TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 PROJECT OVERVIEW

The Plasma Donation Application is to create an e-Information about the donor and organization that are related to donating the plasma. Through this application any person who is interested in donating the blood can register himself in the same way if any organization wants to register itself with this application that can also register. Moreover if any general consumer wants to make request plasma online he/she can also take the help of this app. Admin is the main authority who can do addition, deletion, and modification if required.

1.2 PURPOSE

This project is mainly towards persons who are willing to donate plasma to the patients.Through this app it will be easier to find a donor for exract plasma and easy to build the connection between donor and plasma bank authorities.The main intend of building this software is to formal the prcedure of plasma donation and motivate donors in order to donate plasma.We have tried to maintain all information of donor which is easily understandable to the doctors which makes them easy to find the donor.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING PROBLEM

In the existing app we cannot upload and download the latest updates.Mostly the details of donations and donors were managed and maintained manualy.No use of Web Service and Remoting.That lead to risk in mismanagement and of data when the project is under development .Moreover it is less Secure .There is no proper co-ordination between different applications and users.It is fewer user friendly.There is less connection between the plasma authority and donors .

2.2 REFERENCE

1.)HTML-documentation:- **https://html.org/docs/getting-started.html**

2.)CSS-documentation:- **https://css.org/dist/latest-v14.x/docs//**

3.)Python-documentation:- **https://pyhton.com/en/starter/l**

4.)Cloud-service:- **https://docs.cloud.com/manual/tutorial/getting-started/**

5.)Github:- **https://gist.github.com/hofmannsven/6814451**

6.)W3School

7.)YouTube

2.3 PROBLEM STATEMENT DEFINITION

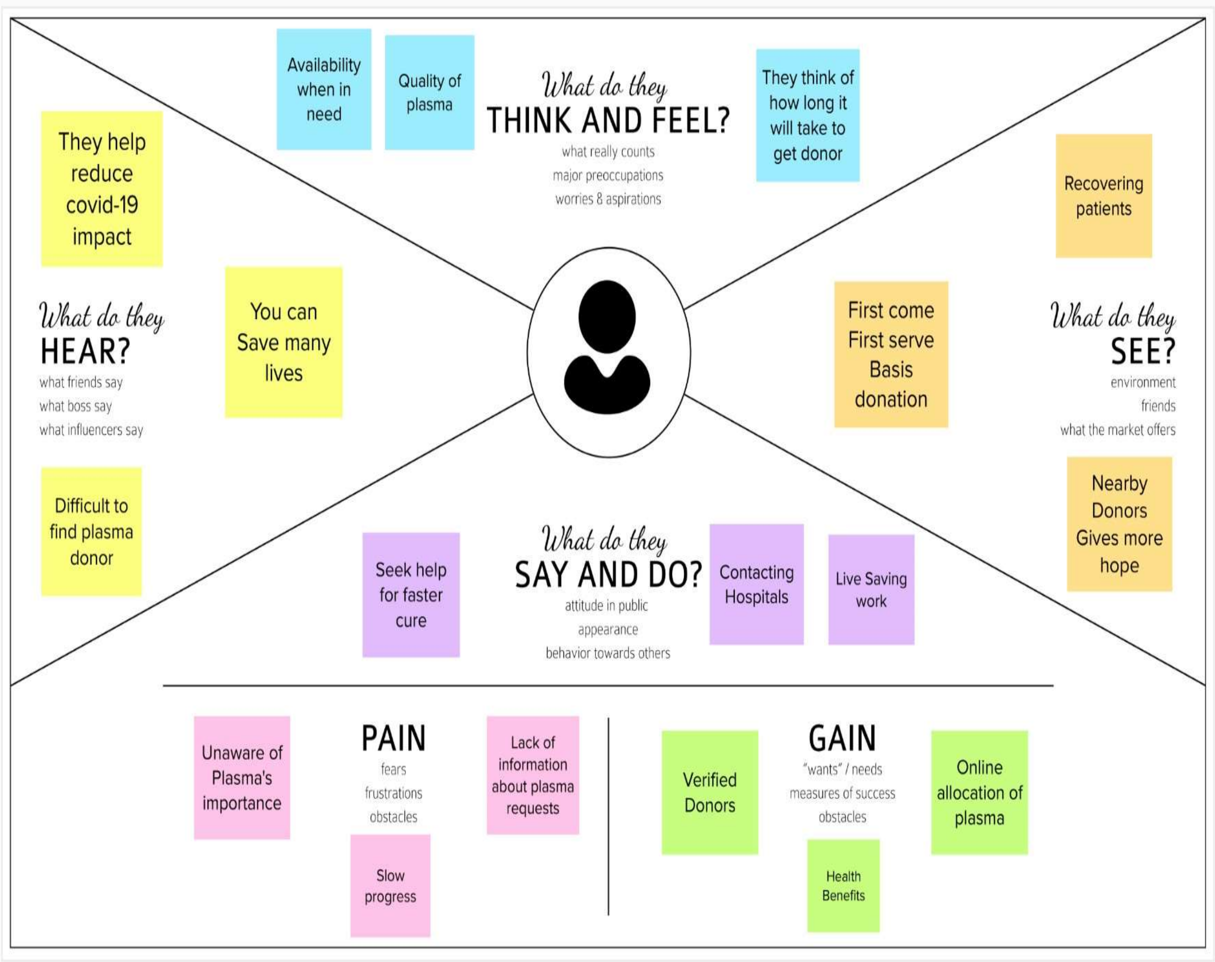
A donor who wants to donate plasma can simply upload their recovered covid19 certificate and can donate the plasma to a blood bank. The blood bank after checking the donor certificate can make a request to the donor when the donor accepts the request, they can add the required number of units they need.Aims to create a plasma donation System based on cutting-edge information technologies such as cloud computing. In addition, utilizing social media and smartphone applications worldwide is helping to make the blood donation process more suitable, offer further services, and develop blood donation centers.

CHAPTER 3

IDEATION AND PROPOSED SOLUTION

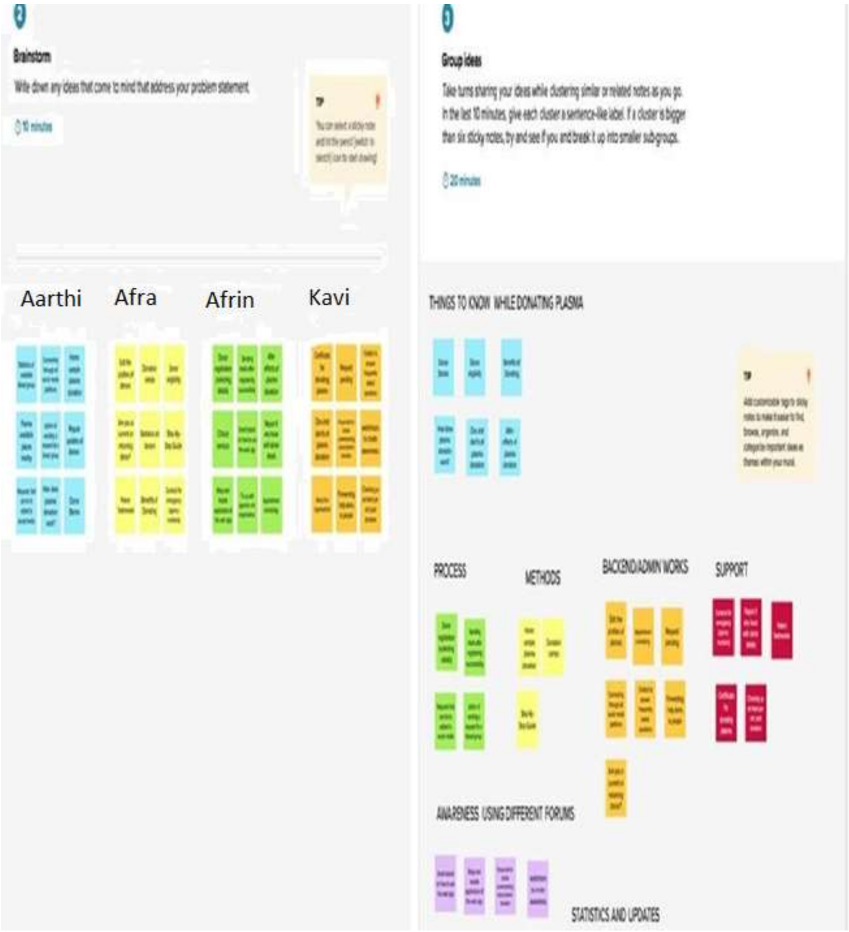
3.1 EMPATHY MAP CANVAS

An **empathy map** is a collaborative visualization used to articulate what we know about a particular type of user. It externalizes knowledge about users in order to create a shared understanding of user needs, and aid in decision making.



3.2 IDEATION & BRAINSROMING

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving.



Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

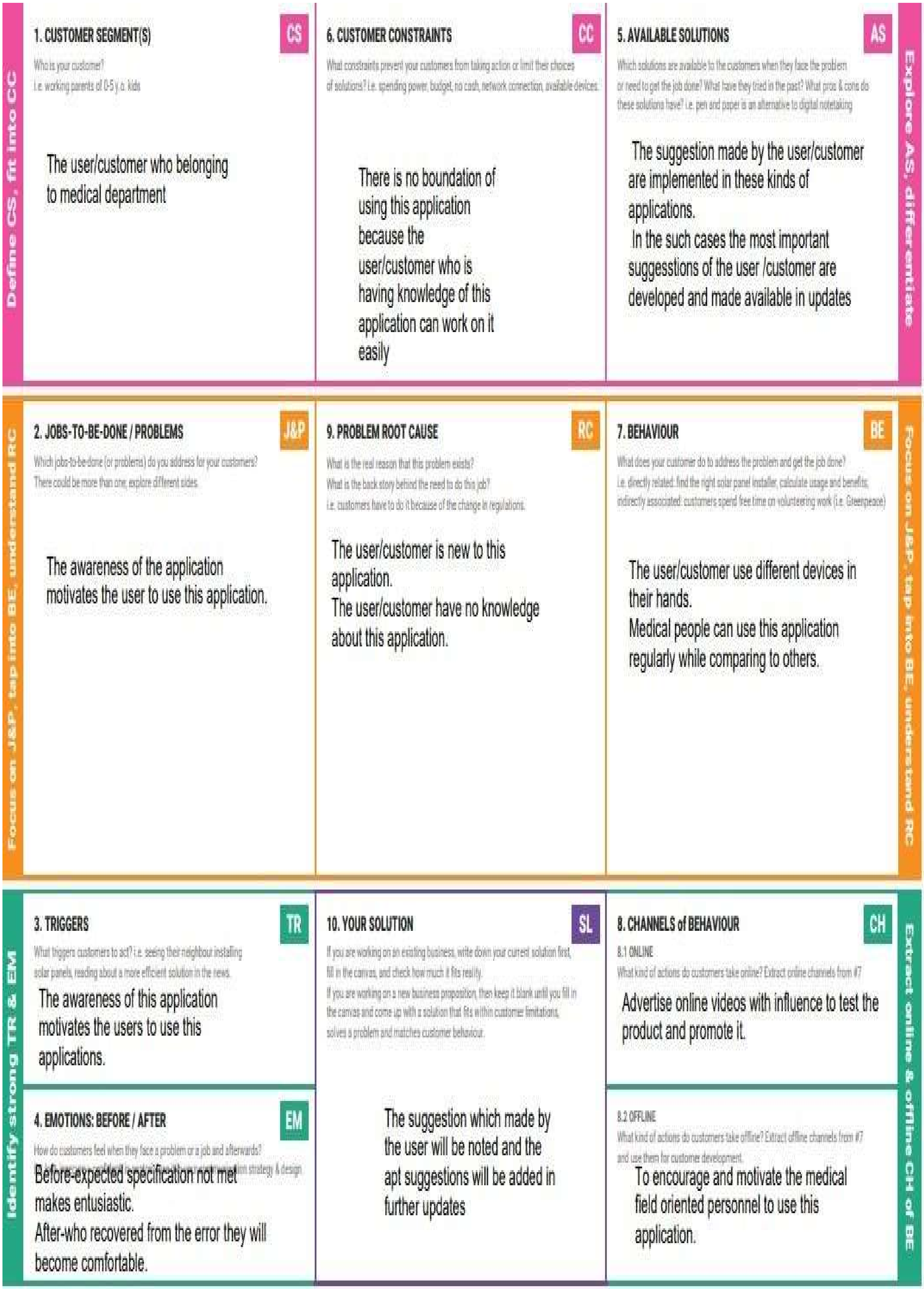
20 minutes

3.3 PROPOSED SOLUTION

This method helps the users to check the availability of donors. The user and the donor both register all relevant information A donor has to register on the website by providing their details. The registered users can get information about the donor count of each blood group.Here donor or Recipient no need to pay any money for registering or plasma donation. This application Shows plasma related Doubts and benifits in the Descrpition Section.This system can be used by any User who wants to donate or find a donor for Plasma. This could be used in Hospitals, Labs, and Health Clinics.

3.4 PROBLEM SOLUTION FIT

The Problem-Solution Fit simply means that you have found a problem with your customer and that the solution you have realized for it actually solves the customer's problem. What do you have with a Problem-Solution Fit?



Define CS, fit into CC

1. CUSTOMER SEGMENT(S)

CS

Who is your customer?
i.e. working parents of 0-5 y.o. kids

The user/customer who belonging to medical department

6. CUSTOMER CONSTRAINTS

CC

What constraints prevent your customers from taking action or limit their choices of solutions? i.e. spending power, budget, no cash, network connection, available devices.

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

5. AVAILABLE SOLUTIONS

AS

Which solutions are available to the customers when they face the problem or need to get the job done? What have they tried in the past? What pros & cons do these solutions have? i.e. pen and paper is an alternative to digital notetaking

The suggestion made by the user/customer are implemented in these kinds of applications.

In the such cases the most important suggesstions of the user /customer are developed and made available in updates

Explore AS, differentiate

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

2. JOBS-TO-BE-DONE / PROBLEMS

J&P

Which jobs-to-be-done (or problems) do you address for your customers? There could be more than one; explore different sides.

The awareness of the application motivates the user to use this application.

9. PROBLEM ROOT CAUSE

RC

What is the real reason that this problem exists?
What is the back story behind the need to do this job?
i.e. customers have to do it because of the change in regulations.

The user/customer is new to this application.

The user/customer have no knowledge about this application.

7. BEHAVIOUR

BE

What does your customer do to address the problem and get the job done?
i.e. directly related: find the right solar panel installer, calculate usage and benefits; indirectly associated: customers spend free time on volunteering work (i.e. Greenpeace)

The user/customer use different devices in their hands.

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

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

Identify strong TR & EM

3. TRIGGERS

TR

What triggers customers to act? i.e. seeing their neighbour installing solar panels, reading about a more efficient solution in the news.

The awareness of this application motivates the users to use this applications.

4. EMOTIONS: BEFORE / AFTER

EM

How do customers feel when they face a problem or a job and afterwards?
[illegible] strategy & design

Before-expected specification not met makes entusiastic.

After-who recovered from the error they will become comfortable.

10. YOUR SOLUTION

SL

If you are working on an existing business, write down your current solution first, fill in the canvas, and check how much it fits reality.
If you are working on a new business proposition, then keep it blank until you fill in the canvas and come up with a solution that fits within customer limitations, solves a problem and matches customer behaviour.

The suggestion which made by the user will be noted and the apt suggestions will be added in further updates

8. CHANNELS of BEHAVIOUR

CH

8.1 ONLINE

What kind of actions do customers take online? Extract online channels from #7

Advertise online videos with influence to test the product and promote it.

8.2 OFFLINE

What kind of actions do customers take offline? Extract offline channels from #7 and use them for customer development.

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

Extract online & offline CH of BE

CHAPTER 4

REQUIREMENT ANALYSIS

4.1FUNCTIONAL REQUIREMNT

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form (WebApp)
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Certification	After the donor donates plasma, we will give them a certificate of appreciation and authentication.
FR-4	Statistical data	The availability of plasma is given in the page as stats, which will be helpful for the users.
FR-5	User Plasma Request	Users can request to donate plasma by filling out the request form on the page. Once the request is submitted, they will get an email
FR-6	Searching/reporting Requirements	Users can use the search bar to look up information about camps and other topics.

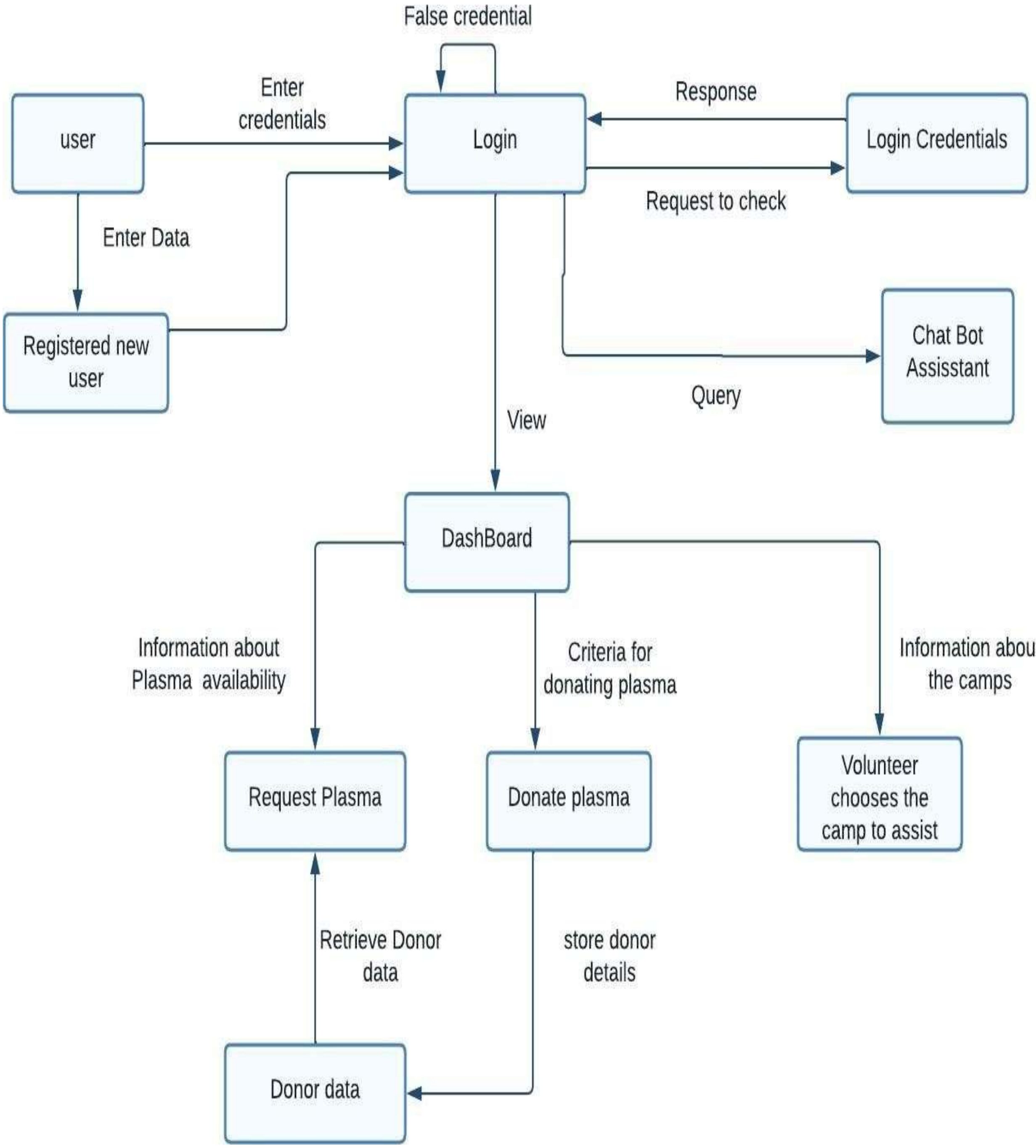
4.2NON-FUNCTIONAL REQUIREMENTS

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	Must have a good looking User friendly interface.
NFR-2	**Security**	It must be secured with the proper username andpassword.
NFR-3	**Reliability**	The system should be made in such a way that it is reliable in its operations and for securing thesensitive details.
NFR-4	**Performance**	Users should have a proper Internet Connection.
NFR-5	**Availability**	The system including the online and offlinecomponents should be available 24/7.
NFR-6	**Scalability**	The application has the ability to handle growingnumbers of users and load without compromising on performance and causing disruptions to userexperience.

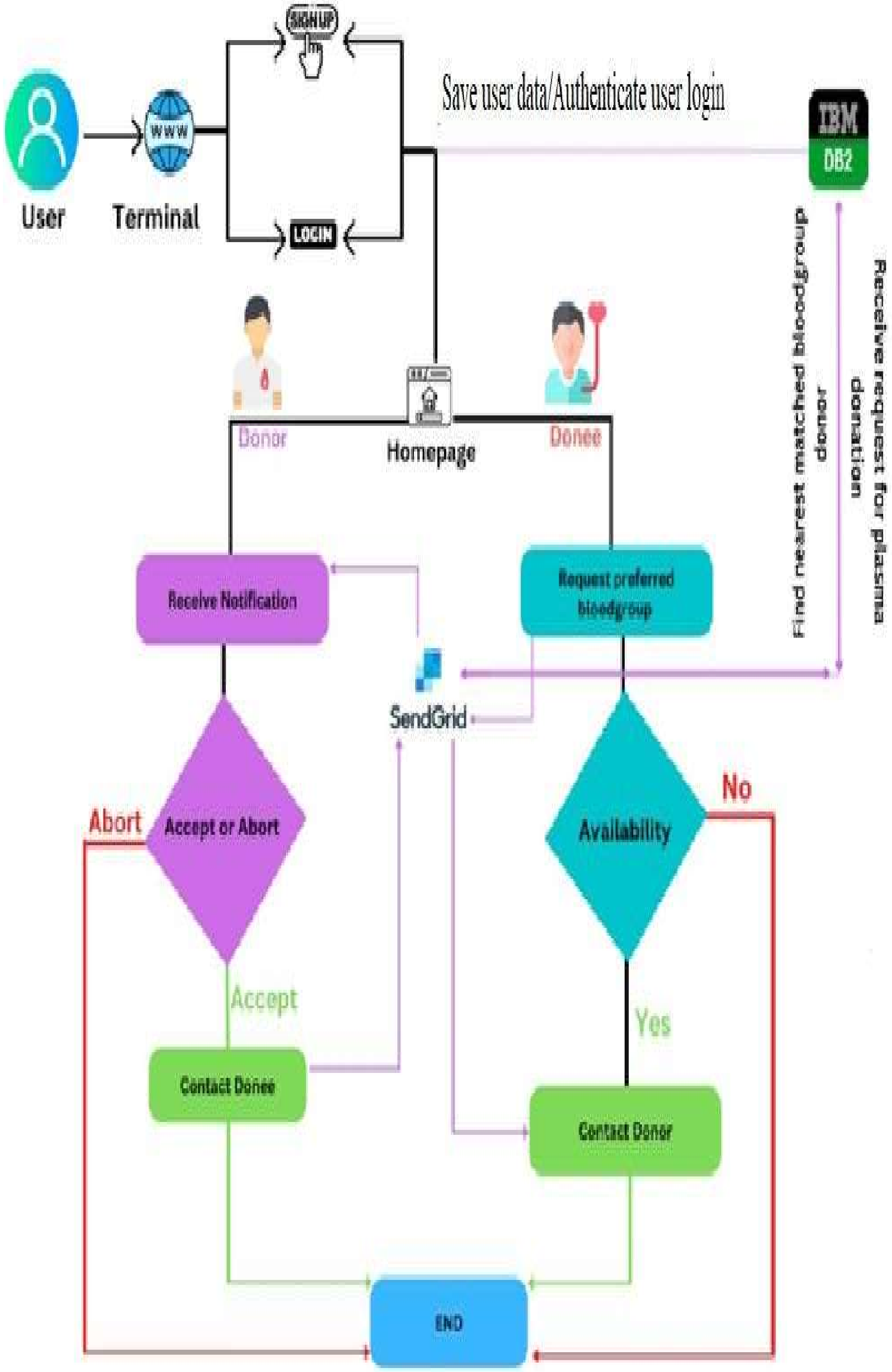
CHAPTER 5

PROJECT DESIGN

5.1DATA FLOW DIAGRAMS



5.2 SOLUTION AND TECHNICAL ARCHITECHTURE



5.3 USER STORIES

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 confirmingmy password.	I can access my account /dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation emailonce I have registered for the application	I can receive confirmationemail & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the applicationthrough Gmail	I can receive confirmation notifications through Gmail	Medium	Sprint-1
	Login	USN-4	As a user, I can log into the application by entering email & password	I can access into my User profile and view details indashboard	High	Sprint-1
	Dashboard	USN-5	As a user,I can send the proper requests todonate and obtain plasma.	I can receive appropriate notifications through email	High	Sprint-1
Customer (Web user)	Login	USN-6	As a user,I can register and application by entering email & password toview the profile	I can access into my User profile and view details indashboard	High	Sprint-1
	Dashboard	USN-7	As a user,I can send the proper requests todonate and obtain plasma.	I can receive appropriate notifications through email	High	Sprint-1
Customer Care Executive	Application	USN-8	As a customer care executive,I can try to address user's concerns and questions	I can view and address their concerns and questions	Medium	Sprint-2
Administrator	Application	USN-9	As an administrator I can help with user-facing aspects of a website, like its appearance,navigation and use of media.	I can change the appearance navigation in	Medium	Sprint-3
		USN-10	As an administrator, I can involve working withthe technical side of websites.	I can help with such as troubleshooting issues, setting up web hosts, ensuring users have access and programming servers	Medium	Sprint-1
Chatbot	Dashboard	USN-11	In addition the Customer care executive,chatbot can try to address user's concerns and questions	I can reply to all the queries related to our application	Medium	Sprint-3

CHAPTER 6

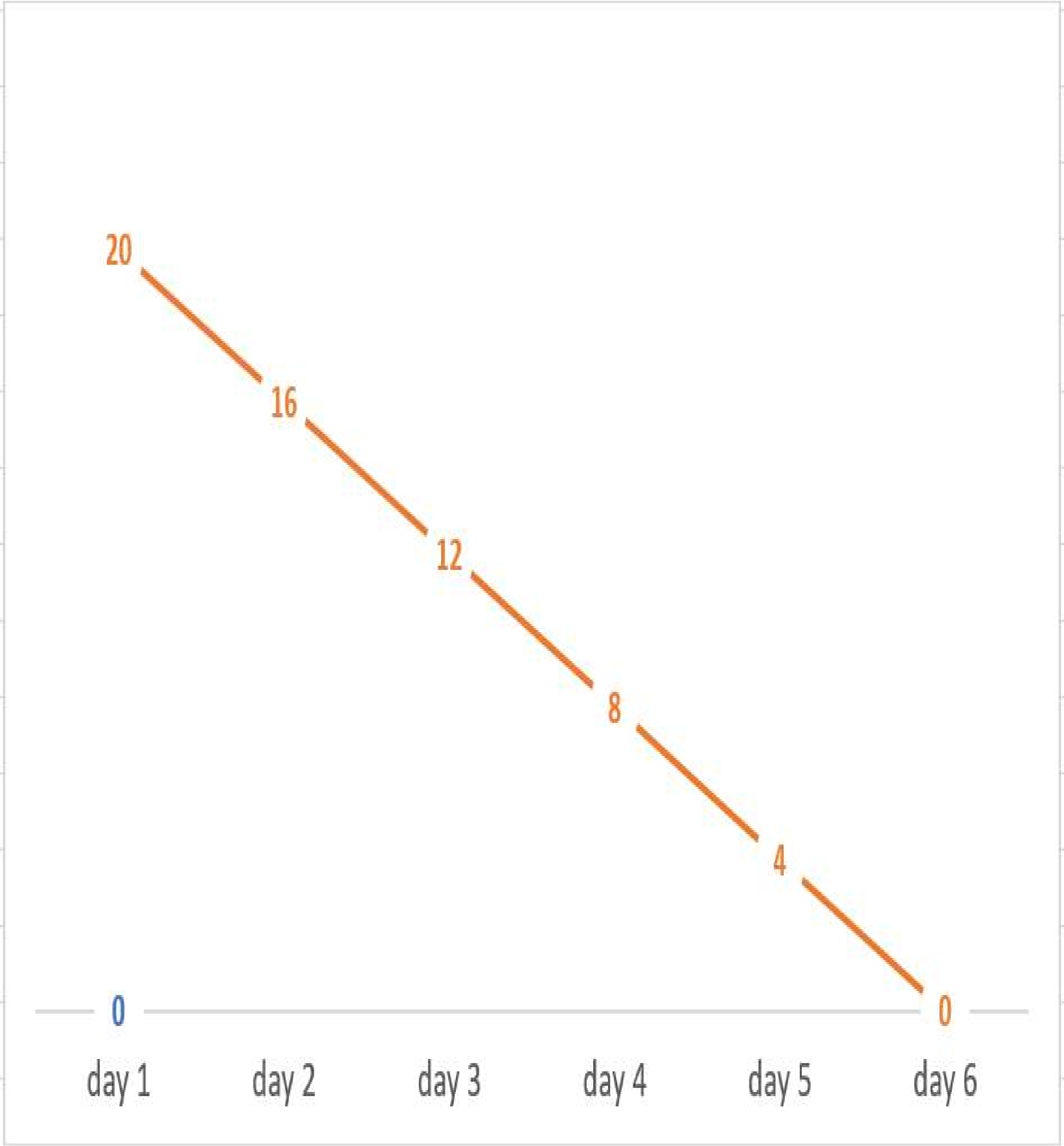
PROJECT PLANNING AND SCHEDULING

6.1 SPRINT PLANNING AND ESTIMATION

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

6.2SPRINT DELIVERY SCHEDULE

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

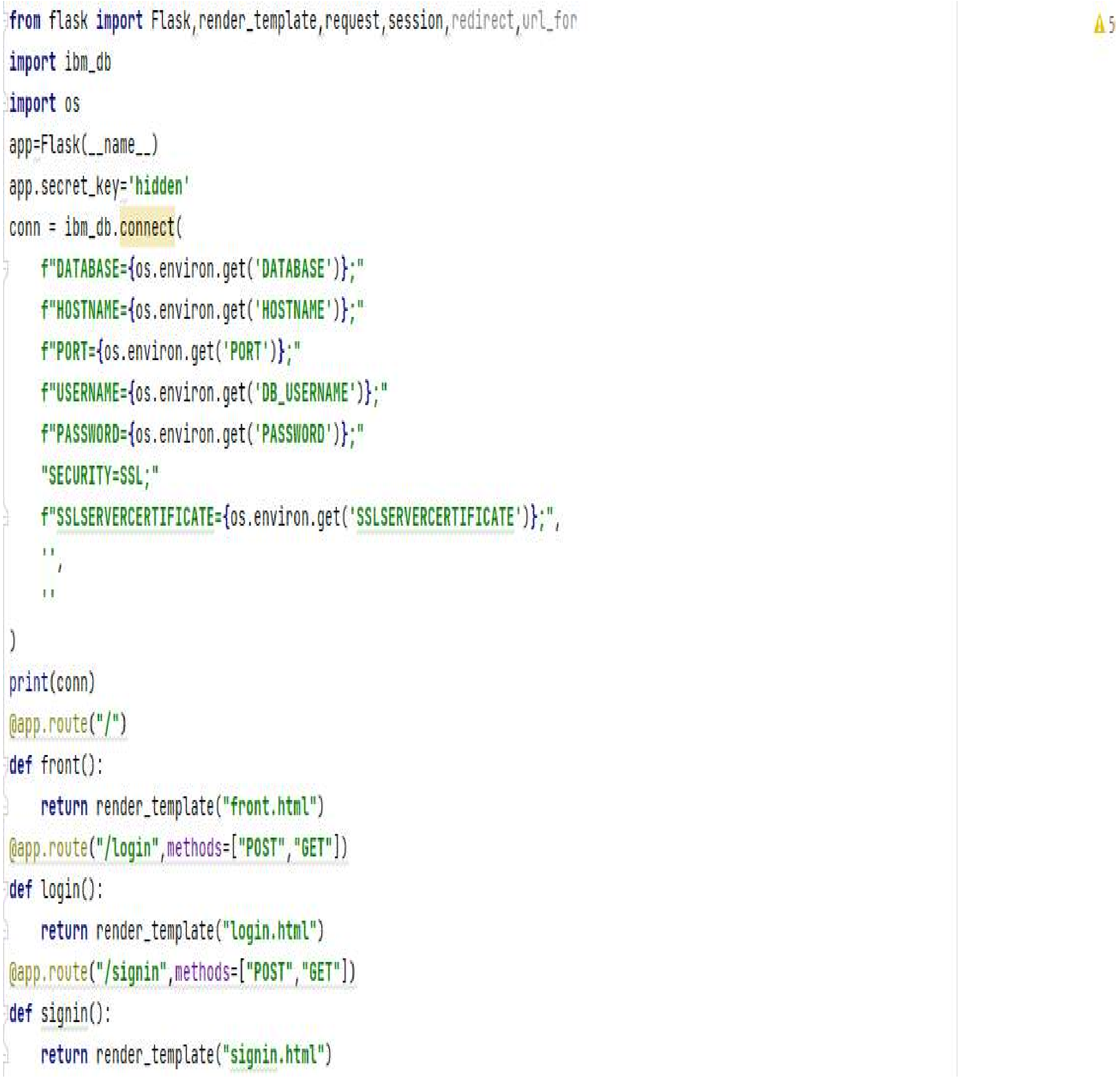


6.3 REPORT FROM JIRA

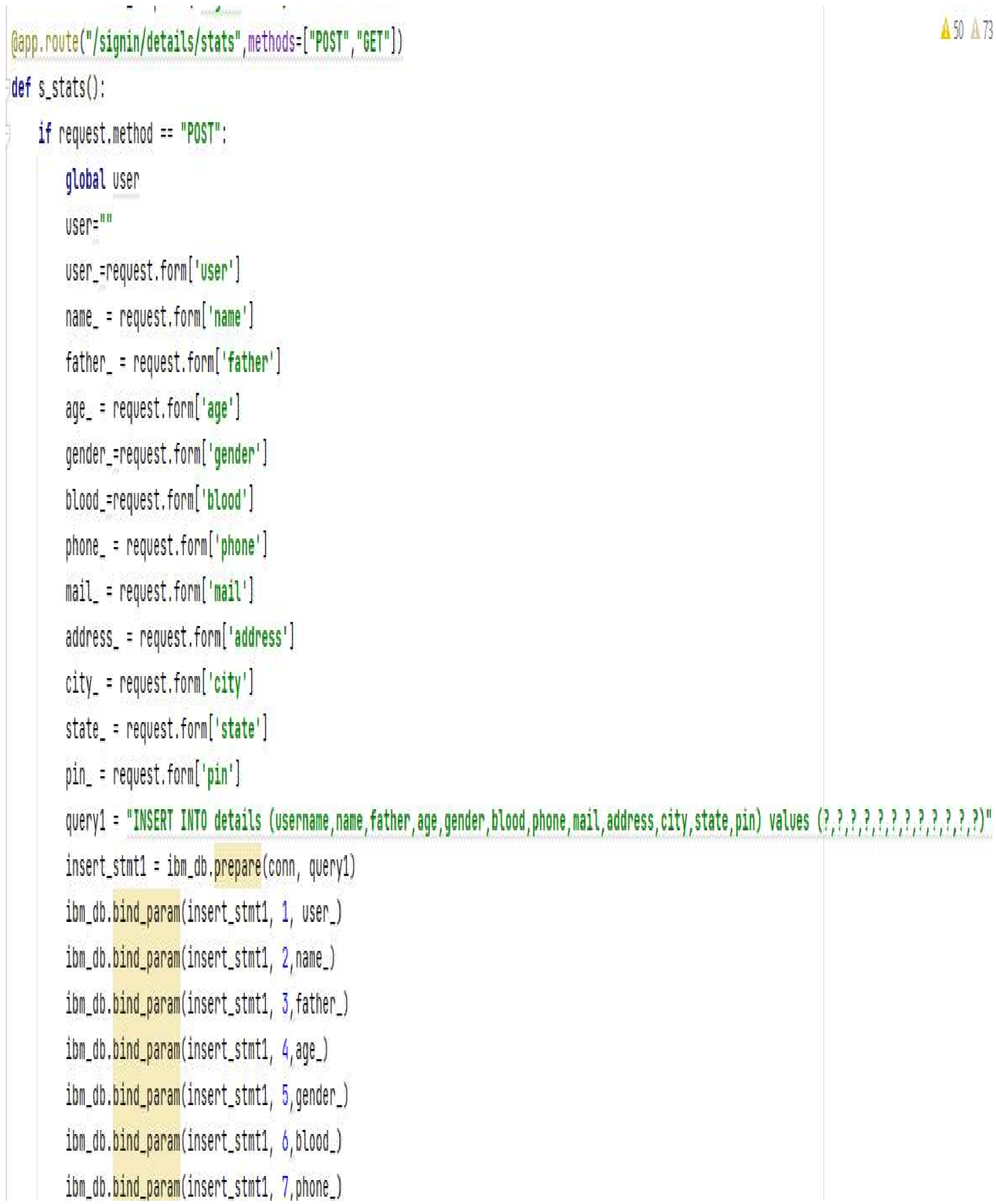
CHAPTER 7

CODING &SOLUTIONING

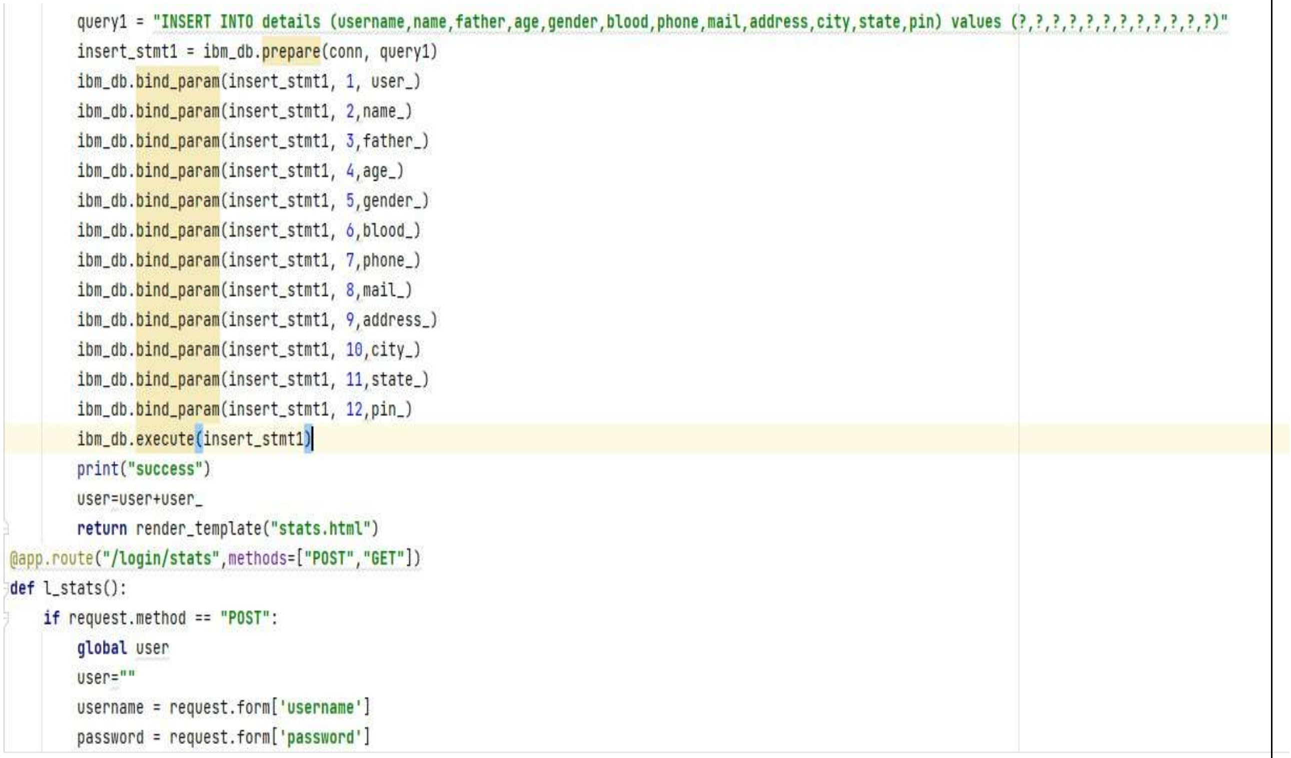
7.1FEATURE 1



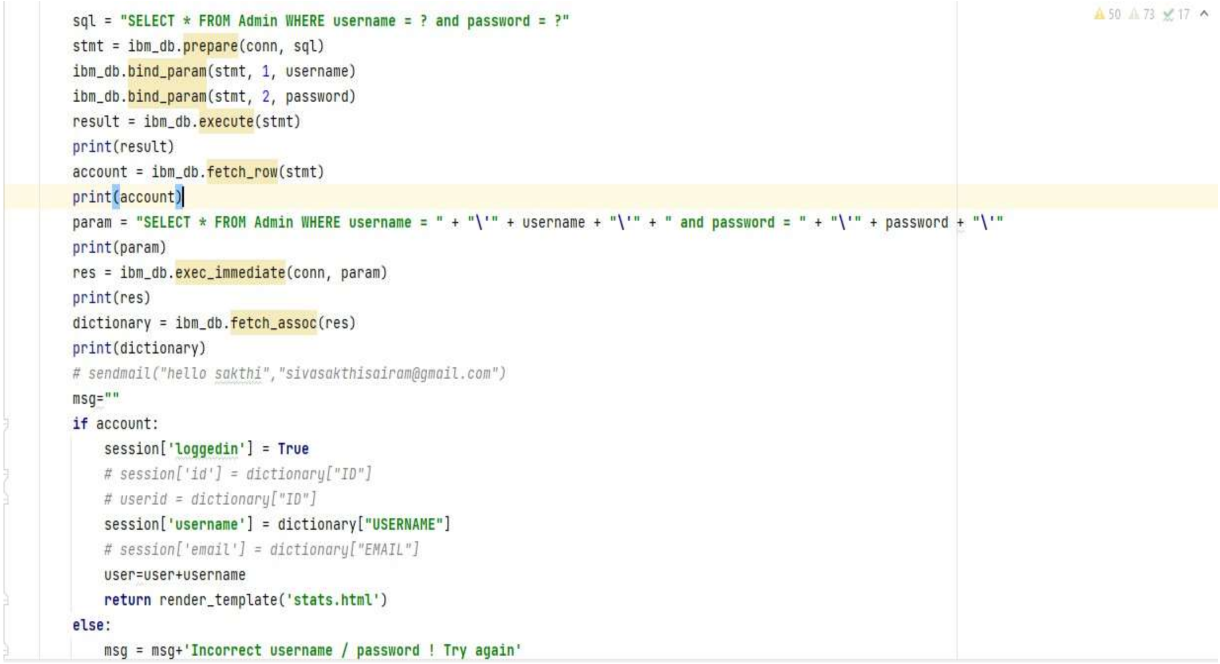
```
from flask import Flask,render_template,request,session,redirect,url_for
import ibm_db
import os
app=Flask(__name__)
app.secret_key='hidden'
conn = ibm_db.connect(
    f"DATABASE={os.environ.get('DATABASE')};"
    f"HOSTNAME={os.environ.get('HOSTNAME')};"
    f"PORT={os.environ.get('PORT')};"
    f"USERNAME={os.environ.get('DB_USERNAME')};"
    f"PASSWORD={os.environ.get('PASSWORD')};"
    "SECURITY=SSL;"
    f"SSLSERVERCERTIFICATE={os.environ.get('SSLSERVERCERTIFICATE')};",
    '',
    ''
)
print(conn)
@app.route("/")
def front():
    return render_template("front.html")
@app.route("/login",methods=["POST","GET"])
def login():
    return render_template("login.html")
@app.route("/signin",methods=["POST","GET"])
def signin():
    return render_template("signin.html")
```



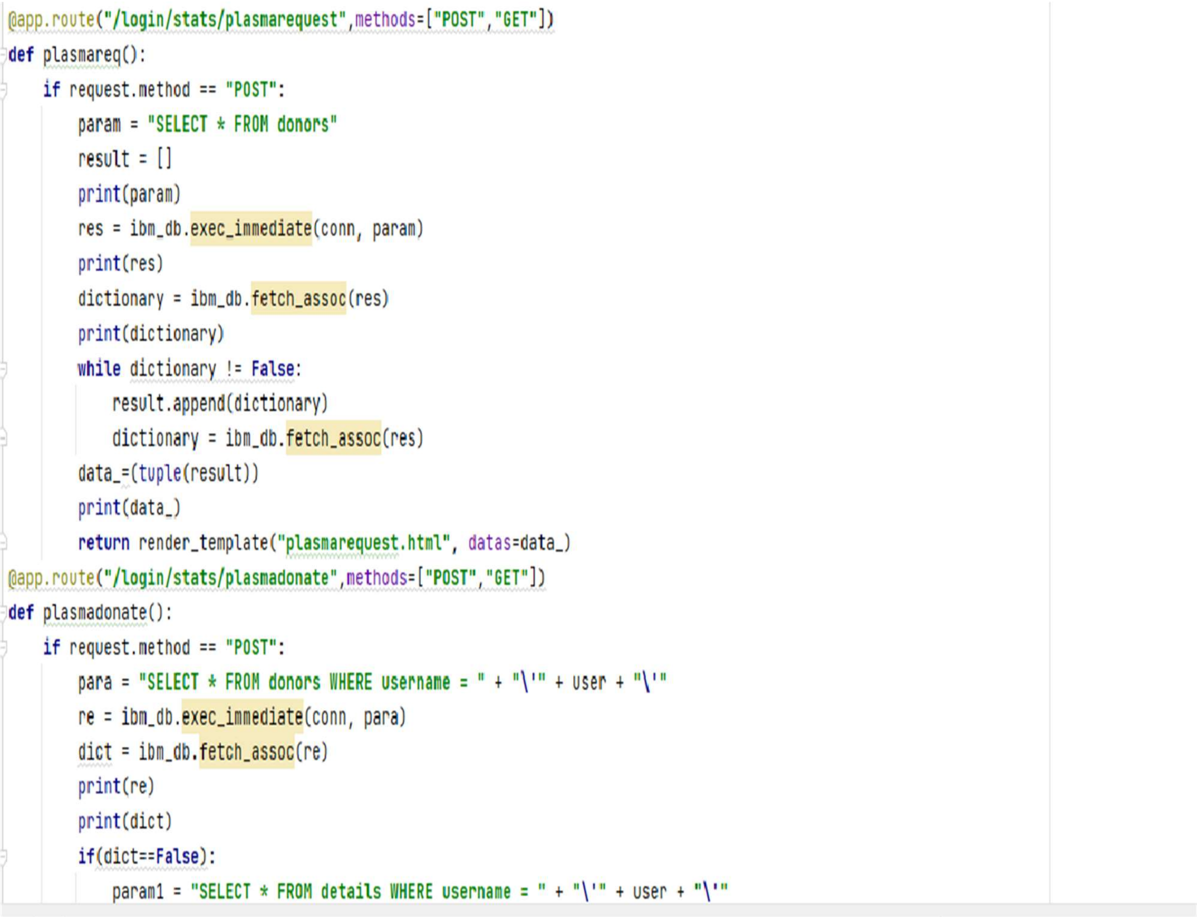
```
@app.route("/signin/details/stats",methods=["POST","GET"])
def s_stats():
    if request.method == "POST":
        global user
        user=""
        user_=request.form['user']
        name_ = request.form['name']
        father_ = request.form['father']
        age_ = request.form['age']
        gender_=request.form['gender']
        blood_=request.form['blood']
        phone_ = request.form['phone']
        mail_ = request.form['mail']
        address_ = request.form['address']
        city_ = request.form['city']
        state_ = request.form['state']
        pin_ = request.form['pin']
        query1 = "INSERT INTO details (username,name,father,age,gender,blood,phone,mail,address,city,state,pin) values (?,?,?,?,?,?,?,?,?,?,?,?)"
        insert_stmt1 = ibm_db.prepare(conn, query1)
        ibm_db.bind_param(insert_stmt1, 1, user_)
        ibm_db.bind_param(insert_stmt1, 2,name_)
        ibm_db.bind_param(insert_stmt1, 3,father_)
        ibm_db.bind_param(insert_stmt1, 4,age_)
        ibm_db.bind_param(insert_stmt1, 5,gender_)
        ibm_db.bind_param(insert_stmt1, 6,blood_)
        ibm_db.bind_param(insert_stmt1, 7,phone_)
```



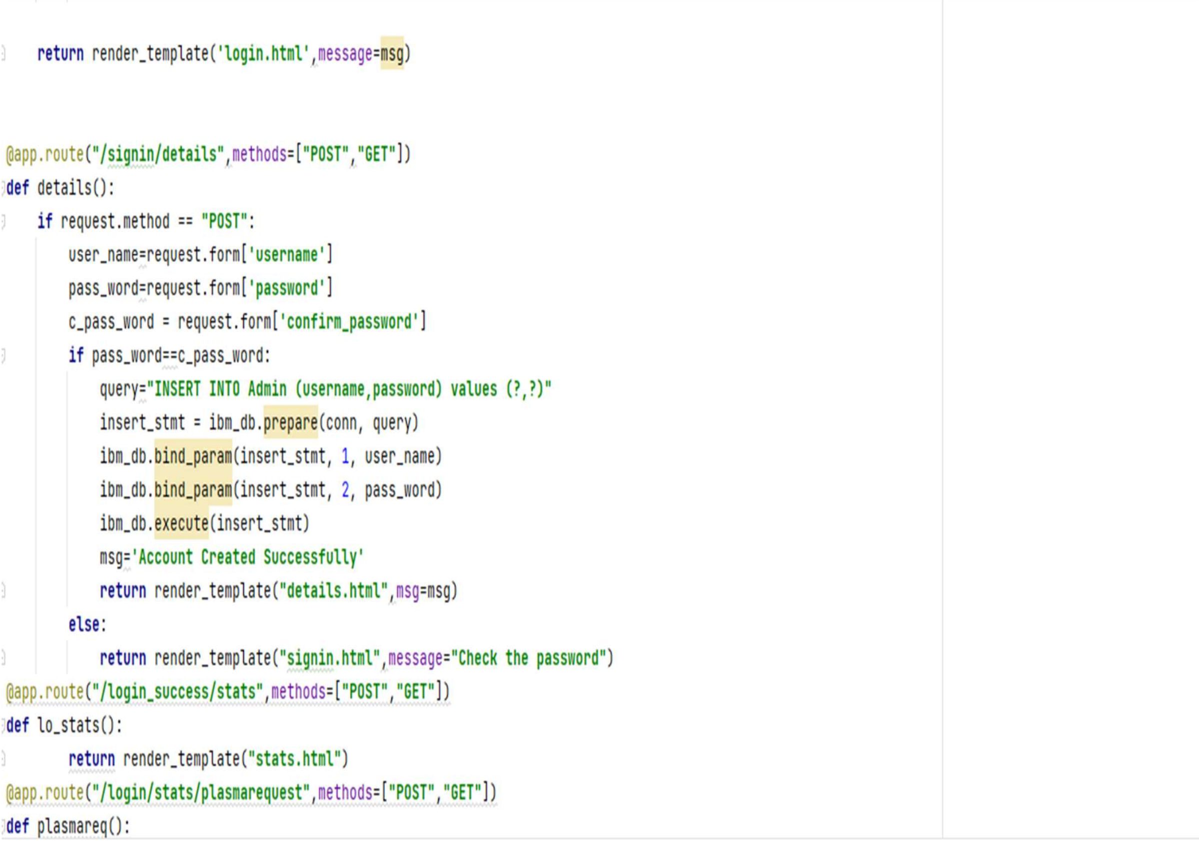
```
        query1 = "INSERT INTO details (username,name,father,age,gender,blood,phone,mail,address,city,state,pin) values (?,?,?,?,?,?,?,?,?,?,?,?)"
        insert_stmt1 = ibm_db.prepare(conn, query1)
        ibm_db.bind_param(insert_stmt1, 1, user_)
        ibm_db.bind_param(insert_stmt1, 2,name_)
        ibm_db.bind_param(insert_stmt1, 3,father_)
        ibm_db.bind_param(insert_stmt1, 4,age_)
        ibm_db.bind_param(insert_stmt1, 5,gender_)
        ibm_db.bind_param(insert_stmt1, 6,blood_)
        ibm_db.bind_param(insert_stmt1, 7,phone_)
        ibm_db.bind_param(insert_stmt1, 8,mail_)
        ibm_db.bind_param(insert_stmt1, 9,address_)
        ibm_db.bind_param(insert_stmt1, 10,city_)
        ibm_db.bind_param(insert_stmt1, 11,state_)
        ibm_db.bind_param(insert_stmt1, 12,pin_)
        ibm_db.execute(insert_stmt1)
        print("success")
        user=user+user_
        return render_template("stats.html")
@app.route("/login/stats",methods=["POST","GET"])
def l_stats():
    if request.method == "POST":
        global user
        user=""
        username = request.form['username']
        password = request.form['password']
```



```
        sql = "SELECT * FROM Admin WHERE username = ? and password = ?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        result = ibm_db.execute(stmt)
        print(result)
        account = ibm_db.fetch_row(stmt)
        print(account)
        param = "SELECT * FROM Admin WHERE username = " + "\'" + username + "\'" + " and password = " + "\'" + password + "\'"
        print(param)
        res = ibm_db.exec_immediate(conn, param)
        print(res)
        dictionary = ibm_db.fetch_assoc(res)
        print(dictionary)
        # sendmail("hello sakthi","sivasakthisairam@gmail.com")
        msg=""
        if account:
            session['loggedin'] = True
            # session['id'] = dictionary["ID"]
            # userid = dictionary["ID"]
            session['username'] = dictionary["USERNAME"]
            # session['email'] = dictionary["EMAIL"]
            user=user+username
            return render_template('stats.html')
        else:
            msg = msg+'Incorrect username / password ! Try again'
```



```
@app.route("/login/stats/plasmarequest",methods=["POST","GET"])
def plasmareq():
    if request.method == "POST":
        param = "SELECT * FROM donors"
        result = []
        print(param)
        res = ibm_db.exec_immediate(conn, param)
        print(res)
        dictionary = ibm_db.fetch_assoc(res)
        print(dictionary)
        while dictionary != False:
            result.append(dictionary)
            dictionary = ibm_db.fetch_assoc(res)
        data_=(tuple(result))
        print(data_)
        return render_template("plasmarequest.html", datas=data_)
@app.route("/login/stats/plasmadonate",methods=["POST","GET"])
def plasmadonate():
    if request.method == "POST":
        para = "SELECT * FROM donors WHERE username = " + "\'" + user + "\'"
        re = ibm_db.exec_immediate(conn, para)
        dict = ibm_db.fetch_assoc(re)
        print(re)
        print(dict)
        if(dict==False):
            param1 = "SELECT * FROM details WHERE username = " + "\'" + user + "\'"
```



```
    return render_template('login.html',message=msg)


@app.route("/signin/details",methods=["POST","GET"])
def details():
    if request.method == "POST":
        user_name=request.form['username']
        pass_word=request.form['password']
        c_pass_word = request.form['confirm_password']
        if pass_word==c_pass_word:
            query="INSERT INTO Admin (username,password) values (?,?)"
            insert_stmt = ibm_db.prepare(conn, query)
            ibm_db.bind_param(insert_stmt, 1, user_name)
            ibm_db.bind_param(insert_stmt, 2, pass_word)
            ibm_db.execute(insert_stmt)
            msg='Account Created Successfully'
            return render_template("details.html",msg=msg)
        else:
            return render_template("signin.html",message="Check the password")
@app.route("/login_success/stats",methods=["POST","GET"])
def lo_stats():
        return render_template("stats.html")
@app.route("/login/stats/plasmarequest",methods=["POST","GET"])
def plasmareq():
```

CHAPTER 8

8.1 TEST CASE

Test Case ID	Purpose	TestCases	Result
TC1	Authentication	Password with length less than 4 characters	Password cannot be less than 4 characters
TC2	Authentication	User name with length less than 2 characters	User name cannot be less than 2 characters
TC3	Authentication	Valid user name with minimum 2 characters	User name accepted

TC4	Authentication	User name left blank	User name cannot be less than 2 characters
TC5	Authentication	Password field left blank	Password cannot be empty
TC6	Authentication	Minimum 4 characters valid password	Password accepted
TC7	Authentication	Password and Confirm Password did not match	Please enter same password

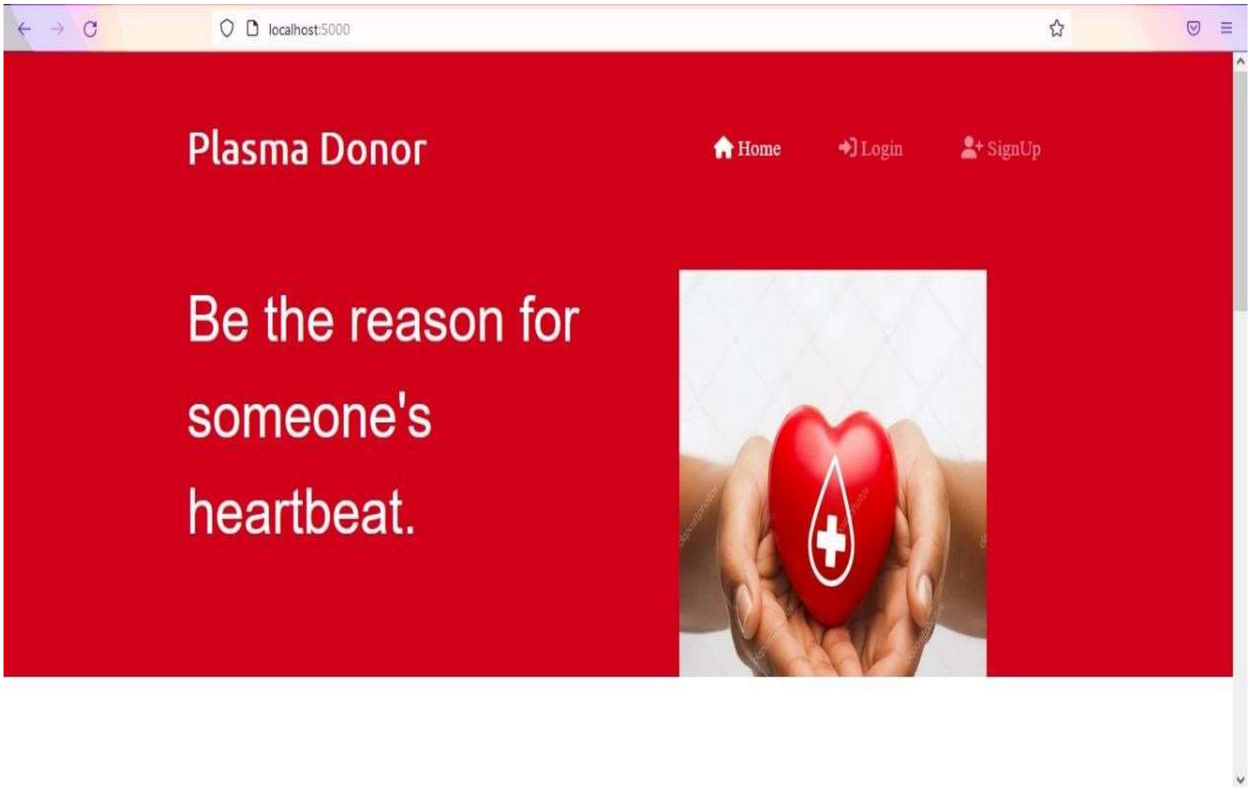
8.2 USER ACCEPTANCE TESTING

TEST CASE ID	TEST CASE DESCRIPTION
TC_001	Verify if user is able to login .
TC_002	Verify if user is able to create account.
TC_003	Verify if user can request for plasma donation.
TC_004	Verify if user can see the donors details.
TC_005	Verify if the registered voluntees details are valid one.
TC_006	Verify if the details are correctly stored in the database
TC_007	Verify if there is required storage space to store upcoming users details.

CHAPTER 9

RESULTS

9.1PERFORMANCE MATRICES



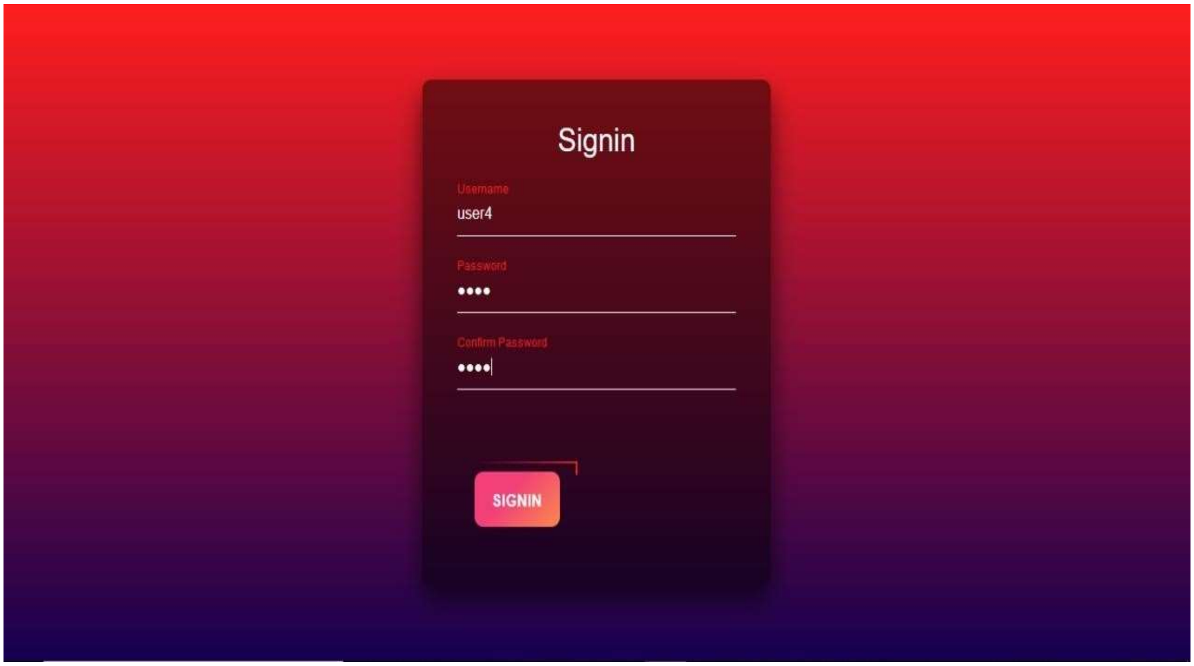
localhost:5000

Plasma Donor

Home Login SignUp

Be the reason for someone's heartbeat.

FIG.HOME PAGE

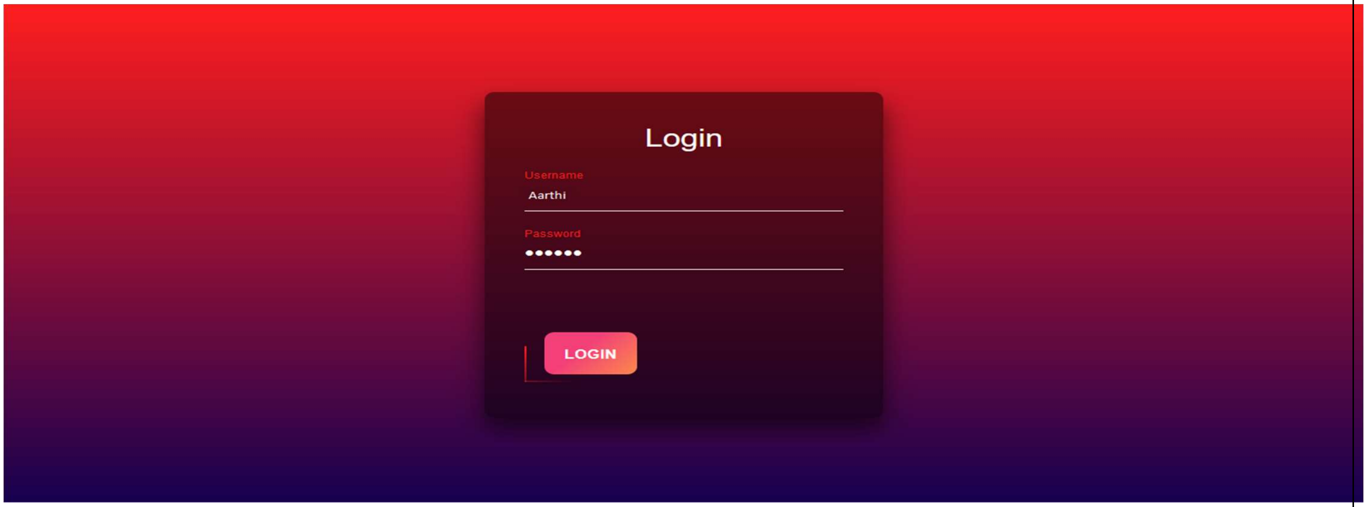

Signin

Username: user4

Password: ••••

Confirm Password: ••••

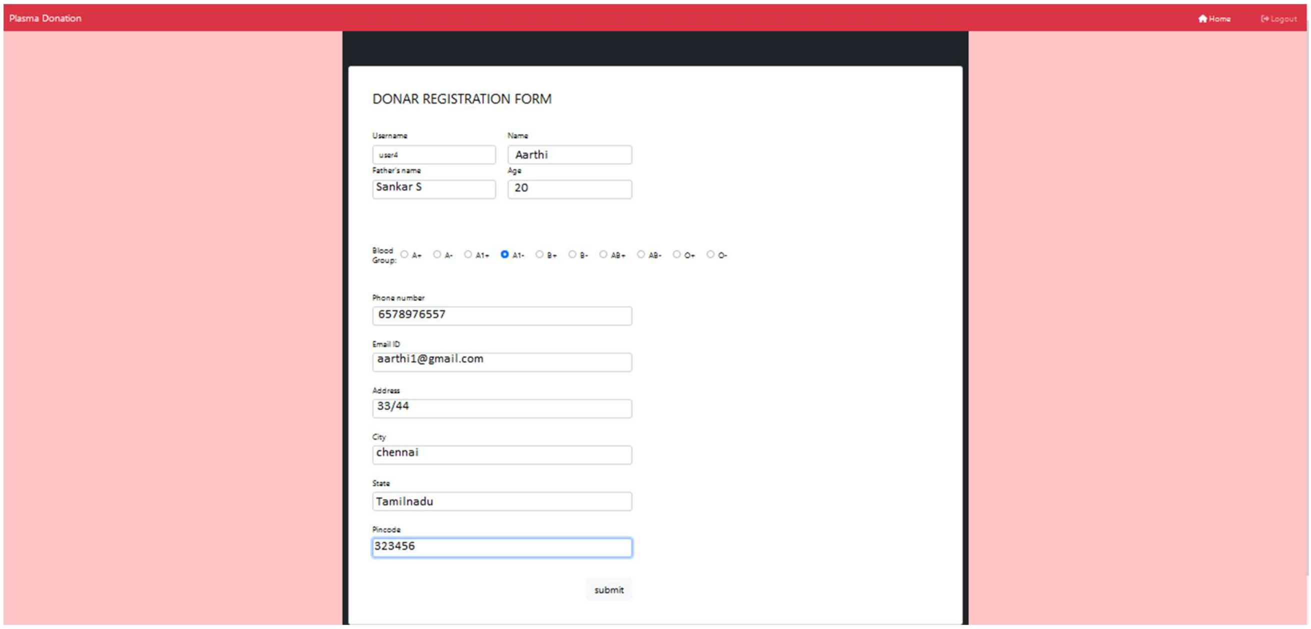
FIG.SIGIN PAGE


Login

Username: Aarthi

Password: ••••••

FIG.LOGIN PAGE



DONAR REGISTRATION FORM

Username: user4
Name: Aarthi
Father's name: Sankar S
Age: 20

Blood Group: ○ A+ ○ A- ○ A1+ ● A1- ○ B+ ○ B- ○ AB+ ○ AB- ○ O+ ○ O-

Phone number: 6578976557

Email ID: aarthi1@gmail.com

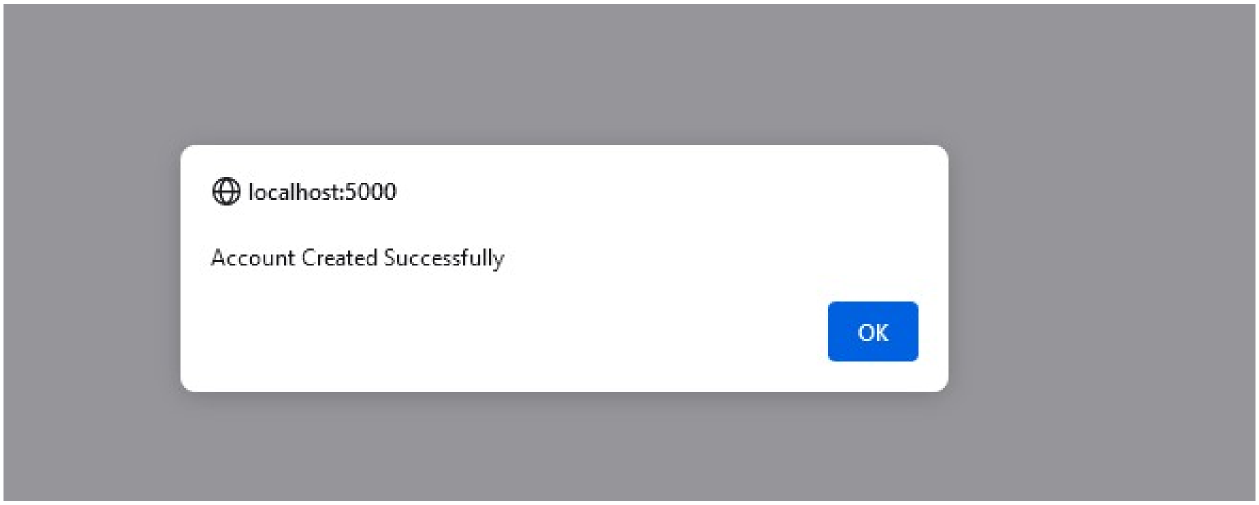
Address: 33/44

City: chennai

State: Tamilnadu

Pincode: 323456

FIG.ACCOUNT CREATION



Account Created Successfully

FIG.ACCOUNT CREATION SUCCESSFULL

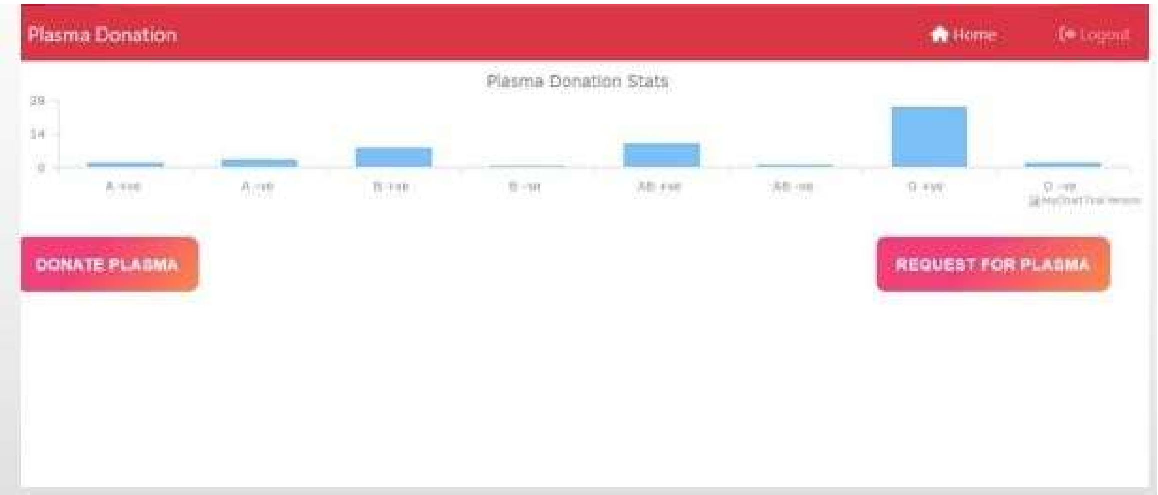
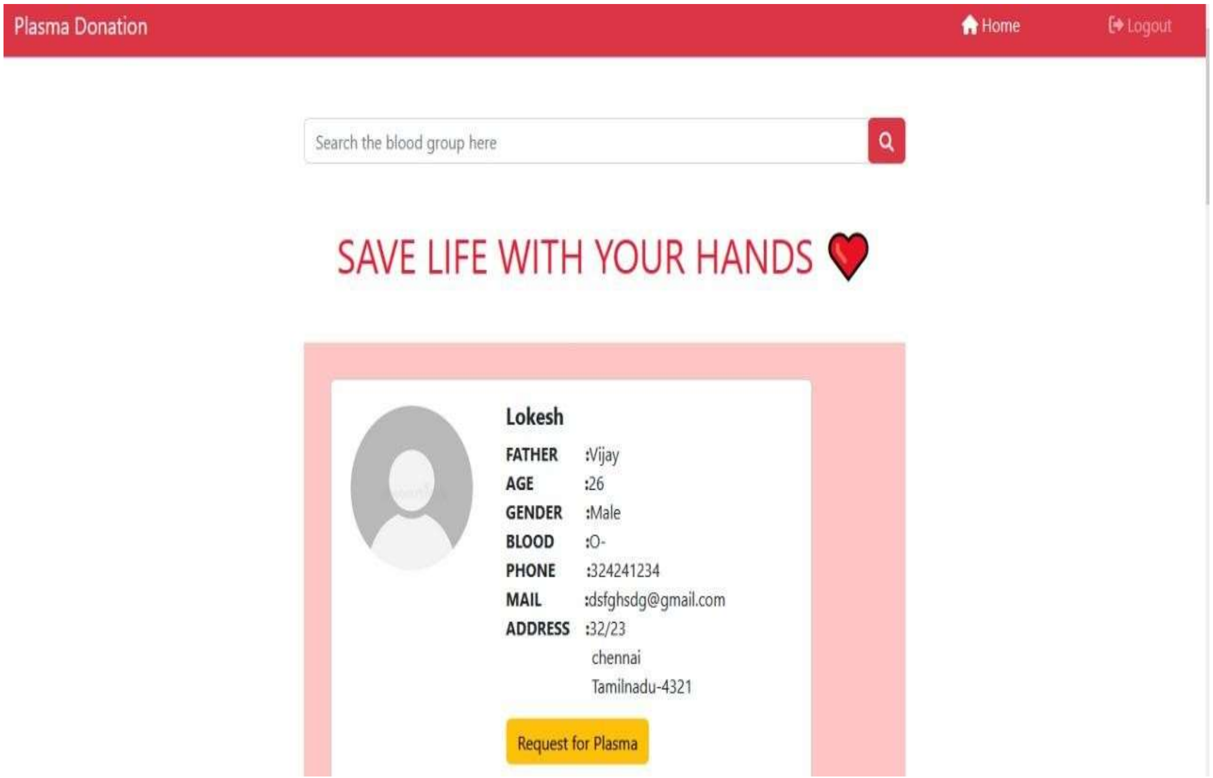


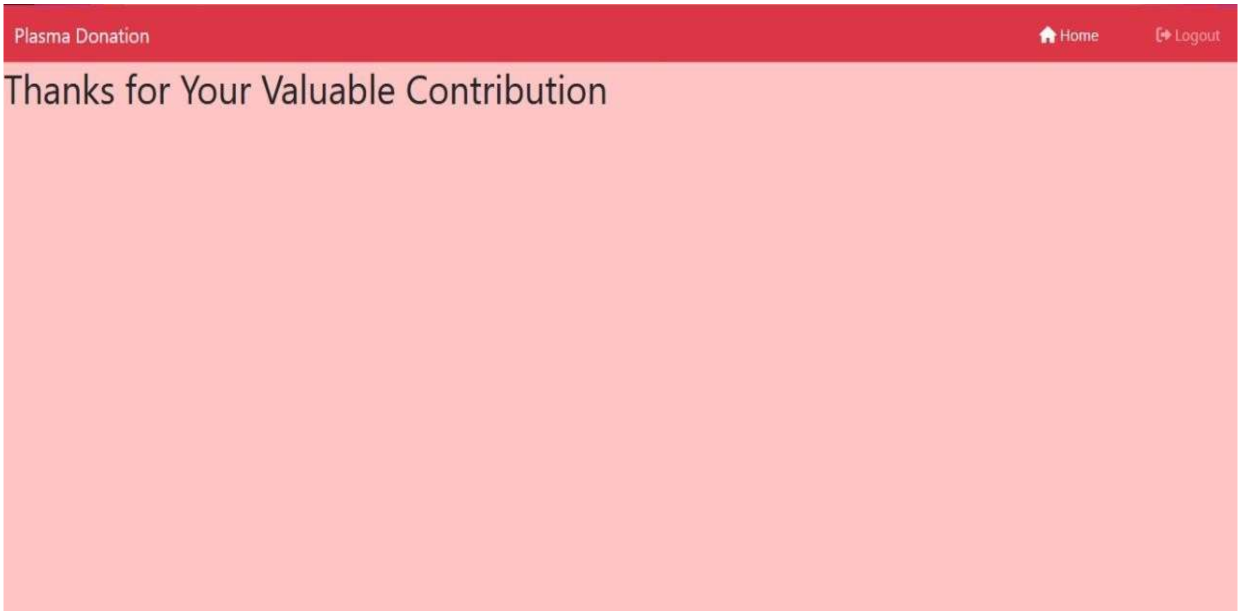
FIG.DONATION STATISTICS PAGE



Lokesh

FATHER :Vijay
AGE :26
GENDER :Male
BLOOD :O-
PHONE :324241234
MAIL :dsfghsdg@gmail.com
ADDRESS :32/23
chennai
Tamilnadu-4321

FIG. SEARCH & REQUEST PLASMA FOR DONOR PAGE

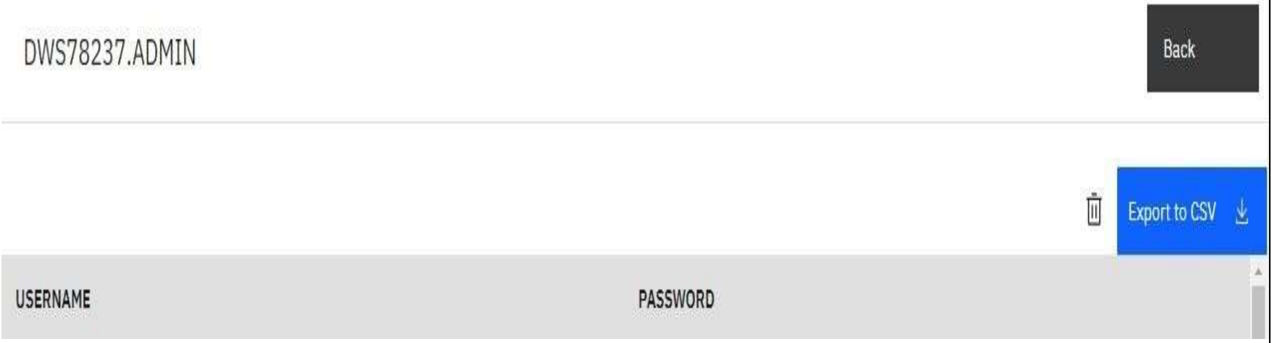


Thanks for Your Valuable Contribution

FIG .FEEDBACK PAGE

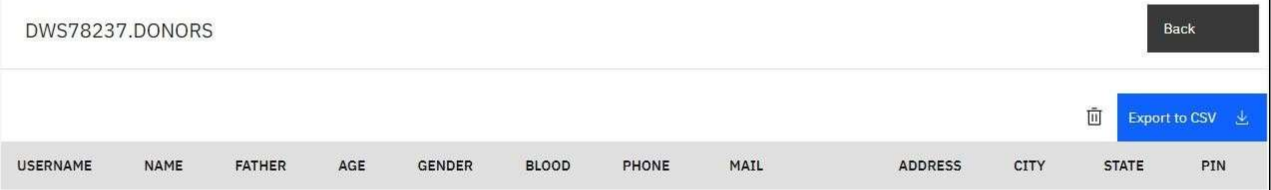
FIG.MESSAGE DELIVERABLE PAGE

ADMIN SIDE ACTIVITIES



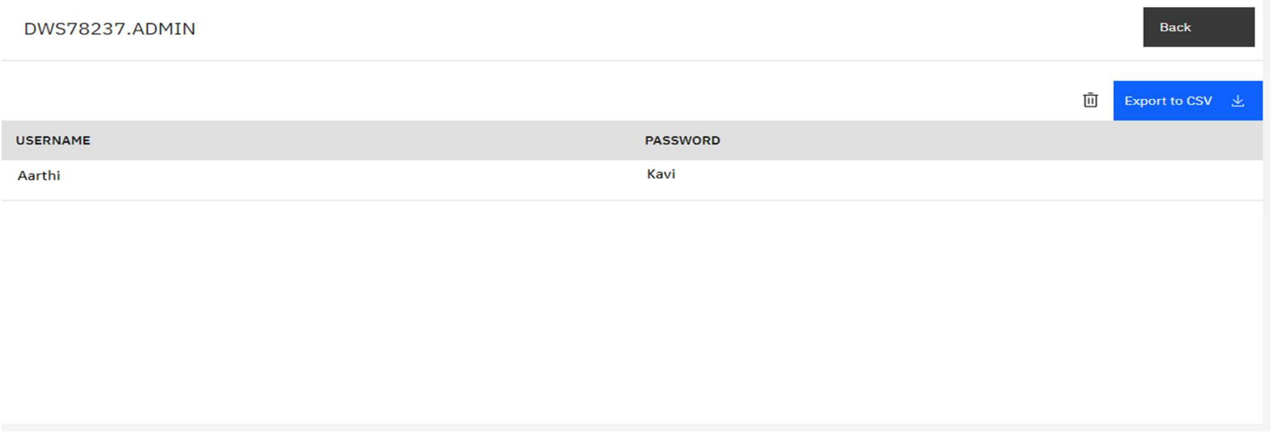
DWS78237.ADMIN

USERNAME	PASSWORD



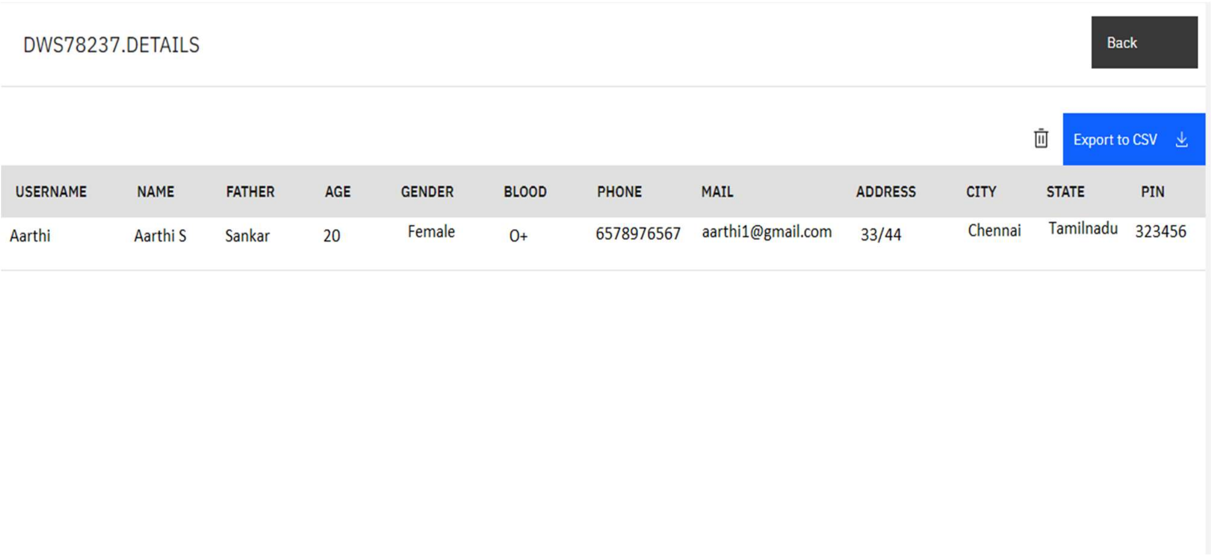
DWS78237.DONORS

USERNAME	NAME	FATHER	AGE	GENDER	BLOOD	PHONE	MAIL	ADDRESS	CITY	STATE	PIN



DWS78237.ADMIN

USERNAME	PASSWORD
Aarthi	Kavi



DWS78237.DETAILS

USERNAME	NAME	FATHER	AGE	GENDER	BLOOD	PHONE	MAIL	ADDRESS	CITY	STATE	PIN
Aarthi	Aarthi S	Sankar	20	Female	O+	6578976567	aarthi1@gmail.com	33/44	Chennai	Tamilnadu	323456

CHAPTER 10

ADVANTAGES

- User friendliness provided in the application with the various controls.
- The system makes the overall project management much easier and flexible.
- Readily upload the latest updates ,allows user to download the alertsby clicking the url.
- It provides high level of security with different level of authentication.

DISADVANTAGES

- Cannot upload and download the latest updates
- .Mostly the details of donations and donors were managed and maintained manualy.
- No use of Web Service and Remoting.That lead to risk in mismanagement and of data when the project is under development .
- Moreover it is less Secure .There is no proper co-ordination between different applications and users.
- It is fewer user friendly.There is less connection between the plasma

authority and donors .

CHAPTER 11

CONCLUSION

It has been a great pleasure to work on this exciting and challenging project. This project proved good for us,as it provided practical knowledge of not only programming in web development,python and cloud From this project, we are able to manage and get details about the palsma donors.While making this project, wegained a lot of experience of working as a team. We discovered Plasma Donor Application **IBM-Project-12536-1659453057**various predicted andunpredicted problems and we enjoyed alot solving them as a team. We adopted things like video tutorials, text tutorials, internet and learning materials to make
our project complete.

CHAPTER12

FUTURE SCOPE

The project assists well to get details about the plasma donors and individuals can make volunteer themselves by providing their details un our app

However, this project has some limitations:

The application is unable to maintain the backup of data once it

Is uninstalled.

Plasma Donor Application

IBM-Project-12536-1659453057

This application does not provide higher decision capability.

To further enhance the capability of this application, we recommend the following.

- Multiple language interface.
- Provide backup and recovery of data.
- Provide better user interface for user.
- Mobile apps advantage.

CHAPTER 13

APPENDIX

Source Code Github Link :

Plasma Donor Application IBM-Project-12536-1659453057

https://github.com/IBM-EPBL/IBM-Project-12536-1659453057

Project Demo Link :

https://drive.google.com/file/d/1CtJ_O1IBsIvabL01mAOvQSCXcEfGIzEn/view?usp=drivesdk