PLASMA DONOR APPLICATION

Team id	PNT2022TMID45809
Project Name	Plasma Donor Application
Team Members	1) R.ABINESH-813519104002 2) S.JEYANTH RAJESH-8135191016 3) S.MADHAVAN-813519104023 4) S.VIGNESH-813519104056

Table Of Contents

INTRODUCTION

PROJECT OVERVIEW:

The main goal of our project is to design a user-friendly web application that is like a scientific vehicle from which we can help reduce mortality or help those affected by COVID19 by donating plasma from patients who have recovered without approved antiretroviral therapy planning for a deadly COVID19 infection, plasma therapy is an experimental approach to treat those COVID-positive patients and help them recover faster.

Therapy, which is considered reliable and safe. If a particular person has fully recovered from COVID19, they are eligible to donate their plasma. As we all know, the traditional methods of finding plasma, one has to find out for oneself by looking at hospital records and contacting donors have been recovered, sometimes may not be available at home and move to other places. In this type of scenario, the health of those who are sick becomes disastrous. Therefore, it is not considered a rapid process to find plasma.

PURPOSE:

During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low.

The Purpose of this Application is Saving the donor information and helping the needy by notifying the current donors list, would be a helping hand. In regard to the problem faced, This application is to be built which would take the donor details, store them and inform them upon a request.

2 LITERATURE SURVEY

EXISTING PROBLEM:

- Cannot Upload and Download the latest updates.
- No use of Web Services and Remoting.
- Risk of mismanagement and of data when the project is under development.
- Less Security.
- No proper coordination between different Applications and Users.
- Fewer Users – Friendly

REFERENCE:

[1] R. C. Gojko Adzic, –Serverless computing: Economic and architectural impact,‖ ESEC/FSE, 2017.

[2] P. C. P. C. a. V. I. M. Yan, –Building a chatbot with server less computing,‖ IBM watson research center, 2016.

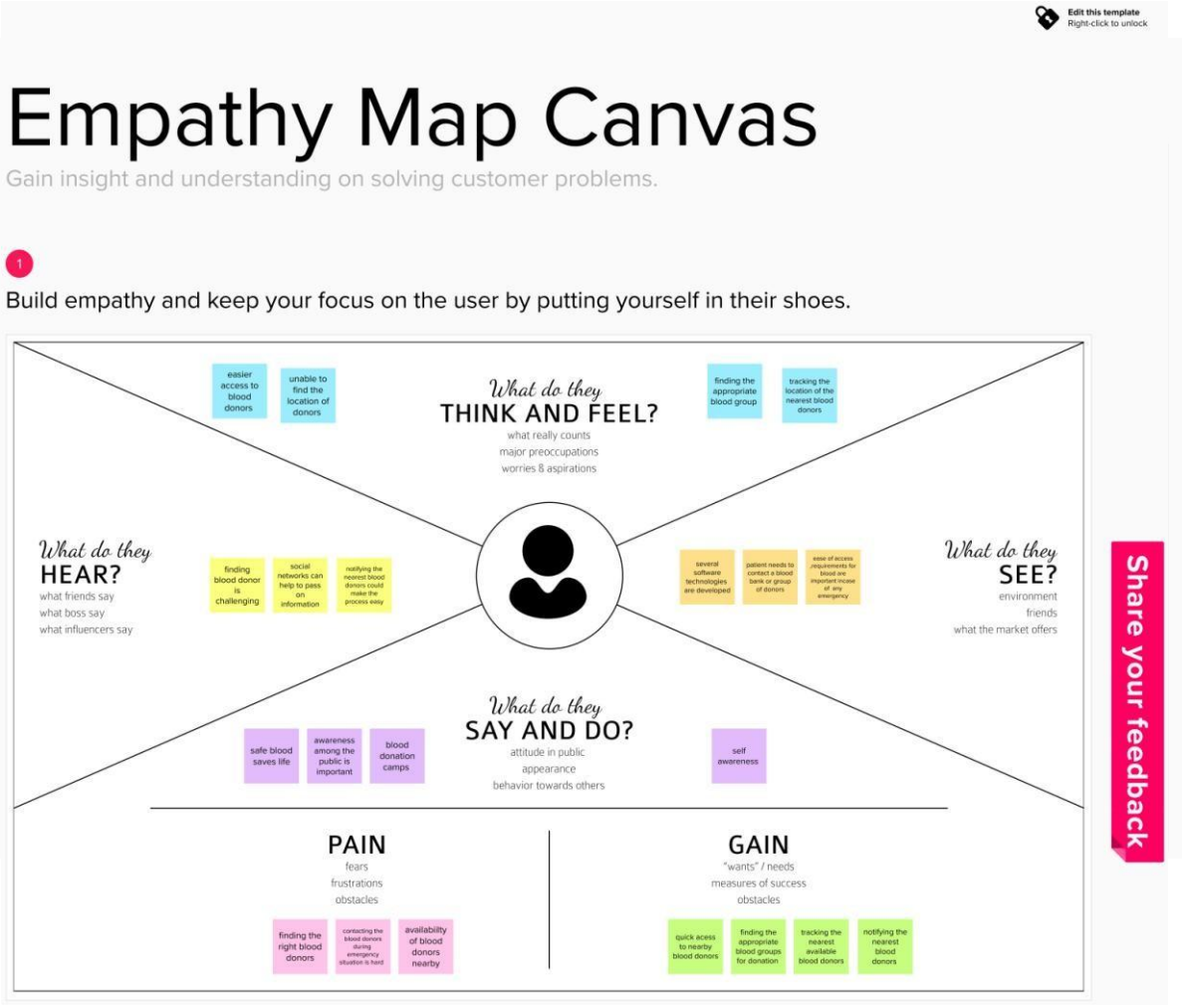
[3] S. E. a. B. J. J. Short, ––Cloud Event Programming Paradigms: Applications and Analysis,‖,‖ 9th IEEE International Conference on Cloud Computing (CLOUD), pp. pp. 400-406, 2017.

[4] Z. Al-Ali, ––Making Server less Computing More Server less,‖,‖ IEEE 11th International Conference on Cloud Computing (CLOUD), pp. pp. 456-459, 2018., 2018.

[5] A. S. a. S. Jindal, ––EMARS: Efficient Management and Allocation of Resources in Serverless,‖,‖ IEEE 11th International Conference on Cloud Computing (CLOUD), pp. pp. 827-830, 2018.

3. IDEATION & PROPOSED SOLUTION

Empathy Map Canvas:



Brainstorm & Idea Prioritization Template:

step-1: Team Gathering, Collaboration and Select the Problem Statement

Step-2: Brainstorm, Idea Listing and Grouping

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

Based on Data Collection

Donor details

Blood Type

data collection

medical history

Blood Type

Based on DB Connectivity

Scalability

Scalability

Based on DB connectivity

API Connectivity

Scalability

Based on Cloud

Immediate data recovery

Microservices

Based on cloud infrastructure

Kubernetes

Storage

Based on User Interface

Easier navigation

Faster Loading

Based on user application

Needed Features

Low Processing overhead

Step-3: Idea Prioritization

4

Prioritize

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

20 minutes

Proposed Solution Template:

Project team shall fill the following information in proposed solution template

S. No.	Parameter	Description
1.	Problem Statement (Problem 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 of a 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 The donor to the centre.
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 Model)	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 .

PROBLEM SOLUTION FIT:

1. CUSTOMER SEGMENT(S) CS

Adding features like above age of 21 can donate. Donor/Recipient/Hospitals can utilize this platform for their Plasma sharing process.

6. CUSTOMER LIMITATIONS EG. BUDGET, DEVICES CL

Once blood is donated means, the donor could not able to donate the plasma for another 28 days. Our web application doesn,t allow the users multiple times in a period of 28 days.

5. AVAILABLE SOLUTIONS PROS & CONS AS

Available solutions are uncomfortable and needs a admin user so it is much needs a better solutions.

2. PROBLEMS / PAINS + ITS FREQUENCY PR

During the COVID 19 crisis, the requirement of plasma became a high priority and the donor count has become low. Saving the donor information and helping the needy by notifying the current donors list, would be a helping hand. In regard to the problem faced, an application is to be built which would take the donor details, store them and inform them upon a request.

9. PROBLEM ROOT / CAUSE RC

The root/cause of this problem is COVID-19 and the donor count of the plasma becomes low. So this made the users to suffer a lot. In regard to the problem faced, an application is to be built which would take the donor details, store them and inform them upon a request.

7. BEHAVIOR + ITS INTENSITY BE

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

3. TRIGGERS TO ACT TR

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

4. EMOTIONS BEFORE / AFTER EM

Donor get fear, anxiety prior to donation give way to largely positive emotional states like clearing all their doubts in this web application.

10. YOUR SOLUTION SL

Our web application is able to give the user friendly environment and doesn't needs an admin user for maintaining the website. Hospitals , Donors and Recipients can get more satisfied by using this application. We making the donors to enter their deails and providing their details to hospitals and recipients an get their plasma fromnearest locations available.

8. CHANNELS of BEHAVIOR CH

ONLINE

Online web application allows user to make donation and receiving process easier.send request from anywhere anytime.

OFFLINE

Donors to visit nearby hospital and donate as well as receive plasma.

4. REQUIREMENT ANALYSIS:

FUNCTIONAL REQUIREMENTS:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form (WebApp)
FR-2	User Confirmation	Confirmation via EmailConfirmation via OTP
FR-3	Certification	After the donor donates plasma, we will give them a certificate 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 the request form on the page. Once the request is submitted, they will get an email
FR-6	Searching/reporting requirements	Users can use the search bar to look up information about 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 and details of plasma donation.

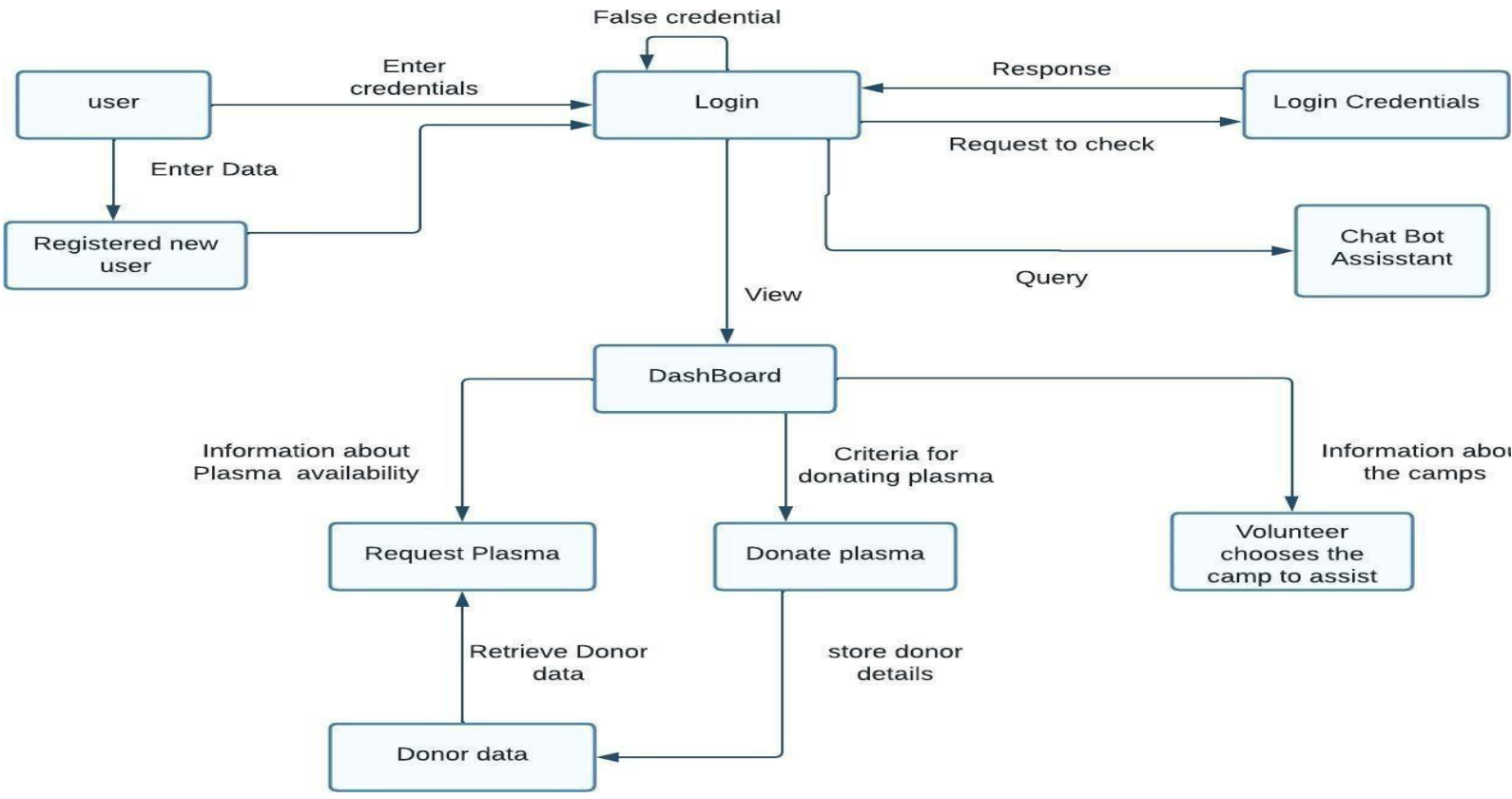
NON-FUNCTIONALREQUIREMENTS:

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

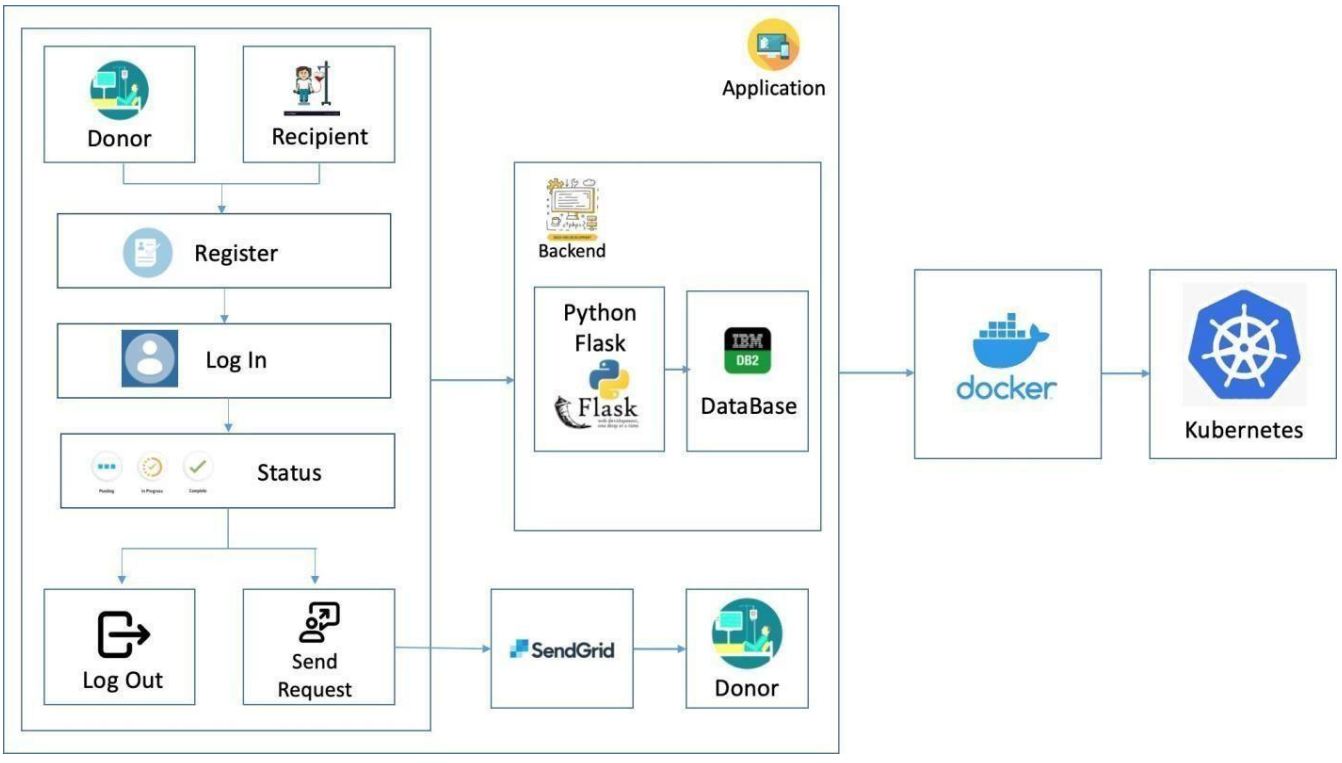
NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	Must have a good-looking User-friendly interface.
NFR-2	Security	It must be secured with the proper username and password.
NFR-3	Reliability	The system should be made in such a way that it is reliable in its operations and for securing the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

5. PROJECT DESIGN

Data Flow Diagram:



Solution & Technical Architecture:



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 confirmingmy password.	I can access my account /dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation emailonce I have registered for the application	I can receive confirmationemail & 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 indashboard	High	Sprint-1
	Dashboard	USN-5	As a user, I can send the proper requests todonate 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 toview the profile	I can access into my User profile and view details indashboard	High	Sprint-1
	Dashboard	USN-7	As a user, I can send the proper requests todonate 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	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 appearance friendly manner	Medium	Sprint-3

		USN-10	As an administrator, I can involve working withthe technical side of websites.	I can help with such as troubleshooting issues, setting up web hosts, ensuring users have access and programmingservers	Medium	Sprint-1

6. PROJECT PLANNING & SCHEDULING

Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	Plasma Donor Application	User Story / Task	Story Points	Priority	Team members
Sprint-1	Registration	PDA-1	As a user, I can register for the application by entering my Name, email, password, confirming my password, Age, BloodGroup.	3	High	Jeyanth Rajesh
Sprint-3		PDA-2	As a user, I will receive confirmation email once I have registered for theapplication	3	Medium	Abinesh
Sprint-2		PDA-3	As a user, I can register for theapplication through Gmail	5	Medium	Madhavan
Sprint-1	Login	PDA-4	As a user, I can log into the application by entering email and password	2	High	Vignesh
Sprint-3		PDA-5	As a user, I can reset mypassword using Forgot Password option	4	Medium	Jeyanth Rajesh
Sprint-4		PDA-6	As a user, I can view mypast requests for plasma donation	3	Low	Madhavan
Sprint-4		PDA-7	As a user, I can close past requests I made for plasma	2	Low	Abinesh

Sprint-1	Home Page	PDA-8	As a user, I can view the homepage of the website	2	Medium	Vignesh
Sprint - 1	About Page	PDA-9	As a user, I can view the about page on the website and get information related to Plasma Donation	2	Medium	Abinesh
Sprint - 2	Register as Donor	PDA-11	As a user, I can register as a donor by submitting a form and uploading certificate of recovery from Covid-19	3	High	Jeyanth Rajesh
Sprint	Functional Requirem	User Story Num	User Story / Task	Story Points	Priority	Team Members

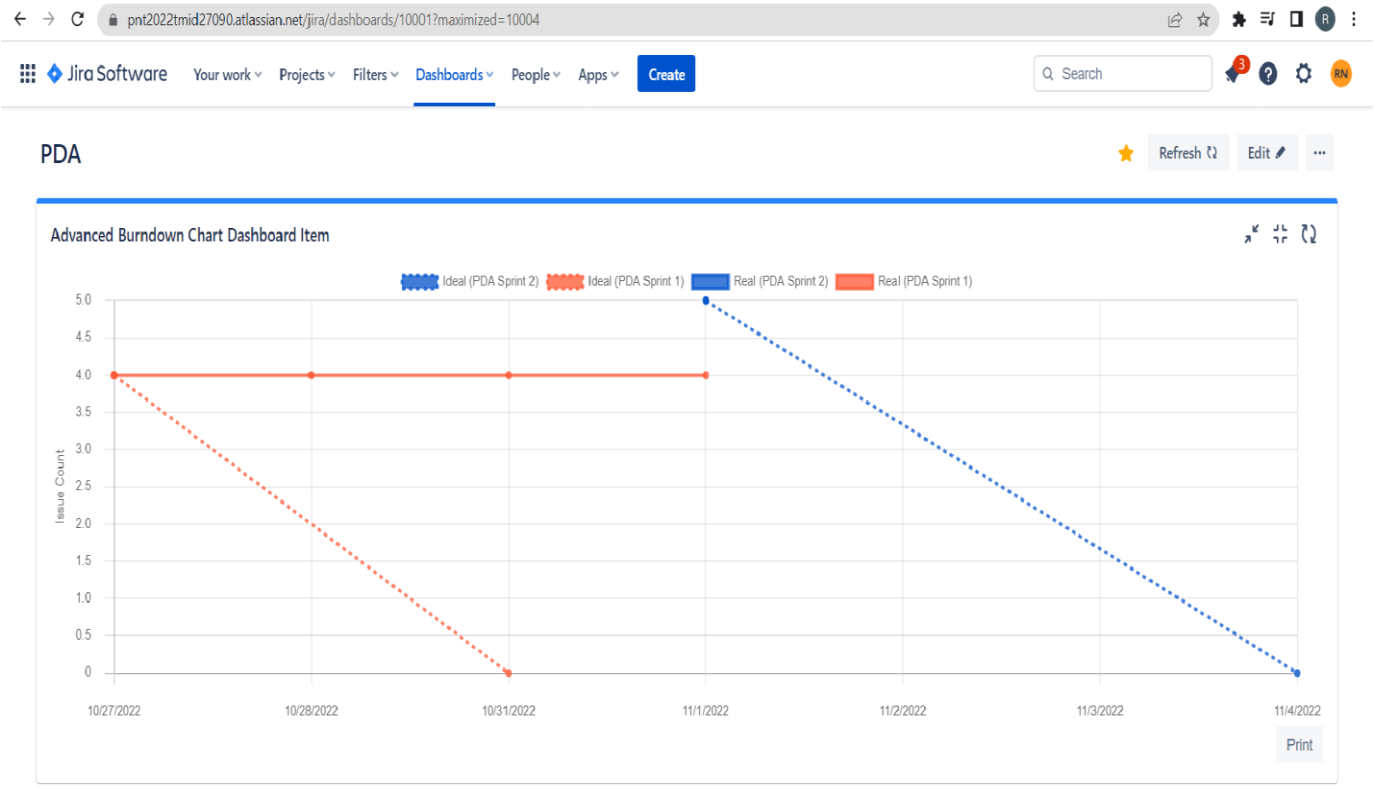
	ent(Epic)	ber				

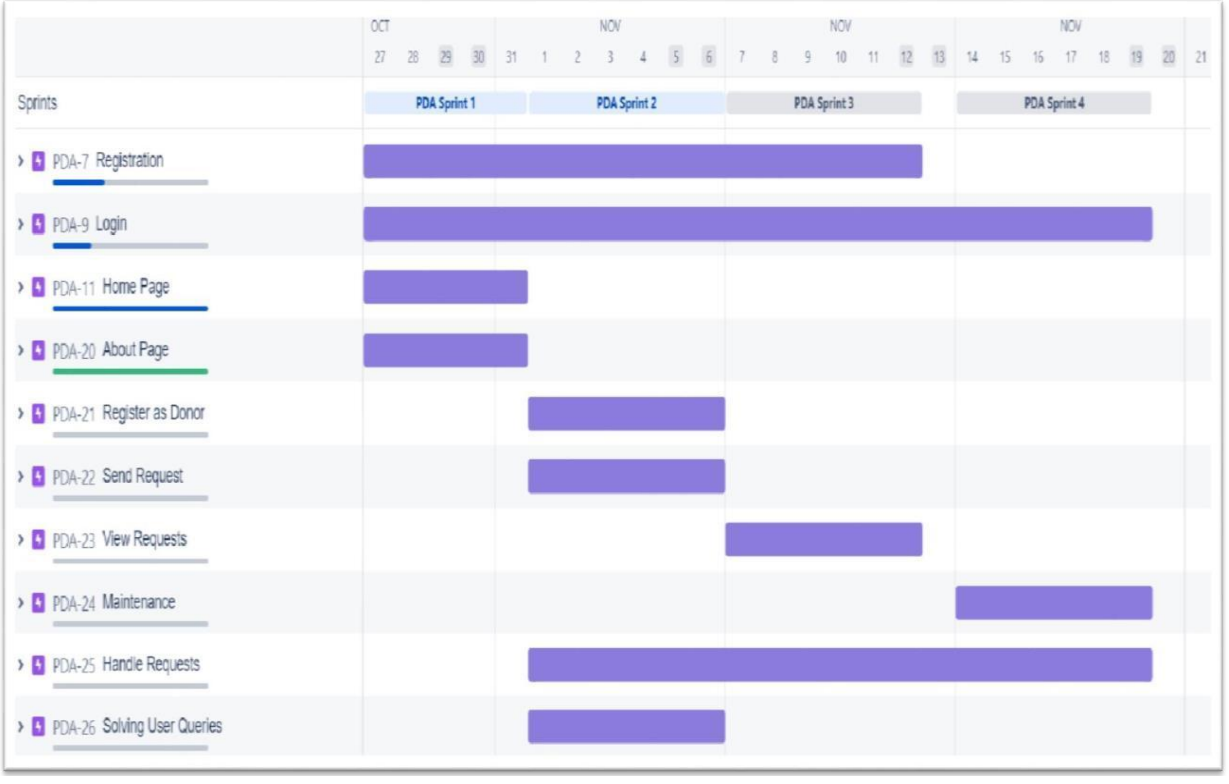
Sprint-2	Send Request	PDA-12	As a user, I can raise a request for plasma donation withspecific requirements through the request page.	2	High	Vignesh
Sprint-3	View Requests	PDA-13	As a user, I can view requests for plasma donation verified by admin	4	Medium	Jeyanth Rajesh
Sprint-4	Maintenance	PDA-14	As an admin, I can maintain the databases involved	2	Medium	Madhavan
Sprint-2	Handle Requests	PDA-15	As an admin, I can view all requests for plasma donation	1	High	Abinesh
Sprint-4		PDA-16	As an admin, I can delete requests that are past some timeperiod or have been closed	3	Low	Vignesh
Sprint-2	Solving User Queries	PDA-17	Creating a ChatBot that helps to solve the queries of the user.	2	High	Jeyanth Rajesh

Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	8	5 Days	27 Oct 2022	31 Nov 2022	8	03 Nov 2022
Sprint-2	13	4 Days	01 Nov 2022	06 Nov 2022	12	07 Nov 2022
Sprint-3	11	5 Days	07 Nov 2022	12 Nov 2022	11	09 Nov 2022
Sprint-4	9	5 Days	14 Nov 2022	19 Nov 2022	8	15 Nov 2022

Reports from JIRA





7 CODING & SOLUTIONING

FEATURE 1:

Python

It is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.[33]

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming.

It is often described as a "batteries included" language due to its comprehensive standard library.[34][35]

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.[36]

Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.[37]

Python consistently ranks as one of the most popular programming languages

FEATURE 2:

Flask

Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries.[2]

It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions.

However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools.

Database Schema

IBM Db2 -

a hybrid ANSI-compliant data virtualization tool for accessing, querying and summarizing data across the enterprise which:

• Provides a massively parallel processing (MPP) architecture Exploits Hive, HBase and Apache Spark concurrently for best-in-class analytic capabilities

• Requires only a single database connection or query to connect disparate sources such as HDFS, RDMS, NoSQL databases, object stores and Web HDFS

• Provides low latency support for ad-hoc and complex queries, high performance, and federation capabilities

• Understands dialects from other vendors and various products from Oracle, IBM® Db2® and IBM Netezza®

• Enables advanced row and column security

KUBERNATES-

Kubernetes — also known as –k8s‖ or –kube‖ — is a container orchestration platform for scheduling and automating the deployment, management, and scaling of containerized applications.

Kubernetes was first developed by engineers at Google before being open sourced in 2014. It is a descendant of Borg, a container orchestration platform used internally at Google. Kubernetes is Greek for *helmsman* or *pilot*, hence the helm in the Kubernetes logo (link resides outside IBM).

Today, Kubernetes and the broader container ecosystem are maturing into a general-purpose computing platform and ecosystem that rivals — if not surpasses — virtual machines (VMs) as the basic building blocks of modern cloud infrastructure and applications.

This ecosystem enables organizations to deliver a high-productivity Platform-as-a-Service (PaaS) that addresses multiple infrastructure-related and operations-related tasks and issues surrounding cloud-native development so that development teams can focus solely on coding and innovation.

8 TESTING

TESTING CASE:

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product.

It provides a way to check the functional it your components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectation and does not fail in an unacceptable manner.

There are various types of test. Each test type addresses a specific testing requirement

ACCEPTANCE TESTING

Acceptance Testing
UAT Execution & Report Submission

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).

2 .Defect Analysis

This report shows the number of resolved or closed bugs at eachseverity 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

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
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

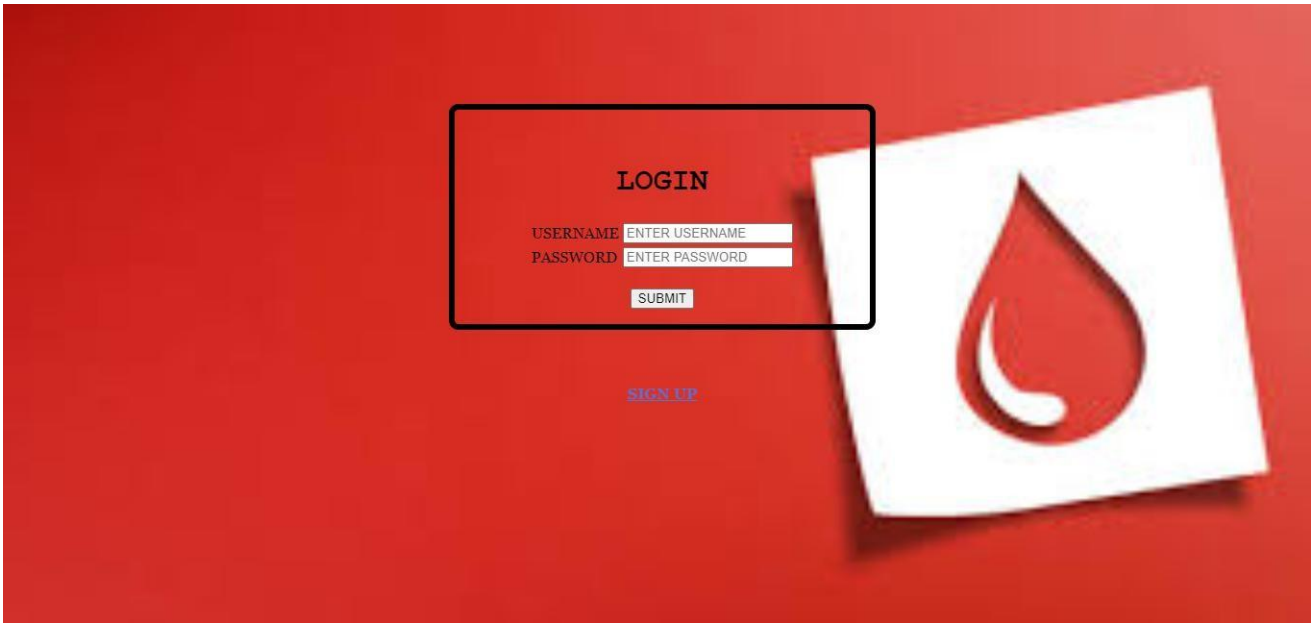
9 RESULTS

PERFORMANCE METRICS:

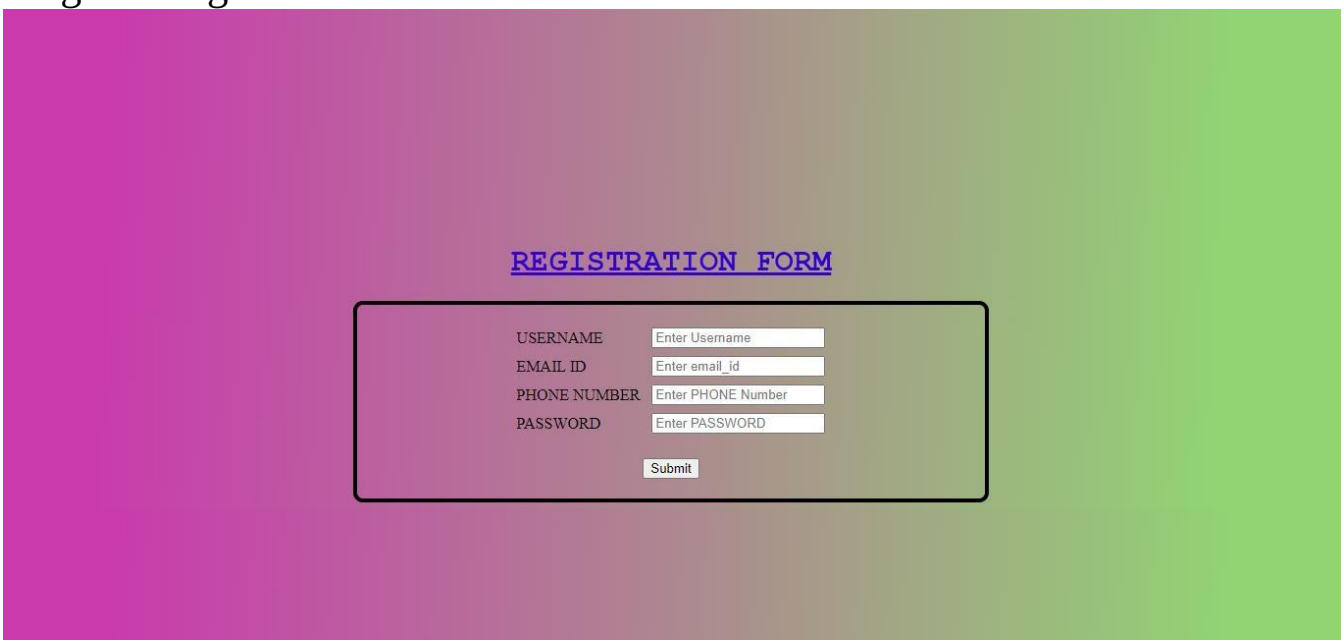
- Project metrics are used to track the progress and performance of a project.
- Monitoring parts of a project like **productivity, scheduling, and scope** make it easier for team leaders to see what's on track.
- As a project evolves, managers need access to changing deadlines or budgets to meet their client's expectations

OUTPUT SCREENS:

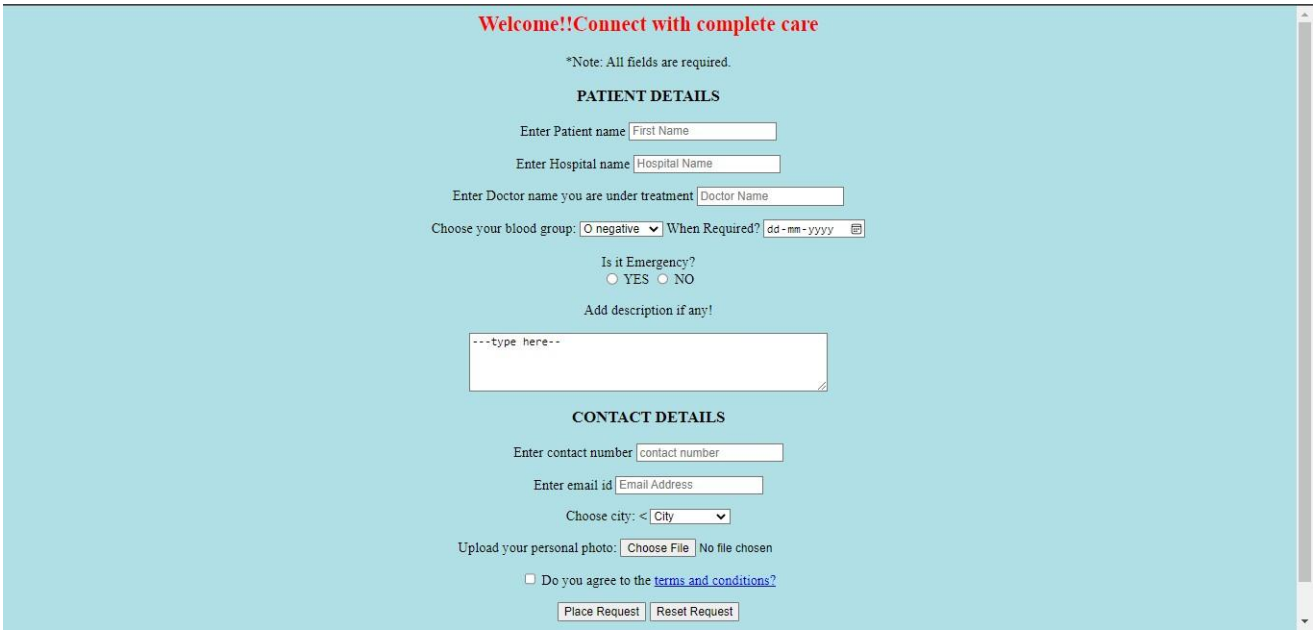
Login Page



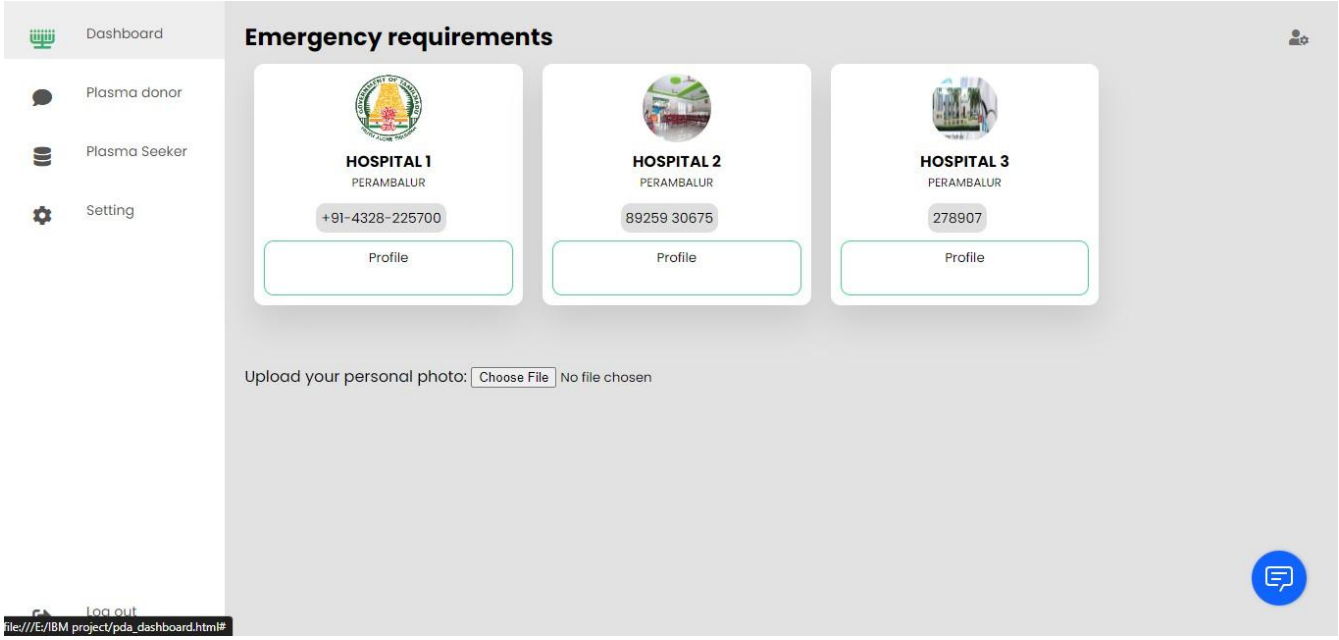
Register Page:



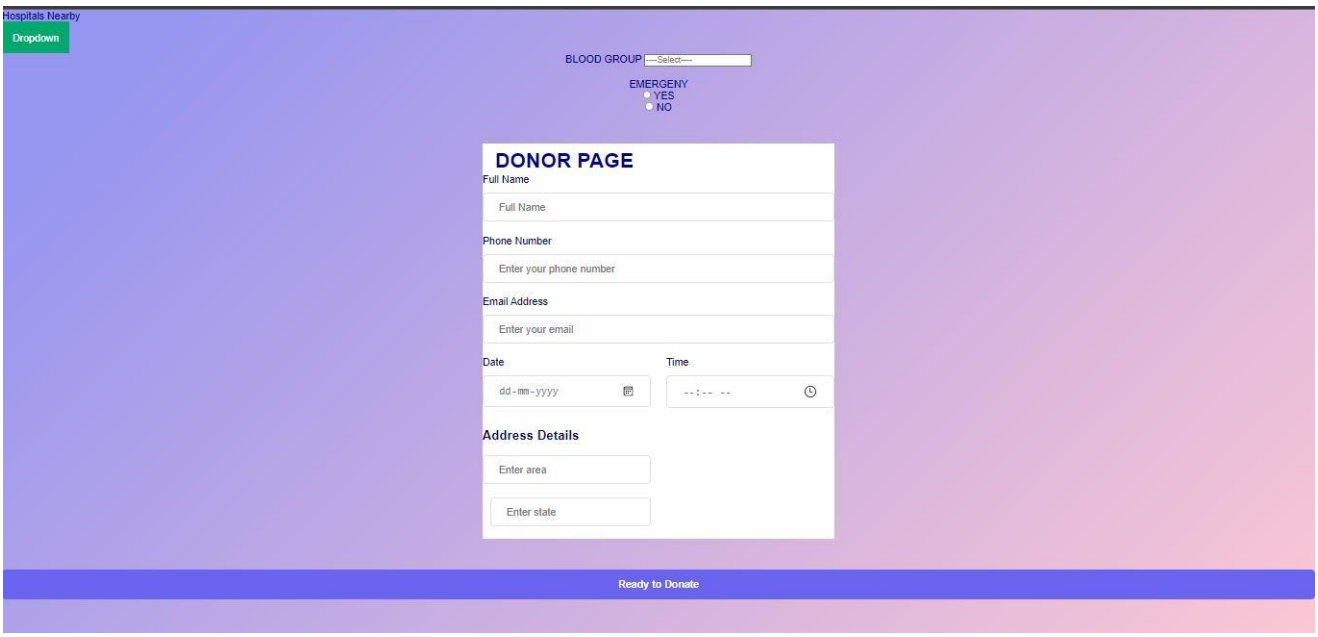
Request Page:



Dashboard Page:

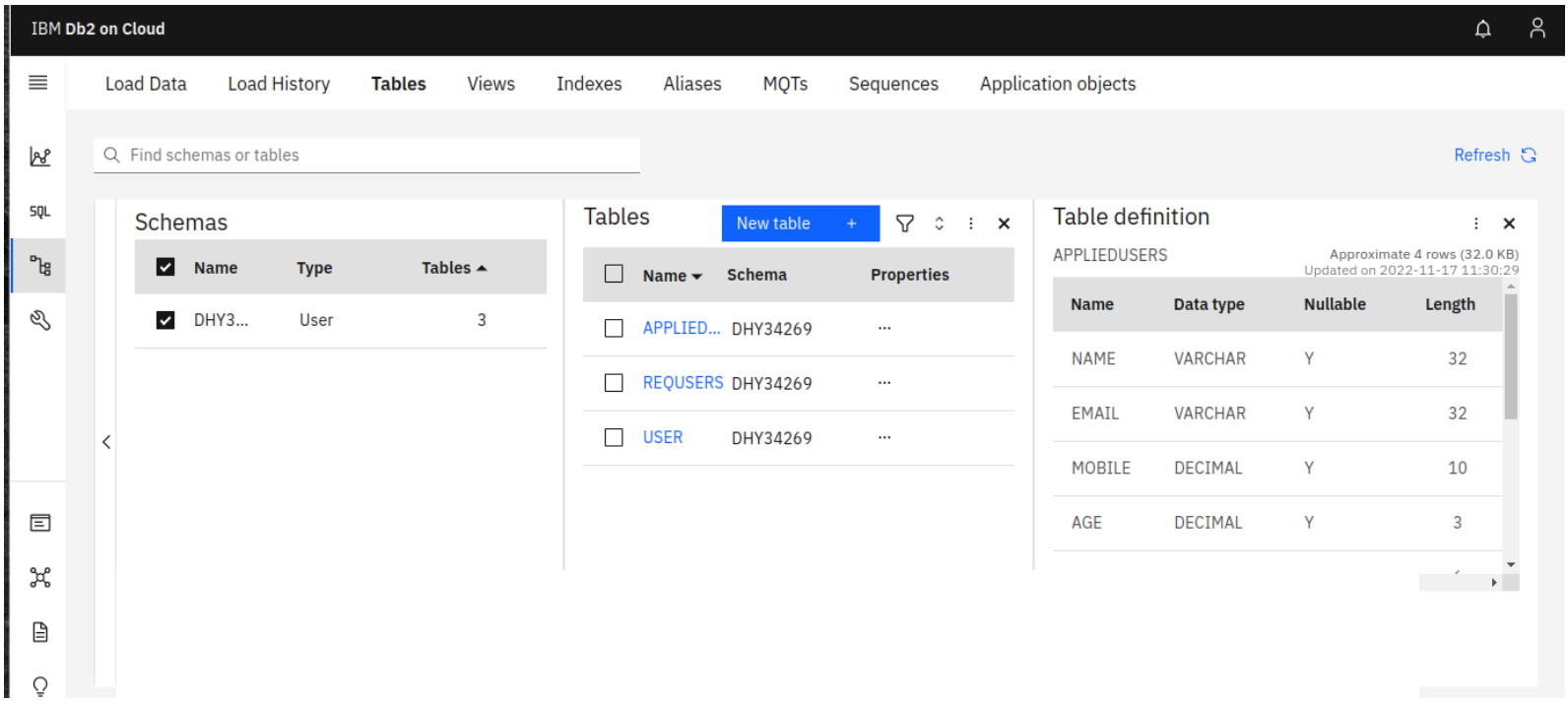


Plasma Donor Page



Send grid:

IBM Db 2



DHY34 269.APPLY EDUSER S

Back

Export to CSV

10 ADVANTAGES & DISADVANTAGES

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 upon the 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 CONCLUSIONS

The efficient way of finding plasma door for the infected people is implemented using the plasma donor website that is hosted on IBM Cloud platform.

To ensure the smooth functioning of the web site operation. I have hosted the website in IBM Db2 & Kubernates Cluster to make sure the operations are running successfully Cloud lambda function is used and to deploy the application IBM Db2 service is used.

12 FUTURE ENHANCEMENTS

Upgrading the UI that is more user-friendly which will help many users to access the website and also ensures that many plasma donors can be added into the community.

Using elastic load balancer, it helps to handle multiple requests at the same time which will maintain the uptime of the website with negligible downtime.

Source code:

Login Page:

```
<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>Login</title>
</head>

<style>
    body {
        font-family: Georgia, 'Times New Roman', Times, serif;
        background-image: url("https://encrypted-
tbn0.gstatic.com/images?q=tbn:ANd9GcSuf06V3IAppe36LZG6IzjIjG7GnnWHInt0SA&usqp=CAU");
        background-repeat: no-repeat;
        background-position: center;
        background-size: cover;
        position: fixed;
  top: 0;
  left: 0;

  /* Preserve aspet ratio */
  min-width: 100%;
  min-height: 100%;
 }


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

    h1 {
        font-family: 'Courier New', Courier, monospace;
        color: rgb(0, 0, 0);
        top: 10em;
    }

    .container1 {
        border: 6px solid black;
        border-color: black;
        border-radius: 10px;
        width: 400px;
        padding: 16px;

    }

    .top {
        margin-top: 100px;
```

```
    }

    input:hover {
        border-color: rgb(25, 20, 20);
    }

    a {
        text-decoration: none;
    }

    a:link {
        color: #0c0c0c;
        text-decoration: underline;
    }

    a:visited {
        color: rgb(92, 112, 215);
        text-decoration: none;
    }

    a:hover {
        color: rgb(128, 105, 255);
        text-decoration: none;
    }

    a:active {
        color: rgb(75, 202, 155);
        text-decoration: none;
    }
</style>

<body>
    <center>
        <h1 class="top"></h1>
        <div class="container1">
            <br>
            <h1>LOGIN</h1>

                <table>
                    <tr>
                        <td><label for="text">USERNAME</label></td>
                        <td><input type="text" name="username" placeholder="ENTER USERNAME" /></td>
                    </tr>
                    <tr>
                        <td><label for="text">PASSWORD</label></td>
                        <td><input type="text" name="password" placeholder="ENTER PASSWORD"></td>
                    </tr>
                </table>
                <br>
                <button onclick ="location.href='pda_welcomepage.html';">SUBMIT

            <br>
        </div>
        <br>
         <br> <br> <b><a href="pda_register.html">SIGN UP</a></b></label>
    </center>
</body>

</html>
```

Register Page:

```
<!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(92.7deg, rgb(201, 59, 173) 8.5%, rgb(146, 211, 116) 90.2%);
        font-family: 'Times New Roman', Times, serif;
    }

    input:hover {
        border-color: rgb(25, 20, 20);
    }

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

    h1 {
        font-family: 'Courier New', Courier, monospace;
        color: rgb(53, 2, 206);
        text-decoration: underline;
    }

    .container2 {
        border: 4px solid black;
        border-color: black;
        border-radius: 10px;
        width: 600px;
        padding: 20px;

    }

    #qwerty {
        margin-top: 15em;
    }
</style>

<body>
    <center id="qwerty">
        <H1>REGISTRATION FORM</H1>
        <div class="container2">
        <!-- -->

            <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>
                <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>
                <td><label for="text">PHONE NUMBER</label></td>
                <td> </td>
                <td><input type="text" placeholder="Enter PHONE Number" name="phone_no" id="phone_no"
maxlength="10"></td>
            </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>
        </table>
        <br>
        <center><button onclick="location.href='pda_loginpage.html';">Submit
        </center>

    </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');
        if (username1.value == "" || phone_no.value == "" || password.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";
        }


    }


</script>

</html>
```

Home Page:

```
<!DOCTYPE html>
<html lang="en">
<head>
	<meta charset="UTF-8">
	<title>Plasma home</title>
	<link rel="stylesheet" href="style.css">
</head>
<style>*{
	padding: 0;
	margin: 0;
}
.wrapper{
	background: url(bg1.jpg) no-repeat;
	background-size: cover;
	height: 100vh;
}
.navbar{
	position: fixed;
	height: 80px;
	width: 100%;
	top: 0;
	left: 0;
	background: rgba(0,0,0,0.4);
}
.navbar .logo{
	width: 140px;
	height: auto;
	padding: 20px 100px;
}
.navbar ul{
	float: right;
	margin-right: 20px;
}
.navbar ul li{
	list-style: none;
	margin: 0 8px;
	display: inline-block;
	line-height: 80px;
}
.navbar ul li a{
	font-size: 20px;
	font-family: 'Roboto', sans-serif;
	color: white;
	padding: 6px 13px;
	text-decoration: none;
	transition: .4s;
}
.navbar ul li a.active,
.navbar ul li a:hover{
	background: red;
	border-radius: 2px;
}
.wrapper .center{
	position: absolute;
	left: 50%;
	top: 55%;
	transform: translate(-50%, -50%);
	font-family: sans-serif;
	user-select: none;
}
```

```
.center h1{
        color: white;
        font-size: 70px;
        width: 900px;
        font-weight: bold;
        text-align: center;
}
.center h2{
        color: white;
        font-size: 70px;
        font-weight: bold;
        margin-top: 10px;
        width: 885px;
        text-align: center;
}
.center .buttons{
        margin: 35px 280px;
}
.buttons button{
        height: 50px;
        width: 150px;
        font-size: 18px;
        font-weight: 600;
        color: #ffb3b3;
        background: red;
        outline: none;
        cursor: pointer;
        border: 1px solid #cc0000;
        border-radius: 25px;
        transition: .4s;
}
.buttons .btn2{
        margin-left: 25px;
}
.buttons button:hover{
        background: #cc0000;
}</style>
<body>
        <div class="wrapper">
                <nav class="navbar">
                        <img class="logo" src="logo.png">
                        <ul>
                                <li><a class="active" href="pda_homepage.html">Home</a></li>
                                <li><a href="pda_helpdesk.html">Help Desk</a></li>
                                <li><a href="pda_contactpage.html">Contact</a></li>
                                <li><a href="pda_feedbackform.html">Feedback</a></li>
                        </ul>
                </nav>
                <div class="center">
                <h1>Donate plasma</h1>
                <h2>save lives</h2>
                <div class="buttons">
                <button onclick ="location.href='pda_loginpage.html';"> Login
                <button onclick ="location.href='pda_register.html';">Register
        </div>
        </div>
</body>
</html>
```

Dashboard:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.4/css/all.min.css"/>
<title>Dashboard</title>
</head>
<style>/* import google fonts */
    @import url("https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=swap");
    *{
      margin: 0;
      padding: 0;
      outline: none;
      border: none;
      text-decoration: none;
      box-sizing: border-box;
      font-family: "Poppins", sans-serif;
    }
    body{
      background: rgb(226, 226, 226);
    }
    nav{
      position: sticky;
      top: 0;
      bottom: 0;
      height: 100vh;
      left: 0;
      width: 90px;
      /* width: 280px; */
      background: #fff;
      overflow: hidden;
      transition: 1s;
    }
    nav:hover{
      width: 280px;
      transition: 1s;
    }
    .logo{
      text-align: center;
      display: flex;
      margin: 10px 0 0 10px;
      padding-bottom: 3rem;
    }

    .logo img{
      width: 45px;
      height: 45px;
      border-radius: 50%;
    }
    .logo span{
      font-weight: bold;
      padding-left: 15px;
      font-size: 18px;
      text-transform: uppercase;
    }
    a{
      position: relative;
      width: 280px;
      font-size: 14px;
```

```
  color: rgb(85, 83, 83);
  display: table;
  padding: 10px;
}
nav .fas{
  position: relative;
  width: 70px;
  height: 40px;
  top: 20px;
  font-size: 20px;
  text-align: center;
}
.nav-item{
  position: relative;
  top: 12px;
  margin-left: 10px;
}
a:hover{
  background: #eee;
}
a:hover i{
  color: #34AF6D;
  transition: 0.5s;
}
.logout{
  position: absolute;
  bottom: 0;
}

.container{
  display: flex;
}

/* MAin Section */
.main{
  position: relative;
  padding: 20px;
  width: 100%;
}
.main-top{
  display: flex;
  width: 100%;
}
.main-top i{
  position: absolute;
  right: 0;
  margin: 10px 30px;
  color: rgb(110, 109, 109);
  cursor: pointer;
}
.main .users{
  display: flex;
  width: 100%;
}
.users .card{
  width: 25%;
  margin: 10px;
  background: #fff;
  text-align: center;
  border-radius: 10px;
  padding: 10px;
```

```
  box-shadow: 0 20px 35px rgba(0, 0, 0, 0.1);
}
.users .card img{
  width: 70px;
  height: 70px;
  border-radius: 50%;
}
.users .card h4{
  text-transform: uppercase;
}
.users .card p{
  font-size: 12px;
  margin-bottom: 15px;
  text-transform: uppercase;
}
.users table{
  margin: auto;
}
.users .per span{
  padding: 5px;
  border-radius: 10px;
  background: rgb(223, 223, 223);
}
.users td{
  font-size: 14px;
  padding-right: 15px;
}
.users .card button{
  width: 100%;
  margin-top: 8px;
  padding: 7px;
  cursor: pointer;
  border-radius: 10px;
  background: transparent;
  border: 1px solid #4AD489;
}

/*Attendance List serction  */
.attendance{
  margin-top: 20px;
  text-transform: capitalize;
}
.attendance-list{
  width: 100%;
  padding: 10px;
  margin-top: 10px;
  background: #fff;
  border-radius: 10px;
  box-shadow: 0 20px 35px rgba(0, 0, 0, 0.1);
}
.table{
  border-collapse: collapse;
  margin: 25px 0;
  font-size: 15px;
  min-width: 100%;
  overflow: hidden;
  border-radius: 5px 5px 0 0;
}
table thead tr{
  color: #fff;
  background: #34AF6D;
```

```
      text-align: left;
      font-weight: bold;
    }
    .table th,
    .table td{
      padding: 12px 15px;
    }
    .table tbody tr{
      border-bottom: 1px solid #ddd;
    }
    .table tbody tr:nth-of-type(odd){
      background: #f3f3f3;
    }
    .table tbody tr.active{
      font-weight: bold;
      color: #4AD489;
    }
    .table tbody tr:last-of-type{
      border-bottom: 2px solid #4AD489;
    }
    .table button{
      padding: 6px 20px;
      border-radius: 10px;
      cursor: pointer;
      background: transparent;
      border: 1px solid #4AD489;
    }
    .table button:hover{
      background: #4AD489;
      color: #fff;
      transition: 0.5rem;
    }</style>
<body>
  <div class="container">
    <nav>
      <ul>
        <li><a href="#">
          <i class="fas fa-menorah"></i>
          <span class="nav-item">Dashboard</span>
        </a></li>
        <li><a href="pda_donorpage.html">
          <i class="fas fa-comment"></i>
          <span class="nav-item">Plasma donor</span>
        </a></li>
        <li><a href="pda_patientpage.html">
          <i class="fas fa-database"></i>
          <span class="nav-item">Plasma Seeker</span>
        </a></li>
        <li><a href="#">
          <i class="fas fa-cog"></i>
          <span class="nav-item">Setting</span>
        </a></li>
        <li><a href="pda_homepage.html" class="logout">
          <i class="fas fa-sign-out-alt"></i>
          <span class="nav-item">Log out</span>
        </a></li>
      </ul>
    </nav>


    <section class="main">
```

```
  <div class="main-top">
    <h1>Emergency requirements</h1>
    <i class="fas fa-user-cog"></i>
  </div>
  <div class="users">
    <div class="card">
      <img src="1.png">
      <h4>Hospital 1</h4>
      <p>Perambalur</p>
      <div class="per">
        <table>
          <tr>
            <td><span>+91-4328-225700</span></td>

          </tr>
          <tr>
         <td></td>
          </tr>
        </table>
      </div>
      <button>Profile<a href=https://perambalur.nic.in/public-utility-category/hospitals/> </a></button>
    </div>
    <div class="card">
      <img src="2.jpg">
      <h4>Hospital 2</h4>
      <p>Perambalur</p>
      <div class="per">
        <table>
          <tr>
            <td><span>89259 30675</span></td>
          </tr>
          <tr>
            <td></td>
            </tr>
         </table>
      </div>
      <button>Profile<a href=http://mghospital.in/></a></button>
    </div>
    <div class="card">
      <img src="3.jpg">
      <h4>Hospital 3</h4>
      <p>Perambalur</p>
      <div class="per">
        <table>
          <tr>
            <td><span>278907</span></td>
            </tr>
          <tr>
            <td></td>
            </tr>
        </table>
      </div>
      <button>Profile<a href=https://www.nhp.gov.in/hospital/siva-hospital-perambalur_-tamil_nadu></a></button>
    </div>

  </div>
  <br>
  <br><div>
<p class="col-xs-12"><form action="upload.php" method="post" enctype="multipart/form-data">
Upload your personal photo:
  <input type="file" name="fileToUpload" id="fileToUpload">
```

```
    </p>
</div>
<script>
  window.watsonAssistantChatOptions = {
    integrationID: "d8d944f9-1a1a-422a-871e-2525ead3bdda", // The ID of this integration.
    region: "au-syd", // The region your integration is hosted in.
    serviceInstanceID: "4210dafc-90a2-49e4-87ba-fe14d68be3bc", // 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>
```

tryy.py

```
# -*- coding: utf-8 -*-
"""
Created on Tue Nov 15 10:05:54 2022

@author: Rithiha
"""

from flask import Flask, render_template, request, redirect, url_for, session
import ibm_db
import bcrypt
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=125f9f61-9715-46f9-9399-
c8177b21803b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30426;Security=SSL;SSLServerCertificate=Digi
CertGlobalRootCA.crt;UID=wpy86763;PWD=ib01vP5v5WYQDNRY",",")



app = Flask(_name_)
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

@app.route("/",methods=['GET'])
def home():
    if 'email' not in session:
      return redirect(url_for('pda_homepage'))
    return render_template('pda_homepage.html',name='Home')
@app.route("/pda_homepage")
def pda_homepage():
  return render_template('pda_homepage.html')

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

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

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

@app.route("/pda_welcomepage")
def pda_welcomepage():
  return render_template('pda_welcomepage.html')
```

```
@app.route("/pda_dashboard")
def pda_dashboard():
  return render_template('pda_dashboard.html')

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

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



@app.route("/pda_register",methods=['GET','POST'])
def register():
  if request.method == 'POST':
    name = request.form['name']
    phn = request.form['phn']
    email = request.form['email']
    psw = request.form['psw']

    if not name or not email or not phn or not psw:
      return render_template('pda_register.html',error='Please fill all fields')
    hash=bcrypt.hashpw(psw.encode('utf-8'),bcrypt.gensalt())
    query = "SELECT * FROM user_detail WHERE email=? OR phn=?"
    stmt = ibm_db.prepare(conn, query)
    ibm_db.bind_param(stmt,1,email)
    ibm_db.bind_param(stmt,2,phn)
    ibm_db.execute(stmt)
    isUser = ibm_db.fetch_assoc(stmt)
    if not isUser:
      insert_sql = "INSERT INTO user_detail(name, email, phn, psw) VALUES (?,?,?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, name)
      ibm_db.bind_param(prep_stmt, 2, email)
      ibm_db.bind_param(prep_stmt, 3, phn)
      ibm_db.bind_param(prep_stmt, 4, hash)
      ibm_db.execute(prep_stmt)
      return render_template('pda_register.html',success="You can login")
    else:
      return render_template('pda_register.html',error='Invalid Credentials')

  return render_template('pda_register.html',name='Home')

@app.route("/pda_loginpage",methods=['GET','POST'])
def login():
    if request.method == 'POST':
      email = request.form['email']
      psw = request.form['psw']

      if not email or not psw:
        return render_template('pda_loginpage.html',error='Please fill all fields')
      query = "SELECT * FROM user_detail WHERE email=?"
      stmt = ibm_db.prepare(conn, query)
      ibm_db.bind_param(stmt,1,email)
      ibm_db.execute(stmt)
      isUser = ibm_db.fetch_assoc(stmt)
      print(isUser,psw)
```

```
    if not isUser:
      return render_template('pda_loginpage.html',error='Invalid Credentials')

    isPasswordMatch = bcrypt.checkpw(psw.encode('utf-8'),isUser['PSW'].encode('utf-8'))

    if not isPasswordMatch:
      return render_template('pda_loginpage.html',error='Invalid Credentials')

    session['email'] = isUser['EMAIL']
    return redirect(url_for('pda_welcomepage'))

  return render_template('pda_loginpage.html',name='Home')



@app.route("/pda_donorpage",methods=['GET','POST'])
def donar():
  if request.method == 'POST':
    bldgrp=request.form['bldgrp']
    fname = request.form['fname']
    phn = request.form['phn']
    email = request.form['email']
    date = request.form['date']
    time = request.form['time']
    area = request.form['area']
    state = request.form['state']

    insert_sql = "INSERT INTO donar(bldgrp,fname, phn, email,date,time,area,state) VALUES (?,?,?,?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, bldgrp)
    ibm_db.bind_param(prep_stmt, 2, fname)
    ibm_db.bind_param(prep_stmt, 3, phn)
    ibm_db.bind_param(prep_stmt, 4, email)
    ibm_db.bind_param(prep_stmt, 5, date)
    ibm_db.bind_param(prep_stmt, 6, time)
    ibm_db.bind_param(prep_stmt, 7, area)
    ibm_db.bind_param(prep_stmt, 8, state)
    ibm_db.execute(prep_stmt)
    return render_template('pda_dashboard.html',success="Thanks for your support")
  else:
      return render_template('pda_dashboard.html',error='Invalid Credentials')

@app.route("/pda_patientpage",methods=['GET','POST'])
def patientt():
  if request.method == 'POST':
    pname=request.form['pname']
    hname = request.form['hname']
    dname = request.form['dname']
    bldgrp = request.form['bldgrp']
    when = request.form['when']
    num = request.form['num']
    email = request.form['email']
    city = request.form['city']
    insert_sql = "INSERT INTO patient(pname,hname,dname,bldgrp,when,num,email,city) VALUES (?,?,?,?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, pname)
    ibm_db.bind_param(prep_stmt, 2, hname)
    ibm_db.bind_param(prep_stmt, 3, dname)
    ibm_db.bind_param(prep_stmt, 4, bldgrp)
    ibm_db.bind_param(prep_stmt, 5, when)
    ibm_db.bind_param(prep_stmt, 6, num)
```

```
    ibm_db.bind_param(prep_stmt, 7, email)
    ibm_db.bind_param(prep_stmt, 8, city)
    ibm_db.execute(prep_stmt)
    return render_template('pda_dashboard.html',success="Connect with complete care")
  else:
      return render_template('pda_dashboard.html',error='Invalid Credentials')


@app.route("/data")
def display():
  donar_list=[]
 # patient_list=[]




  #selecting_donar
  sql = "SELECT * FROM donar "
  stmt = ibm_db.exec_immediate(conn, sql)
  donar = ibm_db.fetch_both(stmt)
  while donar!= False :
      donar_list.append(donar)
      donar = ibm_db.fetch_both(stmt)
  print(donar_list)
  return render_template('pda_dashboard.html',bldgrp=donar_list)

  #selecting_col1
 # sql = "SELECT * FROM donar"
 # stmt = ibm_db.exec_immediate(conn, sql)
  #donar = ibm_db.fetch_both(stmt)
  #while donar!= False :
    # donar_list.append(donar)
    # donar = ibm_db.fetch_both(stmt)
  #print(donar_list)
  #return render_template('pda_dashboard.html',fname=donar_list)


@app.route('/logout')
def logout():
    session.pop('email', None)
    return redirect(url_for('login'))
if_name_== "_main_":
    app.run(debug=True)
```

Github link:

[IBM-EPBL/IBM-Project-41293-1660640944: Plasma Donor Application (github.com)]

Demo Link:

[https://drive.google.com/file/d/1EqpylHy7d96bxQUR4VV2b9VQrPsEgjoX/view?usp=share_link]