|  |
| --- |
| **ONLINE BLOOD BANK WITH HOSPITAL MANAGEMENT**  **21CSS101J – PROGRAMMING FOR PROBLEM-SOLVING**  **Mini Project Report**  *Submitted by*  **ANURAG SANYAL [Reg. No.: RA2311003011823]**  **B.Tech. CSE - CORE**  **TUHIN SARKAR [Reg. No.: RA2311003011883]**  **B.Tech. CSE - CORE**  **SRMIST-01.jpg**  **SCHOOL OF COMPUTING**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  CHENGALPATTU DISTRICT  **November 2023**  **COLLEGE OF ENGINEERING AND TECHNOLOGY**  **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY**  **(Under Section 3 of UGC Act, 1956)**  S.R.M. NAGAR, KATTANKULATHUR – 603 203  **SRMIST-01.jpg**  **BONAFIDE CERTIFICATE**  Certified that Mini project report titled ONLINE BLOOD BANK WITH HOSPITAL MANAGEMENT is the bonafide work of Reg.No RA2311003011823 Name ANURAG SANYAL who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.  **SIGNATURE SIGNATURE**  **(GUIDE) (HEAD OF THE DEPARTMENT)** |

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S No.** | **Title** | **Page No.** |
| 1 | Problem Statement | 4 |
| 2 | Methodology / Procedure/ Algorithm | 5 |
| 3 | Flowchart | 6 |
| 4 | Coding (C/Python) | 7 |
| 5 | Front-end code (HTML, CSS, Javascript) [Optional] | 11 |
| 6 | Modules of the proposed work | 13 |
| 7 | Results/Screenshots | 14 |
| 8 | Conclusion | 15 |
| 9 | References | 16 |

1. **Problem Statement**

**Despite the crucial role that blood banks play in ensuring a stable and sufficient blood supply for medical treatments, there exist significant challenges that need to be addressed. These challenges encompass issues related to blood donation, inventory management, distribution logistics, and overall efficiency in the blood banking system. Inefficiencies in donor recruitment, retention, and collection processes, as well as suboptimal utilization of available blood resources, contribute to periodic shortages and hinder the ability to meet the ever-growing demand for blood products. Additionally, ensuring the safety, quality, and timely delivery of blood units to healthcare facilities poses a continuous challenge. The problem statement for the blood bank, therefore, revolves around the need to optimize and streamline the entire blood supply chain—from donor recruitment to blood transfusion—in order to enhance the effectiveness, accessibility, and sustainability of the blood banking system.**

Blood is a vital component of the human body, carrying oxygen and nutrients to cells and removing waste products. It is essential for life, and its loss can have serious consequences, including death. Blood banks play a critical role in meeting the demand for blood products, which are used in a variety of medical procedures, including:

* **Trauma care:** Patients who have suffered severe injuries often require large amounts of blood to replace lost blood volume and restore oxygen delivery to tissues.
* **Surgery:** Blood transfusions are often needed during surgery to compensate for blood loss and maintain stable blood pressure.
* **Cancer treatment:** Many cancer treatments, such as chemotherapy and radiation therapy, can damage the bone marrow, reducing the body's ability to produce blood cells. Blood transfusions are often necessary to support these patients during treatment.
* **Chronic diseases:** People with certain chronic diseases, such as sickle cell anemia and hemophilia, may require regular blood transfusions to manage their condition.

Despite the critical role that blood banks play in saving lives, there is often a shortage of blood products. This is due to a number of factors, including:

* **Limited number of blood donors:** Only about 3% of eligible people in the United States donate blood each year.
* **Strict eligibility criteria:** Certain medical conditions, medications, and travel history can make people ineligible to donate blood.
* **Short shelf life of blood products:** Red blood cells can only be stored for up to 42 days, while platelets can only be stored for 5 days. Plasma can be stored for up to a year, but it is often in high demand.

2.**Methodology / Procedure/ Algorithm**

1.Import necessary libraries:

.mysql.connector for MySQL database connection.

.tkinter for GUI

.ttk for themed Tkinter widgets.

.messagebox for displaying messages.

2.Establish a connection to the MySQL database with the specified credentials.

.Define a list ls containing blood groups.

.Define a function donate() to create a donation window:

.Create a new Tkinter window(window2())

.Add labels, entry fields, and a dropdown menu for user input.

.Define a function adduser() to handle the donation process.

.Update the blood bank database with the donated blood and add the donor to the user database.

3.Define a function checkstock() to create a window for checking blood stock:

.Add buttons for checking stock by blood group and listing all blood in stock.

.Define functions check\_by\_blood() and check\_all\_blood() to handle the respective actions.

4.Define a function check\_donor() to create a window for checking donors:

.Add buttons for checking donors by name and listing all donors.

.Define functions check\_by\_name() and check\_all\_user() to handle the respective actions.

5.Define functions (check\_by\_blood,check\_all\_blood,check\_by\_name,check\_all\_user) to query the database and display results using Tkinter Treeview.

6.Define the main window (window) with buttons for donation, checking stock, and checking donors.

7.Run the main Tkinter event loop (window.mainloop()) to display the main window and handle user interactions.

**2.Flow chart**

Enter Donor’s name

Database Updated

YES

Display results

Check by blood type

Want to check blood stock?

*NO*

**4. Coding (Python)**

import mysql.connector as c

import tkinter as tk

from tkinter import ttk, messagebox

from tkinter import \*

from tkinter import Tk, ttk

con = c.connect(user = 'root', host = "localhost", passwd = "admin",database = "blood\_bank")

cur = con.cursor()

ls = ["A+","B+","AB+","O+","A-","B-","AB-","O-"]

def donate():

window2 = Tk()

window2.geometry("900x400")

l1 = tk.Label(window2, text = "Name of Donator", width = 30 , font = "bold")

l1.grid(row= 1, column = 1)

e1 = tk.Entry(window2, width=30)

e1.grid(row =1, column = 3)

l2 = tk.Label(window2, text = "Blood Group", width = 30, font = "bold")

l2.grid(row =3, column =1)

e2 = StringVar(window2)

e2.set("Blood Group")

drop = OptionMenu(window2, e2, \*ls)

drop.grid(row= 3, column = 3)

b1 = tk.Button(window2, text = "Proceed",width = 30, command = lambda:add\_user(e1,e2))

b1.grid(row= 5, column = 2)

window2.mainloop()

def add\_user(e1,e2):

name = e1.get()

blood = e2.get()

try:

cmd = "update bank set qty = qty + 1 where blood\_type = '{}'".format(blood)

cur.execute(cmd)

con.commit()

cmd = "insert into user values('{}','{}')".format(name,blood)

cur.execute(cmd)

con.commit()

messagebox.showinfo("Success","Successfully Updated the database")

except Exception as e:

messagebox.showerror("Error","We ran down some error --> " + str(e) + " please try again...")

def check\_stock():

window3 = Tk()

window3.geometry("900x400")

l1 = tk.Label(window3, text = "Blood Group",width = 30, font = "bold")

l1.grid(row= 1, column =1 )

e1 = StringVar(window3)

e1.set("Blood Group")

drop = OptionMenu(window3, e1,\*ls)

drop.grid(row= 1, column =3)

b1 = tk.Button(window3, text = "Check By Blood Group",width = 30, command = lambda: check\_by\_blood(e1))

b1.grid(row =2 , column = 2)

b2 = tk.Button(window3, text = "List All in Store",width = 30, command = lambda: check\_all\_blood())

b2.grid(row =3 , column = 2)

window3.mainloop()

def check\_by\_blood(e1):

blood\_type = e1.get()

window4 = Tk()

window4.geometry("600x600")

tree = ttk.Treeview(window4, columns = ("c1","c2"), show = "headings", height = 30)

tree.column("#1", anchor = "center")

tree.heading("#1", text = "Blood Type")

tree.column("#2", anchor = "center")

tree.heading("#2", text = "Quantity")

cmd = "select \* from bank where blood\_type = '{}'".format(blood\_type)

cur.execute(cmd)

data = cur.fetchone()

bl = data[0]

qty = data[1]

tree.insert('','end', text = "1", values =(bl,qty))

tree.grid(row =1 , column =1)

window4.mainloop()

def check\_all\_blood():

window4 = Tk()

window4.geometry("600x600")

tree = ttk.Treeview(window4, columns = ("c1","c2"), show = "headings", height = 30)

tree.column("#1", anchor = "center")

tree.heading("#1", text = "Blood Type")

tree.column("#2", anchor = "center")

tree.heading("#2", text = "Quantity")

cmd = "select \* from bank"

cur.execute(cmd)

data = cur.fetchall()

c = 1

for i in data:

bl = i[0]

qty = i[1]

tree.insert('','end', text = str(c), values =(bl,qty))

c +=1

tree.grid(row =1 , column =1)

window4.mainloop()

def check\_donor():

window5 = Tk()

window5.geometry("900x600")

l1 = tk.Label(window5, text = "Name of Donator", width = 30 , font = "bold")

l1.grid(row= 1, column = 1)

e1 = tk.Entry(window5, width=30)

e1.grid(row =1, column = 3)

b1 = tk.Button(window5, text = "Check By Name",width = 30, command = lambda: check\_by\_name(e1))

b1.grid(row =2 , column = 2)

b2 = tk.Button(window5, text = "List All Donor",width = 30, command = lambda: check\_all\_user())

b2.grid(row =3 , column = 2)

window5.mainloop()

def check\_by\_name(e1):

name = e1.get()

try:

window6 = Tk()

window6.geometry("600x600")

tree = ttk.Treeview(window6, columns = ("c1","c2"), show = "headings", height = 30)

tree.column("#1", anchor = "center")

tree.heading("#1", text = "Donor Name")

tree.column("#2", anchor = "center")

tree.heading("#2", text = "Blood Group")

cmd = "select \* from user where name = '{}'".format(name)

cur.execute(cmd)

data = cur.fetchall()

c = 1

for i in data:

name = i[0]

bl\_gp = i[1]

tree.insert('','end', text = str(c), values =(name,bl\_gp))

c +=1

tree.grid(row =1 , column =1)

except Exception as e:

messagebox.showerror("Error","We ran into some error --> "+ str(e) + " Please try again...")

def check\_all\_user():

try:

window7 = Tk()

window7.geometry("600x600")

tree = ttk.Treeview(window7, columns = ("c1","c2"), show = "headings", height = 30)

tree.column("#1", anchor = "center")

tree.heading("#1", text = "Donor Name")

tree.column("#2", anchor = "center")

tree.heading("#2", text = "Blood Group")

cmd = "select \* from user"

cur.execute(cmd)

data = cur.fetchall()

c = 1

for i in data:

name = i[0]

bl\_gp = i[1]

tree.insert('','end', text = str(c), values =(name,bl\_gp))

c +=1

tree.grid(row =1 , column =1)

except Exception as e:

messagebox.showerror("Error","We ran into some error --> "+ str(e) + " Please try again...")

window =Tk()

window.geometry("500x500")

window.configure(bg = "blue")

l1 = tk.Label(window, text = "Welcome to Blood Bank", width = 30, font = "bold")

l1.pack()

l1.configure(bg ="blue", fg = "white")

b1 = tk.Button(window, text = "Donate", width = 30 ,command = lambda:donate())

b1.pack()

b2 = tk.Button(window, text = "Check Stock", width = 30 , command = lambda:check\_stock())

b2.pack()

b3 = tk.Button(window,text = "Check Donor", width = 30, command = lambda: check\_donor())

b3.pack()

window.mainloop()

**5. Front-end code (HTML, CSS, Javascript)**

import tkinter as tk

from tkinter import ttk, messagebox

ls = ["A+", "B+", "AB+", "O+", "A-", "B-", "AB-", "O-"]

def donate():

window2 = tk.Tk()

window2.geometry("900x400")

l1 = tk.Label(window2, text="Name of Donator", width=30, font="bold")

l1.grid(row=1, column=1)

e1 = tk.Entry(window2, width=30)

e1.grid(row=1, column=3)

l2 = tk.Label(window2, text="Blood Group", width=30, font="bold")

l2.grid(row=3, column=1)

e2 = tk.StringVar(window2)

e2.set("Blood Group")

drop = tk.OptionMenu(window2, e2, \*ls)

drop.grid(row=3, column=3)

b1 = tk.Button(window2, text="Proceed", width=30, command=lambda: add\_user(e1, e2))

b1.grid(row=5, column=2)

window2.mainloop()

def add\_user(e1, e2):

name = e1.get()

blood = e2.get()

try:

# Update blood bank quantity

cmd = "UPDATE bank SET qty = qty + 1 WHERE blood\_type = '{}'".format(blood)

cur.execute(cmd)

con.commit()

# Insert donor into user table

cmd = "INSERT INTO user VALUES('{}','{}')".format(name, blood)

cur.execute(cmd)

con.commit()

messagebox.showinfo("Success", "Successfully Updated the database")

except Exception as e:

messagebox.showerror("Error", "Error: " + str(e) + ". Please try again...")

def check\_stock():

window3 = tk.Tk()

window3.geometry("900x400")

l1 = tk.Label(window3, text="Blood Group", width=30, font="bold")

l1.grid(row=1, column=1)

e1 = tk.StringVar(window3)

e1.set("Blood Group")

drop = tk.OptionMenu(window3, e1, \*ls)

drop.grid(row=1, column=3)

b1 = tk.Button(window3, text="Check By Blood Group", width=30, command=lambda: check\_by\_blood(e1))

b1.grid(row=2, column=2)

b2 = tk.Button(window3, text="List All in Store", width=30, command=lambda: check\_all\_blood())

b2.grid(row=3, column=2)

window3.mainloop()

# ... (similar functions for check\_by\_blood, check\_all\_blood, check\_donor, etc.)

window = tk.Tk()

window.geometry("500x500")

window.configure(bg="blue")

l1 = tk.Label(window, text="Welcome to Blood Bank", width=30, font="bold")

l1.pack()

l1.configure(bg="blue", fg="white")

b1 = tk.Button(window, text="Donate", width=30, command=lambda: donate())

b1.pack()

b2 = tk.Button(window, text="Check Stock", width=30, command=lambda: check\_stock())

b2.pack()

b3 = tk.Button(window, text="Check Donor", width=30, command=lambda: check\_donor())

b3.pack()

window.mainloop()

**6. Modules of the proposed work**

The implementation of the blood bank system in the provided code facilitates a basic system for blood donation management. Let's conclude the key aspects and implications of this blood bank system:

Donation process:

Users can donate blood by providing their name and selecting their blood group.

The system updates the blood bank database, incrementing the quantity of the corresponding blood group.

The donor information is stored in a separate user database.

Stock Checking:

Users can check the stock of a specific blood group or list all available blood in the blood bank.

The system provides a visual representation of the available blood stock, helping users and administrators monitor inventory.

Donor information:

Users can check donor information by searching for a specific donor's name or listing all donors.

This functionality allows for easy tracking of donors and their respective blood groups.

Graphical User Interface(GUI):

The system incorporates a GUI using the tkinter library, making it user-friendly and accessible.

GUI elements such as labels, entry fields, buttons, and dropdown menus enhance the user experience.

Database Interaction:

The system interacts with a MySQL database to store and retrieve information.

SQL queries are used to update blood bank quantities and manage donor information.

Error Handling:

While the code includes basic error handling through try and except blocks, further enhancements could be made to handle specific errors more gracefully.

Error messages are displayed using messagebox to alert users about issues.

Room for Improvement:

The system can be extended and improved with additional features, such as user authentication, more detailed donor information, and real-time updates on blood stock.

Enhanced validation checks and improved error messages could provide a more robust user experience.

Usability:

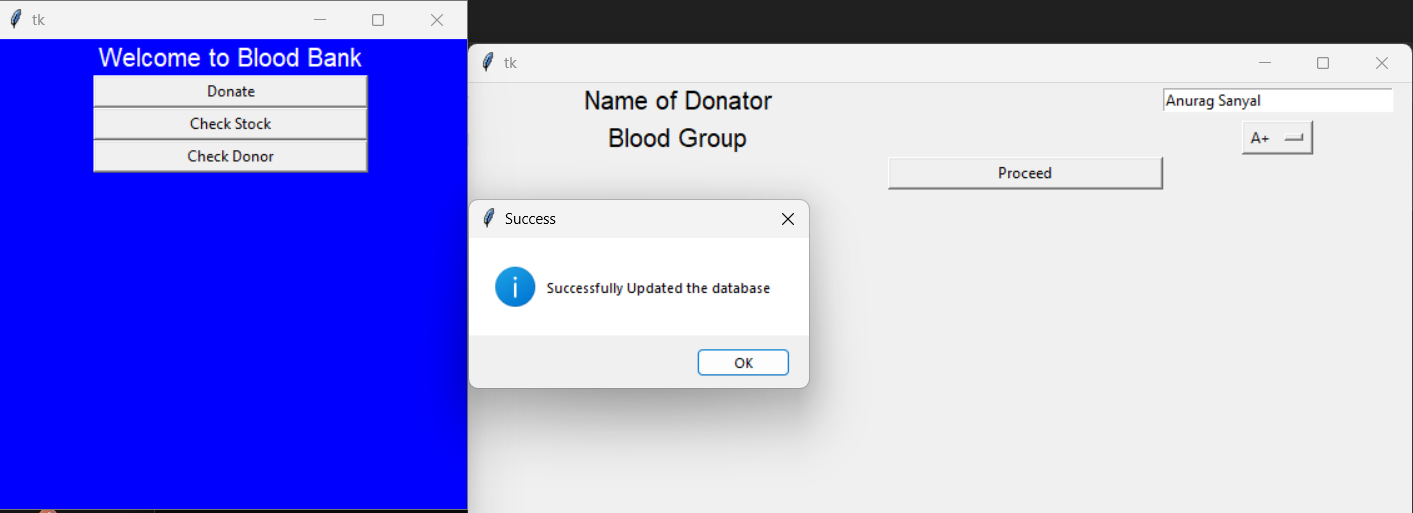
The blood bank system aims to streamline the blood donation process, making it more organized and efficient.

It provides a tool for administrators to manage blood stock and donor information effectively.

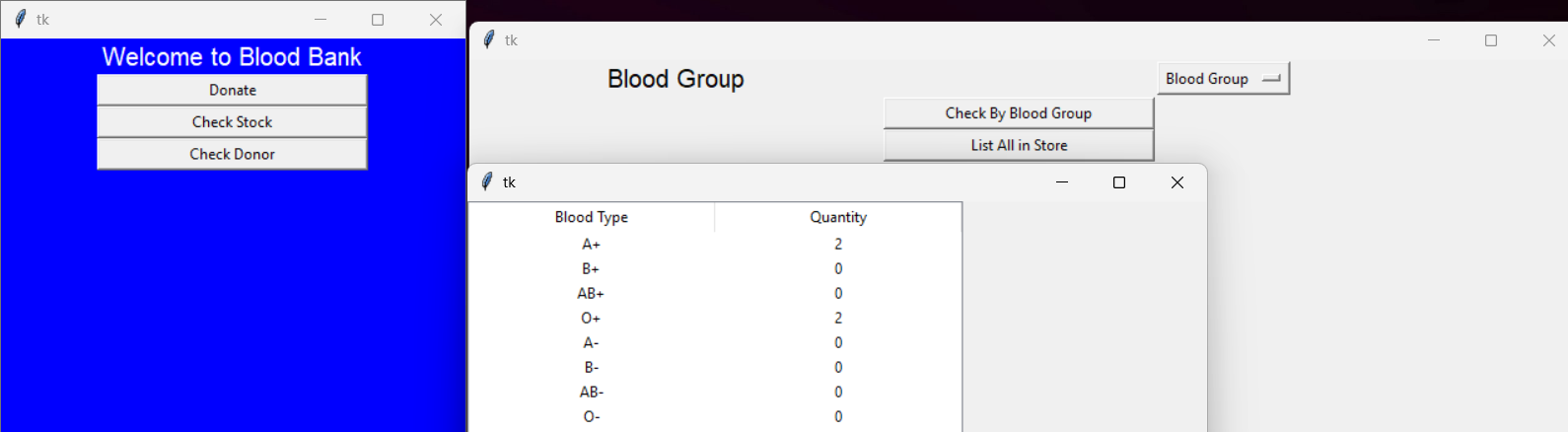
In conclusion, while the provided blood bank system serves as a starting point, there is room for further development and refinement to meet the specific needs and requirements of a real-world blood bank. Additionally, considerations such as security, scalability, and real-time updates should be taken into account for a comprehensive and effective blood bank management system.

**7. Results/Screenshots**

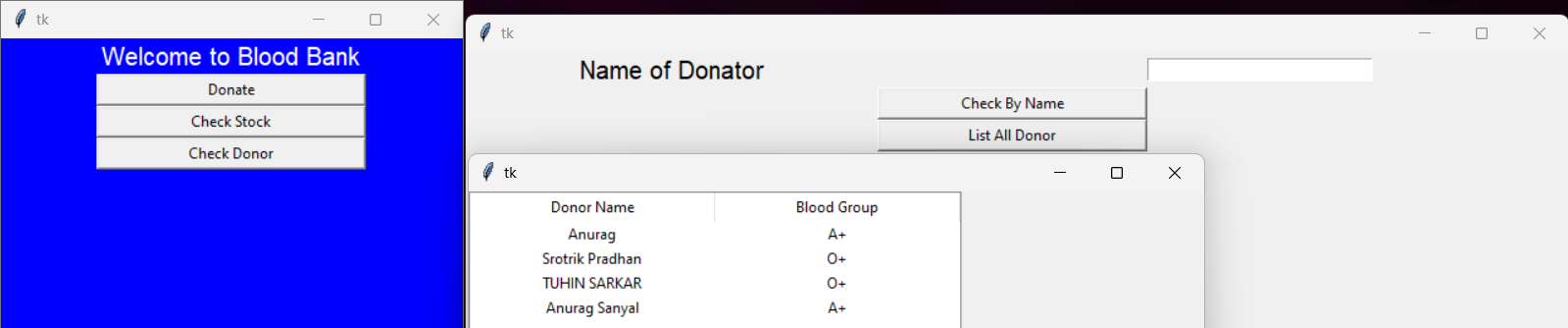
**Adding Donor**

****

**Check blood stock :-**

****

**Check Donor**

****

**8. Conclusion**

An online blood bank system can be a valuable tool for managing blood donation and transfusion activities. It can help to improve the efficiency and effectiveness of blood banks by automating many of the tasks involved in the donation process, such as scheduling appointments, tracking donor records, and managing inventory. Additionally, an online blood bank system can make it easier for people to find and schedule blood donation appointments, which can help to increase the number of blood donors.

Benefits of an online blood bank system:

* Increased efficiency and effectiveness of blood banks: An online blood bank system can automate many of the tasks involved in the donation process, such as scheduling appointments, tracking donor records, and managing inventory. This can free up staff time to focus on other important tasks, such as donor education and recruitment.
* Improved communication and coordination: An online blood bank system can improve communication and coordination between blood banks, hospitals, and other healthcare providers. This can help to ensure that blood products are available when and where they are needed.
* Increased access to blood donation information: An online blood bank system can provide potential donors with easy access to information about blood donation, such as eligibility criteria, appointment locations, and the importance of blood donation. This can help to increase the number of blood donors.
* Improved donor retention: An online blood bank system can make it easier for donors to schedule and track their donations. This can help to improve donor retention rates.

Overall, an online blood bank system can be a valuable tool for improving the efficiency, effectiveness, and accessibility of blood donation and transfusion services.

Image of an online blood bank system

**9. References**

**.** Youtube :- ([https:/youtu.beAjgREAoEGCc?si=teiy4hfyA\_tAG0p2](https://youtu.be/AjgREAoEGCc?si=teiy4hfyA_tAG0p2))

Got the idea of how to do the program and the basic flowcharts

. Books :- Pearson

Authors :- Sheetal Taneja

Naveen Kumar

Developed the idea of connecting Python with Mysql (Backend) and the coding part.

. Partner :- Tuhin Sarkar(RA2311003011883)

.github :- <https://github.com/topics/blood-bank-management>

Helped in do the front end of the program