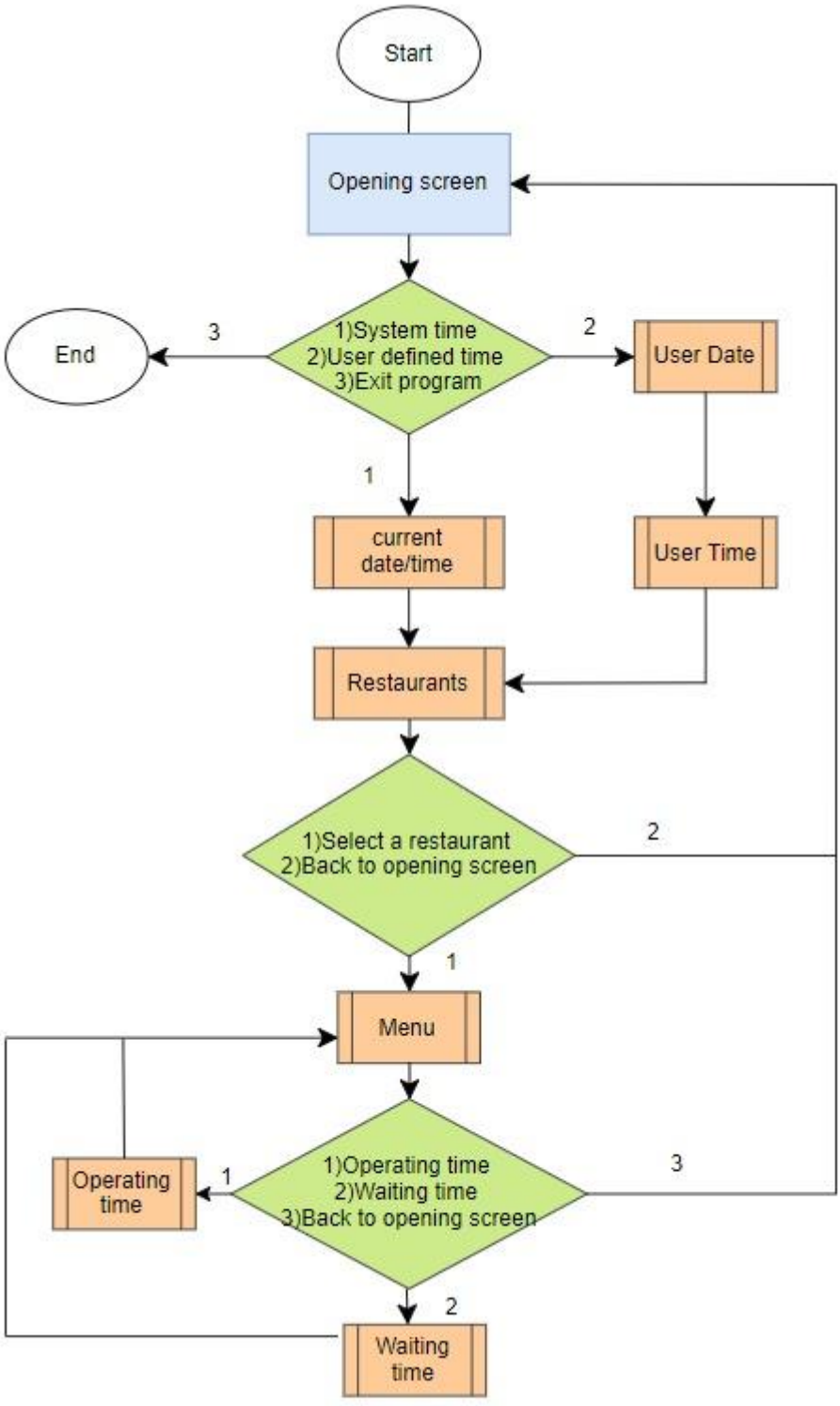
1003 Report Assessment

Table of Contents

Flowchart:


Start
Opening screen
1)System time
2)User defined time
3)Exit program
3
End
2
User Date
1
current date/time
User Time
Restaurants
1)Select a restaurant
2)Back to opening screen
2
1
Menu
1)Operating time
2)Waiting time
3)Back to opening screen
1
Operating time
3
2
Waiting time

Functions:

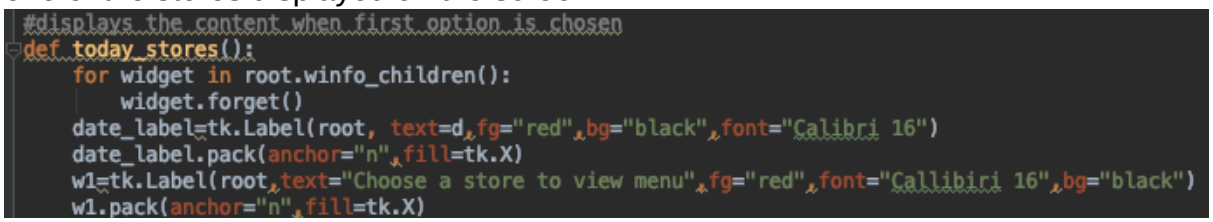
opening_screen():

Upon starting the program, the opening_screen() function is called which displays a message using a label. The screen contains a label and 3 buttons which are of different functions that will direct the user to different screens upon clicking.

```
def opening_screen():
    for widget in root.winfo_children():
        widget.forget()
    img=ImageTk.PhotoImage(Image.open("2.png"))
    open_message= "Welcome to North Spine Canteen!"
    open_message_label= tk.Label(root, font='Calibri 25', fg="red", bg="black", text=open_message)
    open_message_label.pack(anchor="n",fill=tk.X)
    option1_button=tk.Button(root, text="View today's stores", font="Chalkboard 15", fg="red", background='blue', command=today_stores)
    option1_button.pack(anchor="n",fill=tk.X)
    option2_button=tk.Button(root, text="View stores on other date",font="Chalkboard 15", fg="red",bg='blue',command=other_day_stores)
    option2_button.pack(anchor="n",fill=tk.X)
    image_label=tk.Label(root,image=img,bg="black")
    image_label.pack(anchor="n",fill="both",expand="yes")
    image_label.image=img
    option3_button=tk.Button(root, text="Exit",font="Chalkboard 15", fg="red", command=exit)
    option3_button.pack(side="bottom",fill=tk.X)
```

today_stores():

This function first displays the current date and time using datetime.datetime.now().time().replace(microsecond=0) to display the time (.replace(microsecond=0) is used to not show the microseconds.) and datetime.datetime.now().date() to express the date and prompts the user to choose one of the stores displayed on the screen.



```
#displays the content when first option is chosen
def today_stores():
    for widget in root.winfo_children():
        widget.forget()
    date_label=tk.Label(root, text=d,fg="red",bg="black",font="Calibri 16")
    date_label.pack(anchor="n",fill=tk.X)
    w1=tk.Label(root,text="Choose a store to view menu",fg="red",font="Callibiri 16",bg="black")
    w1.pack(anchor="n",fill=tk.X)
```

The stores displayed are according to the date and time. Using the elif conditions, the code checks what day it is and if it is a Saturday or Sunday, only the mcdonalds, long johns and kfc will be open. Else, it will display all the stores, as the north spine food court is closed on weekends. The program then goes through another elif check to determine if the store is open depending on its opening hours.

```
if(current_day=="Saturday" or current_day=="Sunday"):
    store1_button=tk.Button(root,text="1.McDonald's",fg="red",font="Chalkboard 15",command= lambda: menu1(current_time,current_date,current_day))
    store1_button.pack(anchor="n",fill=tk.X)
    store2_button=tk.Button(root,text="2.KFC",fg="red",font="Chalkboard 15",command=lambda: menu2(current_time,current_date,current_day))
    store2_button.pack(anchor="n",fill=tk.X)
    store5_button=tk.Button(root,text="3.Long John Silver's",fg="red",font="Chalkboard 15", command=lambda: menu5(current_time,current_date,current_day))
    store5_button.pack(anchor="n",fill=tk.X)
    img1 = ImageTk.PhotoImage(Image.open("mcd logo.jpg"))
    img1_label=tk.Label(root,image=img1,bg="black")
    img1_label.pack(anchor="n")
    img1_label.image=img1
    img2=ImageTk.PhotoImage(Image.open("kfc_logo.jpg"))
    img2_label=tk.Label(root,image=img2,bg="black")
    img2_label.pack(anchor="n")
    img2_label.image=img2
    img3=ImageTk.PhotoImage(Image.open("ljs_logo.jpg"))
    img3_label=tk.Label(root,image=img3,bg="black")
    img3_label.pack(anchor="n")
    img3_label.image=img3
```

Each of the stores have a button and calls the respective menu function when clicked on. When the user chooses one of the stores, it calls the menu(t,d,m) function of that store.

```
    else:
        store1_button = tk.Button(root, text="1.McDonald's",fg="red",font="Chalkboard 13",command=lambda: menu1(current_time, current_date, current_day))
        store1_button.pack(side="top", fill=tk.X)
        store2_button = tk.Button(root, text="2.KFC",fg="red",font="Chalkboard 13", command=lambda: menu2(current_time, current_date, current_day))
        store2_button.pack(side="top", fill=tk.X)
        store3_button = tk.Button(root, text="3.Chinese Delight",fg="red",font="Chalkboard 13",command=lambda: menu3(current_time, current_date, current_day))
        store3_button.pack(side="top", fill=tk.X)
        store4_button = tk.Button(root, text="4.Western Store",fg="red",font="Chalkboard 13", command=lambda: menu4(current_time, current_date, current_day))
        store4_button.pack(side="top", fill=tk.X)
        store5_button = tk.Button(root, text="5.Long John Silver's",fg="red",font="Chalkboard 13",command=lambda: menu5(current_time, current_date, current_day))
        store5_button.pack(side="top", fill=tk.X)
        img1 = ImageTk.PhotoImage(Image.open("s1.jpg"))
        img1_label = tk.Label(root, image=img1, bg="black")
        img1_label.pack(anchor="n")
        img1_label.image = img1
        img2 = ImageTk.PhotoImage(Image.open("s2.jpg"))
        img2_label = tk.Label(root, image=img2, bg="black")
        img2_label.pack(anchor="n")
        img2_label.image = img2
        img3 = ImageTk.PhotoImage(Image.open("s3.jpg"))
        img3_label = tk.Label(root, image=img3, bg="black")
        img3_label.pack(anchor="n")
        img3_label.image = img3
        img4=ImageTk.PhotoImage(Image.open("s4.jpg"))
        img4_label=tk.Label(root,image=img4,bg="black")
        img4_label.pack(anchor="n")
        img4_label.image=img4
    back_button=tk.Button(root,text="Back",fg="red",font="Chalkboard 15",command=opening_screen)
    back_button.pack(side="bottom",fill=tk.X)
```

date_func():

After importing tkcalender, we use the selection_get() method to get the date selected by the user. The choose time button then calls the time_func() function.

```
def date_func():
    def print_sel():
        for widget in root.winfo_children():
            widget.forget()
        l=tk.Label(root, text="Date chosen: " + str(ent.selection_get()),fg='red',bg="black",font="Calibri 22")
        l.pack(anchor="n",fill=tk.X)
        user_date=ent.selection_get()
        user_day=weekday[user_date.weekday()]
        di = ImageTk.PhotoImage(Image.open("t.jpg"))
        di_label = tk.Label(root, image=di, bg="black")
        di_label.pack(anchor="n",fill="both",expand="yes")
        di_label.image = di
        back_button = tk.Button(root, text="Back",font="Chalkboard 15",fg="red", command=opening_screen)
        back_button.pack(side="bottom", fill=tk.X)
        time_button=tk.Button(root, text="Choose time",font="Chalkboard 15",fg="red", command=lambda: time_func(user_date,user_day))
        time_button.pack(side="bottom",fill=tk.X)
    for widget in root.winfo_children():
        widget.forget()
    ent = tkcalendar.Calendar(root,font="Comic 20",selectmode="day",year=2019,month=10,day=28,foreground="red",bg="black", background="blue",selectforeground="red")
    ent.pack(side="top",expand=True,fill=tk.BOTH)
    ok_button=tk.Button(root,text="OK",fg="red",font="Chalkboard 16",highlightbackground="black", command=print_sel)
    ok_button.pack(fill=tk.X)
    back_button=tk.Button(root,text="Back",fg="red",font="Chalkboard 16",highlightbackground="black",command=opening_screen)
    back_button.pack(fill=tk.X)
```

time_func(d,m):

The window prompts the user to enter hours and minutes into the entry widget of tkinter. We check if the input is correct or not with exception handling. If there is an error, a message is displayed using a messagebox.showerror() and the user is prompted to enter again by calling the function again. Otherwise, it combines the 2 user inputs and converts them into a singular time value using datetime module. Then, the menu(t,d,m) function is called.

```
#to get time input from user
def time_func(d, m):
    for widget in root.winfo_children():
        widget.forget()
    l1 = tk.Label(root, text="Enter hours:", fg="red", bg="black", font="Calibri 20")
    l1.pack(anchor="nw")
    hr_entry = tk.Entry(root)
    hr_entry.pack(anchor="nw")
    l2 = tk.Label(root, text="Enter minutes:", fg="red", bg="black", font="Calibri 20")
    l2.pack(anchor="nw")
    min_entry = tk.Entry(root)
    min_entry.pack(anchor="nw")
#checks if input entered is correct or not
    def check_inp():
            try:
                t = hr_entry.get() + min_entry.get()
                x = int(hr_entry.get())
                y = int(min_entry.get())
                if not (0 <= (x) <= 23 and 0 <= (y) <= 60):
                    messagebox.showerror("Error", "Wrong input.Please try again.")
                    time_func(d, m)
                else:
                    t1 = datetime.datetime.strptime(t, '%H%M').time()
                    menus(t1,d,m)
            except:
                messagebox.showerror("Error", "Wrong input.Please try again.")
                time_func(d, m)
    back_button = tk.Button(root, text="Back", fg="red", font="Chalkboard 15", command=opening_screen)
    back_button.pack(side="bottom", fill=tk.X)
    continue_button = tk.Button(root, text="Continue",fg="red",font="Chalkboard 15", command=check_inp)
    continue_button.pack(side="bottom",fill=tk.X)
    ti = ImageTk.PhotoImage(Image.open("t1.jpg"))
    ti_label = tk.Label(root, image=ti, bg="black")
    ti_label.pack(side="bottom")
    ti_label.image = ti
```

menu(t,d,m): (Note: Each store has their own menu() function)

When the respective store button is clicked, the time, date and day are passed on to the menu(t,d,m) function. The parameters t, d and m store the time, date and day respectively.

First, the date and time given will be displayed and depending on the day and time, the menu is displayed from the dictionary using the label function.

```
#stores all menus
Mcdonald_Menu = {"Breakfast": ["1.Big Breakfast: $8.50", "2.Chicken Muffin: $5.60",
3.Sausage McMuffin: $6.00", "4.Filet-O-Fish: $5.50"], "Afternoon": ["1.Double Cheeseburger Meal: $6.20", "2.Four McWing Meal: $6.80",
3.McChicken: $5.00", "4.Big Mac: $8.20",  "5.Filet-O-Fish: $5.50"]}

KFC_Menu = {"Breakfast": ["1.Riser Burger: $5.20", "2.Porridge: $4.50",
3.American Twister: $7.70", "4.Brekkie Burger: $5.70"], "Afternoon": ["1.Two Piece Chicken Meal: $6.55", "2.Three Piece Chicken Meal: $8.55",
3.Zinger Meal: $6.00", "4.Snack N Share: 12.95",  "5.Shroom Burger Meal: $5.10"]}

Long_John_Silvers = {"Breakfast": ["1.Texas Chicken Double Toast: $6.30", "2.Chicken Sausage Double Toast: $5.90",
3.Double Turkey and Bacon Double Toast: $5.90", "4.French Toast Platter : $9.50"], "Afternoon": ["1.Fish and Chicken: $6.50", "2.Fish and Two piece Shrimp: $6.90",
3.Three Piece Fish: $7.50", "4.Three Piece Chicken: $6.50"]}
```

Elif blocks are used to determine if the store is closed at that time, and if it is, the message “sorry we are closed now” is shown. There is also a button in the menu that allows the user to check the operating hours of that store by calling the op(i) function.

If the store is open, the user can find out the average waiting time of that store by clicking on the waiting time button, calling the average_waiting_time(t,i) function.

```
#displays menus according to date and time chosen in the second feature
def menus(t,d,m):
    for widget in root.winfo_children():
        widget.forget()
    date_time=("Your choice: "+ str(d)+" "+str(m)+" "+str(t))
    d_label=tk.Label(root,text=date_time,fg="red",font="Calibri 15",bg="black")
    d_label.pack(anchor="n",fill=tk.X)
    w1 = tk.Label(root, text="Choose a store to view menu", fg="red", font="Callibiri 15", bg="black")
    w1.pack(anchor="n", fill=tk.X)
    if(m=='Sunday' or m=='Saturday'):
        m1=tk.Button(root, text="1.McDonald's",fg="red",font="Chalkboard 15",command=lambda: menu1(t,d,m))
        m1.pack(anchor="n",fill=tk.X)
        m2=tk.Button(root, text="2.KFC",fg="red",font="Chalkboard 15", command=lambda: menu2(t,d,m))
        m2.pack(anchor="n",fill=tk.X)
        m3=tk.Button(root,text="3.Long John Silver's",fg="red",font="Chalkboard 15", command=lambda: menu5(t,d,m))
        m3.pack(anchor="n",fill=tk.X)
        img1 = ImageTk.PhotoImage(Image.open("mcd logo.jpg"))
        img1_label = tk.Label(root, image=img1, bg="black")
        img1_label.pack(anchor="n")
        img1_label.image = img1
        img2 = ImageTk.PhotoImage(Image.open("kfc_logo.jpg"))
        img2_label = tk.Label(root, image=img2, bg="black")
        img2_label.pack(anchor="n")
        img2_label.image = img2
        img3 = ImageTk.PhotoImage(Image.open("ljs_logo.jpg"))
        img3_label = tk.Label(root, image=img3, bg="black")
        img3_label.pack(anchor="n")
        img3_label.image = img3
```

op(i):

i is the value of the store number. Using elif, we choose which files to show based off the store numbers. Using file handling, we display the operating hours by showing it in the message box by using messagebox.showinfo().

```
def op(i):
    if(i==1 or i==2 or i==5):
        f=open("Mcdonald's.txt","r")
        messagebox.showinfo("Operating Hours", f.read())
    else:
        f = open("Western Store.txt", "r")
        messagebox.showinfo("Operating Hours", f.read())
```

average_waiting_time(t,i):

t is the variable which represents time and i represents the stall number.

We will ask from the user the number of people waiting in line by using simpledialog.askinteger() method. We then use error handling to check the input. If correct, we calculate the waiting time. And the waiting time is extracted from a dictionary wt_store_dict. After going through elif functions which filters the time, random.randint is used to select a random value from the list inside the dictionary and the output is displayed using messagebox.showinfo().

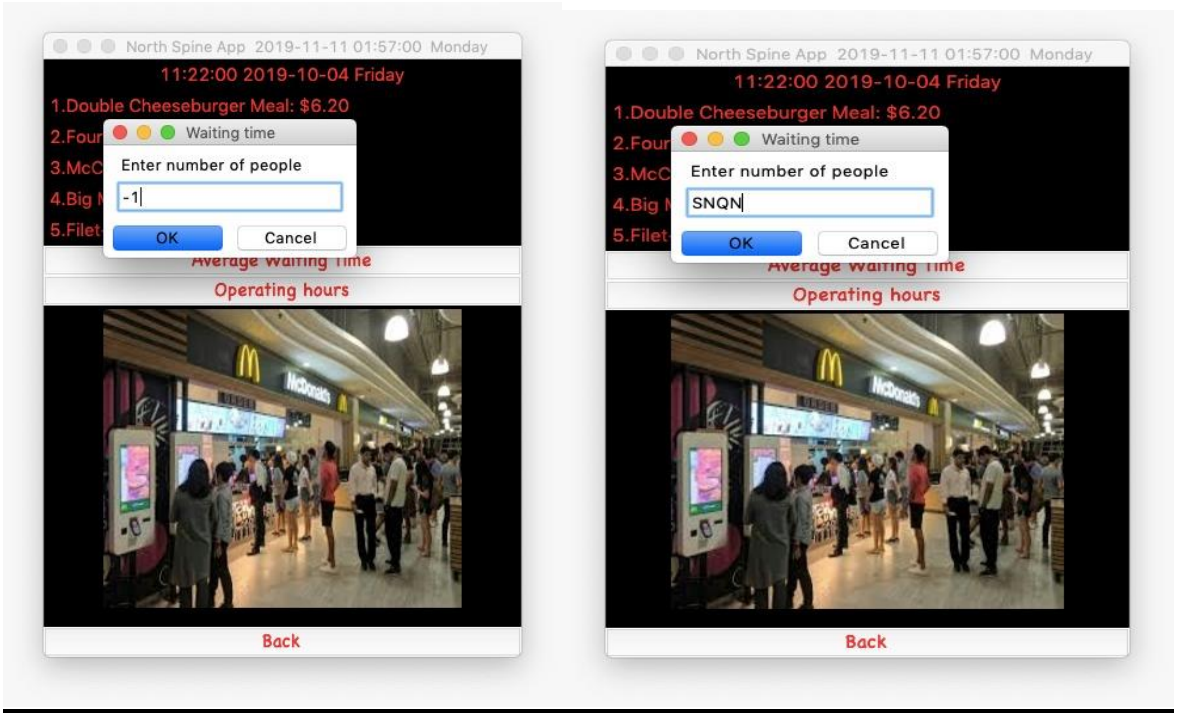
```
def avg_waiting_time(t,i):
    people=simpledialog.askinteger("Waiting time", "Enter
    q=random.randint(0,6)
    waiting_time=0
    if(people!= None):
        if (time_8am <= t < time_11am):
            waiting_time=people*wt_store_dict[i][0][q]
        elif (time_11am <= t < time_17pm):
            waiting_time=people*wt_store_dict[i][1][q]
        elif (time_17pm <= t < time_22pm):
            waiting_time=people*wt_store_dict[i][2][q]
        output=convert(waiting_time)
        messagebox.showinfo("Waiting time", output)
```

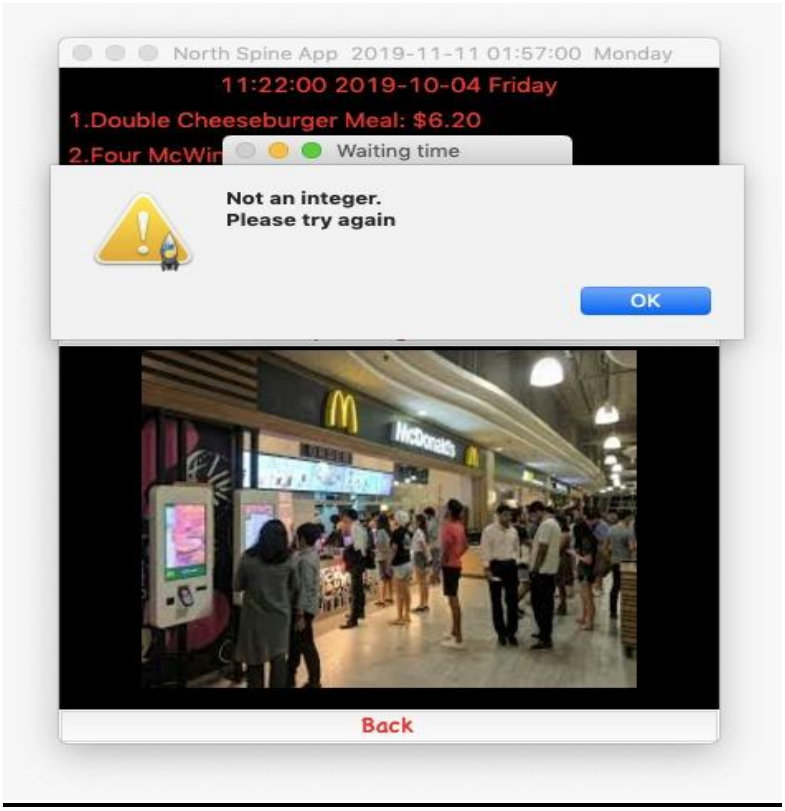
Program testing:

For this project, we have 3 inputs. But, there is only a need for the handling of the time input and the average waiting time input.

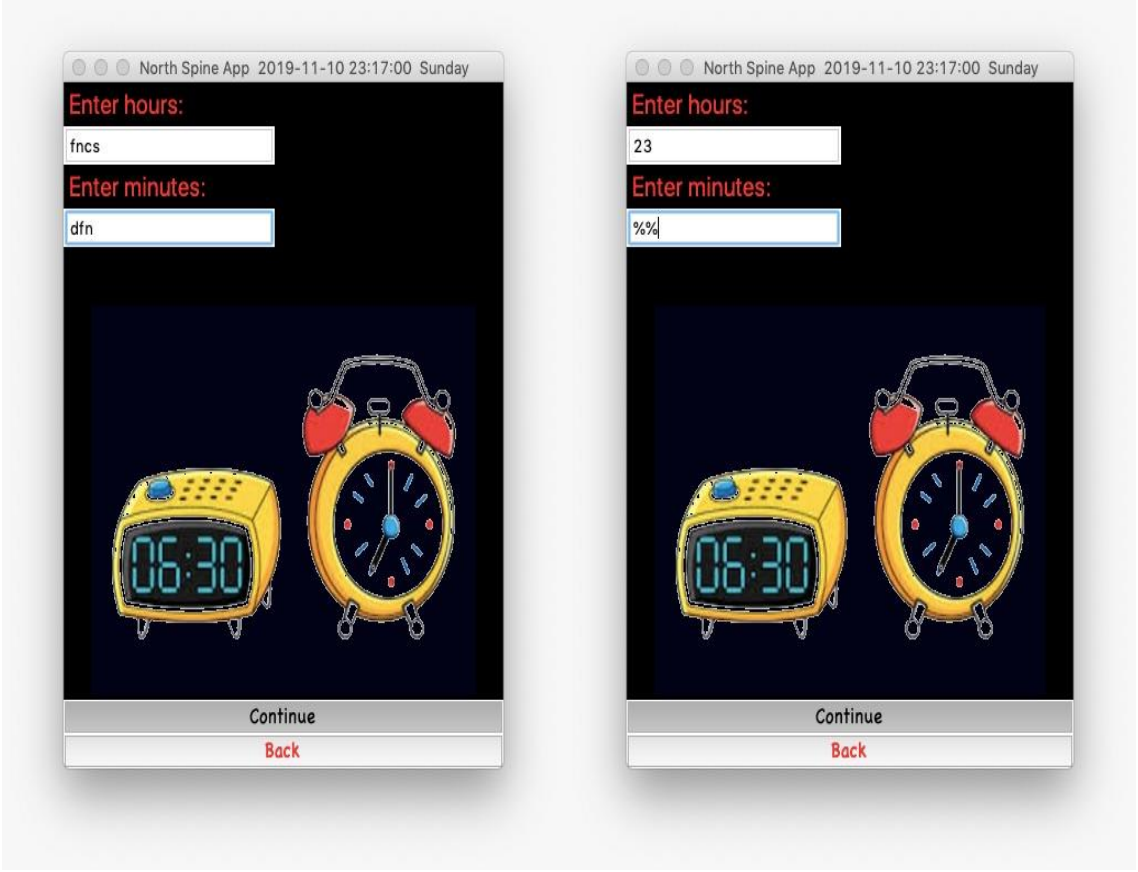
In average waiting time, we cannot accept negative or non-integer values as there cannot be a negative or a decimal number of people. Thus , simpledialogue.askinteger(), will reject any non-numeric, negative integer or non-integer value.

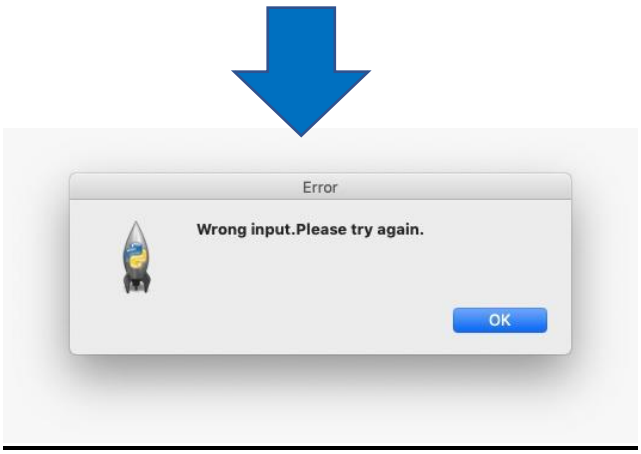
```
messagebox.showerror("Error","Please try again.")
```


North Spine App 2019-11-11 01:57:00 Monday
11:22:00 2019-10-04 Friday
1.Double Cheeseburger Meal: $6.20
Waiting time
Enter number of people
-1
OK
Cancel
Operating hours
Back
North Spine App 2019-11-11 01:57:00 Monday
11:22:00 2019-10-04 Friday
1.Double Cheeseburger Meal: $6.20
Waiting time
Enter number of people
SNQN
OK
Cancel
Operating hours
Back


North Spine App 2019-11-11 01:57:00 Monday
11:22:00 2019-10-04 Friday
1.Double Cheeseburger Meal: $6.20
Waiting time
Not an integer.
Please try again
OK
Back

For time, the conditions are slightly different. Thus, the inputs for time are checked to be between 0-23 for hours and 0-59 for minutes. If the input is non-numeric, or if the input is a number not within the range, using try and except and if-else respectively, an error message is shown and the program is sent back to time_func(d,m).





Reflection

The group has learned more about how to make use of GUI code as well as gained experience in logical thinking for programming. Like how the approach our group has used to tackle the programming of the application is by creating callable functions, facilitating easier use of code and being more efficient in our programming as lesser lines of code needs to be used.

Additionally, as none of the group members had ever handled a GUI before, tkinter was a completely new experience for the all of us, making us all gain more knowledge on how to use tkinter and python at the end of this project.

Another issue that we had faced was the usage of widgets in this project. As they are an essential part of the project, we had used them multiple times throughout our code. However, there are multiple widgets and different attributes and syntax specific to each. As such, we took awhile to learn the different widgets and refine our knowledge of them before starting to code. Some examples include how we take input from the user; entry widget for time, simpledialog for average waiting time and widget.forget() to clear the previous page.

We learnt how to display data on tkinter and aligning it by making use of labels and pack(anchor='direction') respectively. To display any error message, messagebox is used.

Datetime module is used to get the current date and time and convert string data to date and time.

We learned how to insert images in a program. Images are stored in labels. pack() function is used to align the images and anchor attribute. A parameter is given and by using s for south, n for north, etc, the desired direction is chosen and aligned to the direction accordingly.

Another point we learned from this project is management of work between the members. For example, as one of the members is new to programming, more of the work was given to the experienced members. However, he was able to learn much more about python and GUI thanks to their experience. Meanwhile, as he is more proficient in writing reports and creating presentations, he focused on those aspects and compiled what was experienced and learned. Such division of labor was good for the team as each member can play to their strengths and improve the project.

One portion we can improve on is in-cooperating google maps to use gps to give directions to the user. A search function can also be implemented for the user to search out stores.

Contributions:

- Khush

-Created GUI using tkinter

-Connected GUI code with menu dictionaries

-Added and altered different features according to GUI

- Samuel

-Created the lists of dictionaries for the menus

-Including odd and even day menu logic

-Provided current date and time using datetime

- Joshua

-Provided logics for average waiting time

-In charge of compiling the report

-Preparing for the presentation