IBM NALAIYATHIRAN PROJECT REPORT

PLASMA DONOR APPLICATION

Team Id	**PNT2022TMID02106**
Project Name	**Plasma Donor Application**
Team Members	**- Chinna Sakthi(2116190701029)** **- Eshwaran(2116190701046)** **- Abinav (2116190701005)** **- Gokul (2116190701052)**

Table Of Contents

1.INTRODUCTION

1.1 Project Overview

Plasma is a critical part of the treatment for many serious health problems. Therefore, there are blood drives asking people to donate blood plasma. The main goal of our project is to make it easier for the COVID-19 patients to get a plasma donor easily as well as donate plasma if they have recovered. The system targets two types of users: the people who want to donate plasma and the people who need plasma. The user can also view the total active cases, nearby vaccine centres, hospitals address.

The main objective of developing the website is to make it easier for the COVID-19 patients to get a plasma donor easily and as soon as possible. Yet, the need for plasma-derived products has been strongly increasing for some years, and blood collection agencies have to adapt if they want to meet this demand. This article aims to review the main motivations and deterrents to whole blood donation, and to compare them with those that we already know concerning plasma donation. Current evidence shows similarities between both behaviours, but also differences that indicate a need for further research regarding plasma donation.

1.2 Purpose

During the COVID 19 crisis, the requirement of plasma became a high priority, and the donor count has become low.

Saving the donor information and helping the needy by notifying the current donors list, would be a helping hand. Regarding the problem faced, an application is to be built which would take the donor details, store them and inform them upon a request.

2.LITERATURE SURVEY

2.1 EXIXTING PROBLEM

- Only mobile based system is available web-based system is available
- Less Security
- No proper coordination between different applications and users
- Cannot upload and download the latest updates at right time Fewer users-friendly

2.2 REFERENCE

Several experiments have been carried out over the years by different groups of researchers. Here are some of the following groups:

[1] Denuis O'Neil (1999). "Blood component" Archived from the original on June 5, 2013.

[2] ways to keep your plasma healthy, Original Archived November 1, 2013, Accessed November 11, 2011.

[3] Ripathis S, Kumar V, Prabhakar A, Joshi S, Agarwal A (2015). "Microscale Passive Plasma Separation: A Review of Design Principles and Microdevices," J. Micromech Micro 25 (8): 083001;

[4] P. C. P. C. a. V. I. M. Yan, "Building a chatbot with server less computing," IBM watson research center, 2016.

[5] S. E. a. B. J. J. Short, ""Cloud Event Programming Paradigms: Applications and Analysis,"," 9th IEEE International Conference on Cloud Computing (CLOUD), pp. pp. 4 00-406, 2017.

2.3 Problem Statement Definition

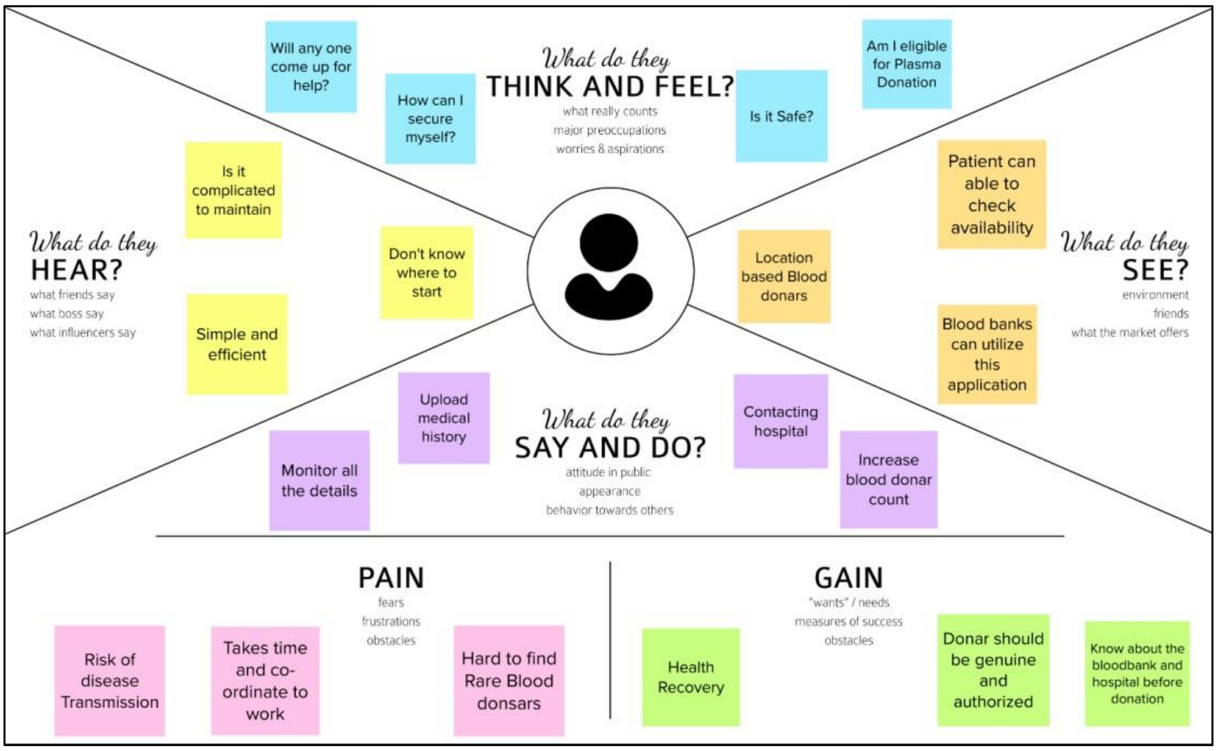
During COVID 19 crisis the requirement for plasma increased drastically as there were no vaccinations found in order to treat the infected patients.

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.

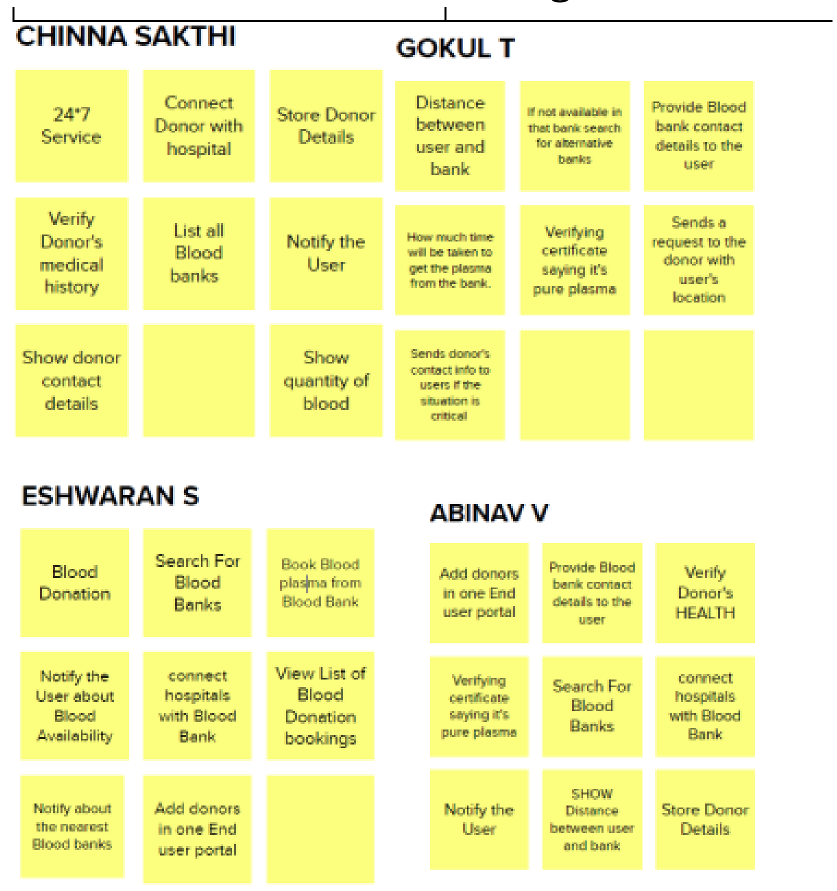
As the plasma therapy was one of the ways to treat the infected patients getting the donor details played a major role.

3. IDEATION AND PROPOSED SYSTEM

3.1 Empathy Map Canvas



3.2 Ideation and Brainstorming


CHINNA SAKTHI
24*7 Service
Connect Donor with hospital
Store Donor Details
Verify Donor's medical history
List all Blood banks
Notify the User
Show donor contact details
Show quantity of blood
GOKUL T
Distance between user and bank
If not available in that bank search for alternative banks
Provide Blood bank contact details to the user
How much time will be taken to get the plasma from the bank.
Verifying certificate saying it's pure plasma
Sends a request to the donor with user's location
Sends donor's contact info to users if the situation is critical
ESHWARAN S
Blood Donation
Search For Blood Banks
Book Blood plasma from Blood Bank
Notify the User about Blood Availability
connect hospitals with Blood Bank
View List of Blood Donation bookings
Notify about the nearest Blood banks
Add donors in one End user portal
ABINAV V
Add donors in one End user portal
Provide Blood bank contact details to the user
Verify Donor's HEALTH
Verifying certificate saying it's pure plasma
Search For Blood Banks
connect hospitals with Blood Bank
Notify the User
SHOW Distance between user and bank
Store Donor Details

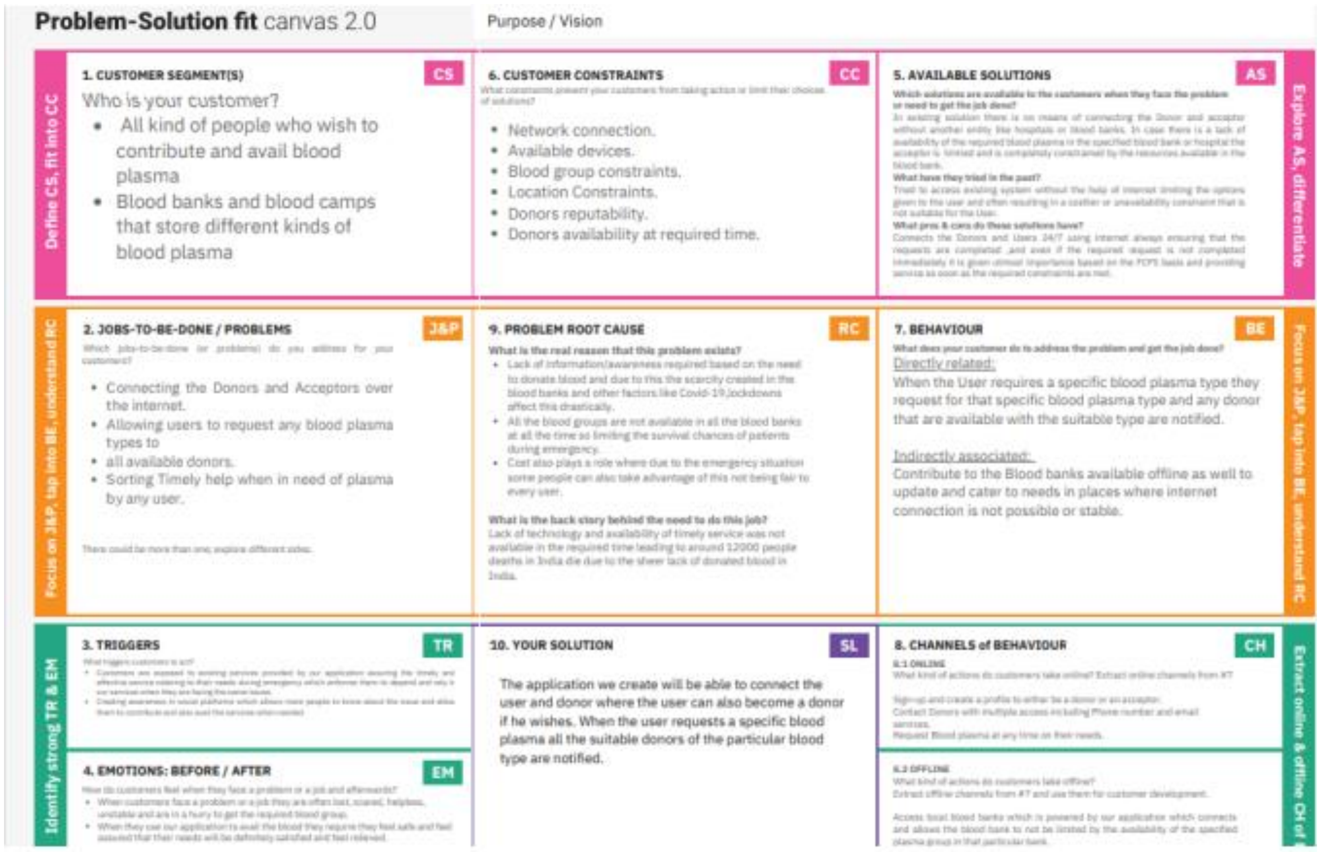
3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	1.During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low. 2.Saving the donor information and helping the needy by notifying the current donors list has become an important issue. 3.We have to create an application that can connect the User and the Donor seamlessly

2.	Idea / Solution description	.The application we create will be able to connect the user and donor where the user can also become a donor if he wishes. 2.When the user requests a specific blood plasma all the suitable donors of the particular blood type are notified
3.	Novelty / Uniqueness	1.User-friendly interface with an efficient, fast , and seamless connectivity between donor and acceptor. 2.Creates a Plasma donation community that has both contributors and end users that are equally profited and create a sense of safety and assurance when referring to their needs for immediate blood plasma requirement
4.	Social Impact / Customer Satisfaction	With the right implementation of the software you can benefit in many ways and also it makes the management very easy and error free. The software helps in tracking donors, getting Prompt and Correct Reports when required, and Centralized data storage with security. And last but not the least; the software will help in Customer Satisfaction.
5.	Business Model (Revenue Model)	1.Global connectivity that creates a community all over the world ensuring all the emergency needs are acknowledged and are catered to at the required time. 2.Establish a reliability factor of each user that ensures the delivery of service based on the user rating.

6.	Scalability of the Solution	Instead of scouring the entire world for plasma donors, this programme enables users to find donors while sitting at home. once there is an emergency, send a plasma request to all people. the donor is prepared to Donor recipient is informed of the donation. Receiver may get in touch with the donor. Due to this Donors can check their eligibility on an app as well as making it simpler to find a suitable donor.

3.4 Problem Statement Fit



Problem-Solution fit canvas 2.0

Purpose / Vision

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

Who is your customer?

- All kind of people who wish to contribute and avail blood plasma
- Blood banks and blood camps that store different kinds of blood plasma

6. CUSTOMER CONSTRAINTS CC

What constraints prevent your customers from taking action or limit their choices of solutions?

- Network connection.
- Available devices.
- Blood group constraints.
- Location Constraints.
- Donors reputability.
- Donors availability at required time.

5. AVAILABLE SOLUTIONS AS

Which solutions are available to the customers when they face the problem or need to get the job done?

In existing solution there is no means of connecting the Donor and acceptor without another entity like hospitals or blood banks. In case there is a lack of availability of the required blood plasma in the specified blood bank or hospital the acceptor is limited and is completely constrained by the resources available in the blood bank.

What have they tried in the past?

Tried to access existing system without the help of internet limiting the options given to the user and often resulting in a costlier or unavailability constraint that is not suitable for the User.

What pros & cons do these solutions have?

Connects the Donors and Users 24/7 using internet always ensuring that the requests are completed ,and even if the required request is not completed immediately it is given utmost importance based on the FCFS basis and providing service as soon as the required constraints are met.

Explore AS, differentiate

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

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

Which jobs-to-be-done (or problems) do you address for your customers?

- Connecting the Donors and Acceptors over the internet.
- Allowing users to request any blood plasma types to
- all available donors.
- Sorting Timely help when in need of plasma by any user.

There could be more than one; explore different sides.

9. PROBLEM ROOT CAUSE RC

What is the real reason that this problem exists?

- Lack of information/awareness required based on the need to donate blood and due to this the scarcity created in the blood banks and other factors like Covid-19 lockdowns affect this drastically.
- All the blood groups are not available in all the blood banks at all the time so limiting the survival chances of patients during emergency.
- Cost also plays a role where due to the emergency situation some people can also take advantage of this not being fair to every user.

What is the back story behind the need to do this job?

Lack of technology and availability of timely service was not available in the required time leading to around 12000 people deaths in India die due to the sheer lack of donated blood in India.

7. BEHAVIOUR BE

What does your customer do to address the problem and get the job done?

Directly related:

When the User requires a specific blood plasma type they request for that specific blood plasma type and any donor that are available with the suitable type are notified.

Indirectly associated:

Contribute to the Blood banks available offline as well to update and cater to needs in places where internet connection is not possible or stable.

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

Identify strong TR & EM

3. TRIGGERS TR

What triggers customers to act?

- Customers are exposed to existing services provided by our application assuring the timely and effective service catering to their needs during emergency which enforces them to depend and rely on our services when they are facing the same issues.
- Creating awareness in social platforms which allows more people to know about the issue and allow them to contribute and also avail the services when needed.

4. EMOTIONS: BEFORE / AFTER EM

How do customers feel when they face a problem or a job and afterwards?

- When customers face a problem or a job they are often lost, scared, helpless, unstable and are in a hurry to get the required blood group.
- When they use our application to avail the blood they require they feel safe and feel assured that their needs will be definitely satisfied and feel relieved.

10. YOUR SOLUTION SL

The application we create will be able to connect the user and donor where the user can also become a donor if he wishes. When the user requests a specific blood plasma all the suitable donors of the particular blood type are notified.

8. CHANNELS of BEHAVIOUR CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from #7

Sign-up and create a profile to either be a donor or an acceptor.

Contact Donors with multiple access including Phone number and email services.

Request Blood plasma at any time on their needs.

8.2 OFFLINE

What kind of actions do customers take offline?

Extract offline channels from #7 and use them for customer development.

Access local blood banks which is powered by our application which connects and allows the blood bank to not be limited by the availability of the specified plasma group in that particular bank.

Extract online & offline CH of

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)

FR-1	**Access Website**	Software operator should be capable to access web- application through either an application browser or similar on the pc.
FR-2	**Software operator Registration**	The software operator should be able to register through the web-application. The donor software operator must provide user name,gender,blood/plasma group,location,contact.
FR-3	**Login/logout/update details**	The login information will be stored on the database for future use.
FR-4	**Search for donor**	Search result can be viewed in a list.Each element in the list represents a specific donor with the donor details.
FR-5	**User plasma request**	Users can request to donate plasma by filling out the request form on the page. Once the request is submitted, they will get an email.
FR-6	**View distribution details**	The plasma bank should be able to view the status of the distribution details.

4.2 Non-functional Requirements:

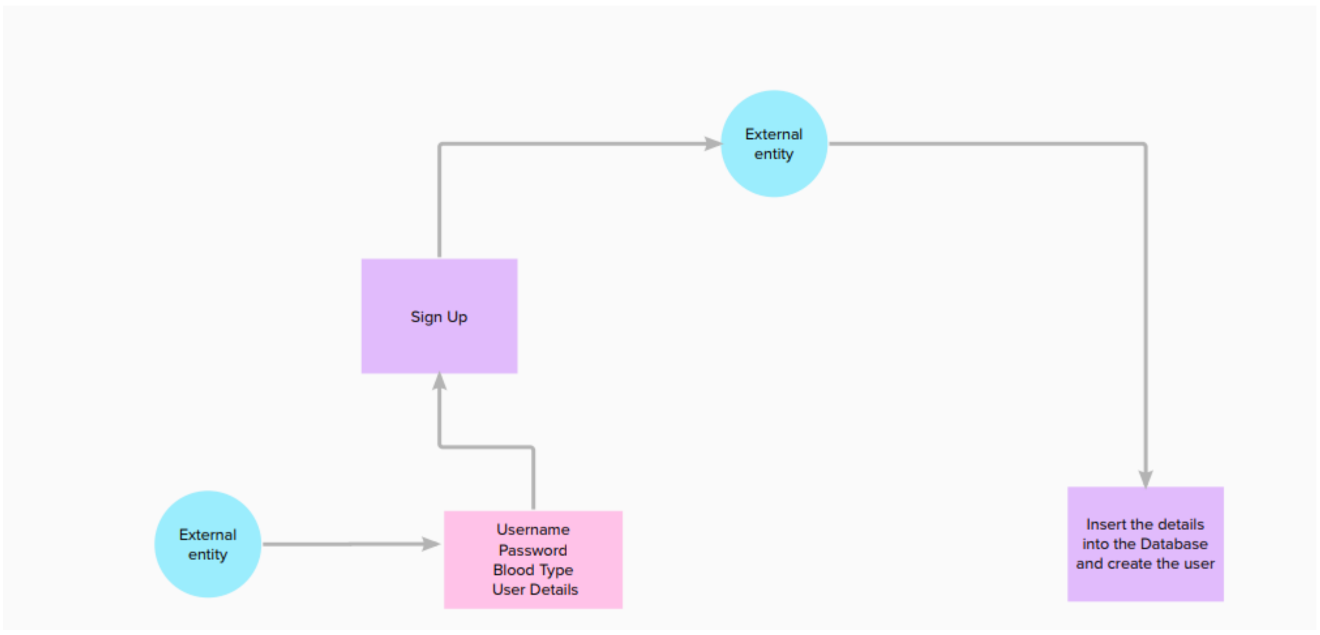
Following are the non-functional requirements of the proposed solution.

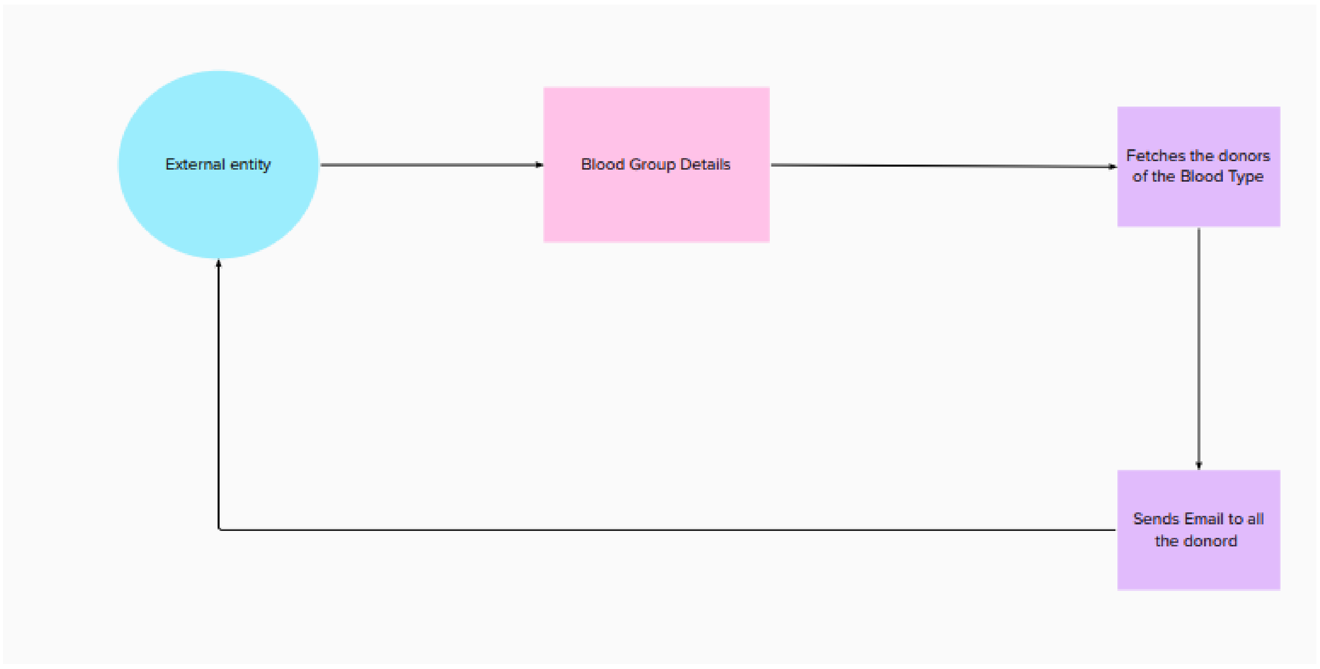
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	The plasma donor application must have a good looking user friendly interface.
NFR-2	**Security**	The plasma donor application must be secured with proper user name and passwords.
NFR-3	**Reliability**	The plasma donor application should work properly,even when faults occur.
NFR-4	**Performance**	The plasma donor application must perform well
		in different scenarios.
NFR-5	**Availability**	The plasma donor application must available 24 hours a day with no bandwidth issues.

NFR-6	**Scalability**	The plasma donor application should able to increase or decrease in performance and cost in response to changes in application and system processing demands.

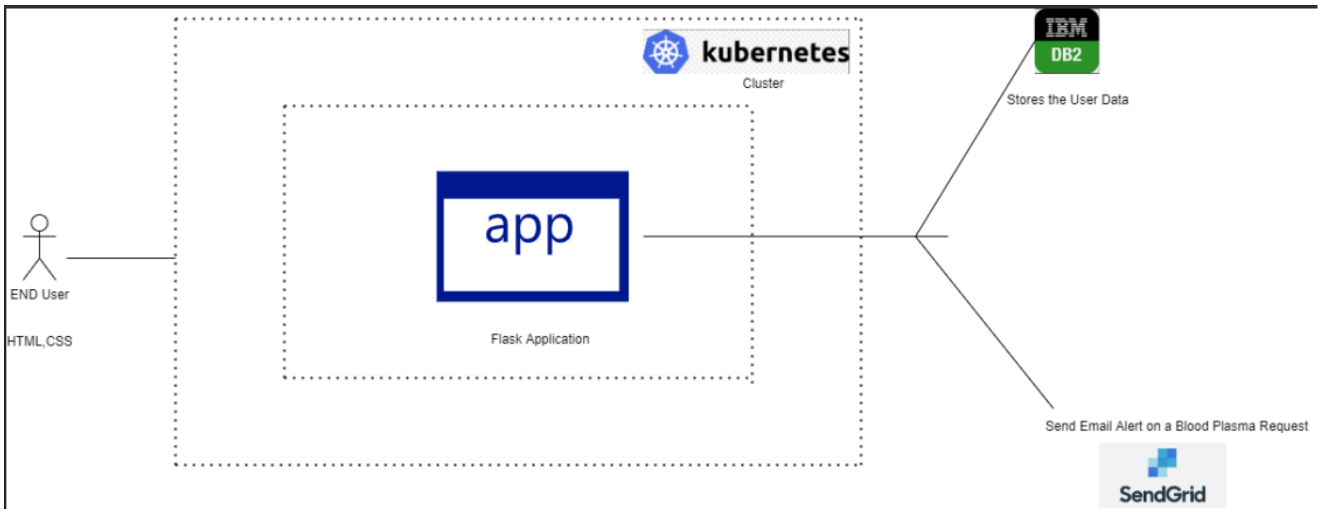
5.PROJECT DESIGN

5.1 Data Flow Diagram:

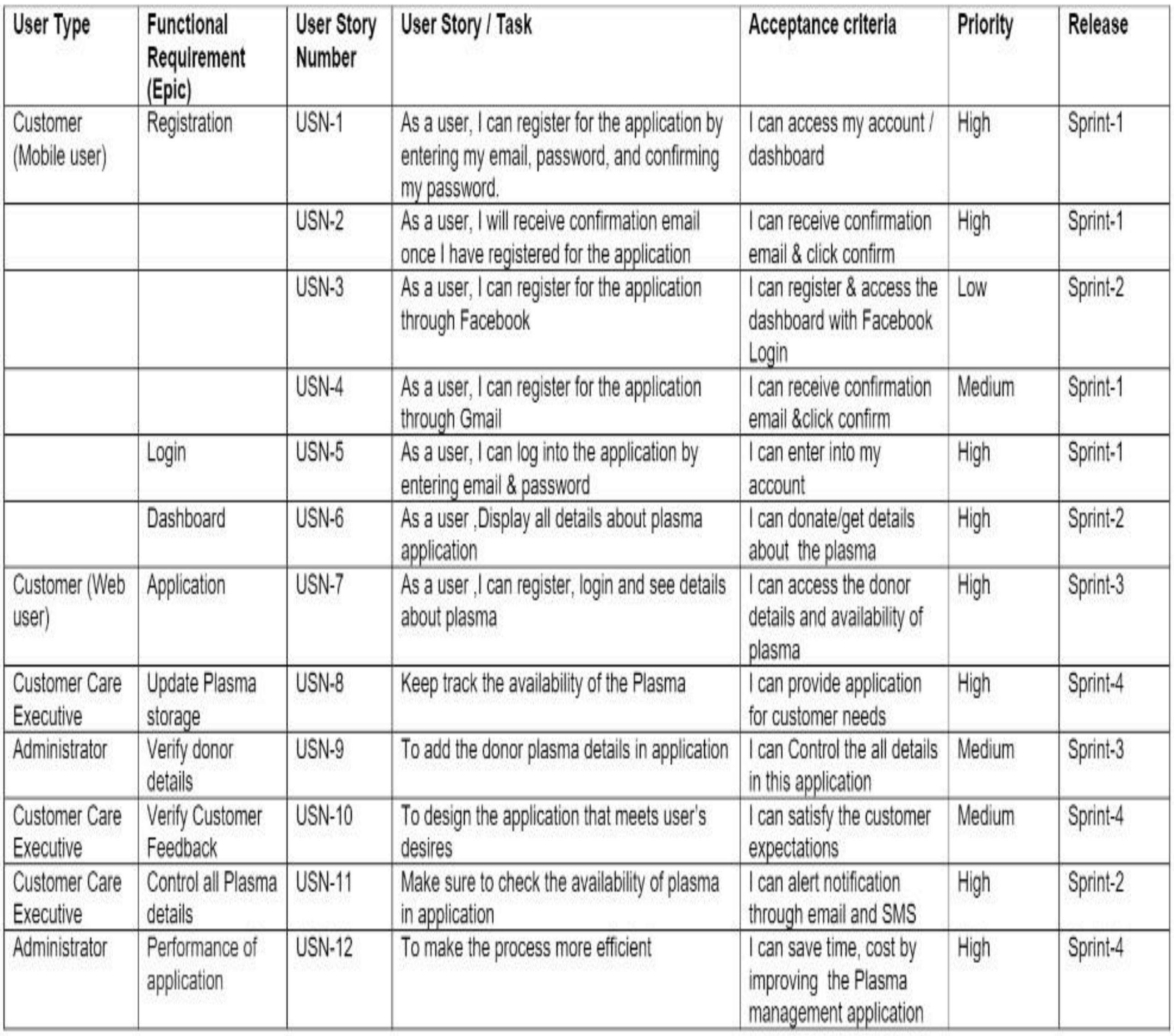



External entity
Blood Group Details
Fetches the donors of the Blood Type
Sends Email to all the donord

5.2 Solution and Technical Architecture :



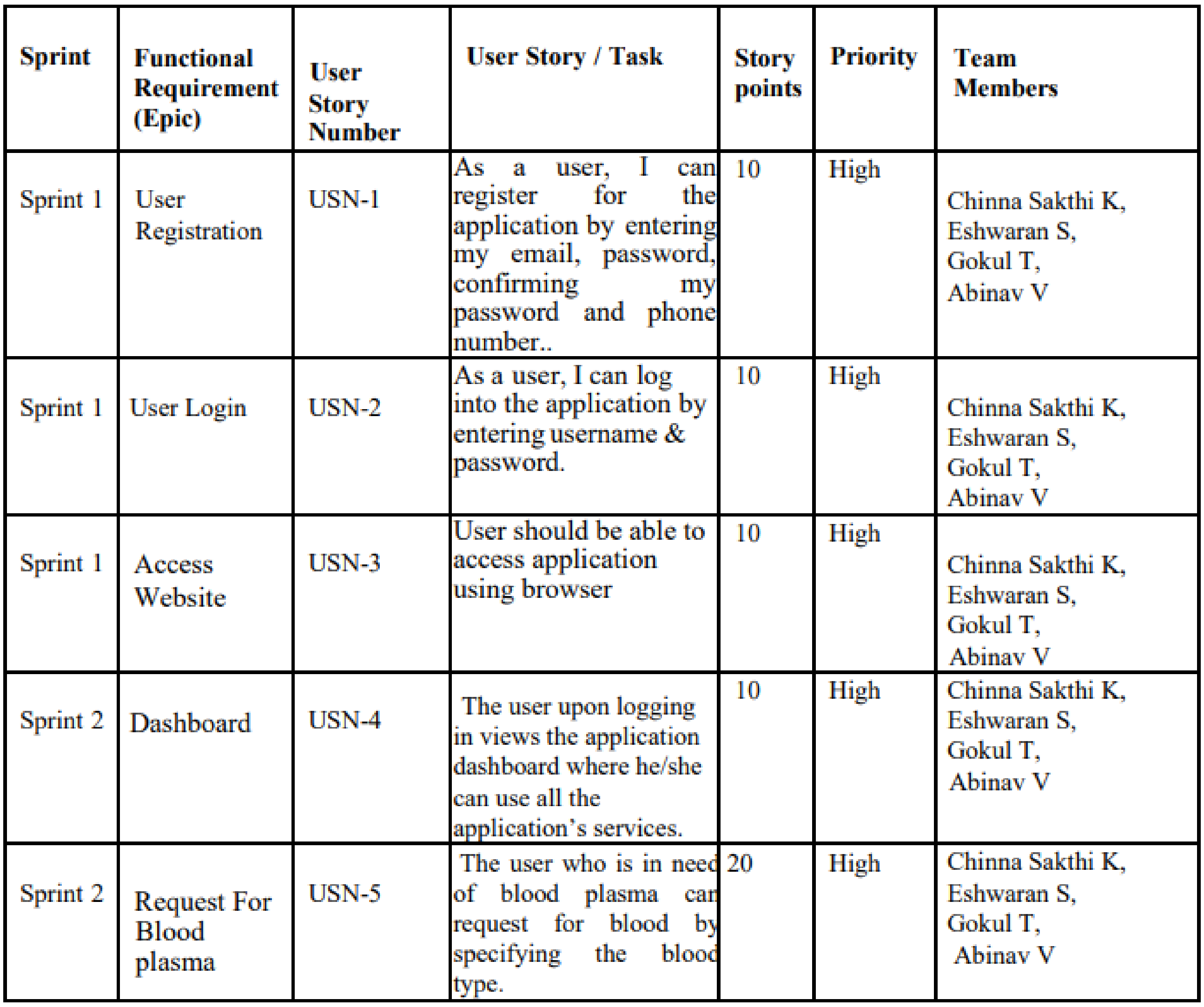
5.3 User Stories:



User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register 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 Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail	I can receive confirmation email &click confirm	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password	I can enter into my account	High	Sprint-1
	Dashboard	USN-6	As a user ,Display all details about plasma application	I can donate/get details about the plasma	High	Sprint-2
Customer (Web user)	Application	USN-7	As a user ,I can register, login and see details about plasma	I can access the donor details and availability of plasma	High	Sprint-3
Customer Care Executive	Update Plasma storage	USN-8	Keep track the availability of the Plasma	I can provide application for customer needs	High	Sprint-4
Administrator	Verify donor details	USN-9	To add the donor plasma details in application	I can Control the all details in this application	Medium	Sprint-3
Customer Care Executive	Verify Customer Feedback	USN-10	To design the application that meets user's desires	I can satisfy the customer expectations	Medium	Sprint-4
Customer Care Executive	Control all Plasma details	USN-11	Make sure to check the availability of plasma in application	I can alert notification through email and SMS	High	Sprint-2
Administrator	Performance of application	USN-12	To make the process more efficient	I can save time, cost by improving the Plasma management application	High	Sprint-4

6.PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning & Estimation



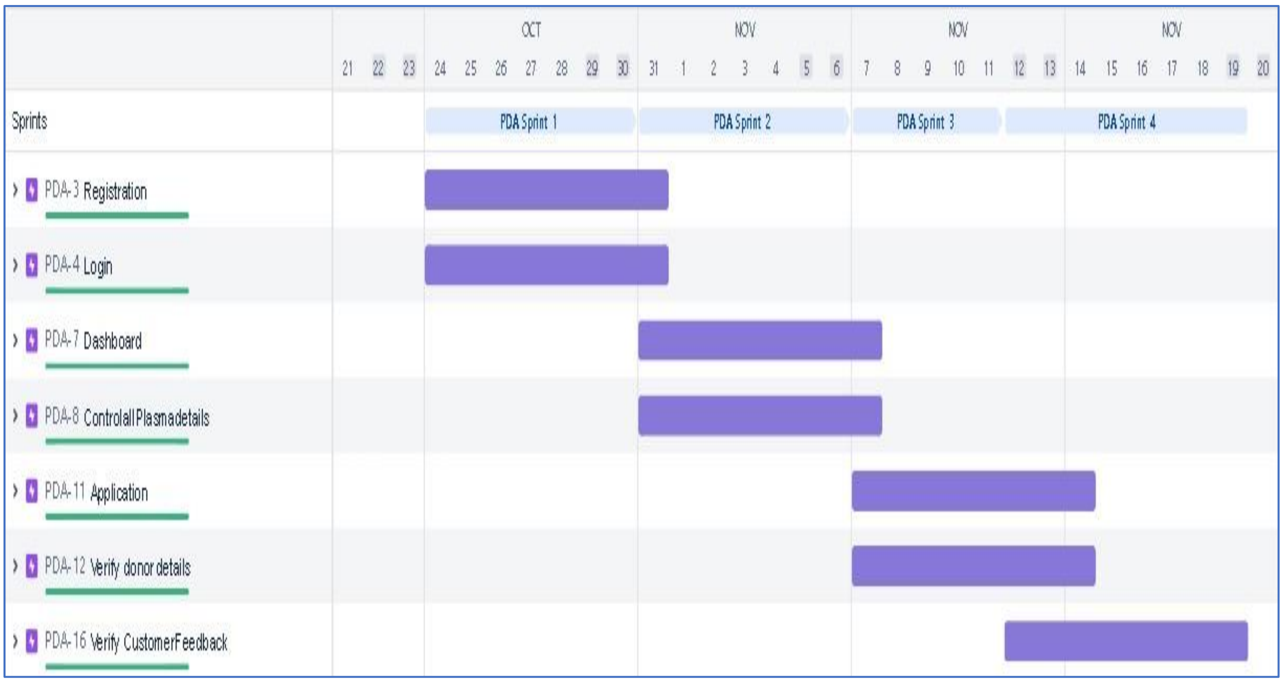
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story points	Priority	Team Members
Sprint 1	User Registration	USN-1	As a user, I can register for the application by entering my email, password, confirming my password and phone number..	10	High	Chinna Sakthi K, Eshwaran S, Gokul T, Abinav V
Sprint 1	User Login	USN-2	As a user, I can log into the application by entering username & password.	10	High	Chinna Sakthi K, Eshwaran S, Gokul T, Abinav V
Sprint 1	Access Website	USN-3	User should be able to access application using browser	10	High	Chinna Sakthi K, Eshwaran S, Gokul T, Abinav V
Sprint 2	Dashboard	USN-4	The user upon logging in views the application dashboard where he/she can use all the application's services.	10	High	Chinna Sakthi K, Eshwaran S, Gokul T, Abinav V
Sprint 2	Request For Blood plasma	USN-5	The user who is in need of blood plasma can request for blood by specifying the blood type.	20	High	Chinna Sakthi K, Eshwaran S, Gokul T, Abinav V

Sprint 2	Switch User Roles	USN-6	As a user, he/she can switch roles between Donor and Receiver.	20	High	Chinna Sakthi K, Eshwaran S, Gokul T, Abinav V
Sprint 3	View Plasma Request	USN-7	A donor receives an Email of about the receiver's details of the same blood type.	20	High	Chinna Sakthi K, Eshwaran S, Gokul T, Abinav V
Sprint 3	View Donor Details	USN-8	The receiver can view the list of Donors of the blood type requested.	10	High	Chinna Sakthi K, Eshwaran S, Gokul T, Abinav V
Sprint 4	Logout Process	USN-9	The User will be able to Logout of the application.	10	High	Chinna Sakthi K, Eshwaran S, Gokul T, Abinav V
Sprint 4	Bot service in the website	USN-10	The user can use Bot Service to request for Blood Plasma and also switch between roles.	10	High	Chinna Sakthi K, Eshwaran S, Gokul T, Abinav V

6.2 Sprint delivery schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint 1	30	8 days	22-10-2022	29-10-2022	30	29-10-2022
Sprint 2	50	8 days	29-11-2022	05-11-2022	50	05-11-2022
Sprint 3	30	8 days	05-11-2022	12-11-2022	30	12-11-2022
Sprint 4	20	8 days	12-11-2022	19-11-2022	20	19-11-2022

6.3 Reports from JIRA



7.CODING & SOLUTIONING

7.1 Feature 1:

Python

- Python is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics, used for generalpurpose programming. It's everywhere, and people use numerous Python-powered devices on a daily basis, whether they realize it or not.
- Python was created by Guido van Rossum, and first released on February 20, 1991.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages.
- Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL)

- It is easy to learn – the time needed to learn Python is shorter than for many other languages; this means that it’s possible to start the actual programming fast
- It is easy to use for writing new software – it’s often possible to write code faster when using Python.
- It is easy to obtain, install and deploy – Python is free, open and multiplatform; not all languages can boast that.
- Programming skills prepare you for careers in almost any industry and are required if you want to continue to more advanced and higherpaying software development and engineering roles.
- Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

7.2 Feature 2:

Flask

- **Flask** is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries.
- It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself.
- Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.
- Applications that use the Flask framework include Pinterest and LinkedIn.

7.3 Database Scheme

IBM Db2

- DB2 is a database product from IBM.
- It is a Relational Database Management System (RDBMS). DB2 is designed to store, analyze and retrieve the data efficiently.
- DB2 product is extended with the support of Object-Oriented features and non-relational structures with XML.
- Provide a massively parallel processing (MPP) architecture Exploits Hive, HBase and Apache Spark concurrently for best-in-class analytic capabilities.
- Provides low latency support for ad-hoc and complex queries, high performance, and federation capabilities Understands dialects from other vendors and various products from Oracle, IBM® Db2® and IBM Netezza® Enables advanced row and column security

Kubernates

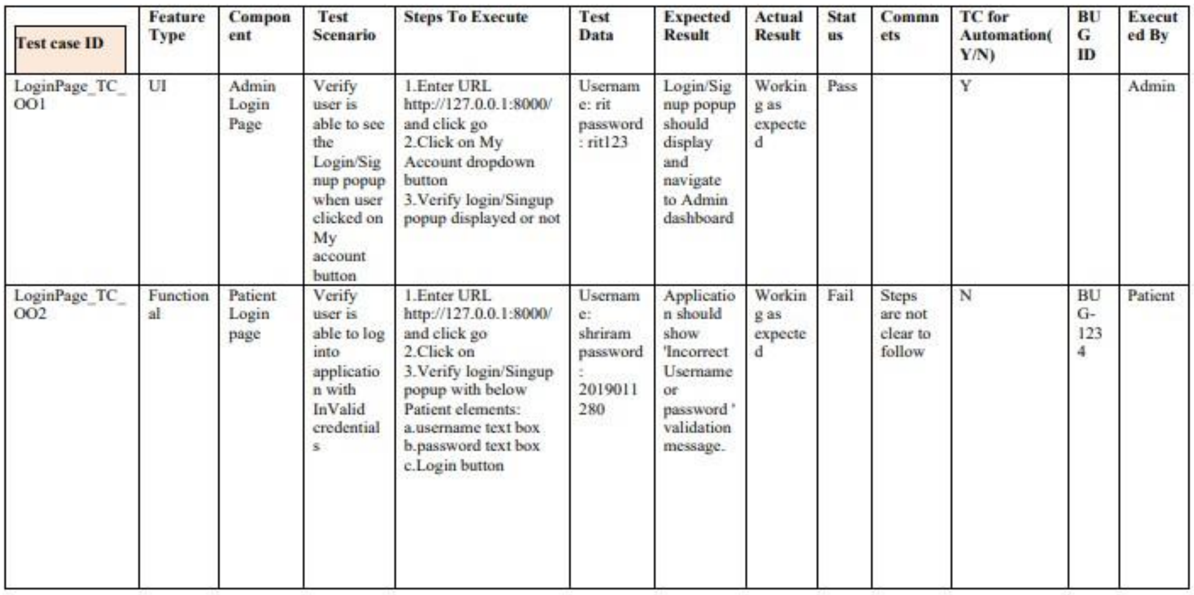
- **Kubernetes** is also known as **'k8s'.**
- **Kubernetes** is an extensible, portable, and open-source platform designed by **Google** in **2014**.
- It is mainly used to automate the deployment, scaling, and operations of the container-based applications across the cluster of nodes.
- Kubernetes helps to manage containerised applications in various types of physical, virtual, and cloud environments.
- Google Kubernetes is a highly flexible container tool to consistently deliver complex applications running on clusters of hundreds to thousands of individual servers
- Kubernetes is the Linux kernel which is used for distributed systems.

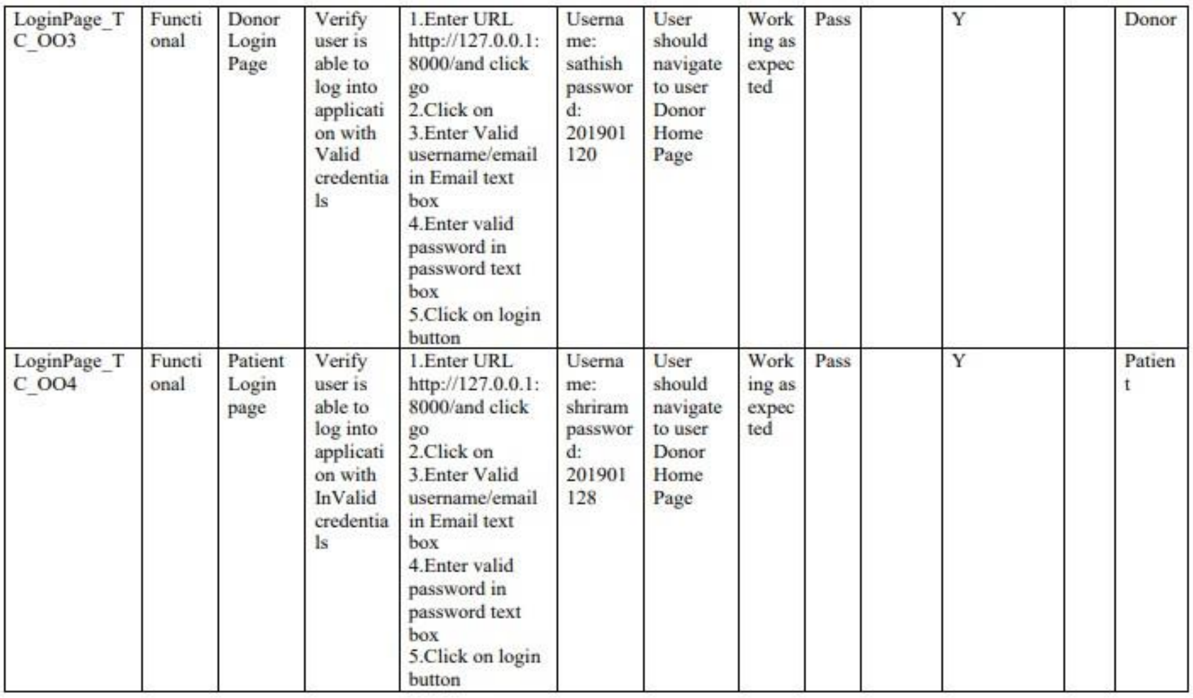
- It helps you to be abstract the underlying hardware of the nodes(servers) and offers a consistent interface for applications that consume the shared pool of resources.

8.TESTING

8.1 Test case

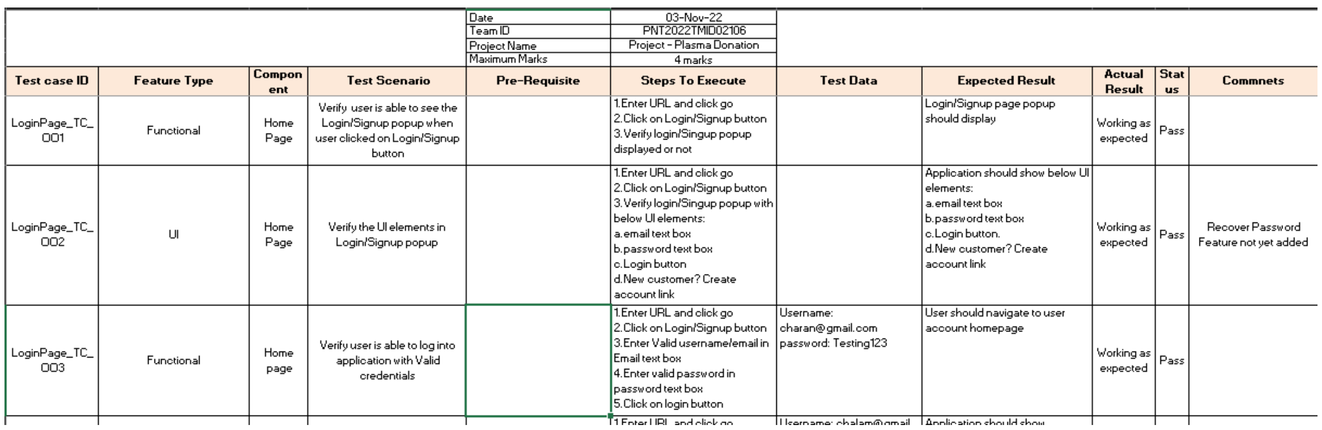
- It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectation and does not fail in an unacceptable manner.
- There are various types of test. Each test type addresses a specific testing requirement





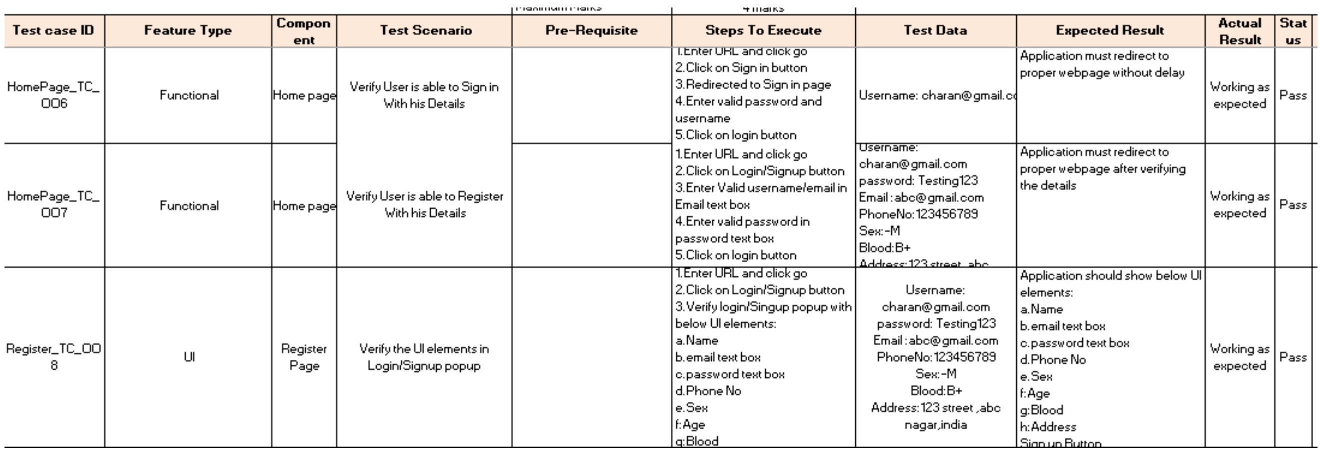
Test case ID	Feature Type	Component	Test Scenario	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	UI	Admin Login Page	Verify user is able to see the Login/Signup popup when user clicked on My account button	1.Enter URL http://127.0.0.1:8000/ and click go 2.Click on My Account dropdown button 3.Verify login/Singup popup displayed or not	Username: rit password: rit123	Login/Signup popup should display and navigate to Admin dashboard	Working as expected	Pass		Y		Admin
LoginPage_TC_OO2	Functional	Patient Login page	Verify user is able to log into application with InValid credentials	1.Enter URL http://127.0.0.1:8000/ and click go 2.Click on 3.Verify login/Singup popup with below Patient elements: a.username text box b.password text box c.Login button	Username: shriram password: 2019011280	Application should show 'Incorrect Username or password ' validation message.	Working as expected	Fail	Steps are not clear to follow	N	BUG-1234	Patient
LoginPage_TC_OO3	Functional	Donor Login Page	Verify user is able to log into application with Valid credentials	1.Enter URL http://127.0.0.1:8000/and click go 2.Click on 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: sathish password: 201901120	User should navigate to user Donor Home Page	Working as expected	Pass		Y		Donor
LoginPage_TC_OO4	Functional	Patient Login page	Verify user is able to log into application with InValid credentials	1.Enter URL http://127.0.0.1:8000/and click go 2.Click on 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: shriram password: 201901128	User should navigate to user Donor Home Page	Working as expected	Pass		Y		Patient

8.2 User Acceptance Testing



				Date	03-Nov-22
				Team ID	PNT2022TMID02106
				Project Name	Project - Plasma Donation
				Maximum Marks	4 marks

Test case ID	Feature Type	Compon ent	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Stat us	Commnets
LoginPage_TC_001	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on Login/Signup button		1.Enter URL and click go 2.Click on Login/Signup button 3.Verify login/Singup popup displayed or not		Login/Signup page popup should display	Working as expected	Pass	
LoginPage_TC_002	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on Login/Signup button 3.Verify login/Singup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link		Application should show below UI elements: a.email text box b.password text box c.Login button. d.New customer? Create account link	Working as expected	Pass	Recover Password Feature not yet added
LoginPage_TC_003	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL and click go 2.Click on Login/Signup button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: charan@gmail.com password: Testing123	User should navigate to user account homepage	Working as expected	Pass	
					1.Enter URL and click go	Username: chalam@gmail	Application should show			



Test case ID	Feature Type	Compon ent	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Stat us
HomePage_TC_006	Functional	Home page	Verify User is able to Sign in With his Details		1.Enter URL and click go 2.Click on Sign in button 3.Redirected to Sign in page 4.Enter valid password and username 5.Click on login button	Username: charan@gmail.co	Application must redirect to proper webpage without delay	Working as expected	Pass
HomePage_TC_007	Functional	Home page	Verify User is able to Register With his Details		1.Enter URL and click go 2.Click on Login/Signup button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: charan@gmail.com password: Testing123 Email : abc@gmail.com PhoneNo: 123456789 Sex:-M Blood:B+ Address:123 street ,abc	Application must redirect to proper webpage after verifying the details	Working as expected	Pass
Register_TC_008	UI	Register Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on Login/Signup button 3.Verify login/Singup popup with below UI elements: a.Name b.email text box c.password text box d.Phone No e.Sex f:Age g:Blood	Username: charan@gmail.com password: Testing123 Email : abc@gmail.com PhoneNo: 123456789 Sex:-M Blood:B+ Address: 123 street ,abc nagar,india	Application should show below UI elements: a.Name b.email text box c.password text box d.Phone No e.Sex f:Age g:Blood h:Address Sign up Button	Working as expected	Pass

1.Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Plasma Donation Application project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

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

Resolution	**Severity 1**	**Severity 2**	**Severity 3**	**Severity 4**	**Sub total**
By Design	8	4	2	3	17
Duplicate	1	0	2	1	4
External	2	3	0	1	6

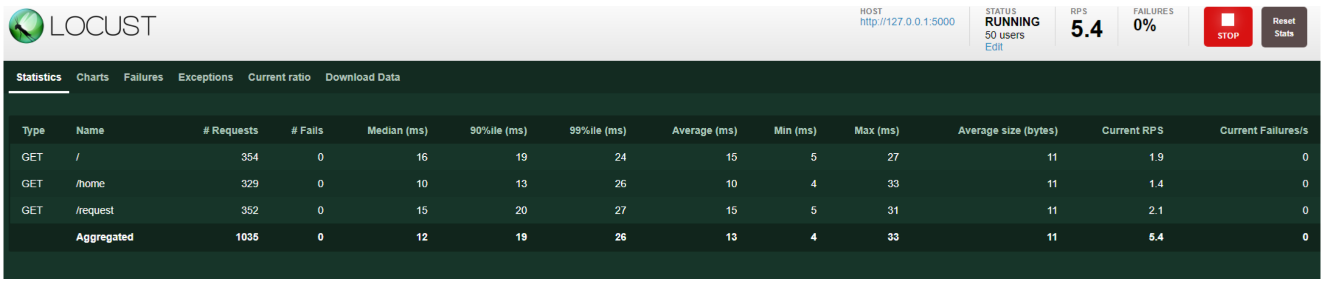
Fixed	10	2	5	18	35
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	3	2	1	6
Totals	21	12	13	25	71

3.Test Case Analysis

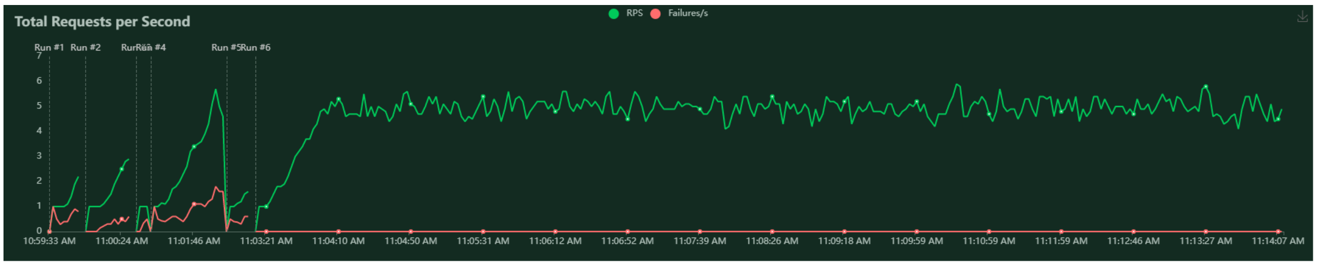
This report shows the number of test cases that have passed, failed, and untested

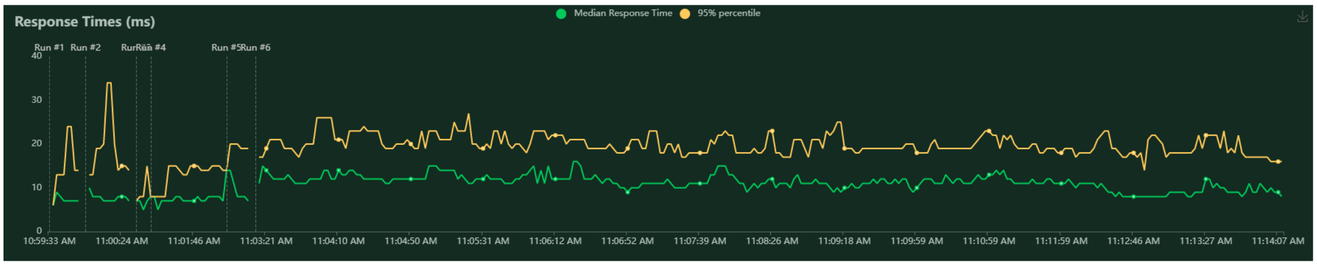
Section	Total Cases	Not Tested	Fail	Pass
Print Engine	8	0	0	8
Client Application	50	0	0	50
Security	2	0	0	2
Outsource Shipping	3	0	0	3

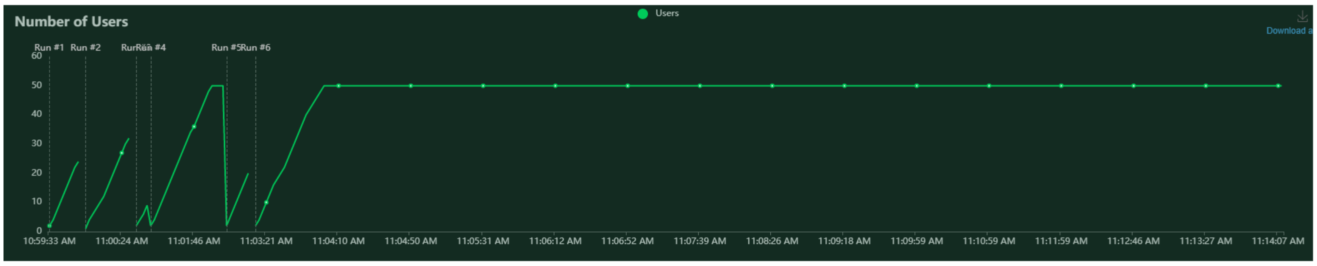
Exception Reporting	10	0	0	10
Final Report Output	6	0	0	6
Version Control	3	0	0	3



Type	Name	# Requests	# Fails	Median (ms)	90%ile (ms)	99%ile (ms)	Average (ms)	Min (ms)	Max (ms)	Average size (bytes)	Current RPS	Current Failures/s
GET	/	354	0	16	19	24	15	5	27	11	1.9	0
GET	/home	329	0	10	13	26	10	4	33	11	1.4	0
GET	/request	352	0	15	20	27	15	5	31	11	2.1	0
	Aggregated	**1035**	**0**	**12**	**19**	**26**	**13**	**4**	**33**	**11**	**5.4**	**0**







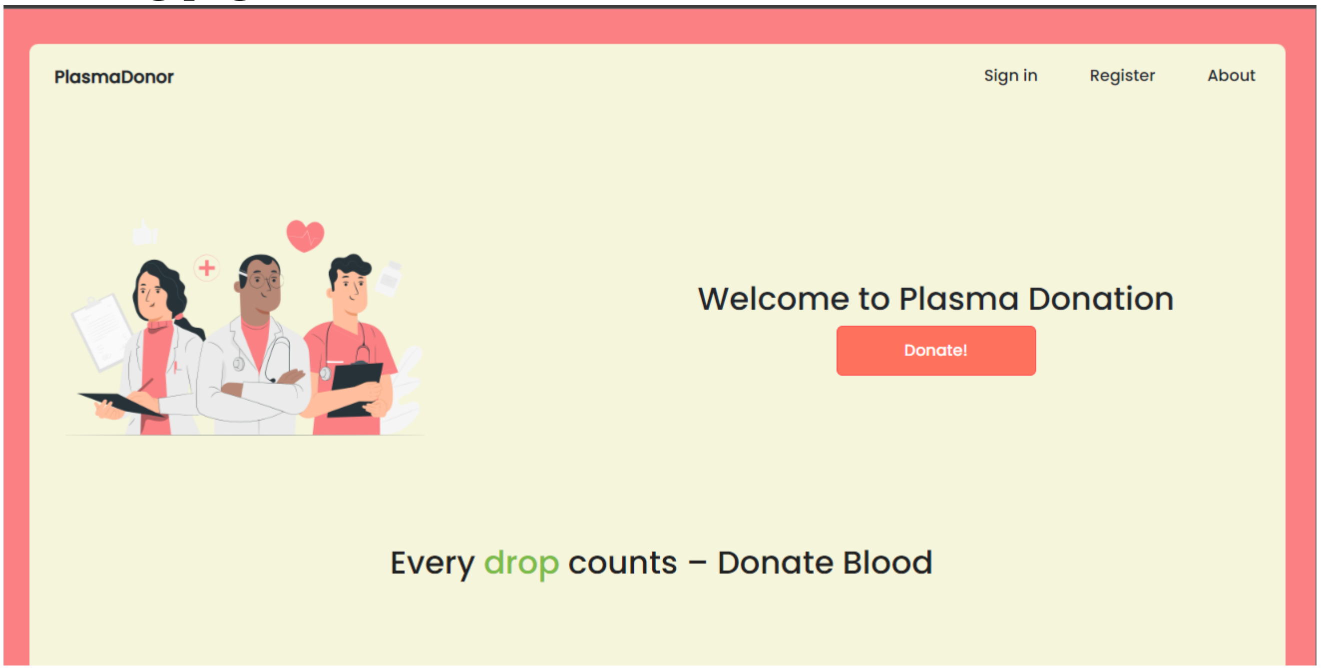
9.RESULTS

9.1 Performance Metrics

- Project metrics are used to track the progress and performance of a project.
- Monitoring parts of a project like productivity, scheduling, and scope make it easier for team leaders to see what's on track.
- As a project evolves, managers need access to changing
- deadlines or budgets to meet their client's expectations

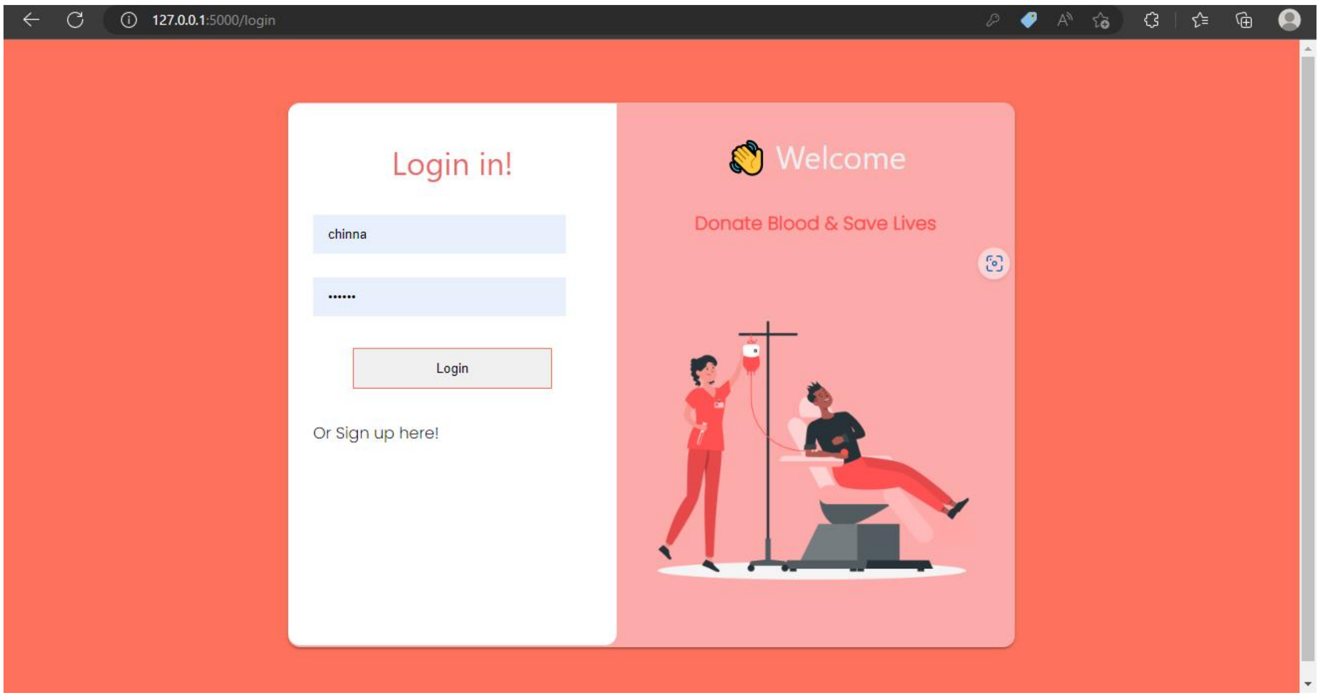
OUTPUT SCREENS

Landing page



Welcome to Plasma Donation

Every drop counts – Donate Blood


127.0.0.1:5000/login

Login in!

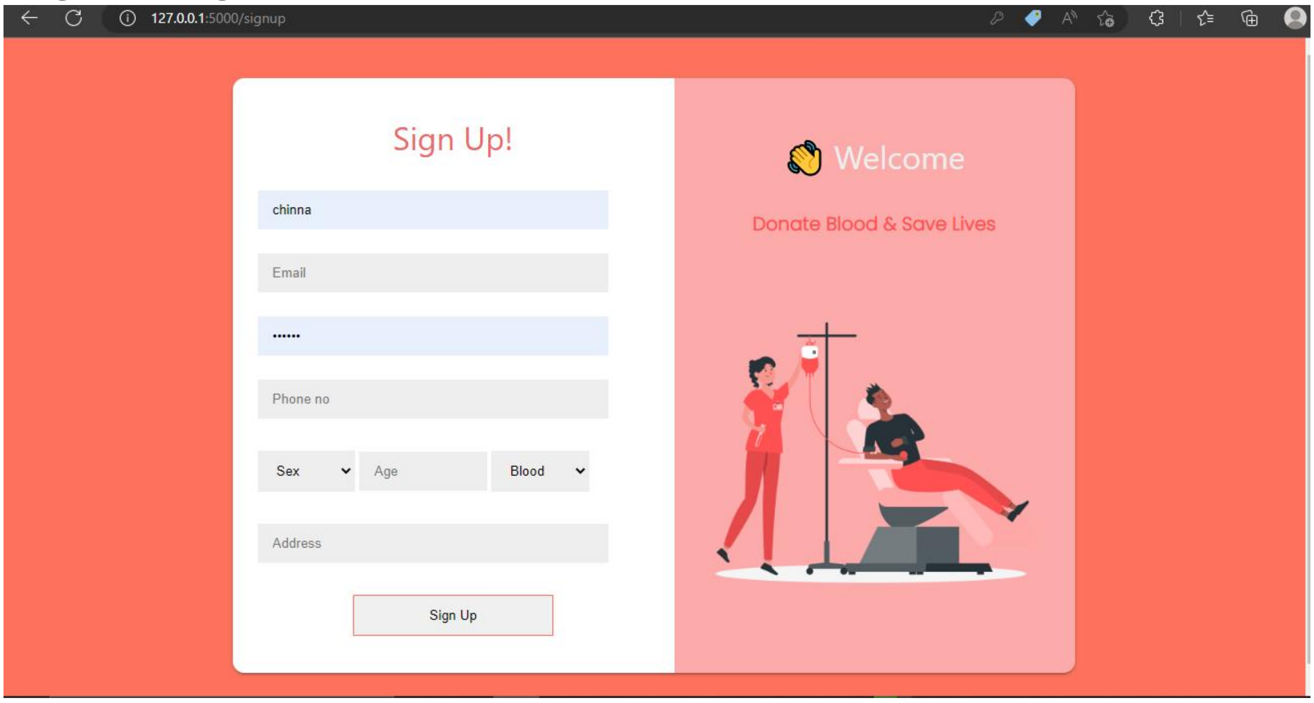
chinna

••••••

Login

Or Sign up here!

Register Page


127.0.0.1:5000/signup

Sign Up!

chinna

Email

••••••

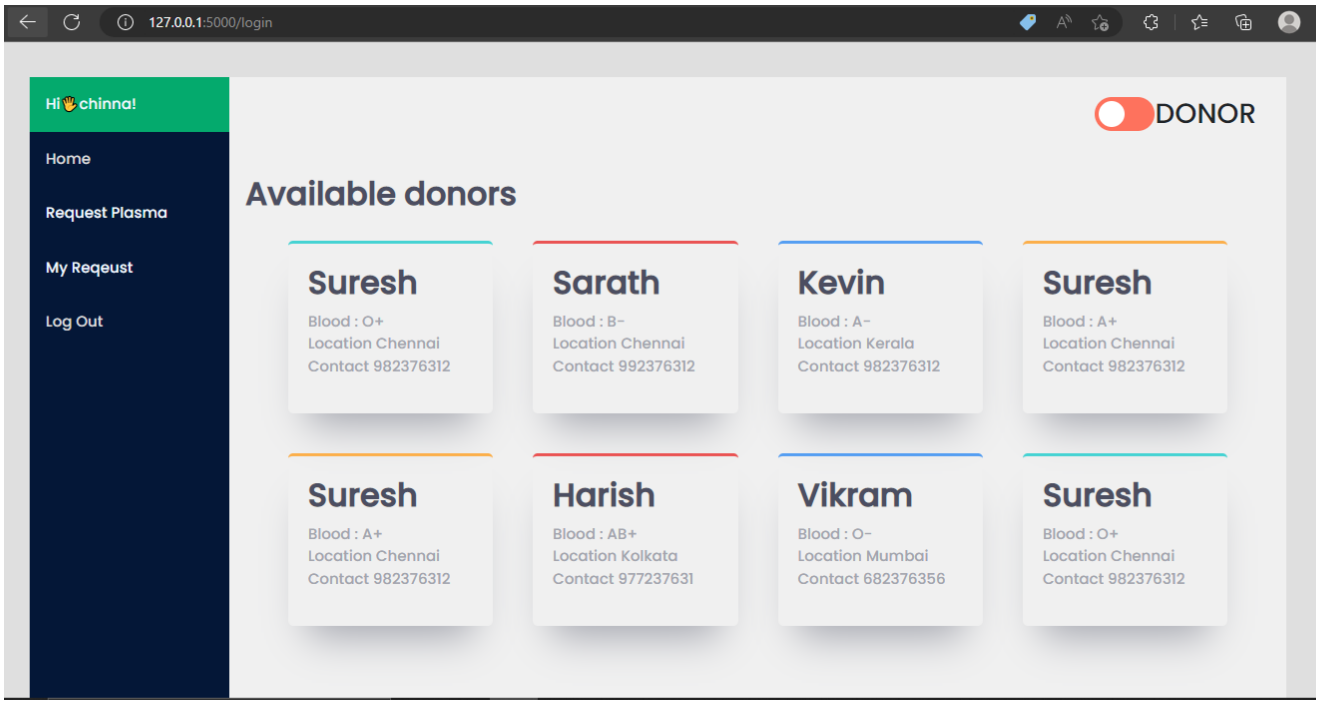
Phone no

Sex | Age | Blood

Address

Sign Up

Dashboard


127.0.0.1:5000/login

Hi 🖐chinna!

Home

Request Plasma

My Reqeust

Log Out

DONOR

Available donors

Suresh
Blood : O+
Location Chennai
Contact 982376312

Sarath
Blood : B-
Location Chennai
Contact 992376312

Kevin
Blood : A-
Location Kerala
Contact 982376312

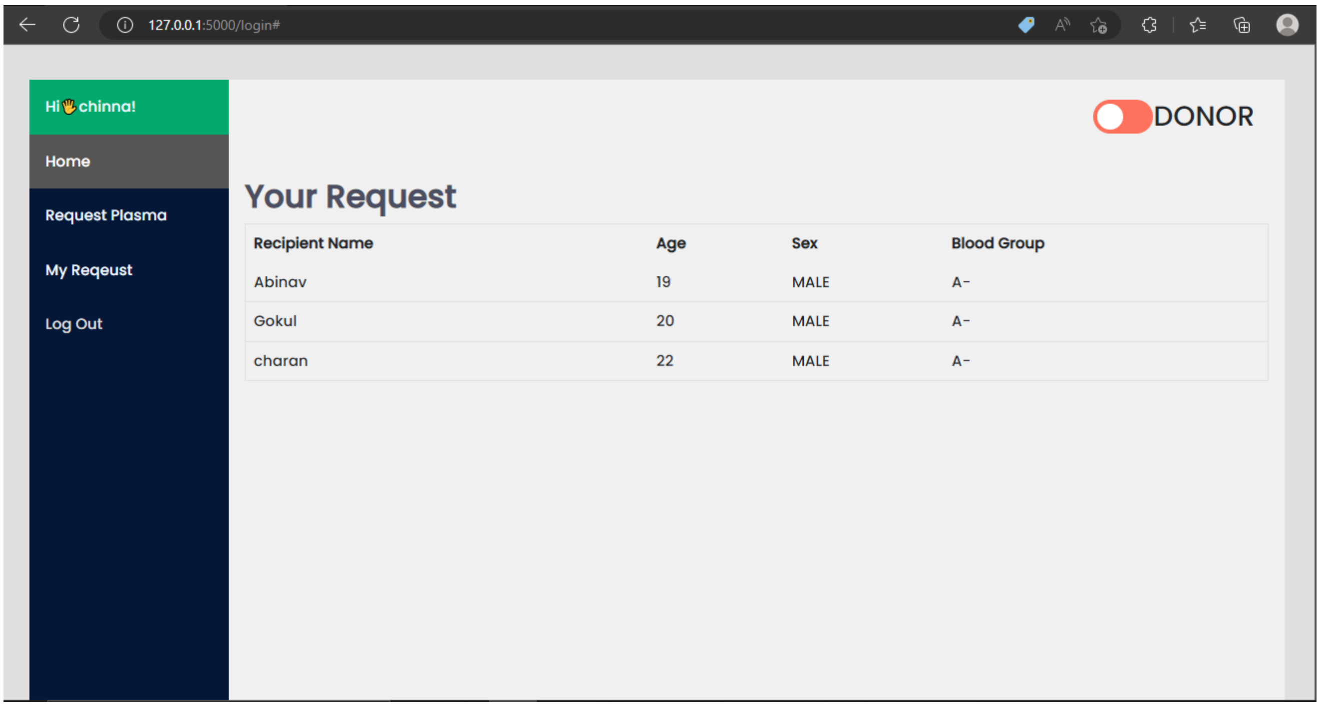
Suresh
Blood : A+
Location Chennai
Contact 982376312

Suresh
Blood : A+
Location Chennai
Contact 982376312

Harish
Blood : AB+
Location Kolkata
Contact 977237631

Vikram
Blood : O-
Location Mumbai
Contact 682376356

Suresh
Blood : O+
Location Chennai
Contact 982376312


127.0.0.1:5000/login#

Hi 🖐chinna!

Home

Request Plasma

My Reqeust

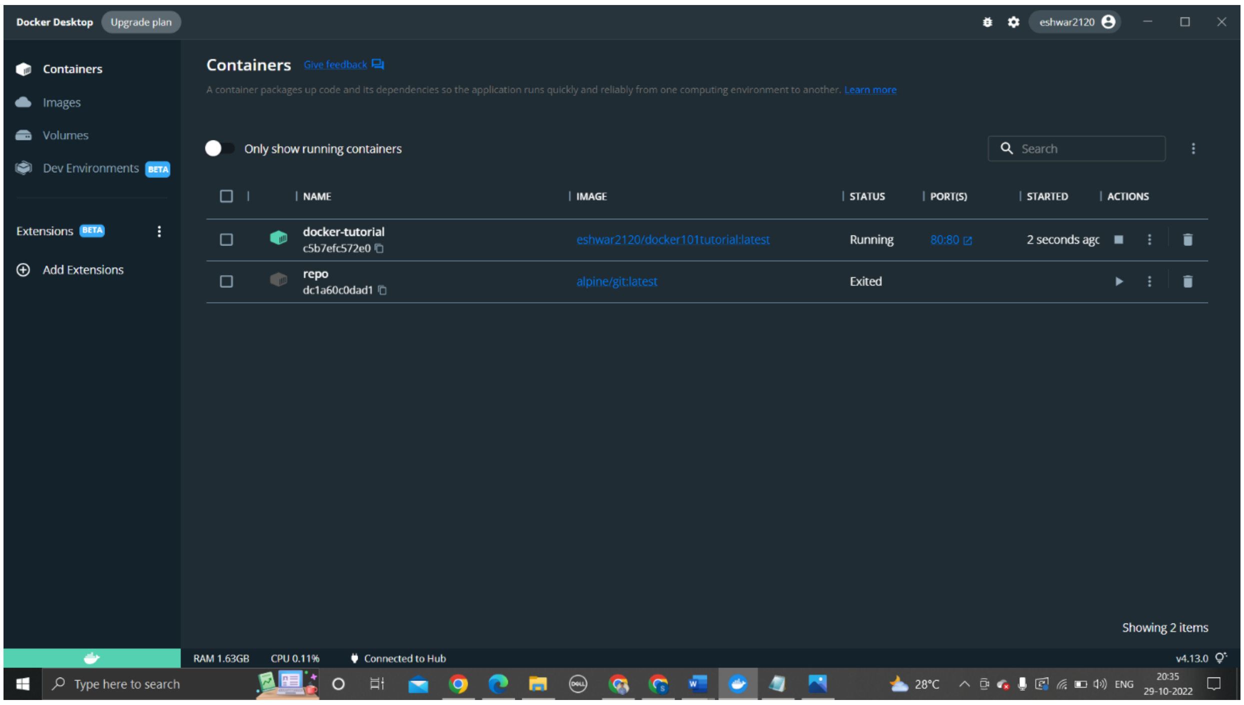
Log Out

DONOR

Your Request

Recipient Name	Age	Sex	Blood Group
Abinav	19	MALE	A-
Gokul	20	MALE	A-
charan	22	MALE	A-

Dockerize the app



Containers

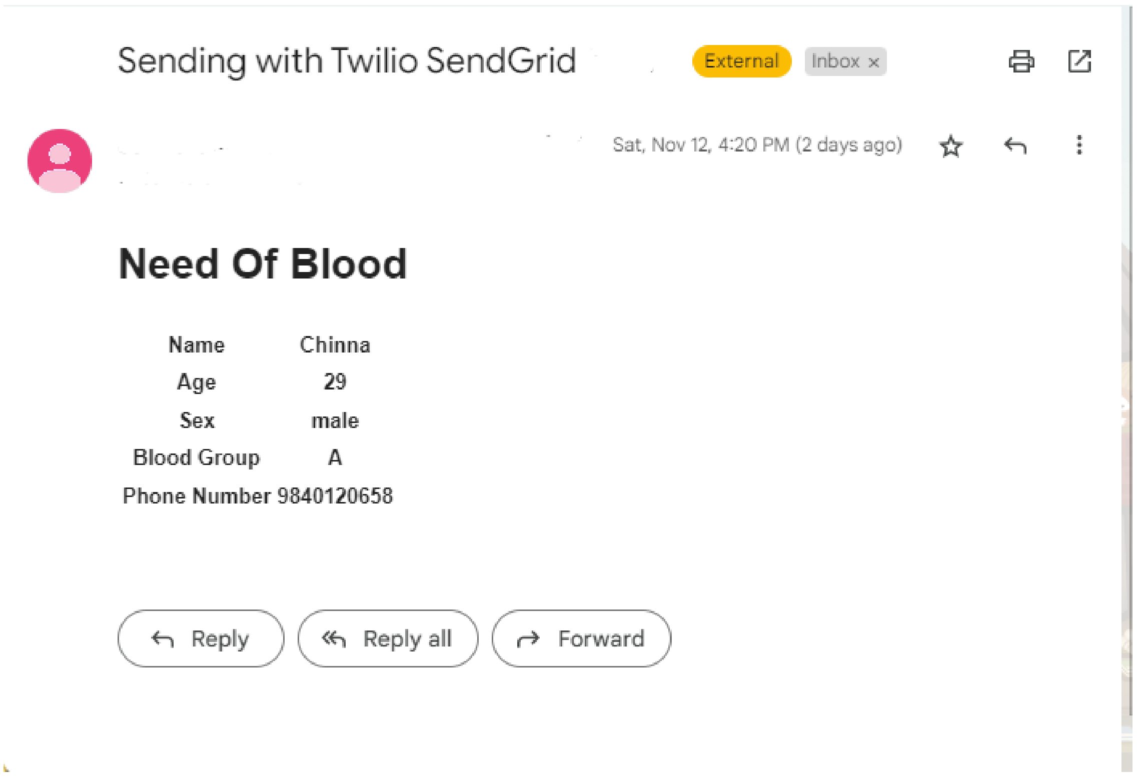
A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. Learn more

NAME	IMAGE	STATUS	PORT(S)	STARTED
docker-tutorial c5b7efc572e0	eshwar2120/docker101tutorial:latest	Running	80:80	2 seconds agc
repo dc1a60c0dad1	alpine/git:latest	Exited		

Showing 2 items

RAM 1.63GB CPU 0.11% Connected to Hub v4.13.0

SENDING MAIL



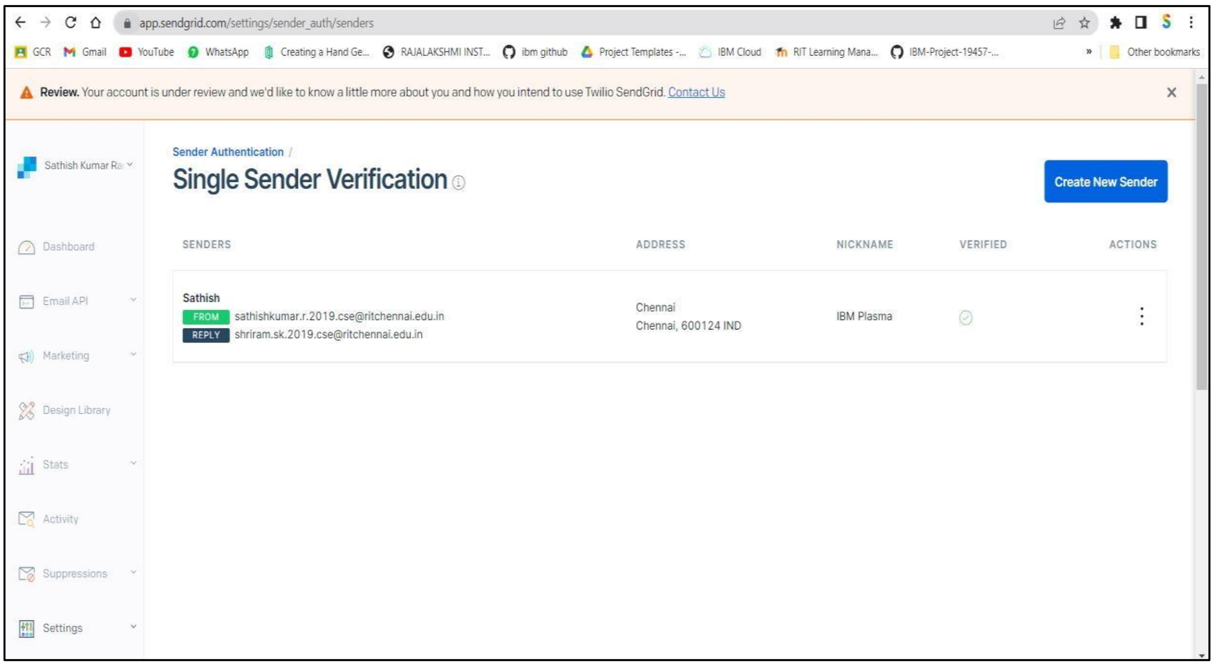
Sending with Twilio SendGrid

Sat, Nov 12, 4:20 PM (2 days ago)

Need Of Blood

Name	Chinna
Age	29
Sex	male
Blood Group	A
Phone Number	9840120658

SEND GRID


app.sendgrid.com/settings/sender_auth/senders
GCR · Gmail · YouTube · WhatsApp · Creating a Hand Ge... · RAJALAKSHMI INST... · ibm github · Project Templates -... · IBM Cloud · RIT Learning Mana... · IBM-Project-19457-... · Other bookmarks
Sathish Kumar Ra
Dashboard
Email API
Marketing
Design Library
Stats
Activity
Suppressions
Settings
Create New Sender

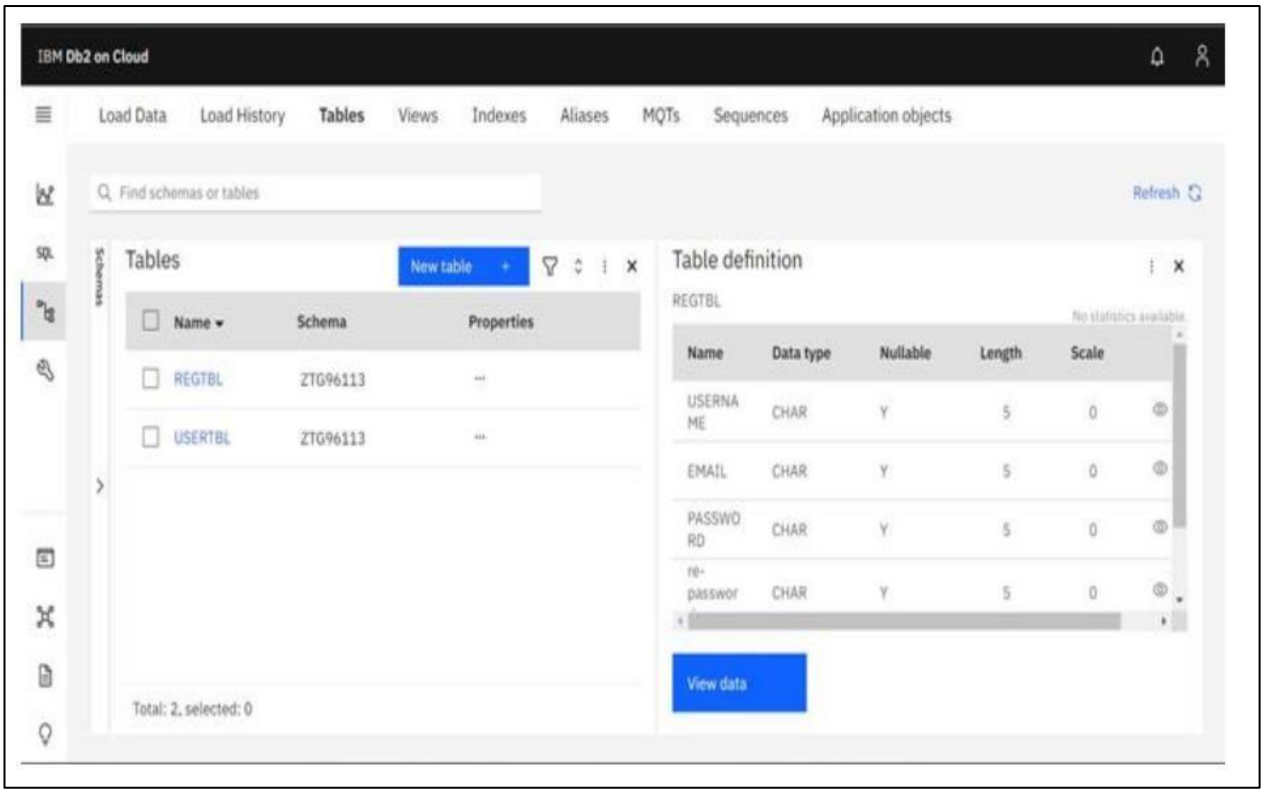
Review. Your account is under review and we'd like to know a little more about you and how you intend to use Twilio SendGrid. Contact Us

Sender Authentication /

Single Sender Verification

SENDERS	ADDRESS	NICKNAME	VERIFIED	ACTIONS
Sathish FROM sathishkumar.r.2019.cse@ritchennai.edu.in REPLY shriram.sk.2019.cse@ritchennai.edu.in	Chennai Chennai, 600124 IND	IBM Plasma	✓	⋮

IBM DB2



Tables

Name	Schema	Properties
REGTBL	ZTG96113	...
USERTBL	ZTG96113	...

Total: 2, selected: 0

Table definition

REGTBL

No statistics available

Name	Data type	Nullable	Length	Scale
USERNAME	CHAR	Y	5	0
EMAIL	CHAR	Y	5	0
PASSWORD	CHAR	Y	5	0
re-passwor	CHAR	Y	5	0

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:

- **Speed**: This website is fast and offers great accuracy as compared to manual registered keeping.
- **Maintenance:** Less maintenance is required
- **User Friendly:** It is very easy to use and understand. It is easily workable and accessible for everyone.
- **Fast Results:** It would help you to provide plasma donors easily depending upon the availability of it.

DISADVANTAGES:

- **Internet:** It would require an internet connection for the working of the website.
- **Auto- Verification:** It cannot automatically verify the genuine users.

11. CONCLUSION

- The efficient way of finding plasma door for the infected people is implemented using the plasma donor website that is hosted on IBM Cloud platform.

- To ensure the smooth functioning of the web site operation. I have hosted the website in IBM Db2 & Kubernates Cluster to make sure the operations are running successfully Cloud lambda function is used and to deploy the application IBM Db2 service is used.

12. FUTURE SCOPE

- Upgrading the UI that is more user-friendly which will help many users to access the website and also ensures that many plasma donors can be added into the community.

- Using elastic load balancer, it helps to handle multiple requests at the same time which will maintain the uptime of the website with negligible downtime

13.APPENDIXES

13.1 SAMPLE SOURCE CODE: DONOR

MAIN.py

```
from flask import Flask, redirect, url_for, render_template, request, make_response, 
jsonify, request

import ibm_db

from flask import request

import json


conn = ibm_db.connect(

    "DATABASE=bludb;HOSTNAME=764264db-9824-4b7c-82df-
40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32536;SEC
URITY=SSL;SSLServerCertificate=abc.crt;UID=gnq12618;PWD=0glS4tFaR2ciK8fB
",

    '', '')

print(conn)

print("connection successful...")

app = Flask(__name__)

import os

from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail



@app.route('/')

def home():

    return render_template("landing.html")
```

```
@app.route('/home')
def dash():
    return render_template("dashboard.html")


@app.route('/login', methods=['POST', 'GET'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        sql = "select * from user 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)
        dic = ibm_db.fetch_assoc(stmt)
        print(dic)
        role = str()
        requests = []
        if dic:
            role = dic['ROLE']
            # sql = "select * from user where blood_group=?"
            # stmt = ibm_db.prepare(conn, sql)
            # ibm_db.bind_param(stmt, 1, username)
            # ibm_db.execute(stmt)
            # dic = ibm_db.fetch_assoc(stmt)
            # while dic != False:
            #     single_request = {
            #         'name': dic['NAME'],
            #         'age': dic['AGE'],
            #         'sex': dic['SEX'],
            #         'blood_type': dic['BLOOD_TYPE']
            #     }
            #     print(single_request)
            #     requests.append(single_request)
            #     dic = ibm_db.fetch_assoc(stmt)
            return render_template('dashboard.html', username=username, role=role)


        else:
            return redirect(url_for('login'))
        return redirect(url_for('home'))
    elif request.method == 'GET':
        return render_template('login.html')


@app.route('/signup', methods=['POST', 'GET'])
def signup():
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        roll_no = request.form['roll_no']
        sex = request.form['sex']
        age = request.form['age']
        address = request.form['address']
        blood_group = request.form['blood_group']
        sql = "insert into user values(?,?,?,?,?,?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, 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, roll_no)
        ibm_db.bind_param(prep_stmt, 5, sex)
        ibm_db.bind_param(prep_stmt, 6, age)
        ibm_db.bind_param(prep_stmt, 7, "USER")
        ibm_db.bind_param(prep_stmt, 8, address)
        ibm_db.bind_param(prep_stmt, 9, blood_group)
        ibm_db.execute(prep_stmt)
        # db post operation
        return redirect(url_for('login'))
    elif request.method == 'GET':
        return render_template('signup.html')


@app.route('/toggle', methods=['POST'])
def toggle_user():
    data = request.get_json(force=True)

    username = data['username']
    role = data['role']
    print(username)
    print(role)
    sql = "update user set role=? where username=?"
    prep_stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(prep_stmt, 1, role)
    ibm_db.bind_param(prep_stmt, 2, username)
    ibm_db.execute(prep_stmt)
    return jsonify(
        status="success",
        role=role
    )


@app.route('/requestPalsma', methods=['POST'])
def requestBloodPlasma():
    # fetch mail address of the donors
    data = request.get_json(force=True)
    username = data['username']
    name = data['name']
    age = data['age']
    sex = data['sex']
    blood_type = data['bloodtype']
    phone_number = data['phone_num']
    sql = "select email from user where blood_group=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, blood_type)
    ibm_db.execute(stmt)
    dic = ibm_db.fetch_assoc(stmt)
    email_list = []
    while dic != False:
        email_list.append(dic['EMAIL'])
        print(dic['EMAIL'])
        dic = ibm_db.fetch_assoc(stmt)
    # send mail

    message = Mail(
        from_email='eshwaran.s.2019.cse@rajalakshmi.edu.in',
        to_emails=email_list,
        subject='Sending with Twilio SendGrid is Fun',
        html_content='<h1>Need Of Blood</h1><table><tr><th>Name</th><th>' +
name + '</th></tr><tr><th>Age</th><th>' + age + '</th></tr><tr><th>Sex</th><th>'
+ sex + '</th></tr><tr><th>Blood Group</th><th>' + blood_type +
'</th></tr><tr><th>Phone Number</th><th>' + phone_number + '</th></tr></table>'
    )
    try:
        sg = SendGridAPIClient("SG.3iBLSgAYTEuVbfSHu9dCPA.-
nrnikWJvaRlNLMONA04_CuKAyPeV69c46vPAh3vUX0")
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
    except Exception as e:
        print(e.message)
    # insert data into requests table
    sql = "insert into bloodrequests(username,name,age,sex,blood_type) values
(?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(prep_stmt, 1, username)
    ibm_db.bind_param(prep_stmt, 2, name)
    ibm_db.bind_param(prep_stmt, 3, age)
    ibm_db.bind_param(prep_stmt, 4, sex)
    ibm_db.bind_param(prep_stmt, 5, blood_type)
    ibm_db.execute(prep_stmt)

    return jsonify(
        name=name,
        age=age,
        sex=sex,
        bloodtype=blood_type,
        status="yes"
    )


@app.route('/getrequests', methods=['POST'])
def getBloodRequests():
    data = request.get_json(force=True)
    username = data['username']
    sql = "select * from bloodrequests where username=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)
    dic = ibm_db.fetch_assoc(stmt)
    requests = []
    print(type(dic))
    while dic != False:
        single_request = {
            'name': dic['NAME'],
            'age': dic['AGE'],
            'sex': dic['SEX'],
            'blood_type': dic['BLOOD_TYPE']
        }
        print(single_request)
        requests.append(single_request)
        dic = ibm_db.fetch_assoc(stmt)
    return jsonify(
        username=username,
        requests=requests
    )


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

13.2 GITHUB

https://github.com/IBM-EPBL/IBM-Project-14293-1659548348

PROJECT DEMO LINK

https://vimeo.com/manage/videos/772118796