DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
R.M.K. COLLEGE OF ENGINEERING AND TECHNOLOGY
PUDUVOYAL – 601206

PROJECT NAME : PLASMA DONOR APPLICATION

TEAM ID: PNT2022TMID14456

DONE BY:

SOWMYA P - 111619104143
SWATHI N - 111619104151
SHAMITHA R V - 111619104137
SOWMIYA E - 111619104142

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Project Name : **Plasma Donor Application**

Category : Cloud App Development

Project Description:

During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low. Saving the donor information and helping the needy by notifying the current donors list, would be a helping hand. In regard to the problem faced, an application is to be built which would take the donor details, store them and inform them upon a request.

Skills Required :

IBM Cloud,HTML,Javascript,IBM Cloud Object Storage,Python-Flask,Kubernetes,Docker,IBM DB2,IBM Container Registry

Technical Architecture :

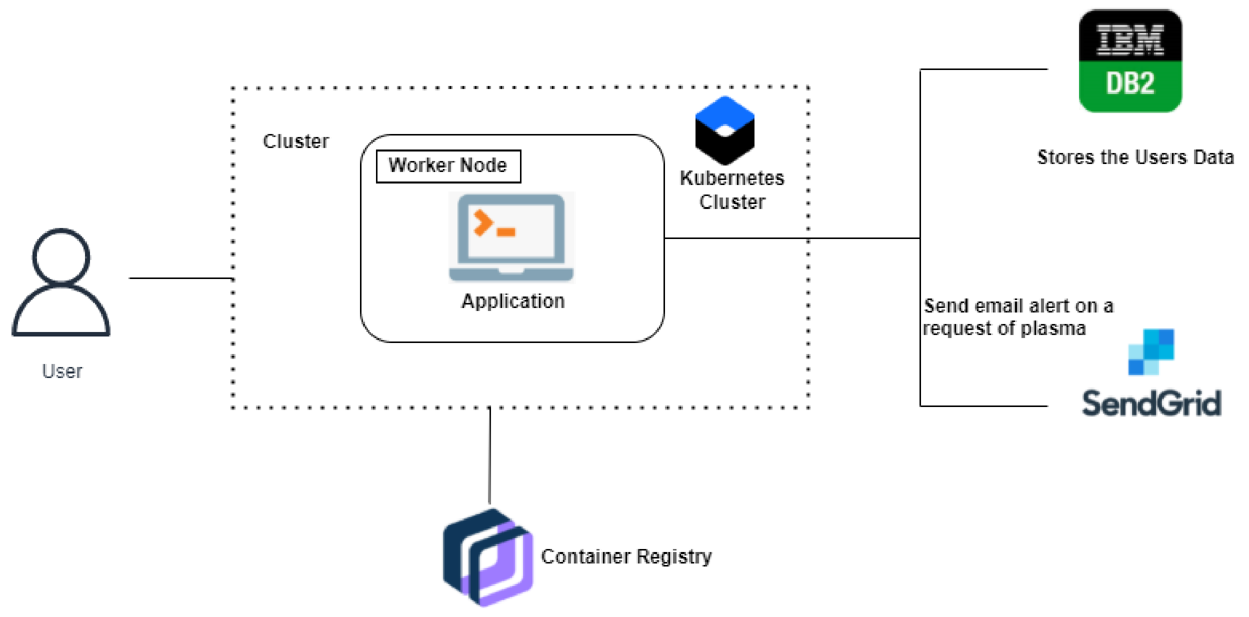


Fig 1.1 Technical Architecture

1.2 PROJECT PURPOSE

Blood plasma donations are used for slightly more specific purposes than a general blood donation. The most common uses of plasma donations include individuals who have experienced a severe trauma, burn or shock, adults or children with cancer, and people with liver or clotting factor disorders.

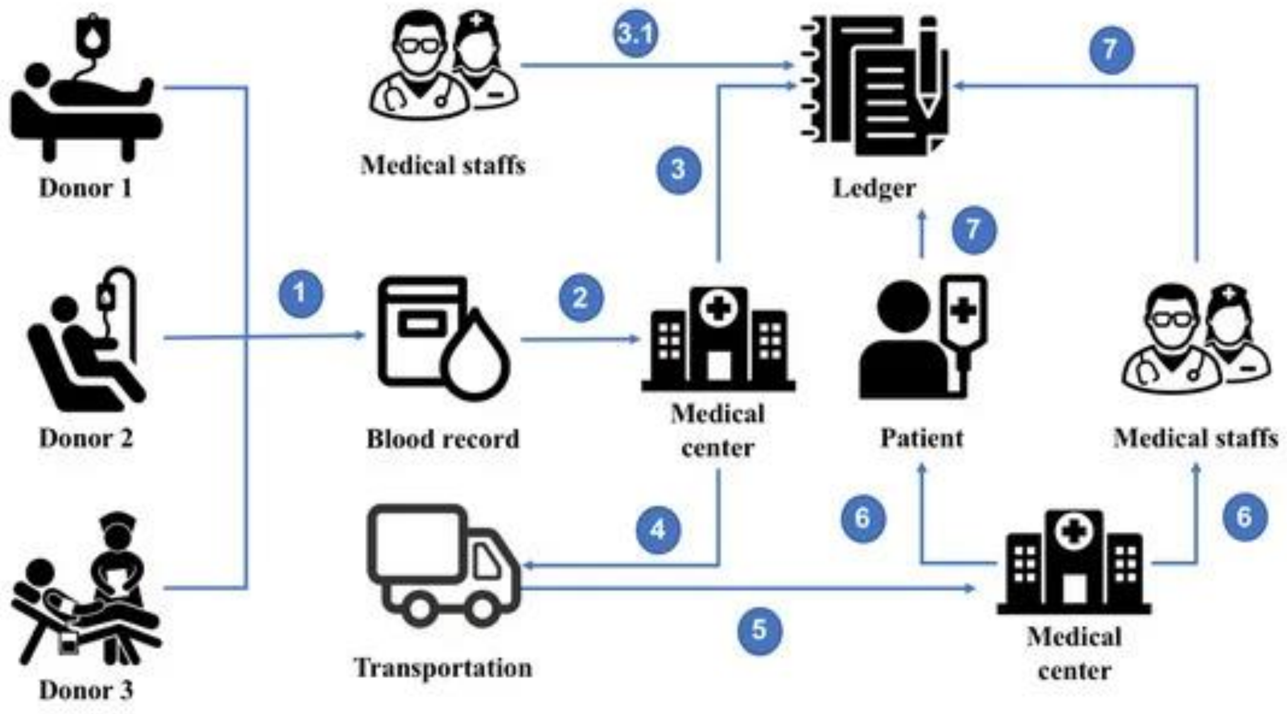


Fig1.2 Project flowchart

Main Objective :

To create the web application for helping the hospital for collecting the plasma from blood donors and supplying it to the needed hospitals with robust and low latency application.

The main role of plasma is to take nutrients, hormones, and proteins to the parts of the body that need it. Cells also put their waste products into the plasma. The plasma then helps remove this waste from the body. Blood plasma also carries all parts of the blood through your circulatory system.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

Optimization can address population health problems such as matching organ donors and receivers or designing radiation treatment plans that minimize harm to the patient. Financially, optimization can specify how to allocate funds to various service lines of a hospital. Application of logistic methods is developing as a major area of study in recent times. Numerous healthcare-related issues, from resource management in hospitals to the provision of care services in a territory, have given rise to optimization methodologies. However, optimization strategies can also enhance other healthcare services that have so far received only sporadic attention. One of these is the Plasma Donation (BD) system, which aims to supply hospitals and transfusion centers with an appropriate supply of plasma. Plasma is still a scarce resource but is required for many procedures and therapies. About 10 million units of plasma are required annually in the USA, 2.1 in Italy, and 2 in Turkey. Additionally, there are still deaths related to a lack of plasma products in some nations (World Health Organisation 2014). As a result, BD is crucial to healthcare systems since it aims to ensure that there is enough Plasma available to fulfill demand and save lives.

2.2 REFERENCES

A Health-IoT Platform Based on the Integration of Intelligent Packaging, Unobtrusive Bio-Sensor, and Intelligent Medicine Box was the subject of an IEEE article published in 2015 and written by Geng Yang, Li Xie, Matti Mantysalo, Xiaolin Zhou, Zhibo Pang, Li Da Xu, Sharon Kao-Walter, Qiang Chen, and Lirong Zheng. This study proposes and implements an intelligent home-based healthcare platform. It entails the connectivity-enabled iMedBox, the RFID-enabled iMedPack, the Biopatch, and the SOC. With IoT, it fuses. The wearable Bio-Patch can instantly detect and communicate the user's biosignals to the iMedBox. The absence of a complete platform is the only limitation. Additionally, the physical size, rigidity, and short battery life constitute a hindrance to long-term use. Data Mining for Better Healthcare: A Path Toward Automated Data Analysis? was the title of an IEEE paper published in 2016. by Lia Morra, Silivia Chiusano, Tania Cerquitelli, Elena Baralis, and others. This paper discusses mining from the standpoint of a medical database. With little user involvement, the mining system should be able to identify the knowledge that would be most useful to the user and extract it from a big medical dataset. The system should be able to extract manageable sets of knowledge that can be put to use. To imagine a system capable of assessing and contrasting numerous data-mining technique configurations at once, large parameter spaces must be examined at an abstraction level. Amiya Kumar Tripathy, Rebeck Carvalho, Keshav Pawaskar, Suraj Yadav, and Vijay Yadav wrote an IEEE article on Mobile Based Healthcare Management using Artificial Intelligence in 2015. The health-care management system that is presented in this study will include mobile heart rate measurement so that the data can be transferred and diagnosis based on heart rate may be immediately delivered with the push of a button. To communicate with a doctor at a distance, a video conferencing technology will be used. Additionally, Doc-Bot and an online Blood Bank will be part of the system. Due to noise in the input data, the heart rate computation in this project differs from the actual one. As a result, the performance is ineffective in real life. Clustering, Text Mining, Pattern Matching, Support Vector Machine, Partitioning Algorithm, and the DonorHART programme were among the methodologies utilized to gather data on donor reactions. Limitations include a lack of adequate security against misuse of personal information and difficulty in resolving emergency situations

2.3 PROBLEM STATEMENT DEFINITION

The goal of this project is to help to connect donors And the people in need through the help of technology. Because plasma has a short shelf life from donation to use and needs special handling during collection and storage, the system must be continuously refueled. This application helps in saving the lives of the people who are in need of plasma by an efficient donation process.

3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

Empathy map is used to prioritize the ideas from the list of ideas from brainstorming

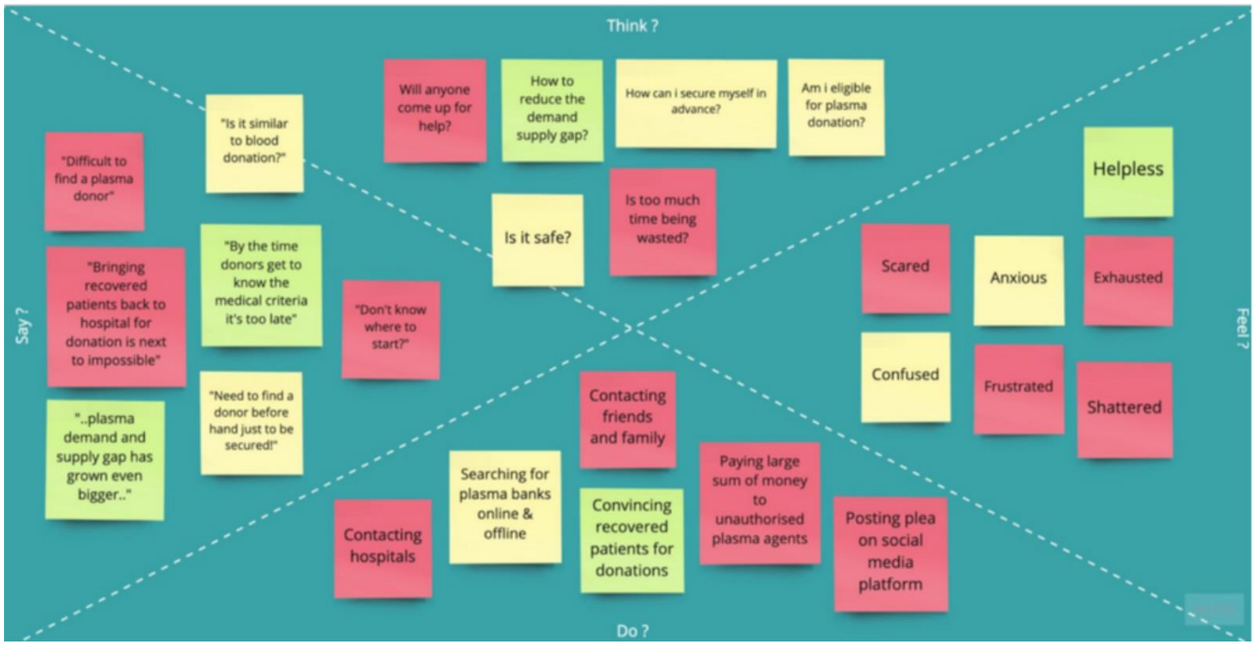
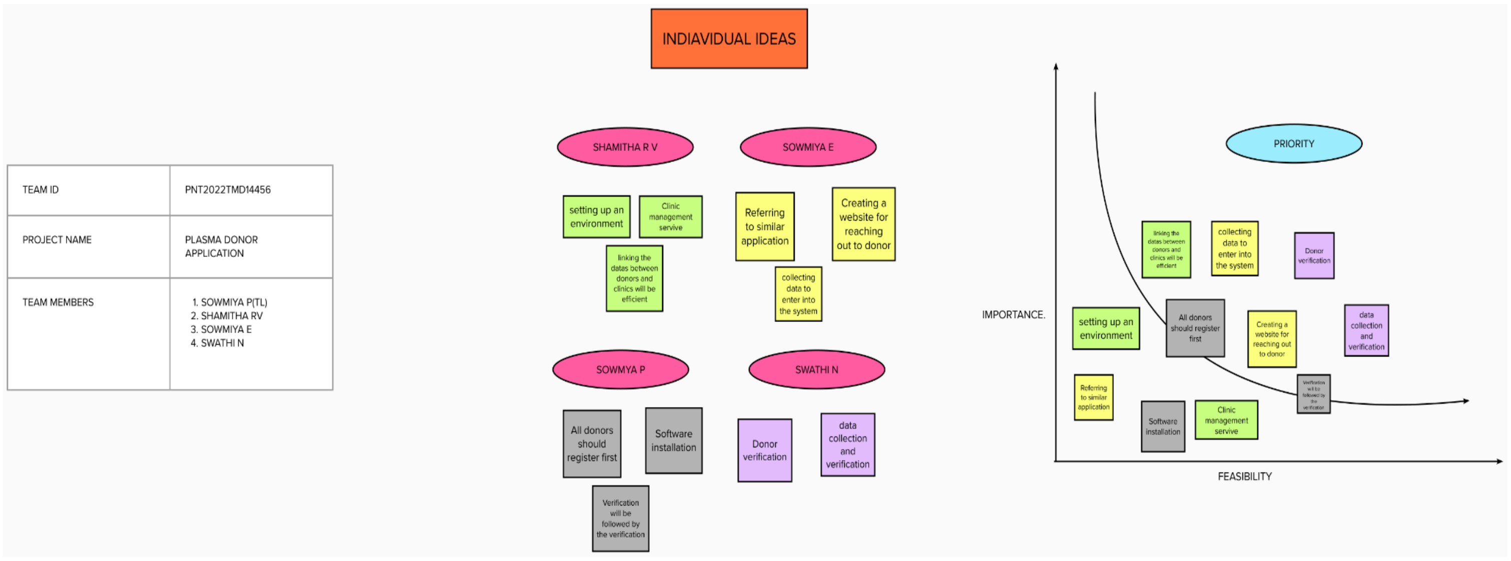

Think ?
Say ?
Feel ?
Do ?
"Difficult to find a plasma donor"
"Is it similar to blood donation?"
Will anyone come up for help?
How to reduce the demand supply gap?
How can i secure myself in advance?
Am i eligible for plasma donation?
Helpless
"Bringing recovered patients back to hospital for donation is next to impossible"
"By the time donors get to know the medical criteria it's too late"
"Don't know where to start?"
Is it safe?
Is too much time being wasted?
Scared
Anxious
Exhausted
"..plasma demand and supply gap has grown even bigger.."
"Need to find a donor before hand just to be secured!"
Confused
Frustrated
Shattered
Contacting friends and family
Searching for plasma banks online & offline
Convincing recovered patients for donations
Paying large sum of money to unauthorised plasma agents
Contacting hospitals
Posting plea on social media platform

Fig. 3.1 Empathy map

3.2 IDEATION & BRAINSTORMING



TEAM ID	PNT2022TMID14456
PROJECT NAME	PLASMA DONOR APPLICATION
TEAM MEMBERS	1. SOWMIYA P(TL) 2. SHAMITHA RV 3. SOWMIYA E 4. SWATHI N

INDIAVIDUAL IDEAS
SHAMITHA R V
setting up an environment
Clinic management servive
linking the datas between donors and clinics will be efficient
SOWMIYA E
Referring to similar application
Creating a website for reaching out to donor
collecting data to enter into the system
SOWMYA P
All donors should register first
Software installation
Verification will be followed by the verification
SWATHI N
Donor verification
data collection and verification
PRIORITY
IMPORTANCE.
FEASIBILITY

Fig 3.2 Individual cases

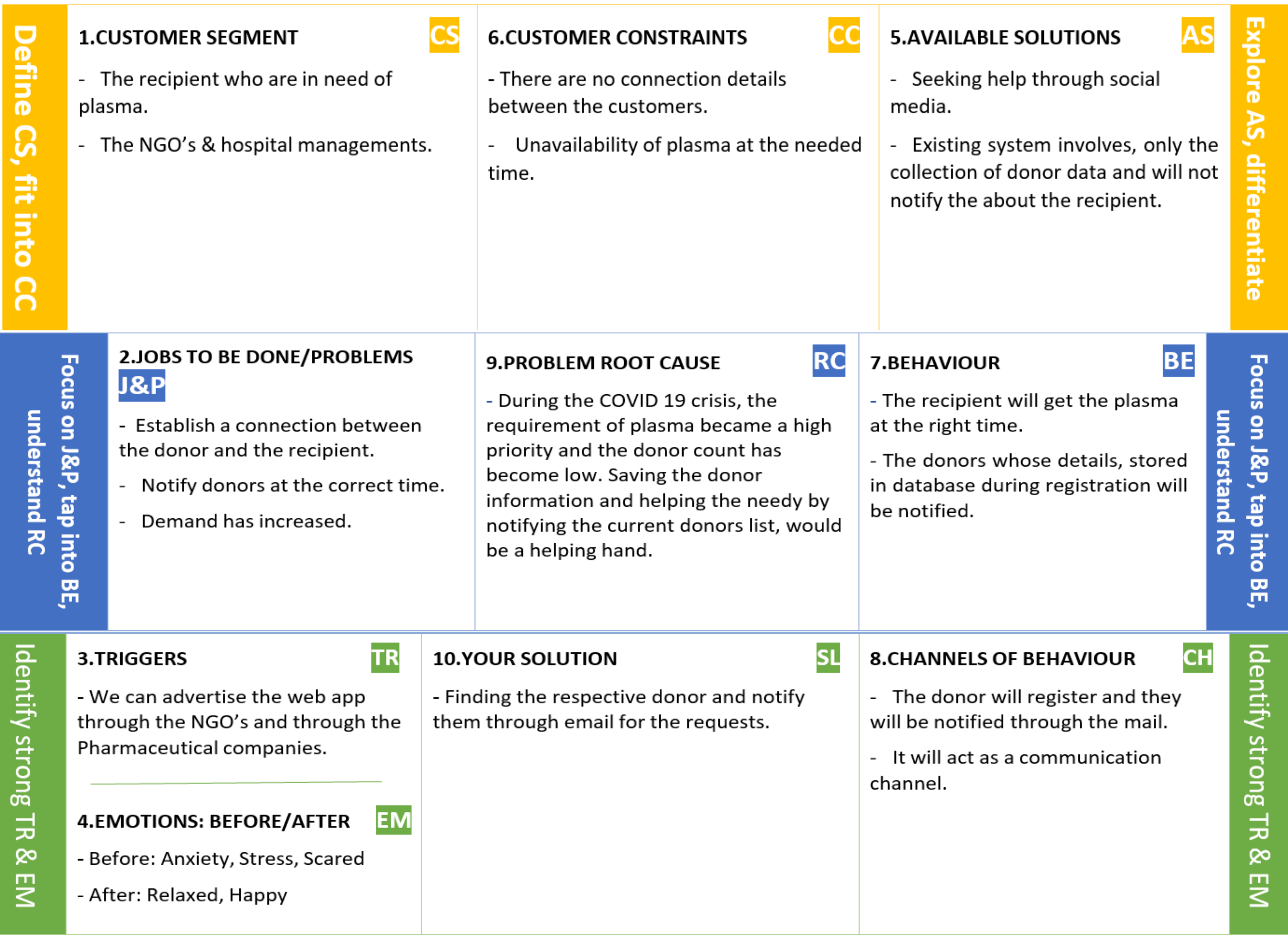
3.3 PROPOSED SOLUTION

S. No	Parameter	Description
1.	Problem Statement (Problem to be solved)	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.
2.	Idea / Solution description	In regard to the problem, an application is to be built which would take the donor details, store and inform them upon a request.
3.	Novelty / Uniqueness	The fresh & responsive interface along with the quick response on giving notification to recipient & donor has been the novelty of the model. A chatbot, which is fully dedicated to that application environment will interact with the users.
4.	Social Impact / Customer Satisfaction	In society, it will create an awareness among the people about donation of plasma which will be done in an easy way of connecting the donor and the recipient. For sure, this application will satisfy the customers with its services.
5.	Business Model (Revenue Model)	The proposed model can be deployed in collaboration with the NGO's and in turn can yield revenue for each and every request from the user.

6.	Scalability of the Solution	The model can be deployed on a large scale based on the requirement and usage by the users.

3.4 PROBLEM SOLUTION

Problem solution consists of 6 parts which includes customer segment, customer constraints, behavior, problem root cause, behavior, already existing solutions, currently developed solutions. The possible triggers and how the solution behaves.



Define CS, fit into CC

1.CUSTOMER SEGMENT CS

- The recipient who are in need of plasma.
- The NGO's & hospital managements.

6.CUSTOMER CONSTRAINTS CC

- There are no connection details between the customers.
- Unavailability of plasma at the needed time.

5.AVAILABLE SOLUTIONS AS

- Seeking help through social media.
- Existing system involves, only the collection of donor data and will not notify the about the recipient.

Explore AS, differentiate

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

2.JOBS TO BE DONE/PROBLEMS J&P

- Establish a connection between the donor and the recipient.
- Notify donors at the correct time.
- Demand has increased.

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

- The recipient will get the plasma at the right time.
- The donors whose details, stored in database during registration will be notified.

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

Identify strong TR & EM

3.TRIGGERS TR

- We can advertise the web app through the NGO's and through the Pharmaceutical companies.

4.EMOTIONS: BEFORE/AFTER EM

- Before: Anxiety, Stress, Scared
- After: Relaxed, Happy

10.YOUR SOLUTION SL

- Finding the respective donor and notify them through email for the requests.

8.CHANNELS OF BEHAVIOUR CH

- The donor will register and they will be notified through the mail.
- It will act as a communication channel.

Identify strong TR & EM

4 . REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Gmail
FR-2	User Confirmation	Confirmation via Email
FR-3	User Login	Login using registered mail id
FR-4	Sent Request	If plasma is required, the receiver will contact the donor
FR-5	Contact Donor	Contact the donor directly if a phone number is given
FR-6	View donation campus	View the list of donation camps happening nearby

4.2 NON - FUNCTIONAL REQUIREMENTS

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The user interface of the plasma donor system must be well-designed and welcoming.
NFR-2	**Security**	Data storage is required by the security system, just like it is by many other applications.
NFR-3	**Reliability**	The system has the ability to word all the times without failures apart from network failure. A donor can have faith in the system. The authorities will keep the privacy of all donors in a proper manner.
NFR-4	**Performance**	The plasma donor system must perform well in different scenarios. The system is interactive and delays involved are less.
NFR-5	**Availability**	The system, including the online components, should be available 24/7.
NFR-6	**Scalability**	The system offers the proper resources for issue solutions and is designed to protect sensitive information during all phases of operation.

5.PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS:

Plasma Donor Application DataFlow

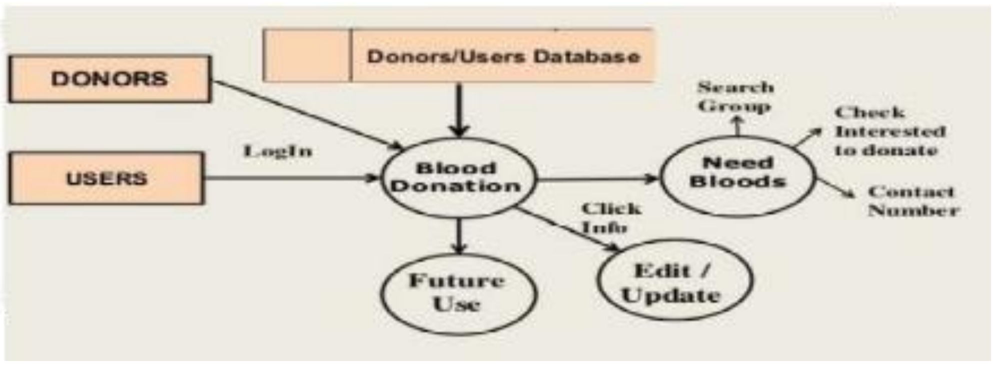


Fig 5.1.1 Database Flowchart

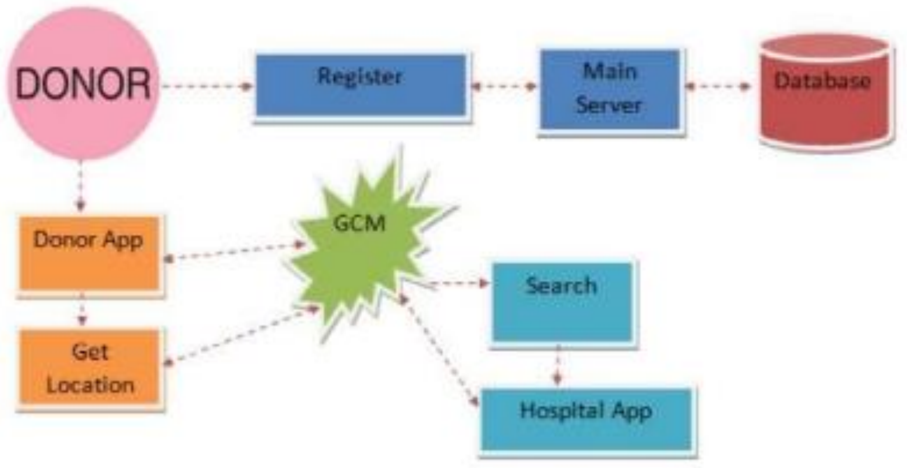


Fig. 5.1.2 Software flowchart

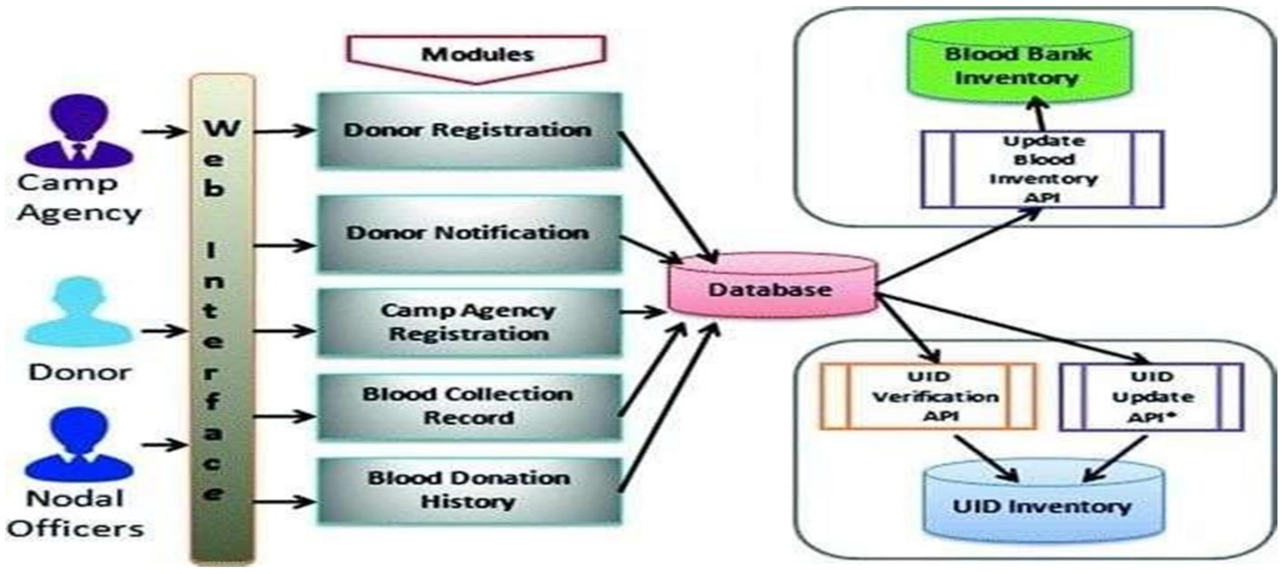


Fig 5.1.3 Project Architecture

5.2 SOLUTION AND TECHNICAL ARCHITECTURE

Problem Statement

During COVID 19 crisis the requirement for plasma increased drastically. The average donation rate for plasma has decreased from an already low 20% to a dismal 11%.
Considering the complex manufacturing process to fractionate plasma into the therapies patients rely on can take 7-12 months, any decline in donations is concerning.

Compounding the effects of ongoing decline checking the donor history, i.e., whether he /she was infected previously and was recovered, and which donor is eligible to donate plasma was a challenging task.

Also, saving the healthy donor information, notifying the interested patients and matching the donors with the requestees proved to be a strenuous job.

Proposed Solution

The proposed method creates an application which aims to solve the aforementioned drawbacks. The system works with the registration of a donor by providing the required details which get stored in the database.

Features

Whenever a new user posts a request, the donors with the matching blood group are notified about the request. Interested donors can then respond and donate their plasma.

End User

The user will be plasma requiring patients and the interested blood donors.

Software Requirements

Python, Flask, Docker.

ARCHITECTURE

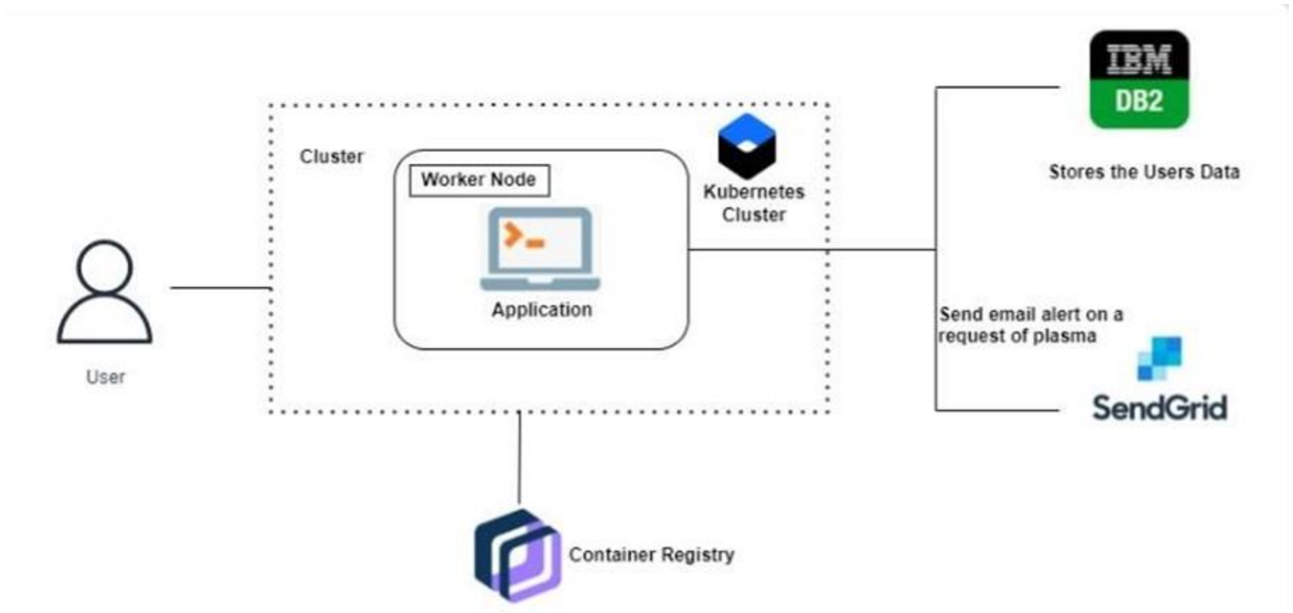


Fig 5.2 Project Architecture

Project Workflow

The user interacts with the application.Registers by giving the details as a donor. The database will have all the details and if a user posts a request, then the concerned blood group donors will get notified about it.

5.3 USER STORIES

Our testing goal was to ask the real users (donors) how they feel about the app, investigate the potential usability problems, and then introduce changes if needed. The big plus was to test the app on actual devices, so we could identify real problems and gather constructive feedback. Again, the good relation with the client was the helpful aspect to get to the real users – blood donors. We introduced the app in the store in developer mode and conducted the semi-closed tests on future app users. As a result of the tests, we made changes to improve usability and user experience.

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

MILESTONES	DESCRIPTION	DATE
Literature Survey **(ASSIGNED TO TEAM MEMBER: SOWMIYA E)**	Literature Survey is the collection of facts and information from recognized authors and articles. Our literature survey focuses mainly on the growing demand of blood plasma and ways to satisfy the demand through an application that interacts with users to find a donor.	12 October 2022
Empathy Map **(ASSIGNED TO TEAM LEADER: P. SOWMYA)**	Empathy map is an easy to convey visual about the user's mindset regarding a particular issue. As of our case, it describes the problems and mindset of people who are in need of blood plasma during an emergency.	12 October 2022
Brainstorming and Idea Prioritizing **(ASSIGNED TO TEAM MEMBER: SHAMITHA R V)**	Brainstorming is the collection of ideas from all members in the team to arrive at a solution for solving a problem. On other hand, prioritizing those ideas helps to find the most needed and common ideas among the team members.	12 October 2022
Proposed Solution **(ASSIGNED TO TEAM MEMBER: SOWMIYA E)**	Proposed Solution is made in response to the needs of the customer, providing a solution to the problem with uniqueness thereby satisfying the customer needs.	12 October 2022

Problem-Solution Fit (ASSIGNED TO TEAM MEMBER: SHAMITHA R V)	Problem-Solution Fit actually verifies if the proposed solution matches with the customer problems considering the behavioral patterns of the customers. It helps entrepreneurs, marketers and corporate innovators to recognize what would work and why.	12 October 2022
Solution Architecture (ASSIGNED TO TEAM MEMBERS: SWATHI N)	Solution Architecture is an intricate process with many branches that connect the space between users' problems and technology solutions.	12 October 2022
Customer Journey (ASSIGNED TO TEAM MEMBERS: SWATHI N)	Customer Journey is the interaction of the customer with the product features. Out here, our customer feels free to have direct and indirect interactions and queries regarding our application.	20 October 2022
Functional requirements (ASSIGNED TO TEAM LEADER :P SOWMYA)	Functional requirements describe for the user, the methods to have ties with the application ie.., account creation method for the first login, etc. Also assures many Non-functional requirements like security, etc.	19 October 2022
Data flow diagram (ASSIGNED TO TEAM MEMBER: SOWMIYA E)	Data Flow Diagram visualizes how data moves inside the application through pictorial representations using shapes and symbols.	20 October 2022
Technology Architecture (ASSIGNED TO TEAM MEMBER: R V SHAMITHA)	This is where all the technological requirements, including both Software and Hardware facilities are sequenced in a proper format like in the Solution Architecture.	20 October 2022

Preparation Milestone and Activity List (ASSIGNED TO TEAM MEMBERS: R V SHAMITHA & E SOWMIYA)	Milestones provide a way to more accurately estimate the time it will take to complete your project by marking important dates and events, making them essential for precise project planning and scheduling	10 November 2022
Sprint Delivery Plan (ASSIGNED TO TEAM MEMBERs: R V SHAMITHA & E SOWMIYA)	Sprint Delivery Plan means to split the project output to phases called sprints to deliver accordingly at scheduled time. This may be very much useful as each Sprints are carefully reviewed for perfection. This reduces flaws during submission.	10 November 2022

6.2 SPRINT DELIVERY SCHEDULE

Sprint	Total Points	Duration	Sprint Start Date	Sprint End date	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	17 Nov 2022	20	17 Nov 2022

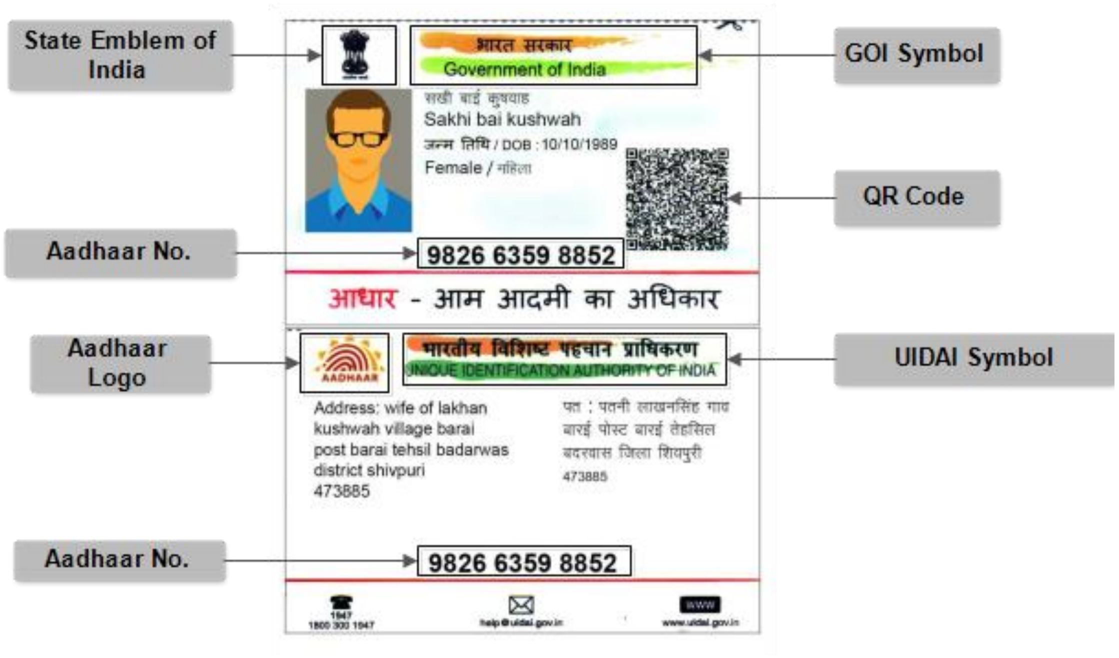
7.CODING & SOLUTIONING

7.1 FEATURE 1

An Aadhaar card is a unique number issued to every citizen in India and is a centralized and universal identification number. The Aadhaar card is a biometric document that stores an individual’s details in a government database and is fast becoming the government’s base for public welfare and citizen services.

Every document has some unique features which makes it different from other documents. Distinguishable features of Aadhaar Card are -

1. State Emblem of India
2. GOI Symbol
3. QR Code
4. Aadhaar Logo
5. UIDAI Symbol



भारत सरकार
Government of India

सखी बाई कुशवाह
Sakhi bai kushwah
जन्म तिथि / DOB : 10/10/1989
Female / महिला

9826 6359 8852

आधार - आम आदमी का अधिकार

भारतीय विशिष्ट पहचान प्राधिकरण
UNIQUE IDENTIFICATION AUTHORITY OF INDIA

Address: wife of lakhan kushwah village barai post barai tehsil badarwas district shivpuri 473885

पता : पत्नी लाखनसिंह गाव बाराई पोस्ट बाराई तेहसिल बदरवास जिला शिवपुरी 473885

9826 6359 8852

1947 1800 300 1947 help@uidai.gov.in www.uidai.gov.in

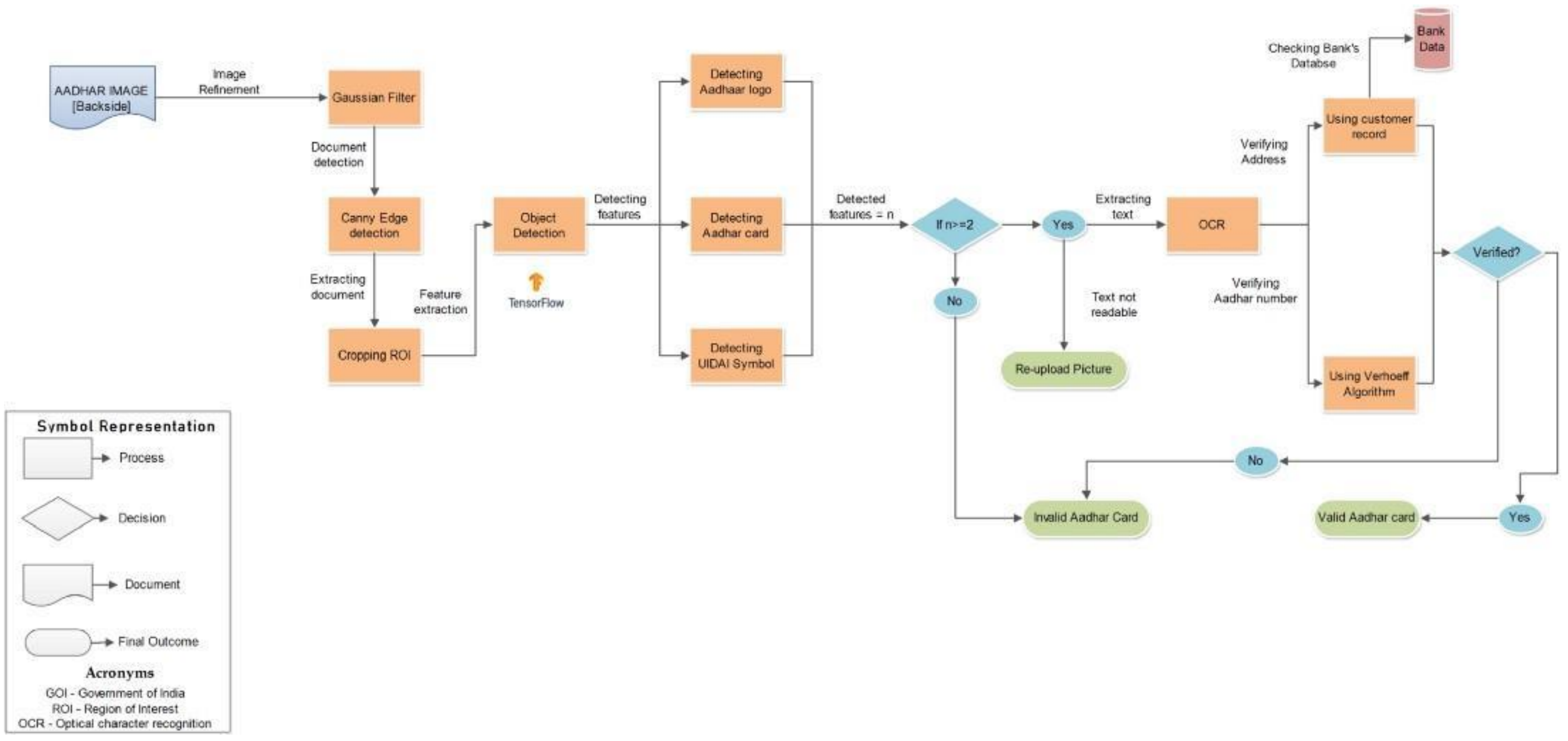
Unique Features of an Aadhaar Card [Front & Back]

STEP 1: Verification of Document

The Object Detection model will be used to verify whether the input document is a valid Aadhaar or not. If it is, we will proceed to the next step, or else the document will be declared invalid and the process will end.

STEP 2: Extracting Data using OCR

After it is verified that the submitted document is an Aadhaar then information present on Aadhaar will be extracted by the means of Optical Character Recognition (OCR). This information will mainly contain the address of the user, which can be verified from the bank's database.



7.3 DATABASE SCHEMA

```
CREATE TABLE users_details (
id SERIAL NOT NULL PRIMARY KEY ,
password varchar(50) NOT NULL,
email varchar(255) NOT NULL,
name varchar(255) NOT NULL,
phoneNo  BIGSERIAL NOT NULL,
address varchar(255) NOT NULL,
role int NOT NULL,
);

CREATE TABLE  users(
  id SERIAL NOT NULL PRIMARY KEY ,
  details_id int NOT NULL ,
  plasma id int NOT NULL ,
created_time timestamp NOT NULL,
CONSTRAINT user_userDetails FOREIGN KEY(details) REFERENCES users_details(id),
CONSTRAINT blood _plasme FOREIGN KEY( blood) REFERENCES library(id)
);

CREATE TABLE  Entry(
  id SERIAL NOT NULL PRIMARY KEY ,
  issue_Date timestamp NOT NULL,
            returned_Date timestamp ,
   plasma id int NOT NULL ,
  user_Id int NOT NULL ,
  status int NOT NULL,
CONSTRAINT entry_book FOREIGN KEY(book_id ) REFERENCES book_details(id),
CONSTRAINT entry_user FOREIGN KEY(user_Id) REFERENCES users(id)
);
```

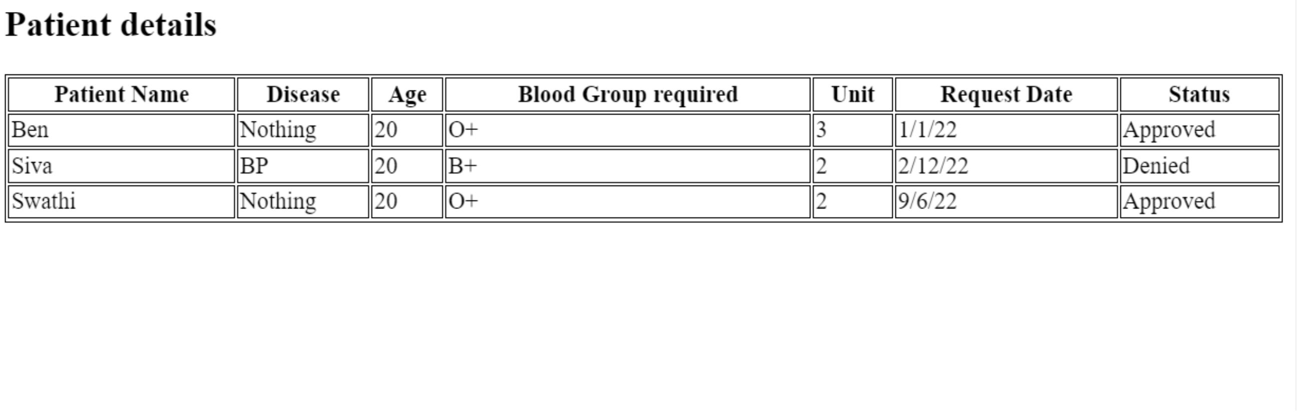
8 .TESTING

Test Case 1 :

Patient Name : Ben
Disease : Nothing
Age : 20
Blood Group Required : O+
Unit : 2
Request Date : 1/1/22
Status : Approved

Test Case 2 :
Patient Name : siva
Disease : Nothing
Age : 20
Blood Group Required : B+
Unit : 2
Request Date : 2/12/22
Status : Denied

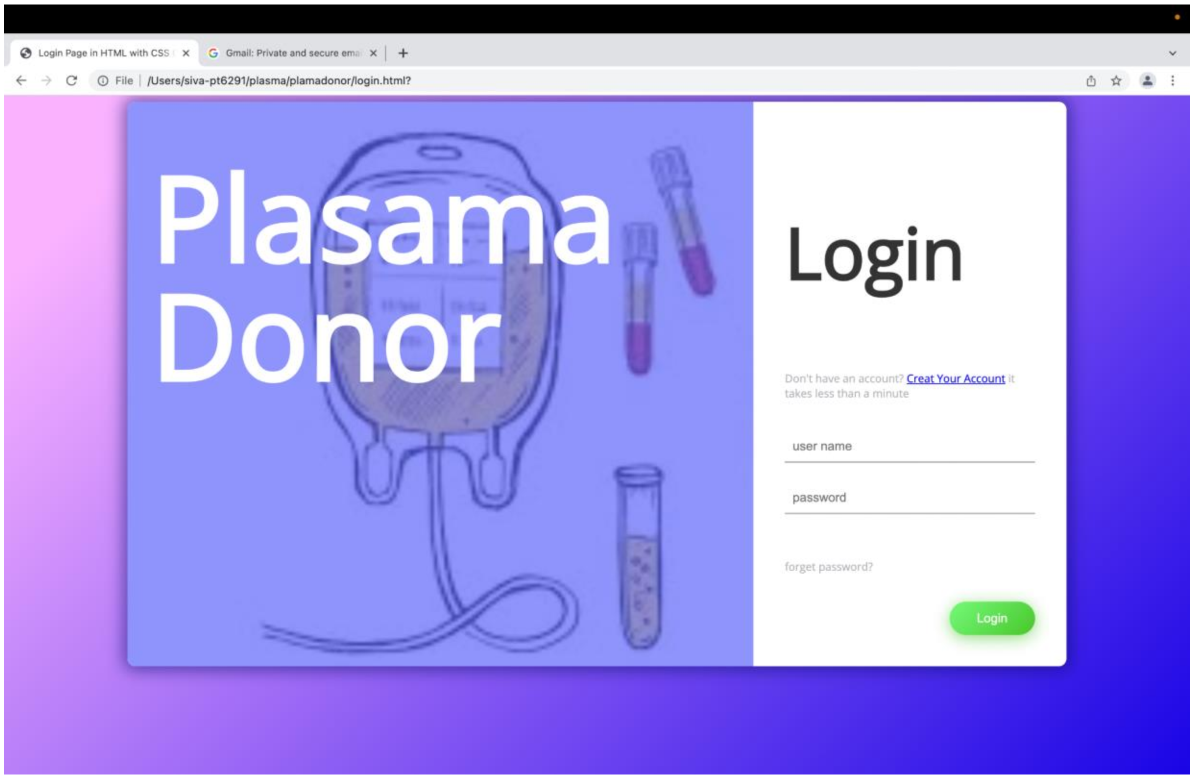
Test Case 3 :
Patient Name : Swathi
Disease : Nothing
Age : 20
Blood Group Required : O+
Unit : 2
Request Date : 2/2/22
Status : Denied



Patient details

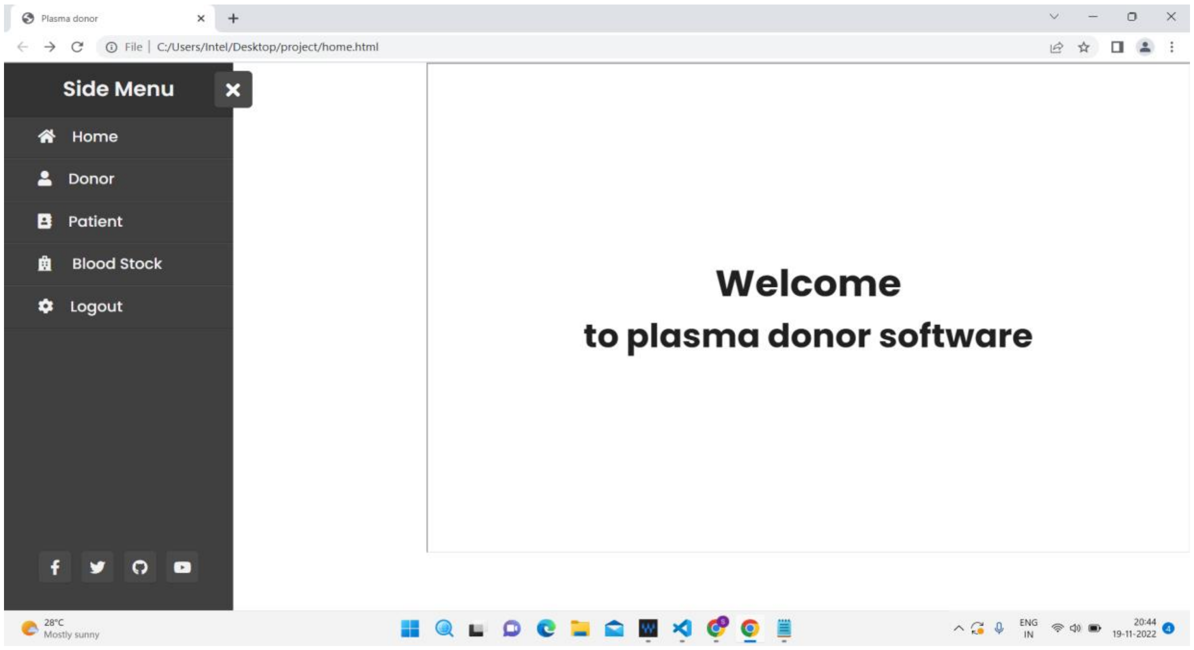
Patient Name	Disease	Age	Blood Group required	Unit	Request Date	Status
Ben	Nothing	20	O+	3	1/1/22	Approved
Siva	BP	20	B+	2	2/12/22	Denied
Swathi	Nothing	20	O+	2	9/6/22	Approved

RESULTS



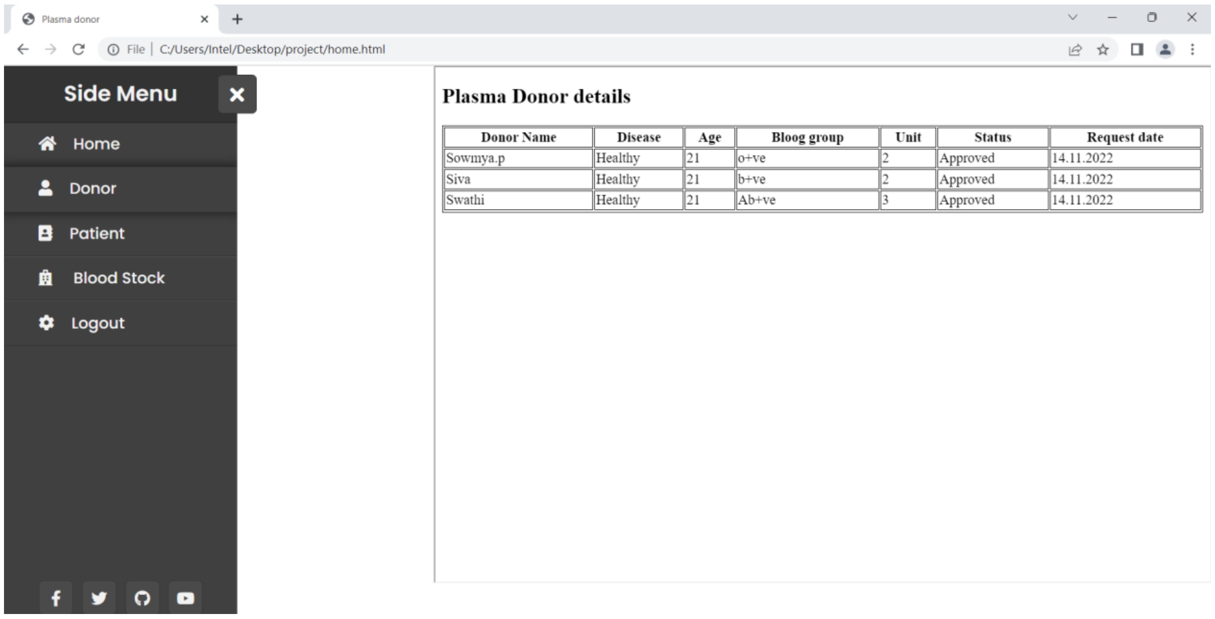
Login Page in HTML with CSS
Gmail: Private and secure ema
File | /Users/siva-pt6291/plasma/plamadonor/login.html?
Plasama Donor
Login
Don't have an account? Creat Your Account it takes less than a minute
user name
password
forget password?
Login

Fig 9.1 Login



Welcome
to plasma donor software

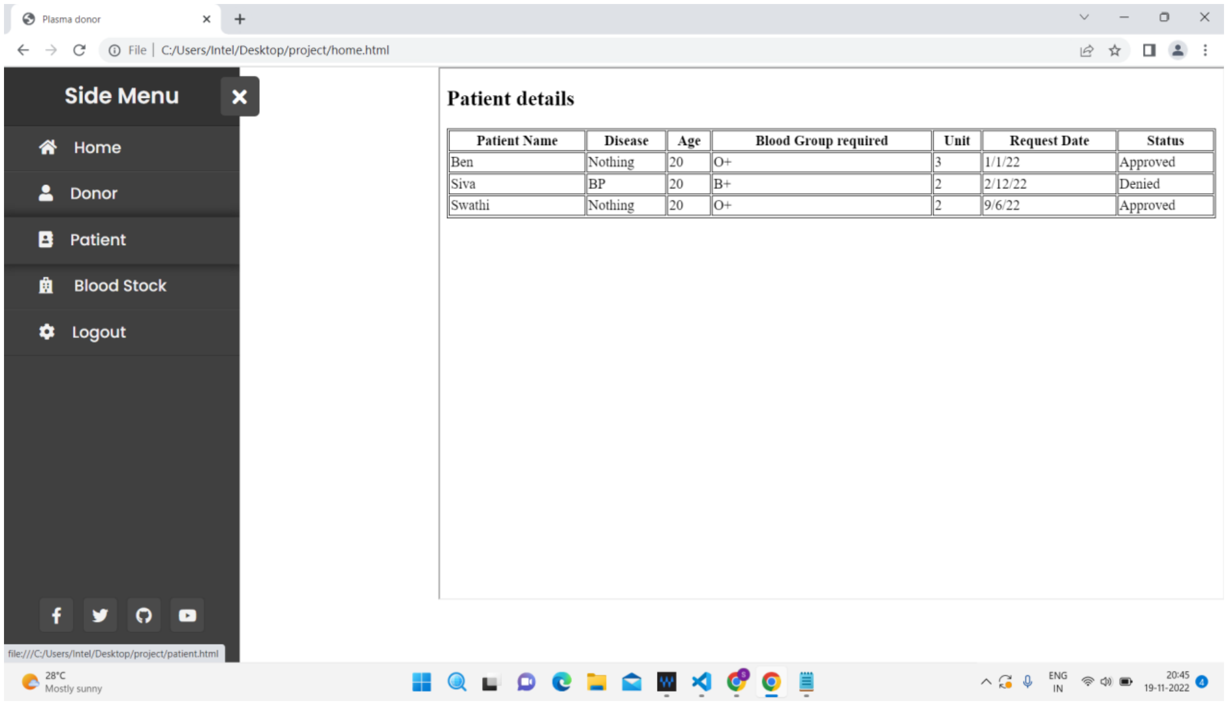
Fig 9.1 Home



Plasma Donor details

Donor Name	Disease	Age	Bloog group	Unit	Status	Request date
Sowmya.p	Healthy	21	o+ve	2	Approved	14.11.2022
Siva	Healthy	21	b+ve	2	Approved	14.11.2022
Swathi	Healthy	21	Ab+ve	3	Approved	14.11.2022

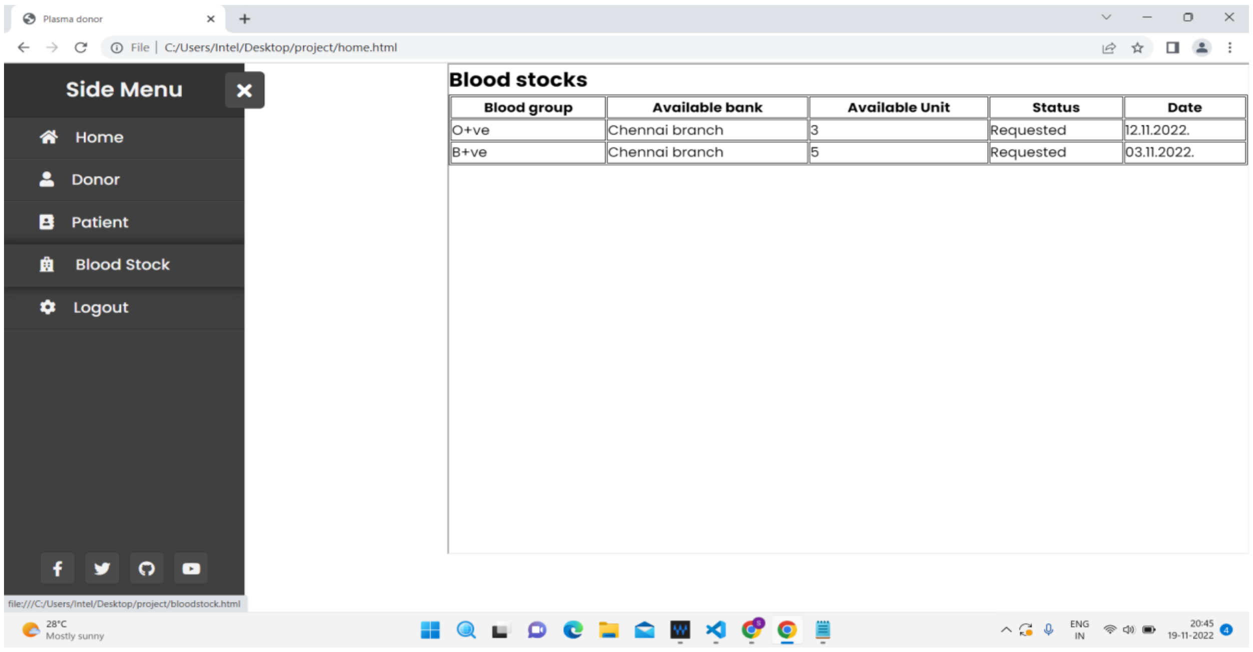
Fig 9.3 Donor


Plasma donor
File | C:/Users/Intel/Desktop/project/home.html
Side Menu
Home
Donor
Patient
Blood Stock
Logout
file:///C:/Users/Intel/Desktop/project/patient.html
28°C
Mostly sunny
ENG
IN
20:45
19-11-2022

Patient details

Patient Name	Disease	Age	Blood Group required	Unit	Request Date	Status
Ben	Nothing	20	O+	3	1/1/22	Approved
Siva	BP	20	B+	2	2/12/22	Denied
Swathi	Nothing	20	O+	2	9/6/22	Approved

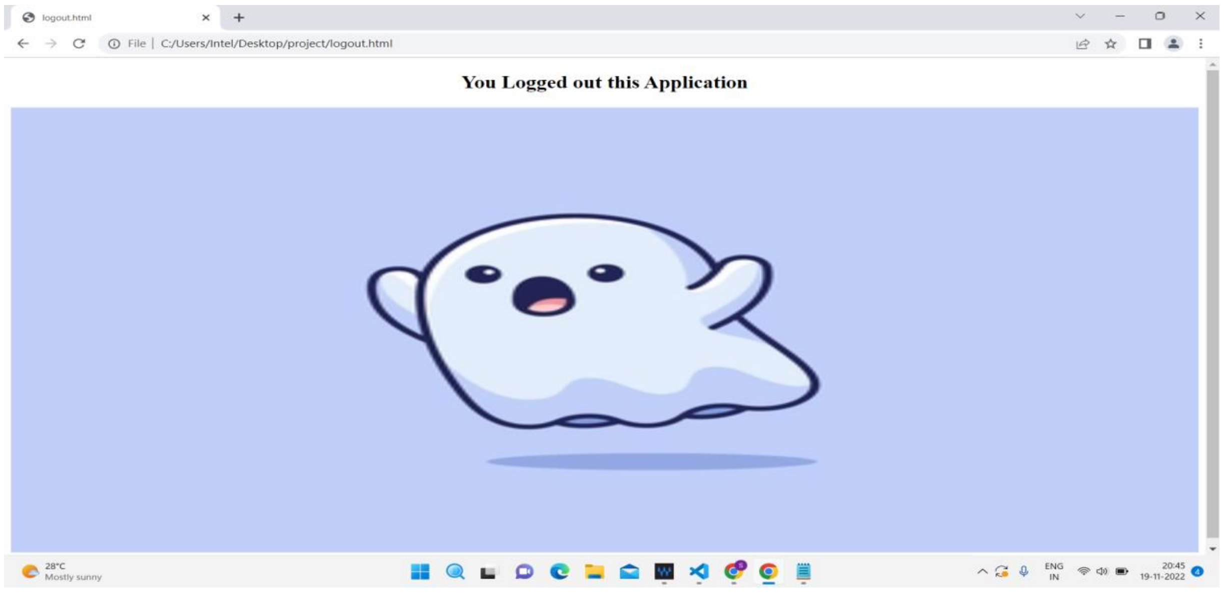
Fig 9.4 Patient


Plasma donor
File | C:/Users/Intel/Desktop/project/home.html
Side Menu
Home
Donor
Patient
Blood Stock
Logout
file:///C:/Users/Intel/Desktop/project/bloodstock.html
28°C
Mostly sunny
ENG
IN
20:45
19-11-2022

Blood stocks

Blood group	Available bank	Available Unit	Status	Date
O+ve	Chennai branch	3	Requested	12.11.2022.
B+ve	Chennai branch	5	Requested	03.11.2022.

Fig 9.5 Blood Stock



You Logged out this Application

Fig 9.6 Logout

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

- User friendly.
- Helps connect patients with donors .
- Develops the medical industry.

DISADVANTAGES

- Lack of verification might lead to medical accidents.

11.CONCLUSION

Our project is only a humble venture to satisfy the needs of a realistic Plasma Donor Application. The objective of software planning is to provide a framework that enables the working of Plasma Donor.We are grateful that we got the opportunity to understand how software is designed in the real world. We got to design a small software through which we learnt a lot.This project shed light on how this subject is useful in practical terms. The purpose of this theory subject was justified.

12.FUTURE SCOPE

The sole purpose of this project is to develop a computer system that will link all donors, control a blood transfusion service and create a database to hold data on stocks of blood in each area. Furthermore, people will be able to see which patients need blood supplies via the android

13.APPENDIX

SOURCE CODE

```
from flask import *
from flask import Flask, render_template, request, redirect, url_for, session
from twilio.rest import Client
from werkzeug.utils import secure_filename
import ibm_db
import re
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
import csv
app=Flask(__name__)
app.secret_key="don't share"
myconn=ibm_db.connect('DATABASE=bludb;HOSTNAME=fbd88901-ebdb-4a4f-a32e-
9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32731;SECURITY=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=ftq82843;PWD=u4VCEInPvFw43qix', '', ''
        )
@app.route("/signup")
def signup():
    return render_template("signup.html")



@app.route('/register1', methods =['GET', 'POST'])
def register1():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        query = 'SELECT * FROM admin WHERE username =?;'
        stmt=ibm_db.prepare(myconn,query)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            query = "INSERT INTO ADMIN VALUES (?,?,?)"
            stmt=ibm_db.prepare(myconn,query)
            ibm_db.bind_param(stmt,1,username)
            ibm_db.bind_param(stmt,2,email)
            ibm_db.bind_param(stmt,3,password)
            ibm_db.execute(stmt)
            msg = 'You have successfully registered !'
            return render_template('login.html', msg = msg)


@app.route("/login",methods=['GET','POST'])
def login():
            if request.method=="POST":
                    Username=request.form['Username']
                    Password=request.form['Password']
                    query="select * from admin where Username=? and password=?;"
                    stmt=ibm_db.prepare(myconn, query)
                    ibm_db.bind_param(stmt, 1, Username)
                    ibm_db.bind_param(stmt, 2, Password)
                    ibm_db.execute(stmt)
                    data=ibm_db.fetch_assoc(stmt)
                    if data:
                            session['logged in']=True
                            flash("Login Successfully")
                            return  render_template('info.html')

                    else:
                            flash("Incorrect Username or Password")
            return render_template("login.html")


@app.route("/")
@app.route("/bloodbank")
def blood bank():
                    return render_template("bloodbank.html")



@app.route("/home")
def home():

            query="select count(*) from donor where status=1"
            stmt = ibm_db.prepare(myconn, query)
            ibm_db.execute(stmt)
            data = ibm_db.fetch_tuple(stmt)
            return render_template("index.html",data=[data])



@app.route("/register",methods=['GET','POST'])
def register():
            if request.method=="POST":
                    name=request.form['name']
                    email=request.form['email']
                    phno=request.form['phno']
                    blood_group=request.form['blood_group']
                    weight=request.form['weight']
                    gender=request.form['gender']
                    dob=request.form['dob']
                    address=request.form['address']
                    adharno=request.form['adharno']
                    status=1

                    query="select * from donor where adhar no=(?);"
                    stmt = ibm_db.prepare(myconn, query)
                    ibm_db.bind_param(stmt, 1, adharno)
                    ibm_db.execute(stmt)
                    data = ibm_db.fetch_assoc(stmt)
                    if (data)==0:
                            query = "INSERT INTO donor
(NAME,EMAIL,PHNO,BLOOD_GROUP,WEIGHT,GENDER,DOB,ADDRESS,ADHARNO,ST
ATUS) values(?,?,?,?,?,?,?,?,?,?)"
                            stmt = ibm_db.prepare(myconn, query)
                            ibm_db.bind_param(stmt, 1, name)
                            ibm_db.bind_param(stmt, 2, email)
                            ibm_db.bind_param(stmt, 3, phno)
                            ibm_db.bind_param(stmt, 4, blood_group)
                            ibm_db.bind_param(stmt, 5, weight)
                            ibm_db.bind_param(stmt, 6, gender)
                            ibm_db.bind_param(stmt, 7, dob)
                            ibm_db.bind_param(stmt, 8, address)
                            ibm_db.bind_param(stmt, 9, adharno)
                            ibm_db.bind_param(stmt, 10, status)
                            ibm_db.execute(stmt)
                            msg = 'You have successfully Logged In!!'
```

```
                    return redirect(url_for('viewall'))

                else:
                    flash("Already Registered")
                return redirect(url_for('register'))
        else:
                return render_template("about.html")



@app.route("/view",methods=['GET','POST'])
def view():
        if not session.get('logged in'):
                return  render_template("login.html")
        query="select * from donor where status=1"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.execute(stmt)
        data=[]
        tuple = ibm_db.fetch_tuple(stmt)
        while tuple!=False:
                data.append(tuple)
                tuple=ibm_db.fetch_tuple(stmt)
        return render_template("view.html",data=data)



@app.route("/delete",methods=['GET','POST'])
def delete():
        if not session.get('logged in'):
                return  render_template("login.html")
        if request.method=="POST":
                id=request.form['delete']

                query="delete from donor where sno=?"
                stmt = ibm_db.prepare(myconn, query)
                ibm_db.bind_param(stmt, 1, id)
                ibm_db.commit(stmt)
                flash("Deleted Successfully")
                return redirect(url_for('view'))
```

```
@app.route("/edit",methods=['GET','POST'])
def edit():
        if not session.get('logged in'):
                return  render_template("login.html")
        if request.method=="POST":
                id=request.form['edit']

                query="select * from donor where sno=?"

                stmt = ibm_db.prepare(myconn, query)
                ibm_db.bind_param(stmt, 1, id)
                ibm_db.execute(stmt)
                data = ibm_db.fetch_tuple(stmt)
                return render_template("edit.html",data=data)



@app.route("/update",methods=['GET','POST'])
def update():
        if not session.get('loggedin'):
                return  render_template("login.html")
        if request.method=="POST":
                id=request.form['id']
                name=request.form['name']
                email=request.form['email']
                phno=request.form['phno']
                blood_group=request.form['blood_group']
                weight=request.form['weight']
                gender=request.form['gender']
                dob=request.form['dob']
                address=request.form['address']
                adharno=request.form['adharno']


                query = "INSERT INTO USER1 values(?,?,?,?,?,?,?,?,?,?)"
                stmt = ibm_db.prepare(myconn, query)
                ibm_db.bind_param(stmt, 1, id)
                ibm_db.bind_param(stmt, 2, name)
                ibm_db.bind_param(stmt, 3, email)
                ibm_db.bind_param(stmt, 4, phno)
                ibm_db.bind_param(stmt, 5, blood_group)
                ibm_db.bind_param(stmt, 6, weight)
                ibm_db.bind_param(stmt, 7, gender)
                ibm_db.bind_param(stmt, 8, dob)
                ibm_db.bind_param(stmt, 9, address)
                ibm_db.bind_param(stmt, 10, adharno)
                ibm_db.commit(stmt)
                return redirect(url_for('view'))



@app.route("/view2",methods=['GET','POST'])
def view2():

        query="select distinct blood_group from donor where status=1"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.execute(stmt)
        data=[]
        tuple = ibm_db.fetch_tuple(stmt)
        while tuple!=False:
                data.append(tuple)
                tuple=ibm_db.fetch_tuple(stmt)
        return render_template("select.html",data=data)


@app.route("/viewselected",methods=['GET','POST'])
def viewselected():
        blood_group=request.form['blood_group']
        query="select * from donor where blood_group= ? and status=1"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.bind_param(stmt, 1, blood_group)
        ibm_db.execute(stmt)
        data=[]
        tuple = ibm_db.fetch_tuple(stmt)
        while tuple!=False:
                data.append(tuple)
                tuple=ibm_db.fetch_tuple(stmt)
        return render_template("view2.html",data=data)


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

        query="select * from donor where status=1"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.execute(stmt)
        data=[]
        tuple = ibm_db.fetch_tuple(stmt)
        while tuple!=False:
                data.append(tuple)
                tuple=ibm_db.fetch_tuple(stmt)
        return render_template("view2.html",data=data)


@app.route("/")
@app.route("/send",methods=['GET','POST'])
def send():
        if request.method=="POST":
                id=request.form['send']

                query="select email from donor where sno=?"

                stmt = ibm_db.prepare(myconn, query)
                ibm_db.bind_param(stmt, 1, id)
                ibm_db.execute(stmt)
                data = ibm_db.fetch_assoc(stmt)
                print(data)
                message =
Mail(from_email='empire44440@gmail.com',to_emails=data['EMAIL'],subject='Hi
Donor!!!',html_content='<strong>We searching for a plasma donor and last we found your match is
safe with our patient.Kindly contact us at your convenience</strong>')
                try:
                        sg =
SendGridAPIClient('SG.ktA7YoLdR42S9fv1UsluhA.3wrD69UzKSrNPGyFwAwkt2s00X5zIF9iAf
Zptg4ejXU')
                        response = sg.send(message)
                        print(response.status_code)
                        print(response.body)
                        print(response.headers)
                except Exception as e:
                        print(e)
        return redirect('/viewall')


@app.route("/logout")
def logout():
    session['logged in']=False
    return render_template("index.html")




@app.route("/hold",methods=['GET','POST'])
def hold():
    if not session.get('logged in'):
        return  render_template("login.html")
    if request.method=="POST":
        id=request.form['hold']
        query="update donor set  status=0 where sno=?"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)

        return redirect(url_for('view'))




@app.route("/activate",methods=['GET','POST'])
def activate():
    if not session.get('logged in'):
        return  render_template("login.html")
    if request.method=="POST":
        id=request.form['hold']

        query="update donor set  status=1 where sno=?"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute3(stmt)

        return redirect(url_for('inactive'))



@app.route("/inactive",methods=['GET','POST'])
def inactive():
    if not session.get('logged in'):
        return  render_template("login.html")

    query="select * from donor where status=0"
    stmt = ibm_db.prepare(myconn, query)
```

```
        ibm_db.execute(stmt)
        data = ibm_db.fetch_tuple(stmt)
        print(data)
        return render_template('inactive.html',data=[data])



if __name__=="__main__":
        app.run(debug=True)
```

GitHub :

https://github.com/IBM-EPBL/IBM-Project-30909-1660192298

Project Demo Link :

https://drive.google.com/file/d/1y4-iTq52oQJZzw_1UaJjcB1QqdPP247l/view?usp=share_link