PROJECT REPORT

PLASMA DONOR APPLICATION

Submitted By:

Bauma Ranjith S
(813819106014)

Bharani Kumar S
(813819106015)

Deepikha R
(813819106021)

Harithaa S
(813819106034)

CONTENTS

CHAPTER 1

INTRODUCTION

The Project describes the Plasma Donor Application ,which is a online web based one. This report will help you to know in deep the actual work that has been done as teamwork. The main objective of this application is to automate the complete operations of the donation of Plasma. Nowadays, many people requesting for plasma in need. Such that, it needs to maintain hundreds and thousands of records. In order to search for the plasma in an effective and simple way, we created this application which will be easy for the user to get plasma at right time.

1.1 PROJECT OVERVIEW

In our project, we used some of the popular software technologies which include languages like Python and Flask a Framework written in Python, in order to create a PLASMA DONOR APPLICATION.

The technologies include Python along with its framework Flask, to design a website, we used HTML and CSS and for creating, storing, and deleting a database, IBM Cloud is used.

Python is a computer programming language often used to **build websites and software, automate tasks, and conduct data analysis**. Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems.

Flask is a popular **micro framework** for building web applications. Since it is a micro-framework, it is very easy to use

1.2 PURPOSE

During the COVID 19 crisis, the requirement of plasma has 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. With respect to the problem faced, an application needs to be built which would take the donor details to store and inform them upon a request.

Hence, the main goal of our project is to design a user-friendly web application that serves human welfare. Thereby, we have all the information, you will ever need . The Donor can help recipients by registering on our website. The person who needs a blood donor, can search and find blood donors. After searching a list of donors will be displayed and user users brief details about their contact details, and email including their location, at last, they can communicate.

CHAPTER -2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

There are a quite good number of software packages that exist for the PLASMA DONOR APPLICATION system. But when I visited most plasma donor center system portals. I found the that existing system is limited only to those particular plasma center.

Problem Found In Existing System

- At present there is no software to keep any records in the plasma center.
- It becomes difficult to provide any record immediately at times of emergency.
- Required more human efforts in maintaining branch-related information. Manual keeping the accounts is also a tedious & risky job & maintaining those accounts in ledgers for a long period is also very difficult.
- Difficult to manage and maintain the files.
- Chance of damage to files, if the data is stored in the files for a duration of time.
- Time-consuming is retrieving, storing, and updating the data.
- It is difficult to keep track of the record of the donor & receiver he has donated or received the plasma at the last time.

2.2 REFERENCES

CASE STUDY - I

Name: Blood Bank Information System Based on Cloud Computing in Indonesia

Author: Muhammad Nur Sahid Ramadhan

Published year: July 2019 ICCOMSET 2018

Basic description:

Cloud computing can facilitate the controlled distribution of blood by using an information system. The problems that occur in the distribution of blood at the hospitals in Indonesia are sometimes the distributions are not controlled, so the blood may run out of stock caused by inequality of blood needs, difficulty in finding blood
donors, and don't have an integrated system. The purpose of this study is to propose a connecting system that integrates Indonesian society who can be a personal donors to help blood supply availability of UTD PMI using BBIS (Blood Bank Management System) based on cloud computing
.

Highlights of this project:

- Information management plays a major role in the blood donor system
- Cloud computing helps in giving immediate info to the recipient

Disadvantages of the project:

Overall, the current system still manual methods, and there is no interconnection using applications based on web service or mobile service. This proposed system
refers to digital transformation.

Limitations of the project:

- Cloud computing infrastructure Cloud computing is a model for enabling portable and convenient on-demand network access to a shared pool of configurable computing resources such as network, server, storage, application, and services

- Cloud computing also accommodates data storage with large capacity, the system allocating resources can be managed easily and data or information can be provided easily to access in real time. IaaS also can be hosted for data storage.

• virtual data stored can be found in the BBIS(blood bank info system) which can be accessed through a virtual machine and various integrated applications that can be accessed by both providers and users Only 40 lakh units of blood is available out of about 4 crore units of blood in each year in the nation.

• Direct contact donor and the recipient is not offered by many of the blood banks. This is critical when there is an emergency in need of blood.

• This aims to provide a direct link between the donor and the recipient. In this paper the blood donation management system (BDMS) through the web application.

References of this paper

[1]. Sulaiman, S., Abdul Hamid, A. A. K., & Najihah Yusri, N. A. (2015). Development of a Blood Bank Management System. Procedia - Social and Behavioral Sciences, 195, 2008-2013.

[2] Li, B. N., Chao, S., & Dong, M. C. (2007). SIBAS: A blood bank information system and its 5- year
implementation at Macau. Computers in Biology and Medicine, 37(5), 588-597

CASE STUDY - II

Name: Blood Donating System Web Application Project Software Requirement Specification.

Author: Benjamin King

Published year: April 27, 2022

Basic description:

A Blood Donating System Web Application can help many people to donate blood those who are in need. So, they will have a pure registration, with necessary details, and those who need blood will log in and check for the donors' list based on their locality and contact facility will be given such that both will be in contact and, they shall do the process necessary for blood donation.

Highlights of this project:

- Help many people to donate blood to those who are in need.
- In a very short span, it provides users with many facilities.
- Developing a website for blood donation.
- There are three phases involved in this project.
- Development process functional and non-functional process are major for the project to succeed.

Disadvantages of the project:

- Less Security
- Internet Connection Required

Limitations of the project:

- only web-based system available
- No Mobile based system available.

CASE STUDY - III

Name: REVIEW ON BLOOD BANK MANAGEMENT SYSTEMS

Author: Mohit

Published year: April 2021

Basic description:

In blood bank management systems there is a database that has all information of blood donors and blood banks so whenever any receiver wants blood then the system checks all compatible blood availability in the database and works according to that.

Highlights of this project:

- Contact donor via Toll-free no
- compare all existing Blood bank management systems
- Login to the app and get information about blood donors
- System to manage information about donors and patients and only authorized personnel have the authority to use that information

Disadvantages of the project:

- Less security.
- Internet connection required.

Limitations of the project:

- Loss of Data due to mismanagement.
- save information in the form of an excel sheet.
- lacks data security & leads to error-prone results.

References of this paper

G. Muddu Krishna; S. Nagaraju(2016),"Design and implementation of short message service (SMS) based blood bank", 2016 International Conference on Inventive Computation Technologies (ICICT) Muhammad Arif; S. Sreevas; K. Nafseer; R. Rahul (2012) "Automated online Blood bank database",
2012 Annual IEEE India Conference (INDICON) "Benefits of Management Information System in Blood Bank" by

1, Vikas Kulshreshtha, 2, Dr. Sharad Maheshwari [1], Research Scholar,[2], Associate Professor 2 1, Singhania University, Jhunjhunu, Rajasthan, India 2, Government Engineering College Jhalawar, Rajasthan, India [4] The Optimization of Blood Donor Information and Management System by Technopedia P. Priya1, V. Saranya2, S. Shabana3, Kavitha Subramani 4 Department of Computer Science and Engineering, Panimalar Engineering College, Chennai, India 1, 2, 3, 4 [5] Anish Hamlin M R, Albert Mayan J (2016), "Blood Donation And Life Saver-Blood Donation App", 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT) [6] "Android Blood Bank" by Prof. Snigdha1, Varsha Anabhavane2, Pratiksha lokhande3, Siddhi Kasar4, Pranita More5 Lecturer, Information Technology, Atharva College of Engineering, Mumbai, India 1 Student, Information Technology, Atharva College of Engineering, Mumbai, India 2,3,4,5 [7] "A Study on Blood Bank Management System" by A. Clemen Teena, K. Sankar and S. Kannan, Department of MCA, Bharath University, Selaiyur, Chennai-73, Tamil Nadu, India

CASE STUDY - IV

Name: A New Concept of Blood Bank Management System using Cloud Computing for Rural Area (INDIA)

Author: Javed Akhtar Khan and M.R. Alony

Published year: February 2015

Basic description:

An efficient blood bank management system using a cloud computing concept with mobile SMS facilities should be developed, with the aim of ensuring that every patient has access to an adequate quantity of safe blood in a centralized manner. The
The management system should solve the issue of demand and wastage and lead to self-sufficiency in blood requirements. This should encourage new donors and retain old donors to donate blood.

Highlights of this project:

we want to revisit autonomic computing, which defines a set of architectural characteristics to manage a system, where complexity is increasing but must be managed without increasing cost or the size of the management team, where a system must be quickly adaptable to new technologies integrated into it, and where a system must be extensible from within corporation out to the broader ecosystem and vice versa.

Limitations of the project:

- Absence of a GPS System for tracking the current location of the user.

References of this paper

[1]. Articles from Asian Journal of Transfusion Science are provided here courtesy of Medknow Publications Asian J Transfus Sci. 2009 July; 3(2): 57– 59. doi: 10.4103/0973-6247.53871 N. Choudhury.
“http://www.ncbi.nlm.nih.gov/pmc/articles/PMC29204 72/”
[2]. Benefits of Management Information System in Blood Bank Vikas Kulshreshtha, 2, Dr. Sharad Maheshwari RESEARCH INVENTY: International Journal Of Engineering And Science ISSN: 2278-4721, Vol. 1, Issue (12December 2012), PP 05-07 Www.Researchinventy.Com.
[3]. [3]. 91-US-31-1_Cloud_Computing White Paper Cloud Computing: Thin clients in the clouds.
[4] Bharat BloodBank,http://www.bharatbloodbank.com 3.Jeevan BloodBank,http://www.jeevan.org/11
[5] http://www.ehow.com/about_556819101_importance- blood-banks.html

[6] http://www.cancer.org/treatment/treatmentsandsideeffects/treatmenttypes/bloodproductdonationandtransfusion/blood-transfusion-and-donationdonating-blood
[7] http://seminarprojects.com/Thread-blood-bank- management-system advantages-and- disadvantages#ixzz2pmvg7qv4 [8] www.wikipedia.com [9] Buecker, Lodewijkx, Moss, Skapinetz, Waidner,

2.3 PROBLEM SOLUTION DEFINITION

PROBLEM STATEMENT – I

PROBLEM STATEMENT – II

PROBLEM STATEMENT – III

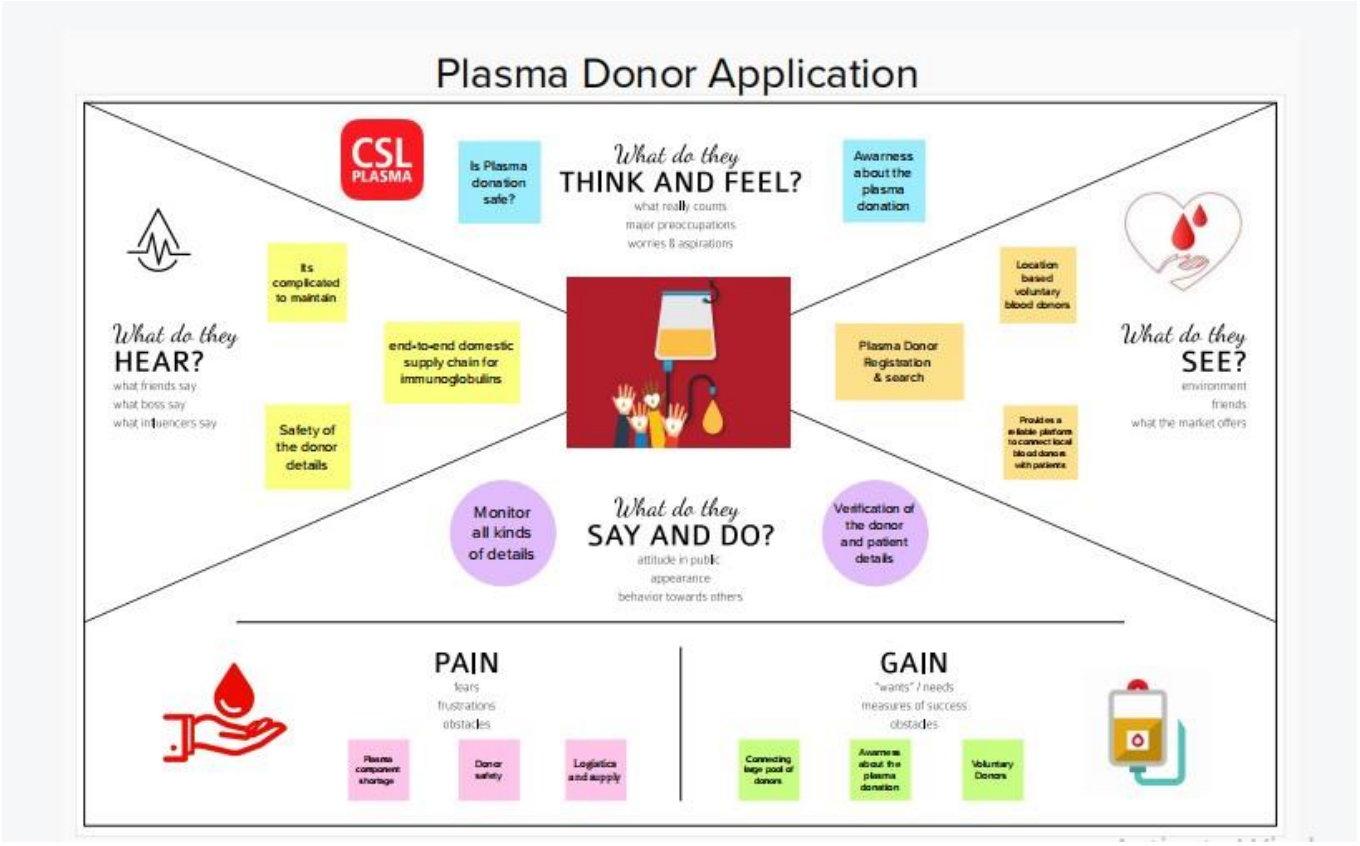
PROBLEM STATEMENT – IV

PROBLEM STATEMENT – V

CHAPTER 3

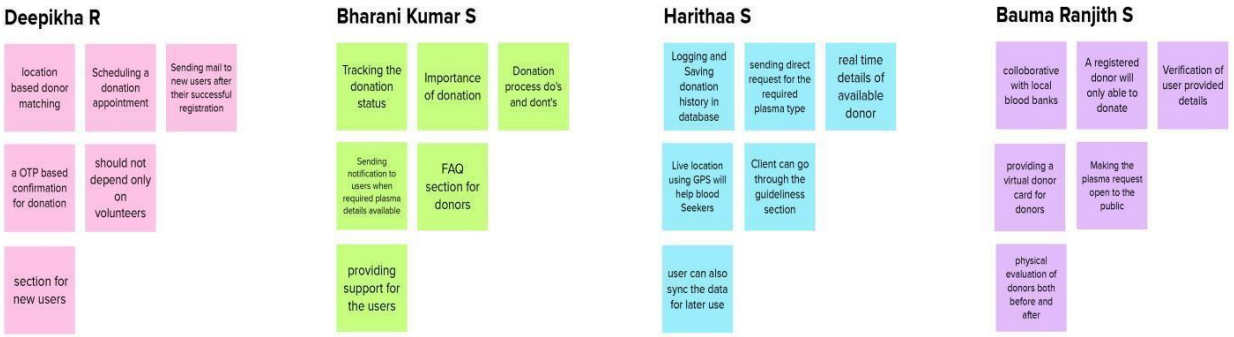
IDEATION AND PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS



3.2 IDEATION AND BRAINSTORMING

Step-1: Brainstorm, Idea Listing and Grouping



Team ID : PNT2022TMID32723
Team leader : Deepikha R

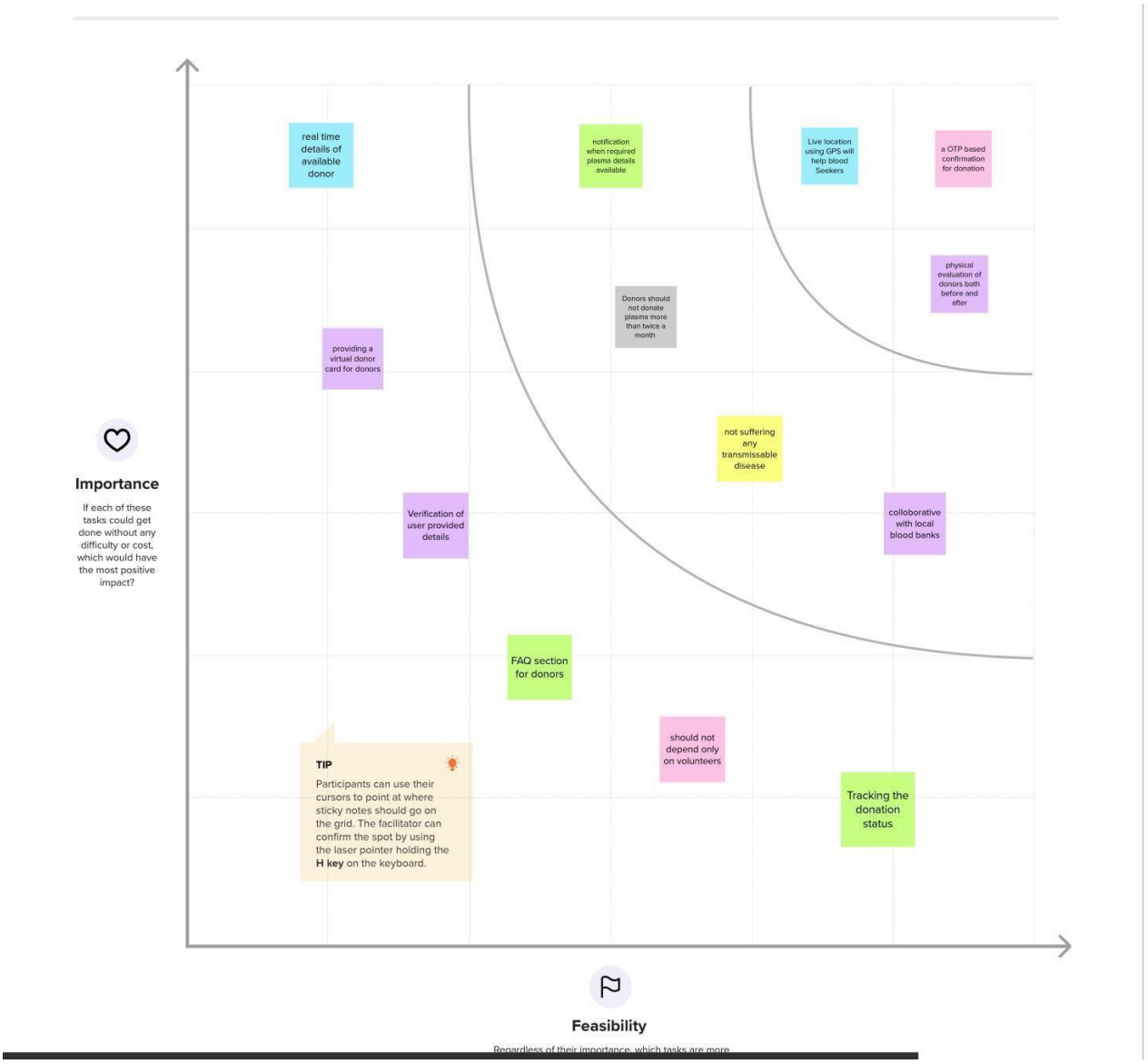
Team member : Bauma Ranjith S

Team member : Bharani Kumar S

Team member : Harithaa S


IDEA DESCRIPTION:
TIP
Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.
PROCESS:
Verification of user provided details
real time details of available donor
Make a donation appointment
physical evaluation of donors both before and after
Donation process do's and dont's
DESCRIPTION:
location based donor matching
user can also sync the data for later use
Sending mail to new users after their successful registration
sending direct request for the required plasma type
notify users when required plasma details available
FUTURE SCOPE:
colloborative with local blood banks
Making the plasma request open to the public
should not depend only on volunteers
FEATURES:
user can also sync the data for later use
a OTP based confirmation for donation
Tracking the donation status
online chat service between donors and consumers
Sending mail to new users after their successful registration
Scheduling a donation appointment
estimated locations of the donors,hopitals and blood banks
sending direct request for the required plasma type
ADDITIONAL FEATURES:
Importance of donation
Donation process do's and dont's
User friendly
providing a virtual donor card for donors
confidential and secure medical reports .Improved medical service delivery.
HELP SECTION:
FAQ section for Donors
section for new users
Importance of donation
guidelines section to view the useful precautions needed before and after blood transfusion.
physical evaluation of donors both before and after

Step-2: Idea Prioritization



3.3 PROPOSED SOLUTION

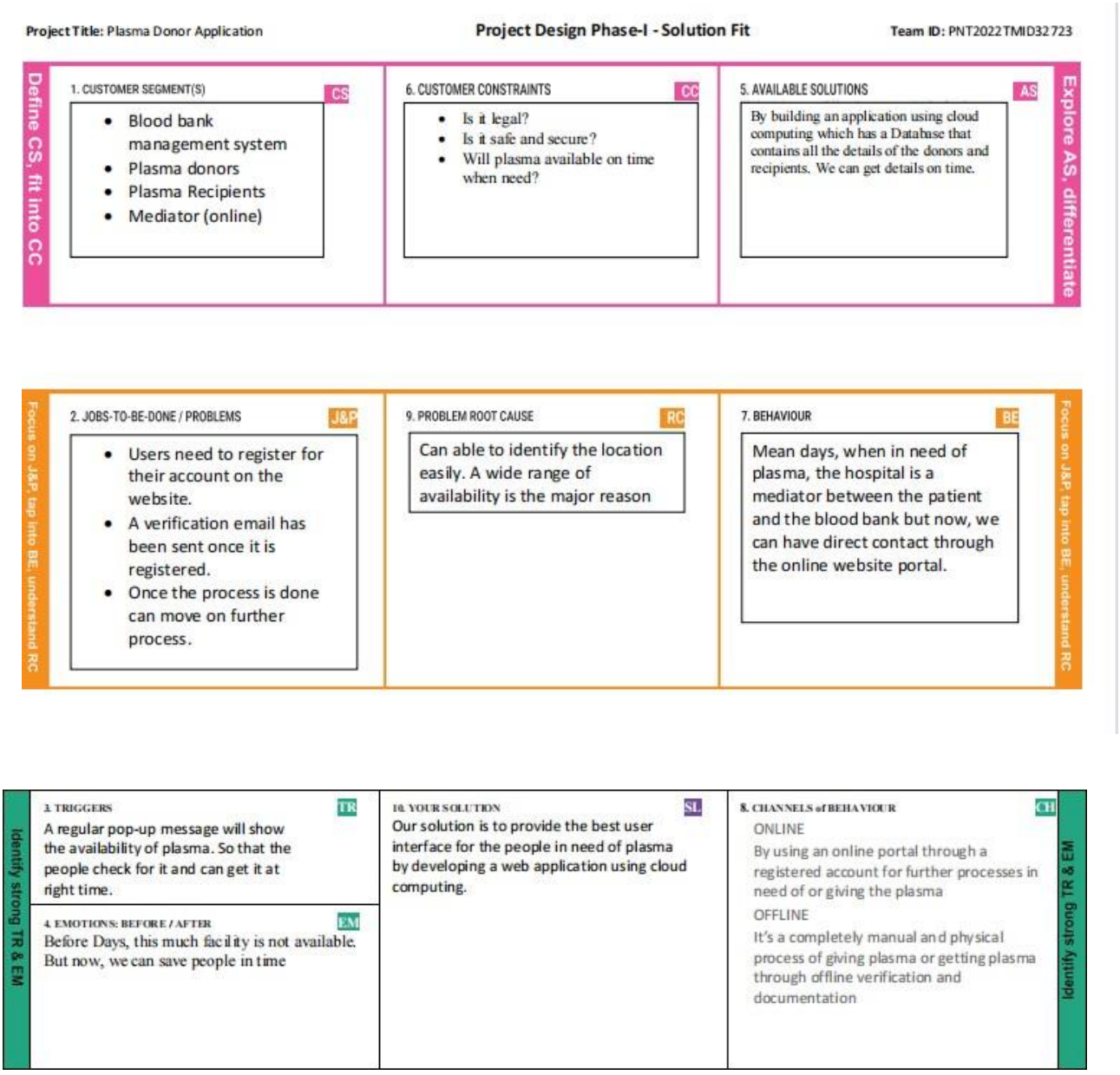
Proposed solution

S.No.	Parameter	Description
1.	Problem Statement (Problem to besolved)	Plasma plays a critical role in maintaining healthy blood pressure, blood volume, and a proper pH balance,the mostimportant element for humans to survive. Developing an application to interconnect plasma requester and donors will help people to save their lives during an emergency byusing the cloud-based techniques.
2.	Idea / Solution description	In day-to-day life requirement for plasma became high, especially during the COVID-19 crisis. But the donor count was low. Saving the donor information and helping the needy by notifying the current donors would help in need. Itis very difficult to find the respective blood group donors when anyone is in need. Regarding the problem faced, an application is to be built which would take the donor detailsstore them and inform them upon request. And, for plasma donation centre, it is easy to find donors The basic solution is to create a centralized system to keep track of the upcoming as well as past Plasma Donation Events. The recommended solution is as follows: The application contains, · If the user wants to donate or receive, they mustregister with their details. · After successful registration of the user, an E-mail has been sent to the user. · Then the user will be directed to the home page. · They will be asked to press whether they will bedonors or receivers. · If the user is a donor, then he/she will fill out the donation interest form which includes their Name,blood group details, location, last time donated to date, phone number, and email id.

		· After filling out the donation form, he/she will beredirected to the page on which he/she can download the e certificate · If the user is a receiver, then he/she can see the list of donors available, and they can raise their requestand contact the donor directly.
3.	Novelty / Uniquen ess	· We help the donor to access the location of a bloodcentre which is nearby him/her. · We Notify them by sending confirmation emails after they get registered for the plasma donation and, we notify them once the appointment is fixed in the centre. · Furthermore, the GPS map option is available todirect the donor to the centre. · Somebody wants to donate blood and plasma, butthey don't know the way to donate then they use this application which will simple to use and it will save the lives of many people. · Today many of them have mobile phones they can install this application and use it to save the lives of people.
4.	Social Impact / Customer Satisfaction	By using this application, the user will experience a user- friendly and responsive interface and they get satisfactionby saving thousands of people's life.
5.	Business Model (Revenue Model)	This application is accessible to everyone,it is free. Because of the trouble in finding givers who match a specific blood bunch, this application empowers clients toenlist individuals who wish to give plasma and keep their data in a data set. Nowadays the need for plasma increases. Anyone with basic knowledge can access this app.This can be used anywhere anytime. The user's information is encrypted. We maintain this app by automation for savingadmin and user time. Users get profited as we take care of them even after the plasma donation by giving them hospitality details also, we have an FAQ section where the users can clarify their doubts.

6.	Scalabilit y ofthe Solution	· Whatever the requirements, the application provides a clear solution for the requirements whenthere is an emergency then a plasma request to send to everyone. · Once the donor is ready to donate receiver isnotified about the donation. · With this app,donors can know their eligibility todonate and making it easier to locate suitable · donors at right time

3.4 PROBLEM SOLUTION FIT



CHAPTER-4

REQUIREMENT ANALYSIS

4.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Website
FR-2	User Confirmation	Confirmation via Email Confirmation via Verification code
FR-3	User Login	Login using registered email ID once registered
FR-4	Request sent to Donor	In need of plasma, User can able to contact Donor
FR-5	Verification procedure	Contacting Donor through verification ,then furtherprocess is made
FR-6	Pop up Message	Availability of Updated information all time

4.2 Non - Functional Requirements:

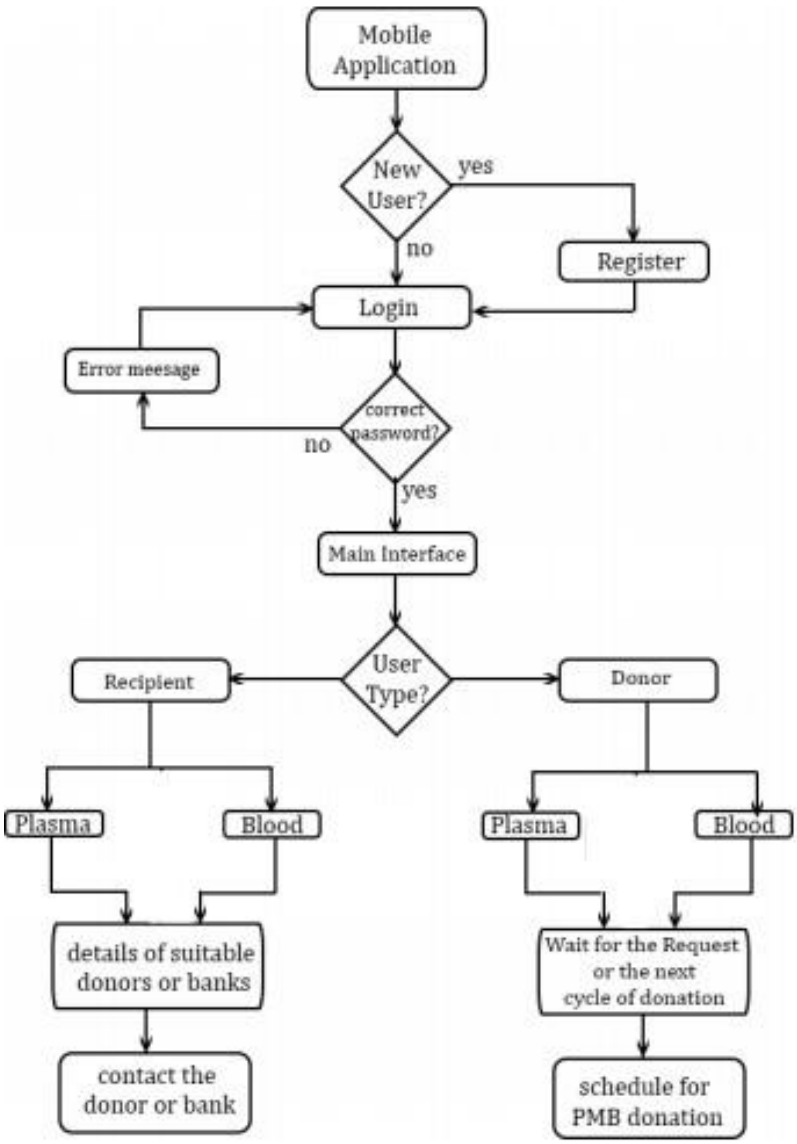
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The user Interface of the Plasma Donor Application should be able to make the user to think ,it is safe andsecure on usage
NFR-2	**Security**	Databases must be kept as secured so that users canable to view the details of Donors and recipient according to the user requirements. Meanwhile verification is done before any process will undergofurther.
NFR-3	**Reliability**	This website will provide intended service on time forall the Users without any negligence.
NFR-4	**Performance**	The process is much interaction with users in need ofclarifications.
NFR-5	**Availability**	This Website is available for 24/7
NFR-6	**Scalability**	There is interruption during the process of registrationwith or without network

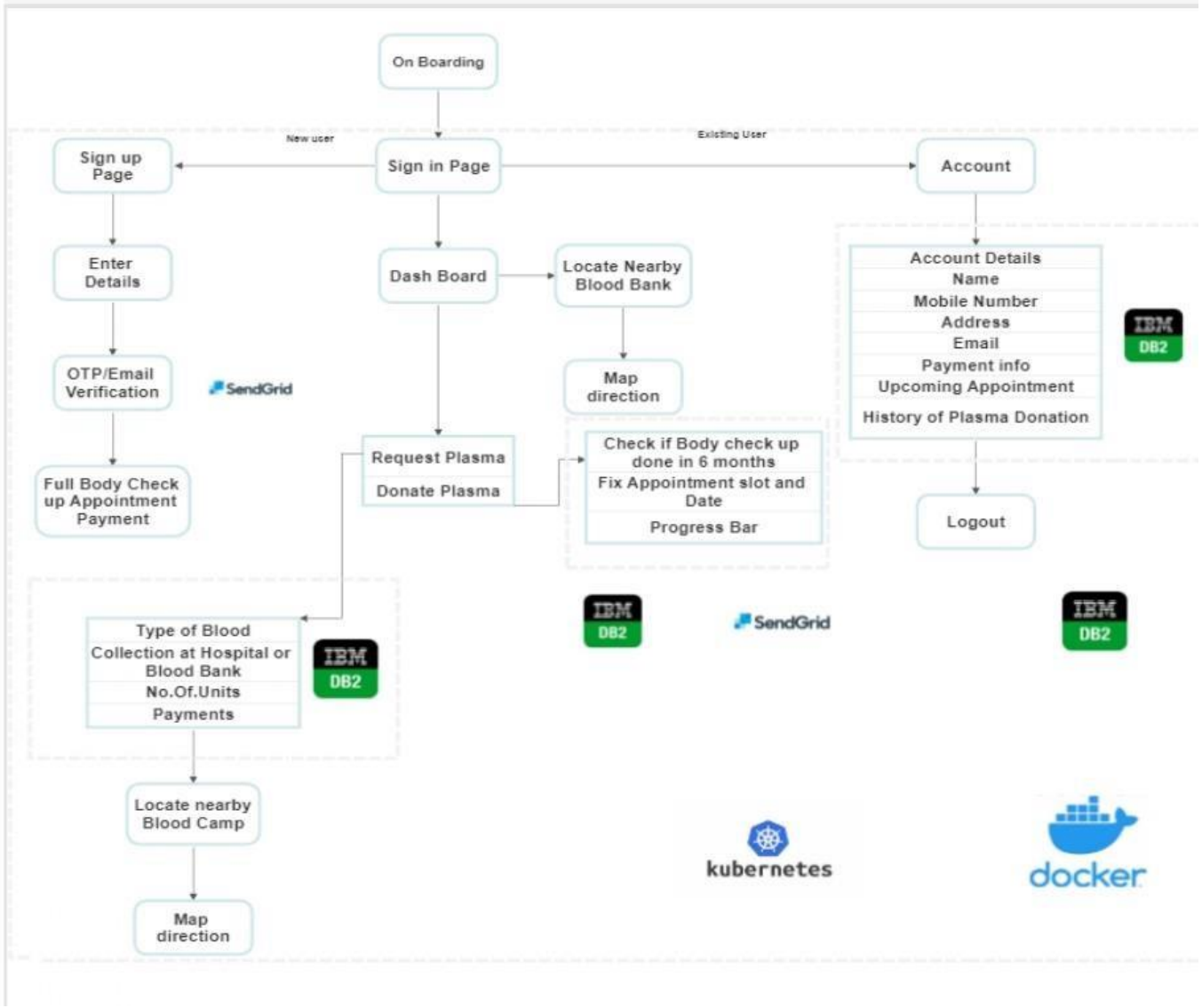
CHAPTER-5

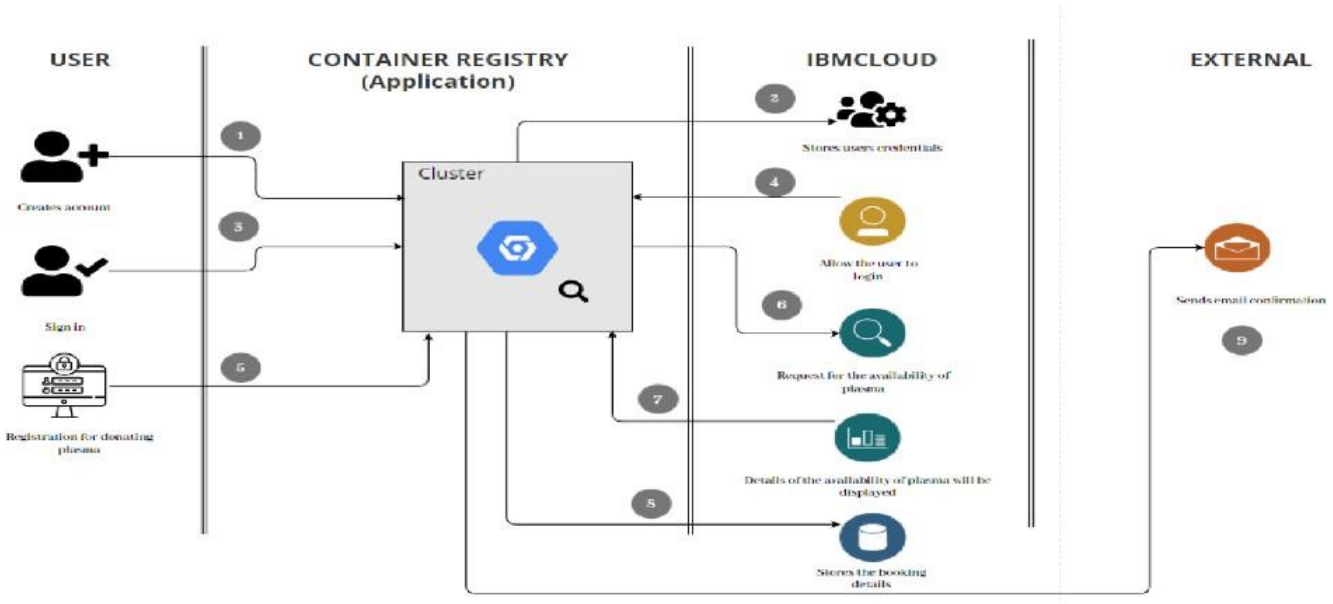
PROJECT DESIGN

5.1 DATA FLOW DIAGRAM



5.2 SOLUTION AND TECHNOLOGY ARCHITECTURE




USER
CONTAINER REGISTRY
(Application)
IBMCLOUD
EXTERNAL
Cluster
Creates account
Sign in
Registration for donating plasma
Stores users credentials
Allow the user to login
Request for the availability of plasma
Details of the availability of plasma will be displayed
Stores the booking details
Sends email confirmation
1
2
3
4
5
6
7
8
9

5.3 USER STORIES

User Stories

Use the below template to list all the user stories for the product.

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	I can access my account / dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Gmail		Medium	Sprint-2
	Login	USN-4	As a user, I can log into the application by entering email & password		High	Sprint-1
	Verification	USN-4	As a donor, I van verify my donor eligibility criteria	It can check my eligibility	Medium	Sprint-2
	Dashboard	USN-6	User can provide their personal details and location	I can complete my donor profile	low	Sprint-2
Customer (Plasma Receiver)	Registration	USN-1	As a receiver, it can register for the appointment by entering my email/phone number, password and confirming my password	It can create receiver account	High	Sprint-2
	Login	USN-2	Registered receiver can log into the application by entering username and password	It can access my account/dashboard	High	Sprint-2
	Verification	USN-3	As a receiver, I can enter my details to check the receiver eligibility criteria.	It can check my eligibility to receive plasma	medium	Sprint-2
Administrator	Login	USN-1	Admin can log into the application by entering email and password	It can access my account/dashboard	High	Sprint-1
	Dashboard	USN-2	Admin can modify add and remove features from the database and application	It can modify, add and remove features from the database and application	Low	Sprint-3

CHAPTER-6

SPRINT PLANNING AND ESTIMATION

6.1 Project Tracker, Velocity & Burndown Chart

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	5 Days	27 Oct 2022	01 Nov 2022	20	01 Nov 2022
Sprint-2	20	6 Days	02Oct 2022	07 Nov 2022	20	07 Nov 2022
Sprint-3	20	6 Days	08 Nov 2022	13 Nov 2022	20	13 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

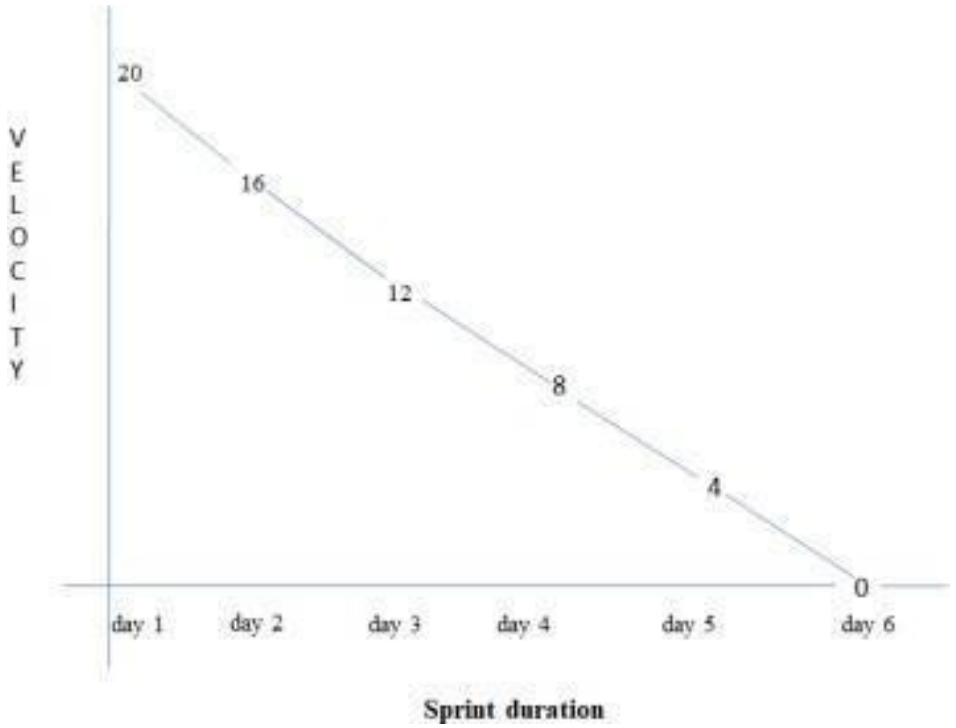
Velocity:

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) periteration unit (story points per day)

Sprint duration = 6 Days
Velocity of the team = 20
AV = 20 / 6 = 3.34
Average Velocity = 3.34

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time.It is often used in agile software development methodologies suchas Scrum. However, burn down charts can be applied to any project containing measurable progress over time.



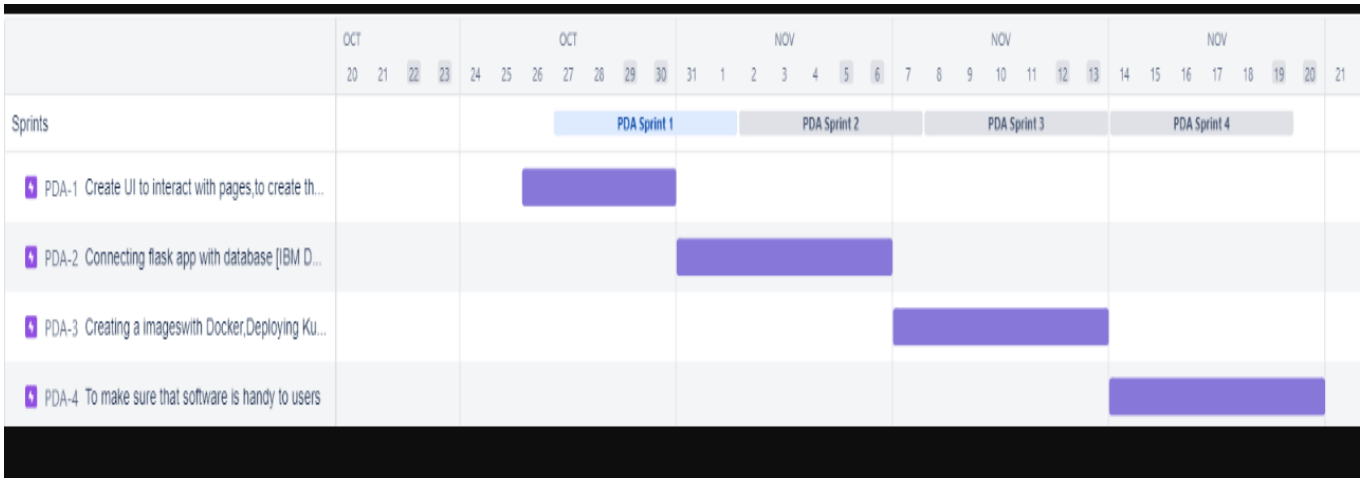
6.2 SPRINT DELIVERY SCHEDULE

Product Backlog, Sprint Schedule, and Estimation

Use the below template to create product backlog and sprint schedule

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration andLogin	USN-1	Create UI to interact with pages. To create theuser and admin login functionality	20	High	Bauma Ranjith S Harithaa S
Sprint-2	Cloud and Database	USN-2	Connecting flask app with database[IBMDB2] Implementation of IBM chatbot	20	High	Deepikha R Bauma Ranjith S
Sprint-3	Deployment in DevOps phase	USN-3	Creating images with docker, Deploying Kubernetes and adding the mailing service	20	High	Harithaa S Bharani Kumar S
Sprint-4	Testing and Deployment to user	USN-4	To make sure that the software is handy tousers.	20	High	Deepikha R Bharani Kumar S

6.3 REPORT FROM JIRA



CHAPTER-7

CODING AND SOLUTIONING

7.1 FEATURE CODE

Server:

```
from flask import render_template
import sqlite3
import requests
from flask import Flask
from flask import request,redirect,url_for,session,flash
from flask_wtf import Form
from wtforms import TextField
app = Flask(__name__)
app.secret_key = "super secret key"

@app.route('/')
def hel():
    conn = sqlite3.connect('database.db')
    print("Opened database successfully")
    conn.execute('CREATE TABLE IF NOT EXISTS users (name TEXT, addr TEXT, city TEXT, pin TEXT, bg TEXT,email TEXT UNIQUE, pass TEXT)')
    print( "Table created successfully")
    conn.close()
    if session.get('username')==True:
        messages = session['username']

    else:
        messages = ""
    user = {'username': messages}
    return redirect(url_for('index',user=user))


@app.route('/reg')
def add():
    return render_template('register.html')


@app.route('/addrec',methods = ['POST', 'GET'])
def addrec():
   msg = ""
   #con = None
   if request.method == 'POST':
      try:
         nm = request.form['nm']
         addr = request.form['add']
         city = request.form['city']
         pin = request.form['pin']
         bg = request.form['bg']
         email = request.form['email']
         passs = request.form['pass']

         with sqlite3.connect("database.db") as con:
             cur = con.cursor()
             cur.execute("INSERT INTO users (name,addr,city,pin,bg,email,pass) VALUES (?,?,?,?,?,?,?)",(nm,addr,city,pin,bg,email,passs) )
             con.commit()
             msg = "Record successfully added"



      except:
             con.rollback()
             msg = "error in insert operation"
```

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



    if request.method == 'POST':
        if session.get('username') is not None:
            messages = session['username']

        else:
            messages = ""
        user = {'username': messages}
        print(messages)
        val = request.form['search']
        print(val)
        type = request.form['type']
        print(type)
        if type=='blood':
            con = sqlite3.connect('database.db')
            con.row_factory = sqlite3.Row

            cur = con.cursor()
            cur.execute("select * from users where bg=?",(val,))
            search = cur.fetchall();
            cur.execute("select * from users ")

            rows = cur.fetchall();


            return render_template('index.html', title='Home', user=user,rows=rows,search=search)

        if type=='donorname':
            con = sqlite3.connect('database.db')
            con.row_factory = sqlite3.Row

            cur = con.cursor()
            cur.execute("select * from users where name=?",(val,))
            search = cur.fetchall();
            cur.execute("select * from users ")

            rows = cur.fetchall();


            return render_template('index.html', title='Home', user=user,rows=rows,search=search)



    if session.get('username') is not None:
        messages = session['username']

    else:
        messages = ""
    user = {'username': messages}
    print(messages)
```

```
    if session.get('username') is not None:
        messages = session['username']

    else:
        messages = ""
    user = {'username': messages}
    print(messages)
    if request.method=='GET':
        con = sqlite3.connect('database.db')
        con.row_factory = sqlite3.Row

        cur = con.cursor()
        cur.execute("select * from users ")

        rows = cur.fetchall();
        return render_template('index.html', title='Home', user=user, rows=rows)

    #messages = request.args['user']



@app.route('/list')
def list():
   con = sqlite3.connect('database.db')
   con.row_factory = sqlite3.Row

   cur = con.cursor()
   cur.execute("select * from users")

   rows = cur.fetchall();
   print(rows)
   return render_template("list.html",rows = rows)

@app.route('/drop')
def dr():
        con = sqlite3.connect('database.db')
        con.execute("DROP TABLE request")
        return "dropped successfully"

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

        email = request.form['email']
        password = request.form['pass']
        if email == 'admin@bloodbank.com' and password == 'admin':
            a = 'yes'
            session['username'] = email
            #session['logged_in'] = True
            session['admin'] = True
            return redirect(url_for('index'))
        #print((password,email))
```

```
            session['admin'] = True
            return redirect(url_for('index'))
        #print((password,email))
        con = sqlite3.connect('database.db')
        con.row_factory = sqlite3.Row

        cur = con.cursor()
        cur.execute("select email,pass from users where email=?",(email,))
        rows = cur.fetchall();
        for row in rows:
            print(row['email'],row['pass'])
            a = row['email']
            session['username'] = a
            session['logged_in'] = True
            print(a)
            u = {'username': a}
            p = row['pass']
            print(p)

            if email == a and password == p:
                return redirect(url_for('index'))
            else:
                return render_template('/login.html')
        return render_template('/login.html')
    else:
        return render_template('/')


@app.route('/logout')
def logout():
   # remove the username from the session if it is there
   session.pop('username', None)
   session.pop('logged_in',None)
   try:
       session.pop('admin',None)
   except KeyError as e:
       print("I got a KeyError - reason " +str(e))


   return redirect(url_for('login'))


@app.route('/dashboard')
def dashboard():
   totalblood=0
   con = sqlite3.connect('database.db')
   con.row_factory = sqlite3.Row

   cur = con.cursor()
   cur.execute("select * from blood")

   rows = cur.fetchall();
   for row in rows:
       totalblood  += int(row['qty'])

   cur.execute("select * from users")
   users = cur.fetchall();
```

```
cur = con.cursor()
cur.execute("select * from blood")

rows = cur.fetchall();
for row in rows:
    totalblood  += int(row['qty'])

cur.execute("select * from users")
users = cur.fetchall();

Apositive=0
Opositive=0
Bpositive=0
Anegative=0
Onegative=0
Bnegative=0
ABpositive=0
ABnegative = 0

print(rows)
cur.execute("select * from blood where type=?",('A+',))
type = cur.fetchall();
for a in type:
    Apositive += int(a['qty'])

cur.execute("select * from blood where type=?",('A-',))
type = cur.fetchall();
for a in type:
    Anegative += int(a['qty'])


cur.execute("select * from blood where type=?",('O+',))
type = cur.fetchall();
for a in type:
    Opositive += int(a['qty'])

cur.execute("select * from blood where type=?",('O-',))
type = cur.fetchall();
for a in type:
    Onegative += int(a['qty'])

cur.execute("select * from blood where type=?",('B+',))
type = cur.fetchall();
for a in type:
    Bpositive += int(a['qty'])


cur.execute("select * from blood where type=?",('B-',))
type = cur.fetchall();
for a in type:
    Bnegative += int(a['qty'])



cur.execute("select * from blood where type=?",('AB+',))
type = cur.fetchall();
for a in type:
```

```
    cur.execute("select * from blood where type=?",('AB+',))
    type = cur.fetchall();
    for a in type:
        ABpositive += int(a['qty'])


    cur.execute("select * from blood where type=?",('AB-',))
    type = cur.fetchall();
    for a in type:
        ABnegative += int(a['qty'])


    bloodtypestotal = {'apos': Apositive,'aneg':Anegative,'opos':Opositive,'oneg':Onegative,'bpos':Bpositive,'bneg':Bnegative,'abpos':ABpositive,'abneg':ABnegative}







    return render_template("requestdonors.html",rows = rows,totalblood = totalblood,users=users,bloodtypestotal=bloodtypestotal)


@app.route('/bloodbank')
def bl():
    conn = sqlite3.connect('database.db')
    print("Opened database successfully")
    conn.execute('CREATE TABLE IF NOT EXISTS blood (id INTEGER PRIMARY KEY AUTOINCREMENT, type TEXT, donorname TEXT, donorsex TEXT, qty TEXT, dweight TEXT, donoremail TE
    print( "Table created successfully")
    conn.close()
    return render_template('/adddonor.html')


@app.route('/addb',methods =['POST','GET'])
def addb():
    msg = ""
    if request.method == 'POST':
        try:
           type = request.form['blood_group']
           donorname = request.form['donorname']
           donorsex = request.form['gender']
           qty = request.form['qty']
           dweight = request.form['dweight']
           email = request.form['email']
           phone = request.form['phone']



           with sqlite3.connect("database.db") as con:
              cur = con.cursor()
              cur.execute("INSERT INTO blood (type,donorname,donorsex,qty,dweight,donoremail,phone) VALUES (?,?,?,?,?,?,?)",(type,donorname,donorsex,qty,dweight,email,ph
              con.commit()
              msg = "Record successfully added"
        except:
           con.rollback()
           msg = "error in insert operation"
```

```
    else:
        return render_template("rest.html",msg=msg)

@app.route("/editdonor/<id>", methods=('GET', 'POST'))
def editdonor(id):
    msg =""
    if request.method == 'GET':
        con = sqlite3.connect('database.db')
        con.row_factory = sqlite3.Row

        cur = con.cursor()
        cur.execute("select * from blood where id=?",(id,))
        rows = cur.fetchall();
        return render_template("editdonor.html",rows = rows)
    if request.method == 'POST':
        try:
           type = request.form['blood_group']
           donorname = request.form['donorname']
           donorsex = request.form['gender']
           qty = request.form['qty']
           dweight = request.form['dweight']
           email = request.form['email']
           phone = request.form['phone']



           with sqlite3.connect("database.db") as con:
              cur = con.cursor()
              cur.execute("UPDATE blood SET type = ?, donorname = ?, donorsex = ?, qty = ?,dweight = ?, donoremail = ?,phone = ? WHERE id = ?",(type,donorname,donorsex,q
              con.commit()
              msg = "Record successfully updated"
        except:
           con.rollback()
           msg = "error in insert operation"

        finally:
            flash('saved successfully')
            return redirect(url_for('dashboard'))
            con.close()

@app.route("/myprofile/<email>", methods=('GET', 'POST'))
def myprofile(email):
    msg =""
    if request.method == 'GET':


        con = sqlite3.connect('database.db')
        con.row_factory = sqlite3.Row

        cur = con.cursor()
        cur.execute("select * from users where email=?",(email,))
        rows = cur.fetchall();
        return render_template("myprofile.html",rows = rows)
    if request.method == 'POST':
        try:
```

```
    if request.method == 'POST':
        try:
            name = request.form['name']
            addr = request.form['addr']
            city = request.form['city']
            pin = request.form['pin']
            bg = request.form['bg']
            emailid = request.form['email']

            print(name,addr)



            with sqlite3.connect("database.db") as con:
                cur = con.cursor()
                cur.execute("UPDATE users SET name = ?, addr = ?, city = ?, pin = ?,bg = ?, email = ? WHERE email = ?",(name,addr,city,pin,bg,emailid,email) )
                con.commit()
                msg = "Record successfully updated"
        except:
            con.rollback()
            msg = "error in insert operation"

        finally:
            flash('profile saved')
            return redirect(url_for('index'))
            con.close()



@app.route('/contactforblood/<emailid>', methods=('GET', 'POST'))
def contactforblood(emailid):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        print("Opened database successfully")
        conn.execute('CREATE TABLE IF NOT EXISTS request (id INTEGER PRIMARY KEY AUTOINCREMENT, toemail TEXT, formemail TEXT, toname TEXT, toaddr TEXT)')
        print( "Table created successfully")
        fromemail = session['username']
        name = request.form['nm']
        addr = request.form['add']

        print(fromemail,emailid)
        conn.execute("INSERT INTO request (toemail,formemail,toname,toaddr) VALUES (?,?,?,?)",(emailid,fromemail,name,addr) )
        conn.commit()
        conn.close()
        flash('request sent')
        return redirect(url_for('index'))
    if request.method == 'POST':
        conn = sqlite3.connect('database.db')
        print("Opened database successfully")
        conn.execute('CREATE TABLE IF NOT EXISTS request (id INTEGER PRIMARY KEY AUTOINCREMENT, toemail TEXT, formemail TEXT, toname TEXT, toaddr TEXT)')
        print( "Table created successfully")
        fromemail = session['username']
        name = request.form['nm']
        addr = request.form['add']

        print(fromemail,emailid)
```

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


            conn = sqlite3.connect('database.db')
            print("Opened database successfully")
            conn.row_factory = sqlite3.Row

            cur = conn.cursor()
            cor = conn.cursor()
            cur.execute('select * from request where toemail=?',(session['username'],))
            cor.execute('select * from request where toemail=?',(session['username'],))
            row = cor.fetchone();
            rows = cur.fetchall();
            if row==None:
                return render_template('notifications.html')
            else:
                return render_template('notifications.html',rows=rows)
```

```python
@app.route('/deleteuser/<useremail>',methods=('GET', 'POST'))
def deleteuser(useremail):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from users Where email=?',(useremail,))
        flash('deleted user:'+useremail)
        conn.commit()
        conn.close()
        return redirect(url_for('dashboard'))


@app.route('/deletebloodentry/<id>',methods=('GET', 'POST'))
def deletebloodentry(id):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from blood Where id=?',(id,))
        flash('deleted entry:'+id)
        conn.commit()
        conn.close()
        return redirect(url_for('dashboard'))
```

```python
@app.route('/deleteuser/<useremail>',methods=('GET', 'POST'))
def deleteuser(useremail):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from users Where email=?',(useremail,))
        flash('deleted user:'+useremail)
        conn.commit()
        conn.close()
        return redirect(url_for('dashboard'))


@app.route('/deletebloodentry/<id>',methods=('GET', 'POST'))
def deletebloodentry(id):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from blood Where id=?',(id,))
        flash('deleted entry:'+id)
        conn.commit()
        conn.close()
        return redirect(url_for('dashboard'))

@app.route('/deleteme/<useremail>',methods=('GET', 'POST'))
def deleteme(useremail):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from users Where email=?',(useremail,))
        flash('deleted user:'+useremail)
        conn.commit()
        conn.close()
        session.pop('username', None)
        session.pop('logged_in',None)
        return redirect(url_for('index'))

@app.route('/deletenoti/<id>',methods=('GET', 'POST'))
def deletenoti(id):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from request Where id=?',(id,))
        flash('deleted notification:'+id)
        conn.commit()
        conn.close()
        return redirect(url_for('notifications'))



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

AddDonor:

```
<!doctype html>
{% extends "base.html" %}
{% block content %}
<center>

  {% with messages = get_flashed_messages() %}
  {%if messages%}
      {%for mess in messages%}
    <div class="alert alert-warning alert-dismissible fade show" role="alert">
  <strong>{{mess}}</strong>
  <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
</div>

      {%endfor%}
  {%endif%}
  {% endwith %}

<div class="card text-left mt-5" style="width: 35rem;">
  <div class="card-body">
      <form action = "{{url_for('addb')}}" method = "POST">
         <h3>Donor Information</h3>
         BLOOD Group<br>

         <select name="blood_group" class="form-control">
            <option value="O+" selected>O+</option>
           <option value="O-">O-</option>
           <option value="A+">A+</option>
           <option value="A-">A-</option>
           <option value="B+">B+</option>
           <option value="B-">B-</option>
           <option value="AB+">AB+</option>
           <option value="AB-">AB-</option>
         </select>

         <label for="name">Name</label>
         <input type = "text" name = "donorname" class="form-control" required/><br>

         <label for="gender">gender</label>
         <div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios1" value="male" checked>
  <label class="form-check-label" for="exampleRadios1">
    Male
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios2" value="female">
  <label class="form-check-label" for="exampleRadios2">
    Female
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios2" value="other">
  <label class="form-check-label" for="exampleRadios2">
    Other
  </label>
```

```
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios2" value="other">
  <label class="form-check-label" for="exampleRadios2">
    Other
  </label>
</div>

<label for="qty">qty</label>

        <input type = "text" name = "qty" class="form-control"  required/><br>
<label for="dweight">donor weight</label>

        <input type = "text" name ="dweight" class="form-control"  required/><br>

  <label for="email">Email</label>
        <input type = "text" name ="email" class="form-control"  required/><br>

     <label for="phone">Phone</label>
        <input type = "text" name ="phone" class="form-control"  required/><br>
        <input type = "submit" value = "submit" class="btn btn-primary" /><br>

     </form>
    </div>
  </div>
</center>
    {% endblock %}
```

Base:

```
<html>
    <head>
        {% if title %}
        <title>{{ title }} - Plasma Bank</title>
        {% else %}
        <title>Welcome to Plasma Bank</title>
        {% endif %}
        <link rel="shortcut icon" href="{{url_for('static', filename='d.ico')}}"/>
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm8
        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anor
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" cr
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="{{url_for('static', filename='numscroller-1.0.js')}}"></script>
<style>div.someclass {
background-size: cover;
}
}</style>

    </head>
    <body>



      {% if session['admin'] == True %}

      <nav class="navbar navbar-expand-lg navbar-dark" style="background-color: #b71c1c;">


        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="fals
          <span class="navbar-toggler-icon"></span>
        </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <a class="navbar-brand" href="/">Plasma Bank</a>
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
        </li>
      <li class="nav-item">
        <a class="nav-link" href="/dashboard">Dashboard</a>
      </li>

      <li class="nav-item">
        <a class="nav-link" href="/bloodbank">add Plasma</a>
      </li>
      </ul>
      <ul class="navbar-nav ml-auto">
        <li class="nav-item">
          <a class="nav-link" href="{{url_for('logout')}}">logout</a>
        </li>
        </ul>
    </div>

  </nav>

      {% elif session['logged_in'] == True %}
```

```
    <nav class="navbar navbar-expand-lg navbar-dark" style="background-color: #b71c1c;">


      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="fals
        <span class="navbar-toggler-icon"></span>
      </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <a class="navbar-brand" href="/">Plasma Bank</a>
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
      </li>
    <li class="nav-item">
      <a class="nav-link" href="{{url_for('myprofile',email=session['username'])}}">myprofile</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{{url_for('notifications')}}">notifications</a>
    </li>
    </ul>


    <ul class="navbar-nav ml-auto">

  <li class="nav-item">
    <a class="nav-link" href="#"> Hi, {{ session['username']}}</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="{{url_for('logout')}}">logout</a>
  </li>
  </ul>

 </div>

/nav>


    {%else%}

    <nav class="navbar navbar-expand-lg navbar-dark" style="background-color: #b71c1c;">


      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="fals
        <span class="navbar-toggler-icon"></span>
      </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <a class="navbar-brand" href="/">Plasma Bank</a>
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
      </li>
    <li class="nav-item">
      <a class="nav-link" href="/login">login</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{{url_for('add')}}">register</a>
    </li>
```

```
    {%endif%}




          {% block content %}

           {% endblock %}
           <script>
  window.watsonAssistantChatOptions = {
    integrationID: "83eee642-f2e7-40a1-93a0-ccb1488eac71", // The ID of this integration.
    region: "jp-tok", // The region your integration is hosted in.
    serviceInstanceID: "e45ccddc-9187-4027-920c-4ed65319101b", // The ID of your service instance.
    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" + (window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.
    document.head.appendChild(t);
  });
</script>

     </body>
</html>
```

Edit Donor:

```
{% extends "base.html" %}
{% block content %}

{%for row in rows%}
<center>
<div class="card text-left mt-5" style="width: 35rem;">
  <div class="card-body">
      <form action = "{{url_for('editdonor',id=row['id'])}}" method = "POST">
         <h3>Donor Information</h3>
         BLOOD Group<br>

         <select name="blood_group" class="form-control">
            <option value="O+" selected>O+</option>
           <option value="O-">O-</option>
           <option value="A+">A+</option>
           <option value="A-">A-</option>
           <option value="B+">B+</option>
           <option value="B-">B-</option>
           <option value="AB+">AB+</option>
           <option value="AB-">AB-</option>
         </select>

         <label for="name">Name</label>
         <input type = "text" name = "donorname" class="form-control" value="{{row['donorname']}}" required/><br>

         <label for="gender">gender</label>
         <div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios1" value="male" checked>
  <label class="form-check-label" for="exampleRadios1">
    Male
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios2" value="female">
  <label class="form-check-label" for="exampleRadios2">
    Female
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios2" value="other">
  <label class="form-check-label" for="exampleRadios2">
    Other
  </label>
</div>

<label for="qty">qty</label>

         <input type = "text" name = "qty" class="form-control" value="{{row['qty']}}" required/><br>
<label for="dweight">donor weight</label>

         <input type = "text" name ="dweight" class="form-control" value="{{row['dweight']}}" required/><br>

  <label for="email">Email</label>
         <input type = "text" name ="email" class="form-control" value="{{row['donoremail']}}" required/><br>

      <label for="phone">Phone</label>
         <input type = "text" name ="phone" class="form-control" value="{{row['phone']}}" required/><br>
```

Index:

```
<div class="container-fluid">
        <center>
        <div class="card text-left mt-3" style="width: 35rem;">
          <div class="card-body">

        <form class="form-group" action = "{{url_for('index')}}" method = "POST">
          <label for="sel"> Search by:</label>
          <select name="type" class="form-control" id="exampleFormControlSelect1" required>
            <option value="blood">blood group</option>
            <option value="donorname">donorname</option>
          </select><br>
         <input class="form-control mr-sm-2" type="search" name="search" placeholder="Search" aria-label="Search" required><br>
         <center><button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button></center>
    </form>
      </div>
    </div>
  </center>
</div>
    <div class="container-fluid mb-2 mt-2">
        {% if search!=NULL %}
        <nav aria-label="breadcrumb">
        <ol class="breadcrumb">
          <li class="breadcrumb-item active" aria-current="page">  Search results:
      </li>
        </ol>
      </nav>



          <center>
           {% for s in search %}
           <div class="card text-left" style="width: 70rem;">
             <div class="card-header"> {{s["name"]}} </div>
             <div class="card-body ml-3">
             <h5 class="card-title ml-3">  {{s['email']}} </h5>
             <h6 class="card-subtitle mb-2  ml-3 text-muted"> Blood Group: {{s['bg']}}</h6>
              <p class="card-text ml-3"> Address:
                 {{s["addr"]}}
                  {{ s["city"]}}
                 {{s['pin']}}
                 <br>


                 {% if session['username'] == s['email']%}

                 <span class="d-inline-block" data-toggle="popover" data-content="Disabled popover">
    <button class="btn btn-primary mt-2 content-justify-end" style="pointer-events: none;" type="button" disabled>contact for Plasma</button>
  </span>

                 {% elif session['logged_in'] == True %}

                 <button type="button" class="btn btn-primary mt-1 content-justify-end" data-toggle="modal" data-target="#exampleModalCenter">
   contact for Plasma
  </button>
              </p>
```

```
</div>
        <div class="container-fluid">
  <!-- Content here -->
  <nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item active" aria-current="page">Registered donores:
</li>
  </ol>
</nav>




        <center>
          {% for row in rows %}
          <div class="card text-left" style="width: 70rem;">
            <div class="card-header">
    {{row["name"]}}
  </div>
           <div class="card-body ml-3">
           <h5 class="card-title ml-3"> {{row['email']}}</h5>
           <h6 class="card-subtitle mb-2  ml-3 text-muted"> Blood Group: {{row['bg']}}</h6>
            <p class="card-text ml-3"> Address:
              {{row["addr"]}}
              {{ row["city"]}}
              {{row['pin']}}
              <br>
              {% if session['username'] == row['email']%}

              <span class="d-inline-block" data-toggle="popover" data-content="Disabled popover">
   <button class="btn btn-primary mt-2" style="pointer-events: none;" type="button" disabled>contact for Plasma</button>
 </span>

              {% elif session['logged_in'] == True %}
              <button type="button" class="btn btn-primary mt-1" data-toggle="modal" data-target="#exampleModalCenter">
  contact for Plasma
</button>
<div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalCenterTitle">contact for Plasma</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <form method="POST" action="{{url_for('contactforblood',emailid=row['email'])}}">
          <label for="name">Name</label>
          <input type ="text" name = "nm" class="form-control" required/>
        <label for="addr"> confirm your Address</label>
       <textarea name ="add" class="form-control" required></textarea>
       <button type="button" class="btn btn-secondary mt-1" data-dismiss="modal">Close</button>
        <button type="submit" class="btn btn-primary mt-1">send request</button>
      </div>
    </div>
```

List:

```
<!doctype html>
<html>
   <body>

      <table border = 1>
         <thead>
            <td>Name</td>
            <td>Address</td>
            <td>city</td>
            <td>Pincode</td>
            <td>bg</td>
            <td>email</td>
            <td>pass</td>
         </thead>

         {% for row in rows %}
            <tr>
               <td>{{row["name"]}}</td>
               <td>{{row["addr"]}}</td>
               <td> {{ row["city"]}}</td>
               <td>{{row['pin']}}</td>
               <td>{{row['bg']}}</td>
               <td>{{row['email']}}</td>
               <td>{{row['pass']}}</td>
            </tr>
         {% endfor %}
      </table>

      <a href = "/">Go back to home page</a>

   </body>
</html>
```

Login:

```
<br>
<br>
<center>
<div class="card text-left" style="width: 35rem;">
  <div class="card-body">

    <form action = "{{url_for('login')}}" method = "POST">
<div class="form-group">
  <label for="exampleInputEmail1">Email address</label>
  <input type="email"  name ="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email" required>
  <small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>
</div>
<div class="form-group">
  <label for="exampleInputPassword1">Password</label>
  <input type="password"  name ="pass" class="form-control" id="exampleInputPassword1" placeholder="Password" required>
</div>
<button type="submit" class="btn btn-primary">Login</button>
</form>

  </div>
</div>
</center>



{% endblock %}
```

My Profile:

```
<center>
<div class="card text-left mt-5" style="width: 35rem;">
  <div class="card-body">
      <form action = "{{url_for('myprofile',email=session['username'])}}" method = "POST">
        <div class="form-group">
         <h3>Donor Information</h3>


         <label for="name">name</label>
         <input type = "text"  class="form-control" name = "name" value="{{row['name']}}" required/><br>



         <label for="name">address</label>
         <input type = "text" name = "addr"  class="form-control" value="{{row['addr']}}" required/><br>

         <label for="name">city</label>
         <input type = "text" name ="city" class="form-control" value="{{row['city']}}" required/><br>

         <label for="name">pin</label>
         <input type = "text" name ="pin" class="form-control" value="{{row['pin']}}" required/><br>

         <label for="name">blood group</label>
         <input type = "text" name ="bg" class="form-control" value="{{row['bg']}}" required/><br>
         <label for="exampleInputEmail1">Email address</label>
        <input type = "email" name ="email" class="form-control" value="{{row['email']}}" required/><br>
         <input type = "submit" value = "submit" class="btn btn-primary" /><br>
       </div>
      </form>
      </div>
      </div>
      </center>



<center>
  <nav aria-label="breadcrumb  ">
    <ol class="breadcrumb mt-3 justify-content-center">
      <li class="breadcrumb-item active" aria-current="page">or</li>
    </ol>
  </nav>

 <div class="card text-white bg-dark mb-3 mt-5" style="max-width: 18rem;">
  <div class="card-header">Delete My account !!</div>
  <div class="card-body">
    <h5 class="card-title">Warning</h5>
    <p class="card-text">this will delete your account permanently</p>
      <a  class="btn btn-danger" data-toggle="modal" data-target="#exampleModalCenter">delete my account</a>
      <div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content text-dark ">
      <div class="modal-header">
```

```
<center>
  <nav aria-label="breadcrumb  ">
    <ol class="breadcrumb mt-3 justify-content-center">
      <li class="breadcrumb-item active" aria-current="page">or</li>
    </ol>
  </nav>

 <div class="card text-white bg-dark mb-3 mt-5" style="max-width: 18rem;">
  <div class="card-header">Delete My account !!</div>
  <div class="card-body">
    <h5 class="card-title">Warning</h5>
    <p class="card-text">this will delete your account permanently</p>
      <a  class="btn btn-danger" data-toggle="modal" data-target="#exampleModalCenter">delete my account</a>
      <div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content text-dark ">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalCenterTitle">Delete your account</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">

        are you sure you want to delete your account

      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
        <a  class="btn btn-danger" href = "{{url_for('deleteme',useremail=row['email'])}}">delete my account</a>
      </div>
    </div>
  </div>
</div>
  </div>
</div>

</center>

{%endfor%}
```

Notification:

```
        <div class="card text-left border-secondary mb-3 mt-2 mx-auto" style="width: 70rem;">

          <div class="card-header border-secondary">from {{row['formemail']}} </div>

         <div class="card-body text-success ml-3">




                  <h5 class="card-title ml-3">{{row['toname']}} sent you a Plasma request
                his address is </h5>
                  <p class="card-text ml-3">


                       {{row['toaddr']}}

                    </p>
                    <div class="d-flex justify-content-end">
                  <a class="btn btn-outline-secondary" href = "{{url_for('deletenoti',id=row['id'])}}">delete notification</a>
                    </div>
                    </div>

                    </div>

{% endfor %}
{% else %}
<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h1 class="display-4">No Notifications</h1>
    <p class="lead">sorry you have ho new notifications</p>
  </div>
</div>
```

Register:

```
<br>
<br>
<center>
<div class="card text-left" style="width: 35rem;">
  <div class="card-body">
       <div class="form-group">
      <form action = "{{url_for('addrec')}}" method = "POST">
         <h3>Register as Donor</h3>
         <label for="name">Name</label>
         <input type = "text" name = "nm" class="form-control" required/>

          <label for="addr">Address</label>
         <textarea name = "add" class="form-control" required></textarea>

          <label for="city">City</label>
         <input type = "text" name = "city" class="form-control" required/>

          <label for="pin">postal code</label>
         <input type = "text" name = "pin" class="form-control" required/>

         <label for="Bloodgroup">Blood Group</label>
         <select name="bg"  class="form-control" id="exampleFormControlSelect1">
           <option value="O+" selected>O+</option>
           <option value="O-">O-</option>
           <option value="A+">A+</option>
           <option value="A-">A-</option>
           <option value="B+">B+</option>
           <option value="B-">B-</option>
           <option value="AB+">AB+</option>
           <option value="AB-">AB-</option>
         </select>

         <label for="exampleInputEmail1">Email address</label>
         <input type = "text" name ="email" class="form-control" required/>

         <label for="exampleInputPassword1">Password</label>
         <input type = "password" name ="pass" class="form-control" required/>
         <br>
         <button type="submit" class="btn btn-primary">Register</button>
      </form>
      </div>
      </div>
      </center>
      {% with messages = get_flashed_messages() %}
```

Request Donor:

```
  <h4>Plasma donations and their details</h4>
</div>


        {% for row in rows %}
          <div class="card">
            <h5 class="card-header">{{row["donorname"]}}</h5>
            <div class="card-body">
              <h5 class="card-title">{{row["type"]}}</h5>
              <p class="card-text">{{ row["donoremail"]}}
              {{row['donorsex']}}
              {{row['qty']}} pints
              {{row['dweight']}}kg's
              {{row['phone']}}</p>
              <div class="row">
    <div class="col">
      <form action="{{url_for('editdonor',id=row['id'])}}" method="GET">
      <input type="submit" class="btn btn-primary" name="button" value="edit"/>
      </form>
    </div>
    <div class="col">
      <form action="{{url_for('deletebloodentry',id=row['id'])}}" method="GET">
      <input type="submit" name="button" class="btn btn-danger" value="delete entry"/>
      </form>
    </div>
  </div>


            </div>
          </div>




        {% endfor %}
<br>
        <nav aria-label="breadcrumb">
          <ol class="breadcrumb">
            <li class="breadcrumb-item active" aria-current="page">  Registered donores:</li>
          </ol>
        </nav>
<br>



       <table class="table">
         <thead>
    <tr>
      <th scope="col">name</th>
      <th scope="col">address</th>
      <th scope="col">city</th>
      <th scope="col">pin</th>
      <th scope="col">blood group</th>
      <th scope="col">email</th>
```

```
    </div>
    </div></div>
    <div class="col"><div class="card text-white bg-success mb-3" style="max-width: 18rem;">
      <div class="card-header">B positive</div>
    <div class="card-body">
      <h5 class="card-title"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.bpos}}' data-delay='5' data-increment='10'>{{bloodtypestotal.bpos}}</span> u

    </div>
    </div></div>
    <div class="col"><div class="card text-white bg-danger mb-3" style="max-width: 18rem;">
      <div class="card-header">B negative</div>
    <div class="card-body">
      <h5 class="card-title"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.bneg}}' data-delay='5' data-increment='10'>{{bloodtypestotal.bneg}}</span> u
      <p class="card-text"></p>
    </div>
    </div></div>

    <div class="w-100"></div>

    <div class="col"><div class="card text-white bg-warning mb-3" style="max-width: 18rem;">
      <div class="card-header">AB positive</div>
    <div class="card-body">
      <h5 class="card-title"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.abpos}}' data-delay='5' data-increment='10'>{{bloodtypestotal.abpos}}</span>

    </div>
    </div></div>
    <div class="col"><div class="card text-white bg-info mb-3" style="max-width: 18rem;">
      <div class="card-header">AB negative</div>
    <div class="card-body">
      <h5 class="card-title"><span class="count"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.abneg}}' data-delay='5' data-increment='10'>{{bloodtypes

    </div>
    </div></div>
    <div class="col"><div class="card bg-light mb-3" style="max-width: 18rem;">
      <div class="card-header">O positive</div>
    <div class="card-body">
      <h5 class="card-title"><span class="count"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.opos}}' data-delay='5' data-increment='10'>{{bloodtypest

    </div>
    </div></div>
    <div class="col"><div class="card text-white bg-dark mb-3" style="max-width: 18rem;">
      <div class="card-header">O negtive</div>
    <div class="card-body">
      <h5 class="card-title"><span class="count"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.oneg}}' data-delay='5' data-increment='10'>{{bloodtypest

    </div>
    </div></div>
  </div>
</div>



<div class="alert alert-primary" role="alert">
  <h4>Plasma donations and their details</h4>
</div>
```

```
<!doctype html>
{% extends "base.html" %}
{% block content %}

{% with messages = get_flashed_messages() %}
{%if messages%}
    {%for mess in messages%}
  <div class="alert alert-warning alert-dismissible fade show" role="alert">
<strong>{{mess}}</strong>
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
  <span aria-hidden="true">&times;</span>
</button>
</div>

    {%endfor%}
{%endif%}
{% endwith %}

     {% if session['logged_in'] == True %}

     {%else%}

     {%endif%}


    <br>

    <div class="card border-danger text-center">
      <div class="card-header">Total Plasma in Plasma bank</div>
      <div class="card-body text-danger">
        <h5 class="card-title"><span class='numscroller' data-min='1' data-max='{{totalblood}}' data-delay='5' data-increment='10'>{{totalblood}}</span> units</h5>
      </div>
    </div>

<br>



        <div class="container">
  <div class="row">
    <div class="col"><div class="card text-white bg-primary mb-3" style="max-width: 18rem;">
      <div class="card-header">A positive</div>
    <div class="card-body">
      <h5 class="card-title"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.apos}}' data-delay='5' data-increment='10'>{{bloodtypestotal.apos}}</span> u

    </div>
  </div></div>
    <div class="col"><div class="card text-white bg-secondary mb-3" style="max-width: 18rem;">
      <div class="card-header">A negative</div>
    <div class="card-body">
      <h5 class="card-title"><span class="count"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.aneg}}' data-delay='5' data-increment='10'>{{bloodtypest

    </div>
    </div></div>
    <div class="col"><div class="card text-white bg-success mb-3" style="max-width: 18rem;">
```

```
      <th scope="col">delete</th>
    </tr>

  </thead>

          {% for r in users %}
              <tr>
                  <td>{{r["name"]}}</td>
                  <td>{{r["addr"]}}</td>
                  <td> {{ r["city"]}}</td>
                  <td>{{r['pin']}}</td>
                  <td>{{r['bg']}}</td>
                  <td>{{r['email']}}</td>
                  <td><button type="button" class="btn btn-primary mt-1" data-toggle="modal" data-target="#exampleModalCenter">
    contact for Plasma
  </button>
  <div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalCenterTitle">contact for Plasma</h5>
          <button type="button" class="close" data-dismiss="modal" aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
        <div class="modal-body">
          <form method="POST" action="{{url_for('contactforblood',emailid=r['email'])}}">
            <label for="name">Name</label>
            <input type = "text" name = "nm" value="admin@bloodbank.com" class="form-control" required/>
          <label for="addr"> confirm your Address</label>
         <input type="text" name="add" class="form-control" value="admin's address" required></textarea>
         <button type="button" class="btn btn-secondary mt-1" data-dismiss="modal">Close</button>
          <button type="submit" class="btn btn-primary mt-1">send request</button>
        </div>
      </div>
    </div>
  </div>
</td>
                  <td><a href = "{{url_for('deleteuser',useremail=r['email'])}}" class="btn btn-danger">delete user</a></td>
              </tr>
          {% endfor %}
```

Reset:

```
<!doctype html>
<html>
    <body>

        result of addition : {{ msg }}
        <h2><a href = "\">go back to home page</a></h2>

    </body>
</html>
```

Responsive:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></sc
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" cro
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crosso

<style>
body, html {
    height: 100%;
    margin: 0;
}

.bg {
  /* Set rules to fill background */
  min-height: 100%;
  min-width: 1024px;

  /* Set up proportionate scaling */
  width: 100%;
  height: auto;

  /* Set up positioning */
  position: fixed;
  top: 0;
  left: 0;
}

@media screen and (max-width: 1024px) { /* Specific to this particular image */
  .bg {
    left: 50%;
    margin-left: -512px;   /* 50% */
  }
}
</style>
</head>
<body>
  <img src="photo.jpg" class="bg"/>

<p><font color='red'>This example creates a full page background image. Try to resize the browser window to see how it always will cover the full screen (when scrolled to top)

</body>
</html>
```

CHAPTER-8

TESTING

8.1 TEST CASE

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO1	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on Login/Signup button		1.Enter URL and click go 2.Click on Login/Signup button 3.Verify login/Singup popup displayed or not		Login/Signup page popup should display	Working as expected	Pass				
LoginPage_TC_OO2	UI	Home Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on Login/Signup button 3.Verify login/Singup popup with below UI elements: a.email text box b.password text box c.Login button d.New user? Create Register link		Application should show below UI elements: a.email text box b.password text box c.Login button. d.New user? Create Register link	Working as expected	Pass	Recover Password Feature not yet added		BUG-1234	
LoginPage_TC_OO3	Functional	Home page	Verify user is able to log into application with Valid credentials		1.Enter URL and click go 2.Click on Login/Signup button 3.Enter Valid email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: arun2@gmail.com password: arun1234	User should navigate to user account homepage	Working as expected	Pass				
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go. 2.Click on Login button 3.Enter InValid email in Email text box 4.Enter valid password in password text box 5.Click on login button	Username: arun@gmail password: arun1234	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
LoginPage_TC_OO4	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go 2.Click on Log in button 3.Enter Valid email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: deepi@gmail.com password: Testing123456789	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				
LoginPage_TC_OO5	Functional	Login page	Verify user is able to log into application with InValid credentials		1.Enter URL and click go 2.Click on Login button 3.Enter InValid email in Email text box 4.Enter Invalid password in password text box 5.Click on login button	Username: arun password: Testing123456789	Application should show 'Incorrect email or password ' validation message.	Working as expected	Pass				
HomePage_TC_OO6	Functional	Home page	Verify User is able to Sign in With his Details		1.Enter URL and click go 2.Click on Sign in button 3.Redirected to Sign in page 4.Enter valid password and username 5.Click on login button	name: arun password: arun1234 Address:2/24 sastri road city:Trichy Postalcode:620001 Blood Group:O+	Application must redirect to proper webpage without delay	Working as expected	Pass				
HomePage_TC_OO7	Functional	Home page	Verify User is able to Register With his Details		1.Enter URL and click go 2.Click on Login/Signup button 3.Enter Valid email in Email text box 4.Enter valid password in password text box 5.Click on login button	name: arun2 Address:2/24 sastri road city:Trichy Postalcode:620001 Blood Group:O+ Email address:arun2@gmail.com		Working as expected	Pass				

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
Register_TC_OO8	UI	Register Page	Verify the UI elements in Login/Signup popup		1.Enter URL and click go 2.Click on Login/Signup button 3.Verify login/Singup popup with below UI elements: a.Name b.email text box c.password text box d.Phone No e.Sex f.Age g.Blood	name:arun2 Address:2/24 sastri road city:Trichy Postalcode:620001 Blood Group:O+ Email address:arun2@gmail.com Password:arun12	Application should show below UI elements: a.Name b.address c.city d.Postalcode e.Blood f.Email address g.password Register	Working as expected	Pass				
Register_TC_OO9	Functional	Register	Verify that NewUser is able to		1.Enter URL and click go	Username: arun2@gmail.com	Application must redirect to proper	Working as	Pass				
Register_TC_OO10	Functional	Register Page	Verify that New User when registering with invalid details is prompted		1.Enter URL and click go 2.Click on Login/Signup button 3.Fill the required fills mentioned below: a.Name b.address c.city d.Postalcode e.Blood f.Email address g.password Register	name:arun2 Address:2/24 sastri road city:Trichy Postalcode:620001 Blood Group:O+ Email address:arun2@gmail.com Password:arun12	Application must redirect to the same page with prompts saying that fields are incorrect or not properly filled.	Working as expected	Pass				

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
Main_TC_0011	Functional	Main Page	User can search for Donor using name or Blood group	Successfull in search and produced correct result	search by: Name Blood group	Name:Ramesh	Application must show the required search result	Working as expected	Pass				
Main_TC_0012	Functional	Main Page	User can search for Donor using name or Blood group	Successfull in search and produced correct result	search by: Name Blood group	Blood group:B+	Application must show the required search result	Working as expected	Pass				
Main_TC_0012	Functional	Main Page	User can send request for plasma from the given set of Registered donors	Successful in sending request to the donors	Contact for Plasma: Name: Address:	Name:Amith Address: 7/45 Nsb road Chennai	Application must send the request to the given donor	Working as expected	Pass				
Main_TC_0013	Functional	Main Page	User can see his/her profile	Successful in displaying the Profile details	a.Name b.address c.city d.Postalcode e.Blood f.Email address	name:arun2 Address:2/24 sastri road city:Trichy Postalcode:620001 Blood Group:O+ Email	Application must display the profile details	Working as expected	Pass				
Main_TC_0014	Functional	Main Page	Verify that User Can Log out after his requirement or work is complete	Successfull Login/Register	1.After successfully login go to main page 2. Click on the LogOut button 3.Redirected to Home Page		Application must Log out the User from the system	Working as expected	Pass				
Main_TC_0015	Functional	Home page	Admin is able to log into application with Valid credentials	Successful Login	1.Enter URL and click go 2.Click on Login/Signup button 3.Enter Valid email in Email text box 4.Enter valid password in password text box	Username: adminbank@gmail.com password: admin	admin should navigate to admin account homepage	Working as expected	Pass				

Test case ID	Feature Type	Component	Test Scenario	Pre-Requisite	Steps To Execute	Test Data	Expected Result	Actual Result	Status	Commnets	TC for Automation(Y/N)	BUG ID	Executed By
Main_TC_0016	Functional	admin page	Admin dashboard should display the total Plasma stores in the bank	Successful in displaying the details	Dashboard			Working as expected	Pass				
Main_TC_0017	Functional	admin page	Admin can edit or delete the donation details and can view the updated dashboard	Successful in the editing and deleting the details	A.Blood group B.name C.gender D.qty E.Donor weight	A.Blood group:O+ B.name :Ajay C.gender :M D.qty :3 E.Donor weight :65	Application should saved the details and updated the dashboard successfully	Working as expected	pass				
Main_TC_0018	Functional	Home page	Admin can contact for Plasma using the given list of donor list	Successful in contacting the donor for Plasma	Contact for Plasma: Name: Address: Send request	Contact for Plasma: Name:adminbank@gmail.com Address: 2/20 abc road Send request	Application successfully sent the request to the specified donor	Working as expected	Pass				
Main_TC_0019	UI	Main Page	Verify the UIelements in Main Page	Successfull Login/Register	1.Enter URL and click go 2.Click on Login/Signup button 3.Verify login/Singup popup displayed or not		Login/Signup page popup should display	Working as expected	Pass				

Team ID	PNT2022TMID32723
Project Name	Plasma Donor Application
date	19-Nov-22

NFT - Risk Assessment

S.No	Project Name	Scope/feature	Functional Changes	Hardware Changes	Software Changes	Load/Voluem Changes	Risk Score
1	PDA LOGIN PAGE	New	Low	No Changes	Moderate	NO CHANGES	GREEN
2	PDA DASHBOARD	NEW	HIGH	No Changes	NO	LOW	GREEN
3	PDA PLASMA REQEUST	NEW	MODERATE	NO Changes	NO	LOW	GREEN

NFT - Detailed Test Plan

S.No	Project Overview	NFT Test approach	Approvals/SignOff
1	PDA LOGIN PAGE	Using python and flask	
2	PDA DASHBOARD	Using python and flask	
3	USER WEB APPLICATION	Using python and flask	

End Of Test Report

S.No	Project Overview	NFT Test approach	NFR - Met	Test Outcome	GO/NO-GO decision	Identified Defects (Detected/Closed/Open)	Approvals/SignOff
1	CCR LOGIN PAGE	Using python and flask	N0	Expectaions met	GO	Identified /closed	
2	CCR ADMIN PAGE	Using python and flask	NO	Expectation partially met	GO	indentified/rectified	

8.2 USER ACCEPTANCE TEST

1. Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the [Customer Care Registry] project at the time of the release to User Acceptance Testing (UAT).

2. Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and howthey were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	10	4	5	5	24
Duplicate	2	0	2	0	4
External	5	3	2	1	11
Fixed	15	5	5	10	35
Not Reproduced	0	0	0	0	0
Skipped	0	0	1	1	2
Won't Fix	0	5	2	1	8
Totals	32	17	17	18	84

3. Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested

Section	Total Cases	Not Tested	Fail	Pass
Print Engine	10	0	0	10
Client Application	40	0	0	40
Security	5	0	0	2
Outsource Shipping	3	0	0	3
Exception Reporting	10	0	0	10
Final Report Output	4	0	0	4
Version Control	4	0	0	4

CHAPTER-9

RESULT

9.1 PERFORMANCE METRICS

• **Formal code metrics** - Such as Lines of Code (LOC), code complexity, Instruction Path Length, etc. In modern development environments, these are considered less useful.

• **Developer productivity metrics**—Such as active days, assignment scope, Efficiency, and code churn. These metrics can help you understand how much time and work developers are investing in a software project.

• **Agile process metrics**—Such as lead time, cycle time, and velocity. They measure the progress of a DevOps team in producing working, shipping-quality software features.

• **Operational metrics**—Such as Mean Time Between Failures (MTBF) and Mean Time to Recover (MTTR). This checks how software is running in production and how effective operations staff are at maintaining it.

• **Test metrics**—Such as code coverage, percent of automated tests, and defects in production. This measures how comprehensively a system is tested, which should be correlated with software quality.

• **Customer satisfaction**—Such as Net Promoter Score (NPS), Customer Effort Score (CES) and Customer Satisfaction Score (CSAT). The ultimate measurement of how customers experience the software and their interaction with the software vendor.

CHAPTER – 10

ADVANTAGES AND DISADVANTAGES

Advantages

- **Speed**: This website is fast and offers great accuracy as compared to manual registered keeping.
- **Maintenance**: Less maintenance is required
- **User Friendly**: It is very easy to use and understand. It is easily workable and accessible for everyone.
- **Fast Results**: It would help you to provide plasma donors easily depending upon the availability of it.

Disadvantages

- **Internet**: It would require an internet connection for the working of the website.

CHAPTER-11

CONCLUSION

The efficient way of finding plasma donors for the infected people is implemented using the plasma donor website that is hosted by using Flask Framework. To ensure the smooth functioning of the website operations.

I have hosted the website with the help of Flask and Python together to make sure the operations are running successfully. Flask @app.route() function is used to deploy the application where the IBM_DB2 cloud-based service is used.

CHAPTER-12

FUTURE SCOPE

Upgrading the UI that is more user-friendly will help many users to access the website and also ensures that many plasma donors can be added into the community.

Using elastic load balance helps to handle multiple requests at the same time which will maintain the uptime of the website with negligible downtime.

CHAPTER-13

APPENDIX

13.1 SOURCE CODE:

AddDonor:

```
<!doctype html>
{% extends "base.html" %}
{% block content %}
<center>

  {% with messages = get_flashed_messages() %}
  {%if messages%}
      {%for mess in messages%}
    <div class="alert alert-warning alert-dismissible fade show" role="alert">
  <strong>{{mess}}</strong>
  <button type="button" class="close" data-dismiss="alert" aria-label="Close">
    <span aria-hidden="true">&times;</span>
  </button>
</div>

      {%endfor%}
  {%endif%}
  {% endwith %}

<div class="card text-left mt-5" style="width: 35rem;">
  <div class="card-body">
      <form action = "{{url_for('addb')}}" method = "POST">
         <h3>Donor Information</h3>
         BLOOD Group<br>

         <select name="blood_group" class="form-control">
            <option value="O+" selected>O+</option>
           <option value="O-">O-</option>
           <option value="A+">A+</option>
           <option value="A-">A-</option>
           <option value="B+">B+</option>
           <option value="B-">B-</option>
           <option value="AB+">AB+</option>
           <option value="AB-">AB-</option>
         </select>

         <label for="name">Name</label>
         <input type = "text" name = "donorname" class="form-control" required/><br>

         <label for="gender">gender</label>
         <div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios1" value="male" checked>
  <label class="form-check-label" for="exampleRadios1">
    Male
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios2" value="female">
  <label class="form-check-label" for="exampleRadios2">
    Female
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios2" value="other">
  <label class="form-check-label" for="exampleRadios2">
    Other
  </label>
```

```
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios2" value="other">
  <label class="form-check-label" for="exampleRadios2">
    Other
  </label>
</div>

<label for="qty">qty</label>

        <input type = "text" name = "qty" class="form-control"  required/><br>
<label for="dweight">donor weight</label>

        <input type = "text" name ="dweight" class="form-control"  required/><br>

  <label for="email">Email</label>
        <input type = "text" name ="email" class="form-control"  required/><br>

     <label for="phone">Phone</label>
        <input type = "text" name ="phone" class="form-control"  required/><br>
        <input type = "submit" value = "submit" class="btn btn-primary" /><br>

     </form>
    </div>
  </div>
</center>
    {% endblock %}
```

Base:

```
<html>
    <head>
        {% if title %}
        <title>{{ title }} - Plasma Bank</title>
        {% else %}
        <title>Welcome to Plasma Bank</title>
        {% endif %}
        <link rel="shortcut icon" href="{{url_for('static', filename='d.ico')}}"/>
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm8
        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anor
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49"
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" cr
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js"></script>
<script src="{{url_for('static', filename='numscroller-1.0.js')}}"></script>
<style>div.someclass {
background-size: cover;
}
}</style>

    </head>
    <body>



      {% if session['admin'] == True %}

      <nav class="navbar navbar-expand-lg navbar-dark" style="background-color: #b71c1c;">


        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="fals
          <span class="navbar-toggler-icon"></span>
        </button>
      <div class="collapse navbar-collapse" id="navbarNav">
        <a class="navbar-brand" href="/">Plasma Bank</a>
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
        </li>
      <li class="nav-item">
        <a class="nav-link" href="/dashboard">Dashboard</a>
      </li>

      <li class="nav-item">
        <a class="nav-link" href="/bloodbank">add Plasma</a>
      </li>
      </ul>
      <ul class="navbar-nav ml-auto">
        <li class="nav-item">
          <a class="nav-link" href="{{url_for('logout')}}">logout</a>
        </li>
        </ul>
    </div>

  </nav>

      {% elif session['logged_in'] == True %}
```

```
    <nav class="navbar navbar-expand-lg navbar-dark" style="background-color: #b71c1c;">


      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="fals
        <span class="navbar-toggler-icon"></span>
      </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <a class="navbar-brand" href="/">Plasma Bank</a>
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
      </li>
    <li class="nav-item">
      <a class="nav-link" href="{{url_for('myprofile',email=session['username'])}}">myprofile</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{{url_for('notifications')}}">notifications</a>
    </li>
    </ul>


    <ul class="navbar-nav ml-auto">

  <li class="nav-item">
    <a class="nav-link" href="#"> Hi, {{ session['username']}}</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="{{url_for('logout')}}">logout</a>
  </li>
  </ul>

 </div>

/nav>


    {%else%}

    <nav class="navbar navbar-expand-lg navbar-dark" style="background-color: #b71c1c;">


      <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="fals
        <span class="navbar-toggler-icon"></span>
      </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <a class="navbar-brand" href="/">Plasma Bank</a>
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" href="/">Home <span class="sr-only">(current)</span></a>
      </li>
    <li class="nav-item">
      <a class="nav-link" href="/login">login</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="{{url_for('add')}}">register</a>
    </li>
```

```
    {%endif%}




          {% block content %}

           {% endblock %}
           <script>
  window.watsonAssistantChatOptions = {
    integrationID: "83eee642-f2e7-40a1-93a0-ccb1488eac71", // The ID of this integration.
    region: "jp-tok", // The region your integration is hosted in.
    serviceInstanceID: "e45ccddc-9187-4027-920c-4ed65319101b", // The ID of your service instance.
    onLoad: function(instance) { instance.render(); }
  };
  setTimeout(function(){
    const t=document.createElement('script');
    t.src="https://web-chat.global.assistant.watson.appdomain.cloud/versions/" + (window.watsonAssistantChatOptions.clientVersion || 'latest') + "/WatsonAssistantChatEntry.
    document.head.appendChild(t);
  });
</script>

     </body>
</html>
```

Edit Donor:

```
{% extends "base.html" %}
{% block content %}

{%for row in rows%}
<center>
<div class="card text-left mt-5" style="width: 35rem;">
  <div class="card-body">
      <form action = "{{url_for('editdonor',id=row['id'])}}" method = "POST">
         <h3>Donor Information</h3>
         BLOOD Group<br>

         <select name="blood_group" class="form-control">
            <option value="O+" selected>O+</option>
           <option value="O-">O-</option>
           <option value="A+">A+</option>
           <option value="A-">A-</option>
           <option value="B+">B+</option>
           <option value="B-">B-</option>
           <option value="AB+">AB+</option>
           <option value="AB-">AB-</option>
         </select>

         <label for="name">Name</label>
         <input type = "text" name = "donorname" class="form-control" value="{{row['donorname']}}" required/><br>

         <label for="gender">gender</label>
         <div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios1" value="male" checked>
  <label class="form-check-label" for="exampleRadios1">
    Male
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios2" value="female">
  <label class="form-check-label" for="exampleRadios2">
    Female
  </label>
</div>
<div class="form-check">
  <input class="form-check-input" type="radio" name="gender" id="exampleRadios2" value="other">
  <label class="form-check-label" for="exampleRadios2">
    Other
  </label>
</div>

<label for="qty">qty</label>

         <input type = "text" name = "qty" class="form-control" value="{{row['qty']}}" required/><br>
<label for="dweight">donor weight</label>

         <input type = "text" name ="dweight" class="form-control" value="{{row['dweight']}}" required/><br>

  <label for="email">Email</label>
         <input type = "text" name ="email" class="form-control" value="{{row['donoremail']}}" required/><br>

      <label for="phone">Phone</label>
         <input type = "text" name ="phone" class="form-control" value="{{row['phone']}}" required/><br>
```

Index:

```
<div class="container-fluid">
        <center>
        <div class="card text-left mt-3" style="width: 35rem;">
          <div class="card-body">

        <form class="form-group" action = "{{url_for('index')}}" method = "POST">
          <label for="sel"> Search by:</label>
          <select name="type" class="form-control" id="exampleFormControlSelect1" required>
            <option value="blood">blood group</option>
            <option value="donorname">donorname</option>
          </select><br>
         <input class="form-control mr-sm-2" type="search" name="search" placeholder="Search" aria-label="Search" required><br>
         <center><button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button></center>
    </form>
      </div>
    </div>
  </center>
</div>
    <div class="container-fluid mb-2 mt-2">
        {% if search!=NULL %}
        <nav aria-label="breadcrumb">
        <ol class="breadcrumb">
          <li class="breadcrumb-item active" aria-current="page">  Search results:
      </li>
        </ol>
      </nav>



          <center>
           {% for s in search %}
           <div class="card text-left" style="width: 70rem;">
             <div class="card-header"> {{s["name"]}} </div>
             <div class="card-body ml-3">
             <h5 class="card-title ml-3">  {{s['email']}} </h5>
             <h6 class="card-subtitle mb-2  ml-3 text-muted"> Blood Group: {{s['bg']}}</h6>
              <p class="card-text ml-3"> Address:
                 {{s["addr"]}}
                  {{ s["city"]}}
                 {{s['pin']}}
                 <br>


                 {% if session['username'] == s['email']%}

                 <span class="d-inline-block" data-toggle="popover" data-content="Disabled popover">
    <button class="btn btn-primary mt-2 content-justify-end" style="pointer-events: none;" type="button" disabled>contact for Plasma</button>
  </span>

                 {% elif session['logged_in'] == True %}

                 <button type="button" class="btn btn-primary mt-1 content-justify-end" data-toggle="modal" data-target="#exampleModalCenter">
   contact for Plasma
  </button>
              </p>
```

```
</div>
        <div class="container-fluid">
  <!-- Content here -->
  <nav aria-label="breadcrumb">
  <ol class="breadcrumb">
    <li class="breadcrumb-item active" aria-current="page">Registered donores:
</li>
  </ol>
</nav>




        <center>
          {% for row in rows %}
          <div class="card text-left" style="width: 70rem;">
            <div class="card-header">
    {{row["name"]}}
  </div>
           <div class="card-body ml-3">
           <h5 class="card-title ml-3"> {{row['email']}}</h5>
           <h6 class="card-subtitle mb-2  ml-3 text-muted"> Blood Group: {{row['bg']}}</h6>
            <p class="card-text ml-3"> Address:
              {{row["addr"]}}
              {{ row["city"]}}
              {{row['pin']}}
              <br>
              {% if session['username'] == row['email']%}

              <span class="d-inline-block" data-toggle="popover" data-content="Disabled popover">
   <button class="btn btn-primary mt-2" style="pointer-events: none;" type="button" disabled>contact for Plasma</button>
 </span>

              {% elif session['logged_in'] == True %}
              <button type="button" class="btn btn-primary mt-1" data-toggle="modal" data-target="#exampleModalCenter">
  contact for Plasma
</button>
<div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalCenterTitle">contact for Plasma</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <form method="POST" action="{{url_for('contactforblood',emailid=row['email'])}}">
          <label for="name">Name</label>
          <input type ="text" name = "nm" class="form-control" required/>
        <label for="addr"> confirm your Address</label>
       <textarea name ="add" class="form-control" required></textarea>
       <button type="button" class="btn btn-secondary mt-1" data-dismiss="modal">Close</button>
        <button type="submit" class="btn btn-primary mt-1">send request</button>
      </div>
    </div>
```

List:

```
<!doctype html>
<html>
   <body>

      <table border = 1>
         <thead>
            <td>Name</td>
            <td>Address</td>
            <td>city</td>
            <td>Pincode</td>
            <td>bg</td>
            <td>email</td>
            <td>pass</td>
         </thead>

         {% for row in rows %}
            <tr>
               <td>{{row["name"]}}</td>
               <td>{{row["addr"]}}</td>
               <td> {{ row["city"]}}</td>
               <td>{{row['pin']}}</td>
               <td>{{row['bg']}}</td>
               <td>{{row['email']}}</td>
               <td>{{row['pass']}}</td>
            </tr>
         {% endfor %}
      </table>

      <a href = "/">Go back to home page</a>

   </body>
</html>
```

Login:

```
<br>
<br>
<center>
<div class="card text-left" style="width: 35rem;">
  <div class="card-body">

    <form action = "{{url_for('login')}}" method = "POST">
<div class="form-group">
  <label for="exampleInputEmail1">Email address</label>
  <input type="email"  name ="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email" required>
  <small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>
</div>
<div class="form-group">
  <label for="exampleInputPassword1">Password</label>
  <input type="password"  name ="pass" class="form-control" id="exampleInputPassword1" placeholder="Password" required>
</div>
<button type="submit" class="btn btn-primary">Login</button>
</form>

  </div>
</div>
</center>



{% endblock %}
```

My Profile:

```
<center>
<div class="card text-left mt-5" style="width: 35rem;">
  <div class="card-body">
      <form action = "{{url_for('myprofile',email=session['username'])}}" method = "POST">
        <div class="form-group">
         <h3>Donor Information</h3>


         <label for="name">name</label>
         <input type = "text"  class="form-control" name = "name" value="{{row['name']}}" required/><br>



         <label for="name">address</label>
         <input type = "text" name = "addr"  class="form-control" value="{{row['addr']}}" required/><br>

         <label for="name">city</label>
         <input type = "text" name ="city" class="form-control" value="{{row['city']}}" required/><br>

         <label for="name">pin</label>
         <input type = "text" name ="pin" class="form-control" value="{{row['pin']}}" required/><br>

         <label for="name">blood group</label>
         <input type = "text" name ="bg" class="form-control" value="{{row['bg']}}" required/><br>
         <label for="exampleInputEmail1">Email address</label>
        <input type = "email" name ="email" class="form-control" value="{{row['email']}}" required/><br>
         <input type = "submit" value = "submit" class="btn btn-primary" /><br>
       </div>
      </form>
      </div>
      </div>
      </center>



<center>
  <nav aria-label="breadcrumb  ">
    <ol class="breadcrumb mt-3 justify-content-center">
      <li class="breadcrumb-item active" aria-current="page">or</li>
    </ol>
  </nav>

 <div class="card text-white bg-dark mb-3 mt-5" style="max-width: 18rem;">
  <div class="card-header">Delete My account !!</div>
  <div class="card-body">
    <h5 class="card-title">Warning</h5>
    <p class="card-text">this will delete your account permanently</p>
      <a  class="btn btn-danger" data-toggle="modal" data-target="#exampleModalCenter">delete my account</a>
      <div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content text-dark ">
      <div class="modal-header">
```

```
<center>
  <nav aria-label="breadcrumb  ">
    <ol class="breadcrumb mt-3 justify-content-center">
      <li class="breadcrumb-item active" aria-current="page">or</li>
    </ol>
  </nav>

 <div class="card text-white bg-dark mb-3 mt-5" style="max-width: 18rem;">
  <div class="card-header">Delete My account !!</div>
  <div class="card-body">
    <h5 class="card-title">Warning</h5>
    <p class="card-text">this will delete your account permanently</p>
      <a  class="btn btn-danger" data-toggle="modal" data-target="#exampleModalCenter">delete my account</a>
      <div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered" role="document">
    <div class="modal-content text-dark ">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalCenterTitle">Delete your account</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">

        are you sure you want to delete your account

      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
        <a  class="btn btn-danger" href = "{{url_for('deleteme',useremail=row['email'])}}">delete my account</a>
      </div>
    </div>
  </div>
</div>
  </div>
</div>

</center>

{%endfor%}
```

Notification:

```
        <div class="card text-left border-secondary mb-3 mt-2 mx-auto" style="width: 70rem;">

          <div class="card-header border-secondary">from {{row['formemail']}} </div>

         <div class="card-body text-success ml-3">




                  <h5 class="card-title ml-3">{{row['toname']}} sent you a Plasma request
                his address is </h5>
                  <p class="card-text ml-3">


                       {{row['toaddr']}}

                    </p>
                    <div class="d-flex justify-content-end">
                  <a class="btn btn-outline-secondary" href = "{{url_for('deletenoti',id=row['id'])}}">delete notification</a>
                    </div>
                    </div>

                    </div>

{% endfor %}
{% else %}
<div class="jumbotron jumbotron-fluid">
  <div class="container">
    <h1 class="display-4">No Notifications</h1>
    <p class="lead">sorry you have ho new notifications</p>
  </div>
</div>
```

Register:

```
<br>
<br>
<center>
<div class="card text-left" style="width: 35rem;">
  <div class="card-body">
       <div class="form-group">
      <form action = "{{url_for('addrec')}}" method = "POST">
         <h3>Register as Donor</h3>
         <label for="name">Name</label>
         <input type = "text" name = "nm" class="form-control" required/>

          <label for="addr">Address</label>
         <textarea name = "add" class="form-control" required></textarea>

          <label for="city">City</label>
         <input type = "text" name = "city" class="form-control" required/>

          <label for="pin">postal code</label>
         <input type = "text" name = "pin" class="form-control" required/>

         <label for="Bloodgroup">Blood Group</label>
         <select name="bg"  class="form-control" id="exampleFormControlSelect1">
           <option value="O+" selected>O+</option>
           <option value="O-">O-</option>
           <option value="A+">A+</option>
           <option value="A-">A-</option>
           <option value="B+">B+</option>
           <option value="B-">B-</option>
           <option value="AB+">AB+</option>
           <option value="AB-">AB-</option>
         </select>

         <label for="exampleInputEmail1">Email address</label>
         <input type = "text" name ="email" class="form-control" required/>

         <label for="exampleInputPassword1">Password</label>
         <input type = "password" name ="pass" class="form-control" required/>
         <br>
         <button type="submit" class="btn btn-primary">Register</button>
      </form>
      </div>
      </div>
      </center>
      {% with messages = get_flashed_messages() %}
```

Request Donor:

```
  <h4>Plasma donations and their details</h4>
</div>


        {% for row in rows %}
          <div class="card">
            <h5 class="card-header">{{row["donorname"]}}</h5>
            <div class="card-body">
              <h5 class="card-title">{{row["type"]}}</h5>
              <p class="card-text">{{ row["donoremail"]}}
              {{row['donorsex']}}
              {{row['qty']}} pints
              {{row['dweight']}}kg's
              {{row['phone']}}</p>
              <div class="row">
    <div class="col">
      <form action="{{url_for('editdonor',id=row['id'])}}" method="GET">
      <input type="submit" class="btn btn-primary" name="button" value="edit"/>
      </form>
    </div>
    <div class="col">
      <form action="{{url_for('deletebloodentry',id=row['id'])}}" method="GET">
      <input type="submit" name="button" class="btn btn-danger" value="delete entry"/>
      </form>
    </div>
  </div>


            </div>
          </div>




        {% endfor %}
<br>
        <nav aria-label="breadcrumb">
          <ol class="breadcrumb">
            <li class="breadcrumb-item active" aria-current="page">  Registered donores:</li>
          </ol>
        </nav>
<br>



       <table class="table">
         <thead>
    <tr>
      <th scope="col">name</th>
      <th scope="col">address</th>
      <th scope="col">city</th>
      <th scope="col">pin</th>
      <th scope="col">blood group</th>
      <th scope="col">email</th>
```

```
    </div>
    </div></div>
    <div class="col"><div class="card text-white bg-success mb-3" style="max-width: 18rem;">
      <div class="card-header">B positive</div>
    <div class="card-body">
      <h5 class="card-title"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.bpos}}' data-delay='5' data-increment='10'>{{bloodtypestotal.bpos}}</span> u

    </div>
    </div></div>
    <div class="col"><div class="card text-white bg-danger mb-3" style="max-width: 18rem;">
      <div class="card-header">B negative</div>
    <div class="card-body">
      <h5 class="card-title"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.bneg}}' data-delay='5' data-increment='10'>{{bloodtypestotal.bneg}}</span> u
      <p class="card-text"></p>
    </div>
    </div></div>

    <div class="w-100"></div>

    <div class="col"><div class="card text-white bg-warning mb-3" style="max-width: 18rem;">
      <div class="card-header">AB positive</div>
    <div class="card-body">
      <h5 class="card-title"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.abpos}}' data-delay='5' data-increment='10'>{{bloodtypestotal.abpos}}</span>

    </div>
    </div></div>
    <div class="col"><div class="card text-white bg-info mb-3" style="max-width: 18rem;">
      <div class="card-header">AB negative</div>
    <div class="card-body">
      <h5 class="card-title"><span class="count"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.abneg}}' data-delay='5' data-increment='10'>{{bloodtypes

    </div>
    </div></div>
    <div class="col"><div class="card bg-light mb-3" style="max-width: 18rem;">
      <div class="card-header">O positive</div>
    <div class="card-body">
      <h5 class="card-title"><span class="count"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.opos}}' data-delay='5' data-increment='10'>{{bloodtypest

    </div>
    </div></div>
    <div class="col"><div class="card text-white bg-dark mb-3" style="max-width: 18rem;">
      <div class="card-header">O negtive</div>
    <div class="card-body">
      <h5 class="card-title"><span class="count"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.oneg}}' data-delay='5' data-increment='10'>{{bloodtypest

    </div>
    </div></div>
  </div>
</div>



<div class="alert alert-primary" role="alert">
  <h4>Plasma donations and their details</h4>
</div>
```

```
<!doctype html>
{% extends "base.html" %}
{% block content %}

{% with messages = get_flashed_messages() %}
{%if messages%}
    {%for mess in messages%}
  <div class="alert alert-warning alert-dismissible fade show" role="alert">
<strong>{{mess}}</strong>
<button type="button" class="close" data-dismiss="alert" aria-label="Close">
  <span aria-hidden="true">&times;</span>
</button>
</div>

    {%endfor%}
{%endif%}
{% endwith %}

     {% if session['logged_in'] == True %}

     {%else%}

     {%endif%}


    <br>

    <div class="card border-danger text-center">
      <div class="card-header">Total Plasma in Plasma bank</div>
      <div class="card-body text-danger">
        <h5 class="card-title"><span class='numscroller' data-min='1' data-max='{{totalblood}}' data-delay='5' data-increment='10'>{{totalblood}}</span> units</h5>
      </div>
    </div>

<br>



        <div class="container">
  <div class="row">
    <div class="col"><div class="card text-white bg-primary mb-3" style="max-width: 18rem;">
      <div class="card-header">A positive</div>
    <div class="card-body">
      <h5 class="card-title"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.apos}}' data-delay='5' data-increment='10'>{{bloodtypestotal.apos}}</span> u

    </div>
  </div></div>
    <div class="col"><div class="card text-white bg-secondary mb-3" style="max-width: 18rem;">
      <div class="card-header">A negative</div>
    <div class="card-body">
      <h5 class="card-title"><span class="count"><span class='numscroller' data-min='1' data-max='{{bloodtypestotal.aneg}}' data-delay='5' data-increment='10'>{{bloodtypest

    </div>
    </div></div>
    <div class="col"><div class="card text-white bg-success mb-3" style="max-width: 18rem;">
```

```
      <th scope="col">delete</th>
    </tr>

  </thead>

          {% for r in users %}
              <tr>
                  <td>{{r["name"]}}</td>
                  <td>{{r["addr"]}}</td>
                  <td> {{ r["city"]}}</td>
                  <td>{{r['pin']}}</td>
                  <td>{{r['bg']}}</td>
                  <td>{{r['email']}}</td>
                  <td><button type="button" class="btn btn-primary mt-1" data-toggle="modal" data-target="#exampleModalCenter">
    contact for Plasma
  </button>
  <div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-labelledby="exampleModalCenterTitle" aria-hidden="true">
    <div class="modal-dialog modal-dialog-centered" role="document">
      <div class="modal-content">
        <div class="modal-header">
          <h5 class="modal-title" id="exampleModalCenterTitle">contact for Plasma</h5>
          <button type="button" class="close" data-dismiss="modal" aria-label="Close">
            <span aria-hidden="true">&times;</span>
          </button>
        </div>
        <div class="modal-body">
          <form method="POST" action="{{url_for('contactforblood',emailid=r['email'])}}">
            <label for="name">Name</label>
            <input type = "text" name = "nm" value="admin@bloodbank.com" class="form-control" required/>
          <label for="addr"> confirm your Address</label>
         <input type="text" name="add" class="form-control" value="admin's address" required></textarea>
         <button type="button" class="btn btn-secondary mt-1" data-dismiss="modal">Close</button>
          <button type="submit" class="btn btn-primary mt-1">send request</button>
        </div>
      </div>
    </div>
  </div>
</td>
                  <td><a href = "{{url_for('deleteuser',useremail=r['email'])}}" class="btn btn-danger">delete user</a></td>
              </tr>
          {% endfor %}
```

Reset:

```
<!doctype html>
<html>
   <body>

      result of addition : {{ msg }}
      <h2><a href = "\">go back to home page</a></h2>

   </body>
</html>
```

Responsive:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></sc
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" cro
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js" integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy" crosso

<style>
body, html {
    height: 100%;
    margin: 0;
}

.bg {
  /* Set rules to fill background */
  min-height: 100%;
  min-width: 1024px;

  /* Set up proportionate scaling */
  width: 100%;
  height: auto;

  /* Set up positioning */
  position: fixed;
  top: 0;
  left: 0;
}

@media screen and (max-width: 1024px) { /* Specific to this particular image */
  .bg {
    left: 50%;
    margin-left: -512px;   /* 50% */
  }
}
</style>
</head>
<body>
  <img src="photo.jpg" class="bg"/>

<p><font color='red'>This example creates a full page background image. Try to resize the browser window to see how it always will cover the full screen (when scrolled to top)

</body>
</html>
```

Server:

```
from flask import render_template
import sqlite3
import requests
from flask import Flask
from flask import request,redirect,url_for,session,flash
from flask_wtf import Form
from wtforms import TextField
app = Flask(__name__)
app.secret_key = "super secret key"

@app.route('/')
def hel():
    conn = sqlite3.connect('database.db')
    print("Opened database successfully")
    conn.execute('CREATE TABLE IF NOT EXISTS users (name TEXT, addr TEXT, city TEXT, pin TEXT, bg TEXT,email TEXT UNIQUE, pass TEXT)')
    print( "Table created successfully")
    conn.close()
    if session.get('username')==True:
        messages = session['username']

    else:
        messages = ""
    user = {'username': messages}
    return redirect(url_for('index',user=user))


@app.route('/reg')
def add():
    return render_template('register.html')


@app.route('/addrec',methods = ['POST', 'GET'])
def addrec():
   msg = ""
   #con = None
   if request.method == 'POST':
      try:
         nm = request.form['nm']
         addr = request.form['add']
         city = request.form['city']
         pin = request.form['pin']
         bg = request.form['bg']
         email = request.form['email']
         passs = request.form['pass']

         with sqlite3.connect("database.db") as con:
             cur = con.cursor()
             cur.execute("INSERT INTO users (name,addr,city,pin,bg,email,pass) VALUES (?,?,?,?,?,?,?)",(nm,addr,city,pin,bg,email,passs) )
             con.commit()
             msg = "Record successfully added"



      except:
             con.rollback()
             msg = "error in insert operation"
```

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



    if request.method == 'POST':
        if session.get('username') is not None:
            messages = session['username']

        else:
            messages = ""
        user = {'username': messages}
        print(messages)
        val = request.form['search']
        print(val)
        type = request.form['type']
        print(type)
        if type=='blood':
            con = sqlite3.connect('database.db')
            con.row_factory = sqlite3.Row

            cur = con.cursor()
            cur.execute("select * from users where bg=?",(val,))
            search = cur.fetchall();
            cur.execute("select * from users ")

            rows = cur.fetchall();


            return render_template('index.html', title='Home', user=user,rows=rows,search=search)

        if type=='donorname':
            con = sqlite3.connect('database.db')
            con.row_factory = sqlite3.Row

            cur = con.cursor()
            cur.execute("select * from users where name=?",(val,))
            search = cur.fetchall();
            cur.execute("select * from users ")

            rows = cur.fetchall();


            return render_template('index.html', title='Home', user=user,rows=rows,search=search)



    if session.get('username') is not None:
        messages = session['username']

    else:
        messages = ""
    user = {'username': messages}
    print(messages)
```

```
    if session.get('username') is not None:
        messages = session['username']

    else:
        messages = ""
    user = {'username': messages}
    print(messages)
    if request.method=='GET':
        con = sqlite3.connect('database.db')
        con.row_factory = sqlite3.Row

        cur = con.cursor()
        cur.execute("select * from users ")

        rows = cur.fetchall();
        return render_template('index.html', title='Home', user=user, rows=rows)

    #messages = request.args['user']



@app.route('/list')
def list():
   con = sqlite3.connect('database.db')
   con.row_factory = sqlite3.Row

   cur = con.cursor()
   cur.execute("select * from users")

   rows = cur.fetchall();
   print(rows)
   return render_template("list.html",rows = rows)

@app.route('/drop')
def dr():
        con = sqlite3.connect('database.db')
        con.execute("DROP TABLE request")
        return "dropped successfully"

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

        email = request.form['email']
        password = request.form['pass']
        if email == 'admin@bloodbank.com' and password == 'admin':
            a = 'yes'
            session['username'] = email
            #session['logged_in'] = True
            session['admin'] = True
            return redirect(url_for('index'))
        #print((password,email))
```

```
            session['admin'] = True
            return redirect(url_for('index'))
        #print((password,email))
        con = sqlite3.connect('database.db')
        con.row_factory = sqlite3.Row

        cur = con.cursor()
        cur.execute("select email,pass from users where email=?",(email,))
        rows = cur.fetchall();
        for row in rows:
            print(row['email'],row['pass'])
            a = row['email']
            session['username'] = a
            session['logged_in'] = True
            print(a)
            u = {'username': a}
            p = row['pass']
            print(p)

            if email == a and password == p:
                return redirect(url_for('index'))
            else:
                return render_template('/login.html')
        return render_template('/login.html')
    else:
        return render_template('/')


@app.route('/logout')
def logout():
   # remove the username from the session if it is there
   session.pop('username', None)
   session.pop('logged_in',None)
   try:
       session.pop('admin',None)
   except KeyError as e:
       print("I got a KeyError - reason " +str(e))


   return redirect(url_for('login'))


@app.route('/dashboard')
def dashboard():
   totalblood=0
   con = sqlite3.connect('database.db')
   con.row_factory = sqlite3.Row

   cur = con.cursor()
   cur.execute("select * from blood")

   rows = cur.fetchall();
   for row in rows:
       totalblood  += int(row['qty'])

   cur.execute("select * from users")
   users = cur.fetchall();
```

```
cur = con.cursor()
cur.execute("select * from blood")

rows = cur.fetchall();
for row in rows:
    totalblood  += int(row['qty'])

cur.execute("select * from users")
users = cur.fetchall();

Apositive=0
Opositive=0
Bpositive=0
Anegative=0
Onegative=0
Bnegative=0
ABpositive=0
ABnegative = 0

print(rows)
cur.execute("select * from blood where type=?",('A+',))
type = cur.fetchall();
for a in type:
    Apositive += int(a['qty'])

cur.execute("select * from blood where type=?",('A-',))
type = cur.fetchall();
for a in type:
    Anegative += int(a['qty'])


cur.execute("select * from blood where type=?",('O+',))
type = cur.fetchall();
for a in type:
    Opositive += int(a['qty'])

cur.execute("select * from blood where type=?",('O-',))
type = cur.fetchall();
for a in type:
    Onegative += int(a['qty'])

cur.execute("select * from blood where type=?",('B+',))
type = cur.fetchall();
for a in type:
    Bpositive += int(a['qty'])


cur.execute("select * from blood where type=?",('B-',))
type = cur.fetchall();
for a in type:
    Bnegative += int(a['qty'])



cur.execute("select * from blood where type=?",('AB+',))
type = cur.fetchall();
for a in type:
```

```
    cur.execute("select * from blood where type=?",('AB+',))
    type = cur.fetchall();
    for a in type:
        ABpositive += int(a['qty'])


    cur.execute("select * from blood where type=?",('AB-',))
    type = cur.fetchall();
    for a in type:
        ABnegative += int(a['qty'])


    bloodtypestotal = {'apos': Apositive,'aneg':Anegative,'opos':Opositive,'oneg':Onegative,'bpos':Bpositive,'bneg':Bnegative,'abpos':ABpositive,'abneg':ABnegative}







    return render_template("requestdonors.html",rows = rows,totalblood = totalblood,users=users,bloodtypestotal=bloodtypestotal)


@app.route('/bloodbank')
def bl():
    conn = sqlite3.connect('database.db')
    print("Opened database successfully")
    conn.execute('CREATE TABLE IF NOT EXISTS blood (id INTEGER PRIMARY KEY AUTOINCREMENT, type TEXT, donorname TEXT, donorsex TEXT, qty TEXT, dweight TEXT, donoremail TE
    print( "Table created successfully")
    conn.close()
    return render_template('/adddonor.html')


@app.route('/addb',methods =['POST','GET'])
def addb():
    msg = ""
    if request.method == 'POST':
        try:
           type = request.form['blood_group']
           donorname = request.form['donorname']
           donorsex = request.form['gender']
           qty = request.form['qty']
           dweight = request.form['dweight']
           email = request.form['email']
           phone = request.form['phone']



           with sqlite3.connect("database.db") as con:
              cur = con.cursor()
              cur.execute("INSERT INTO blood (type,donorname,donorsex,qty,dweight,donoremail,phone) VALUES (?,?,?,?,?,?,?)",(type,donorname,donorsex,qty,dweight,email,ph
              con.commit()
              msg = "Record successfully added"
        except:
           con.rollback()
           msg = "error in insert operation"
```

```
    else:
        return render_template("rest.html",msg=msg)

@app.route("/editdonor/<id>", methods=('GET', 'POST'))
def editdonor(id):
    msg =""
    if request.method == 'GET':
        con = sqlite3.connect('database.db')
        con.row_factory = sqlite3.Row

        cur = con.cursor()
        cur.execute("select * from blood where id=?",(id,))
        rows = cur.fetchall();
        return render_template("editdonor.html",rows = rows)
    if request.method == 'POST':
        try:
           type = request.form['blood_group']
           donorname = request.form['donorname']
           donorsex = request.form['gender']
           qty = request.form['qty']
           dweight = request.form['dweight']
           email = request.form['email']
           phone = request.form['phone']



           with sqlite3.connect("database.db") as con:
              cur = con.cursor()
              cur.execute("UPDATE blood SET type = ?, donorname = ?, donorsex = ?, qty = ?,dweight = ?, donoremail = ?,phone = ? WHERE id = ?",(type,donorname,donorsex,q
              con.commit()
              msg = "Record successfully updated"
        except:
           con.rollback()
           msg = "error in insert operation"

        finally:
            flash('saved successfully')
            return redirect(url_for('dashboard'))
            con.close()

@app.route("/myprofile/<email>", methods=('GET', 'POST'))
def myprofile(email):
    msg =""
    if request.method == 'GET':


        con = sqlite3.connect('database.db')
        con.row_factory = sqlite3.Row

        cur = con.cursor()
        cur.execute("select * from users where email=?",(email,))
        rows = cur.fetchall();
        return render_template("myprofile.html",rows = rows)
    if request.method == 'POST':
        try:
```

```
    if request.method == 'POST':
        try:
            name = request.form['name']
            addr = request.form['addr']
            city = request.form['city']
            pin = request.form['pin']
            bg = request.form['bg']
            emailid = request.form['email']

            print(name,addr)



            with sqlite3.connect("database.db") as con:
                cur = con.cursor()
                cur.execute("UPDATE users SET name = ?, addr = ?, city = ?, pin = ?,bg = ?, email = ? WHERE email = ?",(name,addr,city,pin,bg,emailid,email) )
                con.commit()
                msg = "Record successfully updated"
        except:
            con.rollback()
            msg = "error in insert operation"

        finally:
            flash('profile saved')
            return redirect(url_for('index'))
            con.close()



@app.route('/contactforblood/<emailid>', methods=('GET', 'POST'))
def contactforblood(emailid):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        print("Opened database successfully")
        conn.execute('CREATE TABLE IF NOT EXISTS request (id INTEGER PRIMARY KEY AUTOINCREMENT, toemail TEXT, formemail TEXT, toname TEXT, toaddr TEXT)')
        print( "Table created successfully")
        fromemail = session['username']
        name = request.form['nm']
        addr = request.form['add']

        print(fromemail,emailid)
        conn.execute("INSERT INTO request (toemail,formemail,toname,toaddr) VALUES (?,?,?,?)",(emailid,fromemail,name,addr) )
        conn.commit()
        conn.close()
        flash('request sent')
        return redirect(url_for('index'))
    if request.method == 'POST':
        conn = sqlite3.connect('database.db')
        print("Opened database successfully")
        conn.execute('CREATE TABLE IF NOT EXISTS request (id INTEGER PRIMARY KEY AUTOINCREMENT, toemail TEXT, formemail TEXT, toname TEXT, toaddr TEXT)')
        print( "Table created successfully")
        fromemail = session['username']
        name = request.form['nm']
        addr = request.form['add']

        print(fromemail,emailid)
```

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


            conn = sqlite3.connect('database.db')
            print("Opened database successfully")
            conn.row_factory = sqlite3.Row

            cur = conn.cursor()
            cor = conn.cursor()
            cur.execute('select * from request where toemail=?',(session['username'],))
            cor.execute('select * from request where toemail=?',(session['username'],))
            row = cor.fetchone();
            rows = cur.fetchall();
            if row==None:
                return render_template('notifications.html')
            else:
                return render_template('notifications.html',rows=rows)









@app.route('/deleteuser/<useremail>',methods=('GET', 'POST'))
def deleteuser(useremail):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from users Where email=?',(useremail,))
        flash('deleted user:'+useremail)
        conn.commit()
        conn.close()
        return redirect(url_for('dashboard'))


@app.route('/deletebloodentry/<id>',methods=('GET', 'POST'))
def deletebloodentry(id):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from blood Where id=?',(id,))
        flash('deleted entry:'+id)
        conn.commit()
        conn.close()
        return redirect(url_for('dashboard'))
```

```python
@app.route('/deleteuser/<useremail>',methods=('GET', 'POST'))
def deleteuser(useremail):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from users Where email=?',(useremail,))
        flash('deleted user:'+useremail)
        conn.commit()
        conn.close()
        return redirect(url_for('dashboard'))


@app.route('/deletebloodentry/<id>',methods=('GET', 'POST'))
def deletebloodentry(id):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from blood Where id=?',(id,))
        flash('deleted entry:'+id)
        conn.commit()
        conn.close()
        return redirect(url_for('dashboard'))

@app.route('/deleteme/<useremail>',methods=('GET', 'POST'))
def deleteme(useremail):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from users Where email=?',(useremail,))
        flash('deleted user:'+useremail)
        conn.commit()
        conn.close()
        session.pop('username', None)
        session.pop('logged_in',None)
        return redirect(url_for('index'))

@app.route('/deletenoti/<id>',methods=('GET', 'POST'))
def deletenoti(id):
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        cur = conn.cursor()
        cur.execute('delete from request Where id=?',(id,))
        flash('deleted notification:'+id)
        conn.commit()
        conn.close()
        return redirect(url_for('notifications'))



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

13.2 GitHub link and demo link

https://github.com/IBM-EPBL/IBM-Project-16489-1659615802.git

https://clipchamp.com/watch/uOpb6eVa9Mh