[**IBM-Project-49854-1660881153**](https://github.com/IBM-EPBL/IBM-Project-49854-1660881153/upload/main)

**Plasma Donor Application**

Team ID : **PNT2022TMID23657**

* Team Leader : M.Pooja
* Team member :V.Nehaa
* Team member :G.Suvetha
* Team member :S.Srinithi

**Status** :

* Ideation Phase - done
* Problem Statement - done
* Project Design & Planning (1) - done
* Project Design & Planning (2) - done
* Project planning - done
* Sprint 1 - done
* Sprint 2 - done
* Sprint 3 - done

● Sprint 4 - done

# INTRODUCTION

## 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.

## 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 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

## 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

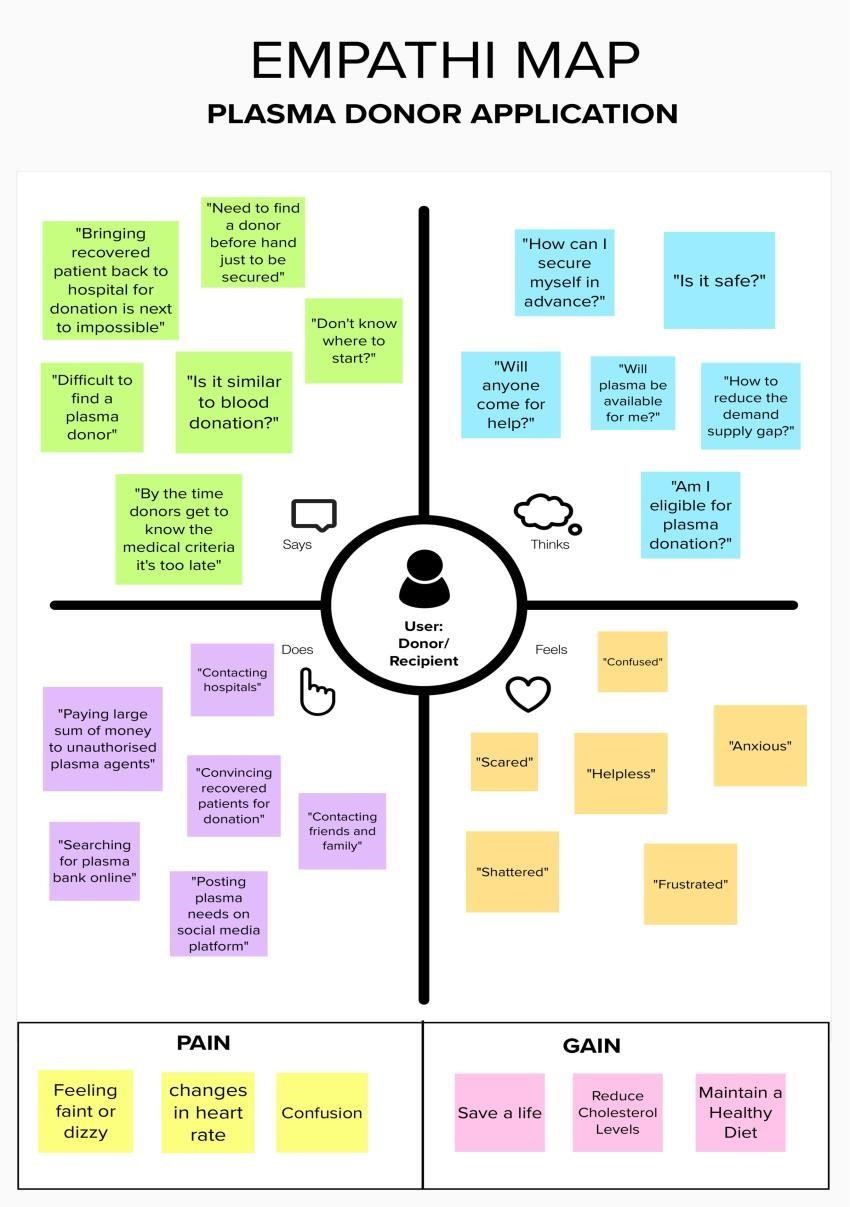
During COVID 19 crisis the requirement for plasma increased drastically as there were no vaccinations found in order to treat the infected patients.

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.

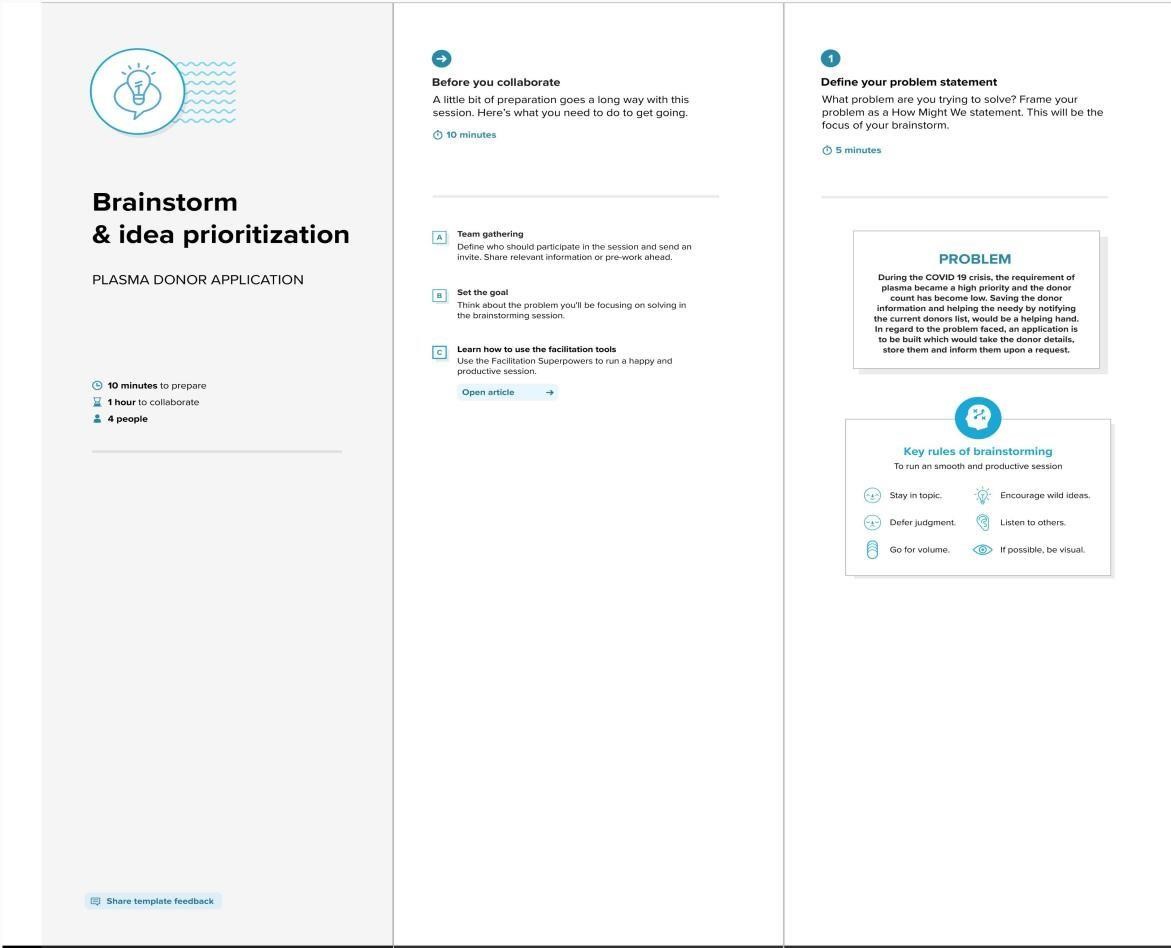
**IDEATION AND PROPOSED SYSTEM**

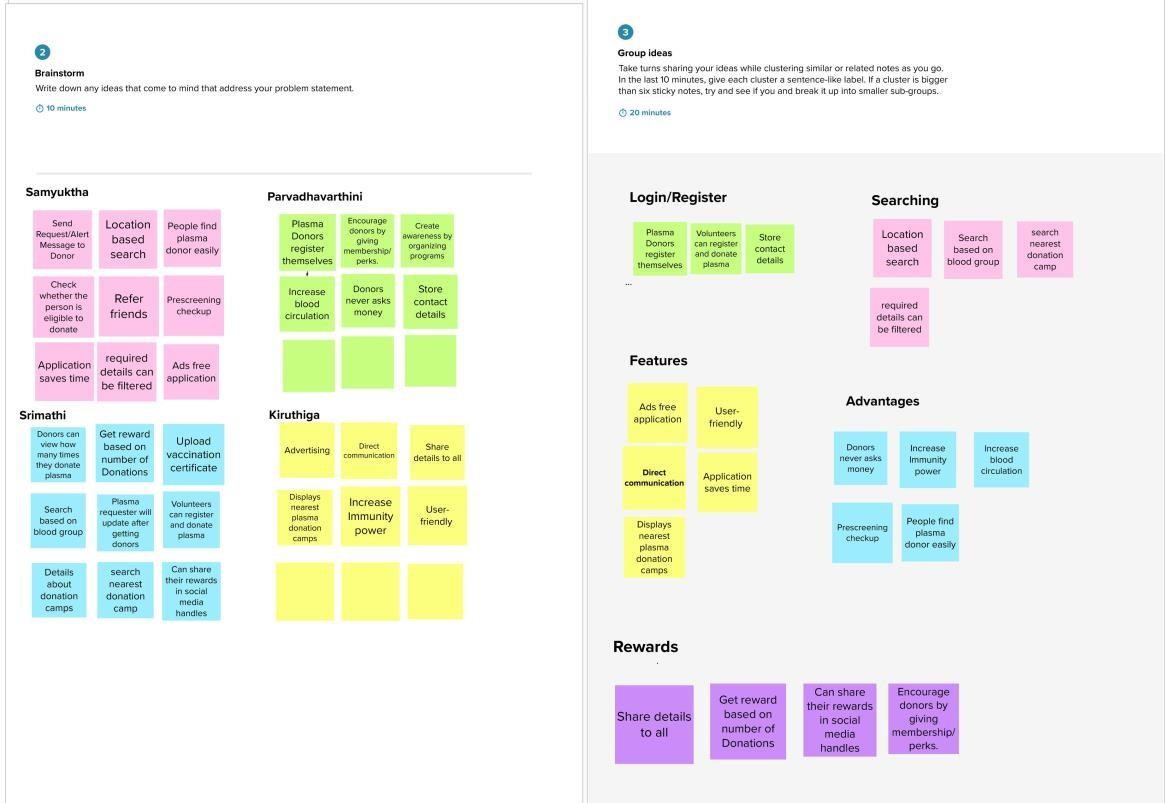
**EMPATHY MAP**



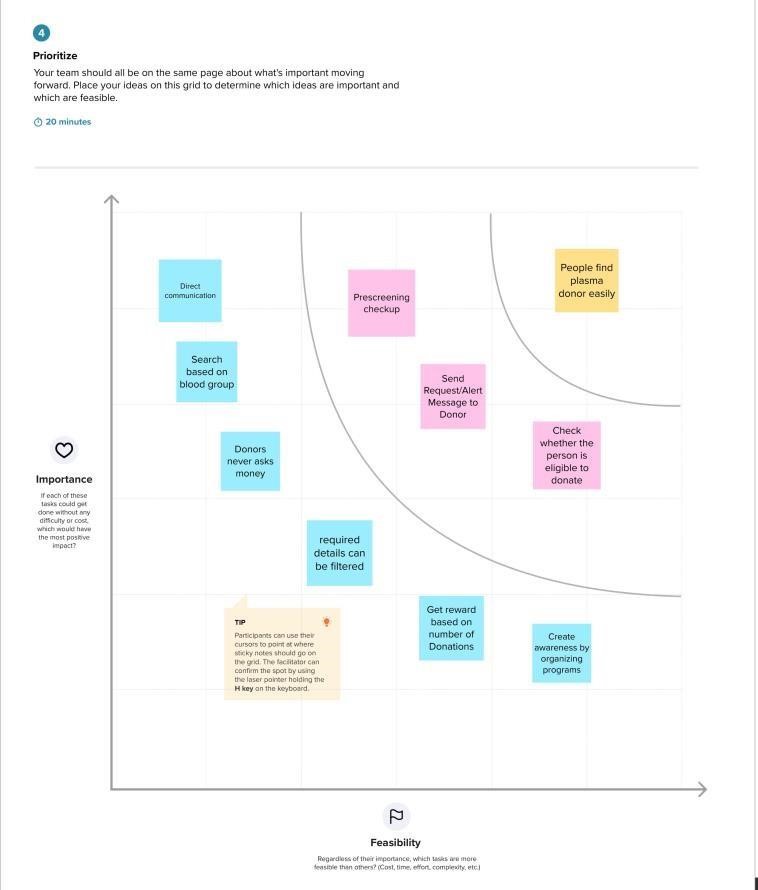
**IDEATION AND BRAINSTORM**

Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping

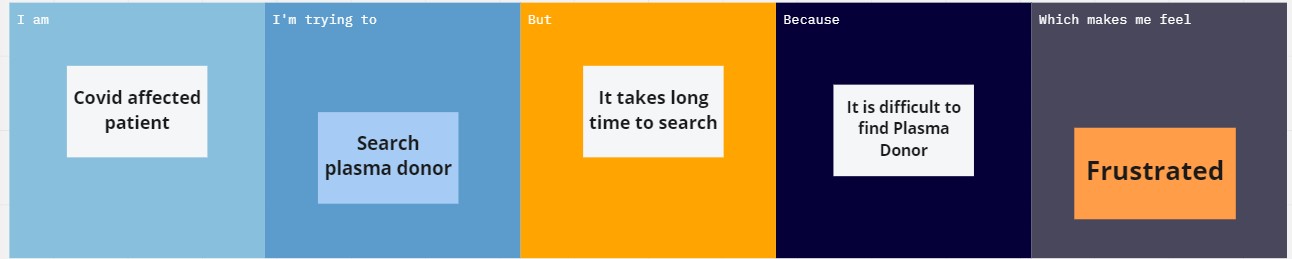
Step-3: Idea Prioritization



**PROBLEM STATEMENTS**

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. In regard to the problem faced, an application is to be built which would take the donor details, store them and inform them upon a request.

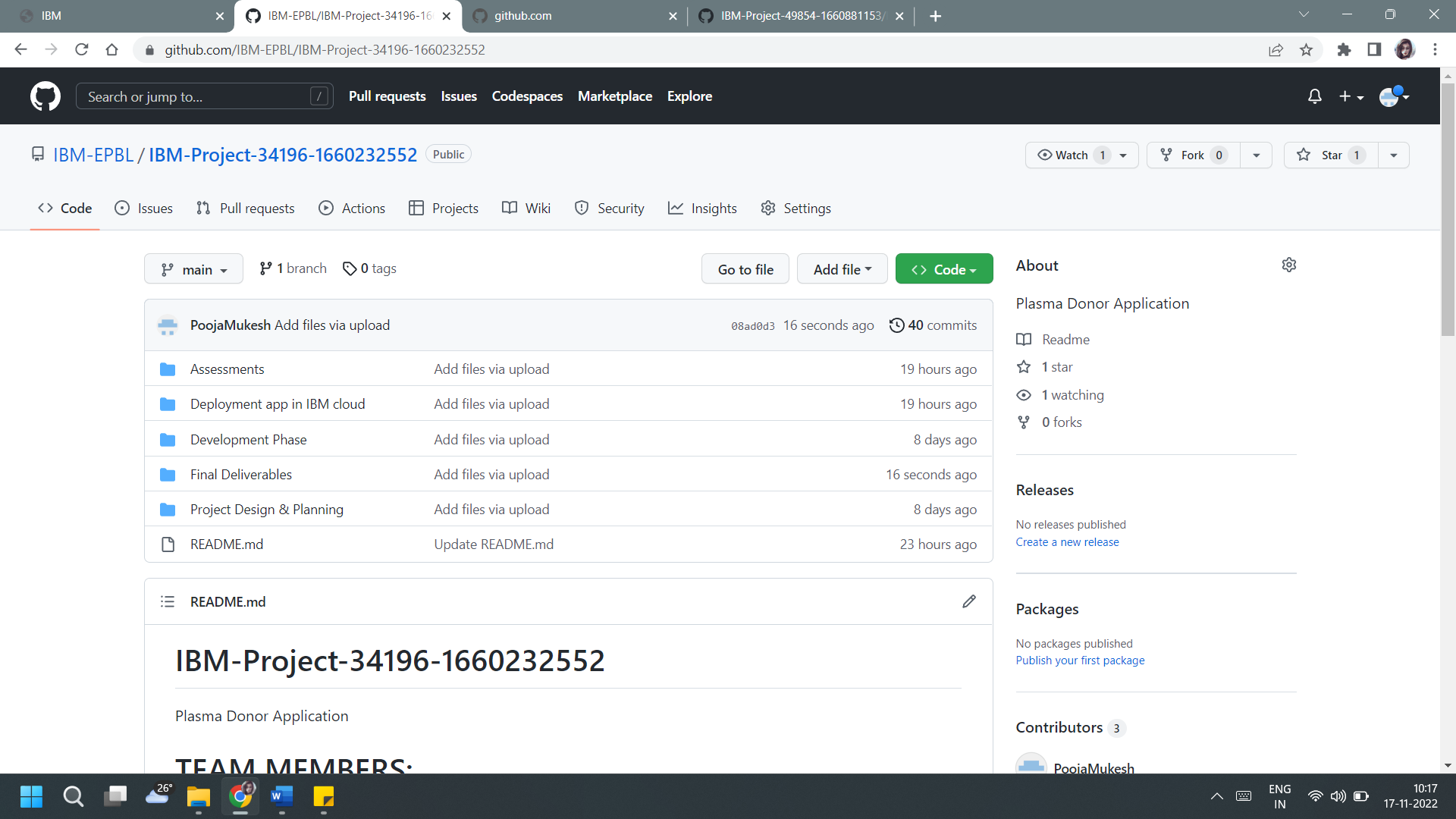
Lydia is thirty-five years old. She was affected by COVID-19 and had been admitted in the hospital. Due to COVID, her immunity power became low. She immediately wants a plasma donor. Here, she checks the availability of the plasma donor through our plasma donor application. She checks for the matching criteria of the donor through our application and also verifies whether the donor is within the range(location). Then after finding a perfect donor, he/she donates the plasma. Thus, our application is useful for a COVID patient to improve the immunity and blood circulation.

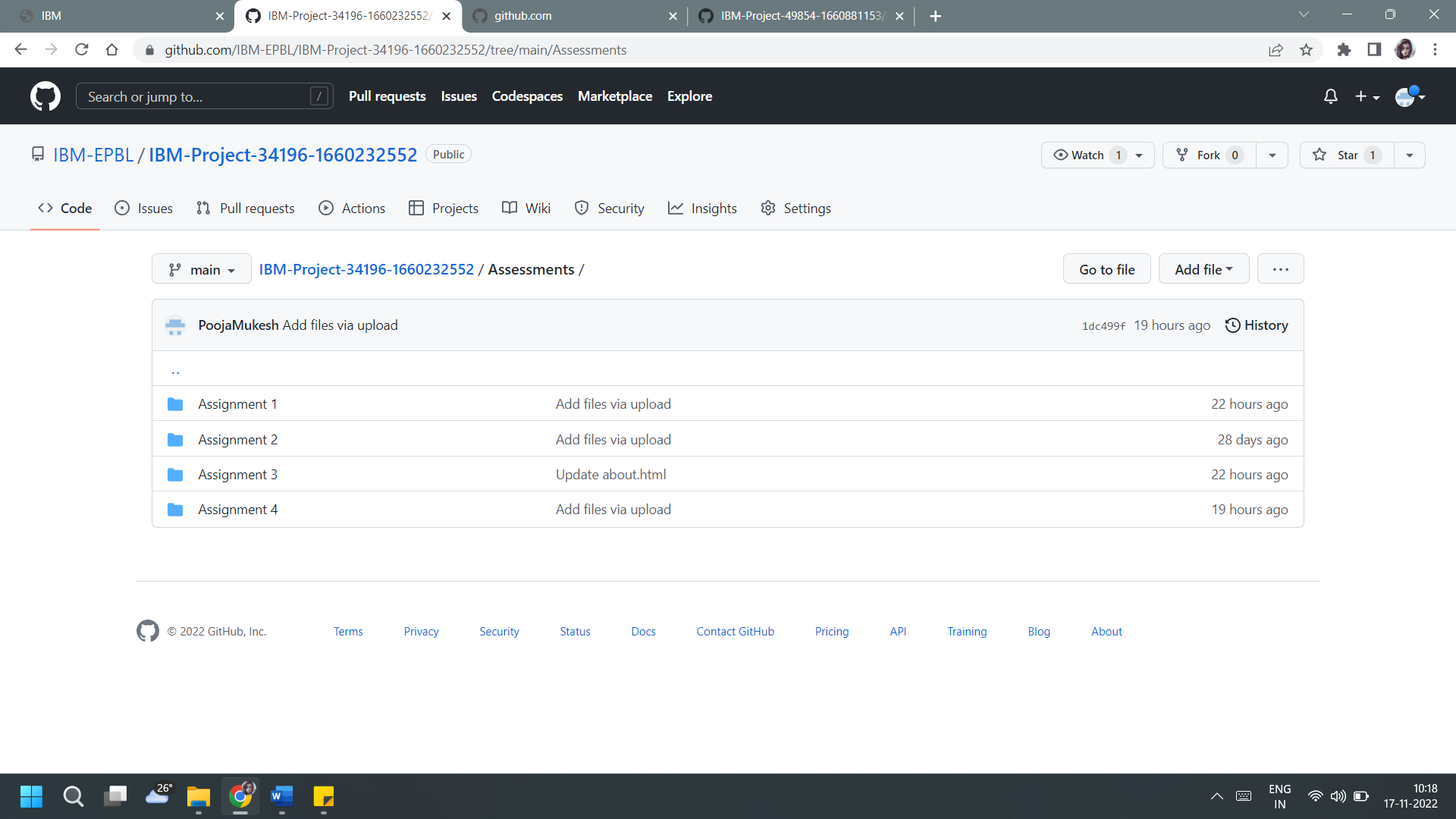


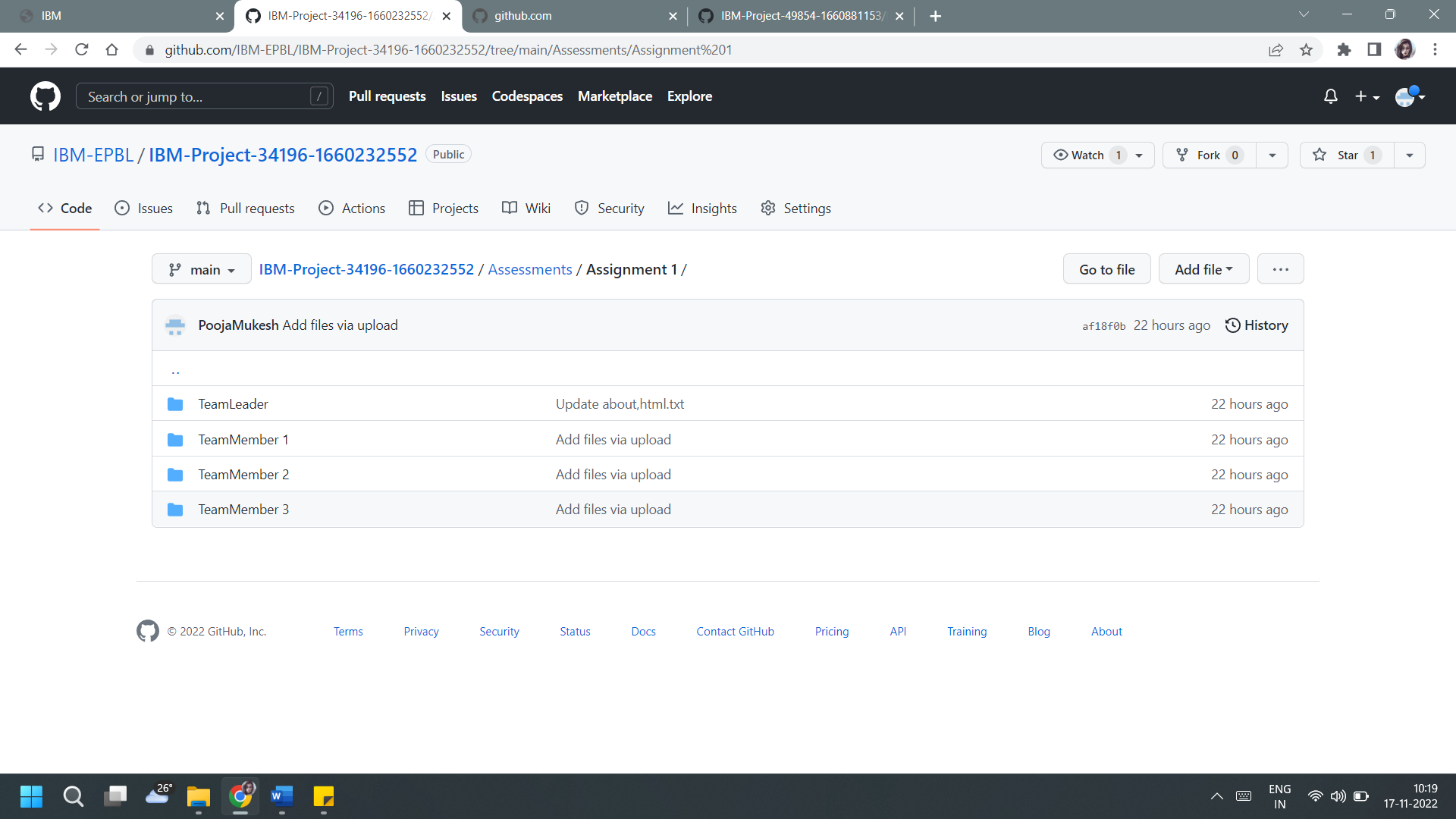
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problem**  **Statement (PS)** | **I am**  **(Customer)** | **I’m trying to** | **But** | **Because** | **Which makes me feel** |
| PS-1 | Covid affected patient | Search  Plasma donor | It takes long time to search | It is difficult  to find plasma donor | Frustrated |
| PS-2 | Volunteer | Donate plasma | I don’t know to  start | It is difficult to find who need plasma(who needs donation) | Sad |

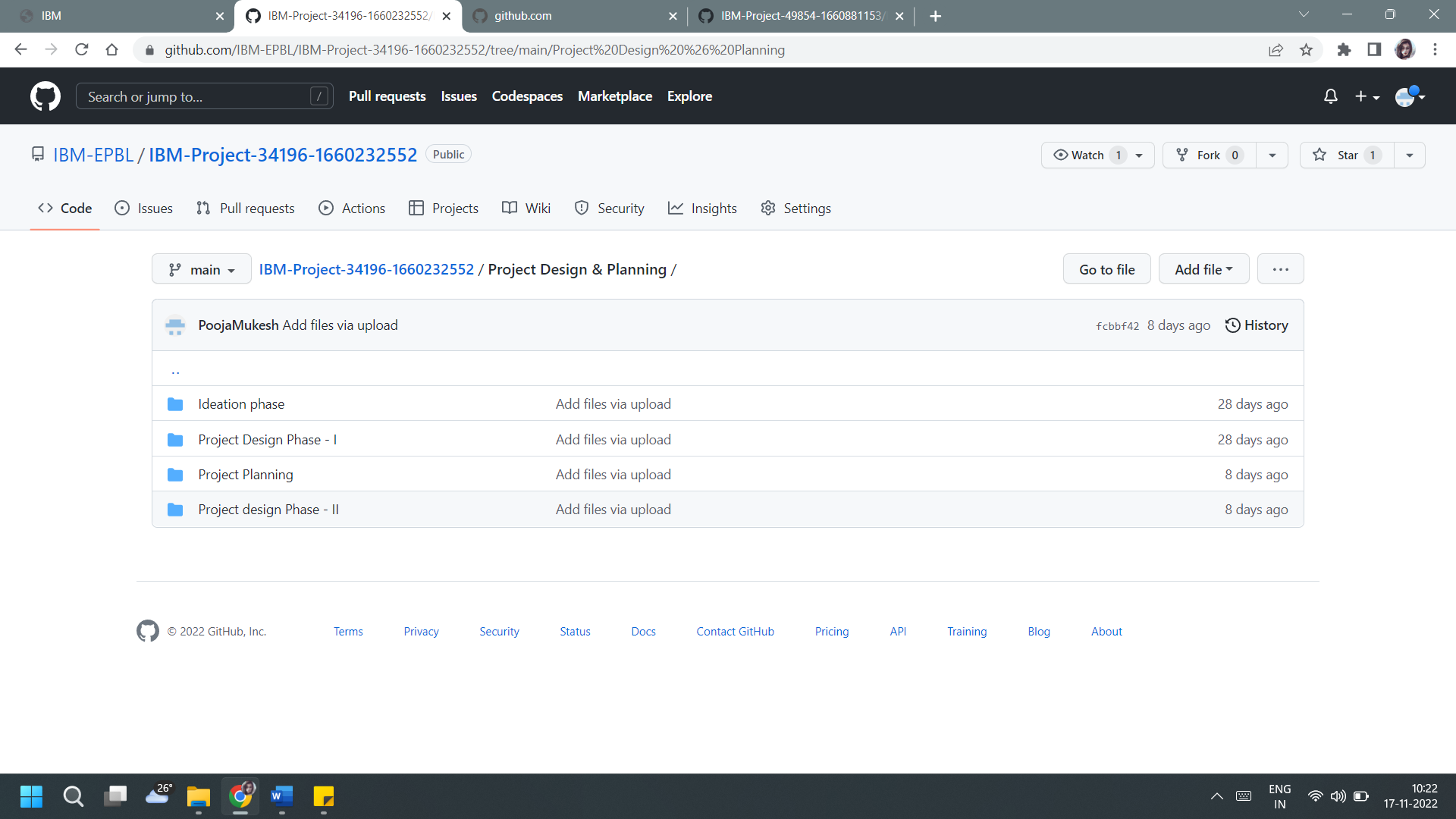
**Project completed**

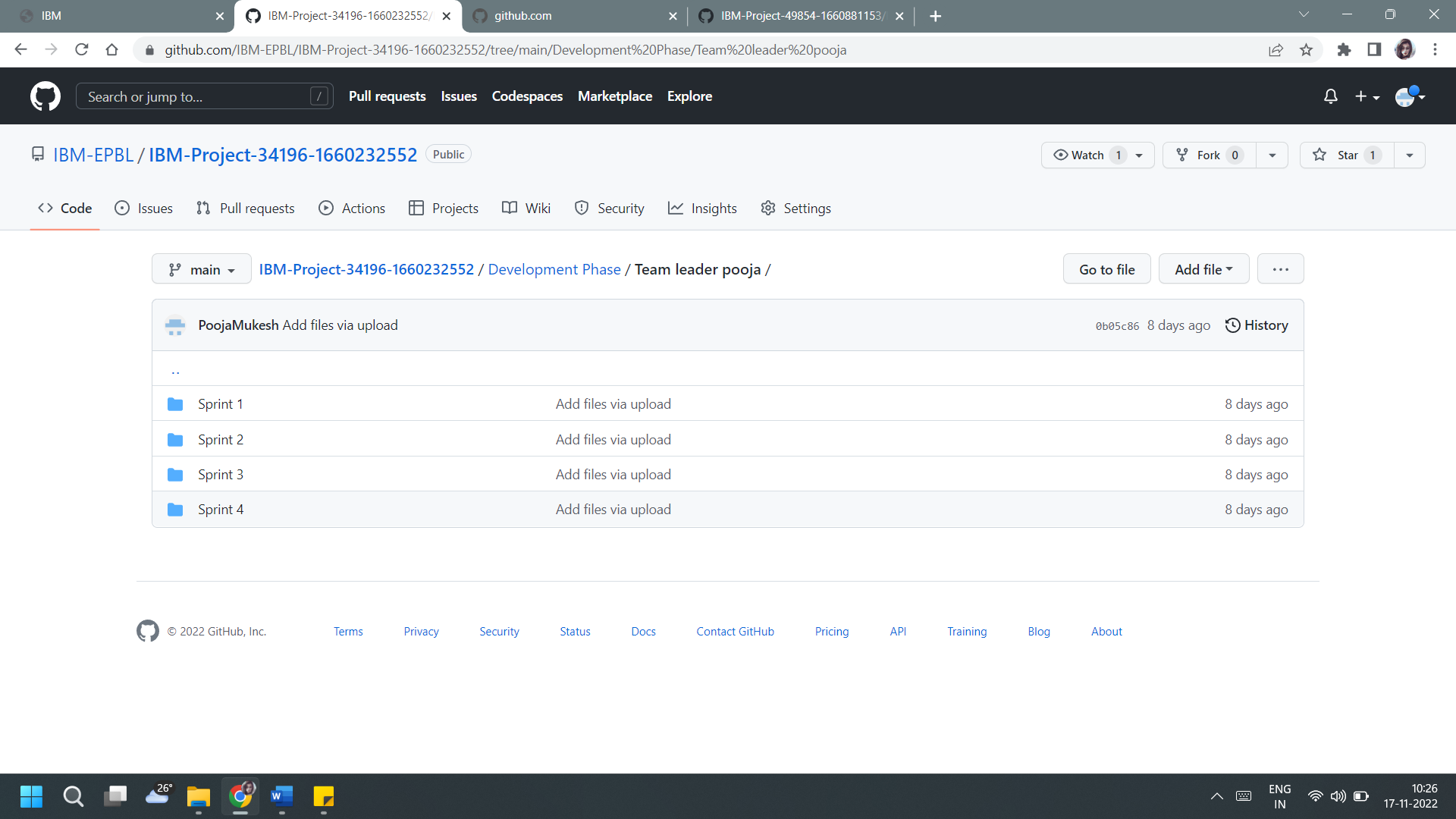
GITHUB Files

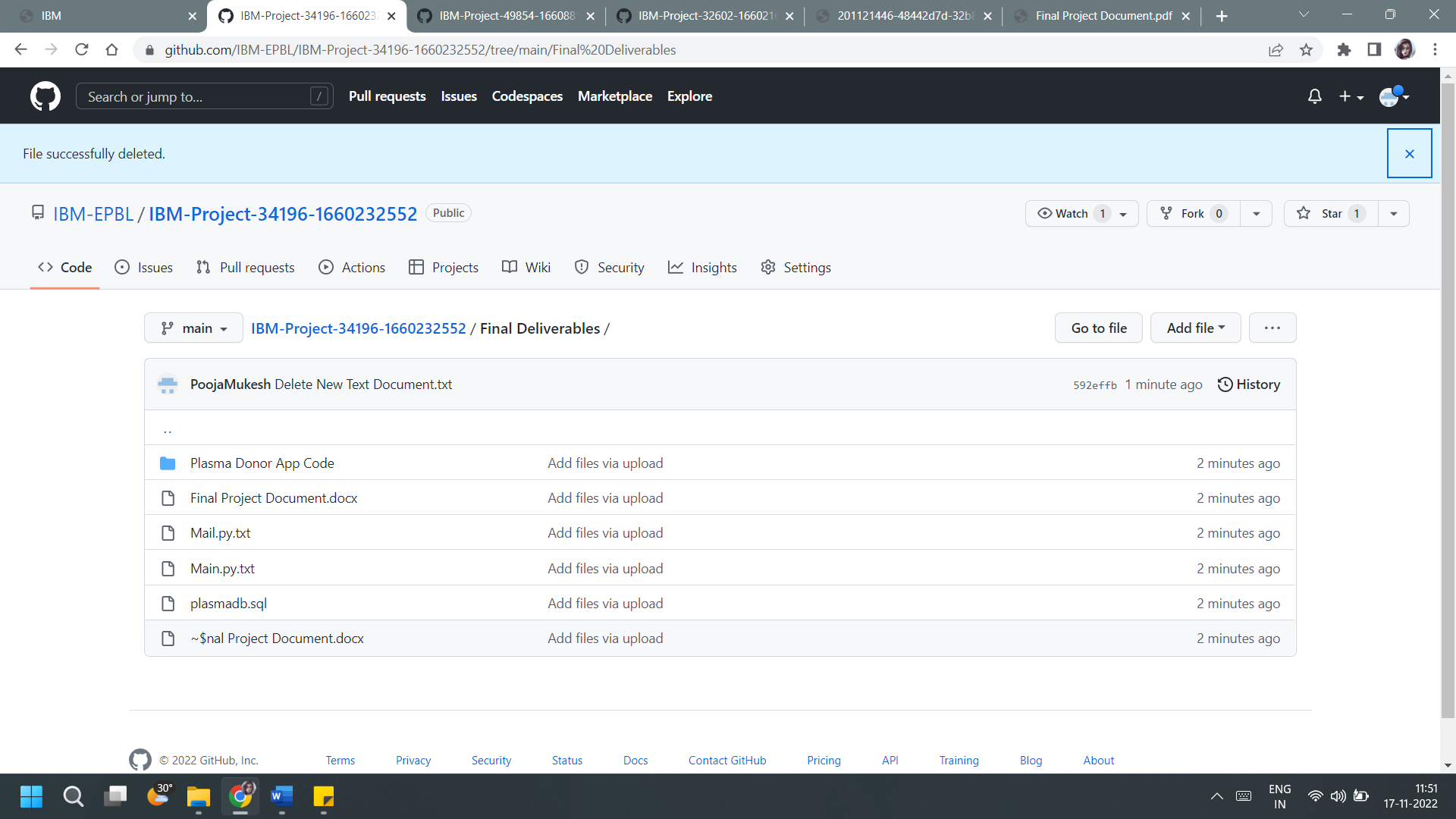


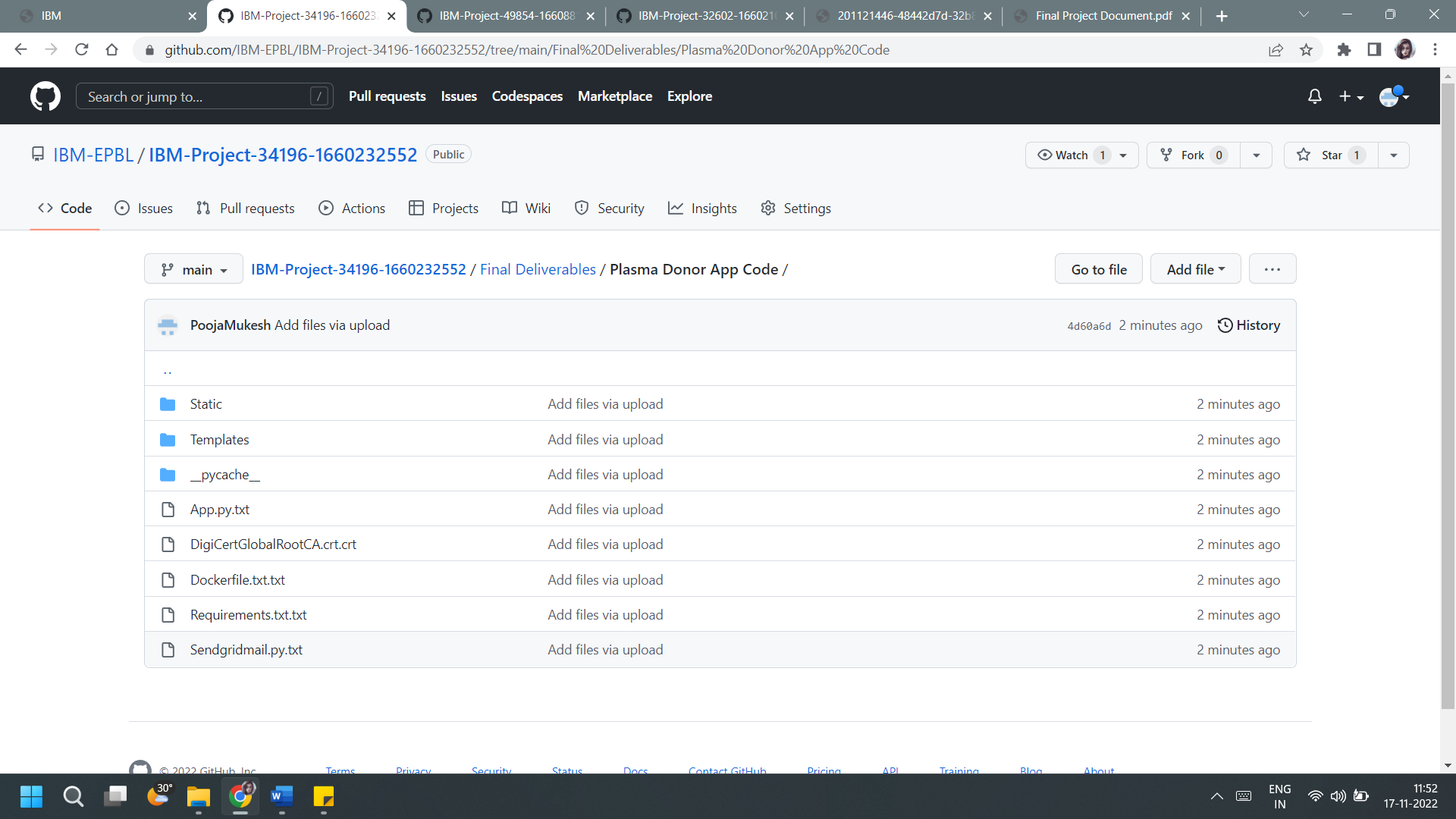












1. **CODING & SOLUTIONING**

**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,](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.

**7.2 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 [Micro framework](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

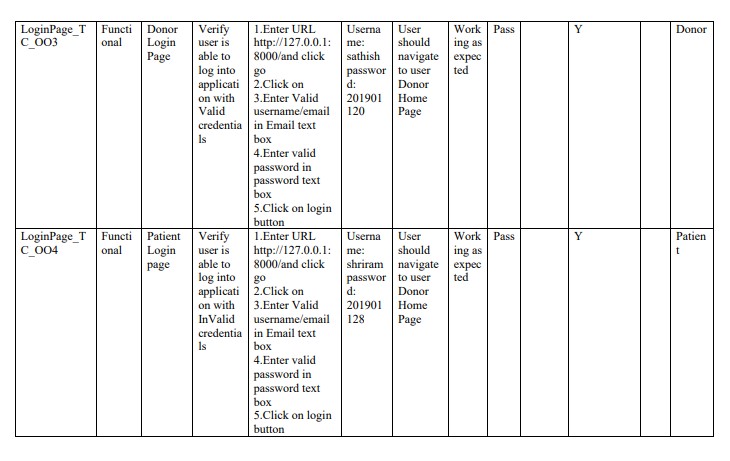
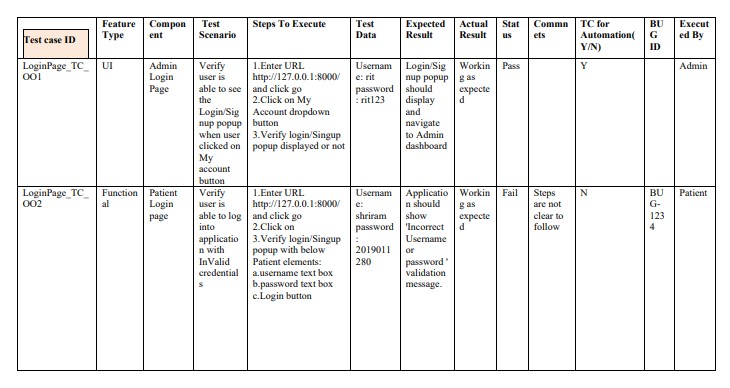
# Kubernetes

* **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.

**TESTING**

## 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



**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 | 7  1 |

### 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 |

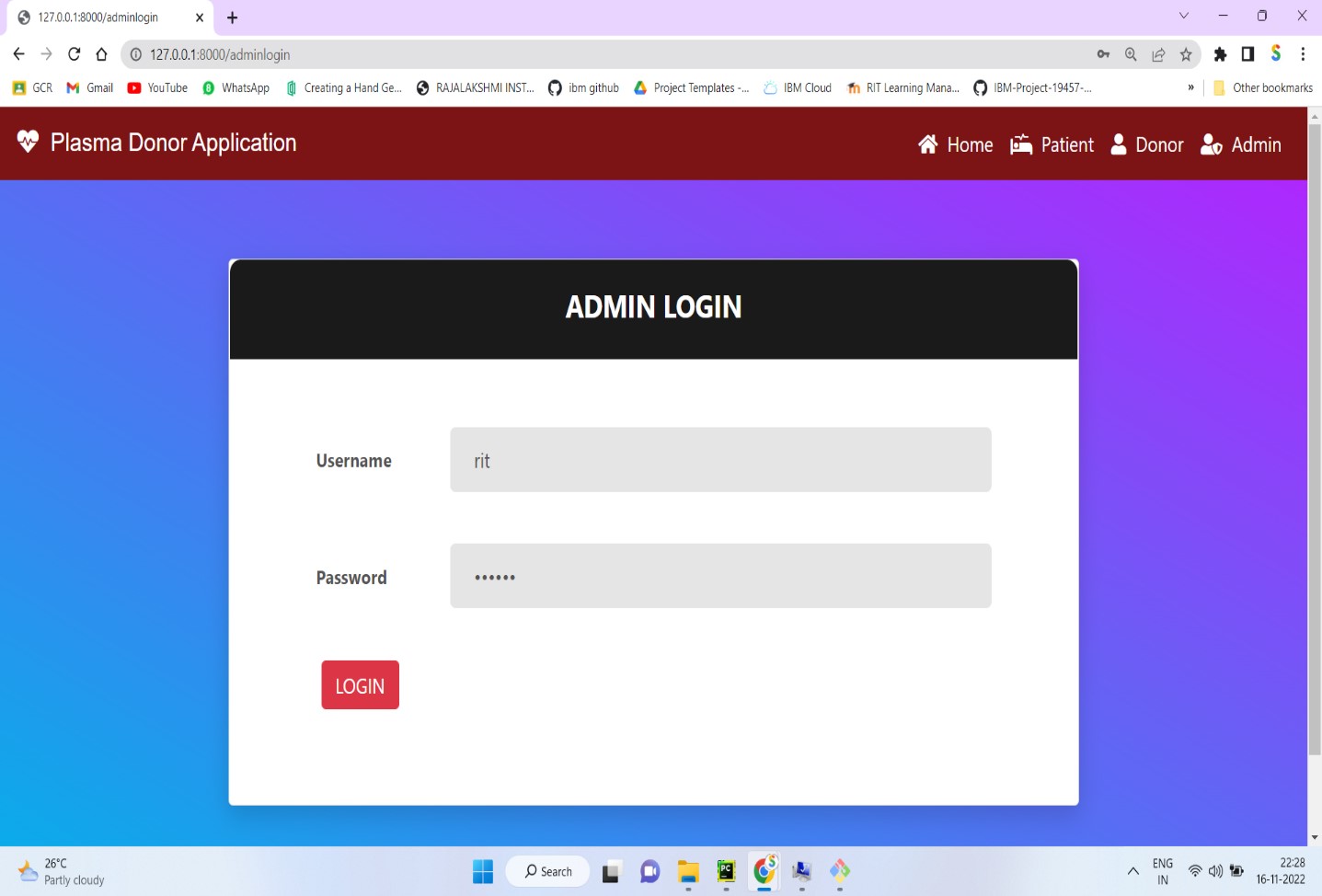
# 9. RESULTS

## 9.1 Performance Metrics

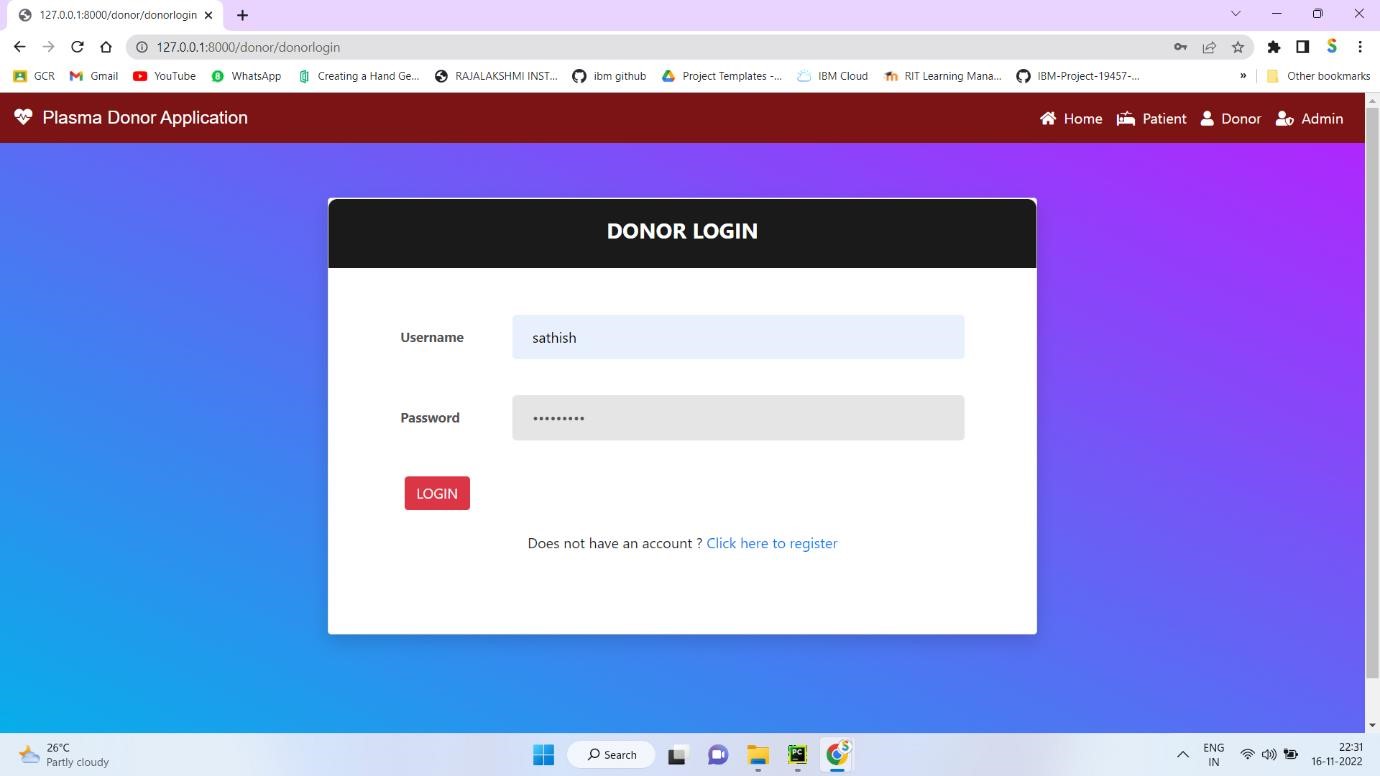
* Project metrics are used to track the progress and performance of a project.
* Monitoring parts of a project like productivity, scheduling, and scope make it easier for team leaders to see what's on track.
* As a project evolves, managers need access to changing
* deadlines or budgets to meet their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

# OUTPUT SCREENS

**Login page**

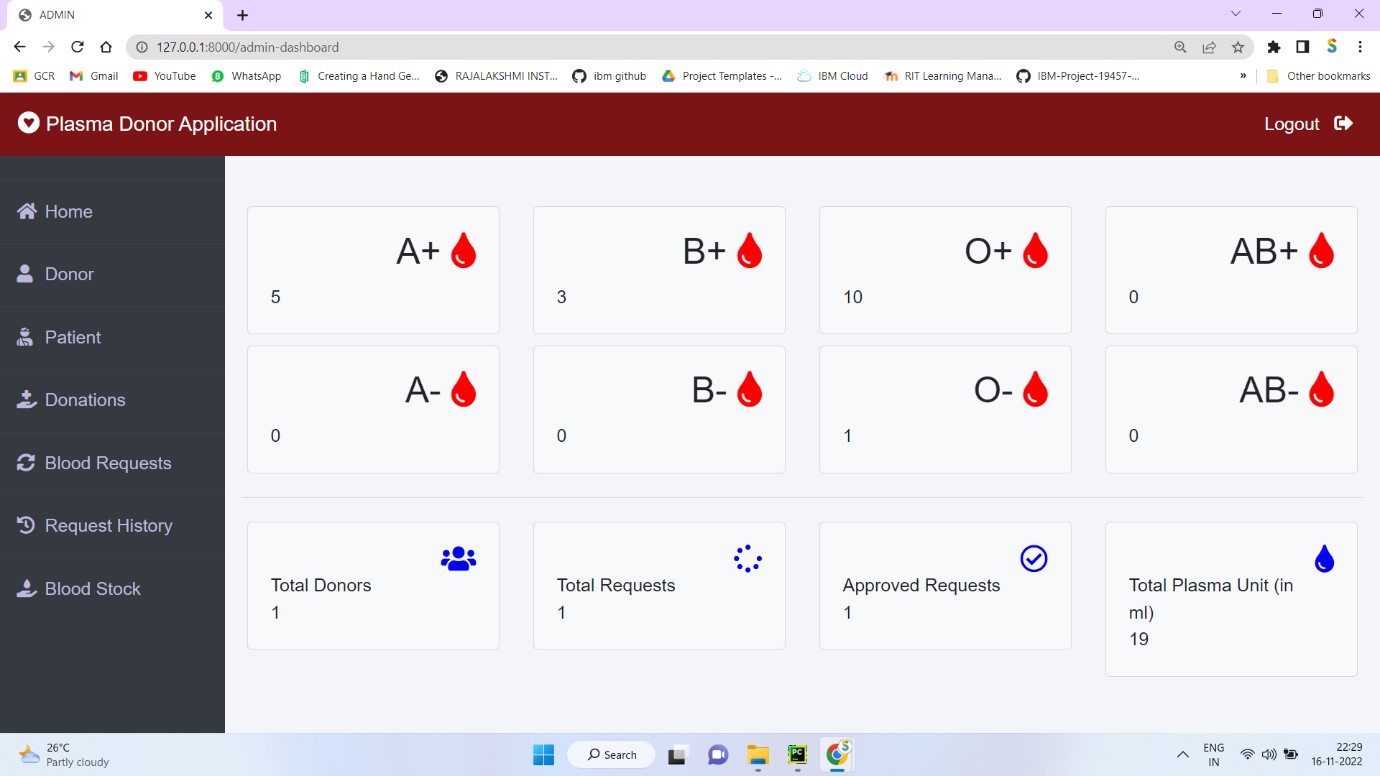


**Donor Login**

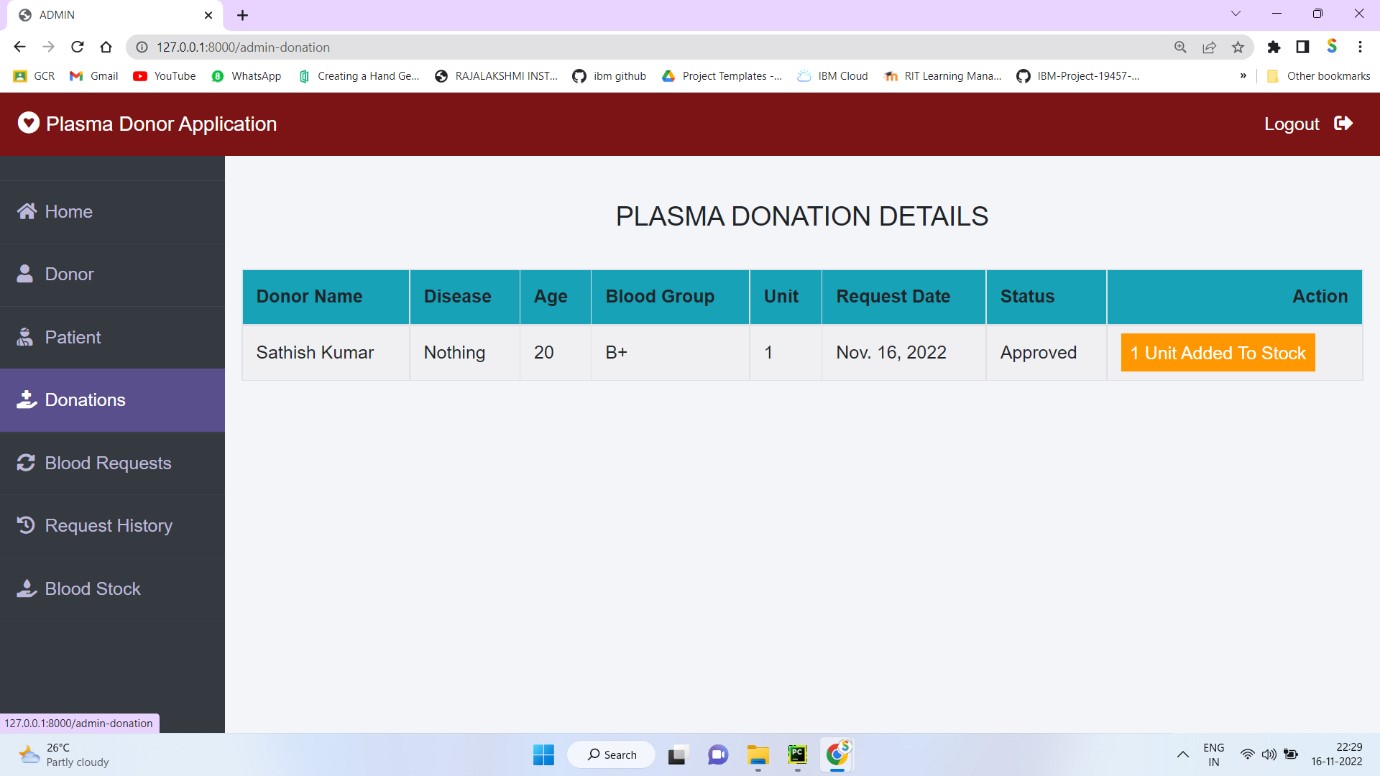


**Admin**

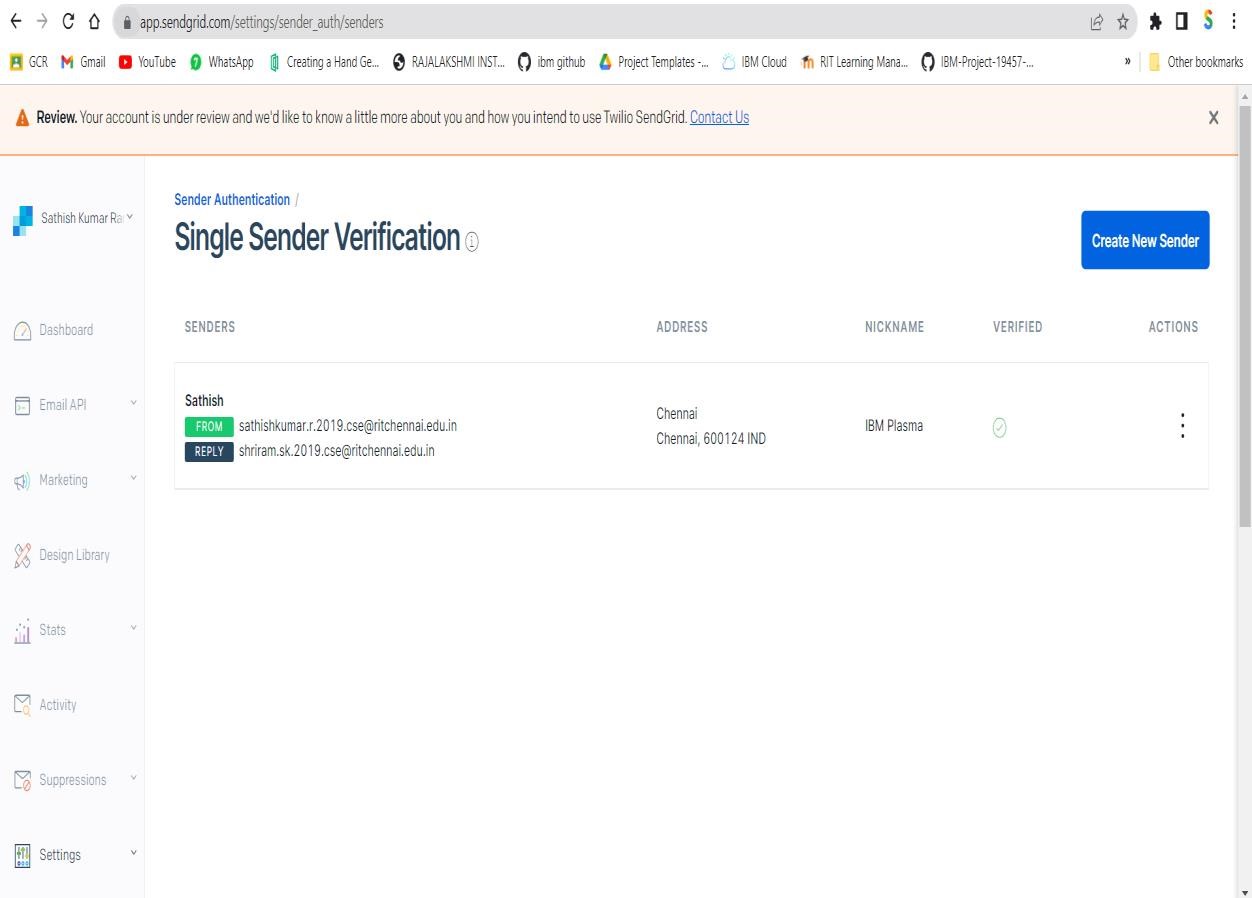
**Dashboard Page**



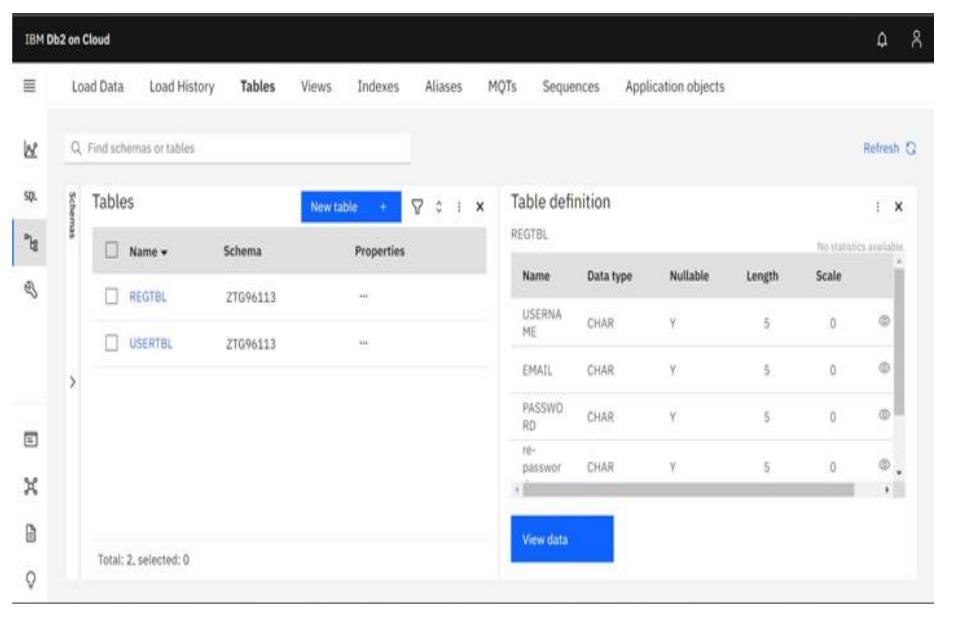
## Plasma Donor Page



**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

## -0 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)](http://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)](http://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)](http://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)](http://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)](http://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-itemscenter 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>

**Source code :** [**https://github.com/IBM-EPBL/IBM-Project-34196-1660232552**](https://github.com/IBM-EPBL/IBM-Project-34196-1660232552%20)