Python Project Report

Hospital Management System

Prepared by:

Nidhi Roshan Mundhada

Email id: nidhirmundhada@gmail.com

Contact no: 8379926546

Index

Problem Statement

Since hospitals are associated with the lives of common people and their day-to-day routines which have been drastically affected by this covid 19 pandemic, we decided to work on this project.

The manual handling of the record is time consuming and highly prone to error. Observing the continuous increase in the covid 19 cases and number of people visiting the hospital, recording and maintaining all these records is highly unreliable, inefficient and error-prone.

Solution

The project that we have undertaken aims to develop a hospital management system that is clean, user-friendly and multi functional. Development of this application includes a number of fields such that the user feels comfortable and the system appears as dynamic to him.

The project "Hospital management system" includes the following functionalities:

- ❏ Appointments can be fixed by filling only a single interface page
- ❏ All appointments and updates are stored in a database created in mySQL
- ❏ If patients want any updation or deletion it can be done and stored
- ❏ Patients can register themselves online and can edit their profile

Libraries Used

- ❏ SQLite Library
- ❏ Tkinter Library
- ❏ PIL Library

Python Code

```
# import modules
from tkinter import *
from tkinter import ttk
import sqlite3
import tkinter.messagebox
from PIL import Image, ImageTk
import tkinter as tk
from tkinter import font as tkfont

# connect to the database
conn = sqlite3.connect('database.db')

# cursor to move around the databse
c = conn.cursor()

#empty list to later append the ids from the database
ids = []
class SampleApp(tk.Tk):

    def ___init___(self):
```

```
        tk.Tk.__init__(self)
        self.__frame = None
        self.switch_frame(Application)

    def switch_frame(self, frame_class):

        #Destroys current frame and replaces it with a new one.
        new_frame = frame_class(self)
        print(new_frame)
        if self.__frame is not None:
            self.__frame.destroy()
            old = self.__frame
            old.destroy()



        self.__frame = new_frame
        self.__frame.pack()



# tkinter window
class Application(tk.Frame):
    def __init__(self, master):
```

```
    tk.Frame.__init__(self, master)



    button1 = tk.Button(self,text="To UPDATE/DELETE details",

                    command=lambda: master.switch_frame(Updation), font =
('verdana 8 bold'), bg='navy blue', fg='light blue', padx=5, pady=5, relief=RIDGE)

    button1.pack()



    #creating the frames in the master

    self.left = Frame(self, width = 950, height= 720, bg = 'white', padx=27,
pady=5,relief= RIDGE, borderwidth=8)

    self.left.pack(side = LEFT)


    self.right = Frame(self, width = 450, height = 720, bg = 'navy blue',  padx=27,
pady=5,relief= RIDGE, borderwidth=8)

    self.right.pack(side = RIGHT)


    # photo

    img = Image.open(r"C:\Users\Lenovo\Desktop\pic.jpg")

    img=img.resize((200,170),Image.ANTIALIAS)

    self.photo1=ImageTk.PhotoImage(img)

    new_label = Label(self, image = self.photo1)
```

```
new_label.place(x=720,y=440)



#labels for the window

self.heading = Label(self.left, text="            Hospital Appointments      \n
for Covid-19 Patients",width=24,  font=("verdana 37 bold"), fg='navy blue', bg =
'light blue', padx=45, pady=50,relief= RIDGE, borderwidth=8)

self.heading.place(x= -20, y= 20)


# photo

img = Image.open(r"C:\Users\Lenovo\Desktop\logo2.jpg")

img=img.resize((170,150),Image.ANTIALIAS)

self.photo=ImageTk.PhotoImage(img)

new_label = Label(self, image = self.photo)

new_label.place(x=40,y=110)



#patient's name

self.name= Label(self.left, text = "Patient's Name", font = ('arial 18 bold'), fg =
'white', bg ='navy blue', padx=25, pady=5,relief= RIDGE, borderwidth=5)

self.name.place(x=0, y=270)


#age
```

```
self.age= Label(self.left, text = "Patient's Age", font = ('arial 18 bold'), fg = 'white', bg ='navy blue', padx=35, pady=5,relief= RIDGE, borderwidth=5)

self.age.place(x=0, y=330)


#gender

self.gender= Label(self.left, text = "Patient's Gender", font = ('arial 18 bold'), fg = 'white', bg ='navy blue', padx=16, pady=5,relief= RIDGE, borderwidth=5)

self.gender.place(x=0, y=390)



#location

self.location= Label(self.left, text = "Patient's Address", font = ('arial 18 bold'), fg = 'white', bg ='navy blue', padx=11, pady=5,relief= RIDGE, borderwidth=5)

self.location.place(x=0, y=450)


#appointment time

self.time= Label(self.left, text = "Appointment Time", font = ('arial 18 bold'), fg = 'white', bg ='navy blue', padx=5, pady=5,relief= RIDGE, borderwidth=5)

self.time.place(x=0, y=510)


#phone

self.phone= Label(self.left, text = "Phone number", font = ('arial 18 bold'), fg = 'white', bg ='navy blue', padx=27, pady=5,relief= RIDGE, borderwidth=5)
```

```
self.phone.place(x=0, y=570)


#Entries for all the labels==========================================================
self.name_ent = Entry(self.left, font =('arial', 19), width = 20,relief= RIDGE, borderwidth=8)
self.name_ent.place(x=320, y=270 )


self.age_ent = Entry(self.left, font =('arial', 19), width = 20 ,relief= RIDGE, borderwidth=8)
self.age_ent.place(x=320, y=330)


self.gender_ent = Entry(self.left, font =('arial', 19),width = 20 ,relief= RIDGE, borderwidth=8)
self.gender_ent.place(x=320,y=390)


self.location_ent = Entry(self.left, font =('arial', 19), width = 20, relief= RIDGE, borderwidth=8)
self.location_ent.place(x=320, y=450)


self.time_ent = Entry(self.left,font =('arial', 19), width = 20, relief= RIDGE, borderwidth=8)
self.time_ent.place(x=320, y=510)
```

```
    self.phone_ent = Entry(self.left, font =('arial', 19),width = 20 ,relief= RIDGE, borderwidth=8)

    self.phone_ent.place(x=320, y=570)


    #button to perform a command

    self.submit = Button(self.left, text="Add Appointment", font=('verdana 15 bold'), width=14, height=1, fg= 'light blue',bg='navy blue', command=self.add_appointment , padx=4, pady=10,relief= RIDGE, borderwidth=10)

    self.submit.place(x=670, y=580)


    sql2 = "SELECT ID FROM appointments "

    self.result = c.execute(sql2)

    for self.row in self.result:

        self.id = self.row[0]

        ids.append(self.id)


    #ordering the ids

    self.new = sorted(ids)

    self.final_id = self.new[len(ids)-1]


    #displaying the logs in our right frame
```

```
    self.logs = Label(self.right, text="Appointment \nLogs", font=('verdana 22 bold'), fg = 'white', bg = 'navy blue')

    self.logs.place(x=60, y=0)


    self.box = Text(self.right, width=29, height=19, bg='white', fg='black', font=('arial 18 bold'))

    self.box.place(x=-17, y=95)

    self.box.insert(END, "These are the total \nappointments till now : " + str(self.final_id))


    #function to call when the submit button is clicked

  def add_appointment(self):

    #getting the user inputs

    self.val1 = self.name_ent.get()

    self.val2 = self.age_ent.get()

    self.val3 = self.gender_ent.get()

    self.val4 = self.location_ent.get()

    self.val5 = self.time_ent.get()

    self.val6 = self.phone_ent.get()


    #checking if the user input is empty

    if self.val1 == '' or self.val2 == '' or self.val3 == '' or self.val4 == '' or self.val5 == '' or self.val6 == '':
```

```
        tkinter.messagebox.showinfo("Warning", "Please fill up all boxes")
    else:
        #Now we add to the database
        sql = "INSERT INTO 'appointments' (name, age, gender, location, scheduled_time, phone) VALUES(?,?, ?, ?, ?, ?)"
        c.execute(sql, (self.val1, self.val2, self.val3, self.val4, self.val5, self.val6))
        conn.commit()
        tkinter.messagebox.showinfo("Success", "Appointment for " +str(self.val1) + " has been created")


        #getting the number of appointments fixed to view in the log



        self.box.insert(END, '\nAppointment fixed for  ' + str(self.val1) + ' at ' + str(self.val5))



class Updation(tk.Frame):
  def ___init___(self, master):
    tk.Frame.___init___(self, master)


    # heading label
```

```
self.heading = Label(master, text="      Update Appointments", width=24,
fg='navy blue', bg='light blue', font=('Verdana 37 bold'), padx=45,
pady=45,relief= RIDGE, borderwidth=8)

self.heading.place(x=150, y=10)



img = Image.open(r"C:\Users\Lenovo\Desktop\logo2.jpg")

img=img.resize((150,130),Image.ANTIALIAS)

self.photo=ImageTk.PhotoImage(img)

new_label = Label( image = self.photo)

new_label.place(x=170,y=25)


img = Image.open(r"C:\Users\Lenovo\Desktop\guide.png")

img=img.resize((400,400),Image.ANTIALIAS)

self.photo1=ImageTk.PhotoImage(img)

new_label = Label( image = self.photo1)

new_label.place(x=860,y=180)


img = Image.open(r"C:\Users\Lenovo\Desktop\shss3.png")

img=img.resize((200,150),Image.ANTIALIAS)

self.photo2=ImageTk.PhotoImage(img)

new_label = Label( image = self.photo2)

new_label.place(x=1080,y=25)
```

```
    # search criteria -->name
    self.name = Label(master, text="Enter Patient's Name", fg='white', bg='navy blue',font =('arial 18 bold'), width = 18,padx=25, pady=5,relief= RIDGE, borderwidth=8)
    self.name.place(x=60, y=200)

    # entry for  the name
    self.namenet = Entry(master, width=20, font =('arial', 19),relief= RIDGE, borderwidth=8)
    self.namenet.place(x=500, y=200)

    # search button
    self.search = Button(master, text="Search",font =('arial 18 bold'), width = 12,bg='light blue', fg='navy blue', relief= RIDGE, borderwidth=8 , command=self.search_db)
    self.search.place(x=350, y=260)

  # function to search
  def search_db(self):
    self.input = self.namenet.get()

    # execute sql
```

```
sql = "SELECT * FROM appointments WHERE name LIKE ?"
self.res = c.execute(sql, (self.input,))
for self.row in self.res:
    self.name1 = self.row[1]
    self.age = self.row[2]
    self.gender = self.row[3]
    self.location = self.row[4]
    self.time = self.row[6]
    self.phone = self.row[5]

# creating the update form
self.uname = Label(self.master, text="Patient's Name", font = ('arial 18 bold'), fg = 'white', bg ='navy blue', width = 18, padx=25, pady=5,relief= RIDGE, borderwidth=5)
self.uname.place(x=60, y=330)

self.uage = Label(self.master, text="Patient's Age", font = ('arial 18 bold'), fg = 'white', bg ='navy blue', width = 18, padx=25, pady=5,relief= RIDGE, borderwidth=5)
self.uage.place(x=60, y=386)

self.ugender = Label(self.master, text="Patient's Gender",font = ('arial 18 bold'), fg = 'white', bg ='navy blue', width = 18, padx=25, pady=5,relief= RIDGE, borderwidth=5)
```

```
self.ugender.place(x=60, y=442)


self.ulocation = Label(self.master, text="Patient's address", font = ('arial 18 bold'), fg = 'white', bg ='navy blue', width = 18, padx=25, pady=5,relief= RIDGE, borderwidth=5)

self.ulocation.place(x=60, y=498)


self.utime = Label(self.master, text="Appointment Time", font = ('arial 18 bold'), fg = 'white', bg ='navy blue', width = 18, padx=25, pady=5,relief= RIDGE, borderwidth=5)

self.utime.place(x=60, y=554)


self.uphone = Label(self.master, text="Patient's Contact no", font = ('arial 18 bold'), fg = 'white', bg ='navy blue', width = 18, padx=25, pady=5,relief= RIDGE, borderwidth=5)

self.uphone.place(x=60, y=610)


# entries for each labels==========================================================

#==============================================filling the search result in the entry box to update

self.ent1 = Entry(self.master, width=20,  font =('arial', 19),relief= RIDGE, borderwidth=8)

self.ent1.place(x=500, y=330)
```

```
self.ent1.insert(END, str(self.name1))


self.ent2 = Entry(self.master, width=20,  font =('arial', 19),relief= RIDGE,
borderwidth=8)

self.ent2.place(x=500, y=386)

self.ent2.insert(END, str(self.age))


self.ent3 = Entry(self.master, width=20,  font =('arial', 19),relief= RIDGE,
borderwidth=8)

self.ent3.place(x=500, y=442)

self.ent3.insert(END, str(self.gender))


self.ent4 = Entry(self.master, width=20,  font =('arial', 19),relief= RIDGE,
borderwidth=8)

self.ent4.place(x=500, y=498)

self.ent4.insert(END, str(self.location))


self.ent5 = Entry(self.master, width=20,  font =('arial', 19),relief= RIDGE,
borderwidth=8)

self.ent5.place(x=500, y=554)

self.ent5.insert(END, str(self.time))


self.ent6 = Entry(self.master, width=20,  font =('arial', 19),relief= RIDGE,
borderwidth=8)
```

```
        self.ent6.place(x=500, y=610)

        self.ent6.insert(END, str(self.phone))


        # button to execute update

        self.update = Button(self.master, text="Update",font =('arial 18 bold'), width
= 12,bg='navy blue', fg='light blue', relief= RIDGE, borderwidth=8 ,
command=self.update_db)

        self.update.place(x=820, y=610)


        # button to delete

        self.delete = Button(self.master, text="Delete", font =('arial 18 bold'), width =
12,bg='red', fg='light blue', relief= RIDGE, borderwidth=8 ,
command=self.delete_db)

        self.delete.place(x=1050, y=610)




    def update_db(self):

        # declaring the variables to update

        self.var1 = self.ent1.get() #updated name

        self.var2 = self.ent2.get() #updated age

        self.var3 = self.ent3.get() #updated gender

        self.var4 = self.ent4.get() #updated location
```

```
        self.var5 = self.ent5.get() #updated phone
        self.var6 = self.ent6.get() #updated time



        query = "UPDATE appointments SET name=?, age=?, gender=?, location=?, scheduled_time=?, phone=? WHERE name LIKE ?"
        c.execute(query, (self.var1, self.var2, self.var3, self.var4, self.var5, self.var6, self.namenet.get(),))
        conn.commit()
        tkinter.messagebox.showinfo("Updated", "Successfully Updated.")


    def delete_db(self):
        # delete the appointment
        sql2 = "DELETE FROM appointments WHERE name LIKE ?"
        c.execute(sql2, (self.namenet.get(),))
        conn.commit()
        tkinter.messagebox.showinfo("Success", "Deleted Successfully")
        self.ent1.destroy()
        self.ent2.destroy()
        self.ent3.destroy()
        self.ent4.destroy()
        self.ent5.destroy()
        self.ent6.destroy()
```

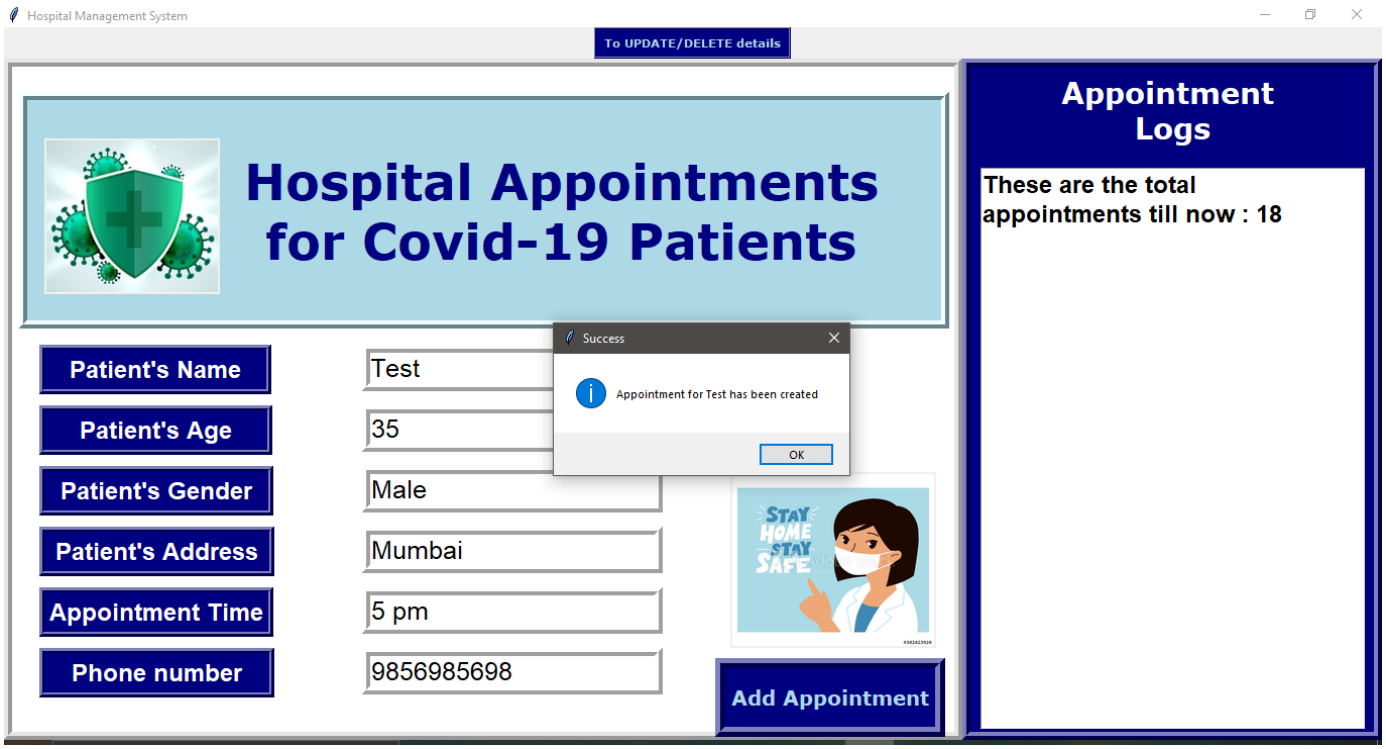
```
# #end the loop
if __name__ == "__main__":
    app = SampleApp()
    app.title("Hospital Management System")
    app.geometry("1350x680+0+0")
    app.resizable(1,1)
    app.mainloop()
```

Output Screenshot

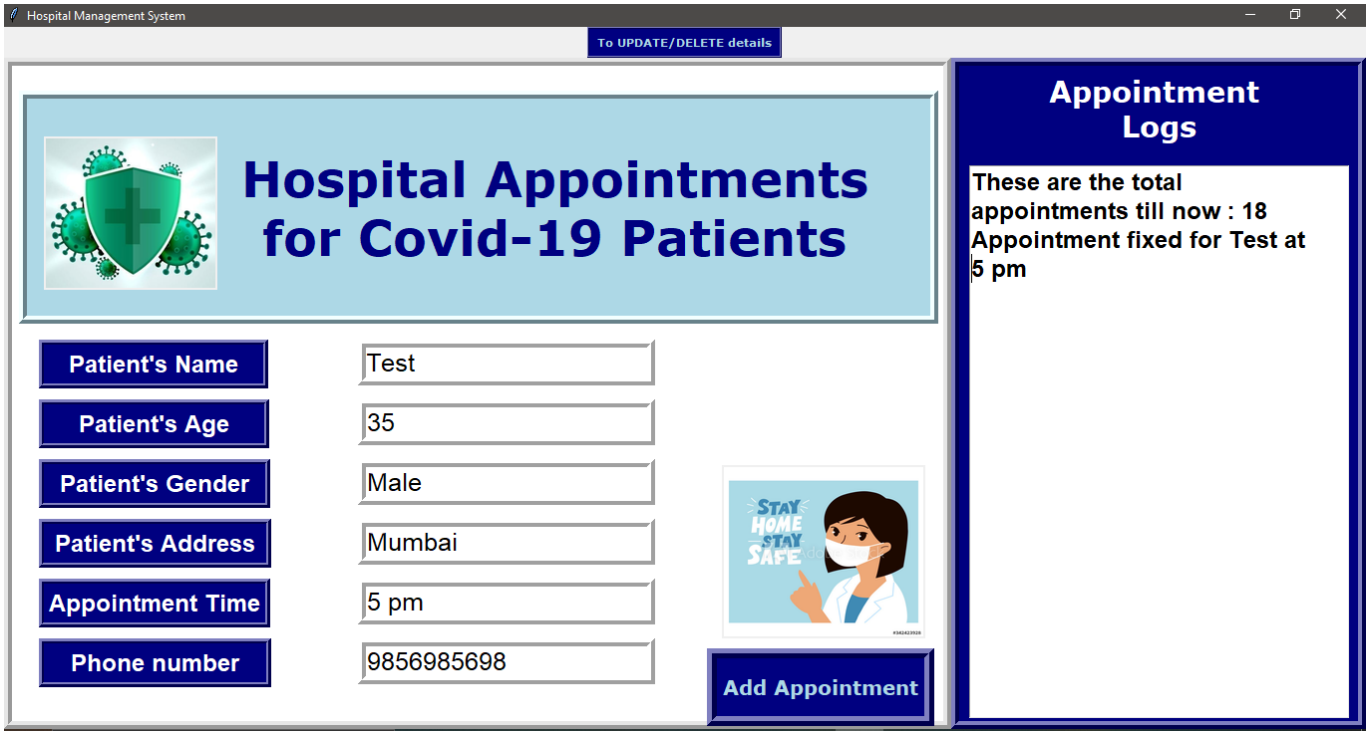
1) First Page

Prepared by: Nidhi Mundhada

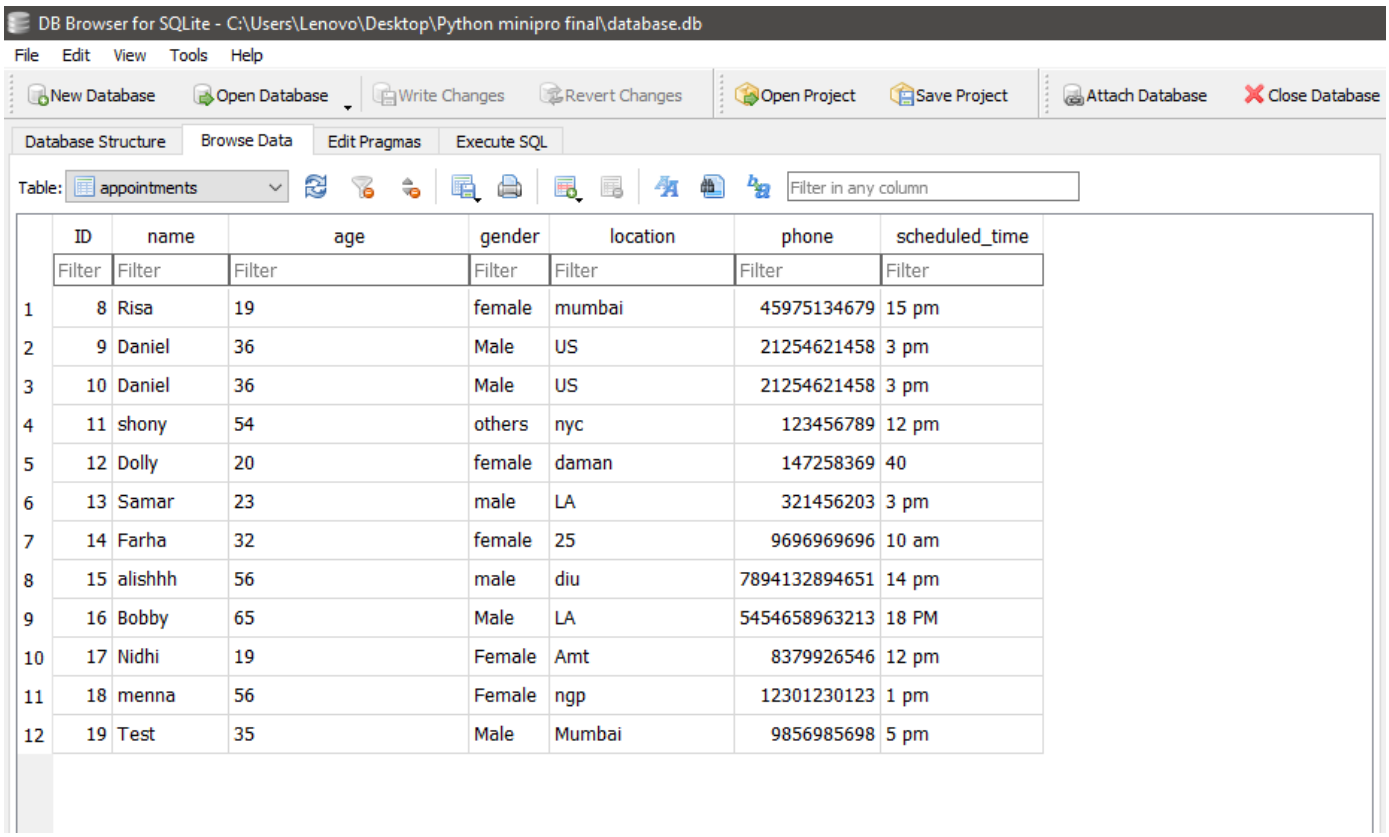
2) **Adding** an Appointment



3) Appointment **Logs** section **updation**

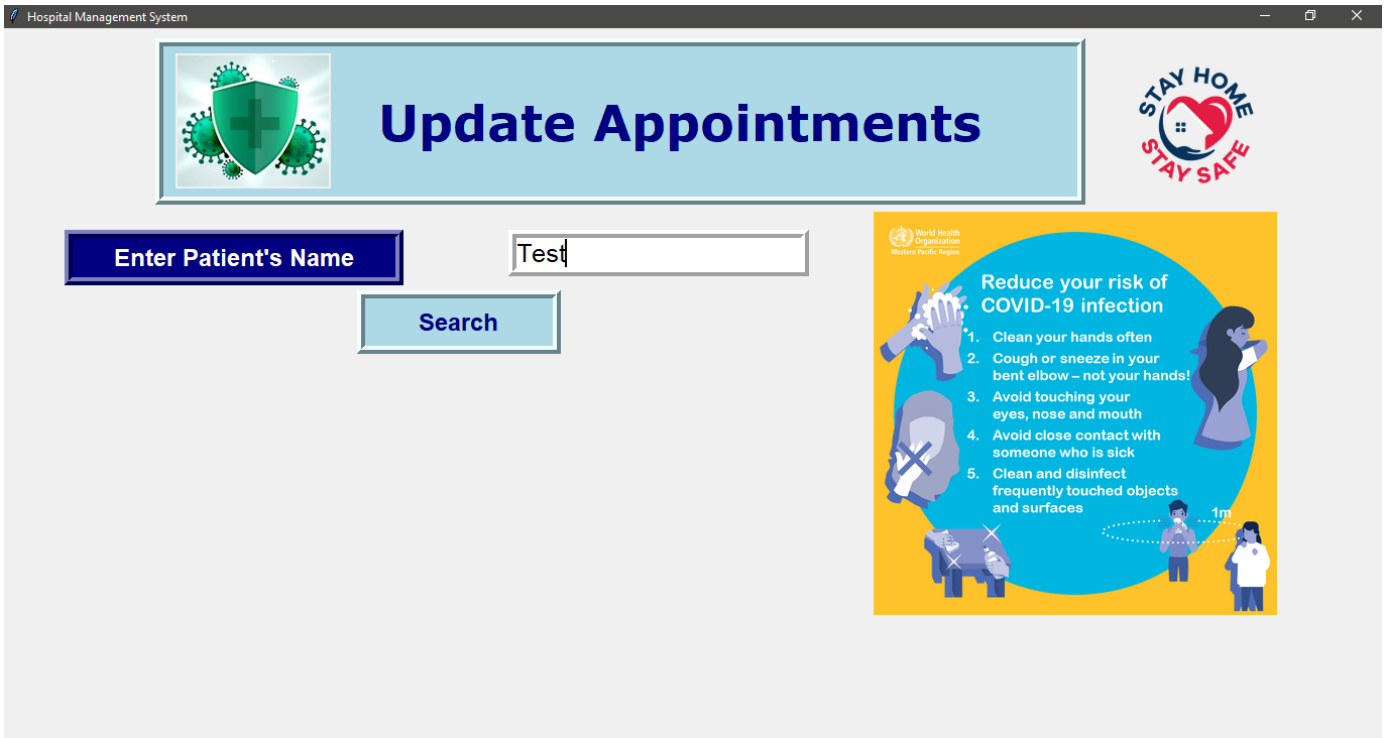


4) Appointment **added** in **database**

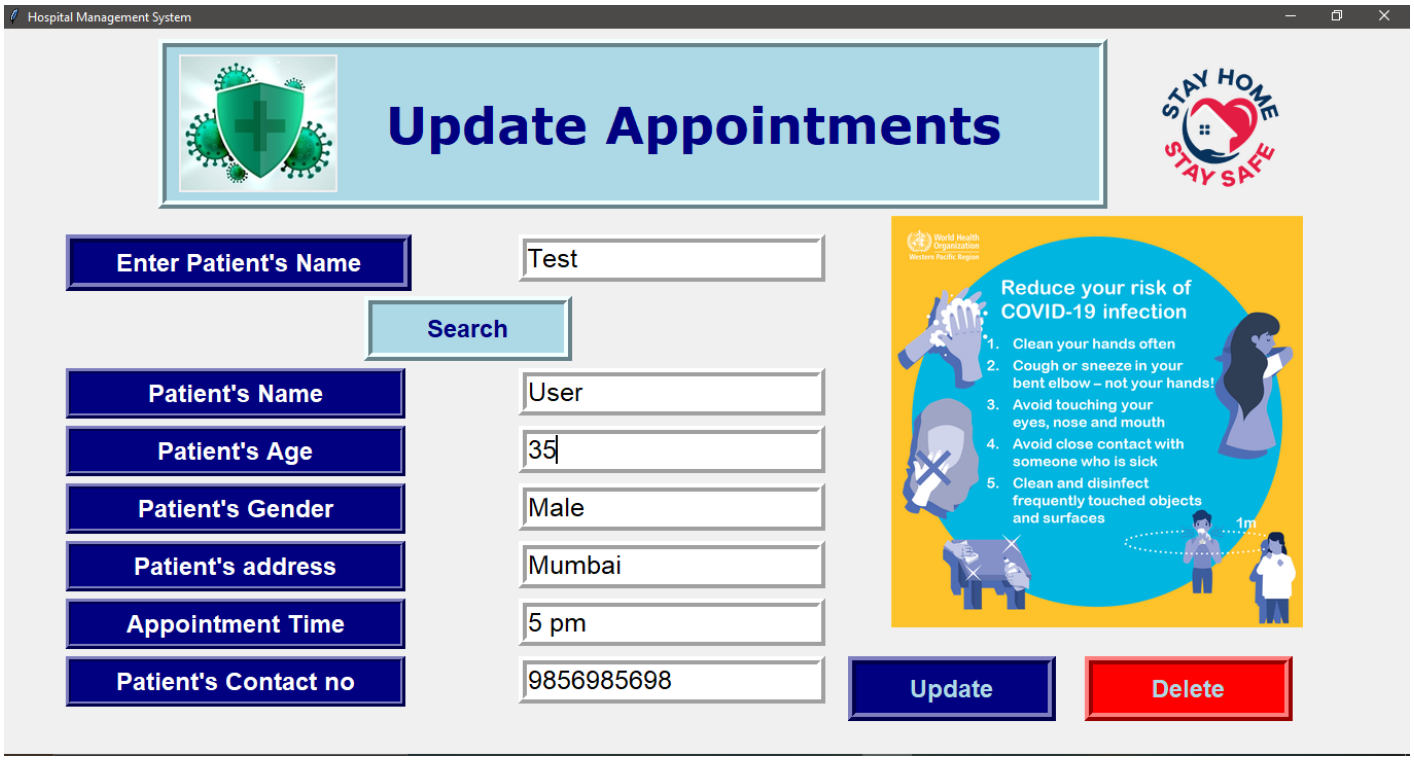


	ID	name	age	gender	location	phone	scheduled_time
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	8	Risa	19	female	mumbai	45975134679	15 pm
2	9	Daniel	36	Male	US	21254621458	3 pm
3	10	Daniel	36	Male	US	21254621458	3 pm
4	11	shony	54	others	nyc	123456789	12 pm
5	12	Dolly	20	female	daman	147258369	40
6	13	Samar	23	male	LA	321456203	3 pm
7	14	Farha	32	female	25	9696969696	10 am
8	15	alishhh	56	male	diu	7894132894651	14 pm
9	16	Bobby	65	Male	LA	5454658963213	18 PM
10	17	Nidhi	19	Female	Amt	8379926546	12 pm
11	18	menna	56	Female	ngp	12301230123	1 pm
12	19	Test	35	Male	Mumbai	9856985698	5 pm

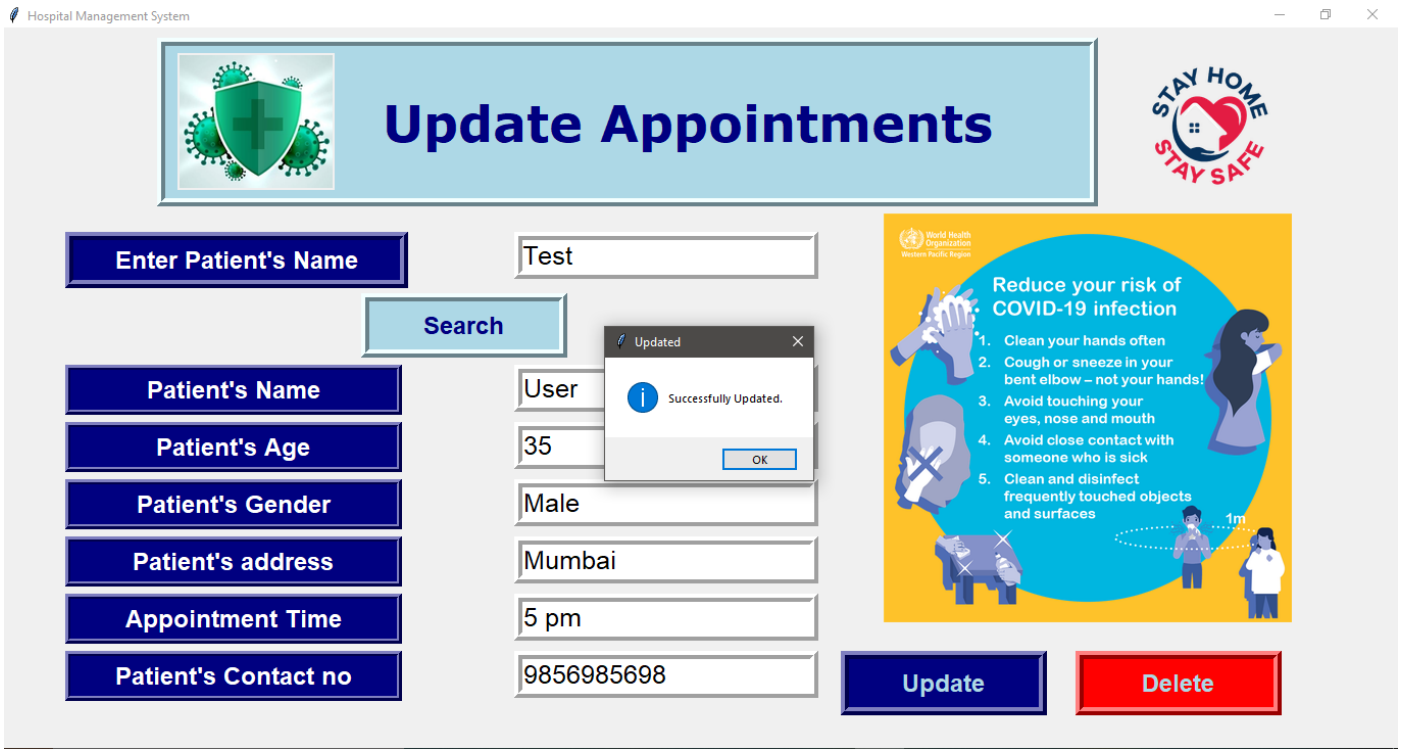
5) **Updating** already **created** appointments.



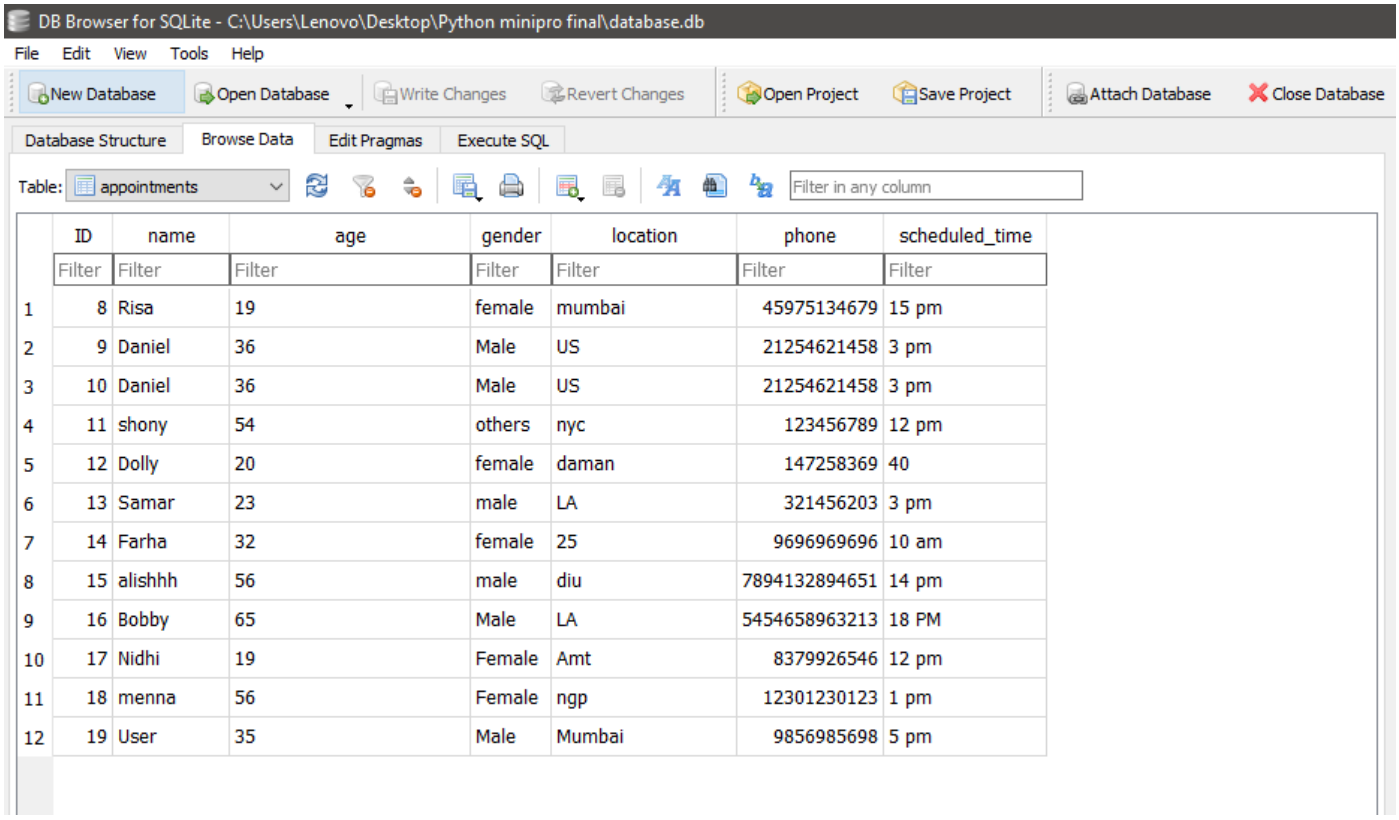
6) To **update** patient details.



7) Successful **updation message**

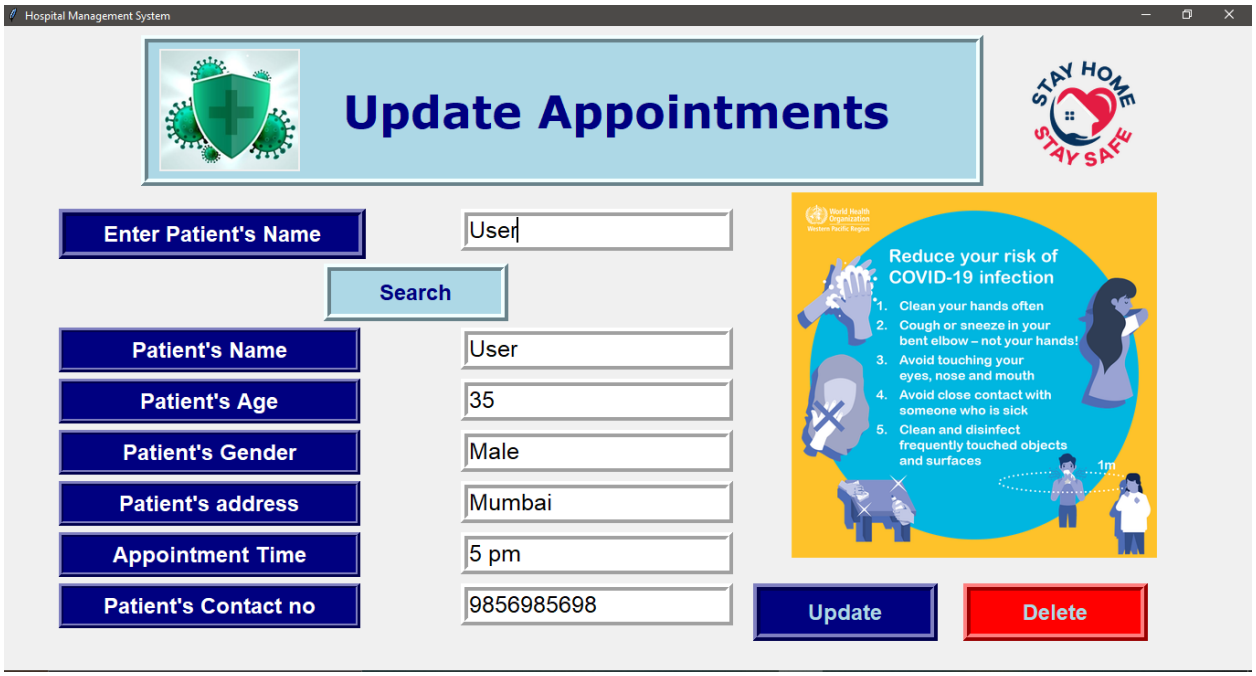


8) Successful **Updation** of patient's details in **database**

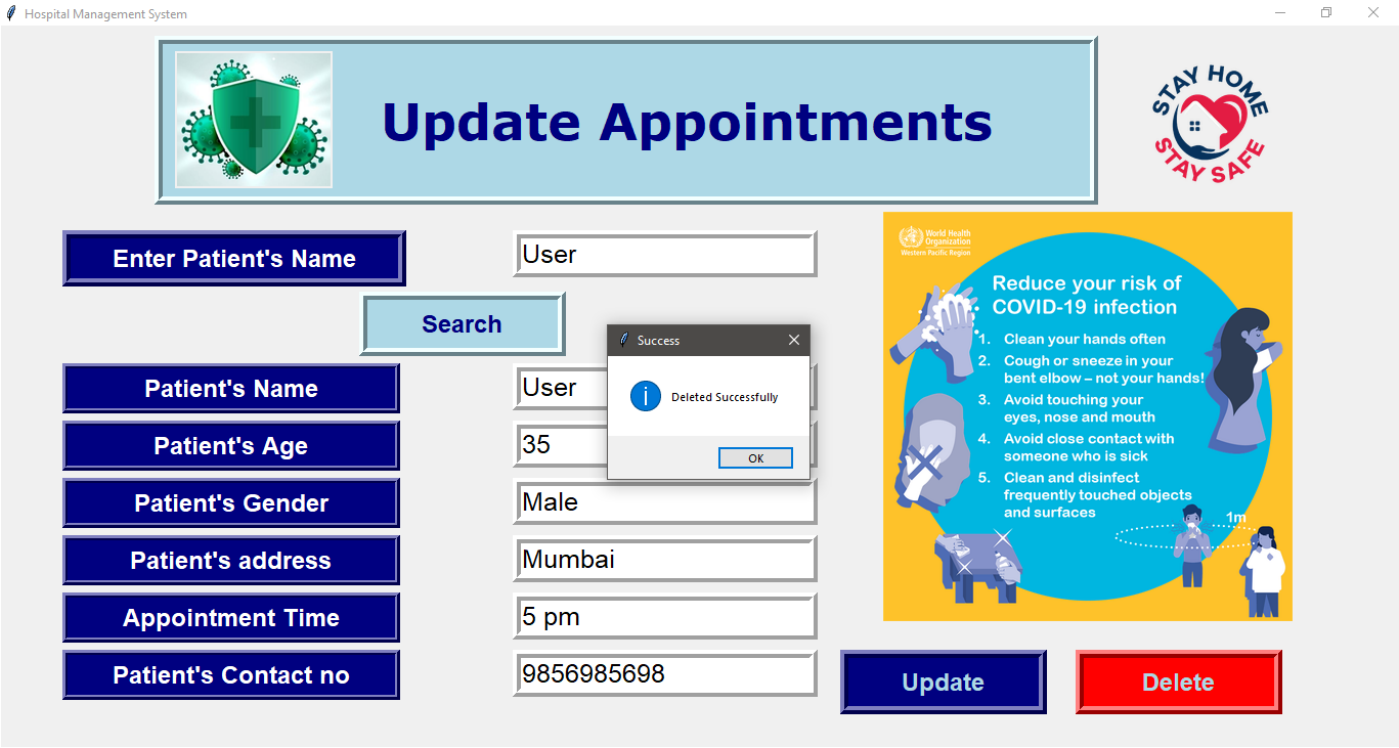


	ID	name	age	gender	location	phone	scheduled_time
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	8	Risa	19	female	mumbai	45975134679	15 pm
2	9	Daniel	36	Male	US	21254621458	3 pm
3	10	Daniel	36	Male	US	21254621458	3 pm
4	11	shony	54	others	nyc	123456789	12 pm
5	12	Dolly	20	female	daman	147258369	40
6	13	Samar	23	male	LA	321456203	3 pm
7	14	Farha	32	female	25	9696969696	10 am
8	15	alishhh	56	male	diu	7894132894651	14 pm
9	16	Bobby	65	Male	LA	5454658963213	18 PM
10	17	Nidhi	19	Female	Amt	8379926546	12 pm
11	18	menna	56	Female	ngp	12301230123	1 pm
12	19	User	35	Male	Mumbai	9856985698	5 pm

9) To **delete** patient's details

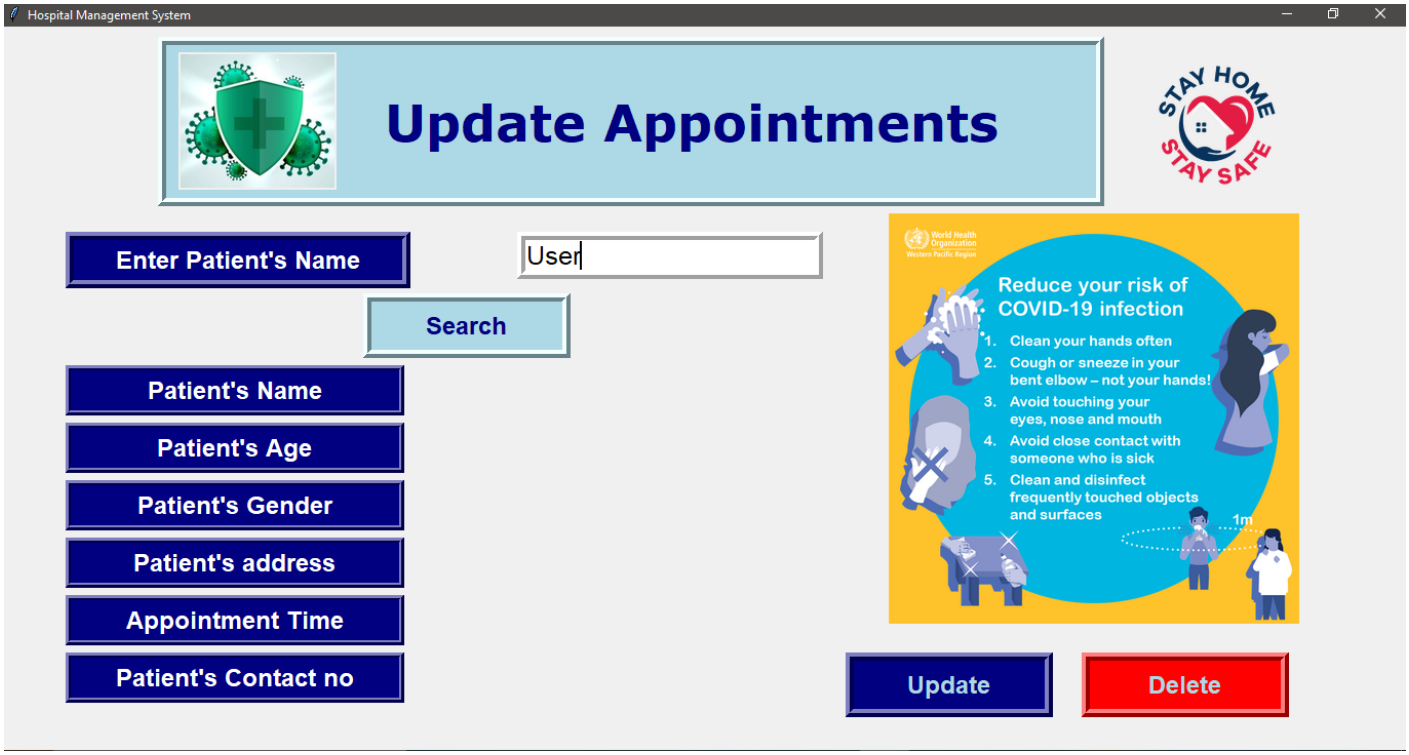


10) Successful **deletion message** of patient's details

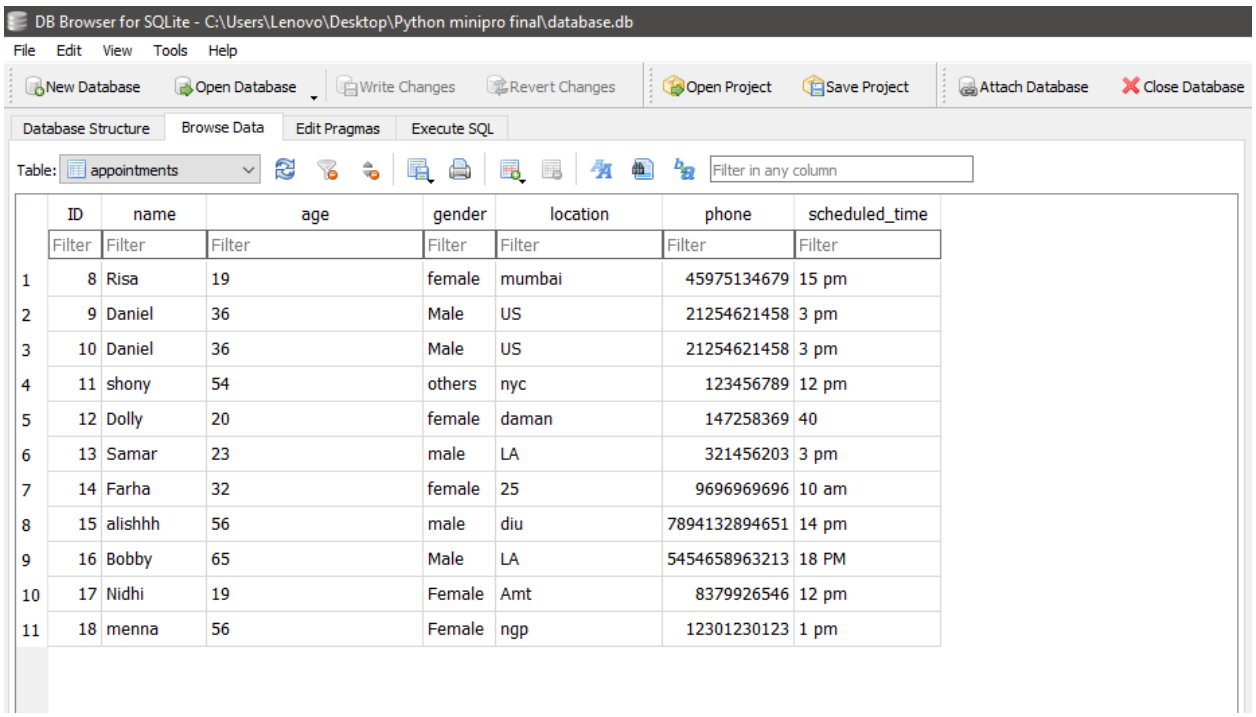


Prepared by: Nidhi Mundhada

11) After **deletion, entries** get **destroyed** automatically.



12) **Database** updation after **deletion**.



	ID	name	age	gender	location	phone	scheduled_time
	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	8	Risa	19	female	mumbai	45975134679	15 pm
2	9	Daniel	36	Male	US	21254621458	3 pm
3	10	Daniel	36	Male	US	21254621458	3 pm
4	11	shony	54	others	nyc	123456789	12 pm
5	12	Dolly	20	female	daman	147258369	40
6	13	Samar	23	male	LA	321456203	3 pm
7	14	Farha	32	female	25	9696969696	10 am
8	15	alishhh	56	male	diu	7894132894651	14 pm
9	16	Bobby	65	Male	LA	5454658963213	18 PM
10	17	Nidhi	19	Female	Amt	8379926546	12 pm
11	18	menna	56	Female	ngp	12301230123	1 pm

Conclusion

Since we are entering details of the patients electronically in the Hospital Management System, data will be secured. Using this application we can retrieve a patient's details with a single click. Thus processing information will be faster. It guarantees accurate maintenance of Patient details. It easily reduces the book keeping task and thus reduces the human effort and increases accuracy speed.

The main aim of our project is to provide a paperless hospital management system up to 90%.