Project Report

1. INTRODUCTION

1.1 Project overview

Cloud computing helps in on-demand deliver of IT resources over the internet with pay-as-you-go pricing model where users have to pay only for the resource that they use. This helps to reduce the additional infrastructural cost and users can access technology services such as power, storage, compute, database, networking, analytics and also intelligence over the internet in order to offer flexible, innovation, and economies of scale. Users can run their infrastructure more efficiently and scale their business according to their requirement.

Cloud deployment modules such as public cloud, private cloud, hybrid cloud and community cloud helps the users to choose the type of deployment options that are beneficial for their company. Cloud service models consist of software as a service (saas), platform as a service (paas) and infrastructure as a service (iaas). In Software as a service a third party service providers will host the applications and make them available over the internet. Some a requires purchasing of licensed version with involves huge cost and with the help of software as a service those applications can also be used without having to buy the license of the software which is more cost effective. With the help of platform-as-a-service customers can run, develop and manage the applications without any complexity of building and maintaining the infrastructure which is associated with developing and launching the applications. Infrastructure as a service allows the enterprise to rent or lease the servers for compute and storage in cloud.

1.2 Purpose

Blood plasma donations are used for slightly more specific purposes than a general blood donation. The most common uses of plasma donations include individuals who have experienced a severe trauma, burn or shock, adults or children with cancer, and people with liver or clotting factor disorders.

Plasma.org was designed to provide information of availability of plasma to the receiver, and easy way of contacting for both donors and receivers.

2. LITERATURE SURVEY

2.1 Existing problem

1. Web Application

This plasma therapy is an experimental approach to treat corona-positive patients and help them recover. This plasma therapy is considered to be safe & promising. A person who has recovered from Covid can donate his/her plasma to a person who is infected with the corona virus. This system proposed here aims at connecting the donors & the patients by an online web application. By using this web application, the users can either raise a request for plasma donation or requirement. This system is used if anyone needs a Plasma Donor. This system comprises of Admin and User where both can request for a Plasma. In this system there is something called an active user, which means the user is an Active member of the Application and has recovered from Covid 19, only such people are recommended here for Plasma Donation. Both parties can accept or Reject the request.

2. Mobile Application

This plasma therapy is an experimental approach to treat corona-positive patients and help them recover. This plasma therapy is considered to be safe & promising. A person who has recovered from Covid can donate his/her plasma to a person who is infected with the corona virus. This system proposed here aims at connecting the donors & the patients by an online Mobile application. By using this Mobile application, the users can either raise a request for plasma donation or requirement. This system is used if anyone needs a Plasma Donor. This system comprises of Admin and User where both can request for a Plasma. In this system there is something called an active user, which means the user is an Active member of the Application and has recovered from Covid 19, only such people are recommended here for Plasma Donation. Both parties can Accept or Reject the request.

3. Blogging & reviewing

Blogging is a great way to disseminate your message in a casual manner. Businesses that blog receives 97% more links to their websites. If you want to see a fellow blood center that is knocking it out of the park with a wonderful blog, take a look at Stanford Blood Center. Every time you publish a blog, it's one page on your website, which means one more opportunity for you to show up on the search engine results page (or SERP) and more organic traffic to your website. Blogging is a great opportunity for your blood centers to stay present on social media and generate engagement to your website, and it allows you to diversify your marketing efforts against other blood centers. Ask donors to provide reviews. Post to their pages. Give them photo props to use for taking selfish in the chair. Have a costume for selfish. Be silly, embrace the fun side of social media, and give your donors something to talk about. In an article, it is said that 92% of consumers trust organic, user generated content more than they trust traditional advertising, so user generated content should be a top-of-mind priority to your blood donor recruitment strategies. Also, 68% of consumers say positive reviews make them trust a local business more. User-generated content can help you get the community connection back to your donor centers. Leverage relationships to get great content!

2.2 References

1.https://www.academia.edu/17573428/Online_Blood_Donation_management_System_reprt

2.https://www.studocu.com/row/document/sindh-madressatul-islam-university/software-engineering/project-report-on-blood-bank-management-system/9154276

3. https://dev.to/nehasoni__/plasma-donation-website-using-mern-stack-26f5

2.3 Problem Statement Definition

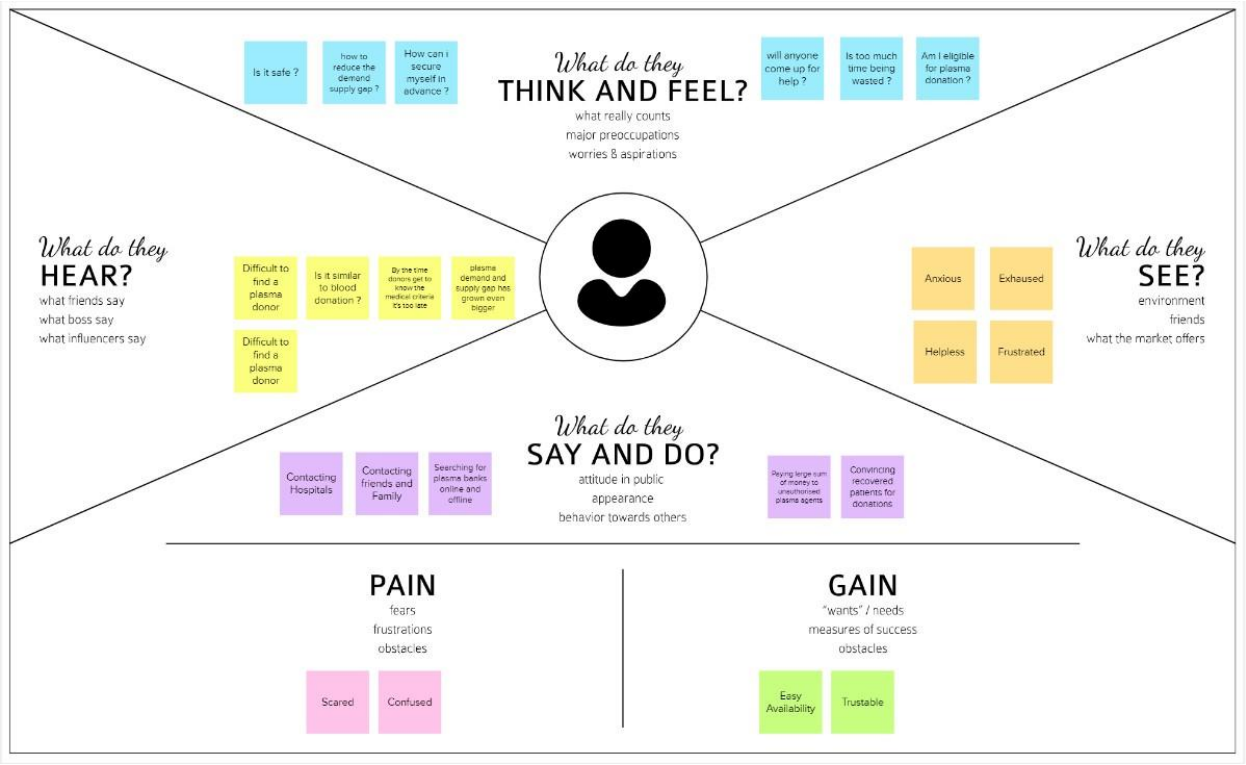
A Plasma is a liquid portion of the blood, over 55% of human blood is plasma. Plasma is used to treat various infectious diseases and it is one of the oldest methods known as plasma therapy. Plasma therapy is a process where blood is donated by recovered patients in order to establish antibodies that fight the infection. For instance, during COVID 19 crisis the requirement for plasma increased drastically as there were no vaccination found in order to treat the infected patients, with plasma therapy the recovery rates where high but the donor count was very low and in such situations it was very important to get the information about the plasma donors. Saving the donor information and notifying about the current donors would be a helping hand as it can save time and help the users to track down the necessary information about the donors.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to helps teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.

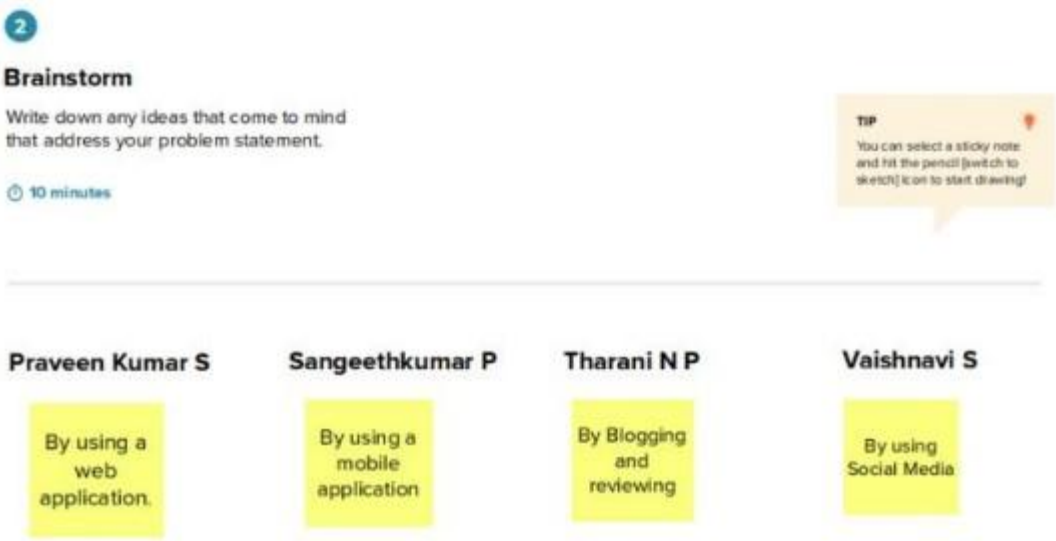
Plasma Donor Application



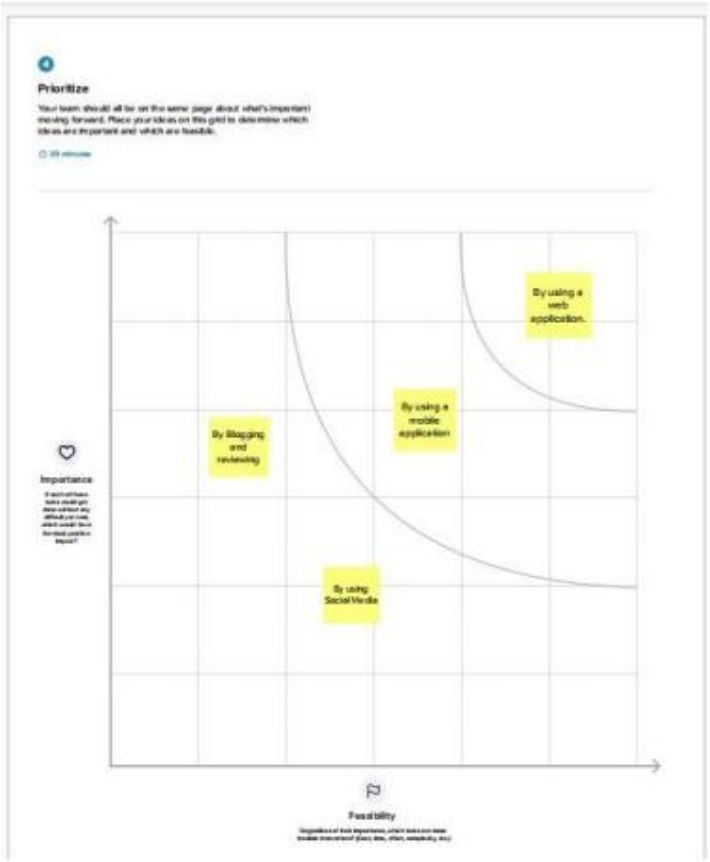
3.2 Ideation & Brainstorming

1. Team Gathering, Collaboration and Select the Problem Statement

2. Brainstorm, Idea Listing and Grouping



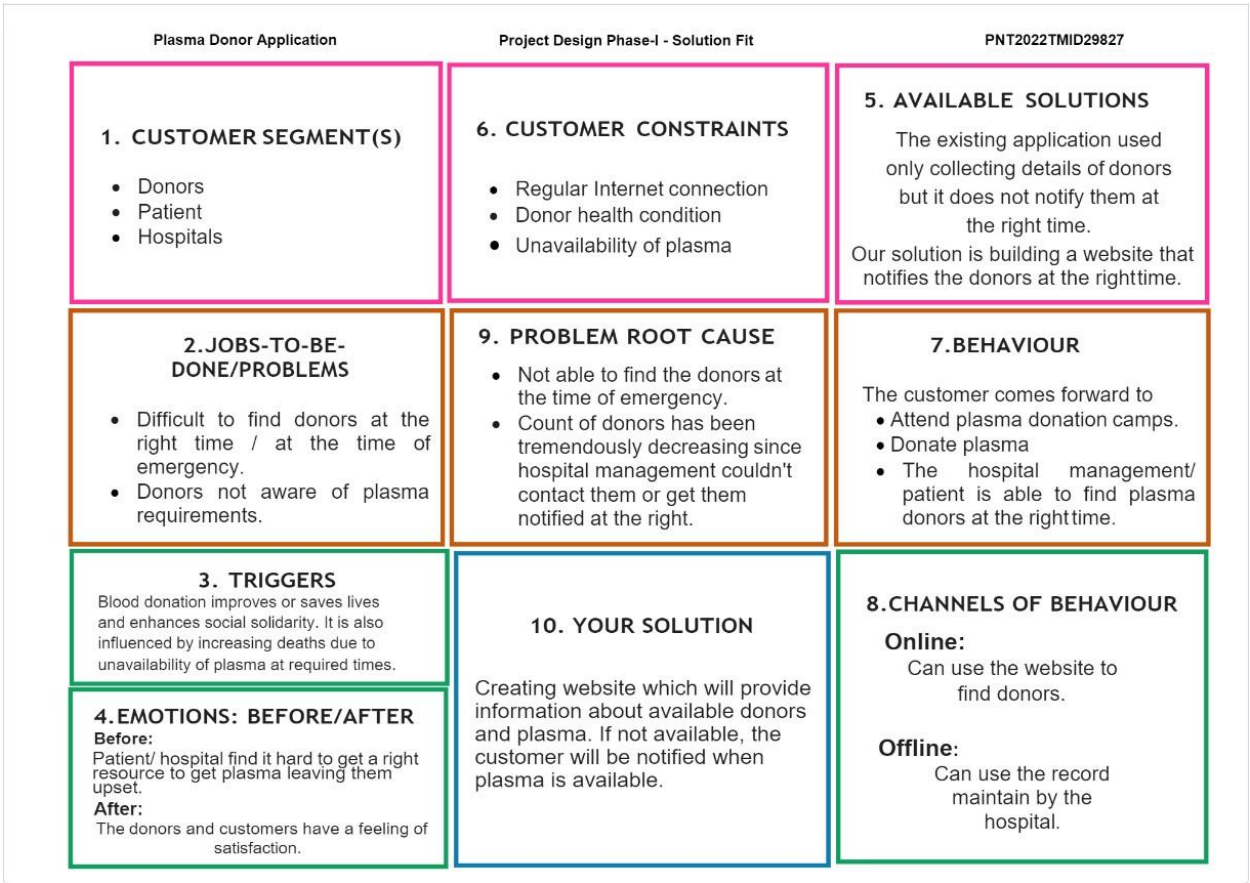
3. Idea Prioritization



3.3 Proposed Solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Easy availability of plasma Maintaining donors list.
2.	Idea / Solution description	Now a day, the demand for plasma is increasing day by day. So to solve this problem we are implement our web application project which stores the donor list in a database and it notifies the donors on a request. we are yet to design a real-time, intelligent, and rational recommendation system using sentiment analysis of the user's feedback, response rate of the donor, and the current Geo-location information and finally develop cross-platform application for blood collection and distribution system.
3.	Novelty / Uniqueness	The proposed solution maintains all the donor's personal information with high security, which is the unique feature of the proposed solution.
4.	Social Impact / Customer Satisfaction	More patients will be benefited through this application.
5.	Business Model (Revenue Model)	Revenue can be generated by connecting patients and donors.
6.	Scalability of the Solution	Whatever may be the situation, the patient will get plasma within 4-5hrs.products by resolving customer complaints.

.4 Problem Solution fit



4. REQUIREMENT ANALYSIS

4.1 Functional Requirements

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
1	User Registration	Registration through Form Registration through Gmail

2	User Confirmation	Confirmation via Email Confirmation via OTP
3	Objective	Describe what the product does
4	End result	Define product features
5	Focus	Focus on user requirements
6	Documentation	Captured in use case

4.2 Non-functional Requirements

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

FR No.	**Non-Functional Requirement**	**Description**
1	Usability	Human Factors, overall aesthetics, consistency and documentation.
2	Security	A system's ability to prohibit unauthorized access, usage or behavior modification while providing service to authorized users.
3	Reliability	Frequency and severity of failure, recoverability, predictability , accuracy and

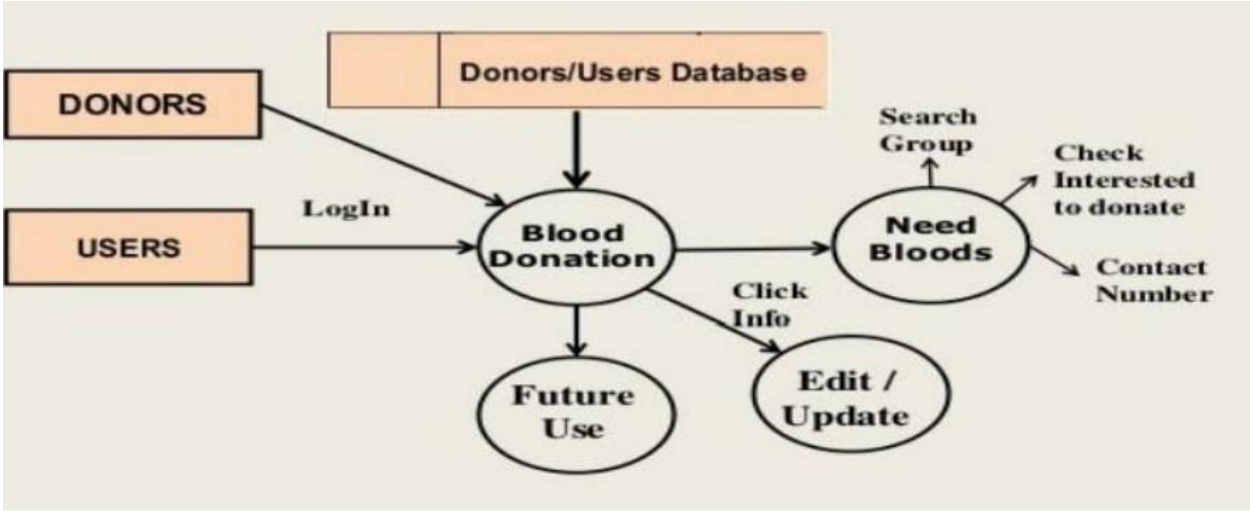
		mean time between failures(MTBF
4	Performance	Processing speed, response time, resource consumption, throughput and efficiency.
5	Availability	How long a system is available for users. This is time the system is not down due to outages or maintenance activities. Mean Time Between Failure (MTBF) is one metric that helps us characterize system availability.
6	Scalability	The ability of solution or system to increase its capacity to serve clients and/or increase processing rates to match demand.

5. PROJECT DESIGN

5.1 Data Flow Diagrams

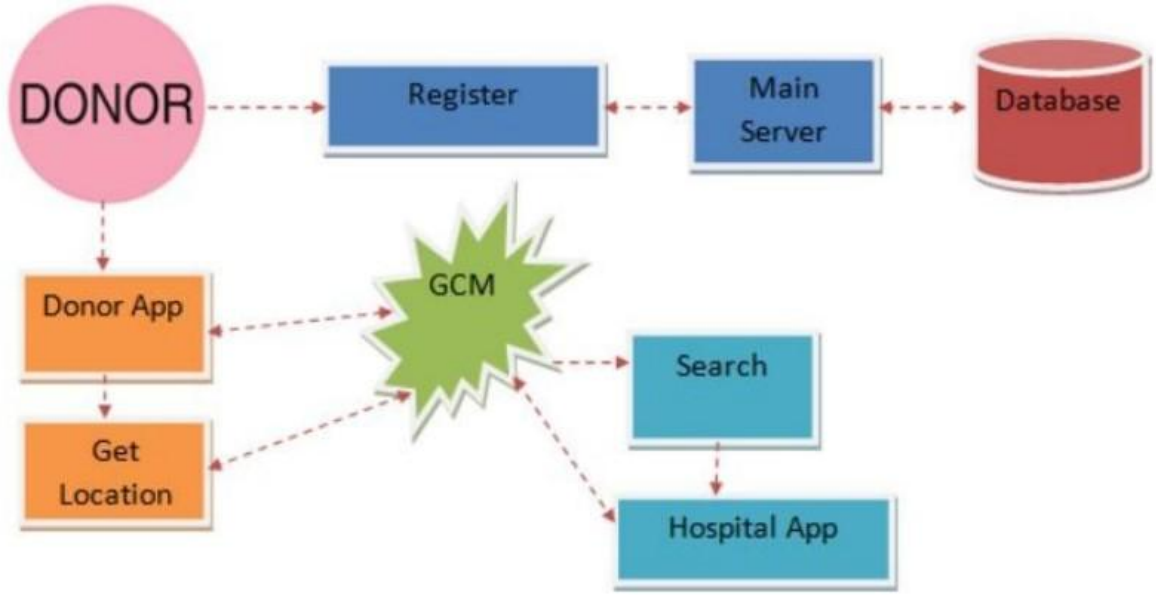
A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.

Level 0

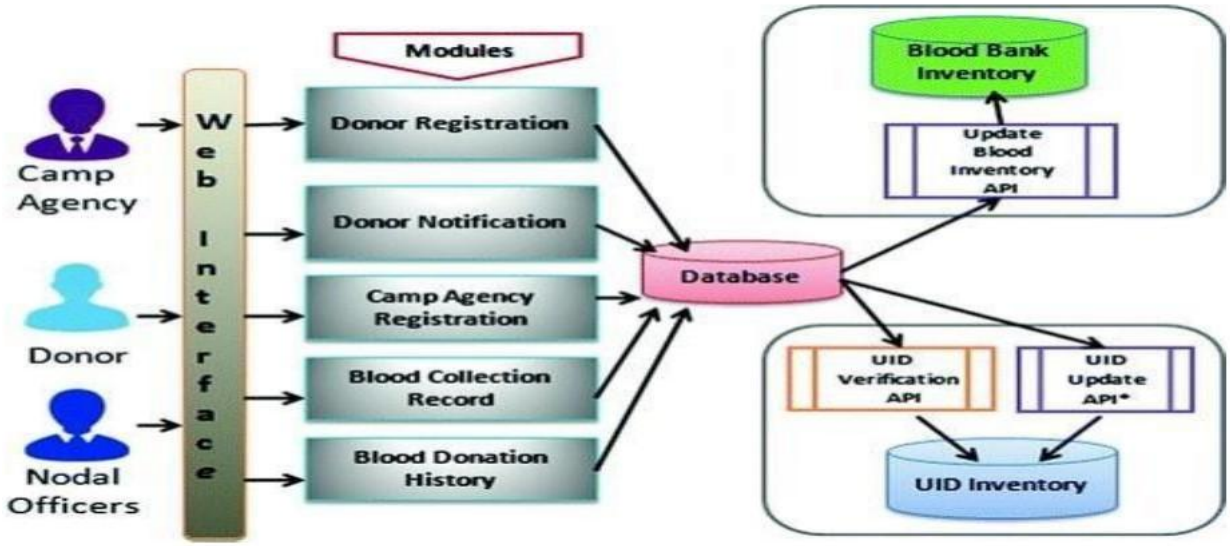


5.2 Solution & Technical Architecture

Level 1



Level 2



5.3 User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Donor	App Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
	login	USN-2	As a customer, I can login to the application by entering correct email and password.	I can access me account/dashboard.	High	Sprint-2

	Register For Donate	USN-3	As a user, I can log into the application and find the current bank to donate plasma and confirm my booking	I can register & access the dashboard with Facebook Login.	Medium	Sprint-3
patient/doctor	Find the bank	USN-4	As a patient, I can directly access the application and find the plasma available bank	I can access my account / dashboard	High	Sprint-1,2
	Request for plasma	USN-5	As a user, I can enter into the application and find the current bank and request for plasma and state the emergency	I can register & access the dashboard with Face book and login.	Medium	Sprint-3
Administrator	Maintain the application	USN-6	As Administrator I can log into the application by entering email & password and maintaining details for users.	I can access my accou nt /dashb oard	High	Sprint-3
	Connect The Bank With Users	USN-7	As Administrator can hold the good communication between bank and user	I can access my account /dashboard	Low	Sprint-4
	Maintain Database	USN-8	As Administrator i can hold the exact details of donor and patient and also bank for requesting and available of plasma.	I can access my account/dash board .	Medium	Sprint-4
Plasma Bank	Connect The Bank With Users	USN-7	As Bank, i can hold the good communication between Administrator and use	I can access my account / dashboard	Medium	Sprint-3

	Maintain Database	USN-8	As Bank i can hold the exact details of donor and patient and also bank for requesting and available of plasma	I can access my account / dashboard	High	Sprint-4
BOT	Help the user my bot mesg in application	USN-9	As AI bot, i can hold the good communication between bank and user also help the use	I can access my account / dashboard	Medium	Sprint-4

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	USN-1 A user can register for the application by entering their email, password, and confirming the password.	3	High	Praveen Kumar S Vaishnavi S Tharani NP
Sprint-1	Email verification	USN-2	A user will receive confirmation email once they have registered for the application.	3	High	Praveen Kumar S SangeethKumar P
Sprint-1		USN-3	A user can register for the application through Google.	2	Medium	Praveen Kumar S Tharani NP
Sprint-1	Login	USN-4	A user can log into the application by entering email & password.	2	High	Praveen Kumar S Vaishnavi S
Sprint-1	Donor Profile	USN-5	A user is able to register themselves as verified plasma donor.	3	High	Praveen Kumar S SangeethKumar P

Sprint-2	Virtual Certificate	USN-6	A user will get a virtual donor certificate after a verified successful plasma donation.	2	Medium	Praveen Kumar S Tharani NP

Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint-2	Plasma Request	USN-7	A verified clinic is able to make a plasma request in the application.	3	High	Praveen Kumar S SangeethKumar P
Sprint-2	Verification of Donor's details	USN-8	We the administrators will verify the details provided by the donors so only the genuine donors are able to use the application.	2	Medium	Praveen Kumar S Tharani NP
Sprint-3	Accept the donation request	USN-9	A user and a registered donor will get a notification to accept the plasma request for their specific blood type.	3	High	Praveen Kumar S SangeethKumar P
Sprint-3	Communication Channel	USN-10	A patient is able to communicate with the donor personally within the application.	3	Medium	Praveen Kumar S Vaishnavi S
Sprint-3		USN-11	A user and a registered donor is able to share their location with the recipient after accepting their plasma request.	3	Medium	Praveen Kumar S Tharani NP

Sprint-3	Administrator	USN-12	An admin will store the registered donor's details after verification into the database.	3	High	Praveen Kumar S SangeethKumar P
Sprint-4	Support	USN-13	A user is able to ask basic question related to plasma donation with the help of chat-bot.	2	Medium	Praveen Kumar S Vaishnavi S

Sprint	**Functional Requirement (Epic)**	**User Story Number**	**User Story / Task**	**Story Points**	**Priority**	**Team Members**
Sprint- 4		USN-14	A user can find the answers for the frequently asked question about the plasma donation in the FAQ section.	3	High	Praveen Kumar S Tharani NP
Sprint- 4	About	USN-15	A new user can read about plasma and plasma donation in dedicated about section.	2	Medium	Praveen Kumar S SangeethKumar P
Sprint-4	Administrator	USN-16	An admin will approve all the plasma transaction in the application after proper verification.	3	High	Praveen Kumar S Vaishnavi S
Sprint-4		USN-17	An admin, I will update the plasma availability and donors count periodically.	3	Medium	Praveen Kumar S Tharani NP

Project Tracker, Velocity & Burndown Chart

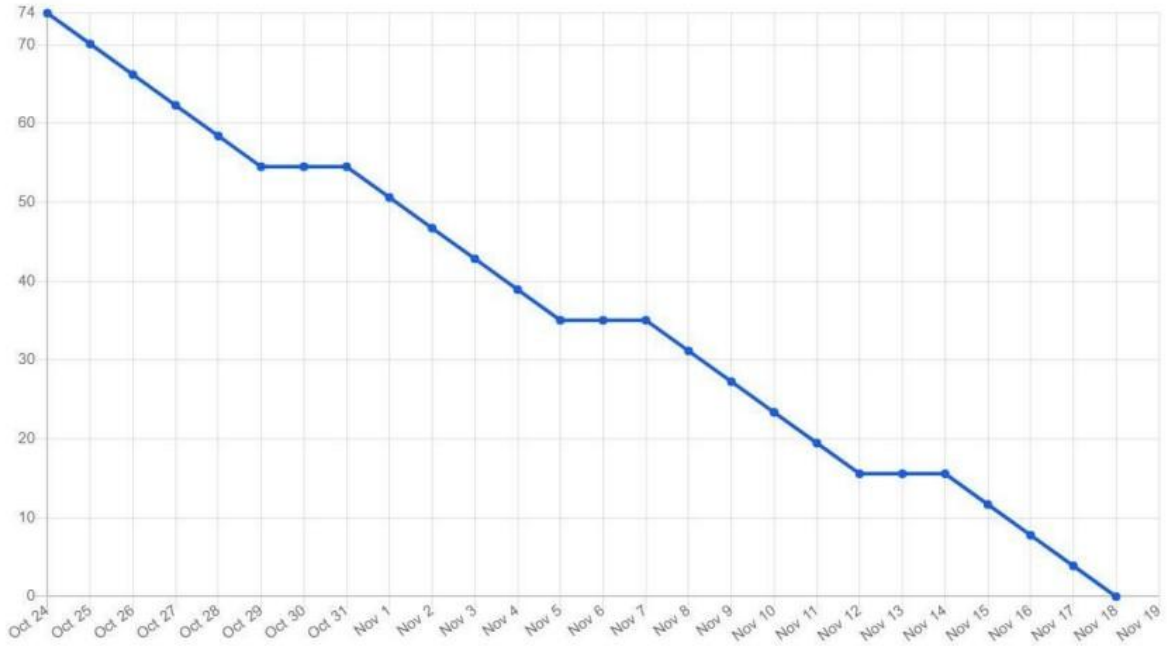
Sprint	Total Story points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date(Actual)
Sprint-1	18	6 Days	24 Oct 2022	29 Oct 2022	18	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	18	6 Days	07 Nov 2022	12 Nov 2022	18	12 Nov 2022
Sprint-4	18	6 Days	14 Nov 2022	19 Nov 2022	18	19 Nov 2022

Velocity:

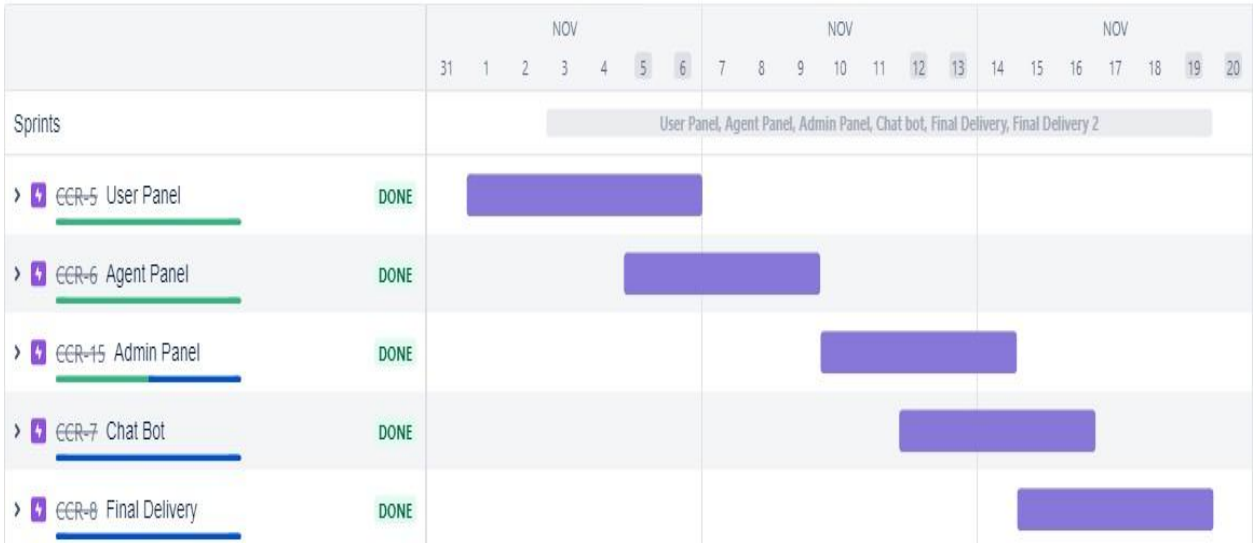
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity(AV) per iteration unit (story points per day).

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Burndown Chart:



6.2 Reports from JIRA



7. CODING & SOLUTIONING

Base.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">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.2/css/all.min.css"/>
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">

    <title>Document</title>

    {% block head %}

    {% endblock %}
</head>
<body>
    {% block body %}

    {% endblock %}
</body>
</html>
```

Feature 1 :

Index.html

It is Landing page of this Application

```
{% extends 'base.html' %}

{% block head %}
  <link rel="stylesheet" href="{{ url_for('static', filename= 'css/style.css') }}">
  <link rel="stylesheet" href="{{ url_for('static', filename= 'css/FAQ.css') }}">
{% endblock %}

{% block body %}
<!-- Move to up button -->
  <div style="background-color: red;color: aliceblue;">{{msg}}</div>
  <div class="scroll-button">
    <a href="#home"><i class="fas fa-arrow-up"></i></a>
  </div>
  <!-- navgaition menu -->
  <nav>
    <div class="navbar">
      <div class="logo"><a href="#">Plasma Bank</a></div>
      <ul class="menu">
          <li><a href="#home">Home</a></li>
          <li><a href="#about">About</a></li>
          <li><a href="#product">Categories</a></li>
```

```
          <li><a href="#faq">FAQ</a></li>
          <li><a href="#contact">Contact</a></li>
          <li><a href="{{ url_for('signup') }}">Signup</a>
          </li>
          <li><a href="{{ url_for('login') }}">Login</a>
          </li>

          <div class="cancel-btn">
            <i class="fas fa-times"></i>
          </div>
      </ul>
      <!-- <div class="media-icons">
        <a href="#"><i class="fab fa-facebook-f"></i></a>
        <a href="#"><i class="fab fa-twitter"></i></a>
        <a href="#"><i class="fab fa-instagram"></i></a>
      </div> -->
    </div>
    <div class="menu-btn">
      <i class="fas fa-bars"></i>
    </div>
  </nav>

<!-- Home Section Start -->
 <section class="home" id="home">
   <div class="home-content">
     <div class="text">
       <div class="text-one">Welcome,</div>
       <div class="text-two">Plasma Bank</div>
       <div class="text-three">Donate Blood, Save Life</div>
       <div class="text-four">From India</div>
     </div>
     <div class="button">
       <a href="/login"><button>Donate Now</button></a>    <a
href="/login"><button>Need Plasma</button></a>
     </div>
   </div>
 </section>

<!-- About Section Start -->
<section class="about" id="about">
  <div class="content">
    <div class="title"><span>About Us</span></div>
  <div class="about-details">
    <div class="left">
      <img src="static/images/home_pic1.jpg" style="height: 100%; width: 100%;" alt="">
    </div>
    <div class="right" style="padding-left: 70px;">
      <div class="topic">Plasma Bank Info</div>
      <p>We aim to provide prompt, economical and reliable services of the safest blood and other blood
products like RBCs, Platelets, et. Offering the industry-leading, advanced technology and well-equipped
inventory. We make all types of blood available for the patients and many hospitals. We contribute in
saving many lives in the time of need or in an emergency.</p>
      <div class="button">

      </div>
    </div>
```

```
  </div>
  </div>
</section>



<!-- My Skill Section Start -->
<!-- Section Tag and Other Div will same where we need to put same CSS -->


<!-- My Services Section Start -->
 <section class="services" id="product">
   <div class="content">
     <div class="title"><span>Categories</span></div>
     <div class="boxes">
       <div class="box">
         <div class="icon">
          <i class="fa fa-heartbeat"></i>
       </div>
       <div class="topic">Whole Blood Donation</div>
       <p>Whole blood is the most flexible type of donation. It can be transfused in its original form, or
used to help multiple people when separated into its specific components of red cells, plasma and
platelets.</p>
     </div>
       <div class="box">
         <div class="icon">
           <i class="fa fa-car"></i>
       </div>
       <div class="topic">Power Red Donation</div>
       <p>During a Power Red donation, you give a concentrated dose of red cells, the part of your blood
used every day for those needing transfusions as part of their care. This type of donation uses an
automated process that separates your red blood cells from the other blood components,
         </p>
     </div>
       <div class="box">
         <div class="icon">
           <i class="fa fa-home"></i>
       </div>
       <div class="topic">Platelet Donation</div>
       <p>In a platelet donation, an apheresis machine collects your platelets along with some plasma,
returning your red cells and most of the plasma back to you. A single donation of platelets can yield
several transfusable units, whereas it takes about five whole blood donations to make up a single
transfusable unit of platelets.      </p>
     </div>
       <div class="box">
         <div class="icon">
           <i class="fa fa-plug"></i>
       </div>
       <div class="topic">Plasma Donation</div>
       <p>During an AB Elite donation, you give plasma, a part of your blood used to treat patients in
emergency situations. AB plasma can be given to anyone regardless of their blood type. Plasma is collected
through an automated process that separates plasma from other blood components,       </p>
     </div>
       <div class="box">
         <div class="icon">
           <i class="fab fa-android"></i>
```

```
        </div>
        <div class="topic">About Blood Types</div>
        <p>There are actually more than 8 different blood types, some of which are not compatible with each
other. Find out how your blood type can help hospital patients in need of a transfusion.      </p>
      </div>
        <div class="box">
          <div class="icon">
            <i class="fa fa-cart-plus"></i>
        </div>
        <div class="topic">About Blood Components</div>
        <p>During medical treatment, patients may receive whole blood or just the specific blood components
they need. Learn more about how blood components impact patient transfusions.      </p>
      </div>
    </div>
    </div>
</section>



<div class="section" id="faq">
         <div class="content">
           <h1 class="title">FAQ Section</h1>
           <div class="container-1">

               <div class="faq">
                   <div class="question">
                       <h2>Who can donate?</h2>
                       <i class="fa fa-arrow-circle-o-right"></i>
                   </div>
                   <div class="answer">
                       <p>Generally, plasma donors must be 18 years of age and weigh at least 110 pounds
(50kg). All individuals must pass two separate medical examinations, a medical history screening and
testing for transmissible viruses, before their donated plasma can be used to manufacture plasma protein
therapies.</p>
                   </div>
               </div>

               <div class="faq">
                   <div class="question">
                       <h2>Does it hurt?</h2>
                       <i class="fa fa-arrow-circle-o-right"></i>
                   </div>
                   <div class="answer">
                       <p>Most people compare the feeling of the needle to a mild bee sting. You will also
be required to submit to a finger stick test each time you donate so the collection center medical staff
can evaluate your protein and hemoglobin levels.                      </p>
                   </div>
               </div>

               <div class="faq">
                   <div class="question">
                       <h2>Is donating plasma safe?                      </h2>
                       <i class="fa fa-arrow-circle-o-right"></i>
                   </div>
                   <div class="answer">
```

```
                    <p>Yes. Plasma donation in IQPP certified collection centers is performed in a
highly controlled, sterile environment by professionally trained medical staff. All plasma collection
equipment is sterilized and any equipment that comes into contact with you is used only once to eliminate
the possibility of transmitting viral infections.
                    </p>
                  </div>
              </div>

          </div></div>

        <div class="container-2">

            <div class="faq">
                <div class="question">
                    <h2>How long does it take?
                    </h2>
                    <i class="fa fa-arrow-circle-o-right"></i>
                </div>
                <div class="answer">
                    <p>Your first donation will take approximately 2 hours. Return visits on average take
about 90 minutes.
                    </p>
                </div>
            </div>

            <div class="faq">
                <div class="question">
                    <h2>What do you do with my plasma?
                    </h2>
                    <i class="fa fa-arrow-circle-o-right"></i>
                </div>
                <div class="answer">
                    <p>Nearly 500 different types of proteins have been found in human blood plasma.
Approximately 150 of these may be used for diagnosing disease or manufacturing therapies.
                    </p>
                </div>
            </div>

            <div class="faq">
                <div class="question">
                    <h2>What type of medical screening and testing is done?
                    </h2>
                    <i class="fa fa-arrow-circle-o-right"></i>
                </div>
                <div class="answer">
                    <p>You must have a pre-donation physical which includes answering medical history
questions, tests for viruses such as HIV and Hepatitis and evaluating your protein and hemoglobin levels.
                    </p>
                </div>
            </div>

        </div>

    </div>
```

```
<!--<section class="contact" id="contact">
  <div class="content">
    <div class="title"><span>click button </span></div>
    <div class="text">

      <div class="button">
        <p><a href="../register.html"><button>register now</button></a></p>

      </div>
    </div>
  </div>
</section>
<br>
<br>-->


<!--<section class="contact" id="contact">
  <div class="content">
    <div class="title"><span>click button </span></div>
    <div class="text">

      <div class="button">
        <p><a href="../Customer Care Registry/login.html"><button>Sign up</button></a></p>

      </div>
    </div>
  </div>
</section>
<br>
<br>-->


<!-- Contact Me section Start -->
<section class="contact" id="contact">
  <div class="content">
    <div class="title"><span>Contact Us</span></div>
    <div class="text">
      <div class="button">
        <button>Let's Chat</button>
        <br><br>
        <div class="media-icons">
          <a href="#"><i class="fab fa-facebook-f" style="font-size:
30px;"></i></a>      
          <a href="#"><i class="fab fa-twitter" style="font-size:
30px;"></i></a>      
          <a href="#"><i class="fab fa-instagram" style="font-size: 30px;"></i></a>
        </div>
      </div>
    </div>
  </div>
</section>


<!-- Footer Section Start -->
<footer>
  <div class="text">
```

```
    <span>Created By   </span>
    <br>
    <span>Praveenkumar.S   |</span>
    <span>Sangeeth Kumar P  |</span>
    <span>Tharani N P  |</span>
    <span>Vaishnavi.S</span>
  </div>
</footer>
  <script>
    window.watsonAssistantChatOptions = {
      integrationID: "0032bdd2-5b59-4312-a00c-cece7ea268f0", // The ID of this integration.
      region: "au-syd", // The region your integration is hosted in.
      serviceInstanceID: "30b01793-f193-4ada-a147-4610ae753688", // 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);
    });

    // Sticky Navigation Menu JS Code
let nav = document.querySelector("nav");
let scrollBtn = document.querySelector(".scroll-button a");
console.log(scrollBtn);
let val;
window.onscroll = function() {
  if(document.documentElement.scrollTop > 20){
    nav.classList.add("sticky");
    scrollBtn.style.display = "block";
  }else{
    nav.classList.remove("sticky");
    scrollBtn.style.display = "none";
  }

}

// Side NavIgation Menu JS Code
let body = document.querySelector("body");
let navBar = document.querySelector(".navbar");
let menuBtn = document.querySelector(".menu-btn");
let cancelBtn = document.querySelector(".cancel-btn");
menuBtn.onclick = function(){
  navBar.classList.add("active");
  menuBtn.style.opacity = "0";
  menuBtn.style.pointerEvents = "none";
  body.style.overflow = "hidden";
  scrollBtn.style.pointerEvents = "none";
}
cancelBtn.onclick = function(){
  navBar.classList.remove("active");
  menuBtn.style.opacity = "1";
  menuBtn.style.pointerEvents = "auto";
  body.style.overflow = "auto";
  scrollBtn.style.pointerEvents = "auto";
```

```
}

// Side Navigation Bar Close While We Click On Navigation Links
let navLinks = document.querySelectorAll(".menu li a");
for (var i = 0; i < navLinks.length; i++) {
  navLinks[i].addEventListener("click" , function() {
    navBar.classList.remove("active");
    menuBtn.style.opacity = "1";
    menuBtn.style.pointerEvents = "auto";
  });
}


//faq



const question = document.querySelectorAll('.faq');

question.forEach(faq => {
    faq.addEventListener("click", () => {
        faq.classList.toggle("active");
    })
})
</script>
{% endblock %}
```

Feature 2 :

Signup.html :

This page contains a form. It will be used by new Users to register.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename= 'css/reg.css') }}" />
</head>
<body>


<div class="container" id="container">
    <div class="form-container sign-up-container">

    </div>
    <div class="form-container sign-in-container">
        <form action="{{ url_for('register') }}" method="post">
            <h1>Register now</h1><br>


            <input type="text" placeholder="name"  name="uname"/>
            <input type="email" placeholder="Email" name="email"/>
            <input type="text" placeholder="phone no" name="phone"/>
            <input type="password" placeholder="creat new Password" name="password"/>
            <input type="password" placeholder="Confirm new Password" />
```

```
            <input type="text" name="bloodgroup" placeholder="Your Blood Group">
            <p>Already have an account <a href="{{ url_for('login') }}">login</a></p>
            <br>
            <button type="submit">register</button>
        </form>
    </div>
    <div class="overlay-container">
        <div class="overlay">
            <div class="overlay-panel overlay-left">


            </div>
            <div class="overlay-panel overlay-right">
                <h1>I'm Admin!</h1>
                <p>Enter your personal details and start journey </p>
                <img src="static/images/admin.png" width="200px">
            </div>
        </div>
    </div>
</div>



</body>
```

Login.html :

This page is Login page. The verification of userid and password is done once the user enter their login crwedentials.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename= 'css/login.css') }}" />
</head>
<body>


<div class="container" id="container">
    <div class="form-container sign-up-container">

    </div>
    <div class="form-container sign-in-container">
        <form action="{{url_for('signin')}}" method="POST">
            <h1>Sign in</h1>
            <span>or use your account</span>
            <br>
            <input type="email" placeholder="Email" name="email" />
            <input type="password" placeholder="Password" name="password" />
            <p>Don't have an account <a href="{{ url_for('signup') }}">Sign up</a></p>
            <button>Sign In</button>
        </form>
    </div>
    <div class="overlay-container">
        <div class="overlay">
```

```
            <div class="overlay-panel overlay-left">
                <h1>Welcome Back!</h1>
                <p>To keep connected with us please login with your personal info</p>
                <button class="ghost" id="signIn">Sign In</button>
            </div>
            <div class="overlay-panel overlay-right">
                <h1>Hello, Friend!</h1>
                <p>Enter your personal details and start journey with us</p>
                <img src="static/images/lady0.png" width="200px">
            </div>
        </div>
    </div>
</div>



</body>
```

Feature 3 :

Plasmareq.html :

This Page contains a form. In which, the receiver fill their details (needed blood group, place, district….)

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename = 'css/reg.css') }}" />
</head>
<body>


<div class="container" id="container">
    <div class="form-container sign-up-container">

    </div>
    <div class="form-container sign-in-container">
        <form action="{{ url_for('needplasma') }}" method="post">
            <h1>Request now</h1><br>


            <input type="text" placeholder="name"  name="uname" required>
            <input type="text" placeholder="phone no" name="phone" required>
            <input type="text" name="bloodgroup" placeholder="Your Blood Group" required>
            <input type="text" name="place" placeholder="Your Location" required>
            <input type="text" name="district" placeholder="Your District" value="" required>
            <!-- <p>Already have an account <a href="{{ url_for('login') }}">login</a></p> -->
            <br>
            <button type="submit">request</button>
        </form>
    </div>
    <div class="overlay-container">
        <div class="overlay">
```

```
            <div class="overlay-panel overlay-left">


            </div>
            <div class="overlay-panel overlay-right">
                <h1>I'm Admin!</h1>
                <p>Enter your personal details and start journey </p>
                <img src="static/images/admin.png" width="200px">
            </div>
        </div>
    </div>
</div>



</body>
```

Comments.html :

This file show the output with respect to the input given. The datas are fetched from the database and appended in it based on the users request.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
<link rel="stylesheet" type="text/css" href="{{ url_for('static', filename= 'css/login.css') }}" />
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
<style>
    body{
        background-color: #87aca3;
    }
</style>
</head>
<body>


<div class="container" id="container" style="padding: 10px;">
    <div class="form-container sign-up-container">

    </div>

    <div class="form-container sign-in-container" style="padding: 10px;">
         <h3 style="text-align: center;  font-family: 'Ubuntu', sans-serif;">Results</h3>
         <br><br>
        <table class="table" style="padding: 20px;">
            <thead class="thead-dark">
                <tr>
                  <th scope="col">NAME</th>
                  <th scope="col">PLACE</th>
                  <th scope="col">PHONENUMBER</th>

                </tr>
              </thead>
```

```
              <tbody>
            {% for row in apcustomer %}
                                    <tr>
                                        <td style="color: black; padding-right:
10px;">{{row["NAME"]}}</td>
                                        <td style="color: black; padding-right:
10px;">{{row["PLACE"]}}</td>
                                        <td style="color: black;">{{row["PHONENUMBER"]}}</td>
                                        <!-- <td style="color: black;">{{row["PASSWORD"]}}</td> -->
                                        <!-- <td style="color: black;">{{row["BLOODGROUP"]}}</td> -->
                                    </tr>
            {% endfor %}

            {% for row in ancustomer %}
                                    <tr>
                                        <td style="color: black; padding-right:
10px;">{{row["NAME"]}}</td>
                                        <td style="color: black; padding-right:
10px;">{{row["PLACE"]}}</td>
                                        <td style="color: black;">{{row["PHONENUMBER"]}}</td>
                                        <!-- <td style="color: black;">{{row["PASSWORD"]}}</td>
                                        <td style="color: black;">{{row["BLOODGROUP"]}}</td> -->
                                    </tr>
            {% endfor %}
            {% for row in bpcustomer %}
                                    <tr>
                                        <td style="color: black; padding-right:
10px;">{{row["NAME"]}}</td>
                                        <td style="color: black; padding-right:
10px;">{{row["PLACE"]}}</td>
                                        <td style="color: black;">{{row["PHONENUMBER"]}}</td>
                                        <!-- <td style="color: black;">{{row["PASSWORD"]}}</td>
                                        <td style="color: black;">{{row["BLOODGROUP"]}}</td> -->
                                    </tr>
            {% endfor %}

            {% for row in bncustomer %}
                                    <tr>
                                        <td style="color: black; padding-right:
10px;">{{row["NAME"]}}</td>
                                        <td style="color: black; padding-right:
10px;">{{row["PLACE"]}}</td>
                                        <td style="color: black;">{{row["PHONENUMBER"]}}</td>
                                        <!-- <td style="color: black;">{{row["PASSWORD"]}}</td>
                                        <td style="color: black;">{{row["BLOODGROUP"]}}</td> -->
                                    </tr>
            {% endfor %}

            {% for row in abpcustomer %}
                                    <tr>
                                        <td style="color: black; padding-right:
10px;">{{row["NAME"]}}</td>
                                        <td style="color: black; padding-right:
10px;">{{row["PLACE"]}}</td>
                                        <td style="color: black;">{{row["PHONENUMBER"]}}</td>
                                        <!-- <td style="color: black;">{{row["PASSWORD"]}}</td>
```

```
                                        <td style="color: black;">{{row["BLOODGROUP"]}}</td> -->
                                    </tr>
            {% endfor %}

            {% for row in abncustomer %}
                                    <tr>
                                        <td style="color: black; padding-right:
10px;">{{row["NAME"]}}</td>
                                        <td style="color: black; padding-right:
10px;">{{row["PLACE"]}}</td>
                                        <td style="color: black;">{{row["PHONENUMBER"]}}</td>
                                        <!-- <td style="color: black;">{{row["PASSWORD"]}}</td>
                                        <td style="color: black;">{{row["BLOODGROUP"]}}</td> -->
                                    </tr>
            {% endfor %}
            {% for row in opcustomer %}
                                    <tr>
                                        <td style="color: black; padding-right:
10px;">{{row["NAME"]}}</td>
                                        <td style="color: black; padding-right:
10px;">{{row["PLACE"]}}</td>
                                        <td style="color: black;">{{row["PHONENUMBER"]}}</td>
                                        <!-- <td style="color: black;">{{row["PASSWORD"]}}</td>
                                        <td style="color: black;">{{row["BLOODGROUP"]}}</td> -->
                                    </tr>
            {% endfor %}

            {% for row in oncustomer %}
                                    <tr>
                                        <td style="color: black; padding-right:
10px;">{{row["NAME"]}}</td>
                                        <td style="color: black; padding-right:
10px;">{{row["PLACE"]}}</td>
                                        <td style="color: black;">{{row["PHONENUMBER"]}}</td>
                                        <!-- <td style="color: black;">{{row["PASSWORD"]}}</td>
                                        <td style="color: black;">{{row["BLOODGROUP"]}}</td> -->
                                    </tr>
            {% endfor %}
        </tbody>
        </table>
    </div>
    <div class="overlay-container">
        <div class="overlay">
            <div class="overlay-panel overlay-left">
                <h1>Welcome Back!</h1>
                <p>To keep connected with us please login with your personal info</p>
                <button class="ghost" id="signIn">Sign In</button>
            </div>
            <div class="overlay-panel overlay-right">
                <h1>Hello, Friend!</h1>
                <p>You Have got a result.</p><br>
                <!-- <div style="height: 100px; width: 100px; border: 2px solid black; background-image:
url('../static/images/happy.jpg');"></div> -->
                <img src="../static/images/happy1.png" width="200px">
            </div>
        </div>
```

```
    </div>
    </div>
    <script>

        function openURL()
{
    var loc = document.getElementByID('location').value;
    var url = 'http://www.localhost:5000/adminname/' + encodeURIComponent(loc);
    // In current window
    // In new window
    window.open(url);
}
    </script>
</body>
```

Feature 4 :

Complaints.html :

This HTML file designed to get the donors details

```
<!DOCTYPE html>
<head>
    <title>Donate Plasma</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='css/complaint.css') }}">
</head>
<body>
    <div class="contact-us">
      <form action="{{ url_for('donateplasma') }}" method="post">
        <h1>Donate now</h1><br>


              <input type="text" placeholder="name"  name="uname"/>
              <input type="text" placeholder="phone no" name="phone"/>
        <input type="text" name="bloodgroup" placeholder="Your Blood Group">
              <input type="text" name="place" placeholder="Your Location">
              <input type="text" name="district" placeholder="Your District">
        <!-- <p>Already have an account <a href="{{ url_for('login') }}">login</a></p> -->
              <br>
        <button type="submit" style="margin-left: 45px;">Donate</button>
      </form>
      </div>
</body>
</html>
```

Style.css :

This file includes all the css styles involved in index.html page

```
/* Google Font CDN Link */
@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&family=Ubuntu:wght@400;500;700&
display=swap');
*{
  margin: 0;
  padding: 0;
```

```
  box-sizing: border-box;
  text-decoration: none;
  scroll-behavior: smooth;
}

/* Custom Scroll Bar CSS */
::-webkit-scrollbar {
    width: 10px;
}
::-webkit-scrollbar-track {
    background: #f1f1f1;
}
::-webkit-scrollbar-thumb {
    background: #6e93f7;
    border-radius: 12px;
    transition: all 0.3s ease;
}

::-webkit-scrollbar-thumb:hover {
    background: #4d69d5;
}
/* navbar styling */
nav{
  position: fixed;
  width: 100%;
  padding: 20px 0;
  z-index: 998;
  transition: all 0.3s ease;
  font-family: 'Ubuntu', sans-serif;
}
nav.sticky{
  background: #082187;
  padding: 13px 0;
}
nav .navbar{
  width: 90%;
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin: auto;
}
nav .navbar .logo a{
  font-weight: 500;
  font-size: 35px;
  color: #4d69d5;
}
nav.sticky .navbar .logo a{
  color: #fff;
}
nav .navbar .menu{
  display: flex;
  position: relative;
}
nav .navbar .menu li{
  list-style: none;
  margin: 0 8px;
```

```
}
.navbar .menu a{
  font-size: 18px;
  font-weight: 500;
  color: #0E2431;
  padding: 6px 0;
  transition: all 0.4s ease;
}
.navbar .menu a:hover{
  color: #4d69d5;
}
nav.sticky .menu a{
  color: #FFF;
}
nav.sticky .menu a:hover{
  color: #0E2431;
}
.navbar .media-icons a{
  color: #4d69d5;
  font-size: 18px;
  margin: 0 6px;
}
nav.sticky .media-icons a{
  color: #FFF;
}

#signup{
  background: none;
  font-size: 18px;
  font-weight: 500;
  color: #0E2431;
  padding: 8px 16px;
  transition: all 0.4s ease;
  font-family: 'Ubuntu', sans-serif;
  border: 1px solid transparent;
  border-color: transparent transparent rgba(0, 0, 0, 0.1) transparent;
  cursor: pointer;
  user-select: none;
  position: relative;
}
#signin{
  background: none;
  font-size: 18px;
  font-weight: 500;
  color: #0E2431;
  padding: 8px 16px;
  transition: all 0.4s ease;
  font-family: 'Ubuntu', sans-serif;
  border: 1px solid transparent;
  border-color: transparent transparent rgba(0, 0, 0, 0.1) transparent;
  cursor: pointer;
  user-select: none;
  position: relative;
}
```

```
/* Side Navigation Menu Button CSS */
nav .menu-btn,
.navbar .menu .cancel-btn{
  position: absolute;
  color: #fff;
  right: 30px;
  top: 20px;
  font-size: 20px;
  cursor: pointer;
  transition: all 0.3s ease;
  display: none;
}
nav .menu-btn{
  color: #4d69d5;
}
nav.sticky .menu-btn{
  color: #FFF;
}
.navbar .menu .menu-btn{
  color: #fff;
}

/* home section styling */
.home{
  height: 100vh;
  width: 100%;
  background: url("../images/bgimg.png") no-repeat;
  background-size: 50% 50%;
  background-position: right;
  background-attachment: fixed;
  font-family: 'Ubuntu', sans-serif;
}
.home .home-content{
  width: 90%;
  height: 100%;
  margin: auto;
  display: flex;
  flex-direction: column;
  justify-content: center;
}
.home .text-one{
  font-size: 25px;
  color: #0E2431;
}
.home .text-two{
  color: #0E2431;
  font-size: 75px;
  font-weight: 600;
  margin-left: -3px;
}
.home .text-three{
  font-size: 40px;
  margin: 5px 0;
  color: #4d69d5;
}
.home .text-four{
```

```
  font-size: 23px;
  margin: 5px 0;
  color: #0E2431;
}


/* About Section Styling */
/* Those Elements Where We Have Apply Same CSS,
 I'm Selecting Directly 'Section Tag' and 'Class'  */
section{
  padding-top: 40px;
}
section .content{
  width: 80%;
  margin: 40px auto;
  font-family: 'Poppins', sans-serif;
}
.about .about-details{
  display: flex;
  justify-content: space-between;
  align-items: center;
}
section .title{
  display: flex;
  justify-content: center;
  margin-bottom: 40px;
}
section .title span{
  color: #0E2431;
  font-size: 30px;
  font-weight: 600;
  position: relative;
  padding-bottom: 8px;
}
section .title span::before,
section .title span::after{
  content: '';
  position: absolute;
  height: 3px;
  width: 100%;
  background: #4d69d5;
  left: 0;
  bottom: 0;
}
section .title span::after{
  bottom: -7px;
  width: 70%;
  left: 50%;
  transform: translateX(-50%);
}
.about .about-details .left{
  width: 45%;
}
.about .left img{
  height: 400px;
  width: 400px;
```

```
  object-fit: cover;
  border-radius: 12px;
}
.about-details .right{
  width: 55%;
}
section  .topic{
  color: #0E2431;
  font-size: 25px;
  font-weight: 500;
  margin-bottom: 10px;
}
.about-details .right p{
  text-align: justify;
  color: #0E2431;
}
section .button{
  margin: 16px 0;
}
section .button button{
  outline: none;
  padding: 8px 16px;
  border-radius: 4px;
  font-size: 25px;
  font-weight: 400;
  background: #4d69d5;
  color: #fff;
  border: 2px solid transparent;
  cursor: pointer;
  transition: all 0.4s ease;
}
section .button button:hover{
  border-color: #4d69d5;
  background-color: #fff;
  color: #4d69d5;
}

 /* My Skills CSS */
 .skills{
   background: #F0F8FF;
 }
 .skills .content{
   padding: 40px 0;
 }
 .skills .skills-details{
   display: flex;
   justify-content: space-between;
   align-items: center;
 }
 .skills-details .text{
   width: 50%;
 }
 .skills-details p{
   color: #0E2431;
   text-align: justify;
 }
```

```
.skills .skills-details .experience{
  display: flex;
  align-items: center;
  margin: 0 10px;
}
.skills-details .experience .num{
  color: #0E2431;
  font-size: 80px;
}
.skills-details .experience .exp{
  color: #0E2431;
  margin-left: 20px;
  font-size: 18px;
  font-weight: 500;
  margin: 0 6px;
}
.skills-details .boxes{
  width: 45%;
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
}
.skills-details .box{
  width: calc(100% / 2 - 20px);
  margin: 20px 0;
}
.skills-details .boxes .topic{
  font-size: 20px;
  color: #4d69d5;
}
.skills-details .boxes .per{
  font-size: 60px;
  color: #4d69d5;
}

/* My Services CSS */
.services .boxes{
  display: flex;
  flex-wrap: wrap;
  justify-content: space-between;
}
.services .boxes .box{
  margin: 20px 0;
  width: calc(100% / 3 - 20px);
  text-align: center;
  border-radius: 12px;
  padding: 30px 10px;
  box-shadow: 0 5px 10px rgba(0, 0, 0, 0.12);
  cursor: default;
  transition: all 0.4s ease;
}
.services .boxes .box:hover{
  background: #4d69d5;
  color: #fff;
}
.services .boxes .box .icon{
```

```
  height: 50px;
  width: 50px;
  background: #4d69d5;
  border-radius: 50%;
  text-align: center;
  line-height: 50px;
  font-size: 18px;
  color: #fff;
  margin: 0 auto 10px auto;
  transition: all 0.4s ease;
}
.boxes .box:hover .icon{
  background-color: #fff;
  color: #4d69d5;
}
.services .boxes .box:hover .topic,
.services .boxes .box:hover p{
  color: #0E2431;
  transition: all 0.4s ease;
}
.services .boxes .box:hover .topic,
.services .boxes .box:hover p{
  color: #fff;
}




/* Contact Me CSS */
.contact{
  background: #F0F8FF;
}
.contact .content{
  margin: 0 auto;
  padding: 30px 0;
}
.contact .text{
  width: 80%;
  text-align: center;
  margin: auto;
}

/* Footer CSS */
footer{
  background: #4d69d5;
  padding: 15px 0;
  text-align: center;
  font-family: 'Poppins', sans-serif;
}
footer .text span{
  font-size: 17px;
  font-weight: 400;
  color: #fff;
}
footer .text span a{
```

```
  font-weight: 500;
  color: #FFF;
}
footer .text span a:hover{
  text-decoration: underline;
}
/* Scroll TO Top Button CSS */
.scroll-button a{
  position: fixed;
  bottom: 20px;
  right: 20px;
  color: #fff;
  background: #4d69d5;
  padding: 7px 12px;;
  font-size: 18px;
  border-radius: 6px;
  box-shadow: rgba(0, 0, 0, 0.15);
  display: none;
}

/* Responsive Media Query */
@media (max-width: 1190px) {
  section .content{
    width: 85%;
  }
}
@media (max-width: 1000px) {
  .about .about-details{
    justify-content: center;
    flex-direction: column;
  }
  .about .about-details .left{
    display: flex;
    justify-content: center;
    width: 100%;
  }
  .about-details .right{
    width: 90%;
    margin: 40px 0;
  }
  .services .boxes .box{
    margin: 20px 0;
    width: calc(100% / 2 - 20px);
  }
}
@media (max-width: 900px) {
  .about .left img{
    height: 350px;
    width: 350px;
  }
}

@media (max-width: 750px) {
  nav .navbar{
    width: 90%;
  }
```

```
nav .navbar .menu{
  position: fixed;
  left: -100%;
  top: 0;
  background: #0E2431;
  height: 100vh;
  max-width: 400px;
  width: 100%;
  padding-top: 60px;
  flex-direction: column;
  align-items: center;
  transition: all 0.5s ease;
}
.navbar.active .menu{
  left: 0;
}
nav .navbar .menu a{
  font-size: 23px;
  display: block;
  color: #fff;
  margin: 10px 0;
}
nav.sticky .menu a:hover{
  color: #4d69d5;
}
nav .navbar .media-icons{
  display: none;
}
nav .menu-btn,
.navbar .menu .cancel-btn{
  display: block;
}
.home .text-two{
  font-size: 65px;
}
.home .text-three{
  font-size: 35px;
}
.skills .skills-details{
  align-items: center;
  justify-content: center;
  flex-direction: column;
}
.skills-details .text{
  width: 100%;
  margin-bottom: 50px;
}
.skills-details .boxes{
  justify-content: center;
  align-items: center;
  width: 100%;
}
.services .boxes .box{
  margin: 20px 0;
  width: 100%;
}
```

```
  .contact .text{
    width: 100%;
}
}

@media (max-width: 500px){
  .home .text-two{
    font-size: 55px;
  }
  .home .text-three{
    font-size: 33px;
  }
  .skills-details .boxes .per{
    font-size: 50px;
    color: #4d69d5;
  }
}



/*comments*/
```

App.py :

This file includes all the python code, database connectivity and fast2sms service codes in it.

```
import email
from email import message
from importlib.resources import contents
from tkinter import S
from turtle import title
from flask import Flask,redirect, render_template, request, session, url_for, flash
# from flask_restful import Resource, Api, reqparse
import sendgrid
import sys
import os
import json
import requests
from pyexpat import model
from sqlalchemy import PrimaryKeyConstraint
from werkzeug.utils import secure_filename
import ibm_db
from flask_mail import Mail, Message
from markupsafe import escape

# import required module
import requests
import json
# mention url
url = "https://www.fast2sms.com/dev/bulkV2"
```

```
# create a dictionary


# create a dictionary
headers = {
    'authorization': 'QqbHW076SFDTledzUu4yhiYNIK2tf3LEnkc9Br5ZasOjp1VwxMLsyMZXA8IUPcEbdB6GJgvnDhwFfV2a',
    'Content-Type': "application/x-www-form-urlencoded",
    'Cache-Control': "no-cache"
}
# make a post request



app = Flask(__name__)

app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=3883e7e4-18f5-4afe-be8c-
fa31c41761d2.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=31498;SECURITY=SSL;SSLServerCertificate=D
igiCertGlobalRootCA.crt;UID=bvx19292;PWD=yDuuJH7Oqdzbnxnk;", "", "")
print(conn)
print("connection successful...")




@app.route('/')
def home():
    message = "TEAM ID : PNT2022TMID37544" +" "+ "BATCH ID : B1-1M3E "
    return render_template('index.html',mes=message)




@app.route('/anegative/<andis>')
def anegative(andis):
    ancustomer = []
    sql = f"SELECT * FROM ANEGATIVE where district = '{escape(andis)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        ancustomer.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    sql = "SELECT PHONENUMBER FROM ANEGATIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)
    nums = ''

    length = len(ancustomer)

    for i in range(0,length):
        nums = nums + ancustomer[i][1] + ','
    print(nums)

    my_data = {
        # Your default Sender ID
        'sender_id': 'FTWSMS',
```

```
        # Put your message here!
        'message': 'Urgent.....There is a demand for your blood group. We request you to donate your blood
in your nearby BloodBank connect with our Organization.',

        'language': 'english',
        'route': 'p',

        # You can send sms to multiple numbers
        # separated by comma.

         'numbers': nums
        }
    response = requests.request("POST",
                            url,
                            data = my_data,
                            headers = headers)
        #load json data from sourc
    returned_msg = json.loads(response.text)

        # print the send message
    print(returned_msg['message'])



    return render_template('comments.html', ancustomer = ancustomer)




@app.route('/apositive/<apdis>')
def apositive(apdis):
    apcustomer = []
    sql = f"SELECT * FROM APOSITIVE where district = '{escape(apdis)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        apcustomer.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if apcustomer:
        sql = "SELECT * FROM APOSITIVE"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    nums = ''

    length = len(apcustomer)

    for i in range(0,length):
        nums = nums + apcustomer[i][1] + ','
    print(nums)

    my_data = {
        # Your default Sender ID
```

```
        'sender_id': 'FTWSMS',

        # Put your message here!
        'message': 'Argent.....There is a demand for your blood group. We request you to donate your blood
in your nearby BloodBank connect with our Organization.',

        'language': 'english',
        'route': 'p',

        # You can send sms to multiple numbers
        # separated by comma.

         'numbers': nums
        }
    response = requests.request("POST",
                            url,
                            data = my_data,
                            headers = headers)
        #load json data from sourc
    returned_msg = json.loads(response.text)

        # print the send message
    print(returned_msg['message'])

    return render_template('comments.html', apcustomer = apcustomer)

@app.route('/bnegative/<bndis>')
def bnegative(bndis):
    bncustomer = []
    sql = f"SELECT * FROM BNEGATIVE where district = '{escape(bndis)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        bncustomer.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if bncustomer:
        sql = "SELECT * FROM BNEGATIVE"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    nums = ''

    length = len(bncustomer)

    for i in range(0,length):
        nums = nums + bncustomer[i][1] + ','
    print(nums)

    my_data = {
        # Your default Sender ID
        'sender_id': 'FTWSMS',

        # Put your message here!
        'message': 'Argent.....There is a demand for your blood group. We request you to donate your blood
in your nearby BloodBank connect with our Organization.',
```

```
        'language': 'english',
        'route': 'p',

        # You can send sms to multiple numbers
        # separated by comma.

         'numbers': nums
        }
    response = requests.request("POST",
                            url,
                            data = my_data,
                            headers = headers)
        #load json data from sourc
    returned_msg = json.loads(response.text)

        # print the send message
    print(returned_msg['message'])

    return render_template('comments.html', bncustomer = bncustomer)

@app.route('/bpositive/<bpdis>')
def bpositive(bpdis):
    bpcustomer = []
    sql = f"SELECT * FROM BPOSITIVE where district = '{escape(bpdis)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        bpcustomer.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if bpcustomer:
        sql = "SELECT * FROM BPOSITIVE"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    nums = ''

    length = len(bpcustomer)

    for i in range(0,length):
        nums = nums + bpcustomer[i][1] + ','
    print(nums)

    my_data = {
        # Your default Sender ID
        'sender_id': 'FTWSMS',

        # Put your message here!
        'message': 'Argent.....There is a demand for your blood group. We request you to donate your blood
in your nearby BloodBank connect with our Organization.',

        'language': 'english',
        'route': 'p',

        # You can send sms to multiple numbers
```

```
        # separated by comma.

         'numbers': nums
        }
    response = requests.request("POST",
                            url,
                            data = my_data,
                            headers = headers)
        #load json data from sourc
    returned_msg = json.loads(response.text)

        # print the send message
    print(returned_msg['message'])
    return render_template('comments.html', bpcustomer = bpcustomer)


@app.route('/abnegative/<abndis>')
def abnegative(abndis):
    abncustomer = []
    sql = f"SELECT * FROM ABNEGATIVE where district = '{escape(abndis)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        abncustomer.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if abncustomer:
        sql = "SELECT * FROM ABNEGATIVE"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    nums = ''

    length = len(abncustomer)

    for i in range(0,length):
        nums = nums + abncustomer[i][1] + ','
    print(nums)

    my_data = {
        # Your default Sender ID
        'sender_id': 'FTWSMS',

        # Put your message here!
        'message': 'Argent.....There is a demand for your blood group. We request you to donate your blood
in your nearby BloodBank connect with our Organization.',

        'language': 'english',
        'route': 'p',

        # You can send sms to multiple numbers
        # separated by comma.

         'numbers': nums
        }
    response = requests.request("POST",
```

```
                            url,
                            data = my_data,
                            headers = headers)
        #load json data from sourc
    returned_msg = json.loads(response.text)

        # print the send message
    print(returned_msg['message'])

    return render_template('comments.html', abncustomer = abncustomer)

@app.route('/abpositive/<abpdis>')
def abpositive(abpdis):
    abpcustomer = []
    sql = f"SELECT * FROM ABPOSITIVE where district = '{escape(abpdis)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        abpcustomer.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if abpcustomer:
        sql = "SELECT * FROM ABPOSITIVE"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    nums = ''

    length = len(abpcustomer)

    for i in range(0,length):
        nums = nums + abpcustomer[i][1] + ','
    print(nums)

    my_data = {
        # Your default Sender ID
        'sender_id': 'FTWSMS',

        # Put your message here!
        'message': 'Argent.....There is a demand for your blood group. We request you to donate your blood
in your nearby BloodBank connect with our Organization.',

        'language': 'english',
        'route': 'p',

        # You can send sms to multiple numbers
        # separated by comma.

         'numbers': nums
        }
    response = requests.request("POST",
                            url,
                            data = my_data,
                            headers = headers)
        #load json data from sourc
    returned_msg = json.loads(response.text)
```

```
        # print the send message
    print(returned_msg['message'])

    return render_template('comments.html', abpcustomer = abpcustomer)


@app.route('/onegative/<ondis>')
def onegative(ondis):
    oncustomer = []
    sql = f"SELECT * FROM ONEGATIVE where district = '{escape(ondis)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        oncustomer.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if oncustomer:
        sql = "SELECT * FROM ONEGATIVE"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    nums = ''

    length = len(oncustomer)

    for i in range(0,length):
        nums = nums + oncustomer[i][1] + ','
    print(nums)

    my_data = {
        # Your default Sender ID
        'sender_id': 'FTWSMS',

        # Put your message here!
        'message': 'Argent.....There is a demand for your blood group. We request you to donate your blood
in your nearby BloodBank connect with our Organization.',

        'language': 'english',
        'route': 'p',

        # You can send sms to multiple numbers
        # separated by comma.

         'numbers': nums
        }
    response = requests.request("POST",
                            url,
                            data = my_data,
                            headers = headers)
        #load json data from sourc
    returned_msg = json.loads(response.text)

        # print the send message
    print(returned_msg['message'])
```

```
    return render_template('comments.html', oncustomer = oncustomer)

@app.route('/opositive/<opdis>')
def opositive(opdis):
    opcustomer = []
    sql = f"SELECT * FROM OPOSITIVE where district = '{escape(opdis)}'"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        opcustomer.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if opcustomer:
        sql = "SELECT * FROM OPOSITIVE"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    nums = ''

    length = len(opcustomer)

    for i in range(0,length):
        nums = nums + opcustomer[i][1] + ','
    print(nums)

    my_data = {
        # Your default Sender ID
        'sender_id': 'FTWSMS',

        # Put your message here!
        'message': 'Argent.....There is a demand for your blood group. We request you to donate your blood
in your nearby BloodBank connect with our Organization.',

        'language': 'english',
        'route': 'p',

        # You can send sms to multiple numbers
        # separated by comma.

         'numbers': nums
        }
    response = requests.request("POST",
                            url,
                            data = my_data,
                            headers = headers)
        #load json data from sourc
    returned_msg = json.loads(response.text)

        # print the send message
    print(returned_msg['message'])

    return render_template('comments.html', opcustomer = opcustomer)



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

```python
def login():
    return render_template('login.html')


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

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


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


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


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


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


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

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

    customer = []
    sql = f"SELECT * FROM CUSTOMER;"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        customer.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if customer:
        sql = "SELECT * FROM CUSTOMER"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    apcustomer = []
    sql = f"SELECT * FROM APOSITIVE;"
    stmt = ibm_db.exec_immediate(conn, sql)
```

```python
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    apcustomer.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if apcustomer:
    sql = "SELECT * FROM APOSITIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

ancustomer = []
sql = f"SELECT * FROM ANEGATIVE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    ancustomer.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if ancustomer:
    sql = "SELECT * FROM ANEGATIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

bpcustomer = []
sql = f"SELECT * FROM BPOSITIVE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    bpcustomer.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if bpcustomer:
    sql = "SELECT * FROM BPOSITIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

bncustomer = []
sql = f"SELECT * FROM BNEGATIVE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    bncustomer.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if bncustomer:
    sql = "SELECT * FROM BNEGATIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

abpcustomer = []
sql = f"SELECT * FROM ABPOSITIVE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    abpcustomer.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)
```

```
if abpcustomer:
    sql = "SELECT * FROM ABPOSITIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

abncustomer = []
sql = f"SELECT * FROM ABNEGATIVE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    abncustomer.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if abncustomer:
    sql = "SELECT * FROM ABNEGATIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

opcustomer = []
sql = f"SELECT * FROM OPOSITIVE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    opcustomer.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if opcustomer:
    sql = "SELECT * FROM OPOSITIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

oncustomer = []
sql = f"SELECT * FROM ONEGATIVE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    oncustomer.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if oncustomer:
    sql = "SELECT * FROM ONEGATIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

apcount = []
sql = f"SELECT PLACE, count(*) as num FROM APOSITIVE GROUP BY PLACE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    apcount.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if apcount:
    sql = "SELECT * FROM APOSITIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
```

```
    user = ibm_db.fetch_both(stmt)

ancount = []
sql = f"SELECT PLACE, count(*) as num FROM ANEGATIVE GROUP BY PLACE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    ancount.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if ancount:
    sql = "SELECT * FROM ANEGATIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

bpcount = []
sql = f"SELECT PLACE, count(*) as num FROM BPOSITIVE GROUP BY PLACE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    bpcount.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if bpcount:
    sql = "SELECT * FROM BPOSITIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

bncount = []
sql = f"SELECT PLACE, count(*) as num FROM BNEGATIVE GROUP BY PLACE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    bncount.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if bncount:
    sql = "SELECT * FROM BNEGATIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

abpcount = []
sql = f"SELECT PLACE, count(*) as num FROM ABPOSITIVE GROUP BY PLACE;"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
    abpcount.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

if abpcount:
    sql = "SELECT * FROM ABPOSITIVE"
    stmt = ibm_db.exec_immediate(conn, sql)
    user = ibm_db.fetch_both(stmt)

abncount = []
sql = f"SELECT PLACE, count(*) as num FROM ABNEGATIVE GROUP BY PLACE;"
```

```
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        abncount.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if abncount:
        sql = "SELECT * FROM ABNEGATIVE"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    opcount = []
    sql = f"SELECT PLACE, count(*) as num FROM OPOSITIVE GROUP BY PLACE;"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        opcount.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if opcount:
        sql = "SELECT * FROM OPOSITIVE"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    oncount = []
    sql = f"SELECT PLACE, count(*) as num FROM ONEGATIVE GROUP BY PLACE;"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        oncount.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if oncount:
        sql = "SELECT * FROM ONEGATIVE"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)

    selled = []
    sql = f"SELECT * FROM REQPLASMA;"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        selled.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if selled:
        sql = "SELECT * FROM REQPLASMA"
        stmt = ibm_db.exec_immediate(conn, sql)
        user = ibm_db.fetch_both(stmt)


    return render_template('admin.html', customer = customer, apcustomer = apcustomer, ancustomer =
ancustomer,bpcustomer = bpcustomer, bncustomer = bncustomer, abncustomer = abncustomer, abpcustomer =
abpcustomer, opcustomer = opcustomer, oncustomer = oncustomer, apcount = apcount, ancount = ancount,
bpcount = bpcount, bncount = bncount, abpcount = abpcount, abncount = abncount, opcount = opcount, oncount
= oncount, selled = selled )
```

```
@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        uname = request.form['uname']
        mail = request.form['email']
        phone = request.form['phone']
        password = request.form['password']
        bloodgrp = request.form['bloodgroup']

        sql = "SELECT * FROM customer WHERE name=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt,1,uname)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)

    if account:
        return render_template('index.html', msg="You are already a member, please login using your
details....")

    else:
      insert_sql = "INSERT INTO customer VALUES (?,?,?,?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, uname)
      ibm_db.bind_param(prep_stmt, 2, mail)
      ibm_db.bind_param(prep_stmt, 3, phone)
      ibm_db.bind_param(prep_stmt, 4, password)
      ibm_db.bind_param(prep_stmt, 5, bloodgrp)
      ibm_db.execute(prep_stmt)

    return render_template('login.html', msg="Student Data saved successfuly..")


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

        mail = request.form['email']
        password = request.form['password']
        print(mail, password)



        if mail == 'abcd@gmail.com' and password == 'kil':
            return redirect(url_for('admin'))

        else:
            sql = f"select * from customer where email='{escape(mail)}' and password=
'{escape(password)}'"
            stmt = ibm_db.exec_immediate(conn, sql)
            data = ibm_db.fetch_both(stmt)

        if data:
```

```
            session["mail"] = escape(mail)
            session["password"] = escape(password)
            return redirect(url_for('dashboard'))

        else:
            flash("Mismatch in credetials", "danger")




@app.route('/needplasma', methods=['GET', 'POST'])
def needplasma():
    if request.method == 'POST':
        uname = request.form['uname']
        phone = request.form['phone']
        bloodgrp = request.form['bloodgroup']
        place = request.form['place']
        district = request.form['district']


        if bloodgrp == 'A-VE':

            insert_sql = "INSERT INTO reqplasma VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)

            return redirect(url_for("anegative", andis = district))


        elif bloodgrp == 'A+VE':
            insert_sql = "INSERT INTO reqplasma VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)
            return redirect(url_for("apositive", apdis = district))

        elif bloodgrp == 'B+VE':
            insert_sql = "INSERT INTO reqplasma VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)
            return redirect(url_for("bpositive", bpdis = district))
```

```
elif bloodgrp == 'B-VE':
    insert_sql = "INSERT INTO reqplasma VALUES (?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, uname)
    ibm_db.bind_param(prep_stmt, 2, phone)
    ibm_db.bind_param(prep_stmt, 3, bloodgrp)
    ibm_db.bind_param(prep_stmt, 4, place)
    ibm_db.bind_param(prep_stmt, 5, district)
    ibm_db.execute(prep_stmt)
    return redirect(url_for("bnegative", bndis = district))

elif bloodgrp == 'AB-VE':
    insert_sql = "INSERT INTO reqplasma VALUES (?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, uname)
    ibm_db.bind_param(prep_stmt, 2, phone)
    ibm_db.bind_param(prep_stmt, 3, bloodgrp)
    ibm_db.bind_param(prep_stmt, 4, place)
    ibm_db.bind_param(prep_stmt, 5, district)
    ibm_db.execute(prep_stmt)
    return redirect(url_for("abnegative", abndis = district))

elif bloodgrp == 'AB+VE':
    insert_sql = "INSERT INTO reqplasma VALUES (?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, uname)
    ibm_db.bind_param(prep_stmt, 2, phone)
    ibm_db.bind_param(prep_stmt, 3, bloodgrp)
    ibm_db.bind_param(prep_stmt, 4, place)
    ibm_db.bind_param(prep_stmt, 5, district)
    ibm_db.execute(prep_stmt)
    return redirect(url_for("abpositive", abpdis = district))

elif bloodgrp == 'O-VE':
    insert_sql = "INSERT INTO reqplasma VALUES (?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, uname)
    ibm_db.bind_param(prep_stmt, 2, phone)
    ibm_db.bind_param(prep_stmt, 3, bloodgrp)
    ibm_db.bind_param(prep_stmt, 4, place)
    ibm_db.bind_param(prep_stmt, 5, district)
    ibm_db.execute(prep_stmt)
    return redirect(url_for("onegative", ondis = district))

elif bloodgrp == 'O+VE':
    insert_sql = "INSERT INTO reqplasma VALUES (?,?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, uname)
    ibm_db.bind_param(prep_stmt, 2, phone)
    ibm_db.bind_param(prep_stmt, 3, bloodgrp)
    ibm_db.bind_param(prep_stmt, 4, place)
    ibm_db.bind_param(prep_stmt, 5, district)
    ibm_db.execute(prep_stmt)
    return redirect(url_for("opositive", opdis = district))
```

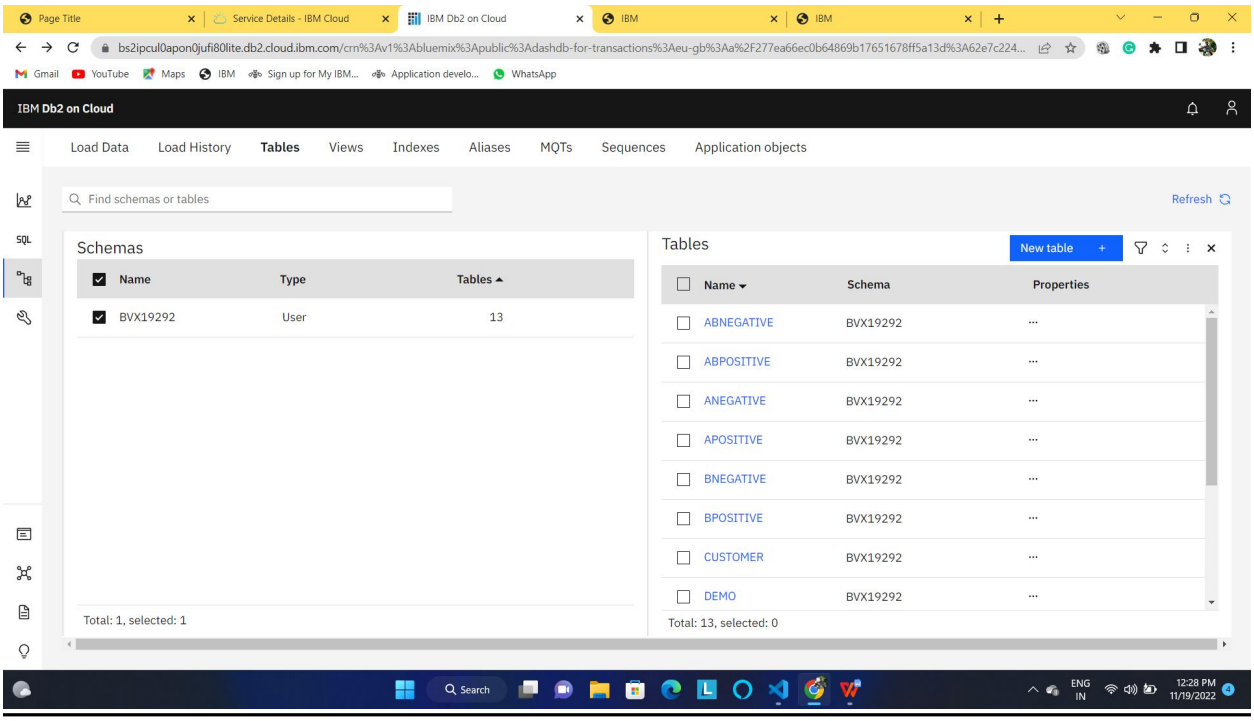
```
        else:
            return "Please INSERT a valid Blood Group and Enter the Blood group in CAPITAL LETTERS like
(A+VE, B-VE, AB+VE)..."

    # return render_template('comments.html', msg="Student Data saved successfuly..")


@app.route('/donateplasma', methods=['GET', 'POST'])
def donateplasma():
    if request.method == 'POST':
        uname = request.form['uname']
        phone = request.form['phone']
        bloodgrp = request.form['bloodgroup']
        place = request.form['place']
        district = request.form['district']

        # sql = "SELECT * FROM donateplasma WHERE name=?"
        # stmt = ibm_db.prepare(conn, sql)
        # ibm_db.bind_param(stmt,1,uname)
        # ibm_db.execute(stmt)
        # account = ibm_db.fetch_assoc(stmt)

    # if account:
    #     return render_template('index.html', msg="You are already a member, please login using your
details....")

    # else:
        if bloodgrp == 'A+VE':
            insert_sql = "INSERT INTO APOSITIVE VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)

        elif (bloodgrp == 'A-VE'):
            insert_sql = "INSERT INTO ANEGATIVE VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)

        elif (bloodgrp == 'B+VE'):
            insert_sql = "INSERT INTO BPOSITIVE VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)
```

```
        elif (bloodgrp == 'B-VE'):
            insert_sql = "INSERT INTO BNEGATIVE VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)

        elif (bloodgrp == 'AB+VE'):
            insert_sql = "INSERT INTO ABPOSITIVE VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)

        elif (bloodgrp == 'AB-VE'):
            insert_sql = "INSERT INTO ABNEGATIVE VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)

        elif (bloodgrp == 'O+VE'):
            insert_sql = "INSERT INTO OPOSITIVE VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)


        elif (bloodgrp == 'O-VE'):
            insert_sql = "INSERT INTO ONEGATIVE VALUES (?,?,?,?,?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, uname)
            ibm_db.bind_param(prep_stmt, 2, phone)
            ibm_db.bind_param(prep_stmt, 3, bloodgrp)
            ibm_db.bind_param(prep_stmt, 4, place)
            ibm_db.bind_param(prep_stmt, 5, district)
            ibm_db.execute(prep_stmt)

        else:
            return "Please INSERT a valid Blood Group and Enter the Blood group in CAPITAL LETTERS like
(A+VE, B-VE, AB+VE)..."
```

```
    return render_template('thanks.html', msg="Student Data saved successfuly..")
    # return redirect(url_for("place",plc = place))
#------------------------------------- Count -----------------------------------
if __name__ == "__main__":
    app.run(debug=True)
```

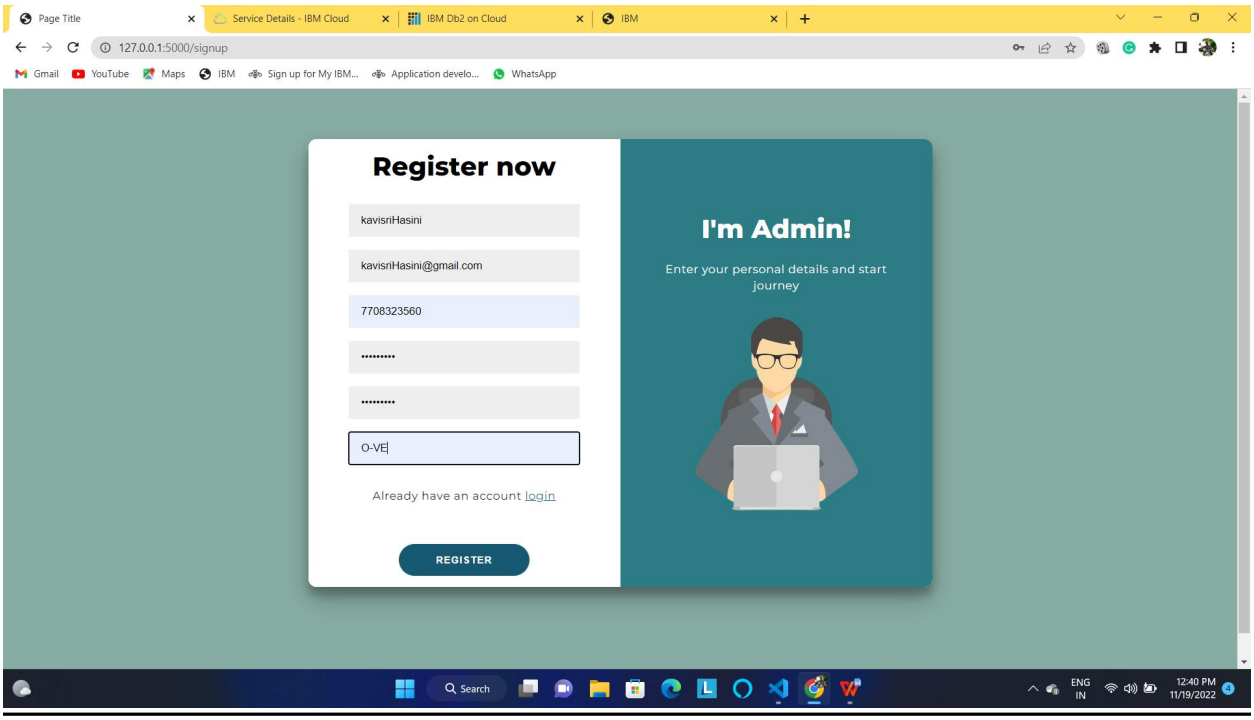
Feature 5 :

Database Schemas :

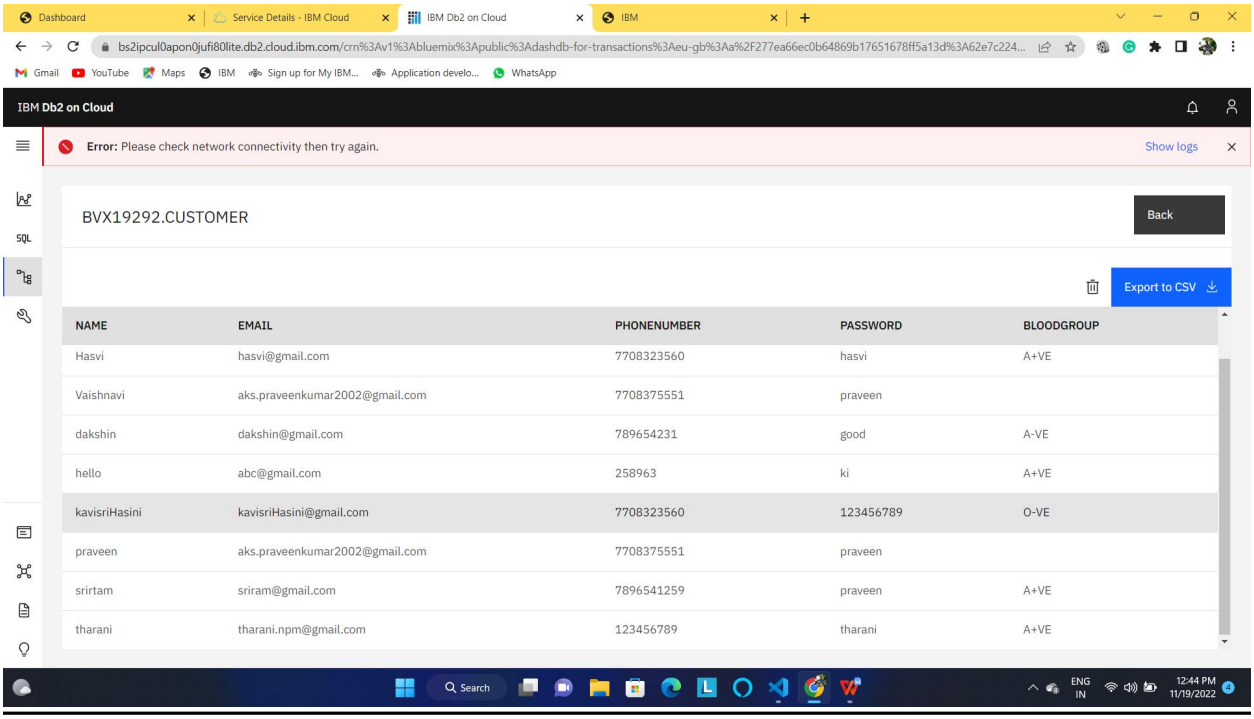


8 .Testing :

Registration Form:

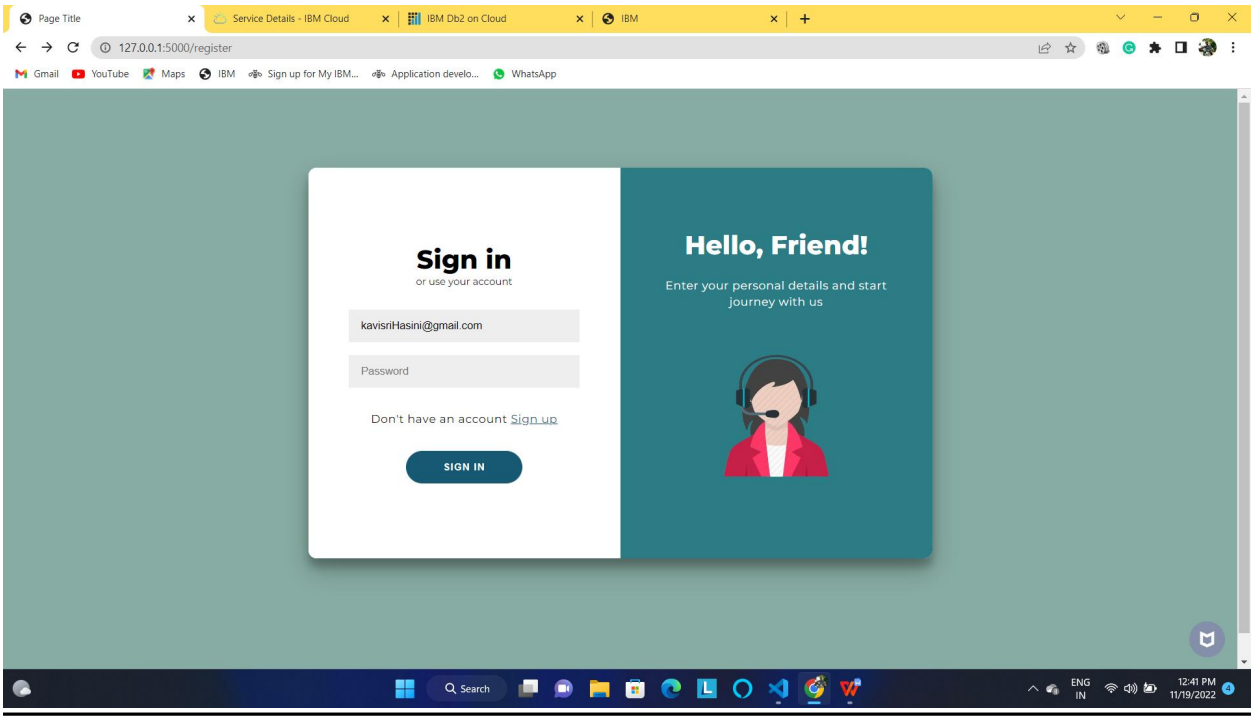


Data Stored in DataBase :

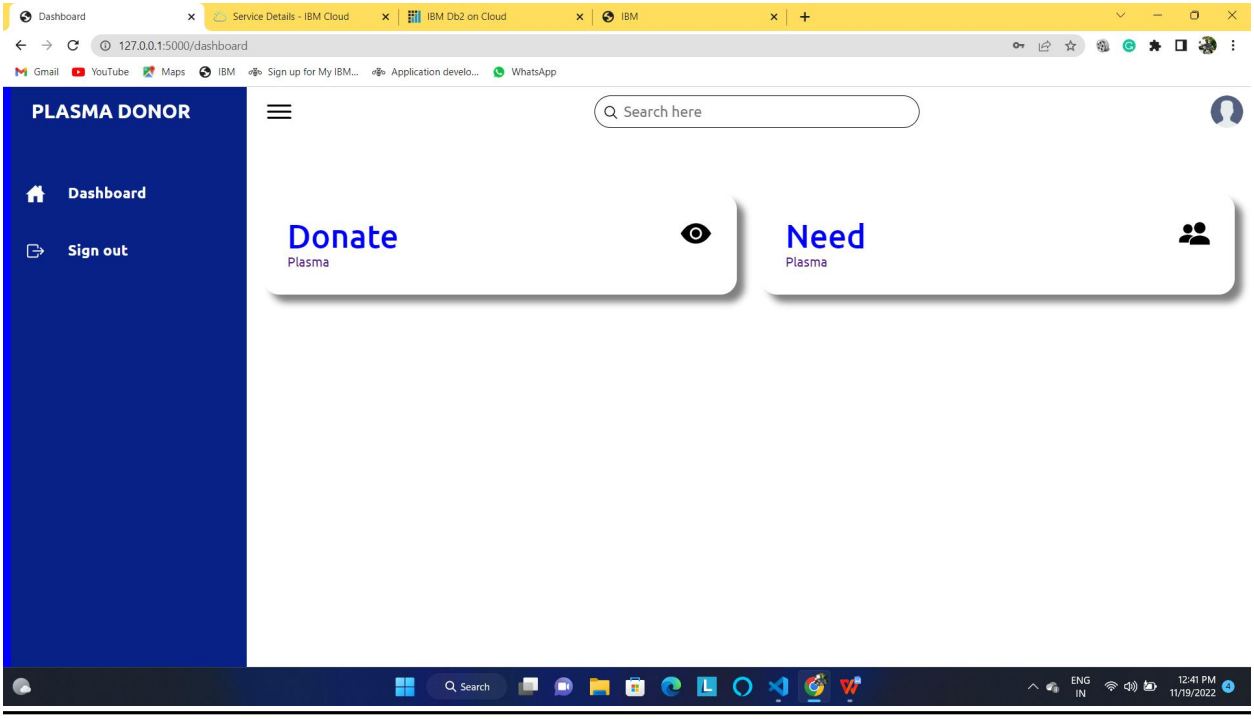


NAME	EMAIL	PHONENUMBER	PASSWORD	BLOODGROUP
Hasvi	hasvi@gmail.com	7708323560	hasvi	A+VE
Vaishnavi	aks.praveenkumar2002@gmail.com	7708375551	praveen	
dakshin	dakshin@gmail.com	789654231	good	A-VE
hello	abc@gmail.com	258963	ki	A+VE
kavisriHasini	kavisriHasini@gmail.com	7708323560	123456789	O-VE
praveen	aks.praveenkumar2002@gmail.com	7708375551	praveen	
srirtam	sriram@gmail.com	7896541259	praveen	A+VE
tharani	tharani.npm@gmail.com	123456789	tharani	A+VE

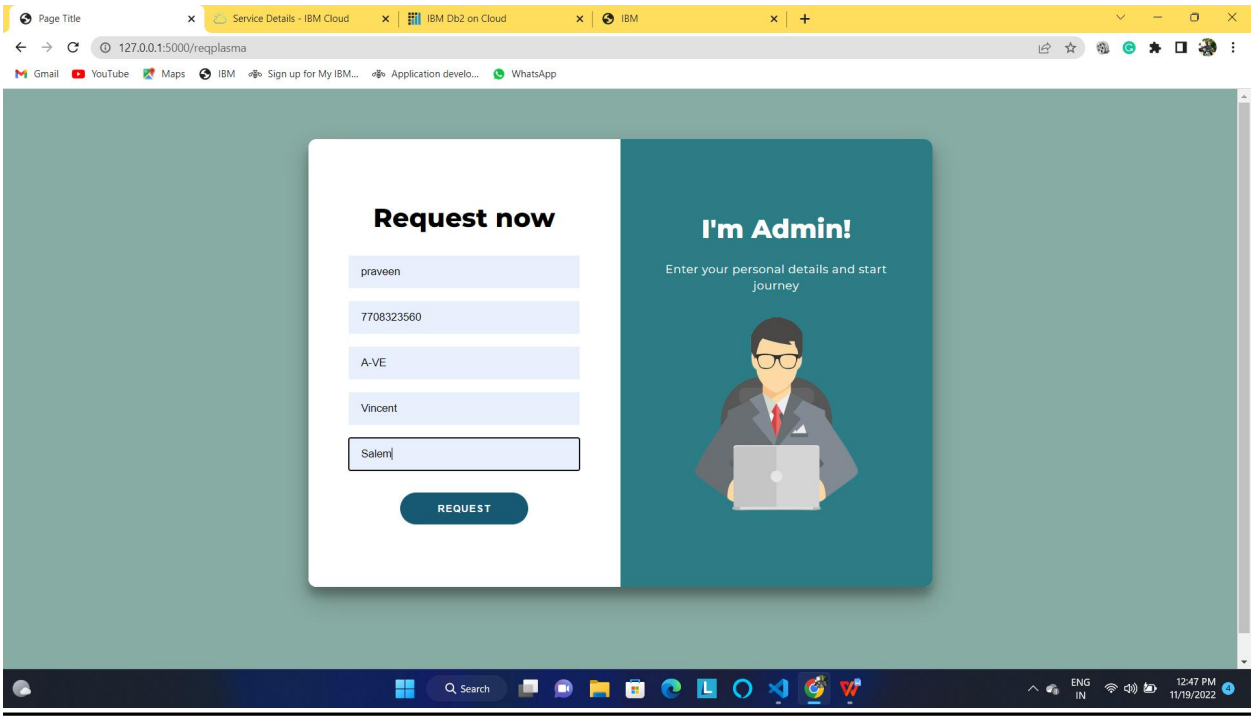
Login Form :



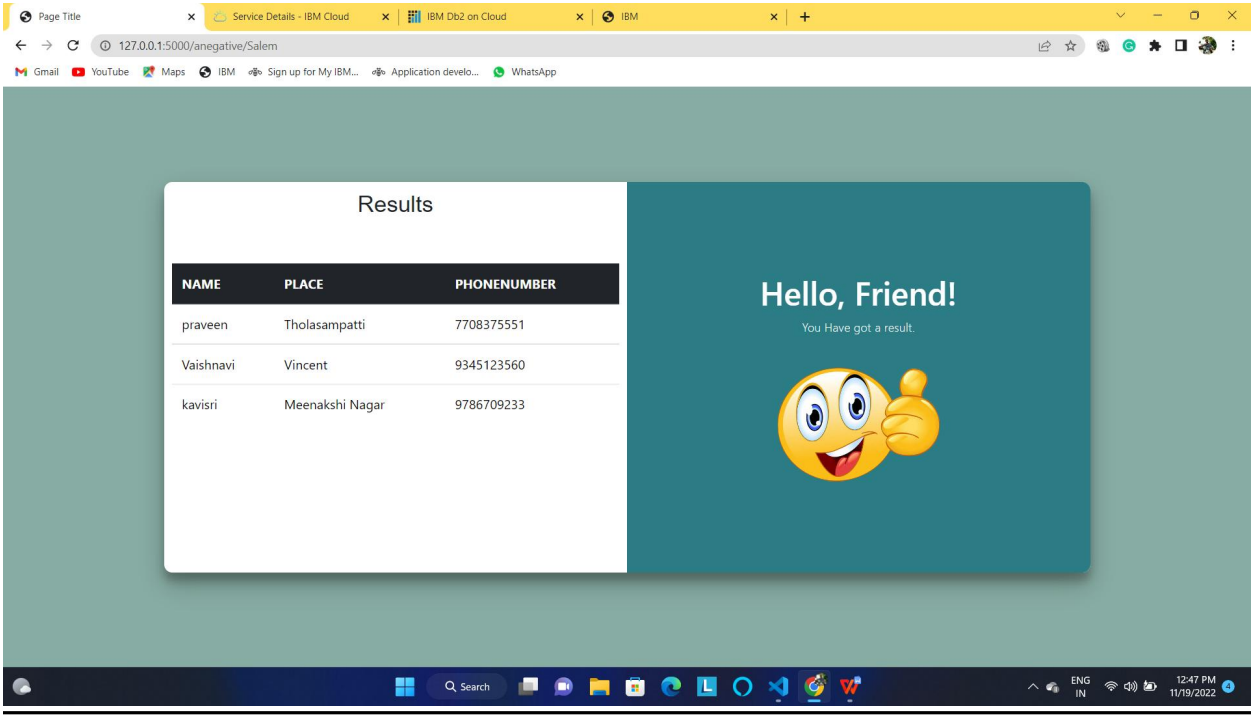
Authentication Done and Opened Dashboard:



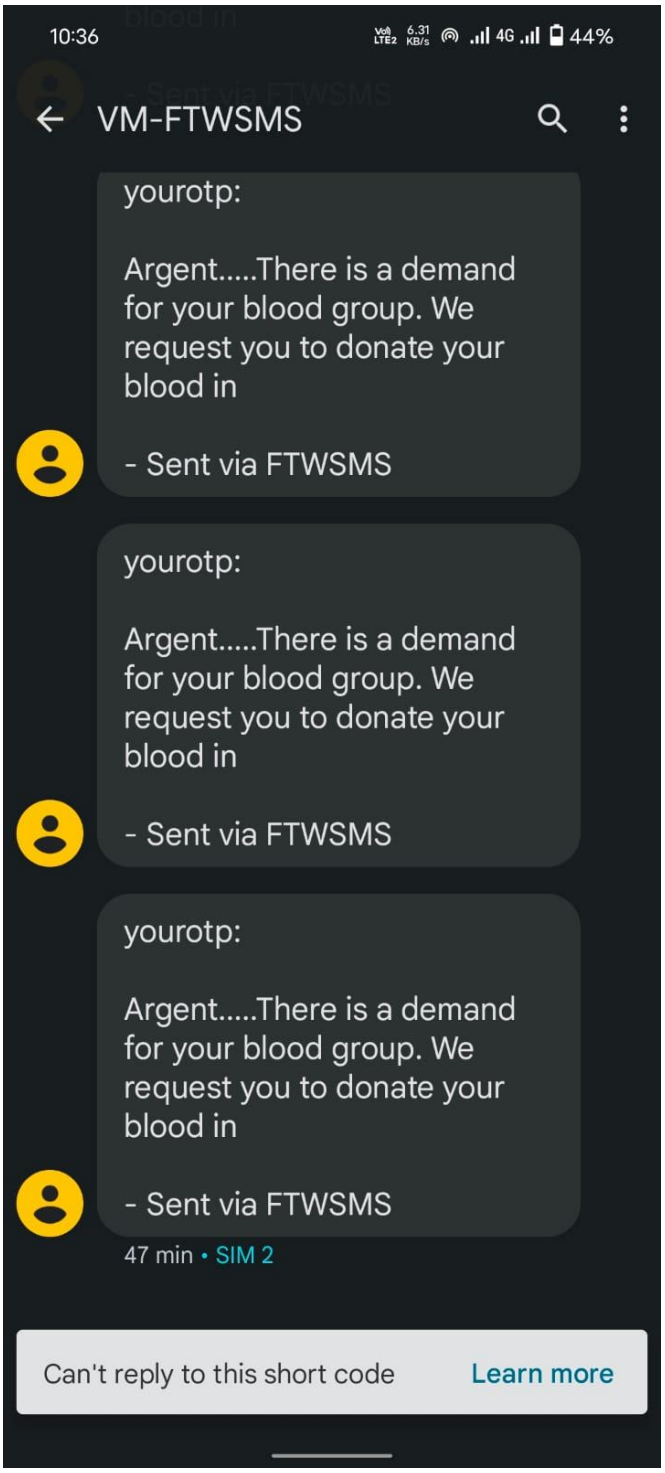
Request for Blood :



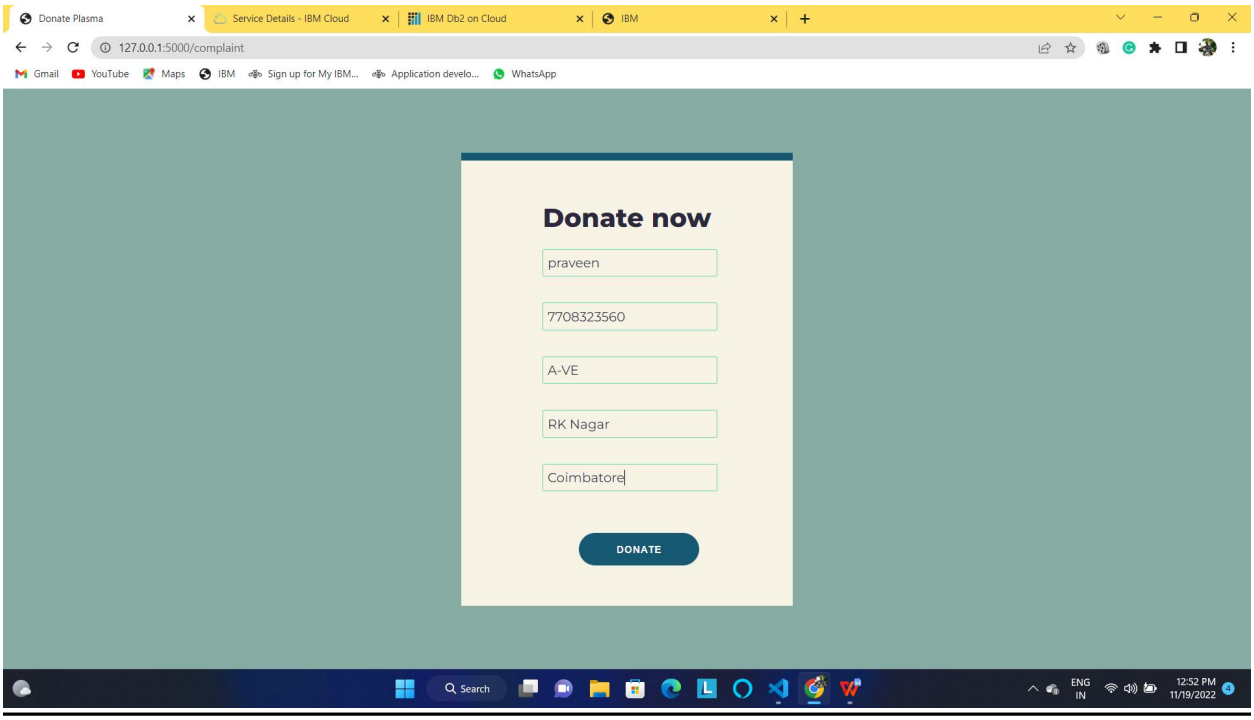
Results for the Request :

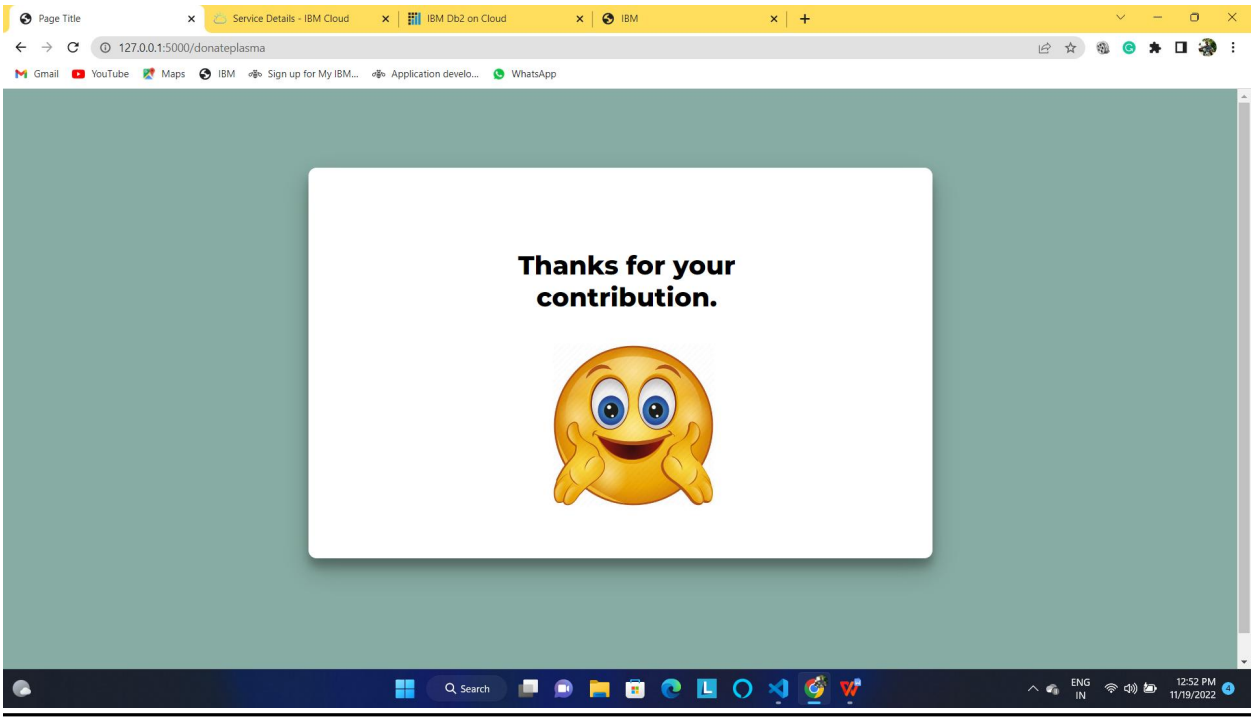


Sent SMS to the phone numbers in the database of the Donors :

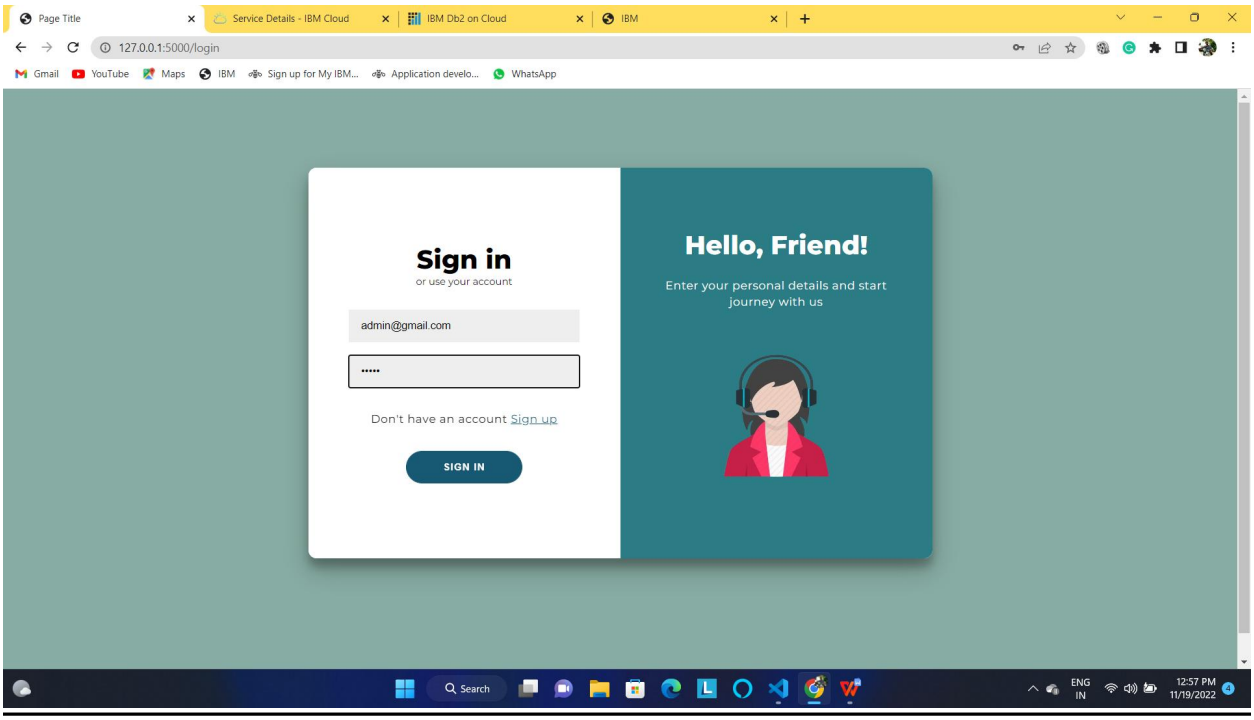


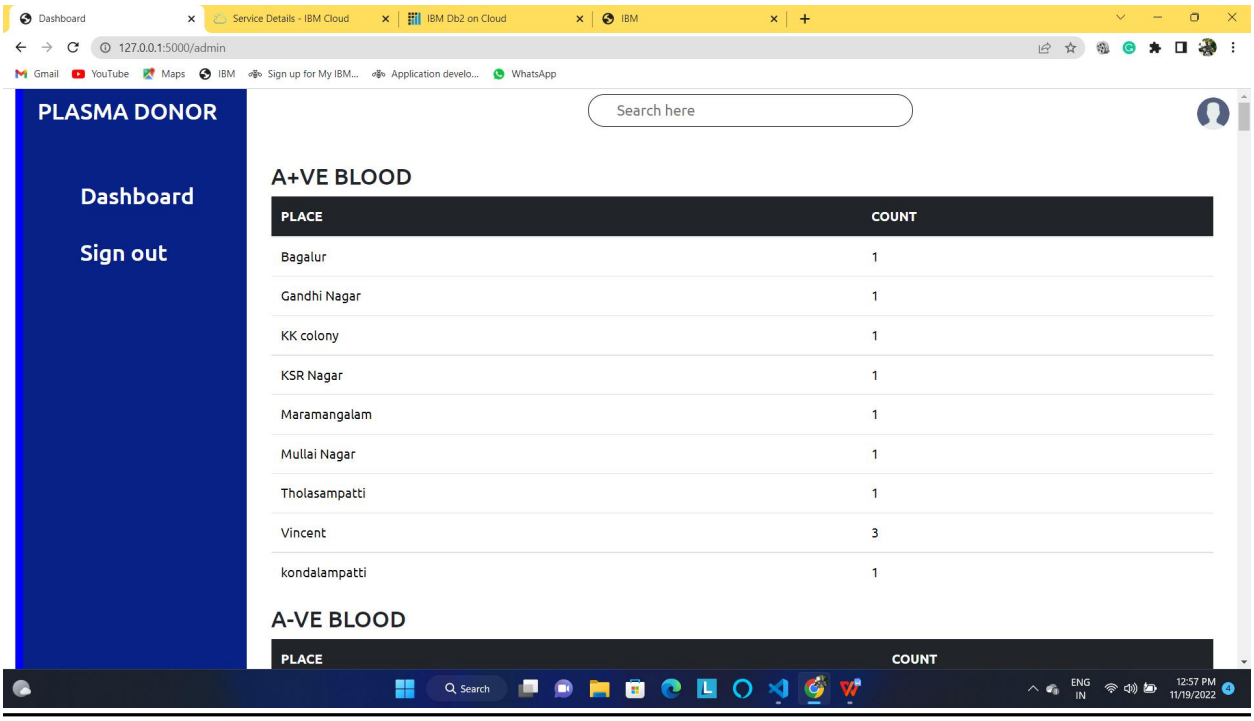
DONATION FORM :





Admin login :


Page Title
Service Details - IBM Cloud
IBM Db2 on Cloud
IBM
127.0.0.1:5000/login
Gmail
YouTube
Maps
IBM
Sign up for My IBM...
Application develo...
WhatsApp
Sign in
or use your account
admin@gmail.com
Don't have an account Sign up
SIGN IN
Hello, Friend!
Enter your personal details and start journey with us
Search
ENG IN
12:57 PM
11/19/2022


Dashboard
Service Details - IBM Cloud
IBM Db2 on Cloud
IBM
127.0.0.1:5000/admin
Gmail
YouTube
Maps
IBM
Sign up for My IBM...
Application develo...
WhatsApp
PLASMA DONOR
Dashboard
Sign out
Search here
A+VE BLOOD
PLACE
COUNT
Bagalur 1
Gandhi Nagar 1
KK colony 1
KSR Nagar 1
Maramangalam 1
Mullai Nagar 1
Tholasampatti 1
Vincent 3
kondalampatti 1
A-VE BLOOD
PLACE
COUNT
Search
ENG IN
12:57 PM
11/19/2022

14. RESULTS

The application is designed in such a way that, It helps to overcome the demand for plasma. The Users can get the plasma easily.

15. ADVANTAGES

The main advantage of a blood bank management system is easy and effective information retrieval. Hence, the staff can view precise information quickly. The staff can now store all the details in the blood bank management system. Therefore, they can get rid of the manual procedures.

16. CONCLUSION

The efficient way of finding plasma donor 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 website operations. I have hosted the website in IBM Cloud platform to make sure the operations are running successfully IBM Cloud lambda function is used and to deploy the application Docker service is used.

17. APPENDIX

Source Code -

GitHub Link - https://github.com/IBM-EPBL/IBM-Project-36740-1660297430