PLASMA DONOR APPLICATION

A PROJECT REPORT

Submitted-by

SRI RAAM A P (Reg.No:810019104073)
SNEKA M (Reg.No:810019104069)
YUVAKRUBA S A (Reg.No:810019104091)
PRIYADHARSHINI N (Reg.No:810019104701)

in partial fulfillment for the award of degree

of

BACHELOR OF ENGINEERING IN

COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING (BIT-CAMPUS)

ANNA UNIVERSITY TIRUCHIRAPALLI-620021

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that his project report "**PLASMA DONOR APPLICATION"** is the bonafide work of " SRI RAAM A P (810019104073)", "SNEKA M (810019104069)", "YUVAKRUBA S A (810019104091)" and "PRIYADHARSHINI S (810019104701)" who carried out the project work under my supervision.

SIGNATURE

Dr. G. ANNAPOORANI

HEAD OF THE DEPARTMENT
Assistant Professor

Department of CSE

University College of Engineering
(BIT Campus) Anna University
Tiruchirapalli- 620021

SIGNATURE

Mr. C. SURESHKUMAR

PROJECT SUPERVISOR
Teaching Faculty

Department of CSE

University College of Engineering
(BIT Campus) Anna University
Tiruchirapalli- 620021

ANNA UNIVERSITY: CHENNAI 600 025

PROJECT COMPLETION CERTIFICATE

Certified that his project report "**PLASMA DONOR APPLICATION** is the bonafide work of " SRI RAAM A P (810019104073)", "SNEKA M (810019104069)", "YUVAKRUBA S A (810019104091)" and "PRIYADHARSHINI S (810019104701)" who carried out the project work under my supervision.

SIGNATURE	SIGNATURE
Dr. G. ANNAPOORANI	Mr. C. SURESHKUMAR
HEAD OF THE DEPARTMENT Assistant Professor	PROJECT SUPERVISOR Teaching Faculty
Department of CSE	Department of CSE
University College of Engineering (BIT Campus) Anna University Tiruchirapalli- 620021	University College of Engineering (BIT Campus) Anna University Tiruchirapalli- 620021

Project Submitted date _____

Faculty Mentor Supervisor Industry Mentor

LIST OF CONTENTS

TITLE

1 INTRODUCTION

1.1 OVERVIEW

During the COVID 19 crisis, the requirement for plasma became high and the donor count being low. Saving the donor information and helping the need by notifying the current donors would be a helping hand.

In regard to the problem faced, an application is to be built which would take the donor details, store it and inform them upon a request. Users need to register an account and login to the application. Once the user logins, he will have a dashboard to view the total number of donors and count of people with specific blood groups.

There are some blood donor finder applications such as Blood app by Red Cross which allows the donor to book appointment with blood banks and also can find local blood drives and donation centers quickly and easily . However, there is no direct communication between the donor and that clinic in need of a specific blood type. As a result, this app is more beneficial for donors but not for clinics to find needed blood type directly and promptly.

Blood Donor Finder application by Neologix allows users in need for blood to find nearest donors. Although this application helps finding donors, but the ease of communication with those donors is not prompt and it requires man power as the requester (patient or clinic) has to contact each donor individually. Also, there is no application that provides a proper communication channel to notify donors about the blood donation requirements

1.2 PURPOSE

- The main objective of this project is to provide the recipient with a donor who is in good form with no health ailments to donate blood of the corresponding blood group. This project provides quick access to donors for an immediate requirement of blood. In case of an emergency/surgery, blood procurement is always a major problem which consumes a lot of time. This helps serve the major time-lapse in which a life can be saved. Users will have the option to request the blood. Once the user requests, all the people with that blood group will be notified with an mail

- Blood Plasma Donations are used for slightly more specific purposes than a general blood donation. The most common uses of plasma donations include individuals who have experienced a severe trauma, burn or shock, adults or children with cancer, and people with liver or clotting factor disorders. Donated plasma can be frozen and stored for up to one year. Nearly 10,000 units of plasma are needed every day in the United States, and plasma transfusions are often lifesaving.

- The proposed method helps the users to check the availability of donors. A donor has to register to the website providing their details. The registered users can get the information about the donor count of each blood group. The database will have all the details such as name, email, phone number, infected status. Whenever a user requests for a particular blood group then the concerned blood group donors will receive the notification regarding the requirement. A Json code is written to store the information, to fetch the requested information in lambda.

- When a user request for a particular blood group an API will invoke the lambda function and the lambda function will trigger operation and fetches the information of a particular blood group donor from the dynamo-db and it will then fetch it back to the API and this API will display the information in the user interface. when a user requests for a particular blood group a request sms will be sent to the particular blood group

2 LITERATURE SURVEY

2.1 EXISTING PROBLEM

- Despite advances in technology, nowadays, most blood bank systems are manual systems. As such, there is a prevalent problem in the availability of needed blood types. For instance, when a person needs a certain type of blood and this type is not available in the hospital, family members send messages through social media to those who can donate to them and this process takes longer than the life of the patient to the most dangerous.

- In recent days, it is noticed the increase in blood request posts on social media such as Facebook, Twitter, and Instagram. Interestingly there are many people across the world interested in donating blood when there is a need, but those donors don't have an access to know about the blood donation requests in their local area. This is because that there is no platform to connect local blood donors with patients.

- BLOODR solves the problem and creates a communication channel through authorized clinics whenever a patient needs blood donation. It is a useful tool to find compatible blood donors who can receive blood request posts in their local area. Clinics can use this web application to maintain the blood donation activity. Collected data through this application can be used to analyze donations to requests rates in a local area to increase the awareness of people by conducting donations camps.

- BLOODR Application can be developed to further improve user accessibility via integrating this application with various social networks application program interfaces (APIs). Consequently, users can login and sign up using various social networks. This would increase number of donors and enhances the process of blood donation.

2.2 PROPOSED SOLUTION

- To build a web application that is capable of acting as a medium for recipients and donors of blood. The application must be deployed on Cloud Foundry.

- Create an API Endpoint for the model with the help of IBM Cloud Functions.

- An alert is to be sent using the send-grid mail Service to all the registered users whenever a request for blood is posted.

- In recent days, it is noticed the increase in blood request posts on social media such as Facebook, Twitter, and Instagram. Interestingly there are many people across the world interested in donating blood when there is a need, but those donors don't have access to know about the blood donation requests in their local area. This is because there is no platform to connect local blood donors with patients.

- BLOODY solves the problem and creates a communication channel through authorized clinics whenever a patient needs blood donation. It is a useful tool to find compatible blood donors who can receive blood request posts in their local area. Clinics can use this web application to maintain the blood donation activity. Collected data through this application can be used to analyze donations to requests rates in a local area to increase the awareness of people by conducting donation camps.

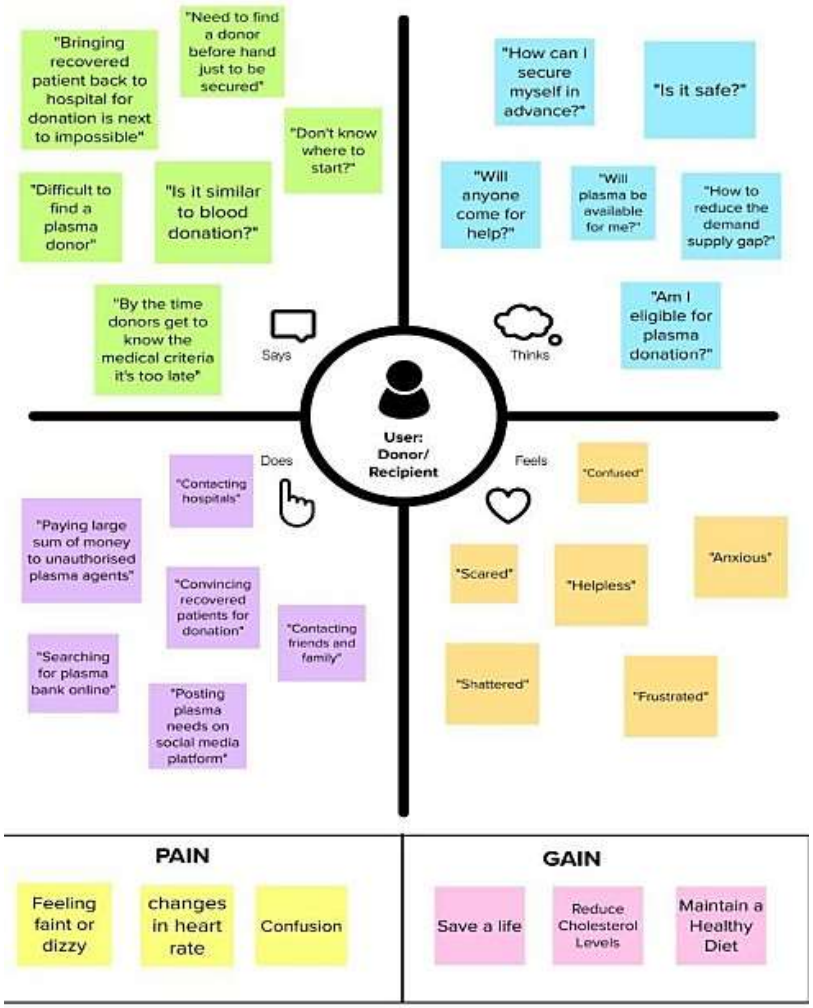
- BLOODY Application can be developed to further improve user accessibility via integrating this application with various social networks application program interfaces (APIs). Consequently, users can login and sign up using various social networks. This would increase the number of donors and enhances the process of blood donation.

3 IDEATION & PROPOSED SOLUTION

3.1 EMPATHY MAP CANVAS:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behavior and attitudes.

Creating an effective solution requires understanding the true problem and the person who is experiencing it.

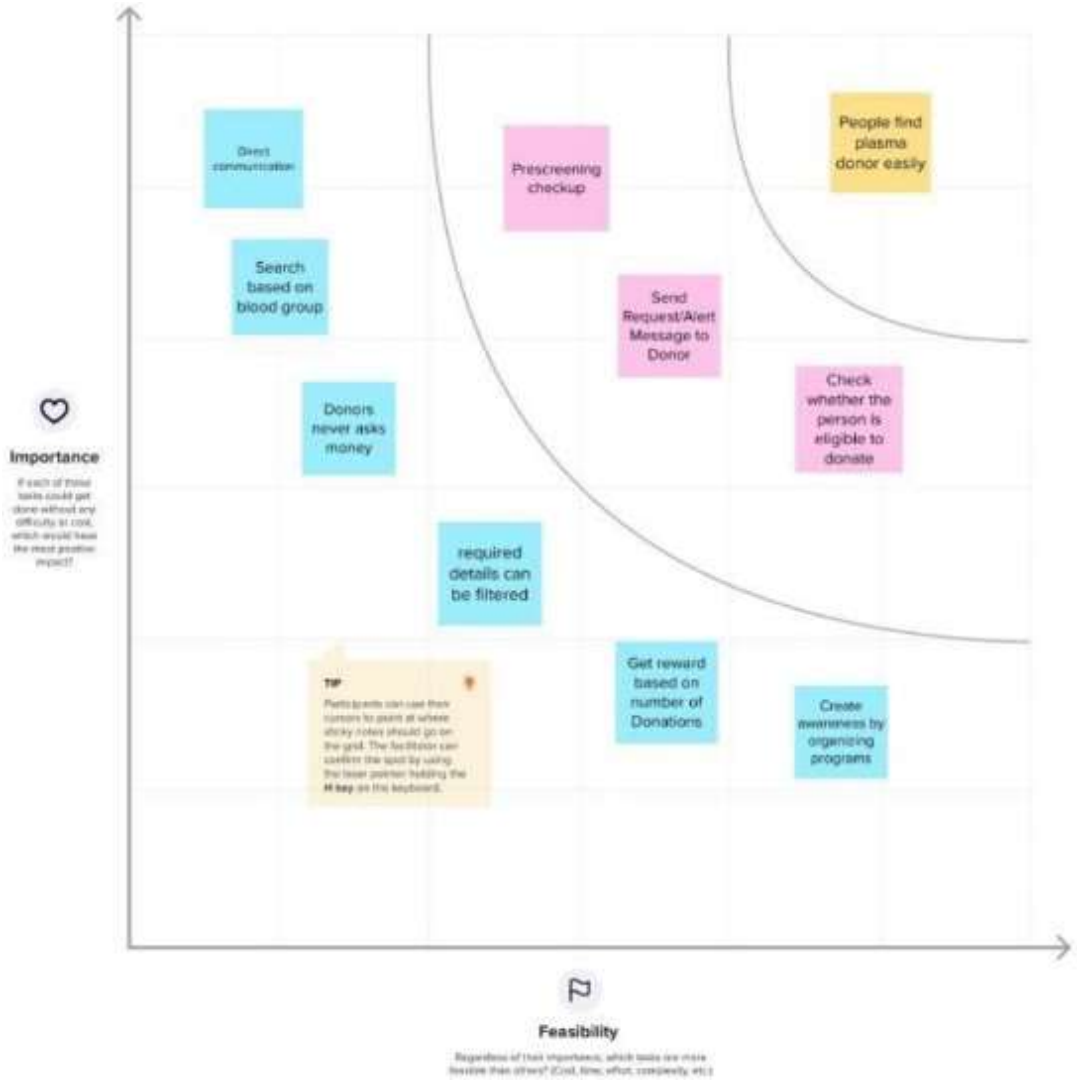


3.2 BRAINSTORMING & IDEATION

Brainstorming is a creative technique by which people try to find adequate solutions for specific problems by gathering inspirations and ideas, proposed by different members of a team, and then exploring them. It can involve making lists, drawing diagrams, or simply discussion.

It is made to dig into each member's mind to mine out each grain of creativity they have,to find a solution or even the problem itself, a chaotic technique that values quantity over quality, trying to find that special idea, that "gold nugget"

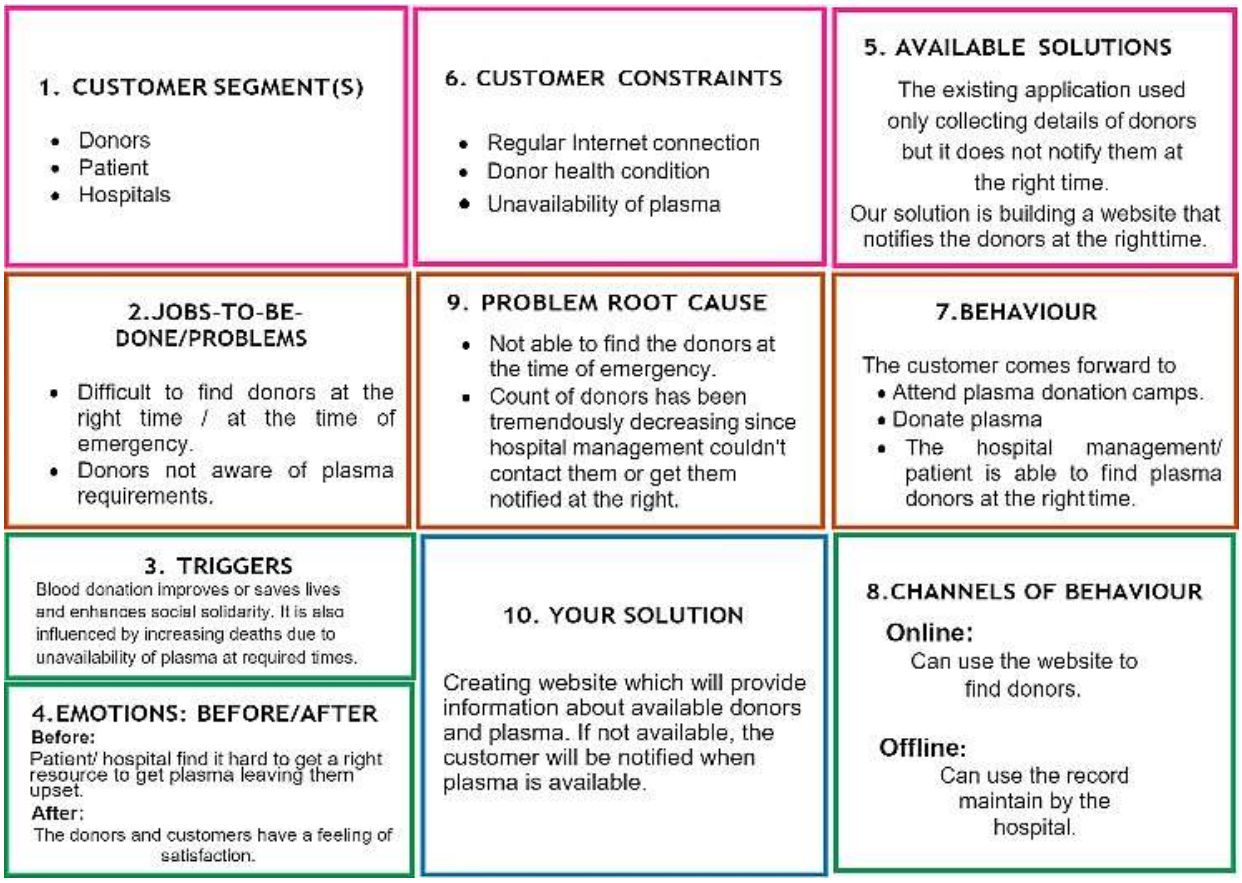
Brainstorming needs a lot of patience, perseverance and motivation in order to reach what we aim for . it requires a mind full of concentration and determination.



3.2 PROBLEM SOLUTION FIT

So you have a great idea for a new business. You're excited to get started and make your mark on the world. But before you do anything, it's important to make sure that there is a demand for your offering in the market.

The least glamorous but most important part of starting a successful business is determining whether your idea actually solves a real problem for people. This process is known as finding a problem-solution fit.



3.4 PROPOSED SOLUTION

Having hooked your audience into the problem, now you want to paint a picture of what the world will be like when you solve the problem. Your proposed solution should relate the current situation to a desired result and describe the benefits that will accrue when the desired result is achieved. So, begin your proposed solution by briefly describing this desired result.

Begin by describing a significant real world problem that your audience can relate to. Show a concrete example of this problem. This will serve as a "hook" to get the attention of your audience and convince them of the importance of the project. Next you want to focus in on the part of the problem you want to attack. So follow your example by focusing on a small but significant portion of the problem. This is the part that you are proposing to tackle

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Plasma needer face more difficult to find plasma donor and get more panic
2.	Idea / Solution description	Made an easy connection between plasma donor and needer(patient)
3.	Novelty / Uniqueness	No third person interact between them and easy user interface
4.	Social Impact / user Satisfaction	Impact between the users on the application is made easy communication and make them more secured and comfort
5.	Business Model (Revenue Model)	Non-revenue model
6.	Scalability of the Solution	The main goal of the application is to provide high Scalability by given more option for user to select their interest(donate/assist)

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 mobile/ laptop/ PC Registration through telegram group
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Donor Notification	Get notification through register mobile number Get notification through register Email
FR-4	Plasma needer details(person)	Availability details in App Availability details in telegram group
FR-5	Plasma availability(blood)	Availability details in App Availability details in telegram group

4.3 SYSTEM REQUIREMENTS

We'll be able to work on IBM DB2, creating flask application, making an application into an docker image and deploying app in kubernetes space. Build a flask application which-will take the user inputs,update the IBMDB2 database and notify the user upon request.

HARDWAREREQUIREMENTS

APC with internet connective

SoftwareRequirements

OperatingSystem -Windows11 or 10

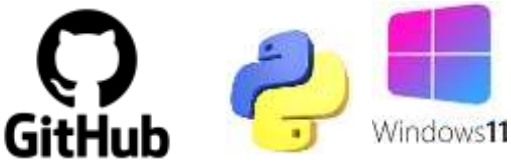
FrontEndTool - Python 3.10.5

IDE -VisualStudioCode

CodeRepository - GitHub

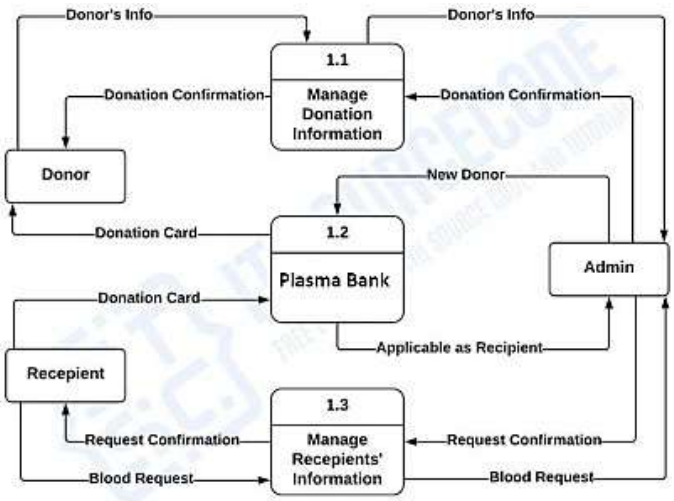
BackEndTool - IBM DB2 service on IBM

CloudAppDeployment Environment - kuberster with docer

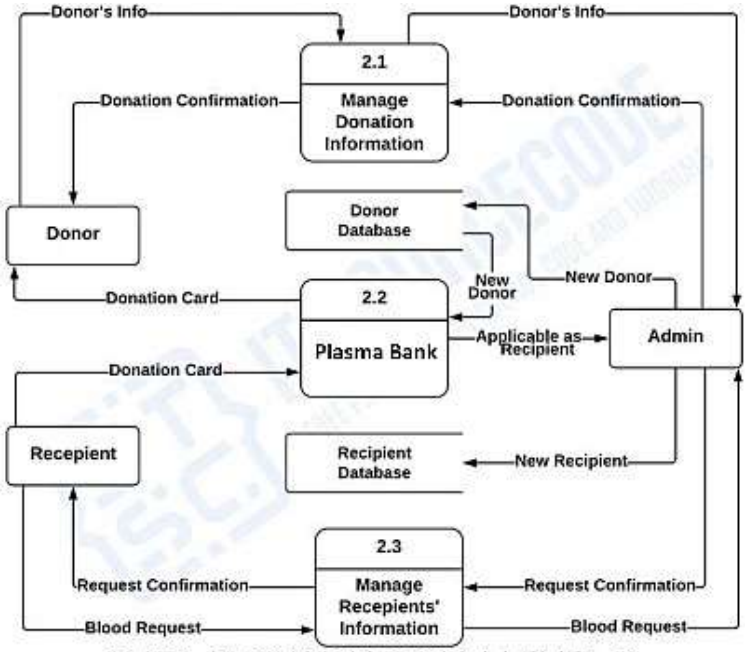


5 PROJECT FLOW

5.1 DATA FLOW DIAGRAMS



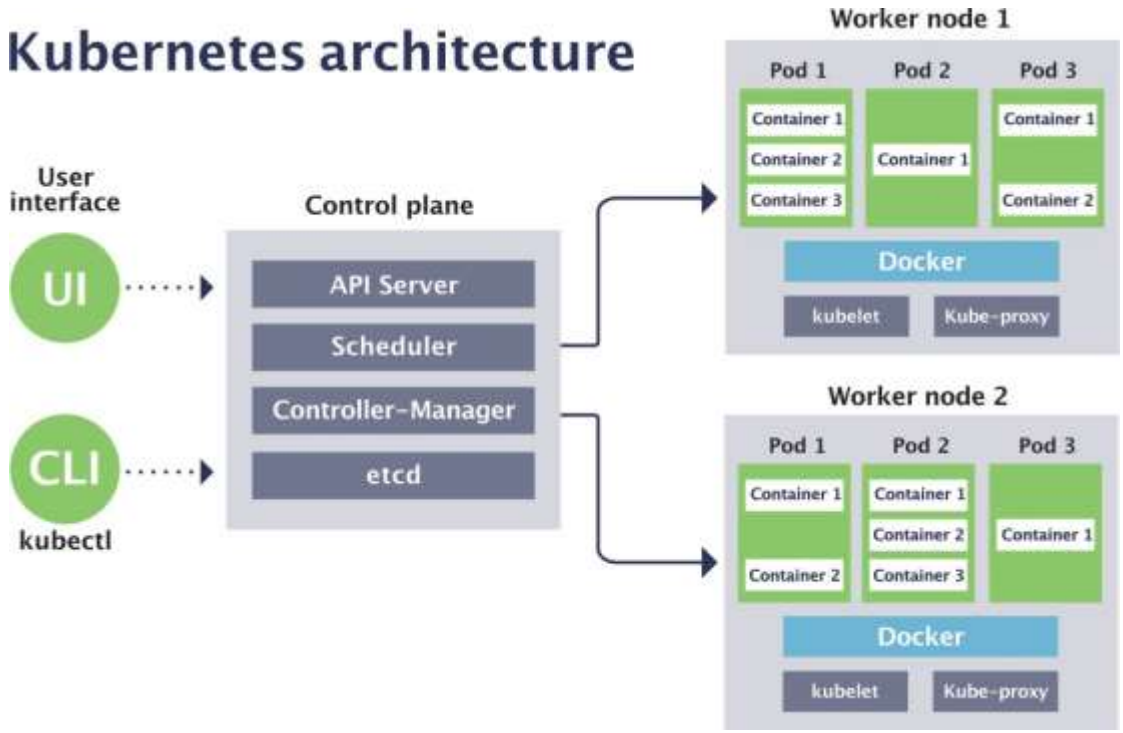
DATA FLOW DIAGRAM LEVEL 1



DATA FLOW DIAGRAM LEVEL 2

5.2 SOLUTION ARCHITECTURE

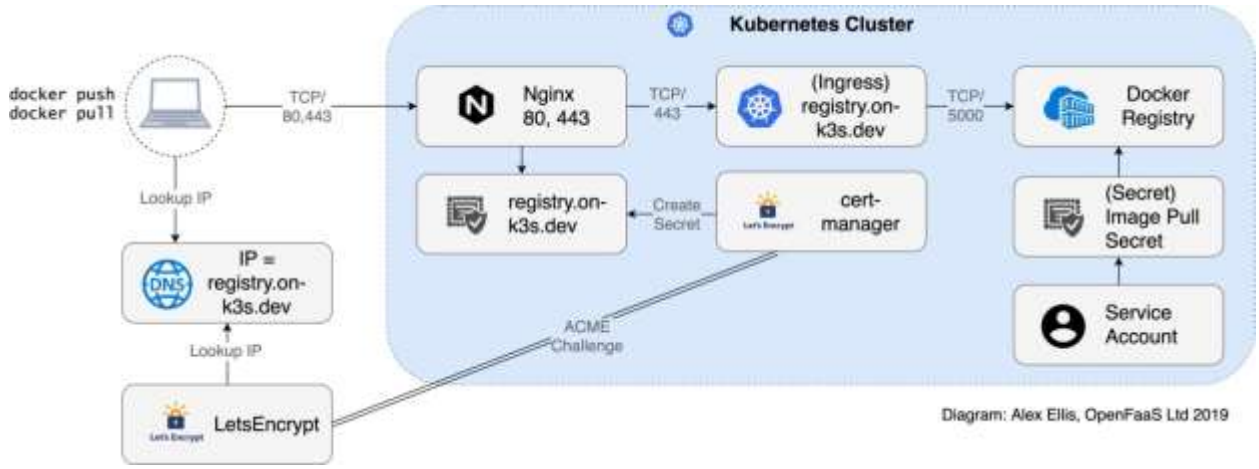
KUBERNETES CLUSTER



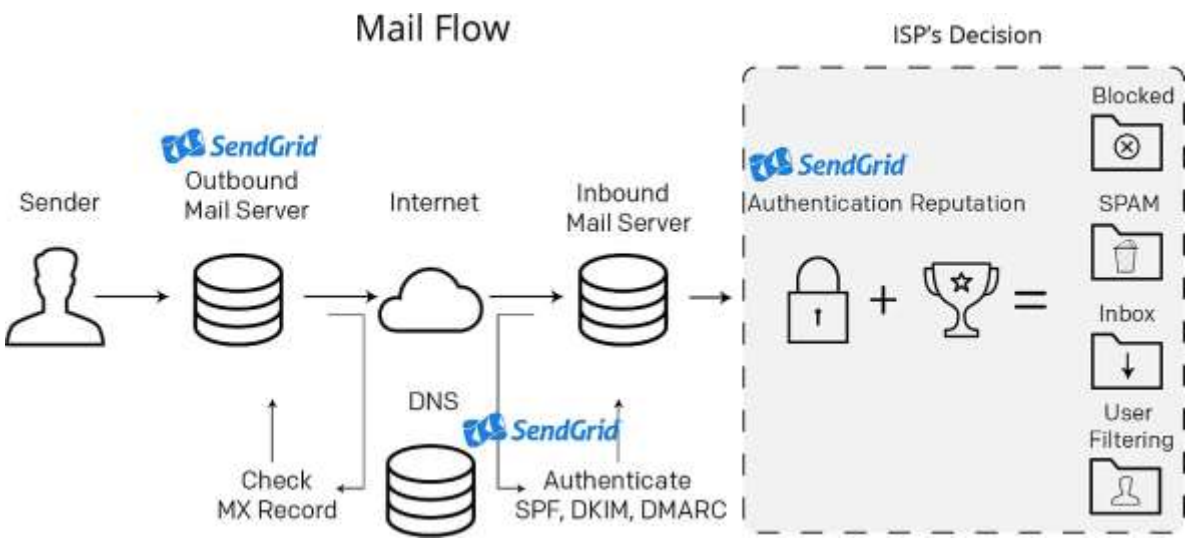
Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling,and management of containerized applications Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications

CONTAINER REGISTRY

A container registry is a repository—or collection of repositories—used to store and access container images. Container registries can support container-based application development, often as part of DevOps processes. Container registries can connect directly to container platforms like Docker and Kubernetes



SENDGRID



First, a sender puts together the content that their recipients will love. Then it's time for the "SMTP conversation" to take place. SMTP stands for Simple Mail Transfer Protocol, and this conversation is what makes email messages get from the sender to the recipient. It's easiest to think of an SMTP conversation as a "handshake". Imagine that a

sender is a host at a party and of the other guests are the recipients of the message. The host will shake every guest's hand and during that "handshake" they will have this SMTP conversation. In the end, the guest (i.e.recipient and its recipient server) will determine if they will accept the message or not. In this, you can think of SendGrid as a person at the party grabbing both the host's and guest's hands and making the handshake and discussion actually happen.

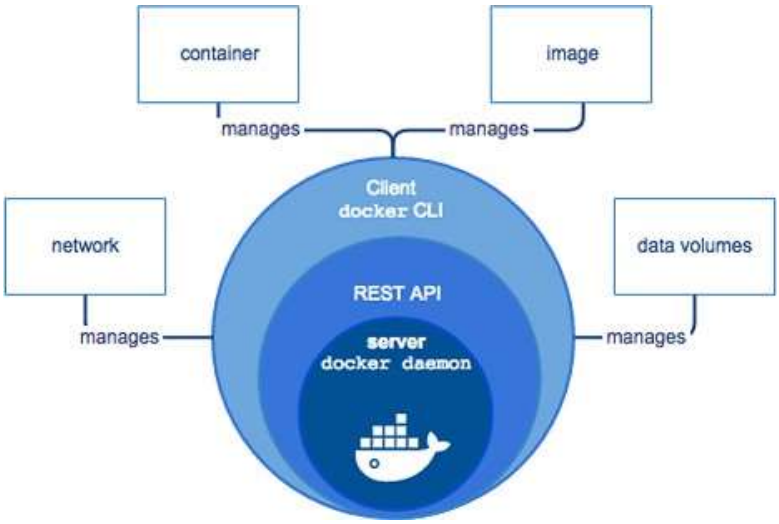
IBM DATABASE2

The architecture of the DB2 is mainly divided into the DB2 client and DB2 server.

The DB2 server consists of different agents and sub agent. The two important components are the pre fetchers and the page cleaners which maintain the data in buffer pools for fast and effective retrieval of data.

Docker

Docker can build images automatically by reading the instructions from a Docker file. A Docker file is a text document that contains all the commands a user could call on the command line to assemble an image. It is used for building and seamlessly integrating legacy projects enabling organizations to achieve high-velocity innovations. It encourages the concept of DevOps methodology through CI/CD (Continuous



integration/Continuous Development). Thus, the developers can integrate their code into a shared repository as early as possible and deploy it quickly and efficiently. Thus, the local development setup behaves like a live server. It comes up with integrated developer tools. Also, the virtual machine image is openly accessible and shareable. The applications developed on it can be reused and are shareable. It is open-source and available on Github. It reduces the setup cost on the part of the customers and increases the efficiency and the productivity of the existing application workflow as both are open source technologies. This also ensures the scalability of the existing application workflow. Also, as both of the technologies form an integral part of the cloud platform, they can be used independently.

6 PROJECT PLANNING

6.1 SPRINT PLANNING & ESTIMATION

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	**Registration and Login**	USN-1	Create UI to interact with pages. To create the userand admin login functionality	20	High	Yuvaraj chandru
Sprint-2	**Cloud and Database**	USN-2	Connecting flask app with database[IBMDB2] Implementation of IBM chatbot	20	High	Chandru vasuki
Sprint-3	**Deployment in Devops phase**	USN-3	Creating images with docker, Deploying Kubernetes and add the mailing service.	20	High	Vasuki krishnave ni
Sprint-4	**Testing and Deployment to user**	USN-4	To make sure that the software is handy to users.	20	High	Krishnave ni Yuvaraj

6.2 SPRINT DELIVERYSCHEDULE

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

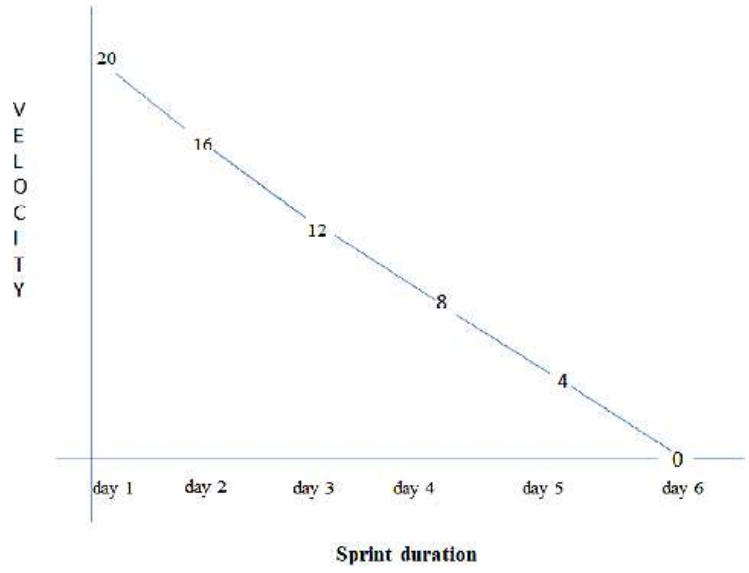
Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let's calculate

the team's average velocity (AV) per iteration unit (story points per day)

Sprint duration = 6 days

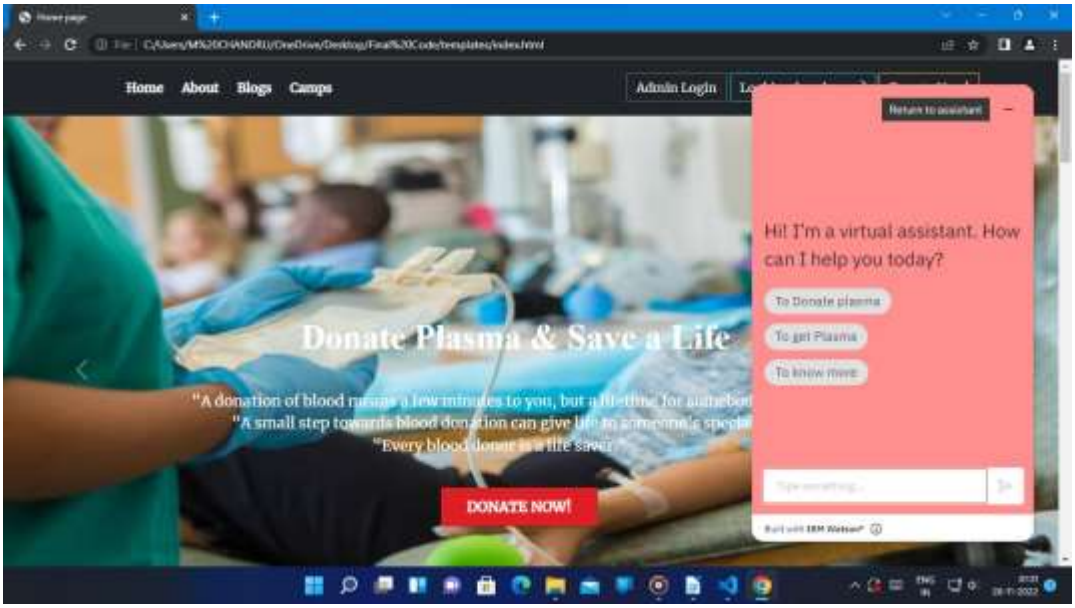
Velocity of the team = 20 points

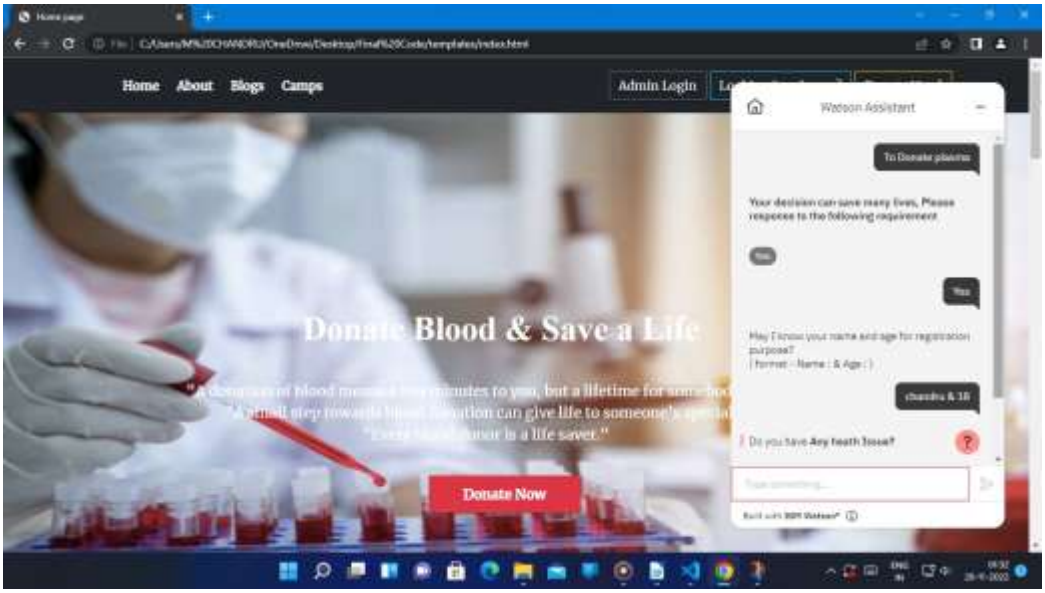


7 CODING SOLUTION

7.1 WATSON ASSISTANT BOT

Chat bots are computer programs that run automated tasks over the internet via artificial intelligence (AI). In an age where large corporations have already taken advantage of the technology, it is time for businesses to leverage its power too.





7.2 SENDING MAIL TO DONOR

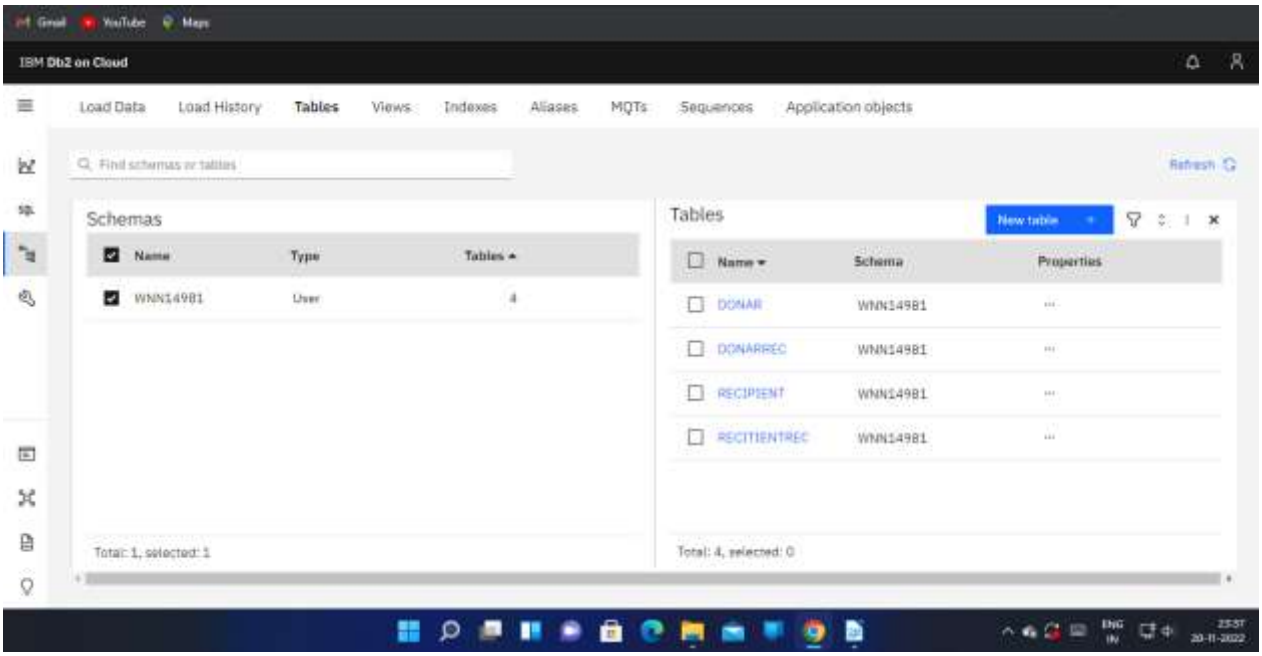
Be sure to perform the following prerequisites to complete this tutorial. You can skip ahead if you've already completed these tasks.

1. *Sign up for a SendGrid account.*
2. *Enable Two-factor authentication.*
3. *Create and store a SendGrid API Key with Mail Send > Full Access permissions.*
4. *Complete Domain Authentication.*
5. *Install Python*

Send Grid is a cloud-based SMTP provider that allows you to send email without having to maintain email servers. Send Grid manages all of the technical details, from scaling the infrastructure to ISP outreach and reputation monitoring to whitelist services and real time analytics.

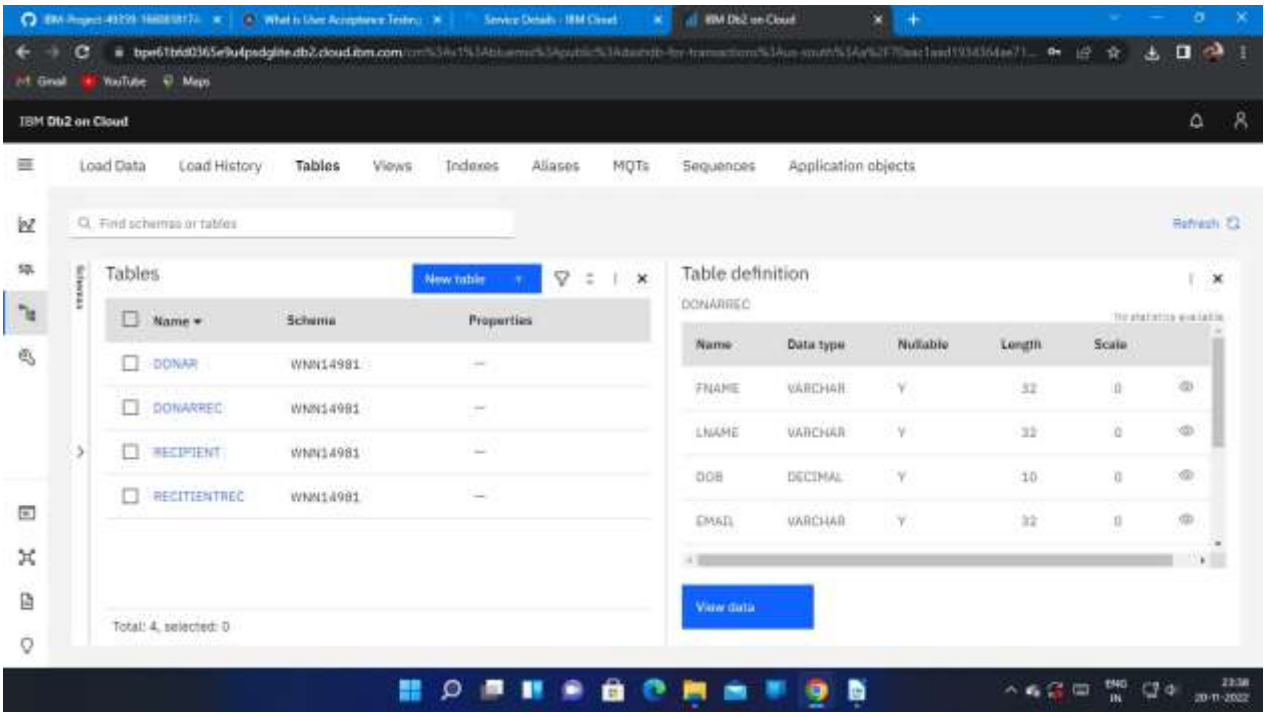
7.3 DETAILS ARE STORE IN CLOUD

IBM's DB2 has been around for a number of years and has matured into a robust relational database management system (RDBMS). While DB2 has its share of competitor, such as Oracle, MS SQL and MySQL, it continues to enjoy strong market presence. This staying power is due to several advantages that DB2 offers



POWERFUL STRUCTURE QUERY LANGUAGE

DB2 has a more powerful Structured Query Language (SQL) dialect than Microsoft's SQL offering. DB2 has features such as object tables, before triggers, Java method support, multiple user-defined functions and support for arrays. None of these features are supported by MS SQL. SQL allows direct access to the data via the database management interface of DB2. It also can be imbedded in the code of application programs written in other languages, such as COBOL and Java. This flexibility and feature list combine to make DB2 a powerful relational database management system



IBM produces versions of DB2 that run on all available platforms, rather than just Windows-based platforms. Included in the DB2 list of supported platforms are AIX, HP-UX, Linux and Sun. This ability to run on a variety of hardware/software platforms brings a flexibility to the table that other database engines do not offer. IBM also offers DB2 as part of a pre-configured bundle along with the operating system. This combination produces better performance since the operating system is configured for DB2 database support out of the box, saving the system administrator time.

A major advantage of DB2 is that it is an IBM offering. Developed many years ago at IBM's database labs, DB2 has gained feature after feature over the years. IBM produces software updates and patches on a controlled basis after thorough testing. IBM's high-quality software support is a factor in DB2's popularity, along with the system stability that results from that support. IBM's research and development dollars continue to improve the product making DB2 an ideal choice for a RDBMS

8 TESTING

8.1 USERACCEPTANCETESTING (UA TESTING)

Need of User Acceptance Testing arises once software has undergone Unit, Integration and System testing because developers might have built software based on requirements document by their own understanding and further required changes during development may not be effectively communicated to them, so for testing whether the final product is accepted by client/end-user, user acceptance testing is needed

STEP 1) Preparation of Test Data:

It is best advised to use live data for UAT. Data should be scrambled for privacy and security reasons. Tester should be familiar with the database flow.

Step 2) Run and record the results:

Execute test cases and report bugs if any. Re-test bugs once fixed. Test Management tools can be used for execution.

Step 3) Confirm Business Objectives met:

Business Analysts or UAT Testers needs to send a sign off mail after the UAT testing. After sign-off, the product is good to go for production. Deliverables for UAT testing are Test Plan, UAT Scenarios and Test Cases, Test Results and Defect Log.

1 PurposeofDocument

Thepurposeofthisdocumentistobrieflyexplainthetestcoverageandopenissuesofthe[Product Name]projectatthetimeofthereleasetoUserAcceptanceTesting (UAT)

2 DefectAnalysis

Thisreportshowsthenumberofresolvedor closedbugsateachseverity level,andhowthey wereresolved

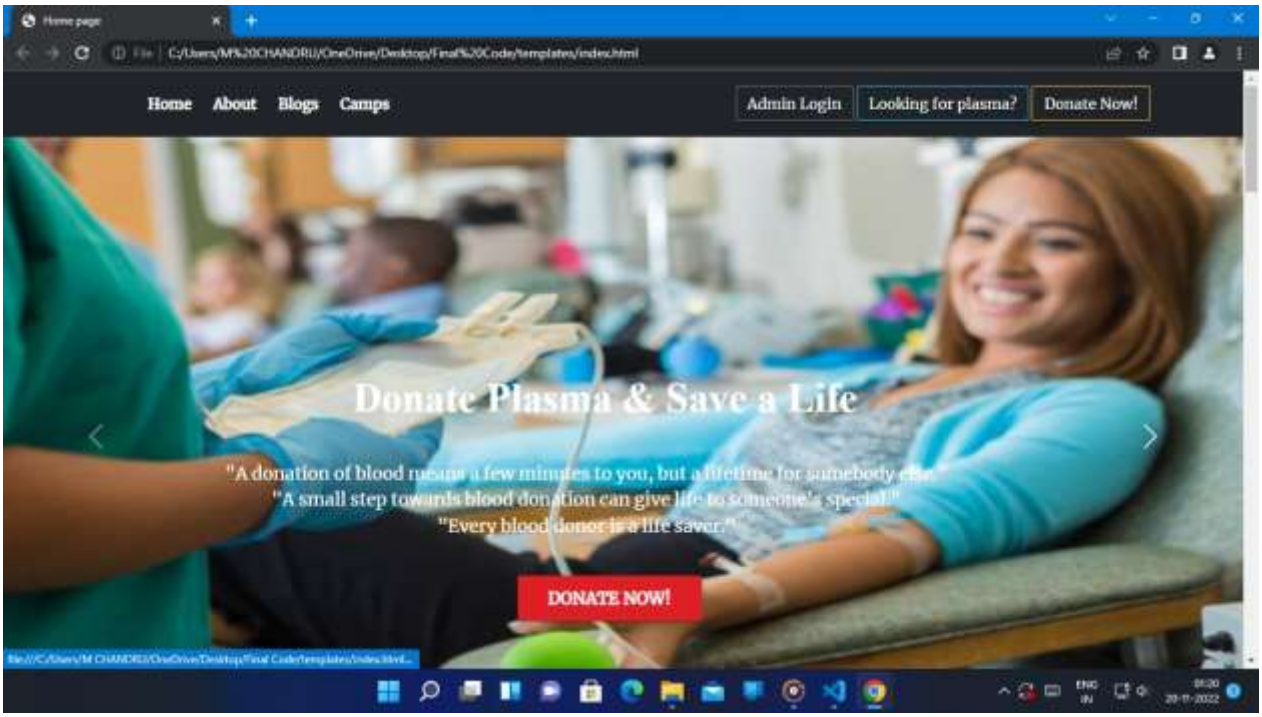
Resolution	Severity1	Severity2	Severity3	Severity4	Subtotal
ByDesign	10	4	2	3	20
Duplicate	1	0	1	0	2
External	2	2	1	1	6
Fixed	4	1	1	10	16
NotReproduced	0	0	0	0	0
Skipped	1	1	0	1	3
Won'tFix	0	2	2	0	4
Totals	24	14	13	26	51

3 TestCaseAnalysis

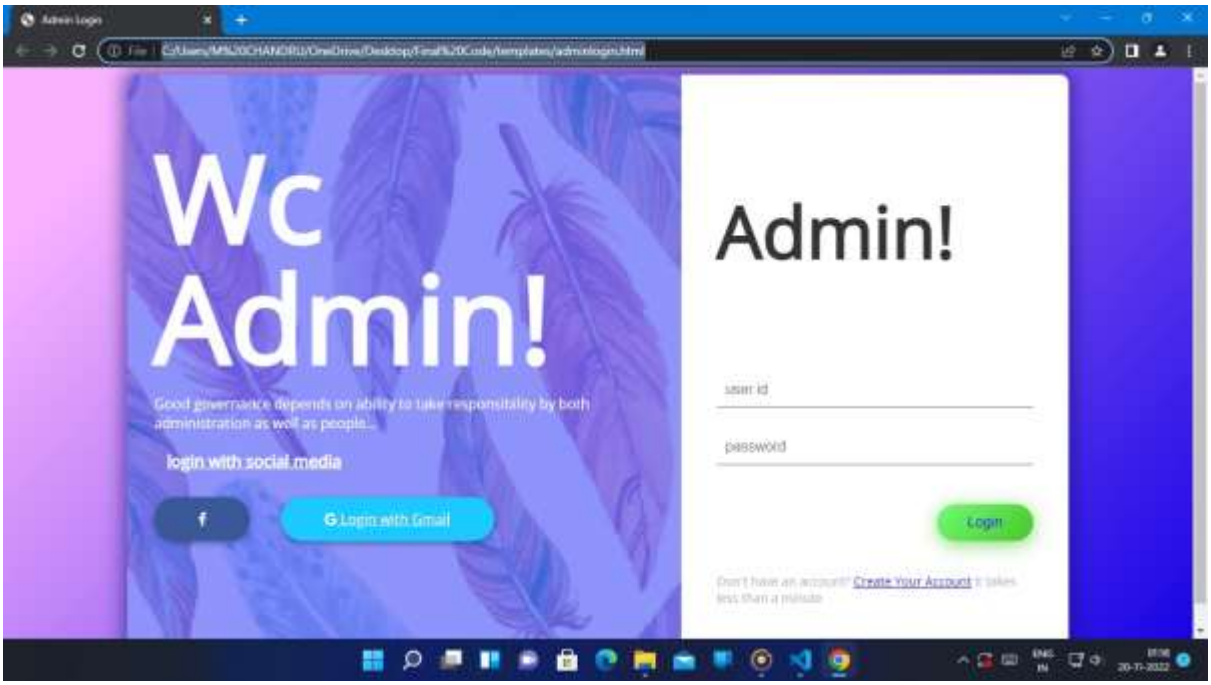
Thisreportshowsthenumberoftestcasesthathavepassed,failed,anduntested

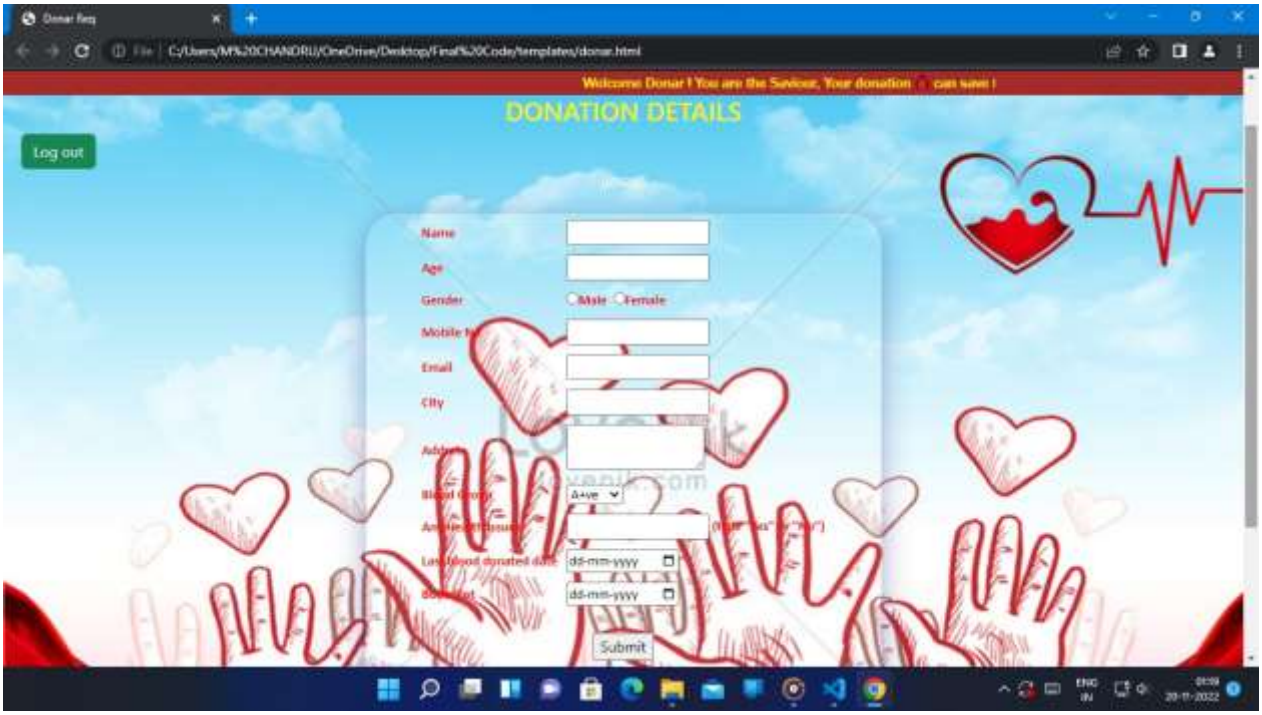
Section	TotalCases	NotTested	Fail	Pass
PrintEngine	9	0	0	9
ClientApplication	10	0	0	10
Security	1	0	0	1
ExceptionReporting	9	0	0	9
FinalReportOutput	9	0	0	9
VersionControl	1	0	0	1
OutsourceShipping	0	0	0	0

10 RESULT

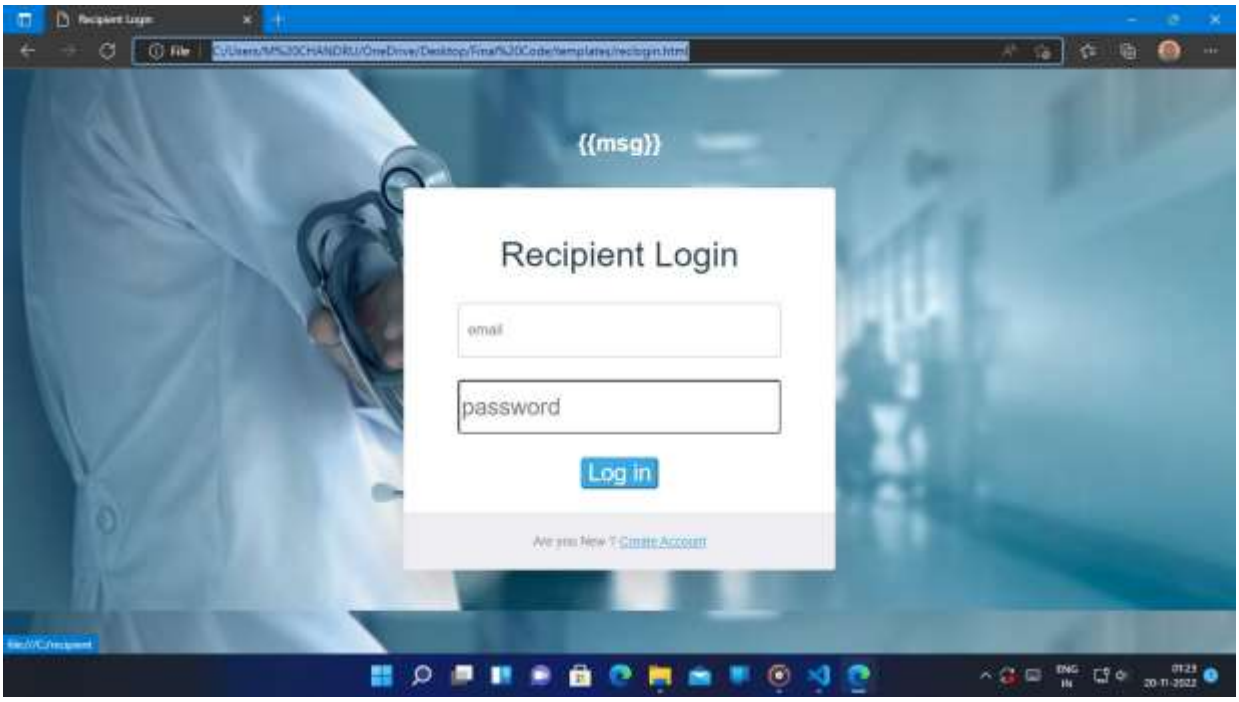


HOME PAGE

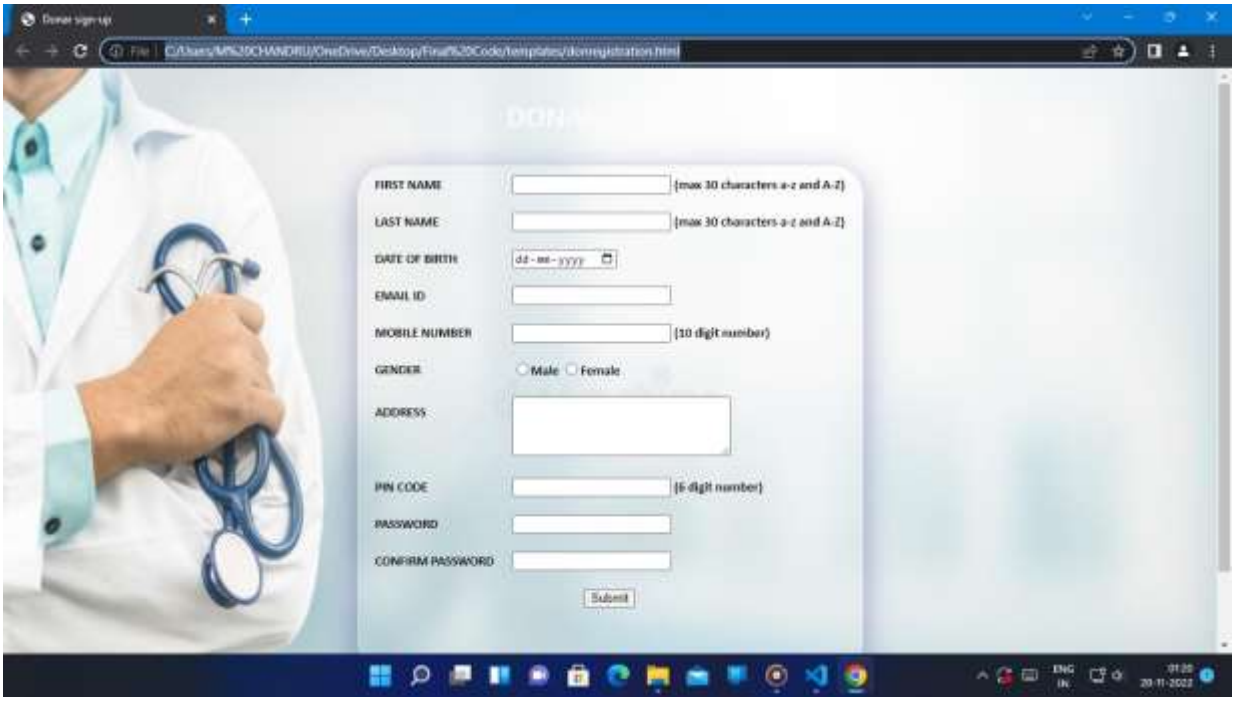




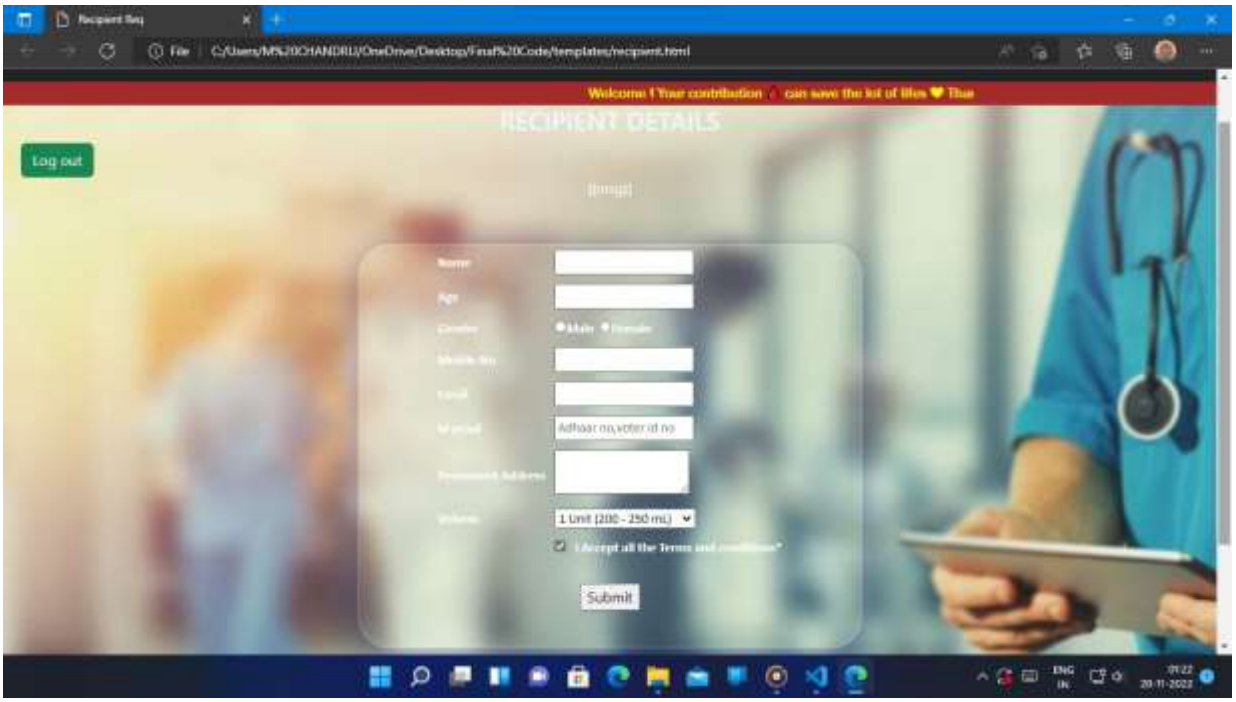
DONOR DASHBOARD



RECIPIENT LOGIN



DONER SIGN-UP



RECIPINT DASHBOARD

11 CONCLUSION

During the days of COVID-19, it is noticed the increase in blood request posts on social media such as Facebook, Twitter, and Instagram. Interestingly there are many people across the world interested in donating blood when there is a need, but those donors don’t have an access to know about the blood donation requests. This is because there is no platform to connect blood donors with the person in need. Plasma Donor App solves the problem and creates a communication channel whenever a patient needs blood. It is a useful tool to find compatible blood donors who can receive blood requests from different areas. Clinics can use this web application to maintain the blood donation activity.

Plasma Donor Application can further improve user accessibility via integrating this application with various social networks application program interfaces (APIs). Consequently, users can login and sign up using various social networks. This would increase the number of donors and enhance the process of blood donation.User interface (UI) can be improved in future to accommodate more audiences by supporting different languages across the country.

12 APPENDIX

12.1 APPLICATION SOURCE CODE

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

import ibm_db
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=19af6446-6171-4641-8aba-
9dcff8e1b6ff.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=30699;SECURITY=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=WNN14981;PWD=LsEEBW71f9uXFNsf","","")


app = Flask(__name__)


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

# admin credentials

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

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


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

@app.route('/recipientlogin')
def recipientlogin():
```

```
return render_template('reclogin.html')       ## recipt login page


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

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

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


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

return "success..."


### donar credietial

@app.route('/donregistration')
def donregistration():
```

```
return render_template('donregistration.html')  ## donar signup page uh


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


# @app.route('/donarrequest')
# def donarrequest():
#  return render_template('donar.html')  ## plasma requesting page


## donar details table

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

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

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


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

```
ibm_db.bind_param(prep_stmt, 7, address)
ibm_db.bind_param(prep_stmt, 8, pin)
ibm_db.execute(prep_stmt)
return render_template('donlogin.html', msg="Account has been created successfully..")

return "success..."


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

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


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

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

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

if account:
return render_template('donlogin.html', msg="You are already a member, please login using your
details")
else:
insert_sql = "INSERT INTO donar VALUES (?,?,?,?,?,?,?,?,?,?,?)"
ibm_db.bind_param(prep_stmt, 1, name)
```

```
ibm_db.bind_param(prep_stmt, 2, age)
ibm_db.bind_param(prep_stmt, 3, gender)
ibm_db.bind_param(prep_stmt, 4, mnumb)
ibm_db.bind_param(prep_stmt, 5, email)
ibm_db.bind_param(prep_stmt, 6, city)
ibm_db.bind_param(prep_stmt, 7, address)
ibm_db.bind_param(prep_stmt, 8, bloodgroup)
ibm_db.bind_param(prep_stmt, 9, issue)
ibm_db.bind_param(prep_stmt, 10, lastbd)
ibm_db.bind_param(prep_stmt, 11, slot)
ibm_db.execute(prep_stmt)
return render_template('donar.html', msg="Your request for donation is successfully
submitted..")

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

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

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

donar = []
sql = "SELECT * FROM donar"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
donar.append(dictionary)
```

```
dictionary = ibm_db.fetch_both(stmt)
if donar:
return render_template("plasmadon.html", donar = donar, msg="Delete successfully")


# # while student != False:
# #  print ("The Name is : ", student)

# print(student)
return "success..."


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

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


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

name = request.form['name']
age = request.form['age']
gender = request.form['gender']
mnumb = request.form['mnumb']
proof = request.form['proof']
address = request.form['address']
plasma = request.form['plasma']


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

if account:
return render_template('reclogin.html', msg="You are already a member, please login using your
details")
else:
insert_sql = "INSERT INTO recipient VALUES (?,?,?,?,?,?,?)"prep_stmt
= ibm_db.prepare(conn, insert_sql)
```

```
ibm_db.bind_param(prep_stmt, 1, name)
ibm_db.bind_param(prep_stmt, 2, age)
ibm_db.bind_param(prep_stmt, 3, gender)
ibm_db.bind_param(prep_stmt, 4, mnumb)
ibm_db.bind_param(prep_stmt, 5, proof)
ibm_db.bind_param(prep_stmt, 6, address)
ibm_db.bind_param(prep_stmt, 7, plasma)
ibm_db.execute(prep_stmt)
return render_template('recipient.html', msg="Registration succesfull for Plasma request..")


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

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


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

recipient = []
sql = "SELECT * FROM recipient"
stmt = ibm_db.exec_immediate(conn, sql)
dictionary = ibm_db.fetch_both(stmt)
while dictionary != False:
recipient.append(dictionary)
dictionary = ibm_db.fetch_both(stmt)
if recipient:
```

```
return render_template("plasmareq.html", recipient = recipient, msg="Delete successfully")

return "Deleted Successfully"


if__name___== "_main_":
app.run(debug=True)
```

12.2 REFERANCE

BOOK NAME (AUTHOR NAME & PUBLISHED YEAR)

1.A Cross-Platform Blood Donation Application with a Real-Time, Intelligent, and Rational Recommendation System *(Rashik Rahman, September 2021)*

2. Blood donor app usage behavior and perceptions: Considerations for a blood donation app *(Andrea Potgieter, May 2022)*

3. Evaluation of the Wateen App in the Blood-Donation Process in Saudi Arabia *(Tourkiah Alessa, April 2022)*

4. Location-based Mobile Application for Blood Donor Search *(Fernando Alex Sierra-Linan, January 2022*)

5. Preferences and features of a blood donation smart phone app: A multi center mixed-methods study in Riyadh, Saudi Arabia (*Afaf Ali Batis, March 2021*)

6. Instant plasma donor recipient connector web application (*Kalpana Devi Guntoju, Tejsvini Jalli, Sreejauppla, June 2022)*

12.3 LINKS

PROJECT LINK:

GITHUB LINKS: https://github.com/IBM-EPBL/IBM-Project-7562-1658892256

GITHHUB LINK (<<<Click Here<<<)

VIDEO LINK:

BACKGROUND IMAGES LINK:

HOME PAGE- 1

https://media.gettyimages.com/id/641911250/photo/businesswoman-donates-blood-on-her-lunch-hour.jpg?s=612x612&w=0&k=20&c=odf4NADe9yWkwPh0BBxwS5KDsYz2LEirRpd8Hnybj1U=

HOME PAGE-2

https://media.gettyimages.com/id/1181023667/photo/laboratory-assistant-analyzing-a-blood-sample.jpg?s=612x612&w=0&k=20&c=ahCynm6yeoyhF-A9a9eXgp9VEIRPltj4DH8exTWSTyI=

ADMIN LOGIN

https://i.pinimg.com/736x/5d/73/ea/5d73eaabb25e3805de1f8cdea7df4a42--tumblr-backgrounds-iphone-phone-wallpapers-iphone-wallaper-tumblr.jpg

ADMIN DASBOARD

DONOR & RECIPITION LOGIN

https://thumbs.dreamstime.com/b/doctor-stethoscope-hand-hospital-background-gown-94227568.jpg

DONOR & RECIPITION SIGN-UP

https://thumbs.dreamstime.com/b/doctor-hospital-background-copy-space-doctor-hospital-background-copy-space-healthcare-medical-concept-108682352.jpg

MAIL PAGE

https://regmedia.co.uk/2020/05/07/shuttrsteock_email_storm.jpg