IBM PROJECT DOCUMENTATION
PLASMA DONOR APPLICATION

PNT2022TMID02821

INDUSTRY MENTOR	**:**	**NAVYA**
FACULTY MENTOR	**:**	**SUJARITHA M**

TEAM ID	**:**	PNT2022TMID02821
TEAM LEAD	**:**	SUBASH R
TEAM MEMBER	**:**	SURYA V
TEAM MEMBER	**:**	TAMIL SELVAN A
TEAM MEMBER	**:**	SUDHAKAR T

INDEX

CHAPTER 1
INTRODUCTION

1.1 Project overview

Recent Covid-19 Pandemic has raised alarms over one of the most overlooked areas to focus: Healthcare Management. While healthcare management has various use cases for using data science, patient length of stay is one critical parameter to observe and predict if one wants to improve the efficiency of the healthcare management in a hospital.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.

1.2 Purpose

This system's goal is to use an web application to link donors and patients. Patient of this application may post requests for plasma donations or requests.The fundamental solution is to establish a centralised system is that a admin will keep track of current and previous Plasma Donation Events and also keep track of the location of the donor's plasma using google map.

CHAPTER 2
LITERATURE SURVEY

2.1 **Existing Problem**

- The already existing model is trained with minimal parameters by leaving the necessary parameter
- Low accuracy in prediction
- No feature extraction done
- High complexity.

2.1References

1. Yang J.-J., Li J., Mulder J., Wang Y., Chen S., Wu H., Wang Q., Pan H. Emerginginformation technologies for enhanced healthcare. *Comput. Ind.* 2015;69:3–11.doi:10.1016/j.compind.2015.01.012.[CrossRef] [GoogleScholar]
2. Cortada J.W., Gordon D., Lenihan B. *The Value of Analytics in Healthcare.* IBM Institute for Business Value; Armonk, NY, USA: 2012. Report No.: GBE03476-USEN-00. [Google Scholar]
3. Berwick D.M., Hackbarth A.D. Eliminating waste in US health care. *J. Am. Med. Assoc.* 2012;307:1513–1516. doi: 10.1001/jama.2012.362. [PubMed] [CrossRef]
4. Makary M.A., Daniel M. Medical error-the third leading cause of death in the US. *Br. Med. J.* 2016;353:i2139. doi: 10.1136/bmj.i2139. [PubMed] [CrossRef] [Google Scholar]
5. Prokosch H.-U., Ganslandt T. Perspectives for medical informatics. *Methods Inf. Med.* 2009;48:38–44. doi: 10.3414/ME9132. [PubMed] [CrossRef] [Google Scholar]
6. Simpao A.F., Ahumada L.M., Gálvez J.A., Rehman M.A. A review of analytics andclinical informatics in health care. *J. Med. Syst.* 2014;38:45. doi:10.1007/s10916-014-0045-x. [PubMed] [CrossRef] [GoogleScholar]

7. Ghassemi M., Celi L.A., Stone D.J. State of the art review: The data revolution incritical care. *Crit. Care.* 2015;19:118. doi: 10.1186/s13054-015- 0801-4. [PMC free article] [PubMed] [CrossRef] [Google Scholar]

8. Tomar D., Agarwal S. A survey on Data Mining approaches for Healthcare.*Int. J.Bio-Sci. Bio-Technol.*2013;5:241–266.doi:10.14257/ijbsbt.2013.5.5.25. [CrossRef] [Google Scholar]

9. PanagiotaGaletsia ,KorinaKatsaliakia , Sameer Kumarb,⁎ a School of Economics, Business Administration & Legal Studies, International Hellenic University, 14th km Thessaloniki-N. Moudania, Thessaloniki, 57001, Greece b Opus College of Business, University of St. Thomas Minneapolis Campus, 1000 LaSalle Avenue, Schulze Hall 435, Minneapolis, MN 55403, USA

10. K. Jee and G. H. Kim, "Potentiality of big data in the medical sector: Focus on how to reshape the healthcare system," *Healthc. Inform. Res.*, vol. 19, no. 2, pp. 79–85, Jun. 2013. doi: 10.4258/hir.2013.19.2.79

11. J. King, V. Patel, and M. F. Furukawa, "Physician adoption of electronic health record technology to meet meaningful use objectives: 2009–2012," The Office of the National Coordinator for Health Information Technology, Tech. Rep., Dec. 2012.

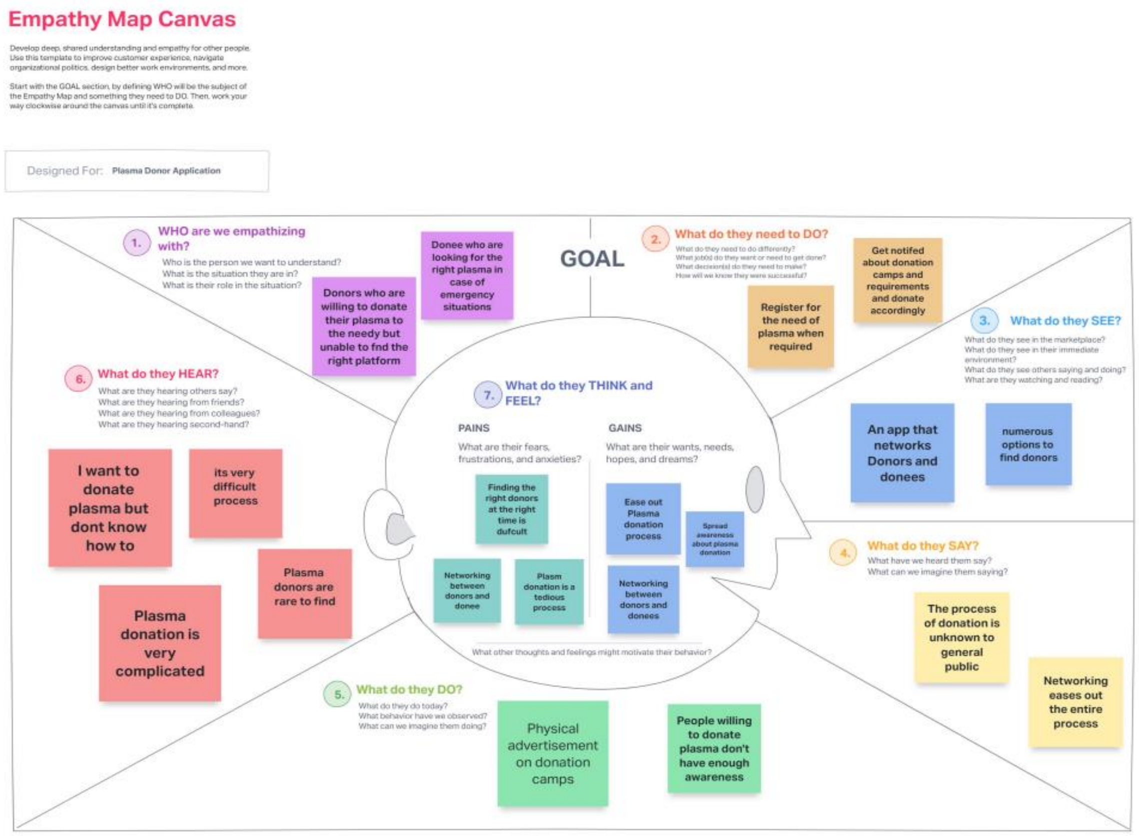
12. V. Mayer-Schönberger and K. Cukier, *Big Data: A Revolution That Will Transform How We Live, Work, and Think*. Eamon Dolan, 2014.

2.3Problem Statement Definition

Many major medical conditions are treated by plasma.Oneof the most well- known techniques known as plasma treatment,plasma is used to cure various incurable diseases.As there were no vaccines available to treat the infected patients during the Covid-19 emergency,theneed for plasma increased dramatically.Plasma therapy had a high probability of recovery but a very low donor count, therefore it was crucial to learn moreabout the donors in these circumstances. It would be helpful to save the contributor information and let clients know about the recurring donors because it can help them find the crucial information more quickly.

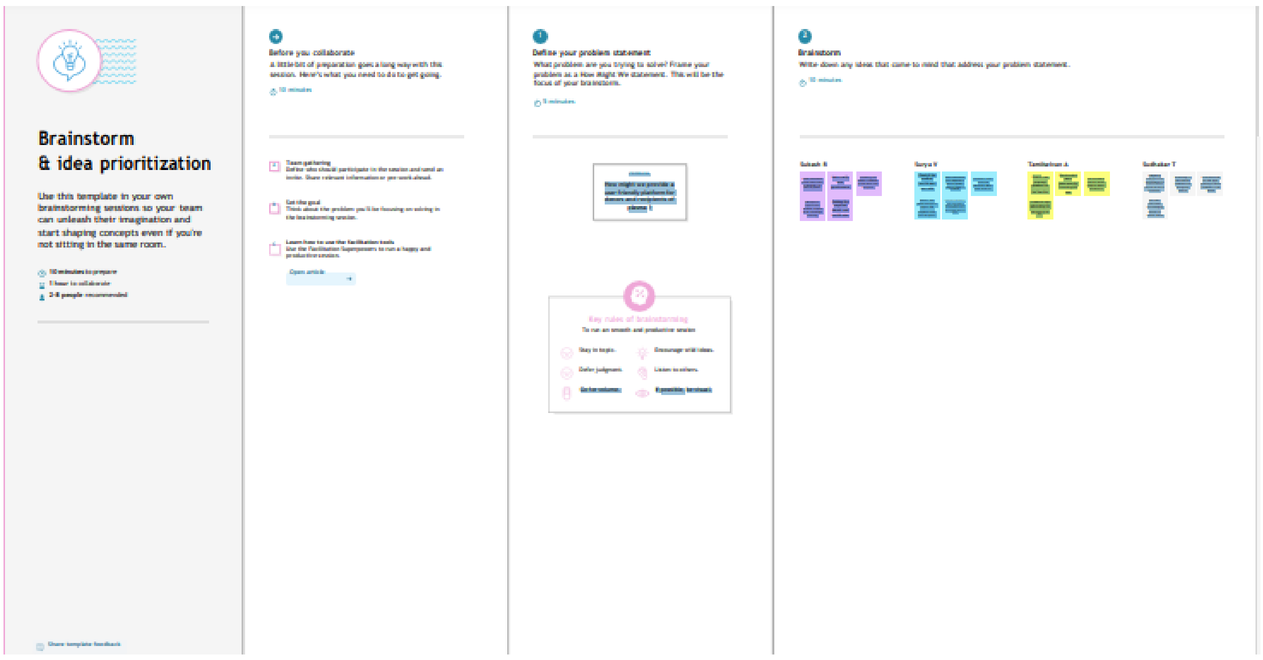
CHAPTER-3
IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas



Empathy Map Canvas
Develop deep, shared understanding and empathy for other people. Use this template to improve customer experience, navigate organizational politics, design better work environments, and more.
Start with the GOAL section, by defining WHO will be the subject of the Empathy Map and something they need to DO. Then, work your way clockwise around the canvas until it's complete.
Designed For: Plasma Donor Application
GOAL
1. WHO are we empathizing with?
Who is the person we want to understand?
What is the situation they are in?
What is their role in the situation?
Donors who are willing to donate their plasma to the needy but unable to fnd the right platform
Donee who are looking for the right plasma in case of emergency situations
2. What do they need to DO?
What do they need to do differently?
What job(s) do they want or need to get done?
What decision(s) do they need to make?
How will we know they were successful?
Register for the need of plasma when required
Get notifed about donation camps and requirements and donate accordingly
3. What do they SEE?
What do they see in the marketplace?
What do they see in their immediate environment?
What do they see others saying and doing?
What are they watching and reading?
An app that networks Donors and donees
numerous options to find donors
4. What do they SAY?
What have we heard them say?
What can we imagine them saying?
The process of donation is unknown to general public
Networking eases out the entire process
5. What do they DO?
What do they do today?
What behavior have we observed?
What can we imagine them doing?
Physical advertisement on donation camps
People willing to donate plasma don't have enough awareness
6. What do they HEAR?
What are they hearing others say?
What are they hearing from friends?
What are they hearing from colleagues?
What are they hearing second-hand?
I want to donate plasma but dont know how to
its very difficult process
Plasma donors are rare to find
Plasma donation is very complicated
7. What do they THINK and FEEL?
PAINS
What are their fears, frustrations, and anxieties?
Finding the right donors at the right time is dufcult
Networking between donors and donee
Plasm donation is a tedious process
GAINS
What are their wants, needs, hopes, and dreams?
Ease out Plasma donation process
Spread awareness about plasma donation
Networking between donors and donees
What other thoughts and feelings might motivate their behavior?

3.2 Ideation & Brainstorming



Brainstorm & idea prioritization
Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.
Before you collaborate
Define your problem statement
Brainstorm
Key rules of brainstorming

3 Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, 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

4 Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes

After you collaborate

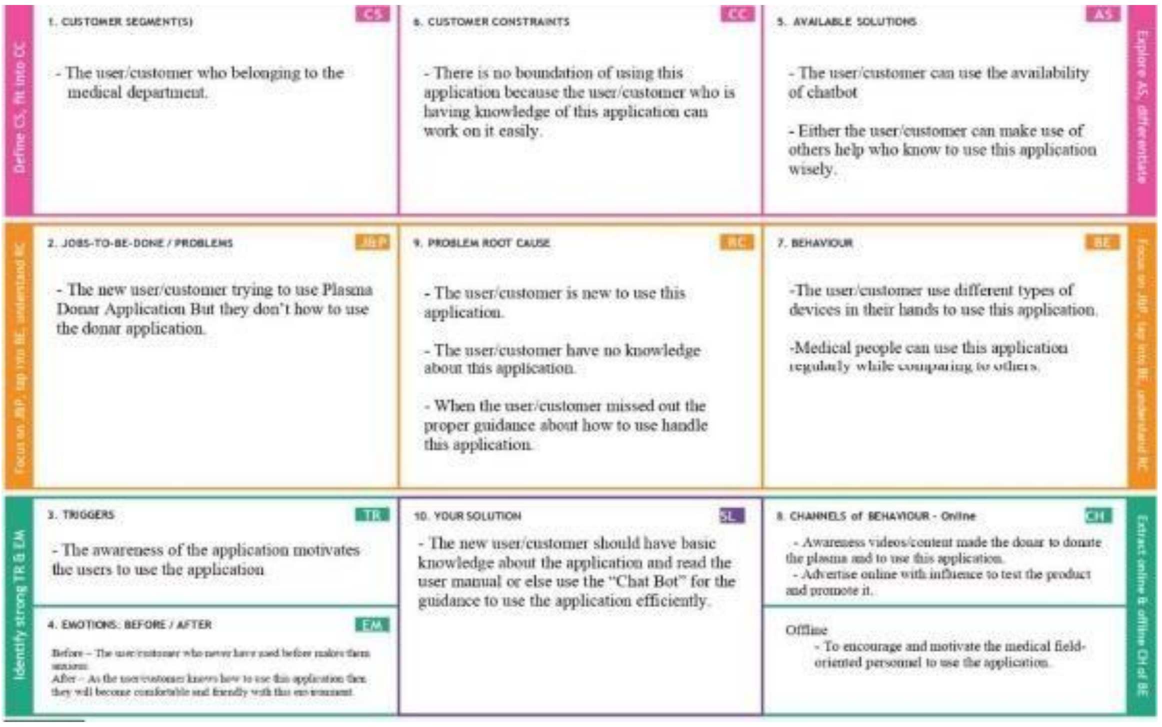
You can export the mural as an image or pdf to share with members of your company who might find it helpful.

3.3 Proposed Solution

Predict the length of stay of patients.

The length of the stay can be predicted using either Random forest or Decision Tree for more accuracy. Certain parameters like age, stage of the diseases, disease diagnosis, severity of illness, type of admission, facilities allocated, etc., are used for prediction. IBM Cognos will be used for data analytic s. The model will betrained using colab.It predicts the length of stay (LOS) of the patients with more accuracy. As a result proper resources and therapy can be provided.Patients can get proper treatment and better medical care than before which helps them for their faster recovery. So the prediction minimizes the overflow of patients and helps in resource management and optimize their resource utilization. Hence this leads to faster recovery and lower the expenses for treatment. It improves the trust in hospital management. It avoids the major risk of spreading infection among the hospital staff. This leads to overall safety of hospital staff and patients.Resource consumption is optimized.This model can be used by all government hospitals, private hospitals, and even in The model is trained with the real world hospital survey for better prediction small clinics.Length of the stay will be predicted with more accuracy.This model predicts the length of the stay for all kinds of patients and predicts with more accuracy.

3.4 Problem Solution fit



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- The user/customer who belonging to the medical department.

6. CUSTOMER CONSTRAINTS CC

- There is no boundation of using this application because the user/customer who is having knowledge of this application can work on it easily.

5. AVAILABLE SOLUTIONS AS

- The user/customer can use the availability of chatbot
- Either the user/customer can make use of others help who know to use this application wisely.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

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

- The new user/customer trying to use Plasma Donar Application But they don't how to use the donar application.

9. PROBLEM ROOT CAUSE RC

- The user/customer is new to use this application.
- The user/customer have no knowledge about this application.
- When the user/customer missed out the proper guidance about how to use handle this application.

7. BEHAVIOUR BE

-The user/customer use different types of devices in their hands to use this application.

-Medical people can use this application regularly while comparing to others.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

- The awareness of the application motivates the users to use the application

4. EMOTIONS: BEFORE / AFTER EM

Before – The user/customer who never have used before makes them [illegible]

After – As the user/customer knows how to use this application then they will become comfortable and friendly with this environment.

10. YOUR SOLUTION SL

- The new user/customer should have basic knowledge about the application and read the user manual or else use the "Chat Bot" for the guidance to use the application efficiently.

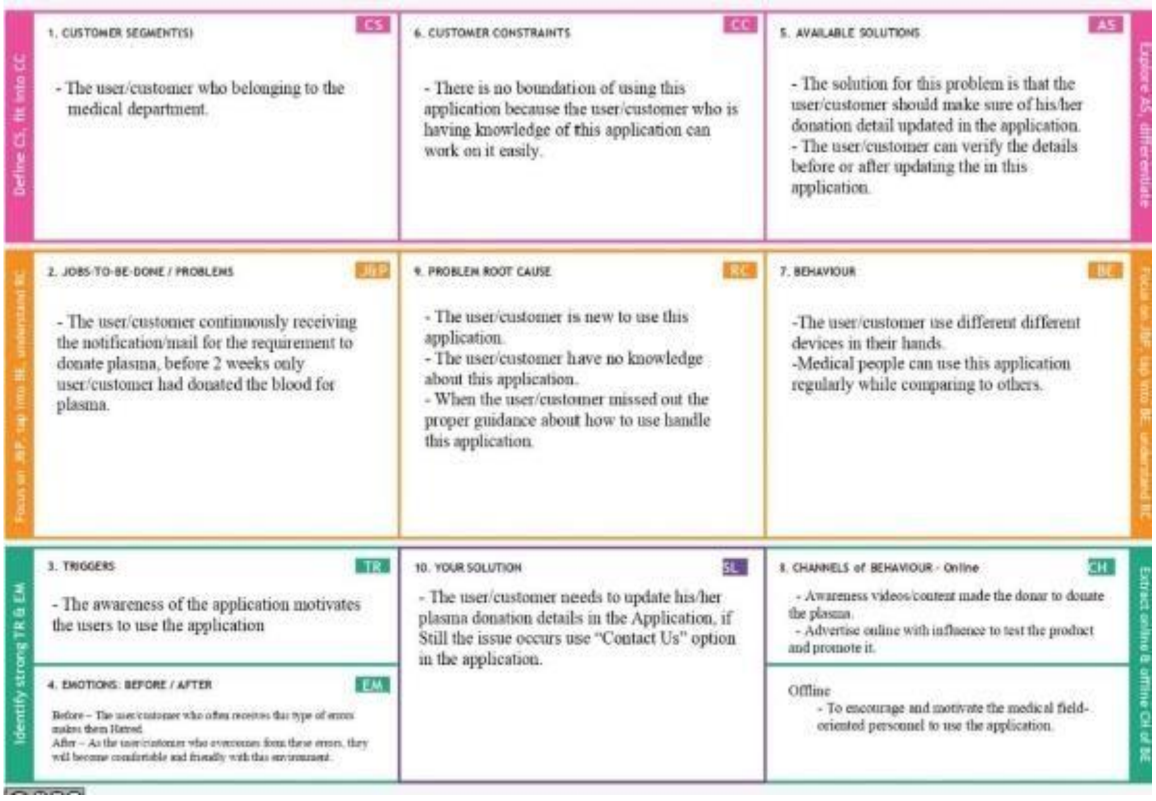
8. CHANNELS of BEHAVIOUR - Online CH

- Awareness videos/content made the donar to donate the plasma and to use this application.
- Advertise online with influence to test the product and promote it.

Offline

- To encourage and motivate the medical field-oriented personnel to use the application.

Extract online & offline CH of BE



Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

- The user/customer who belonging to the medical department.

6. CUSTOMER CONSTRAINTS CC

- There is no boundation of using this application because the user/customer who is having knowledge of this application can work on it easily.

5. AVAILABLE SOLUTIONS AS

- The solution for this problem is that the user/customer should make sure of his/her donation detail updated in the application.
- The user/customer can verify the details before or after updating the in this application.

Explore AS, differentiate

Focus on J&P, tap into BE, understand RC

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

- The user/customer continuously receiving the notification/mail for the requirement to donate plasma, before 2 weeks only user/customer had donated the blood for plasma.

9. PROBLEM ROOT CAUSE RC

- The user/customer is new to use this application.
- The user/customer have no knowledge about this application.
- When the user/customer missed out the proper guidance about how to use handle this application.

7. BEHAVIOUR BE

-The user/customer use different different devices in their hands.

-Medical people can use this application regularly while comparing to others.

Focus on J&P, tap into BE, understand RC

Identify strong TR & EM

3. TRIGGERS TR

- The awareness of the application motivates the users to use the application

4. EMOTIONS: BEFORE / AFTER EM

Before – The user/customer who often receives this type of errors makes them Hatred.

After – As the user/customer who overcomes from these errors, they will become comfortable and friendly with this environment.

10. YOUR SOLUTION SL

- The user/customer needs to update his/her plasma donation details in the Application, if Still the issue occurs use "Contact Us" option in the application.

8. CHANNELS of BEHAVIOUR - Online CH

- Awareness videos/content made the donar to donate the plasma.
- Advertise online with influence to test the product and promote it.

Offline

- To encourage and motivate the medical field-oriented personnel to use the application.

Extract online & offline CH of BE

CHAPTER – 4
REQUIREMENT ANALYSIS

4.1 Functional Requirement

Following are the functional requirements of the proposed solution.

FRNo.	**FunctionalRequirement(Epic)**	**SubRequirement(Story/Sub-Task)**
FR-1	User Registration	Registration through Form(WebApp)
FR-2	User Confirmation	Confirmation via Email
FR-3	Certification	After the donor donates plasma, we will give them a certificate of appreciation and authentication.
FR-4	Statistical data	The availability of plasma is given in the page as stats,which will be helpful for the users.
FR-5	User Plasma Request	Users can request to donate plasma by filling out the request form on the page.
FR-6	Searching/reporting requirements	Users can use the search bar to lookup information about camps and othertopics.

4.2 Non-Functional requirements

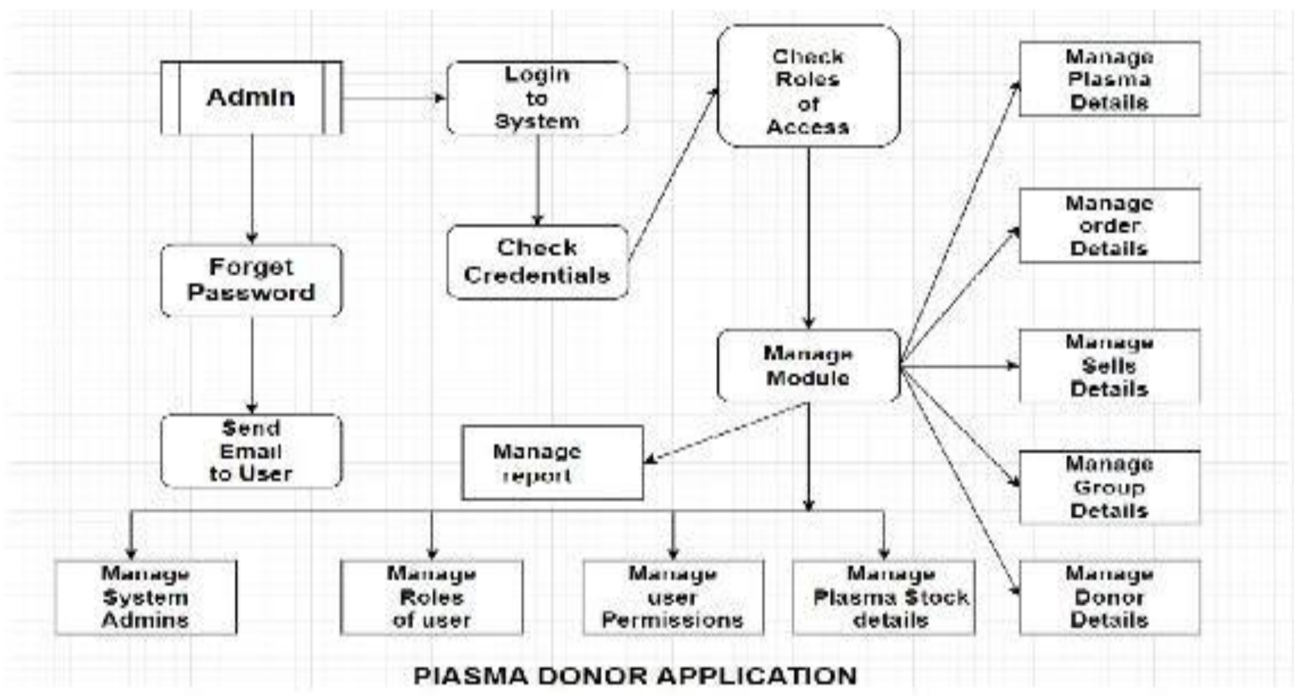
NFR-4	**Performance**	Users should have a proper Internet Connection.
NFR-5	**Availability**	The system including the online and offline components should be available 24/7.
NFR-6	**Scalability**	The application has the ability to handle growing number of users and load without compromising on Performance and causing disruptions to user experience.

CHAPTER-5
PROJECT DESIGN

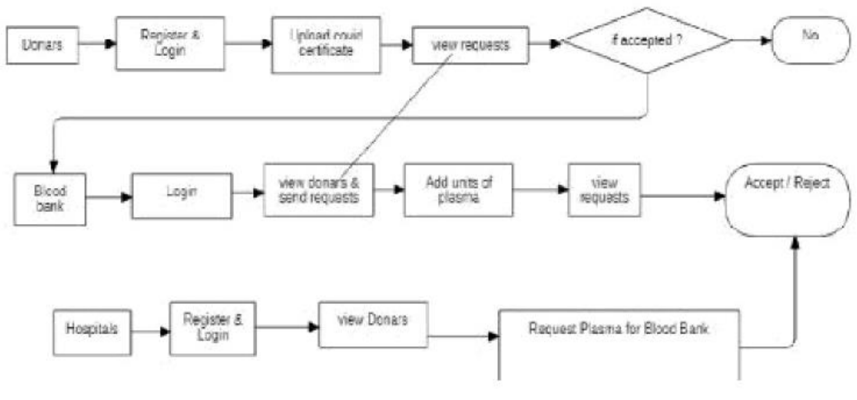
5.1 Data Flow Diagrams

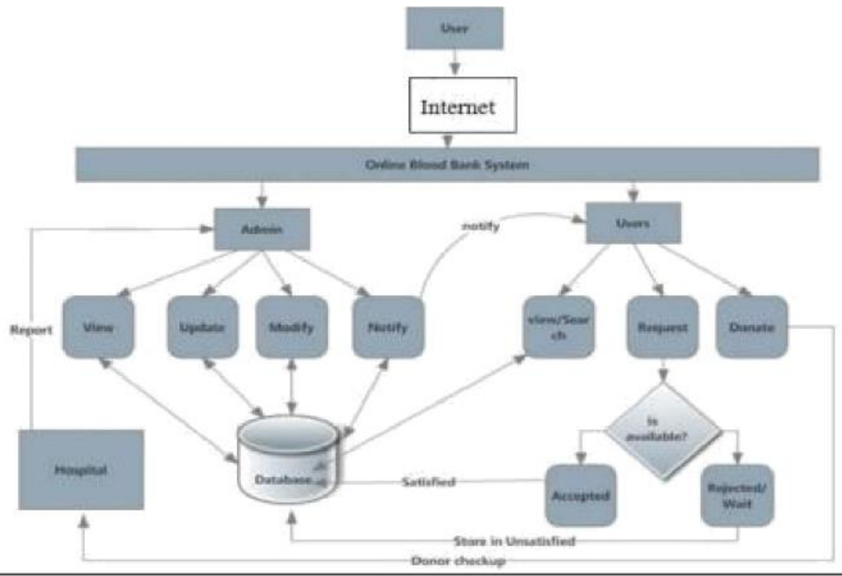
A data flow diagram is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically.it shows how data enters and leaves the system, what changes the information and, where data is stored

Diagram:



5.2 Solution & 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	A User can register and create the user account.	6	High	Subash R Surya V
Sprint-1	Login	USN-2	A User can sign-in to the application by entering the registered email id and password.	6	High	Subash R Surya V
Sprint-1	Admin Register	USN-3	An admin can register through the admin registry.	4	Medium	Sudhakar T Tamil Selvan A
Sprint-1	Register Admin Via Script	USN-4	Creating an Admin Account using a python script. As for security reasons we should implement a separate python script.	4	High	Sudhakar T Subash R Surya V
Sprint-2	Implementing Authentication System	USN-5	creating an authentication system for both admin and users using flask application	6	High	Sudhakar T Tamil Selvan A
Sprint-2	Creating Tables	USN-6	Creating Db2 account and creating the tables in DB2 in IBM cloud db2	4	Medium	Subash R Tamil Selvan A
Sprint-2	Creating SSL certificate and integrating with python code	USN-7	Creating the SSL certificate to connect db2 via python code.	6	High	Surya V Sudhakar T
Sprint-2	Creating dashboard	USN-8	Admin and Donor can interact with our application.	4	Medium	Subash R
Sprint-3	Plasma request and donor acknowledge feature	USN-9	Admin can create plasma requests which will be shown in the user portal.	6	High	Subash R Surya V
Sprint-3	Creating dashboard for admin	USN-10	Admin dashboard, admin can view the total request has been requested for plasma by the recipient/user.	6	High	Sudhakar T Tamil Selvan A
Sprint-3	Integrating the Watson chat bot	USN-11	Users can use the chatbot for basic clarification using the chatbot.	4	Medium	Subash R Tamil Selvan A
Sprint-3	Integration with SendGrid.	USN-12	The source/verification mail for both user(donar and recipient) .	4	Medium	Surya V Sudhakar T
Sprint-4	Docker installation	USN-13	Installing Docker CLI	4	Low	Surya V

CHAPTER-6

PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 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)

$$AV = \frac{sprint\ duration}{velocity} = \frac{20}{10} = 2$$

Velocity: Sprint - 1

Sprint duration = 6 days
Velocity of the team = 20 points

$$\text{average velocity (AV)} = \frac{\text{Velocity}}{\text{Sprint duration}}$$

AV = 20/6 = 3.34

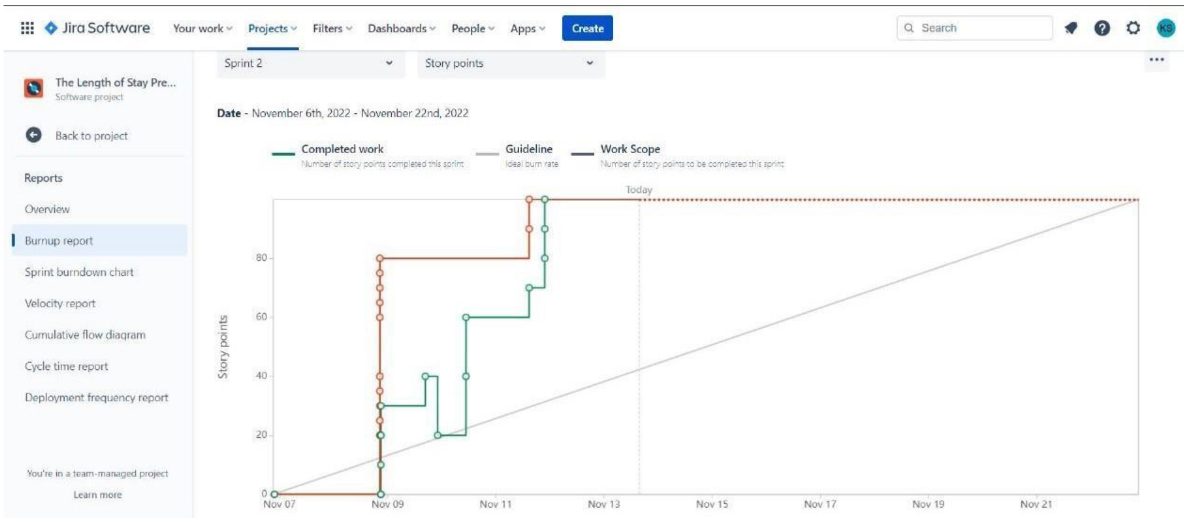
Average Velocity = 3.34

6.2 Sprint Delivery Schedule

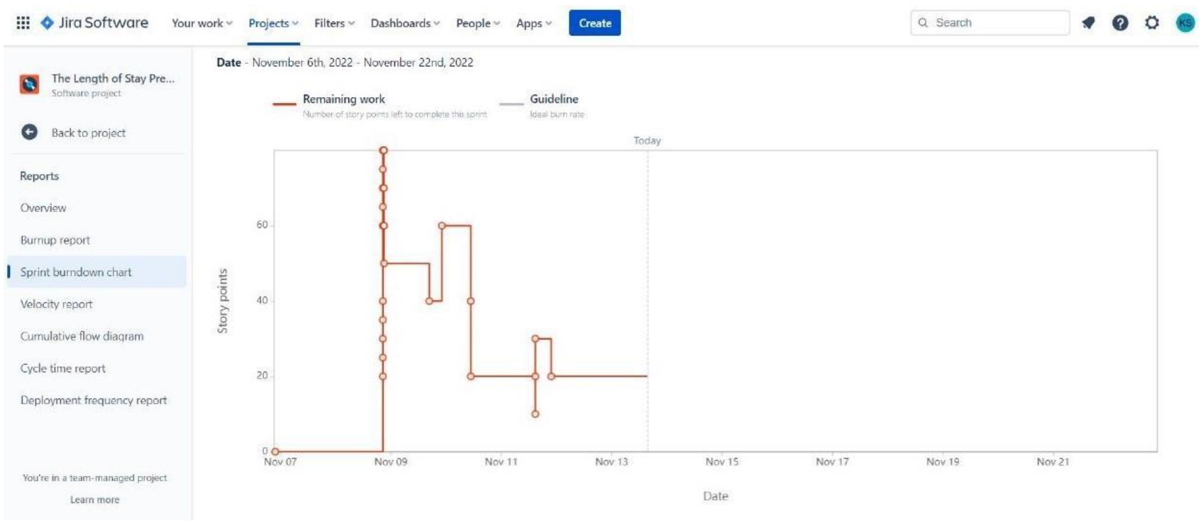
Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Initial creation process	USN-1	Create template, Static and python flask app.	20	High	Subash R Surya V Sudhakar T Tamil Selvan A
Sprint-2	Cloud and database	USN-2	Connecting the python flask app with database, object storage created in Cloud and implementation of chatbot	20	High	Subash R Surya V Sudhakar T Tamil Selvan A
Sprint-3	Deployment in DevOps, Mailing	USN-3	Develop the project, create it as image with docker, containerize in container registry and deploy in Kubernetes. Add the mailing service	20	High	Subash R Surya V Sudhakar T Tamil Selvan A
Sprint-4	Testing, Deployment and user experience	USN-4	To do all the testing and to make sure the use of the software handy to user.	20	High	Subash R Surya V Sudhakar T Tamil Selvan A

6.3 Reports from JIRA

Burnt Up Chart


Sprint 2
Story points
Date - November 6th, 2022 - November 22nd, 2022

Burnt Down Chart



CHAPTER-7

CODING & SOLUTIONING

7.1 Code

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

load_dotenv()

app = Flask(_name_)

try:

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=
54a2f15b-5c0f-46df-8954-
7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.c
loud;PORT=32733;SECURITY=SSL;SSLServerCertificate=
DigiCertGlobalRootCA.crt;PROTOCOL=TCPIP;UID=ssz03
200;PWD=ROEcnQjP3TOa1Yxw;", "", "")
  print("success")
except:
  print(ibm_db.conn_errormsg())

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


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 uname =?
AND psw=?"
        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 account:
            session['loggedin'] = True
            session['id'] = account['UNAME']
            userid = account['UNAME']
            session['username'] = account['UNAME']
            return redirect(url_for("about"))
        else:
            msg = 'Incorrect Username and Password'
            flash(msg)
            return redirect(url_for("login"))

# 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 uname =?"
        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
deonor(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 deonor 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('/dash')
```

```
def dash():
    if session['loggedin'] == True:
        sql = "SELECT COUNT(), (SELECT COUNT()
FROM DEONOR WHERE BloodType= 'O Positive'),
(SELECT COUNT() FROM DEONOR WHERE
BloodType='A Positive'), (SELECT COUNT() FROM
DEONOR WHERE BloodType='B Positive'), (SELECT
COUNT() FROM DEONOR WHERE BloodType='AB
Positive'), (SELECT COUNT() FROM DEONOR WHERE
BloodType='O Negative'), (SELECT COUNT() FROM
DEONOR WHERE BloodType='A Negative'), (SELECT
COUNT() FROM DEONOR WHERE BloodType='B
Negative'), (SELECT COUNT(*) FROM DEONOR
WHERE BloodType='AB Negative') FROM dEonor"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        return render_template('dashboard.html',b=account)
    else:
        msg = 'Please login!'
        return render_template('login.html', msg = msg)

@app.route('/requested',methods=['POST'])
def requested():
    bloodgrp = request.form['bloodgrp']
    address = request.form['address']
    name = request.form['name']
    email = request.form['email']
    phone = request.form['phone']
    insert_sql = "INSERT INTO requested VALUES (?, ?, ?,
?, ?, ?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prep_stmt, 1, session["username"])
    ibm_db.bind_param(prep_stmt, 2, bloodgrp)
    ibm_db.bind_param(prep_stmt, 3, address)
    ibm_db.bind_param(prep_stmt, 4, name)
    ibm_db.bind_param(prep_stmt, 5, email)
    ibm_db.bind_param(prep_stmt, 6, phone)
    ibm_db.execute(prep_stmt)
    sendmail(email,'Plasma donor App plasma request', name
,'Your request for plasma is received.')

    # send_sql = "SELECT EMAIL FROM donor where
BloodTypeTYPE = ?"


 # prep_stmt = ibm_db.prepare(conn, send_sql)
    # ibm_db.bind_param(prep_stmt, 1, bloodgrp)
    # ibm_db.execute(prep_stmt)
    # send = ibm_db.fetch_assoc(prep_stmt)
    # print(send)
    # for i in send:
    #     sendmail(i.strip(), 'Plasma donor App plasma request',
name, 'A Donee has requested for Blood.')

    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 deonor 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)
    deonors = {}
    for values in account:
        if type(account[values]) == str:
            deonors[values] = account[values].strip()
        else:
            deonors[values] = account[values]

    print(deonors)
    return render_template("about.html", account = deonors)

@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 deonor 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 deonor 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"))
            # return render_template('about.html', pred="Your 
details   updated")

        else:
            insert_sql = "INSERT INTO deonor 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, (str(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')
```

HOME.HTML :

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8" />
    <title>Home</title>

    <link
      rel="stylesheet"

href="https://use.fontawesome.com/releases/v5.8.1/css/all.css"
    />
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge"
/>
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />

    <link
      rel="stylesheet"
      href="{{ url_for('static',filename='css/style.css') }}"
    />
  </head>
  <body>
    {% extends "template.html" %} {% block content %}
    <form action="{{ url_for('requested')}}" method="post">
      <input
        type="text"
        name="name"
        placeholder="Enter Name"
        required="required"
        style="color: black"
      />
      <input
        type="email"
        name="email"
        placeholder="Enter Email"
        required="required"
        style="color: black"
      />
      <input

    type="text"
        name="phone"
        placeholder="Enter 10-digit mobile number"
        required="required"
        style="color: black"
      />
      <select name="bloodgrp">
        <option value="select" selected>Choose your blood
group</option>
        <option value="O+">O Positive</option>
        <option value="A+">A Positive</option>
        <option value="B+">B Positive</option>
        <option value="AB+">AB Positive</option>
        <option value="O-">O Negative</option>
        <option value="A-">A Negative</option>
        <option value="B-">B Negative</option>
        <option value="AB-">AB Negative</option>
      </select>
      <textarea
        rows="4"
        placeholder="Enter the address"
        required="required"
        style="color: black"
        name="address"
      ></textarea>
      <button type="submit" class="btn btn-primary btn-block
btn-large">
        Submit the request
      </button>
    </form>

    <div>{{ pred }}</div>

    {% endblock %}
  </body>
</html>
```

CHAPTER 8
TESTING

8.1 TEST CASES

test case is a set of actions performed on a system to determine if it satisfies software requirements and functions correctly. The purpose of a test case is to determine if different features within a system are performing as expected and to confirm that the system satisfies all related standards, guidelines and customer requirements. The process of writing a test case can also help reveal errors or defects within the system.

Test cases are typically written by members of the quality assurance (QA) team or the testing team and can be used as step-by-step instructions for each system test. Testing begins once the development team has finished a system feature or set of features. A sequence or collection of test cases is called a test suite.

A test case document includes test steps, test data, preconditions and the postconditions that verify requirements.

Types of test cases

- Functionality Test Case
- User Interface Test Case
- Performance Test Case
- Integration Test Case
- Usability Test Case
- Database Test Case
- Security Test Case
- User Acceptance

8.2 USER ACCEPTANCE TESTING

Acceptance testing can take many forms, such as user acceptance testing, operational acceptance testing, contract acceptance testing and others. In this article, the focus is on user acceptance testing.

When people ask, “What is UAT testing?” One commonly cited definition of user acceptance testing is:

“Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.” (ISTQB Glossary V3.1)

There are other definitions as well, which highlight the need of understanding the context of user acceptance testing.

For example, in agile methods “acceptance testing” is defined as:
“An acceptance test is a formal description of the behavior of a software product, generally expressed as an example or a usage scenario…..Teams mature in their practice of agile use acceptance tests as the main form of functional specification and the only formal expression of business requirements. Other teams use acceptance tests as a complement to specification documents containing uses cases or more narrative text.” (Agile Alliance)

In addition, the Agile Alliance also adds:

“The terms “functional test”, “acceptance test” and “customer test” are used more or less interchangeably. A more specific term “story test”, referring to user stories is also used, as in the phrase “story test driven development”.
From these differences, we can see that traditional acceptance testing is seen in a validation context while agile methods tend to view acceptance testing as verification. In the agile context, acceptance criteria are basically described along with a user story to show specific cases that should be tested to show the user story has been implemented correctly and are basically the same as functional tests.

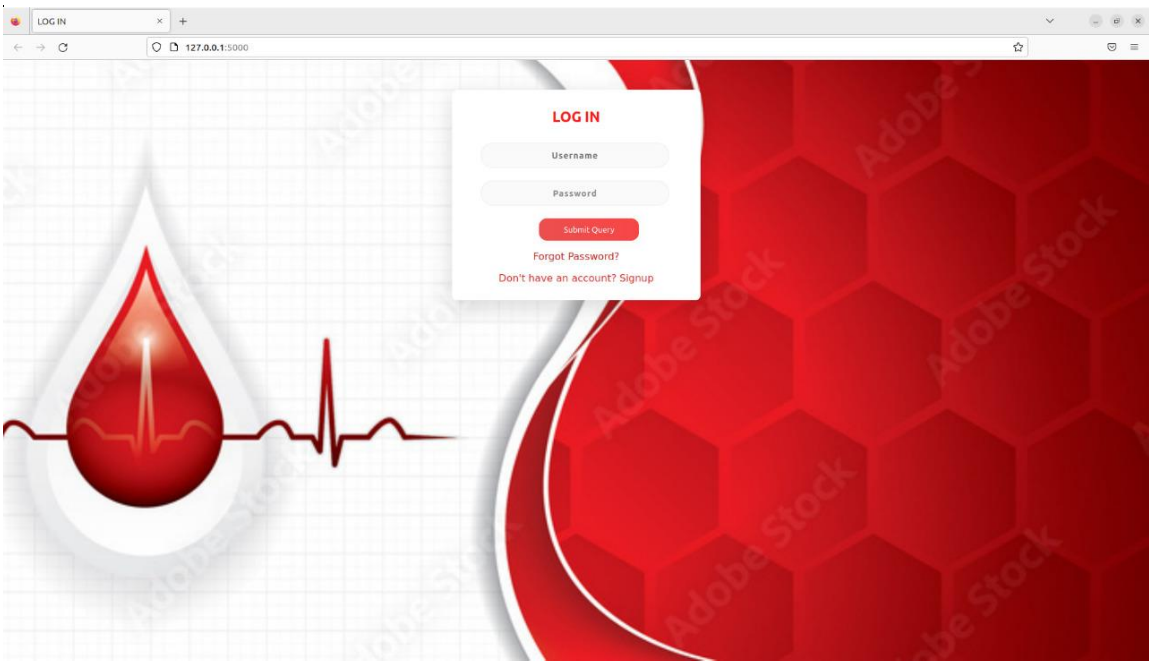
I often say that user acceptance testing is one of the most valuable levels of testing, but often performed at the worst possible time. That is because if process gaps or other major flaws are discovered in UAT testing, there is little time to fix them before release. Also, the options available to fix late-stage defects may be very limited at the end of a project.

CHAPTER 9

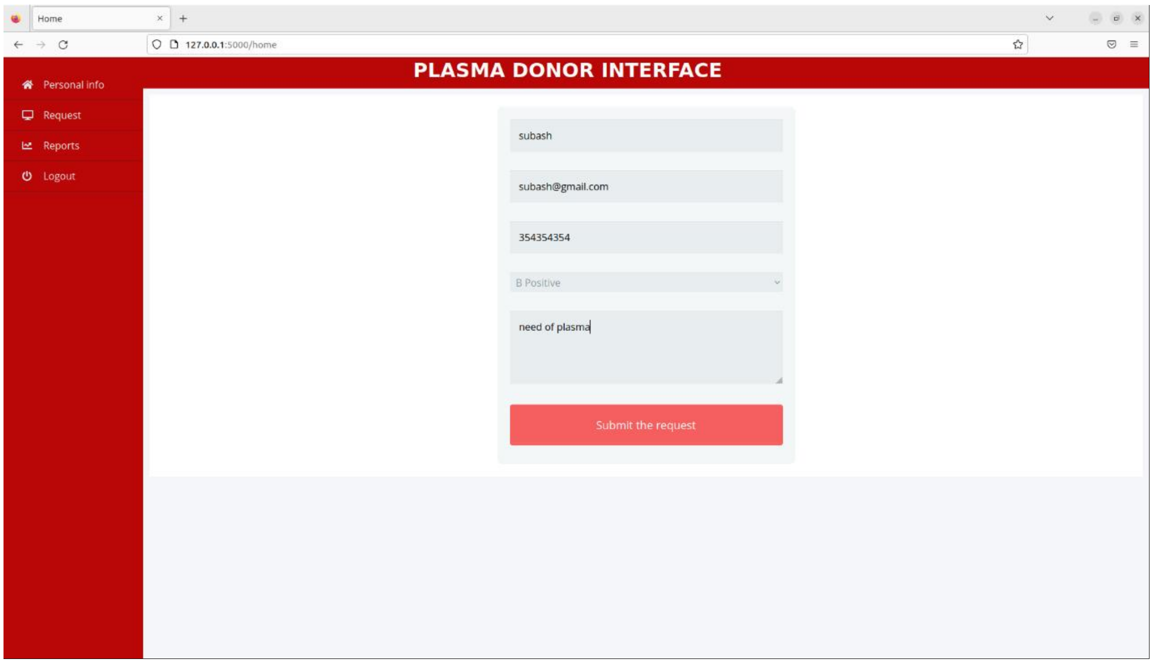
RESULTS

9.1 PERFORMANCE METRICS

Login Page :



Interface :



subash

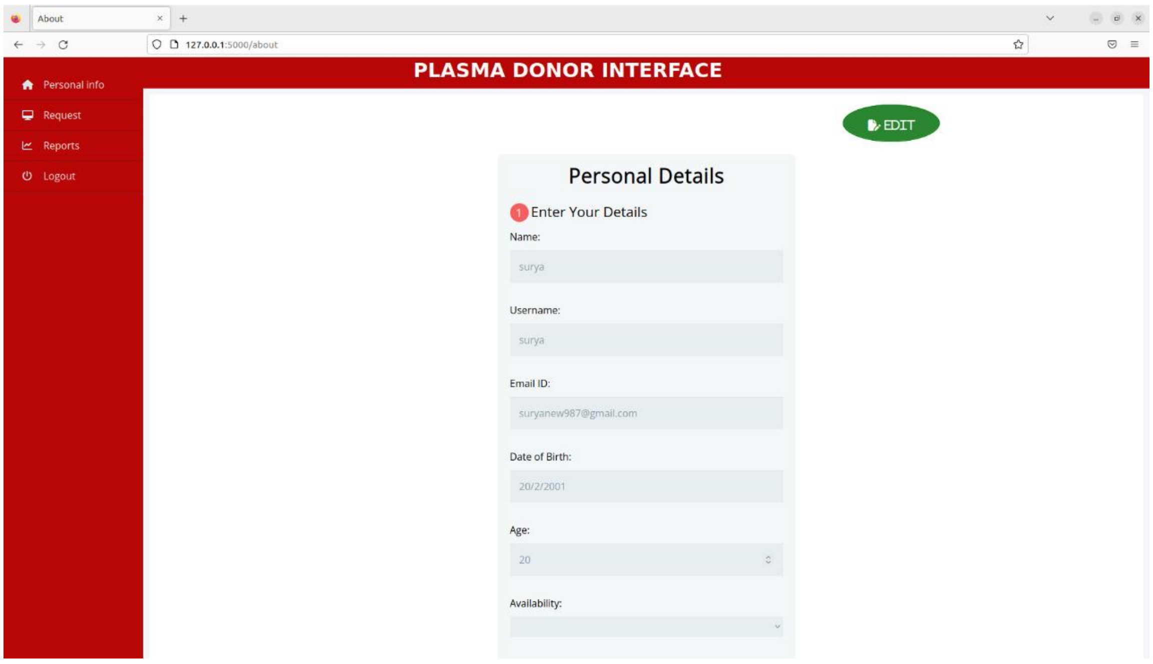
subash@gmail.com

354354354

B Positive

need of plasma

Personal Detail page:



Personal Details

1 Enter Your Details

Name: surya

Username: surya

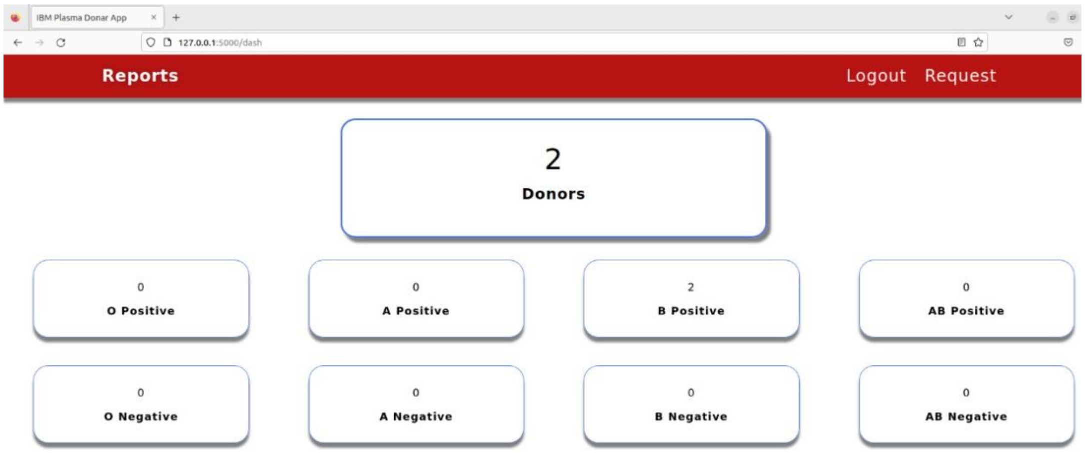
Email ID: suryanew987@gmail.com

Date of Birth: 20/2/2001

Age: 20

Availability:

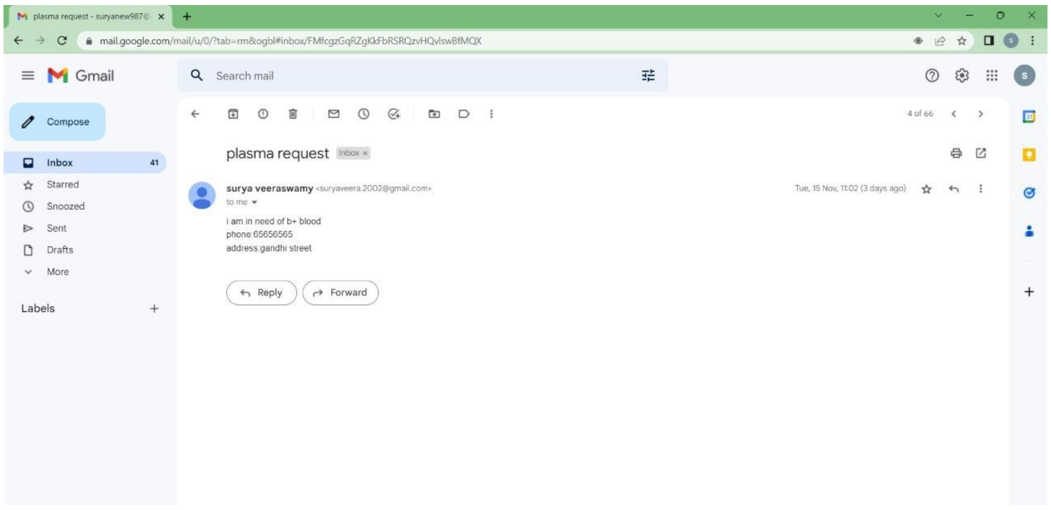
Donor Reports :



2 Donors

0 O Positive	0 A Positive	2 B Positive	0 AB Positive
0 O Negative	0 A Negative	0 B Negative	0 AB Negative

Sending Mail :


plasma request - suryanew987@
mail.google.com/mail/u/0/?tab=rm&ogbl#inbox/FMfcgzGqRZgKkFbRSRQzvHQvlswBfMQX
Gmail
Search mail
Compose
Inbox 41
Starred
Snoozed
Sent
Drafts
More
Labels
4 of 66

plasma request Inbox ×

surya veeraswamy <suryaveera.2002@gmail.com>
to me
Tue, 15 Nov, 11:02 (3 days ago)

i am in need of b+ blood
phone:65656565
address:gandhi street

CHAPTER 10

ADVANTAGES & DISADVANTAGES

Advantages

- Analysing clinical data to improve medical research
- Using patient data to improve health outcomes
- Gaining operational insights from healthcare provider data
- Improved staffing through health business management analytics
- Research and prediction of disease.
- Automation of hospital administrative processes.
- Early detection of disease.
- Prevention of unnecessary doctor's visits.
- Discovery of new drugs.
- More accurate calculation of health insurance rates.
- More effective sharing of patient data.

Disadvantages

Replacing Medical Personnel

Application of technology in every sphere of human life is improving the way things are done. These technologies are are also posing some threat to world of works. Robotics are replacing human labour.

Data Safety

Data security is another challenge in applying big data in healthcare. Big data storage is usually targets of hackers. This endangers the safety of medical data. Healthcare organisations are very much concerned about the safety of patients' sensitive personal data. For this, all healthcare applications must meet the requirement for data security and be HIPAA compliant before they can be deployed for healthcare services

Privacy

One of the major drawbacks in the application of big data in healthcare industry is the issue of lack of privacy. Application of big data technologies involves monitoring of patient's data, tracking of medical inventory and assets, organizing collected data, and visualization of data on the dashboard and the reports. So visualization of sensitive medical data especially that of the patients creates negative impression of big data as it violets privacy

Man Power

`Applying big data solutions in healthcare requires special skills, and such kills are scarce. Handling of big data requires the combination of medical, technological and statistical knowledge.

CHAPTER 11

CONCLUSION

Data analytics is the science of analysing raw datasets in order to derive a conclusion regarding the information they hold. It enables us to discover patterns in the raw data and draw valuable information from them.To some, the domain of healthcare data analytics may look new, but it has a lot of potential, especially if you wish to engage in challenging job roles and build a strong data analytics profile in the upcoming years. In this blog, we have covered some of the major topics such as what is healthcare data analytics, its applications, scope, and benefits, etc. We hope it helps you in your decision-making as a healthcare data analytics professional.

CHAPTER 12

FUTURE SCOPE

The Future of Healthcare, Intel provides a foundation for big data platforms and AI to advance health analytics. Predictive data analytics is helping health organizations enhance patient care, improve outcomes, and reduce costs by anticipating when, where, and how care should be provided.The future of big datain healthcare will be determined by technological breakthroughs from 2022 to 2030. Complete patient care and cost-effective prescription procedures are required for population health management. To assess clinical and claims data, they must be combined on the same platform.

Countries around the world have started to invest more capital in medical infrastructure, pharmaceuticals, and healthcare smart analytics solutions. The market is growing and will continue to expand, given the benefits of healthcare data analytics. It has also risen as a good career option for fresh data science and data analytics graduates or professionals who wish to build their career in the healthcare sector. Due to the sensitivity of the profession, the salary offers for healthcare data analysts are lucrative around the world. Apart from the remuneration, the opportunities to work with some of the biggest names in the healthcare sector is also worth mentioning. Hence, healthcare data analytics is growing to be one of the most rewarding branches of data analytics in the coming future.

CHAPTER 13

APPENDIX

13.1 Source Code

Home.html

```
<!DOCTYPE html>
<html lang="en" dir="ltr">
  <head>
    <meta charset="utf-8" />
    <title>Home</title>

    <link
      rel="stylesheet"
      href="https://use.fontawesome.com/releases/v5.8.1/css/all.css"
    />
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <link
      rel="stylesheet"
      href="{{ url_for('static',filename='css/style.css') }}"
    />
  </head>
  <body>
    {% extends "template.html" %} {% block content %}
    <form action="{{ url_for('requested')}}" method="post">
      <input
        type="text"
        name="name"
        placeholder="Enter Name"
        required="required"
        style="color: black"
      />
      <input
        type="email"
        name="email"
        placeholder="Enter Email"
        required="required"
        style="color: black"
      />
      <input
        type="text"
        name="phone"
        placeholder="Enter 10-digit mobile number"
```

```
            required="required"
            style="color: black"
          />
          <select name="bloodgrp">

      <option value="select" selected>Choose your blood group</option>
            <option value="O+">O Positive</option>
            <option value="A+">A Positive</option>
            <option value="B+">B Positive</option>
            <option value="AB+">AB Positive</option>
            <option value="O-">O Negative</option>
            <option value="A-">A Negative</option>
            <option value="B-">B Negative</option>
            <option value="AB-">AB Negative</option>
          </select>
          <textarea
            rows="4"
            placeholder="Enter the address"
            required="required"
            style="color: black"
            name="address"
          ></textarea>
          <button type="submit" class="btn btn-primary btn-block btn-large">
            Submit the request
          </button>
        </form>

        <div>{{ pred }}</div>

        {% endblock %}
      </body>
    </html>
```

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

load_dotenv()

app = Flask(_name_)

try:

conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=54a2
f15b-5c0f-46df-8954-
7e38e612c2bd.c1ogj3sd0tgtu0lqde00.databases.appdomain.clou
d;PORT=32733;SECURITY=SSL;SSLServerCertificate=DigiCe
rtGlobalRootCA.crt;PROTOCOL=TCPIP;UID=ssz03200;PWD
```

```
=ROEcnQjP3TOa1Yxw;", "", "")
  print("success")
except:
  print(ibm_db.conn_errormsg())

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



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

```
# 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 uname =?"
        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
             deonor(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 deonor 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"))
```

13.2 GITHUB & PROJECT DEMO LINK

https://github.com/IBM-EPBL/IBM-Project-24388-1659942338