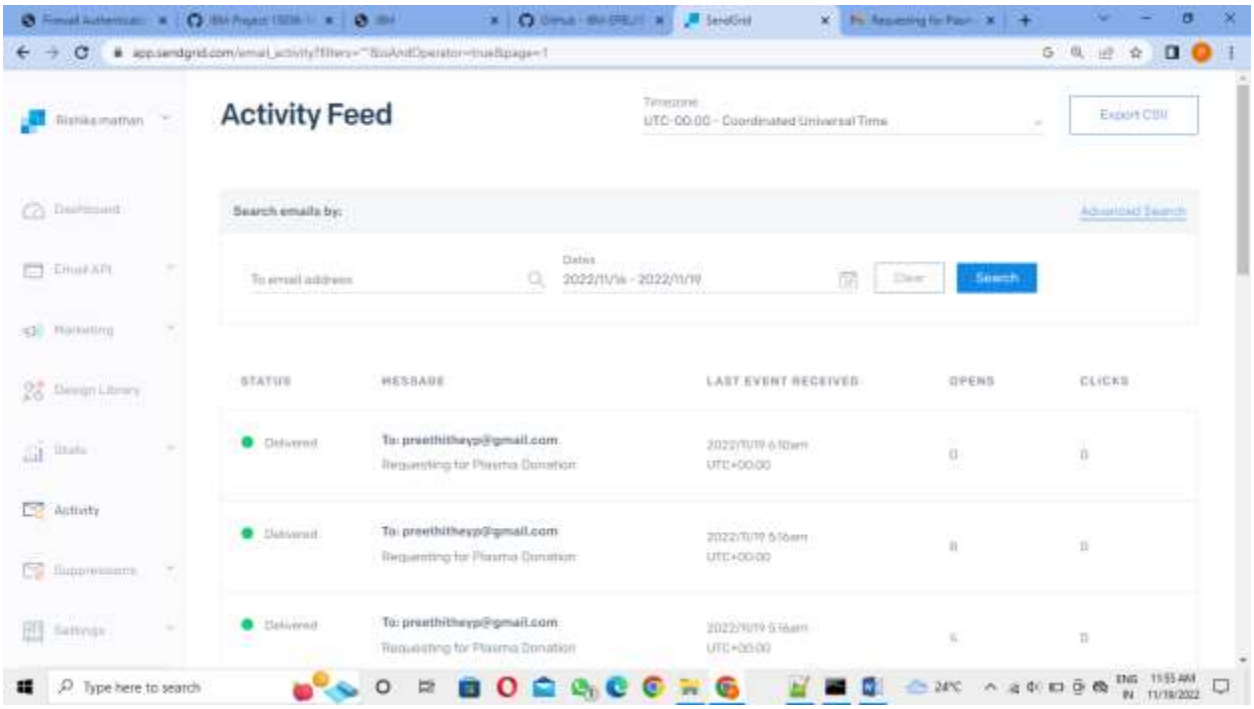
INTEGRATING SENDGRID SERVICE

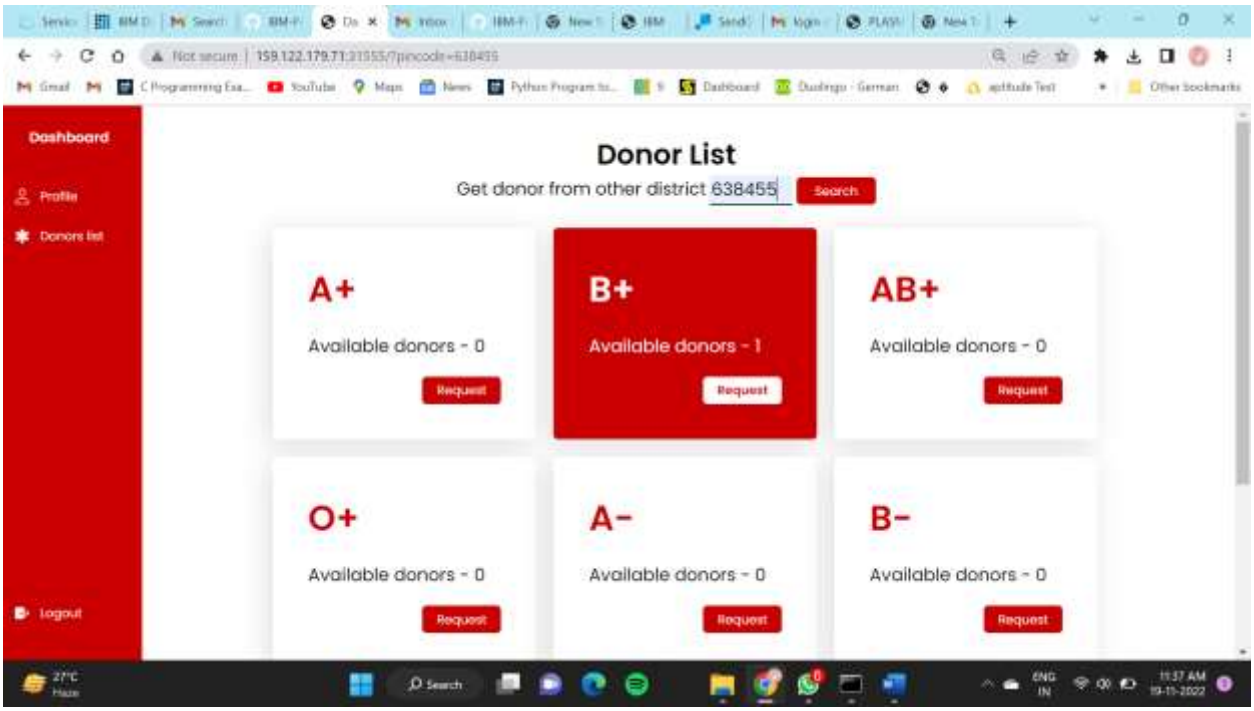
Team ID	**PNT2022TMID04437**
Project Name	**Plasma Donor Application**

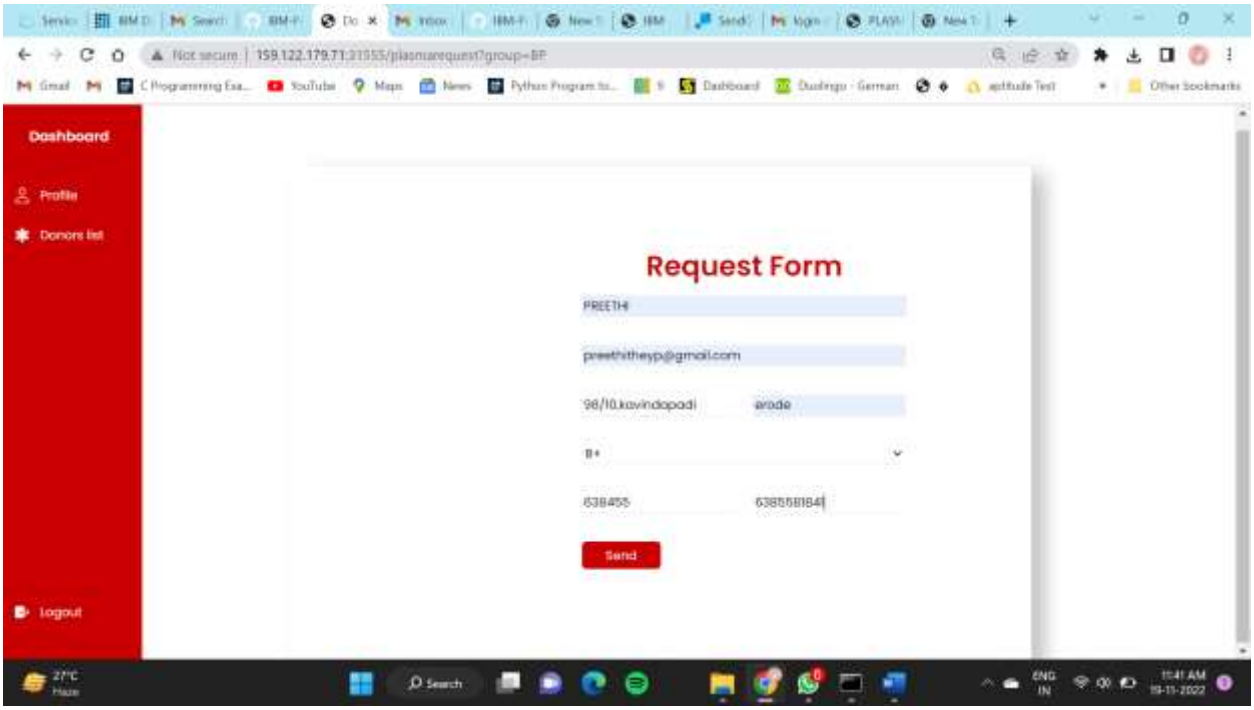
STEP 1: REQUIREMENTS: Python 2.6, 2.7, 3.4 or 3.5.

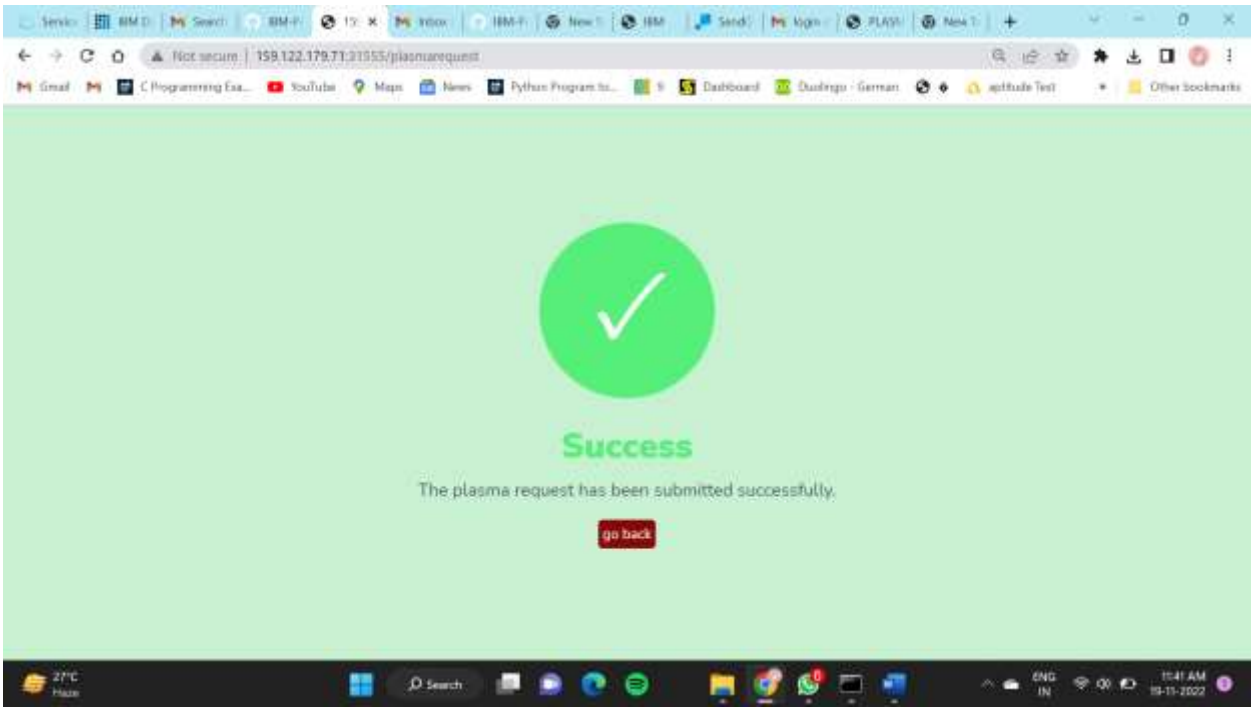
STEP 2: Create an API key



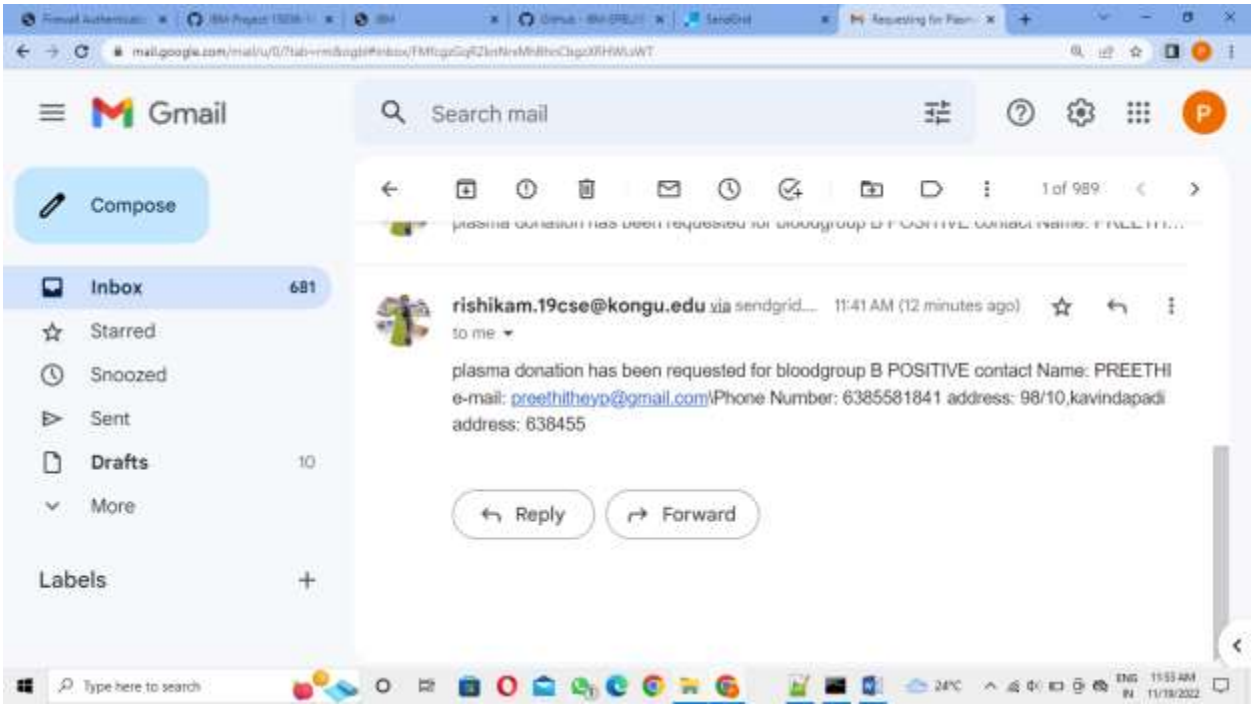
INSTALL PACKAGE: > pip install sendgrid

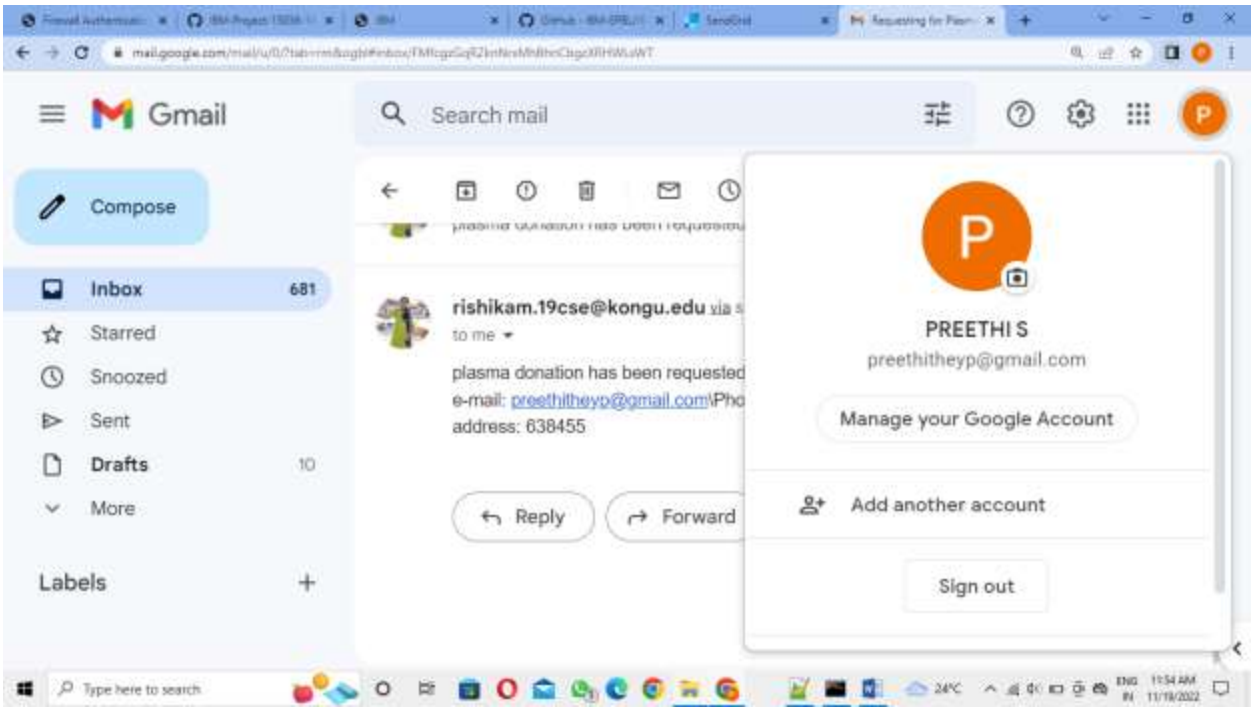

Dashboard
Profile
Donors list
Logout
Donor List
Get donor from other district 638455
Search
A+
Available donors - 0
Request
B+
Available donors - 1
Request
AB+
Available donors - 0
Request
O+
Available donors - 0
Request
A-
Available donors - 0
Request
B-
Available donors - 0
Request


Dashboard
Profile
Donors list
Logout
Request Form
PREETHI
preethitheyp@gmail.com
98/10,kavindapadi
erode
B+
638455
6385581841
Send

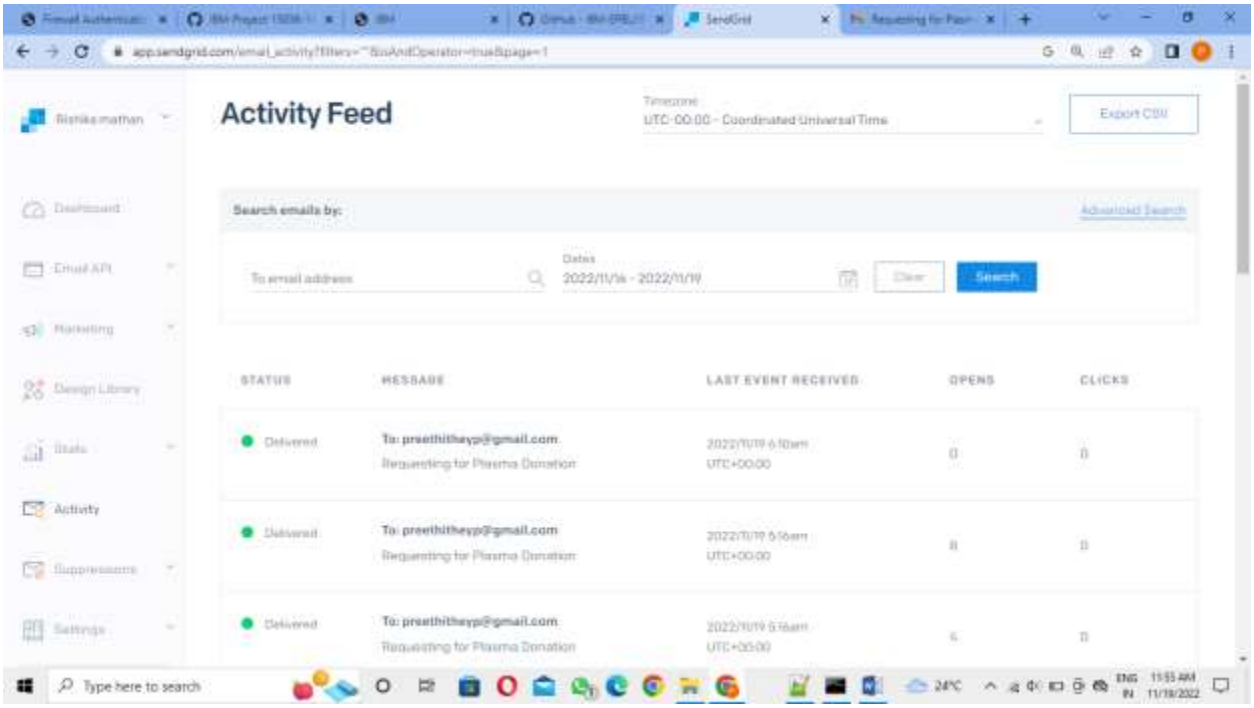

Not secure
159.122.179.71:31555/plasmarequest
Success
The plasma request has been submitted successfully.
go back

SENT MAIL


Gmail
Search mail
Compose
Inbox 681
Starred
Snoozed
Sent
Drafts 10
More
Labels
1 of 989
rishikam.19cse@kongu.edu via sendgrid.... 11:41 AM (12 minutes ago)
to me
plasma donation has been requested for bloodgroup B POSITIVE contact Name: PREETHI e-mail: preethitheyp@gmail.com\Phone Number: 6385581841 address: 98/10,kavindapadi address: 638455
Reply
Forward



SENDGRID STATUS



SENDGRID PYTHON CODE

```
@app.route('/plasmarequest',methods=['GET','POST'])
def requestPlasma():
    if session.get('loggedin')==None:
        return redirect(url_for('.login'))
    if request.method == 'GET':
        return render_template('request.html',blooddata={"status": True,
"data":{"blood":request.args.get('group')}})
    if not (request.form.get('group') and  request.form.get('pincode')
    and request.form.get('address') and request.form.get('email') and request.form.get('name') and
request.form.get('phone')):
        return render_template('requestfail.html')
    contact = {
     "address":request.form.get('address'),
     "email" :request.form.get('email'),
     "name":request.form.get('name'),
     "phone":request.form.get('phone'),
    }
    bloodGroup = request.form.get('group')
    pincode = request.form.get('pincode',type=int)
    print(bloodGroup)
    email_list = getEmail(bloodGroup,pincode,contact)
    print("list"+str(email_list))
    return render_template('requestsuccess.html')
```

HTTP CLIENT PROGRAM

```
import re

from urllib import request

from sendgrid import SendGridAPIClient

from sendgrid.helpers.mail import Mail

import ibm_db

from flask import *

import datetime


sendGrid=SendGridAPIClient('SG.e7AXgb4PQle6LyoDxclRWw.-BtQpG-CZ1AYMxA9GdFvTwuaO5-
eJChkW-VYJ9Rguhc')


plasma_donor_chart = {

   "OP":('OP', 'ON'),

   "ON":('ON'),

   "AP":('OP',  'ON','AP','AN'),

   "AN":('ON','AN'),

   "BP":('OP', 'ON','BP','BN'),

   "BN":('ON','BN'),

   "ABP":('OP',  'ON','AP','AN','BP','BN','ABP','ABN'),

   "ABN":('ON','AN','BN','ABN')

}

group_abbreviation={

   "OP":"O POSITIVE",

   "ON":"O  NEGATIVE",

   "AP":"A  POSITIVE",

   "AN":"A  NEGATIVE",

   "BP":"B POSITIVE",

   "BN":"B  NEGATIVE",

   "ABP":"AB  POSITVE",

   "ABN":"AB  NEGATIVE"
```

```
}
plasma_group_list = ['OP', 'ON','AP','AN','BP','BN','ABP','ABN']


conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=2d46b6b4-cbf6-40eb-bbce-6251e6ba0300.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=32328;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=bmb02607;PWD=9qglrICwkM11bGTw;", '', '')




app = Flask(__name__)
app.secret_key = "uafabdfouabdoibaidnaou"
def sendMail(to_email,subject,content):
    message = Mail(
    from_email='rishikam.19cse@kongu.edu',
    to_emails=to_email,
    subject=subject,
    html_content=content)
    try:
        response = sendGrid.send(message)
        print(response.status_code)
        # print(response.body)
        # print(response.headers)
    except Exception as e:
        print(e.message)


def getCount(pincode):
    count_list ={}
    for group in plasma_group_list:
        count_query = 'select count(*) from userdata where bloodgroup = ? and pincode between ? and ?'
        fetch_count = ibm_db.prepare(conn,count_query)
        # print(group)
```

```
        ibm_db.bind_param(fetch_count, 1,group)
        ibm_db.bind_param(fetch_count, 2,int((int(pincode/1000))*1000))
        ibm_db.bind_param(fetch_count, 3,int((int(pincode/1000))*1000)+1000)
        ibm_db.execute(fetch_count)
        count = ibm_db.fetch_assoc(fetch_count)
        count_list[group] = count['1']
    print(count_list)
    return count_list

def getEmail(group,pincode,contact):
    count_query = 'select email from userdata where bloodgroup = ? and pincode between ? and ?'
    fetch_count = ibm_db.prepare(conn,count_query)
    print(group,pincode,contact)
    ibm_db.bind_param(fetch_count, 1,group)
    ibm_db.bind_param(fetch_count, 2,int((int(pincode/1000))*1000))
    ibm_db.bind_param(fetch_count, 3,int((int(pincode/1000))*1000)+1000)
    ibm_db.execute(fetch_count)
    email = ibm_db.fetch_assoc(fetch_count)
    while email!= False:
        mailBody = "plasma donation has been requested for bloodgroup "+group_abbreviation[group]+ "\ncontact\nName: "+ contact['name'] + "\ne-mail: "+contact['email']+ "\Phone Number: "+contact['phone']+ "\naddress: "+contact['address']+ "\naddress: "+str(pincode)
        sendMail(email['EMAIL'],"Requesting for Plasma Donation",mailBody)
        print(email)
        email = ibm_db.fetch_assoc(fetch_count)
    return email

def getUserData(email):
    sql = "SELECT * FROM userdata WHERE email = ?"
    userdatastatement = ibm_db.prepare(conn, sql)
    ibm_db.bind_param(userdatastatement, 1, email)
```

```
    ibm_db.execute(userdatastatement)
    user_details = ibm_db.fetch_assoc(userdatastatement)
    return user_details



@app.route("/login", methods=['POST',  'GET'])
def login():
    print(session.get('loggedin'))
    if session.get('loggedin')!=None:
        return redirect(url_for('.dash'))
    if request.method  == "POST":
        email = request.form['email']
        password = request.form['password']
        print(email)
        sql =  "SELECT email FROM Users WHERE email = ? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print("acc "+str(account))
        if account:
            print('login')
            session['loggedin'] = True
            session['id'] = account['EMAIL']
            sendMail(email,"login detected","login detected on some random device with ip ")
            return redirect(url_for('.dash'))
        else :
            return render_template('login.html',message="user  not found or invalid credentials")
    else:
```

```
    return  render_template('login.html')



@app.route('/register', methods=['GET',  'POST'])
def register():
   if request.method  == 'POST':
      name=request.form.get('name')
      bloodGroup = request.form.get('group')
      pincode = request.form.get('pincode')
      print(name)
      email=request.form.get('email')
      password=request.form.get('password')
      date = request.form.get('lastdonated')
      dateSplit = date.split("-")
      lastdonated = datetime.datetime(int(dateSplit[0]), int(dateSplit[1]), int(dateSplit[2]), 0, 0, 0).timestamp()
      lastdonated = int(lastdonated)
      sql = "SELECT * FROM users WHERE email = ?"
      stmt = ibm_db.prepare(conn, sql)
      ibm_db.bind_param(stmt, 1, email)
      ibm_db.execute(stmt)
      account=ibm_db.fetch_assoc(stmt)
      print(account)
      if account:
         msg="Account already exists!"
         return render_template('register.html',resp=msg)
      if not re.match(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b',str(email)): #1234@gmail.com
         msg = "Invalid email address" + str(email)
         return render_template('register.html',resp=msg)
      if not (request.form.get('group') and  request.form.get('pincode') and request.form.get('lastdonated') and request.form.get('email') and request.form.get('name') and request.form.get('password')):
         msg = "fill all the fields"
```

```
            return render_template('register.html',resp=msg)
        else:
            try:
                insert_user_table="INSERT INTO users VALUES ( ?, ?)"
                user_create = ibm_db.prepare(conn,insert_user_table)
                ibm_db.bind_param(user_create, 1, email)
                ibm_db.bind_param(user_create, 2, password)
                ibm_db.execute(user_create)
                insert_userdata_table = "INSERT INTO USERDATA VALUES (?,?,?,?,?)"
                user_data_create = ibm_db.prepare(conn,insert_userdata_table)
                print(email,name,bloodGroup,pincode,lastdonated)
                ibm_db.bind_param(user_data_create, 1, email)
                ibm_db.bind_param(user_data_create, 2, name)
                ibm_db.bind_param(user_data_create, 3, bloodGroup)
                ibm_db.bind_param(user_data_create, 4, int(pincode))
                ibm_db.bind_param(user_data_create, 5, int(lastdonated))
                ibm_db.execute(user_data_create)
                msg="You have successfully registered"
                # sendMail(email,"registered successfully",msg)
                print(msg)
            except:
                render_template('register.html',msg="error")
            return redirect(url_for('.login'))

    return render_template('register.html')



@app.route('/',methods=['GET'])
def dash():
    if session.get('loggedin')==None:
```

```
        return redirect(url_for('.login'))


    userData= getUserData(session['id'])
    userData['PINCODE']=request.args.get('pincode',type=int) if request.args.get('pincode',type=int) else userData['PINCODE']
    count = getCount(userData['PINCODE'])
    return render_template('homepage.html',data={'status':True,"user":userData,"count":count})
    # except:
    #     print("error")
    #     return render_template('.html',data={'status':False})


# @app.route('/request/<blood>', methods=['GET'])
# def requestForm(blood):
#     return render_template('request.html',})


@app.route('/plasmarequest',methods=['GET','POST'])
def requestPlasma():
    if session.get('loggedin')==None:
        return redirect(url_for('.login'))
    if request.method == 'GET':
        return render_template('request.html',blooddata={"status": True, "data":{"blood":request.args.get('group')}})
    if not (request.form.get('group') and request.form.get('pincode')
    and request.form.get('address') and request.form.get('email') and request.form.get('name') and request.form.get('phone')):
        return render_template('requestfail.html')
    contact = {
     "address":request.form.get('address'),
     "email" :request.form.get('email'),
     "name":request.form.get('name'),
     "phone":request.form.get('phone'),
    }
```

```
    bloodGroup = request.form.get('group')
    pincode = request.form.get('pincode',type=int)
    print(bloodGroup)
    email_list = getEmail(bloodGroup,pincode,contact)
    print("list"+str(email_list))
    return render_template('requestsuccess.html')

@app.route('/profile',methods=['GET'])
def profile():
    if session.get('loggedin')==None:
        return redirect(url_for('.login'))
    try:
        user_data = getUserData(session.get('id'))
        return render_template('profile.html',resp=user_data)
    except:
        return render_template('requestfail.html')
@app.route('/logout')

def logout():
   session.pop('loggedin', None)
   session.pop('id', None)
   return redirect(url_for('.login'))



if __name__ == "__main__":
    app.run('0.0.0.0',port=5001)
```