PLASMA DONOR APPLICATION

A PROJECT REPORT

Submitted by

S.DEVA (814619104005)

R.DINESHKUMAR (814619104007)

S.K.SHEDHUBATHI (814619104017)

R.VASIM AKRAM (814619104021)

In partial fulfillment for the award of the degree

of

BACHELOROFENGINEERING

IN

COMPUTERSCIENCEANDENGINEERING

TRICHYENGINEERINGCOLLEGE,TRICHY

ANNAUNIVERSITY:CHENNAI 600025

NOVEMBER 2022

ABSTRACT

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

The main goal of our project is to design a user-friendly web application that is like a scientific vehicle from which we can help reduce mortality or help those affected by COVID19 by donating plasma from patients.The aim is to build a Lifesaver EBlood Donation App using Cloud with advanced features that will help to overcome the barrier between blood bank, blood donor and patient. To build an android application that will help people to get blood in emergency situations like natural disasters using features like geo-tagging, SMS Gateway and payment gateway.

At this present time plasma banks are in short supply. Not only that, but the number of plasma donors is low too.And some people do not know what plasma donation is and where to donate plasma.

The main goal of our project is to make it easier for the COVID-19 patients to get a plasma donor easily as well as donate plasma if they have recovered. The system targets two types of users: the people who want to donate plasma and the people who need plasma.

TABLEOFCONTENTS

CHAPTER 1

INTRODUCTION

Blood donation is a vital part of worldwide healthcare. It relates to blood transfusion as a life-sustaining and life-saving procedure as well as a form of therapeutic phlebotomy as a primary medical intervention. Over one hundred million units of blood are donated each year throughout the world.This article will concisely discuss a short history of blood donation origin and purpose, blood testing, donor eligibility and selection, adverse effects of donation, blood donation as a primary medical intervention, and a brief discussion of pathogen reduction and inactivation for donated blood.

It allows for blood transfusion as a life-sustaining and life-saving procedure. Over one hundred million units of blood are donated each year throughout the world. This activity reviews donor eligibility and selection, adverse effects of donation, and pathogen reduction and inactivation for donated blood. This activity highlights the role of the interprofessional team in ensuring appropriate protocol is followed.

1.1 PROJECT OVERVIEW

The PLASMA DONATION APPLICATION SYSTEM is great project. this project is designed for successful completion of project on Online Blood Donation management System.the basic building aim is to provide blood donation service to the city recently.x

Plasma Donation Application System is a browser based system that is designed store, process, retrieve and analyze information concerned with the administrative and inve ntory management within a blood bank. This project aims at maintaining all the information pertaining to blood donors, different blood groups available in each blood bank and help them manage in a better way.

Aim is to provide transparency in this field, make the process of obtaining blood from a blood bank hassle free and corruption free and make the system of blood bank management effective.

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**.

CHAPTER 2

LITERATURE SURVEY

TITLE	AUTHORS AND YEAR	MERITS
Developing a plasma donor application using Function-as-aservice in AWS	Aishwarya R Gowri , Jain University, Department of MCA, computer science2021	In this paper, the author has carried out analysis based on the opportunities presented by serverless computing. Authors work presented a handson experience of serverless technologies using different services from different cloud provides such as Amazon, Google, IBM, Microsoft Azure
Instant Plasma Donor Recipient Connector Web Application	Kalpana Devi Guntoju, Tejaswini Jalli, Sreeja Uppala, Sanjay Mallisetti2021	The main goal of our project is to design a user-friendly web application that is like a scientific vehicle from which we can help reduce mortality or help those affected by COVID19 by donating plasma from patients..
Lifesaver E-Blood Donation App Using Cloud	Rishab Chakrabarti, Asha Darade, Neha Jadhav, Prof. S. M. Chitalkar-2020	The aim is to build a Lifesaver EBlood Donation App using Cloud with advanced features that will help to overcome the barrier between blood bank, blood donor and patient. To build an android application that will help people to get blood in emergency situations like natural disasters using features like geo-tagging, SMS Gateway and payment gateway

Enhanced Mobile Application Development For Plasma, Mother's Milk And Blood Banks.	Dr. S. Brindha, Ms. D. Priya, Mr. S. Ajith Kannan, Mr. D. Joyal Victor, Mr. R. Gunachandran-2021	At this present time plasma banks are in short supply. Not only that, but the number of plasma donors is low too.And some people do not know what plasma donation is and where to donate plasma. Set up a system to alleviate this situation and help needy people to identify plasma donors and plasma banks
Plasma Donation Website using MERN stack	Neha Soni-2021	The main goal of our project is to make it easier for the COVID-19 patients to get a plasma donor easily as well as donate plasma if they have recovered. The system targets two types of users: the people who want to donate plasma and the people who need plasma.

2.1 EXISTING PROBLEM

The method helps the users to check the availability of donors. A donor has to register to the website providing their details. The registered users can get the information about the donor count of each blood group. The database will have all the details such as name, email, phone number, infected status. Whenever a user requests for a particular blood group then the concerned blood group donors will receive the notification regarding the requirement.

People have to find them physically by visiting hospitals register book and reaching out recovered donors' home and sometimes they will be not available at their places and will be went on work. In this type of scenarios, diseased persons health gets more worsened. This is an expensive and will not work as effectively at emergency situations.

As per blood requirement, user can quickly get notification from blood bank within the radius of 5-10km. If requested blood group is available in the blood bank then it will send positive reply message to the users. If requested stock is not available in the blood bank then blood bank send notification to all donors. If anyone is able to donate then he will reply to blood bank Blood bank App provides list of blood banks in their space. solely a registered person, with disposition to gift blood, are going to be able to access the service. In this application they are using the GPS technology to trace the way to the blood bank.

2.2 REFERENCES

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

[2] P. C. P. C. a. V. I. M. Yan, “Building a chatbot with serverless computing,” IBM watson research center, 2016.

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

[4] Z. Al-Ali, ““Making Serverless Computing More Serverless,”,” IEEE 11th International Conference on Cloud Computing (CLOUD), pp. pp. 456-459, 2018., 2018.

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

2.3 PROBLEM STATEMENT DEFINITION

Customer Problem Statement Template:

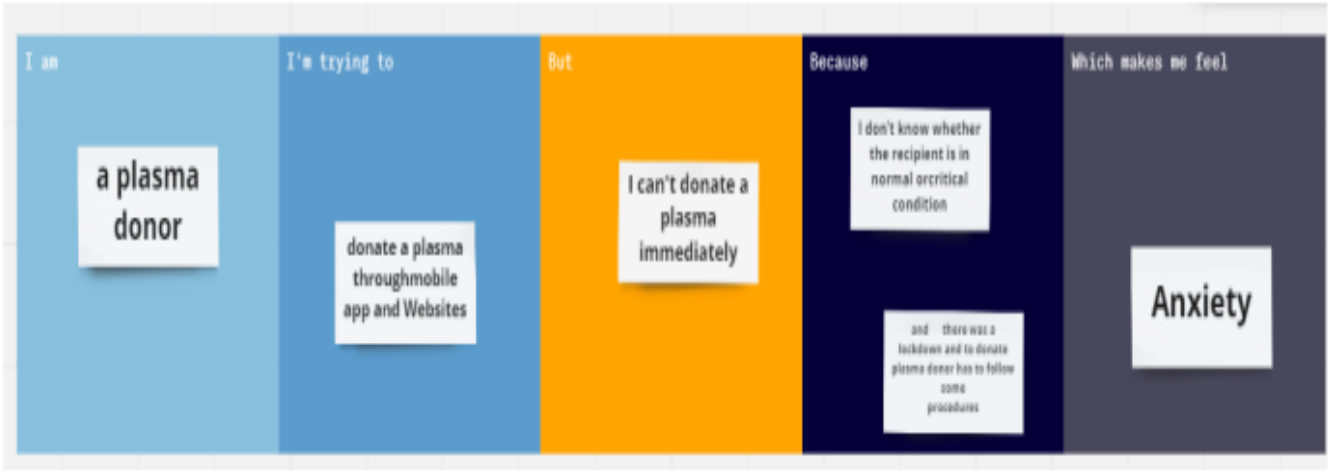
Create a problem statement to understand your customer's point of view. The Customer Problem Statement template helps you focus on what matters to create experiences people will love. A well-articulated customer problem statement allows you and your team to find the ideal solution for the challenges your customers face.

Throughout the process, you’ll also be able to empathize with your customers, which helps you better understand how they perceive your product or service.

I am	Describe customer with 3-4 key characteristics - *who are they?*	Describe the customer and their attributes here
I'm trying to	*List their outcome or "job" the care about - what are they trying to achieve?*	List the thing they are trying to achieve here
but	Describe what problems or barriers stand in the way – *what bothers them most?*	Describe the problems or barriers that get in the way here
because	Enter the "root cause" of why the problem or barrier exists – *what needs to be solved?*	Describe the reason the problems or barriers exist
which makes me feel	Describe the emotions from the customer's point of view – *how does it impact them emotionally?*	Describe the emotions the result from experiencing the problems or barriers

Problem Statement 1:

Problem Statement 2:



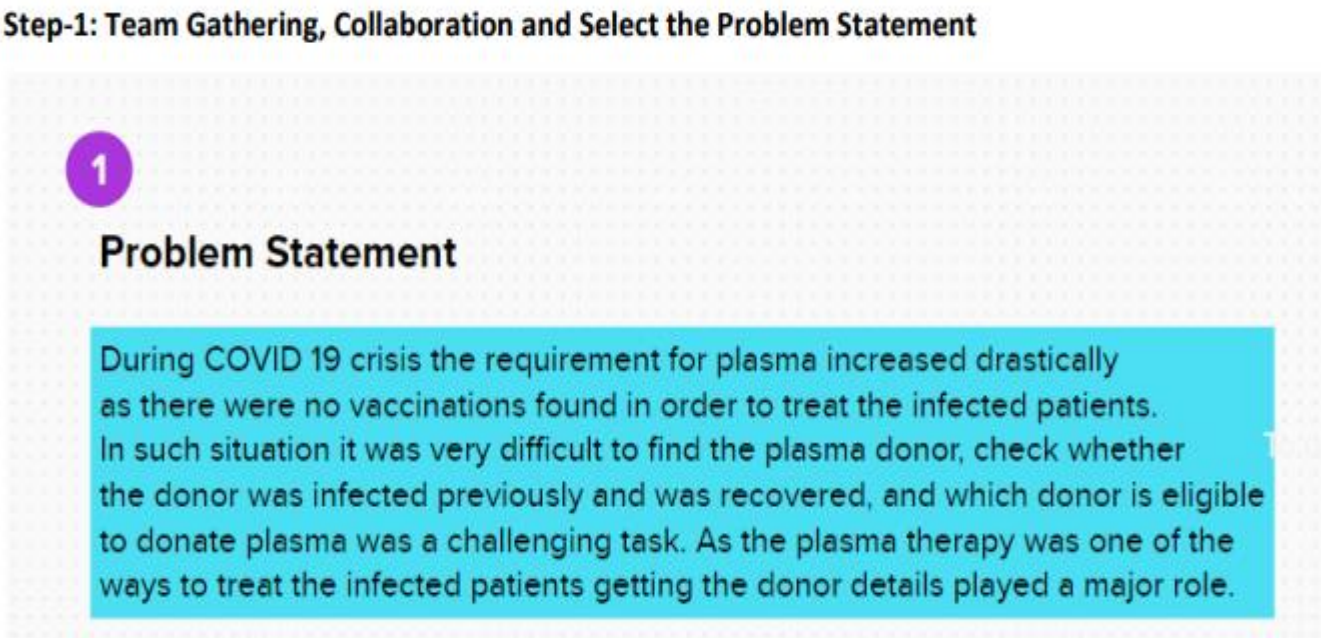
Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	a plasma recipient	get a plasmafrom donor through online and bloodbanks	I am unable to find donor	the covid 19reduce the supply and increase the need of plasma.	worry and depressed about their health and dissapointment
PS-2	a plasma donor	donate a plasma through mobile app and Websites.	I can't donate a plasma immediately	I don't know whether the recipient is in normal or critical condition andthere was a lockdown andto donate plasma donor has to follow some procedures	Anxiety

CHAPTER 3

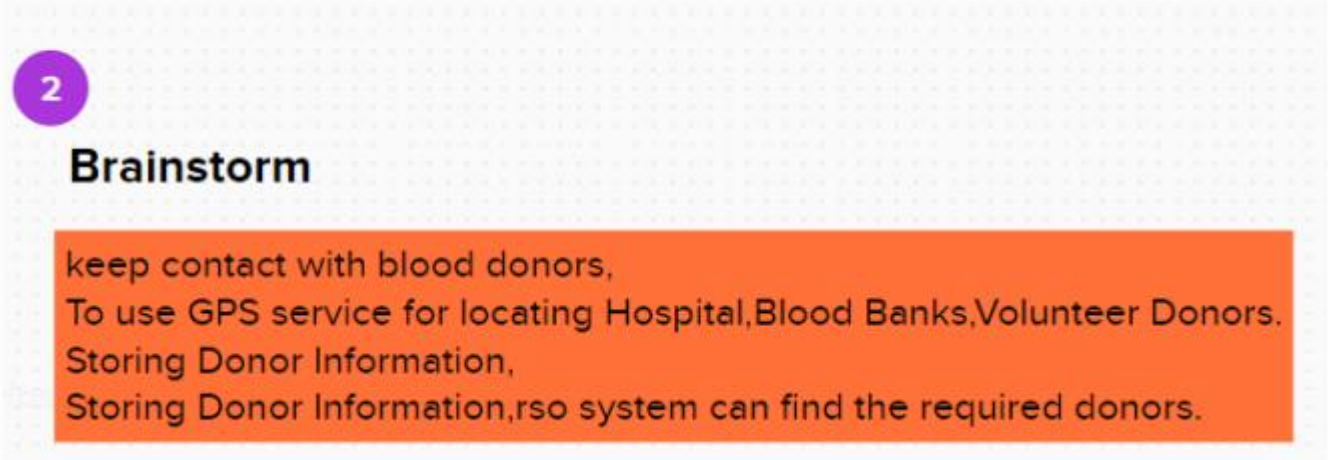
IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

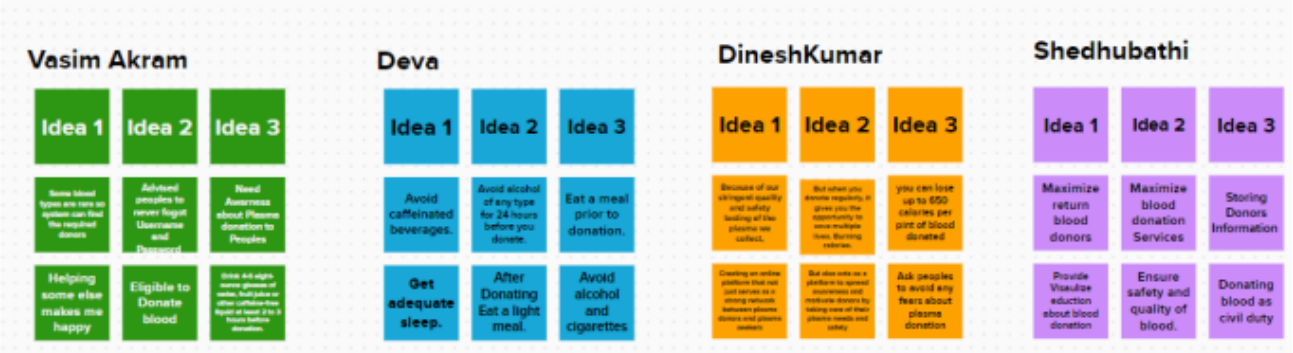
3.2 IDEATION & BRAINSTORMING



Step-2: Brainstorm, Idea Listing and Grouping

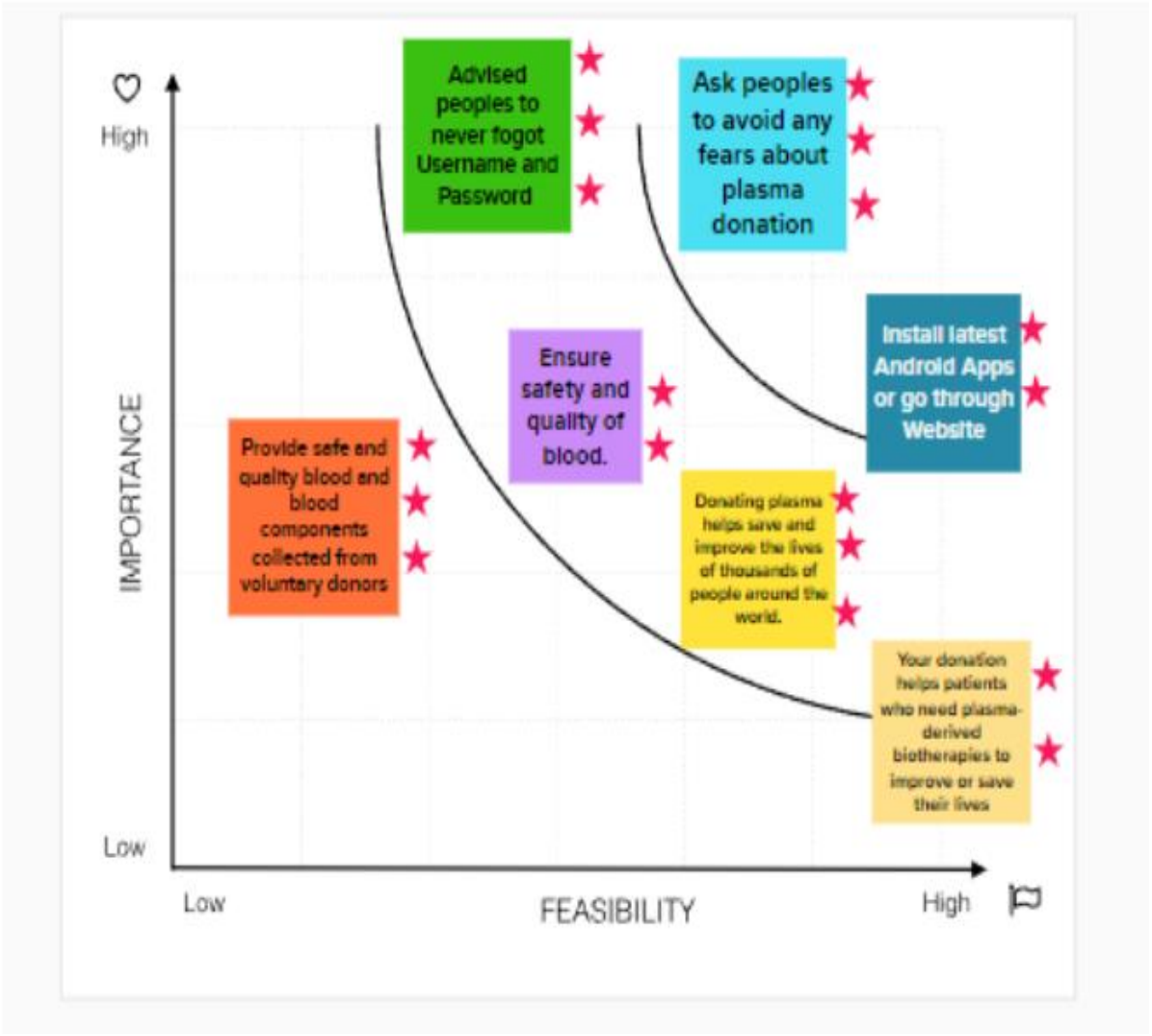


Steps 3:Idea Listing and Grouping




3
Group Ideas
Mobile Apps and Streaming Services.
Podcasts and Social media
App Creation and Developing
User generated Content and Testing
Press Relase video
Retargetting and live chat.
Blogging Email
Google adwords display ads

Step-4: Idea Prioritization



3.3 PROPOSED SOLUTION

Proposed Solution Template:

Project team shall fill the following information in proposed solution template.

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• During COVID 19 crisis the requirement for plasma increased drastically as there were no vaccinations found in order to treat the infected patients. In such situation it was very difficult to find the plasma donor, check whether the donor was infected previously and was recovered, and which donor is eligible to donate plasma was a challenging task. As the plasma therapy was one of the ways to treat the infected patients getting the donor details played a major role.
2.	Idea / Solution description	• Get the information about the plamsa donor • Reduced the workload by storing the details in cloud storage • By using GPS easily track the plasma donor • Websites to find check the availability of Donors
3.	Novelty / Uniqueness	• Upgrading the UI that is more user-friendly which will help many users to access the website and also ensures that many plasma donors can be added into the community. • Using elastic load balancer, it helps to handle multiple requests at the same time which will maintain the uptime of the website with negligible downtime.
4.	Social Impact / Customer Satisfaction	• It improves the efficiency and easy to check the availability of the plasma donor.

		• It will provide the better way to track the plasma donor and get the notification through the mobile to the plasma donor.
5.	Business Model (Revenue Model)	• To build an android application that will help people to get blood in emergency situations like natural disasters using features like geo - tagging.To motivate people for blood donation and to help patients receive blood in emergency situations, we have designed an application to overcome all the problems which the current offline as well as online systems face. • If in emergency a patient requires blood, using this application we'll not just be able to contact Blood Bank and Hospitals but can also seek help from individual registered Donors
6.	Scalability of the Solution	• Instead of scouring the entire world for plasma donors, this APP enables users to find donors while sitting at home.When there is an emergency, plasma requests that everyone sends a message. When a donor is prepared to donate, the recipient is informed. Receiver may get in touch with the donor. This software helps donors find potential donors quickly and easily by letting them know if they are eligible to donate. • Increasing the scalability of plasma donars easily through web application

3.4 PROBLEM SOLUTION FIT

CUSTOMER SEGMENT:	CUSTOMER CONSTRAINT:	AVAILABLE SOLUTION:
Searching and Check the Availability of plasma donor.	Install Latest Android Apps or Go through Website and storing the information in a cloud	When they face the problem in searching or contacting the plasma donor. It is very helpful for searching the plasma donor. And Age Registration to donating Plasma.

JOBS TO BE DONE/PROBLEM:	PROBLEM ROOT CAUSE:	BEHAVIOUR:
Provide safe and quality blood and blood components collected from voluntary donors.	Upgrading the UI that is more user-friendly which will help many users to access the website and also ensures that many plasma donors can be added into the community	This App or website is used for anyone is easily find and searching the available plasma donors.

TRIGGERS:	YOUR SOLUTION:	CHANNELS OF BEHAVIOUR:
By using this app with the efficient and reliable UI that is more user-friendly which will help many users to access the website understand by all the people efficiently	The application/website is used give the better result in searching the plasma donor and contact them when the patient in the emergency situation .Donors details have been stored in the cloud .Reduce the searching time of the plasma required person. Contact the plasma donors via mobile.	Users to Visit nearby camp or hospital and donate as well as receive plasma.
EMOTION BEFORE/AFTER: Plasma Donors have some fears. So Need Awarness about the plasma donation for donors. Feels good to know all peoples helping and donate plasma.		

CHAPTER 4

REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT

Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration via Website Registration Through G-mail Registration via Application
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP Confirmation via Text message
FR-3	User profile	User Name User Age User Gender User Weight User Blood Group User Contact Number User Address
FR-4	User Login	Login through registered email id Login through registered Website id
FR-5	Required Data	User Age User Gender User Health Conditions User Blood Group Covid Certificate
FR-6	Send Request	If plasma required then donor get the notification
FR-7	Estimation	Create the Plasma Donor Application Collect and store Plasma Donor/Recipient details in IBM cloud.

Non-functional Requirements:

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

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	• Navigate to home page • Locate Donor's Blood type • Find Donor next appointment • Find information about quarantine
NFR-2	**Security**	The users/donor details are stored in the cloud and it is secured with the user email id and password
NFR-3	**Reliability**	Ability of a solution to perform its required functions under stated condition for a specified period The system have the ability to work all the times without failure apart from network failure. The contact list of the donor are provided
NFR-4	**Performance**	The plasma donor application works well in every emergency situation. The easy interactive with the user and less interrupts.
NFR-5	**Availability**	The plasma Application is an online web application and it monitor 24/7
NFR-6	**Scalability**	The application offers multiple users and it is designed to protect the users information and Details

CHAPTER 5

PROJECT DESIGN

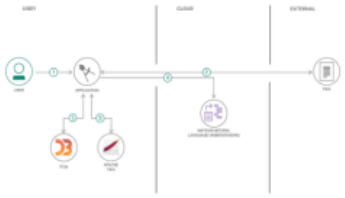
5.1 DATA FLOW DIAGRAMS

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.

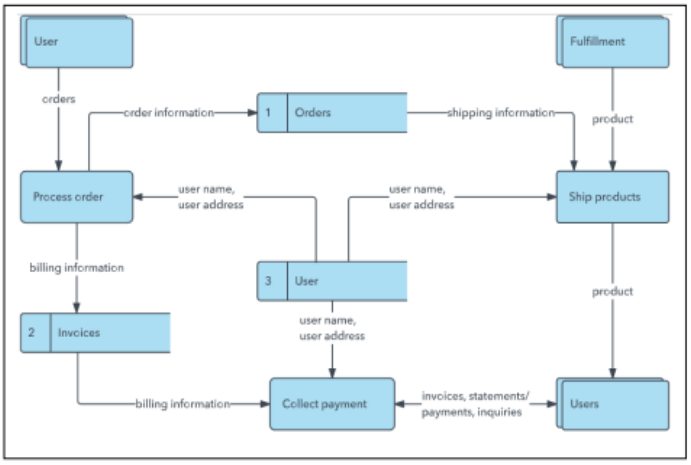
Example: (Simplified)

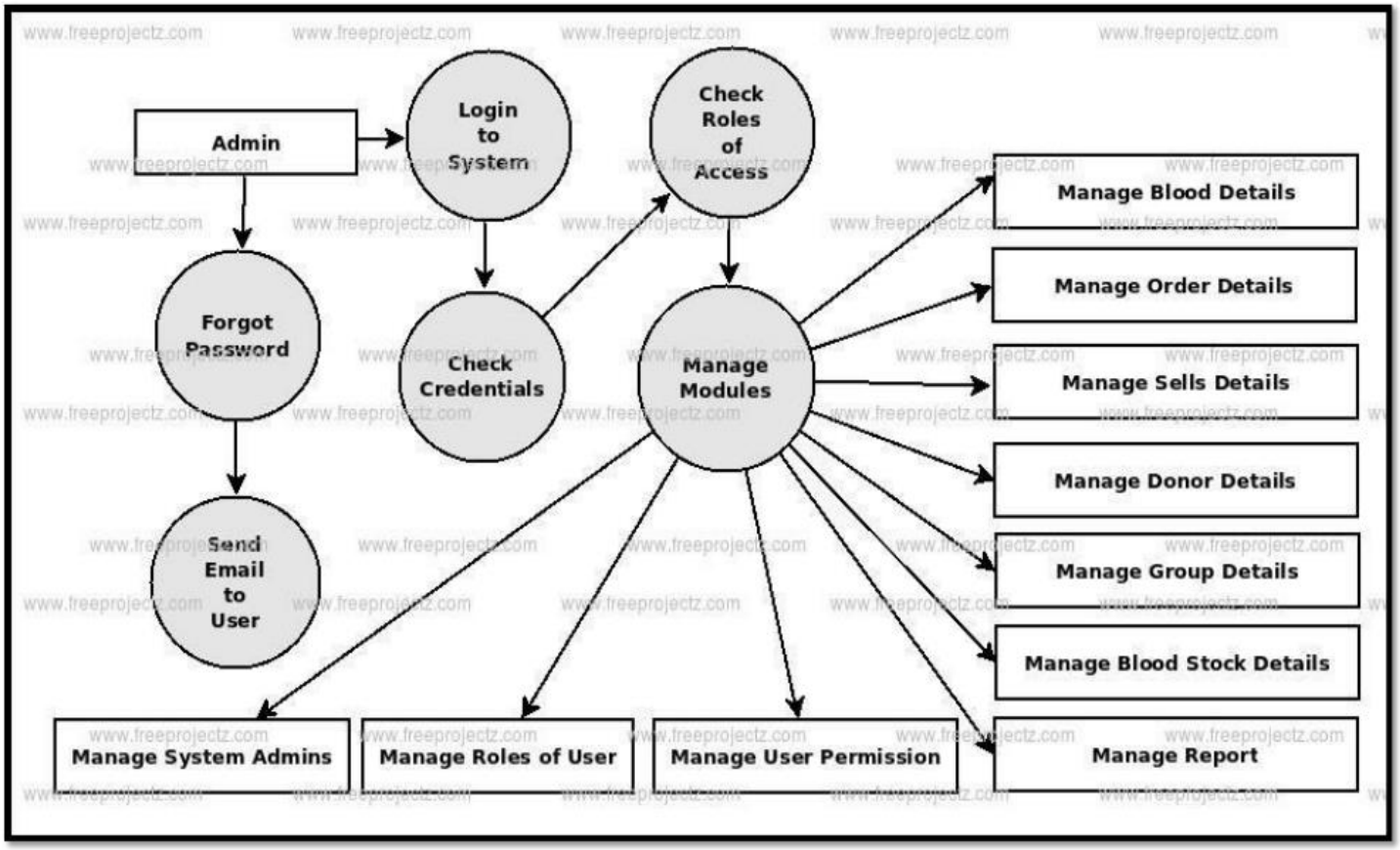
Flow

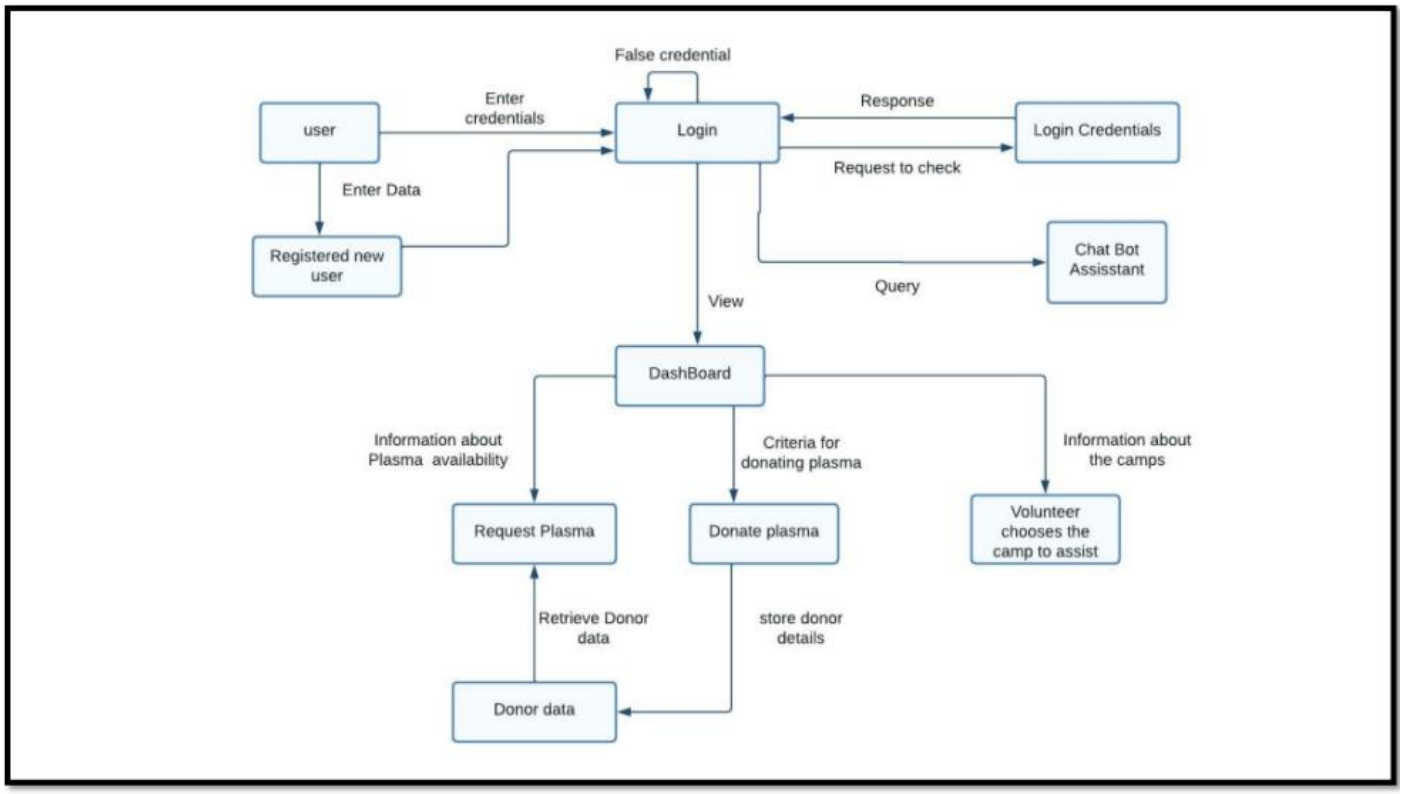


1. User configures credentials for the Watson Natural Language Understanding service and starts the app.
2. User selects data file to process and load.
3. Apache Tika extracts text from the data file.
4. Extracted text is passed to Watson NLU for enrichment.
5. Enriched data is visualized in the UI using the D3.js library.

Example: DFD Level 0 (Industry Standard)




Admin
Login to System
Check Roles of Access
Forgot Password
Check Credentials
Manage Modules
Send Email to User
Manage Blood Details
Manage Order Details
Manage Sells Details
Manage Donor Details
Manage Group Details
Manage Blood Stock Details
Manage Report
Manage System Admins
Manage Roles of User
Manage User Permission

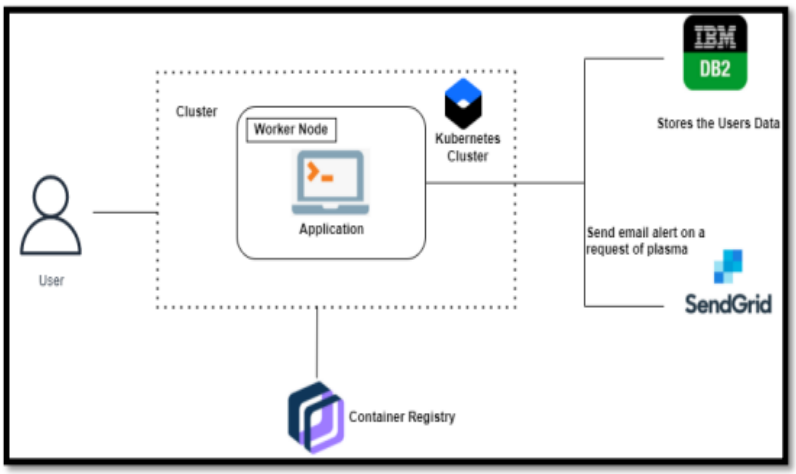


5.2 SOLUTION & TECHNICAL ARCHITECTURE

Technical Architecture:

The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2

Example: Order processing during pandemics for offline mode



Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicate external interfaces (third party API's etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g. Web UI, Mobile App etc.	HTML, CSS, JavaScript / Angular Js / React Js etc.
2.	Application Logic-1	Logic for a process in the application	Java / Python, Flask, HTML, CSS
3.	Application Logic-2	Logic for a process in the application	IBM DB2, Flask, HTML, CSS
4.	Application Logic-3	Logic for a process in the application	IBM Watson Assistant
5.	Database	Data Type, Configurations etc.	MySQL.
6.	Cloud Database	Database Service on Cloud	IBM DB2
7.	File Storage	File storage requirements	Docker, Kubernetes
8.	External API-1	Purpose of External API used in the application	Sendgrid Connect to Backend
9.	External API-2	Purpose of External API used in the application	Connect to Third party application
10.	Infrastructure (Server / Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration :	Local, Cloud Foundry, Kubernetes, etc.

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Docker, Kubernetes
2.	Security Implementations	List all the security / access controls implemented, use of firewalls etc.	Transport Layer Security (TLS),Doctor Content Trust(DCT),
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Lambda
4.	Availability	Justify the availability of application (e.g. use of load balancers, distributed servers etc.)	kubernetes
5.	Performance	Design consideration for the performance of the application (number of requests per sec, use of Cache, use of CDN's) etc.	Send GRID Docker and kubernetes

5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer Care Executive	Application	USN-8	As a customer care executive, I can try to address user's concerns and questions	I can view and address their concerns and questions	Medium	Sprint-2
Administrator	Application	USN-9	As an administrator I can help with user-facing aspects of a website, like its appearance, navigation and use of media.	I can change the appearance and navigation in a user friendly manner	Medium	Sprint-3
		USN-10	As an administrator, I can involve working with the technical side of websites.	I can help with such as troubleshooting issues, setting up web hosts, ensuring users have access and programming servers	Medium	Sprint-1
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Chatbot	Dashboard	USN-11	In addition the Customer care executive, chatbot can try to address user's concerns and questions	I can reply to all the queries related to our application	Medium	Sprint-3

CHAPTER 6

PROJECT PLANNING & SCEDULING

6.1 SPRINT PLANNING & ESTIMATION

Product Backlog, Sprint Schedule, and Estimation (4 Marks)

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	8	High	Team Lead
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application	5	High	Team Member-1
Sprint-2		USN-3	As a user, I can register for the application through Facebook	8	Low	Team Member-2
Sprint-1		USN-4	As a user, I can register for the application through Gmail	8	Medium	Team Member-3
Sprint-1	Login	USN-5	As a user, I can log into the application by entering email & password	10	High	Team Lead, Team Member-1
Sprint-2	Dashboard	USN-6	As a user , I can search the blood group for which I need plasma.	7	High	Team Member-1, Team Member-2
Sprint-2	Dashboard	USN-7	As a user, I can see login page and registration page for which the user login and searches for the required blood group plasma.	8	Medium	Team Lead, Team Member-3
Sprint-3	Dashboard	USN-8	As a customer care executive, I can solve the queries of the users.	9	High	Team Member-2,

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
						Team Member-3
Sprint-4	Registration	USN-9	As an Administrator, I can view the database of the registered users.	7	Medium	Team Lead, Team Member-2
Sprint-4	Dashboard	USN-10	As an Administrator, I can view how many members need what kind of blood group for plasma.	9	Low	Team Member-2, Team Member-3
Sprint-4	Dashboard	USN-11	In addition to the customer care executive. I can solve all the queries of the donors as well as the recipient.	5	Medium	Team Lead, Team Member-2
Sprint-1	Home Page	USN-12	As a user, I can view the homepage of the website and look up general plasma treatment information	5	Low	Team Member-2
Sprint-1	Send Request	USN-13	As a user, I can raise a request for plasma donation with specific requirements.	6	High	Team Lead
Sprint-3	View Request	USN-14	As a user, I can view requests for plasma donation verified by admin	8	Medium	Team Member-3
Sprint-4	Maintenance	USN-15	As an admin, I can maintain the databases involved	5	Medium	Team Member-4
Sprint-2	Search for Donor	USN-16	As an admin, I can search for suitable donors in the database based on requests received.	7	High	Team Member-2

6.2 SPRINT DELIVERY SCHEDULE

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

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

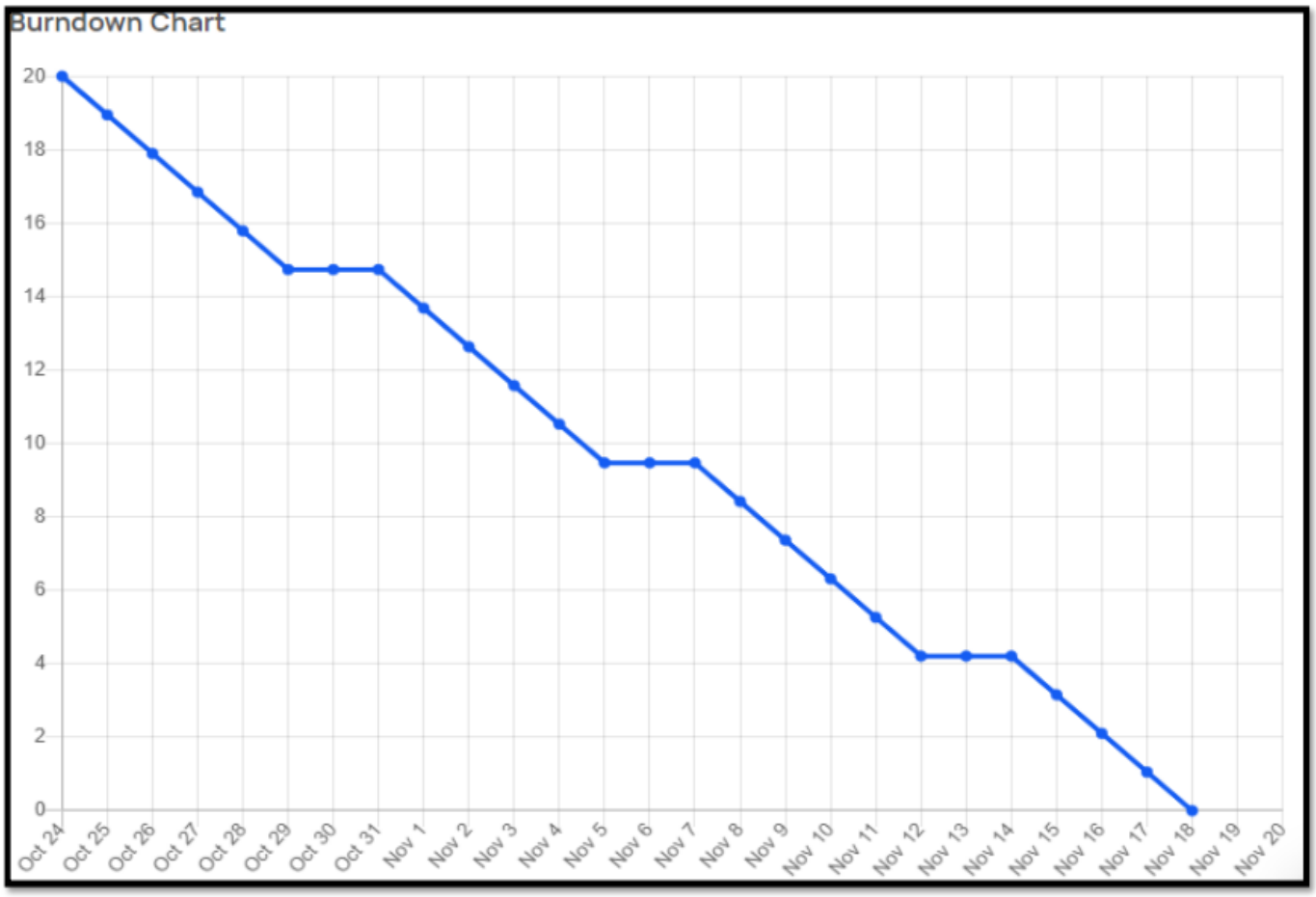
Velocity:
Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate the team's average velocity (AV) per iteration unit (story points per day)

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

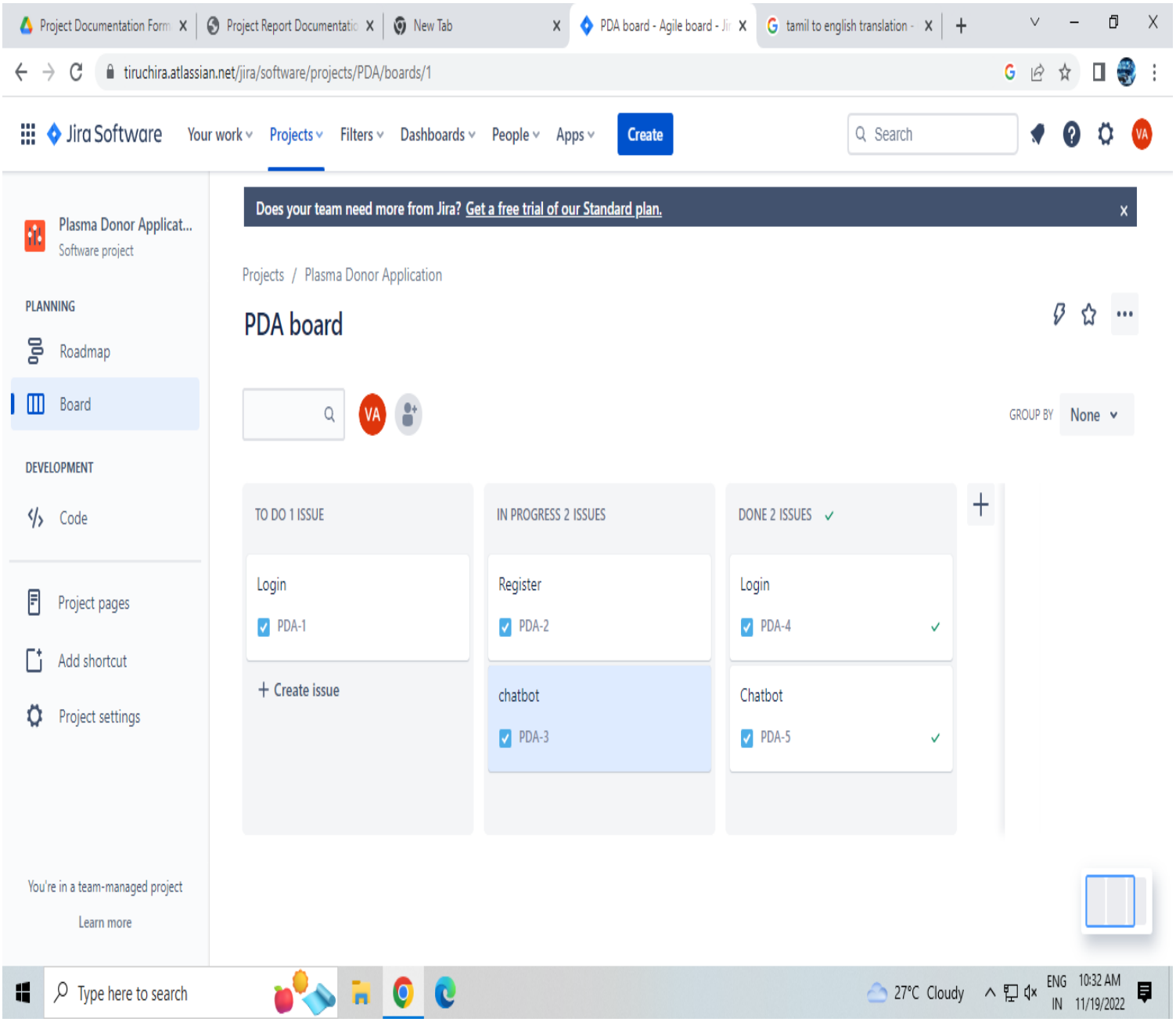
Sprint	Average Velocity
Sprint-1	6.5
Sprint-2	8
Sprint-3	7.6
Sprint-4	8.5

Total Average Velocity =7.65

BURNDOWN CHART:



6.3 REPORTS FROM JIRA



CHAPTER 7

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

7.1 FEATURE 1

Login:

```
{% extends "base.html" %}
{% block content %}

<div class="container">
   <div class="col-sm-12 d-flex p-2 justify-content-center" style="background-image: url('{{
url_for('static',filename='img/bg.png')}}');  background-size; 555px;  margin-top: 7%;">
   <div class="card">
     <div class="card-body alert-info">
<form action = "/login" method="post">
  <div class="form-group">
    <label for="email">Username:</label>
    <input type="text" class="form-control" placeholder="Enter username" id="username"
name="username">
  </div>
  <div class="form-group">
    <label for="pwd">Password:</label>
    <input type="password" class="form-control" placeholder="Enter password"
id="password" name="password">
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
<br><br>
<div class="alert-danger">{{msg}} </div>
</form>
  </div></div></div>
    </div>
</body>
</html>
{% endblock %}
```

Base:

```
<!doctype html>
```

```
<html lang="en">
  <head>
<style>
      img:hover {
        animation: shake 0.5s;
        animation-iteration-count: infinite;
      }
     @keyframes shake {
        0% { transform: translate(1px, 1px) rotate(0deg); }
        10% { transform: translate(-1px, -2px) rotate(-1deg); }
        20% { transform: translate(-3px, 0px) rotate(1deg); }
        30% { transform: translate(3px, 2px) rotate(0deg); }
        40% { transform: translate(1px, -1px) rotate(1deg); }
        50% { transform: translate(-1px, 2px) rotate(-1deg); }
        60% { transform: translate(-3px, 1px) rotate(0deg); }
        70% { transform: translate(3px, 1px) rotate(-1deg); }
        80% { transform: translate(-1px, -1px) rotate(1deg); }
        90% { transform: translate(1px, 2px) rotate(0deg); }
        100% { transform: translate(1px, -2px) rotate(-1deg); }
      }
      </style>
     <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
     <link rel="stylesheet" href="{{
url_for('static',filename='stylesheet/bootstrap.min.css')}}">
    <link rel="stylesheet" href="{{
url_for('static',filename='stylesheet/jquery.dataTables.min.css')}}">

    <link rel="stylesheet" href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css"
integrity="sha384-
AYmEC3Yw5cVb3ZcuHtOA93w35dYTsvhLPVnYs9eStHfGJvOvKxVfELGroGkvsg+p"
crossorigin="anonymous"/>

    <!--Bootstrap JSS-->
```

```
    <script src="{{ url_for('static',filename='stylesheet/jquery-3.5.1.js')}}"
crossorigin="anonymous"></script>
     <script src="{{ url_for('static',filename='stylesheet/bootstrap.min.js')}}"
crossorigin="anonymous"></script>


<title>Donate Plasma and Save lives</title>

 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a class="navbar-brand" href="/"><img src="{{ url_for('static',filename='img/logo.png')}}"
style="width: 60px; height: 50px;"></a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse justify-content-end" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/doner">Search Donor</a>
      </li>

          <li class="nav-item">
        <a class="nav-link" href="/contact_us">Contact US</a>
      </li>

      <li class="nav-item">
        <a class="nav-link" href="/about_us">About US</a>
      </li>




      <li class="nav-item">
        <a class="nav-link disabled" href="/login">Admin</a>
      </li>
```

```
    </ul>
  </div>
</nav>

</head>

<body>
{% block content %}
{% endblock %}
</body>
<hr>
<div class="text-center py-3 text-white bg-dark">© 2022 Copyright | PLSAMA DONOR
APPLICATION  </div>
<!-- footer core files start-->
<script>$(document).ready(function() {
    $('#needyamin_data').DataTable();
} );</script>

<script src="{{ url_for('static',filename='stylesheet/jquery.dataTables.min.js')}}"
crossorigin="anonymous"></script>
<!-- footer core files end-->
 </html>
```

7.1 FEATURE 2:

footer:

```
<div class="text-center py-3 text-white bg-dark">© 2022 Copyright | PLASMA DONOR

APPLICATION  </div>

 </html>
```

Header:

```
<!doctype html>
<html lang="en">
  <head>


    <!-- Required meta tags -->
```

```
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel= "stylesheet" type= "text/css" href= "{{
url_for('static',filename='stylesheet/bootstrap.min.css') }}">

  <!--Bootstrap JSS-->
    <script src="{{ url_for('static',filename='stylesheet/jquery-3.5.1.js')}}"
crossorigin="anonymous"></script>
     <script src="{{ url_for('static',filename='stylesheet/bootstrap.min.js')}}"
crossorigin="anonymous"></script>


<title>Donate Plasma and Save lives</title>

 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a class="navbar-brand" href="index.html"><img src="{{
url_for('static',filename='img/logo.png')}}" style="width: 60px; height: 50px;"></a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse justify-content-end" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
      </li>
```

```
      <li class="nav-item">
        <a class="nav-link" href="/doner">Search Donor</a>
      </li>

      <li class="nav-item">
        <a class="nav-link disabled" href="login.html">Admin</a>
      </li>
    </ul>
  </div>
</nav>


</head>
```

Index:

```
<!doctype html>
<html lang="en">
  <head>

    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel= "stylesheet" type= "text/css" href= "{{
url_for('static',filename='stylesheet/bootstrap.min.css') }}">
```

```
    <!--Bootstrap JSS-->
    <script src="{{ url_for('static',filename='stylesheet/jquery-3.5.1.js')}}"
crossorigin="anonymous"></script>
     <script src="{{ url_for('static',filename='stylesheet/bootstrap.min.js')}}"
crossorigin="anonymous"></script>


<title>Plasma Index </title>

 <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <a class="navbar-brand" href="index.html"><img src="{{
url_for('static',filename='img/logo.png')}}" style="width: 60px; height: 50px;"></a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse justify-content-end" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item active">
        <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="/doner">Search Donor</a>
      </li>

      <li class="nav-item">
        <a class="nav-link disabled" href="login.html">Admin</a>
```

```
      </li>
    </ul>
  </div>
</nav>
</head>
<body>
<div class="container-fluid">
<div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">
  <ol class="carousel-indicators">
    <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>
    <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
    <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
  </ol>
  <div class="carousel-inner" >
    <div class="carousel-item active">
      <img class="d-block w-100" src="{{ url_for('static',filename='img/slider1.jpg')}}"
height="350px" alt="First slide">
    </div>
    <div class="carousel-item">
      <img class="d-block w-100" src="{{ url_for('static',filename='img/slider2.jpg')}}"
height="350px" alt="Second slide">
    </div>
    <div class="carousel-item">
      <img class="d-block w-100" src="{{ url_for('static',filename='img/slider3.png')}}"
height="350px" alt="Third slide">
    </div>
  </div>
  <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-
slide="prev">
```

```
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
<div class="card">
  <div class="card-body bg-success text-white" style="border-radius: 5px;">
<div class="row justify-content-center">
    <h2>Bring a life back to power. Make Plasma donation your responsibility</h2></div>
  </div>
</div>
<div class="row p-1">

<div class="col-sm-6">
    <div class="card">
      <div class="card-body">
        <h5 class="card-title">Why Plasma Is Needed</h5>
<p class="card-text">
```

The reason to donate is simple…it helps save lives. In fact, the corona positive cases rising every day, saving lives has become the prime matter of concern. As per the data provided by WHO more than 3 million people have died due to the coronavirus (https://covid19.who.int/). However, apart from vaccination, there is another scientific method by which a covid infected person can be treated and the death risk can be reduced. 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 coronavirus.

There are many reasons patients need plasma. A common misunderstanding about plasma usage is that corono victims are the patients who use the most plasma. Actually, people needing the most plasma include those:

 Being treated for cancer
 Undergoing orthopedic surgeries
 Undergoing cardiovascular surgeries
Being treated for inherited blood disorders

In fact, according to a 2011 survey by the Department of Health and Human Services, here is the actual breakdown of how blood is used in the U.S.
Platelet Usage Red Blood Cells Usage
What Is Plasma

<img src="{{ url_for('static',filename='img/notice.jpg')}}" class="img-fluid" width="370">

```
        </div>
      </div>
 </div>
 <div class="col-sm-6">
   <div class="card">
     <div class="card-body">
       <h5 class="card-title" style="font-style: italic;">GIVE BLOOD SAVE LIFE
(Registration Form)</h5>


<form class="form-horizontal" style="background: skyblue; border-radius: 15px;" action =
"/blood_database" method="post">
<fieldset>

<!-- Form Name -->

<!-- Text input-->
<div class="form-group p-1">
  <label class="col-md-8 control-label" for="name" style="font-style: italic;">Name</label>
  <div class="col-md-12">
  <input id="name" name="name" type="text" placeholder="Enter your name" class="form-
control input-md" required="">

  </div>
</div>

<!-- Text input-->
<div class="form-group">
  <label class="col-md-4 control-label" for="email" style="font-style: italic;">Email</label>
```

```
  <div class="col-md-12">
  <input id="email" name="email" type="text" placeholder="Enter your email id"
class="form-control input-md" required="">

  </div>
</div>
<!-- Text input-->
<div class="form-group">
  <label class="col-md-4 control-label" for="contact" style="font-style: italic;">Contact
no.</label>
  <div class="col-md-12">
  <input id="number" name="number" type="number" placeholder="Enter your contact no."
class="form-control input-md" required="">

  </div>
</div>
<div class="row md-12 justify-content-center">

<div class="form-group">
  <label class="col-md-12 control-label" for="blood_group" style="font-style: italic;">Blood
Group</label>
  <div class="col-md-12">
    <select id="blood_group" name="blood_group" class="form-control">
      <option value="-1">Select</option>
      <option value="1">A+</option>
      <option value="2">B+</option>
      <option value="3">AB+</option>
      <option value="4">O+</option>
      <option value="5">A-</option>
```

```
      <option value="6">B-</option>
      <option value="7">AB-</option>
      <option value="8">O-</option>
    </select>
  </div>
</div>

<div class="form-group">
  <label class="col-md-12 control-label" for="contact" style="font-style: italic;">Age</label>
  <div class="col-md-12">
<select id="age" name="age" class="form-control">
      <option value="-1">Select</option>
      <option value="1">18</option>
      <option value="2">19</option>
      <option value="3">20</option>
      <option value="4">21</option>
      <option value="5">22</option>
      <option value="6">23</option>
      <option value="7">24</option>
      <option value="8">25</option>
    <option value="8">26</option>
    <option value="8">27</option>
    <option value="8">28</option>
    <option value="8">29</option>
    <option value="8">30</option>
    <option value="8">31</option>
    <option value="8">32</option>

    </select>
```

```
  </div>
</div>
 <div class="form-group ">
  <label class="col-md-12 control-label" for="contact" style="font-style: italic;">Doner Type</label>
  <div class="col-md-12">
  <select id="donar_type" name="donar_type" class="form-control">
      <option value="-1">Select</option>
      <option value="1">New</option>
      <option value="2">Regular</option>

    </select>
  </div>
</div>
 <div class="form-group ">
  <label class="col-md-12 control-label" for="contact" style="font-style: italic;">Gender</label>
  <div class="col-md-12">
  <select id="gender" name="gender" class="form-control">
      <option value="-1">Select</option>
      <option value="1">Male</option>
      <option value="2">Female</option>
       <option value="2">Other</option>
    </select>
  </div>
</div>

    </div>
```

```
<!-- Text input-->
<div class="form-group">
  <label class="col-md-4 control-label" for="street" style="font-style: italic;">Street</label>
  <div class="col-md-12">
  <input id="street" name="street" type="text" placeholder="Enter your street" class="form-control input-md" required="">

  </div>
</div>

<!-- Text input-->
<div class="form-group">
  <label class="col-md-4 control-label" for="area" style="font-style: italic;">Area</label>
  <div class="col-md-12">
  <input id="area" name="area" type="text" placeholder="Enter your area" class="form-control input-md" required="">

  </div>
</div>

<!-- Text input-->
<div class="form-group">
  <label class="col-md-4 control-label" for="city" style="font-style: italic;">City</label>
  <div class="col-md-12">
  <input id="city" name="city" type="text" placeholder="Enter your city" class="form-control input-md" required="">

  </div>
```

```
</div>

<!-- Text input-->
<div class="form-group">
  <label class="col-md-4 control-label" for="dist" style="font-style: italic;">District</label>
  <div class="col-md-12">
  <input id="district" name="district" type="text" placeholder="Enter your district"
class="form-control input-md" required="">

  </div>
</div>

<div class="form-group">
  <div class="row justify-content-center">
    <input type="submit" name="submit" id="submit" value="Submit" class="btn btn-
primary" style="border-radius: 10px;"> </div> </div>

</fieldset>
</form>
 </div>
    </div>
  </div>
 </div>
 </body>
<div class="text-center py-3 text-white bg-dark">© 2020 Copyright | YAMIN HOSSAIN
SHOHAN  </div>
</html>
```

about_us:

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

```
{% block content %}
 <div class="container">
  <div class="row">
    <div class="col">

<div class="container p-1">
<div class="row justify-content-center p-2 bg-success text-white">
    <h2>Give a Plasma Saves a lifes</h2></div> </div>
      <hr>

<h1> ABOUT US</h1>
This system proposed here aims at connecting the donors & the patients by an online web application. <br><br>

By using this application, the users can either raise a request for plasma donation or requirement.<br><br>

This system is used if anyone needs a Plasma Donor. This system comprises of Admin and User where both can request for a Plasma.<br><br>
In this system there is something called an active user, which means the user is an Active member of the App and has recovered from Covid 19, only such people are recommended here for Plasma Donation.<br>
 </div> </div></div>

    {% endblock %}
```

add:

```
<!DOCTYPE html>
<html>
```

```
<head>
   <title>Add Employee</title>
</head>
<body>
   <h2>Employee Information</h2>
   <form action = "/savedetails" method="post">
   <table>
      <tr><td>Name</td><td><input type="text" name="name"></td></tr>
      <tr><td>Email</td><td><input type="email" name="email"></td></tr>
      <tr><td>Address</td><td><input type="text" name="address"></td></tr>
      <tr><td><input type="submit" value="Submit"></td></tr>
   </table>
   </form>
</body>
</html>
```

all:

```
{% extends "base.html" %}
{% block content %}
<link rel="stylesheet" href="{{ url_for('static',filename='stylesheet/style.css')}}">
<div class="container">
  <div class="row">
    <div class="col">
<div class="container p-1">
<div class="row justify-content-center p-2 bg-success text-white">
   <h2>All Plasma Donater List</h2></div> </div>
     <hr>
     <div class="alert-danger">{{msg}} </div>
```

```
<table id="needyamin_data" class="table-sm table-striped table-bordered" style="text-align: center;">

      <thead>
        <tr>
          <th>Name</th>
          <th>Email</th>
          <th>Number</th>
          <th>Gender</th>
          <th>Blood Group</th>
          <th>IP</th>
          <th>Creation Date</th>
          <th> </th>
        </tr>
      </thead>
 <tbody>
     {% for row in rows %}
          <tr>
           <td><a href="/user/{{row['id']}}">{{row["name"]}}</a></td>
<td>
   <div class="btn btn-light" data-toggle="collapse" data-target="#email{{row['id']}}" title="Click for View Email">***@***.com</div>
              <div id="email{{row['id']}}" class="collapse btn-dark" style="padding: 5px;">
                {{row["email"]}} </div>
              </td>
              <td>
<div class="btn btn-light" data-toggle="collapse" data-target="#phone{{row['id']}}" title="Click for View Number">+****</div>
<div id="phone{{row['id']}}" class="collapse btn-dark" style="padding: 5px;">
```

```
  {{row["number"]}} </div>
  </td>
            <td>{{row["gender"]}}</td>
            <td> {{row["blood_group"]}}</td>
            <td>{{row["ip"]}} </td>
            <td>{{row["date"]}}</td>
            <td>
       <div class="row">
     <div class="col-sm">
      <form action="pending_del" method="POST">
     <input type="hidden" name="id" value="{{row['id']}}">
  <button type="submit" name="submit" value="Delete" class="alert-danger"> <i class="fas
fa-trash-alt"></i> </button></form>
    </div>
  </div>
  </td>
          </tr>
         {% endfor %}
    </tbody>
      </table>
  </div> </div></div>
   {% endblock %}
```

Contact_us

```
    {% extends "base.html" %}
{% block content %}
<div class="container">
    <div class="row">
      <div class="col">
```

```
   <div class="container p-1">
  <div class="row justify-content-center p-2 bg-success text-white">
     <h2>Contact Us Form</h2></div> </div>
  <div class="row">
    <div class="col-sm-6">
     <div class="card">
     <div class="card-body">
              <h2 class="card-title">Our Location</h2>
    <iframe
src="https://www.google.com/maps/embed?pb=!1m14!1m8!1m3!1d2006223.649497653!2d78.706355!3d10.856726!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x3baafa469e37c771%3A0xd44017e1b9c97cee!2sTrichy%20Engineering%20College!5e0!3m2!1sen!2sin!4v1668665143010!5m2!1sen!2sin" width="500" height="450" style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-downgrade"></iframe>
        </div> </div>
          </div>
  <div class="col-sm-6">
        <div class="card">
       <div class="card-body">
  <form action = "/contact_us" method="post">
          <div class="form-group">
            <label for="recipient-name" class="col-form-label">Your Name:</label>
            <input type="text" name="name" class="form-control" id="recipient-name" required="">
          </div>
         <div class="form-group">
            <label for="recipient-name" class="col-form-label">Your Email:</label>
            <input type="email" name="email" class="form-control" id="recipient-name" required="">
```

```
        </div>
        <div class="form-group">
          <label for="recipient-name" class="col-form-label">Phone:</label>
          <input type="phone" name="phone" class="form-control" id="recipient-name"
required="">
        </div>
         <div class="form-group">
          <label class="col-md-12 control-label" for="blood_group" style="font-style:
italic;">Request for blood group</label>
          <div class="col-md-12">
            <select id="blood_group" name="blood_group" class="form-control">
              <option>Select</option>
              <option value="A+">A+</option>
              <option value="B+">B+</option>
              <option value="AB+">AB+</option>
              <option value="O+">O+</option>
              <option value="A-">A-</option>
              <option value="B-">B-</option>
              <option value="AB-">AB-</option>
              <option value="0-">O-</option>
            </select>
          </div>
        </div>
        <div class="form-group">
        <label for="message-text" class="col-form-label">Message:</label>
          <textarea class="form-control" id="message" name="message"></textarea>
        </div>

          <div class="form-group">
```

```
            <input style="text-align: center; width: 100%;" type="submit" name="submit"
value="Submit" class="btn btn-primary">
          </div>

        </form>
          </div></div>
      </div>
          </div>
       </div> </div></div>
{% endblock %}
```

Contact_users

```
{% extends "base.html" %}
{% block content %}
<link rel="stylesheet" href="{{ url_for('static',filename='stylesheet/style.css')}}">
<div class="container">
  <div class="row">
    <div class="col">
<div class="container p-1">
<div class="row justify-content-center p-2 bg-success text-white">
    <h2>Contact Us Request List</h2></div> </div>
      <hr>
      <div class="alert-danger">{{msg}} </div>

<table id="needyamin_data" class="table-sm table-striped table-bordered" style="text-align:
center;">

      <thead>
          <tr>
```

```
            <th>Name</th>
            <th>Email</th>
            <th>Number</th>
            <th>Gender</th>
            <th>Blood Group</th>
            <th>Message</th>
            <th>IP</th>
            <th>Creation Date</th>
            <th> </th>
          </tr>
        </thead>
        <tbody>
          {% for row in rows %}
            <tr>
              <td>{{row["name"]}}</td>
              <td>
              <div class="btn btn-light" data-toggle="collapse" data-target="#email{{row['id']}}"
title="Click for View Email">***@***.com</div>
                <div id="email{{row['id']}}" class="collapse btn-dark" style="padding: 5px;">
                  {{row["email"]}} </div>

              </td>
              <td>

 <div class="btn btn-light" data-toggle="collapse" data-target="#phone{{row['id']}}"
title="Click for View Number">+****</div>
<div id="phone{{row['id']}}" class="collapse btn-dark" style="padding: 5px;">
  {{row["number"]}} </div>
```

```
</td>
            <td>{{row["gender"]}}</td>
            <td> {{row["blood_group"]}}</td>
            <td>
              {{row['message']}}
            </td>
            <td>{{row["ip"]}} </td>
            <td>{{row["date"]}}</td>
            <td>

      <div class="row">
      <div class="col-sm">
     <form action="contact_del" method="POST">
    <input type="hidden" name="id" value="{{row['id']}}">
    <button type="submit" name="submit" value="Delete" class="alert-danger">
    <i class="fas fa-trash-alt"></i></button>
   </form>
   </div>
  </div>
   </td>
          </tr>
         {% endfor %}
  </tbody>
    </table>
  </div> </div></div>
  {% endblock %}
```

dashboard:

```
{% extends "base.html" %}
{% block content %}
```

```
  <div class="container p-1">
     <div class="card text-white bg-primary p-3">
  <div class="row">
    <div class="col">
<div href="login.html">
Welcome back! <b>{{username}}</b> (<a href='/logout' style='color:white;'>Logout</a>)
<br>
Current IP: {{ip}} </div>
</div>
<div class="col">

<div href="login.html">
Last login IP: {% for row in rows %} {{row["ip"]}}  {% endfor %}<br>
New Pending: <span class="badge badge-dark">{% for p in pending %} {{p[0]}}  {% endfor
%}</span>
</div>
    </div
        </div>
        </div>
    </div>
   <div class="container">
<div class="card">
  <div class="row p-2">
    <div class="col">
    <div class="text-center">
<button class="btn btn-primary btn-circle btn-xl" onclick="location.href='/all'">Total Doner
<span class="badge badge-dark">{% for t in total %} {{t[0]}}  {% endfor
%}</span></button>
```

```
<button class="btn btn-warning btn-circle btn-xl" onclick="location.href='/all'">Male  <span class="badge badge-dark">{% for m in male %} {{m[0]}}  {% endfor %}</span></button>
<button class="btn btn-danger btn-circle btn-xl" onclick="location.href='/all'">Female  <span class="badge badge-dark">{% for f in female %} {{f[0]}}  {% endfor %}</span></button>
<button class="btn btn-success btn-circle btn-xl" onclick="location.href='/pending'">Pending <span class="badge badge-dark">{% for p in pending %} {{p[0]}}  {% endfor %}</span></button>
<button class="btn btn-success btn-circle btn-xl" onclick="location.href='/contact_users'">Contact  <span class="badge badge-dark">{% for c in contact %} {{c[0]}}  {% endfor %}</span></button>
<button class="btn btn-success btn-circle btn-xl" onclick="location.href='/request'">Request <span class="badge badge-dark">{% for r in requestx %} {{r[0]}}  {% endfor %}</span></button>
</div>

<div class="col-md-12 d-flex justify-content-center">
<div id="piechart"></div>
<script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"></script>
<script type="text/javascript">
// Load google charts
google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(drawChart);

// Draw the chart and set the chart values
function drawChart() {
  var data = google.visualization.arrayToDataTable([
  ['Task', 'Count'],
  ['Total Users', {% for t in total %} {{t[0]}}  {% endfor %}],
  ['Female', {% for f in female %} {{f[0]}}  {% endfor %}],
```

```
  ['Male', {% for m in male %} {{m[0]}}  {% endfor %}],
]);  // Optional; add a title and set the width and height of the chart
  var options = {'title':'Average Report', 'width':550, 'height':400};
// Display the chart inside the <div> element with id="piechart"
  var chart = new google.visualization.PieChart(document.getElementById('piechart'));
  chart.draw(data, options);
}
</script>
    </div>
          </div></div></div></div>
  {% endblock %}

  delete:
<!DOCTYPE html>
<html>
<head>
    <title>delete record</title>
</head>
<body>
   <h3>Remove Employee from the list</h3>
   <form action="/deleterecord" method="post">
Employee Id <input type="text" name="id">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

delete_record:

```
<!DOCTYPE html>
```

```
<html>
<head>
   <title>delete record</title>
</head>
<body>
<h3>{{msg}}</h3>
<a href="/view">View List</a>
</body>
</html>
```

doner:

```
{% extends "base.html" %}
{% block content %}

<div class="container">
  <div class="row">
    <div class="col">

<div class="container p-1">
<div class="row justify-content-center p-2 bg-success text-white">
   <h2>Your plasma can save others life.. Plasma donation is your responsibility</h2></div>
</div>
     <hr>

<table id="needyamin_data" class="table-sm table-striped table-bordered" style="text-align:
center;">

       <thead>
          <tr>
```

```
            <th>District</th>
            <th>City</th>
            <th>Area</th>
            <th>Age</th>
            <th>Blood Group</th>
            <th>Type</th>
            <th> </th>
          </tr>
        </thead>
        <tbody>
          {% for row in rows %}
            <tr>
            <td>{{row["district"]}}</td>
            <td>{{row["city"]}}</td>
            <td>{{row["area"]}}</td>
            <td>{{row["age"]}}</td>
            <td>{{row["blood_group"]}}</td>
            <td>{{row["donar_type"]}}</td>
            <td><button type="button" class="btn-sm btn-info" data-toggle="modal" data-
target="#needyaminModal" data-whatever="@getbootstrap">Request</button> </td>
            </tr>

<!--Model form start-->
   <div class="modal fade" id="needyaminModal" tabindex="-1" role="dialog" aria-
labelledby="needyaminModalLabel" aria-hidden="true">
     <div class="modal-dialog" role="document">
       <div class="modal-content">
         <div class="modal-header">
           <h5 class="modal-title" id="needyaminModalLabel">Request</h5>
```

```
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">

        <form action="/requests" method="post">
          <div class="form-group">
            <input type="hidden" name="id" value="{{row['id']}}" class="form-control"
id="recipient-name">

            <label for="recipient-name" class="col-form-label">Your Name:</label>
            <input type="text" name="name" class="form-control" id="recipient-name"
required="">
          </div>

        <div class="form-group">
            <label for="recipient-name" class="col-form-label">Your Email:</label>
            <input type="email" name="email" class="form-control" id="recipient-name"
required="">
          </div>

         <div class="form-group">
            <label for="recipient-name" class="col-form-label">Phone:</label>
            <input type="phone" name="phone" class="form-control" id="recipient-name"
required="">
          </div>
            <div class="form-group">
            <label for="recipient-name" class="col-form-label">Hospital/Clinic Name</label>
```

```
            <input type="text" name="hospital" class="form-control" id="recipient-name">
          </div>

          <div class="form-group">
            <label for="message-text" class="col-form-label">Message:</label>
            <textarea class="form-control" id="message" name="message"></textarea>
          </div>

        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-secondary" data-
dismiss="modal">Close</button>
          <input type="submit" name="submit" id="submit" value="submit" class="btn btn-
primary">
        </div>
    </form>
       </div>
     </div>
   </div>
<!--Model form ends-->
{% endfor %}
 </tbody>
 </table>
 </div> </div></div>    {% endblock %}
```

pending:

```
  {% extends "base.html" %}
{% block content %}
<link rel="stylesheet" href="{{ url_for('static',filename='stylesheet/style.css')}}">
```

```
<div class="container">
  <div class="row">
    <div class="col">

<div class="container p-1">
<div class="row justify-content-center p-2 bg-success text-white">
    <h2>Pending Blood Donater List</h2></div> </div>
      <hr>
      <div class="alert-danger">{{msg}} </div>

<table id="needyamin_data" class="table-sm table-striped table-bordered" style="text-align:
center;">

        <thead>
            <tr>
                <th>Name</th>
                <th>Email</th>
                <th>Number</th>
                <th>Gender</th>
                <th>Blood Group</th>
                <th>IP</th>
                <th>Creation Date</th>
                <th> </th>
            </tr>
        </thead>
        <tbody>

          {% for row in rows %}
```

```
            <tr>
              <td>{{row["name"]}}</td>
              <td>
                <div class="btn btn-light" data-toggle="collapse" data-
target="#email{{row['id']}}" title="Click for View Email">***@***.com</div>
                <div id="email{{row['id']}}" class="collapse btn-dark" style="padding: 5px;">
                  {{row["email"]}} </div>

              </td>
              <td>
 <div class="btn btn-light" data-toggle="collapse" data-target="#phone{{row['id']}}"
title="Click for View Number">+****</div>
<div id="phone{{row['id']}}" class="collapse btn-dark" style="padding: 5px;">
  {{row["number"]}} </div>

</td>
              <td>{{row["gender"]}}</td>
              <td> {{row["blood_group"]}}</td>
              <td>{{row["ip"]}} </td>
              <td>{{row["date"]}}</td>
              <td>
         <div class="row">
        <div class="col-sm">
         <form action="update" method="POST">
          <input type="hidden" name="id" value="{{row['id']}}">
          <input type="submit" name="submit" value="Approve" class="alert-info">
         </form>
        </div>
    <div class="col-sm">
```

```
     <form action="pending_del" method="POST">
     <input type="hidden" name="id" value="{{row['id']}}">
     <input type="submit" name="submit" value="Delete" class="alert-danger">
    </form>
    </div
  </div>
    </td>
           </tr>
          {% endfor %}
   </tbody>
        </table>
</div> </div></div>
    {% endblock %}
```

profile:

```
{% extends "base.html" %}
{% block content %}

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" integrity="sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQVnN" crossorigin="anonymous">

<div class="row">
<div class="col-sm-3"> </div>
<div class="col-sm-6 p-2">
<div class="card">
    <div class="card-header">
```

```
      <strong class="card-title mb-3"> <a href="#"  onclick="Export()" title="Download As PDF"><i class="fa fa-file-pdf-o" aria-hidden="true"></i></a></strong>
    </div>
    <div class="card-body">
      <div class="mx-auto d-block">
 <h5 class="text-sm-center mt-2 mb-1">{% for row in rows %} {{row["name"]}}  {% endfor %}</h5>
        <div class="location text-sm-center"><i class="fa fa-map-marker"></i> {% for row in rows %} {{row["city"]}}  {% endfor %},{% for row in rows %} {{row["district"]}}  {% endfor %}
        </div>
      </div>
      <hr>
      <div class="card-text text-sm-center">

        <a href="tel:{% for row in rows %} {{row['number']}} {% endfor %}"><i class="fa fa-phone-square" aria-hidden="true"></i></a>
        <a href="mailto:{% for row in rows %} {{row['email']}} {% endfor %}"><i class="fa fa-envelope" aria-hidden="true"></i></a>

      </div>
    </div>
    <table class="table table-user-information" id="table">
     <tbody>
      <tr>
        <td>Email:</td>
        <td><a href="mailto:{% for row in rows %} {{row['email']}}  {% endfor %}">{% for row in rows %} {{row["email"]}}  {% endfor %}</a></td>
      </tr>
```

```
        <tr>
          <td>Phone:</td>
          <td>{% for row in rows %} {{row["number"]}}  {% endfor %}</td>
        </tr>
        <tr>
          <td>Gender</td>
          <td>{% for row in rows %} {{row["gender"]}}  {% endfor %}</td>
        </tr>

          <tr>
              <tr>
          <td>Blood Group</td>
          <td>{% for row in rows %} {{row["blood_group"]}}  {% endfor %}</td>
        </tr>
          <tr>
          <td>Home Address</td>
          <td>{% for row in rows %} {{row["street"]}}  {% endfor %}, {% for row in rows %}
{{row["area"]}}  {% endfor %}</td>
        </tr>
      </tbody>
    </table>
</div>
<div class="col-sm-3"> </div>
</div>
</div>
  <!-- pdfmake start -->
<center><div class="card"></div> <input type="button" id="btnExport" value="Export"
onclick="Export()" /> </div> </center>
  <script src="https://cdn.jsdelivr.net/npm/html-to-pdfmake/docs/browser.js"></script>
```

```
<!-- pdfmake files: -->
<script src='https://cdn.jsdelivr.net/npm/pdfmake@latest/build/pdfmake.min.js'></script>
<script src='https://cdn.jsdelivr.net/npm/pdfmake@latest/build/vfs_fonts.min.js'></script>
<!-- html-to-pdfmake file: -->
<script src="https://cdn.jsdelivr.net/npm/html-to-pdfmake/docs/browser.js"></script>
<script type="text/javascript">
  function Export() {
  //var val = htmlToPdfmake("profile.html");
  var val = htmlToPdfmake(`
  <div class="card-body">
     <div class="mx-auto d-block">


  <h5 class="text-sm-center mt-2 mb-1">{% for row in rows %} {{row["name"]}}  {% endfor %}</h5>
        <div class="location text-sm-center"><i class="fa fa-map-marker"></i> {% for row in rows %} {{row["city"]}}  {% endfor %},{% for row in rows %} {{row["district"]}}  {% endfor %}

      </div>
   </div>
     <hr>

     <div class="card-text text-sm-center">

        <a href="tel:{% for row in rows %} {{row['number']}} {% endfor %}"><i class="fa fa-phone-square" aria-hidden="true"></i></a>
        <a href="mailto:{% for row in rows %} {{row['email']}} {% endfor %}"><i class="fa fa-envelope" aria-hidden="true"></i></a>
```

```
      </div>
    </div>


    <table class="table table-user-information" id="table">
     <tbody>
       <tr>
         <td>Email:</td>
         <td><a href="mailto:{% for row in rows %} {{row['email']}}  {% endfor %}">{% for row in rows %} {{row["email"]}}  {% endfor %}</a></td>
       </tr>
       <tr>
         <td>Phone:</td>
         <td>{% for row in rows %} {{row["number"]}}  {% endfor %}</td>
       </tr>
       <tr>
         <td>Gender</td>
         <td>{% for row in rows %} {{row["gender"]}}  {% endfor %}</td>
       </tr>

         <tr>
            <tr>
         <td>Blood Group</td>
         <td>{% for row in rows %} {{row["blood_group"]}}  {% endfor %}</td>
       </tr>
         <tr>
         <td>Home Address</td>
```

```
        <td>{% for row in rows %} {{row["street"]}}  {% endfor %}, {% for row in rows %}
{{row["area"]}}  {% endfor %}</td>
       </tr>
     </tbody>
   </table> `, {
 tableAutoSize:true
});
   var dd = {content:val};
   pdfMake.createPdf(dd).download();
    }</script>
<!-- pdfmake end -->
{% endblock %}
```

request:

```
{% extends "base.html" %}
{% block content %}
<link rel="stylesheet" href="{{ url_for('static',filename='stylesheet/style.css')}}">

<div class="container">
  <div class="row">
    <div class="col">
<div class="container p-1">
<div class="row justify-content-center p-2 bg-success text-white">
   <h2> Blood Request List</h2></div> </div>
     <hr>
     <div class="alert-danger">{{msg}} </div>
<table id="needyamin_data" class="table-sm table-striped table-bordered" style="text-align:
center;">
```

```
        <thead>
          <tr>
            <th>Name</th>
            <th>Email</th>
            <th>Number</th>
            <th>Hospital</th>
            <th>Blood Group</th>
            <th>Message</th>
            <th>Requested For</th>
            <th>IP</th>
            <th>Creation Date</th>
            <th> </th>
          </tr>
        </thead>
        <tbody>
         {% for row in rows %}
            <tr>
              <td>{{row["r_name"]}}</td>
              <td>

               <div class="btn btn-light" data-toggle="collapse" data-
target="#email{{row['rid']}}" title="Click for View Email">***@***.com</div>
                <div id="email{{row['rid']}}" class="collapse btn-dark" style="padding: 5px;">
                  {{row["r_email"]}} </div>
              </td>
              <td>
<div class="btn btn-light" data-toggle="collapse" data-target="#phone{{row['rid']}}"
title="Click for View Number">+****</div>
<div id="phone{{row['rid']}}" class="collapse btn-dark" style="padding: 5px;">
```

```
 {{row["r_phone"]}} </div>
</td>
            <td>{{row["hospital"]}}</td>
            <td> {{row["blood_group"]}}</td>
            <td>
              {{row['message']}}

            </td>
            <td> <a href="/user/{{row['id']}}">{{row["name"]}}</a>
            </td>
            <td>{{row["rip"]}} </td>
            <td>{{row["rdate"]}}</td>
            <td>

     <div class="row">
    <div class="col-sm">
    <form action="request_del" method="POST">
   <input type="hidden" name="id" value="{{row['rid']}}">
   <button type="submit" name="submit" value="Delete" class="alert-danger">
   <i class="fas fa-trash-alt"></i></button>
  </form>
  </div>
 </div>
  </td>
        </tr>
        {% endfor %}
    </tbody>
     </table>
</div> </div></div>
```

{% endblock %}

success:

```
{% extends "base.html" %}
{% block content %}

<div class="container-fluid">


<div class='alert bg-success' role='alert' style="background-image: url('{{
url_for('static',filename='img/slider3.jpg')}}');  background-color: white;">
<div class="card">
  <div class="card-body bg-success text-white" style="border-radius: 5px;">
<div class="row justify-content-center">

        <h2>{{msg}}</h2></div>
  </div>
</div></div>


<script> //windows-alert hidden future needyamin
window.setTimeout(function() {
    $(".alert").fadeTo(500, 0).slideUp(500, function(){
        $(this).remove();
    });
}, 2000);
</script>

<div class="text-center">

<style>


.hide {
  display: none;
}
.myDIV:hover + .hide {
  display: block;
  color: red;

text-align: top;

}
```

```
</style>

<div class="container">
<div class="row p-1">
  <div class="col-sm-12">
   <div class="card">
     <div class="card-body">

<div class="myDIV">
<img src="{{ url_for('static',filename='img/drop.png')}}" class="img-fluid" width="40%">

 </div> <div class="hide">You are a real superhero!</div>

       <h2>Thank you for share your blood group..</h2>

          </div>
            </div>
              </div>
                 </div>
                    </div></div> </div>
 {% endblock %}
```

view:

```
<!DOCTYPE html>
<html>
<head>
   <title>List</title>
```

```
</head>
<body>

<h3>Employee Information</h3>
<table border=5>
   <thead>
      <td>ID</td>
      <td>Name</td>
      <td>Email</td>
      <td>Address</td>
   </thead>

{% for row in rows %}
      <tr>
         <td>{{row["id"]}}</td>
         <td>{{row["name"]}}</td>
         <td>{{row["email"]}}</td>
         <td>{{row["address"]}}</td>
      </tr>

   {% endfor %}
</table>
<br><br>

<a href="/">Go back to home page</a>

</body>
</html>
?
```

CHAPTER 8

8.1 TEST CASES

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Comments	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_001	Functional	Home Page	Verify user is able to see the Login/Signup page whenever get into the application		1.Enter URL and click go 2.Click on the login/signup page 3.Verify login/Signup by entering the details	-	Login/Signup page should display	Working as expected	Pass				kosalaraman
LoginPage_TC_002	UI	Home Page	Verify the UI elements in Login/Signup page		1.Enter URL and click go 2.Click on Login/signup and get into next respective page. 3.Verify login/Signup page with below UI elements: a.email text box b.password text box c.Login button d.New User? Create account link	-	Application should show below UI elements: a.email text box b.password text box c.Login button with orange colour d.New customer? Create account link	Working as expected	Pass				Kishore kumar
LoginPage_TC_003	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL and click go 2.Click on login button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: demo@gmail.com password: 12345678	User should navigate to Donar/Recipient requesting page	Working as expected	pass				Madhankumar
LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go 2.Click on login button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: demo@gmail password: Testing123	Application should show "Incorrect email or password" validation message.	Working as expected	pass				Bharath
LoginPage_TC_005	Functional	Login page	Verify Admin is able to log into application with Valid credentials		1.Enter URL and click go 2.Click on login button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: adminm@gmail.com password: admin@m	Admin should navigate to Donar/Recipient requesting page	Working as expected	pass				Kishore kumar
LoginPage_TC_006	Functional	Login page	Verify Admin is able to log into application with InValid credentials		1.Enter URL(https://shopenzer.com/) and click go 2.Click on My Account dropdown button 3.Enter InValid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: adminm@gmail.com password: Adminm@	Application should show "Incorrect email or password" validation message.	Working as expected	pass				kosalaraman

8.2 USER ACCEPTANCE TESTING

1. Purpose of Document

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

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity1	Severity2	Severity3	Severity4	Subtotal
By Design	10	4	2	4	20
Duplicate	1	0	1	0	2
External	2	2	1	1	6
Fixed	4	1	1	10	16
Not Reproduced	0	0	0	0	0
Skipped	1	1	0	1	3
Won't Fix	0	2	2	0	4
Totals	18	10	7	16	51

3. Test Case Analysis

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	9	0	0	9
Client Application	10	0	0	10
Security	1	0	0	1
Outsource Shipping	0	0	0	0
Exception Reporting	9	0	0	9
Final Report Output	9	0	0	9
Version Control	1	0	0	1

CHAPTER 9

9.1 PERFORMANCE METRICS

Formal code metrics - Such as Lines of Code (LOC), code complexity, Instruction Path Length, etc. In modern development environments, these are considered less useful.

Developer productivity metrics—Such as active days, assignment scope, efficiency and code churn. These metrics can help you understand how much time and work developers are investing in a software project.

Agile process metrics—Such as lead time, cycle time and velocity. They measure the progress of a dev team in producing working, shipping-quality software features.

Operational metrics—Such as Mean Time Between Failures (MTBF) and Mean Time to Recover (MTTR). This checks how software is running in production and how effective operations staff are at maintaining it.

Test metrics—Such as code coverage, percent of automated tests, and defects in production. This measures how comprehensively a system is tested, which should be correlated with software quality

Customer satisfaction—Such as Net Promoter Score (NPS), Customer Effort Score (CES) and Customer Satisfaction Score (CSAT). The ultimate measurement of how customers experience the software and their interaction with the software vendor.

CHAPTER 10
ADVANTAGE & DISADVANTAGE

ADVANTAGE:

1. Earn Up to $4,000 per Year

What attracts many people to plasma donation is the fact that you can earn a substantial amount of money every time you donate. From start to finish the appointment takes about an hour, which means you can take home up to $40 for just 1 hour of your time. By the end of the year, your earnings can add up to $4,000 if you make just two donations every seven days.

2. Make an Impact

Plasma is used to create medicine that treats chronic illnesses including immune deficiencies, bleeding disorders, and other rare diseases. People around the world are in need of plasma transfusions, convalescent plasma, and medicine made from plasma every day. Since plasma can't be produced synthetically, patients who receive plasma protein therapies such as intravenous immunoglobulin and clotting factors are dependent on the generosity of donors.

3. Boost Your Mood

Research shows that helping others actually helps you. A study published in the International Journal of Behavioral Medicine found that altruism, or the concern for the well-being of others, is linked to greater emotional well-being, health, and can lead to an overall happier outlook. Whether you donate plasma, money, gifts, or your time, the simple act of giving is an effective way to boost your own happiness, lower stress levels, increase endorphins, and even improve social connections.

4. Maintain a Healthy Diet

Regular plasma donations can guide you toward healthier eating habits. At a donor center, donors are always encouraged to eat nutritious foods - filled with iron, protein, and vitamin C - and drink enough water to make their donations go smoothly. When the body is adequately hydrated, veins become more dilated, which allows for better blood flow during the appointment. It also helps your body replenish lost fluids faster, which is crucial, given that plasma is mostly water.

5. Reduce Cholesterol Levels

Donating plasma doesn't only make you more aware of your diet. It may also have a positive impact on your physical health. One health benefit of regular plasma donation is the potential reduction of bad cholesterol levels and the increase of good cholesterol, especially in women. A study has shown that if you have high total cholesterol levels, the collection process may be able to help regulate cholesterol in general.

6. Lower Blood Pressure

In addition to reducing cholesterol levels, becoming a regular plasma donor may have some residual effects on vital signs like blood pressure. A 2015 study indicates that blood pressure may decrease following plasma donation in individuals with high baseline blood pressure levels.

Give and Gain

- When you really think about it, the benefits of donating plasma are significant. Not only can you make thousands of dollars every year for a minimal amount of time and effort, but you can also experience lasting health benefits from the donation experience.
- Donating plasma —especially on a regular basis—may contribute to improvements in your overall health and mood. As you experience these benefits your donation will help members of our community live more comfortably, safely, and independently. Let that sink in.

DISADVANTAGE:

Dehydration

Plasma contains a lot of water. For that reason, some people experience dehydration after donating plasma. Dehydration after donating plasma is usually not severe.

Dizziness, fainting, and lightheadedness

Plasma is rich in nutrients and salts. These are important in keeping the body alert and functioning properly. Losing some of these substances through plasma donation can lead to an electrolyte imbalance. This can result in dizziness, fainting, and lightheadedness.

Fatigue

Fatigue can occur if the body has low levels of nutrients and salts. Fatigue after plasma donation is another common side effect, but it's usually mild.

Bruising and discomfort

Bruising and discomfort are among the milder and more common side effects of plasma donation.When the needle pierces the skin, you may experience a pinching feeling. You may also experience a dull, pulling sensation at the needle site as blood is drawn from your vein, into the tubing, and then into the machine collecting your plasma.

Infection

Any time a needle is used to pierce the skin, there is always a small risk of infection. Punctured skin tissue allows bacteria from outside the body to get in. The needle may carry bacteria not only beneath the skin's surface, but into a vein. This can lead to an infection at the injection site and surrounding body tissue or in the blood.

Signs of an infection include skin that feels warm and tender and looks red and swollen, with pain at and around the injection site. If you notice signs of infection, it's important to see a doctor right away to prevent complications.

Citrate reaction

A citrate reaction is a very serious but very rare side effect of plasma donation.During a plasma donation, the technician will infuse a substance known as an anticoagulant into the blood collected in the plasma-separating machine before the blood is returned to your body. This anticoagulant is meant to prevent blood clots from forming. The plasma in the machine retains most of the citrate, but some will also enter your bloodstream.

CHAPTER 11

CONCLUSION

Plasma donor application provides a reliable platform to connect local blood donors with patients. Plasma donor creates a communication channel through authenticated clinics whenever a patient needs Plasma donation.

It is a useful tool to find compatible Plasma donor who can receive Plasma request posts in their local area. Clinics can use this web application to maintain the Plasma donation activity. Future improvement of the Plasma donor is explained.

We show screenshots for the BLOODR application for different types of users including requester, donor, and administrator. Various features of the application are described and their needs of use are analyzed.

If a patient needs a blood at a clinic, blood donors in vicinity can be contacted through using a clinic management service provided in this application. Registered donors will get notification for the blood requests only if their blood group is compatible with the requested blood type and in the same city/region. Then matching blood donors can go to the requesting clinic and donate.

Plasma donation is a kind of citizen's social responsibility in which an individual can willingly donate .Plasma donor via our app. An authorised user at the centre and donor will keep his or her account, which is a significant innovation in our research.

This system guarantees the recipient's protection and the donor's privacy using J48 decision tree algorithm implemented in WEKA. The authorised user will look for several Plasma donors in his or her area or in other particular areas, and then message, notify, and call them. Furthermore, we checked our platform with a few people.

Applications with a better solution remove the obstacle to current Plasma donation. This Application has been created with the concept and has sought to make sure that the donor gives blood to community.

This model is made user friendly so anybody can download and maintain his/her account. Donor app will break the chain of business through Plasma donor and help the poor to find donor at free of cost. This project will help new Plasma donor banks improve their services and progress from traditional to user-friendly frameworks**.**

CHAPTER 12

FUTURE SCOPE

In future, our algorithm more congenial with more features such as

- The analysis such as Frequently requested zone or hospital for Blood, Number of donors, mostly asked Blood Group, Age group of Patients need for blood etc. can be added as additional features.

- The BDOOR can be implemented using Artificial Intelligence and Deep Learning Algorithms.

- NGOs and NCC Units information's can be made available in the application.

- Donors last donated details can be automatically updated in the App.

- Notification to Donors about the nearest Blood Donation Camp.

CHAPTER 13

APPENDIX

SOURCE CODE:

app.py

```
from flask import Flask, render_template, request, session, redirect
from datetime import datetime
from datetime import timedelta
import sqlite3

app = Flask(__name__)
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'
app.permanent_session_lifetime = timedelta(minutes=5)

con = sqlite3.connect("needyamin.db")
#print("Database opened successfully")
#con.execute("create table Employees (id INTEGER PRIMARY KEY AUTOINCREMENT,
name TEXT NOT NULL, email TEXT UNIQUE NOT NULL, address TEXT NOT NULL)")
#print("Table created successfully")

#index start#
@app.route("/")
def index():
    return render_template("index.html");
#index end#

#index start#
@app.route("/contact_us")
```

```
def contact():
    return render_template(“contact_us.html”);
#index end#


#about_us start#
@app.route(“/about_us”)
def about_us():
    return render_template(“about_us.html”);
#about_us end#

#about_us start#
@app.route(“/profile”)
def profile():
    return render_template(“profile.html”);
#about_us end#


#success start#
@app.route(“/success”)
def success():
    return render_template(“success.html”);
#dcma end#


##start login request POST##
@app.route(‘/login’,methods = [“POST”])
def login_post():
    session.permanent = True
```

```
    username = request.form[“username”]
    password = request.form[“password”]
    session[“user”] = username
    username = session[“user”]
    con = sqlite3.connect(“needyamin.db”)
    con.row_factory = sqlite3.Row
    cur = con.cursor()
    cur.execute(“select * from admin where username= ? AND password =
?;”,[username,password])
    user = cur.fetchall()
    msg = “username or password not match”
    if len(user)>0:
        return redirect(‘dashboard’)

    return render_template(“login.html”,msg = msg)
##end login request POST##



##redirect dashboard start##
@app.route(‘/dashboard’,methods=[‘GET’])
def dashboard():
 if request.method == “GET”:
     with sqlite3.connect(“needyamin.db”) as con:
         try:
             con.row_factory = sqlite3.Row
             cur = con.cursor()
             m = con.cursor()
             cur.execute(“SELECT * from admin limit 1”) #Admin last login
```

```
        rows = cur.fetchall()
        m.execute("SELECT COUNT(*) FROM users where gender='Male'") #Total Male
        male = m.fetchall()
        m.execute("SELECT COUNT(*) FROM users where gender='Female'") #Total
Female
        female = m.fetchall()
        m.execute("SELECT COUNT(*) FROM users") #Total Users
        total = m.fetchall()
        m.execute("SELECT COUNT(*) FROM users where status='0'") #Pending Users
        pending = m.fetchall()
        m.execute("SELECT COUNT(*) FROM contact_us") #Contact
        contact = m.fetchall()
        m.execute("SELECT COUNT(*) FROM request") #Request
        89equest = m.fetchall()
        session["user"]
        username = session["user"]
        session["user"] = username
        username = session["user"]
        ip = request.environ.get('HTTP_X_REAL_IP', request.remote_addr)
        if len(username)>0:
            return render_template('dashboard.html',username = username, ip = ip, total =
total, male = male,female = female, pending = pending, contact = contact, 89equest =
89equest, rows = rows)
    except:
            return render_template('login.html')
##redirect dashboard end##


###########Login check GET start####################
```

```
@app.route(‘/login’,methods=[‘GET’])
def loginxx():
 if request.method == “GET”:
    with sqlite3.connect(“needyamin.db”) as con:
        try:
            model.save() #hideValueError
            con.row_factory = sqlite3.Row
            cur = con.cursor()
            cur.execute(“select * from admin”)
            cur.execute(“select * from users”)
            rows = cur.fetchall()
            session[“user”]
            username = session[“user”]
            session[“user”] = username
            username = session[“user”]
            ip = request.environ.get(‘HTTP_X_REAL_IP’, request.remote_addr)
            if len(username)>0:
                return render_template(‘dashboard.html’,username = username, ip = ip, rows =
rows)
        except:
                    return render_template(‘login.html’)
####################Login check GET end####################


##auth start##
@app.route(‘/mem’,methods=[‘GET’, ‘POST’])
def mem():
 if request.method == “GET”:
    try:
```

```
        session[“user”]
        username = session[“user”]
        if len(username)>0:
            return render_template(‘dashboard.html’)
    except:
        return redirect(“login”)
##auth end##



###logout###
@app.route(‘/logout’)
def logout():
    session.clear()
    return redirect(‘login’)

###logout###

##loop donar table start#
@app.route(“/doner”)
def doner():
    con = sqlite3.connect(“needyamin.db”)
    con.row_factory = sqlite3.Row
    cur = con.cursor()
    cur.execute(“select * from users where status=’1’”)
    rows = cur.fetchall()
    return render_template(“doner.html”,rows = rows)
##loop donar table end#
```

```
##auth all users start##
@app.route('/all',methods=['GET', 'POST'])
def all_users():
 if request.method == "GET":
    with sqlite3.connect("needyamin.db") as con:
        try:
            con.row_factory = sqlite3.Row
            cur = con.cursor()
            cur.execute("select * from users")
            rows = cur.fetchall()
            session["user"]
            username = session["user"]
            if len(username)>0:
                return render_template("all.html",rows = rows)
        except:
                    return redirect("login")
##auth all users end##



##auth pending start##
@app.route('/pending',methods=['GET', 'POST'])
def pending():
 if request.method == "GET":
    with sqlite3.connect("needyamin.db") as con:
        try:
            con.row_factory = sqlite3.Row
            cur = con.cursor()
```

```
            cur.execute("select * from users where status='0'")
            rows = cur.fetchall()
            session["user"]
            username = session["user"]
            if len(username)>0:
                return render_template("pending.html",rows = rows)
        except:
                    return redirect("login")
##auth pending end##

#auth contact start#
@app.route('/contact_users',methods=['GET', 'POST'])
def contact_users():
 if request.method == "GET":
    with sqlite3.connect("needyamin.db") as con:
        try:
            con.row_factory = sqlite3.Row
            cur = con.cursor()
            cur.execute("select * from contact_us")
            rows = cur.fetchall()
            session["user"]
            username = session["user"]
            if len(username)>0:
                return render_template("contact_users.html",rows = rows)
        except:
                    return redirect("login")
#auth contact end##
```

```
###contacts del end
@app.route("/contact_del", methods=["POST"])
def contact_delete():
    id = request.form["id"]
    with sqlite3.connect("needyamin.db") as con:
        try:
            con.row_factory = sqlite3.Row
            cur = con.cursor()
            cur.execute("select * from contact_us")
            rows = cur.fetchall()
            cur.execute("delete from contact_us where id = ?",(id,))
            con.commit()
            msg = "record successfully deleted"
        except:
            msg = "can't be deleted"
        finally:
            return render_template("request.html",msg = msg, rows = rows)
            ### contacts del request end



#auth request start#
@app.route('/request',methods=['GET', 'POST'])
def request_users():
 if request.method == "GET":
    with sqlite3.connect("needyamin.db") as con:
        try:
            con.row_factory = sqlite3.Row
            cur = con.cursor()
```

```
        cur.execute("select * from request inner join users ON request.request_id = users.id")
        rows = cur.fetchall()
        session["user"]
        username = session["user"]
        if len(username)>0:
            return render_template("request.html",rows = rows)
    except:
            return redirect("login")
#auth request end##

###blood request end
@app.route("/request_del", methods=["POST"])
def request_delete():
    id = request.form["id"]
    with sqlite3.connect("needyamin.db") as con:
        try:
            con.row_factory = sqlite3.Row
            cur = con.cursor()
            cur.execute("select * from request inner join users ON request.request_id = users.id")
            rows = cur.fetchall()
            cur.execute("delete from request where rid = ?",(id,))
            con.commit()
            msg = "record successfully deleted"
        except:
            msg = "can't be deleted"
        finally:
            return render_template("request.html",msg = msg, rows = rows)
            ### blood request end
```

```
##blood request contact start##
@app.route("/requests",methods = ["POST","GET"])
def blood_request():
    msg = ""
    id = request.form["id"]
    if request.method == "POST":
        try:
            name = request.form["name"]
            email = request.form["email"]
            phone = request.form["phone"]
            hospital = request.form["hospital"]
            message = request.form["message"]
            today = datetime.utcnow()
            date = today.strftime('%Y-%m-%d')
            ip = request.environ.get('HTTP_X_REAL_IP', request.remote_addr)
            with sqlite3.connect("needyamin.db") as con:
                cur = con.cursor()
            cur.execute("INSERT into request (request_id, r_name, r_email, r_phone, hospital,
message, rip, rdate) values (?,?,?,?,?,?,?,?)",(id,name,email,phone,hospital,message,ip,date))
            con.commit()
            msg = "Thank you! Your request has been submitted.."

        except:
            con.rollback()
            msg = "We can not add your blood group request to the list"

        finally:
            return render_template("success.html",msg = msg)
```

```
            con.close()
    ##blood request contact end###


@app.route("/pending_del", methods=["POST"])
def pending_delete():
    id = request.form["id"]
    with sqlite3.connect("needyamin.db") as con:
        try:
            con.row_factory = sqlite3.Row
            cur = con.cursor()
            cur.execute("select * from users where status='1'")
            rows = cur.fetchall()
            cur.execute("delete from users where id = ?",(id,))
            con.commit()
            msg = "record successfully deleted"
        except:
            msg = "can't be deleted"
        finally:
            return render_template("pending.html",msg = msg, rows = rows)

##loop donar table pending end#


##update conditions start
@app.route("/update", methods=["GET"])
def update_router():
    return render_template("pending.html")
```

```
@app.route(“/update”, methods=[“POST”])
def update_users():
    id = request.form[“id”]
    with sqlite3.connect(“needyamin.db”) as con:
        try:
            con.row_factory = sqlite3.Row
            cur = con.cursor()
            cur.execute(“select * from users where status=’0’”)
            rows = cur.fetchall()
            cur.execute(“update users set status=’1’ where id = ?”,(id,))
            con.commit()
            msg = “record successfully updated”
        except:
            msg = “can’t be update”
        finally:
            return render_template(“pending.html”,msg = msg,rows = rows)
            #update condition end




##store blood donar information start##
@app.route(“/blood_database”,methods = [“POST”,”GET”])
def blood_g():
    msg = “”
    if request.method == “POST”:
        try:
            name = request.form[“name”]
```

```
        email = request.form[“email”]
        number = request.form[“number”]
        blood_group = request.form[“blood_group”]
        age = request.form[“age”]
        donar_type = request.form[“donar_type”]
        gender = request.form[“gender”]
        street = request.form[“street”]
        area = request.form[“area”]
        city = request.form[“city”]
        district = request.form[“district”]
        today = datetime.utcnow()
        date = today.strftime(‘%Y-%m-%d’)
        ip = request.environ.get(‘HTTP_X_REAL_IP’, request.remote_addr)
        status = 0
        with sqlite3.connect(“needyamin.db”) as con:
            cur = con.cursor()
        cur.execute(“INSERT into users (name, email, number, blood_group, age,
donar_type, gender, street, area, city, district, date, ip, status) values
(?,?,?,?,?,?,?,?,?,?,?,?,?,?)”,(name,email,number,blood_group,age,donar_type,gender,street,ar
ea,city,district,date,ip,status))
        con.commit()
        msg = “Thank you! Your information has been submitted..”

    except:
        con.rollback()
        msg = “We can not add your blood group request to the list”

    finally:
        return render_template(“success.html”,msg = msg)
```

```
            con.close()
    ##store blood donar information end###



##Contact Us start##
@app.route("/contact_us",methods = ["POST","GET"])
def contact_form():
    msg = "msg"
    if request.method == "POST":
        try:
            name = request.form["name"]
            email = request.form["email"]
            phone = request.form["phone"]
            blood_group = request.form["blood_group"]
            message = request.form["message"]
            today = datetime.utcnow()
            date = today.strftime('%Y-%m-%d')
            ip = request.environ.get('HTTP_X_REAL_IP', request.remote_addr)
            with sqlite3.connect("needyamin.db") as con:
                cur = con.cursor()
            cur.execute("INSERT into contact_us (name, email, phone, blood_group, message, ip,
date) values (?,?,?,?,?,?,?)",(name,email,phone,blood_group,message,ip,date))
            con.commit()
            msg = "Thank you! Your request has been submitted.."

        except:
            con.rollback()
            msg = "We can not add the your blood group to the list"
```

```
        finally:
            return render_template("success.html",msg = msg)
            con.close()
    ##Contact Us end###


##user router start##
@app.route('/user/<int:user_id>')
def show_post(user_id):
    with sqlite3.connect("needyamin.db") as con:
        try:
            con.row_factory = sqlite3.Row
            cur = con.cursor()
            cur.execute("select * from users where id= ?",(user_id,))
            rows = cur.fetchall()
            msg = "success"
        except:
            msg = "wrong method"
        finally:
         return render_template("profile.html", rows = rows);
##user router end##


 ##debugger command###
if __name__ == "__main__":
    #export FLASK_ENV=development
    #source venv/bin/activate
    app.run(debug=True)
```

style.css

```
* {box-sizing: border-box}

/* Set a style for all buttons */
button {
  background-color: #4CAF50;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
  opacity: 0.9;
}

button:hover {
  opacity:1;
}

/* Float cancel and delete buttons and add an equal width */
```

```
.cancelbtn, .deletebtn {
  float: left;
  width: 50%;
}

/* Add a color to the cancel button */
.cancelbtn {
  background-color: #ccc;
  color: black;
}

/* Add a color to the delete button */
.deletebtn {
  background-color: #f44336;
}

/* Add padding and center-align text to the container */
.container {
  padding: 16px;
  text-align: center;
}
```

```
/* The Modal (background) */
.modal {
  display: none; /* Hidden by default */
  position: fixed; /* Stay in place */
  z-index: 1; /* Sit on top */
  left: 0;
  top: 0;
  width: 100%; /* Full width */
  height: 100%; /* Full height */
  overflow: auto; /* Enable scroll if needed */
  background-color: #474e5d;
  padding-top: 50px;
}

/* Modal Content/Box */
.modal-content {
  background-color: #fefefe;
  margin: 5% auto 15% auto; /* 5% from the top, 15% from the bottom and centered */
  border: 1px solid #888;
  width: 80%; /* Could be more or less, depending on screen size */
}
/* Style the horizontal ruler */
hr {
```

```
    border: 1px solid #f1f1f1;
    margin-bottom: 25px;
  }

  /* The Modal Close Button (x) */
  .close {
    position: absolute;
    right: 35px;
    top: 15px;
    font-size: 40px;
    font-weight: bold;
    color: #f1f1f1;
  }
.close:hover,
  .close:focus {
    color: #f44336;
    cursor: pointer;
  }
/* Clear floats */
  .clearfix::after {
    content: "";
    clear: both;
    display: table;
```

```
    }
/* Change styles for cancel button and delete button on extra small screens */
    @media screen and (max-width: 300px) {
      .cancelbtn, .deletebtn {
        width: 100%;
      }
         }
```

GITHUB & PROJECT DEMO LINK

GITHUB LINK

https://github.com/IBM-EPBL/IBM-Project-38882-1660386385

DEMO LINK

https://youtu.be/LfrXMvEEy6M