Nalaiya Thiran

Batch No: B2 – 2M4E

PSG Institute of Technology and Applied Research

Department of Computer Science and Engineering

Plasma Donor Application

Team ID: PNT2022TMID43307

Team Members:

715519104020	Keshav Adithya S P
715519104026	M H N S Sriram Raju
715519104028	Naveenkumar S
715519104004	Abubakar Siddick K

Project Guide:

Industry Mentor	Ms. Navya
Faculty mentor	Ms. P. Priya Ponnusamy

ABSTRACT

During COVID-19, the requirement for plasma became high and finding a perfect donor became very difficult for the patients who are in need. Plasma donation is one of the scientific ways in which we can help reduce mortality or help people affected by COVID19 from recovered patients. In the absence of an approved antiviral treatment plan for a fatal COVID19 infection, plasma therapy is an experimental approach to treat COVID19-positive patients and help them recover faster. In the recommendation system, the donor who wants to donate plasma can donate by uploading their COVID-19 certificate and the blood bank can see the donors who have uploaded the certificate and can make a request to the donor and the hospital can register/log in and search for the necessary things. Plasma is from a blood bank and they can request a blood bank and obtain plasma from the blood bank.

INDEX

CHAPTER 1: INTRODUCTION

1.1 Project Overview

The main goal of our project is to design a user-friendly web application that help those affected by COVID19 by donating plasma from patients who have recovered without approved antiretroviral therapy planning for deadly COVID19 infection, plasma therapy is an experimental approach to treat those COVID-positive patients and help them recover faster. Therapy is considered reliable and safe. If a particular person has fully recovered from COVID19, they are eligible to donate their plasma.

As we all know, the traditional methods of finding plasma, one must find out for oneself by looking at hospital records and contacting donors have been recovered, sometimes may not be available at home and move to other places. In this type of scenario, the health of those who are sick becomes worse. Therefore, it is not considered a rapid process to find plasma. The main purpose of the proposed system, the donor who wants to donate plasma can simply upload their covid19 traced certificate and can donate the plasma to the blood bank, the blood bank can apply for the donor and once the donor has accepted the request, the blood-bank can add the units they need and the hospital can also send the request to the blood bank that urgently needs the plasma for the patient and can take the plasma from the blood bank.

1.2 Purpose

Blood plasma is needed for many modern medical therapies. These include treatments for immune system conditions, bleeding, and respiratory disorders, as well as blood transfusions and wound healing. Plasma donation is necessary to collect enough plasma for medical treatments.

Our project aims at designing a user-friendly web application that aims at delivering plasma by identifying and validating both the recipient and the donor thereby saving lives of people in need of it. Plasma being a vital part in reducing mortality rate related to COVID-19, our project aims at facilitating the grounds to bridge the gap between such donor and recipients.

CHAPTER 2: LITERATURE SURVEY

2.1 Existing Problem

Several experiments have been carried out over the years by different groups of researchers. Some of the problems identified are as follows :

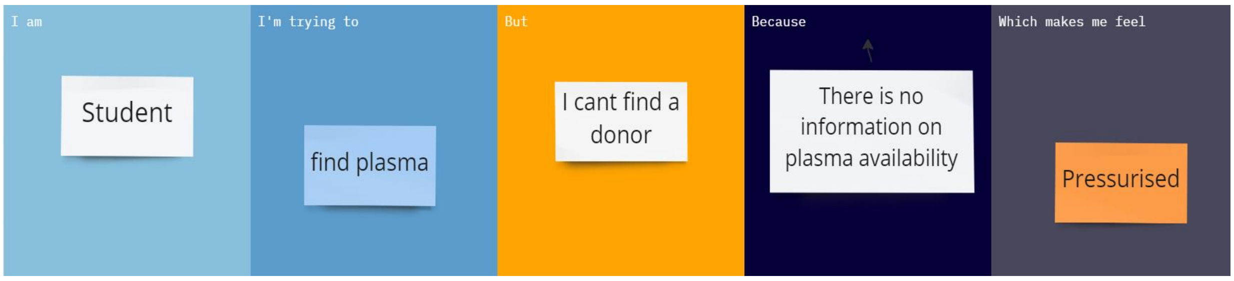
- There is currently no software in the plasma centre to keep any records.
- The plasma centre does not currently use any software to maintain records.
- In an emergency, it becomes challenging to promptly offer any record.
- Requiring extra manual labour to manage branch-related data.
- Keeping the accounts manually is a time-consuming and dangerous task, and maintaining such accounts in ledgers for an extended period of time is also highly challenging.
- The files are challenging to handle and keep up with.
- The possibility of file degradation if the data is kept in the files for a long time.
- Retrieving, storing, and updating the data all take time.
- It is challenging to maintain a record of the donor and recipient who have most recently given or received plasma.

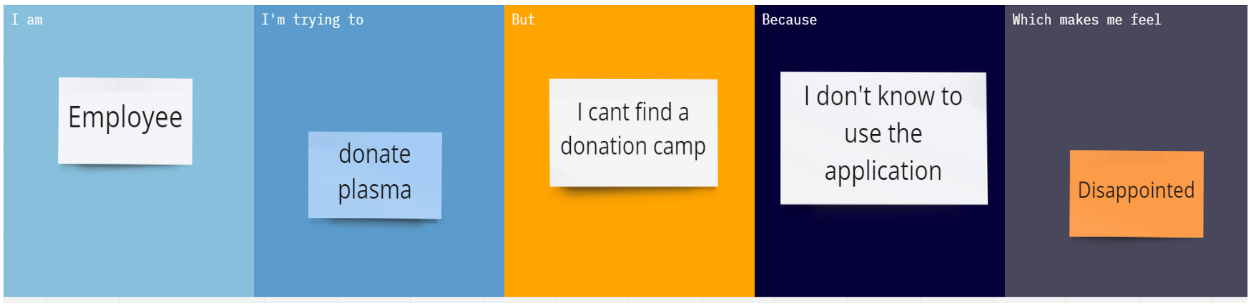
2.2 References

Paper Title	Author	Outcome
Treatment of 5 Critically Ill Patients With COVID-19 With Convalescent Plasma	Chenguang Shen, PhD; Zhaoqin Wang, PhD; Fang Zhao, PhD	In this preliminary uncontrolled case series of 5 critically ill patients with COVID-19 and ARDS, administration of convalescent plasma containing neutralizing antibody was followed by improvement in their clinical status. The limited sample size and study design preclude a definitive statement about the potential effectiveness of this treatment, and these observations require evaluation in clinical trials.
Instant Plasma Donor Recipient Connector Web Application	Ripathis S, Kumar V, Prabhakar A, Joshi S, Agarwal A	Microscale PassivePlasma Separation: A Review of Design Principles and Microdevices," J. Micro mech Micro 25 (8): 083001; Plasma separation is of great importance in the fields of diagnosis and healthcare. Due to the lagging transition to microscale, these recent trends are a rapid shift towards shrinking complex macro processes.

Fresh frozen plasma transfusion does not affect outcomes following hepatic resection for hepatocellular carcinoma	TakeakiIshizawa MD, PhD; KiyoshiHasegawa MD, PhD; Nelson HirokazuTsuno MD, PhD	PAPD was safe in patients with underlying liver disease and can be beneficial in simulating the liver synthetic function in advance of the operation. Autologous fresh frozen plasma transfusion was effective for avoiding allogeneic blood products in liver resection for hepatocellular carcinoma

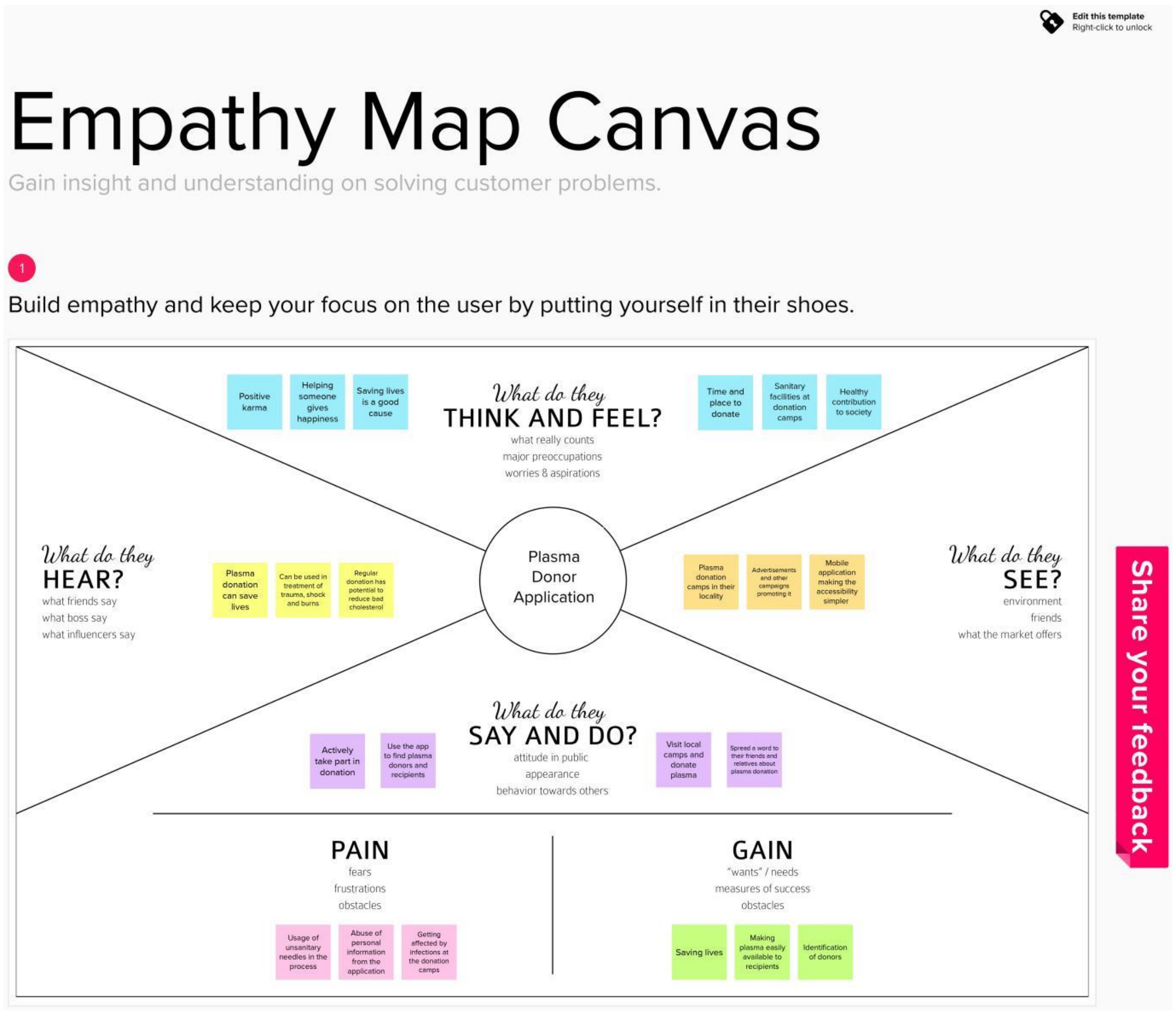
2.3 Problem Statement Definition





CHAPTER 3: IDEATION & PROPOSED SOLUTION

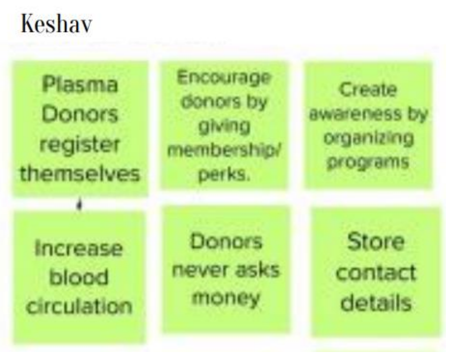
3.1 Empathy Map Canvas


Edit this template
Right-click to unlock
Empathy Map Canvas
Gain insight and understanding on solving customer problems.
1
Build empathy and keep your focus on the user by putting yourself in their shoes.
What do they THINK AND FEEL?
what really counts
major preoccupations
worries & aspirations
Positive karma
Helping someone gives happiness
Saving lives is a good cause
Time and place to donate
Sanitary facilities at donation camps
Healthy contribution to society
What do they HEAR?
what friends say
what boss say
what influencers say
Plasma donation can save lives
Can be used in treatment of trauma, shock and burns
Regular donation has potential to reduce bad cholesterol
Plasma Donor Application
What do they SEE?
environment
friends
what the market offers
Plasma donation camps in their locality
Advertisements and other campeigns promoting it
Mobile application making the accessibility simpler
What do they SAY AND DO?
attitude in public
appearance
behavior towards others
Actively take part in donation
Use the app to find plasma donors and recipients
Visit local camps and donate plasma
Spread a word to their friends and relatives about plasma donation
PAIN
fears
frustrations
obstacles
Usage of unsanitary needles in the process
Abuse of personal information from the application
Getting affected by infections at the donation camps
GAIN
"wants" / needs
measures of success
obstacles
Saving lives
Making plasma easily available to recipients
Identification of donors
Share your feedback

3.2 Ideation & Brainstorming

3.2.1 Brainstorm by team members

Keshav Adithya SP:



Keshav

Plasma Donors register themselves

Encourage donors by giving membership/ perks.

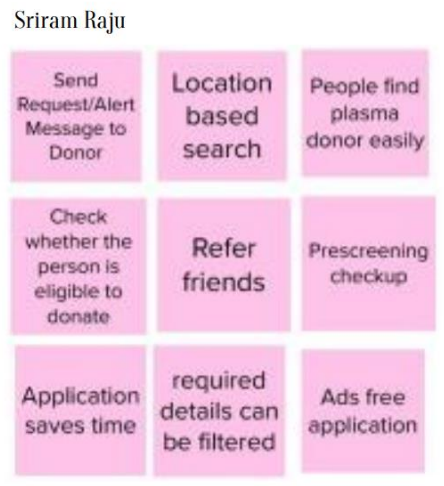
Create awareness by organizing programs

Increase blood circulation

Donors never asks money

Store contact details

M H N S Sriram Raju:



Sriram Raju

Send Request/Alert Message to Donor

Location based search

People find plasma donor easily

Check whether the person is eligible to donate

Refer friends

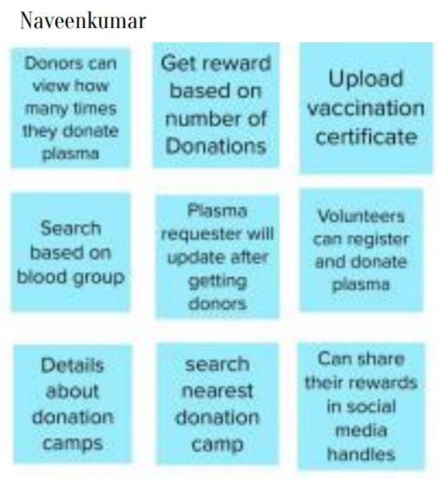
Prescreening checkup

Application saves time

required details can be filtered

Ads free application

Naveenkumar S:



Naveenkumar

Donors can view how many times they donate plasma

Get reward based on number of Donations

Upload vaccination certificate

Search based on blood group

Plasma requester will update after getting donors

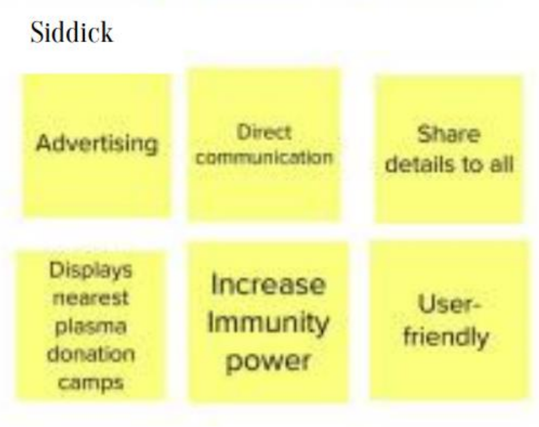
Volunteers can register and donate plasma

Details about donation camps

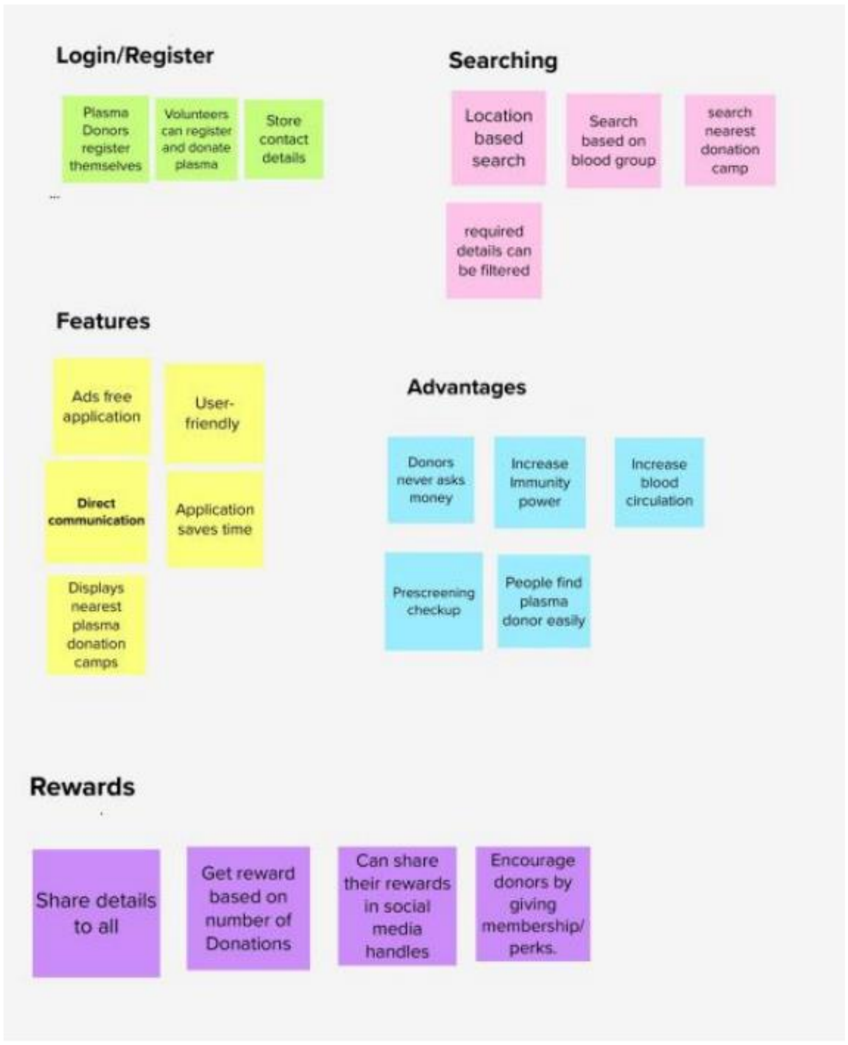
search nearest donation camp

Can share their rewards in social media handles

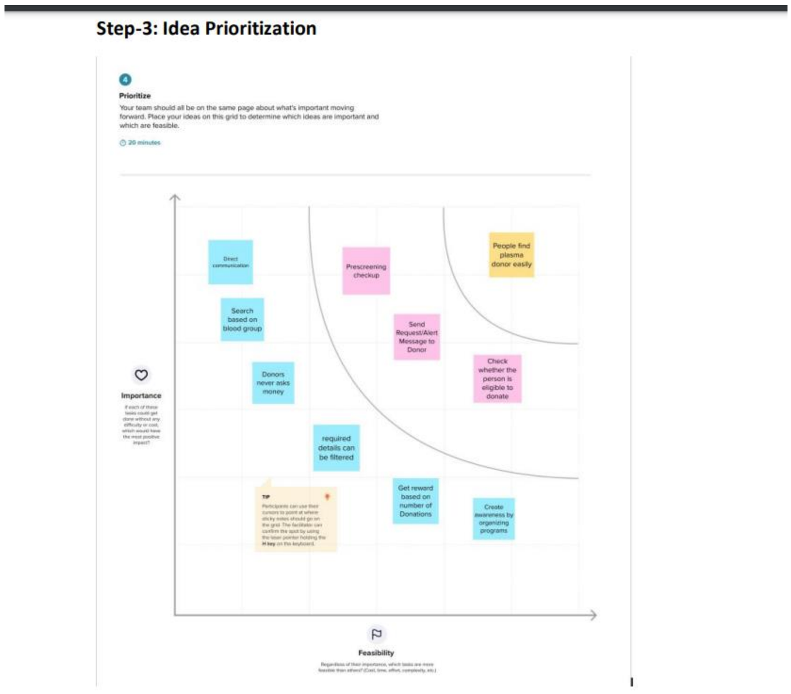
Abubakar Siddick K:



3.2.2 Group ideas


Login/Register
Plasma Donors register themselves
Volunteers can register and donate plasma
Store contact details
Searching
Location based search
Search based on blood group
search nearest donation camp
required details can be filtered
Features
Ads free application
User-friendly
Direct communication
Application saves time
Displays nearest plasma donation camps
Advantages
Donors never asks money
Increase Immunity power
Increase blood circulation
Prescreening checkup
People find plasma donor easily
Rewards
Share details to all
Get reward based on number of Donations
Can share their rewards in social media handles
Encourage donors by giving membership/ perks.

3.2.3 Prioritize


Step-3: Idea Prioritization

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

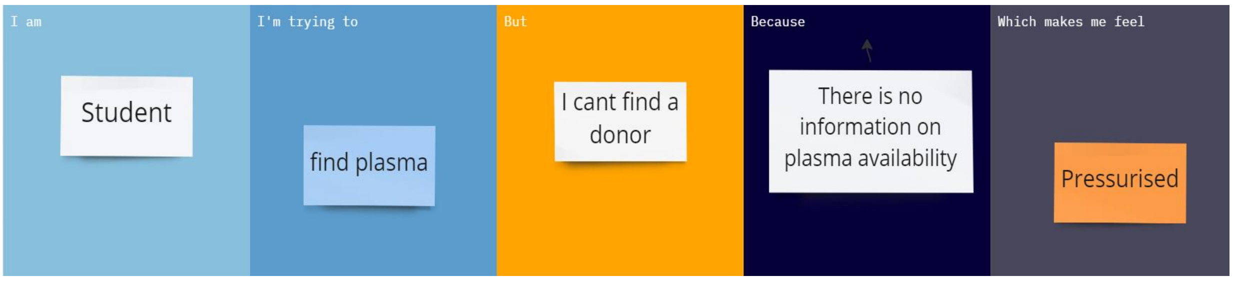
TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the H key on the keyboard.

Direct communication
Prescreening checkup
People find plasma donor easily
Search based on blood group
Send Request/Alert Message to Donor
Donors never asks money
Check whether the person is eligible to donate
Importance
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?
required details can be filtered
Get reward based on number of Donations
Create awareness by organizing programs
Feasibility
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

3.3 Proposed Solution

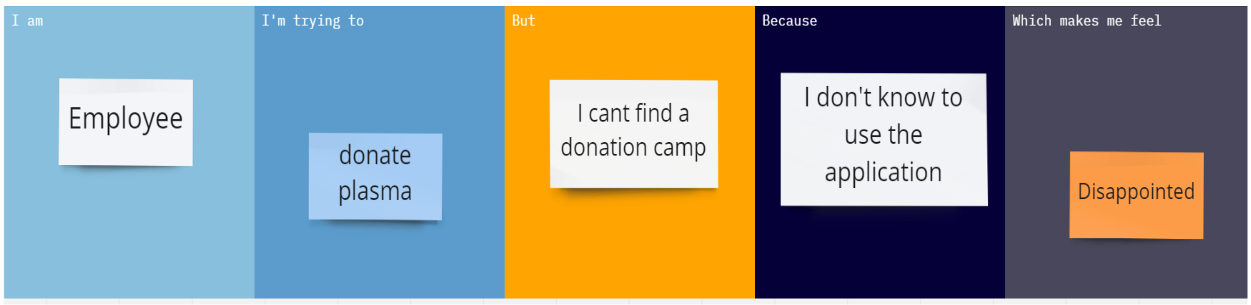
Problem statement 1



S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	I'm a student looking for plasma since I need it in an emergency, but I have no idea if the unit I need is even accessible, which depresses me.

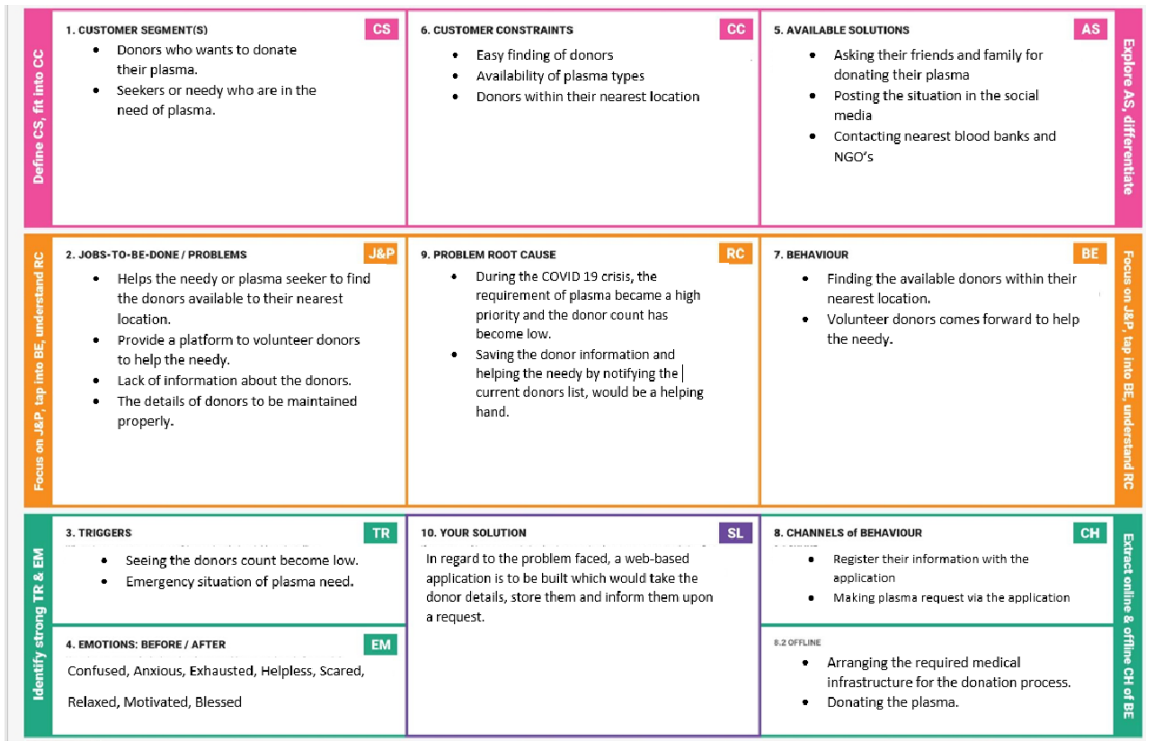
2.	Idea / Solution description	The user should be aware of the unit of plasma that is needed befo re checking the application to see if there is any plasma available.
3.	Novelty / Uniqueness	New users and those who are unsure o f how to use an application frequently experience these kinds of problems.
4.	Social Impact / Customer Satisfaction	There will be less likelihood of these problems recurring and the user will be more satisfied with the solution.
5.	Business Model (Revenue Model)	This donation application will brin g in money for hospitals, non-profits, and private businesses base d on revenue.
6.	Scalability of the Solution	The user's perspective of the applicat ion will change, and the user's flexib ility will allow for changes to the req uirements.

Problem statement 2



S. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	I work as an employee and I am attempting to utilise the Plasma Donar application because I want to use it, but I don't know how to use it and I've never used it before, which makes me feel anxious.
2.	Idea / Solution description	To use the application effectively, the user should have a basic understanding of it, read the user guide, or use the "Chat Bot" for assistance.
3.	Novelty / Uniqueness	It is common problem face by the new users who are trying to use the application. If the user once learns how to use, then there will be no issue.
4.	Social Impact / Customer Satisfaction	It is a typical issue that new users run into when attempting to use the application. There won't be a problem if the user learns how to use.
5.	Business Model (Revenue Model)	This donation application will bring in money for hospitals, non-profits, and private businesses based on revenue.
6.	Scalability of the Solution	The donar problem was resolved, and the user's flexibility allows for modification of the requirements.

3.4 Proposed solution fit



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- Donors who wants to donate their plasma.
- Seekers or needy who are in the need of plasma.

6. CUSTOMER CONSTRAINTS CC

- Easy finding of donors
- Availability of plasma types
- Donors within their nearest location

5. AVAILABLE SOLUTIONS AS

- Asking their friends and family for donating their plasma
- Posting the situation in the social media
- Contacting nearest blood banks and NGO's

Explore AS, differentiate

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

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

- Helps the needy or plasma seeker to find the donors available to their nearest location.
- Provide a platform to volunteer donors to help the needy.
- Lack of information about the donors.
- The details of donors to be maintained properly.

9. PROBLEM ROOT CAUSE RC

- 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.

7. BEHAVIOUR BE

- Finding the available donors within their nearest location.
- Volunteer donors comes forward to help the needy.

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

Identify strong TR & EM

3. TRIGGERS TR

- Seeing the donors count become low.
- Emergency situation of plasma need.

4. EMOTIONS: BEFORE / AFTER EM

Confused, Anxious, Exhausted, Helpless, Scared,

Relaxed, Motivated, Blessed

10. YOUR SOLUTION SL

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

8. CHANNELS of BEHAVIOUR CH

- Register their information with the application
- Making plasma request via the application

8.2 OFFLINE

- Arranging the required medical infrastructure for the donation process.
- Donating the plasma.

Extract online & offline CH of BE

CHAPTER 4: REQUIRMENT ANALYSIS

4.1 Functional Requirements

FR No.	**Non-Functional Requirement**	**Description**
FR-1	Registration	Registration through Form Registration through Gmail
FR-2	Registration Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Donor Profilecreation	A volunteer donor can create their donor profile by filling out the form with their medical data and previous donations.
FR-4	Request for plasma	The user can request plasma by completing the plasma request form.
FR-5	Virtual Donor Card	A virtual donor card that symbolises their giving activity will be sent to active donors.
FR-6	Dashboard	A statistical dashboard with information on the availability of donors will be made available to each user.
FR-7	Request Details	The clinic, blood bank, and ability to read the blood or plasma requestId, time the blood or plasma request was made, name of the clinic, and

FR-8	Distribution Status	The Clinic, Blood Bank, or Plasma Bank should have access to the distribution time status. The clinic manager must be able to contact the person in charge of distribution if it appears that the distribution is running behind schedule.

4.2 Non-functional Requirements

NFR No.	**Non-Functional Requirement**	**Description**
NFR-1	Maintainability	High levels of maintainability are required for the plasma donor application system.
NFR-2	Servicability	If a problem develops with the plasma donar application system, the project must be programmed so that the developer can fix it once more.
NFR-3	Environmental	The use of plasma donar System must function well under the most recent versions of Windows 7, Windows 8, Windows 10, and Linux.
NFR-4	Data Integrity	The Plasma Donor Application System must contain only correct and trustworthy data.

NFR-5	Usability	A user-friendly interface is essential for the plasma donor application system.
NFR-6	Recoverability	A suitable data backup system must be included in the plasma donor application system.
NFR-7	Interoperability	The plasma donar application system must work with or use the parts or equipment of another system.

CHAPTER 5: PROJECT DESIGN

5.1 Data Flow Diagram:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

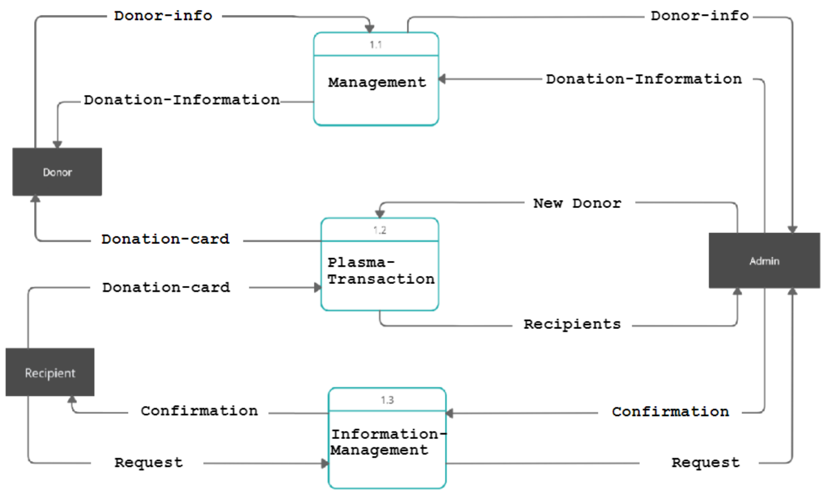


Fig.5.1. Data flow Diagram

5.2 Solution and Technical Architecture

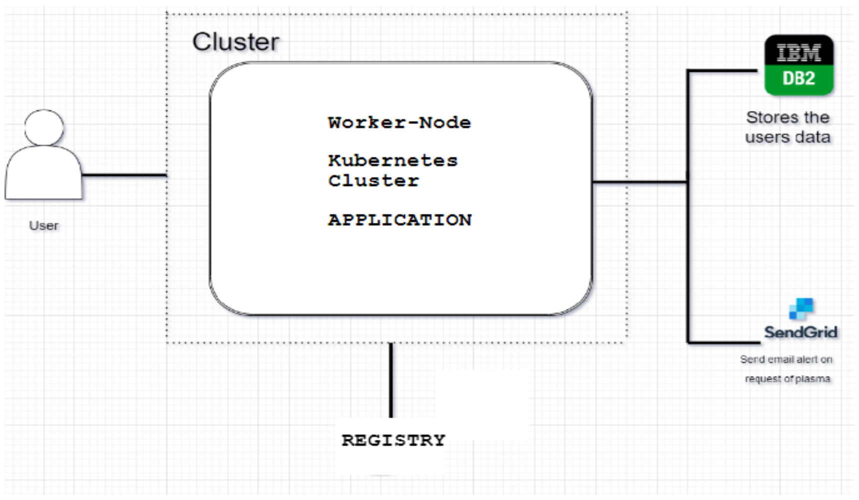


Fig.5.2. Solution and Technical Architecture

5.3 User Stories

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I mist be able to register my account using my details	5	High	Keshav Adithya SP, MHNS Sriram Raju
Sprint-1	Verification of email	USN-2	As a user, I should receive a confirmation mail on registering	4	High	MHNS Sriram Raju, Naveenkumar S
Sprint-1	User Login	USN-3	As a user, I must be able to log into my profile	5	High	Keshav Adithya SP, Abubakar Siddick K
Sprint-1	Donor Profile	USN-4	As a user, I must be able to register as a donor	5	High	Naveenkumar S, Abubakar Siddick K
Sprint-2	Dashbo ard	USN-5	As a user, I must be able to see availbility of donors and other information on my dashboard	5	High	Keshav Adithya SP, MHNS Sriram Raju, Naveenkumar S
Sprint-2	Plasma Request	USN-6	As a user, requesting for plasma through an application must be implemented	5	High	Naveenkumar S, Abubakar Siddick K
Sprint-2		USN-7	As a user, I must be able to upload related documentation and get verified as a donor	5	High	MHNS Sriram Raju, Naveenkum ar S, Abubakar Siddick K
Sprint-3	Acceptance of request	USN-8	As a verified donor, I must be able to accept the donation requests from the recipients	5	High	Keshav Adithya SP, Naveenkumar S, Abubakar Siddick K
Sprint-3	Appoint ment for donating	USN-9	As a verified donor, I must be able to book an appointment to donate.	4	High	Keshav Adithya SP, MHNS Sriram Raju, Naveenkum ar S
Sprint-3		USN-10	As a verified donor, sharing of information must be made plausible between donor and recipient	3	Medium	MHNS Sriram Raju, Naveenkuma r S
Sprint-3	Admin	USN-15	As an admin, I must be able to manage the entire management of the application	5	High	Keshav Adithya SP, MHNS Sriram Raju, Naveenkumar S, Abubakar Siddick K
Sprint-4	About	USN-18	As a user and if I am new to plasma donation, I can read about the plasmaand plasma donation in dedication	3	Medium	Keshav Adithya SP

			about section			
Sprint-4	Administrator	USN-19	As an admin, I will approve all the plasma transaction in the application after the proper verification	5	High	MHNS Sriram Raju, Naveenkumar S

CHAPTER 6: PROJECT PLANNING & 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 mist be able to register my account using my details	5	High	Keshav Adithya SP, MHNS Sriram Raju
Sprint-1	Verification of email	USN-2	As a user, I should receive a confirmation mail on registering	4	High	MHNS Sriram Raju, Naveenkumar S
Sprint-1	User Login	USN-3	As a user, I must be able to log into my profile	5	High	Keshav Adithya SP, Abubakar Siddick K
Sprint-1	Donor Profile	USN-4	As a user, I must be able to register as a donor	5	High	Naveenkumar S, Abubakar Siddick K
Sprint-2	Dashbo ard	USN-5	As a user, I must be able to see availbility of donors and other information on my dashboard	5	High	Keshav Adithya SP, MHNS Sriram Raju, Naveenkumar S
Sprint-2	Plasma Request	USN-6	As a user, requesting for plasma through an application must be implemented	5	High	Naveenkumar S, Abubakar Siddick K
Sprint-2		USN-7	As a user, I must be able to upload related documentation and get verified as a donor	5	High	MHNS Sriram Raju, Naveenkum ar S, Abubakar Siddick K
Sprint-3	Acceptance of request	USN-8	As a verified donor, I must be able to accept the donation requests from the recipients	5	High	Keshav Adithya SP, Naveenkumar S, Abubakar Siddick K
Sprint-3	Appoint ment for donating	USN-9	As a verified donor, I must be able to book an appointment to donate.	4	High	Keshav Adithya SP, MHNS Sriram Raju, Naveenkum ar S
Sprint-3		USN-10	As a verified donor, sharing of information must be made plausible between donor and recipient	3	Medium	MHNS Sriram Raju, Naveenkuma r S
Sprint-3	Admin	USN-15	As an admin, I must be able to manage the entire management of the application	5	High	Keshav Adithya SP, MHNS Sriram Raju, Naveenkumar S, Abubakar Siddick K

Sprint-4	About	USN-18	As a user and if I am new to plasma donation, I can read about the plasmaand plasma donation in dedication about section	3	Medium	Keshav Adithya SP
Sprint-4	Administrator	USN-19	As an admin, I will approve all the plasma transaction in the application after the proper verification	5	High	MHNS Sriram Raju, Naveenkum ar S

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	19	6 Days	30th Oct 2022	05th Oct 2022	19	05th Oct 2022
Sprint-2	20	6 Days	06th Nov 2022	11th Nov 2022	20	11th Nov 2022
Sprint-3	17	6 Days	12th Nov 2022	17th Nov 2022	17	17th Nov 2022
Sprint-4	8	6 Days	18th Nov 2022	19th Nov 2022	8	19th Nov 2022

6.3 Reports from JIRA:

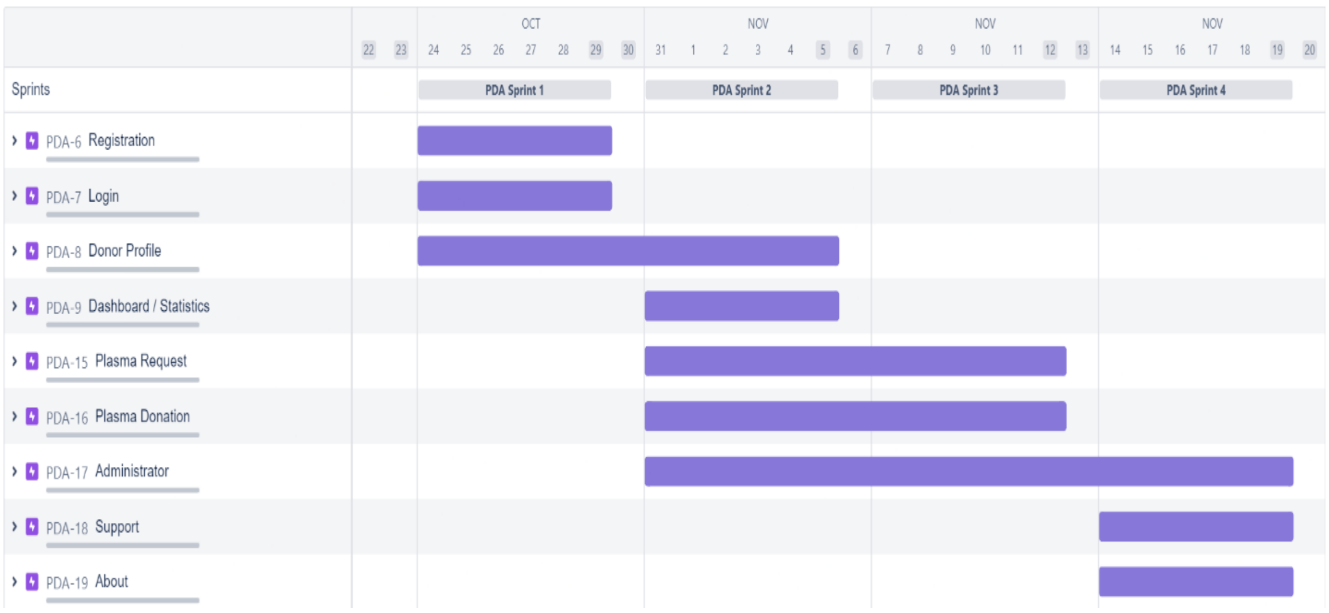
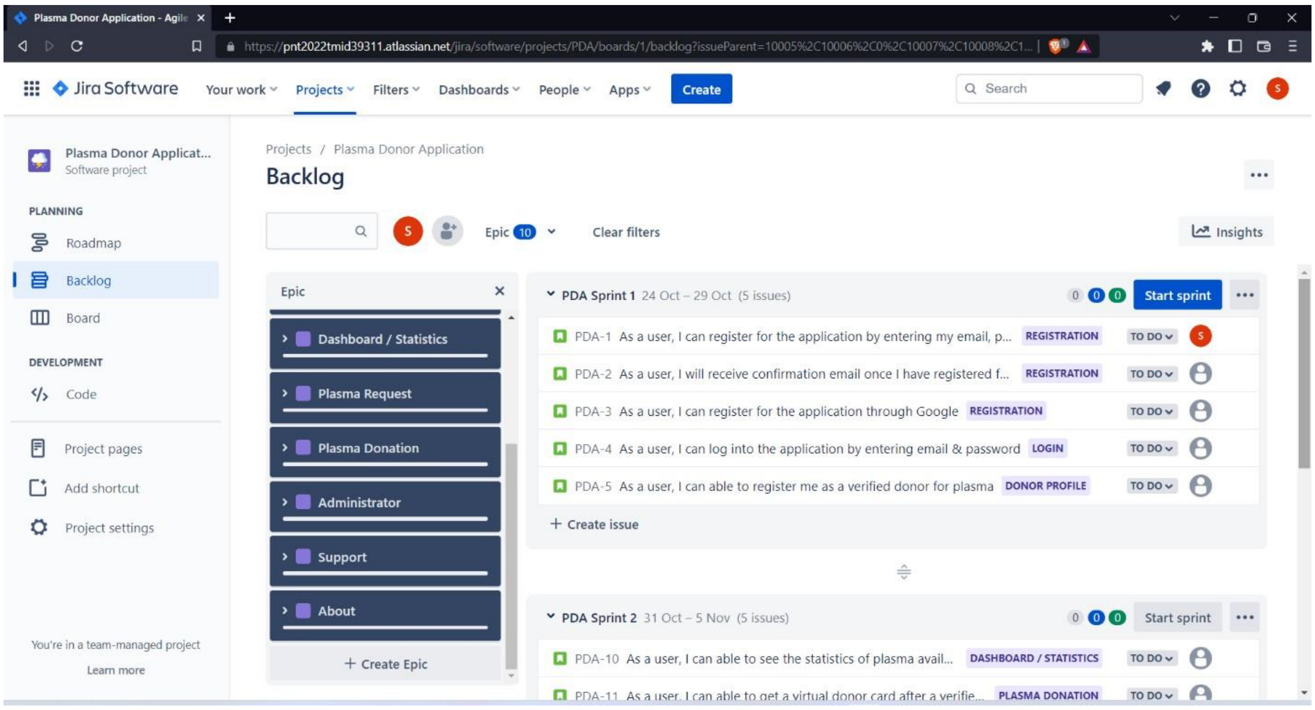


Fig 6.1.1



Plasma Donor Application - Agile

https://pnt2022tmid39311.atlassian.net/jira/software/projects/PDA/boards/1/backlog?issueParent=10005%2C10006%2C0%2C10007%2C10008%2C1...

Jira Software | Your work | Projects | Filters | Dashboards | People | Apps | Create | Search

Plasma Donor Applicat... Software project

PLANNING: Roadmap, Backlog, Board

DEVELOPMENT: Code

Project pages, Add shortcut, Project settings

You're in a team-managed project. Learn more

Projects / Plasma Donor Application

Backlog

Epic 10 | Clear filters | Insights

Epic: Dashboard / Statistics, Plasma Request, Plasma Donation, Administrator, Support, About, + Create Epic

PDA Sprint 1 24 Oct – 29 Oct (5 issues) 0 0 0 Start sprint

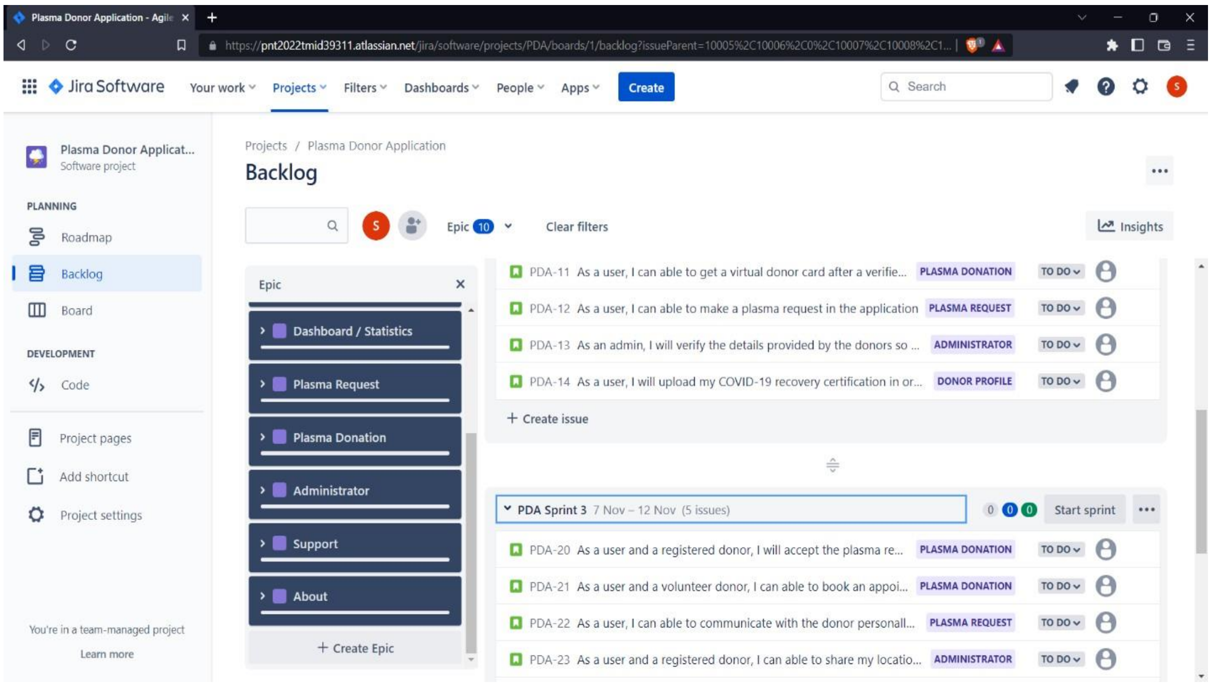
PDA-1	As a user, I can register for the application by entering my email, p...	REGISTRATION	TO DO
PDA-2	As a user, I will receive confirmation email once I have registered f...	REGISTRATION	TO DO
PDA-3	As a user, I can register for the application through Google	REGISTRATION	TO DO
PDA-4	As a user, I can log into the application by entering email & password	LOGIN	TO DO
PDA-5	As a user, I can able to register me as a verified donor for plasma	DONOR PROFILE	TO DO

+ Create issue

PDA Sprint 2 31 Oct – 5 Nov (5 issues) 0 0 0 Start sprint

PDA-10	As a user, I can able to see the statistics of plasma avail...	DASHBOARD / STATISTICS	TO DO
PDA-11	As a user, I can able to get a virtual donor card after a verifie...	PLASMA DONATION	TO DO

Fig 6.1.2



Plasma Donor Application - Agile

https://pnt2022tmid39311.atlassian.net/jira/software/projects/PDA/boards/1/backlog?issueParent=10005%2C10006%2C0%2C10007%2C10008%2C1...

Jira Software | Your work | Projects | Filters | Dashboards | People | Apps | Create | Search

Plasma Donor Applicat... Software project

PLANNING: Roadmap, Backlog, Board

DEVELOPMENT: Code

Project pages, Add shortcut, Project settings

You're in a team-managed project. Learn more

Projects / Plasma Donor Application

Backlog

Epic 10 | Clear filters | Insights

Epic: Dashboard / Statistics, Plasma Request, Plasma Donation, Administrator, Support, About, + Create Epic

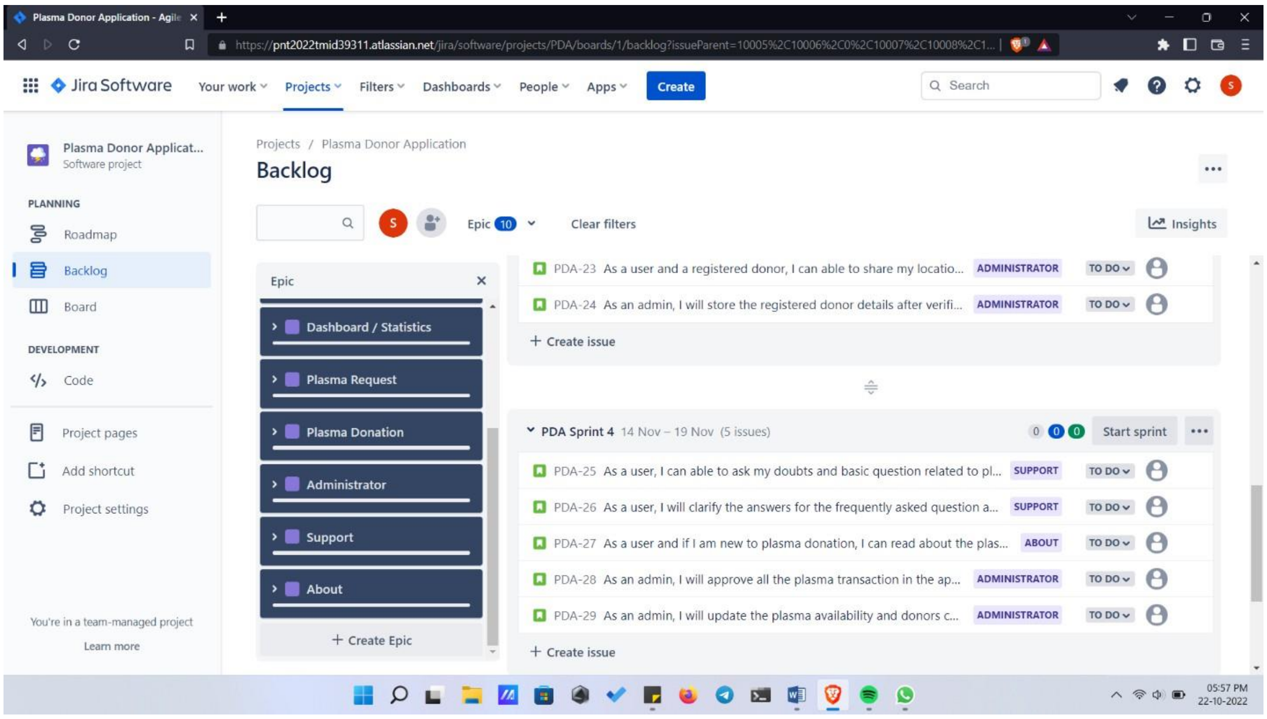
PDA-11	As a user, I can able to get a virtual donor card after a verifie...	PLASMA DONATION	TO DO
PDA-12	As a user, I can able to make a plasma request in the application	PLASMA REQUEST	TO DO
PDA-13	As an admin, I will verify the details provided by the donors so ...	ADMINISTRATOR	TO DO
PDA-14	As a user, I will upload my COVID-19 recovery certification in or...	DONOR PROFILE	TO DO

+ Create issue

PDA Sprint 3 7 Nov – 12 Nov (5 issues) 0 0 0 Start sprint

PDA-20	As a user and a registered donor, I will accept the plasma re...	PLASMA DONATION	TO DO
PDA-21	As a user and a volunteer donor, I can able to book an appoi...	PLASMA DONATION	TO DO
PDA-22	As a user, I can able to communicate with the donor personall...	PLASMA REQUEST	TO DO
PDA-23	As a user and a registered donor, I can able to share my locatio...	ADMINISTRATOR	TO DO

Fig 6.1.3



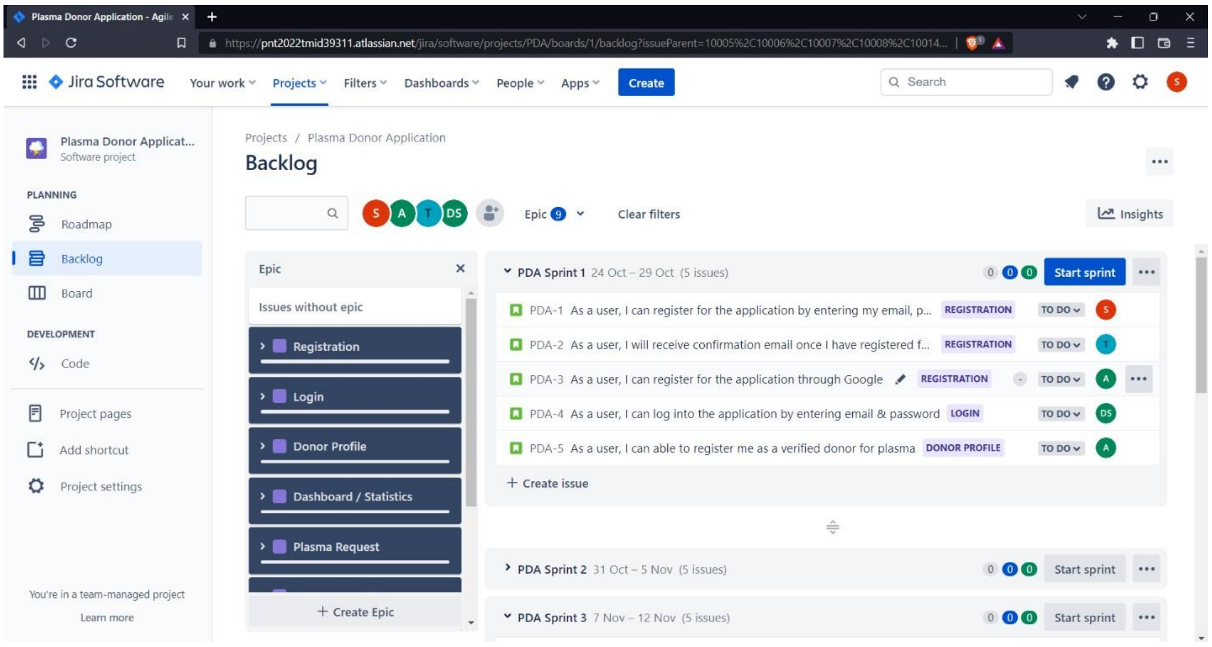
Backlog

PDA-23	As a user and a registered donor, I can able to share my locatio...	ADMINISTRATOR	TO DO
PDA-24	As an admin, I will store the registered donor details after verifi...	ADMINISTRATOR	TO DO

PDA Sprint 4 14 Nov – 19 Nov (5 issues) 0 0 0

PDA-25	As a user, I can able to ask my doubts and basic question related to pl...	SUPPORT	TO DO
PDA-26	As a user, I will clarify the answers for the frequently asked question a...	SUPPORT	TO DO
PDA-27	As a user and if I am new to plasma donation, I can read about the plas...	ABOUT	TO DO
PDA-28	As an admin, I will approve all the plasma transaction in the ap...	ADMINISTRATOR	TO DO
PDA-29	As an admin, I will update the plasma availability and donors c...	ADMINISTRATOR	TO DO

Fig 6.1.4



Backlog

PDA Sprint 1 24 Oct – 29 Oct (5 issues) 0 0 0

PDA-1	As a user, I can register for the application by entering my email, p...	REGISTRATION	TO DO	S
PDA-2	As a user, I will receive confirmation email once I have registered f...	REGISTRATION	TO DO	T
PDA-3	As a user, I can register for the application through Google	REGISTRATION	TO DO	A
PDA-4	As a user, I can log into the application by entering email & password	LOGIN	TO DO	DS
PDA-5	As a user, I can able to register me as a verified donor for plasma	DONOR PROFILE	TO DO	A

PDA Sprint 2 31 Oct – 5 Nov (5 issues) 0 0 0

PDA Sprint 3 7 Nov – 12 Nov (5 issues) 0 0 0

Fig 6.1.5

CHAPTER 7: CODING AND SOLUTIONING

7.1 Plasma request and donation:

Web framework :

Web application developers can design apps without having to be concerned about low-level aspects like protocol, thread management, and other issues thanks to a web framework, also known as a web application framework.

Flask :

Python is used to create the Flask web application framework. It was created by Armin Ronacher, who served as the team leader of Poocco, an international group of Python aficionados. The Werkzeg WSGI toolkit and the Jinja2 template engine serve as the foundation for Flask. They're both Pocco projects.

WSGI :

For the creation of Python web applications, the Web Server Gateway Interface (WSGI) has been the de facto standard. The WSGI specification describes a standard interface for web servers and online applications.

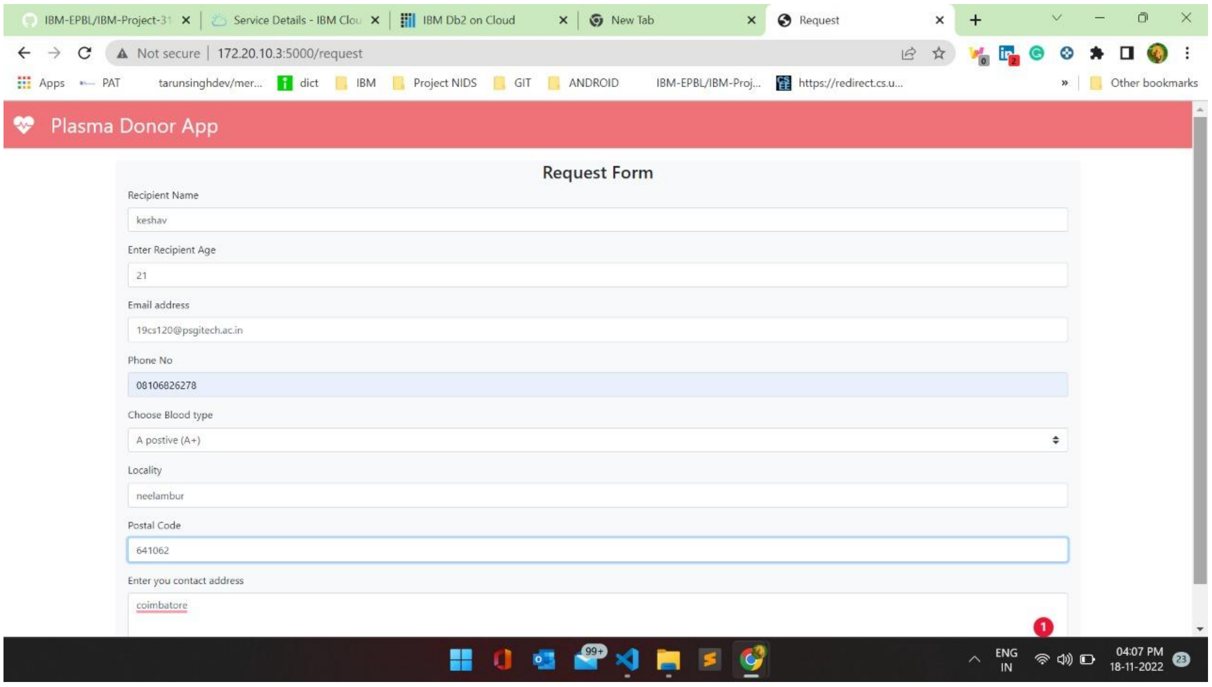
Werkzeug :

Requests, response objects, and utility functions are all implemented by the WSGI toolkit known as Werkzeug. On top of it can now be created a web frame. Werkzeg serves as one of the foundations of the Flask framework.

Jinja2 :

A well-liked Python template engine is Jinja2. A web template system renders a dynamic web page by fusing a template with a particular data source.

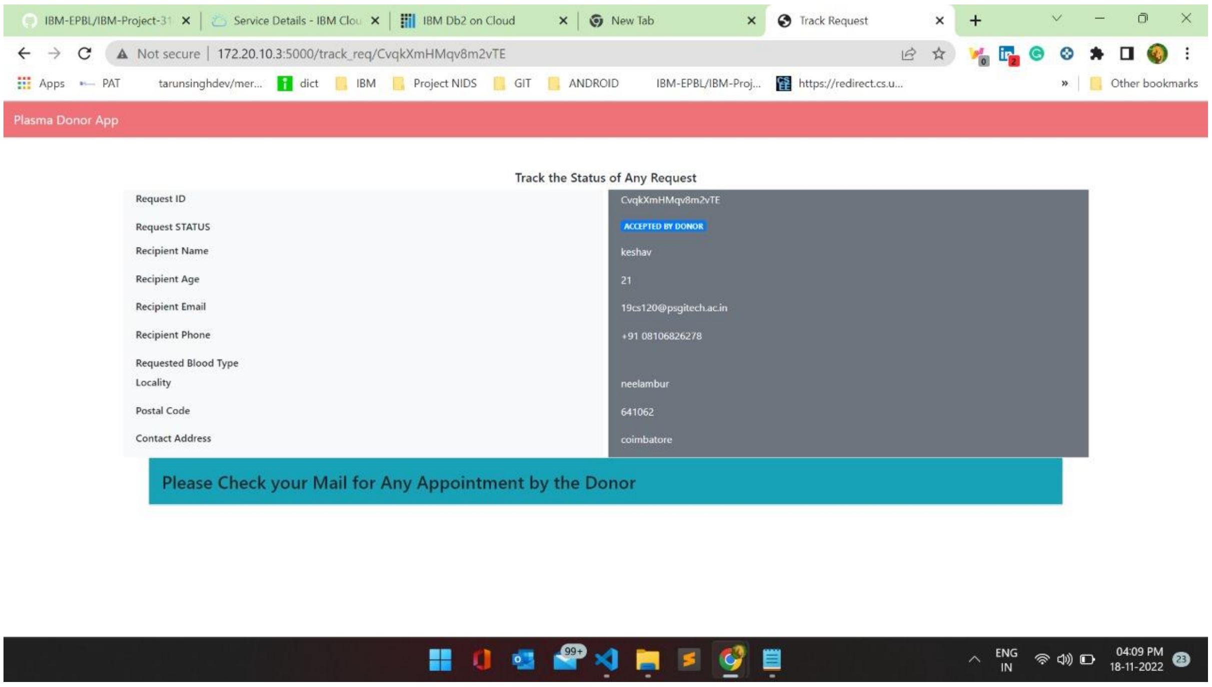
Screenshots:


Plasma Donor App

Request Form

Recipient Name: keshav
Enter Recipient Age: 21
Email address: 19cs120@psgitech.ac.in
Phone No: 08106826278
Choose Blood type: A postive (A+)
Locality: neelambur
Postal Code: 641062
Enter you contact address: coimbatore

Fig 7.1.1

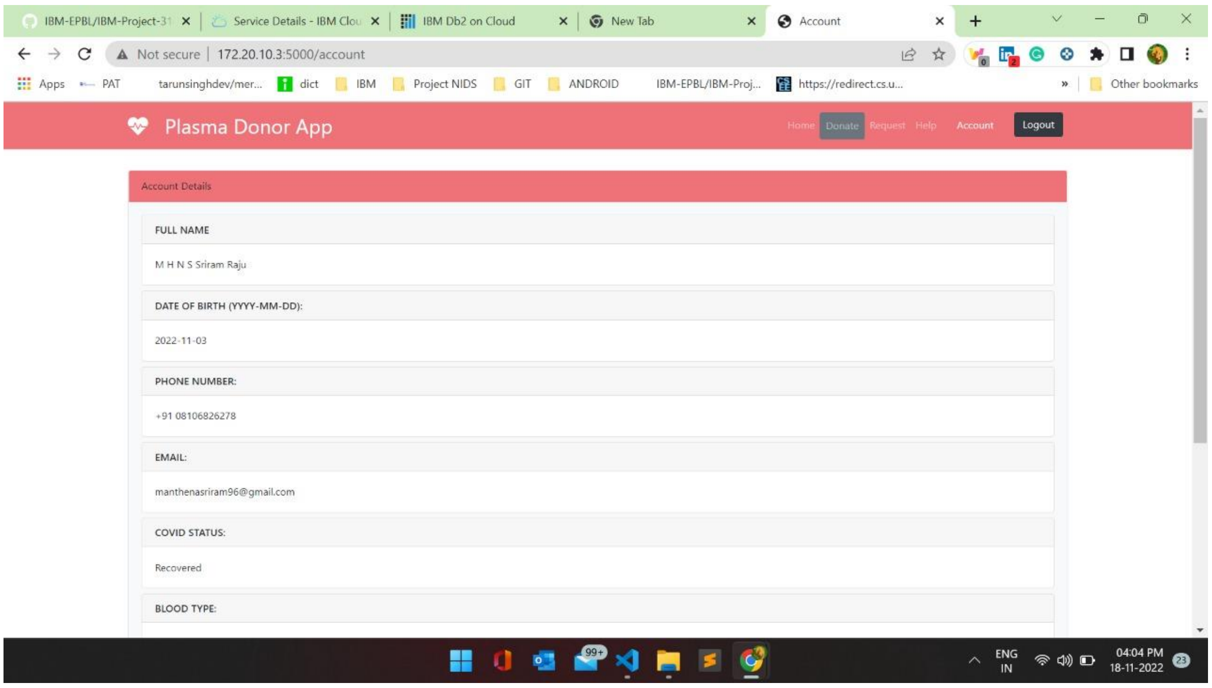

Plasma Donor App

Track the Status of Any Request

Request ID	CvqkXmHMqv8m2vTE
Request STATUS	ACCEPTED BY DONOR
Recipient Name	keshav
Recipient Age	21
Recipient Email	19cs120@psgitech.ac.in
Recipient Phone	+91 08106826278
Requested Blood Type	
Locality	neelambur
Postal Code	641062
Contact Address	coimbatore

Please Check your Mail for Any Appointment by the Donor

Fig 7.1.2



Account Details

FULL NAME

M H N S Sriram Raju

DATE OF BIRTH (YYYY-MM-DD):

2022-11-03

PHONE NUMBER:

+91 08106826278

EMAIL:

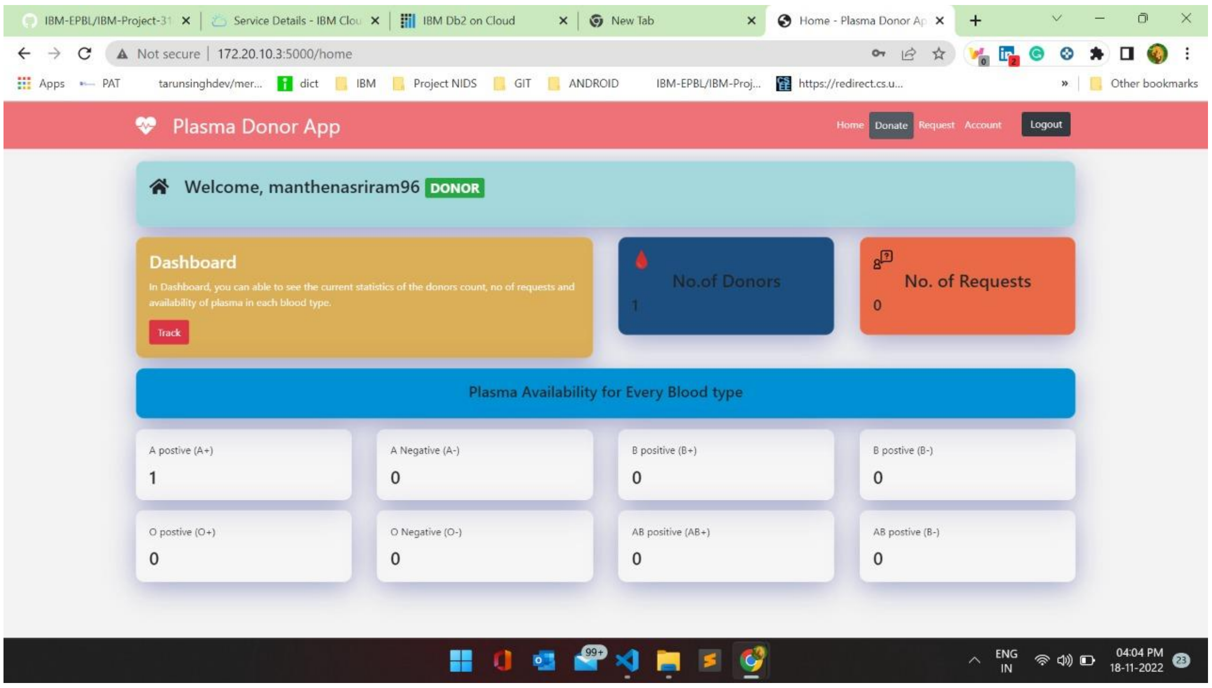
manthenasriram96@gmail.com

COVID STATUS:

Recovered

BLOOD TYPE:

Fig 7.1.3



Welcome, manthenasriram96 DONOR

Dashboard

In Dashboard, you can able to see the current statistics of the donors count, no of requests and availability of plasma in each blood type.

Track

No.of Donors

1

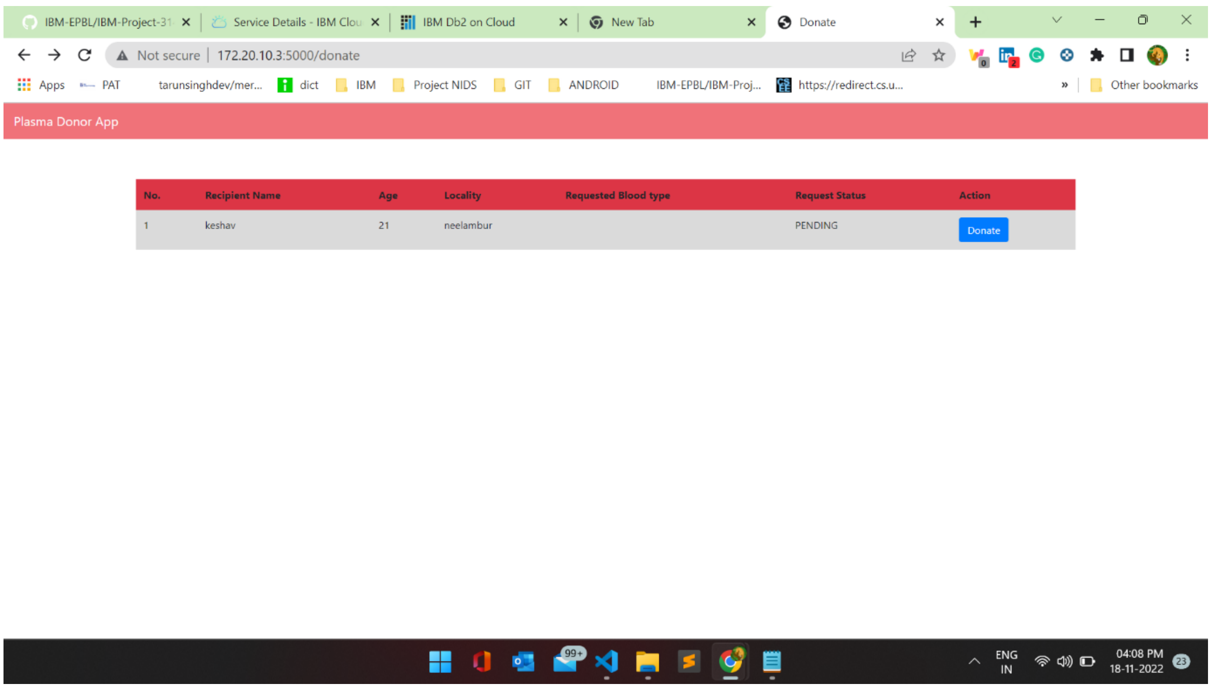
No. of Requests

0

Plasma Availability for Every Blood type

A postive (A+) 1	A Negative (A-) 0	B positive (B+) 0	B postive (B-) 0
O postive (O+) 0	O Negative (O-) 0	AB positive (AB+) 0	AB postive (B-) 0

Fig 7.1.4



Plasma Donor App

No.	Recipient Name	Age	Locality	Requested Blood type	Request Status	Action
1	keshav	21	neelambur		PENDING	Donate

Fig 7.1.5

App.py

```
from flask import Flask, render_template, redirect
from flask import url_for, session, request
from dotenv import load_dotenv
from mailer import send_the_email
from datetime import datetime
from generator import generate_unique_id
from fetch import fetch_home
from check import check_the_acc_info
import os
import hashlib
import re
import ibm_db
load_dotenv()

app = Flask(__name__)
app.secret_key = os.urandom(16)

try:
    # conn = ibm_db.connect(os.getenv('CREDENTIALS'), '', '')
    # conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=764264db-9824-4b7c-82df-
40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32536;SECURIT
Y=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=hmd83768;PWD=4WzDtnP
yc6CW98X2", '', '')
    conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=ea286ace-86c7-4d5b-8580-
3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31505;SECURITY
=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lxj14894;PWD=Gz86RyxT6U
VIv15C", '', '')
```

```
except Exception as err:
    print(ibm_db.conn_errormsg())


@app.route('/')
def index():
    if not session:
        return render_template('index.htm')

    return redirect(url_for('home'))


@app.route('/login')
def login():
    if not session or not session['login_status']:
        return render_template('login.htm')

    return redirect(url_for('home'))


@app.route('/register')
def register():
    return render_template('register.htm')


@app.route('/account')
def account():
    if not session:
        return redirect(url_for('home'))
    if session['account-type'] == 'Donor':
        useremail = session['user_email']
        sql = "SELECT
FIRSTNAME,LASTNAME,DOB,PHONE,USER_EMAIL,BLOOD_TYPE,COVID_STAT
US,GENDER,STATE,PINCODE FROM DONORS WHERE USER_EMAIL=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, useremail)
        ibm_db.execute(stmt)
        res = ibm_db.fetch_assoc(stmt)
        return render_template('account.htm', res=res)
    if session['account-type'] == 'user':
        useremail = session['user_email']
        sql = "SELECT FULLNAME,USER_DOB,PHONE_NO,EMAIL FROM USERS
WHERE EMAIL=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, useremail)
        ibm_db.execute(stmt)
        result = ibm_db.fetch_assoc(stmt)
        return render_template('account.htm', res=result)
```

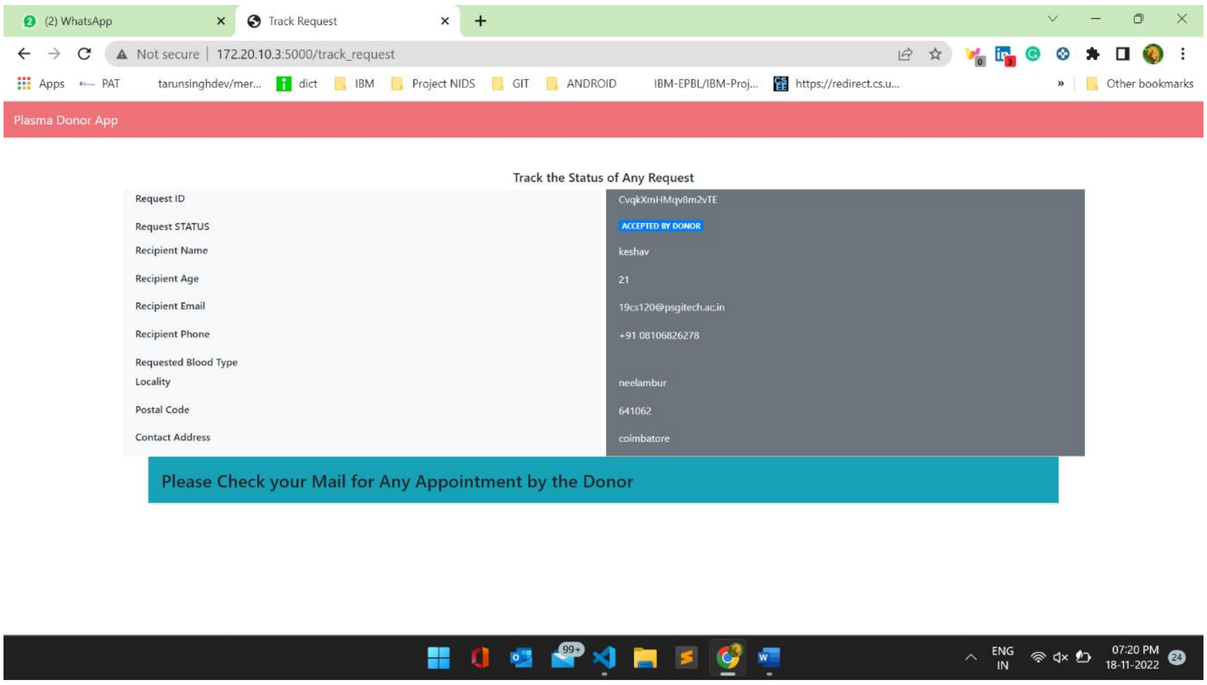
```
@app.route('/donate')
def donate():
    if not session or not session['login_status']:
        return render_template('login.htm')

    if session['account-type'] == 'user':
        return redirect(url_for('register'))

    results = {}
    sql = "SELECT * FROM Requests WHERE REQUEST_STATUS=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, 'PENDING')
    ibm_db.execute(stmt)
    result = ibm_db.fetch_assoc(stmt)
    i = 1
    while result:
        results.update({i: result})
        i = i + 1
        result = ibm_db.fetch_assoc(stmt)
    return render_template('donate.htm', results=results)


@app.route('/BookAppointment/<req_id>')
def book_appointment(req_id):

    return render_template('donateForm.htm', req_id=req_id)


@app.route('/err')
def err():
    return render_template('err.htm', err_msg)


@app.route('/track')
def track():
    session['track_id'] = False
    return render_template('track.htm')


@app.route('/request')
def _request():
    if not session or not session['login_status']:
        return render_template('user_registration.htm')

    return render_template('request.htm')
```

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

    if request.method == 'POST':
        track_id = request.form['tracking-id']

        sql = "SELECT * FROM REQUESTS WHERE REQUEST_ID=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, track_id)
        ibm_db.execute(stmt)
        res = ibm_db.fetch_assoc(stmt)
        if res:
            session['track_id'] = True
            return render_template('track.htm', res=res)
        if not res:
            err_msg = 'There is no such request with this request id. '
            err_msg += 'Please Check Your Request ID once again'
            return render_template('err.htm', err_msg=err_msg)


@app.route('/track_req/<req_id>')
def track_req(req_id):
    track_id = req_id
    sql = "SELECT * FROM REQUESTS WHERE REQUEST_ID=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, track_id)
    ibm_db.execute(stmt)
    res = ibm_db.fetch_assoc(stmt)
    if res:
        session['track_id'] = True
        return render_template('track.htm', res=res)
    if not res:
        err_msg = 'There is no such request with this request id. '
        err_msg += 'Please Check Your Request ID once again'
        return render_template('err.htm', err_msg=err_msg)


@app.route('/user_register', methods=['GET', 'POST'])
def user_register():
    if request.method == 'POST':
        user_name = request.form['username']
        user_dob = request.form['dob']
        user_phone = request.form['user-phone']
        user_email = request.form['useremail']
        password = request.form['password']
        cnf_password = request.form['cnf-password']

        # hashing the password
        if password != cnf_password:
            msg = "Password Doesn't Match"
            return render_template('err.htm', err_msg=msg)

        password = bytes(password, 'utf-8')
        password = hashlib.sha256(password).hexdigest()
        # password hashed

    # case 1: check if user does exists already
        sql = "SELECT * FROM users WHERE email =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, user_email)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_assoc(stmt)
        if acc:
            msg = "Account already Exists, Please login"
            return render_template('err.htm', err_msg=msg)

         # case 2: validate the input if it matches the required pattern
        if not re.match(r"^\S+@\S+\.\S+$", user_email):
            msg = "Please Enter Valid Email Address "
            return render_template('err.htm', err_msg=msg)

        insert_sql = "INSERT INTO  users VALUES (?, ?, ?, ?, ?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, user_name)
        ibm_db.bind_param(prep_stmt, 2, user_dob)
        ibm_db.bind_param(prep_stmt, 3, user_phone)
        ibm_db.bind_param(prep_stmt, 4, user_email)
        ibm_db.bind_param(prep_stmt, 5, password)
        ibm_db.execute(prep_stmt)

        to_email = user_email
        subject = "Confirmation on Registration with Plasma-Donor-App as User"
        html_content = '''

          <h1>Registration Successfull</h1><br>
          <p> Thank you so much for registering with us </p><br>
         <p> You are now registered user </p>

        '''
        send_the_email(to_email, subject, html_content)
        return redirect(url_for('login'))


@app.route('/home')
def home():
    if not session:
        return redirect(url_for('login'))
    if session['login_status']:
        req, res = fetch_home(conn=conn)
        return render_template('home.htm', username=session['user_id'], req=req, res=res)

    return redirect(url_for('login'))


@app.route('/do_register', methods=['GET', 'POST'])
def do_register():
    if request.method == 'POST':
        first_name = request.form['fname']
        last_name = request.form['lname']
        email = request.form['email']
        addrss1 = request.form['Locality']
        addrss2 = request.form['address']
        state = request.form['State']
        pincode = request.form['Zip']
        dob = request.form['dob']
        gender = request.form['gender']
        phone = request.form['phone']
        covid_status = request.form['covid-report']
        blood_type = request.form['b-type']
        # ------------------
        # password hashing
        password = request.form['password']
        cnf_password = request.form['cnf-password']
        if password != cnf_password:
            msg = "Password Doesn't Match"
            return render_template('err.htm', err_msg=msg)

        password = bytes(password, 'utf-8')
        password = hashlib.sha256(password).hexdigest()

        # case 1: check if user does exists already
        sql = "SELECT * FROM donors WHERE user_email =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_assoc(stmt)
        if acc:
            msg = "Account already Exists, Please login "
            return render_template('err.htm', err_msg=msg)

        # case 2: validate the input if it matches the required pattern
        if not re.match(r"^\S+@\S+\.\S+$", email):
            msg = "Please Enter Valid Email Address "
            return render_template('err.htm', err_msg=msg)

        insert_sql = "INSERT INTO  donors VALUES (?, ?, ?, ?, ?, ?, ?,?, ?, ?, ?, ?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, first_name)
        ibm_db.bind_param(prep_stmt, 2, last_name)
        ibm_db.bind_param(prep_stmt, 3, email)
        ibm_db.bind_param(prep_stmt, 4, addrss1)
        ibm_db.bind_param(prep_stmt, 5, addrss2)
        ibm_db.bind_param(prep_stmt, 6, state)
        ibm_db.bind_param(prep_stmt, 7, pincode)
        ibm_db.bind_param(prep_stmt, 8, dob)
        ibm_db.bind_param(prep_stmt, 9, gender)
        ibm_db.bind_param(prep_stmt, 10, phone)
        ibm_db.bind_param(prep_stmt, 11, covid_status)
        ibm_db.bind_param(prep_stmt, 12, blood_type)
        ibm_db.bind_param(prep_stmt, 13, password)
        ibm_db.execute(prep_stmt)

        to_email = email
        subject = 'Confirmation on Registration with Plasma-Donor-App'
        html_content = '''
            <h1>Registration Successfull</h1><br>
            <p> Thank you so much for registering with us </p><br>
            <p> You are now registered donor </p>
        '''
        send_the_email(to_email, subject, html_content)
        return redirect(url_for('login'))
    return redirect(url_for('register'))


@app.route('/do_login', methods=['GET', 'POST'])
def do_login():
    if request.method == 'POST':
        user_email = request.form['user_email']
        password = request.form['password']
        # salt the password
        password = bytes(password, 'utf-8')
        password = hashlib.sha256(password).hexdigest()

        # query the db
        sql = "SELECT * FROM donors WHERE user_email =? AND pass_word=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, user_email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_assoc(stmt)
        if not acc:
            # check if present in users
            sql = "SELECT * FROM users WHERE email =? AND password=?"
            stmt = ibm_db.prepare(conn, sql)
            ibm_db.bind_param(stmt, 1, user_email)
            ibm_db.bind_param(stmt, 2, password)
            ibm_db.execute(stmt)
            acc = ibm_db.fetch_assoc(stmt)
            session['account-type'] = 'user'
            session['login_status'] = True
            session['user_email'] = user_email
            session['user_id'] = user_email.split('@')[0]
            return redirect(url_for('home'))
        if acc:
            session['login_status'] = True
            session['account-type'] = 'Donor'
            session['user_email'] = user_email
            session['user_id'] = user_email.split('@')[0]
            return redirect(url_for('home'))

        # check if the acc exists
        sql = "SELECT * FROM donors WHERE user_email=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, user_email)
        ibm_db.execute(stmt)
        res = ibm_db.fetch_assoc(stmt)
        if res:
            msg = "Account already Exists, Please login "
            return render_template('err.htm', err_msg=msg)
        else:
            msg = "Don't you have an account ? try register with us "
            return render_template('err.htm', err_msg=msg)


@app.route('/do_request', methods=['GET', 'POST'])
def do_request():
    if request.method == 'POST':
        name = request.form['name']
        age = request.form['age']
        email = request.form['email']
        phone = request.form['phone']
        requested_blood_type = request.form['blood-type']
        locality = request.form['locality']
        postal_code = request.form['postal-code']
        address = request.form['contact-addrss']

        # generate request id
        request_id = generate_unique_id()
        # initial status of the request
        request_status = 'PENDING'

        insert_sql = "INSERT INTO  requests VALUES (?, ?, ?, ?, ?, ?, ?,?, ?, ?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, request_id)
        ibm_db.bind_param(prep_stmt, 2, request_status)
        ibm_db.bind_param(prep_stmt, 3, name)
        ibm_db.bind_param(prep_stmt, 4, age)
        ibm_db.bind_param(prep_stmt, 5, email)
        ibm_db.bind_param(prep_stmt, 6, phone)
        ibm_db.bind_param(prep_stmt, 7, requested_blood_type)
        ibm_db.bind_param(prep_stmt, 8, locality)
        ibm_db.bind_param(prep_stmt, 9, postal_code)
        ibm_db.bind_param(prep_stmt, 10, address)
        ibm_db.execute(prep_stmt)

        return render_template('success.htm', request_id=request_id)


@app.route('/make_donation', methods=['GET', 'POST'])
def make_donation():
    if request.method == 'POST':
        request_id = request.form['req_id']
        donor_name = request.form['donor-name']
        donor_age = request.form['donor-age']
        blood_type = request.form['blood-type']
        medical_status = request.form['medical-status']
        location = request.form['location']
        date_time = request.form['datetime']
        date_time = datetime.strptime(date_time, '%Y-%m-%dT%H:%M')
        phone_number = request.form['phone-number']
        contact_address = request.form['contact-address']

        datenow = datetime.now().strftime('%Y-%m-%dT%H:%M')
        if str(date_time) < datenow:
            msg = "The Date you've entered is not suitable for making this appointment"
            return render_template('err.htm', err_msg=msg)

        chck = "SELECT * FROM Appointments WHERE request_id=?"
        stmt = ibm_db.prepare(conn, chck)
        ibm_db.bind_param(stmt, 1, request_id)
        ibm_db.execute(stmt)
        res = ibm_db.fetch_assoc(stmt)
        if res:
            msg = " The Request was Already Engaged"
            return render_template('err.htm', err_msg=msg)

        sql = "INSERT INTO  Appointments VALUES (?, ?, ?, ?, ?, ?, ?,?, ?)"
        prep_stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(prep_stmt, 1, request_id)
        ibm_db.bind_param(prep_stmt, 2, donor_name)
        ibm_db.bind_param(prep_stmt, 3, donor_age)
        ibm_db.bind_param(prep_stmt, 4, blood_type)
        ibm_db.bind_param(prep_stmt, 5, medical_status)
        ibm_db.bind_param(prep_stmt, 6, location)
        ibm_db.bind_param(prep_stmt, 7, date_time)
        ibm_db.bind_param(prep_stmt, 8, phone_number)
        ibm_db.bind_param(prep_stmt, 9, contact_address)
        ibm_db.execute(prep_stmt)

        upt_sql = "UPDATE requests SET request_status=? WHERE request_id=?"
        status = "ACCEPTED BY DONOR"
        upt_stmt = ibm_db.prepare(conn, upt_sql)
        ibm_db.bind_param(upt_stmt, 1, status)
        ibm_db.bind_param(upt_stmt, 2, request_id)
        ibm_db.execute(upt_stmt)

        msql = "SELECT recipient_email FROM requests WHERE request_id=?"
        mstmt = ibm_db.prepare(conn, msql)
        ibm_db.bind_param(mstmt, 1, request_id)
        ibm_db.execute(mstmt)
        res = ibm_db.fetch_assoc(mstmt)
        to_email = res['RECIPIENT_EMAIL']
        subject = f'Your Request ID {request_id} has been Accepted By The Donor and Please
refer the content of this mail'
        content = f'''
            <h1>Donor Found </h1>
            <h2>Details of the Donor and Appointment</h2>
            <body>
            <pre>
            Request ID      : {request_id}
            Donor's Name    : {donor_name}
            Donor's Age     : {donor_age}
            Medical Status  : {medical_status}
            Blood Type      : {blood_type}
            Location        : {location}
            Date and Time   : {date_time}
            Contact Address : {contact_address}
            </pre>
            <h3> You May contact the Donor For Full Details</h3>
            <h3>Get Well Soon</h3>
            </body>
        '''
        send_the_email(to_email, subject, content)

        return redirect('/track_req/'+request_id)


@app.route('/logout')
def logout():
    # session['login_status'] = False
    session.pop('login_status', None)
    session.pop('user_id', None)
    session.pop('user_email', None)
    session.pop('account-type', None)
    session.pop('track_id', None)

    return redirect(url_for('index'))


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

7.2 Fetching request details:

Details about the request made can be visualized and analysed using the implementation of this module.

Screenshots:



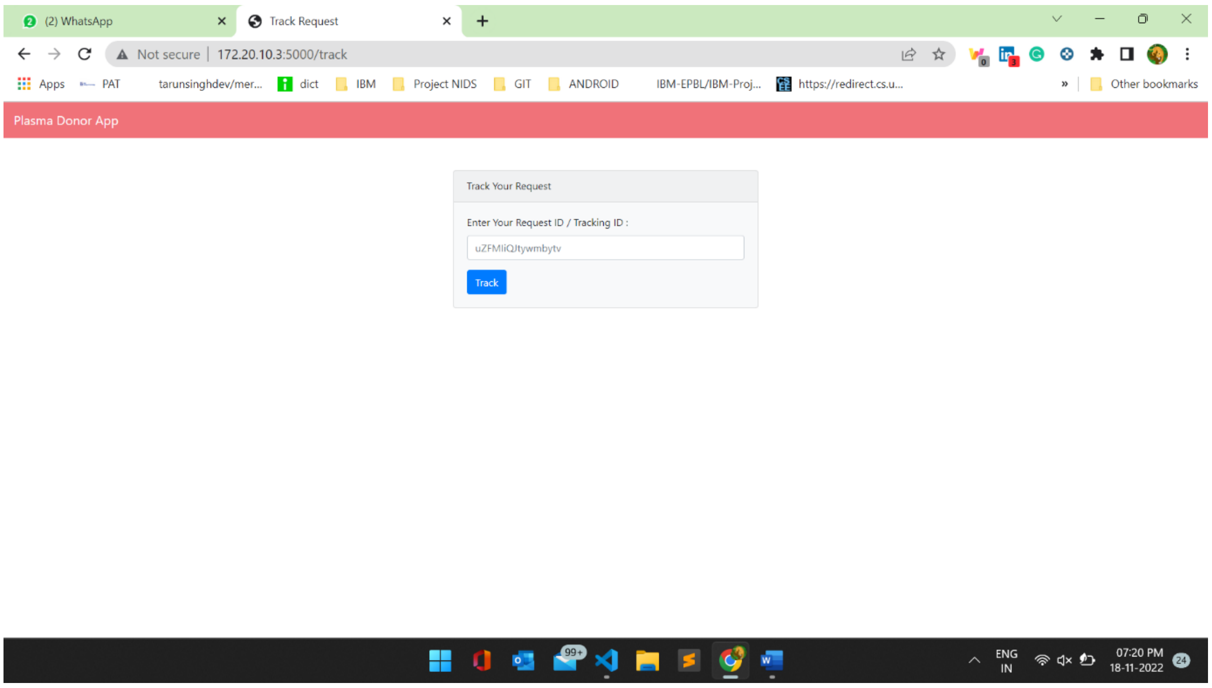
Plasma Donor App

Track the Status of Any Request

Request ID	CvqkXmHMqv8m2vTE
Request STATUS	ACCEPTED BY DONOR
Recipient Name	keshav
Recipient Age	21
Recipient Email	19cs120@psgitech.ac.in
Recipient Phone	+91 08106826278
Requested Blood Type	
Locality	neelambur
Postal Code	641062
Contact Address	coimbatore

Please Check your Mail for Any Appointment by the Donor

Fig 7.2.1



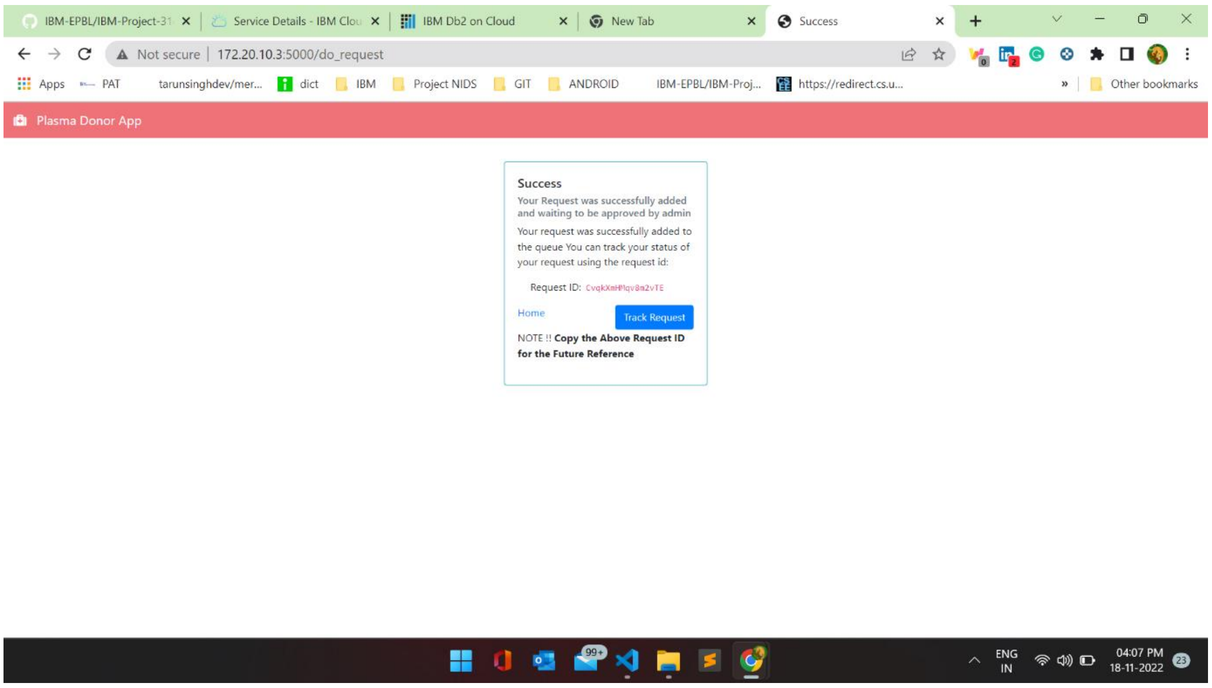
Track Your Request

Enter Your Request ID / Tracking ID :

uZFMIiQJtywmbytv

Track

Fig 7.2.2



Success

Your Request was successfully added and waiting to be approved by admin

Your request was successfully added to the queue You can track your status of your request using the request id:

Request ID: CvqkXmHMqv8m2vTE

Home

Track Request

NOTE !! **Copy the Above Request ID for the Future Reference**

Fig 7.2.3

Code :

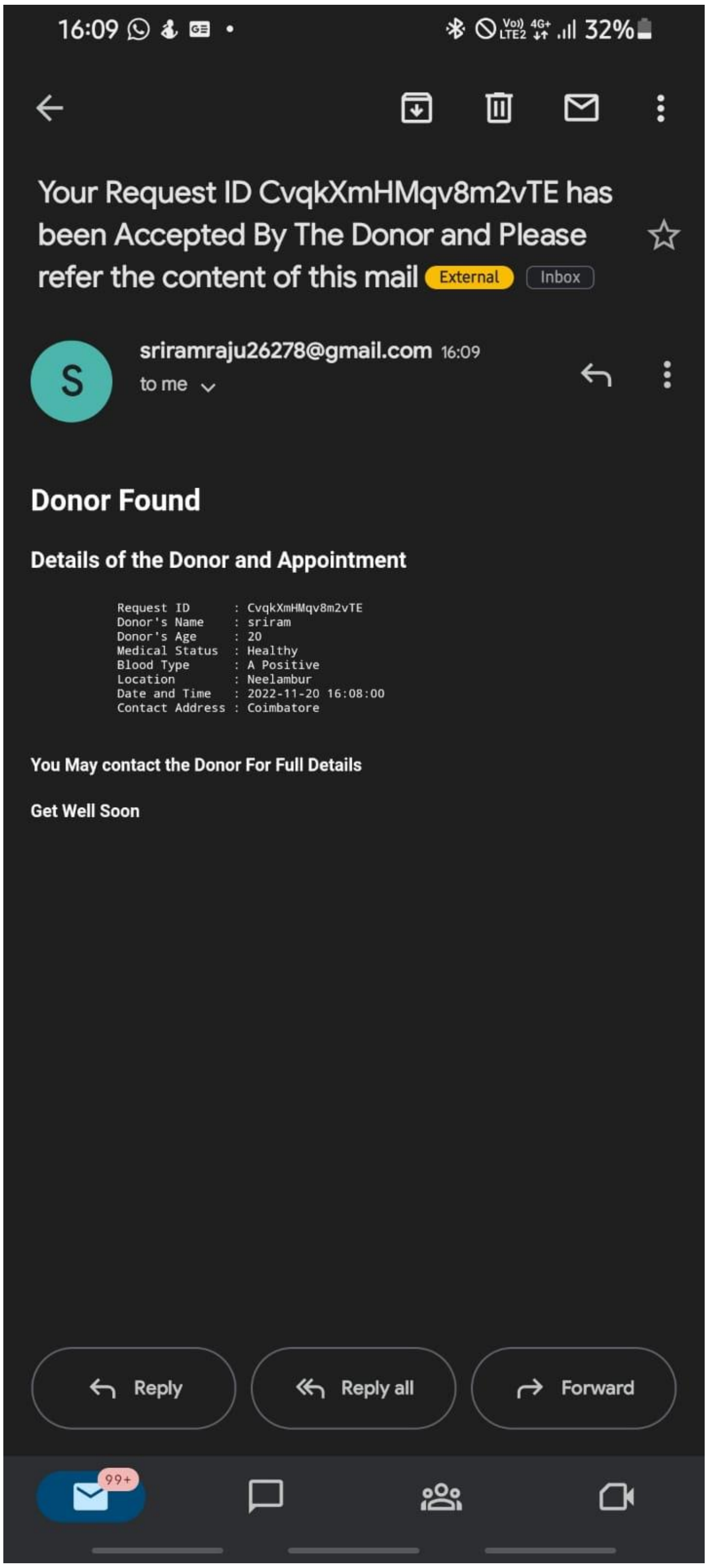
Fetch.py:

```
from dotenv import load_dotenv
import os
import ibm_db



def fetch_home(conn):
    sql = "SELECT COUNT(*) , (SELECT COUNT(*) FROM DONORS WHERE
blood_type= 'A Positive'),"
    sql += "(SELECT COUNT(*) FROM DONORS WHERE blood_type='A Negative'),
(SELECT COUNT(*) FROM DONORS WHERE blood_type='B Positive'),"
    sql += "(SELECT COUNT(*) FROM DONORS WHERE blood_type='B Negative'),
(SELECT COUNT(*) FROM DONORS WHERE blood_type='O Positive'),"
    sql += "(SELECT COUNT(*) FROM DONORS WHERE blood_type='O Negative'),
(SELECT COUNT(*) FROM DONORS WHERE blood_type='AB Positive'),"
    sql += "(SELECT COUNT(*) FROM DONORS WHERE blood_type='AB Negative')
from donors"

    req_sql = "SELECT COUNT(*) FROM REQUESTS WHERE REQUEST_STATUS !=
'ACCEPTED'"
    req_stmt = ibm_db.prepare(conn,req_sql)
    ibm_db.execute(req_stmt)
    req = ibm_db.fetch_assoc(req_stmt)
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.execute(stmt)
    res = ibm_db.fetch_assoc(stmt)
    return req,res
```

7.3 Mailer

An additional feature added to the plasma donor application is the mailer. It send the details about the appointments and other information to the donors, doctors and the recipients.

Screenshots:



16:09 32%

Your Request ID CvqkXmHMqv8m2vTE has been Accepted By The Donor and Please refer the content of this mail External Inbox

sriramraju26278@gmail.com 16:09
to me

Donor Found

Details of the Donor and Appointment

```
Request ID       : CvqkXmHMqv8m2vTE
Donor's Name     : sriram
Donor's Age      : 20
Medical Status   : Healthy
Blood Type       : A Positive
Location         : Neelambur
Date and Time    : 2022-11-20 16:08:00
Contact Address  : Coimbatore
```

You May contact the Donor For Full Details

Get Well Soon

Fig 7.3.1

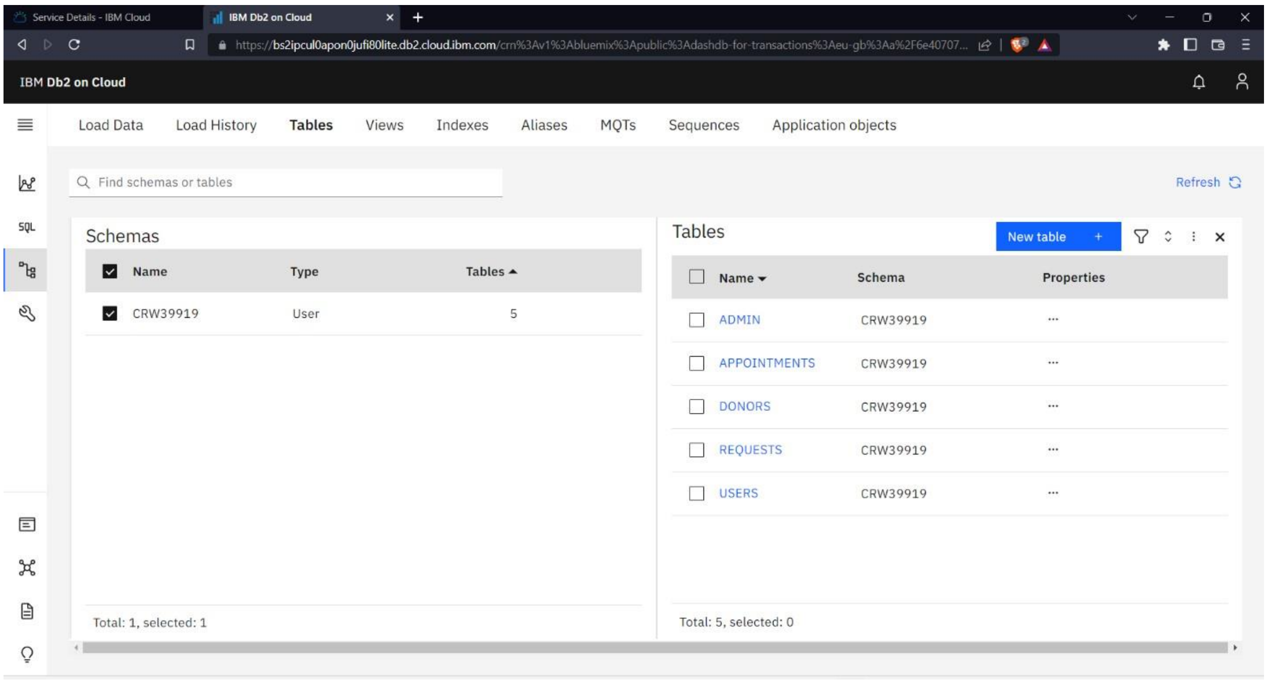
Code:

mailer.py:

```
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
from dotenv import load_dotenv
import os

load_dotenv()


def send_the_email(to_email,subject,html_content):
    message = Mail(from_email='sriramraju26278@gmail.com',
    to_emails=to_email,subject=subject,
    html_content=html_content)

    try:
        sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
        response = sg.send(message)
        print(response.status_code)
        print(response.body)
        print(response.headers)
        return
    except Exception as e:
        print(e.message)
        return
```

Database Schema:

For Database, IBM Cloud DB2 instance is used. The following images explain the manner in which the databases are instantitated.


Service Details - IBM Cloud
IBM Db2 on Cloud
https://bs2ipcul0apon0jufi80lite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aeu-gb%3Aa%2F6e40707...
IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
New table

Schemas

Name	Type	Tables
CRW39919	User	5

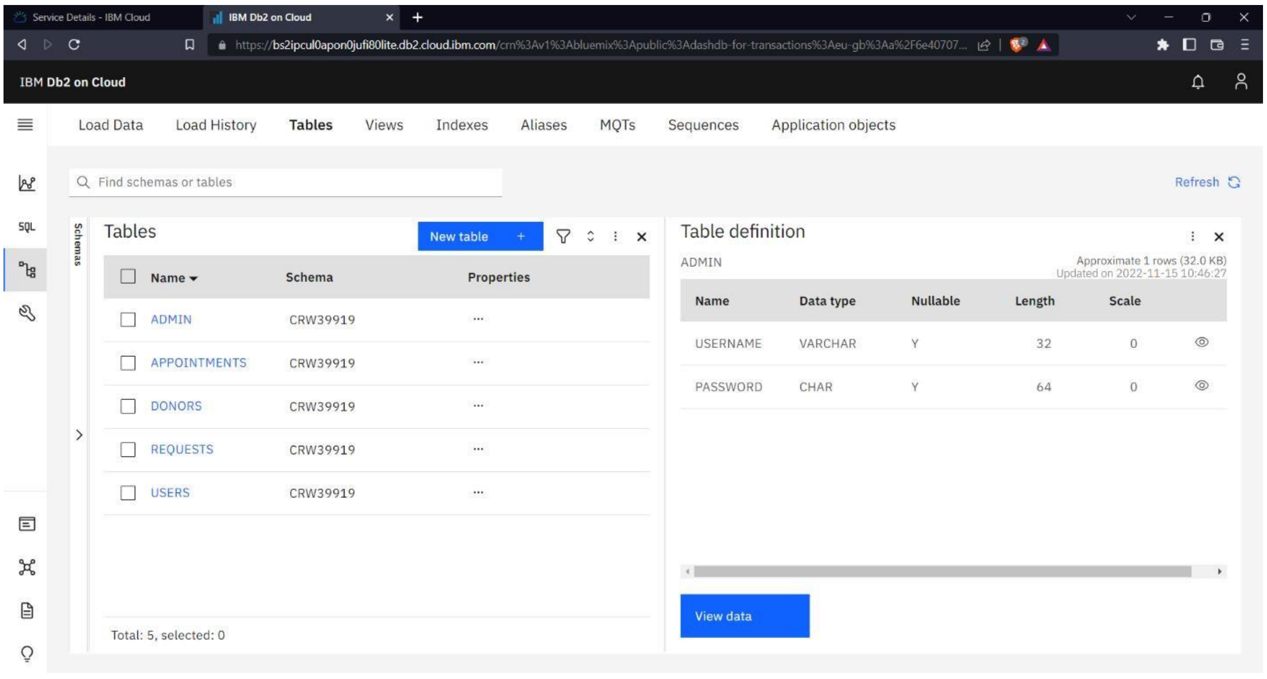
Total: 1, selected: 1

Tables

Name	Schema	Properties
ADMIN	CRW39919	...
APPOINTMENTS	CRW39919	...
DONORS	CRW39919	...
REQUESTS	CRW39919	...
USERS	CRW39919	...

Total: 5, selected: 0

Fig 7.4.1



Tables

Name	Schema	Properties
ADMIN	CRW39919	...
APPOINTMENTS	CRW39919	...
DONORS	CRW39919	...
REQUESTS	CRW39919	...
USERS	CRW39919	...

Total: 5, selected: 0

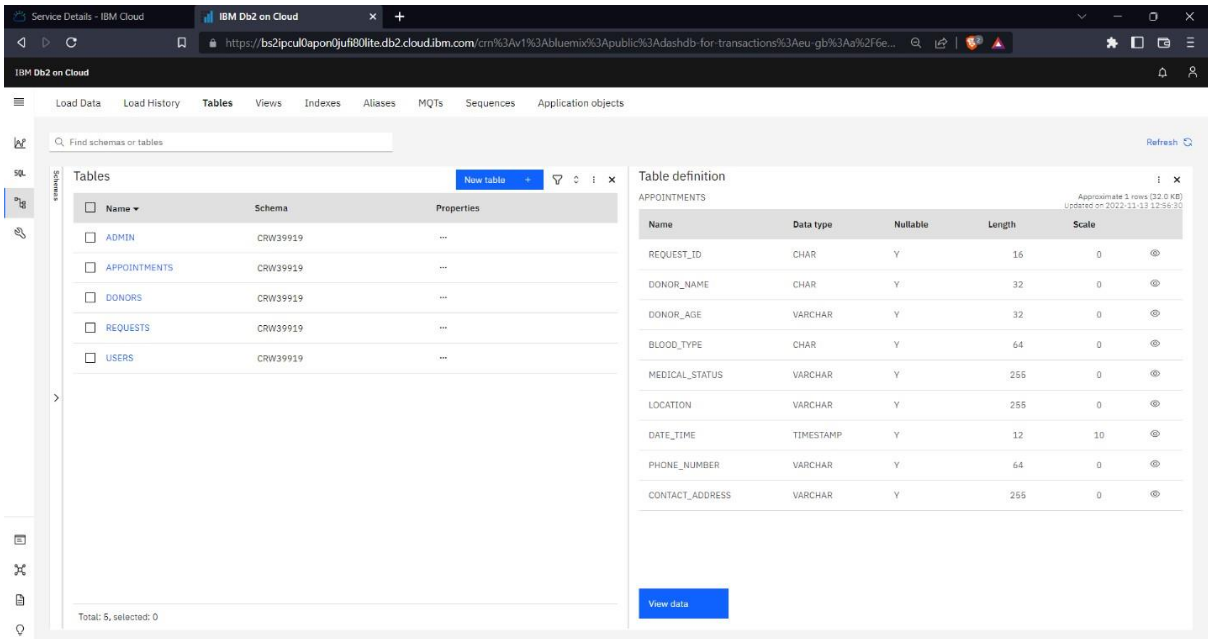
Table definition

ADMIN

Approximate 1 rows (32.0 KB)
Updated on 2022-11-15 10:46:27

Name	Data type	Nullable	Length	Scale
USERNAME	VARCHAR	Y	32	0
PASSWORD	CHAR	Y	64	0

Fig 7.4.2



Tables

Name	Schema	Properties
ADMIN	CRW39919	···
APPOINTMENTS	CRW39919	···
DONORS	CRW39919	···
REQUESTS	CRW39919	···
USERS	CRW39919	···

Total: 5, selected: 0

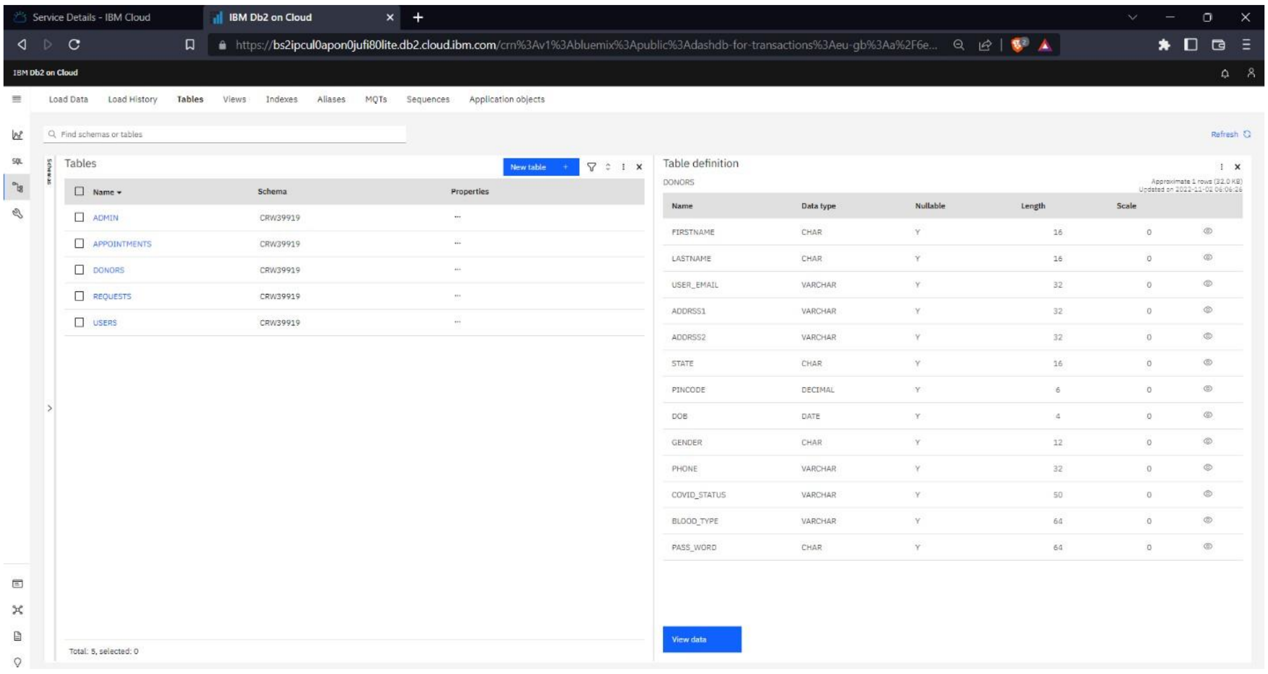
Table definition

APPOINTMENTS

Approximate 1 rows (32.0 KB)
Updated on 2022-11-13 12:56:30

Name	Data type	Nullable	Length	Scale
REQUEST_ID	CHAR	Y	16	0
DONOR_NAME	CHAR	Y	32	0
DONOR_AGE	VARCHAR	Y	32	0
BLOOD_TYPE	CHAR	Y	64	0
MEDICAL_STATUS	VARCHAR	Y	255	0
LOCATION	VARCHAR	Y	255	0
DATE_TIME	TIMESTAMP	Y	12	10
PHONE_NUMBER	VARCHAR	Y	64	0
CONTACT_ADDRESS	VARCHAR	Y	255	0

Fig 7.4.3



Tables

Name	Schema	Properties
ADMIN	CRW39919	···
APPOINTMENTS	CRW39919	···
DONORS	CRW39919	···
REQUESTS	CRW39919	···
USERS	CRW39919	···

Total: 5, selected: 0

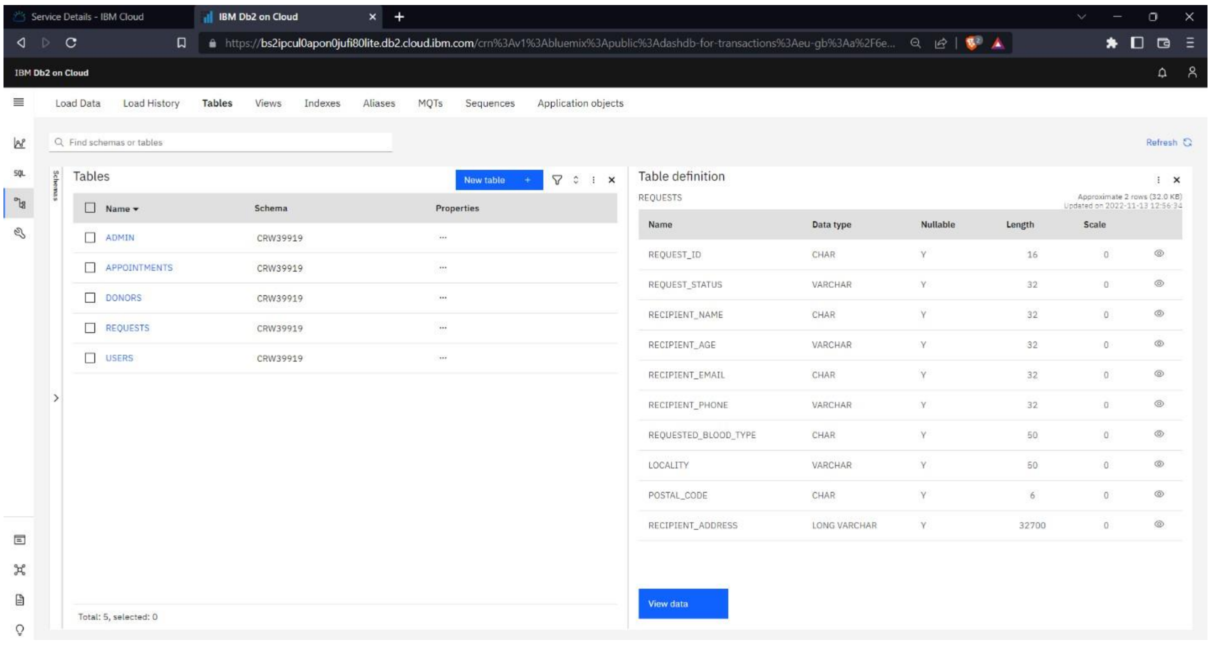
Table definition

DONORS

Approximate 1 rows (32.0 KB)
Updated on 2022-11-02 06:06:26

Name	Data type	Nullable	Length	Scale
FIRSTNAME	CHAR	Y	16	0
LASTNAME	CHAR	Y	16	0
USER_EMAIL	VARCHAR	Y	32	0
ADDRSS1	VARCHAR	Y	32	0
ADDRSS2	VARCHAR	Y	32	0
STATE	CHAR	Y	16	0
PINCODE	DECIMAL	Y	6	0
DOB	DATE	Y	4	0
GENDER	CHAR	Y	12	0
PHONE	VARCHAR	Y	32	0
COVID_STATUS	VARCHAR	Y	50	0
BLOOD_TYPE	VARCHAR	Y	64	0
PASS_WORD	CHAR	Y	64	0

Fig 7.4.4



Tables

Name	Schema	Properties
ADMIN	CRW39919	···
APPOINTMENTS	CRW39919	···
DONORS	CRW39919	···
REQUESTS	CRW39919	···
USERS	CRW39919	···

Total: 5, selected: 0

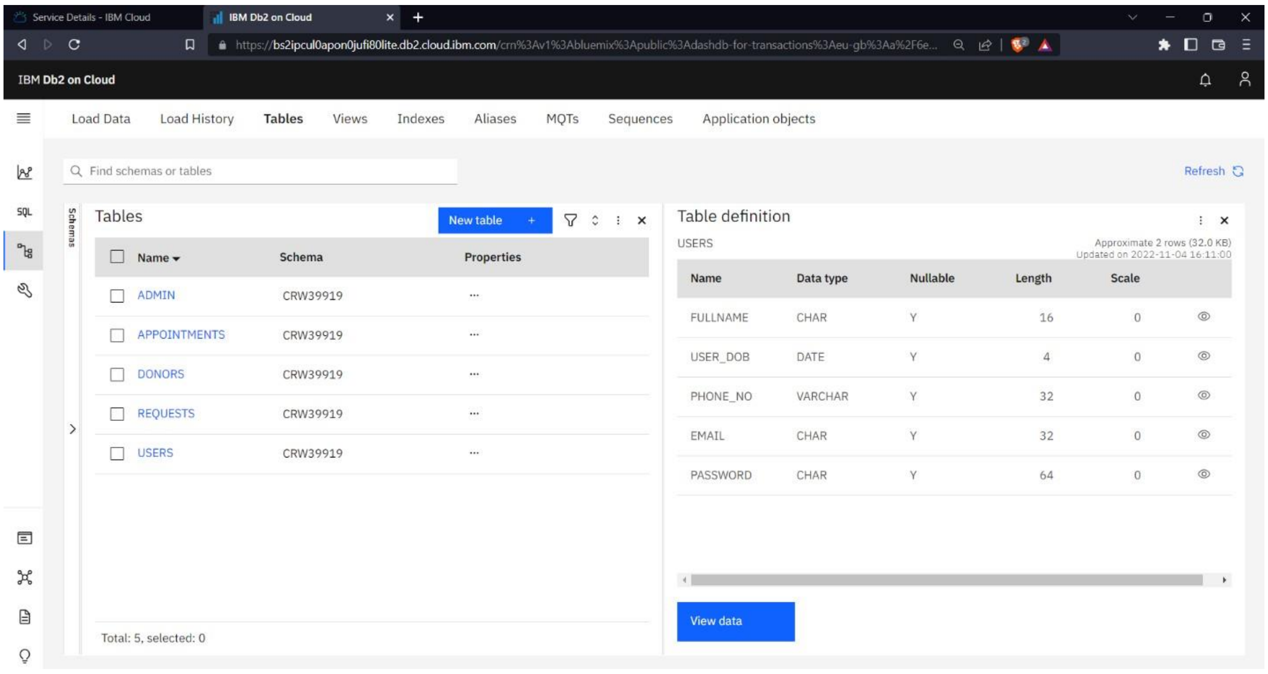
Table definition

REQUESTS

Approximate 2 rows (32.0 KB)
Updated on 2022-11-13 12:56:34

Name	Data type	Nullable	Length	Scale
REQUEST_ID	CHAR	Y	16	0
REQUEST_STATUS	VARCHAR	Y	32	0
RECIPIENT_NAME	CHAR	Y	32	0
RECIPIENT_AGE	VARCHAR	Y	32	0
RECIPIENT_EMAIL	CHAR	Y	32	0
RECIPIENT_PHONE	VARCHAR	Y	32	0
REQUESTED_BLOOD_TYPE	CHAR	Y	50	0
LOCALITY	VARCHAR	Y	50	0
POSTAL_CODE	CHAR	Y	6	0
RECIPIENT_ADDRESS	LONG VARCHAR	Y	32700	0

Fig 7.4.5



Tables

Name	Schema	Properties
ADMIN	CRW39919	···
APPOINTMENTS	CRW39919	···
DONORS	CRW39919	···
REQUESTS	CRW39919	···
USERS	CRW39919	···

Total: 5, selected: 0

Table definition

USERS

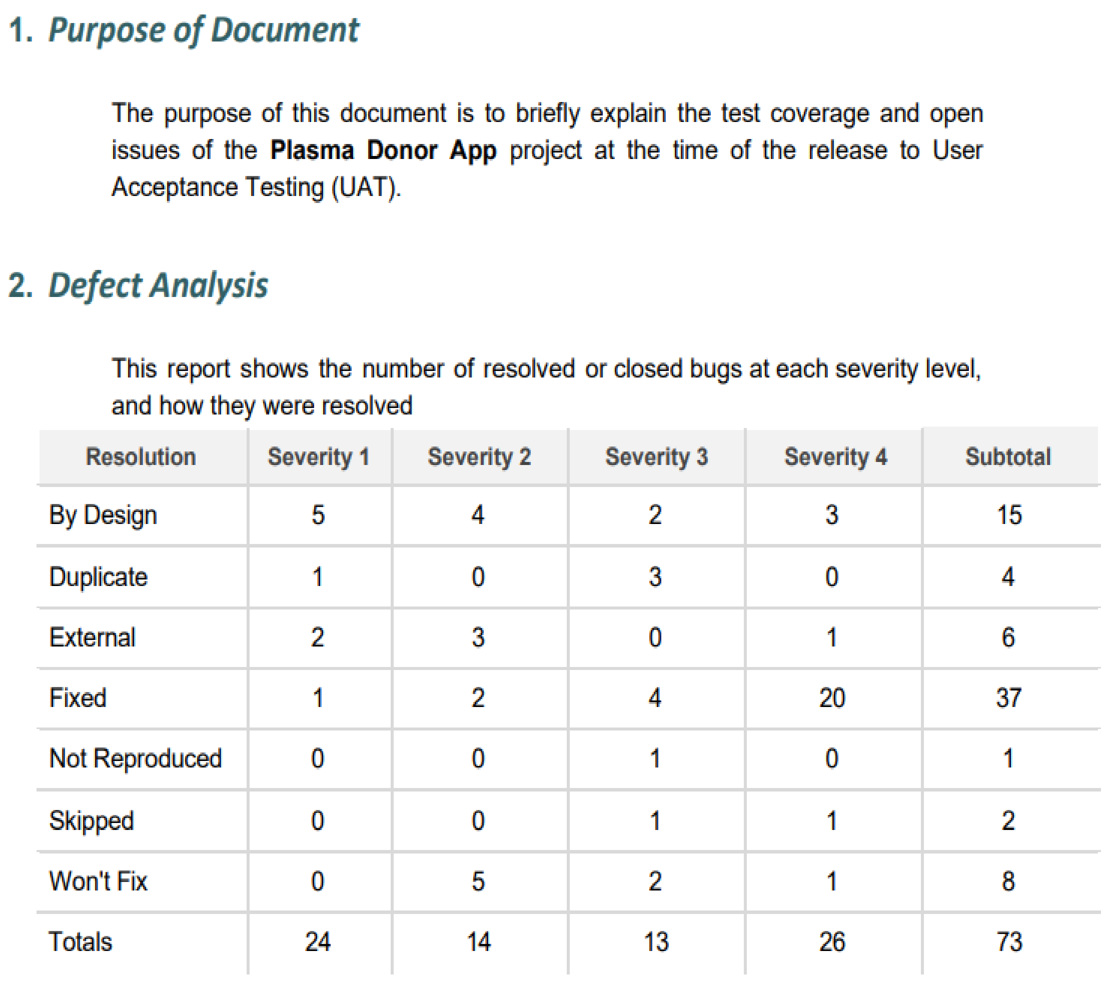
Approximate 2 rows (32.0 KB)
Updated on 2022-11-04 16:11:00

Name	Data type	Nullable	Length	Scale
FULLNAME	CHAR	Y	16	0
USER_DOB	DATE	Y	4	0
PHONE_NO	VARCHAR	Y	32	0
EMAIL	CHAR	Y	32	0
PASSWORD	CHAR	Y	64	0

Fig 7.4.6

CHAPTER 8: TESTING

8.1 User acceptance testing



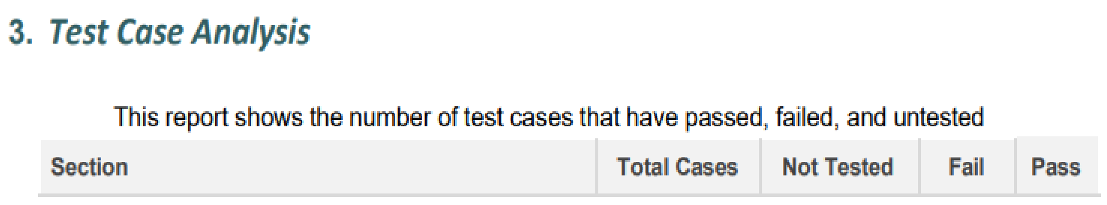
1. *Purpose of Document*

The purpose of this document is to briefly explain the test coverage and open issues of the **Plasma Donor App** 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	Subtotal
By Design	5	4	2	3	15
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	1	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	73



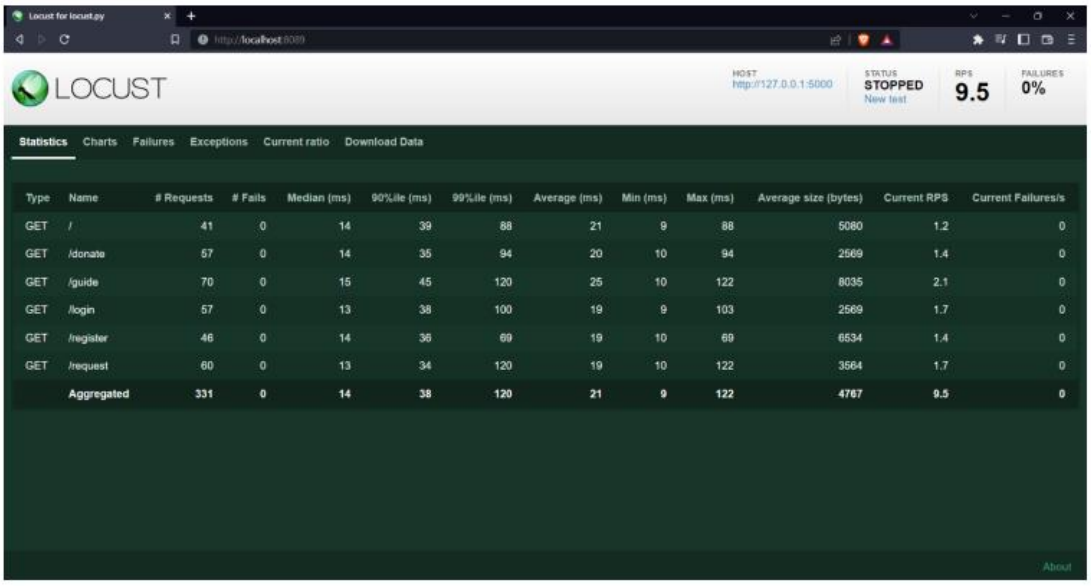
3. *Test Case Analysis*

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

Section	Total Cases	Not Tested	Fail	Pass

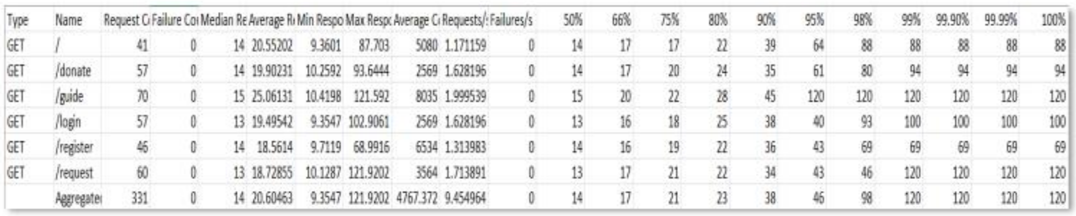
CHAPTER 9: RESULTS

9.1 Performance Metrics:



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	/	41	0	14	39	88	21	9	88	5080	1.2	0
GET	/donate	57	0	14	35	94	20	10	94	2569	1.4	0
GET	/guide	70	0	15	45	120	25	10	122	8035	2.1	0
GET	/login	57	0	13	38	100	19	9	103	2569	1.7	0
GET	/register	46	0	14	36	69	19	10	69	6534	1.4	0
GET	/request	60	0	13	34	120	19	10	122	3564	1.7	0
	Aggregated	**331**	**0**	**14**	**38**	**120**	**21**	**9**	**122**	**4767**	**9.5**	**0**

Fig.9.1.1



Type	Name	Request C	Failure Co	Median Re	Average R	Min Respo	Max Respc	Average C	Requests/	Failures/s	50%	66%	75%	80%	90%	95%	98%	99%	99.90%	99.99%	100%
GET	/	41	0	14	20.55202	9.3601	87.703	5080	1.171159	0	14	17	17	22	39	64	88	88	88	88	88
GET	/donate	57	0	14	19.90231	10.2592	93.6444	2569	1.628196	0	14	17	20	24	35	61	80	94	94	94	94
GET	/guide	70	0	15	25.06131	10.4198	121.592	8035	1.999539	0	15	20	22	28	45	120	120	120	120	120	120
GET	/login	57	0	13	19.49542	9.3547	102.9061	2569	1.628196	0	13	16	18	25	38	40	93	100	100	100	100
GET	/register	46	0	14	18.5614	9.7119	68.9916	6534	1.313983	0	14	16	19	22	36	43	69	69	69	69	69
GET	/request	60	0	13	18.72855	10.1287	121.9202	3564	1.713891	0	13	17	21	22	34	43	46	120	120	120	120
	Aggregate	331	0	14	20.60463	9.3547	121.9202	4767.372	9.454964	0	14	17	21	23	38	46	98	120	120	120	120

Fig.9.2.1

CHAPTER 10: ADVANTAGES AND DISADVANTAGES

Advantages

- **Compatibility**
    - Since the application is purely web-based, the user is able to access the application from any kind of device. Hence it provides cross-platform compatibility for the users.
    -
- **Speed**
    - The application is completely light-weight and can able to response much faster and provides user with real-time experience.
- **Amazing UI**
    - The users can able to find its very easy to use and they can also smooth experience while using the application.
- **Scalability**
    - Since the application is developed using the micro-services architecture which provides vertical scaling the application can able to grow and shrink on its own based on the traffic

Disadvantages

- **Self-Verification**
    - The application cannot have the capability of distinguish between the fake user and genuine user on its own. It demands the admin work to getting things done

CHAPTER 11: CONCLUSION

The number of vaccines produced is not enough for all the population to get vaccinated at present. And with the corona positive cases rising every day, saving lives has become the prime matter of concern. As per the data provided by WHO more than 3 million people have died due to the coronavirus. However, apart from vaccination, there is another scientific method by which a covid infected person can be treated and the death risk can be reduced. A person who has recovered from Covid can donate his/her plasma to a person who is infected with the coronavirus. This system proposed here aims at connecting the donors & the patients by an online application. By using this application, the users can either raise a request for plasma donation or requirement. Both parties can Accept or Reject the request. User has to Upload a Covid Negative report to be able to Donate Plasma. This system is used if anyone needs a Plasma Donor Blood and Plasma donation is a kind of citizen's social responsibility in which an individual can willingly donate blood/plasma via our app. This Application has been created with the concept and has sought to make sure that the donor gives plasma to the community. This model is made user friendly so anybody can view and maintain his/her account. This application will break the chain of business through blood/plasma and help the poor to find donors at free of cost.

CHAPTER 12: FUTURE WORK

User interface (UI) can be improved in future to accommodate a global audience by supporting different languages across countries. Appointments can be synchronised with Google and Outlook calendars for the ease of users and for improvement.

Improving the accessibility via integrating this application with various social networks application program interfaces (APIs). Consequently, users can login and sign up using various social networks. This would increase the number of donors and enhance the process of blood donation.

Donors will be able to view and share personal experiences about their donation; Beneficiaries can share their experiences of receiving blood transfusion which contributed to their improved health and lives.

CHAPTER 13: APPENDIX

13.1 Source code:

Index.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Plasma Donor App</title>
   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.14.0/css/all.css">
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<body style="background-image: url('https://img.freepik.com/free-vector/donate-blood-
healthcare-medical-promo-design_1017-
26842.jpg?w=900&t=st=1668712331~exp=1668712931~hmac=f07053f1a9283470e6db859f
9dafd42ebb5a39fe4bef3b021cbc9e687a120c97');width: 100vw;
height: 100vh;
background-size: 100% 100%;
background-repeat: no-repeat;
position: relative; background-color: rgba(0, 0, 0, 0.2);
  background-blend-mode: multiply;">
   <div class="header navbar-wrapper">
      <nav class="navbar navbar-dark navbar-expand-sm fixed-top" style="background-
color:#f07279;box-shadow: 20px;">
         <div class="container">
         <a href="/" class="navbar-brand" style="font-size: xx-large;">
         <i class="fas fa-heartbeat"></i> &nbsp;
         Plasma Donor App
         </a>
```

```
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarCollapse">
          <span class="navbar-toggler-icon"></span>
        </button>


        <div id="navbarCollapse" class="collapse navbar-collapse">
        <ul class="navbar-nav ml-auto">
          <li class="nav-item">
            <a href="/home" class="nav-link active">
              Home
            </a>
          </li>

          <li class="nav-item">
            <a href="#" class="mr-3 nav-link active">
              Help
            </a>
          </li>
          <li class="nav-item">
            <a href="/request" class="ml-1 mr-3 btn btn-warning">
              Request
            </a>
          </li>
          <li class="nav-item">
            <a href="/login" class="ml-2 mr-2 btn btn-dark">
              Log in
            </a>
          </li>
          <li class="nav-item">
            <a href="/register" class="ml-2 btn btn-outline-dark text-white">
              Register
            </a>
          </li>
```

```
            </ul>
            </div>
          </div>
        </div>

    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
</body>
</html>
```

Account.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Account</title>
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.14.0/css/all.css">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<style>
    main{
        margin-top: 80px;
    }
</style>
<body>
    <header>
        <div class="header navbar-wrapper">
            <nav class="navbar navbar-dark navbar-expand-sm fixed-top" style="background-
color:#f07279;box-shadow: 20px;">
                <div class="container">
                <a href="/" class="navbar-brand" style="font-size: xx-large;">
                <i class="fas fa-heartbeat"></i> &nbsp;
                Plasma Donor App
                </a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarCollapse">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div id="navbarCollapse" class="collapse navbar-collapse">
                <ul class="navbar-nav ml-auto">
                    <li class="nav-item">
                        <a href="/home" class="nav-link ">
                            Home
                        </a>
                    </li>
                    <li class="nav-item ">
                        <a href="/donate" class="nav-link  btn btn-secondary ">
                            Donate
                        </a>
                    </li>
                    <li class="nav-item">
                        <a href="/request" class="nav-link ">
                            Request
                        </a>
                    </li>
                <li class="nav-item">
                    <a href="/about" class="mr-3 nav-link ">
                        Help
                    </a>
                </li>
                <li class="nav-item">
                    <a href="/account" class="nav-link active">
                        Account
                    </a>
                </li>
                <li class="nav-item">
                    <a href="/logout" class="ml-4 mr-2 btn btn-dark">
                        Logout
                    </a>
                </li>
                </ul>
                </div>
            </div>
        </div>
    </header>
    <main>
        <div class="container">
            <div class="card glass-effect" ;>
                <div class="card-header" style="background-color:#f07279;box-shadow: 20px;">
                    Account Details
                </div>
                <div class="card-body bg-light">
                    <div class="card mb-2">
                        <h6 class="card-header">FULL NAME</h6>
                        <div class="card-body">
                         <p class="card-text">
                            {% if session['account-type'] == 'Donor'    %}
                            {{ res['FIRSTNAME']+' '+res['LASTNAME'] }}
                            {% else %}
```

```
          {{res['FULLNAME']}}
          {%  endif %}


        </p>
       </div>
    </div>
    <div class="card mb-2">
       <h6 class="card-header">DATE OF BIRTH (YYYY-MM-DD):</h6>
       <div class="card-body">
        <p class="card-text">
          {% if session['account-type'] == 'Donor'    %}
          {{ res['DOB'] }}
          {% else %}
          {{res['USER_DOB']}}
          {%  endif %}
        </p>
       </div>
    </div>
    <div class="card mb-2">
       <h6 class="card-header">PHONE NUMBER:</h6>
       <div class="card-body">
        <p class="card-text">+91
          {% if session['account-type'] == 'Donor'    %}
          {{ res['PHONE'] }}
          {% else %}
          {{res['PHONE_NO']}}
          {%  endif %}
        </p>
       </div>
    </div>
    <div class="card mb-2">
       <h6 class="card-header">EMAIL:</h6>
       <div class="card-body">
        <p class="card-text">
        {% if session['account-type'] == 'Donor'    %}
        {{ res['USER_EMAIL'] }}
        {% else %}
        {{res['EMAIL']}}
        {%  endif %}
     </p>
    </div>
</div>
{% if session['account-type'] == 'Donor' %}
<div class="card mb-2">
    <h6 class="card-header">COVID STATUS:</h6>
    <div class="card-body">
     <p class="card-text">
        {{ res['COVID_STATUS'] }}
     </p>
    </div>
</div>
<div class="card mb-2">
    <h6 class="card-header">BLOOD TYPE:</h6>
    <div class="card-body">
     <p class="card-text">
        {{ res['BLOOD_TYPE'] }}
     </p>
    </div>
</div>
<div class="card mb-2">
    <h6 class="card-header">PINCODE</h6>
    <div class="card-body">
     <p class="card-text">
        {{ res['PINCODE'] }}
     </p>
    </div>
</div>
<div class="card mb-2">
```

```
                    <h6 class="card-header">STATE</h6>
                    <div class="card-body">
                     <p class="card-text">
                        {{ res['STATE'] }}
                     </p>
                    </div>
                </div>
                {% endif %}
                <div class="card mb-2">
                    <h6 class="card-header">Is this account Donor or Not Donor?</h6>
                    <div class="card-body">
                     <p class="card-text">
                        {% if session['account-type'] == 'Donor': %}
                        <span class="badge badge-success">Donor</span>
                        {% else %}
                        <span class="badge badge-danger">Not Donor</span></p>
                        {% endif %}
                    </div>
                </div>
                <!-- Button trigger modal -->
<button type="button" class="btn btn-danger float-right" data-toggle="modal" data-
target="#exampleModal">
    Delete Account
  </button>

  <!-- Modal -->
  <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-
labelledby="exampleModalLabel" aria-hidden="true">
    <div class="modal-dialog" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalLabel">Confirmation Required</h5>
          <button type="button" class="close" data-dismiss="modal" aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
        <div class="modal-body">
          Are you really want to delete your account ?
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-secondary" data-
dismiss="modal">Cancel</button>
          <a href='/home'><button type="button"  class="btn btn-danger">Delete
Now</button></a>
        </div>
      </div>
    </div>
  </div>
                </div>
            </div>
        </div>
    </main>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
</body>
</body>
</html>
```

Donate.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Donate</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<style>
  main{
    margin-top: 90px;
  }
</style>
<body>
  <nav class="navbar navbar-dark navbar-expand-sm fixed-top" style="background-
color:#f07279;box-shadow: 20px;">
    <a href="/" class="navbar-brand">
      Plasma Donor App
    </a>
  </nav>
  <main>
    <div class="container">
      <table class="table table-borderless table-responsive-md">
        <thead>
          <tr class="bg-danger">
            <th scope="col">No.</th>
            <th scope="col">Recipient Name</th>
            <th scope="col">Age</th>
            <th scope="col">Locality</th>
            <th scope="col">Requested Blood type</th>
```

```
                <th scope="col">Request Status</th>
                <th scope="col">Action</th>
              </tr>
            </thead>
            <tbody>
                {% for key,result in results.items(): %}
                <tr class="table-active">
                <td>{{key}}</td>
                <td>{{result['RECIPIENT_NAME']}}</td>
                <td>{{result['RECIPIENT_AGE']}}</td>
                <td>{{result['LOCALITY']}}</td>
                <td>{{result['REQUESTED_BLOOD_TYPE']}}</td>
                <td>{{result['REQUEST_STATUS']}}</td>
                <td><a class="btn btn-primary"
                   href="/BookAppointment/{{result['REQUEST_ID']}}" >Donate</a></td>
                </tr>
                {% endfor %}
            </tbody>
          </table>
          </div>
    </main>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
</body>
</html>
```

Home.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Home - Plasma Donor App</title>
   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.14.0/css/all.css">
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
   <link rel="stylesheet" href="{{ url_for('static', filename='dash.css') }}">
</head>
<body style="background-color: #f1f1f1d6;">
   <header>
   <div class="header navbar-wrapper">
      <nav class="navbar navbar-dark navbar-expand-sm fixed-top" style="background-
color:#f07279;box-shadow: 20px;">
         <div class="container">
         <a href="/" class="navbar-brand" style="font-size: xx-large;">
         <i class="fas fa-heartbeat"></i> &nbsp;
         Plasma Donor App
         </a>
         <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarCollapse">
            <span class="navbar-toggler-icon"></span>
         </button>
         <div id="navbarCollapse" class="collapse navbar-collapse">
         <ul class="navbar-nav ml-auto">
            <li class="nav-item">
               <a href="/home" class="nav-link active">
                  Home
```

```
          </a>
        </li>
        <li class="nav-item ">
          <a href="/donate" class="nav-link active btn btn-secondary ">
            Donate
          </a>
        </li>
        <li class="nav-item">
          <a href="/request" class="nav-link active">
            Request
          </a>
        </li>

        <li class="nav-item">
          <a href="/account" class="nav-link active">
            Account
          </a>
        </li>
        <li class="nav-item">
          <a href="/logout" class="ml-4 mr-2 btn btn-dark">
            Logout
          </a>
        </li>
        </ul>
        </div>
    </div>
</div>
</header>
<main>
  <div class="container">
          <div class="card glass-effect" style="background-color: #A5D8DD;">
            <div class="card-body">
              <h3 class="text-content"><i class="fas fa-home"></i>
                <span class="pl-3">Welcome, {{username}}</span>
```

```
                    {% if session['account-type'] == 'Donor' %}
                    <p class="badge badge-success">DONOR</p>
                    {% endif %}
                  </h3>
                </div>
              </div>
          <div class="row">
            <div class="col-6 mt-3">
              <div class="card mb-3 glass-effect" style="background-color: #DBAE58;">
                <div class="card-body" style="color: #fff;">
                  <div class="card-title">
                    <h3 style="font-style: bold;">Dashboard</h3>
                  </div>
                  <p class="card-text">
                    In Dashboard, you can able to see the current statistics of the
                    donors count, no of requests and availability of plasma in each blood
type.
                  </p>
                  <a href="/track"><button class="btn btn-danger">Track</button></a>
                </div>
              </div>
            </div>
            <div class="col-3  mt-3  ">
              <div class="card mb-3 glass-effect " style="background-color:#1C4E80 ;">

                <div class="card-body">
                  <!-- <i class="fa fa-tint fa-lg"></i> -->
                   <img src="/static/imgs/blood-drop.png" width="10%">
                  <div class="card-title">
                    <h3 class="text-center">No.of Donors</h3>
                    <span class="card-text"><h4>{{res['1']}}</h4></span>
                  </div>
                </div>
              </div>
            </div>
            <div class="col-3 mt-3">
                <div class="card card mb-3 glass-effect " style="background-color:
#EA6A47;">

                    <div class="card-body">
                        <img src="/static/imgs/request.png" width="10%">
                        <div class="card-title">
                            <h3 class="text-center">
                                No. of Requests
                            </h3>
                            <span class="card-text"><h4>{{req['1']}}</h4></span>
                        </div>
                    </div>
                </div>
            </div>
        </div>
        <div class="card glass-effect"style="background-color: #0091d5;">
            <div class="card-body">
                <h4 class="text-center">
                    Plasma Availability for Every Blood type
                </h4>
            </div>
        </div>
        <div class="row">
            <div class="col-3 mt-3">
                <div class="card glass-effect">
                    <div class="card-body">
                        <div class="card-title">A postive (A+)</div>
                        <h3 class="card-text">{{res['2']}}</h3>
                    </div>
                </div>
            </div>
            <div class="col-3 mt-3">
        <div class="card glass-effect">
            <div class="card-body">
                <div class="card-title">A Negative (A-)</div>
                <h3 class="card-text">{{res['3']}}</h3>
            </div>
        </div>
    </div>
    <div class="col-3 mt-3">
        <div class="card glass-effect">
            <div class="card-body">
                <div class="card-title">B positive (B+)</div>
                <h3 class="card-text">{{res['4']}}</h3>
            </div>
        </div>
    </div>
    <div class="col-3 mt-3">
        <div class="card glass-effect ">
            <div class="card-body">
                <div class="card-title">B postive (B-)</div>
                <h3 class="card-text">{{res['5']}}</h3>
            </div>
        </div>
    </div>
</div>
<div class="row">
    <div class="col-3 mt-3">
        <div class="card glass-effect">
            <div class="card-body">
                <div class="card-title">O postive (O+)</div>
                <h3 class="card-text">{{res['6']}}</h3>
            </div>
        </div>
    </div>
    <div class="col-3 mt-3 mb-3">
```

```
                <div class="card glass-effect ">
                    <div class="card-body">
                        <div class="card-title">O Negative (O-)</div>
                        <h3 class="card-text">{{res['7']}}</h3>
                    </div>
                </div>
            </div>
            <div class="col-3 mt-3 mb-3">
                <div class="card glass-effect">
                    <div class="card-body">
                        <div class="card-title">AB positive (AB+)</div>
                        <h3 class="card-text">{{res['8']}}</h3>
                    </div>
                </div>
            </div>
            <div class="col-3 mt-3 mb-3">
                <div class="card glass-effect">
                    <div class="card-body">
                        <div class="card-title">AB postive (B-)</div>
                        <h3 class="card-text">{{res['9']}}</h3>
                    </div>
                </div>
            </div>
        </div>
      </div>
    </main>
  <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
```

```
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
</body>
</html>
```

Login.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
    <nav class="navbar navbar-dark navbar-expand-sm fixed-top" style="background-
color:#f07279;box-shadow: 20px;">
        <a href="/" class="navbar-brand"  style="font-size: xx-large;">
            <i class="fas fa-heartbeat"></i> &nbsp;
            Plasma Donor App
        </a>
    </nav>
        <div class="global-container">
            <div class="card login-form">
            <div class="card-body">
                <h3 class="card-title text-center">Log in to App</h3>
                <div class="card-text">
                    <form action="{{url_for('do_login')}}" method="post">
```

```
                        <div class="form-group">
                            <label for="email">Email address</label>
                            <input type="email" name="user_email" class="form-control form-control-
sm" id="InputEmail1" aria-describedby="emailHelp">
                        </div>
                        <div class="form-group">
                            <label for="password">Password</label>
                            <a href="#" style="float:right;font-size:12px;">Forgot password?</a>
                            <input type="password" name="password" class="form-control form-
control-sm" id="InputPassword1">
                        </div>
                        <button type="submit" class="btn btn-primary btn-block">Sign in</button>

                        <div class="sign-up">
                            Don't have an account? <a href="/register">Register Now</a>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
</body>
</html>
```

Register.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Donor Registration</title>
   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.14.0/css/all.css">
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
   <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}">
</head>
<body>
   <div class="nav">
      <nav class="navbar navbar-dark navbar-expand-sm fixed-top" style="background-
color:#f07279;box-shadow: 20px;">
         <a href="/" class="navbar-brand" style="font-size: xx-large;">
             <i class="fas fa-heartbeat"></i> &nbsp;
            Plasma Donor App
      </a>
      </nav>
   </div>

   <div class="container bg-light">
      <form action="{{url_for('do_register')}}" method="post">
            <h4 class="text-center ">New Registration</h4>
            <h5 class="text-center">(Register as Donor)</h5>
         <div class="row p-2">
            <div class="col-sm-6 form-group">
               <label for="name-f">First Name</label>
```

```
                    <input type="text" class="form-control" name="fname" id="name-f"
placeholder="Enter your first name." required>
                </div>
                <div class="col-sm-6 form-group">
                    <label for="name-l">Last name</label>
                    <input type="text" class="form-control" name="lname" id="name-l"
placeholder="Enter your last name." required>
                </div>
                <div class="col-sm-6 form-group">
                    <label for="email">Email</label>
                    <input type="email" class="form-control" name="email" id="email"
placeholder="Enter your email." required>
                </div>
                <div class="col-sm-6 form-group">
                    <label for="address-1">Address Line-1</label>
                    <input type="address" class="form-control" name="Locality" id="address-1"
placeholder="Locality/House/Street no." required>
                </div>
                <div class="col-sm-6 form-group">
                    <label for="address-2">Address Line-2</label>
                    <input type="address" class="form-control" name="address" id="address-2"
placeholder="Village/City Name." required>
                </div>
                <div class="col-sm-4 form-group">
                    <label for="State">State</label>
                    <input type="address" class="form-control" name="State" id="State"
placeholder="Enter your state name." required>
                </div>
                <div class="col-sm-2 form-group">
                    <label for="zip">Postal-Code</label>
                    <input type="zip" class="form-control" name="Zip" id="zip"
placeholder="Postal-Code." required>
                </div>
                <div class="col-sm-6 form-group">
```

```
          <label for="Date">Date Of Birth</label>
          <input type="Date" name="dob" class="form-control" id="Date" placeholder=""
required>
        </div>
        <div class="col-sm-6 form-group">
          <label for="sex">Gender</label>
          <select id="sex" name="gender" class="form-control browser-default custom-
select">
          <option value="male">Male</option>
          <option value="female">Female</option>
          <option value="unspesified">Unspecified</option>
        </select>
        </div>
        <div class="col-sm-3 form-group">
          <label for="tel">Phone</label>
          <input type="tel" name="phone" class="form-control" id="tel"
placeholder="Enter Your Contact Number." required>
        </div>
        <div class="col-sm-3 form-group">
          <label for="covid-record">Covid-19 Record:</label>
          <select id="covid-record" name="covid-report" class="form-control browser-
default custom-select">
            <option value="Recovered">Recovered / Tested Negative</option>
            <option value="Uninfected">Uninfected / No Covid History</option>
          </select>
        </div>
        <div class="col-sm-6 form-group">
          <label for="blood-group">Choose your Blood Type:</label>
          <select id="blood-group" name="b-type" class="form-control browser-default
custom-select">
            <option value="A Positive">A postive (A+)</option>
            <option value="A Negative">A Negative (A-)</option>
            <option value="B Positive">B postive (B+)</option>
            <option value="B Negative">B Negative (B-)</option>
```

```
                    <option value="O Positive">O postive (O+)</option>
                    <option value="O Negative">O Negative (O-)</option>
                    <option value="AB Positive">AB postive (AB+)</option>
                    <option value="AB Negative">AB Negative (AB-)</option>
                  </select>
                </div>
                <div class="col-sm-6 form-group">
                  <label for="pass">Password</label>
                  <input type="Password" name="password" class="form-control" id="pass"
placeholder="Enter your password." required>
                </div>
                <div class="col-sm-6 form-group">
                  <label for="pass2">Confirm Password</label>
                  <input type="Password" name="cnf-password" class="form-control" id="cnf-
pass" placeholder="Re-enter your password." required>
                </div>
                <div class="col-sm-12 form-group mb-0">
                  <button class="btn btn-primary float-right" style="background-
color:#2DCFFF;box-shadow: 20px; border: #f07279;">Register</button>
                </div>
            </form>

    </div>







    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
```

```
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
</body>
</html>
```

Request.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Request</title>
    <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.14.0/css/all.css">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<style>
    main{
        margin: 70px 0 0 0 ;
    }
</style>
<body>
    <header>
        <nav class="navbar navbar-dark navbar-expand-sm fixed-top" style="background-
color:#f07279;box-shadow: 20px;">
            <a href="/" class="navbar-brand" style="font-size: xx-large;">
            <i class="fas fa-heartbeat"></i> &nbsp;
            Plasma Donor App
          </a>
        </nav>
      </header>
      <main>
        <div class="container-sm bg-light glass-effect">

            <h3 class="text-center">Request Form</h3>

          <form action="{{url_for('do_request')}}" method="post">
            <div class="form-group">
              <label for="recipient-name">Recipient Name</label>
              <input type="text" class="form-control" name="name"  placeholder="Enter
Recipient Full Name" required>
            </div>
            <div class="form-group">
                <label for="r-age">Enter Recipient Age</label>
                <input type="number" class="form-control" name="age" id="age"
placeholder="Enter Recipient Age" required>
            </div>
            <div class="form-group">
                <label for="email">Email address</label>
                <input type="email" class="form-control" name="email" placeholder="Enter
Recipient's email address" required>
              </div>
              <div class="form-group">
                <label for="phone">Phone No</label>
                <input type="tel" class="form-control" name="phone" placeholder="Enter 10
Digit Phone Number" required>
              </div>
              <div class="form-group">
                <label for="b-type">Choose Blood type</label>
```

```
                    <select id="blood-group" name="blood-type" class="form-control browser-
default custom-select" required>
                      <option value="A Positive">A postive (A+)</option>
                      <option value="A Negative">A Negative (A-)</option>
                      <option value="B Positive">B postive (B+)</option>
                      <option value="B Negative">B Negative (B-)</option>
                      <option value="O Positive">O postive (O+)</option>
                      <option value="O Negative">O Negative (O-)</option>
                      <option value="AB Positive">AB postive (AB+)</option>
                      <option value="AB Negative">AB Negative (AB-)</option>
                    </select>
                  </div>
                  <div class="form-group">
                    <label for="locality">Locality</label>
                    <input type="text"  required class="form-control" name="locality" id="location"
placeholder="Enter Area Name (eg. City/town Name)">
                  </div>
                  <div class="form-group">
                    <label for="postal-code">Postal Code</label>
                    <input type="zip" required class="form-control" name="postal-code"
placeholder="Enter 6 digit postal code">
                  </div>
                  <div class="form-group">
                    <label for="contact-address">
                      Enter you contact address
                    </label>
                    <textarea class="form-control" required name="contact-addrss" id="address"
cols="4" rows="3"></textarea>
                  </div>
                <button type="submit" class="btn btn-primary float-right" style="background-
color:#2DCFFF;box-shadow: 20px; border: #f07279;">Submit</button>
              </form>
          </div>
      </main>
```

```
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
</body>
</html>
```

Track.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Track Request</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<style>
    main{
        margin-top: 80px;
    }
</style>
<body>
```

```
    <nav class="navbar navbar-dark navbar-expand-sm fixed-top" style="background-
color:#f07279;box-shadow: 20px;">
        <a href="/" class="navbar-brand">
            Plasma Donor App
        </a>
    </nav>
    <main>
        {% if session['track_id'] == False %}
        <div class="container d-flex justify-content-center">
            <div class="card mb-2  bg-light" style="width: 30rem;">
                <div class="card-header">
                    Track Your Request
                </div>
                <div class="card-body">
                    <form action="{{url_for('track_request')}}" method="post">
                        <div class="form-group">
                            <label for="tracking-id">Enter Your Request ID / Tracking ID :</label>
                            <input type="text" value="{{req_id}}" placeholder="uZFMIiQJtywmbytv"
class="form-control" name="tracking-id">
                        </div>
                        <button type="submit" class="btn btn-primary">Track</button>
                    </form>
                </div>
            </div>
        </div>
        {% endif %}


        <div class="container">
             {% if session['track_id'] == True %}
            <h5 class="text-center">Track the Status of Any Request</h5>

            <div class="row">
                <div class="col bg-light mr-2">
        <h6 class="mt-1">Request ID</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
            {{res['REQUEST_ID']}}
        </p>
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Request STATUS</h6>
    </div>
    <div class="col bg-secondary">
        {% if res['REQUEST_STATUS'] == 'PENDING': %}
        <p class="text-white badge badge-danger mt-1">
            {{res['REQUEST_STATUS']}}
        </p>
        {% else %}
        <p class="text-white badge badge-primary mt-1">
            {{res['REQUEST_STATUS']}}
        </p>
        {% endif %}
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Recipient Name</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
            {{res['RECIPIENT_NAME']}}
        </p>
    </div>
</div>
```

```
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Recipient Age</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
            {{res['RECIPIENT_AGE']}}
        </p>
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Recipient Email</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
            {{res['RECIPIENT_EMAIL']}}
        </p>
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Recipient Phone</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
            +91 {{res['RECIPIENT_PHONE']}}
        </p>
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Requested Blood Type</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
          {{res['REQUESTED_BLOOD_TYPE']}}
        </p>
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Locality</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
           {{res['LOCALITY']}}
        </p>
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Postal Code</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
           {{res['POSTAL_CODE']}}
        </p>
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Contact Address</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
           {{res['RECIPIENT_ADDRESS']}}
        </p>
```

```
                </div>
            </div>
            <!-- <a href="/cancel"><button type="submit" class="btn btn-danger float-right mt-
5">Cancel Request</button></a> -->
            {% if res['REQUEST_STATUS'] != "PENDING" : %}
            <div class="container">
                <div class="card">
                    <div class="card-body bg-info mt-10">
                        <h3 class="card-text">Please Check your Mail for Any Appointment by the
Donor</h3>
                    </div>
                </div>
            </div>
            {% endif %}
            {% endif %}

        </div>

    </main>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
</body>
</html>
```

User_registration.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Track Request</title>
   <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
</head>
<style>
   main{
      margin-top: 80px;
   }
</style>
<body>
   <nav class="navbar navbar-dark navbar-expand-sm fixed-top" style="background-
color:#f07279;box-shadow: 20px;">
      <a href="/" class="navbar-brand">
         Plasma Donor App
      </a>
   </nav>
   <main>
      {% if session['track_id'] == False %}
      <div class="container d-flex justify-content-center">
         <div class="card mb-2  bg-light" style="width: 30rem;">
            <div class="card-header">
               Track Your Request
            </div>
            <div class="card-body">
               <form action="{{url_for('track_request')}}" method="post">
                  <div class="form-group">
```

```
                        <label for="tracking-id">Enter Your Request ID / Tracking ID :</label>
                        <input type="text" value="{{req_id}}" placeholder="uZFMIiQJtywmbytv"
class="form-control" name="tracking-id">
                    </div>
                    <button type="submit" class="btn btn-primary">Track</button>
                </form>
            </div>
        </div>
    </div>
    {% endif %}


    <div class="container">
         {% if session['track_id'] == True %}
        <h5 class="text-center">Track the Status of Any Request</h5>

        <div class="row">
            <div class="col bg-light mr-2">
                <h6 class="mt-1">Request ID</h6>
            </div>
            <div class="col bg-secondary">
                <p class="text-white  mt-1">
                    {{res['REQUEST_ID']}}
                </p>
            </div>
        </div>
        <div class="row">
            <div class="col bg-light mr-2">
                <h6 class="mt-1">Request STATUS</h6>
            </div>
            <div class="col bg-secondary">
                {% if res['REQUEST_STATUS'] == 'PENDING': %}
                <p class="text-white badge badge-danger mt-1">
                    {{res['REQUEST_STATUS']}}
        </p>
        {% else %}
        <p class="text-white badge badge-primary mt-1">
          {{res['REQUEST_STATUS']}}
        </p>
        {% endif %}
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Recipient Name</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
          {{res['RECIPIENT_NAME']}}
        </p>
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Recipient Age</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
          {{res['RECIPIENT_AGE']}}
        </p>
    </div>
</div>
<div class="row">
    <div class="col bg-light mr-2">
        <h6 class="mt-1">Recipient Email</h6>
    </div>
    <div class="col bg-secondary">
        <p class="text-white  mt-1">
```

```
            {{res['RECIPIENT_EMAIL']}}
          </p>
        </div>
      </div>
      <div class="row">
        <div class="col bg-light mr-2">
          <h6 class="mt-1">Recipient Phone</h6>
        </div>
        <div class="col bg-secondary">
          <p class="text-white  mt-1">
            +91 {{res['RECIPIENT_PHONE']}}
          </p>
        </div>
      </div>
      <div class="row">
        <div class="col bg-light mr-2">
          <h6 class="mt-1">Requested Blood Type</h6>
        </div>
        <div class="col bg-secondary">
          <p class="text-white  mt-1">
           {{res['REQUESTED_BLOOD_TYPE']}}
          </p>
        </div>
      </div>
      <div class="row">
        <div class="col bg-light mr-2">
          <h6 class="mt-1">Locality</h6>
        </div>
        <div class="col bg-secondary">
          <p class="text-white  mt-1">
            {{res['LOCALITY']}}
          </p>
        </div>
      </div>
```

```
        <div class="row">
            <div class="col bg-light mr-2">
                <h6 class="mt-1">Postal Code</h6>
            </div>
            <div class="col bg-secondary">
                <p class="text-white  mt-1">
                    {{res['POSTAL_CODE']}}
                </p>
            </div>
        </div>
        <div class="row">
            <div class="col bg-light mr-2">
                <h6 class="mt-1">Contact Address</h6>
            </div>
            <div class="col bg-secondary">
                <p class="text-white  mt-1">
                    {{res['RECIPIENT_ADDRESS']}}
                </p>
            </div>
        </div>
        <!-- <a href="/cancel"><button type="submit" class="btn btn-danger float-right mt-
5">Cancel Request</button></a> -->
        {% if res['REQUEST_STATUS'] != "PENDING" : %}
        <div class="container">
            <div class="card">
                <div class="card-body bg-info mt-10">
                    <h3 class="card-text">Please Check your Mail for Any Appointment by the
Donor</h3>
                </div>
            </div>
        </div>
        {% endif %}
        {% endif %}
        </div>

    </main>
    <script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj" crossorigin="anonymous"></script>
    <script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js" integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN" crossorigin="anonymous"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js" integrity="sha384-B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV" crossorigin="anonymous"></script>
</body>
</html>
```

Style.css:

```
* {
    margin: 0;
    padding: 0;
}

html,body {
    height: 100%;
}
.global-container{
    height:100%;
    display: flex;
    align-items: center;
    justify-content: center;
    background-color: #f5f5f5;
}
```

```
form{
   padding-top: 10px;
   font-size: 14px;
   margin-top: 30px;
}

.card-title{ font-weight:300; }

.btn{
   font-size: 14px;
   margin-top:20px;
}


.login-form{
   width:330px;
   margin:20px;
}

.sign-up{
   text-align:center;
   padding:20px 0 0;
}
.text-center{
   margin-top: 1.3rem;
   font-family:Arial, Helvetica, sans-serif;
}
.alert{
   margin-bottom:-30px;
   font-size: 13px;
   margin-top:20px;

}
```

app.py

```
from flask import Flask, render_template, redirect
from flask import url_for, session, request
from dotenv import load_dotenv
from mailer import send_the_email
from datetime import datetime
from generator import generate_unique_id
from fetch import fetch_home
from check import check_the_acc_info
import os
import hashlib
import re
import ibm_db
load_dotenv()

app = Flask(__name__)
app.secret_key = os.urandom(16)

try:
    # conn = ibm_db.connect(os.getenv('CREDENTIALS'), '', '')
    # conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=764264db-9824-4b7c-82df-40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32536;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=hmd83768;PWD=4WzDtnPyc6CW98X2", '', '')
    conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=ea286ace-86c7-4d5b-8580-3fbfa46b1c66.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31505;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=lxj14894;PWD=Gz86RyxT6UVIv15C", '', '')

except Exception as err:
    print(ibm_db.conn_errormsg())
```

```
@app.route('/')
def index():
    if not session:
        return render_template('index.htm')

    return redirect(url_for('home'))


@app.route('/login')
def login():
    if not session or not session['login_status']:
        return render_template('login.htm')

    return redirect(url_for('home'))


@app.route('/register')
def register():
    return render_template('register.htm')


@app.route('/account')
def account():
    if not session:
        return redirect(url_for('home'))
    if session['account-type'] == 'Donor':
        useremail = session['user_email']
        sql = "SELECT 
FIRSTNAME,LASTNAME,DOB,PHONE,USER_EMAIL,BLOOD_TYPE,COVID_STAT
US,GENDER,STATE,PINCODE FROM DONORS WHERE USER_EMAIL=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, useremail)
        ibm_db.execute(stmt)
        res = ibm_db.fetch_assoc(stmt)
        return render_template('account.htm', res=res)
    if session['account-type'] == 'user':
        useremail = session['user_email']
        sql = "SELECT FULLNAME,USER_DOB,PHONE_NO,EMAIL FROM USERS
WHERE EMAIL=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, useremail)
        ibm_db.execute(stmt)
        result = ibm_db.fetch_assoc(stmt)
        return render_template('account.htm', res=result)


@app.route('/donate')
def donate():
    if not session or not session['login_status']:
        return render_template('login.htm')

    if session['account-type'] == 'user':
        return redirect(url_for('register'))

    results = {}
    sql = "SELECT * FROM Requests WHERE REQUEST_STATUS=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, 'PENDING')
    ibm_db.execute(stmt)
    result = ibm_db.fetch_assoc(stmt)
    i = 1
    while result:
        results.update({i: result})
        i = i + 1
        result = ibm_db.fetch_assoc(stmt)
    return render_template('donate.htm', results=results)


@app.route('/BookAppointment/<req_id>')
```

```
def book_appointment(req_id):

    return render_template('donateForm.htm', req_id=req_id)


@app.route('/err')
def err():
    return render_template('err.htm', err_msg)


@app.route('/track')
def track():
    session['track_id'] = False
    return render_template('track.htm')


@app.route('/request')
def _request():
    if not session or not session['login_status']:
        return render_template('user_registration.htm')

    return render_template('request.htm')


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

    if request.method == 'POST':
        track_id = request.form['tracking-id']

        sql = "SELECT * FROM REQUESTS WHERE REQUEST_ID=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, track_id)
        ibm_db.execute(stmt)
        res = ibm_db.fetch_assoc(stmt)
        if res:
            session['track_id'] = True
            return render_template('track.htm', res=res)
        if not res:
            err_msg = 'There is no such request with this request id. '
            err_msg += 'Please Check Your Request ID once again'
            return render_template('err.htm', err_msg=err_msg)


@app.route('/track_req/<req_id>')
def track_req(req_id):
    track_id = req_id
    sql = "SELECT * FROM REQUESTS WHERE REQUEST_ID=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, track_id)
    ibm_db.execute(stmt)
    res = ibm_db.fetch_assoc(stmt)
    if res:
        session['track_id'] = True
        return render_template('track.htm', res=res)
    if not res:
        err_msg = 'There is no such request with this request id. '
        err_msg += 'Please Check Your Request ID once again'
        return render_template('err.htm', err_msg=err_msg)


@app.route('/user_register', methods=['GET', 'POST'])
def user_register():
    if request.method == 'POST':
        user_name = request.form['username']
        user_dob = request.form['dob']
        user_phone = request.form['user-phone']
        user_email = request.form['useremail']
        password = request.form['password']
        cnf_password = request.form['cnf-password']

        # hashing the password
        if password != cnf_password:
            msg = "Password Doesn't Match"
            return render_template('err.htm', err_msg=msg)

        password = bytes(password, 'utf-8')
        password = hashlib.sha256(password).hexdigest()
        # password hashed

    # case 1: check if user does exists already
        sql = "SELECT * FROM users WHERE email =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, user_email)
        ibm_db.execute(stmt)
        acc = ibm_db.fetch_assoc(stmt)
        if acc:
            msg = "Account already Exists, Please login"
            return render_template('err.htm', err_msg=msg)

         # case 2: validate the input if it matches the required pattern
        if not re.match(r"^\S+@\S+\.\S+$", user_email):
            msg = "Please Enter Valid Email Address "
            return render_template('err.htm', err_msg=msg)

        insert_sql = "INSERT INTO  users VALUES (?, ?, ?, ?, ?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, user_name)
        ibm_db.bind_param(prep_stmt, 2, user_dob)
        ibm_db.bind_param(prep_stmt, 3, user_phone)
        ibm_db.bind_param(prep_stmt, 4, user_email)
        ibm_db.bind_param(prep_stmt, 5, password)
        ibm_db.execute(prep_stmt)
```

```
        to_email = user_email
        subject = "Confirmation on Registration with Plasma-Donor-App as User"
        html_content = '''

          <h1>Registration Successfull</h1><br>
          <p> Thank you so much for registering with us </p><br>
         <p> You are now registered user </p>

        '''
        send_the_email(to_email, subject, html_content)
        return redirect(url_for('login'))


@app.route('/home')
def home():
    if not session:
        return redirect(url_for('login'))

    if session['login_status']:
        req, res = fetch_home(conn=conn)
        return render_template('home.htm', username=session['user_id'], req=req, res=res)

    return redirect(url_for('login'))


@app.route('/do_register', methods=['GET', 'POST'])
def do_register():
    if request.method == 'POST':
        first_name = request.form['fname']
        last_name = request.form['lname']
        email = request.form['email']
        addrss1 = request.form['Locality']
        addrss2 = request.form['address']
```

```
state = request.form['State']
pincode = request.form['Zip']
dob = request.form['dob']
gender = request.form['gender']
phone = request.form['phone']
covid_status = request.form['covid-report']
blood_type = request.form['b-type']
# ------------------
# password hashing
password = request.form['password']
cnf_password = request.form['cnf-password']
if password != cnf_password:
    msg = "Password Doesn't Match"
    return render_template('err.htm', err_msg=msg)

password = bytes(password, 'utf-8')
password = hashlib.sha256(password).hexdigest()

# case 1: check if user does exists already
sql = "SELECT * FROM donors WHERE user_email =?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, email)
ibm_db.execute(stmt)
acc = ibm_db.fetch_assoc(stmt)
if acc:
    msg = "Account already Exists, Please login "
    return render_template('err.htm', err_msg=msg)

# case 2: validate the input if it matches the required pattern
if not re.match(r"^\S+@\S+\.\S+$", email):
    msg = "Please Enter Valid Email Address "
    return render_template('err.htm', err_msg=msg)

insert_sql = "INSERT INTO  donors VALUES (?, ?, ?, ?, ?, ?, ?,?, ?, ?, ?, ?,?)"
```

```
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, first_name)
        ibm_db.bind_param(prep_stmt, 2, last_name)
        ibm_db.bind_param(prep_stmt, 3, email)
        ibm_db.bind_param(prep_stmt, 4, addrss1)
        ibm_db.bind_param(prep_stmt, 5, addrss2)
        ibm_db.bind_param(prep_stmt, 6, state)
        ibm_db.bind_param(prep_stmt, 7, pincode)
        ibm_db.bind_param(prep_stmt, 8, dob)
        ibm_db.bind_param(prep_stmt, 9, gender)
        ibm_db.bind_param(prep_stmt, 10, phone)
        ibm_db.bind_param(prep_stmt, 11, covid_status)
        ibm_db.bind_param(prep_stmt, 12, blood_type)
        ibm_db.bind_param(prep_stmt, 13, password)
        ibm_db.execute(prep_stmt)

        to_email = email
        subject = 'Confirmation on Registration with Plasma-Donor-App'
        html_content = '''
            <h1>Registration Successfull</h1><br>
            <p> Thank you so much for registering with us </p><br>
            <p> You are now registered donor </p>
        '''
        send_the_email(to_email, subject, html_content)
        return redirect(url_for('login'))
    return redirect(url_for('register'))


@app.route('/do_login', methods=['GET', 'POST'])
def do_login():
    if request.method == 'POST':
        user_email = request.form['user_email']
        password = request.form['password']
        # salt the password
```

```
password = bytes(password, 'utf-8')
password = hashlib.sha256(password).hexdigest()

# query the db
sql = "SELECT * FROM donors WHERE user_email =? AND pass_word=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, user_email)
ibm_db.bind_param(stmt, 2, password)
ibm_db.execute(stmt)
acc = ibm_db.fetch_assoc(stmt)
if not acc:
    # check if present in users
    sql = "SELECT * FROM users WHERE email =? AND password=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, user_email)
    ibm_db.bind_param(stmt, 2, password)
    ibm_db.execute(stmt)
    acc = ibm_db.fetch_assoc(stmt)
    session['account-type'] = 'user'
    session['login_status'] = True
    session['user_email'] = user_email
    session['user_id'] = user_email.split('@')[0]
    return redirect(url_for('home'))
if acc:
    session['login_status'] = True
    session['account-type'] = 'Donor'
    session['user_email'] = user_email
    session['user_id'] = user_email.split('@')[0]
    return redirect(url_for('home'))

# check if the acc exists
sql = "SELECT * FROM donors WHERE user_email=?"
stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(stmt, 1, user_email)
```

```
        ibm_db.execute(stmt)
        res = ibm_db.fetch_assoc(stmt)
        if res:
            msg = "Account already Exists, Please login "
            return render_template('err.htm', err_msg=msg)
        else:
            msg = "Don't you have an account ? try register with us "
            return render_template('err.htm', err_msg=msg)


@app.route('/do_request', methods=['GET', 'POST'])
def do_request():
    if request.method == 'POST':
        name = request.form['name']
        age = request.form['age']
        email = request.form['email']
        phone = request.form['phone']
        requested_blood_type = request.form['blood-type']
        locality = request.form['locality']
        postal_code = request.form['postal-code']
        address = request.form['contact-addrss']

        # generate request id
        request_id = generate_unique_id()
        # initial status of the request
        request_status = 'PENDING'

        insert_sql = "INSERT INTO  requests VALUES (?, ?, ?, ?, ?, ?, ?,?, ?, ?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, request_id)
        ibm_db.bind_param(prep_stmt, 2, request_status)
        ibm_db.bind_param(prep_stmt, 3, name)
        ibm_db.bind_param(prep_stmt, 4, age)
        ibm_db.bind_param(prep_stmt, 5, email)
        ibm_db.bind_param(prep_stmt, 6, phone)
        ibm_db.bind_param(prep_stmt, 7, requested_blood_type)
        ibm_db.bind_param(prep_stmt, 8, locality)
        ibm_db.bind_param(prep_stmt, 9, postal_code)
        ibm_db.bind_param(prep_stmt, 10, address)
        ibm_db.execute(prep_stmt)

        return render_template('success.htm', request_id=request_id)


@app.route('/make_donation', methods=['GET', 'POST'])
def make_donation():
    if request.method == 'POST':
        request_id = request.form['req_id']
        donor_name = request.form['donor-name']
        donor_age = request.form['donor-age']
        blood_type = request.form['blood-type']
        medical_status = request.form['medical-status']
        location = request.form['location']
        date_time = request.form['datetime']
        date_time = datetime.strptime(date_time, '%Y-%m-%dT%H:%M')
        phone_number = request.form['phone-number']
        contact_address = request.form['contact-address']

        datenow = datetime.now().strftime('%Y-%m-%dT%H:%M')
        if str(date_time) < datenow:
            msg = "The Date you've entered is not suitable for making this appointment"
            return render_template('err.htm', err_msg=msg)

        chck = "SELECT * FROM Appointments WHERE request_id=?"
        stmt = ibm_db.prepare(conn, chck)
        ibm_db.bind_param(stmt, 1, request_id)
        ibm_db.execute(stmt)
        res = ibm_db.fetch_assoc(stmt)
        if res:
            msg = " The Request was Already Engaged"
            return render_template('err.htm', err_msg=msg)

        sql = "INSERT INTO  Appointments VALUES (?, ?, ?, ?, ?, ?, ?,?, ?)"
        prep_stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(prep_stmt, 1, request_id)
        ibm_db.bind_param(prep_stmt, 2, donor_name)
        ibm_db.bind_param(prep_stmt, 3, donor_age)
        ibm_db.bind_param(prep_stmt, 4, blood_type)
        ibm_db.bind_param(prep_stmt, 5, medical_status)
        ibm_db.bind_param(prep_stmt, 6, location)
        ibm_db.bind_param(prep_stmt, 7, date_time)
        ibm_db.bind_param(prep_stmt, 8, phone_number)
        ibm_db.bind_param(prep_stmt, 9, contact_address)
        ibm_db.execute(prep_stmt)

        upt_sql = "UPDATE requests SET request_status=? WHERE request_id=?"
        status = "ACCEPTED BY DONOR"
        upt_stmt = ibm_db.prepare(conn, upt_sql)
        ibm_db.bind_param(upt_stmt, 1, status)
        ibm_db.bind_param(upt_stmt, 2, request_id)
        ibm_db.execute(upt_stmt)

        msql = "SELECT recipient_email FROM requests WHERE request_id=?"
        mstmt = ibm_db.prepare(conn, msql)
        ibm_db.bind_param(mstmt, 1, request_id)
        ibm_db.execute(mstmt)
        res = ibm_db.fetch_assoc(mstmt)
        to_email = res['RECIPIENT_EMAIL']
        subject = f'Your Request ID {request_id} has been Accepted By The Donor and Please
refer the content of this mail'
        content = f'''
            <h1>Donor Found </h1>
            <h2>Details of the Donor and Appointment</h2>
            <body>
            <pre>
            Request ID      : {request_id}
            Donor's Name    : {donor_name}
            Donor's Age     : {donor_age}
            Medical Status  : {medical_status}
            Blood Type      : {blood_type}
            Location        : {location}
            Date and Time   : {date_time}
            Contact Address : {contact_address}
            </pre>
            <h3> You May contact the Donor For Full Details</h3>
            <h3>Get Well Soon</h3>
            </body>
        '''
        send_the_email(to_email, subject, content)

        return redirect('/track_req/'+request_id)


@app.route('/logout')
def logout():
    # session['login_status'] = False
    session.pop('login_status', None)
    session.pop('user_id', None)
    session.pop('user_email', None)
    session.pop('account-type', None)
    session.pop('track_id', None)

    return redirect(url_for('index'))


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

check.py

```
import ibm_db
from dotenv import load_dotenv
import os


load_dotenv()


try:
    # conn = ibm_db.connect(os.getenv('CREDENTIALS'),",")
    conn = ibm_db.connect(
        "DATABASE=bludb;HOSTNAME=764264db-9824-4b7c-82df-
40d1b13897c2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32536;SECURIT
Y=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=hmd83768;PWD=4WzDtnP
yc6CW98X2", ", ")
except Exception as err:
     print("Exception occurs->", err)

def check_the_acc_info(user_email):
    sql = "SELECT * FROM donors WHERE user_email=?"
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.bind_param(stmt,1,user_email)
    ibm_db.execute(stmt)
    donor_acc = ibm_db.fetch_assoc(stmt)

    user_sql = "SELECT * FROM users WHERE email=?"
    user_stmt = ibm_db.prepare(conn,user_sql)
    ibm_db.bind_param(user_stmt,1,user_email)
    ibm_db.execute(user_stmt)
    user_acc = ibm_db.fetch_assoc(user_stmt)
    result = ""
    if donor_acc and user_acc:
        result = 'donor-user-account'
    elif donor_acc:
        result = 'donor-account'
    elif user_acc:
        result = 'user-account'
    else:
        return False

    return result
```

fetch.py

```
from dotenv import load_dotenv
import os
import ibm_db



def fetch_home(conn):
    sql = "SELECT COUNT(*) , (SELECT COUNT(*) FROM DONORS WHERE
blood_type= 'A Positive'),"
    sql += "(SELECT COUNT(*) FROM DONORS WHERE blood_type='A Negative'),
(SELECT COUNT(*) FROM DONORS WHERE blood_type='B Positive'),"
    sql += "(SELECT COUNT(*) FROM DONORS WHERE blood_type='B Negative'),
(SELECT COUNT(*) FROM DONORS WHERE blood_type='O Positive'),"
    sql += "(SELECT COUNT(*) FROM DONORS WHERE blood_type='O Negative'),
(SELECT COUNT(*) FROM DONORS WHERE blood_type='AB Positive'),"
    sql += "(SELECT COUNT(*) FROM DONORS WHERE blood_type='AB Negative')
from donors"
```

```
    req_sql = "SELECT COUNT(*) FROM REQUESTS WHERE REQUEST_STATUS != 
'ACCEPTED'"
    req_stmt = ibm_db.prepare(conn,req_sql)
    ibm_db.execute(req_stmt)
    req = ibm_db.fetch_assoc(req_stmt)
    stmt = ibm_db.prepare(conn,sql)
    ibm_db.execute(stmt)
    res = ibm_db.fetch_assoc(stmt)
    return req,res
```

generator.py:

```
import random
import string


STRING_ID_SIZE=16

# function to generate the 16 digit unique id for track the request
def generate_unique_id():
    unique_id_16 = ''.join([random.choice(string.ascii_letters + string.digits) for n in 
range(STRING_ID_SIZE)])
    return unique_id_16
```

mailer.py:

```
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
from dotenv import load_dotenv
import os

load_dotenv()


def send_the_email(to_email,subject,html_content):
```

```
message = Mail(from_email='sriramraju26278@gmail.com',
to_emails=to_email,subject=subject,
html_content=html_content)

try:
    sg = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sg.send(message)
    print(response.status_code)
    print(response.body)
    print(response.headers)
    return
except Exception as e:
    print(e.message)
    return
```

13.2 Project link:

https://github.com/IBM-EPBL/IBM-Project-39336-1660406778