



NEIL GOGTE INSTITUTE OF TECHNOLOGY

(A Unit of Keshav Memorial Technical Education (KMTES))

approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)

A

SUMMER INTERNSHIP

on

BLOOD BANK MANAGEMENT SYSTEM

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

SRI GANESH : 245318733155
G VISHAL : 245318733139
MVS PRANAVI : 245318733153

Under the Guidance of

MRS V. VIJAYA MADHAVI



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Neil Gogte Institute of Technology

Kachawanisingaram Village, Hyderabad, Telangana 500058.

March 2021



NEIL GOGTE INSTITUTE OF TECHNOLOGY

(A Unit of Keshav Memorial Technical Education (KMTES))

approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)

CERTIFICATE

This is to certify that the project report titled “**BLOOD BANK MANAGEMENT SYSTEM**” is being submitted by **SRI GANESH (245318733155), G VISHAL (245318733139), MVS PRANAVI (245318733153)** IV-year B.E.VII Semester **Computer Science and Engineering** is a record of bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Internal Guide

HOD

External Examiner



NEIL GOGTE INSTITUTE OF TECHNOLOGY

(A Unit of Keshav Memorial Technical Education (KMTES))

approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)

DECLARATION

We hereby declare that the Summer Internship entitled, “**BLOOD BANK MANAGEMENT SYSTEM**” submitted for the B.E degree is entirely our work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree.

Date:

1. SRI GANESH

(2453-18-733-155)

2. G VISHAL

(2453-18-733-139)

3. MVS PRANAVI

(2453-18-733-153)

ACKNOWLEDGEMENT

We are happy to express our deep sense of gratitude to the principal of the college **Dr D Jaya Prakash**, Professor, Neil Gogte Institute of Technology, for having provided us with adequate facilities to pursue our project.

We would like to thank **Dr K.V. Ranga Rao**, Professor and Head, Department of Computer Science and Engineering, Neil Gogte Institute of Technology, for having provided the freedom to use all the facilities available in the department, especially the laboratories and the library.

We are very grateful to our project guide “**Vijaya Madhavi**”, Designation, Department of Computer Science and Engineering, Neil Gogte Institute of Technology, for his extensive patience and guidance throughout our project work.

We sincerely thank our seniors and all the teaching and non-teaching staff of the Department of Computer Science & Engineering and Information Technology for their timely suggestions, healthy criticism and motivation during the course of this work.

We would also like to thank classmates for always being there whenever we needed help or moral support. With great respect and obedience, we thank our parents, sisters and brother who were the backbone behind our deeds.

Finally, we express our immense gratitude with pleasure to the other individuals who have either directly or indirectly contributed to our need at the right time for the development and success of this work.



ABSTRACT

The basic building aim is to provide blood donation service to the city recently. Blood Bank Management System (BBMS) is a Web-based application that is designed to store, process, retrieve and analyse information concerned with the administrative and inventory management within a blood bank. This project aims at maintaining all the information pertaining to blood donors, different blood groups available in blood bank and help them manage in a better way. Also, project aim is to provide transparency in this field, make the process of obtaining blood from a blood bank hassle-free and corruption-free and make the system of blood bank management effective.

CONTENTS

1. INTRODUCTION

- 1.1 Motivation
- 1.2 Problem Statement
- 1.3 Solution
- 1.4 Objective
- 1.5 Limitations

2. LITERATURE SURVEY

- 2.1 Surveys
- 2.2 Existing Systems
- 2.3 Disadvantages of Existing systems
- 2.4 Proposal systems
- 2.5 Advantages of proposal systems
- 2.6 Conclusions

3. ANALYSIS

- 3.1 Software and Hardware Requirements
- 3.2 Contents diagram
 - a. Architecture Diagram
 - b. Data Flow Diagram



NEIL GOGTE INSTITUTE OF TECHNOLOGY

(A Unit of Keshav Memorial Technical Education (KMTES))

approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)

4. DESIGN

4.1 Use Case Diagram

4.2 Schema Diagram

4.3 ER Diagram

5. IMPLEMENTATION

5.1 Source code

6. TESTING AND VALIDATION

6.1 Project Screenshots

7. CONCLUSION AND FUTURE ENHANCEMENT

8. REFERENCES

1. INTRODUCTION

The project blood bank management system is known to a pilot project that is designed for the blood bank to gather blood from various sources and distribute to the needy people who have high requirements for it. The software designed to handle the daily transaction of the blood bank and search the details when required. It also helps the details of the donors, blood collection details as well as blood issued reports. The software Application is designed in such a manner that can suit the needs of all the blood bank requirements in the course of future. It will help us to find the blood group with its most efficient time to take care of blood and it easier to hand over the blood to the hospital to help to get blood on time. This all things are been stored and been seen in this blood bank management system. To help more people trying best to do so.

1.1 Motivation

Today you can easily connect with anything through internet services. Blood bank aims serving for human welfare. Many people here for you to help you, willing to donate blood anytime. You can help by registering on blood bank if you're willing to donate your blood when needed. As a proud member of blood bank and a responsible human being you can help someone in need.

Person who needs to donate blood may register on website with the help of username and password. The persons who need blood donor, they can search and find blood donors by using website. After searching list of donors will be displayed and user can get brief details about their contact details, email including their location, so they can communicate.

1.2 Problem Statement

Despite advances in technology, nowadays, most blood bank systems are running in manual system. 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 addition, it seems that there is lack of proper documentation about blood donors and its medical history. This may lead to blood bag contamination and may affect the blood transfusion safety. Generally, this study aims to determine how the use of online bank management system enhance blood transfusion safety. Subsequently, this study seeks to answer the following specific problems:

1. What is the level of perception among blood bank's stakeholders on manual-based system?
2. What is the level of perception among blood bank's stakeholders on online blood bank management system?

1.3 Solution

The findings of this study will benefit blood banks in managing blood donation donors, activities, and blood bags. This will allow the hospital to take decision if a particular type of blood is needed and currently unavailable in the hospital, however, available in another nearby hospitals. Furthermore, managing the blood bags in the blood bank will be much easier because each blood bag has an information about the donor, donation activity details, and the expiration date. Also, doctor can use this system to serve blood bags to their patient and monitor the details of the donor.

The main advantages of the system are:

- ☐ Blood bank staff can find and manage the donor details on the system easily.
- ☐ The expiration date of blood bags can be viewed in the system.
- ☐ Hospital can be alerted about issued blood bags and its availability.

- ☐ The system is systematized, and organized in managing blood donor records and blood donation activities.

1.4 Objective

This applied research aims to design, develop and implement online blood bank management system. This web-based application provides:

- ☐ To ensure hospital to have good supply or inventories of blood bags.
- ☐ To check the availability of blood bags anytime.
- ☐ To manage the information of its blood donor.
- ☐ Function to check if the person donate blood for the last 3 months.
- ☐ To allow good documentation about the donor and its blood donation activities.
- ☐ Support fast searching to find match blood bags for the right person.

1.5 Limitations

This research study does not cover the actual blood collection activity, and actual blood transfusion operation. Blood donors and patients or recipients of blood donation are not system users, their registration or information will be encoded by the blood bank receptionists.

2. LITERATURE SURVEY

2.1 Surveys

The purpose of this survey to find a way to implement a system that will provide a solution not only to blood centres but able to the numerous patients and willing blood donors. To do that a lot of effort has been put to study a number of researches in this field and to gather enough information that will help achieve that goal.

In order to appreciate the facts within the research, there is necessary to analyse the current state and the overall nature of the blood bank management system as well as the effort being put in order to appreciate the centralized blood bank repository.

2.2 Existing Systems

Existing system of blood bank is very much impeded and disrupt. There is a constant need to replenish stocks in blood bank. Blood bank faces lack of donors resulting in shortage of blood units. Hence customers have to go around in search of blood unit.

2.3 Disadvantages of Existing System

Since most blood banks are still in paper-based system, various disadvantages are healthcare system. As such, the researchers aimed to design, develop, and implement an online blood bank management system (OBBMS). This web-based application allows hospital to check the availability of blood bags anytime.

2.4 Proposed System

The aim of proposed system is to develop a system of improved facilities. The proposed system can overcome all the limitations of the existing system. The system, provide proper security and reduces the manual work.

2.5 Advantages of proposed system

Ensure data accuracies. Proper control of the higher officials.

Minimum time needed for the various processing and Greater efficiency and better service.

2.6 Conclusion

The purpose of these was to collect information on how an information system helped the management of blood banks. Based on the reviews, it was found out that web-based blood bank systems provide convenience, efficiency and security to the system users and hospitals compared to the manual systems. It was found out that manual systems have many disadvantages that disappoint and dissatisfy users. Indeed, online blood bank applications make work easy, and ensures fast retrieval of data when needed.



3. ANALYSIS

3.1 Software and Hardware Requirements

Hardware Requirements

PROCESSOR: intel Pentium or Higher Version

RAM: Minimum GB

HARD DISK: 60GB and above

Software Requirements

SOFTWARE: Python 3.3 or greater

DATABASE: SQLite

SUPPORTED BROWSERS: Google Chrome / Mozilla Firefox / Internet Explorer

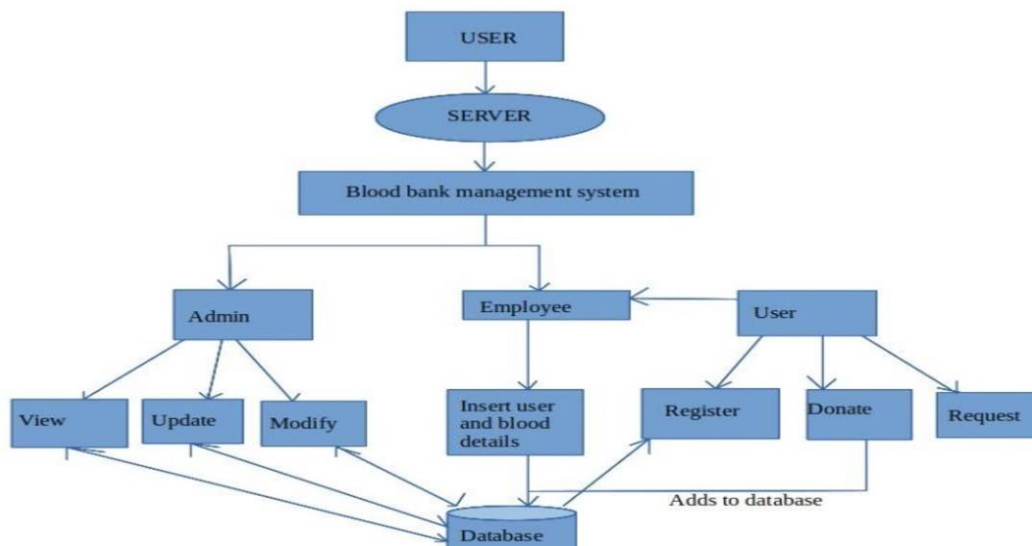
EDITOR: Visual Studio Code

FRAMEWORK: Flask 1.1.1, Bootstrap

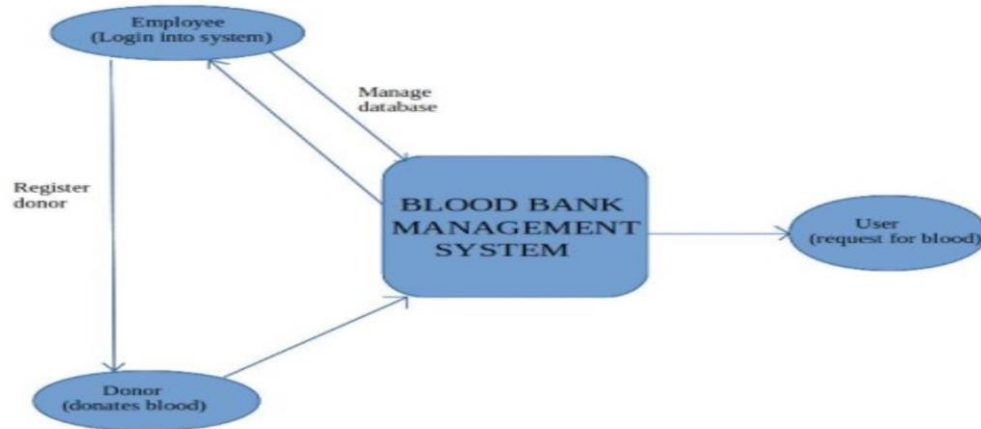
OPERATING SYSTEM: Windows or MAC OS or Linux(32/64bit)

3.2 Contents Diagram

Architectural diagram

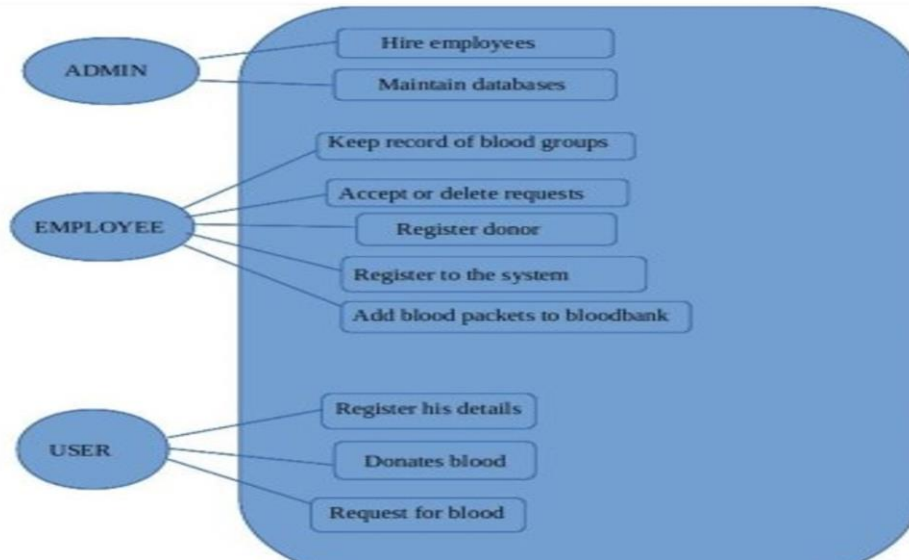


Data flow Diagram

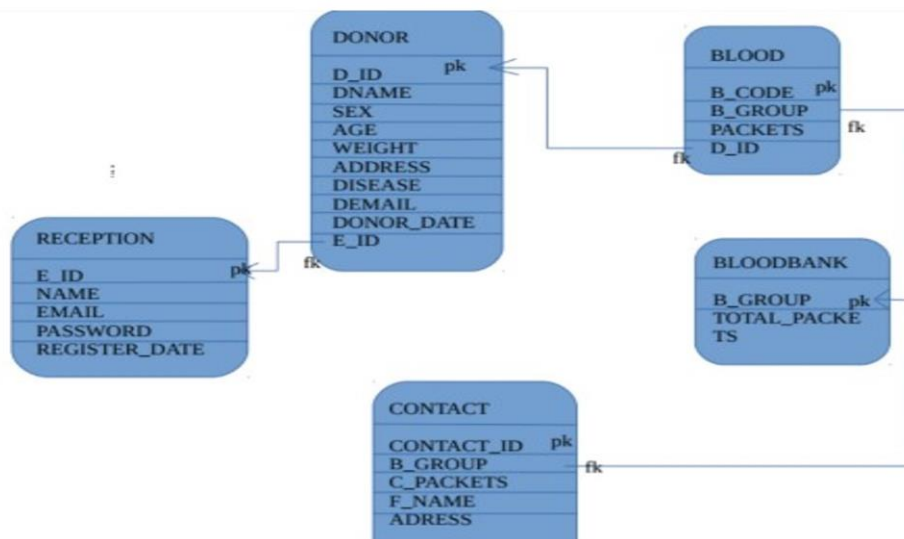


4. DESIGN

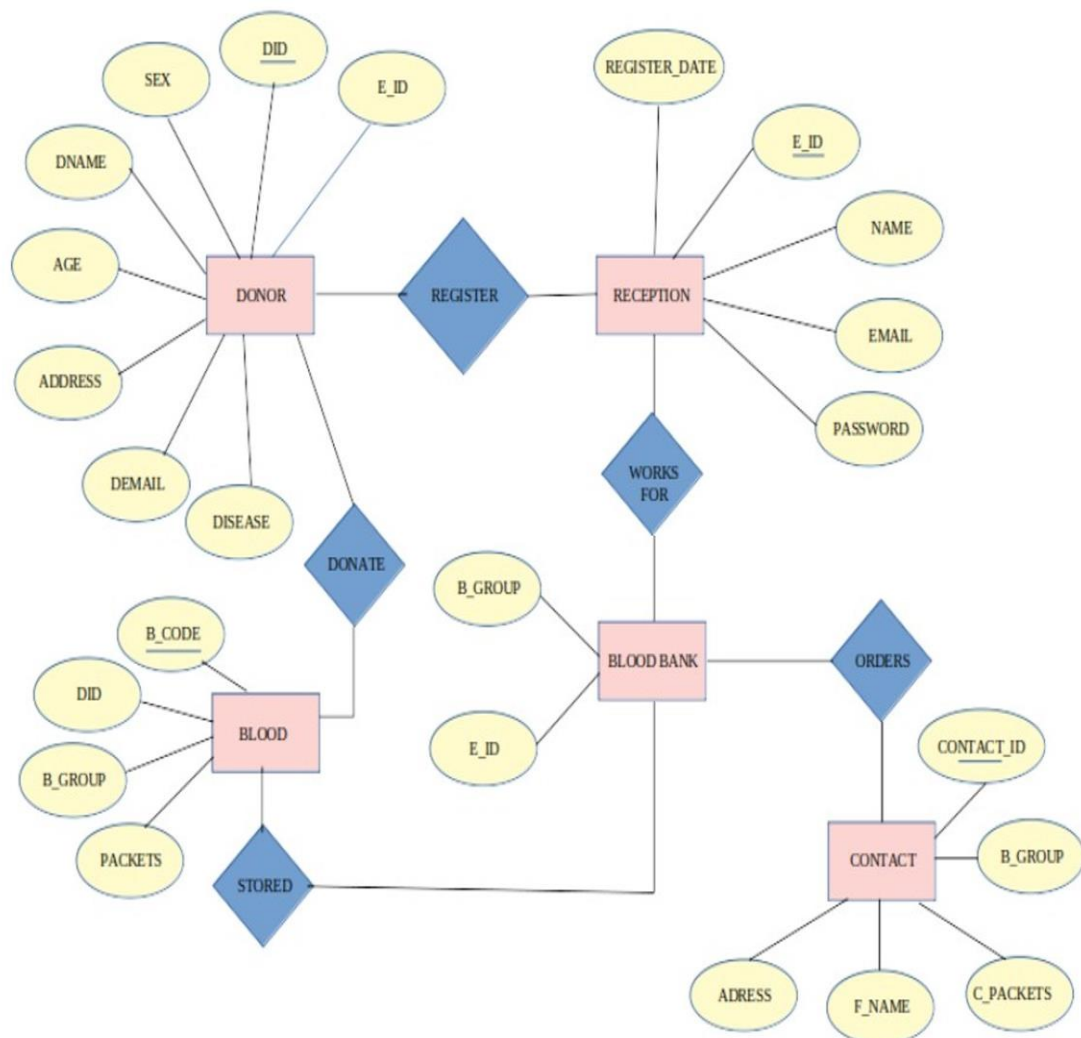
Case Diagram



Schema Diagram



Entity Relationship Diagram



5. IMPLEMENTATION

During the software-testing phase each module of software is thoroughly tested for bugs and for accuracy of output. The system developed is very user-friendly and the detailed documentation is also given to the user as online help wherever necessary. The implementation phase normally ends with the formal test involving all the components.

5.1 Source code

```
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']    pass =
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,
pass) )          con.commit()          msg
= "Record successexcept:
con.rollback()

msg = "error in insert operation"
finally:

    flash('done')

    return redirect(url_for('index'))

con.close()

@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
        type=='pincode':

        con =
        sqlite3.connect('database.db')
```

```
con.row_factory = sqlite3.Row
cur = con.cursor()
    cur.execute("select * from users where
pin=?", (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
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'))
```



NEIL GOGTE INSTITUTE OF TECHNOLOGY

(A Unit of Keshav Memorial Technical Education (KMTES))

approved by AICTE, New Delhi & Affiliated to Osmania University, Hyderabad)

```
#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();
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:
    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,'bn
eg':Bnegativ e,'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 TEXT, phone
TEXT)')  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,phone) )
    con.commit()

        msg = "Record successfully added"
    except:
        con.rollback()        msg
    = "error in insert operation"
    finally:
```

```
        flash("added new entry!")

return redirect(url_for('dashboard'))

con.close() 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,qty,dweight,email,phone,id) )
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:
```

```
name = request.form['name']
addr = request.form['addr']      city =
request.form['city']      pin =
request.form['pin']      bg =
request.form['bg']      emailid =
request.form['email']      id =
request.form['id']      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 = ? ,id = ?",(name,addr,city,pin,bg,emailid,email,id) )
    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,a
ddr) )    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)

    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'))

@app.route('/notifications',methods=('GET','PO
ST')) def notifications():
    if request.method == 'GET':
        conn = sqlite3.connect('database.db')
        print("Opened databas 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'))

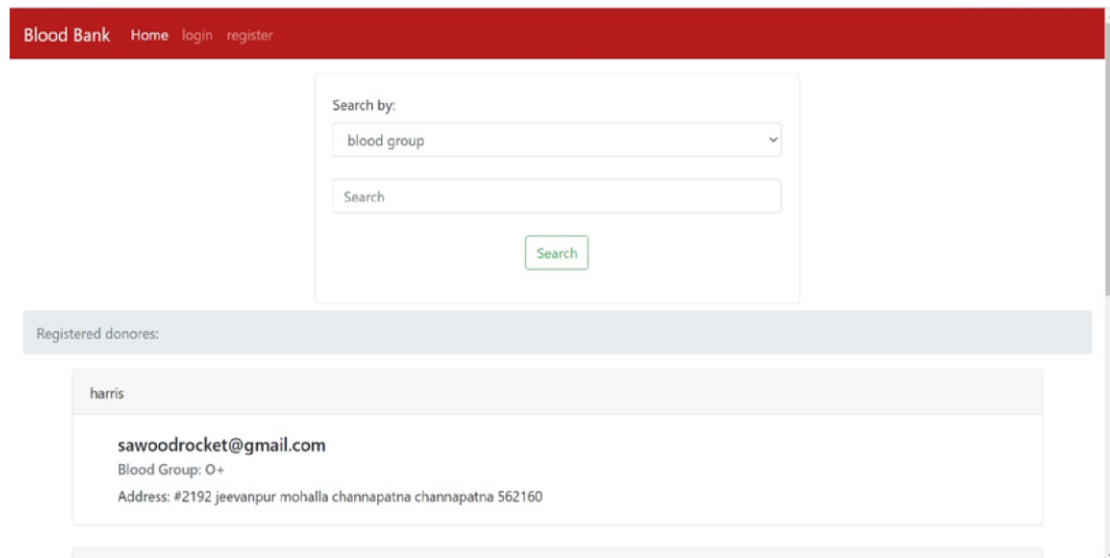
@app.route('/deleteme/<useremail>',methods=('GET',
'POST')) def deleteme(useremail):    if request.method
== 'GET':

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)
```

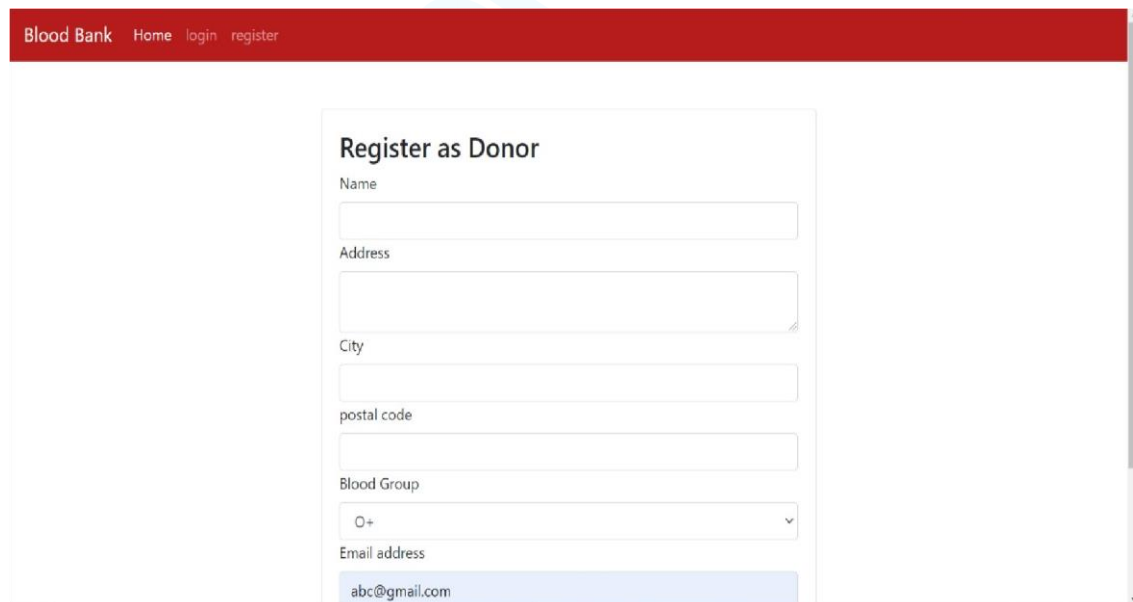
6. TESTING AND VALIDATION

6.1 Project Screenshots



The screenshot shows the home page of a Blood Bank application. At the top is a red navigation bar with links: Blood Bank, Home, login, and register. Below the navigation bar is a search section with a 'Search by:' dropdown menu set to 'blood group', a text input field for the search term, and a green 'Search' button. Underneath the search section is a light blue box labeled 'Registered donores:'. Below this is a list of registered donors, with one entry visible: 'harris'. The entry details include the email 'sawoodrocket@gmail.com', the blood group 'O+', and the address '#2192 jeevanpur mohalla channapatna channapatna 562160'.

Figure 1 home page



The screenshot shows the 'Register as Donor' page. It features a red navigation bar with links: Blood Bank, Home, login, and register. The main content area is titled 'Register as Donor' and contains a form with the following fields: Name, Address, City, postal code, Blood Group (a dropdown menu with 'O+' selected), and Email address. The email address field is filled with 'abc@gmail.com'.

Figure 2 register donor page

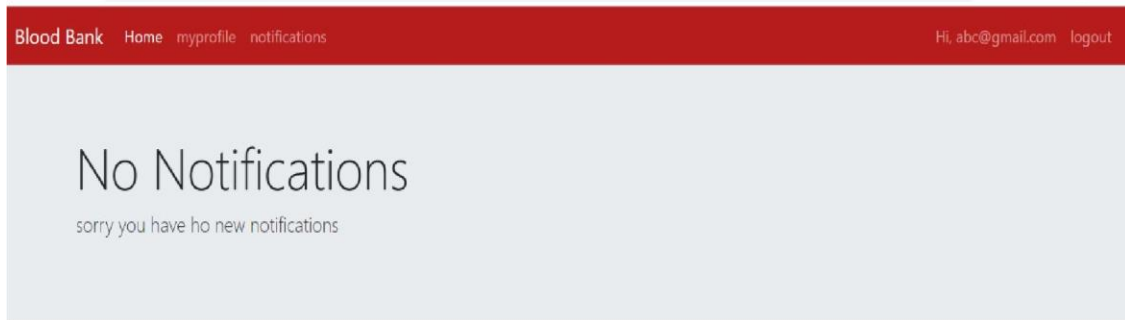
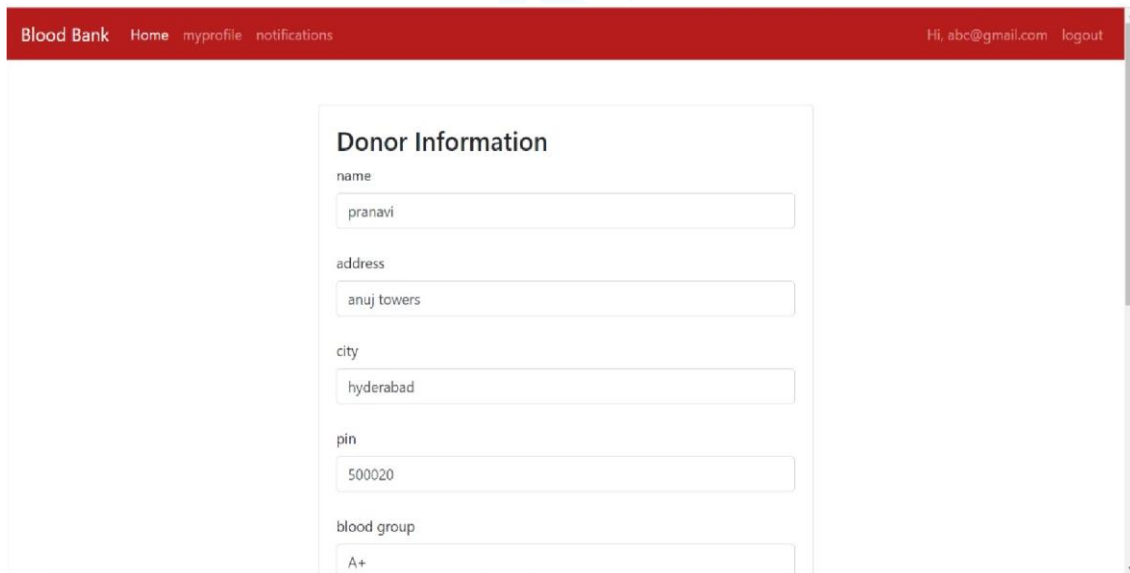


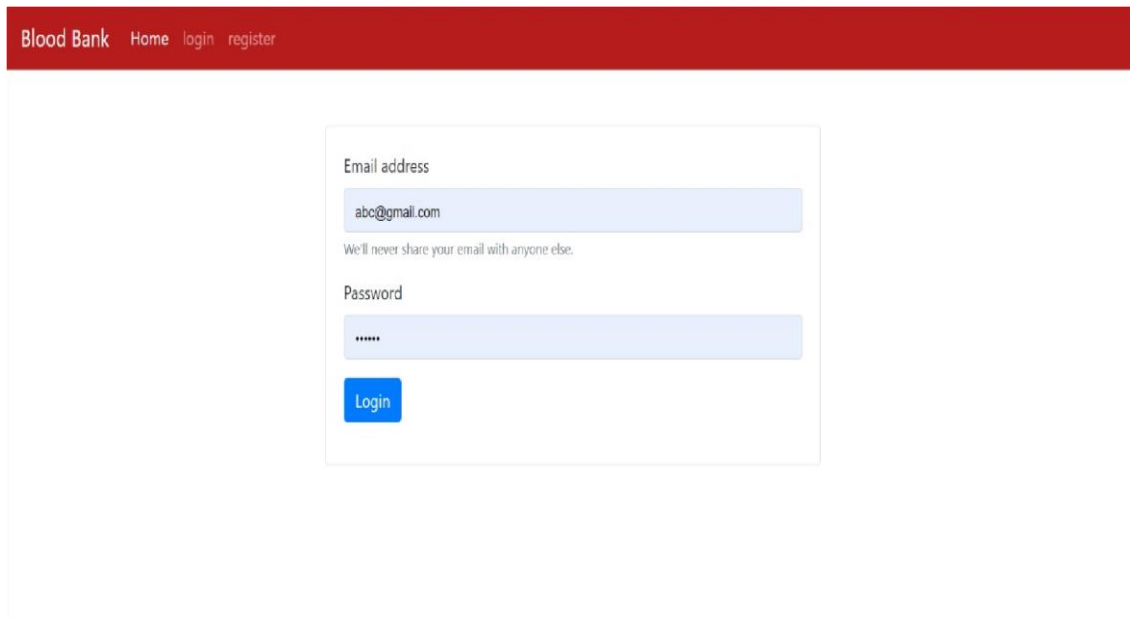
Figure 3 notifications page



The screenshot shows a web application interface with a red header bar. The header contains navigation links: "Blood Bank", "Home", "myprofile", and "notifications". On the right side of the header, it displays "Hi, abc@gmail.com" and a "logout" link. The main content area displays a form titled "Donor Information". The form has the following fields:

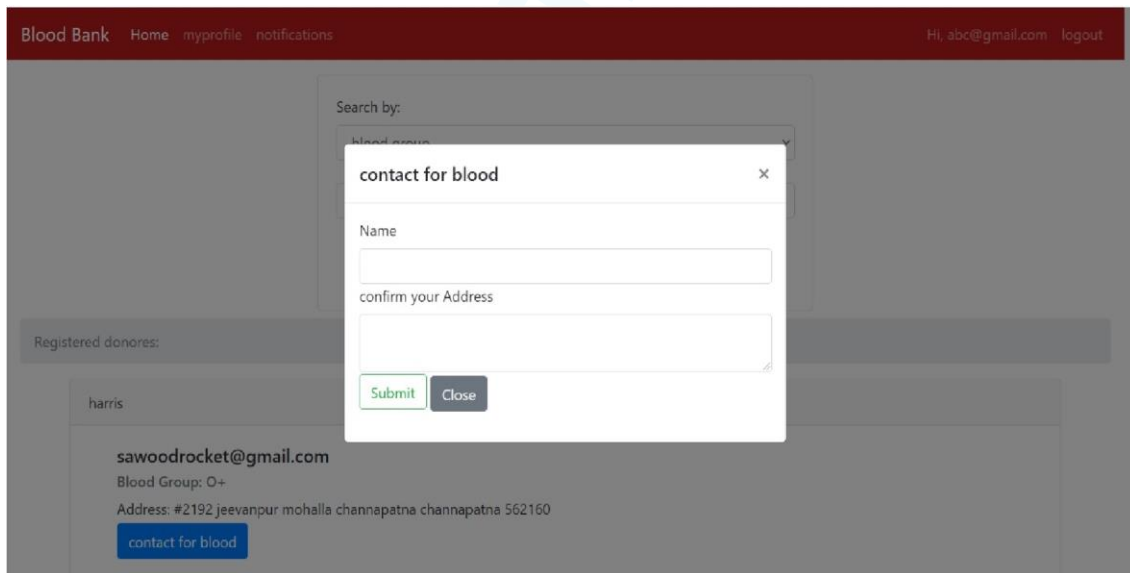
- name:
- address:
- city:
- pin:
- blood group:

Figure 4 donor information



The screenshot shows the login page of the Blood Bank system. At the top, a red navigation bar contains the links "Blood Bank", "Home", "login", and "register". The main content area is white and features a login form. The form has two input fields: "Email address" with the value "abc@gmail.com" and "Password" with masked characters "*****". Below the password field is a blue "Login" button. A small text line between the fields states, "We'll never share your email with anyone else."

Figure 5 login page



The screenshot displays the "contact for blood" form, which is a modal window overlaid on a blurred background of the Blood Bank interface. The modal has a title bar "contact for blood" with a close button (X). It contains two input fields: "Name" and "confirm your Address". At the bottom of the modal are two buttons: "Submit" (green) and "Close" (grey). The background interface shows a navigation bar with "Blood Bank", "Home", "myprofile", and "notifications", and a user greeting "Hi, abc@gmail.com" with a "logout" link. Below the navigation bar, there is a "Registered donors:" section with a list of donors, including "harris" and "sawoodrocket@gmail.com". The donor "sawoodrocket@gmail.com" has a "contact for blood" button next to their details, which include "Blood Group: O+" and "Address: #2192 jeevanpur mohalla channapatna channapatna 562160".

Figure 6 contact for blood

7. CONCLUSION AND FUTURE ENHANCEMENTS

Based on results, this study concluded that the blood bank management system is much better than the manual system. The findings showed that respondents prefer to use online blood bank management system rather than the manual system because it offers many advantages and benefits that lead to its effectiveness, and efficiency. Because of the increased confidence on the users on the system, it can be concluded that the online blood bank management system enhances blood transfusion safety because it provides better ways of handling the various processes in blood bank.

As there was a little number of contact person's information given, some people may face difficulty in getting blood fast. So i like to gather more information regarding the contact persons in other cities as well as villages and will provide much more services for the people and help everyone with humanity.

8. REFERENCES

Python: <https://docs.python.org/3/tutorial/>

Frame work: <https://www.fullstackpython.com/flask.html>

SQLite: <http://www.mysqltutorial.org/>

<https://www.javatpoint.com/mysql-tutorial>

Database: <https://www.coursera.org/learn/python-databases>

<https://code.tutsplus.com/tutorials/creating-a-web-app-from-scratchusing-python-flask-and-MySQL--cms-22972>

Google Chrome: <https://www.google.com/>