IBM NALAIYATHIRAN
PROJECT REPORT

PLASMA DONOR APPLICATION

Team id	**PNT2022TMID04796**
Project Name	**Plasma Donor Application**
Team Members	**Aravind S** **Arul Prasanna S** **ArulMurugan P** **Arun K**

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 web-based system is available no mobile 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

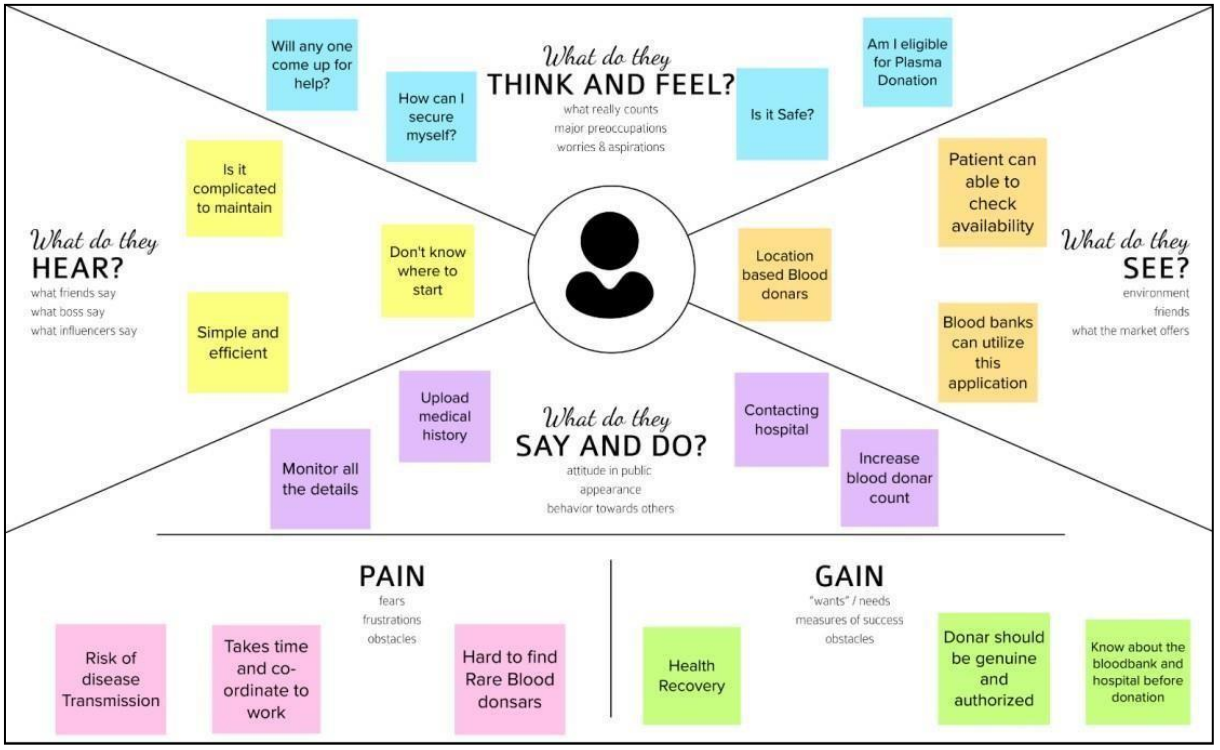
During COVID 19 crisis the requirement for plasma increased drastically as there were no vaccinations found in order to treat the infected patients.

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.

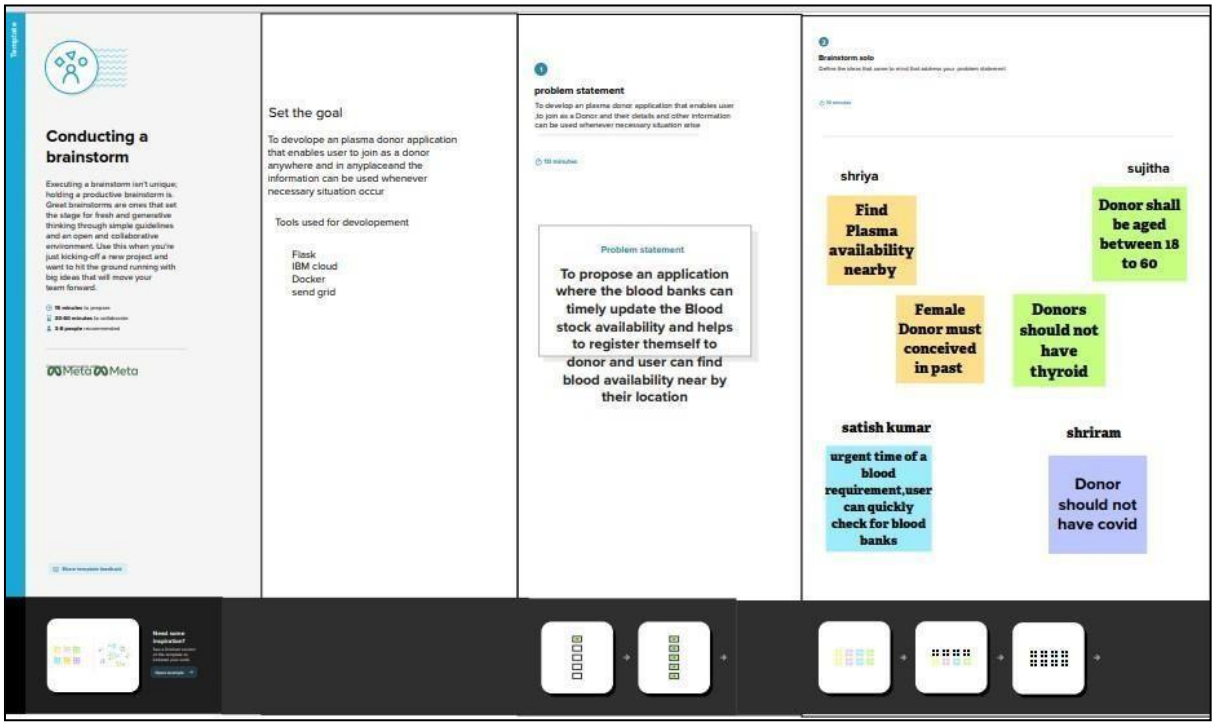
As the plasma therapy was one of the ways to treat the infected patients getting the donor details played a major role.

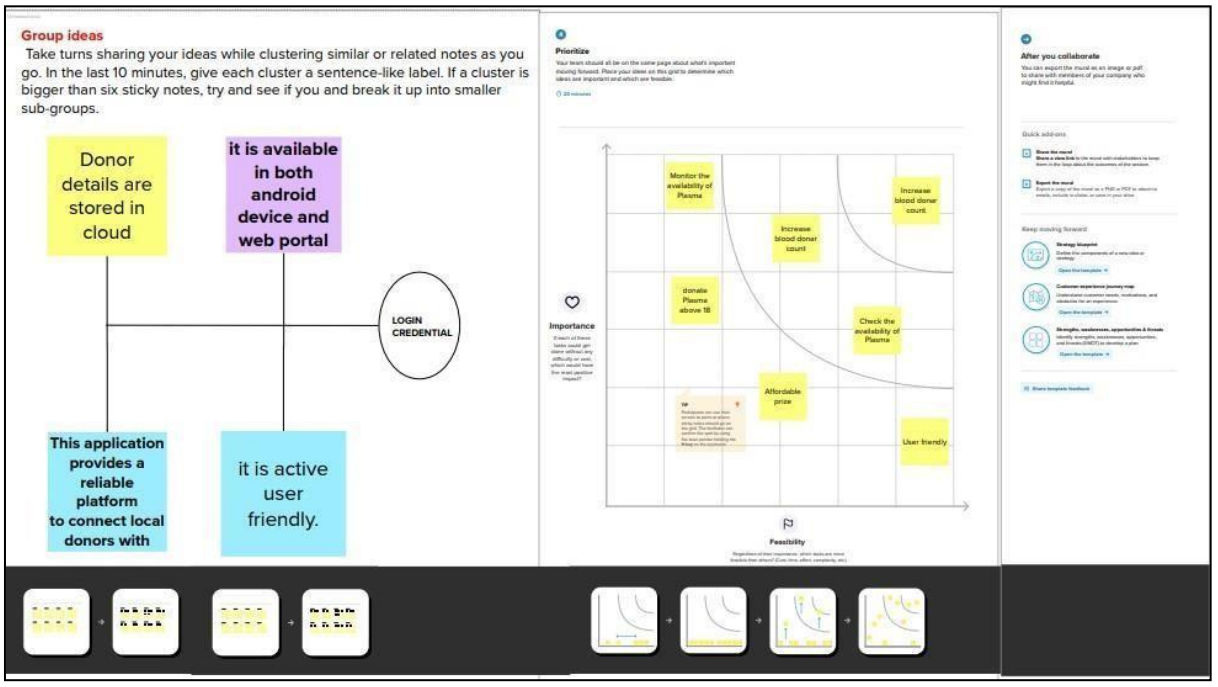
3. IDEATION AND PROPOSED SYSTEM

3.1 Empathy Map Canvas



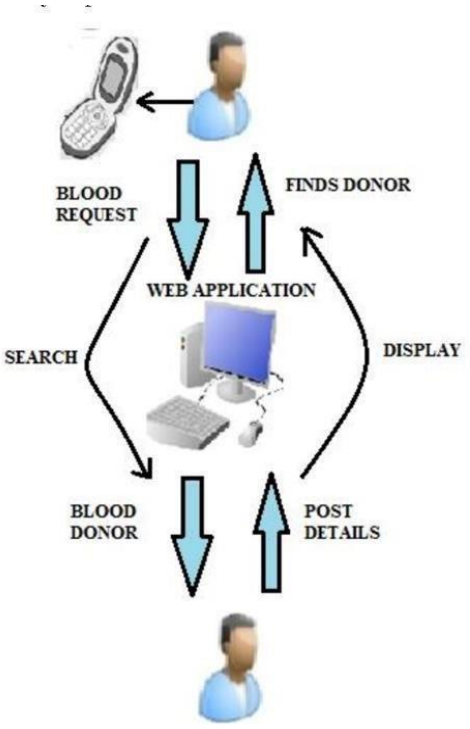
3.2 Ideation and Brainstorming



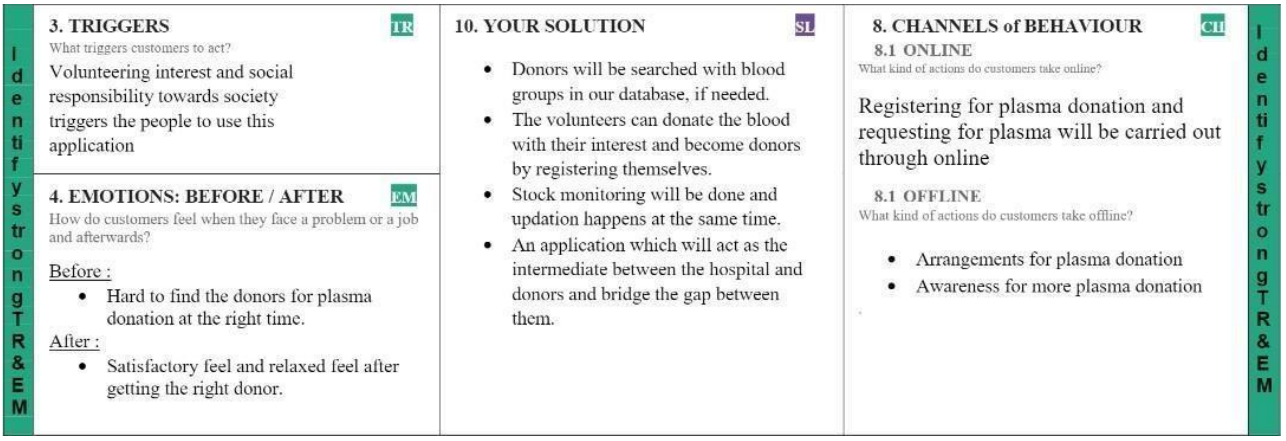


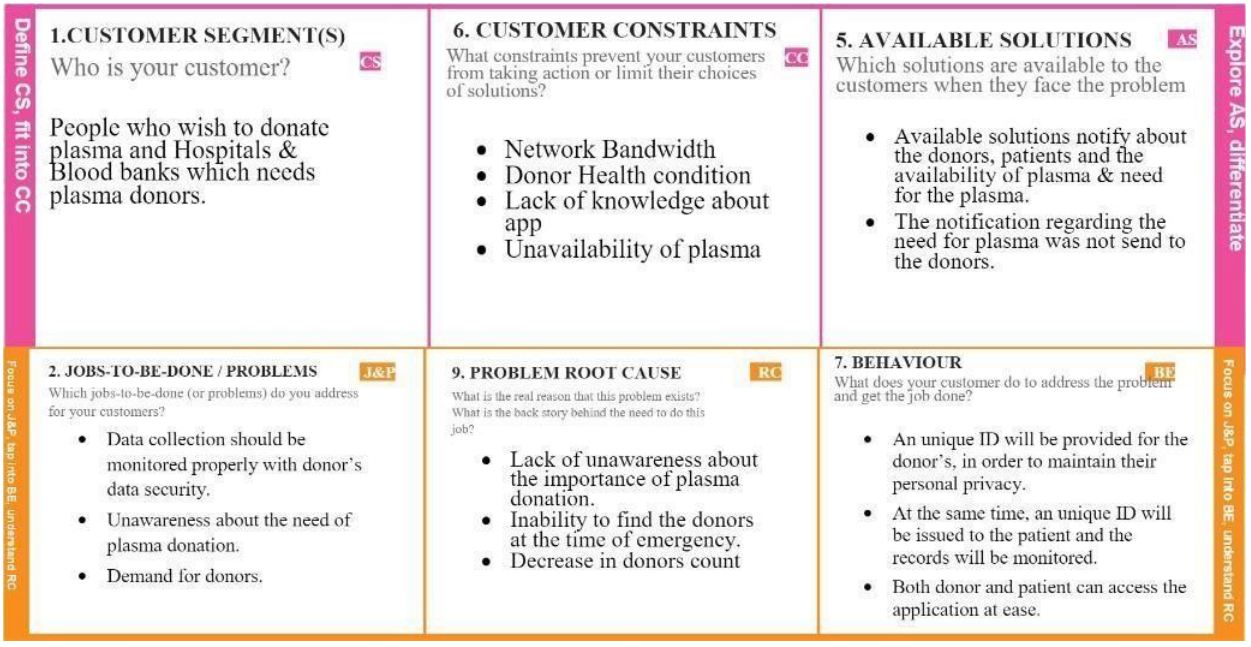
3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The main aim of this project is to help the people who needs blood in emergency and to associat some donors who are willing to donate their blood to needy people and save their lives.
2.	Idea / Solution description	The user will be able to : ➢ Search donors of suitable blood groups and contac them if needed. ➢ Donate blood by registering themselves with our system and can also become donors. ➢ Will be able to see the stock of various blood groups. ➢ Send request for blood via "contact us". ➢ Get information about all the blood campaigns..
3.	Novelty / Uniqueness	All of them have different ideas and different queries. Based on the user request and experience we will update our project based on user convenience .
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)	Hospitals, NGOs, and private groups will profit from this donation application. Anyone with a basic understanding can use this software. This can be utilised at any time, anywhere. working With the assistance of the government, we can a programme to assist persons in need of plasma.

		
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





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:

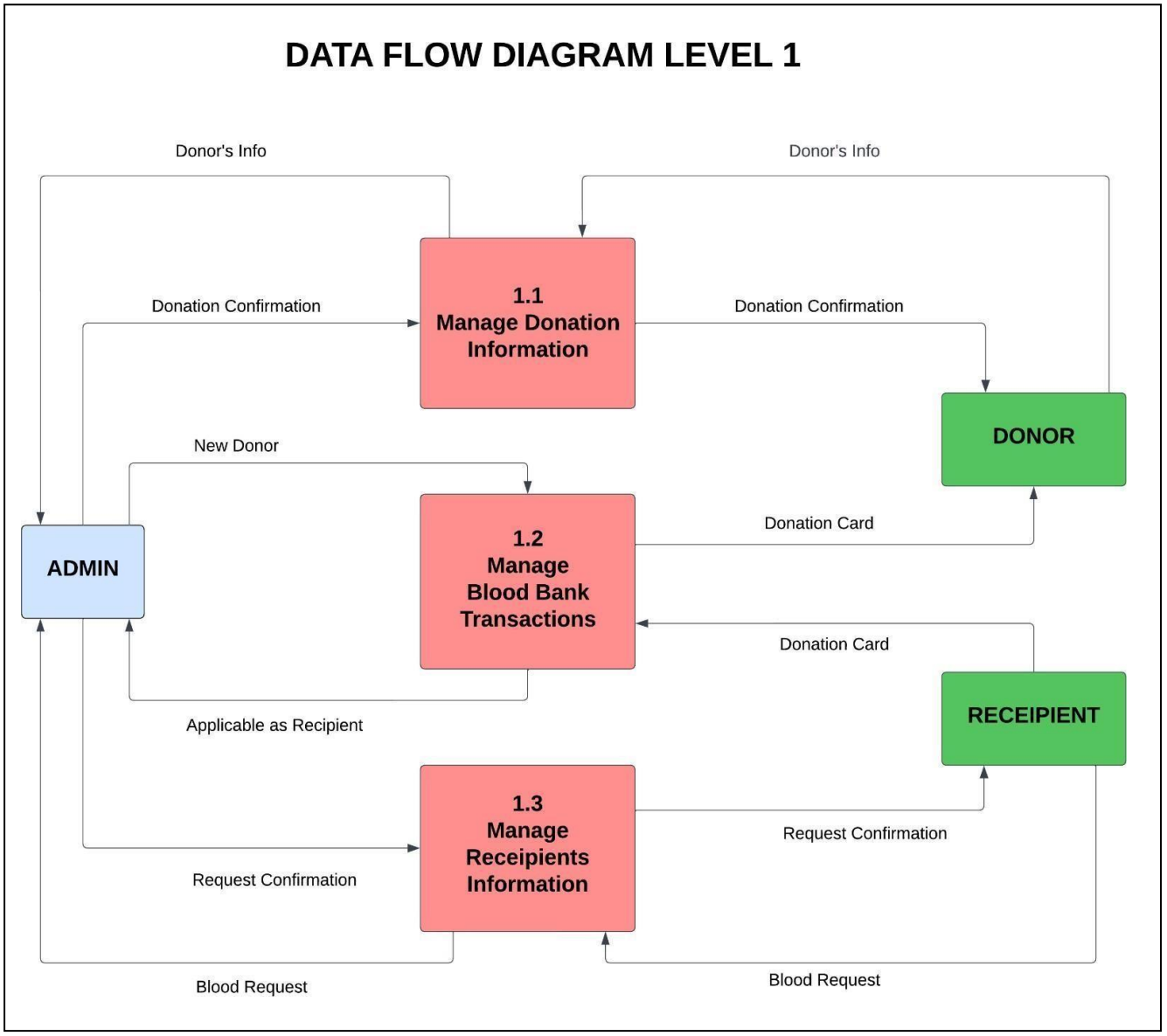
Following are the non-functional requirements of the proposed solution.

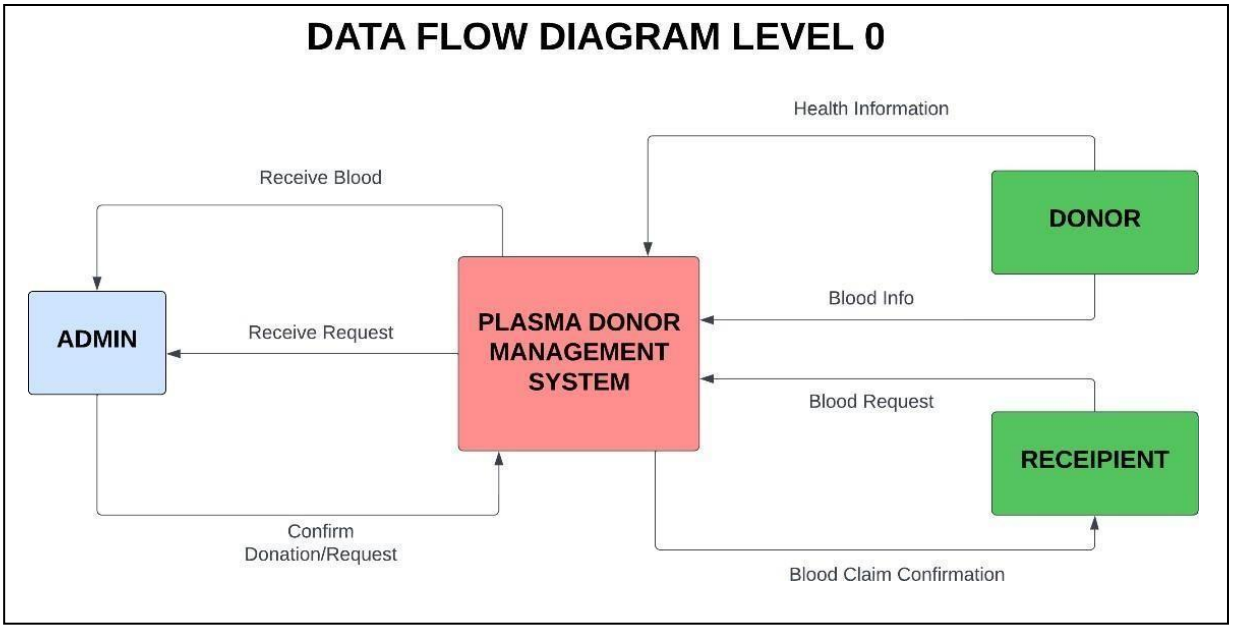
FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The plasma donor application must have a good looking user friendly interface.
NFR-2	**Security**	The plasma donor application must be secured with proper user name and passwords.
NFR-3	**Reliability**	The plasma donor application should 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.

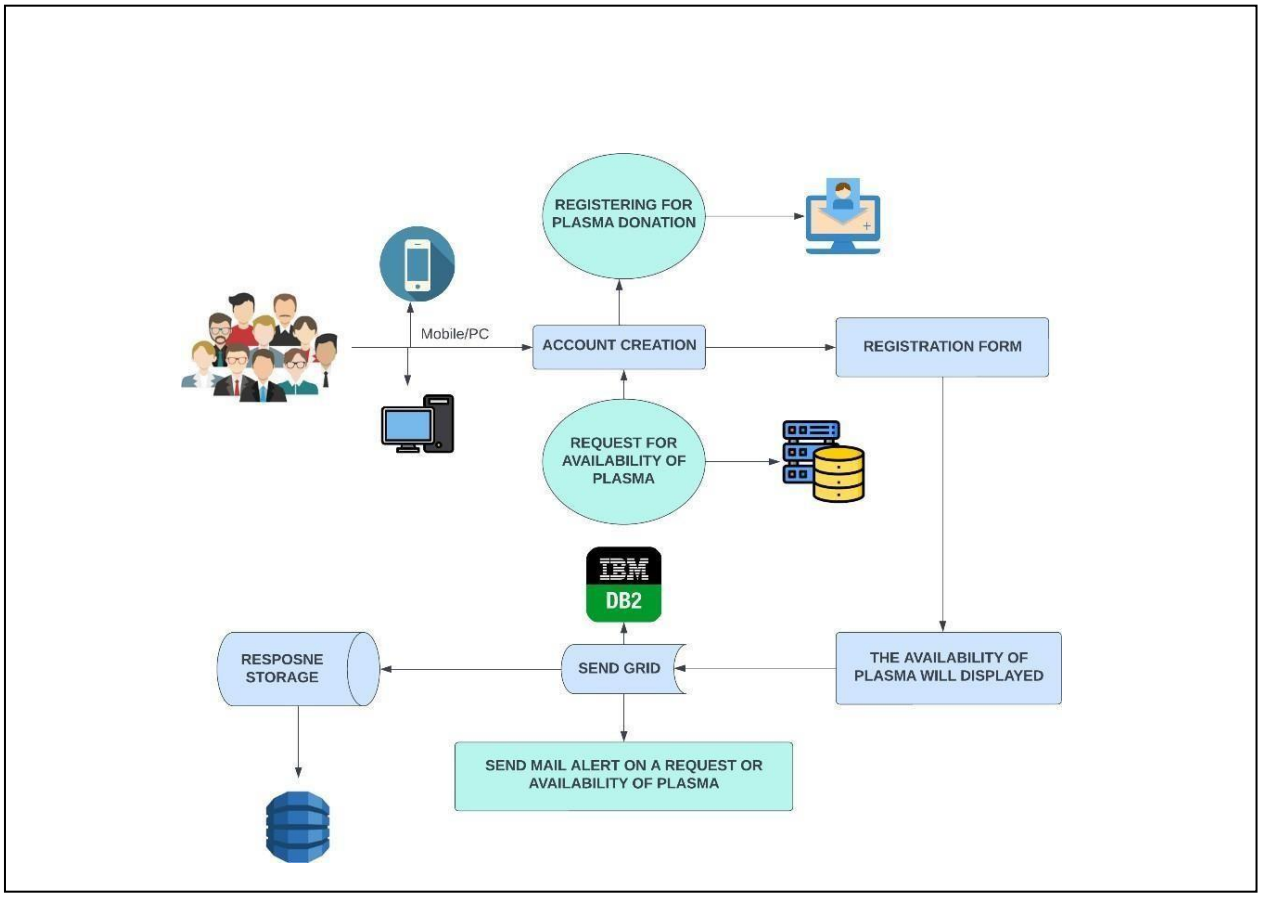
5. PROJECT DESIGN

5.1 Data Flow Diagram:





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

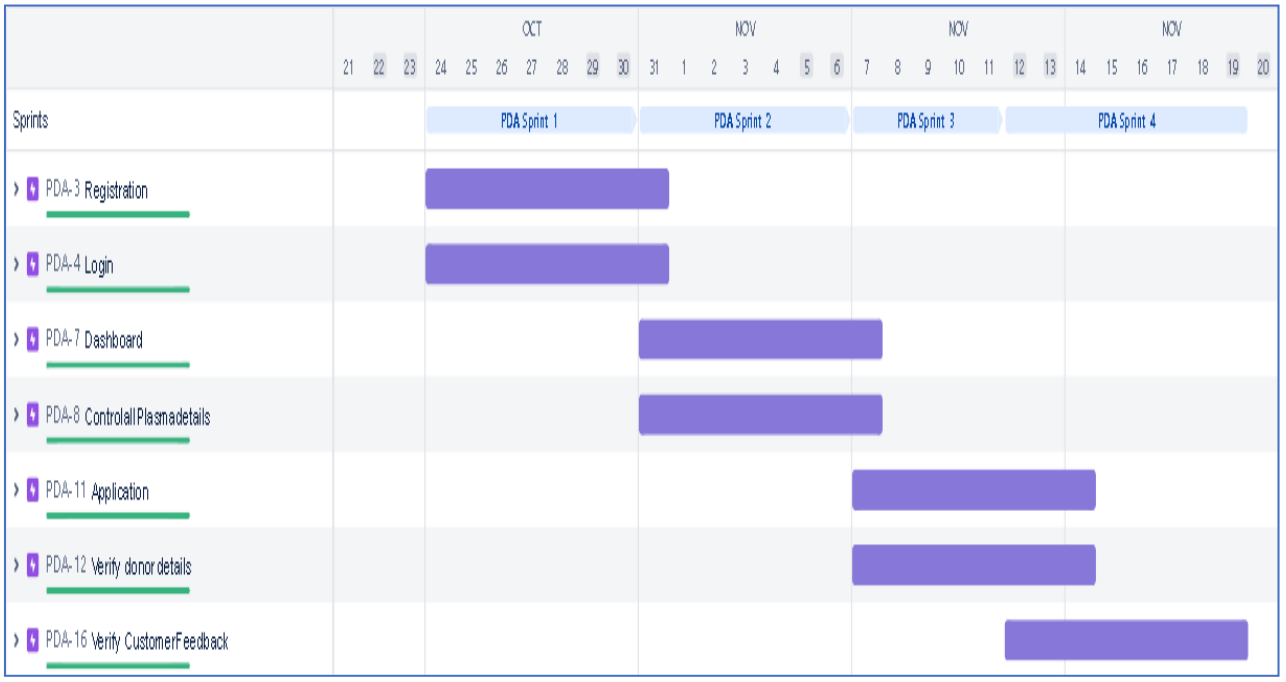
6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	1	High	Arul Prasanna Arun
Sprint-1	Login	USN-2	As a user, I can log into the application by entering email & password	1	High	Arun Arul Murugan
Sprint-2	Dashboard	USN-3	As a user ,Display all details about plasma application	1	High	Arul Prasanna Aravind
Sprint-3	Application	USN-4	As a user ,I can register, login and see details about plasma	1	High	ArulMurugan Arun
Sprint-3	Verify donor details	USN-5	To add the donor plasma details in application	1	Medium	Arun Aravind
Sprint-2	Control all Plasma details	USN-6	Make sure to check the availability of plasma in application	1	High	Arul Murugan Arul Prasanna
Sprint-4	Verify Customer Feedback	USN-7	To design the application that meets user's desires	2	Medium	Arul Prasanna Aravind

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	6 Days	25 Oct 2022	30 Oct 2022	30	30 Oct 2022
Sprint-2	30	6 Days	1 Nov 2022	6 Nov 2022	30	6 Nov 2022
Sprint-3	30	6 Days	8 Nov 2022	13 Nov 2022	30	13 Nov 2022
Sprint-4	30	6 Days	15Nov 2022	20 Nov 2022	30	20 Nov 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 general-purpose 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 higher-paying 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	Comments	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 : 20190112 80	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

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

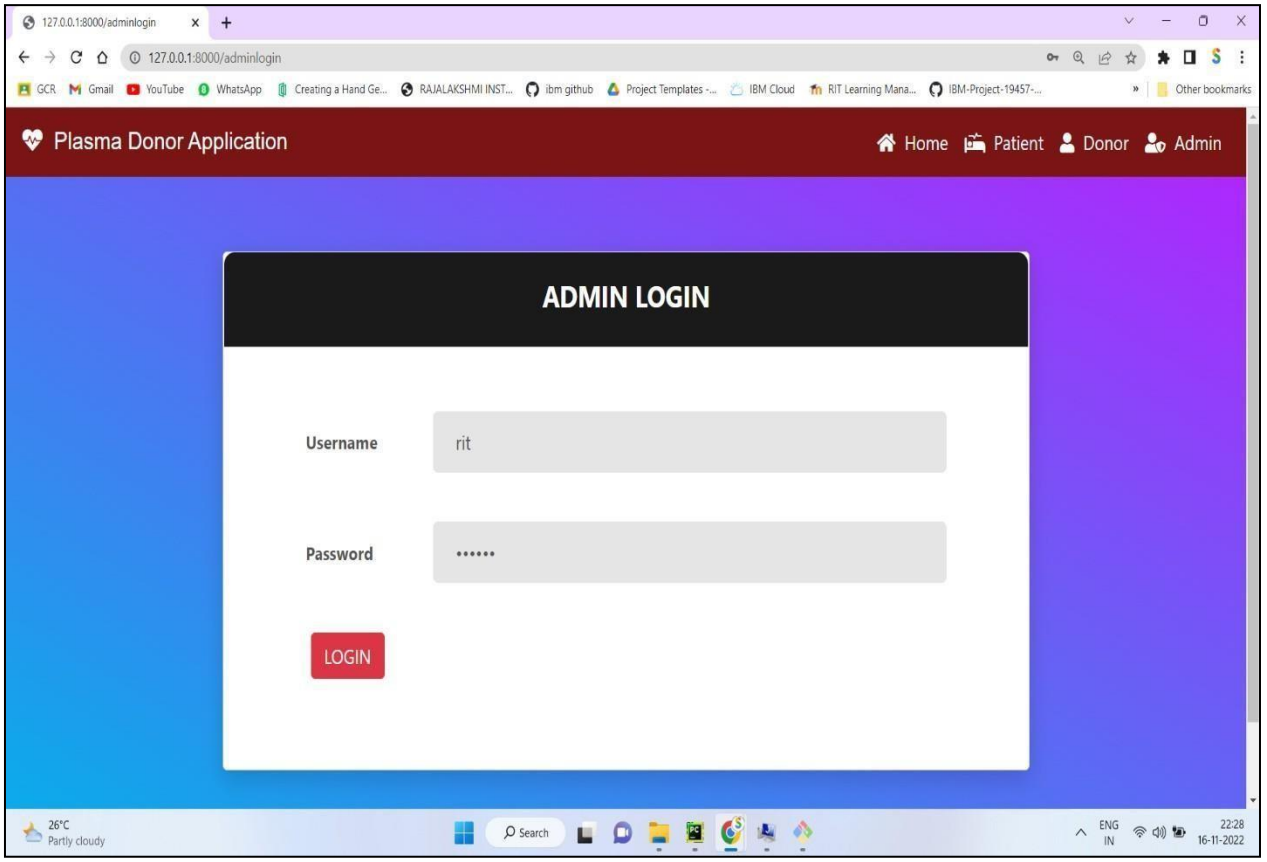
9. RESULTS

9.1 Performance Metrics

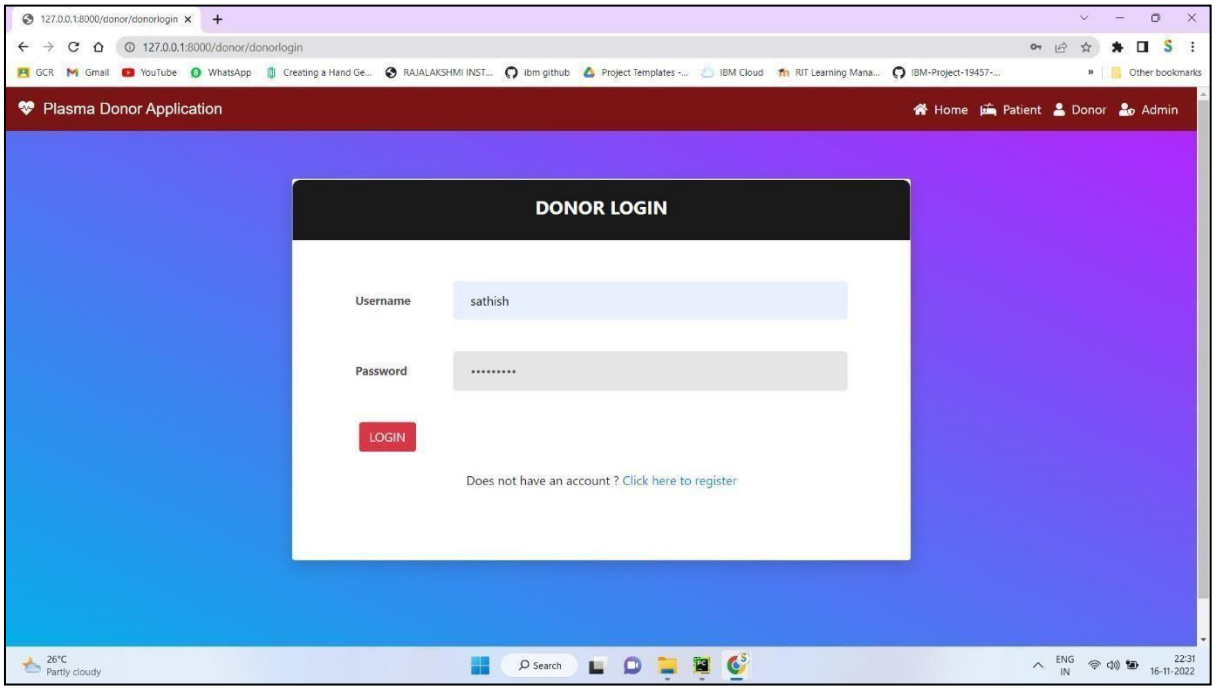
- Project metrics are used to track the progress and performance of a project.
- Monitoring parts of a project like productivity, scheduling, and scope make it easier for team leaders to see what's on track.
- As a project evolves, managers need access to changing
- deadlines or budgets to meet their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

OUTPUT SCREENS

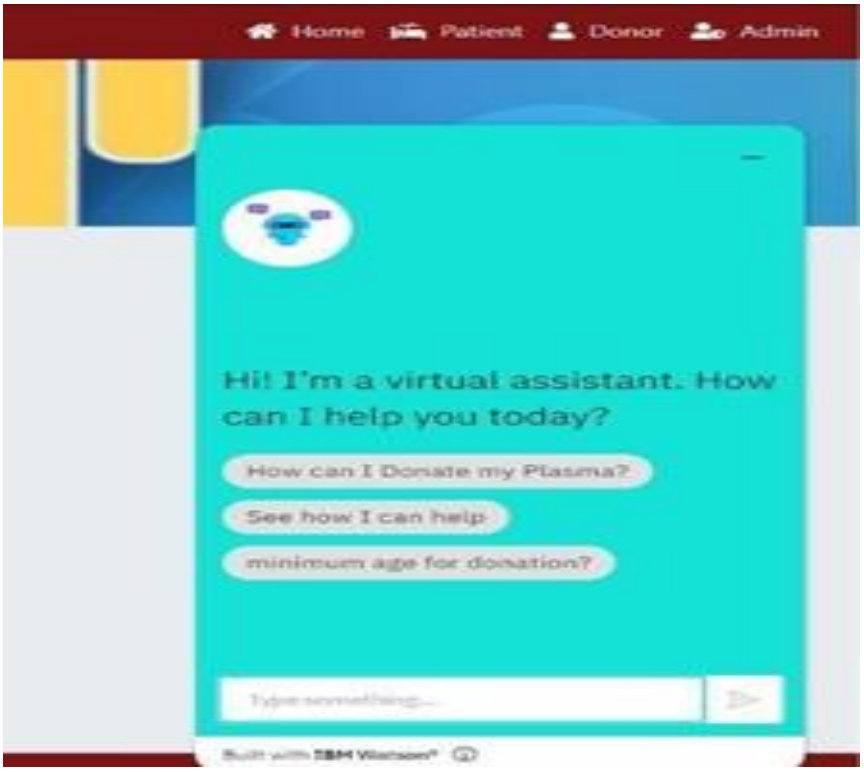
Login page



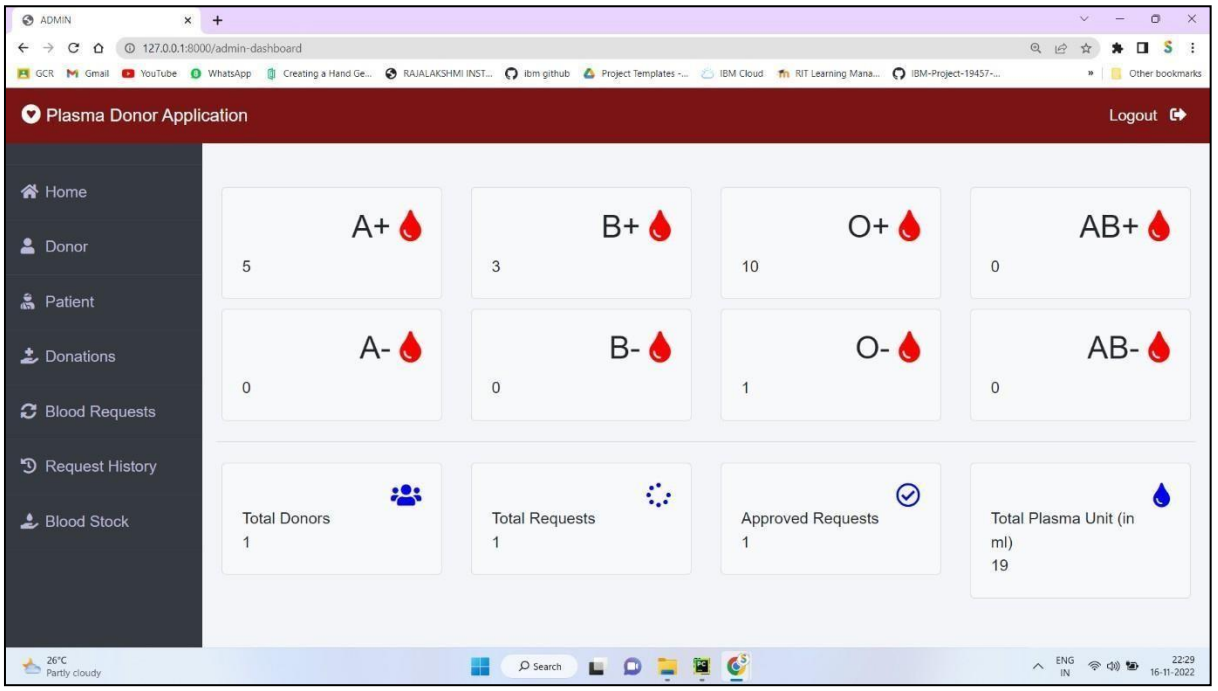
Donor Login



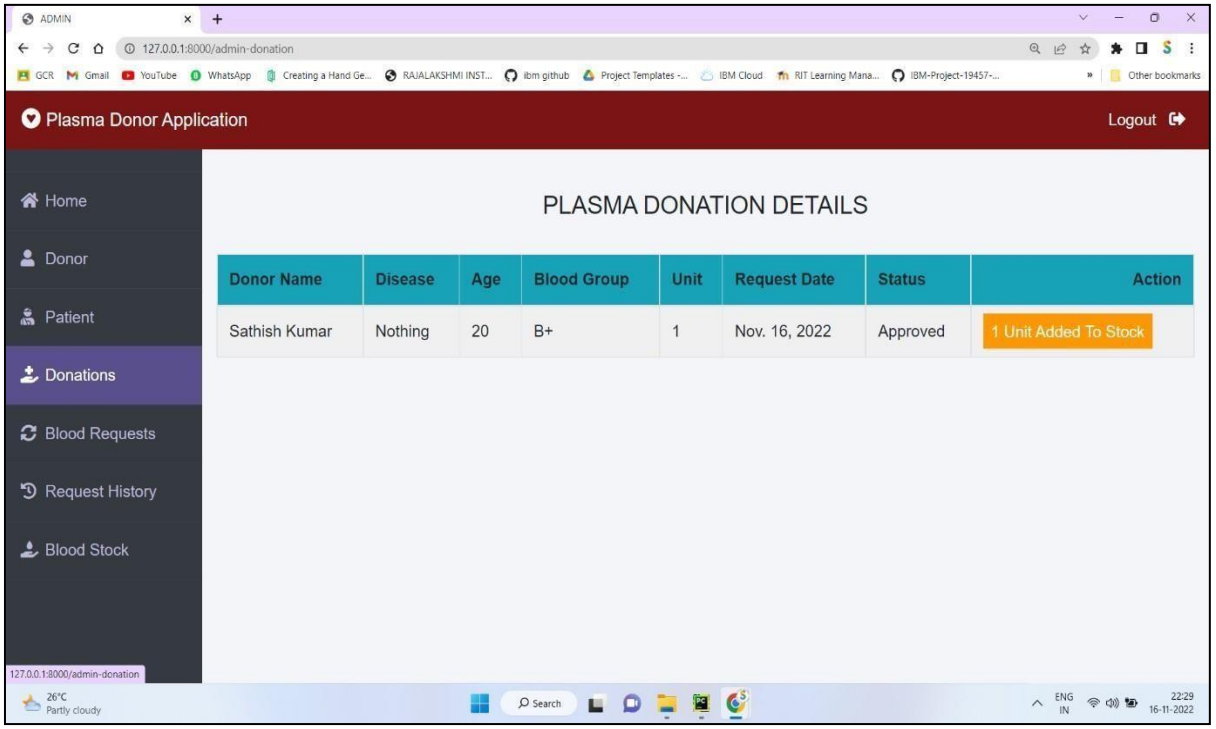
Register Page



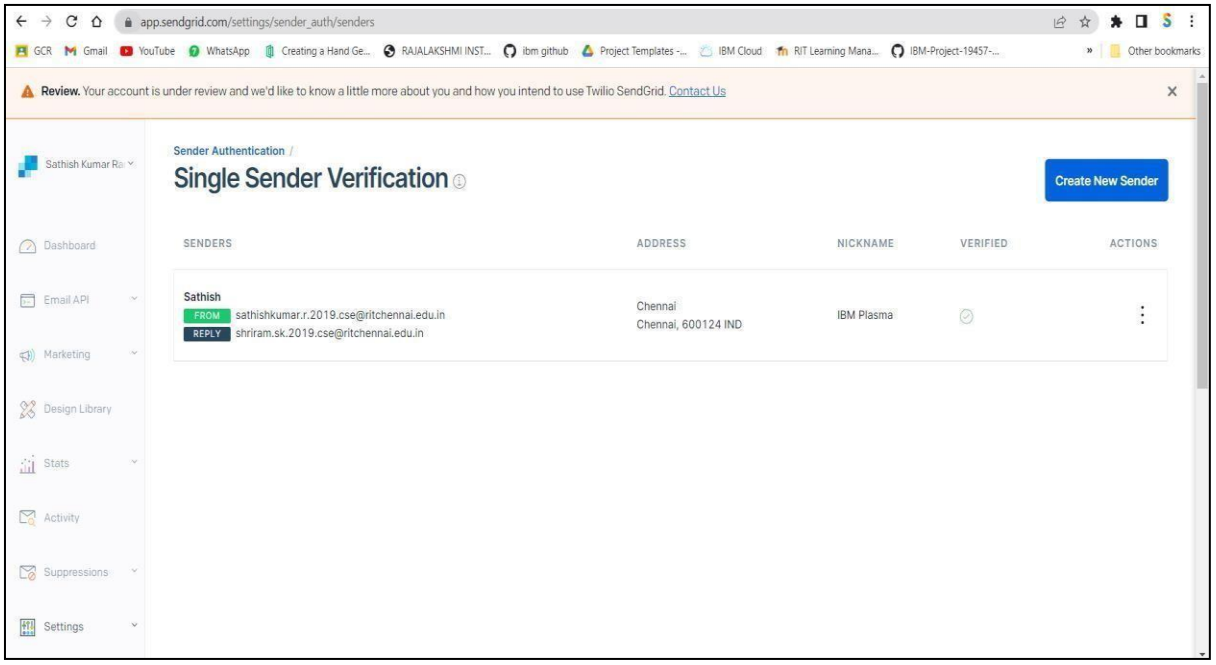
Admin Dashboard Page



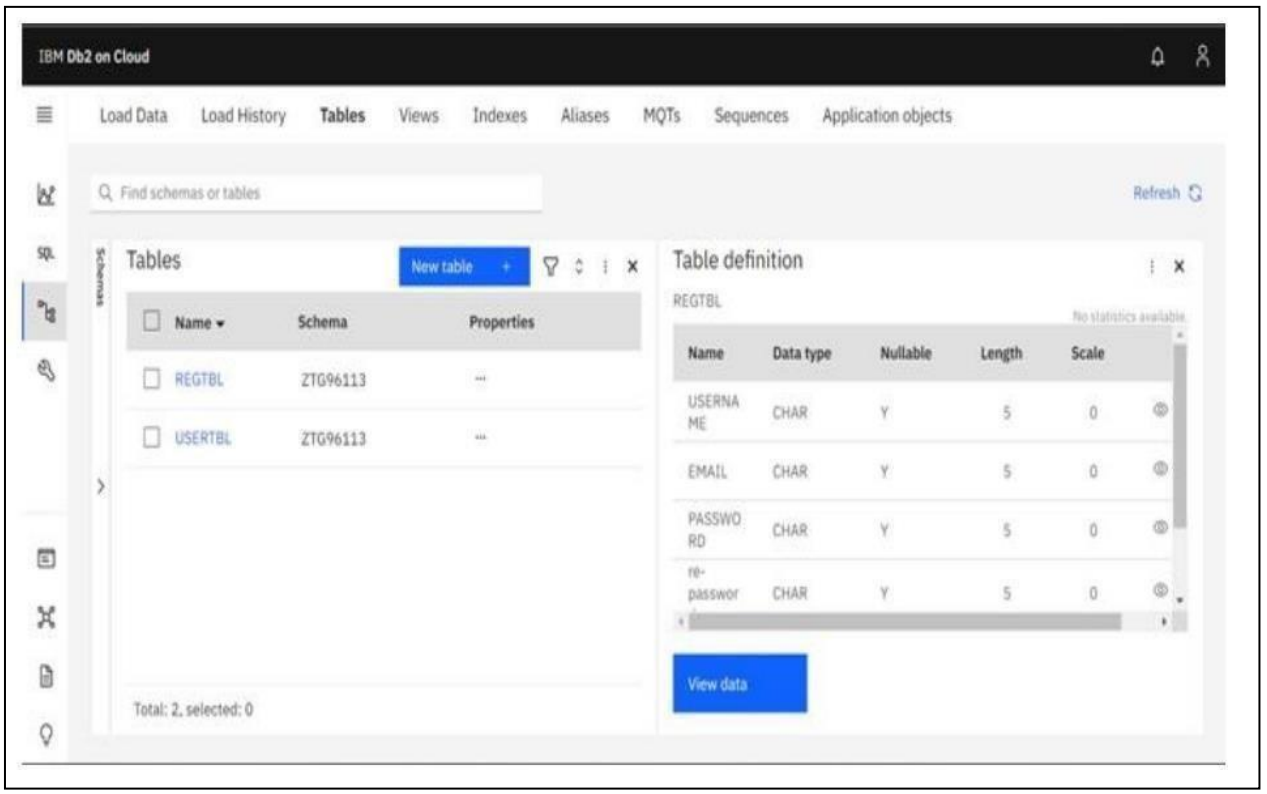
Plasma Donor Page

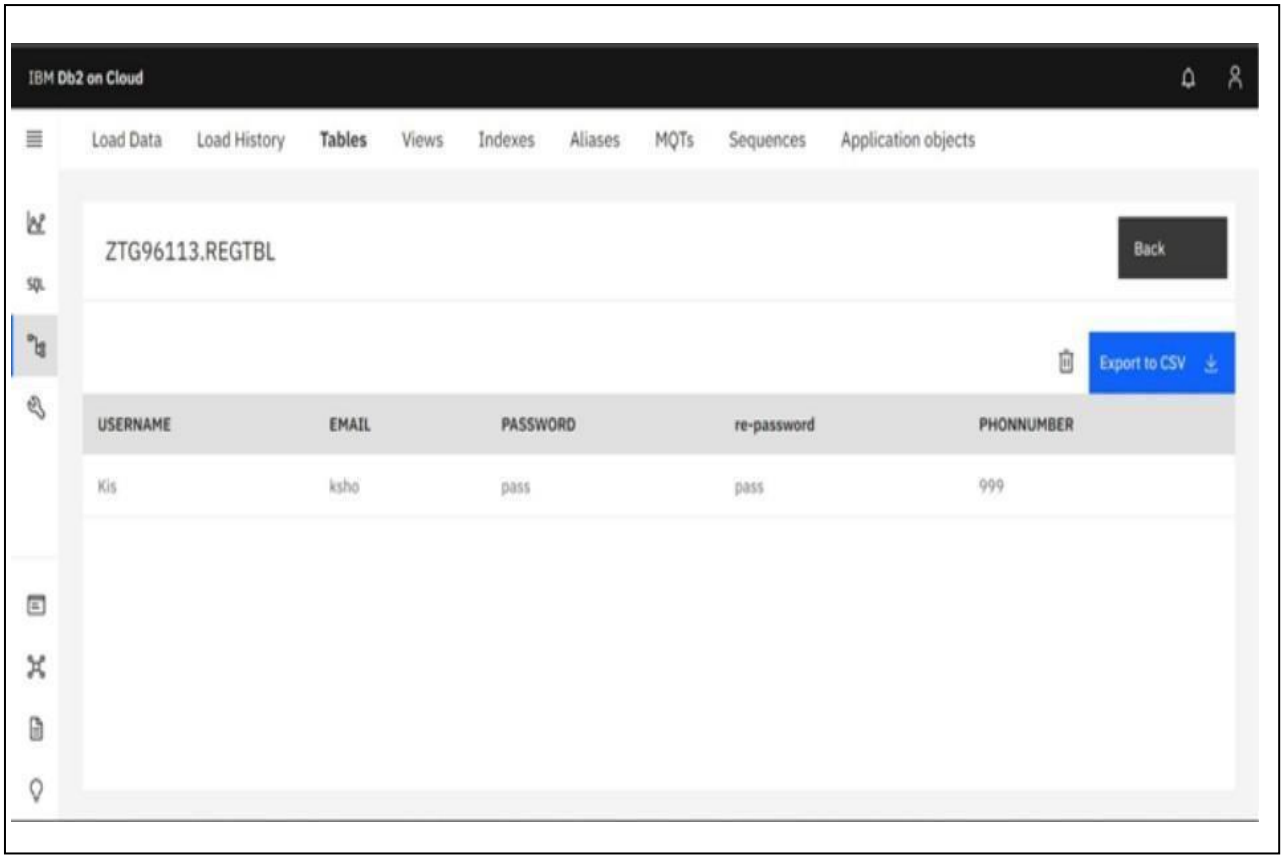


SEND GRID



IBM DB2


IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Tables
New table
Schemas
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
View data


IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
ZTG96113.REGTBL
Back
Export to CSV
USERNAME
EMAIL
PASSWORD
re-password
PHONNUMBER
Kis
ksho
pass
pass
999

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

form.py

```
class DonorUserForm(forms.ModelForm):

    class Meta:
        model=User
        fields=['first_name','last_name','username','password']
        widgets = {
        'password': forms.PasswordInput()
        }

class DonorForm(forms.ModelForm):
    class Meta:
        model=models.Donor
        fields=['bloodgroup','address','mobile','profile_pic']

class DonationForm(forms.ModelForm):
    class Meta:
        model=models.BloodDonate
        fields=['age','bloodgroup','disease','unit']
```

model.py

```
class Donor(models.Model):
    user=models.OneToOneField(User,on_delete=models.CASCADE)
                                                                profile_pic=
models.ImageField(upload_to='profile_pic/Donor/',null=True,blank=True)


    bloodgroup=models.CharField(max_length=10)


    address = models.CharField(max_length=40)
    mobile = models.CharField(max_length=20,null=False)

    @property
```

```
    def get_name(self):
        return self.user.first_name+" "+self.user.last_name
    @property
    def get_instance(self):
        return self
    def__str__(self):
        return self.user.first_name

class BloodDonate(models.Model):
    donor=models.ForeignKey(Donor,on_delete=models.CASCADE)
    disease=models.CharField(max_length=100,default="Nothing")
    age=models.PositiveIntegerField()
    bloodgroup=models.CharField(max_length=10)
    unit=models.PositiveIntegerField(default=0)
    status=models.CharField(max_length=20,default="Pending")
    date=models.DateField(auto_now=True)
    def__str__(self):
        return self.donor
```

view.py

```
def donor_signup_view(request):
    userForm=forms.DonorUserForm()
    donorForm=forms.DonorForm()
    mydict={'userForm':userForm,'donorForm':donorForm}
    if request.method=='POST':
        userForm=forms.DonorUserForm(request.POST)
        donorForm=forms.DonorForm(request.POST,request.FILES)
        if userForm.is_valid() and donorForm.is_valid():
            user=userForm.save()
            user.set_password(user.password)
            user.save()
            donor=donorForm.save(commit=False)
```

```
            donor.user=user
            donor.bloodgroup=donorForm.cleaned_data['bloodgroup']
            donor.save()
            my_donor_group = Group.objects.get_or_create(name='DONOR')
            my_donor_group[0].user_set.add(user)
        return HttpResponseRedirect('donorlogin')
    return render(request,'donor/donorsignup.html',context=mydict)



def donor_dashboard_view(request):
    donor= models.Donor.objects.get(user_id=request.user.id)
    dict={
                                                    'requestpending':
bmodels.BloodRequest.objects.all().filter(request_by_donor=donor).filter(status
='Pending').count(),
                                                    'requestapproved':
bmodels.BloodRequest.objects.all().filter(request_by_donor=donor).filter(status
='Approved').count(),
                                                    'requestmade':
bmodels.BloodRequest.objects.all().filter(request_by_donor=donor).count(),
                                                    'requestrejected':
bmodels.BloodRequest.objects.all().filter(request_by_donor=donor).filter(status
='Rejected').count(),
    }
    return render(request,'donor/donor_dashboard.html',context=dict)



def donate_blood_view(request):
    donation_form=forms.DonationForm()
    if request.method=='POST':
```

```
        donation_form=forms.DonationForm(request.POST)
        if donation_form.is_valid():
            blood_donate=donation_form.save(commit=False)
            blood_donate.bloodgroup=donation_form.cleaned_data['bloodgroup']
            donor= models.Donor.objects.get(user_id=request.user.id)
            blood_donate.donor=donor
            blood_donate.save()
            return HttpResponseRedirect('donation-history')
                                                                    return
render(request,'donor/donate_blood.html',{'donation_form':donation_form})

def donation_history_view(request):
    donor= models.Donor.objects.get(user_id=request.user.id)
    donations=models.BloodDonate.objects.all().filter(donor=donor)
    return render(request,'donor/donation_history.html',{'donations':donations})

def make_request_view(request):
    request_form=bforms.RequestForm()
    if request.method=='POST':
        request_form=bforms.RequestForm(request.POST)
        if request_form.is_valid():
            blood_request=request_form.save(commit=False)
            blood_request.bloodgroup=request_form.cleaned_data['bloodgroup']
            donor= models.Donor.objects.get(user_id=request.user.id)
            blood_request.request_by_donor=donor
            blood_request.save()
            return HttpResponseRedirect('request-history')
```

```
                                                                    return
render(request,'donor/makerequest.html',{'request_form':request_form})


def request_history_view(request):
   donor= models.Donor.objects.get(user_id=request.user.id)
    blood_request=bmodels.BloodRequest.objects.all().filter(request_by_donor=
donor)
                                                                    return
render(request,'donor/request_history.html',{'blood_request':blood_request})
```

BLOOD

admin.html

```
{% extends 'blood/adminbase.html' %}
{% block content %}
{% load widget_tweaks %}
<style>
        .xyz{
     display: table;
     margin-right: auto;
     margin-left: auto;
        }
</style>
<br><br>
<div class="container">
<div class="row">
     <div class="col-sm-3">
```

```
  <div class="card bg-light">
    <div class="card-body">
        <div class="blood">
            <h2>A+ <i class="fas fa-tint"></i></h2>
        </div><br><br>
        <div>
            {{A1.unit}}
        </div>
    </div>
  </div>
</div>
<div class="col-sm-3">
    <div class="card bg-light">
        <div class="card-body">
            <div class="blood">
                <h2>B+ <i class="fas fa-tint"></i></h2>
            </div><br><br>
            <div>
              {{B1.unit}}
            </div>
        </div>
      </div>
</div>
<div class="col-sm-3">
```

```
        <div class="card bg-light">
          <div class="card-body">
            <div class="blood">
              <h2>O+ <i class="fas fa-tint"></i></h2>
            </div><br><br>
            <div>
              {{O1.unit}}
            </div>
          </div>
        </div>
      </div>
      <div class="col-sm-3">
        <div class="card bg-light">
          <div class="card-body">
            <div class="blood">
              <h2>AB+ <i class="fas fa-tint"></i></h2>
            </div><br><br>
            <div>
              {{AB1.unit}}
            </div>
          </div>
        </div>
      </div>
    </div>
```

```
<div class="row">
    <div class="col-sm-3">
      <div class="card bg-light">
        <div class="card-body">
          <div class="blood">
            <h2>A- <i class="fas fa-tint"></i></h2>
          </div><br><br>
          <div>
            {{A2.unit}}
          </div>
        </div>
      </div>
    </div>
    <div class="col-sm-3">
        <div class="card bg-light">
          <div class="card-body">
            <div class="blood">
              <h2>B- <i class="fas fa-tint"></i></h2>
            </div><br><br>
            <div>
              {{B2.unit}}
            </div>
          </div>
        </div>
```

```
</div>
<div class="col-sm-3">
    <div class="card bg-light">
        <div class="card-body">
            <div class="blood">
                <h2>O- <i class="fas fa-tint"></i></h2>
            </div><br><br>
            <div>
              {{O2.unit}}
            </div>
        </div>
    </div>
</div>
<div class="col-sm-3">
  <div class="card bg-light">
      <div class="card-body">
            <div class="blood">
                <h2>AB- <i class="fas fa-tint"></i></h2>
            </div><br><br>
            <div>
              {{AB2.unit}}
            </div>
        </div>
    </div>
```

```
        </div>
    </div>
<hr>
<br>
<h3 class="text-center">Update Blood Unit</h3><br>
<div class="xyz">
   <form class="form-inline" method="POST">
        {% csrf_token %}
        <div class="form-group mx-sm-3 mb-6">
<select name="bloodgroup" class="form-control">
            <option disabled="disabled" selected="selected">Choose Blood Group</option>
            <option>O+</option>
            <option>O-</option>
            <option>A+</option>
            <option>A-</option>
            <option>B+</option>
            <option>B-</option>
            <option>AB+</option>
            <option>AB-</option>
          </select>
         </div>
        <div class="form-group mx-sm-3 mb-6">
 <input type="number" class="form-control" name="unit" placeholder="Unit">
        </div>
```

```
        <button type="submit" class="btn btn-primary mb-2">Update</button>
      </form>
</div>
</div>
{% endblock content %}
```

Index.html

```
{% load static %}
<!DOCTYPE html>
<head>
    <style>
        .xyz{
          margin-bottom: 0px;
          background-image: url('{% static "image/homepage3.png" %}');
          background-size: cover;
          background-repeat: no-repeat;
        }
      </style>

</head>
<body>
  {% include "blood/navbar.html" %}
<br>
```

```
<section id="section-jumbotron" style="margin-bottom: 0px;" class="jumbotron jumbotron-fluid d-flex justify-content-center align-items-center xyz">

   <div class="container text-center">

      <br>

<br>

</div>

 </section>

<div class="jumbotron" style="margin-top: 0px;margin-bottom: 0px;">

    <p class="lead text-center"><h3 align = "center">“PNT2022TMID26437"</h3></p>

      <p align="center">SHRIYA R </p>

        <p align="center">SATHISH KUMAR R </p>

         <p align="center">SHRIRAM S K</p>

          <p align="center">SUJITHA A</p><br>

<p class="lead text-center" ><h5 align = "center"><b>“Saving a life won’t cost you anything. Go ahead and donate Plasma”</b></h5>

</p>

<p class="text-center">- Anonymous</p

</div>

 {% include "blood/footer.html" %}

</body>
```

13.2 GITHUB

https://github.com/IBM-EPBL/IBM-Project-29134-1660121370