PLASMA DONOR APPLICATION

IBM PROJECT REPORT

SUBMITTED BY

EMPIRE E	- 962319104037
ANAND RAJ D P	- 962319104018
ANAND BOOJESH R S	- 962319104016
JOOHIB PRAVITHA V T	- 962319104049

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

AMRITA COLLEGE OF ENGINEERING AND TECHNOLOGY

ERACHAKULAM, NAGERCOIL.

ANNA UNIVERSITY::CHENNAI 600 025

CERTIFICATE OF EVALUATION

COLLEGE NAME	: AMRITA COLLEGE OF ENGINEERING AND TECHNOLOGY
BRANCH	: COMPUTER SCIENCE AND ENGINEERING
SEMESTER	: VII
TITLE	: PLASMA DONOR APPLICATION
TEAM ID	: PNT2022TMID51933

STUDENT NAMES	**REGISTRATION NUMBER**	**SUPERVISOR**
EMPIRE E	-962319104037	
ANAND RAJ D P	-962319104018	Mrs. JOTHI LAKSHMI S L
ANAND BOOJESH R S	-962319104016	
JOOHIB PRAVITHA V T	-962319104049	

ACKNOWLEDGEMENT

First and foremost we would like to express our sincere gratitude to our respected Founder Amma,Mata Amritanandamayi Devi and Chairman Mr. K S Ramasubban IAS(Retd) for their blessings and grace in making our project great success.

We would like to place a record with deep sense of gratitude to our Honorable Principal Dr. T Kannan for having given us the opportunity to pursue B.E., course in this prestigious institution.

We would like to express our sincere thanks to our beloved Head of the Department Dr. P M Siva Raja and Project Coordinator Mrs. S L Jothi Lakshmi, for creating a supportive and a model environment for us to work and build up our innovative skills. We also thank our project Mentor Mr. Anant Raj I V for the kind encouragement and moral support, who has been a constant source of inspiration to us.

We wish to express our sincere sense of gratitude to Dr. P M Siva Raja, Head, Department of Computer Science and Engineering and to our Project Guide who enabled us to complete our project successfully.

Contents **Page No**

1.INTRODUCTION

1.1 Project Overview

Objective: -

The main objective is to create an easy-to-use application that can be used by donors to donate their blood to blood banks. Hospitals in need for blood plasma can request blood from blood banks. This application should have a wider reach to appeal to more potential blood donors.

Abstract: -

The application must have a simple accessible interface for donors to register and donate blood to the allocated blood bank. Additionally, they can upload a COVID-19 negative certificate, so that their blood plasma be used for treating COVID-19 patients. The users can create an account that can be used for registration and scheduling appointments at the nearest blood bank for blood donation. Blood banks can look at potential donors and book them for an appointment. The donors who receive these requests can either accept and book an appointment or reject them. Hospitals in need for blood plasma can request blood banks according to their needs. Blood banks can look at requests from hospitals and can either accept or reject them.

1.2 Purpose: -

The main purpose is to connect the donors & the patients by an online application. By using this application, the users can either raise a request for plasma donation or requirement. This system is used if anyone needs a Plasma Donor. It comprises of Admin and User where both can request for a Plasma.

In this system there is something called an active user, which means the user is an Active member of the App and has recovered from Covid 19, only such people are recommended here for Plasma Donation. Both parties can Accept or Reject the request. User has to Upload a Covid Negative report to be able to Donate Plasma.

2.LITERATURE SURVEY

2.1 Existing problem

There has been an increase in demand for blood plasma among hospitals and blood banks as they are additionally used in an experimental treatment for COVID-19. Hence there is a requirement for new infrastructure to facilitate donors, blood banks and hospitals for easier donation and access of blood plasma that could potentially satisfy the excess demand for it to be used for treatment.

2.2 References

- https://ccpp19.org/donors/index.html
- https://www.atlassian.com/agile/project-management
- https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software
- https://www.atlassian.com/agile/tutorials/epics
- https://www.atlassian.com/agile/tutorials/sprints
- https://www.atlassian.com/agile/project-management/estimation
- https://www.atlassian.com/agile/tutorials/burndown-charts

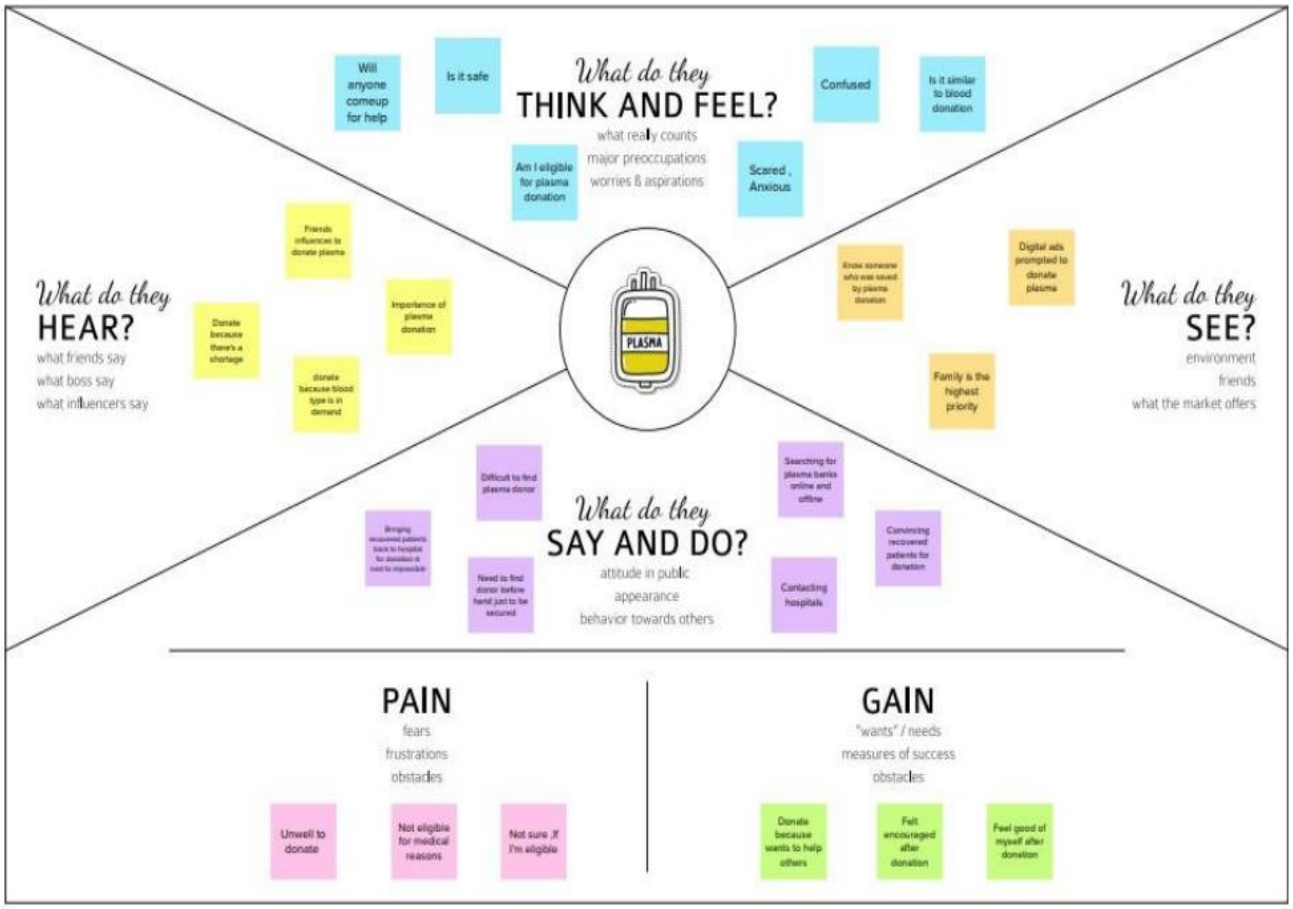
2.3 Problem Statement Definition

There has been an increase in demand for blood plasma among hospitals and blood banks. Convalescent plasma therapy uses blood from people who've recovered from an illness to help others recover. This is an experimental form of therapy which is also used to treat COVID-19. Blood donated by people who've recovered from COVID-19 has antibodies to the virus that causes it. The donated blood is processed to remove blood cells, leaving behind liquid (plasma) and antibodies. These can be given to people with COVID19 to boost their ability to fight the virus. As there is no effective anti-viral treatment for COVID-19 there is a higher prevalence of other forms of therapy such as convalescent plasma therapy which has increased the demand for blood plasma.

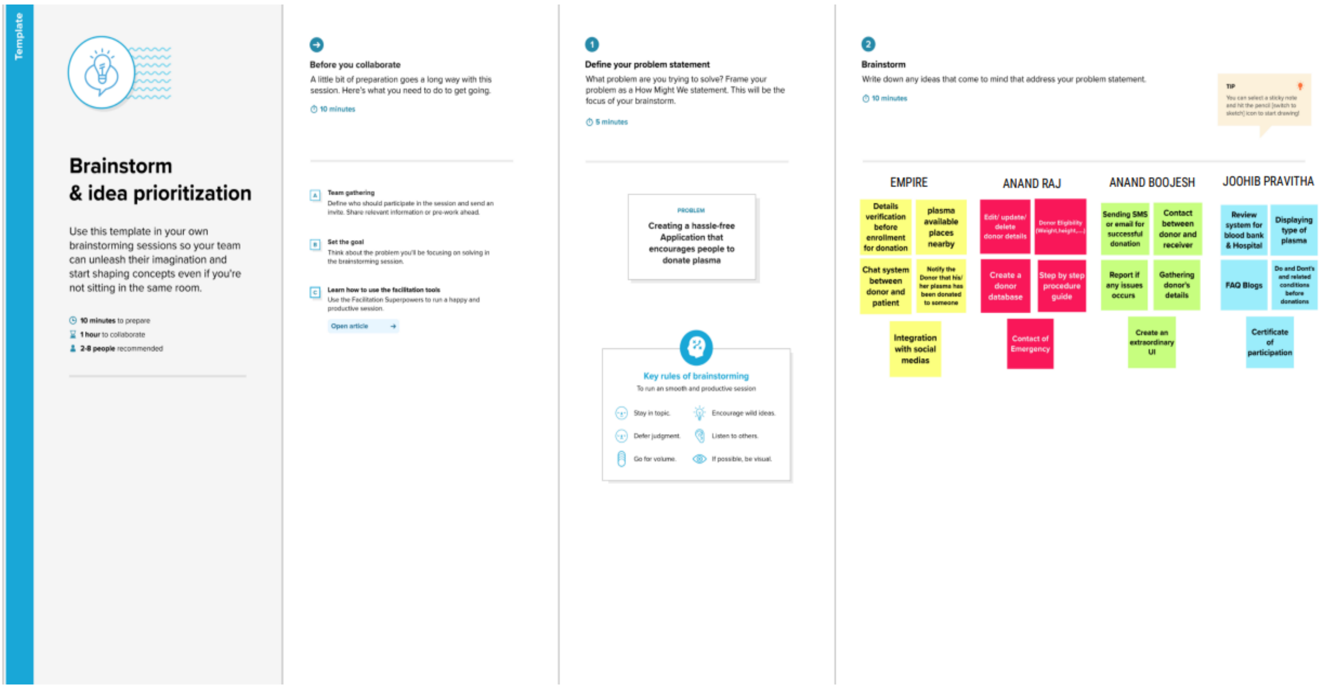
3.IDEATION & PROPOSED SOLUTION

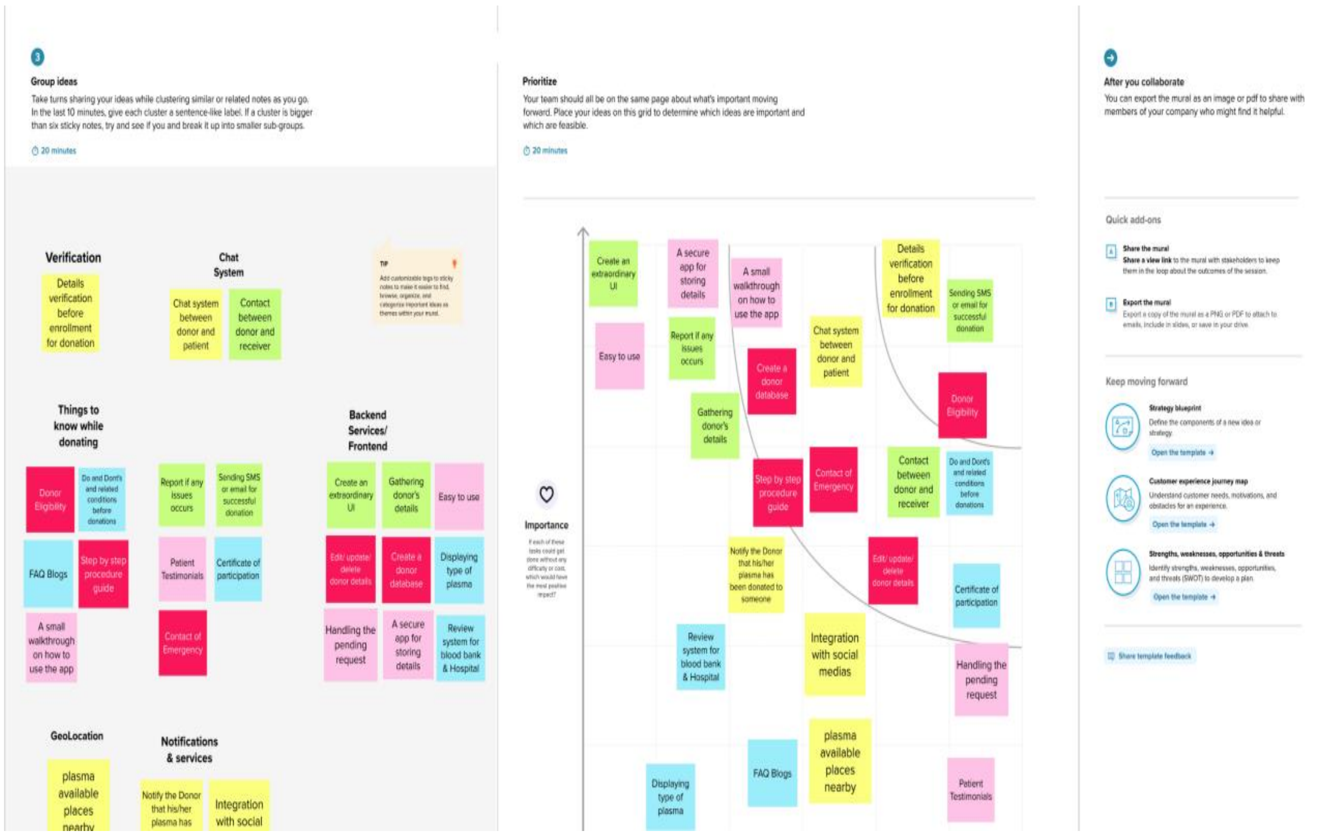
3.1 Empathy Map Canvas

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


What do they
THINK AND FEEL?
what realy counts
major preoccupations
worries & aspirations
Will anyone comeup for help
Is it safe
Am I eligible for plasma donation
Confused
Is it similar to blood donation
Scared , Anxious
What do they
HEAR?
what friends say
what boss say
what influencers say
Friends influences to donate plasma
Donate because there's a shortage
Importance of plasma donation
donate because blood type is in demand
PLASMA
What do they
SEE?
environment
friends
what the market offers
Know someone who was saved by plasma donation
Digital ads prompted to donate plasma
Family is the highest priority
What do they
SAY AND DO?
attitude in public
appearance
behavior towards others
Difficult to find plasma donor
Bringing recovered patients back to hospital for donation is next to impossible
Need to find donor before hand just to be secured
Searching for plasma banks online and offline
Convincing recovered patients for donation
Contacting hospitals
PAIN
fears
frustrations
obstacles
Unwell to donate
Not eligible for medical reasons
Not sure ,If I'm eligible
GAIN
"wants" / needs
measures of success
obstacles
Donate because wants to help others
Felt encouraged after donation
Feel good of myself after donation

3.2 Ideation & Brainstorming


Template
Brainstorm & idea prioritization
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.
10 minutes to prepare
1 hour to collaborate
2-8 people recommended
Before you collaborate
Define your problem statement
PROBLEM
Creating a hassle-free Application that encourages people to donate plasma
Key rules of brainstorming
Brainstorm
Write down any ideas that come to mind that address your problem statement.
10 minutes
EMPIRE
Details verification before enrollment for donation
plasma available places nearby
Chat system between donor and patient
Notify the Donor that his/ her plasma has been donated to someone
Integration with social medias
ANAND RAJ
Edit/ update/ delete donor details
Donor Eligibility (Weight,height,...)
Create a donor database
Step by step procedure guide
Contact of Emergency
ANAND BOOJESH
Sending SMS or email for successful donation
Contact between donor and receiver
Report if any issues occurs
Gathering donor's details
Create an extraordinary UI
JOOHIB PRAVITHA
Review system for blood bank & Hospital
Displaying type of plasma
FAQ Blogs
Do and Dont's and related conditions before donations
Certificate of participation



3.3 Proposed Solution

Solution description:

This proposed system aims at connecting the donors & the patients by an online application. By using this application, the users can either raise a request for plasma donation or requirements. The basic solution is to create a centralized system to keep a track on the upcoming as well as past Plasma Donation Events.

Novelty:

A User Interface is simple for users to understand. We can use the application anywhere anytime. The user immediately needs the plasma for their treatment but the plasma is not available in nearby hospitals, then user can use this application to raise request and directly contact the donor, request them to donate the plasma. Today many of them have mobile phones they can install this application and use it to save the lives of people.

Customer satisfaction:

Effect of donor motivation on donor satisfaction and loyalty are variable due to the influence of common donorship attitudes prevailing in donor population, impact of social marketing programs, focused on promotion of donor commitment and deliberate donorship. Thus, we have predicted that effect of donor motivation on donor relationship satisfaction and loyalty change.

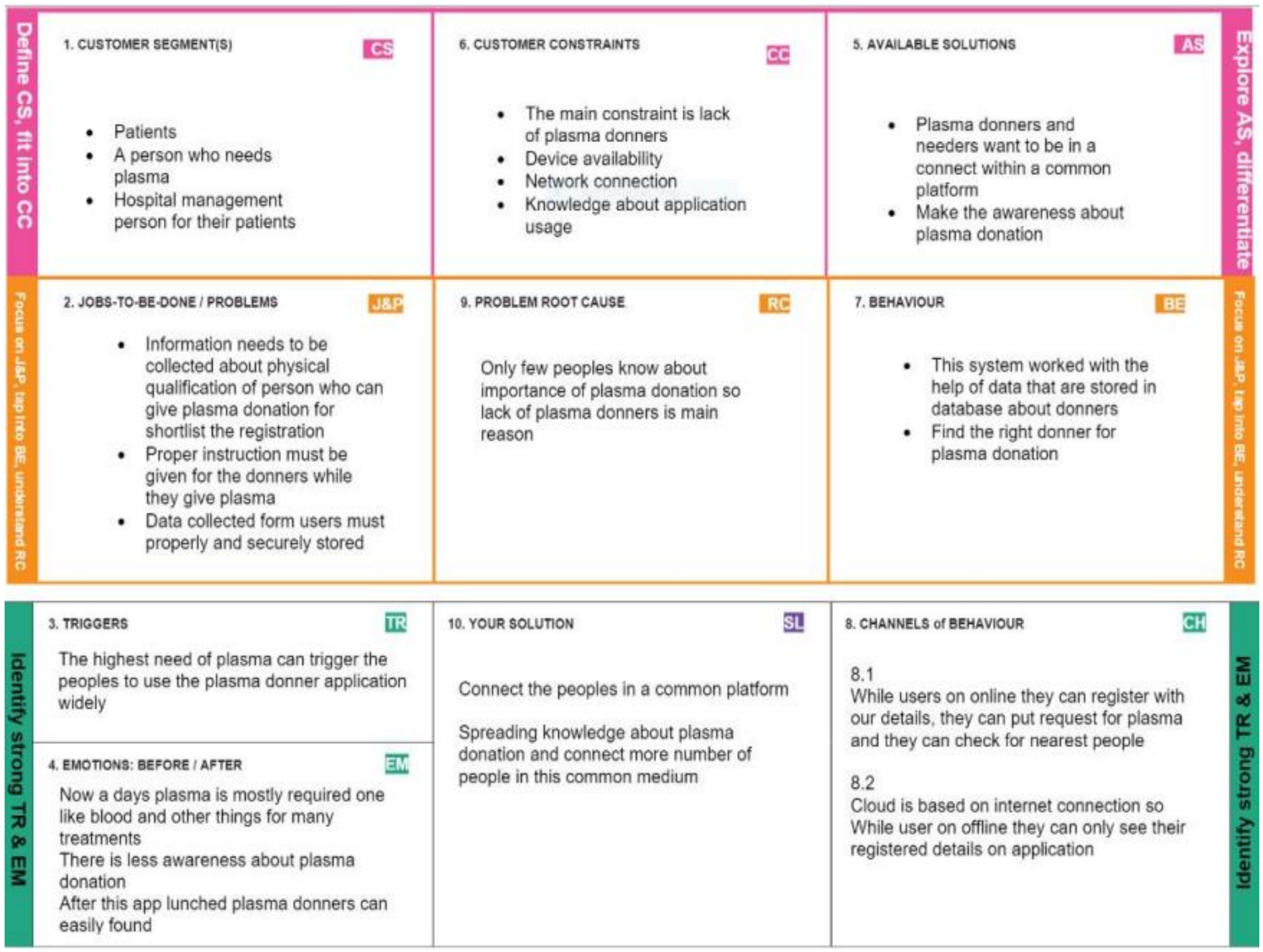
Business model:

This application is accessible by everyone. It is free. Because of the trouble in finding givers who match a specific blood bunch, this application empowers clients to enlist individuals who wish to give plasma and keep their data in a data set. Nowadays the need for plasma increases. Anyone with basic knowledge can access this app.

Scalability of the solution:

This application helps users to find plasma donors by sitting in home itself instead of searching donors everywhere. When there is a emergency then plasma request to send to everyone. Once the donor is ready to donate receiver is notified about donation. Receiver can contact the donor. With this app donor can know the eligibility to donate and making it easier to locate suitable donor at right time.

3.4 Problem Solution fit


Define CS, fit into CC
Explore AS, differentiate
Focus on J&P, tap into BE, understand RC
Focus on J&P, tap into BE, understand RC
Identify strong TR & EM
Identify strong TR & EM

1. CUSTOMER SEGMENT(S) CS

- Patients
- A person who needs plasma
- Hospital management person for their patients

6. CUSTOMER CONSTRAINTS CC

- The main constraint is lack of plasma donners
- Device availability
- Network connection
- Knowledge about application usage

5. AVAILABLE SOLUTIONS AS

- Plasma donners and needers want to be in a connect within a common platform
- Make the awareness about plasma donation

2. JOBS-TO-BE-DONE / PROBLEMS J&P

- Information needs to be collected about physical qualification of person who can give plasma donation for shortlist the registration
- Proper instruction must be given for the donners while they give plasma
- Data collected form users must properly and securely stored

9. PROBLEM ROOT CAUSE RC

Only few peoples know about importance of plasma donation so lack of plasma donners is main reason

7. BEHAVIOUR BE

- This system worked with the help of data that are stored in database about donners
- Find the right donner for plasma donation

3. TRIGGERS TR

The highest need of plasma can trigger the peoples to use the plasma donner application widely

4. EMOTIONS: BEFORE / AFTER EM

Now a days plasma is mostly required one like blood and other things for many treatments
There is less awareness about plasma donation
After this app lunched plasma donners can easily found

10. YOUR SOLUTION SL

Connect the peoples in a common platform

Spreading knowledge about plasma donation and connect more number of people in this common medium

8. CHANNELS of BEHAVIOUR CH

8.1
While users on online they can register with our details, they can put request for plasma and they can check for nearest people

8.2
Cloud is based on internet connection so While user on offline they can only see their registered details on application

4.REQUIREMENT ANALYSIS

4.1 Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	**Functional Requirement (Epic)**	**Sub Requirement (Story/Sub-Task)**
FR-1	User Registration	Registration through website
FR-2	User Confirmation	Confirmation via mail
FR-3	User Login	Login using Registered mail id
FR-4	Sent Request	If Plasma is required, the receiver will contact the donor.
FR-5	Contact Donor	Contact the donor directly if a phone number is given.
FR-6	View Donation Camps	View the list of donation camps happening nearby.

4.2 Non-Functional requirements

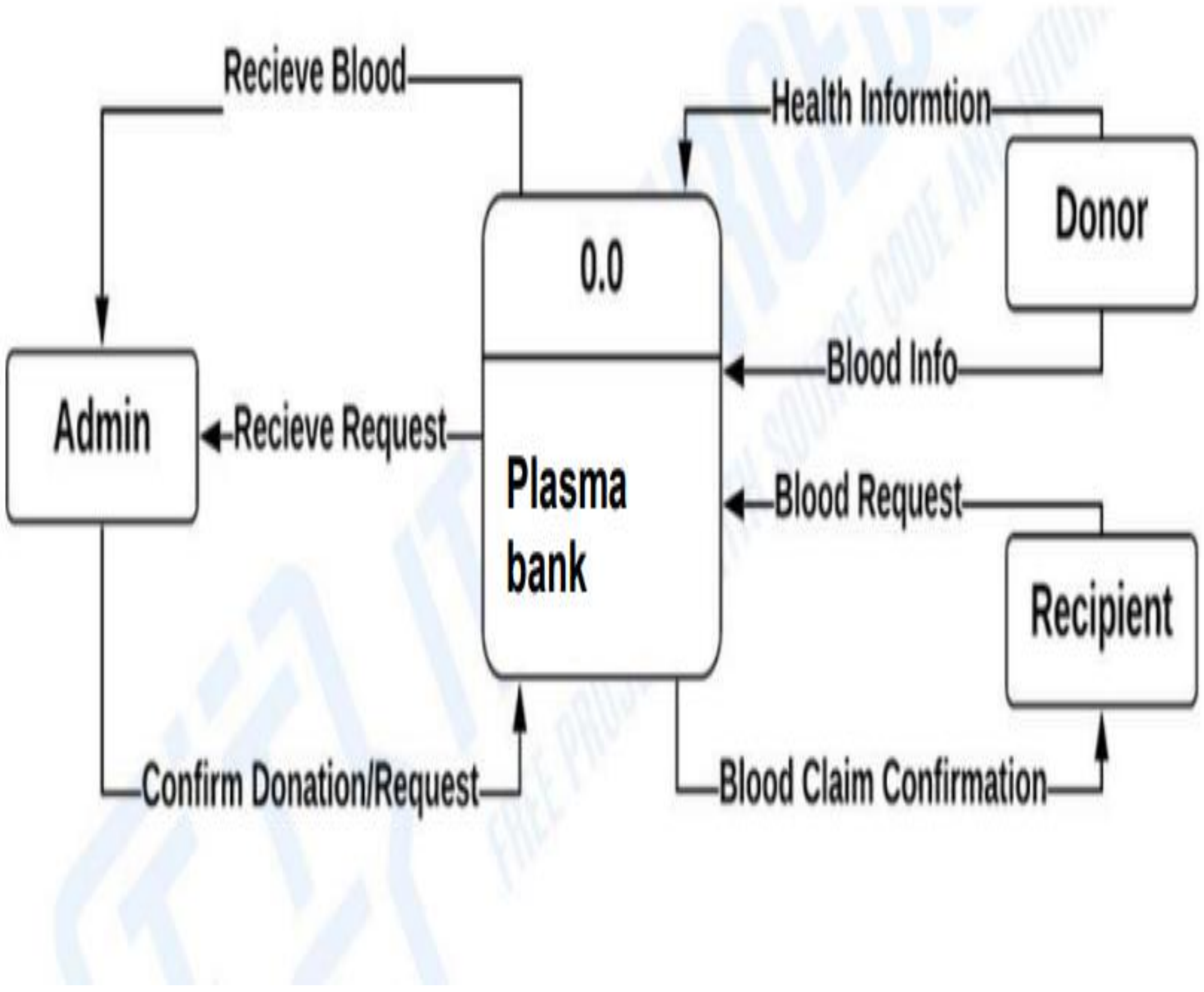
Following are the non-functional requirements of the proposed solution.

FR No.	**Non-Functional Requirement**	**Description**
NFR-1	Usability	The user interface of the plasma donor system must be well-designed and welcoming.
NFR-2	Security	Data storage is required by security systems, just like it is by many other applications. Databases are able to keep all the donor information that is viewed by applications. It must be secured with email Id and password.
NFR-3	Reliability	The system has the ability to work all the times without failures apart from network failure. A donor can have the faith on the system. The authorities will keeps the privacy of all donors in a proper manner
NFR-4	Performance	The Plasma donor System must perform well in different scenarios. The system is interactive and delays involved are less.
NFR-5	Availability	The system, including the online components, should be available 24/7.
NFR-6	Scalability	The system offers the proper resources for issue solutions and is designed to protect sensitive information during all phases of operation.

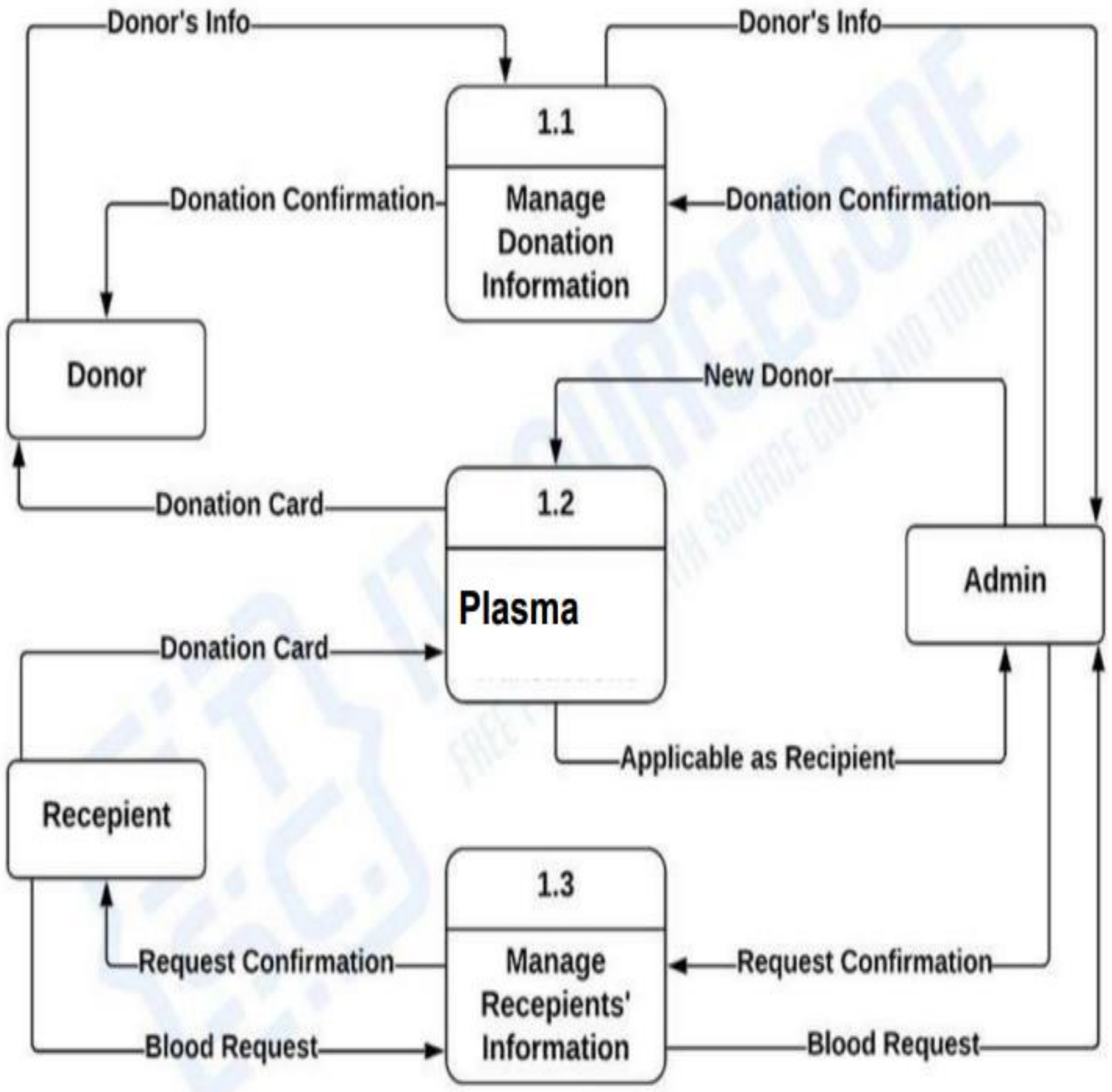
5.PROJECT DESIGN

5.1 Data Flow Diagrams

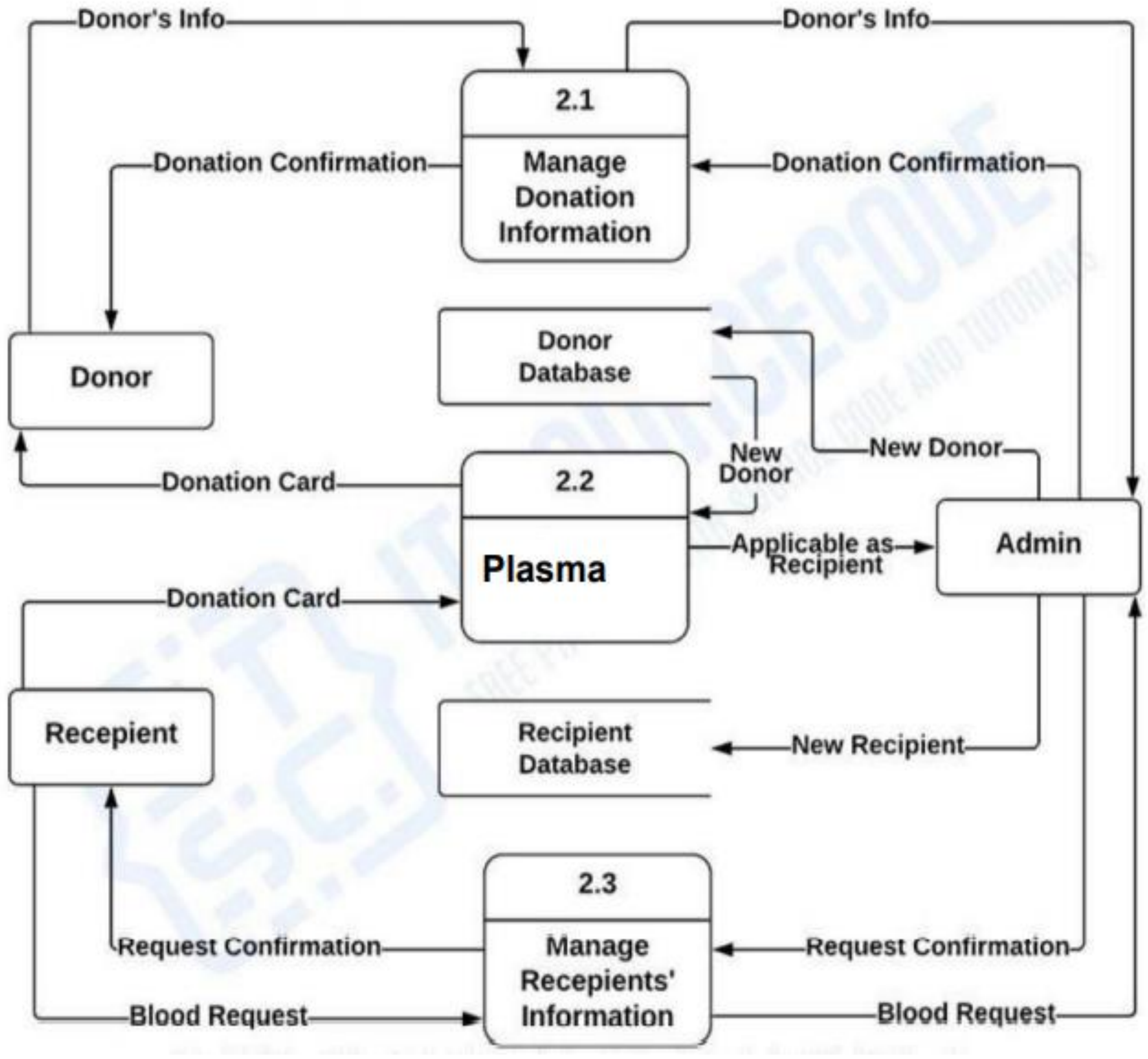
Data flow diagram level 0 :-



Data flow diagram level 1 :-

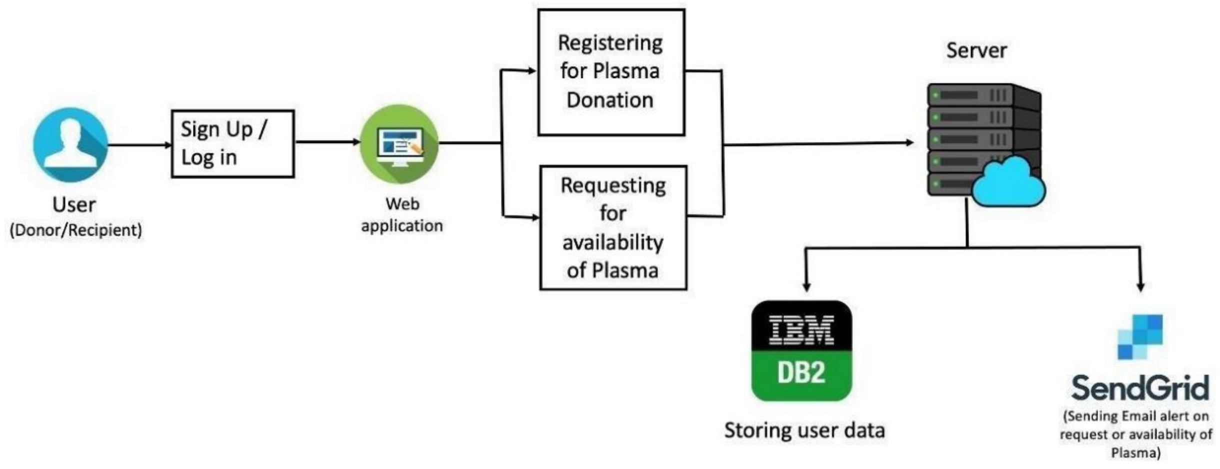


Data flow diagram level 2 :-

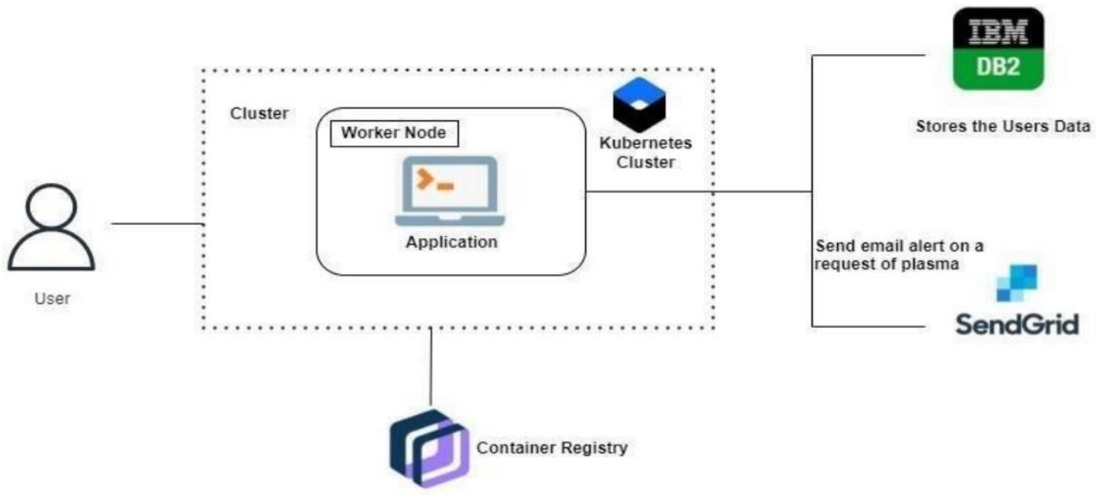


5.2 Solution & Technical Architecture

Solution Architecture :-


User
(Donor/Recipient)
Sign Up / Log in
Web application
Registering for Plasma Donation
Requesting for availability of Plasma
Server
IBM DB2
Storing user data
SendGrid
(Sending Email alert on request or availability of Plasma)

Technical Architecture :-



5.3 User Stories

The below template is used to list all the user stories for the product.

User type	Functional Requirement (Epic)	User Story Number	User story / Task	Acceptance criteria	Priority	Release
Donor	App Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account/ dashboard	High	Sprint-1
	Login	USN-2	As a user, I can log into the application by entering email & password	I can receive confirmation email & click confirm	High	Sprint-2
	Register for Donate	USN-3	As a user, I can log into the application and find the current bank to donate plasma and confirm my booking	I can register & access the dashboard with Facebook Login	Medium	Sprint-3
Patient/Doctor	Find the Bank	USN-4	As a patient, I can directly access the application and find the plasma available bank	I can access my account / dashboard	High	Sprint-1,2
	Request for Plasma	USN-5	As a user, I can enter into the application and find the current bank and request for plasma and state the emergency	can register & access the dashboard with Facebook Login	Medium	Sprint-3
Administrator	Maintain the applications	USN-6	As Administrator I can log into the application by entering email & password and maintaining details for users	I can access my account / dashboard	High	Sprint-3
	Connect the Bank with users	USN-7	As Administrator, i can hold the good communication between bank and user	I can access my account / dashboard	Low	Sprint-4
	Maintain Database	USN-8	As Administrator I can hold the exact details of donor and patient and also bank for requesting and available of plasma	I can access my account / dashboard	Medium	Sprint-4
Plasma Bank	Connect the Bank with users	USN-7	As Bank, I can hold the good communication between Administrator and user	I can access my account / dashboard	Medium	Sprint-3
	Maintain Database	USN-8	As Bank I can hold the exact details of donor and patient and also bank for requesting and available of plasma	I can access my account / dashboard	High	Sprint-4
Bot	Help the user my bot message in application	USN-9	As AI bot, I can hold the good communication between bank and user also help the user	I can access my account / dashboard	Medium	Sprint-4

6.PROJECT PLANNING & SCHEDULING

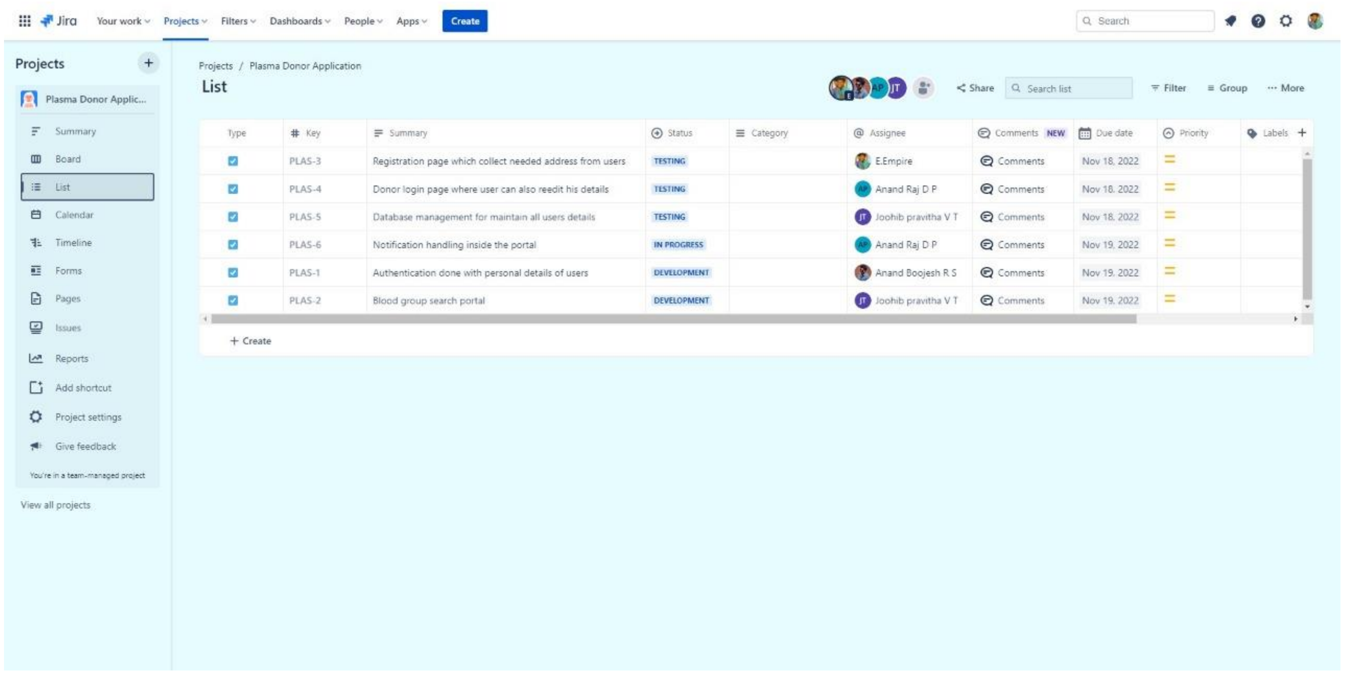
6.1 Sprint Planning & Estimation

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

6.2 Sprint Delivery Schedule:

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

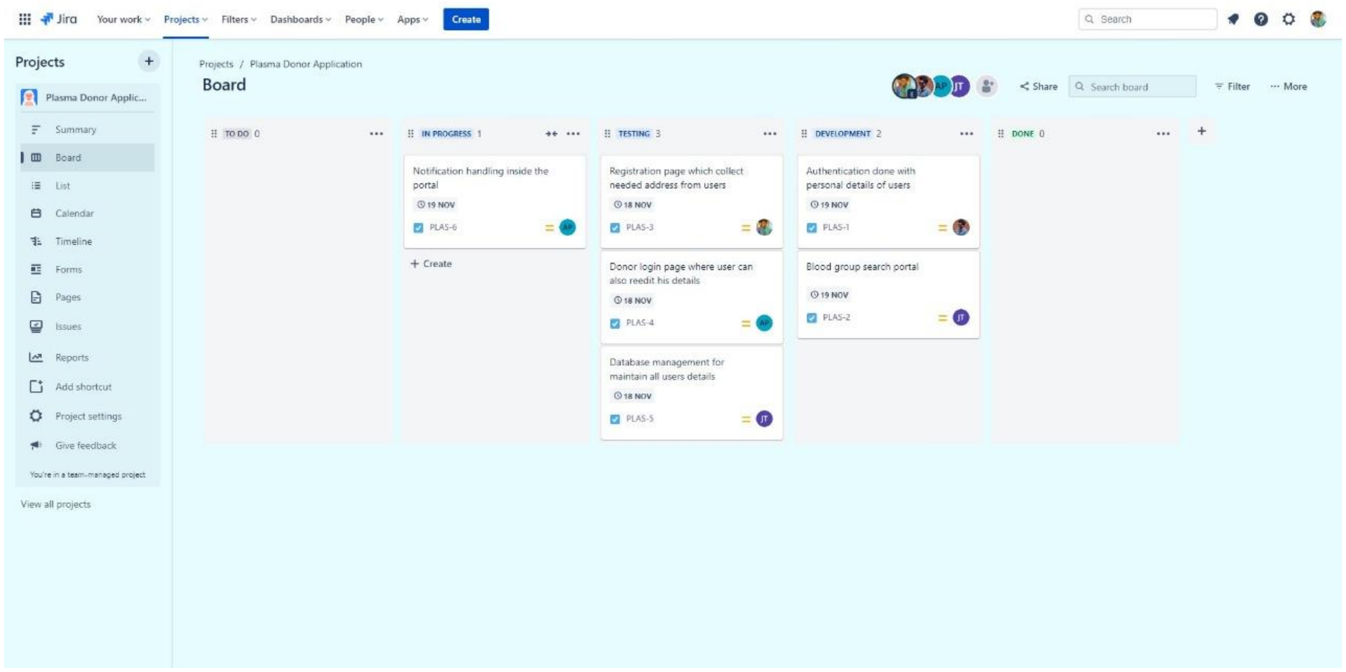
6.3 Reports from JIRA:



Projects / Plasma Donor Application

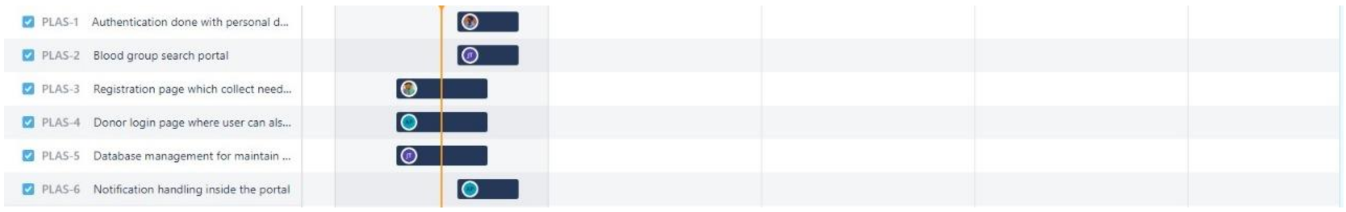
List

Type	Key	Summary	Status	Category	Assignee	Comments NEW	Due date	Priority	Labels
✓	PLAS-3	Registration page which collect needed address from users	TESTING		E.Empire	Comments	Nov 18, 2022	=	
✓	PLAS-4	Donor login page where user can also reedit his details	TESTING		Anand Raj D P	Comments	Nov 18, 2022	=	
✓	PLAS-5	Database management for maintain all users details	TESTING		Joohib pravitha V T	Comments	Nov 18, 2022	=	
✓	PLAS-6	Notification handling inside the portal	IN PROGRESS		Anand Raj D P	Comments	Nov 19, 2022	=	
✓	PLAS-1	Authentication done with personal details of users	DEVELOPMENT		Anand Boojesh R S	Comments	Nov 19, 2022	=	
✓	PLAS-2	Blood group search portal	DEVELOPMENT		Joohib pravitha V T	Comments	Nov 19, 2022	=	



Board

TO DO 0	IN PROGRESS 1	TESTING 3	DEVELOPMENT 2	DONE 0
	Notification handling inside the portal · 19 NOV · PLAS-6	Registration page which collect needed address from users · 18 NOV · PLAS-3	Authentication done with personal details of users · 19 NOV · PLAS-1	
		Donor login page where user can also reedit his details · 18 NOV · PLAS-4	Blood group search portal · 19 NOV · PLAS-2	
		Database management for maintain all users details · 18 NOV · PLAS-5		



PLAS-1	Authentication done with personal d...
PLAS-2	Blood group search portal
PLAS-3	Registration page which collect need...
PLAS-4	Donor login page where user can als...
PLAS-5	Database management for maintain ...
PLAS-6	Notification handling inside the portal

7.CODING & SOLUTIONING

CODING

Login.html:

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

```
<!DOCTYPE html>
<html>
<head>
  <title>Login Form</title>
  <link rel="stylesheet" type="text/css" href="..\static\css\login.css">
  <link href="https://fonts.googleapis.com/css?family=Poppins:600&display=swap"
rel="stylesheet">
  <script src="https://kit.fontawesome.com/a81368914c.js"></script>
  <meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
  <img class="wave" src="..\static\images\wave.png">
  <div class="container">
  <div class="img">
   <div id="png"><a href="/" title="HOME"><img style="width:75px; height:75px ; "
            src="..\static\images\home-page.png"></a></div>
      <img src="https://cdn.iconscout.com/icon/premium/png-256-thumb/blood-
plasma-2500160-2092385.png">
    </div>
    <div class="login-content">
      <form action="{{url_for('login')}}" method="POST">
        <div class="msg">{{ msg }}</div>
        <img src="..\static\images\avatar.svg">
        <h2 class="title">Welcome</h2>
        <div class="input-div one">
          <div class="i">
            <i class="fas fa-user"></i>
          </div>
          <div class="div">
            <input type="text" name="Username" placeholder="username"
class="input" required>
          </div>
        </div>
        <div class="input-div pass">
          <div class="i">
            <i class="fas fa-lock"></i>
          </div>
          <div class="div">

            <input type="password" name="Password" placeholder="password"
class="input" required>
          </div>
        </div>
        <a href="#">Forgot Password?</a>
        <div class="btn">
          <button type="login" class="btn btn-default">Login</button>
        </div><br><br><br><br>
        <div class="app"><b>Don't have an account?</b><a id="app1"
href="\signup">Signup Here !</a></div>
      </form>
    </div>
  </div>
  <script type="text/javascript" src="..\static\js\login.js"></script>
</body>
</html>
```

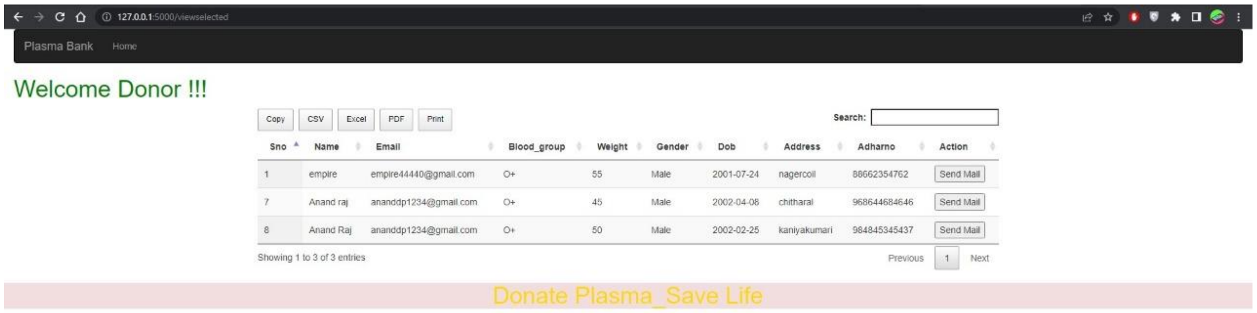
Signup.html :

- A signup page (also known as a registration page) enables users and organizations to independently register and gain access to the system.
- It is common to have multiple signup pages depending on the types of people and organizations you want to register.

```
<html>
<head>
<meta charset="utf-8">
<title>Sign-up</title>
<link href="..\static\css\signup.css" rel="stylesheet">
<script src="https://kit.fontawesome.com/a81368914c.js"></script>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
</head>
<body>
 <!--container---------------------->
<div class="container" >
 <!--sign-up-box-container--->
<div class="sign-up">

    <div id="png"><a href="/" title="HOME"><img style="width:55px; height:55px ;
" src="..\static\images\home-page.png"></a></div>
 <!--heading-->
 <form action="/register1" method="post">

 <h1 class="heading">Hello,Friend</h1><br>
<div
class="para">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&n
bsp;&nbsp;&nbsp;&nbsp;<b>Welcome</b>,Please Fill in the blanks for sign
up</div><br>
 <!--name-box-->
 <div class="text">
 <img height="20px" src="..\static\images\user.png" />
 <input placeholder="Name" type="text" name="username"/>
 </div>
 <!--Email-box-->
 <div class="text">
 <img height="12px" src="..\static\images/email.png" />
 <input placeholder=" Example@gmail.com" type="email" name="email" />
 </div>
 <!--Password-box-->
 <div class="text">
 <img height="20px" src="..\static\images\password.png" />
 <input placeholder=" Password" type="password" name="password"/>
 </div>

<!--trems-->
```

```
<div class="trems">
    <input class="check" type="checkbox" required/>
 <p class="conditions">I read and agree to <a href="#">Terms &amp;
Conditions</a></p>
 </div>
<!--button-->
<div class="toop">
<button type="submit" class="btn btn-primary" >CREATE ACCOUNT</button> </div><br>

</form>
<!--sign-in-->
<div
class="t">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp
;&nbsp;<p class="conditions" id="p3">Already have an account ?&nbsp;&nbsp;<a
href="/login">Login</a></p> </div></div>
 </div>
<!--text-container-->
<div class="text-container">

 <h1 style="color: #2d2c2c;font-family:cursive;">Glad to see you</h1>

 <div class="diag"><img class="fig1" width="70%" height="100%"
src="..\static\images\signup2.png"></div>


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

7.1 Feature 1



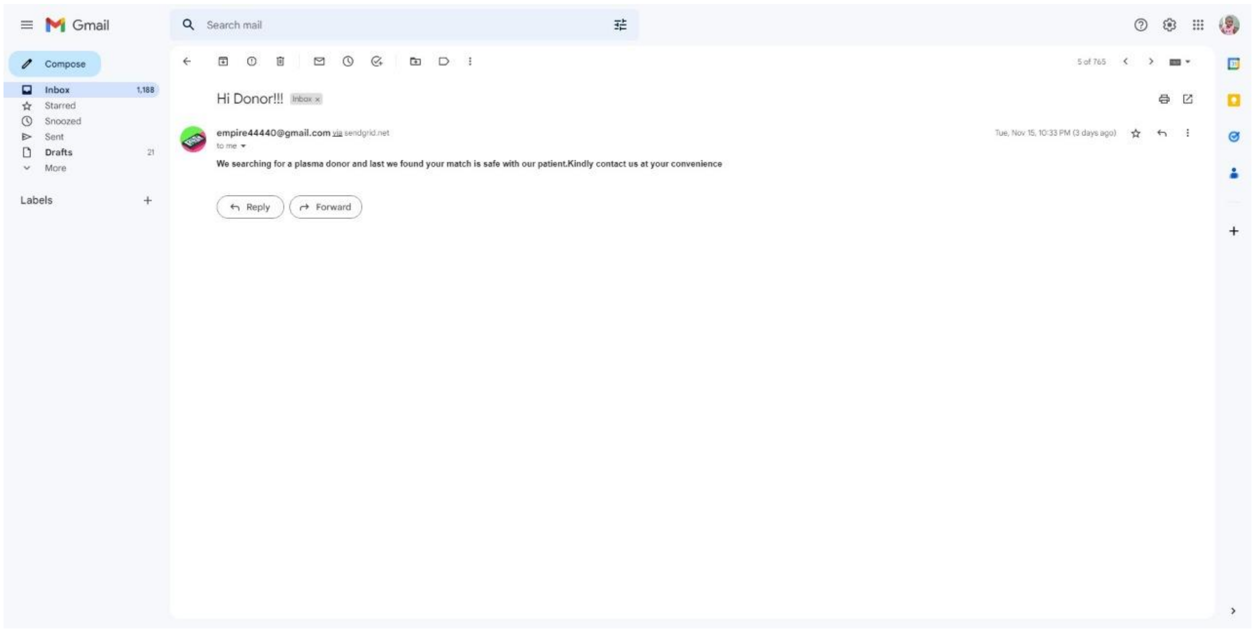
Welcome Donor !!!

Sno	Name	Email	Blood_group	Weight	Gender	Dob	Address	Adharno	Action
1	empire	empire44440@gmail.com	O+	55	Male	2001-07-24	nagercoil	88662354762	Send Mail
7	Anand raj	ananddp1234@gmail.com	O+	45	Male	2002-04-08	chitharal	968644684646	Send Mail
8	Anand Raj	ananddp1234@gmail.com	O+	50	Male	2002-02-25	kaniyakumari	984845345437	Send Mail

Showing 1 to 3 of 3 entries

Donate Plasma_Save Life

7.2 Feature 2



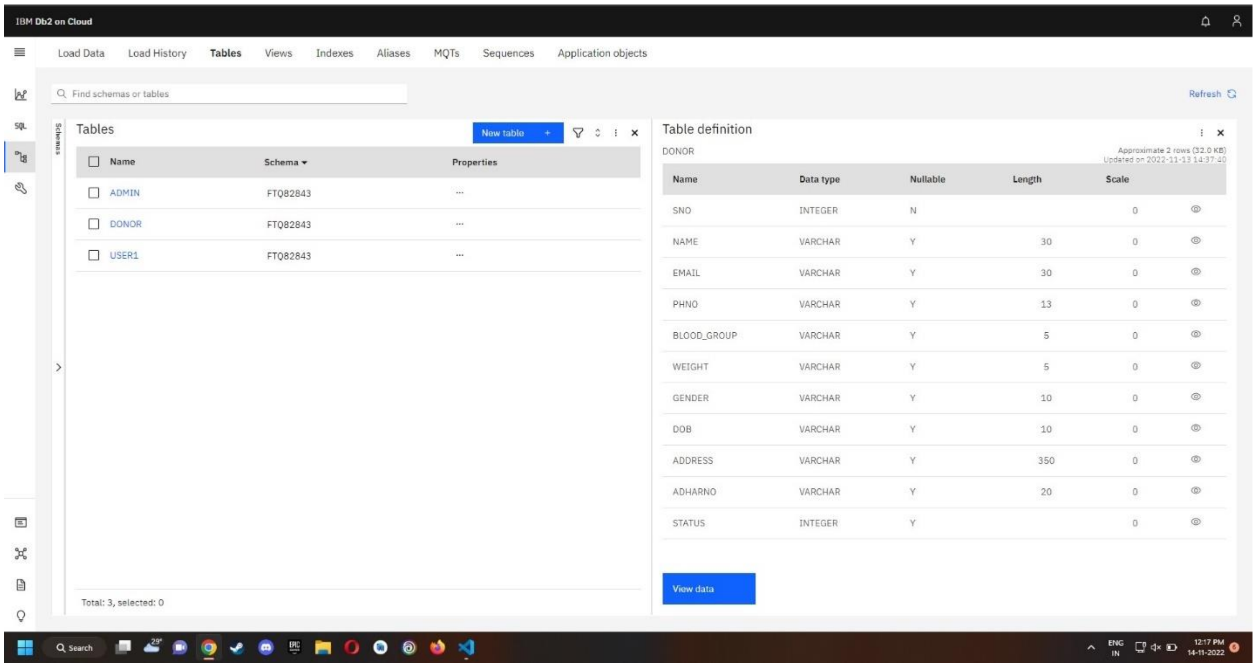
Hi Donor!!!

empire44440@gmail.com via sendgrid.net
to me
Tue, Nov 15, 10:33 PM (3 days ago)

We searching for a plasma donor and last we found your match is safe with our patient.Kindly contact us at your convenience

7.3 DATABASE SCHEMA

A database schema defines how data is organized within a relational database; this is inclusive of logical constraints such as, table names, fields, data types, and the relationships between these entities. Schemas commonly use visual representations to communicate the architecture of the database, becoming the foundation for an organization’s data management discipline. This process of database schema design is also known as data modeling.These data models serve a variety of roles, such as database users, database administrators, and programmers. A database schema is considered the “blueprint” of a database which describes how the data may relate to other tables or other data models. However, the schema does not actually contain data.



Tables

Name	Schema	Properties
ADMIN	FTQ82843	...
DONOR	FTQ82843	...
USER1	FTQ82843	...

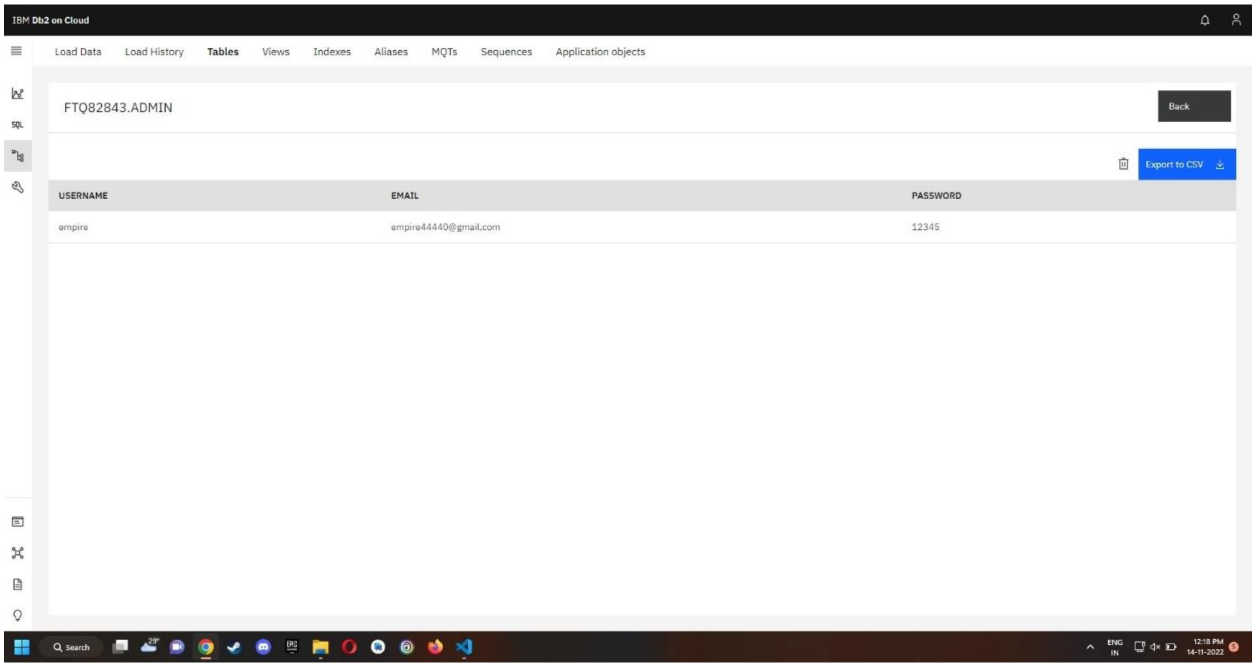
Total: 3, selected: 0

Table definition

DONOR

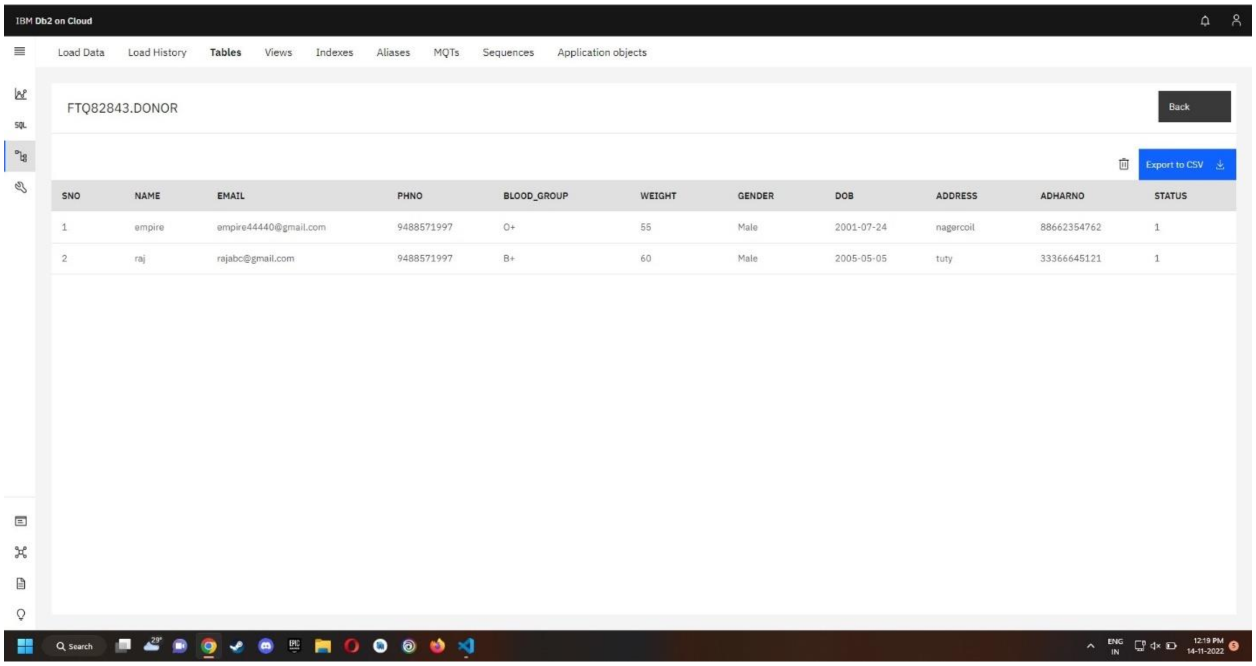
Approximate 2 rows (32.0 KB)
Updated on 2022-11-13 14:37:40

Name	Data type	Nullable	Length	Scale
SNO	INTEGER	N		0
NAME	VARCHAR	Y	30	0
EMAIL	VARCHAR	Y	30	0
PHNO	VARCHAR	Y	13	0
BLOOD_GROUP	VARCHAR	Y	5	0
WEIGHT	VARCHAR	Y	5	0
GENDER	VARCHAR	Y	10	0
DOB	VARCHAR	Y	10	0
ADDRESS	VARCHAR	Y	350	0
ADHARNO	VARCHAR	Y	20	0
STATUS	INTEGER	Y		0



FTQ82843.ADMIN

USERNAME	EMAIL	PASSWORD
empire	empire44440@gmail.com	12345

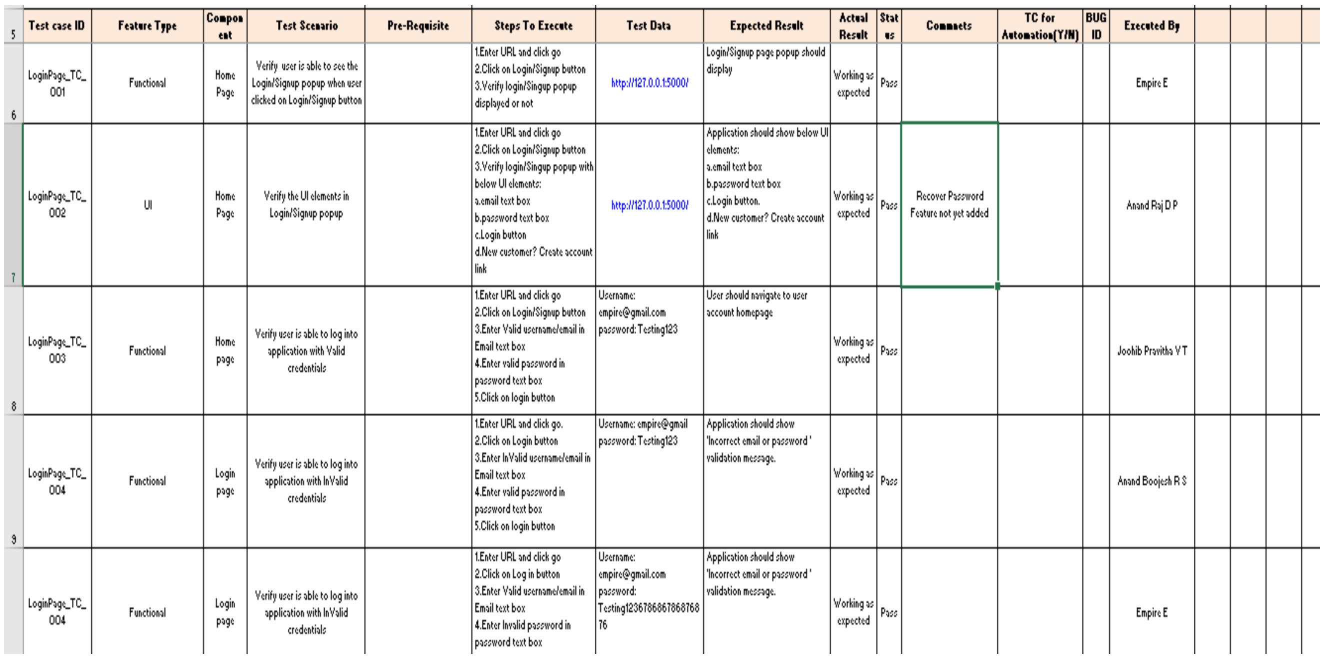


FTQ82843.DONOR

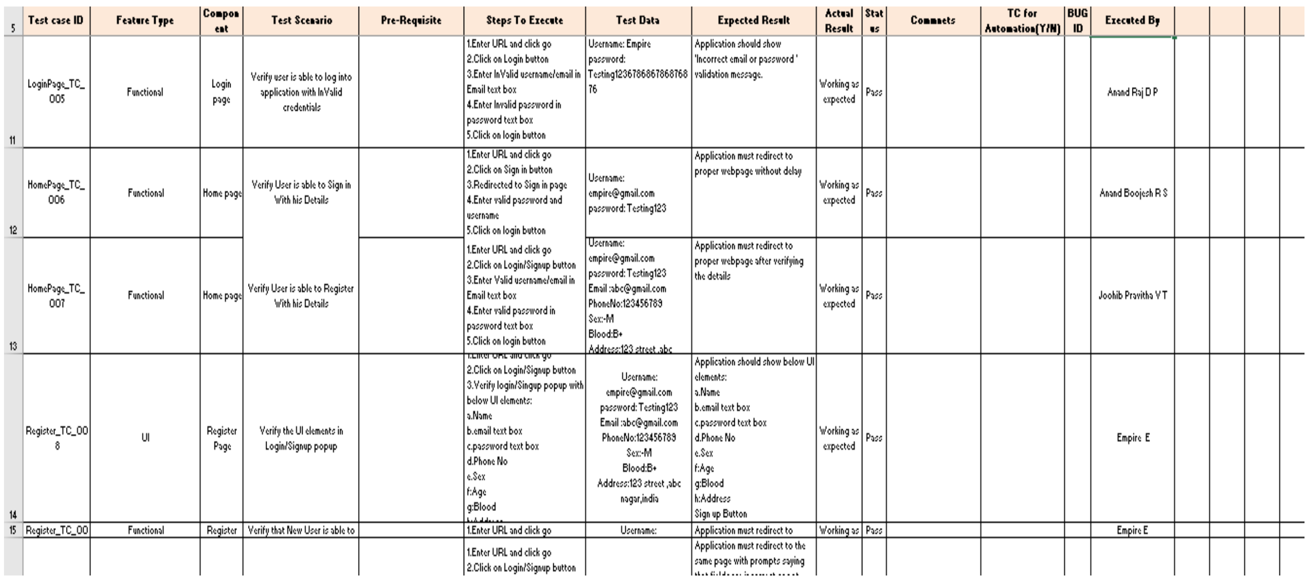
SNO	NAME	EMAIL	PHNO	BLOOD_GROUP	WEIGHT	GENDER	DOB	ADDRESS	ADHARNO	STATUS
1	empire	empire44440@gmail.com	9488571997	O+	55	Male	2001-07-24	nagercoil	88662354762	1
2	raj	rajabc@gmail.com	9488571997	B+	60	Male	2005-05-05	tuty	33366645121	1

8.TESTING

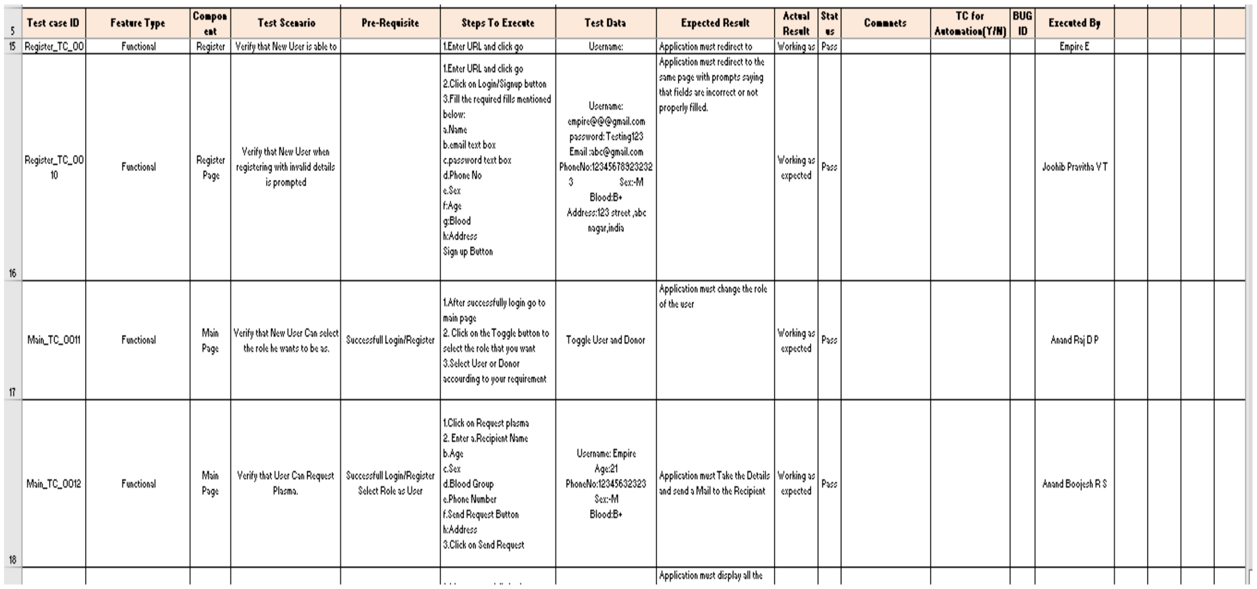
8.1 Test Cases:



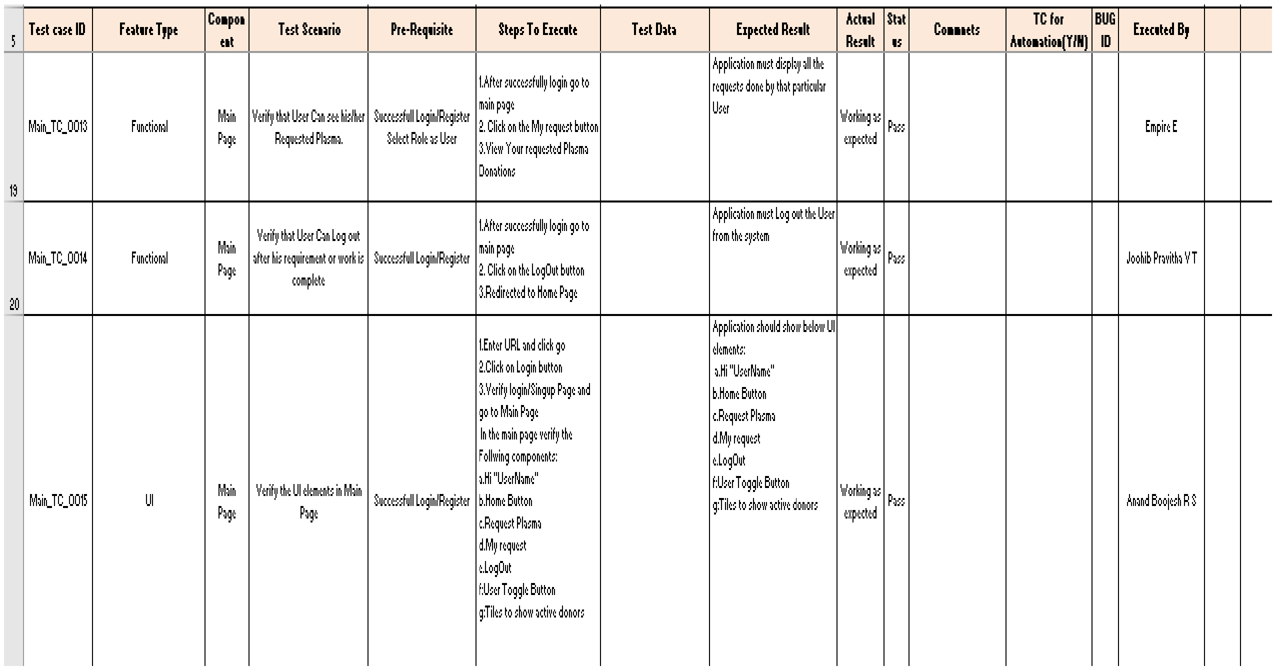
	Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
6	LoginPage_TC_001	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on Login/Signup button		1.Enter URL and click go 2.Click on Login/Signup button 3.Verify login/Singup popup displayed or not	http://127.0.0.1:5000/	Login/Signup page popup should display	Working as expected	Pass				Empire E
7	LoginPage_TC_002	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on Login/Signup button 3.Verify login/Singup popup with below UI elements: a.email text box b.password text box c.Login button d.New customer? Create account link	http://127.0.0.1:5000/	Application should show below UI elements: a.email text box b.password text box c.Login button. d.New customer? Create account link	Working as expected	Pass	Recover Password Feature not yet added			Anand Raj D P
8	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/Signup button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: empire@gmail.com password: Testing123	User should navigate to user account homepage	Working as expected	Pass				Joohib Pravitha V T
9	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 InValid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: empire@gmail password: Testing123	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				Anand Boojesh R S
	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 Log in button 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text box	Username: empire@gmail.com password: Testing123678686786876876	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				Empire E



	Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
11	LoginPage_TC_005	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 InValid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: Empire password: Testing123678686786876876	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				Anand Raj D P
12	HomePage_TC_006	Functional	Home page	Verify User is able to Sign in With his Details		1.Enter URL and click go 2.Click on Sign in button 3.Redirected to Sign in page 4.Enter valid password and username 5.Click on login button	Username: empire@gmail.com password: Testing123	Application must redirect to proper webpage without delay	Working as expected	Pass				Anand Boojesh R S
13	HomePage_TC_007	Functional	Home page	Verify User is able to Register With his Details		1.Enter URL and click go 2.Click on Login/Signup button 3.Enter Valid username/email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: empire@gmail.com password: Testing123 Email :abc@gmail.com PhoneNo:123456789 Sex:-M Blood:B+ Address:123 street ,abc	Application must redirect to proper webpage after verifying the details	Working as expected	Pass				Joohib Pravitha V T
14	Register_TC_008	UI	Register Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on Login/Signup button 3.Verify login/Singup popup with below UI elements: a.Name b.email text box c.password text box d.Phone No e.Sex f:Age g:Blood	Username: empire@gmail.com password: Testing123 Email :abc@gmail.com PhoneNo:123456789 Sex:-M Blood:B+ Address:123 street ,abc nagar,india	Application should show below UI elements: a.Name b.email text box c.password text box d.Phone No e.Sex f:Age g:Blood h:Address Sign up Button	Working as expected	Pass				Empire E
15	Register_TC_00	Functional	Register	Verify that New User is able to		1.Enter URL and click go	Username:	Application must redirect to	Working as	Pass				Empire E
						1.Enter URL and click go 2.Click on Login/Signup button		Application must redirect to the same page with prompts saying						



	Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
15	Register_TC_00	Functional	Register	Verify that New User is able to		1.Enter URL and click go	Username:	Application must redirect to	Working as	Pass				Empire E
16	Register_TC_00 10	Functional	Register Page	Verify that New User when registering with invalid details is prompted		1.Enter URL and click go 2.Click on Login/Signup button 3.Fill the required fills mentioned below: a.Name b.email text box c.password text box d.Phone No e.Sex f:Age g:Blood h:Address Sign up Button	Username: empire@@@gmail.com password: Testing123 Email :abc@gmail.com PhoneNo:12345678923232 3 Sex:-M Blood:B+ Address:123 street ,abc nagar,india	Application must redirect to the same page with prompts saying that fields are incorrect or not properly filled.	Working as expected	Pass				Joohib Pravitha V T
17	Main_TC_0011	Functional	Main Page	Verify that New User Can select the role he wants to be as.	Successfull Login/Register	1.After successfully login go to main page 2. Click on the Toggle button to select the role that you want 3.Select User or Donor accourding to your requirement	Toggle User and Donor	Application must change the role of the user	Working as expected	Pass				Anand Raj D P
18	Main_TC_0012	Functional	Main Page	Verify that User Can Request Plasma.	Successfull Login/Register Select Role as User	1.Click on Request plasma 2. Enter a.Recipient Name b.Age c.Sex d.Blood Group e.Phone Number f.Send Request Button h:Address 3.Click on Send Request	Username: Empire Age:21 PhoneNo:12345632323 Sex:-M Blood:B+	Application must Take the Details and send a Mail to the Recipient	Working as expected	Pass				Anand Boojesh R S
								Application must display all the						



	Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
19	Main_TC_0013	Functional	Main Page	Verify that User Can see his/her Requested Plasma.	Successfull Login/Register Select Role as User	1.After successfully login go to main page 2. Click on the My request button 3.View Your requested Plasma Donations		Application must display all the requests done by that particular User	Working as expected	Pass				Empire E
20	Main_TC_0014	Functional	Main Page	Verify that User Can Log out after his requirement or work is complete	Successfull Login/Register	1.After successfully login go to main page 2. Click on the LogOut button 3.Redirected to Home Page		Application must Log out the User from the system	Working as expected	Pass				Joohib Pravitha V T
	Main_TC_0015	UI	Main Page	Verify the UI elements in Main Page	Successfull Login/Register	1.Enter URL and click go 2.Click on Login button 3.Verify login/Singup Page and go to Main Page In the main page verify the Follwing components: a.Hi "UserName" b.Home Button c.Request Plasma d.My request e.LogOut f:User Toggle Button g:Tiles to show active donors		Application should show below UI elements: a.Hi "UserName" b.Home Button c.Request Plasma d.My request e.LogOut f:User Toggle Button g:Tiles to show active donors	Working as expected	Pass				Anand Boojesh R S

8.2 User Acceptance Testing

- Defect Analysis

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

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

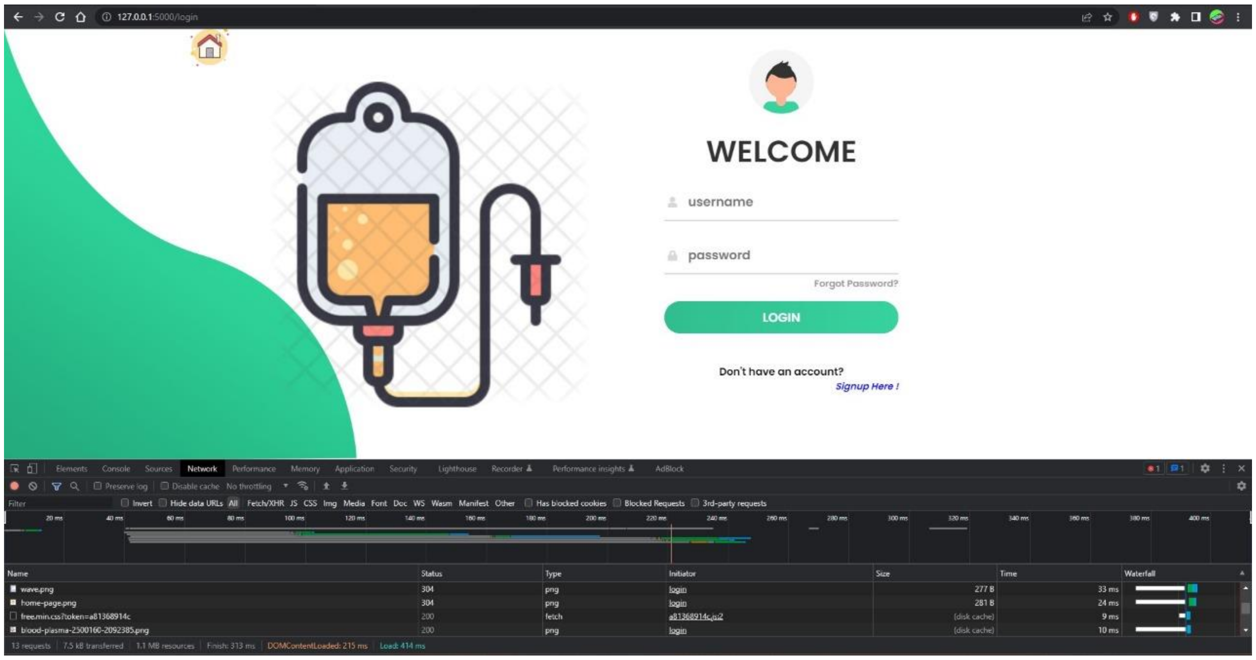
- ## TEST CASE ANALYSIS:

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

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	7	0	0	7
Client Application	51	0	0	51
Security	2	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	9	0	0	9
Final Report Output	4	0	0	4
Version Control	2	0	0	2

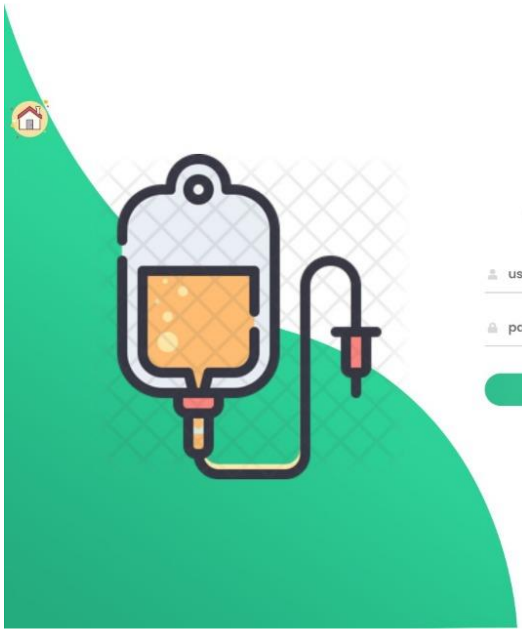
9. RESULTS

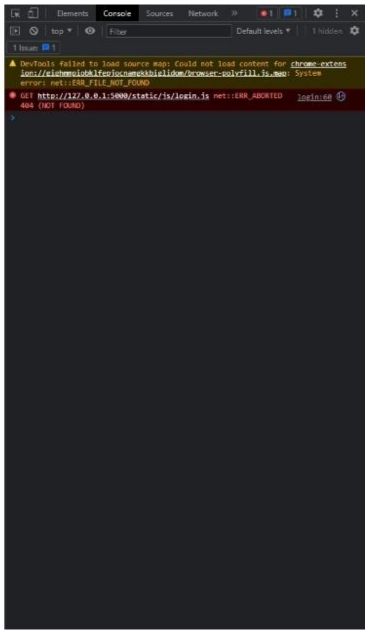
9.1 Performance Metrics


127.0.0.1:5000/login
WELCOME
username
password
Forgot Password?
LOGIN
Don't have an account?
Signup Here !
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights AdBlock
Preserve log Disable cache No throttling
Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies Blocked Requests 3rd-party requests

Name	Status	Type	Initiator	Size	Time
wave.png	304	png	login	277 B	33 ms
home-page.png	304	png	login	281 B	24 ms
free.min.css?token=a81368914c	200	fetch	a81368914c.js:2	(disk cache)	9 ms
blood-plasma-2300160-2092385.png	200	png	login	(disk cache)	10 ms

```
13 requests | 7.5 kB transferred | 1.1 MB resources | Finish: 313 ms | DOMContentLoaded: 215 ms | Load: 414 ms
```





```
DevTools failed to load source map: Could not load content for chrome-extension://eiehmmpiobklfepjocnamgkkbiglidom/browser-polyfill.js.map: System error: net::ERR_FILE_NOT_FOUND
GET http://127.0.0.1:5000/static/js/login.js net::ERR_ABORTED 404 (NOT FOUND)    login:60
```

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES:-

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

Some of the advantages of the application include :

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

DISADVANTAGES:-

Some of the disadvantages of the application include :

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

11. CONCLUSION

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

An Plasma Donor application is an exclusive suite of services for people who are in need of Plasma. It helps you track all the details about the Plasma Donors. The basic solution is to create a centralized system to keep a track on the upcoming as well as past Plasma Donation Events.

12. FUTURE SCOPE

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

- The analysis such as Frequently requested zone or hospital for Plasma, Number of donors, mostly asked Blood Group, Age group of Patients need for plasma etc. can be added as additional features.
- The application can be implemented using Artificial Intelligence and Deep Learning Algorithms.
- NGOs and NCC Units information's can be made available in the application.
- Donors last donated details can be automatically updated in the App.
- Notification to Donors about the nearest Plasma Donation Center.
- Increase efficiency and customer satisfaction with an app aligned to their needs.
- Seamlessly integrate with existing infrastructure.
- Chats: Equip Plasma Donor app with a bot that can understand and answer all user queries and address their needs.

13. APPENDIX

Source Code

```
from flask import *
from flask import Flask, render_template, request, redirect, url_for, session
from twilio.rest import Client
from werkzeug.utils import secure_filename
import ibm_db
import re
import os
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
import csv
app=Flask(__name__)
app.secret_key="don't share"
myconn=ibm_db.connect('DATABASE=bludb;HOSTNAME=fbd88901-ebdb-4a4f-a32e-
9822b9fb237b.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32731;SECURITY=S
SL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=ftq82843;PWD=u4VCEInPvFw43qi
x', '', ''
    )
@app.route("/signup")
def signup():
    return render_template("signup.html")


@app.route('/register1', methods =['GET', 'POST'])
def register1():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']
        query = 'SELECT * FROM admin WHERE username =?;'
        stmt=ibm_db.prepare(myconn,query)
        ibm_db.bind_param(stmt,1,username)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            query = "INSERT INTO ADMIN VALUES (?,?,?)"
            stmt=ibm_db.prepare(myconn,query)
            ibm_db.bind_param(stmt,1,username)
            ibm_db.bind_param(stmt,2,email)
            ibm_db.bind_param(stmt,3,password)
            ibm_db.execute(stmt)
            msg = 'You have successfully registered !'
            return render_template('login.html', msg = msg)

@app.route("/login",methods=['GET','POST'])
def login():
    if request.method=="POST":
        Username=request.form['Username']
        Password=request.form['Password']
        query="select * from admin where Username=? and password=?;"
        stmt=ibm_db.prepare(myconn, query)
        ibm_db.bind_param(stmt, 1, Username)
        ibm_db.bind_param(stmt, 2, Password)
        ibm_db.execute(stmt)
        data=ibm_db.fetch_assoc(stmt)
        if data:
            session['loggedin']=True
            flash("Login Successfully")
            return  render_template('info.html')
        else:
            flash("Incorrect Username or Password")
    return render_template("login.html")
@app.route("/")
@app.route("/bloodbank")
def bloodbank():
        return render_template("bloodbank.html")
@app.route("/home")
def home():

    query="select count(*) from donor where status=1"
    stmt = ibm_db.prepare(myconn, query)
    ibm_db.execute(stmt)
    data = ibm_db.fetch_tuple(stmt)
    return render_template("index.html",data=[data])
@app.route("/register",methods=['GET','POST'])
def register():
    if request.method=="POST":
        name=request.form['name']
        email=request.form['email']
        phno=request.form['phno']
        blood_group=request.form['blood_group']
        weight=request.form['weight']
        gender=request.form['gender']
        dob=request.form['dob']
        address=request.form['address']
        adharno=request.form['adharno']
        status=1
        query="select * from donor where adharno=(?);"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.bind_param(stmt, 1, adharno)
        ibm_db.execute(stmt)
        data = ibm_db.fetch_assoc(stmt)
        if (data)==0:
            query = "INSERT INTO donor
(NAME,EMAIL,PHNO,BLOOD_GROUP,WEIGHT,GENDER,DOB,ADDRESS,ADHARNO,STATUS)
values(?,?,?,?,?,?,?,?,?,?)"
            stmt = ibm_db.prepare(myconn, query)
            ibm_db.bind_param(stmt, 1, name)
            ibm_db.bind_param(stmt, 2, email)
            ibm_db.bind_param(stmt, 3, phno)
            ibm_db.bind_param(stmt, 4, blood_group)
            ibm_db.bind_param(stmt, 5, weight)
            ibm_db.bind_param(stmt, 6, gender)
            ibm_db.bind_param(stmt, 7, dob)
            ibm_db.bind_param(stmt, 8, address)
            ibm_db.bind_param(stmt, 9, adharno)
            ibm_db.bind_param(stmt, 10, status)
            ibm_db.execute(stmt)
            msg = 'You have successfully Logged In!!'
            return redirect(url_for('viewall'))
        else:
            flash("Already Registered")
        return redirect(url_for('register'))
    else:
        return render_template("about.html")
@app.route("/view",methods=['GET','POST'])
def view():
    if not session.get('loggedin'):
        return  render_template("login.html")
    query="select * from donor where status=1"
    stmt = ibm_db.prepare(myconn, query)
    ibm_db.execute(stmt)
    data=[]
    tuple = ibm_db.fetch_tuple(stmt)
    while tuple!=False:
        data.append(tuple)
        tuple=ibm_db.fetch_tuple(stmt)
    return render_template("view.html",data=data)
@app.route("/delete",methods=['GET','POST'])
def delete():
    if not session.get('loggedin'):
        return  render_template("login.html")
    if request.method=="POST":
        id=request.form['delete']
        query="delete from donor where sno=?"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.commit(stmt)
        flash("Deleted Successfully")
        return redirect(url_for('view'))
@app.route("/edit",methods=['GET','POST'])
def edit():
    if not session.get('loggedin'):
        return  render_template("login.html")
    if request.method=="POST":
        id=request.form['edit']
        query="select * from donor where sno=?"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)
        data = ibm_db.fetch_tuple(stmt)
        return render_template("edit.html",data=data)
@app.route("/update",methods=['GET','POST'])
def update():
    if not session.get('loggedin'):
        return  render_template("login.html")
    if request.method=="POST":
        id=request.form['id']
        name=request.form['name']
        email=request.form['email']
        phno=request.form['phno']
        blood_group=request.form['blood_group']
        weight=request.form['weight']
        gender=request.form['gender']
        dob=request.form['dob']
        address=request.form['address']
        adharno=request.form['adharno']
        query = "INSERT INTO USER1 values(?,?,?,?,?,?,?,?,?,?)"
```

```
            stmt = ibm_db.prepare(myconn, query)
            ibm_db.bind_param(stmt, 1, id)
            ibm_db.bind_param(stmt, 2, name)
            ibm_db.bind_param(stmt, 3, email)
            ibm_db.bind_param(stmt, 4, phno)
            ibm_db.bind_param(stmt, 5, blood_group)
            ibm_db.bind_param(stmt, 6, weight)
            ibm_db.bind_param(stmt, 7, gender)
            ibm_db.bind_param(stmt, 8, dob)
            ibm_db.bind_param(stmt, 9, address)
            ibm_db.bind_param(stmt, 10, adharno)
            ibm_db.commit(stmt)
            return redirect(url_for('view'))
@app.route("/view2",methods=['GET','POST'])
def view2():
    query="select distinct blood_group from donor where status=1"
    stmt = ibm_db.prepare(myconn, query)
    ibm_db.execute(stmt)
    data=[]
    tuple = ibm_db.fetch_tuple(stmt)
    while tuple!=False:
        data.append(tuple)
        tuple=ibm_db.fetch_tuple(stmt)
    return render_template("select.html",data=data)
@app.route("/viewselected",methods=['GET','POST'])
def viewselected():
    blood_group=request.form['blood_group']
    query="select * from donor where blood_group= ? and status=1"
    stmt = ibm_db.prepare(myconn, query)
    ibm_db.bind_param(stmt, 1, blood_group)
    ibm_db.execute(stmt)
    data=[]
    tuple = ibm_db.fetch_tuple(stmt)
    while tuple!=False:
        data.append(tuple)
        tuple=ibm_db.fetch_tuple(stmt)
    return render_template("view2.html",data=data)
@app.route("/viewall",methods=['GET','POST'])
def viewall():
    query="select * from donor where status=1"
    stmt = ibm_db.prepare(myconn, query)
    ibm_db.execute(stmt)
    data=[]
    tuple = ibm_db.fetch_tuple(stmt)
    while tuple!=False:
        data.append(tuple)
        tuple=ibm_db.fetch_tuple(stmt)
    return render_template("view2.html",data=data)
@app.route("/")
@app.route("/send",methods=['GET','POST'])
def send():
    if request.method=="POST":
        id=request.form['send']
        query="select email from donor where sno=?"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)
        data = ibm_db.fetch_assoc(stmt)
        print(data)
        message =
Mail(from_email='empire44440@gmail.com',to_emails=data['EMAIL'],subject='Sending
with Twilio SendGrid is Fun',html_content='<strong>and easy to do anywhere, even
with Python</strong>')
        try:
            sg =
SendGridAPIClient('SG.ktA7YoLdR42S9fv1UsluhA.3wrD69UzKSrNPGyFwAwkt2s00X5zIF9iAfZp
tg4ejXU')
            response = sg.send(message)
            print(response.status_code)
            print(response.body)
            print(response.headers)
        except Exception as e:
            print(e)
    return redirect('/viewall')
@app.route("/logout")
def logout():
    session['loggedin']=False
    return render_template("index.html")
@app.route("/hold",methods=['GET','POST'])
def hold():
    if not session.get('loggedin'):
        return  render_template("login.html")
    if request.method=="POST":
        id=request.form['hold']
        query="update donor set  status=0 where sno=?"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute(stmt)
        return redirect(url_for('view'))
@app.route("/activate",methods=['GET','POST'])
```

```
def activate():
    if not session.get('loggedin'):
        return  render_template("login.html")
    if request.method=="POST":
        id=request.form['hold']
        query="update donor set  status=1 where sno=?"
        stmt = ibm_db.prepare(myconn, query)
        ibm_db.bind_param(stmt, 1, id)
        ibm_db.execute3(stmt)
        return redirect(url_for('inactive'))
@app.route("/inactive",methods=['GET','POST'])
def inactive():
    if not session.get('loggedin'):
        return  render_template("login.html")
    query="select * from donor where status=0"
    stmt = ibm_db.prepare(myconn, query)
    ibm_db.execute(stmt)
    data = ibm_db.fetch_tuple(stmt)
    print(data)
    return render_template('inactive.html',data=[data])
if __name__=="__main__":
    app.run(debug=True)
```

GitHub & Project Demo Link

GitHub Link :

https://github.com/IBM-EPBL/IBM-Project-23490-1659884059

Project Demo Link :

https://github.com/IBM-EPBL/IBM-Project-23490-1659884059