PROJECT REPORT

Plasma Donor Application

Team ID :PNT2022TMID01682

Team Members :

- AJAY KOUSHIK K N
- NAVEEN PRASANTH M
- NITHARSHAN D J
- PRAVIN M

TABLE OF CONTENT

INTRODUCTION

1.1 PROJET OVERVIEW:

A way in which one can help the COVID 19 affected people is by donating Plasma from recovered patients. With no approved antiviral treatment plan for the deadly COVID-19 infection, plasma therapy is an experimental approach to treat COVID positive patients and help them recover faster. The therapy is considered to be safe and promising. If a particular person is fully recovered from COVID 19 he/she is eligible to donate their plasma. In the proposed system, donors who need to donate plasma can donate by uploading covid-19 certificate and blood bank can view donors and can raise requests to donors and the hospital can register/login and can search for plasma, they can raise requests to blood bank and can get the plasma.

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:

- To develop a system that provides functions to support donors to view and manage their information conveniently.
- To maintain records of plasma donors, plasma donation information and plasma stock in a centralized database system.

- To support searching, matching and requesting for blood convenient for administrators.
- To provide a function to send an email directly to the donor for their user account.

LITERATURE SURVEY

The world is suffering from the COVID 19 crisis and no vaccine has been found yet.. But there is another scientific way in which we can help reduce mortality or help people affected by COVID19 by donating plasma from recovered patients. In the absence of an approved antiviral treatment plan for a fatal COVID19 infection, plasma therapy is an experimental approach to treat COVID19-positive patients and help them recover faster recovery. Therapy is considered competent. In the recommendation system, the donor who wants to donate plasma can donate by uploading their COVID19 certificate and the blood bank can see the donors who have uploaded the certificate and they can make a request to the donor and the hospital can register/login and search for the necessary things. plasma from a blood bank and they can request a blood bank and obtain plasma from the blood bank.

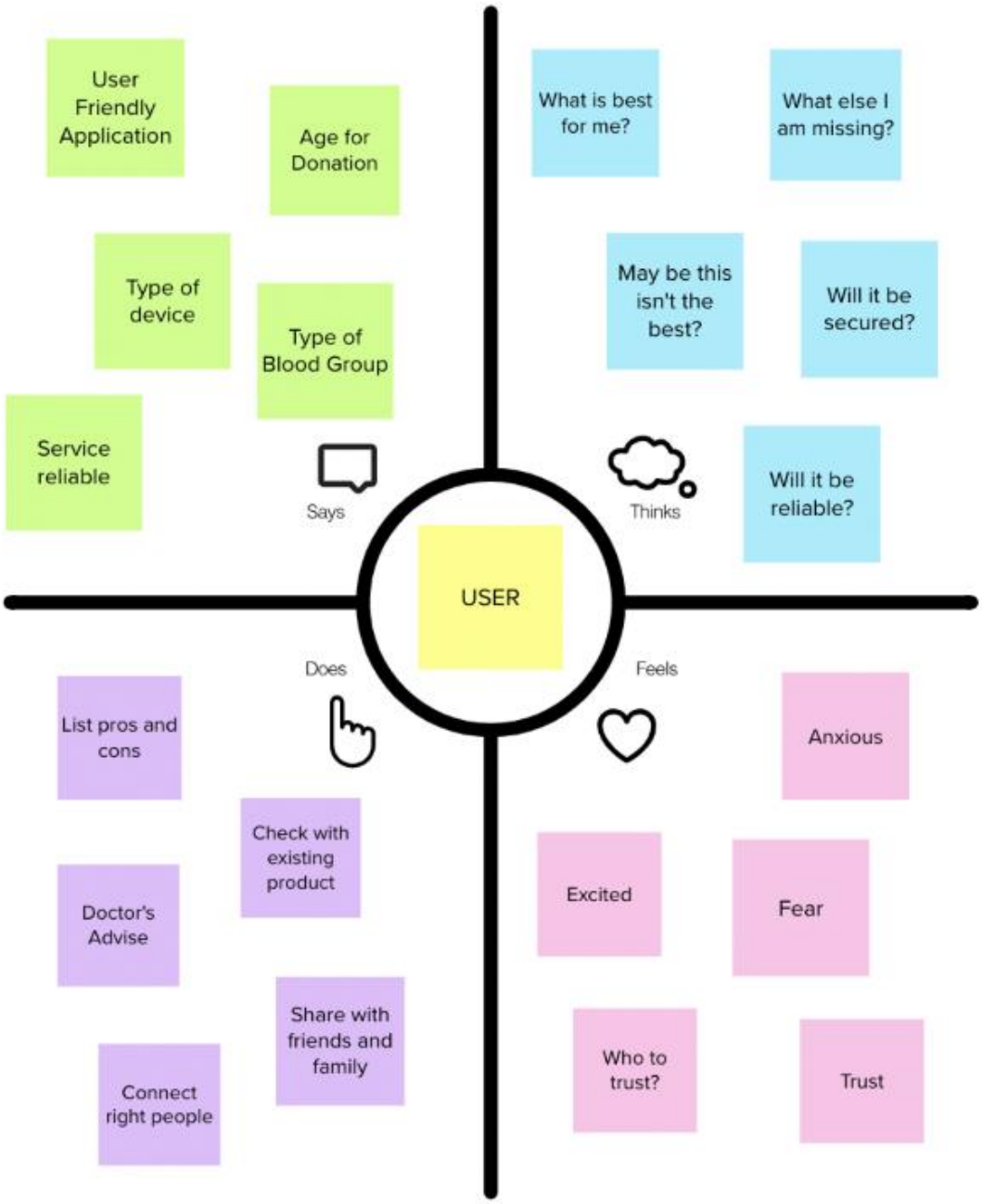
- Denuis O'Neil (1999). "Blood component" Archived from the original on June 5, 2013. Normally, certain amount of human body weight comes from blood. For adults, it is 4-6 litresof blood. This essential liquid plays an important role in transporting oxygen and nutrients to cells and removing carbon dioxide, ammonia and other waste products. Blood is a very common tissue composed of over 4000 different types of components.
- Ways to keep your plasma healthy, Original Archived November 1, 2013, Accessed November 11, 2011. Plasma donation is one of the most accepted practices for saving lives,While earning a few dollars. The whole process can take some time, but it's well worth it once you experience it a few times. Accepting money in exchange for plasma is welcome. It's a move when you feel like you're not just a hero, but you're adding value to yourself. The term "healthy" does not mean only in the absence of disease. It also means that you are healthy enough.

- Ripathis S, Kumar V, Prabhakar A, Joshi S, Agarwal A (2015). "Microscale Passive Plasma Separation: A Review of Design Principles and Microdevices," J. Micromech Micro 25 (8): 083001; Plasma separation is of great importance in the fields of diagnosis and healthcare. Due to the lagging transition to microscale, these recent trends are a rapid shift towards shrinking complex macro processes.
- Published in International Research Journal of Modernization in Engineering Technology and Science - e-ISSN: 2582-5208.

IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP:

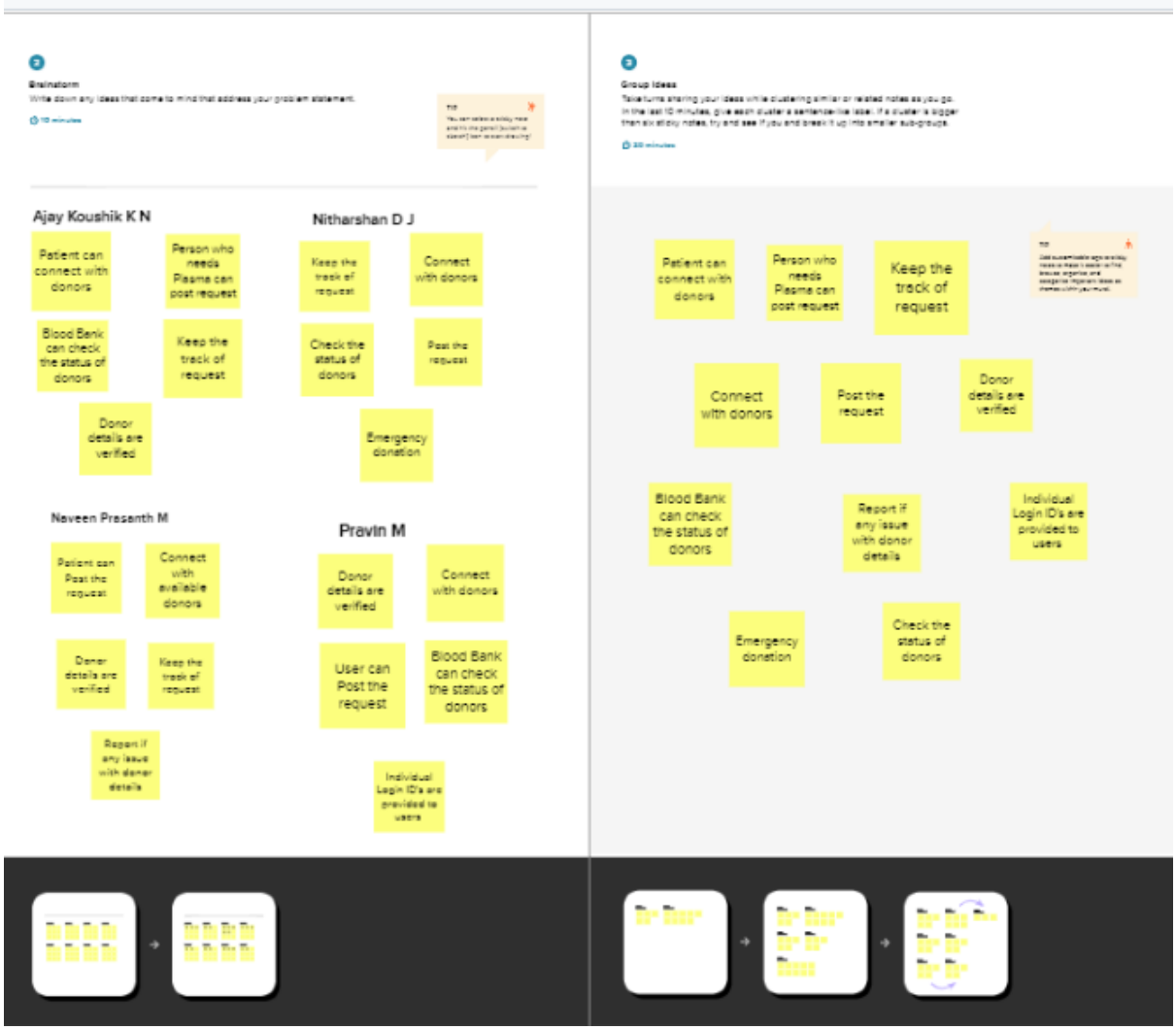
An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise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



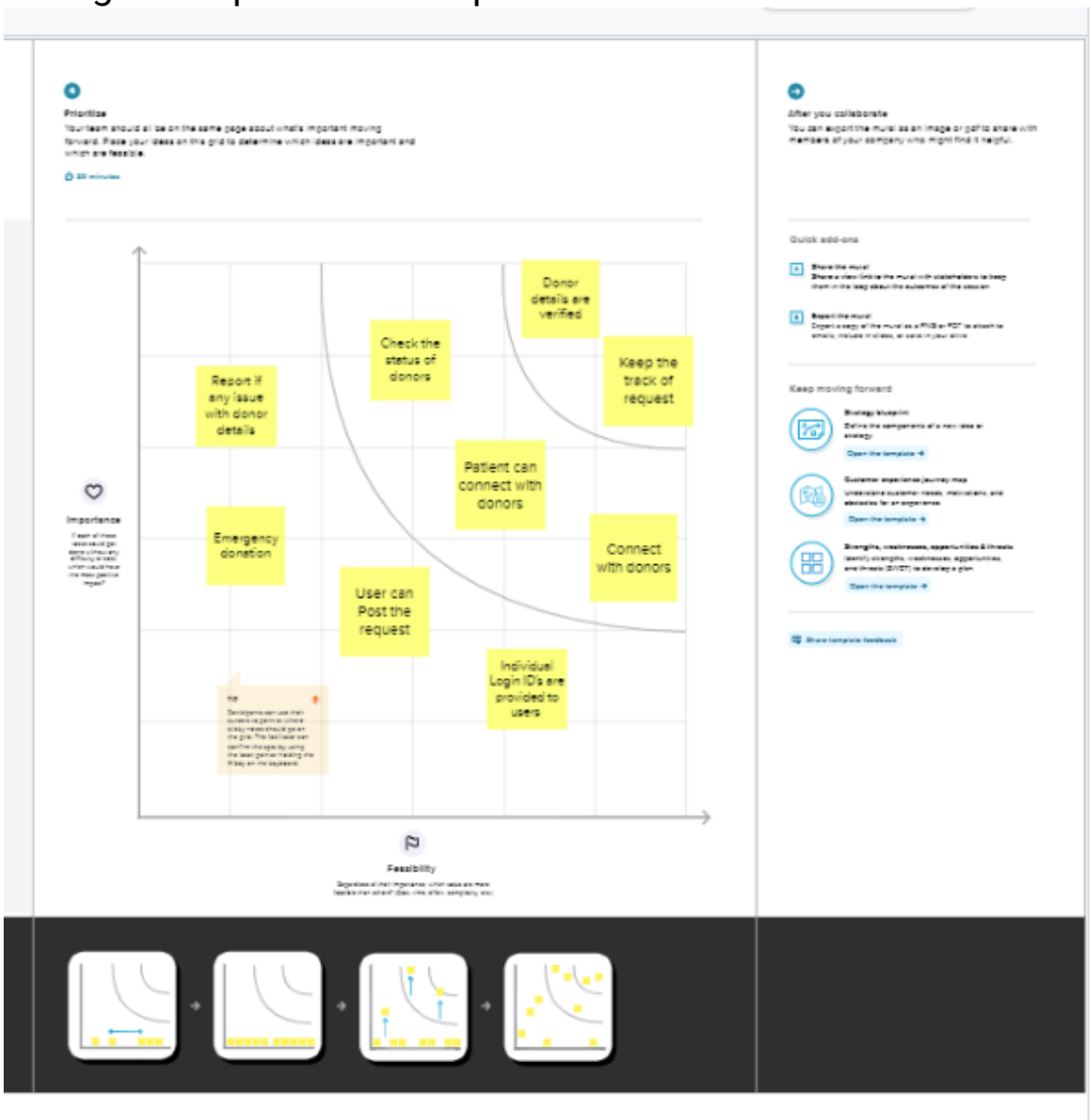
3.2 IDEATION AND BRAINSTORMING:

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.

Ideation is a creative process where designers generate ideas in sessions. The participants gather with open minds to produce as many ideas as they can to address a problem statement in a facilitated, judgment-free environment.



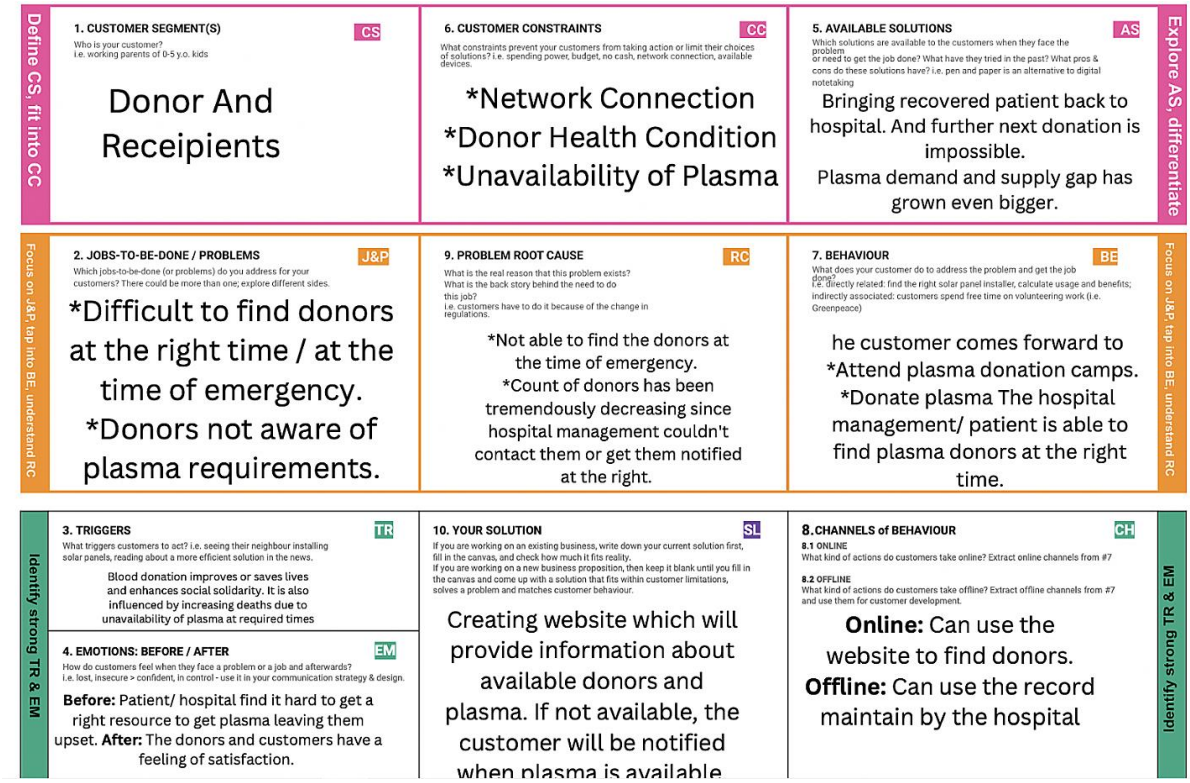
The ideas gathered in ideation phase are then gathered, and given a priority ranking, so as to select the high priority ideas during the implementation phase.



3.3 PROPOSED SOLUTION:

Our unified platform is aimed at all donors and requesters so they can communicate easily. Donors can be registered, verified and listed on the platform database. Patients or people in urgent need of plasma can use their location and other attributes to find a compatible plasma donor match right in time. Although this benefits the donors by offering them a universal location to list themselves, it poses a more significant impact to the patients who can find the resources they need in a way smaller timeframe. Patients and requesters don't have to be mentally pressured and frustrated about being able to find a compatible donor before it is too late.

3.4 PROBLEM SOLUTION FIT



REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENT:

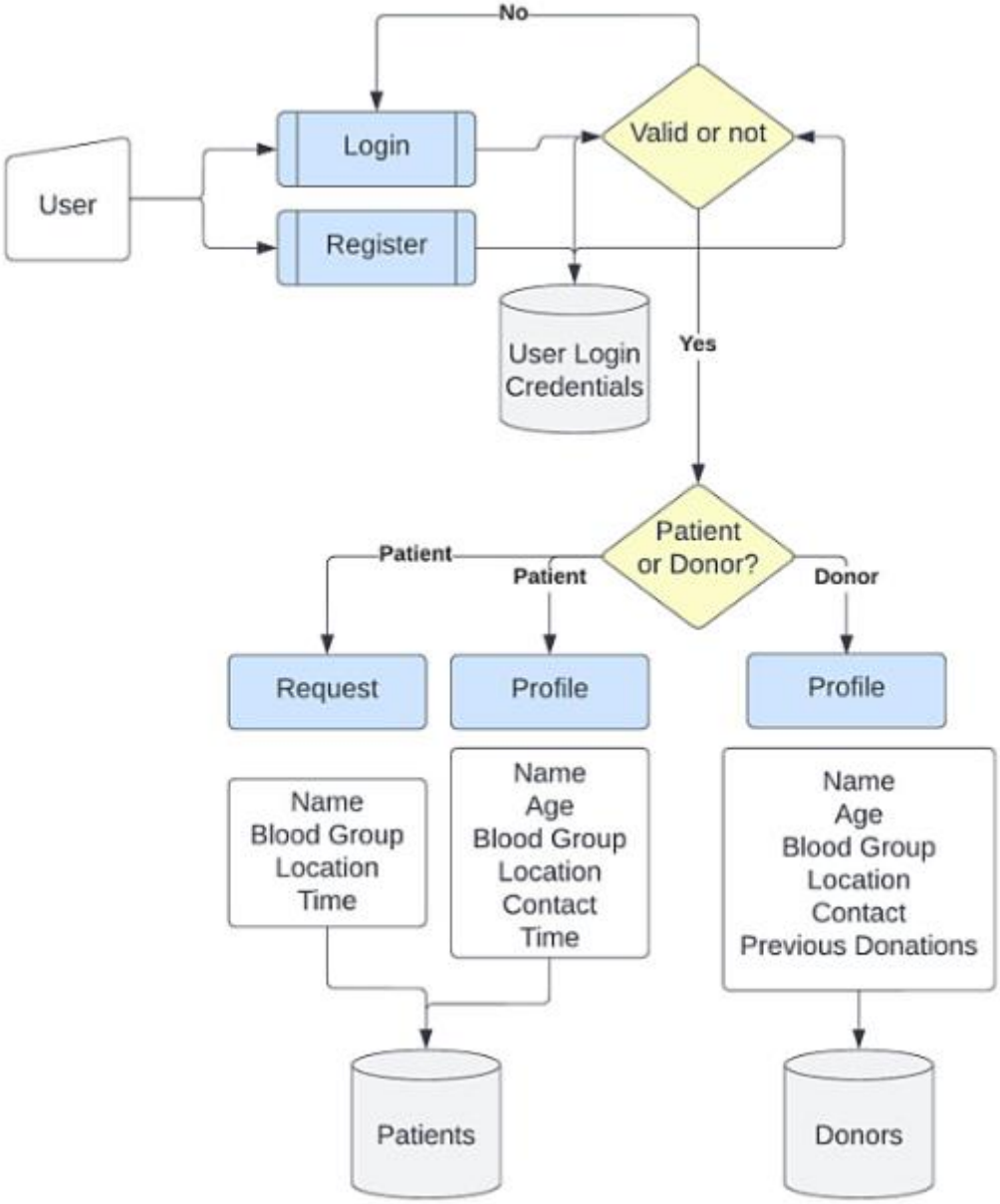
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	**User Registration**	Registration through Mobile Number Registration through Gmail
FR-2	**User Registration**	Confirmation via Email Confirmation via OTP
FR-3	**User Validation**	Donor – Check health conditions Patient – Check credibility of user
FR-4	**Chatbot**	Help the user understand the process and navigate the website
FR-5	**Search**	Patient enters details to search for compatible donors
FR-6	**Request**	Patient can send a donor request to obtain their plasma
FR-7	**Email notification**	Donor will get notified through email when a compatible patient places request
FR-8	**Donation completion**	Patient and Donor to confirm completion of donation

4.2 NON FUNCTIONAL REQUIREMENT:

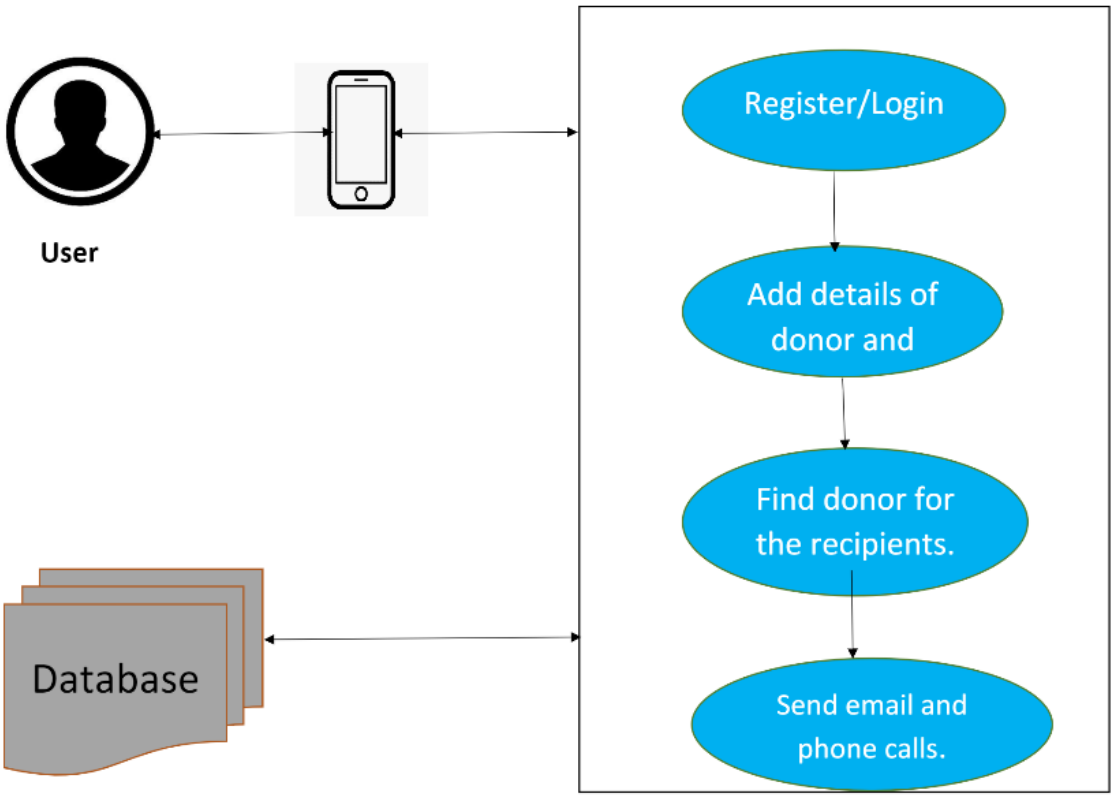
NFR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The user experience must be simple and the user must be able to perform all actions available on the platform
NFR-2	**Security**	The database layer and the logic layer are hosted using IBM services, hence the security of all data involved in the process is kept secure
NFR-3	**Reliability**	All processes involved in the platform must be uniform whenever accessed. It must also function without any bugs and errors
NFR-4	**Performance**	Immediate feedback from the platform is provided to user, so they are not discouraged to use the application
NFR-5	**Availability**	The application must be usable at all time, the database and servers need to be available and reachable anytime from anywhere
NFR-6	**Scalability**	The platform must adapt so that it can support a high volume of concurrent users. Meanwhile, the platform must also be loosely couples to ensure it can scale vertically too, by adding more functionality

PROJECT DESIGN

5.1 DATA FLOW DIAGRAM:



5.2 SOLUTION AND TECHNICAL ARCHITECTURE:



5.3 USER STORIES:

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a donor, I can register for the application by entering my email /Phone number, password, and confirming my password.	4	High	Ajay Koushik KN (Leader)
Sprint-1	Login	USN-2	Registered donor can log into the application by entering donor email & password	3	High	Pravin M (Member 3)
Sprint-2	Verification	USN-3	As a donor, I can enter my details to check the donor eligibility criteria,	10	Medium	Naveen Prasanth M (Member 2)
Sprint-3	Dashboard	USN-4	User can provide their personal details and location	7	Low	Nitharshan DJ (Member 1)
Sprint-4	Acceptance	USN-5	User can accept their willingness to donate plasma	10	Medium	Ajay Koushik KN (Leader)

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a receiver, I can register for the application by entering my email /Phone number, password, and confirming my password	4	High	Pravin M (Member 3)
Sprint-1	Login	USN-2	Registered receiver can log into the application by entering receiver email & password	3	High	Naveen Prasanth M (Member 2)
Sprint-2	Verification	USN-3	As a receiver, I can enter my details to check the receiver eligibility criteria	10	Medium	Nitharshan DJ (Member 1)
Sprint-3	Dashboard	USN-4	User can search the list of available donor	7	Low	Ajay Koushik KN (Leader)
Sprint-4	Access	USN-5	User can access the available donors list ,then they can choose the donor who is nearby to receiver	10	Medium	Pravin M (Member 3)
Sprint-1	Registration	USN-1	Third Party user can register for the application by entering my email /Phone number, password, and confirming my password.	3	High	Naveen Prasanth M (Member 2)
Sprint-1	Login	USN-2	Registered user can log into the application by entering user email & password	3	High	Nitharshan DJ (Member 1)
Sprint-3	Query System	USN-3	User can ask their queries via Chabot which is available 24/7 to sort user issues	6	Medium	Ajay Koushik KN (Leader)

PROJECT PLANNING & SCHEDULING

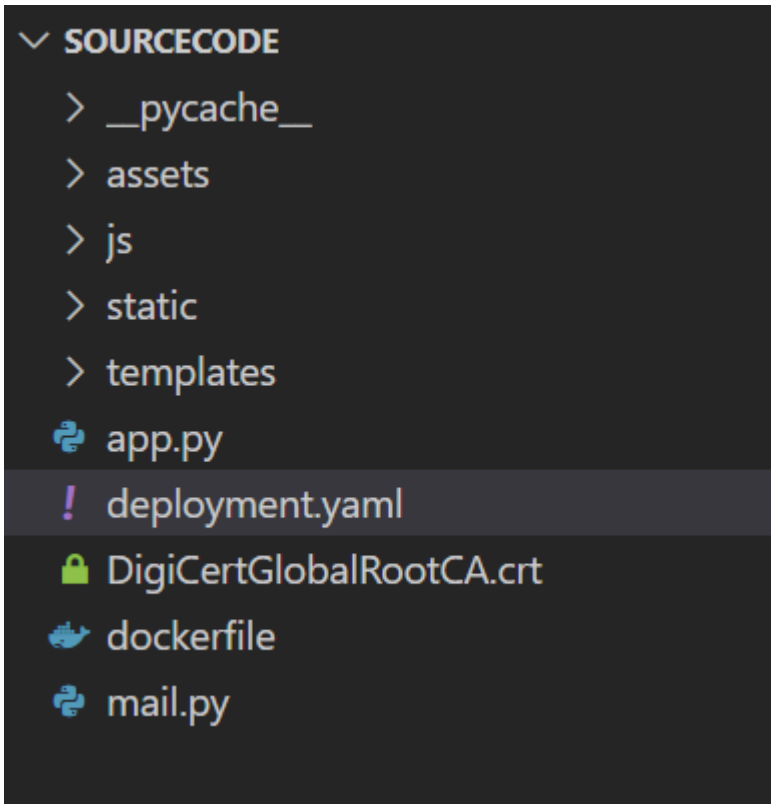
6.1 SPRINT PLANNING & ESTIMATION:

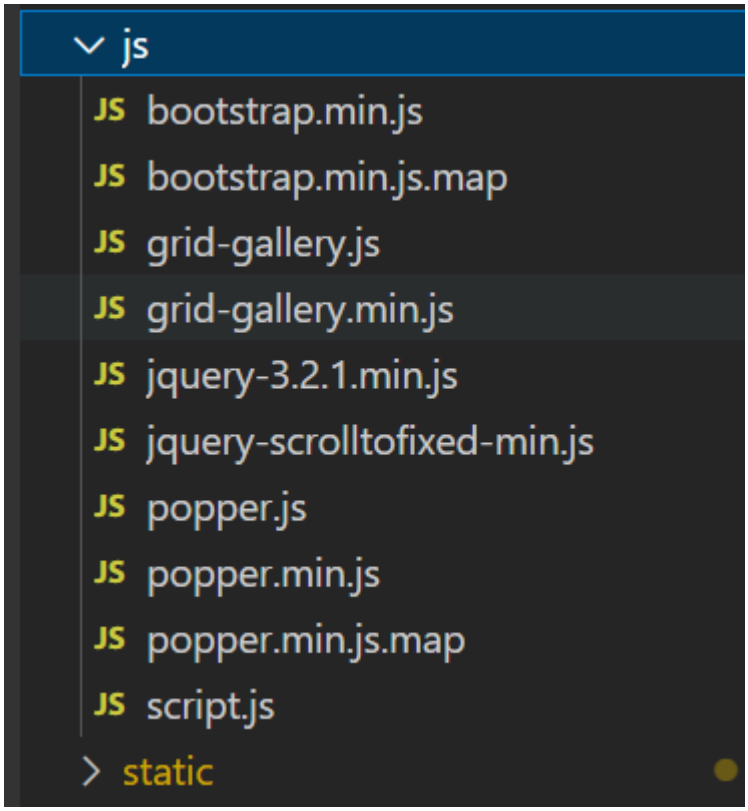
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022

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

CODING AND SOLUTIONING





static

- adminlogin.css
- adminreg.css
- animate.css
- bootstrap.min.css
- donar.css
- fontawsom-all.min.css
- grid-gallery.css
- grid-gallery.min.css
- logincss.css
- recipient.css
- reciptreg.css
- style.css

templates

- admin.html
- adminlogin.html
- adminreg.html
- donar.html
- donlogin.html
- donregistration.html
- index.html
- mail.html
- plasmadon.html
- plasmareq.html
- recipient.html
- recipregistration.html
- reclogin.html

app.py > ...

```python
from flask import Flask,render_template,request,url_for,redirect
from flask_mail import *
from markupsafe import escape

import ibm_db
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=2f3279a5-73d1-4859-88f0-a6c3e6b4b907.c3n41cm


app = Flask(__name__)


@app.route('/')
def index():
    return render_template('index.html')      # index - home page

# admin credentials

@app.route('/adminlogin')
def adminlogin():
  return render_template('adminlogin.html')   # admin log in page

@app.route('/adminreg')
def adminreg():
  return render_template('adminreg.html')  # admin sign up page

```

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

    fname = request.form['fname']
    lname = request.form['lname']
    dob = request.form['dob']
    email = request.form['email']
    mnumb = request.form['mnumb']
    gender = request.form['gender']
    address = request.form['address']
    pin = request.form['pin']

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

```

```
    if account:
        return render_template('reclogin.html', msg="Already your account exists, please try to log in")
    else:
     insert_sql = "INSERT INTO recipientrec VALUES (?,?,?,?,?,?,?,?)"
     prep_stmt = ibm_db.prepare(conn, insert_sql)
     ibm_db.bind_param(prep_stmt, 1, fname)
     ibm_db.bind_param(prep_stmt, 2, lname)
     ibm_db.bind_param(prep_stmt, 3, dob)
     ibm_db.bind_param(prep_stmt, 4, email)
     ibm_db.bind_param(prep_stmt, 5, mnumb)
     ibm_db.bind_param(prep_stmt, 6, gender)
     ibm_db.bind_param(prep_stmt, 7, address)
     ibm_db.bind_param(prep_stmt, 8, pin)
     ibm_db.execute(prep_stmt)

 return render_template('reclogin.html', msg="Account has been created successfully..")

 return "success..."

```

```
@app.route('/recipregistration')
def recipregistration():
  return render_template('recipregistration.html')   ## recipient signup page uh

@app.route('/recipientlogin')
def recipientlogin():
  return render_template('reclogin.html')      ## recipt login page


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

    fname = request.form['fname']
    lname = request.form['lname']
    dob = request.form['dob']
    email = request.form['email']
    mnumb = request.form['mnumb']
    gender = request.form['gender']
    address = request.form['address']
    pin = request.form['pin']

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

```

```
    if account:
        return render_template('reclogin.html', msg="Already your account exists, please try to log in")
    else:
     insert_sql = "INSERT INTO recipientrec VALUES (?,?,?,?,?,?,?,?)"
     prep_stmt = ibm_db.prepare(conn, insert_sql)
     ibm_db.bind_param(prep_stmt, 1, fname)
     ibm_db.bind_param(prep_stmt, 2, lname)
     ibm_db.bind_param(prep_stmt, 3, dob)
     ibm_db.bind_param(prep_stmt, 4, email)
     ibm_db.bind_param(prep_stmt, 5, mnumb)
     ibm_db.bind_param(prep_stmt, 6, gender)
     ibm_db.bind_param(prep_stmt, 7, address)
     ibm_db.bind_param(prep_stmt, 8, pin)
     ibm_db.execute(prep_stmt)

  return render_template('reclogin.html', msg="Account has been created successfully..")

  return "success..."

```

```

@app.route('/donregistration')
def donregistration():
    return render_template('donregistration.html')   ## donar signup page uh


@app.route('/donarlogin')
def donarlogin():
    return render_template('donlogin.html')      ## donar login page

```

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

        fname = request.form['fname']
        lname = request.form['lname']
        dob = request.form['dob']
        email = request.form['email']
        mnumb = request.form['mnumb']
        gender = request.form['gender']
        address = request.form['address']
        pin = request.form['pin']

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

```

```python
        if account:
            return render_template('donlogin.html', msg="Already your account exists, please try to log in")
        else:
        insert_sql = "INSERT INTO donarrec VALUES (?,?,?,?,?,?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, fname)
        ibm_db.bind_param(prep_stmt, 2, lname)
        ibm_db.bind_param(prep_stmt, 3, dob)
        ibm_db.bind_param(prep_stmt, 4, email)
        ibm_db.bind_param(prep_stmt, 5, mnumb)
        ibm_db.bind_param(prep_stmt, 6, gender)
        ibm_db.bind_param(prep_stmt, 7, address)
        ibm_db.bind_param(prep_stmt, 8, pin)
        ibm_db.execute(prep_stmt)

    return render_template('donlogin.html', msg="Account has been created successfully..")

    return "success..."
```

```python
@app.route('/admin')
def admin():
    return render_template('admin.html')

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


## donar registering for donation
@app.route('/giveplasma',methods = ['POST', 'GET'])
def giveplasma():
    if request.method == 'POST':
```

```python
    name = request.form['name']
    age = request.form['age']
    gender = request.form['gender']
    mnumb = request.form['mnumb']
    email = request.form['email']
    city = request.form['city']
    address = request.form['address']
    bloodgroup = request.form['bloodgroup']
    issue = request.form['issue']
    lastbd = request.form['lastbd']
    slot = request.form['slot']

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

```python
    if account:
      return render_template('donlogin.html', msg="You are already a member, please login usir
    else:
      insert_sql = "INSERT INTO donar VALUES (?,?,?,?,?,?,?,?,?,?,?)"
      prep_stmt = ibm_db.prepare(conn, insert_sql)
      ibm_db.bind_param(prep_stmt, 1, name)
      ibm_db.bind_param(prep_stmt, 2, age)
      ibm_db.bind_param(prep_stmt, 3, gender)
      ibm_db.bind_param(prep_stmt, 4, mnumb)
      ibm_db.bind_param(prep_stmt, 5, email)
      ibm_db.bind_param(prep_stmt, 6, city)
      ibm_db.bind_param(prep_stmt, 7, address)
      ibm_db.bind_param(prep_stmt, 8, bloodgroup)
      ibm_db.bind_param(prep_stmt, 9, issue)
      ibm_db.bind_param(prep_stmt, 10, lastbd)
      ibm_db.bind_param(prep_stmt, 11, slot)
      ibm_db.execute(prep_stmt)
```

```python
@app.route('/plasmadon')
def plasmadon():
  donar = []
  sql = "SELECT * FROM donar"
  stmt = ibm_db.exec_immediate(conn, sql)
  dictionary = ibm_db.fetch_both(stmt)
  while dictionary != False:
    # print ("The Name is : ",  dictionary)
    donar.append(dictionary)
    dictionary = ibm_db.fetch_both(stmt)

  if donar:
    return render_template("plasmadon.html", donar = donar)
```

```python
@app.route('/plasmareq')
def plasmareq():
    recipient = []
    sql = "SELECT * FROM recipient"
    stmt = ibm_db.exec_immediate(conn, sql)
    dictionary = ibm_db.fetch_both(stmt)
    while dictionary != False:
        # print ("The Name is : ",  dictionary)
        recipient.append(dictionary)
        dictionary = ibm_db.fetch_both(stmt)

    if recipient:
        return render_template("plasmareq.html", recipient = recipient)

```

```python
@app.route('/delete/<name>')
def deleted(name):
    sql = f"SELECT * FROM recipient WHERE name='{escape(name)}'"
    print(sql)
    stmt = ibm_db.exec_immediate(conn, sql)
    recipient = ibm_db.fetch_row(stmt)
    print ("The Name is : ",  recipient)
    if recipient:
        sql = f"DELETE FROM recipient WHERE name='{escape(name)}'"
        print(sql)
        stmt = ibm_db.exec_immediate(conn, sql)

        recipient = []
        sql = "SELECT * FROM recipient"
        stmt = ibm_db.exec_immediate(conn, sql)
        dictionary = ibm_db.fetch_both(stmt)
        while dictionary != False:
            recipient.append(dictionary)
            dictionary = ibm_db.fetch_both(stmt)
        if recipient:
            return render_template("plasmareq.html", recipient = recipient, msg="Accepted successfully")

        return "Accepted Successfully"


if __name__ == "__main__":
    app.run(debug=True)
```

TESTING

Testing was managed on a platform called Testlodge. This website helps us create test suites and individual test cases for accounting purposes. We can describe individual test cases, their pass and fail criteria. These test cases can be then assigned to individual team mebers for them to carry out the test and report the result on the platform. We performed unit testing, integration testing and acceptance testing on our final application. Unit testing was done by creating testcases for individual modules to work in separation from the rest of the application. Each of the html pages, their scripting and their input and output was tested. Integration testing was performed to check if a complete logical process can be done without any errors in the workflow. As some of our functions and features work together, they need to be assessed if the testcases pass.

8.1 TEST CASES:

Database: Entering invalid data into html forms should be prevented by using appropriate scripting to validate data. Retrieval of data should be accurate for mission critical functions.
Login: Enter invalid credentials to test if login is allowed.
Registration: Date of negative covid test must be at least 14 days before current date.

8.2 ACCEPTANCE TESTING:

Acceptance testing is testing the overall service flow and checking if each logical function works with all ranges of input from the perspective of an end user. This is similar to blackbox testing, as the user doesn't know the internal functioning of the application. They only provide their input and we need to validate if the necessary output has been acquired as a result.

RESULT

9.1 SOFTWARE TEST METRICS

1. Test design efficiency = Number of tests designed /Total time

- 25 Tests designed / 10 Hours = 2.5 Tests designed per hour

2. Passed Test Cases Percentage = (Number of Passed Tests/Total number of tests executed) X 100

- 25 passed Test cases / 25 Test cases x 100 = 100%

3. Failed Test Cases Percentage = (Number of Failed Tests/Total number of tests executed) X 100

- 0 failed Test cases / 25 Test cases x 100 = 0%

4. Schedule slippage = (Actual end date – Estimated end date) / (Planned End Date – Planned Start Date) X 100

- (19/11/2022 - 17/11/2022) / (17/11/2022 - 10/09/2022) x 100 = 3 days / 70 days x 100 = 4.28%

9.2 PERFORMANCE METRICS

Our project uses the IBM Lite plan for all the cloud services like DB2, Watson Assistant, Container Registry, Kuberenetes which provides minimal access and performance for free of cost. There are performance constraints for the deployed application. All requests by the user must be processed and rendered on screen before 5 seconds of initiation. If this is not possible due to errors from any side, a buffering screen must be displayed. Database needs to be online at all times to facilitate information transfer whenever a user logs on and accesses the platform.

ADVANTAGES & DISADVANTAGES

The vision of this project is to provide a unified platform where all the requests and information can be shared for higher visibility and faster correspondence. By using an application to serve both donors and patients who are in need of plasma, we are bridging the gap in between them. Donors can register themselves so they may be contacted when a certain request is placed. Patients can also place a request on the platform so their needs can be fulfilled as soon as possible.

Disadvantages of this platform include a transparency about sensitive information provided by both patients and donors. Administrators who are responsible for managing requests would be able to see information regarding the users health. Verification also needs to be implemented to make sure that the credibility of both patients and donors are upheld, to increase the reliability of the platform for public usage.

CONCLUSION

In conclusion, we have created an application and provided an interface with which donors can contact hospitals and hospitals can in turn find donors to meet the plasma requirements. This will allow hospitals to meet the demands of plasma for the patients undergoing treatment. Plasma from Covid recovered patients can be used to treat covid patients. This allows an easy way for donors to interact with hospitals and to donate plasma when needed.

Donors who need to donate plasma can donate by uploading their details and blood bank can view donors and can raise requests to donors and the hospital can register/login and can search for plasma, they can raise requests to blood bank and can get the plasma.

People need not have feelings of anxiety and fear that they might not receive the help they seek. Using our platform they can gather information about the donors available near them and compatible so the donation can be done.

Such an application will be beneficial to the public who currently don't have a common and dedicated platform to share such emergencies and contact details.

FUTURE SCOPE

There is scope for more features to be integrated in our application for a better user experience and more efficient process.

A messenger service can be included on the platform for the donors and patients to enable direct communication without the need for a medium.

Verification using any photo ID can be done as to automate the process by integrating government service to increase the credibility and reliability of the application.

A ticketing system can be used to collect, track and manage user queries and problems they face during the usage of the application.

Although the app is already cross platform supported due to it being a website, a mobile application can be designed and published for users to have instant access to our application on their smartphones.

APPENDIX

Github link:
https://github.com/IBM-EPBL/IBM-Project-18807-1659690339

Demo link:

https://drive.google.com/file/d/1cvk5umtO4UOPXVJ-8Tv1LCTo2YsurTVu/view?usp=share_link