PLASMA DONAR APLLICATION

IBM-Project-28833-1660117238

PLASMA DONAR APLLICATION

NALAIYA THIRAN PROJECT BASED LEARNING ON PROFESSIONAL READLINESS FOR INNOVATION, EMPLOYNMENT AND ENTERPRENEURSHIP

PROJECT REPORT

Priyadharshini R(714019104075)

Rukash DG(714019104086)

Sineka AS(714019104096)

Shobika R(210419104183)

BACHELOR OF ENGINEERING IN COMPUTR SCIENCE AND ENGINEERING

Chennai Institute Of Technology

Chennai-600 069

INDEX

3. Reports from JIRA

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

1. Feature 1
2. Feature 2

8. TESTING

1. Test Cases
2. User Acceptance Testing

9. RESULTS

1. Performance Metrics

10. ADVANTAGES & DISADVANTAGES

11. CONCLUSION

12. FUTURE SCOPE

13. APPENDIX

Source Code

GitHub & Project Demo Link

1. INTRODUCTION

1.1 PROJECT OVERVIEW

Category: Cloud App Development

Team ID : PNT2022TMID24579

☎Skills Required: IBM Cloud,HTML,Javascript,IBM Cloud Object Storage,PythonFlask,Kubernetes,Docker,IBM DB2,IBM Container Registry

The Blood Donation Agent is to create an e-Information about the donor and organization that are related to donating the blood. Through this application any person who is interested in donating the 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 make 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

This project is aimed to developing an online Blood Donation Information. The entire project has been developed keeping in view of the distributed client server computing technology" in mind.

The Blood Donation Agent is to create an e-Information about the donor and organization that are related to donating the blood. Through this application any person who is interested in donating the 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 make 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.
The project has been planned to be having the view of distributed architecture" with centralized storage of the database. The application for

the storage of the data has been planned. Using the constructs of MS-SQL server and all the user interfaces have been designed using the ASP.Net technologies.

The database connectivity is planned using the "SQL Connection" methodology. The standards of security and data protective mechanism have been given a big choice for proper usage.The application takes care of different modules and their associated reports" which are produced as per the applicable strategies and standards that are put forwarded by the administrative staff.

The entire project has been developed keeping in view of the distributed client server computing technology" in mind. The specification has been normalized up to 3NF to eliminate all the anomalies that may arise due to the database transaction that are executed by the general users and the organizational administration. The user interfaces are browser specific to give distributed accessibility for the overall system. The internal database has been selected as MS-SQL server 2000.

The basic constructs of table spaces" clusters and inde0es have been e0ploited to provide higher consistency and reliability for the data storage. The MS-SQL server 2000 was a choice a sit provides the constructs of high-level reliability and security. The total front end was dominatedusing the A%(.)et technologies. At all proper levels high care was taken to check that the system manages the data consistency with proper business rules or validations.

The database connectivity was planned using the latest "SQL Connection" technology provided by Microsoft corporation. The authentication and authorization was cross checked at all the relevant stages. The user level accessibility has been restricted into two zones namely.

2. LITERATURE SURVEY

2.1 EXISTING PROBLEM

Introduction

Applying optimization methods to healthcare management and logistics is a developing research area with numerous studies. Specifically, facility location, staff rostering, patient allocation, and medical supply transportation are the main themes analysed. Optimization approaches have been developed for several healthcare related problems, ranging from the resource management in hospitals to the delivery of care services in a territory. However, optimization approaches can also improve other services in the health system that have been only marginally addressed, yet. One of them is the Blood Donation (BD) system, aiming at providing an adequate supply of blood to Transfusion Centres (TCs) and hospitals. Blood is necessary for several treatments and surgeries, and still a limited resource.

The need for blood is about ten million units per year in the USA, 2.1 in Italy and 2 in Turkey; moreover, people still die in some countries because of inadequate supply of blood products (World Health Organization 2014). Hence, BD plays a fundamental role in healthcare systems, aiming at guaranteeing an adequate blood availability to meet the demand and save lives. In Western countries, blood is usually collected from donors, i.e., unpaid individuals who give blood voluntarily. Blood is classified into groups (A and subgroups, B, 0 or AB) and based on the Rhesus factor (Rh+ or Rh-), and each donor should be correctly matched with the patient who receives his/her blood. Moreover, as it may transmit diseases, blood must be screened before utilization.

2.2 REFERENCES

S.NO	TITLE	AUTHORS	ABSTRACT	DRAWBACKS
1	Developing a plasma	Aishwarya R Gowri Jain	A plasma is a liquid portion of the blood, over 55% of	• Internet: It would require

	donor application using Function-asa-service in AWS	University, Department of MCA, computer science	human blood is plasma. Plasma is used to treat various infectious diseases and it is one of the oldest methods known as plasma therapy. Plasma therapy is a process where blood is donated by recovered patients in order to establish antibodies that fights the infection. In this project plasma donor application is being developed by using AWS services. The services used are AWS Lambda, API gateway, DynamoDB, AWS Elastic Compute Cloud with the help of these AWS services, it eliminates the need of configuring the servers and reduces the infrastructural costs associated with it and helps to achieve serverless computing. For instance, during COVID 19 crisis the requirement for plasma increased drastically as there were no vaccination found in order to treat the infected patients, with plasma therapy the recovery rates where high but the donor count was very low and in such situations it was very important to get the information about the plasma donors. Saving the donor information and notifying about the current donors would be a helping hand as it can save time and help the users to track down the necessary information about the donors.	an internet connection for the working of the website. • handle multiple requests at the same time
2	Optimization of Blood Donor Information and	• K. Yamini, M. E(CSC), SVCET, Thirupachur, India	Emergency situations, such as accidents, create an immediate, critical need for specific blood type. In addition to emergency	• The accuracy of the location displayed on the map was beyond the

	Management System	• R. Devi, Asst. Professor, SVCET, Thirupachur, India	requirements, advances in medicine have increased the need for blood in many ongoing treatments and elective surgeries. Despite increasing requirements for blood, only about 5% of the Indian population donates blood. In this paper we propose a new and efficient way to overcome such scenarios with our project. We have to create a new idea, just touch the button. Donor will be prompted to enter an individual's details, like name, phone number, and blood type. After that your contact details will appear in alphabetical order on the screen; the urgent time of a blood requirement, you can quickly check for contacts matching a particular or related blood group and reach out to them via Phone Call/SMS through the Blood donor App.	scope of this Project. • Only Android was used as a mobile operating system to test the application
3	Blood Bank Management Information System in India	• Vikas Kulshreshtha Research Scholar, • Dr.Sharad Maheshwari, Associate Professor	A blood bank is a bank of blood or blood components, gathered as a result of blood donation, stored and preserved for later use in blood transfusion. To provide web based communication there are numbers of online web based blood bank management system exists for communicating between department of blood centers and hospitals, to satisfy blood necessity, to buy, sale and stock the blood, to give information about this blood. Manual systems as compared to Computer Based Information Systems are time consuming, laborious, and costly. This paper	• Do not provide the better inventory solution to the end use • It requires an active internet connection.

			introduces the review of the main features, merits and demerits provided by the existing Web -Based Information System for Blood Banks. This study shows the comparison of various existing system and provide some more idea for improve the existing system. First I will give some basic introduction about blood banks then I will try to provide comparative study of some existing web based blood bank system. After that I will introduce some new idea for improving the existing techniques used in web based blood bank system and at end I will conclude this paper	
4	A Research Paper on Blood Donation Management System	• Devanjan K. Srivastava • Utkarsh Tanwar • M.G.Krishna Rao • Priya Manohar • Balraj Singh	Blood donation and transfusion has been an ever - serious issue and the shortage of blood throughout the world has caused many people to lose their life. The lack of a centralized system for blood donation is majorly responsible for those losses. Now in the era of online and digital processes, the conventional methods of collecting blood are absolute. An automated system is required to manage the centers and to showcase the information to the interested parties. We have developed a website that singlehandedly solves all these issues related to blood donation and reception. We have designed a SQLite database as an integral part of the integrated framework to store historical blood donation data in a centralized database for	• Internet Connection is mandatory • There is no proper centralized database for registered donors

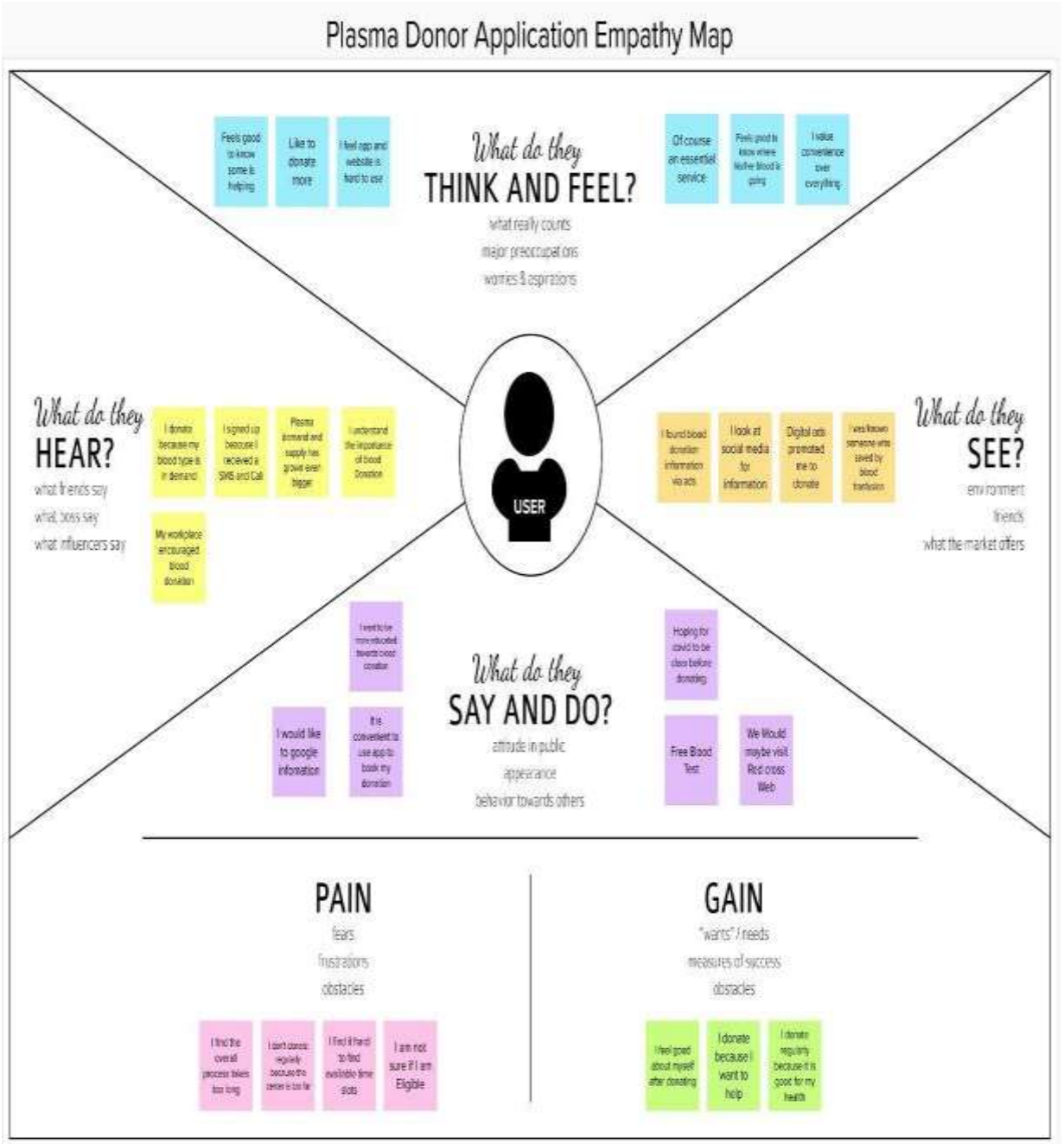
			analytical processing. The proposed system would enable people to register as a donor to make themselves available whenever in need of their blood type. We have introduced a search tab to search available people ready to donate. In our proposed system in the donor registration, health - related details would be updated in the blood management system database for all to see	
5	A Study on Blood Bank Management	• A. Clemen Teena, K • Sankar • S. Kannan	'Blood Bank Information System' will be an information management system which helps to manage the records of donors and patients at a blood bank. The system will allow the authorized blood bank officer to login using a secret password and easily manage the records of the blood donors and the patients in need of blood	• No search filter available • UI improvem ent in Login page

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.

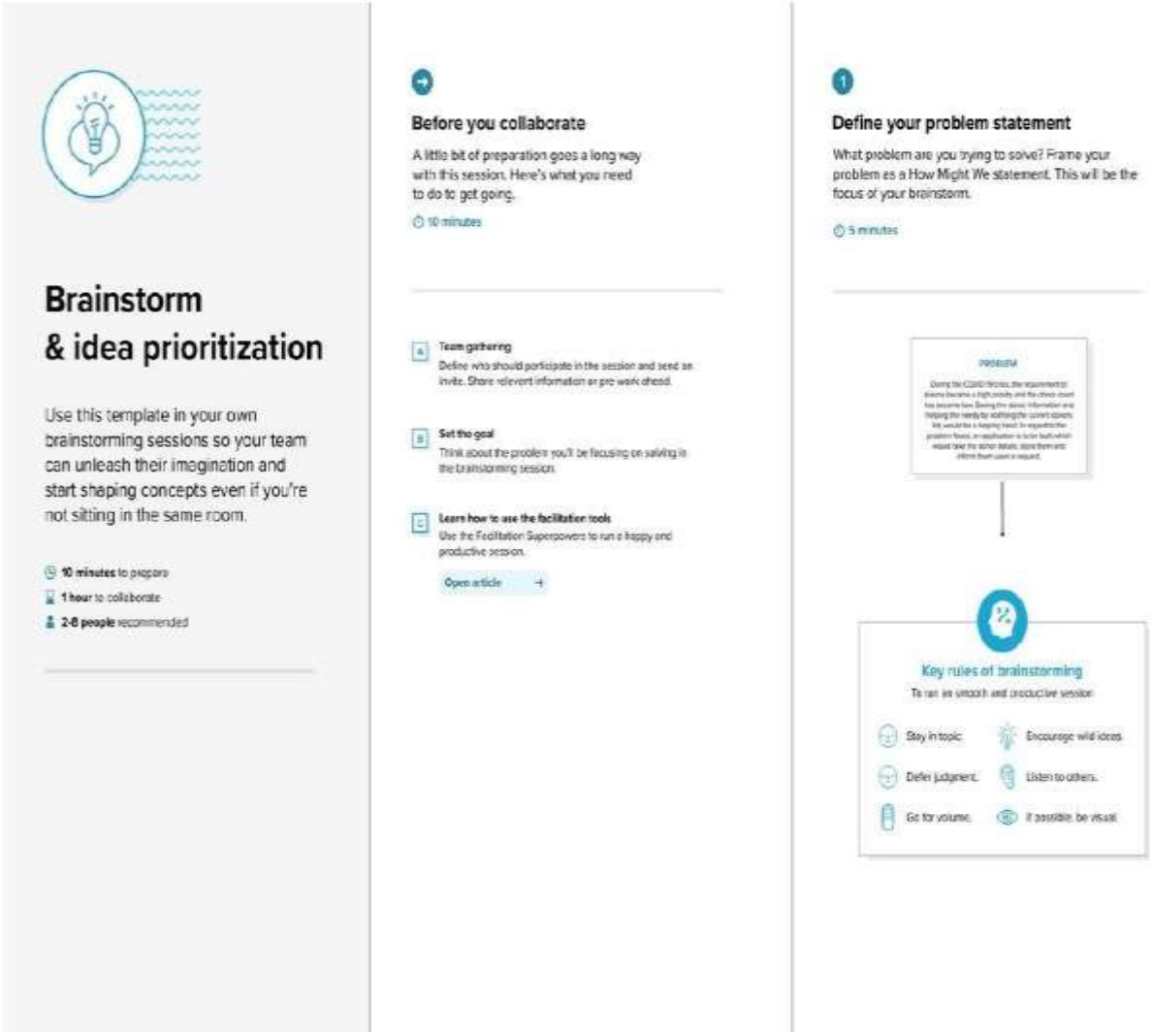
3. IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS

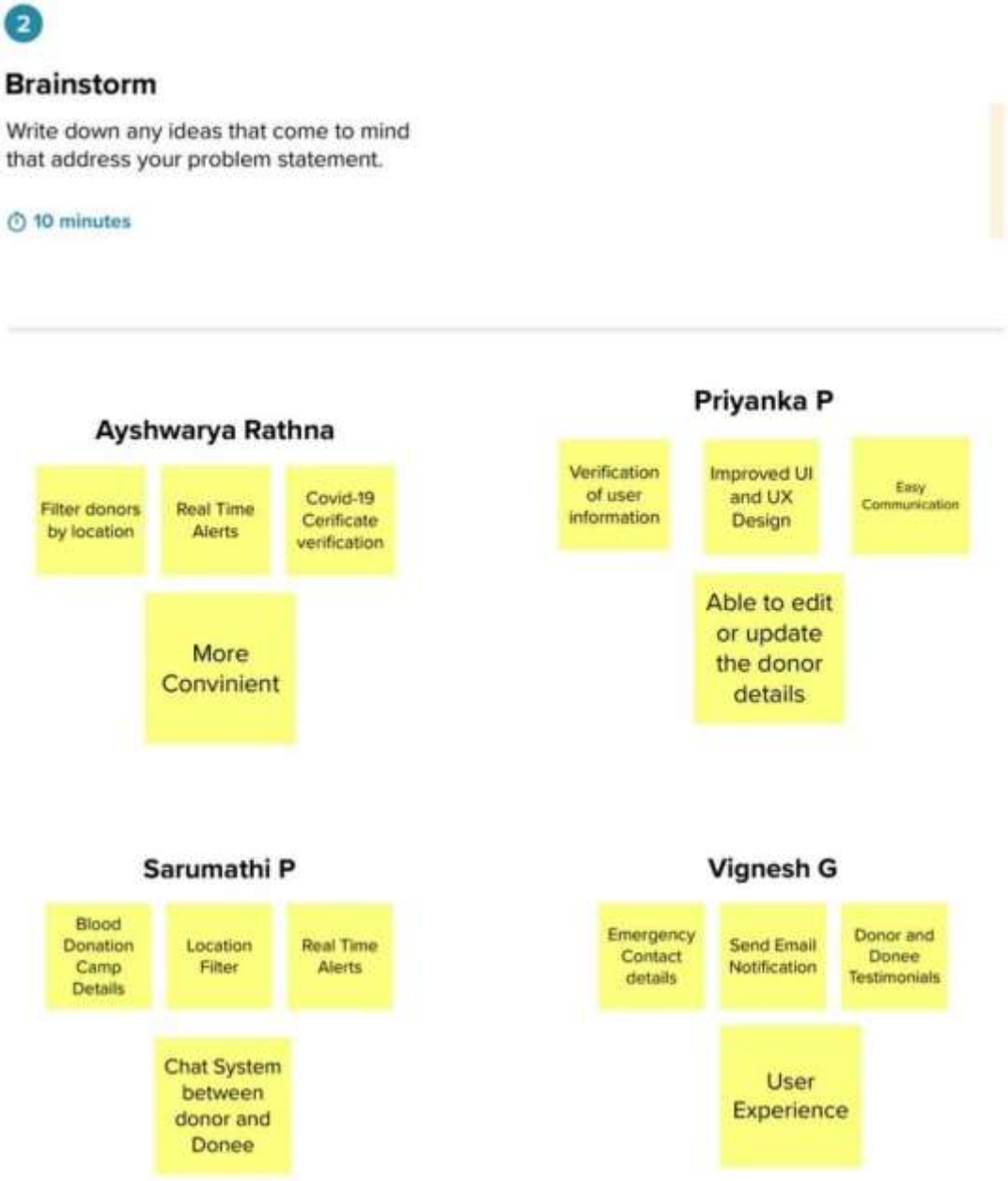


3.2 IDEATION & BRAINSTORMING

Step 1:Team Gathering Collaboration and Select the Problem Statement



Step 2:BrainStorm And Idea Listing



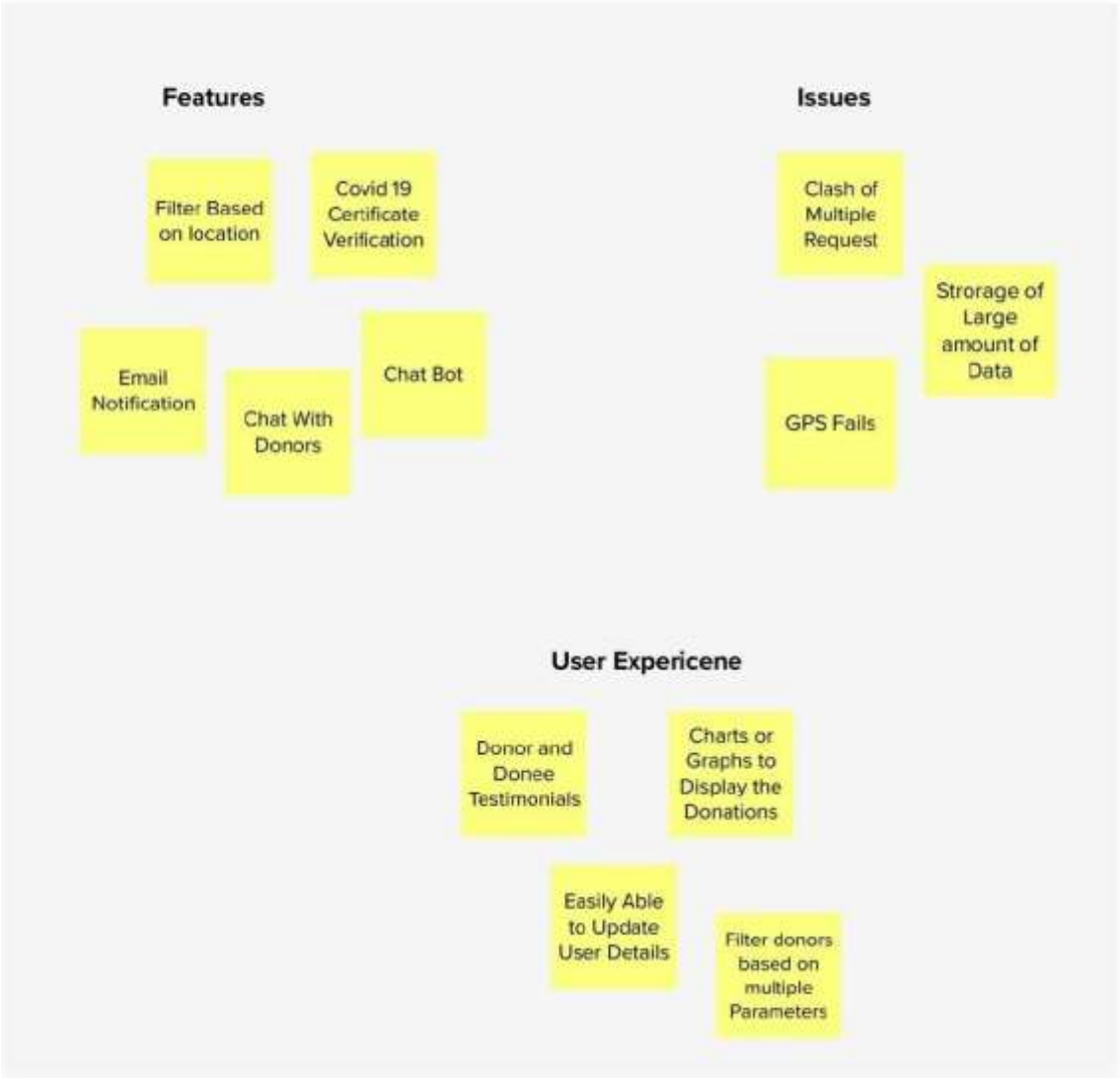
Step 3:Grouping

3

Group ideas

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

20 minutes



Step 4:Idea Prioritization

Step 5:Top Ideas

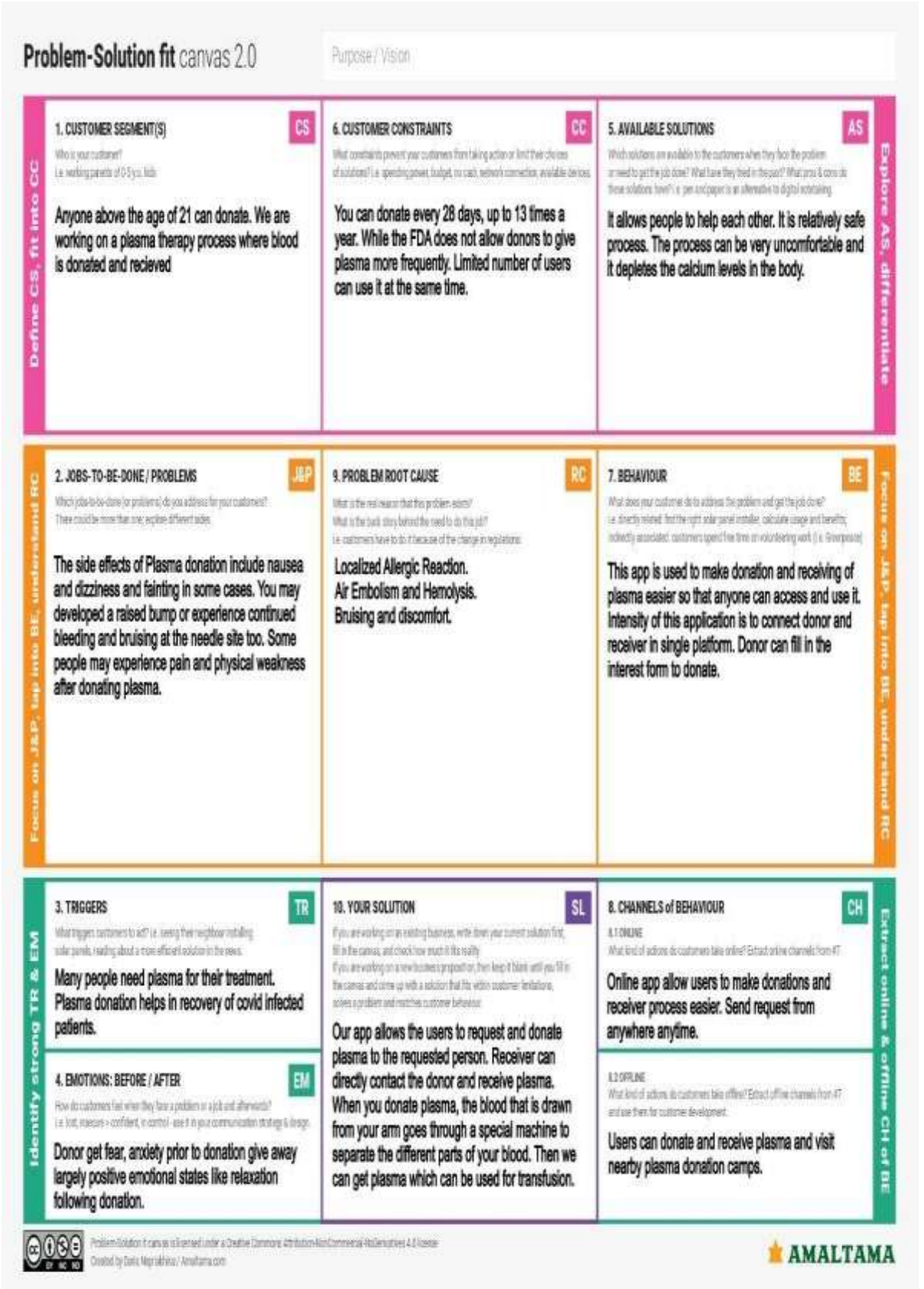
3.3 PROPOSED SOLUTION

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.

S.No	Parameter	Description
1.	Problem statement (Problem to be solved)	With the number of people affected by COVID-19 infection the demand for the plasma of recovered patients has gone up tremendously. This creates chaotic situation for everyone as this is very crucial because this may risk many lives. So, this situation needs a systematic and quick solution. Searching eligible donor would surely be strenuous job.
2.	Idea / Solution	Smart application would be the perfect solution to manage donating and searching donors for plasma. So, this application searches perfect donor. The system works with the registration of a donor by providing the required details that gets stored in the database.
3.	Novelty / Uniqueness	There exist applications that allow donors to register for donations. But out application also allow patients to register and the application searches the most eligible donor.
4.	Solution Impact / Customer Satisfaction	Due to Covid-19, supply to the plasma demand became a serious issue. This application aims to ease the procedure of finding the most eligible donor for the patient. Now the user will be able to donate and receive plasma donation with a lot of ease.

5.	Business Model (Revenue Model)	➢ **Key partners**: SSN and IBM both together will work to develop the application. ➢ **Key resources**: Resources for development are IDEs, IBM's database, several software, etc. ➢ **Activities**: The main activities include development of the application using flask, interfacing with IBM db2, SendGrid and hosting it on cloud. ➢ **Value proposition**: Users will get a friendly GUI and will serve all the tasks. Data will be secure and privacy will be maintained. ➢ **Cost structure**: No such cost is required. IBM provides the software. Except that, some software may require payments. ➢ **Revenue streams**: NA ➢ **Customer segments**: Students, medical professionals, patients, donors ➢ **Customer relationships**: There will be confidentiality within the users. All users will be treated with fair means. Channels: The website application will be hosted on various social media platforms.
6.	Scalability of Solution	The application will be scalable in future also. This application could be used by NGOs and govt hospitals. Further, developers need to maintain and update the website for future requirements. New features will promote the application and will further attract more users.

3.4 PROBLEM SOLUTION FIT



4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREENTS

Functional Requirement (Epic)	User Story Number	User Story / Task
Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.
Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application
Login	USN-4	As a user, I can log into the application by entering email & password
Registration	USN-3	As a user, I can register for the application
Dashboard	USN-5, USN-6, USN-7	I am a Donor and need to access only Donor registration with my credentials
Donor's Page	USN-8	As a Donor, I can enter my details and check my eligibility, and book my slot for donation
Recipient's Page	USN-9	As a Recipient, I can enter my details and book my slot in a hospital as any nearby.
Hospital In-Charge Page	USN-10	As a Hospital In-Charge, I can enter my details and hospital details as per the conditions.
At last feedback page	US-11	Finally, all users enter their feedback and receive feedbacks and issues.

4.2 NON-FUNCTIONAL REQUIREENTS

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Effectiveness, efficiency and overall satisfaction of the user while interacting with our application.
NFR-2	**Security**	Authentication, authorization, encryption of the application.
NFR-3	**Reliability**	Probability of failure-free operations in a specified environment for a specified time.
NFR-4	**Performance**	How the application is functioning and how responsive the application is to the end-users.
NFR-5	**Availability**	Without near 100% availability, application reliability and the user satisfaction will affect the solution.
NFR-6	**Scalability**	Capacity of the application to handle growth, especially in handling more users.

5. PROJECT DESIGN

5.1 DATA FLOW DIAGRAMS

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

STEPS:

1. Donor can enter their details and check their eligibility.

2. Hospital In-Charge enter their hospital details and register themselves.

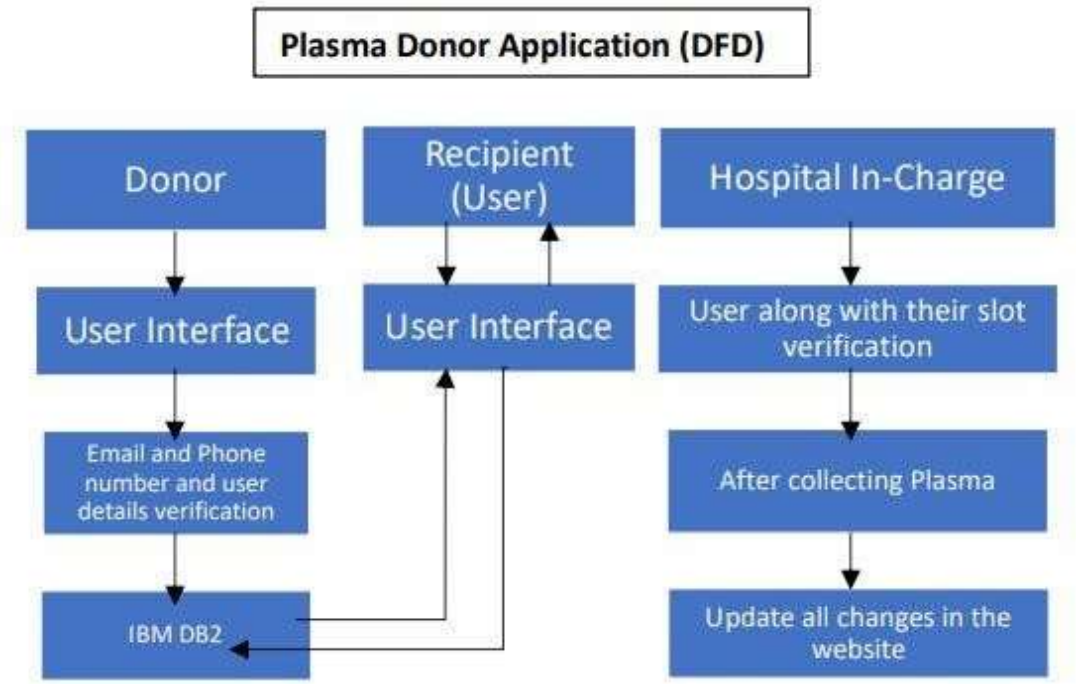
3. Recipients can enter their details and book their slots.

4. After Donor's donation finished, In-charge update the details in database.

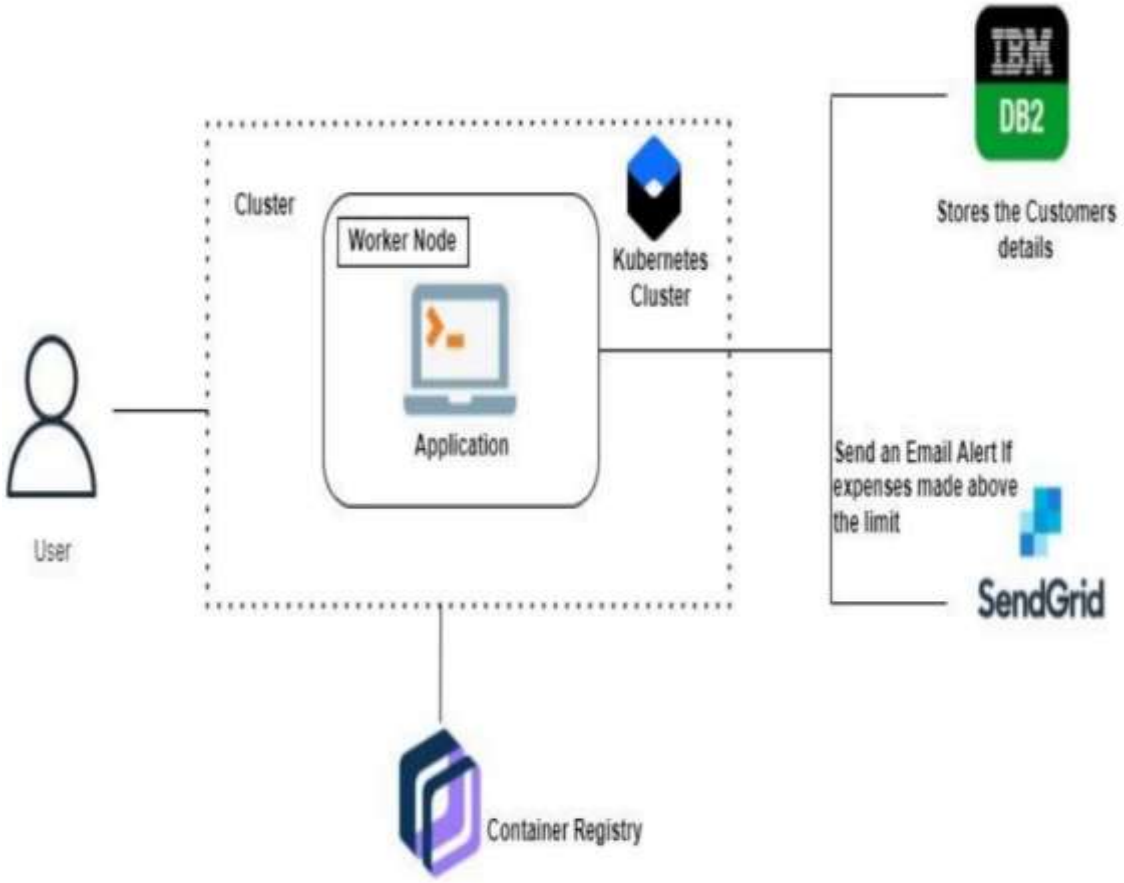
5. After Recipient's request for plasma, In-charge has to allocate the the appropriate plasma for recipient.

6. After the process finished, all users enter their feedback to their appropriate requests.

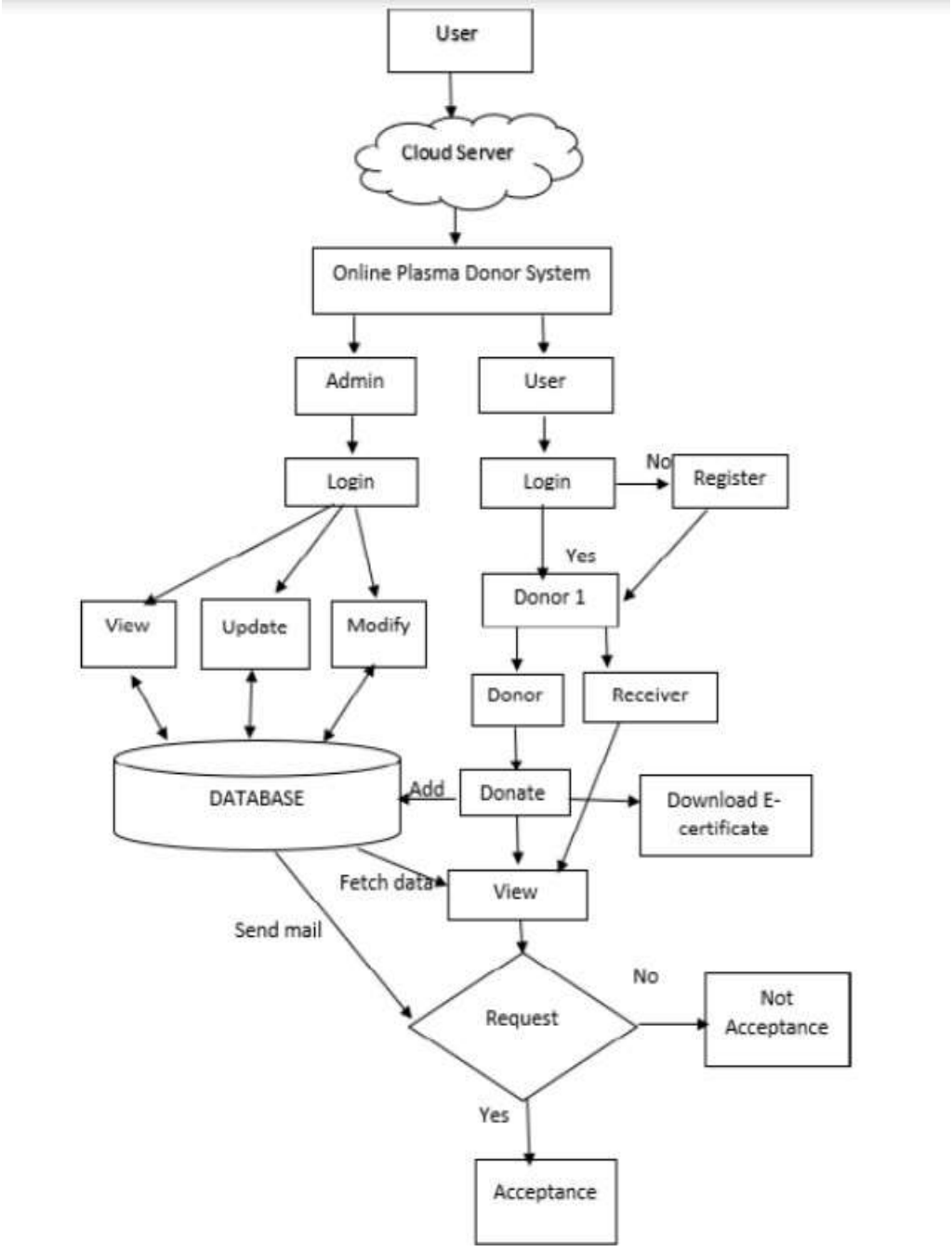
7. All the changes can enter into DB2.



5.2 SOLUTION & TECHNICAL ARCHITECTURE



SOLUTION ARCHITECTURE



5.3 USER STORIES

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Donor / Recipient / Hospital In-Charge (Mobile/Desktop user)	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
		USN-2	As a user, I will receive confirmation email or SMS once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Gmail and Phone Number.	I can register & access the dashboard with Gmail or any kind of Login	Medium	Sprint-2
	Login	USN-4	As a user, I can log into the application by entering email or phone number & password	I can Log into the Application by using Email ID and Password	High	Sprint-1
Donor / Recipient / Hospital In-Charge (Web user)	Dashboard	USN-5	As a user, I can be allowed to choose the three options like Donor, Recipient and Hospital In-Charge.	I am a Donor and need to access only Donor registration with my credentials	Medium	Sprint-3
		USN-6		I am a Recipient and need to access only Recipient registration with my credentials.	Medium	Sprint-3
		USN-7		I am a Hospital In-Charge and need to access only In-Charge registration with my hospital's credentials	Medium	Sprint-3
Donor	Donor's Page	USN-8	As a Donor, I can enter my details and check my eligibility, and book my slot for donation	I am donor, I can get the slot timings and nearby hospital details.	High	Sprint-4
Recipient	Recipient's Page	USN-9	As a Recipient, I can enter my details and book my slot in a hospital as any nearby.	I am a recipient; I can get the appropriate Plasma present in nearby areas.	High	Sprint-4
Hospital In-Charge	Hospital In-Charge Page	USN-10	As a Hospital In-Charge, I can enter my details and hospital details as per the conditions.	I am a Hospital In-Charge; I can check the user credentials and do my process	High	Sprint-4
All users (Donor, Recipient, Hospital In-Charge)	At last feedback page	USN-11	Finally, all users enter their feedback and receive feedbacks and issues.	I am a user; I can send and receive queries through feedback pages.	Medium	Sprint-4

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	10	High	Priyanka
Sprint-1	Registration	USN-2	As a user, I will receive confirmation email once I have registered for the application	5	High	Priyanka
Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	5	High	Ayshwarya
Sprint-2	Registration	USN-3	As a user, I can register for the application	20	Low	Ayshwarya
Sprint-3	Dashboard	USN-5, USN-6, USN-7	I am a Donor and need to access only Donor registration with my credentials	20	High	Vignesh
Sprint-4	Donor's Page	USN-8	As a Donor, I can enter my details and check my eligibility, and book my slot for donation	5	High	Vignesh
Sprint-4	Recipient's Page	USN-9	As a Recipient, I can enter my details and book my slot in a hospital as any nearby.	5	High	Sarumathi
Sprint-4	Hospital In-Charge Page	USN-10	As a Hospital In-Charge, I can enter my details and hospital details as per the conditions.	9	High	Sarumathi
Sprint-4	At last feedback page	US-11	Finally, all users enter their feedback and receive feedbacks and issues.	1	Medium	Priyanka

6.2 SPRINT DELIVERY SCHDULE

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	4 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	13 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	14 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 duration = 6 days

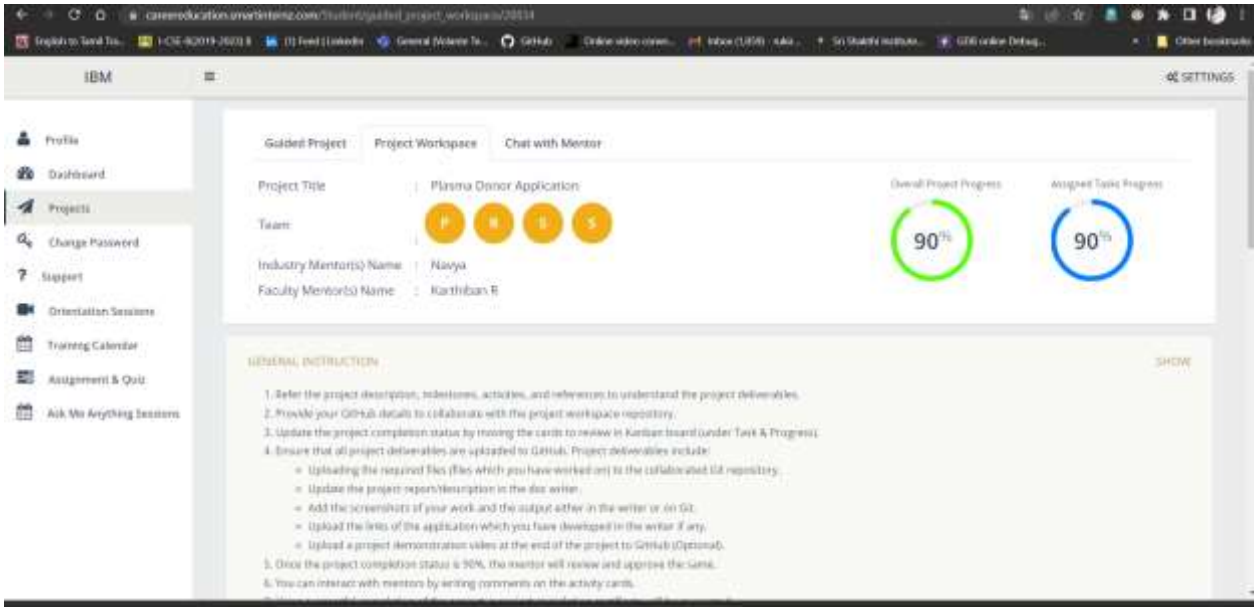
velocity = 20

AV = VELOCITY / SPRINT DURATION

AV = 20 / 6

AV = 3.333

6.3 REPORTS FROM JIRA



7. CODING & SOLUTIONING

7.1 FEATURE 1

LOGIN

```python
# Login
@app.route("/loginmethod", methods = ['GET'])
def loginmethod():
    global userid
    msg = ''

    if request.method == 'GET':
        uname = request.args.get("uname")
        psw = request.args.get("psw")

        sql = "SELECT * FROM accounts WHERE username =? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, uname)
        ibm_db.bind_param(stmt, 2, psw)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)

        if uname == 'admin' and psw == 'admin':
            return redirect(url_for('admin'))

        if account:
            session['loggedin'] = True
            session['id'] = account['USERNAME']
            userid = account['USERNAME']
            session['username'] = account['USERNAME']
            return redirect(url_for("about"))
        else:
            msg = 'Incorrect Username and Password'
            flash(msg)
            return redirect(url_for("login"))
```

SIGNUP

```python
# Signup
@app.route("/signupmethod", methods = ['POST'])
def signupmethod():
    msg = ''
    if request.method == 'POST':
        uname = request.form['uname']
        email = request.form['email']
        name = request.form['name']
        dob = request.form['dob']
        psw = request.form['psw']
        con_psw = request.form['con_psw']

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

        if account:
            msg = 'Account already exists !'
            flash(msg)
            return redirect(url_for("signup"))
        elif psw != con_psw:
            msg = "Password and Confirm Password do not match."
            flash(msg)
            return redirect(url_for("signup"))
        else:
            insert_sql = "INSERT INTO accounts VALUES (?, ?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, dob)
            ibm_db.bind_param(prep_stmt, 4, uname)
            ibm_db.bind_param(prep_stmt, 5, psw)
            ibm_db.execute(prep_stmt)

            insert_donor = "INSERT INTO donor(Name,Username,Email,DOB,Availability) VALUES (?, ?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_donor)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, uname)
            ibm_db.bind_param(prep_stmt, 3, email)
            ibm_db.bind_param(prep_stmt, 4, dob)
            ibm_db.bind_param(prep_stmt, 5, "Not Available")
            ibm_db.execute(prep_stmt)

            sendmail(email,'Plasma donor App login',name, 'You are successfully Registered!')

            return redirect(url_for("login"))
```

FEATURE 2

SEND MAIL TO SELECTED USER

```
@app.route('/sendEmail', methods = ["GET", "POST"])
def sendEmail():
    if request.method == 'POST':
        if request.form['select'] == 'select':
            email = request.form["Email"]
            uname = request.form['Username']
            curr_uname = session["username"]
            name = request.form['Name']
            select = "SELECT * from requests where Username = ? and Requestuname = ?"
            stmt = ibm_db.prepare(conn, select)
            ibm_db.bind_param(stmt, 1, uname)
            ibm_db.bind_param(stmt, 2, curr_uname)
            ibm_db.execute(stmt)
            bool = ibm_db.fetch_assoc(stmt)

            print("boolean"+str(bool))
            if not bool:
                request_sql = "INSERT INTO requests VALUES (?, ?)"
                stmt = ibm_db.prepare(conn, request_sql)
                ibm_db.bind_param(stmt, 1, uname)
                ibm_db.bind_param(stmt, 2, curr_uname)
                ibm_db.execute(stmt)
                sendmail(email, 'Plasma donor App plasma request', name,'You have received a request for Plasma Donation from a
            else:
                print(bool)
            print(email)
            print(name)
    return render_template("donorlist.html", value=value)
```

SEARCH ACCORDING BLOOD TYPE AND LOCATION

```
140
141 @app.route('/requested',methods=['POST'])
142 def requested():
143     global value
144     bloodgrp = request.form['bloodgrp']
145     city = request.form['city']
146
147     send_sql = "SELECT * FROM donor where BLOODTYPE = ? and CITY = ? and USERNAME != ? and AVAILABILITY = ?"
148     prep_stmt = ibm_db.prepare(conn, send_sql)
149     ibm_db.bind_param(prep_stmt, 1, bloodgrp)
150     ibm_db.bind_param(prep_stmt, 2, city)
151     ibm_db.bind_param(prep_stmt, 3, session['username'])
152     ibm_db.bind_param(prep_stmt, 4, 'Available')
153     ibm_db.execute(prep_stmt)
154     row = ibm_db.fetch_assoc(prep_stmt)
155
156     value = {}
157     ind = 0
158     while row != False:
159         value[ind] = row
160         ind += 1
161         row = ibm_db.fetch_assoc(prep_stmt)
162     print(value)
163
164     return render_template("donorlist.html", value=value)
```

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
1	Functional	Login Page	Verify user is able to Login into the Application		1) Open the Plasma Donor Applicaion 2) Login with user Credentials	Username: Priyanka Password: test	Login Successful	Working as expected	Pass
2	Functional	Signup Page	Verify user is able to Signup in the Application		1) Open the Plasma Donor Applicaion 2) Enter the Details and Create a new User 3) Verify if user is created and	Username: Ayshu Password: test Name: Ayshu DOB: 12/9/2001 Password: test	Account Created Successfully	Working as expected	Pass
3	Functional	Personal Details page	Verify if all the user details are stored in Database		1) Open the Plasma Donor Applicaion 2) Enter the Details and Create a new User 3) Verify if user is created and	Username: chalam@gmail.com password: Testing123	User should navigate to user account homepage		
4	Functional	Login page	Verify user 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 valid password in password	Username: chalam@gmail password: Testing123	Application should show 'Incorrect email or password ' validation message.		
5	Functional	Login page	Verify user 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 Valid username/email in Email text box	Username: chalam@gmail.com password: Testing123678686786876876	Application should show 'Incorrect email or password ' validation message.		

Test Scenarios

1. Verify user is able to see login page
2. Verify user is able to login to application or not?
3. Verify user is able to navigate to create your account page?
4. Verify user is able to recovery password
5. Verify login page elements

Search

1. Verify user is able to search by entering keywords in search box
2. Verify user is able to see suggestions based on keyword entered in search box
3. Verify user is able to see related auto suggestions displaying based on keyword entered in search box
4. Verify user is able to see no matches found message when no results are matching with entered keyword
5. Verify user is able to see seach detailed page when nothing entered in textbox

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	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	5	0	0	0	5
Duplicate	1	0	0	0	1
External	0	0	0	0	0
Fixed	3	0	0	0	3
Not Reproduced	2	0	0	0	2
Skipped	0	0	0	0	0
Won't Fix	0	0	0	0	0
Totals	10	0	0	0	10

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	0	0	0	0
Client Application	5	0	0	5
Security	0	0	0	0
Outsource Shipping	0	0	0	0
Exception Reporting	0	0	0	0

9. RESULTS

9.1 PERFORMANCE METRICS

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status
1	Functional	Login Page	Verify user is able to Login into the Application		1) Open the Plasma Donor Applicaion 2) Login with user Credentials	Username: Priyanka Password: test	Login Successful	Working as expected	Pass
2	Functional	Signup Page	Verify user is able to Signup in the Application		1) Open the Plasma Donor Applicaion 2) Enter the Details and Create a new User 3) Verify if user is created and	Username: Ayshu Password: test Name: Ayshu DOB: 12/9/2001 Password: test	Account Created Successfully	Working as expected	Pass
3	Functional	Personal Details page	Verify if all the user details are stored in Database		1) Open the Plasma Donor Applicaion 2) Enter the Details and Create a new User 3) Verify if user is created and	Username: chalam@gmail.com password: Testing123	User should navigate to user account homepage		
4	Functional	Login page	Verify user 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 valid password in password	Username: chalam@gmail password: Testing123	Application should show 'Incorrect email or password ' validation message.		
5	Functional	Login page	Verify user 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 Valid username/email in Email text box	Username: chalam@gmail.com password: Testing123678686786876876	Application should show 'Incorrect email or password ' validation message.		

10. ADVANTAGES & DISADVANTAGES

1. ADVANTAGES

The project is identified by the merits of the system offered to the user. The merits of this project are as follows; -

• It's a web-enabled project.

• This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.

• The user is mainly more concerned about the validity of the data, whatever he is entering. There are checks on every stages of any new creation, data entry or updation so that the user cannot enter the invalid data, which can create problems at later date.
• Sometimes the user finds in the later stages of using project that he needs to update some of the information that he entered earlier. There are options for him by which he can update the records. Moreover there is restriction for his that he cannot change the primary data field. This keeps the validity of the data to longer e0tent.
• User is provided the option of monitoring the records he entered earlier. He can see the desired records with the variety of options provided by him.
• From every part of the project the user is provided with the links through framing so that he can go from one option of the project to other as per the requirement. This is bound to be simple and very friendly as per the user is concerned. That is" we can sat that the project is user friendly which is one of the primary concerns of any good project.
• Data storage and retrieval will become faster and easier to maintain because data is stored in a systematic manner and in a single database.
• Decision making process would be greatly enhanced because of faster processing of information since data collection from information available on computer takes much less time then manual system.
• Allocating of sample results becomes much faster because at a time the user can see the records of last years.
• Easier and faster data transfer through latest technology associated with the computer and communication.
• Through these features it will increase the efficiency, accuracy and transparency.

2. DISADVANTAGES

- Wrong inputs will affect the project outputs.
- Internet Connection is mandatory.
- Reports are not Verified

11. CONCLUSION

This project proved good for me as it provided practical knowledge of not only programming in ASP.NET and VB.NET web based application and no some extent windows Application and SQL Server, but also about all handling procedure related with "Plasma Donor Application". It also provides knowledge about the latest technology used in developing web enabled application and client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently.

12. FUTURE SCOPE

Plasma Donor Application is a web application to build such a way that it should suits for all type of blood banks in future. One important future scope is availability of location-based blood bank details and extraction of location-based donor’s detail, which is very helpful to the acceptant people. All the time the network facilities cannot be use. This time donor request does not reach in proper time, this can be avoided through adding some message sending procedure this will help to find proper blood donor in time. This will provide availability of blood in time.

SOURCE CODE

```
from flask import *
import ibm_db
from sendgridmail import sendmail
import os
from dotenv import load_dotenv

load_dotenv()
```

```
conn = ibm_db.connect(os.getenv('DB_KEY'),",")

app = Flask(_name_)

app.app_context().push()
app.config["TEMPLATES_AUTO_RELOAD"] = True
app.config['SECRET_KEY'] = 'AJDJRJS24$($(#$$33--'

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

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

# Login
@app.route("/loginmethod", methods = ['GET'])
def loginmethod():
    global userid
    msg = ''

    if request.method == 'GET':
        uname = request.args.get("uname")
        psw = request.args.get("psw")
```

```
        sql = "SELECT * FROM accounts WHERE username =? AND
password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, uname)
        ibm_db.bind_param(stmt, 2, psw)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)

        if uname == 'admin' and psw == 'admin':
            return redirect(url_for('admin'))

        if account:
            session['loggedin'] = True
            session['id'] = account['USERNAME']
            userid = account['USERNAME']
            session['username'] = account['USERNAME']
            return redirect(url_for("about"))
        else:
            msg = 'Incorrect Username and Password'
            flash(msg)
            return redirect(url_for("login"))

@app.route("/admin")
def admin():
    send_sql = "SELECT * FROM donor"
    prep_stmt = ibm_db.prepare(conn, send_sql)
    ibm_db.execute(prep_stmt)
```

```
    row = ibm_db.fetch_assoc(prep_stmt)

    values = {}
    ind = 0
    while row != False:
        values[ind] = row
        ind += 1
        row = ibm_db.fetch_assoc(prep_stmt)
    print(values)

    return render_template('admin.html',values=values)

# Signup
@app.route("/signupmethod", methods = ['POST'])
def signupmethod():
    msg = ''
    if request.method == 'POST':
        uname = request.form['uname']
        email = request.form['email']
        name = request.form['name']
        dob = request.form['dob']
        psw = request.form['psw']
        con_psw = request.form['con_psw']

        sql = "SELECT * FROM accounts WHERE username =?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, uname)
```

```
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)

        if account:
            msg = 'Account already exists !'
            flash(msg)
            return redirect(url_for("signup"))
        elif psw != con_psw:
            msg = "Password and Confirm Password do not match."
            flash(msg)
            return redirect(url_for("signup"))
        else:
            insert_sql = "INSERT INTO accounts VALUES (?, ?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, email)
            ibm_db.bind_param(prep_stmt, 3, dob)
            ibm_db.bind_param(prep_stmt, 4, uname)
            ibm_db.bind_param(prep_stmt, 5, psw)
            ibm_db.execute(prep_stmt)

            insert_donor = "INSERT INTO
donor(Name,Username,Email,DOB,Availability) VALUES (?, ?, ?, ?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_donor)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, uname)
            ibm_db.bind_param(prep_stmt, 3, email)
```

```
            ibm_db.bind_param(prep_stmt, 4, dob)
            ibm_db.bind_param(prep_stmt, 5, "Not Available")
            ibm_db.execute(prep_stmt)

            sendmail(email,'Plasma donor App login',name, 'You are successfully Registered!')

            return redirect(url_for("login"))

    elif request.method == 'POST':
        msg = 'Please fill out the form !'
        flash(msg)
        return redirect(url_for("signup"))

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

@app.route('/requester')
def requester():
    if session['loggedin'] == True:
        return render_template('home.html')
    else:
        msg = 'Please login!'
        return render_template('login.html', msg = msg)

@app.route('/requested',methods=['POST'])
def requested():
```

```
    global value
    bloodgrp = request.form['bloodgrp']
    city = request.form['city']

    send_sql = "SELECT * FROM donor where BLOODTYPE = ? and CITY = ? and USERNAME != ? and AVAILABILITY = ?"
    prep_stmt = ibm_db.prepare(conn, send_sql)
    ibm_db.bind_param(prep_stmt, 1, bloodgrp)
    ibm_db.bind_param(prep_stmt, 2, city)
    ibm_db.bind_param(prep_stmt, 3, session['username'])
    ibm_db.bind_param(prep_stmt, 4, 'Available')
    ibm_db.execute(prep_stmt)
    row = ibm_db.fetch_assoc(prep_stmt)

    value = {}
    ind = 0
    while row != False:
        value[ind] = row
        ind += 1
        row = ibm_db.fetch_assoc(prep_stmt)
    print(value)

    return render_template("donorlist.html", value=value)

    # return render_template('home.html', pred="Your request is sent to the concerned people.")

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

```
def about():
    print(session["username"], session['id'])

    display_sql = "SELECT * FROM donor WHERE username = ?"
    prep_stmt = ibm_db.prepare(conn, display_sql)
    ibm_db.bind_param(prep_stmt, 1, session['id'])
    ibm_db.execute(prep_stmt)
    account = ibm_db.fetch_assoc(prep_stmt)
    print(account)
    donors = {}
    for values in account:
        if type(account[values]) == str:
            donors[values] = account[values].strip()
        else:
            donors[values] = account[values]
    print(donors)
    return render_template("about.html", account = donors)

@app.route('/sendEmail', methods = ["GET", "POST"])
def sendEmail():
    if request.method == 'POST':
        if request.form['select'] == 'select':
            email = request.form["Email"]
            uname = request.form['Username']
            curr_uname = session["username"]
            name = request.form['Name']
            select = "SELECT * from requests where Username = ? and
Requestuname = ?"
```

```
        stmt = ibm_db.prepare(conn, select)
        ibm_db.bind_param(stmt, 1, uname)
        ibm_db.bind_param(stmt, 2, curr_uname)
        ibm_db.execute(stmt)
        bool = ibm_db.fetch_assoc(stmt)

        print("boolean"+str(bool))
        if not bool:
            request_sql = "INSERT INTO requests VALUES (?, ?)"
            stmt = ibm_db.prepare(conn, request_sql)
            ibm_db.bind_param(stmt, 1, uname)
            ibm_db.bind_param(stmt, 2, curr_uname)
            ibm_db.execute(stmt)
            sendmail(email, 'Plasma donor App plasma request', name,'You have
received a request for Plasma Donation from a donee.')
        else:
            print(bool)
        print(email)
        print(name)
    return render_template("donorlist.html", value=value)


@app.route('/requests')
def requests():
    req_sql = "SELECT * From requests where Username = ?"
    stmt = ibm_db.prepare(conn, req_sql)
    ibm_db.bind_param(stmt, 1, session['username'])
    ibm_db.execute(stmt)
    req = ibm_db.fetch_assoc(stmt)
```

```
    print(req)
    print(session['username'])
    values = {}
    ind = 0
    while req != False:
        get_data = "Select * from donor where Username = ?"
        prep_stmt = ibm_db.prepare(conn, get_data)
        ibm_db.bind_param(prep_stmt, 1, req['REQUESTUNAME'])
        ibm_db.execute(prep_stmt)
        req1 = ibm_db.fetch_assoc(prep_stmt)
        values[ind] = req1
        ind += 1
        req = ibm_db.fetch_assoc(stmt)
    print(values)

    return render_template("requests.html", value=values)

@app.route('/details', methods = ['POST'])
def details():
    if request.method == 'POST':
        uname = request.form['uname']
        email = request.form['email']
        name = request.form['name']
        dob = request.form['dob']
        age = request.form['age']
        phone = request.form['phone']
        city = request.form['city']
```

```
        state = request.form['state']
        country = request.form['country']
        bloodtype = request.form['bloodtype']
        description = request.form['description']
        avail = request.form['avail']

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

        if account:
            update_sql = "UPDATE donor set Name=?, Username=?, Email=?,
DOB=?, Age=?, Phone=?, City=?, State=?, Country=?,
BloodType=?,Description=?,Availability=? where Username = ?"
            prep_stmt = ibm_db.prepare(conn, update_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, uname)
            ibm_db.bind_param(prep_stmt, 3, email)
            ibm_db.bind_param(prep_stmt, 4, dob)
            ibm_db.bind_param(prep_stmt, 5, age)
            ibm_db.bind_param(prep_stmt, 6, phone)
            ibm_db.bind_param(prep_stmt, 7, city)
            ibm_db.bind_param(prep_stmt, 8, state)
            ibm_db.bind_param(prep_stmt, 9, country)
            ibm_db.bind_param(prep_stmt, 10, bloodtype)
            ibm_db.bind_param(prep_stmt, 11, description)
```

```
            ibm_db.bind_param(prep_stmt, 12, avail)
            ibm_db.bind_param(prep_stmt, 13, uname)
            ibm_db.execute(prep_stmt)
            print("Update Success")
            return redirect(url_for("about"))
        else:
            insert_sql = "INSERT INTO donor VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?)"
            prep_stmt = ibm_db.prepare(conn, insert_sql)
            ibm_db.bind_param(prep_stmt, 1, name)
            ibm_db.bind_param(prep_stmt, 2, uname)
            ibm_db.bind_param(prep_stmt, 3, email)
            ibm_db.bind_param(prep_stmt, 4, dob)
            ibm_db.bind_param(prep_stmt, 5, age)
            ibm_db.bind_param(prep_stmt, 6, phone)
            ibm_db.bind_param(prep_stmt, 7, city)
            ibm_db.bind_param(prep_stmt, 8, state)
            ibm_db.bind_param(prep_stmt, 9, country)
            ibm_db.bind_param(prep_stmt, 10, bloodtype)
            ibm_db.bind_param(prep_stmt, 11, description)
            ibm_db.bind_param(prep_stmt, 12, avail)
            ibm_db.bind_param(prep_stmt, 13, False)

            ibm_db.execute(prep_stmt)
            print("Sucess")
            return redirect(url_for("about"))

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

```
def logout():
    session.pop('loggedin', None)
    session.pop('id', None)
    session.pop('username', None)
    return render_template('login.html')

if _name_ == '_main_':
    app.run(host='0.0.0.0',debug='TRUE')
```

GITHUB LINK

https://github.com/IBM-EPBL/IBM-Project-28833-1660117238

PROJECT DEMO LINK