INDEX

PROJECT PLAN

Name: **Jeffrey Jaijo**

Class: **12 B**

REG NO:

Project Name: **Pharmacy Manager**

Members: **Lince Louis**

Duration: **2 Months**

Date of submission:

ACKNOWLEDGEMENT

I sincerely thank our management for giving this golden opportunity and for providing the facilities for the competition of this computer science project.

I sincerely thank our principal Mrs Reena Rajesh for supporting us throughout the project.

I express my special thanks to my teacher Mrs. Reshmi B for guiding and supporting us throughout the project which helped a lot for this project.

I would like to express my deep gratitude to all my teachers, classmates, parents for their timely help and motivation which helped me to continue in a pace throughout the project.

INTRODUCTION

We have developed a project on stock management for a pharmacy.

Stock management is a difficult process that involves many people. We believe that we can greatly reduce the labor required to keep up with a large store, by providing an efficient and robust stock management system.

We believe that this program can help pharmacies or even other stores for managing their stocks.

FILES AND FUNCTIONS

Types of files used – Database files

Modules used

Pillow

Tkinter

Customtkinter

SQLite3

OS

CtkTable

Files used

Ctktable.py

userdat.db

loginbackground.jpg

Logo.ico

Functions Used

def newuse()

def submitted()

def loginsubev()

def loginsub()

def gridmake()

def frames()

def adbt()

def rembt()

def adtotree()

def remfromtree()

def modbt()

def modifytree()

def search()

def add()

def rembut()

SOURCE CODE

```
import customtkinter as ctk #UI elements
import tkinter as tk # for extra ui elements
from tkinter import messagebox #for displaying messsages
from PIL import Image #for background images
from ctktable import CTkTable #Module for ctk tables
import sqlite3 as mydb #for storing data in database
import os # for deletion of files

ctk.set_appearance_mode('Dark')

#database for user data
userdatdir = os.path.dirname(__file__)
userdatpath = os.path.join(userdatdir, 'userdat.db')
con = mydb.connect(userdatpath)
cur = con.cursor()

#image background
global backgroundpath
backgrounddir = os.path.dirname(__file__)
backgroundpath = os.path.join(backgrounddir, 'Image
files\loginbackground.jpg')

#icon for program
global iconpath
icondir = os.path.dirname(__file__)
iconpath = os.path.join(icondir, 'Image files\Logo.ico')

#variable for checking if signed in or not
global signed
signed = False



#login window
class App(ctk.CTk):
    def __init__(self):
        super().__init__()

        screen_width = self.winfo_screenwidth()
        screen_height = self.winfo_screenheight()
        x_cordinate = int((screen_width/2) - (800/2))
        y_cordinate = int((screen_height/2) - (600/2))
        self.geometry(f'800x600+{x_cordinate}+{y_cordinate}')
        self.resizable(False, False)
```

```
        self.iconbitmap(iconpath)

        self.title('Pharmacy Manager')

        # background
        back = ctk.CTkImage(light_image=Image.open(backgroundpath),
            dark_image=Image.open(backgroundpath),
            size=(1600, 800))

        label = ctk.CTkLabel(self, image=back)
        label.pack()

        self.frame = loginfr(self)

        self.mainloop()

#frame inside login window
class loginfr(ctk.CTkFrame):
    def __init__(self, parent):
        super().__init__(parent, corner_radius=15, bg_color='#75b1a9',
width=350, height=400)

        # submit command
        def newuse(event):
            parent.destroy()

            class newuswin(ctk.CTk):
                def __init__(self):
                    super().__init__()

                    self.iconbitmap(iconpath)
                    self.title('Pharmacy Manager')
                    self.geometry('800x700')
                    self.resizable(False, False)

                    # background
                    back =
ctk.CTkImage(light_image=Image.open(backgroundpath),
                        dark_image=Image.open(backgroundpath),
                        size=(1600, 800))

                    label = ctk.CTkLabel(self, image=back)
                    label.pack()

                    self.newframe = newframe(self)

                    self.mainloop()
```

```
            class newframe(ctk.CTkFrame):
                def __init__(self, parentnew):
                    super().__init__(parentnew, corner_radius=15,
bg_color='#75b1a9', width=350, height=470)

                    def submitted():
                        if newpassvar.get() == confpassvar.get():
                            if newuservar.get() != '':
                                cur.execute('select * from user')
                                users = cur.fetchall()
                                for i in users:

                                    if newuservar.get() in i:
                                        messagebox.showerror('Error', 'User
already exists', icon='error')
                                        break

                                else:
                                    if newpassvar.get() == '' or
confpassvar.get() == '':
                                        messagebox.showerror('error',
'Password must be entered', icon='error')

                                    else:
                                        q = 'insert into user values("{}",
"{}")'.format(newuservar.get(), newpassvar.get())
                                        cur.execute(q)
                                        con.commit()

                                        parentnew.destroy()
                                        App()

                            else:
                                messagebox.showerror('Error', 'Username not
entered', icon='error')

                        else:
                            messagebox.showerror('Error', 'Password does not
match', icon='error')

                    # login label
                    newloginlab = ctk.CTkLabel(self, text='New user',
font=('Futura', 30))
                    newloginlab.place(x=175, y=55, anchor=tk.CENTER)

                    # username label, entry
                    newuserlab = ctk.CTkLabel(self, text='Username',
font=('Futura', 18))
```

```
                    newuserlab.place(x=30, y=110)

                    newuservar = ctk.StringVar()
                    newuserent = ctk.CTkEntry(self, width=290,
textvariable=newuservar)
                    newuserent.place(x=30, y=140)

                    # password label, entry
                    newpassvar = ctk.StringVar()
                    newpasslab = ctk.CTkLabel(self, text='Password',
font=('Futura', 18))
                    newpasslab.place(x=30, y=190)

                    newpassent1 = ctk.CTkEntry(self, width=290,
textvariable=newpassvar, show='*')
                    newpassent1.place(x=30, y=220)

                    # confirm pass
                    confpassvar = ctk.StringVar()
                    confpasslab = ctk.CTkLabel(self, text='Confirm Password',
font=('Futura', 18))
                    confpasslab.place(x=30, y=270)

                    newpassent = ctk.CTkEntry(self, width=290,
textvariable=confpassvar, show='*')
                    newpassent.place(x=30, y=300)

                    # submit
                    subbutton = ctk.CTkButton(self, text='Submit',
font=('Futura', 24), fg_color='#75b1a9',
                                              text_color='Black',
                                              hover_color='light green',
command=submitted)
                    subbutton.place(x=175, y=380, anchor=tk.CENTER)

                    self.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

            newuswin()

        # login label
        loginlab = ctk.CTkLabel(self, text='Login', font=('Futura', 30))
        loginlab.place(x=175, y=55, anchor=tk.CENTER)

        # username label, entry
        userlab = ctk.CTkLabel(self, text='Username', font=('Futura', 18))
        userlab.place(x=30, y=110)

        uservar = ctk.StringVar()
```

```
        userent = ctk.CTkEntry(self, width=290, textvariable=uservar)
        userent.place(x=30, y=140)

        # password label, entry
        passlab = ctk.CTkLabel(self, text='Password', font=('Futura', 18))
        passlab.place(x=30, y=190)

        passvar = ctk.StringVar()
        passent = ctk.CTkEntry(self, width=290, textvariable=passvar,
show='*')
        passent.place(x=30, y=220)



        # newuser
        newlab = ctk.CTkLabel(self, text='New User?', font=('Futura', 15))
        newlab.place(x=250, y=250)
        newlab.bind('<Button-1>', newuse)

        def loginsubev(event):
            cur.execute('select * from user')
            rows = cur.fetchall()
            global user
            user = uservar.get()
            for row in rows:
                if row[0] == uservar.get() and row[1] == passvar.get():
                    parent.destroy()
                    global signed
                    signed = True
                    break



            else:
                messagebox.showerror('Error', 'Invalid username or password',
icon='error')

        def loginsub():
            cur.execute('select * from user')
            rows = cur.fetchall()
            global user
            user = uservar.get()
            for row in rows:
                if row[0] == uservar.get() and row[1] == passvar.get():
                    parent.destroy()
                    global signed
                    signed = True
                    break
```

```
            else:
                messagebox.showerror('Error', 'Invalid username or password',
icon='error')

        passent.bind('<Return>', loginsubev)

        # submit
        subbutton = ctk.CTkButton(self, text='Submit', font=('Roboto Medium',
24), fg_color='#75b1a9',
                                  hover_color='light green',
text_color='Black',
                                  command=loginsub)
        subbutton.place(x=175, y=330, anchor=tk.CENTER, )

        self.place(relx=0.5, rely=0.5, anchor=tk.CENTER)


App()

#window for managing all the shelves
class manage(ctk.CTk):
    def __init__(self):
        super().__init__()

        self.geometry('800x600')
        self.resizable(False, False)
        self.iconbitmap(iconpath)
        self.title('Pharmacy Manager')

        # background
        back = ctk.CTkImage(light_image=Image.open(
            backgroundpath),
            dark_image=Image.open(
                backgroundpath),
            size=(1600, 800))

        label = ctk.CTkLabel(self, image=back)
        label.pack()

        self.contfr = contfr(self)

        self.mainloop()

#frame inside managing window
class contfr(ctk.CTkFrame):
    def __init__(self, parent):
```

```
        super().__init__(parent, corner_radius=15, bg_color='#75b1a9',
width=740, height=540)

        #to make the button grid for each shelf
        def gridmake(id, user):
            framedir = os.path.dirname(__file__)
            framepath = os.path.join(framedir, f'button_frame{id}{user}.db')
            buttoncon = mydb.connect(framepath)
            buttoncur = buttoncon.cursor()
            cur.execute('select * from userprofiles')
            rows = cur.fetchall()
            for row in rows:
                if row[1] == id:
                    columns = row[3]
                    rowtab = row[2]
                    parent.destroy()
                    buttonGrid = ctk.CTk()
                    buttonGrid.iconbitmap(iconpath)
                    buttonGrid.resizable(False, False)
                    buttonGrid.title(f'Shelf:  {id}')
                    screen_width = buttonGrid.winfo_screenwidth()
                    screen_height = buttonGrid.winfo_screenheight()
                    width = screen_width*46.8/100
                    height = screen_height*92.59/100
                    buttonGrid.geometry(f'{int(width)}x{int(height)}')
                    # background
                    back = ctk.CTkImage(light_image=Image.open(
                        backgroundpath),
                        dark_image=Image.open(
                            backgroundpath),
                        size=(2200, 1100))
                    label = ctk.CTkLabel(buttonGrid, image=back)
                    label.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

                    def backbuttoncom():
                        buttonGrid.destroy()
                        buttoncon.close()
                        manage()


                    backbuttonfr = ctk.CTkFrame(buttonGrid, corner_radius=15,
bg_color='#75b1a9', width=100, height=60)
                    backbuttonfr.rowconfigure(1, weight=1)
                    backbuttonfr.columnconfigure(1, weight=1)

                    backbuttonfr.place(x= 30, y= 20)
                    backbutton = ctk.CTkButton(backbuttonfr, text='<-Back',
font=('Roboto Medium', 24), fg_color='#75b1a9',
```

```
                                hover_color='light green',
text_color='Black', command=backbuttoncom)
                    backbutton.grid(row=1, column=1, padx = 10, pady = 10)

                    gridfr = ctk.CTkFrame(buttonGrid, corner_radius=15,
bg_color='#75b1a9', width=int(width)-40, height=int(height)-140)
                    for i in range(columns):
                        gridfr.grid_columnconfigure(i, weight=1)
                    for i in range(rowtab + 1):
                        gridfr.grid_rowconfigure(i, weight=1)
                    gridfr.place(relx=0.5, rely=0.53, anchor=tk.CENTER)
                    gridfr.grid_propagate(False)


                    #windows which display what is inside each shelf for
adding, modifying, removing items
                    def frames(r, c):
                        shelf = ctk.CTkToplevel(buttonGrid)
                        shelf.iconbitmap(iconpath)
                        shelf.config(bg='#75b1a9')
                        shelf.title(f'Shelf {r + 1}-{c + 1}')
                        shelf.wm_transient(buttonGrid)
                        shelf.geometry('600x600')
                        shelf.resizable(False, False)

                        # background
                        back = ctk.CTkImage(light_image=Image.open(
                            backgroundpath),
                            dark_image=Image.open(
                                backgroundpath),
                            size=(1600, 800))

                        label = ctk.CTkLabel(shelf, image=back)
                        label.pack()

                        shfr = ctk.CTkFrame(shelf, corner_radius=15,
bg_color='#75b1a9', width=540, height=540)
                        shfr.place(relx=0.5, rely=0.5, anchor=tk.CENTER)
                        shfr.grid_propagate(False)
                        shfr.columnconfigure((1, 2, 3), weight=1)
                        shfr.rowconfigure(1, weight=1)
                        shfr.rowconfigure(2, weight=2)
                        shfr.rowconfigure(3, weight=2)

                        shlab = ctk.CTkLabel(shfr, text=f'Shelf {r + 1}-{c +
1}', font=('Roboto Medium', 24))
                        shlab.grid(row=1, column=1, columnspan=3)
```

```
                        buttoncur.execute(f'select * from button{r + 1}_{c +
1}')
                        vals = buttoncur.fetchall()
                        vl = [('Item', 'Qty', 'Price', 'ID')] + vals

                        self.tree = CTkTable(shfr, column=4, row=len(vals) +
1, values=vl)

                        self.tree.grid(row=2, column=1, columnspan=3,
sticky='n', padx=20)


                        #function to add items
                        def adbt():
                            adtp = ctk.CTkToplevel(shelf)
                            adtp.title('Add Item')
                            adtp.wm_transient(shelf)
                            adtp.resizable(False, False)
                            adtp.geometry('480x360')

                            # background
                            back = ctk.CTkImage(light_image=Image.open(
                                backgroundpath),
                                dark_image=Image.open(
                                    backgroundpath),
                                size=(1600, 800))

                            label = ctk.CTkLabel(adtp, image=back)
                            label.pack()

                            adfr = ctk.CTkFrame(adtp, corner_radius=15,
bg_color='#75b1a9', width=420, height=300)
                            adfr.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

                            itmlab = ctk.CTkLabel(adfr, text='Item:',
font=('Roboto Medium', 24))
                            itmlab.place(x=40, y=40)

                            # itmennt
                            itmentvar = ctk.StringVar()
                            itment = ctk.CTkEntry(adfr,
textvariable=itmentvar, width=240)
                            itment.place(x=140, y=40)

                            # qty lab and entry
                            qtylab = ctk.CTkLabel(adfr, text='Qty:',
font=('Roboto Medium', 24))
                            qtylab.place(x=40, y=80)
```

```
                            qtyentvar = ctk.StringVar()
                            qtyent = ctk.CTkEntry(adfr,
textvariable=qtyentvar, width=240)
                            qtyent.place(x=140, y=80)

                            # price lab and ent
                            prlab = ctk.CTkLabel(adfr, text='Price:',
font=('Roboto Medium', 24))
                            prlab.place(x=40, y=120)

                            prentvar = ctk.StringVar()
                            prent = ctk.CTkEntry(adfr, textvariable=prentvar,
width=240)
                            prent.place(x=140, y=120)

                            # something id ent and var
                            partlab = ctk.CTkLabel(adfr, text='ID:',
font=('Roboto Medium', 24))
                            partlab.place(x=40, y=160)

                            partentvar = ctk.StringVar()
                            partent = ctk.CTkEntry(adfr,
textvariable=partentvar, width=240)
                            partent.place(x=140, y=160)


                            #for adding items to database and table
                            def adtotree():
                                if itmentvar.get() and qtyentvar.get() and
prentvar.get() and partentvar.get() != '':

                                    buttoncur.execute(f'select * from button{r
+ 1}_{c + 1}')
                                    records = buttoncur.fetchall()
                                    for i in records:
                                        if i[3] == partentvar.get():
                                            messagebox.showerror('error', 'ID
cannot be repeated', icon='error')
                                            break
                                    else:
                                        self.tree.destroy()
                                        itm = itmentvar.get()
                                        qty = int(qtyentvar.get())
                                        pr = float(prentvar.get())
                                        part = partentvar.get()
                                        q = f'insert into button{r + 1}_{c +
1} values("{itm}", {qty}, {pr}, "{part}")'
                                        buttoncur.execute(q)
                                        buttoncon.commit()
```

```
                                        value = [['Item', 'Qty', 'Price']]

                                        buttoncur.execute(f'select * from
button{r + 1}_{c + 1}')
                                        vals = buttoncur.fetchall()
                                        vl = [('Item', 'Qty', 'Price', 'ID')]
+ vals

                                        self.tree = CTkTable(shfr, column=4,
row=len(vals) + 1, values=vl)

                                        self.tree.grid(row=2, column=1,
columnspan=3, sticky='n', padx=20)

                                        adtp.destroy()

                                else:
                                    messagebox.showerror('error', 'All fields
not entered', icon='error')

                            # subbutt
                            sbbutton = ctk.CTkButton(adfr, text='Submit',
font=('Futura', 24), fg_color='#75b1a9',
                                                     hover_color='light
green', command=adtotree)
                            sbbutton.place(x=210, y=260, anchor=tk.CENTER)

                        adbt = ctk.CTkButton(shfr, text='Add', font=('Roboto
Medium', 24), fg_color='#75b1a9',
                                             hover_color='light green',
text_color='black', command=adbt)
                        adbt.grid(row=3, column=1, padx=10, pady=10)


                        #remove buttin for removing items
                        def rembt():
                            remtp = ctk.CTkToplevel(shelf)
                            remtp.title('Remove Item')
                            remtp.wm_transient(shelf)
                            remtp.resizable(False, False)
                            remtp.geometry('240x240')

                            # background
                            back = ctk.CTkImage(light_image=Image.open(
                                backgroundpath),
                                dark_image=Image.open(
                                    backgroundpath),
                                size=(1600, 800))
```

```
                            label = ctk.CTkLabel(remtp, image=back)
                            label.pack()

                            remfr = ctk.CTkFrame(remtp, corner_radius=15,
bg_color='#75b1a9', width=200, height=200)
                            remfr.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

                            remlab = ctk.CTkLabel(remfr, text='Name:',
font=('Roboto Medium', 24))
                            remlab.place(x=20, y=40)

                            rementvar = ctk.StringVar()
                            rement = ctk.CTkEntry(remfr,
textvariable=rementvar, width=70)
                            rement.place(x=100, y=40)


                            #function to remove from database and table
                            def remfromtree():
                                if rementvar.get() != '':
                                    itm = rementvar.get()
                                    buttoncur.execute(f'select Item from
button{r+1}_{c+1}')
                                    result = buttoncur.fetchall()
                                    if result == []:
                                        messagebox.showerror('error', 'Nothing
to remove', icon='error')

                                    else:

                                        q = f'delete from button{r + 1}_{c +
1} where Item = "{itm}"'
                                        buttoncur.execute(q)
                                        buttoncon.commit()

                                        remtp.destroy()
                                        self.tree.destroy()

                                        buttoncur.execute(f'select * from
button{r + 1}_{c + 1}')
                                        vals = buttoncur.fetchall()
                                        vl = [('Item', 'Qty', 'Price', 'ID')]
+ vals

                                        self.tree = CTkTable(shfr, column=4,
row=len(vals) + 1, values=vl)

                                        self.tree.grid(row=2, column=1,
columnspan=3, sticky='n', padx=20)
```

```
                                else:
                                    messagebox.showerror('error', 'Error Name
not entered', icon='error')

                            subbt = ctk.CTkButton(remfr, text='Submit',
font=('Roboto Medium', 18), fg_color='#75b1a9',
                                                    hover_color='light green',
text_color='black', command=remfromtree)
                            subbt.place(x=100, y=140, anchor=tk.CENTER)

                        rembt = ctk.CTkButton(shfr, text='Remove',
font=('Roboto Medium', 24), fg_color='#75b1a9',
                                                hover_color='light green',
text_color='black', command=rembt)
                        rembt.grid(row=3, column=3, padx=10, pady=10)


                        #button to modify items
                        def modbt():
                            modtp = ctk.CTkToplevel(shelf)
                            modtp.title('Modify Item')
                            modtp.wm_transient(shelf)
                            modtp.resizable(False, False)
                            modtp.geometry('360x280')

                            # background
                            back = ctk.CTkImage(light_image=Image.open(
                                backgroundpath),
                                dark_image=Image.open(
                                    backgroundpath),
                                size=(1600, 800))

                            label = ctk.CTkLabel(modtp, image=back)
                            label.pack()

                            modfr = ctk.CTkFrame(modtp, corner_radius=15,
bg_color='#75b1a9', width=320, height=240)
                            modfr.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

                            modidlab = ctk.CTkLabel(modfr, text='ID:',
font=('Roboto Medium', 24))
                            modidlab.place(x=20, y=20)

                            modidentvar = ctk.StringVar()
                            modident = ctk.CTkEntry(modfr,
textvariable=modidentvar, width=200)
                            modident.place(x=100, y=20)
```

```
                            moditmlab = ctk.CTkLabel(modfr, text='Item:',
font=('Roboto Medium', 24))
                            moditmlab.place(x=20, y=60)

                            moditmentvar = ctk.StringVar()
                            moditment = ctk.CTkEntry(modfr,
textvariable=moditmentvar, width=150)
                            moditment.place(x=100, y=60)

                            itmcheck = ctk.CTkCheckBox(modfr, text='')
                            itmcheck.place(x = 275, y= 60)

                            modqtylab = ctk.CTkLabel(modfr, text='Qty:',
font=('Roboto Medium', 24))
                            modqtylab.place(x=20, y=100)

                            modqtyentvar = ctk.StringVar()
                            modqtyent = ctk.CTkEntry(modfr,
textvariable=modqtyentvar, width=150)
                            modqtyent.place(x=100, y=100)

                            qtycheck = ctk.CTkCheckBox(modfr, text='')
                            qtycheck.place(x = 275, y= 100)

                            modpricelab = ctk.CTkLabel(modfr, text='Price:',
font=('Roboto Medium', 24))
                            modpricelab.place(x=20, y=140)

                            modpriceentvar = ctk.StringVar()
                            modpriceent = ctk.CTkEntry(modfr,
textvariable=modpriceentvar, width=150)
                            modpriceent.place(x=100, y=140)

                            pricecheck = ctk.CTkCheckBox(modfr, text='')
                            pricecheck.place(x = 275, y= 140)



                            #funtion to modify item in database and table
                            def modifytree():
                                if pricecheck.get() or qtycheck.get() or
itmcheck.get() == 1:

                                    if pricecheck.get() == 1:
                                        global qr
                                        qr = f'update button{r+1}_{c+1} set
Price = {int(modpriceentvar.get())} where ID = {int(modidentvar.get())}'
```

```
                                        if pricecheck.get() and qtycheck.get()
and itmcheck.get() == 1:
                                            qr = f'update button{r+1}_{c+1}
set Price = {int(modpriceentvar.get())}, Qty = {int(modqtyentvar.get())}, Item
= "{moditmentvar.get()}" where ID = {int(modidentvar.get())}'

                                        elif pricecheck.get() and
itmcheck.get() == 1:
                                            qr = f'update button{r+1}_{c+1}
set Price = {int(modpriceentvar.get())}, Item = "{moditmentvar.get()}" where
ID = {int(modidentvar.get())}'

                                        elif itmcheck.get() and qtycheck.get()
== 1:
                                            qr = f'update button{r+1}_{c+1}
set Item = "{moditmentvar.get()}", Qty = {int(modqtyentvar.get())} where ID =
{int(modidentvar.get())}'

                                        elif pricecheck.get() and
qtycheck.get() == 1:
                                            qr = f'update button{r+1}_{c+1}
set Price = {int(modpriceentvar.get())}, Qty = {int(modqtyentvar.get())} where
ID = {int(modidentvar.get())}'


                                    elif qtycheck.get() == 1:
                                        qr = f'update button{r+1}_{c+1} set
Qty = {int(modqtyentvar.get())} where ID = {int(modidentvar.get())}'

                                        if pricecheck.get() and qtycheck.get()
and itmcheck.get() == 1:
                                            qr =  f'update button{r+1}_{c+1}
set Price = {int(modpriceentvar.get())}, Qty = {int(modqtyentvar.get())}, Item
= "{moditmentvar.get()}" where ID = {int(modidentvar.get())}'

                                        elif pricecheck.get() and
itmcheck.get() == 1:
                                            qr = f'update button{r+1}_{c+1}
set Price = {int(modpriceentvar.get())}, Item = "{moditmentvar.get()}" where
ID = {int(modidentvar.get())}'

                                        elif itmcheck.get() and qtycheck.get()
== 1:
                                            qr = f'update button{r+1}_{c+1}
set Item = "{moditmentvar.get()}", Qty = {int(modqtyentvar.get())} where ID =
{int(modidentvar.get())}'

                                        elif pricecheck.get() and
qtycheck.get() == 1:
```

```
                                            qr = f'update button{r+1}_{c+1}
set Price = {int(modpriceentvar.get())}, Qty = {int(modqtyentvar.get())} where
ID = {int(modidentvar.get())}'


                                    elif itmcheck.get() == 1:
                                        qr = f'update button{r+1}_{c+1} set
Item = "{moditmentvar.get()}" where ID = {int(modidentvar.get())}'

                                        if pricecheck.get() and qtycheck.get()
and itmcheck.get() == 1:
                                            qr =  f'update button{r+1}_{c+1}
set Price = {int(modpriceentvar.get())}, Qty = {int(modqtyentvar.get())}, Item
= "{moditmentvar.get()}" where ID = {int(modidentvar.get())}'

                                        elif pricecheck.get() and
itmcheck.get() == 1:
                                            qr = f'update button{r+1}_{c+1}
set Price = {int(modpriceentvar.get())}, Item = "{moditmentvar.get()}" where
ID = {int(modidentvar.get())}'

                                        elif itmcheck.get() and qtycheck.get()
== 1:
                                            qr = f'update button{r+1}_{c+1}
set Item = "{moditmentvar.get()}", Qty = {int(modqtyentvar.get())} where ID =
{int(modidentvar.get())}'

                                        elif pricecheck.get() and
qtycheck.get() == 1:
                                            qr = f'update button{r+1}_{c+1}
set Price = {int(modpriceentvar.get())}, Qty = {int(modqtyentvar.get())} where
ID = {int(modidentvar.get())}'



                                query = qr
                                buttoncur.execute(query)
                                buttoncon.commit()

                                self.tree.destroy()

                                buttoncur.execute(f'select * from button{r +
1}_{c + 1}')
                                vals = buttoncur.fetchall()
                                vl = [('Item', 'Qty', 'Price', 'ID')] + vals

                                self.tree = CTkTable(shfr, column=4,
row=len(vals) + 1, values=vl)
```

```
                                self.tree.grid(row=2, column=1, columnspan=3,
sticky='n', padx=20)


                            subbt = ctk.CTkButton(modfr, text='Submit',
font=('Roboto Medium', 18), fg_color='#75b1a9',
                                                  hover_color='light green',
text_color='black',width=280, command=modifytree)
                            subbt.place(x=20, y=190)



                        modbtn = ctk.CTkButton(shfr, text='Modify',
font=('Roboto Medium', 24), fg_color='#75b1a9',
                                               hover_color='light green',
text_color='black', command=modbt)
                        modbtn.grid(row=3, column=2, padx = 10, pady=10)

                    #loops for creating a table in database for each shelf
                    for r in range(rowtab):

                        for c in range(columns):

                            buttoncur.execute('SELECT name FROM sqlite_master
WHERE type="table"')
                            tbs = buttoncur.fetchall()

                            button = ctk.CTkButton(gridfr, text=f"Shelf {r +
1}-{c + 1}", hover_color='light green',
                                                   fg_color='#75b1a9',
text_color='Black', font=('Roboto Medium', 20),
                                                   bg_color='#302c2c',
command=lambda r=r, c=c: frames(r, c))
                            button.grid(row=r, column=c, padx=15, pady=15,
sticky='nsew')
                            if (f'button{r + 1}_{c + 1}',) in tbs:
                                continue
                            else:
                                q = "create table button{}_{}(Item
varchar(30), Qty int, Price float, ID varchar(15) primary key)".format(r + 1,
c + 1)
                                buttoncur.execute(q)
                                buttoncon.commit()


                    #function for searching in the shelf selected
                    def search():
                        buttoncur.execute('SELECT name FROM sqlite_master
WHERE type="table"')
```

```
                        tables = buttoncur.fetchall()
                        found = False

                        for i in tables:
                            q = f'select * from {i[0]}'
                            buttoncur.execute(q)
                            record = buttoncur.fetchall()
                            for j in record:
                                if j[0] == searchentvar.get():
                                    butname = i[0]
                                    seashelf = f'Shelf {butname[6]}-
{butname[8]}'
                                    messagebox.showinfo('Search',
f'{searchentvar.get()} is present in {seashelf}',
                                                        icon='info')
                                    found = True
                                    break
                        if not found:
                            messagebox.showinfo('Search',
f'{searchentvar.get()} is not present in this shelf',
                                                icon='info')
                        searchent.delete(0, 'end')

                    # search
                    searchentvar = ctk.StringVar()
                    searchent = ctk.CTkEntry(gridfr, height=50,
textvariable=searchentvar)
                    searchent.grid(row=rowtab, column=0, columnspan=columns -
1, sticky='ew', padx=15, pady=15)

                    searchbutton = ctk.CTkButton(gridfr, text='Search',
font=('Roboto Medium', 20),
                                                 hover_color='light green',
                                                 fg_color='#75b1a9',
text_color='Black', bg_color='#302c2c',
                                                 command=search)
                    searchbutton.grid(row=rowtab, column=columns - 1, padx=15,
pady=15, sticky='nsew', )

                    buttonGrid.mainloop()

        # to check which button was pressed, using 'id' variable
        cur.execute('select * from userprofiles')
        usevals = cur.fetchall()
        global bt
        bt = []
        for j in usevals:
            if j[4] == user:
```

```
                bt = []
                cur.execute('select * from userprofiles where User =
"{}"'.format(user))
                rows = cur.fetchall()
                ypos = 20
                if len(rows) == 0:
                    pass
                else:
                    rowsno = False
                    noofbut = len(rows)

                    for i in range(len(rows)):
                        bt.append(
                            ctk.CTkButton(self, text=rows[i][1], width=700,
height=80, hover_color='light green',
                                          fg_color='#75b1a9',
                                          text_color='Black', font=('Roboto
Medium', 40),
                                          command=lambda id=rows[i][1]:
gridmake(id, user)))
                        bt[i].place(x=20, y=ypos)
                        ypos = ypos + 100
                break

        #function to add a shelf to the managing window
        def add():
            cur.execute('select * from userprofiles')
            rows = cur.fetchall()
            if len(rows) < 4:

                top = ctk.CTkToplevel(parent)
                top.wm_transient(parent)
                top.iconbitmap(iconpath)
                top.geometry('480x360')
                top.title('Save as')

                # background
                back = ctk.CTkImage(light_image=Image.open(
                    backgroundpath),
                    dark_image=Image.open(
                        backgroundpath),
                    size=(1600, 800))

                label = ctk.CTkLabel(top, image=back)
                label.pack()
```

```
                topframe = ctk.CTkFrame(top, corner_radius=15,
bg_color='#75b1a9', width=420, height=300)
                topframe.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

                # Name label and entry
                namlab = ctk.CTkLabel(topframe, text='Name:', font=('Futura',
18))
                namlab.place(x=30, y=20)

                namentvar = ctk.StringVar()
                nament = ctk.CTkEntry(topframe, textvariable=namentvar,
width=240)
                nament.place(x=150, y=20)

                # row column entry and label
                # row
                rowlab = ctk.CTkLabel(topframe, text='Row:', font=('Futura',
18))
                rowlab.place(x=30, y=60)

                rowentvar = ctk.StringVar()
                rowent = ctk.CTkEntry(topframe, textvariable=rowentvar,
width=240)
                rowent.place(x=150, y=60)

                # col
                columnlab = ctk.CTkLabel(topframe, text='Column:',
font=('Futura', 18))
                columnlab.place(x=30, y=100)

                colentvar = ctk.StringVar()
                colent = ctk.CTkEntry(topframe, textvariable=colentvar,
width=240)
                colent.place(x=150, y=100)


                ypo = 20

                def profadd(ypo):
                    if int(rowentvar.get()) == 1:
                        messagebox.showerror('error', 'Error, must have 2 or
more rows/columns', icon='error')

                    elif int(colentvar.get()) == 1:
                        messagebox.showerror('error', 'Error, must have 2 or
more rows/columns', icon='error')

                    else:
```

```
                        cur.execute('select * from userprofiles where User =
"{}"'.format(user))
                        rows = cur.fetchall()
                        if len(rows) == 0:
                            ypo = 20

                        else:
                            for i in rows:
                                ypo += 100

                        cur.execute('select * from userprofiles')
                        rows = cur.fetchall()

                        cur.execute('insert into userprofiles values("{}",
"{}", {}, {}, "{}")'.format(user+namentvar.get(),namentvar.get(),
rowentvar.get(), colentvar.get(), user))
                        con.commit()

                        tabledir = os.path.dirname(__file__)
                        tablepath = os.path.join(tabledir,
f'button_frame{namentvar.get()}{user}.db')
                        tablecon = mydb.connect(tablepath)
                        tablecur = tablecon.cursor()
                        top.destroy()

                        bt.append(ctk.CTkButton(self, text=namentvar.get(),
width=700, height=80, hover_color='light green',
                                                fg_color='#75b1a9',
text_color='Black', font=('Roboto Medium', 40),
                                                command=lambda
id=namentvar.get(): gridmake(id, user)))
                        bt[-1].place(x=20, y=ypo)
                        ypo = ypo + 100

                # submit
                subbuttontop = ctk.CTkButton(topframe, text='Submit',
font=('Futura', 24), fg_color='#75b1a9',
                                             hover_color='light
green',  text_color='Black', command=lambda ypo=ypo: profadd(ypo))
                subbuttontop.place(x=210, y=240, anchor=tk.CENTER)


            else:
                messagebox.showerror('Error', 'You have reached maximum number
of profiles', icon='error')


        #funcion to remove a shelf from the managing window
```

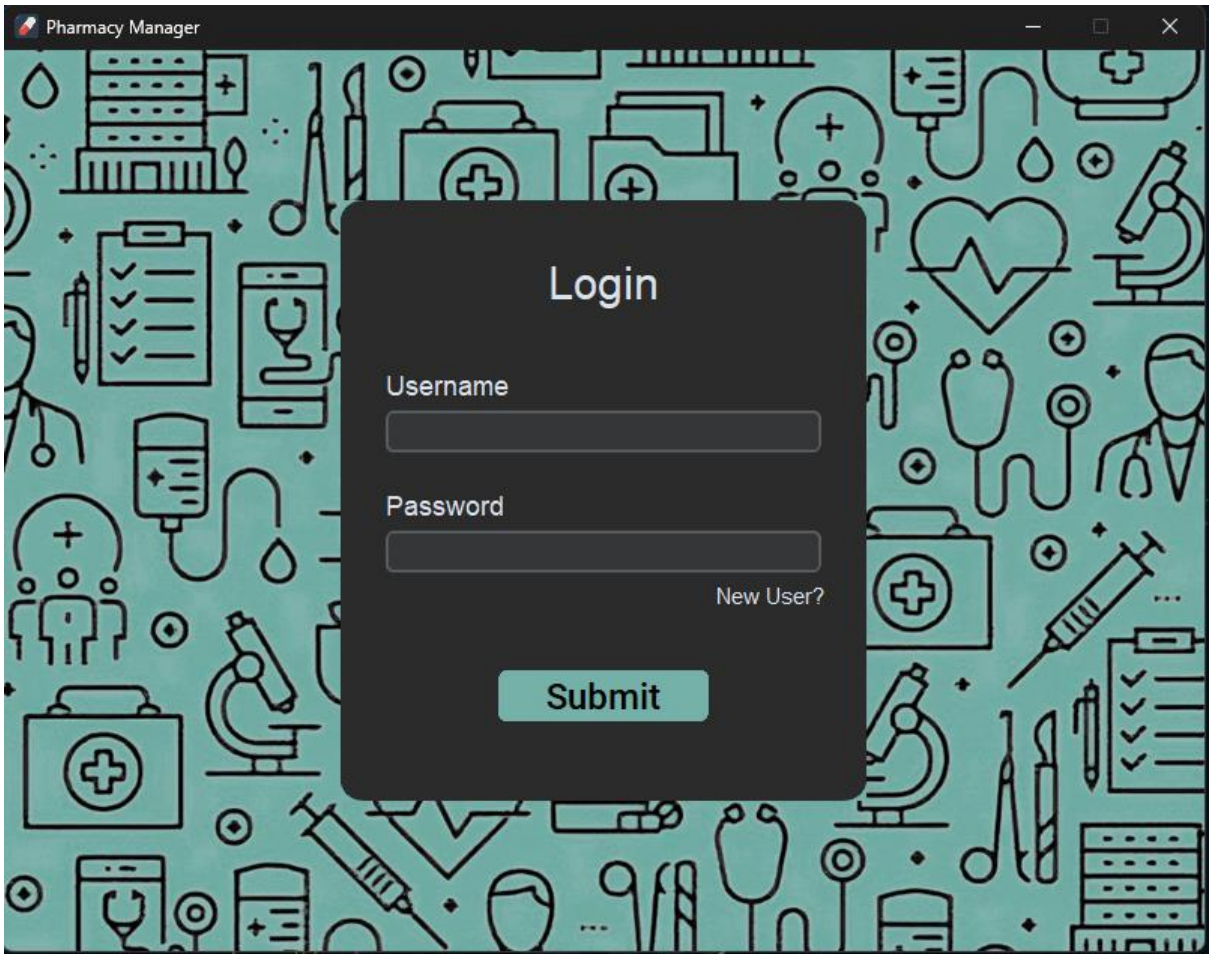
```
        def rembut():

            if len(bt) == 0:
                messagebox.showerror('Error', 'No profiles to remove',
icon='error')

            else:
                top = ctk.CTkToplevel(parent)
                top.wm_transient(parent)
                top.iconbitmap(iconpath)
                top.geometry('360x200')
                top.title('Remove')

                # background
                back = ctk.CTkImage(light_image=Image.open(
                    backgroundpath),
                    dark_image=Image.open(
                        backgroundpath),
                    size=(1600, 800))
                label = ctk.CTkLabel(top, image=back)
                label.pack()

                topframe = ctk.CTkFrame(top, corner_radius=15,
bg_color='#75b1a9', width=300, height=140)
                topframe.place(relx=0.5, rely=0.5, anchor=tk.CENTER)

                remlab = ctk.CTkLabel(topframe, text='Name:', font=('Futura',
18))
                remlab.place(x=30, y=20)

                def rem():
                    cur.execute('select * from userprofiles')
                    uservals = cur.fetchall()
                    for j in uservals:
                        if j[4] == user:
                            messagebox.askquestion('Delete', 'Are you sure you
want to delete?')
                            recrem = rementvar.get()
                            cur.execute('select * from userprofiles')
                            rows = cur.fetchall()
                            ypos = 20
                            for i in range(0, len(bt)):
                                bt[i].place_forget()

                            for i in range(0, len(bt)):
                                if bt[i].cget('text') == recrem:
                                    del bt[i]
                                    break
```

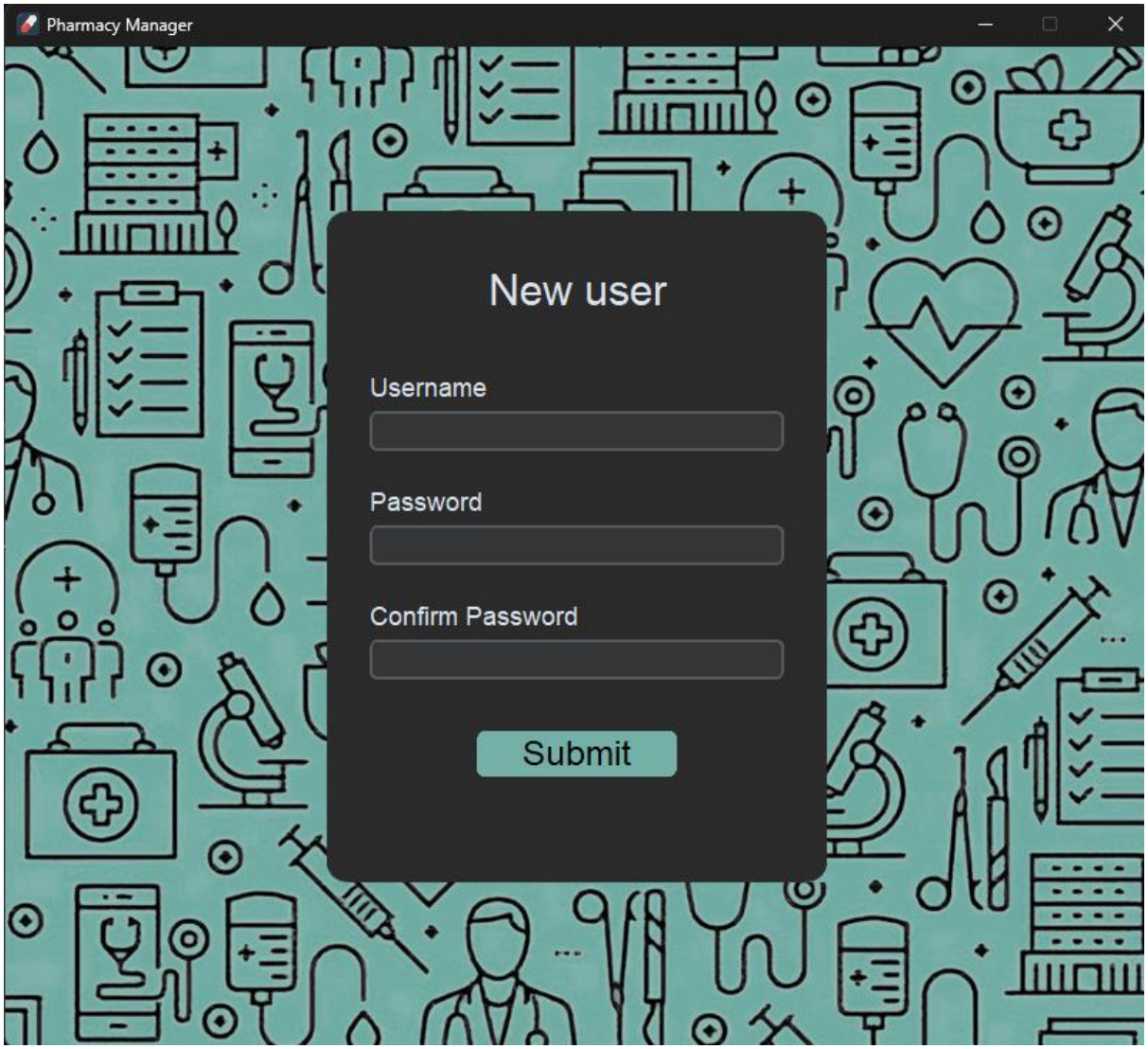
```
                            if len(bt) == 1:
                                bt[0].pack_forget()

                            q = 'delete from userprofiles where ID =
"{}"'.format(user+recrem)
                            cur.execute(q)
                            con.commit()
                            rement.delete(0, 'end')
                            top.destroy()
                            cur.execute('select * from userprofiles')
                            rows = cur.fetchall()

                            ypos = 20

                            if len(rows) == 0:
                                pass
                            else:
                                rowsno = False

                                for i in range(len(bt)):
                                    bt[i].place(x=20, y=ypos)
                                    ypos = ypos + 100


                            deltabdir = os.path.dirname(__file__)
                            tables = os.listdir(deltabdir)


                            if f'button_frame{recrem}{user}.db' in tables:
                                deldir = os.path.dirname(__file__)
                                delpath = os.path.join(deldir,
f'button_frame{recrem}{user}.db')
                                os.remove(delpath)

                            break

                rementvar = ctk.StringVar()
                rement = ctk.CTkEntry(topframe, textvariable=rementvar,
width=180)
                rement.place(x=100, y=20)

                rembutton = ctk.CTkButton(topframe, text='Remove',
fg_color='#75b1a9', hover_color='light green', text_color='Black',
                                          font=('Futura', 24), command=rem)
                rembutton.place(x=150, y=100, anchor=tk.CENTER)

        # add button
```

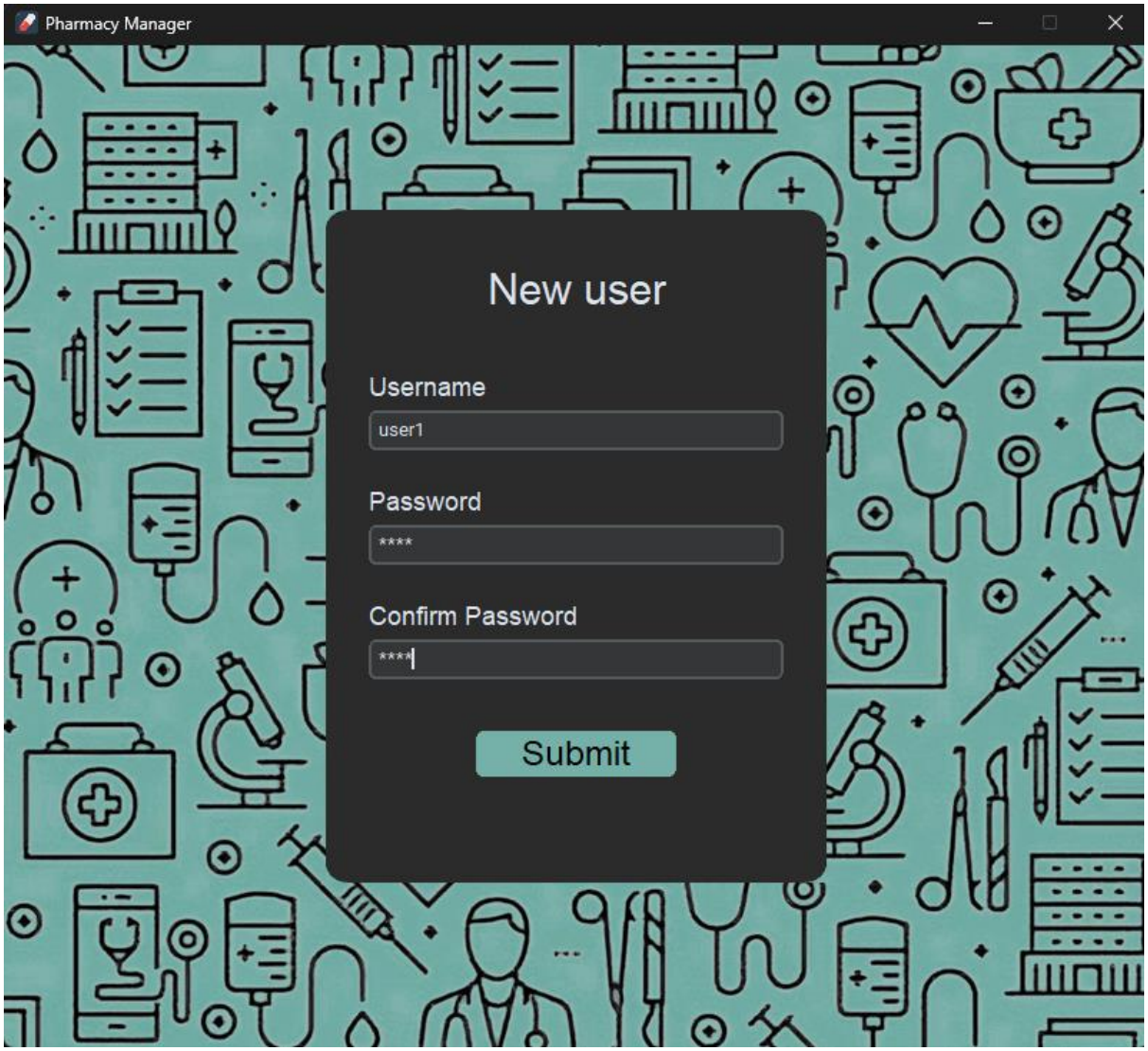
```
        addbutton = ctk.CTkButton(self, text='Add', font=('Futura', 24),
fg_color='#75b1a9', hover_color='light green', text_color='Black',
                                  command=add)
        addbutton.place(x=20, y=490)

        # remove button
        rembutton = ctk.CTkButton(self, text='Remove', font=('Futura', 24),
fg_color='#75b1a9', text_color='Black',
                                  hover_color='light green',
                                  command=rembut)
        rembutton.place(x=580, y=490)

        self.place(relx=0.5, rely=0.5, anchor=tk.CENTER)


#to check if user signed in
if signed is True:
    manage()
```

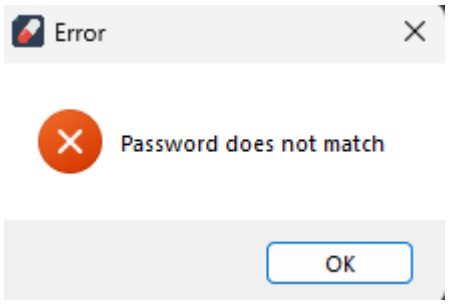
SAMPLE OUTPUT

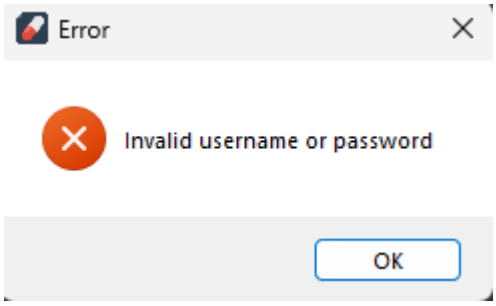
****

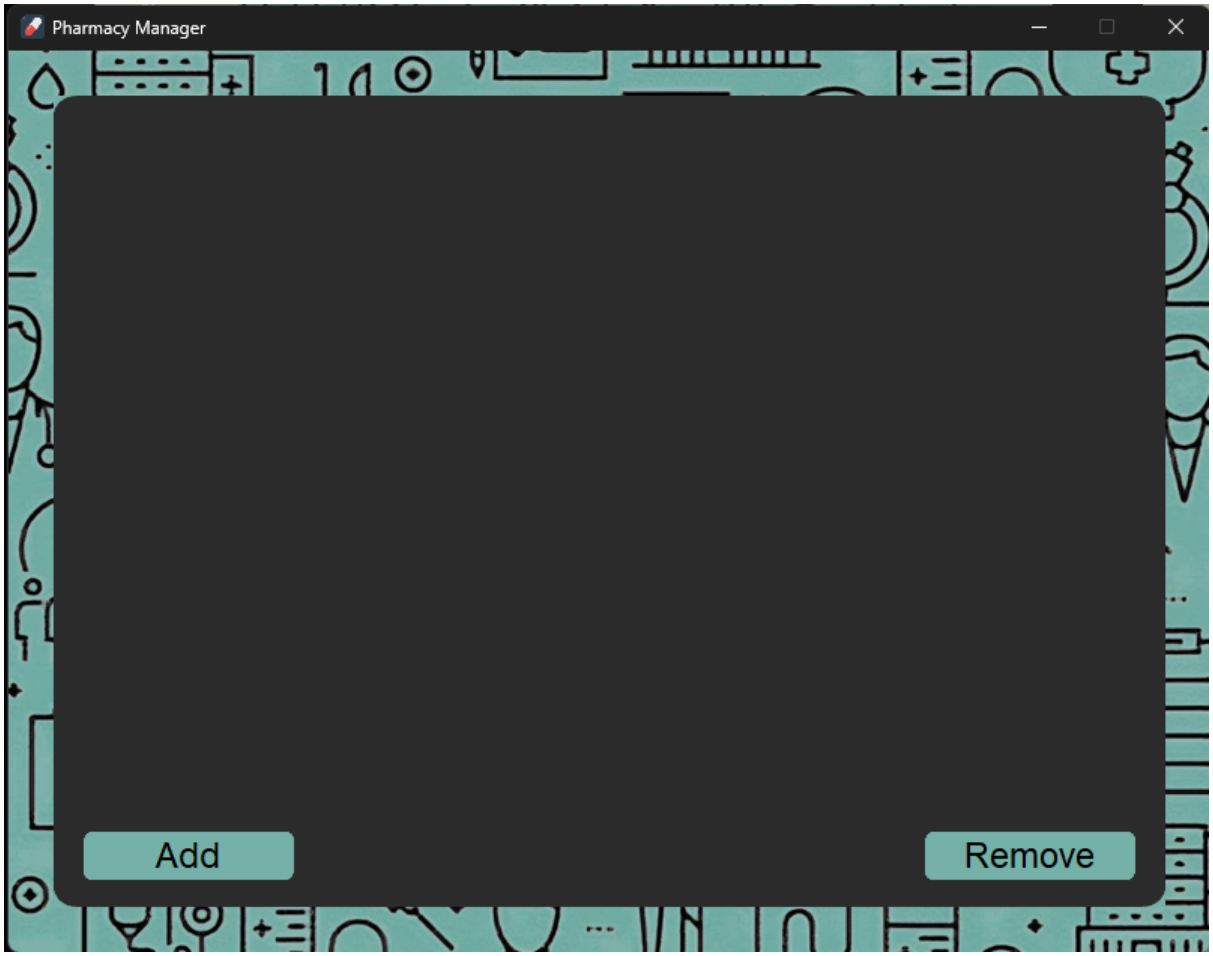


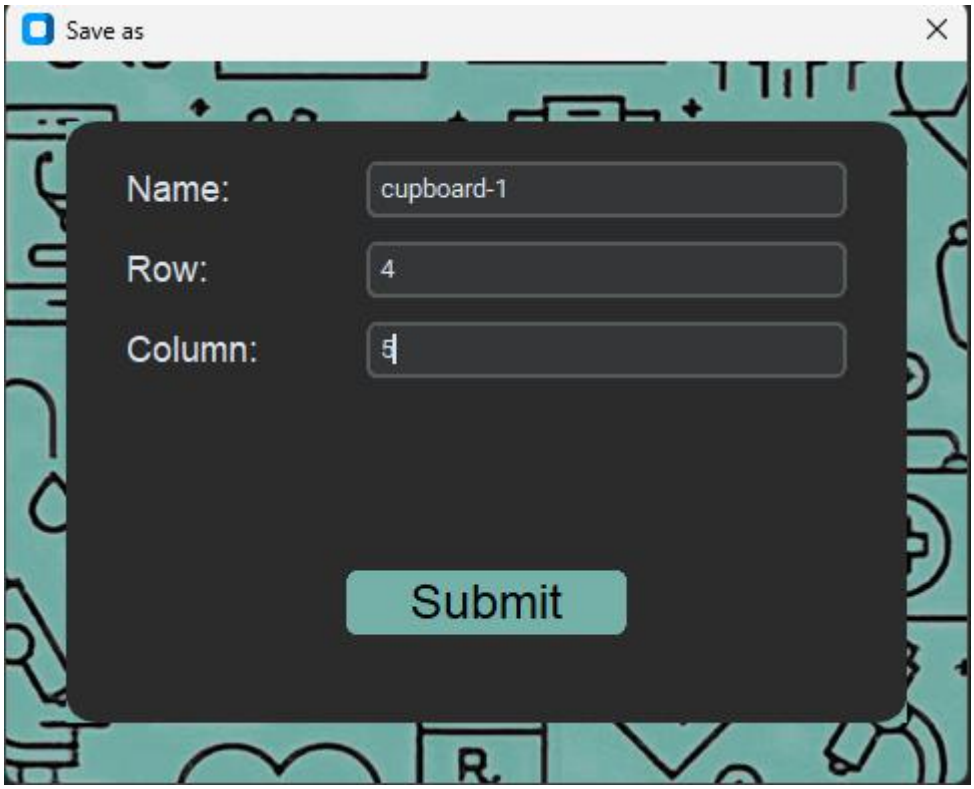

Pharmacy Manager
New user
Username
Password
Confirm Password
Submit

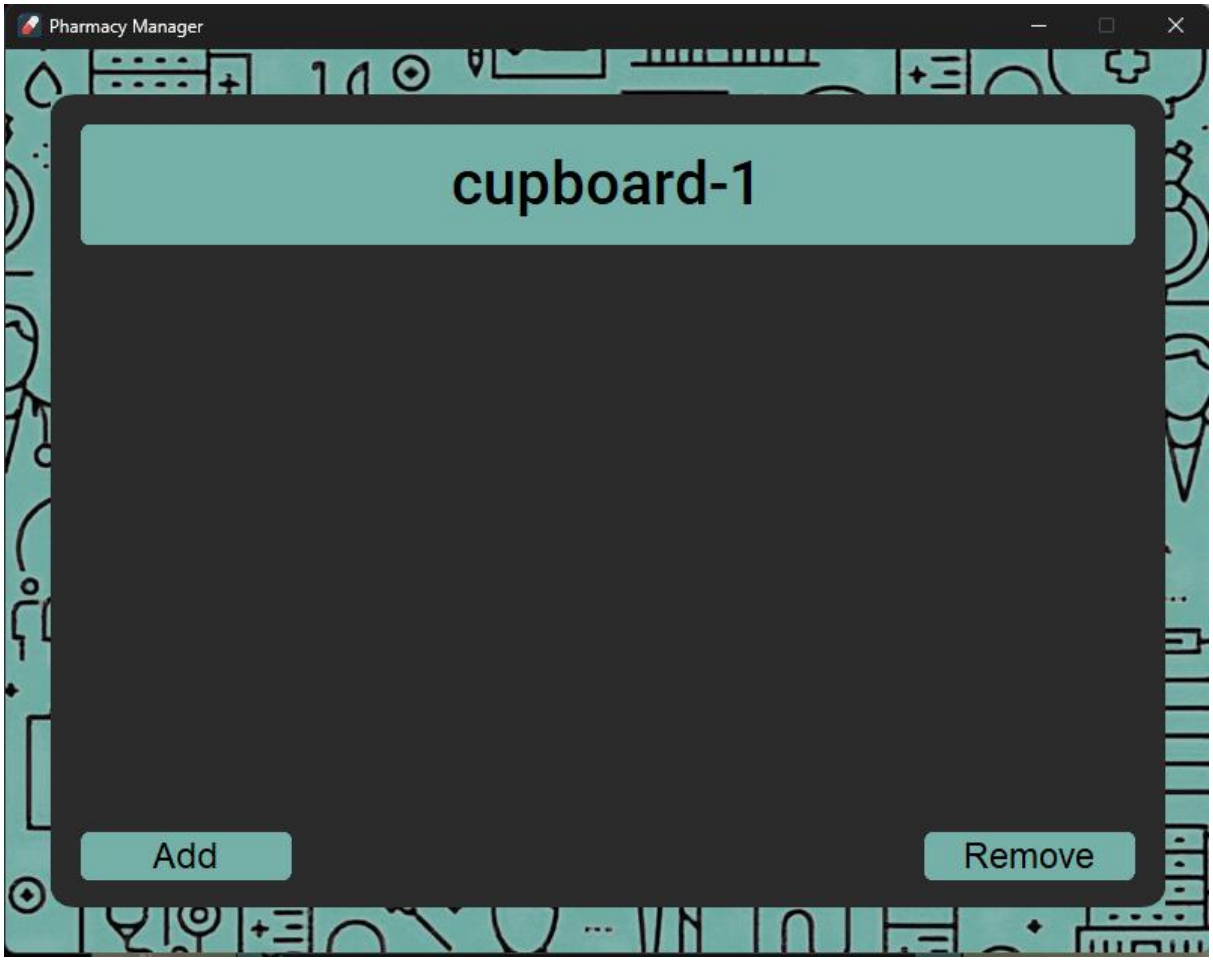

Pharmacy Manager
New user
Username
user1
Password
****
Confirm Password
****
Submit

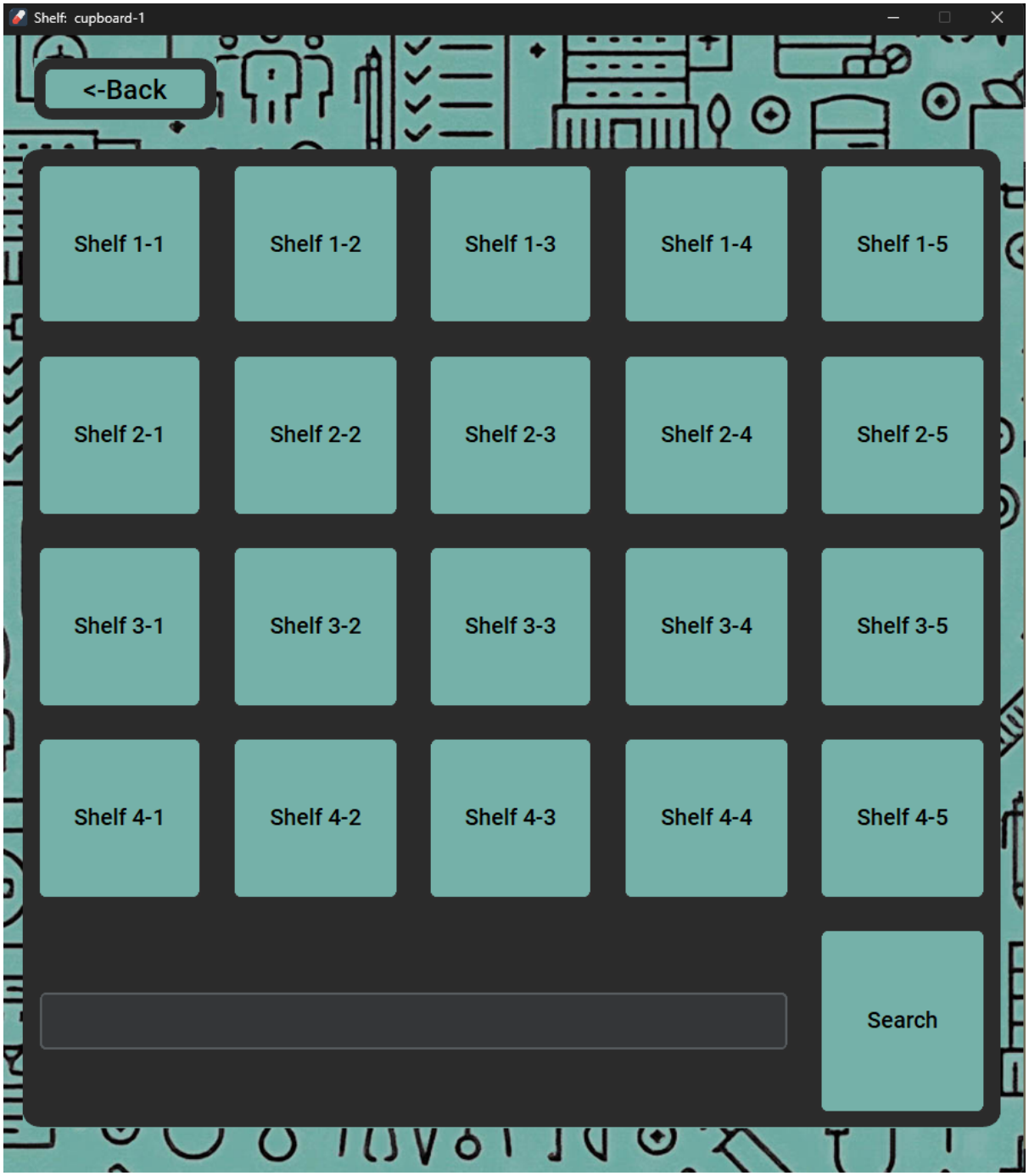

Error
Password does not match
OK

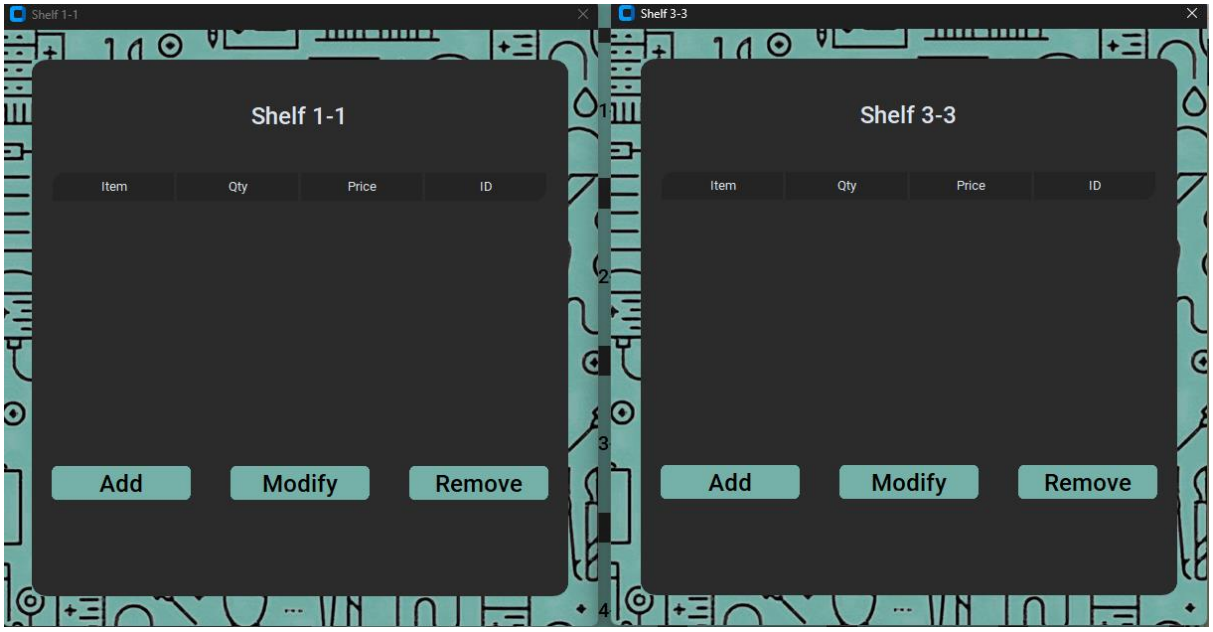

Error
Invalid username or password
OK

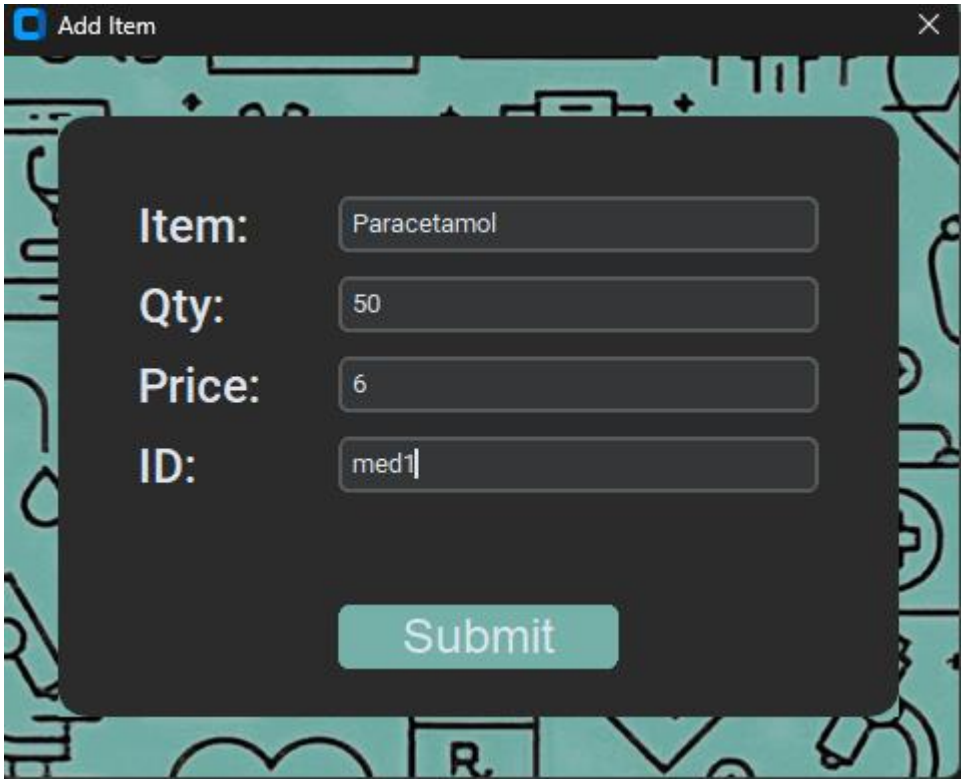

Pharmacy Manager
Add
Remove

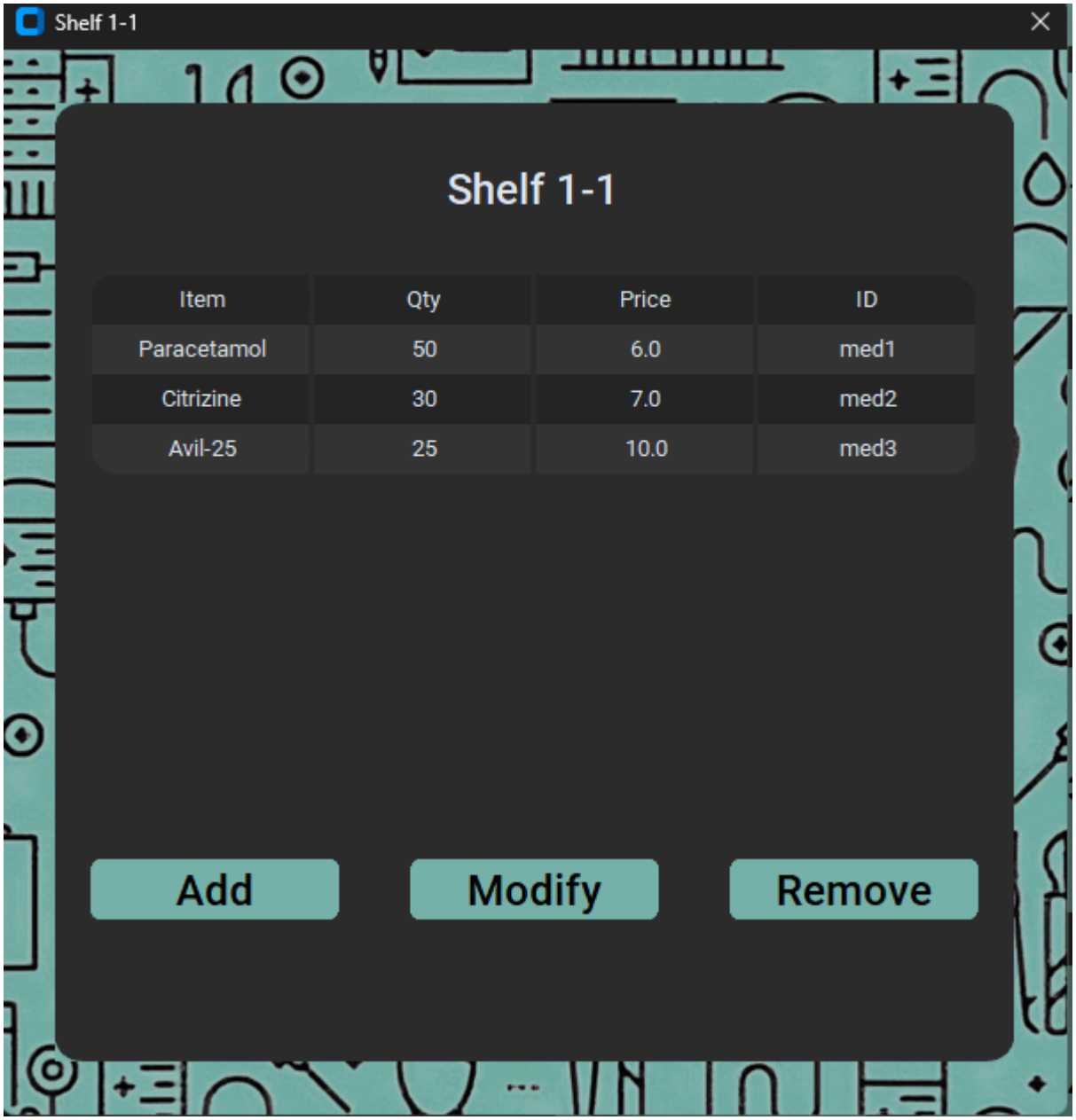

Save as
Name:
cupboard-1
Row:
4
Column:
5
Submit

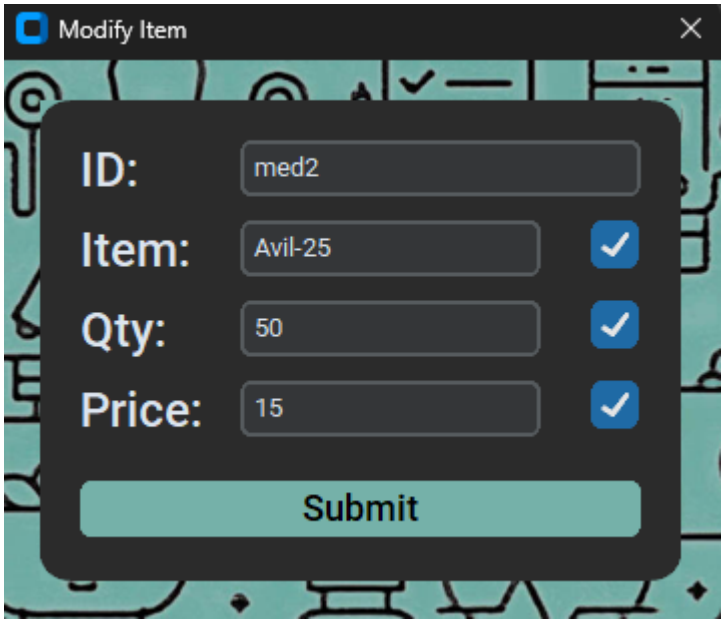

Pharmacy Manager
cupboard-1
Add
Remove

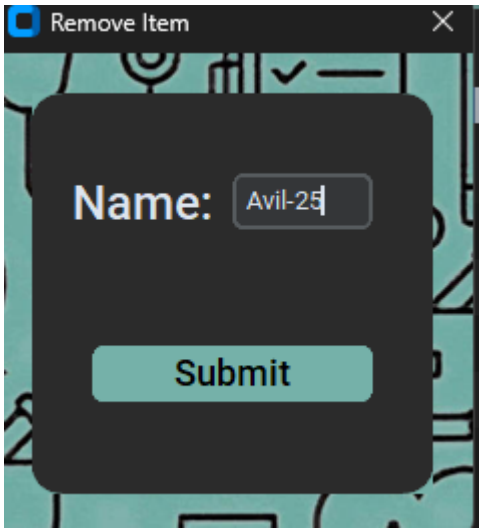

Shelf: cupboard-1
<-Back
Shelf 1-1
Shelf 1-2
Shelf 1-3
Shelf 1-4
Shelf 1-5
Shelf 2-1
Shelf 2-2
Shelf 2-3
Shelf 2-4
Shelf 2-5
Shelf 3-1
Shelf 3-2
Shelf 3-3
Shelf 3-4
Shelf 3-5
Shelf 4-1
Shelf 4-2
Shelf 4-3
Shelf 4-4
Shelf 4-5
Search

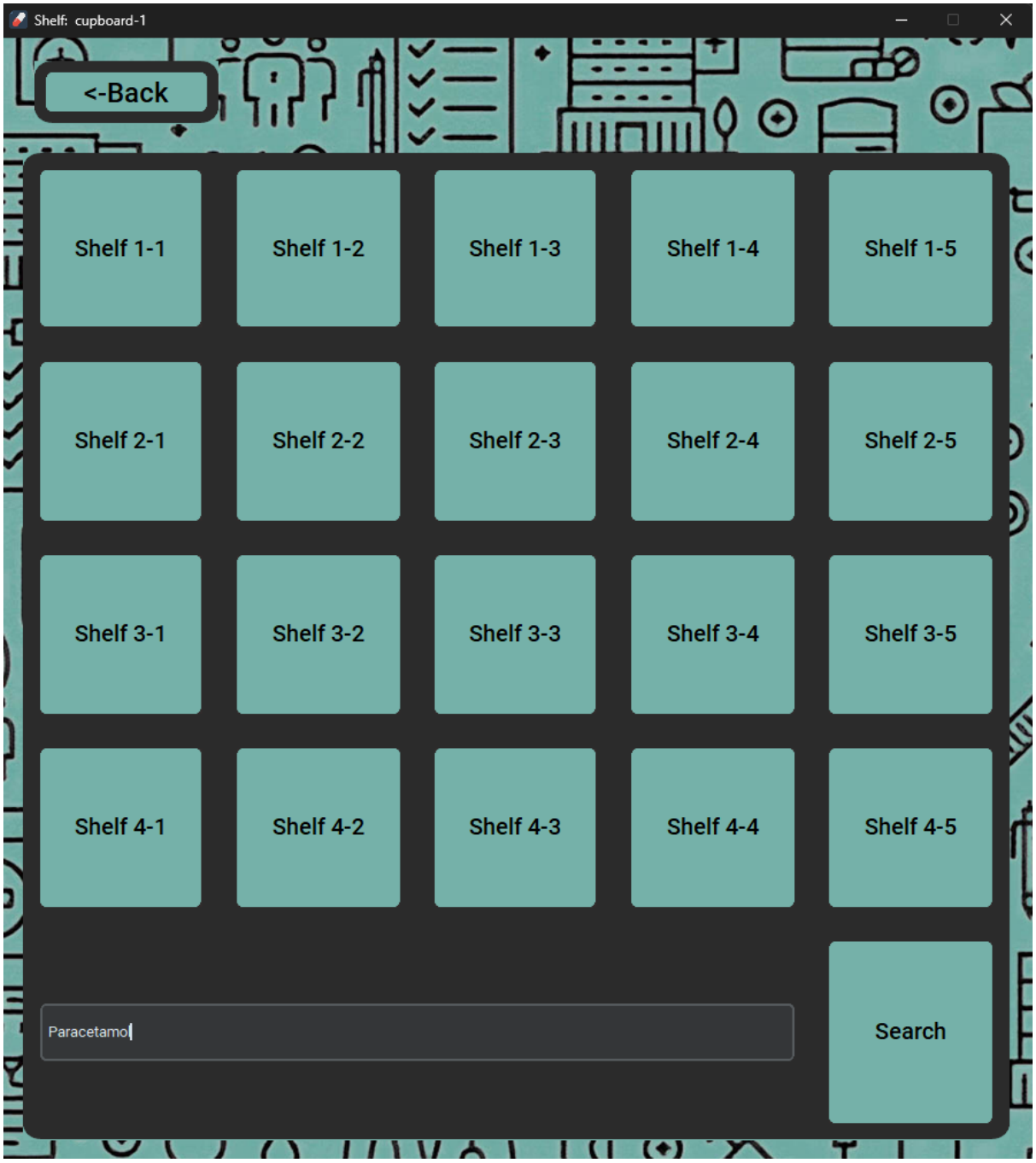

Shelf 1-1
Shelf 1-1
Item
Qty
Price
ID
Add
Modify
Remove
Shelf 3-3
Shelf 3-3
Item
Qty
Price
ID
Add
Modify
Remove

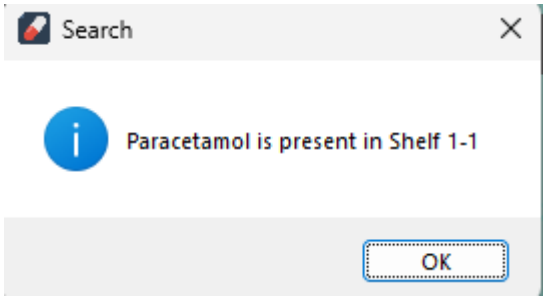

Add Item
Item:
Paracetamol
Qty:
50
Price:
6
ID:
med1
Submit

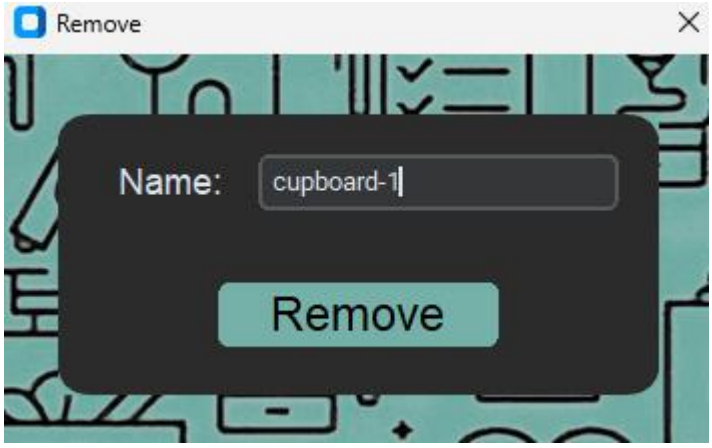

Shelf 1-1
Shelf 1-1
Item
Qty
Price
ID
Paracetamol
50
6.0
med1
Citrizine
30
7.0
med2
Avil-25
25
10.0
med3
Add
Modify
Remove

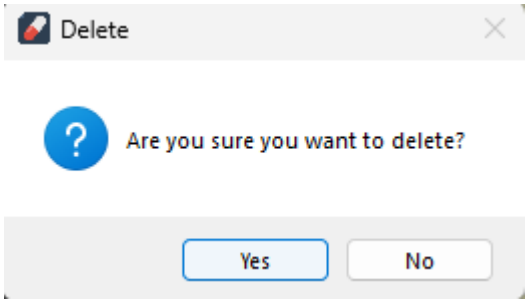

Modify Item
ID:
med2
Item:
Avil-25
Qty:
50
Price:
15
Submit

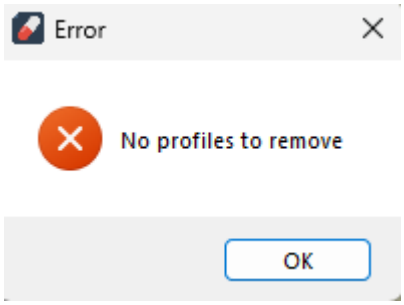

Remove Item
Name:
Avil-25
Submit


Shelf: cupboard-1
<-Back
Shelf 1-1
Shelf 1-2
Shelf 1-3
Shelf 1-4
Shelf 1-5
Shelf 2-1
Shelf 2-2
Shelf 2-3
Shelf 2-4
Shelf 2-5
Shelf 3-1
Shelf 3-2
Shelf 3-3
Shelf 3-4
Shelf 3-5
Shelf 4-1
Shelf 4-2
Shelf 4-3
Shelf 4-4
Shelf 4-5
Paracetamol
Search


Search
Paracetamol is present in Shelf 1-1
OK


Remove
Name:
cupboard-1
Remove


Delete
Are you sure you want to delete?
Yes
No


Error
No profiles to remove
OK

****

BIBLIOGRAPHY

Google

Class 12 computer science textbook

Stack overflow

Chat-GPT

CONCLUSION

We have successfully completed our Pharmacy Manger project , where we performed management(add, remove, update, search, display , etc) of the inventory of a pharmacy with the help of customtkinter as well as databases, reading and writing data using sqlite3.