

PLASMA DONOR APPLICATION

DOMAIN: CLOUD APP DEVELOPMENT

TEAM ID: PNT2022TMID11654

BATCH: B10-4A6E

TEAM

Team Leader : MEDUNAN S

Team member : MAKESH S

Team member : MADHUSUTHANAN G

Team member: MOHAMMED ATHANAN M

CONTENTS

1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

2. LITERATURE SURVEY

2.1 Existing problem

2.2 References

2.3 Problem Statement Definition

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution fit

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

4.2 Non-Functional requirements

5. PROJECT DESIGN

5.1 Data Flow Diagrams

5.2 Solution & Technical Architecture

5.3 User Stories

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

6.3 Reports from JIRA

7. CODING & SOLUTIONING

7.1 Feature 1

7.2 Feature 2

7.3 Feature 3

7.4 Feature 4

7.5 Database Schema

8. TESTING

8.1 Test Cases

8.2 User Acceptance Testing

9. RESULTS

9.1 Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX (Source Code)

1. INTRODUCTION

1.1 Project Overview

The application has a simple accessible interface for donors to register and donate blood to the allocated blood bank. Additionally, they can upload a COVID-19 negative certificate, so that their blood plasma can be used for treating COVID-19 patients. The users can create an account and enter their information (name, age, address, blood group etc.) that can be used for registration and scheduling appointments at the nearest blood bank for blood donation.

Blood banks can look at potential donors and book them for an appointment. They can also request potential donors who are registered to the application for their blood. The donors who receive these requests can either accept and book an appointment or reject them.

Hospitals in need of blood plasma can request blood banks according to their needs. Blood banks can look at requests from hospitals and can either accept or reject them.

1.2 Purpose

There has been an increase in demand for blood plasma among hospitals and blood banks as they are additionally used in an experimental treatment for COVID-19. Hence there is a requirement for new infrastructure to facilitate donors, blood banks and hospitals for easier donation and access of blood plasma that could potentially satisfy the excess demand for it to be used for treatment.

Convalescent plasma therapy uses blood from people who've recovered from an illness to help others recover.

This is an experimental form of therapy which is also used to treat COVID-19 by using blood plasma with high antibody levels against COVID-19. It may be used for some hospitalized people ill with COVID-19 who are either early in their illness or who have weakened immune systems.

Blood donated by people who've recovered from COVID-19 has antibodies to the virus that causes it. The donated blood is processed to remove blood cells, leaving behind liquid (plasma) and antibodies. These can be given to people with COVID19 to boost their ability to fight the virus.

As there is no effective antiviral treatment for COVID-19 there is a higher prevalence of other forms of therapy such as convalescent plasma therapy which has increased the demand for blood

plasma. Also, with lockdowns and increased restrictions it has become harder to facilitate donors, furthermore contributing to the scarcity of resources that can be used for treatment.

2. LITERATURE SURVEY

2.1 Existing problem

There has been an increase in demand for blood plasma among hospitals and blood banks as they are additionally used in an experimental treatment for COVID-19. Hence there is a requirement for new infrastructure to facilitate donors, blood banks and hospitals for easier donation and access of blood plasma that could potentially satisfy the excess demand for it to be used for treatment.

2.2 References

1. COVID-19 Convalescent Plasma: from donation to treatment - A Systematic Review & Single Center Experience: <https://tinyurl.com/plasma-systematic-review>
2. Developing a plasma donor application using Function-as-a-service in AWS: <http://ijiird.com/wp-content/uploads/050140.pdf>
3. Blood donors appointment booking and managing system using PC and mobile web browsers in current pandemic (COVID-19): <https://tinyurl.com/bdms-pc-mobile-browser>
4. Blood Bank Management System: <https://www.irjet.net/archives/V8/i6/IRJETV8I668.pdf>
5. A Cross-Platform Blood Donation Application with a Real-Time, Intelligent, and Rational Recommendation System: <https://tinyurl.com/blood-donor-rational-reccsys>
6. Android Blood Bank: <https://tinyurl.com/android-bbms>
7. Android Based Health Care App: <https://www.jetir.org/papers/JETIR2205390.pdf>


2.3 Problem Statement Definition

The main objective is to create an easy-to-use application that can be used by donors to donate their blood to blood banks. Hospitals in need for blood plasma can request blood from blood banks. This application should have a wider reach to appeal to more potential blood donors.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

3.2 Ideation & Brainstorming



Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

10 minutes to prepare
1 hour to collaborate
3-8 people recommended

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes

- 1. **Team gathering**
Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.
- 2. **Set the goal**
Think about the problem you'll be focusing on solving in the brainstorming session.
- 3. **Learn how to use the facilitation tools**
Use the facilitation superpowers to run a happy and productive session.

Open article

Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes

PROBLEM

Creating a hassle-free Application that encourages people to donate plasma

Key rules of brainstorming

To run an impactful and productive session:

- Stay in logic
- Encourage wild ideas
- Defer judgement
- Listen to others
- Go for volume
- If possible, be visual

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Gokul PS

- Details verification before enrollment for donation
- plasma available places nearby
- Chat system between donor and patient
- Integration with social medias

Aswin Tony

- Full system details before details
- Donor eligibility through eligibility
- Create a donor database
- Step by step procedure guide
- Contact of emergency

Krupa

- Sending SMS or email for successful donation
- Contact between donor and receiver
- Report if any issues occurs
- Gathering donor's details
- Create an extraordinary UI

Guhab Priya

- Review system for blood bank & Hospital
- Displaying type of plasma
- Do security and avoid malware before donation
- Certificate of participation

Abinash

- A small walkthrough on how to use the app
- Handling the pending request
- A secure app for storing details
- Patient Testimonials
- Easy to use

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes

Verification

- Details verification before enrollment for donation

Chat System

- Chat system between donor and patient
- Contact between donor and receiver

Things to know while donating

- Donor Eligibility
- Do use Donor's last shared coordinates before donation
- FAQ Blogs
- Step by step procedure guide
- A small walkthrough on how to use the app

support

- Report if any issues occurs
- Sending SMS or email for successful donation
- Patient Testimonials
- Certificate of participation
- Contact of emergency

Backend Services/ Frontend

- Create an extraordinary UI
- Gathering donor's details
- Easy to use
- Full system details before details
- Create a donor database
- Displaying type of plasma
- Handling the pending request
- A secure app for storing details
- Review system for blood bank & Hospital

GeoLocation

- plasma available places nearby

Notifications & services

- Notify the Donor that his/her plasma has been donated to someone
- Integration with social medias

Prioritize

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

20 minutes

Importance

A cluster of ideas that would give clear direction on what to build next. (Does this idea really matter?)

Feasibility

Regardless of their importance, which ideas are more realistic than others? (Can this idea actually be done?)

After you collaborate

You can export the mural as an image or pdf to share with members of your company who might find it helpful.

Quick add-ons

- 1. **Share the mural**
Share a view link to the mural with stakeholders to bring them in the loop about the outcomes of the session.
- 2. **Report the mural**
Export a copy of the mural as a PDF or PPT to attach to emails, internal or external, as a record to your ideas.

Keep moving forward

- 1. **Strategy blueprint**
Define the components of a new idea or strategy.
[Open this template](#)
- 2. **Customer experience journey map**
Understand customer needs, motivations, and obstacles for an experience.
[Open this template](#)
- 3. **Strengths, weaknesses, opportunities & threats**
Identify strengths, weaknesses, opportunities, and threats (SWOT) to develop a plan.
[Open this template](#)

[Share template feedback](#)

3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	There has been an increase in demand for blood plasma among hospitals and blood banks. Hence there is a requirement for new infrastructure to facilitate donors, blood banks and hospitals for easier donation and access of blood plasma that could potentially satisfy the excess demand for it to be used for treatment.
2.	Idea / Solution description	This proposed system aims at connecting the donors & the patients by an online application. By using this application, the users can either raise a request for plasma donation or requirements. The basic solution is to create a centralized system to keep a track on the upcoming as well as past Plasma Donation Events.

3.	Novelty / Uniqueness	<p>A User Interface is simple for users to understand. We can use the application anywhere anytime. The user immediately needs the plasma for their treatment but the plasma is not available in nearby hospitals, then user can use this application to raise request and directly contact the donor, request them to donate the plasma. Today many of them have mobile phones they can install this application and</p>
		<p>use it to save the lives of people.</p>
4.	Social Impact / Customer Satisfaction	<p>Effect of donor motivation on donor satisfaction and loyalty are variable due to the influence of common donorship attitudes prevailing in donor population, impact of social marketing programs, focused on promotion of donor commitment and deliberate donorship. Thus, we have predicted that effect of donor motivation on donor relationship satisfaction and loyalty change.</p>

5.	Business Model (Revenue Model)	<p>This application is accessible by everyone. It is free. Because of the trouble in finding givers who match a specific blood bunch, this application empowers clients to enlist individuals who wish to give plasma and keep their data in a data set. Nowadays the need for plasma is increasing. Anyone with basic technical knowledge can access this app.</p>
----	--------------------------------	---

6	Scalability of the Solution	<p>This application helps users to find plasma donors by sitting in home itself instead of searching donors everywhere. When there is a emergency then plasma request to send to everyone. Once the donor is ready to donate receiver is notified about donation. Receiver can contact the donor. With this app donor can know the eligibility to donate and making it easier to locate suitable donor at right time.</p>
---	-----------------------------	---

4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form
FR-2	User Confirmation	Confirmation via Email
FR-3	Search for donor	Search result can be viewed in a list. Each element in the list represents a specific donor with the donor details.
FR-4	Administrator	Monitor the overall functionalities of the application and ensure quality of service
FR-5	Send Request	If plasma is required, the requester places a request to the admin.

FR-6	User Credentials	Administrator has registered then the software administrator should be able to login to the web application. The login information will be stored on the database for future use.
------	------------------	---

4.2 Non-Functional Requirements

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

FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Plasma donor Application is very useful to the emergency situation patient, because that application gives the information of the nearby plasma donors and request to donate their plasma to patient via email, SMS etc.
NFR-2	Security	Secured website and application that provides various security features Email Verification, password login etc.
NFR-3	Reliability	The plasma donor application should work properly, even when faults occur.

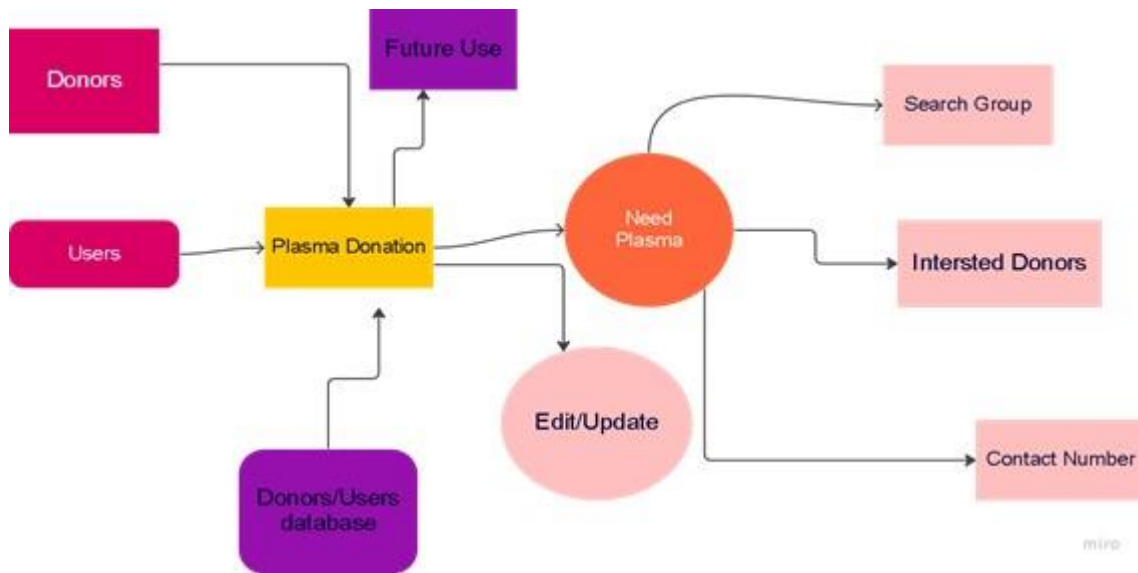
NFR-4	Performance	The plasma donor application must perform well in different scenarios. The Load Time should be less and there needs to be quick response to the request saying that delay is less.
-------	-------------	--

NFR-5	Availability	The system should be available all times, meaning the user can access it using application. In case if a hardware failure or database corruption ,a replacement page will be shown. Also, in case of a hardware failure or database corruption, backups of the database should be retrieved from the application data folder and saved by the administrator. Must be available 24/7.
NFR-6	Scalability	In the application to handle an increase in workload without performance degradation, or its ability to quickly enlarge. The solution must allow the hardware end of the deployed software services and components to be scaled horizontally as well as vertically.

5. PROJECT DESIGN

5.1 Data Flow Diagrams

A data-flow diagram is a way of representing a flow of data through a process or a system. The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops.



5.2 Solution & Technical Architecture

Solution Architecture

- Software Required: Python, Flask, Docker
- System Required: 8GB RAM, Intel Core i3, OS-Windows/Linux/MAC, Laptop or Desktop
- 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.

- In other words, there is a requirement for new infrastructure to facilitate donors, blood banks and hospitals for easier donation and access of blood plasma that could potentially satisfy the excess demand for it to be used for treatment

5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story/Task	Acceptance criteria	Priority	Release
Donor	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
	Login	USN-2	As a user, I can log into the application by entering email and password.	I can receive confirmation email and click confirm	High	Sprint-2

	Register for donor	USN-3	As a user, I can log into the application and search result can be viewed in a list to donate plasma and confirm by booking	I can register & access the dashboard with Facebook Login	Medium	Sprint-3
Patient/donor	Find the bank	USN-4	As a patient, I can directly access the application and find the plasma	I can access my account/dashboard	High	Sprint--1
	Request for plasma	USN-5	As a user, I can enter into the application and find the current bank and request for plasma and state the emergency.	I can register & access the dashboard with Facebook Login	Medium	Sprint-3
Administrator	Maintain the applications	USN-6	As Administrator I can log into the application by entering email & password and maintaining details for users	I can access my account/dashboard	High	Sprint-3

Plasma Bank	Send request	USN-7	As a bank if plasma is required, the receiver will contact the donor.	I can access my account/dashboard	Medium	Sprint-3
	User credentials	USN-8	Operator has registered then the software operator should be able to login to the web application. It will be	I can access my account/dashboard	Medium	Sprint-4
			stored in database for future use.			

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

6.2 Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	12	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	12	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	12	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	12	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

Velocity:

sprint duration = 6 days velocity =
20

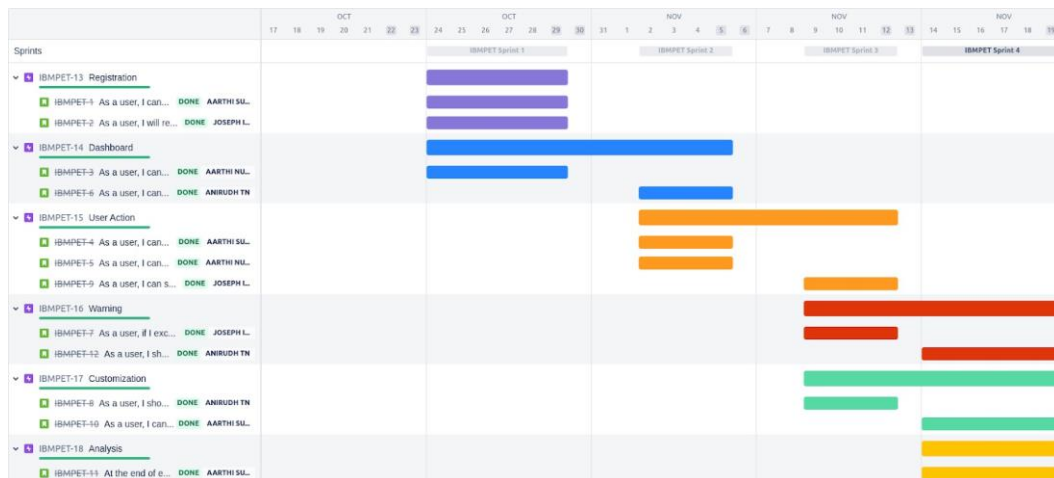
$$AV = VELOCITY / SPRINT DURATION$$

$$AV = 20 / 6$$

$$AV = 3.333$$

6.3 Reports from JIRA

Burndown Chart:



7. CODING & SOLUTIONING

Feature 1 - User Registration with Sendgrid verification

Html code

```
<!DOCTYPE html>
<html>

<head>
  <title>REGISTER</title>
  <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/logins.css') }}">
  <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
</head>

<body>
  <div class="flex items-center justify-center min-h-screen">
    <div class="px-8 py-6 mt-4 text-left bg-black shadow-lg">
      <h3 class="text-2xl font-bold text-center" style="color: white;">Login to your account</h3>
      <form class="regforms" action="/register" method="post">
        <div class="flasher">{{ msg }}</div><br><br>
        <div class="mt-20">
          <div>
            <label class="block" style="color: white;" for="username">Username<label>
              <input type="text" id="username" name="username" placeholder="Enter New Username"
                class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-white"
                style="color: black;">
            </div>
            <div>
              <label class="block" style="color: white;" for="email">Email Id<label>
                <input type="text" id="email" name="email" placeholder="Enter Your Email Id"
                  class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-white"
                  style="color: black;">
              </div>
            </div>
            <div>
              <label class="block" style="color: white;" for="email">Email Id<label>
                <input type="text" id="email" name="email" placeholder="Enter Your Email Id"
                  class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-white"
                  style="color: black;">
              </div>
            <div class="mt-4">
              <label class="block" style="color: white;" for="password">Password<label>
                <input type="password" id="password" name="password" placeholder="Password"
                  class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-white"
                  style="color: black;">
              </div>
            <div class="flex items-baseline justify-between">
              <button class="px-6 py-2 mt-4 text-white bg-blue-600 rounded-lg hover:bg-blue-900">Register</button>
            </div>
          </div>
          </form>
          <p class="text-sm font-light text-gray-500 dark:text-gray-400 my-3">
            Already have an account?
            <a href="/login" class="font-medium text-primary-600 hover:underline dark:text-primary-500">Login</a>
          </p>
        </div>
      </div>
    </div>
  </body>

</html>
```

App.py

```
print("modules imported")
app = Flask(__name__)
print("Flask app created with __name__")
app.config['SECRET_KEY'] = 'top-secret!'
app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] = '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'
app.config['MAIL_DEFAULT_SENDER'] = 'abinash.dr1169@gmail.com'
mail = Mail(app)
print("setup default configs")

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=6667d8e9-9d4d-4ccb-ba32-21da3bb5aafc.c1ogj3sd0tgtu0lqde00.c
print("db conn made")

@app.route('/register', methods=['GET', 'POST'])
def register():
    msg = "Please fill out the form."
    if request.method == 'POST':
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        sql = "SELECT * FROM users WHERE username=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = "Username already exists!"
        elif not re.match(r'^[^\s@]+@[^\s@]+\.[^\s@]+', email):
            msg = "Invalid email address!"
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = "Username must contain only letters and numbers!"
        else:
            insertsql = "INSERT INTO users VALUES(?,?,?)"
            prepstmt = ibm_db.prepare(conn, insertsql)
            ibm_db.bind_param(prepstmt, 1, username)
            ibm_db.bind_param(prepstmt, 2, email)
            ibm_db.bind_param(prepstmt, 3, password)
            ibm_db.execute(prepstmt)

            mailmsg = Message('Registration Verification', recipients=[email])
```

Feature 2 -User login

HTML code:

```
<!DOCTYPE html>
<html>

<head>
  <title>USER LOGIN</title>

  <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/logins.css') }}">
  <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
</head>

<body>
  <div class="flex items-center justify-center min-h-screen">
    <div class="px-8 py-6 mt-4 text-left bg-black shadow-lg">
      <h3 class="text-2xl font-bold text-center" style="color: white;">Login to your account</h3>
      <form class="regforms" action="/login" method="post">
        <div class="flasher">{{ msg }}</div><br><br>
        <div class="mt- 20">
          <div>
            <label class="block" style="color: white;" for="username">Username<label>
              <input type="text" id="username" name="username" placeholder="Username"
                class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-blue-500"
                style="color: black;">
            </div>
            <div class="mt-4">
              <label class="block" style="color: white;" for="password">Password<label>
                <input type="password" id="password" name="password" placeholder="Password"
                  class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-blue-500"
                  style="color: black;">
                <input type="password" id="password" name="password" placeholder="Password"
                  class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-blue-500"
                  style="color: black;">
              </div>
              <div class="flex items-baseline justify-between">
                <button class="px-6 py-2 mt-4 text-white bg-blue-600 rounded-lg hover:bg-blue-900">Login</button>
              </div>
            </div>
          </form>
          <p class="text-sm font-light text-gray-500 dark:text-gray-400 my-3">
            Don't have an account yet?
            <a href="/register" class="font-medium text-primary-600 hover:underline dark:text-primary-500">Sign Up</a>
          </p>
          <p class="text-sm font-light text-gray-500 dark:text-gray-400">
            Admin Login <a href="/adminlogin" class="font-medium text-primary-600 hover:underline dark:text-primary-500">Admin Login</a>
          </p>
        </div>
      </div>
    </div>
  </body>

</html>
```


App.py

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    global userid
    msg = ""
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        sql = "SELECT * FROM users WHERE username=? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['Loggedin'] = True
            session['id'] = account['USERNAME']
            userid = account['USERNAME']
            session['username'] = account['USERNAME']
            return render_template('dashboard.html', msg=username)
        else:
            msg = "Invalid login credentials!"
    return render_template('login.html', msg=msg)
```

Feature 3 - Admin login

Html code:

```
<!DOCTYPE html>
<html>

<head>
    <title>ADMINISTRATOR LOGIN</title>
    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/logins.css') }}">
    <link href="https://unpkg.com/tailwindcss@2/dist/tailwind.min.css" rel="stylesheet">
</head>

<body>
    <div class="flex items-center justify-center min-h-screen">
        <div class="px-8 py-6 mt-4 text-left bg-black shadow-lg">
            <h3 class="text-2xl font-bold text-center" style="color: white;">Admin Login</h3>
            <form action="/adminlogin" method="post">
                <div class="flasher">{{ msg }}</div><br><br>
                <div class="mt-20">
                    <div>
                        <label class="block" style="color: white;" for="username">Username</label>
                        <input type="text" id="username" name="username" placeholder="Username"
                            class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-blue-500"
                            style="color: black;">
                    </div>
                    <div class="mt-4">
                        <label class="block" style="color: white;" for="password">Password</label>
                        <input type="password" id="password" name="password" placeholder="Password"
                            class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-blue-500"
                            style="color: black;">
                    </div>
                </div>
            </form>
        </div>
    </div>
```

```

<div class="mt- -20">
  <div>
    <label class="block" style="color: □white;" for="username">Username<label>
      <input type="text" id="username" name="username" placeholder="Username"
        class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:rin
        style="color: ■black;">
    </div>
    <div class="mt-4">
      <label class="block" style="color: □white;" for="password">Password<label>
        <input type="password" id="password" name="password" placeholder="Password"
          class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:rin
          style="color: ■black;">
      </div>
      <div class="flex items-baseline justify-between">
        <button class="px-6 py-2 mt-4 text-white bg-blue-600 rounded-lg hover:bg-blue-900">Login</butt
      </div>
    </div>
  </form>
  <p class="text-sm font-light text-gray-500 dark:text-gray-400 my-3">
    Click here to login as normal user ->
    <a href="login" class="font-medium text-primary-600 hover:underline dark:text-primary-500">Login</
  </p>
</body>

</html>

```

App.py

```

@app.route('/adminlogin', methods=['GET', 'POST'])
def adminlogin():
    global adminid
    msg = ""
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        sql = "SELECT * FROM admins WHERE username=? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['Loggedin'] = True
            session['id'] = account['USERNAME']
            adminid = account['USERNAME']
            session['username'] = account['USERNAME']
            return redirect(url_for('admindashboard'))
        else:
            msg = "Invalid admin credentials!"
    return render_template('adminlogin.html', msg=msg)

```

Feature 4 - User Views Profile Depending on Donor status

Html code:

```
<!DOCTYPE html>
<html>

<head>
  <title>CURRENT USER'S PROFILE</title>

  <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/tables.css') }}">
  <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/results.css') }}">
  <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
</head>

<body>
  <!-- navbar -->
  <nav>
    <ul>
      <div style="text-align: right;">
        <li><a type="button" href="dashboard">DASHBOARD</a></li>
        <li><a type="button" href="logout">LOGOUT</a></li>
      </div>
    </ul>
    <div class="class-container" id="container">
      <h4 style="text-align: center; color: □white; font-size:25px;">Welcome {{ msg }}!</h4>
    </div>
  </nav>
  <!-- navbar ends -->

  <nav>
    <ul>
      <div style="text-align: right;">
        <li><a type="button" href="dashboard">DASHBOARD</a></li>
        <li><a type="button" href="logout">LOGOUT</a></li>
      </div>
    </ul>
    <div class="class-container" id="container">
      <h4 style="text-align: center; color: □white; font-size:25px;">Welcome {{ msg }}!</h4>
    </div>
  </nav>
  <!-- navbar ends -->

  <div class="flex justify-center mt-2">
    <table class="table" style="margin-top: 80px;">
      <caption>USER PROFILE</caption>
      {% for key,value in dictprofile.items() %}
      <tr class="table__row">
        <td class="table__cell"> {{ key }} </td>
        <td class="table__cell"> {{ value }} </td>
      </tr>
      {% endfor %}
    </table>
  </div>
  <pre style="font-weight:bold;font-size:20px">To make any changes to your profile, please contact our s
</body>

</html>
```

App.py

```
@app.route('/profile')
def profile():
    username = userid
    dictprofile = {'Username': '', 'Email ID': '', 'Is Donor?': 'NO'}
    dictprofile['Username'] = username

    sql = "SELECT email FROM users WHERE username=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)
    dictprofile['Email ID'] = ibm_db.fetch_tuple(stmt)[0]

    sql = "SELECT * FROM donor WHERE username=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)
    gotprofile = ibm_db.fetch_tuple(stmt)
    if gotprofile:
        dictprofile['Is Donor?'] = 'YES'
        dictprofile['Full Name'] = gotprofile[1]
        dictprofile['Age'] = gotprofile[2]
        dictprofile['Gender'] = gotprofile[3]
        dictprofile['Phone Number'] = gotprofile[4]
        dictprofile['Address'] = gotprofile[5]
        dictprofile['City'] = gotprofile[6]
        dictprofile['State'] = gotprofile[7]
        dictprofile['Blood Group'] = gotprofile[8]
        dictprofile['Date of Positive Covid Test'] = gotprofile[9]

    sql = "SELECT email FROM users WHERE username=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)
    dictprofile['Email ID'] = ibm_db.fetch_tuple(stmt)[0]

    sql = "SELECT * FROM donor WHERE username=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)
    gotprofile = ibm_db.fetch_tuple(stmt)
    if gotprofile:
        dictprofile['Is Donor?'] = 'YES'
        dictprofile['Full Name'] = gotprofile[1]
        dictprofile['Age'] = gotprofile[2]
        dictprofile['Gender'] = gotprofile[3]
        dictprofile['Phone Number'] = gotprofile[4]
        dictprofile['Address'] = gotprofile[5]
        dictprofile['City'] = gotprofile[6]
        dictprofile['State'] = gotprofile[7]
        dictprofile['Blood Group'] = gotprofile[8]
        dictprofile['Date of Positive Covid Test'] = gotprofile[9]
        dictprofile['Date of Negative Covid Test'] = gotprofile[10]
    return render_template('userprofile.html', msg=userid, dictprofile = dictprofile)
```


8. TESTING

8.1 Test Cases

			Date	18-Nov-22									
			Team ID	PNT2022TMD20045									
			Project Name	Project - Plasma Donor Application									
			Maximum Marks	4 marks									
Test case ID	Feature Type	Component	Test Scenario	Pre-Requlite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
Registration	Functional	Home page	Verify user is able to register in the front page		1. Click the page and go 2. Register in the front page for donor as a user 3. Verify whether the user can register or not	Username : ket email:kethan@gmail.com pass: 12345	Username already exists	Working as expected	Fail		N		
Registration	Functional	Home page	Verify user is able to register in the front page		1. Click the page and go 2. Register in the front page for donor as a user 3. Verify whether the user can register or not	Username: gahabp@as email:gah@gmail.com pass: gaga	Register should display and the registration has been done	Working as expected	Pass		N		
LoginPage	Functional	Home Page	Verify user is able to see the Login/signup popup when user clicked login page		1. click go 2. Click on My Account dropdown button 3. Verify login/signup popup displayed or not		Login/signup popup should display	Working as expected	Pass		N		
LoginPage	UI	Home Page	Verify the UI elements in Login/signup popup		1. click go 2. Click on My Account dropdown button 3. Verify login/signup popup with below UI elements: a. Username text box b. password text box c. Login button d. New customer? Create account link		Application should show below UI elements: a. Username text box b. password text box c. Login button with orange color d. New customer? Create account link	Working as expected	Pass		N		
LoginPage	Functional	Home page	Verify user is able to log into application with valid credentials		1. click go 2. Click on My Account dropdown button 3. Enter valid username in Email text box 4. Enter valid password in password text box 5. Click on login button	Username: ket password: 12345	User should navigate to user account homepage	Working as expected	Pass		N		
LoginPage	Functional	Login page	Verify user is able to log into application with invalid credentials		1. click go 2. Click on My Account dropdown button 3. Enter invalid username/email in Email text box 4. Enter valid password in password text box 5. Click on login button	Username: ket password: abcde	Application should show "incorrect email or password" validation message	Working as expected	Fail		N		
Donor Registration	Functional	Registration page	Verify the donor can able to register and give the covid test within the correct date		1. Click on positive covid test 2. Date of negative covid test 3. Date of negative covid test	20/12/2022 20/12/2022 17/11/2022	Donor has to be registered before 14 days or they should not be able to register	Working as expected	Fail		N		
Donor Registration	Functional	Registration page	Verify the donor can able to register and give the covid test within the correct date		1. Donor has to do the registration before 14 days	4-11-2022	Donor has to be registered before 14 days and the registration has been done and it should be display	Working as expected	Pass		N		
Donor Registration	Functional	Registration page	Verify the donor has been already existed or not in DB donor		Type username and password	Username: ket	ket should be already existed	Working as expect	Fail		N		
Donor Registration	Functional	Registration page	Verify the donor has been already existed or not in DB donor		Type username and password	Username: priya	New user to registration should be done	Working as expect	Pass		N		
Donor Request	Functional	Request Page	The donor has been requested in the plasma donor		Type the patient name	patient: Cythella	This patient is already existed	Working as expect	Fail		N		
Donor Request	Functional	Request Page	The donor has been requested in the plasma donor		Type the patient name	patient: megan	The patient is new so request has been	Working as expect	Pass		N		
Past request	Functional	Past request page	Verify the past request of the donor has been already there		Type the username for request	username: ket	I listed the past request	Working as expect	Pass		N		
Past request	Functional	Past request page	Verify the past request of the donor has been already there		Type the username for request	username: priya	I miss past request	Working as expected	Fail		N		
Admin blood group request	Functional	Admin request page	Verify the user is not a donor		Type the request in the donor side	Username: priya	Its not a donor	Working as expected	Fail		N		
Admin blood group request	Functional	Admin request page	Verify the user blood group is found in Database		Type the request in the donor side	Username: ket	Its a donor	Working as expected	Pass		N		
Admin side: All request	In database, request has been there	Admin all request page	Verify the admin all request been displayed		Type the request in the donor side	Username: ket	Approved the request	Working as expected	Pass		N		
Admin side: All request	In database, request hasn't been there	Admin all request page	Verify the admin all request been displayed		Type the request in the donor side	Username: priya	Not approved the request	Working as expected	Fail		N		
Admin side: All donor	In database donor, no donor is there	Admin all Donor page	Verify the admin all donor has been displayed		Type the donor request in the donor side	Username: ket	Approved the request	Working as expected	Pass		N		
Admin side: All donor	In database donor, donor is there	Admin all Donor page	Verify the admin all donor has been displayed		Type the donor request in the donor side	Username: priya	Not approved the request	Working as expected	Fail		N		

8.2 User Acceptance Testing

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	0	0	0	0	0
Duplicate	0	0	0	0	0
External	1	0	0	1	1
Fixed	0	2	0	0	2
Not Reproduced	0	0	0	0	0
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	1	2	0	0	3

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Login	3	0	0	3
Registration	4	0	0	4
Dashboard	1	0	0	1
Plasma Addition	3	0	1	2
Rewards and Goals	4	0	0	4
Deleting a Request	2	0	0	2
Donor Registration	2	0	0	2

9. RESULTS

9.1 Performance Metrics

NFT-Risk assessment:

NFT - Risk Assessment									
S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Impact of Downtime	Load/Volumen Changes	Risk Score	Justification
1	Plasma Donor v1	New	High	No Changes	Moderate	Considerable time taken to insert into DB	No Changes	ORANGE	Adding this feature makes it coherent with features like adding group and hence, functional changes is high.
2	Plasma Donor v2	Existing	Moderate	No changes	Moderate	Faster DB operations	NO changes	ORANGE	Updated EDS, minor updates for better DB operations.

NFT - Detailed Test Plan:

NFT - Detailed Test Plan				
S.No	Project Overview	NFT Test approach	Assumptions/Dependencies/Risks	Approvals/SignOff
1	Plasma Donor v1	Load testing	Flask installed,app deployed with docker	
2	Plasma Donor v2	Stress testing	Flask installed,app deployed with docker	
3	Plasma Donor v3	Spike testing	Flask installed,app deployed with docker	
4	Plasma Donor v4	Reliability Testing	Flask installed,app deployed with docker	

End Of Test Report:

			End Of Test Report					
S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Recommendations	(Detected/Closed/Open)	Approvals/SignOff
1	Plasma Donor v1	Load testing	Failures	Number of failures spikes from 0.3 to 0.9 with increase in number of users from 70 to 100	Go	Provide checks from input value before reading from database	Bad request error on viewing approved requests when not a donor	
2	Plasma Donor v2	Stress testing	Response time	Increases from 3100 for one user to 1900ms for 20 users	Go			
3	Plasma Donor v3	Spike testing	Response time	Varies from minimum if 1367 ms and maximum of 25609ms for a specific feature of adding expenses	Go			
4	Plasma Donor v4	Reliability Testing	No of failures	41% failures with adding expense with spike in user data and incorrect input values, but maintained average of 13000ms response	Go	Provide checks from input value before inserting	Bad request error on viewing approved requests when not a donor	

10. ADVANTAGES & DISADVANTAGES

Advantages

1. The application makes the registration process of potential donors easier because of the app's user-friendly intuitive UI.
2. Blood banks and hospitals are able to access information of potential donors in a convenient manner.
3. Donor requests can be accepted/rejected at the discretion of the hospital/blood bank.
4. Provides email support for new users. Registered donors can be contacted for requesting plasma resources.
5. Database for donors and history of donation helps with tracking of these donors for future needs.

Disadvantages

1. Application cannot sort out platelet donors based on previous diseases.
2. Unable to handle spam accounts or ingenuine requests.
3. Automated email only works for new users. Regular request email to donors must be done manually.

11. CONCLUSION

A plasma donor application was developed which streamlines the process for potential donors to donate blood plasma wherever there is demand for such resources. This in turn aids in satisfying the demand for specific treatments that require blood plasma for the benefit of countless patients. This application was developed using IBM's cloud based tools such as DB2,etc. SendGrid is used for the automation of delivering mail to users. Docker and Kubernetes are used for deployment of the application.

12. FUTURE SCOPE

Future augmentations/scope on this application would include:

1. Application could be made resilient to spam and ingenuine requests.
2. Tracking of donors with respect to donor history could be automated.
3. Requests from potential donors with certain immunities to diseases could be forwarded to/suggested to clinics/hospitals that require plasma with those immunities.
4. Frequent contact and outreach towards donors could be automated.

13. APPENDIX

13.1 Source code

```
import os

from flask import Flask, render_template, url_for, redirect, session, request, flash
import re
from datetime import * import ibm_db
from flask_mail import Mail, Message

print("modules imported") app =
Flask(__name__)
print("Flask app created with __name__") app.config['SECRET_KEY']
= 'top-secret!' app.config['MAIL_SERVER'] = 'smtp.sendgrid.net'
app.config['MAIL_PORT'] = 587 app.config['MAIL_USE_TLS'] = True
app.config['MAIL_USERNAME'] = 'apikey'
app.config['MAIL_PASSWORD'] =

'SG.htIJZqmoTh23vNjENnDKSg.7RK2sw7JEkP7KuFq4qpKc98OKRyvCNrG01AwYND1FY'
app.config['MAIL_DEFAULT_SENDER'] = 'abinash.drl169@gmail.com' mail = Mail(app)
print("setup default configs")

conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=6667d8e9-9d4d-4ccb-ba32-
21da3bb5aafc.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30376;
```

```
SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=kkt29633;
PWD=w2Nriolg1HfcHAWB", ", ") print("db conn
made")
```

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

```
#-----NORMAL USER ROUTES-----
```

```
@app.route('/login', methods=['GET', 'POST']) def login():
    global userid
    msg = ""
    if request.method == "POST":
        username = request.form["username"]
        password = request.form["password"]
        sql = "SELECT * FROM users WHERE username=? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)  ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['Loggedin'] = True
            session['id'] = account['USERNAME']
            userid = account['USERNAME']
            session['username'] = account['USERNAME']
            return render_template('dashboard.html', msg=username)
        else:
            msg = "Invalid login credentials!"
    return render_template('login.html', msg=msg)
```

```

@app.route('/register', methods=['GET', 'POST']) def register():
    msg = "Please fill out the form."
    if request.method == 'POST':
        username = request.form['username']    email =
request.form['email']
        password = request.form['password']
        sql = "SELECT * FROM users WHERE username=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = "Username already exists!" elif not
            re.match(r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z0-9]+$', email):
                msg = "Invalid email address!" elif not re.match(r'^[A-Za-z0-9]+$', username):
                    msg = "Username must contain only
                    letters and numbers!"
            else:
                insertsql = "INSERT INTO users VALUES(?,?,?)"
                prepstmt = ibm_db.prepare(conn, insertsql)
                ibm_db.bind_param(prepstmt, 1, username)    ibm_db.bind_param(prepstmt, 2, email)
                ibm_db.bind_param(prepstmt, 3, password)
                ibm_db.execute(prepstmt)

                mailmsg = Message('Registration Verification', recipients=[email])    mailmsg.body =
('Congratulations! Welcome new user!')    mailmsg.html = ('<h1>Plasma App Registration
Verification</h1>'    '<p>Congratulations! Welcome new user</p>'    '<br><br>'
'<b>Plasma Donor App Team</b>!</p>')

                mail.send(mailmsg)

```

```

    print("mail has been sent out")

    msg = "You have successfully created an account!"
    flash(msg)
    return redirect(url_for('index'))
return render_template('register.html', msg=msg)

@app.route('/dashboard') def
dashboard():
    return render_template('dashboard.html', msg=userid)

@app.route('/newdonor') def
newdonor(): username = userid

    sql = "SELECT username FROM donor WHERE username=?" stmt =
    ibm_db.prepare(conn, sql) ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    print("reached account", account)
    if account:
        msg = "You have already registered as a donor! Donor registration is only 1 time!"
        flash(msg)
        return redirect(url_for('dashboard'))
    return render_template('newdonor.html')

@app.route('/newrequest') def
newrequest():
    return render_template('newrequest.html')

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

```

```

username = userid
name = request.form['name']
age = request.form['age']
gender = request.form['gender']
phno = request.form['phone']
addr = request.form['address']
city = request.form['city'] state =
    request.form['state'] bgp =
    request.form['bloodgp'] dop =
    request.form['dop']

don = request.form['don']
if int(age)<18:
    msg="You have to be an adult to donate plasma!"
elif not validate(dop, don):
    msg="Your Covid Negative test should have been received atleast 14 days ago!"
else:
    insertsql = "INSERT INTO donor VALUES(?,?,?,?,?,?,?,?,?,?)"
    prepstmt = ibm_db.prepare(conn, insertsql)
    ibm_db.bind_param(prepstmt, 1, username)
    ibm_db.bind_param(prepstmt, 2, name)
    ibm_db.bind_param(prepstmt, 3, age)
    ibm_db.bind_param(prepstmt, 4, gender)
    ibm_db.bind_param(prepstmt, 5, phno)
    ibm_db.bind_param(prepstmt, 6, addr)
    ibm_db.bind_param(prepstmt, 7, city)
    ibm_db.bind_param(prepstmt, 8, state)
    ibm_db.bind_param(prepstmt, 9, bgp)
    ibm_db.bind_param(prepstmt, 10, dop)
    ibm_db.bind_param(prepstmt, 11, don)
    print("prep insert - "+dop+"-"+don+"-"+addr)
    ibm_db.execute(prepstmt)

```



```

        msg = "You have successfully applied as a donor."        print("executed insert")

flash(msg)

return redirect(url_for('dashboard'))

def validateDate(dop, don): d,m,y = [int(x) for x in dop.split('-')]
    d1 = date(d,m,y)

    d,m,y = [int(x) for x in don.split('-')] d2 = date(d,m,y)
    if d2>d1 and date.today()>d2+timedelta(days=14):
        return True
    return False

@app.route('/regrequest', methods=['GET', 'POST']) def regrequest():
    msg = ""
    if request.method == "POST":
        username = userid
        pname = request.form['pname']
        print(userid, "\t", session['id'], "pname = ", pname)
        sql = "SELECT * FROM requests WHERE username=? AND pname=?" #patient's name = pname
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, pname)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print("reached account here for pname", account)
        if account:
            msg = "Register for a patient only once!"
            flash(msg)
            return redirect(url_for('dashboard'))

    phno = request.form['phone']

```

```
paddr = request.form['paddress'] city =  
    request.form['city'] state =  
    request.form['state'] bgp =  
    request.form['bloodgp']
```

```
insertsql = "INSERT INTO requests VALUES(?,?,?,?,?,?)"
```

```
prepstmt = ibm_db.prepare(conn, insertsql)
```

```
ibm_db.bind_param(prepstmt, 1, username)
```

```
ibm_db.bind_param(prepstmt, 2, pname)    ibm_db.bind_param(prepstmt, 3, phno)
```

```
ibm_db.bind_param(prepstmt, 4, paddr)    ibm_db.bind_param(prepstmt, 5, city)
```

```
ibm_db.bind_param(prepstmt, 6, state)
```

```
ibm_db.bind_param(prepstmt, 7, bgp)
```

```
print("prep insert - "+username+"--"+pname+"--"+bgp)
```

```
ibm_db.execute(prepstmt)
```

```
msg = "You have successfully requested plasma."    print("executed insert")
```

```
flash(msg)
```

```
return redirect(url_for('dashboard'))
```

```
@app.route('/pastrequests') def
```

```
pastrequests():
```

```
    username = userid
```

```
    flag = 0
```

```
    data = []
```

```
    sql = "SELECT pname,phone_number,paddress,city,state,blood_gp FROM requests WHERE  
        username=?"
```

```
    stmt = ibm_db.prepare(conn, sql)
```

```
    ibm_db.bind_param(stmt, 1, username)
```

```
    ibm_db.execute(stmt)
```

```
    request = ibm_db.fetch_tuple(stmt) while request != False:
```

```
        flag = 1 #atleast 1 match is found
```

```
        data.append(request)
```

```

        request = ibm_db.fetch_tuple(stmt) if not flag:
        msg = "You have made 0 requests!"
        flash(msg)

    return redirect(url_for('dashboard')) else:
    print("No of requests = ", len(data))
    data = tuple(data)
    headings = ("Patient's Name", "Emergency Contact", "Patient's Address", "City", "State", "Blood
    Group Requested", "Options")
    return render_template('userrequests.html', msg=userid, data=data, headings=headings)

```

```

@app.route('/adminrequests') def

```

```

adminrequests():

```

```

    username = userid

```

```

    sql = "SELECT blood_gp FROM donor WHERE username=?"

```

```

    stmt = ibm_db.prepare(conn, sql)

```

```

    ibm_db.bind_param(stmt, 1, username)

```

```

    ibm_db.execute(stmt)

```

```

    mybgp = ibm_db.fetch_tuple(stmt)

```

```

    print("My bgp is: ", mybgp)

```

```

    if not mybgp:

```

```

        msg = "You have not registered as a donor yet! Please register first to view other's requests!"

```

```

        flash(msg)

```

```

        return redirect(url_for('dashboard'))

```

```

    flag = 0 data =

```

```

    []

```

```

sql = "SELECT pname,phone_number,state,blood_gp FROM approved_requests WHERE blood_gp=?"

```

```

stmt = ibm_db.prepare(conn, sql)

```

```

ibm_db.bind_param(stmt, 1, mybgp[0])

```

```

    ibm_db.execute(stmt)

```

```

    request = ibm_db.fetch_tuple(stmt)

```

```

    while request != False:

```

```

    flag = 1 #atleast 1 match is found
    data.append(request)
    request = ibm_db.fetch_tuple(stmt)
if not flag:
    print("My bgp is: ", mybgp)    msg = "No requests for your blood group were found!"
    flash(msg)
    return redirect(url_for('dashboard'))
else:
    print("No of mybgp requests = ", len(data))
    data = tuple(data)
    headings = ("Patient's Name", "Emergency Contact", "State", "Blood Group Requested")
    return render_template('bgprequests.html', msg=userid, data=data, headings=headings)

@app.route('/profile') def
profile():
    username = userid
    dictprofile = {'Username': '', 'Email ID': '', 'Is Donor?': 'NO'}
    dictprofile['Username'] = username

    sql = "SELECT email FROM users WHERE username=?" stmt =
    ibm_db.prepare(conn, sql) ibm_db.bind_param(stmt, 1, username)
    ibm_db.execute(stmt)

    dictprofile['Email ID'] = ibm_db.fetch_tuple(stmt)[0]

    sql = "SELECT * FROM donor WHERE username=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)    ibm_db.execute(stmt)
    gotprofile = ibm_db.fetch_tuple(stmt)
    if gotprofile:
        dictprofile['Is Donor?'] = 'YES'
        dictprofile['Full Name'] = gotprofile[1]

```

```

dictprofile['Age'] = gotprofile[2]
dictprofile['Gender'] = gotprofile[3]
dictprofile['Phone Number'] = gotprofile[4]
dictprofile['Address'] = gotprofile[5]
dictprofile['City'] = gotprofile[6]
dictprofile['State'] = gotprofile[7]
dictprofile['Blood Group'] = gotprofile[8]
dictprofile['Date of Positive Covid Test'] = gotprofile[9]    dictprofile['Date of Negative Covid Test'] =
gotprofile[10]
return render_template('userprofile.html', msg=userid, dictprofile = dictprofile)

```

#-----ADMIN ROUTES-----

```

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

msg = ""
if request.method == "POST":
    username = request.form["username"]
    password = request.form["password"]
    sql = "SELECT * FROM admins WHERE username=? AND password=?"
    stmt = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(stmt, 1, username)    ibm_db.bind_param(stmt, 2, password)
    ibm_db.execute(stmt)
    account = ibm_db.fetch_assoc(stmt)
    print(account)
    if account:
        session['Loggedin'] = True
        session['id'] = account['USERNAME']
        adminid = account['USERNAME']

```

```

        session['username'] = account['USERNAME']

        return redirect(url_for('admindashboard'))

    else:

        msg = "Invalid admin credentials!"

    return render_template('adminlogin.html', msg=msg)

@app.route('/admindashboard') def
admindashboard():

    sql = "SELECT COUNT(*) FROM requests"
    stmt = ibm_db.exec_immediate(conn, sql)
    numreqs = ibm_db.fetch_tuple(stmt)[0]

    sql = "SELECT COUNT(*) FROM donor"
    stmt = ibm_db.exec_immediate(conn, sql)
    numdonors = ibm_db.fetch_tuple(stmt)[0]

    sql = "SELECT COUNT(*) FROM approved_requests" stmt =
    ibm_db.exec_immediate(conn, sql) numappreqs =
    ibm_db.fetch_tuple(stmt)[0]

    if not numreqs: numreqs = 0
    if not numappreqs: numappreqs = 0
    if not numdonors: numdonors = 0

    print(numreqs, numdonors, numappreqs)
    return render_template('admindashboard.html', msg=adminid, numreqs=numreqs,
        numdonors=numdonors, numappreqs=numappreqs)

@app.route('/allrequests') def
allrequests():

    data = []

```

```

flag = 0
sql = "SELECT * FROM requests"
stmt = ibm_db.exec_immediate(conn, sql)
request = ibm_db.fetch_tuple(stmt)
while request != False:
    flag = 1 #atleast 1 match is found
    data.append(request)
    request = ibm_db.fetch_tuple(stmt)
if not flag:
    msg = "There are 0 requests!"
    flash(msg)
    return redirect(url_for('admindashboard'))
else:
    print("No of requests = ", len(data))
    data = tuple(data)
    headings = ("Username", "Patient's Name", "Emergency Contact", "Patient's Address", "City", "State",
    "Blood Group Requested", "Options")
    return render_template('viewallreqs.html', msg=adminid, data=data, headings=headings)

@app.route('/approvereq', methods=['POST', 'GET']) def approvereq():
    if request.method == "POST":
        bgp = request.form["bgp"]
        print("Request approved for bgp = ", bgp)

        pname = request.form['pname']
        phno = request.form['phone']
        state = request.form['state']

        sql = "SELECT * FROM approved_requests WHERE pname=? AND blood_gp=? AND
        phone_number=?"
        stmt = ibm_db.prepare(conn, sql)

```

```

ibm_db.bind_param(stmt, 1, pname)
ibm_db.bind_param(stmt, 2, bgp)
ibm_db.bind_param(stmt, 3, phno)
ibm_db.execute(stmt)
appreq = ibm_db.fetch_assoc(stmt)
print("reached appreq", appreq)
if appreq:
    msg = "Request already approved! Approve only 1 time!"
    flash(msg)
    return redirect(url_for('admindashboard'))

insertsql = "INSERT INTO approved_requests VALUES(?,?,?,?)"
prepstmt = ibm_db.prepare(conn, insertsql)
    ibm_db.bind_param(prepstmt, 1, pname) ibm_db.bind_param(prepstmt, 2, phno)
    ibm_db.bind_param(prepstmt, 3, state)
    ibm_db.bind_param(prepstmt, 4, bgp)
ibm_db.execute(prepstmt)

msg = "You have successfully approved a request."    print("executed insert")
flash(msg)
return redirect(url_for('admindashboard'))

@app.route('/alldonors') def
alldonors():
    data = []
    flag = 0
    sql = "SELECT * FROM donor"
    stmt = ibm_db.exec_immediate(conn, sql)
    donor = ibm_db.fetch_tuple(stmt)
    while donor != False:
        flag = 1 #atleast 1 match is found
        data.append(donor)

```



```

        donor = ibm_db.fetch_tuple(stmt)
    if not flag:
        msg = "There are 0 donors!"
        flash(msg)
        return redirect(url_for('admindashboard'))
    else:
        print("No of donors = ", len(data))        data = tuple(data)
                                                headings = ("Username", "Full Name", "Age", "Gender", "Phone
                                                Number", "Donor's Address", "City", "State", "Blood Group", "+ve Covid
                                                Test", "-ve Covid Test")
        return render_template('viewalldonors.html', msg=adminid, data=data, headings=headings)

```

#-----GENERAL LOGOUT ROUTES-----

```

@app.route('/deletereq', methods=['GET', 'POST']) def deletereq():
    if request.method == 'POST':
        username = request.form['username']
        pname = request.form['pname']
        delsql = "DELETE FROM requests WHERE username=? AND pname=?"
        prepstmt = ibm_db.prepare(conn, delsql)
        ibm_db.bind_param(prepstmt, 1, username)
        ibm_db.bind_param(prepstmt, 2, pname)
        ibm_db.execute(prepstmt)
        msg = "You have successfully deleted the request."        print("executed delete: ", pname)
        flash(msg)
        if 'adminid' in globals():
            return redirect(url_for('admindashboard'))
        elif 'userid' in globals():
            return redirect(url_for('dashboard'))

```

```

@app.route('/logout') def logout():
    session.pop('Loggedin', None) session.pop('id',
        None) session.pop('username', None) if
        'userid' in globals():

        global userid del
        userid

        elif 'adminid' in globals():
            global adminid
            del adminid
    return render_template('index.html')

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

```

Admindashboard.html

```

<!DOCTYPE html>
<html>

<head>
    <title>ADMIN DASHBOARD</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/dashboards.css')
        }}">

    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/tables.css') }}">

        <link rel="stylesheet"
            href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css">
</head>

<body>

```

```

<!-- navbar -->

<ul>

    <li><a style="color: white" href="logout">LOGOUT</a></li> </ul>

<div class="class-container" id="container">

    <div class="header">

        <h2 class="header__title">        Welcome Admin
{{msg}}!

        </h2>

    </div>

</div>

<div class="flasher">

    {% with messages = get_flashed_messages() %}

    {% if messages %}

    {% for message in messages %}

    <p>{{ message }}</p>

    {% endfor %}

    {% endif %}

    {% endwith %}

</div>

<div class="flex justify-center mt-2">

    <table class="table">

        <caption>KEY SYSTEM STATS</caption>

        <tr class="table__row">

            <td class="table__cell">Number of Donors</td>

            <td class="table__cell">{{ numdonors }}</td>

        </tr>

        <tr class="table__row">

            <td class="table__cell">Number of Requests</td>

            <td class="table__cell">{{ numreqs }}</td>

```

```

        </tr>

        <tr class="table__row">

            <td class="table__cell">Number of Approved Requests</td>

            <td class="table__cell">{{ numappreqs }}</td>        </tr>

    </table>

</div>

<div class="flex justify-center mt-2">

    <div class="block p-6 rounded-lg shadow-lg bg-black max-w-sm">

        <a href="allrequests">

            <button type="button"

                class="text-white bg-gradient-to-r from-blue-500 viablue-600 to-blue-700 hover:bg-gradient-
to-br focus:ring-4 focus:outline-none focus:ring-blue-300 dark:focus:ring-blue-800 shadowlg shadow-
blue-500/50 dark:shadow-lg dark:shadow-blue-800/80 fontmedium rounded-lg text-sm px-5 py-2.5
text-center mr-2 mb-2 ">VIEW

                ALL REQUESTS FOR PLASMA</button>

            </a>

        </div>

    </div>

<div class="flex justify-center mt-10">

    <div class="block p-6 rounded-lg shadow-lg bg-black max-w-sm">

        <a href="alldonors">

            <button type="button"

                class="text-white bg-gradient-to-r from-blue-500 viablue-600 to-blue-700 hover:bg-gradient-
to-br focus:ring-4 focus:outline-none focus:ring-blue-300 dark:focus:ring-blue-800 shadowlg shadow-
blue-500/50 dark:shadow-lg dark:shadow-blue-800/80 fontmedium rounded-lg text-sm px-5 py-2.5
text-center mr-2 mb-2 ">VIEW

                ALL DONORS ON SYSTEM</button>

            </a>

        </div>

    </div>

</body>

```

</html>

Adminlogin.html

<!DOCTYPE html>

<html>

<head>

<title>ADMINISTRATOR LOGIN</title>

<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/logins.css') }}">

<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">

</head>

<body>

<div class="flex items-center justify-center min-h-screen "> <div class="px-8 py-6 mt-4
text-left bg-black shadow-lg">

<h3 class="text-2xl font-bold text-center" style="color: white;">Admin Login</h3>

<form action="/adminlogin" method="post">

<div class="flasher">{{ msg }}</div>

<div class="mt- -20">

<div>

<label class="block" style="color: white;" for="username">Username<label>

<input type="text" id="username" name="username" placeholder="Username"

class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1
focus:ring-blue-600"

style="color: black;">

</div>

<div class="mt-4">

<label class="block" style="color: white;" for="password">Password<label>

<input type="password" id="password" name="password" placeholder="Password"

```

        class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-
blue-600"
        style="color: black;">
    </div>
    <div class="flex items-baseline justify-between">
        <button class="px-6 py-2 mt-4 text-white bg-blue-600 roundedlg hover:bg-blue-
900">Login</button>
    </div>
</div>
</form>
<p class="text-sm font-light text-gray-500 dark:text-gray-400 my-3">
    Click here to login as normal user ->
    <a href="login" class="font-medium text-primary-600 hover:underline dark:text-primary-
500">Login</a>
</p>
</body>

</html>

```

Bgprerequisite.html

```

!DOCTYPE html>
<html>

<head>
    <title>REQUESTS TO ME</title>

    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/tables.css') }}">

    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/results.css') }}">

    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">

```

```
</head>
```

```
<body>
```

```
<!-- navbar starts -->
```

```
<ul>
```

```
  <div style="text-align: right;">
```

```
    <li><a type="button" href="dashboard">DASHBOARD</a></li>
```

```
    <li><a type="button" href="profile">PROFILE</a></li>
```

```
    <li><a type="button" href="logout">LOGOUT</a></li>
```

```
  </div>
```

```
</ul>
```

```
<div class="class-container" id="container">
```

```
<h4 style="text-align: center; color:white; font-size:25px; marginbottom: 100px;">The following  
requests are open for your blood group! Do donate!</h4>
```

```
</div>
```

```
<!-- navbar ends -->
```

```
<div class="flex justify-center mt-2">
```

```
  <table class="table">
```

```
    <tr class="table__header">
```

```
      {% for header in headings %}
```

```
        <th class="table__cell">{{ header }}</th>
```

```
      {% endfor %}
```

```
    </tr>
```

```
    {% for row in data %}
```

```
      <tr class="table__row">
```

```
        {% for cell in row %}
```

```
          <td class="table__cell">{{ cell }}</td>
```

```
        {% endfor %}
```

```
      </tr>
```

```
    {% endfor %}
```

```
</table>

</div>

<pre style="font-weight:bold;font-size:20px">To make a direct plasma donation, please contact the
    emergency contact provided.</pre>

</body>

</html>

Dashboard.html

<!DOCTYPE html>

<html>

<head>
    <title>USER DASHBOARD</title>

    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/dashboards.css')
        }}">

    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/tables.css') }}">

    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
</head>

<body>
    <!-- navbar -->

    <ul>
        <li><a style="color: white" href="logout">LOGOUT</a></li>
        <li><a style="color: white" href="profile">PROFILE</a></li>
    </ul>

    <div class="class-container" id="container">
        <div class="header">
            <h2 class="header__title">
```



```

        Welcome {{msg}}!

    </h2>

</div>

</div>

<!-- navbar ends -->


<div class="flasher">
{% with messages = get_flashed_messages() %}
    {% if messages %}
        {% for message in messages %}
            <p>{{ message }}</p>
        {% endfor %}
    {% endif %}
{% endwith %}
</div>


<div class="flex justify-center mt-2">
<div class="block p-6 rounded-lg shadow-lg bg-black max-w-sm">
    <a href="newdonor">
        <button type="button"
            class="text-white bg-gradient-to-r from-blue-500 viablue-600 to-blue-700   hover:bg-
            gradient-to-br   focus:ring-4 focus:outline-none focus:ring-blue-300 dark:focus:ring-blue-800
            shadowlg shadow-blue-500/50 dark:shadow-lg dark:shadow-blue-800/80 fontmedium rounded-lg text-
            sm px-5 py-2.5 text-center mr-2 mb-2 ">REGISTER AS DONOR</button>
    </a>
</div>
</div>

<div class="flex justify-center mt-10">
    <div class="block p-6 rounded-lg shadow-lg bg-black max-w-sm">
        <a href="newrequest">
            <button type="button"

```

```
class="text-white bg-gradient-to-r from-blue-500 viablue-600 to-blue-700 hover:bg-
gradient-to-br focus:ring-4 focus:outline-none focus:ring-blue-300 dark:focus:ring-blue-800
shadowlg shadow-blue-500/50 dark:shadow-lg dark:shadow-blue-800/80 fontmedium rounded-lg text-
sm px-5 py-2.5 text-center mr-2 mb-2 ">REQUEST PLASMA</button>
```

```
</a>
```

```
</div>
```

```
</div>
```

```
<div class="flex justify-center mt-10">
```

```
<div class="block p-6 rounded-lg shadow-lg bg-black max-w-sm">
```

```
<a href="pastrequests">
```

```
<button type="button"
```

```
class="text-white bg-gradient-to-r from-blue-500 viablue-600 to-blue-700 hover:bg-gradient-
to-br focus:ring-4 focus:outline-none focus:ring-blue-300 dark:focus:ring-blue-800 shadowlg shadow-
blue-500/50 dark:shadow-lg dark:shadow-blue-800/80 fontmedium rounded-lg text-sm px-5 py-2.5
text-center mr-2 mb-2 ">PAST REQUESTS</button>
```

```
</a>
```

```
</div>
```

```
</div>
```

```
<div class="flex justify-center mt-10">
```

```
<div class="block p-6 rounded-lg shadow-lg bg-black max-w-sm">
```

```
<a href="adminrequests">
```

```
<button type="button"
```

```
class="text-white bg-gradient-to-r from-blue-500 viablue-600 to-blue-700 hover:bg-
gradient-to-br focus:ring-4 focus:outline-none focus:ring-blue-300 dark:focus:ring-blue-800
shadowlg shadow-blue-500/50 dark:shadow-lg dark:shadow-blue-800/80 fontmedium rounded-lg text-
sm px-5 py-2.5 text-center mr-2 mb-2 ">ADMIN REQUESTS</button>
```

```
</a>
```

```
</div>
```

```
</div>
```

```
<!-- <section style="text-align: center;"> <div><a type="button" href="newdonor"
style="font-size:30px">REGISTER AS DONOR</a></div> <hr> <div><a type="button"
```

```
href="newrequest" style="font-size:30px">REQUEST PLASMA</a></div>    <hr>    <div><a
type="button" href="pastrequests" style="font-size:30px">PAST REQUESTS</a></div>    <hr>

    <div><a type="button" href="adminrequests" style="font-size:30px">ADMIN REQUESTS
    TO ME</a></div> </section> -->
```

```
</body>
```

```
</html>
```

Index.html

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<title>PLASMA DONOR APPLICATION</title>
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/index.css') }}">
```

```
<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
```

```
</head>
```

```
<body> <ul>
```

```
<li><a href="login">LOGIN</a></li>
```

```
<li><a href="register">REGISTER</a></li>
```

```
</ul>
```

```
<!-- navbar ends -->
```

```
<div class="class-container" id="container">
```

```
<div class="header">
```

`<h2 class="header__title"> PLASMA`

`DONOR APPLICATION`

`</h2>`

`<h3 class="header__subtitle">`

`Welcome to the Application`

`<i class="fa fa-hand-o-down" style="font-size: 25px;"></i>`

`</h3>`

`</div>`

`</div>`

`<div class="flasher">`

`{% with messages = get_flashed_messages() %}`

`{% if messages %}`

`{% for message in messages %}`

`<p>{{ message }}</p>`

`{% endfor %}`

`{% endif %}`

`{% endwith %}`

`</div>`

`<!-- what we focus on -->`

`<div class="bdy">`

``

`<p class="para">`

`</br>`

`</br>`

The main objective is to create an easy-to-use application that can be used by donors to donate their blood to blood banks. Hospitals in need for blood plasma can request blood from blood banks. This application should have a wider reach to appeal to more potential blood donors.

</br>

</br>

Vision: To donate plasma in a hassle free condition by using our app.

</p> </div>

<!-- focus section ends -->

<!-- footer starts -->

<div class="footer-basic">

<footer>

<ul class="list-inline">

<li class="list-inline-item">

Email Us

<p class="copyright">SSNCE 2022 - PNT2022TMID53045</p>

</footer>

</div>

</body>

</html>

Login.html

<!DOCTYPE html>

<html>

<head>

<title>USER LOGIN</title>

<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/logins.css') }}">

<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">

</head>


```

        <a href="register" class="font-medium text-primary-600 hover:underline dark:text-primary-500">Sign up</a>
    </p>
    <p class="text-sm font-light text-gray-500 dark:text-gray-400">
        Admin Login <a href="adminlogin"
            class="font-medium text-primary-600 hover:underline dark:textprimary-500">Admin Login</a>
    </p> </div>
</div>
</body>

</html>

```

Login.html

```

<!DOCTYPE html>
<html>

<head>
    <title>USER LOGIN</title>

    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/logins.css') }}">
    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
</head>

<body>
    <div class="flex items-center justify-center min-h-screen ">    <div class="px-8 py-6 mt-4
text-left bg-black shadow-lg">
        <h3 class="text-2xl font-bold text-center" style="color: white;">Login to your account</h3>
        <form class="regforms" action="/login" method="post">
            <div class="flasher">{{ msg }}</div><br><br>
            <div class="mt- -20">
                <div>

```

```

    <label class="block" style="color: white;" for="username">Username</label>
    <input type="text" id="username" name="username" placeholder="Username"
      class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-
blue-600"
      style="color: black;">
  </div>
  <div class="mt-4">
    <label class="block" style="color: white;" for="password">Password</label>
    <input type="password" id="password" name="password" placeholder="Password"
      class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-blue-
600"
      style="color: black;">
  </div>
  <div class="flex items-baseline justify-between">
    <button class="px-6 py-2 mt-4 text-white bg-blue-600 rounded-lg hover:bg-blue-
900">Login</button>
  </div>
</div>
</div>
</form>
<p class="text-sm font-light text-gray-500 dark:text-gray-400 my-3">
  Don't have an account yet?
  <a href="register" class="font-medium text-primary-600 hover:underline dark:text-primary-
500">Sign up</a>
</p>
<p class="text-sm font-light text-gray-500 dark:text-gray-400">
  Admin Login <a href="adminlogin"
    class="font-medium text-primary-600 hover:underline dark:textprimary-500">Admin Login</a>
</p> </div>
</div>
</body>

</html>

```


Newdonor.html

```
<!DOCTYPE html>

<html>

<head>
    <title>DONOR REGISTRATION</title>

    <link type="text/css" rel="stylesheet" href="{{ url_for('static',
        filename='/stylesheets/dashboards.css') }}">

    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/forms.css') }}">

    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
</head>

<body>
    <nav>
        <div style="text-align: right;">
            <ul>
                <li><a style="color: white" href="profile">PROFILE</a></li>
                <li><a style="color: white" href="dashboard">DASHBOARD</a></li>
                <li><a type="button" href="logout">LOGOUT</a></li>
            </ul>
        </div>
    </nav>
    <!-- navbar ends -->

    <!-- Donor Registration form -->
    <div>
        <form action="/regdonor" method="post">
```

<h2 style="text-align: center; font-size: larger;"> NEW DONOR REGISTRATION </h2>

<fieldset>

<div style="margin-left:25%">

<h3 class="mt-5">Full Name:
<input type="text" name="name" placeholder="Enter Full Name" required

style="color: black; font-weight: bold;" />

</h3></br>

<h3>Age:
<input type="text" name="age" placeholder="Enter Age" pattern="[0-9]{1,3}"

required

style="color: black; font-weight: bold;" />

</h3></br>

<h3>Gender:
<select name="gender" required style="color: black;">

<option value="select" selected>Select gender</option>

<option value="male">Male</option>

<option value="female">Female</option>

<option value="other">Other</option>

</select>

</h3></br>

<h3>Mobile Number:
<input type="text" name="phone" placeholder="Enter Mobile Number" pattern="[0-9]{10}"

required style="color: black; font-weight: bold;" />

</h3></br>

<h3>Address:
<textarea name="address" rows="3" cols="50" placeholder="Enter Address"

required

style="color: black; font-weight: bold;"></textarea>

</h3></br>

<h3>City:
<input type="city" name="city" placeholder="Enter City" required

style="color: black; font-weight: bold;" />

</h3></br>

<h3>State:
<input type="state" name="state" placeholder="Enter State" required

style="color: black; font-weight: bold;" />

</h3></br>

```

<h3>Blood group:</h3><select name="bloodgp" required style="color: black;">
  <option value="select" selected>Select your blood group</option>
  <option value="O+">O+</option>
  <option value="A+">A+</option>
  <option value="B+">B+</option>
  <option value="AB+">AB+</option>
  <option value="O-">O-</option>
  <option value="A-">A-</option>
  <option value="B-">B-</option>
  <option value="AB-">AB-</option>
</select>
</h3></br>
<h3>
  <label for="dop">Date of Positive Covid Test: </label></br>
  <input type="date" name="dop" required style="color: black; font-weight: bold;" /></br></br>
</h3>
<h3>
  <label for="don">Date of Negative Covid Test: </label></br>
  <input type="date" name="don" required style="color: black; font-weight: bold;" /></br></br>
</h3>
  <button class="button" type="submit"> Register As Donor </button>
</div>
</fieldset>
</form>
</div>
</body>

</html>

```

Newrequest.html

```

<!DOCTYPE html>
<html>

```

```
<head>

<title>REQUEST PLASMA</title>

<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/dashboards.css')
    }}">

<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/forms.css') }}">

<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
</head>

<body>

<nav>

<div style="text-align: right;">

<ul>

<li><a style="color: white" href="profile">PROFILE</a></li>

<li><a style="color: white" href="dashboard">DASHBOARD</a></li>

<li><a type="button" href="logout">LOGOUT</a></li>

</ul>

</div>

</nav>

<!-- navbar ends -->

<!-- Request Application form -->

<div>

<form action="/regrequest" method="post">

<h2 style="text-align: center; font-size: larger;">NEW PLASMA REQUEST </h2>

<fieldset>

<div style="margin-left:25%">

<h3 class="mt-5"><label for="pname">Patient's Name:
```

</label></h3>

<input type="text" name="pname" placeholder="Enter Full Name" style="color: black; font-weight: bold;" required /></br></br>

<h3>

<label for="phone">Emergency Contact Number: </label></br>

<input type="text" name="phone" placeholder="Enter Mobile Number" pattern="[0-9]{10}" required style="color: black; font-weight: bold;" /></br></br>

</h3>

<h3>

<label for="paddress">Patient's Address: </label></br>

<textarea name="paddress" rows="3" cols="50" placeholder="Enter Address" required style="color: black; font-weight: bold;"></textarea></br></br>

</h3>

<h3>

<label for="city">Patient's City: </label></br>

<input type="city" name="city" placeholder="Enter City" required style="color: black; font-weight: bold;" /></br></br>

</h3>

<h3>

<label for="state">Patient's State: </label></br>

<input type="state" name="state" placeholder="Enter State" required style="color: black; font-weight: bold;" /></br></br>

</h3>

<h3>

<label for="bloodgp">Blood Group: </label></br>

<select name="bloodgp" style="color: black;" required>

<option value="select" style="color: black;" selected>Select blood group required

</option>

<option value="O+" style="color: black;">O+</option>

<option value="A+" style="color: black;">A+</option>

```

        <option value="B+" style="color: black;">B+</option>
        <option value="AB+" style="color: black;">AB+</option>
        <option value="O-" style="color: black;">O-</option>
        <option value="A-" style="color: black;">A-</option>
        <option value="B-" style="color: black;">B-</option>
        <option value="AB-" style="color: black;">AB-</option>
    </select>

    </br></br>

</h3>

<button class="button" type="submit"> Apply for Plasma </button>

</div>

</fieldset>

</form>

</div>

</body>

</html>

```

Register.html

```

<!DOCTYPE html>

<html>

<head>
    <title>REGISTER</title>
    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/logins.css') }}">
    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
</head>

<body>
    <div class="flex items-center justify-center min-h-screen ">    <div class="px-8 py-6 mt-4
text-left bg-black shadow-lg">

```

```

<h3 class="text-2xl font-bold text-center" style="color: white;">Login to your account</h3>
<form class="regforms" action="/register" method="post">
  <div class="flasher">{{ msg }}</div></br></br>
  <div class="mt- -20">
    <div>
      <label class="block" style="color: white;" for="username">Username</label>
      <input type="text" id="username" name="username" placeholder="Enter New Username"
        class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-
blue-600"
        style="color: black;">
    </div>
    <div>
      <label class="block" style="color: white;" for="email">Email Id</label>
      <input type="text" id="email" name="email" placeholder="Enter Your Email Id"
        class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-
blue-600"
        style="color: black;">
    </div>
    <div class="mt-4">
      <label class="block" style="color: white;" for="password">Password</label>
      <input type="password" id="password" name="password" placeholder="Password"
        class="w-full px-4 py-2 mt-2 border rounded-md focus:outline-none focus:ring-1 focus:ring-
blue-600"
        style="color: black;">
    </div>
    <div class="flex items-baseline justify-between">
      <button class="px-6 py-2 mt-4 text-white bg-blue-600 roundedlg hover:bg-blue-
900">Register</button>
    </div>
  </div>
</form>

```

```
<p class="text-sm font-light text-gray-500 dark:text-gray-400 my-3">
```

Already have an account?

```
<a href="/login" class="font-medium text-primary-600 hover:underline dark:text-primary-500">Login</a>
```

```
</p> </div>
```

```
</div>
```

```
</body>
```

```
</html>
```

Userprofile.html

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<title>CURRENT USER'S PROFILE</title>
```

```
<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/tables.css') }}">
```

```
<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/results.css') }}">
```

```
<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
```

```
</head>
```

```
<body>
```

```
<!-- navbar -->
```

```
<nav>
```

```
<ul>
```

```
<div style="text-align: right;">
```

```
<li><a type="button" href="dashboard">DASHBOARD</a></li>
```

```
<li><a type="button" href="logout">LOGOUT</a></li>
```

```
</div>
```



```

</ul>

<div class="class-container" id="container">
    <h4 style="text-align: center; color:white; fontsize:25px;">Welcome {{ msg }}!</h4>
</div>

</nav>

<!-- navbar ends -->

<div class="flex justify-center mt-2">
    <table class="table" style="margin-top: 80px;">
        <caption>USER PROFILE</caption>
        {% for key,value in dictprofile.items() %}
            <tr class="table__row">
                <td class="table__cell"> {{ key }} </td>
                <td class=" table__cell"> {{ value }} </td>
            </tr>
        {% endfor %}
    </table>
</div>

<pre style="font-weight:bold;font-size:20px">To make any changes to your profile, please contact our
    system administrator at gokhul19032@cse.ssn.edu.in.</pre>

</body>

</html>

```

Userrequest.html

```

<!DOCTYPE html>

<html>

<head>
    <title>CURRENT USER'S REQUESTS</title>

```

```
<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/tables.css') }}">
```

```
<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/results.css') }}">
```

```
<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/forms.css') }}">
```

```
<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
```

```
</head>
```

```
<body>
```

```
<!-- navbar starts -->
```

```
<ul>
```

```
    <div style="text-align: right;">
```

```
        <li><a type="button" href="dashboard">DASHBOARD</a></li>
```

```
        <li><a type="button" href="profile">PROFILE</a></li>
```

```
        <li><a type="button" href="logout">LOGOUT</a></li>
```

```
    </div>
```

```
</ul>
```

```
<div class="class-container" id="container">
```

```
    <h4 style="text-align: center; color:white; font-size:25px; margin-bottom: 100px;">You have made  
    the following
```

```
        requests for plasma.</h4>
```

```
</div>
```

```
<!-- navbar ends -->
```

```
<div class="flex justify-center mt-2">
```

```
    <table class="table">
```

```
        <caption>YOUR PAST REQUEST</caption>
```

```
        <tr class="table__header">
```

```
            {% for header in headings %}
```

```
            <th class="table__cell">{{ header }}</th>
```

```

        {% endfor %}

    </tr>

    {% for row in data %}

    <tr class="table__row">

        {% for cell in row %}

    <td class="table__cell">{{ cell }}</td>

    {% endfor %}

    <td>

        <form action="/deleterequest" method="POST" onsubmit="alert('YOU CANNOT UNDO THIS
ACTION!')">

            <input type="hidden" name="username" value="{{ msg }}">

            <input type="hidden" name="pname" value="{{ row[0] }}">

            <button id="deletebutton" class="button" type="submit"> Delete Request </button>

        </form>

    </td>

</tr>

    {% endfor %}

</table>

</div>

<pre style="font-weight:bold; font-size:20px; text-align: center;">

    Don't see a past request you made?

    It was probably closed by the administrator.

    If you think this is a mistake, please contact us at +91 XXXXXX
    XXXXX.

</pre>

</body>

</html>

Viewalldonors.html
<!DOCTYPE html>

<html>

```

<head>

<title>ALL REQUESTS</title>

<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/tables.css') }}">

<link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/results.css') }}">

<link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">

</head>

<body>

<!-- navbar -->

<div style="text-align: right;">

DASHBOARD

LOGOUT

</div>

<div class="class-container" id="container">

<h4 style="text-align: center; color:white; font-size:25px; margin-bottom: 100px;">System has the following

donors.</h4> </div>

<!-- navbar ends -->

<!-- requests table starts -->

<div class="flex justify-center mt-2"> <table

class="table">

<caption>ALL DONORS</caption>

<tr class="table__header">

{% for header in headings %}

```

        <th class="table__cell">{{ header }}</th>
    {% endfor %}
</tr>
{% for row in data %}
<tr class="table__row">
    {% for cell in row %}
        <td class="table__cell">{{ cell }}</td>
    {% endfor %}
</tr>
{% endfor %}
</table>
</div>
</body>

</html>

```

Viewallreqs.html

```

<!DOCTYPE html>
<html>

<head>
    <title>ALL REQUESTS</title>

    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/tables.css') }}">

    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/results.css') }}">

    <link type="text/css" rel="stylesheet" href="{{ url_for('static', filename='/stylesheets/forms.css') }}">

    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
</head>

```

```

<body>

<!-- navbar -->

<ul>

  <div style="text-align: right;">

    <li><a type="button" href="admindashboard">DASHBOARD</a></li>

    <li><a type="button" href="logout">LOGOUT</a></li>

  </div>

</ul>

<div class="class-container" id="container">

  <h4 style="text-align: center; color:white; font-size:25px; margin-bottom: 100px;">System has the
  following
  requests.</h4>

</div>

<!-- navbar ends -->

<!-- requests table starts -->

<div class="flex justify-center mt-2">

  <table class="table">

    <caption>ALL REQUESTS</caption>

    <tr class="table__header">

      {% for header in headings %}

        <th class="table__cell">{{ header }}</th>

      {% endfor %}

    </tr>

    {% for row in data %}

      <tr class="table__row">

        {% for cell in row %}

          <td class="table__cell">{{ cell }}</td>

          {% endfor %}

        <td>

```

```

        <form action="/deletereq" method="POST" onsubmit="alert('YOU CANNOT UNDO THIS ACTION!')">
            <input type="hidden" name="username" value="{{ row[0] }}">
            <input type="hidden" name="pname" value="{{ row[1] }}">
            <button id="deletebutton" class="button" type="submit"> Delete Request </button>
        </form>
        <form action="/approvereq" method="POST">
            <input type="hidden" name="pname" value="{{ row[1]
        }}">
            <input type="hidden" name="phone" value="{{ row[2]
        }}">
            <input type="hidden" name="state" value="{{ row[5] }}">
            <input type="hidden" name="bgp" value="{{ row[6] }}">
            <button class="button" type="submit"> Approve Request </button>
        </form>
    </td>
    </tr>
    {% endfor %}
</table>

</div>
</body>

</html>

```

Dashboards.css

```

@media screen and (min-width: 801px) {
    html {
        background-image: url("https://s3.jp-tok.cloud-object-
storage.appdomain.cloud/plasma-53045-bucket/blood.jpg");
        background-repeat: no-repeat;
        background-size: cover;
    }
}

```

```
    overflow-x: hidden;

    min-width: 100%;

    min-height: 100%;
}
```

```
body {

    background: linear-gradient(rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 0.7));

    height: 100vh;

    font-family: "Montserrat";

    color: #fff;

    font-size: 22px;
}
```

```
* {

    margin: 0; padding: 0;

    overflow-x: hidden;
}
```

```
ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

    overflow: hidden;

    margin-left: 10px;
}
```

```
li {

    float: left;
}
```



```
li a {  
    display: block;  
    color: white;  
    text-align: center;    padding: 14px 16px;  
    text-decoration: none;  
}
```

```
li a:hover:not(.active) {  
    background-color: black;  
}
```

```
.contfoot {  
    display: flex; justify-content: center;  
}
```

```
.active {  
    background-color: #17e3d9;  
}
```

```
#head {  
    justify-content: center;  
    text-align: center;  
    align-items: center;  
    font-size: 2.5vw;  
    margin-top: 40px;  
    margin-bottom: 40px;  
    position: relative;  
    animation-name: headers;  
    animation-duration: 3s;
```

```
    animation-iteration-count: infinite;
}
```

```
@keyframes headers {
```

```
    0% {
        left: 0vw;
        top: 0vw;
    }
```

```
    33% {
        left: -20vw;
        top: 0vw;
    }
```

```
    66% {
        left: 20vw;
        top: 0vw;
    }
```

```
    100% {
        left: 0vw;
        top: 0vw;
    }
}
```

```
.navlist {
    text-decoration: none;
    color: #fff;
    font-size: 1.5vw;
    padding: 0.2px;
```

```
}
```

```
.ulist {
```

```
    display: flex;
```

```
    flex-direction: row;
```

```
    list-style: none;
```

```
}
```

```
.item {
```

```
    float: left;
```

```
    list-style: none;    margin-left: 2vw; border:
```

```
1.5px solid whitesmoke;
```

```
}
```

```
.item:hover {
```

```
    float: left;
```

```
    list-style: none;    margin-left: 2vw;
```

```
    border: 0.01vw solid red;
```

```
    background-color: red;
```

```
    color: white;
```

```
}
```

```
#navi {
```

```
    margin-top: 2vw;
```

```
    justify-content: center;
```

```
    align-items: center;
```

```
    height: 5vw;
```

```
}
```

```
.itemselect {
```

```
float: left;

list-style: none;    margin-left: 2vw;

border: 0.01vw solid red;

background-color: red;

color: white;
}
```

```
.bdy {

margin-top: -7.5vw; margin-left: 2vw;

margin-right: 2vw;

}
```

```
#imag {

float: left;

margin-right: 3vw;

width: 25vw;

height: 25vw;

}
```

```
.class-container {

width: 100%;

position: relative;

padding: 20px 0;

transition: 0.3s ease 0s;

background-attachment: fixed;

background-size: cover;

}
```

```
.class-container:before {

position: absolute;
```

```
    left: 0;

    top: 0;

    width: 10%;

    height: 10%;    content: "";
}
```

```
.header {

    width: 100%;

    text-align: center;

    margin-bottom: 30px;

    position: relative;
}
```

```
.header__title {

    color: #fff;

    font-size: 35px;

    font-family: 'Montserrat', sans-serif;

    font-weight: normal;

    margin: 0;
}
```

```
.flasher {

    background-color: light grey;

    font-weight: bold;

    color: red;

    font-family: 'Times New Roman', Times, serif;

    text-align: center;

    margin-bottom: 10px;
}
```

```
}
```

Forms.css

```
.button {
```

```
    display: inline-block;
```

```
    padding: 15px 25px;
```

```
    font-size: 24px;    cursor: pointer;
```

```
text-align: center; text-decoration:
```

```
none;
```

```
    outline: none;
```

```
    color: #fff;
```

```
    background-color: #4CAF50;
```

```
    border: none;
```

```
    border-radius: 15px;
```

```
}
```

```
.button:hover {
```

```
    background-color: #3e8e41
```

```
}
```

```
#deletebutton {
```

```
    background-color: red;
```

```
}
```

```
input, option {
```

```
    color: black;
```

```
}
```

Index.css

```
@media screen and (min-width: 801px) {
```

```
    html {
```

```
/* background-image: url("/static/images/blood.jpg"); */
```

```
background-image: url("https://s3.jp-tok.cloud-object-storage.appdomain.cloud/plasma-53045-bucket/blood.jpg");  
background-repeat: no-repeat;  
background-size: cover;  
overflow-x: hidden;  
min-width: 100%; min-height: 100%;  
}
```

```
body {  
background: linear-gradient(rgba(0, 0, 0, 0.7), rgba(0, 0, 0, 0.7));  
height: 100vh;  
font-family: "Montserrat";  
color: #fff;  
font-size: 22px;  
}
```

```
* {  
margin: 0;  
padding: 0;  
overflow-x: hidden;  
}
```

```
ul {  
list-style-type: none;  
margin: 0;  
padding: 0;  
overflow: hidden;  
margin-left: 10px;
```

```
}
```

```
li {
```

```
    float: left;
```

```
}
```

```
    li a {
```

```
        display: block;
```

```
        color: white;
```

```
text-align: center;    padding: 14px 16px;
```

```
text-decoration: none;
```

```
}
```

```
li a:hover:not(.active) {
```

```
    background-color: black;
```

```
}
```

```
.contfoot {
```

```
    display: flex;
```

```
    justify-content: center;
```

```
}
```

```
.active {
```

```
    background-color: #17e3d9;
```

```
}
```

```
#head {
```

```
    justify-content: center;
```

```
text-align: center;
```

```
align-items: center;
```



```
font-size: 2.5vw;  
margin-top: 40px;  
margin-bottom: 40px;  
position: relative;  
animation-name: headers; animation-duration: 3s;  
        animation-iteration-count: infinite;  
}
```

```
@keyframes headers {
```

```
0% {  
    left: 0vw;  
    top: 0vw;  
}
```

```
33% {  
    left: -20vw;  
    top: 0vw;  
}
```

```
66% {  
    left: 20vw;  
    top: 0vw;  
}
```

```
100% {  
    left: 0vw;  
    top: 0vw;  
}
```

```
}
```

```
.navlist {  
  text-decoration: none;  
  color: #fff;  
  font-size: 1.5vw;  
    padding: 0.2px;  
}  
  
.ulist {  
  display: flex;  
  flex-direction: row;  
  list-style: none;  
}  
  
.item {  
  float: left;  
  list-style: none;  margin-left: 2vw;  
  border: 1.5px solid whitesmoke;  
}  
  
.item:hover {  float: left;  
  list-style: none;  margin-left: 2vw;  
  border: 0.01vw solid red;  
  background-color: red;  
  color: white;  
}  
  
#navi {  
  margin-top: 2vw;  
  justify-content: center;  
  align-items: center;
```

height: 5vw;

}

.itemselect {

float: left;

list-style: none; margin-left: 2vw;

border: 0.01vw solid red;

background-color: red;

color: white;

}

.bdy {

margin-left: 2vw;

margin-right: 2vw;

}

#imag {

float: left;

margin-right: 3vw;

width: 25vw;

height: 25vw;

}

.para {

font-size: 1.5vw;

}

.paras {

font-size: 1.2vw;

margin-top: 20px;

```
}
```

```
.footer-basic ul {  
    padding: 0;  
    list-style: none;  
    text-align: center;  
    font-size: 18px;  
    line-height: 1.6;  
    margin-bottom: 0;  
}
```

```
.footer-basic li { padding: 0 10px;  
    margin-top: 10px;  
    margin-left: 10px;  
}
```

```
.footer-basic .social { text-align: center;  
    padding-bottom: 25px;  
}
```

```
.footer-basic .copyright {  
    margin-top: 15px;  
    text-align: center;  
    font-size: 13px;  
    color: #aaa;  
    margin-bottom: 0;  
}
```

```
.flasher {  
    background-color: lightgrey;
```

```
    font-weight: bold;

    color: red;

    font-family: 'Times New Roman', Times, serif;

    text-align: center;

    margin-bottom: 10px;
}
}
```

```
.history__img {

    max-width: 100%;

    box-shadow: 0 10px 15px rgba(0, 0, 0, 0.4);
}
```

```
.class-container {

    width: 100%;

    position: relative;

    padding: 60px 0;

    transition: 0.3s ease 0s;

    background-attachment: fixed;

    background-size: cover;
}
```

```
.class-container:before {

    position: absolute;

    left: 0;

    top: 0; width:

    100%; height:

    100%; content:

    "";
```

```
}
```

```
.header {  
  width: 100%;  
  text-align: center;  
  margin-bottom: 80px;  
  position: relative;  
}
```

```
.header__title {  
  color: #fff;  
  font-size: 46px;  
  font-family: "Montserrat", sans-serif;  
  font-weight: normal;  
  margin: 0;  
}
```

```
.header__subtitle {  
  color: white;  
  font-family: "Montserrat", sans-serif;  
  font-size: 16px;  
  letter-spacing: 5px; margin: 10px 0 0  
0;  
  font-weight: normal;  
}
```

Logins.css

```
@media screen and (min-width: 801px) {  html {  
    background-image: url("https://s3.jp-tok.cloud-objectstorage.appdomain.cloud/plasma-53045-  
    bucket/blood.jpg");
```

```
background-repeat: no-repeat;

background-size: cover;

overflow-x: hidden;

min-width: 100%;

min-height: 100%;

}
```

```
body {

background: linear-gradient(rgba(0, 0, 0, 0.7), rgba(0, 0, 0,

0.7));

height: 100vh;

color: #fff;

font-family: "Montserrat";

font-size: 22px;

}
```

```
* {

margin: 0;

padding: 0;

overflow-x: hidden;

}
```

```
#head {

justify-content: center;

text-align: center;

align-items: center;

font-size: 2.5vw;

margin-top: 40px;

margin-bottom: 40px;
```

```
position: relative;

animation-name: headers;

animation-duration: 3s;

animation-iteration-count: infinite;
}
```

```
@keyframes headers {
```

```
0% {

    left: 0vw;

    top: 0vw;

}
```

```
33% {

    left: -20vw;

    top: 0vw;

}
```

```
66% {

    left: 20vw;

    top: 0vw;

}
```

```
100% {

    left: 0vw;

    top: 0vw;

}
}
```

```
.bdy {

    margin-left: 2vw;
```



```
margin-right: 2vw;  
}
```

```
#imag {  
float: left;  
margin-right: 3vw;  
width: 25vw;  
height: 25vw;  
}
```

```
.flasher {  
background-color: lightgrey;  
font-weight: bold;  
color: red;  
font-family: 'Times New Roman', Times, serif;  
text-align: center;  
margin-top: 10px;  
margin-bottom: 10px;  
}
```

```
}
```

Results.css

```
@media screen and (min-width: 801px) {  
html {  
background-image: url("https://s3.jp-tok.cloud-object-  
storage.appdomain.cloud/plasma-53045-bucket/blood.jpg");  
background-repeat: no-repeat;  
background-size: cover;  
overflow-x: hidden;  
min-width: 100%;
```

```
    min-height: 100%;  
}
```

```
body {  
    background: linear-gradient(rgba(0, 0, 0, 0.7), rgba(0, 0, 0,  
0.7));  
    height: 100vh;  
    font-family: "Montserrat";  
    color: #fff;  
    font-size: 22px;  
}
```

```
* {  
    margin: 0;  
    padding: 0;  
    overflow-x: hidden;  
}
```

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    overflow: hidden;  
    margin-left: 10px;  
}
```

```
li {  
    float: left;  
}
```

```
li a {  
    display: block;  
    color: white;  
    text-align: center;    padding: 14px 16px;  
    text-decoration: none;  
}
```

```
li a:hover:not(.active) {  
    background-color: black;  
}  
}
```

Tables.css

```
.table {  
    border-spacing: 0;  
    margin: 1rem;  
    background-color: #f5f5f5;  
    margin-bottom: 40px;  
}  
  
.table__row:nth-child(even) {  
    background-color: #e5e5e5;  
}  
  
.table__header {  
    text-align: left;  
}  
  
.table__cell {  
    color: black;    padding: 8px;
```

}
