

A Project Report
on
PLASMA DONOR APPLICATION

Submitted in partial fulfillment for the award of the degree
of
BACHELOR OF TECHNOLOGY
in
INFORMATION TECHNOLOGY

Submitted by

TEAM ID: PNT2022TMID45773

SANTHANA G (813019205014)

NIVETHA K (813019205009)

PRITHIKA S (813019205010)

SANDHIYA (813019205012)

**NAALAIYA THIRAN – EXPERIENTIAL PROJECT BASED LEARNING
INITIATIVE**

**18CSE040L - PROFESSIONAL READINESS FOR INNOVATION, EMPLOYABILITY AND
ENTREPRENEURSHIP**

**DEPARTMENT OF BACHELOR OF TECHNOLOGY
(INFORMATION TECHNOLOGY)**

**OXFORD ENGINEERING COLLEGE, TRICHY
ANNA UNIVERSITY : 2019 - 2023**

November, 2022

TABLE OF CONTENTS

CHAPTER No.	TITLE	PAGE No.
1	INTRODUCTION	05
2	LITERATURE SURVEY	06
3	IDEATION & PROPOSED SOLUTION	10
4	REQUIREMENT ANALYSIS	15
5	PROJECT DESIGN	17
6	PROJECT PLANNING & SCHEDULING	22
7	CODING & SOLUTIONING	25
8	TESTING	28
9	RESULTS	31
10	ADVANTAGES & DISADVANTAGES	42
11	CONCLUSION	43
12	FUTURE SCOPE	44
13	APPENDIX	45

LIST OF TABLES

TABLE No.	TITLE	PAGE No.
2.1	Existing Problem	7
2.2	Problem statement definition	9
3.1	Proposed Solution table	13
4.1	Functional Requirement	15
4.2	Non-Functional Requirements	16
5.1	Components & Technologies	18
5.2	Application characteristics	19
5.3	User Stories	20
6.1	Sprint Planning & Estimation table	23
6.2	Sprint Delivery Schedule table	23
8.1	User Acceptance Testing	30

LIST OF FIGURES

FIGURE No.	TITLE	PAGE NO
3.1	Empathy map canvas	10
3.2	Team gathering, collection and select the problem statement	10
3.3	Brain storm	11
3.4	Idea listing and grouping	12
3.5	Idea Prioritization	12
3.6	Problem Solution Fit	14
5.1	Data FlowDiagrams	17
6.1	Burndown chart	24
6.2	Burnup chart	24
7.1	Database Schema	27

CHAPTER 1

INTRODUCTION

1.1 Project Overview

The Plasma Donor Application is the project developed for the donor and recipients has to get the Plasma at from their nearby hospital in their district. The Overview of the project is, First, the donor can enter their details and check their eligibility whether he can donate the Plasma or not. Second, the donor can enter their district, hospital name and date of donation. Third, the donor will receive the mail from that confirming the slot for booking of donation. After the details entered, the details of the donor will be stored on the IBM db2. The Hospital In charge can enter their details and hospital details. These details have stored on the IBM db2. After the donor's donation of Plasma, The Hospital In charge will update the donated Plasma list with certain necessary details. These list also have stored in the IBM db2. Next the Recipient can enter their details and he/she can request for the Plasma. If the Plasma has available in their district, it shows available hospitals to the recipients. The recipient can select the hospital and the recipient can receive the email as well as the hospital in charge can also receive the email about the requisition of the plasma. After the Recipient get the Plasma, the hospital having Plasma list have been updated and recipient receive the email about the confirmation of receiving of Plasma. The email plays a major role in this project. Because the project based on the live reality of the users involved in the Plasma donation and receiving process.

1.2 Purpose

During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low. Saving the donor information and helping the needy by notifying the current donors list, would be a helping hand. 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 using emails.

CHAPTER 2

LITERATURE SURVEY

2.1 Existing problem

S.No.	Title & Author	Year	Proposed System
1.	Convalescent Plasma Therapy: Data driven approach for finding the Best Plasma Donors. - M N Noorshidha, Dr.G.Aghila.	2021	The proposed system can help the health authorities approach the most probably efficient donors for the therapy rather than wasting time and test kits on a random donor who may or may not be eligible. The results from various ML algorithms trained on a synthetic clinical history dataset are examined and assessed as significant to some degree. The system has to be validated against real data to arrive at reasonable conclusions. This paper demonstrates how a data-driven solution is more beneficial than the conventional methods for donor search
2.	Nearest Blood & Plasma Donor Finding: A Machine Learning Approach. - Nayan Das, MD. Asif Iqbal	2021	Recently a life-threatening virus, COVID-19, spreading throughout the globe, which is more vulnerable for older people and those with pre-existing medical conditions. For them, plasma is needed to recover their illness. Our Purpose is to build a platform with clustering algorithms which will jointly help to provide the quickest solution to find blood or plasma donor. Closest blood or plasma donors of the same group in a particular area can be explored within less time and more efficiently.
3.	Implementation of Blood Donation Application Using Android Smartphone. - Ms. Pradnya Jagtap, Ms. Monika Mandale, Ms. Prachi Mhaske, Ms. Sonali Vidhate, Mr. S. S. Patil.	2015	The problem can be solve using the android application. The system will make sure that in case of need, the blood will be made available to the patient. There will be android app to make this communication faster. It aims to create an information about the donor and organization that are related to donating the blood. The methodology used to build this system uses GPS. The Proposed system will be used in Blood banks, Hospitals, for Donors and Requester whoever registers to the system.

4.	Enhanced Mobile Application Development for Plasma, Mother's Milk and Blood Banks - Dr. S. Brindha, Ms. D. Priya, Mr. S. Ajith Kannan, Mr. D. Joyal Victor, Mr. R. Gunachandran	2021	Most of the apps selected are available to help users search for donors. Few of the apps could not be installed and/or accessed. Of those that could be installed: half of them do not require any kind of authentication; a few of them are available in more than one language; half of them have a geographical restriction; around 60 % of them do not notify the user of BD events and requests; one, which is available for Android and iOS, can connect with a laboratory; around 45 % of them allow users to share information via social networks, and the majority of them do not provide BD recommendations. These results are used as a basis to provide app developers with certain recommendations. There is a need for better BD apps with more features in order to increase the number of volunteer donors.
5.	Instant Plasma Donor Recipient Connector Web Application - Kalpana Devi Guntoju, Tejaswini Jalli, Sreeja Uppala, Sanjay Malliseti	2022	In the recommendation system, the donor who wants to donate plasma can donate by uploading their COVID19 certificate and the blood bank can see the donors who have uploaded the certificate and they can make a request to the donor and the hospital can register/login and search for the necessary things. plasma from a blood bank and they can request a blood bank and obtain plasma from the blood bank.

Table 2.1 - Existing problem

2.2. References:

1. Noorshidha.M & Aghila.G.. (2021). Convalescent Plasma Therapy: Data driven approach for finding the Best Plasma Donors.432 439.10.1109/ICAIS50930.2021.939601
2. N. Das and M. A. Iqbal, "Nearest Blood & Plasma Donor Finding: A Machine Learning Approach," 2020 23rd International Conference on Computer and Information Technology (ICCIT), 2020, pp. 1-6, Doi: 10.1109/ICCIT51783.2020.9392739.

3. Ms. Pradnya Jagtap, Ms. Monika Mandale, Ms. Prachi Mhaske, Ms. Sonali Vidhate, Mr. S. S. Patil. "Implementation Of Blood Donation Application Using Android Smartphone."
Doi:10.51397

4. Dr. S. Brindha, Ms. D. Priya, Mr. S. Ajith Kannan, Mr. D. Joyal Victor, Mr. R. Gunachandran "ENHANCED MOBILE APPLICATION DEVELOPMENT FOR PLASMA, MOTHER'S MILK AND BLOOD BANKS"

5. Kalpana Devi Guntoju, Tejaswini Jalli, Sreeja Uppala, Sanjay Mallisetti " INSTANT PLASMA DONOR RECIPIENTCONNECTOR WEB APPLICATION"

2.3 Problem Statement Definition

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. As we develop plasma donor application for finding the plasma donor who is willing to donate their blood. Here the problem faced by the donor and the recipient can be bridged by using our Plasma donor application which collects the necessary data from the donor and make them able to the recipients.

I am	The donor willing to donate plasma. The recipient needs the plasma.
I am trying to	The recipient wants to reach the plasma available. The donor wants to check the eligibility to donate plasma and to donate it.
But	I need a platform to donate / receive the plasma.
Because	Due to the COVID-19, a greater number of people needs plasma. But the people don't know how to reach the donor / recipient.
Which makes me feel	We connect the donor and recipient through our platform named plasma donor application.

Table 2.2 - Problem statement definition

CHAPTER 3

IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

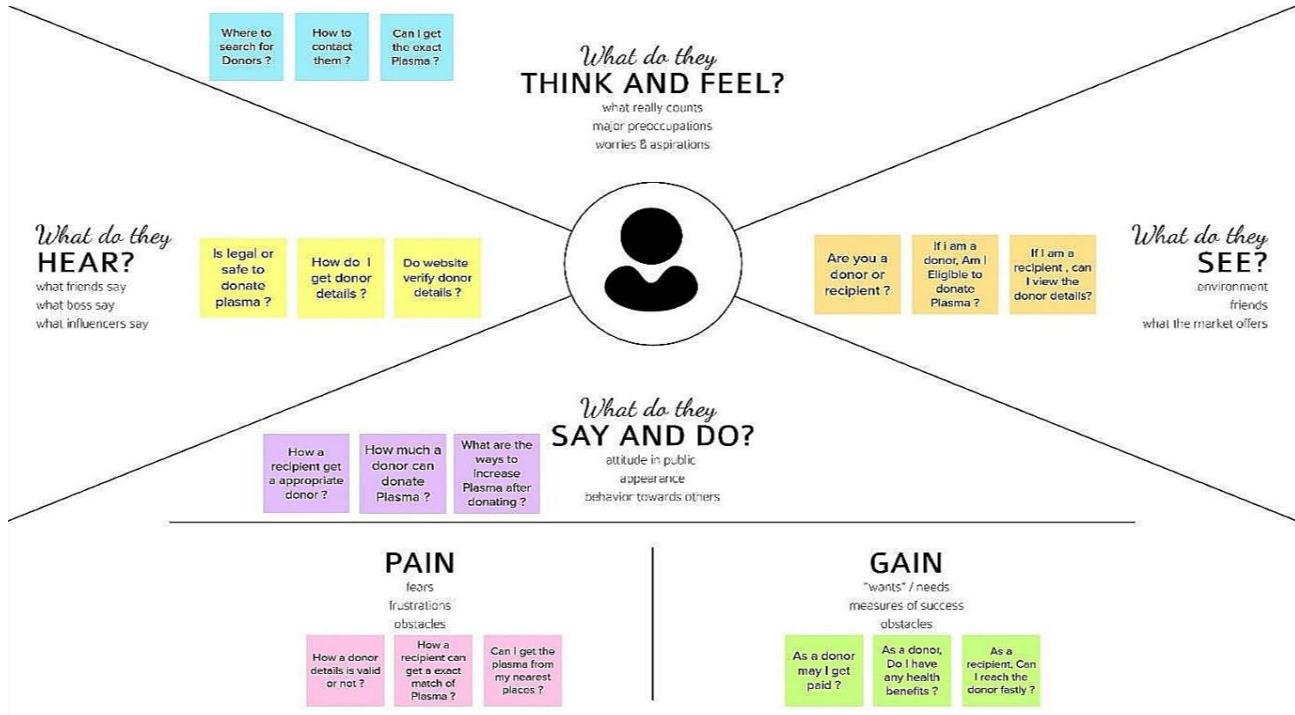


Figure 3.1 - Empathy map canvas

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community.

3.2 Ideation & Brainstorming:

Brainstorm & Idea Prioritization:

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



Figure 3.2 - Team gathering, collection and Select the problem statement

Idea prioritization is just a part of the idea management process. Having a structured idea management process and a systematic way of gathering, evaluating and prioritizing new ideas takes time. To make it work, the entire idea management process should be integrated to the everyday ways of working

Step 2: Brain Strom.

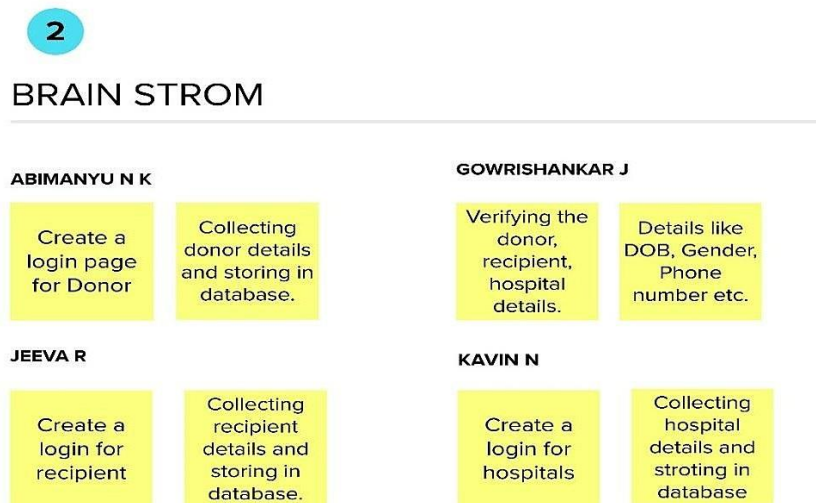


Figure 3.3 - Brain strom

Brainstorming is a group problem-solving method that involves the spontaneous contribution of creative ideas and solutions. This technique requires intensive, freewheeling discussion in which every member of the group is encouraged to think aloud and suggest as many ideas as possible based on their diverse knowledge.

Step 3: Idea listing and Grouping.

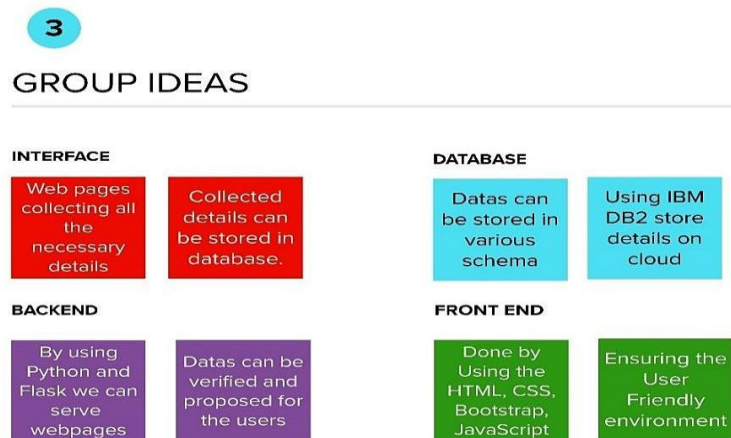


Figure 3.4 - Idea listing and grouping

A grouping is a set of people or things that have something in common. There were two main political groupings pressing for independence.

Step 4: Idea Prioritization:

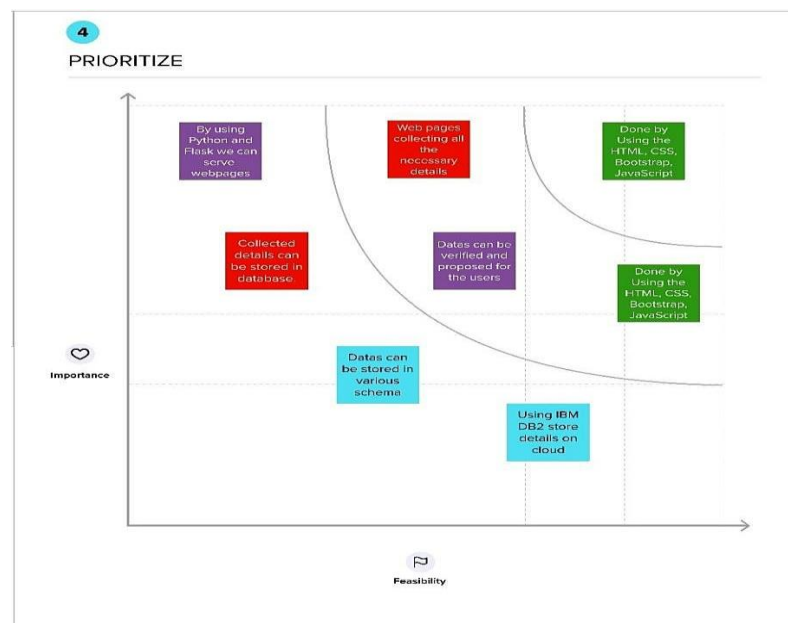


Figure 3.5 - Idea Prioritization

Idea prioritization is just a part of the idea management process. Having a structured idea management process and a systematic way of gathering, evaluating and prioritizing new ideas takes time. To make it work, the entire idea management process should be integrated to the everyday ways of working.

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	We develop plasma donor application for finding the plasma donor who is willing to donate their blood.
3.	Novelty / uniqueness	We store all the enough necessary user information on the cloud so it could be secured and checking, verifying its originality.
4.	Social impact/customer satisfaction	The application mostly helpful for the needy to get the plasma from their nearest location and nearby hospitals also.
5.	Business model (revenue model)	The Hospitals who use this application can be eligible to claim their revenue from the Government as they are only storing the plasma from the donor. Also, the donor should be paid and healthy foods also provided by the Government.
6.	Scalability of the solution	In COVID-19 the requirement of plasma became the high priority but the donor count become low. By saving the donor information and helping the needs by identifying the correct donor with their blood group. In regards to the problem faced an application is to be built with would take the donor details store them and make utilizing the needs. As we develop plasma donor application for finding the plasma donor who is willing to donate their blood. The donor can check their eligibility and available himself to the hospital and donate the plasma and receive their rewards from government and recipients also be satisfied.

Table 3.1 - Proposed Solution table

3.4 Problem Solution Fit:

1. CUSTOMER SEGMENT(S) CS <p>Adding features like above age of 21 can donate. Donor/Recipient/Hospitals can utilize this platform for their Plasma sharing process.</p>	6. CUSTOMER LIMITATIONS <small>EG. BUDGET, DEVICES</small> CL <p>Once blood is donated means, the donor could not able to donate the plasma for another 28 days. Our web application doesn't allow the users multiple times in a period of 28 days.</p>	5. AVAILABLE SOLUTIONS <small>PROS & CONS</small> AS <p>Available solutions are uncomfortable and needs a admin user so it is much needs a better solutions.</p>
2. PROBLEMS / PAINS <small>+ ITS FREQUENCY</small> PR <p>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.</p>	9. PROBLEM ROOT / CAUSE RC <p>The root/cause of this problem is COVID-19 and the donor count of the plasma becomes low. So this made the users to suffer a lot. 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.</p>	7. BEHAVIOR <small>+ ITS INTENSITY</small> BE <p>This web application is used to make donation and receiving process easier so that anyone can easily access and use it. Intensity of this application is to connect donor, hospital and recipient in single platform. donor can fill the interest form to donate.</p>
3. TRIGGERS TO ACT TR <p>Many people needs plasma for their treatment. Plasma donation really used for covid affected people for recovering faster.</p>	10. YOUR SOLUTION SL <p>Our web application is able to give the user friendly environment and doesn't needs an admin user for maintaining the website. Hospitals , Donors and Recipients can get more satisfied by using this application. We making the donors to enter their deals and providing their details to hospitals and recipients an get their plasma from nearest locations available.</p>	8. CHANNELS of BEHAVIOR CH <p>ONLINE Online web application allows user to make donation and receiving process easier.send request from anywhere anytime.</p>
4. EMOTIONS <small>BEFORE / AFTER</small> EM <p>Donor get fear, anxiety prior to donation give way to largely positive emotional states like clearing all their doubts in this web application.</p>		<p>OFFLINE Donors to visit nearby hospital and donate as well as receive plasma.</p>

Figure 3.6 - Problem Solution Fit

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem.

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirements (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form and Registration through Gmail
FR-2	User Confirmation	Confirmation via Email, Phone Number Confirmation via OTP
FR-3	Hospital Details	Need all nearby hospitals and their details.
FR-4	Communication through SMSs and Emails	Sending SMSs and Emails about the slot timings and necessities.
FR-5	Filling Feedback form	Filling the feedback form according to their treatment.

Table 4.1 - Functional Requirements

4.2 Non-functional Requirements:

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

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	This application will be an interface to the needy people and Plasma donation slots can be booked.
NFR-2	Security	We have providing security for data userenters and verifying the phone number or email.
NFR-3	Reliability	This website helps the users to register and book their slots for donating and receiving which reduces their wandering for plasma.
NFR-4	Performance	This website helps the users to provide the all details to know and prerequisites to the users.
NFR-5	Availability	The Availability of this users helps the people who are in urgent necessary of plasma for receiving or donating.
NFR-6	Scalability	This website helps the users only with the online but all other things has to deals with directly face to face.

Table 4.2 - Non-Functional Requirements

CHAPTER 5

PROJECT DESIGN

5.1 Data Flow Diagrams

1. Donor can enter their details and check their eligibility.
2. Hospital In-Charge enter their hospital details and register themselves.
3. Recipients can enter their details and book their slots.
4. After Donor's donation finished, In-charge update the details in database.
5. After Recipient's request for plasma, In-charge has to allocate the appropriate plasma for recipient.
6. After the process finished, all users enter their feedback to their appropriate Requests.
7. All the changes can enter into DB2.

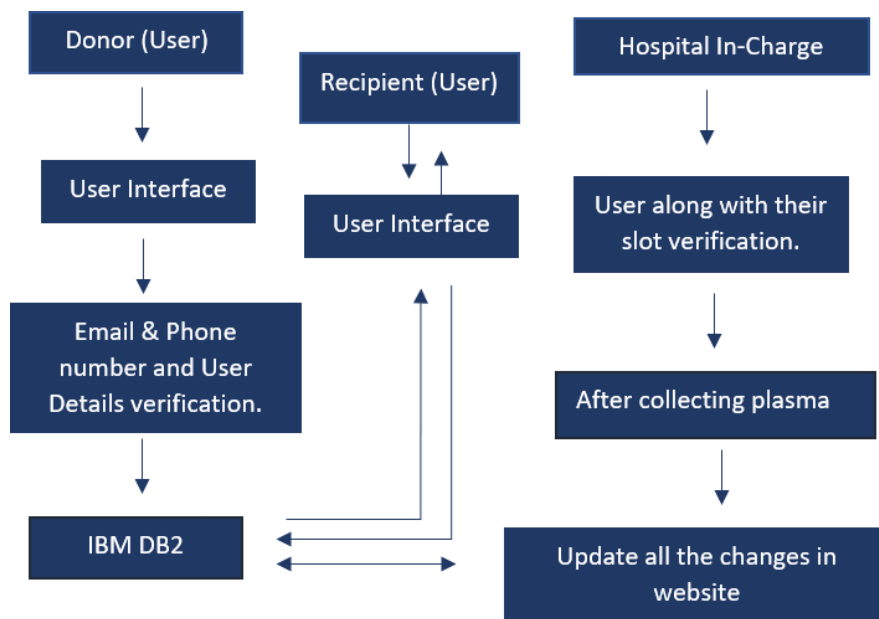


Figure 5.1 - Data Flow Diagrams

The route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically labeled with a short data name

5.2 Solution & Technical Architecture

The Architecture consists of storing the database on db2 and files on object storage.

Table-1: Components & Technologies:

S.No.	Component	Description	Technology
1.	User Interface	User interacts with application with three kinds of logins.	HTML, CSS, JavaScript, Bootstrap.
2.	Donor Registration and Login	Using Email/SMSs verification	Python and Python Flask, SendGrid
3.	Recipient Registration and Login	Using Email/SMSs verification	Python and Python Flask, SendGrid
4.	Hospital In-Charge Registration and Login	Using Email/SMSs verification	Python and Python Flask, SendGrid
5.	Database	Storing all the data in the database.	IBM DB2.
6.	Cloud Database	Database Service on Cloud	IBM DB2.
7.	File Storage	File storage requirements	IBM Cloud Object Storage
8.	Chat Bot / Assistant	Helps the users to give certain instructions based on the experience.	IBM Watson Assistant
9.	Hospital API-1	To used for provide nearby hospitals and details.	Hospitals API, etc.
10.	Aadhar API-1	To check the entered Aadhar number is valid one or not.	Aadhar API, etc.
11.	Infrastructure (Server / Cloud)	Application Deployment on System. Cloud Server Configuration: Docker / Kubernetes.	Docker, Kubernetes, etc.

Table 5.1 - Components & Technologies

Table-2: Application Characteristics:

S. No.	Characteristics	Description	Technology
1.	Open-Source Frameworks	Used Web technologies for Front end and Back end.	HTML, CSS, JS, Python Flask
2.	Security Implementations	User verification through Email Service	SendGrid
3.	Scalable Architecture	Run the app in Local and Cloud System	Docker and Kubernetes
4.	Availability	Justify the availability of application (e.g., use of load balancers, distributed servers etc.)	Docker, IBM Cloud
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	IBM Cloud, KubernetesCluster, Container Registry.

Table 5.2 - Application characteristics

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Donor / Recipient /Hospital InCharge (Mobile/Desktop user)	Registration	USN-1	As a user,I can register for the application by entering my email, password, and Confirming my password.	I can access my account/ dashboard	High	Sprint-1

		USN-2	As a user, I will receive confirmation email or SMS once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through gmail and Phone Number.	I can register & access the dashboard with or any kind of Login	Medium	Sprint-2
	Login	USN-4	As a user, I can log into the application by entering email or phone number & password		High	Sprint-1
Donor / Recipient / Hospital In-Charge (Webuser)	Dashboard	USN-5	As a user, I can be allowed to choose the three options like Donor, Recipient and Hospital In-Charge.	I am a Donor or and need to access only Donor registration with my credentials	Medium	Sprint-3
		USN-6		I am a Recipient and need to access only Recipient registration with my credentials	Medium	Sprint-3
		USN-7		I am a Hospital In-Charge and access only In-Charge registration with my hospital's credentials	Medium	Sprint-3

Donor	Donor's Page	USN-8	As a Donor,I can enter my details andcheck my eligibility,and book my slot for donation	I am donor, I canget the slot timings and nearby hospital details.	High	Sprint-4
Recipient	Recipient's Page	USN-9	As a Recipient, I canenter my details andbook my slot in a hospital as any nearby.	I am a recipient; I can get the appropriate Plasma present in nearby areas.	High	Sprint-4
Hospital In-Charge	Hospital In-Charge Page	USN-10	As a Hospital In-Charge, I can enter my details and hospital details as per the conditions.	I am a Hospital In-Charge; I can check the u ser credentials and do my process.	High	Sprint-4
All users (Donor, Recipient, Hospital In-Charge)	At last feedback page	USN-11	Finally, all users enter their feedback and receive feedbacks and issues.	I am a user; I can send and receive queries thro ugh feedback pages.	Medium	Sprint-4

Table 5.3 - User Stories

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 can register for the application by entering my email, password, and confirming my password.	3	High	Abimanyu N K Gowrishankar J Jeeva R Kavin N
Sprint-1		USN-2	As a user, I will receive confirmation email or SMS once I have registered for the application	3	High	Abimanyu N K Gowrishankar J Jeeva R Kavin N
Sprint-2		USN-3	As a user, I can register for the application through Gmail and Phone Number.	2	Medium	Abimanyu N K Gowrishankar J Jeeva R Kavin N
Sprint-2	Login	USN-4	As a user, I can log into the application by entering email or phone number & password	3	High	Abimanyu N K Gowrishankar J Jeeva R Kavin N
Sprint-3	Dashboard	USN-5	As a user, I can be allowed to choose the three options like Donor, Recipient and Hospital In-Charge.	2	Medium	Abimanyu N K Gowrishankar J Jeeva R Kavin N
Sprint-3	Donor's Page	USN-6	As a Donor, I can enter my details and check my eligibility, and book my slot for donation	3	High	Abimanyu N K Gowrishankar J Jeeva R Kavin N
Sprint-3	Recipient's Page	USN-7	As a Recipient, I can enter my details and book my slot in a hospital as any nearby.	3	High	Abimanyu N K Gowrishankar J Jeeva R Kavin N
Sprint-3	Hospital In-Charge Page	USN-8	As a Hospital In-Charge, I can enter my details and hospital details as per the conditions.	3	High	Abimanyu N K Gowrishankar J Jeeva R Kavin N

Sprint-4	Connecting them	USN-9	Connecting the Hospital In-Chargewith donor with an E-mail.	2	Medium	Abimanyu N K Gowrishankar J Jeeva R Kavin N
Sprint-4	Connecting them	USN-10	Connecting the Hospital In-Chargewith recipient with an E-mail.	2	Medium	Abimanyu N K Gowrishankar J Jeeva R Kavin N
Sprint-4	At last feedback page	USN-11	Finally, all users enter their feedbackand receive feedbacks and issues.	1	Low	Abimanyu N K Gowrishankar J Jeeva R Kavin N

Table 6.1 - Sprint Planning & Estimation table

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	6	6 Days	24 Oct 2022	29 Oct 2022	6	29 Oct 2022
Sprint-2	5	6 Days	31 Oct 2022	05 Nov 2022	5	05 Nov 2022
Sprint-3	11	6 Days	07 Nov 2022	12 Nov 2022	11	12 Nov 2022
Sprint-4	5	6 Days	14 Nov 2022	19 Nov 2022	5	19 Nov 2022

Table 6.2 - Sprint Delivery Schedule table

6.3 Reports from JIRA

Burndown chart:

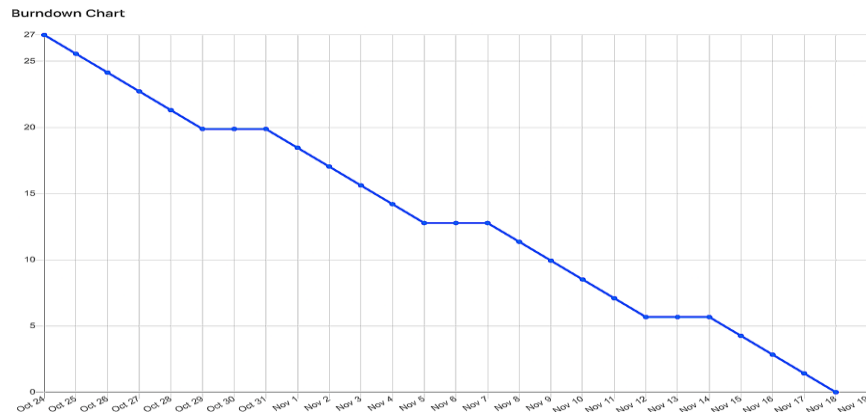


Figure 6.1 - Burndown chart

A burndown chart shows the amount of work that has been completed in an epic or sprint, and the total work remaining. Burndown charts are used to predict your team's likelihood of completing their work in the time available. They're also great for keeping the team aware of any scope creep that occurs.

Burnup chart:

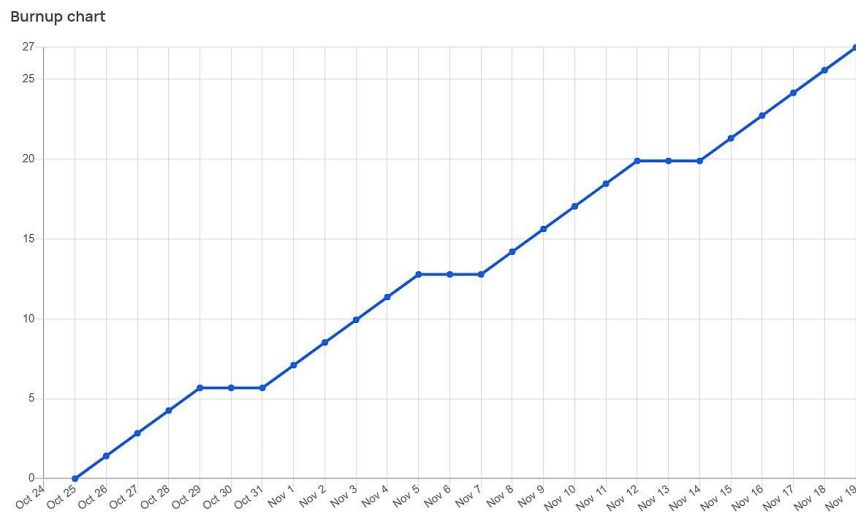


Figure 6.2 - Burnup chart

A burn up chart is a visual diagram commonly used on Agile projects to help measure progress. Agile burn up charts allow project managers and teams to quickly see how their workload is progressing and whether project completion is on schedule.

CHAPTER 7

CODING & SOLUTIONING

Feature 1: Python Flask

Python flask is the first feature that helps to complete this project. It allows the user to create local server and host the website in a local machine.

```
from flask import Flask,render_template,request,url_for,redirect,session
```

Here we import all the necessary features of this project involving in Python flask.

```
@app.route("/",methods=['POST','GET'])
def index():
    return render_template("indexpage.html")
```

Here we created a local client's own server which serves the .html pages to the users.

```
@app.route("/donorlogin",methods=['POST','GET'])
def donorlogin():
    if request.method=="POST":
        printDonorData(conn)
        text=request.form['text']
        password=str(request.form['password'])
        try:
            if dictionaryForEmailDonor[text] == (password):
                session['donorloggedin']=True
                session['donoremail']=text
                return redirect(url_for('donorhome'))
        except Exception as e:
            print(e)
            return "invalid email or password"
    return render_template("donorlogin.html")
```

Here we use the inputs from the html pages which has to be get by using request method in Python Flask. By validating the values from the database, we allow the user to access the home page.

render_template: Used for rendering html pages on browser.

url_for: Passing the control of the program to another function.

session: Creates a separate session for the individual user.

Feature 2: IBM db2

IBM db2 is the cloud database which stored the data we pass to the tables in the database. We can access the database by using unique username, password, host-name, etc. These details match with the DigiCertGlobalRootCA.crt certificate.

```
import ibm_db

try:
    conn=ibm_db.connect("DATABASE=database-name;
        HOSTNAME=unique-host-name;
        PORT=port-number;
        SECURITY=SSL;
        SSLServerCertificate=DigiCertGlobalRootCA.crt;
        PROTOCOL=TCPIP;
        UID=username;
        PWD=password;", "", "")
    print("Db connected")
except:
    print("Error")
```

Importing ibm_db from python and connecting using unique credentials provided for each cloud users.

```
status='' # data we need, to be stored in this str
query="SELECT donated FROM donationdetails WHERE email =
'{}'.format(session['donoremail'])"
stmt = ibm_db.exec_immediate(conn,query)
while ibm_db.fetch_row(stmt)!=False: #to store the db data
    status=(ibm_db.result(stmt,0))
print("status = ",status)
```

Here,

status is a local variable,

query is the appropriate query that we need to execute,

stmt is the statement that need to be processed by IBM db2 and IBM db2 returns unreadable output to this variable,

while iterating this **stmt** until it gets false, we could get the appropriate values that we need to be stored in **status** local variable,

Lastly printing the **status** variable to check with the data in table and the printed output as same.

Database Schema

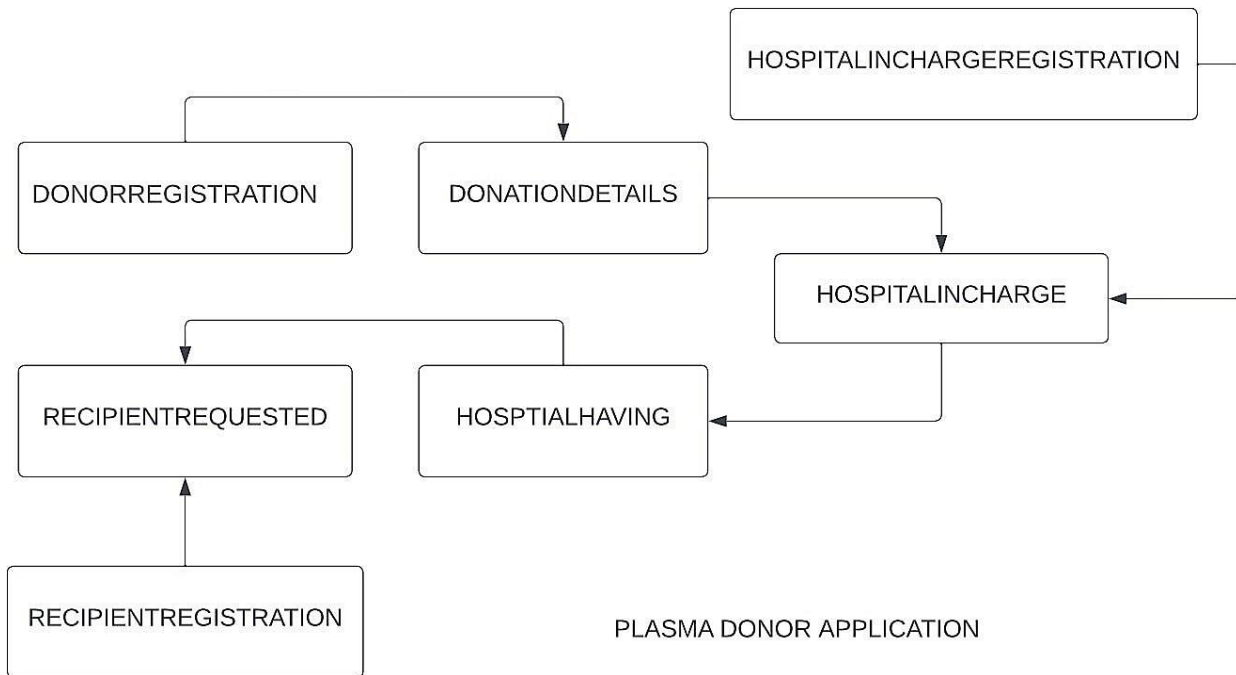


Figure 7.1 - Database Schema

A database schema is considered the “blueprint” of a database which describes how the data may relate to other tables or other data models. However, the schema does not actually contain data. A sample of data from a database at a single moment in time is known as a database instance.

1. **DONORREGISTRATION** get details and **DONATIONDETAILS** save the donors who have book for their slots.
2. **HOSPITALINCHARGEREGISTRATION** get details and **HOSPITALHAVING** save the donated plasma and available plasma to the recipients.
3. **RECIPIENTREGISTRATION** get details and **RECIPIENTREQUESTED** save the requested plasma from the **HOSPITALHAVING**.

CHAPTER 8

TESTING

8.1 Test Cases

The purpose of this part is to check, analyze and report the results of each test case. Each user story has taken as a test case. By analyzing the test cases, we made the coverage and open issues of the Plasma Donor Application. The list of test cases is listed below.

- I. As a user, I can register for the application by entering my email, password, and confirming my password.
- II. As a user, I will receive confirmation email or SMS once I have registered for the application
- III. As a user, I can register for the application through Gmail and Phone Number.
- IV. As a user, I can log into the application by entering email or phone number & password
- V. As a user, I can be allowed to choose the three options like Donor, Recipient and Hospital In-Charge.
- VI. As a Donor, I can enter my details and check my eligibility, and book my slot for donation
- VII. As a Recipient, I can enter my details and book my slot in a hospital as any nearby.
- VIII. As a Hospital In-Charge, I can enter my details and hospital details as per the conditions.
- IX. Connecting the Hospital In-Charge with donor with an E-mail.
- X. Connecting the Hospital In-Charge with recipient with an E-mail.
- XI. Finally, all users enter their feedback and receive feedbacks and issues.

8.2 User Acceptance Testing

Test Case No.	Description	Steps to Execute	Expected Result	Actual Result	Pass/ Fail/ Passed/ Partially	Additions/ Defects
1	As a user, I can register for the application by entering my email, password, and confirming my password.	Need to store the data in IBM DB2. After that move to the login page.	To be stored on the Db2	Stored on the Db2	Pass	No defects found
2	As a user, I will receive confirmation email or SMS once I have registered for the application	Sending OTP to register through the email.	To be received an email from the application	Received OTP through email	Pass	Username and the Subject of the email need to change.
3	As a user, I can register for the application through Gmail and Phone Number.	Sending OTP to register through the email.	To be received an email from the application	Received OTP through email	Pass	Username and the Subject of the email need to change.
4	As a user, I can log into the application by entering email or phone number & password	Check for both the email and phone number for logging in.	Need to be logged in using both email and phone number	Only Logged in using the email.	Passed partially	Need to include phone number for log in progress.
5	As a user, I can be allowed to choose the three options like Donor, Recipient and Hospital In-Charge	A page with three cards for three kinds of users.	This page has to be displayed before register or log in.	This page has displayed before register or log in.	Pass	No defects found.
6	As a Donor, I can enter my details and check my eligibility, and book my slot for donation	Store the data in the DB2 and send an email for both the hospital and donor.	Need to be receive an email from the application.	Received an email from application.	Pass	Username and the Subject of the email need to change.
7	As a Recipient, I can enter my details and book my slot in a hospital as any nearby.	Store the data in the DB2 and send an email for both the hospital and recipient.	Need to be receive an email from the application.	Received an email from application.	Pass	Username and the Subject of the email need to change.
8	As a Hospital In-Charge, I can enter my details and hospital details as per the conditions.	Store the details of the hospital in-charge user	Need to be store the data at registering the In-Charge user.	Stored the data at registering the In-Charge user.	Pass	No defects found.
9	Connecting the Hospital In-Charge with donor with an E-mail.	Send an email from the hospital in-charge to the donor.	Need to be done after the slot booking for confirming the slot.	Done after the slot booking for confirming the slot and received email.	Pass	Username and the Subject of the email need to change.

10	Connecting the Hospital In-Charge with recipient with an E-mail.	Send an email from the hospital in-charge to the recipient.	Need to be done after the slot booking for confirming the slot.	Done after the slot booking for confirming the slot and received email.	Pass	Username and the Subject of the email need to change.
11	Finally, all users enter their feedback and receive feedbacks and issues	Every user after the completion of this process need to fill a feedback page and store the feedback in the DB2	Need feedback page at last after the donating or receiving process.	Only Donor and Recipient can get the feedback page but not the Hospital In-Charge user.	Passed partially	Users can fill the form before the donating or receiving process

Table 8.1 - User Acceptance testing table

CHAPTER 9

RESULTS

9.1 Performance Metrics:

Donor Pages Screenshots

First, we get enter into the donor page.

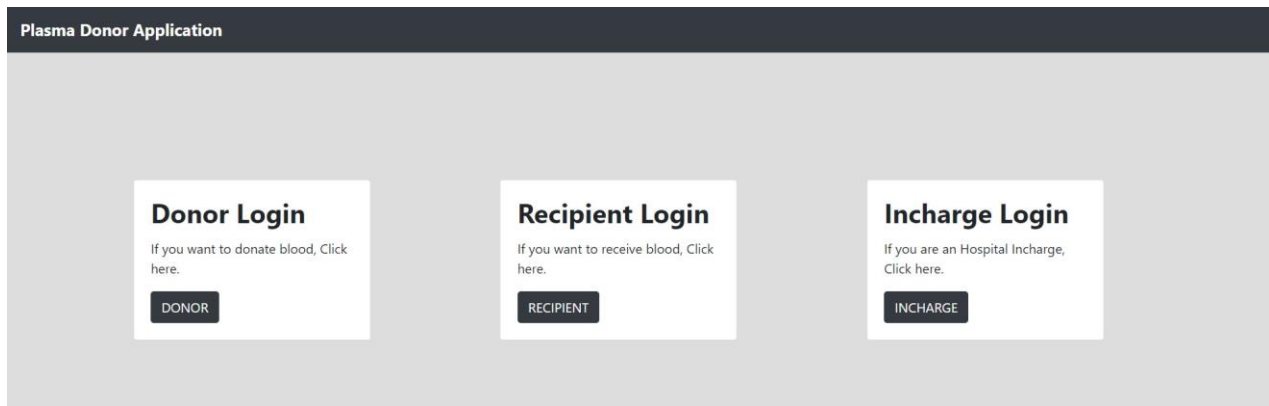


Figure 9.1 - Home page

Clicking on sign-up to register,

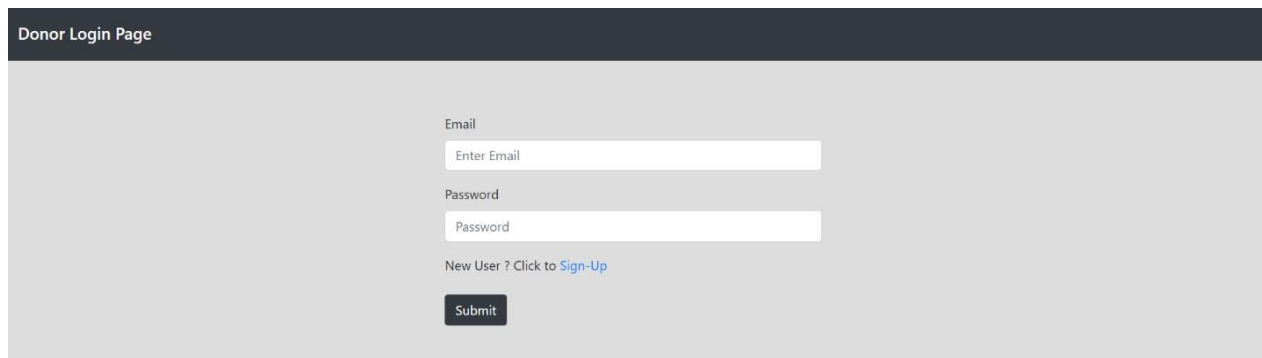
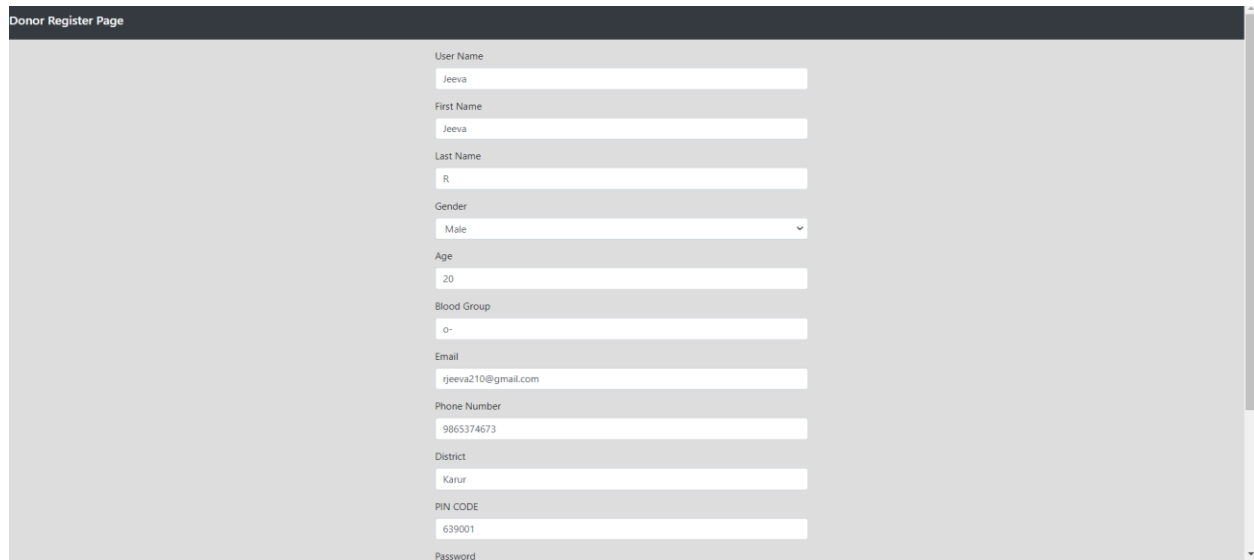


Figure 9.2 - Login page

Entering details of the donor,



Donor Register Page

User Name
Jeeva

First Name
Jeeva

Last Name
R

Gender
Male

Age
20

Blood Group
O-

Email
jeeva210@gmail.com

Phone Number
9865374673

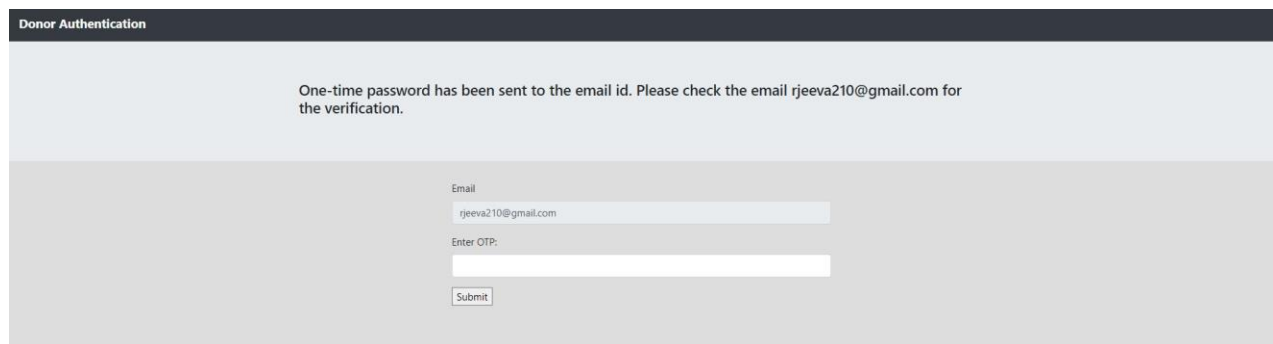
District
Karur

PIN CODE
639001

Password

Figure 9.3 - Register page

Getting OTP from Random user through user entered mail:



Donor Authentication

One-time password has been sent to the email id. Please check the email jeeva210@gmail.com for the verification.

Email
jeeva210@gmail.com

Enter OTP:

Submit

Figure 9.4 - OTP Authentication page

OTP from mail



Figure 9.5 - OTP received from email screenshot

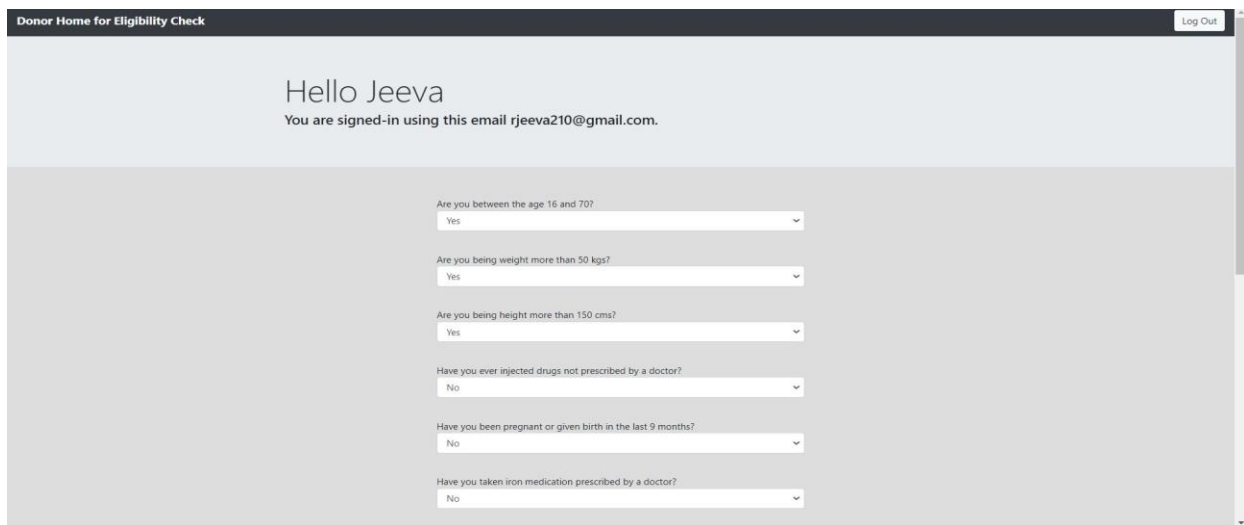
After entering OTP, Navigated to the Login page



The screenshot shows the 'Donor Login Page' with a dark header. The main content area is light gray and contains a login form. The form has two input fields: 'Email' with the value 'rjeeva210@gmail.com' and 'Password' with masked characters '****'. Below the password field is a link 'New User ? Click to Sign-Up' and a 'Submit' button.

Figure 9.6 - Login page

Checking Eligibility of the Donor After Login

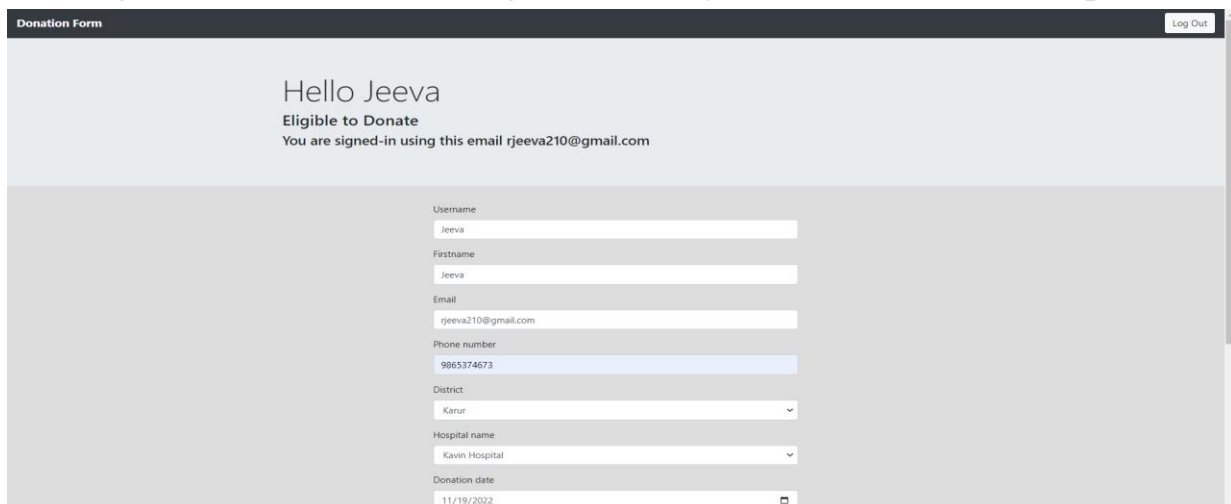


The screenshot shows the 'Donor Home for Eligibility Check' page. The header is dark with a 'Log Out' button. The main content area is light gray and displays a greeting 'Hello Jeeva' and a message 'You are signed-in using this email rjeeva210@gmail.com.'. Below this is a series of six eligibility questions, each with a dropdown menu:

- Are you between the age 16 and 70? (Yes)
- Are you being weight more than 50 kgs? (Yes)
- Are you being height more than 150 cms? (Yes)
- Have you ever injected drugs not prescribed by a doctor? (No)
- Have you been pregnant or given birth in the last 9 months? (No)
- Have you taken iron medication prescribed by a doctor? (No)

Figure 9.7 - Eligibility page

After Eligible check, we started to give necessary details for the donation process

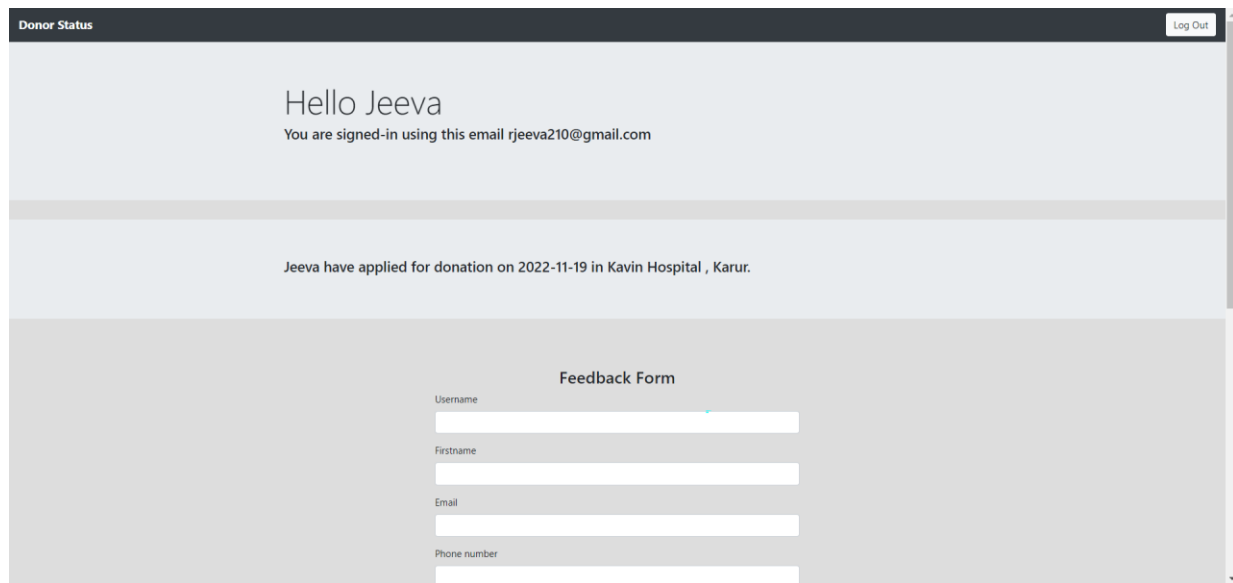


The screenshot shows the 'Donation Form' page. The header is dark with a 'Log Out' button. The main content area is light gray and displays a greeting 'Hello Jeeva' and a message 'Eligible to Donate' and 'You are signed-in using this email rjeeva210@gmail.com.'. Below this is a series of input fields for donation details:

- Username: Jeeva
- Firstname: Jeeva
- Email: rjeeva210@gmail.com
- Phone number: 9865374673
- District: Karur
- Hospital name: Kavin Hospital
- Donation date: 11/19/2022

Figure 9.8 - Donation Form page

After Entering Details, we get the status page with feedback form.



The screenshot shows a web application titled "Donor Status" in the top left corner. In the top right corner, there is a "Log Out" button. The main content area has a light blue header with the text "Hello Jeeva" and "You are signed-in using this email rjeeva210@gmail.com". Below this, a message states "Jeeva have applied for donation on 2022-11-19 in Kavin Hospital , Karur." The bottom section is titled "Feedback Form" and contains four input fields: "Username", "Firstname", "Email", and "Phone number".

Figure 9.9 - Donor Status Page

Also Receiving the confirmation mails too.



Figure 9.10 - Donor Slot booking email screenshot

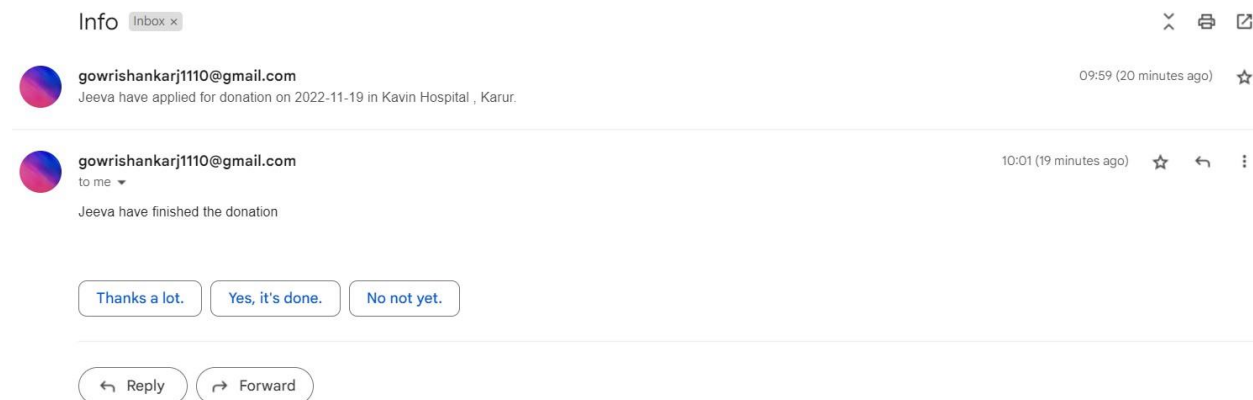


Figure 9.11 - Donor finished donation email screenshot

Recipient Pages Screenshots

First, we get enter into the recipient page.

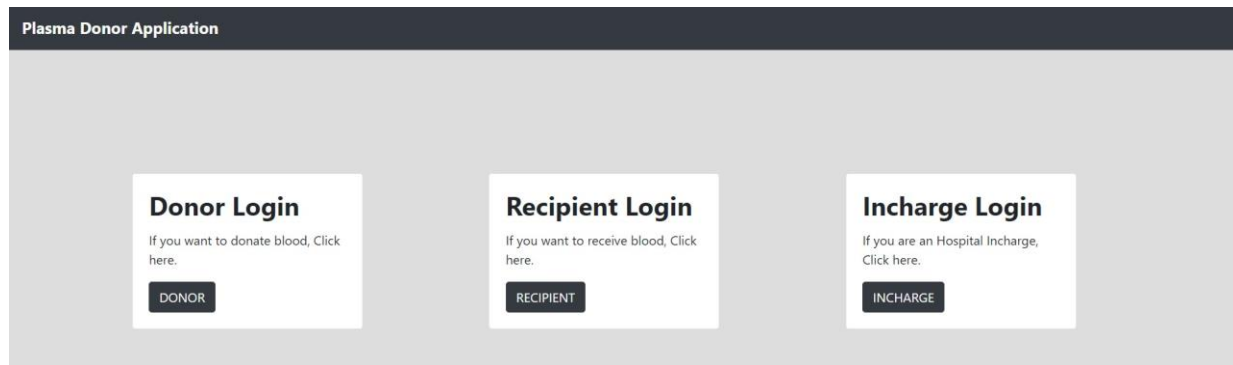


Figure 9.12 - Home page

Clicking on sign-up to register,

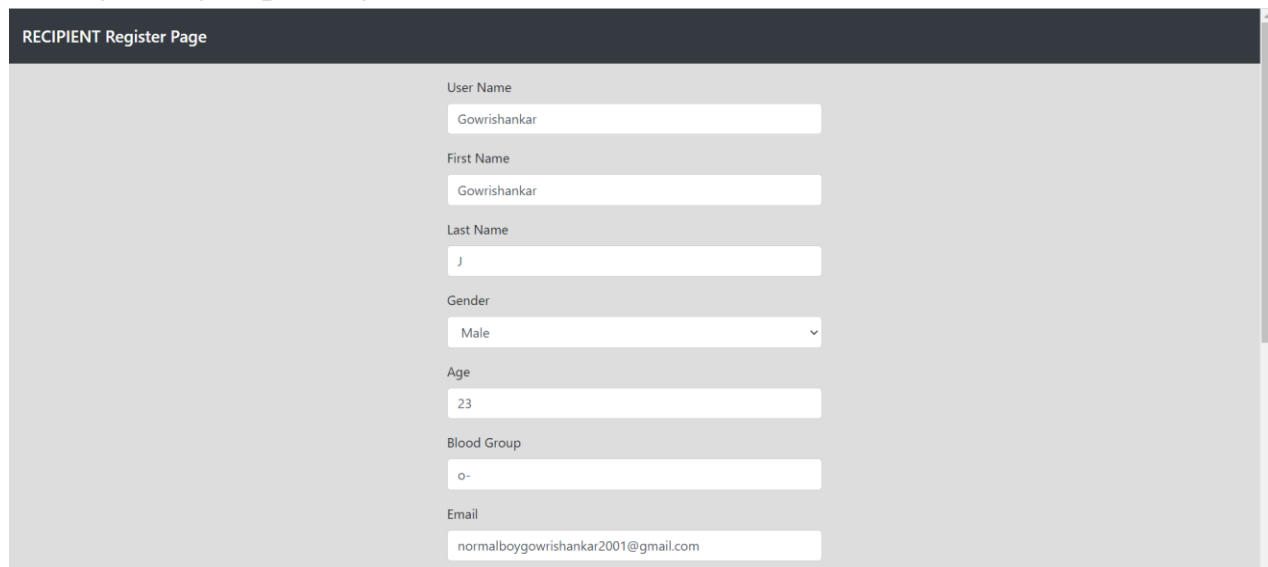
The screenshot shows the 'RECIPIENT Register Page'. It has a dark header with the title. Below the header, there is a form with the following fields: 'User Name' (Gowrishankar), 'First Name' (Gowrishankar), 'Last Name' (J), 'Gender' (Male), 'Age' (23), 'Blood Group' (O-), and 'Email' (normalboygowrishankar2001@gmail.com). Each field has a corresponding input box or dropdown menu.

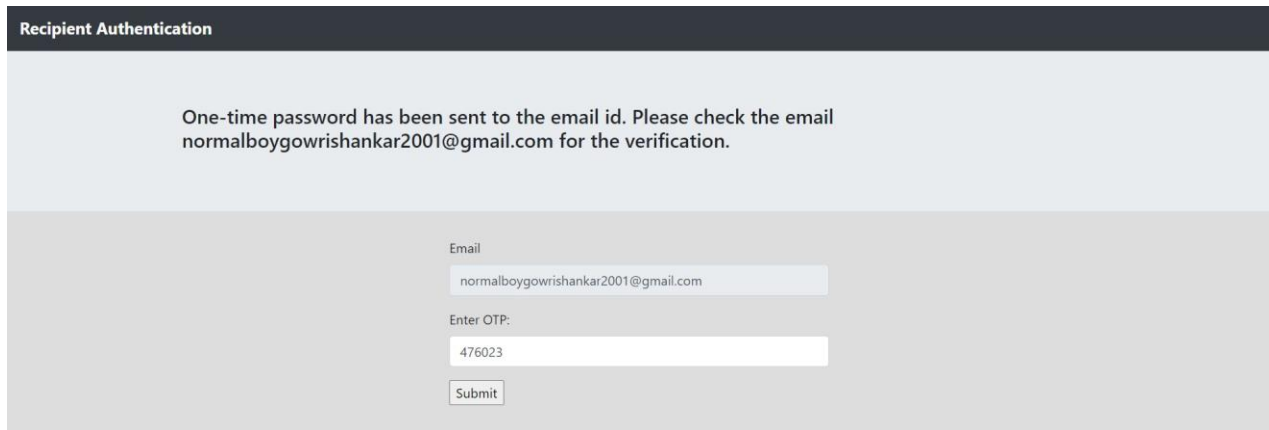
Figure 9.13 - Register page

After register we got an OTP,



Figure: 9.14 - OTP received from email screenshot

Entering the OTP from mail



Recipient Authentication

One-time password has been sent to the email id. Please check the email normalboygowrishankar2001@gmail.com for the verification.

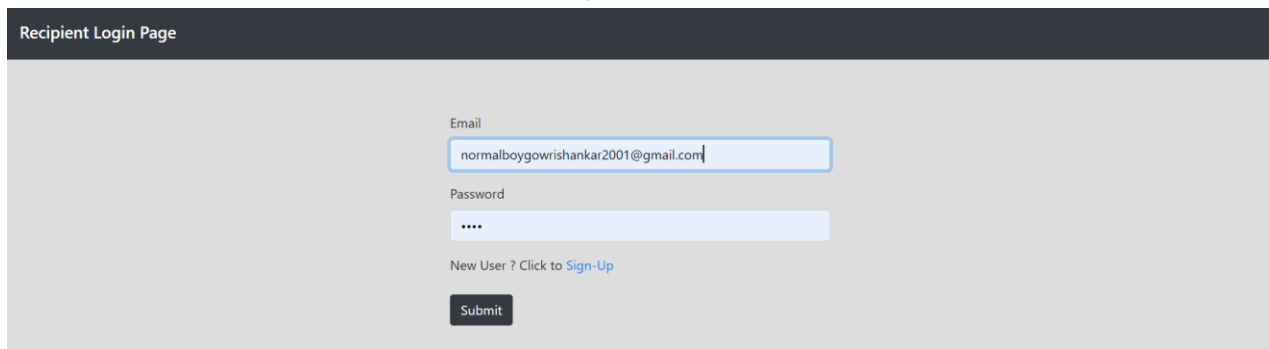
Email
normalboygowrishankar2001@gmail.com

Enter OTP:
476023

Submit

Figure 9.15 - Recipient Authentication page

After OTP authentication, we need to log in



Recipient Login Page

Email
normalboygowrishankar2001@gmail.com

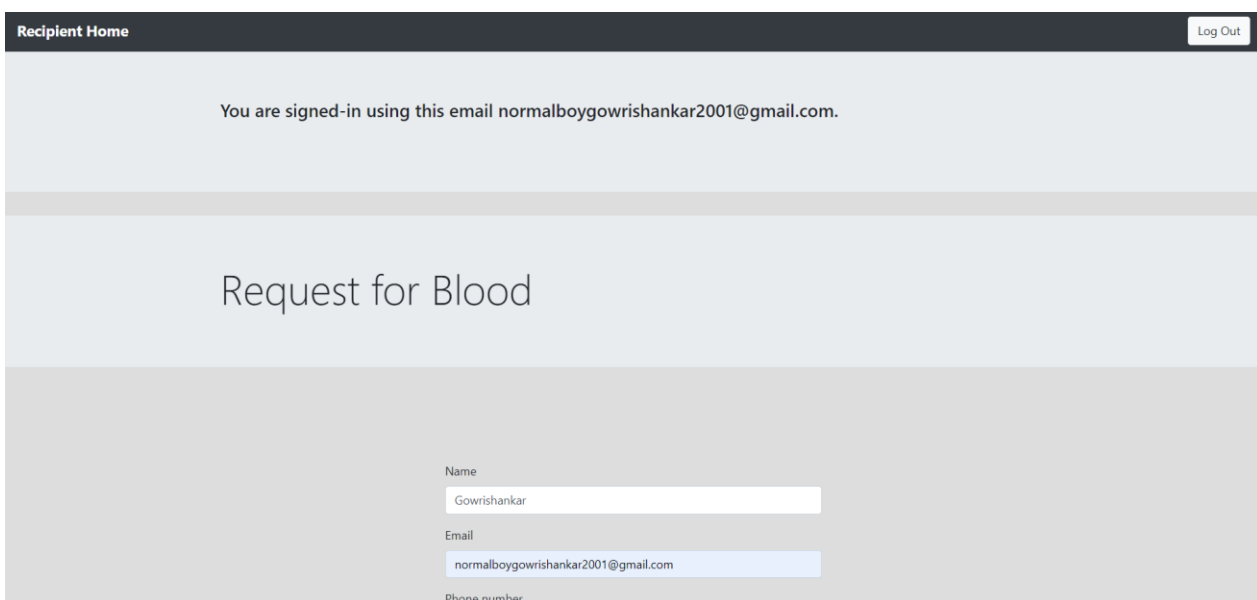
Password
....

New User ? Click to [Sign-Up](#)

Submit

Figure 9.16 - Recipient Login page

Recipient form for requesting blood



Recipient Home Log Out

You are signed-in using this email normalboygowrishankar2001@gmail.com.

Request for Blood

Name
Gowrishankar

Email
normalboygowrishankar2001@gmail.com

Phone number

Figure 9.17 - Recipient Home page

Enter the details for the requisition

A screenshot of a web form for entering requisition details. The form is set against a light gray background. It contains the following fields: 'Name' with the value 'Gowrishankar'; 'Email' with the value 'normalboygowrishankar2001@gmail.com'; 'Phone number' with the value '9685741425'; 'District' with a dropdown menu showing 'Karur'; 'Hospital name' with a dropdown menu showing 'Kavin Hospital'; and 'Blood Group' with a dropdown menu showing 'O-'. A 'Submit' button is located at the bottom of the form.

Figure 9.18 - Entering details screenshots

Getting email about confirmation of the receiving plasma tomorrow.

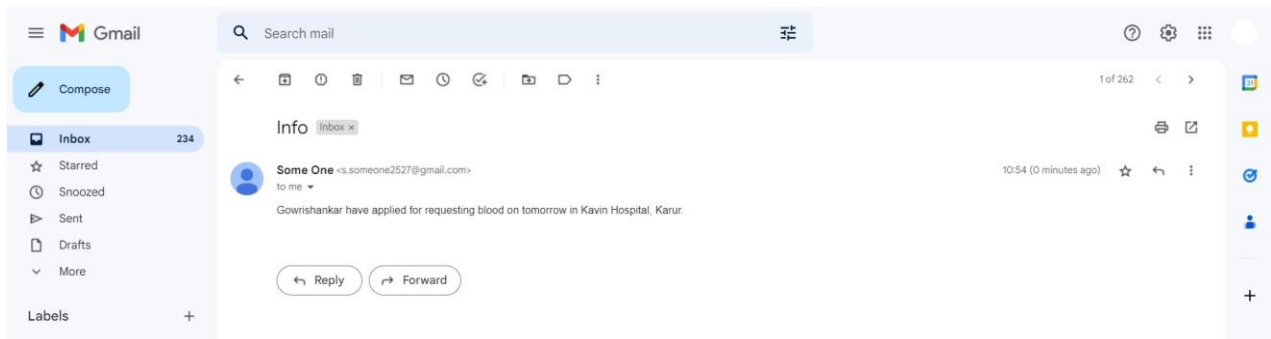


Figure 9.19 - Confirmation receiving plasma blood email screenshot

Finally, we navigated to status page of the recipient.

A screenshot of a web page titled 'Recipient Status'. At the top right is a 'Log Out' button. The main content area has a light blue background with the text 'Hello' and 'You are signed-in using this email normalboygowrishankar2001@gmail.com'. Below this, a message states: 'Gowrishankar have applied for requesting blood on tomorrow in Kavin Hospital, Karur. You need get it tomorrow.' At the bottom, there is a 'Feedback Form' with fields for 'Username' and 'Firstname'.

Figure 9.20 - Recipient Status page with feedback form

After receiving blood, we got the email for the confirmation of receiving plasma.

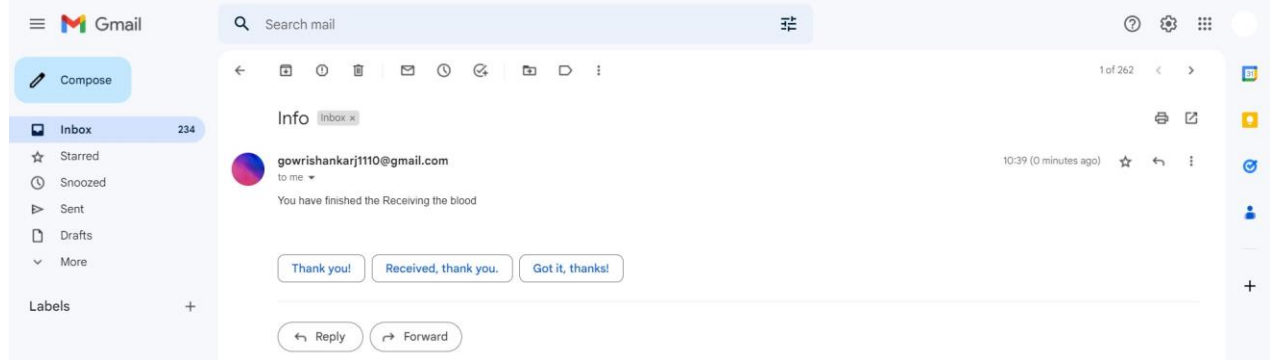


Figure 9.21 - Confirmation finished receiving plasma blood email screenshot

In-charge Pages Screenshots

First, we get enter into the In-Charge page.

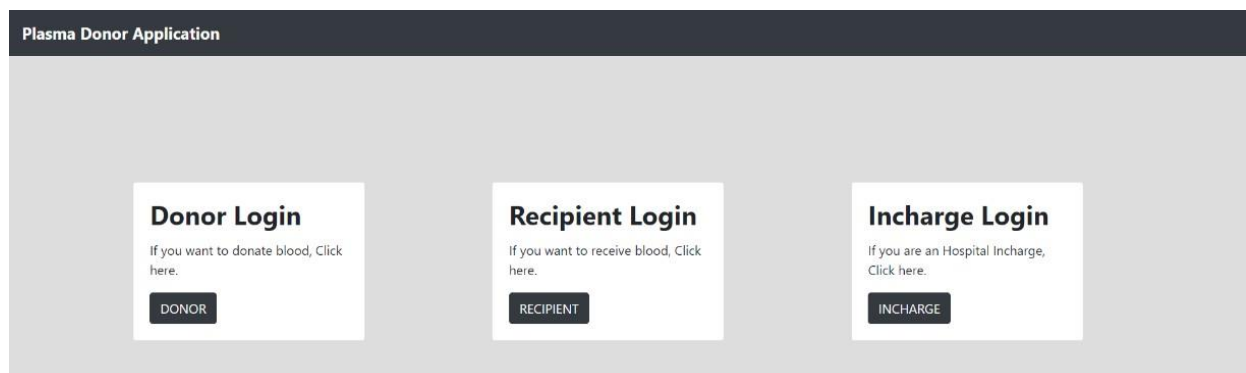


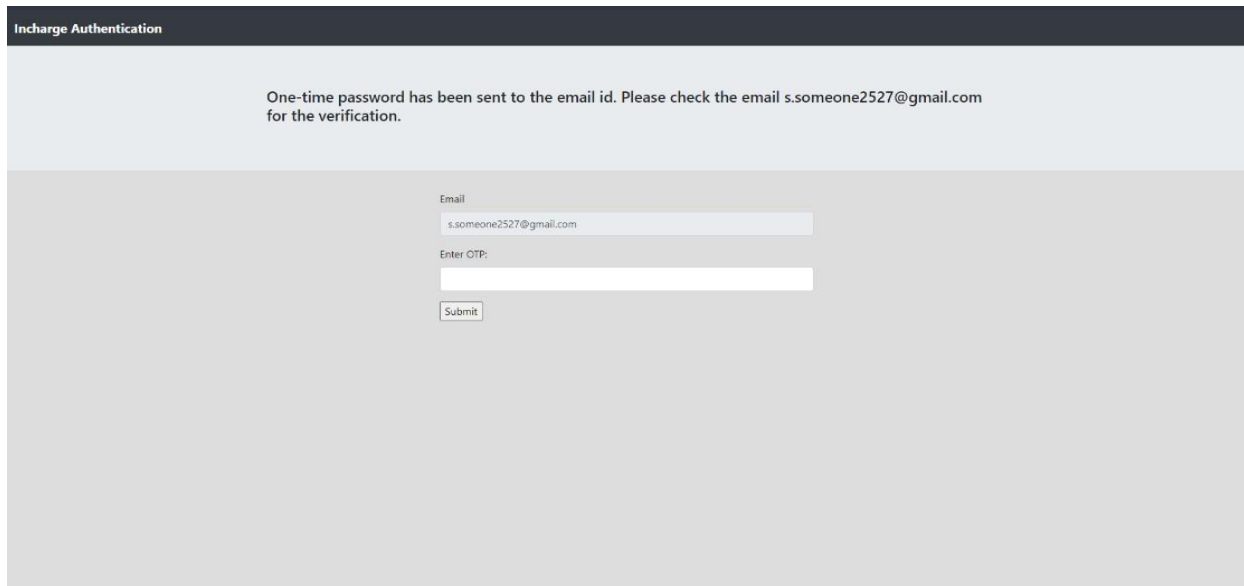
Figure 9.22 - Home page

Clicking on sign-up to register and enter the necessary details,

A screenshot of the 'Register Page' form. It contains several input fields: 'Hospital Name' (filled with 'Kavin Hospital'), 'Email' (filled with 's.someone2527@gmail.com'), 'Phone Number' (filled with '9886565515'), 'District' (filled with 'Karur'), 'PIN CODE' (filled with '639001'), and 'Address' (filled with '123 , car street'). There are also fields for 'Password' and 'Re-Type Password', both filled with '****'. A 'Matching' status is shown below the password fields. At the bottom, there is a link 'Already a User ? Click to Login' and a 'Submit' button.

Figure 9.23 - Register page

After getting the OTP from user, enter into the page



The screenshot shows a web page titled "Incharge Authentication". It has a dark header bar with the title. Below the header, a light blue box contains the text: "One-time password has been sent to the email id. Please check the email s.someone2527@gmail.com for the verification." Below this, there is a form with two input fields: "Email" (containing "s.someone2527@gmail.com") and "Enter OTP:". A "Submit" button is located below the OTP field.

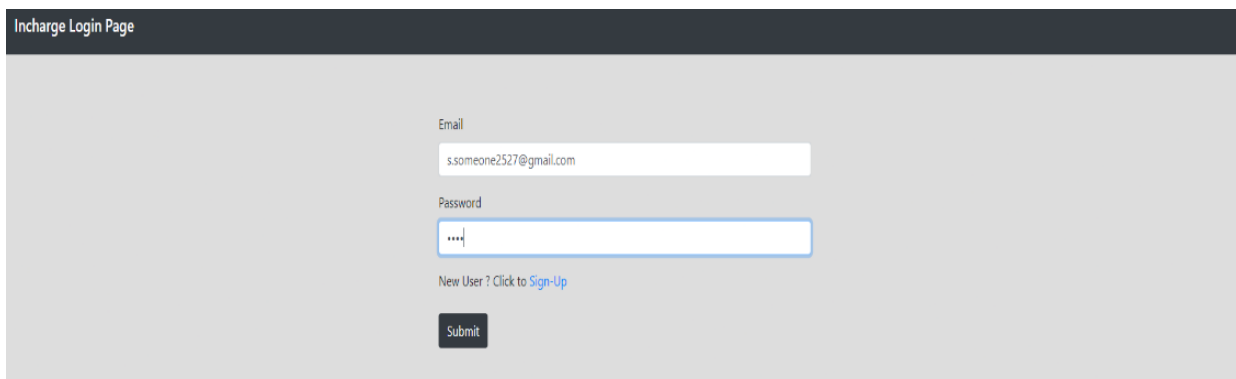
Figure 9.24 - In-charge Authentication page

Getting OTPs,



Figure 9.25 - OTP received from email screenshot

After entering OTP, We navigated into the login page.



The screenshot shows a web page titled "Incharge Login Page". It has a dark header bar with the title. Below the header, there is a form with two input fields: "Email" (containing "s.someone2527@gmail.com") and "Password" (containing "****"). Below the password field, there is a link "New User ? Click to Sign-Up". A "Submit" button is located below the form.

Figure 9.26 - In-charge Login page

After logged in In-charge home page will be displayed

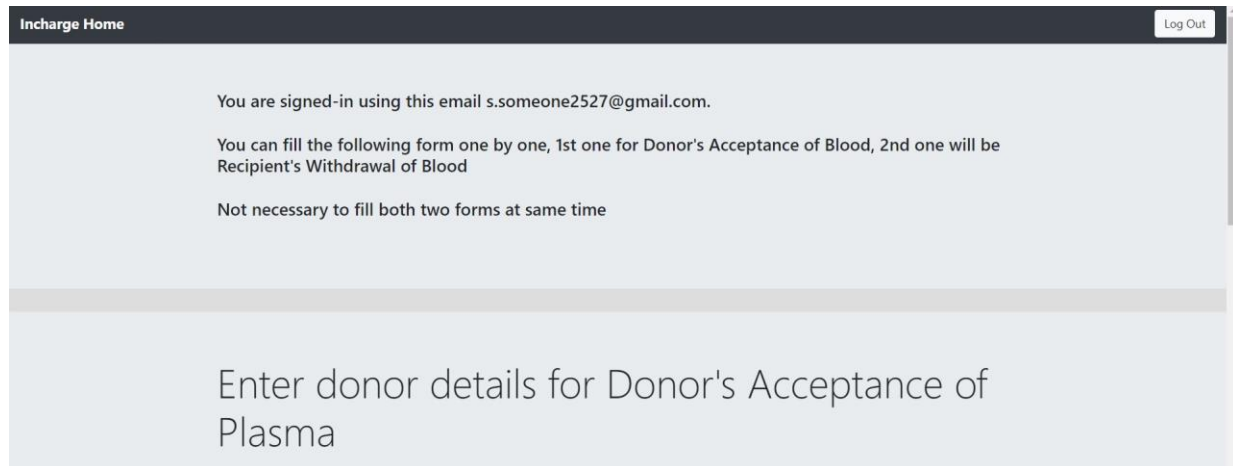


Figure 9.27 - In-charge home page

In-charge fill the form for donor's donation acceptance.

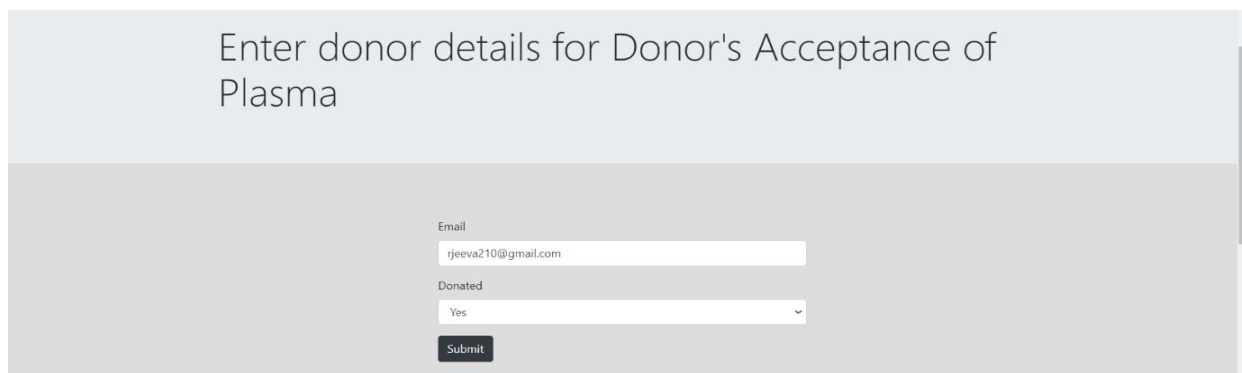


Figure 9.28 - In-charge Donor Acceptance Screenshot

Getting email confirmation from donor's acceptance from the In-charge.



Figure 9.29 - In-charge Donor Slot Booking to In-charge Screenshot

Getting email from the user, about the recipient's slot booking.



Figure 9.30 - In-charge Recipient requisition Screenshot

Confirming the recipient's plasma donation form

A screenshot of a web form titled 'Enter recipient details for Recipient's Withdrawal of Blood'. The form is set against a light blue header and a grey body. It contains several input fields: 'Name of the Recipient' with the value 'Gowrishankar', 'Email of the blood Receiver' with the value 'normalboygowrishankar2001@gmail.com', 'Email of the blood donator' with the value 'rjeeva210@gmail.com', 'Withdrawal of blood succeeded ?' with a dropdown menu showing 'Yes', and 'Hospital Name' with the value 'Kavin Hospital'. A 'Submit' button is located at the bottom of the form.

Figure 9.31 - In-charge Recipient Receiving Acceptance Screenshot

CHAPTER 10

ADVANTAGES & DISADVANTAGES

ADVANTAGES

1. The main advantage of this project is that we can book slots from anytimeanywhere using this application.
2. The details have stored on cloud database so it can be portable and availablethrough various locations.
3. The emails used in this process have been verified through OTPs.
The donor can also check their eligibility for checking whether donation can be possible or not.
4. The recipient also added feature to this application. They can also be satisfied by this application.

DISADVANTAGES

1. There will be more live interaction. Genuine progress of each and every workermatters a lot.
2. Any error happened due to the ungenuine workers; it cannot be resolved by theapplication developers.
3. The emails are the only source of providing or booking of slots in hospitals.
4. The timeout error plays a major role in this application if any slow internet beingin progress.
5. Duplication of data not allowed, each person have to use their own email addressfor their donating and receiving process.

CHAPTER 11

CONCLUSION

The Plasma Donor Application have used for donation of blood through eligibility check and receiving of blood through the hospitals Plasma availability. 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. For avoiding another situation like that donor count reduces this project have developed for the pandemic or any kind of plasma usage in the present world. To avoid such low donor count for plasma there is an emergency situation also to be happened. So, we need to keep this website active and update this application further as better as possible. Technology wisely, it needs more improvements in dynamic web pages to get transformed. Also, Faster Internet requires. Project wisely, a donor cannot book slot till 15 to 30 days of plasma donation happens, but it needs to be improved on this website. Till now it has been recorded through the paper works.

CHAPTER 12

FUTURE SCOPE

This Plasma Donor Application have developed for the resolving the donor count for Plasma and there is an more enhancement needed in future. As this project take over by the Government, it can be work genuinely throughout the entire progress in the plasma donation. For that it needs more data security, additional features and the more compatible to work. Any pandemics or in case of any urgency for the plasma, every single person can use this web application. As it helps throughout the districts only, it has to be implemented for throughout the state or country. The scope clearly defines the boundaries of the proposed system. The functional areas of this application that lies under the scope of the proposed system are the management of the availability of donors, hospitals, Plasma banks to the user or member at any time. Also, many changes that needed in this project for its betterment. These are all the future scopes needed and available.

CHAPTER 13

Source Code:

index.py:

```
from random import randint

from flask import Flask,render_template,request,url_for,redirect,session,jsonify

from flask_mail import *

import ibm_db

import os


try:

    conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=0c77d6f2-5da9-48a9-81f8-
86b520b87518.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31198;SECURITY=SSL;SSLServerCertificate=
DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=bjn03696;PWD=ef96tLJX2VjzaCPX;", "", "")

    print("Db connected")

except:

    print("Error")


dictionaryForEmailDonor={}

def printDonorData(conn):

    sql = "SELECT * FROM donorregistration"

    out = ibm_db.exec_immediate(conn, sql)

    document = ibm_db.fetch_assoc(out)

    while document != False:

        dictionaryForEmailDonor.update({document['EMAIL']:document['PASSWORD']})

        document = ibm_db.fetch_assoc(out)


dictionaryForEmailRecipient={}

def printRecipientData(conn):

    sql = "SELECT * FROM recipientregistration"

    out = ibm_db.exec_immediate(conn, sql)

    document = ibm_db.fetch_assoc(out)
```

```

while document != False:
    dictionaryForEmailRecipient.update({ document['EMAIL']:document['PASSWORD']})
    document = ibm_db.fetch_assoc(out)

dictionaryForEmailIncharge={}
def printInchargeData(conn):
    sql = "SELECT * FROM inchargeregistration"
    out = ibm_db.exec_immediate(conn, sql)
    document = ibm_db.fetch_assoc(out)
    while document != False:
        dictionaryForEmailIncharge.update({ document['EMAIL']:document['PASSWORD']})
        document = ibm_db.fetch_assoc(out)

def insertDonorData(conn,username,firstname,lastname,age,gender,blood,email,phonenumber,password,district,pincode):
    sql="INSERT INTO
donorregistration(username,firstname,lastname,age,gender,blood,email,phonenumber,password,district,pincode) VALUES
('{}','{}','{}',{},{},'{}','{}','{}',{},{},'{}','{}','{}').format(username,firstname,lastname,age,gender,blood,email,phonenumber,password
,district,pincode)
    out = ibm_db.exec_immediate(conn,sql)
    print('Number of affected rows : ',ibm_db.num_rows(out),"\\n")

def insertRecipientData(conn,username,firstname,lastname,age,gender,blood,email,phonenumber,password,district,pincode):
    sql="INSERT INTO
recipientregistration(username,firstname,lastname,age,gender,blood,email,phonenumber,password,district,pincode) VALUES
('{}','{}','{}',{},{},'{}','{}','{}',{},{},'{}','{}','{}').format(username,firstname,lastname,age,gender,blood,email,phonenumber,password
,district,pincode)
    out = ibm_db.exec_immediate(conn,sql)
    print('Number of affected rows : ',ibm_db.num_rows(out),"\\n")

def insertInchargeData(conn,hospitalname,email,phonenumber,password,district,address,pincode):
    sql="INSERT INTO inchargeregistration(hospitalname,email,phonenumber,password,district,address,pincode) VALUES
('{}','{}',{},{},'{}','{}','{}',{},{},'{}','{}','{}').format(hospitalname,email,phonenumber,password,district,address,pincode)
    out = ibm_db.exec_immediate(conn,sql)
    print('Number of affected rows : ',ibm_db.num_rows(out),"\\n")

```

```

def deleteDonorData(conn,email):
    sql = "DELETE FROM donorregistration WHERE email={ }".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    print('Number of affected rows : ', ibm_db.num_rows(out), "\n")

def deleteRecipientData(conn,email):
    sql = "DELETE FROM recipientregistration WHERE email={ }".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    print('Number of affected rows : ', ibm_db.num_rows(out), "\n")

def deleteInchargeData(conn,email):
    sql = "DELETE FROM inchargeregistration WHERE email={ }".format(email)
    out = ibm_db.exec_immediate(conn, sql)
    print('Number of affected rows : ', ibm_db.num_rows(out), "\n")

app=Flask(__name__)
app.config['SECRET_KEY']='-RFins9nLKTN-FHawdrPAQ'
app.config['MAIL_SERVER']='smtp.gmail.com'
app.config['MAIL_PORT']=465
app.config['MAIL_USERNAME']= 'gowrishankarj1110@gmail.com'
app.config['MAIL_PASSWORD']= 'lzwzepnteeyvcfibb' # to be get changed
app.config['MAIL_USE_TLS']= False
app.config['MAIL_USE_SSL']= True
mail = Mail(app)
otp = randint(000000,999999)

@app.route("/",methods=['POST','GET'])
def index():
    return render_template("indexpage.html")

@app.route("/inchargelogin",methods=['POST','GET'])
def inchargelogin():
    if request.method=="POST":
        printInchargeData(conn)
        text=request.form['text']
        password=request.form['password']

```

```

try:
    if dictionaryForEmailIncharge[text] == (password):
        session['inchargeloggedin']=True
        session['inchargeemail']=text
        return redirect(url_for('inchargehome'))
except Exception as e:
    print(e)
    return "invalid email or password"
return render_template("inchargelogin.html")

@app.route("/inchargeregister",methods=['POST','GET'])
def inchargeregister():
    if request.method=="POST":
        hospitalname = request.form['hospitalname']
        email = request.form['email']
        phonenumber=request.form['phonenumber']
        district=request.form['district']
        pincode=request.form['pincode']
        address=request.form['address']
        password = request.form['password']
        insertInchargeData(conn,hospitalname,email,phonenumber,password,district,address,pincode)
        return redirect(url_for('inchargeauthentication',email=email))
    return render_template('inchargeregister.html')

@app.route("/recipientlogin",methods=['POST','GET'])
def recipientlogin():
    if request.method=="POST":
        printRecipientData(conn)
        text=request.form['text']
        password=request.form['password']
        try:
            if dictionaryForEmailRecipient[text] == password:
                session['recipientloggedin']=True
                session['recipientemail']=text
                return redirect(url_for('recipienthome'))
        except Exception as e:

```



```

        print(e)
        return "invalid email or password"
    return render_template("recipientlogin.html")

@app.route("/recipientregister",methods=['POST','GET'])
def recipientregister():
    if request.method=="POST":
        username = request.form['username']
        firstname=request.form['firstname']
        lastname=request.form['lastname']
        gender=request.form['gender']
        age=request.form['age']
        blood=request.form['blood']
        email = request.form['email']
        phonenumber=request.form['phonenumber']
        district=request.form['district']
        pincode=request.form['pincode']
        password = request.form['password']
        insertRecipientData(conn,username,firstname,lastname,age,gender,blood,email,phonenumber,password,district,pincode)
        return redirect(url_for('recipientauthentication',email=email))
    return render_template('recipientregister.html')

@app.route("/inchargeauthentication/<email>",methods=['GET','POST'])
def inchargeauthentication(email):
    msg = Message('OTP',sender = 'gowrishankarj1110@gmail.com', recipients = [email])
    msg.body = str(otp)
    mail.send(msg)
    return render_template('inchargeauthentication.html',email=email)

@app.route('/inchargevalidate',methods=["POST"])
def inchargevalidate():
    user_otp = request.form['otp']
    email=request.form['email']
    email.lower()
    if otp == int(user_otp):
        return render_template("inchargelogin.html")

```

```
else:
    deleteInchargeData(conn,email)
    return "<h3>failure</h3>"
```

```
@app.route("/recipientauthentication/<email>",methods=['GET','POST'])
```

```
def recipientauthentication(email):
    msg = Message('OTP',sender = 'gowrishankarj1110@gmail.com', recipients = [email])
    msg.body = str(otp)
    mail.send(msg)
    return render_template('recipientauthentication.html',email=email)
```

```
@app.route('/recipientvalidate',methods=["POST"])
```

```
def recipientvalidate():
    user_otp = request.form['otp']
    email=request.form['email']
    email.lower()
    if otp == int(user_otp):
        return render_template("recipientlogin.html")
    else:
        deleteRecipientData(conn,email)
        return "<h3>failure</h3>"
```

```
@app.route("/donorauthentication/<email>",methods=['GET','POST'])
```

```
def donorauthentication(email):
    msg = Message('OTP',sender = 'gowrishankarj1110@gmail.com', recipients = [email])
    msg.body = str(otp)
    mail.send(msg)
    return render_template('donorauthentication.html',email=email)
```

```
@app.route('/donorvalidate',methods=["POST"])
```

```
def donorvalidate():
    user_otp = request.form['otp']
    email=request.form['email']
    email.lower()
    if otp == int(user_otp):
        return render_template("donorlogin.html")
```

```

else:
    deleteDonorData(conn,email)
    return "<h3>failure</h3>"

@app.route("/donorlogin",methods=['POST','GET'])
def donorlogin():
    if request.method=="POST":
        printDonorData(conn)
        text=request.form['text']
        password=str(request.form['password'])
        try:
            if dictionaryForEmailDonor[text] == (password):
                session['donorloggedin']=True
                session['donoremail']=text
                return redirect(url_for('donorhome'))
        except Exception as e:
            print(e)
            return "invalid email or password"
    return render_template("donorlogin.html")

@app.route("/donorregister",methods=['POST','GET'])
def donorregister():
    if request.method=="POST":
        username = request.form['username']
        firstname=request.form['firstname']
        lastname=request.form['lastname']
        gender=request.form['gender']
        age=request.form['age']
        blood=request.form['blood']
        email = request.form['email']
        phonenumber=request.form['phonenumber']
        district=request.form['district']
        pincode=request.form['pincode']
        password = request.form['password']
        insertDonorData(conn,username,firstname,lastname,age,gender,blood,email,phonenumber,password,district,pincode)
        return redirect(url_for('donorauthentication',email=email))

```

```

return render_template('donorregister.html')

@app.route("/donorhome",methods=['POST','GET'])
def donorhome():
    try:
        if session['donorloggedin']==True:
            status=" # data we need, to be stored in this str
            query="SELECT donated FROM donationdetails WHERE email = '{ }'".format(session['donoremail'])
            stmt = ibm_db.exec_immediate(conn,query)
            while ibm_db.fetch_row(stmt)!=False: #to store the db data
                status=(ibm_db.result(stmt,0))
            print("status = ",status)

        if status=='yes':
            send = "You have already donated, After 28 days from your donation date only, you can donate"
            firstname=" # data we need, to be stored in this str
            query="SELECT firstname FROM donationdetails WHERE email = '{ }'".format(session['donoremail'])
            stmt = ibm_db.exec_immediate(conn,query) # do the task
            while ibm_db.fetch_row(stmt)!=False: #to store the db data
                firstname=(ibm_db.result(stmt,0))

            date=" # data we need, to be stored in this str
            query="SELECT donationdate FROM donationdetails WHERE email = '{ }'".format(session['donoremail'])
            stmt = ibm_db.exec_immediate(conn,query) # do the task
            while ibm_db.fetch_row(stmt)!=False: #to store the db data
                date=(ibm_db.result(stmt,0))
            print("date = ",date)
            import datetime
            if (date.today() - date)>=datetime.timedelta(28):
                print("Yes")
                query="delete from donationdetails WHERE email = '{ }'".format(session['donoremail'])
                stmt = ibm_db.exec_immediate(conn,query)
                return render_template('donorhome.html',email=session['donoremail'],firstname=firstname)
            else:
                print(date.today() - date)
                print('No')

```

```

return render_template('donorstatus.html',firstname=firstname,email=session['donoremail'],msg=send)

districts=[]
query3="select distinct district from inchargeregistration"
stmt3=ibm_db.exec_immediate(conn,query3)
while ibm_db.fetch_row(stmt3)!=False:
    districts.append(ibm_db.result(stmt3,0))
    print(districts)
print(type(districts))

hospitals=[]
query2='select hospitalname from inchargeregistration'
stmt2=ibm_db.exec_immediate(conn,query2)
while ibm_db.fetch_row(stmt2)!=False:
    hospitals.append(ibm_db.result(stmt2,0))
    print(hospitals)

firstname=" # data we need, to be stored in this str
query="SELECT firstname FROM DONORREGISTRATION WHERE email = '{ }'".format(session['donoremail'])
stmt = ibm_db.exec_immediate(conn,query) # do the task
while ibm_db.fetch_row(stmt)!=False: #to store the db data
    firstname=(ibm_db.result(stmt,0))

if request.method=='POST':
    print(request.form['age'])
    if request.form['age']=='yes':
        if request.form['weight']=='yes':
            if request.form['height']=='yes':
                if request.form['drug']=='no':
                    if request.form['pregnant']=='no':
                        if request.form['iron']=='no':
                            if request.form['endoscopy']=='no':
                                if request.form['cough']=='no':
                                    if request.form['vomit']=='no':
                                        if request.form['covid']=='no':
                                            if request.form['antibodies']=='no':

```

```
msg="Eligible to Donate"
```

```
return
```

```
username=request.form['username']
```

```
firstname=request.form['firstname']
```

```
email=request.form['email']
```

```
phonenumber=request.form['phonenumber']
```

```
district=request.form['district']
```

```
hospitalname=request.form['hospitalname']
```

```
donationdate=request.form['donationdate']
```

```
blood=request.form['blood']
```

```
sql="INSERT INTO donationdetails(username,firstname,email,phonenumber,district,hospitalname,donationdate,blood  
) VALUES ('{}','{}','{}','{}','{}','{}','{}','{}').format(username,firstname,email,phonenumber,district,hospitalname,donationdate  
,blood)
```

```
stmt = ibm_db.exec_immediate(conn,sql) # do the task
```

```
donated=True
```

```
send="{} have applied for donation on {} in {}, {}".format(firstname,donationdate,hospitalname,district)
```

```
hospitalemail="
```

```
query="SELECT email FROM inchargeregistration WHERE hospitalname = '{}'.format(hospitalname)
```

```
stmt = ibm_db.exec_immediate(conn,query) # do the task
```

```
while ibm_db.fetch_row(stmt)!=False: #to store the db data
```

```
    hospitalemail=(ibm_db.result(stmt,0))
```

```
msg = Message('Info',sender = 'gowrishankarj1110@gmail.com', recipients = [hospitalemail])
```

```
msg.body = send
```

```
mail.send(msg)
```

```
msg = Message('Info',sender = 'gowrishankarj1110@gmail.com', recipients = [email])
```

```
msg.body = send
```

```
mail.send(msg)
```

```
return render_template('donorstatus.html',donated=donated,email=session['donoremail'],firstname=firstname,msg=send)
```

```
return render_template('donorhome.html',donated=donated)
```

```
@app.route("/donorlogout",methods=['POST','GET'])
```

```
def donorlogout():
```

```
session.pop('donorloggedin',None)
```

```
session.pop('donoremail',None)
```

```
return redirect(url_for('index'))
```

```

@app.route("/recipienthome",methods=['POST','GET'])
def recipienthome():
try:
    if session['recipientloggedin']==True:
        if request.method=='POST':
            recipientname=request.form['recipientname']
            email=request.form['email']
            phonenumber=request.form['phonenumber']
            district=request.form['district']
            hospital=request.form['hospitalname']
            requestedblood=request.form['blood']

            sql="INSERT INTO recipientrequested(recipientname,email,phonenumber,district,hospital,requestedblood) VALUES
('
            {'','{}','{}','{}','{}','{}')".format(recipientname,email,phonenumber,district,hospital,requestedblood)
            stmt = ibm_db.exec_immediate(conn,sql)
            hospitalemail=""
            query="SELECT email FROM inchargeregistration WHERE hospitalname = '{}'.format(hospital)
            stmt = ibm_db.exec_immediate(conn,query) # do the task
            while ibm_db.fetch_row(stmt)!=False: #to store the db data
                hospitalemail=(ibm_db.result(stmt,0))
            send="{} have applied for requesting blood on tomorrow in {}, {}".format(recipientname,hospital,district)
            msg = Message('Info',sender = 'gowrishankarj1110@gmail.com', recipients = [hospitalemail])
            msg.body = send
            mail.send(msg)
            msg1='You need get it tomorrow.'
            return render_template('recipientstatus.html',email=session['recipientemail'],send=send,msg1=msg1)
        districts=[]
        query3="select distinct district from hospitalhaving"
        stmt3=ibm_db.exec_immediate(conn,query3)
        while ibm_db.fetch_row(stmt3)!=False:
            districts.append(ibm_db.result(stmt3,0))
            print(districts)
        print(type(districts))
        return render_template('recipienthome.html',email=session['recipientemail'],districts=districts)
    except KeyError:
        return "You have been logged out."

```

```
@app.route("/recipientlogout",methods=['POST','GET'])
```

```
def recipientlogout():
```

```
    session.pop('recipientloggedin',None)
```

```
    session.pop('recipientemail',None)
```

```
    return redirect(url_for('index'))
```

```
@app.route("/inchargehome",methods=['POST','GET'])
```

```
def inchargehome():
```

```
    try:
```

```
        if session['inchargeloggedin']==True:
```

```
            return render_template('inchargehome.html',email=session['inchargeemail'])
```

```
    except KeyError:
```

```
        return "You have been logged out."
```

```
@app.route("/inchargelogout",methods=['POST','GET'])
```

```
def inchargelogout():
```

```
    session.pop('inchargeloggedin',None)
```

```
    session.pop('inchargeemail',None)
```

```
    return redirect(url_for('index'))
```

```
@app.route("/inchargedonor",methods=['POST','GET'])
```

```
def inchargedonor():
```

```
    try:
```

```
        if session['inchargeloggedin']==True:
```

```
            if request.method=='POST':
```

```
                email=request.form['email']
```

```
                donated=request.form['donated']
```

```
                query="update donationdetails set donated = '{ }' where email = '{ }';".format(donated,email)
```

```
                stmt = ibm_db.exec_immediate(conn,query)
```

```
                firstname="#" # data we need, to be stored in this str
```

```
                query="SELECT firstname FROM donationdetails WHERE email = '{ }'".format(email)
```

```
                stmt = ibm_db.exec_immediate(conn,query)
```

```
                while ibm_db.fetch_row(stmt)!=False: #to store the db data
```

```
                    firstname=(ibm_db.result(stmt,0))
```

```
                send="{ } have finished the donation".format(firstname)
```



```

        msg = Message('Info',sender = 'gowrishankarj1110@gmail.com', recipients = [email])
        msg.body = send
        mail.send(msg)
        return render_template('inchargehome.html',email=session['inchargeemail'])
    return render_template('inchargehome.html',email=session['inchargeemail'])
except KeyError:
    return "You have been logged out."

@app.route("/inchargerecipient",methods=['POST','GET'])
def inchargerecipient():
    try:
        if session['inchargeloggedin']==True:
            if request.method=='POST':
                donoremail=request.form['email']
                hospitalname=request.form['hospitalname']
                query="delete from hospitalhaving where hospital = '{ }' and email='{ }';".format(hospitalname,donoremail)
                stmt = ibm_db.exec_immediate(conn,query)
                receiveremail=request.form['emailreceiver']
                send="You have finished the Receiving the blood"
                msg = Message('Info',sender = 'gowrishankarj1110@gmail.com', recipients = [receiveremail])
                msg.body = send
                mail.send(msg)
                query="delete from recipientrequested where email='{ }';".format(receiveremail)
                stmt = ibm_db.exec_immediate(conn,query)
                return render_template('inchargehome.html',email=session['inchargeemail'])
            return render_template('inchargehome.html',email=session['inchargeemail'])
    except KeyError:
        return "You have been logged out."

@app.after_request #after session pop when we click back python flask
def after_request(response):
    response.headers.add('Cache-Control', 'no-store, no-cache, must-revalidate, post-check=0, pre-check=0')
    return response

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

```

```

if request.method=="POST":
    username=request.form['username']
    firstname=request.form['firstname']
    email=request.form['email']
    phonenumber=request.form['phonenumber']
    district=request.form['district']
    hospitalname=request.form['hospitalname']
    donationdate=request.form['donationdate']
    blood=request.form['blood']
    treat=request.form['treat']
    me=request.form['me']

    sql="INSERT INTO feedback(username,firstname,email,phonenumber,district,hospitalname,donationdate,blood,me
,treatment) VALUES
('{}','{}','{}',{},{},{},{},{},{},{},{},{},{})".format(username,firstname,email,phonenumber,district,hospitalname,donationdate
,blood,me,treat)

    stmt = ibm_db.exec_immediate(conn,sql) # do the task

    if me == "donor":
        return "<h3><center>Your feedback has submitted to the database.<a href={ { url_for('donorlogout') } }><button>Log
Out</button></a></center></h3>"

    elif me == "recipient":
        return "<h3><center>Your feedback has submitted to the database.<a href={ { url_for('recipientlogout') } }><button>Log
Out</button></a></center></h3>"

if __name__ == "__main__":
    #app.run(debug=True)
    port=int(os.environ.get('PORT',5000))
    app.run(port=port,host='0.0.0.0')

```

HTML PAGES:

donationform.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 Home</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"
    integrity="sha384-gg R0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
    crossorigin="anonymous">
<style>
    body{
        background-color: #DDDDDD;
    }
    .colormine{
        color: white;
    }
    .bold{
        font-weight: bold;
    }
    .center {
        position: absolute;
        left: 0;
        right: 0;
        margin: auto;
    }
    form{
        width: 30%;
    }
</style>
</head>
<body>

<nav class="navbar navbar-expand-lg navbar-light bg-dark">
    <a class="navbar-brand" href="#"><span class="colormine bold">Donation Form</span></a>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="#"> <span class="sr-only">(current)</span></a>
            </li>
        </ul>
    </div>

```

```

    <a class="form-inline my-2 my-lg-0" href="{{ url_for('donorlogout') }}"><button class="bg-light btn">Log Out</button>
</a>
</div>
</nav>

<div class="jumbotron jumbotron-fluid">
    <div class="container">
        <h1 class="display-4">Hello {{ firstname }}</h1>
        <h3>{{ msg }}</h3>
        <h4 class="display-6">You are signed-in using this email {{ email }} </h4>
    </div>
</div>

<form action="/donationdetails" class="center" method="post">
    <div class="form-group">
        <label>Username</label>
        <input type="text" class="form-control" name="username">
    </div>
    <div class="form-group">
        <label for="firstname">Firstname</label>
        <input type="text" class="form-control" name="firstname">
    </div>
    <div class="form-group">
        <label for="email">Email</label>
        <input type="text" name="email" class="form-control">
    </div>
    <div class="form-group">
        <label for="phonenumber">Phone number</label>
        <input type="text" name="phonenumber" class="form-control">
    </div>
    <div class="form-group">
        <label for="district">District</label>
        <select class="form-control" id="district" name="district">
            {% for district in districts %}
            <option value="{{ district }}">{{ district }}</option>
            {% endfor %}
        </select>
    </div>

```

```

</div>
<div class="form-group">
  <label for="hospitalname">Hospital name</label>
  <select name="hospitalname" class="form-control" id="hospitalname"></select>
</div>
<div class="form-group">
  <label for="donationdate">Donation date</label>
  <input type="date" class="form-control" name="donationdate">
</div>
<div class="form-group">
  <label for="blood">Blood Group</label>
  <input type="text" class="form-control" name="blood">
</div>
<br>
  <br>
  <button class="btn bg-dark colormine" type="submit">Submit</button>
  <br><br><br><br><br><br><br><br><br><br><br>
</form>
<br>
<script type="text/javascript">
  district_select = document.getElementById('district')
  hospital_select = document.getElementById('hospitalname')
  district_select.onchange=function(){
    district = district_select.value;
    //alert(district);
    fetch('/hospitals/'+district).then(function(response){
    response.json().then(function(data){
      optionHTML="";
      for ( hospital of data.districthospital){
        optionHTML+="

```

```

});
}
</script>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

donorauthentication.html

```

<!DOCTYPE html>
<html>
<head>
<title>OTP Verification</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
<style>
body{
background-color: #DDDDDD;
}
.colormine{
color: white;
}
.bold{
font-weight: bold;
}
.center {
position: absolute;
left: 0;
right: 0;
margin: auto;
}
form{
width: 30%;
}
</style>

</head>

```

```

<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-dark">
    <a class="navbar-brand" href="#"><span class="colormine bold">Donor Authentication</span></a>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="#"> <span class="sr-only">(current)</span></a>
        </li>
      </ul>
    </div>
  </nav>

  <div class="jumbotron jumbotron-fluid">
    <div class="container">
      <h4 class="display-6"> One-time password has been sent to the email id. Please check the email {{email}} for the
verification.</h4>
    </div>
  </div>

  <form action = "/donorvalidate" method="post" class="center">
    <div class="form-group">
      <label>Email</label>
      <input type="text" value="{{email}}" class="form-control" name="email" readonly>
    </div>
    <div class="form-group">
      <label>Enter OTP:</label>
      <input type="text" class="form-control" name="otp">
    </div>
    <input type="submit" value="Submit">
  </form>

  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abfTE1Pi6jizo" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdsJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

donorhome.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>Eligibility Check</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <style>
    body{
      background-color: #DDDDDD;
    }
    .colormine{
      color: white;
    }
    .bold{
      font-weight: bold;
    }
    .center {
      position: absolute;
      left: 0;
      right: 0;
      margin: auto;
    }
    form{
      width: 30%;
    }
  </style>
</head>
<body>

<nav class="navbar navbar-expand-lg navbar-light bg-dark">
  <a class="navbar-brand" href="#"><span class="colormine bold">Donor Home for Eligibility Check</span></a>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#"> <span class="sr-only">(current)</span></a>
      </li>
    </ul>
    <a class="form-inline my-2 my-lg-0" href="{ { url_for('donorlogout') } }"><button class="bg-light btn">Log Out</button></a>
  </div>
```



```

</nav>

<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h1 class="display-4">Hello { {firstname} }</h1>
    <h4 class="display-6">You are signed-in using this email { {email} }.</h4>
  </div>
</div>

<form action="/donorhome" method="post" class="center" >
  <div class="form-group">
    <br>Are you between the age 16 and 70?<br>
    <select id="age" class="form-control" name="age">
      <option value="">---</option>
      <option value="yes">Yes</option>
      <option value="no">No</option>
    </select>
  </div>
  <div class="form-group">
    <br>Are you being weight more than 50 kgs?<br>
    <select id="weight" class="form-control" name="weight">
      <option value="">---</option>
      <option value="yes">Yes</option>
      <option value="no">No</option>
    </select>
  </div>
  <div class="form-group">
    <br>Are you being height more than 150 cms?<br>
    <select id="height" class="form-control" name="height">
      <option value="">---</option>
      <option value="yes">Yes</option>
      <option value="no">No</option>
    </select>
  </div>
  <div class="form-group">
    <br>Have you ever injected drugs not prescribed by a doctor?<br>
    <select id="drug" class="form-control" name="drug">
      <option value="">---</option>
      <option value="yes">Yes</option>
      <option value="no">No</option>
    </select>
  </div>
  <div class="form-group">

```

```

<br>Have you been pregnant or given birth in the last 9 months?<br>
  <select id="pregnant" class="form-control" name="pregnant">
    <option value="">---</option>
    <option value="yes">Yes</option>
    <option value="no">No</option>
  </select>
</div>
<div class="form-group">
  <br>Have you taken iron medication prescribed by a doctor?<br>
  <select id="iron" class="form-control" name="iron">
    <option value="">---</option>
    <option value="yes">Yes</option>
    <option value="no">No</option>
  </select>
</div>
<div class="form-group">
  <br>Have you had an endoscopy in the last 4 months?<br>
  <select id="endoscopy" class="form-control" name="endoscopy">
    <option value="">---</option>
    <option value="yes">Yes</option>
    <option value="no">No</option>
  </select>
</div>
<div class="form-group">
  <br>Have you had a cough, cold, sore throat or influenza in last 28 days?<br>
  <select id="cough" class="form-control" name="cough">
    <option value="">---</option>
    <option value="yes">Yes</option>
    <option value="no">No</option>
  </select>
</div>
<div class="form-group">
  <br>Have you, or anyone in your household, had diarrhoea or vomiting symptoms in the last 28 days?<br>
  <select id="vomit" class="form-control" name="vomit">
    <option value="">---</option>
    <option value="yes">Yes</option>
    <option value="no">No</option>
  </select>
</div>
<div class="form-group">
  <br>Have you tested positive for COVID-19 in the last seven days?<br>
  <select id="covid" class="form-control" name="covid">
    <option value="">---</option>

```

```

        <option value="yes">Yes</option>
        <option value="no">No</option>
    </select>
</div>
<div class="form-group">
    <br>Have you taken antibiotics in the last seven days?<br>
    <select id="antibodies" class="form-control" name="antibodies">
        <option value="">---</option>
        <option value="yes">Yes</option>
        <option value="no">No</option>
    </select>
</div>
<br>
<br>
<button type="submit" class="bg-dark btn colormine">Check</button>
<br>
<br><br>
<br><br>
<br><br>
<br><br>
<br><br>
<br><br>
<br>
</form>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

donorlogin.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
    <title>Login Page</title>
    <style type="text/css">
        body{

```

```

    background-color: #DDDDDD;
}

.msg,.login{
    text-align: center;
    margin-left: auto;
    margin-right: auto;
}

form{
    width: 30%;
}

.colormine{
    color: white;
}

.center {
    position: absolute;
    left: 0;
    right: 0;
    margin: auto;
}

</style>
</head>
<body>
<nav class="h1 navbar navbar-light p-3 mb-2 bg-dark text-white justify-content-between">
  <a class="navbar-brand">Donor Login Page</a>
</nav>
<br>
<h5 class="msg">{{ msg }}</h5>
<br>
<form action="/donorlogin" method="POST" class="center">
  <div class="form-group">
    <label>Email</label>
    <input type="text" name="text" class="form-control" placeholder="Enter Email">
  </div>
  <div class="form-group">
    <label>Password</label>
    <input type="password" name="password" class="form-control" placeholder="Password">
  </div>
  <div>
    New User ? Click to <a href="{{ url_for('donorregister') }}">Sign-Up</a>
  </div>
<br>

```

```

    <button type="submit" class="btn bg-dark colormine">Submit</button>
  </form>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KChRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous">
</script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
</body>
</html>

```

donorregister.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Donor Register page</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <style type="text/css">
body{
  background-color: #DDDDDD;
}
.colormine{
  color: white;
}
.bold{
  font-weight: bold;
}
.center {
  position: absolute;
  left: 0;
  right: 0;
  margin: auto;
}
form{
  width: 30%;
}
</style>
</head>
<body>
  <nav class="h1 navbar navbar-light p-3 mb-2 bg-dark text-white justify-content-between">

```

```

<a class="navbar-brand">Donor Register Page</a>
</nav>
<form action="/donorregister" method="POST" class="center">
  <h5 class="msg" style="color:red;">{{ msg1 }}</h5>
  <h5 class="msg" style="color:red;">{{ msg2 }}</h5>
  <div class="form-group">
    <label for="Username">User Name</label>
    <input type="text" name="username" class="form-control" id="Username" onkeypress="clsAlphaNoOnly(event)"
onpaste="return false;" placeholder="Enter Username" required>
  </div>
  <div class="form-group">
    <label for="Firstname">First Name</label>
    <input type="text" name="firstname" class="form-control" id="Firstname" onkeypress="clsAlphaOnly(event)"
onpaste="return false;" placeholder="Enter Firstname" required>
  </div>
  <div class="form-group">
    <label for="Lastname">Last Name</label>
    <input type="text" name="lastname" class="form-control" id="Lastname" onkeypress="clsAlphaOnly(event)"
onpaste="return false;" placeholder="Enter Lastname" required>
  </div>
  <div class="form-group">
    <label for="gender" class="control-label">Gender</label>
    <div>
      <select class="form-control" id="Gender" name="gender">
        <option value="Others">Others</option>
        <option value="Male">Male</option>
        <option value="Female">Female</option>
      </select>
    </div>
  </div>
  <div class="form-group">
    <label for="Age">Age</label>
    <input type="number" name="age" id="Age" class="form-control" placeholder="Enter Age" min="18"
max="65" step="1" required>
  </div>
  <div class="form-group">
    <label for="Blood">Blood Group</label>
    <input type="text" name="blood" id="Blood" class="form-control" placeholder="Enter Blood Group as
A+ or O+ etc." required>
  </div>
  <div class="form-group">
    <label for="Email">Email</label>
    <input type="text" name="email" class="form-control" id="Email" placeholder="Enter Email" required>
  </div>

```

```

</div>
<div class="form-group">
    <label for="PhoneNumber">Phone Number</label>
    <input type="tel" id="phone" name="phonenummer" class="form-control" placeholder="1234567890" pattern="[0-9]{3}[0-9]{3}[0-9]{4}" required>
</div>
<div class="form-group">
    <label for="District">District</label>
    <input type="text" name="district" class="form-control" id="District" placeholder="Enter District" required>
</div>
<div class="form-group">
    <label for="PINCODE">PIN CODE</label>
    <input type="text" name="pincode" class="form-control" id="PINCODE" placeholder="Enter PINCODE" required>
</div>
<div class="form-group">
    <label for="Password">Password</label>
    <input type="password" name="password" class="form-control" id="Password" placeholder="Password"
onkeyup='check();' required>
</div>
<div class="form-group">
    <label for="RetypePassword">Re-Type Password</label>
    <input type="password" name="retypepassword" class="form-control" id="RetypePassword" placeholder="Re-Type
Your Password" onkeyup='check();' required>
    <span id='message'></span>
</div>
<div>
    Already a User ? Click to <a href="{{ url_for('donorlogin')}}">Login</a>
</div>
<br>
<button type="submit" class="btn btn-dark colormine">Submit</button>
<br>
<br>
<br>
<br>
</form>
<script>
var check = function() {
    if (document.getElementById('Password').value == document.getElementById('RetypePassword').value) {
        document.getElementById('message').style.color = 'green';
        document.getElementById('message').innerHTML = 'Matching';
    }
    else {
        document.getElementById('message').style.color = 'red';
    }
}

```

```

        document.getElementById('message').innerHTML = 'Not Matching';
    }
}

function clsAlphaNoOnly (e) { // Accept only alpha numerics, no special characters
    var regex = new RegExp("[a-zA-Z0-9 ]+$");
    var str = String.fromCharCode(!e.charCode ? e.which : e.charCode);
    if (regex.test(str)) {
        return true;
    }

    e.preventDefault();
    return false;
}

function clsAlphaOnly (e) { // Accept only alpha numerics, no special characters
    var regex = new RegExp("[a-zA-Z]+$");
    var str = String.fromCharCode(!e.charCode ? e.which : e.charCode);
    if (regex.test(str)) {
        return true;
    }

    e.preventDefault();
    return false;
}
</script>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abfTE1Pi6jizo"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdSjQ6hJty5KVphtPhzWj9WO1cHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

donorstatus.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 Status</title>
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"
integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
crossorigin="anonymous">
<style>
  body{
    background-color: #DDDDDD;
  }
  .colormine{
    color: white;
  }
  .bold{
    font-weight: bold;
  }
  .center {
    position: absolute;
    left: 0;
    right: 0;
    margin: auto;
  }
  form{
    width: 30%;
  }
</style>
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-dark">
    <a class="navbar-brand" href="#"><span class="colormine bold">Donor Status</span></a>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="#"> <span class="sr-only">(current)</span></a>
        </li>
      </ul>
      <a class="form-inline my-2 my-lg-0" href="{ { url_for('donorlogout') } }"><button class="bg-light btn">Log
Out</button></a>
    </div>
  </nav>
  <div class="jumbotron jumbotron-fluid">
    <div class="container">
      <h1 class="display-4">Hello { {firstname} }</h1>
      <h4 class="display-6">You are signed-in using this email { {email} } </h4>
      <br>

```

```

    </div>
</div>
<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h4 class="display-6">{ { msg } }</h4>
  </div>
</div>
<br>
<br>
<h3> <center>Feedback Form</center> </h3>
<form action="/feedback" class="center" method="post">
  <div class="form-group">
    <label>Username</label>
    <input type="text" class="form-control" name="username">
  </div>
  <div class="form-group">
    <label for="firstname">Firstname</label>
    <input type="text" class="form-control" name="firstname">
  </div>
  <div class="form-group">
    <label for="email">Email</label>
    <input type="text" name="email" class="form-control">
  </div>
  <div class="form-group">
    <label for="phonenumber">Phone number</label>
    <input type="text" name="phonenumber" class="form-control">
  </div>
  <div class="form-group">
    <label for="district">District</label>
    <input type="text" name="district" class="form-control">
  </div>
  <div class="form-group">
    <label for="hospitalname">Hospital name</label>
    <input type="text" name="hospitalname" class="form-control">
  </div>
  <div class="form-group">
    <label for="donationdate">Donation date</label>
    <input type="date" class="form-control" name="donationdate">
  </div>
  <div class="form-group">
    <label for="blood">Blood Group</label>
    <input type="text" class="form-control" name="blood">
  </div>

```

```

<div class="form-group">
  <label for="me">I'm a Donor/Recipient</label>
  <select id="me" class="form-control" name="me">
    <option value="">---</option>
    <option value="donor">Donor</option>
    <option value="recipient">Recipient</option>
  </select>
</div>
<div class="form-group">
  <label for="treat">How they treated you ?</label>
  <textarea type="text" class="form-control" name="treat"></textarea>
</div>
<button type="submit" class="btn btn-dark">Submit</button>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
</form>

```

```

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

inchargeauthentication.html

```

<!DOCTYPE html>
<html>
<head>
  <title>OTP Verification</title>

```

```

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css"
    integrity="sha384-
    ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
    crossorigin="anonymous">
<style>
  body{
    background-color: #DDDDDD;
  }
  .colormine{
    color: white;
  }
  .bold{
    font-weight: bold;
  }
  .center {
    position: absolute;
    left: 0;
    right: 0;
    margin: auto;
  }
  form{
    width: 30%;
  }
</style>

</head>

<body>
<nav class="navbar navbar-expand-lg navbar-light bg-dark">
  <a class="navbar-brand" href="#"><span class="colormine bold">Incharge Authentication</span></a>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#"> <span class="sr-only">(current)</span></a>
      </li>
    </ul>
  </div>
</nav>

<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h4 class="display-6"> One-time password has been sent to the email id. Please check the email {{email}} for the
    verification.</h4>
  </div>

```

```

</div>

<form action = "/inchargevalidate" method="post" class="center">
  <div class="form-group">
    <label>Email</label>
    <input type="text" value="{ {email} }" class="form-control" name="email" readonly>
  </div>
  <div class="form-group">
    <label>Enter OTP:</label>
    <input type="text" class="form-control" name="otp">
  </div>
  <input type="submit" value="Submit">
</form>

  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

inchargehome.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>Incharge Home</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <style>
body{
  background-color: #DDDDDD;
}
.colormine{
  color: white;
}
.bold{
  font-weight: bold;
}

```

```

.center {
    position: absolute;
    left: 0;
    right: 0;
    margin: auto;
}
form{
    width: 30%;
}
</style>
</head>
<body>

<nav class="navbar navbar-expand-lg navbar-light bg-dark">
    <a class="navbar-brand" href="#"><span class="colormine bold">Incharge Home</span></a>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" href="#"> <span class="sr-only">(current)</span></a>
            </li>
        </ul>
    </div>
<div class="jumbotron jumbotron-fluid">
    <div class="container">
        <h1 class="display-4">Enter donor details for Donor's Acceptance of Blood</h1>
    </div>
</div>
<br><br>
<form action="/inchargedonor" method="POST" class="center" >
    <div class="form-group">
        <label for="email">Email</label>
        <input type="email" class="form-control" name="email">
    </div>
    <div class="form-group">
        <label for="donated">Donated</label>
        <select name="donated" class="form-control" id="donated">
            <option value="yes">Yes</option>
            <option value="no">No</option>
        </select>
    </div>
    <button type="submit" class="bg-dark btn colormine">Submit</button>
</form>
<br><br>
<br><br><br><br><br><br><br><br><br>

```

```

<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h1 class="display-4">Enter recipient details for Recipient's Withdrawal of Blood</h1>
  </div>
</div>
<br><br>
<form action="/inchargerecipient" method="POST" class="center">
  <div class="form-group">
    <label for="name">Name of the Recipient</label>
    <input type="name" class="form-control" name="name">
  </div>
  <div class="form-group">
    <label for="emailreceiver">Email of the blood Receiver</label>
    <input type="emailreceiver" class="form-control" name="emailreceiver">
  </div>
  <div class="form-group">
    <label for="email">Email of the blood donator</label>
    <input type="email" class="form-control" name="email">
  </div>
  <div class="form-group">
    <label for="donated">Withdrawal of blood succeeded ?</label>
    <select name="donated" class="form-control" id="donated">
      <option value="yes">Yes</option>
      <option value="no">No</option>
    </select>
  </div>
  <div class="form-group">
    <label for="hospitalname">Hospital Name</label>
    <input type="hospitalname" class="form-control" name="hospitalname">
  </div>
  <button type="submit" class="bg-dark btn colormine">Submit</button>
  <br><br><br><br><br><br><br><br><br>
</form>
  <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdsJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

inchargelogin.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Incharge Login Page</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
  <style type="text/css">
body{
  background-color: #DDDDDD;
}

.msg,.login{
  text-align: center;
  margin-left: auto;
  margin-right: auto;
}

form{
  width: 30%;
}

.colormine{
  color: white;
}

.center {
  position: absolute;
  left: 0;
  right: 0;
  margin: auto;
}
</style>
</head>
<body>
<nav class="h1 navbar navbar-light p-3 mb-2 bg-dark text-white justify-content-between">
  <a class="navbar-brand">Incharge Login Page</a>
</nav>
<br>
<h5 class="msg">{{ msg }}</h5>
<br>
<form action="/inchargelogin" method="POST" class="center">
  <div class="form-group">
```



```

<label for="inputEmail1">Email</label>
<input type="text" name="text" class="form-control" id="inputEmail1" placeholder="Enter Email">
</div>
<div class="form-group">
  <label for="InputPassword1">Password</label>
  <input type="password" name="password" class="form-control" id="InputPassword1" placeholder="Password">
</div>
<div>
  New User ? Click to <a href="{ {url_for('inchargeregister') }}">Sign-Up</a>
</div>
<br>
<button type="submit" class="btn btn-dark">Submit</button>
</form>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DkTlIkVYIK3UENzmM7KCKRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl" crossorigin="anonymous"></script>
</body>
</html>

```

inchargeregister.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Incharge Register page</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <style type="text/css">
body{
  background-color: #DDDDDD;
}
.colormine{
  color: white;
}
.bold{
  font-weight: bold;
}
.center {

```

```

position: absolute;
left: 0;
right: 0;
margin: auto;
}
form{
width: 30%;
}
</style>
</head>
<body>
<nav class="h1 navbar navbar-light p-3 mb-2 bg-dark text-white justify-content-between">
<a class="navbar-brand">Register Page</a>
</nav>
<form action="/inchargeregister" method="POST" class="center">
<h5 class="msg" style="color:red;">{{ msg1 }}</h5>
<h5 class="msg" style="color:red;">{{ msg2 }}</h5>
<div class="form-group">
<label for="Hospitalname">Hospital Name</label>
<input type="text" name="hospitalname" class="form-control" id="Hospitalname" placeholder="Enter Hospital name"
required>
</div>
<div class="form-group">
<label for="Email">Email</label>
<input type="text" name="email" class="form-control" id="Email" placeholder="Enter Email" required>
</div>
<div class="form-group">
<label for="PhoneNumber">Phone Number</label>
<input type="tel" id="phone" name="phonenumber" class="form-control" placeholder="1234567890" pattern="[0-9]
{3}[0-9]{3}[0-9]{4}" required>
</div>
<div class="form-group">
<label for="District">District</label>
<input type="text" name="district" class="form-control" id="District" placeholder="Enter District" required>
</div>
<div class="form-group">
<label for="PINCODE">PIN CODE</label>
<input type="text" name="pincode" class="form-control" id="PINCODE" placeholder="Enter PINCODE" required>
</div>
<div class="form-group">
<label for="Address">Address</label>
<input type="text" name="address" class="form-control" id="Address" placeholder="Enter Address" required>
</div>

```

```

<div class="form-group">
  <label for="Password">Password</label>
  <input type="password" name="password" class="form-control" id="Password" placeholder="Password"
onkeyup='check();' required>
</div>
<div class="form-group">
  <label for="RetypePassword">Re-Type Password</label>
  <input type="password" name="retypepassword" class="form-control" id="RetypePassword" placeholder="Re-Type
Your Password" onkeyup='check();' required>
  <span id='message'></span>
</div>
<div>
  Already a User ? Click to <a href="{ {url_for('inchargelogin')}}">Login</a>
</div>
<br>
<button type="submit" class="btn btn-dark">Submit</button>
<br>
<br>
<br>
<br>
<br><br>
<br>
<br>
<br><br>
<br>
<br>
<br>
</form>
<script>
var check = function() {
  if (document.getElementById('Password').value === document.getElementById('RetypePassword').value) {
    document.getElementById('message').style.color = 'green';
    document.getElementById('message').innerHTML = 'Matching';
  }
  else {
    document.getElementById('message').style.color = 'red';
    document.getElementById('message').innerHTML = 'Not Matching';
  }
}
</script>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>

```

```

<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIly6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

indexpage.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 Application</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <style>
    .cards{
      padding-bottom: 0%;
      padding-left: 10%;
      padding-right: 5%;
      margin-top: 10%;
    }
    body{
      background-color: #DDDDDD;
    }
    .colormine{
      color: white;
      font-size: 100%;
    }
    .bold{
      font-weight: bold;
    }
    .center {
      position: absolute;
      left: 0;
      right: 0;
      margin: auto;
    }
    form{
      width: 30%;
    }
  </style>
</head>

```

```

<body>
<nav class="navbar navbar-expand-lg navbar-light bg-dark">
  <a class="navbar-brand" href="#"><span class="colormine bold">Plasma Donor Application</span></a>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#"> <span class="sr-only">(current)</span></a>
      </li>
    </ul>
  </div>
</nav>
<div class="cards">
  <div class="row">
    <div class="col-md-4">
      <div class="card" style="width: 18rem;" >
        <div class="card-body">
          <h2 class="card-title bold">Donor Login</h2>
          <p class="card-text">If you want to donate blood, Click here.</p>
          <a href="{{ url_for('donorlogin') }}" class="btn btn-dark">DONOR</a>
        </div>
      </div>
    </div>
    <div class="col-md-4">
      <div class="card" style="width: 18rem;" >
        <div class="card-body">
          <h2 class="card-title bold">Recipient Login</h2>
          <p class="card-text">If you want to receive blood, Click here.</p>
          <a href="{{ url_for('recipientlogin') }}" class="btn btn-dark">RECIPIENT</a>
        </div>
      </div>
    </div>
    <div class="col-md-4">
      <div class="card" style="width: 18rem;" >
        <div class="card-body">
          <h2 class="card-title bold">Incharge Login</h2>
          <p class="card-text">If you are an Hospital Incharge, Click here.</p>
          <a href="{{ url_for('inchargelogin') }}" class="btn btn-dark">INCHARGE</a>
        </div>
      </div>
    </div>
  </div>
</div>

```

```

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abfTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

recipientauthentication.html

```

<!DOCTYPE html>
<html>
<head>
  <title>OTP Verification</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
  <style>
body{
  background-color: #DDDDDD;
}
.colormine{
  color: white;
}
.bold{
  font-weight: bold;
}
.center {
  position: absolute;
  left: 0;
  right: 0;
  margin: auto;
}
form{
  width: 30%;
}
</style>
</head>

<body>
<nav class="navbar navbar-expand-lg navbar-light bg-dark">
  <a class="navbar-brand" href="#"><span class="colormine bold">Recipient Authentication</span></a>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">

```

```

    <li class="nav-item active">
    <a class="nav-link" href="#"> <span class="sr-only">(current)</span></a>
    </li>
  </ul>
</div>
</nav>

<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h4 class="display-6"> One-time password has been sent to the email id. Please check the email {{email}} for the
    verification.</h4>
  </div>
</div>

  <form action = "/recipientvalidate" method="post" class="center">
<div class="form-group">
  <label>Email</label>
  <input type="text" value="{{email}}" class="form-control" name="email" readonly>
</div>
<div class="form-group">
  <label>Enter OTP:</label>
  <input type="text" class="form-control" name="otp">
</div>
  <input type="submit" value="Submit">
</form>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1cLHTMga3JDZwnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

recipienthome.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>Recipient Home</title>

```

```

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-
ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
<style>
  body{
    background-color: #DDDDDD;
  }
  .colormine{
    color: white;
  }
  .bold{
    font-weight: bold;
  }
  .center {
    position: absolute;
    left: 0;
    right: 0;
    margin: auto;
  }
  form{
    width: 30%;
  }
</style>
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-light bg-dark">
    <a class="navbar-brand" href="#"><span class="colormine bold">Recipient Home</span></a>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="#"> <span class="sr-only">(current)</span></a>
        </li>
      </ul>
      <a class="form-inline my-2 my-lg-0" href="{{ url_for('recipientlogout') }}"><button class="bg-light btn">Log
Out</button></a>
    </div>
  </nav>

  <div class="jumbotron jumbotron-fluid">
    <div class="container">
      <h4 class="display-6">You are signed-in using this email {{ email }}.</h4>
      <br>
    </div>

```



```

</div>

<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h1 class="display-4">Request for Blood</h1>
  </div>
</div>
<br><br>
<form action="/recipienthome" method="POST" class="center">
  <br><br>
  <div class="form-group">
    <label for="recipientname">Name</label>
    <input type="text" name="recipientname" class="form-control">

  </div>
  <div class="form-group">
    <label for="email">Email</label>
    <input type="text" class="form-control" name="email">

  </div>
  <div class="form-group">
    <label for="phonenumber">Phone number</label>
    <input type="text" class="form-control" name="phonenumber">

  </div>
  <div class="form-group">
    <label for="district">District</label>
    <select class="form-control" class="form-control" id="district" name="district">
      { % for district in districts % }
      <option value="{{ district }}">{{ district }}</option>
      { % endfor % }
    </select>
  </div>
  <div class="form-group">
    <label for="hospitalname">Hospital name</label>
    <select name="hospitalname" class="form-control" id="hospitalname"></select>
  </div>
  <div class="form-group">
    <label for="blood">Blood Group</label>
    <select name="blood" class="form-control" id="bloodname"></select>
  </div>
  <button type="submit" class="bg-dark btn colormine">Submit</button>
  <br>

```

```
<br><br><br><br><br><br><br><br><br>
</form>
```

```
<script type="text/javascript">
```

```
district_select = document.getElementById('district')
hospital_select = document.getElementById('hospitalname')
blood_select=document.getElementById('bloodname')
```

```
district_select.onchange=function(){
    district = district_select.value;
    //alert(district);
    fetch('/hospitalhaving/'+district).then(function(response){
    response.json().then(function(data){
        optionHTML="";
        for ( hospital of data.districthospital){
            optionHTML+=<option value="" +hospital+"">'+hospital+'</option>';
        }
        for (var i = 0; i < data.districthospital.length; i++) {
            console.log(data.districthospital[i]);
        }
        hospital_select.innerHTML=optionHTML;
    });
    });
}
```

```
hospital_select.onchange=function(){
    hospital = hospital_select.value;
    fetch('/blood/'+hospital).then(function(response1){
        response1.json().then(function(data1){
            optionHTML="";
            for(blood of data1.bloodshospital){
                optionHTML+=<option value="" +blood+"">'+blood+'</option>';
            }
            for(var i=0; i<data1.bloodshospital.length;i++){
                console.log(data1.bloodshospital[i]);
            }
            blood_select.innerHTML=optionHTML;
        });
    });
}
```

```
</script>
```

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-
q8i/X+965DzO0rT7abK41JstQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
```

```

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
UO2eT0CpHqdSJK6hJty5KVphtPhzWj9WO1clHTMGa3JDZwnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
</body>
</html>

```

recipientlogin.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Login Page</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
  <style type="text/css">
    body{
      background-color: #DDDDDD;
    }

    .msg,.login{
      text-align: center;
      margin-left: auto;
      margin-right: auto;
    }

    form{
      width: 30%;
    }

    .colormine{
      color: white;
    }

    .center {
      position: absolute;
      left: 0;
      right: 0;
      margin: auto;
    }
  </style>
</head>
<body>
  <nav class="h1 navbar navbar-light p-3 mb-2 bg-dark text-white justify-content-between">

```

```

<a class="navbar-brand">Recipient Login Page</a>
</nav>
<br>
<h5 class="msg">{{ msg }}</h5>
<br>
<form action="/recipientlogin" method="POST" class="center">
  <div class="form-group">
    <label for="exampleInputEmail1">Email</label>
    <input type="text" name="text" class="form-control" id="exampleInputEmail1" placeholder="Enter Email">
  </div>
  <div class="form-group">
    <label for="exampleInputPassword1">Password</label>
    <input type="password" name="password" class="form-control" id="exampleInputPassword1"
placeholder="Password">
  </div>
  <div>
    New User ? Click to <a href="{{ url_for('recipientregister') }}">Sign-Up</a>
  </div>
  <br>
  <button type="submit" class="btn btn-dark">Submit</button>
</form>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KCKRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous">
</script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmY1" crossorigin="anonymous"></script>
</body>
</html>

```

recipientregister.html

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Register page</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
  <style type="text/css">
    body{
      background-color: #DDDDDD;

```

```

}

.msg,.login{
    text-align: center;
    margin-left: auto;
    margin-right: auto;
}

form{
    width: 30%;
}

.colormine{
    color: white;
}

.center {
    position: absolute;
    left: 0;
    right: 0;
    margin: auto;
}
</style>
</head>
<body>
<nav class="h1 navbar navbar-light p-3 mb-2 bg-dark text-white justify-content-between">
    <a class="navbar-brand">RECIPIENT Register Page</a>
</nav>
<form action="/recipientregister" method="POST" class="center">
    <h5 class="msg" style="color:red;">{{ msg1 }}</h5>
    <h5 class="msg" style="color:red;">{{ msg2 }}</h5>
    <div class="form-group">
        <label for="Username">User Name</label>
        <input type="text" name="username" class="form-control" id="Username" onkeypress="clsAlphaNoOnly(event)"
onpaste="return false;" placeholder="Enter Username" required>
    </div>
    <div class="form-group">
        <label for="Firstname">First Name</label>
        <input type="text" name="firstname" class="form-control" id="Firstname" onkeypress="clsAlphaOnly(event)"
onpaste="return false;" placeholder="Enter Firstname" required>
    </div>
    <div class="form-group">
        <label for="Lastname">Last Name</label>
        <input type="text" name="lastname" class="form-control" id="Lastname" onkeypress="clsAlphaOnly(event)"
onpaste="return false;" placeholder="Enter Lastname" required>

```

```

</div>
<div class="form-group">
  <label for="gender" class="control-label">Gender</label>
  <div>
    <select class="form-control" id="Gender" name="gender">
      <option value="Others">Others</option>
      <option value="Male">Male</option>
      <option value="Female">Female</option>
    </select>
  </div>
</div>
<div class="form-group">
  <label for="Age">Age</label>
  <input type="number" name="age" id="Age" class="form-control" onkeypress="clsNoOnly(event)"
onpaste="return false;" placeholder="Enter Age" min="1" max="120" step="1" required>
</div>
<div class="form-group">
  <label for="Blood">Blood Group</label>
  <input type="text" name="blood" id="Blood" class="form-control" placeholder="Enter Blood Group as A+ or O+ etc."
required>
</div>
<div class="form-group">
  <label for="Email">Email</label>
  <input type="text" name="email" class="form-control" id="Email" placeholder="Enter Email" required>
</div>
<div class="form-group">
  <label for="PhoneNumber">Phone Number</label>
  <input type="tel" id="phone" name="phonenumber" class="form-control" onkeypress="clsNoOnly(event)"
onpaste="return false;" placeholder="1234567890" pattern="[0-9]{3}[0-9]{3}[0-9]{4}" required>
</div>
<div class="form-group">
  <label for="District">District</label>
  <input type="text" name="district" class="form-control" id="District" onkeypress="clsAlphaOnly(event)"
onpaste="return false;" placeholder="Enter District" required>
</div>
<div class="form-group">
  <label for="PINCODE">PIN CODE</label>
  <input type="text" name="pincode" class="form-control" id="PINCODE" onkeypress="clsNoOnly(event)"
onpaste="return false;" placeholder="Enter PINCODE" required>
</div>
<div class="form-group">
  <label for="Password">Password</label>
  <input type="password" name="password" class="form-control" id="Password" placeholder="Password"

```

```

onkeyup='check();' required>
</div>
<div class="form-group">
  <label for="RetypePassword">Re-Type Password</label>
  <input type="password" name="retypepassword" class="form-control" id="RetypePassword"
placeholder="Re-Type Your Password" onkeyup='check();' required>
  <span id='message'></span>
</div>
<div>
  Already a User ? Click to <a href="{ {url_for('recipientlogin') }}">Login</a>
</div>
<br>
<button type="submit" class="btn btn-dark">Submit</button>
<br>
<br>
<br>
<br>
</form>
<script>
  var check = function() {
    if (document.getElementById('Password').value == document.getElementById('RetypePassword').value) {
      document.getElementById('message').style.color = 'green';
      document.getElementById('message').innerHTML = 'Matching';
    }
    else {
      document.getElementById('message').style.color = 'red';
      document.getElementById('message').innerHTML = 'Not Matching';
    }
  }
  function clsAlphaNoOnly (e) { // Accept only alpha numerics, no special characters
    var regex = new RegExp("[a-zA-Z0-9 ]+$");
    var str = String.fromCharCode(!e.charCode ? e.which : e.charCode);
    if (regex.test(str)) {
      return true;
    }

    e.preventDefault();
    return false;
  }
  function clsAlphaOnly (e) { // Accept only alphabets, no special characters
    var regex = new RegExp("[a-zA-Z]+$");
    var str = String.fromCharCode(!e.charCode ? e.which : e.charCode);
    if (regex.test(str)) {

```

```

        return true;
    }

    e.preventDefault();
    return false;
}

function clsNoOnly (e) { // Accept only numerics, no special characters
    var regex = new RegExp("^[0-9 ]+$");
    var str = String.fromCharCode(!e.charCode ? e.which : e.charCode);
    if (regex.test(str)) {
        return true;
    }

    e.preventDefault();
    return false;
}
</script>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous">
</script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYI" crossorigin="anonymous"></script>
</body>
</html>

```

recipientstatus.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>Recipient Status</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
    <style type="text/css">
        body{
            background-color: #DDDDDD;
        }
    </style>

```



```

.msg,.login{
    text-align: center;
    margin-left: auto;
    margin-right: auto;
}

form{
    width: 30%;
}

.colormine{
    color: white;
}

.center {
    position: absolute;
    left: 0;
    right: 0;
    margin: auto;
}
</style>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-light bg-dark">
  <a class="navbar-brand" href="#"><span class="colormine bold">Recipient Status</span></a>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#"> <span class="sr-only">(current)</span></a>
      </li>
    </ul>
    <a class="form-inline my-2 my-lg-0" href="{{ url_for('recipientlogout') }}"><button class="bg-light btn">Log
Out</button></a>
  </div>
</nav>
<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h1 class="display-4">Hello {{firstname}}</h1>
    <h4 class="display-6">You are signed-in using this email {{email}} </h4>
    <br>
  </div>
</div>
<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h4 class="display-6">{{ send }}</h4>

```

```

    <h4 class="display-6">{ { msg1 } }</h4>
</div>
</div>
<br>
<br>
<h3> <center>Feedback Form</center> </h3>
<form action="/feedback" class="center" method="post">
    <div class="form-group">
        <label>Username</label>
        <input type="text" class="form-control" name="username">
    </div>
    <div class="form-group">
        <label for="firstname">Firstname</label>
        <input type="text" class="form-control" name="firstname">
    </div>
    <div class="form-group">
        <label for="email">Email</label>
        <input type="text" name="email" class="form-control">
    </div>
    <div class="form-group">
        <label for="phonenumber">Phone number</label>
        <input type="text" name="phonenumber" class="form-control">
    </div>
    <div class="form-group">
        <label for="district">District</label>
        <input type="text" name="district" class="form-control">
    </div>
    <div class="form-group">
        <label for="hospitalname">Hospital name</label>
        <input type="text" name="hospitalname" class="form-control">
    </div>
    <div class="form-group">
        <label for="donationdate">Donation date</label>
        <input type="date" class="form-control" name="donationdate">
    </div>
    <div class="form-group">
        <label for="blood">Blood Group</label>
        <input type="text" class="form-control" name="blood">
    </div>
    <div class="form-group">
        <label for="me">I'm a Donor/Recipient</label>
        <select id="me" class="form-control" name="me">
            <option value="">---</option>

```

```

        <option value="donor">Donor</option>
        <option value="recipient">Recipient</option>
    </select>
</div>
<div class="form-group">
    <label for="treat">How they treated you ?</label>
    <textarea type="text" class="form-control" name="treat"></textarea>
</div>
<button type="submit" class="btn btn-dark">Submit</button>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
<br>
</form>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js" integrity="sha384-
KJ3o2DKtIkVYIK3UENzmM7KCKRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js" integrity="sha384-
ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js" integrity="sha384-
JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYI" crossorigin="anonymous"></script>
</body>
</html>

```

Dockerfile

```

FROM python
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
CMD ["python","index.py"]
EXPOSE 5000

```

requirements.txt

```

ibm-db==3.1.3
Flask==2.0.2
Flask-Mail==0.9.1

```

SendGrid Intergration With Python Code.py

```
import os
import json

from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import *

# NOTE: you will need move this file to the root
# directory of this project to execute properly.

def build_hello_email():
    ## Send a Single Email to a Single Recipient

    message = Mail(from_email=From('from@example.com', 'Example From Name'),
        to_emails=To('to@example.com', 'Example To Name'),
        subject=Subject('Sending with SendGrid is Fun'),
        plain_text_content=PlainTextContent('and easy to do anywhere, even with Python'),
        html_content=HtmlContent('<strong>and easy to do anywhere, even with Python</strong>'))

    try:
        print(json.dumps(message.get(), sort_keys=True, indent=4))
        return message.get()

    except SendGridException as e:
        print(e.message)

    mock_personalization = Personalization()
    personalization_dict = get_mock_personalization_dict()

    for cc_addr in personalization_dict['cc_list']:
        mock_personalization.add_to(cc_addr)

    for bcc_addr in personalization_dict['bcc_list']:
        mock_personalization.add_bcc(bcc_addr)

    for header in personalization_dict['headers']:
        mock_personalization.add_header(header)

    for substitution in personalization_dict['substitutions']:
        mock_personalization.add_substitution(substitution)
```

```

for arg in personalization_dict['custom_args']:
    mock_personalization.add_custom_arg(arg)

mock_personalization.subject = personalization_dict['subject']
mock_personalization.send_at = personalization_dict['send_at']

message.add_personalization(mock_personalization)

return message

def get_mock_personalization_dict():
    """Get a dict of personalization mock."""
    mock_pers = dict()

    mock_pers['to_list'] = [To("test1@example.com",
                               "Example User"),
                           To("test2@example.com",
                               "Example User")]

    mock_pers['cc_list'] = [To("test3@example.com",
                               "Example User"),
                           To("test4@example.com",
                               "Example User")]

    mock_pers['bcc_list'] = [To("test5@example.com"),
                             To("test6@example.com")]

    mock_pers['subject'] = ("Hello World from the Personalized "
                             "SendGrid Python Library")

    mock_pers['headers'] = [Header("X-Test", "test"),
                             Header("X-Mock", "true")]

    mock_pers['substitutions'] = [Substitution("%name%", "Example User"),
                                   Substitution("%city%", "Denver")]

    mock_pers['custom_args'] = [CustomArg("user_id", "343"),
                                 CustomArg("type", "marketing")]

    mock_pers['send_at'] = 1443636843
    return mock_pers

def build_multiple_emails_personalized():

```

Note that the domain for all From email addresses must match

```
message = Mail(from_email=From('from@example.com', 'Example From Name'),
               subject=Subject('Sending with SendGrid is Fun'),
               plain_text_content=PlainTextContent('and easy to do anywhere, even with Python'),
               html_content=HtmlContent('<strong>and easy to do anywhere, even with Python</strong>'))
```

```
mock_personalization = Personalization()
mock_personalization.add_to(To('test@example.com', 'Example User 1'))
mock_personalization.add_cc(Cc('test1@example.com', 'Example User 2'))
message.add_personalization(mock_personalization)
```

```
mock_personalization_2 = Personalization()
mock_personalization_2.add_to(To('test2@example.com', 'Example User 3'))
mock_personalization_2.set_from(From('from@example.com', 'Example From Name 2'))
mock_personalization_2.add_bcc(Bcc('test3@example.com', 'Example User 4'))
message.add_personalization(mock_personalization_2)
```

```
try:
    print(json.dumps(message.get(), sort_keys=True, indent=4))
    return message.get()
```

```
except SendGridException as e:
    print(e.message)
```

```
return message
```

```
def build_attachment1():
    """Build attachment mock. Make sure your content is base64 encoded before passing into attachment.content.
    Another example: https://github.com/sendgrid/sendgrid-python/blob/HEAD/use\_cases/attachment.md"""
```

```
attachment = Attachment()
attachment.file_content = ("TG9yZW0gaXBzdW0gZG9sb3Igc2l0IGFtZXQsIGNvbml"
                           "Y3RldHVyIGFkaXBpc2NpbmcgZWxpdC4gQ3JhcyBwdW12")
attachment.file_type = "application/pdf"
attachment.file_name = "balance_001.pdf"
attachment.disposition = "attachment"
attachment.content_id = "Balance Sheet"
return attachment
```

```
def build_attachment2():
    """Build attachment mock."""
```

```

attachment = Attachment()
attachment.file_content = "BwdW"
attachment.file_type = "image/png"
attachment.file_name = "banner.png"
attachment.disposition = "inline"
attachment.content_id = "Banner"
return attachment

def build_kitchen_sink():
    """All settings set"""
    from sendgrid.helpers.mail import (
        Mail, From, To, Cc, Bcc, Subject, PlainTextContent,
        HtmlContent, SendGridException, Substitution,
        Header, CustomArg, SendAt, Content, MimeType, Attachment,
        FileName, FileContent, FileType, Disposition, ContentId,
        TemplateId, Section, ReplyTo, Category, BatchId, Asm,
        GroupId, GroupsToDisplay, IpPoolName, MailSettings,
        BccSettings, BccSettingsEmail, BypassListManagement,
        FooterSettings, FooterText, FooterHtml, SandBoxMode,
        SpamCheck, SpamThreshold, SpamUrl, TrackingSettings,
        ClickTracking, SubscriptionTracking, SubscriptionText,
        SubscriptionHtml, SubscriptionSubstitutionTag,
        OpenTracking, OpenTrackingSubstitutionTag, Ganalytics,
        UtmSource, UtmMedium, UtmTerm, UtmContent, UtmCampaign)
    import time
    import datetime

    message = Mail()

    # Define Personalizations

    message.to = To('test1@sendgrid.com', 'Example User1', p=0)
    message.to = [
        To('test2@sendgrid.com', 'Example User2', p=0),
        To('test3@sendgrid.com', 'Example User3', p=0)
    ]

    message.cc = Cc('test4@example.com', 'Example User4', p=0)
    message.cc = [
        Cc('test5@example.com', 'Example User5', p=0),
        Cc('test6@example.com', 'Example User6', p=0)
    ]

```

```

message.bcc = Bcc('test7@example.com', 'Example User7', p=0)
message.bcc = [
    Bcc('test8@example.com', 'Example User8', p=0),
    Bcc('test9@example.com', 'Example User9', p=0)
]

message.subject = Subject('Sending with SendGrid is Fun 0', p=0)

message.header = Header('X-Test1', 'Test1', p=0)
message.header = Header('X-Test2', 'Test2', p=0)
message.header = [
    Header('X-Test3', 'Test3', p=0),
    Header('X-Test4', 'Test4', p=0)
]

message.substitution = Substitution('%name1%', 'Example Name 1', p=0)
message.substitution = Substitution('%city1%', 'Example City 1', p=0)
message.substitution = [
    Substitution('%name2%', 'Example Name 2', p=0),
    Substitution('%city2%', 'Example City 2', p=0)
]

message.custom_arg = CustomArg('marketing1', 'true', p=0)
message.custom_arg = CustomArg('transactional1', 'false', p=0)
message.custom_arg = [
    CustomArg('marketing2', 'false', p=0),
    CustomArg('transactional2', 'true', p=0)
]

message.send_at = SendAt(1461775051, p=0)

message.to = To('test10@example.com', 'Example User10', p=1)
message.to = [
    To('test11@example.com', 'Example User11', p=1),
    To('test12@example.com', 'Example User12', p=1)
]

message.cc = Cc('test13@example.com', 'Example User13', p=1)
message.cc = [
    Cc('test14@example.com', 'Example User14', p=1),
    Cc('test15@example.com', 'Example User15', p=1)
]

```



```

message.bcc = Bcc('test16@example.com', 'Example User16', p=1)
message.bcc = [
    Bcc('test17@example.com', 'Example User17', p=1),
    Bcc('test18@example.com', 'Example User18', p=1)
]

message.header = Header('X-Test5', 'Test5', p=1)
message.header = Header('X-Test6', 'Test6', p=1)
message.header = [
    Header('X-Test7', 'Test7', p=1),
    Header('X-Test8', 'Test8', p=1)
]

message.substitution = Substitution('%name3%', 'Example Name 3', p=1)
message.substitution = Substitution('%city3%', 'Example City 3', p=1)
message.substitution = [
    Substitution('%name4%', 'Example Name 4', p=1),
    Substitution('%city4%', 'Example City 4', p=1)
]

message.custom_arg = CustomArg('marketing3', 'true', p=1)
message.custom_arg = CustomArg('transactional3', 'false', p=1)
message.custom_arg = [
    CustomArg('marketing4', 'false', p=1),
    CustomArg('transactional4', 'true', p=1)
]

message.send_at = SendAt(1461775052, p=1)

message.subject = Subject('Sending with SendGrid is Fun 1', p=1)

# The values below this comment are global to entire message

message.from_email = From('help@twilio.com', 'Twilio SendGrid')

message.reply_to = ReplyTo('help_reply@twilio.com', 'Twilio SendGrid Reply')

message.subject = Subject('Sending with SendGrid is Fun 2')

message.content = Content(MimeType.text, 'and easy to do anywhere, even with Python')
message.content = Content(MimeType.html, '<strong>and easy to do anywhere, even with Python</strong>')
message.content = [
    Content('text/calendar', 'Party Time!!'),

```

```

Content('text/custom', 'Party Time 2!!')
]

message.attachment = Attachment(FileContent('base64 encoded content 1'),
    FileName('balance_001.pdf'),
    FileType('application/pdf'),
    Disposition('attachment'),
    ContentId('Content ID 1'))
message.attachment = [
    Attachment(FileContent('base64 encoded content 2'),
        FileName('banner.png'),
        FileType('image/png'),
        Disposition('inline'),
        ContentId('Content ID 2')),
    Attachment(FileContent('base64 encoded content 3'),
        FileName('banner2.png'),
        FileType('image/png'),
        Disposition('inline'),
        ContentId('Content ID 3'))
]

message.template_id = TemplateId('13b8f94f-bcae-4ec6-b752-70d6cb59f932')

message.section = Section('%section1%', 'Substitution for Section 1 Tag')
message.section = [
    Section('%section2%', 'Substitution for Section 2 Tag'),
    Section('%section3%', 'Substitution for Section 3 Tag')
]

message.header = Header('X-Test9', 'Test9')
message.header = Header('X-Test10', 'Test10')
message.header = [
    Header('X-Test11', 'Test11'),
    Header('X-Test12', 'Test12')
]

message.category = Category('Category 1')
message.category = Category('Category 2')
message.category = [
    Category('Category 1'),
    Category('Category 2')
]

```

```

message.custom_arg = CustomArg('marketing5', 'false')
message.custom_arg = CustomArg('transactional5', 'true')
message.custom_arg = [
    CustomArg('marketing6', 'true'),
    CustomArg('transactional6', 'false')
]

message.send_at = SendAt(1461775053)

message.batch_id = BatchId("HkJ5yLYULb7Rj8GKSx7u025ouWVIMgAi")

message.asm = Asm(GroupId(1), GroupsToDisplay([1,2,3,4]))

message.ip_pool_name = IpPoolName("IP Pool Name")

mail_settings = MailSettings()
mail_settings.bcc_settings = BccSettings(False, BccSettingsTo("bcc@twilio.com"))
mail_settings.bypass_list_management = BypassListManagement(False)
mail_settings.footer_settings = FooterSettings(True, FooterText("w00t"), FooterHtml("<string>w00t!<strong>"))
mail_settings.sandbox_mode = SandBoxMode(True)
mail_settings.spam_check = SpamCheck(True, SpamThreshold(5), SpamUrl("https://example.com"))
message.mail_settings = mail_settings

tracking_settings = TrackingSettings()
tracking_settings.click_tracking = ClickTracking(True, False)
tracking_settings.open_tracking = OpenTracking(True, OpenTrackingSubstitutionTag("open_tracking"))
tracking_settings.subscription_tracking = SubscriptionTracking(
    True,
    SubscriptionText("Goodbye"),
    SubscriptionHtml("<strong>Goodbye!</strong>"),
    SubscriptionSubstitutionTag("unsubscribe"))
tracking_settings.ganalytics = Ganalytics(
    True,
    UtmSource("utm_source"),
    UtmMedium("utm_medium"),
    UtmTerm("utm_term"),
    UtmContent("utm_content"),
    UtmCampaign("utm_campaign"))
message.tracking_settings = tracking_settings

return message

def send_multiple_emails_personalized():

```

```

# Assumes you set your environment variable:
# https://github.com/sendgrid/sendgrid-python/blob/HEAD/TROUBLESHOOTING.md#environment-variables-and-your-
sendgrid-api-key
message = build_multiple_emails_personalized()
sendgrid_client = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
response = sendgrid_client.send(message=message)
print(response.status_code)
print(response.body)
print(response.headers)

def send_hello_email():
    # Assumes you set your environment variable:
    # https://github.com/sendgrid/sendgrid-python/blob/HEAD/TROUBLESHOOTING.md#environment-variables-and-your-
sendgrid-api-key
    message = build_hello_email()
    sendgrid_client = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sendgrid_client.send(message=message)
    print(response.status_code)
    print(response.body)
    print(response.headers)

def send_kitchen_sink():
    # Assumes you set your environment variable:
    # https://github.com/sendgrid/sendgrid-python/blob/HEAD/TROUBLESHOOTING.md#environment-variables-and-your-
sendgrid-api-key
    message = build_kitchen_sink()
    sendgrid_client = SendGridAPIClient(os.environ.get('SENDGRID_API_KEY'))
    response = sendgrid_client.send(message=message)
    print(response.status_code)
    print(response.body)
    print(response.headers)

## this will actually send an email
# send_hello_email()

## this will send multiple emails
# send_multiple_emails_personalized()

## this will only send an email if you set SandBox Mode to False
# send_kitchen_sink()

```