|  |  |
| --- | --- |
| Team Id | PNT2022TMID50023 |
| Project Name | Plasma Donor Application |

#### 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 &amp; 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 (Explain the features added in the project along with code)

7.1 Feature 1

7.2 Feature 2

7.3 Database Schema (if Applicable)

#### 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

GitHub & Project Demo Link

## 

## CHAPTER 1

## INTRODUCTION

### 1.1.Project Overview

During the COVID 19 crisis, the need for plasma increased, while the number of donors has decreased. A plasma is a liquid portion of the blood, over 55% of human blood is plasma. Plasma therapy is a process where blood is donated by recovered patients in order to establish antibodies that fights the infection.For instance, during COVID 19 crisis the requirement for plasma increased drastically as there were no vaccination found in order to treat the infected patients, with plasma therapy the recovery rates where high but the donor count was very low and in such situations it was very important to get the information about the plasma donors. Saving the donor information and notifying about the current donors would be a helping hand as it can save time and help the users to track down the necessary information about the donors.Plasma is also necessary for the survival of people with cancer, rare disorders,immunological problems, and genetic anomalies. Every blood bank claims to be out of blood, so we need to make people aware of the issue and offer support. Numerous camps,seminars, and applications can be of great help.

1.2.Purpose

The main purpose of our project is to connect the donor and the receipient,and to notify the receipient when there is relative blood group donor.

The Blood Donation Agent is to create an e-Information about the donor and organization that are related to donating the blood. Through this application any person who is interested in donating the blood can register himself in the same way if any organization wants to register itself with this site that can also register. Moreover if any general consumer wants to make request blood online he can also take the help of this site. Admin is the main authority who can do addition" deletion" and modification if required.

## CHAPTER 2

## LITERATURE SURVEY:

2.1 Existing Problem

In existing system,not all users can get access to the information because of the low working of the application or is not able to access any site.sometimes the information is not updated or available for a particular place.In existing system the security is less and latest updates and uploads or not so frequent.

2.2 References

1. **Title**: Serverless computing: Economic and architectural impact .

**Source**: ESEC/FSE .

**Author**: R. C. Gojko Adzic.

**Date**: 2017 .

2.**Title**: Building a chatbot with serverless computing .

**Source:** IBM watson research center .

**Author:** P. C. P. C. a. V. I. M. Yan .

**Date**: 2016 .

3. **Title**: Cloud Event Programming Paradigms: Applications and Analysis .

**Source**: 9th IEEE International Conference on Cloud Computing (CLOUD), pp.

pp. 400-406 .

**Author:** S. E. a. B. J. J. Short.

**Date**:2017 .

4. **Title**: Making Serverless Computing More Serverless .

**Source**: IEEE 11th International Conference on Cloud Computing (CLOUD), pp.

pp. 456-459.

**Author**: Z. Al-Ali .

**Date**: 2018 .

5. **Title**: EMARS: Efficient Management and Allocation of Resources in Serverless .

**Source**: IEEE 11th International Conference on Cloud Computing (CLOUD), pp.

pp. 827-830 .

**Author**: A. S. a. S. Jindal.

**Date**: 2018.

**2.3 Problem Statement Definition**

During COVID 19 crisis the requirement for plasma increased drastically as there were no vaccination found in order to treat the infected patients, with plasma therapy the recovery rates where high but the donor count was very low and in such situations it was very important to get the information about the plasma donors. Saving the donor information and notifying about the current donors would be a helping hand as it can save time and help the users to track down the necessary information about the donors

|  |  |
| --- | --- |
| Who does the problem affect | People suffering with corono |
| What is the issue | When people unamble to find the donor at the right time it leads to loss of life. |
| When does the issue occur | When people in urgency to find the donor |
| Why it is important that we fix the problem | It will save the life of many people |

## CHAPTER

## IDEATION &PROPOSED SOLUTION

3.1 Empathy Map Canvas

**THINK AND FEEL**

* Donars are able to donate their plasma to the receipient.
* Needy person can recover from their diesease
* We can collaborate with Government and save many lives.

**SAY AND DO**

* Creating awareness of donating plasma in public is important.
* Our project is to connect plasma donor and the recipient.
* It should be portable to everyone.

**SEE**

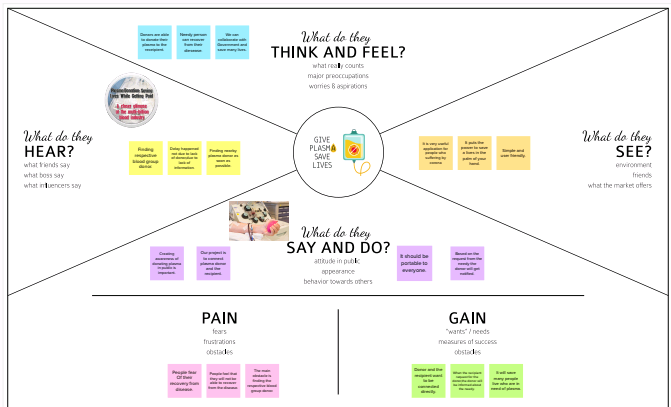
* It is very useful application for people who suffering by corona.
* It puts the power to save a lives in the palm of your hand.
* Simple and user friendly.

**GAIN**

* When the recipient request for the donor,the donor will be informed about the needy.
* It will save many people live who are in need of plasma.

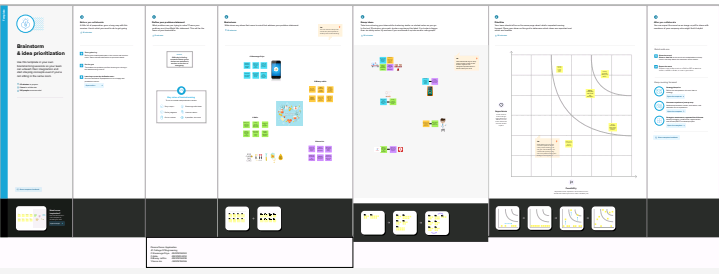
**PAIN**

* People fear Of their recovery from disease.
* People feel that they will not be able to recover from the disease.
* The main obstacle is fnding the respective blood group donor.



**3.2 IDEATION & BRAINSTORMING**

* Difculty in fnding respective blood group donors and ambulance services at the time of emergency.
* Sharing information of ambulance
* Finding respective blood group donors
* Doctors can see the details and surge the patient easil
* Making emergency calls, mails and message



**3.3 PROPOSED SOLUTION**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1. | Problem Statement(Problem to be solved) | * Difficulty in finding respective blood group donars and ambulance services at the time of emergency. |
| 2. | Idea / Solution description | * This system proposed here aims at connecting the donars & the patients by an online application. * It is helpful to find respective blood group donars when anyone is in need. |
| 3. | Novelty / Uniqueness | * Sharing nearby ambulance services. |
| 4. | Social Impact / Customer Satisfaction | * We can save life who are in need of plasma. |
| 5. | Business Model (Revenue Model) | * We can collaborate with Government and it can utilise app to help the people who are need of plasma. |
| 6. | Scalability of the Solution | * Performance and speed do not slow down even if large number of users access the application at the same time. |

**3.4 PROBLEM SOLUTION FIT**



## CHAPTER 4

## REQUIREMENT ANALYSIS

4.1 Functional Requirements

|  |  |  |
| --- | --- | --- |
| **FR No.** | **Functional Requirement (Epic)** | **Sub Requirement (Story / Sub-Task)** |
| FR-1 | User Registration | 1. Registration through Form 2. Registration through Gmail 3. Registration through LinkedIN |
| FR-2 | User Confirmation | 1.Confirmation via Email  2.Confirmation via OTP |
| FR-3 | Data entry | 1.Collect the data from the user  2.Fill the information |
| FR-4 | Analysing data | 1.Analysing data and giving result based on their request |
| FR-5 | Technology Used | 1.Flask  2.IBM DB2  3.Docker  4.Kubernets  5.IBM Cloud |
| FR-6 | Phases | 1.Request phase  2.Response phase |

4.2 Non-Functional Requirements

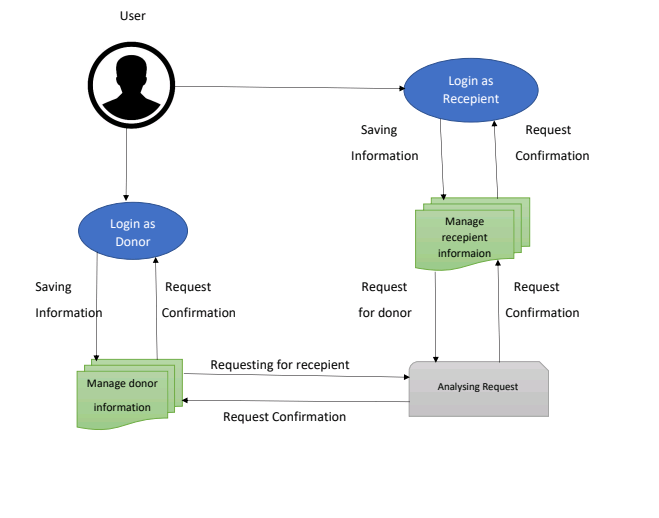
|  |  |  |
| --- | --- | --- |
| **FR No.** | **Non-Functional Requirement** | **Description** |
| NFR-1 | **Usability** | 1.Saving life of people.  2.Finding donor at the right time. |
| NFR-2 | **Security** | 1.Encrypt your data.  2.Avoid security misconfiguration. |
| NFR-3 | **Reliability** | Result will be accurate. |
| NFR-4 | **Performance** | Filtering respective blood group donors will be accurate. |
| NFR-5 | **Availability** | It can be accessed at anytime,  anywhere and anyplace. |
| NFR-6 | **Scalability** | 1. Memory utilization.  2. CPU usage.  3. Networkinput/output.  4. Disk input/output |

## CHAPTER

## PROJECT DESIGN

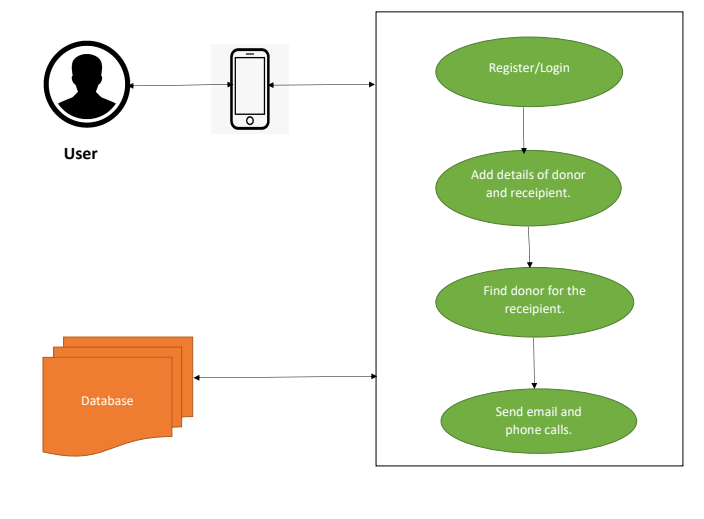
5.1 DataFlow Diagrams

1. The user can login as a donor or receipient.
2. The data will be collected and stored in the databse.
3. When it receives the matching blood group donors it will notification to the receipeint as well as donor.
4. Hence it wil be used to help the needy at the right time.

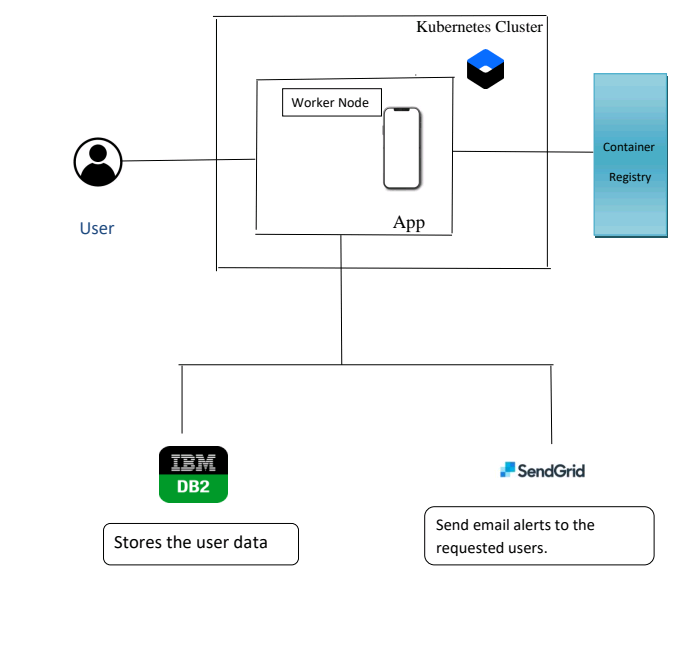


5.2 Solution &Technical Architecture

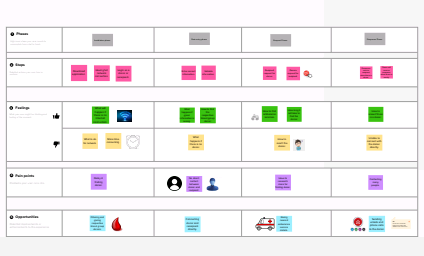
**Solution Architecture**



**Technical Architecture**



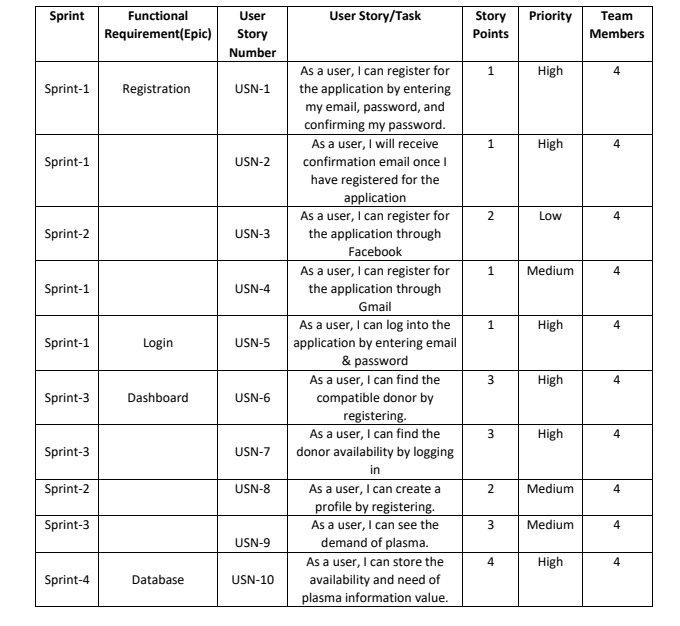
**5.3 User Stories**



## CHAPTER 6

## PROJECT PLANNING AND SCHEDULING

6.1 Sprint Planning And Estimation



**6.2 Sprint Delivery Plan**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **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 | 9 | 6 Days | 23 Oct 2022 | 28 Oct 2022 | 9 | 28 Oct 2022 |
| Sprint-2 | 10 | 8 Days | 29 Oct 2022 | 05 Nov 2022 | 10 | 05 Nov 2022 |
| Sprint-3 | 8 | 6 Days | 05 Nov 2022 | 11 Nov 2022 | 8 | 11 Nov 2022 |
| Sprint-4 | 9 | 7 Days | 12 Nov 2022 | 19 Nov 2022 | 9 | 19 Nov 2022 |

Velocity

1) AV = Sprint Duration/Velocity = 9/6 =1.5

2) AV = Sprint Duration/Velocity = 10/8 = 1.25

3) AV = Sprint Duration/Velocity = 8/6 = 1.33

4) AV = Sprint Duration/Velocity = 9/7 = 1.28

6.3 Reports From JIRA

## 

## CHAPTER 7

## CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Feature 1

login.html

|  |  |
| --- | --- |
| <!DOCTYPE html> |  |
|  | <html> |
|  | <head> |
|  | <title>Plasma donor</title> |
|  | <link rel="stylesheet" type="text/css" href="{{ url\_for('static', filename='style.css')}}"> |
|  | <link href="https://fonts.googleapis.com/css2?family=Jost:wght@500&display=swap" rel="stylesheet"> |
|  | </head> |
|  | <body> |
|  | <div class="main"> |
|  | <h4>{{msg}}</h4> |
|  | <input type="checkbox" id="chk" aria-hidden="true"> |
|  |  |
|  | <div class="login"> |
|  | <form action = "{{ url\_for('login') }}" method = "POST"> |
|  | <label for="chk" aria-hidden="true">Login</label> |
|  | <input type="email" name="email" placeholder="Email" required=""> |
|  | <input type="password" name="password" placeholder="Password" required=""> |
|  | <button>Login</button> |
|  | </form> |
|  | </div> |
|  | </div> |
|  | </body> |
|  | </html> |
|  | <!-- partial --> |
|  | </body> |
|  | </html> |

Register.html

|  |  |
| --- | --- |
| <html> |  |
|  | <head> |
|  | </head> |
|  | <body> |
|  | {% extends "layout1.html" %} |
|  |  |
|  | {% block content %} |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | <h1>Registration page</h1> |
|  | <form class="form-horizontal" action="{{url\_for('register')}}" method="POST"> |
|  |  |
|  | <div class="form-group"> |
|  | <label class="control-label col-sm-2" for="pwd">Name:</label> |
|  | <div class="col-sm-6"> |
|  | <input type="text" class="form-control" id ="name" name="name" placehloder="Enter Name" required> |
|  | </div> |
|  | </div> |
|  |  |
|  | <div class="form-group"> |
|  | <label class="control-label col-sm-2" for="pwd">Email:</label> |
|  | <div class="col-sm-6"> |
|  | <input type="email" class="form-control" id ="email" name="email" placehloder="Enter Email" required> |
|  | </div> |
|  | </div> |
|  |  |
|  |  |
|  | <div class="form-group"> |
|  | <label class="control-label col-sm-2" for="pwd">Phno:</label> |
|  | <div class="col-sm-6"> |
|  | <input type="text" class="form-control" id ="phno" name="phno" placehloder="Enter Phno" required> |
|  | </div> |
|  | </div> |
|  |  |
|  | <div class="form-group"> |
|  | <label class="control-label col-sm-2" for="pwd">Age:</label> |
|  | <div class="col-sm-6"> |
|  | <input type="text" class="form-control" id ="age" name="age" placehloder="Enter age" required> |
|  | </div> |
|  | </div> |
|  |  |
|  | <div class="form-group"> |
|  | <label class="control-label col-sm-2" for="pwd">Blood Group:</label> |
|  | <div class="col-sm-6"> |
|  | <select class="form-control" id="blood\_group" name="blood\_group"> |
|  | <option value="A+">A+</option> |
|  | <option value="A-">A-</option> |
|  | <option value="B+">B+</option> |
|  | <option value="B-">B-</option> |
|  | <option value="AB+">AB+</option> |
|  | <option value="AB-">AB-</option> |
|  | <option value="O+">O+</option> |
|  | <option value="O-">O-</option> |
|  | <option value="Don't know">Don't Know</option> |
|  | </select> |
|  | </div> |
|  | </div> |
|  |  |
|  | <div class="form-group"> |
|  | <label class="control-label col-sm-2" for="pwd">Address:</label> |
|  | <div class="col-sm-6"> |
|  | <input type="text" class="form-control" id ="address" name="address" placehloder="Enter address" required> |
|  | </div> |
|  |  |
|  |  |
|  | <div class="form-group"> |
|  | <label class="control-label col-sm-2" for="pwd">Password:</label> |
|  | <div class="col-sm-6"> |
|  | <input type="text" class="form-control" id ="password" name="password" placehloder="Enter password" required> |
|  | </div> |
|  | </div> |
|  |  |
|  |  |
|  | <div class="form-group"> |
|  | <div class="col-sm-offset-2 col-sm-10 "> |
|  | <button type="submit" class="btn btn-default">Submit</button> |
|  | </div> |
|  | </div> |
|  | </form> |
|  |  |
|  | {% endblock %} |
|  | </body> |
|  | </html> |

about.html

|  |  |
| --- | --- |
| <html lang="en" dir="ltr"> |  |
|  | <head> |
|  |  |
|  | <title>Plasma Donor Application</title> |
|  | <link rel="stylesheet" href="{{ url\_for('static', filename='style2.css')}}"> |
|  | <meta name="viewport" content="width=device-width, initial-scale=1.0"> |
|  | <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"/> |
|  | <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.3/css/all.min.css"/> |
|  |  |
|  |  |
|  |  |
|  |  |
|  | </head> |
|  | <body1 > |
|  |  |
|  | <div class="main"> |
|  | <div class="navbar"> |
|  | <div class="menu"> |
|  | <ul> |
|  | <li><a href="/">HOME</a></li> |
|  | <li><a href="about">ABOUT</a></li> |
|  | <li><a href="loginn">LOGIN</a></li> |
|  | <li><a href="reques">REQUEST</a></li> |
|  | <li><a href="donor">DONATE</a></li> |
|  | </ul> |
|  | </div> |
|  |  |
|  | <div class="search"> |
|  | <input class="srch" type="search" name="" placeholder="Type To text"> |
|  | <a href="#"> <button class="btn">Search</button></a> |
|  | </div> |
|  |  |
|  | </div> |
|  |  |
|  | <div class="content-wrapper"> |
|  | <div class="wrapper"> |
|  | <div class="center-line"> |
|  | <!-- <a href="#" class="scroll-icon"></a> --> |
|  |  |
|  | </div> |
|  | <div class="row row-1"> |
|  | <section> |
|  | <i class="icon fa fa-tint"></i> |
|  |  |
|  | <div class="details"> |
|  | <span class="title" style="color: #302b62;">NEED PLAMSA TO SAVE LIVES?</span> |
|  | </div> |
|  | <p> Register in our website by clicking on our REGISTER button on the navigation bar</p> |
|  | <!-- <div class="bottom"> |
|  | <a href="#">Read more</a> |
|  | </div> --> |
|  | </section> |
|  | </div> |
|  | <div class="row row-2"> |
|  | <section> |
|  | <i class="icon fa fa-tint"></i> |
|  | <div class="details"> |
|  | <span class="title"style="color: #302b62;">WANNA FIND A MATCHING DONAR?</span> |
|  |  |
|  | </div> |
|  | <p> Click on Login and search for your matching donar</p> |
|  | <!-- <div class="bottom"> |
|  | <a href="#">Read more</a> |
|  | <i>- Someone famous</i> |
|  | </div> --> |
|  | </section> |
|  | </div> |
|  | <div class="row row-1"> |
|  | <section> |
|  | <i class="icon fa fa-tint"></i> |
|  | <div class="details"> |
|  | <span class="title" style="color: #302b62;">INTERESTED TO DONATE PLASMA?</span> |
|  | </div> |
|  | <p>Register in our website to Donate Plasma in our camps</p> |
|  | </section> |
|  | </div> |
|  | <div class="row row-2"> |
|  | <section> |
|  | <i class="icon fa fa-tint"></i> |
|  | <div class="details"> |
|  | <span class="title" style="color: #302b62;">URGENT NEED OF PLASMA?</span> |
|  | </div> |
|  | <p>Plasma Donor Website helps you in your critical situations</p> |
|  | </section> |
|  | </div> |
|  |  |
|  | <div class="row row-2"> |
|  | <section> |
|  | <i class="icon fa fa-tint"></i> |
|  | </section> |
|  | </div> |
|  |  |
|  | </div> |
|  | </div> |
|  | </div> |
|  | </div> |
|  | <!-- <div class="content1"> |
|  | <div class="text">Fully Responsive Footer Section</div> |
|  | <div class="p">HTML and CSS (Flexbox)</div> |
|  | </div> --> |
|  | <footer> |
|  | <div class="main-content"> |
|  | <div class="leftbox"> |
|  | <h2 style="color:#fff;">About us</h2> |
|  | <div class="content2" style="color:#fff;"> |
|  | <p>Plasma Donor is an application exclusively made for<br> |
|  | the recipients who are in search of plasma for operations, <br> |
|  | Covid, cancer and many other causes and the donors who <br> |
|  | are interested to donate their plasma to save other lives. |
|  | </p> |
|  | <div class="social"> |
|  | <a href="#"><span class="fab fa-facebook-f"></span></a> |
|  | <a href="#"><span class="fab fa-twitter"></span></a> |
|  | <a href="#"><span class="fab fa-instagram"></span></a> |
|  | <a href="#"><span class="fab fa-youtube"></span></a> |
|  | </div> |
|  | </div> |
|  | </div> |
|  |  |
|  | <div class="rightbox"> |
|  | <h2 style="color:#fff;">Contact us</h2> |
|  | <div class="content3"> |
|  | <form action="#"> |
|  | <div class="email"> |
|  | <div class="text">Email </div> |
|  | <input type="email" placeholder="Email" required> |
|  | </div> |
|  | <div class="msg"> |
|  | <div class="text">Message</div> |
|  | <textarea rows="2" cols="25" placeholder="Message" required></textarea> |
|  | </div> |
|  | <div class="btn"> |
|  | <button type="submit">Send</button> |
|  | </div> |
|  | </form> |
|  | </div> |
|  | </div> |
|  | </div> |
|  |  |
|  | <div class="bottom"> |
|  | <center> |
|  | <span class="credit">Created By <a href="#">avc college </a> |</span> |
|  | <span class="far fa-copyright"></span><span> 2022 All rights reserved.</span> |
|  | </center> |
|  | </div> |
|  | </footer> |
|  | </body> |
|  | </html> |
|  | <script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"> |
|  |  |
|  |  |
|  | </script> |
|  | </body1> |
|  | </html> |

dashboard.html

|  |  |
| --- | --- |
| <div > |  |
|  | <h3>Welcome {{session['user']}} </h3> |
|  | <br> |
|  | </div> |
|  |  |
|  |  |
|  |  |
|  | {% if role=="donor" %} |
|  | <!--Donor Dashboard Start--> |
|  | {% if session['role']=="donor" %} |
|  |  |
|  | <nav class="stroke"> |
|  | <ul> |
|  | <li><a href="/dashboard-allrequests">All Requests</a></li> |
|  | <li><a href="/dashboard">My Dashboard</a></li> |
|  | <li><a href="/dashboard-savedrequests">Saved Requests</a></li> |
|  |  |
|  |  |
|  | </ul> |
|  | </nav> |
|  |  |
|  | {% endif %} |
|  |  |
|  | <hr> |
|  | <h1>Donor-Dashboard</h1> |
|  | <div class="donor-dashboard"> |
|  | <h3 align = "center">Requests</h3> |
|  | <br><br> |
|  |  |
|  | <div class="table-all"> |
|  |  |
|  | <table name="list" class="dashboard-table"> |
|  | <tr class="dashboard-tr"> |
|  | {% for heading in headings %} |
|  | <th class="dashboard-th">{{heading}}</th> |
|  |  |
|  | {% endfor %} |
|  | </tr> |
|  |  |
|  | {% for users in requesters\_list %} |
|  | <tr class="dashboard-tr"> |
|  |  |
|  | <td class = "dashboard-td">{{ users['NAME'] }}</td> |
|  | <td class = "dashboard-td">{{ users['EMAIL'] }}</td> |
|  | <td class = "dashboard-td">{{ users['PHONE'] }}</td> |
|  | <td class = "dashboard-td">{{ users['ADDRESS'] }}</td> |
|  | <td class = "dashboard-td">{{ users['AGE'] }}</td> |
|  | <td class = "dashboard-td">{{ users['BLOODGROUP'] }}</td> |
|  | <!--Accept button |
|  | <td class = "dashboard-td"><button class="btn btn-primary btn-md" id="accept-btn-{{users[0]}}" onclick="accept(id)">Accept</button></td> |
|  | --> |
|  | </tr> |
|  | {% endfor %} |
|  |  |
|  |  |
|  | </table> |
|  | </div> |
|  | </div> |
|  |  |
|  |  |
|  |  |
|  |  |
|  | <div id = "beneficiary-info"> |
|  |  |
|  | <div id="request-accepted" > |
|  | <br> |
|  | <center>You have accepted the request!</center> |
|  | <br> |
|  |  |
|  | </div> |
|  | <h3><center>Beneficiary Information</center></h3> |
|  | <table align="center" class = "donor-table"> |
|  | <tr class="donor-table-row"> |
|  | <th class="donor-table-header">Name</th> |
|  | <td class="donor-table-data">Abcde</td> |
|  | </tr> |
|  | <tr class="donor-table-row"> |
|  | <th class="donor-table-header">Blood Group</th> |
|  | <td class="donor-table-data">B+</td> |
|  | </tr> |
|  | <tr class="donor-table-row"> |
|  | <th class="donor-table-header">Email</th> |
|  | <td class="donor-table-data">abcde@gmail.com</td> |
|  | </tr> |
|  | <tr class="donor-table-row"> |
|  | <th class="donor-table-header">Phone</th> |
|  | <td class="donor-table-data">9999999999</td> |
|  | </tr> |
|  |  |
|  | </table> |
|  | <br><br> |
|  |  |
|  | </div> |
|  |  |
|  |  |
|  | <!--donor Dashboard End--> |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | <script> |
|  | function requestDonors(){ |
|  |  |
|  | document.getElementById('donor-req-btn').innerHTML="Requested"; |
|  | document.getElementById('donor-req-btn').disabled=true; |
|  | document.getElementById('request-pending').style.display="block"; |
|  |  |
|  | } |
|  |  |
|  | function endrequestingDonor(){ |
|  | document.getElementById('donor-req-btn').innerHTML="Requesting stopped"; |
|  | document.getElementById('donor-req-btn').disabled=true; |
|  | document.getElementById('request-pending').style.display="block"; |
|  |  |
|  | } |
|  |  |
|  | function accept(id){ |
|  |  |
|  | document.getElementById(id).disabled=true; |
|  | document.getElementById(id).innerHTML="Accepted"; |
|  |  |
|  | document.getElementById('request-accepted').style.display="block"; |
|  | document.getElementById('beneficiary-info').style.display="block"; |
|  |  |
|  | } |
|  | function acceptDonor(){ |
|  | document.getElementById('accept-btn-donor').disabled=true; |
|  | document.getElementById('accept-btn-donor').innerHTML="Accepted"; |
|  |  |
|  | document.getElementById('donor-accepted').style.display="block"; |
|  | } |
|  |  |
|  | </script> |

request.html

<!DOCTYPE html>

<html lang="en">

<head>

<title>Request</title>

<script>

window.watsonAssistantChatOptions = {

integrationID: "45bd3710-4a8e-472f-a697-c5c4ad557f26", // The ID of this integration.

region: "au-syd", // The region your integration is hosted in.

serviceInstanceID: "fc35279c-db04-4a5a-828f-d392eb7be3d5", // The ID of your service instance.

onLoad: function(instance) { instance.render(); }

};

setTimeout(function(){

const t=document.createElement('script');

t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" + (window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.js";

document.head.appendChild(t);

});

</script>

<style>

body {

background: url(https://wallpaper.dog/large/20389692.jpg) #f0f0f4;

height: 553px;

text-align: center;

font-family: cursive;

}

h3 {

color: rgb(5, 168, 106);

font-size: 40px;

}

.normal

{

border: 2px;

border-style: groove;

border-radius: 7px;

padding-top: 8px;

padding-left: 8px;

padding-bottom: 8px;

padding-right: 8px;

border-color: rgb(5, 168, 106);

height: 25px;

width: 400px;

}

button {

border: 2px;

border-radius: 10px;

padding-top: 8px;

padding-left: 8px;

padding-bottom: 8px;

padding-right: 8px;

border-style: groove;

width: 90px;

opacity: 90%;

border-color: rgb(0, 0, 0);

cursor: pointer;

}

.right {

width: 50%;

float: right;

}

.left {

width: 50%;

float: left;

}

p {

text-align: left;

padding-left: 7px;

}

.address {

border: 2px;

border-style: groove;

border-radius: 7px;

padding-top: 8px;

padding-left: 8px;

padding-bottom: 8px;

padding-right: 8px;

border-color: rgb(0, 0, 0);

height: 95px;

width: 395px;

}

.req\_cont {

margin-top: 50px;

width: 40%;

border: 1px solid #9c9c9c;

border-radius: 9px;

box-shadow: 4px 3px 11px 0px grey;

background-color: #cc9143;

}

.req\_cont\_main {

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

}

.bottom\_py {

width: 100%;

display: flex;

flex-direction: column;

align-items: center;

}

.form{

display:flex;

flex-direction: column;

align-items: center;

justify-content:center;

}

</style>

</head>

<body>

<div class="req\_cont\_main">

<div class="req\_cont">

<h3>APPLY FOR PLASMA DONATION</h3>

<form method="post" class="form">

<input class="normal" type="text" name="name" placeholder="Name" required /><br>

<input class="normal" type="email" name="email" placeholder="Email" required /><br>

<input class="normal" type="number" name="mobile" placeholder="Mobile" required /><br>

<input class="normal" type="number" name="age" placeholder="Age" required /><br>

<input class="normal" type="text" name="sex" placeholder="Gender" required /><br>

<input class="normal" type="text" name="blood\_group" placeholder="Blood Group" required /><br>

<textarea class="address" name="address" placeholder="Address" required ></textarea><br>

<input class="normal" type="text" name="covid" placeholder="Date tested covid positive in this format yyyy-mm-dd" required /><br>

<button style="text-align: center;" type="submit">Donate</button>

</form>

<a href="/login">Revert to login</a>

<div class="bottom\_py">

<p style="color: rgb(27, 161, 27);">{{success}}</p>

<p style="color: rgb(214, 25, 25);">{{error}}</p>

</div>

</div>

</div>

</body>

</html>

7.2 Feature 2

mail.py

# importing libraries

from flask import Flask

from flask\_mail import Mail, Message

app = Flask(\_\_name\_\_)

mail = Mail(app) # instantiate the mail class

# configuration of mail

app.config['MAIL\_SERVER'] = 'smtp.gmail.com'

app.config['MAIL\_PORT'] = 465

app.config['MAIL\_USERNAME'] = 'sampletest685@gmail.com'

app.config['MAIL\_PASSWORD'] = 'hneucvnontsuwgpj'

app.config['MAIL\_USE\_TLS'] = False

app.config['MAIL\_USE\_SSL'] = True

mail = Mail(app)

# message object mapped to a particular URL ‘/’

@app.route("/")

def index():

msg = Message(

'Hello',

sender='sampletest685@gmail.com',

recipients=['priya5535@gmail.com']

)

msg.body = 'Hello Flask message sent from Flask-Mail'

mail.send(msg)

return 'Sent'

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

app.py

from distutils.log import debug

from sendgridmail import sendmail

from flask import Flask, render\_template, request, redirect, url\_for, session

import ibm\_db

import re

import os

from dotenv import load\_dotenv

load\_dotenv()

app = Flask(\_\_name\_\_)

app.secret\_key = 'a'

conn = ibm\_db.connect("DATABASE=bludb;HOSTNAME=fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32731;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=yys87370;PWD=J35NHAilIxej1m1d;",'','')

@app.route('/')

@app.route('/login')

def login():

return render\_template('login.html')

@app.route('/loginpage',methods=['GET', 'POST'])

def loginpage():

global userid

msg = ''

if request.method == 'POST' :

username = request.form['username']

password = request.form['password']

sql = "SELECT \* FROM donors 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']

msg = 'Logged in successfully !'

sendmail(account['EMAIL'],'Plasma donor App login','You are successfully logged in!')

return redirect(url\_for('dash'))

else:

msg = 'Incorrect username / password !'

return render\_template('login.html', msg = msg)

@app.route('/registration')

def home():

return render\_template('register.html')

@app.route('/register',methods=['GET', 'POST'])

def register():

msg = ''

if request.method == 'POST' :

username = request.form['username']

email = request.form['email']

password = request.form['password']

phone = request.form['phone']

city = request.form['city']

infect = request.form['infect']

blood = request.form['blood']

sql = "SELECT \* FROM donors 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 = 'Account already exists !'

elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

msg = 'Invalid email address !'

elif not re.match(r'[A-Za-z0-9]+', username):

msg = 'name must contain only characters and numbers !'

else:

insert\_sql = "INSERT INTO donors VALUES (?, ?, ?, ?, ?, ?, ?)"

prep\_stmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(prep\_stmt, 1, username)

ibm\_db.bind\_param(prep\_stmt, 2, email)

ibm\_db.bind\_param(prep\_stmt, 3, password)

ibm\_db.bind\_param(prep\_stmt, 4, city)

ibm\_db.bind\_param(prep\_stmt, 5, infect)

ibm\_db.bind\_param(prep\_stmt, 6, blood)

ibm\_db.bind\_param(prep\_stmt, 7, phone)

ibm\_db.execute(prep\_stmt)

msg = 'You have successfully registered !'

sendmail(email,'Plasma donor App Registration','You are successfully Registered {}!'.format(username))

elif request.method == 'POST':

msg = 'Please fill out the form !'

return render\_template('register.html', msg = msg)

@app.route('/dashboard')

def dash():

if session['loggedin'] == True:

sql = "SELECT COUNT(\*), (SELECT COUNT(\*) FROM DONORS WHERE blood= 'O Positive'), (SELECT COUNT(\*) FROM DONORS WHERE blood='A Positive'), (SELECT COUNT(\*) FROM DONORS WHERE blood='B Positive'), (SELECT COUNT(\*) FROM DONORS WHERE blood='AB Positive'), (SELECT COUNT(\*) FROM DONORS WHERE blood='O Negative'), (SELECT COUNT(\*) FROM DONORS WHERE blood='A Negative'), (SELECT COUNT(\*) FROM DONORS WHERE blood='B Negative'), (SELECT COUNT(\*) FROM DONORS WHERE blood='AB Negative') FROM donors"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.execute(stmt)

account = ibm\_db.fetch\_assoc(stmt)

print(account)

return render\_template('dashboard.html',b=account)

else:

msg = 'Please login!'

return render\_template('login.html', msg = msg)

@app.route('/requester')

def requester():

if session['loggedin'] == True:

return render\_template('request.html')

else:

msg = 'Please login!'

return render\_template('login.html', msg = msg)

@app.route('/requested',methods=['POST'])

def requested():

bloodgrp = request.form['bloodgrp']

address = request.form['address']

name= request.form['name']

email= request.form['email']

phone= request.form['phone']

insert\_sql = "INSERT INTO requested VALUES (?, ?, ?, ?, ?)"

prep\_stmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(prep\_stmt, 1, bloodgrp)

ibm\_db.bind\_param(prep\_stmt, 2, address)

ibm\_db.bind\_param(prep\_stmt, 3, name)

ibm\_db.bind\_param(prep\_stmt, 4, email)

ibm\_db.bind\_param(prep\_stmt, 5, phone)

ibm\_db.execute(prep\_stmt)

sendmail(email,'Plasma donor App plasma request','Your request for plasma is recieved.')

return render\_template('request.html', pred="Your request is sent to the concerned people.")

@app.route('/logout')

def logout():

session.pop('loggedin', None)

session.pop('id', None)

session.pop('username', None)

return render\_template('login.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0',debug='TRUE')

## CHAPTER 8

## TESTING

8.1 Test Cases

By filterig the blood group we can find the matching plasma donor for the patient and we can notify both the donor and the receipient at the same time.

|  |  |  |
| --- | --- | --- |
| **TEST CASE NO** | **TEST CASE SCENARIO** | **RESULT** |
| 1 | Verify user is able to fill id and password and login to the website | Pass |
| 2 | Verify user has to fill the information | Pass |
| 3 | Verify user is able to navigate from home page to other pages? | Pass |
| 4 | Verify user is able to get the expected results accurately | Pass |
| 5 | Verify information collected page elements | Pass |

8.2 User Acceptance Testing

**1. Purpose of Document**

The purpose of this document is to briefly explain the test coverage and open issues of the Plasma Donor Application project at the time of the release to User Acceptance Testing (UAT).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Section** | **Total**  **Cases** | **Not Tested** | **Fail** | **Pass** |
| Print Engine | 7 | 0 | 0 | 7 |
| Client Application | 51 | 0 | 0 | 51 |
| Security | 2 | 0 | 0 | 2 |
| Outsource Shipping | 3 | 0 | 0 | 3 |
| Exception Reporting | 9 | 0 | 0 | 9 |
| Final Report  Output | 4 | 0 | 0 | 4 |
| Version Control | 2 | 0 | 0 | 2 |

CHAPTER 9

## RESULTS

9.1 Performance Metrics

We are connecting the donor and receipient in a short period of time.Filtering the representive blood group donors and notifing the reeipient.It is also used to know the nearby ambulance services.User can add Donors.User can Add Patients.User can search specifically for Donors.User can search specifically for Patients.User can add Edit Profile

## CHAPTER 10

## ADVANTAGES & DISADVANTAGES

* Blood Plasma Donations are used for slightly more specific purposes than a general blood donation.
* Donated plasma can be frozen and stored for up to one year.
* Plasma transfusions are often lifesaving.
* By connecting donor and receipient ,people who are in need of plasma can get donor at the right time.
* It's a web-enabled project.
* This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.
* Easier and faster data transfer through latest technology associated with the computer and communication.
* Through these features it will increase the efficiency, accuracy and transparency.

Disadvantages:

* People in rural areas unable to connect to the donors easily due to network issues.
* Wrong inputs will affect the project outputs.
* Internet Connection is mandatory.

## 

## 

## CHAPTER 11

## CONCLUSION

In recent days, it is noticed the increase in plasma request posts on social media such as Facebook, Twitter, and Instagram. Interestingly there are many people across the world interested in donating plasma when there is a need, but those donors don’t have an access to know about the plasma donation requests in their local area. This is because that there is no platform to connect local plasma donors with patients. This application solves the problem and creates a communication channel through authorized clinics whenever a patient needs plasma donation. It is a useful tool to find compatible blood donors who can receive plasma request posts in their local area. Clinics can use this web application to maintain the plasma donation activity. Collected data through this application can be used to analyse donations to requests rates in a local area to increase the awareness of peopleAdvantages: by conducting donations camps.

Plasma Donor Application can be developed to further improve user accessibility via integrating this application with various social networks application program interfaces (APIs). Consequently, users can login and sign up using various social networks. This would increase number of donors and enhances the process of plasma donation.

## CHAPTER 12

## FUTURE SCOPE

User interface (UI) can be improved in future to accommodate global audience by supporting different languages across countries. Data scraping can be done from different social networks and can be shown in the PlasmaRequest Feeds. Appointments can be synchronized with Google and Outlook calendars for the ease of users.

## CHAPTER 13

## APPENDIX

#### **SOURCE CODE**

from distutils.log import debug

from sendgridmail import sendmail

from flask import Flask, render\_template, request, redirect, url\_for, session

import ibm\_db

import re

import os

from dotenv import load\_dotenv

load\_dotenv()

app = Flask(\_\_name\_\_)

app.secret\_key = 'a'

conn = ibm\_db.connect("DATABASE=bludb;HOSTNAME=fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32731;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=yys87370;PWD=J35NHAilIxej1m1d;",'','')

@app.route('/')

@app.route('/login')

def login():

return render\_template('login.html')

@app.route('/loginpage',methods=['GET', 'POST'])

def loginpage():

global userid

msg = ''

if request.method == 'POST' :

username = request.form['username']

password = request.form['password']

sql = "SELECT \* FROM donors 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']

msg = 'Logged in successfully !'

sendmail(account['EMAIL'],'Plasma donor App login','You are successfully logged in!')

return redirect(url\_for('dash'))

else:

msg = 'Incorrect username / password !'

return render\_template('login.html', msg = msg)

@app.route('/registration')

def home():

return render\_template('register.html')

@app.route('/register',methods=['GET', 'POST'])

def register():

msg = ''

if request.method == 'POST' :

username = request.form['username']

email = request.form['email']

password = request.form['password']

phone = request.form['phone']

city = request.form['city']

infect = request.form['infect']

blood = request.form['blood']

sql = "SELECT \* FROM donors 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 = 'Account already exists !'

elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):

msg = 'Invalid email address !'

elif not re.match(r'[A-Za-z0-9]+', username):

msg = 'name must contain only characters and numbers !'

else:

insert\_sql = "INSERT INTO donors VALUES (?, ?, ?, ?, ?, ?, ?)"

prep\_stmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(prep\_stmt, 1, username)

ibm\_db.bind\_param(prep\_stmt, 2, email)

ibm\_db.bind\_param(prep\_stmt, 3, password)

ibm\_db.bind\_param(prep\_stmt, 4, city)

ibm\_db.bind\_param(prep\_stmt, 5, infect)

ibm\_db.bind\_param(prep\_stmt, 6, blood)

ibm\_db.bind\_param(prep\_stmt, 7, phone)

ibm\_db.execute(prep\_stmt)

msg = 'You have successfully registered !'

sendmail(email,'Plasma donor App Registration','You are successfully Registered {}!'.format(username))

elif request.method == 'POST':

msg = 'Please fill out the form !'

return render\_template('register.html', msg = msg)

@app.route('/dashboard')

def dash():

if session['loggedin'] == True:

sql = "SELECT COUNT(\*), (SELECT COUNT(\*) FROM DONORS WHERE blood= 'O Positive'), (SELECT COUNT(\*) FROM DONORS WHERE blood='A Positive'), (SELECT COUNT(\*) FROM DONORS WHERE blood='B Positive'), (SELECT COUNT(\*) FROM DONORS WHERE blood='AB Positive'), (SELECT COUNT(\*) FROM DONORS WHERE blood='O Negative'), (SELECT COUNT(\*) FROM DONORS WHERE blood='A Negative'), (SELECT COUNT(\*) FROM DONORS WHERE blood='B Negative'), (SELECT COUNT(\*) FROM DONORS WHERE blood='AB Negative') FROM donors"

stmt = ibm\_db.prepare(conn, sql)

ibm\_db.execute(stmt)

account = ibm\_db.fetch\_assoc(stmt)

print(account)

return render\_template('dashboard.html',b=account)

else:

msg = 'Please login!'

return render\_template('login.html', msg = msg)

@app.route('/requester')

def requester():

if session['loggedin'] == True:

return render\_template('request.html')

else:

msg = 'Please login!'

return render\_template('login.html', msg = msg)

@app.route('/requested',methods=['POST'])

def requested():

bloodgrp = request.form['bloodgrp']

address = request.form['address']

name= request.form['name']

email= request.form['email']

phone= request.form['phone']

insert\_sql = "INSERT INTO requested VALUES (?, ?, ?, ?, ?)"

prep\_stmt = ibm\_db.prepare(conn, insert\_sql)

ibm\_db.bind\_param(prep\_stmt, 1, bloodgrp)

ibm\_db.bind\_param(prep\_stmt, 2, address)

ibm\_db.bind\_param(prep\_stmt, 3, name)

ibm\_db.bind\_param(prep\_stmt, 4, email)

ibm\_db.bind\_param(prep\_stmt, 5, phone)

ibm\_db.execute(prep\_stmt)

sendmail(email,'Plasma donor App plasma request','Your request for plasma is recieved.')

return render\_template('request.html', pred="Your request is sent to the concerned people.")

@app.route('/logout')

def logout():

session.pop('loggedin', None)

session.pop('id', None)

session.pop('username', None)

return render\_template('login.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(host='0.0.0.0',debug='TRUE')

### GITHUP LINK:

<https://github.com/IBM-EPBL/IBM-Project-41305-1660641135>

### PROJECT DEMO LINK:

