1/3/2022

PROJECT REPORT

Hospital Management System

Name: MUHAMMAD ZAIN UL ABEDIN

Roll no.: SE-081 / Batch 2021

Class: SOFTWARE ENGINEERING (*B*)

Course: CT-175 (*PL*)

Semester: First Year (*FALL*)

ABSTRACT

The purpose of choosing this theme for Database application was to hone the previously learned skills to date. The application is written in python programming language and due to time constraints might be lacking in some areas. The Database software used is Microsoft Access due to its user-friendly interface. What's good about this application is that a dynamic approach is taken wherever possible, reflecting code flexibility. The whole source code is provided at the end of the report, if one wishes to understand the code He\She must have some basic knowledge about SQL, tkinter module, basic python, and list comprehensions. The application is currently no match for industry standards but the student hopes to refine it over time.

Modules

1. datetime
   Utilized for getting current datetime.
2. math
   Utilized for ceiling floating-point numbers.
3. random
   Utilized for generating random receipt numbers.
4. tkinter
   Utilized as Front-end for Database Application

   Classes used: -

   - ttk
   - filedialog

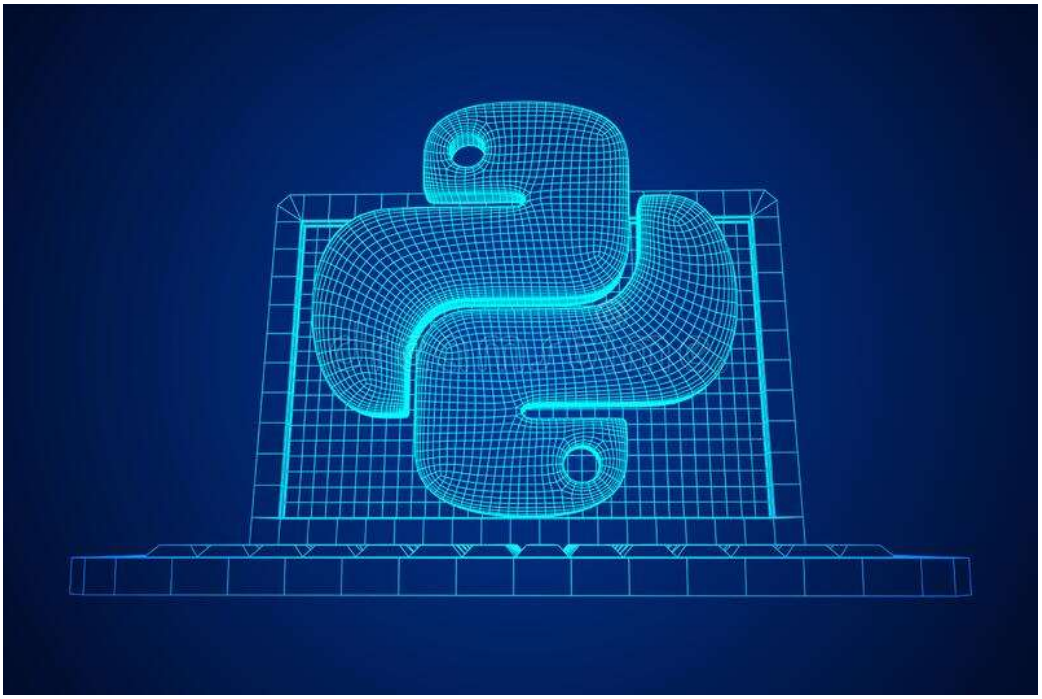
   Widgets used: -

   - Toplevel
   - OptionMenu
   - Treeview
   - ScrollBar
   - Labels
   - Frames
   - messagebox

Libraries

1. pyautogui
   utilized for saving generated receipt in form of images.
2. pyodbc
   Utilized for accessing ODBC databases.
3. pillow
   Image library used for displaying images in front-end.

BACKEND



FUNCTIONS

1. windowG()

Generates a non-resizable window of desired dimensions and name along with title and icon.

2. Insert_update()

Umbrella Function containing

1. In_up_sub() : Contains:-
2. add_record() : Enables user to input unit entered record in database
3. sel_record() : Allows the user to select a particular record in Treeview
   It defines a GLOBAL variable named previousEntry
4. up_record() : It updates a selected record
5. remove_record(): It deletes a selected record
6. clr_entries() : Clear all the entries, mostly used as RECURSION in above functions

3. receipt()

Umbrella function for Patient's receipt

1. Trace Callback Functions: used as a call for parent variable trace in receipt generation
   - a_1) name_opt
   - b_1) gender_opt
   - c_1) age_opt
   - d_1) cont_opt
   - e_1) status_opt
   - a_2) doct_name
   - b_2) doct_fee
   - a_3) test_cost
   - b_3) med_cost
2. total: Calculates total amount
3. stamp: insert management stamp
4. med_quantity_update: updates remaining quantity of purchased product to the database
5. reportshot: Saves Screenshot of ready receipt

4. receipt_hist()

Display the saved receipt as images

1. open_dialog: Opens a window dialog box for displaying the desired image

DATABASE

Connection String :-

pyodbc.connect(r''Driver={Microsoft Access Driver (*.mdb, *.accdb)};"

r''DBQ=Drive:\Path\fileName.extension;'')

connections = 2

cnxn = 4 cursors

cnxn1 = 1 cursor

Methods :-

.execute() : Executes Database Query or Command

.commit() : Commit any pending transactions to database

.fetchall() : It fetches all the rows where cursor object is pointing through query

.description() : It returns sequence of 7 items first being the names of columns of specified table

Note : In order for above 2 methods to work the Query should be executed so that the cursor object is in position

Queries:-

Note: Only **Dynamic Queries** are discussed in this section consult code section for static Queries

table_dict[table name corresponding to radio button value]

1. insert_Query

   It Inserts a new unit row in the Database in specified Table

   Explanation: replace method is used to Handle 'programming Error'

   isdigit method is used to Handle 'Data type Mismatch Criteria Error'

2. remove_Query

   It will Delete selected unit row in the Database

   Explanation: columns iterator gives the Primary key Field of Record of Database at 0^{th}

   The index which is essential for WHERE Clause

   values iterator gives the unit selected row of Treeview which is in

   correspondence with database

3. update_Query

   it will update Selected

   Explanation: column_names_string variable unpacks the columns of the selected table in such a way that Query is parameterized through qmarks (?)

   the API 2.0 Feature of pyodbc allows? to serve as a placeholder for values that can be passed as positional arguments in execute method

   columns iterator gives the Primary key Field of Record of Database at 0th index which is essential for WHERE Clause

   updated_entries variable uses isdigit method to Handle Data type Mismatch in Criteria Error

   previousEntry is a GLOBALLY assigned variable that stores the unmodified form of primary key Field Record of selected Treeview row corresponding to Database

4. fetched_table_Query

   It fetches the table upon submission of the selected radio button

5. query5

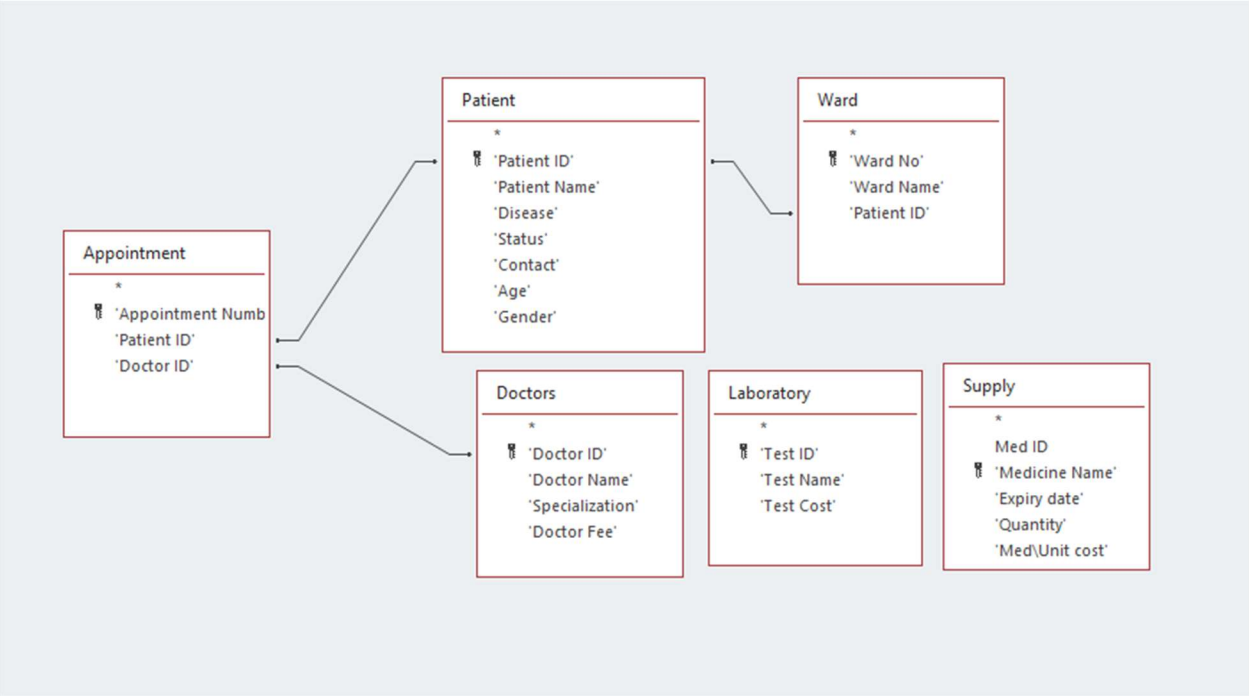
   It gets the quantity of selected medicine in the option menu

   Explanation: sel12 variable stores the selected medicine in the option menu

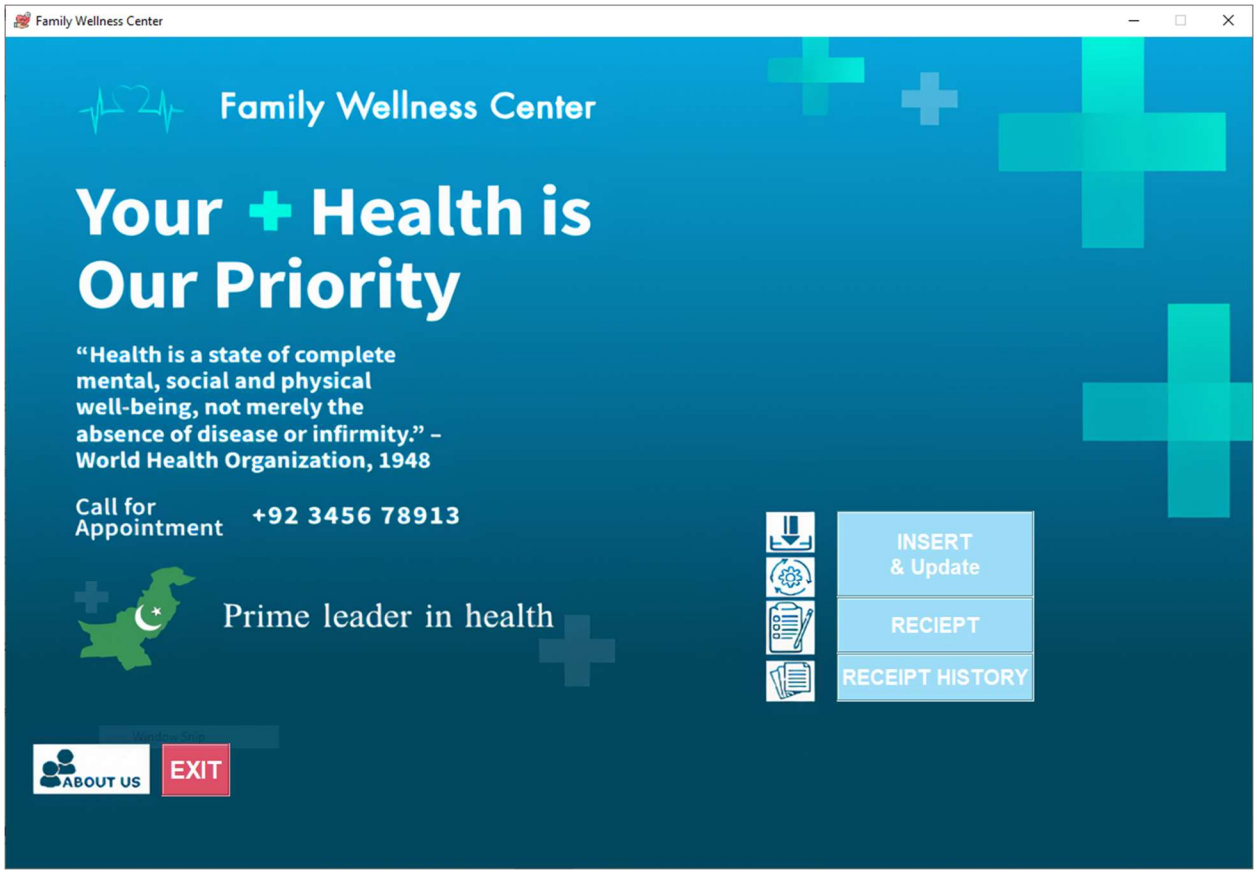
6. med_update_Query

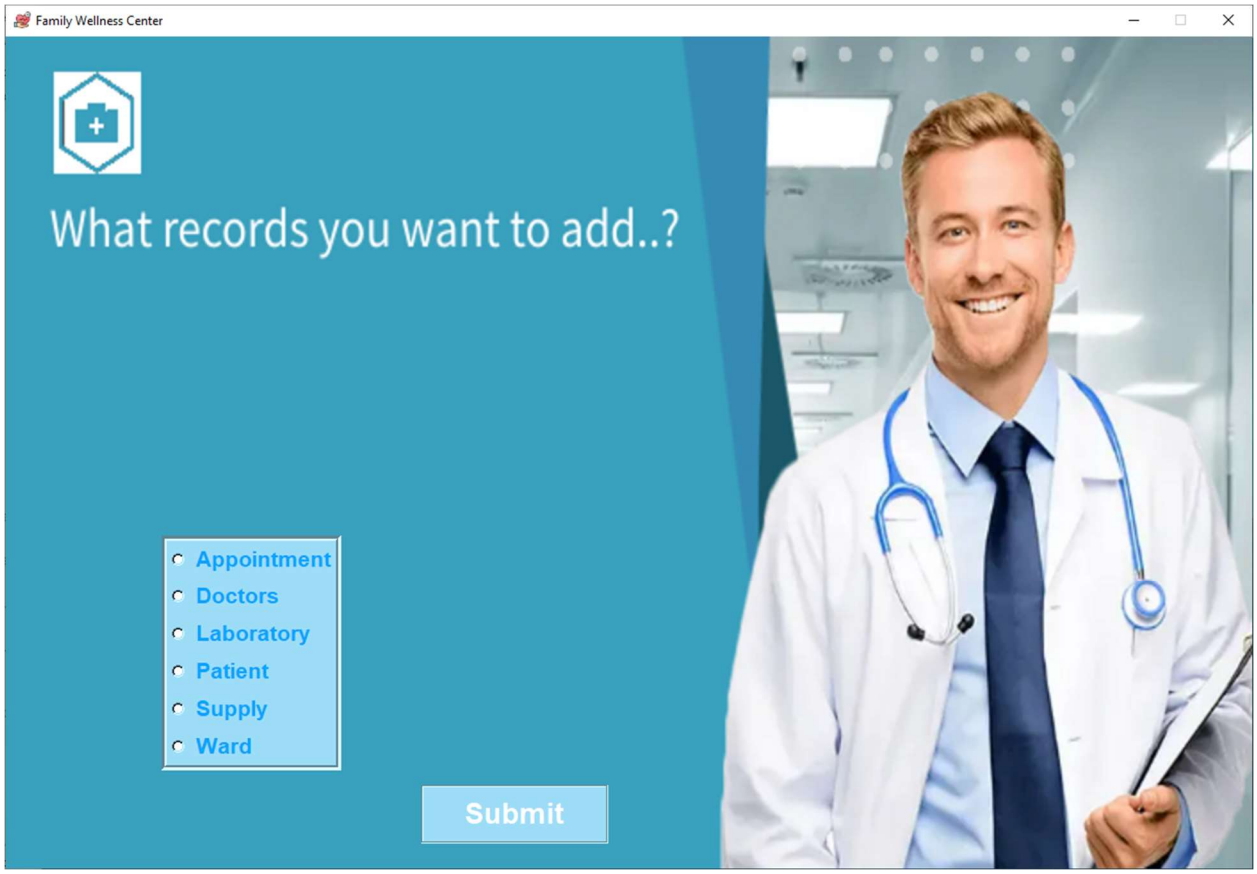
   it subtracts unit selected medicine fetched by query5

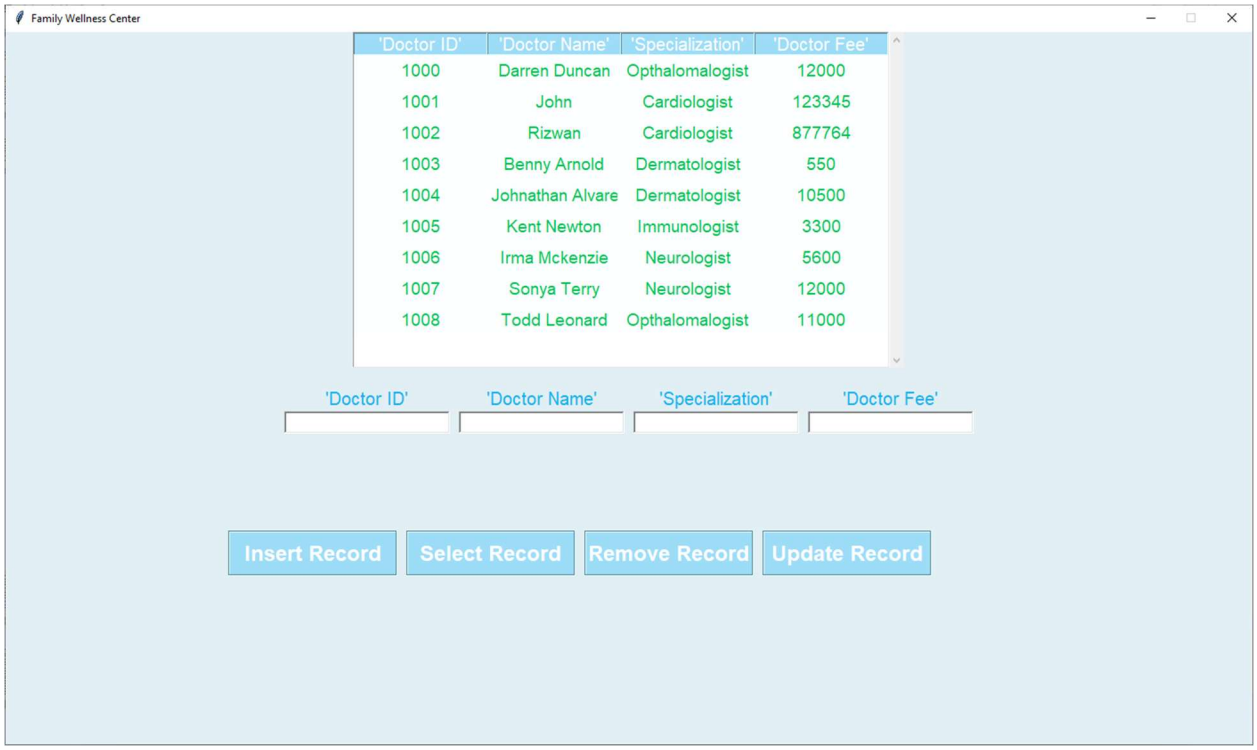
Query Relations:-

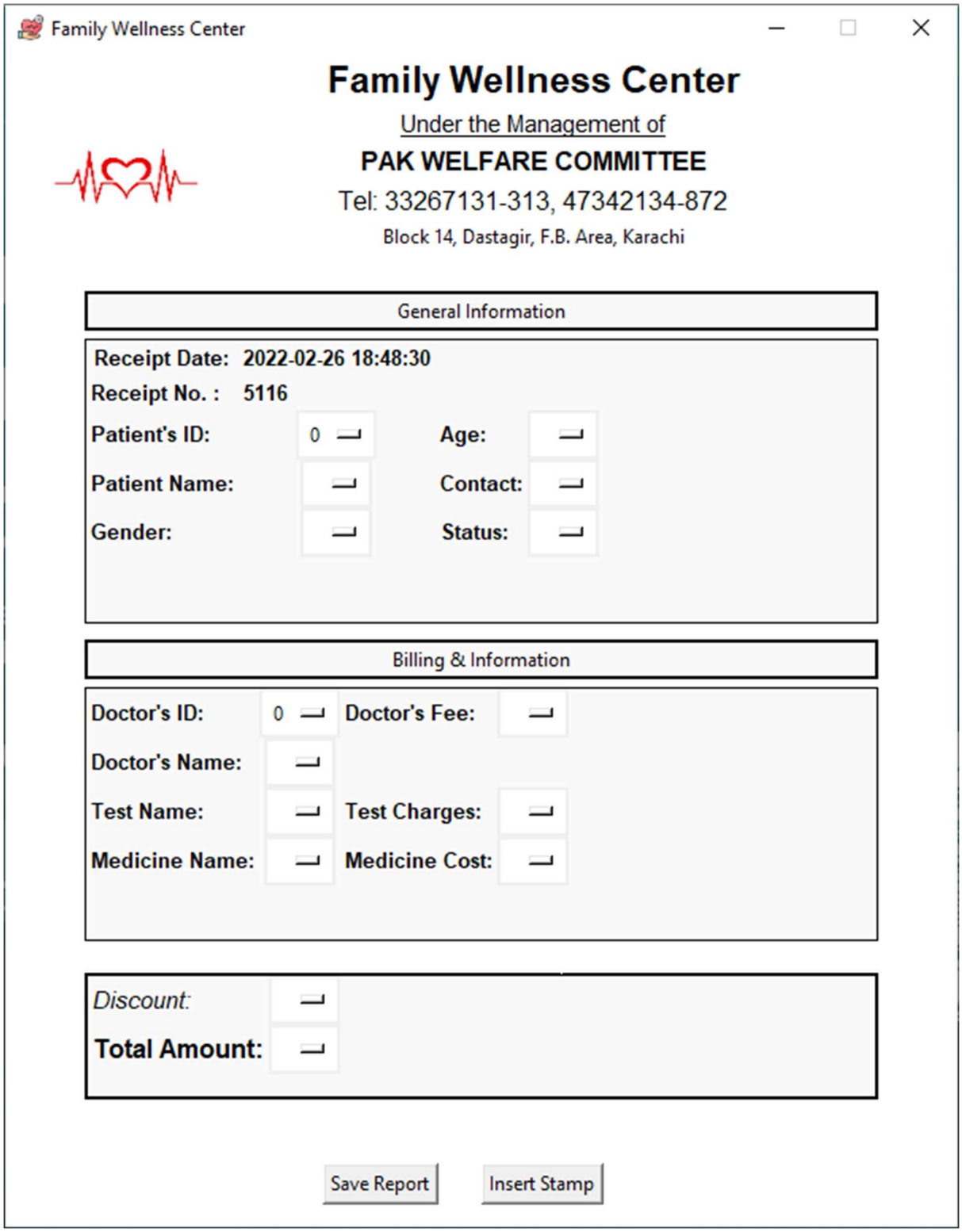


TK INTERFACE




Family Wellness Center
What records you want to add..?
Appointment
Doctors
Laboratory
Patient
Supply
Ward
Submit


Family Wellness Center
'Doctor ID' 'Doctor Name' 'Specialization' 'Doctor Fee'
1000 Darren Duncan Opthalomalogist 12000
1001 John Cardiologist 123345
1002 Rizwan Cardiologist 877764
1003 Benny Arnold Dermatologist 550
1004 Johnathan Alvare Dermatologist 10500
1005 Kent Newton Immunologist 3300
1006 Irma Mckenzie Neurologist 5600
1007 Sonya Terry Neurologist 12000
1008 Todd Leonard Opthalomalogist 11000
'Doctor ID'
'Doctor Name'
'Specialization'
'Doctor Fee'
Insert Record
Select Record
Remove Record
Update Record


Family Wellness Center
Family Wellness Center
Under the Management of
PAK WELFARE COMMITTEE
Tel: 33267131-313, 47342134-872
Block 14, Dastagir, F.B. Area, Karachi
General Information
Receipt Date: 2022-02-26 18:48:30
Receipt No. : 5116
Patient's ID: 0
Age:
Patient Name:
Contact:
Gender:
Status:
Billing & Information
Doctor's ID: 0
Doctor's Fee:
Doctor's Name:
Test Name:
Test Charges:
Medicine Name:
Medicine Cost:
Discount:
Total Amount:
Save Report
Insert Stamp

SOURCE CODE

```
import datetime
import math
import random
from tkinter import *
from tkinter import ttk
from tkinter import messagebox
from tkinter import filedialog
from PIL import Image, ImageTk
import pyautogui
import pyodbc

# Initializing 1st Master
root = Tk()

# window Creator Fx
def windowG(name, width, height):
    name.geometry(f"{width}x{height}")
    name.resizable(width=FALSE, height=FALSE)
    name.iconbitmap(r'C:\Users\NCS\Desktop\images\win1.ico')
    name.title('Family Wellness Center')
windowG(root, 1200, 800)


frame1 = Frame(root, relief=FLAT)
frame1.pack()
img1 = Image.open(r'C:\Users\NCS\Desktop\images\back1.png')
img1res = img1.resize((1200, 800))
photo1 = ImageTk.PhotoImage(img1res)
img1_label = Label(frame1, image=photo1)
img1_label.pack()
```

```
# INSERT & UPDATE FUNCTION
def Insert_update():
    # 2nd master
    # Connection to Database
    cnxn = pyodbc.connect(r"DRIVER={Microsoft Access Driver (*.mdb, *.accdb)};"
                          r"DBQ=C:\Users\NCS\Desktop\HM SYSTEM1.accdb;")
    cur1 = cnxn.cursor()
    win_ins = Toplevel()
    windowG(win_ins,1200 ,800)
    # Image Frame1
    frame3 = Frame(win_ins, relief=FLAT)
    frame3.pack()
    img2 = Image.open(r'C:\Users\NCS\Desktop\images\back2.png')
    img2res = img2.resize((1200, 800))
    photo2 = ImageTk.PhotoImage(img2res)
    img2_label = Label(frame3, image=photo2)
    img2_label.pack()
    # radio selection buttons and frame
    frame4 = Frame(win_ins, relief=GROOVE, background="#9EDDF8", bd=5)
    frame4.place(x=150, y=480)
    var1 = IntVar()
    # Relative Dictionary
    table_dict = {1: 'Appointment', 2: 'Doctors', 3: 'Laboratory', 4: 'Patient', 5: 'Supply', 6: 'Ward'}
    for (x, y) in table_dict.items():
        radio = Radiobutton(frame4, text=y, variable=var1, value=x, font=("lucida", "16", "bold"), justify=LEFT)
        radio.configure(background="#9EDDF8", activeforeground="Green", activebackground="#9EDDF8", foreground="#0da2ff")
        radio.pack(side=TOP, anchor="w")


    def In_up_Sub():
        ent_list = []
```

```
# INSERTS A NEW RECORD
def add_record():
    msg_reponse = messagebox.askyesno("ADD RECORD", 'Do you want to add the entered record', parent=win_in_sub)
    print(msg_reponse)
    if msg_reponse is True:

        appTree.insert('', index='end', values=[ent_list[j].get() for j in range(len(ent_list))])

        # a)Replace method Used To remove '' mod
        # b)Checks if the String is convertible to int
        insert_Query = f'''INSERT INTO {table_dict[Table]} {tuple([[i.replace("'", "")] for i in columns])}
                        VALUES {tuple([int(j.get()) if (j.get()).isdigit() else j.get() for j in ent_list])}'''
        cur1.execute(insert_Query)
        cnxn.commit()
        clr_entries()
    else:
        pass

# CLEAR ALL ENTRY FIELDS
def clr_entries():
    for text in ent_list:
        text.delete(0, END)
# SELECTS A UNIT FEILD
def sel_record():
    msg_reponse = messagebox.askyesno("Select RECORD", 'Do you want to Select the Existing record',
                                      parent=win_in_sub)
    if msg_reponse is True:
        sel_record = appTree.focus()
        values = appTree.item(sel_record, 'values')
        for i in range(len(columns)):
            ent_list[i].insert(0, values[i])
        # Globally assigned to to use in other functions
        global previousEntry
        previousEntry = ent_list[0].get()
```

```
        else:
            pass
    # REMOVES SELECTED RECORD
    def remove_record():
        msg_reponse = messagebox.askyesno("Remove RECORD", 'Do you want to Remove the Selected record',
                                          parent=win_in_sub)
        if msg_reponse is True:
            sel_record = appTree.focus()
            values = appTree.item(sel_record, 'values')
            print(columns[0])
            appTree.delete(sel_record)

            renove_Query = f'''DELETE FROM {table_dict[Table]} WHERE [{columns[0]}] = {values[0]}'''
            cur1.execute(renove_Query)
            cur1.commit()
        else:
            pass
    # UPDATES A SELECTED RECORD
    def up_record():
        msg_reponse = messagebox.askyesno("Update RECORD", 'Do you want to Update the Selected record', parent=win_in_sub)
        if msg_reponse is True:
            # RETURNS coresponding selected treeview Tuple
            sel_record = appTree.focus()
            appTree.item(sel_record, text='', values=[i.get() for i in ent_list])

            column_names_string = ', '.join([f'''{(f"[{i}]")}= ?''' for i in columns])
            updated_entries = [int(j.get()) if (j.get()).isdigit() else j.get() for j in ent_list]

            update_Query = f'''UPDATE {table_dict[Table]} SET {column_names_string} WHERE [{columns[0]}]={previousEntry}'''

            cur1.execute(update_Query, updated_entries)
            cnxn.commit()
            clr_entries()
        else:
            pass
```

```
# 3rd window
win_in_sub = Toplevel()
win_in_sub.geometry('1400x800')
win_in_sub.resizable(width=FALSE, height=FALSE)
win_in_sub.configure(background='#e1eff5')
# Get to radio button value
Table = var1.get()

# Tree Frame1
frame7 = Frame(win_in_sub, relief=FLAT, width=1100)

frame7.pack()

# Tree Scrollbar
scrollbar_T = Scrollbar(frame7)
scrollbar_T.pack(side=RIGHT, fill=Y)

# Styling our Treeview
tree_style = ttk.Style()
tree_style.theme_use('default')
tree_style.configure('Treeview.Heading', background="#9EDDF8", foreground='White', font=("lucida", "15"),
                     relief=GROOVE, border=10)

tree_style.configure("Treeview",
                     background="#FDFEFF",
                     foreground="#06C258",
                     rowheight=35, font=("TimesNewRoman", "14"))

tree_style.map("Treeview", background=[("selected", "green")])

# Tree View
appTree = ttk.Treeview(frame7, selectmode=BROWSE, yscrollcommand=scrollbar_T.set, show='headings')
appTree.pack()
scrollbar_T.config(command=appTree.yview)
```

```
# Fetching Column Names
fetched_Table_Query = f"SELECT * FROM {table_dict[Table]}"
fetched_table = cur1.execute(fetched_Table_Query)
columns = [i[0] for i in cur1.description]

# Initializing  Columns by list comp.
appTree['columns'] = [(j) for j in range(len(columns))]

appTree.column('#0', width=0, stretch=NO)
appTree.heading('#0', text='l')

# Creating Columns and headings
for j in range(len(columns)):
    appTree.column(str(j), width=150, anchor=CENTER, stretch=NO)
    appTree.heading(str(j), text=columns[j])


# Configuring columns for specific iid
global count
count = 0
for i in fetched_table:
    p = list(i)
    appTree.insert("", 'end', iid=str(count), values=[p[j] for j in range(len(columns))])
    count += 1

in_frame = Frame(win_in_sub, background='#e1eff5')
in_frame.pack(pady=20)

# Entries
for i in range(len(columns)):
    in_label = Label(in_frame, text=f'{columns[i]}', padx=5, background='#e1eff5', font=("Arial", "15"),
                     foreground='#17b1ea')
    in_label.grid(row=0, column=i)

    in_entry = Entry(in_frame, borderwidth=2, font=("lucidaConsolas", "13", 'bold'), foreground='Green')
```

```
        in_entry = Entry(in_frame, borderwidth=2, font=("lucidaConsolas", "13", 'bold'), foreground='Green')
        in_entry.grid(row=1, column=i, padx=5)
        ent_list.append(in_entry)

    # TREEVIEW BUTTONS
    add_button = Button(win_in_sub, text="Insert Record", command=add_record, font=("lucida", "18", "bold"),
                        relief=GROOVE, foreground="white", background="#9EDDF8", activebackground="white",
                        activeforeground="#0BA7E0", width=12)
    add_button.place(x=250, y=560)

    select = Button(win_in_sub, text="Select Record", command=sel_record, font=("lucida", "18", "bold"),
                    relief=GROOVE, foreground="white", background="#9EDDF8", activebackground="white",
                    activeforeground="#0BA7E0", width=12)
    select.place(x=450, y=560)

    remove_button = Button(win_in_sub, text="Remove Record", command=remove_record, font=("lucida", "18", "bold"),
                           relief=GROOVE, foreground="white", background="#9EDDF8", activebackground="white",
                           activeforeground="#0BA7E0", width=12)
    remove_button.place(x=650, y=560)

    up_button = Button(win_in_sub, text="Update Record", command=up_record, font=("lucida", "18", "bold"),
                       relief=GROOVE, foreground="white", background="#9EDDF8", activebackground="white",
                       activeforeground="#0BA7E0", width=12)
    up_button.place(x=850, y=560)

# Submit to get table
sub1_button = Button(frame3, text="Submit", command=In_up_Sub, font=("lucida", "20", "bold"),
                     relief=GROOVE, foreground="white", background="#9EDDF8", activebackground="white",
                     activeforeground="#0BA7E0", width=10)
sub1_button.place(x=400, y=720)

win_ins.mainloop()
```

```
# RECEIPT FUNCTION
def receipt():
    # connection to Database

    cnxn = pyodbc.connect(r"DRIVER={Microsoft Access Driver (*.mdb, *.accdb)};"
                          r"DBQ=C:\Users\NCS\Desktop\HM SYSTEM1.accdb;")
    cur1 = cnxn.cursor()
    cur2 = cnxn.cursor()
    cur3 = cnxn.cursor()
    cur4 = cnxn.cursor()

    receipt = Toplevel()
    windowG(receipt,600 ,740)
    receipt.configure(background='White')

    backf1 = Frame(receipt)
    backf1.place(x=0, y=0)
    # Logo
    img3 = Image.open(r'C:\Users\NCS\Desktop\images\heat1.png')
    img3_res = img3.resize((150, 140))
    global photo3
    photo3 = ImageTk.PhotoImage(img3_res)
    img3_label = Label(backf1, image=photo3, background='White')
    img3_label.grid(row=0, column=0)

    # Hospital INFO
    backf2 = Frame(receipt)
    backf2.place(x=200, y=0)

    label_h1 = Label(backf2, text='Family Wellness Center', font=('Futura', '17', 'bold'), background='White')
    label_h1.pack(anchor='center', fill=X)
    label_h2 = Label(backf2, text='Under the Management of', font=('Futura', '11', 'underline'), background='White')
    label_h2.pack(anchor='center', fill=X)
    label_h3 = Label(backf2, text='PAK WELFARE COMMITTEE', font=('Futura', '12', 'bold'), background='White')
```

```
label_h3 = Label(backf2, text='PAK WELFARE COMMITTEE', font=('Futura', '12', 'bold'), background='White')
label_h3.pack(anchor='center', fill=X)
label_h4 = Label(backf2, text='Tel: 33267131-313, 47342134-872', font='11', background='White')
label_h4.pack(anchor='center', fill=X)
label_h5 = Label(backf2, text='Block 14, Dastagir, F.B. Area, Karachi', background='White')
label_h5.pack(anchor='center', fill=X)

backf3 = Frame(receipt, highlightbackground='Black', highlightthickness=2, background='#F9F9F9')
backf3.place(x=50, y=150)
label_a1 = Label(backf3, text='General Information', background='#F9F9F9').grid(row=0, column=0, padx=192)

# Patients info
backf4 = Frame(receipt, highlightbackground='Black', highlightthickness=1, width=500, height=180,background='#F9F9F9')
backf4.place(x=50, y=180)
backf4.grid_propagate(False)

label_p1 = Label(backf4, text='Receipt Date:', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=0,column=0)
label_pa = Label(backf4, text=f'{datetime.datetime.now().replace(microsecond=0)}',font=('Futura', '10', 'bold'),background='#F9F9F9')
label_pa.grid(row=0, column=1)
random_val = random.randint(0000, 9500)
label_p2 = Label(backf4, text='Receipt No. :', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=1,column=0,sticky='w')
label_pb = Label(backf4, text=f'{str(random_val)}', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=1,column=1,sticky='w')

label_p3 = Label(backf4, text="Patient's ID:", font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=2,column=0, sticky='w')
label_p4 = Label(backf4, text='Patient Name:', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=3,column=0)
label_p5 = Label(backf4, text='Gender:', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=4, column=0,sticky='w')
label_p6 = Label(backf4, text='Age:', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=2, column=2,sticky='w')
label_p7 = Label(backf4, text='Contact:', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=3, column=2,sticky='w')
label_p8 = Label(backf4, text='Status:', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=4, column=2,sticky='w')

# Static Queries
query1 = "SELECT ['Patient ID'], ['Patient Name'], ['Gender'], ['Age'], ['Contact'], ['Status'] FROM Patient"
cur1.execute(query1)

patient_val = cur1.fetchall()
```

```
ID      = [i[0] for i in patient_val]
name    = [i[1] for i in patient_val]
gender  = [i[2] for i in patient_val]
age     = [i[3] for i in patient_val]
contact = [i[4] for i in patient_val]
status  = [i[5] for i in patient_val]

ID_name    = dict(zip(ID, name))
ID_gen     = dict(zip(ID, gender))
ID_age     = dict(zip(ID, age))
ID_contact = dict(zip(ID, contact))
ID_stat    = dict(zip(ID, status))

def name_opt(a, b, c):
    value = ID_name[sel1.get()]
    sel2.set(value)

def gender_opt(a, b, c):
    value = ID_gen[sel1.get()]
    sel3.set(value)

def age_opt(a, b, c):
    value = ID_age[sel1.get()]
    sel4.set(value)

def cont_opt(a, b, c):
    value = ID_contact[sel1.get()]
    sel5.set(value)

def status_opt(a, b, c):
    value = ID_stat[sel1.get()]
    sel6.set(value)
```

```
sel1 = IntVar()
sel2 = StringVar()
sel3 = StringVar()
sel4 = StringVar()
sel5 = StringVar()
sel6 = StringVar()

Pid = OptionMenu(backf4, sel1, *ID_gen.keys())

Pid.configure(relief=FLAT, background='White')
Pid.grid(row=2, column=1)

# Patient Gender

Pname = OptionMenu(backf4, sel2, sel1.trace('w', name_opt))
Pname.configure(relief=FLAT, background='White')
Pname.grid(row=3, column=1)

Pgen = OptionMenu(backf4, sel3, sel1.trace('w', gender_opt))
Pgen.configure(relief=FLAT, background='White')
Pgen.grid(row=4, column=1)

Page = OptionMenu(backf4, sel4, sel1.trace('w', age_opt))
Page.configure(relief=FLAT, background='White')
Page.grid(row=2, column=3)

Pcont = OptionMenu(backf4, sel5, sel1.trace('w', cont_opt))
Pcont.configure(relief=FLAT, background='White')
Pcont.grid(row=3, column=3)

Pstat = OptionMenu(backf4, sel6, sel1.trace('w', status_opt))
Pstat.configure(relief=FLAT, background='White')
Pstat.grid(row=4, column=3)
```

```
# Billing info
# General information
backf5 = Frame(receipt, highlightbackground='Black', highlightthickness=2, background='#F9F9F9')
backf5.place(x=50, y=370)

label_a2 = Label(backf5, text='Billing & Information', background='#F9F9F9').grid(row=0, column=0, padx=189)

backf6 = Frame(receipt, highlightbackground='Black', highlightthickness=1, width=500, height=160,
               background='#F9F9F9')
backf6.place(x=50, y=400)
backf6.grid_propagate(False)

label_b1 = Label(backf6, text="Doctor's ID:", font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=0,
                                                                                                         column=0,sticky='w')

label_b2 = Label(backf6, text="Doctor's Name:", font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=1,
                                                                                                          column=0,sticky='w')
label_b3 = Label(backf6, text="Test Name:", font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=2,
                                                                                                      column=0,sticky='w')
label_b4 = Label(backf6, text='Medicine Name:', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=3,
                                                                                                          column=0,sticky='w')
label_b5 = Label(backf6, text="Doctor's Fee:", font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=0,
                                                                                                         column=2,sticky='w')
label_b6 = Label(backf6, text='Test Charges:', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=2,
                                                                                                         column=2,sticky='w')
label_b7 = Label(backf6, text='Medicine Cost:', font=('Futura', '10', 'bold'), background='#F9F9F9').grid(row=3,
                                                                                                          column=2,sticky='w')
query2 = "SELECT ['Doctor ID'], ['Doctor Name'], ['Doctor Fee'] FROM Doctors"
cur2.execute(query2)
query3 = "SELECT ['Test Name'], ['Test Cost'] FROM Laboratory"
cur3.execute(query3)
query4 = r"SELECT ['Medicine Name'], ['Med\Unit cost'] FROM Supply"
cur4.execute(query4)
```

```
doctor_val = cur2.fetchall()
test_val = cur3.fetchall()
med_val = cur4.fetchall()

doct_id   = [i[0] for i in doctor_val]
doct_name = [i[1] for i in doctor_val]
doct_fee  = [i[2] for i in doctor_val]

test_name = [i[0] for i in test_val]
test_cost = [i[1] for i in test_val]

med_name  = [i[0] for i in med_val]
med_cost  = [i[1] for i in med_val]

doct_id_name = dict(zip(doct_id, doct_name))
doct_id_cost = dict(zip(doct_id, doct_fee))

test_name_cost = dict(zip(test_name, test_cost))

med_name_cost = dict(zip(med_name, med_cost))

def doct_name(a, b, c):
    value = doct_id_name[sel7.get()]
    sel8.set(value)

def doct_fee(a, b, c):
    value = doct_id_cost[sel7.get()]
    sel9.set(value)

def test_cost(a, b, c):
    value = test_name_cost[sel10.get()]
    sel11.set(value)
```

```
def med_cost(a, b, c):
    value = med_name_cost[sel12.get()]
    sel13.set(value)

sel7  = IntVar()
sel8  = StringVar()
sel9  = StringVar()
sel10 = StringVar()
sel11 = StringVar()
sel12 = StringVar()
sel13 = StringVar()

doct_id_opt = OptionMenu(backfó, sel7, *doct_id_name.keys())
doct_id_opt.configure(relief=FLAT, background='White')
doct_id_opt.grid(row=0, column=1)

doct_name_opt = OptionMenu(backfó, sel8, sel7.trace('w', doct_name))
doct_name_opt.configure(relief=FLAT, background='White')
doct_name_opt.grid(row=1, column=1)

doct_fee_opt = OptionMenu(backfó, sel9, sel7.trace('w', doct_fee))
doct_fee_opt.configure(relief=FLAT, background='White')
doct_fee_opt.grid(row=0, column=3)


test_name_opt = OptionMenu(backfó, sel10, *test_name_cost.keys())
test_name_opt.configure(relief=FLAT, background='White')
test_name_opt.grid(row=2, column=1)

test_cost_opt = OptionMenu(backfó, sel11, sel10.trace('w', test_cost))
test_cost_opt.configure(relief=FLAT, background='White')
test_cost_opt.grid(row=2, column=3)
```

```
med_name_opt = OptionMenu(backf6, sel12, *med_name_cost.keys())
med_name_opt.configure(relief=FLAT, background='White')
med_name_opt.grid(row=3, column=1)

med_cost_opt = OptionMenu(backf6, sel13, sel12.trace('w', med_cost))
med_cost_opt.configure(relief=FLAT, background='White')
med_cost_opt.grid(row=3, column=3)

backf7 = Frame(receipt, highlightbackground='Black', highlightthickness=2, background='#F9F9F9', width=500,
               height=80)
backf7.place(x=50, y=580)
backf7.grid_propagate(False)

sel14 = StringVar()
sel15 = StringVar()

# total
def total(a, b, c):
    try:
        if sel14.get() == 'Free':
            sel15.set('0.000')
        elif sel14.get() == '5%':
            total = float(sel9.get()) + float(sel11.get()) + float(sel13.get())
            total_disc = total - total * 0.05
            sel15.set(math.ceil(total_disc))
        elif sel14.get() == '15%':
            total = float(sel9.get()) + float(sel11.get()) + float(sel13.get())
            total_disc = total - total * 0.15
            sel15.set(math.ceil(total_disc))
        elif sel14.get() == '20%':
            total = float(sel9.get()) + float(sel11.get()) + float(sel13.get())
            total_disc = total - total * 0.20
            sel15.set(math.ceil(total_disc))
    except ValueError as v:
        print(v)
```

```
label_Discount = Label(backf7, text='Discount:', font=('Futura', '11', 'italic'), background='#F9F9F9').grid(row=0,
                                                                                                              column=0,sticky='w')
label_total = Label(backf7, text='Total Amount:', font=('Futura', '12', 'bold'), background='#F9F9F9').grid(row=1,
                                                                                                            column=0,sticky='w')

disc_opt = OptionMenu(backf7, sel14, '5%', '15%', '20%', 'Free')
disc_opt.configure(relief=FLAT, background='White')
disc_opt.grid(row=0, column=1)

total_opt = OptionMenu(backf7, sel15, sel14.trace('w', total))
total_opt.configure(relief=FLAT, background='White')
total_opt.grid(row=1, column=1)

def stamp():
    stamp_button.destroy()
    img4 = Image.open(r'C:\Users\NCS\Desktop\images\stamp1.png')
    img4_res = img4.resize((90, 90))
    # Global to avoid python garbage collection
    global photo4
    photo4 = ImageTk.PhotoImage(img4_res)
    img4_label = Label(backf8, image=photo4, background='White')
    img4_label.pack()

backf8 = Frame(receipt)
backf8.place(x=350, y=580)

stamp_button = Button(receipt, text='Insert Stamp', command=stamp)
stamp_button.place(x=300, y=700)

def med_quantity_update():
    cnxn1 = pyodbc.connect(r"DRIVER={Microsoft Access Driver (*.mdb, *.accdb)};"
                           r"DBQ=C:\Users\NCS\Desktop\HM SYSTEM1.accdb;")
```

```
    cur5 = cnxn1.cursor()
    query5 = f"Select ['Quantity'] FROM Supply WHERE ['Medicine Name'] = '{sel12.get()}'"
    cur5.execute(query5)
    fetched_Quantity = list(cur5.fetchone())[0] # Extracting value

    med_update_Query = f"Update Supply SET ['Quantity'] = '{fetched_Quantity - 1}' WHERE ['Medicine Name'] = '{sel12.get()}'"
    cur4.execute(med_update_Query)
    cnxn1.commit()
    cnxn1.close()
    cnxn.commit()
    cnxn.close()

# Saving the screenshot.
def reportshot():

    med_quantity_update()
    x = receipt.winfo_rootx()
    y = receipt.winfo_rooty()
    w = receipt.winfo_width()
    h = receipt.winfo_height()
    data = [("Image Files", "*.png *.jpg")]
    report_sc = pyautogui.screenshot(region=(x, y, w, h - 60))

    filename = filedialog.asksaveasfilename(filetypes=data, initialdir=r'C:\Users\NCS\Desktop\Reciepts',
                                            initialfile=f'{random_val}', defaultextension='*.*')
    if filename:
        report_sc.save(filename)

sc_button = Button(receipt, text='Save Report', command=reportshot)
sc_button.place(x=200, y=700)
```

```
# 1st window buttons

frame2 = Frame(root, relief=FLAT)
frame2.place(x=800, y=456)
insert = Button(frame2, text="INSERT\n& Update", command=Insert_update, font=("lucida", "15", 'bold'), width=15, pady=10, foreground="white"
              , background="#9EDDF8", activebackground="white", activeforeground="#0BA7E0", relief=GROOVE)
insert.grid(row=0, column=0)

receipt_gen = Button(frame2, text="RECIEPT", command=receipt, font=("lucida", "15", 'bold'), width=15, pady=7,foreground="white",
                    background="#9EDDF8", activebackground="white", activeforeground="#0BA7E0",relief=GROOVE)
receipt_gen.grid(row=2, column=0)

def receipt_hist():
    receipt_hist = Toplevel()
    window6(receipt_hist,600 ,710)
    recieptd_frame = Frame(receipt_hist,width=600,height=680)
    recieptd_frame.pack()
    reciept_label = Label(recieptd_frame)
    reciept_label.pack()


    def open_dialog():
        name = filedialog.askopenfilename(initialdir= r'C:\Users\NCS\Desktop\Reciepts', title='Select Receipt',
                                          filetypes=(("PNG files", "*.png"), ("All files", "*.*")),parent=receipt_hist)
        global imgr
        display_receipt = Image.open(name)
        imgr = ImageTk.PhotoImage(display_receipt)
        reciept_label.configure(image=imgr)
        reciept_label.image = imgr


    browse = Button(receipt_hist, text = 'BROWSE',command=open_dialog,background='White', highlightthickness=2,highlightcolor='Black', fg = 'blue')
    browse.pack(anchor=W,side=LEFT)

    leave = Button(receipt_hist, text= '  EXIT  ',command=receipt_hist.destroy, background='White', highlightthickness=2,highlightcolor='Black',fg = 'red')









    browse = Button(receipt_hist, text = 'BROWSE',command=open_dialog,background='White', highlightthickness=2,highlightcolor='Black', fg = 'blue')
    browse.pack(anchor=W,side=LEFT)

    leave = Button(receipt_hist, text= '  EXIT  ',command=receipt_hist.destroy, background='White', highlightthickness=2,highlightcolor='Black',fg = 'red')
    leave.pack(anchor=E,side=RIGHT)

receipt_hist = Button(frame2, text="RECEIPT HISTORY", command=receipt_hist,font=("lucida", "15", 'bold'), width=15, pady=3, foreground="white",
                    background="#9EDDF8", activebackground="white", activeforeground="#0BA7E0",relief=GROOVE)
receipt_hist.grid(row=3, column=0)

close = Button(root, text="EXIT",command=exit ,font=("lucida", "17", 'bold'), width=4, pady=3, foreground="white",
                  background="#dd4e67", activebackground="white", activeforeground="#d52342", relief=GROOVE)
close.place(x=150, y=680)


root.mainloop()
```