BLOOD BANK

MANAGEMENT SYSTEM

BATCH

2023-24

COMPUTER SCIENCE

S HARIPRIYA
XII - F1

Chettinad Vidyashram

(Affiliated to Central Board of Secondary Education, New Delhi)
(Chettinad House, R.A.Puram, Chennai – 600 028)

COMPUTER SCIENCE

Certified to be the Bonafide Record of work done by

_____ of Std XII Sec ____

in the Computer Science Lab of the CHETTINAD VIDYASHRAM, CHENNAI, during the year 2023 – 2024.

Date: Teacher-in-charge

REGISTER NO. _____

Submitted for All India Senior Secondary Practical Examination in Computer Science held on _____at Chettinad Vidyashram, Chennai – 600 028.

Principal Internal Examiner External Examiner

ACKNOWLEDGEMENT

I would like to express my sincere thanks to Meena Aunty, Principal Mrs. S.Amudhalakshmi for their encouragement and support to work on this Project. I am grateful to my computer science teacher Mrs. V. MALA and to the computer science department for the constant guidance and support to complete the project.

TABLE OF CONTENTS

INDEX

OVERVIEW OF PYTHON

PYTHON IS A HIGH-LEVEL, INTERPRETED, INTERACTIVE AND OBJECT-ORIENTED SCRIPTING LANGUAGE. PYTHON IS DESIGNED TO BE HIGHLY READABLE. IT USES ENGLISH KEYWORDS FREQUENTLY WHERE AS OTHER LANGUAGES USE PUNCTUATION, AND IT HAS FEWER SYNTACTICAL CONSTRUCTIONS THAN OTHER LANGUAGES.

- **PYTHON IS INTERPRETED** − PYTHON IS PROCESSED AT RUNTIME BY THE INTERPRETER.
- **PYTHON IS INTERACTIVE** − YOU CAN ACTUALLY SIT AT A PYTHON PROMPT AND INTERACT WITH THE INTERPRETER DIRECTLY TO WRITE YOUR PROGRAMS.
- **PYTHON IS OBJECT-ORIENTED** − PYTHON SUPPORTS OBJECT-ORIENTED STYLE.
- **PYTHON IS A BEGINNER'S LANGUAGE** − PYTHON IS A GREAT LANGUAGE FOR THE BEGINNER-LEVEL PROGRAMMERS .

PYTHON IS AN OPEN-SOURCE AND CROSS-PLATFORM PROGRAMMING LANGUAGE. IT IS AVAILABLE FOR USE UNDER **PYTHON SOFTWARE FOUNDATION LICENSE** (COMPATIBLE TO GNU GENERAL PUBLIC LICENSE) ON ALL THE MAJOR OPERATING SYSTEM PLATFORMS LINUX, WINDOWS AND MAC OS.

PROJECT DESCRIPTION

THIS PROJECT IS A POTENTIAL AID FOR A PERSON MANAGING A BLOOD DONATION CAMP. IT FACILITATES EASY ACCESS, MANIPULATION AND STORAGE OF DATA AND RECORDS PERTAINING TO BLOOD DONATION/RECEPTION ACTIVITIES.

IT COMES WITH 5 MENUS:

1. DONATION RECORDS- allows the user to keep in track of the data of the donor including the name, blood group, number of units donated, mobile number, date and time.

2.RECEPTION RECORDS-allows the user to keep in track of the data of the donor including the name, blood group, number of units donated, mobile number, date and time.

3. BLOOD BANK- provides an insight on the number of units available for each blood group in the blood bank.

4. NEW DONATION- gets information about the new donation taking place including the donor name, blood group, number of units donated, mobile number and make necessary changes in the blood bank data and make an entry in the donation record **automatically.**

5.NEW RECEPTION- gets information about the new donation taking place including the donor name, blood group, number of units donated, mobile number and make necessary changes in the blood bank data and make an entry in the donation record automatically.

In addition to that, it'll show the number of units available for each matching blood group as per requirement in the form of a table based on red blood cell compatibility.

Blood Type Compatibility

Blood Type	Gives	Receives
A+	A+, AB+	A+, A-, O+, O-
O+	O+, A+, B+, AB+	O+, O-
B+	B+, AB+	B+, B-, O+, O-
AB+	AB+	Everyone
A-	A+, A-, AB+, AB-	A-, O-
O-	Everyone	O-
B-	B+, B-, AB+, AB-	B-, O-
AB-	AB+, AB-	AB-, A-, B-, O-

USER DEFINED FUNCTIONS USED

1. insert_d():- this udf is used for inserting the donor's data entered by the user on the tkinter window into the mysql table for donation records and also update the number of units available in the blood bank.

2.insert_r():-this udf is used for inserting the receiver's data entered by the user on the tkinter window into the mysql table for reception records and also update the number of units available in the blood bank.

3.retrieval():- this udf is used to map the compatible blood groups by taking in the blood group of the receiver and then provide those groups along with the number of units available in a tabular form.

4.dr():- this udf is used to access all the donation records from the corresponding mysql table and present it on a tkinter window.

5.rr():- this udf is used to access all the reception records from the corresponding mysql table and present it on a tkinter window.

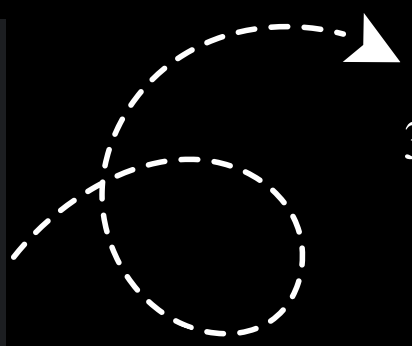
6.nd(): this udf is used to set up a tkinter window that gets donor information and stores it into variables which are in turn passed into insert_d() function for further manipulation.

7.nr():-this udf is used to set up a tkinter window that gets receiver information and stores it into variables which are in turn passed into insert_r() function for further manipulation.

8. ua():-this udf is used to access the bloodbank from mysql and display it in a tkinter window.

TABLES USED FOR PYTHON-SQL CONNECTIVITY PROGRAM

```
mysql> show tables;
+-------------------------+
| Tables_in_blooddonation |
+-------------------------+
| bloodbank               |
| donation                |
| reception               |
+-------------------------+
3 rows in set (0.00 sec)
```



3 TABLES

1. bloodbank

```
mysql> desc bloodbank;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| BloodType | char(11) | NO   | PRI | NULL    |       |
| Units     | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
2 rows in set (0.03 sec)
```

HOLDS THE BLOOD GROUPS AND THE NUMBER OF UNITS AVAILABLE FOR EACH BLOOD GROUP.

2.donation

```
mysql> desc donation;
+-----------+---------------+------+-----+-------------------+-------+
| Field     | Type          | Null | Key | Default           | Extra |
+-----------+---------------+------+-----+-------------------+-------+
| SNo       | int(11)       | NO   | PRI | NULL              |       |
| DonorName | char(20)      | YES  |     | NULL              |       |
| BloodType | char(11)      | YES  |     | NULL              |       |
| Units     | int(11)       | YES  |     | NULL              |       |
| MobileNo  | decimal(10,0) | YES  |     | NULL              |       |
| DateTime  | timestamp     | NO   |     | CURRENT_TIMESTAMP |       |
+-----------+---------------+------+-----+-------------------+-------+
6 rows in set (0.00 sec)
```

MAINTAINS THE DONOR RECORDS

3.reception

```
mysql> desc reception;
+-----------+---------------+------+-----+-------------------+-------+
| Field     | Type          | Null | Key | Default           | Extra |
+-----------+---------------+------+-----+-------------------+-------+
| SNo       | int(11)       | NO   | PRI | NULL              |       |
| DonorName | char(20)      | YES  |     | NULL              |       |
| BloodType | char(11)      | YES  |     | NULL              |       |
| Units     | int(11)       | YES  |     | NULL              |       |
| MobileNo  | decimal(10,0) | YES  |     | NULL              |       |
| DateTime  | timestamp     | NO   |     | CURRENT_TIMESTAMP |       |
+-----------+---------------+------+-----+-------------------+-------+
6 rows in set (0.01 sec)
```

MAINTAINS THE RECEPIENT RECORDS

SOURCE CODE

```
import mysql.connector as h
import tkinter  as tk
import tkinter.messagebox as mb
from tkinter import *
from tkinter import ttk

def insert_d(b_dname,b_dbg,b_dunits,b_dsno,b_dmno):
  global x
  dname=b_dname.get()
  dbg=b_dbg.get()
  dsno=b_dsno.get()
  dmno=b_dmno.get()
  global dunits
  dunits=b_dunits.get()
  if dname=='' or dbg=='' or dunits=='' or dmno==''or dsno='':
      print('Fill all fields')
  else:
      cur.execute("insert into donation
values("+dsno+",'"+dname+"','"+dbg+"',"+dunits+","+dmno+",DEFAUL
T)")
      cur.execute("select Units from bloodbank where
BloodType='"+dbg+"'")
      x=cur.fetchall()
      y=int(x[0][0])
      y+= int(dunits)
      cur.execute("update bloodbank set Units="+str(y)+" where
BloodType='"+dbg+"'")
      cur.execute('commit')
      mb.showinfo("Insert status","Inserted into records")
      b_dname.delete(0,'end')
      b_dbg.delete(0,'end')
      b_dunits.delete(0,'end')
      b_dsno.delete(0,'end')
```

```
        b_dmno.delete(0,'end')

def insert_r(b_rname,b_rbg,b_runits,b_rsno,b_rmno):
    global rbg
    global runits
    rname=b_rname.get()
    rbg=b_rbg.get()
    runits=b_runits.get()
    rsno=b_rsno.get()
    rmno=b_rmno.get()
    if rname=='' or rbg==''  or runits=='' or rsno=='' or rmno=='' :
        print('Fill all fields')
    else:
        cur.execute("insert into reception values("+rsno+",'" + rname + "','" +
rbg + "'," + runits + ","+rmno+",DEFAULT)")
        cur.execute("select Units from bloodbank where BloodType='" + rbg +
"'")
        x=cur.fetchall()
        y=int(x[0][0])
        y -= int(runits)
        cur.execute("update bloodbank set Units=" + str(y) + " where
BloodType='" + rbg + "'")
        cur.execute('commit')
        mb.showinfo("Insert status","Inserted into records")
        b_rname.delete(0,'end')
        b_rbg.delete(0,'end')
        b_runits.delete(0,'end')
        b_rsno.delete(0,'end')
        b_rmno.delete(0,'end')
def retrieval(b_rbg):
    global rbg
    rbg=b_rbg.get()
    global s
    ss = Toplevel ()
    ss.geometry("400x250")
    ss.configure(bg='firebrick')
```

```
  ss.title("                                    AVAILABILITY")
  e = Label(ss, width=20, text='BLOOD TYPE', borderwidth=2,
relief='ridge', anchor='w', bg='brown1')
  e.grid(row=0, column=0)
  e = Label(ss, width=20, text='UNITS', borderwidth=2, relief='ridge',
anchor='w', bg='brown1')
  e.grid(row=0, column=1)
  if rbg=='Apos':
            s=['Apos','Aneg','Opos','Oneg']
  elif rbg=='Aneg':
            s=['Aneg','Oneg']
  elif rbg=='Bpos':
            s=['Bpos','Bneg','Opos','Oneg']

  elif  rbg=='Bneg':
            s=['Bneg','Oneg']

  elif rbg=='Opos':
            s=['Opos','Oneg']

  elif rbg=='Oneg':
            s=['Oneg']

  elif rbg=='ABpos':

            s=['Apos','Aneg','Bpos','Bneg','Opos','Oneg','ABpos','ABneg']
  elif rbg=='ABneg':
            s=['Aneg','Bneg','Oneg','ABneg']


  for j in range(len(s)):
          cur.execute("select*from bloodbank where BloodType='"+s[j]+"'")
          v=cur.fetchall()
          for k in v:
           for i in range(len(k)):
             e = Entry(ss, width=20, fg='black')
```

```
                e.grid(row=j+1, column=i)
                e.insert(tk.END,k[i])
con=h.connect(host='localhost',user='root',passwd='hari',port='3306',data
base='blooddonation')
cur=con.cursor()
cur.execute("use blooddonation")
def dr():
#VIEW DONOR'S LIST
ye= Toplevel()
ye.geometry("1000x600")
ye.configure(bg='grey1')
bgx = PhotoImage(file=r"C:\Users\Haripriya\Downloads\img2.png")
# Show image using label
label1 = Label(ye, image=bgx)
label1.place(x=1000, y=300)
ye.title("                                    DONATION RECORDS")
e=Label(ye,width=20,text='DONOR NO',borderwidth=2,
relief='ridge',anchor='w',bg='snow')
e.grid(row=0,column=0)
e=Label(ye,width=20,text='DONOR NAME',borderwidth=2,
relief='ridge',anchor='w',bg='snow')
e.grid(row=0,column=1)
e=Label(ye,width=20,text='BLOOD TYPE',borderwidth=2,
relief='ridge',anchor='w',bg='snow')
e.grid(row=0,column=2)
e=Label(ye,width=20,text='UNITS',borderwidth=2,
relief='ridge',anchor='w',bg='snow')
e.grid(row=0,column=3)
e=Label(ye,width=20,text='MOBILE NO.',borderwidth=2,
relief='ridge',anchor='w',bg='snow')
e.grid(row=0,column=4)
e = Label(ye, width=20, text='DATE & TIME', borderwidth=2,
relief='ridge', anchor='w', bg='snow')
e.grid(row=0, column=5)

cur.execute("SELECT * FROM donation")
```

```
i=1
for k in cur:#cur has the records as tuple of tuples
  for j in range(len(k)):# how many tuples in that list i.e. how many
records/rows
      e = Entry(ye, width=20, fg='black')
      e.grid(row=i+1, column=j)
      e.insert(tk.END, k[j])

  i=i+1

ye.mainloop()
#VIEW RECEPIENT'S LIST
def rr():
  ye2 = Toplevel()
  ye2.geometry("1000x600")
  ye2.configure(bg="grey1")
  ye2.title('                                    RECEPTION RECORDS')
  e = Label(ye2, width=20, text='RECEPIENT NO.', borderwidth=2,
relief='ridge', anchor='w', bg='snow')
  e.grid(row=0, column=0)
  e = Label(ye2, width=20, text='RECEPIENT NAME', borderwidth=2,
relief='ridge', anchor='w', bg='snow')
  e.grid(row=0, column=1)
  e = Label(ye2, width=20, text='BLOOD TYPE', borderwidth=2,
relief='ridge', anchor='w', bg='snow')
  e.grid(row=0, column=2)
  e = Label(ye2, width=20, text='UNITS', borderwidth=2, relief='ridge',
anchor='w', bg='snow')
  e.grid(row=0, column=3)
  e = Label(ye2, width=20, text='MOBILE NO.', borderwidth=2,
relief='ridge', anchor='w', bg='snow')
  e.grid(row=0, column=4)
  e = Label(ye2, width=20, text='DATE & TIME', borderwidth=2,
relief='ridge', anchor='w', bg='snow')
  e.grid(row=0, column=5)
  p2 = 1
```

```
  cur.execute("SELECT * FROM reception")
  i = 1
  for k in cur:  # cur has the records as a list of tuples
      for j in range(len(k)):  # how many tuples in that list i.e. how many
records/rows
          e = Entry(ye2, width=20, fg='black')
          e.grid(row=i+1, column=j)
          e.insert(tk.END, k[j])
      i = i + 1
  ye2.mainloop()
#NEW DONATION
def nd():
  cur.execute('select count(*) from donation')
  f = cur.fetchall()
  ye3=Toplevel()
  ye3.geometry('1500x1000')
  l_dname=Label(ye3,text='ENTER NAME OF
DONOR:',font=('bold','30'))
  l_dname.place(x=20,y=50)
  l_dbg = Label(ye3, text='ENTER BLOOD GROUP OF DONOR:',
font=('bold', '30'))
  l_dbg.place(x=20, y=100)
  l_dunit= Label(ye3,text='ENTER NO. OF UNITS
DONATED:',font=('bold','30'))
  l_dunit.place(x=20,y=150)
  l_dsno = Label(ye3, text='ENTER DONOR NO.:', font=('bold', '30'))
  l_dsno.place(x=20, y=200)
  l_dmno = Label(ye3, text='ENTER MOBILE NO.:', font=('bold', '30'))
  l_dmno.place(x=20, y=250)
  l=Label(ye3,text='NO. OF DONATIONS DONE:',font=('bold','25'))
  l.place(x=1000,y=200)
  l1 = Label(ye3, text=f[0][0], font=('bold', '25'))
  l1.place(x=1000, y=250)
  b_dname=Entry(ye3 )
  b_dname.place(x=750,y=60)
  choices=['ABneg','ABpos','Aneg','Apos','Bneg','Bpos','Oneg','Opos']
```

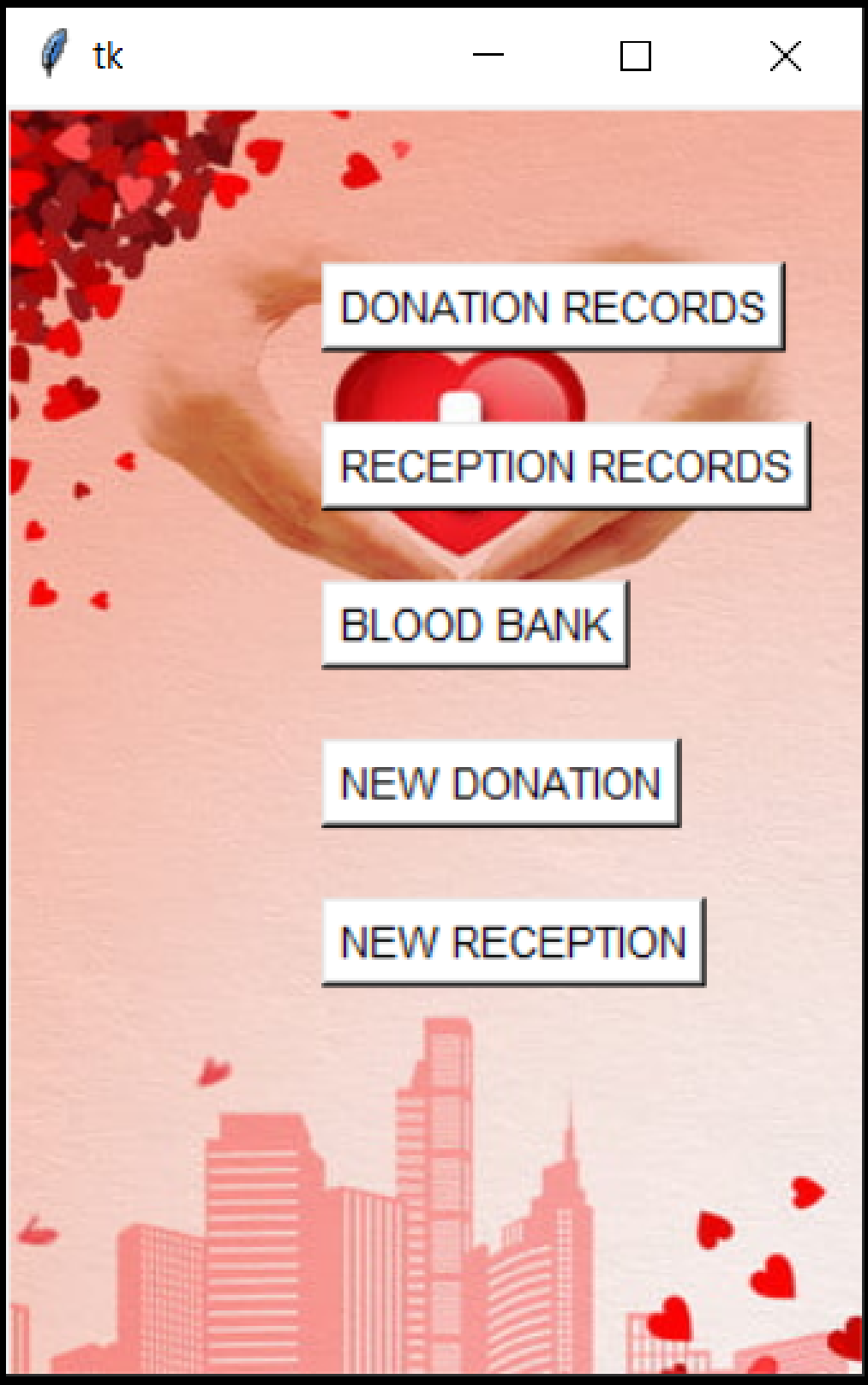
```
  b_dbg=ttk.Combobox(ye3,values=choices)
  b_dbg.place(x=750,y=110)
  b_dunits=Entry(ye3)
  b_dunits.place(x=750,y=160)
  b_dsno=Entry(ye3)
  b_dsno.place(x=750,y=210)
  b_dmno = Entry(ye3)
  b_dmno.place(x=750, y=260)

dsubmit=Button(ye3,text='SUBMIT',font=('italic',50),bg='white',command
=lambda: insert_d(b_dname,b_dbg,b_dunits,b_dsno,b_dmno))
  dsubmit.place(x=400,y=600)
  ye3.mainloop()
#new reception
def nr():
  cur.execute('select count(*) from reception')
  f=cur.fetchall()
  cur.execute('set @n1=@n1+1')
  ye4 = Toplevel ()
  ye4.geometry('1500x1000')
  l_rname = Label(ye4, text='ENTER NAME OF RECIPIENT:',
font=('bold', '30'))
  l_rname.place(x=20, y=50)
  l_rbg = Label(ye4, text='ENTER BLOOD GROUP OF RECIPIENT:',
font=('bold', '30'))
  l_rbg.place(x=20, y=100)
  l_runit = Label(ye4, text='ENTER NO. OF UNITS RECEIVED:',
font=('bold', '30'))#what if no. of units available is lesser than required amt
  l_runit.place(x=20, y=150)
  l_rsno = Label(ye4, text='ENTER RECIPIENT NO.:', font=('bold', '30'))
  l_rsno.place(x=20, y=200)
  l_rmno = Label(ye4, text='ENTER MOBILE NO.:', font=('bold', '30'))
  l_rmno.place(x=20, y=250)
  l = Label(ye4, text='NO. OF RECEPTIONS DONE:', font=('bold', '25'))
  l.place(x=1000, y=200)
  l1=Label(ye4,text=f[0][0],font=('bold','25'))
```

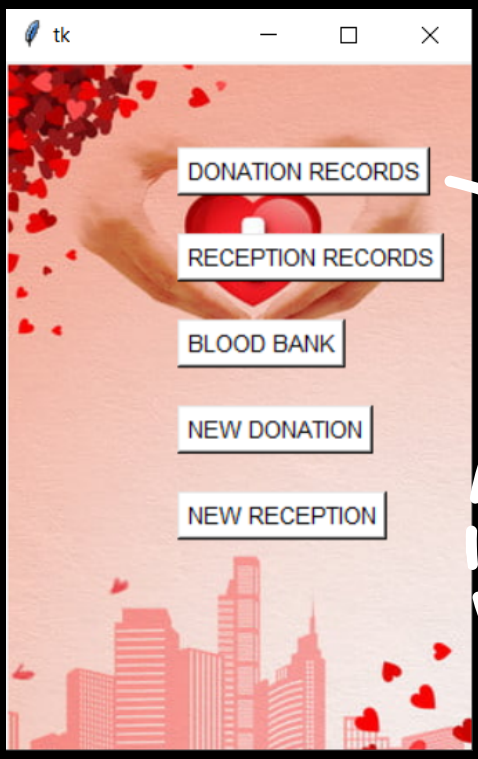
```
  l1.place(x=1000,y=250)
  b_rname = Entry(ye4)
  b_rname.place(x=800, y=60)
  choices = ['ABneg', 'ABpos', 'Aneg', 'Apos', 'Bneg', 'Bpos', 'Oneg',
'Opos']
  b_rbg = ttk.Combobox(ye4,values=choices)
  b_rbg.place(x=800, y=110)
  b_runits=Entry(ye4)
  b_runits.place(x=800,y=160)
  b_rsno=Entry(ye4)
  b_rsno.place(x=800,y=210)
  b_rmno = Entry(ye4)
  b_rmno.place(x=800, y=260)
  denter = Button(ye4, text='SUBMIT', font=('italic', 50), bg='white',
command=lambda: insert_r(b_rname,b_rbg,b_runits,b_rsno,b_rmno))
  denter.place(x=100, y=400)
  dshow=Button(ye4,text='SHOW
INFO',font=('italic',50),bg='white',command=lambda: retrieval(b_rbg))
  dshow.place(x=400, y=600)
  ye4.mainloop()
#displaying available units
def ua():
  ye5 = Toplevel ()
  ye5.geometry("300x200")
  bg = PhotoImage(file=r"C:\Users\Haripriya\Downloads\img4.png")

  # Show image using label
  label1 = Label(ye5, image=bg)
  label1.place(x=0, y=0)
  ye5.title('                                NUMBER OF UNITS AVAILABLE')
  e = Label(ye5, width=20, text='BLOOD TYPE', borderwidth=2,
relief='ridge', anchor='w', bg='brown1')
  e.grid(row=0, column=0)
  e = Label(ye5, width=20, text='UNITS', borderwidth=2, relief='ridge',
anchor='w', bg='brown1')
  e.grid(row=0, column=1)
```

```
  p3 = 1
  cur.execute("SELECT * FROM bloodbank")
  i = 1
  for k in cur:  # cur has the records as a list of tuples
      for j in range(len(k)):  # how many tuples in that list i.e. how many
records/rows
          e = Entry(ye5, width=20, fg='black')
          e.grid(row=i, column=j)
          e.insert(tk.END, k[j])
      i = i + 1
  ye5.mainloop()

mw = tk.Tk()
mw.geometry("270x400")
bg1 = PhotoImage(file = r"C:\Users\Haripriya\Downloads\img5.png")
lab = Label(mw, image=bg1)
lab.place(x=0, y=0)
B1=Button(mw,text='DONATION
RECORDS',font=('italic',10),bg='white',command=dr)
B1.place(x=100,y=50)
B2=Button(mw,text='RECEPTION
RECORDS',font=('italic',10),bg='white',command=rr)
B2.place(x=100,y=100)
B3=Button(mw,text='BLOOD
BANK',font=('italic',10),bg='white',command=ua)
B3.place(x=100,y=150)
B4=Button(mw,text='NEW
DONATION',font=('italic',10),bg='white',command=nd)
B4.place(x=100,y=200)
B5=Button(mw,text='NEW
RECEPTION',font=('italic',10),bg='white',command=nr)
B5.place(x=100,y=250)
mw.mainloop()
```

SAMPLE OUTPUTS

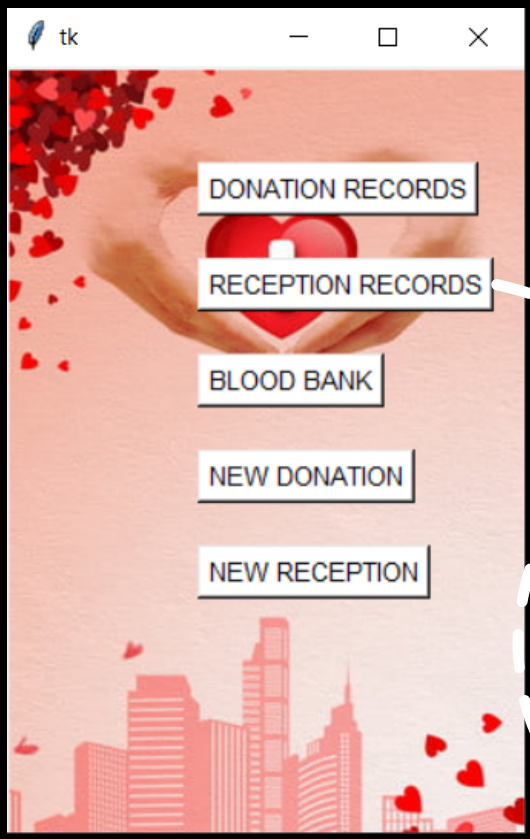
THE MAIN WINDOW





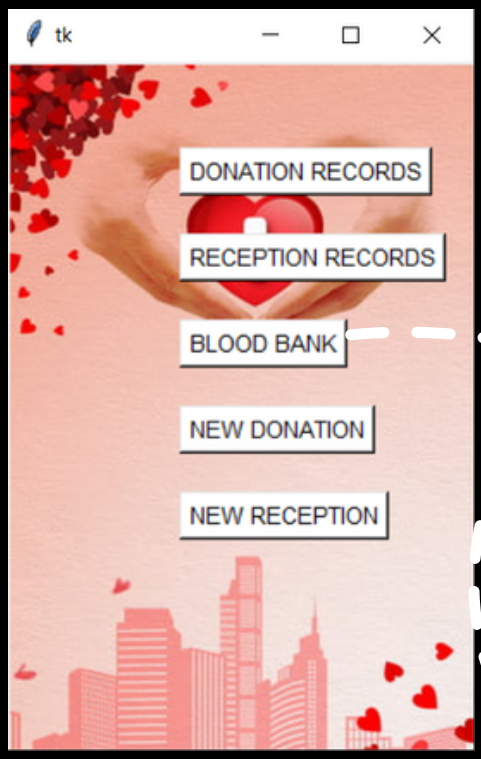
DONATION RECORDS

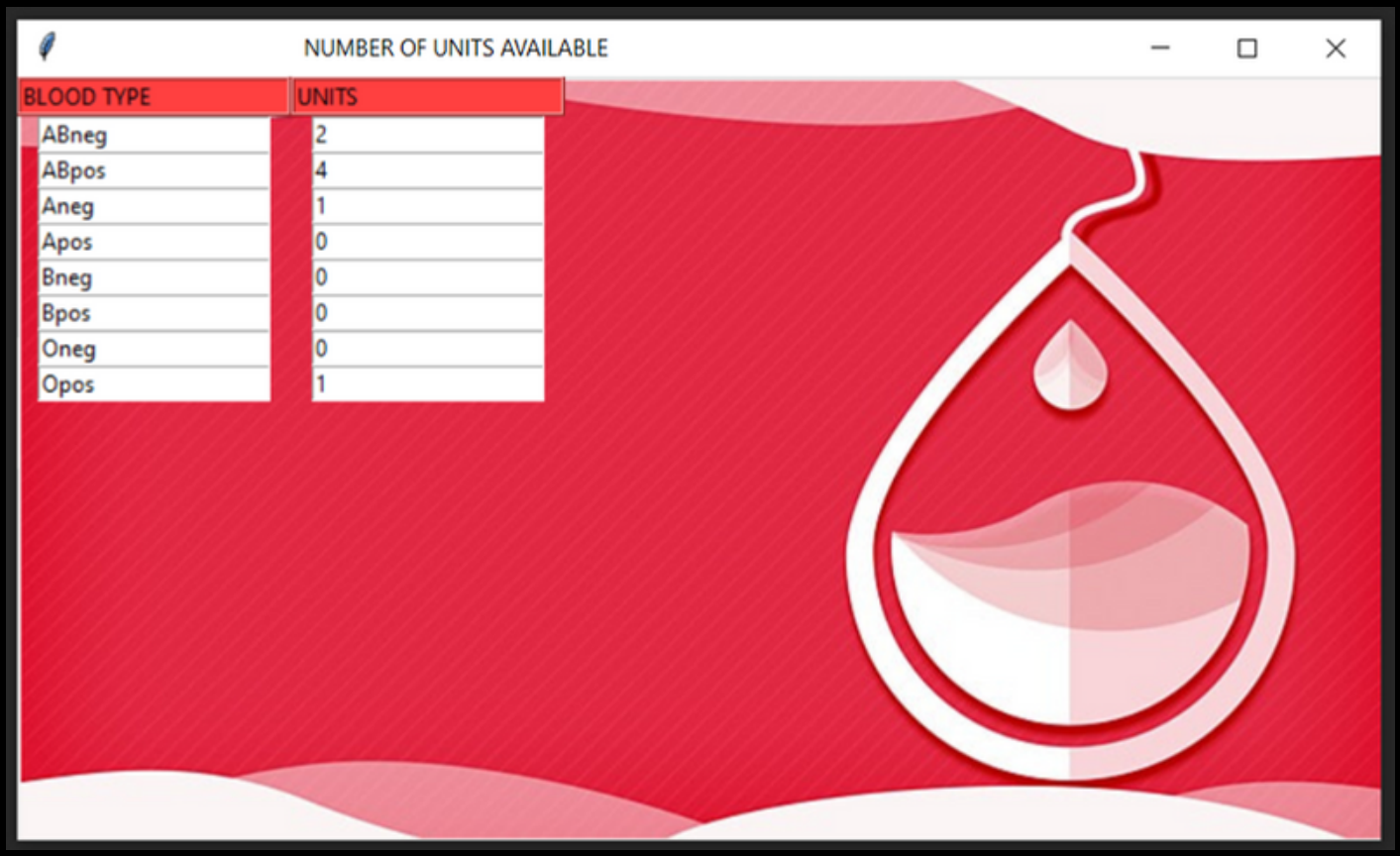
DONOR NO	DONOR NAME	BLOOD TYPE	UNITS	MOBILE NO.	DATE & TIME
1	Hari	ABneg	5	9111111111	2023-10-01 18:53:00
2	Vishnu	ABpos	4	9222222222	2023-10-01 18:53:46
3	Ganesh	Aneg	1	9333333333	2023-10-01 18:56:06
4	Raji	Opos	1	8888888888	2023-10-01 20:14:44

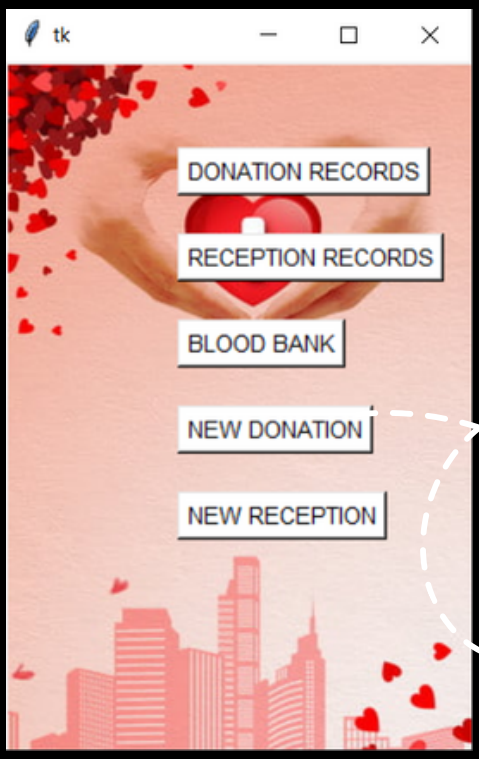


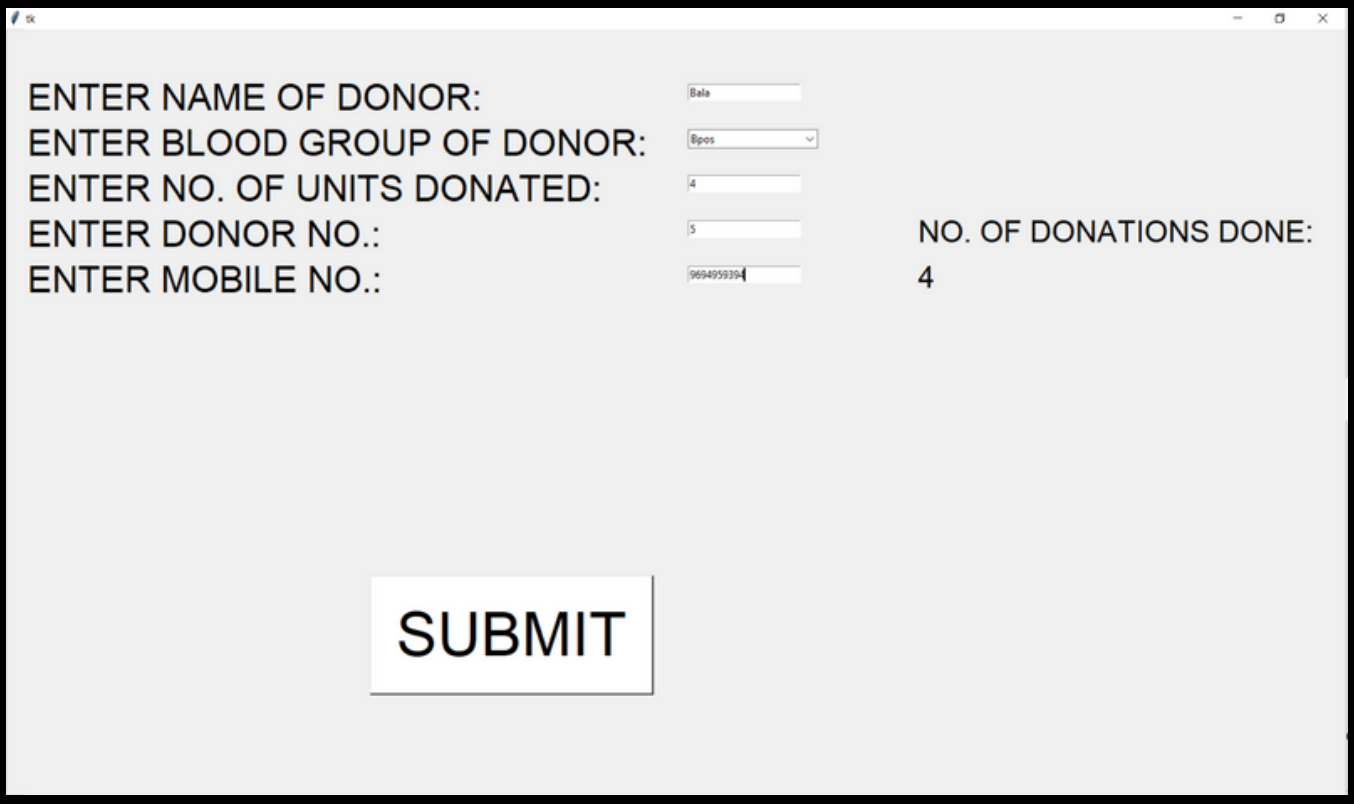
RECEPTION RECORDS

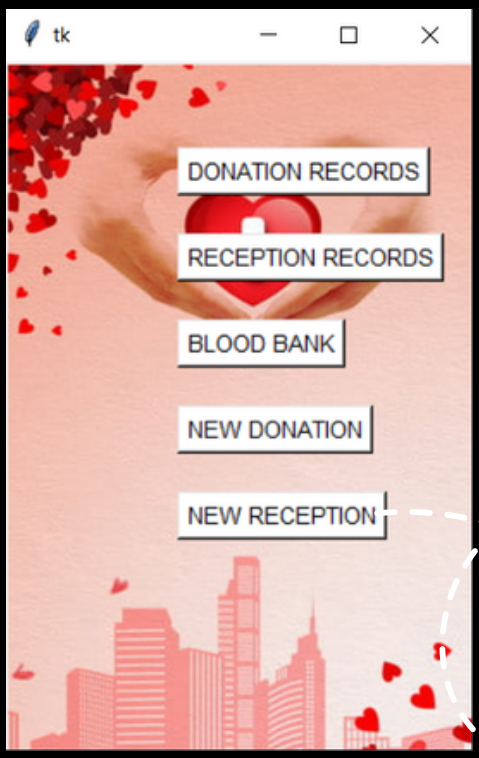
RECEPIENT NO.	RECEPIENT NAME	BLOOD TYPE	UNITS	MOBILE NO.	DATE & TIME
1	Vasu	ABneg	1	9696969696	2023-10-01 19:01:48
2	Balu	ABneg	2	7777777777	2023-10-01 20:16:15

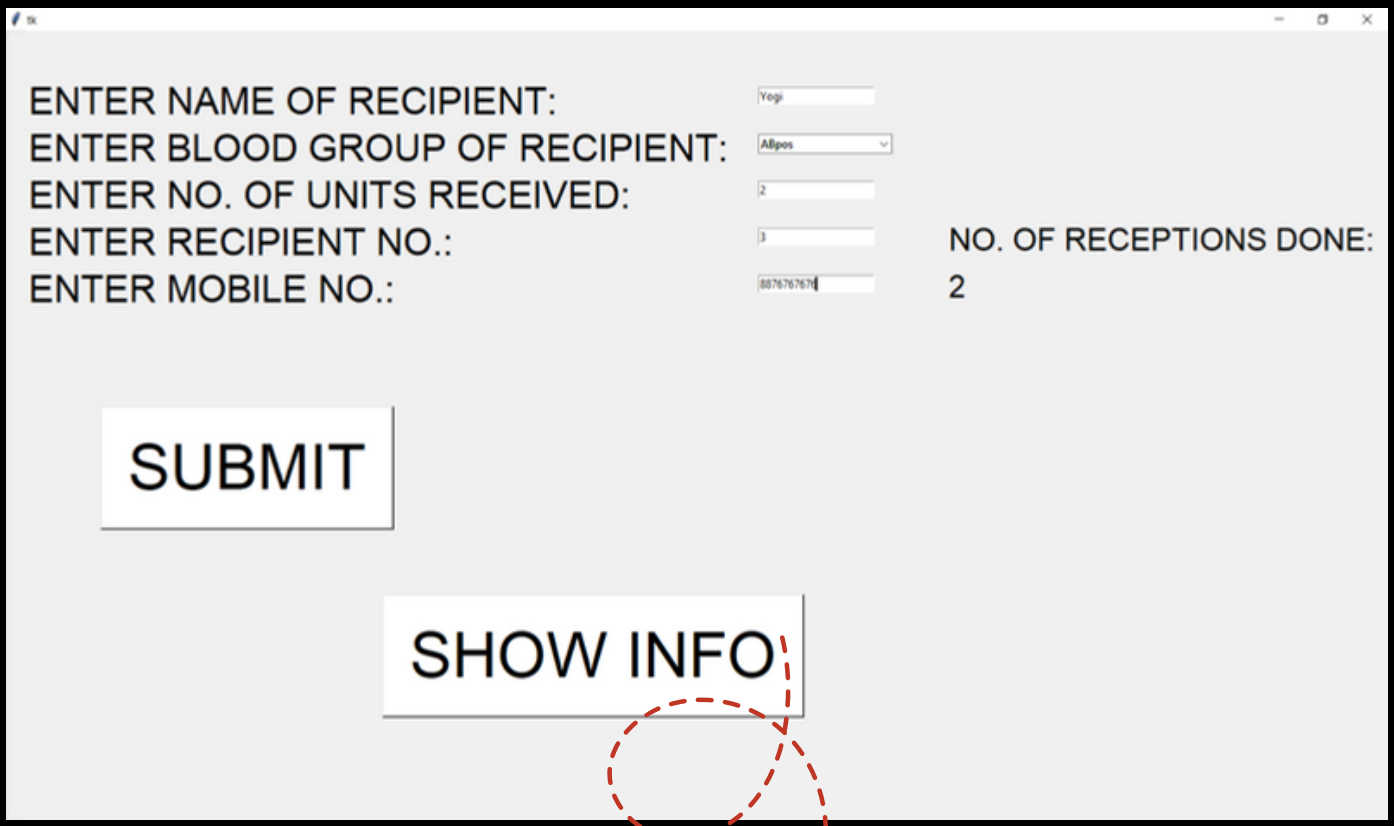

tk
DONATION RECORDS
RECEPTION RECORDS
BLOOD BANK
NEW DONATION
NEW RECEPTION

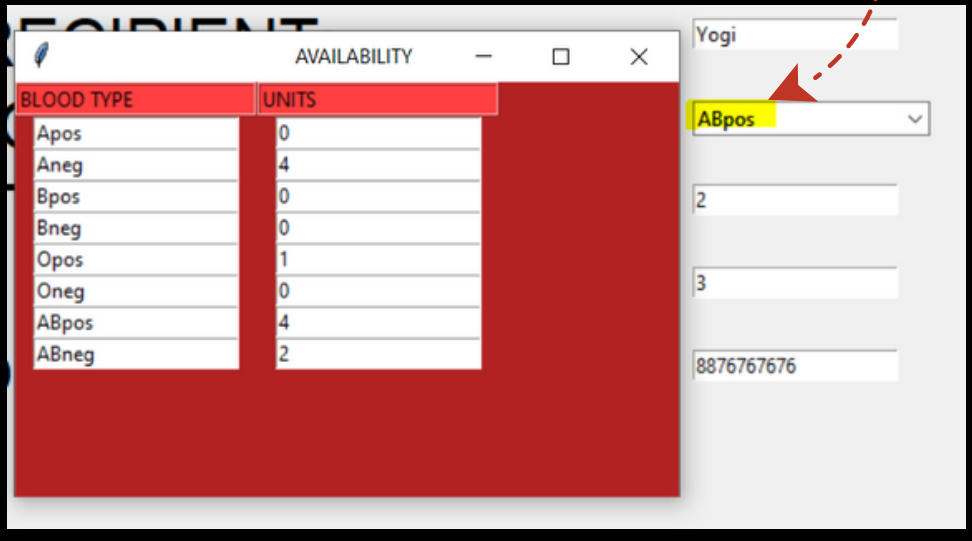

NUMBER OF UNITS AVAILABLE
BLOOD TYPE
UNITS
ABneg 2
ABpos 4
Aneg 1
Apos 0
Bneg 0
Bpos 0
Oneg 0
Opos 1


tk
DONATION RECORDS
RECEPTION RECORDS
BLOOD BANK
NEW DONATION
NEW RECEPTION

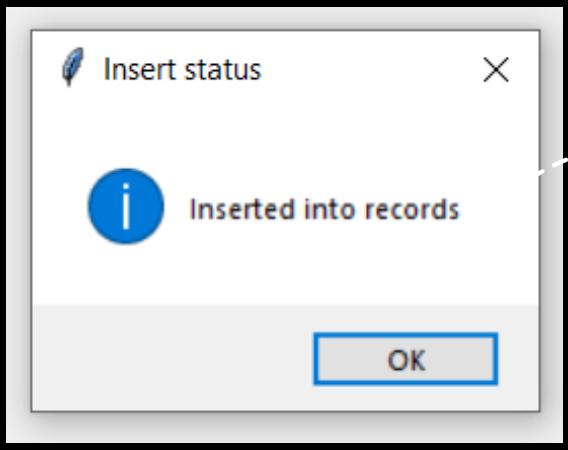

ENTER NAME OF DONOR:
Bala
ENTER BLOOD GROUP OF DONOR:
Bpos
ENTER NO. OF UNITS DONATED:
4
ENTER DONOR NO.:
5
ENTER MOBILE NO.:
9694959394
NO. OF DONATIONS DONE:
4
SUBMIT



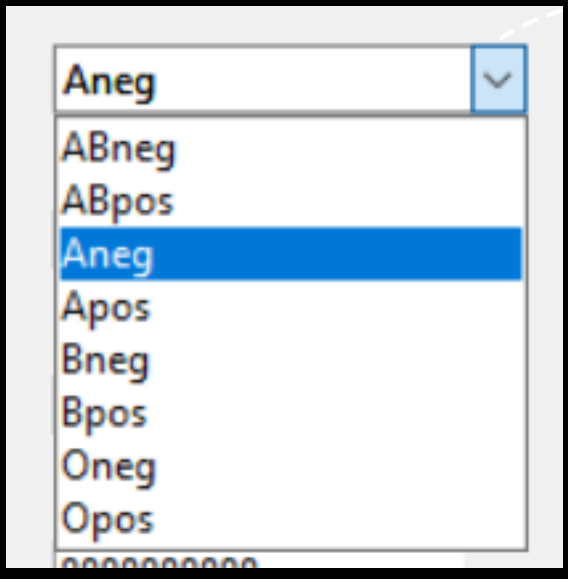




SHOWS MATCHING BLOOD GROUPS AND NUMBER OF UNITS AVAILABLE AS PER RBC COMPATIBILITY



MESSAGE BOX DISPLAYED UPON SUCCESSFUL SUBMISSION OF DATA



DROP DOWN MENU FOR CHOOSING BLOOD GROUP

CONCLUSION

THUS, WE CAN SAY THAT THIS PROJECT HAS A GOOD SCOPE OF UTILIZATION IN BLOOD DONATION CAMPS AND BLOOD BANK MANAGEMENT SYSTEM.

WE CAN MAKE ARRANGEMENTS FOR THE MYSQL TABLES TO BE TRANSFERRED TO THE NEW USER'S SYSTEM AS HE/SHE GETS ACCESS TO THE PROGRAM APPLICATION.

MOREOVER, THE ADDITION OF COLOURS AND BACKGROUND IMAGES SUGGESTS THE SCOPE
FOR AESTHETICS AND APPEAL.

THE ADDITION FURTHER FACILITIES LIKE TRACKING THE FREQUENCY OF DONATIONS/ RECEPTIONS TO CONTROL SUSPICIOUS ACTIVITES COULD MAKE IT FURTHER EASY FOR PRACTICAL USAGE.

BIBLIOGRAPHY

- https://www.tutorialspoint.com/index.htm
- https://www.geeksforgeeks.org/
- https://www.plus2net.com/python/tkinter-mysql.php
- https://www.w3schools.com/sql/sql_update.asp
- https://www.plus2net.com/python/tkinter-colors.php
- https://www.medicalnewstoday.com/articles/326279#summary