PLASMA DONOR APPLICATION

(CLOUD APPLICATION)

Submitted By

Raagavan J S (Team Leader) - 917719IT127

Ashok Kumar S S - 917719IT121

Madhusudhanan C B - 917719IT124

Prasanna S - 917719IT126

PROJECT ID: PNT2022TMID21590

Department of Information Technology

Thiagarajar College of Engineering, Madurai – 625015

Institution Mentor: Dr. P. Karthikeyan

Industrial Mentor: Navya

TABLE OF CONTENTS

1. INTRODCTION

1.1. Project Overview

In the modern world finding a donor is considered as more complex thing so, we came up with new concept which aligns the existing problem with our new solution. The complete view of the project is to provide plasma using plasma donor application, that the plasma donor application which will be helpful for the healthcare unit. That the application finds the perfect donors and get the plasma in time.

1.2. Purpose

The main goal of creating a system for plasma donation by using cloud development is to help the patients who are in need of plasmas. It helps to fetch the donors and getting lifesaving plasma. The plasma directory feature shows the details briefly about the donated donors

2. LITERATURE SURVEY

2.1. Existing Solutions

BLOODR: BLOOD DONOR AND REQUESTER MOBILE APPLICATION

With rapid increase in the usage of social networks sites across the world, there is also a steady increase in blood donation requests as being noticed in the number of posts on these sites such as Facebook and twitter seeking blood donors. Finding blood donor is a challenging issue in every country.

INSTANT PLASMA DONOR RECIPIENTCONNECTOR WEB APPLICATION

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 faster recovery.

PLASMA DELIVERY FOR BIOMEDICAL APPLICATIONS

Plasma's unique properties and interactions with other states of matter offer many promising opportunities for investigation and discovery. These biomedical applications include sterilization, wound healing, blood coagulation, oral/dental diseases treatment, cancer therapy, and immunotherapy. Effective delivery of plasma constituents is critical for these applications.

2.2. References

https://pubmed.ncbi.nlm.nih.gov/29184892/

https://www.irjmets.com/uploadedfiles/paper//issue_6_june_2022/26076/final/fin_irjmets1655361213.pdf

https://www.sciencedirect.com/science/article/abs/pii/S1369702122000608?casa_token=DlUbcXWRg9QAAAAA:1DlmCITBzx4sQ0z83momAYAdB6amBI1JQH9KiWlNHHc0u9oA4LckodJMbmDL-1p_P3wVgqKVKpaR

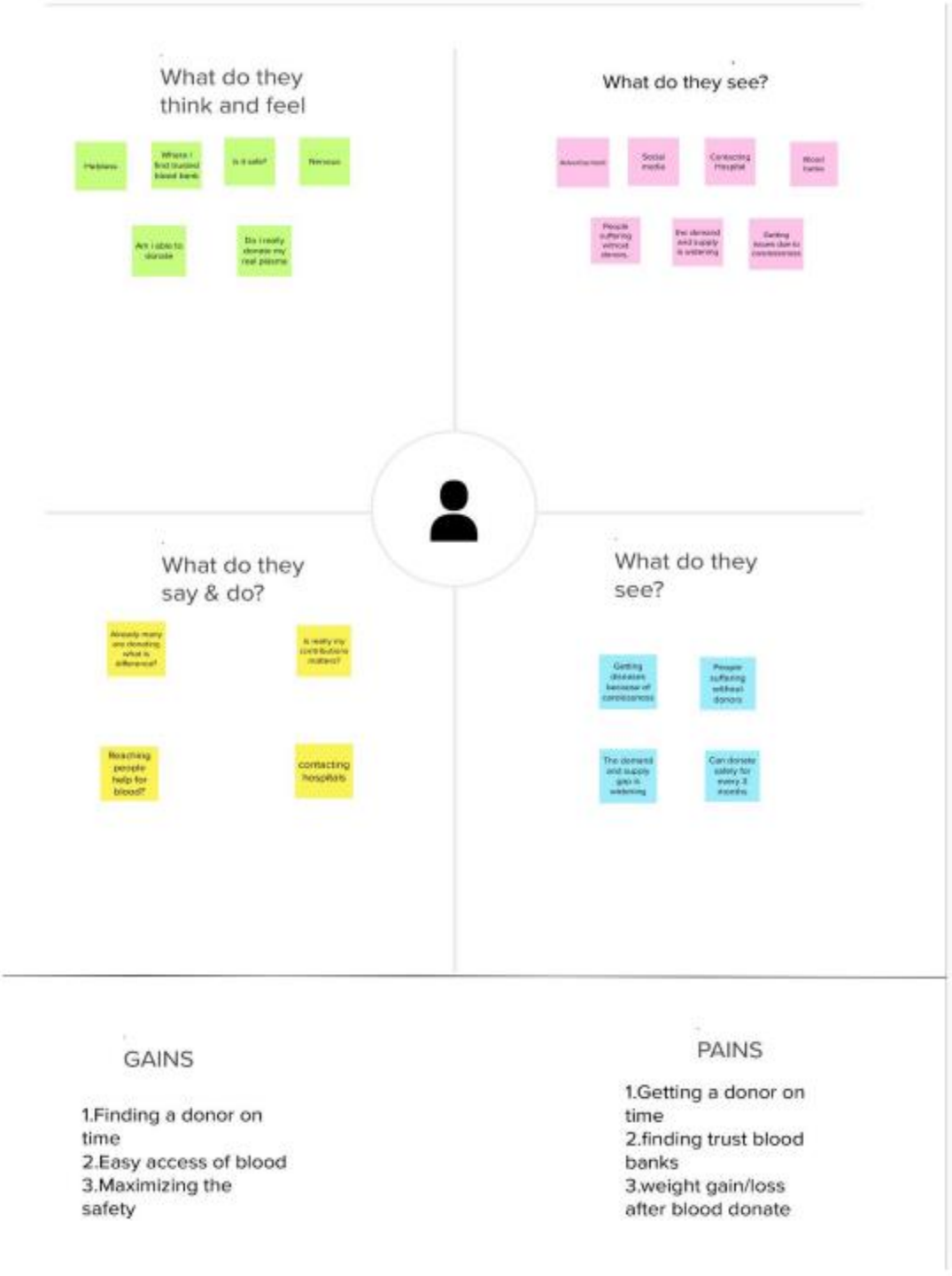
2.3. Problem Statement Definition

During the COVID 19 situation, plasma became a high, and the plasma donor has count become low. saving donor information and helping to the patient notifying the current donors list, would be a helping hand. the problem faced during the donor information, haver application help to the patient to find the correct donor details in application. we going to develop the models like donor details, store the blood according to donor blood and get the inform from the request reach from the donor request.

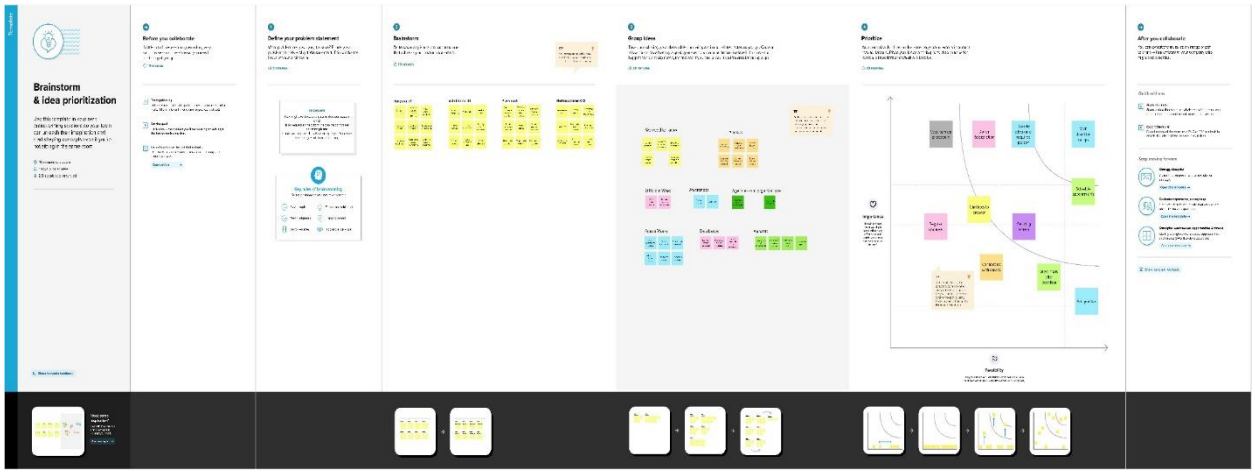
A plasma is a liquid portion of the blood, over 55% of human blood is plasma. Plasma is used to treat various infectious diseases and it is one of the oldest methods known as plasma therapy. Plasma therapy is a process where blood is donated by recovered patients to establish antibodies that fights the infection. For instance, during COVID 19 crisis the requirement for plasma increased drastically as there were no vaccination found to treat the infected patients, with plasma therapy the recovery rates where high but the donor count was extremely low and, in such situations, it was 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.

3. IDEATION AND PROPOSED SOLUTION

3.1. Empathy Map Canvas



3.2. Ideation and Brainstorming



3.3. Proposed Solution

Sl. No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	• When the details are maintained manually, it is complicated for donors and patients. • Physical Man power is required to manage the Data and process the Plasma Donation. • In Pandemic situation, it is difficult to have manpower, so it is difficult to get the Plasma donor data. • Needed an Automated system to Manage donor and Patient data. • The data is needed to be accessible from anywhere and anytime
2.	Idea / Solution description	• Making a Web application which is user friendly as well as has more features for serving the people better. • Reduced workload by storing the details in cloud storage. • No Manpower / Remote Manpower only will be needed. • Data Availability for 24x7x365.

3.	Novelty / Uniqueness	• User friendly UI to access the web application by all the people • If a Donating user is available, they can request for plasma. • The web application will automatically send the email containing the Patient's contact details. • The Donor may contact the Patient and can reach the patient to donate the blood. • Voluntary donors can fill out a registration form and can get the Request Email on demand.
4.	Social Impact / Customer Satisfaction	• Find the donors in near places • Connect the donors and patients Easily. • With all of the authenticated information, this platform will assist the public in donating or obtaining their plasma needs.
5.	Business Model (Revenue Model)	By collaborating with government and organizing Plasma Donation Camps and store them instead requesting Plasma on demand.
6.	Scalability of the Solution	• The aim is to build a web application using Cloud with advanced features that will help to overcome the barrier between Plasma bank, Donor and Patient • Since the project uses IBM DB2 database it can handle with multiple requests in various regions • As this is a web application and uses cloud storage, any further enhancements in technology can be incorporated within this application. • Chatbot for Queries • Genuineness of the Patient will be tested

3.4. Problem Solution Fit

<table>
<tr>
<td>1.CUSTOMER SEGMENTS(S)

• Donors
• Patient
• Hospitals</td>
<td>5. AVAILABLE SOLUTIONS

The existing application used only collecting details pf donors but it does not notify them at a right time.
Our solution is building a website that notifies the donors at a right time.</td>
<td>8. CHANNELS OF BEHAVIOR

ONLINE:
Can use the website to find donors
OFFLINE:
Can use the record maintain by the hospital</td>
</tr>
<tr>
<td>2.JOB-TO-BE-DONE

• Difficult to find donors at the right time
• Donors not aware of plasma requirements</td>
<td>6.CUSTOMER CONSTRAINTS

• Regular interval connection
• Donor health condition
• Unavailability of plasma</td>
<td>9.PROBLEM ROOT CAUSE
• Not able to find donors at the right time of emergency
• Count of donors has been tremendously decreasing since hospital management couldn’t contact them</td>
</tr>
<tr>
<td>3.TRIGGERS
Blood donation improves of saves lives and enhances social solidarity. It is also influenced by increasing deaths due to unavailability of plasma at required times.</td>
<td rowspan="2">7.BEHAVIOUR
The customer comes forward to
• Attend plasma donation camps
• Donate plasma</td>
<td rowspan="2">10.YOUR SOLUTION

Creating website which will provide information about the available donors and plasma. If not available the customer will be notified when plasma is available.</td>
</tr>
<tr>
<td>4.EMOTIONS:
Before:
Patient /Hospital find it hard to get a right resource to get A plasma leaving them upset.
After:
The donors and customers haves a feeling of satisfaction</td>
</tr>
</table>

4. REQIREMENT ANALYSIS

4.1. Functional Requirements

FR. No.	Functional Requirement	Sub Requirement
FR 1	User Registration and Login	User Registration and Login access through Website
FR 2	Request Plasma	Patient should fill the necessary details in forms.
FR 3	Request Notification	The Donor should get notified about the Plasma request via email.
FR 4	Profile Updation	The Donor should fill the profile completely so as to get notification in case of Plasma Request.

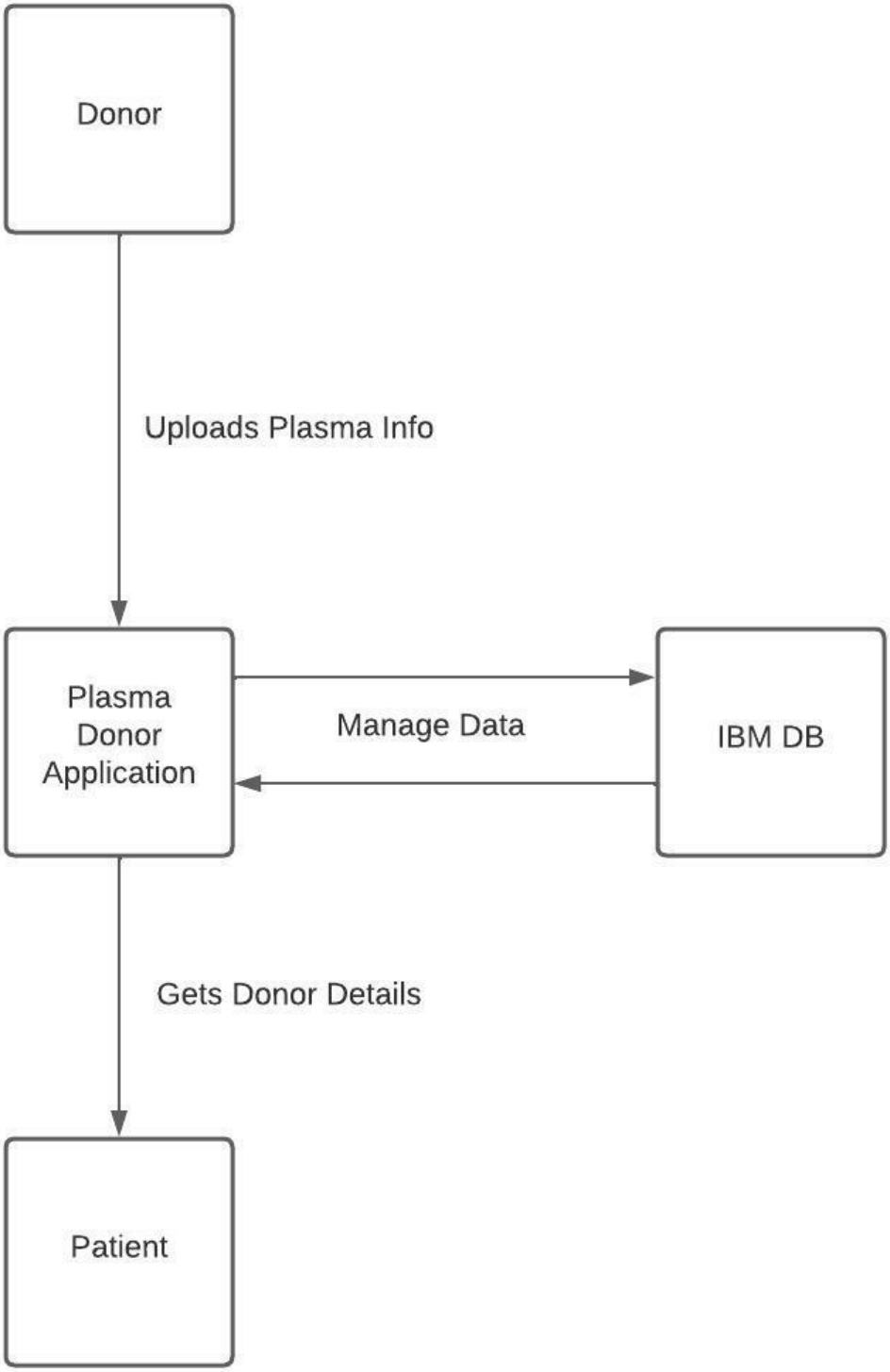
4.2. Non-Functional Requirements

NFR No.	Non-Functional Requirement	Description
NFR-1	Usability	The usability of the website is to make all users will be satisfied with our requirements of the plasma Details.
NFR-2	Security	The security system of the project is to develop the website that prevents SQL injection attacks attack and DOS attack
NFR-3	Reliability	The reliability of the system is to make sure the website does not go offline. The users can be reach and use program at any time, so maintenance should not be a big issue.
NFR-4	Performance	The performance of the website is to provide data to all users without unnecessary delay and provide 24*7 Availability,
NFR-5	Availability	The availability of the website is that the website will be active on the Internet and people will be able to browse to it.
NFR-6	Scalability	The scalability of the system is we have limited our projectto Indian cities and we have future plans to scale it to Continents level.

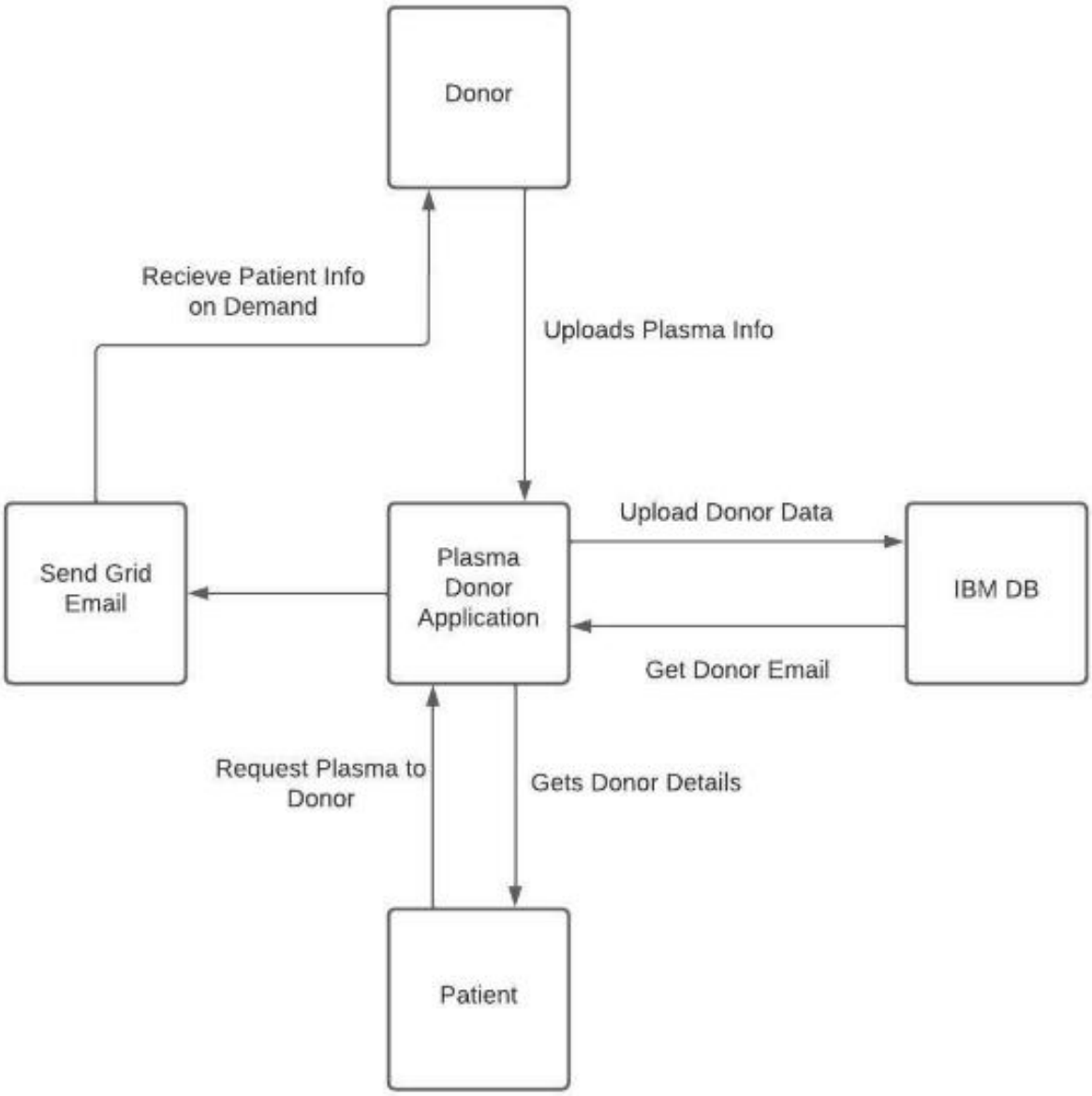
5. PROJECT DESIGN

5.1. Data Flow Diagrams

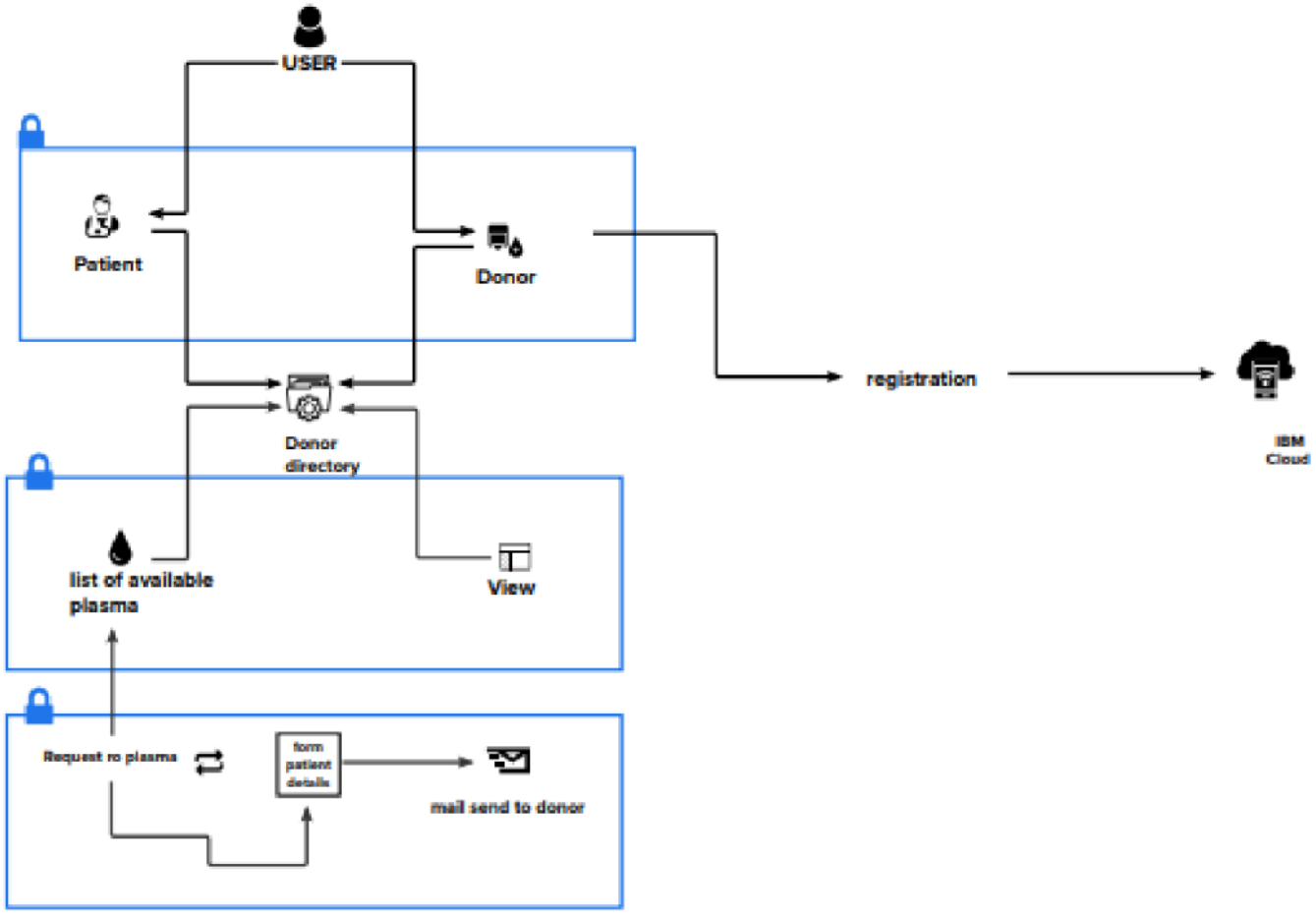
Level 0:



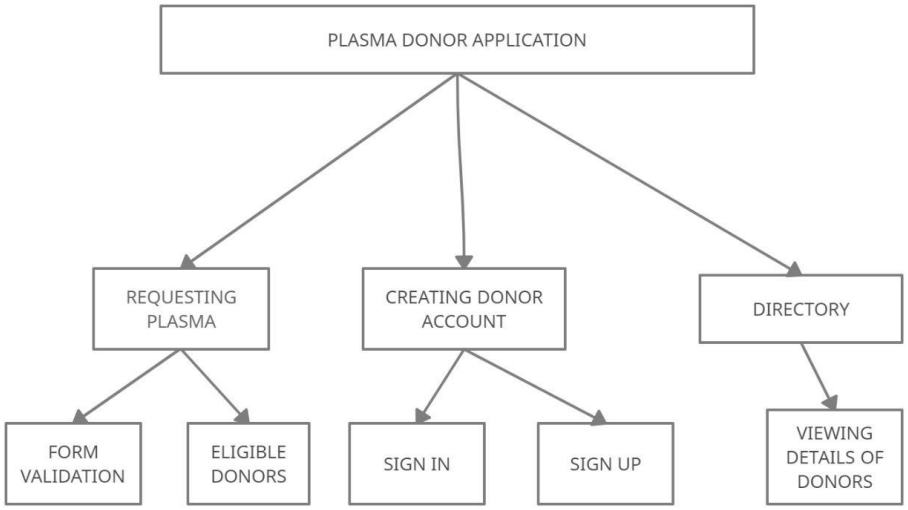
Level 1:


Donor
Recieve Patient Info on Demand
Uploads Plasma Info
Send Grid Email
Plasma Donor Application
Upload Donor Data
IBM DB
Get Donor Email
Request Plasma to Donor
Gets Donor Details
Patient

5.2. Solution & Technical Architecture



5.3. User Stories



6. PROJECT PLANNING & SCHEDULING

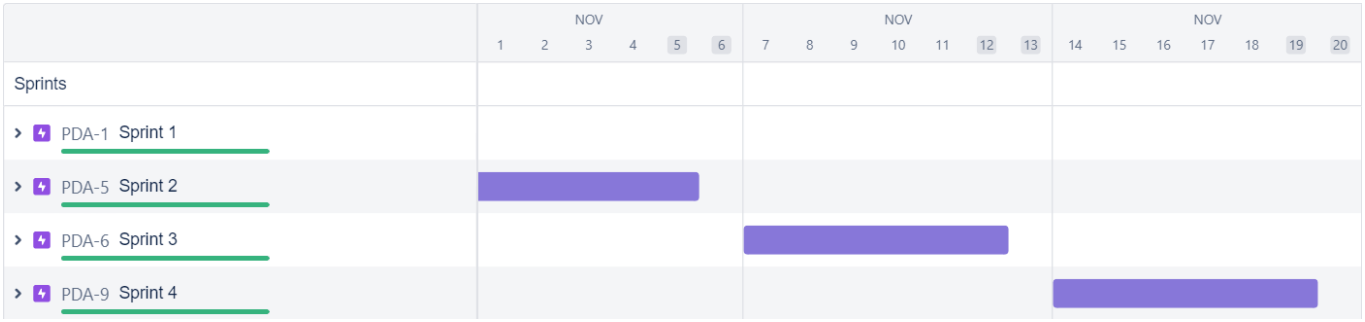
6.1. Sprint Planning & Estimation

Sprint	Functional Requirement	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint - 1	Develop Flask Webpage	USN -1	A Flask web application should be developed.	4	High	Madhusudhanan CB
	Registration	USN-2	As a user, I can register for the application by entering my email, password.	8	High	Prasanna S
	Login	USN-3	As a user, I can log into the application by entering email & password,	8	High	Raagavan JS
Sprint - 2	Register for Plasma Donation	USN-4	As a donor, I have to register to intimate the Users that I am interested in Donating the Plasma.	10	High	Ashok Kumar S S
	Request for Plasma	USN-5	As a recipient, I have to request the plasma from the donors.	10	High	Ashok Kumar S S, Prasanna S
Sprint - 3	Awareness	USN-6	As a public, I have to get aware about plasma donation.	10	Low	Raagavan J S, Madhusudhanan C B
	Database	USN-7	The user data has to be store and needs to maintained in the database.	10	High	Ashok Kumar S S, Madhusudhanan C B
Sprint – 4	Send Notification	USN-9	As a donor, User have to be notified for donation of the Plasma when it is in need.	12	High	Ashok Kumar S S, Raagavan J S
	Software Testing	USN-10	As a user, I want to access the applicationwithout any bugs and drawbacks.	8	High	Prasanna S, Raagavan J S, Madhusudhanan C B

6.2. Sprint Delivery Schedule

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release 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

6.3. Reports from JIRA



7. CODING & SOLUTIONING

7.1. Donation Eligibility

Any user easily views the Plasma donation requirements. It is not mandatory to sign in to view these details and New interested donor can view the requirements and check whether donation eligibility will be applicable to him/her.

```
div class="eligiblity">
  <h1>General Donation Eligibility
    <div class="line"></div>
  </h1>
  <div class="ele">
    <i class="material-icons" style="font-size: 150px">cake</i>
    <text>Donor must be at least 18-years-old</text>
  </div>
  <div class="ele">
```

```
    <text>
      Only Males & Nulliparous Female donors <br />
      of Weight > 55Kg can donate</text>
    <i class="material-icons" style="font-size: 150px">monitor_weight</i>
  </div>
  <div class="ele">
    <i class="material-icons" style="font-size: 150px">badge</i>
    <text> Government ID Card of Donor required</text>
  </div>
  <div class="ele">
    <text> Must have Postivie history of COVID-19 </text>
    <i class="material-icons" style="font-size: 150px">coronavirus</i>
  </div>
  <div class="ele">
    <i class="material-icons" style="font-size: 150px">block</i>
    <text> COVID-19 Symptom-Free for >= 28 Days</text>
  </div>
  <div class="ele">
    <text> Any Vaccination Status is OK </text>
    <i class="material-icons" style="font-size: 120px">vaccines</i>
  </div>
  <div class="ele">
    <i class="material-icons" style="font-size: 150px">no_drinks</i>
    <text> No Consumption of Alcohol</text>
  </div>
```

7.2. Plasma Matching

Whenever a Patient requests Plasma, the form is validated. The validated details are run through DB and find eligible donor who can contribute plasma based on patient blood group. Specific supporters are alone notified.

```
def plasmaMatchCheck(pbgrp, dbgrp):
  if (dbgrp == "AB+" or dbgrp == "AB-"):
    return True
  elif ((dbgrp == "A+" or dbgrp == "A-") and (pbgrp == "A+" or pbgrp == "A-" or pbgrp == "O+" or pbgrp
== "O-")):
```

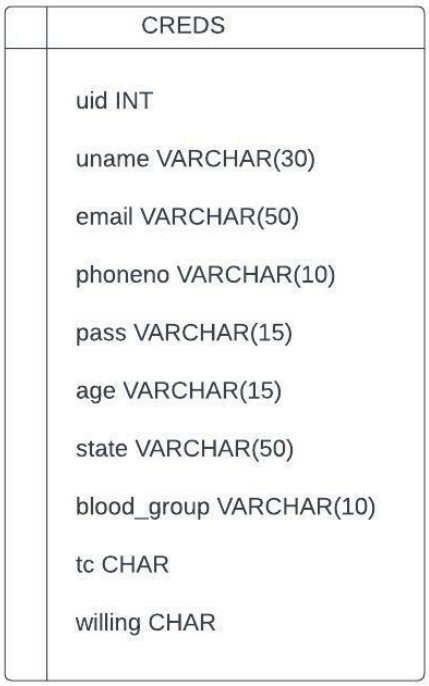
```
        return True
    elif ((dbgrp == "B+" or dbgrp == "B-") and (pbgrp == "B+" or pbgrp == "B-" or pbgrp == "O+" or pbgrp ==
"O-")):
        return True
    elif ((dbgrp == "O+" or dbgrp == "O-") and (pbgrp == "O+" or pbgrp == "O-")):
        return True
    else:
        return False
```

7.3. Option for Donation

Donors can use the option Willingness to Donate field. Once a Donor feels that he is incapable to Donate Plasma or doesn't meet the requirements over a particular period of time. The donor can deselect it, so that any request mail will not be forwarded to donor. Once he is eligible and becomes interested, he can select the option as Willing.

```
<input id="will" type="checkbox" name="will" value="Y" style="all: none" lass="default" />
        <label style="margin-right: 15px"> Willing To Donate Plasma</label>
```

7.4. Database Schema



8. TESTING

8.1. Test Cases

Test Case ID	Test Case Description	Test Steps	Test Data	Excepted Result	Actual Results	Pass/Fail
TU01	Submitting the form using invalid mobile number	1.Go to the website 2.Go to Request plasma and give the Invalid number 3.Click the Submit button	Mobile No.: 938530	Notify user about Invalid number	Invalid number has been accepted	Fail
TU02	Submitting the form using valid mobile number	1.Go to the website 2.Go to Request Plasma and give the invalid number 3.Click the Submit button	Mobile number: 9385304153	Process the form without any errors	No errors were notified	Pass
TU03	Submitting the Request donors page without filling any mandatory option	1.Go to the website 2.Go to Request Plasma and don't fill any options 3.Click the Notify donors button	-	Request user to fill Data	Empty data were accepted	Fail

TU04	Submitting the Request donors page by filling all the mandatory option	1.Go to the website 2.Go to Request Plasma and fill all the options 3.click the Notify donors button	Patient name Age Blood group Hospital Name Mobile number Hospital Address Hospital District	Send email to Donors	Email was sent to Donors	Pass
TU05	In the request plasma page, we are going to give the age in negative	1.Go to the website 2.click the Request plasma 3.Click the Age field	-10	Value should not be accepted	Value has processed	Fail

8.2. User Acceptance Testing

Section	Total Cases	Not Tested	Fail	Pass
Home Page	2	0	0	2
Request donors	6	0	0	6
Eligibility Criteria	0	0	0	0
Donors Directory	0	0	0	0

9. ADVANTAGES & DISADVANTAGES

9.1. Advantages

- No need of Patient Login. Patient's can directly fill the form and Notify donors
- Any user can become Donor by verifying the Donation Eligibility details
- Donors at the Patient's City will be notified. Avoid unwanted request mail to other district donors.
- Donors are provided by patient data and Mobile number making them Easy to contact.

9.2. Disadvantages

- Genuineness of Patient should be filtered.
- Donor should have the Donated data in dashboard
- Once a patient gets Eligible donor at Hospital, Other donors should get notified so that they help other patients.

10. CONCLUSION

Thus, the Plasma Donor Application will be helpful for the patients and our application is capable to quickly fetch the donors through our database. We display plasma donation eligibility parameters so that to create awareness about it. Our application notifies Donor immediately with all patient details to enter into the process of life saving.

11. FUTURE SCOPE

The Application has to expanded in future with the following process,

- Verify the Patient Identity. Trustful patient details alone should be process and block fake data.
- Once the Patient gets Donor. Other Contributors should get notified this status.
- Donation Dashboard should be present for Donors.
- Display of Current Donation Centre or Camps

12. APPENDIX

GitHub Link: https://github.com/IBM-EPBL/IBM-Project-27145-1660047541

Website Link: http://169.51.204.11:32321/

Demo Link:
https://drive.google.com/file/d/167bcxH3CMROnzLV_r_0u4gvz6u5zt1I1/view?usp=sharing

Code (app.py):

```
from urllib import request
from flask import Flask, render_template, redirect, request, session
from flask_session import Session
from connection import *
from flask_mail import Mail, Message

app = Flask(__name__)
app.config["SESSION_PERMANENT"] = False
app.config["SESSION_TYPE"] = "filesystem"
Session(app)

mail = Mail(app)  # instantiate the mail class
mail.init_app(app)

# configuration of mail
app.config['MAIL_SERVER'] = 'mail.smartinternz.com'
app.config['MAIL_PORT'] = 587
app.config['MAIL_USERNAME'] = '917719it127@smartinternz.com'
app.config['MAIL_PASSWORD'] = 'PNTIBMDh'
app.config['MAIL_USE_TLS'] = False
app.config['MAIL_USE_SSL'] = False
mail = Mail(app)


@app.route('/')
def index():
    if not session.get("name"):
        return render_template("index.html")
    return redirect("/homepage")


@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST' and 'ml' in request.form and 'p' in request.form:
        ml = request.form['ml']
        pas = request.form['p']

        stmt = "select * from creds where email=? and pass=?"
        prep = ibm_db.prepare(conn, stmt)
        ibm_db.bind_param(prep, 1, ml)
        ibm_db.bind_param(prep, 2, pas)
        res = ibm_db.execute(prep)
        data = ibm_db.fetch_assoc(prep)
        if data:
```

```
            session["name"] = data["UNAME"]
            session["email"] = data["EMAIL"]
            return redirect("/homepage")
        else:
            print("Invalid Login")
    return render_template('login.html')


@ app.route("/logout")
def logout():
    session["name"] = None
    return render_template("index.html")


@ app.route("/homepage")
def homepage():
    if not session.get("name"):
        return redirect("/index")
    return render_template('homepage.html')


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


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


@ app.route('/donordir', methods=['GET', 'POST'])
def donordir():
    l = []
    pdata = []
    if not session.get("name"):
        if request.method == 'POST':
            pname = request.form['name']
            age = request.form['age']
            pbgrp = request.form['bgrp']
            pmno = request.form['mno']
            hname = request.form['hname']
            haddr = request.form['addr']
            pdis = request.form['dis']
            stmt = "SELECT uname,email,blood_group FROM creds WHERE willing='Y'
AND  district=?"
```

```
            prep = ibm_db.prepare(conn, stmt)
            ibm_db.bind_param(prep, 1, pdis)
            res = ibm_db.execute(prep)
            data = ibm_db.fetch_assoc(prep)
            print("Plus")
            l.append(data)
            pdata.append(pname)
            pdata.append(age)
            pdata.append(pmno)
            pdata.append(hname)
            pdata.append(haddr)
            pdata.append(pdis)
            if (data):
                for item in l:
                    uname = item['UNAME']
                    email = item['EMAIL']
                    dbgrp = item['BLOOD_GROUP']
                    print(uname+" "+dbgrp)
                    if (plasmaMatchCheck(pbgrp, dbgrp)):
                        plasmaRequestMail(email, pdata)
                return render_template('donordir.html', req=len(l))
            else:
                # Display No Donor Avail
                print("D Unavail")
        return render_template('donordir.html', req=0)
    else:
        return render_template('logdonordir.html')


@ app.route('/logdonordir')
def logdonordir():
    stmt = "SELECT uname,age,district,blood_group FROM creds WHERE willing='Y'"
    res = ibm_db.exec_immediate(conn, stmt)
    data = ibm_db.fetch_assoc(res)
    l = []
    l.append(data)
    if not session.get("name"):
        return render_template('donordir.html', data=l)
    return render_template('logdonordir.html', data=l)


@ app.route("/profile", methods=['GET', 'POST'])
def profile():
    if not session.get("name"):
        return redirect("/index")
```

```
stmt = "select * from creds where email=?"
sql = session["email"]
prep = ibm_db.prepare(conn, stmt)
ibm_db.bind_param(prep, 1, sql)
res = ibm_db.execute(prep)
data = ibm_db.fetch_assoc(prep)
print(data)
if (data):
    phno = data["PHONENO"]
    age = data["AGE"]
    sex = data["SEX"]
    state = data["STATE"]
    dis = data["DISTRICT"]
    bgrp = data["BLOOD_GROUP"]
    will = data["WILLING"]

if request.method == 'POST':
    phno = request.form['phno']
    age = request.form.get('age')
    sex = request.form.get('sex')
    state = request.form.get('state')
    dis = request.form.get('dis')
    bgrp = request.form.get('bgrp')
    will = request.form.get('will')

    if data["PHONENO"] != phno:
        stmt = "UPDATE CREDS SET PHONENO= ? WHERE EMAIL=?;"
        prep = ibm_db.prepare(conn, stmt)
        ibm_db.bind_param(prep, 1, phno)
        ibm_db.bind_param(prep, 2, session['email'])
        res = ibm_db.execute(prep)

    if data["AGE"] != age:
        print("CHNAGE")
        stmt = "UPDATE CREDS SET AGE= ? WHERE EMAIL=?;"
        prep = ibm_db.prepare(conn, stmt)
        ibm_db.bind_param(prep, 1, age)
        ibm_db.bind_param(prep, 2, session['email'])
        res = ibm_db.execute(prep)

    if data["SEX"] != sex:
        stmt = "UPDATE CREDS SET SEX= ? WHERE EMAIL=?;"
        prep = ibm_db.prepare(conn, stmt)
        ibm_db.bind_param(prep, 1, sex)
```

```
            ibm_db.bind_param(prep, 2, session['email'])
            res = ibm_db.execute(prep)

        if data['STATE'] != state:
            stmt = "UPDATE CREDS SET STATE= ? WHERE EMAIL=?;"
            prep = ibm_db.prepare(conn, stmt)
            ibm_db.bind_param(prep, 1, state)
            ibm_db.bind_param(prep, 2, session['email'])
            res = ibm_db.execute(prep)

        if data['DISTRICT'] != dis:
            stmt = "UPDATE CREDS SET DISTRICT= ? WHERE EMAIL=?;"
            prep = ibm_db.prepare(conn, stmt)
            ibm_db.bind_param(prep, 1, dis)
            ibm_db.bind_param(prep, 2, session['email'])
            res = ibm_db.execute(prep)

        if data['BLOOD_GROUP'] != bgrp:
            stmt = "UPDATE CREDS SET BLOOD_GROUP= ? WHERE EMAIL=?;"
            prep = ibm_db.prepare(conn, stmt)
            ibm_db.bind_param(prep, 1, bgrp)
            ibm_db.bind_param(prep, 2, session['email'])
            res = ibm_db.execute(prep)

        if data['WILLING'] != will:
            stmt = "UPDATE CREDS SET WILLING= ? WHERE EMAIL=?;"
            prep = ibm_db.prepare(conn, stmt)
            ibm_db.bind_param(prep, 1, will)
            ibm_db.bind_param(prep, 2, session['email'])
            res = ibm_db.execute(prep)

    return render_template('profile.html', phno=phno, age=age, sex=sex,
state=state, dis=dis, bgrp=bgrp, will=will)


@ app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST' and 'u' in request.form and 'mail' in
request.form and 'phno' in request.form and 'p' in request.form:
        uname = request.form['u']
        ml = request.form['mail']
        pas = request.form['p']
        ph = request.form['phno']

        stmt = "select * from creds where email= ?"
```

```
        prep = ibm_db.prepare(conn, stmt)
        ibm_db.bind_param(prep, 1, ml)
        ibm_db.bind_param(prep, 2, pas)
        res = ibm_db.execute(prep)
        tuple = ibm_db.fetch_row(prep)

        if (tuple == False):
            stmt = "insert into creds values(default,?,?,?,?);"
            prep = ibm_db.prepare(conn, stmt)
            ibm_db.bind_param(prep, 1, uname)
            ibm_db.bind_param(prep, 2, ml)
            ibm_db.bind_param(prep, 3, ph)
            ibm_db.bind_param(prep, 4, pas)
            res = ibm_db.execute(prep)
            print("Account Created Successfully")
        else:
            print("Account Already Exist")
    return render_template('register.html')


def plasmaRequestMail(demail, pdata):
    msg = Message(
        'Important - Plasma Request',
        sender='requestplasma@plasmadonorapp.com',
        recipients=[demail]
    )
    try:
        cont = 'Hello Donor, We are in need of your help to save a Life. \n
Patient Name: ' + \
            pdata[0]+"\n Patient Age: "+pdata[1]+"\n Patient Mobile Number: " + \
            pdata[2]+"\n Hospital Name: "+pdata[3]+"\n Hospital Address:
"+pdata[4] + \
            "\n\n We kindly request Donors to verify the Paitent data via Mobile
Number and Hospital data before proceeding."
        msg.body = cont
        mail.send(msg)
    except:
        print('Unable to send Email')


if __name__ == "__main__":
    app.run(host='0.0.0.0', use_reloader=True)
```