VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

COMPUTER IN SOCIETY

BLOOD BANK MANAGEMENT SYSTEMS

FINAL PROJECT REPORT

Submitted by:
Deepak Malpani
17BCE0306

Objective:

The main objective of the project on blood bank management system is to manage the details of blood bank, blood group, donor, blood stock. The project is totally built at administrative end and thus only administrator is guaranteed the process.

Introduction:

The “Blood Bank Management System” has been developed to override the problems prevailing in the practicing manual system. The inspiration of this project is to improve blood banks and to develop a blood bank information system which focuses on making an online system that is accessible for both donors and administrators. The system is developed for the administrators, who are the main authority in the system. Administrators can add, modify, delete, and query any donation 1 information if necessary. The administrator is also responsible for responding to the hospital’s blood requests and checking the stocks in the blood bank’s inventory. They can also update the personal information through the system, without having to contact the blood bank registry.

Literature Survey:

Survey of Existing Models:

During a literature survey we collect some of information about the blood bank management system located in city and rural area, some of the hospital have its own blood bank unit with each and all technical facilities. BD process starts with the arrival of the donor at the blood center. Donors can be divided in returning donors, who donate on an almost regular basis, and walk-in donors, who are entering the system occasionally or for the first time. In any case, donations can be made after a defined rest period from the previous one, which is defined by law. When a donor centers in the system for the first time, he/she is requested to provide personal (e.g., name, address, age, job, gender) and medical/health (e.g., diagnosis, lab results, treatments) data, which are digitally collected. Digital registration provides a good traceability of the transfusion cycle, from collection to blood distribution and transfusion.

Observation of Literature Survey:

The conduction of blood in rural areas is poor. The inventory management is not up to the mark. An effective, well-organized screening program and a good quality system is required for provisioning safe blood bags to patients and meeting the transfusion requirements. Several management problems are, both at a planning level (e.g., blood collection center location or staff dimensioning) and at an operational level (e.g., appointment scheduling).

Overview of Proposed Work:

Introduction:

The project provides the searching facilities based on various factors such as blood bank, donor, blood group, blood stock. It tracks all the information of blood bank, donor, blood group, blood stock. It is easy to understand by user and operator.

Framework for the proposed system:

The project has a back-end database which is managed in SQL Plus and a front-end UI in Python. The application is installed on Windows.

Module involved in your project:

Donor Module, Blood details module

Donor module:

```
Name                                    Null?    Type
 ----------------------------------------- -------- ----------------------------
 NAME                                              VARCHAR2(30)
 AGE                                               NUMBER
 GENDER                                            VARCHAR2(10)
 ADDRESS                                           VARCHAR2(50)
 CONTACTNO                                         NUMBER(10)
 ID                                                NUMBER
```

Blood details module:

```
Name                                Null?    Type
 ----------------------------------------- -------- ----------------------------
 BLOODGROUP                                          CHAR(3)
 PLATELET                                        NUMBER
 RBC                                        NUMBER
 REGDATE                                         DATE
 ID                                       NUMBER
```

Techniques used:

Model View Controller or MVC as it is called, is a software design pattern for developing web applications. A Model View Controlled is made up of the following parts:

Model: The lowest level of the pattern which is responsible for maintaining data.

View: This is responsible for displaying all or a portion of the data for the user.

Controller: Software code that controls the interactions between model and view.

Connections:

Connections are used to 'talk to' databases, and are represented by provider-specific classes such as **SqlConnection**. Commands travel over connections and resultsets are returned in the form of streams which can be read by a **DataReader** object, or pushed into a **DataSet** object.

Commands:

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as **SqlCommand**. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an INSERT statement against the **Northwind** database.

DataReaders:

The **DataReader** object is somewhat synonymous with a read-only/forward-only cursor over data. The **DataReader** API supports flat as well as hierarchical data. A **DataReader** object is returned after executing a command against a database. The format of the returned **DataReader** object is different from a recordset. For example, you might use the **DataReader** to show the results of a search list in a web page.

SQL SERVER

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an SQL Server Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields.

SQL Server Tables

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

Primary Key

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

Relational Database

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the table and enables you to define relationships between the tables.

Foreign Key

When a field is one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table.

Referential Integrity

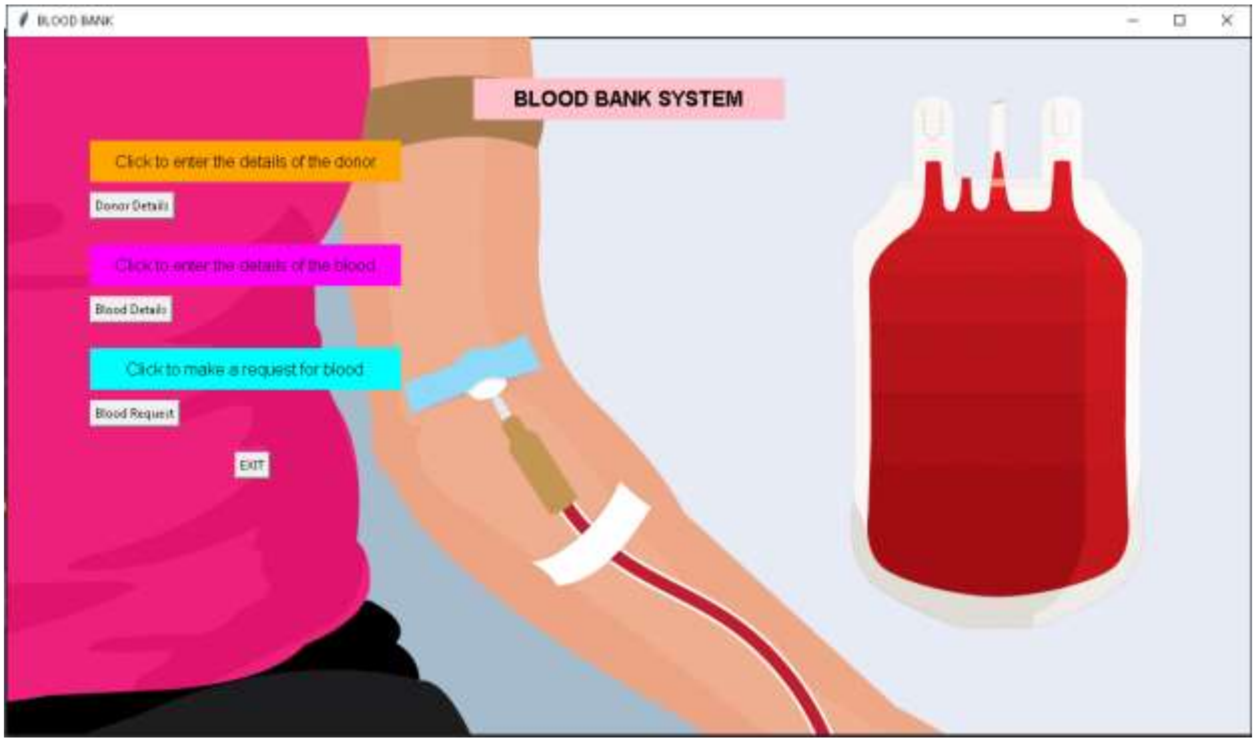
Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity.

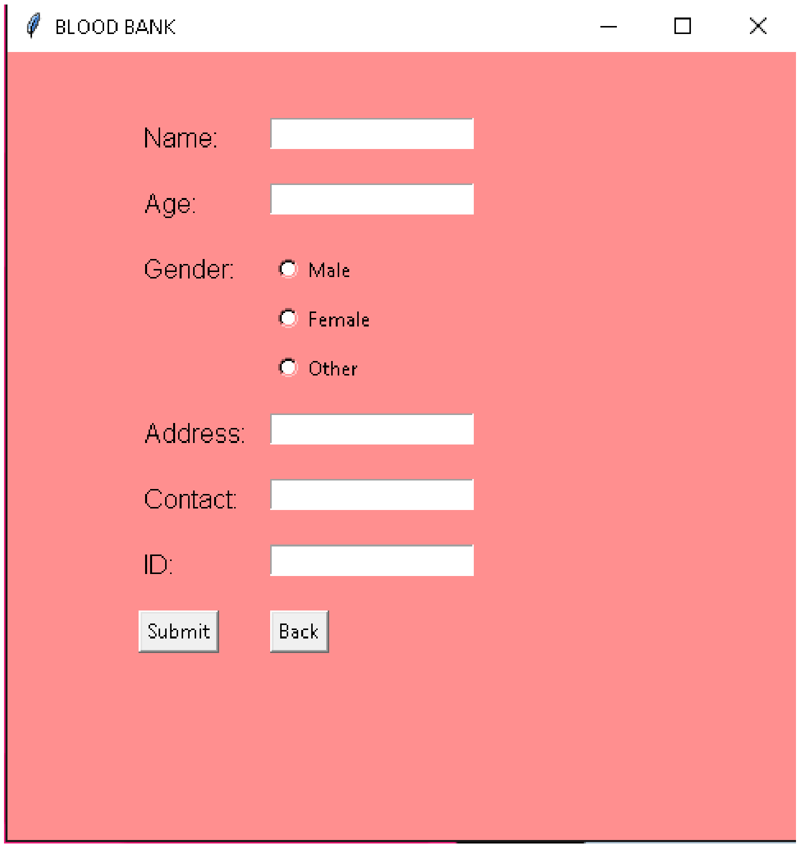
Python libraries used are cx_Oracle(for SQL Plus and Python connection) and tkinter(for UI design0

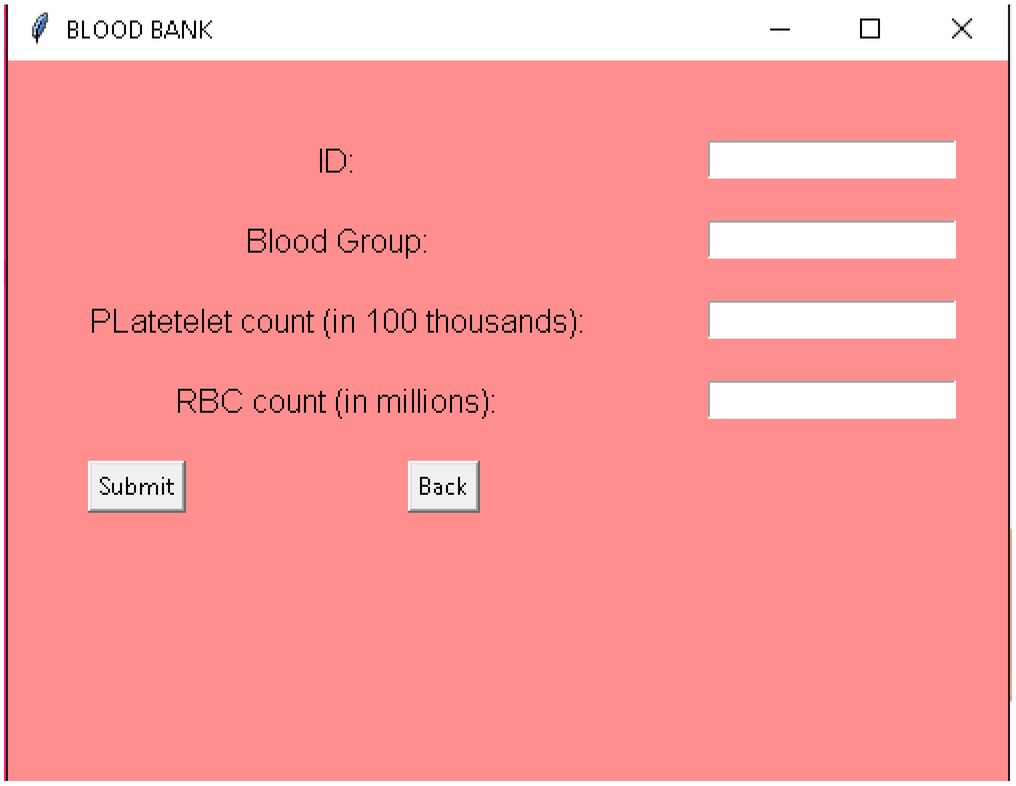
Software and Hardware Requirements:

Windows, Oracle Database 11g, SQL Plus, SQL Plus connector, Intel Core i3, Python Tkinter

Screenshots:


BLOOD BANK
BLOOD BANK SYSTEM
Click to enter the details of the donor
Donor Details
Click to enter the details of the blood
Blood Details
Click to make a request for blood
Blood Request
EXIT



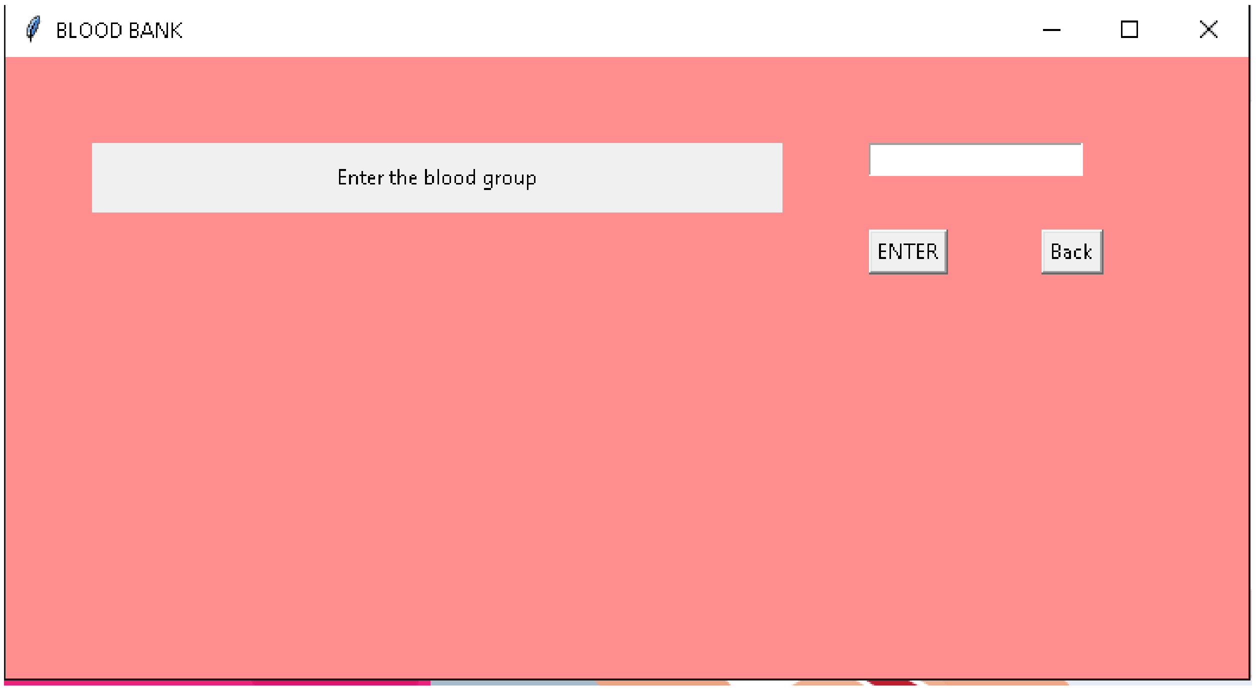

BLOOD BANK

ID:

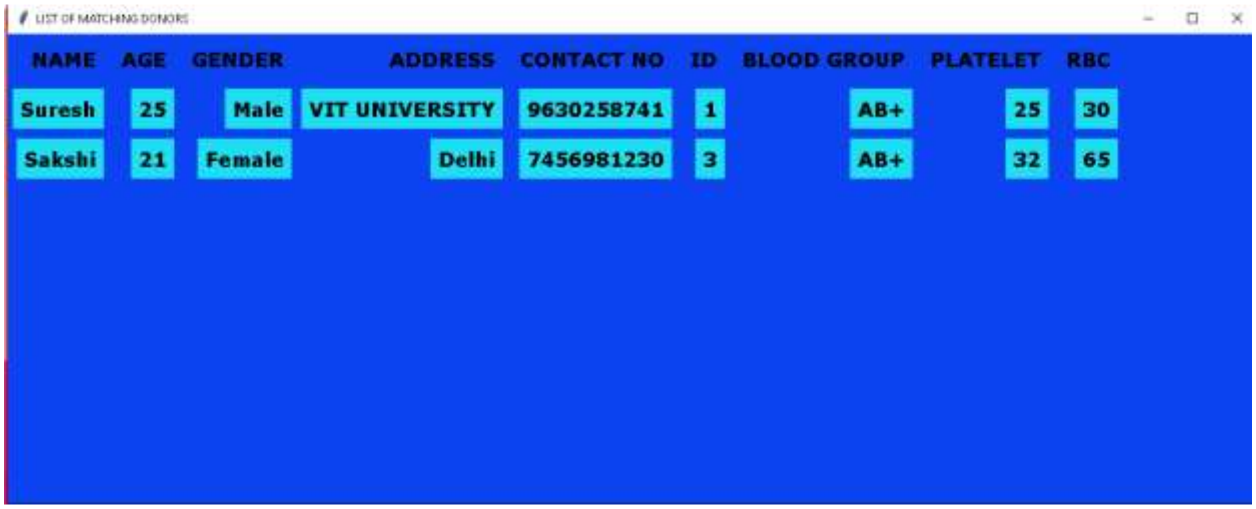
Blood Group:

PLatetelet count (in 100 thousands):

RBC count (in millions):


BLOOD BANK

Enter the blood group



LIST OF MATCHING DONORS

NAME	AGE	GENDER	ADDRESS	CONTACT NO	ID	BLOOD GROUP	PLATELET	RBC
Suresh	25	Male	VIT UNIVERSITY	9630258741	1	AB+	25	30
Sakshi	21	Female	Delhi	7456981230	3	AB+	32	65

Code:

```
from tkinter import *

import cx_Oracle

import os


connectString = os.getenv('con_connect')

con = cx_Oracle.connect('system/deepak123@127.0.0.1/InsuranceManagement')

cursor = con.cursor()

root = Tk()

image1 = PhotoImage(file="C://users/admin/Desktop/blood.png")

panel = Label(root, image=image1, bg="black").place(x=0, y=0, relwidth=1, relheight=1)

root.title("BLOOD BANK")

root.geometry("1200x672")

root.configure(background='white')

l3 = Label(root, text="BLOOD BANK SYSTEM", bg='pink', font="Helvetica 15
bold",fg='Black').place(x=450, y=40, w=300, h=40)

l1 = Label(root, text="Click to enter the details of the donor", bg='orange', font="Helvetica
12").place(x=80, y=100,

                                                          w=300, h=40)

b1 = Button(root, text="Donor Details", command=lambda: donordetails()).place(x=80, y=150)
```

```
l2 = Label(root, text="Click to enter the details of the blood",bg='magenta', font="Helvetica
12").place(x=80, y=200,
                                                                        w=300, h=40)

b2 = Button(root, text="Blood Details", command=lambda: blooddetails()).place(x=80, y=250)

l3 = Label(root, text="Click to make a request for blood", bg='cyan', font="Helvetica 12").place(x=80,
y=300, w=300,
                                                                    h=40)

b3 = Button(root, text="Blood Request", command=lambda: requestblood()).place(x=80, y=350)

b2 = Button(root, text="EXIT", command=lambda: stop(root)).place(x=220, y=400)

v = StringVar()



def insertDonor(name, age, gender, address, contactno,id):

    insert = "INSERT INTO donor(name,age,gender,address,contactno,id) VALUES(:1,:2,:3,:4,:5,:6)"

    cursor.execute(insert,(name,int(age),gender,address,int(contactno),int(id)))

    con.commit()




def insertBlood(bloodgroup, platelet, rbc,id):

    insert = "INSERT INTO blood(bloodgroup,platelet,rbc,regdate,id) VALUES(:1,:2,:3,SYSDATE,:4)"



    cursor.execute(insert,(bloodgroup,platelet,rbc,id))

    con.commit()



def retrieve(bg):

    request = "select * from donor d,blood b where d.id=b.id and b.bloodgroup='" + bg + "'"

    cursor.execute(request)

    rows = cursor.fetchall()
    con.commit()

    print(len(rows))

    return rows




def sel():

    selection = "You selected the option " + str(v.get())

    print(selection)




def donordetails():

    # global v

    root = Toplevel()

    root.title("BLOOD BANK")

    root.geometry("480x480")

    root.configure(background='#FF8F8F')

    l1 = Label(root, text="Name:", bg='white', font="Helvetica 12",background='#FF8F8F').place(x=80,
y=40)

    l2 = Label(root, text="Age:", bg='white', font="Helvetica 12",background='#FF8F8F').place(x=80,
y=80)

    l3 = Label(root, text="Gender:", bg='white', font="Helvetica 12",background='#FF8F8F').place(x=80,
y=120)

    l4 = Label(root, text="Address:", bg='white', font="Helvetica 12",background='#FF8F8F').place(x=80,
y=220)

    l5 = Label(root, text="Contact:", bg='white', font="Helvetica 12",background='#FF8F8F').place(x=80,
y=260)

    l6 = Label(root, text="Name:", bg='white', font="Helvetica 12",background='#FF8F8F').place(x=80,
y=300)

    e1 = Entry(root)

    e1.place(x=160, y=40)

    e2 = Entry(root)

    e2.place(x=160, y=80)
    r1 = Radiobutton(root, text="Male", variable=v, value="Male",
command=sel,background='#FF8F8F').place(x=160, y=120)

    v.set(3)

    r2 = Radiobutton(root, text="Female", variable=v, value="Female",
command=sel,background='#FF8F8F').place(x=160, y=150)

    r3 = Radiobutton(root, text="Other", variable=v, value="Other",
command=sel,background='#FF8F8F').place(x=160, y=180)

    # e3=Entry(root)

    # e3.place(x=100,y=120)

    e4 = Entry(root)

    e4.place(x=160, y=220)

    e5 = Entry(root)

    e5.place(x=160, y=260)

    e6=Entry(root)

    e6.place(x=160,y=300)

    b2=Button(root,text="Back",command=lambda : stop(root)).place(x=160,y=340)


    b1=Button(root,text="Submit",command=lambda :
insertDonor(e1.get(),e2.get(),str(v.get()),e4.get(),e5.get(),e6.get())).place(x=80,y=340)


    root.mainloop()




def blooddetails():

    root = Tk()

    root.title("BLOOD BANK")

    root.geometry("500x360")

    root.configure(background='#FF8F8F')

    l1 = Label(root, text="ID:", font="Helvetica 12",background='#FF8F8F').place(x=40, y=40, w=250,
h=20)

    l1 = Label(root, text="Blood Group:", font="Helvetica 12",background='#FF8F8F').place(x=40, y=80,
w=250, h=20)
```

```
    l2 = Label(root, text="PLatetelet count (in 100 thousands):", font="Helvetica
12",background='#FF8F8F').place(x=40, y=120, w=250, h=20)

    l3 = Label(root, text="RBC count (in millions):", font="Helvetica
12",background='#FF8F8F').place(x=40, y=160, w=250, h=20)

    # l4=Label(root,text="Date Of Entry count:").place(x=40,y=160)

    e4=Entry(root)

    e4.place(x=350,y=40)

    e1 = Entry(root)

    e1.place(x=350, y=80)

    e2 = Entry(root)

    e2.place(x=350, y=120)

    e3 = Entry(root)

    e3.place(x=350, y=160)

    b2 = Button(root, text="Back", command=lambda: stop(root)).place(x=200, y=200)

    b1 = Button(root, text="Submit", command=lambda: insertBlood(e1.get(), e2.get(),
e3.get(),e4.get())).place(x=40, y=200)


    # img = PhotoImage(file="/home/aishwarya/Downloads/b1.gif")

    # panel = Label(root, image = img,bg="#F6B88D").place(x=200,y=200,w=400,h=400)

    root.mainloop()

def grid1(bg):

    root = Tk()

    root.title("LIST OF MATCHING DONORS")

    root.geometry("1280x480")

    root.configure(background='#0C43F0')

    rows = retrieve(bg)

    Label(root, text='NAME', bg="#0C43F0", font="Verdana 15 bold").grid(row=0, column=0, sticky='E',
padx=5,

                                                                        pady=5, ipadx=5, ipady=5)

    Label(root, text='AGE', bg="#0C43F0", font="Verdana 15 bold").grid(row=0, column=1, sticky='E',
padx=5,

                                                                        pady=5, ipadx=5, ipady=5)
    Label(root, text='GENDER', bg="#0C43F0", font="Verdana 15 bold").grid(row=0, column=2,
sticky='E', padx=5,
                                                    pady=5, ipadx=5, ipady=5)
    Label(root, text='ADDRESS', bg="#0C43F0", font="Verdana 15 bold").grid(row=0, column=3,
sticky='E', padx=5,
                                                    pady=5, ipadx=5, ipady=5)
    Label(root, text='CONTACT NO', bg="#0C43F0", font="Verdana 15 bold").grid(row=0, column=4,
sticky='E', padx=5,
                                                    pady=5, ipadx=5, ipady=5)
    Label(root, text='ID', bg="#0C43F0", font="Verdana 15 bold").grid(row=0, column=5, sticky='E',
padx=5,
                                                    pady=5, ipadx=5, ipady=5)
    Label(root, text='BLOOD GROUP', bg="#0C43F0", font="Verdana 15 bold").grid(row=0, column=6,
sticky='E', padx=5,
                                                    pady=5, ipadx=5, ipady=5)
    Label(root, text='PLATELET', bg="#0C43F0", font="Verdana 15 bold").grid(row=0, column=7,
sticky='E', padx=5,
                                                    pady=5, ipadx=5, ipady=5)
    Label(root, text='RBC', bg="#0C43F0", font="Verdana 15 bold").grid(row=0, column=8, sticky='E',
padx=5,
                                                    pady=5, ipadx=5, ipady=5)

    x = 1
    for row in rows:
        Label(root, text=row[0], bg="#1EDEF2", font="Verdana 15 bold").grid(row=x, column=0,
sticky='E', padx=5,
                                                        pady=5, ipadx=5, ipady=5)
        Label(root, text=row[1], bg="#1EDEF2", font="Verdana 15 bold").grid(row=x, column=1,
sticky='E', padx=5,
                                                        pady=5, ipadx=5, ipady=5)
        Label(root, text=row[2], bg="#1EDEF2", font="Verdana 15 bold").grid(row=x, column=2,
sticky='E', padx=5,
                                                        pady=5, ipadx=5, ipady=5)
        Label(root, text=row[3], bg="#1EDEF2", font="Verdana 15 bold").grid(row=x, column=3,
sticky='E', padx=5,
                                                                                pady=5, ipadx=5, ipady=5)
        Label(root, text=row[4], bg="#1EDEF2", font="Verdana 15 bold").grid(row=x, column=4,
sticky='E', padx=5,
                                                                                pady=5, ipadx=5, ipady=5)
        Label(root, text=row[5], bg="#1EDEF2", font="Verdana 15 bold").grid(row=x, column=5,
sticky='E', padx=5,
                                                                                pady=5, ipadx=5, ipady=5)
        Label(root, text=row[6], bg="#1EDEF2", font="Verdana 15 bold").grid(row=x, column=6,
sticky='E', padx=5,
                                                                                pady=5, ipadx=5, ipady=5)
        Label(root, text=row[7], bg="#1EDEF2", font="Verdana 15 bold").grid(row=x, column=7,
sticky='E', padx=5,
                                                                                pady=5, ipadx=5, ipady=5)
        Label(root, text=row[8], bg="#1EDEF2", font="Verdana 15 bold").grid(row=x, column=8,
sticky='E', padx=5,
                                                                                pady=5, ipadx=5, ipady=5)

        x = x + 1
    root.mainloop()



def requestblood():
    root = Tk()
    root.title("BLOOD BANK")
    root.geometry("720x360")
    root.configure(background='#FF8F8F')
    l = Label(root, text="Enter the blood group").place(x=50, y=50, w=400, h=40)
    e = Entry(root)
    e.place(x=500, y=50)
    b2 = Button(root, text="Back", command=lambda: stop(root)).place(x=600, y=100)
    b = Button(root, text="ENTER", command=lambda: grid1(e.get())).place(x=500, y=100)
    root.mainloop()



def stop(root):
    root.destroy()



root.mainloop()
```

Conclusion:

This project is only a humble venture to satisfy the needs to manage their project work. Several user-friendly coding has also been adopted. The description of requirement specification and the actions that can be done on these is provided. Finally, the system is implemented and tested according to test cases.

References:

www.google.com

www.wikipedia.com

www.geeksforgeeks.com