

PROJECT REPORT

Date	14.11.2022
Team ID	PNT2022TMID02933
Project Name	Plasma Donor Application
Team Members	Kavin(19EUIT072) Kishore(19EUIT078) Kavinkumar(19EUIT073) Priyadharshan(19EUIT121)

I Introduction

1.1 Project Overview

1.2 Purpose

II Literature Survey

2.1 Existing Problem

2.2 Problem Statement Definition

III Ideation & Proposed Solution

3.1 Empathy Map

3.2 Ideation & Brainstorming

3.3 Proposed Solution

3.4 Problem Solution Fit

IV Requirement Analysis

4.1 Functional Requirements

4.2 Non-Functional Requirements

V Project Design

5.1 Data Flow Diagram

5.2 Solution & Technical Architecture

VI Project Planning and Scheduling

6.1 Sprint planning and Estimation

6.2 Sprint Delivery Schedule

VII Coding and Solution

7.1 Feature I

7.2 Feature II

VIII Advantages

IX Conclusion

X References

1. INTRODUCTION

1.1 Project Overview

The Plasma donor app is to create details about the donor and organizations that are related to donating the blood. Through this application any person who is interested in donating blood can register himself in the same way if any organization wants to register itself with this site that can also register. Moreover if any general consumer wants to request blood online he can also take the help of this site. Admin is the main authority who can do addition" deletion" and modification if required.

1.2 Purpose

The goal of the project is to develop a web application for plasma banks to manage information about their donors and plasma stock. The main objectives of this website development can be defined as follows:

1. To develop a system that provides functions to support donors to view and manage their information conveniently.
2. To maintain records of plasma donors, plasma donation information and plasma stock in a centralized database system.
3. To support searching, matching and requesting for blood convenient for administrators.
4. To provide a function to send an email directly to the donor for their useraccount.

2. LITERATURE SURVEY

2.1 Existing problem

- Risk of mismanagement and of data when the protection is under development.
- No use of web services and remoting.

2.2 Problem Statement Definition

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.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to help 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.



3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

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.

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

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.

- 15 minutes to prepare
- 1 hour to brainstorm
- 10 minutes to debrief

Before you collaborate

1. Identify a problem or goal to work on with this session. Write it down. Read it out loud to get going.

10 minutes

1. Theme gathering

Brainstorm ideas related to the problem and write them down. (How do we solve this problem?)

2. Idea mapping

Group ideas into clusters. Write the ideas on sticky notes and place them on the wall.

3. Select ideas to develop

Choose the ideas you want to develop further. Write them down.

10 minutes

2. Define your problem statement

Write a problem statement that you want to solve. Write it down. Read it out loud to get going. This will be the focus of your brainstorm.

10 minutes

3. Key notes for brainstorming

Brainstorm ideas related to the problem and write them down. (How do we solve this problem?)

4. Idea mapping

Group ideas into clusters. Write the ideas on sticky notes and place them on the wall.

5. Select ideas to develop

Choose the ideas you want to develop further. Write them down.

10 minutes

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.

15 minutes to prepare
1 hour to brainstorm
10 minutes to debrief

Step-2: Brainstorm, Idea Listing and Grouping

Instructions

Write down the ideas that come to mind
when you think about the topic.

Topic:

Tip:

The more ideas you write down, the better your ideas will be.

Project

1. What is the problem?

2. What are the goals?

3. What are the constraints?

4. What are the resources?

5. What are the risks?

6. What are the benefits?

7. What are the challenges?

8. What are the opportunities?

9. What are the risks?

10. What are the benefits?

Brainstorming

1. What is the problem?

2. What are the goals?

3. What are the constraints?

4. What are the resources?

5. What are the risks?

6. What are the benefits?

7. What are the challenges?

8. What are the opportunities?

9. What are the risks?

10. What are the benefits?

Brainstorming

1. What is the problem?

2. What are the goals?

3. What are the constraints?

4. What are the resources?

5. What are the risks?

6. What are the benefits?

7. What are the challenges?

8. What are the opportunities?

9. What are the risks?

10. What are the benefits?

Brainstorming

1. What is the problem?

2. What are the goals?

3. What are the constraints?

4. What are the resources?

5. What are the risks?

6. What are the benefits?

7. What are the challenges?

8. What are the opportunities?

9. What are the risks?

10. What are the benefits?

Project

1. What is the problem?

2. What are the goals?

3. What are the constraints?

4. What are the resources?

5. What are the risks?

6. What are the benefits?

7. What are the challenges?

8. What are the opportunities?

9. What are the risks?

10. What are the benefits?

Brainstorming

1. What is the problem?

2. What are the goals?

3. What are the constraints?

4. What are the resources?

5. What are the risks?

6. What are the benefits?

7. What are the challenges?

8. What are the opportunities?

9. What are the risks?

10. What are the benefits?



3.3 Proposed Solution

To debug the existing system" remove procedures that cause data redundancy" make navigational sequence proper. To provide information about audits on different levels and also to reflect the current work status depending on organization or date. To build strong password mechanism.

3.4 Problem Solution fit

Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS <ul style="list-style-type: none"> Patients Plasma Donors Blood banks Hospitals 	6. CUSTOMER CONSTRAINTS CC <ul style="list-style-type: none"> Unstable network connection Rarity of Patients' Blood Group Availability of donors/patients in the locality at a given time 	5. AVAILABLE SOLUTIONS AS <ul style="list-style-type: none"> Existing solutions provide: <ul style="list-style-type: none"> Contact details of donors Availability of blood groups Existing solutions lack: <ul style="list-style-type: none"> Immediate notifications Real-time data updation Solutions to FAQs 	Explore AS, differentiate
	2. JOBS-TO-BE-DONE / PROBLEMS J&P <ul style="list-style-type: none"> Patients find it difficult to get plasma at the right time Donors find it difficult to connect with patients nearby immediately 	9. PROBLEM ROOT CAUSE RC <ul style="list-style-type: none"> Lack of an application that works quickly and notifies users to deal with emergencies Complexity of the provided donor list with insufficient details and filters to get what the user needs 	7. BEHAVIOUR BE <ul style="list-style-type: none"> Get help from social media Seek help from friends and wellwishers Search the web for plasma donation related queries (such as Do's and Don'ts) 	
Identify strong TR & EM	3. TRIGGERS TR <ul style="list-style-type: none"> See friends and wellwishers donating plasma Reading articles about benefits of plasma donation Learning about the demand for plasma and its effects 	10. YOUR SOLUTION SL <ul style="list-style-type: none"> Immediate notifications upon requests/availability of plasma Real-time updation of donor data Chatbot to resolve queries within the app itself Find donors or patients nearby easily 	8. CHANNELS of BEHAVIOUR CH 8.1 ONLINE <ul style="list-style-type: none"> Social Media Chat applications Search Engines (for FAQs) 8.2 OFFLINE <ul style="list-style-type: none"> In-person meetings Phone calls text messages 	Identify strong TR & EM
	4. EMOTIONS: BEFORE / AFTER EM <ul style="list-style-type: none"> Patients: <ul style="list-style-type: none"> fear > calmness helpless > grateful Donors: <ul style="list-style-type: none"> depression > satisfaction regretful > delightful 			

4. REQUIREMENT ANALYSIS

4.1 Functional requirement

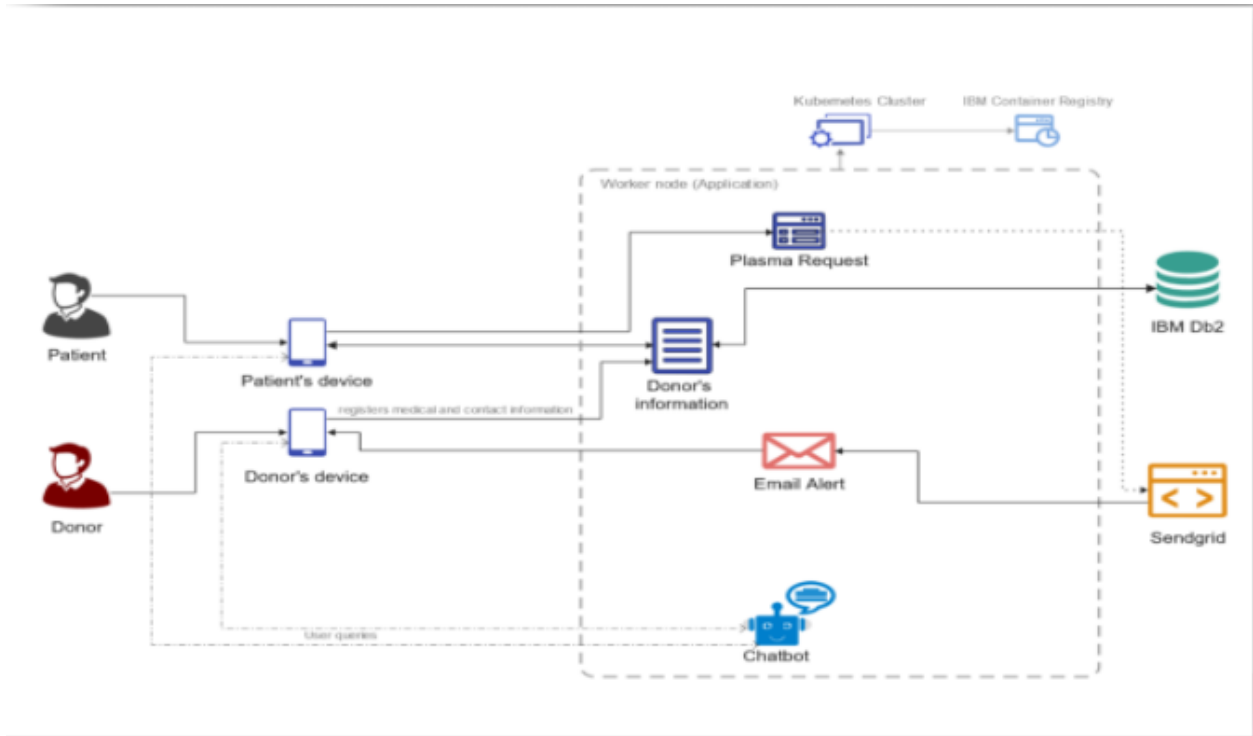
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through our website
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Email alert	Using send grid to notify users
FR-4	Eligibility of Donor	Proper medical information must be provided by Donor
FR-5	Validation of Patient	Plasma request must be validated properly
FR-6	Handle User Queries	Using a Chatbot for FAQs
FR-7	Donor Profile	Maintain and display donor's medical and contact information

4.2 Non-Functional requirements

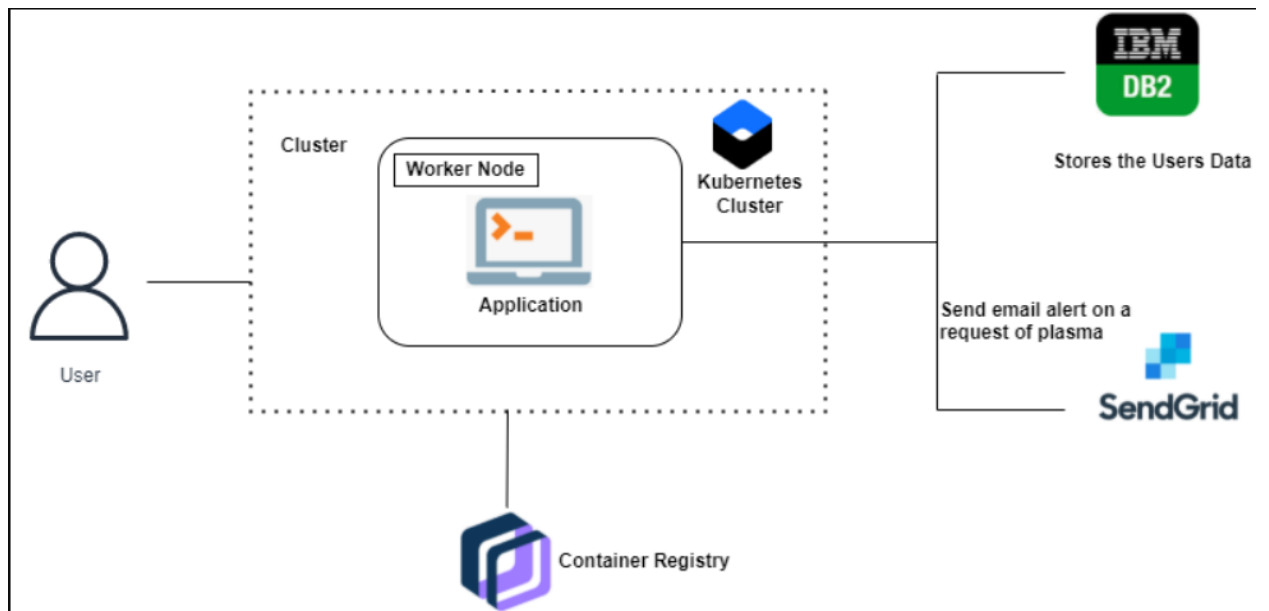
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	Anyone should be able to use the product effortlessly
NFR-2	Security	Sensitive details must be store securely (like medical information)
NFR-3	Reliability	Application must be usable even with lower bandwidth
NFR-4	Performance	Application must be able to perform up to at least 10,000 request per second
NFR-5	Availability	Application must be available for 24/7
NFR-6	Scalability	Application must be capable of handling huge number of users

5. PROJECT DESIGN

5.1 Data Flow Diagrams



5.2 Solution & Technical Architecture



6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

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

6.2 Sprint Delivery Schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.
Sprint-1		USN-2	As a user, I will receive confirmation email once I have registered for the application
Sprint-2		USN-3	As a user, I can register for the application through phone number and log in using it
Sprint-1	Login	USN-4	As a user, I can log into the application using my registered email & password
Sprint-2	Dashboard	USN-5	As a user, I want to enter/update my medical and contact information
Sprint-4	Chatbot	USN-6	As a user, I can ask questions to the chatbot
Sprint-3	Receive Alerts	USN-7	As a donor, I want to receive immediate alerts upon requests from patient
Sprint-2	Request Plasma	USN-8	As a patient, I want a list of donors
Sprint-4		USN-9	As a patient, I want to sort out donor list
Sprint-3		USN-10	As a patient, I want to request for plasma

7. CODING & SOLUTION

(Explain the features added in the project along with code)

7.1 Feature 1

Plasma Donor Application - Application to store plasma donor and recipient details

7.2 Feature 2

```
APP.PY

from flask import Flask, render_template, redirect, url_for,
request, session, flash

import ibm_db

import sendgrid

import os

from dotenv import load_dotenv

from sendgrid.helpers.mail import Mail, Email, To, Content

app = Flask(__name__)

#secret key required to maintain unique user sessions
```



```

app.secret_key =
'f39c244d6c896864abe3310b839091799fed56007a438d637baf526007609f
e0'

#establish connection with IBM Db2 Database

connection =
ibm_db.connect("DATABASE=bludb;HOSTNAME=815fa4db-dc03-4c70-
869a-
a9cc13f33084.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;POR
T=30367;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.
crt;UID=fzx82079;PWD=Gemfh13b2DRTeUqB;", "", "")

load_dotenv()    #load keys from .env

sg = sendgrid.SendGridAPIClient(api_key =
os.environ.get('SENDGRID_API_KEY'))    #set SendGrid API Key

from_email = Email("getplasmaproject@gmail.com")    #the
address that sends emails to the users


@app.route('/')

@app.route('/dashboard')

def dashboard():

```

```
        if 'username' not in session:

            return redirect(url_for('signin')) #ask user to
sign in if not done already

            return render_template('dashboard.html',
pred=session['username']) #go to homepage if signed in


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

    session.pop('username', None) #remove user session
upon signing out

    return redirect('/')


@app.route('/register')
def register():
```

```

        if 'username' in session:    #inform user if they're
already signed in the same session

            flash('You are already signed in! Sign out to login
with a different account')

            return redirect(url_for('dashboard'))

    else:

        return render_template('register.html') #take user
to the registration page


@app.route('/regform', methods=['POST'])

def regform():

    i = [i for i in request.form.values()] #get user
details from the registration form

    uname = i[0]

    uid = i[1]

    pwd = i[2]

```

```

        sql = 'SELECT * from donor WHERE email=?'    #check if
user is already registered

        pstmt = ibm_db.prepare(connection, sql)

        ibm_db.bind_param(pstmt, 1, uid)

        ibm_db.execute(pstmt)

        acc = ibm_db.fetch_assoc(pstmt)

        if acc:    #inform user to sign in if they have an
existing account

            flash('You are already a member. Please sign in
using your registered credentials')

        else:

            sql = 'INSERT INTO donor VALUES(?,?,?)' #insert
credentials of new user to the database

            pstmt = ibm_db.prepare(connection, sql)

            ibm_db.bind_param(pstmt, 1, uid)

            ibm_db.bind_param(pstmt, 2, pwd)

            ibm_db.bind_param(pstmt, 3, uname)

            ibm_db.execute(pstmt)

```

```

        to_email = To(uid)    #set user as recipient for
confirmation email

        subject = "Welcome to GetPlasma"

        content = Content("text/html", "<p>Hello " + uname +
",</p><p>Thank you for registering to the GetPlasma
Application!</p><p>If this wasn't you, then immediately report
to our <a href=\"mailto:getplasmaproject@gmail.com\">admin</a>
or just reply to this email.</p>")

        email = Mail(from_email, to_email, subject, content)
#construct email format

        email_json = email.get()    #get JSON-ready
representation of the mail object

        response = sg.client.mail.send.post(request_body =
email_json)    #send email by invoking an HTTP/POST request to
/mail/send

        flash('Registration Successful! Sign in using the
registered credentials to continue')
```

```
        return redirect(url_for('signin')) #ask users to sign
in after registration
```

```
@app.route('/signin')
```

```
def signin():
```

```
    if 'username' in session: #inform user if they're
already signed in the same session
```

```
        flash('You are already signed in! Sign out to login
with a different account')
```

```
        return redirect(url_for('dashboard'))
```

```
        return render_template('signin.html') #take user to
the sign in page
```

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

```

def signinform():

    uid = request.form['uid']    #get user id and password
from the form

    pwd = request.form['pwd']

    sql = 'SELECT uname from donor WHERE email=? AND pwd=?'
#check user credentials in the database

    pstmt = ibm_db.prepare(connection, sql)

    ibm_db.bind_param(pstmt, 1, uid)

    ibm_db.bind_param(pstmt, 2, pwd)

    ibm_db.execute(pstmt)

    acc = ibm_db.fetch_assoc(pstmt)

    if acc: #if the user is already registered to the
application

        session['username'] = acc['UNAME']

        flash('Signed in successfully!')

        return redirect(url_for('dashboard'))

    else:    #warn upon entering incorrect credentials

```

```
flash('Incorrect credentials. Please try again!')  
  
return render_template('signin.html')
```

Style.css

```
@import  
url('https://fonts.googleapis.com/css2?family=Poppins:wght@100;  
200;300;400;500;600;700;800;900&display=swap');
```

```
:root{  
    --pri: #4158D0;  
    --sec: #C850C0;  
    --ter: #FFCC70;  
}
```

```
.text-custom-accent{
```



```
    color: #E57373;
}

.text-custom-primary{
    color: var(--pri);
}

.text-custom-primary-light{
    color: #798CE8;
}

.text-custom-secondary{
    color: var(--sec)
}

.text-custom-tertiary{
    color: var(--ter)
}

.custom-primary{
```

```
background-color: var(--pri);  
  
}  
  
.custom-primary-light{  
    background-color: #798CE8;  
}  
  
.custom-secondary{  
    background-color: var(--sec)  
}  
  
.custom-secondary-bg{  
    background-color: #f48ded65;  
}  
  
.custom-tertiary{  
    background-color: var(--ter)  
}  
  
.custom-accent{
```

```
background-color: #E57373;

}

body{

    font-family: 'Poppins', Arial, Helvetica, sans-serif;

}

.main-bg{

    background-color: #4158D0;

    background-image: linear-gradient(43deg, #4158D0 0%,
#C850C0 46%, #FFCC70 100%);

}

.main-bg-medium{

    background-color: #667be2;

    background-image: linear-gradient(43deg, #8294ef 0%,
#ed85e6 46%, #ffdfa4 100%);

}

.main-bg-light{
```

```
background-color: #798CE8;

background-image: linear-gradient(43deg, #dfe5ff 0%,
#ffeffe 46%, #fff2e1 100%);

}

.rounded-rem{

border-radius: 1rem;

}

.nav-link:hover{

color: var(--sec);

}

.glass{

backdrop-filter: blur(10px);

background: #ffffff88;

}

.glass-white{
```

```
background: #ffffffaa;

}

.glass:disabled{

background: #dddddd88;

}

.glass, .glass-white, .glass:disabled{

backdrop-filter: blur(10px);

}
```

8. ADVANTAGES

- User friendliness , provided in the application with various controls.
- The system makes the overall protect management much easier and flexible.
- Readily upload the latest updates "allows users to download the alerts by clicking the url.
- It provides a high level of security with different levels of authentication.

9. CONCLUSION

Even with the rapid technological advancements and social media usage across the world, there is a lack of a quick and easy way to find plasma donors around the locality of the needy. Finding plasma donors is a challenging issue almost worldwide, even in developed countries.

So, this project aims to create a web application where the donors register themselves by providing their general and medical information, so that they will be notified when a request for plasma is received from other users residing nearby, having an acceptable blood group.

10 .References

- <https://ieeexplore.ieee.org/document/9392739>
- <https://ieeexplore.ieee.org/document/9396012>
- <https://ieeexplore.ieee.org/document/9271296>