Name: - JITENDRA SAH

Usn: - 1NH17CS059

MEDICINE PRESCRIPTION

CHAPTER 1

INTRODUCTION

1.1 COURSE OBJECTIVES

Computer technology has been proposed to solve the different medical problems in the medical fields. Medicine prescription is one of the medical fields where people are facing different problems. To solve this problem this program is meant to developed.

It is a user-interactive program which is used for simplifying the task of the doctors. It will simply check the patient and suggest the suitable prevention for the same. At first my programs will welcome the patient. Then it will ask the valid user-id and required password to operate the application. If the user-id and password will be matched then only user can go ahead and if not then the user can create a new user-id and password then only he/she can access the application. The above requirement is fulfilled then user have to give the patient full details like patient name, age, gender, mobile_no and email address and these details will be stored in the application databases for further enquiry. After that there is some list of symptoms of diseases the user can select their symptoms and the user can see the prevention

1.2 PROBLEM DEFINITION

People in this rural area are poor that they can't achieve the medical facility in time. Due to this they will die before their time. This program is designed for all of those poor people all around the world where they can't get their basic medical facility. Doctors cost much more for diagnosis a disease. It will be difficult for a poor and middle-class family to pay for those facility. In order to overcome this, I have designed and developed this program. People may need to travel long way to get their basic medical facility, which is also one of the great problems. In this world there are many numbers of patient. According to WHO (World Health Organization) 100 patients are allocated for a single doctor so, a doctor can't give proper time for a patient.

1.3 MOTIVATION OF THE PROJECT

The main of the project are discussed below:

- People in the India as well as people of other developing country still not getting proper medical facility. This program is developed to reach their medical needs and facilities.
- This project is designed to create awareness in people about the requirement of the good health.
- It will enhance the basic knowledge of the people in the medical field (i.e it will give knowledge on the symptoms and prevention of the basic disease)
- It will motivate people to manage their health and apply the prevention which is suggested by the program.
- It makes the medical field easier and more efficient.
- This is mainly motivated to reach the medical needs of the rural area people.

1.4 Limitations of the project

The main drawback of the program is that it is not designed for diagnosing the fatal diseases. It only designed to diagnosis the small diseases.

This is not purposed or make to be used for the emergency cases in the hospitals.

CHAPTER 2

REQUIREMENT SPECIFICATION

2.1 HARDWARE REQUIREMENT

Processor	:	i3 and higher which is X86 Compatible
RAM	:	512 MB or more
Hard disk	:	500 GB

2.2 SOFTWARE REQUIREMENT

- Python IDLE, Jupyter notebook
- Windows 7 Operating System or higher

CHAPTER 3

ER-DIAGRAM AND DATABASE ASSUMPTION

3.1 MODULES

In this we will discuss different kind of screens that appear in the program. They are discussed below: -

- Welcoming screen:

  This screen will welcome the Patients or Users. And give a list from which they can choose. The lists are:

  1.Enter User-id and password

  2. Registers for a new User-id and password with details

  3.Entering the patient details

  4.Displaying the patient details

  5. Entering the symptoms and getting the prevention

  6. Visiting doctors if they want

It is compulsory for all User and Patients to enter details at first. By mistake if they didn't enter their details then programs will automatically show the message invalid captcha.

- Symptoms and prevention screen:

The screen contains all the symptoms that is stored in a file. The patient will read the symptoms page and enter the symptoms which they are suffering with. On the basis of the symptoms the program will display the prevention.

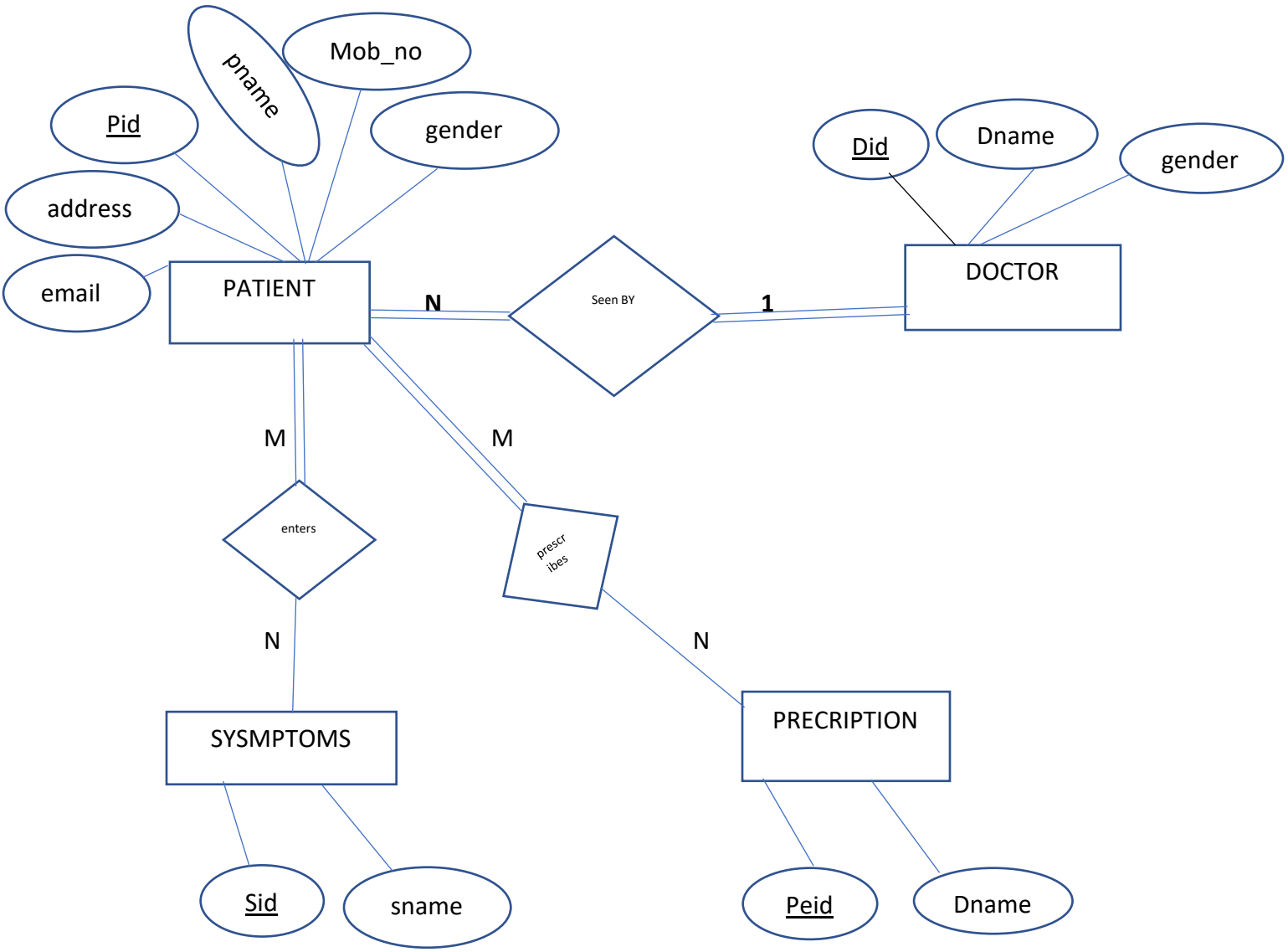
- Visiting doctors:

This will appear when the patients will enter the options next button. If only when the patients are not satisfied with the prevention. The patients can appoint one doctors for treatment with filling time. The system will automatically shows that your prefer doctor name and time to consult with them.

- Display screen

All the patient information which they have entered previously are stored in a database.

3.2 ER-DIAGRAM



3.3 CREATING TABLE

- CREATE TABLE PATIENTS (PID INT PRIMARY KEY AUTOINCREAMENT, PNAME VARCHAR (20) NOT NULL, LNAME VARCHAR (20) NOT NULL, GENDER VARCHAR (10), MOBILE_NO BIGINT UNIQUE KEY, ADDRESS VARCHAR (25), EMAIL VARCHAR (30));

- CREATE TABLE DOCTOR (DID INT PRIMARY KEY AUTOINCREAMENT, DNAME VARCHAR (20) NOT NULL, GENDER VARCHAR (20) CONSTRAINT FK1 FOREIGN KEY DID REFERENCES PATIENT(PID) ON DELETE CASCADE ON UPDATE CASCADE);

- CREATE TABLE SYMPTOMS (SID INT PRIMARY KEY AUTOINCREAMENT, SNAME VARCHAR (20) NOT NULL, CONSTRAINT FK2 FOREIGN KEY SID REFERENCES TO PATIENT(PID) ON DELETE CASCASE ON UPDATE CASCADE, CONSTRAINT FK3 FOREIGN KEY SID REFERENCES TO DOCTOR(DID) ON DELETE CASCADE ON UPDATE CASCASDE);

- CREATE TABLE PRESCRIPTION (PEID INT PRIMARY KEY AUTO INCREAMENT, PNAME VARCHAR (20) NOT NULL, CONSTRAINT FK4 REFERENCES TO PATIENT(PID) ON DELETE CASCADE ON UPDATE CASCADE);

CHAPTER 4

PYTHON FEATURES

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default:

1. String
2. List
3. Tuple
4. Set
5. Dictionary

4.1 STRING

In Python, String is a collection of one or more characters at a time put in single, double or triple quotes. String handling in python is very straightforward task since there are various in-built functions and operators already provided. In case of string handling, the operator + is used to concatenate two strings. For example, the operation "hello"+" python" returns "hello python" as output. The operator * is known as repetition operator as the operation "Python" *2 returns "Python".

```
str1 = 'hello javatpoint' #string str1
str2 = ' how are you' #string str2
print (str1[0:2]) #printing first two character using slice operator
print (str1[4]) #printing 4th character of the string
print (str1*2) #printing the string twice
print (str1 + str2) #printing the concatenation of str1 and str2
```

Output:

```
he
o
hello javatpointhello javatpoint
hello javatpoint how are you
```

Fig. 4.1 String Operations

4.2 LIST

A List in python is an 'ordered collection' of one or more data items not necessarily of the same type put in 'square brackets'. Lists are similar to arrays in C. However, the list can contain data of different types. The items stored in the list are separated with a comma " , " and enclosed within square brackets "[ ]".

We can use slice ' : ' operators to access the data of the list. The concatenation operator (+) and repetition operator ' * ' works with the list in the same way as they were working with the strings.

Consider the following example.

```
l = [1, "hi", "python", 2]
print (l[3:]);
print (l[0:2]);
print (l);
print (l + l);
print (l * 3);
```

Output:

```
[2]
[1, 'hi']
[1, 'hi', 'python', 2]
[1, 'hi', 'python', 2, 1, 'hi', 'python', 2]
[1, 'hi', 'python', 2, 1, 'hi', 'python', 2, 1, 'hi', 'python', 2]
```

Fig 4.2 List Operations

4.3 TUPLE

A tuple object is an ordered collection of one or more data items, not necessarily of same data type. A tuple is similar to the list in lots of ways. Like lists, tuples also contain the collection of items of different data types. The items of the tuple are separated by a comma " , " and enclosed in parentheses " ( ) ". A tuple is a read-only data structure as we can't modify the size and value of the items of a tuple.

```
t = ("hi", "python", 2)
print (t[1:]);
print (t[0:1]);
print (t);
print (t + t);
print (t * 3);
print (type(t))
t[2] = "hi";
```

Output:

```
('python', 2)
('hi',)
('hi', 'python', 2)
('hi', 'python', 2, 'hi', 'python', 2)
('hi', 'python', 2, 'hi', 'python', 2, 'hi', 'python', 2)
<type 'tuple'>
Traceback (most recent call last):
  File "main.py", line 8, in <module>
    t[2] = "hi";
TypeError: 'tuple' object does not support item assignment
```

Fig 4.3 Tuple Operations

4.4 SET

Set is an unordered collection of unique items. Set is defined by values separated by comma inside braces " { } ". Items in a set are not ordered. We can perform set operations like union, intersection on two sets. Set accepts only unique values. Since, set are unordered collection, indexing has no meaning. Hence the slicing operator " [ ] " does not work.

```
>>> a = {1,2,2,3,3,3}
>>> a
{1, 2, 3}
```

Fig 4.4 Set Operations

4.5 DICTIONARY

Dictionary is an ordered set of a key-value pair of items. It is like an associative array or a hash table where each key stores a specific value. It is mostly used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key in order to retrieve the value.

```
d = {1:'Jimmy', 2:'Alex', 3:'john', 4:'mike'};
print("1st name is "+d[1]);
print("2nd name is "+ d[4]);
print (d);
print (d.keys());
print (d.values());
```

Output:

```
1st name is Jimmy
2nd name is mike
{1: 'Jimmy', 2: 'Alex', 3: 'john', 4: 'mike'}
[1, 2, 3, 4]
['Jimmy', 'Alex', 'john', 'mike']
```

Fig 4.5 Dictionary Operations

CHAPTER 5

TKINTER WIDGETS

Tkinter is standard GUI library for Python. When combined with Tkinter, Python provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit. Some of the GUI Widgets provided by Tkinter are as follows:

1. Button
2. Label
3. Entry
4. Menu
5. Message box

5.1 BUTTON

The Button widget is used to add buttons in your application. The general syntax is: "w=Button (master, option=value)" , master is the parameter used to represent the parent window. For Example:

Output:

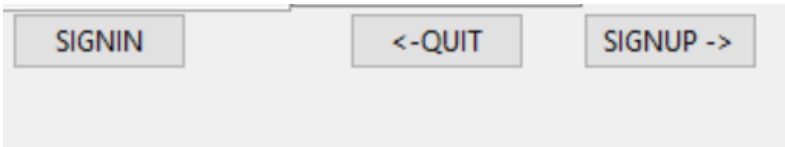


Fig 5.1 Button

5.2 LABEL

The Label widget is used to provide a single-line caption for other widgets. It refers to the display box where you can put any text or image which can be updated any time as per the code. The general syntax is:

"w=Label(master, option=value)" , master is the parameter used to represent the parent window. Example:

```
label1=Label(frame1,text="UserName",width=20,relief="ridge",font="calibri 12")
label1.grid(row=1,column=1)

label2=Label(frame1,text="Password",width=20,relief="ridge",font="calibri 12")
label2.grid(row=5,column=1)
```

Output:

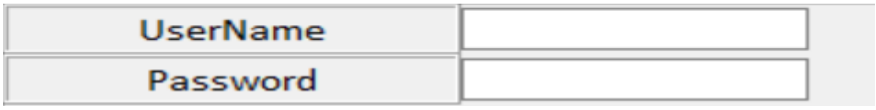


Fig 4.2 Label

5.3 ENTRY

Entry is used to input the single line text entry from the user. For multi-line text input, Text widget is used. The general syntax is:

"w=Entry(master, option=value)" , master is the parameter used to represent the parent window. Example:

```
entry1=ttk.Entry(frame1)   #entry for user
entry1.grid(row=1,column=5)
entry2=ttk.Entry(frame1,show="*")     #entry for password
entry2.grid(row=5,column=5)
```

Output:

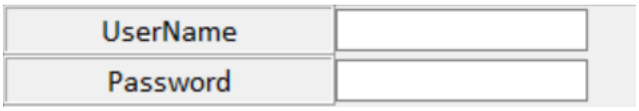


Fig 5.3 Entry Widget

5.4 MENU

The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton. It is a part of top-down menu which stays on the window all the time. Every menubutton has its own functionality. The general syntax is:

“w = MenuButton(master, option=value)” , master is the parameter used to represent the parent window. Menu is of three types: Top Level, Pull Down and Pull Up.

Example:

```
top = Tk()
mb =  Menubutton ( top, text = &quot;GfG&quot;)
mb.grid()
mb.menu  =  Menu ( mb, tearoff = 0 )
mb[&quot;menu&quot;]  =  mb.menu
cVar  = IntVar()
aVar = IntVar()
mb.menu.add_checkbutton ( label ='Contact', variable = cVar )
mb.menu.add_checkbutton ( label = 'About', variable = aVar )
mb.pack()
top.mainloop()
```

Output:

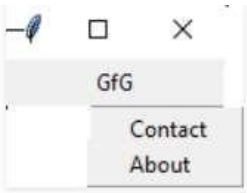


Fig 5.4 Menu button

5.5 MESSAGE BOX

The Message widget is used to display multiline text fields for accepting values from a user. Its work is same as Label. The general syntax is:

“w = Message(master, option=value)” , master is the parameter used to represent the parent window.

Example:

```
def msg():
    messagebox.showinfo("Information","Files have been organized!")
```

Output:

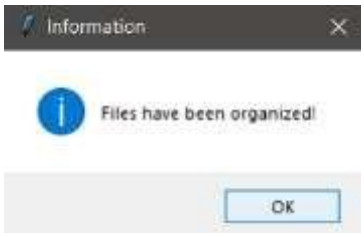


Fig 5.5 Messagebox

CHAPTER 6

IMPLEMENTATION

This mini project has been implemented using Python Programming Language. The code for the following project is:

```
from tkinter import *
from tkinter import ttk      #displaying its to be better
import sqlite3
from tkinter import messagebox

root=Tk()
root.geometry("500x500+100+100")


# working with frame to display output in same window

frame1=Frame(root)
frame2=Frame(root)
frame3=Frame(root)
frame4=Frame(root)
frame5=Frame(root)
frame6=Frame(root)

#to store the value that is entered in the ENTRY box
str1=StringVar()
str2=StringVar()

#funciton to raise the tkinter or frame window
def dashboard(frame):
    frame.tkraise()
```

```
#looping throug frame
for frame in (frame1,frame2,frame3,frame4,frame5,frame6):
    frame.grid(row=0,column=0,sticky='news')

#connecting to the sql data bse
db = sqlite3.connect("E:\mprescribe.db")
cursor1 =db.cursor()

#creating the cursor

#cursor1.execute("DROP TABLE DETAILS")
#cursor1.execute("DROP TABLE USER")
#cursor1.execute("DROP TABLE SYMPTOMS")
#cursor1.execute("DROP TABLE TEXTSEARCH")

#creatingn the table USER
cursor1.execute("CREATE TABLE IF NOT EXISTS 'USER'(UID INTEGER PRIMARY KEY AUTOINCREMENT,UNAME VARCHAR(20) NOT NULL,PASSWORD
db.commit()


label1=Label(frame1,text="UserName",width=20,relief="ridge",font="calibri 12")
label1.grid(row=1,column=1)

label2=Label(frame1,text="Password",width=20,relief="ridge",font="calibri 12")
label2.grid(row=5,column=1)
```

```
entry1=ttk.Entry(frame1)   #entry for user
entry1.grid(row=1,column=5)

entry2=ttk.Entry(frame1,show="*")     #entry for password
entry2.grid(row=5,column=5)


#function to check that user we enter is in the dataabse and its password is correct.also to raise the frame2
def dashboard2(frame):
    x=entry1.get()
    y=entry2.get()
    if(x=="" or y=="" ):
        messagebox.showinfo("ERROR","PLEASE ENTER ALL THE REQUIRED FIELD")
    else:
        cursor1.execute("SELECT * FROM USER")
        records = cursor1.fetchall()   #fetching all the data from USER table
        temp=0     #temp as temporary variable
        for row in records:
            if(x==row[1] and y==row[2]):
                frame.tkraise()
                temp=temp+1
                cursor1.execute("INSERT INTO 'USER'(UNAME,PASSWORD) VALUES(?,?)",(x,y))
                db.commit()

        if(temp==0):
            messagebox.showinfo("ERROR","WRONG INPUT")

#creating button SIGNIN
button1=ttk.Button(frame1,text="SIGNIN",command=lambda:dashboard2(frame2))
```

```
frame2_label1=Label(frame2,text="ENTER PATIENT DETAIL")
frame2_label1.grid(row=5,column=5)

frame2_entry1=ttk.Entry(frame2)
frame2_entry1.grid(row=10,column=5)

frame2_label1=Label(frame2,text="NAME",width=15,relief="solid",font="calibri 12")
frame2_label1.grid(row=10,column=1,sticky="W")

frame2_label2=Label(frame2,text="PHONE_NO",width=15,relief="solid",font="calibri 12")
frame2_label2.grid(row=15,column=1,sticky="W")

frame2_entry2=ttk.Entry(frame2)
frame2_entry2.grid(row=15,column=5)

frame2_label3=Label(frame2,text="EMAIL",width=15,relief="solid",font="calibri 12")
frame2_label3.grid(row=20,column=1,sticky="W")

frame2_entry3=ttk.Entry(frame2)
frame2_entry3.grid(row=20,column=5)

frame2_label4=Label(frame2,text="AGE",width=15,relief="solid",font="calibri 12")
frame2_label4.grid(row=25,column=1,sticky="W")

frame2_entry4=ttk.Entry(frame2)
frame2_entry4.grid(row=25,column=5)

frame2_label5=Label(frame2,text="GENDER M/F",width=15,relief="solid",font="calibri 12")
frame2_label5.grid(row=30,column=1,sticky="W")
```

```
def dashboard5(frame):
    x=frame2_entry1.get()
    y=frame2_entry2.get()
    z=frame2_entry3.get()
    a=frame2_entry4.get()
    if(x=="" or y=="" or z=="" or a==""):
        messagebox.showinfo("ERROR","PLEASE ENTER ALL THE REQUIRED FIELD")
    else:
        frame.tkraise()
        cursor1.execute("INSERT INTO 'USER'(UNAME,PASSWORD) VALUES(?,?)",(x,y))
        db.commit()
        frame_label1=Label(frame,text="WELCOME TO MEDICINE PRESCRIBTION")
        frame_label1.grid(row=1,column=5)

        frame_button1=Button(frame,text="BACK",command=lambda:dashboard(frame2))
        frame_button1.grid(row=5,column=1)

        f = open("symptom.txt", "r")    #reading a file from text file
        x=19
        y=1
        count=1
        scrollbar = Scrollbar(ttk)
        scrollbar.pack(side=RIGHT,fill=Y)

        mylsit=Listbox(root,yscrollcommand=scrollbar.set,width=80)
        #logic to form the label using loop
        for i in f:
            mylist.insert(END,i)
            frame4_label2=Label(frame4,text=str(count)+" "+i)
            frame4_label2.grid(row=x,column=y,sticky="W")
```

```
        x= x+5
        count = count+1
        mylist.pack(side=LEFT,fill=BOTH)
        scrollbar.config(command=mylist.yview)



frame2_button2=ttk.Button(frame2,text="ENTER",command=lambda:dashboard5(frame4))
frame2_button2.grid(row=40,column=10)

#defining the farme 4
def frame_Label1():
    for i in range(50):
        frame_Label1(frame,text=i).grid(row=i,column=0)
        frame_Label1(frame,text="my text"+str(i)).grid(row=i,column=1)
        frame_Label1(frame,text="..........").grid(row=i,column=2)



#function to display the prevention table
def click1():
    w1= Tk()
    w1.title("PREVENTION")
    w1.geometry("500x500+100+100")
    b=str1.get()
    list1 = []
    list1 = b.split()
    x=1
    y=1
```

```
w1_label=Label(w1,text="PREVENTION:")
w1_label.grid(row=x ,column=y)
x=x+19

for i in list1:
    if(i==1):
        w1_label1=Label(w1,text="TREAT YOUR ALLERGIES OR PROTECT YOURSELF FROM ENVIRONMENTAL HAZARD AND TRY A HOMEOPATHI
        w1_label1.grid(row=x ,column=y)
        x=x+5

    elif(i==2):
        w1_label2=Label(w1,text="AVOID ITRRITANTS INCLUDING SMOKE AND DUST OR KEEP HYDRAITED BY DRINKING PLENTY OF WATEF
        w1_label2.grid(row=x ,column=y)
        x=x+5

    elif(i==3):
        w1_label2=Label(w1,text="MAINTAIN GOOD HYGINE,AVOID CONTACT WITH SICK PEOPLE,MAKE SURE IMMUNIZATION ARE UPTO DAT
        w1_label2.grid(row=x ,column=y)
        x=x+5

    elif(i==4):
        w1_label2=Label(w1,text="APPLY VICKS ON HEAD OR SLEEP AT LEAST 8 HRS PER DAY AND AVOID STRESS")
        w1_label2.grid(row=x ,column=y)
        x=x+5
    elif(i==5):
        w1_label2=Label(w1,text="EAT SLOWLY,AVAOID HARD TO DIGEST FOOD,USE VICKS THROAT")
        w1_label2.grid(row=x ,column=y)
        x=x+5
    elif(i==6):
        w1_label2=Label(w1,text="DON'T SMOKE,USE PAPER TOWEL OR USE VICKS OR KEEP CLEAN HOUSE SURFACE")
```

```
#making GUI for frame5

frame5_label1=Label(frame5,text="CONSULTING DOCTOR")
frame5_label1.grid(row=1,column=5)

frame5_label1=Label(frame5,text="TIME")
frame5_label1.grid(row=5,column=1,sticky="W")

frame5_entry1=Entry(frame5,textvariable=str2)
frame5_entry1.grid(row=5,column=5)


def click2():
    w2 = Tk()
    w2.title("DETAILS")
    w2.geometry("500x500+100+100")
    w2_label1=Label(w2,text="PATIENT REPORT")
    w2_label1.grid(row=1,column=5)

    cursor1.execute("SELECT * FROM USER")
    records = cursor1.fetchall()   #fetching all the data from USER table
    x=1     #temp as temporary variable
    for row in records:
        w2_label2=Label(w2,text=row)
        w2_label2.grid(row=x+5,column=1)
        db.commit()

    frame5_button1=Button(frame5,text="ENTER",command=click2)
    frame5_button1.grid(row=10,column=1)
```

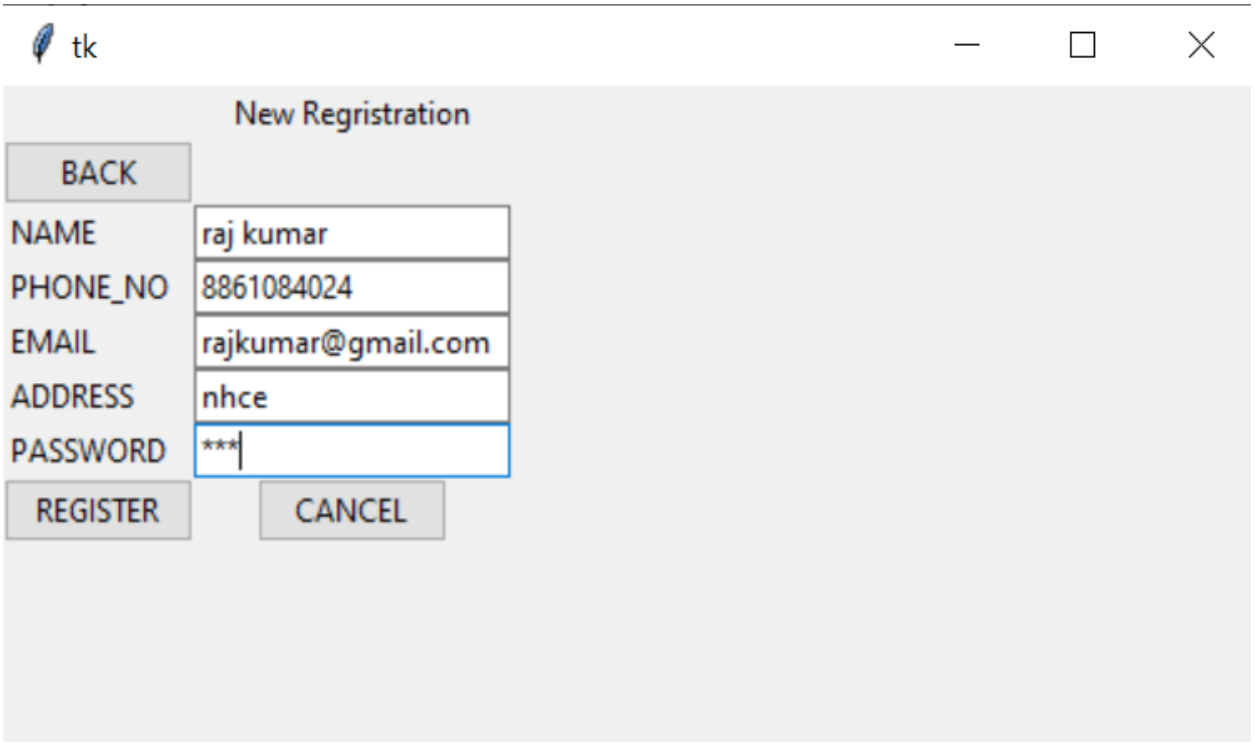
```
    frame4_entry1=Entry(frame4,textvariable=str1)
    frame4_entry1.grid(row=x+5,column=1,sticky="W")

    frame4_button=Button(frame4,text="NEXT",command= click1)
    frame4_button.grid(row=x+10,column=1,sticky="W")

#if we click quit

button2=ttk.Button(frame1,text="<-QUIT",command=lambda:root.destroy())
button2.grid(row=20,column=5)


button3=ttk.Button(frame1,text="SIGNUP ->",command=lambda:dashboard(frame3))
button3.grid(row=20,column=10)

#defining frame3

frame3_label1=Label(frame3,text="New Regristration")
frame3_label1.grid(row=1,column=5)

cursor1.execute("CREATE TABLE IF NOT EXISTS 'DETAILS'(PID INTEGER PRIMARY KEY AUTOINCREMENT,NAME VARCHAR(20) NOT NULL,ADDRES


frame3_button1=ttk.Button(frame3,text="BACK",command=lambda:dashboard(frame1))
frame3_button1.grid(row=5,column=1,sticky="W")

frame3_label1=Label(frame3,text="NAME",width=15,relief="ridge",font="calibri 12")
frame3_label1.grid(row=10,column=1,sticky="W")

frame3_entry1=ttk.Entry(frame3)
```

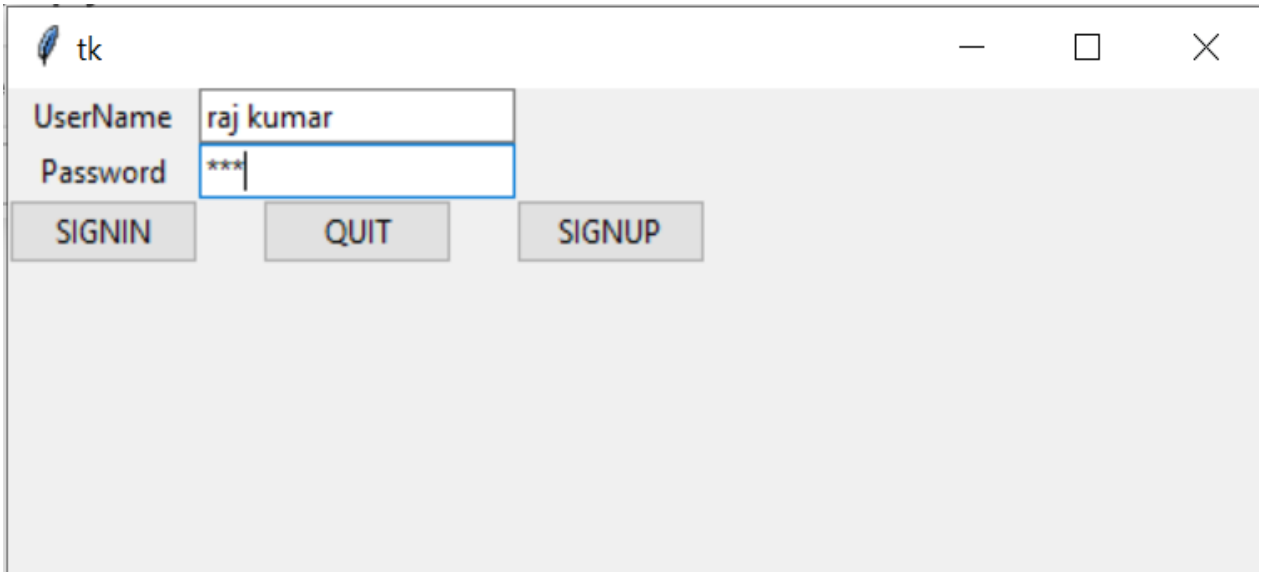
```
frame3_entry2.grid(row=15,column=5)

frame3_label3=Label(frame3,text="EMAIL",width=15,relief="ridge",font="calibri 12")
frame3_label3.grid(row=20,column=1,sticky="W")

frame3_entry3=ttk.Entry(frame3)
frame3_entry3.grid(row=20,column=5)

frame3_label4=Label(frame3,text="ADDRESS",width=15,relief="ridge",font="calibri 12")
frame3_label4.grid(row=25,column=1,sticky="W")

frame3_entry4=ttk.Entry(frame3)
frame3_entry4.grid(row=25,column=5)

frame3_label5=Label(frame3,text="PASSWORD",width=15,relief="ridge",font="calibri 12")
frame3_label5.grid(row=30,column=1,sticky="W")

frame3_entry5=ttk.Entry(frame3,show="*")
frame3_entry5.grid(row=30,column=5)

def dashboard4(frame):
    x=frame3_entry1.get()
    y=frame3_entry2.get()
    z=frame3_entry3.get()
    a=frame3_entry4.get()
    b=frame3_entry5.get()
    if(x=="" or y=="" or z=="" or a=="" or b=="" ):
        messagebox.showinfo("ERROR","PLEASE ENTER ALL THE FIELD")

    else:
```

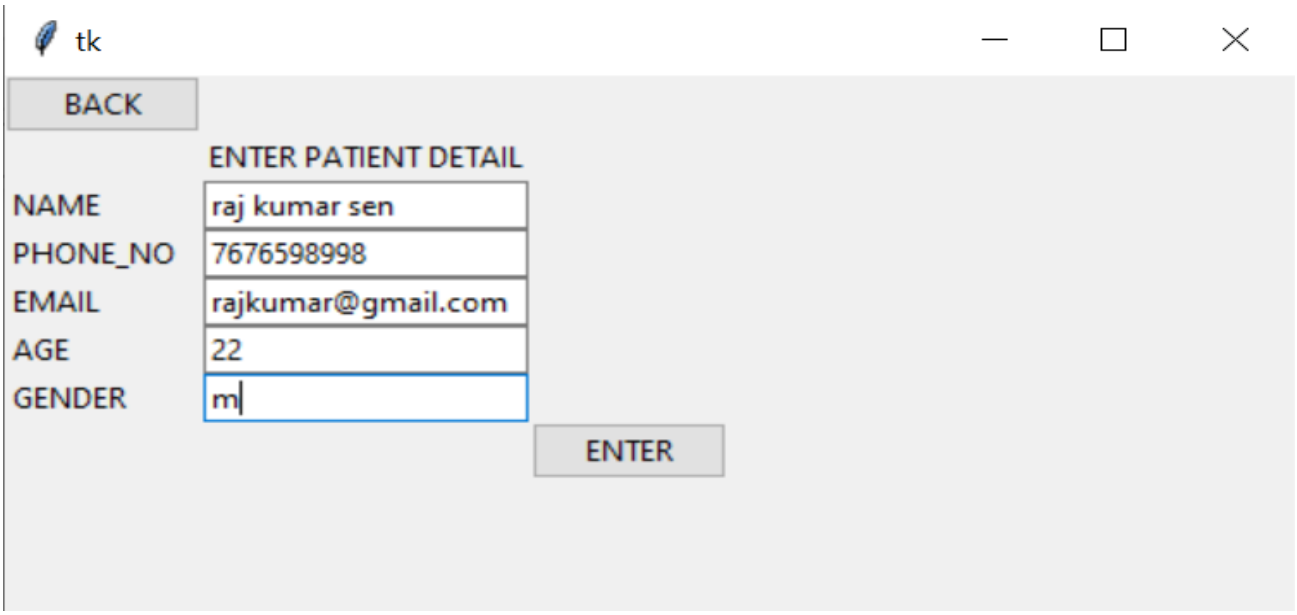
```
    else:
        frame.tkraise()
        cursor1.execute("INSERT INTO 'DETAILS'(NAME,ADDRESS,EMAIL,PHONE_NO) VALUES(?,?,?,?)",(x,a,z,y))
        cursor1.execute("INSERT INTO 'USER'(UNAME,PASSWORD) VALUES(?,?)",(x,b))

        db.commit()

frame3_button2=ttk.Button(frame3,text="REGISTER",command=lambda:dashboard4(frame1))
frame3_button2.grid(row=35,column=1)


frame3_button3=ttk.Button(frame3,text="CANCEL",command=lambda:dashboard(frame1))
frame3_button3.grid(row=35,column=5)

dashboard(frame1)
root.mainloop()
```

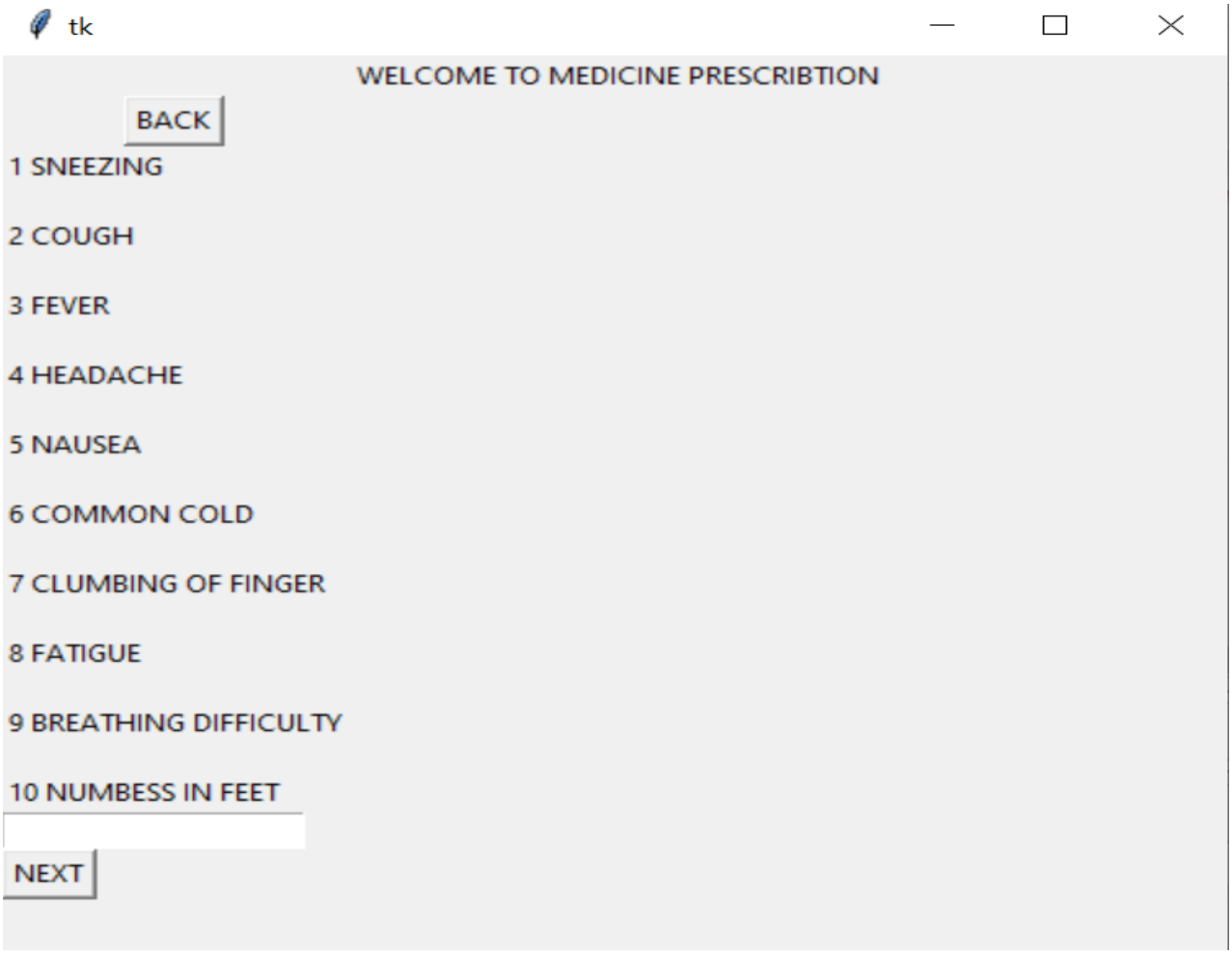
CHAPTER 7

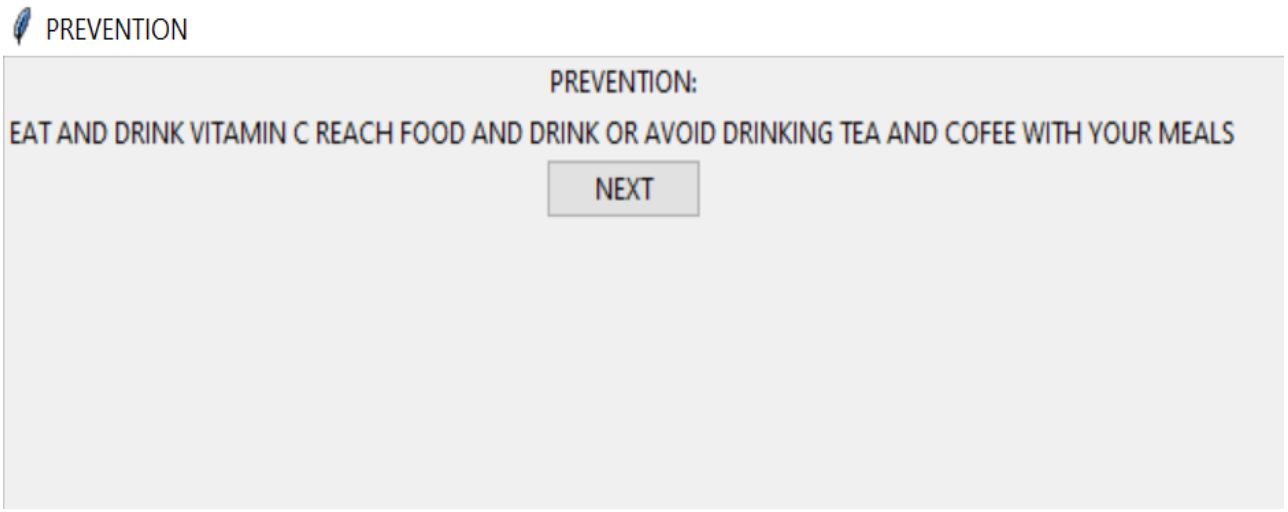
OUTPUTS






tk
BACK
ENTER PATIENT DETAIL
NAME
raj kumar sen
PHONE_NO
7676598998
EMAIL
rajkumar@gmail.com
AGE
22
GENDER
m
ENTER


tk
WELCOME TO MEDICINE PRESCRIBTION
BACK
1 SNEEZING
2 COUGH
3 FEVER
4 HEADACHE
5 NAUSEA
6 COMMON COLD
7 CLUMBING OF FINGER
8 FATIGUE
9 BREATHING DIFFICULTY
10 NUMBESS IN FEET
NEXT


PREVENTION
PREVENTION:
EAT AND DRINK VITAMIN C REACH FOOD AND DRINK OR AVOID DRINKING TEA AND COFEE WITH YOUR MEALS
NEXT


tk
CONSULTING DOCTOR
TIME
12
ENTER


DETAILS
PATIENT REPORT
2 {raj kumar} 7676598998 rajkumar@gmail.com 22 m

CONCLUSION

The Entry Box won't take any invalid address. The entry box always checks for availability of files in the address that is provided with. If the address contains any files that needs organization, this file organizer will organize them into their respective sophisticated folders, else it will simply throw an error message.

This software makes user friendly environment for the user to get the job done exactly what they hope this software to offer. The user can simply enter the address of the folder they want to get organized and just click the "Go ->" button. And there you have it, your organized folder with its respective file types. This software mainly deals with recognizing extinctions of files and place them into their respective group-folder so that the window looks well organized.

This software is generally designed to well organize any folder on a PC and have the user the easiness to find a specific file type in a pile of junk files. This project really does make file organizing easy and faster with literally just one click of a button.