IBM NALAIYATHIRAN PROJECT REPORT

**PLASMA DONOR APPLICATION**

|  |  |
| --- | --- |
| **Team Id** | **PNT2022TMID02610** |
| **Project Name** | **Plasma Donor Application** |
| **Team Members** | **- NITHYASHREE KP**  **- SHAINA S**  **- LEESHA**  **- STALIN** |

# Table Of Contents

|  |  |  |
| --- | --- | --- |
| **SI**  **No** | **Title** | **Page No** |
| 1 | **INTRODUCTION**   * 1. Project Overview   2. Purpose | 4  4 |
| 2 | **LITERATURE SURVEY**   * 1. Existing problem   2. References   3. Problem Statement Definition | 5  5  5 |
| 3 | **IDEATION & PROPOSED SOLUTION**   * 1. Empathy Map Canvas   2. Ideation & Brainstorming   3. Proposed Solution   4. Problem Solution fit | 6  7  8  10 |
| 4 | **REQUIREMENT ANALYSIS**   * 1. Functional requirement   2. Non-Functional requirements | 11  11 |
| 5 | **PROJECT DESIGN**   * 1. Data Flow Diagrams   2. Solution & Technical Architecture   3. User Stories | 12  13  14 |

|  |  |  |
| --- | --- | --- |
| 6 | **PROJECT PLANNING & SCHEDULING**   * 1. Sprint Planning & Estimation   2. Sprint Delivery Schedule   3. Reports from JIRA | 15  16  16 |
| 7 | **CODING & SOLUTIONING**   * 1. Feature 1   2. Feature 2   3. Database Schema (if Applicable) | 17  18  18 |
| 8 | **TESTING**   * 1. Test Cases   2. User Acceptance Testing | 20  21 |
| 9 | **RESULTS**  9.1 Performance Metrics | 22 |
| 10 | **ADVANTAGES & DISADVANTAGES** | 28 |
| 11 | **CONCLUSION** | 29 |
| 12 | **FUTURE SCOPE** | 29 |
| 13 | **APPENDIX**   * 1. Source Code   2. Github & Demo link | 30  42 |

* 1. **Project Overview**

# INTRODUCTION

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.

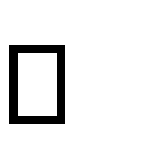
## Purpose

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.

# LITERATURE SURVEY

## 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

## 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.

## Problem Statement Definition

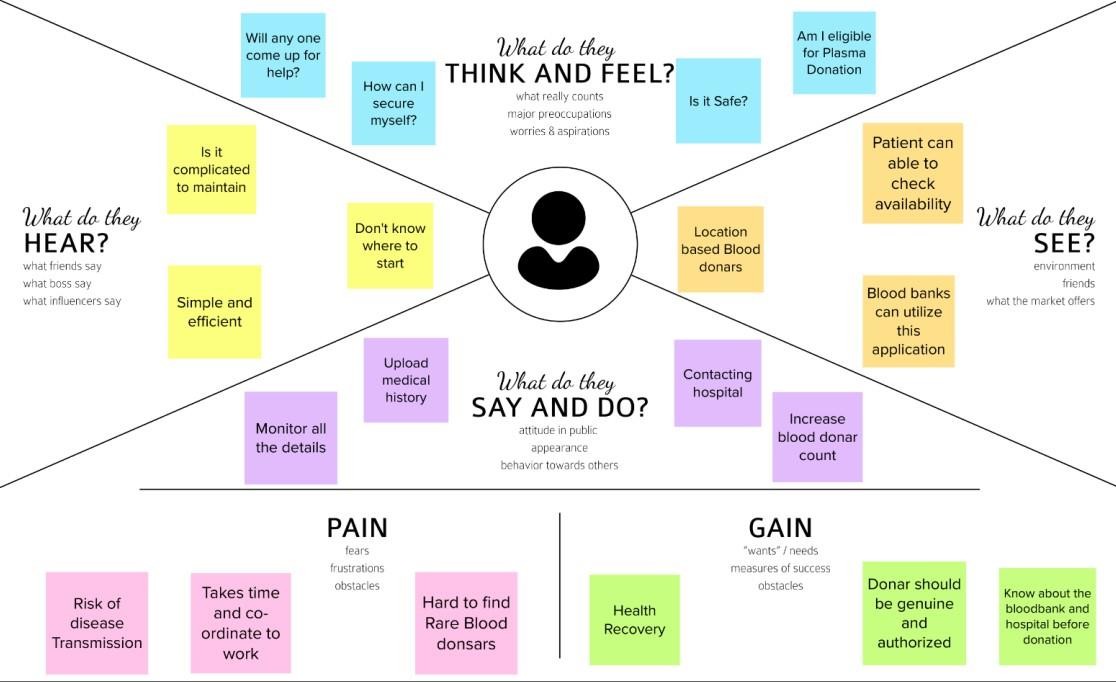
During COVID 19 crisis the requirement for plasma increased drastically as there were no vaccinations found in order to treat the infected patients.

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.

## IDEATION AND PROPOSED SYSTEM

* 1. **Empathy Map Canvas**



## Ideation and Brainstorming



**2**

**Brainstorm**

Write down any ideas that come to mind that address your problem statement.

**10 minutes**

**leesha**

**STALIN**

**nithyashree**

**SHAINA**

symptomatic at time of contracting the disease

healthy body

women with no kids

donor found

blood request

register to the system

not suffering from any disease

tested negative for the virus

age limit

check database

database

broadcast request

immunization

health screening

recharge and re- hydrate

donor volumetric analysis

health system science

determining bioavailability

health history questionnarie

blood draw

register required

develpoing an Ai model for classification

clinical decision making

pathology and radiology

**TIP**

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

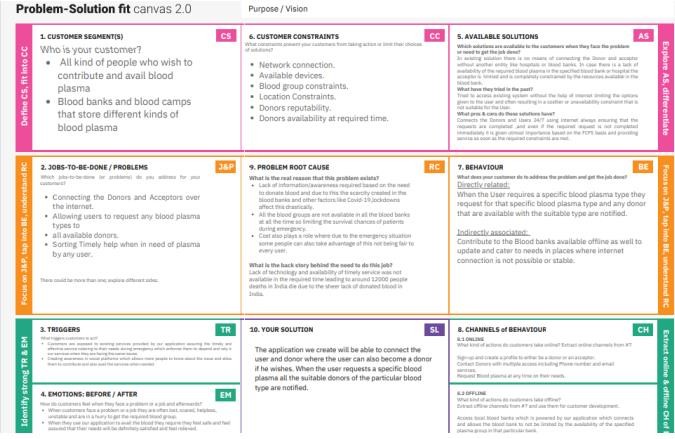
* 1. **Proposed Solution**

|  |  |  |
| --- | --- | --- |
| **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 Solution | of | the | 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. |

* 1. **Problem Statement Fit**



# REQUIREMENT ANALYSIS

## Functional Requirements:

Following are the functional requirements of the proposed solution.

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

|  |  |  |
| --- | --- | --- |
| FR-1 | **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. |

## Non-functional Requirements:

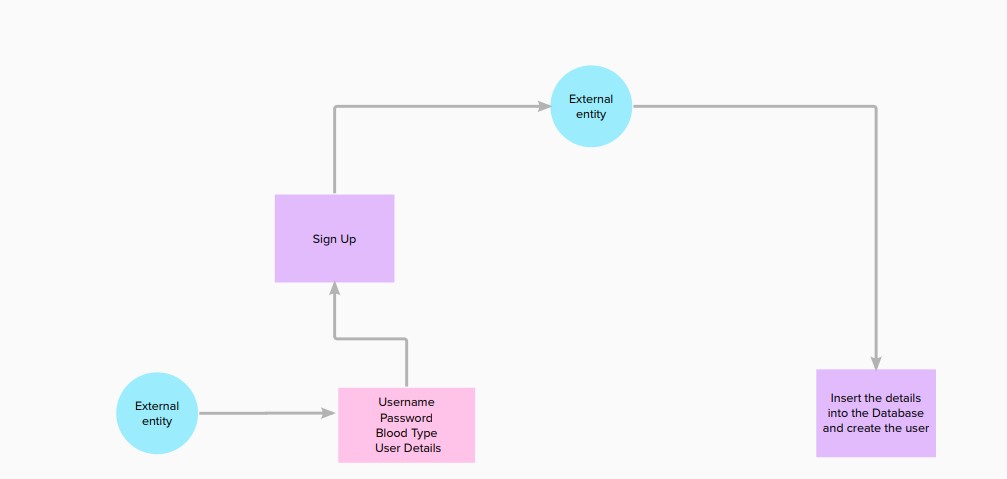
Following are the non-functional requirements of the proposed solution.

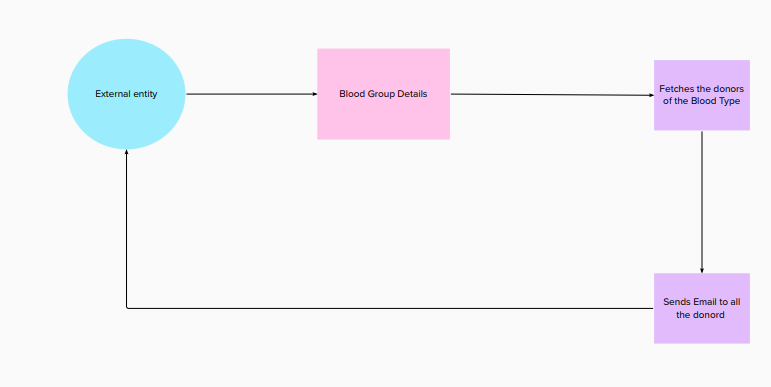
|  |  |  |
| --- | --- | --- |
| **FR**  **No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | The plasma donor application must have a good looking user friendly interface. |
| NFR-2 | **Security** | The plasma donor application must be secured with proper user name 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. |

* 1. **Data Flow Diagram:**

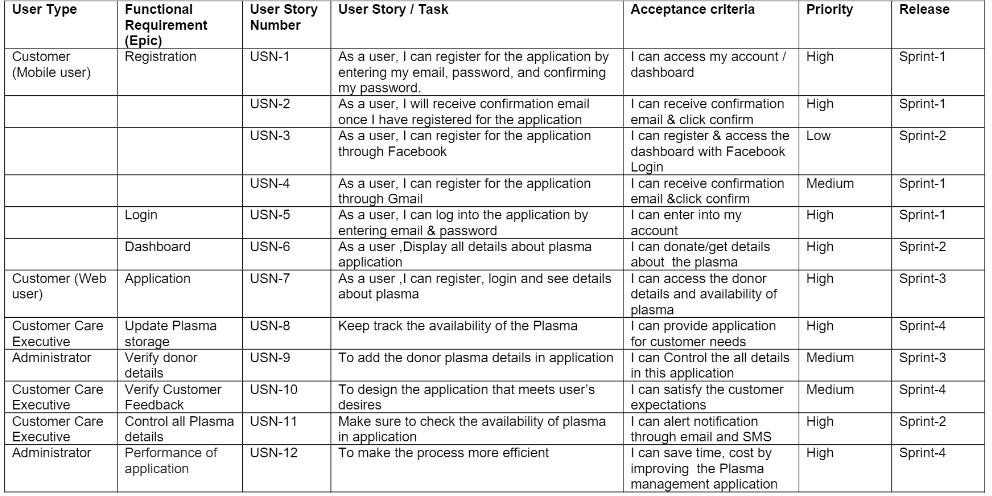
# PROJECT DESIGN





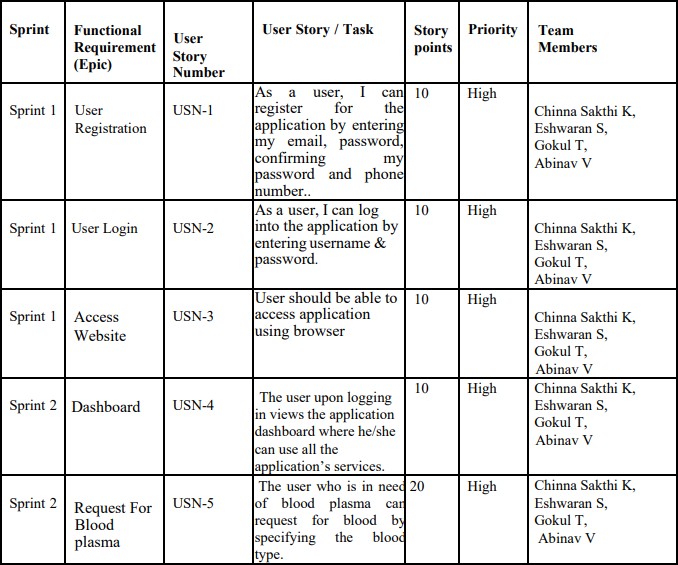
## Solution and Technical Architecture :

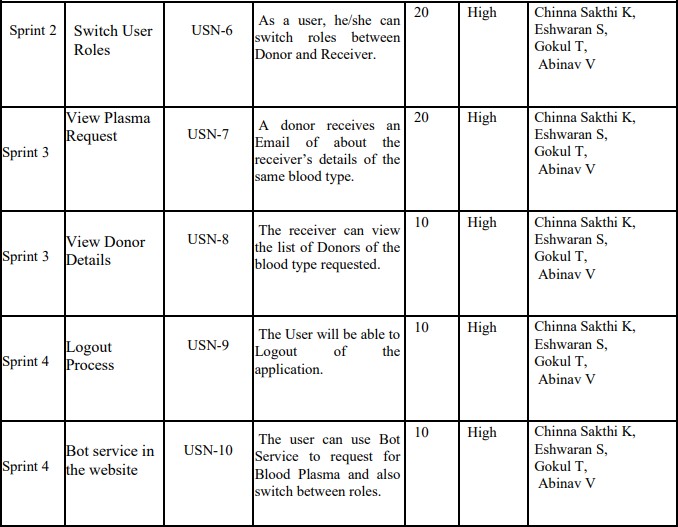
* 1. **User Stories:**



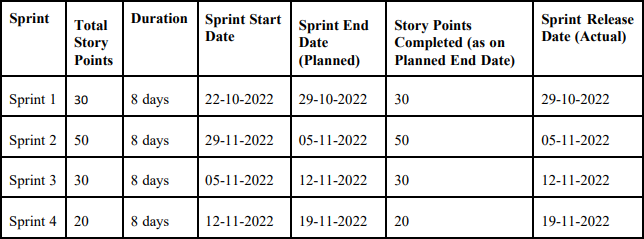
## PROJECT PLANNING AND SCHEDULING

* 1. **Sprint Planning & Estimation**

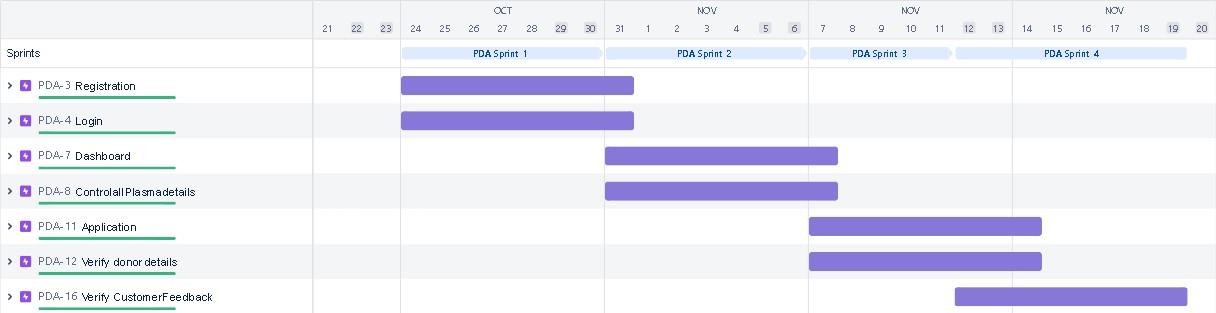




## Sprint delivery schedule



* 1. **Reports from JIRA**



## Feature 1:

1. **CODING & SOLUTIONING**

## 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,](https://gvanrossum.github.io/) 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.

## Feature 2:

**Flask**

* + - **Flask** is a micro [web framework](https://en.wikipedia.org/wiki/Web_framework) written in [Python.](https://en.wikipedia.org/wiki/Python_(programming_language)) It is classified as a [microframework](https://en.wikipedia.org/wiki/Microframework) because it does not require particular tools or libraries.
    - It has no [database](https://en.wikipedia.org/wiki/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,](https://en.wikipedia.org/wiki/Object%E2%80%93relational_mapping) form validation, upload handling, various open authentication technologies and several common framework related tools.
    - Applications that use the Flask framework include [Pinterest](https://en.wikipedia.org/wiki/Pinterest) and [LinkedIn.](https://en.wikipedia.org/wiki/LinkedIn)

## Database Scheme

**IBM Db2**

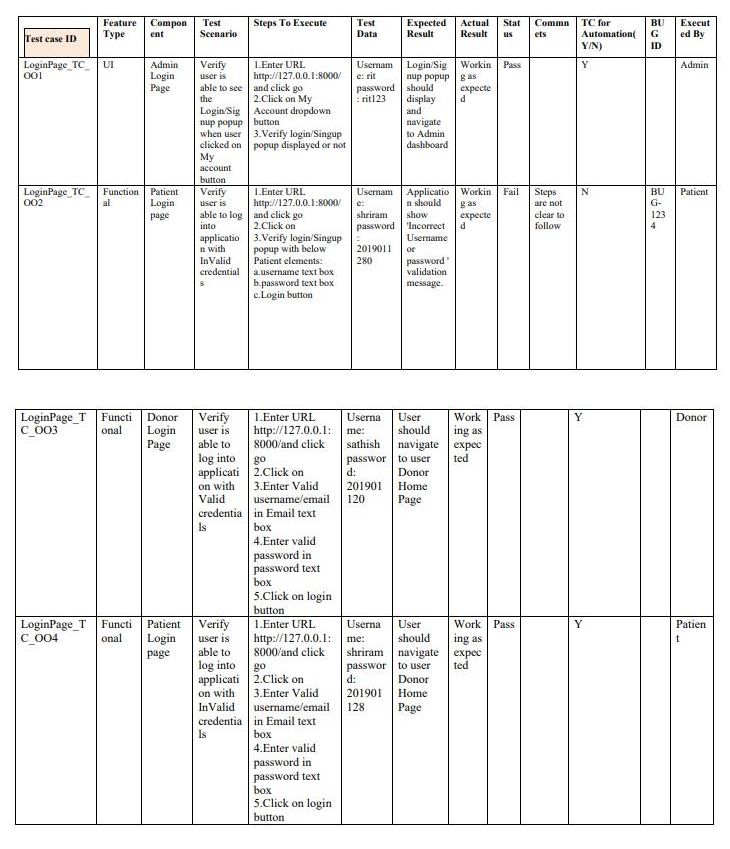
* + - DB2 is a database product from IBM.
    - It is a Relational Database Management System (RDBMS). DB2 is designed to store, analyze and retrieve the data efficiently.
    - DB2 product is extended with the support of Object-Oriented features and non-relational structures with XML.
    - Provide 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

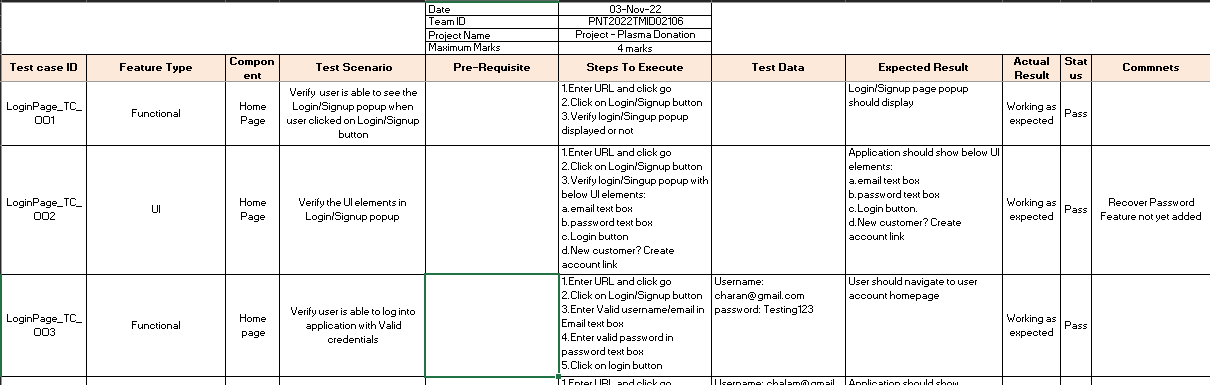
* + - **Kubernetes** is also known as **'k8s'.**
    - **Kubernetes** is an extensible, portable, and open-source platform designed by **Google** in **2014**.
    - It is mainly used to automate the deployment, scaling, and operations of the container-based applications across the cluster of nodes.
    - 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.
  1. **Test case**

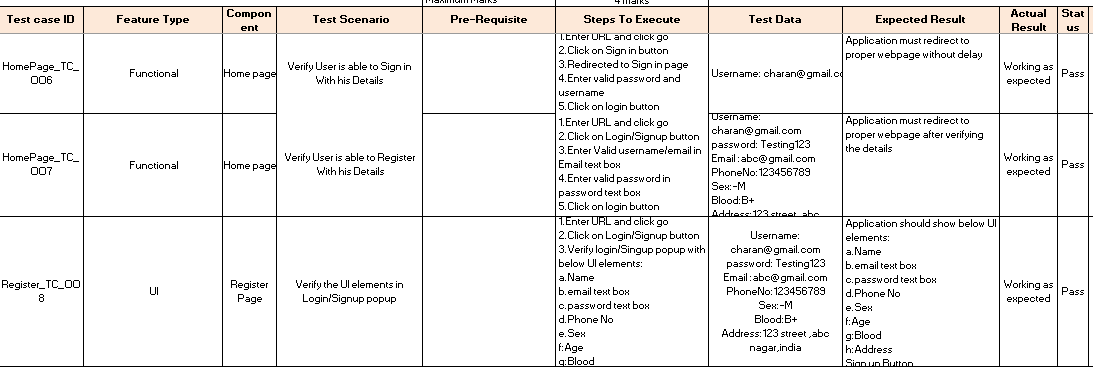
# TESTING

* + 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



## User Acceptance Testing





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).

## 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 | 7  1 |

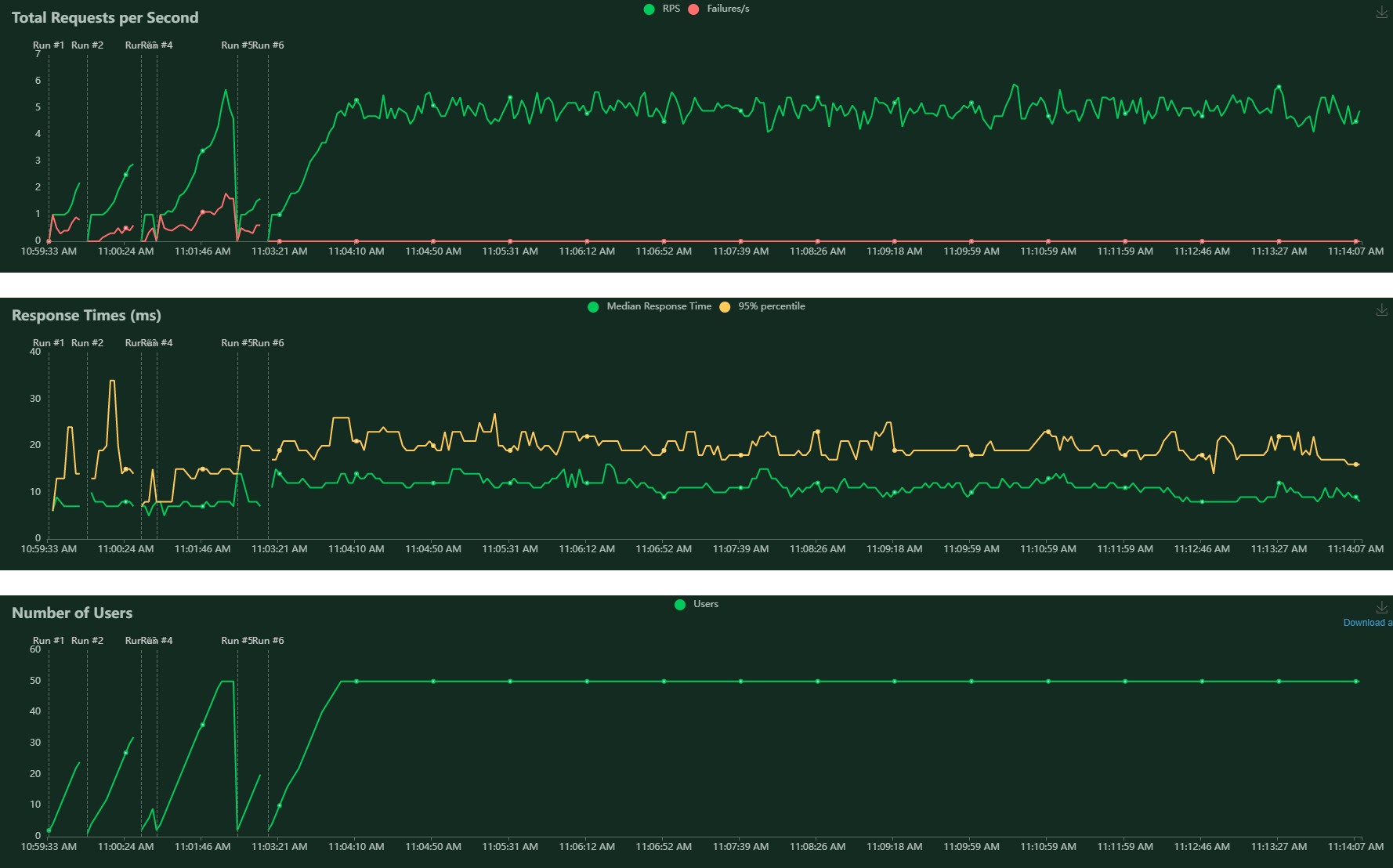
1. Test Case Analysis

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 |



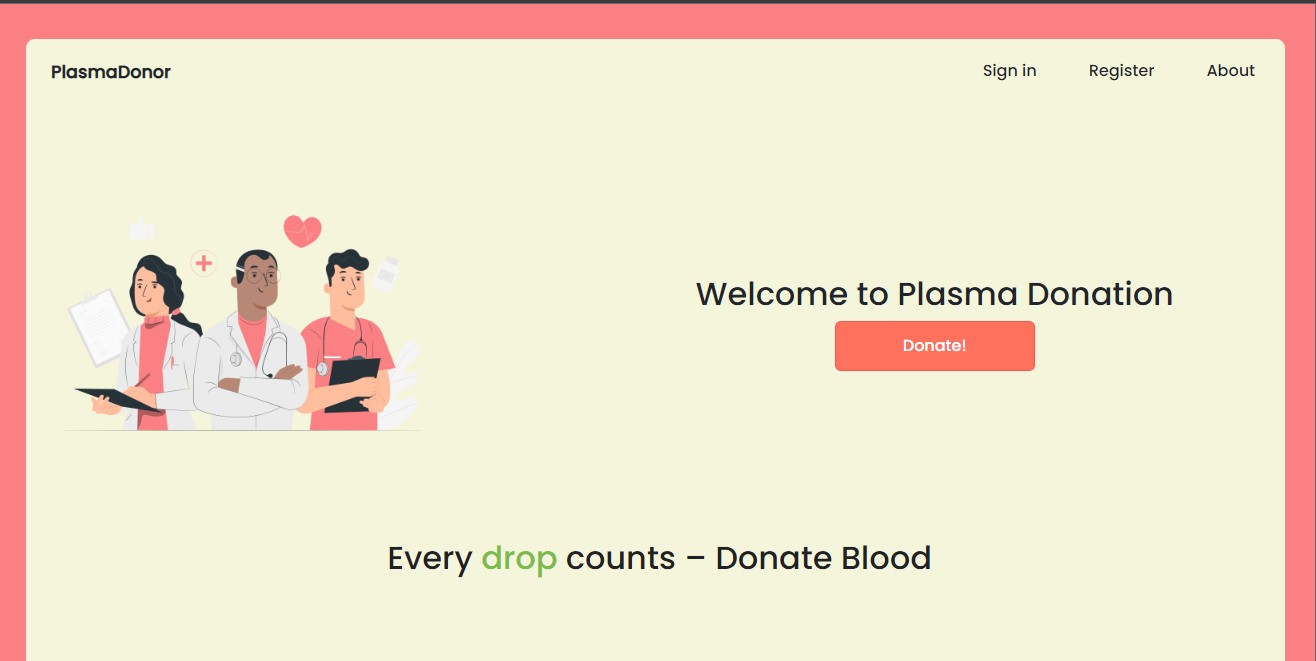


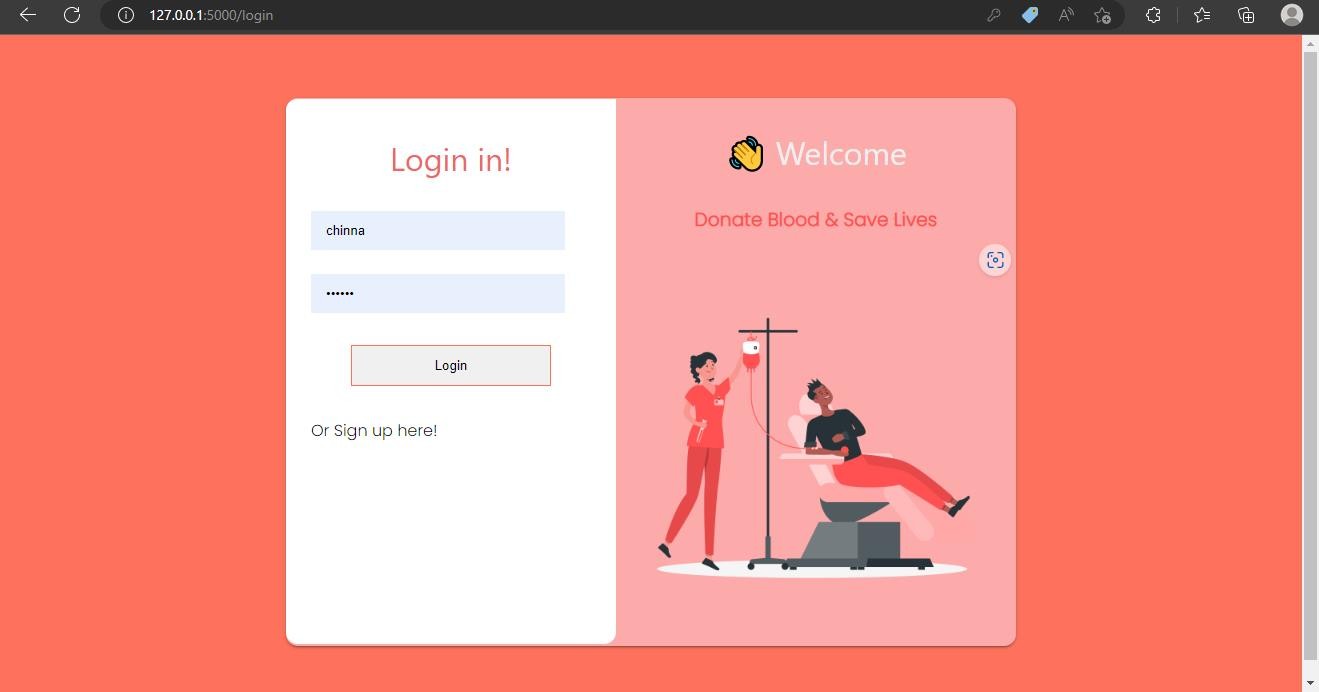
**9.1 Performance Metrics**

# RESULTS

* 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

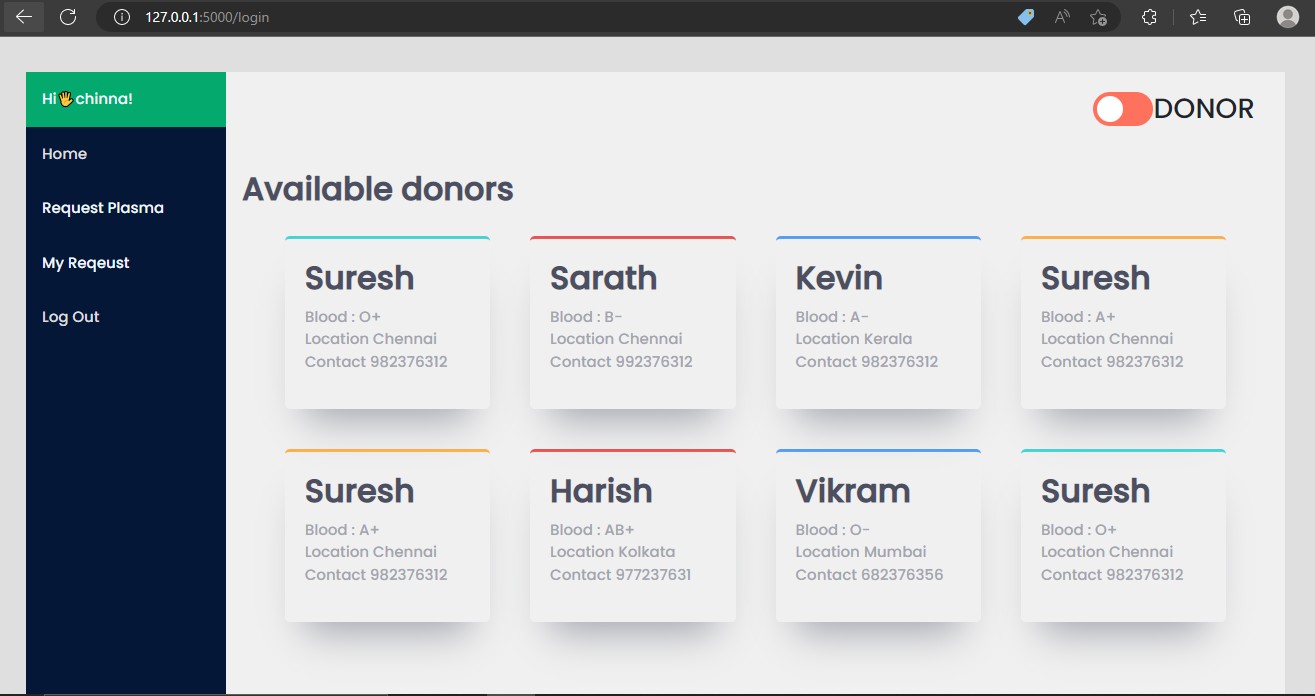
# OUTPUT SCREENS

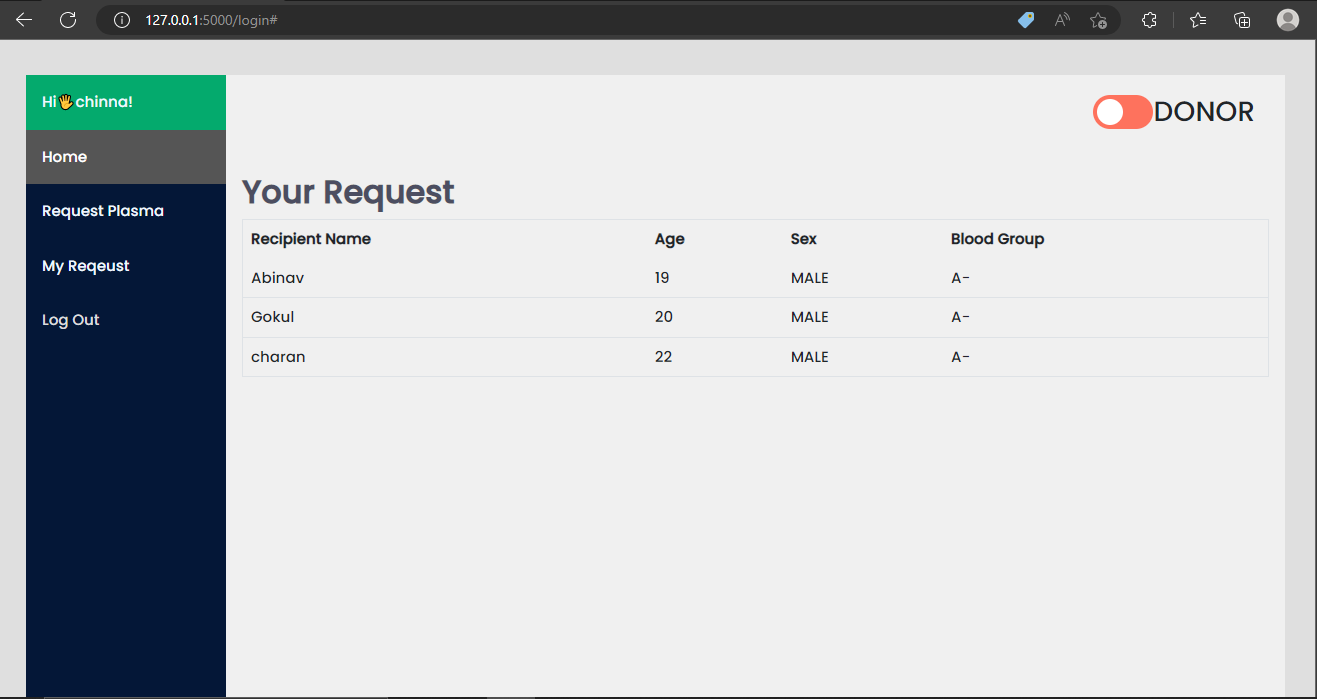
Landing page



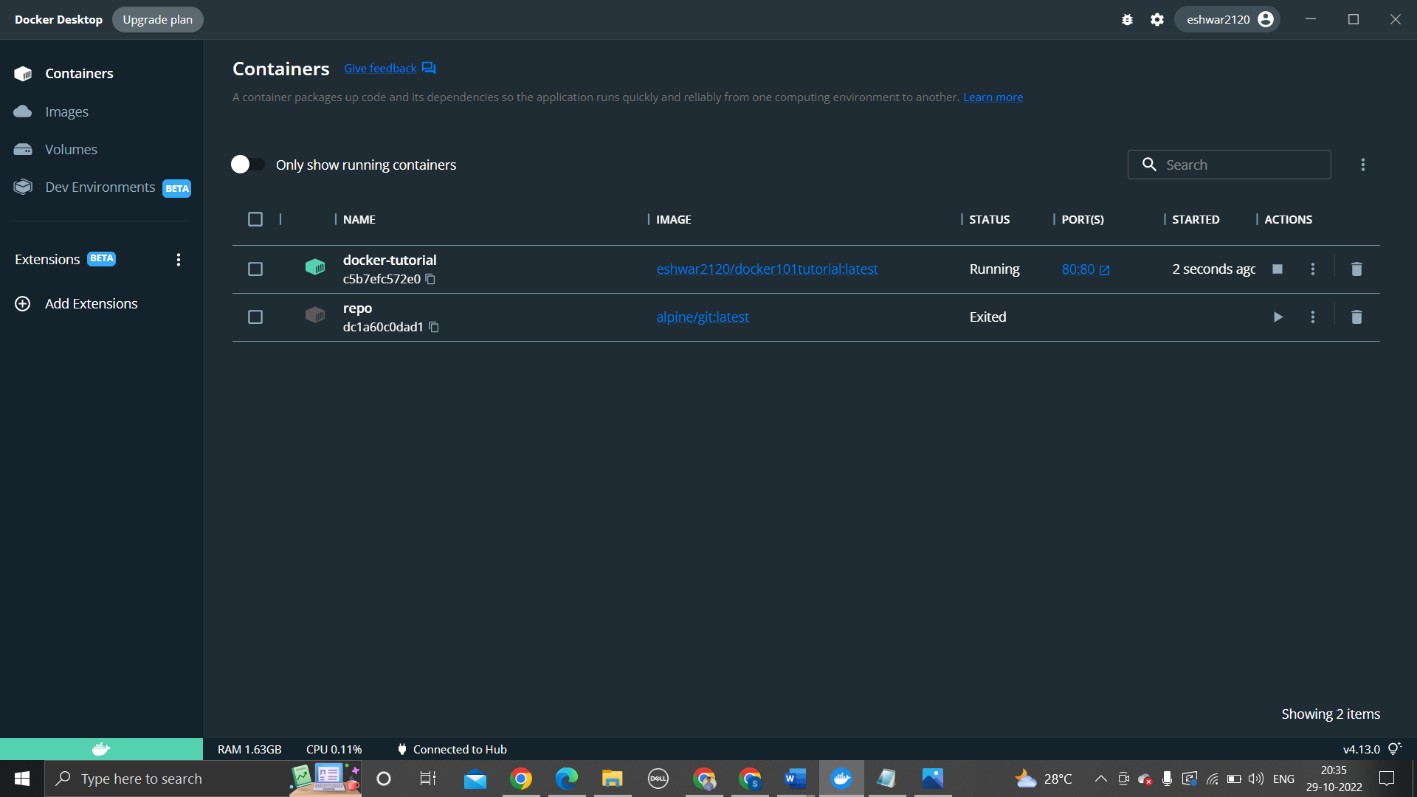
## Register Page

Dashboard

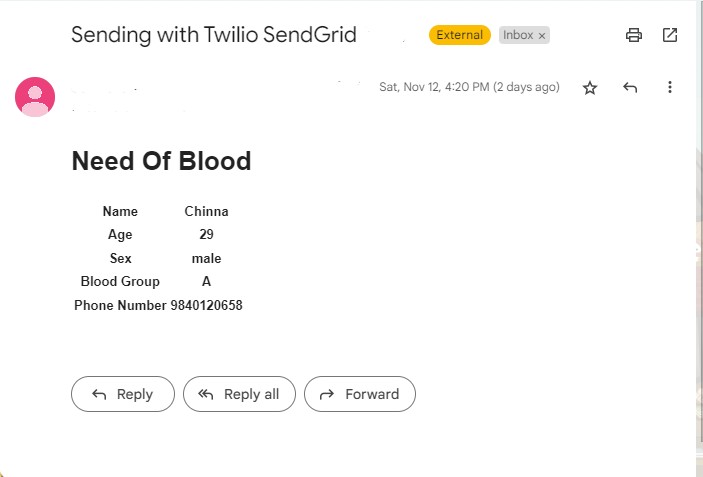




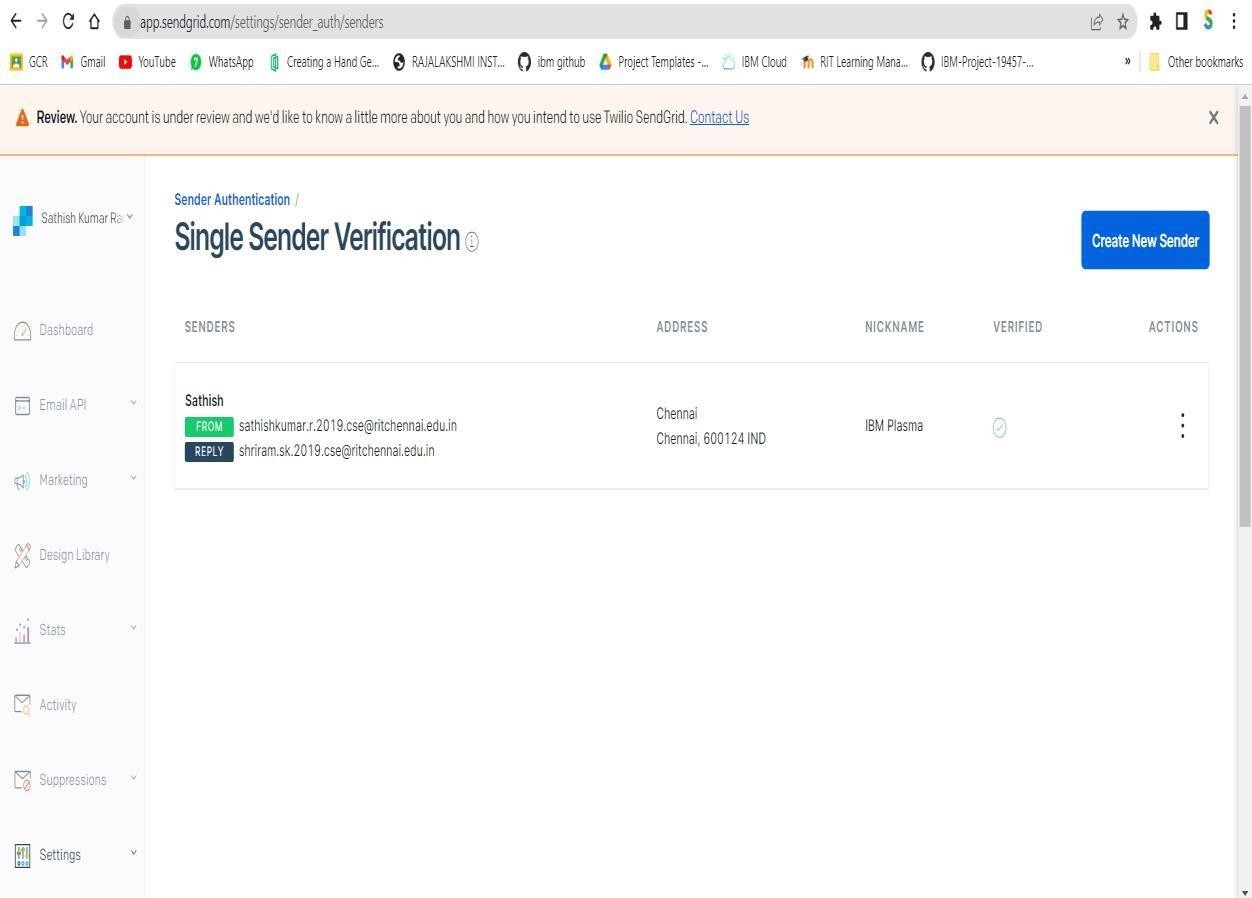
Dockerize the app



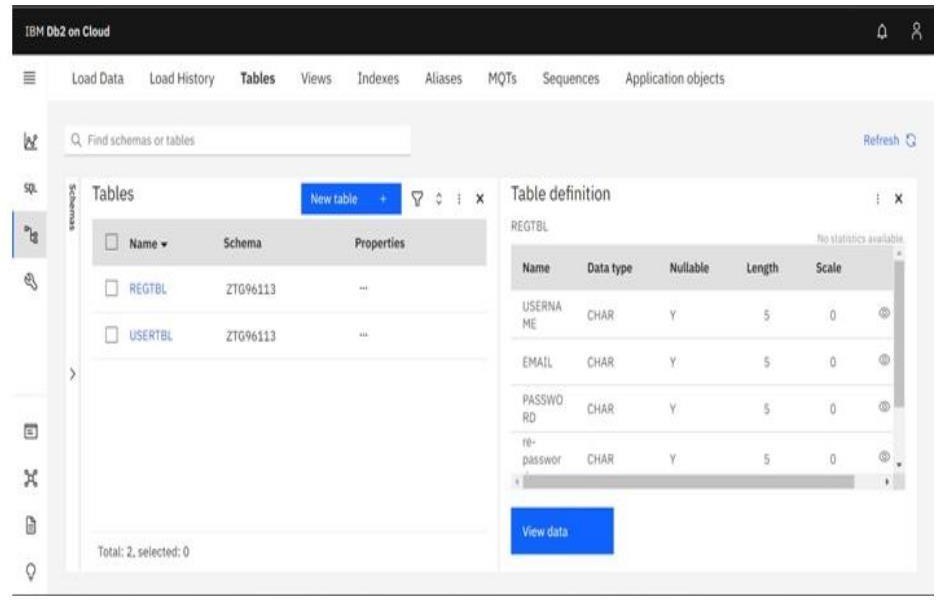
SENDING MAIL



## SEND GRID



**IBM DB2**



# 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.

# 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.

# 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

# APPENDIXES

## 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='nithyashree.kp.2019.it@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)

## GITHUB

https://github.com/IBM-EPBL/IBM-Project-18642-1659687967

## PROJECT DEMO LINK

https://drive.google.com/file/d/11JYD99Z6\_l05vXJUrrs40RTVpGu\_-y4a/view?usp=sharing