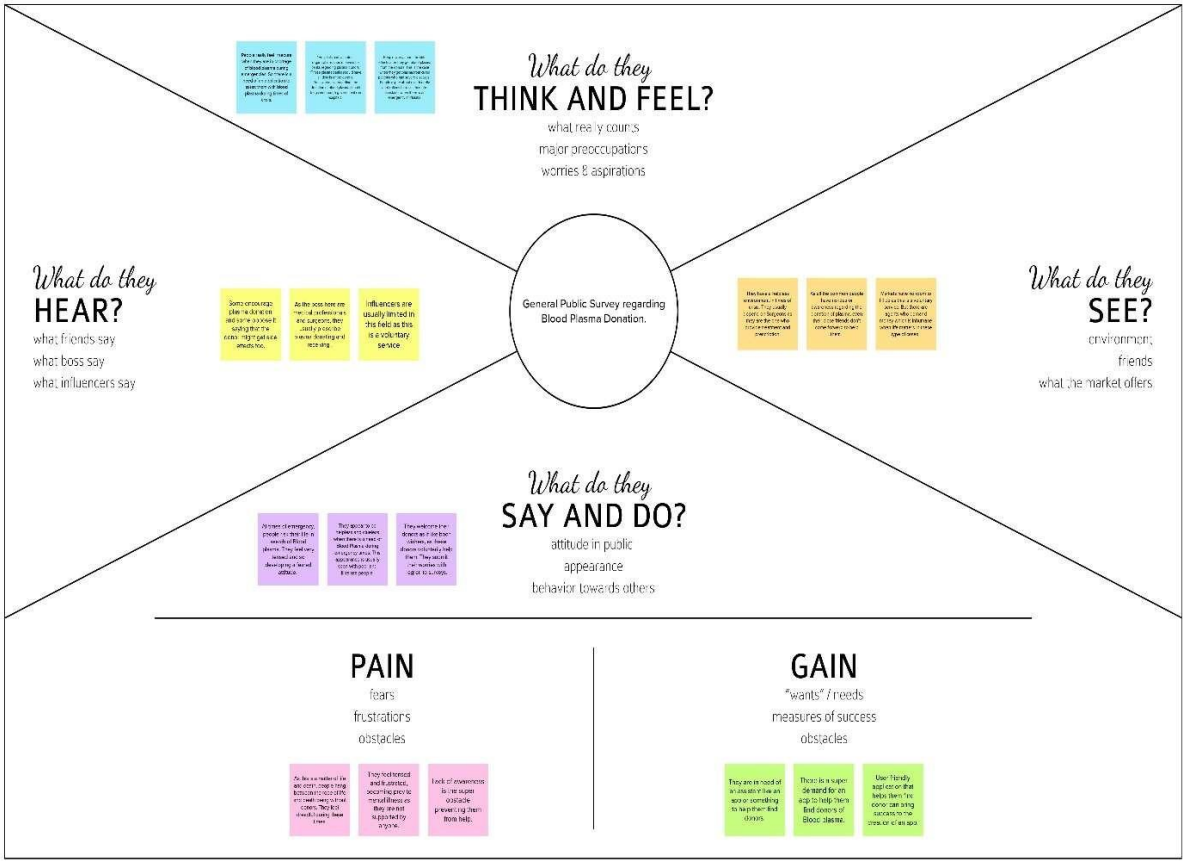
The most common problem faced by a person who is in need of blood plasma is reported below after a detailed analysis,

"As a patient in need of blood plasma, I am trying to contact a donor to help me with blood plasma, but I cannot find one easily as it takes long time and hardly possible, because the donors are usually far away from me and not ready to help me, which makes me feel hopeless and annoyed"

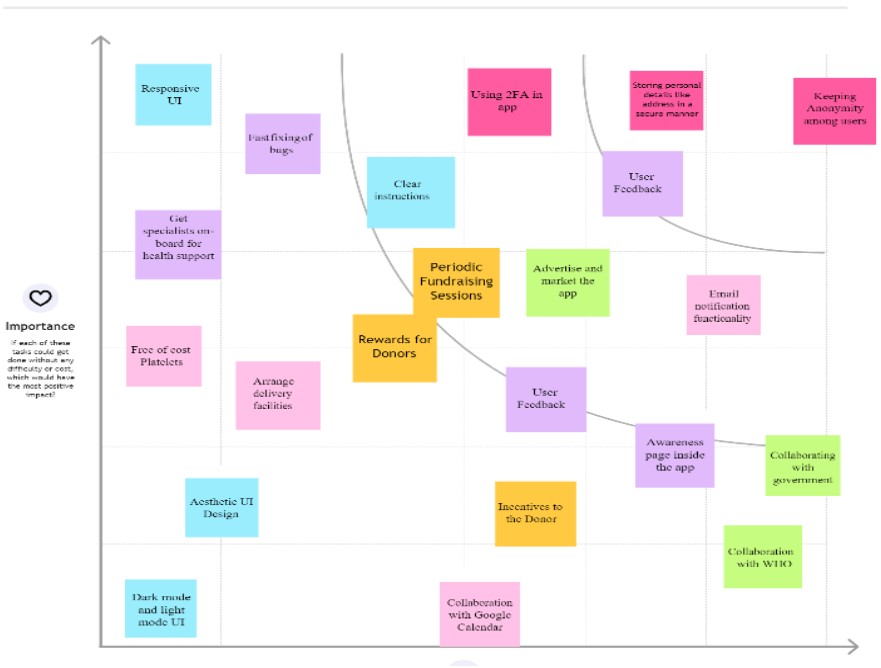
This is the most common problem statement derived after long surveys from the general public.

3.IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION & BRAINSTORMING



3.3 PROPOSED SOLUTION

Project Design
Phase-I
Proposed Solution

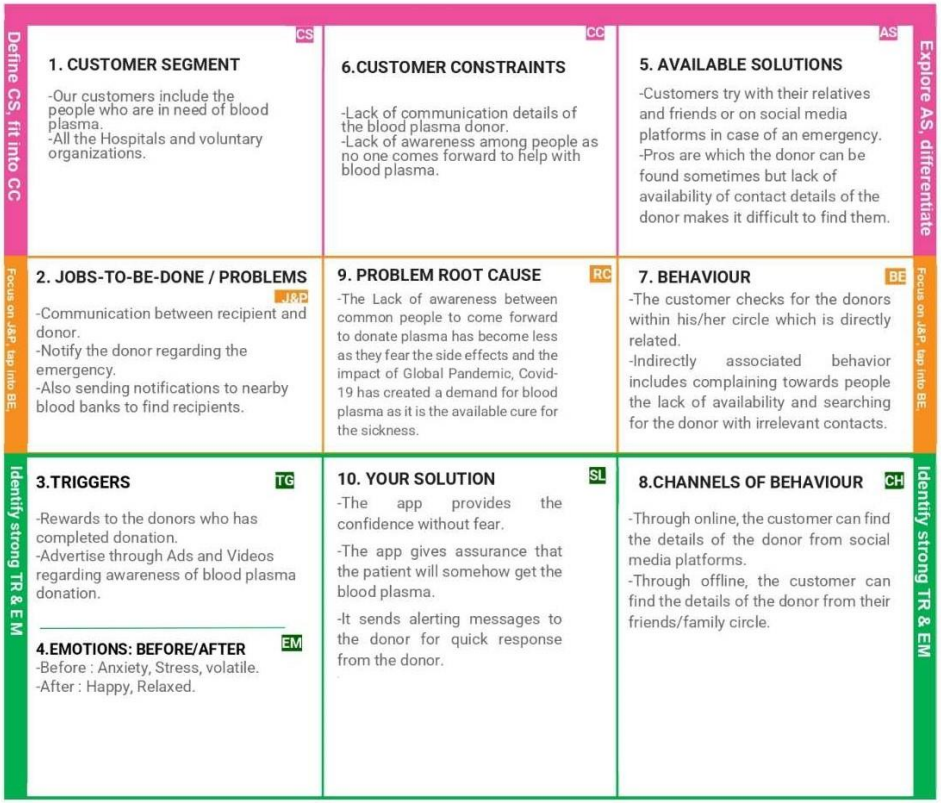
Date	10 October 2022
Team ID	lBMSI20220014320
Project Name	Project – Plasma Donor Application
Maximum Marks	2 Marks

Proposed Solution :

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Many major medical conditions are treated by plasma. For this reason, blood drives are held to solicit donations of plasma and blood. One of the most well-known techniques known as plasma treatment, plasma is used to cure various incurable diseases. Plasma therapy had a high success rate but a very low donor count during these times, thus it was crucial to learn more about the donors. It would be helpful to save the contributor data and provide information about the ongoing donors because it will help clients find the crucial contributor data faster and save time.
2.	Idea / Solution description	This system's goal is to use an online application to link donors and patients. Users of this application may post requests for plasma donations or requests for services. The fundamental solution is to establish a centralized system to keep track of current and previous Plasma Donation Events. The suggested solution is as follows: This application has two roles: ■ Admin ■ User

		User: • Users must register with their personal information in order to give or receive. • Following a user's successful registration. • The user receives an email upon successful registration. • The user will be taken to the home page after a successful registration. • If they want to be a donor or a recipient, they must press. • If the user is a donor, they must fill out the donation interest form with their name, blood group information, location, the date they last contributed, a phone number, and an email address. • The donor will be sent to a page where they can obtain the e-certificate after completing the donation form. • If the user is a recipient, he or she can raise a request and get in touch with the donor immediately after viewing the list of possible donors. Admin: • Administrators can log in using their credentials. • Admin may change the request. • The request may be revoked by the administrator. • The admin can add volunteers.

3.4 PROBLEM SOLUTION FIT



4.REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

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	User Examination	Medical Examination before donating
FR-5	Recipient Request	The recipient makes request for blood type for plasma
FR-6	Donor Request Alert	The Donor gets alerted through email
FR-7	Closed Request Verification	Donor gets an e-certificate and rewards once donation is completed
FR-8	Videos and Donation camps	Users can look up the benefits of plasma donation and information related
FR-9	Chat Assistant	Helps to solve queries related to donation within the app

4.2 NON-FUNCTIONAL REQUIREMENT

Non-functional Requirements:

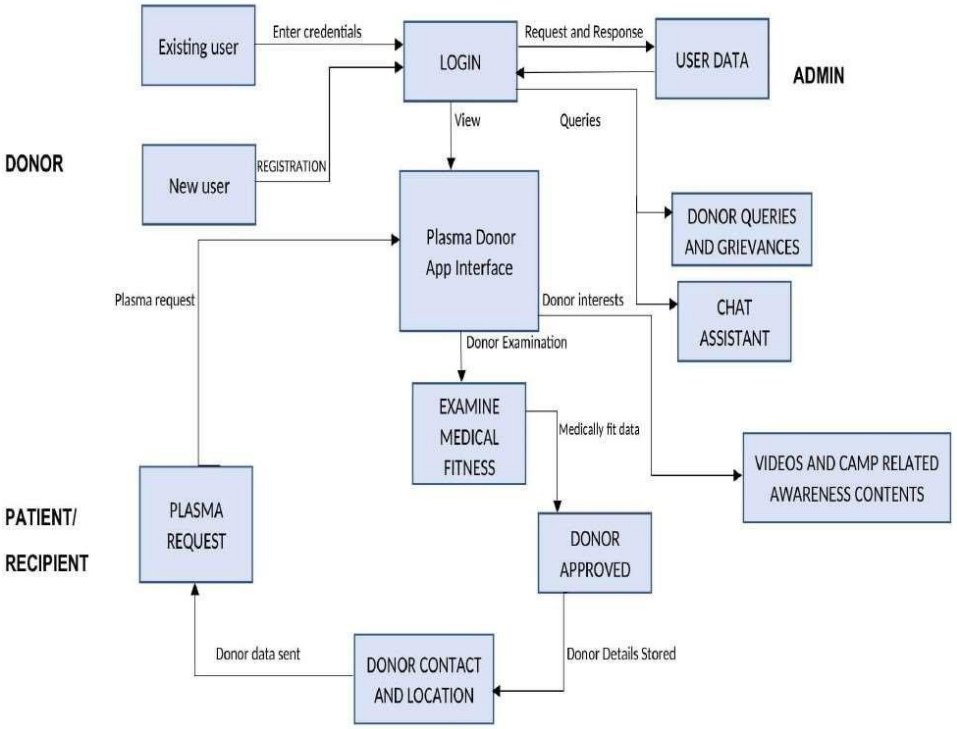
Following are the non-functional requirements of the proposed solution.

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 centres 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 App 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

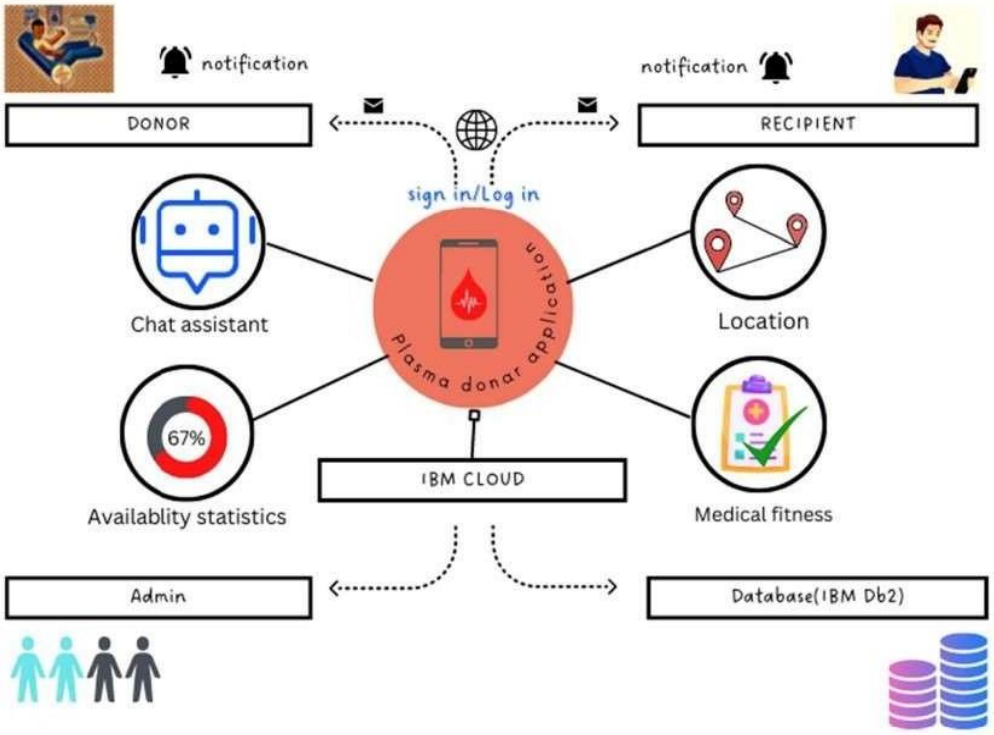
5.1 DATA FLOW DIAGRAM

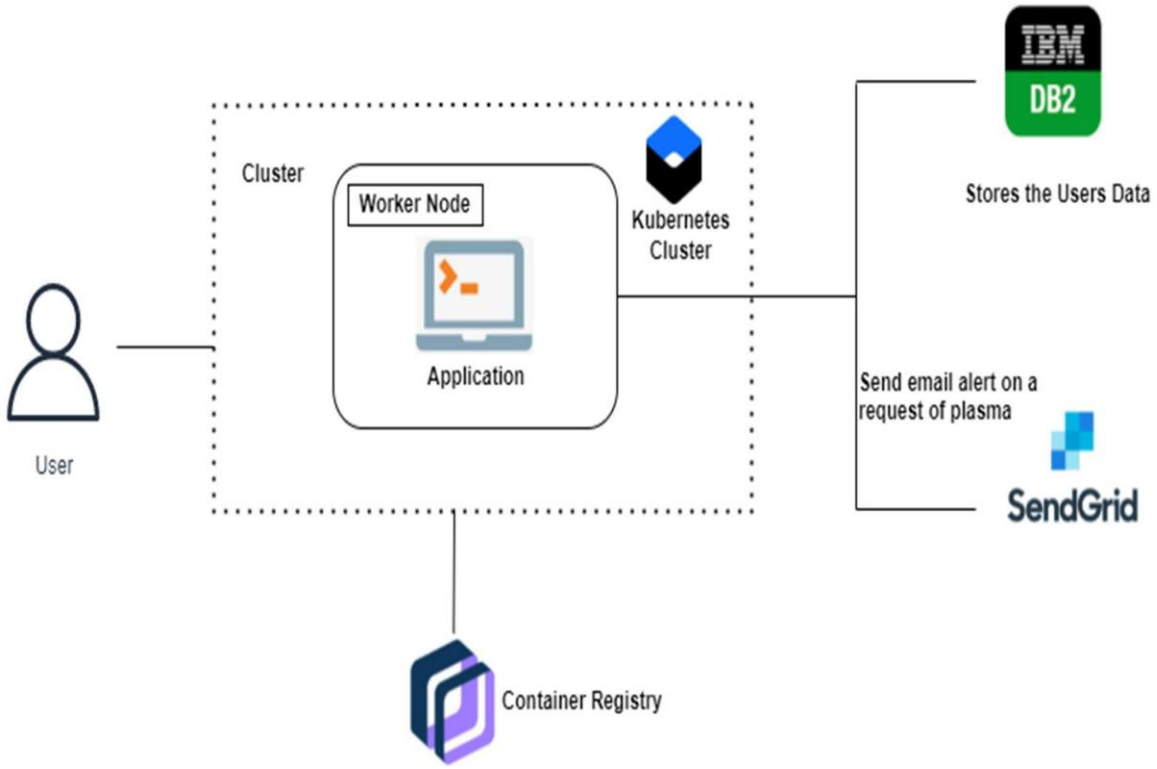
Data Flow Diagrams:



5.2 SOLUTION AND TECHNICAL ARCHITECTURE

Plasma Donor Application:





5.3 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

6.PROJECT PLANNING AND SCHEDULE

6.1 SPRINT PLANNING & ESTIMATION

Project Planning Phase

Sprint Delivery Plan (Product Backlog, Sprint Planning, Stories, Story points)

Date	18 October 2022
Team ID	PNT2022TMID14320
Project Name	Project - PLASMA DONOR APPLICATION
Maximum Marks	8 Marks

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

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 for the application by entering my email, password, and confirming my password.	4	High	Lankipalli Nitheesh Kumar
Sprint-1	Email Confirmation	USN-2	As a user, I will receive confirmation email once I have registered for the application	4	High	Matcha Chandra Kiran
Sprint-1	Registration	USN-3	As a user, I can register for the application through Gmail and other Email services	2	Medium	NR Mukesh
Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	4	High	NR Mukesh
Sprint-1	Profile	USN-5	As a user, I am able to register myself as a registered plasma donor and view my profile page.	4	High	Pinnamareddy Sai Tharun Reddy
Sprint-2	Social Media	USN-6	As a user, I can link and register to the application through social media accounts	2	Low	Matcha Chandra Kiran
Sprint-2	Virtual Donor Badge	USN-7	As a user, I can receive a virtual donor badge once I am successfully registered.	4	Medium	Lankipalli Nitheesh Kumar
Sprint-2	Plasma Request	USN-8	As a user, I can place a plasma request or donate plasma. I will include the Hospital details with the request.	4	High	Matcha Chandra Kiran
Sprint-2	Verifying Request	USN-9	As a user, I will wait until my request is verified through Administrators of the app. (We Admins	4	High	P Sai Tharun Reddy

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
			will verify the request after confirming with the concerned Hospital)			NR mukesh
Sprint-2	Verifying Donor	USN-10	As a user, I will wait until my donorship is verified through administrators of the app. (We Admins will verify the donor from a list of registered donors and share his details to the requester.)	4	High	Lankipalli Nitheesh Kumar
Sprint-3	Donation Alarm	USN-11	The Registered Donor is notified with an alarm and a message regarding the request.	5	High	Pinnamareddy Sai Tharun Reddy
Sprint-3	Accept the Request	USN-12	As a Donor, I will accept the plasma request based on my interest and volunteer for the donation.	4	Medium	NR Mukesh
Sprint-3	Communication Channel	USN-13	The Communication details of the donor will be sent to the Requester and vice versa. The Requester can personally communicate with the Donor. (Details of the donor will be provided according to the level of urgency)	5	High	Lankipalli Nitheesh Kumar
Sprint-3	Donor Details	USN-14	The details of the volunteered donor are stored in the database.	4	Medium	Matcha Chandra Kiran
Sprint-4	Support	USN-15	As a user, I can chat with a chatbot regarding my queries and doubts.	3	Medium	Lankipalli Nitheesh Kumar
Sprint-4	Grievances and FAQ	USN-16	As a user, I can post my worries and grievances in the comment section. I can also find Frequently asked Questions with answers in the FAQ section.	3	Medium	Piannama reddy Sai Tharun Reddy
Sprint-4	Certificate and Rewards	USN-17	As a donor, I will receive an e-certificate after donations. Virtual rewards are also provided to the donor.	3	Low	Lankipalli Nitheesh Kumar NR Mukesh
Sprint-4	About	USN-18	As a user, I will find about the importance of plasma donation in this section of the application.	3	Medium	Matcha Chandra Kiran
Sprint-4	Administrator		We admins will approve all the plasma transaction in the application after proper verification.	3	High	Pinnamareddy Sai Tharun Reddy

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
						Lankipalli Nitheesh Kumar NR Mukesh Matcha Chandra kiran
Sprint-4			We admins will update the plasma availability and donor count periodically.	3	Medium	Pinnamareddy Sai Tharun Reddy Lankipalli Nitheesh Kumar NR Mukesh Matcha Chandra kiran
Sprint-4			We admins will give fine touch to the application based on any updates needed in the future.	3	Medium	Pinnamareddy Sai Tharun Reddy Lankipalli Nitheesh Kumar NR Mukesh Matcha Chandra kiran

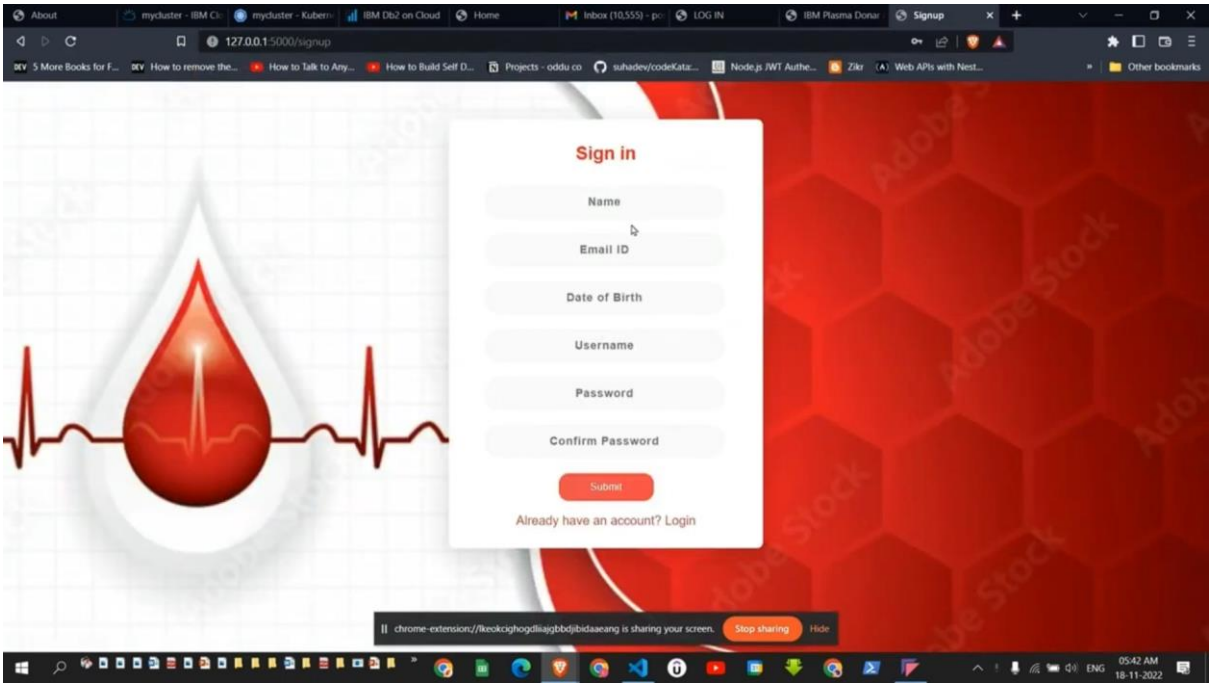
6.2 SPRINT DELIVERY SCHEDULE

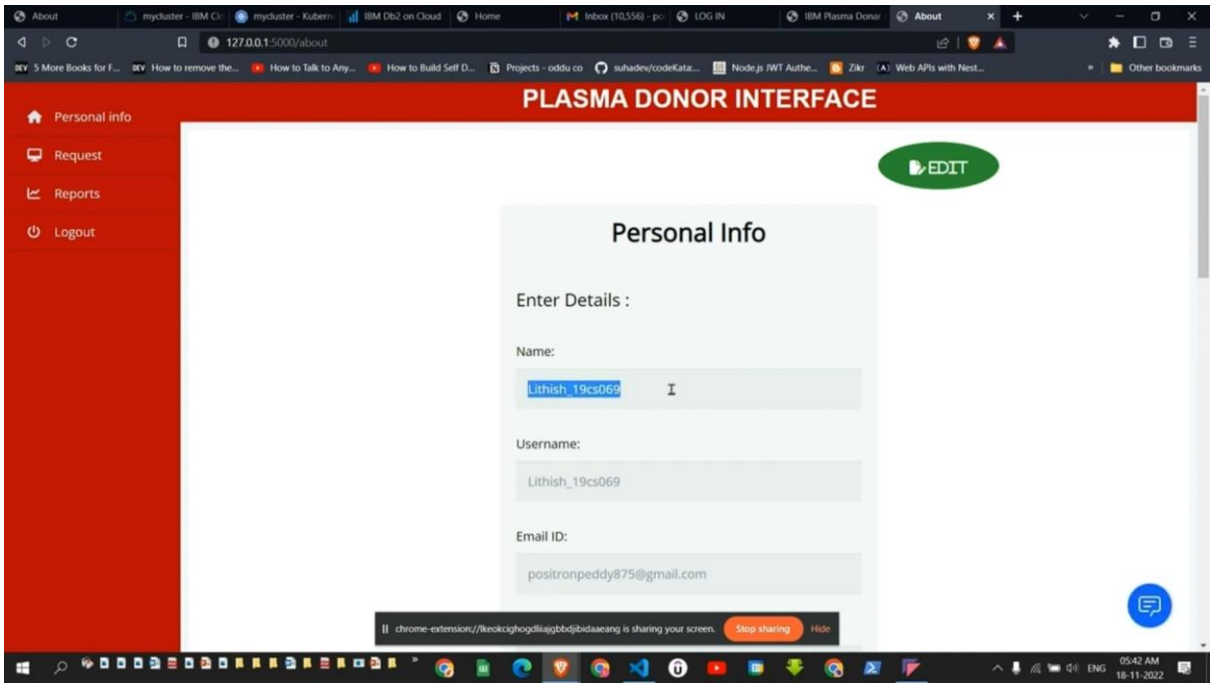
Project Tracker, Velocity & Burndown Chart: (4 Marks)

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	18	6 Days	24 Oct 2022	29 Oct 2022	18	29 Oct 2022
Sprint-2	18	6 Days	31 Oct 2022	05 Nov 2022	18	05 Nov 2022
Sprint-3	18	6 Days	07 Nov 2022	12 Nov 2022	18	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

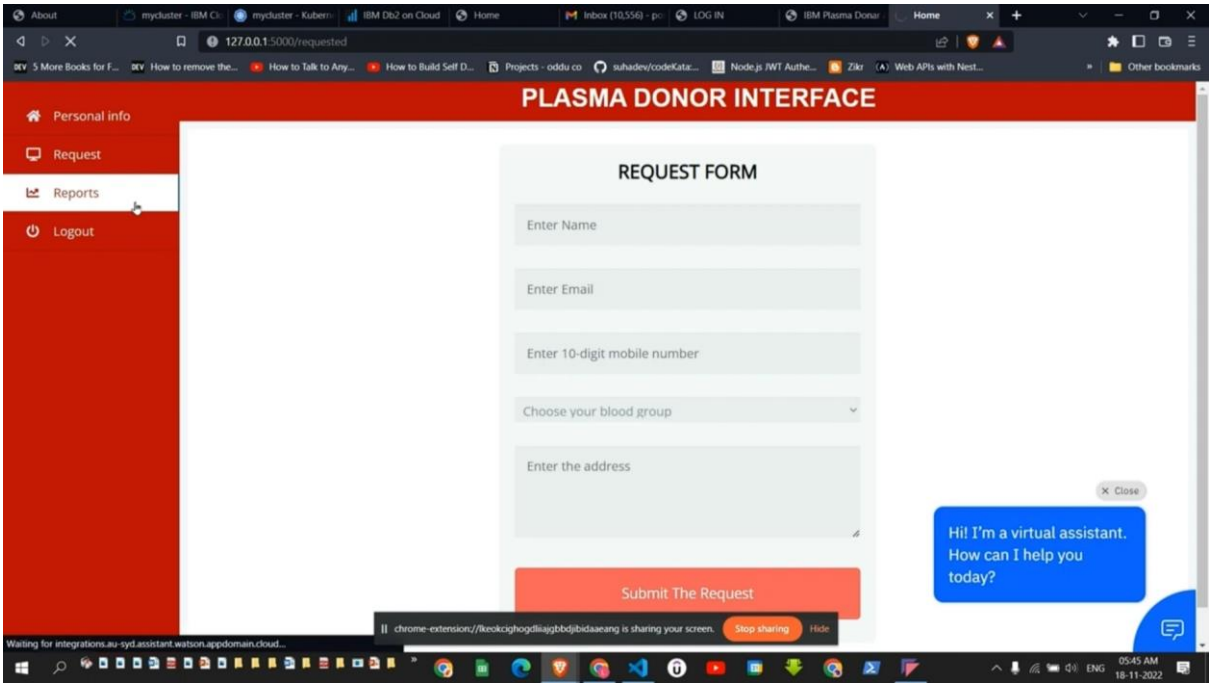
7. CODING AND SOLUTIONING

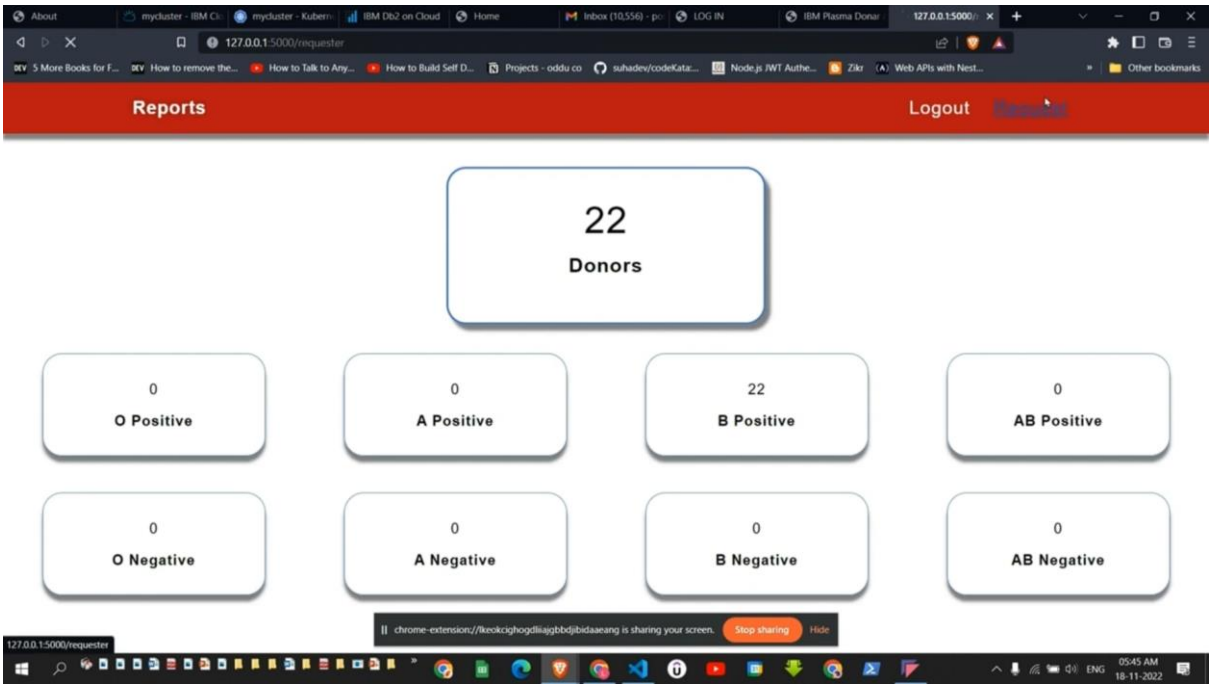
7.1 FEATURE 1



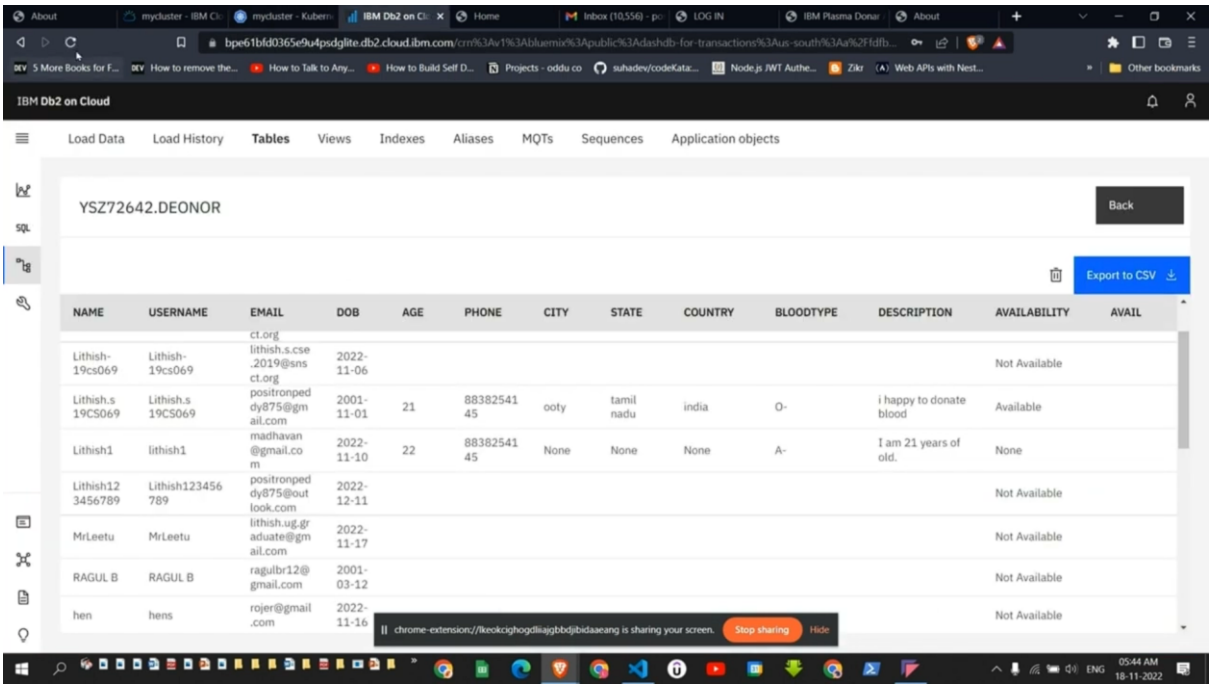

PLASMA DONOR INTERFACE
Personal info
Request
Reports
Logout
EDIT
Personal Info
Enter Details :
Name:
Lithish_19cs069
Username:
Lithish_19cs069
Email ID:
positronpeddy875@gmail.com
chrome-extension://ikeokcighogdliiajgbbdjibidaaeang is sharing your screen.
Stop sharing
Hide

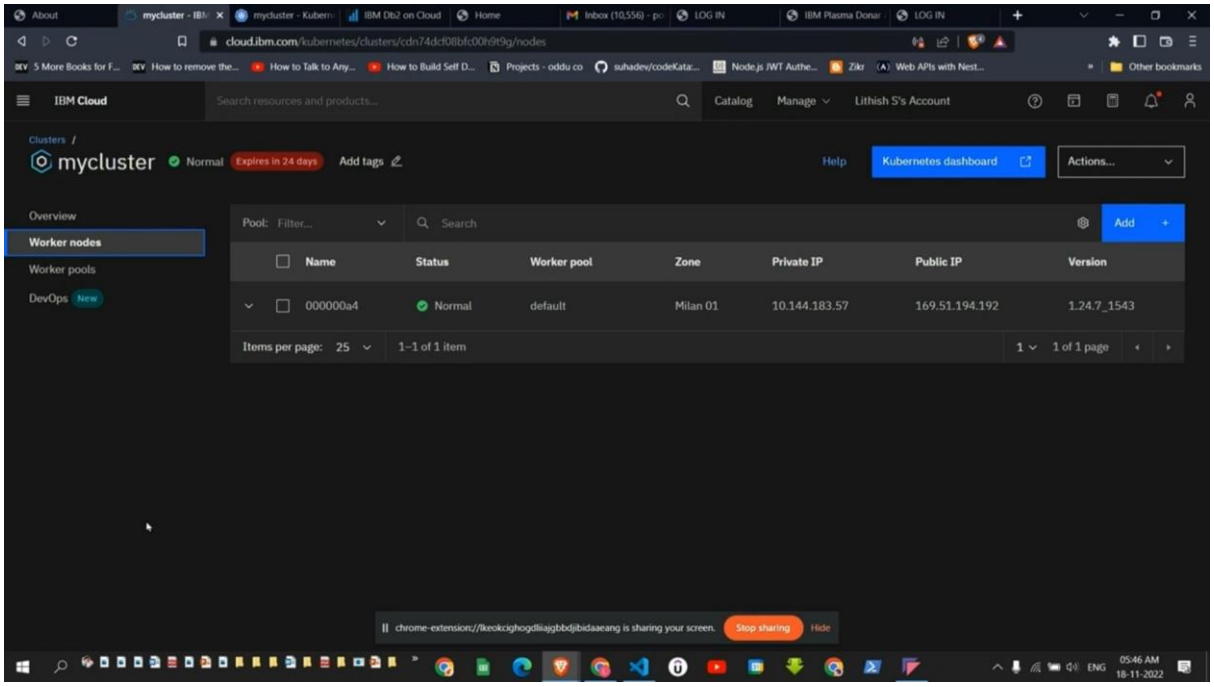
7.2 FEATURE 2

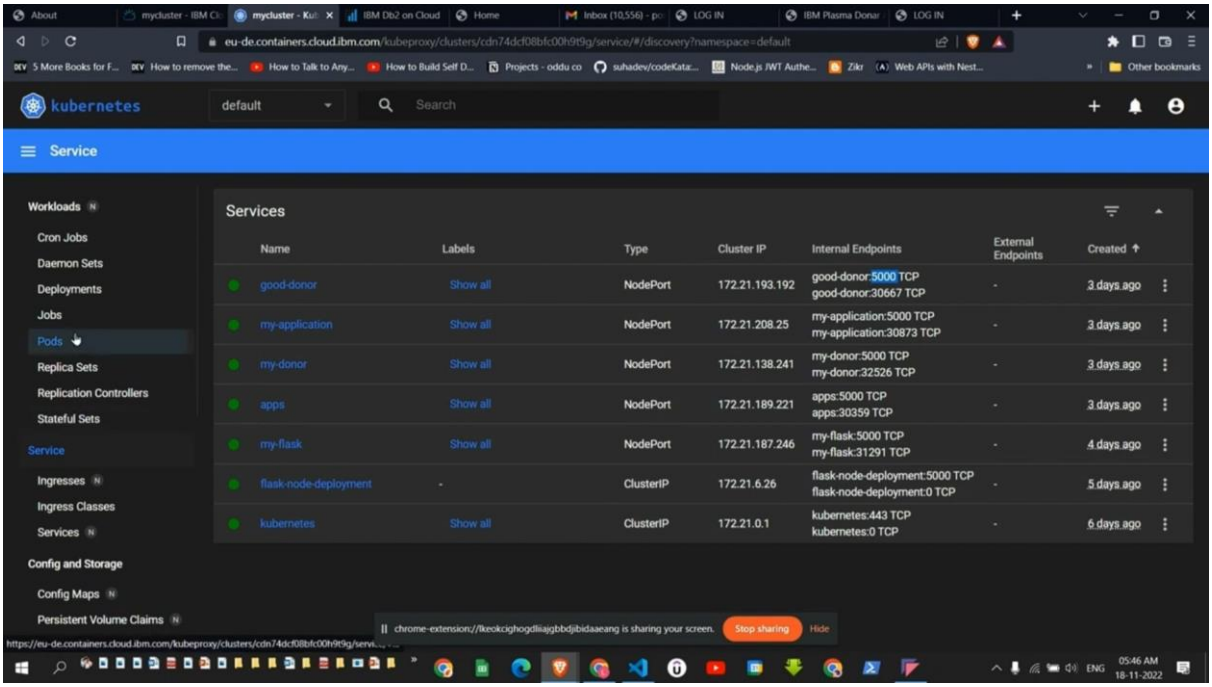

PLASMA DONOR INTERFACE
Personal info
Request
Reports
Logout
REQUEST FORM
Enter Name
Enter Email
Enter 10-digit mobile number
Choose your blood group
Enter the address
Submit The Request
Hi! I'm a virtual assistant. How can I help you today?


Reports
Logout
22
Donors
0
O Positive
0
A Positive
22
B Positive
0
AB Positive
0
O Negative
0
A Negative
0
B Negative
0
AB Negative

7.3 DATABASE SCHEMA




cloud.ibm.com/kubernetes/clusters/cdn74dcf08bfc00h9t9g/nodes
IBM Cloud
Search resources and products...
Catalog
Manage
Lithish S's Account
Clusters /
mycluster
Normal
Expires in 24 days
Add tags
Help
Kubernetes dashboard
Actions...
Overview
Worker nodes
Worker pools
DevOps
New
Pool: Filter...
Search
Add
Name
Status
Worker pool
Zone
Private IP
Public IP
Version
000000a4
Normal
default
Milan 01
10.144.183.57
169.51.194.192
1.24.7_1543
Items per page: 25
1–1 of 1 item
1 of 1 page
chrome-extension://lkeokcighogdliiajgbbdjibidaaeang is sharing your screen.
Stop sharing
Hide
ENG
05:46 AM
18-11-2022


eu-de.containers.cloud.ibm.com/kubeproxy/clusters/cdn74dcf08bfc00h9t9g/service/#/discovery?namespace=default
kubernetes
default
Search
Service
Workloads
Cron Jobs
Daemon Sets
Deployments
Jobs
Pods
Replica Sets
Replication Controllers
Stateful Sets
Service
Ingresses
Ingress Classes
Services
Config and Storage
Config Maps
Persistent Volume Claims
Services
Name
Labels
Type
Cluster IP
Internal Endpoints
External Endpoints
Created
good-donor
Show all
NodePort
172.21.193.192
good-donor:5000 TCP
good-donor:30667 TCP
3 days ago
my-application
Show all
NodePort
172.21.208.25
my-application:5000 TCP
my-application:30873 TCP
3 days ago
my-donor
Show all
NodePort
172.21.138.241
my-donor:5000 TCP
my-donor:32526 TCP
3 days ago
apps
Show all
NodePort
172.21.189.221
apps:5000 TCP
apps:30359 TCP
3 days ago
my-flask
Show all
NodePort
172.21.187.246
my-flask:5000 TCP
my-flask:31291 TCP
4 days ago
flask-node-deployment
-
ClusterIP
172.21.6.26
flask-node-deployment:5000 TCP
flask-node-deployment:0 TCP
5 days ago
kubernetes
Show all
ClusterIP
172.21.0.1
kubernetes:443 TCP
kubernetes:0 TCP
6 days ago
chrome-extension://lkeokcighogdliiajgbbdjibidaaeang is sharing your screen.
Stop sharing
Hide
ENG
05:46 AM
18-11-2022

8.TESTING

8.1 TEST CASES

8.1.1 FUNCTIONAL TESTING

Functional test can be defined as testing two or more modules together with the intent of finding defects, demonstrating that defects are not present, verifying that the module performs its intended functions as stated in the specification and establishing confidence that a program does what it is supposed to do.

8.1.2 WHITE BOX TESTING:

Testing based on an analysis of internal workings and structure of a piece of software. This testing can be done sing the percentage value of load and energy. The tester should know what exactly is done in the internal program. Includes techniques such as Branch Testing and Path Testing. Also known as Structural Testing and Glass Box Testing.

8.1.3 BLACK BOX TESTING:

Testing without knowledge of the internal workings of the item being tested. Tests are usually functional. This testing can be done by the user who has no knowledge of how the shortest path is found.

8.2 USER ACCEPTANCE TESTING

Acceptance testing can be defined in many ways, but a simple definition is the succeeds when the software functions in a manner that can be reasonable expected by the customer. After the acceptance test has been conducted, one of the two possible conditions exists. This is to fine whether the inputs are accepted by the database or other validations. For example accept only numbers in the numeric field, date format data in the date field. Also the null check for the not null fields. If any error occurs then show the error messages. The function of performance characteristics to specification and is accepted. A deviation from specification is uncovered and a deficiency list is created. User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

8.3 TEST CASES

TEST CASE ID	Feature TYPE	Component	Test Scenario	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	Bug ID
TC-1	Functional	Home Page	Top Navigation bar login Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-2	Functional	Home Page	Top Navigation bar Home Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-3	UI	Home Page	"Plasma Donation" Letters in Green	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-4	UI	Home Page	Bgnd Blood Image and centre image	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-5	Functional	Home Page	Top next Nav_bar Login	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-6	Functional	Home Page	Top next Nav_bar Register	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-7	Functional	Home Page	Top next Nav_bar About	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-8	Functional	Home Page	Top next Nav_bar Queries	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-9	Functional	Home Page	Chatbot	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-10	Functional	Login Page	Top Left Home Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-11	UI	Login Page	Login Page' Letters	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-12	Functional	Login Page	Email Enter tab	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-13	Functional	Login Page	Password Enter Tab	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-14	Functional	Login Page	Login Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-15	UI	Login Page	Dummy Social Media Buttons	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-16	UI	Login Page	Background Heart image	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-17	Functional	Login Page	Wrong credentials' if entered wrong	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-18	Functional	Login Page	"Account Already exists" if same email	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-19	Functional	Login Page	Redirect to Accounts if entered right	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-20	Functional	Login User Interface	Top Navigation bar login Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-21	Functional	Login User Interface	Top Navigation bar Home Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-22	UI	Login User Interface	"Plasma Donation" Letters in Green	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-23	Functional	Login User Interface	Top next Nav_bar Active button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-24	Functional	Login User Interface	Top next Nav_bar Inactive button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-25	Functional	Login User Interface	Top next Nav_bar Request button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-26	Functional	Login User Interface	Top next Nav_bar Profile	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-27	Functional	Login User Interface	Top next Nav_bar Logout	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-28	UI	Login User Interface	Centre image	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-29	UI	Login User Interface	Login user name	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-30	UI	Login User Interface	Login user Welcome message	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-31	UI	Login User Interface	Login user footer Message	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-32	UI	Login User Interface	White background	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-33	Functional	Login User Interface	Chatbot	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-34	Functional	Active Donors Page	Show Details of active donors	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-35	Functional	Active Donors Page	Download data in Excel button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-36	Functional	Active Donors Page	Download data in PDF button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-37	Functional	Active Donors Page	Print Data Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-38	Functional	Active Donors Page	CSV Data Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-39	Functional	Active Donors Page	Copy Data Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-40	Functional	Active Donors Page	Inactive Donors Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-41	Functional	Active Donors Page	Dashboard button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-42	Functional	Active Donors Page	Logout Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-43	Functional	Active Donors Page	Search Tab	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-44	Functional	Active Donors Page	Chatbot	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-45	Functional	Inactive Donors page	Show Details of active donors	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-46	Functional	Inactive Donors page	Download data in Excel button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-47	Functional	Inactive Donors page	Download data in PDF button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-48	Functional	Inactive Donors page	Print Data Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-49	Functional	Inactive Donors page	CSV Data Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-50	Functional	Inactive Donors page	Copy Data Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-51	Functional	Inactive Donors page	Inactive Donors Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-52	Functional	Inactive Donors page	Dashboard button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-53	Functional	Inactive Donors page	Logout Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-54	Functional	Inactive Donors page	Search Tab	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-55	Functional	Request Page	Top next Nav_bar Active button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-56	Functional	Request Page	Top next Nav_bar Inactive button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-57	Functional	Request Page	Top next Nav_bar Dashboard button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-58	Functional	Request Page	Top next Nav_bar Profile button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-59	Functional	Request Page	Top next Nav_bar Logout button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-60	Functional	Request Page	Request or Details Getting Tabs	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-61	Functional	Request Page	Place Request Buttons	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-62	Functional	Request Page	"Request Placed Successfully"message	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-63	UI	Request Page	Footer image	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-64	Functional	Request Page	Chatbot	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-65	Functional	Donor Page	Top next Nav_bar Active button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-66	Functional	Donor Page	Top next Nav_bar Inactive button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-67	Functional	Donor Page	Top next Nav_bar Dashboard button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-68	Functional	Donor Page	Top next Nav_bar Profile button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-69	Functional	Donor Page	Top next Nav_bar Logout button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-70	Functional	Donor Page	Donor Willingness form	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-71	Functional	Donor Page	Ready To Donate Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-72	Functional	Donor Page	Donor Badge on Top next Nav_bar	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-73	Functional	Donor Page	Chatbot	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-74	Functional	Profile Page	Top Nav_bar HOME &LOGIN buttons	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-75	Functional	Profile Page	User Details	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-76	UI	Profile Page	Footer image	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-77	UI	Profile Page	Dummy Social Media Buttons	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-78	Functional	Profile Page	Chatbot	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-79	Functional	Ready to Donate Page	Top Nav_bar HOME &LOGIN buttons	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-80	Functional	Ready to Donate Page	Dashboard button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-81	Functional	Ready to Donate Page	Logout Button	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-82	UI	Ready to Donate Page	Dummy Certificate Button	http://127.0.0.1:5000/	Not Responsive	Working as expected	Pass	Successful	Y	Nil
TC-83	Functional	Ready to Donate Page	Chatbot	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil
TC-84	Functional	Logout	Logout Process	http://127.0.0.1:5000/	Responsive	Working as expected	Pass	Successful	Y	Nil

9.RESULTS

9.1 PERFORMANCE METRICES

PERFORMANCE TESTING REPORT

NFT - Risk Assessment

1	Plasma Donor Applica	Existing	Low	No Changes	Moderate		>5 to 10%	GREEN	As we have seen the changes
2	Plasma Donor Applica	Existing	Low	No Changes	High		>5 to 10%	GREEN	As we have seen the changes
3	Plasma Donor Applica	Existing	Low	No Changes	Moderate		>5 to 10%	ORANGE	As we have seen the changes
4	Plasma Donor Applica	Existing	Low	No Changes	Moderate		>5 to 10%	GREEN	As we have seen the changes
5	Plasma Donor Applica	Existing	Low	No Changes	High		>5 to 10%	ORANGE	As we have seen the changes

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 verision 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 flask import Flask, render_template,request,redirect,url_for,session

import ibm_db

import sendgrid

import os

from sendgrid.helpers.mail import *

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=764264db-9824-4b7c-82df-
40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32536;SECURITY=SSL;S
SLServerCertificate=DigiCertGlobalRootCA.crt;UID=ydk44341;PWD=eENReIXIS7xOOBBa","")


app = Flask(__name__)

app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'


@app.route('/')

def home():

    return render_template('home.html')


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

def register():

    session['msg']=""

    if request.method == 'POST':

        name = request.form['name']

        email = request.form['email']

        phno = request.form['phno']

        dob = request.form['dob']

        gender = request.form['gender']

        bloodgroup = request.form['bloodgroup']

        weight = request.form['weight']

        password = request.form['newpassword']


        sql = "SELECT * FROM Members WHERE email =?"
```

```
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt,1,email)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)

    if account:
        session['msg']= 'Account already exists'
        return redirect(url_for("login"))
    else:
        insert_sql = "INSERT INTO Members VALUES (?,?,?,?,?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, name)
        ibm_db.bind_param(prep_stmt, 2, email)
        ibm_db.bind_param(prep_stmt, 3, phno)
        ibm_db.bind_param(prep_stmt, 4, dob)
        ibm_db.bind_param(prep_stmt, 5, gender)
        ibm_db.bind_param(prep_stmt, 6, bloodgroup)
        ibm_db.bind_param(prep_stmt, 7, weight)
        ibm_db.bind_param(prep_stmt, 8, password)
        ibm_db.execute(prep_stmt)
        session['msg']= 'Account created Successfully '
        sg = sendgrid.SendGridAPIClient(api_key=os.environ.get('SENDGRID_API_KEY'))
        from_email = Email("matc19cs076@rmkcet.ac.in")
        to_email = To(email)
        subject = "Plasma Donor App"
        content = Content("text/plain", "You are successfully registered to Plasma Donor App")
        mail = Mail(from_email, to_email, subject, content)
        response = sg.client.mail.send.post(request_body=mail.get())
        print(response.status_code)
        print(response.body)
        print(response.headers)
        return redirect(url_for("login"))


return render_template('register.html')
```

```
@app.route('/login',methods=['GET', 'POST'])
def login():

    if request.method == 'POST':
        email = request.form['email']
        password = request.form['newpassword']



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

        session['user'] = account
        accounts=account

        if (account):
            if  (password == accounts['PASSWORD'] ):
                return render_template('accounts.html',name=account['NAME'])
            else :
                return render_template('login.html',msg='wrong Password')
        else :
            return render_template('login.html',msg='wrong credentials')



    else:
        return  render_template('login.html')

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

@app.route('/view2')
def view2():
```

```
    return render_template('view2.html')

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

@app.route('/request1',methods=['GET', 'POST'])
def request1():
    session['msg']=""
    if request.method == 'POST':
        patientname = request.form['patientname']
        bloodgroupneeded = request.form['bloodgroupneeded']
        reasonforneed = request.form['reasonforneed']
        hospitalname = request.form['hospitalname']
        hospitaladdress = request.form['hospitaladdress']
        hospitalno = request.form['hospitalno']
        patientgender = request.form['patientgender']
        contactno = request.form['contactno']
        email = request.form['email']
        insert_sql = "INSERT INTO Requesters VALUES (?,?,?,?,?,?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, patientname)
        ibm_db.bind_param(prep_stmt, 2, bloodgroupneeded)
        ibm_db.bind_param(prep_stmt, 3, reasonforneed)
        ibm_db.bind_param(prep_stmt, 4, hospitalname)
        ibm_db.bind_param(prep_stmt, 5, hospitaladdress)
        ibm_db.bind_param(prep_stmt, 6, hospitalno)
        ibm_db.bind_param(prep_stmt, 7, patientgender)
        ibm_db.bind_param(prep_stmt, 8, contactno)
        ibm_db.bind_param(prep_stmt, 9, email)
        ibm_db.execute(prep_stmt)
        session['msg']= 'Request Placed Successfully'
        sg = sendgrid.SendGridAPIClient(api_key=os.environ.get('SENDGRID_API_KEY'))
        from_email = Email("matc19cs076@rmkcet.ac.in")
        to_email = To(email)
        subject = "Plasma Donor App-Request"
```

```
        content = Content("text/plain", "Your Request is under review")
        mail = Mail(from_email, to_email, subject, content)
        response = sg.client.mail.send.post(request_body=mail.get())
        print(response.status_code)
        print(response.body)
        print(response.headers)

    return render_template('request.html')

@app.route('/donate',methods=['GET','POST'])
def donate():
    session['msg']=""
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        fitness = request.form['fitness']
        fitreason = request.form['fitreason']
        disease = request.form['disease']
        specdisease = request.form['specdisease']
        vaccination = request.form['vaccination']
        vaccinationdate = request.form['vaccinationdate']
        agreement = request.form['agreement']
        sharecontact = request.form['sharecontact']


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

        if account:
            session['msg']= 'Already registered for Donation'
            return redirect(url_for("donate"))
        else:
            insert_sql = "INSERT INTO Donors VALUES (?,?,?,?,?,?,?,?,?,?)"
```

```
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, fitness)
            ibm_db.bind_param(prep_stmt, 4, fitreason)
            ibm_db.bind_param(prep_stmt, 5, disease)
            ibm_db.bind_param(prep_stmt, 6, specdisease)
            ibm_db.bind_param(prep_stmt, 7, vaccination)
            ibm_db.bind_param(prep_stmt, 8, vaccinationdate)
            ibm_db.bind_param(prep_stmt, 9, agreement)
            ibm_db.bind_param(prep_stmt, 10, sharecontact)
            ibm_db.execute(prep_stmt)
            return redirect(url_for("readytodonate", msg='Your Donorship is under review'))

    return render_template('donate.html')

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

@app.route('/profile')
def profile():
    return render_template('profile.html',user=session['user'])

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

@app.route('/queries',methods=['GET', 'POST'])
def queries():
    session['msg']=""
    if request.method == 'POST':
        name = request.form['name']
        email = request.form['email']
        contents = request.form['contents']
        insert_sql = "INSERT INTO Post VALUES (?,?,?)"
```

```
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, name)
        ibm_db.bind_param(prep_stmt, 2, email)
        ibm_db.bind_param(prep_stmt, 3, contents)
        ibm_db.execute(prep_stmt)
        session['msg']= 'Request Placed Successfully'
        return redirect(url_for("queries"))
    return render_template('queries.html')


if__name__== "__main__":
            app.run(host ='0.0.0.0', port = 5000, debug = True)
```

GITHUB & DEMO LINK:

GITHUB: https://github.com/IBM-EPBL/IBM-Project-42131-1660650833

DEMO LINK:

https://drive.google.com/file/d/1dZSEdzi1_J97Qnd8Zjllu0QgFg8kr0Ew/view?usp=share_link