KONGU ENGINEERING COLLEGE

(An Autonomous Institution)

PLASMA DONOR APPLICATION PROJECT

DONE BY TEAM ID: PNT2022TMID04437

PREETHI S (737819CSR145)

RISHIKA M (737819CSR160)

NIVANJITHA K (737819CSR126)

PONMATHI K (737819CSR133)

TABLE OF CONTENTS

1.INTRODUCTON

1.1 Project Overview:

Patients with severe liver disease or numerous clotting factor deficits, as well as those who have undergone trauma, burns, or shock, frequently get plasma. The patient's blood volume is increased as a result, which can aids in blood coagulation and helps to prevent shock. The number of people with Covid-19 infection has increased, as has the demand for the plasma of patients who have recovered. The antibodies that are already in our systems can aid someone in overcoming the infection. Plasma donation saves lives, and donors' and blood/plasma facilities' communication is key to this. Smart apps are increasingly viewed as a crucial communication tool, and if they are created with the users' requirements and preferences in mind, plasma donation could make the best use of them.

1.2 Purpose:

In our opinion we intend to create an application that is user-friendly for people who require plasma or who wish to donate plasma to anyone who is in need. However, during design and development, areas of concern including privacy and secrecy should be taken into account. Age was found to be a contributing factor that might reduce donors' propensity to use apps. This system is used if anyone needs a Plasma Donor.

This system comprises the user where he/she can request for a Plasma.

• Donor/Needer can Accept or Reject the request.

• The person who wants to donate his/her plasma needs to register in our application providing required information which are name, age, blood group, phone number, and location, etc.

• Patients who need plasma can also fill the form to request the plasma. Patients can directly call the donor by taking his/her contact number from the application.

• User can also search based on location they are living .

• Just a single search allows anyone to reach maximum number of plasma donors in minimum possible time

2.Literature Survey

2.1 Existing Problem:

In most of the existing plasma donor application then system is closed for general plasma donation and mainly focused on COVID-19 patients for plasma donation, the android mobile user will not be able to insert or view details if the server goes down and a disadvantage of single point of failure. Most of the user details remains unverified and its difficult to track the fake users. The user interface of the application is not being user friendly and the user must have a device with android operating system with an active internet connection to interact with this application.

2.2 Problem Statement Definition:

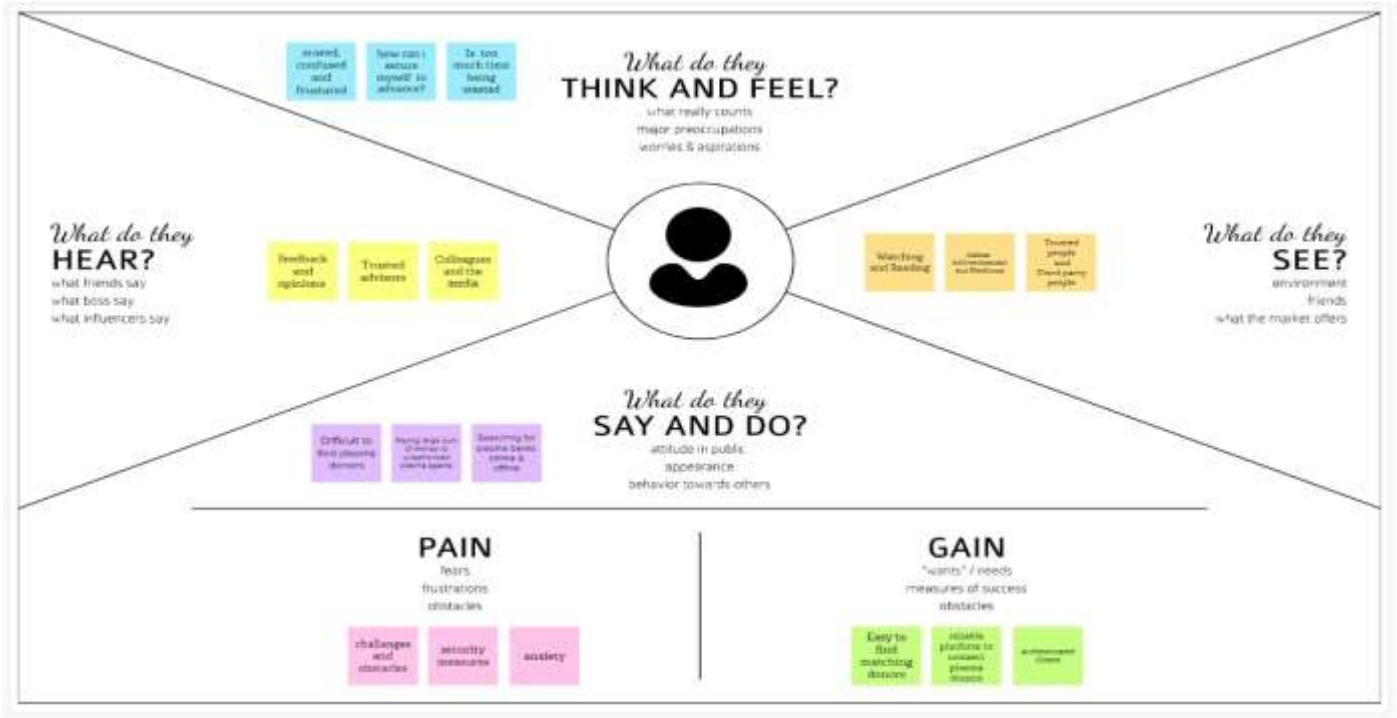
Plasma donation saves lives, and the communication between blood/plasma centres and donors plays a vital role in this. Smart apps are now considered an important communication tool, and could be best utilized in plasma donation if they are designed to fit the users' needs and preferences. We plan

to make a User-friendly application for users who are in need for plasma or who wish to donate plasma to anyone who are in need. However, areas of concern, including privacy and confidentiality, should be considered during design and development. Age was identified as a contributing factor that might decrease the likelihood of app usage among donors. The donation centre staff focused on the educational features of the app and emphasized the importance of the app providing statistics and sending notifications and reminders to donors. Ideation and Proposed Solution

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:

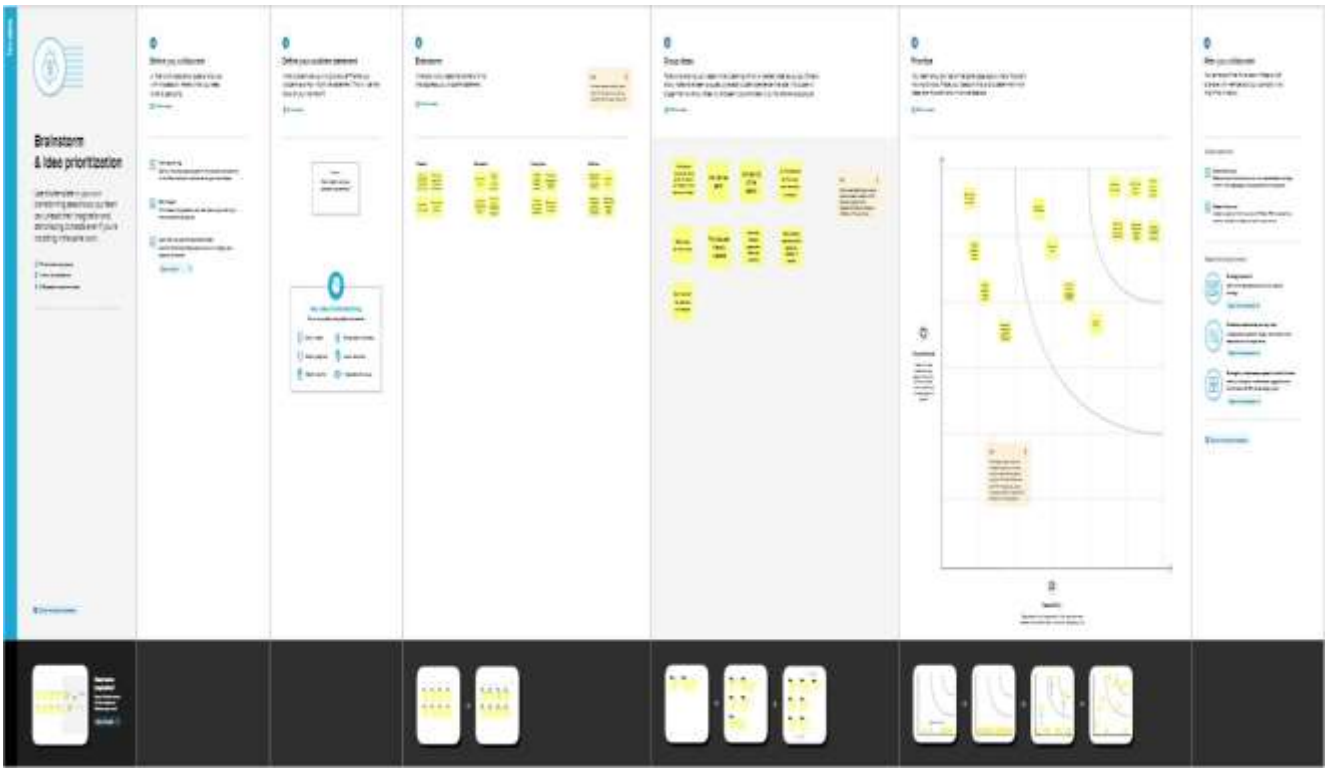
An empathy map canvas helps brands provide a better experience for users by helping teams understand the perspectives and mind set of their customers. Using a template to create an empathy map canvas reduces the preparation time and standardizes the process so you create empathy map canvases of similar quality



3.2 Ideation and Brainstorming:

Brainstorming is a group activity where everyone comes together to discuss strategies for growth and improvement. You can exchange ideas, share important information and use these meetings as informal catch-up sessions with your co-workers. Brainstorming combines a relaxed, informal approach to problem solving with lateral thinking. It encourages people to come up with thoughts and ideas that can, at first, seem a bit crazy. Some of these ideas can be crafted into original, creative solutions to a problem, while others can spark even more ideas.

Ideation is the process where you generate ideas and solutions through sessions such as Sketching, Prototyping, Brainstorming, Brain writing, Worst Possible Idea, and a wealth of other ideation techniques. Ideation is also the third stage in the Design Thinking process.



As you can see, ideation is not just a one-time idea generation or a brainstorming session. In fact, we can divide ideation in these three stages: generation, selection, and development.

3.3 Proposed Solution:

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. The recommendation solution is as follows: In this application if the userwants to receive they have to register with their personal details. After successful registration of user. A successful registration email is send to the user . After successful registration user will be directed to home page or he/she can login with your username and password. You can see the donor details in the donor list page. If the user is donor then he/she will fill the donation interest form which includes their Name, blood group details, location, last time donated date , phone number, email id. After that he/she may receive the donation request where he/she can contact the receiver. If the user is receiver then he/she can see the list of donors available and they can raise their request and contact donor directly.

S.No.	Parameter	Description
1.	Problem Statement (Problem to besolved)	Plasma needer face more difficulties in findingthe plasma donor and get panic
2.	Idea / Solution description	Make more connection between plasma donorand needer
3.	Novelty / Uniqueness	No third party interaction between donor andneeder and easy user interface
4.	Social Impact / user Satisfaction	Impact between the users by the application ismade easy communication and make them more secured and comfort
5.	Business Model (Revenue Model)	Non-revenue model
6.	Scalability of the Solution	The main goal of the application is to provide high Scalability by given more option for user toselect their interest like donate or assist

3.4 Problem Fit:

1.CUSTOMER SEGMENT	5.AVAILABLE SOLUTION	9.PROBLEM ROOT CAUSE
Receiver Donor	Application is designed with the ability to store donor information and received when required	Communication delay between donors and receivers has been resolved. Proper customer support is provided to reduce the technical issues.

2.JOBS TO BE DONE/PROBLEM	6.CUSTOMER CONSTRAINTS	10. YOUR SOLUTION
I) Inform about the receiver to the donor. II) Provide quick solutions and response to the request	Lack of knowledge about the donor available	Generate the unique id to hide the personal information about the donor and receiver to satisfy the customer request as soon as possible.

3.TRIGGERS	7. BEHAVIOR	
Watson service and assistance , Chat bot serves as a trigger	The receiver is informed about the donor using a unique id such that their personal information are hidden	

4) EMOTIONS:BEFORE/AFTER	8) CHANNELS of BEHAVIOR	
BEFORE : Lesser notifications , Increased waiting time AFTER : Information about the donor is easily available and customer satisfaction.	• Login to site • Fill the credentials • Generation of Unique ID. • Information about the donor or receiver	

4. Requirement Analysis:

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement	Sub Requirement
FR-1	User Registration	Register for the application by filling required information
FR-2	User Confirmation	Registration Confirmation email will be sent
FR-3	User Login	User can Login using registered email and password
FR-4	Sent Request	If plasma is required, the needed person will request the donor
FR-5	Search for plasma	Search for same category donor
FR-6	Updating	Admin can update plasma donor's donation information
FR-7	Notification	Needer get notified if request is accepted or rejected by the donor. If accepts needer can contact with donor directly

4.2 Non-Functional Requirements:

Following are the non-functional requirements of the proposed solution

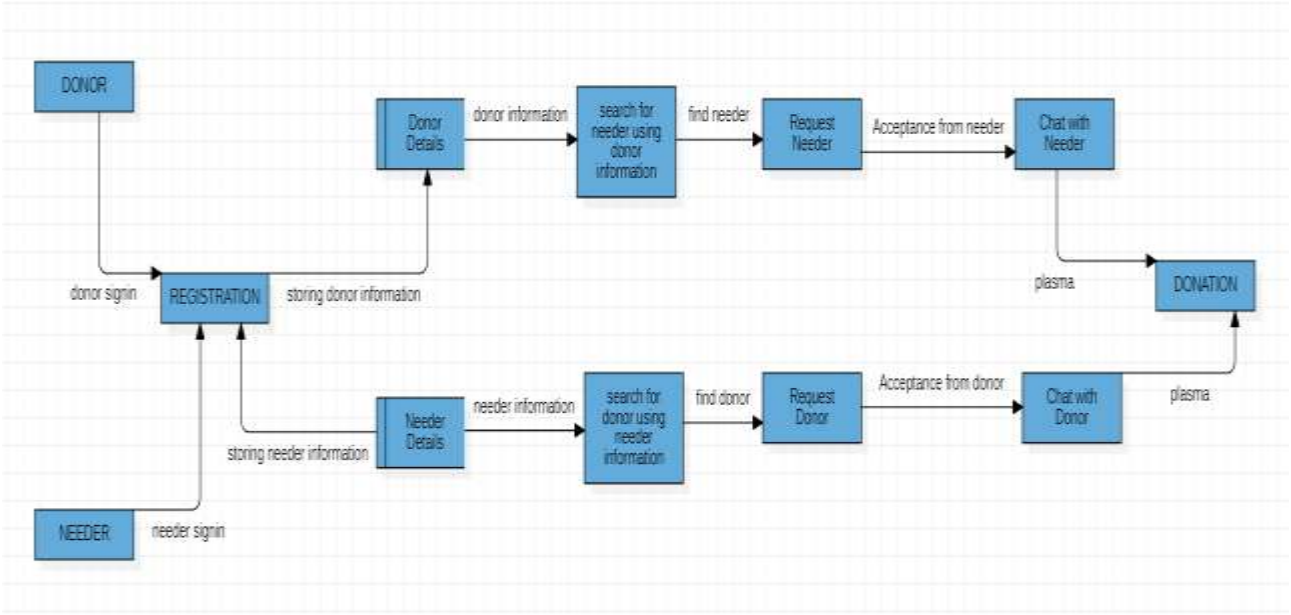
FR No	Non-Functional Requirement	Description
NFR-1	Usability	There should be a well designed user interface for plasm donor application.
NFR-2	Security	Data shared by donors must be secured and storage space must be more. Databases are able to keep all the donor information that isviewed by applications. It must be secured with email Id and password.

NFR-3	Scalability	The system offers the proper resources for issue solutions and is designed to protect sensitive information during all phases of operation.
NFR-4	Availability	The service should available all time i.e 24/7Without any issues.

5. Project Design

5.1 Data flow Diagrams:

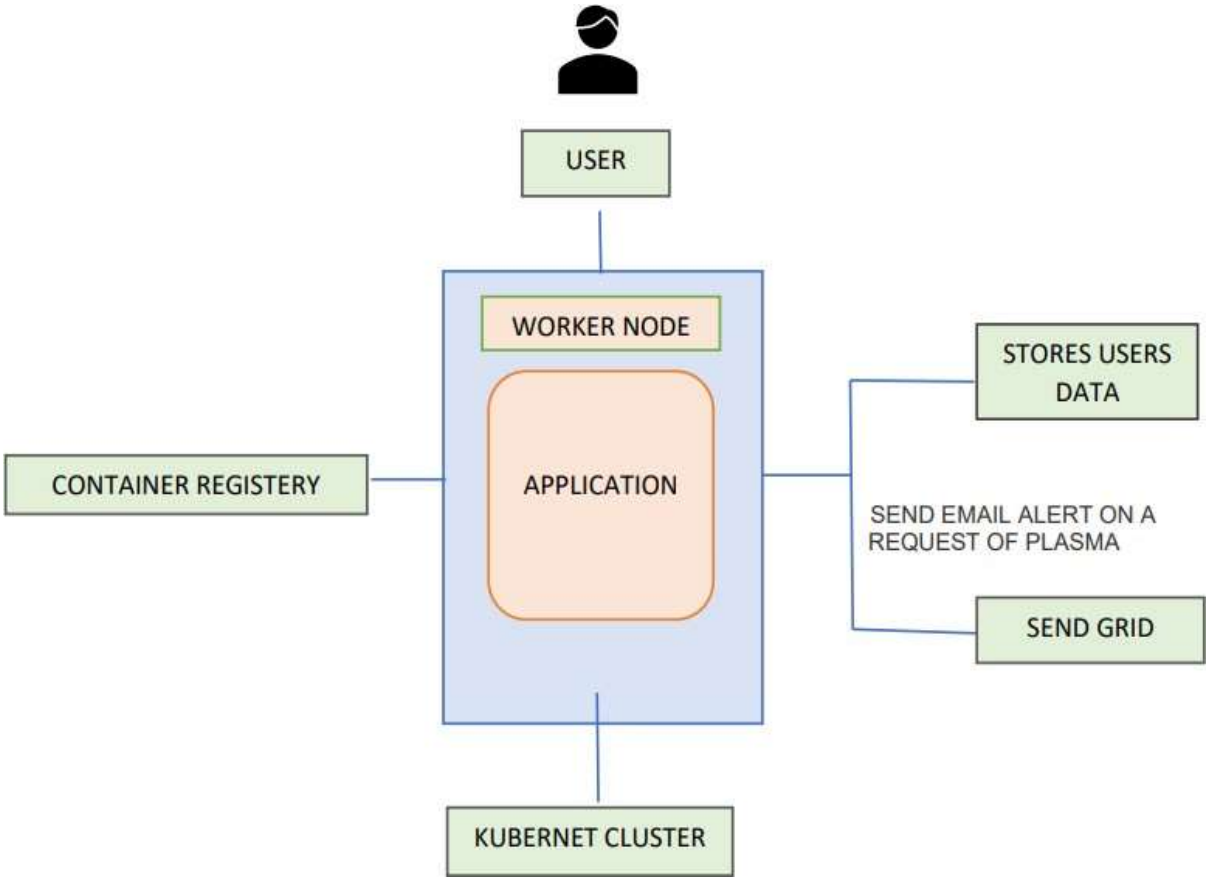
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within asystem. A neat and clear DFD can depict the right amount of the system requirement graphically. Itshows how data enters and leaves the system, what changes the information, and where data is stored.



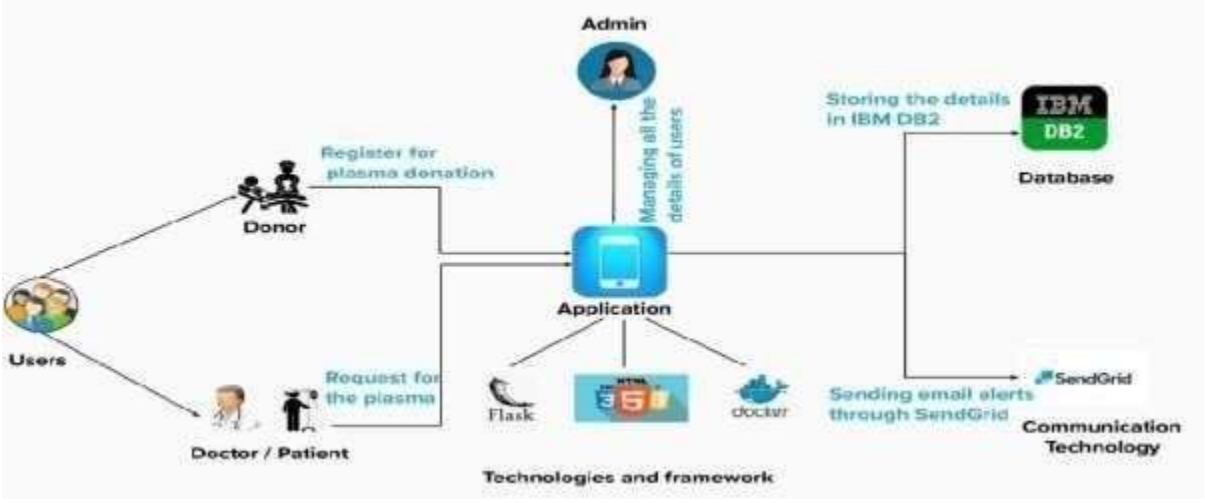
5.2 Solution & Technical Architecture:

Technical Architecture (TA) is a form of IT architecture that is used to design computer systems. Itinvolves the development of a technical blueprint with regard to the arrangement, interaction, and interdependence of all elements so that system-relevant requirements are met.

Solution Architecture:



Technical Architecture:



6 Project Planning & Scheduling:

6.1 Sprint Planning & Estimation:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-6	As a user, I can see the details for the new request / accepted request	I can view my details for the request	High	Sprint-1
		USN-7	As a user, I can search for the nearby	I can see the nearby donor/needer	High	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
			donor/needer matching with donor/needer information	information		
		USN-8	As a user, I can make request to the donor/needer	I can see only the unique id of the donor/needer	High	Sprint-1
		USN-9	As a user, I can accept the request from the donor/needer	I can chat with the donor/needer	High	Sprint-1
Customer (Web user)						
Customer Care Executive						
Administrator	Login		As an admin, I can login to the applicaton by entering email & password			
	Dashboard		As an admin, I can see the no of donors and needers	I can see the donor/needer information		

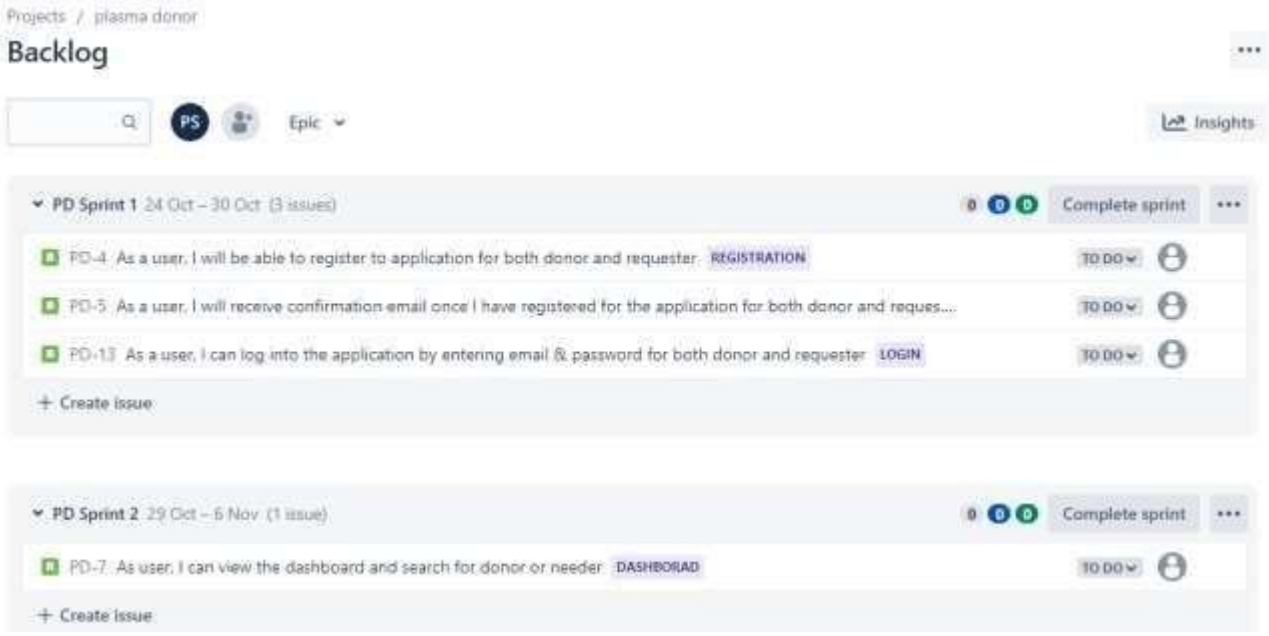
6.2 Sprint Delivery Schedule:

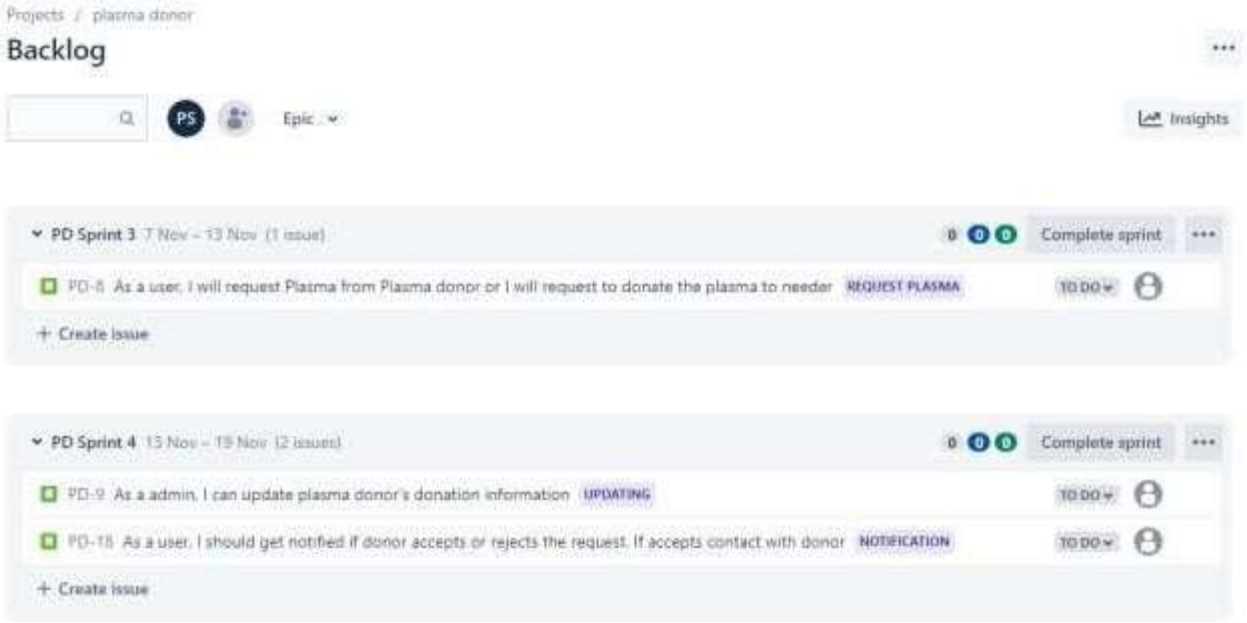
A sprint schedule is a document that outlines sprint planning from end to end. It's one of the first steps in the agile sprint planning process—and something that requires adequate research, planning, and communication.

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	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.3 Reports from JIIRA:

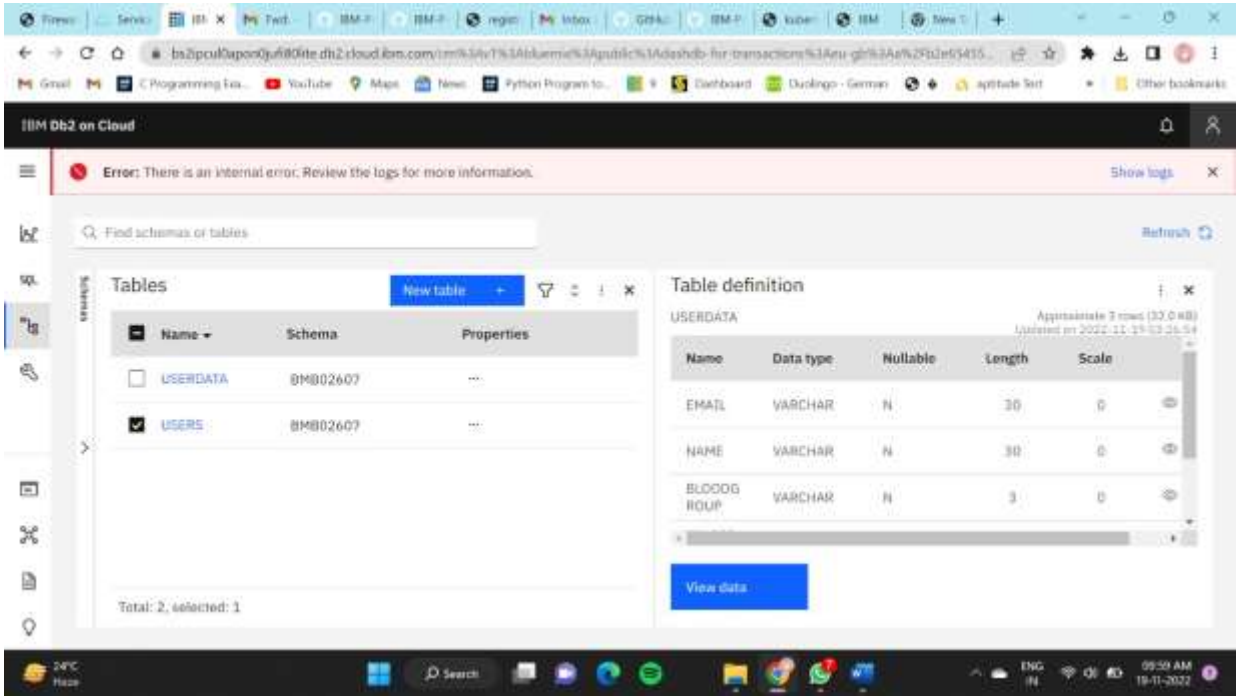
JIIRA is a software application used for issue tracking and project management. The tool, developed by the Australian software company Atlassian, has become widely used by agile development teams to track bugs, stories, epics, and other tasks

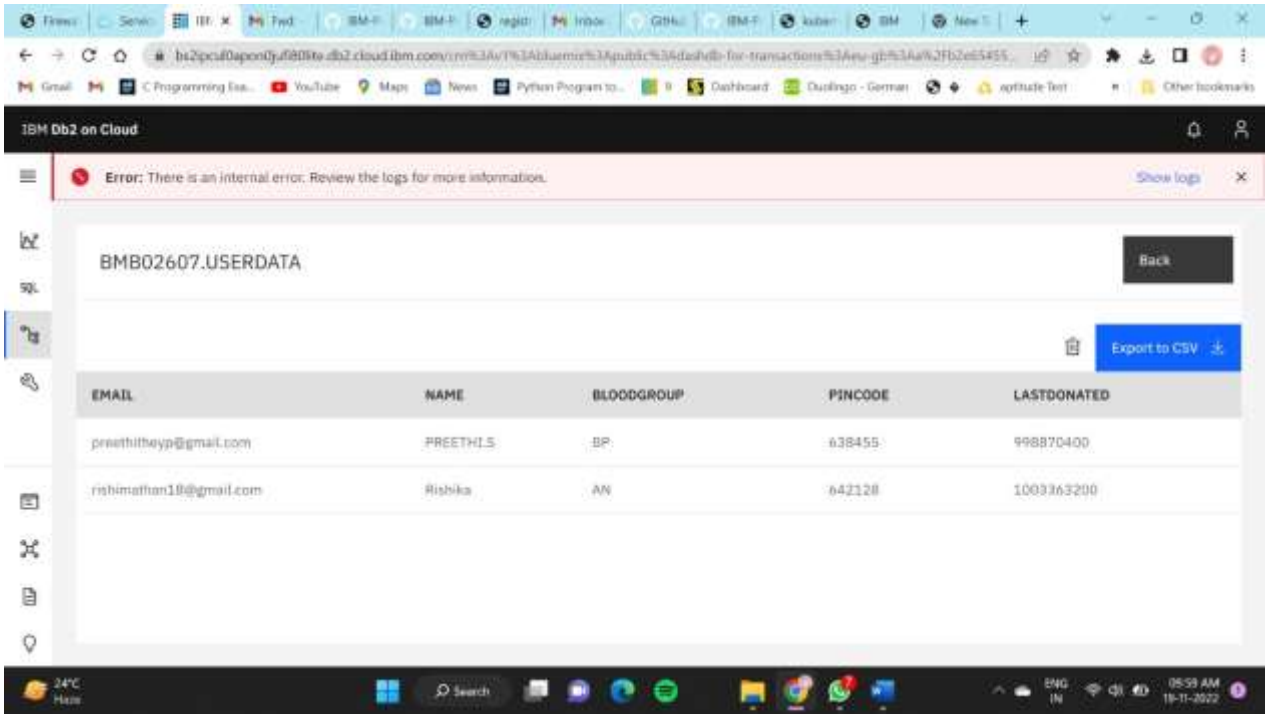


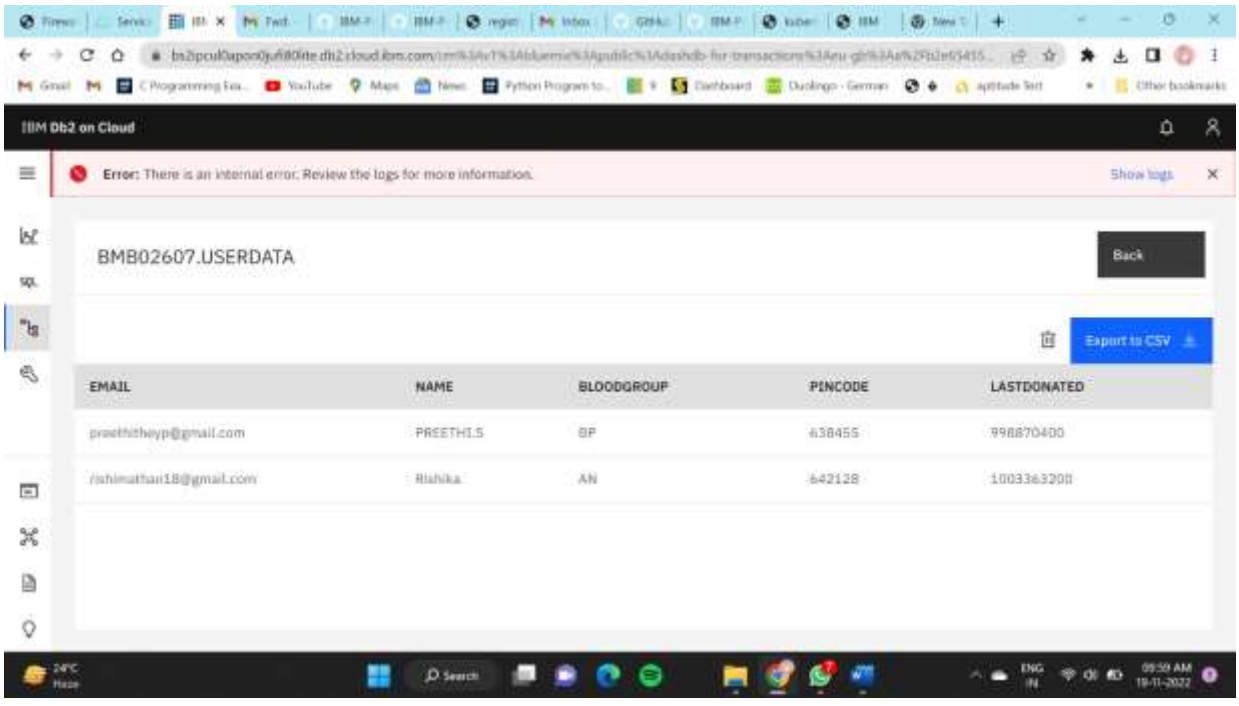


7. Coding & Solution:

7.1 Database Schema:




IBM Db2 on Cloud
Error: There is an internal error. Review the logs for more information.
Show logs
BMB02607.USERDATA
Back
Export to CSV
EMAIL
NAME
BLOODGROUP
PINCODE
LASTDONATED
preethitheyp@gmail.com
PREETHI.S
BP
638455
998870400
rishimathan18@gmail.com
Rishika
AN
642128
1003363200


IBM Db2 on Cloud
Error: There is an internal error. Review the logs for more information.
Show logs
BMB02607.USERDATA
Back
Export to CSV
EMAIL
NAME
BLOODGROUP
PINCODE
LASTDONATED
preethitheyp@gmail.com
PREETHI.S
BP
638455
998870400
rishimathan18@gmail.com
Rishika
AN
642128
1003363200

8. Testing

Testing is the process of evaluation a software item to detect differences between given input and expected output. Testing is a process should be done during the development phase.

Test Case Id	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
TC-1	Install PET app in android phone	Transfer PET app	Open Application with it home page	Application executed with home page	Pass
TC-2	Enter valid data in username and password field	lyall	Show home page for user lyall	Displayed home page for user lyall	Pass
TC-3	Enter a valid data in username and leave password field empty	lyall	Show error	Didn't show any error	Fail

9. Results

Source Code:

#Home.html

```
<html>
   <head>
     <meta charset="UTF-8">
       <meta name="viewport" contents="width=device-width, initial-scale=1.0">
       <link rel="stylesheet" href="/static/css/dashboardstyle.css">
       <title> Dashboard </title>
       <title>Donor Details</title>

       <script

       src="https://code.jquery.com/jquery-3.3.1.js"
       src="jquery.js"
       integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60="
       crossorigin="anonymous">

       </script>
   </head>
   <body>
     <div class="navigation" id="navbar">
       <nav class="nav">
          <div>
             <div class="nav-symbol">
                <strong href="#" class="nav_logo">Dashboard</strong>
```

```
        </div>


        <div class="nav-list">
            <a href="/profile" class="nav_link">
                <ion-icon name="person-outline" class="nav_icon"></ion-icon>
                <span class="nav_name" >Profile</span>

            </a>
            <a href="/" class="nav_link">
                <ion-icon name="medical" class="nav_icon"></ion-icon>
                <span class="nav_name" >Donors list</span>
            </a>
        </div>
      </div>

      <a href="/logout" class="nav_link">
         <ion-icon name="log-out" class="nav_icon"></ion-icon>
         <span class="nav_name">Logout</span>
      </a>
   </nav>
</div>

<script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"></script>

<h1 style="text-align: center;">Donor List</h1>
<div class="head">
    <form action="/" method="get">
```

```
        <label style="font-size: 20px;">Get donor from other district</label>
        <input class="input_1" style="font-size: 20px;" type="text" name="pincode"
placeholder="Pincode">
        <button class="search" type="submit">Search</button>
      </form>
   </div>

    <div class="row">
      <div class="column">
        <div class="card">
          <h1 style="text-align: left">A+</h1>
          <p style="font-weight: 200px;text-align: left;">Available donors - {{data.count.AP}}</p>
          <button class="numbtn" type="submit"
onclick="window.location.href='/plasmarequest?group=AP'">Request</button>
        </div>
      </div>
      <div class="column">
        <div class="card">
          <h1 style="text-align: left">B+</h1>
          <p style="font-weight: 200px;text-align: left;">Available donors - {{data.count.BP}}</p>
          <button class="numbtn" type="submit"
onclick="window.location.href='/plasmarequest?group=BP'" >Request</button>
        </div>
      </div>

      <div class="column">
        <div class="card">
          <h1 style="text-align: left">AB+</h1>
          <p style="font-weight: 200px;text-align: left;">Available donors - {{data.count.ABP}}</p>
```

```
          <button class="numbtn" type="submit"
onclick="window.location.href='/plasmarequest?group=ABP'">Request</button>
        </div>
      </div>

      <div class="column">
        <div class="card">
          <h1 style="text-align: left">O+</h1>
          <p style="font-weight: 200px;text-align: left;">Available donors - {{data.count.OP}}</p>
          <button class="numbtn" type="submit"
onclick="window.location.href='/plasmarequest?group=OP'">Request</button>
        </div>
      </div>

      <div class="column">
        <div class="card">
          <h1 style="text-align: left">A-</h1>
          <p style="font-weight: 200px;text-align: left;">Available donors - {{data.count.AN}}</p>
          <button class="numbtn" type="submit"
onclick="window.location.href='/plasmarequest?group=AN'">Request</button>
        </div>
      </div>

      <div class="column">
        <div class="card">
          <h1 style="text-align: left">B-</h1>
          <p style="font-weight: 200px;text-align: left;">Available donors - {{data.count.BN}}</p>
          <button class="numbtn" type="submit"
onclick="window.location.href='/plasmarequest?group=BN'">Request</button>
        </div>
```

```
        </div>


        <div class="column">
          <div class="card">
            <h1 style="text-align: left">AB-</h1>
            <p style="font-weight: 200px;text-align: left;">Available donors -
{{data.count.ABN}}</p>
            <button class="numbtn" type="submit"
onclick="window.location.href='/plasmarequest?group=ABN'">Request</button>
          </div>
        </div>


        <div class="column">
          <div class="card">
            <h1 style="text-align: left">O-</h1>
            <p style="font-weight: 200px;text-align: left;">Available donors - {{data.count.ON}}</p>
            <button class="numbtn" type="submit"
onclick="window.location.href='/plasmarequest?group=ON'">Request</button>
          </div>
        </div>
      </div>


    <script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"></script>
    </body>
</html>
```

#login.html

```
<!DOCTYPE html>


<html>
```

```
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Login</title>
  <meta name="description" content="">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="/static/css/login.css" />
</head>
<body>
  <div class="login_main">
    <div class="topnav" id="myTopnav">
      <a href="/register">Register</a>
      <a href="/login">Login</a>
    </div>
  <div class="login">
    <h1 class="fact">Be the reason for someone's heartbeat</h1>
    {{message}}
    <form  method="post">
      <div class="login-page">
      <h1>Login</h1>
      <input type="text" placeholder="Email" name="email"/>
      <input
        type="password"
```

```
          placeholder="password"
          name="password"/>
        <input type="submit" class="submit" value="Login"/>
      </div>
      </form>
    </div>
  </div>
  </body>
</html>
```

#profile.html

```
<!DOCTYPE html>


<html>
    <head>
        <title>Donor Details</title>
        <link rel="stylesheet" href="/static/css/dashboardstyle.css">
        <title> Dashboard </title>
        <title>Donor Details</title>


        <script
        src="https://code.jquery.com/jquery-3.3.1.js"
        src="jquery.js"
        integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60="
```

```
    crossorigin="anonymous">
    </script>
    </head>
<body>
    <div class="navigation" id="navbar">
        <nav class="nav">
            <div>
                <div class="nav-symbol">
                    <strong href="#" class="nav_logo">Dashboard</strong>
                </div>


                <div class="nav-list">
                    <a href="/profile" class="nav_link">
                        <ion-icon name="person-outline" class="nav_icon"></ion-icon>
                        <span class="nav_name" >Profile</span>

                    </a>
                    <a href="/" class="nav_link">
                        <ion-icon name="medical" class="nav_icon"></ion-icon>
                        <span class="nav_name" >Donors list</span>
                    </a>
                </div>
```

```
            </div>

            <a href="/logout" class="nav_link">
               <ion-icon name="log-out" class="nav_icon"></ion-icon>
               <span class="nav_name">Logout</span>
            </a>
         </nav>
      </div>


   <div class="card_3">
     <h1 class="profile_h1">Profile</h1>
     <div class="profile">
        <div class="align">
           <label><p><b>Name:  </b></p><p><b>{{resp.NAME}}</b></p></label>
        </div>
        <br>
        <div class="align">
           <label><p><b>Blood Group:
</b></p><p><b>{{resp.BLOODGROUP}}</b></p></label>

        </div>
        <br>
        <div  class="align">
```

```
            <label><p><b>Pincode:  </b></p><p><b>{{resp.PINCODE}}</b></p></label>
        </div>
        <br>
        <div  class="align">
            <label><p><b>Email: </b></p><p><b>{{resp.EMAIL}}</b></p></label>
        </div>

    </div>
  </div>
  <script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"></script>
  </body>
</html>
```

#register.html

```
<html>
  <head>
    <title>Donor Details</title>
    <link rel="stylesheet" href="/static/css/dashboardstyle.css">
    <title> Dashboard </title>
    <title>Donor Details</title>

    <script
    src="https://code.jquery.com/jquery-3.3.1.js"
    src="jquery.js"
```

```
    integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60="
    crossorigin="anonymous">
    </script>
    </head>
<body>
    <div class="navigation" id="navbar">
        <nav class="nav">
            <div>
                <div class="nav-symbol">
                    <strong href="#" class="nav_logo">Dashboard</strong>
                </div>


                <div class="nav-list">
                    <a href="/profile" class="nav_link">
                        <ion-icon name="person-outline" class="nav_icon"></ion-icon>
                        <span class="nav_name" >Profile</span>

                    </a>
                    <a href="/" class="nav_link">
                        <ion-icon name="medical" class="nav_icon"></ion-icon>
                        <span class="nav_name" >Donors list</span>
                    </a>
```

```
            </div>
          </div>

          <a href="/logout" class="nav_link">
            <ion-icon name="log-out" class="nav_icon"></ion-icon>
            <span class="nav_name">Logout</span>
          </a>
        </nav>
      </div>

    <div class="card_3">
      <h1 class="profile_h1">Profile</h1>
      <div class="profile">
        <div class="align">
          <label><p><b>Name:  </b></p><p><b>{{resp.NAME}}</b></p></label>
        </div>
        <br>
        <div class="align">
          <label><p><b>Blood Group:
</b></p><p><b>{{resp.BLOODGROUP}}</b></p></label>

        </div>
        <br>
```

```
        <div  class="align">
            <label><p><b>Pincode:  </b></p><p><b>{{resp.PINCODE}}</b></p></label>
        </div>
        <br>
        <div  class="align">
            <label><p><b>Email: </b></p><p><b>{{resp.EMAIL}}</b></p></label>
        </div>

    </div>
  </div>
  <script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"></script>
  </body>
</html>
```

#request.html

```
<html>
  <head>
    <title>Donor Details</title>
    <link rel="stylesheet" href="/static/css/dashboardstyle.css">
    <title> Dashboard </title>
    <title>Donor Details</title>

    <script
    src="https://code.jquery.com/jquery-3.3.1.js"
    src="jquery.js"
    integrity="sha256-2Kok7MbOyxpgUVvAk/HJ2jigOSYS2auK4Pfzbm7uH60="
```

```
    crossorigin="anonymous">
    </script>
    </head>
<body>
    <div class="navigation" id="navbar">
        <nav class="nav">
            <div>
                <div class="nav-symbol">
                    <strong href="#" class="nav_logo">Dashboard</strong>
                </div>


              <div class="nav-list">
                    <a href="/profile" class="nav_link">
                        <ion-icon name="person-outline" class="nav_icon"></ion-icon>
                        <span class="nav_name" >Profile</span>

                    </a>
                    <a href="/" class="nav_link">
                        <ion-icon name="medical" class="nav_icon"></ion-icon>
                        <span class="nav_name" >Donors list</span>
                    </a>
              </div>
            </div>

            <a href="/logout" class="nav_link">
                <ion-icon name="log-out" class="nav_icon"></ion-icon>
                <span class="nav_name">Logout</span>
            </a>
        </nav>
```

```
      </div>



     <div class="card_3">
      <h1 class="profile_h1">Profile</h1>
      <div class="profile">
         <div class="align">
            <label><p><b>Name:  </b></p><p><b>{{resp.NAME}}</b></p></label>
         </div>
         <br>
         <div class="align">
            <label><p><b>Blood Group:
</b></p><p><b>{{resp.BLOODGROUP}}</b></p></label>


         </div>
         <br>
         <div  class="align">
            <label><p><b>Pincode:  </b></p><p><b>{{resp.PINCODE}}</b></p></label>
         </div>
         <br>
         <div  class="align">
            <label><p><b>Email: </b></p><p><b>{{resp.EMAIL}}</b></p></label>
         </div>


      </div>
   </div>
   <script src="https://unpkg.com/ionicons@5.4.0/dist/ionicons.js"></script>
   </body>
</html>
```

#requestfail.html

```
<!DOCTYPE html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

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

    <title>Document</title>

    <style>

        body {

            text-align: center;

            display: flex;

            align-items: center;

            justify-content: center;

            font-family: "Nunito Sans", "Helvetica Neue", sans-serif;

            background: #eaaaaa;

        }

          h1 {

            color: #880808;

            font-weight: 900;

            font-size: 40px;

            margin-bottom: 10px;

          }
```

```
        p {
          color: #404F5E;
          font-size:20px;
          margin: 0;
        }
       pre{
        color: white;
        font-size: 100px;
        line-height: 200px;
       }
    </style>
</head>
<body>
   <body>
      <div class="card">
      <div style="border-radius:200px; height:200px; width:200px; background: #880808; margin:0 auto;">
        <pre class="checkmark">X</pre>
       </div>
        <h1>Failed</h1>
        <p style="margin-bottom: 25px;">The plasma request has not been submitted
          Try Again.</p>
        <a style="padding: 5px 5px;cursor:pointer;text-decoration: none;color:white;background-color:#880808;width: 90px;border-radius: 5px;" href="/">go back</a>
```

```
      </div>
    </body>
</body>
</html>
```

#requestsuccess.html

```
<html>
  <head>
    <link
href="https://fonts.googleapis.com/css?family=Nunito+Sans:400,400i,700,900&display=swap"
rel="stylesheet">
  </head>
    <style>
      body {
        text-align: center;
        display: flex;
        align-items: center;
        justify-content: center;
        font-family: "Nunito Sans", "Helvetica Neue", sans-serif;
        background: #c7f1d1;
      }
        h1 {
          color: #55ef79;
          font-weight: 900;
          font-size: 40px;
          margin-bottom: 10px;
        }
        p {
          color: #404F5E;
          font-size:20px;
          margin: 0;
```

```
      }
     i {
      color: white;
      font-size: 100px;
      line-height: 200px;
      margin-left:-15px;
     }


   </style>
   <body>
     <div class="card">
     <div style="border-radius:200px; height:200px; width:200px; background: #55ef79; margin:0 auto;">
       <i class="checkmark">✓</i>
     </div>
       <h1>Success</h1>
       <p style="margin-bottom: 25px;">The plasma request has been submitted successfully.</p>
       <a style="padding: 5px 5px;cursor:pointer;text-decoration: none;color:white;background-color:#880808;border-radius: 5px;" href="/">go back</a>
     </div>
   </body>
</html>
```

#app.py

```
import re
from urllib import request
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
import ibm_db
from flask import *
import datetime
```

```
sendGrid = SendGridAPIClient('<send grid api key>')



plasma_donor_chart = {
   "OP":('OP', 'ON'),
   "ON":('ON'),
   "AP":('OP', 'ON','AP','AN'),
   "AN":('ON','AN'),
   "BP":('OP', 'ON','BP','BN'),
   "BN":('ON','BN'),
   "ABP":('OP', 'ON','AP','AN','BP','BN','ABP','ABN'),
   "ABN":('ON','AN','BN','ABN')
}
group_abbreviation={
   "OP":"O POSITIVE",
   "ON":"O NEGATIVE",
   "AP":"A POSITIVE",
   "AN":"A NEGATIVE",
   "BP":"B POSITIVE",
   "BN":"B NEGATIVE",
   "ABP":"AB POSITVE",
   "ABN":"AB NEGATIVE"
}
plasma_group_list = ['OP', 'ON','AP','AN','BP','BN','ABP','ABN']


conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=2d46b6b4-cbf6-40eb-bbce-
6251e6ba0300.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32328;SECURITY=SSL;
```

```
SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=bmb02607;PWD=9qglrICwkM11bGTw;", '',
'')



app = Flask(__name__)
app.secret_key = "uafabdfouabdoibaidnaou"
def sendMail(to_email,subject,content):
    message = Mail(
    from_email='rishikam.19cse@kongu.edu',
    to_emails=to_email,
    subject=subject,
    html_content=content)
    try:
        response = sendGrid.send(message)
        print(response.status_code)
        # print(response.body)
        # print(response.headers)
    except Exception as e:
        print(e.message)


def getCount(pincode):
    count_list ={}
    for group in plasma_group_list:
        count_query = 'select count(*) from userdata where bloodgroup = ? and pincode between
? and ?'
        fetch_count = ibm_db.prepare(conn,count_query)
        # print(group)
        ibm_db.bind_param(fetch_count, 1,group)
```

```
        ibm_db.bind_param(fetch_count, 2,int((int(pincode/1000))*1000))
        ibm_db.bind_param(fetch_count, 3,int((int(pincode/1000))*1000)+1000)
        ibm_db.execute(fetch_count)
        count = ibm_db.fetch_assoc(fetch_count)
        count_list[group] = count['1']
    print(count_list)
    return count_list


def getEmail(group,pincode,contact):
    count_query = 'select email from userdata where bloodgroup = ? and pincode between ? and
?'
    fetch_count = ibm_db.prepare(conn,count_query)
    print(group,pincode,contact)
    ibm_db.bind_param(fetch_count, 1,group)
    ibm_db.bind_param(fetch_count, 2,int((int(pincode/1000))*1000))
    ibm_db.bind_param(fetch_count, 3,int((int(pincode/1000))*1000)+1000)
    ibm_db.execute(fetch_count)
    email = ibm_db.fetch_assoc(fetch_count)
    while email!= False:
        mailBody = "plasma donation has been requested for bloodgroup
"+group_abbreviation[group]+ "\ncontact\nName: "+ contact['name'] + "\ne-mail:
"+contact['email']+ "\Phone Number: "+contact['phone']+ "\naddress: "+contact['address']+
"\naddress: "+str(pincode)
        sendMail(email['EMAIL'],"Requesting for Plasma Donation",mailBody)
        print(email)
        email = ibm_db.fetch_assoc(fetch_count)
    return email


def getUserData(email):
    sql =  "SELECT * FROM userdata WHERE email = ?"
```

```
    userdatastatement = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(userdatastatement, 1, email)
    ibm_db.execute(userdatastatement)
    user_details = ibm_db.fetch_assoc(userdatastatement)
    return user_details



@app.route("/login", methods=['POST', 'GET'])
def login():
    print(session.get('loggedin'))
    if session.get('loggedin')!=None:
        return redirect(url_for('.dash'))
    if request.method == "POST":
        email = request.form['email']
        password = request.form['password']
        print(email)
        sql =  "SELECT email FROM Users WHERE email = ? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print("acc "+str(account))
        if account:
            print('login')
            session['loggedin'] = True
            session['id'] = account['EMAIL']
```

```
            sendMail(email,"login detected","login detected on some random device with ip ")
            return redirect(url_for('.dash'))
        else :
            return render_template('login.html',message="user not found or invalid credentials")
    else:
        return  render_template('login.html')




@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        name=request.form.get('name')
        bloodGroup = request.form.get('group')
        pincode = request.form.get('pincode')
        print(name)
        email=request.form.get('email')
        password=request.form.get('password')
        date = request.form.get('lastdonated')
        dateSplit = date.split("-")
        lastdonated = datetime.datetime(int(dateSplit[0]), int(dateSplit[1]), int(dateSplit[2]), 0, 0,
0).timestamp()
        lastdonated = int(lastdonated)
        sql = "SELECT * FROM users WHERE email = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.execute(stmt)
        account=ibm_db.fetch_assoc(stmt)
```

```
        print(account)
        if account:
            msg="Account already exists!"
            return render_template('register.html',resp=msg)
        if not re.match(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b',str(email)):
#1234@gmail.com
            msg = "Invalid email address" + str(email)
            return render_template('register.html',resp=msg)
        if not (request.form.get('group') and request.form.get('pincode') and
request.form.get('lastdonated') and request.form.get('email') and request.form.get('name') and
request.form.get('password')):
            msg = "fill all the fields"
            return render_template('register.html',resp=msg)
        else:
            try:
                insert_user_table="INSERT INTO users VALUES ( ?, ?)"
                user_create = ibm_db.prepare(conn,insert_user_table)
                ibm_db.bind_param(user_create, 1, email)
                ibm_db.bind_param(user_create, 2, password)
                ibm_db.execute(user_create)
                insert_userdata_table = "INSERT INTO USERDATA VALUES (?,?,?,?,?)"
                user_data_create = ibm_db.prepare(conn,insert_userdata_table)
                print(email,name,bloodGroup,pincode,lastdonated)
                ibm_db.bind_param(user_data_create, 1, email)
                ibm_db.bind_param(user_data_create, 2, name)
                ibm_db.bind_param(user_data_create, 3, bloodGroup)
                ibm_db.bind_param(user_data_create, 4, int(pincode))
                ibm_db.bind_param(user_data_create, 5, int(lastdonated))
                ibm_db.execute(user_data_create)
                msg="You have successfully registered"
```

```
            # sendMail(email,"registered successfully",msg)
            print(msg)
        except:
            render_template('register.html',msg="error")
        return redirect(url_for('.login'))

    return render_template('register.html')



@app.route('/',methods=['GET'])
def dash():
    if session.get('loggedin')==None:
        return redirect(url_for('.login'))


    userData= getUserData(session['id'])
    userData['PINCODE']=request.args.get('pincode',type=int) if
request.args.get('pincode',type=int) else userData['PINCODE']
    count = getCount(userData['PINCODE'])
    return
render_template('homepage.html',data={'status':True,"user":userData,"count":count})
    # except:
    #     print("error")
    #     return render_template('.html',data={'status':False})


# @app.route('/request/<blood>', methods=['GET'])
# def requestForm(blood):
#     return render_template('request.html',})
```

```
@app.route('/plasmarequest',methods=['GET','POST'])
def requestPlasma():
    if session.get('loggedin')==None:
        return redirect(url_for('.login'))
    if request.method == 'GET':
        return render_template('request.html',blooddata={"status": True,
"data":{"blood":request.args.get('group')}})
    if not (request.form.get('group') and  request.form.get('pincode')
    and request.form.get('address') and request.form.get('email') and request.form.get('name')
and request.form.get('phone')):
        return render_template('requestfail.html')
    contact = {
    "address":request.form.get('address'),
    "email" :request.form.get('email'),
    "name":request.form.get('name'),
    "phone":request.form.get('phone'),
    }
    bloodGroup = request.form.get('group')
    pincode = request.form.get('pincode',type=int)
    print(bloodGroup)
    email_list = getEmail(bloodGroup,pincode,contact)
    print("list"+str(email_list))
    return render_template('requestsuccess.html')


@app.route('/profile',methods=['GET'])
def profile():
    if session.get('loggedin')==None:
        return redirect(url_for('.login'))
    try:
```

```
        user_data = getUserData(session.get('id'))
        return render_template('profile.html',resp=user_data)
    except:
        return render_template('requestfail.html')
@app.route('/logout')

def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    return redirect(url_for('.login'))



if __name__== "__main__":
    app.run('0.0.0.0',port=5001)
```

10. Advantages:

- Speed This website is fast and offers great accuracy as compared to manual registered keeping.
- Maintenance Less maintenance is required
- User Friendly It is very easy to use and understand.
- It is easily workable and accessible for everyone.
- Fast Results It would help you to provide plasma donors easily depending uponthe availability of it.

Disadvantages:

- Internet It would require an internet connection for the working of the website.
- Auto- Verification It cannot automatically verify the genuine users.

11. Conclusion:

Plasma donor application provides a reliable platform to connect local blood donors with patients. Plasma donor creates a communication channel through websites whenever a patient needs blood 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. This Application is used to find donors very easily and fastly comparedto manual searching.

GitHub Link:

https://github.com/IBM-EPBL/IBM-Project-23949-1659934186

Project Demo Link:
https://drive.google.com/file/d/1LujEUKP-rIW2XJDhcJnrEy7i0zs- 2pfd/view?usp=share_link