PLASMA DONOR APPLICATION

USING CLOUD

A Project report submitted in partial fulfilment of 7th semester in degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AD ENGINEERING

Submitted by

Team ID: PNT2022TMID51971

Sabrish Deepak R 962319104075

Pranesh P 962319104069

Shalini V 962319104080

Sree Lakshmi K J 962319104086

DEPARTMENT OF COMPUTER SCIENCEAND ENGINEERING

AMRITA COLLEGE OF ENGINEERING AND TECHNOLOGY

ANNA UNIVERSITY: CHENNAI 600025

NOV-2022

AMRITA COLLEGE OF ENGINEERING AND TECHNOLOGY

(A Constituent Collegeof Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report ”**PLASMA DONOR APPLICATION”** is the bonafide record work done by **Mr SABRISH DEEPAK R**(**962319104075**), **Mr PRANESH P** (**962319104069**), **Ms SHALINI V** (**962319104080**) and **Ms SREE LAKSHMI K J** (**962319104086**)for **IBM-NALAIYATHIRAN** in **VII** semester of **B.E.,** degree course **in ComputerScience and Engineering** branch during the academic year of 2022 - 2023.

Staff-Incharge
Mrs Karpagavalli C

Evaluator
Mrs JothiLakshmi S L

Head of the Department
Dr. P M Siva Raja

ACKNOWLEDGEMENT

We express our breathless thanks to our **Dr.T.Kannan, M.E., Ph.D.,** the principal, Amrita College of Engineering and Technology, Erachakulam, for giving constant motivation in succeeding in our goal.

We acknowledge our sincere thanks to Head of the Department (i/c) **Dr. P M Siva Raja M.Tech., Ph.D.,** for giving us valuable suggestion and help towards us throughout thisProject.

We are highly grateful to thank our Project coordinator **Mrs Karpagavalli C,** and our Project Evaluator **Mrs JothiLakshmi S L,** Department of Computer Science and Engineering, Amrita College of Engineering and technology,for the coordinating us throughout this Project.

We are very much indebted to thank all the faculty members of Department of Computer science and Engineering in our Institute, for their excellent moral support and suggestions to complete our Project work successfully.

Finally our acknowledgment does our parents, sisters and friends those who had extended their excellent support and ideasto make our Project a pledge one.

Sabrish Deepak R
Pranesh P
Shalini V
Sree Lakshmi K J

PROJECT REPORT FORMAT

1.INTRODUCTION

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 secrect 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 of Admin and User where both can request for a Plasma.

- Both party scan 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

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 servergoes 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 interactwith this application.

2.2 References:

YEAR	TITLE	AUTHOR(s)	TECHNIQUE(s)	PROS	CONS
2022	Instant Plasma Donor Recipient connector web application	Kalpana Devi Guntoju, Tejaswini Jalli, Sreeja Uppala, Sanjay Mallisetti	Web Technologies, API, Database	The Donor needs to upload their recovered COVID-19 Certificate and it required to verifiedby the blood bank.It is a user- friendly application. It will help people to find plasma easily.	This is system is closed for general plasma donation and mainly focused on COVID-19 patients for plasma donation
2021	BDonor App-Blood Donation Application using Android Studio	S Periyanayagi, A Manikandan,M Muthukrishnan,and M Ramakrishnan	Android, FlutterUI, Dart, Firebase, Decision tree algorithm	The Donor details are verified before they allow to donate and have to authorised by institution. The Verification and validation are done in Email base.	The android mobile user will not be able to insert or view details if the server goes down. Thus, there is disadvantage of single point failure.

2020	Lifesaver E-Blood Donation App Using Cloud	Rishab Chakrabarti, Asha Darade, Neha Jadhav, Prof. S. M. Chitalkar	E-health, GPS, Blood bank database, Cloud Computing	Reduction in the errors ofblood bank using most eligible donor method. Direct Communication Between donor and the person in need of blood During the Emergency situation.	The user given detailsare maintained unverified.
2020	Developing a plasma donor application using Function-as-a-service in AWS	Aishwarya R Gowri	Serverless, aws, plasma theory, covid19, dynamoDB, cloud	The efficient way of findingplasma donor for the infected people. Aws lambda function is used and to deploy the application AWS EC2 service is used.	The user interface canbe better than now.
2019	D'WORLD: Blood Donation App Using Android	A. Meiyappan, K. Loga Vignesh, R. Prasanna, T. Sakthivel	Android, Global Positioning System (GPS), Mobile Computing	When the giver gives the blood, it will naturally evacuate the contributor detail for next three months.It additionally confirms with the Department of Health and Welfare to guarantee the benefactor medical case history.	The user must have an device with android operating system with an active internet connection to interact with this application.

2018	Automated blood bank system using Raspberry PI	Ashlesha C. Adsul, V. K. Bhosale, R. M. Autee	Raspberry Pi, Embedded Blood Bank, GSM, Android	When there is urgent need for blood then If this model is adopted the caller is immediately connected to the donor	Tackling the fakeusers.

2.3 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-friendlyapplication 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.

3.IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas:

3.2 Ideation & Brainstorming:

3.3 Proposed Solution:

S.NO	PARAMETERS	DESCRIPTION
1.	Problem statement (problems to be solved	To help the plasma donor and seeker by developing a cloud-based application.
2.	Idea/solution description	In day-to-day life requirement for plasma became high, especially during the COVID-19 crisis. But the donor count was low. Saving the donor information and helping the needy by notifying the current donors would be a helping hand. It is very difficult to find the respective blood group donors when anyone is in need. Regarding the problem faced, an application is to be built which would take the donor details store them and inform them upon request. And also for plasma donation centre, it is Easy to find donors.
3.	Novelty/Uniqueness	We help the donor to access the location ofa blood centre which is nearby him/her. We Notify them by sending a confirmation emails after they get registered for the plasma donation and alsowe notify them once the appointment is fixed in the centre. Further , more the GPS map option is available to direct.
4.	Social Impact/Customer satisfaction	By using this application, the user will experience a user-friendly and responsive interface and they get satisfaction by Saving thousand so people's life.
5.	Business Model (Revenue)	Donating Plasma with the help of an application makes our idea realistic. The user's information is encrypted. We maintain this app by automation for saving admin and user time. Users get profited as we take care of them even after the plasma donation by giving them hospitality details. Also, we use the Chabott answer FAQs ,asset helps the user to get immediate Answer to their doubts.
6.	Scalability of the solution	Whatever the requirements, the application provides a clear solution for the requirements. It can handle more users who use the application at the same time

3.4 Problem Solution fit:

roblem-Solution Fit canvas

1. CUSTOMER SEGMENT(S)
Anyone above the age of 21 can donate.We working on plasma therapy is process where blood is donated and received

6. CUSTOMER LIMITATIONS
You can donate plasma every 28 days, up to 13 times per year. While the FDA does not allow donors to give plasma more frequently. Limited no of users can use it at the same time.

5. AVAILABLE SOLUTIONS
It allows people to help others
It is a relatively safe process
The process can be very uncomfortable and it depletes the calcium levels in the body

2. PROBLEMS / PAINS
The side effects of plasma donation include nausea and dizziness and fainting in some cases. You may develop a raised bump or experience continued bleeding and bruising at the needle site too. Some people might experience pain and physical weakness after donating plasma.

9. PROBLEM ROOT / CAUSE
Localized allergic reaction
Air embolism and Hemolysis
Bruising and discomfort

7. BEHAVIOR
This app is used to make donation and receiving process easier so that anyone can easily access and use it. Intensity of this application is to connect donor and receiver in single platform. donor can fill the interest form to donate.

3. TRIGGERS TO ACT
Many people needs plasma for their treatment.Plasma donation really used for covid affected people for recovering faster.

4. EMOTIONS
Donor get fear,anxiety prior to donation give way to largely positive emotional states like relaxation following donation

10. YOUR SOLUTION
our app allows the user to request and donate plasma to requested person. Receiver can directly contact the donor and receive plasma. When you donate plasma, the blood that's drawn from your arm goes through a special machine to separate the different parts of your blood. Then we get plasma which can be used for transfusion.

8. CHANNELS of BEHAVIOR
Online app allows user to make donation and receiver process easier.send request from anywhere anytime.

users to visit nearby camp or hospital and donate as well as receive plasma.

4.REQUIREMENT ANALYSIS

4.1 Functional requirement:

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 (WebApp)
FR-2	User Confirmation	Confirmation via EmailConfirmationvia OTP
FR-3	Certification	After the donor donates plasma, we will give them acertificate of appreciation and authentication.
FR-4	Statistical data	The availability of plasma is given in the page as stats, which will be helpful for the users.
FR-5	User Plasma Request	Users can request to donate plasma by filling out therequest form on the page. Once the request is submitted, they will get an email
FR-6	Searching/reportingrequirements	Users can use the search bar to look up informationabout camps and other topics.
FR-7	Virtual Assistants	A virtual assistant is a software agent that can carry out tasks or provide services on behalf of a person in response to commands or inquiries. When users enter their inquiries, the system will respond with pertinent information about plasma anddetails of plasma donation.

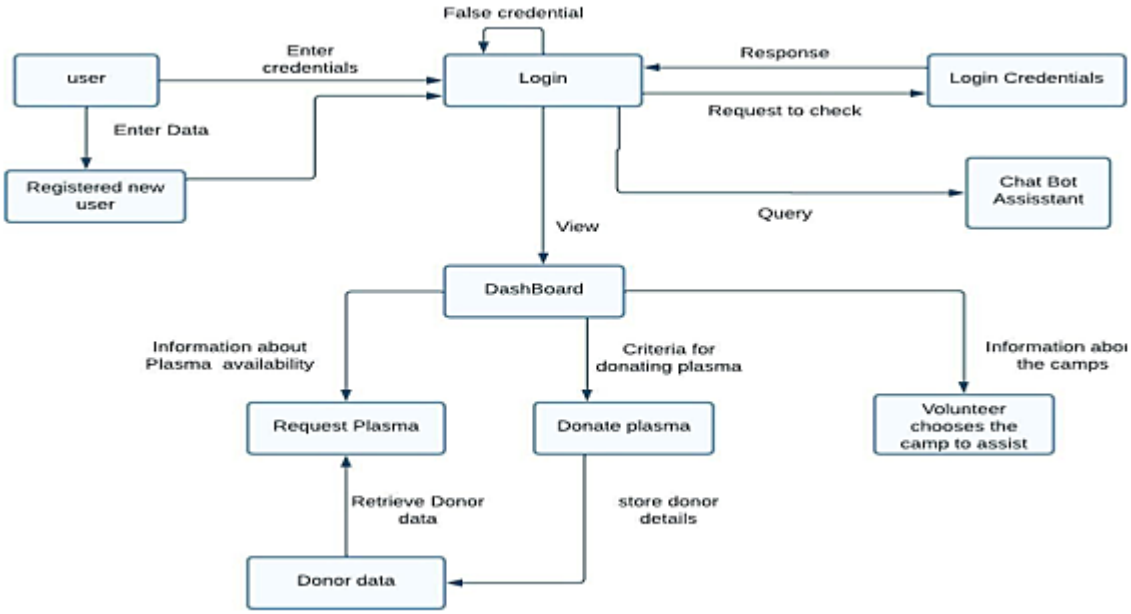
4.2 Non-functional requirement:

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

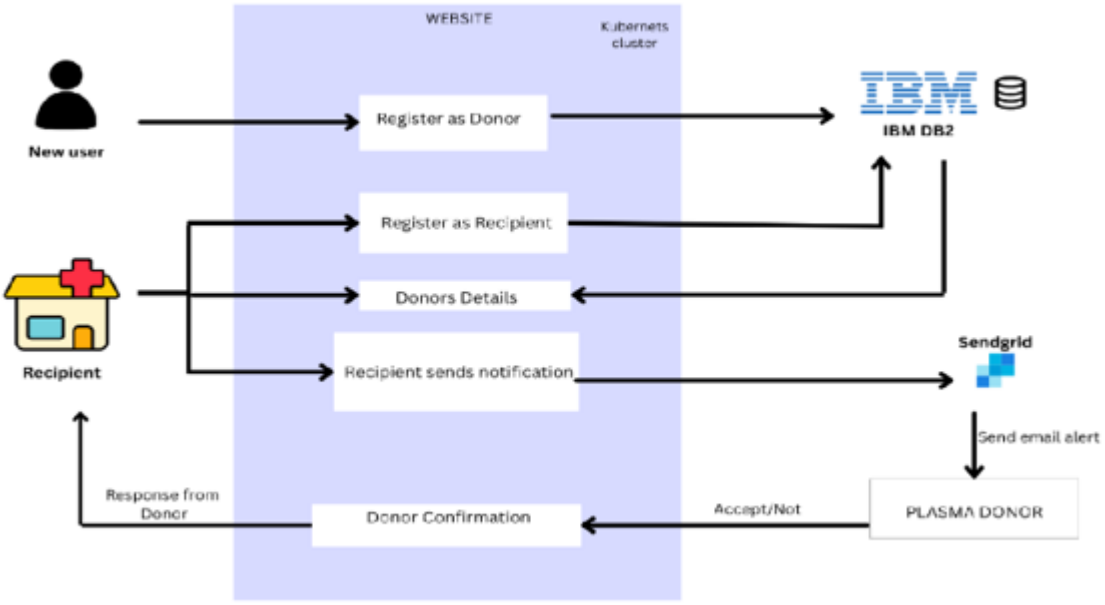
NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	Must have a good-looking User-friendly interface.
NFR-2	Security	It must be secured with the proper username and password.
NFR-3	Reliability	The system should be made in such a way that it is reliable in its operations and for securing the sensitive details.

5.PROJECT DESIGN

5.1 Data Flow Diagram:



5.2 Solution & Technical Architecture:



5.3 User Stories:

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 Gmail	I can receive confirmation notifications through Gmail	Medium	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password	I can access into my User profile and view details in dashboard	High	Sprint-1
	Dashboard	USN-5	As a user,I can send the proper requests to donate and obtain plasma.	I can receive appropriate notifications through email	High	Sprint-1
Customer (Web user)	Login	USN-6	As a user,I can register and log into the application by entering email & password to view the profile	I can access into my User profile and view details in dashboard	High	Sprint-1
	Dashboard	USN-7	As a user,I can send the proper requests to donate and obtain plasma.	I can receive appropriate notifications through email	High	Sprint-1
Customer Care Executive	Application	USN-8	As a customer care executive,I can try to address user's concerns and questions	I can view and address their concerns and questions	Medium	Sprint-2
Administrator	Application	USN-9	As an administrator I can help with user-facing aspects of a website, like its appearance, navigation and use of media.	I can change the appearance and navigation in a user friendly manner	Medium	Sprint-3
		USN-10	As an administrator, I can involve working with the technical side of websites.	I can help with such as troubleshooting issues, setting up web hosts, ensuring users have access and programming servers	Medium	Sprint-1

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Chatbot	Dashboard	USN-11	In addition the Customer care executive,chatbot can try to address user's concerns and questions	I can reply to all the queries related to our application	Medium	Sprint-3

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation:

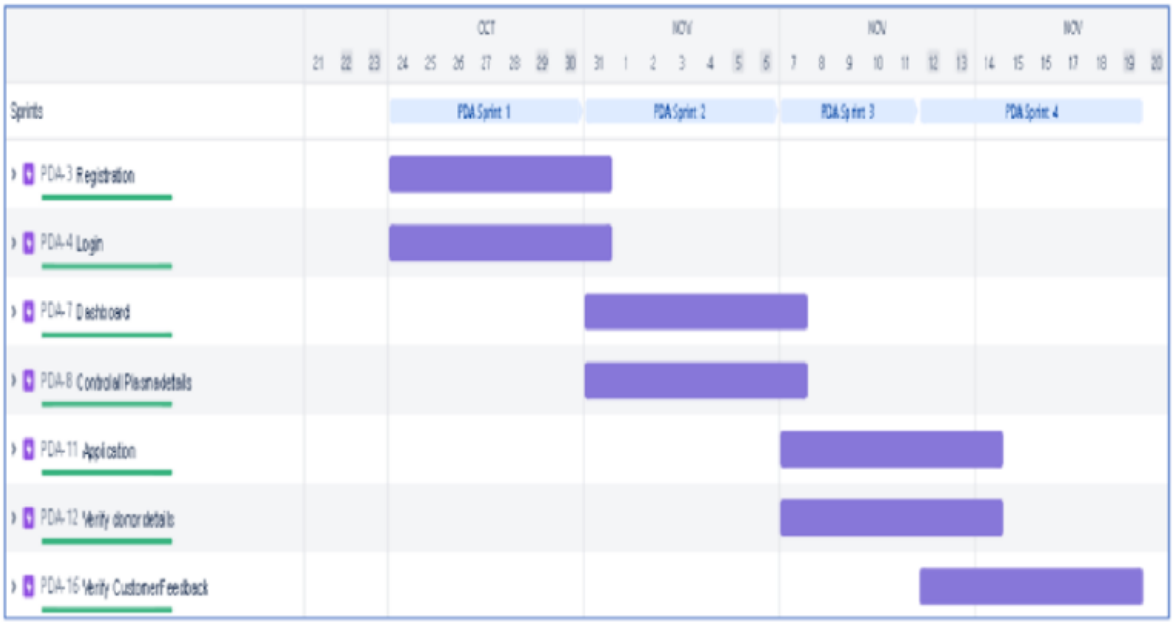
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	4
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	4
Sprint-2		USN-3	As a user, I can register for the application through Facebook	2	Low	4
Sprint-1		USN-4	As a user, I can register for the application through Gmail	2	Medium	4
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	1	High	4
Sprint-3	Dashboard	USN-6	As a user, I can find the compatible donor by registering.	3	High	4
Sprint-3		USN-7	As a user, I can find the donor availability by logging in.	3	High	4
Sprint-2		USN-8	As a user, I can create a profile by registering.	2	Medium	4
Sprint-3		USN-9	As a user, I can see the demand of plasma.	3	Medium	4
Sprint-4	Database	USN-10	As a user, I can store the availability and need of plasma information value.	4	High	4

6.2 Sprint Delivery Schedule:

Project Tracker, Velocity & Burndown Chart: (4 Marks)

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 Jira:



7.CODING & SOLUTIONING

7.1 Feature 1:

Login.html:

- The loginpage allows a user to gain access to the application by entering theirusername and password.
- There are two possible results during login :
  - Authentication is successful and the user is directedto the landing page
  - Authentication fails and the user remains on the login page. If authentication fails, the screen should show an informational or error message about the failure.

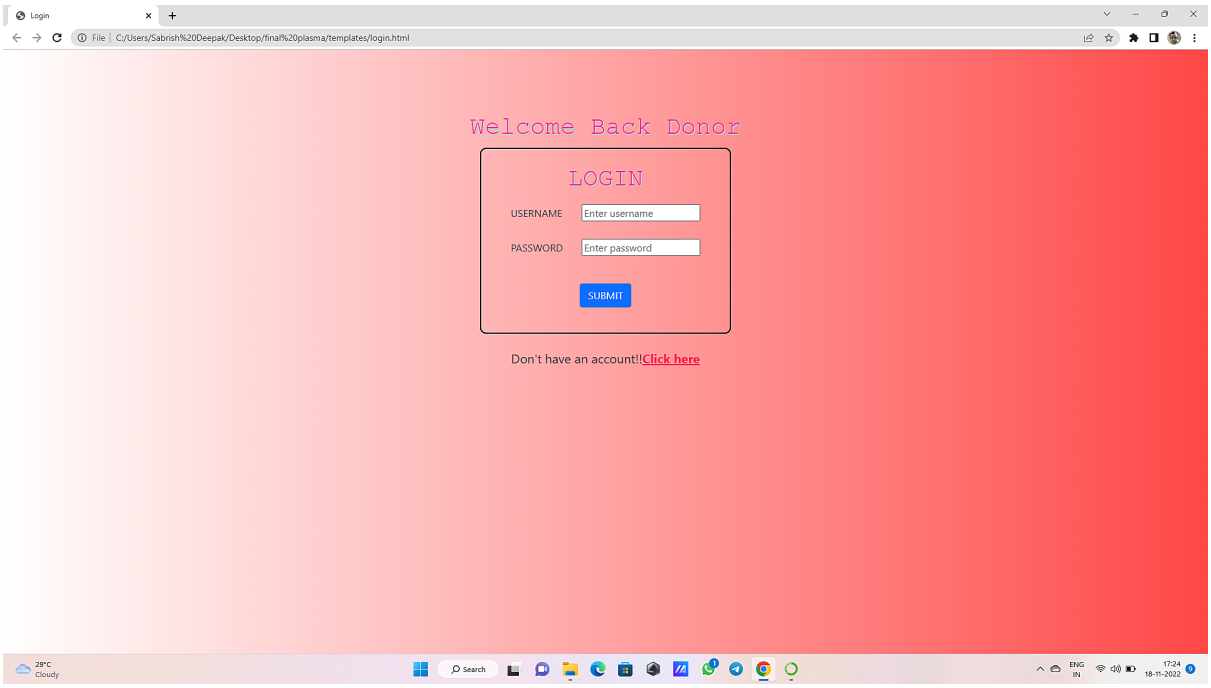
```
<html lang="en">


<head>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">


    <meta charset="UTF-8">
```

```
8.      <meta http-equiv="X-UA-Compatible" content="IE=edge">
9.      <meta name="viewport" content="width=device-width, initial-scale=1.0">
10.     <title>Login</title>
11. </head>
12.
13. <style>
14.
15.     body {
16.         background-image: linear-gradient(to right, rgba(255, 255, 255, 0.726), rgba(255, 0, 0,
    0.719));
17.
18.     }
19.     button:hover {
20.         background-color: rgb(223, 241, 224);
21.         border-color: black;
22.     }
23.
24.     h1 {
25.         font-family: 'Courier New', Courier, monospace;
26.         text-shadow: 2px 2px #bacfcf;
27.         color: rgb(252, 10, 159);
28.         top: 10em;
29.     }
30.
31.     .container1 {
32.         border: 2px solid black;
33.         border-color: black;
```

```
34.     border-radius: 10px;
35.     width: 400px;
36. }
37.
38. .top {
39.     margin-top: 100px;
40. }
41.
42. input:hover {
43.     border-color: rgb(25, 20, 20);
44. }
45.
46. a:link {
47.     color: #ff0256;
48.
49. }
50.
51. a:visited {
52.     color: rgb(226, 10, 129);
53.
54. }
55.
56. a:hover {
57.     color: rgb(128, 105, 255);
58.
59. }
60.
```

```
    a:active {
        color: rgb(226, 10, 129);
        text-decoration: none;
    }
    table, th, td {
  border: none;
  border-collapse: collapse;
}
th, td {
  padding-top: 8px;
  padding-bottom: 20px;
  padding-left: 15px;
  padding-right: 15px;
}
</style>

<body>
    <center>
        <h1 class="top">Welcome Back Donor</h1>
        <div class="container1">
            <br>
            <h1>LOGIN</h1>
            <form action="{{ url_for('login') }}" method="POST">
                <table >
                    <tr>
```

```
                    <td><label for="text">USERNAME  </label></td>
                    <td><input type="text" name="username1" placeholder="Enter username" /></td>
                </tr>


                <tr >
                    <td ><label for="text">PASSWORD  </label></td>
                    <td><input type="text" name="password1" placeholder="Enter password"></td>
                </tr>
            </table>
            <br>
            <button class="btn btn-primary" type="submit">SUBMIT</button>
        </form>
        <br>
                </div>
                <label style="font-size:20px;" for="text">Don't have an account!!<b ><a  href="{{ url_for('register') }}">Click here</a></b></label>
            </center>
        ' </body>

        </html>
```



7.2 Feature 2:

Register.html :

- A registration page enables users and organizations to independently registerand gain access to the system.
- It is common to have multiple signup pages depending on the types ofpeople and organizations you want to register.

```
<!DOCTYPE html>
<html>

<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>REGISTRATION PAGE</title>
```

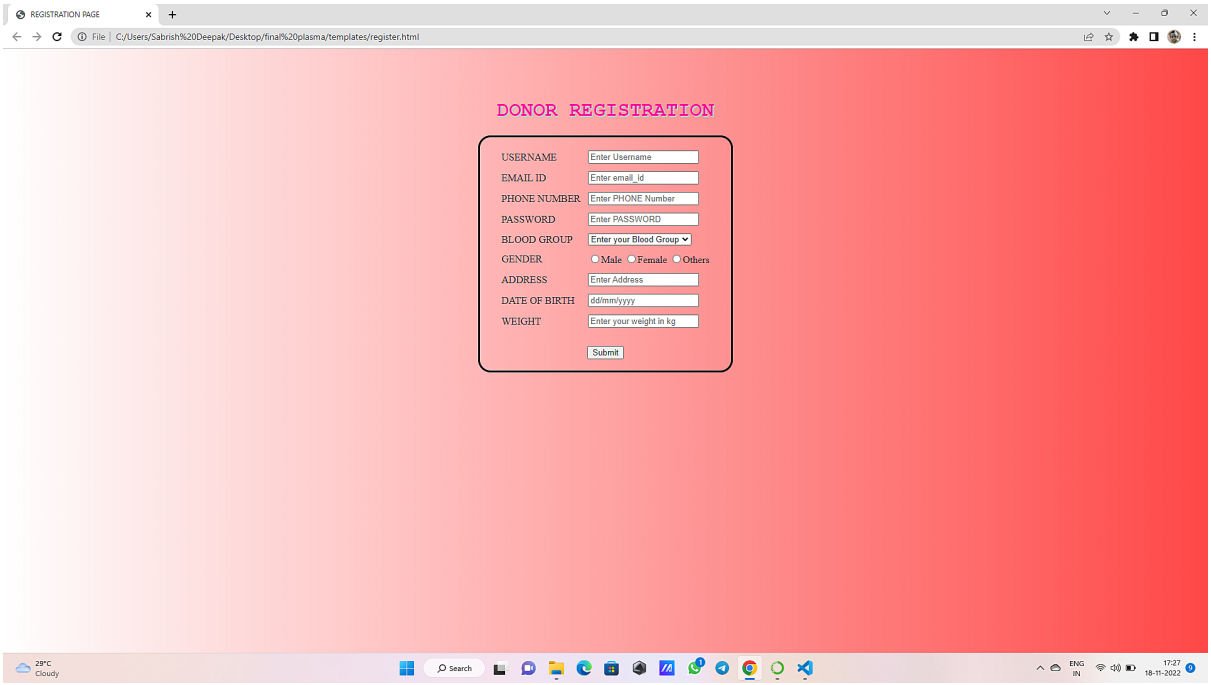
```
</head>
<style>
  body {
    background-image: linear-gradient(to right, rgba(255, 255, 255, 0.726), rgba(255, 0, 0, 0.719));
  }

  input:hover {
    border-color: rgb(25, 20, 20);
  }
  .container1 {
    border: 3px solid black;
    border-color: black;
    border-radius: 20px;
    width: 400px;
  }

  button:hover {
    background-color: darkgray;
    border-color: black;
  }

  h1 {
    font-family: 'Courier New', Courier, monospace;
    text-shadow: 2px 2px #bacfcf;
    color: rgb(252, 10, 159);
  }

  #qwerty {
    margin-top: 5em;
  }
</style>

<body>
  <center id="qwerty">
    <H1>DONOR REGISTRATION</H1>
   <div class="container1">
    <form action="http://localhost:5000/register" method="POST">
        <table>
          <tr>
            <td><label for="text">USERNAME</label></td>
            <td> </td>
            <td><input type="text" placeholder="Enter Username" name="username"
id="username"></td>
          </tr>
          <tr></tr>
```

```
            <tr></tr>
            <tr></tr>
            <tr></tr><br>
            <tr>
               <td><label for="text">EMAIL ID</label></td>
               <td> </td>
               <td><input type="text" placeholder="Enter email_id" name="email_id" id="email_id"></td>
            </tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr>
               <td><label for="text">PHONE NUMBER</label></td>
               <td> </td>
               <td><input type="text" placeholder="Enter PHONE Number" name="phone_no"
id="phone_no"></td>
            </tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr>
               <td><label for="text">PASSWORD</label></td>
               <td> </td>
               <td><input type="text" placeholder="Enter PASSWORD" name="password"
id="password"></td>
            </tr>

            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr>
               <td><label for="text">BLOOD GROUP</label></td>
               <td> </td>
               <td><select id="blood_grp" name="blood_grp">
                  <option value="null">Enter your Blood Group</option>
                  <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>
               </select>
               </td>
            </tr>
```

```
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr>
              <td><label>GENDER</label></td>
              <td> </td>
              <td><label><input type="radio" name="gender" value="Male">Male</label>
              <label><input type="radio" name="gender" value="Female">Female</label>
              <label><input type="radio" name="gender" value="Others">Others</label></td>
            </tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr>
              <td><label for="text">ADDRESS</label></td>
              <td> </td>
              <td><input type="text" placeholder="Enter Address" name="address" id="address"></td>
            </tr>

            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr>
              <td><label for="text">DATE OF BIRTH</label></td>
              <td> </td>
              <td><input type="text" placeholder="dd/mm/yyyy" name="dob" id="dob"></td>
            </tr>

            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>

            <tr>
              <td><label for="text">WEIGHT</label></td>
              <td> </td>
              <td><input type="text" placeholder="Enter your weight in kg" name="weight"
id="weight"></td>
            </tr>

            <tr></tr>
            <tr></tr>
            <tr></tr>
            <tr></tr>
          </table>
          <br>
          <center><button onclick="asd()" type="submit">Submit</button>
```

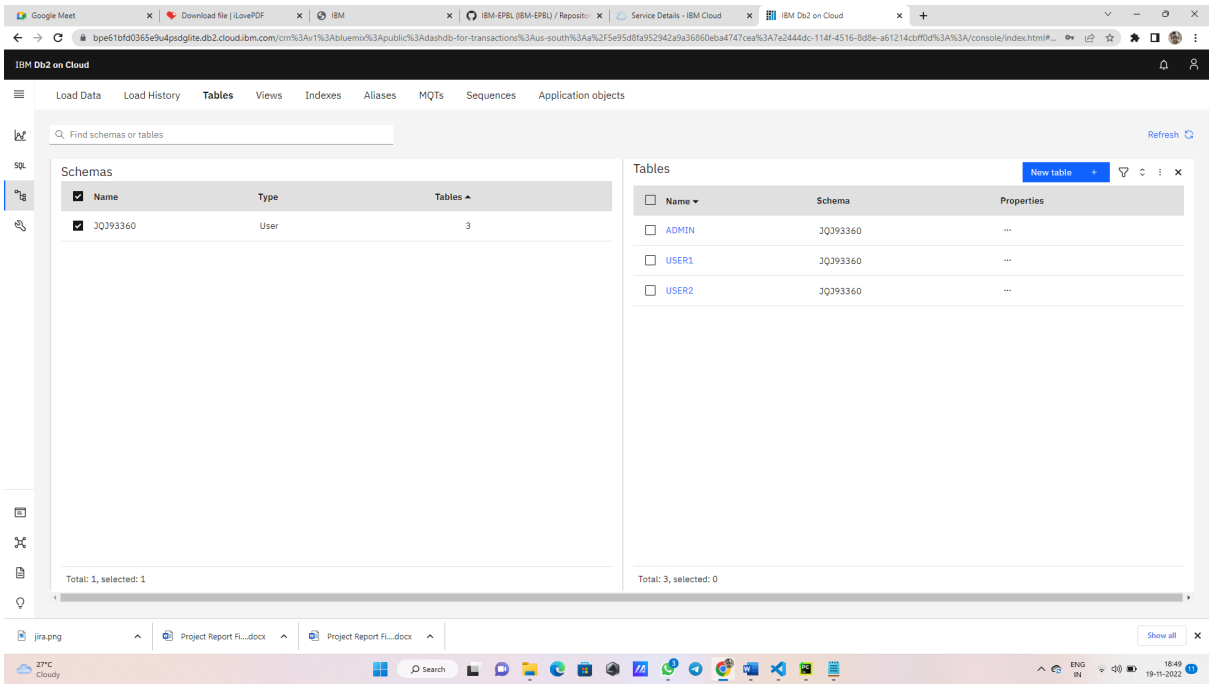
```
        </center>
      </form><br>
      </div>
   </center>
</body>
<script>
   function asd() {
      var username1 = document.getElementById("username");
      var email_id = document.getElementById('email_id');
      var phone_no = document.getElementById('phone_no');
      var password = document.getElementById('password');
      var blood_grp = document.getElementById('blood_grp');
      var gender = document.getElementById('gender');
      var address = document.getElementById('address');
      var address = document.getElementById('dob');
      var weight = document.getElementById('weight');
      if (username1.value == "" || phone_no.value == "" || password.value ==
""||BloodGroup.value=="null"||gender.value==""||Address.value=="") {
         username.style.borderColor = "red";
      }
      else if (email_id.value == "") {
         email_id.style.borderColor = "red";
      }
      else if (phone_no.value == "") {
         phone_no.style.borderColor = "red";
      }
      else if (phone_no.value == "") {
         password.style.borderColor = "red";
      }
       else if (BloodGroup.value == "null") {
         BloodGroup.style.borderColor = "red";
      }
      else if (value.gender=="") {
         gender.style.borderColor = "red";
      }
       else if (Address.value == "") {
         Address.style.borderColor = "red";
      }


   }


</script>
```

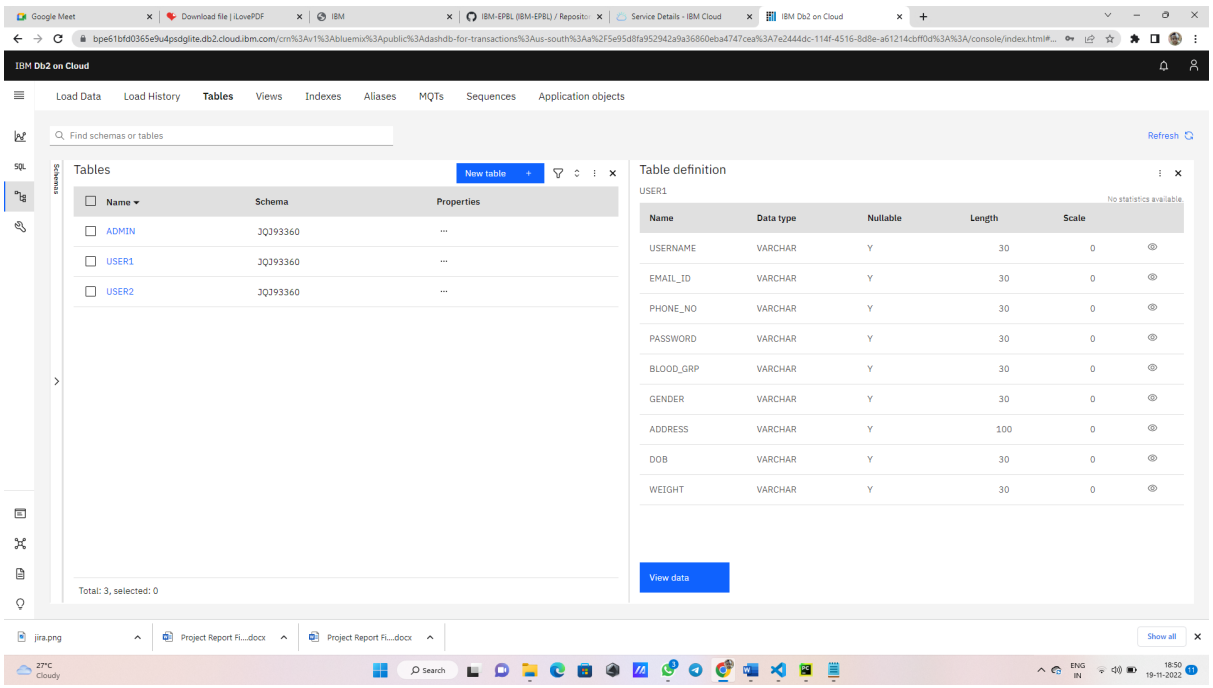
```
</html>
```



7.3 Database Schema:

A database schema defines how data is organized within a relational database; this is inclusive of logical constraints such as, table names, fields, data types, and the relationships between these entities. Schemas commonly use visual representations to communicate the architecture of the database, becoming the foundation for an organization’s data management discipline. This process of database schema design is also known as data modeling.These data models serve a variety of roles, such as databaseusers, database administrators, and programmers. A database schema is considered the “blueprint” of a database which describes how the data may relate to other tables or other data models. However, the schema doesnot actually contain data


IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Schemas
Name
Type
Tables
JQJ93360
User
3
Total: 1, selected: 1
Tables
New table
Name
Schema
Properties
ADMIN
JQJ93360
USER1
JQJ93360
USER2
JQJ93360
Total: 3, selected: 0


IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Tables
New table
Name
Schema
Properties
ADMIN
JQJ93360
USER1
JQJ93360
USER2
JQJ93360
Total: 3, selected: 0
Table definition
USER1
No statistics available.
Name
Data type
Nullable
Length
Scale
USERNAME
VARCHAR
Y
30
0
EMAIL_ID
VARCHAR
Y
30
0
PHONE_NO
VARCHAR
Y
30
0
PASSWORD
VARCHAR
Y
30
0
BLOOD_GRP
VARCHAR
Y
30
0
GENDER
VARCHAR
Y
30
0
ADDRESS
VARCHAR
Y
100
0
DOB
VARCHAR
Y
30
0
WEIGHT
VARCHAR
Y
30
0
View data

8.TESTING

8.1 Test Cases:

A test case has components that describe input, action and an expected response, in order to determine if a feature of an application is working correctly. A test case is a set of instructions on “HOW” to validate a particular test objective/target, which when followed will tell us if the expected behavior of the system is satisfied or not.

Characteristics of a good test case:

- Accurate: Exacts the purpose.
- Economical: No unnecessary steps or words.
- Traceable: Capable of being traced to requirements.
- Repeatable: Can be used to perform the test over and over.
- Reusable: Can be reused if necessary.

S.NO	Scenario	Input	Excepted output	Actual output
1	Admin Login Form	User name and password	Login	Login success.
2	Donor Registration Form	Donor basic details	Registration	Donor registration details stored in database.
3	User Registration Form	User basic details	Registration	User registration details stored in database.
4	User Login Form	User name and password	Login	Login success.

8.2 User Acceptance Testing

- 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	10	4	2	3	20
Duplicate	1	0	3	0	4
External	2	3	0	1	6
Fixed	11	2	4	20	37
Not Reproduced	0	0	1	0	1
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	24	14	13	26	77

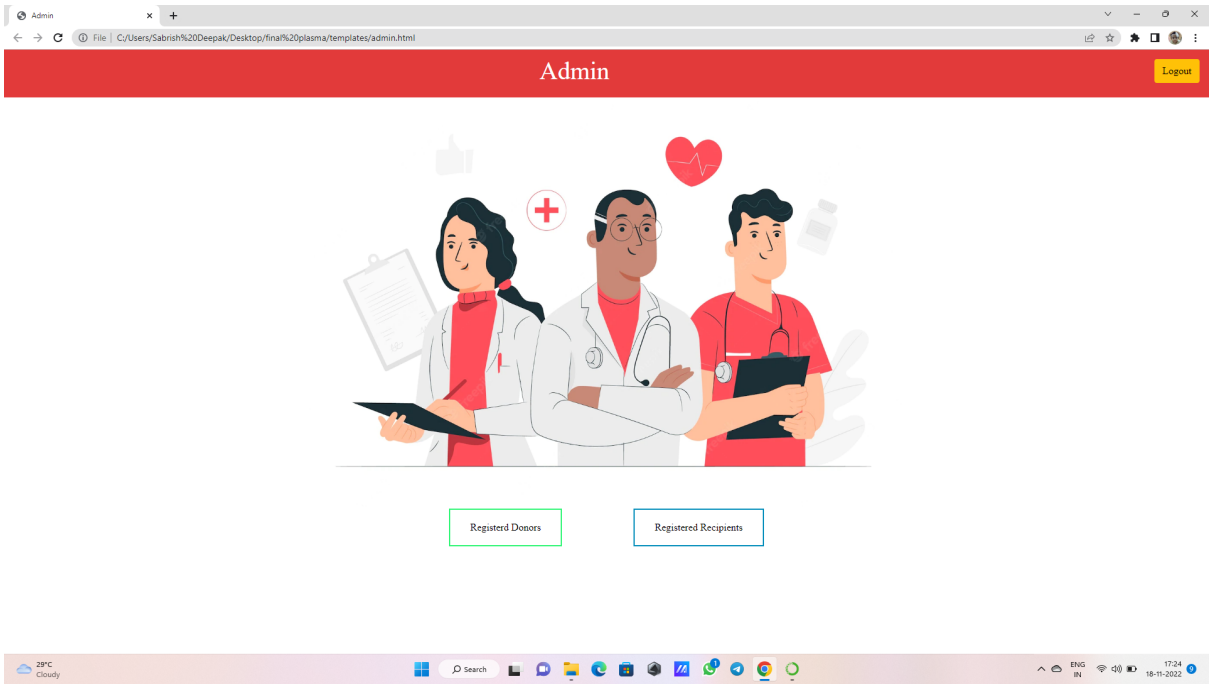
- Test Case Analysis:

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

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

9.RESULTS

9.1 Performance Metrics



10.ADVANTAGES & DISADVANTAGES

ADVANTAGES:-

Whenever an individual has a cut or injury, these clotting factors ensure that they do not lose too much blood. Plasma donations ensure that these individuals can receive a plasma transfusion to supplement their body's clotting ability and stop excessive bleeding from occurring.

Some of the advantages of the application include :

- It is a user-friendly application.
- It will help people to find plasma easily.
- App already filters the Active Members.
- Here a User can be a giver as well as a borrower.

DISADVANTAGES:-

- Wrong inputs will affect the project outputs.
- It cannot auto verify user genuineness.
- Internet Connection is mandatory.

11.CONCLUSION

The Plasma donor application was developed out of a need to make finding plasma supplies or a willing donor on time and using lesser time in searching for either ofthe two. This system should be made available to everyone because it will help the search of plasma supplies during emergency cases faster. This helps to avoid health complication and also possible deaths due to delays in search of Plasma.

An PlasmaDonor application is an exclusivesuite of servicesfor people who are in need of Plasma.It helps you track all the detailsabout the PlasmaDonors. The basic solution is to create a centralized system to keep a track on the upcomingas well as pastPlasma Donation Events.

12. FUTURE SCOPE

In future, our algorithm more congenial with more featuressuch as

- The analysis such as Frequently requestedzone or hospital for Plasma,Number of donors, mostly asked Blood Group, Age group of Patients need for plasma etc. can be addedas additional features.
- The application can be implemented using Artificial Intelligence and Deep Learning Algorithms.
- NGOs and NCC Units information's can be made available in the application.
- Donors last donateddetails can be automatically updatedin the App.
- Notification to Donorsabout the nearestPlasma Donation Center.
- Increase efficiency and customersatisfaction with an app alignedto theirneeds.
- Seamlessly integrate with existing infrastructure.

13.APPENDIX

Source Code

```
from flask import Flask, render_template, request, redirect, url_for, session
import ibm_db
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
import re
from twilio.rest import Client

app = Flask(__name__)

hostname = '98538591-7217-4024-b027-
8baa776ffad1.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud'
uid = 'jqj93360'
pwd = 'FljjHGlh2Y8TAce9'
driver = "{IBM DB2 ODBC DRIVER}"
db_name = 'bludb'
port = '30875'
protocol = 'TCPIP'
cert = "DigiCertGlobalRootCA.crt"
dsn = (
    "DATABASE ={0};"
    "HOSTNAME ={1};"
    "PORT ={2};"
    "UID ={3};"
    "SECURITY=SSL;"
    "PROTOCOL={4};"
    "PWD ={6};"
).format(db_name, hostname, port, uid, protocol, cert, pwd)
connection = ibm_db.connect(dsn, "", "")

print()

app.secret_key = 'a'
```

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    msg = " "
    if request.method == 'POST':
        username = request.form['username']
        email_id = request.form['email_id']
        phone_no = request.form['phone_no']

        blood_grp = request.form['blood_grp']
        gender= request.form['gender']
        address= request.form['address']
        dob = request.form['dob']
        weight= request.form['weight']
        password = request.form['password']
        query = "SELECT * FROM USER1 WHERE username=?;"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if (account):

            msg = "Account already exists!"
            return render_template('register.html', msg=msg)
        #elif not re.match(r'[^@]+@[^@]+\.[^@]+', email_id):
            # msg = "Invalid email addres"
        #elif not re.match(r'[A-Za-z0-9+', username):
           # msg = "Name must contain only characters and numbers"
        else:
            query = "INSERT INTO USER1 values(?,?,?,?,?,?,?,?,?)"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, username)
            ibm_db.bind_param(stmt, 2, email_id)
            ibm_db.bind_param(stmt, 3, phone_no)
            ibm_db.bind_param(stmt, 4, password)
            ibm_db.bind_param(stmt, 5, blood_grp)
            ibm_db.bind_param(stmt, 6, gender)
            ibm_db.bind_param(stmt, 7, address)
            ibm_db.bind_param(stmt, 8, dob)
            ibm_db.bind_param(stmt, 9, weight)
            ibm_db.execute(stmt)
```

```
            msg = 'You have successfully Logged In!!'
            return render_template('login.html', msg=msg)
    else:
        msg = 'PLEASE FILL OUT OF THE FORM'
        return render_template('register.html', msg=msg)

@app.route('/register2', methods=['GET', 'POST'])
def register2():
    msg = " "
    if request.method == 'POST':
        username = request.form['username']
        email_id = request.form['email_id']
        phone_no = request.form['phone_no']

        blood_grp = request.form['blood_grp']
        gender= request.form['gender']
        address= request.form['address']
        dob = request.form['dob']
        weight= request.form['weight']
        password = request.form['password']
        query = "SELECT * FROM USER2 WHERE username=?;"
        stmt = ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        if (account):

            msg = "Account already exists!"
            return render_template('register2.html', msg=msg)

        else:
            query = "INSERT INTO USER2 values(?,?,?,?,?,?,?,?,?)"
            stmt = ibm_db.prepare(connection, query)
            ibm_db.bind_param(stmt, 1, username)
            ibm_db.bind_param(stmt, 2, email_id)
            ibm_db.bind_param(stmt, 3, phone_no)
            ibm_db.bind_param(stmt, 4, password)
            ibm_db.bind_param(stmt, 5, blood_grp)
            ibm_db.bind_param(stmt, 6, gender)
            ibm_db.bind_param(stmt, 7, address)
            ibm_db.bind_param(stmt, 8, dob)
            ibm_db.bind_param(stmt, 9, weight)
```

```
            ibm_db.execute(stmt)
            msg = 'You have successfully Logged In!!'
            return render_template('login2.html', msg=msg)
    else:
        msg = 'PLEASE FILL OUT OF THE FORM'
        return render_template('register2.html', msg=msg)


@app.route("/login",methods=['GET','POST'])
def login():
    if request.method=="POST":
        username=request.form['username1']
        password=request.form['password1']
        query="select * from USER1 where username=? and password=?;"
        stmt=ibm_db.prepare(connection, query)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        data=ibm_db.fetch_assoc(stmt)
        if data:
            session['loggedin']=True
            msg='Login Successfully'
            return redirect(url_for('view2'))

        else:
            msg='Incorrect Username or Password'
    return render_template("login.html")

@app.route('/loginadmin', methods=['GET', 'POST'])
def loginadmin():
    global userid
    msg = ' '
    if request.method == "POST":
        username = request.form['username']
        password = request.form['password']
        query = "select * from admin where username=? and password=?"
        stmt = ibm_db.prepare(connection, query)
        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']
            session['username'] = account['USERNAME']
            msg = 'Logged in Successfully'
            return render_template('admin.html', msg=msg, username=str.upper(username))
        else:
            msg = 'Incorrect Username or Password'
            return render_template('loginadmin.html')
    else:
        msg = 'PLEASE FILL OUT OF THE FORM'
        return render_template('loginadmin.html', msg=msg)

@app.route('/login2', methods=['GET', 'POST'])
def login2():
    global userid
    msg = ' '
    if request.method == "POST":
        username = request.form['username2']
        password = request.form['password2']
        query = "select * from user2 where username=? and password=?"
        stmt = ibm_db.prepare(connection, query)
        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']
            session['username'] = account['USERNAME']
            msg = 'Logged in Successfully'
            return redirect(url_for('view'))
        else:
            msg = 'Incorrect Username or Password'
            return render_template('view.html', msg=msg)
    else:
        msg = 'PLEASE FILL OUT OF THE FORM'
        return render_template('login2.html', msg=msg)
```

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

 query = "SELECT * FROM USER1"
 stmt = ibm_db.prepare(connection, query)
 ibm_db.execute(stmt)
 data=[]
 tuple = ibm_db.fetch_tuple(stmt)
 while tuple!=False:
       data.append(tuple)
       tuple=ibm_db.fetch_tuple(stmt)
 return render_template("view.html",data=data)


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


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

 query = "SELECT * FROM USER2"
 stmt = ibm_db.prepare(connection, query)
 ibm_db.execute(stmt)
 data=[]
 tuple = ibm_db.fetch_tuple(stmt)
 while tuple!=False:
       data.append(tuple)
       tuple=ibm_db.fetch_tuple(stmt)
 return render_template("view2.html",data=data)




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

```
        query="select email_id from USER1"

        stmt = ibm_db.prepare(connection, query)

        ibm_db.execute(stmt)
        data = ibm_db.fetch_assoc(stmt)
        print(data)
        message =
Mail(from_email='sabrishdeepak2512@gmail.com',to_emails=data['EMAIL_ID'],subject='I want plasma
of your Blood Group.Kindly contact us',html_content='<strong>and easy to do anywhere, even with
Python</strong>')
        try:
            sg =
SendGridAPIClient('SG.9xVSY64UQSi2bQRTnK_TwA.vdgxQhG9sSTCQrY97duxg4L7tAg6hoH8iDAs
wG1IkWc')
            response = sg.send(message)
            print(response.status_code)
            print(response.body)
            print(response.headers)
        except Exception as e:
            print(e)
    return redirect('/view')

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

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

GitHub & Project Demo Link:

Github Link:

https://github.com/IBM-EPBL/IBM-Project-39173-1660399296

Project Demo Link:

https://youtu.be/_69galZqwzE